Machine Learning Methods

Exercise 1

October 23, 2025

1 Part A: Calculus

1.1 Calculus-Q1

CQ1.1 Use the chain rule to calculate the gradient of

$$h(\boldsymbol{\sigma}) = \frac{1}{2}\|f(\boldsymbol{\sigma}) - y\|^3$$

where $\boldsymbol{\sigma} \in \mathbb{R}^m$ and f is some arbitrary function from $\mathbb{R}^m$ to $\mathbb{R}^n$.
CQ1.2 Compute the expression in the case where:

$$f(\boldsymbol{\sigma}) = \begin{bmatrix} \sigma_1 \cdot \sigma_2 \\ \sigma_1^2 + \sigma_2^2 \\ \sigma_1 \end{bmatrix}$$

$$\boldsymbol{\sigma} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

1.2 Calculus-Q2

The softmax function $S : \mathbb{R}^k \to [0,1]^k$, is defined as follows:

$$S(x)_j = \frac{e^{x_j}}{\sum_{l=1}^{k} e^{x_l}}$$

This function takes an input vector $x \in \mathbb{R}^d$ and outputs a probability vector (non-negative entries that sum up to 1), corresponding to the weight of original entries of x.
CQ2: Calculate the Jacobian of the softmax function S.

2 Code Exercise: General Description

In this exercise you will obtain practical experience of core machine learning ideas including: *Empirical Risk Minimization* (ERM), *train, test sets*, and the *bias-complexity tradeoff*. Continuing with the prophet theme explored in the lecture, you will use *ERM* on your training set to select a prophet out of a hypothesis class. Then, you will estimate the error of the chosen prophet on the *test set*. Another aim of this exercise is to give you some familiarity with *numpy*.

You are expected to hand a report, no longer than 5 pages, describing your experiments and answers to the questions in the exercise. The submission guidelines are described in Sec. 6, please read them carefully.

We provided you with a skeleton code for this exercise, you must complete the necessary code within it only. Do not add any files or imports. For each question, print the necessary values for it in the question function.

Note 1: You **must** explain your results in each question! Unless specified otherwise, an answer without explanation will **not** be receive any points.

Note 3: You will probably find the following NumPy functions useful: *np.random.choice*, *np.random.uniform* and *np.random.randint*. We advise you to briefly try them before starting the exercise.

3 Creating the Testing Environment

All answers in this section **do not** require an explanation.

3.1 Task & Data

Our prophets will aim to predict the outcome of basketball games. The games are sequentially ordered, the input data ($\mathcal{X}$) is an integer describing the game id. The possible outcomes ($\mathcal{Y}$) are 1 if the home (first) team wins and 0 if the away (second) team wins. For the purposes of this questions, there can be no ties.

For example, say we only have 5 games, where the first team wins in the first 3 and the second team wins the last 2. Then, $X = (0, 1, 2, 3, 4)$ and $Y = (1, 1, 1, 0, 0)$.

NOTE: Due to the way we implement our prophets and the simplicity of the data, our implementation does not need to define $\mathcal{X}$ at all, and will only use $\mathcal{Y}$.

NOTE: In practice, both the data and the prophets are represented as boolean values, with False as 0 and True as 1.

3.2 Data

With pickle [1] load data.pkl. Inside are the results of 10,000 games in *train_set* and a 1,000 results in the *test_set*. The results are represented as *numpy* arrays of 0 and 1 for wins and losses accordingly

3.3 The Prophets

Each prophet has true risk *1-p*. For this data distribution, it will predict the correct outcome with probability *p*, and a wrong answer with probability *1-p*. For each question read the appropriate prophets. The prophets are represented as numpy arrays in the following shape: $[num\_prophets, num\_games]$. We also provided you the true risk of each prophet

4 Questions

- In the following questions you may be asked to sample a smaller number of prophets or a smaller number of games then what's presented when loading the dataset and the prophets, in these cases, sample the prophets and the games uniformly.
- In some of your experiments you may get that several prophets win the same number of games, yet you have to choose between them by ERM. In these cases, sample a prophet uniformly from among those that minimize the risk.

Scenario 1: Two prophets, One game.

Load the file *scenario_one_and_two_prophets.pkl*
You have two prophets, one prophet with an 20% error and another prophet with 40% error. You decide to evaluate each prophet on a single random game (i.e. train set of size of 1), using the ERM algorithm to choose the best one. Repeat this experiment 100 times each time selecting the best prophet using the ERM algorithm. For each experiment:

1. Select a prophet based on the ERM algorithm Evaluate your selected prophet on the test set and compute its average error. Calculate the approximation[2] and estimation error[3]

In your report

- Report the average error of the selected prophets over the experiments.
- In how many experiments did you choose the best[4] prophet?

[1] You can read more about loading python object with pickle here.
[2] approximation error is the true risk of the best available prophet
[3] estimation error is the difference between the true risk of the best and selected prophet
[4] 'best' in terms of the true risk which is provided for each prophet

- Calculate the approximation and mean estimation errors over the experiments.

Scenario 2: Two Prophets, Ten Games.

Repeat the experiment from Scenario 1, but this time choose the best prophet by ERM using a train set of 10 games. Repeat this experiment 100 times. Answer the questions from Scenario 1 for this scenario. Discuss the factors contributing to the observed changes.

Scenario 3: Many Prophets, Ten Games.

Load the file *scenario_three_and_four_prophets.pkl* which contains prophets with 'true-risk' that is uniformly distributed in the range $[0, 1]$
Evaluate the prophets on a train set of 10 games. Repeat the experiment 100 times, each time selecting the best prophet using the ERM algorithm. For each experiment:

1. Select a prophet based on the ERM algorithm Evaluate your selected prophet on the test set and compute its average error Calculate the approximation and estimation errors.

