hw4

May 29, 2020

[1]:
```
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt
```

1 Problem 1

[7]:
```
# Problem 1
## Question b)
s = 1
sigma = 0.5
X = np.array([-1.41,-0.57,1.12,3.05])
Y = np.array([1.69,0.51,-1.08,-2.44])
## Posterior parameters estimation.
Sigma = ((1/sigma**2) * np.dot(X,X) + (1/s**2))**(-1)
M_pos = (1/sigma**2) * Sigma * np.dot(X,Y)

xnew = 2.18
### MonteCarlo to estimate mu_new
n = 100000
beta_simul = np.random.normal(loc=M_pos, scale= Sigma**(0.5), size=n)
MC_estimate = np.mean(xnew * beta_simul)
std_simul = np.std(beta_simul * xnew)
### CI
alpha = 0.05
q = scipy.stats.norm.ppf(1-alpha/2)
sup_bound = (q * std_simul) / np.sqrt(n) + MC_estimate
inf_bound = - (q * std_simul) / np.sqrt(n) + MC_estimate
print("MC estimate : {}".format(MC_estimate))
print("{}% CI : [{},{}]".format((1-alpha)*100, inf_bound, sup_bound))

## Question c)
n = 100000
error_simul = np.random.normal(loc=0, scale= 0.5, size= n)
Y_new_simul = error_simul + xnew * np.random.normal(loc=M_pos, scale= Sigma**(0.
↪5), size=n)
count = 0
for i in range(len(Y_new_simul)) :
```

```
    if Y_new_simul[i] < -2.1 :
        count += 1
print("Estimated P(Y_new < -2.1) = {}".format(count/n))
```

```
MC estimate : -1.882505185231479
95.0% CI : [-1.884369783570375,-1.8806405868925828]
Estimated P(Y_new < -2.1) = 0.35472
```

2 Problem 3

[3]:
```
def solveStationary( A ):
    """ x = xA where x is the answer
    x - xA = 0
    x( I - A ) = 0 and sum(x) = 1
    """
    n = A.shape[0]
    a = np.eye( n ) - A
    a = np.vstack( (a.T, np.ones( n )) )
    b = np.matrix( [0] * n + [ 1 ] ).T
    return np.linalg.lstsq( a, b )[0]


#Problem 3
P = np.array([[0,1/3,1/3,1/3,0],
              [1/3,0,1/3,0,1/3],
              [1/2,1/2,0,0,0],
              [1/2,0,0,0,1/2],
              [0,1/2,0,1/2,0]])
n = np.shape(P)[0]
## Question a)

print("Question a)")
print("Average time spent in A on the long run : {}".
 ↪format(solveStationary(P)[0]))
print("="*50)
## Question b)
print("Question b)")
print("Number of steps before returning to A : {}".format(1/
 ↪solveStationary(P)[0]))
print("="*50)
## Question d)
b = [0,-1,0,0,0]
P_mod = P.copy()
P_mod[:,0] = [0] * n
print("Question d)")
print("mu_c = {}".format(np.linalg.solve(P_mod.T - np.eye(n),b)[2]))
```

