ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Tangent Graph Convolutional Network

Luca Pasa, Nicolò Navarin and Alessandro Sperduti *

University of Padua
Department of Mathematics "Tullio Levi-Civita"
Human Inspired Technology Research Centre (HIT)
via Trieste 63, 35121, Padua - Italy

Abstract. Most Graph Convolutions (GCs) proposed in the Graph Neural Networks (GNNs) literature share the principle of computing topologically enriched node representations based on the ones of their neighbors. In this paper, we propose a novel GNN named Tangent Graph Convolutional Network (TGCN) that, in addition to the traditional GC approach, exploits a novel GC that computes node embeddings based on the differences between the attributes of a vertex and the attributes of its neighbors. This allows the GC to characterize each node's neighbor by computing its tangent space representation with respect to the considered vertex.

1 Introduction

In the last few years, the definition of machine learning methods, particularly neural networks, for graph-structured inputs has been gaining increasing attention in literature. Neural Networks for Graphs (GNNs), while dating back to more than 20 years ago [1], have recently gained popularity due to the good results in tasks such as semi-supervised node classification [2], link prediction [2], graph classification [3] and graph generation [4]. The first works extending neural networks to inputs in the graph domain [1, 5, 6] are based on the idea of aggregating the representation of a node and its neighbors, either in a recursive or a feed-forward (convolutive) way. This idea has been re-branded later as *graph convolution* or *neural message passing*. In general, the main idea is to define the neural architecture following the topology of the graph. For each vertex, a new hidden representation is computed through an aggregation function that involves the vertex and its neighborhood. The aggregation function depends on some parameters, that may be shared among all the vertices.
Recently, many graph convolutions have been proposed, the majority of them sharing the basic principle of generating a (fixed-size) node representation considering its local neighborhood, obtaining topologically enriched representations of the node. Almost all proposals, however, do not consider the differences between the attributes of a vertex and the attributes of each single neighbour. In this paper, we take a step forward in exploiting these differences by considering the graph as a set of related samples from a manifold in the space of vertex attributes. According to this view, it is possible to characterize the neighborhood of a vertex by computing the attributes' tangent space with respect to the considered vertex.

*This research was supported by the Department of Mathematics, University of Padua with the SID/BIRD 2020 project "Deep Graph Memory Networks" and with the provision of the necessary HPC resources.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

In a nutshell, the attributes' tangent space describes what are the directions to follow to transform the attributes of the considered vertex into the attributes of its neighbours. The idea of using an operator based on the tangent space was already exploited in a handwritten digit classification task [7] to learn a classifier invariant to legal transformations of the morphology of a handwritten digit, i.e. small rotations, scale transformations, etc. The rationale behind our proposal is that existing convolutions transform the set of node neighbors representations into a single point that is a sort of centroid in some affine space. The tangent space generated by the edge-based vertex-neighbor differences, on the other side, encodes the principal directions of change in such a set, thus it encodes complementary information. Starting from this idea, we propose a novel graph convolution that exploits the principal subspace of dimension k of the differences between the attributes of a vertex and the attributes of its neighbours, i.e. the tangent space of dimension k. This convolution is then exploited, in conjunction with a traditional graph convolution, to define a novel network combining both of them: the traditional graph convolution focuses on the attributes of a vertex and its neighbors, while the tangent graph convolution focuses on the differences between the attributes of adjacent vertices. The tangent subspace basis for each layer is initialized using an SVD-based approach and adapted during training via a specifically designed. loss function that includes a term to enforce the orthonormality of the basis. The cost of the initialization phase turns out to be negligible. Note that the initialization process is performed only once, before the start of the training phase. Experimental results on several datasets show the benefit of the proposed approach.

2 Background

We use italic letters to refer to variables, bold lowercase to refer to vectors, and bold uppercase letters to refer to matrices. We use uppercase letters to refer to sets or tuples. Let $G = (V, E, \mathbf{X})$ be a graph, where $V = \{v_0, \ldots, v_{n-1}\}$ denotes the set of vertices (or nodes) of the graph, $E \subseteq V \times V$ is the set of edges, and $\mathbf{X} \in \mathbb{R}^{n \times s}$ is a multivariate signal on the graph nodes with the i-th row representing the attributes of v_i. In the following, we also consider the training set Tr which is the set containing all the graphs used to train the model. Moreover, we also define the set $E^{Tr} = \{E^G | \forall G \in Tr\}$ containing all the edges of the graphs belonging to the training set Tr. With $\mathcal{N}(v)$ we denote the set of nodes adjacent to node v. In the following definitions, for the sake of simplicity, we ignore the bias terms.

