mlp

December 3, 2021

0.0.1 MT21MCS013 Jay - MLP

[1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

data = pd.read_csv('train.csv')
from sklearn.metrics import classification_report
```

[3]:
```
data = np.array(data)
m, n = data.shape
np.random.shuffle(data) # shuffle before splitting into dev and training sets
print(data.shape)
data_dev = data[0:1000].T
Y_dev = data_dev[0]
X_dev = data_dev[1:n]
X_dev = X_dev / 255.

data_train = data[1000:m].T
Y_train = data_train[0]
X_train = data_train[1:n]
X_train = X_train / 255.
_,m_train = X_train.shape
```

```
(42000, 785)
```

[3]:
```
def init_params():
    W1 = np.random.rand(10, 784) - 0.5
    b1 = np.random.rand(10, 1) - 0.5
    W2 = np.random.rand(10, 10) - 0.5
    b2 = np.random.rand(10, 1) - 0.5
    return W1, b1, W2, b2

def ReLU(Z):
    return np.maximum(Z, 0)

def softmax(Z):
    A = np.exp(Z) / sum(np.exp(Z))
```

```
    return A

def forward_prop(W1, b1, W2, b2, X):
    Z1 = W1.dot(X) + b1
    A1 = ReLU(Z1)
    Z2 = W2.dot(A1) + b2
    A2 = softmax(Z2)
    return Z1, A1, Z2, A2

def ReLU_deriv(Z):
    return Z > 0

def one_hot(Y):
    one_hot_Y = np.zeros((Y.size, Y.max() + 1))
    one_hot_Y[np.arange(Y.size), Y] = 1
    one_hot_Y = one_hot_Y.T
    return one_hot_Y

def backward_prop(Z1, A1, Z2, A2, W1, W2, X, Y):
    one_hot_Y = one_hot(Y)
    dZ2 = A2 - one_hot_Y
    dW2 = 1 / m * dZ2.dot(A1.T)
    db2 = 1 / m * np.sum(dZ2)
    dZ1 = W2.T.dot(dZ2) * ReLU_deriv(Z1)
    dW1 = 1 / m * dZ1.dot(X.T)
    db1 = 1 / m * np.sum(dZ1)
    return dW1, db1, dW2, db2

def update_params(W1, b1, W2, b2, dW1, db1, dW2, db2, alpha):
    W1 = W1 - alpha * dW1
    b1 = b1 - alpha * db1
    W2 = W2 - alpha * dW2
    b2 = b2 - alpha * db2
    return W1, b1, W2, b2
```

[14]:
```
def get_predictions(A2):
    return np.argmax(A2, 0)

def get_accuracy(predictions, Y):
    print(predictions, Y)
    return np.sum(predictions == Y) / Y.size

def gradient_descent(X, Y, alpha, iterations):
    W1, b1, W2, b2 = init_params()
    for i in range(iterations):
        Z1, A1, Z2, A2 = forward_prop(W1, b1, W2, b2, X)
        dW1, db1, dW2, db2 = backward_prop(Z1, A1, Z2, A2, W1, W2, X, Y)
```

```
        W1, b1, W2, b2 = update_params(W1, b1, W2, b2, dW1, db1, dW2, db2,␣
↪alpha)
        if i % 10 == 0:
#             print("Iteration: ", i)
            predictions = get_predictions(A2)
#             print(get_accuracy(predictions, Y))
    print(classification_report(Y,get_predictions(A2)))
    return W1, b1, W2, b2
```

[16]:
```
W1, b1, W2, b2 = gradient_descent(X_train, Y_train, 0.10, 1000)
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94      4042
           1       0.94      0.97      0.95      4567
           2       0.89      0.86      0.87      4082
           3       0.85      0.85      0.85      4265
           4       0.88      0.88      0.88      3969
           5       0.84      0.82      0.83      3711
           6       0.90      0.92      0.91      4033
           7       0.90      0.89      0.90      4298
           8       0.85      0.83      0.84      3965
           9       0.82      0.84      0.83      4068

    accuracy                           0.88     41000
   macro avg       0.88      0.88      0.88     41000
weighted avg       0.88      0.88      0.88     41000
```

[ ]:
```

```