Amir Hossein Eyvazkhani

1747696

Ex6

Task I)

① $P(h=1|v) = \frac{\alpha_h + N_h}{\alpha_h + \alpha_t + N_h + N_t} = \frac{1+4}{1+1+4+3} = \frac{5}{9}$

② $P(b=1|v) = \frac{\alpha_h + N_h}{\alpha_h + \alpha_t + N_h + N_t} = \frac{1+4}{1+1+4+3} = \frac{5}{9}$

③ $P(c=1|F=1, b=1, v) = \frac{\alpha_h + N_h}{\alpha_h + \alpha_T + N_h + N_t} = \frac{1+2}{1+1+2+0} = \frac{3}{4}$

④ $P(c=1|F=1, b=0, v) = \frac{\alpha_h + N_h}{\alpha_h + \alpha_T + N_h + N_T} = \frac{1+1}{1+1+1+1} = \frac{2}{4}$

⑤ $P(c=1|F=0, b=1, v) = \frac{\alpha_h + N_h}{\alpha_h + \alpha_T + N_h + N_T} = \frac{1+1}{1+1+1+1} = \frac{2}{4}$

⑥ $P(c=1|F=0, b=0, v) = \frac{\alpha_h + N_h}{\alpha_h + \alpha_T + N_h + N_t} = \frac{1+0}{1+1+0+1} = \frac{1}{3}$

Task II)

ⓐ Loss function for probabilistic regression is the likelihood which we try to maximize as:

$$\theta^* = \arg\max_\theta P(y|X, \theta) = \arg\max_\theta \prod_{n=1}^{N} P(y_n|x_n, \theta)$$

then we apply log: which is strictly monotone and multiply by -1

$$\theta^* = \arg\min_\theta -\log P(y|X, \theta) \quad \text{(I)}$$

And in our case:

$$P(y|X, \theta) = N(y|x^T\theta, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y - x^T\theta)^2}{2\sigma^2}} \quad \text{(II)}$$

$$\text{I}, \text{II} \Longrightarrow L(\theta) = \sum_{n=1}^{N} \log p(y_n | x_n, \theta)$$

$$= \sum_{n=1}^{N} \log\left[\frac{1}{\delta^2 \sqrt{2\pi}} e^{\frac{-(y_n - x_n^T\theta)^2}{2\delta^2}} \right]$$

$$= \sum_{n=1}^{N} \log\left(\frac{1}{\delta^2 \sqrt{2\pi}} \right) + \sum_{n=1}^{N} \log\left[e^{\frac{-(y_n - x_n^T\theta)^2}{2\delta^2}} \right]$$

$$= \underbrace{-N \log\left(\delta^2 \sqrt{2\pi}\right)}_{\text{constant}} - \underbrace{\underbrace{\frac{1}{2\delta^2}}_{\text{Constant}} \sum_{i=1}^{N} (y_i - x_n^T\theta)^2}_{\text{MSE}}$$

Hence it is the Same as minimizing the MSE.

(b) $P(\theta) = N(\theta | 0, \delta^2 I)$, $p(y | X, \theta) = N(y | x^T\theta, \sigma^2)$

$$p(\theta | y, X) = \frac{1}{Z} P(y | X, \theta) P(\theta)$$

by bayes rule we have

$$\Rightarrow \theta^* = \underset{\theta}{\operatorname{argmax}}\ N(y | x^T\theta, \sigma^2)\ N(\theta | 0, \delta^2 I)$$

we apply log and minus and the likelihood function will be

$$\Rightarrow L(\theta) = \sum_{n=1}^{N} \log\left[\frac{1}{\delta^2\sqrt{2\pi}} e^{\frac{-(y_n - x_n^T\theta)^2}{2\delta^2}} \right] + \sum_{n=1}^{D} \log\left[\frac{1}{\delta^2\sqrt{2\pi}} e^{\frac{-\theta_n^2}{2\delta^2}} \right]$$

$$= -N \log \delta^2\sqrt{2\pi} - \frac{1}{2\delta^2} \sum_{n=1}^{N} (y_n - x_n^T\theta)^2 - D\log(\delta^2\sqrt{2\pi}) - \frac{1}{2\delta^2} \sum_{n=1}^{D} \theta_n^2$$

$$= \underbrace{-\sum_{n=1}^{N} (y_i - x_i^T\theta)^2}_{\text{MSE}} - \underbrace{\frac{1}{\delta^2} \|\theta\|_2^2}_{\substack{\text{Regularization} \\ \text{term with} \\ \lambda = \frac{1}{\delta^2}}} - \underbrace{D\log(\delta^2\sqrt{2\pi}) - N\log \delta^2\sqrt{2\pi}}_{\text{Constant}}$$

Hence it is equivalent to minimizing with MSE and regularization term of type L2.

