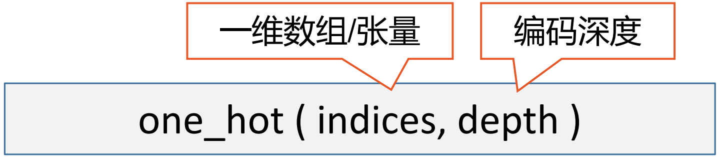
11.6 实例：实现多分类问题

中国大学MOOC

11.6 实例：实现多分类问题

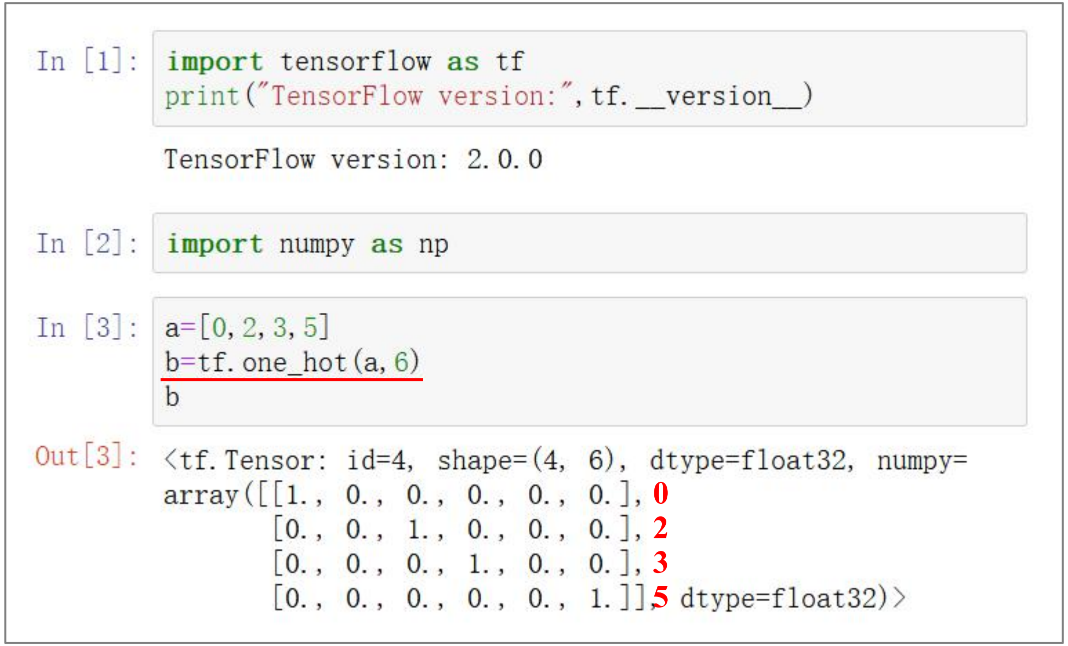
■ 独热编码



一维数组/张量

编码深度

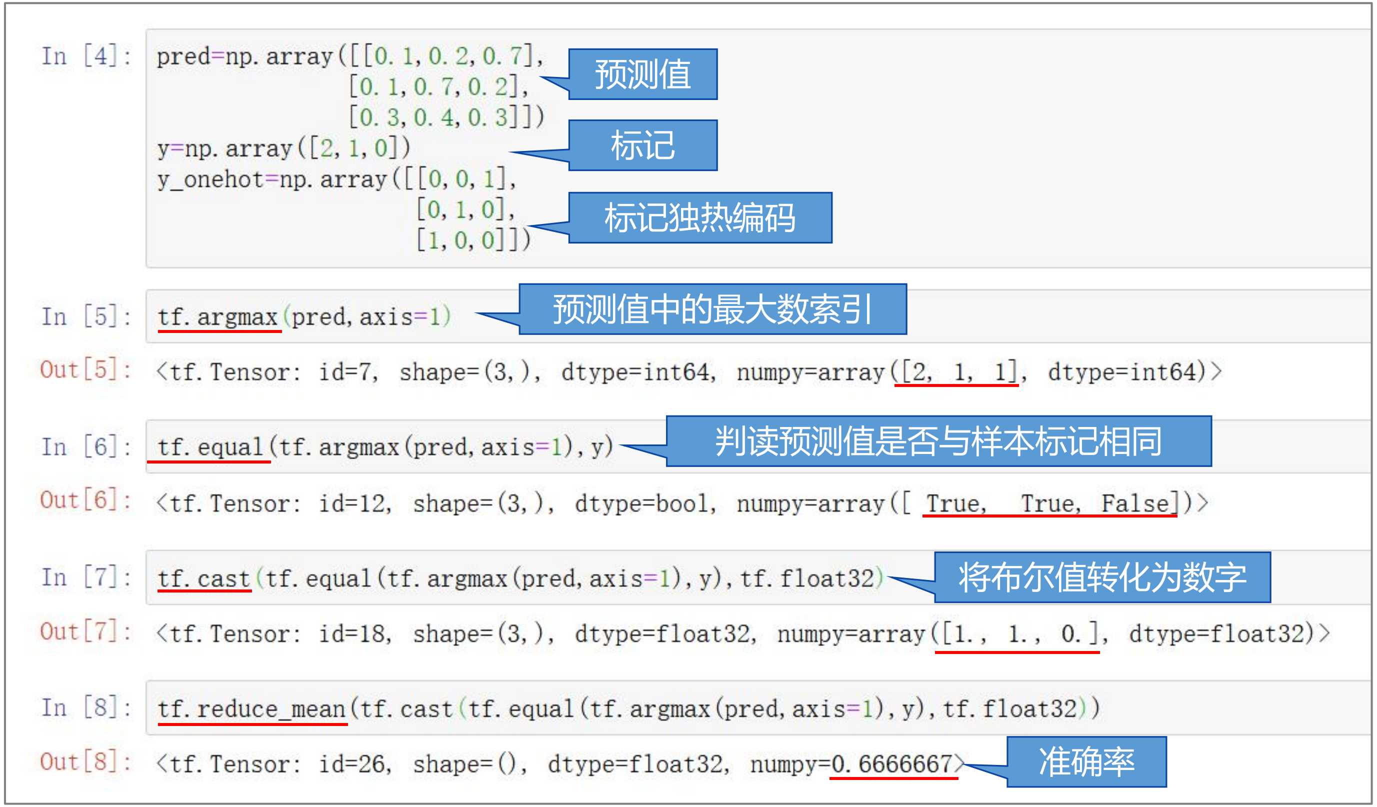
one_hot (indices, depth)



