神经网络和机器学习之鸢尾花分类

案例说明：鸢尾花(Iris)分类，使用全连接神经网络层。

鸢(yuān)尾花分类相当于机器学习中的Helloworld问题。鸢尾花可以分为很多类，一般通过花萼长度、花萼宽度、花瓣长度、花瓣宽度进行区分。我们让机器来学习关于这个鸢尾花分类的一组数据，然后建立模型，训练。后面直接给出花的四个特征，让机器判断花的分类。

案例选择了keras框架，需要先安装keras和tensorflow。虚谷号教育版已经预装必要的库，可以直接使用。

本案例已经提供了训练好的模型，放在model文件夹中，文件名称为：2-model-vv.h5。如果想直接测试模型，请跳到“导入模型”环节，输入数据开始识别。

1.环境搭建

下面是安装命令：

pip install keras

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple (https://pypi.tuna.tsinghua.edu.cn/simple) tensorflow

建议选择清华源，速度将快很多。参考命令如下：

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple (https://pypi.tuna.tsinghua.edu.cn/simple) tensorflow

2.数据准备

鸢尾花分类数据集在 `data` 中，文件名称为 `iris.csv` 。数据分为5列，前4列为花萼长度，花萼宽度，花瓣长度，花瓣宽度等4个用于识别鸢尾花的属性，第5列为鸢尾花的类别（包括Setosa，Versicolour，Virginica三类）。

这个数据集可以从UCI数据集上直接下载，具体地址为：http://archive.ics.uci.edu/ml/datasets/Iris (http://archive.ics.uci.edu/ml/datasets/Iris) 打开页面后点击Datafolder就可以下载到本地磁盘上，默认格式为逗号分隔的文本文件。

也可以直接从sklearn包里datasets里导入，语法为：from sklearn.datasets import load_iris。 如果从本地磁盘上读入该数据集，可以采用pandas包里的read_excel或者read_csv方法，也可以利用python里面的csv包来处理。

开始导入数据集吧。

In [2]:

```
import pandas as pd
data=pd.read_csv('./data/iris.csv')
```

该问题属于比较典型的多分类问题，因此在训练数据预处理中，首先对分类结果标签"Species"进行独热编码化。所谓独热编码(One-Hot)，是指用0/1构成的数组来表示一种情况，比如在鸢尾花分类中，顺序编码可以用0、1、2来表示不同的鸢尾花品种，而独热编码可以用[1,0,0]表示setosa，用[0,1,0]表示versicolor,用[0,0,1]表示virginica。独热编码相对于顺序编码避免了神经网络把没有数值大小意义的数据错误的理解为有数值意义。比如如果用顺序编码来表示鸢尾花品种，神经网络会错误的认为2表示的品种与0表示的品种之间的差距比较大，而与1表示的品种差距比较小。

In [54]:

```
data=pd.get_dummies(data,columns=['Species']) #把种类(列名称为“Species”)进行独热编码
x=data[['Sepal.Length', 'Sepal.Width', 'Petal.Length','Petal.Width']]
y=data.iloc[:,-3:]
```

此时x与y的形状分别是(150,3)和(150,1)，即x具有150行、3列，y具有150行、1列。其中x是输入的数据(鸢尾花的属性)，y是输出的结果(鸢尾花的类别)。 输出来看一下，你会发现x与y的行数一定是相同的。

In [55]:

```
data.head()
```

Out[55]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species_Iris-setosa	Species_Iris-versicolor	Species_Iris-virginica
0	4.6	3.1	1.5	0.2	1	0	0
1	5.5	2.6	4.4	1.2	0	1	0
2	6.1	2.6	5.6	1.4	0	0	1
3	7.2	3.0	5.8	1.6	0	0	1
4	5.2	3.4	1.4	0.2	1	0	0

In [56]:

```
data.tail()
```

Out[56]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species_Iris-setosa	Species_Iris-versicolor	Species_Ir virgini
145	6.9	3.1	5.4	2.1	0	0	
146	6.0	3.0	4.8	1.8	0	0	
147	6.9	3.1	5.1	2.3	0	0	
148	5.1	3.7	1.5	0.4	1	0	
149	5.4	3.9	1.7	0.4	1	0	

In [5]:

```
len(x),len(y)
```

Out[5]:

```
(150, 150)
```

3.建立模型

多分类问题是二分类问题的扩展。当分类数大于2时，就是多分类问题。比如把笔分成铅笔、圆珠笔、钢笔等等，就是多分类问题。多分类问题需要神经网络将最后一层神经元个数设置为与分类数目相同以输出一个数组，这个

