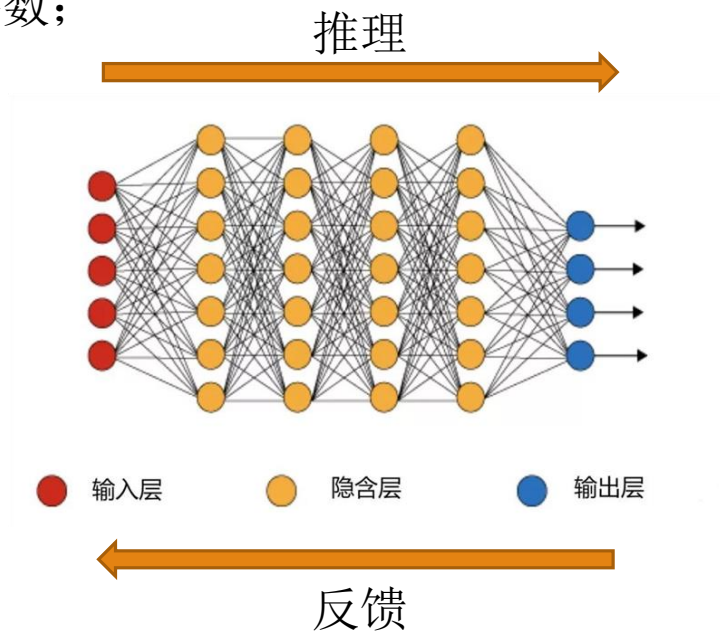
Tensorflow搭建神经网络

目 录

一. 神经网络的搭建流程

神经网络的实现过程

（1）准备数据集，提取特征，作为输入喂给神经网络；
（2）搭建神经网络结构，从输入到输出；
（3）搭建损失函数，优化函数；
（4）将数据喂给 NN，迭代优化 NN 参数；
（5）使用训练好的模型预测 。



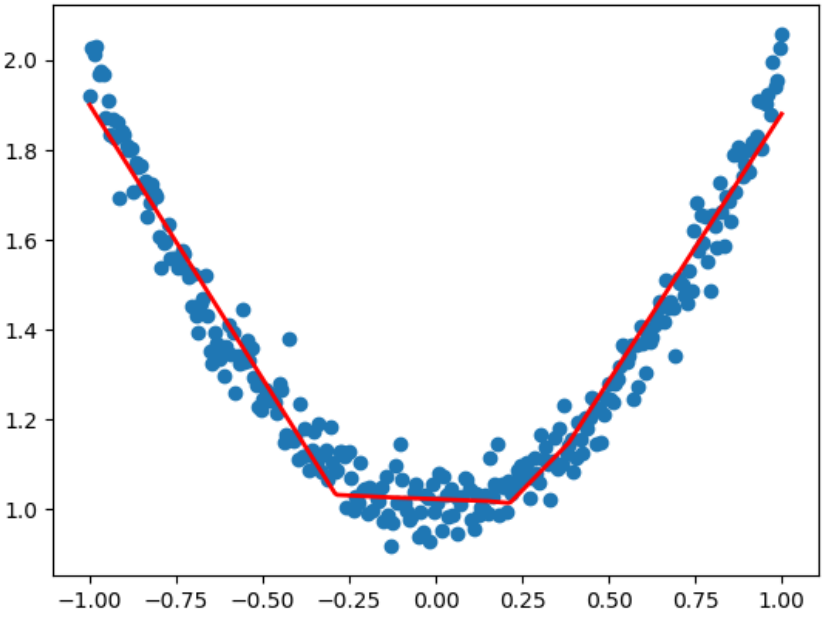
二. 函数拟合网络的搭建例子

拟合任务

本节用Tensorflow来搭建一个神经网络，实现y=x*x+1的拟合。

输入：
X：300个均匀分布在[-1,1]之间的数
Y：300个带噪声的计算的结果
即y=x*x+1+noise

输出：
Y`：网络推理的结果



二. 函数拟合网络的搭建例子

神经网络层的创建

搭建一个神经网络层，需要有数据输入、数据输出、激活函数。

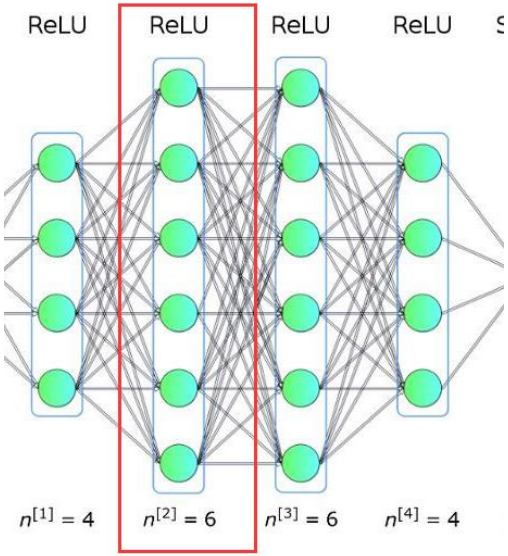
```
#创建一个神经网络层
def add_layer(input,in_size,out_size,activation_function=None):
    """
    :param input: 数据输入
    :param in_size: 输入大小（前一层神经元个数）
    :param out_size: 输出大小（本层神经元个数）
    :param activation_function: 激活函数（默认没有）
    :return:
    """
    Weight=tf.Variable(tf.random_normal([in_size,out_size]) )
    biases=tf.Variable(tf.zeros([1,out_size]) +0.1 )
    W_mul_x_plus_b=tf.matmul(input,Weight) + biases
    #根据是否有激活函数
    if activation_function == None:
        output=W_mul_x_plus_b
    else:
        output=activation_function(W_mul_x_plus_b)
    return output
```

二. 函数拟合网络的搭建例子

神经网络层的创建

搭建一个神经网络层，需要有数据输入、数据输出、激活函数。

实例化下图红色框的神经网络层时：
add_layer(input=X, in_size=4, out_size=6, activation_function=tf.nn.relu)



