Performance Evaluation of Graph Convolutional Networks with Siamese Training for Few-Shot Classification of Nodes

Nichita Uțiu [1]

Abstract

Siamese networks learn embeddings which impose geometric constraints on the embedding space and are used in the context of one-/few-shot learning. In this paper, we empirically investigate applying this framework to Graph Convolutional Networks (GCNs). We test whether some lightweight architectures yield performance increases over plain Multi-Layer Perceptrons (MLPs) in tasks of one-/few-shot learning for nodes. We show that, for our benchmark, good performance can be achieved even with a fast Simplified Graph Convolutional Network (SGCN).

1. Introduction

Graphs are a ubiquitous data structure with applications in many fields. They allow the modeling of many systems such as networks, geo-spatial data, and knowledge bases. In many cases, vertices may have values attached to them. A particular example is when the values are discrete and may semantically correspond to a set of classes.

Graphs can grow over time due to additional data (eg. knowledge graphs). Classes may have semantics associated to them so labeling nodes can require expert knowledge. This may be unfeasible for large graphs. Labeling can be automated through prediction with ML by leveraging information about the topology and vertex data. A particular class of deep learning models that do this are Graph Convolutional Networks (GCNs) (Kipf & Welling, 2016a).

This paper focuses on a particular formulation of this classification task, the one-/few-shot learning task. Here, the set of classes is only partially observed during training, and the inference on *new classes* is based on one or a few support examples from them. For graphs, this problem can appear in the context of knowledge graphs where the number of types of relationships is ever-growing. Other problem-specific applications may exist as well.

We naively adapt the training regimen used by Siamese networks to GCNs. We propose some architectures and perform an initial step of model selection. We test our best model's few-shot accuracy against a simple Siamese MLP baseline and showcase the results. We show that better-than-MLP performance can be achieved even with a simple and computationally-fast Simplified Graph Convolutional Network (SGCN) architecture.

In Section 2 we review existing literature in the field of GCNs and few-shot learning. In Section 3 we describe our models and training procedure and put them in the context of existing methods. Section 4 presents our model-selection analysis, while Section 5 outlines the experimental design for testing few-shot accuracy against an MLP baseline. The numerical results of the experiments are presented in Section 6.

2. Related work

GCNs have been proposed in the context of node/graph embedding/learning (Kipf & Welling, 2016a;b; Hamilton et al., 2017). They improve upon the performance of previous deep learning methods such as the recurrent (Gori et al., 2005) or walk-based (Perozzi et al., 2014) approaches. Since then, numerous improvements to the architecture have been published (Wu et al., 2020).

Although not formally introduced as such, one-/few-shot learning dates back to the 90s when Siamese networks were proposed for signature recognition (Bromley et al., 1994; Wang et al., 2019). More recently, few-shot learning has become popular for tasks where retraining is impractical. One such task is facial recognition (Schroff et al., 2015).

In the last few years, GCNs have been used for Knowledge Graphs through contrastive learning (Xiong et al., 2018; Saebi et al., 2020; Dettmers et al., 2018). These applications usually involve using GCNs in conjunction with matching networks (Vinyals et al., 2016) for the task of link prediction. These methods are similar to the contrastive training we use.

[1]Romanian Institute of Science and Technology, Cluj-Napoca, Romania. Correspondence to: Nichita Uțiu <utiu@rist.ro>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

3. Method

The goal of this section is to outline both the architecture and the training method we propose for learning node embeddings. We define two sets of experiments:

- Compare the performance of different architectures in terms of loss (Section 4);
- Test the performance of our best model in terms of one-/few-shot accuracy (Section 5).

3.1. Losses

The general framework we use for training is the Siamese network framework (Bromley et al., 1994). In this framework, a network G is used to generate embeddings from the feature vector x_i of every sample i. Given prior knowledge of the classes y_i, this network is trained with an objective that seeks to impose a distance (L1 or L2) constraints on the embeddings. We explore the performance of two loss types: *triplet loss* (Chopra et al., 2005) and a modified version of it called *margin loss* (Wu et al., 2017).

- **Triplet loss** (Chopra et al., 2005) is an improvement over *contrastive loss* (Bromley et al., 1994). We define a training triplet as the triplet (x_i, x_j, x_k) with the property that $y_i = y_j$ and $y_i \neq y_k$. The elements of the triplet are called *anchor*, *positive* and *negative*.

$$\text{TripletLoss}(a, p, n) = \left[\|G(a) - G(p)\|_2^2 - \|G(a) - G(n)\|_2^2 + \alpha\right]_+ \quad (1)$$

  Here, α acts as a hard margin between the positive and negative distances, relaxing the constraint of the contrastive loss. Minimizing this loss ensures that all positive points of an anchor are closer by at least α than the negative ones. This loss, unlike the contrastive loss (Bromley et al., 1994), does not cause positive examples to collapse into single points.

