tensorflow / tensorflow

TensorFlow: Open sou

Clone this repo:

```
git clone https://tensor
```

Branches

master

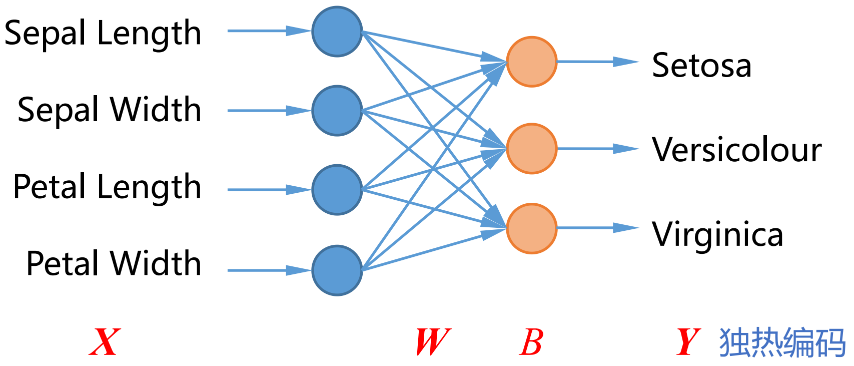
db0b

12.6 实例：实现多层神经网络

中国大学MOOC

课程回顾

单层神经网络



Sepal Length
Sepal Width
Petal Length
Petal Width

Setosa
Versicolour
Virginica

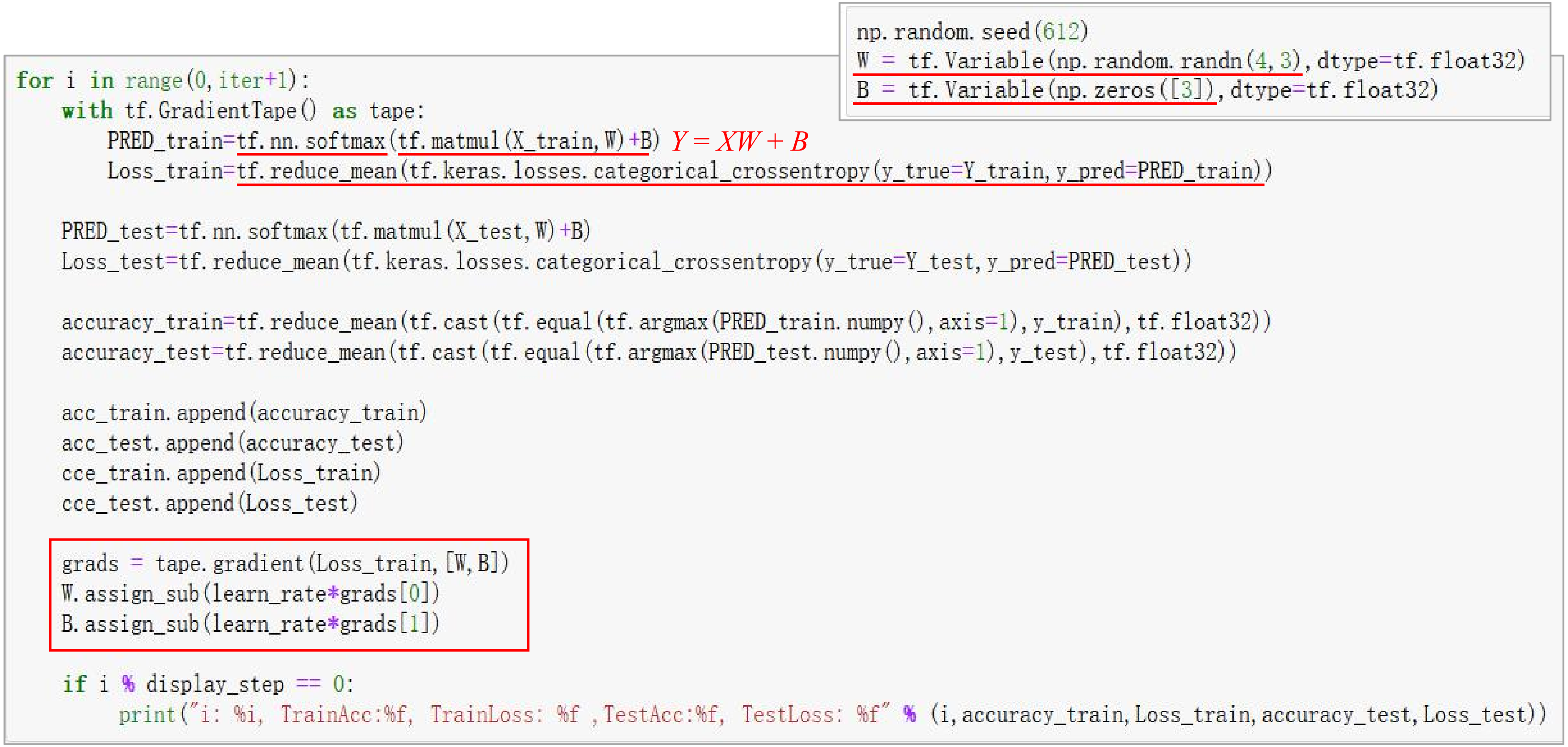
X *W* *B* *Y* 独热编码

- 激活函数 softmax函数
- 损失函数 交叉熵损失函数

$$Y = X\,W + B$$

输出 输入属性 权值矩阵 阈值

课程回顾

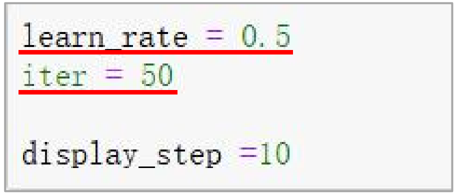


```
np.random.seed(612)
W = tf.Variable(np.random.randn(4,3),dtype=tf.float32)
B = tf.Variable(np.zeros([3]),dtype=tf.float32)
```

```
for i in range(0,iter+1):
    with tf.GradientTape() as tape:
        PRED_train=tf.nn.softmax(tf.matmul(X_train,W)+B)  Y = XW + B
        Loss_train=tf.reduce_mean(tf.keras.losses.categorical_crossentropy(y_true=Y_train,y_pred=PRED_train))

    PRED_test=tf.nn.softmax(tf.matmul(X_test,W)+B)
    Loss_test=tf.reduce_mean(tf.keras.losses.categorical_crossentropy(y_true=Y_test,y_pred=PRED_test))

    accuracy_train=tf.reduce_mean(tf.cast(tf.equal(tf.argmax(PRED_train.numpy(),axis=1),y_train),tf.float32))
    accuracy_test=tf.reduce_mean(tf.cast(tf.equal(tf.argmax(PRED_test.numpy(),axis=1),y_test),tf.float32))

