MIDSEM MACHINE LEARNING

Q1) Given a standard observation model in additive noise

$$d(i) = U_i^H w + n(i)$$

in matrix form for all observations

$$d = Uw + n$$

d is observed data vector
U is matrix of known column vectors
w is unknown weight vector. to be estimated
n is noise

Least Square Solution

find estimate $\hat{w}$ that minimizes the squared distance (error) between d and $U\hat{w}$

$$\min_w \|d - Uw\|^2$$

The solution is given by normal equation

$$U^H U \hat{w} = U^H d$$

Solving for $\hat{w}$

$$\boxed{\hat{w} = (U^H U)^{-1} U^H d}$$

$\hat{w}$ is the weight vector that best fits the observed data in least squares sense.

U^H denoted conjucate transpose (Hermitian transpose)

PAGE NO.:

For real valued matrices U^H is just transpose U^T
For Complex-valued matrices U^H is transpose of U with each element replaced by its Complex Conjugate

Qy) a) Show that the matrix $\Phi(\Phi^T\Phi)^{-1}\Phi^T$ takes any vector v and projects it onto the space spanned by the columns of Φ. use this result to show that the least square solution $w_{ML} = \Phi(\Phi^T\Phi)^{-1}\Phi^T t$ corresponds to an orthogonal projection of the vector t onto the manifold S (Space Span by Column of Φ)

Soln

if we denote $v^* = (\Phi^T\Phi)^{-1}\Phi^T v$ then

$$\Phi(\Phi^T\Phi)^{-1}\Phi^T v = \Phi v^* \quad \text{---(1)}$$

By defination Φv^* is in the Column Space of Φ

In other words we have prove

$\Phi(\Phi^T\Phi)^{-1}\Phi^T$ can project a vector v into the Column Space Φ Next we are required to prove the residue of the projection shown in below fig (y-t is orthogonal to Column Space of Φ)

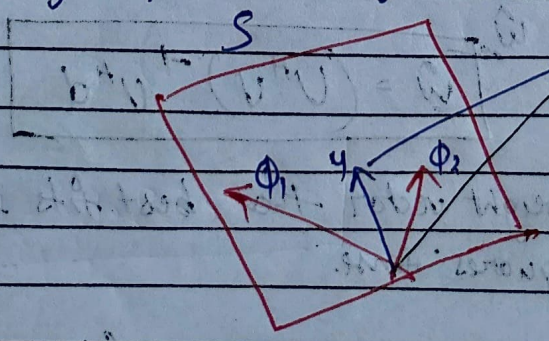


figure:- The least square regression function is obtained by

PAGE NO.:

finding orthogonal projection of data vector t onto the subspace spanned by the basis function $\phi_j(x)$ in which each basis function is viewed as a vector ϕ_j of length N with elements $\phi_j(x_n)$

Since we have

$$(y-t)^T\phi = (\phi w_{ML} - t)^T$$

$$= (\phi(\phi^T\phi)^{-1}\phi^T t - t)^T\phi$$

$$= t^T(\phi(\phi^T\phi)^{-1}\phi^T - I)^T\phi$$

$$= t^T(\phi(\phi^T\phi)^{-1}\phi^T - I)\phi \quad -(2)$$

$$= t^T(\phi - \phi)$$

$$= 0$$

which means $(y-t)$ is the left null space of ϕ and its orthogonal to the column space ϕ

Q6) Given joint probability for 3 observed data points

$$x_1 = 9 \quad x_2 = 9.5 \quad x_3 = 11$$

$$p(9, 9.5, 11; \mu, \beta) = \prod_{i=1}^{3} \frac{1}{\beta\sqrt{2\pi}} \exp\left(-\frac{(x_i-\mu)^2}{2\beta^2}\right)$$

To maximize probability w.r.t μ and β

Find MLE for normal distribution.

