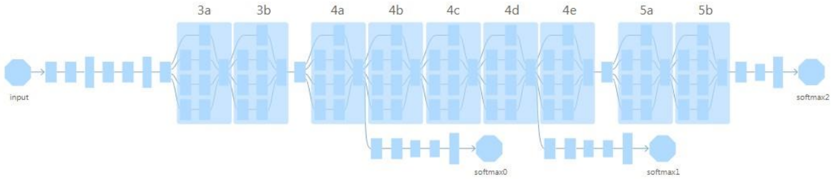


深度学习应用开发

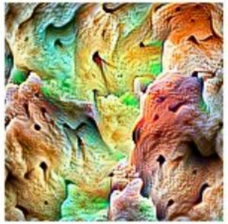
基于TensorFlow的实践

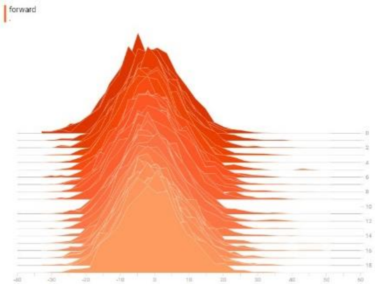
吴明晖 李卓蓉 金苍宏

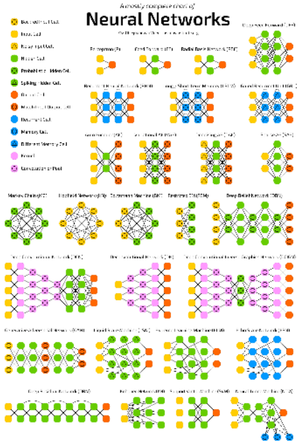
浙江大学城市学院

计算机与计算科学学院

Dept. of Computer Science
Zhejiang University City College

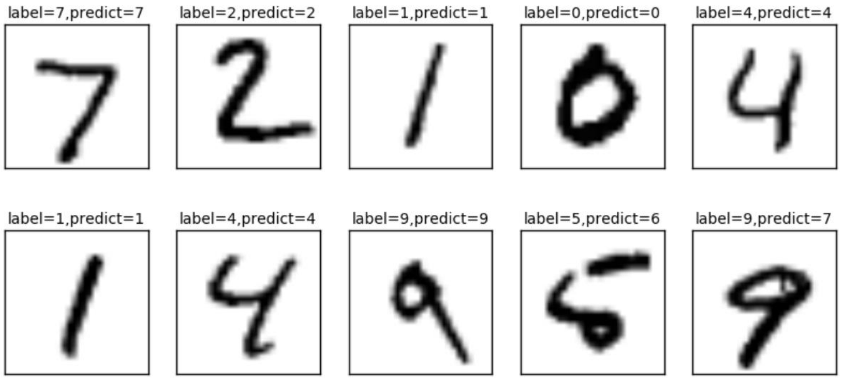



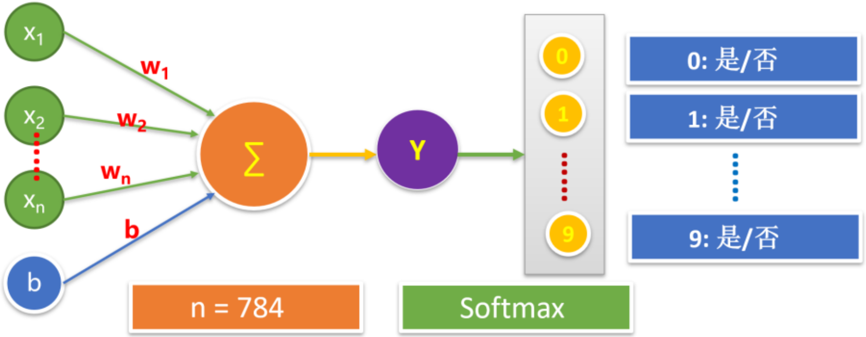


浙江大学城市学院
ZHEJIANG UNIVERSITY CITY COLLEGE

MNIST手写数字识别进阶
多层神经网络与应用

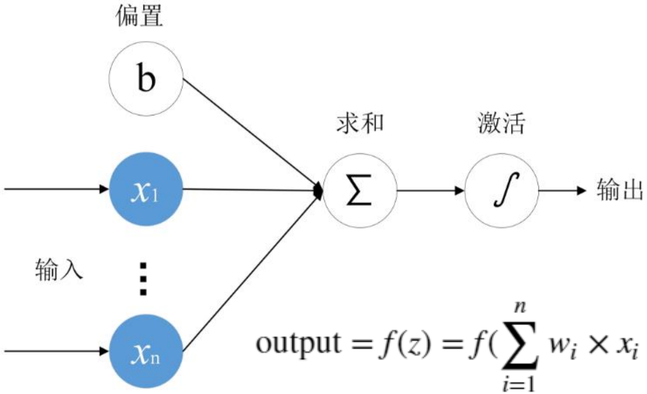
MNIST手写数字识别：分类应用入门






```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
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
```