    acc_train.append(accuracy_train)
    acc_test.append(accuracy_test)
    cce_train.append(Loss_train)
    cce_test.append(Loss_test)

    grads = tape.gradient(Loss_train,[W,B])
    W.assign_sub(learn_rate*grads[0])
    B.assign_sub(learn_rate*grads[1])

    if i % display_step == 0:
        print("i: %i, TrainAcc:%f, TrainLoss: %f ,TestAcc:%f, TestLoss: %f" % (i,accuracy_train,Loss_train,accuracy_test,Loss_test))
```

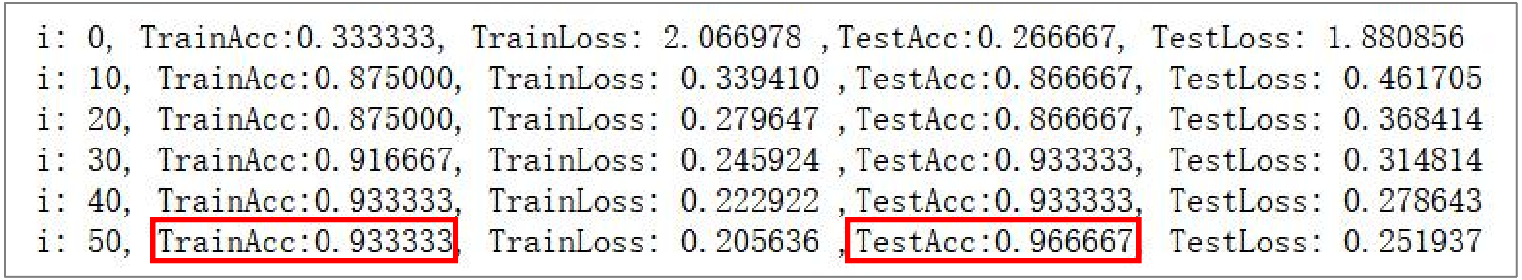
课程回顾

设置超参数



```
learn_rate = 0.5
iter = 50

display_step =10
```

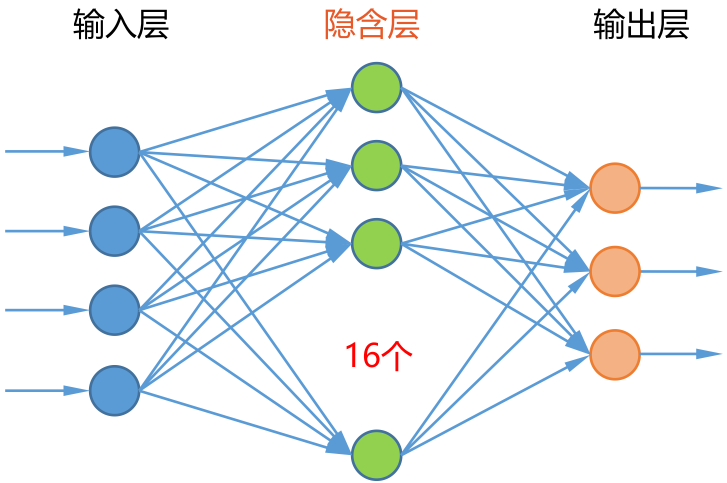
训练结果



```
i: 0,  TrainAcc:0.333333,  TrainLoss: 2.066978 ,TestAcc:0.266667,  TestLoss: 1.880856
i: 10,  TrainAcc:0.875000,  TrainLoss: 0.339410 ,TestAcc:0.866667,  TestLoss: 0.461705
i: 20,  TrainAcc:0.875000,  TrainLoss: 0.279647 ,TestAcc:0.866667,  TestLoss: 0.368414
i: 30,  TrainAcc:0.916667,  TrainLoss: 0.245924 ,TestAcc:0.933333,  TestLoss: 0.314814
i: 40,  TrainAcc:0.933333,  TrainLoss: 0.222922 ,TestAcc:0.933333,  TestLoss: 0.278643
i: 50,  TrainAcc:0.933333,  TrainLoss: 0.205636 ,TestAcc:0.966667,  TestLoss: 0.251937
```

12.6 实例：实现多层神经网络

□ 多层神经网络**结构设计**



W_1	B_1	W_2	B_2
(4,16)	(16,)	(16,3)	(3,)

