Graph Condensation Problem Set

CSCI 4140 - Machine Learning and Optimization

December 2, 2024

Problem 1: Theoretical Foundations

Consider a graph G with N nodes, E edges, and F-dimensional node features.

Part (a)

Write out the mathematical formulation for the gradient matching objective in graph condensation. Explain each term in the equation.

Part (b)

Why is gradient matching preferred over directly optimizing the bi-level objective? Discuss the computational implications.

Part (c)

In graph condensation, we model the condensed adjacency matrix A' as a function of condensed node features X'. Write out this relationship and explain why this parameterization is beneficial compared to treating A' as free parameters.

Part (d)

Calculate the number of parameters needed to represent:

- A condensed graph with free parameters for both A' and X'
- A condensed graph using the functional relationship $A' = g_\Phi(X')$

Express your answer in terms of n (number of condensed nodes), F (feature dimension), and h (hidden dimension of Φ).

Part (e)

Prove that if the original graph G is undirected, the functional form used for $A' = g_\Phi(X')$ in the paper maintains symmetry.

Solutions to Problem 1

Solution to Part (a)

The gradient matching objective is:

$$\min_{S} \left[\sum_{t=0}^{T-1} D(\nabla_\theta \mathcal{L}(GNN_{\theta_t}(A', X'), Y'), \nabla_\theta \mathcal{L}(GNN_{\theta_t}(A, X), Y)) \right] \tag{1}$$

Term Explanations:

- $S = \{A', X', Y'\}$: The condensed graph parameters being optimized
- T: Number of training steps to match
- D: Distance function between gradient vectors
- $\nabla_\theta \mathcal{L}$: Gradients of loss with respect to GNN parameters
- GNN_{θ_t}: GNN with parameters at step t

Solution to Part (b)

Advantages of Gradient Matching:

1. **Computational Efficiency**

- Bi-level optimization requires solving the complete inner optimization loop for each outer step
- Gradient matching only requires one step of gradient computation
- Avoids repeated full training of GNN models

2. **Memory Benefits**

- Bi-level optimization must store entire optimization trajectory
- Gradient matching only needs current step gradients
- Significantly reduced memory footprint

3. **Parallelization Opportunities**

- Gradient computations can be parallelized across multiple GPUs
- Inner loop optimization in bi-level approach is inherently sequential
- Better hardware utilization

4. **Training Stability**

- More stable optimization process
- Direct alignment of training trajectories
- Better convergence properties

Solution to Part (c)

The adjacency matrix A' is modeled as:

$$A'_{ij} = \sigma\left(\frac{\text{MLP}_\Phi([x'_i; x'_j]) + \text{MLP}_\Phi([x'_j; x'_i])}{2}\right) \tag{2}$$

where σ is the sigmoid function and $[;]$ denotes concatenation.

Benefits of this Parameterization:

1. **Parameter Efficiency**

- Free parameters: $O(n^2)$ for adjacency matrix
- Functional form: $O(nF + h^2)$ where h is MLP hidden size
- Significant reduction in parameter count

2. **Structural Properties**

- Automatically maintains symmetry for undirected graphs
- Ensures edge weights are in [0,1] through sigmoid
- Captures node feature similarities naturally

3. **Generalization Capabilities**

- Can generalize to different graph sizes
- Learns feature-based edge formation rules
- Better transfer to unseen graphs

Solution to Part (d)

1. Free Parameters Approach:

$$\begin{aligned} \text{A' parameters} &= n \times n \text{ (adjacency matrix)} \\ \text{X' parameters} &= n \times F \text{ (node features)} \\ \text{Total} &= n^2 + nF \text{ parameters} \end{aligned}$$

2. Functional Relationship Approach:

$$\begin{aligned} \text{X' parameters} &= n \times F \text{ (node features)} \\ \text{MLP parameters} &= (2F \times h) + (h \times h) + (h \times 1) \text{ (network)} \\ \text{Total} &= nF + 2Fh + h^2 + h \text{ parameters} \end{aligned}$$

Numerical Example ($n = 100$, $F = 128$, $h = 256$):

- Free parameters: $100^2 + 100 \times 128 = 22,800$
- Functional: $12,800 + 65,536 + 65,536 + 256 = 144,128$

Solution to Part (e)

Symmetry Proof:

We need to show $A'_{ij} = A'_{ji}$ for all i, j.

Given:

$$A'_{ij} = \sigma\left(\frac{\text{MLP}_\Phi([x'_i; x'_j]) + \text{MLP}_\Phi([x'_j; x'_i])}{2}\right) \tag{3}$$

For A'_{ji}:

$$\begin{aligned} A'_{ji} &= \sigma\left(\frac{\text{MLP}_\Phi([x'_j; x'_i]) + \text{MLP}_\Phi([x'_i; x'_j])}{2}\right) \\ &= \sigma\left(\frac{\text{MLP}_\Phi([x'_i; x'_j]) + \text{MLP}_\Phi([x'_j; x'_i])}{2}\right) \text{ (by commutativity)} \\ &= A'_{ij} \end{aligned}$$

Key Points:

- The symmetry is enforced by the averaging operation
- The order of feature concatenation doesn't affect the final result
- Sigmoid function preserves the symmetry property

Therefore, A' is symmetric by construction, making it suitable for undirected graphs.