Technical University of Munich
Department of Informatics
Prof. Dr. Stephan Günnemann

Exercise to the lecture
***Mining Massive Datasets* in WS16/17**
Aleksandar Bojchevski (bojchevs@in.tum.de)
http://www.kdd.in.tum.de/mmds

Sheet No. 6

Exercise 1 - Projected Gradient Descent

Given the following set of constraints that define the (convex) domain $X \subseteq R^2$: $x_1 + x_2 \leq 4 \wedge 0 \leq x_1 \leq 3 \wedge 0 \leq x_2 \leq 2.5$

a) Visualize the set X

b) Derive a closed form for the projection $\pi_X(p) = argmin_{x \in X} ||x - p||_2^2$. That is, given an arbitrary point p, what is it projection on X?

c) Given the following constrained optimization problem:
$\min_x[(x_1 - 2)^2 + (2 \cdot x_2 - 7)^2]$ subject to $x \in X$.
Perform two steps of projected gradient descent starting at position $(2.5, 1)$. Use a constant learning rate/step size of $\tau = 0.05$.

Exercise 2 - Map-Reduce

a) Write a MapReduce program (in pseudo-code describing the map-phases and reduce-phases) that implements a simple "People You Might Know" social network friendship recommendation algorithm. The key idea is that if two people have a lot of mutual friends, then the system should recommend that they connect with each other.

Input: Assume the the input file contains the adjacency list and has multiple lines in the following format:
<User><TAB><Friends>
Here, <User> is a unique integer ID corresponding to a unique user and <Friends> is a comma separated list of unique IDs corresponding to the friends of the user with the unique ID <User>. Note that the friendships are mutual (i.e., edges are undirected): if A is friend with B then B is also friend with A.

Algorithm: Let us use a simple algorithm such that, for each user U, the algorithm recommends N = 10 users who are not already friends with U, but have the most number of mutual friends in common with U.

Output: The output should contain one line per user in the following format:
<User><TAB><Recommendations>
where <User> is a unique ID corresponding to a user and <Recommendations> is a comma separated list of unique IDs corresponding to the algorithm's recommendation of people that <User> might know, ordered in decreasing number of mutual friends.

Remarks: It is possible to solve this question in various ways. Please also note that the keys and values used in Map-Reduce might be 'complex' structures (e.g. tuples). Also note that it is possible to have multiple map and reduce steps in one MR-job.

b) Discuss the advantage and potential disadvantages of your solution.

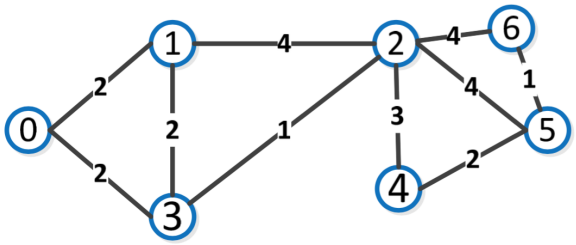
Exercise 3 - Efficient Triangle Counting Prove that the number of triangles in a graph is equal to $\frac{1}{6}trace(A^3) = \frac{1}{6}\sum_i \lambda_i^3$ where λ_i are the eigenvalues of the adjacency matrix A of the graph.

Exercise 4 - Ranking

Consider a directed graph $G = (V, E)$ with $V = \{1, 2, 3, 4, 5\}$, and $E = \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 4), (3, 5), (4, 5), (5, 4)\}$.

1. Set up the equations to compute pagerank (using random teleports) for G. Assume that the probability of teleport is 0.2.
2. Set up the equations for topic-specific pagerank for the same graph, with teleport set $\{1, 2\}$. Solve the equations and compute the ranking vector.
3. Give examples of pairs (S, v), where $S \subseteq V$ and $v \in V$, such that the topic-specific pagerank of v for the teleport set S is equal to 0. Explain why these values are equal to 0.

Exercise 5 - Cuts and Spectral Clustering



a) Given the above graph, find the following partitionings of the graph for k=2:
   - The partitioning minimizing the global Minimum Cut
   - The partitioning minimizing the Ratio Cut
   - The partitioning minimizing the Normalized Cut

b) Prove that the three properties mentioned on slide 79 (Chapter 6: Graphs/Networks) are correct.

Project task 3 - Song Ranking

For this task we are going to use the **Last.fm** dataset, the official song tag and song similarity dataset of the Million Song Dataset. The dataset consists of two kinds of data for each song: **tags** and **similar songs**.

The general idea behind this project task is to create and analyze a song **similarity network**. If we consider the tags of each songs as topics, we can rank popular songs in the network given some specific tags (topics) using **Topic-Specific PageRank**.

The data is available as a set of json-encoded text files where the keys are: track_id, artist, title, timestamp, similars and tags. To get acquainted with the data you can start with the provided subset or look at the smaller test set. However, your implementation should be able to handle the entire train set.

Your tasks are as follows:

a) Parse the json files and create the song similarity network. Each song is a node in the network and there is an edge between two songs if they are similar.

   Note that the resulting graph is directed due to last.fm's methodology in calculating the similarity scores. For example, users who listen to song A often listen to song B, but users who listen to song B never listens to song A. More specifically, in each json file representing a song you are given the **outgoing** weighted edges from that song to other similar songs, where the weight represents the similarity score between 0 and 1.

   In your implementation introduce a threshold parameter t controlling whether you form and edge between two songs or not. That is, only connect two songs with an edge if the similarity score is strictly greater the specified threshold t. By default set $t = 0$, meaning you form all edges between similar songs.

   The result should be a non-symmetric binary adjacency matrix A representing the song similarity network. To be able to handle large networks be careful to use sparse matrix format.

b) Parse the json files and extract the set of tags for each song. Each tag of each song is associated with a count (e.g. ["Hip-Hop", "50"]). Introduce a threshold parameter g (default $g = 50$) and only keep the tags with counts greater or equal to g.

c) Implement Topic-Specific PageRank. Use a default teleport probability $\beta = 0.2$ and expose β as a parameter. Remember, when a random walker teleports we pick a song from a set of relevant songs S. Choose S to be equal to all songs associated with some user specified set of tags. The user should should be able to specify one or more tags of interest, e.g. songs that have the tag "Hip-Hop", or songs that have both "Rock" and "Pop" tags.

c) Report the top n (default $n = 5$) most popular songs according to the Topic-Specific PageRank for a given set of tags.

The deadline for this project task is **25.01.2017 23:55**.