二. 函数拟合网络的搭建例子

训练数据的准备

```
# 创建输入数据 np.newaxis分别是在列(第二维)上增加维度
# 原先是（300，）变为（300，1）
x_data=np.linspace(-1,1,300)[:,np.newaxis]
noise=np.random.normal(0,0.05,x_data.shape) # 噪声
# 创建输入数据加噪后对应的输出
y_data=np.square(x_data)+1+noise
```

二. 函数拟合网络的搭建例子

定义数据格式

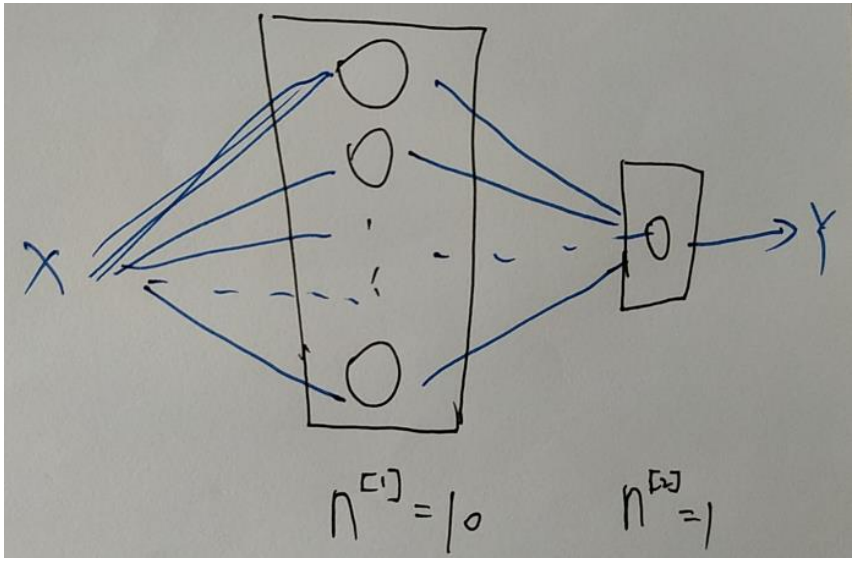
定义占位符，便于下边训练

```
# 定义输入数据的格式（xs是输入，ys是标签）
xs = tf.placeholder(tf.float32, [None, 1])
ys = tf.placeholder(tf.float32, [None, 1])
```

二. 函数拟合网络的搭建例子

定义神经网络

```
#定义一个隐藏层，含有10个神经元
hidden_layer1=add_layer(xs,1,10,activation_function=tf.nn.relu)

#定义一个输出层，含有1个神经元
prediction=add_layer(hidden_layer1,10,1,activation_function=None)
```



二. 函数拟合网络的搭建例子

定义损失与优化方法

```
# 求解神经网络参数
# 1.定义损失函数
loss = tf.reduce_mean(tf.reduce_sum(tf.square(ys - prediction), reduction_indices=[1]))

# 2.定义优化方法
ops = tf.train.GradientDescentOptimizer(0.1)
trian_step = ops.minimize(loss)
```

二. 函数拟合网络的搭建例子

创建会话与初始化环境

```
# 创建初始化器
init = tf.global_variables_initializer()
#创建会话
sess = tf.Session()
# 初始化环境
sess.run(init)
```

二. 函数拟合网络的搭建例子

定义训练过程

```
# 3.进行训练
for i in range(1000):
    sess.run(train_step, feed_dict={xs: x_data, ys: y_data})
    if i % 100 == 0:
        loss_res = sess.run(loss, feed_dict={xs: x_data, ys: y_data})
        print('step:%03d  loss:%.6f'%(i, loss_res))

#关闭会话
sess.close()
```

二. 函数拟合网络的搭建例子

结果输出

完整程序在：project->tf_build_net->tf_function_fit.py

训练输出：

```
step:000  loss:0.474701
step:100  loss:0.028231
step:200  loss:0.011180
step:300  loss:0.009257
step:400  loss:0.008132
step:500  loss:0.007320
step:600  loss:0.006670
step:700  loss:0.006128
step:800  loss:0.005683
step:900  loss:0.005317
```

发现随着训练次数的增加，loss越来越小。

二. 函数拟合网络的搭建例子

结果的可视化

（1）matplotlib在Python中应用最多的2D图像的绘图工具包，使用matplotlib能够非常简单的可视化数据。
（2）在matplotlib中使用最多的模块就是pyplot，下边用pyplot简单的展示以下训练过程：

```
# 首先，需要导入pyplot包
import matplotlib.pyplot as plt
```

二. 函数拟合网络的搭建例子

结果的可视化

```
# 绘制xs-ys的散点图
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.scatter(x_data, y_data) # 绘制散点图
plt.ion()  #Turn interactive mode on 开启互动模式
plt.show() #Display a figure
```