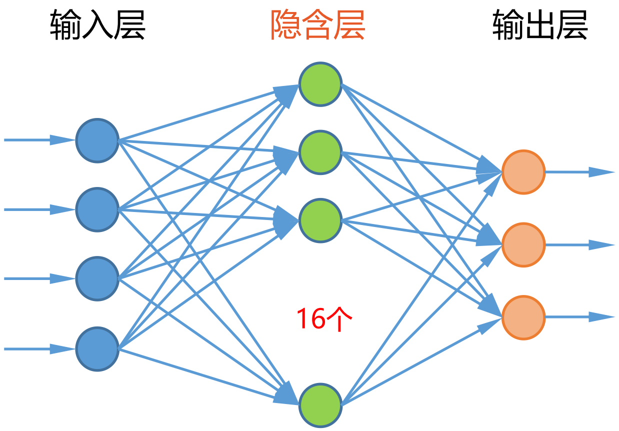
$H = X\,W_1 + B_1$　　$Y = H\,W_2 + B_2$

隐含层激活函数	relu函数
输出层激活函数	softmax函数
损失函数	交叉熵损失函数

西安科技大学
计算机科学与技术学院

12.6 实例：实现多层神经网络

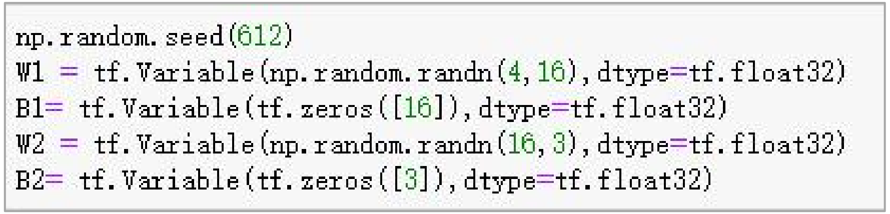
◻ 多层神经网络的**实现**



W_1 (4,16)　B_1 (16,)　W_2 (16,3)　B_2 (3,)

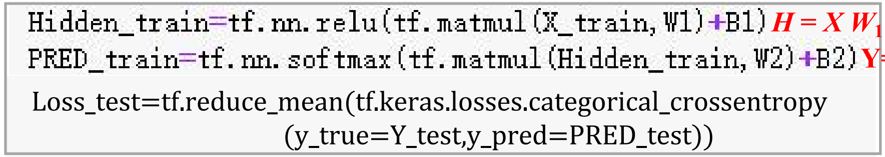
$H = X\,W_1 + B_1$　$Y = X\,W_2 + B_2$

■ 设置模型参数



```
np.random.seed(612)
W1 = tf.Variable(np.random.randn(4,16),dtype=tf.float32)
B1= tf.Variable(tf.zeros([16]),dtype=tf.float32)
W2 = tf.Variable(np.random.randn(16,3),dtype=tf.float32)
B2= tf.Variable(tf.zeros([3]),dtype=tf.float32)
```

■ 定义网络结构

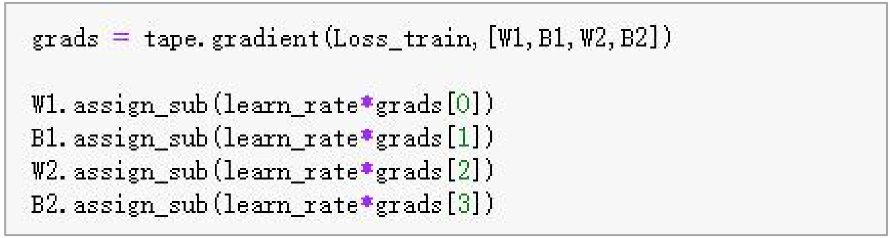


```
Hidden_train=tf.nn.relu(tf.matmul(X_train,W1)+B1)
PRED_train=tf.nn.softmax(tf.matmul(Hidden_train,W2)+B2)
Loss_test=tf.reduce_mean(tf.keras.losses.categorical_crossentropy
                         (y_true=Y_test,y_pred=PRED_test))
```

$H = X\,W_1 + B_1$

$Y=HW_2+B_2$

■ 更新模型参数

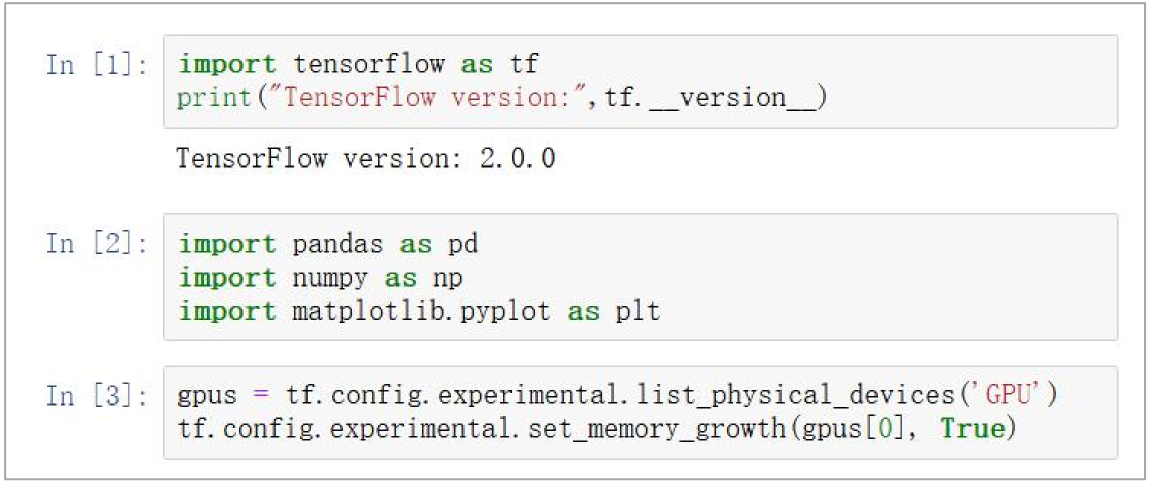


```
grads = tape.gradient(Loss_train,[W1,B1,W2,B2])

W1.assign_sub(learn_rate*grads[0])
B1.assign_sub(learn_rate*grads[1])
W2.assign_sub(learn_rate*grads[2])
B2.assign_sub(learn_rate*grads[3])
```

