CS 768: Learning With Graphs **Autumn 2020-2021**

Lecture 12: Deep learning over Graphs

Instructor: Prof. Abir De *Scribe: Anshul Tomar & Anurag Kumar*

In this lecture, we talk about the advantages of deep learning over ”shallow approaches” and an overview of deep learning methods on graphs.

12.1 What is Deep Learning

Given data points $\{(X, y)\}$, using the techniques we have learnt until now(for e.g linear predictor) we cannot model non-linear functions. A workaround for this is the use of proper basis functions ($\phi(\vec{x})$) and kernel functions. The basis functions need not be linear in x but the predictor is linear in the weights which allows it to easily model non-linear functions. For eg, one such formulation can be (here the parameters to be trained are the c_i's:-

$$K(x, x') = \Phi(x)^T \Phi(x') \tag{12.1}$$

$$y(x) = \sum_{i \in X} c_i K(x, x_i) \tag{12.2}$$

Another method is neural networks. We can tell how powerful neural net can be by stating the fact that any Turing machine can be simulated by multilayer perceptron. Also, in a paper in 1999 Universal approximator was introduced which stated that if f‘ is a non linear and continuous function,then we can approximate it as $f \circ h \circ g$ where f and g are polynomial function and h is a non-linear continuous function(e.g. RELU).

12.2 Deep learning with respect to Graphs

Thing learnt till now are

1. LP heuristics :- For supervised learning just take a weighted sum of scores ($w_{AA}.AA + w_{CN}.CN$) and learn for the weights
2. Supervised random walk
3. Collaborative filtering

What is the feature of each node? : Node embeddings are nothing but compression of the graph. For each node we try to compress the $O(|V|)$ sized adjacency vector to a smaller size d being the embedding dimension. So our goal will be to produce task agnostic node embedddings (independent of the task) which will be then converted to task specific node embeddings according to the task.

The node embeddings should be permutation invariant, i.e given the raw features z_u and the embeddings x_u

$$\begin{aligned} x_u &= f(z_1, z_2, ..., z_n) \\ &= f(z_2, z_1, ..., z_n) \end{aligned}$$

or for any other permutation of z_i's. For this to happen the function should be of the form $\Psi(\sum_i \Phi(z_i))$ as given in [1] where Ψ is any real valued function and Φ are the basis functions (this is a necessary and sufficient condition for a function to be a set function).

Given any model to calculate the node embeddings, one would want the model parameters to be independent of $|V|$ and $|E|$. The learning of the model can be done in two ways: -

1. Transductive learning:- Given a new node, we will have to learn the whole model again so as to incorporate the new node (like node2vec where the node embeddings are like model parameters).
2. Inductive learning:- We instead learn a function f which models what the node embeddings should be ($x_u = f(N(u))$) and hence when given a new node we can directly output its embedding based on its neighbours (like GCN, graphSAGE, GNN).

The main applications of node embeddings are Node Classification, Link Prediction and Graph Classification.

References

[1] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. Deep sets, 2018.