Linear Algebra Midterm Project

2021-12-06

컴퓨터학부 심화컴퓨터전공 2020114658 이윤서

■ There are two iterative methods to estimate an eigenvalue. One is the power method for estimating a strictly dominant eigenvalue. The other is the inverse power method for estimating an eigenvalue λ with roughly estimated eigenvalues. The iterative process of these two methods are described as below.

THE POWER METHOD FOR ESTIMATING A STRICTLY DOMINANT EIGENVALUE

1. Select an initial vector $\mathbf{x}_0$ whose largest entry is 1.
2. For $k = 0, 1, \ldots,$
   a. Compute $A\mathbf{x}_k$.
   b. Let μ_k be an entry in $A\mathbf{x}_k$ whose absolute value is as large as possible.
   c. Compute $\mathbf{x}_{k+1} = (1/\mu_k)A\mathbf{x}_k$.
3. For almost all choices of $\mathbf{x}_0$, the sequence $\{\mu_k\}$ approaches the dominant eigenvalue, and the sequence $\{\mathbf{x}_k\}$ approaches a corresponding eigenvector.

THE INVERSE POWER METHOD FOR ESTIMATING AN EIGENVALUE λ OF A

1. Select an initial estimate α sufficiently close to λ.
2. Select an initial vector $\mathbf{x}_0$ whose largest entry is 1.
3. For $k = 0, 1, \ldots,$
   a. Solve $(A - \alpha I)\mathbf{y}_k = \mathbf{x}_k$ for $\mathbf{y}_k$.
   b. Let μ_k be an entry in $\mathbf{y}_k$ whose absolute value is as large as possible.
   c. Compute $\nu_k = \alpha + (1/\mu_k)$.
   d. Compute $\mathbf{x}_{k+1} = (1/\mu_k)\mathbf{y}_k$.
4. For almost all choices of $\mathbf{x}_0$, the sequence $\{\nu_k\}$ approaches the eigenvalue λ of A, and the sequence $\{\mathbf{x}_k\}$ approaches a corresponding eigenvector.

1. (15pt) Estimate a strictly dominant eigenvalue of a matrix A with initial vector $\mathbf{x}_0$ described as following.

$$A = \begin{bmatrix} 6 & 5 \\ 1 & 2 \end{bmatrix}, \mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Q1.1. What kind of method do you choose? Explain the reason of your selection.

Power Method 방식을 선택하였다. 이 방식은 eigenvalue 중 가장 큰 값에 해당하는 eigenvector 만 구할 수 있고, 이를 통해 strictly dominant eigenvalue 를 구할 수 있기 때문이다. 그러나 이 방식에 있어서는, 언제 반복을 종료할 것인지를 결정하는 것이 관건이고 반복을 종료하는 시점에 따라 오차율이 달라짐이 자명하다.

반복을 거듭할수록 오차율이 줄어들 것이라 가정하였고, 기준을 잡기 위해 앞선 반복에서 나온 eigenvalue (μ_{k-1}) 와 계산된 eigenvalue (μ_k) 의 차이가 0.00001 미만 일 때를 반복을 종료하는 Termination Condition 으로 설정하였다.

Q1.2. Write a code to estimate a strictly dominant eigenvalue of A with initial vector $\mathbf{x}_0$.

파일 첨부하였습니다.

Q1.3. Fill in the blanks in the table below.