(c) $P(\theta) = \prod_{m=1}^{M} p(\theta_m) = \prod_{m=1}^{M} \frac{1}{2b} e^{\frac{-|\theta_m|}{b}}$ $(\mu = 0)$

Like the previous Question, we have:

$$p(\theta | x, y) = \frac{1}{Z} p(y | x, \theta)\, p(\theta) = \frac{1}{Z} N(y | \theta^T x, \delta^2)\, L(\theta | 0, b)$$

$$\Rightarrow L(\theta) = \sum_{i=1}^{N} \log \frac{1}{\delta^2 \sqrt{2\pi}} e^{\frac{-(y_i - x_i \theta)^2}{2\delta^2}} + \sum_{j=1}^{m} \log \frac{1}{2b} e^{\frac{-|\theta_m|}{b}}$$

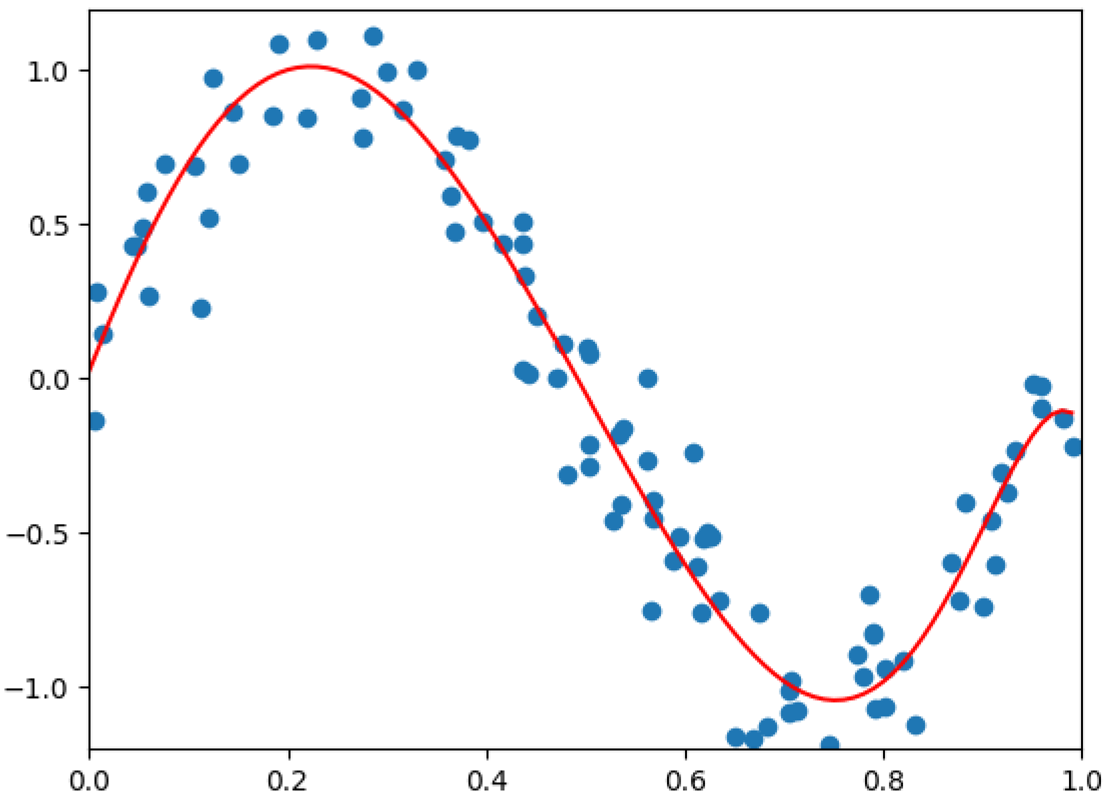
$$= \underbrace{\frac{-1}{2\delta^2} \sum_{i=1}^{N} (y_i - \theta x_i)^2}_{\text{MSE}} \underbrace{- \frac{1}{b} \sum_{j=1}^{m} ||\theta_j||}_{\text{regularization}} \underbrace{- N \log \delta^2 \sqrt{\pi} - \log 2b}_{\text{constant}}$$

Hence it's like minimizing with MSE with L1 regularization with $\lambda = \frac{2\delta^2}{b}$

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt


# Input: number of samples N
# Output: one-dimensional data set of N points where y_n = sin(2 pi x_n) + epsilon_n as in lecture
def sine_data_set(N):
  np.random.seed(1234)
  x = np.random.uniform(0,1,(N))
  y = np.sin(x*2*np.pi)+np.random.normal(scale=0.2,size=(N))
  return x,y


# Input: instances X as N x M matrix, model theta
# Output: predictions of model theta on X
def predict_regression(X,theta):
  return X @ theta


# Input: one-dimensional inputs x as vector of length N, polynomial degree d
# Output: polynomial feature representation of the inputs as N x (d+1) matrix
def poly_features(x,d):
  X = np.zeros((x.shape[0],d+1))
  for i in range(0,d+1):
    X[:,i] = np.power(x,i)
  return X
```

In [ ]:
```
# Generate and plot toy data set
N = 100 #3,10,20,100
x,y = sine_data_set(N)
plt.xlim([0,1])
plt.ylim([-1.2,1.2])
plt.scatter(x,y)