- **Margin loss** was introduced in (Wu et al., 2017) as an extension to triplet loss and it introduces a learnable margin parameter β_0 and individual learnable margins for each sample and each of the classes. This way α no longer acts as a hard margin between the distances because it can be offset by β_i,

$$\beta_i = \beta_0 + \beta^i_{\text{(sample)}} + \beta^{y_i}_{\text{(class)}}. \quad (2)$$

$$\text{MarginLoss}(x_i, p, n) = \left[\alpha + \|G(x_i) - G(p)\|_2^2 - \beta_i\right]_+ + \left[\alpha - \|G(x_i) - G(n)\|_2^2 + \beta_i\right]_+ \quad (3)$$

  The β parameters are then optimized alternatively with the parameters of the embedding function (network).

3.2. Few-Shot Learning

In a few-shot classification task (Koch et al., 2015), the dataset is split into a training and a test set of samples (X_{train}, Y_{train}) and (X_{test}, Y_{test}) with disjoint sets of classes: for all $y \in Y_{train}$ and $y' \in Y_{test}$, $y \neq y'$. For all classes $c \in Y_{test}$ we also build a support set:

$$S_c = \{X_c^{(i)}\}_{i=1}^K, \text{ where } Y(X_c^{(i)}) = c \text{ and } X_c^{(i)} \in X_{test} \quad (4)$$

We call this set *the K-shot support set of class c*. Given a sample $x \in X_{test}$, we can predict its class y^* w.r.t. our embedding function G_Θ, by finding the class whose supports are closest in the embedding space:

$$y^* = \underset{c}{\text{argmin}} \frac{1}{K} \sum_{x_s \in S_c} \|G_\Theta(x) - G_\Theta(x_s)\|_2^2 \quad (5)$$

Based on these predictions and the size of the support set (K), we compute the K-shot accuracy of our models.

3.3. Architecture

GCNs are a class of models for representational learning applied to graphs and their nodes (Hamilton et al., 2017; Wu et al., 2020). For each node of a graph $\mathcal{G} = (V, E)$ with feature vectors $\{x_v \mid v \in V\}$, the algorithm consists of applying an arbitrary function f (usually a linear transformation with a non-linearity afterward) to get some intermediate values h_v for each node. These values are aggregated using a differentiable function (such as mean or sum) over each node's neighborhood 2^v to get the embeddings z_v. This process can be repeated several times to include information from nodes at distances greater than 1. The resulting embeddings can be trained end-to-end with an objective function.

The first graph convolutional model we tested is the GCN proposed by Kipf & Welling (2016a) where each linear transformation is followed by a non-linearity. We use LeakyReLU and sigmoid as opposed to $tanh$ because we noticed slightly faster convergence during training. We refer to this model as `lgcn`.

The other GCN architecture we test is the Simplified Graph Convolutional Networks (SGNC) (Wu et al., 2019). It applies the non-linearity after the final convolution and applies the same size-preserving linear transformation for each step of convolution. This means all transformations before the non-linearity can be written as matrix multiplications/powers. Therefore, this is a very lightweight model.

As for the MLP architecture, we use a fully connected network with ReLU as an activation function for the hidden layers and sigmoid for the last layer. The sizes of the layers decrease linearly from the number of input features to the number of classes used. To reduce overfitting, we use dropout after each layer with a probability of 0.5.

3.4. Sampling

Inspired by Wu et al. (2017) and Schroff et al. (2015), we also experiment with different sampling strategies for our triplets. The authors claim performance and stability increase through using what they call *semi-hard sampling* and *distance-weighted sampling*. As a baseline, we also sample triplets *uniformly*.

Semi-hard sampling was introduced by Schroff et al. (2015) and consists of selecting only *relatively difficult* triplets from a batch of samples. Within a batch, *hard-positive* pairs are defined as (x_a, x_p) where $p = \text{argmax}_p \|G(x_a) - G(x_p)\|_2^2$ and $y_a = y_p$ and similarly *hard-negative* pairs are those pairs (x_a, x_n) where $y_a \neq y_n$ and $n = \text{argmin}_n \|G(x_a) - G(x_n)\|_2^2$.

The authors experimented with both hard-negative and hard-positive mining. They propose selecting those triplets (x_a, x_p, x_n) which satisfy the following constraint:

$$n = \underset{n}{\text{argmin}} \|G(x_a) - G(x_n)\|_2^2 \tag{6}$$

where:

$$\|G(x_a) - G(x_p)\|_2^2 \leq \|G(x_a) - G(x_n)\|_2^2 \tag{7}$$

Another method is proposed by Wu et al. (2017), citing that *uniform* and *semi-hard* sampling provide a biased sample of distances as training progresses. As a result, they propose a method of sampling triplets where the distance between samples in a batch are L1-normalized, projecting the embeddings on the unit sphere. Given an anchor a, the negatives of a are sampled with the following probability:

$$P(n) = q\left(\|z'_a - z'_n\|_2^2\right)^{-1} \tag{8}$$

Here, q is the probability density function of distances of two different points on the sphere and z'_a and z'_n are the L1-normalized values of $G(x_a)$ and $G(x_n)$.

