CS 461: Machine Learning Principles
Fall 2021

Problem Set #1 (80 Points)

Out: Thursday, September 9

Due: Thursday, September 23, 8:00pm

Instructions

How and what to submit? Please submit your solutions electronically via Canvas. Please submit two files:

1. A PDF file with the written component of your solution including derivations, explanations, etc. You should create this PDF in LaTeX. Please name this document ⟨`firstname-lastname`⟩`-sol1.pdf`.

2. The empirical component of the solution (Python code and the documentation of the experiments you are asked to run, including figures) in a Jupyter notebook file. Rename the notebook ⟨`firstname-lastname`⟩`-sol1.ipynb`.

Late submissions: there will be a penalty of 20 points for any solution submitted within 24 hours past the deadline. No submissions will be accepted past then.

What is the required level of detail? See below.

(a) When asked to **derive** something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations.

(b) When asked to **plot** something, please include in the `ipynb` file the figure as well as the code used to plot it. If multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers) and references in a legend or in a caption.

(c) When asked to **provide a brief explanation or description**, try to make your answers concise, but do not omit anything you believe is important. If there is a mathematical answer, provide it precisely (and accompany by succinct wording, if appropriate).

(d) When **submitting code (in Jupyter notebook)**, please make sure it's reasonably well-documented, runs, and produces all the requested results. If **discussion** is required/warranted, you should include it directly in the notebook (using the markdown cell).

Collaboration policy: collaboration is allowed and encouraged, as long as you (1) write your own solution entirely on your own, and (2) specify names of student(s) you collaborated with in your writeup.

1 Linear Regression

This is the mathematical part.

1.1 Least Squares Estimation

Let $(x^{(1)}, y^{(1)}) \dots (x^{(N)}, y^{(N)}) \in \mathbb{R}^d \times \mathbb{R}$ denote a training dataset for regression. We fit a linear model $w \in \mathbb{R}^d$ on this dataset using the least-squares criterion, yielding the optimal parameter $w^* \in \mathbb{R}^d$. We will assume that the first input dimension is always 1 (i.e., $x_1^{(i)} = 1$ for all $i = 1 \dots N$) so that the linear model has a bias parameter: $w^\top x = w_1 + w_2 x_2 + \cdots w_d x_d$. We will also assume that the data matrix $X \in \mathbb{R}^{N \times d}$ (i.e., the i-th row of X is $(x^{(i)})^\top$) has linearly independent columns. In the next two problems we will consider the effect that **transforming the training data** has on the optimal regression model.

Problem 1 [10 points]
Suppose we create a new dataset by transforming the labels as $\tilde{y}^{(i)} = a y^{(i)} + b$ for some constants $a, b \in \mathbb{R}$. Let $\tilde{w} \in \mathbb{R}^d$ denote the least-square estimate on this new dataset $(x^{(1)}, \tilde{y}^{(1)}) \dots (x^{(N)}, \tilde{y}^{(N)}) \in \mathbb{R}^d \times \mathbb{R}$ (note that the inputs are unchanged). Can we obtain $\tilde{w}$ directly from w^*, without training on the new dataset? In other words, can we define a mapping $g : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^d$ such that $\tilde{w} = g(w^*, a, b)$? If yes, how precisely? If not, why not?

End of problem 1

Problem 2 [10 points]
Suppose we create a new dataset by transforming the inputs as $\bar{x}_j^{(i)} = c_j x_j^{(i)}$ for some nonzero constants $c_1 \dots c_d \in \mathbb{R}$. Let $\bar{w} \in \mathbb{R}^d$ denote the least-square estimate on this new dataset $(\bar{x}^{(1)}, y^{(1)}) \dots (\bar{x}^{(N)}, y^{(N)}) \in \mathbb{R}^d \times \mathbb{R}$ (note that the labels are unchanged). Can we obtain $\bar{w}$ directly from w^*, without training on the new dataset? In other words, can we define a mapping $h : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ such that $\bar{w} = h(w^*, (c_1 \dots c_d))$? If yes, how precisely? If not, why not?

End of problem 2

1.2 Maximum Likelihood Estimation

Recall the Gaussian noise model for regression: $y = w_{\text{true}}^\top x + \epsilon$ where $\epsilon \in \mathbb{R}$ is a zero-mean Gaussian random noise. Consequently, $y \in \mathbb{R}$ also follows a Gaussian distribution. In the class, we assume that the noises are iid (independently and identically distributed) so that for any input $x \in \mathbb{R}^d$, the corresponding label is independently generated by $y = w_{\text{true}}^\top x + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is identically distributed for all inputs. In this problem, we will consider a slightly different model in which the noises are independent but *not* identically distributed.

