ECON526: Midterm 1 Practice

Jesse Perla

Ungraded practice problems for Midterm 1.

```
# Packages available. DO NOT MODIFY
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
```

Short Question 1

For a square matrix A which is $n \times n$:

1. What is a definition of the **condition number**?
2. What is the smallest it could be, and when?
3. Intuitively, what does a very large but non-infinite condition number mean? Where does it urge caution?

Answer:

(double click to edit your answer)

Short Question 2

Consider a 2×2 matrix A as

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

Give an a, b, c and d examples with:

1. The smallest possible condition number
2. A very large but finite condition number
3. A rank 1 matrix where the two columns are not identical.

Answer:

(double click to edit your answer)

Question 1

Given an eigendecomposition $A = Q\Lambda Q^{-1}$, the matrix square $A^2 = Q\Lambda^2 Q^{-1}$ where Λ^2 is the componentwise square of the eigenvalues, since $Q^{-1}Q = I$ cancels in the middle of $A \cdot A$.

Question 1.1

There is nothing special about taking the square. Just as you would define the square root of a number x as a $x^{1/2}$ such that $x^{1/2} \times x^{1/2} = x$, we can do the same thing with a matrix square root $A^{1/2}$ is such that $A^{1/2} \cdot A^{1/2} = A$.

Use the eigendecomposition to find $A^{1/2}$ for the A given below. Verify that $A^{1/2} \cdot A^{1/2} \approx A$.

```
# MODIFY CODE HERE
A = np.array([[2, 1], [1, 2]])
```

Question 1.2

Consider that both $(-2) \times (-2) = 2 \times 2 = 4$. There may be multiple square roots of a matrix, and many matrices do not have a square root. For example, the matrix $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ does not have a square root.

Take this matrix and mention a property of it that suggests why this matrix looks suspicious and does not have a unique square root or why the method might fail

```
# MODIFY CODE HERE, add comments and an explanation on what the problems here would be
A = np.array([[0, 1], [0, 0]])
```

Question 1.3

Argue that if a matrix is symmetric positive definite then its square root will also be symmetric, positive definite.

Answer:

(double click to edit your answer)

Question 2

Question 2.1

Take a random variable with the multivariate normal distribution

$$Y \sim N(\mu, \Sigma)$$

with $\mu \equiv \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $\Sigma \equiv \begin{bmatrix} 0.1 & 0.05 \\ 0.05 & 0.1 \end{bmatrix}$

Take 10,000 draws using this distribution. Hint: See `np.random.multivariate_normal` in the lecture notes.

```
# MODIFY CODE HERE
N = 10000
# mu = ...
# Sigma = ...
# Y_draws = ...
```

Question 2.2

Take the `Y_draws` above, calculate the mean and covariance matrix, and compare them to μ and Σ to see how close they are. Hint: use `np.cov(Y_draws, rowvar=False)` to get the covariance matrix. What is a natural way to compare how close matrices or vectors are?

```
# MODIFY CODE HERE
# make sure to state how you are comparing vectors/matrices
```

Answer:

(double click to add an explanation here)

Question 2.3

A key property of Gaussian random variables is that for any $Y \sim N(\mu, \Sigma)$ we can write it as a function of a unit multivariate normal $X \sim N(0, I)$ and a matrix C such that $Y = \mu + CX$. This is called the Cholesky decomposition of Σ.

1. Find the Cholesky of the covariance matrix Σ, C. You can use upper or lower Cholesky, but be consistent
2. Draw the same N number of draws from $X \sim N(0, I)$ as the previous parts of Q2
3. Transform those draws Y
4. Compare the mean and covariance of those transformed Y to μ and Σ to see how well the transformation did

```
# MODIFY CODE HERE
```

Question 2.4

The same transformation works with the matrix square root. That is, if $Y \sim N(\mu, \Sigma)$, then $Y = \mu + \Sigma^{1/2}X$ where $X \sim N(0, I)$ where $\Sigma^{1/2}$ is the matrix square root of Σ.

