ASSIGNMENT: 1

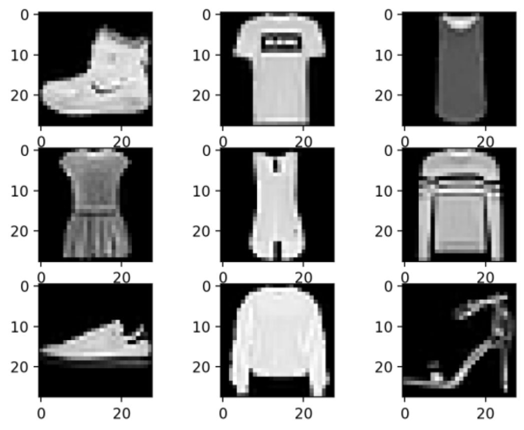
Assignment Document

Scenario 1

DESCRIPTION

Problem Statement:

The dataset is similar to MNIST, but includes images of certain clothing and accessory. The objective is to classify images into specific classes using single layer perceptron and multilayer perceptron.



Dataset:

Total Images :- 70,000

Train Images :- 60,000

Test Images :- 10,000

Image Size :- 28 X 28

Different Classes:

Classes: 'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle Boot'

Note: Please use google collab to work on this project. Also, make sure to select GPU backend while selecting a runtime.

https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d

Question 1:

A) Reshape and Normalize the data:

- reshape the input image
    - train; 60000,28 * 28 * 1
    - test; 10000, 28 * 28*1
- change the type of data to float32
- normalize the data by dividing with 255

B) Convert the y_train and y_test to categorical by using keras to_categorical function and define num_classes=10

HINT:-

```
# Loading data from Keras library
fashion_mnist = keras.datasets.fashion_mnist
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
print ("The shape of data for train", X_train.shape)
```

```
The shape of data for train (60000, 28, 28)
```

Question 2:

Write a multi layer perceptron using tensorflow for fashion MNIST data to classify all 10 classes. Solve for each questions:

Hint:

```
#define the important parameters and variable to work with the tensors
learning_rate = 0.001
training_epochs = 100
cost_history = np.empty(shape=[1], dtype=float)
n_dim = X_train.shape[1]
n_class = 10

#define the number of hidden layers and number of neurons for each layer
n_hidden_1 = 256
n_hidden_2 = 256
n_hidden_3 = 256
n_hidden_4 = 256

x = tf.placeholder(tf.float32, [None, n_dim])
W = tf.Variable(tf.zeros([n_dim, n_class]))
b = tf.Variable(tf.zeros([n_class]))
y_ = tf.placeholder(tf.float32, [None, n_class])
```

Take epoch= 2000

A) Define a function that creates 4 layers using TensorFlow layers:

HINT: -

```
# Hidden layer with RELU activationsd
layer_1 = tf.add(tf.matmul(x, weights['h1']), biases['b1'])
layer_1 = tf.nn.sigmoid(layer_1)
```

B) Define a function that creates 4 layers using TensorFlow layers:

HINT:-

i. 'h1': tf.Variable(tf.truncated_normal([n_dim, n_hidden_1])),
ii. 'b1': tf.Variable(tf.truncated_normal([n_hidden_1])),

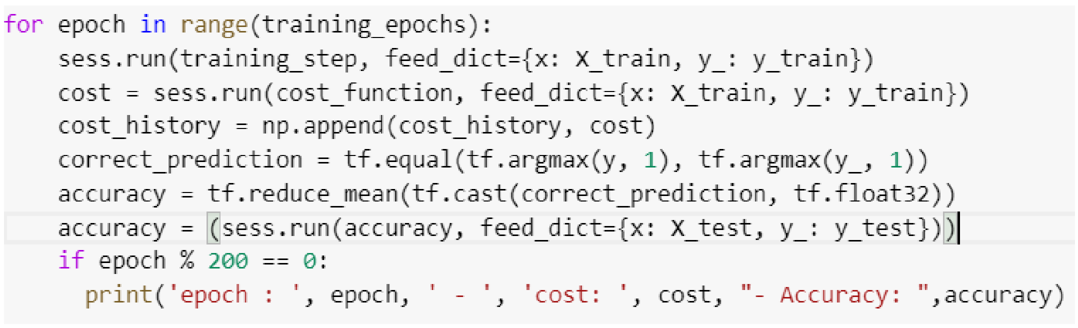
C) Call the function:

y = multilayer_perceptron(x, weights, biases)

D) Define Cost function and Optimizer:

i. Cost: tf.nn.softmax_cross_entropy_with_logits
ii. Optimizer: tf.train.GradientDescentOptimizer

Initiate the training:



```
for epoch in range(training_epochs):
    sess.run(training_step, feed_dict={x: X_train, y_: y_train})
    cost = sess.run(cost_function, feed_dict={x: X_train, y_: y_train})
    cost_history = np.append(cost_history, cost)
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    accuracy = (sess.run(accuracy, feed_dict={x: X_test, y_: y_test}))
    if epoch % 200 == 0:
      print('epoch : ', epoch, ' - ', 'cost: ', cost, "- Accuracy: ",accuracy)
```

Expected Output:

```
epoch:  0 - cost:  192.05133 - Accuracy:  0.1558
epoch:  200 - cost:  17.27994 - Accuracy:  0.3904
epoch:  400 - cost:  11.775045 - Accuracy:  0.488
epoch:  600 - cost:  9.359716 - Accuracy:  0.5298
epoch:  800  -  cost:  7.8768625 - Accuracy:  0.5556
```

Question 3:

Similarly like MLP, define a function that can create single layer network called perceptron on Fashion MNIST.

- Learning rate: 0.0001
- Epochs: 5000

Hint: Use the hints given in MLP questions. Also, make sure to correctly check for layers that are going to be input to output layer.

Question 4:

Now, initialize variables, cost function and optimizer to initiate the training.

- Adam Optimizer

```
for epoch in range(training_epochs):
    sess.run(training_step, feed_dict={x: X_train, y_: y_train})
    cost = sess.run(cost_function, feed_dict={x: X_train, y_: y_train})
    cost_history = np.append(cost_history, cost)
    correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
    accuracy = (sess.run(accuracy, feed_dict={x: X_test, y_: y_test}))
    if epoch % 200 == 0:
      print('epoch : ', epoch, ' - ', 'cost: ', cost, "- Accuracy: ",accuracy)
```

Expected Output:-

```
Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.

epoch :  0  -  cost:  12.307046 - Accuracy:  0.0841
```