3.5. Learning Rate

Initial experiments with our models showed poor convergence when optimizing the network with an *Adam* (Kingma & Ba, 2014) optimizer and learning rates (LR) in the $(10^{-4}, 10^{-2})$ interval. To find some values which yielded better convergence, we tried using an SGD optimizer and used the *LR range* test proposed by Smith (2017). Using this method, we saw learning rates between 0.1 and 1 offer the best optimization for our models. Lower or larger values lead to poor convergence.

Hand tuning also showed that using *cyclical learning rates* (Smith, 2017) led to better results in most cases. For our experiments, we initially tested using the cosine annealing proposed for the Stochastic Gradient Descent with Restarts (SGDR) (Loshchilov & Hutter, 2016). However, due to its simplicity and alleged performance, we opted for the *1cycle* policy proposed by Smith & Topin (2017); Smith (2018).

4. Model Selection

We propose an experiment to compare our GCN models. The evaluation metric we use is the *triplet loss*. To have statistically-sound results, for each combination of models and hyperparameters, we run on 30 different 70%/15%/15% splits for 1000 epochs. We compute the 95% confidence intervals over all the runs. We use the CORA dataset (London & Getoor, 2014).

Initially, the models exhibited high sensibility to the LR, resulting in poor convergence. Thus, for every combination of parameters, we perform an initial step of LR tuning. Using an arbitrary split, we perform the *LR range test* to find the maximum LR for which the model is still stable. As recommended by Smith (2017), we set the minimum LR to $LR_{max}/4$.

As we can see from Figure 1, in terms of triplet loss, LCGNs and SGCNs have similar performance, and both provide a significant performance increase over MLPs.

5. Experiments

For the few-shot learning experiments, we need a dataset with a significant number of classes. The CORA dataset was not satisfactory so we used a preprocessed version of the NELL dataset (Yang et al., 2016; Carlson et al., 2010). Unlike CORA, which has 7 classes, NELL has 210, making it an ideal candidate for few-shot learning (Xiong et al., 2018).

For the experimental setup, a 70%/15%/15% split is performed on the dataset. In the few-shot learning task. This means splitting the dataset into disjoint sets of 148/31/31 classes. The NELL dataset is much larger then CORA and didn't fit into our GPU memory forcing us to run the model on the CPU. Therefore, we used only 4 different splits on runs of 300 epochs each.

Based on the results from the model selection experiment,

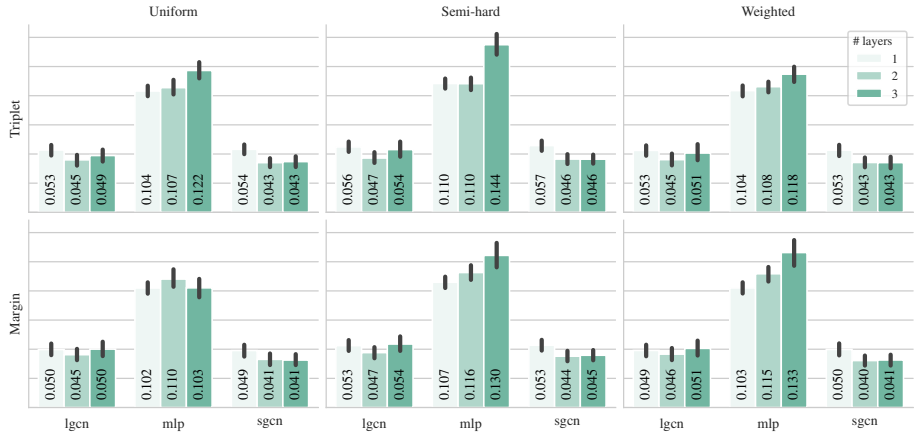

Figure 1. Results of the model selection experiment. Average triplet loss on the CORA dataset over 30 runs for every combination of: sampling method, model, and the number of layers. The error bars represent the 95% confidence interval for the values. `lgcn` refers to our modified vanilla GCN with LeakyReLU and sigmoid.

