神经网络&深度学习 TENSORFLOW Google

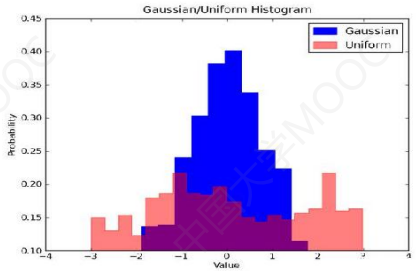
Matplotlib数据可视化

西安科技大学 牟琦
muqi@xust.edu.cn

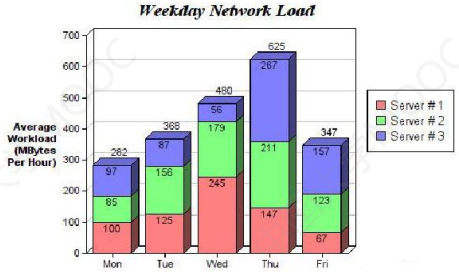
中国大学MOOC

6.1 Matplotlib绘图基础

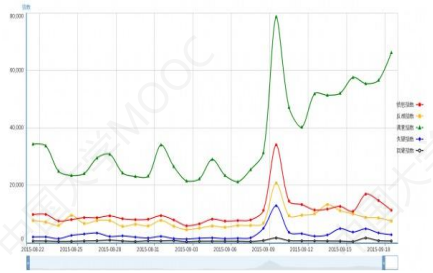
- **数据可视化**
  - **数据分析**阶段：理解和洞察数据之间的关系
  - **算法调试**阶段：发现问题，优化算法
  - **项目总结**阶段：展示项目成果
- **Matplotlib**：第三方库，可以快速方便地生成高质量的图表



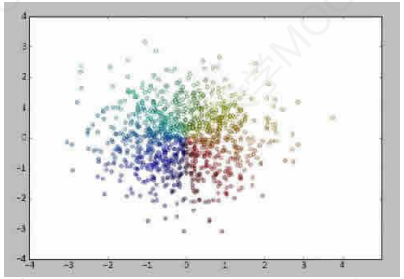
直方图



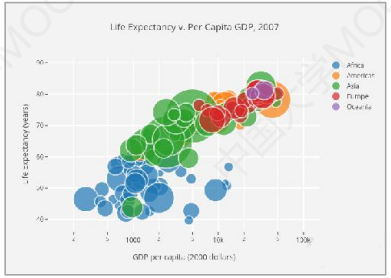
柱形图



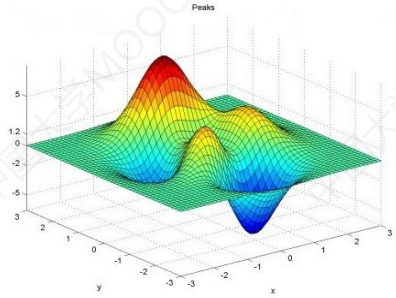
折线图



散点图



气泡图



三维图

安装Matplotlib库

- Anaconda：安装了anaconda之后，Matplotlib就已经被安装好了
- pip安装

```
pip install matplotlib
```

导入Matplotlib库

```
import matplotlib.pyplot as plt
```

Figure 对象：创建画布

figure(num,figsize,dpi,facecolor,edgecolor,frameon)

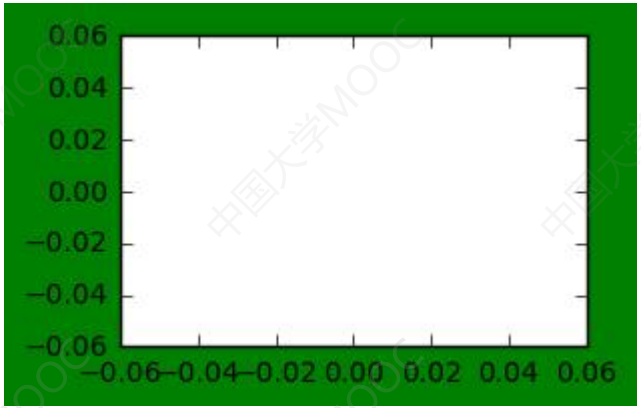
- num：图形**编号**或**名称**，取值为数字/字符串。
- figsize：绘图对象的**宽**和**高**，单位为英寸。
- dpi：绘图对象的**分辨率**，缺省值为80。
- facecolor：**背景颜色**。
- edgecolor：**边框颜色**。
- frameon：表示**是否显示边框**。

```
1  import matplotlib.pyplot as plt
2
3  plt.figure(figsize=(3,2),facecolor="green")
4  plt.plot()
5  plt.show()
```

