Prof. Dr. M. Grohe J. Tönshoff

Exercise Sheet 4

Due date: Tuesday, June 14th, 23:59

- The content of your master branch at the time of the deadline is your submission.
- You should use the following Python packages in this exercise:
  - PyTorch
  - NetworkX
  - Numpy
  - Scikit-Learn
  - argparse

Exercise 1 (Random Walks) 9 points

Implement a dataset that samples random pq-walks from a given graph G. The dataset should inherit from the `torch.utils.data.IterableDataset` class. The constructor should take G and the walk parameters $q, p, \ell, \ell_{\text{ns}}$ as input. You must implement the `__iter__()` method. This method must return a python iterator that iteratively returns random walks w in G (and the corresponding negative samples $\bar{w}$). For training, you can pass this dataset to the `DataLoader`. You can set the `num_workers` argument of the loader fur mulit-processing.

Reminder Random Walks:

- Input: Graph $G = (V, E)$, parameters $p, q > 0$, $\ell, \ell_{\text{ns}} \in \mathbb{N}$
- Starting from a random node $s = v_0$, jump along the edges of G for ℓ steps. If v_i is the current node and v_{i-1} is the previous node, then the probability of jumping to $x \in \mathcal{N}(v_i)$ in the next step is given by:

$$\mathbf{P}(x|v_i, v_{i-1}) = \frac{\alpha_{p,q}(x; v_i, v_{i-1})}{\sum\limits_{y \in \mathcal{N}(v_i)} \alpha_{p,q}(y; v_i, v_{i-1})}, \quad \alpha_{p,q}(x; v_i, v_{i-1}) = \begin{cases} \frac{1}{p} & \text{if } d(v_{i-1}, x) = 0 \\ 1 & \text{if } d(v_{i-1}, x) = 1 \\ \frac{1}{q} & \text{if } d(v_{i-1}, x) = 2 \end{cases}$$

  where $d(v, u)$ is the shortest distance between v and u in G.
- The initial node $s = v_0 \sim V$ is sampled uniformly at random.
- The first neighbor $v_1 \sim \mathcal{N}(v_0)$ is also sampled uniformly at random.
- The output is the walk $w = (s, v_1, \ldots, v_\ell)$ and a set of negative samples $\bar{w} \subset V$ with $|\bar{w}| = \ell_{\text{ns}}$ and $w \cap \bar{w} = \emptyset$.

Exercise 2 (Node2Vec) 9 points

Implement a method that computes a Node2Vec embedding for a given graph G using PyTorch. Use the dataset from Task 1 to generate the random walks. Throughout all tasks you may use a walk length of $\ell = 5$, a negative sample size of $\ell_{\text{ns}} = 5$ and an embedding dimension of $d = 128$.

Reminder Node2Vec

- Probability of observing a node $v \in V$ and walk $w \subset V$ when starting at $s \in V$:

$$\mathbf{P}(v|s) = \frac{\exp(\langle x_v, x_s \rangle)}{\sum_{u \in w \cup \bar{w}} \exp(\langle x_u, x_s \rangle)} \quad , \quad \mathbf{P}(w|s) = \prod_{i=1}^{\ell} \mathbf{P}(v_i|s)$$

- Loss for a walk $w = (s, v_1, \ldots, v_\ell)$:

$$\mathcal{L}(X; w) = -\sum_{i=1}^{\ell} \log \mathbf{P}(v_i|s)$$

Hints

- You can represent Node2Vec as PyTorch Module that holds the embedding matrix $X \in \mathbb{R}^{|V| \times d}$ as a parameter. These are initialized randomly.
- The `forward` method of the model should take 3 inputs: A starting node s, a sampled walk w starting at s and its negative samples $\bar{w}$ (stacked along dimension 0 for all walks of the batch W).
- The output is the loss over the batch $\mathcal{L}(X, W)$.
- For training we recommend using the Adam Optimizer. Perform SGD on X until $\mathcal{L}(X; W)$ converges.

Exercise 3 (Node Classification) 7 points

Use your Node2Vec implementation to perform node classification on the *Citeseer* and *Cora* networks. You should use logistic regression as the downstream classifier. Use the embedding vectors as node features to train and evaluate the classifier.

Compute embeddings with parameters $(p = 1, q = 1)$, $(p = 0.1, q = 1)$ and $(p = 1, q = 0.1)$ for each dataset. For each of these node embeddings, perform 10-fold cross validation on the node classification task. Compute the mean accuracy and standard deviation achieved with each embedding. Report the results in your readme. Your mean accuracy on the evaluation data should be above the following thresholds for at least one choice of p and q (-5 Points if not):

- Cora: 75%
- Citeseer: 60%

Hints

- The embedding must be computed for the whole networks, which can be downloaded from moodle. Do not use the split graphs from sheet 2.
- The node attributes are not used in this exercise. The classification should be based entirely on the computed embedding vectors.
- You can use the logistic regression classifier from Scikit-Learn.

Exercise 4 (Link Prediction) 10 points

Use your Node2Vec implementation to perform link prediction on the *Facebook* and *PPI* networks. Use logistic regression as a classifier.

Experimental Procedure:

- Initially, choose 20% of the edges as evaluation data $E_{\text{eval}} \subset E$. These edges should be chosen randomly. Make sure that the original connected components are still connected with $E_{\text{train}} = E \setminus E_{\text{eval}}$.
- Sample two disjoint sets $N_{\text{train}}, N_{\text{eval}} \subset (V \times V) \setminus E$ as negative samples, such that $|E_{\text{eval}}| = |N_{\text{eval}}|$ and $|E_{\text{train}}| = |N_{\text{train}}|$.
- Compute a node2vec embedding for $G_{\text{train}} = (V, E_{\text{train}})$. As parameters you can use $p = q = 1$ in this exercise.
- Turn the node embedding into an edge embedding. For every edge (u, v) the embedding is defined as the element wise product (Hadamard Product) of the connected node embeddings: $x_{(u,v)} = x_u \odot x_v$.
- Use the embeddings of E_{train} and N_{train} to train a classifier for link prediction. Evaluate this classifier on E_{eval} und N_{eval}.

Repeat this experiment 5 times on both graphs and compute the mean accuracy and the mean ROC-AUC score (as well as the standard deviation of these metrics).

Your mean accuracy on the evaluation data should be above the following thresholds (-5 Points if not):

- PPI: 70%
- Facebook: 90%

Exercise 5 (Code Quality, Comments and Presentation) 15 points

Clean your code and add useful comments. Your repository must contain a `README.md` file which provides the following information:

- A brief description of the structure of your repository
- How to run every executable script. For each script, all command line options must be fully specified
- The mean accuracy and standard deviation achieved for every dataset for node classification and link prediction. For link prediction, the ROC-AUC score and its standard deviation should also be reported.

Prepare a short presentation (∼5 min) for your group meeting (held in the week after the submission deadline). It should briefly provide the following information:

- What you implemented and how the work was split
- The results you obtained
- A brief interpretation and discussion of the results