ECON526: Problem Set 3 (Solutions)

Jesse Perla

— INSTRUCTOR SOLUTIONS —

Instructions

- Ensure you modify the field above with your **name and student number above immediately**
- Modify directly and do not rename the file as Canvas will automatically append your name to it.
- Submit both the `.ipynb` and an exported `html` version of this file.
- See here for instructions.
- Do not change the filenames. Canvas automatically appends your name/student number to submitted files.

Setup

Feel free to use the following packages (and we have added a few convenience imports)

```
import numpy as np
import matplotlib.pyplot as plt
import scipy
from numpy.linalg import cond, matrix_rank, norm
from scipy.linalg import inv, solve, det, eig, lu, eigvals
from scipy.linalg import solve_triangular, eigvalsh, cholesky
# The rank in Question 4 depends on an almost-zero singular value, so
# other seeds may report rank 4 instead of 3
np.random.seed(4)
```

Question 1

Consider the unemployment dynamics from the lecture, where each year a fraction α of employed workers lose their job and a fraction ϕ of unemployed workers find one, with no births or deaths. Take $\alpha = 0.03$, $\phi = 0.15$, and $X_0 = \begin{bmatrix} E_0 \\ U_0 \end{bmatrix} = \begin{bmatrix} 900000 \\ 100000 \end{bmatrix}$.

$$\underbrace{\begin{bmatrix} E_{t+1} \\ U_{t+1} \end{bmatrix}}_{X_{t+1}} = \underbrace{\begin{bmatrix} 1-\alpha & \phi \\ \alpha & 1-\phi \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} E_t \\ U_t \end{bmatrix}}_{X_t}$$

Question 1.1

Using the `simulate` function from the lecture, simulate X_t for $T = 50$ periods and report X_T.

```
def simulate(A, X_0, T):
    X = np.zeros((len(X_0), T+1))
    X[:,0] = X_0
    for t in range(T):
        X[:,t+1] = A @ X[:,t]
    return X
alpha, phi = 0.03, 0.15
X_0 = np.array([900000, 100000])
T = 50
# modify here
# A = ...
# X = simulate(A, X_0, T)
```

SOLUTIONS

```
# SOLUTIONS
A = np.array([[1 - alpha, phi],
              [alpha, 1 - phi]])
X = simulate(A, X_0, T)
print(f"X_{T} = {X[:,T]}")
```

```
X_50 = [833336.60373699 166663.39626301]
```

Question 1.2

Calculate the eigenvalues and eigenvectors of A and check that one of the eigenvalues is 1. Rescale the corresponding eigenvector so that it sums to the total population $E_0 + U_0$ to get the steady state $\bar{X}$, and compare it to your X_T from Q1.1.

```
# modify here
```

SOLUTIONS

```
# SOLUTIONS
Lambda, Q = eig(A)
Lambda = np.real(Lambda)
print(f"eigenvalues = {Lambda}")
i = np.argmin(np.abs(Lambda - 1.0))
print(f"eigenvalue {i} = 1? {np.isclose(Lambda[i], 1.0)}")
X_bar = Q[:, i] / np.sum(Q[:, i]) * np.sum(X_0)
print(f"X_bar = {X_bar}")
print(f"X_{T} = {X[:,T]}")
print(f"||X_bar - X_T|| = {norm(X_bar - X[:,T])}")
```

```
eigenvalues = [1.   0.82]
eigenvalue 0 = 1? True
X_bar = [833333.33333333 166666.66666667]
X_50 = [833336.60373699 166663.39626301]
||X_bar - X_T|| = 4.625049209424981
```

SOLUTIONS

The eigenvalues are 1 and $1 - \alpha - \phi = 0.82$. The steady state is $\bar{X} \approx \begin{bmatrix} 833333 \\ 166667 \end{bmatrix}$, and X_{50} is within a few workers of it since $0.82^{50} \approx 5 \times 10^{-5}$. Which eigenvalue comes first depends on the solver, so full marks for either ordering as long as they pick the one equal to 1.

Question 1.3

Show that the steady-state unemployment rate $\bar{U}/(\bar{E}+\bar{U})$ is $\alpha/(\alpha+\phi)$, and verify it numerically using your $\bar{X}$ from Q1.2.