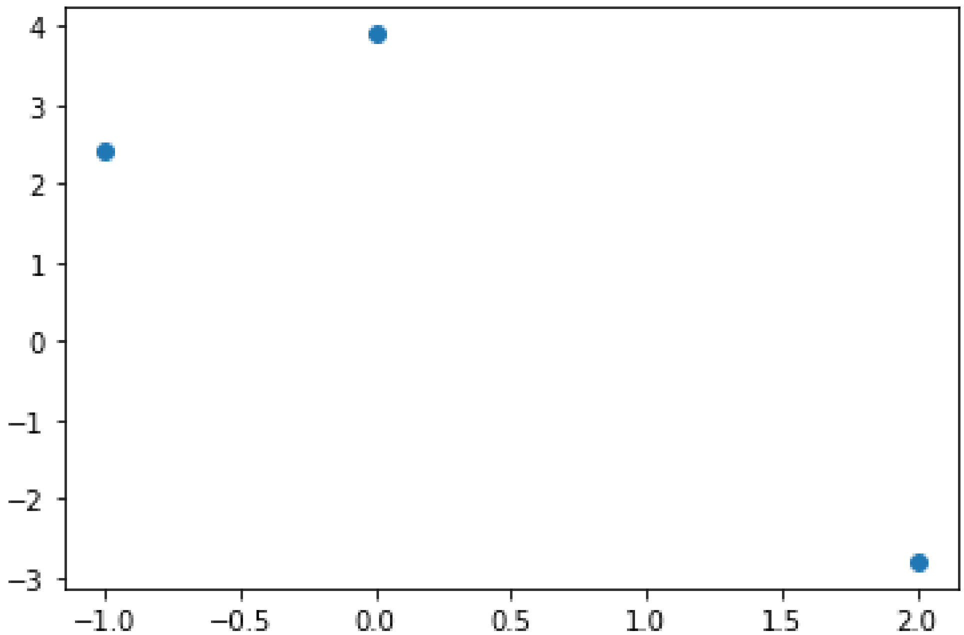
```python
print("="*50)

## Question e)
P_mod2 = np.array([[0,0,1/3],
                   [0,0,0.5],
                   [1/2,1/2,0]])
b = [-1/3,-1/2,0]

# np.linalg.solve(P_mod2 - np.eye(n-2),b)
print("Question e)")
print("p_B = {}".format(np.linalg.solve(P_mod2 - np.eye(n-2),b)[0]))
print("="*50)

## Question f)
b = [-1] * n
np.linalg.solve(P_mod-np.eye(n),b)
print("Question f)")
print("Expected number of steps starting from C before returning to A : {}".
 ↪format(np.linalg.solve(P_mod-np.eye(n),b)[2]))
print("="*50)
```

```
Question a)
Average time spent in A on the long run : [[0.25]]
==================================================
Question b)
Number of steps before returning to A : [[4.]]
==================================================
Question d)
mu_c = 0.5454545454545454
==================================================
Question e)
p_B = 0.5714285714285714
==================================================
Question f)
Expected number of steps starting from C before returning to A :
2.6363636363636362
==================================================

/Users/thibaudbruyelle/opt/anaconda3/envs/myenv/lib/python3.7/site-
packages/ipykernel_launcher.py:10: FutureWarning: `rcond` parameter will change
to the default of machine precision times ``max(M, N)`` where M and N are the
input matrix dimensions.
To use the future default and silence this warning we advise to pass
`rcond=None`, to keep using the old, explicitly pass `rcond=-1`.
  # Remove the CWD from sys.path while we load stuff.
```

