BSIL: A Brain Storm-based Framework for Imbalanced Text Classification

Jiachen Tian, Shizhan Chen, Xiaowang Zhang(✉), and Zhiyong Feng

College of Intelligence and Computing, Tianjin University, Tianjin, China
Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin, China
{jiachen6677,shizhan,xiaowangzhang,zyfeng}@tju.edu.cn

Abstract. All neural networks are not always effective in processing imbalanced datasets when dealing with text classification due to most of them designed under a balanced assumption. In this paper, we present a novel framework named BSIL to improve the capability of neural networks in imbalanced text classification built on brain storm optimization (BSO). With our framework BSIL, the simulation of human brainstorming process of BSO can sample imbalanced datasets in a reasonable way. Firstly, we present an approach to generate multiple relatively balanced subsets of an imbalanced dataset by applying scrambling segmentation and global random sampling in BSIL. Secondly, we introduce a parallel method to train a classifier for a subset efficiently. Finally, we propose a decision-making layer to accept "suggestions" of all classifiers in order to achieve the most reliable prediction result. The experimental results show that BSIL associated with CNN, RNN and Self-attention model can performs better than those models in imbalanced text classification.

1 Introduction

A neural networks (NN)-based model often needs a large number of datasets for training when dealing with text classification task with the assumption that the distribution of datasets is balanced and the error cost is equal [10]. However, real-world data is often heavily skewed and, as a result, standard classification algorithms tend to ignore the minority class and overwhelm the majority one [14]. The imbalance of datasets mainly determined as one of essential characteristics of datasets cannot be avoided in our real life such as natural disasters, network intrusion, cancer detection, etc. [7]. Although the sample size of the minority class is much smaller than that of the majority one, they usually carry more important information not be ignored in processing. It becomes interesting and important to improve the capability of existing models to correctly classify samples of the minority class [12].

In this paper, inspired from the idea of brain storm optimization (BSO) algorithm [5], we design a framework for dealing with the imbalanced learning problem, named *brain storm imbalanced learning* (BSIL). This algorithm can simulate our brain well, where the left and right hemispheres collaboratively

work together on a job [4]. The purpose of BSIL is to utilize a distributed extensible framework and within this framework, a deep learning algorithm can improve train models with good performance for an imbalanced dataset. Based on this idea, we first apply "scrambling segmentation" or "global random sampling" to obtain multiple sets of relatively balanced datasets. During the model training process, all classifiers train weight through a parallel ensemble learning method and then iteratively update the optimal weight among them. Finally, all classifiers put all results of their training together for discussion in the decision-making layer for obtaining the best classification scheme.

The novel sampling method in our model can guarantee that all information is retained and each subset has the same distribution as the original dataset. In a parallel training layer, each classifier acts as an agent and is responsible for working within its subset. The built-in weight selection module can ensure that the initial weight of each training epoch is optimal. The efficiency of each classifier is much improved due to the reduced workload. The decision-making layer has a higher robustness for finding the "best available" prediction result of all classifiers.

In a short, we summarize three main contributions as follows:

- We present a brain storm-based framework by applying scrambling segmentation and global random sampling in generating multiple relatively balanced subsets of an imbalanced dataset.
- We develop a distributed and extensible training method to computing relatively balanced subsets efficiently.
- We introduce a decision-making method to generate optimal results for multiple predictions in decision-making layer.

This paper is further organized as follows. In the next section, we discuss related works. Section 3 addresses our method. Section 4 is devoted to experiments and evaluation. We conclude the work in Section 5.

2 Related Works

He et al. analyzed the causes regarding the performance degradation of the classifier in detail and summarized the existing research methods [7], which can be divided into data-based level, algorithm-based level, and transfer learning. Kubat et al. raised the problem of imbalanced learning when dealing with the detection of the oil spilled problem in 1998 [9]. Many studies have revealed that imbalanced learning problem is relevant to data complexity. Datta et al. proposed a method for identifying the disjunction problem in a dataset [6].