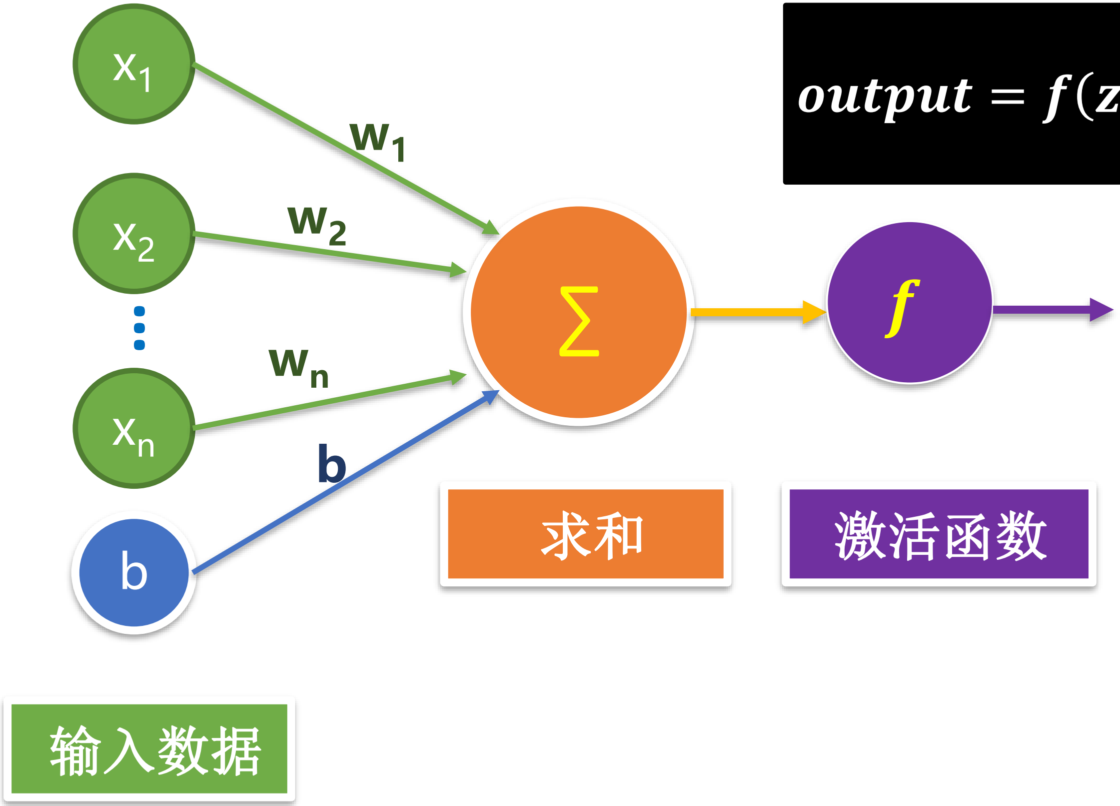


$$\text{output} = f(z) = f(\sum_{i=1}^{n} w_i \times x_i + b)$$

一个神经元处理分类问题

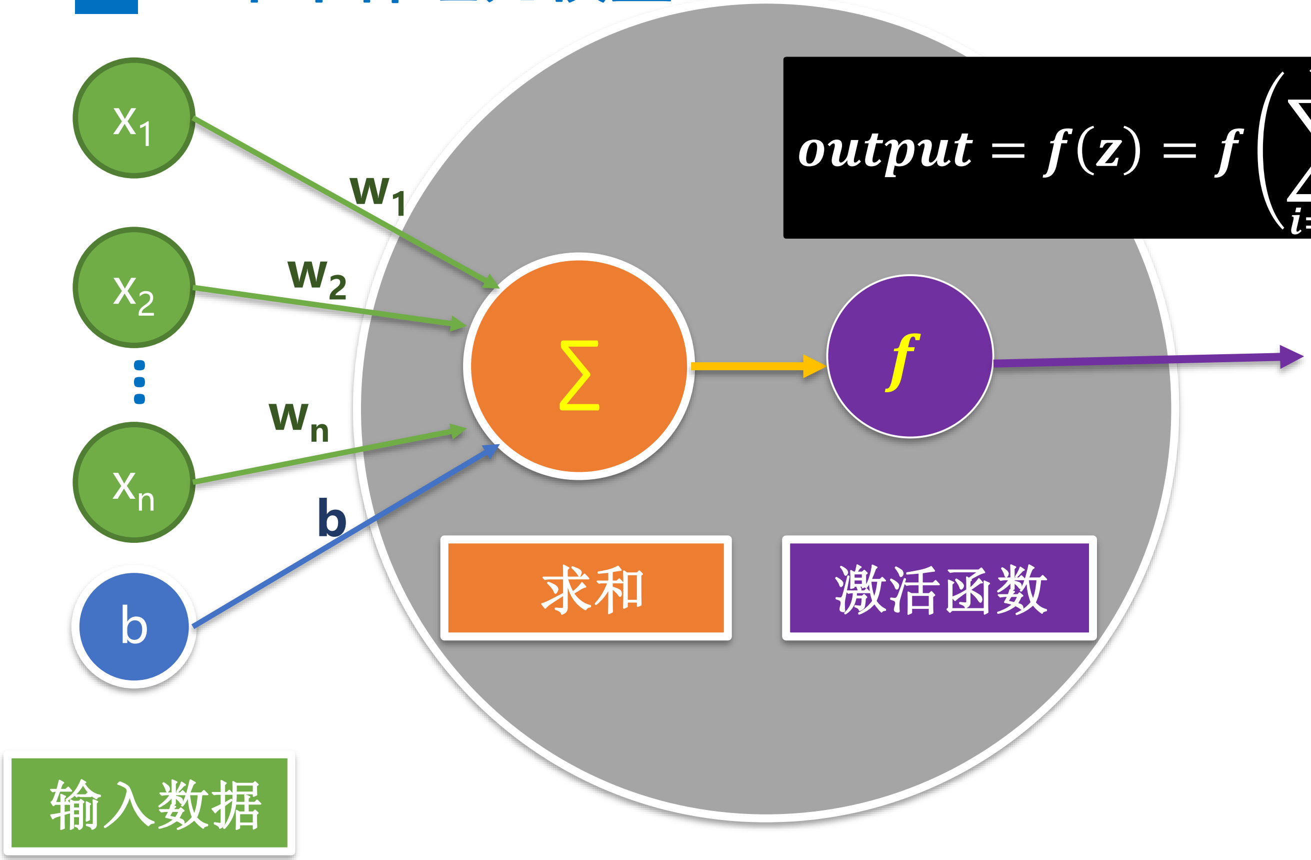
单个神经元模型

$$output = f(z) = f\left(\sum_{i=1}^{n}(x_i * w_i + b)\right)$$

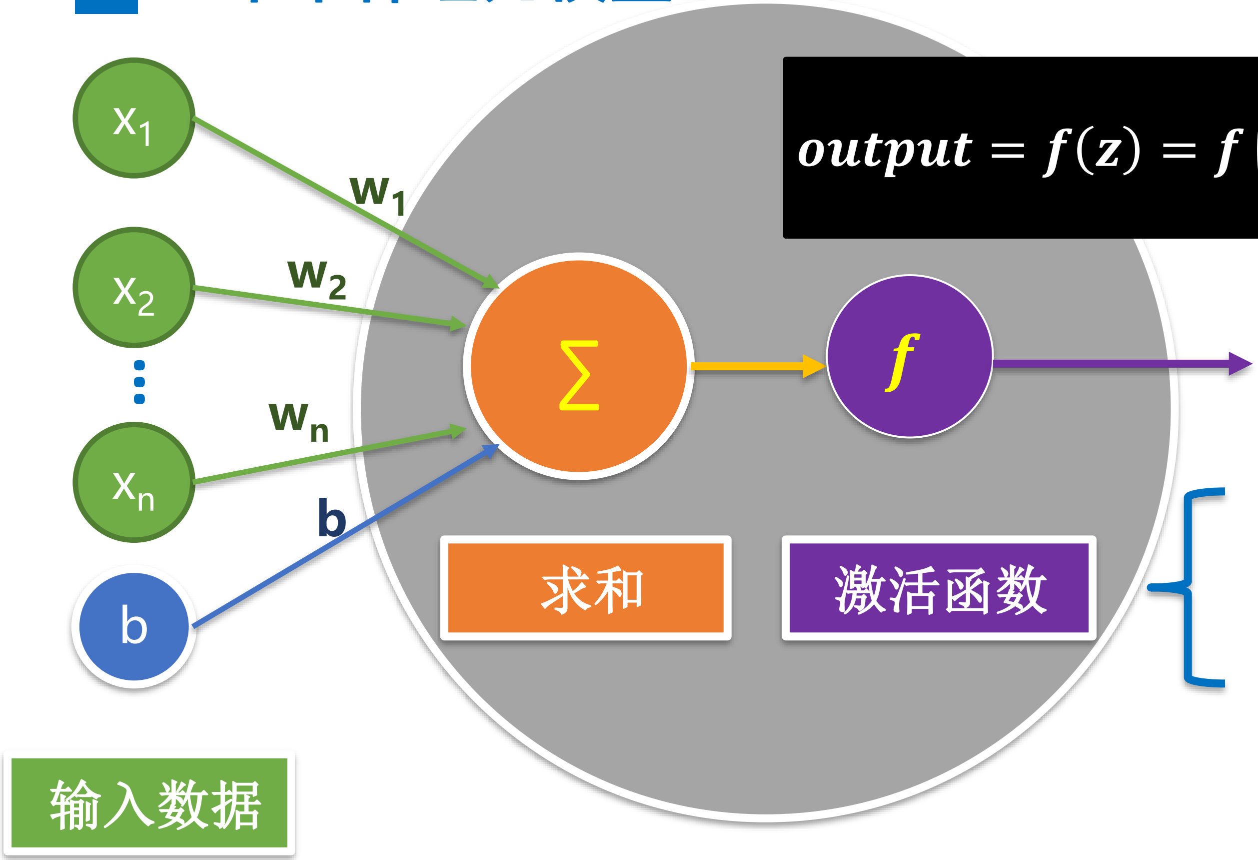


单个神经元模型

$$output = f(z) = f\left(\sum_{i=1}^{n}(x_i * w_i + b)\right)$$



单个神经元模型



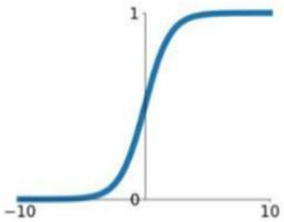
$$output = f(z) = f\left(\sum_{i=1}^{n}(x_i * w_i + b)\right)$$

Sigmod
ReLU
tanh
Softmax

常见激活函数

Sigmoid

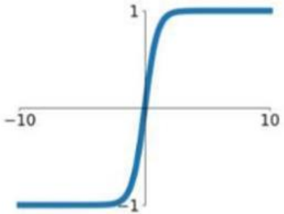
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



