Article Title

Frederik Valdemar Schrøder, Jens Petur Tróndarson, Mathias Møller Lybech

Aalborg University

fschra16@student.aau.dk jtrand16@student.aau.dk mlybec16@student.aau.dk

October 20, 2020

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

I. Introduction

Lorem ipsum dolor sit amet, consectetur adipiscing elit [3].

II. Related work

III. Basic Theory

i. Nerual Networks

Nerual networks are a multi layered collection of nodes which have an input layer, hidden layer and an output layer [2]. The input of each node is the output of all nodes in the previous layer. Each of these connections in the network has an individual weight associated to it. To get the value of a node not in the input layer, the output of all the nodes in the previous layer are added togehther with each individual weight along the connection that it travels. All these weighted values are then fed to an aggregation function which result is given to an activation function. The activation function is usually the Sigmoid, Sign or Relu function.

Neural networks can have any number of nodes in the input layer, hidden layers and outputlayer and they do not need to be the same amount. Furthermore it can have any number of hidden layers.

Neural networks have had considerable succes in low-level reasoning where lots of training data is available. They learn by giving the input layer values and then have the network compute an output value. The ouput value is then compared to the real value of the inputs and based on the margin of error we go backwards through the network adjusting each weight. This is called back propagation. After doing this enough times or when the margin of error is lower than a predefined value the network is considered trained and can be tested on new data or be applied to a real world sceneario.

ii. Graph Convolutional Network

Graph Convolutional Network (GCN) is a neural network architecture that operates on graphs. Given a graph $G = (V, E)$ a GCN takes the input of a adjacency matrix A with size of NxN that represents graph G and a feature matrix NxF^0, where N is the total amount

of nodes and F^0 is the total amount of input features for each node. A hidden layer in a GCN can be defined as $H^i = f(H^{i-1}, A)$ where i indicates the layer and H^0 is the previously mentioned NxF^0 feature matrix and f is a propagation function [1]. There are many different types of propagation functions. A simple example could be $f(H^i, A) = \sigma(AH^iW^i) = H^{i+1}$ where W^i is the weight matrix at layer i and σ is a non-linear activation function [1]. The intuition behind this propagation function is that the future representation of each node is calculated based on its neighbors nodes. Because of this, each time i is increased, a node's new value is therefore affected not only by its neighbors but its neighbors' neighbors and so on. An issue with this propagation function could be that the value of each node now is a sum of each of its neighbors, and therefore loses its own value. This could be solved by replacing A with $\hat{A} = A + I$ where I is the identity matrix. Doing this the node considers itself a neighbor.

IV. Method

V. Discussion

VI. Conclusion

References

[1] Tobias Skovgaard Jepsen. *How to do Deep Learning on Graphs with Graph Convolutional Networks*. 2018. URL: https://towardsdatascience.com/how-to-do-deep-learning-on-graphs-with-graph-convolutional-networks-7d2250723780 (visited on 10/14/2020).

[2] David L. Poole and Alan K. Mackworth. "Artificial Intelligence: Foundations of computational agents (second edition)". In: ed. by Cambridge University Press 2017. 2017.

[3] Rex Ying et al. "Graph Convolutional Neural Networks for Web-Scale Recommender Systems". In: KDD '18. 2018.