Solution of Quiz 1

June 24, 2023

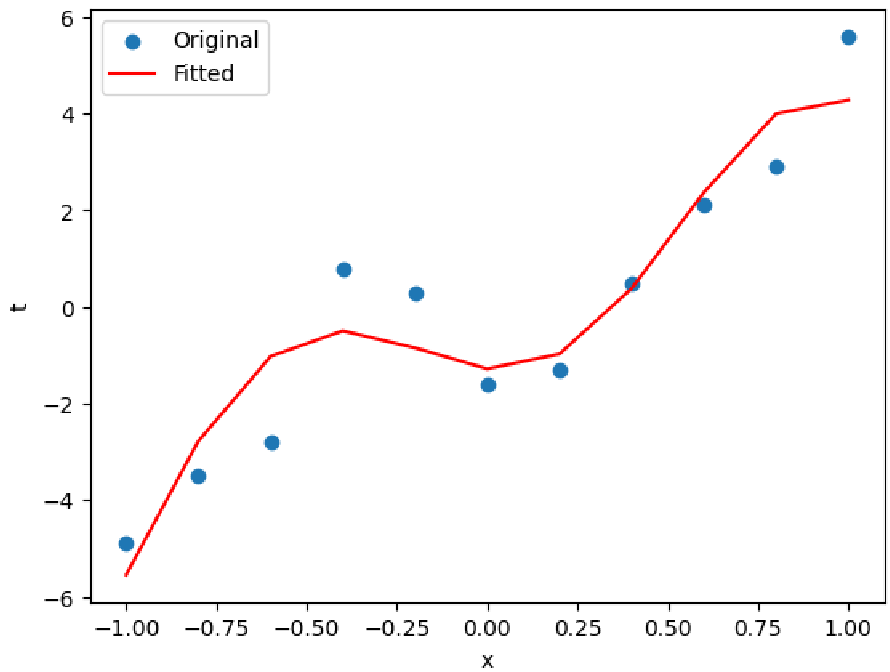
Solution of Q1

$$
\begin{aligned}
mse &= \|d - UW\|^2 \\
\frac{\partial}{\partial w} mse &= \frac{\partial}{\partial w} \|d - UW\|^2 \\
\frac{\partial}{\partial w} (d - UW)^T (d - UW) &= 0 : \text{for minima} \\
\frac{\partial}{\partial w} (d^T d - d^T UW - W^T U^T d + W^T U^T UW) &= 0 \\
-2U^T d + 2U^T UW &= 0 \\
U^T UW &= U^T d \\
W &= (U^T U)^{-1} U^T d
\end{aligned}
\tag{1}
$$

```
#### Solution of Q2.

import numpy as np
def gradient_loop(runs=3):
    """ Repeatedly computes the gradient of a function
        Computes the gradient given the starting points and then uses the result o
        Prints out the result of the function at each iteration
        :param: runs: number of iterations to compute
    """
    # starting points
    x = np.array([1, 2, 3])

    # quadratic function, a parabola
    y = x**2

    for run in range(0, runs):
        print("Iter " + str(run) + ": Y=" + str(y))

        # compute first derivative
        x = np.gradient(y, 1)
        # update the function output
        y = x ** 2
gradient_loop()
```

```
Iter 0: Y=[1 4 9]
Iter 1: Y=[ 9. 16. 25.]
Iter 2: Y=[49. 64. 81.]
```

```
#### Solution of Q3.


import numpy as np
import matplotlib.pyplot as plt


def gaussian_basis(x, mu, gamma=1):
  return np.exp(-gamma * np.linalg.norm(mu-x)**2)

x = np.array([-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1])
t = np.array([-4.9, -3.5, -2.8, 0.8, 0.3, -1.6, -1.3, 0.5, 2.1, 2.9, 5.6])

M = 4

# Calculate design matrix Phi
Phi = np.ones((t.shape[0], M))
for m in range(M-1):
  mu = m/M
  Phi[:, m+1] = np.vectorize(gaussian_basis)(x, mu)



# Calculate parameters w and alpha
w = np.linalg.inv(Phi.T @ Phi) @ Phi.T @ t
alpha = sum((t - Phi @ w)**2) / len(t)


t_fit=Phi @ w


plt.scatter(x,t ,label='Original')
plt.plot(x,t_fit,color='red', label='Fitted')
plt.xlabel('x')
plt.ylabel('t')
plt.legend()
plt.show()
```



Colab paid products - Cancel contracts here

✓ 0s completed at 15:04

Solution of Q4 (a) For a wide-sense stationary

$$\begin{aligned} r_x(k) &= E\{x(n+k)x^*(n)\} \\ &= E\{x^*(n)x(n+k)\} \\ &= r_x^*(-k) \end{aligned} \tag{2}$$

(b) The cost function of first order linear predictor is

$$\begin{aligned} J(w) &= E\{|w^H x - x(n+1)|^2\} \\ &= w^H R_x W = W^H r_{xd} - r_{xd}^H w + r_x(0) \end{aligned} \tag{3}$$

where

$$R_x = \begin{bmatrix} r_x(0) & r_x * (1) \\ r_x(1) & r_x(0) \end{bmatrix}$$

$$r_{xd} = E\{[x(n)\ x(n-1)]^T x^*(n+1)\} = [r_x(1)\ r_x(2)]^T \tag{5}$$

Then the optimum solution is W$_{opt} = R_x^{-1} r_{xd}$