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

         TensorFlow version: 2.0.0

In [2]:  import numpy as np

In [3]:  a=[0,2,3,5]
         b=tf.one_hot(a,6)
         b

Out[3]:  <tf.Tensor: id=4, shape=(4, 6), dtype=float32, numpy=
         array([[1., 0., 0., 0., 0., 0.], 0
                [0., 0., 1., 0., 0., 0.], 2
                [0., 0., 0., 1., 0., 0.], 3
                [0., 0., 0., 0., 0., 1.]],5 dtype=float32)>
```

11.6 实例：实现多分类问题

■ 准确率



```
In [4]:  pred=np.array([[0.1,0.2,0.7],
                        [0.1,0.7,0.2],
                        [0.3,0.4,0.3]])
         y=np.array([2,1,0])
         y_onehot=np.array([[0,0,1],
                            [0,1,0],
                            [1,0,0]])
```

预测值

标记

标记独热编码

```
In [5]:  tf.argmax(pred,axis=1)
Out[5]:  <tf.Tensor: id=7, shape=(3,), dtype=int64, numpy=array([2, 1, 1], dtype=int64)>
```

预测值中的最大数索引

```
In [6]:  tf.equal(tf.argmax(pred,axis=1),y)
Out[6]:  <tf.Tensor: id=12, shape=(3,), dtype=bool, numpy=array([ True,  True, False])>
```

判读预测值是否与样本标记相同

```
In [7]:  tf.cast(tf.equal(tf.argmax(pred,axis=1),y),tf.float32)
Out[7]:  <tf.Tensor: id=18, shape=(3,), dtype=float32, numpy=array([1., 1., 0.], dtype=float32)>
```

将布尔值转化为数字

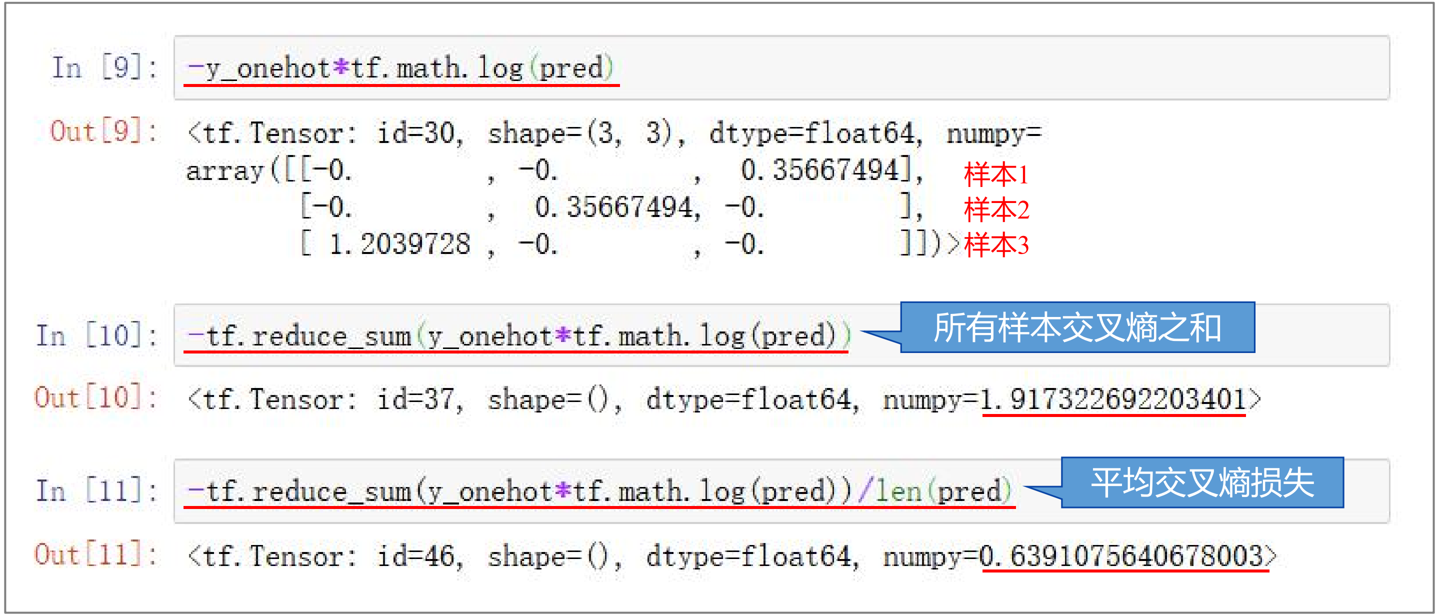
```
In [8]:  tf.reduce_mean(tf.cast(tf.equal(tf.argmax(pred,axis=1),y),tf.float32))
Out[8]:  <tf.Tensor: id=26, shape=(), dtype=float32, numpy=0.6666667>
```

准确率

11.6 实例：实现多分类问题

■ 交叉熵损失函数

$$Loss = -\sum_{i=1}^{n}\sum_{p=1}^{C} y_{i,p}\ln(\hat{y}_{i,p})$$



```
In [9]: -y_onehot*tf.math.log(pred)

