Mining Massive Data (SS2025)

Pen & Paper 1

Availability date: 06.03.2025
Exercise session: 20.03.2025 (attendance is mandatory)
Checkmark & upload date: 19.03.2025, 23:00

Number of tasks: 6
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 [2]. 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: The search for ESP

(maximum achievable points: 4 points)

Consider the search for extra-sensory perception (ESP) example from the lecture in a slightly modified version, i.e., a person is considered to have extra-sensory perception if it can guess the color of 6 randomly selected cards (the cards can be *red*, *black*, or *blue*). Assume that each individual guesses *red*, *black*, and *blue* for each card uniformly at random with probability $p = 1/3$.

(a) **[1 point(s)]** What is the probability that at least 1 participant out of 1000 gets all 6 cards right?

(b) **[1 point(s)]** What is the probability that a participant succeeds in getting 12 cards right? (6 cards in a first trial, and 6 cards in a second trial).

(c) **[2 point(s)]** Assume that there actually exists a person with some form of ESP in the sense that they get 6 cards right with probability 0.01. How can we distinguish between a person with ESP and one without? Provide a description of an experiment to conduct to this end. Which parameters of the experiment characterize how certain we are about a person having ESP? Can we ever be certain that a person has ESP?

Hint: This task requires knowledge from Lecture 1.

Task 2: MapReduce – Counting Palindromes

(maximum achievable points: 4 points)

Consider a large collection of text files $\mathcal{D} = \{\mathbf{t}_1, \ldots, \mathbf{t}_n\}$, where each $\mathbf{t}_i$ is a text file. We want to compute the number of appearances of palindromes in the collection of files. A palindrome is any sequence of characters separated by white spaces that reads the same backwards as forwards. As the amount of data is large, we want to use a map-reduce approach to compute the counts.

(a) **[2 point(s)]** Specify the `map` and `reduce` functions for computing the counts in the form of pseudo-code. Specify the inputs and outputs for the functions.

(b) [**2 point(s)**] Implement the map and reduce functions in Python. Build on the code provided in [3]. The template code downloads several books from "Project Guttenberg" and provides their content to the map function (`mapper`). The results of the mapper functions are sorted and passed to reduce function (`reducer`). The parts of the code that you have to adjust are marked by comments. Make sure that your code executes without error. Provide examples of the found palindromes.

Hint: This task requires knowledge from Lecture 2.

Task 3: MapReduce – Mean Absolute Error of Linear Regression

(maximum achievable points: 6 points)

Let $\mathcal{D} = \{(\mathbf{x}_1, y_i), \ldots, (\mathbf{x}_n, y_n)\}$ be a dataset, where $\mathbf{x}_i \in \mathbb{R}^d$ is the set of features (explanatory variables) and $y_i \in \mathbb{R}$ is the scalar target of the ith data point, respectively. Assume that we want to evaluate the mean absolute error of a linear regressor with parameter vector $\mathbf{w} \in \mathbb{R}^d$, i.e., compute

$$R(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n} |y_i - \mathbf{w}^T\mathbf{x}_i|.$$

Assume some parameter vector $\mathbf{w}$ and that we want to evaluate $R(\mathbf{w})$ but the dataset $\mathcal{D}$ is so large that it is stored across multiple machines and we want to use a map-reduce approach to evaluate it. In particular, each mapper is provided a subset of all data points.

(a) [**2 point(s)**] Specify the `map` and `reduce` functions for computing $R(\mathbf{w})$ in the form of pseudo-code. Specify the inputs and outputs for the functions.

(b) [**2 point(s)**] Which property of the objective function of ordinary least squares enables us to easily parallelize its computation as in the previous subtask? What kinds of objectives functions would make the parallelization more complicated (construct an example)? Justify your answer.

(c) [**2 point(s)**] Implement the map and reduce functions in Python. Build on the code provided in [4]. Make sure that your code executes without error.

Hint: This task requires knowledge from Lecture 2. You need to install scikit-learn to run the code-template.

Task 4: Min-hashing for Approximate Retrieval

(maximum achievable points: 5 points)

Assume that we want to use min-hashing for approximate retrieval of text-documents. To this end, we assume that each text-document is represented by an indicator vector (bag-of-words) indicating which words from a global dictionary (consisting of all the words in the text-documents) are present in a specific document.