The research on data level mainly lies in the resampling method. Wang et al. used resampling to cope with online class imbalances. They proposed two resampling techniques (i.e., OOB and UOB), defined class imbalances and verified that data distribution determines the imbalance problem [16]. Charte et al. proposed specializing measures to assess the imbalance level for multilabel classification.

Using these measures to test which multilabel datasets are imbalanced, and presented both random undersampling and random oversampling algorithms to reduce the degree of imbalance [3]. Lin et al. proposed two resampling strategies to assist in the data preprocessing [11]. Charte et al. proposed a procedure to hybridize some resampling method with a novel algorithm designed to decouple imbalanced labels [2].

The cost-sensitive model mainly utilizes a cost matrix to impose distinct penalties on different classes. Khan et al. proposed a neural network combined with the cost-sensitive method, which can automatically robust extract feature vectors of the majority and minority class [8].

Transfer learning is to train the model on a related large-scale balanced dataset and then fine-tuning weight according to the target one. Al-Stouhi et al. presented a model combined with the transfer learning mechanism. This method used auxiliary data to make up for the lack of the imbalanced dataset [1]. Wang et al. used the conditional and marginal distribution discrepancies to treat the imbalanced learning problem [15].

3 Our Method

The structure of BSIL adopts the idea of distributed machine learning which is utilized to deal with the problem that large-scale data cannot be processed in a single node. BSIL utilizes a data preprocessing layer to transform the incoming imbalanced datasets into multi-group balanced subsets, then works a distributed ensemble learning framework with optimal weight selection strategy to train multiple agents, and finally uses a brainstorming algorithm to integrate the output of each agent to obtain the final results. This section will focus on the data preprocessing layer (DPL), the parallel training layer (PTL, the distributed ensemble learning framework) and the decision-making layer (DML, the brainstorming algorithm).

3.1 Data preprocessing layer (DPL)

To transform an imbalanced dataset into multi-group balanced subsets and ensure that their original distribution characteristics are not broken, we utilize "scrambling segmentation" or "global random sampling" to adjust the dataset (see Fig.1).

"Scrambling segmentation" randomly scrambles the training dataset, then divides it into the corresponding subset according to the number of classifiers in PTL. This method retains every sample so that the dataset information is not lost. "Global random sampling" randomly samples the data of the entire majority class. It guarantees that the subset and the original training dataset are independently and identically distributed.

Suppose our dataset has only two classes, the majority class $P : \{x_i, y_i\}$ and the minority class $N : \{x_j, y_j\}$. $|P|$ is much large than $|N|$ and $|\cdot|$ is the size of the corresponding class.

We first randomly sample m subsets P_i from P according to the size of the minority class, then we get m training subset $\{P_i, N\}$ which is obtained by combining P_i and N. We use $\{P_i, N\}$ to train the corresponding $agent_i$ and evaluate the performance of each agent and obtain the $F1\_Score$ value F_i related to them. If one of the F_i is greater than the threshold value λ (in our experiment, λ is 60%. This mainly depends on the baseline of the selected model), i.e., one of agents (classifiers) obtains a "good enough" prediction result, $\{P_i, N\}$ can be retained, otherwise, we will sample $m+1$ subsets of P and generate $m+1$ agents.

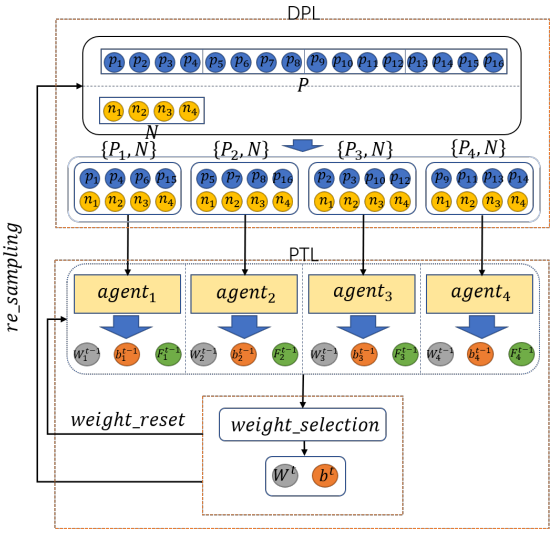

Fig. 1. Training process and internal structure.