12.6 实例：实现多层神经网络

完整程序：多层神经网络实现鸢尾花分类

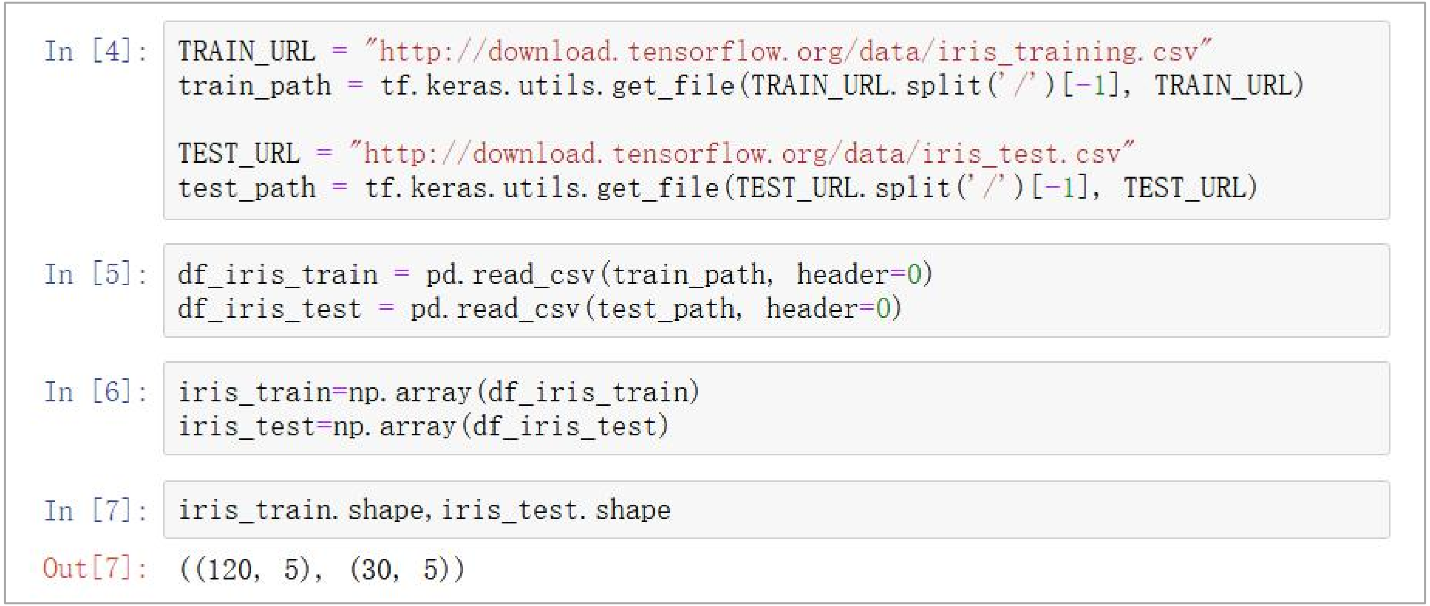
- 导入库，设置GPU模式



```
In [1]: import tensorflow as tf
        print("TensorFlow version:",tf.__version__)

        TensorFlow version: 2.0.0

In [2]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt

In [3]: gpus = tf.config.experimental.list_physical_devices('GPU')
        tf.config.experimental.set_memory_growth(gpus[0], True)
```

12.6 实例：实现多层神经网络

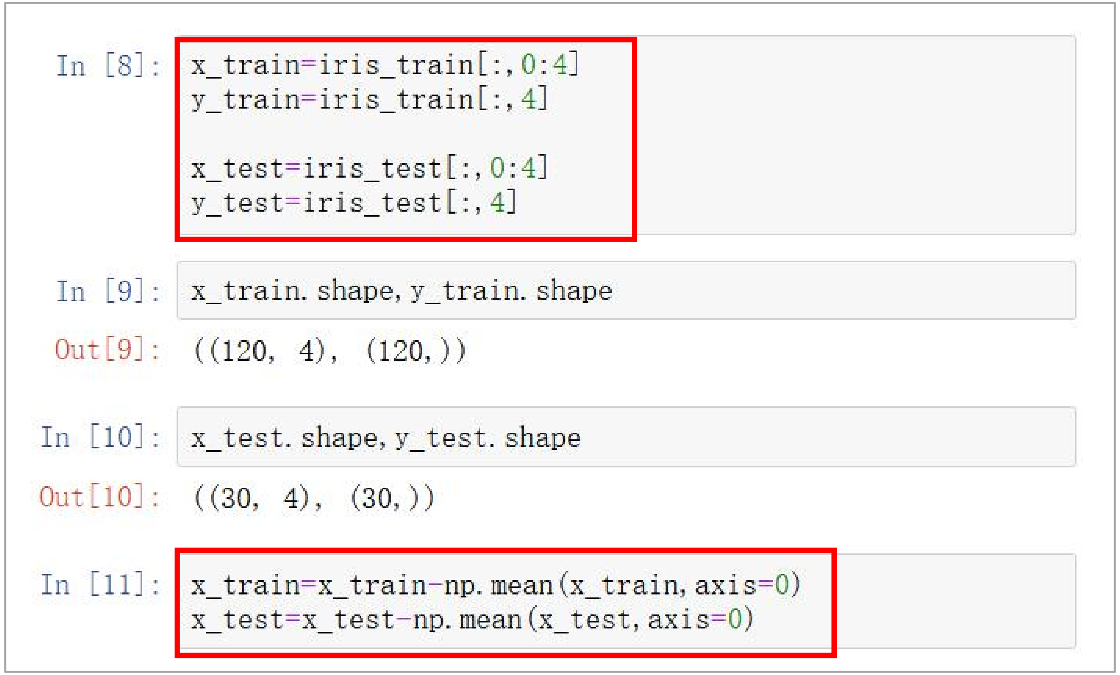
■ 加载数据，转换为NumPy数组



```
In [4]:  TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
         train_path = tf.keras.utils.get_file(TRAIN_URL.split('/')[-1], TRAIN_URL)

         TEST_URL = "http://download.tensorflow.org/data/iris_test.csv"
         test_path = tf.keras.utils.get_file(TEST_URL.split('/')[-1], TEST_URL)

In [5]:  df_iris_train = pd.read_csv(train_path, header=0)
         df_iris_test = pd.read_csv(test_path, header=0)

In [6]:  iris_train=np.array(df_iris_train)
         iris_test=np.array(df_iris_test)

In [7]:  iris_train.shape,iris_test.shape

Out[7]:  ((120, 5), (30, 5))
```