创建画布

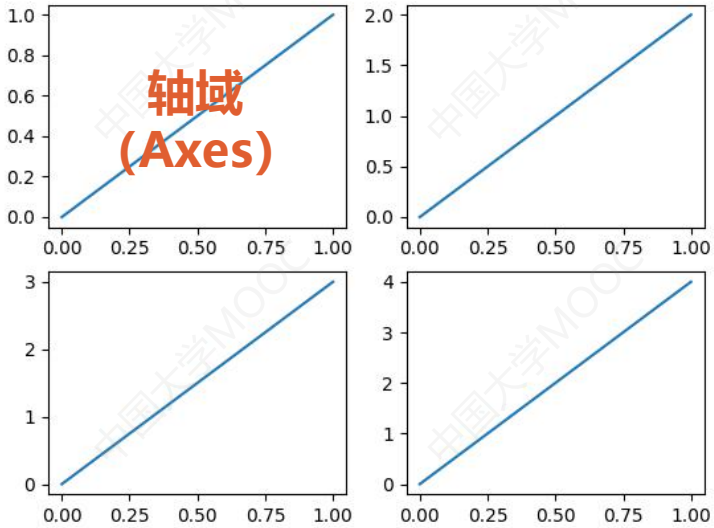
绘制空白图形

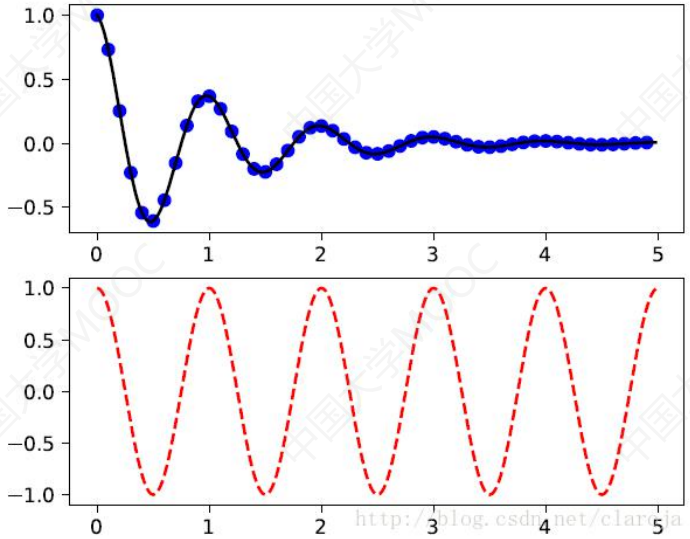
显示绘图



颜色	缩略字符	颜色	缩略字符
blue	b	black	k
green	g	white	w
red	r	cyan	c
yellow	y	magenta	m

划分子图





subplot()函数——划分子图

subplot(行数, 列数, 子图序号)

1	
2	

1	2
3	4

1	2	3
4	5	6

```
1  plt.subplot(2,2,1)
2  plt.subplot(2,2,2)
3  plt.subplot(2,2,3)
4  plt.subplot(2,2,4)
```

每个subplot()函数只创建一个子图。
要创建4个子图，就需要4条语句

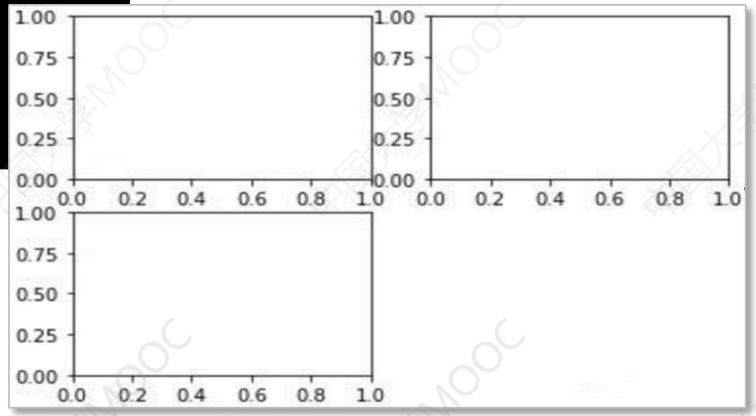
当subplot()函数中的3个参数都**小于10时**，可以省略参数间的逗号，用一个3位数来表示

```
1  plt.subplot(221)
2  plt.subplot(222)
3  plt.subplot(223)
4  plt.subplot(224)
```

例：将画布划分为2×2的子图区域，并绘制3个子图

```
1  import matplotlib.pyplot as plt
2
3  fig = plt.figure()
4
5  plt.subplot(221)
6  plt.subplot(222)
7  plt.subplot(223)
8
9  plt.show()
```

运行结果：



设置中文字体

```
plt.rcParams [" font.sans-serif" ] = "SimHei"
```

run configuration Params　字体　中文黑体

运行配置参数：指定所绘制图表中的各种**默认属性**，是matplotlib中的**全局变量**

中文字体	英文描述	中文字体	英文描述
宋体	SimSun	楷体	KaiTi
黑体	SimHei	仿宋	FangSong
微软雅黑	Microsoft YaHei	隶书	LiSu
微软正黑体	Microsoft JhengHei	幼圆	YouYuan

- 恢复标准默认配置

```
plt.rcdefaults( )
```

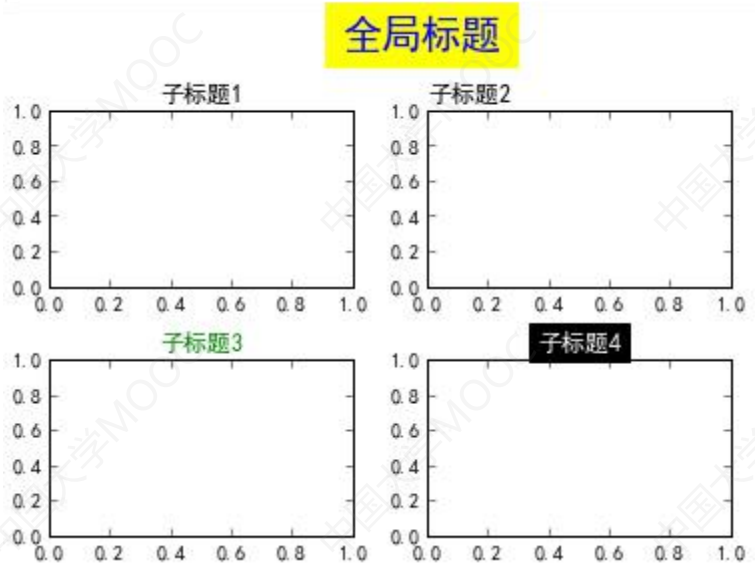
添加标题

添加全局标题

suptitle (标题文字)

添加子标题

title (标题文字)



suptitle()函数的主要参数

参 数	说 明	默认值
x	标题位置的x坐标	0.5
y	标题位置的y坐标	0.98
color	标题颜色	黑色
backgroundcolor	标题背景颜色	12
fontsize	标题的字体大小	
fontweight	字体粗细	normal
fontstyle	设置字体类型	
horizontalalignment	标题水平对齐方式	center
verticalalignment	标题的垂直对齐方式	top

