Machine Learning

2021-2022

Home Assignment 1

Yevgeny Seldin **Sadegh Talebi**
Department of Computer Science
University of Copenhagen

The deadline for this assignment is **30 November 2021, 22:00**. You must submit your *individual* solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your full source code in the PDF file, only selected lines if you are asked to do so.
- A .zip file with all your solution source code with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF. The programming language of the course is Python.
- IMPORTANT: Do NOT zip the PDF file, since zipped files cannot be opened in speed grader. Zipped PDF submissions will not be graded.
- Your PDF report should be self-sufficient. I.e., it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Handwritten solutions will not be accepted, please use the provided latex template to write your report.

1 Make Your Own (7 points)

Imagine that you would like to write a learning algorithm that would predict the final grade of a student in the Machine Learning course based on their profile, for example, their grades in prior courses, their study program, etc. Such an algorithm would have been extremely useful: we could save significant time on grading and predict the final grade when the student just signs up for the course. We expect that the students would also appreciate such service and avoid all the worries about their grades. Anyhow,

1. What profile information would you collect and what would be the sample space $\mathcal{X}$?
2. What would be the label space $\mathcal{Y}$?
3. How would you define the loss function $\ell(y', y)$?
4. Assuming that you want to apply K-Nearest-Neighbors, how would you define the distance measure $d(x, x')$?
5. How would you evaluate the performance of your algorithm? (In terms of the loss function you have defined earlier.)
6. Assuming that you have achieved excellent performance and decided to deploy the algorithm, would you expect any issues coming up? How could you alleviate them?

There is no single right answer to the question. The main purpose is to help you digest the definitions we are working with. Your answer should be short, no more than 2-3 sentences for each bullet point. For example, it is sufficient to mention 2-3 items for the profile information, you should not make a page-long list.

2 Digits Classification with K Nearest Neighbors (40 points)

In this question you will implement and apply the K Nearest Neighbors learning algorithm to classify handwritten digits. You should make your own implementation (rather than use libraries), but it is allowed to use library functions for vector and matrix operations.

Preparation

- Download `MNIST-5-6-Subset.zip` file from Absalon. The file contains `MNIST-5-6-Subset.txt`, `MNIST-5-6-Subset-Labels.txt`, `MNIST-5-6-Subset-Light-Corruption.txt`, `MNIST-5-6-Subset-Moderate-Corruption.txt`, and `MNIST-5-6-Subset-Heavy-Corruption.txt` files.
- `MNIST-5-6-Subset.txt` is a space-separated file of real numbers (written as text).[1] It contains a 784×1877 matrix, written column-by-column (the first 784 numbers in the file correspond to the first column; the next 784 numbers are the second column, and so on).
- Each column in the matrix above is a 28×28 grayscale image of a digit, stored column-by-column (the first 28 out of 784 values correspond to the first column of the 28×28 image, the next 28 values correspond to the second column, and so on). Test yourself: reshape the first column into a 28×28 matrix and display it as an image - did you get an image of digit "5"?
- `MNIST-5-6-Subset-Labels.txt` is a space-separated file of 1877 integers. The numbers label the images in `MNIST-5-6-Subset.txt` file: the first number ("5") is the number drawn in the image corresponding to the first column; the second number corresponds to the second column, and so on.
- `MNIST-5-6-Subset-Light-Corruption.txt`, `MNIST-5-6-Subset-Moderate-Corruption.txt`, and `MNIST-5-6-Subset-Heavy-Corruption.txt` are corrupted versions of the digits in `MNIST-5-6-Subset.txt`, the order is preserved. It is a good idea to visualize the corrupted images to get some feeling of the corruption.

High-Level Idea We pursue several goals in this question:

1. Get your hands on implementation of K-NN.
2. Explore fluctuations of validation error as a function of the size of validation set.
3. Explore dependence of validation error on the number of neighbors K.
4. Explore dependence of validation error on the number of neighbors K when the data are corrupted.

[1] It is a subset of digits '5' and '6' from the famous MNIST dataset (LeCun et al.).

Detailed Instructions

Tast #1 In this task we explore fluctuations of validation error as a function of the size of validation set and the dependence of the validation error on the number of neighbors K.

In order to explore fluctuations of the validation error as a function of the size of the validation set we use the following construction:

- Use the first 100 digits for training the K-NN model.
- Consider five validation sets, where for $i \in \{1, \ldots, 5\}$ the set i consists of digits $100 + i \times n + 1, \ldots, 100 + (i + 1) \times n$, and where n is the size of each of the five validation sets (we will specify n in a moment).
- Calculate the validation error for each of the sets as a function of K, for $K \in \{1, \ldots, 50\}$. Plot the validation error for each of the five validation sets as a function of K in the same figure (you will get five lines in the figure).
- Execute the experiment above with $n \in \{10, 20, 40, 80\}$. You will get four figures for the four values of n, with five lines in each figure.
- Create one more figure, where for each n you plot the variance of the validation error over the five validation sets, as a function of K. You will get four lines in this figure, one for each n.
- What can you say about fluctuations of the validation error as a function of n?
- What can you say about the prediction accuracy of K-NN as a function of K?
- To include in the report: four figures with five lines, as described above, where each figure corresponds to a different value of n, plus one figure with the variance, plus an answer to the two questions above.

Tast #2 In this question we explore the influence of corruptions on the performance of K-NN and on the optimal value of K. Here are the instructions:

- Take the uncorrupted set, $n = 80$, and construct training and validation sets as above. Report a figure with five lines for the five validation sets, as a function of K, for $K \in \{1, \ldots, 50\}$.

