|  |
| --- |
| from \_\_future\_\_ import print\_function  import tensorflow as tf  from tensorflow.examples.tutorials.mnist import input\_data  import numpy as np  import matplotlib.pyplot as plt      #选去1到10的数字  mnist = input\_data.read\_data\_sets('MNIST\_data', one\_hot=True)    def add\_layer(inputs, in\_size, out\_size, activation\_function=None):  W = tf.Variable(tf.random\_normal([in\_size, out\_size]))  b = tf.Variable(tf.zeros([1,out\_size])+0.1)  Wb = tf.matmul(inputs, W)+b  if activation\_function is None:  outputs = Wb  else:  outputs = activation\_function(Wb)  return outputs    def compute\_accuracy\_CNN(v\_xs, v\_ys):  global prediction\_CNN  y\_pre = sess.run(prediction\_CNN, feed\_dict = {xs:v\_xs, keep\_prob:1})  correct\_prediction = tf.equal(tf.argmax(y\_pre,1), tf.argmax(v\_ys,1))  accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, tf.float32))  result = sess.run(accuracy, feed\_dict = {xs: v\_xs, ys:v\_ys, keep\_prob:1})  return result    def kernel\_variable(shape):  initial = tf.truncated\_normal(shape=shape, stddev = 0.1)  return tf.Variable(initial)    def bias\_variable(shape):  initial = tf.constant(0.1, shape=shape)  return tf.Variable(initial)    def conv2d(x,W):  return tf.nn.conv2d(x, W, strides = [1,1,1,1], padding = 'SAME') #x为输入，W为卷积权重，strides为滤波器移动范围，取步长为1    def max\_pool\_2x2(x):  return tf.nn.max\_pool(x, ksize= [1,2,2,1], strides=[1,2,2,1], padding='SAME')    #占位操作  xs = tf.placeholder(tf.float32, [None, 784]) #输入信息，输入向量为1\*784的矩阵  ys = tf.placeholder(tf.float32, [None, 10]) #类别标签共有10个类别  keep\_prob = tf.placeholder(tf.float32)  x\_image = tf.reshape(xs, [-1,28,28,1]) #将xs转化为28\*28的形式    # 卷积层  w\_conv1 = kernel\_variable([7,7,1,32]) #核 5\*5, 入层大小 1, 出层大小 32  b\_conv1 = bias\_variable([32])  h\_conv1 = tf.nn.relu(conv2d(x\_image, w\_conv1)+b\_conv1) #输出大小 28\*28\*32  h\_pool1 = max\_pool\_2x2(h\_conv1) #输出大小 14\*14\*32    # 卷积层2  w\_conv2 = kernel\_variable([7,7,32,64]) #核 5\*5, 入层大小 32, 出层大小 64  b\_conv2 = bias\_variable([64])  h\_conv2 = tf.nn.relu(conv2d(h\_pool1, w\_conv2)+ b\_conv2) #输出大小 14\*14\*64  h\_pool2 = max\_pool\_2x2(h\_conv2) #输出大小 7\*7\*64    # 全连接层1  w\_fc1 = kernel\_variable([7\*7\*64, 1024])  b\_fc1 = bias\_variable([1024])  h\_pool2\_flat = tf.reshape(h\_pool2, [-1,7\*7\*64])  h\_fc1 = tf.nn.relu(tf.matmul(h\_pool2\_flat, w\_fc1)+b\_fc1)  h\_fc1\_drop = tf.nn.dropout(h\_fc1, keep\_prob)    # 全连接层2  w\_fc2 = kernel\_variable([1024,10])  b\_fc2 = bias\_variable([10])  prediction\_CNN = tf.nn.softmax(tf.matmul(h\_fc1\_drop,w\_fc2)+b\_fc2)      #对数据进行预测，784个节点10个分类  prediction = add\_layer(xs, 784, 10, activation\_function= tf.nn.softmax)    #计算预测误差  cross\_entropy\_CNN = tf.reduce\_mean(-tf.reduce\_sum(ys\* tf.log(prediction\_CNN), reduction\_indices=[1]))    train\_step\_CNN = tf.train.AdamOptimizer(1e-4).minimize(cross\_entropy\_CNN)  saver = tf.train.Saver()    #结果输出 ，循环1500次  Total\_test\_loss\_CNN = np.zeros((int(1501/100)+1), float)  Total\_test\_acc\_CNN = np.zeros((int(1501/100)+1), float)  count =0  with tf.Session() as sess:  if int((tf.\_\_version\_\_).split('.')[1]) <12 and int((tf.\_\_version\_\_).split('.')[0])<1:  init = tf.initialize\_all\_veriables()  else:  init = tf.global\_variables\_initializer()  print(tf.\_\_version\_\_)  sess. run(init)  print('类型 ',' 迭代次数 ',' 百分比 ')    for i in range(1501):  batch\_xs, batch\_ys = mnist.train.next\_batch(100)  sess.run(train\_step\_CNN, feed\_dict={xs: batch\_xs, ys: batch\_ys, keep\_prob: 0.5})    if i%100 ==0:  Total\_test\_acc\_CNN[count] = compute\_accuracy\_CNN(mnist.test.images, mnist.test.labels)  print('正确率: ', i,' ', Total\_test\_acc\_CNN[count])  loss\_CNN = sess.run(cross\_entropy\_CNN,  feed\_dict={xs: mnist.test.images, ys: mnist.test.labels, keep\_prob: 1.0})  print('损失: ', i,' ', loss\_CNN)  Total\_test\_loss\_CNN[count] = loss\_CNN  count += 1    #数据存储与绘图  plt.figure(1, figsize=(12, 4))  plt.subplot(121)  plt.ylabel('损失')  plt.plot(Total\_test\_loss\_CNN, 'b\*-', lw=2)    plt.subplot(122)  plt.ylabel('精度')  plt.plot(Total\_test\_acc\_CNN, 'b\*-', lw=2)  plt.show() |