就是多分类问题。多分类问题需要神经网络有最后一层神经元个数设置为与分类数目相同以输出一个数组，这个数组的长度就是分类数目，数组中每个数值对应在不同类别上的可能性。一般的，多分类问题通过softmax函数激活，损失函数使用类别交叉熵损失(categorical_crossentropy)。

keras支持很多类型的神经网络层，这里使用add方法添加2个全连接神经网络层（Dense层）。第一层通过input_dim参数指定接收输入数据的维度为4（鸢尾花的属性），units=8表示将这个4维数据全连接到8个神经元，activation定义了激活函数为relu。第二层神经元，也就是最后一层神经元的个数设置要和分类的数目相同，所以设置为3，激活函数为softmax。

代码如下：

In [6]:

```
import keras
from keras import layers
```

```
Using TensorFlow backend.
```

In [7]:

```
model=keras.models.Sequential()
model.add(layers.Dense(units=8, input_dim=4, activation='relu'))
model.add(layers.Dense(units=3, activation='softmax'))
```

定义好模型的层之后，需要对模型进行编译，同时指定训练模型所需要的优化器以及损失的估算方法。在keras中，可以通过optimizer参数来指定优化器。这里选择了adam。loss定义了损失函数为category_crossentropy。

In [8]:

```
model.compile(optimizer='adam',loss='categorical_crossentropy')
```

4.训练模型

最后对模型进行训练，一下代码利用现有数据x和y对模型进行训练500次，epochs表示训练轮次，batch_size表示每次有多少行数据参与训练，最后把整个训练过程记录到history中。程序运行后，在控制台会打印出每轮次的训练情况。

In [9]:

```
history=model.fit(x,y,batch_size=150,epochs=500)
```

```
Epoch 1/500
150/150 [==============================] - 1s 8ms/step - loss: 2.4921
Epoch 2/500
150/150 [==============================] - 0s 82us/step - loss: 2.474
6
Epoch 3/500
150/150 [==============================] - 0s 72us/step - loss: 2.454
5
Epoch 4/500
150/150 [==============================] - 0s 57us/step - loss: 2.431
2
Epoch 5/500
150/150 [==============================] - 0s 126us/step - loss: 2.40
80
Epoch 6/500
150/150 [==============================] - 0s 57us/step - loss: 2.384
9
Epoch 7/500
150/150 [==============================] - 0s 56us/step - loss: 2.362
```

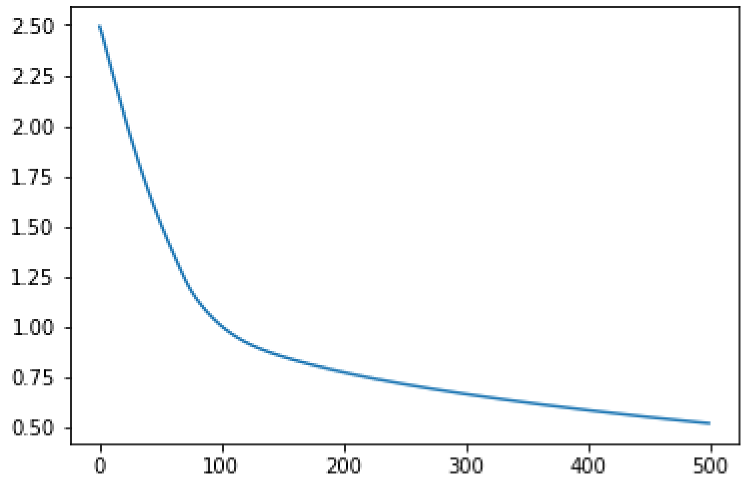
一开始loss非常大，而随着训练不断的进行，loss在逐渐减小。将history中的数据通过matplotlib绘图表现出来，就非常直观了。因为在jupyter上调试，我加入%matplotlib inline命令。

In [10]:

```
#将图片内嵌在交互窗口，而不是弹出一个图片窗口
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot(range(500),history.history['loss'])
```

Out[10]:

```
[<matplotlib.lines.Line2D at 0x7f6f883e48>]
```



我们如何评价一个模型训练是否成功？首先，训练过程中训练集loss要下降到一个较小的值，表示模型收敛较好，没有欠拟合；其次，测试集loss最后与训练集loss要尽可能相似，差距越小越好小，说明该模型没有过拟合。

模型成功训练出来后，便可以使用该模型对输入的鸢尾花数据，判断是属于哪一种类别了。这里选择一个最简单的部分，随机读取数据集中的几条数据，看看这个模型会输出什么结果。