fontsize:
xx-small
x-small
small
medium
large
x-large
xx-large

fontweight:
light
normal
medium
semibold
bold
heavy
black

normal / italic / oblique

left / right / center

center / top / bottom / baseline

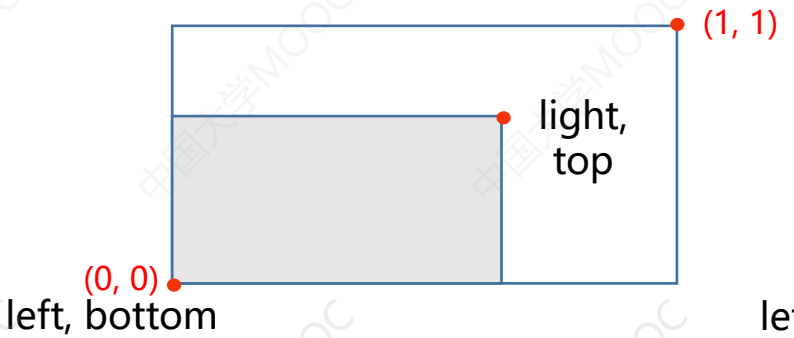
title()函数的主要参数

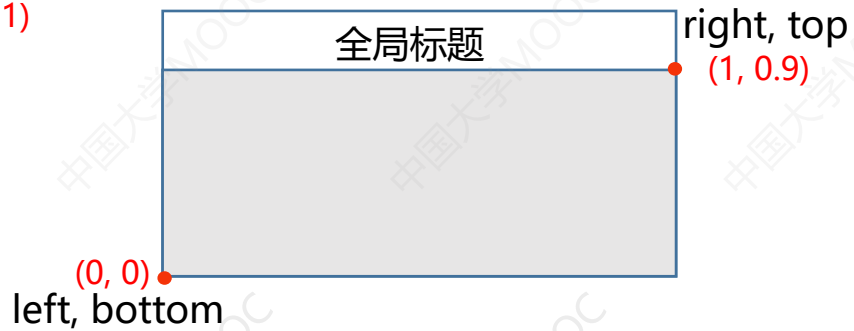
参 数	说 明	取 值
loc	标题位置	left，right
rotation	标题文字旋转角度	
color	标题颜色	黑色
fontsize	标题的字体大小	
fontweight	字体粗细	normal
fontstyle	设置字体类型	
horizontalalignment	标题水平对齐方式	center
verticalalignment	标题的垂直对齐方式	top
fontdict	设置参数字典	

tight_layout()函数

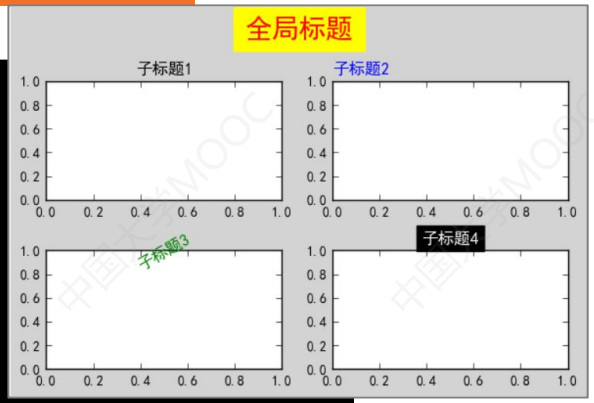
检查坐标轴标签、刻度标签、和子图标题，**自动调整子图**，使之**填充整个绘图区域**，并**消除子图之间的重叠**。

```
tight_layout( rect=[left, bottom, right, top])
```





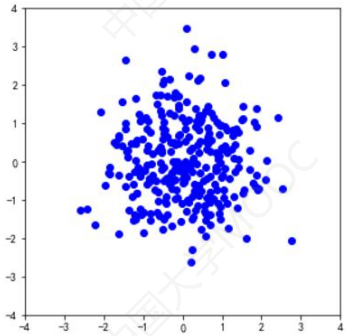
```python
import matplotlib.pyplot as plt

plt.rcParams["font.family"] = "SimHei"

fig = plt.figure(facecolor="lightgrey")

plt.subplot(2,2,1)
plt.title('子标题1')
plt.subplot(2,2,2)
plt.title('子标题2',loc="left",color="b")
plt.subplot(2,2,3)
myfontdict = {"fontsize":12, "color":"g", "rotation":30}
plt.title('子标题3',fontdict=myfontdict)
plt.subplot(2,2,4)
plt.title('子标题4',color='white', backgroundcolor="black")

plt.suptitle("全局标题", fontsize=20, color="red",backgroundcolor="yellow")

plt.tight_layout(rect=[0,0,1,0.9])

plt.show()
```

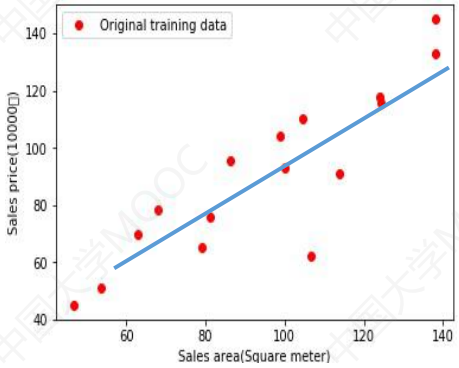


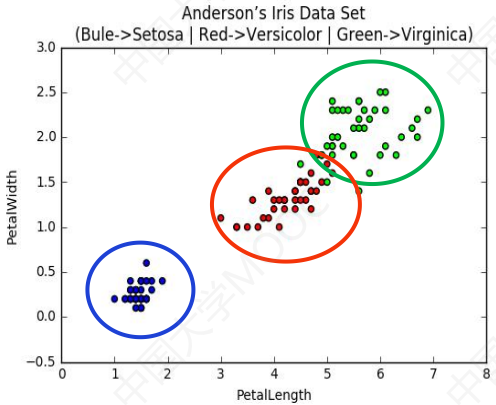

tensorflow / tensorflow
TensorFlow: Open sou
Clone this repo:
git clone https://tensor
Branches
master

6.2 散点图

- **散点图**（Scatter）：是数据点在直角坐标系中的分布图
  - 数据分布规律
  - 数据变化趋势
  - 数据分组







scatter() 函数

scatter(x, y, scale, color, marker, label)

参 数	说 明	默认值
x	数据点的x坐标	不可省略
y	数据点的y坐标	不可省略
scale	数据点的大小	36
color	数据点的颜色	
marker	数据点的样式	’o’（圆点）
label	图例文字	