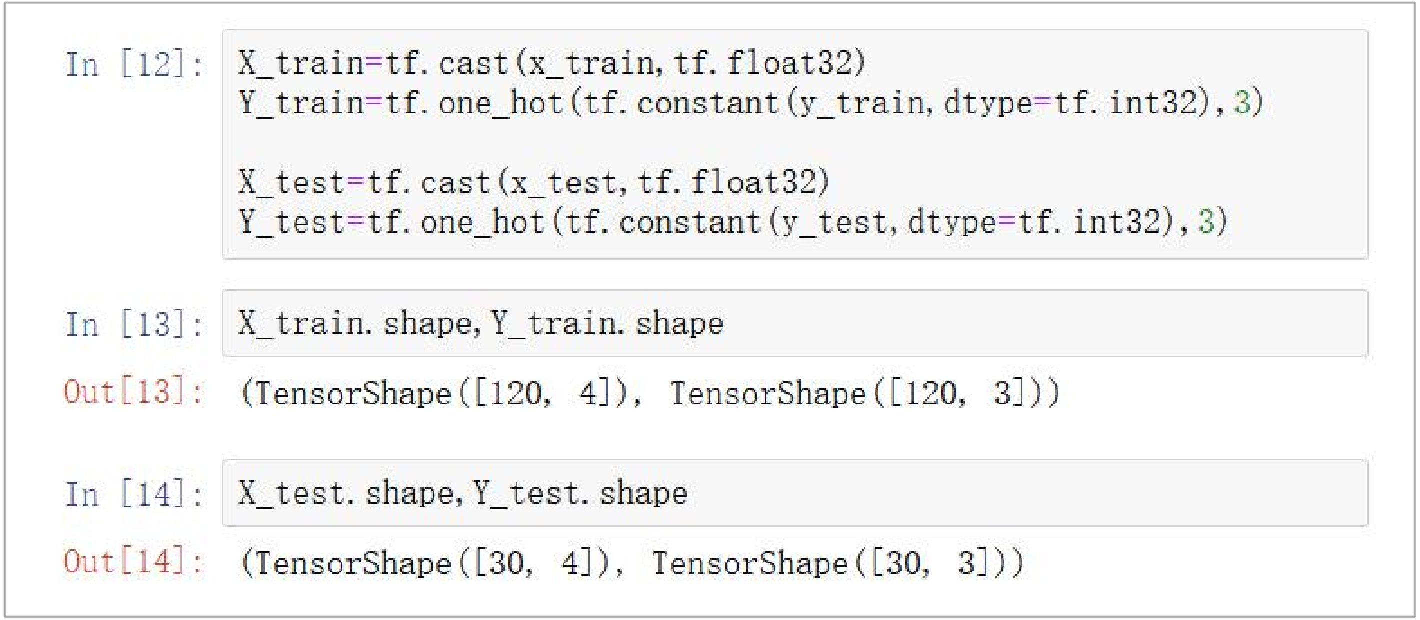
12.6 实例：实现多层神经网络

数据预处理



```
In [8]:  x_train=iris_train[:,0:4]
         y_train=iris_train[:,4]

         x_test=iris_test[:,0:4]
         y_test=iris_test[:,4]

In [9]:  x_train.shape,y_train.shape

Out[9]:  ((120, 4), (120,))

In [10]: x_test.shape,y_test.shape

Out[10]: ((30, 4), (30,))

In [11]: x_train=x_train-np.mean(x_train,axis=0)
         x_test=x_test-np.mean(x_test,axis=0)
```

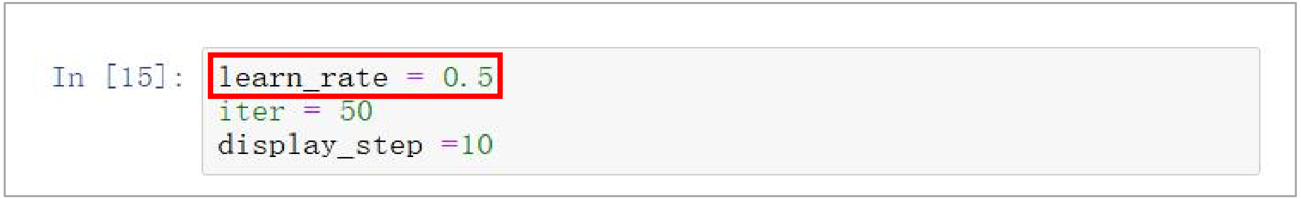
12.6 实例：实现多层神经网络



```
In [12]: X_train=tf.cast(x_train,tf.float32)
         Y_train=tf.one_hot(tf.constant(y_train,dtype=tf.int32),3)

         X_test=tf.cast(x_test,tf.float32)
         Y_test=tf.one_hot(tf.constant(y_test,dtype=tf.int32),3)

In [13]: X_train.shape,Y_train.shape

Out[13]: (TensorShape([120, 4]), TensorShape([120, 3]))

In [14]: X_test.shape,Y_test.shape

Out[14]: (TensorShape([30, 4]), TensorShape([30, 3]))
```

