Heidelberg University
Winter semester 2017/2018

Group Parallel and Distributed Systems (PVS)
Artur Andrzejak, Diego Costa

Problem Set 5 for lecture Mining Massive Datasets

Due November 27, 2017, 11:59 pm

Exercise 1 (3 points)

Recall the following utility matrix from the last problem set. It contains the ratings of users A-C for the items a-h.

	a	b	c	d	e	f	g	h
A	4	5		5	1		3	2
B		3	4	3	1	2	1	
C	2		1	3		4	5	3

In this exercise, we cluster the items in this matrix. Perform the following steps (on paper, programming is not necessary - unless you prefer to do it).

(a) Cluster the eight items hierarchically into four clusters. The following method should be used to cluster. Replace all 3's, 4's, and 5's by 1 and replace 1's, 2's, and blanks by 0. Use the Jaccard distance to measure the distance between the resulting column vectors. For clusters of more than one element, take the distance between clusters to be the minimum distance between pairs of elements, one from each cluster (single linkage).

(b) Then, construct (from the original matrix) a new matrix whose rows correspond to users, and whose columns correspond to clusters. Compute each entry by averaging the non-blank entries for that user and all the items in the cluster. For instance, given the user A and a fictional cluster $C_1 = \{a, d, f\}$, the entry A at column C_1 should be 9/2.

(c) Compute the cosine distance between each pair of users, according to your matrix from part (b).

Exercise 2 (3 points)

Recall the award-winning Netflix recommendation system by the BellKor team (slide "BellKor Recommender System"). It takes into account three levels of effects: global effects (biases), regional effects (latent factors) and local effects (collaborative filtering). Imagine three systems G, R and L, that use only global, regional and local effects, respectively.

a) What would the lists of movies recommended by each system look like? Recall the long tail of items that an electronic system is able to offer. Does any of the systems G, R and L introduce users to more exotic items from the long tail?

b) Assume that a movie production team wants to artificially push the popularity of a specific movie X. It creates a lot of fake user accounts and gives highest ratings for this movie X. What system would be most robust to such an attempt? Which users' recommendation lists would be affected most?

c) Users whose opinions do not consistently agree or disagree with any group of people are called *gray sheep*. Users that have extremely specific and odd tastes are called *black sheep*. Can these types of users benefit from either of the three systems?

Exercise 3 (5 points)

Write a program in Python implementing Gradient Descent method (GD) for fitting a polynomial of degree d to n training samples (x, y). That is, the program should take n 2-dimensional points and find the polynomial of degree d that minimizes the Sum of Squared Errors (SSE) of that point set. The parameter d should be passed to the algorithm via command line options. Begin by formulating the corresponding minimization problem and calculating the gradient on paper. Then explain how to generalize this for a general input of n points and degree d of the polynomial.

Furthermore, test your implementation as follows. First, construct an arbitrary polynomial. Then, generate $n > 1000$ samples as slight perturbations of points on the graph of the polynomial. Run your program on this input und check, how good it is at recovering the original polynomial. Submit as a solution code of the SGD and code which tests your solution. Note: Instead of GD, you can implement the Stochastic Gradient Descent method (see slides of lecture 6 and the Wikipedia link[1]).

Exercise 4 (3 points)

The Movielens dataset captures movie ratings by user along with user and movie attributes. This exercise uses a subset of the Movielens dataset[2], which contains 100,000 ratings by 943 users on 1,682 items, with each having rated at least 20 movies. The ratings data (movielens.txt) is a tab-delimited text file with the following structure:`<user id>\t<item id>\t<rating>\t<timestamp>`. For this exercise, submit the source code as well as the below mentioned artifacts for a complete solution.

a) Load the data into Spark (RDD), converting each row into Ratings[3] data structure. Split your dataset through random sample, 50% into training and the remaining 50% into test data.

b) Use the Spark Mlib implementation of the Alternating Least Squares (ALS)[4]to train the ratings. Train your model using 10 latent factors and 5 iterations. Save the model to disk after training and submit the serialized model as part of the solution.

c) Predict the ratings of the test data and estimate the prediction quality through Mean Squared Error (MSE). Submit the obtained MSE as part of the solution.

Exercise 5 (2 + 3 points)

Study the code of Albert Au Yeung for matrix factorization by GD (see slide "References" on lecture 06).

a) Apply it to the utility matrix used in lecture 06 (slide "Recall: Utility Matrix"). Do you obtain the same matrices Q and P as shown in the lecture (slide "Latent Factor Models")? Submit as solutions your matrices Q, P and the full matrix R.

b) Modify the code so that the error e is stored (or printed after each iteration), and plot the error over iterations (for matrix in **a)**), as well as the differences of the error in subsequent steps.

[1] `https://en.wikipedia.org/wiki/Stochastic_gradient_descent`
[2] Available on Moodle as Dataset for Problem Set 5 - Ex 4
[3] https://spark.apache.org/docs/2.2.0/api/python/pyspark.mllib.html#pyspark.mllib.recommendation.Rating
[4] https://spark.apache.org/docs/2.2.0/api/python/pyspark.mllib.html#pyspark.mllib.recommendation.ALS

c) Bonus, 3 points: In this code the learning rate α is a constant ($\alpha = 0.0002$). How could you change the code to speed up the convergence? Explain your idea and implement your solution.