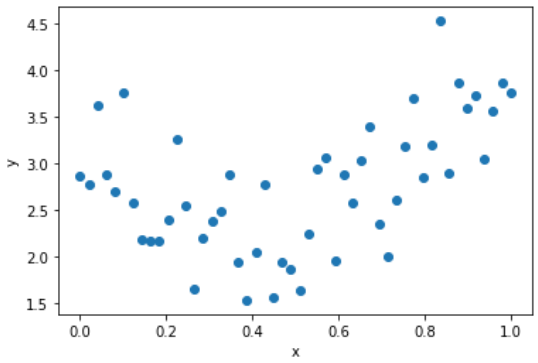
Solution to Homework 1

1. Data set generation: Generate 50 data points using the following python code.

```
import numpy as np
from matplotlib import pyplot as plt
n = 50
x = np.linspace(0, 1, n)
a = 5; b = -4; c = 3
y = a*x*x + b*x + c + 0.5*np.random.randn(n)
plt.plot(x,y,'o')
plt.show()
```

Show the graph of the data points. (1 point)

Answer:



2. In a quadratic regression we model the data relationship using the quadratic model $y = ax^2 + bx + c$ where a, b, and c are the model parameters to be determined based on a loss function. Suppose we use the least squares loss function

$$L(a,b,c) = \frac{1}{2}\sum_{i=1}^{50}\left|ax_i^2 + bx_i + c - y_i\right|^2$$

Determine the derivatives $\frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}, \frac{\partial L}{\partial c}$ (3 points)

Answer:

$$\frac{\partial L}{\partial a} = \sum_{i=1}^{50}\left(ax_i^2 + bx_i + c - y_i\right)x_i^2$$

$$\frac{\partial L}{\partial b} = \sum_{i=1}^{50}\left(ax_i^2 + bx_i + c - y_i\right)x_i$$

$$\frac{\partial L}{\partial c} = \sum_{i=1}^{50}\left(ax_i^2 + bx_i + c - y_i\right)$$

3. Derive the linear system of equations for determining the model parameters *a*, *b*, *c* using the direct method. Write the final form of the system in the matrix form. (3 points)

Answer:

At the global minimum of *L* we have

$$\mathbf{g} = \begin{bmatrix} \frac{\partial L}{\partial a} \\ \frac{\partial L}{\partial b} \\ \frac{\partial L}{\partial c} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{50} \left(ax_i^2 + bx_i + c - y_i \right) x_i^2 \\ \sum_{i=1}^{50} \left(ax_i^2 + bx_i + c - y_i \right) x_i \\ \sum_{i=1}^{50} \left(ax_i^2 + bx_i + c - y_i \right) \end{bmatrix} = \mathbf{0}$$

Rearranging this yields the linear system

$$\begin{bmatrix} \sum_{i=1}^{50} x_i^4 & \sum_{i=1}^{50} x_i^3 & \sum_{i=1}^{50} x_i^2 \\ \sum_{i=1}^{50} x_i^3 & \sum_{i=1}^{50} x_i^2 & \sum_{i=1}^{50} x_i \\ \sum_{i=1}^{50} x_i^2 & \sum_{i=1}^{50} x_i & n \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{50} x_i^2 y_i \\ \sum_{i=1}^{50} x_i y_i \\ \sum_{i=1}^{50} y_i \end{bmatrix}$$

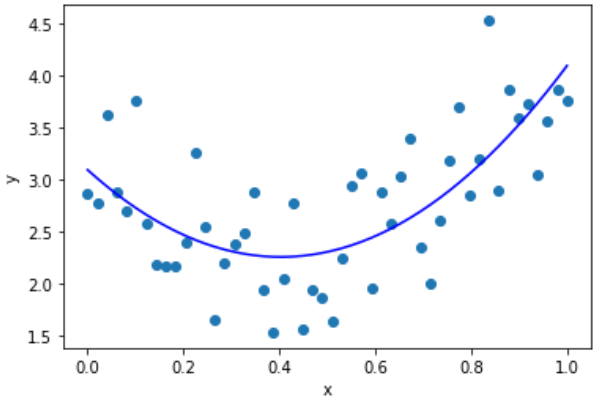
4. Directly solve the corresponding linear system and provide the numerical values of *a*, *b*, *c*. Also provide your source code for solving the system. (2 points)

Answer:

The source code for solving the corresponding system is as follows.

```
x4 = np.sum(x**4)
x3 = np.sum(x**3)
x2 = np.sum(x*x)
x1 = np.sum(x)
A = np.array([[x4,x3,x2],[x3,x2,x1],[x2,x1,n]])
b = np.array([np.sum(x*x*y), np.sum(x*y), np.sum(y)])
m = np.linalg.inv(A).dot(b)
y1 = m[0]*x*x + m[1]*x + m[2]
plot(x,y,'o',x,y1,'b-');xlabel('x');ylabel('y')
```

The obtained values are *a* = 5.14897934, *b* = -4.14947441, *c* = 3.08881005. The obtained quadratic model is shown as the solid line in the below figure.



5. Using the derivatives $\frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}, \frac{\partial L}{\partial c}$ to form the gradient of the loss function with respect to the model parameter $\mathbf{m} = [a, b, c]$, $\mathbf{g} = \left[\frac{\partial L}{\partial a}, \frac{\partial L}{\partial b}, \frac{\partial L}{\partial c}\right]^T$, we can use the gradient descent scheme $\mathbf{m}_{i+1} = \mathbf{m}_i - \alpha_i \mathbf{g}_i$ to find the optimal values of a, b, c. Derive the formula for computing the optimal step length α_i. (3 points)

Answer:

The formula for the optimal step length can be obtained by solving the following equation for α_i.

$$\frac{dL}{d\alpha}(\mathbf{m} - \alpha\mathbf{g}) = \frac{d}{d\alpha}\left[\frac{1}{2}\sum_{j=1}^{50}\left((a-\alpha g_a)x_j^2 + (b-\alpha g_b)x_j + (c-\alpha g_c) - y_j\right)^2\right] = 0$$

$$\frac{1}{2}\sum_{j=1}^{50}\frac{d}{d\alpha}\left((a-\alpha g_a)x_j^2 + (b-\alpha g_b)x_j + (c-\alpha g_c) - y_j\right)^2 = 0$$

$$\sum_{j=1}^{50}\left((a-\alpha g_a)x_j^2 + (b-\alpha g_b)x_j + (c-\alpha g_c) - y_j\right)\left(-g_a x_j^2 - g_b x_j - g_c\right) = 0$$

$$\alpha\sum_{j=1}^{50}\left(g_a x_j^2 + g_b x_j + g_c\right)\left(g_a x_j^2 + g_b x_j + g_c\right) = \sum_{j=1}^{50}\left(ax_j^2 + bx_j + c - y_j\right)\left(g_a x_j^2 + g_b x_j + g_c\right)$$

$$\alpha = \frac{\sum_{j=1}^{50}\left(ax_j^2 + bx_j + c - y_j\right)\left(g_a x_j^2 + g_b x_j + g_c\right)}{\sum_{j=1}^{50}\left(g_a x_j^2 + g_b x_j + g_c\right)^2}$$

So, evaluating at the i^{th} iterate of the model parameter $\mathbf{m}_i = [a_i, b_i, c_i]$, the formula then becomes

$$\alpha_i = \frac{\sum_{j=1}^{50}\left(a_i x_j^2 + b_i x_j + c_i - y_j\right)\left(g_a x_j^2 + g_b x_j + g_c\right)}{\sum_{j=1}^{50}\left(g_a x_j^2 + g_b x_j + g_c\right)^2}$$

where

$$g_a = \frac{\partial L}{\partial a} = \sum_{j=1}^{50}\left(a_i x_j^2 + b_i x_j + c - y_j\right)x_j^2$$

$$g_b = \frac{\partial L}{\partial b} = \sum_{j=1}^{50}\left(a_i x_j^2 + b_i x_j + c - y_j\right)x_j$$

$$g_c = \frac{\partial L}{\partial c} = \sum_{j=1}^{50}\left(a_i x_j^2 + b_i x_j + c - y_j\right)$$

6. Write a computer program to implement the steepest descent method to find the parameters a, b, c. You can choose your own stopping criterion, e.g., the number of iterations reaches 1000 or the loss function is smaller than 10^{-4}. Show the values of a, b, c you obtain and also show your source code. (4 points)

Answer:

The source code of the steepest descent implementation is as follows.