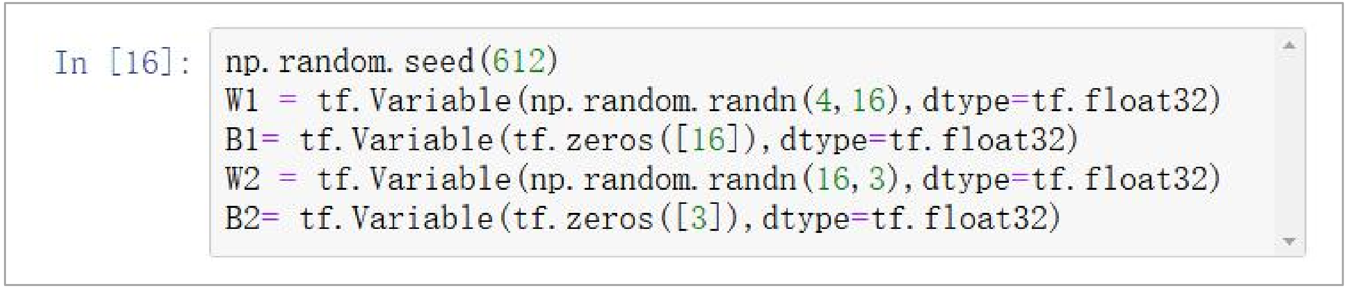
12.6 实例：实现多层神经网络

■ 设置超参数和显示间隔



```
In [15]: learn_rate = 0.5
         iter = 50
         display_step =10
```

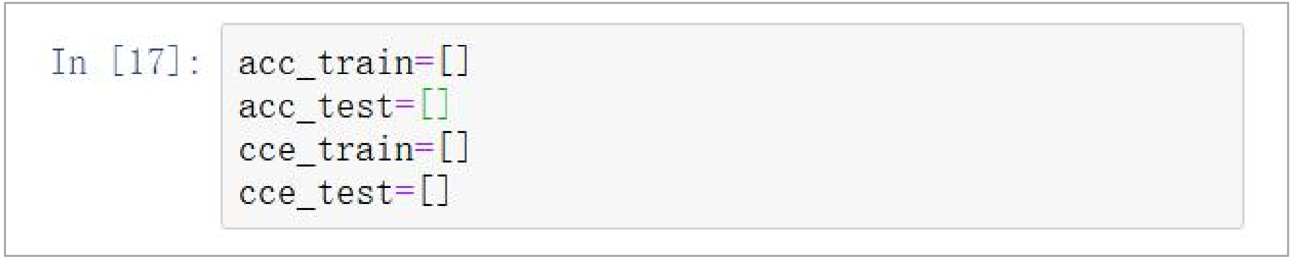
■ 设置模型参数初始值



```
In [16]: np.random.seed(612)
         W1 = tf.Variable(np.random.randn(4,16),dtype=tf.float32)
         B1= tf.Variable(tf.zeros([16]),dtype=tf.float32)
         W2 = tf.Variable(np.random.randn(16,3),dtype=tf.float32)
         B2= tf.Variable(tf.zeros([3]),dtype=tf.float32)
```