# Polynomial feature representation of degree d
d = 10
X = poly_features(x,d)


sigma = 0.1
```

```python
sigma_p = 100
M = d + 1

# Compute MAP model
I = np.eye(M)
sigma_sqr = sigma ** -2
sigma_p_sqr = sigma_p ** -2
theta_MAP = sigma_sqr * np.linalg.inv((sigma_sqr * X.T @ X) + (sigma_p_sqr * I)) @ X.T @ y
# Plot MAP model
x_axis = np.arange(0, 1, 0.01)
x_values = poly_features(x_axis, d)
predictions = predict_regression(x_values, theta_MAP)
plt.plot(x_axis, predictions, label="predictions", c='r')
plt.show()
```



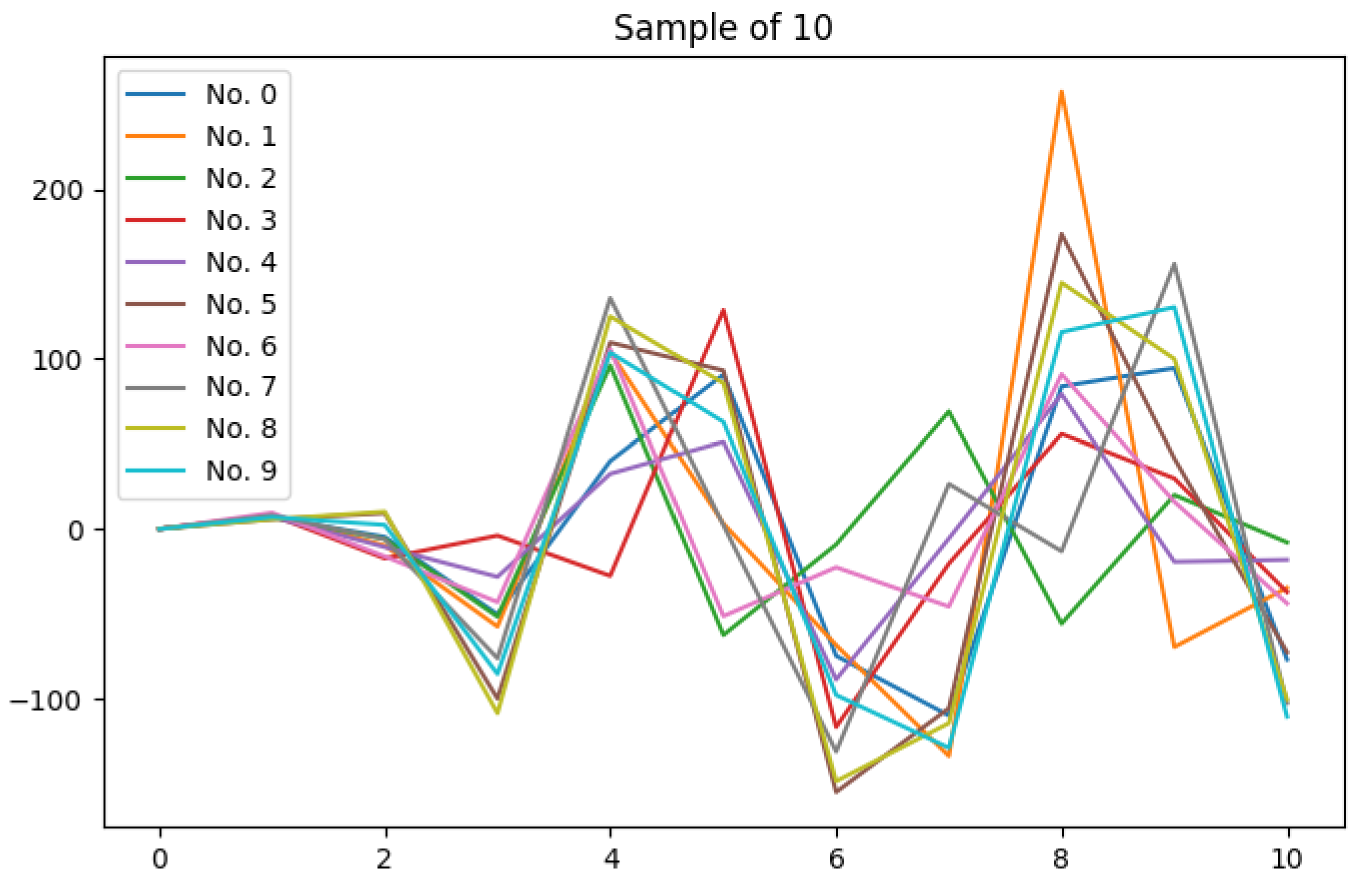
In [ ]:
```python
# Compute posterior distribution
I = np.eye(M)
```

```python
A = (sigma_sqr * X.T @ X) + (sigma_p_sqr * I)
A_inv = np.linalg.inv(A)
theta_bar = sigma_sqr * A_inv @ X.T @ y
print(f'A is {A} - and theta_bar is {theta_bar}')
```