3.2 Parallel Training Layer (PTL)

When we have m training subset $\{P_i, N\}$, the number of the agent in the parallel training layer is also fixed to m. In our framework, we used three basic NN-based models as agents(see Fig.1):

- **Convolutional Neural Networks (CNN)** is the deep neural network with the convolution structure. It has three key operations, including the convolution layer, weight sharing, and pooling layer.
- **Recurrent Neural Network (RNN)** is a type of neural network for processing sequence data. We used the most basic LSTM model which can get long short-term memory information.
- **Self-attention Neural Network (SaNN)** is a very popular NLP neural network which uses a global self-attention mechanism to enable the model to obtain more important feature information.

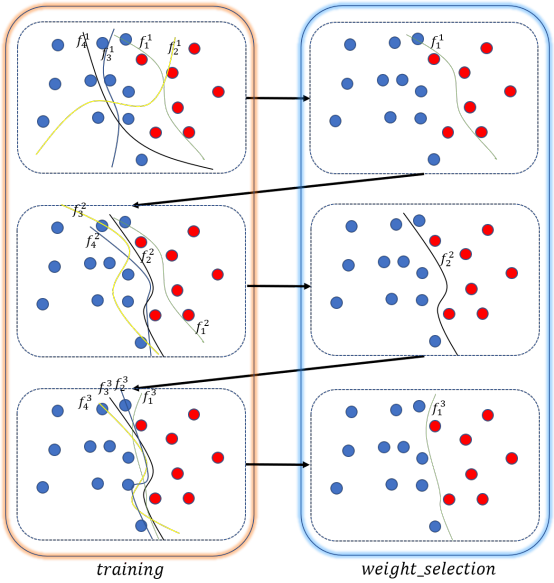

Fig. 2. Weight Selection Process.

Weight Selection Process Different agents can be considered as any functions f, which map the input samples $\{P_i, N\}$ into a new feature space. In the initial step of model training, since the weight is arbitrarily initialized, the mapping function f obtained by the training does not fit the feature space well. At time $t-1$, the weight selection mechanism selects the best weight $\{W_i^{t-1}, b_i^{t-1}\}$ as the initial weight of the succeeding time t by comparing the accuracy of all agents. Since the model has multiple sets of weights to choose at each time, the feature selection method improves the robustness of the model. Taking Fig.2 as an example. In the first step, all agents get a set of mapping functions $\{f_1^1, f_2^1, f_3^1, f_4^1\}$ through training. The weight selection method finds that f_1^1 has the highest accuracy and uses $\{W_1^1, b_1^1\}$ as the initial weight of the next step. During time 2, all agents use $\{W_1^1, b_1^1\}$ to train their mapping functions f_i^2. Repeat the above steps, the weight of f_1^3 is finally obtained. We can see that f_1^3 can distinguish the imbalanced dataset very well at this time.

3.3 Decision-making Layer (DML)

We generally divide the dataset into a training set, a validation set, and a test set. The training set is used to fit the parameters of the classifier. The validation set is used to tune the hyperparameters (i.e., the architecture) of a classifier and can evaluate the performance of each epoch's classifier. The test set is a dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset and the validation dataset.

Due to the consistency of the above probability distribution, if a classifier has higher accuracy in the validation set, then the classifier will also get higher one in the test set. In our framework, if one agent predict the validation set to obtain a higher $F1\_Score$ in the PTL module, then we believe that it can give us a better result for predicting the test set, so the DML gives it higher confidence. In our experiments, the accuracy of each agent play a role of confidence, which tells us which output of the agent should be trusted.

