Data Mining

Homework 4

Due: 29/12/2024, 23:59

Instructions

You must hand in the homeworks electronically and before the due date and time.

The first homework has to be done by each **person individually**.

Handing in: You must hand in the homeworks by the due date and time by an email to Gianluca (`decarlo@diag.uniroma1.it`) that will contain as attachment **(not links to some file-uploading server!)** a .zip file with your answers. The filename of the attachment should be
`DM_Homework_4__StudentID_StudentName_StudentLastname.zip`;
for example:
`DM_Homework_4__1235711_Robert_Anthony_De_Niro.zip`.
The email subject should be
`[Data Mining] Homework_4 StudentID StudentName StudentLastname`;
For example:
`[Data Mining] Homework_4 1235711 Robert Anthony De Niro`.
After you submit, you will receive an acknowledgement email that your project has been received and at what date and time. If you have not received an acknowledgement email within 2 days after the deadline then contact Gianluca.

The solutions for the theoretical exercises must contain your answers either typed up or hand written clearly and scanned.

If you have any questions, feel free to first email Gianluca: `decarlo@diag.uniroma1.it`

For information about collaboration, and about being late check the web page.

Problem 1. The goal of this exercise is to apply Graph Neural Networks (GNNs) as well as other node embedding approaches to a node classification task, evaluate their generalization through cross-dataset testing, extend their application to link prediction, analyze graph embeddings, and explore model explainability.

We will use the **Cora** dataset, along with other datasets such as **CiteSeer** and **PubMed**, available in the PyTorch Geometric Planetoid dataset library. These datasets provide diverse graph structures and node feature distributions to test the generalization and adaptability of GNNs.

In addition, we will analyze the graph embeddings generated by the GNN and apply explainability techniques to understand the model's predictions.

Graph Neural Networks (GNNs).

1. **Node Classification on Cora:**
   Implement a GNN for node classification on the Cora dataset. Use a model architecture like GCN, GraphSAGE, or GAT, available in the PyTorch Geometric convolutional layers. Define an appropriate architecture, including layers, activation functions, and an output layer for multi-class classification.

2. **Training and Evaluation:**
   Train the GNN model on the Cora training set and perform hyperparameter tuning using the

validation set. Evaluate the model on the Cora test set and report metrics such as accuracy, precision, recall, and F1-score.

3. **Saving the Model:**
   Save the trained model for use in cross-dataset testing.

Testing on CiteSeer and PubMed. Load the CiteSeer and PubMed datasets. Use the saved GNN model trained on Cora to perform inference directly on these datasets without retraining. Evaluate the model's performance on the test sets of CiteSeer and PubMed using the same metrics (accuracy, precision, recall, and F1-score). Discuss the performance differences observed across datasets. Address the following questions:

- How well does the model generalize to the new datasets?
- Are there datasets where the model performs acceptably? Why might this be the case (e.g., similarities in graph structure, feature distributions, or task objectives)?
- What factors contribute to poor generalization on some datasets?

Link Prediction. Modify your GNN model to perform link prediction on the Cora dataset. The goal is to predict whether an edge exists between two nodes based on their features and graph structure.

Data Preparation:

- Sampling positive edges (existing edges in the graph).
- Generating negative edges (nonexistent edges) by sampling random node pairs that are not connected.
- Splitting the edges into training, validation, and test sets.

Model Implementation: Update the GNN model to output edge-level predictions. Use methods such as concatenating node embeddings followed by a feedforward layer or using a dot product of node embeddings to predict edge existence.

Training and Evaluation: Train the model using the training set and tune hyperparameters on the validation set. Evaluate the model on the test set, reporting metrics such as accuracy, precision, recall, F1-score, and ROC-AUC.

Analysis: Discuss the model's ability to predict links and how it might be used in practical applications, such as recommender systems, knowledge graph completion, or social network analysis.

Graph Embeddings. Explore graph embeddings generated by the trained GNN. Extract node embeddings from one of the intermediate layers of the GNN, and use dimensionality reduction techniques like t-SNE or PCA to project embeddings into 2D space. Visualize the embeddings to observe how nodes from different classes are clustered.

Node2Vec. Use node2vec to embed nodes on a Euclidean space, and use them for link prediction. Compare the quality of these embeddings against those generated by the GNN. Use the embeddings for node classification task and flink prediction and analyze the results. Plot the embeddings and compare them with the ones obtanined using GNNs.

Explainability. Apply methods like GNNExplainer to identify important subgraphs or neighbor nodes that contribute to a node's classification. Provide interpretations of the model's decisions based on the explainability analyses.

What to deliver

Deliver a PDF report documenting your entire work, including your methodology, results, and discussions, as well as a well-documented script or notebook with modularized and clear code.