二. 函数拟合网络的搭建例子

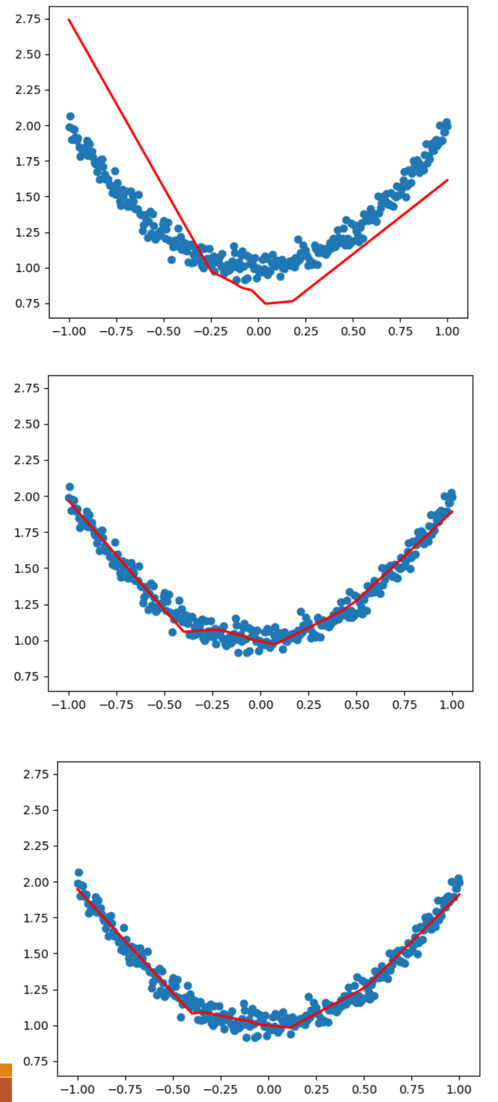
结果的可视化

```
for i in range(1000):
    sess.run(train_step, feed_dict={xs: x_data, ys: y_data})
    if i % 100 == 0:
        loss_res = sess.run(loss, feed_dict={xs: x_data, ys: y_data})
        print('step:%03d  loss:%.6f'%(i, loss_res))

        try:
            ax.lines.remove(lines[0])
        except Exception:
            pass

        # 计算预测值
        prediction_value = sess.run(prediction, feed_dict={xs: x_data})
        # 绘制预测值
        lines = ax.plot(x_data, prediction_value, 'r-', lw=2)
        plt.pause(0.1)
```

二. 函数拟合网络的搭建例子

结果的可视化输出

```
step:000   loss:0.123170
step:100   loss:0.004502
step:200   loss:0.003993
step:300   loss:0.003750
step:400   loss:0.003577
step:500   loss:0.003442
step:600   loss:0.003334
step:700   loss:0.003243
step:800   loss:0.003173
step:900   loss:0.003123
```



实现y=x*x+1的拟合

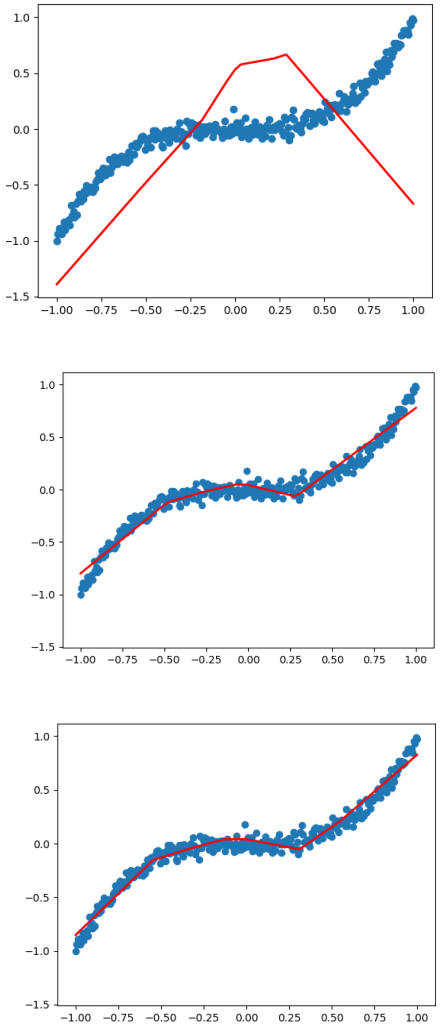
通过此方法可视化训练过程，可看到曲线渐渐拟合

二. 函数拟合网络的搭建例子

结果的可视化输出

实现y=x*x*x的拟合

```
step:000   loss:0.329593
step:100   loss:0.011349
step:200   loss:0.008293
step:300   loss:0.006794
step:400   loss:0.005993
step:500   loss:0.005497
step:600   loss:0.005166
step:700   loss:0.004888
step:800   loss:0.004630
step:900   loss:0.004424
```



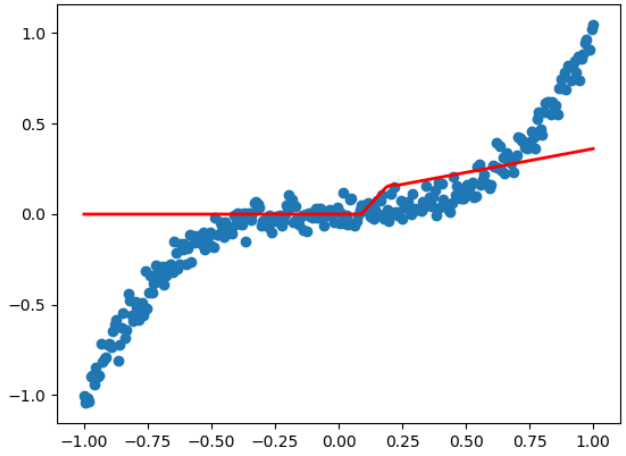
通过此方法可视化训练过程，可看到曲线渐渐拟合

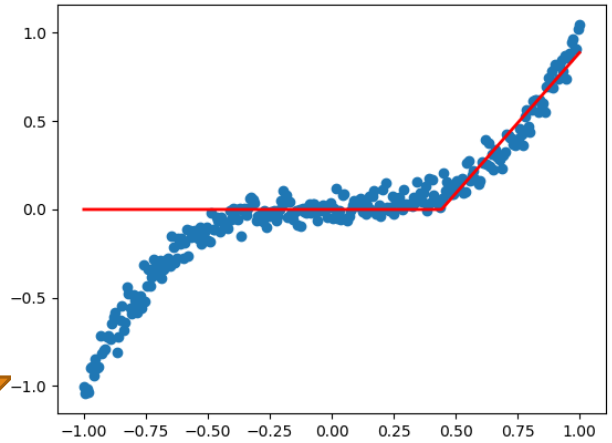
二. 函数拟合网络的搭建例子

结果的可视化输出

实现y=x*x*x的拟合
输出层使用ReLU激活

```
step:000   loss:0.100494
step:100   loss:0.079255
step:200   loss:0.078487
step:300   loss:0.078329
step:400   loss:0.078306
step:500   loss:0.078301
step:600   loss:0.078300
step:700   loss:0.078300
step:800   loss:0.078300
step:900   loss:0.078300
```