Out[9]: <tf.Tensor: id=30, shape=(3, 3), dtype=float64, numpy=
        array([[-0.        , -0.        ,  0.35667494],
               [-0.        ,  0.35667494, -0.        ],
               [ 1.2039728 , -0.        , -0.        ]])>
```

样本1

样本2

样本3

```
In [10]: -tf.reduce_sum(y_onehot*tf.math.log(pred))

Out[10]: <tf.Tensor: id=37, shape=(), dtype=float64, numpy=1.917322692203401>
```

所有样本交叉熵之和

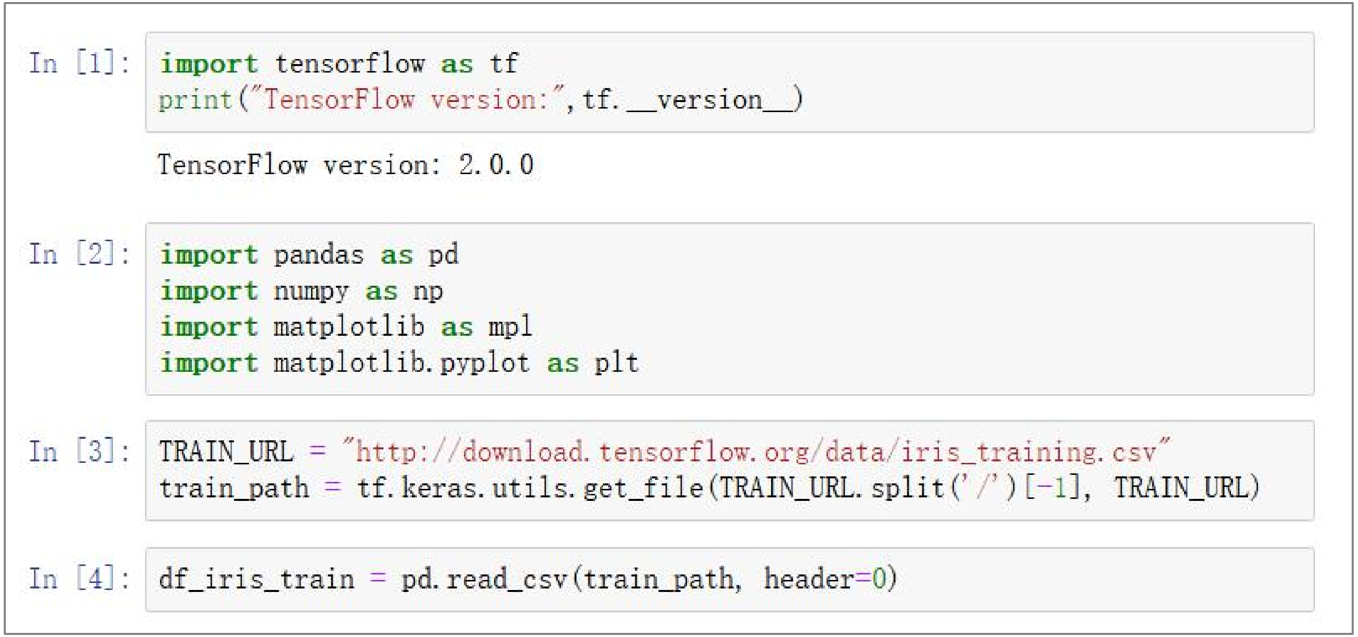
```
In [11]: -tf.reduce_sum(y_onehot*tf.math.log(pred))/len(pred)

Out[11]: <tf.Tensor: id=46, shape=(), dtype=float64, numpy=0.6391075640678003>
```

平均交叉熵损失

11.6 实例：实现多分类问题

例：使用**花瓣长度**、**花瓣宽度**将**三种**鸢尾花区分开

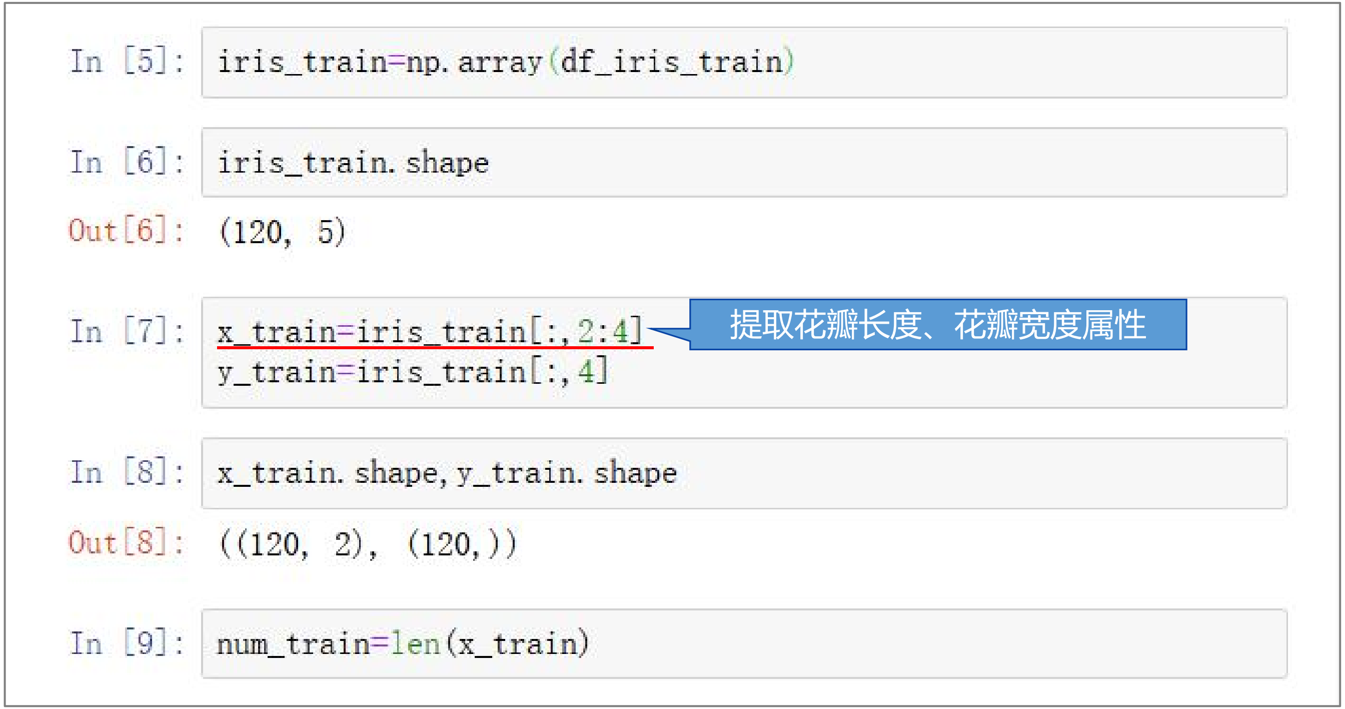
□ 加载数据