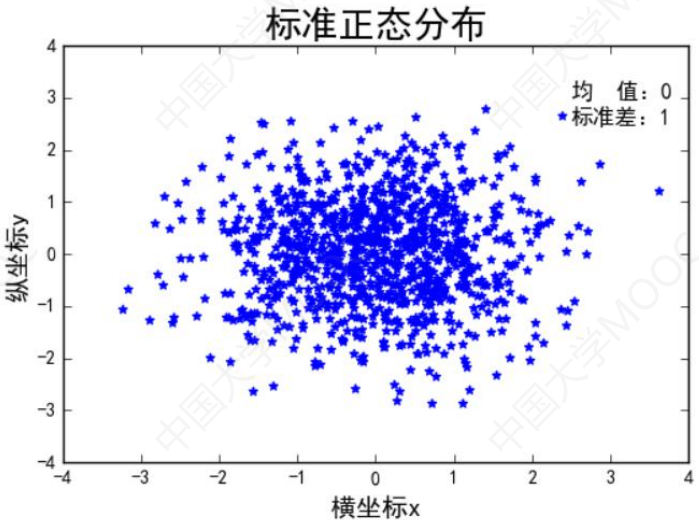
color参数——常用颜色

颜 色	缩略字符	颜 色	缩略字符
blue	b	black	k
green	g	white	w
red	r	cyan	c
yellow	y	magenta	m

marker参数——数据点样式

取值	中文描述	取值	中文描述	取值	中文描述
-	实线	1	朝下的三角	v	朝下的三角形
--	虚线	2	朝上的三角	^	朝上的三角形
-.	点线	3	朝左的三角	<	朝左的三角形
:	点虚线	4	朝右的三角	>	朝右的三角形
.	点	s	正方形	D	钻石形
,	像素	p	五角形	d	小版钻石形
o	圆形	*	星型	|	垂直线形
+	+号标记	h	1号六角形	_	水平线行
x	x号标记	H	2号六角形		

例：绘制**标准正态分布**的散点图



- 设置默认字体为中文黑体

```
plt.rcParams['font.sans-serif']="SimHei"
```

- 标准正态分布的散点坐标

```
n=1024
x = np.random.normal(0,1,n)
y = np.random.normal(0,1,n)
```

- 绘制散点图

```
plt.scatter(x, y, color="blue",marker='*' )
```

- 设置标题

```
plt.title("标准正态分布",fontsize=20)
```

添加文字——text() 函数

text(x, y, s, fontsize,color)

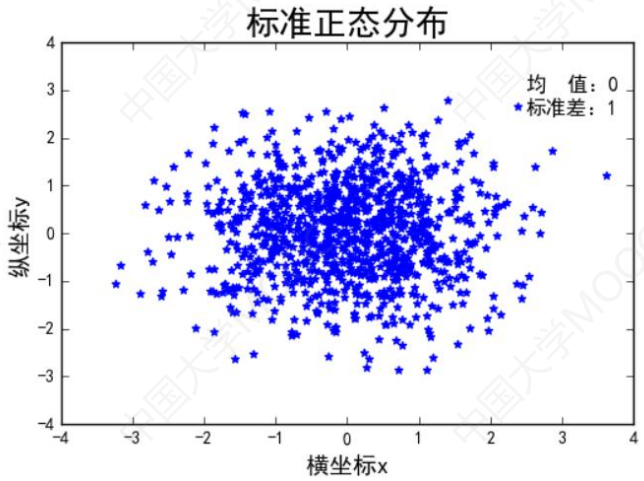
参 数	说 明	默认值
x	文字的x坐标	不可省略
y	文字的y坐标	不可省略
s	显示的文字	不可省略
fontsize	文字的大小	12
color	文字的颜色	黑色

坐标轴设置

plt.rcParams["axes.unicode_minus"] = False

函数	说 明
xlabel(x, y, s, fontsize,color)	设置x轴标签
ylabel(x, y, s, fontsize,color)	设置y轴标签
xlim(xmin, xmax)	设置x轴坐标的范围
ylim(ymin, ymax)	设置y轴坐标的范围
tick_params(labelsize)	设置刻度文字的字号

例：绘制标准正态分布的散点图



- 设置文本

```
plt.text(2.5,2.5,"均 值：0\n标准差：1")
```

- 设置坐标轴范围

```
plt.xlim(-4,4)
plt.ylim(-4,4)
```

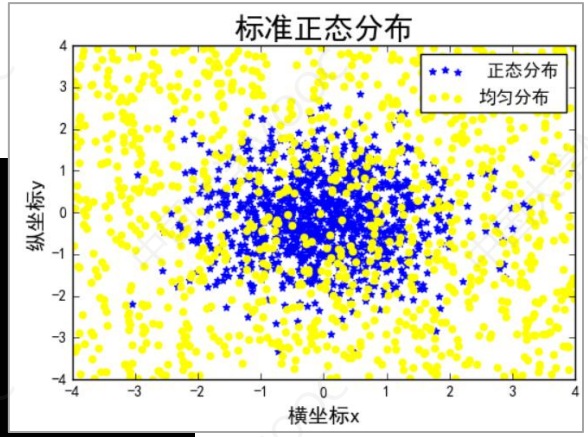
- 设置坐标轴标签

```
plt.xlabel('横坐标x', fontsize=14)
plt.ylabel('纵坐标y', fontsize=14)
```

```python
import matplotlib.pyplot as plt                 # 导入绘图库
import numpy as np                              # 导入numpy库

plt.rcParams['font.sans-serif']="SimHei"        # 设置中文黑体为默认字体
plt.rcParams['axes.unicode_minus']=False        # 正常显示负号

n =1024                                         # 随机点个数：1024
x = np.random.normal(0,1,n)                     # 生成数据点x坐标
y = np.random.normal(0,1,n)                     # 生成数据点y坐标

plt.scatter(x, y, color="blue",marker='*' )           # 绘制数据点

plt.title("标准正态分布",fontsize=20)                  # 设置标题
plt.text(2.5,2.5,"均　值：0\n标准差：1")               # 显示文本

plt.xlim(-4,4)                                  # x轴范围
plt.ylim(-4,4)                                  # y轴范围

plt.xlabel('横坐标x', fontsize=14)                     # 设置x轴标签文本
plt.ylabel('纵坐标y', fontsize=14)                     # 设置y轴标签文本

plt.show()                                      # 显示绘图
```