12.6 实例：实现多层神经网络

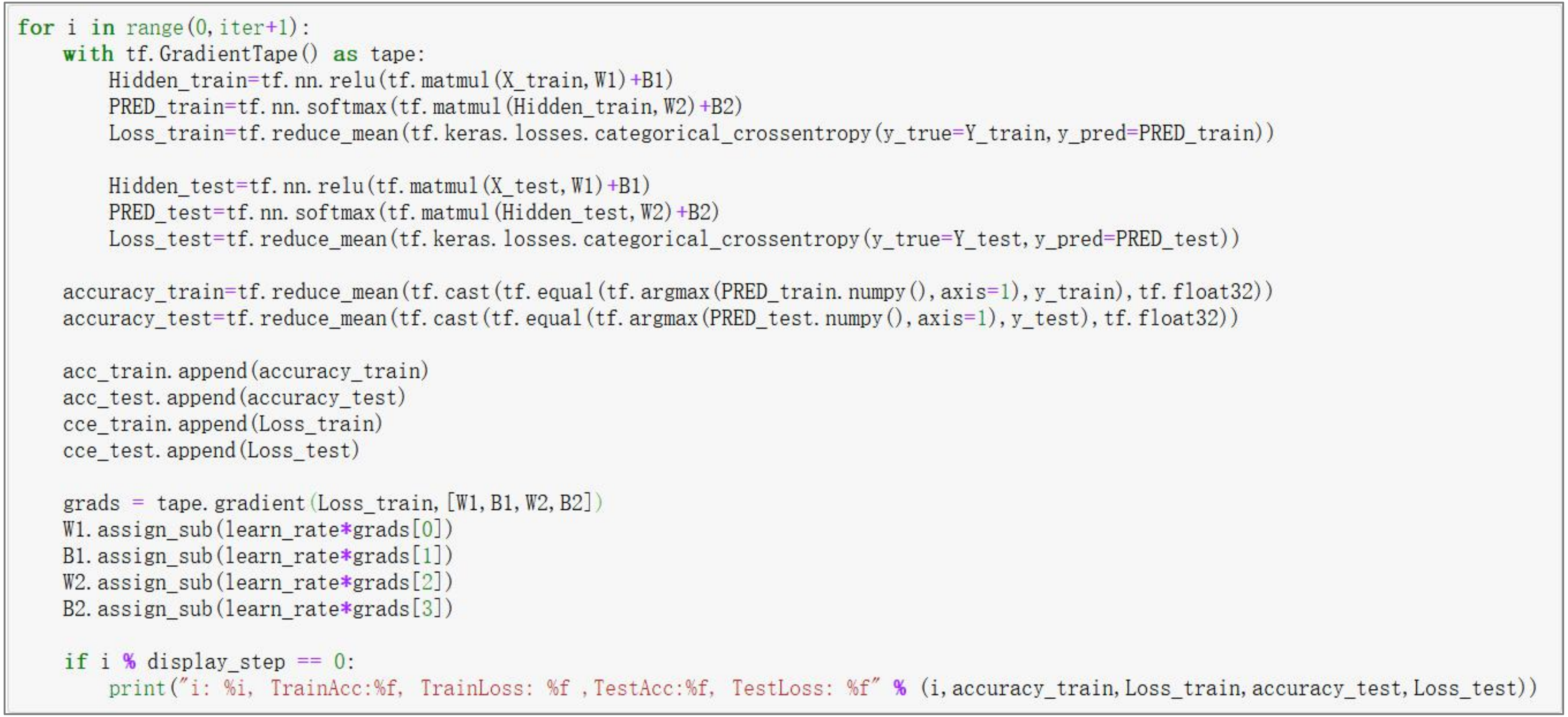
- 训练模型



In [17]:

```
acc_train=[]
acc_test=[]
cce_train=[]
cce_test=[]
```

12.6 实例：实现多层神经网络

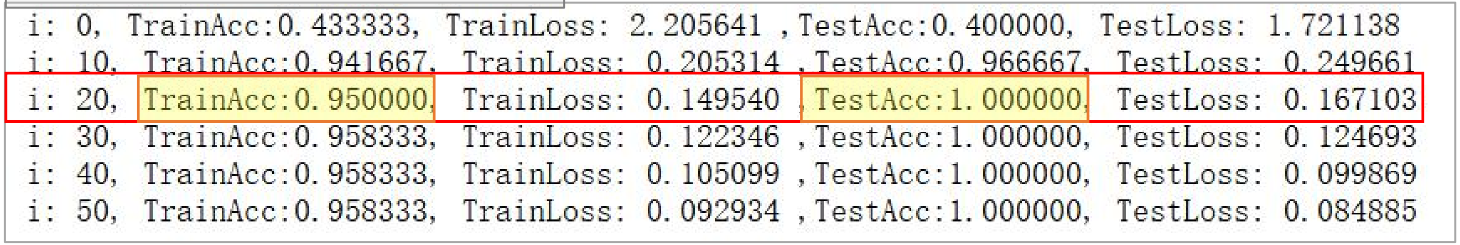


```
for i in range(0,iter+1):
    with tf.GradientTape() as tape:
        Hidden_train=tf.nn.relu(tf.matmul(X_train,W1)+B1)
        PRED_train=tf.nn.softmax(tf.matmul(Hidden_train,W2)+B2)
        Loss_train=tf.reduce_mean(tf.keras.losses.categorical_crossentropy(y_true=Y_train,y_pred=PRED_train))

        Hidden_test=tf.nn.relu(tf.matmul(X_test,W1)+B1)
        PRED_test=tf.nn.softmax(tf.matmul(Hidden_test,W2)+B2)
        Loss_test=tf.reduce_mean(tf.keras.losses.categorical_crossentropy(y_true=Y_test,y_pred=PRED_test))

    accuracy_train=tf.reduce_mean(tf.cast(tf.equal(tf.argmax(PRED_train.numpy(),axis=1),y_train),tf.float32))
    accuracy_test=tf.reduce_mean(tf.cast(tf.equal(tf.argmax(PRED_test.numpy(),axis=1),y_test),tf.float32))

    acc_train.append(accuracy_train)
    acc_test.append(accuracy_test)
    cce_train.append(Loss_train)
    cce_test.append(Loss_test)

    grads = tape.gradient(Loss_train,[W1,B1,W2,B2])
    W1.assign_sub(learn_rate*grads[0])
    B1.assign_sub(learn_rate*grads[1])
    W2.assign_sub(learn_rate*grads[2])
    B2.assign_sub(learn_rate*grads[3])

    if i % display_step == 0:
        print("i: %i, TrainAcc:%f, TrainLoss: %f ,TestAcc:%f, TestLoss: %f" % (i,accuracy_train,Loss_train,accuracy_test,Loss_test))
```

12.6 实例：实现多层神经网络

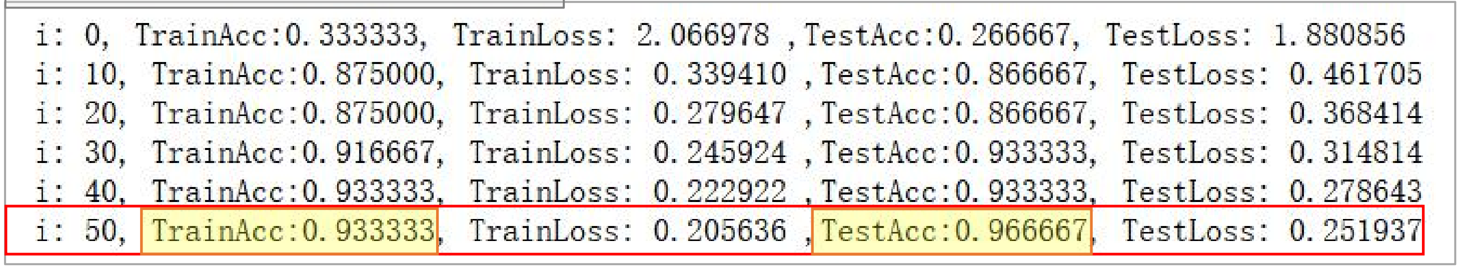
训练结果

learn_rate=0.5，2层神经网络



```
i: 0,  TrainAcc:0.433333,  TrainLoss: 2.205641 ,TestAcc:0.400000,  TestLoss: 1.721138
i: 10,  TrainAcc:0.941667,  TrainLoss: 0.205314 ,TestAcc:0.966667,  TestLoss: 0.249661
i: 20,  TrainAcc:0.950000,  TrainLoss: 0.149540 ,TestAcc:1.000000,  TestLoss: 0.167103
i: 30,  TrainAcc:0.958333,  TrainLoss: 0.122346 ,TestAcc:1.000000,  TestLoss: 0.124693
i: 40,  TrainAcc:0.958333,  TrainLoss: 0.105099 ,TestAcc:1.000000,  TestLoss: 0.099869
i: 50,  TrainAcc:0.958333,  TrainLoss: 0.092934 ,TestAcc:1.000000,  TestLoss: 0.084885
```

