Artificial Intelligence and Machine Learning Fundamentals

Activity 14: Written Digit Detection

1. This section will discuss how to provide more security for the cryptocurrency traders via the detection of hand-written digits. We will be using assuming that you are a software developer at a new Cryptocurrency trader platform.

The latest security measure you are implementing requires the recognition of hand-written digits. Use the MNIST library to train a neural network to recognize digits. You can read more about this dataset on https://www.tensorflow.org/tutorials/.

2. Improve the accuracy of the model as much as possible. And to ensure that it happens correctly, you will need to complete the previous topic.
3. Load the dataset and format the input

```
import tensorflow.keras.datasets.mnist as mnist
(features_train, label_train),
(features_test, label_test) = mnist.load_data()
features_train = features_train / 255.0
features_test = features_test / 255.0
def flatten(matrix):
return [elem for row in matrix for elem in row]
features_train_vector = [
flatten(image) for image in features_train
]
features_test_vector = [
flatten(image) for image in features_test
]
import numpy as np
label_train_vector = np.zeros((label_train.size,
10))
for i, label in enumerate(label_train_vector):
label[label_train[i]] = 1
label_test_vector = np.zeros((label_test.size, 10))
for i, label in enumerate(label_test_vector):
label[label_test[i]] = 1
```

4. Set up the Tensorflow graph. Instead of the sigmoid function, we will now use the relu function.

```
import tensorflow as tf
f = tf.nn.softmax
x = tf.placeholder(tf.float32, [None, 28 * 28 ])
W = tf.Variable( tf.random_normal([784, 10]))
b = tf.Variable( tf.random_normal([10]))
y = f(tf.add(tf.matmul( x, W ), b ))
```

5. Train the model.

```
import random
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)
```

6. Test the model

```
label_predicted = session.run(classify( x ),
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([[0, 0, 223, 80, 29, 275, 372, 0, 0, 1],
[0, 915, 4, 10, 1, 13, 192, 0, 0, 0],
[0, 39, 789, 75, 63, 30, 35, 0, 1, 0],
[0, 6, 82, 750, 13, 128, 29, 0, 0, 2],
[0, 43, 16, 16, 793, 63, 49, 0, 2, 0],
[0, 22, 34, 121, 40, 593, 76, 5, 0, 1],
[0, 29, 34, 6, 44, 56, 788, 0, 0, 1],
[1, 54, 44, 123, 715, 66, 24, 1, 0, 0],
[0, 99, 167, 143, 80, 419, 61, 0, 4, 1],
[0, 30, 13, 29, 637, 238, 58, 3, 1, 0]], dtype=int64)

7. Calculate the accuracy score:

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

The output is as follows:

0.4633

8. By re-running the code segment responsible for training the data set, we can improve the accuracy:

```
for _ in range(iterations):
indices = random.sample(range(sample_size),
batch_size)
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)
```

Second run: 0.5107
Third run: 0.5276
Fourth run: 0.5683
Fifth run: 0.6002
Sixth run: 0.6803
Seventh run: 0.6989
Eighth run: 0.7074
Ninth run: 0.713
Tenth run: 0.7163
Twentieth run: 0.7308
Thirtieth run: 0.8188
Fortieth run: 0.8256
Fiftieth run: 0.8273

At the end of the fiftieth run, the improved confusion matrix looks as follows:

```
array([
[946, 0, 6, 3, 0, 1, 15, 2, 7, 0],
[ 0,1097, 3, 7, 1, 0, 4, 0, 23, 0],
[11, 3, 918, 11, 18, 0, 13, 8, 50, 0],
[3, 0, 23, 925, 2, 10, 4, 9, 34, 0],
[2, 2, 6, 1, 929, 0, 14, 2, 26, 0],
[16, 4, 7, 62, 8, 673, 22, 3, 97, 0],
[8, 2, 4, 3, 8, 8, 912, 2, 11, 0],
[5, 9, 33, 6, 9, 1, 0, 949, 16, 0],
[3, 4, 5, 12, 7, 4, 12, 3, 924, 0],
[8, 5, 7, 40, 470, 11, 5, 212, 251, 0]
],
dtype=int64)
```

Not a bad result. More than 8 out of 10 digits are accurately recognized.