Homework 8: Multiple Linear Regression

EECS 245, Fall 2025 at the University of Michigan

due Friday, October 31st, 2025 at 11:59PM Ann Arbor Time

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Homeworks are due to Gradescope by 11:59PM on the due date. See the **syllabus** for details on the slip day policy.

Homework will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should always explain and justify your conclusions, using sound reasoning. Your goal should be to convince the reader of your assertions. If a question does not require explanation, it will be explicitly stated.

Before proceeding, make sure you're familiar with the **collaboration policy**.

Total Points: 8 + 10 + 8 + 16 + 11 + 12 + 10 + 6 = 81

Problem 1: Getting Started (8 pts)

Suppose we'd like to fit a hypothesis function of the form $h(x_i) = w_0 + w_1 x_i^2$. Notice the squared term; this is **not** a simple linear regression line.

To do so, we'll find the optimal parameter vector $\vec{w}^*$ that satisfies the normal equations. The first 5 rows of our dataset are as follows, though note that our dataset has n rows in total.

x_i	y_i
2	4
-1	2
3	5
-7	3
3	-7
$\vdots$	$\vdots$

Suppose $x_1, x_2, ..., x_n$ have a mean of $\bar{x} = 5$ and a variance of $\sigma_x^2 = 8$.

a) (2 pts) Write the first five rows of the design matrix, X.

b) (3 pts) Suppose that after solving the normal equations, we find $\vec{w}^* = \begin{bmatrix} 5 \\ -2 \end{bmatrix}$. Find the augmented feature vector $\text{Aug}(\vec{x}_3)$ and squared loss for row 3 of the dataset.

c) (3 pts) Let $X_{\text{tri}} = 3X$, where X is the full design matrix for our dataset with n rows. Determine the bottom-left value in the matrix $X_{\text{tri}}^T X_{\text{tri}}$, i.e. the value in the second row and first column. Your answer should be an expression involving n. *Hint: Use the hint from **Problem 2b of Homework 7**; you can use it without proof, since you already proved it there.*

Problem 2: Moving Things Around (10 pts)

Let X be an $n \times 4$ design matrix whose first column is all 1s, let $\vec{y}$ be an observation vector, and let $\vec{w}^* = (X^T X)^{-1} X^T \vec{y}$, where $\vec{w}^* = \begin{bmatrix} w_0^* \\ w_1^* \\ w_2^* \\ w_3^* \end{bmatrix}$.

In this problem, you'll reason about modifications to the design matrix and see how they affect the components of $\vec{w}^*$.

a) (3 pts) Let X_a be the design matrix that results from **swapping the first two columns of** X. Let $\vec{v}^* = (X_a^T X_a)^{-1} X_a^T \vec{y}$. Express the components of $\vec{v}^*$ in terms of $w_0^*, w_1^*, w_2^*, w_3^*$.

b) (3 pts) Let X_b be the design matrix that results from **adding 3 to each entry in the *first* column of** X. Let $\vec{v}^* = (X_b^T X_b)^{-1} X_b^T \vec{y}$. Express the components of $\vec{v}^*$ in terms of $w_0^*, w_1^*, w_2^*, w_3^*$.

c) (4 pts) Let X_c be the design matrix that results from **adding 3 to each entry in the *second* column of** X. Let $\vec{v}^* = (X_c^T X_c)^{-1} X_c^T \vec{y}$. Express the components of $\vec{v}^*$ in terms of $w_0^*, w_1^*, w_2^*, w_3^*$.

Problem 3: The Sum of Errors (8 pts)

Consider a set of n points, $(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_n, y_n)$, where each $\vec{x}_i$ is a feature vector in $\mathbb{R}^d$ and each y_i is a scalar.

a) (4 pts) To fit the model

$$h(\vec{x}_i) = w_0 + w_1 x_i^{(1)} + w_2 x_i^{(2)} + ... + w_d x_i^{(d)} = \vec{w} \cdot \text{Aug}(\vec{x}_i)$$

we minimize mean squared error,

$$R(\vec{w}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \vec{w} \cdot \text{Aug}(\vec{x}_i))^2 = \frac{1}{n} \|\vec{y} - X\vec{w}\|^2$$

meaning that $\vec{w}^*$ is chosen to satisfy the normal equations. Explain why the components of the error vector,

$$\vec{e} = \vec{y} - X\vec{w}^*$$

are **guaranteed** to sum to 0.

b) (4 pts) If we decide instead to fit the model

$$h(\vec{x}_i) = w_1 x_i^{(1)} + w_2 x_i^{(2)} + ... + w_d x_i^{(d)} = \vec{w} \cdot \vec{x}_i$$

which has no intercept term, are the components of the error vector $\vec{e} = \vec{y} - X\vec{w}^*$ still guaranteed to sum to 0? If they are, explain why. If they are not, explain why not, but give at least one example dataset where they still do sum to 0.