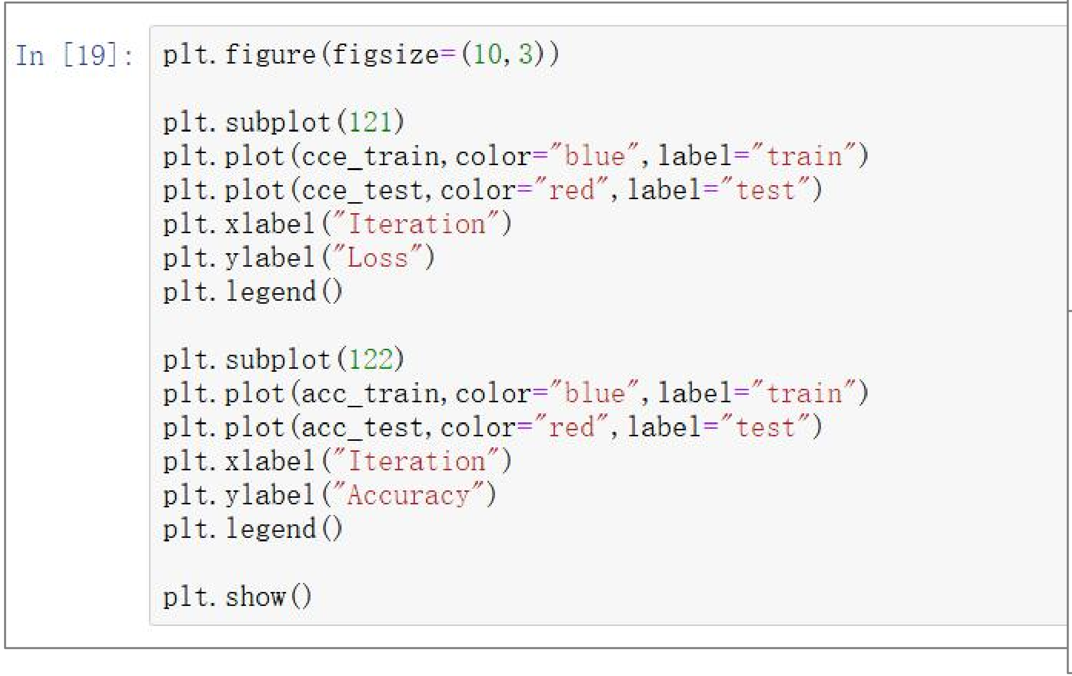
learn_rate=0.5，单层神经网络



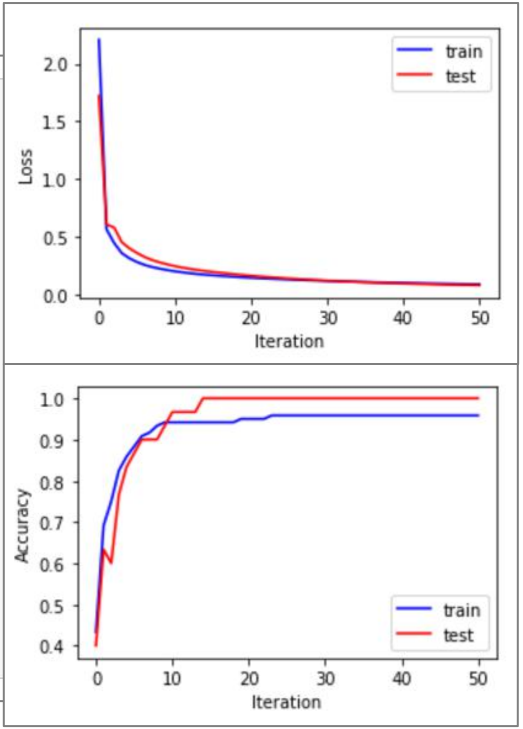
```
i: 0,  TrainAcc:0.333333,  TrainLoss: 2.066978 ,TestAcc:0.266667,  TestLoss: 1.880856
i: 10,  TrainAcc:0.875000,  TrainLoss: 0.339410 ,TestAcc:0.866667,  TestLoss: 0.461705
i: 20,  TrainAcc:0.875000,  TrainLoss: 0.279647 ,TestAcc:0.866667,  TestLoss: 0.368414
i: 30,  TrainAcc:0.916667,  TrainLoss: 0.245924 ,TestAcc:0.933333,  TestLoss: 0.314814
i: 40,  TrainAcc:0.933333,  TrainLoss: 0.222922 ,TestAcc:0.933333,  TestLoss: 0.278643
i: 50,  TrainAcc:0.933333,  TrainLoss: 0.205636 ,TestAcc:0.966667,  TestLoss: 0.251937
```

12.6 实例：实现多层神经网络

■ 结果可视化



```
In [19]: plt.figure(figsize=(10,3))

         plt.subplot(121)
         plt.plot(cce_train,color="blue",label="train")
         plt.plot(cce_test,color="red",label="test")
         plt.xlabel("Iteration")
         plt.ylabel("Loss")
         plt.legend()

         plt.subplot(122)
         plt.plot(acc_train,color="blue",label="train")
         plt.plot(acc_test,color="red",label="test")
         plt.xlabel("Iteration")
         plt.ylabel("Accuracy")
         plt.legend()

         plt.show()
```


Loss
Iteration
train
test
Accuracy
Iteration
train
test