Assignment #1

MNIST classifier

한양대학교 컴퓨터소프트웨어학부 2015004911 임성준
딥러닝및응용(ITE4053, 12878) - 최용석 교수님

---

1. 개요

MNIST classifier에 dropout과 adam optimizer을 적용한다.

2. 개발환경

```
OS : macOS Big Sur 11.2.2
Language : Python 3.7.10
Source code editor : Visual Studio Code
Runtime environment : Jupyter notebook
```

3. 코드 설명

ppt의 코드를 참고했습니다.

- Gradient Descent Optimizer.py

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("./mnist/data/", one_hot=True)

X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

W1 = tf.Variable(tf.random_uniform([784, 256], -1., 1.))
b1 = tf.Variable(tf.random_uniform([256], -1., 1.))
L1 = tf.nn.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.Variable(tf.random_uniform([256, 256], -1., 1.))
b2 = tf.Variable(tf.random_uniform([256], -1., 1.))
L2 = tf.nn.sigmoid(tf.matmul(L1, W2) + b2)

W3 = tf.Variable(tf.random_uniform([256, 10], -1., 1.))
b3 = tf.Variable(tf.random_uniform([10], -1., 1.))
```

```
logits = tf.matmul(L2, W3) + b3
hypothesis = tf.nn.softmax(logits)
cost =
tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(labels=Y,
logits=logits))

# Gradient Descent Optimizer
opt = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

batch_size = 100

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch in range(15):
        avg_cost = 0
        total_batch = int(mnist.train.num_examples/batch_size)
        for i in range(total_batch):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            c, _ = sess.run([cost, opt], feed_dict={X: batch_xs, Y:
batch_ys})
            avg_cost += c / total_batch
        print('Epoch: ', '%d' % (epoch+1), 'cost =',
'{:.9f}'.format(avg_cost))

    is_correct = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
    accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
    print("Accuracy", sess.run(accuracy, feed_dict={X: mnist.test.images,
Y: mnist.test.labels}))
```

- `Dropout` 없음
- `opt` : GradientDescentOptimizer를 사용한다.

- Adam Optimizer.py

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets("./mnist/data/", one_hot=True)

X = tf.placeholder(tf.float32, [None, 784])
Y = tf.placeholder(tf.float32, [None, 10])

# Dropout
keep_prob = tf.placeholder(tf.float32)
```

```
W1 = tf.Variable(tf.random_uniform([784, 256], -1., 1.))
b1 = tf.Variable(tf.random_uniform([256], -1., 1.))
L1 = tf.nn.sigmoid(tf.matmul(X, W1) + b1)
L1 = tf.nn.dropout(L1, keep_prob)

W2 = tf.Variable(tf.random_uniform([256, 256], -1., 1.))
b2 = tf.Variable(tf.random_uniform([256], -1., 1.))
L2 = tf.nn.sigmoid(tf.matmul(L1, W2) + b2)
L2 = tf.nn.dropout(L2, keep_prob)

W3 = tf.Variable(tf.random_uniform([256, 10], -1., 1.))
b3 = tf.Variable(tf.random_uniform([10], -1., 1.))
logits = tf.matmul(L2, W3) + b3

hypothesis = tf.nn.softmax(logits)
cost =
tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(labels=Y,
logits=logits))

# Adam Optimizer
opt = tf.train.AdamOptimizer(
    learning_rate=0.001,
    beta1=0.9,
    beta2=0.999,
    epsilon=1e-08,
    use_locking=False,
    name='Adam').minimize(cost)

batch_size = 100

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for epoch in range(15):
        avg_cost = 0
        total_batch = int(mnist.train.num_examples/batch_size)
        for i in range(total_batch):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            c, _ = sess.run([cost, opt], feed_dict={X: batch_xs, Y:
batch_ys, keep_prob: 0.75})
            avg_cost += c / total_batch
        print('Epoch: ', '%d' % (epoch+1), 'cost =',
'{:.9f}'.format(avg_cost))

    is_correct = tf.equal(tf.argmax(hypothesis, 1), tf.argmax(Y, 1))
    accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

```
    print("Accuracy", sess.run(accuracy, feed_dict={X: mnist.test.images,
Y: mnist.test.labels, keep_prob: 1}))
```

- GD 코드에서 `Dropout`을 첫번째와 두번째 layerdp 적용.
- `opt` : GD가 아닌 adam optimizer를 사용한다.

4. 실행 결과

- **`Gradient Descent Optimizer.py`**

```
Epoch:  1 cost = 0.932951244
Epoch:  2 cost = 0.439127001
Epoch:  3 cost = 0.355977076
Epoch:  4 cost = 0.310988340
Epoch:  5 cost = 0.279746349
Epoch:  6 cost = 0.256110537
Epoch:  7 cost = 0.237630512
Epoch:  8 cost = 0.222235265
Epoch:  9 cost = 0.209309288
Epoch:  10 cost = 0.198324174
Epoch:  11 cost = 0.187842718
Epoch:  12 cost = 0.179427245
Epoch:  13 cost = 0.171128712
Epoch:  14 cost = 0.164215577
Epoch:  15 cost = 0.157797423
Accuracy 0.9375
```

`Gradient Descent Optimizer 를 사용하고, Dropout 은 적용하지 않은 프로그램의 결과값이다.`
`약 93%의 정확도를 보여준다.`

- **`Adam Optimizer.py`**

```
Epoch:  1 cost = 2.466165378
Epoch:  2 cost = 0.887344141
Epoch:  3 cost = 0.637736198
Epoch:  4 cost = 0.490708571
Epoch:  5 cost = 0.403325002
Epoch:  6 cost = 0.334114783
Epoch:  7 cost = 0.292240168
Epoch:  8 cost = 0.259041767
Epoch:  9 cost = 0.231581055
Epoch:  10 cost = 0.213874695
Epoch:  11 cost = 0.188580029
Epoch:  12 cost = 0.172168033
Epoch:  13 cost = 0.159294778
Epoch:  14 cost = 0.146381579
Epoch:  15 cost = 0.141470343
Accuracy 0.9682
```

```
Adam Optimizer 를 사용하고, Dropout 도 적용한 프로그램의 결과값이다.
약 96%의 정확도를 보여준다.
```

결론 : Adam optimizer와 dropout을 적용하기 전보다 더 높은 정확도를 보여주는 것을 알 수 있다.