S型函数

tanh

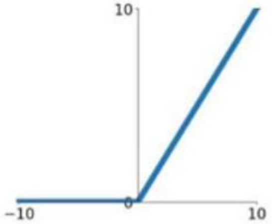
$$\tanh(x)$$



双曲正切函数

ReLU

$$\max(0, x)$$



修正线性单元函数

MNIST手写数字识别：单神经元模型效果

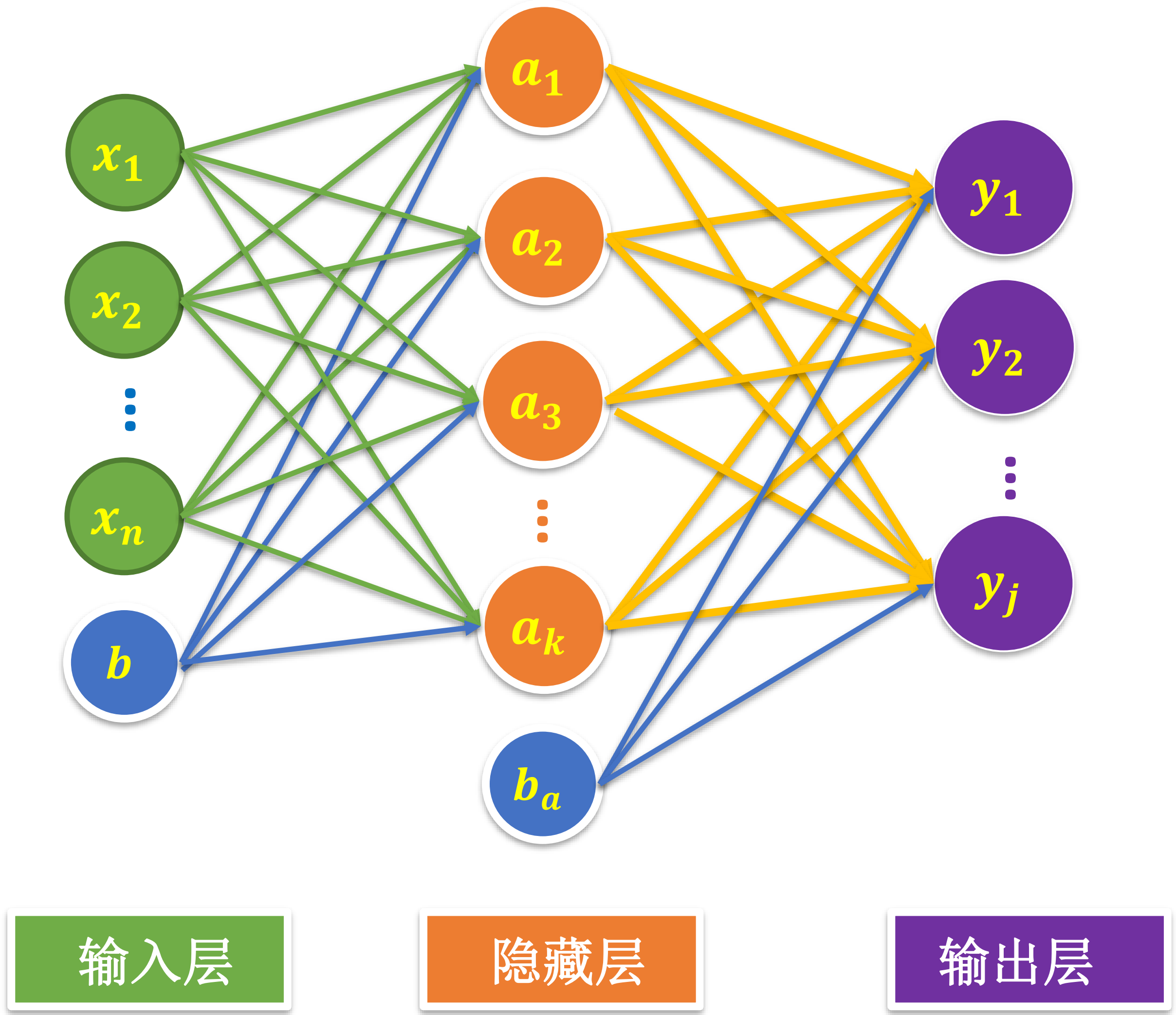
设置训练参数

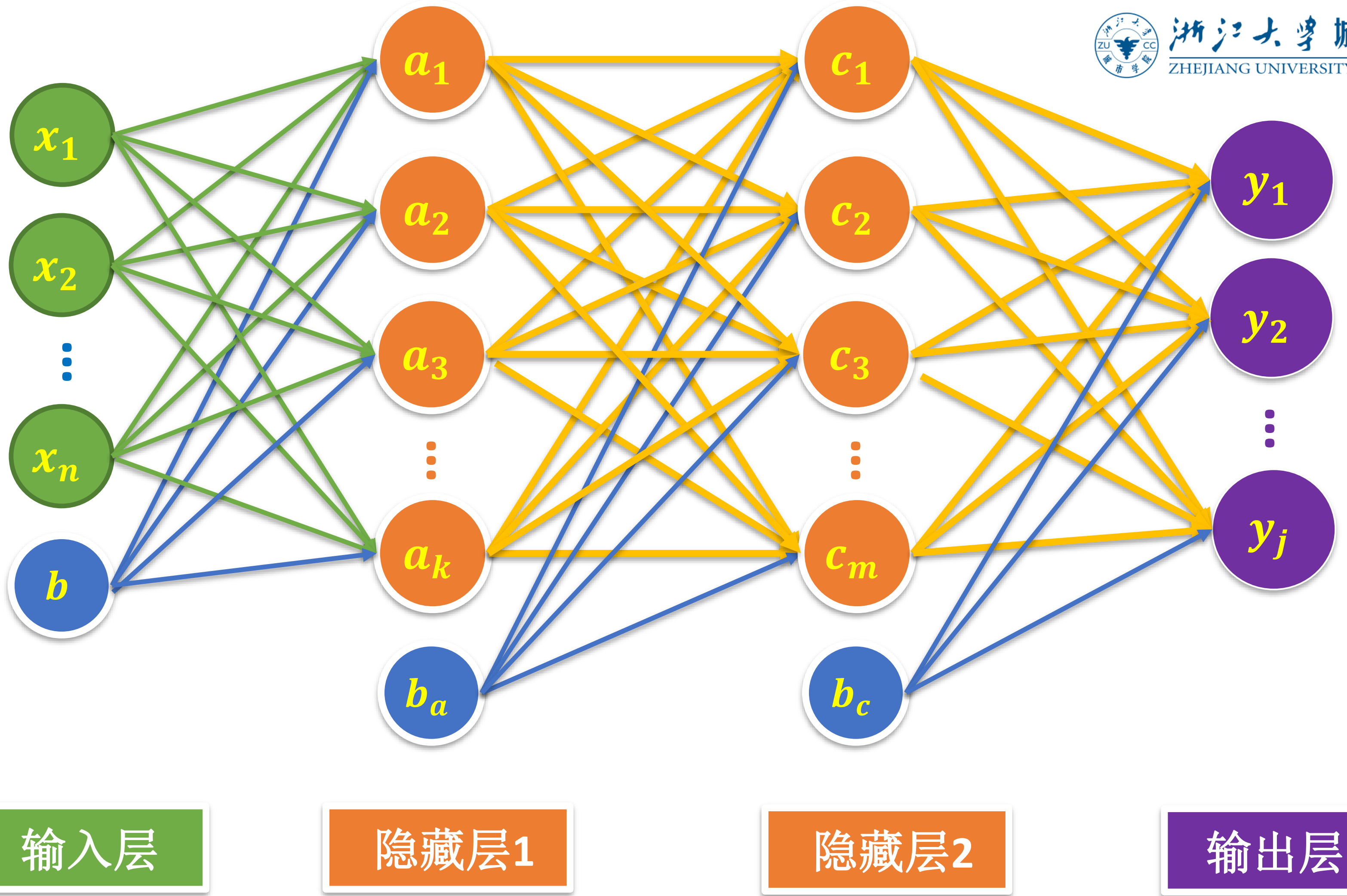
```
train_epochs = 150 # 训练轮数
batch_size = 50   # 单次训练样本数（批次大小）
total_batch= int(mnist.train.num_examples/batch_size)   # 一轮训练有多少批次
display_step = 1   # 显示粒度
learning_rate= 0.01   # 学习率
```

```
Train Epoch: 140 Loss= 0.364900649  Accuracy= 0.9058
Train Epoch: 141 Loss= 0.364611208  Accuracy= 0.9068
Train Epoch: 142 Loss= 0.364730358  Accuracy= 0.9072
Train Epoch: 143 Loss= 0.363039315  Accuracy= 0.9068
Train Epoch: 144 Loss= 0.362545907  Accuracy= 0.9064
Train Epoch: 145 Loss= 0.362331033  Accuracy= 0.9068
Train Epoch: 146 Loss= 0.361542165  Accuracy= 0.9064