```
A is [[10000.0001      5183.62907879  3452.73380164  2545.0681362
   1995.32757289  1632.80140724  1378.61667748  1191.66060306
   1048.75864724   936.05510082   844.8568212 ]
 [ 5183.62907879  3452.73390164  2545.0681362   1995.32757289
   1632.80140724  1378.61667748  1191.66060306  1048.75864724
    936.05510082   844.8568212    769.48071413]
 [ 3452.73380164  2545.0681362   1995.32767289  1632.80140724
   1378.61667748  1191.66060306  1048.75864724   936.05510082
    844.8568212    769.48071413   706.07603138]
 [ 2545.0681362   1995.32757289  1632.80140724  1378.61677748
   1191.66060306  1048.75864724   936.05510082   844.8568212
    769.48071413   706.07603138   651.95015955]
 [ 1995.32757289  1632.80140724  1378.61667748  1191.66060306
   1048.75874724   936.05510082   844.8568212    769.48071413
    706.07603138   651.95015955   605.16728762]
 [ 1632.80140724  1378.61667748  1191.66060306  1048.75864724
    936.05510082   844.8569212    769.48071413   706.07603138
    651.95015955   605.16728762   564.30111942]
 [ 1378.61667748  1191.66060306  1048.75864724   936.05510082
    844.8568212    769.48071413   706.07613138   651.95015955
    605.16728762   564.30111942   528.27770302]
 [ 1191.66060306  1048.75864724   936.05510082   844.8568212
    769.48071413   706.07603138   651.95015955   605.16738762
    564.30111942   528.27770302   496.27268659]
 [ 1048.75864724   936.05510082   844.8568212    769.48071413
    706.07603138   651.95015955   605.16728762   564.30111942
    528.27780302   496.27268659   467.64240921]
 [  936.05510082   844.8568212    769.48071413   706.07603138
    651.95015955   605.16728762   564.30111942   528.27770302
    496.27268659   467.64250921   441.87659063]
 [  844.8568212    769.48071413   706.07603138   651.95015955
    605.16728762   564.30111942   528.27770302   496.27268659
    467.64240921   441.87659063   418.56524927]] - and theta_bar is [ 2.51848606e-02  8.09443585e+00 -9.26870087e+00
-5.02794255e+01
  8.29629116e+01  7.92012748e+00 -6.18397554e+01 -2.96464931e+01