Repeat the Q2.3 using this transformation instead. That is,

1. Find the square root of the covariance matrix Σ, $\Sigma^{1/2}$. Hint: You can do this yourself as in Question 1, use the `scipy` function `scipy.linalg.sqrtm` or use the scipy function `scipy.linalg.matrix_power` with the argument `0.5`.
2. Draw the same N number of draws from $X \sim N(0, I)$ as the previous parts of Q2
3. Transform those draws to form Y samples
4. Compare the mean and covariance of those transformed Y to μ and Σ to see how well the transformation did

```
# MODIFY CODE HERE
```

Question 3

Consider the function

$$f(x_1, x_2) = 2x_1^2 - 2x_1x_2 + 3x_2^2 - 4x_1 + 1$$

Question 3.1

Write f in the canonical quadratic form $f(x) = \frac{1}{2}x^\top Ax + b^\top x + c$ for a symmetric matrix A, vector b, and scalar c. Give A, b, and c and then define them in code.

Hint: define $x \equiv \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top$. The diagonal entries of A are 4 and 6, and A is symmetric, so you only need to find a single off-diagonal term. The b and c are especially easy.

Answer:

(double click to edit your answer)

```
# MODIFY CODE HERE
# A = ...
# b = ...
# c = ...
```

Question 3.2

Use the eigenvalues of A to classify it as positive definite, negative definite, semi-definite, or indefinite. What does this say about the curvature of f?

```
# MODIFY CODE HERE
```

Answer:

(double click to edit your answer)

Question 3.3

Solve the first order condition $\nabla f(x^*) = Ax^* + b = 0$ for x^* using `solve`. Is x^* a minimum, a maximum, or a saddle point, and is it unique?

```
# MODIFY CODE HERE
```

Answer:

(double click to edit your answer)

Question 3.4

Change one coefficient of f so that A becomes indefinite. Give the new A, verify with its eigenvalues, and explain why the function no longer has a minimum or a maximum.

```
# MODIFY CODE HERE
```

Answer:

(double click to edit your answer)

Question 4

Here is code that generates a matrix and runs linear least squares

```
# DO NOT MODIFY
X = np.array([[1.0, 2.0],
              [0.5, 3.1],
              [1.2, 1.0]])
y = np.array([1.0, 2.0, 3.0])
beta_hat, residuals, rank, s = scipy.linalg.lstsq(X, y)
print(f"beta_hat = {beta_hat}")
```

```
beta_hat = [1.70960187 0.21077283]
```

Question 4.1

What optimization problem is the least squares problem solving? Also, what are the **normal equations** which provides an equivalent way to solve this problem in many cases?

Answer:

(double click to add an explanation here)

Question 4.2

Write code to solve the least squares problem using the normal equations from the previous part. Modify the code below which has a template for using solve. Compare your results to the `beta_hat` from above.

```
# MODIFY CODE HERE
# Below is a template for solve to replace with your own matrices
beta_hat_solve = solve(np.array([[1.0, 0.0],[0.0, 1.0]]), np.array([1.0, 1.0]))
print(f"beta_hat_solve = {beta_hat_solve}, beta_hat = {beta_hat}")
```

```
beta_hat_solve = [1. 1.], beta_hat = [1.70960187 0.21077283]
```

Answer:

(double click to add an explanation here)

Question 4.3

Modify the original X matrix to make it a 3x3 matrix but one that still have a rank of 2. Check the rank in the code to verify.

```
# MODIFY CODE HERE
X = np.array([[1.0, 0.0], [0.0, 1.0]])
```

Can you use the normal equations to solve for the new least squares problem using your modified X (and the old y)? Why or why not?

Answer:

(double click to add an explanation here)

Question 5

The design matrix below has a fourth column equal to $x_1 + 2x_2$, so it has rank 3 and the least squares solution is not unique. `lstsq` returns one of the minimizers.

```
# DO NOT MODIFY
N, M = 200, 3
X = np.random.randn(N, M)
beta = np.array([1.0, -2.0, 0.5])
y = X @ beta + 0.1 * np.random.randn(N)
X_col = np.column_stack([X, X[:, 0] + 2 * X[:, 1]])
beta_col, residuals, rank, s = scipy.linalg.lstsq(X_col, y)
print(f"rank = {rank}, lstsq beta = {beta_col}")
```

```
rank = 3, lstsq beta = [ 1.49928684 -0.99989547  0.51352305 -0.5005041 ]
```

Ridge regression replaces the normal equations with $(X^\top X + \lambda I)\beta = X^\top y$ for $\lambda > 0$.