Train Epoch: 147 Loss= 0.361528486  Accuracy= 0.9070
Train Epoch: 148 Loss= 0.360670209  Accuracy= 0.9070
Train Epoch: 149 Loss= 0.360280544  Accuracy= 0.9076
Train Epoch: 150 Loss= 0.360107958  Accuracy= 0.9072
Train Finished!
```

想要更加准确？多一点神经元

全连接单隐含层神经网络





x_1
x_2
x_n
b
a_1
a_2
a_3
a_k
b_a
c_1
c_2
c_3
c_m
b_c
y_1
y_2
y_j
输入层
隐藏层1
隐藏层2
输出层

全连接单隐藏层网络建模实现

载入数据

载入数据

```
import tensorflow as tf

# 导入Tensorflow提供的读取MNIST的模块
import tensorflow.examples.tutorials.mnist.input_data as input_data

# 读取MNIST数据
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
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
```

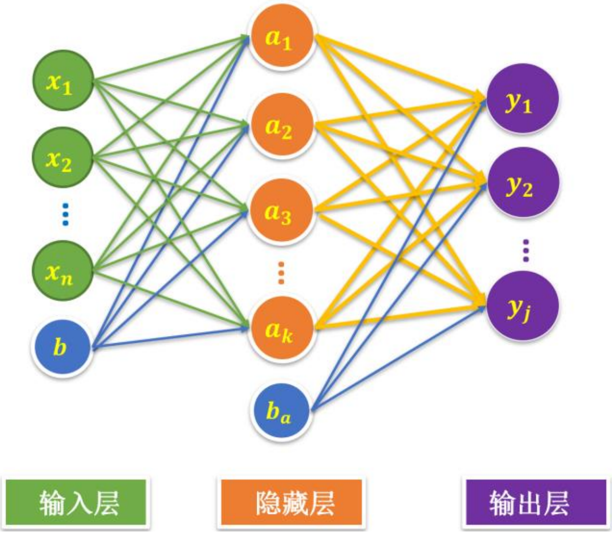
构建输入层

构建输入层

定义标签数据占位符

```
x = tf.placeholder(tf.float32, [None, 784], name="X")
```

```
y = tf.placeholder(tf.float32, [None, 10],name="Y")
```