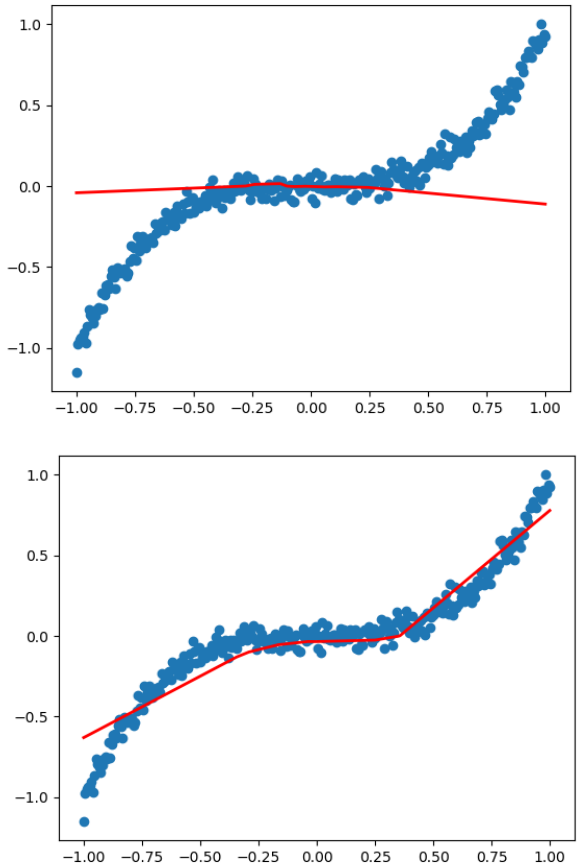
通过此方法可视化训练过程，可看到大于0部分曲线渐渐拟合，小于0的处于0的状态

二. 函数拟合网络的搭建例子

结果的可视化输出

实现y=x*x*x的拟合
输出层使用Leaky ReLU

```
step:000   loss:0.159705
step:100   loss:0.029575
step:200   loss:0.026761
step:300   loss:0.023233
step:400   loss:0.019851
step:500   loss:0.017565
step:600   loss:0.015844
step:700   loss:0.014434
step:800   loss:0.013245
step:900   loss:0.012218
```

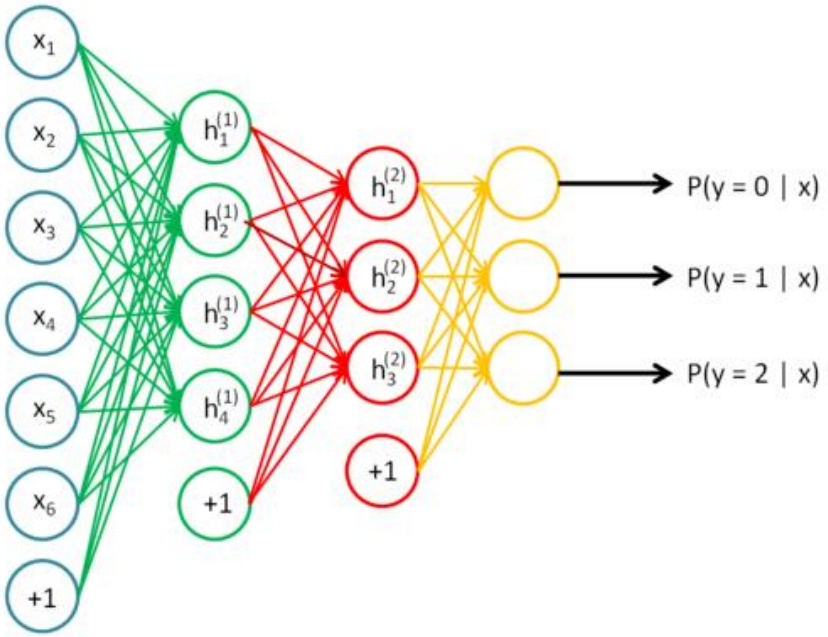


通过此方法可视化训练过程，可看到大于0部分曲线拟合较快，小于0的曲线拟合较慢

三. 多分类的问题

关于多分类

（1）常见的逻辑回归等常用于解决二分类问题，对于多分类问题，比如识别手写数字，它就需要10个分类。
（2）多分类同样也可以用逻辑回归，只是需要多个二分类来组成多分类。但这里讨论另外一种方式来解决多分类——softmax。



三. 多分类的问题

关于softmax

对于多分类问题，用C表示种类个数，神经网络中输出层就有C个神经元，即n[L]=C。其中，每个神经元的输出依次对应属于该类的概率，即P(y=c|x)。为了处理多分类问题，我们一般使用Softmax回归模型。Softmax回归模型输出层的激活函数如下所示：

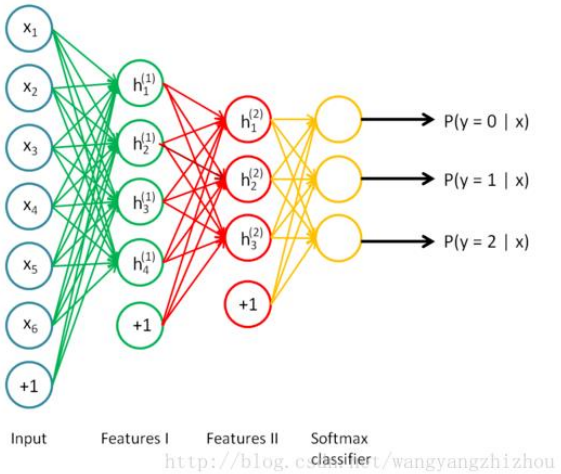
$$z^{[L]} = W^{[L]} a^{[L-1]} + b^{[L]}$$

