Mining Massive Data (SS2025)

Pen & Paper 2

Availability date: 20.03.2025
Exercise session: 03.04.2025 (attendance is mandatory)
Checkmark & upload date: 02.04.2025, 23:00

Number of tasks: 7
Maximum achievable points: 30 points

Submission & Grading

This pen & paper exercise is graded. You have to indicate which tasks you have solved and upload your solutions by the *checkmark & upload deadline* (see the header) on the course's Moodle webpage [3]. To be awarded the points for these tasks, you have to be present in the respective pen & paper session, be ready to present your solutions, and, if you are selected, provide a concise and comprehensible presentation of your solution which demonstrates your understanding of the problem & solution. The selection process will mainly be uniformly at random. **If you fail to be present in the session or are not able to explain your solution, you will not be awarded any points for the whole pen & paper session.**

Because of the large number of students enrolled in the course, we do not provide individual feedback on your solutions by default but we will provide sample solutions. If you want feedback on your solutions beyond the sample solutions, contact us via email (see the contact details below).

Questions

If you have any questions, please ask them in the respective Moodle forum. If you don't get an answer from your colleagues or us within 48 hours, reach out to us via email (ensure that you start the subject of your email with `[MMD25]`):

- Timo Klein (`timo.klein@univie.ac.at`)

Tasks

Task 1: Similarities from Min-hash Signatures I

(maximum achievable points: 4 points)

In this exercise we study the effect of the number of min-hash functions used on the accuracy of estimating the Jaccard-similarity of documents. For that purpose, download the template code [4] and the prepared data [1]. Upon execution, the code loads data derived from the *20 newsgroups* dataset and a pre-computed min-hash signature matrix for the dataset (using 10.000 hash-functions, so the corresponding matrix has 10.000 rows). Code for computing the Jaccard-similarity of the documents is already in place (the similarities are stored in the matrix `sim` which is a square matrix containing all pair-wise similarities).

(a) **[3 point(s)]** Compute approximations to the Jaccard-similarity from the signature matrix using $1, 100, 200, 300, \ldots$ hash-functions (i.e., pick a subset of the rows of signature matrix for computing the similarities). Create a plot showing the sum of squared errors between `sim` and the approximate Jaccard-similarites for the different considered number of hash-functions. What do you observe?

You can use the functions `pdist` *and* `squareform` *from* `scipy.spatial.distance`*.*

(b) **[1 point(s)]** Compute the 10 nearest neighbors for document `0` from `sim` and from the approximate Jaccard-similarity for the different numbers of considered hash-functions. What do you observe?

Task 2: Similarities from Min-hash Signatures I

(maximum achievable points: 4 points)

[4 point(s)] Assume that we want to approximate the Jaccard-similarity of two documents from their min-hash signatures. We want to estimate how many hash functions we need to use for the signature matrix such that with at least probability $\delta = 0.99$ the absolute error between the approximate Jaccard-similarity computed from the min-hash signature and the true Jaccard-similarity is at most $\epsilon = 0.01$. Compute a lower bound on the number of hash functions to use.

Hint: Hoeffding's inequality.

Task 3: Min-Hashing Using MapReduce

(maximum achievable points: 4 points)

[4 point(s)] Suppose we want to use a MapReduce framework to compute min-hash signatures of a set of documents represented by a binary matrix $\mathbf{M}$, such that each column corresponds to a document and each row to a shingle. If the matrix is stored in chunks that correspond to some columns, then it is quite easy to exploit parallelism. Each map task gets some of the columns and all the hash functions, and computes the min-hash signatures of its given columns. However, suppose the matrix were chunked by rows, so that a `map` function is given the hash functions and a set of rows to work on. Design `map` and `reduce` functions to exploit MapReduce with data in this form.

Credit: Exercise taken from the MMDS book (with slight adjustments).

Task 4: LSH Using MapReduce

(maximum achievable points: 4 points)

Suppose we wish to implement LSH by MapReduce. Specifically, assume chunks of the signature matrix consist of columns, and elements are key-value pairs where the key is the column number and the value is the signature itself (i.e., a vector of values).

(a) **[2 point(s)]** Show how to produce the buckets for all the bands as output of a single MapReduce process.

Hint: Remember that a `map` function can produce several key-value pairs from a single element.

(b) **[2 point(s)]** Show how another MapReduce process can convert the output of (a) to a list of pairs that need to be compared. Specifically, for each column i, there should be a list of those columns $j > i$ with which i needs to be compared.

Credit: Exercise taken from the MMDS book (with slight adjustments).

Task 5: LSH for Manhattan Distance

(maximum achievable points: 4 points)

Let $x, y \in \{0,1\}^d$ be binary vectors of length d. The *Hamming distance* between vectors x, y is $d_H(x,y) = |\{j | j = 1, \ldots, d : x_j \neq y_j\}|$.

(a) **[2 point(s)]** Show that d_H is indeed a distance. Consider the hash function family $\mathcal{H} = \{h_i\}_{i=1}^{d}$, where $h_i(x) = x_i$. Prove that $\mathcal{H}$ defines an LSH family. What are its "sensitivity parameters"?

(b) **[2 point(s)]** For $d = 10$, can you boost the hash family such that for $d_H(x, y) \leq 2$, the probability of collision is at least 95% and for $d_H(x, y) \geq 4$ at most 5%? If not, prove your statement, if yes, provide a possible construction.

Task 6: LSH for the Euclidean Distance

(maximum achievable points: 5 points)

[5 point(s)] Suppose we have points in a 3-dimensional Euclidean space: $\mathbf{p}_1 = (1, 2, 3)$, $\mathbf{p}_2 = (0, 2, 4)$, and $\mathbf{p}_3 = (4, 3, 2)$. Consider the three hash functions defined by the three axes (to make our calculations very easy). Let buckets be of length a, with one bucket the interval $[0, a)$ (i.e., the set of points x such that $0 \leq x < a$), the next $[a, 2a)$, the previous one $[-a, 0)$, and so on.

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

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

(c) What are the candidate pairs for the cases $a = 1$ and $a = 2$?

(d) For each pair of points, for what values of a will that pair be a candidate pair?

Credit: Exercise taken from the MMDS book [2].

Task 7: LSH for the Cosine Distance

(maximum achievable points: 5 points)

[5 point(s)] Consider LSH for the cosine distance, i.e., read and understand section 3.7.2 of the MMDS book [2]. Be prepared to explain how hashing works and explain how this can define a locality-sensitive family of functions. Explain the corresponding proof.

Hint: It might be useful to prepare several sketches that help you explain your thoughts in the exercise session.

References

[1] *Data for studying certain aspects of min-hasing*. URL: https://tschiatschek.net/courses/MMD/SS2023/PP2/data.hdf5.

[2] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge university press, 2020. URL: http://www.mmds.org/.

[3] *Mining Massive Data Moodle Page*. URL: https://moodle.univie.ac.at/course/view.php?id=445688.

[4] *Source code basis for studying certain aspects of min-hashing*. URL: https://tschiatschek.net/courses/MMD/SS2023/PP2/task1-template.py.