例：绘制**标准正态分布**、**均匀分布**的散点图

```
1  n =1024
2  x1 = np.random.normal(0,1,n)
3  y1 = np.random.normal(0,1,n)
4
5  x2=np.random.uniform(-4,4,(1,n))
6  y2=np.random.uniform(-4,4,(1,n))
7
8  plt.scatter(x1, y1, color="blue",marker='*' )
9  plt.scatter(x2, y2, color="yellow",marker='o' )
```



增加图例

指定图例内容

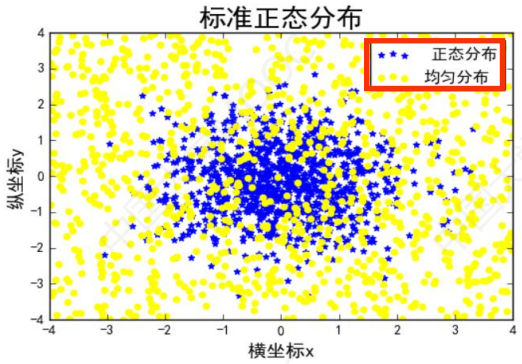
```
scatter( x, y, scale, color, marker, label )
```

```
legend(loc, fontsize)
```

显示图例

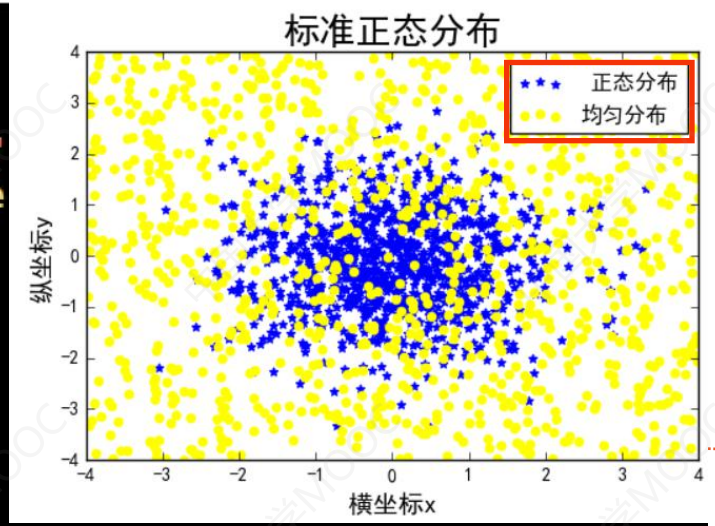
loc参数——指定图例的位置

取值	图例位置	取值	图例位置
0	best	6	center left
1	upper right	7	center right
2	upper left	8	lower center
3	lower left	9	upper center
4	lower right	10	center
5	right		



6.2 散点图

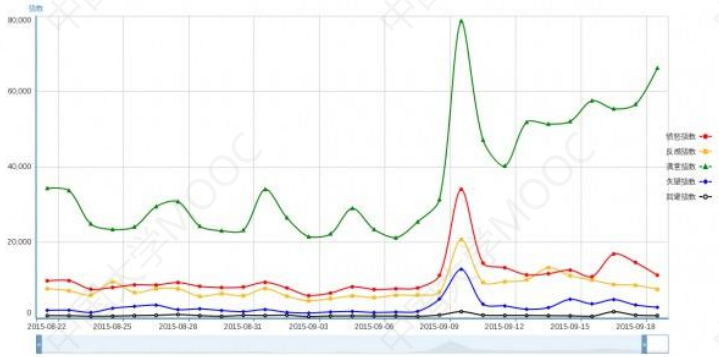

```
import matplotlib.pyplot as plt
import numpy as np

plt.rcParams['font.sans-serif']="SimHei"
plt.rcParams['axes.unicode_minus']=False

n =1024
x1 = np.random.normal(0,1,n)
y1 = np.random.normal(0,1,n)

x2=np.random.uniform(-4,4,(1,n))
y2=np.random.uniform(-4,4,(1,n))

plt.scatter(x1, y1, color="blue",marker='*',label=" 正态分布" )
plt.scatter(x2, y2, color="yellow",marker='o' ,label="均匀分布")

plt.legend( )
plt.title("标准正态分布",fontsize=20)

plt.xlim(-4,4)
plt.ylim(-4,4)
```

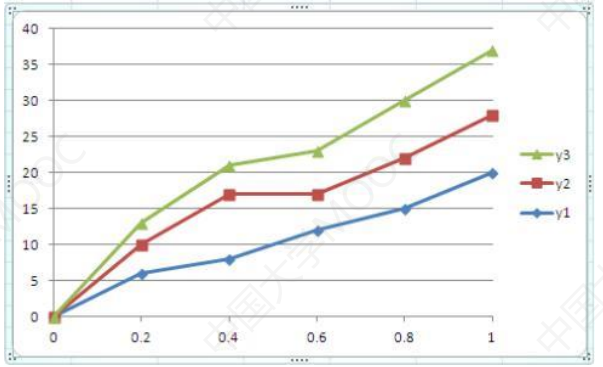



tensorflow / tensorflow
TensorFlow: Open sou
Clone this repo:
git clone https://tensor
Branches
master

6.3折线图和柱状图

- **折线图**（Line Chart）：散点图的基础上，**将相邻的点用线段相连接**
  - 描述变量变化的趋势



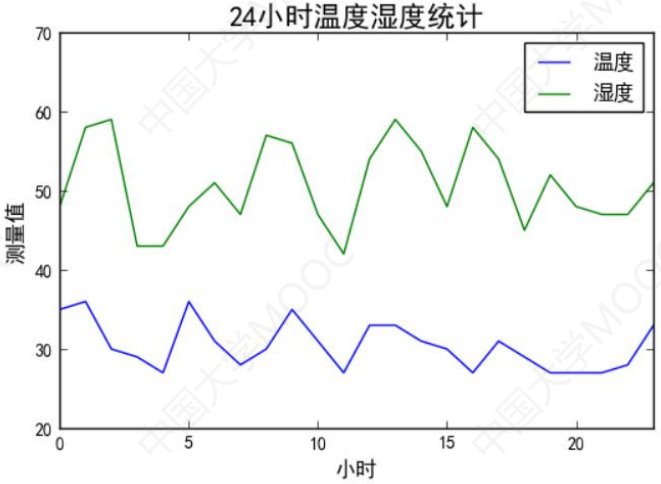


plot()函数

plot(**x, y,** color, marker, label, linewidth, markersize)