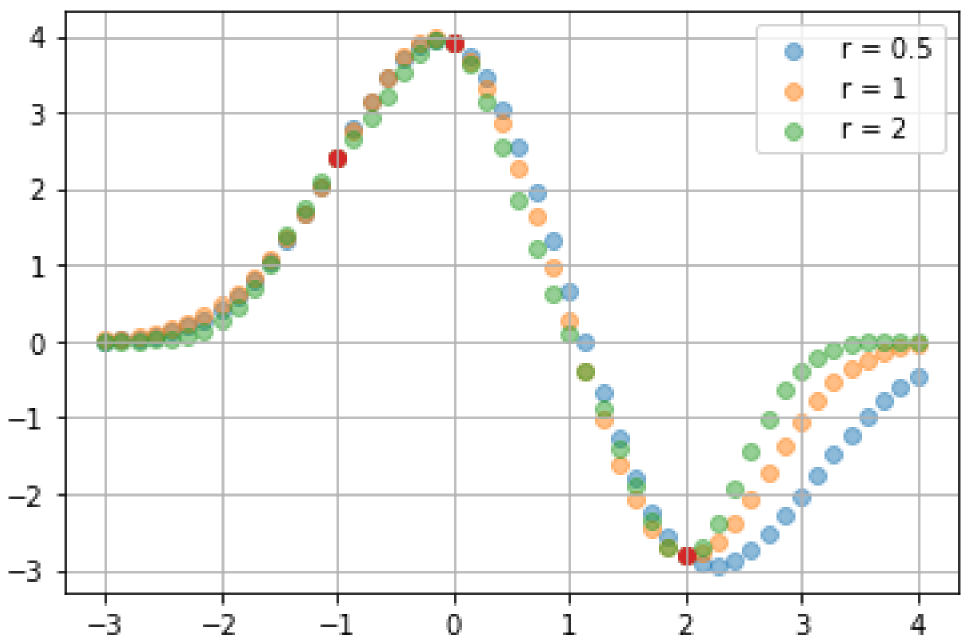
3 Problem 5

3.1 Question b)

[4]:
```python
## Problem 5)
def kernel_1(x,y,sigma,r = 0.5) :
    return (sigma**2) * np.exp(-r * (x-y)**2)

### Estimation of sigma.
# def estimate_sigma(Y) :
#     sigma_estimates = [i**2 for i in Y]
#     return np.mean(sigma_estimates)**0.5

## Question b)
X = np.array([-1,0,2])
Y = np.array([2.4,3.9,-2.8])
plt.scatter(X,Y)
plt.show()

# sigma = 1/2
# print(estimate_sigma(Y))

def mu_new(x_new, r=0.5) :
    K33 = np.zeros((3, 3))
    for i in range(3):
        for j in range(3):
            X_i = X[i]
            X_j = X[j]
            K33[i, j] = kernel_1(X_i, X_j, sigma,r)
    K_3new = np.array([kernel_1(x_new,x,sigma,r) for x in X])
    return np.matmul(K_3new, np.matmul(np.linalg.inv(K33), Y))

r_val = [0.5,1,2]
for r in r_val :
    X_new = np.linspace(-3,4,50)
    Y_new_expected_values = np.array([mu_new(x_new,r) for x_new in X_new])
    print("mu_new(1) = {}".format(mu_new(1,r)))
    plt.scatter(X_new, Y_new_expected_values, label = "r = {}".format(r), alpha␣
 ↪= 0.5)

plt.scatter(X,Y)
plt.legend()
plt.grid()
plt.show()
```



```
mu_new(1) = 0.6516208944611015
mu_new(1) = 0.27193391171686354
mu_new(1) = 0.11454252616787614
```



3.2 Question c)

We can observe that when r varies, the slope of the function changes around the point 0, -1 and 2. In fact, when r increases, it reduces the variance between f(x) and f(y) so this is why we can see on the plot that the expected values tend to be higher (for r = 2) (because the expected value is proportional to the inverse of the covariance matrix).

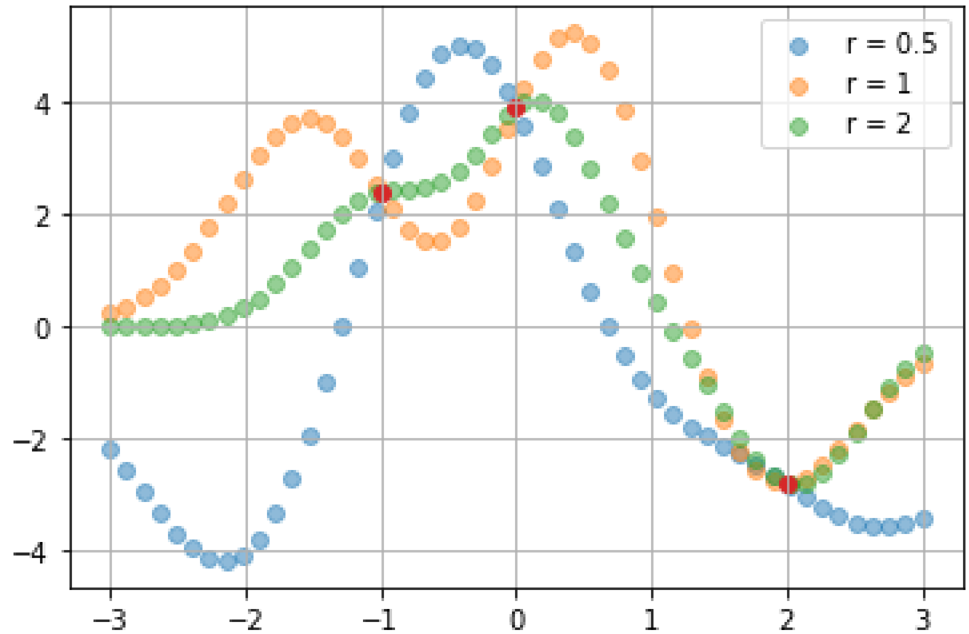
4 Question d)

