Center Loss in Graph Neural Networks

Hugo Robalino (799951)
University of Potsdam

Prof. Dr. Constantine Kotropoulos
Aristotle University of Thessaloniki

Prof. Dr. Manfred Stede
University of Potsdam

Individual Module: April 19th, 2022.

Abstract

Fake news detection using graph neural networks has been demonstrated to be successful. Nevertheless, performance could be increased further by the center loss, which, in addition to separability, contributes with discriminative classes. It was shown that although the center loss equipped classes with discriminative power, it did not increase nor decrease model performance.

1 Introduction

Numerous methods already exist for fake news detection, that not only include textual information but also the graph on how the news spread among users. Fake news detection through Graph Neural Networks (GNN) has already achieved an impressive accuracy; for instance, 0.9722 on the GossipCop dataset (Dou et al., 2021). Nevertheless, this method may be improved by the discriminative power in classification of an additional loss function, the center loss function.

To determine whether the center loss is beneficial in the context of fake news detection through with GNNs, two datasets are introduced: Politifact and GossipCop. Next, the model architecture is described as well as the center loss function. Finally, the experiments and results are discussed, as well as the limitations and conclusion of this project.

2 Dataset

The fundamental dataset for this project is FakeNewsNet (Shu et al., 2020), which is a corpus of fake news on social media and includes news content, social context, and temporal information. FakeNewsNet makes use of two fact-checking websites: PolitiFact[1] and GossipCop[2] in order to collect ground truth labels.

Based on the aforementioned dataset, the User Preference-aware Fake News Detection dataset (UPFD) (Dou et al., 2021) is then built, which is available in PyTorch Geometric (PyG) (Fey and Lenssen, 2019).

In order to build the UPFD dataset, the authors fuse endogenous preference (user's historical data) and exogenous context (users' engagement on social media) as follows.

2.1 Endogenous Preference

Since the FakeNewsNet dataset provides the social engagement of users, it is possible to collect the last 200 tweets from a user and encode each of them with BERT (Devlin et al., 2018) and subsequently average all the tweet embeddings in order to obtain a user preference.

2.2 Exogenous Context

Exogenous context denotes how the news propagated among users. In other words, a graph representing propagation of a news piece among users.

Only two types of nodes exist in the news propagation graph: the root node and the user nodes. The root node is formed based on the news textual embedding, while the user nodes are formed by the user's endogenous preference embedding.

2.3 Information Fusion

The final step is to combine the endogenous preference and the exogenous context with a GNN, which is discussed in the Section 3. The statistics of the UPFD dataset are shown in Table 1.

[1] https://www.politifact.com/
[2] https://www.gossipcop.com/

Table 1: Dataset Statistics

Dataset	PolitiFact	GossipCop
#Graphs	314	5464
#Fake News	157	2732
#Total Nodes	41,054	314,262
#Total Edges	40,740	308,798
#Avg. Nodes per Graph	131	58

3 Model

It is worth noting that this is a supervised classification task, since the ground truth labels are provided by the UPFD dataset.

The first layer of the architecture is a GNN, more specifically, the Graph Convolutional Network (GCNConv) (Kipf and Welling, 2016) provided by PyG is used. GCNConv is capable of efficiently encoding node features and graph structure thanks to a first-order approximation of spectral convolutions on graphs.

Once the node features (endogenous preference and news piece embedding) and the graph structure (exogenous context) are encoded by the GCNConv, they are activated by the ReLU function. Next, a readout function, a max pooling operation, which aggregates features of adjacent nodes in order to learn the embedding of a node, is applied over all nodes in order to get the graph embedding.

Subsequently, the news piece textual embedding is concatenated to the already processed data with means of a linear layer. Finally, a fully connected linear layer with two output neurons (fake news vs. real news) in combination with a logarithmic softmax function is applied so that the logarithmic probabilities can be obtained. Figure 1 briefly summarizes the described architecture.

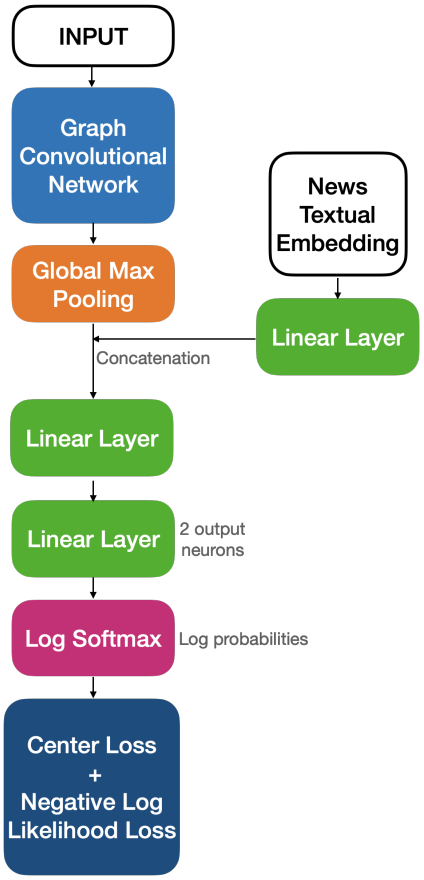


Figure 1: The model architecture. Each color box represents a different layer of the GNN, except the last one, which represents the joint loss function.