参 数	说 明	默认值
x	数据点的x坐标	0,1,2,3...
y	数据点的y坐标	不可省略
color	数据点的颜色	
marker	数据点的样式	’o’（圆点）
label	图例文字	
linewidth	折线的宽度	
markersize	数据点的大小	

例：绘制温度和湿度数据的折线图



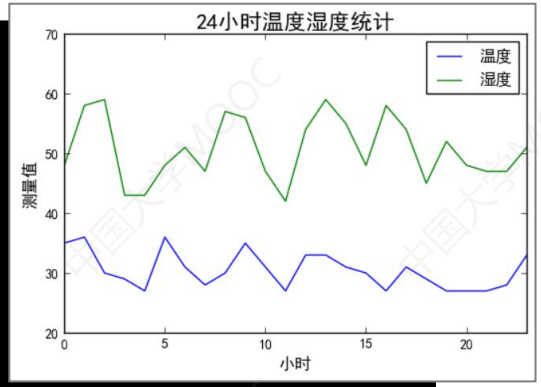
- 生成随机数列

```
n=24
y1 = np.random.randint(27,37,n)
y2 = np.random.randint(40,60,n)
```

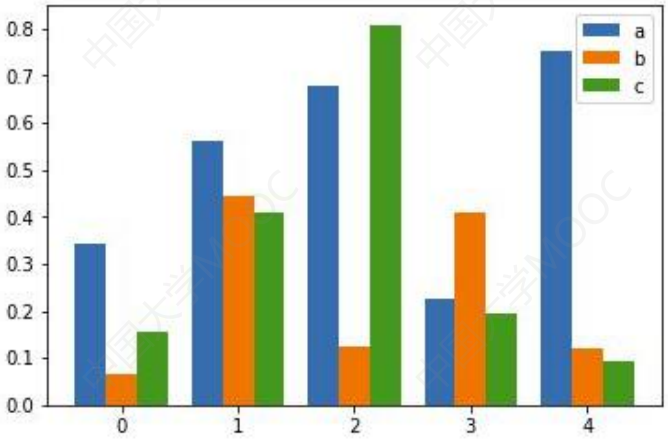
- 绘制折线图

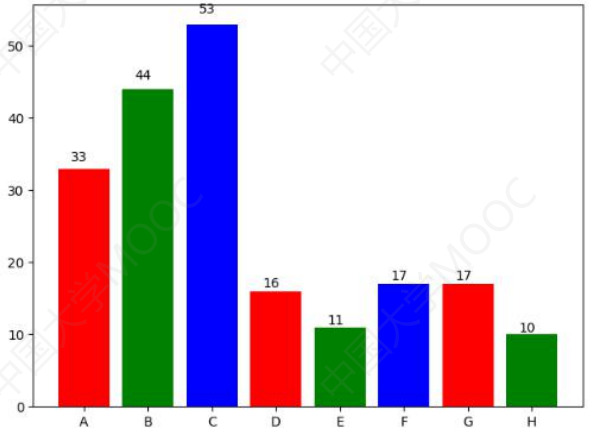
```
plt.plot(y1, label='温度')
plt.plot(y2, label='湿度')
```

```python
import matplotlib.pyplot as plt
import numpy as np

plt.rcParams['font.sans-serif'] = 'SimHei'

n = 24
y1 = np.random.randint(27,37,n)
y2 = np.random.randint(40,60,n)

plt.plot(y1, label='温度')
plt.plot(y2, label='湿度')

plt.xlim(0,23)
plt.ylim(20,70)
plt.xlabel('小时', fontsize=12)
plt.ylabel('测量值', fontsize=12)

plt.title('24小时温度湿度统计', fontsize=16)

plt.legend()
plt.show()
```



- **柱形图**（Bar Chart）：由一系列高度不等的柱形条纹表示数据分布的情况

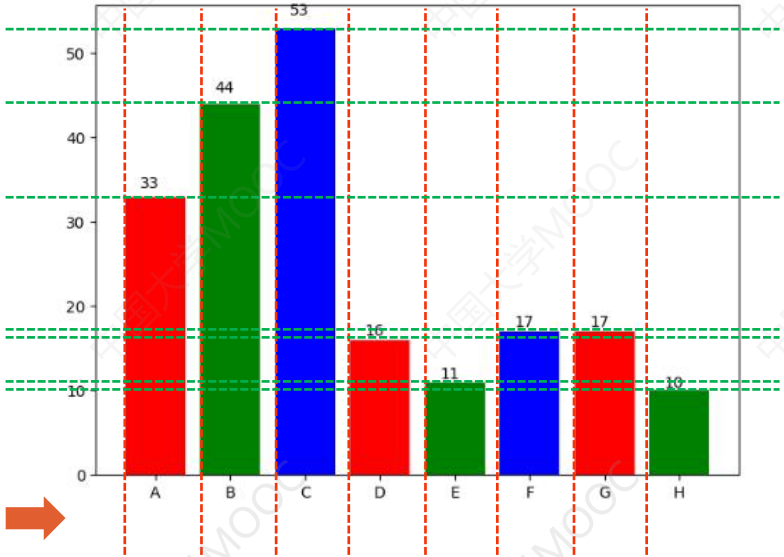




bar()函数

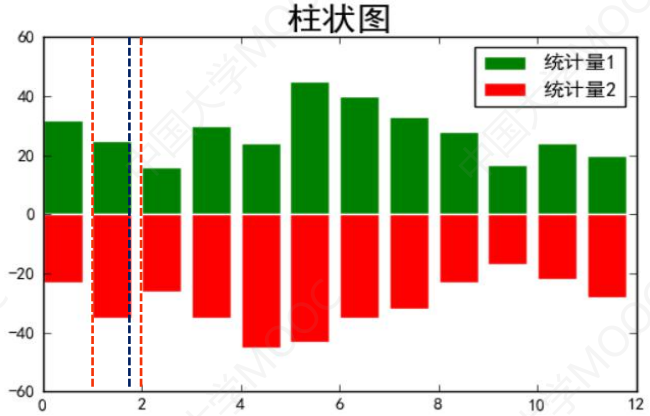
bar(left, **height**, width, facecolor, edgecolor, label)

height



left

例：绘制柱形图



- 条纹高度

```
y1=[32,25,16,30,24,45,40,33,28,17,24,20]
y2=[-23,-35,-26,-35,-45,-43,-35,-32,-23,-17,-22,-28]
```