$$a_i^{[L]} = \frac{e^{z_i^{[L]}}}{\sum_{i=1}^{C} e^{z_i^{[L]}}}$$

输出层每个神经元的输出$a_i^{[L]}$对应属于该类的概率，满足：

$$\sum_{i=1}^{C} a_i^{[L]} = 1$$

所有的$a_i^{[L]}$，即$\hat{y}$，维度为(C, 1)。

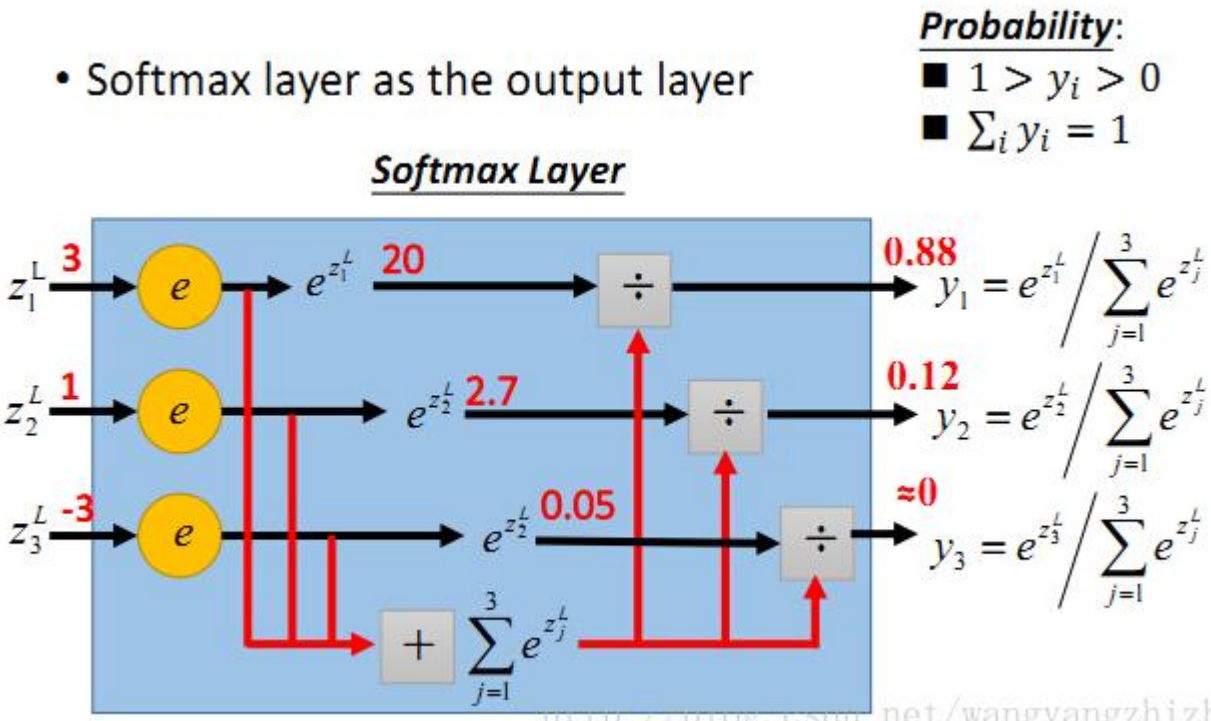


C=3的多分类网络

三. 多分类的问题

Softmax的计算流程

计算过程直接看下图，其中z_i^L即为$\theta_i^T x$，三个输入的值分别为3、1、-3，e^z的值为20、2.7、0.05，再分别除以累加和得到最终的概率值，0.88、0.12、0。



三. 多分类的问题

Softmax的损失函数

（1）softmax定义比较简单，在应用多分类的时候一般来说找概率最大的那一个值即可；
（2）需要有一个损失函数来判定实际输出和期望输出的差距，交叉熵就是用来判定实际的输出与期望的输出的接近程度，也就是交叉熵的值越小，两个概率分布就越接近；
（3）交叉熵损失函数又称softmax损失函数，是目前卷积神经网络中最常用的分类目标函数。其形式为：

$$\mathcal{L}_{\text{cross entropy loss}} = \mathcal{L}_{\text{softmax loss}} = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{e^{h_{y_i}}}{\sum_{j=1}^{C}e^{h_j}}\right).$$

即通过指数化变换使网络输出h 转换为概率形式。

四. 数字识别网络的搭建例子

数字识别的数据集

MNIST数据集是机器学习领域中非常经典的一个数据集，由60000个训练样本和10000个测试样本组成，每个样本都是一张28 * 28像素的灰度手写数字图片。

文件名称	大小	内容
train-images-idx3-ubyte.gz	9,681 kb	55000张训练集，5000张验证集
train-labels-idx1-ubyte.gz	29 kb	训练集图片对应的标签
t10k-images-idx3-ubyte.gz	1,611 kb	10000张测试集
t10k-labels-idx1-ubyte.gz	5 kb	测试集图片对应的标签

四. 数字识别网络的搭建例子

mnist数据集读取

（1）直接下载下来的数据是无法通过解压或者应用程序打开的，因为这些文件不是任何标准的图像格式而是以字节的形式进行存储的，所以必须编写程序来打开它。

（2）使用TensorFlow中input_data.py脚本来读取数据及标签，使用这种方式时，可以不用事先下载好数据集，它会自动下载并存放到指定的位置。 当指定的位置存在压缩文件时，程序会自动解压读取。

```
# MNIST_data指的是存放数据的文件夹路径
# one_hot=True 为采用one_hot的编码方式编码标签
mnist = input_data.read_data_sets(train_dir='MNIST_data', one_hot=True)
```

输出解压过程：

```
Extracting MNIST_data\train-images-idx3-ubyte.gz
Extracting MNIST_data\train-labels-idx1-ubyte.gz
Extracting MNIST_data\t10k-images-idx3-ubyte.gz
Extracting MNIST_data\t10k-labels-idx1-ubyte.gz
```