PAGE NO.:

Step 2

⇒ Log likelihood

$$l(\mu,\sigma) = \log P = \sum_{i=1}^{3}\left[-\log\left(\sigma\sqrt{2\pi}\right) - \frac{(x_i-\mu)^2}{2\sigma^2}\right]$$

$$= -3\log\left(\sigma\sqrt{2\pi}\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{3}(x_i-\mu)^2$$

Step 3

Maximize w.r.t. μ

$$\frac{\partial l}{\partial \mu} = \frac{-1}{2\sigma^2}\cdot 2\sum_{i=1}^{3}(x_i-\mu)(-1)$$

$$= \frac{1}{\sigma^2}\sum_{i=1}^{3}(x_i-\mu)$$

Set to zero

$$\sum_{i=1}^{3}(x_i-\mu) = 0 \Rightarrow \mu = \frac{1}{3}(x_1+x_2+x_3)$$

$$\mu = \frac{1}{3}(9+9.5+11) = \frac{29.5}{3} = 9.833$$

$$\boxed{\mu = 9.833}$$

PAGE NO.:

Step 3: Maximize w.r.t β

$$\frac{\partial l}{\partial \beta} = -\frac{3}{\beta} + \frac{1}{\beta^3} \sum_{i=1}^{3} (x_i - \mu)^2 = 0$$

multiply by β^3

$$\Rightarrow -3\beta^2 + \sum_{i=1}^{3} (x_i - \mu)^2 = 0$$

$$\beta^2 = \frac{1}{3} \sum_{i=1}^{3} (x_i - \mu)^2$$

put value of x_1, x_2, x_3 and μ

$$\beta^2 = \frac{1}{3} \left[(9 - 9.833)^2 + (9.5 - 9.833)^2 + (11 - 9.833)^2 \right]$$

$$= \frac{(-0.833)^2 + (-0.333)^2 + (1.167)^2}{3}$$

$$\beta^2 = \frac{2.166}{3} = 0.722$$

$$\boxed{\beta = \sqrt{0.722} = 0.85}$$

PAGE NO.:

Q) Determine the condition under which the least square solution is equivalent to maximum likelihood estimates

Soln Suppose we have linear model

$$y = Xw + \epsilon$$

y is vector of observed output
X is design matrix
w is the vector of parameters to estimate
ϵ is error (noise) vector)

Least squares

The least square method find w that minimizes the sum of squared errors.

$$\min_w \|y - Xw\|^2$$

Maximum likelihood

under Gaussian noise assumption the likelihood of observing y given X and w is

$$p(y|X, w) = N(y|Xw, \beta^2 I)$$

log likelihood is

$$\log p(y|X, w) = \frac{1}{2\beta^2} \|y - Xw\|^2 + \text{const}$$

PAGE NO.:

maximizing log likelihood w.r.t w is equivalent to minimizing $\|y - Xw\|^2$

⇒ if the noise is i.i.d Gaussian with zero mean and constant variance, the least square estimator and the M.L.E for w are same

⇒ if the noise is not Gaussian least square is not generally the same as MLE

Q2) Code block is attached

Q3) Code block is attached

Question 2

Computes the gradient of the quadratic function of x given the starting points x=[1,2,3] and then uses the result of the gradient to feed the next iterations, with new points. Prints out the result of the function at each iteration till 3rd run. Use Python script to print the results.

```
import numpy as np

# Define the quadratic function: f(x) = x^T x
def f(x):
    return np.dot(x, x)

# Its gradient: grad_f(x) = 2x
def grad_f(x):
    return 2 * x

# Starting point
x = np.array([1.0, 2.0, 3.0])

# Learning rate
alpha = 0.1

print("Iteration results:")
for i in range(3):
    fx = f(x)
    print(f"Iteration {i+1}: f(x) = {fx:.4f}, x = {x}")
    x = x - alpha * grad_f(x)
```