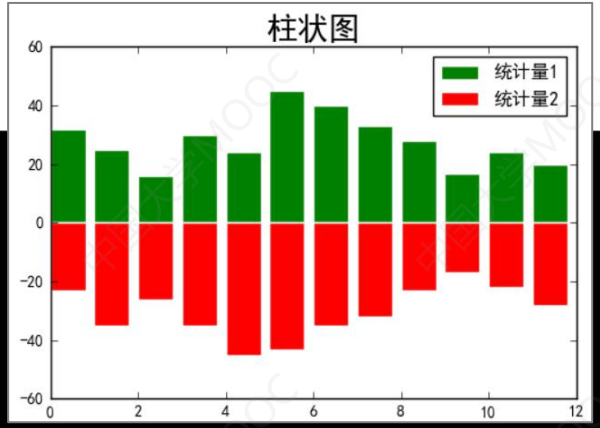
- 条纹left坐标

条纹的宽度**0.8**，每隔**1cm**开始画一个条纹

自动生成**left坐标**序列

```
plt.bar(range(len(y1)), y1,width=0.8,facecolor='green',edgecolor='white',label='统计量1')
plt.bar(range(len(y2)), y2,width=0.8,facecolor='red',edgecolor='white',label='统计量2')
```

例：绘制柱形图




```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  plt.rcParams['font.sans-serif']="SimHei"
5  plt.rcParams["axes.unicode_minus"] = False
6
7  y1=[32,25,16,30,24,45,40,33,28,17,24,20]
8  y2=[-23,-35,-26,-35,-45,-43,-35,-32,-23,-17,-22,-28]
9
10 plt.bar(range(len(y1)), y1,width=0.8,facecolor='green',edgecolor='white',label='统计量1')
11 plt.bar(range(len(y2)), y2,width=0.8,facecolor='red',edgecolor='white',label='统计量2')
12
13 plt.title("柱状图",fontsize=20)
14
15 plt.legend()
16 plt.show()
```