In this work, along with the Tangent Graph Convolution, we also use the *GraphConv* convolution proposed in [8], defined as:

$$GraphConv(\mathbf{h}_v^{(i-1)}) = \sigma(\mathbf{h}_v^{(i-1)}\mathbf{W}_1^{(i)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(i-1)}\mathbf{W}_2^{(i)}), \tag{1}$$

where $\mathbf{W}_1^{(i)}, \mathbf{W}_2^{(i)} \in \mathbb{R}^{m_{i-1} \times m_i}$ (with $m_0 = s$) are the convolution parameters, and σ is an element-wise (usually, nonlinear) activation function.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

3 Tangent GCN

In this section, we introduce our tangent-based graph neural network. We start by defining our Tangent Graph Convolution (TGC) as:

$$TGC(\mathbf{h}_v^{(i-1)}) = \sigma(\mathbf{h}_v^{(i-1)} + \sum_{u \in \mathcal{N}(v)} (\mathbf{h}_u^{(i-1)} - \mathbf{h}_v^{(i-1)})[\mathbf{t}_1^{(i)}, \ldots, \mathbf{t}_k^{(i)}][\mathbf{t}_1^{(i)}, \ldots, \mathbf{t}_k^{(i)}]^\top \mathbf{W}^{(i)}),$$

where $\mathbf{W}^{(i)} \in \mathbb{R}^{m_{i-1} \times m_i}$, $m_0 = s$, $\mathbf{h}_v^{(i-1)} \in \mathbb{R}^{1 \times m_{i-1}}$, and $\mathbf{t}_j^{(i)} \in \mathbb{R}^{m_{i-1} \times 1}$ is such that $\|\mathbf{t}_j^{(i)}\| = 1$, and ${\mathbf{t}_j^{(i)}}^\top \mathbf{t}_p^{(i)} = 0$ if and only if $j \neq p$, i.e., $\mathbf{T}^{(i)} = [\mathbf{t}_1^{(i)}, \ldots, \mathbf{t}_k^{(i)}]$ is a basis of the tangent space to be learned. The convolution computes the differences between the representation of a vertex and each of its neighbours. Each difference vector is then projected onto the tangent space to obtain the coefficients that will be used to represent the projection of the difference vector in the tangent space. The vector is then linearly transformed by the weight matrix $\mathbf{W}^{(i)}$.

The Tangent GCN (TGCN) is then defined as a multi-layer model, where each layer (except for the first) is composed of two covolutional operators: a TGC and a *GraphConv*. The model, with l convolutional layers, is defined as follows ($i \in [2 \ldots l]$):

$$\begin{aligned}
\mathbf{h}_v^{G^{(1)}} &= \mathbf{h}_v^{T^{(1)}} = \mathbf{h}_v^{(1)} = GraphConv(\mathbf{x}_v), \\
\mathbf{h}_v^{G^{(i)}} &= GraphConv(\mathbf{h}_v^{G(i-1)}),\ \mathbf{h}_v^{G^{(i)}} \in \mathbb{R}^{1 \times m_i^G}, \\
\mathbf{h}_v^{T^{(i)}} &= TGC(\mathbf{h}_v^{T^{(i-1)}}),\ \mathbf{h}_v^{T^{(i)}} \in \mathbb{R}^{1 \times m_i^T}, \\
\mathbf{h}_v^{(i)} &= [\mathbf{h}_v^{G^{(i)}}, \mathbf{h}_v^{T^{(i)}}].
\end{aligned} \tag{2}$$

The vertex embeddings are then aggregated to obtain a single graph-level representation by concatenating the sum, the maximum and the average operators [3] for all the layers obtaining $\mathbf{s} = [sum(\{\mathbf{h}_v^{(i)}\}), avg(\{\mathbf{h}_v^{(i)}\}), max(\{\mathbf{h}_v^{(i)}\}) |\ 1 \leq i \leq l]$, where *sum*, *max* and *avg* are computed element-wise. The readout is defined by a composition of zero or more dense layers $\mathbf{y}_j = ReLu(\mathbf{y}_{j-1}\mathbf{W}_j^{readout})$, $j = 1, \ldots, q,\ \mathbf{y}_0 = \mathbf{s}$, followed by an output layer $\mathbf{o} = LogSoftmax(\mathbf{y}_q \mathbf{W}^{out})$.

