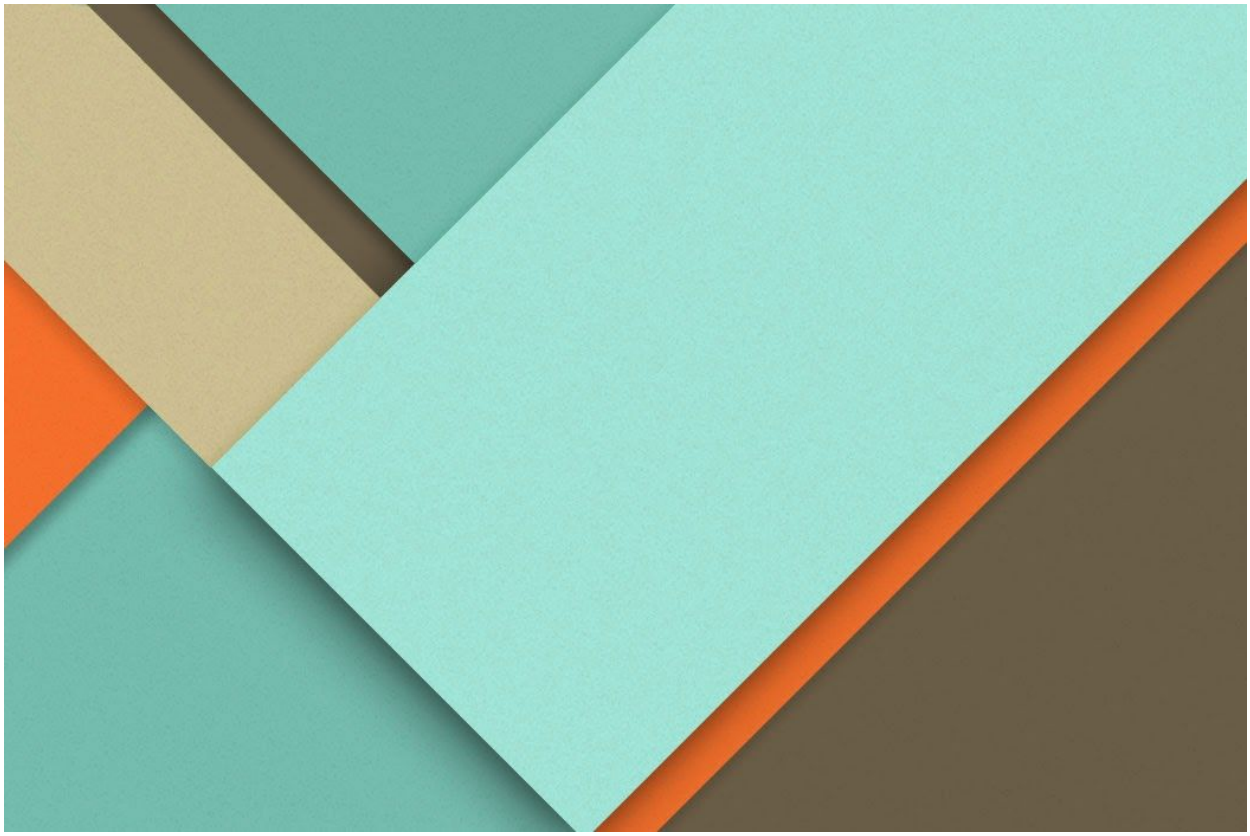

# COSI126a_HW4

10.07.2020

**Menghong Han, MSBA**

International Business School

Brandeis University

## Write a program to solve the node classification problem by Node2vec, and test it on Cora. Report your parameter settings of Node2vec, and the accuracy of your prediction. You can use the scikit-learn package for classification.

Node2vec is based on DeepWalk, it adapts a biased random walk instead of the random walk in DeepWalk. Given the current node, the probability of accessing the next node is first calculated based on the weights of all edges, and then two parameters P and Q are added to control the strategy of wandering.

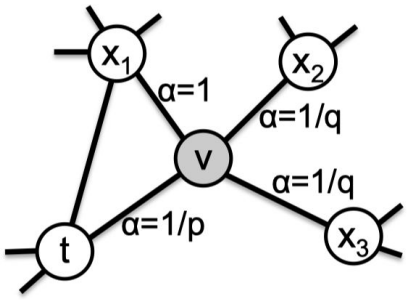


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from $t$ to $v$ and is now evaluating its next step out of node $v$. Edge labels indicate search biases $\alpha$.

Except for the biased random walk, node2vec used alias sampling which has the O(1), greatly increasing the efficiency.

For the parameter settings, I tried multiple combinations, num_walk=10, walk_length = 40 as settled in the original article and then tried

(p = 1, q = 1),

(p = 0.5, q = 2),

(p = 2, q = 0.5),

(p = 0.5, q = 1),

(p = 1, q = 0.5).

I used logistic regression in the scikit-learn package to evaluate the result.

The final results:

| p | q | num_walk | walk_length | accuracy | time |
|---|---|---|---|---|---|
| 1 | 1 | 10 | 40 | 0.747 | 10.76s |
| 1 | 0.5 | 10 | 40 | 0.77 | 9.98s |
| 2 | 0.5 | 10 | 40 | 0.749 | 10.2s |
| 0.5 | 1 | 10 | 40 | 0.747 | 9.89s |
| 0.5 | 2 | 10 | 40 | 0.743 | 9.81 |

From the above result, we can see DFS perform better than BFS while in a slower speed with Cora data, and the smaller the difference between p and q, the higher accuracy becomes.

**Run the code:**

python node2vec/main.py --input cora/network.npz --output cora/cora.node2vec.embeddings --num-walks 10 --walk-length 40

python evaluate_cora.py --emb cora/cora.node2vec.embeddings --net cora/network.npz --labels cora/labels.npz