More formally, each agent train a set of $\{W_i^t, b_i^t\}$ and predict the evaluation dataset to get the $F1\_Score$ value F_i in PTL. F_i participates in the Eq.1 as a confidence.

$$pred_{final} = \frac{\sum_{n=1}^{N}(F_i \cdot pred_i)}{Z} \tag{1}$$

Here N is the number of agents, F_i is the $F1\_Score$ value, $pred_i$ is the prediction value of each agent. Z is a normalization factor to ensure that each output is 0 or 1. Due to the existence of F_i, the result tends to the outputs of several agents with higher F_i.

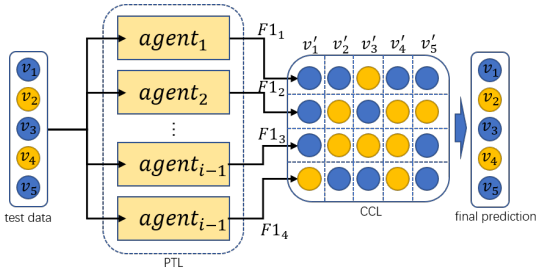

Fig. 3. Decision-making Process in DML.

Taking Fig.3 as an example, the ground true of the test dataset is [1, 0, 1, 0, 1]. After training, the prediction result of $agent_1$ is [1,1,0,1,1], and F_1 is 40%. The predicted result of $agent_2$ is [1,0,1,0,0], and F_2 is 80%. The predicted result of $agent_3$ is [1,0,0,0,1], and F_3 is 80%. The predicted result of $agent_4$ is [0, 1, 1, 0, 1], and F_4 is 60%. For sample 1, $agent_1$, $agent_2$, $agent_3$ predict it as a positive sample, so the final prediction is 1. For sample 2, although $agent_1$, $agent_4$ predict it as a positive sample, $agent_2$, $agent_3$ predict it as a negative sample, but $agent_2$, $agent_3$ have higher F_i, we should believe them more, so sample 2 is finally predicted to be a negative sample. By analogy, it can be seen that although they did not classify all the samples correctly in their respective task, they finally reached the correct result through discussion.

4 Experiments and Evaluation

In this section, we use the BSIL framework to combine with three different agents(CNN, RNN, and SaNN). We investigate the classification performance of the BSIL framework and compare it with state-of-the-art models for text classification and analyze the robustness of the DML on various datasets.

4.1 Experimental Evaluation Criteria

In imbalanced learning, accuracy has not been a good indicator of the performance of the classifier. The $Precision$, $Recall$, and $F1\_Score$ values are more reflective of the classification effect of the classifier on the minority class. In our experiments, we used the $F1\_Score$ value, which is a combination of $Precision$ and $Recall$, see Eq.2.

$$F1\_Score = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision} \tag{2}$$

$Precision$ is referred to as a positive predictive rate. $Recall$ is also referred to as the true positive rate or sensitivity (see Eq.3).

$$Precision = \frac{TP}{TP + FP}, \qquad Recall = \frac{TP}{TP + FN} \tag{3}$$

Where TP is the number of the positive sample which is right predicted. FP is the number of the negative sample which is predicted to be a positive one. FN is the number of the positive sample which is predicted to be a negative one.

4.2 Datasets

We choose five text classification datasets, which include English and Chinese datasets. They can be divided into the sentence and document levels. We used random sampling technology to construct their imbalances. We briefly summarize these datasets as follows:

- **MR**: Movie reviews with one sentence per review. Positive and negative sentiment polarity is included in the classification. Imbalance ratio (IR) is 1250:4250.
- **SST-2**: An extension of MR but with train/dev/test splits provided. IR is 1009:3310.
- **IMDB**: A document-level text classification dataset containing 100,000 movie reviews with binary labels. IR is 12500:50000.
- **SPAM**: A Chinese classification dataset for spam detection. IR is 635:2743.
- **CR**: A car review dataset. IR is 6000:22800.

The first two datasets are for sentence-level classification and the last three datasets are for document-level classification. IMDB is the biggest dataset and SPAM is the Chinese dataset. These five datasets involve as many different as possible(see Tab.1).

Table 1. Description of Datasets.

