MNIST Classifier 보고서

컴퓨터소프트웨어학부

2016025996

조성진

MNIST Classifier의 정확도를 높이는 방법으로 Gradient Descent Optimizer에서 Adam Optimizer을 사용하고, Dropout까지 사용하는 방법을 실험해보았다. 우선 결과적으로 Adam Optimizer와 Dropout을 적용한 소스코드는 다음과 같다.

```
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])
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.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.sigmoid(tf.matmul(L1, W2) + b1)
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))
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("AdamOptimizer with dropout Accuracy", sess.run(accuracy, feed_dict={X:mnist.test.images, Y:mnist.test.labels, keep_prob:
+++ 1}))
```

우선 mnist dataset을 onehot encoding방식으로 가져온다. 그리고 이미지의 크기가 28x28이므로 X 를 [None, 784]로 reshape시켜 None에는 데이터의 갯수가 들어가게 한다. Y는 0부터 9까지를 판단하는것이므로 마찬가지고 [None, 10]으로 설정한다. Keep_prob은 dropout의 rate를 결정하는 것으로 0.8이면 80%의 노드가 계산에 참여한다는 의미이다. 3 Layer로 만들었으며, Softmax를 활용하여 cost function을 정의하였다. 실제 학습 과정에서 dropout의 정도를 75퍼센트의 노드가 학습에 참여하도록 설정하였고, 실제 테스트시 모든 노드가 계산에 참여하도록 설정하였다.

단계별로 4단계에 나눠서 정확도를 측정해보았다.

1. Gradient Desecent without Dropout

```
Epoch: 1 cost = 0.892350501
Epoch: 2 cost = 0.440936436
Epoch: 3 cost = 0.358827192
Epoch: 4 cost = 0.313688144
Epoch: 5 cost = 0.282312231
Epoch: 6 cost = 0.258939710
Epoch: 7 cost = 0.239403321
Epoch: 8 cost = 0.224221227
Epoch: 9 cost = 0.210872863
Epoch: 10 cost = 0.199189265
Epoch: 11 cost = 0.188791566
Epoch: 12 cost = 0.179362233
Epoch: 13 cost = 0.171396273
Epoch: 14 cost = 0.163755979
Epoch: 15 cost = 0.156958546
Gradient Descent Optimizer without dropout Accuracy 0.9418
```

2. Gradient Descent with Dropout

```
Epoch: 1 cost = 2.477562950
Epoch: 2 cost = 1.048745017
Epoch: 3 cost = 0.797257242
Epoch: 4 cost = 0.673317288
Epoch: 5 cost = 0.594336745
Epoch: 6 cost = 0.556587153
Epoch: 7 cost = 0.520681180
Epoch: 8 cost = 0.496563695
Epoch: 9 cost = 0.471160038
Epoch: 10 cost = 0.454649786
Epoch: 11 cost = 0.437919404
Epoch: 12 cost = 0.424705436
Epoch: 13 cost = 0.409638864
Epoch: 14 cost = 0.401276407
Epoch: 15 cost = 0.392515749
GradientDescentOptimizer with dropout Accuracy 0.9296
```

3. Adam without Dropout

```
Epoch: 1 cost = 0.683638918
Epoch: 2 cost = 0.233097939
Epoch: 3 cost = 0.161020875
Epoch: 4 cost = 0.118217868
Epoch: 5 cost = 0.090033272
Epoch: 6 cost = 0.069167798
Epoch: 7 cost = 0.052845907
Epoch: 8 cost = 0.040471172
Epoch: 9 cost = 0.031075576
Epoch: 10 cost = 0.023271651
Epoch: 11 cost = 0.017780950
Epoch: 12 cost = 0.013315898
Epoch: 13 cost = 0.009798160
Epoch: 14 cost = 0.006876620
Epoch: 15 cost = 0.005356720
AdamOptimizer without dropout Accuracy 0.9686
```

4. Adam with Dropout

```
Epoch: 1 cost = 2.545025762
Epoch: 2 cost = 0.895673696
Epoch: 3 cost = 0.639127929
Epoch: 4 cost = 0.487988961
Epoch: 5 cost = 0.399608218
Epoch: 6 cost = 0.333538633
Epoch: 7 cost = 0.293240752
Epoch: 8 cost = 0.253047765
Epoch: 9 cost = 0.232500749
Epoch: 10 cost = 0.209272902
Epoch: 11 cost = 0.187411715
Epoch: 12 cost = 0.174105939
Epoch: 13 cost = 0.157739318
Epoch: 14 cost = 0.146478296
Epoch: 15 cost = 0.134956823
AdamOptimizer with dropout Accuracy 0.9698
```

결과를 보면 Adam optimizer을 사용했을때 정확도가 매우 많이 올라가고, Gradient descent optimizer을 사용했을때는 dropout을 사용했을때 오히려 정확도가 떨어지는 경우도 볼 수 있었고, Adam에 dropout을 사용시 정확도가 올라갔다. 해당 결과도 여러번 실행하면 바뀔 수 있고, Dropout rate나 Layer의 갯수, learning rate등에 따라서도 결과가 달라지겠지만 테스트 상황에서 Adam optimizer과 dropout을 함께 사용했을때 가장 정확도가 높은걸 확인할 수 있었다.