```
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 as mpl
        import matplotlib.pyplot as plt

In [3]: TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
        train_path = tf.keras.utils.get_file(TRAIN_URL.split('/')[-1], TRAIN_URL)

In [4]: df_iris_train = pd.read_csv(train_path, header=0)
```

11.6 实例：实现多分类问题

□ 处理数据

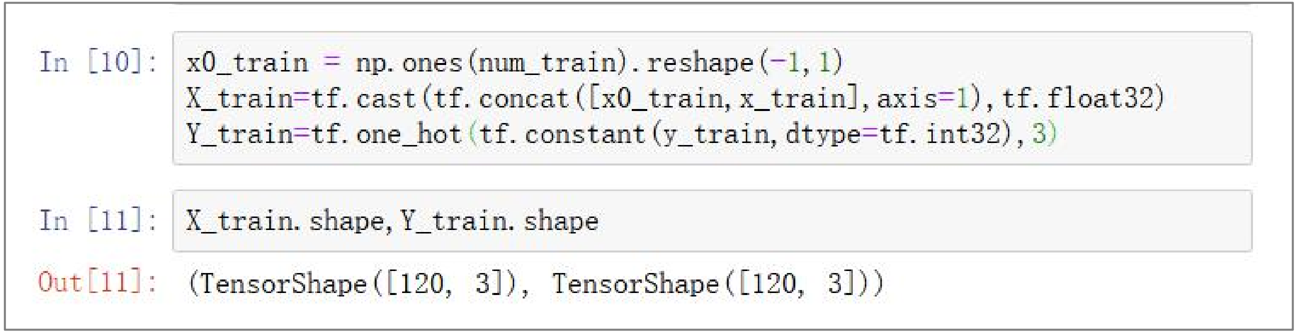


```
In [5]: iris_train=np.array(df_iris_train)

In [6]: iris_train.shape

Out[6]: (120, 5)

In [7]: x_train=iris_train[:,2:4]
        y_train=iris_train[:,4]

In [8]: x_train.shape,y_train.shape

Out[8]: ((120, 2), (120,))

In [9]: num_train=len(x_train)
```

提取花瓣长度、花瓣宽度属性

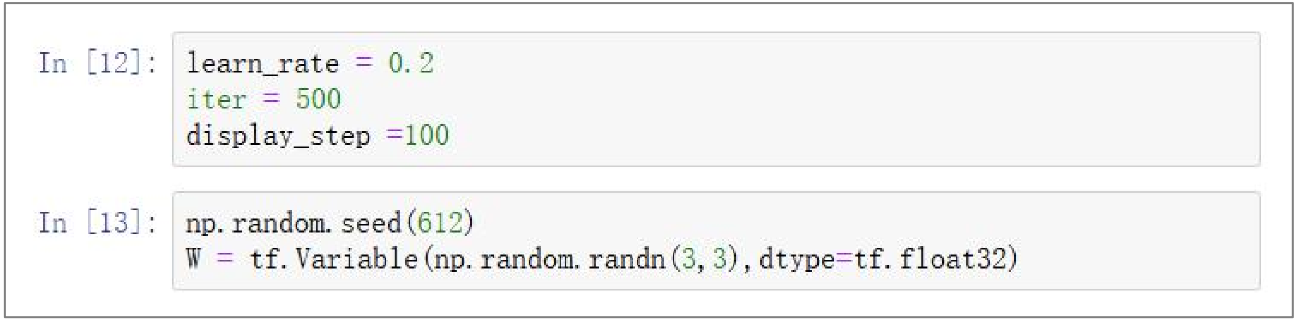
11.6 实例：实现多分类问题

□ 处理数据



```
In [10]: x0_train = np.ones(num_train).reshape(-1,1)
         X_train=tf.cast(tf.concat([x0_train,x_train],axis=1),tf.float32)
         Y_train=tf.one_hot(tf.constant(y_train,dtype=tf.int32),3)