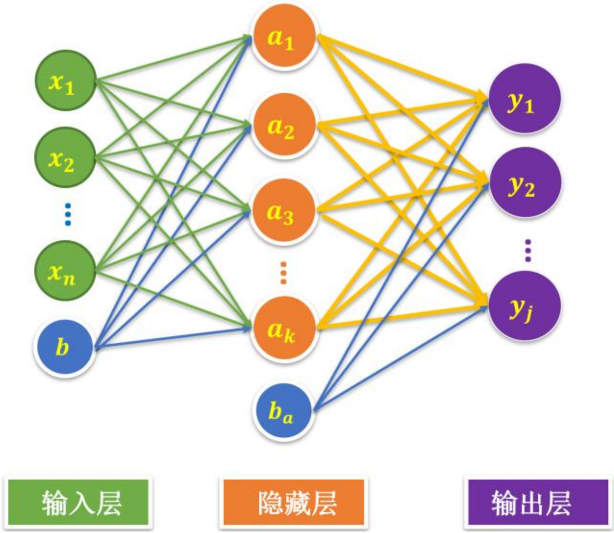
构建隐藏层

构建隐藏层

```
# 隐藏层神经元数量
H1_NN = 256

W1 = tf.Variable(tf.random_normal([784, H1_NN]))
b1 = tf.Variable(tf.zeros([H1_NN]))

Y1 = tf.nn.relu(tf.matmul(x, W1) + b1)
```

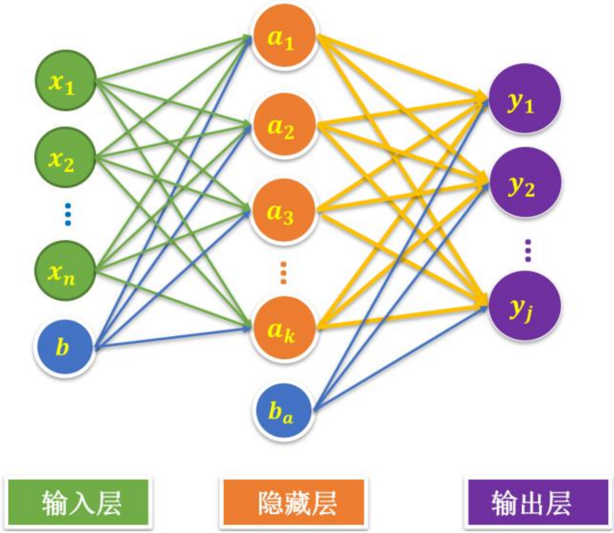


构建输出层

构建输出层

```
W2 = tf.Variable(tf.random_normal([H1_NN, 10]))
b2 = tf.Variable(tf.zeros([10]))

forward = tf.matmul(Y1, W2) + b2
pred = tf.nn.softmax(forward)
```



定义损失函数

训练模型

定义损失函数

```
# 交叉熵
loss_function = tf.reduce_mean(-tf.reduce_sum(y*tf.log(pred),
                                              reduction_indices=1))
```

训练模型

设置训练参数

```
train_epochs = 40
batch_size = 50
total_batch = int(mnist.train.num_examples/batch_size)
display_step = 1
learning_rate = 0.01
```

选择优化器

```
optimizer = tf.train.AdamOptimizer(learning_rate).minimize(loss_function)
```

训练模型

定义准确率

```
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(pred, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

训练模型

```
# 记录训练开始时间
from time import time
startTime=time()

sess = tf.Session()
sess.run(tf.global_variables_initializer())

for epoch in range(train_epochs):
    for batch in range(total_batch):
        xs, ys = mnist.train.next_batch(batch_size)# 读取批次数据
        sess.run(optimizer,feed_dict={x: xs,y: ys}) # 执行批次训练

    #total_batch个批次训练完成后，使用验证数据计算误差与准确率
    loss,acc = sess.run([loss_function,accuracy],
                        feed_dict={x: mnist.validation.images,
                                   y: mnist.validation.labels})

    if (epoch+1) % display_step == 0:
        print("Train Epoch:", '%02d' % (epoch+1),
              "Loss=", "{:.9f}".format(loss)," Accuracy=","{:.4f}".format(acc))

# 显示运行总时间
duration =time()-startTime
print("Train Finished takes:","{:.2f}".format(duration))
```

训练结果

```
Train Epoch: 01 Loss= nan  Accuracy= 0.0958
Train Epoch: 02 Loss= nan  Accuracy= 0.0958
Train Epoch: 03 Loss= nan  Accuracy= 0.0958
Train Epoch: 04 Loss= nan  Accuracy= 0.0958
Train Epoch: 05 Loss= nan  Accuracy= 0.0958
Train Epoch: 06 Loss= nan  Accuracy= 0.0958
Train Epoch: 07 Loss= nan  Accuracy= 0.0958
Train Epoch: 08 Loss= nan  Accuracy= 0.0958

Train Epoch: 35 Loss= nan  Accuracy= 0.0958
Train Epoch: 36 Loss= nan  Accuracy= 0.0958
Train Epoch: 37 Loss= nan  Accuracy= 0.0958
Train Epoch: 38 Loss= nan  Accuracy= 0.0958
Train Epoch: 39 Loss= nan  Accuracy= 0.0958
Train Epoch: 40 Loss= nan  Accuracy= 0.0958
Train Finished takes: 85.74
```

原因分析


```
Train Epoch: 01 Loss= nan  Accuracy= 0.0958
Train Epoch: 02 Loss= nan  Accuracy= 0.0958
Train Epoch: 03 Loss= nan  Accuracy= 0.0958
Train Epoch: 04 Loss= nan  Accuracy= 0.0958
Train Epoch: 05 Loss= nan  Accuracy= 0.0958
Train Epoch: 06 Loss= nan  Accuracy= 0.0958
Train Epoch: 07 Loss= nan  Accuracy= 0.0958
Train Epoch: 08 Loss= nan  Accuracy= 0.0958
```

log(0)引起的数据不稳定

定义损失函数

