Homework 12

CAS CS 132: Geometric Algorithms

Due: **Tuesday December 12, 2023 at 11:59PM**

Submission Instructions

This week there is only a programming assignment. See below for instructions on submission to Gradescope.

Practice Problems

The following list of problems comes from *Linear Algebra and its Application 5th Ed* by David C. Lay, Steven R. Lay, and Judi J. McDonald. They may be useful for solidifying your understanding of the material and for studying in general. **They are optional, so please don't submit anything for them**.

- 6.6.2-4, 6.6.6, 6.6.9, 6.6.10

1 Linear Models (Programming)

In this problem, you will be finding models for the cost of homes in California based on 1990 census data using multiple regression. This is very similar to the example given in the section "Multiple Regression in Practice" from the text so please make sure to read this example if you want additional guidance.

The objective of this problem will not be implementing model fitting (as is the case for the example from the text). This will be implemented for you via a single call to `numpy.linalg.lstsq`. Rather, your primary objective is to build design matrices for linear, quadratic, and cubic functions. You can use any function in NumPy we've seen in this course, but it may be useful to look at the following functions before you get started:

- `np.ones`
- `np.column_stack`
- `np.linalg.norm`
- Recall that `a[:,i]` is the `i`th column of `a`.

You are given starter code in the file `hw12prog.py`. **Do not change the name of this file when you submit.** Also don't change the names of any functions included in the starter code. **The only changes you should make are to fill in the provided TODO items.** In particular, you cannot change the imports to the file. You may also notice that this file depends on scikit-learn. **Please verify early that you can work with this dependency.** If you set everything up for Homework 10, then this should require no additional work.

There are four function you need to fill in.

- `distance`, which calculates the distance between two vectors in $\mathbb{R}^n$, represented as 1D NumPy arrays.
- `linear_design_matrix`, which builds the design matrix for simple multiple regression, i.e., for finding a model of the form

$$f(x_1, \ldots, x_n) = \beta_0 + \sum_{1 \leq i \leq n} \beta_i x_i$$

  This matrix should be identical to the one in the text. The independent variables are given to you as a matrix, where each row represent a datapoint. The dataset has 8 independent variables, so the input `ind_vars` is a 2D NumPy array with 8 columns, and the design matrix you return on the input `ind_vars` should have 9 columns.

- `quadratic_design_matrix`, which builds the design matrix for multiple regression with a quadratic modeling function, i.e.,

$$f(x_1, \ldots, x_n) = \beta_0 + \sum_{1 \leq i \leq n} \beta_i x_i + \sum_{1 \leq i \leq j \leq n} \beta_{ij} x_i x_j$$

This will require adding many new columns; the design matrix you return on the input `ind_vars` should have 45 columns.

- `cubic_design_matrix`, which builds the design matrix for multiple regression with a cubic modeling function, i.e.,

$$f(x_1, \ldots, x_n) = \beta_0 + \sum_{1 \le i \le n} \beta_i x_i + \sum_{1 \le i \le j \le n} \beta_{ij} x_i x_j + \sum_{1 \le i \le j \le k \le n} \beta_{ijk} x_i x_j x_k$$

  The design matrix you return on the input `ind_vars` should have 165 columns. As a hint, note that many of the columns you have to add are the same ones you added in the previous function; don't redo your work.

When you run the file, you should see a small report on the data and three error calculations for each kind of model. If you've done everything correctly, your errors should be around 104, 93, and 86 for linear, quadratic, and cubic functions, respectively (note that the error goes down as the models become more complex). There is also some commented-out code for graphing the predictions against the actual values (as in the text). I highly recommend running this to see how the prediction accuracy "looks" for each model, but you don't have to submit anything for this.

You will upload a single file `hw12prog.py` to Gradescope, where you can verify that it passes some (but not all) autograder tests.