[6]:

```
## Question d)
def covariance2(r, x, y):
    return -2 * r * (x - y) * np.exp(-r * (x - y) ** 2)


def covariance3(r, x, y):
    return 2 * r * np.exp(-r * (x - y) ** 2) * (1 - 2 * r * (x - y) ** 2)


Y = np.array([2.4, 3.9, -2.8, 6.4, -2.0, -0.8])


def mu_new_2(xnew, r=0.5):
    K_6new = [kernel_1(i, xnew, sigma, r) for i in X] \
             + [covariance2(r, i, xnew) for i in X]

    K66 = np.zeros((6, 6))
    for i in range(0, 6):
        if i < 3:
            tab = [kernel_1(j, X[i], sigma, r) for j in X] \
                  + [covariance2(r, j, X[i]) for j in X]
        if i >= 3:
            tab = [covariance2(r, X[i - 3], j) for j in X] \
              + [covariance3(r, X[i - 3], j) for j in X]
        K66[i] = tab
    # print(K66)
    return np.matmul(K_6new, np.matmul(np.linalg.inv(K66), Y))

# Plots
r_val = [0.5, 1, 2]
for r in r_val:
    X_new = np.linspace(-3, 3, 50)
    Y_new_expected_values = np.array([mu_new_2(x_new, r) for x_new in X_new])
    print("mu_new_2(1) = {}".format(mu_new_2(1, r)))
    plt.scatter(X_new, Y_new_expected_values, label="r = {}".format(r), alpha=0.
 ↪5)

plt.scatter(X, Y[0:3], alpha= 1)
```

```
plt.legend()
plt.grid()
plt.show()
```

```
mu_new_2(1) = -1.1897778071888667
mu_new_2(1) = 2.317844848622231
mu_new_2(1) = 0.6051814078535704
```



[ ]:

```

```

Homework 4 - Thibaud Bruyelle

Problem 1

a) $(\beta_0, \dots, \beta_d) \sim \mathcal{N}(0, s^2 I)$ which means that $(\beta_i)_{0 \le i \le d}$ are iid as $(\beta_0, \dots, \beta_d)$ is Gaussian.

$$\underbrace{f(\beta \mid data)}_{\text{posterior}} \propto \underbrace{f(data \mid \beta)}_{\text{likelihood}} \times \underbrace{f(\beta)}_{\text{prior}}$$

We have:

$$f(\beta) = \frac{1}{(2\pi)^{\frac{d+1}{2}} s^{(d+1)}} \times \exp\left[-\frac{1}{2}\beta^T\left(\frac{1}{s^2} I\right)\beta\right]$$

as $\det(s^2 I) = s^{2(d+1)}$.

Given β and the data:

$$Y_i \sim \mathcal{N}\left(\beta_0 + \sum_{j=1} \beta_j x_{ij}, \sigma^2\right)$$

and since $(\varepsilon_i)_i$ are iid,

the $(Y_i)_i$ are iid, so finally

$$f(\text{data} \mid \beta) =$$

$$\frac{1}{(2\pi)^{\frac{n}{2}}\sigma^n} \cdot \exp\left[-\frac{1}{2}(Y-\mu)^T\left(\frac{1}{\sigma^2}I\right)(Y-\mu)\right]$$

where $\mu = \left[\beta_0 + \sum_{j=1}^{d}\beta_j x_{1j}, \ldots, \beta_0 + \sum_{j=1}^{d}\beta_j x_{nj}\right]^T$

Finally:

$$f(\beta \mid \text{data}) \propto e^{-\frac{1}{2}\left[\underbrace{(Y-\mu)^T\left(\frac{1}{\sigma^2}I\right)(Y-\mu) + \beta^T\frac{1}{s^2}I\beta}_{(1)}\right]}$$

Then we also have:

$\mu = [\beta^T X_1, \ldots, \beta^T X_n]$, so

$$(Y-\mu)^T\left(\frac{1}{\sigma^2}I\right)(Y-\mu) = \begin{bmatrix} Y_1 - \beta^T X_1 \end{bmatrix}$$

$$\left[\frac{Y_1-\beta^T x_1}{\sigma^2}, \dots, \frac{Y_n-\beta^T x_n}{\sigma^2}\right]\left[\begin{array}{c} \vdots \end{array}\right]$$

$$= \frac{1}{\sigma^2}\left[Y^TY - 2Y^TX\beta + \beta^TX^TX\beta\right]$$

where $Y = [Y_1, \dots, Y_n]^T$

$X = (X_{ij})_{\substack{0 \leq i \leq n-1 \\ 0 \leq j \leq d}}$, $\beta = [\beta_0, \dots, \beta_d]^T$

