Matthieu Boyer
matthieu.boyer@ens.fr

Lab session # 7
ALTEGRAD 2023

01/01/80

1 Question 1

1. For a single attention head $k \in \{1, \dots, K\}$, the output is computed as
$$z_i^{(t+1,k)} = \sum_{j \in \mathcal{N}_i} \alpha_{ij}^{(t+1,k)} W^{(k)} z_j^{(t)},$$
where the attention coefficients for head k are
$$\alpha_{ij}^{(t+1,k)} = \frac{\exp\left(\text{LeakyReLU}\left(a^{(k)\top}[W^{(k)}z_i^{(t)} \| W^{(k)}z_j^{(t)}]\right)\right)}{\sum_{\ell \in \mathcal{N}_i} \exp\left(\text{LeakyReLU}\left(a^{(k)\top}[W^{(k)}z_i^{(t)} \| W^{(k)}z_\ell^{(t)}]\right)\right)}.$$
Here, $W^{(k)} \in \mathbb{R}^{F_{in} \times F'_{out}}$ is the learnable weight matrix for head k, and $a^{(k)} \in \mathbb{R}^{2F'_{out}}$ is the learnable attention vector for head k.
2. The final node representation $z_i^{(t+1)}$ is obtained by concatenating the outputs of all K heads
$$z_i^{(t+1)} = \|_{k=1}^{K} z_i^{(t+1,k)} = [z_i^{(t+1,1)} \| z_i^{(t+1,2)} \| \cdots \| z_i^{(t+1,K)}].$$
The total dimensionality of $z_i^{(t+1)}$ is
$$\dim(z_i^{(t+1)}) = K \times F'_{out}.$$
Since each head produces an output of dimension F'_{out}, concatenating K heads results in a $K \cdot F'_{out}$ dimensional vector.
3. For each head k, we have
   - Weight matrix $W^{(k)}$: $F_{in} \times F'_{out}$ parameters;
   - Attention vector $a^{(k)}$: $2F'_{out}$ parameters.

   This gives $F_{in} \cdot F'_{out} + 2F'_{out}$ parameters per head in total. Then, for all K heads, the total number of learnable parameters is
$$\text{Total parameters} = K \times (F_{in} \cdot F'_{out} + 2F'_{out}) = K \cdot F'_{out} \cdot (F_{in} + 2).$$

2 Task 3

Test set results: `loss` $= 0.0006$, `accuracy` $= 1.0000$

3 Question 2

Suppose all nodes have identical feature vectors: $x_i = c$ for all $v_i \in V$, where $c \in \mathbb{R}^d$ is a constant vector.

1. For any edge (i, j), the unnormalized attention score is
$$e_{ij} = \text{LeakyReLU}\left(a^\top [Wz_i \| Wz_j]\right).$$
Since $z_i = z_j = c$ for all nodes (identical features), we have
$$e_{ij} = \text{LeakyReLU}\left(a^\top [Wc \| Wc]\right).$$
Let $a = [a_1 \| a_2]$ where $a_1, a_2 \in \mathbb{R}^h$ (splitting the attention vector), then
$$e_{ij} = \text{LeakyReLU}\left(a_1^\top Wc + a_2^\top Wc\right) = \text{LeakyReLU}\left((a_1 + a_2)^\top Wc\right).$$
This value is constant for all edges, independent of i and j. Let us denote this constant as $e_{ij} = \beta$ for all $(i, j) \in E$. The normalized attention coefficient becomes
$$\alpha_{ij} = \frac{\exp(\beta)}{\sum_{k \in \mathcal{N}_i} \exp(\beta)} = \frac{\exp(\beta)}{|\mathcal{N}_i| \cdot \exp(\beta)} = \frac{1}{|\mathcal{N}_i|},$$
where $|\mathcal{N}_i|$ is the degree of node v_i.

2. When all nodes have identical features, the GAT layer loses its attention mechanism entirely. The attention coefficients $\alpha_{ij} = \frac{1}{|\mathcal{N}_i|}$ are uniform across all neighbors and depend only on the degree of node i. This degenerates into the standard mean aggregation or degree-normalized propagation rule

$$z_i^{(t+1)} = \sum_{j \in \mathcal{N}_i} \alpha_{ij} W z_j^{(t)} = \sum_{j \in \mathcal{N}_i} \frac{1}{|\mathcal{N}_i|} W c = W c.$$

   Since $z_j^{(t)} = c$ for all j, all nodes receive the same aggregated message Wc. This is equivalent to a simple Graph Convolutional Network (GCN) layer with degree normalization $\tilde{D}^{-1}\tilde{A}XW$, where the attention mechanism provides no selectivity.