```
# 交叉熵
loss_function = tf.reduce_mean(-tf.reduce_sum(y*tf.log(pred),
                                              reduction_indices=1))
```

新的损失函数定义方法

定义损失函数

```
# 交叉熵
loss_function = tf.reduce_mean(-tf.reduce_sum(y*tf.log(pred),
                                                reduction_indices=1))
```

TensorFlow提供了结合Softmax的交叉熵损失函数定义方法

```
# TensorFlow 提供了softmax_cross_entropy_with_logits 函数
# 用于避免因为log(0) 值为 NaN 造成的数据不稳定

loss_function = tf.reduce_mean(
                 tf.nn.softmax_cross_entropy_with_logits(logits=forward, labels=y))
```

不做Softmax的数据

修改完损失函数再训练结果

```
Train Epoch: 01 Loss= 1.440694332  Accuracy= 0.9326
Train Epoch: 02 Loss= 0.828203321  Accuracy= 0.9476
Train Epoch: 03 Loss= 0.624041617  Accuracy= 0.9522
Train Epoch: 04 Loss= 0.522492349  Accuracy= 0.9550
Train Epoch: 05 Loss= 0.532067716  Accuracy= 0.9528
Train Epoch: 06 Loss= 0.400931746  Accuracy= 0.9582
Train Epoch: 07 Loss= 0.415431648  Accuracy= 0.9640
Train Epoch: 08 Loss= 0.426045895  Accuracy= 0.9580
Train Epoch: 09 Loss= 0.410735846  Accuracy= 0.9638
Train Epoch: 10 Loss= 0.388119668  Accuracy= 0.9662

Train Epoch: 37 Loss= 0.989025116  Accuracy= 0.9748
Train Epoch: 38 Loss= 1.012039423  Accuracy= 0.9742
Train Epoch: 39 Loss= 1.136060476  Accuracy= 0.9738
Train Epoch: 40 Loss= 0.759704173  Accuracy= 0.9756
Train Finished takes: 87.11
```

评估模型

评估模型

```
accu_test =  sess.run(accuracy,
                      feed_dict={x: mnist.test.images, y: mnist.test.labels})

print("Test Accuracy:",accu_test)
```

Test Accuracy: 0.9718

应用模型

进行预测

```
# 由于pred预测结果是one-hot编码格式，所以需要转换为0~9数字

prediction_result=sess.run(tf.argmax(pred,1),
                          feed_dict={x: mnist.test.images })
```

```
#查看预测结果中的前10项

prediction_result[0:10]
```

```
array([7, 2, 1, 0, 4, 1, 4, 9, 5, 9], dtype=int64)
```

找出预测错误

找出预测错误

```
compare_lists = prediction_result==np.argmax(mnist.test.labels,1)
print(compare_lists)
```

```
[ True  True  True ...,  True  True  True]
```

```
err_lists = [i for i in range(len(compare_lists)) if compare_lists[i]==False]
print(err_lists, len(err_lists))
```