四. 数字识别网络的搭建例子

mnist数据集信息展示

程序如文件‘tf_mnist_show.py’所示。

```
train_X = mnist.train.images                #训练集样本
validation_X = mnist.validation.images      #验证集样本
test_X = mnist.test.images                  #测试集样本
#labels
train_Y = mnist.train.labels                #训练集标签
validation_Y = mnist.validation.labels      #验证集标签
test_Y = mnist.test.labels                  #测试集标签

print(train_X.shape,train_Y.shape)          #输出训练集样本和标签的大小
```

训练集样本和标签的大小

```
(55000, 784) (55000, 10)
```

第一个样本的标签，one-hot编码，#
只有对应位置的值是1，其余都是0

```
print(train_Y[0])
```

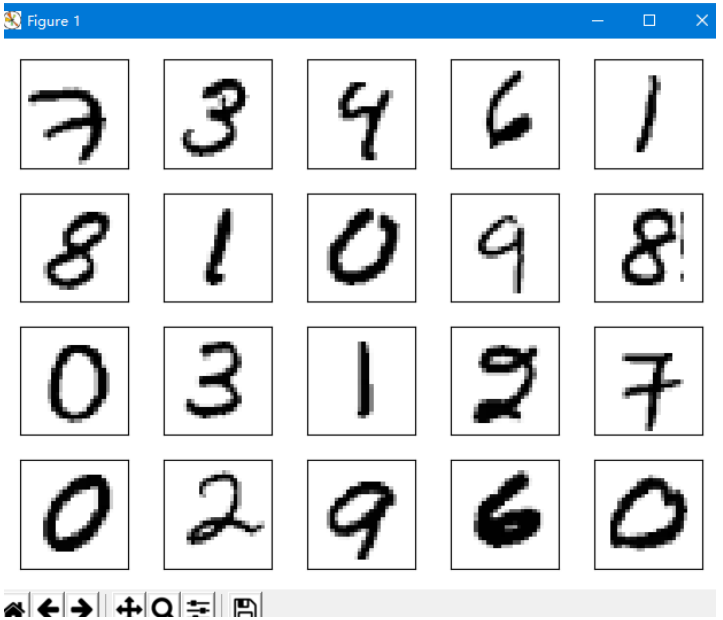
```
[0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
```

四. 数字识别网络的搭建例子

mnist数据集信息展示

程序如文件‘tf_mnist_show.py’所示。

```
#可视化样本，下面是输出了训练集中前20个样本
fig, ax = plt.subplots(nrows=4,ncols=5,
            sharex='all',sharey='all')
ax = ax.flatten()
for i in range(20):
    img = train_X[i].reshape(28, 28)
    ax[i].imshow(img,cmap='Greys')
ax[0].set_xticks([])
ax[0].set_yticks([])
plt.tight_layout()
plt.show()
```



四. 数字识别网络的搭建例子

神经网络的训练术语

Step（步数）: 训练模型的步数

Batch Size（批尺寸）： 计算梯度所需的样本数量，太小会导致效率低下，无法收敛。太大会导致内存撑不住，Batch Size增大到一定程度后，其下降方向变化很小了，所以Batch Size是一个很重要的参数。

Epoch（回合）：代表样本集内所有的数据经过了一次训练。

四. 数字识别网络的搭建例子

神经网络的训练流程

这里采用测试集最优的训练结果保存模型策略。

训练过程：
（1）在全部训练集遍历进行一次训练；
（2）计算验证集上的损失，属于历史最小时，保存模型；
（3）重新开始（1），直到符合一定的退出条件。

测试过程：
（1）加载模型；
（2）在测试集上遍历推理，计算准确度。

四. 数字识别网络的搭建例子

定义数据格式

```
x = tf.placeholder(tf.float32, [None, size_input])  #数据
y = tf.placeholder(tf.float32, [None, size_output]) #标签
```

四. 数字识别网络的搭建例子

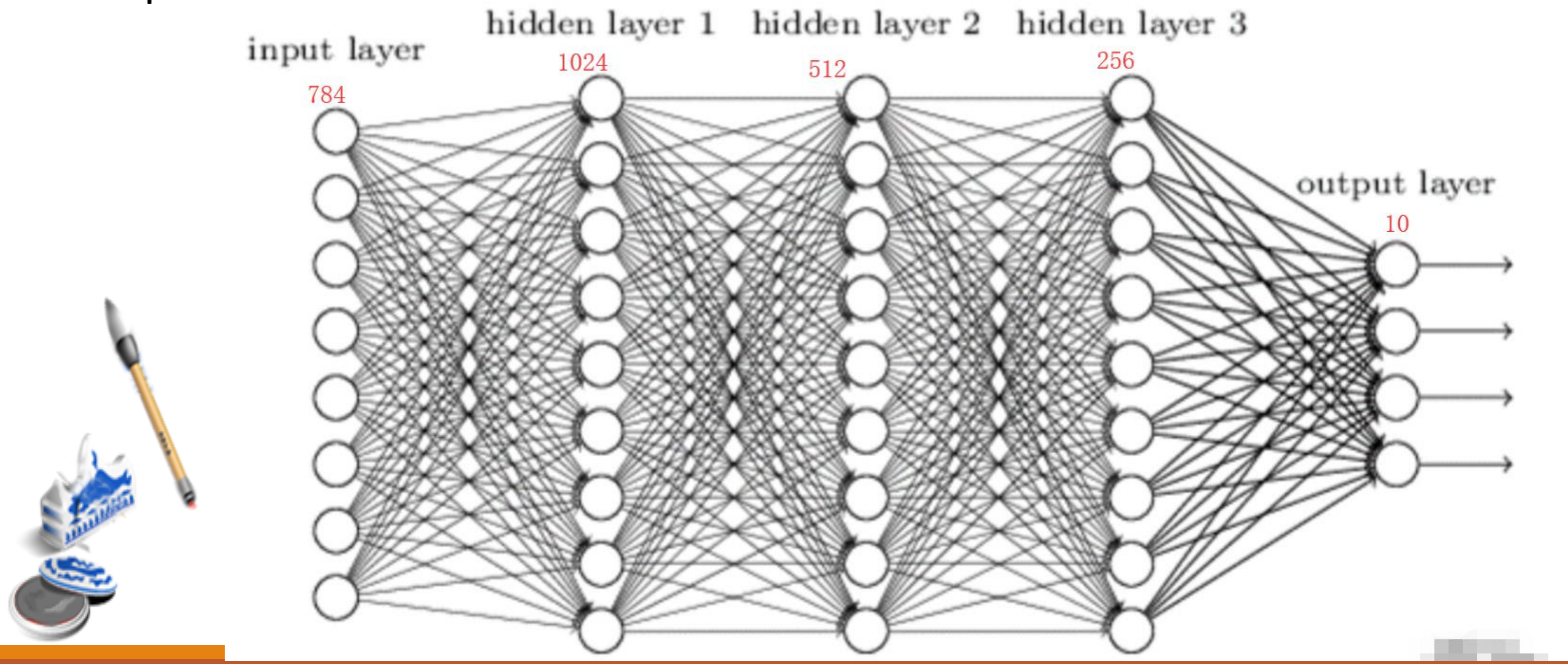
定义数据格式

```
size_input = 28*28
size_output = 10
epoch_max = 50
batch_size = 512
step_max = int(55000/batch_size)
acc_stop = 0.98
model_path = './mnist_model/'

x = tf.placeholder(tf.float32, [None, size_input])  #数据
y = tf.placeholder(tf.float32, [None, size_output]) #标签
```

