**Homework-2-Report**

**Q1: Graph Embedding**

**Q1.1 Node Classification Task**

1. **Introduction**

Accurately addressing the node classification problem requires a thorough understanding of the dataset. The Node2Vec Wikipedia dataset captures the network of links between Wikipedia pages, where each node represents a page, and the edges denote hyperlinks between them. The labels assigned to nodes typically correspond to categories such as academic disciplines or subject areas. By analyzing the structural properties of the network, we can classify unlabeled nodes or predict the likelihood of new links forming between pages.

In this report, I utilize the **Node2Vec** algorithm to generate node embeddings from the Wikipedia dataset. A **logistic regression classifier** is then trained on these embeddings to predict the categories of the nodes. Finally, the performance of the classifier is evaluated using metrics such as accuracy, precision, recall, and F1-score.

2. **Results**

The table below summarizes the model's performance under different Node2Vec hyperparameter settings, when walk\_length=20, num\_walks=100, and dimensions=64:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Group** | **Settings** | **Accuracy** | **Precision** | **Recall** | **F1-Score** |
| S-1 | , | 0.5321 | 0.4564 | 0.5321 | 0.4609 |
| S-2 | , | 0.5286 | 0.4352 | 0.5286 | 0.4590 |
| S-3 | , | 0.5328 | 0.4491 | 0.5328 | 0.4613 |

3. **Analysis and Discussion**

The model's performance exhibited only minor variations across the different hyperparameter settings. Among the three configurations, S-3 (, ) achieved slightly higher accuracy than S-2 (, ). This outcome could be attributed to the fact that when , the Node2Vec algorithm tends to favor depth-first search (DFS), and when , it favors breadth-first search (BFS). The higher performance in S-3 suggests that this specific exploration strategy might better suit the Wikipedia graph’s structure, where BFS may capture more relevant node relationships.

However, it is important to note that the accuracy, across all settings, remained relatively low, hovering around 0.5. This suggests that while Node2Vec modifies the random walk strategy compared to DeepWalk, some information loss is inevitable, especially with limited hyperparameters (such as short walk lengths, fewer walks, and lower dimensions), constrained by computational resources. The reduced walk lengths and embedding dimensions likely restricted the model's ability to fully capture both local and global structural features.

Another potential reason for the model's underperformance is the inherent limitations of the logistic regression classifier, which is better suited to problems with linear decision boundaries. Graph data is often complex and nonlinear, so logistic regression may not fully capture the nuances of the node embeddings.

**Suggestions for Improvement**

1. **Adjusting Hyperparameters**: Increasing walk\_length, num\_walks, and dimensions in the Node2Vec algorithm could help extract richer features, leading to improved classification performance.
2. **Using Alternative Classifiers**: Employing more sophisticated classifiers, such as Support Vector Machines (SVM) or neural networks, which can handle nonlinear boundaries, could also enhance results.

4. **Conclusion**

In conclusion, node embedding serves as a critical step in enabling graph data to be processed by traditional machine learning models, which cannot directly handle graph structures. The Node2Vec algorithm extracts local and global graph features through random walks, converting nodes into vector representations. These vectors can then be used with general learning algorithms to perform tasks such as node classification. Despite the limitations observed in this experiment, the results provide a foundation for further refinement through enhanced hyperparameters and more advanced classifiers.

**Q1.2 Link Prediction Task**

1. **Introduction**

Facebook Social Network dataset mainly records the relationship between users of Facebook app. The node represents ONE user, and the edge connecting two nodes indicates that the two users are friends. Our task is to predict latent friendship in a uncomplete gragh with removing 50% edges (following the experimental setting from the original Node2Vec paper).

The combination of dataset (train and test) are as follows:

* **Training Positive Samples**: These are the edges that remain after the removal (the original edges minus the test edges).
* **Training Negative Samples**: Randomly sample an equal number of negative examples (non-edges) from the original graph to balance the dataset. These are pairs of nodes that do not have a link between them in the original graph.
* **Test Positive Samples**: The removed edges (50% of original edges) will be the positive samples for the test dataset because they are the links that actually exist in the original graph but are now "missing" in the training data.
* **Test Negative Samples**: The pairs of nodes that do not have an edge between them in the original graph. This ensures that the model learns to distinguish between existing and non-existing links.

2. **Results**

The table below summarizes the model's performance under different Node2Vec hyperparameter settings, when walk\_length=20, num\_walks=100, and dimensions=64:

|  |  |  |
| --- | --- | --- |
| Group | Hyperparameters settings | AUC Score |
| S-1 |  | 0.9822 |
| S-2 |  | 0.9803 |
| S-3 |  | 0.9828 |

3. **Analysis and Discussion**

Since a higher AUC value indicates a better-performing model, our trained predictors perform relatively well in different Hyperparameters settings with minor differences. Noted that when , the model performs better in other two conditions. Since the Node2Vec algorithm tends to favor depth-first search (DFS), when and when , it favors breadth-first search (BFS), the higher performance in S-3 suggests that this specific exploration strategy might better suit the Facebook graph’s structure, where BFS may capture more relevant node relationships.

4. **Conclusion**

The primary distinction between Q1.1 and Q1.2 lies in the types of tasks being addressed: Q1.1 focuses on node classification, while Q1.2 centers on edge prediction. However, upon closer examination, the two implementations exhibit similarities. In Q1.2, the prediction task is reformulated as an implicit classification problem. If an edge exists between two nodes, it is assigned a label of '1'; if not, it is assigned a label of '0.' This allows us to apply the same logistic regression model used in Q1.1 for prediction. A classifier output of '1' indicates a predicted edge, signifying a friendship between the two nodes in the Facebook graph, while an output of '0' suggests that the nodes are not connected.

**Q2: Knowledge Graph Completion with KGE**

1. **Introduction**

Our goal is to train TransE, TransR, and DistMult models on the WN18RR dataset and compare their performance with matrics Mean Rank, Hits@k and Mean Reciprocal Rank (MRR).

2. **Results**

The table below summarizes the model's performance under different models (TransE, TransR and DistMult):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Model | MR | MRR | Hits@1 | Hits@3 | Hits@10 |
| 0 | TransE | 14281.856190 | 0.000070 | 0.002394 | 0.005472 | 0.015048 |
| 1 | TransR | 14031.178010 | 0.000071 | 0.005814 | 0.013338 | 0.025650 |
| 2 | DistMult | 12233.733413 | 0.000082 | 0.169289 | 0.202291 | 0.239740 |

3. **Analysis and Discussion**

In conclusion, **DistMult** consistently outperforms both **TransE** and **TransR** across all metrics, indicating it is the most effective model for this particular task. And **TransR** is better than **TransE**, though both are significantly behind DistMult, especially in the top-ranked hits.

The performance differences highlight the potential of DistMult for applications requiring accurate and reliable entity retrieval, while TransE may require further tuning or a different approach to improve its results.

4. **Conclusion**

Compared to DistMult, TransE and TransR share the similar score function, while TransE projects entity and relationship into ONE space, TransR projects into DIFFERENT space and connects entity and relationship via tranform matrix. **DistMult** improves the scoring function by utilizing a **bilinear scoring** approach. Instead of simple vector addition, DistMult uses the outer product of the entity vectors to capture interactions between the head entity, tail entity, and relationship, leading to a more expressive model.