Dataset	Average Length	Vocabulary Size	Train Size	Test Size
MR	20	18K	1250:4250	CV
SST-2	18	15K	1009:3310	1821
IMDB	294	392K	12500:50000	22500
SPAM	574	22K	635:2743	1034
CR	100	133K	6000:22800	6300

4.3 Model Training and Hyper-parameters

In our experiment, the agents are set to CNN, RNN, and SaNN, respectively. The three agents use different parameter settings:

- **BSIL_LSTM**: Our framework combined with the LSTM model, i.e., all agents utilize the LSTM model. We use a bidirectional RNN, the learning rate is 0.3, the Sequence length is set according to different datasets, the dropout rate is 0.8, the size of the hidden layer is 32, and there are 2 hidden layers.
- **BSIL_CNN**: Our framework combined with the CNN model, i.e., all agents utilize the CNN model. This model uses a total of 100 convolution kernels. The size of the kernel is 1, i.e., each convolution kernel filters a word. The defined length of a sentence is also set according to different datasets.
- **BSIL_SaNN**: Our framework combined with the SaNN model, i.e., all agents utilize the SaNN model. The learning rate is 0.003, the number of scaled dot-product attention module is 8 and the number of encoder is 6.

The word embeddings initialized with the 300-dimensional word vector published by Glove. The objective function uses a cross-entropy loss function. The epochs of training are 10 and the batch size is 32.

4.4 Baselines

Our framework combined with CNN, RNN, and SaNN, so we need to compare with them to prove that they are not suitable for imbalanced dataset. Besides, we have compared many existing state-of-the-art models to illustrate that our models are indeed more suitable for imbalanced dataset than them.

- **LSTM**: The LSTM model mentioned in Section 3.2. We use it to compare with BSIL_LSTM.
- **LSTM+attention**: We implement flatten variants of the attention-based LSTM model for different levels of classification.
- **CNN**: The CNN model mentioned in Section 3.2. We use it to compare with BSIL_CNN.
- **SaNN**: The SaNN model mentioned in Section 3.2. A self-attention mechanism model for classification. We use it to compare with BSIL_SaNN.
- **FastText**: The FastText model is based on RNN.

- **TextCNN**: The TextCNN model is based on CNN.
- **SMOTE_(CNN/RNN/SaNN)**: The SMOTE algorithm is an oversampling method. It generates a new one between two samples, avoiding the problem of repeated sampling.
- **RandomUnderSampler_(CNN/RNN/SaNN)**: RandomUnderSampler is a undersampling method, we use the RandomUnderSampler made by scikit_learn.

4.5 Compared with Baselines

To illustrate the performance improvement of our BSIL model, we compare it with the robust baseline model, which has achieved the best results in their respective tasks.

Table 2. Classification perfermance of the different approaches.

Model	MR	SST-2	IMDB	SPAM	CR
LSTM	61.4	76.1	78.3	87.6	37.5
CNN	51.5	30.8	73.7	80.6	66.1
SaNN	58.4	54.3	60.2	68.0	24.4
LSTM+attention	64.5	78.7	80.2	89.2	44.4
FastText	60.3	74.1	79.2	88.2	50.5
TextCNN	52.9	49.1	77.0	82.9	60.2
SMOTE_(CNN/RNN/SaNN)	73.3	79.0	81.6	89.6	54.9
RandomUnderSampler_(CNN/RNN/SaNN)	74.5	80.0	81.4	**91.0**	74.8
BSIL_LSTM	**76.1**	**80.4**	**82.1**	90.9	67.7
BSIL_CNN	72.5	67.4	80.8	88.9	**76.4**
BSIL_SaNN	75.4	74.8	80.5	86.9	71.4

Due to our framework combines CNN, RNN, and SaNN, we first compare them with these three models shown in Table.2. We can see that LSTM, CNN, and SaNN have a very low performance on the imbalanced dataset. The F_Score of LSTM model on CR dataset only has 37.5% and the SaNN model on CR dataset only has 24.4%. The reason is CR has the more professional vocabulary. Therefore, CR is more suitable for CNN which is better at capturing keywords but lacks the ability of long short-term memory. When the dataset is skewed, the professional vocabulary in the majority class is more than that in the minority one, so the classifier has a serious over-fitting problem. the CNN model on SST-2 only has 30.8%. SST has a high degree of context-dependency, and there are many turning words in the sentence. Therefore, the LSTM model is more suitable for capturing this kind of semantic information. In the imbalanced dataset, the semantic information of minority class is also easily covered by the majority class, resulting in semantic over-fitting. Besides, CR is also a document-level dataset. Each sample is too long, which makes it difficult for

SaNN's self-attention mechanism to capture topic information. Therefore, its accuracy rate is only 24.4%.