Assume the following text-documents $\mathbf{t}_1, \ldots, \mathbf{t}_4$:

$\mathbf{t}_1$	this is mining massive data
$\mathbf{t}_2$	mining massive data is cool
$\mathbf{t}_3$	machine learning is cool
$\mathbf{t}_4$	I just had a good lunch

(a) **[1 point(s)]** Determine the global dictionary and represent each text-document as a bag-of-words.

(b) **[2 point(s)]** Consider the hash functions

$$\begin{aligned} h_1(x) &= 3x + 1 \mod 14, \text{ and} \\ h_2(x) &= 5x + 3 \mod 14. \end{aligned}$$

Compute the signature matrix of the documents.

(c) **[2 point(s)]** Compute the similarity of the documents. What is the most similar document to $\mathbf{t}_1$? What about $\mathbf{t}_3$? What is the similarity of $\mathbf{t}_4$ to the other documents?

Hint: This task requires knowledge from Lecture 3.

Task 5: Curse of Dimensionality

(maximum achievable points: 5 points)

When the number of features p is large, there tends to be a deterioration in the performance of k nearest neighbor and other local approaches that perform prediction using only observations that are near the test observation for which a prediction must be made. This phenomenon is known as the curse of dimensionality, and it ties into the fact that so-called non-parametric approaches often perform poorly when p is large. We will now investigate this curse.

(a) **[1 point(s)]** Suppose that we have a set of observations, each with measurements on $p = 1$ feature, X. We assume that X is uniformly (evenly) distributed on $[0, 1]$. Associated with each observation is a response value. Suppose that we wish to predict a test observation's response using only observations that are within 5% of the range of X closest to that test observation. For instance, in order to predict the response for a test observation with $X = 0.6$, we will use observations in the range $[0.575, 0.625]$. On average, what fraction of the available observations will we use to make the prediction?

(b) **[1 point(s)]** Now suppose that we have a set of observations, each with measurements on $p = 2$ features, X_1 and X_2. We assume that (X_1, X_2) are uniformly distributed on $[0, 1] \times [0, 1]$. We wish to predict a test observation's response using only observations that are within 5% of the range of X_1 and within 5% of the range of X_2 closest to that test observation. For instance, in order to predict the response for a test observation with $X_1 = 0.6$ and $X_2 = 0.35$, we will use observations in the range $[0.575, 0.625]$ for X_1 and in the range $[0.325, 0.75]$ for X_2. On average, what fraction of the available observations will we use to make the prediction

(c) **[3 point(s)]** Now suppose that we wish to make a prediction for a test observation by creating a p-dimensional hypercube centered around the test observation that contains, on average, 5% of the training observations. For $p \in \{1, 2, 5, 10, 100\}$, what is the length of each side of the hypercube? Comment on your answer.

Note: A hypercube is a generalization of a cube to an arbitrary number of dimensions. When $p = 1$*, a hypercube is simply a line segment, when* $p = 2$ *it is a square, and when* $p = 100$ *it is a* 100*-dimensional cube.*

Hint: This task requires knowledge from Lecture 3.

Credits: This task is taken (and slightly adjusted)from the book "Introduction to Statistical Learning" by G. James, D. Witten, T. Hastie, and R. Tibshirani.

Task 6: MapReduce – Matrix-vector Multiplication

(maximum achievable points: 6 points)

[Reading exercise] We want to use MapReduce for computing the matrix-vector product $\mathbf{Mv}$ of a matrix $\mathbf{M} \in \mathbb{R}^{n \times d}$ and a vector $\mathbf{v} \in \mathbb{R}^d$. Therefore, read sections 2.3.1 and 2.3.2 in the MMDS book [1].

(a) [**3 point(s)**] Explain how map-reduce can be applied when the matrix **M** is distributed among multiple-machines but the vector **v** can be fit into the main memory of a compute node. Specify the `map` and `reduce` functions for computing the matrix-vector product in the form of pseudo-code. Specify the inputs and outputs for the functions.

(b) [**3 point(s)**] Consider the case in which **v** is too large to be read into the main memory of a compute node and must be partitioned into stripes as explained in section 2.3.2 in the MMDS book. Specify the `map` and `reduce` functions for computing the matrix-vector product in the form of pseudo-code. Specify the inputs and outputs for the functions.

Hint: This task requires knowledge from Lecture 2.

References

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

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

[3] *Source code basis for counting palindromes.* URL: `https://tschiatschek.net/courses/MMD/SS2024/PP1/task2-template.py`.

[4] *Source code basis for evaluating OLS.* URL: `https://tschiatschek.net/courses/MMD/SS2024/PP1/task3-template.py`.