Weights initialization. All the weights of *GraphConv*s are randomly initialized, as well as the weight matrices of the *TGC*, except for the matrices $\mathbf{T}^{(i)}$, which are initialized as follows. First of all, we set $\mathbf{h}_v^{T^{(1)}} = \mathbf{h}_v^{G^{(1)}}$. Then, for $i \in [2 \ldots l]$, the matrices $\mathbf{T}^{(i)} = \mathbf{U}_k^{(i)}$ are defined, where $\mathbf{V}_k^{(i)} \mathbf{S}_k^{(i)} {\mathbf{U}_k^{(i)}}^\top$ is the k-truncated SVD decomposition of matrices $\Delta\mathbf{H}^{(i)}$ composed of a row vector $\mathbf{h}_u^{T^{(i-1)}} - \mathbf{h}_v^{T^{(i-1)}}$ for each edge $(v, u) \in E^{Tr}$:

$$\Delta\mathbf{H}^{(i)} = \left[\ \mathbf{h}_u^{T^{(i-1)}} - \mathbf{h}_v^{T^{(i-1)}}\ \right]_{\forall (v,u) \in E^{Tr}} \in \mathbb{R}^{|E^{Tr}| \times m_{i-1}^T}.$$

Please, notice that $\Delta\mathbf{H}^{(2)}$ can be readily defined starting from $\mathbf{h}_v^{T^{(1)}}$. Moreover, by construction $\mathbf{U}_k^{(i)}$ will constitute a basis of the k-subspace of the difference vectors. Finally, for computational reasons, $\mathbf{U}_k^{(i)}$ is computed from ${\Delta\mathbf{H}^{(i)}}^\top \Delta\mathbf{H}^{(i)}$.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Model \ Dataset	PTC	NCI1	PROTEINS	D&D	COLLAB	IMDB-B	IMDB-M
PSCN[14]	**60.0±4.8**	76.3±1.7	**75.0±2.5**	76.3±2.6	72.6±2.2	71.0±2.3	45.2±2.8
FGCNN[3]	58.8±1.8	81.5±0.4	74.6±0.8	77.5±0.9	-	-	-
DGCNN[3]	57.1±2.2	72.9±0.9	73.9±0.4	78.1±0.7	-	-	-
DGCNN[19]	-	76.4±1.7	72.9±3.5	76.6±4.3	57.4±1.9	53.3±5.0	38.6±2.2
GIN[19]	-	80.0±1.4	73.3±4.0	75.3±2.9	**75.9±1.9**	66.8±3.9	42.2±4.6
DIFFPOOL[19]	-	76.9±1.9	73.7±3.5	75.0±3.5	67.7±1.9	68.3±6.1	45.1±3.2
GraphSAGE[19]	-	76.0±1.8	73.0±4.5	72.9±2.0	71.6±1.5	69.9±4.6	47.2±3.6
TGCN	58.6±8.9	**82.8±2.0**	73.9±3.6	**78.7±3.8**	74.9±1.9	**72.1±4.9**	**49.1±3.6**

Table 1: TGCN accuracy comparison. Epoch selected on validation set.

Loss Function. The matrices $\mathbf{T}^{(i)}$ are learnable parameters. Therefore, during training the value of the tangent weights are updated. In order to maintain the orthonormality property through the optimization process, the loss function for the TGCN model, in addition to the term used to learn the task (e.g. the negative log likelihood for classification tasks), includes a term to guarantee the orthonormality of the $\mathbf{T}^{(i)}$ matrices, i.e. $\mathcal{L} = \mathcal{C}(\mathbf{o}, \hat{\mathbf{o}}) + \gamma(\sum_{i=2}^{l} ||{\mathbf{T}^{(i)}}^{\top}\mathbf{T}^{(i)} - \mathbf{I}||)$, where $\mathcal{C}$ is the cost function used to learn the considered task, and $\hat{\mathbf{o}}$ is the target. The second term enforces orthonormality by penalizing the covariance matrix of $\mathbf{T}^{(i)}$ to be far from the identity matrix $\mathbf{I}$. In our experiments, the value of γ was selected through the model selection process.