  4.59159505e+01  6.33046024e+01 -5.73302872e+01]
```

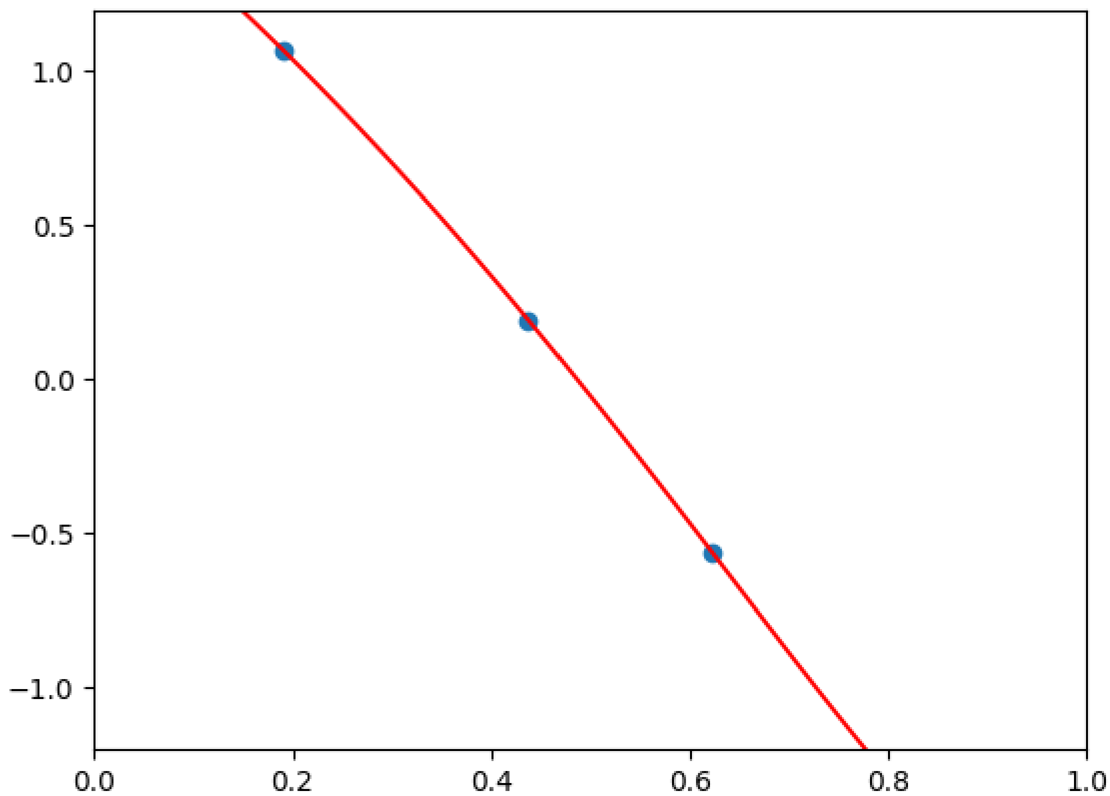
In [ ]:
```python
# Draw 10 samples from posterior distribution and plot the corresponding models
n_samples = 10
samples = np.random.multivariate_normal(mean=theta_bar, cov=A_inv, size=n_samples)
plt.figure(figsize=(8,5))
for i in range(n_samples):
    plt.plot(samples[i], label=f"No. {i}")
plt.title(f'Sample of {n_samples}')
plt.legend()
plt.show()
```

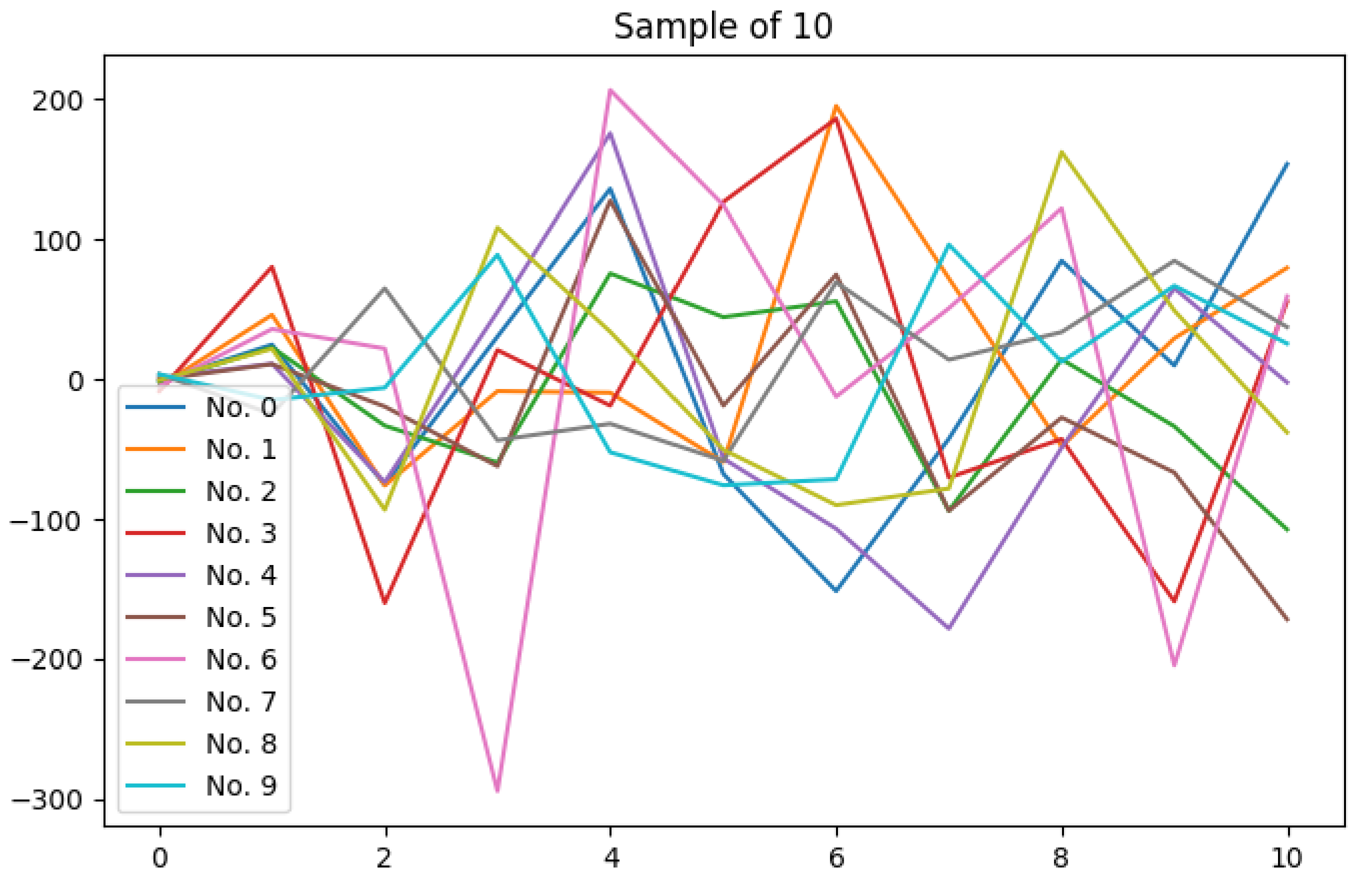
```
C:\Users\Amir Hossein\AppData\Local\Temp\ipykernel_19400\2040710156.py:3: RuntimeWarning: covariance is not symmetric positive-semidefinite.
  samples = np.random.multivariate_normal(mean=theta_bar, cov=A_inv, size=n_samples)
```

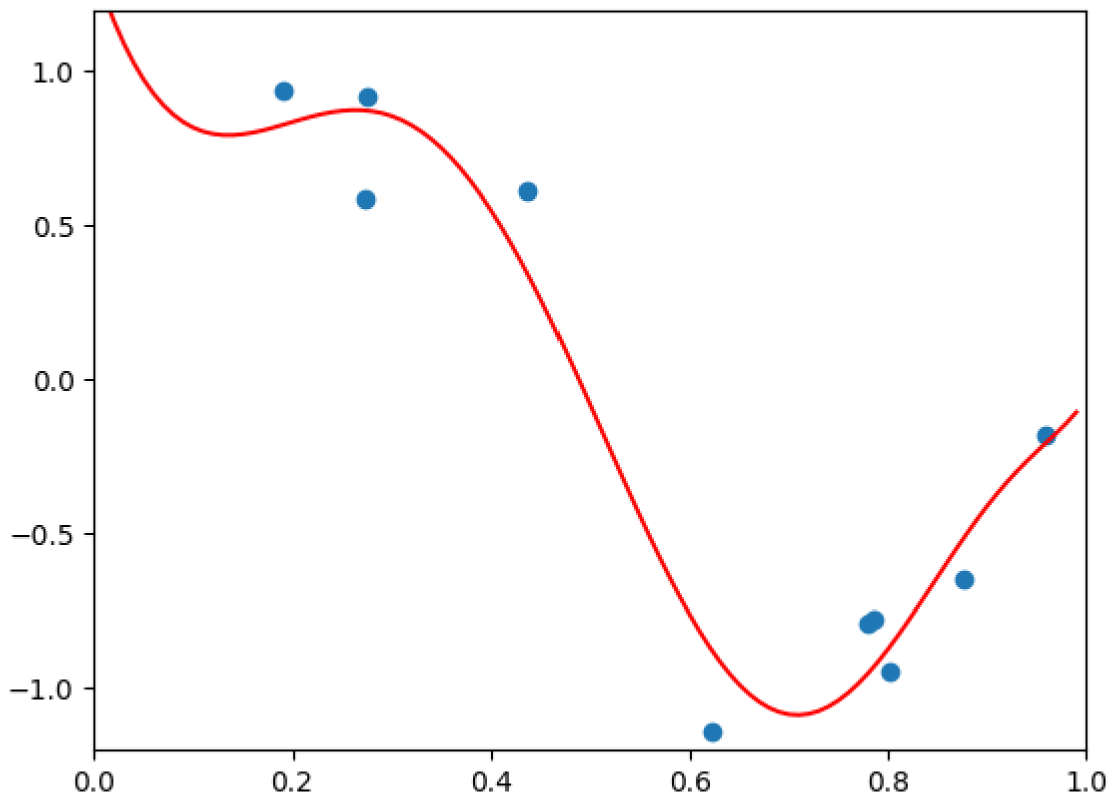


In [ ]:

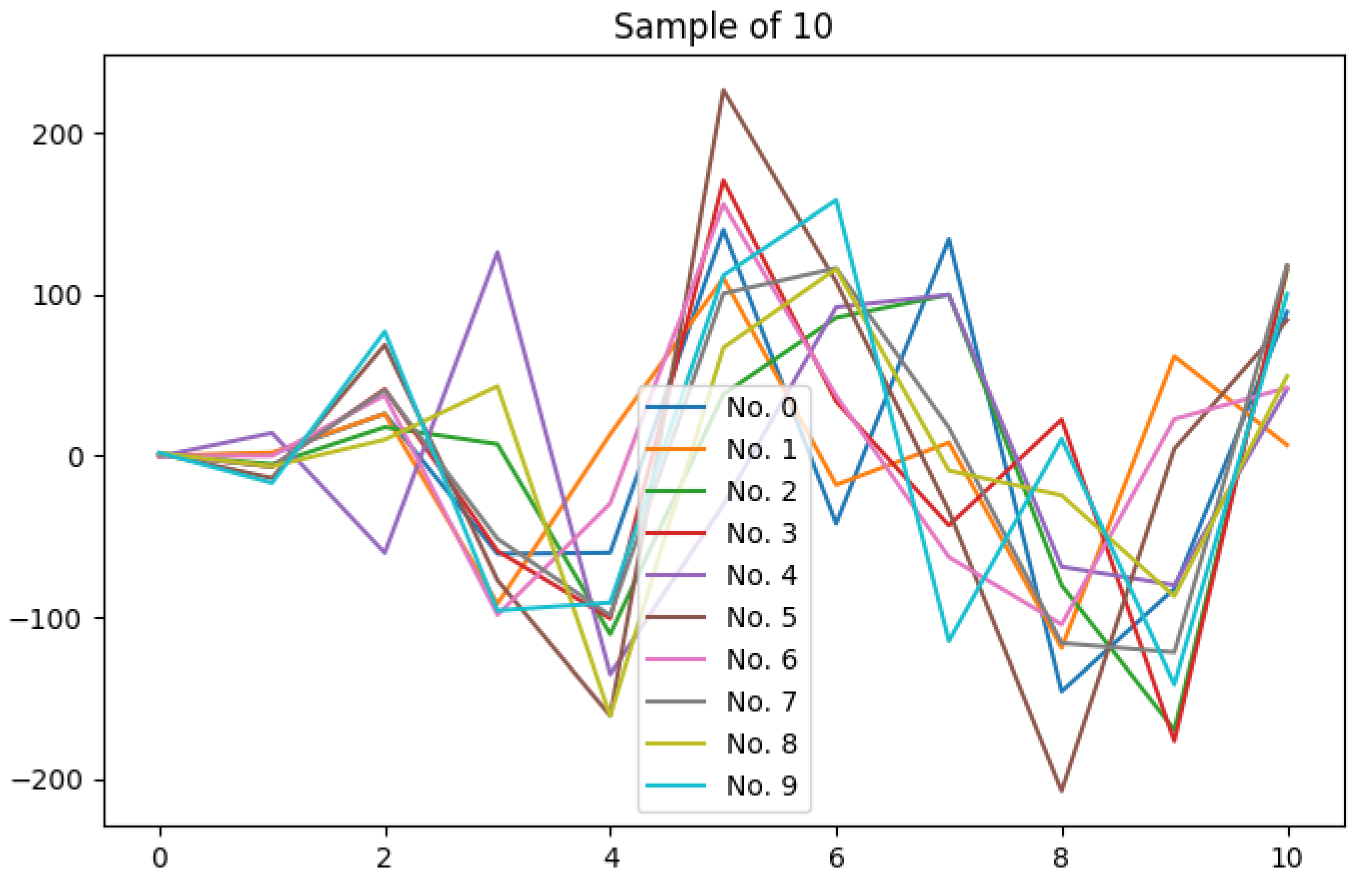
```python
N_array = [3, 10 , 100]
for N in N_array:
    x,y = sine_data_set(N)
    plt.xlim([0,1])
    plt.ylim([-1.2,1.2])
    plt.scatter(x,y)
    d = 10
    X = poly_features(x,d)


