Checkpoint 보고서

컴퓨터소프트웨어학부
2016025996
조성진

Assignment3는 checkpoint를 작성하고 load & store해보는 과제이다.

```
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.get_variable("w1", shape=[784,256], initializer=tf.contrib.layers.xavier_initializer())
b1 = tf.get_variable("b1", shape=[256], initializer=tf.contrib.layers.xavier_initializer())
L1 = tf.sigmoid(tf.matmul(X, W1) + b1)

W2 = tf.get_variable("w2", shape=[256,256], initializer=tf.contrib.layers.xavier_initializer())
b2 = tf.get_variable("b2", shape=[256], initializer=tf.contrib.layers.xavier_initializer())
L2 = tf.sigmoid(tf.matmul(L1, W2) + b1)

W3 = tf.get_variable("w3", shape=[256,10], initializer=tf.contrib.layers.xavier_initializer())
b3 = tf.get_variable("b3", shape=[10], initializer=tf.contrib.layers.xavier_initializer())
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(0.1).minimize(cost)

batch_size = 100

ckpt_path = "./model/checkpoint.ckpt"

with tf.Session() as sess:
    saver = tf.train.Saver()
    sess.run(tf.global_variables_initializer())
    saver.restore(sess, ckpt_path)
    '''
    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("Gradient Descent Optimizer without dropout Accuracy", sess.run(accuracy, feed_dict={X:mnist.test.images, Y:mnist.test.labels}))

    saver.save(sess, ckpt_path)
```

전체 코드는 다음과 같다. 우선 이번주 실습 강의에서 배운대로 random_uniform 방식으로 variable initialize방식을 xavier initializer를 사용하여 바꿨다. 그리고 저장할 checkpoint 경로를 ./model/checkpoint.ckpt로 지정하였다. 해당 코드는 현재 학습한 후 restore하는 코드로 학습하는 부분이 주석처리 되어있다. Saver이라는 체크포인트 저장할때 사용하는 변수를 선언하였다.

저장후 불러오기 단계를 한단계씩 보면

```
with tf.Session() as sess:
    saver = tf.train.Saver()
    sess.run(tf.global_variables_initializer())
    # saver.restore(sess, ckpt_path)
    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("Gradient Descent Optimizer without dropout Accuracy", sess.run(accuracy, feed_dict={X:mnist.test.images, Y:mnist.test.
+++ labels}))

    saver.save(sess, ckpt_path)
```

모델을 저장하는 코드는 다음과 같다. 처음에는 불러올 모델이 없으므로 restore부분을 주석처리 해주었다. 그리고 학습이 끝난 후 saver.save(sess, ckpt_path)를 한다.

```
Epoch: 1 cost = 1.567205059
Epoch: 2 cost = 0.563195192
Epoch: 3 cost = 0.410527270
Epoch: 4 cost = 0.359643044
Epoch: 5 cost = 0.331825657
Epoch: 6 cost = 0.312969850
Epoch: 7 cost = 0.298542852
Epoch: 8 cost = 0.285475733
Epoch: 9 cost = 0.275210558
Epoch: 10 cost = 0.265434111
Epoch: 11 cost = 0.255431150
Epoch: 12 cost = 0.246731867
Epoch: 13 cost = 0.237854360
Epoch: 14 cost = 0.229767187
Epoch: 15 cost = 0.221186335
Gradient Descent Optimizer without dropout Accuracy 0.9367
```

다음은 해당 코드를 실행시킨 결과이다.

```
~/DeepLearning/assignment3/model on git master !1 ?1
❯ ls
checkpoint                                checkpoint.ckpt.index
checkpoint.ckpt.data-00000-of-00001 checkpoint.ckpt.meta
```

이렇게 체크포인트 파일들도 지정한 경로에 생긴것을 확인할 수 있다.

```
with tf.Session() as sess:
    saver = tf.train.Saver()
    sess.run(tf.global_variables_initializer())
    saver.restore(sess, ckpt_path)
    '''
    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("Gradient Descent Optimizer without dropout Accuracy", sess.run(accuracy, feed_dict={X:mnist.test.images, Y:mnist.test.
+++ labels}))

    saver.save(sess, ckpt_path)
```

다음은 위에 전체코드에 나와있는것과 같이 학습과정을 주석처리 한 코드이다. Saver.restore(sess, ckpt_path) 로 저장된 모델을 불러올 수 있고 해당 코드를 실행시켰을때 저장된 모델이 load된다면 위와 같은 accuracy가 나올것이다.

```
Instructions for updating:
Use standard file APIs to check for files with this prefix.
Gradient Descent Optimizer without dropout Accuracy 0.9367
```

나온 바와 같이 학습과정은 없고 바로 모델의 정확도가 위와 같음을 확인

할 수 있었다. 이번 과제에서 이렇게 모델의 save & restore을 각각 한번씩 해보고, initializer도 xavier initializer로 바꿔보았다.