- Repeat the experiment with the lightly corrupted set (both training and test images should be taken from the lightly corrupted set), then with the moderately corrupted set, and then with the heavily corrupted set. Produce a figure as above for each of the experiments. (Four figures in total, including the previous bullet point.)
- Discuss how corruption magnitude influences the prediction accuracy of K-NN and the optimal value of K.
- To include in the report: four figures and the discussion mentioned above.

IMPORTANT: Please, remember to include axis labels and legend in your plots!

Practical Details and Some Practical Advice

- Use square Euclidean distance to calculate the distance between the images. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are two 784-long vectors representing two images, then the distance is $\|\mathbf{x}_1 - \mathbf{x}_2\|^2 = (\mathbf{x}_1 - \mathbf{x}_2)^T(\mathbf{x}_1 - \mathbf{x}_2)$.
- If you work with an interpreted programming language, such as Python, do your best to use vector operations and avoid for-loops as much as you can. This will make your code orders of magnitude faster.
- Assume that $\mathbf{X} = \begin{pmatrix} \begin{pmatrix} | \\ \mathbf{x}_1 \\ | \end{pmatrix}, \cdots, \begin{pmatrix} | \\ \mathbf{x}_n \\ | \end{pmatrix} \end{pmatrix}$ is a matrix holding data vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and you want to calculate distances between all these points and a test point $\mathbf{x}$. *Do your best to avoid a for-loop!* One way of doing so is to create another matrix $\mathbf{X}' = \begin{pmatrix} \begin{pmatrix} | \\ \mathbf{x} \\ | \end{pmatrix}, \cdots, \begin{pmatrix} | \\ \mathbf{x} \\ | \end{pmatrix} \end{pmatrix}$ and calculate all n distances in one shot using matrix and vector operations.
- Note that for a single data point you can compute the output of K-NN for all K in one shot using vector operations. No need in for-loops!
- You may find the following functions useful:
  - Built-in sorting functions for sorting the distances.
  - Built-in functions for computing a cumulative sum of elements of a vector `v` (for computing the predictions of K-NN for all K at once).
- It may be a good idea to debug your code with a small subset of the data.

Optional, not for submission: You are very welcome to experiment further with the data.

3 Illustration of Markov's, Chebyshev's, and Hoeffding's Inequalities (20 points)

2.a Make 1,000,000 repetitions of the experiment of drawing 20 i.i.d. Bernoulli random variables $X_1, \ldots, X_{20}$ (20 coins) with bias $\frac{1}{2}$ and answer the following questions.

1. Plot the empirical frequency of observing $\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha$ for $\alpha \in \{0.5, 0.55, 0.6, \ldots, 0.95, 1\}$.
2. Explain why the above granularity of α is sufficient. I.e., why, for example, taking $\alpha = 0.51$ will not provide any extra information about the experiment.
3. In the same figure plot the Markov's bound[2] on $\mathbb{P}\big(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha\big)$.
4. In the same figure plot the Chebyshev's bound[3] on $\mathbb{P}\big(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha\big)$. (You may have a problem calculating the bound for some values of α. In that case and whenever the bound exceeds 1, replace it with the trivial bound of 1, because we know that probabilities are always bounded by 1.)
5. In the same figure plot the Hoeffding's bound[4] on $\mathbb{P}\big(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha\big)$.
6. Compare the four plots.
7. For $\alpha = 1$ and $\alpha = 0.95$ calculate the exact probability $\mathbb{P}\big(\frac{1}{20}\sum_{i=1}^{20} X_i \geq \alpha\big)$. (No need to add this one to the plot.)

2.b Repeat the question with $X_1, \ldots, X_{20}$ with bias 0.1 (i.e., $\mathbb{E}[X_1] = 0.1$) and $\alpha \in \{0.1, 0.15, \ldots, 1\}$.

2.c Discuss the results.

Do not forget to put axis labels and a legend in your plot!

4 Basic Linear Algebra (13 points)

Let $h_{w,b}$ be a hyperplane given by the equation $w^T x + b = 0$. (I.e., $h_{w,b}$ is the set of points $\big\{x : w^T x + b = 0\big\}$.)

Calculate the distance from the hyperplane to the origin.

[2] Markov's bound is the right hand side of Markov's inequality.
[3] Chebyshev's bound is the right hand side of Chebyshev's inequality.
[4] Hoeffding's bound is the right hand side of Hoeffding's inequality.

5 Regression (20 points)

A cannonball is shot from a cannon that is located at the origin (the coordinates of the cannon are $(0, 0)$). It is then observed at the following locations (x, y): $(1, 14), (2, 21), (3, 25), (4, 35), (5, 32)$, where x is the distance from the cannon and y is the height measurement. The x-measurements are precise, but the y-measurements are noisy.

It is well-known that in presence of gravitation cannonballs fly on parabolic trajections. Use regression to estimate the trajectory of the flight and location of the expected fall.

Report the equation describing the parabolic trajection, the distance from the cannon where the cannonball is expected to fall, and a visualization (a plot with the estimated parabolic trajection and the observed locations). **The parabolic trajection in your solution must pass through the origin** $(0, 0)$**, because the location of the cannon is known with certainty.** It is part of the exercise for you to understand what you need to do to make this happen.

References

Y. LeCun, C. Cortes, and C. J. C. Burges. The mnist database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.