我们可以在数据集中随机选择几条，略作修改后进行测试。看看输出的结果对不对。

In [11]:

```
data=pd.read_csv('./data/iris.csv')
data=data.sample(frac=1).reset_index(drop=True)    #打乱数据的先后顺序
x_input=data.iloc[:,0:-1]
x_input=x_input[:5]
data[:5].head()
```

Out[11]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	4.6	3.1	1.5	0.2	Iris-setosa
1	5.5	2.6	4.4	1.2	Iris-versicolor
2	6.1	2.6	5.6	1.4	Iris-virginica
3	7.2	3.0	5.8	1.6	Iris-virginica
4	5.2	3.4	1.4	0.2	Iris-setosa

In [12]:

```
model.predict(x_input)
```

Out[12]:

```
array([[0.81645006, 0.08945613, 0.09409384],
       [0.05531117, 0.4335839 , 0.51110494],
       [0.01924712, 0.40849388, 0.57225895],
       [0.03210969, 0.40018922, 0.56770104],
       [0.9145784 , 0.04051452, 0.04490715]], dtype=float32)
```

这里的数据是按照“Species_Iris-setosa，Species_Iris-versicolor，Species_Iris-virginica”来排序的。模型输出的数据中，每一列都是估算，哪一列数据大，我们就选择哪一个分类结果。

如果发现不准确，我们继续训练一下模型，即再运行几次，等到loss的值没有显著变化的时候，再来测试模型。一般来说，只要loss值到达0.1左右，识别效果就很不错了。

5.利用自带数据集

参考资料：https://keras.io/zh/ (https://keras.io/zh/)

5.1.数据导入

直接使用sklearn自带的数据集，即导入sklearn.datasets。貌似要从网络下载数据的，第一次使用，要等一会儿。

In [13]:

```
import keras
from keras import layers
from sklearn.datasets import load_iris
i_data = load_iris()
print(i_data.feature_names)
print(i_data.target_names)
x = i_data.data
y = i_data.target
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal
width (cm)']
['setosa' 'versicolor' 'virginica']
```

5.2.数据预处理

Scikit-Learn已经帮我们把类别编码成了数字，不过是一维数组（None,）（样本的个数不固定，用None表示），而Keras多分类接受的类别输入是一个二维数组，是y的one-hot编码形式。one-hot编码，简单来讲，就是将原来由0开始的类别值转换成向量，比如3个类别0,1,2，那么类别向量长度为3，以原类别值作为位置索引，对应位置置为1，其它位置置为0，即类别0对应：[1, 0, 0]，类别1对应[0, 1, 0]，类别2对应[0, 0, 1]。全部转换后，y变为二维数组（None，3），可以打印前3行看看。

In [14]:

```
from keras.utils.np_utils import to_categorical
y = to_categorical(y, 3)
print(y.shape)
print(y[0:3, :])
```

```
(150, 3)
[[1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
```

5.3.定义模型（搭建神经网络）

In [15]:

```
from keras.models import Sequential
model = Sequential()
from keras.layers import Dense
model.add(Dense(units=8, input_dim=4, activation='relu'))
model.add(Dense(units=3, activation='softmax'))
```

5.4.编译模型

In [16]:

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy
```

5.5.训练模型

这段代码可以多运行几次，你会发现loss多值会越来越小。到了0.05后，变化就不大了。

In [21]:

```python
history=model.fit(x, y, epochs=200, batch_size=10)
```

```
Epoch 1/200
150/150 [==============================] - 0s 1ms/step - loss: 0.3501
- acc: 0.9600
Epoch 2/200
150/150 [==============================] - 0s 974us/step - loss: 0.349
7 - acc: 0.9667
Epoch 3/200
150/150 [==============================] - 0s 916us/step - loss: 0.346
7 - acc: 0.9600
Epoch 4/200
150/150 [==============================] - 0s 874us/step - loss: 0.346
0 - acc: 0.9667
Epoch 5/200
150/150 [==============================] - 0s 979us/step - loss: 0.345
1 - acc: 0.9467
Epoch 6/200
150/150 [==============================] - 0s 994us/step - loss: 0.342
7 - acc: 0.9533
Epoch 7/200
150/150 [==============================] - 0s 952us/step - loss: 0.340
5 - acc: 0.9600 0s - loss: 0.3139 - acc: 0.966
Epoch 8/200
150/150 [==============================] - 0s 931us/step - loss: 0.338
4 - acc: 0.9600
Epoch 9/200
150/150 [==============================] - 0s 980us/step - loss: 0.336
8 - acc: 0.9733
Epoch 10/200
150/150 [==============================] - 0s 907us/step - loss: 0.335
2 - acc: 0.9600 0s - loss: 0.3253 - acc: 0.92
Epoch 11/200
150/150 [==============================] - 0s 908us/step - loss: 0.333
5 - acc: 0.9600
Epoch 12/200
150/150 [==============================] - 0s 876us/step - loss: 0.332
0 - acc: 0.9667
Epoch 13/200
150/150 [==============================] - 0s 929us/step - loss: 0.330
0 - acc: 0.9667
Epoch 14/200
150/150 [==============================] - 0s 956us/step - loss: 0.328
4 - acc: 0.9667
Epoch 15/200
150/150 [==============================] - 0s 990us/step - loss: 0.328
0 - acc: 0.9667
Epoch 16/200
150/150 [==============================] - 0s 980us/step - loss: 0.325
5 - acc: 0.9600
Epoch 17/200
150/150 [==============================] - 0s 1ms/step - loss: 0.3236
- acc: 0.9600
Epoch 18/200
150/150 [==============================] - 0s 1ms/step - loss: 0.3220
- acc: 0.9733
Epoch 19/200
150/150 [==============================] - 0s 807us/step - loss: 0.320
```

