AIM-3 Scalable Data Science (WiSe 2017-2018)

Homework Assignment 3

Due on 02.02.2018 at 14:15

Instructions. For all exercises, be sure to show all work to receive full credit, where appropriate follow the additional instructions specific to the individual exercise. Be aware that you may be asked to meet with an instructor to run your codes later. Upload your solutions as a **single archive** that includes a pdf-file of your written solutions, source-files with your implementations and all other relevant parts of your submission to ISIS (HW3_lastname.zip) by <u>no later than February 2, 2018 at 14:15</u>. Also, you will need to drop off a stapled, printed copy of your homework assignment at the start of the lecture held on February 2. Lastly, be certain to work individually, you are free to drop some hints. However, you must solve these problems on your own.

1. Dimensionality Reduction (Total: 5 points)

Dimensionality reduction methods plays a vital role in performing machine learning on high-dimensional data. In class, we learned about the singular value decomposition (SVD). In MMDS, Section 11.3, the authors introduce CUR Decomposition.

(a) What is CUR decomposition? Briefly describe how it works. When is it useful? (2 points)
(b) Solve Exercise 11.4.2 in the MMDS book. (3 points)

2. Item Based Collaborative Filtering (Total: 5 points)

You are given a dataset of ratings as depicted in the table below:

User	Movie A	Movie B	Movie C	Movie D
Ryan	1	5	3	5
Stavros			2	
Brahma		4	1	1
Brodie	4	3		2
Zosimus		5	1	

(a) Compute the item similarity matrix (denoted by R) using Pearson Correlation as a measure.
(b) Compute the item similarity matrix (denoted by S) using Jaccard Coefficient as a measure.
(c) Use matrices R and S to compute the corresponding ratings that Zosimus would give to Movies A & D.
(d) Use matrices R and S to compute the corresponding ratings that Stavros would give to Movies A & B.

For each sub-problem (i) through (iv), show how you arrived at your solution.

3. Network Analysis (Total: 10 points)

(a) Solve exercise 5.1.1 in the MMDS book, on page 175. Compute the PageRank of each page in Fig. 5.7, assuming no taxation. Show your work including the adjacency matrix, the initial PageRank vector, and the values of the PageRank vector for the first three iterations, as well as an approximation of the converged result rounded to two significant figures. (5 points)

(b) Solve exercise 5.1.2 in the MMDS book, on page 176. Compute the PageRank of each page in Fig. 5.7, assuming $\beta = 0.8$. Show your work including the adjacency matrix, the initial PageRank vector, and the values of the PageRank vector for the first three iterations, as well as an approximation of the converged result rounded to two significant figures. (5 points)

4. Support Vector Machines (Total: 10 points)

(a) What are support vector machines (SVMs)? When are they typically used? (2 points)

(b) Can SVMs be employed for multiclass classification? Discuss the difference between the *one-against-rest* (a.k.a. *one-versus-all*) approach and the *one-against-one* (a.k.a. *one-versus-one*) approach? (4 points)

(c) Suppose that a classification training algorithm requires $O(n^r)$ time for training on a data set of size *n*. Here *r* is assumed to be larger than 1. Consider a data set *D* with exactly even distribution across *k* different classes. Compare the running time of the *one-against-rest* approach with the *one-against-one* approach. (4 points)

5. Classification: Building a Spam Filter with Apache Spark (Total: 15 points)

In this exercise, you will use one of Apache Spark's ML libraries to build a model that classifies text-messages as either *spam* or *no spam*. For that purpose, we have provided a dataset of labeled text messages from the UCI Machine Learning Repository[1] that was randomly split into a training- and test-dataset.

Your pipeline should include feasible processing steps that you have learned in the lecture such as, but not limited to:

- feature representation (bag-of-words or similar),
- possibly an appropriate feature weighting (TF-IDF or similar),
- a classification method (e.g., SVM, Naïve Bayes) to be applied to the training dataset,
- an evaluation of your classifier based on the performance on the provided test-dataset.