Problem 4: The Complete Solution (16 pts)

Consider the **rank 2** matrix A and vector $\vec{b}$ defined below.

$$A = \begin{bmatrix} 1 & 1 & 0 & -4 & 4 \\ 0 & 1 & 2 & -5 & 3 \\ 2 & -1 & -6 & 7 & -1 \\ 1 & -1 & -4 & 6 & -2 \end{bmatrix}, \quad \vec{b} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \end{bmatrix}$$

a) (4 pts) By hand, find **one** vector $\vec{x}^*$ that minimizes $\|\vec{b} - A\vec{x}\|^2$. (We're intentionally using different variables here than in the notes to have you think about the problem in general terms.)

b) (4 pts) Find a basis for nullsp(A). *Hint: Try and do so efficiently, since this is the type of problem we'll see on Midterm 2.*

c) (2 pts) Show that if $\vec{x}'$ satisfies the normal equations, and $\vec{x}_0 \in$ nullsp(A), then $\vec{x}' + \vec{x}_0$ also satisfies the normal equations. *Hint: This is two-line solution; we're mostly asking it so that you interalize **what** this means and why it's true.*

d) (3 pts) Describe, using set notation, the complete set of vectors $\vec{x}^*$ that minimize $\|\vec{b} - A\vec{x}\|^2$. Is this set a subspace?

e) (3 pts) Use `sklearn`'s `LinearRegression` class to find a vector $\vec{x}^*$ that minimizes $\|\vec{b} - A\vec{x}\|^2$. Include a screenshot of your code and the model's parameter vector.

You should find that it picked a $\vec{x}^*$ that **is** in the set from **c)**, but almost certainly isn't the one from **a)**. In addition to your screenshot, provide an educated guess of **why** you think it picked the $\vec{w}^*$ that it did. (We don't yet have the mathematical background to explain what it picked — but by Homework 10, we will!)

Hint: You'll need to use `fit_intercept=False` when instantiating the `LinearRegression` object, since matrix 'A' here does not have a column of all 1's.

Problem 5: Home Run! (Not Autograded!) (11 pts)

In this problem, we'll work with a real dataset that describes the number of home runs in the MLB per year. Our goal will be to fit various models that predict the number of home runs, y_i, given the year, x_i. To access the dataset, open the **the supplemental Jupyter Notebook** we've created for Homework 8, which can either be found **here** on DataHub, or **here** in the course GitHub repository.

This problem is **not** autograded. In parts **a)** through **c)**, you are given a model. For each one:

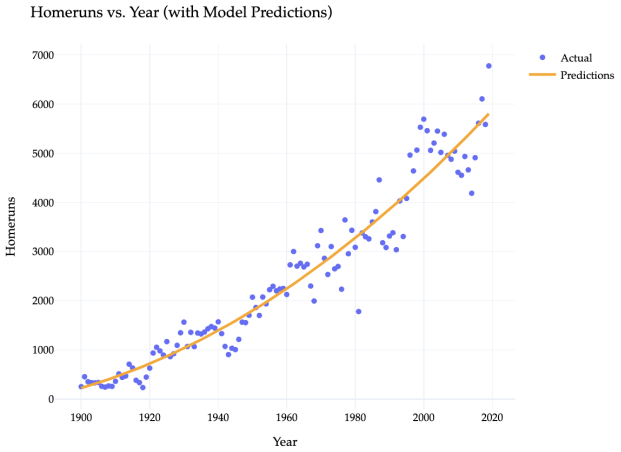
1. Implement the necessary data transformations and use `sklearn`'s `LinearRegression` class to fit the model.
2. In your PDF, **(1) include screenshots of your code and a scatter plot of your model's predictions and (2) report the formula for $h(x_i)$, the *root* mean squared error of the model's predictions, and the R^2 score of the model's predictions.**

Root mean squared error is the square root of mean squared error; its values are in the same units as y. The R^2 score of a model's predictions is equal to correlation(actual y values, predicted y values)^2.

There are helper functions in the supplemental Jupyter Notebook to help you with all steps; start by understanding how they work. Part **a)** has been done for you as an example.

a) $h(x_i) = w_0 + w_1 x_i + w_2 x_i^2$

Solution: The fit model has the formula $h(x_i) = 769420 - 829.91x + 0.22372x^2$ with a root mean squared error of 438.07 home runs and an R^2 score of 0.93346. Code can be found in the supplemental Jupyter Notebook.



b) (4 pts) $h(x_i) = w_0 + w_1 x_i + w_2 x_i^2 + w_3 x_i^3 + w_4 x_i^4$

c) (4 pts) $h(x_i) = w_0 + w_1 x_i + w_2 (x_i - 1900) \cos\left(\frac{2\pi}{25} x_i\right) + w_3 e^{\frac{x_i - 1900}{100}}$

d) (3 pts) Why **can't** we use linear regression to fit the model $h(x_i) = w_0 + w_1 x_i + w_2 (x_i - 1900) \cos(w_3 x_i) + w_4 e^{\frac{x_i - 1900}{w_5}}$?

