2019sp-cs-8501-Deep2Read Scribe Notes

Scribe: Faizan Ahmad ¡fa7pdn@virginia.edu¿

June 1, 2019

Geometric Deep Learning

1 Introduction

In the last few years, deep learning has been used to solve a wide range of problems such as image recognition and text classification. Models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) achieve state of the art results on many vision and text related tasks. However, these models are trained on data in a euclidean space. Their extensions to geometric (non-euclidean) data like graphs and 3d objects had remained elusive until the last 2-3 years. Deep learning for geometric objects is an active area of research and this scribe explains one of the proposed methods.

2 Why Geometric Deep Learning

Most of the widely accepted deep learning methods such as CNNs and RNNs are hard to extend to problems in non-euclidean domain. Therefore, there is a need for models that are able to work on geometric data.

2.1 Euclidean vs Non-Euclidean Geometry

Euclidean geometry, described by a Greek Mathematician Euclid, is an axiom system about points, lines, and planes. It consists of five axioms:

1. A straight line may be drawn between any two points.
2. Any terminated straight line may be extended indefinitely.
3. A circle may be drawn with any given point as center and any given radius.
4. All right angles are equal.
5. If two straight lines in a plane are met by another line, and if the sum of the internal angles on one side is less than two right angles, then the straight lines will meet if extended sufficiently on the side on which the sum of the angles is less than two right angles.

Examples of euclidean data are images which are defined on a 2d plane. If we curve an image, it becomes a 3d object, which can be considered a geometric object. In contrast to euclidean, non-euclidean data do not possess properties like common coordinate system and shift invariance, which makes it hard to extend well known deep learning models to such type of data.

3 Graph Convolutional Networks

One of the methods proposed in the last few years to apply convolutional neural networks on graphs is Graph Convolutional Networks (GCN) [1]. GCN uses spectral convolutions on graphs to learn characteristics of the nodes for different tasks. The goal in these networks is to learn a function on a graph $G \in (V, E)$ that takes as input:

- A feature description x_i for every node in a N x D feature matrix where D is the number of input features.
- Description of the graph in a matrix form. Typically, we take the adjacency matrix A.

and produce a node level output N x Z where Z is the number of output features per node. Each neural network layer is defined as:

$$H^{(l+1)} = f(H^{(l)}, A) \tag{1}$$

$$f(H^{(l)}, A) = \sigma(AH^{(l)}W^{(l)}) \tag{2}$$

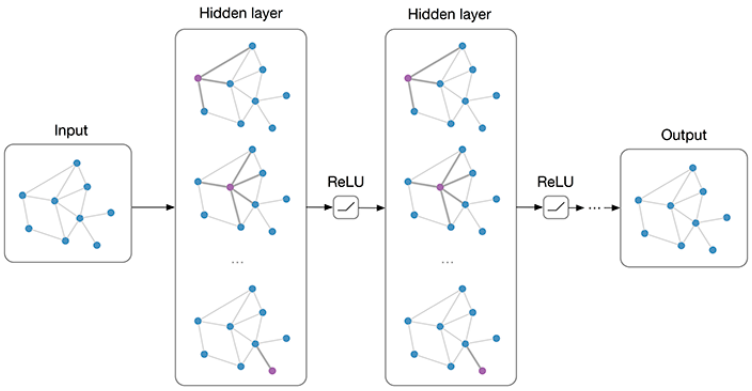

Figure 1: Graph Convolutional Networks

However, there are two main limitations with equation 2. First, for each node, we only sum up the feature vectors of its neighbors, not itself. To fix this, we add a self loop to each node which is equal to adding an identity matrix

to A. Secondly, A is not normalized, which can change the coordinate scales upon multiplication. Multiplying A with inverse of the diagonal node degree matrix D solves this problem. The GCN paper goes one step further and uses symmetric normalization $D^{-1/2}AD^{-1/2}$. The final equation becomes:

$$f(H^{(l)}, A) = \sigma(D^{-1/2}(A + I)D^{-1/2}H^{(l)}W^{(l)}) \tag{3}$$

Figure 1 demonstrates a GCN. An input graph is passed through several layers before we get embeddings for every node. These embeddings can be aggregated together using sum or pooling to get a representation of the whole graph.

3.1 Embeddings Visualization

GCN is a powerful model that extracts effective node embeddings even before the training starts.

