Stat4DS | Homework 01

Pierpaolo Brutti

Due Wednesday, November 29 (on Moodle)

General Instructions

I expect you to upload your solutions (only 1 per team) on Moodle as a **single running** `R Markdown` file (`.rmd`) + its `html` output, **named with your surnames**. Alternatively, a `zip`-file with all the material inside will be fine too.

You will give the commands to answer each question in its own code block, which will also produce plots that will be automatically embedded in the output file. Your responses must be supported by both textual explanations and code.

R Markdown Test

To be sure that everything is working fine, start `RStudio` and create an empty project called `HW1`. Now open a new `R Markdown` file (`File > New File > R Markdown...`); set the output to `HTML mode`, press `OK` and then click on `Knit HTML`. This should produce a web page with the knitting procedure executing the default code blocks. You can now start editing this file to produce your homework submission.

Please Notice

- For more info on `R Markdown`, check the support webpage that explains the main steps and ingredients: R Markdown from RStudio. For more info on how to write math formulas in LaTex: Wikibooks.
- Remember our **policy on collaboration**: *collaboration on homework assignments with fellow students is* ***encouraged****. However, such collaboration should be clearly acknowledged, by listing the names of the students with whom you have had* ***discussions*** *(no more) concerning your solution. You may* ***not****, however, share written work or code after discussing a problem with others. The solutions should be written by* ***you and your group*** *only.*

Exercise 1: Warm-up & Throwback

1. Setup: The importance of being earnest

There is a class of $n = 150$ students handling their first Stat-Homework under very strict **policy on collaboration** rules. By experience, `The Prof` perfectly knows there are k students that are just insolent liars, and $n - k$ clean, no-doped students. For once, you've to play me, `The Prof`, and your goal is to classify each person in the class as liar or honest.
How? Well, easy, by tossing a coin, what else?
Here's the protocol (it should ring a bell at least for some of you!):

- Honest students toss a fair coin and tell the truth, so there will be $n - k$ random variables where $H_j \sim \text{Ber}(p = 0.5)$
- The liars sometimes, well... lie! So there will be k random variables where $L_j \sim \text{Ber}(q)$ where $q \in (0, 1)$ is fixed and greater than 0.5 for all the k variables.
  Remark: students won't know the actual value of q but we do know, we are the *oracle*!

Now, what about you? What is `The Prof`'s goal? Here it is...

- You have to choose a number N of tosses, so each student in the class (honest or not) will toss their coin N times.
- After this N tosses, `The Prof` has to decide whether *that* student is a liar or not.

Clearly you make mistakes. Let's introduce some notation:

- $\alpha = \alpha(q) = \Pr(\texttt{False Positive}) = \Pr(\texttt{You claim the student is a liar} \,|\, \texttt{The student is honest})$;
- $\beta = \beta(q) = \Pr(\texttt{False Negative}) = \Pr(\texttt{You claim the student is honest} \,|\, \texttt{The student is a liar})$.

Remarks: clearly α and β depend on q (as suggested by our notation), and of course, when N grows, α and β decrease since it's easier and easier to separate the distribution of H_j from the distribution of L_j, but you cannot wait till ∞ and beyond!

⇝ Your job ⇜

A generic solution will be the number N chosen, and your **decision rule**, for example:

$$\{N = 50\} + \{\texttt{the student is a liar if among the 50 tosses he gets 40 heads}\}.$$

To this end you will have to:

1. Design a **score function** $f(N \,|\, \alpha^\star, \beta^\star)$ that, for every choice of N and two pre-selected target error levels $\alpha^\star$ and $\beta^\star$ you would like to achieve, outputs a numerical evaluation of the associated strategy: the higher the score, the better the strategy. Clearly justify your choice, possibly using examples or citing relevant references/sources.

   Numerically and graphically study the behavior of $f(N \,|\, \alpha^\star, \beta^\star)$: your strategy should optimize (or at least approximate the one that optimizes) the score function $f(N \,|\, \alpha^\star, \beta^\star)$. Extensively comment your findings.

2. Beside the $\alpha - \beta$ target above, assume now you're also working under a strict time constraint: knowing that each flip costs 3 seconds, you want to check at least half the class in no more that `T` minutes.
   Tweak the original score function to get a new one, say $g(N \,|\, \alpha^\star, \beta^\star, \texttt{T})$, to suitably handle this new scenario.