3.1 Loss Function

The model is then trained with a joint loss, which is a addition of two losses: the negative log likelihood loss, that is useful for classification tasks with multiple classes, and the center loss (Wen et al., 2016).

The center loss was originally intended for face recognition with CNN. Since using just one loss function, negative log likelihood in this case, leads features to only be separable, another supervision signal is needed in order for features to be discriminative as well.

This loss achieves discriminative features by learning a center for each class. The center of a class is a vector with the same dimension as a deep feature vector. While training, this loss learns to minimize the distance between deep features and their class centers, while the other loss keeps deep features of different classes to stay apart. In other words, the joint loss encourages intra-class feature distance to be reduced and inter-class feature dis-

tance to be enlarged.

$$L = L_{NLL} + \alpha L_C \tag{1}$$

Equation (1) shows that the joint loss has the parameter α, which controls the weight of the center loss. Moreover, the center loss can be updated in two ways: with two separate optimizers, one for the negative log likelihood loss and one for the center loss, or with one merged optimizer for both losses. Experiments were done with both methods.

Finally, one of the main motivations for integrating the center loss in conjunction with another loss in a GNN framework is that deeply learned features and class centers would be degraded to zeroes if only the center loss was used.

3.2 Optimizer

As stated in Section 3.1, two methods for optimization exist and, although several optimizers exist, it was decided to only make use of one, the Adaptive Moments Estimation (Adam) Optimizer (Kingma and Ba, 2014) for both methods. The Adam optimizer is a gradient based optimization for stochastic objective functions based on adaptive estimates and lower order moments, which makes it appropriate for large quantities of data or parameters and sparse gradients. Its effectiveness in CNN has also been demonstrated thanks to its adaptive learning rate scale for different layers.

4 Experiments

Regarding the experiments' setup, a default learning rate of 0.01 was established for both the negative log likelihood loss and the center loss. Moreover, a weight decay of 0.01 was set for the Adam optimizer. In addition, the model architecture uses no dropout ratio, a batch size of 128 and is run for 40 epochs. It also needs to be mentioned that the data is split into three parts: training set, validation set and test set, which correspond to 20%, 10% and 70% of the data respectively.

In the original research (Dou et al., 2021), other types of methods are used, such as word2vec from spaCy, but it was decided to only use BERT as it provided the best results among all embeddings. Moreover, two other types of GNN were proposed in the original experiment, such as SAGEConv or GATConv (also found in PyG), but it was also agreed to only use GCNConv for all experiments.

The main goal of this project is to determine if adding the center loss to a Graph Neural Network would improve the performance; to that end, three methods were implemented: no center loss (baseline), merged optimizer for negative log likelihood loss and center loss, and separate optimizers.

In order to construct a merged optimizer,the center loss parameters and the model parameters are added. The Adam optimizer then takes in the processed parameters. Subsequently, they are updated by multiplying each of them with a fraction as in (2):

$$\frac{lr_{cl}}{\alpha + lr} \tag{2}$$

where lr_{cl} is the center loss learning rate, lr is the model learning rate and α represents the weight parameter for the center loss. A value of 0.5 for $alpha$ was fixed for the experiments.

5 Discussion

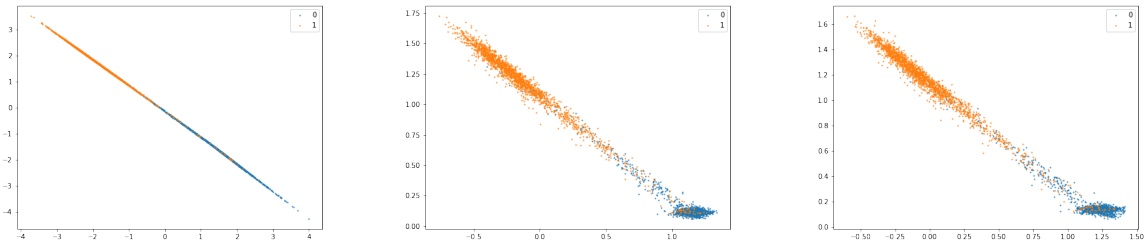
As can be seen in Table 2, the best performance is achieved by the model with no center loss with an accuracy of 0.8416 in the PolitiFact dataset and 0.9556 in the GossipCop dataset. Although, it must be noted that the model with the separate optimizers achieves a slightly better auc score in the PolitiFact dataset (0.8904) compared to the model with no center loss (0.8823).