In [11]: X_train.shape,Y_train.shape

Out[11]: (TensorShape([120, 3]), TensorShape([120, 3]))
```

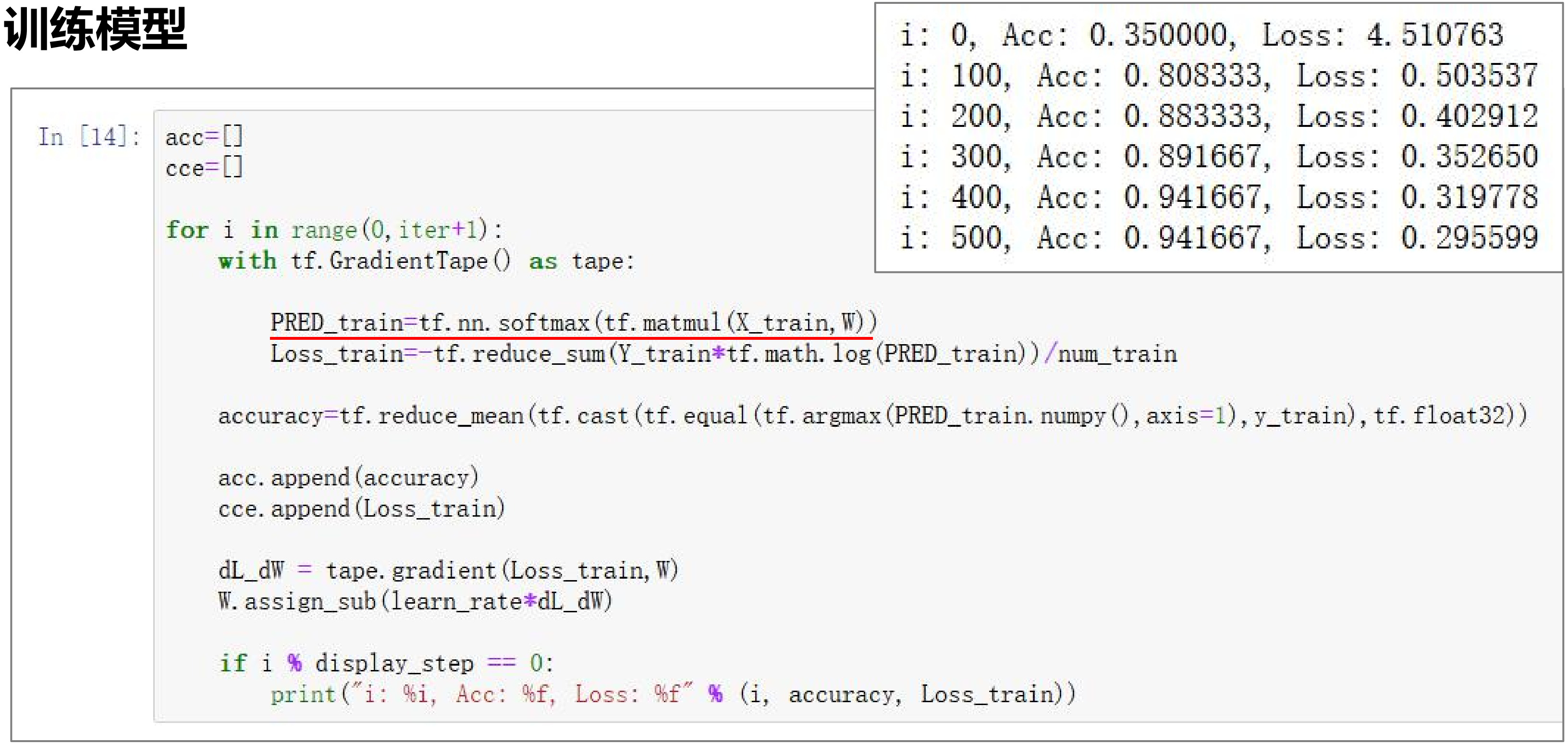
□ 设置超参数、设置模型参数初始值



```
In [12]: learn_rate = 0.2
         iter = 500
         display_step =100

In [13]: np.random.seed(612)
         W = tf.Variable(np.random.randn(3,3),dtype=tf.float32)
```

11.6 实例：实现多分类问题

□ 训练模型

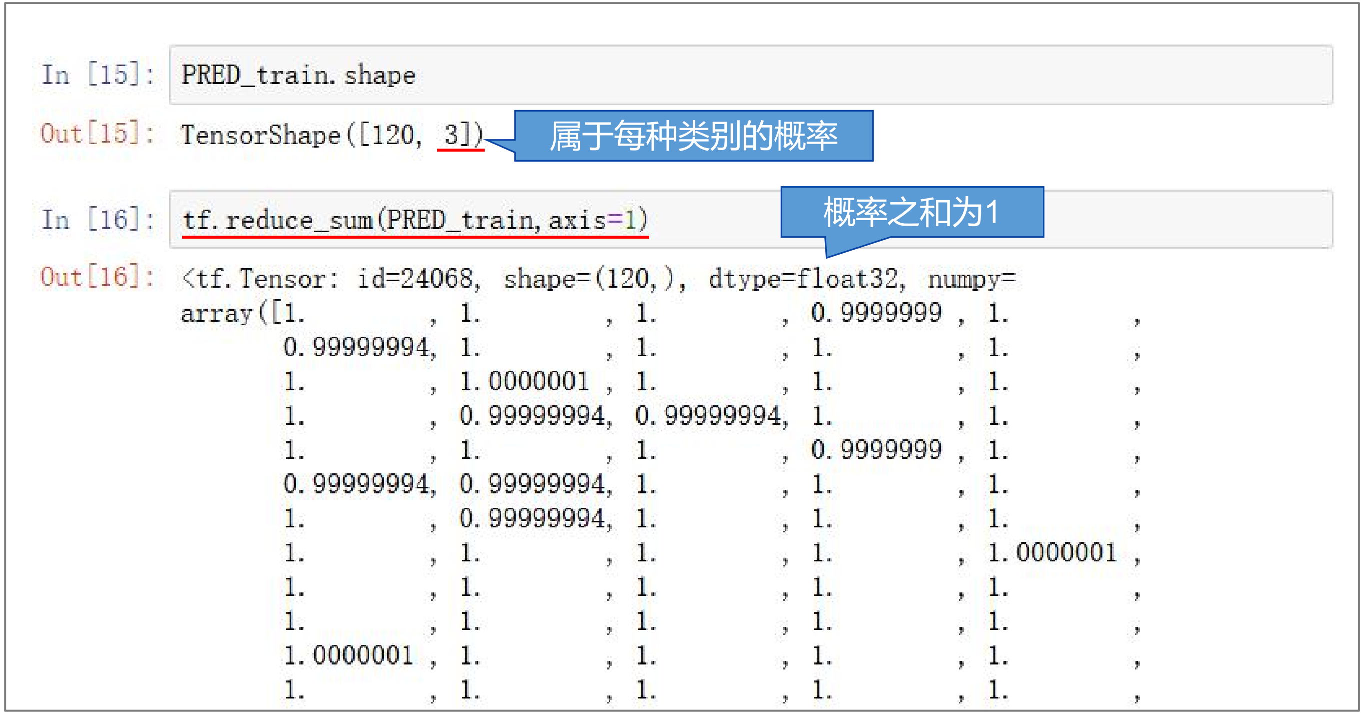