```
1 - acc: 0.9733
Epoch 20/200
150/150 [==============================] - 0s 901us/step - loss: 0.318
8 - acc: 0.9667
Epoch 21/200
150/150 [==============================] - 0s 868us/step - loss: 0.317
2 - acc: 0.9733 0s - loss: 0.3192 - acc: 0.98
Epoch 22/200
150/150 [==============================] - 0s 870us/step - loss: 0.317
5 - acc: 0.9800
Epoch 23/200
150/150 [==============================] - 0s 884us/step - loss: 0.314
6 - acc: 0.9667
Epoch 24/200
150/150 [==============================] - 0s 842us/step - loss: 0.313
0 - acc: 0.9600
Epoch 25/200
150/150 [==============================] - 0s 915us/step - loss: 0.310
4 - acc: 0.9600
Epoch 26/200
150/150 [==============================] - 0s 1ms/step - loss: 0.3084
- acc: 0.9733
Epoch 27/200
150/150 [==============================] - 0s 946us/step - loss: 0.307
4 - acc: 0.9733
Epoch 28/200
150/150 [==============================] - 0s 993us/step - loss: 0.305
2 - acc: 0.9733
Epoch 29/200
150/150 [==============================] - 0s 883us/step - loss: 0.303
8 - acc: 0.9733
Epoch 30/200
150/150 [==============================] - 0s 900us/step - loss: 0.302
3 - acc: 0.9733 0s - loss: 0.3035 - acc: 0.971
Epoch 31/200
150/150 [==============================] - 0s 840us/step - loss: 0.301
0 - acc: 0.9733
Epoch 32/200
150/150 [==============================] - 0s 923us/step - loss: 0.298
5 - acc: 0.9733
Epoch 33/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2987
- acc: 0.9733
Epoch 34/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2967
- acc: 0.9733
Epoch 35/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2942
- acc: 0.9733
Epoch 36/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2923
- acc: 0.9733A: 0s - loss: 0.2950 - acc: 0.95
Epoch 37/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2913
- acc: 0.9733
Epoch 38/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2922
- acc: 0.9600
Epoch 39/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2887
- acc: 0.9800
```

```
Epoch 40/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2863
- acc: 0.9800A: 0s - loss: 0.2743 - acc: 0.990
Epoch 41/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2848
- acc: 0.9800
Epoch 42/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2832
- acc: 0.9733
Epoch 43/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2813
- acc: 0.9733
Epoch 44/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2802
- acc: 0.9733A: 0s - loss: 0.2871 - acc: 1.00
Epoch 45/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2780
- acc: 0.9733
Epoch 46/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2766
- acc: 0.9800
Epoch 47/200
150/150 [==============================] - 0s 909us/step - loss: 0.275
2 - acc: 0.9800
Epoch 48/200
150/150 [==============================] - 0s 953us/step - loss: 0.273
6 - acc: 0.9800
Epoch 49/200
150/150 [==============================] - 0s 944us/step - loss: 0.272
0 - acc: 0.9800
Epoch 50/200
150/150 [==============================] - 0s 966us/step - loss: 0.270
6 - acc: 0.9733
Epoch 51/200
150/150 [==============================] - 0s 875us/step - loss: 0.269
8 - acc: 0.9733
Epoch 52/200
150/150 [==============================] - 0s 863us/step - loss: 0.268
8 - acc: 0.9800
Epoch 53/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2657
- acc: 0.9800
Epoch 54/200
150/150 [==============================] - 0s 843us/step - loss: 0.264
7 - acc: 0.9733
Epoch 55/200
150/150 [==============================] - 0s 892us/step - loss: 0.263
3 - acc: 0.9800
Epoch 56/200
150/150 [==============================] - 0s 894us/step - loss: 0.261
2 - acc: 0.9800
Epoch 57/200
150/150 [==============================] - 0s 928us/step - loss: 0.260
3 - acc: 0.9800
Epoch 58/200
150/150 [==============================] - 0s 922us/step - loss: 0.258
6 - acc: 0.9800
Epoch 59/200
150/150 [==============================] - 0s 918us/step - loss: 0.257
3 - acc: 0.9800
Epoch 60/200
```

