Machine Learning A

2021-2022

Home Assignment 1

Yevgeny Seldin **Christian Igel**
Department of Computer Science
University of Copenhagen

The deadline for this assignment is **14 September 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 source code in the PDF file.
- 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 (10 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 (45 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 $i \in \{1, \dots, 5\}$ the set i consists of digits $100 + i \times n + 1, \dots, 100 + (i + 1) \times n$, 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 $K \in \{1, \dots, 50\}$. 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 $K \in \{1, \dots, 50\}$.

- 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 $\mathbf{X} = \left(\begin{pmatrix} | \\ \mathbf{x}_1 \\ | \end{pmatrix}, \cdots, \begin{pmatrix} | \\ \mathbf{x}_n \\ | \end{pmatrix}\right)$ is a matrix holding data vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$ 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 $\mathbf{X}' = \left(\begin{pmatrix} | \\ \mathbf{x} \\ | \end{pmatrix}, \cdots, \begin{pmatrix} | \\ \mathbf{x} \\ | \end{pmatrix}\right)$ 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 $\mathbf{v}$ (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 Linear regression (45 points)

Consider the data $S = \{(x_1, y_1), \ldots, (x_N, y_n)\}$ in the file `PCB.dt`, which contains the concentrations of polychlorinated biphenyl (PCB) residues in lake trouts from the Cayuga Lake, NY, as reported by Bache et al. (1972). "The ages of the fish were accurately known, because the fish are annually stocked as yearlings and distinctly marked as to year class. Each whole fish was mechanically chopped, ground, and thoroughly mixed, and 5-gram samples taken. The samples were treated and PCB residues in parts per million (ppm) were estimated using column chromatography" (Bates and Watts, 1988).

Each line in `PCB.dt` is one training pattern. The first number is the input (x), the age of the fish in years, and the second is the corresponding output / target / label (y), the PCB concentration (in ppm).

Tasks

1. Implement the linear regression algorithm as described in the lecture (although using the pseudo-inverse is not the best way, e.g., via **QR**-decomposition would be preferable). For vector and matrix operations, such as computing the inverse of a matrix, you can use high-level (library) functions (e.g., from NumPy).

2. The task is to build a model $h : \mathbb{R} \to \mathbb{R}$ of the data in `PCB.dt`. The model should be of the form
$$h(x) = \exp(ax + b) \tag{1}$$
with parameters $a, b \in \mathbb{R}$.

   The parameters can be learned using linear regression in the following way. Before applying linear regression, transform the output data (the y values) by applying the natural logarithm.[2] Then build an affine linear model using the transformed targets. By doing so, you effectively learn the (non-linear) model (1).

   That is, you have to perform the following steps:

   - Construct the data set $S' = \{(x_1, \ln y_1), \ldots, (x_N, \ln y_n)\}$.
   - Fit a model $h'(x) = ax + b$ to S'.
   - Obtain the final model as $h(x) = \exp(h'(x))$

[2] Linear regression without this transformation gives a lower error. Do not get confused by this: We train for a low error "on logarithmic scale", and minimizing this error may not minimize the error on the original scale, which is the error you should consider in this exercise. The idea is that the data looks more linear on logarithmic scale, this is why we – and other researchers – consider a model of the form (1).

Build the model, report the two parameters of the model as well as the mean-squared-error of the model h computed over the training data set S.

3. Plot the data and the model output. The plot must have proper axis labels and a legend. In *all* plots in this assignment, plot the logarithm of the PCB concentration versus the (not transformed) age.

4. Compute the coefficient of determination R^2 as

$$1 - \frac{\sum_{i=1}^{N}(y_i - h(x_i))^2}{\sum_{i=1}^{N}(y_i - \overline{y}))^2} \quad , \tag{2}$$

where $\overline{y}$ denotes the mean of the training labels. Discuss this quantity. What does it mean if R^2 is 1 and especially if R^2 is 0? Can R^2 be negative?

5. Now let us build a non-linear model

$$h(x) = \exp(a\sqrt{x} + b) \tag{3}$$

with $a, b \in \mathbb{R}$. This is the same as before plus additionally applying the non-linear input transformation $x \mapsto \sqrt{x}$ to the input data.

One can view this as the inputs x being mapped to a feature space $\mathcal{Z}$ by the *feature map* $\phi(x) = \sqrt{x}$.

Build the model and report the mean-squared-error. Then plot the target (on logarithmic scale) and the model output *over the original inputs* (linear scale). That is, the unit of the x-axis should be years.

Compute R^2 for the new model and the transformed labels. Discuss the result in comparison to the previous model.

Deliverables: Source code; plot of data and model output; mean-squared error, parameters of regression model, discussion of R^2; mean-squared error of model with transformed inputs, plot of data and the model given by the second model (note axis scaling in years), comparison of R^2 values of the two different models

References

C. A. Bache, J. W. Serum, W. D. Youngs, and D. J. Lisk. Polychlorinated biphenyl residues: Accumulation in cayuga lake trout with age. *Science*, 177 (4055):1191–1192, 1972.

D. M. Bates and D. G. Watts. *Nonlinear Regression Analysis and Its Applications*, volume 2. Wiley New York, 1988.

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