Node Classification using Graph Embedding Algorithms

DeepWalk and Node2Vec

Vishakha Gautam

MSc Computer Science, University of Windsor

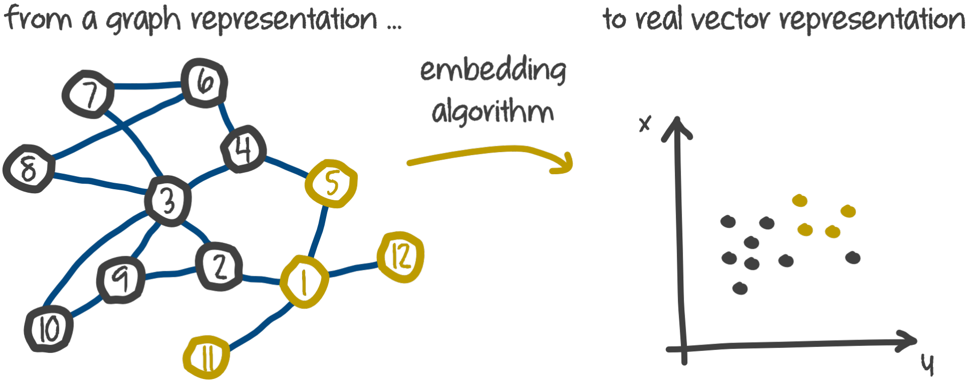
April 7, 2021

Overview

1. Introduction to Graph Embedding Techniques
2. Experiments with DeepWalk and Node2Vec

Graph Embedding Algorithms

- Embeddings help in converting graph datasets to vectors with lesser number of features in comparison to the dimensions of the original dataset.

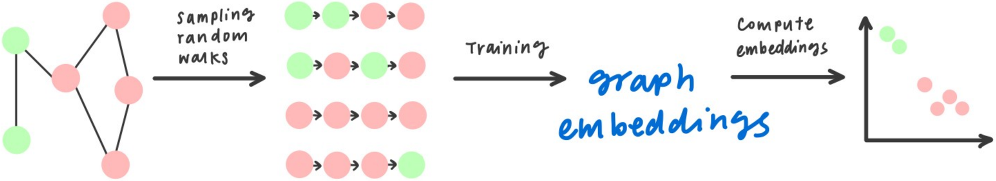

from a graph representation ...
to real vector representation
embedding
algorithm
7
6
8
4
5
3
9
2
1
12
10
11
x
y

Broad Classification of Graph Embeddings in 2 groups

- **Whole Graph Embedding** - Representing the complete graph with one vector
  Examples - Graph2Vec, sub2vec

- **Vertex (Node) Embedding** - Vector Representation of each node in the graph
  Examples - DeepWalk, Node2Vec, GCN, LINE, PTE

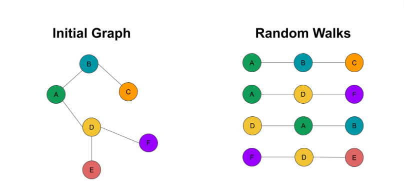
Vertex Embedding Algorithm



Dataset used in the project

- **CORA dataset** - The Cora dataset consists of 2708 Machine Learning publications.
- These papers are classified into one of the following seven classes:
  1. Case Based
  2. Genetic Algorithms
  3. Neural Networks
  4. Probabilistic Methods
  5. Reinforcement Learning
  6. Rule Learning
  7. Theory

DeepWalk

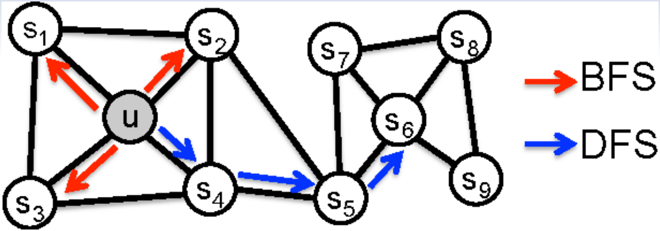


Initial Graph

Random Walks

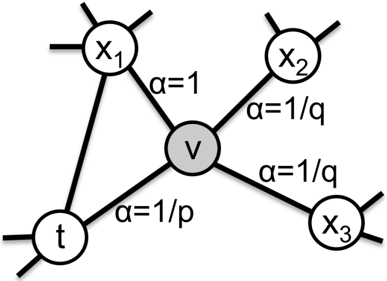
```
randomWalk = UniformRandomWalk(g)
walks = randomWalk.run(
nodes=list(g.nodes()),
length=100,
n=10)
print(”Number of random walks: ”.format(len(walks)))
```