```
i: 0, Acc: 0.350000, Loss: 4.510763
i: 100, Acc: 0.808333, Loss: 0.503537
i: 200, Acc: 0.883333, Loss: 0.402912
i: 300, Acc: 0.891667, Loss: 0.352650
i: 400, Acc: 0.941667, Loss: 0.319778
i: 500, Acc: 0.941667, Loss: 0.295599
```

```
In [14]: acc=[]
         cce=[]

         for i in range(0,iter+1):
             with tf.GradientTape() as tape:

                 PRED_train=tf.nn.softmax(tf.matmul(X_train,W))
                 Loss_train=-tf.reduce_sum(Y_train*tf.math.log(PRED_train))/num_train

             accuracy=tf.reduce_mean(tf.cast(tf.equal(tf.argmax(PRED_train.numpy(),axis=1),y_train),tf.float32))

             acc.append(accuracy)
             cce.append(Loss_train)

             dL_dW = tape.gradient(Loss_train,W)
             W.assign_sub(learn_rate*dL_dW)

             if i % display_step == 0:
                 print("i: %i, Acc: %f, Loss: %f" % (i, accuracy, Loss_train))
```

11.6 实例：实现多分类问题

□ 训练结果



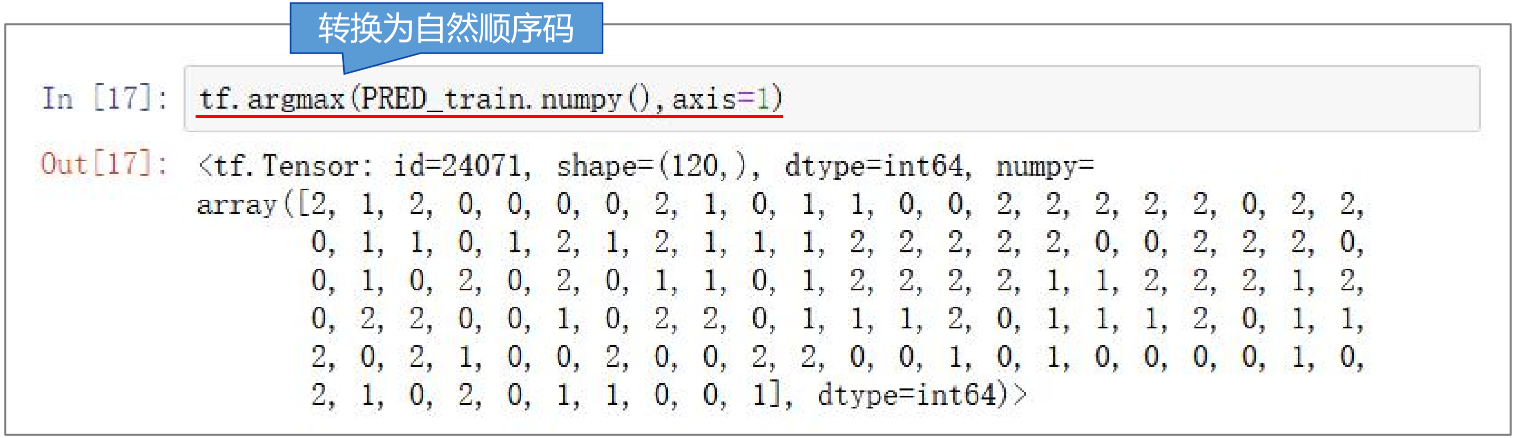
```
In [15]: PRED_train.shape

Out[15]: TensorShape([120, 3])

In [16]: tf.reduce_sum(PRED_train,axis=1)

