HW3

eadebayo

November 2023

1 problem 1

1. $\hat{y} = \theta X$, The loss function is defined as the absolute difference. that is, $|y_i - \hat{y}|$
   The total loss over the entire dataset.
   F(θ) =
   $$\sum_{i=1}^{n} |y_i - \hat{y}|$$
   ,

2 problem 2

1. (a) $w \in \mathbb{R}^4$
$L(W) = w_1^2 + 2w_2^2 + w_3^2 2w_3 w_4 + w_4^2 + 2w_1 4w_2 + 4$
$\frac{\partial L}{\partial w_1} = 2w_1 + 2$
$\frac{\partial L}{\partial w_2} = 4w2 - 4$
$\frac{\partial L}{\partial w_3} = 2w_3 - 2w_4$
$\frac{\partial L}{\partial w_4} = -2w_3 + 2w_4$

∇L(W) =

$$\begin{bmatrix} \frac{\partial L}{\partial w_1} = 2w_1 + 2 \\ \frac{\partial L}{\partial w_2} = 4w_2 - 4 \\ \frac{\partial L}{\partial w_3} = 2w_3 - 2w_4 \\ \frac{\partial L}{\partial w_4} = -2w_3 + 2w_4 \end{bmatrix}$$

(b) if W = (0, 0, 0, 0) with each index corresponding to w1,w2,w3,w4 and the step size is some η, Then,
$w_1 = w_1 - \eta(2w_1 + 2)$
$w_1 = 0 - \eta(2 * 0 + 2)$
$w_1 = -2\eta$

$w_2 = w_2 - \eta(4w_2 - 4)$
$w_2 = 0 - \eta(4 * 0 - 4)$
$w_2 = 4\eta$

$w_3 = w_3 - \eta(2w_3 - 2w_4)$
$w_3 = 0 - \eta(2 * 0 - 2 * 0)$
$w_3 = 0$

$w_4 = w_4 - \eta(-2w_3 + 2w_4)$
$w_4 = 0 - \eta(-2 * 0 + 2 * 0)$
$w_4 = 0$

$W = (-2\eta, 4\eta, 0, 0)$

3 problem 3

1. let $w_1, b_1, w_2, b_2, w_3, b_3$ denote the weights of the first hidden layer, biases of the hidden layer, weights of the second hidden layer, biases of the second hidden layer, weights of the output of layer and the biases of the output layer respectively.
   Assuming we are using the sigmoid activation function, let z_1, z_2, z_3 represent the output of the weighted sum of first, second, and output layer. let a_1, a_2, a_3 represent the activated neuron of the first, second, and output layer.
   Let X represent the data
   The forward propagation is as follows,
   $z_1 = w_1.X + b_1$
   $a_1 = sigmoid(z_1)$
   $z_2 = w_2.a_1 + b_2$
   $a_2 = sigmoid(z_2)$
   $z_3 = w_3.a_2 + b_3$
   $a_3 = sigmoid(z_3)$
   Assuming we are using the logistic loss function
   $J = -[ylog(a_3) + (1-y)log(1-a_3)]$
   $\frac{\partial J}{\partial X}$ is computed as follows,
   $\frac{\partial J}{\partial a_3} = -\frac{y}{a_3} + \frac{1-y}{1-a_3}$
   $\frac{\partial J}{\partial z_3} = \frac{\partial J}{\partial a_3} * \frac{\partial a_3}{\partial z_3}, \frac{\partial a_3}{\partial z_3} = a_3(1-a_3)$
   $\frac{\partial J}{\partial a_2} = \frac{\partial z_3}{\partial a_2} * \frac{\partial z_3}{\partial a_2}, \frac{\partial z_3}{\partial a_2} = w_3$
   $\frac{\partial J}{\partial z_2} = \frac{\partial J}{\partial a_2} * \frac{\partial a_2}{\partial z_2}, \frac{\partial a_2}{\partial z_2} = a_2(1-a_2)$
   $\frac{\partial J}{\partial a_1} = \frac{\partial J}{\partial z_2} * \frac{\partial z_2}{\partial a_1}, \frac{\partial z_2}{\partial a_1} = w_2$
   $\frac{\partial J}{\partial z_1} = \frac{\partial J}{\partial a_1} * \frac{\partial a_1}{\partial z_1}, \frac{\partial a_1}{\partial z_1} = a_1(1-a_1)$
   $\frac{\partial J}{\partial x} = \frac{\partial J}{\partial z_1} * \frac{\partial z_1}{\partial x}, \frac{\partial z_1}{\partial x} = w_1$

4 problem 4

(a)

```python
#read the data into a numpy matrix
def read_data(path):

    #load the file into a numpy array
    data = np.loadtxt(path, dtype='str')

    #return the first two column containing the observed data. x1 and x2
    #return the output label which is the last column
    return data[:, :2], data[:,-1]
```

```python
path_to_file = 'training.txt'
features, annotations = read_data(path_to_file)

print(features)
print(annotations)
```

```
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 % python3 prob4.py
[['blue' 'gloves']
 ['blue' 'hat']
 ['cautious' 'cat']
 ['new' 'hat']
 ['cautious' 'cat']
 ['cautious' 'cat']
 ['cautious' 'hat']
 ['new' 'gloves']
 ['new' 'cat']]
['y' 'y' 'y' 'y' 'y' 'n' 'n' 'n' 'n']
```

