Homework 3

SAAS DF

Spring 2025

Once again, only problems marked with (*) are expected to be completed. However, the other ones could be fun, and a good primer for future classes!

1 Algebra: A Universal Language

"Algebra is, in a sense, an exceedingly precise language, designed to be as general as possible. In this pursuit of generality, modern mathematicians—particularly under the influence of Bourbaki—have shaped one of the most rigorous yet remarkably expressive formal systems in existence."

- Nikola Clinchant

Lecture Review

Note: *Skip this blurb of text for now*, and come back if you're struggling with Problem 1.

Often, if we write things down the right way, we can generalize them. In lecture, we saw one example of this. The "simple" linear regression can be written down like this

$$\text{Given data } x \text{ and } y\text{, fit parameters } m \text{ and } b \text{ so that } y \approx mx + b.$$

Cool, but what if we have multiple inputs, like in the case of our housing example, x_1 the number of rooms, and x_2 the square footage of the house?

It turns out that by tabulating our data into a *vector*, we can use the same sentence to describe our "generalized" linear regression model:

$$\text{Given data } \vec{x} \text{ and } y\text{, fit parameters } \vec{m} \text{ and } b \text{ so that } y \approx \vec{m} \cdot \vec{x} + b.$$

To make this make sense, we just have to specify what we mean by this multiplication between vectors. What are the constraints? $\cdot$ must take in two vectors, and output a scalar. One particularly nice example of this is a "dot product" or "inner product".

We can even generalize further, if we want to predict multiple things. Simply tabluate $\vec{y}$ into a vector, and we can turn our problem into:

$$\text{Given data } \vec{x} \text{ and } \vec{y}\text{, fit parameters } M \text{ and } \vec{b} \text{ so that } \vec{y} \approx M\vec{x} + \vec{b}.$$

where now $M\vec{x}$ is a product between a matrix and a vector. We could go even further if we turned x into a matrix. The benefit of having the same sorts of sentences to describe regression regardless of the specifics of our problem format (a universal language[1]) is that they can all be coded up the same sort of way, pushing the details out to very small and easy-to-write methods. This is the beauty of algebra and abstraction: it means that you as the user won't have to think about these details as much[2] when you implement your model.

[1] I'm sort of making this terminology up by the way, but in mathematics and specifically category theory, it has a more precise meaning.

[2] I do hesitate to say this though, since the more you think about the details, the less likely you are to make a grave mistake. Convenience is sort of a risk you sometimes take for ease and speed if that's what you're looking for.

Problem 1 (*). Logistic regression is usually written down like this:

Given data x and y, fit parameter θ so that $y \approx \frac{1}{1+e^{-\theta x}}$.

Step 1: What are the dimensions of x, y, and θ? *Hint: the dimension of a real number is just 1.*
Step 2: Now I have a different problem, where my x is a $n \times 1$ vector and y is a scalar. What should the dimension of θx be?
Step 3: Propose what the dimension of θ should be, along with an appropriate method of multiplying θ and x so that it has the dimension you specified in Step 2.
Step 4: Rewrite the sentence again so that it looks almost the same, but describes your new problem.

Feel free to update the notation to use common nomenclature like $\vec{}$ or upper case letters to denote vectors or matrices.

2 Mock DM Interview (*)

Now that you've learned about EDA and basic modeling, you might be able to try this mock DM interview (mainly the coding portion)! Take 25 minutes to try out the mock interview in `mock_dm_code.py`. Afterwards, note down what you struggled with or what took the most time to figure out, so that you can improve on it for the remainder of the semester (or if you didn't struggle, mention how easy it was)!

Remark. Depending on what the next semester's DM directors choose to do, this mock interview might not be super representative of what it will be in the future, but it does reflect what past interviews have been like.

3 Overfitting

When you have an overly complex model, you might "overfit your data", which means that your model will work very well on the data you trained with, but will perform pretty badly on new data. Check out `overfitting.ipynb` for an exploration of this.