四. 数字识别网络的搭建例子

定义网络结构

```
def network_mnist(input, in_size, out_size):
    hidden_layer1 = add_layer(input, in_size, 1024, activation_function=tf.nn.relu)
    hidden_layer2 = add_layer(hidden_layer1, 1024, 512, activation_function=tf.nn.relu)
    hidden_layer3 = add_layer(hidden_layer2, 512, 256, activation_function=tf.nn.relu)
    prediction = add_layer(hidden_layer3, 256, out_size, activation_function=tf.nn.softmax)
    return prediction
```



四. 数字识别网络的搭建例子

定义目标等

```
# 定义输出、损失与优化函数
y_pre = network_mnist(x, size_input, size_output)  #预测值,预测标签
cross_entropy = tf.reduce_mean(tf.reduce_sum(-y * tf.log(y_pre), reduction_indices=[1]))
train = tf.train.GradientDescentOptimizer(0.3).minimize(cross_entropy)

# 计算准确度
correct_predictions = tf.equal(tf.argmax(y_pre, 1), tf.argmax(y, 1))
accuracy = tf.reduce_mean(tf.cast(correct_predictions, tf.float32))
```

四. 数字识别网络的搭建例子

定义会话与初始化

```
saver = tf.train.Saver()
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init)
acc_max = -1
```

四. 数字识别网络的搭建例子

定义训练过程

```
for iepoch in range(epoch_max):
    is_save = 0
    acc_train_all = []
    for step in range(step_max):
        batch_x, batch_y = mnist.train.next_batch(batch_size)
        _, acc = sess.run([train, accuracy], feed_dict={x:batch_x, y:batch_y})
        acc_train_all.append(acc)
    acc_train = sum(acc_train_all)/len(acc_train_all)
    acc_val = sess.run(accuracy, feed_dict={x: mnist.validation.images, y: mnist.validation.labels})
    if acc_max < acc_val:
        saver.save(sess, model_path)
        is_save = 1
        acc_max = acc_val
    if acc_max >= acc_stop:
        print('End training early...')
        break
    print('epoch:%3d is_save:%d acc_max:%.4f%% acc_val:%.4f%% acc_train:%.4f%%'%(iepoch, is_save,
                                                    acc_max*100, acc_val*100, acc_train*100))
```

四. 数字识别网络的搭建例子

定义测试集的推理

```
acc_test = sess.run(accuracy, feed_dict={x: mnist.test.images, y: mnist.test.labels})
print('acc_test: %.4f%%'%(acc_test*100))
```