Node2Vec



```
randomWalk = BiasedRandomWalk(g)
walks = randomWalk.run(
nodes=list(g.nodes()),
length=100,
n=10,
p=0.5,
q=2.0)
print(”Number of random walks: ”.format(len(walks)))
```

Node2Vec - Return and In-out parameters



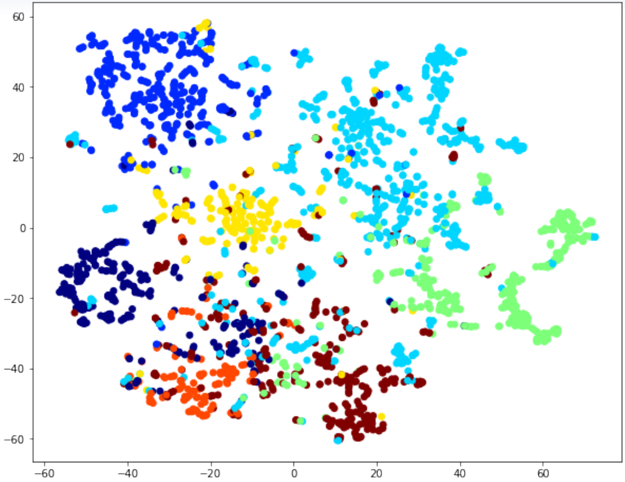
p - Return parameter; should have high value to ensure it does not get stuck in it's local neighbourhood

q - InOut parameter; q less than 1; the walk is more inclined to visit nodes which are further away from the node t.Thus, encouraging outward exploration

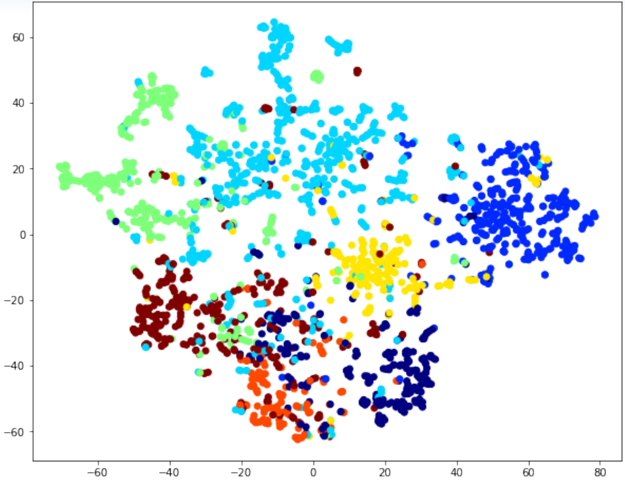
Applying Word2Vec to the corpus

```
from gensim.models import Word2Vec
strwalks = [[str(n) for n in walk] for walk in walks]
model = Word2Vec(strwalks, size=128, window=5, mincount=0, sg=1, workers=4)
```

T-SNE Representation of Embeddings using DeepWalk



T-SNE Representation of Embeddings using Node2Vec



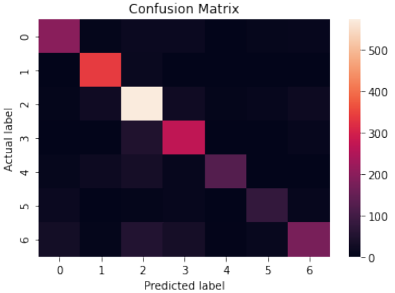
Applying Logistic Regression for Node Classification

```
clf = LogisticRegressionCV(Cs=10, cv=10, scoring=”accuracy”, verbose=False,
multiclass=”ovr”, maxiter=300)
clf.fit(Xtrain, ytrain)
```

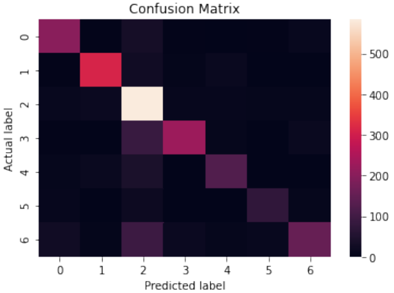
Results

Algorithm	Length of Walk	Classification Accuracy Achieved
DeepWalk	100	74.60
	50	75.62
Node2Vec	100	76.26
	50	75.68

Confusion Matrix for Classification of Research Papers



DeepWalk



Node2Vec

Conclusion

- Node2Vec outperforms DeepWalk when used on Cora Dataset
- Node2Vec gives higher accuracy when length of Random Walks is 100 and even when it was lowered to 50
- DeepWalk is computationally faster than Node2Vec

THANK YOU