In your report

- Report the average error of the selected prophets over the experiments.
- In how many experiments did you choose the best prophet?
- In how many experiments did you choose a prophet that was not 1% worse than the best prophet?
- Calculate the approximation and mean estimation errors over the experiments.
- If the error rates were uniformly distributed between $[0, 0.5]$ instead of $[0, 1]$, what would happen to the approximation and estimation errors? Explain why.

Scenario 4: Many Prophets, Many Games.

Repeat the experiments from Scenario 3, but this time evaluate each prophet on 1000 randomly sampled games for each experiment. Repeat this experiment 100 times. each time selecting the best prophet using the ERM principle. For each experiment:

1. Select a prophet based on the ERM principle as mentioned above Evaluate your selected prophet on the test set and compute its average error Calculate the approximation and estimation errors.

In your report

- Report the average error of the selected prophets over the experiments.
- In how many experiments did you choose the best prophet?
- In how many experiments did you choose a prophet that was not 1% worse than the best prophet?
- Calculate the approximation and mean estimation errors over the experiments.
- How does the generalization gap of the selected prophet differ when evaluating on the train and on the test? Why?

Scenario 5: School of Prophets

Load the file *scenario_five_prophets.pkl* which contains prophets with 'true-risk' that is uniformly distributed in the range $[0, 0.2]$
In this scenario, we explore the impact of drawing prophets from a school of high-quality prophets, representing a strong hypothesis class. For varying values of $k \in \{2, 5, 10, 50\}$, draw k random prophets from the high quality prophets. Use the ERM principle to choose the best prophet by randomly sampling $m \in \{1, 10, 50, 1000\}$ games for each choice of k. Repeat this experiment 100 times.

1. Calculate the mean ERM approximation and estimation errors for every (k, m) combination.
2. Output them in a table, each cell containing the average error, the approximation error and the average estimation on the test set over all experiments with k on the rows and m on the columns.
3. Explain the observed patterns in the table and compare them to the scenarios in Scenarios 1-4.

Scenario 6: Bias-Complexity Tradeoff

In this scenario, we will build 2 competing hypothesis classes, each one representing a different *bias-complexity tradeoff*. We will then analyze the different classes, and choose between them [5].

The bias-complexity tradeoff in practice. Increasing hypothesis class size can tradeoff the approximation and estimation errors by increasing expressiveness but also increasing generalization error. We will simulate such a tradeoff in our experiment.
Load the file *scenario_six_prophets.pkl* which contains *hypothesis*1, inside are five prophets with true risk uniformly distributed in the range $[0.3, 0.6]$, and

[5] In practice, deciding between hypothesis classes should be done using a validation set. This is beyond the scope of this exercise (and will be explained later on in the course)

*hypothesis*2, which includes five-hundred prophets with true risk uniformly distributed in the range $[0.25, 0.6]$
Our training set size will be of 10 games that you shall sample, and our test set sizes will be 1000 games each. Choose the best prophet from each class by ERM on the training set and compute the error of the two chosen prophets using the test set. Repeat this experiment 100 times (changing only the train set each time).

1. What is the average error on the *test_set* for each class?
2. What is the approximation and the average estimation errors for each class?
3. Plot a histogram of the estimation-errors through the experiments and explain the the different bias-complexity tradeoffs that each hypothesis class represents. Make sure the comparison is meaningful by having the same range and number of bins in both histograms

PAC Learning Analysis

In the **PAC (Probably Approximately Correct)** learning framework, a **hypothesis class** $\mathcal{H}$ denotes the set of all possible predictors (or "prophets") that the learning algorithm can choose from. Each hypothesis $h \in \mathcal{H}$ represents one possible rule or model mapping inputs to outputs.

Consider a hypothesis class with $|\mathcal{H}| = 100$, desired accuracy $\epsilon = 0.05$, and confidence level $\delta = 0.01$.

1. Compute the minimal number of samples m required for PAC learnability under these parameters.
2. Analyze how m changes when the number of hypotheses (prophets) is doubled ($|\mathcal{H}| = 200$).
3. Given $|\mathcal{H}| = 100$ and $\delta = 0.01$, plot m as a function of ϵ in the range $\epsilon \in [0.01, 0.2]$. Provide a short interpretation of the graph.
4. Given $|\mathcal{H}| = 100$ and $\epsilon = 0.05$, plot m as a function of δ in the range $\delta \in [10^{-4}, 0.1]$ (logarithmic scale recommended). Provide a short interpretation of the graph.

5 Ethics

We will not tolerate any form of cheating or code sharing. You may only use the *numpy* library, the *pickle* library (to load your data), and *mathplotlib* for the required plots.

6 Submission Guidelines

You should submit your report, code and README for the exercise as $ex1\_\{YOUR\_ID\}.zip$ file. **Other formats (e.g. *.rar*) will not be accepted!**.

The README file should include your name, cse username and ID.

Reports should be in PDF format and be no longer than 6 pages in length. They should include any analysis and questions that were raised during the exercise. Please answer the questions in Sec. 4 in sequential order. **You should submit the filled-in skeleton, Report PDF and README files alone without any additional files.**

6.1 ChatGPT / Github Copilot

For this exercise specifically, we advise you to avoid ChatGPT / Github Copilot, as one of the purposes is to get familiar with **numpy**. Saying that, we do not prohibit using them. If you do choose to use them, write at the end of your report a paragraph detailing for which parts /functionalities you used them.

6.2 Submission

Please submit a zip file named $ex1\_\{YOURID\}.zip$ that includes the following:

1. A README file with your name, cse username and ID.
2. The *.py* files with your filled code.
3. A PDF report.