Graph Attention Networks (GAT)

What is GAT?

"Which of my neighbors should I listen to most?"

Instead of treating all neighbors the same, GAT learns to pay different amounts of attention to each one.

Why Graph Attention Networks?

- In classic Graph Convolutional Networks (GCNs), node i aggregates information from its neighbors using fixed edge weights A_{ij}.
- These weights typically indicate whether nodes are connected and are normalized by node degrees.
- This approach assumes all connected neighbors contribute equally, which isn't true in noisy or complex real-world graphs.
- Some neighbors may carry more relevant or trustworthy information than others.
- Graph Attention Networks (GATs) solve this by learning how much attention each neighbor deserves.
- The attention mechanism assigns learnable importance scores between connected nodes.
- This concept is inspired by attention models used in natural language processing and computer vision.
- As a result, GATs adaptively focus on more informative neighbors during message passing.

Feature Transformation (Make Info Comparable)

- Every node has a starting feature vector (like a little summary about itself).
- We use a simple neural network layer (just a matrix W) to turn all node features into a new space.
- **Why?** So that the info we're comparing is the right shape and type to calculate attention!

$$h_i' = W h_i$$

(Where h_i is the original feature of node i; W is a learnable matrix.)

Attention Score: How Much Should I Pay Attention to You?

- For each *neighbor* j of node i, GAT asks: how "relevant" is neighbor j to node i?
- It combines i's and j's features (from the previous step), and passes them through a little function (a small neural network) to get a *score* e_{ij}.

Simple formula:

$$e_{ij} = \text{NeuralNet}(h'_i, h'_j)$$

(Usually, this combines the info, applies weights & a nonlinearity like LeakyReLU.)

Softmax Normalization: Attention Weights

To make these scores comparable, GAT uses softmax so they sum to 1 over neighbors:

$$\alpha_{ij} = \text{Softmax}_j(e_{ij}) = \frac{\exp(e_{ij})}{\sum_{l \in \mathcal{N}(i)} \exp(e_{il})}$$

Now, α_{ij} is the "attention coefficient"—how much node i should listen to j out of all its neighbors.

Calculating e_{ij}: Using Neural Network

In Veličković et al (2018), a is a feedforward neural network (usually a single layer) using:

- Linear transformation (with weights W_2)
- Nonlinear activation (LeakyReLU, with slope α)
- Concatenation of node vectors

The formula is:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(W_2[WH_i^{k-1} \| WH_j^{k-1}]))}{\sum_{l \in \mathcal{N}(i)} \exp(\text{LeakyReLU}(W_2[WH_i^{k-1} \| WH_l^{k-1}]))}$$

Here, $\|$ means "concatenation"—just sticking the two vectors side-by-side.

Weighted Sum (Neighbourhood Aggregation)

- Node i now combines info from all its neighbors. But instead of simply averaging, it takes a **weighted sum** using its attention weights.
- For every neighbor j, multiply its features by α_{ij}. Add them all up. Then apply a nonlinearity (like ReLU).

$$h_i^* = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} h'_j\right)$$

Multi-Head Attention (Look from Multiple Angles)

- GAT sometimes uses multiple attention "heads." Each head does its own attention calculation with its own weights.
- The outputs from all heads are either concatenated or averaged. This lets the network learn different types of "roles" or "relationships".

Concat style: $h_i^* = h_i^{(1)} \| h_i^{(2)} \| \ldots \| h_i^{(K)}$

Average style: $h_i^* = \frac{1}{K} \sum_{k=1}^{K} h_i^{(k)}$

Summary

Feature transformation: Make info comparable.

Calculate attention score: How much to listen to each neighbor?

Softmax attention: Turn scores into weights (between 0 and 1).

Weighted sum: Build new features by "focusing" on important neighbors.

Multi-head: "See" from different perspectives, combine results.

Quiz 1

What problem does the GAT attention mechanism solve that standard GCNs struggle with?

A. Overfitting on small graphs

B. Assigning different importance to different neighbors

C. Speeding up matrix multiplication

D. Reducing the number of hidden layers

Quiz 2

In GAT, what function is often used to normalize raw attention scores into a probability distribution over a node's neighbors?

A. Sigmoid

B. Softmax

C. Tanh

D. ReLU

Quiz 3

What is the purpose of multi-head attention in GAT?

A. To combine node features with edge features

B. To learn several different types of attention weights and aggregate their results

C. To increase the number of graph layers

D. To make the computation faster

Quiz 4

Which operation is used to combine the outputs from multiple attention heads in the final layer of a GAT?

A. Concatenation

B. Average pooling

C. Maximum pooling

D. Element-wise multiplication

Quiz 5

In the formula for GAT attention coefficients, what does the neural network do with node feature vectors before applying softmax?

A. Adds the vectors

B. Concatenates them and applies Leaky ReLU

C. Multiplies them

D. Divides by the node degree