(b)

```python
def estimate_cond_probs(feature_values, labels):
    #Extract unique values from features x1 and x2
    unique_features_values_w1 = np.unique(feature_values[:,0])
    unique_features_values_w2 = np.unique(feature_values[:,1])
    #occurence of y and n labels
    num_of_y_label = np.sum(labels == 'y')
    num_of_n_label = np.sum(labels == 'n')
    #sum of y and n label counts
    total_num_of_label = len(labels)
    #dictionaries to store p_i|y and p_i|n
    prob_dict_y = {}
    prob_dict_n = {}
    #store the conditional probability in two dictionaries. one for label and one for label n.
    for w1 in unique_features_values_w1:
        prob_dict_y[w1] =  ( count_conditional_prob(feature_values,labels,0,w1,"y") / num_of_y_label )
        prob_dict_n[w1] =  (count_conditional_prob(feature_values,labels,0,w1,"n") / num_of_n_label )
    for w2 in unique_features_values_w2:
        prob_dict_y[w2] =  (count_conditional_prob(feature_values,labels,1,w2,"y") / num_of_y_label)
        prob_dict_n[w2] =  (count_conditional_prob(feature_values,labels,1,w2,"n") / num_of_n_label)
    #return the dictionaries as a tuple
    return (prob_dict_y, prob_dict_n)


def count_conditional_prob(feature_values, labels, column_index, feature_value_to_search, label_to_search):
    count = 0
    for row, value in zip(feature_values, labels):
        #print(row[column_index], value)
        if ( (row[column_index] == feature_value_to_search) and (value == label_to_search)):
            count = count +1
    # print(count)
    return count
```

```
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 % python3 prob4.py
{'blue': 0.4, 'cautious': 0.4, 'new': 0.2, 'cat': 0.4, 'gloves': 0.2, 'hat': 0.4}
{'blue': 0.0, 'cautious': 0.5, 'new': 0.5, 'cat': 0.5, 'gloves': 0.25, 'hat': 0.25}
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 %
```

(c)

```
path_to_file = 'training.txt'
features, annotations = read_data(path_to_file)

p_y, p_n = estimate_cond_probs(features,annotations)

attribute = "blue"
if_y, if_n = get_conditional_log_probability(p_y, p_n, attribute)

print(if_y)
print(if_n)
```

```
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 % python3 prob4.py
the log conditional probability given y of the attribute is blue: -0.3979400086720376
Cannot take the log probability when it is zero
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 %
```

(d)

```
path_to_file = 'training.txt'
features, annotations = read_data(path_to_file)

p_y, p_n = estimate_cond_probs(features,annotations)

attribute = "cat"
if_y, if_n = get_conditional_log_probability(p_y, p_n, attribute)

print(if_y)
print(if_n)
```

```
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 % python3 prob4.py
the log conditional probability given y of the attribute is cat: -0.3979400086720376
the log conditional probability given n of the attribute is cat: -0.3010299956639812
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 %
```

(e)

```python
def estimate__Y(labels):
    #occurence of y and n labels
    num_of_y_label = np.sum(labels == 'y')
    num_of_n_label = np.sum(labels == 'n')
    #sum of y and n label counts
    total_num_of_label = len(labels)

    prior_y = num_of_y_label / total_num_of_label
    prior_n = num_of_n_label / total_num_of_label

    label_prob_dict = {}
    label_prob_dict['y'] = prior_y
    label_prob_dict['n'] = prior_n
    return label_prob_dict
```

```
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 % python3 prob4.py
the prior probability for class y is 0.5555555555555556
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 %
```

```python
estimate_text = 'y'
labels = estimate__Y(annotations)
print("the prior probability for class "+ estimate_text+ " is "+str(labels[estimate_text]))
```

(f)

```
def get_log_prior_probability(y):

    return np.log10(y)
```

```
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 % python3 prob4.py
the log prior probability for class y is -0.25527250510330607
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 %
```

(g)

```
def classify(trained_model, X):

    prob_cond_y, prob_cond_n, prior_prob_y = trained_model

    feature_w1 = X[0]
    feature_w2 = X[1]
    try:
        check_if_yes = np.log10(prob_cond_y[feature_w1]) + np.log10(prob_cond_y[feature_w2]) + np.log10(prior_prob_y['y'])

        check_if_no = np.log10(prob_cond_n[feature_w1]) + np.log10(prob_cond_n[feature_w2]) + np.log10(prior_prob_y['n'])
    except:
        print(f"An attempt to take the log zero has caused an error: ")
    if (check_if_yes > check_if_no):
        return "Y"

    return "N"
```

(h)

```
tuple_of_trained_model = p_y, p_n, estimate__Y(annotations)
unlabeled_new_feature = ["cautious", "gloves"]
print ( classify(tuple_of_trained_model, unlabeled_new_feature) )
```

```
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 % python3 prob4.py
N
emmanueladebayo@Macs-MacBook-Pro-2 assignment3 %
```