Machine Learning Methods
Exercise 1

January 11, 2024

1 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* your training set to select a prophet out of hypothesis classes. 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 the *numpy* library, for which a tutorial can be found here.

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. 7, please read them carefully.

You are provided with a skeleton code for this exercise, you must complete the necessary code in there alone. 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 awarded marks.

Note 2: NumPy is a library which is intended for vectorized operations (i.e. without *which* or *for* loops). Hence, you are only allowed to use *two* loop in each question i.e. looping over the experiments number and the set of prophets.

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.

2 Creating the Testing Environment

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

2.1 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 $\mathcal{Y}$.

1. Fill in the python function *create_data*. Each game will be described by the game id denoted as $x \in \{0, ..., n-1\}$. The label of each game (y) should be drawn randomly (with 50% chance for each label) from the set $\mathcal{Y} = \{0, 1\}$. The function should return the label array Y. **You are not allowed to use any loops in this function**.

2. Fill in the python function *compute_error* that evaluates the average error of a given predicted set of labels (*preds*), with the respect to the the real labels (also known as ground-truth, denoted *gt*). **You are not allowed to use any loops in this function**.

2.2 Creating the Prophets

Each prophet will be assigned a probability p. When called, it should return the correct answer (according to the created data) in probability p, and a wrong answer in probability *1-p*. We recommend doing so by drawing a random number *thresh* in the range $[0, 1]$ with a uniform distribution, then if $p > thresh$ return the correct answer, else return a wrong one. Notice the *predict* function should predict the results of multiple x queries together, in a vectorized manner.

Example: Say we have a prophet that predicts incorrectly with probability $1 - p = 0.3$, and that the game ended with the second team winning ($y = 0$). When the prophet is asked to predict the result of the game, it draws a random number $t \in [0, 1]$. If $0.7 > t$ the prophet returns the correct label, i.e. 0. But, if $0.7 \leq t$ then the prophet will return the wrong answer for that specific game: 1.

1. Implement the prophet class in the given *prophet.py* file.

2. Fill in the python function *sample_prophets*, which samples k prophets, each with a random probability in the range $[min\_p, max\_p]$.

3 Seeding

In our experiments we would like to choose the prophets and games randomly, but we would also like them to be reproducible. This can be done by random seeding. Random seeding forces the computer to draw the same set of numbers each time, but they are chosen in an unpredictable way (making them effectively random for us). For this exercise we already set the random seeding for you, do not change, move or remove it. This will make your results reproducible for us.

4 Data Splits

- The skeleton code prepares in advance 100 sets of 1000 games, which we denote as the training sets, for each experiment, along with a single set of 1000 games which is the test set. When asked to use less games than that (as in most questions), you should sample games from these sets using the *np.random.choice* function.

5 Questions

NOTE: In some of your experiments you might may get that several prophets win the same number of games, yet you have to choose between them by ERM. In these case simply choose the first one.

Scenario 1: Two prophets, One game.

You have two prophets, one with an 20% error and another 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 principle to choose the best one. Repeat this experiment 100 times.

1. Evaluate your selected prophet on the test set and compute its average error. Report the average error of the selected prophets over the experiments.
2. In how many experiments did you choose the best prophet?
3. Calculate the mean approximation and estimation errors.

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.

Randomly draw 500 prophets with error rates randomly distributed between $[0, 1]$. Evaluate the prophets on a train set of 10 games. Repeat the experiment 100 times, each time selecting the best prophet using the ERM principle.

1. Evaluate the selected prophet on the test set and compute its average error. Report the average error of the selected prophets over the experiments.
2. In how many experiments did you choose the best prophet?
3. In how many experiments did you choose a prophet that was not 1% worse than the best prophet?
4. Calculate the mean approximation and estimation errors.
5. 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.

1. Evaluate your selected prophet on the test set and compute its average error. Report the average error of the selected prophets over the experiments.
2. In how many experiments did you choose the best prophet?
3. In how many experiments did you choose a prophet that was not 1% worse than the best prophet?
4. Calculate the mean approximation and estimation errors.
5. How does the generalization error of the selected prophet differ when evaluating on the train and on the test? Why?

Scenario 5: School of Prophets

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 with error probabilities uniformly distributed between $[0, 0.2]$. 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. For this question, you are allowed to use four nested for loops over: (i) the different values of k (ii) the different values of m (iii) number of experiments (iv) current number of prophets - k.

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 estimation and average approximation error 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 [1].

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.

The first hypothesis class will include 5 prophets randomly sampled from the range $[0.3, 0.6]$. The second hypothesis class will include 500 prophets randomly sampled from the range $[0.25, 0.6]$. Our training set size will be of 10 games, 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 approximation and estimation errors for each class? Explain the different bias-complexity tradeoffs that each hypothesis class represents.

6 Ethics

We will not tolerate any form of cheating or code sharing. You may only use the *numpy* library, and if you wish to plot your results, you may use the *mathplotlib* library.

7 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 5 pages in length. They should include any analysis and questions that were raised during the exercise. Please answer the questions in Sec. 5 in sequential order. **You should submit the filled-in skeleton, Report PDF and README files alone without any additional files.**

[1] 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)

7.1 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.

7.2 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.