New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 6923: Written Homework 1.

Due Friday, February 17th, 2023, 11:59pm.

Discussion with other students is allowed for this problem set, but solutions must be written-up individually.

For just this first problem set, 10% extra credit will be given if solutions are typewritten (using LaTeX, Markdown, or some other mathematical formatting program).

Problem 1: Practice Minimizing a Loss Function (10pts)

Consider a linear model of the form:

$$f_\beta(x) = \beta x,$$

which is the same as the linear model we saw in class, but with the intercept forced to zero. Such models are used when we want to force the predicted value $f_\beta(x) = 0$ when $x = 0$. For example, if we are modeling $y =$ output power of a motor vs. $x =$ the input power, we would expect $x = 0 \Rightarrow y = 0$.

(a) Given data $(x_1, y_1), \ldots, (x_n, y_n)$, write the equation for a loss function which measures prediction accuracy using the sum-of-squared distances between the predicted values and target values.

(b) Derive an expression for the β that minimizes this loss function. Do you get the same expression that we got for β_1 in the full linear model?

Problem 2: Machine Learning Does Averages (15pts)

Suppose we have data $y_1, \ldots, y_n \in \mathbb{R}$ and we want to choose a single value $m \in \mathbb{R}$ which is "most representative" of our dataset. This is sometimes called the "central tendency" problem in statistics. A machine learning approach to this problem would measure how representative m is of the data using a loss function. As you will see, different choices of loss function lead to different measures of central tendancy you have probably seen in the past!

(a) Consider the loss function $L(m) = \sum_{i=1}^n (y_i - m)^2$. Show that $L(m)$ is minimized by setting $m = \bar{y}$, where $\bar{y} = \frac{1}{n}\sum_{i=1}^n y_i$ is the ***mean*** of our data.

(b) Consider the loss function $L(m) = \max_i |y_i - m|$. What value of m minimizes this loss? **Hint:** Using derivatives will not help here – try just thinking about the minimization problem directly.

(c) Consider the loss function $L(m) = \sum_{i=1}^n |y_i - m|$. Prove that $L(m)$ is minimized by setting m to the ***median*** of the data. **Hint:** This question is harder than the previous two and takes some creativity! Again derivatives might not be helpful.

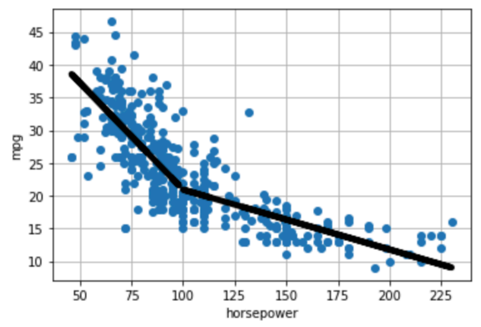
Problem 3: Piecewise Linear Regression via Feature Transformations (15pts)

Your goal is to fit a *piecewise* linear model to a single variate dataset of the form $(x_1, y_1), \ldots, (x_n, y_n)$ where all values are scalars. We will only use two pieces. In other words, for some known value λ,

$$f(x_i) = \begin{cases} a_1 + s_1 x_i & \text{for } x_i < \lambda \\ a_2 + s_2 x_i & \text{for } x_i \geq \lambda \end{cases}$$

with the additional **constraint** that $a_1 + s_1\lambda = a_2 + s_2\lambda$. This constraint ensures that our two linear models actually "meet" at $x = \lambda$, which means we get a continuous prediction function.

For example, when $\lambda = 100$, a piecewise linear fit for our MPG data might look like:



(a) Show that this model is equivalent to the following **unconstrained** model:

$$f(x_i) = \begin{cases} a_1 + s_1 x_i & \text{for } x_i < \lambda \\ a_1 + s_1\lambda - s_2\lambda + s_2 x_i & \text{for } x_i \geq \lambda \end{cases}$$

(b) Show how to fit an optimal f under the squared loss using an algorithm for multiple linear regression. In particular, your approach should:

- Transform the input data to form a data matrix $\mathbf{X}$ with multiple columns.
- Use a multiple regression algorithm to find the $\boldsymbol{\beta}$ which minimizes $\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$.
- Extract from the optimal $\boldsymbol{\beta}$ optimal values for a_1, s_1, s_2.

You need to describe 1) a correct data transformation and 2) a correct mapping from $\boldsymbol{\beta}$ to a_1, s_1, s_2. **Note that in our model λ is known. It is not a model parameter which needs to be optimized.**

(c) Implement your algorithm in Python and apply it to the dataset from `demo_auto_mpg.ipynb`. Produce a piecewise linear fit for MPG as a function of Horsepower using the value $\lambda = 100$. Plot the result. You can attach a Jupyter notebook to your submission, or simply include the printed code and plot.

(d) (**3pts bonus**) Modify your approach to handle the case when λ is unknown. Again obtain a fit for MPG vs. horsepower. What value of λ gives the optimal fit? Include any modified code and a plot of your result.

Problem 4: Thinking About Data Transformations (15pts)

Supposed you are trying to fit a multiple linear regression model for a given data set. You have already transformed your data by appending a column of all ones, which resulted in a final data matrix:

$$X = \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & \dots & x_{1,d} \\ 1 & x_{2,1} & x_{2,2} & \dots & x_{2,d} \\ \vdots & \vdots & & \vdots & \\ 1 & x_{n,1} & x_{n,2} & \dots & x_{n,d} \end{bmatrix}$$

However, your model does not seem to be working well. It obtains poor loss in both training and test.

(a) A friend suggests that you should try mean centering your data columns. In other words, for each i, compute the column mean $\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{j,i}$ and subtract $\bar{x}_i$ from every entry in column i. Note that we won't mean center the first column, as doing so would set the 1s to 0s. You try this, but mean centering gives no improvement in the model loss at all.

Use a mathematical argument to explain why this is the case. **Hint:** It does not depend on the specific data set – mean centering will never help improve a multiple linear regression model!

(b) Another friend suggests normalizing your data columns to have unit standard deviation. In other words for each i, compute the column standard deviation $\sigma_i = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(x_{j,i} - \bar{x}_i)^2}$ and *divide* every column by σ_i. Again you try it, but normalizing gives no improvement in the model loss at all.

Use a mathematical argument to explain why this is the case.

(c) A third friend suggests that the issue may be in how you one-hot encoded some binary data. In particular, each data points contains a class attribute that is either `red`, `blue`, or `green`. You encoded the attributes in three columns so that `red` maps to $[0, 0, 1]$, `blue` maps to $[0, 1, 0]$, and `green` maps to $[1, 0, 0]$. Your friend suggests instead trying red to `red` maps to $[1, 0, 0]$, `green` to $[0, 1, 0]$, and `blue` to $[0, 0, 1]$. You do this, but again the model loss does not improve.

Use a mathematical argument to explain why this is the case.

Ultimately, you give up, and find new friends.