Notes on:
Graph Attention Networks

September 7, 2024

1 Abstract

Graph Attention Networks (GATs) is a novel network architecture that operate on graph-structured data, leveraging masked self-attentional layers to addres the shortcomings of prior methods based on graph convolutions or their approximations. By stacking layers in which nodes are able to attend over their neighborhoods' features, we enable (implicity) specifying different weights to different nodes in a neighborhood. This model is readily applicable to inductive and transductive problems.

2 Introduction

Graph Neural Networks (GNNs) were introduced in 2009, 2005 by Gori and Scarselli as a generalisation of recursive neural networks that can directly deal with a more general class of graphs. A convolution operation is defined in the Fourier domain by computing the eigendecomposition of the graph Laplacian, resulting in potentially intense computations and non-spatially localized filters.

However, spectral methods the learned filters depend on the Laplacian eigenbasis, which depends on the graph structure. Thus, a model trained on a psecific structure can not directly be applied to a graph with a different structure.

Non-spectral approaches which define convolutions directly on the graph, operations on groups of spatially close neighborhoods. One of the challenges of these approaches is to define an operator which works with different sized neighborhoods and maintains the weight sharing property of CNNs.

Inspired by the challenges and advances in recent works, this paper introduces an attention-based architecture to perform node classificication of graph-structured data. The idea is to compute the hidden representations of each node in the graph, by attending over its neighbors, following a self-attention mechanism.

The attention architecture has several interesting properties:

- the operation is efficient, since it is parallelizable across node-neighbor pairs

- it can be applied to graph nodes having different degrees by specifying arbitrary weights to the neighbors.
- the model is directly applicable to inductive learning problems including tasks where the model has to generalise to complete unseen graphs.

3 GAT Architecture

Single graph attention layer, is the sole layer utilised thoughout all the GAT architectures for this project. The input to this layer is a set of node features $\mathbf{h} = \{\vec{h_1}, \vec{h_2}, \ldots, \vec{h_N}\}, \vec{h}_i \in \mathbb{R}^F$, where N is the number of nodes, and F is the number of featues in each node. The layer producs a new set of node features $\mathbf{h}' = \{\vec{h}'_1, \vec{h}'_2, \ldots, \vec{h}'_N\}, \vec{h}'_i \in \mathbb{R}^{F'}$ as its output. In order to obtain sufficient expressive power to transform the input features into higher-level features, at leas one learnable linear transformation is required. A shared linear transformation, parameterised by a *weight matrix*, $\mathbf{W} \in \mathbb{R}^{F \times F'}$, is applied to every node. Then a *self-attention* on the nodes is performed $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \to \mathbb{R}$ computes attention coefficients

$$e_{ij} = a\left(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j\right) \tag{1}$$

Indicating the importance of node j's features to node i. The graph structure is injected into the mechanism by performing masked attention e_{ij} is only computed for nodes $j \in \mathcal{N}_i$ where $\mathcal{N}_i$ is the first-order neighborhood of node i including iteself, in the graph. To make coefficients easily comparable across different nodes, they are normalised using softmax function:

$$\alpha_{ij} = \text{softmax}_j(e_{ij}) = \frac{\exp\left(\text{LeakyReLU}\left(\vec{a}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(\vec{a}^T[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)} \tag{2}$$

The attention mechanism a is a single-layer feedforward neural network, parametrised by a weight vecto $\vec{a} \in \mathbb{R}^{2F'}$ and applying LeakyReLU nonlinearity. The normalised attention coefficients are then used to compute a linear combination of the features corresponding to them, to serve as the final output features for every node:

$$\vec{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j\right) \tag{3}$$

Multi-head attention has been found to be beneficial for the stability of the learning process concatenating K attention mechanisms transformations:

$$\vec{h}'_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}\vec{h}_j\right) \tag{4}$$

Where $\|$ represents concatenation, α_{ij}^k are the normalised attention coefficients computed by the k-th attention mechanism (a^k) and $\mathbf{W}^k$ is the corresponding input linear transoformation's weight matrix. If multi-head attention on the final (prediction) layer of the network is performed, concatenation is no longer sensible, instead, apply averaging and delaying final nonlinearity:

$$\vec{h}_i' = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}_i}\alpha_{ij}\mathbf{W}\vec{h}_j\right) \tag{5}$$