4 Experimental Results

We empirically validated the proposed TGCN on 4 commonly adopted graph classification benchmarks modeling bioinformatics problems (PTC [9], NCI1 [10], PROTEINS [11] and D&D [12]) and on 3 large graph social datasets (COLLAB, IMDB-B, IMDB-M [13]). We compare TGCN versus several state-of-the-art GNN architectures: PSCN [14], Funnel GCNN (FGCNN) [3], DGCNN [15], GIN [16], DIFFPOOL [17] and GraphSage [18]. The results were obtained by performing 5 runs of 10-fold cross-validation. The hyper-parameters of the model were selected using a grid search, where the explored sets of values were changed based on the considered dataset. As validation methodology we decided to follow the method proposed in [19], that in our opinion, turns out to be the fairest. For this reason the results reported in [16, 20, 17] are not considered in our comparison since the model selection strategy is different from the one we adopted and this makes the results not comparable. For the sake of completeness, we report the (higher) results obtained by the TGCN with the validation policy used in [16, 20, 17] in the Github repository of the project that also hosts the code used for the experiments[1]. In the same web page we report the values of the hyper-parameters used to perform the grid search. For the experimental evaluation, we used the *LeakyRelu* activation function for the GC and TGC layers, while we used *Relu* for the readout layers. For what concerns the readout, we considered a funnel

[1] `https://github.com/lpasa/TangentGCN`

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

structure inspired by the one in [3], and we considered $q = 0, 1, 2$ readout layers. The readout structure, as the other hyper-parameters, was chosen independently for each experiment by the validation process.

The results reported in Table 1 show better performances for TGCN compared to competing methods in four datasets, and competitive performances on the remaining three datasets. Specifically, in NCI1, D&D, IMDB-B and IMDB-M TGCN outperforms state-of-the-art results. In the COLLAB dataset, TGCN obtained the second higher result, while in PTC, the smallest considered dataset, TGCN performance is close to FGCNN, but it shows an higher variance.

To asses the influence of each type of convolution, we performed an ablation study on NCI1 and IMDB-B. We developed two baseline models that share the same structure of TGCN. The first model exploits only *GraphConv* operators, therefore the operations defined in eq. (2) are not performed, while the second one does not compute $\mathbf{h}_v^{G^{(i)}}$, $i \in [2 \dots l]$. Both models were validated using the same procedure used for the TGCN. The first baseline model achieved an accuracy of 81.3±1.9 on NCI1 and 70.8±4.4 on IMDB-B, while the second one reached an accuracy of 81.8±1.5 on NCI1 and 70.4±5.2 on IMDB-B, confirming that the two convolutions learn different representations that encode complementary information.

5 Conclusion and Future Directions

In this paper, we proposed a novel graph convolution, dubbed Tangent Graph Convolution (TGC), that aggregates the projections of the *differences* between the representations of a node and its neighbors in a tangent space initialized as the principal components of the training set node attributes differences, and that is refined during training while enforcing orthonormality via an ad-hoc loss term. We exploited the TGC, along with the widely adopted *GraphConv*, to develop a GNN architecture, dubbed TGCN, for graph classification tasks. We empirically validated the TGCN on 7 benchmark datasets, achieving state-of-the-art-result in 4 of them, and competitive results on the remaining 3.

In the future, we plan to study more expressive architectures exploiting the TGC, and to explore more complex aggregation methods. Furthermore, we will study more in deep the influence of the information conveyed by the representation of the vertex' neighbors in the tangent space.

References

[1] Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Trans. Neural Networks*, 8(3):714–735, 1997.

[2] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, pages 1–14, 2017.

[3] Nicolò Navarin, Dinh Van Tran, and Alessandro Sperduti. Learning kernel-based embeddings in graph neural networks. In *24th European Conference on Artificial Intelligence - ECAI 2020*, 2020.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

[4] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. Constrained Graph Variational Autoencoders for Molecule Design. In *NeurIPS*, may 2018.

[5] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

[6] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[7] Diego Sona, Alessandro Sperduti, and Antonina Starita. A constructive learning algorithm for discriminant tangent models. In Michael Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, pages 786–792. MIT Press, 1996.

[8] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609, oct 2019.

[9] Christoph Helma, Ross D. King, Stefan Kramer, and Ashwin Srinivasan. The predictive toxicology challenge 2000–2001. *Bioinformatics*, 17(1):107–108, 2001.

[10] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

[11] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[12] Paul D Dobson and Andrew J Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of molecular biology*, 330(4):771–783, 2003.

[13] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.

[14] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, pages 2014–2023, 2016.

[15] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[16] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

[17] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, pages 1024–1034, 2017.

[19] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *International Conference on Learning Representations*, 2020.

[20] Ting Chen, Song Bian, and Yizhou Sun. Are powerful graph neural nets necessary? a dissection on graph classification. *arXiv preprint arXiv:1905.04579*, 2019.