CSC 395 Homework 6 Spring 2024

Due: Friday, May 10

In this homework assignment, you will be exploring several different instances of least squares and polynomial interpolation. In the first part of this assignment, you will be using linear least squares to help answer questions about how various crime rates scale with population. In the second part of the assignment, you will be provided with code that can create noisy data centered on a given function, and your goal will be to use linear least squares to fit a higher order polynomial to this data. In the final part of the assignment, you will explore polynomial interpolation using Lagrange polynomials. The three parts of this assignment, which you should put in the files `hw6p1.py`, `hw6p2.py`, and `hw6p3.py`, should all be submitted to the course gradescope.

## Part 1. **Crime Rate and Population Size**

You have been provided with part of a data set containing crime statistics for counties in the United States over some time range (which is honestly unclear, but wont affect our analysis) in the file `crime_data.csv`. Note that this is a `csv`, which is a file of "comma separated values," i.e., the columns of the data are separated with commas between the values. The first row of the data contains the column labels, which are:

- `County`: the county recorded in this row.
- `State`: the US State that contains this county.
- `MURDER`: the number of murders in this county.
- `ROBBERY`: the number of robberies in this county.
- `AGASSLT`: the number of aggravated assaults in this county.
- `BURGLRY`: the number of burglaries in this county.
- `LARCENY`: the number of larcenies in this county.
- `MVTHEFT`: the number of motor vehicle thefts in this county.
- `ARSON`: the number of arsons in this county.
- `population`: the population of this county.

Your first task will be to read and process this file. Note that there are counties that are small enough that some of their crime numbers are actually 0. This can cause problems for the method we are going to use, so my advice is to simply remove any county that has *any* 0 entries (there are thousands of counties in the data set, so removing these very small counties wont meaningfully affect the results).

Now that the data is ready, it is time to discuss the background model we will be using. When considering social data like crime rates, social interactions, or communal infrastructure, it is common to ask how the data in question scales with the population size. The commonly used model is a scaling relation of the form

$$S \sim N^{\gamma},$$

where $S$ is the social factor under examination, $N$ is the population size, and $\gamma$ is the scaling factor. When $\gamma = 1$, we say that the social factor scales linearly with the population (doubling the population results in a doubling of the social factor). When $\gamma < 1$, there is a sub-linear relationship, and when $\gamma > 1$, then scaling is super-linear.

A natural first question to ask is: what value of $\gamma$ should we expect?

When we discussed the SIR model of disease propagation, we introduced the idea that factors that scale with human interaction, where humans are effectively acting like reagents in a chemical reaction (like a sick person and a susceptible person interacting to produce two sick people), we expect the number of interactions to scale quadraticly with the population. This would result in a scaling factor of $\gamma = 2$.

But is this a reasonable model for all social factors? We could imagine, for example, that certain types of crime could exhibit this sort of scaling (in order for an assault to occur, two people have to physically interact with each other). But if we consider something like arson, that is the result of a criminal interacting with a physical structure, not another person.

Moreover, while sick vs susceptible is not something a single person can readily control (i.e., if you bump into a sick person and get sick, there is usually not much you can do about it), crime is more commonly a conscious choice on behalf of the criminal, and if often targeted with intention, not just the result of randomly bumping into victims. So perhaps modeling crime as a chemical reaction is not a good fit.

So what sort of scaling do we see for various types of crime? Your remaining tasks for this first part of the assignment will be to use linear least squares to determine the scaling factor $\gamma$ for the various types of crime data provided.

Let's revisit that scaling model $S \sim N^\gamma$. This looks like a very nonlinear equation with respect to $\gamma$, and if we were to try to fit a value of $\gamma$ directly, we would have to employ nonlinear least squares, likely using something like the Gauss-Newton method. However, we can utilize a clever trick that we have already seen a few times in this course: take the logarithm of both sides! This results in the equation

$$\log S = \gamma \log N + c,$$

where $c$ is a constant related to the log of the constant of proportionality (and doesn't have much meaning for us beyond the need to fit both $\gamma$ and $c$).

With this equation in hand, we can see that this model is now linear in our unknown parameters $\gamma$ and $c$, allowing us to use linear least squares.

You should now take the logarithm of all of the data you collected, and proceed to setup the necessary vectors and matrices for use in linear least squares. Note that the parameter vector is defined as

$$\vec{\xi} = \begin{bmatrix} \gamma \\ c \end{bmatrix}.$$

Once you have set up the necessary matrix $J$ and vector $y$ for use in linear least squares, you may use the function `numpy.linalg.solve` to solve the matrix equation to determine the best fit $\xi$:

$$J^T J\xi = J^T y.$$

Recall that $J$ and $y$ come from the equation

$$r = J\xi - y,$$

where $r$ is the residual vector. For our linear fitting, this means that the $i$th component of this vector is given by

$$r_i = \gamma \log N_i + c - \log S_i.$$

This equation should assist you in creating the matrix $J$ and vector $y$.

Perform this linear least squares fitting task for each of the provided crime statistics, and report your scaling.

As a final task for this first part of the assignment, consider the scaling relationships you have found. Provide a brief discussion of your results: Do they make sense to you? Is there anything that surprised you?

You may choose to discuss all of the reported results, or focus your discussion on a specific value that interested you. I am primarily looking for evidence that you have thought about your results and interpreted them in the context of the real world data you just analyzed.

Make sure all of your code for this part of the assignment is placed in the file `hw6p1.py`, and submit this to gradescope.

Part 2. **Least Squares for Polynomial Fitting**

Your next task in this assignment will be to use linear least square for fitting higher order polynomials to some noisy data.