3. Despite the loss of feature information and attention mechanism, the model can still classify nodes better than random guessing on the Karate network because of the following properties

   - Structural information: The GAT layers still perform message passing over the graph structure. Even with identical features, the aggregation process encodes topological information such as node degrees and neighborhood connectivity patterns;
   - Layer composition: Multiple GAT layers allow nodes to aggregate information from multi-hop neighborhoods. Nodes in different positions in the graph will have different k-hop neighborhood structures, leading to different final representations even starting from identical features;
   - Nonlinear transformations: The ReLU activations, dropout, and fully-connected layers introduce nonlinearity and learnable transformations that can exploit structural patterns;
   - Community structure: The Karate network has a clear community structure. Nodes with similar structural roles (e.g., central vs. peripheral, bridge nodes) will accumulate different patterns of information through repeated message passing, enabling the model to learn meaningful class boundaries based purely on graph topology.

   In essence, the model seems to learn to classify based on structural node roles rather than node features, which is sufficient for the Karate network where community membership correlates strongly with network position.

4 Task 4

We found the weight graph of Figure 1

5 Question 3

To introduce conditions into the Variational Graph Autoencoder (VGAE), we can extend the model to a Conditional Variational Graph Autoencoder (CVGAE) by conditioning both the encoder and decoder on additional information c (such as graph properties, labels, or structural constraints).
The conditional encoder then learns a distribution $q_\phi(z|G, c)$ instead of $q_\phi(z|G)$. We modify the encoding process to incorporate the condition

$$\mu = \text{GNN}_\mu(X, A, c)$$
$$\log \sigma = \text{GNN}_\sigma(X, A, c)$$

The condition c can be integrated by one method of the following:

- Concatenation: Append the condition vector to the graph-level representation before computing μ and σ: $h'_G = [h_G \| c]$;
- Feature augmentation: Concatenate c to the node feature matrix: $X' = [X \| \mathbf{1} \otimes c]$, where $\mathbf{1}$ is a vector of ones;
- Conditional layers: Use condition-dependent transformations in the GNN layers.

The conditional decoder then models $p_\theta(G|z, c)$ instead of $p_\theta(G|z)$. We modify the reconstruction to

$$\hat{A} = \text{MLP}([z \| c]),$$

where the condition c is concatenated with the latent representation z before feeding into the decoder.

