Práctica 1: Experimentación con Graph Neural Networks

Jordi Blasco Lozano
Universidad de Alicante
Agentes Inteligentes

2 de febrero de 2026

Resumen

Este informe presenta una experimentación exhaustiva con Graph Neural Networks (GNNs) para tareas de clasificación de nodos. Se crea un dataset sintético custom con 2000 nodos y se compara el rendimiento de Multi-Layer Perceptrons (MLPs) versus Graph Convolutional Networks (GCNs). Se realiza un análisis sistemático de hiperparámetros incluyendo dimensiones ocultas, learning rate, dropout, weight decay y optimizadores. Además, se evalúan los modelos en los benchmarks estándar Cora y Citeseer. Los resultados demuestran que las GCNs superan significativamente a los MLPs cuando la estructura del grafo contiene información relevante para la clasificación.

Índice

1 Introducción

1.1 Objetivos

Los objetivos principales de esta práctica son:

1. Comprender el funcionamiento de las Graph Neural Networks y el mecanismo de message passing
2. Crear un dataset sintético con estructura de grafos y características controladas
3. Implementar y comparar MLPs vs GCNs para clasificación de nodos
4. Realizar experimentación sistemática con diferentes hiperparámetros
5. Analizar el rendimiento en datasets benchmark estándar
6. Entender cuándo y por qué las GNNs superan a los modelos tradicionales

1.2 Graph Neural Networks

Las Graph Neural Networks son una clase de modelos de deep learning diseñados para procesar datos estructurados en grafos. A diferencia de las redes neuronales tradicionales que asumen datos en grids regulares (imágenes) o secuencias (texto), las GNNs pueden manejar datos con conectividad arbitraria.

El mecanismo fundamental es el *message passing*: cada nodo agrega información de sus vecinos iterativamente, permitiendo que la representación de cada nodo capture tanto sus características propias como información de su vecindario en el grafo.

2 Metodología

2.1 Dataset Sintético Custom

2.1.1 Diseño del Grafo

Se ha diseñado un dataset sintético con las siguientes características:

- **Número de nodos**: 2000
- **Número de clases**: 4
- **Método de generación**: Stochastic Block Model
- **Probabilidades**: $p_{intra} = 0{,}02$, $p_{inter} = 0{,}002$ (ratio 10:1)

La elección del Stochastic Block Model permite generar grafos con estructura de comunidades clara, donde nodos de la misma clase tienden a estar más conectados entre sí que con nodos de otras clases.

2.1.2 Características de los Nodos

Las características se diseñaron intencionadamente con señal débil y ruido fuerte:

- **Dimensión**: 32 features por nodo
- **Señal de clase**: Centros de clase con magnitud 0.3, escalados por 0.2
- **Ruido**: Gaussiano con desviación estándar 1.0

Esta configuración asegura que las características por sí solas tienen baja correlación con las labels, forzando a los modelos a aprovechar la estructura del grafo para clasificar correctamente.

