John Sipala

AMS 595 Assignment 5

https://github.com/jdsipala/AMS595Project5

For the first question, I computed the PageRank scores by using scipy.linalg.eig to find the dominant eigenvector of the matrix, which was the first part of the assignment.

Dominant Eigenvector: [0.30460385+0.j 0.40613847+0.j 0.6092077 +0.j 0.6092077 +0.j]

The second part asked for an iterative approach where I started with a vector of ones and repeatedly multiplied by the matrix until the difference between consecutive vectors became very small. After normalizing the vector, each number represents the long-run probability of being on page (1,2,3 or 4). The results showed that the long-run probability for pages 3 and 4 were the highest because they receive more incoming link probability than the others, while pages 1 and 2 ranked lower. The following was the results from my jupyter notebook. PageRank: [0.15789474 0.21052632 0.31578947 0.31578947]

I loaded the height-weight data and computed the covariance matrix to see how the two variables varied together.

covariance matrix:

[[1.02608749 0.11769063]

[0.11769063 1.08134929]]

Then I used scipy.linalg.eigh to get the eigenvalues and eigenvectors, which gave me the principal components.

Eigenvalues:

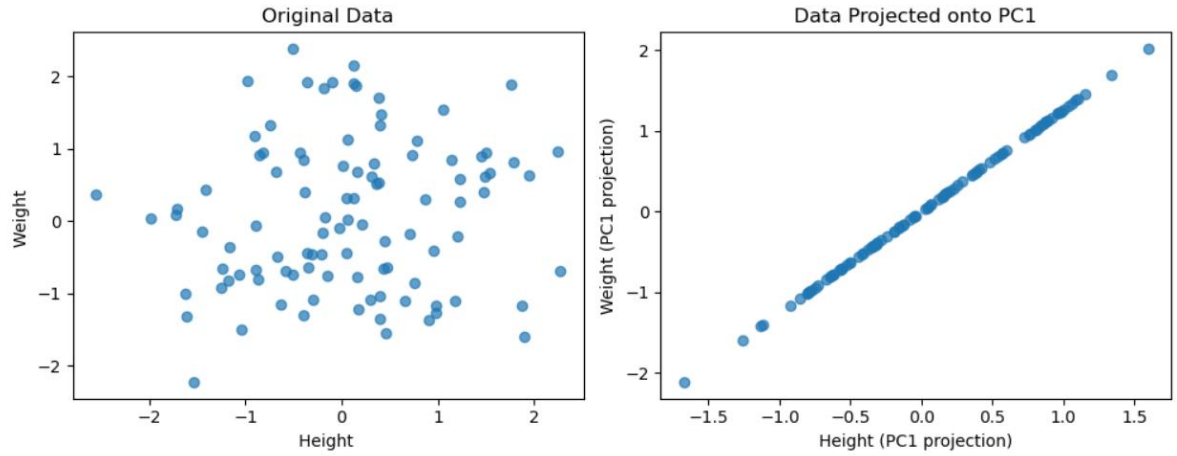
[0.93282774 ,1.17460905]

Eigenvectors (columns are eigenvectors):

[[-0.78376051 0.62106317]

[0.62106317 0.78376051]]

The eigenvector that corresponds with the larger eigenvalue becomes the first principal component since it explains the most variance in the data. To reduce the dataset to 1D, I projected all points onto this direction and then plotted both the original 2D points and their 1D projection. The results shown below show how most of the variation in the dataset lies along a single main trend between height and weight.



For the third question, I set up the housing data in matrix form as XB = y, where X contained square footage, number of bedrooms, and age for each house, and y contained the prices. I used scipy.linalg.lstsq to solve for Beta (Beta coefficients [3.2875, -1628.75 , -77.75]). Using the resulting model, I plugged in the values for a house with 2400 square feet, 3 bedrooms, and 20 years old to get the predicted price which was $1,448,750. Least squares was the right method here since scipy.linalg.solve only works when the system is square and has an exact solution, which is not the case for this dataset.

For the fourth question, I first defined the loss function f(X) = 1/2 * sum (X - A)^2 to measure how close X is to A. Since the gradient of this loss is X - A, I used that in two ways: once with scipy.optimize.minimize and once in a manual gradient descent loop. I initialized both X and A as random 100 by 50 matrices and updated X step by step using X = X − learning_rate * (X − A). In the manual loop, I stopped the iterations when the change in loss between two steps was smaller than a set threshold or when I hit 1000 iterations. The graph below shows how the loss decreased over time as X moved closer to A.

```
final loss: 3.6574953722686076e-06
```