HW4_Final_code

September 30, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import random
%matplotlib inline
```

[2]:
```
from  sklearn.datasets import  load_iris
iris = load_iris()
list(iris.keys())
```

[2]:
```
['data',
 'target',
 'frame',
 'target_names',
 'DESCR',
 'feature_names',
 'filename',
 'data_module']
```

[3]:
```
X = iris.data[:,:2]
y = iris.target
```

[4]:
```
random.seed(64)
sample_size = random.sample( range(0,149),5)

X_train = X[sample_size]
y_train = y[sample_size]
```

[5]:
```
random.seed(64)
sample_size = random.sample( range(0,149),5)

X_train = X[sample_size]
y_train = y[sample_size]
```

[6]:
```
class Parameters:

    def __init__(self, inputs, neurons):
```

```
        self.weights = (np.random.randn(neurons, inputs))
        self.bias = (np.random.randn(neurons, 1))

class Model:

    def forward_prop(self, inputs, weights, bias):
        self.output = np.dot(weights, inputs) + bias

    def cost_func(self, predicted, y):
        #final_value = np.sum(predicted*y, axis = 0)
        self.loss_func = np.mean(-np.log(predicted) )

    def back_prop(self, pred, lr, w, b, y, X):

        n = len(y)
        dw = 1/n * np.dot((pred - y), X)
        db = 1/n* np.sum(pred - y)

        w = w - lr * dw
        b = b - lr * db
        self.new_weights = w
        self.new_bias = b
```

[7]:
```
class Activation:

    def __init__(self, layer_output):
        self.sigmoid = 1 / (1 + np.exp(-layer_output))

        e = np.exp(layer_output - np.max(layer_output, axis = 1,␣
↪keepdims=True)) #To avoid overflow subtract with max value
        self.softmax = e / np.sum(e, axis = 1, keepdims=True)
        self.predicted = np.max(self.softmax, axis = 0,keepdims=False)
```

[8]:
```
def main(rounds, n1, n2):

    #Define Parameters for input layer
    paramters_input = Parameters(X_train.shape[1], n1)

    w1 = paramters_input.weights
    b1 = paramters_input.bias

    #Define Parameters for hidden layer
    paramters_hidden = Parameters(3, n2)

    w2 = paramters_hidden.weights
```

```python
    b2 = paramters_hidden.bias

    for i in range(rounds):


        # Define class object for input layer with 3 neurons
        input_layer = Model()
        input_layer.forward_prop(X_train.T, w1, b1)
        input_layer_out = (input_layer.output)

        # Active Neuron using sigmoid
        input_layer_pred = Activation(input_layer_out)
        final_input_out = input_layer_pred.sigmoid

        # Define class object for hidden layer 1 with 3 neurons
        hidden_layer1 = Model()
        hidden_layer1.forward_prop(final_input_out, w2, b2)

        # Active Neuron using Softmax
        hidden_layer1_pred = Activation(hidden_layer1.output)
        final_hidden_out = hidden_layer1_pred.softmax
        final_prediction = hidden_layer1_pred.predicted
        hidden_layer1.cost_func(final_prediction, y_train)

        print(f"############# Round {i+1} #############")
        accuracy = (np.mean(np.argmax(final_hidden_out, axis = 0) == y_train) * 100)
        print("Accuracy is {}%".format(accuracy))
        print("Cross Entropy Loss is {}".format(hidden_layer1.loss_func))
        print("\n")

        #Backpopgation

        hidden_layer1.back_prop(final_hidden_out, 0.5,
                                w2, b2, y_train, final_input_out.T)

        input_layer.back_prop(final_input_out, 0.5,
                              w1, b1, y_train, X_train)

        w1 = input_layer.new_weights
        b1 = input_layer.new_bias

        w2 = hidden_layer1.new_weights
        b2 = hidden_layer1.new_bias


main(rounds = 2, n1 = 3, n2 = 3)
```

```
############# Round 1 #############
Accuracy is 40.0%
Cross Entropy Loss is 1.5936634209710259


############# Round 2 #############
Accuracy is 80.0%
Cross Entropy Loss is 1.6094150957638855
```

[9]:
```
# bonus question
```

[10]:
```
X = iris.data[:,:4]
y = iris.target
```

[11]:
```
random.seed(64)
sample_size = random.sample( range(0,149),5)

X_train = X[sample_size]
y_train = y[sample_size]
```

[12]:
```
random.seed(64)
sample_size = random.sample( range(0,149),5)

X_train = X[sample_size]
y_train = y[sample_size]
```

[13]:
```
main(rounds = 2, n1 = 3, n2 = 5)
```

```
############# Round 1 #############
Accuracy is 60.0%
Cross Entropy Loss is 1.4523396673498215


############# Round 2 #############
Accuracy is 40.0%
Cross Entropy Loss is 1.6094352280211592
```

[ ]:
```
```

[ ]:
```
```