Quiz 4 Solutions

CSE 4/574

Fall, 2024

Question 1

Consider a classification problem where we are building a logistic regression classifier. The task is to predict if a given city has a risk of a disease epidemic or not. The data is defined using two input features or variables -
logarithm of size of the city
distance to the nearest city with epidemic

The target is a binary (0 - no risk and 1 - risk). Consider the following training data set:

City #	Log Size of City	Distance	Risk?
1	2.18	0.01	1
2	1.09	18.30	1
3	−0.37	3.00	0
4	0.00	4.10	1
5	−0.42	9.00	0
6	0.09	7.20	0
7	−0.48	10.00	0
8	0.62	2.70	0
9	0.57	2.80	1
10	0.44	0.01	1

Assume that a bias term is added in the beginning. Consider the following weight vector for logistic regression: $w_0 = 1.05$, $w_1 = -0.52$, $w_2 = 0.85$. Answer the following:

Solution Details

Sample Code for creating logistic regression mode and making predictions:

The model predictions (0.5 as threshold): 0 1 1 1 1 1 1 1 1 1
The ground truths are: 1 1 0 1 0 0 0 0 1 1

```python
import numpy as np

def sigmoid(x):
  return 1 / (1 + np.exp(-x))

def logistic_regression_epidemic_risk(log_city_size, distance_to_epidemic_city):
  """
  Predicts the risk of a disease epidemic in a city using logistic regression.

  Args:
    log_city_size: The logarithm of the city's size.
    distance_to_epidemic_city: The distance to the nearest city with an epidemic.

  Returns:
    The probability of an epidemic risk in the city.
  """

  w0 = 1.05
  w1 = -0.52
  w2 = 0.85

  z = w0 + w1 * log_city_size + w2 * distance_to_epidemic_city
  probability = sigmoid(z)

  return probability

# Example usage
log_city_size = [2.18,1.09,-0.37,0,-0.42,0.09,-0.48,0.62,0.57,0.44]  # Example log city size
distance_to_epidemic_city = [0.01,18.3,3,4.1,9,7.2,10,2.7,2.8,0.01]  # Example distance to epidemic city

for i in range(len(log_city_size)):
  risk_probability = logistic_regression_epidemic_risk(log_city_size[i], distance_to_epidemic_city[i])
  print("Risk probability:", risk_probability)
```

```
Risk probability: 0.4812338192915899
Risk probability: 0.9999998916826085
Risk probability: 0.9779554777881293
Risk probability: 0.9893869381038362
Risk probability: 0.9998661156102346
Risk probability: 0.999194466573228
Risk probability: 0.9999445292926614
Risk probability: 0.9535847395931248
Risk probability: 0.9582576305377485
Risk probability: 0.6962914926649901
```

Correct Choice

The classifier makes 6 mistakes.

Choice Explanation: Comparing the predictions and ground truths, there are 6 mistakes.

Incorrect Choice 1

The classifier makes 4 mistakes.

Choice Explanation:
Comparing the predictions and ground truths, there are 6 mistakes.

Incorrect Choice 2

The classifier's worst wrong prediction is for city 9.

Choice Explanation:
City 9 was correctly predicted.

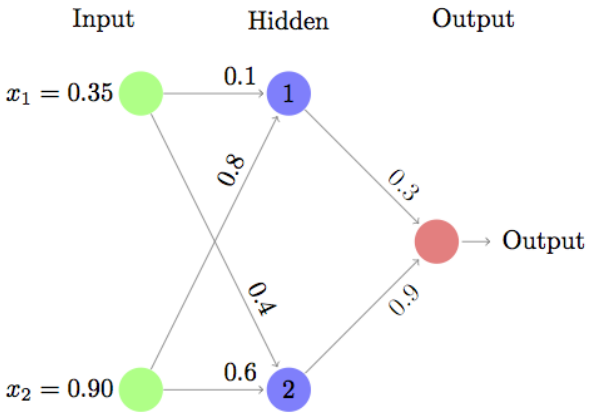
Incorrect Choice 3

The classifier's most confident correct prediction is for city 3.

Choice Explanation:
Referring to the risk probabilities, we see that city 2 has the most confident prediction.

Question 2

Consider the following simple neural network with only one output node. Ignore the bias node for this example. The values on the edges indicate the weights associated with the "receiving node".



Assume that the neurons have a Sigmoid activation function. And we use a squared loss for the loss function $\frac{1}{2}(y-o)^2$, where y represents the true value, o represents the output from the neural network. You can use a small snippet of Python code to compute sigmoid activation s for a value z.
import numpy as np
s = 1.0 / (1.0 + np.exp(-1.0 * z))

Or you can use a scientific calculator, MATLAB, etc.

Please do the following:
Perform a forward pass on the network. Perform a backward pass on the network. Let true input be y = 0.5. Update the weights at the output and hidden nodes using learning rate =1
Choose the option which is correct below:

Correct Choice

The final output after the forward pass is 0.690.

Incorrect Choice 1

The final output after the forward pass is 0.664.

Incorrect Choice 2

One of the new weights for the output layer after gradient descent update will be 0.927.

Incorrect Choice 3

One of the new weights for the output layer after gradient descent update will be 0.328.

Problem Explaination:
Sample code:

```python
import numpy as np

def sigmoid(x):
  return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
  return x * (1 - x)

def build_neural_network_with_gradient_descent(input_data, weights_input_hidden, weights_hidden_output, y_true, learning_rate):
  # Forward pass
  hidden_layer_input = np.dot(input_data, weights_input_hidden)
  hidden_layer_output = sigmoid(hidden_layer_input)
  output_layer_input = np.dot(hidden_layer_output, weights_hidden_output)
  output_layer_output = sigmoid(output_layer_input)

  # Calculate loss
  loss = calculate_loss(output_layer_output, y_true)

  # Backward pass
  output_error = (output_layer_output - y_true) * sigmoid_derivative(output_layer_output)
  hidden_error = output_error.dot(weights_hidden_output.T) * sigmoid_derivative(hidden_layer_output)

  # Update weights
  weights_hidden_output -= hidden_layer_output.T.dot(output_error) * learning_rate
  weights_input_hidden -= input_data.T.dot(hidden_error) * learning_rate

  return output_layer_output, loss, weights_input_hidden, weights_hidden_output


def calculate_loss(y_pred, y_true):
  return np.mean(((y_pred - y_true)**2)*0.5)

# Usage
input_data = np.array([[0.35, 0.9]])
weights_input_hidden = np.array([[0.1, 0.4], [0.8, 0.6]])
weights_hidden_output = np.array([[0.3], [0.9]])
y_true = np.array([[0.5]])
learning_rate = 1

output, loss, updated_weights_input_hidden, updated_weights_hidden_output = build_neural_network_with_gradient_descent(
    input_data, weights_input_hidden, weights_hidden_output, y_true, learning_rate)

print("Output:", output)
print("Loss:", loss)
print("Updated weights input hidden:", updated_weights_input_hidden)
print("Updated weights hidden output:", updated_weights_hidden_output)
```

```
Output: [[0.69028349]]
Loss: 0.01810390383656677
Updated weights input hidden: [[0.09907093 0.39713992]
 [0.79761096 0.59264552]]
Updated weights hidden output: [[0.27232597]
 [0.87299836]]
```