You have been provided with a pre-started `hw6p2.py` file that contains some code that will create noisy data centered around a given function. Your goal will be to find a best fit polynomial for that data using linear least squares.

The function `noisy_data` will take in a function `func` to sample from, two floats `a` and `b` representing the domain, an integer `N` for the number of data points to generate, and a float `sig` that is the desired standard deviation of the noise.

This function will create a random list of $x$ values in the domain $[a, b]$ (sorted in ascending order), obtain the corresponding $y$ values (given by `func(x)`), and add noise to those $y$ values before returning both the $x$ and noisy $y$ data.

Your task is to write a function `poly_fit(x_vals, y_vals, n)` that will take in the noisy $x$ and $y$ data and find the best fit polynomial of order $n$, represented as a list of the coefficients of the various powers of $x$, in ascending order.

For example, if you are fitting a quadratic polynomial to the data, your output would be a list of the three values `[a0, a1, a2]`, where the fitted quadratic is $a_0 + a_1 x + a_2 x^2$.

But how to go about finding the best fit polynomial? At first, it may seem like this will require some form of nonlinear least squares, since you are fitting a nonlinear function to the data. But don't get fooled, this problem can be solved with linear least squares!

For a polynomial of order $n$, the equation for that polynomial might be nonlinear with respect to $x$, but it is linear with respect to the coefficients $a_0, a_1, \ldots, a_{n-1}, a_n$. Those are the unknown we are solving for, so we can employ linear least squares.

You should again work to set up the necessary matrix $J$ and vector $y$ to use with `numpy.linalg.solve` to solve the system

$$J^T J \xi = J^T y.$$

Here the unknown parameter vector $\xi$ is (note the ordering of the coefficients)

$$\xi = \begin{bmatrix} a_0 \\ \ldots \\ a_n \end{bmatrix}.$$

Again recall that $J$ and $y$ come from the equation

$$r = J\xi - y,$$

where $r$ is the residual vector. For a polynomial fit, this means that the $i$th component of this vector is given by

$$r_i = a_0 + a_1 x_i + \cdots + a_{n-1} x_i^{n-1} + a_n x_i^n - y_i.$$

This equation should enable you to create the matrix $J$.

You have been provided with a simple test function,

$$y = (x-2)(x-3).$$

In the main block of your `hw6p2.py` file, you should create noisy data using this function, then use your `poly_fit` code to find the best fit quadratic. Are your results close to the correct polynomial?

As a final note before you submit this part of the assignment: you have been provided with some code to create a scatter plot of the noisy data that has your best fit polynomial overlaid on top, showing that you have (hopefully) found a reasonable fit!

Part 3. **Lagrange and Chebyshev**

For your final task of this assignment, you will be implementing some basic Lagrange Polynomial Interpolation and exploring the behavior of this technique.

Your primary task now is to write a function `lagrange(x, x_vals, y_vals)` that will return the value of $P(x)$, where $P$ is the interpolated polynomial. Note that your function will be given a specific value of $x$, so you will be outputting the single corresponding $y$ value from the interpolated polynomial.

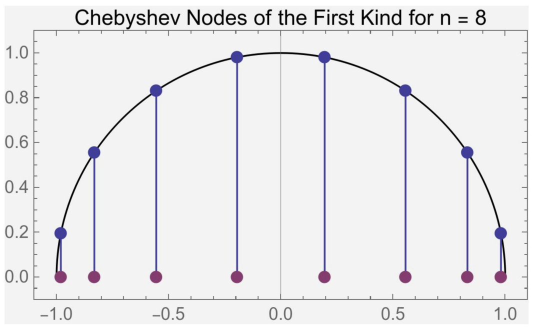
Note: there are a number of different ways to implement this function, and some are more efficient than others. However, it is unusual (and typically undesirable) to use too many points when trying to perform this type of polynomial interpolation, so the efficiency here should not matter (i.e., don't worry about it).

Once you have written your function, you should test it using the code in the main block of the provided `hw6p3.py`. This main block will generate equally spaced $x$ values and use the same quadratic polynomial used to test part 2 of this assignment to generate corresponding $y$ values. There is a variable $n$ you can set at the top of the main block that will control how many points are generated.

Try experimenting with a few different values of $n$, (specifically, $n = 55$, $n = 65$, and $n = 75$ should produce some interesting results, but try other values as well). What do you notice (note: the plotting was setup intentionally to limit the displayed view to better help showcase what is happening).

You should notice an issue that is common for these Lagrange Polynomials: when using too many data points (and therefore a very high order polynomial), the interpolating polynomial fits poorly near the boundary points $a$ and $b$, typically with very large oscillations near those points.

In class we discussed the use of Chebyshev nodes to help address these issues (the second one specifically). The basic idea of Chebyshev nodes is to use points that are not evenly spaced across the domain, but that are instead more clustered near the boundaries where the worst errors are occurring. This is done by considering a unit circle and placing the nodes equally spaced around the circle. The nodes used for the polynomial interpolation are then these nodes projected onto the $x$-axis. This can be seen in the image below.



This idea gives us the following vector of $x$ values

$$x_k = \frac{(a+b)}{2} + \frac{(b-a)}{2}\cos\left(\frac{2k+1}{2n}\pi\right), \quad k = 0, \ldots, n-1,$$

where $n$ is the number of nodes to generate.

Write a function `chebyshev(n, a, b)` that will return the $n$=length list of $x$ values in the domain $[a, b]$. Once you have written this function, use it to repeat the Lagrange fitting task from before, but now using Chebyshev nodes. You should notice an improvement.

Once you have completed this task, submit all parts of this assignment to gradescope.