Question 5.1

For `X_col`, solve the ridge normal equations for $\lambda \in \{10^{-6}, 0.1, 1, 10\}$ and print each β_λ.

```
# MODIFY CODE HERE
lambdas = [1e-6, 0.1, 1.0, 10.0]
```

Question 5.2

Plot or print $\|\beta_\lambda\|_2$ against λ. Explain why the ridge solution is unique for every $\lambda > 0$ even though the least squares solution is not, and how it relates to the `lstsq` answer above.

```
# MODIFY CODE HERE
```

Answer:

(double click to edit your answer)

Question 5.3

Does regularization solve the identification problem? Explain in a few sentences.

Answer:

(double click to edit your answer)

Question 6

Consider a joint probability distribution of two discrete random variables X and Y. Let $X \in \{0, 1\}$ and $Y \in \{0, 1, 2\}$.

Question 6.1

Something isn't quite right with the PMF below. Figure out what the issue is and fix it within the P matrix below, modifying the code directly. Make whatever change you wish, but make it clear what the issue was.

```
# MODIFY CODE HERE
P = np.array([[0.2, 0.3, 0.1],
              [0.3, 0.1, 0.1]])
```

Question 6.2

Using your new PMF of $\mathbb{P}(X, Y)$, calculate the marginal PMFs for X and Y, $\mathbb{P}(X)$ and $\mathbb{P}(Y)$. Hint: `np.array([[1.0, 2.0], [3.0,4.0]]).sum(axis=0)` and `axis=1` are useful

```
# MODIFY CODE HERE
```

Question 6.3

Define independence of two random variables, and write code to check if X and Y are independent with your PMF using the definition.

Write code to verify whether your X and Y are independent for your PMF.

```
# MODIFY CODE HERE
```

Answer:

(double click to add an explanation here)

Question 6.4

Using your PMF, calculate $\mathbb{E}[X \mid Y = 1]$. Hint: You do not need to calculate the conditional probability distribution for all Y to do this.

```
# MODIFY CODE HERE
```

Question 7

Take an exponential distribution with $\lambda = 0.5$, as in the lecture, so that $\mathbb{E}[X] = 2$ and $\mathrm{Var}[X] = 4$.

```
# DO NOT MODIFY
dist = scipy.stats.expon(scale=1/0.5)
mu, sigma = dist.mean(), dist.std()
print(f"mu = {mu}, sigma = {sigma}")
```

```
mu = 2.0, sigma = 2.0
```

Question 7.1

State the Law of Large Numbers, including its assumptions. Then, for $N \in \{10, 100, 1000, 10000\}$, draw N samples with `dist.rvs(size=N)`, calculate the sample mean $\bar{X}_N$, and print $|\bar{X}_N - \mu|$ for each N.

```
# MODIFY CODE HERE
```

Answer:

(double click to edit your answer)

Question 7.2

State the Central Limit Theorem, including its assumptions. Then, for $N = 100$, simulate 10000 independent sample means $\bar{X}_N$ (hint: `dist.rvs(size=(10000, N))` and `mean(axis=1)`), form $Y = \sqrt{N}(\bar{X}_N - \mu)$, and plot a histogram of Y with `density=True`. On the same axes plot the density the CLT predicts, using `scipy.stats.norm(0, sigma).pdf`.

```
# MODIFY CODE HERE
N = 100
```

Answer:

(double click to edit your answer)

Question 7.3

Repeat Q7.2 for a Pareto distribution with $\alpha = 1.5$, `scipy.stats.pareto(1.5)`, using its `mean()` for μ and $\sigma = 1$ for the overlaid density. Does the histogram look normal? Which theorem still applies to this distribution and which does not, and why?

```
# MODIFY CODE HERE
```

Answer:

(double click to edit your answer)