NODE CLASSIFICATION USING GRAPH EMBEDDING ALGORITHMS DeepWalk and Node2Vec

Vishakha Gautam
110011411
MSc, Computer Science, University of Windsor

Abstract

With an increase in social media data, recommender systems and online data; graph data has also started increasing with a fast pace. This graph data should be used to get valuable insights. Converting graph data to latent representations can help in achieving classification and prediction tasks. In this paper, the author has compared two graph embedding algorithms – DeepWalk and Node2Vec for performing Node Classification on CORA dataset. From the results achieved, it can be stated that Node2Vec outperformed DeepWalk on Cora Dataset. It gave a classification accuracy of 76.26% with a Random Walk length of 100. Also, DeepWalk was computationally faster than Node2Vec while performing Walk on the CORA dataset.

Introduction

Converting complex objects like text, image and graph to embeddings has become a popular technique for classification and prediction tasks in Machine Learning. Embeddings help in converting the text, image, and graph datasets to vectors with lesser number of features in comparison to the dimensions of the original dataset. Word2Vec, GLoVe, SVD and co-occurrence matrix are various techniques to convert text data to embeddings. Conversion of data to embeddings or latent representation has helped researchers achieve great enhancements in the field of Natural Language Processing, Speech Recognition and many more. Recently, graph data has become the topic of research. Many algorithms are being formulated to tackle graph data and retrieve features from it. Graph Embedding Algorithms like DeepWalk, Node2Vec, Cluster-GCN, LINE, PTE have gained extreme popularity with an increase in graph data. Embeddings can be used to classify nodes or graph. The two-broad classification of embedding algorithms for graphs are:

- Classifying the complete graph is known as '**Whole Graph Embedding'**
  Examples – Graph2Vec, sub2vec, WL Kernels
- Classification of nodes in the graph is known as '**Node Embedding**' or '**Vertex Embedding**'
  Examples – DeepWalk, Node2Vec, GCN, LINE, PTE

In this paper, the author has tried to compare two Node Embedding algorithms namely – DeepWalk and Node2Vec. The embeddings retrieved using the algorithms are then used to perform node classification on a graph dataset.

DeepWalk – DeepWalk is a very popular graph embedding algorithm given by [1]. DeepWalk uses Random Walks to traverse the nodes of the graph and samples them to build a corpus. The corpus is then given as input to the Word2Vec Skipgram model which helps in predicting the nodes before and after a particular node. The Skipgram model converts the graph nodes to embeddings. These embeddings are then used for downstream tasks such as Node Classification, Link Prediction, Graph Classification and many more.

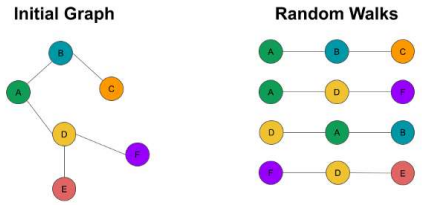

Fig 1 – Random Walks in DeepWalk

Drawback of DeepWalk – DeepWalk performs random walks randomly. Thus, DeepWalk embeddings do not preserve the local neighbourhood of the nodes. This shortcoming of DeepWalk was overcome in Node2Vec method proposed by [2].

Nod2Vec – Node2Vec approach is very similar to DeepWalk. The only difference between the two is that Node2Vec introduces a walk bias variable which is parameterized by 'p' and 'q'. Parameter 'q' prioritizes depth first search (DFS) and parameter 'p' prioritizes breadth first search (BFS). The probabilities 1/p and 1/q determine where to walk next.

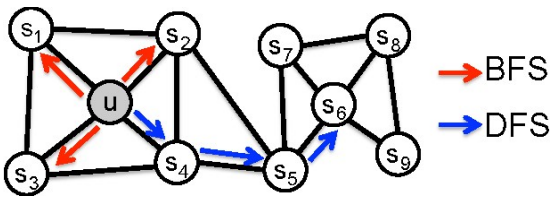

Fig 2 – Node2Vec – BFS and DFS approach

Parameter 'q' is used to define the probability of the random walk to reach the undiscovered parts of the graph, i.e., DFS is used for In Node2Vec, p is **return parameter**, which should have high value to ensure it does not get stuck in the local neighbourhood whereas q is **inOut parameter,** which should be less than 1. Using q, the walk is more inclined to visit nodes which are further away from the last visited node. Thus, encouraging outward exploration.

Fig 3 – Random Walk using DeepWalk

learning global variables. On the other hand, Parameter 'p' is used to define the probability of the random walk to return to its previous node, i.e., BFS is used for learning local neighbours.

Node2Vec uses the corpus generated by the biased random walk and inputs it to a neural network to retrieve embeddings. These latent representations are then used for Classification tasks.

Dataset used

CORA dataset was used to achieve the results in this project. The Cora dataset consists of Machine Learning papers. These papers are classified into one of the following seven classes:

1) Case_Based
2) Genetic_Algorithms
3) Neural_Networks
4) Probabilistic_Method
5) Reinforcement_Learnig
6) Rule_Learning
7) Theory

The papers were selected in a way such that in the final corpus every paper cites or is cited by at least one other paper. There are 2708 papers in the whole corpus.

Experimentation

In this paper, the author has tried to compare Graph Vertex Embedding Algorithms namely – 'DeepWalk' and 'Node2Vec'.

