【机器学习】AlexNet 的tensorflow 实现

2016-04-05 神殿栋梁 阅 24

AlexNet 的tensorflow 实现

```
# 输入数据
import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot=True)

import tensorflow as tf

# 定义网络超参数
learning_rate = 0.001
training_iters = 200000
batch_size = 64
display_step = 20

# 定义网络参数
n_input = 784 # 输入的维度
n_classes = 10 # 标签的维度
dropout = 0.8 # Dropout 的概率

# 占位符输入
x = tf.placeholder(tf.types.float32, [None, n_input])
y = tf.placeholder(tf.types.float32, [None, n_classes])
keep_prob = tf.placeholder(tf.types.float32)

# 卷积操作
def conv2d(name, l_input, w, b):
    return tf.nn.relu(tf.nn.bias_add(tf.nn.conv2d(l_input, w, strides=[1, 1, 1, 1], padding='SAME'),b), name=name)

# 最大下采样操作
def max_pool(name, l_input, k):
    return tf.nn.max_pool(l_input, ksize=[1, k, k, 1], strides=[1, k, k, 1], padding='SAME', name=name)

# 归一化操作
def norm(name, l_input, lsize=4):
    return tf.nn.lrn(l_input, lsize, bias=1.0, alpha=0.001 / 9.0, beta=0.75, name=name)

# 定义整个网络
def alex_net(_X, _weights, _biases, _dropout):
    # 向量转为矩阵
    _X = tf.reshape(_X, shape=[-1, 28, 28, 1])

    # 卷积层
    conv1 = conv2d('conv1', _X, _weights['wc1'], _biases['bc1'])
    # 下采样层
    pool1 = max_pool('pool1', conv1, k=2)
    # 归一化层
    norm1 = norm('norm1', pool1, lsize=4)
    # Dropout
    norm1 = tf.nn.dropout(norm1, _dropout)

    # 卷积
    conv2 = conv2d('conv2', norm1, _weights['wc2'], _biases['bc2'])
    # 下采样
    pool2 = max_pool('pool2', conv2, k=2)
    # 归一化
    norm2 = norm('norm2', pool2, lsize=4)
    # Dropout
    norm2 = tf.nn.dropout(norm2, _dropout)
```

```
    # 卷积
    conv3 = conv2d('conv3', norm2, _weights['wc3'], _biases['bc3'])
    # 下采样
    pool3 = max_pool('pool3', conv3, k=2)
    # 归一化
    norm3 = norm('norm3', pool3, lsize=4)
    # Dropout
    norm3 = tf.nn.dropout(norm3, _dropout)

    # 全连接层，先把特征图转为向量
    dense1 = tf.reshape(norm3, [-1, _weights['wd1'].get_shape().as_list()[0]])
    dense1 = tf.nn.relu(tf.matmul(dense1, _weights['wd1']) + _biases['bd1'], name='fc1')
    # 全连接层
    dense2 = tf.nn.relu(tf.matmul(dense1, _weights['wd2']) + _biases['bd2'], name='fc2') # Relu activation

    # 网络输出层
    out = tf.matmul(dense2, _weights['out']) + _biases['out']
    return out

# 存储所有的网络参数
weights = {
    'wc1': tf.Variable(tf.random_normal([3, 3, 1, 64])),
    'wc2': tf.Variable(tf.random_normal([3, 3, 64, 128])),
    'wc3': tf.Variable(tf.random_normal([3, 3, 128, 256])),
    'wd1': tf.Variable(tf.random_normal([4*4*256, 1024])),
    'wd2': tf.Variable(tf.random_normal([1024, 1024])),
    'out': tf.Variable(tf.random_normal([1024, 10]))
}
biases = {
    'bc1': tf.Variable(tf.random_normal([64])),
    'bc2': tf.Variable(tf.random_normal([128])),
    'bc3': tf.Variable(tf.random_normal([256])),
    'bd1': tf.Variable(tf.random_normal([1024])),
    'bd2': tf.Variable(tf.random_normal([1024])),
    'out': tf.Variable(tf.random_normal([n_classes]))
}

# 构建模型
pred = alex_net(x, weights, biases, keep_prob)

# 定义损失函数和学习步骤
cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(pred, y))
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)

# 测试网络
correct_pred = tf.equal(tf.argmax(pred,1), tf.argmax(y,1))
accuracy = tf.reduce_mean(tf.cast(correct_pred, tf.float32))

# 初始化所有的共享变量
init = tf.initialize_all_variables()
```



```
with tf.Session() as sess:
    sess.run(init)
    step = 1
    # Keep training until reach max iterations
    while step * batch_size < training_iters:
        batch_xs, batch_ys = mnist.train.next_batch(batch_size)
        # 获取批数据
        sess.run(optimizer, feed_dict={x: batch_xs, y: batch_ys, keep_prob: dropout})
        if step % display_step == 0:
            # 计算精度
            acc = sess.run(accuracy, feed_dict={x: batch_xs, y: batch_ys, keep_prob: 1.})
            # 计算损失值
            loss = sess.run(cost, feed_dict={x: batch_xs, y: batch_ys, keep_prob: 1.})
            print "Iter " + str(step*batch_size) + ", Minibatch Loss= " + "{:.6f}".format(loss) + ", Training Accuracy= " + "{:.5f}".format(acc)
        step += 1
    print "Optimization Finished!"
    # 计算测试精度
    print "Testing Accuracy:", sess.run(accuracy, feed_dict={x: mnist.test.images[:256], y: mnist.test.labels[:256], keep_prob: 1.})
```

上一篇：【机器学习】Tensorflow基本使用

下一篇：【机器学习】Tensorflow学习笔记