IFT 6390 Fundamentals of Machine Learning
Ioannis Mitliagkas

Homework 0

- This homework must be done and submitted to Gradescope individually (you will not be collaborating with other students)
- The theoretical part should be submitted as a pdf file. It should preferably be done in LaTeX by copying this project (`Menu -> Copy Project`) and working off of this exact file. Still all **legible** solutions submitted as a pdf will be accepted. We reserve the right to penalize solutions that are very hard to read, even if they are correct.
- The practical part should be coded in python (you can use the numpy and matplotlib libraries) and all code will be submitted as a python file to Gradescope. To enable automated code grading you should work off of the template file given in this homework folder. Do not modify the name of the file or any of the function signatures of the template file or the code grading will not work for you. You may, of course, add new functions and any regular python imports
- The practical report (graphing, charts, or other report parts) should be submitted in a pdf to Gradescope. For the report it is recommended to use a Jupyter notebook, writing math with MathJax and export as pdf. You may alternatively write your report in LaTeX; LyX; Word. In any case, you should export your report to a pdf file that you will submit. You are of course encouraged to draw inspiration from what was done in lab sessions
- NOTE: Many of the math questions can be answered by using online resources and automatic math solvers. This is perfectly fine if it helps you practice, but keep in mind that during the in-class exams you will not have access to those resources and automatic solvers

1 Theoretical Part [5 points]

This part of the homework will not be graded for correctness. You will still need to provide an answer and properly annotate it on Gradescope. You get 1 point for each answer that you properly annotate, even if the answer itself is incorrect. We will provide you with solutions which you can use to check your answers. The goal is for you to i) practice the process of annotating answers to questions on Gradescope and, ii) get an idea of what kind of basic math we will need during the class. If you are having trouble answering these questions you might be missing some important prerequisite knowledge for this class.

1. [1 points] Let X be a random variable representing the outcome of a single roll of a 6-sided dice. Show the steps for the calculation of i) the expectation of X and ii) the variance of X.

2. [1 points] Let $u, v \in \mathbb{R}^d$ be two vectors and let $A \in \mathbb{R}^{n \times d}$ be a matrix. Give the formulas for the euclidean norm of u, for the euclidean inner product (aka dot product) between u and v, and for the matrix-vector product Au.

3. [1 points] Consider the two algorithms below. What do they compute and which algorithm is faster?
   Observez les deux algorithms ci-dessous. Que calculent-ils et lequel est le plus rapide ?

   ```
   ALGO1(n)
   result = 0
   for $i = 1 \dots n$
     result = result + $i$
   return result
   ```

   ```
   ALGO2(n)
   return $(n+1) * n/2$
   ```

4. [1 points] Give the step-by-step derivation of the following derivatives:

$$\text{i)} \quad \frac{df}{dx} = ?, \quad \text{where } f(x, \beta) = x^2 \exp(-\beta x)$$

$$\text{ii)} \quad \frac{df}{d\beta} = ?, \quad \text{where } f(x, \beta) = x \exp(-\beta x)$$

$$\text{iii)} \quad \frac{df}{dx} = ?, \quad \text{where } f(x) = \sin(\exp(x^2))$$

5. [1 points] Let $X \sim N(\mu, 1)$, that is the random variable X is distributed according to a Gaussian with mean μ and standard deviation 1. Show how you can calculate the second moment of X, given by $\mathbb{E}[X^2]$.

1. X is a multinoulli random variable. Hence it is a vector of length 6 where by convention we say the ith entry is 1 if the dice roles i and 0 elsewhere. We assume the dice is fair each entry has probability 1/6. The expected value is thus a vector with 1/6 in each entry.

$$\begin{aligned} P(X_i = 1) &= 1/6 && (1) \\ EX_i &= 1 * 1/6 + 0 * 5/6 = 1/6 && (2) \end{aligned}$$

Since X is a vector its variance is actually a 6 by 6 covariance matrix:

$$Cov[X_i, X_j] = E[X_i X_j] - E[X_i]E[X_j] \tag{3}$$

If $i = j$ we have

$$\begin{aligned} Cov[X_i, X_j] = E[X_i^2] - E[X_i]^2 &= p_i - p_i^2 && (4) \\ &= 1/16 - 1/16^2 && (5) \\ &= 15/256 && (6) \end{aligned}$$

Where we use $X_i^2 = X_i$ since X_i is this random variable can only take values 1 and 0.

Now if $i \neq j$ we have that the product X_i and X_j is zero since only one of the entries of X can be non zero at a time, hence:

$$\begin{aligned} Cov[X_i, X_j] = E[X_i X_j] - E[X_i]E[X_j] &= -E[X_i]E[X_j] && (7) \\ &= -p_i^2 && (8) \\ &= -1/256 && (9) \end{aligned}$$

2.

1. euclidean norm: $\|u\| = \sqrt{\sum_{i=1}^{d} u_i^2}$
2. inner product: $\langle u, v \rangle = \sum_{i=1}^{d} u_i v_i$
3. matrix product: $(Au)_k = \langle A_{k,:}, u \rangle$ where $(Au)_k$ denotes the kth entry in the resulting vector.

3.

Both algorithms compute the sum of the first n natural numbers. The second algorithm is faster because it computes this sum with an expression that contains a number of arithmetic operations constant with respect to the size of the n, i.e. 3 arithmetic operation. The first algorithm on the other hand computes the result in a number of arithmetic operations linear in the size of n. .

4.

1. $\frac{df}{dx} = -\beta x^2 \exp(-\beta x)$
2. $\frac{df}{d\beta} = -x^2 \exp(-\beta x)$
3. $\frac{df}{x} = \cos(\exp(x^2)) \sin(2x \exp(x^2))$

5.

$E[X^2]$ can be computed with the following formula:

$$\begin{aligned} Var(X) &= E[(X - E[X])^2] && (10) \\ &= E[X^2 - 2XE[X] + E[X^2]] && (11) \\ &= E[X^2] - 2E[X]E[X] + E[X^2] && (12) \\ &= E[X^2] - E[X]^2 && (13) \end{aligned}$$

Hence in our case $E[X^2] = Var(X) + E[X]^2 = 1 + \mu^2$

2 Practical Part [7 points]

You should work off of the template file `solution.py` in the project and fill in the basic numpy functions using numpy and python methods

1. [1 points] Create a numpy array from a python list
2. [1 points] Create a numpy array of length 1 from a python number
3. [1 points] Sum two arrays elementwise
4. [1 points] Sum an array and a number
5. [1 points] Mutliple two arrays elementwise
6. [1 points] Dot product of two arrays
7. [1 points] Dot product of an array (1D) and a matrix (2D array)