New York University Tandon School of Engineering
Computer Science and Engineering

CS-GY 6923: Written Homework 1.
Due Monday, Feb. 16th, 2026, 11:59pm.

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

10% *extra credit will be given if solutions are typewritten (using LaTeX, Markdown, or another mathematical formatting program). MSWord with Equation Editor does not count.*

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

Consider the standard multivariate linear model of the form:

$$f_{\boldsymbol{\beta}}(\mathbf{x}) = \langle \boldsymbol{\beta}, \mathbf{x} \rangle.$$

In class, we saw how to optimize this model under the sum-of-squares loss, which for a training data set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ is equal to:

$$L_{SS}(\boldsymbol{\beta}) = \sum_{i=1}^{n} (\langle \boldsymbol{\beta}, \mathbf{x}_i \rangle - y_i)^2.$$

Sometimes, we are interested instead in using a *weighted* ℓ_2 loss that weights different training points differently. Specifically, suppose we are given a list of positive weights $w_1, \ldots, w_n > 0$. We might want to minimize the *weighted sum-of-squares* loss:

$$L_{WSS}(\boldsymbol{\beta}) = \sum_{i=1}^{n} w_i \cdot (\langle \boldsymbol{\beta}, \mathbf{x}_i \rangle - y_i)^2.$$

Weighted loss functions are often used when some of your data is considered "more reliable" (e.g., because it was collected with a more accurate instrument). You might set the weights for the more reliable examples higher than those of less reliable examples. Weighted loss functions will also be important when we discuss a technique called "boosting" later in the semester.

(a) Derive an expression for the gradient of $L_{WSS}(\boldsymbol{\beta})$. Note that $w_1, \ldots, w_n$ are fixed constants given to us in advance. They are not parameters of the model.

(b) Write down a closed form expression for $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} L_{WSS}(\boldsymbol{\beta})$. I.e., for the parameters that minimize the weighted loss. Your expression should include the given weights $w_1, \ldots, w_n$ in some way.

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.

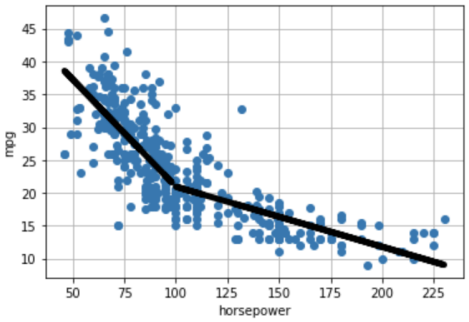
Problem 3: Piecewise Linear Regression via Feature Transformations (15pts)

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.

Problem 4: Thinking About Data Transformations (15pts)

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

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

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) Would your answers above change if you are using ℓ_1 loss or ℓ_∞ loss instead of sum-of-squares (ℓ_2) loss?

Problem 5: Student Question: Slightly More Efficient One-hot Encoding (5pt bonus)

A student in Section B of the class made the following thoughtful observation. Suppose we have a binary categorical variable (e.g., a column that takes values `manual_transmission` or `automatic_transmission` for the car dataset). Typically, we would encode this categorical variable using just one column of 0's and 1's. However, one-hot encoding would actually use two columns:

$$\overbrace{\begin{bmatrix}0\\0\\1\\0\\1\\1\\1\end{bmatrix}}^{\text{standard encoding}} \quad \overbrace{\begin{bmatrix}0 & 1\\0 & 1\\1 & 0\\0 & 1\\1 & 0\\1 & 0\\1 & 0\end{bmatrix}}^{\text{one-hot encoding}}$$

The first column is an indicator for if the car falls into the category `manual_transmission` and the second is an indicator for if it falls into the category `automatic_transmission`. A natural question is if a more compact representation also exists for $k > 2$ categories. Can we always encode k categories with $k - 1$ columns that do not encode unintentional linear relationship?

Prove that this *is possible* for multivariate linear regression. In particular, given a set of k columns obtained via one-hot encoding, prove that we can always remove *any one of the columns* and the output of fitting a linear model with not change. I.e., suppose $\boldsymbol{\beta}^* = \arg\min_{\boldsymbol{\beta}} \|\mathbf{X}\boldsymbol{\beta} - \mathbf{y}\|_2^2$, where $\mathbf{X}$ is our data matrix after one-hot encoding, and $\tilde{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \|\tilde{\mathbf{X}}\boldsymbol{\beta} - \mathbf{y}\|_2^2$, where $\tilde{\mathbf{X}}$ is obtained by removing a single one-hot encoded column from $\mathbf{X}$. For example, if we had a feature with three categories, we might have:

$$\overbrace{\begin{bmatrix} \dots & 0 & 0 & 1\\ \dots & 0 & 1 & 0\\ \dots & 1 & 0 & 0\\ \vdots & \vdots & \vdots & \vdots\\ \dots & 0 & 1 & 0\\ \dots & 1 & 0 & 0\\ \dots & 1 & 0 & 0\\ \dots & 0 & 0 & 1\end{bmatrix}}^{\mathbf{X}} \quad \overbrace{\begin{bmatrix} \dots & 0 & 0\\ \dots & 0 & 1\\ \dots & 1 & 0\\ \vdots & \vdots & \vdots\\ \dots & 0 & 1\\ \dots & 1 & 0\\ \dots & 1 & 0\\ \dots & 0 & 0\end{bmatrix}}^{\tilde{\mathbf{X}}}$$

Your goal is to prove that $\|\tilde{\mathbf{X}}\tilde{\boldsymbol{\beta}} - \mathbf{y}\|_2^2 = \|\mathbf{X}\boldsymbol{\beta}^* - \mathbf{y}\|_2^2$. As usual, assume that $\mathbf{X}$ contains an "intercept" column consisting of all ones.