

Learning from Data

Assignment 2

Question 1:

The Added/Changed Code Statement	Explanation
<pre>def sigmoid(x): return 1 / (1 + math.exp(-x))</pre>	This is a function that returns the sigmoidal value of a parameter passed to it.
<pre>sum = 0 for i in range(len(X)): sig = sigmoid(np.dot(np.transpose(X[i]),w)) sum += (sig - y[i]) * sig * (1 - sig) * X[i] sum = np.asarray(sum) grad = 2 * sum grad.shape = (2,1)</pre>	This is where the gradient is calculated by using the logistic regression gradient descent calculation formula. For each subarrays of X and y, calculate gradient, make the result a numpy array, reshape gradient and use it for the iterations.

The screenshot displays a Jupyter Notebook environment with a Python script for logistic regression. The code includes a sigmoid function definition, a loop for calculating the gradient, and a 3D plot of the cost function. The 3D plot shows a surface with a minimum point, indicating the optimal parameters for the model.

```
cost = compute_cost(w0)
cost_path.append(cost)
w = w0

# start gradient descent loop
max_its = 5000
alpha = 10**(-2)
for k in range(max_its):
    # compute gradient
    #print(X.shape)
    #print(w.shape)
    #print(y.shape)
    sum = 0
    for i in range(len(X)):
        sig = sigmoid(np.dot(np.transpose(X[i]),w))
        sum += (sig - y[i]) * sig * (1 - sig) * X[i]
    sum = np.asarray(sum)
    grad = 2 * sum
    grad.shape = (2,1)

    # take gradient step
    w = w - alpha*grad

    # update path containers
    w_path.append(w)
    cost = compute_cost(w)
    cost_path.append(cost)

# reshape containers for use in plotting in 3d
w_path = np.asarray(w_path)
w_path.shape = (np.shape(w_path)[0],np.shape(w_path)[1])
return w_path,cost_path

# calculate the cost value for a given input weight w
def compute_cost(w):
    term = 1/(1 + math.exp(np.dot(X,w))) - y
```

The 3D plot, titled 'Figure 1', shows a surface representing the cost function. The x and y axes range from -3 to 3, and the z-axis (cost) ranges from 1 to 5. The surface is a smooth, curved plane with a minimum point, indicating the optimal parameters for the model.

Question 2:

The Added/Changed Code Statement	Explanation
<pre>def sigmoid(x): return 1 / (1 + math.exp(-x))</pre>	This is a function that returns the sigmoidal value of a parameter passed to it.
<pre>sum = 0 for i in range(len(X)): sig = sigmoid(np.dot(np.transpose(X[i]),w)) sum += (sig - y[i]) * sig * (1 - sig) * X[i] ar = [] ar.append(0) ar.append(w[1][0]) ar = np.asarray(ar) ar = np.reshape(ar, (2,1)) sum = np.reshape(sum, (2,1)) # regularize it sum += 2 * lam * ar</pre>	This is where the gradient is calculated by using the logistic regression gradient descent calculation formula. The difference between the above one is the regularization term.

