Artificial Intelligence and Machine Learning Fundamentals

Activity 15: Written Digit Detection with Deep Learning

This section will discuss how deep learning improves the performance of your model. We will be assuming that your boss is not satisfied with the results you presented in previous activity and asks you to consider adding two hidden layers to your original model and determine whether new layers improve the accuracy of the model. And to ensure that it happens correctly, you will need to have knowledge of Deep Learning.

1. Execute the code of previous Activity and measure the accuracy of the model.
2. Change the neural network by adding new layers. We will combine the **relu** and **softmax** activator functions:

```
x = tf.placeholder(tf.float32, [None, 28 * 28 ])
f1 = tf.nn.relu
W1 = tf.Variable(tf.random_normal([784, 200]))
b1 = tf.Variable(tf.random_normal([200]))
layer1_out = f1(tf.add(tf.matmul(x, W1), b1))
f2 = tf.nn.softmax
W2 = tf.Variable(tf.random_normal([200, 100]))
b2 = tf.Variable(tf.random_normal([100]))
layer2_out = f2(tf.add(tf.matmul(layer1_out, W2),
b2))
f3 = tf.nn.softmax
W3 = tf.Variable(tf.random_normal([100, 10]))
b3 = tf.Variable( tf.random_normal([10]))
y = f3(tf.add(tf.matmul(layer2_out, W3), b3))
```

3. Retrain the model

```
y_true = tf.placeholder(tf.float32, [None, 10])
cross_entropy =
tf.nn.softmax_cross_entropy_with_logits_v2(
logits=y,
labels=y_true
)
cost = tf.reduce_mean(cross_entropy)
optimizer = tf.train.GradientDescentOptimizer(
learning_rate=0.5).minimize(cost)
session = tf.Session()
session.run(tf.global_variables_initializer())
iterations = 600
batch_size = 200
sample_size = len(features_train_vector)
for _ in range(iterations):
indices = random.sample(range(sample_size),
batchSize)
batch_features = [
features_train_vector[i] for i in indices
]
batch_labels = [
label_train_vector[i] for i in indices
]
min = i * batch_size
max = (i+1) * batch_size
dictionary = {
x: batch_features,
y_true: batch_labels
}
session.run(optimizer, feed_dict=dictionary)
```

4. Evaluate the model

```
label_predicted = session.run(y, feed_dict={
x: features_test_vector
})
label_predicted = [
np.argmax(label) for label in label_predicted
]
confusion_matrix(label_test, label_predicted)
```

The output is as follows:

```
array([[ 801, 11, 0, 14, 0, 0, 56, 0, 61, 37],
[ 2, 1069, 0, 22, 0, 0, 18, 0, 9, 15],
[ 276, 138, 0, 225, 0, 2, 233, 0, 105, 53],
[ 32, 32, 0, 794, 0, 0, 57, 0, 28, 67],
[ 52, 31, 0, 24, 0, 3, 301, 0, 90, 481],
[ 82, 50, 0, 228, 0, 3, 165, 0, 179, 185],
[ 71, 23, 0, 14, 0, 0, 712, 0, 67, 71],
[ 43, 85, 0, 32, 0, 3, 31, 0, 432, 402],
[ 48, 59, 0, 192, 0, 2, 45, 0, 425, 203],
[ 45, 15, 0, 34, 0, 2, 39, 0, 162, 712]],
dtype=int64)
```

5. Calculating the accuracy score.

```
accuracy_score(label_test, label_predicted)
```

The output is **0.4516**.

The accuracy did not improve.

Let's see if further runs improve the accuracy of the model.

Second run: 0.5216
Third run: 0.5418
Fourth run: 0.5567
Fifth run: 0.564
Sixth run: 0.572
Seventh run: 0.5723
Eighth run: 0.6001
Ninth run: 0.6076
Tenth run: 0.6834
Twentieth run: 0.7439
Thirtieth run: 0.7496
Fortieth run: 0.7518
Fiftieth run: 0.7536

Afterwards, we got the following results: 0.755, 0.7605, 0.7598, 0.7653

The output is as follows:

```
        array([[ 801, 11, 0, 14, 0, 0, 56, 0, 61, 37],
        [ 2, 1069, 0, 22, 0, 0, 18, 0, 9, 15],
        [ 276, 138, 0, 225, 0, 2, 233, 0, 105, 53],
        [ 32, 32, 0, 794, 0, 0, 57, 0, 28, 67],
        [ 52, 31, 0, 24, 0, 3, 301, 0, 90, 481],
        [ 82, 50, 0, 228, 0, 3, 165, 0, 179, 185],
        [ 71, 23, 0, 14, 0, 0, 712, 0, 67, 71],
        [ 43, 85, 0, 32, 0, 3, 31, 0, 432, 402],
        [ 48, 59, 0, 192, 0, 2, 45, 0, 425, 203],
        [ 45, 15, 0, 34, 0, 2, 39, 0, 162, 712]],
        dtype=int64)
```

Afterwards, we got the following results: 0.755, 0.7605, 0.7598, 0.7653

The final confusion matrix:

```
array([[ 954, 0, 2, 1, 0, 6, 8, 0, 5, 4],
[ 0, 1092, 5, 3, 0, 0, 6, 0, 27, 2],
[ 8, 3, 941, 16, 0, 2, 13, 0, 35, 14],
[ 1, 1, 15, 953, 0, 14, 2, 0, 13, 11],
[ 4, 3, 8, 0, 0, 1, 52, 0, 28, 886],
[ 8, 1, 5, 36, 0, 777, 16, 0, 31, 18],
[ 8, 1, 6, 1, 0, 6, 924, 0, 9, 3],
[ 3, 10, 126, 80, 0, 4, 0, 0, 35, 770],
[ 4, 0, 6, 10, 0, 6, 4, 0, 926, 18],
[ 4, 5, 1, 8, 0, 2, 2, 0, 18, 969]],
dtype=int64)
```

This deep neural network behaves even more chaotically than the single layer one. It took 600 iterations of 200 samples to get from an accuracy of 0.572 to 0.5723.

Not long after this iteration, we jumped from 0.6076 to 0.6834 in that number of iterations.