```
150/150 [==============================] - 0s 842us/step - loss: 0.256
1 - acc: 0.9800
Epoch 61/200
150/150 [==============================] - 0s 852us/step - loss: 0.254
6 - acc: 0.9800 0s - loss: 0.2505 - acc: 0.95
Epoch 62/200
150/150 [==============================] - 0s 933us/step - loss: 0.253
7 - acc: 0.9733
Epoch 63/200
150/150 [==============================] - 0s 899us/step - loss: 0.252
1 - acc: 0.9800
Epoch 64/200
150/150 [==============================] - 0s 899us/step - loss: 0.249
8 - acc: 0.9800
Epoch 65/200
150/150 [==============================] - ETA: 0s - loss: 0.2491 - ac
c: 0.962 - 0s 785us/step - loss: 0.2490 - acc: 0.9733
Epoch 66/200
150/150 [==============================] - 0s 923us/step - loss: 0.247
0 - acc: 0.9800
Epoch 67/200
150/150 [==============================] - 0s 960us/step - loss: 0.245
6 - acc: 0.9800
Epoch 68/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2449
- acc: 0.9800
Epoch 69/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2426
- acc: 0.9800A: 0s - loss: 0.2431 - acc: 0.981
Epoch 70/200
150/150 [==============================] - 0s 997us/step - loss: 0.241
5 - acc: 0.9800
Epoch 71/200
150/150 [==============================] - ETA: 0s - loss: 0.2508 - ac
c: 0.981 - 0s 1ms/step - loss: 0.2405 - acc: 0.9800
Epoch 72/200
150/150 [==============================] - 0s 975us/step - loss: 0.238
8 - acc: 0.9800
Epoch 73/200
150/150 [==============================] - 0s 953us/step - loss: 0.237
9 - acc: 0.9733
Epoch 74/200
150/150 [==============================] - 0s 959us/step - loss: 0.236
3 - acc: 0.9800
Epoch 75/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2345
- acc: 0.9800
Epoch 76/200
150/150 [==============================] - 0s 988us/step - loss: 0.234
1 - acc: 0.9800
Epoch 77/200
150/150 [==============================] - 0s 799us/step - loss: 0.232
3 - acc: 0.9800
Epoch 78/200
150/150 [==============================] - 0s 870us/step - loss: 0.231
1 - acc: 0.9800
Epoch 79/200
150/150 [==============================] - 0s 964us/step - loss: 0.230
2 - acc: 0.9800
Epoch 80/200
150/150 [==============================] - 0s 895us/step - loss: 0.228
```

```
4 - acc: 0.9800
Epoch 81/200
150/150 [==============================] - 0s 934us/step - loss: 0.227
6 - acc: 0.9800
Epoch 82/200
150/150 [==============================] - 0s 930us/step - loss: 0.225
3 - acc: 0.9800
Epoch 83/200
150/150 [==============================] - 0s 932us/step - loss: 0.224
2 - acc: 0.9800
Epoch 84/200
150/150 [==============================] - 0s 962us/step - loss: 0.224
2 - acc: 0.9800 0s - loss: 0.2103 - acc: 0.98
Epoch 85/200
150/150 [==============================] - 0s 987us/step - loss: 0.221
6 - acc: 0.9800
Epoch 86/200
150/150 [==============================] - 0s 992us/step - loss: 0.220
9 - acc: 0.9800
Epoch 87/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2195
- acc: 0.9800
Epoch 88/200
150/150 [==============================] - 0s 954us/step - loss: 0.218
6 - acc: 0.9800
Epoch 89/200
150/150 [==============================] - 0s 880us/step - loss: 0.217
2 - acc: 0.9800
Epoch 90/200
150/150 [==============================] - 0s 846us/step - loss: 0.215
9 - acc: 0.9800
Epoch 91/200
150/150 [==============================] - 0s 864us/step - loss: 0.214
5 - acc: 0.9800
Epoch 92/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2149
- acc: 0.9800
Epoch 93/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2119
- acc: 0.9800
Epoch 94/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2109
- acc: 0.9800
Epoch 95/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2109
- acc: 0.9800
Epoch 96/200
150/150 [==============================] - ETA: 0s - loss: 0.2122 - ac
c: 0.981 - 0s 1ms/step - loss: 0.2086 - acc: 0.9800
Epoch 97/200
150/150 [==============================] - 0s 975us/step - loss: 0.207
9 - acc: 0.9800
Epoch 98/200
150/150 [==============================] - 0s 1ms/step - loss: 0.2074
- acc: 0.9800
Epoch 99/200
150/150 [==============================] - 0s 984us/step - loss: 0.205
2 - acc: 0.9800
Epoch 100/200
150/150 [==============================] - 0s 992us/step - loss: 0.206
1 - acc: 0.9867 0s - loss: 0.1989 - acc: 0.991
```