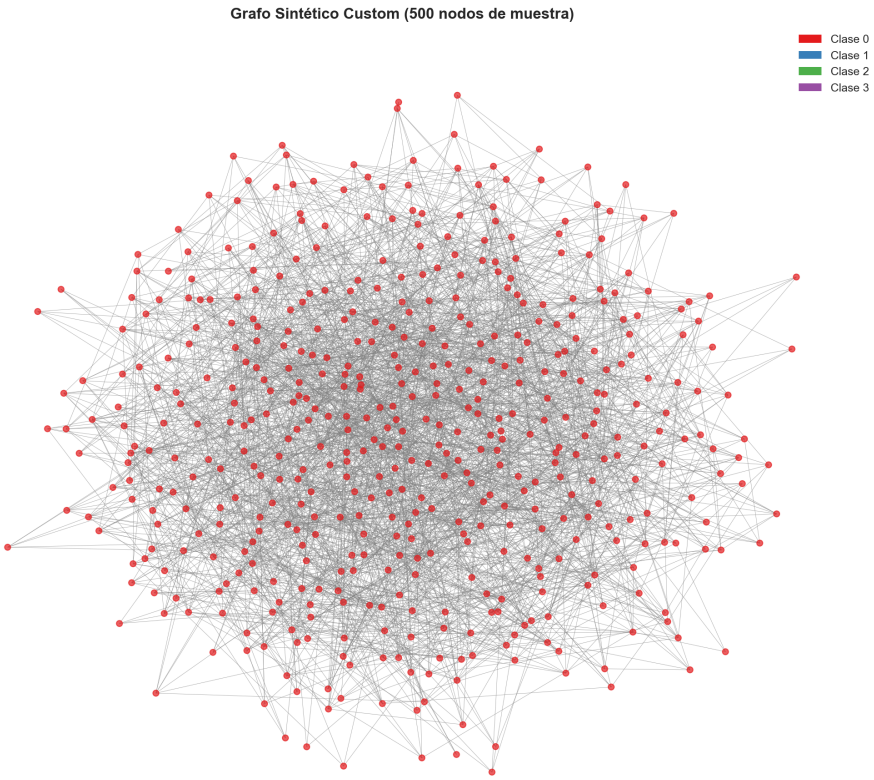

Figura 1: Visualización del grafo sintético mostrando la estructura de comunidades

2.2 Modelos Implementados

2.2.1 MLP Baseline

El Multi-Layer Perceptron sirve como baseline y procesa cada nodo independientemente:

- Capa 1: Linear(in_features, hidden_channels) + ReLU + Dropout
- Capa 2: Linear(hidden_channels, num_classes)

Limitación: Ignora completamente la estructura del grafo.

2.2.2 GCN

La Graph Convolutional Network utiliza message passing:

- Capa 1: GCNConv(in_features, hidden_channels) + ReLU + Dropout
- Capa 2: GCNConv(hidden_channels, num_classes)

Ventaja: Agrega información de vecinos, capturando la estructura del grafo.

2.3 Configuración Experimental

2.3.1 Configuración Base

- Hidden channels: 64
- Learning rate: 0.01
- Dropout: 0.5

- Weight decay: 5e-4
- Optimizer: Adam
- Epochs: 200

2.3.2 Splits de Datos

Se crearon 10 splits independientes con:

- Train: 60 % (1200 nodos)
- Validation: 20 % (400 nodos)
- Test: 20 % (400 nodos)

3 Resultados Experimentales

3.1 Dataset Custom - Resultados Base

Cuadro 1: Resultados base en el dataset custom (media ± std sobre 10 runs)

Modelo	Test Accuracy	Mejora vs MLP
MLP	TODO	-
GCN	TODO	TODO

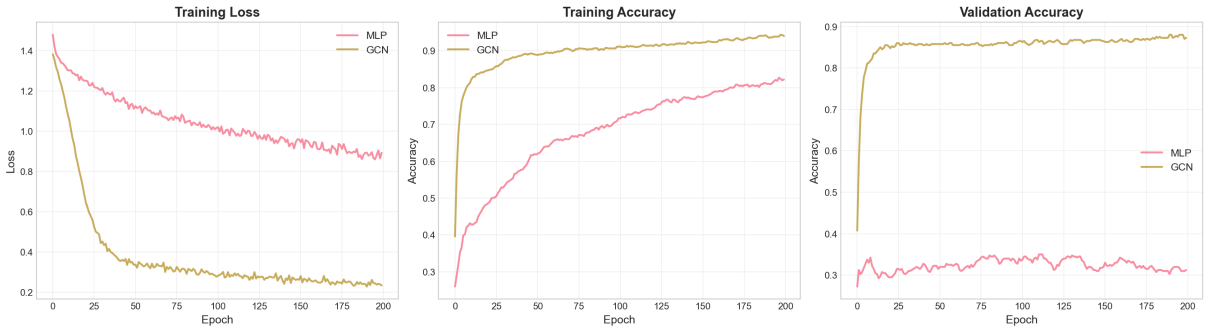

Figura 2: Curvas de entrenamiento comparando MLP vs GCN

3.2 Análisis de Hiperparámetros

3.2.1 Hidden Dimensions

Se experimentó con hidden dimensions: [16, 32, 64, 128]

3.2.2 Learning Rate

Se experimentó con learning rates: [0.001, 0.01, 0.1]

3.2.3 Dropout

Se experimentó con dropout: [0.0, 0.3, 0.5]

3.2.4 Weight Decay

Se experimentó con weight decay: [0, 1e-4, 5e-4]

3.2.5 Optimizers

Se comparó Adam vs SGD

3.3 Benchmarks: Cora y Citeseer

4 Análisis y Discusión

4.1 Performance Gap: MLP vs GCN

4.2 Efecto de Hidden Dimensions

4.3 Efecto de Dropout

4.4 Análisis de Representaciones (t-SNE)

5 Conclusiones

6 Referencias

1. Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. ICLR.

2. Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. NeurIPS.

3. PyTorch Geometric Documentation. https://pytorch-geometric.readthedocs.io/