Out[16]: <tf.Tensor: id=24068, shape=(120,), dtype=float32, numpy=
         array([1.        , 1.        , 1.        , 0.9999999 , 1.        ,
                0.99999994, 1.        , 1.        , 1.        , 1.        ,
                1.        , 1.0000001 , 1.        , 1.        , 1.        ,
                1.        , 0.99999994, 0.99999994, 1.        , 1.        ,
                1.        , 1.        , 1.        , 0.9999999 , 1.        ,
                0.99999994, 0.99999994, 1.        , 1.        , 1.        ,
                1.        , 0.99999994, 1.        , 1.        , 1.        ,
                1.        , 1.        , 1.        , 1.        , 1.0000001 ,
                1.        , 1.        , 1.        , 1.        , 1.        ,
                1.        , 1.        , 1.        , 1.        , 1.        ,
                1.0000001 , 1.        , 1.        , 1.        , 1.        ,
                1.        , 1.        , 1.        , 1.        , 1.        ,
```

属于每种类别的概率

概率之和为1

11.6 实例：实现多分类问题

□ 训练结果

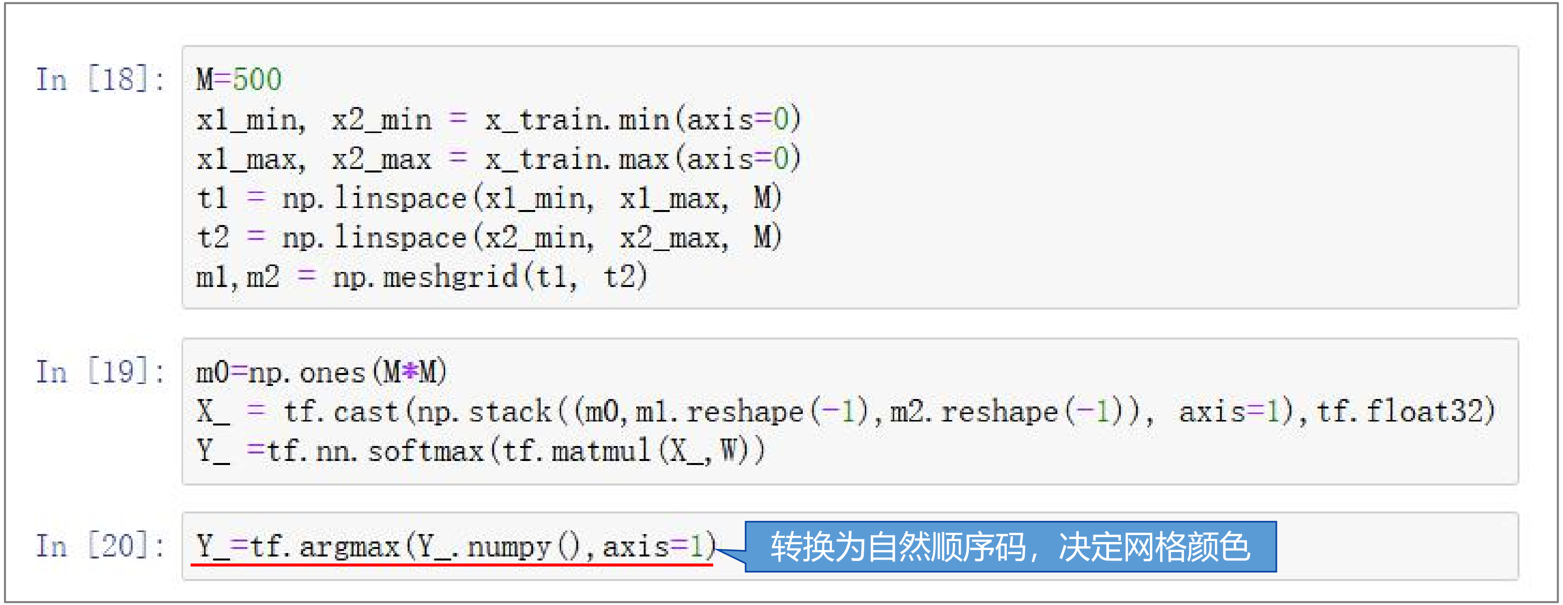


转换为自然顺序码

```
In [17]:  tf.argmax(PRED_train.numpy(),axis=1)

Out[17]:  <tf.Tensor: id=24071, shape=(120,), dtype=int64, numpy=
          array([2, 1, 2, 0, 0, 0, 0, 2, 1, 0, 1, 1, 0, 0, 2, 2, 2, 2, 2, 0, 2, 2,
                 0, 1, 1, 0, 1, 2, 1, 2, 1, 1, 1, 2, 2, 2, 2, 2, 0, 0, 2, 2, 2, 0,
                 0, 1, 0, 2, 0, 2, 0, 1, 1, 0, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 1, 2,
                 0, 2, 2, 0, 0, 1, 0, 2, 2, 0, 1, 1, 1, 2, 0, 1, 1, 1, 2, 0, 1, 1,
                 2, 0, 2, 1, 0, 0, 2, 0, 0, 2, 2, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0,
                 2, 1, 0, 2, 0, 1, 1, 0, 0, 1], dtype=int64)>
```

11.6 实例：实现多分类问题

□ 绘制分类图

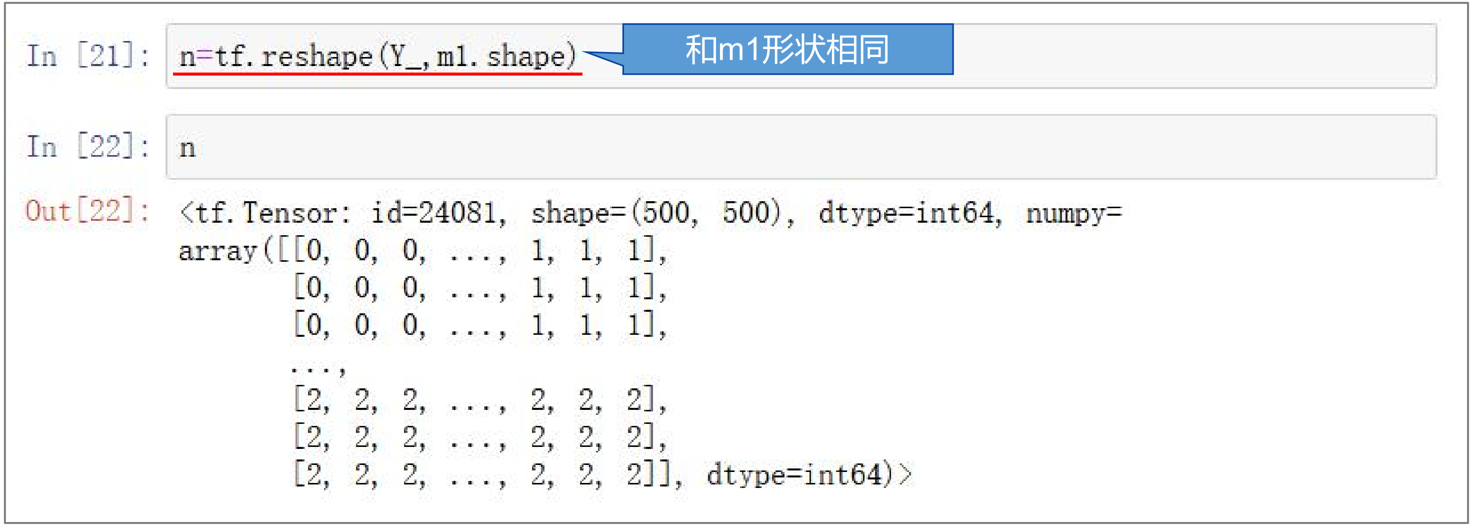