Iteration	1	2	3	4	5
$\mathbf{x}_k$	[[1.] [0.4]]	[[1.] [0.225]]	[[1.] [0.20350877]]	[[1.] [0.2005]]	[[1.] [0.2000714]]
$A\mathbf{x}_k$	[[5] [2]]	[[8.] [1.8]]	[[7.125] [1.45]]	[[7.01754386] [1.40701754]]	[[7.0025] [1.401]]
μ_k	5	8	7.125	7.017543859649123	7.0024999999999995

또한 termination condition 이 변화할 때마다 어떻게 달라지는지 양상을 알아보기 위해, 차잇값 (dis) 를 조절하며 Eigenvalue 를 추정해보았다.

```
ellie at Sh3llT3r in ~/Desktop/KNU 2-2/과 제 /선 형 대 수  21-12-22 - 16:24:45
 ▷  > /usr/local/bin/python "/Users/ellie/Desktop/KNU 2-2/과 제 /선 형 대 수 /power_method.py"
dis = 0.001 일 때 , eigenValue = 7.000050999592004
dis = 0.00001 일 때 , eigenValue = 7.000001040799334
dis = 0.0000001 일 때 , eigenValue = 7.000000003034399
dis = 0.0000001 일 때 , eigenValue = 7.0000000000619265
```

dis 값이 작아질수록, 공식에 대입하여 얻을 수 있는 eigenvalue 값인 7 에 수렴한다는 것을 알 수 있다.

2. (15pt) Estimating an eigenvalue λ, which are the two smallest eigenvalues of A with initial vector $\mathbf{x}_0$ described as following.

$$A = \begin{bmatrix} 10 & -8 & -4 \\ -8 & 13 & 4 \\ -4 & 5 & 4 \end{bmatrix}, \mathbf{x}_0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$ with roughly estimated eigenvalues 21, 3.3, and 1.9

Q2.1. What kind of method do you choose? Explain the reason of your selection.

Inverse power method 방식을 선택하였다. 이 방식은 eigen value 중 가장 작은 값에 해당하는 eigen vector 만 구할 수 있으므로 이를 통해 가장 작은 두 개의 eigen value 값을 구할 수 있기 때문이다.

이 방식 역시 Termination Condition 을, 1 번의 Power method 와 마찬가지로 정해진 반복 횟수가 없기 때문에 앞선 반복에서 나온 eigenvalue (ν_{k-1}) 와 계산된 eigenvalue (υ_k) 의 차이가 0.00001 미만 일 때로 설정하였다.

Q2.2. Write a code to estimate the two smallest eigenvalues of A with initial vector $\mathbf{x}_0$.

파일 첨부하였습니다.

Q2.3. Draw two tables to estimate the two smallest eigenvalues of A with initial vector $\mathbf{x}_0$.

the smallest eigenvalues (1.9)

Iteration	0	1	2	3	4
$\mathbf{x}_k$	[[0.5735901] [0.06464924] [1.]]	[[0.50536384] [0.00445297] [1.]]	[[5.00378330e-01] [3.12807717e-04] [1.00000000e+00]]	[[5.00026625e-01] [2.20071850e-05] [1.00000000e+00]]	[[5.00001873e-01] [1.54845794e-06] [1.00000000e+00]]
$\mathbf{y}_k$	[[4.45037353] [0.50160085] [7.7588047]]	[[5.01305786] [0.04417211] [9.91970041]]	[[5.00124682e+00] [3.12649150e-03] [9.99493086e+00]]	[[5.00008939e+00] [2.20064066e-04] [9.99964631e+00]]	[[5.00000630e+00] [1.54845408e-05] [9.99997513e+00]]
μ_k	7.7588046958377745	9.919700410599539	9.99493086039089	9.999646313779571	9.999975129049833
υ_k	2.02888583218707	2.0008094961145666	2.0000507171053	2.0000035369873026	2.00000024871012

the smallest eigenvalues (3.3)

기존의 Termination Condition 을 적용하니 4 번 반복이 되어서, 이번은 차이가 0.000000001 미만일 때로 설정하였다.