SOLUTIONS

In the steady state $\bar{X} = A\bar{X}$, so the second row gives $\bar{U} = \alpha\bar{E} + (1-\phi)\bar{U}$, which rearranges to $\phi\bar{U} = \alpha\bar{E}$. Then

$$\frac{\bar{U}}{\bar{E}+\bar{U}} = \frac{\alpha\bar{E}/\phi}{\bar{E}+\alpha\bar{E}/\phi} = \frac{\alpha}{\phi+\alpha}$$

With $\alpha = 0.03$ and $\phi = 0.15$ this is $1/6 \approx 0.1667$. Equivalently, the flows into and out of unemployment must balance: $\alpha\bar{E} = \phi\bar{U}$.

```
# modify here to verify numerically
```

SOLUTIONS

```
# SOLUTIONS
print(f"steady state unemployment rate = {X_bar[1] / np.sum(X_bar)}")
print(f"alpha / (alpha + phi) = {alpha / (alpha + phi)}")
```

```
steady state unemployment rate = 0.16666666666666669
alpha / (alpha + phi) = 0.16666666666666666
```

Question 2

Consider five web pages A, B, C, D, E with the adjacency matrix below, in the convention of the lecture: row i is the page you are on and column j is 1 if it links to page j, in the order A, B, C, D, E.

```
M = np.array([[0, 1, 1, 1, 0],
              [0, 0, 1, 0, 0],
              [1, 0, 0, 0, 0],
              [0, 0, 1, 0, 1],
              [1, 0, 0, 0, 0]])
```

Question 2.1

Count the incoming links of each page (i.e. the column sums of M). Then form the matrix S where each row of M is divided by the number of links on that page, and verify that every row of S sums to one.

```
# modify here
# S = ...
```

SOLUTIONS

```
# SOLUTIONS
print(f"incoming links = {M.sum(axis=0)}")
S = M / M.sum(axis=1, keepdims=True)
print(S)
print(f"row sums = {S.sum(axis=1)}")
```

```
incoming links = [2 1 3 1 1]
[[0.         0.33333333 0.33333333 0.33333333 0.        ]
 [0.         0.         1.         0.         0.        ]
 [1.         0.         0.         0.         0.        ]
 [0.         0.         0.5        0.         0.5       ]
 [1.         0.         0.         0.         0.        ]]
row sums = [1. 1. 1. 1. 1.]
```

Question 2.2

Start on page A for certain, i.e. $r_0 = \begin{bmatrix}1 & 0 & 0 & 0 & 0\end{bmatrix}^\top$, and iterate $r_{t+1} = S^\top r_t$ for $T = 50$ clicks. Report r_T. Hint: `simulate` from Q1.1 works for any square matrix.

```
# modify here
```

SOLUTIONS

```
# SOLUTIONS
r_0 = np.array([1.0, 0.0, 0.0, 0.0, 0.0])
r = simulate(S.T, r_0, T)
print(f"r_{T} = {np.round(r[:, T], 4)}")
```

```
r_50 = [0.375  0.125  0.3125 0.125  0.0625]
```

Question 2.3

Find the eigenvector of $S^\top$ with eigenvalue 1, rescale it to sum to one, and compare it to r_T from Q2.2. Which page has the highest rank? Is it the page with the most incoming links? Explain using the structure of the links.

```
# modify here
```

SOLUTIONS

```
# SOLUTIONS
Lambda, Q = eig(S.T)
i = np.argmin(np.abs(Lambda - 1.0))
r_bar = np.real(Q[:, i])
r_bar = r_bar / np.sum(r_bar)
print(f"|eigenvalues| = {np.round(np.abs(Lambda), 3)}")
print(f"r_bar = {np.round(r_bar, 4)}")
print(f"||r_bar - r_T|| = {norm(r_bar - r[:, T])}")
```

```
|eigenvalues| = [1.    0.816 0.816 0.    0.   ]
r_bar = [0.375  0.125  0.3125 0.125  0.0625]
||r_bar - r_T|| = 1.652391088832341e-05
```

SOLUTIONS

$\bar{r} = \begin{bmatrix} 0.375 & 0.125 & 0.3125 & 0.125 & 0.0625 \end{bmatrix}^\top$, which matches r_{50} to about four decimal places since the second largest $|\lambda| \approx 0.82$ and $0.82^{50} \approx 5 \times 10^{-5}$.
Page A ranks highest even though C has three incoming links and A only two. C and E link only to A, so they send all of their traffic there, while the links into C come from A and D, which split their traffic, and B, which sends all its traffic to C. A link from a page with few outgoing links counts for more than one from a page with many.

Question 3

Generate a design matrix $X \in \mathbb{R}^{200\times 3}$ with independent standard normal entries and simulate $y = X\beta + \epsilon$ for the β below with a small amount of noise.

```
N, K = 200, 3
X = np.random.randn(N, K)
beta = np.array([1.0, -2.0, 0.5])
y = X @ beta + 0.1 * np.random.randn(N)
```

Question 3.1

Estimate β with `scipy.linalg.lstsq` and compare it to the true β.

```
# modify here
```

SOLUTIONS

```
# SOLUTIONS
beta_hat, residuals, rank, s = scipy.linalg.lstsq(X, y)
print(f"beta_hat = {beta_hat}")
print(f"||beta_hat - beta|| = {norm(beta_hat - beta)}")
```

```
beta_hat = [ 0.99950396 -1.99187643  0.50740723]
||beta_hat - beta|| = 0.011004794519123336
```

Question 3.2

Estimate β again by forming the normal equations $(X^\top X)\beta = X^\top y$ and solving them with `solve(..., assume_a="pos")`. Compare to Q3.1.

```
# modify here
```

SOLUTIONS

```
# SOLUTIONS
beta_ne = solve(X.T @ X, X.T @ y, assume_a="pos")
print(f"beta_ne = {beta_ne}")
print(f"||beta_ne - beta_hat|| = {norm(beta_ne - beta_hat)}")
```

```
beta_ne = [ 0.99950396 -1.99187643  0.50740723]
||beta_ne - beta_hat|| = 1.9984014443252818e-15
```

Question 3.3

Explain why $X^\top X$ is positive definite for this X, and what that implies about the solution to the least squares problem.