    sigma = 0.1
    sigma_p = 100
    M = d + 1

    # Compute MAP model
    I = np.eye(M)
    sigma_sqr = sigma ** -2
    sigma_p_sqr = sigma_p ** -2
    theta_MAP = sigma_sqr * np.linalg.inv((sigma_sqr * X.T @ X) + (sigma_p_sqr * I)) @ X.T @ y
    # Plot MAP model
    x_axis = np.arange(0, 1, 0.01)
    x_values = poly_features(x_axis, d)
    predictions = predict_regression(x_values, theta_MAP)
    plt.plot(x_axis, predictions, label="predictions", c='r')
    plt.show()
    A = (sigma_sqr * X.T @ X) + (sigma_p_sqr * I)
    A_inv = np.linalg.inv(A)
    theta_bar = sigma_sqr * A_inv @ X.T @ y
    n_samples = 10
    samples = np.random.multivariate_normal(mean=theta_bar, cov=A_inv, size=n_samples)
    plt.figure(figsize=(8,5))
    for i in range(n_samples):
        plt.plot(samples[i], label=f"No. {i}")
    plt.title(f'Sample of {n_samples}')
    plt.legend()
    plt.show()
```

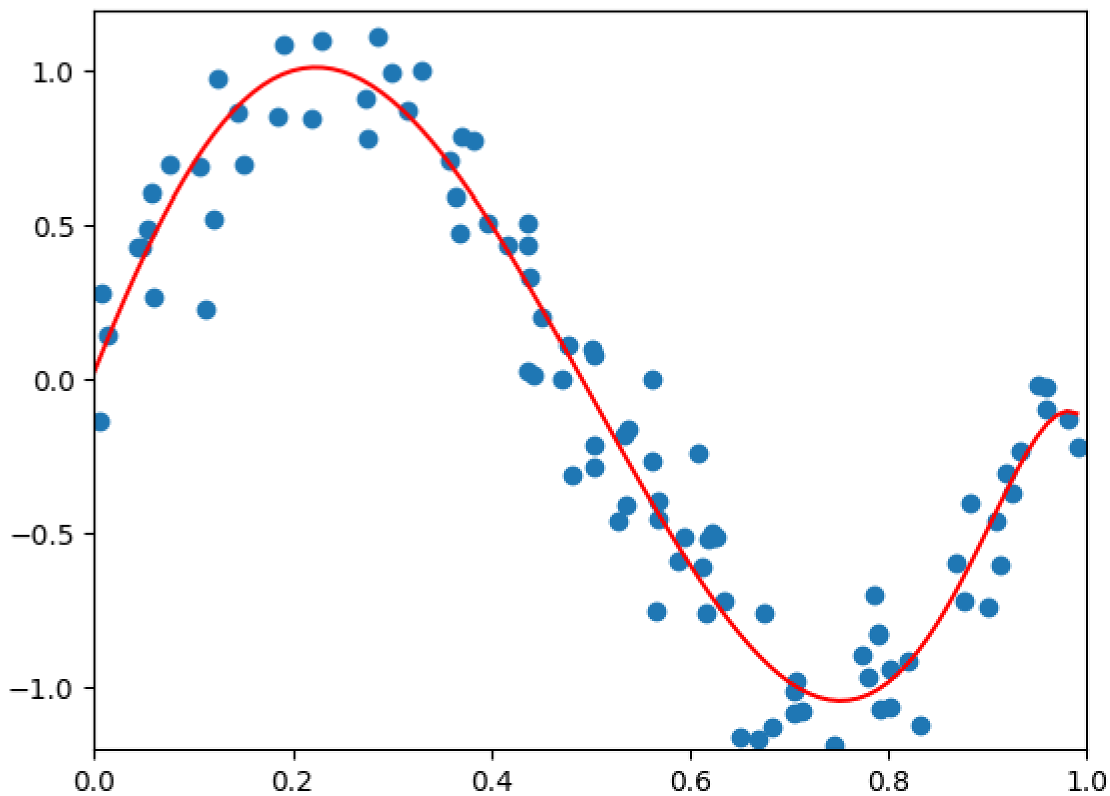

1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0


Sample of 10
200
100
0
−100
−200
−300
0
2
4
6
8
10
No. 0
No. 1
No. 2
No. 3
No. 4
No. 5
No. 6
No. 7
No. 8
No. 9

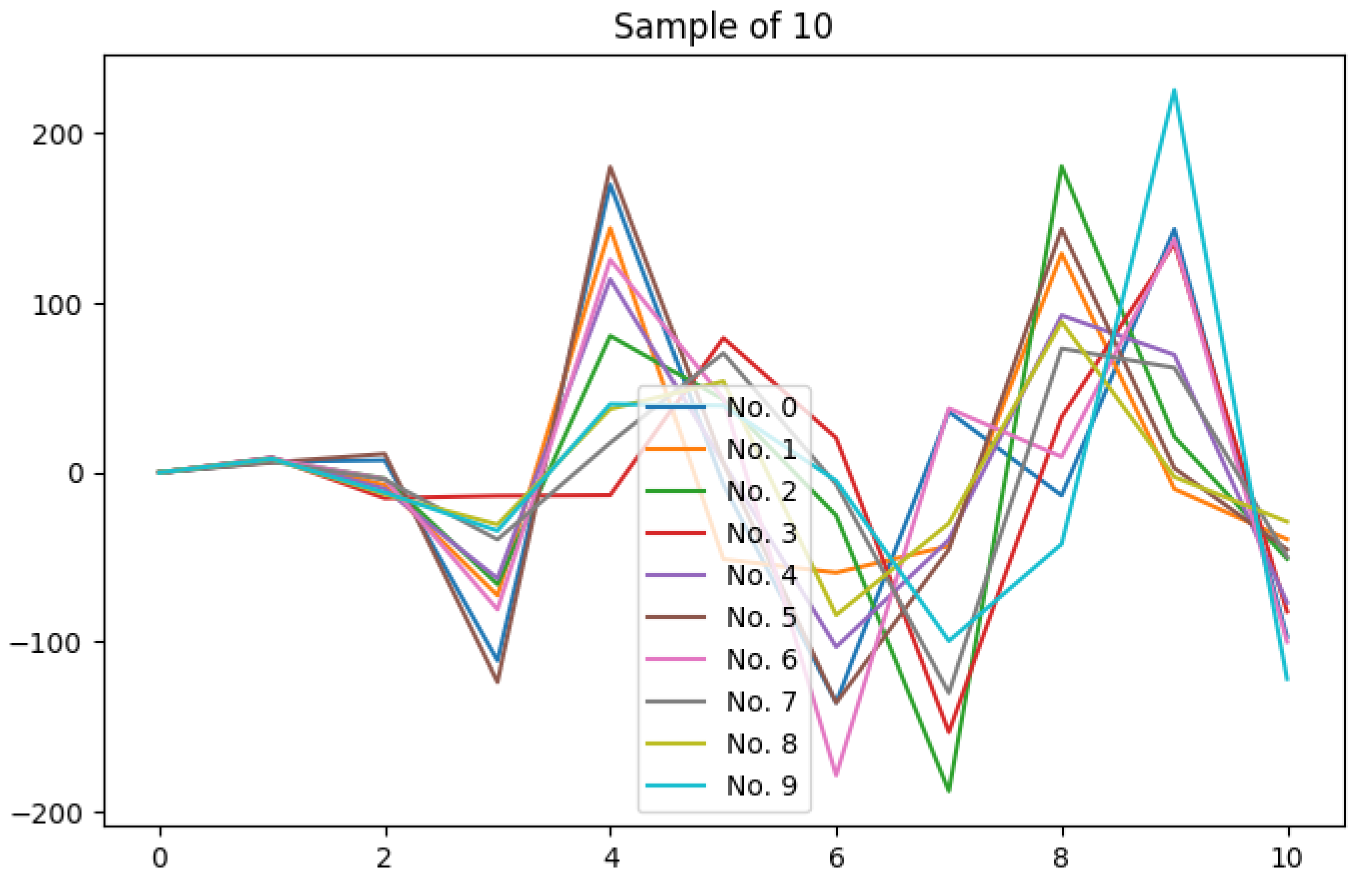


1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0


Sample of 10
200
100
0
−100
−200
0
2
4
6
8
10
No. 0
No. 1
No. 2
No. 3
No. 4
No. 5
No. 6
No. 7
No. 8
No. 9




```
C:\Users\Amir Hossein\AppData\Local\Temp\ipykernel_19400\1959851063.py:30: RuntimeWarning: covariance is not symmetri
c positive-semidefinite.
  samples = np.random.multivariate_normal(mean=theta_bar, cov=A_inv, size=n_samples)
```


Sample of 10
200
100
0
−100
−200
0
2
4
6
8
10
No. 0
No. 1
No. 2
No. 3
No. 4
No. 5
No. 6
No. 7
No. 8
No. 9

```
In [ ]:
```