Compared with the state-of-the-art models in Tab.2, our model give the best performance on four datasets. On the SPAM dataset, the undersampling algorithm is only 0.1% higher than ours. In addition to the CR dataset, BSIL combined with LSTM have better performance, BSIL-CNN has the best performance on the CR dataset because its own CNN model is more suitable for extracting local features.

Analysis The imbalance problem aggravates the skewed distribution of semantic information and keywords in the dataset, and our BSIL model first divides the dataset into multiple balanced data subsets and captures more useful information through the distributed learning model. Unlike the resampling method, our model does not lose useful information or generates virtual information, and ensures the original distribution of the dataset.

4.6 Ablation Experiment for Weight Selection

To illustrate the importance of weight selection in the model, we do ablation experiments on it. We remove the weight selection module from the BSIL model and retain only the distributed model.

Table 3. Ablation Experiment for Weight Selection

Model	MR	SST-2	IMDB	SPAM	CR
Distributed_LSTM	74.5	80.0	81.6	87.6	65.2
Distributed_CNN	71.3	66.3	80.0	88.7	75.1
Distributed_SaNN	75.2	72.9	79.4	86.5	70.3
BSIL_LSTM	**76.1**	**80.4**	**82.1**	**90.9**	**67.7**
BSIL_CNN	**72.5**	**67.4**	**80.8**	**88.9**	**76.4**
BSIL_SaNN	**75.4**	**74.8**	**80.5**	**86.9**	**71.4**

Analysis The experimental results show that the weight selection module can increase the accuracy of BSIL by 1%-2%. This is because the weight selection module chooses the best initialization scheme for the model. When an agent achieves a poor accuracy in the next step, the weight selection module will improve it and make it get rid of the existing bad situation.

4.7 Comparison Experiment for DML

To demonstrate the robustness of the DML, we compare the $F1\_score$ value of each agent with the final prediction result(see Tab.3). The DML uses Eq.1 to make the final predictions more accurate than the results of each agent. We also

find that when BSIL_LSTM trains CR dataset, even though only $agent_3$ achieves 64.3% accuracy, while the other three agents all get very low accuracy, the final result is still 3.4% higher than that of $agent_3$. The same situation occurs when BSIL_SaNN trains CR dataset.

Table 4. BSIL Framework Combined with Different NN-based Models

Model(dataset)	$agent_1$	$agent_2$	$agent_3$	$agent_4$	final result
BSIL_LSTM(MR)	72.4	73.4	72.7	74.5	**76.1**
BSIL_LSTM(SST-2)	78.1	75.4	80.0	-	**80.4**
BSIL_LSTM(IMDB)	77.6	80.4	81.4	80.3	**82.1**
BSIL_LSTM(SPAM)	87.4	90.7	90.9	89.0	**90.9**
BSIL_LSTM(CR)	57.5	54.8	64.3	42.8	**67.7**
BSIL_CNN(MR)	71.3	69.9	69.5	68.0	**72.5**
BSIL_CNN(SST-2)	66.3	57.7	62.8	-	**67.4**
BSIL_CNN(IMDB)	76.5	79.4	78.1	79.9	**80.8**
BSIL_CNN(SPAM)	85.9	87.1	87.7	88.7	**88.9**
BSIL_CNN(CR)	66.1	67.9	74.8	74.2	**76.4**
BSIL_SaNN(MR)	72.4	74.5	73.4	72.7	**75.4**
BSIL_SaNN(SST-2)	70.4	70.8	71.3	-	**74.8**
BSIL_SaNN(IMDB)	77.6	79.2	76.8	79.4	**80.5**
BSIL_SaNN(SPAM)	81.8	83.8	84.6	86.5	**86.9**
BSIL_SaNN(CR)	70.3	41.9	64.2	57.1	**71.4**

Analysis DML uses the brain storm algorithm to evaluate the final classification result for each sample in the test set. Based on the better results achieved by all agents, the final classification results are improved again. The brain storm algorithm effectively utilizes the principle of the minority obeying the majority and eliminates the result with a high probability of error.