SOLUTIONS

For any $v \neq 0$, $v^\top X^\top X v = \|Xv\|_2^2 \geq 0$, with equality only if $Xv = 0$. Since the columns of X are independent Gaussian draws they are linearly independent with probability one (full rank), so $Xv \neq 0$ and $X^\top X$ is positive definite.

The Hessian of $\|y - X\beta\|_2^2$ is $2X^\top X$, so the objective is strictly convex and the normal equations have a unique solution: the minimizer $\widehat{\beta}$ is identified. Numerically, positive definiteness is also what lets `solve` use a Cholesky factorization.

Question 4

Now append a fourth column to X which is a linear combination of the others.

```
X_col = np.column_stack([X, X[:, 0] + 2 * X[:, 1]])
print(X_col.shape)
```

```
(200, 4)
```

Question 4.1

Calculate the rank of `X_col` and the condition number of $X_{\text{col}}^\top X_{\text{col}}$. Compare to the same quantities for X.

```
# modify here
```

SOLUTIONS

```
# SOLUTIONS
print(f"rank(X) = {matrix_rank(X)}, rank(X_col) = {matrix_rank(X_col)}")
print(f"cond(X'X) = {cond(X.T @ X):.4g}")
print(f"cond(X_col'X_col) = {cond(X_col.T @ X_col):.4g}")
```

```
rank(X) = 3, rank(X_col) = 3
cond(X'X) = 1.432
cond(X_col'X_col) = 1.564e+16
```

Question 4.2

Try to solve the normal equations for `X_col` with `solve`, and then estimate with `lstsq`. Describe what happens in each case.

```
# modify here
```

SOLUTIONS

```
# SOLUTIONS
try:
    beta_ne_col = solve(X_col.T @ X_col, X_col.T @ y, assume_a="pos")
    print(f"solve returned {beta_ne_col}")
except np.linalg.LinAlgError as e:
    print(f"solve failed: {e}")
beta_col, residuals, rank, s = scipy.linalg.lstsq(X_col, y)
print(f"lstsq beta = {beta_col}, rank = {rank}")
print(f"singular values = {s}")
print(f"eigenvalues of X_col'X_col = {eigvalsh(X_col.T @ X_col)[::-1]}")
print(f"||X_col @ beta_col - y|| = {norm(X_col @ beta_col - y)}")
```

```
solve failed: A singular matrix detected: slice(s) [0] are singular.
lstsq beta = [ 1.49687877 -0.99712679  0.50740723 -0.49737482], rank = 3
singular values = [3.49591162e+01 1.46410029e+01 1.22892677e+01 2.62416006e-
15]
eigenvalues of X_col'X_col = [ 1.22213980e+03  2.14358966e+02  1.51026101e+02 -
1.28249028e-13]
||X_col @ beta_col - y|| = 1.3271167580153644
```

SOLUTIONS

$X_{\text{col}}^{\top} X_{\text{col}}$ is singular (rank 3 out of 4), so `solve` may raise an error or, with rounding, return a numerically unreliable coefficient vector. `lstsq` always returns a vector, with the residual as small as before, but it reports a rank of 3. The rank is the number of singular values it counts as nonzero, and the fourth is around 10^{-15}, pure rounding noise from the collinear column. The eigenvalues of $X_{\text{col}}^{\top} X_{\text{col}}$ are the squares of the singular values, so its smallest eigenvalue is also numerically zero, which is what makes `solve` fail. With other seeds `lstsq` sometimes keeps that singular value, reports rank 4, and returns coefficients on the order of 10^{12}, which is why the notebook fixes the seed.

Question 4.3

Explain why the individual parameters are not identified for `X_col`, and what the `lstsq` answer represents.

SOLUTIONS

With $x_4 = x_1 + 2x_2$, any β with $\beta_1 + \beta_4$, $\beta_2 + 2\beta_4$, and β_3 fixed gives the same fitted values $X_{\text{col}}\beta$, so a whole line of β minimizes the residual. The Hessian $2X_{\text{col}}^\top X_{\text{col}}$ is only positive semi-definite, with a zero eigenvalue in that direction, so the objective is flat along it and the first order conditions cannot pin down the parameters.

`lstsq` returns the minimum-norm solution: among all β that minimize $\|y - X_{\text{col}}\beta\|_2$, the one with the smallest $\|\beta\|_2$. It fits the data, but the coefficients on the collinear columns are not individually identified.