   As before, justify first and then study the behavior of your choice w.r.t. its various inputs. Finally, characterize/explore your new optimal strategies (still as a function of $\alpha^\star$, $\beta^\star$ and, of course, q) for $\texttt{T} = 15$ minutes. Extensively comment your findings making relevant comparisons with the original solutions.

Exercise 2: Faraway, So Close!

1. Background: Statistical Distances and Losses (more info)

Suppose that $\boldsymbol{X}$ is a random vector in $\mathbb{R}^m$ with distribution $\mathcal{P}_X$ and $\boldsymbol{Y}$ is another random vector in $\mathbb{R}^m$ with distribution $\mathcal{P}_Y$ then, for $p \geqslant 1$, the p–Wasserstein distance based on the ground distance d is defined as

$$W_{d,p}(\mathcal{P}_X, \mathcal{P}_Y) \equiv W_{d,p}(\boldsymbol{X}, \boldsymbol{Y}) = \left(\inf_{\mathcal{J}} \int_{\mathbb{R}^m \times \mathbb{R}^m} d(\mathbf{x}, \mathbf{y})^p \, d\mathcal{J}(\mathbf{x}, \mathbf{y}) \right)^{1/p},$$

where the infimum is over all joint distributions $\mathcal{J}$ having $\mathcal{P}_X$ and $\mathcal{P}_Y$ as given marginals... sounds complicated, does it? Anyway, these optimal transport techniques are extremely useful and widespread nowdays with tons of "*... truly marvelous Machine Learning applications, which this homework is too narrow to contain...*".

Nevertheless, when $m = 1$, that is for univariate distributions, things simplify quite a bit and it can be shown that, by picking the $\mathtt{L}_1$ metric (i.e. the *absolute distance*) as ground distance, we obtain

$$\mathrm{W}_{\mathtt{L}_1, p}(\mathcal{P}_X, \mathcal{P}_Y) = \left(\int_0^1 \left| \mathrm{F}_X^{-1}(z) - \mathrm{F}_Y^{-1}(z) \right|^p dz \right)^{1/p},$$

where $\mathrm{F}_X(\cdot)$ and $\mathrm{F}_Y(\cdot)$ are the cumulative distribution functions of $\mathcal{P}_X$ and $\mathcal{P}_Y$, respectively and, consequently, $\mathrm{F}_X^{-1}(\cdot)$ and $\mathrm{F}_Y^{-1}(\cdot)$ are essentially their associated **quantile functions**.

2. The `Perfect Information` Learning Scenario

1. Think about $\mathrm{F}_X(\cdot)$ as the **true** population model and assume that $X \sim \mathrm{Beta}(\alpha, \beta)$ for some value of (α, β) that you will choose (read about this flexible parametric family with bounded support at page 54-56 of our Gallery).
   Let $f(\cdot)$ denote the associated **true** density and $F^{-1}(\cdot)$ its quantile function.

2. Think about $\mathrm{F}_Y(\cdot) = \mathrm{F}_Y(\cdot\,; \boldsymbol{\theta})$ as an approximation to $\mathrm{F}_X(\cdot)$ that you need to *tune/optimize* with respect to the parameter vector $\boldsymbol{\theta}$ in order to achieve some specific goal.
   For this exercise we will consider a **Kernel Density Estimator** $\widehat{f}_h(\cdot)$ with bandwidth $h > 0$ as the only tunable parameter, and a `boxcar/rectangular` or an `Epanechnikov` kernel (read our notes if you haven't already in early October).

   The `R` function `density()` implements this algorithm, but other solutions can also be used.

3. Assume that we are working under **perfect information**, meaning that we can access/query the **true** model.

3. The Exercise: Comment every line of code and result you get!

⇝ Your job ⇜

1. Pick an implementation of $\widehat{f}_h(\cdot)$ and code (on your own/by hand) its corresponding quantile function, say $\widehat{F}_h^{-1}(\cdot)$, in `R`.
2. Pick two diverse (α, β) pairs, and let $\epsilon > 0$ be an approximation level you would like to achieve in terms of $p = 1$ Wasserstein distance, meaning that you're looking for the **largest** bandwidth h (i.e. the **coarsest** approximation) such that

$$\mathrm{W}_{\mathrm{L}_1,1}\big(f, \widehat{f}_h\big) = \left(\int_0^1 \left|\mathrm{F}^{-1}(z) - \widehat{\mathrm{F}}_h^{-1}(z)\right| dz\right) \leqslant \epsilon$$

   Use any (**reasonable**) technique you like, to study how h must vary with ϵ. Properly comment, compare and visualize the results.