Then we can define $\boxed{\Sigma'^{-1} := \frac{1}{\sigma^2}X^TX + \frac{1}{s^2}I}$

We can rewrite (1) as

$$(\beta - M)^T \Sigma'^{-1} (\beta - M)$$

where $\boxed{M := \frac{1}{\sigma^2}\Sigma'^{-1T}X^TY}$

Finally: $\boxed{\beta \mid \text{data} \sim \mathcal{N}_d(M, \Sigma')}$

b) $\mu_{new} := \mathbb{E}[Y_{new} | \beta] = \beta x_{new}$.

$$= \beta_0 x_{new,0} + \beta_1 x_{new,1}.$$
$$= \beta_1 x_{new,1}.$$

Then:

$$\mathbb{E}(\mu_{new} | data) = x_{new,1} \cdot \mathbb{E}(\beta_1 | data).$$

We can estimate $\mathbb{E}(\beta_1 | data)$ by using a Monte-Carlo technique.

We finally get:

$$\boxed{\widehat{\mathbb{E}(\mu_{new} | data)} = -0,8623}$$

with a 95% "credibility interval":

$$[-0.8810, -0,8438]$$

(Please see the code part).

c) $Y_{new} = \beta x_{new} + \varepsilon$ so we can use a Monte-Carlo technique to estimate :

$$\mathbb{E}\left(\mathbb{1}\{ \beta x_{new} + \varepsilon < -2{,}1 \} \mid data \right)$$

$$= \mathbb{P}\left(Y_{new} < -2{,}1 \mid data \right).$$

Please see the code part.

d) Now the prior distribution of β is given by :

$$f_{(\beta_0, \dots, \beta_d)}(z_0, \dots, z_d) = \prod_{j=0}^{d} e^{-c|z_j|},$$

So:

$$f(\beta \mid data) \propto e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i - \beta^T X_i)^2} \, e^{-c\sum_{j=0}^{d}|\beta_j|}$$

In order to compute the mode of this posterior distribution, we need to solve :

$$\min_{\beta} \sum_{i=1}^{m} (Y_i - \beta^T X_i)^2 + c \sum_{j=0}^{d} |\beta_j|$$

penalty function

which enables shrinkage of the parameter.

Problem 2

a) We have $\Phi^*(\theta, x) = \mathbb{E}_x(\exp(\theta T))$

$$= \sum_{y \in S} \mathbb{E}_x(\exp(\theta T) \mid X_1 = y)\, P(x,y)$$

$$\sum_{y \in C} \mathbb{E}_x(\exp(\theta T) \mid X_1 = y)\, P(x,y)$$

$$+ \sum_{y \in C^c} \mathbb{E}_x(\exp(\theta T) \mid X_1 = y)\, P(x,y)$$

$$= \exp(\theta) \sum_{y \in C^c} P(x,y) + \exp\theta \sum_{y \in C} \mathbb{E}_y(e^{\theta(T+1)}) P(x,y)$$

$$= \exp\theta \sum_{y \in C^c} P(x,y) + \exp\theta \sum_{y \in C} \Phi^*(\theta, y) P(x,y).$$

b) Let us write B the restriction of the transition matrix to $\mathcal{E} \to \mathcal{E}$.

Here
$$B = \frac{1}{2}\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

And Δ the restriction to $\mathcal{E} \to \mathcal{E}^C$.

$$\Delta = [0, 0, 1/2]^T.$$

We want to solve:

$$(I - e^{\theta} B)\, \Phi^{*}(\theta) = \Delta$$

where $\Phi^{*}(\theta) = (\Phi^{*}(\theta, x))_{x \in S}$.

After calculations, $(I - e^{\theta} B)$ is invertible if and only if $\theta \neq \ln(4,49)$ and $\theta \neq \ln(1,1)$

Finally: $\boxed{\Phi^{*}(\theta) = (I - \exp\theta\, B)^{-1} \Delta}$.

c) If we have $(I - \exp\theta B)^{-1}$ with negative entries, it implies that

$$\sum_{0}^{+\infty} (\exp\theta B)^{n} = +\infty$$ and consequently

$\Phi^{k}(\theta)$ takes infinite values.

