MACHINE LEARNING ASSIGNMENT

CH-315, Fall 2025

1. Cost Function (30 points)

Let's focus on linear regression of the form

$$y \approx f(x) = Xw_1 + w_0$$

- What are the rows of X? (2.5 point)

- What are the columns of X? (2.5 point)

If we rewrite the equation as below:

$$y \approx \tilde{X}W$$

- How does $\tilde{X}$ look like in this case (i.e., how does the shape of the matrix change compare to X)? (5 point)

For machine learning, we need a cost function. These are the functions we try to minimize (or sometimes maximize). Two common choices are the mean-squared error (MSE), and the mean-absolute error (MAE).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\big(y_i - f(x_i)\big)^2$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - f(x_i)|$$

- In the Jupyter notebook, write a Python function that computes these two cost functions given an error term $\varepsilon\ = y - \tilde{X}w$ (5 points, 2.5 each).[1]

- What is the shape of these cost functions as a function of the error (i.e., either sketch the shape of the functions as a function of the error or use matplotlib to plot the cost function as a function of the error—you can focus on the one-dimensional case, 5 point).

- Are both loss functions differentiable for all ε?[2] (2.5 point) What implications does this have for gradient-based optimization like gradient descent? (2.5 point)

- Which loss function is more sensitive to outliers (2.5 point) and why (2.5 point)?[3]

[1] Hint: For the implementation, the functions `np.mean` and `np.abs` will be of use. They can be imported using `import numpy as np`. Then you can calculate the mean of an array `a = np.array([1,1,2])` as `np.mean(a)`.

[2] Hint: For analyzing the differentiability, check how the function looks like for zero error

[3] Hint: For understanding the influence of outliers, check which function will give larger function values for larger inputs (=errors)

2. Regularization (70 points)

Assume that the columns of X are linearly independent. As a refresher of linear algebra, recall when the linear system $Xw = y$ has:

- a unique solution (2 point)
- no solution (2 points)
- an infinite number of solutions (2 points)

it is easiest to express these conditions in terms of rank (X) and of the augmented matrix rank $(X|y)$ (i.e., what happens if you have more columns than rows and vice versa).

- Give a geometrical interpretation of the matrix Rank. (4 points, 10 bonus points if you show it in animation using Manim package)
- In general, why can't we solve the linear system using $y = \tilde{X}^{-1}W$? (2 points)

In the least-square problem we aim to minimize

$$\left\|y - \tilde{X}w\right\|_2^2$$

Where the symbol $\|\vec{v}\|_2$is the 2-norm (Euclidean norm) of the vector $\vec{v}$ which is equal to

$$\sqrt{v_1^2 + v_2^2 + \ldots + v_n^2}$$

which in $\|\vec{v}\|_2^2$we just square the Euclidean norm: $v_1^2 + v_2^2 + \cdots + v_n^2$

We can rewrite the least-square problem as

$$\left(y - \tilde{X}w\right)^T \left(y - \tilde{X}w\right)$$

• Differentiate above formula step by step and show what will we have? (5 points)

• if we want $\left\|y - \tilde{X}w\right\|_2^2$ to be minimum, what should the derivative be equal to? (2 points)

• What is the Hat matrix and what does its diagonal values correspond to? (5 points)

• What is William's plot and how does it help in outlier detection? (5 points)

• What happens if some columns are linearly dependent?[4] (2 point) What is the connection to feature selection?[5] (2 point)

One reason to introduce a regularization term $\lambda\|w\|_2^2$to the cost function is that it makes the part of Hat matrix where we want to take the inverse from always reversible.

• What will be the new cost function after adding the regularization term? (3 points)

•What will be the new w when we differentiate the new cost function and set it to zero. (7 points)

• Prove that the part of Hat matrix where we want to take the inverse from is always reversible after we introduce the regularization term. (5 points)

Regularization does not only help linearly dependent features to not cause problems, it helps prevent overfitting. To understand the connection to the reduction of overfitting consider the function $f(x) = ax^2$

[4] Hint: One way to think about it is what rank the X has compared to $X|y$

[5] Hint: on real datasets the columns might not be linearly independent, what does this then imply for the solution of $y \approx \tilde{X}W$.

- What is the shape of the parabola as a function of a? (You can use matplotlib or you can just sketch it by hand, 2 point).

In more complicated models, like polynomial regression, we approximate the function as a linear combination of such terms

$$f(x) = \sum_{i}^{n} a_i x_i^i$$

Let's consider approximating the function $cos(1.5\pi x)$, using noisy samples of it (see code in the notebook)

- Plot the approximation to the function for different order polynomials (N ∈ {1, 2, 16}) and with different regularization strength ($\lambda \in \{0, 10^{-3}, 10^{-2}, 1\}$). What do you observe (explains in terms of the smoothness of the function, you can look up the term Occam's razor[6]). (10 points: plotting, description, explanation)
- What do you observe if you change the number of samples from the function? (5 point)
- Why do we need a test set in machine learning? (3 point)
- If we need to optimize hyperparameters, do we use the test set to select the best hyperparameters?[7] (2 point)

[6] For example, you can have a look at Yann LeCun's lecture notes: https://cs.nyu.edu/~yann/2005f-G22-2565-001/diglib/lecture03-regularization.pdf

[7] Hint: Remember that we want to avoid data leakage. That is, our model should not see any test data during training or any other optimization.