```
Epoch 101/200
150/150 [==============================] - 0s 996us/step - loss: 0.203
2 - acc: 0.9800
Epoch 102/200
150/150 [==============================] - 0s 988us/step - loss: 0.202
7 - acc: 0.9800
Epoch 103/200
150/150 [==============================] - 0s 915us/step - loss: 0.200
7 - acc: 0.9800
Epoch 104/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1999
- acc: 0.9800
Epoch 105/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1988
- acc: 0.9800
Epoch 106/200
150/150 [==============================] - 0s 925us/step - loss: 0.197
7 - acc: 0.9800
Epoch 107/200
150/150 [==============================] - 0s 985us/step - loss: 0.196
7 - acc: 0.9800
Epoch 108/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1955
- acc: 0.9800A: 0s - loss: 0.1988 - acc: 0.975
Epoch 109/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1954
- acc: 0.9800
Epoch 110/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1946
- acc: 0.9800
Epoch 111/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1924
- acc: 0.9800
Epoch 112/200
150/150 [==============================] - 0s 989us/step - loss: 0.191
4 - acc: 0.9800
Epoch 113/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1905
- acc: 0.9800
Epoch 114/200
150/150 [==============================] - 0s 992us/step - loss: 0.189
9 - acc: 0.9800
Epoch 115/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1896
- acc: 0.9800
Epoch 116/200
150/150 [==============================] - 0s 985us/step - loss: 0.187
5 - acc: 0.9800
Epoch 117/200
150/150 [==============================] - 0s 970us/step - loss: 0.186
8 - acc: 0.9800
Epoch 118/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1854
- acc: 0.9800
Epoch 119/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1856
- acc: 0.9800
Epoch 120/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1842
- acc: 0.9800
Epoch 121/200
```

```
150/150 [==============================] - 0s 1ms/step - loss: 0.1837
- acc: 0.9800
Epoch 122/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1847
- acc: 0.9800
Epoch 123/200
150/150 [==============================] - 0s 975us/step - loss: 0.181
0 - acc: 0.9800
Epoch 124/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1821
- acc: 0.9733
Epoch 125/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1840
- acc: 0.9800
Epoch 126/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1794
- acc: 0.9800
Epoch 127/200
150/150 [==============================] - 0s 946us/step - loss: 0.177
5 - acc: 0.9867
Epoch 128/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1768
- acc: 0.9800
Epoch 129/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1761
- acc: 0.9800
Epoch 130/200
150/150 [==============================] - 0s 926us/step - loss: 0.174
2 - acc: 0.9800
Epoch 131/200
150/150 [==============================] - 0s 977us/step - loss: 0.174
3 - acc: 0.9800
Epoch 132/200
150/150 [==============================] - 0s 948us/step - loss: 0.172
7 - acc: 0.9800
Epoch 133/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1720
- acc: 0.9800
Epoch 134/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1718
- acc: 0.9800
Epoch 135/200
150/150 [==============================] - 0s 961us/step - loss: 0.171
8 - acc: 0.9800
Epoch 136/200
150/150 [==============================] - 0s 928us/step - loss: 0.169
9 - acc: 0.9800
Epoch 137/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1707
- acc: 0.9800
Epoch 138/200
150/150 [==============================] - ETA: 0s - loss: 0.1820 - ac
c: 0.972 - 0s 1ms/step - loss: 0.1678 - acc: 0.9800
Epoch 139/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1685
- acc: 0.9800
Epoch 140/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1668
- acc: 0.9800
Epoch 141/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1666
```