```
In [18]: M=500
         x1_min, x2_min = x_train.min(axis=0)
         x1_max, x2_max = x_train.max(axis=0)
         t1 = np.linspace(x1_min, x1_max, M)
         t2 = np.linspace(x2_min, x2_max, M)
         m1,m2 = np.meshgrid(t1, t2)

In [19]: m0=np.ones(M*M)
         X_ = tf.cast(np.stack((m0,m1.reshape(-1),m2.reshape(-1)), axis=1),tf.float32)
         Y_ =tf.nn.softmax(tf.matmul(X_,W))

In [20]: Y_=tf.argmax(Y_.numpy(),axis=1)
```

转换为自然顺序码，决定网格颜色

11.6 实例：实现多分类问题

□ 绘制分类图



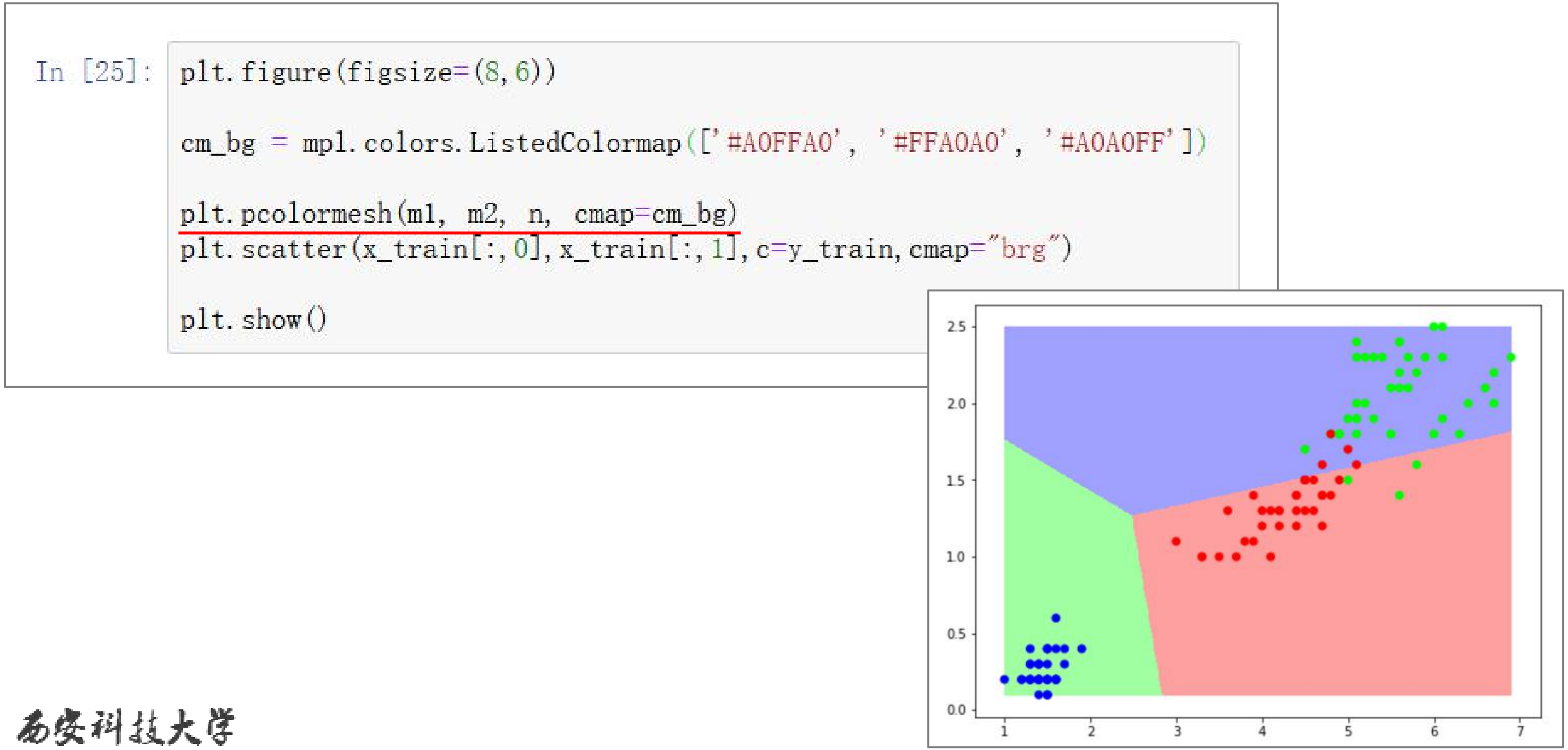
```
In [21]: n=tf.reshape(Y_,m1.shape)
```

和m1形状相同

```
In [22]: n

Out[22]: <tf.Tensor: id=24081, shape=(500, 500), dtype=int64, numpy=
         array([[0, 0, 0, ..., 1, 1, 1],
                [0, 0, 0, ..., 1, 1, 1],
                [0, 0, 0, ..., 1, 1, 1],
                ...,
                [2, 2, 2, ..., 2, 2, 2],
                [2, 2, 2, ..., 2, 2, 2],
                [2, 2, 2, ..., 2, 2, 2]], dtype=int64)>
```

11.6 实例：实现多分类问题

□ 绘制分类图



```
In [25]: plt.figure(figsize=(8,6))

         cm_bg = mpl.colors.ListedColormap(['#A0FFA0', '#FFA0A0', '#A0A0FF'])

         plt.pcolormesh(m1, m2, n, cmap=cm_bg)
         plt.scatter(x_train[:,0],x_train[:,1],c=y_train,cmap="brg")

         plt.show()
```