Currently, Apache Spark features two machine learning libraries that both provide the necessary methods. The older library 'spark.mllib' is RDD based[2][3], while the new library

[1] https://archive.ics.uci.edu/ml/datasets/sms+spam+collection
[2] https://spark.apache.org/docs/2.2.0/mllib-feature-extraction.html
[3] https://spark.apache.org/docs/2.2.0/ml-classification-regression.html

'spark.ml' is based on Spark's dataframes[4][5]. Officially, it is recommended by now to use the newer library[6], but for this exercise you can chose the one you feel more comfortable with (we used MLlib in the previous KMeans exercise).

Your submission should include:

- A single source file with your implementation ('spam.scala' or 'spam.java') (5 points)
- A (short) report that states the accuracy of your model and (more importantly) an explanation for why you have chosen a particular method for the single steps of your pipeline (10 points)

Note that since the Spark documentation that we have referenced in the footnotes provides explicit examples for the individual steps that are asked of you, your explanations are considered more relevant and thus give more points. The accuracy of your classifier will not influence your grade. For this exercise, you will find a repository at https://gitlab.tubit.tu-berlin.de/deuschle/Homework_3.

6. Network Statistics in Apache Flink (Total: 15 points)

In this exercise, you will conduct network analysis in Apache Flink on the *Slashdot-Zoo* dataset (a signed social network of users of the technology news site *Slashdot*), which is connected by directed *friend* and *foe* relations. The network contains 79,120 vertices connected by 515,397 edges. Every edge is marked with either +1 or -1 to denote a *friend* or *foe* relationship, correspondingly. **Tip**: See the README file.

The "friend" and "foe" labels are used on Slashdot to mark users, and influence the scores as seen by each user. For instance, if user A marks user B as a foe, the score of user B's posts will be decreased as shown to user A. If we consider the direction and add up the incoming and outgoing connections we will obtain two numbers for the degree of the node (i.e., in-degree and out-degree).

Use the *slashdot-zoo_stub* available at https://gitlab.tubit.tu-berlin.de/deuschle/Homework_3, as a starting point.

- Adjust the method **de.tuberlin.dima.aim3.assignment2.Cong#pathToSlashdotZoo**() to point to the location of the file *out.matrix* that you unpacked from the downloaded dataset.

- The first part of the assignment deals with the computation of simple network statistics, such as the distribution of the number of outgoing edges per vertex. The class **de.tuberlin.dima.aim3.assignment2.OutDegreeDistribution** describes a Flink program that computes this distribution. Note that the signs of the edges are ignored.

- Write a new Flink program **SignedOutDegreeDistribution** to compute two out-degree distributions, one that solely considers *friend* edges and a second one that solely considers *foe* edges, correspondingly.

- Write a new Flink program **AverageFriendFoeRatio** to compute the average ratio of friends and foes per vertex in the network. Note: Ignore vertices that only have friends or foes.

[4] https://spark.apache.org/docs/2.2.0/ml-features.html
[5] https://spark.apache.org/docs/1.6.1/ml-classification-regression.html
[6] https://spark.apache.org/docs/latest/ml-guide.html

In addition to your source codes, you should also complete and submit the following.

(a) Produce a CSV file that contains the overall ***in-degree*** distribution. Create the corresponding histogram with appropriate labels, accordingly.

(b) Produce a CSV file that contains the overall ***out-degree*** distribution. Create the corresponding histogram with appropriate labels, accordingly.

(c) Produce a CSV file that contains the *out-degree distribution for* ***friend*** *nodes*. Plot the probability distribution and interpret.

(d) Produce a CSV file that contains the *out-degree distribution for* ***foe*** *nodes*. Plot the probability distribution and interpret.

The following reference may be of use to you: http://mathinsight.org/degree_distribution. Lastly, you should ensure that your probability vectors sum to 1!