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. 1

Exercise 1 - Distance functions

Prove or disprove: The following distance functions are metric:

- $dist_1(x, y) = \sum_{i=1}^{D}(x[i] - y[i])$
- $dist_2(x, y) = \sum_{i=1}^{D}(x[i] - y[i])^2$
- $dist_3(x, y) = \sum_{i=1}^{D} \begin{cases} 1 & x[i] = y[i] \\ 0 & else \end{cases}$

Exercise 2 - Min Hashing

	C1	C2	C3	C4
R1	0	1	1	0
R2	1	0	1	1
R3	0	1	0	1
R4	0	0	1	0
R5	1	0	1	0
R6	0	1	0	0

Perform a minhashing of the above shown data – computing the minhash value for every column – with the following orders of rows (permutations):

- p_1: R4, R6, R1, R3, R5, R2
- p_2: R2, R3, R6, R1, R4, R5

Exercise 3 - Locality sensitive Hashing

Given the following signature matrix:

C1	C2	C3	C4	C5	C6	C7
1	2	1	1	2	5	4
2	3	4	2	3	2	2
3	1	2	3	1	3	2
4	1	3	1	2	4	4
5	2	5	1	1	5	1
6	1	6	4	1	1	4

Suppose we use locality-sensitive hashing with three bands of two rows each. Assume there are enough buckets available that the hash function for each band can be the identity function (i.e., columns hash to the same bucket if and only if they are identical in the band). Find all the candidate pairs.

Exercise 4 - Locality Sensitive Hashing

Let f be the function for the S-curve $f_{r,b}(t) = 1 - (1 - t^r)^b$ (b = number of bands; r = number of rows per band).

Let $k = r \cdot b$ be the overall number of hash functions you want to use (accordingly b=k/r).

a) What choice of b and r will satisfy the following requirements:

- We want to find at least 99% of pairs that are 95% similar
- We want to find at most 15% of pairs that are 80% similar
- k is as small as possible

b) Your task is to understand how the shape of the S-curve changes when we vary its parameters. For this purpose, plot the function with a tool of your choice (Matplotlib, Matlab, Excel, ...) and perform the following parameter variations:

- For a fixed $t = 0.90$, and a given r, find the value of b such that $f_{r,b}(t) = t$ holds. What happens to the shape of the S-curve as you change r from 1 to 200?
- Fix the number of hash functions k to 1000. What happens as we increase r from 1 to 100?

Exercise 5 - Jaccard similarity

Suppose we have a set U of n elements, and we randomly choose two subsets S and T, each with m of the n elements.

What is the expected value of the Jaccard similarity of S and T?

Exercise 6 - Hashing for Cosine Similarity

Let us apply the hashing principle for cosine similarity using the following four 'random' vectors:

$$v_1 = [+1, +1, +1, -1] \qquad v_2 = [+1, +1, -1, +1]$$
$$v_3 = [+1, -1, +1, +1] \qquad v_4 = [-1, +1, +1, +1]$$

Compute the sketches/signatures of the following vectors.

1. $a = [2, 3, 4, 5]$
2. $b = [-2, 3, -4, 5]$
3. $c = [2, -3, 4, -5]$

What is the exact angle between each pair of vectors? What is the estimated angle based on the sketches?

Exercise 7 - Hashing for Euclidean distance

Suppose we have points in a 3-dimensional Euclidean space:

$$p_1 = (1, 2, 3), p_2 = (0, 2, 4), p_3 = (4, 3, 2)$$

Consider the three hash functions defined by the three axes (to make our calculations very easy). Let buckets be of length w, with one bucket the interval $[0, w)$ (i.e. the set of points x sucht that $0 \leq x < w$), the next $[w, 2w)$, the previous one $[-w, 0)$, and so on.

(a) For each of the three lines, assign each of the points to buckets, assuming $w = 1$

(b) Repeat part (a), assuming $w = 2$

(c) What are the candidate pairs for the cases $w = 1$ and $w = 2$ when using an AND-construction to combine the hash functions.

(d) What are the candidate pairs for the cases $w = 1$ and $w = 2$ when using an OR-construction?

Project task 1 - Duplicate detection

To goal of this task is to find duplicate songs in the Million Song dataset. You can imagine a scenario were the same song appears on multiple different releases with only small feature variation (e.g. duration or loudness). Your implementation of duplicate detection should satisfy the following requirements:

- Be able to run on a subset of N songs from the complete dataset, where N is a user set parameter. We expect reasonable execution times for N of at least 100000 songs
- Be able to run on a subset of features given by the user. Use the following features as a baseline: ['duration', 'end_of_fade_in', 'key', 'loudness', 'mode', 'start_of_fade_out', 'tempo', 'time_signature'], with the ability to easily add or remove further features.
- For duplicate detection use LSH with cosine similarity:
  - Generate duplicate **candidates** based on LSH with b bands and r rows per band
  - Refine the candidates by computing the exact cosine distance
  - Report all pairs/duplicates with cosine angle $< \epsilon$

  Use the following default values $r = 20$, $b = 3$ and $\epsilon = 5$, however allow these to be changed by the user as well.
- Optional: Compare the running time with a naive implementation that compares all possible pairs
- Optional: Additionally, allow the user to specify a maximum cosine distance ϵ' instead of the maximum cosine angle ϵ

Tip: Don't forget to normalize your features (between 0 and 1) before running duplicate detection.

You can run your implementation initially, for testing purposes, on the small subset of the dataset, but it should be able to run on the complete Million Song Dataset. Further instructions on how to download the complete dataset will be provided in the next days on Moodle, in the meantime make sure your implementation works at least on the subset.

Project Organization

As students you have access to GitLab accounts on https://gitlab.lrz.de/. If you have not already activated your account there please do so. Afterwards, one student from each group should create a project with the following name "group_n" where n is your group-number according to the Moodle group selection. After creating the project please add your fellow group mates as well as the account "mmds" with "developer" privileges.

You should create a separate folder and notebook for each project task, starting with this one. You can have auxiliary python files where you define your own functions, that you later import, however the task should always be presented in a single runnable notebook.