```
[119, 247, 274, 320, 321, 340, 445, 447, 460, 469, 495, 582, 619, 659, 674, 684, 691, 720, 726, 839, 846, 900, 9
47, 956, 992, 1014, 1039, 1107, 1112, 1156, 1178, 1182, 1226, 1232, 1242, 1247, 1272, 1289, 1299, 1319, 1326, 13
28, 1356, 1393, 1403, 1433, 1494, 1496, 1522, 1530, 1549, 1551, 1553, 1670, 1681, 1732, 1754, 1782, 1800, 1813,
1868, 1878, 1901, 1941, 1968, 2004, 2016, 2024, 2035, 2040, 2043, 2053, 2058, 2070, 2098, 2109, 2130, 2135, 218
2, 2185, 2225, 2237, 2292, 2299, 2326, 2369, 2387, 2395, 2406, 2433, 2462, 2488, 2512, 2526, 2573, 2597, 2607, 2
610, 2648, 2654, 2720, 2730, 2743, 2810, 2823, 2863, 2864, 2896, 2921, 2927, 2938, 2939, 2970, 2995, 3030, 3060,
3073, 3115, 3157, 3225, 3263, 3289, 3405, 3422, 3475, 3503, 3520, 3542, 3549, 3558, 3597, 3662, 3702, 3730, 375
1, 3767, 3776, 3796, 3808, 3869, 3906, 3941, 3962, 3984, 3985, 4007, 4027, 4065, 4075, 4154, 4176, 4199, 4201, 4
248, 4265, 4285, 4289, 4294, 4315, 4359, 4360, 4374, 4433, 4477, 4497, 4507, 4536, 4547, 4571, 4578, 4601, 4690,
4731, 4751, 4761, 4855, 4879, 4880, 4943, 4956, 4966, 5067, 5152, 5199, 5228, 5246, 5331, 5457, 5600, 5623, 564
2, 5649, 5654, 5676, 5688, 5714, 5734, 5835, 5854, 5887, 5891, 5906, 5936, 5972, 5985, 6028, 6035, 6053, 6059, 6
101, 6390, 6421, 6571, 6597, 6598, 6599, 6603, 6625, 6641, 6741, 6755, 6769, 6783, 6817, 6847, 6980, 7256, 7401,
7432, 7434, 7451, 7457, 7783, 7822, 7842, 7847, 7849, 7851, 7856, 7860, 7905, 7928, 7971, 7990, 8062, 8091, 809
4, 8246, 8255, 8277, 8311, 8325, 8408, 8416, 8453, 8456, 8502, 8519, 8520, 9009, 9012, 9015, 9024, 9071, 9225, 9
422, 9538, 9540, 9587, 9634, 9664, 9669, 9679, 9709, 9719, 9729, 9745, 9770, 9839, 9858, 9888, 9944] 282
```

定义输出错误分类的函数

定义一个输出错误分类的函数

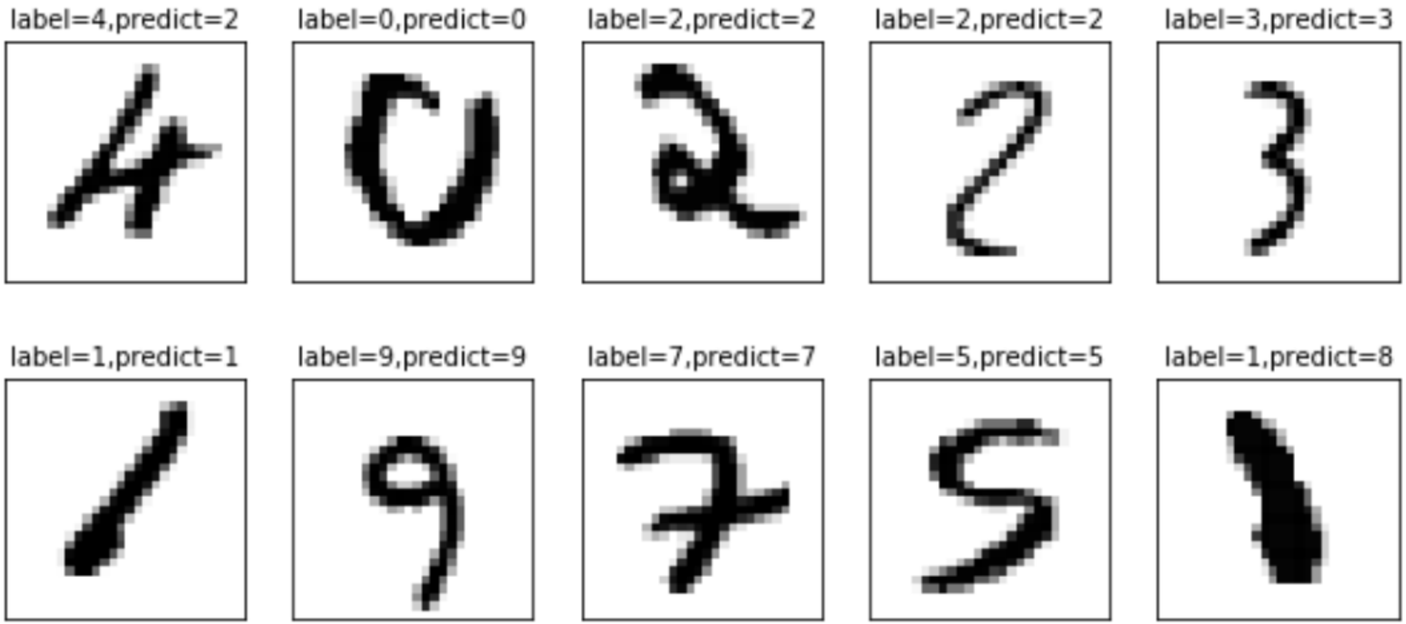
```
import numpy as np
def print_predict_errs(labels,        # 标签列表
                       prediction):   # 预测值列表
    count = 0
    compare_lists = (prediction==np.argmax(labels,1))
    err_lists = [i for i in range(len(compare_lists)) if compare_lists[i]==False]
    for x in err_lists:
        print("index="+str(x)+
              " 标签值=",np.argmax(labels[x]),
              "预测值=",prediction[x])
        count = count + 1
    print("总计:"+str(count))
```

```
print_predict_errs(labels=mnist.test.labels,
                   prediction=prediction_result)
```

```
index=119 标签值= 2 预测值= 8
index=247 标签值= 4 预测值= 2
index=274 标签值= 9 预测值= 3
index=320 标签值= 9 预测值= 8
index=321 标签值= 2 预测值= 7
```

可视化查看预测错误的样本

```
plot_images_labels_prediction(mnist.test.images,
                              mnist.test.labels,
                              prediction_result,610,20)
```



定义可视化函数


```python
import matplotlib.pyplot as plt
import numpy as np
def plot_images_labels_prediction(images,      # 图像列表
                                  labels,      # 标签列表
                                  prediction,  # 预测值列表
                                  index,       # 从第index个开始显示
                                  num=10 ):    # 缺省一次显示 10 幅
    fig = plt.gcf() # 获取当前图表，Get Current Figure
    fig.set_size_inches(10, 12)  # 1英寸等于 2.54 cm
    if num > 25:
        num = 25            # 最多显示25个子图
    for i in range(0, num):
        ax = plt.subplot(5,5, i+1) # 获取当前要处理的子图

        ax.imshow(np.reshape(images[index],(28, 28)),   # 显示第index个图像
                  cmap='binary')

        title = "label=" + str(np.argmax(labels[index]))  # 构建该图上要显示的title信息
        if len(prediction)>0:
            title += ",predict=" + str(prediction[index])

        ax.set_title(title,fontsize=10)    # 显示图上的title信息
        ax.set_xticks([]);  # 不显示坐标轴
        ax.set_yticks([])
        index += 1
    plt.show()
```