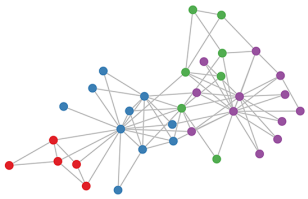

Figure 2: Karate club graph, colors denote communities obtained via modularity-based clustering

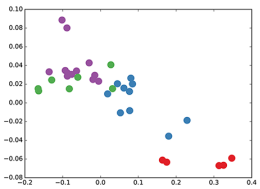

Figure 3: GCN embedding (with random weights) for nodes in the karate club network.

Figure 4: Visualization of node embeddings created after one forward propagation in the network

Figure 2 shows a graph for a karate club. Data for this graph is passed once in the GCN with randomly initialized weights. In the last layer, embeddings for each node are extracted and shown in figure 3. The node embeddings demonstrate that GCNs are effective in capturing patterns in the graph and the embeddings it creates resemble closely the structure of the input graph.

4 Conclusion

Graph convolutional networks uses spectral convolutions to solve graph-related tasks with a neural network. The concept of convolution has been extended from a euclidean space to non-euclidean space (Graphs) by the use of spectral convolutions. These networks have been applied to citation networks and knowledge bases and demonstrate state of the art results.

DeepWalk - Turning Graphs into Features via Network Embeddings

5 Introduction

Generating meaningful representations of nodes in graphs is an important problem. Such representations can be used in a wide variety of tasks such as node classification, link prediction, community clustering etc. DeepWALK proposes a deep learning based method to create latent representations of nodes based on their neighbors. These representations outperform other benchmark methods in a plethora of tasks.

6 Motivation for Node Representation

Creating meaningful representations of nodes can help in a variety of tasks. However, such representations are not easy to create; they usually require global knowledge of the graph and are ineffective when the graphs are small [4], [3]. Moreover, these representations need to encompass some semantic meaning about the structure and neighborhood of the graph. DeepWALK proposes such embeddings that outperform all benchmark methods and demonstrate semantic information about the graph.

7 Algorithm

DeepWALK is an extension of the word2vec algorithm [2] that has been extensively used to create word embeddings. The basic idea behind word2vec is that a word mostly appears in the same neighborhood (skipgram algorithm). More precisely, in word2vec, given a word w_i, we try to predict the neighboring words $w_{i-1}, ..., w_{i-w_n}$ and $w_{i+1}, ..., w_{i+w_n}$ where w_n is the window size. A neural network is trained that takes as input a word w and predicts the probability of other words being the neighbors of w. This simple idea has been extended to graphs by DeepWALK to create node embeddings. It works in three steps:

- **Random Walk Generation:** We randomly pick a node in the graph and perform a random walk to t steps. A random walk from a given node N is a sequence of steps taken uniformly to all neighbors from N.
- **Representation Mapping:** Once we have a random walk W as $N_1, N_2, ..., N_N$, we generate a vector representation of all the nodes in W as one hot encodings.
- **Skipgram:** Given one hot encoding of all nodes in a random walk, we apply skipgram algorithm from word2vec [2] to generate embeddings for each node.

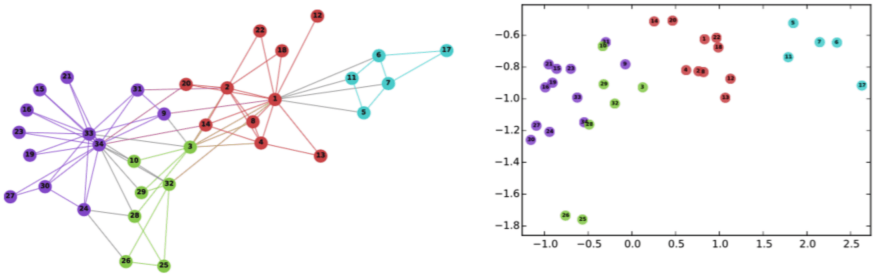

Figure 5: Original graph vs 2D plot of node embeddings

8 Key Insights & Conclusions

DeepWALK create highly effective node embeddings which can achieve state of the art results on variety of tasks. Furthermore, the embeddings encompass properties of node neighbors and graph structure as depicted by figure 5. Furthermore, DeepWALK is a parallelizable algorithm that can run on any graph size. These properties make DeepWALK an effective and desirable solution for creating powerful and semantically meaningful embeddings.

9 Goals Achieved

DeepWALK was proposed to 1) apply deep learning algorithms to generate node embeddings 2) scale the method to any size of graph. It achieved in fulfilling both these goals since the results outperformed most of the benchmark methods and the method can be scaled to any size of graph.

References

[1] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[2] 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*, pages 3111–3119, 2013.

[3] Lei Tang and Huan Liu. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1107–1116. ACM, 2009.

[4] Lei Tang and Huan Liu. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23(3):447–478, 2011.