Advanced Deep Learning TP3

GraphNNs

Romain COLLEDANI
Théo COSTES
Antoine GALLIER

Abstract

This TP aims to deal with GraphNNs. The goal was to implement a model to deal with PPI (protein-protein interaction). Moreover, we had to describe the architecture of our model and explain the difference between utility and similarity attention.

1 Diagram of our model

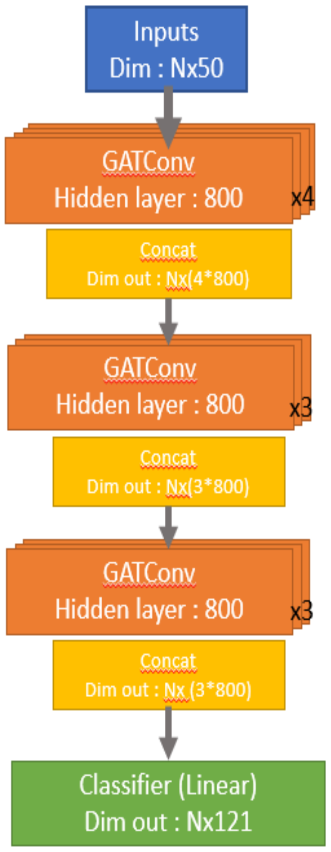

Figure 1 : Architecture of our model

The model above allowed us to reach a score of 0.974.

2 Differentiation between similarity attention and utility

2.1 Similarity attention

The similarity attention is introduced in the article “Attention Is All you need”.

In this paper, researchers use a model architecture that rely entirely on attention mechanism.

The mechanism of similarity attention is based on a simple action: the mapping of a query and a set of key-value pairs to a vector output. The aim of this process is to link queries and values depending on their compatibility (this explains the name of the process: similarity attention). To do that, they use the dot product and SoftMax function, as shown below:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- Q: vector of queries
- K: vector of keys
- V: vector of values
- d_k: dimension ok keys

So, each position of the value vector is compared to all queries, then they assess the similarity between the value position and all queries (helped by the key and SoftMax function). At the end, they obtain a vector of weighted sum of the value, where only query compatible with the value have important weight.

Researchers call their particular attention “Scaled Dot-product attention”. They use dot-product attention instead of additive attention because the latter is much faster and more space-efficient in practice, since it can be implemented using highly optimized matrix multiplication code.

Their model (called “Transformer”) uses multi-head attention in various way :

1) Within their encoder-decoder attention layers, their queries come from the previous decoder layer and the memory keys and values come from the output of the encoder. This allows every positions in the input sequence.

2) As all the keys, values and queries come from the output of the previous layer of the encoder, each position in the encoder can attend to all positions in the previous layer of the encoder.

3) Moreover, self-attention attention layers in the decoder allow each position in the decoder to attend to all positions in the decoder up to and including position.

To Sum up, this mechanism allows us to consider not only the information of one value position, but also the information from query from others positions that are linked to this value.

2.2 Utility

The notion of utility attention is introduced in the article “Graph Attention Networks”.

In this paper, they introduced attention mechanisms as a tool used to deal with variable sized inputs. They developed an attention-based architecture to perform node classification of graph-structured data as they want to compute the hidden representations of each node in the graph.

This mechanism is defined in our case in a “graph environment”. The idea of the utility attention mechanism is to keep from one node features, only the most important feature for this node. To do that, for each node of the graph we follow the process:

- we take all neighbourhood nodes and evaluate the impact of their features with an attention coefficient. It is often computed with a a single-layer feedforward neural network, parametrized by a weight vector a. This gives us the importance of node j’s features to node i.
- Then as we did for the Similarity attention, we use the softmax function to highlight the key features: we compute the normalized attention coefficient.

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\text{LeakyReLU}\left(\vec{\mathbf{a}}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

- Once we have computed all the normalized attention coefficient for a node, we compute the linear combination of the features corresponding to attention coefficient, such that:

$$\vec{h}'_i = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha_{ij}\mathbf{W}\vec{h}_j\right).$$

To stabilize the learning process of self-attention, they found an extending mechanism to employ multi-head attention to be beneficial. Specifically, K independent attention mechanism execute the transformation and their features are concatenated, which gives :

To sum up, the utility attention mechanism has 2 part: a first one where we compute for each nodes the importance of its neighbourhood’s feature, and then a step where we put all in common to upgrade the vector features of the node.