	1-shot @ top-1	1-shot @ top-5	5-shot @ top-1	5-shot @ top-5
mlp-1	0.49 ± 0.24	0.80 ± 0.25	0.62 ± 0.23	0.91 ± 0.06
mlp-2	0.52 ± 0.24	0.84 ± 0.19	0.63 ± 0.20	$\mathbf{0.93 \pm 0.04}$
mlp-3	0.44 ± 0.25	0.79 ± 0.22	0.52 ± 0.20	0.90 ± 0.07
sgcn-1	0.43 ± 0.17	0.81 ± 0.20	0.58 ± 0.17	0.90 ± 0.09
sgcn-2	0.54 ± 0.12	$\mathbf{0.91 \pm 0.05}$	$\mathbf{0.70 \pm 0.18}$	0.93 ± 0.10
sgcn-3	$\mathbf{0.55 \pm 0.12}$	0.91 ± 0.07	0.69 ± 0.17	0.93 ± 0.10

Table 1. One-/few-shot accuracy on for different models on the NELL dataset. The number in the model's name indicates the number of hidden layers for the MLP models and the number of convolutions for the SGCN. Top-K accuracy in both one- and few-shot tasks is shown. The results are given with a 95% confidence interval for 4 runs.

we choose SGCNs over LGCNs as they are computationally-cheaper. Triplet sampling is uniform, as neither semi-hard nor distance-weighted sampling yielded a compelling performance increase to justify their involved implementations.

6. Results

In table 1 we provide the results of the experiment. We present performance in terms of 1 and 5-shot accuracy both as top-1 and top-5. What we can immediately see is that, for all metrics, both `sgcn-2` and `sgcn-3` have better or similar results when compared to the MLP (linear) models with increases as large as 7% in accuracy. Moreover, their results are more consistent, with tighter intervals.

We can also see that `sgn-1` is the worst performing model. This is probably due to the the fact that it behaves exactly like the `mlp-1` model, except it lacks dropout. However we can see that simply adding information even from the first order neighbours, is more than enough to compensate for simplicity of the model, as, in all cases except for *5-shot @ top-5*, `sgcn-2` and `sgnc-3` outperform all the `mlp` baselines.

Although not shown in table 1, the SGCN implementation used was also significantly faster than the MLP.

7. Conclusion

We have shown that GCNs have the potential to be used in one-/few-shot learning tasks on nodes with significant performance increases over naive MLP baselines. SGCNs yield accuracy increases of almost 7% over MLPs in some cases even with fast uniform sampling.

8. Acknowledgements

This work was supported by the European Regional Development Fund and the Romanian Government through the Competitiveness Operational Programme 2014–2020, project ID P_37_679, MySMIS code 103319, contract no. 157/16.12.2016. The work was reviewed and supervised by Răzvan Valentin Florian.

References

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. Signature verification using a "siamese" time delay neural network. In *Advances in neural information processing systems*, pp. 737–744, 1994.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., and Mitchell, T. M. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pp. 539–546 vol. 1, June 2005. doi: 10.1109/CVPR.2005.202.

Dettmers, T., Minervini, P., Stenetorp, P., and Riedel, S. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Gori, M., Monfardini, G., and Scarselli, F. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pp. 729–734. IEEE, 2005.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pp. 1024–1034, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Kipf, T. N. and Welling, M. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Koch, G., Zemel, R., and Salakhutdinov, R. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

London, B. and Getoor, L. Collective classification of network data. *Data Classification: Algorithms and Applications*, 399, 2014.

Loshchilov, I. and Hutter, F. SGDR: Stochastic gradient descent with restarts. *CoRR*, abs/1608.03983, 2016. URL http://arxiv.org/abs/1608.03983.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Saebi, M., Krieg, S., Zhang, C., Jiang, M., and Chawla, N. Heterogeneous relational reasoning in knowledge graphs with reinforcement learning. *arXiv preprint arXiv:2003.06050*, 2020.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Smith, L. N. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472. IEEE, 2017.

Smith, L. N. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. *CoRR*, abs/1803.09820, 2018. URL http://arxiv.org/abs/1803.09820.

Smith, L. N. and Topin, N. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017. URL http://arxiv.org/abs/1708.07120.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Wang, Y., Yao, Q., Kwok, J., and Ni, L. M. Generalizing from a few examples: A survey on few-shot learning. *arXiv preprint arXiv:1904.05046*, apr 2019. URL http://arxiv.org/abs/1904.05046.

Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P. Sampling matters in deep embedding learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2840–2848, 2017.

Wu, F., Zhang, T., Jr., A. H. S., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. *CoRR*, abs/1902.07153, 2019. URL http://arxiv.org/abs/1902.07153.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Xiong, W., Yu, M., Chang, S., Guo, X., and Wang, W. Y. One-shot relational learning for knowledge graphs. *CoRR*, abs/1808.09040, 2018. URL http://arxiv.org/abs/1808.09040.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. *CoRR*, abs/1603.08861, 2016. URL http://arxiv.org/abs/1603.08861.