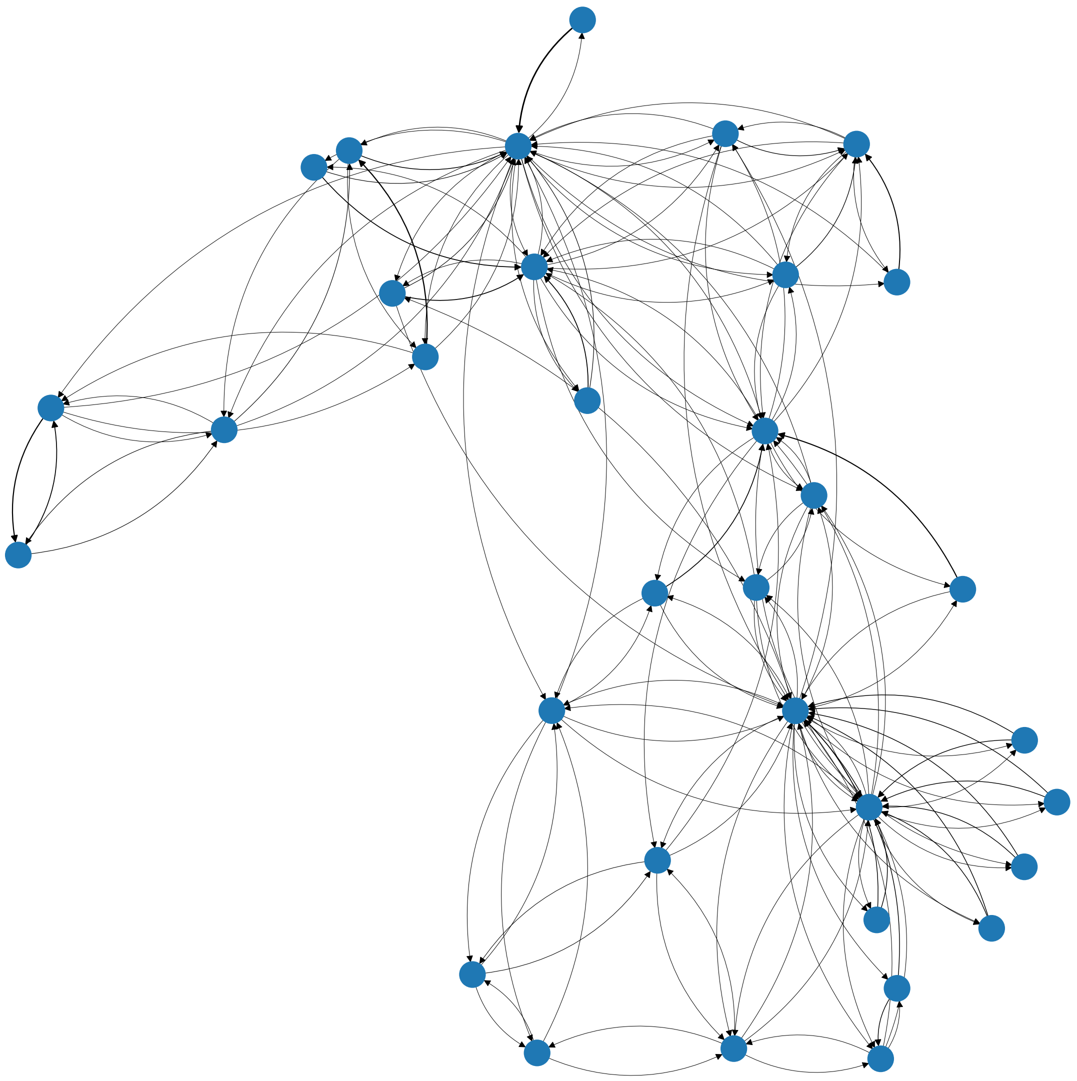


Figure 1: Graph of attention weights for Task 4

The variational lower bound (ELBO) becomes

$$\mathcal{L} = \mathbb{E}_{q_\phi(z|G,c)}[\log p_\theta(G|z,c)] - \mathrm{KL}(q_\phi(z|G,c)\|p(z|c)),$$

where we can either use a standard prior $p(z|c) = \mathcal{N}(0, I)$ or learn a conditional prior.
This allows control over generation properties such as number of nodes, number of communities, edge density, or specific structural patterns in the generated graphs.

6 Question 4

The reparameterization trick $z = \mu + \sigma \odot \epsilon$ (where $\epsilon \sim \mathcal{N}(0, 1)$) is necessary to enable gradient-based optimization through backpropagation when training variational autoencoders.
The key issue is that sampling operations are not differentiable. If we directly sample $z \sim \mathcal{N}(\mu, \sigma^2)$, the sampling operation introduces a stochastic node in the computational graph that blocks gradient flow. Specifically, we cannot compute

$$\frac{\partial z}{\partial \mu} \quad \text{and} \quad \frac{\partial z}{\partial \sigma},$$

when z is obtained through direct sampling, because the sampling operation is a non-deterministic process.
By expressing $z = \mu + \sigma \odot \epsilon$ where $\epsilon \sim \mathcal{N}(0, 1)$, we separate the stochastic component (ϵ) from the learnable parameters (μ and σ). Now,

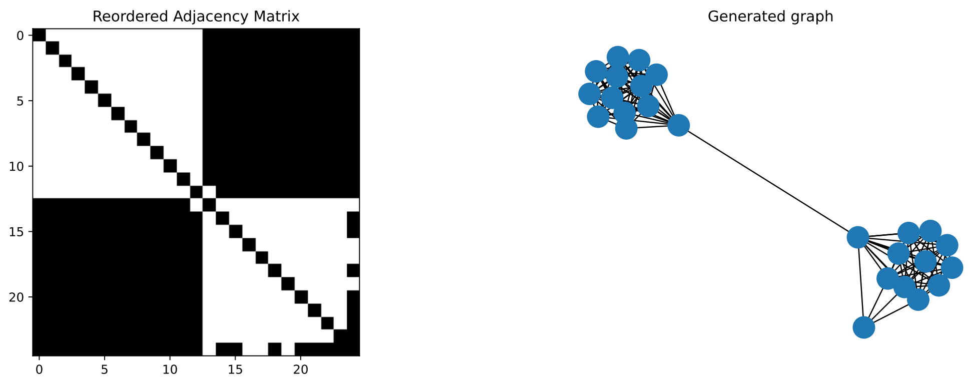
$$\frac{\partial z}{\partial \mu} = 1$$
$$\frac{\partial z}{\partial \sigma} = \epsilon$$

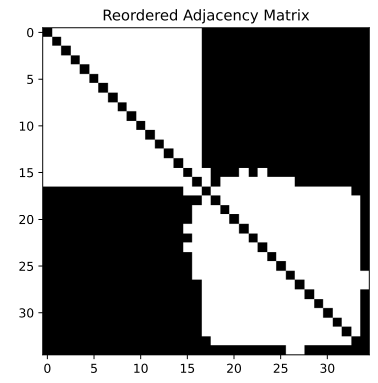
The randomness is isolated in ϵ, which is sampled independently of the model parameters. The operations involving μ and σ are deterministic and differentiable, allowing gradients to flow through them during backpropagation.
If the model directly sampled $z \sim \mathcal{N}(\mu, \sigma^2 I)$ without reparameterization:

