# HOMEWORK 2 — V2

CSC2515 Fall 2021

- **V1 → V2:** Updated 3 (b) to include the regularization parameter $\lambda$.
- **Deadline:** Monday, October 25, 2021 at **16:59**.
- **Submission:** You need to submit two files through MarkUs. One is a PDF file including all your answers and plots. The other is a source file (Python script or Jupyter Notebook) that reproduces your answers. You can produce the file however you like (e.g. LaTeX, Microsoft Word, etc) as long as it is readable. Points will be deducted if we have a hard time reading your solutions or understanding the structure of your code. If the code does not run, you may lose most/all of your points for that question.
- **Late Submission:** 10% of the marks will be deducted for each day late, up to a maximum of 3 days. After that, no submissions will be accepted.
- **Collaboration:** You can discuss the assignment with up to two other students (group of three). You can work on the code together. But each of you need to **write your homework report individually**. You must mention the name of your collaborators clearly in the report and the source code.

**1. Bias and Variance Decomposition for the $\ell_2$-regularized Mean Estimator – 35pts.** This exercise helps you become more comfortable with the bias and variance calculations and decomposition. It focuses on the simplified setting of the mean estimator.

Consider a r.v. $Y$ with an unknown distribution $p$. This random variable has an (unknown) mean $\mu = \mathbb{E}[Y]$ and variance $\sigma^2 = \mathrm{Var}[Y] = \mathbb{E}\big[(Y-\mu)^2\big]$. Consider a dataset $\mathcal{D} = \{Y_1, \ldots, Y_n\}$ with independently sampled $Y_i \sim p$.

(a) **[5pt]** Show that the sample average estimator $h_{\text{avg}} = \frac{1}{n}\sum_{i=1}^n Y_i$ is the solution of the following optimization problem:

$$h_{\text{avg}}(\mathcal{D}) \leftarrow \operatorname*{argmin}_{m\in\mathbb{R}} \frac{1}{n}\sum_{i=1}^n |Y_i - m|^2.$$

Recall that we have the following bias-variance decomposition for the mean estimator $h(\mathcal{D})$:

$$\mathbb{E}_{\mathcal{D}}\Big[|h(\mathcal{D}) - \mu|^2\Big] = \underbrace{|\mathbb{E}_{\mathcal{D}}[h(\mathcal{D})] - \mu|^2}_{\text{bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}\Big[|h(\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[h(\mathcal{D})]|^2\Big]}_{\text{variance}}.$$

(b) **[5pt]** Compute the bias and variance of $h_{\text{avg}}(\mathcal{D})$.

(c) **[5pt]** Consider the $\ell_2$-regularized mean estimator $h_\lambda(\mathcal{D})$ defined for any $\lambda \geq 0$.

$$h_\lambda(\mathcal{D}) \leftarrow \operatorname*{argmin}_{m\in\mathbb{R}} \frac{1}{n}\sum_{i=1}^n |Y_i - m|^2 + \lambda|m|^2.$$

Notice that this is similar to the ridge regression, but only for a single random variable. Provide an explicit formula for this estimator in terms of the sample average and $\lambda$.

(d) **[10pt]** Compute the bias and variance of this regularized estimator.

(e) **[5pt]** Visualize $\mathbb{E}_{\mathcal{D}}\left[|h_\lambda(\mathcal{D}) - \mu|^2\right]$, the bias, and the variance terms for a range of $\lambda$. As a starting point, you can choose $\mu = 1$, $\sigma^2 = 9$, and $n = 10$.

(f) **[5pt]** Explain the visualization from the previous part, in terms of the effect of bias and variance on the expected squared error.

In this rest of this assignment, we will be working with the **Boston Houses dataset**. This dataset contains 506 entries. Each entry consists of a house price and 13 features for houses within the Boston area. We suggest working in Python and using the **scikit-learn package** to load the data.

**2. Learning the basics of regression in Python – 30 pts.** This question will take you step-by-step through performing basic linear regression on the Boston Houses dataset. Using the starter code provided, you need to submit modular code for this question. Your code needs to be functional once downloaded. Non-functional code will result in losing all marks for this question. If your code is non-modular or otherwise difficult to understand, you risk losing a significant portion of the marks, even if the code is functional.