Iteration	0	1	2	3	4
$\mathbf{x}_k$	[[1.] [0.79336857] [0.08217203]]	[[1.] [0.78692102] [0.09589333]]	[[1.] [0.78701363] [0.09566708]]	[[1.] [0.78701211] [0.09567077]]	[[1.] [0.78701211] [0.09567077]
$\mathbf{y}_k$	[[41.04536489] [32.56410256] [3.37278107]]	[[47.48666198] [37.36825244] [4.55365394]]	[[47.11950308] [37.08369119] [4.50778528]]	[[47.12516792] [37.0880776] [4.50850106]]	[[47.12516792] [37.0880776] [4.50850106]]
μ_k	41.04536489151939	47.4866619845003	47.119503079176575	47.12516791946625	47.125167919466625
υ_k	3.3243632868813067	3.3210585448252057	3.3212226346767633	3.3212200835381407	3.3212200835381407

이 방법도 마찬가지로 dis 값을 조절해 보았는데, 이 역시 dis 값을 작게 할수록 특정 값에 수렴하는 양상을 보였다. 이는 dis 값을 작게 해 반복횟수를 증가시킬수록 더 정확한 eigenvalue 값을 구할 수 있다는 것을 의미한다.

```
the smallest eigenvalues (1.9)

dis = 0.001 일 때 , eigenValue = 2.0000507171053
dis = 0.00001 일 때 , eigenValue = 2.00000024871012
dis = 0.0000001 일 때 , eigenValue = 2.0000000012312587


Second smallest eigenvalues (3.3)
dis = 0.001 일 때 , eigenValue = 3.3212226346767633
dis = 0.00001 일 때 , eigenValue = 3.3212200835381407
dis = 0.0000001 일 때 , eigenValue = 3.3212201253281113
```

■ RANSAC(RANdom Sample Consensus) is an iterative method to estimate parameters of a mathematical model from a set of observed data that contains outliers, when outliers are to be accorded no influence on the values of the estimates.

3. (20pt) There are 100 points in $\mathbb{R}^2$ in data.txt. We want to estimate a line $y = mx + b$.

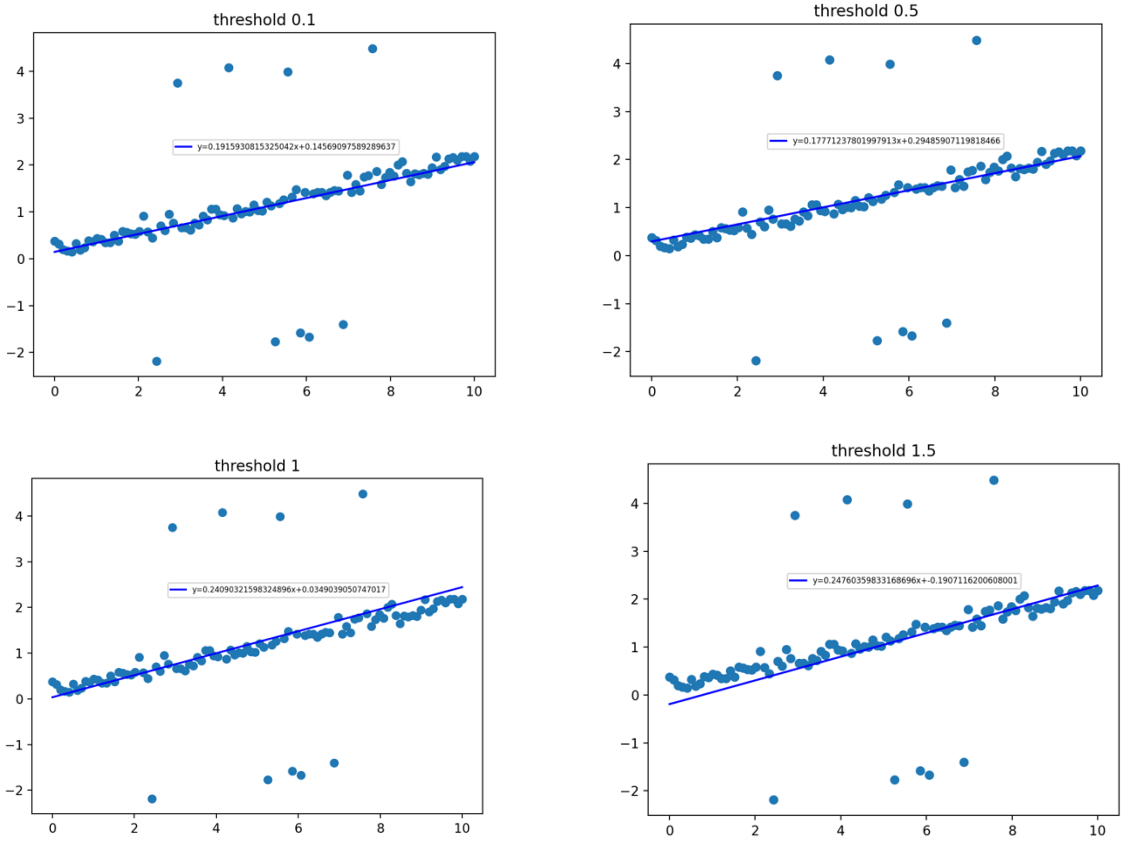
Q3.1. Explain your strategy to estimate a line $y = mx + b$ with 100 observed points included outliers using RANSAC framework.