DeepWalk uses Uniform Random Walk whereas Node2Vec uses Biased Random Walk with a search bias parameter – p and q.

```
random_walk =
UniformRandomWalk(g)
walks = random_walk.run(
nodes=list(g.nodes()),
length=100,
n=10)
```

```
random_walk = BiasedRandomWalk(g)
walks = random_walk.run(
nodes=list(g.nodes()),
   length=100
   n=10,
   p=0.5,
   q=2.0)
```

Fig 4 – Biased Random Walk using Node2Vec

The total number of Random Walks performed by both the algorithms was – **24850** when length was 100 and 50 and n was 10. The number of Random Walks can vary depending upon the change in parameters. If n is decreased, number of random walks is reduced too.

The corpus retrieved after performing walks on the dataset was passed as input tot Word2Vec Skipgram model given by [3]. Word2Vec Skipgram model helps in representing the data as embeddings. These embeddings are vector representations of data points. The T-SNE representations of graph embeddings retrieved for DeepWalk and Node2Vec are as follows –

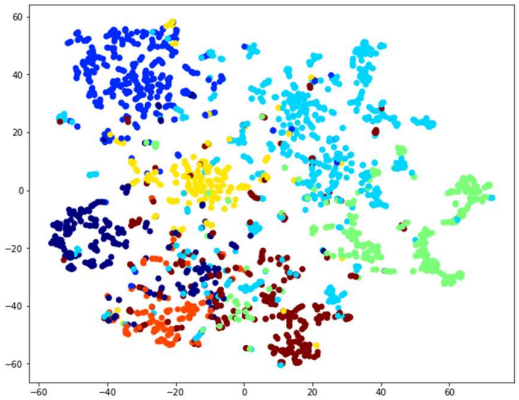

Fig 5 – T-SNE representation (DeepWalk)

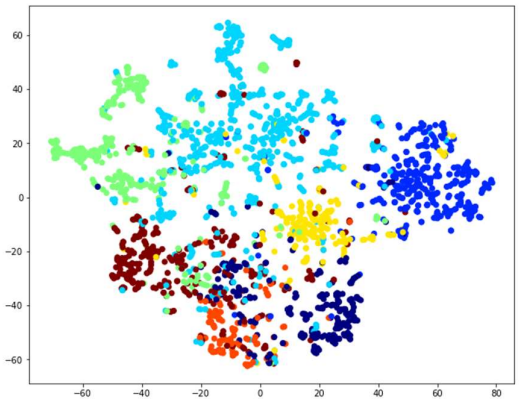

Fig 6 – T-SNE representation (Node2Vec)

We can observe the 7 different colours used to represent the 7 different classes in the dataset.

These embeddings are then used to perform Node Classification Task.

Logistic Regression is used for the Classification Task. Cross – validation with 10 folds is used. We use 75% of the data for training and the remaining 25% for testing as a hold out test set.

Results

Node2Vec outperformed DeepWalk on the CORA dataset in terms of Accuracy and Computational Time. Different parameters and lengths of Random Walks were tested, and the best results obtained are displayed below –

Algorithm	**Length of Walk**	**Classification Accuracy Achieved (in %)**
DeepWalk	20 (n=50)	78.54
	100 (n=10)	74.60
	50 (n=10)	75.62
Node2Vec	100 (n=10)	76.26

	20 (n=50)	74.52
	50 (n=10)	75.68

Table 1 – Results

The confusion matrices retrieved as output to the classification report are as follows –

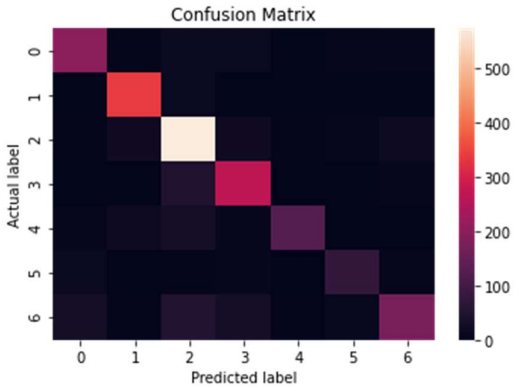

Fig 7 – Confusion Matrix for Classification using DeepWalk

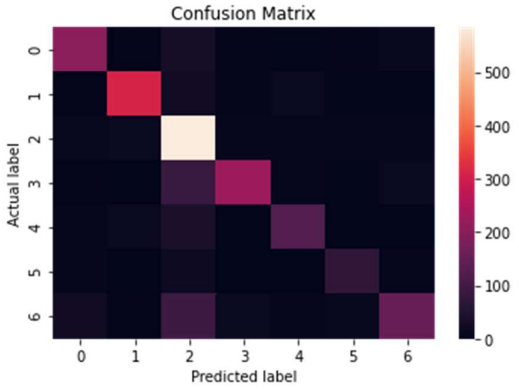

Fig 8 - Confusion Matrix for Classification using Node2Vec

Conclusion

Graph Embedding algorithms are playing a major role in performing downstream tasks such as Node Classification, Graph Classification, Link Prediction, Anomaly Detection [1] and many more. In the experiments shown above –

1. Node2Vec gave better classification accuracy in all the cases.
2. DeepWalk was computationally faster than Node2Vec (see code).
3. Node2Vec outperformed DeepWalk while performing on the CORA dataset.
4. Decreasing the length of random walk and number of random walk per root node results in higher number of random walks but lesser accuracy for Node2Vec but higher accuracy for DeepWalk.

Thus, it can be concluded that for this project, Node2Vec gave better classification accuracy and F1 score than DeepWalk.

References

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[2] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.

[3] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, 2013, pp. 3111–3119.