Exercise Sheet 1

Due date: Tuesday, April 26th, 23:59

- Please register in RWTH-Gitlab and upload your username to RWTHmoodle:

  `https://git.rwth-aachen.de`

- The content of your master branch at the time of the deadline is your submission.

- List of Python tutorials: `https://wiki.python.org/moin/BeginnersGuide/Programmers`
  Style Guide for readable Python code: `https://www.python.org/dev/peps/pep-0008/`
  For documentation within your code: `https://www.python.org/dev/peps/pep-0257/`.

- You should use the following Python packages in this exercise:
  - NetworkX
  - Numpy (Scipy might be useful too)
  - Scikit-Learn
  - argparse

Exercise 1 (Closed Walk Kernel) 5 points

Implement the Closed Walk Kernel. This kernel simply counts the closed walks of different length in a given graph.

The Closed Walk Kernel

- A walk allows for nodes to reappear during the walk, a closed walk is a walk where the first and last node coincide
- The feature vector is a histogram of closed walks of different length up to some maximum length ℓ. The i-th entry of the vector counts the closed walks of length i in the graph.
- Please describe your choice of the maximal length ℓ in the readme.
- You may want to consider a computation through eigenvalues for the computation with large ℓ.

Exercise 2 (Graphlet Kernel) 5 points

Implement the presented Graphlet Kernel for graphlets of size 5 and $m = 1000$ samples.

Notes on the Graphlet Kernel

- There are 34 distinct graphs with 5 nodes (graphlets).
- Randomly sample 5 (distinct) nodes from a given graph and check which graphlet the induced subgraph is isomorphic to.
- Repeat this 1000 times and generate a histogram that counts how often each graphlet occurred.
- This histogram is the feature vector of the graph.

Exercise 3 (WL-Kernel) 15 points

Implement the presented Weisfeiler-Leman-Kernel.

Reminder WL-Kernel:

- Compute $k = 4$ rounds of colour refinement.
- Two nodes are assigned different colors if they had different colors in the previous round or the multiset of colors in their neighborhoods differ.
- The feature vector of a graph is the histogram of colors. Each vector entry counts the occurrence of one a color in the graph. Make sure that each entry counts the same color for every graph.
- This kernel can incorporate node labels by using the labels as the initial node colors. Make use of this for graphs that have node labels.

Exercise 4 (Support Vector Machine) 10 points

You are provided with three graph classification datasets: DD, ENZYMES and NCI1. Use your kernels to train *Support Vector Machines (SVM)* on these datasets. Make use of 10-fold cross validation to measure the accuracy of each kernel on each dataset.

At least one of your kernels must achieve the following mean accuracies on the test data (-5 points if it does not):

- DD: 78%
- ENZYMES: 45%
- NCI1: 80%

In your README, provide a table with the mean accuracies on training and test data as well as the standard deviations. Compare your results with the results from the paper *Weisfeiler-Lehman Graph Kernels* (`http://people.mpi-inf.mpg.de/~mehlhorn/ftp/genWLpaper.pdf`).

Hints

- For some kernels the feature vectors become very large. In this case it is slow and inefficient to pass the vectors directly to the SVM. A key advantage of SVMs is that they only really need the Gram-Matrix (the scalar products of each pair of feature vectors) for training and testing. The size of the feature vectors is 'hidden' behind the scalar values in this matrix, which allows SVMs to efficiently process extremely large vectors. When using the SVM implementation of scikit-learn you may use the 'precomputed' option to pass Gram-Matrices instead of feature vectors.

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 accuracy and standard deviation achieved for every dataset with each kernel.

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 (Which kernel works best? Why?)