1. 100 개의 점(data)을 리스트에 넣고, 무작위로 2 개를 선택한다. (복원 추출)
2. 선택된 두개의 점을 지나는 $y = mx + b$ 를 계산한다.
3. 이 직선과의 거리가, 설정된 threshold 범위 내에 있는 점을 inlier 리스트에 넣는다.
4. 설정된 횟수 (500 번) 만큼 반복을 하여, inlier 갯수가 가장 큰 직선을 선택한다.

Q3.2. Write a code to estimate a line $y = mx + b$ with 100 observed points included outliers using RANSAC framework.

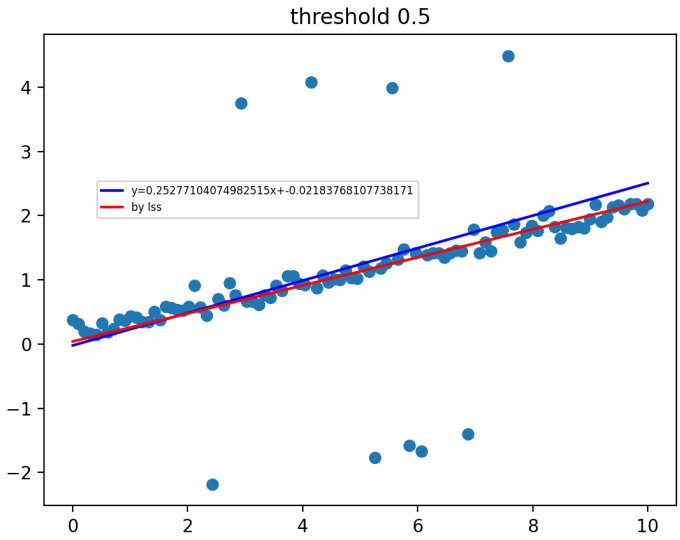
파일 첨부하였습니다.

Q3.3. Draw an estimated line with 100 observed points included outliers.



threshold 값을 조작하며 그래프를 그려본 결과, threshold 값이 작을수록 더 정확한 추정 값을 얻을 수 있다는 것을 알 수 있었다.

Q.3.4. Compare an estimated line using RANSAC framework with total least squares solution with 100 observed points included outliers.



빨간색 직선은 least square solution 으로 추정한 직선이고, 파란색 직선은 threshold 값을 0.5 로 설정한 후 RANSAC framework 를 통해 추정한 직선이다. 추정 결과 이 둘은 상당히 비슷한 양상을 띄는 것을 알 수 있는데, 이는 주어진 데이터가 outlier (노이즈) 의 값이 많거나 크지 않기 때문이다.