5 Conclusion

In this paper, we propose a brain storm-based framework to improve those neural networks which are not always good at imbalanced text classification by applying brain storm optimization (BSO) in sampling imbalanced datasets. Our proposal is independent of neural networks and thus our approach is more adaptable for neural networks. Besides, our approach is easy to perform better in imbalanced text classification even for those neural networks relying on strict conditions of datasets. In the future work, we are interested to investigate a fine-grained metric to characterize imbalance index in text classification since the judgement of imbalanced datasets in BSIL depends on the ratio of positive samples and negative samples.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2017YFB1401200,2017YFC0908401) and the National Natural Science Foundation of China (61672377). Xiaowang Zhang is supported by the Peiyang Young Scholars in Tianjin University (2019XRX-0032).

References

1. Al-Stouhi, S., and Reddy, K. (2016). Transfer learning for class imbalance problems with inadequate data. *Knowl. Inf. Syst.*, 48(1):201–228.
2. Charte, F., Rivera, J., del Jesus, J., and Herrera, F. (2019). REMEDIAL-HwR: Tackling multilabel imbalance through label decoupling and data resampling hybridization. *Neurocomputing*, 326:110–122.
3. Charte, F., Rivera, J., del Jesus, J., and Herrera, F. (2015). Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, 163:3–16.
4. Chen, w., Cao, Y, Sun, Y, Liu Q, and Li Y. (2017). Improving brain storm optimization algorithm via simplex search. *arXiv*, CoRR abs/1712.03166.
5. Cheng, S., Qin, Q., Chen, J., and Shi, Y. (2016). Brain storm optimization algorithm: A review. *Artif. Intell. Rev.*, 46(4): 445–458.
6. Datta, S., Nag, S., Mullick, S., and Das, S. (2017). Diversifying support vector machines for boosting using kernel perturbation: Applications to class imbalance and small disjuncts. *arXiv*, CoRR abs/1712.08493.
7. He, H., and Garcia, A. (2008). Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* (9):1263–1284.
8. Khan, H., Hayat, M., Bennamoun, M., Sohel, A., and Togneri, R. (2018). Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Trans. Neural Netw. Learning Syst.*, 29(8):3573–3587.
9. Kubat, M., Holte, C., and Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine Learning*, 30(2-3):195–215.
10. Lai, S., Xu, L., Liu, K., and Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proc. of AAAI'15*, pp.2267–2273.
11. Lin, C., Tsai, F., Hu, H., and Jhang, S. (2017). Clustering-based undersampling in class-imbalanced data. *Inf. Sci.*, 409:17–26.
12. Moreo A., Esuli A., and Sebastiani F. (2016). Distributional random oversampling for imbalanced text classification. In *Proc. of SIGIR'16*, pp.805–808.
13. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., and Vanderplas, J. (2012). Scikit-learn: Machine learning in Python. *arXiv*, CoRR abs/1201.0490.
14. Sun Y., Kamel M., and Wang Y. (2006). Boosting for learning multiple classes with imbalanced class distribution. In *Proc. of ICDM'17*, pp.592–602.
15. Wang, J., Chen, Y., Hao, S., Feng, W., and Shen, Z. (2017). Balanced distribution adaptation for transfer learning. In: *Proc. of ICDM'17*, pp. 1129–1134.
16. Wang, S., Minku, L., Yao, X. (2015). Resampling-based ensemble methods for online class imbalance learning. *IEEE Trans. Knowl. Data Eng.*, 27(5):1356–1368.