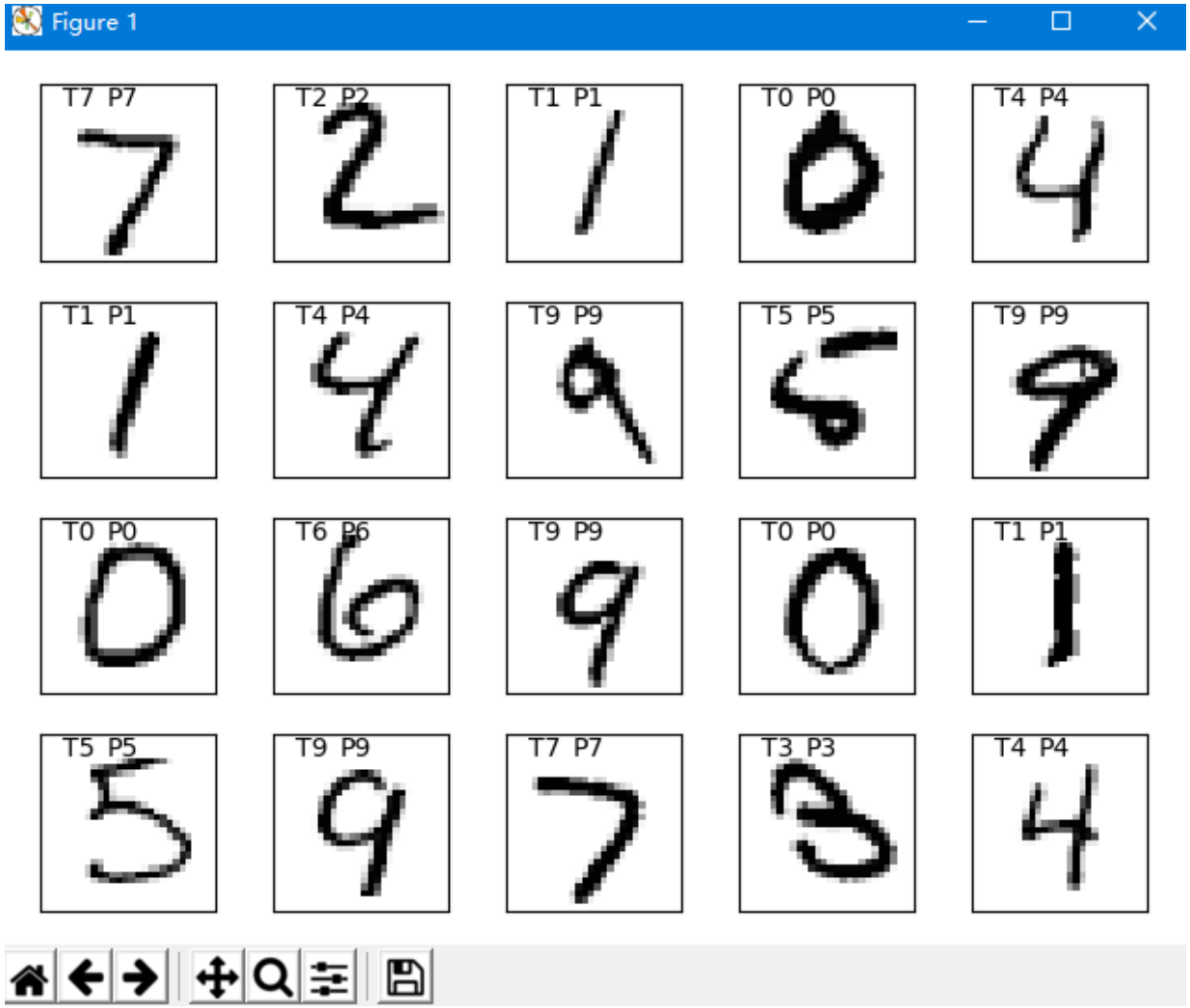
四. 数字识别网络的搭建例子

训练过程的输出：

```
epoch:00 is_save:1 acc_max:89.3800% acc_val:89.3800% acc_train:62.8523%
epoch:01 is_save:1 acc_max:93.4800% acc_val:93.4800% acc_train:89.9241%
epoch:02 is_save:1 acc_max:95.5600% acc_val:95.5600% acc_train:93.8559%
epoch:03 is_save:1 acc_max:96.5800% acc_val:96.5800% acc_train:95.4494%
epoch:04 is_save:1 acc_max:96.6400% acc_val:96.6400% acc_train:96.4844%
epoch:05 is_save:0 acc_max:96.6400% acc_val:96.3600% acc_train:96.9261%
epoch:06 is_save:1 acc_max:97.5600% acc_val:97.5600% acc_train:97.5540%
epoch:07 is_save:1 acc_max:97.6600% acc_val:97.6600% acc_train:97.9501%
epoch:08 is_save:0 acc_max:97.6600% acc_val:95.9600% acc_train:90.6487%
epoch:09 is_save:0 acc_max:97.6600% acc_val:97.4200% acc_train:96.9060%
epoch:10 is_save:1 acc_max:97.7400% acc_val:97.7400% acc_train:97.8589%
epoch:11 is_save:1 acc_max:97.7800% acc_val:97.7800% acc_train:97.8041%
epoch:12 is_save:1 acc_max:98.0400% acc_val:98.0400% acc_train:98.6018%
epoch:13 is_save:1 acc_max:98.0800% acc_val:98.0800% acc_train:98.8610%
epoch:14 is_save:1 acc_max:98.1400% acc_val:98.1400% acc_train:99.1603%
epoch:15 is_save:1 acc_max:98.2600% acc_val:98.2600% acc_train:99.3246%
epoch:16 is_save:1 acc_max:98.3600% acc_val:98.3600% acc_train:99.4579%
epoch:17 is_save:0 acc_max:98.3600% acc_val:98.2200% acc_train:99.5364%
epoch:18 is_save:0 acc_max:98.3600% acc_val:98.3200% acc_train:99.6696%
epoch:19 is_save:0 acc_max:98.3600% acc_val:98.2800% acc_train:99.7463%
epoch:20 is_save:1 acc_max:98.4200% acc_val:98.4200% acc_train:99.8485%
epoch:21 is_save:0 acc_max:98.4200% acc_val:98.4200% acc_train:99.8576%
epoch:22 is_save:1 acc_max:98.4600% acc_val:98.4600% acc_train:99.8960%
epoch:23 is_save:1 acc_max:98.5400% acc_val:98.5400% acc_train:99.9306%
epoch:24 is_save:0 acc_max:98.5400% acc_val:98.4000% acc_train:99.9489%
epoch:25 is_save:0 acc_max:98.5400% acc_val:98.3200% acc_train:99.9708%
```

四. 数字识别网络的搭建例子

测试集推理结果的可视化：
T-真实的数字
P-预测的数字



测试集推理的准确率：

```
acc_test: 98.0900%
```

四. 数字识别网络的搭建例子

完成的程序在‘project\tf_build_net’目录下：

‘tf_mnist_show.py’：展示mnist与可视化数据；

‘tf_mnist_train.py’：训练并得到模型；

‘tf_mnist_test.py’：在测试集上测试，获得准确率与可视化结果。

总结

一. 简述了神经网络的搭建流程；
二. 一步一步介绍如何搭建函数拟合网络；
三. 介绍多分类的问题，介绍sofxmax函数如何进行多分类；
四. 介绍数字识别网络的搭建流程，讲述网络的基本训练流程。

谢谢聆听

参考网页：

函数拟合：https://blog.csdn.net/u012679707/article/details/80485261
Mnist程序：https://blog.csdn.net/qq_33789319/article/details/79734384
Softmax：https://blog.csdn.net/wangyangzhizhou/article/details/75088106
Softmax2：https://blog.csdn.net/red_stone1/article/details/78403416
Mnist介绍：https://www.jianshu.com/p/d282bce1a999