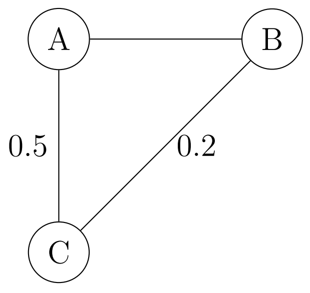
Worksheet 4: Graph Neural Networks

Page 1: Setup

Suppose we have three weather stations:

- A = Athlone
- B = Belfast
- C = Cork

At each station we have temperature (°C) and wind speed (m/s). We represent this as a graph:



We have data at time t and learn to predict, e.g. temperature at time $t + \Delta t$. Define

$$X_t = [\text{Temp, Wind speed}] \quad \text{at time } t \text{ for the three stations.}$$

A = adjacency matrix. D = degree matrix = number of connections for each node + self loop. W = layer weights. b = biases. W_{out} = output weights. b_{out} = output biases.

Page 2: Matrix Example

Suppose

$$X = \begin{bmatrix} 10 & 4 \\ 12 & 6 \\ 8 & 5 \end{bmatrix}, \quad \text{(3 stations} \times \text{2 variables)}$$

Adjacency with self-loops:

$$\tilde{A} = A + I = \begin{bmatrix} 1 & 1 & 0.5 \\ 1 & 1 & 0 \\ 0.5 & 0.2 & 1 \end{bmatrix}.$$

Degree matrix:

$$D = \operatorname{diag}(2.5, 2.2, 1.7).$$

Normalised adjacency:

$$\hat{A} = D^{-\frac{1}{2}} \tilde{A} D^{-\frac{1}{2}} \approx \begin{bmatrix} 0.4 & 0.426 & 0.243 \\ 0.426 & 0.455 & 0.103 \\ 0.243 & 0.103 & 0.588 \end{bmatrix}.$$

Compute

$$S = XW, \quad M = \hat{A}S.$$

Then

$$H = \operatorname{ReLU}(M + b) = \begin{bmatrix} 11 & 0 \\ 10.323 & 0 \\ 8.818 & 0 \end{bmatrix}.$$

Page 3: Node Embeddings

The rows of H are node embeddings. Then compute the output:

$$\hat{y} = HW_{out} + b_{out}.$$

Example:

$$\hat{y} = \begin{bmatrix} 11 & 0 \\ 10.323 & 0 \\ 8.818 & 0 \end{bmatrix} \begin{bmatrix} 0.4 \\ 0.2 \end{bmatrix} + 0.05 = \begin{bmatrix} 4.45 \\ 4.18 \\ 3.58 \end{bmatrix}.$$

Notes

- Can predict different numbers of targets (e.g. temperature and wind, not just temperature).
- Can extend GCNs by stacking linear steps into multiple layers.

Extension with Transformers

Let

$$X_t \in \mathbb{R}^{N \times F} \quad \text{(e.g. 3 stations} \times \text{2 features).}$$

Still have adjacency matrix $\hat{A}$. Now compute

$$H_t = \sigma(\hat{A}(X_t W + b)).$$

So H_t are node embeddings. Do this for each timestep and stack:

$$H^{(i)} = [h_1^{(i)}, \dots, h_T^{(i)}].$$

Add temporal positional encoding. Add a multi-head attention transformer.

Page 4: Additional Techniques in AIFS

- Use a clever mask for attention weights. For station i,

  $$\text{mask}_{ij} = \begin{cases} 0, & j \in W(i) \\ -\infty, & j \notin W(i) \end{cases}$$

  where $W(i)$ is a latitude window for station i.

  Attention:

  $$\text{Att}_{ij} = \frac{\exp\left(\frac{qk^T}{\sqrt{d}} + \text{mask}_{ij}\right)}{\sum_k \exp\left(\frac{qk^T}{\sqrt{d}} + \text{mask}_{ik}\right)}.$$

  So

  $$\hat{A}S = \text{softmax}\left(\frac{2W_q(2W_k)^T}{\sqrt{d}}\right) W_v.$$

- GELU activations $\Rightarrow$ smoother gradients, no kink at zero, better for deeper models.