Problem 3

$$P = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 \\ 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 1/2 & 0 & 1/2 & 0 \end{bmatrix}$$

a) This Markov chain is irreducible

as it has a unique stationary

so M has one unique stationary distribution. In the long run the proportion of time spent in A is given by $\pi(A)$ where π is the stationary distribution for this chain.

b) See the notebook.

c) Let us write μ_X the expected number of visits to state Y before reaching state Z starting from state X, all states in $\{A, B, C, D, E\}$.
We have the linear system:

$$
\begin{cases}
\mu_A = \mathbb{1}_{\{Y=A\}} + \sum_{s \in \{A,\dots,E\}} \left(1 - \mathbb{1}_{\{Z=s\}}\right) P(A,s)\, \mu_s \\
\vdots \\
\mu_E = \mathbb{1}_{\{Y=E\}} + \sum_{s \in \{A,\dots,E\}} \left(1 - \mathbb{1}_{\{Z=s\}}\right) P(E,s)\, \mu_s
\end{cases}
$$

if the state is visited

if s s/t $t=1$ is visited, no need to count it.

d) We want to solve the following system:

$$\begin{cases} \mu_A = \frac{1}{3}\mu_B + \frac{1}{3}\mu_C + \frac{1}{3}\mu_D \\ \mu_B = 1 + 0 \cdot \frac{\mu_A}{3} + \frac{1}{3}\mu_C + \frac{1}{3}\mu_E \\ \mu_C = \frac{1}{2}\mu_B + 0 \\ \mu_D = \frac{1}{2}\mu_E + 0 \\ \mu_E = \frac{1}{2}\mu_B + \frac{1}{2}\mu_D . \end{cases}$$

Numerically, we want to find μ such that:

$(\tilde{P} - I)\mu = b$ where

$\mu = [\mu_A \; \dots \; \mu_E]^T$

$p = [p_A, \dots, p_E]$,

$b = [0, -1, 0, \dots, 0]^T$ and

$$\tilde{P} = \begin{cases} P_{ij} & \text{if } j \neq A \\ 0 & \text{if } j = A \end{cases}$$

See the notebook for the results.

e) Let us write p_i the probability that the walker reaches A before he reaches B, starting from vertex i. We have the following linear system:

$$\begin{cases} p_A = \frac{1}{3} p_B + \frac{1}{3} p_C + \frac{1}{3} p_D \\ p_B = \frac{1}{3} p_A + \frac{1}{3} p_C + \frac{1}{3} p_E \\ p_C = \frac{1}{2} p_A + \frac{1}{2} p_D \end{cases}$$

$$
\begin{cases}
p_D = \frac{1}{2} p_A + \frac{1}{2} p_E \\
p_E = \frac{1}{2} p_D + \frac{1}{2} p_B
\end{cases}
$$

with $p_A = 1$ and $p_C = 0$.

See the notebook for the results.

f) Let us write μ_i the expected number of steps until the walker reaches A when he starts from vertex i. We have the linear system:

$$
\begin{cases}
\mu_A = \frac{1}{3} \mu_B + \frac{1}{3} \mu_C + \frac{1}{3} \mu_D + 1 \\
\mu_B = \frac{1}{3} \mu_C + \frac{1}{3} \mu_E + 1 \\
\mu_C = \frac{1}{2} \mu_B + 1 \\
\mu_D = \frac{1}{2} \mu_E + 1
\end{cases}
$$

$(\mu_E = \frac{1}{2}\mu_D + \frac{1}{2}\mu_B + \dots$

See the notebooks for the results.

Problem 5

a) Let $x, y \in \mathbb{R}$.

- $\mathbb{E}\left[(Z(x), Z'(x)) \right] = (\mathbb{E}(Z(x)), \mathbb{E}(Z'(x)))$
$= (0, 0)$.