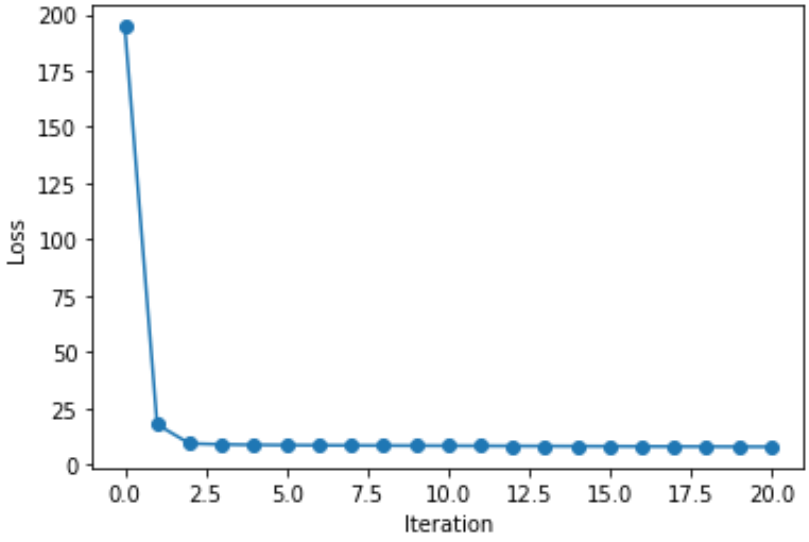
```
# m = [a,b,c]
def grad(m,x,y):
    e = m[0]*x*x+m[1]*x+m[2]-y
    return np.array([np.sum(e*x*x), np.sum(e*x), np.sum(e)])

# g = [ga,gb,gc]
def step(m,x,y,g):
    e = m[0]*x*x+m[1]*x+m[2]-y
    gg = g[0]*x*x+g[1]*x+g[2]
    return np.sum(e*gg)/np.sum(gg*gg)
```

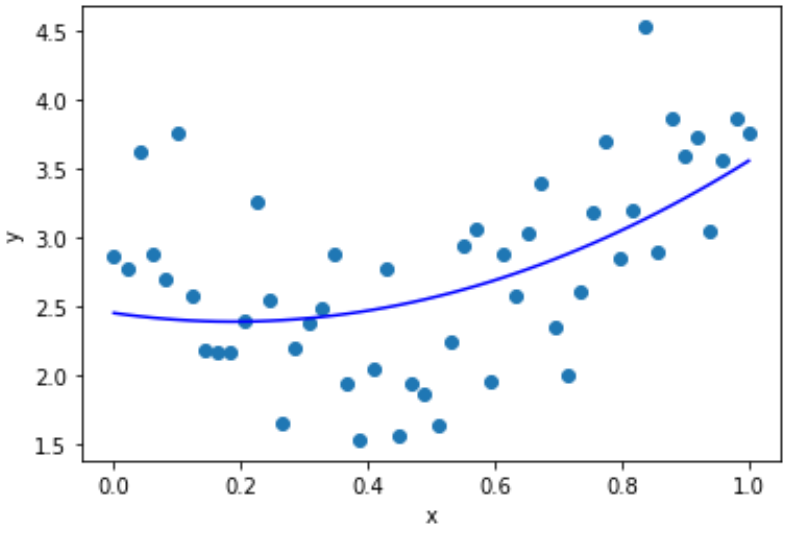
```
def loss(m,x,y):
    e = m[0]*x*x+m[1]*x+m[2]-y
    return 0.5*np.sum(e*e)

# implementation of BGD
m = np.random.randn(3)
niter = 20
l = np.zeros(niter+1)
l[0] = loss(m,x,y)
for i in range(niter):
    g = grad(m,x,y)
    s = step(m,x,y,g)
    m = m - s*g
    l[i+1] = loss(m,x,y)
plot(np.arange(niter+1),l,'o-')
xlabel('Iteration'); ylabel('Loss')
```

The values of loss function over 20 iterations are plotted in the figure below.



The obtained parameter values are a = 1.76757532, b = -0.66425385, c = 2.44787017. Using the obtained parameters, the prediction is shown as the solid line in the figure below. Comparing with the true values, the obtained values are inaccurate.



Instead, using 2000 iterations, the obtained values are a = 5.14896829, b = -4.14946295, c = 3.08880802. According to the results, the batch gradient descent method can converge to the global minimum but requires a large number of iterations, i.e., the batch gradient descent method has a low convergence rate.