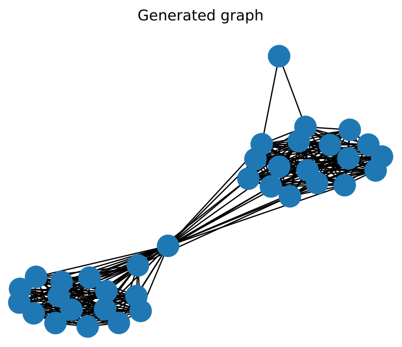
- The gradient $\nabla_{\mu,\sigma}\mathcal{L}$ would be undefined or zero, preventing the encoder from learning;
- We could not optimize the encoder parameters ϕ that produce μ and σ;
- The model would fail to train properly, as the loss signal cannot propagate back through the sampling operation to update the encoder weights;
- Only the decoder could potentially learn (from the reconstruction loss), but the encoder would remain untrainable.

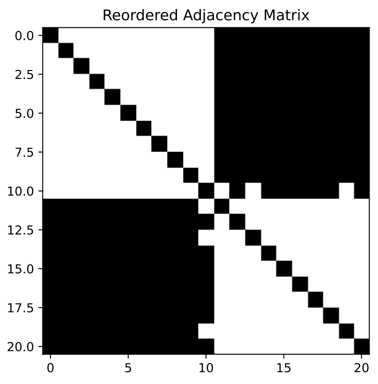
In essence, without the reparameterization trick, the variational autoencoder cannot be trained end-to-end using standard gradient descent methods.

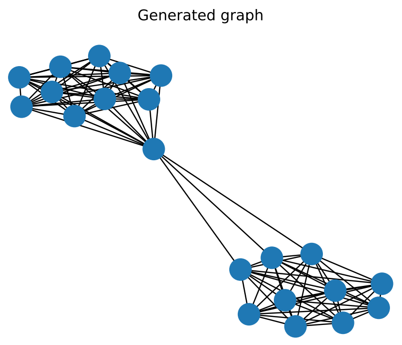
7 Task 8

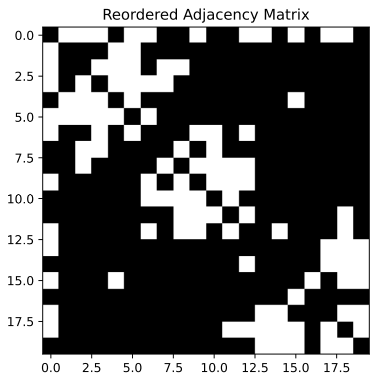


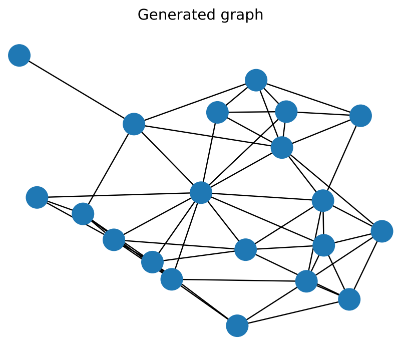

Reordered Adjacency Matrix
0
5
10
15
20
25
30

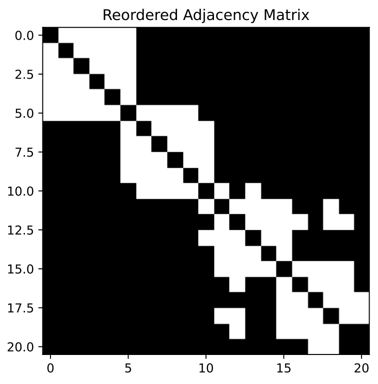

Generated graph

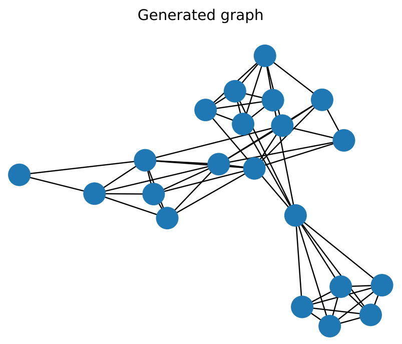

Reordered Adjacency Matrix
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
5
10
15
20


Generated graph


Reordered Adjacency Matrix
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Generated graph


Reordered Adjacency Matrix
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0
5
10
15
20


Generated graph