**Environment setup:** For this question you are strongly encouraged to use the following Python packages:

- sklearn
- matplotlib
- numpy

You will submit a complete regression analysis for the Boston Housing data. To do that, here are the necessary steps:

(a) Load the Boston housing data from the sklearn datasets module

(b) Describe and summarize the data in terms of number of data points, dimensions, target, etc

(c) Visualization: present a single grid containing plots for each feature against the target. Choose the appropriate axis for dependent vs. independent variables.
**Hint**: use `pyplot.tight_layout` function to make your grid readable

(d) Write code to perform linear regression to predict the targets using the training data. Remember to add a bias term to your model.

(e) Tabulate each feature along with its associated weight and present them in a table. Explain what the sign of the weight means in the third column ('INDUS') of this table. Does the sign match what you expected? Why?

(f) Test the fitted model on your test set and calculate the Mean Square Error of the result.

(g) Suggest and calculate two more error measurement metrics; justify your choice.

(h) Feature Selection: Based on your results, what are the most significant features that best predict the price? Justify your answer.

**3. Locally weighted regression – 35 pts.** Locally weighted linear regression is a non-parametric algorithm, that is, the model does not learn a fixed set of parameters as is done in ordinary linear regression. Rather, parameters are computed individually for each data point $x$. By following the steps of this exercise, you derive the algorithm and empirically evaluate it.

(a) **[10pt]** Given dataset $\{(\mathbf{x}^{(1)}, y^{(1)}), .., (\mathbf{x}^{(N)}, y^{(N)})\}$ and positive weights $a^{(1)}, ..., a^{(N)}$ show that the solution to the *weighted* least square problem

$$\mathbf{w}^* = \arg\min \frac{1}{2} \sum_{i=1}^{N} a^{(i)} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 \tag{3.1}$$

is given by the formula

$$\mathbf{w}^* = \left(\mathbf{X}^T \mathbf{A} \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{A} \mathbf{y} \tag{3.2}$$

where $\mathbf{X}$ is the design matrix (defined in class) and $\mathbf{A}$ is a diagonal matrix where $\mathbf{A}_{ii} = a^{(i)}$

(b) **[10pt]** Locally weighted least squares regression algorithm combines ideas from k-NN and ordinary linear regression. For each new test example $\mathbf{x}$ it computes distance-based weights for each training example

$$a^{(i)} = \frac{\exp(-||\mathbf{x} - \mathbf{x}^{(i)}||^2 / 2\tau^2)}{\sum_j \exp(-||\mathbf{x} - \mathbf{x}^{(j)}||^2 / 2\tau^2)},$$

and then computes

$$\mathbf{w}^* = \arg\min \frac{1}{2} \sum_{i=1}^{N} a^{(i)} (y^{(i)} - \mathbf{w}^T \mathbf{x}^{(i)})^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2,$$

and predicts

$$\hat{y} = \mathbf{x}^T \mathbf{w}^*.$$

Complete the implementation of locally reweighted least squares by providing the missing parts for q3.py.
Important implementation details:

- Do not invert any matrices; use a linear solver (numpy.linalg.solve is one example).
- Notice that $\frac{\exp(A_i)}{\sum_j \exp(A_j)} = \frac{\exp(A_i - B)}{\sum_j \exp(A_j - B)}$ but if we use $B = \max_j A_j$ it is much more numerically stable as $\frac{\exp(A_i)}{\sum_j \exp(A_j)}$ overflows/underflows easily. *This is handled automatically in the scipy package with the scipy.misc.logsumexp function.*

(c) **[5pt]** Use k-fold cross-validation to compute the average loss for different values of $\tau$ in the range [10,1000] when performing regression on the Boston Houses dataset. Plot these loss values for each choice of $\tau$ (For this question fix $\lambda = 10^{-5}$).

(d) **[5pt]** How does this algorithm behave when $\tau \to \infty$? When $\tau \to 0$?

(e) **[5pt]** Mention two advantages and two disadvantages of the locally weighted linear regression compared to ordinary linear regression.