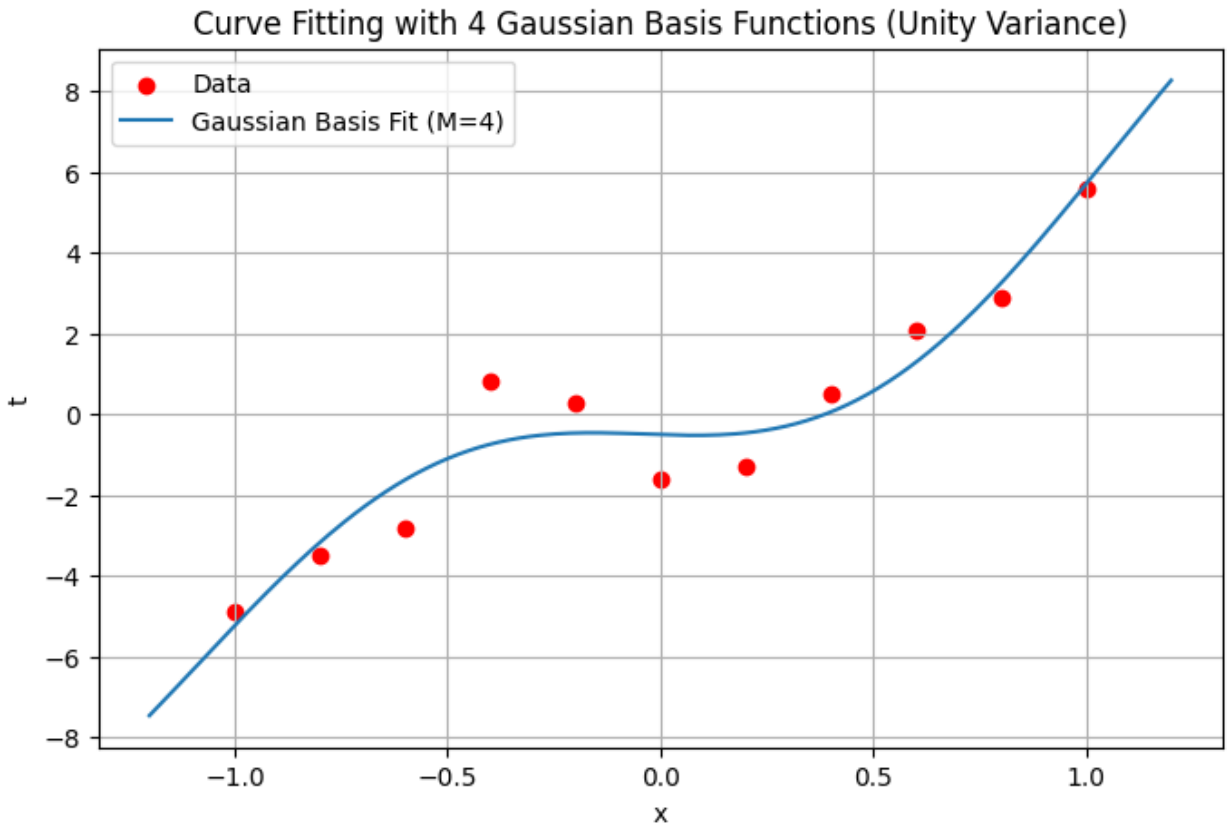
```
Iteration results:
Iteration 1: f(x) = 14.0000, x = [1. 2. 3.]
Iteration 2: f(x) = 8.9600, x = [0.8 1.6 2.4]
Iteration 3: f(x) = 5.7344, x = [0.64 1.28 1.92]
```

Question 3

With given input and output relation x = [-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1] t = [-4.9, -3.5, -2.8, 0.8, 0.3, -1.6, -1.3, 0.5, 2.1, 2.9, 5.6] Please fit a curve with M=4 Gaussian basis functions having unity variance.

```
import numpy as np
import matplotlib.pyplot as plt

# Given data
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])

# Number of Gaussian basis functions
```

```
M = 4

# Centers of the Gaussians (evenly spaced in x range)
centers = np.linspace(np.min(x), np.max(x), M)
variance = 1.0  # Unity variance

# Design matrix using Gaussian basis functions
def design_matrix(x, centers, variance):
    Phi = np.zeros((len(x), len(centers)))
    for i, c in enumerate(centers):
        Phi[:, i] = np.exp(-0.5 * ((x - c) ** 2) / variance)
    return Phi

Phi = design_matrix(x, centers, variance)

# Fit weights using least squares
w = np.linalg.pinv(Phi) @ t

# Predict function
def predict(x_new):
    Phi_new = design_matrix(x_new, centers, variance)
    return Phi_new @ w

# Plotting
x_plot = np.linspace(-1.2, 1.2, 200)
t_pred = predict(x_plot)

plt.figure(figsize=(8, 5))
plt.scatter(x, t, color='red', label='Data')
plt.plot(x_plot, t_pred, label='Gaussian Basis Fit (M=4)')
plt.xlabel('x')
plt.ylabel('t')
plt.title('Curve Fitting with 4 Gaussian Basis Functions (Unity
Variance)')
plt.legend()
plt.grid(True)
plt.show()
```


Curve Fitting with 4 Gaussian Basis Functions (Unity Variance)
Data
Gaussian Basis Fit (M=4)
t
x
8
6
4
2
0
−2
−4
−6
−8
−1.0
−0.5
0.0
0.5
1.0