Jake Betenson
CS 3200 | u0624782

Assignment 5 – Gradient Descent and Eigenvalues

I. Introduction

This experiment involves the use of gradient descent for polynomial interpolation and the power method for predicting population behavior. The entire experiment was performed on my personal machine running a second-generation Intel i5 (Sandy Bridge) processor in MATLAB. Some code was reused from the previous assignment, as well as lecture slides.

II. Gradient Descent – US Population

The first experiment uses gradient descent to calculate coefficients for a polynomial approximation of US population data. Population data is shown in Table 1. I first attempted to compute polynomial solutions with degrees 3 through 10 using US census data. To compute a polynomial solution of degree n, the $n + 1$ most recent data were chosen and transformed into a Vandermonde matrix using MATLAB's built in $vander()$ function. Without any scaling or normalization, gradient descent failed to converge within 1 million iterations and an early-out tolerance of 1×10^{-5}.

Year	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990	2000
Scaled Year	-1	-0.8	-0.6	-0.4	-0.2	0	0.2	0.4	0.6	0.8	1
Pop. (Millions)	75.995	91.972	105.711	123.203	131.669	150.697	179.323	203.212	226.505	249.633	281.422

Table 1 – US Census data from 1900 to 200 in millions of persons.

After inspecting the condition value for the 10th degree Vandermonde matrix (11x11), the condition number, κ, is incredibly high at 4.05×10^{44}. This is an incredibly ill-conditioned and must be transformed. To combat the poor condition, time was scaled according to:

$$s = \frac{t - 1950}{50}$$

resulting in a range of $[-1,1]$. After generating the 10th degree Vandermonde matrix with the new s values, the resulting condition number was $\kappa = 1.40 \times 10^{4}$, which is a sizable reduction. The scaled lower degree Vandermonde matrices are conditioned much better. Using the scaled time values, I attempted to compute polynomial approximations of degrees 2 through 5. The least squares residuals and predictions for 2010 and 2019 populations for each degree can be seen in **Table 2**. My best polynomial was the 2nd degree approximation since it had to lowest sum of least squares residuals and the closest predictions. This could be based on how I selected my points for interpolation, results could have differed if I spread my data points across the set, as opposed to just taking the most recent data for interpolation. After selecting the second-degree polynomial, α was scaled by a factor from 1 to .01 to see if it could be made to converge more quickly. A factor of .5 worked best, reducing the number of iterations to converge from 358 to 197, a reduction nearing a factor of 2.

Degree	Sum LSR	2010 Prediction	2019 Prediction
2	**7.71E-06**	**321.87**	**365.82**
3	8.35E-06	330.69	397.29
4	4.74E+153	3.45E+153	9.81E+152
5	5.30E+153	1.41E+153	1.35E+154
Ground Truth	-	**308.745**	**328.239**

Table 2 – Results of scaled gradient descent method, degrees 2 through 5. Population estimates in millions of persons.

III. Eigenvalues & Biological Application

The second experiment uses the power method to find the largest eigenvalues of various matrices then applies properties of eigenvalues to a real-world biology application. The first step of the experiment is to verify my own method, $simplepm()$, versus MATLAB's built-in $eig()$ function on matrix:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

The largest eigenvalue was calculated using a random 3×1 starting matrix and with a result of 16.11 which matches MATLAB's function. I attempted to use the power method again to find the largest eigenvalues for matrix B using starting guess B_0

$$B = \begin{bmatrix} 2 & 3 & 2 \\ 1 & 0 & -2 \\ -1 & -3 & -1 \end{bmatrix} B_0 = \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix}$$

Attempting to compute the largest eigenvalue via the power method fails to converge within $10k$ iterations likely because two of the eigenvalues for B have the same magnitude. Adjusting the starting guess to:

$$B_0 = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

yields a strange result of 1. This is likely due to the fact the initial guess does not have the components of the largest eigenvalues (3, -3). Applying the power method to the finding the smallest eigenvalue of A can be done by calculating the largest eigenvalue of A inverse. Something strange happened here, but my results agree with MATLAB's result of -2.70×10^{16}. This method can be seen in the attached source code in function $inversepm()$.

Finally, I applied my power method to a biological system given by:

$$P^n = A^n P^0, \qquad A = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \\ 1-d_1 & 0 & 0 & 0 \\ 0 & 1-d_2 & 0 & 0 \\ 0 & 0 & 1-d_3 & 1-d_4 \end{bmatrix}, \qquad P^0 = \begin{bmatrix} 100 \\ 200 \\ 150 \\ 75 \end{bmatrix}$$

Where P^n represents four segments of a population after n years and P^0 is the starting populations of each segment, with birth rates $b = [0.3, 0.3, 0.3, 0.1]$ and death rates $d = [0.1, 0.2, 0.5, 0.9]$. Using my power method on A found the largest eigenvalue to be $.91$ which is less than 1. Meaning the population should eventually die off as time goes by. This stands to reason because there is a 90% death rate in the 4th age segment of the population. To confirm this hypothesis, I calculated the population of each age segment after 1000 years. The results were near zero, in the 1×10^{-38} range confirming my hypothesis. I adjusted d_4 to be 0.1 and recomputed to the largest eigenvalue to be 1.08 which means the population should grow unbounded. Calculation the population after a thousand years resulted in population segments in the $1x10^{35}$ ranges, indicating that the population does grow unbounded.

IV. Conclusion

While I had some strange results with the inverse matrix eigenvalue calculations and the polynomial approximations for the US census data, most of the results seem to follow my intuition. I personally enjoyed the modeling segments in both experiments especially with the real-world applications. Population modeling has readily apparent uses and is easy to understand intuitively.