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. 4

Exercise 1 - Singular Value Decomposition

	Matrix	Alien	Star Wars	Casablanca	Titanic
Joe	1	1	1	0	0
Jim	3	3	3	0	0
John	4	4	4	0	0
Jack	5	5	5	0	0
Jill	0	0	0	4	4
Jenny	0	0	0	5	5
Jane	0	0	0	2	2

Figure 11.6: Ratings of movies by users

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}}_{V^{\mathrm{T}}}$$

Use the SVD shown above. Suppose a new user Leslie assigns rating 3 to Alien and rating 4 to Titanic, giving us a representation of Leslie in the 'original space' of $[0, 3, 0, 0, 4]$. Find the representation of Leslie in concept space. What does that representation predict about how well Leslie would like the other movies appearing in our example data?

Exercise 2 - Linear Regression

The ordinary least squares regression problem is defined as follows: Given a list of n observations $(x_1, y_1), \ldots, (x_n, y_n)$ with $y_i \in R$ being scalars and $x_i \in R^d$ being d-dimensional vectors.[1] Find a vector $b \in R^d$ that minimizes

$$\sum_{i=1}^{n} (y_i - x_i^T b)^2$$

[1]Note: y_i is often called the response and x_i the regressor or explanatory variable. We try to predict the response y_i based on the explanatory variable x_i. That is, we aim to find a function f such that $f(x_i) \approx y_i$ for each i. In the case of linear regression, the function f corresponds to $f(x) = x^T b$.

a) Prove that the optimal solution of b is given by $b = (\frac{1}{n}\sum_{i=1}^{n} x_i x_i^T)^{-1} \cdot \frac{1}{n}\sum_{i=1}^{n} x_i y_i$. Note: Assume that the regression problem is not degenerated (underdetermined).

b) In the above definition of the linear regression, we assumed a function of the form $f(x) = x^T b$. In this definition, the intercept/offset of the 'line' is not included (i.e. we did not use $g(x) = x^T b + c$). Can we still use the above definition even when we want to compute a potential offset?

Exercise 3 - Alternating Optimization

Given the following rating matrix R of 6 movies and 8 users (rows = movies; columns = users; '?' denotes missing/unknown ratings):

	users							
movies	4	4	?	?	3	2	2	?
	1	?	?	4	?	?	1	5
	?	2	3	3	?	2	1	3
	1	?	1	1	2	?	4	5
	4	4	4	1	1	1	?	?
	?	1	1	?	4	3	?	5

The goal is to find the optimal rank k decomposition. That is, find matrices P and Q minimizing

$$f(P,Q) := \sum_{(i,x)\in R} (r_{i,x} - q_i \cdot p_x^T)^2$$

Here, $(i,x) \in R$ denotes all elements in the matrix that are not missing.

To find the decomposition, you should implement the alternating optimization approach as described on slides 87-90. [2] As default, pick $k = 3$.

a) Initialize the matrices Q and P according to the SVD decomposition of R assuming missing ratings are 0. What is the value of the function $f(P,Q)$?

b) Measure the loss $f(P,Q)$ after each full iteration of the alternating optimization. Plot the value of this loss. How does it behave over time?

c) What happens if the rank k equals to 1, 2 or 4? What happens if we increase k above values of 4?

d) What happens if you initialize the matrices P and Q randomly (random values between 0 and 1) rather than using SVD initialization?

Project task 2 - Song recommendation (Part 1)

The idea behind this project task is do song recommendation via Latent Factor Models. More specifically, our dataset consists of the triplets (user, song, play count) describing how many times a certain user played a certain song. Given these triplets we can easily form a user-song matrix that will be target for a lower rank decomposition. Once we've obtained the latent factors we can multiply them, thereby obtaining predictions for possible play counts. To recommend then a song to a user, we simply look at the songs that have high predicted play count.

The dataset can be obtained from the MillionSong dataset website at `http://labrosa.ee.columbia.edu/millionsong/tasteprofile`.

[2] Chapter 3: Dimensionality Reduction & Matrix Factorization

The dataset is in the format of a text file, where each line contains one triplet (user_id, song_id, play count), and overall has:

- 1,019,318 unique users
- 384,546 unique songs
- 48,373,586 (user, song, play count) triplets

Your tasks are as follows:

- Parse the input text file and create the user-song matrix M. Since the data is quite large you won't be able to fit the matrix in memory. For now, for Part 1 of this task you can restrict yourselves to look at the first n triplets (the first n rows in the input) instead. Use $n = 300000$ by default and make it a parameter.

  In the next tutorial we will see how can one handle the complete dataset, by storing the data in sparse matrix format [3] to avoid out of memory issues. This along with other improvements will be considered in Part 2 of this task.

- Preprocess the data by binning the play counts into b bins. More specifically, replace values in M in the range $[2^i, 2^{i+1} - 1]$ with $i + 1$, for $i = 0 \dots b - 1$. Finally, replace all values in $[2^b, +\infty]$ with b. That means for example play counts in the range $[2, 3]$ get replaced with 2, $[4, 7]$ get replaced with 3, $[8, 15]$ get replaced with 4, etc. This step is necessary to transform our data to mean how much a user relatively likes a certain song. Use $b = 10$ by default and make it a parameter.

- Preprocess the data to avoid the cold start issue. That is, recursively remove all users/songs that have less or equal than 5 occurrences in M, until M no longer changes. Having read $n = 300000$ triples you should get a matrix M of shape $(5693, 10966)$ after this preprocessing step.

- To properly evaluate our approach set-aside 200 randomly selected non-zero datapoints from the matrix as a test set. That is, store their values separately and set them to ? in M.

- Find the latent factors using the alternating optimization approach similar to Exercise 3. Use a default value of $k = 30$ and make it a parameter.

- To evaluate the learned model report the $RMSE = \sqrt{\frac{\sum_i^n (y_i - \hat{y}_i)^2}{n}}$ on the test set [4]. You should expect RMSE of around 1.25 or better depending on the random selection of the test set.

Hint: Using the closed form solution for the regression on this real data might lead to singular values. To avoid this issue perform the regression step with an existing package such as scikit-learn. It is advisable to use ridge regression to account for regularization[5]. Hint: If you are using the scikit-learn package remember to set $fit\_intercept = False$ to only learn the coefficients of the linear regression.

The deadline for this project task is **11.12.2016 23:55**.

[3] e.g. scipy.sparse.coo_matrix
[4] There is a typo on the slides in the RMSE definition
[5] e.g. sklearn.linear_model.Ridge