Problem 6: Polynomial Regression (Not Autograded!) (12 pts)

In this problem, we'll explore the implications of adding more and more complex features to a model.

a) (4 pts) Suppose $\vec{y} \in \mathbb{R}^n$ is an observation vector, X is an $n \times (d+1)$ design matrix, and X' is an $n \times (d+2)$ design matrix whose first $d+1$ columns are identical to X's and whose last column contains some other new information.

$$X' = \begin{bmatrix} & & | \\ X & & \vec{x}^{(d+1)} \\ & & | \end{bmatrix}$$

These design matrices correspond to models $h(\vec{x}_i)$ and $h'(\vec{x}_i)$, in which $h'(\vec{x}_i)$ includes one additional feature. (The $'$ is not a derivative.)

Let $\vec{w}^* \in \mathbb{R}^{d+1}$ be a minimizer of $\frac{1}{n}\|\vec{y} - X\vec{w}\|^2$ and $\vec{w}' \in \mathbb{R}^{d+2}$ be a minimizer of $\frac{1}{n}\|\vec{y} - X'\vec{w}\|^2$. In general, the components of $\vec{w}^*$ and $\vec{w}'$ have no direct relationship; adding a new feature can change the optimal parameters for all of the existing features.

Explain why

$$\frac{1}{n}\|\vec{y} - X'\vec{w}'\|^2 \leq \frac{1}{n}\|\vec{y} - X\vec{w}^*\|^2$$

The inequality above is equivalent to the statement "adding a feature never increases the mean squared error of the model's predictions on the data it was trained on".

Hint: We're looking for a 1-2 sentence English explanation that involves a discussion of spans and column spaces; this is not an algebraic manipulation problem.

b) (5 pts) In the previous part, we had you think about how adding a new feature impacts the model's **training** mean squared error, i.e. its performance on the n data points it was trained on. But, adding new features can hurt a model's performance on unseen data, often called **test** data, which is ultimately the performance we care about. So, to assess a model's ability to **generalize** to unseen data, we typically split our data into a training set and test set, fit the model on just the training set, and then evaluate its performance on the test set.

Here's the general idea we'll explore. Suppose we want to fit a polynomial regression model of degree d, i.e. a model of the form

$$h(x_i) = w_0 + w_1 x_i + w_2 x_i^2 + ... + w_d x_i^d$$

where x_i is a scalar. The question is, what value of d should we choose? As d increases, the model's training mean squared error will stay the same or decrease. But, its test mean squared error may increase.

Open the **the supplemental Jupyter Notebook** we've created for Homework 8, which can either be found **here** on DataHub, or **here** in the course GitHub repository.

In that notebook, we provide you with code that implements a train-test split, and code that uses `sklearn`'s `Pipeline` and `PolynomialFeatures` classes to quickly fit polynomial regression models of varying degrees.

This problem is **not** autograded. Your job is to use that code to:

1. Fit 25 polynomial regression models on the training set (degree 1, degree 2, ..., degree 25).
2. For each model, compute both its training and test mean squared error.
3. Create a line plot of the training and test mean squared error vs. degree.
4. **Include screenshots** of all of your code and the resulting plot in this PDF.

c) (3 pts) You'll have noticed that the choice of d (degree) with the smallest **test** mean squared error is a relatively small number. (There will be a few values of d with very similar test mean squared errors, but hover over your plot: there is a unique minimum.)

In 1-3 English sentences, explain **why** adding more and more features (i.e. increasing d) leads to worse model performance on the test set, and what lesson we can take from this in the context of building (not necessarily just polynomial) regression models that generalize to unseen data.

If you'd like to learn more about the core ideas explored in this problem, consult **EECS 398: Practical Data Science**.

Problem 7: One Hot Encoding in `sklearn` (Not Autograded!) (10 pts)

In Chapter 3.2, we showed you how to one hot encode a categorical feature **manually** in `pandas`. In general, we'll want to use `sklearn`'s `OneHotEncoder` class to one hot encode a categorical feature.

Open the **the supplemental Jupyter Notebook** we've created for Homework 8, which can either be found **here** on DataHub, or **here** in the course GitHub repository.

This problem is **not** autograded. Your job is to read through the code provided in the notebook, and then complete the task next to **Your Job**. Here, you should include:

1. Screenshots of your code and the model's training and test mean squared errors.
2. The number of parameters in the model, including the intercept term; include screenshots of the code you used to find this number.
3. A 1-2 sentence explanation of why the model has this number of parameters.

Problem 8: Anonymous Feedback (6 pts)

We'd like to get your feedback on how the course has been going so far, now that we're past the halfway point and Midterm 2 is fast approaching.

You can find the survey **at this link**, which you should complete **after you've finished the rest of Homework 8**. Unlike the Homework 3 survey, **it is anonymous**, so feel free to provide candid feedback.

In order to earn the 6 points for Homework 8, Problem 8, include a screenshot of the confirmation message you see after submitting the form. (We consider it an honor code violation to include a screenshot if you didn't actually submit the form!)

Thank you for your feedback once again!