Problem 3 [15 points]
Let $(x^{(1)}, y^{(1)}) \dots (x^{(N)}, y^{(N)}) \in \mathbb{R}^d \times \mathbb{R}$ denote samples independetly generated as $y^{(i)} =$

$w_{\text{true}}^\top x^{(i)} + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ is a *sample-specific* Gaussian noise. Assume that we know the true variances $\sigma_1^2 \ldots \sigma_N^2 \in \mathbb{R}$ and that they are nonzero. Can we compute a closed-form solution for the maximum-likelihood estimate $w^* \in \mathbb{R}^d$ on this dataset? If yes, describe the procedure precisely. If not, explain why not. Note: To receive points, your reasoning should be specific and detailed. A correct solution with vague reasoning steps will not receive points. Minimal requirements include

- *Likelihood*: Show how to derive the data likelihood, under the model assumption.
- *Optimization*: Show how to optimize the likelihood with respect to the model parameter. Do not just write a single step like "$\arg\max \cdots$".
- *Closed-form solution*: Show how to derive an exact, closed-form solution if it exists. In particular, this excludes numerical optimization procedures like gradient descent.

End of problem 3

2 Loss Functions

This is the coding part. You will need to use **Python** as the programming language, and run program on **Jupyter Notebook** (an interactive environment). To avoid the hassle of properly setting up your local Python environment, we will use Google Colab. Upload the attached notebook file to your Google Drive. Then right click, open with Google Colaboratory, and you should be good to go.

Code and explain. The Jupyter notebook provided with this assignment contains most of the code you need to conduct experiments in this section. You need to (1) fill in some missing pieces of the code (search for "TODO") and (2) explain in the notebook (markdown cell).

Task overview. In this section we will focus on hands-on exploration of the interaction between loss functions used in regression and the models produced. We will work with the Boston Housing dataset in which the task is to predict median home prices in various areas of the Boston suburbs from a variety of features.

Loss. We are going to assume that in our task, it is much worse to overestimate the price of a house than to underestimate it (e.g., to avoid making an offer that's too high). Technically, we will express it as an **asymmetric squared loss**:

$$l_\alpha(\hat{y}, y) = \begin{cases} \alpha(\hat{y} - y)^2 & \text{if } \hat{y} < y \\ (\hat{y} - y)^2 & \text{otherwise} \end{cases}$$

where $\alpha < 1$ specifies how much more we worry about the model prediction $\hat{y}$ overestimating rather than underestimating the house price y. For example, if $\alpha = 0.1$, then overestimating is 10 times worse than underestimating by the same amount.

Model. We will consider fitting polynomial regression models to predict house prices y from observations/measures x.

Problem 4 [15 points]
Complete the missing code in the gradient descent function. Experimenting with models and optimization parameters, fit polynomial models of degrees 1 (linear), 2 (quadratic), and some others (feel free to explore) to the training set. Use the validation set to select one of the models; explain your selection process.

End of problem 4

Advice: Use the code for the closed form solution provided earlier in the notebook to debug your implementation of gradient descent. You should expect very similar (perhaps not quite identical) results when fitting, say, a linear model to the training data using the closed form solution and with gradient descent.

Now we will move to the asymmetric loss. The optimization problem involving the asymmetric loss is convex, but unfortunately does not have a closed-form solution (mainly because the decision to weigh by α or not is itself a function of the model parameter). So the gradient descent code we have will come in handy.

Problem 5 [15 points]
Complete the code for the function computing the asymmetric loss function and its gradient. Using the asymmetric loss with $\alpha = 0.05$, fit polynomial models (of degrees 1, 2, and any others you wish to try) to the training set. Similarly to the symmetric loss case, select one model using the validation set and explain your selection process.

End of problem 5

Problem 6 [15 points]
Finally, evaluate your two chosen models (one trained with symmetric loss and the other trained with asymmetric loss) on the test set. For each model compute: the mean symmetric loss and the mean asymmetric loss (using squared roots of both for comparison is fine). Based on these results, discuss the relative merits of the two models for the data and task at hand. If you had to choose one, given the description of the prediction problem and the assumptions above, which one would you choose, and why?

End of problem 6