Table 2: accuracy and area under the curve results on the test set

Method	**PolitiFact**		**GossipCop**	
	acc	auc	acc	auc
No Center Loss	**0.8416**	0.8823	**0.9556**	**0.9847**
Merged Optimizer	0.8371	0.8896	0.9537	0.9812
Separate Optimizers	0.8145	**0.8904**	0.9493	0.9799

Regarding the performance on the train and validation sets, there is no single model that achieves the best performance in both datasets. This is most probably due to the fact that the model learns from the train and validation sets.

With respect to the performance on GossipCop in Figure 2, one can notice that plotted instances are generally pulled towards the class center with models that make use of the center loss, with merged or separate optimizers. For example, one can compare Figure 2a, where the plotted instances align on a straight line, and Figure 2c,



(a) None (b) Merged (c) Separate

Figure 2: GossipCop test set class clusters

where instances tend to accumulate on the bottom right. Nevertheless, that happens mostly with the GossipCop dataset but not on the PolitiFact dataset, especially in the test set. The main reason for that can be that Gossipcop contains larger data (5464 graphs) than PolitiFact (314 graphs).

Concerning the performance on PolitiFact, Figure 3 shows a contrast on clusters between the train set and test set. It can be seen that the model achieves to form compact class clusters on the train set but not on the test set. This is to be contrasted with Figure 4, where compact class clusters are achieved on the train and test sets. This behavior might also be due to the dataset size difference.

Finally, according to Table 2, it is noticeable that performance among the different models does not vary greatly. In other words, accuracy and auc scores are similar among the three different models. This might be due to the fact that the center loss was originally intended for computer vision, more specifically face recognition, and not for graph deep learning. Other plausible reason can be that it might perform best when there are more than two classes. For instance as in the MNIST dataset.

6 Limitations

Within the research scope, only the GCNConv in combination with BERT text embeddings was used. In order to gain a deeper insight, other graph convolutional networks, such as SAGEConv or GATConv could be used. Additionally, other text embeddings could also be implemented; for example, word2vec from spaCy.

Another limitation is that the value of α is fixed to 0.5 and the value of lr_{cl} is also fixed to 0.01. Other values for both hyperparameters should be tested to gain a wider understanding of their impact on the center loss. It is also important to remember that those values were chosen, since they demonstrated an optimal performance in the original paper.

7 Conclusion

Although the center loss did encourage discriminative features and not only separability as seen in Figure 2 and Figure 4, it did not improve nor worsen the performance in the context of graph convolutional networks. Future research is needed to implement center loss in other graph neural networks and assess its performance including a wider range of hyperparameter values.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

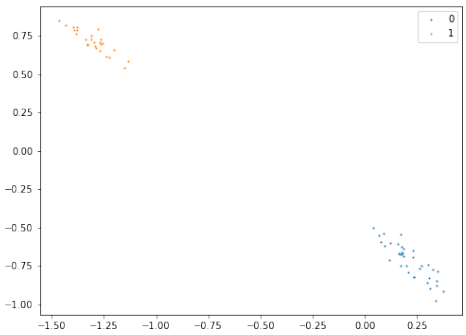
Yingtong Dou, Kai Shu, Congying Xia, Philip S Yu, and Lichao Sun. User preference-aware fake news detection. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2051–2055, 2021.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

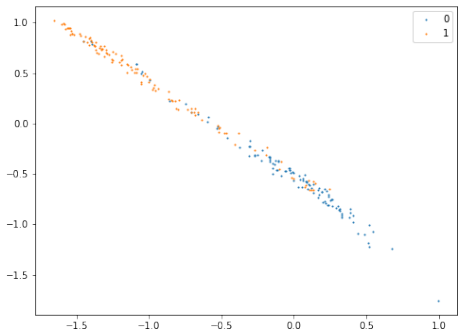
Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2016. URL https://arxiv.org/abs/1609.02907.

Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. Fakenewsnet: A data

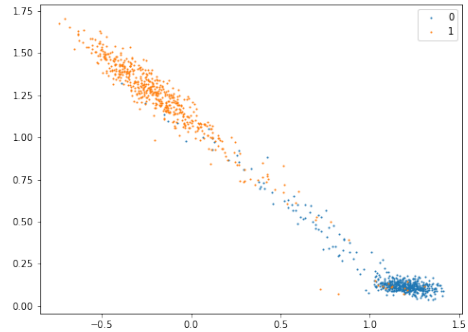


(a) Train set

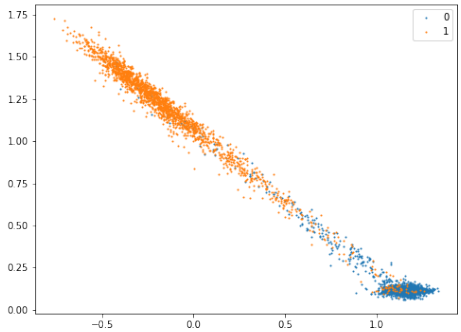


(b) Test set

Figure 3: PolitiFact merged model class clusters



(a) Train set



(b) Test set

Figure 4: GossipCop merged model class clusters

repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big data*, 8(3):171–188, 2020.

Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*, pages 499–515. Springer, 2016.