```
- acc: 0.9800
Epoch 142/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1649
- acc: 0.9800
Epoch 143/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1637
- acc: 0.9800
Epoch 144/200
150/150 [==============================] - 0s 951us/step - loss: 0.163
1 - acc: 0.9800
Epoch 145/200
150/150 [==============================] - 0s 970us/step - loss: 0.162
7 - acc: 0.9800
Epoch 146/200
150/150 [==============================] - 0s 986us/step - loss: 0.162
3 - acc: 0.9800
Epoch 147/200
150/150 [==============================] - 0s 769us/step - loss: 0.161
0 - acc: 0.9800
Epoch 148/200
150/150 [==============================] - 0s 872us/step - loss: 0.160
5 - acc: 0.9800
Epoch 149/200
150/150 [==============================] - 0s 975us/step - loss: 0.160
1 - acc: 0.9800 0s - loss: 0.1735 - acc: 1.00
Epoch 150/200
150/150 [==============================] - 0s 813us/step - loss: 0.159
3 - acc: 0.9800
Epoch 151/200
150/150 [==============================] - 0s 878us/step - loss: 0.158
3 - acc: 0.9800 0s - loss: 0.1651 - acc: 0.97
Epoch 152/200
150/150 [==============================] - 0s 928us/step - loss: 0.157
2 - acc: 0.9800
Epoch 153/200
150/150 [==============================] - 0s 982us/step - loss: 0.157
4 - acc: 0.9800
Epoch 154/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1562
- acc: 0.9800
Epoch 155/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1554
- acc: 0.9800
Epoch 156/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1547
- acc: 0.9800
Epoch 157/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1538
- acc: 0.9800
Epoch 158/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1536
- acc: 0.9800
Epoch 159/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1525
- acc: 0.9800
Epoch 160/200

150/150 [==============================] - 0s 1ms/step - loss: 0.1519
- acc: 0.9800
Epoch 161/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1512
```

```
- acc: 0.9800
Epoch 162/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1511
- acc: 0.9867
Epoch 163/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1503
- acc: 0.9800
Epoch 164/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1500
- acc: 0.9800
Epoch 165/200
150/150 [==============================] - ETA: 0s - loss: 0.1366 - a
cc: 0.990 - 0s 1ms/step - loss: 0.1484 - acc: 0.9800
Epoch 166/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1487
- acc: 0.9800
Epoch 167/200
150/150 [==============================] - ETA: 0s - loss: 0.1618 - a
cc: 0.975 - 0s 987us/step - loss: 0.1479 - acc: 0.9800
Epoch 168/200
150/150 [==============================] - ETA: 0s - loss: 0.1602 - a
cc: 0.972 - 0s 1ms/step - loss: 0.1471 - acc: 0.9800
Epoch 169/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1464
- acc: 0.9800
Epoch 170/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1456
- acc: 0.9867
Epoch 171/200
150/150 [==============================] - 0s 992us/step - loss: 0.14
49 - acc: 0.9800
Epoch 172/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1442
- acc: 0.9800
Epoch 173/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1442
- acc: 0.9800
Epoch 174/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1448
- acc: 0.9800
Epoch 175/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1435
- acc: 0.9800
Epoch 176/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1422
- acc: 0.9800
Epoch 177/200
150/150 [==============================] - 0s 949us/step - loss: 0.14
17 - acc: 0.9800
Epoch 178/200
150/150 [==============================] - 0s 977us/step - loss: 0.14
07 - acc: 0.9800 0s - loss: 0.1434 - acc: 0.976
Epoch 179/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1408
- acc: 0.9800
Epoch 180/200
150/150 [==============================] - ETA: 0s - loss: 0.1402 - a
cc: 0.972 - 0s 1ms/step - loss: 0.1403 - acc: 0.9800
Epoch 181/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1395
- acc: 0.9800
```

```
Epoch 182/200
150/150 [==============================] - 0s 944us/step - loss: 0.13
90 - acc: 0.9800
Epoch 183/200
150/150 [==============================] - 0s 983us/step - loss: 0.13
96 - acc: 0.9800
Epoch 184/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1373
- acc: 0.9800
Epoch 185/200
150/150 [==============================] - 0s 984us/step - loss: 0.13
76 - acc: 0.9800
Epoch 186/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1369
- acc: 0.9800A: 0s - loss: 0.1302 - acc: 0.983
Epoch 187/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1361
- acc: 0.9800A: 0s - loss: 0.1307 - acc: 0.983 - ETA: 0s - loss: 0.14
22 - acc: 0.972
Epoch 188/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1358
- acc: 0.9800
Epoch 189/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1356
- acc: 0.9800
Epoch 190/200
150/150 [==============================] - 0s 985us/step - loss: 0.13
43 - acc: 0.9800
Epoch 191/200
150/150 [==============================] - ETA: 0s - loss: 0.1192 - a
cc: 0.981 - 0s 1ms/step - loss: 0.1337 - acc: 0.9800
Epoch 192/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1337
- acc: 0.9800
Epoch 193/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1334
- acc: 0.9800
Epoch 194/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1331
- acc: 0.9800
Epoch 195/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1319
- acc: 0.9800
Epoch 196/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1319
- acc: 0.9800
Epoch 197/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1307
- acc: 0.9800
Epoch 198/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1307
- acc: 0.9867
Epoch 199/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1318
- acc: 0.9800
Epoch 200/200
150/150 [==============================] - 0s 1ms/step - loss: 0.1293
- acc: 0.9800
```

History对象会被模型的fit方法返回。用dir(history)的方式，可以得到History对象的所有属性，用vars(history)能够看到所有的属性值。

具体可以参考：https://keras.io/zh/callbacks/#history (https://keras.io/zh/callbacks/#history)

In [20]:

```
history.history['loss']
```

Out[20]:

```
[2.5524308999379475,
 2.217859665552775,
 1.9382617394129436,
 1.7239010175069174,
 1.581995193163553 9,
 1.486048420270284,
 1.4173336664835612,
 1.3688759565353394,
 1.323557710647583,
 1.285457436243693,
 1.2532981475194296,
 1.2227479537328085,
 1.1967514117558797,
 1.1723539352416992,
 1.1520002047220865,
 1.1328341484069824,
 1.116115434964498,
 1.101676638921102,
```

5.6.评估模型

In [22]:

```
result = model.evaluate(x, y)
print(result[1])
```

```
150/150 [==============================] - 0s 1ms/step
0.98
```

5.7.模型预测

predict输出概率矩阵，每一行对应预测值在三个类别上的概率；predict_classes输出类别值，可以全部打印出来看一下。

注意：这里采用的是科学计数法。哪个数字小，就说明哪个概率最大。

In [23]:

```
proba = model.predict(x)
print(proba[0:150])
```

```
[[9.94676471e-01 5.32344682e-03 1.37119770e-07]
 [9.92624938e-01 7.37464847e-03 3.18816092e-07]
 [9.92137969e-01 7.86165334e-03 3.76266797e-07]
 [9.91234660e-01 8.76479223e-03 4.98782072e-07]
 [9.94500399e-01 5.49943792e-03 1.49158325e-07]
 [9.95819032e-01 4.18085558e-03 7.34049905e-08]
 [9.91707981e-01 8.29158351e-03 4.31940975e-07]
 [9.94068742e-01 5.93105145e-03 1.81369444e-07]
 [9.89160359e-01 1.08388551e-02 8.65405184e-07]
 [9.93282795e-01 6.71697874e-03 2.50305760e-07]
 [9.95990098e-01 4.00977861e-03 6.58896084e-08]
 [9.93174016e-01 6.82569528e-03 2.60931955e-07]
 [9.92531478e-01 7.46812671e-03 3.29390701e-07]
 [9.89453435e-01 1.05456924e-02 8.05948616e-07]
 [9.97298658e-01 2.70137028e-03 2.37442883e-08]
 [9.97193933e-01 2.80600181e-03 2.61933977e-08]
 [9.95835066e-01 4.16482333e-03 7.26794909e-08]
 [9.94362950e-01 5.63687505e-03 1.58996414e-07]
 [9.96677518e-01 3.32251959e-03 4.05301037e-08]
```

In [24]:

```
classes = model.predict_classes(x)
print(classes[0:150])
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
2 1
 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2
2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2
 2 2]
```

6.模型的保存和导入

6.1.模型保存

使用save可以保存训练好的模型，下次导入即可使用。

In [25]:

```
model.save('./model/2-model-vv.h5')    # HDF5文件
```

6.2.模型导入

使用keras.models的load_model语句载入模型，就可以直接用这个模型来做预测了。

In [26]:

```
from keras.models import load_model
import pandas as pd
model = load_model('./model/2-model-vv.h5')
```

In [60]:

```
#请输入数据，如：5.5,2.6,4.4,1.2
s=input("请输入数据，用“,”分开:")
```

```
请输入数据，用“,”分开:5.5,2.6,4.4,1.2
```

In [61]:

```
i_data=s.split(',')
i_data
```

Out[61]:

```
['5.5', '2.6', '4.4', '1.2']
```

In [62]:

```
#生成一个空的DataFrame，将输入的列表添加为新行
x=pd.DataFrame(columns=['Sepal.Length','Sepal.Width','Petal.Length','Petal.Width'])
x.loc[0]=i_data
```

In [63]:

```
#对应的结果，应该是1，即“Species_Iris-versicolor”
model.predict_classes(x)
```

Out[63]:

```
array([1])
```

In [ ]:

```

```