Assignment #3

Check Point

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

---

1. 개요

Checkpoint를 사용하여 나오는 output의 값을 비교한다.

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

- 9.CheckPoint.py

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

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

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

# loss function and optimizer

## Random uniform initializer
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))
opt = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

batch_size = 100

# Check Point path
ckpt_path = "model/trained"

## Save
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    saver = tf.train.Saver()

    # training
    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}))

    saver.save(sess, ckpt_path)
    saver.restore(sess, ckpt_path)
    print("Accuracy", sess.run(accuracy, feed_dict={X: mnist.test.images, Y:
mnist.test.labels}))
```

- 사용할 데이터로는 MNIST 데이터를 사용한다.
- Random Uniform Initializer를 사용한다.
- Fully-Connected layer로 학습하며 softmax 함수를 통해 예측한다.
- Layer는 총 3계층, loss function으로는 cross-entropy function을 사용하고, GradientDescentOptimizer를 사용한다.
- Epoch를 총 15회 학습하고, ckpt_path에 checkpoint를 저장한다. 후에 정확도를 확인한다.

- ckpt_path에 저장된 파일을 불러와 다시 정확도를 확인한다.

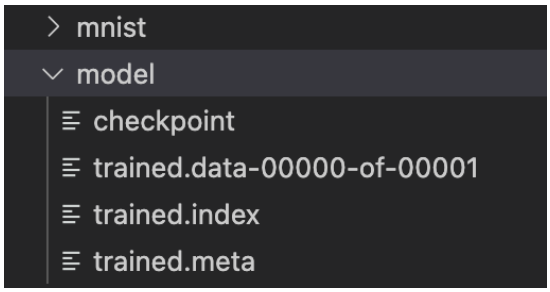
4. 실행 결과

- `9.CheckPoint.py`

```
Epoch:  1 cost = 0.938352898
Epoch:  2 cost = 0.435252389
Epoch:  3 cost = 0.350443836
Epoch:  4 cost = 0.304104505
Epoch:  5 cost = 0.273035085
Epoch:  6 cost = 0.249811470
Epoch:  7 cost = 0.231276927
Epoch:  8 cost = 0.216467966
Epoch:  9 cost = 0.203454141
Epoch:  10 cost = 0.191973586
Epoch:  11 cost = 0.182704784
Epoch:  12 cost = 0.173415969
Epoch:  13 cost = 0.165510685
Epoch:  14 cost = 0.158278988
Epoch:  15 cost = 0.152002619
Accuracy 0.9413
INFO:tensorflow:Restoring parameters from model/trained
Accuracy 0.9413
```

- Epoch가 진행될 수록 cost가 줄어든다.
- 결과적으로 약 94%의 정확도를 보인다.
- saver.save()를 통해 모델을 저장하고, saver.restore()를 통해 복구한 뒤 실행해보니 정확히 같은 값의 정확도가 나타난다.

- `checkpoint file`



- 학습된 모델의 weight 값을 저장한다.
- save/restore 시에 이 파일을 불러와 학습된 모델을 재사용한다.