- $\text{cov}\left[(Z(x), Z'(x)), (Z(y), Z'(y)) \right]$

$$= \mathbb{E}\left[(Z(x), Z'(x))^T (Z(y), Z'(y)) \right]$$

$$= \mathbb{E}\begin{bmatrix} Z(x)Z(y) & Z(x)Z'(y) \\ Z'(x)Z(y) & Z'(x)Z'(y) \end{bmatrix}$$

We have: $\bullet\ \boxed{\mathbb{E}(Z(x)Z(y)) = \sigma^2 e^{-c(x-y)^2}}$

- $\boxed{\mathbb{E}\left(Z(x) Z'(y) \right) = -2r\sigma^2 (x-y) e^{-r(x-y)^2}}$

- $\boxed{\mathbb{E}\left(Z'(x) Z(y) \right) = -2r\sigma^2 (x-y) e^{-r(x-y)^2}}$

- $\boxed{\mathbb{E}\left(Z'(x) Z'(y) \right) = 2\sigma^2 r \left(1 - 2r(x-y)^2\right) e^{-r(x-y)^2}}$

because

$$\mathbb{E}\left(Z'(x) Z'(y) \right) = \frac{\partial^2}{\partial x \partial y} \operatorname{cov}\left(Z(x), Z(y) \right)$$

since $\mathbb{E}\left(Z(x) \right) = 0, \forall x \in \mathbb{R}$.

b) We have $X = [x_1, x_2, x_3]^T$
and $Y = [y_1, y_2, y_3]^T$.

If we write $f^* := f(x^*)$ where

x^* is the new value for x and model f as Z, we have:

$$\begin{bmatrix} Y \\ f^* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_{33} & K_{3*} \\ K_{*3} & K_{**} \end{bmatrix}\right)$$

where $K_{33} = \left(\mathrm{cov}(X_i, X_j)\right)_{1 \leq i,j \leq 3}$

$$K_{*3} = K_{3*}^T = \left(\mathrm{cov}(X^*, X_i)\right)_{1 \leq i \leq 3}$$

and $K_{**} = \mathrm{cov}(X^*, X^*)$

Then (from the lectures notes):

$$Y^* \mid Y, X, X^* \sim \mathcal{N}(\mu^*, \Sigma^*)$$

where: $\left\{ \begin{array}{l} \mu^* = K_{*3} K_{33}^{-1} Y \\ \\ \Sigma^* = K_{**} - K_{*3} K_{33}^{-1} K_{3*} \end{array} \right.$

See the notebook for the numerical results.

c) See the notebook.

d) In this case, we have the Gaussian vector:

$$\begin{bmatrix} Y \\ \underset{\downarrow}{f_*} \end{bmatrix} = \Big[\underbrace{f(X_1), \dots, f(X_3), f'(X_1), \dots}_{Z_2^T} f'(X_3), \underbrace{f_*}_{Z_1^T}\Big]^T$$

new value to predict

$$\text{Here } K_{ff} = \begin{bmatrix} \operatorname{cov}(f(X_1), f(X_1)) & \cdots & \operatorname{cov}[f(X_1), f'(X_3)] \\ & & \vert \\ \operatorname{cov}(f'(X_3)\, f(X_1)] & \text{—} & \operatorname{cov}[f'(X_3), f'(X_3)] \end{bmatrix}$$

Then we still have the same formula for μ^*. See the notebook for the results.

e) We can apply the same method by taking the mean of the observations

Problem 4

a) If $|S| < +\infty$, since X is irreductible, X admits one unique stationary distribution π.
Besides, $\pi = \left(\frac{1}{|S|}\right)_{1 \leq i \leq |S|}$ statisfies

$\pi P = \pi$ so this probability distribution is the stationary distribution of X. Consequently, we have by definition:

$$\boxed{\forall y \in S, \quad P^n(x,y) \xrightarrow[n \to +\infty]{} \pi(y) = \frac{1}{|S|} .}$$

b)

b) Let us define the transition matrix P such that $P(i,j)$ is the probability that element with index i ends up being the j^{th} card in the deck after one shuffle.

For $i \in \{2, \dots, 51\}$, $P(i,i) = \frac{i-1}{52}$ and $P(i, i+1) = \frac{52-i}{52}$.

and $P(i,1) = \frac{1}{52}$.

We also have
$$\begin{cases} P(1,1) = \frac{1}{52} \\ P(1,2) = \frac{51}{52} \end{cases}$$

and
$$\begin{cases} P(52,52) = \frac{51}{52} \\ P(52,1) = \frac{1}{52} \end{cases}$$

So finally we can show that

this markov chain is <u>aperiodic</u>, <u>doubly</u>

stochastic and aperiodic.

Eventually, the algorithm will shuffle the deck because every card can take any position with the same probability on the long run.