CS246 Exam 2025 — Question 8: Graph Neural Networks (18 points)

Part 1 (2 points) — Node2Vec with p = 1, q → ∞

Question: What structural information do the embeddings capture?

When q → ∞, the probability of moving *farther away* becomes 1/q → 0. With p = 1, returning and staying at the same distance have normal probability. Only outward exploration is suppressed.

Answer: This produces **BFS-like exploration** — the walker stays within the local neighborhood. The embeddings capture **local community structure**: nodes sharing similar immediate neighborhoods get similar embeddings.

Part 2 (2 points) — Computational Bottleneck in Node2Vec

Question: The denominator Σ_{n∈V} exp(z_u⊤z_n) costs O(|V|) per term. How to mitigate?

Answer: Use **negative sampling**. Replace the full softmax with:

$$\log \sigma(z_u^\top z_v) - \sum_{i=1}^{k} \log \sigma(z_u^\top z_{n_i})$$

where k random "negative" nodes are sampled (typically k = 5–20). This reduces per-term cost from O(|V|) to O(k).

Part 3 (2 points) — Removing the Normalization Term

Question: What happens to embeddings if we drop the denominator? Be specific.

Answer: Without normalization, the objective becomes simply maximizing Σ z_u⊤z_v, which can grow without bound. **All embeddings collapse to the same direction and grow unboundedly in magnitude** — the model has no penalty for making unrelated nodes' dot products large, so it makes everything large and similar. All contrast between nodes is destroyed.

Part 4 (4 points) — Two Issues with the GNN Update Rule

$$h_v^{(l+1)} = \sigma\left(W_l \sum_{u \in N(v)} h_u^{(l)}\right)$$

Issue 1: The node's own information is lost. The sum is only over neighbors N(v) — node v's own embedding h_v^(l) is never used. The model cannot distinguish a node from a different node with identical neighbors.

Fix: Add a self-connection term: $h_v^{(l+1)} = \sigma\left(W_l \sum_{u \in N(v)} h_u^{(l)} + B_l h_v^{(l)}\right)$

Issue 2: No normalization by degree. High-degree nodes produce large aggregated values, low-degree nodes produce small values, regardless of content. The embedding scale is dominated by degree rather than meaningful structure.

Fix: Normalize by degree: divide the sum by |N(v)|, or use symmetric normalization $1/\sqrt{(|N(u)| \cdot |N(v)|)}$ as in GCN.

Part 5a (2 points) — Bipartite Components

Adjacency matrix edge list: {1–4, 1–5, 1–6, 2–6, 3–4}.

2-coloring the graph:

- Color A: nodes 1, 2, 3
- Color B: nodes 4, 5, 6

All edges connect A↔B: 1(A)–4(B) ✓, 1(A)–5(B) ✓, 1(A)–6(B) ✓, 2(A)–6(B) ✓, 3(A)–4(B) ✓

Answer:

- Component 1: **{1, 2, 3}**
- Component 2: **{4, 5, 6}**

Part 5b (2 points) — Improving Message Passing in a Sparse Graph

Answer: Add a **virtual node connected to all existing nodes**. This ensures every pair of nodes is at most 2 hops apart, dramatically improving information flow during message passing.

Alternative: add virtual edges connecting 2-hop neighbors (use $A + A^2$ as the adjacency). For a bipartite graph this is especially useful since it creates intra-partition connections.

Part 6 (4 points) — Schema Graph

Tables:

- Students (*StudentID*, Name, Age, Major)
- Courses (*CourseID*, Title, Department, Credits)

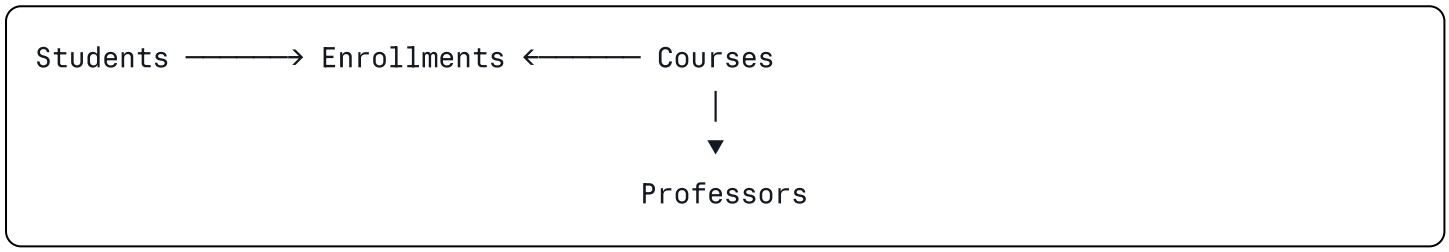
- Enrollments (*EnrollmentID*, StudentID, CourseID, EnrollmentDate, Grade)
- Professors (*ProfessorID*, Name, Department, CourseID)

Rule: Draw an edge from table A to table B whenever A's primary key appears as a foreign key in B.

Foreign key relationships:

1. StudentID (Students' PK) appears in Enrollments → **Students → Enrollments**
2. CourseID (Courses' PK) appears in Enrollments → **Courses → Enrollments**
3. CourseID (Courses' PK) appears in Professors → **Courses → Professors**

Schema Graph:



Three directed edges: Students → Enrollments, Courses → Enrollments, Courses → Professors.