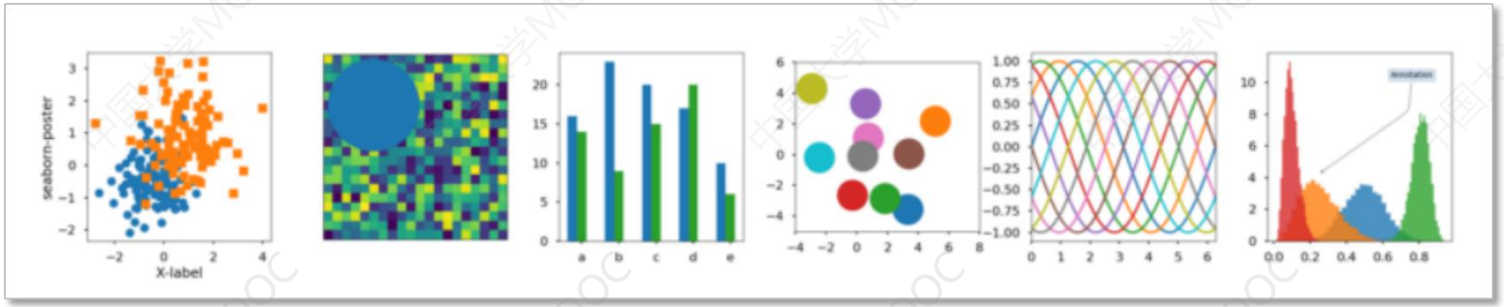
Matplotlib官网

http:// matplotlib.org

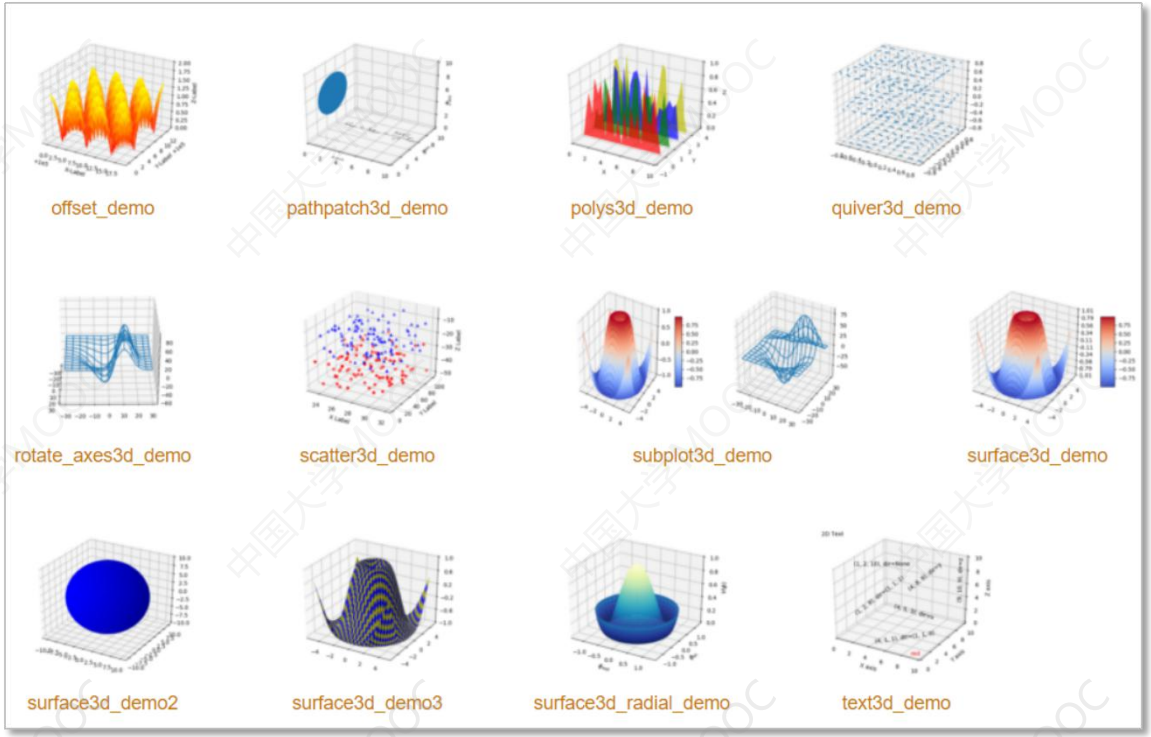
https://matplotlib.org/genindex.html

Gallery页面

https://matplotlib.org/gallery.html



缩略图

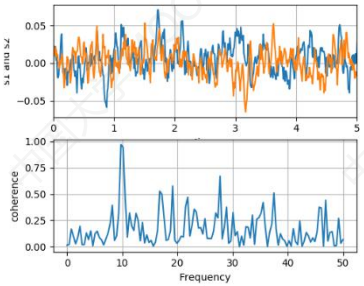

offset_demo
pathpatch3d_demo
polys3d_demo
quiver3d_demo
rotate_axes3d_demo
scatter3d_demo
subplot3d_demo
surface3d_demo
surface3d_demo2
surface3d_demo3
surface3d_radial_demo
text3d_demo

- 详细页面

  清晰大图&源码

pylab_examples example code: cohere_demo.py

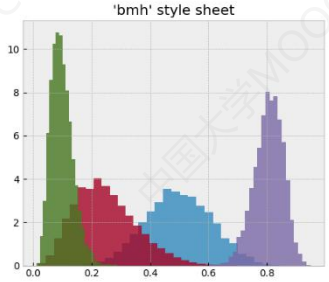
(Source code, png, pdf)




```
"""
Compute the coherence of two signals
"""
import numpy as np
import matplotlib.pyplot as plt

# make a little extra space between the subplots
plt.subplots_adjust(wspace=0.5)

dt = 0.01
t = np.arange(0, 30, dt)
nse1 = np.random.randn(len(t))                 # white noise 1
nse2 = np.random.randn(len(t))                 # white noise 2
r = np.exp(-t/0.05)

cnse1 = np.convolve(nse1, r, mode='same')*dt   # colored noise 1
cnse2 = np.convolve(nse2, r, mode='same')*dt   # colored noise 2

# two signals with a coherent part and a random part
s1 = 0.01*np.sin(2*np.pi*10*t) + cnse1
s2 = 0.01*np.sin(2*np.pi*10*t) + cnse2

plt.subplot(211)
plt.plot(t, s1, t, s2)
plt.xlim(0, 5)
plt.xlabel('time')
plt.ylabel('s1 and s2')
plt.grid(True)

plt.subplot(212)
cxy, f = plt.cohere(s1, s2, 256, 1./dt)
plt.ylabel('coherence')
plt.show()
```

style_sheets example code: plot_bmh.py

(Source code, png, pdf)




```
"""
=====================================
Bayesian Methods for Hackers style sheet
=====================================

This example demonstrates the style used in the Bayesian Methods for Hackers
[1]_ online book.

.. [1] http://camdavidsonpilon.github.io/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers/

"""
from numpy.random import beta
import matplotlib.pyplot as plt


plt.style.use('bmh')


def plot_beta_hist(ax, a, b):
    ax.hist(beta(a, b, size=10000), histtype="stepfilled",
            bins=25, alpha=0.8, normed=True)


fig, ax = plt.subplots()
plot_beta_hist(ax, 10, 10)
plot_beta_hist(ax, 4, 12)
plot_beta_hist(ax, 50, 12)
plot_beta_hist(ax, 6, 55)
ax.set_title("'bmh' style sheet")

plt.show()
```

Keywords: python, matplotlib, pylab, example, codex (see Search examples)

6.4 波士顿房价数据集可视化

Keras

- 是一个高层的**神经网络和深度学习库**。
- 可以**快速搭建神经网络模型**，非常**易于调试和扩展**。
- TensorFlow的**官方API**
- 内置了一些**常用的公共数据集**，可以通过keras.datasets模块加载和访问。

Keras中集成的数据集

序号	名 称	说 明
1	boston_housing	波士顿房价数据集
2	CIFAR10	10种类别的图片集
3	CIFAR100	100种类别的图片集
4	MNIST	手写数字图片集
5	Fashion-MNIST	10种时尚类别的图片集
6	IMDB	电影点评数据集
7	reuters	路透社新闻数据集

波士顿房价数据集

- 卡内基梅隆大学，StatLib库，1978年
- 涵盖了麻省波士顿的506个不同郊区的房屋数据
- 404条训练数据集，102条测试数据集
- 每条数据14个字段，包含13个属性，和1个房价的平均值

6.4 波士顿房价数据集可视化

序号	变量名	说 明	示 例
1	CRIM	城镇人均犯罪率	0.00632
2	ZN	超过25000平方英尺的住宅用地所占比例	18.0
3	INDUS	城镇非零售业的商业用地所占比例	2.31
4	CHAS	是否被Charles河流穿过（取值1：是；取值0：否）	0
5	NOX	一氧化氮浓度	0.538
6	RM	每栋住宅的平均房间数	6.575
7	AGE	早于1940年建成的自住房屋比例	65.2
8	DIS	到波士顿5个中心区域的加权平均距离	4.0900
9	RAD	到达高速公路的便利指数	1
10	TAX	每10000美元的全值财产税率	296
11	PTRATIO	城镇中师生比例	15.3
12	B	反映城镇中的黑人比例的指标，越靠近0.63越小；$B=1000*(BK-0.63)^2$，其中BK是黑人的比例。	396.90
13	LSTAT	低收入人口的比例	7.68
14	MEDV	自住房屋房价的平均房价（单位为1000美元）	24.0

加载数据集——.load_data()方法

前缀　　数据集名称

tensorflow.keras.datasets.boston_housing

```
import tensorflow as tf
boston_housing = tf.keras.datasets.boston_housing

(train_x, train_y), (test_x, test_y) = boston_housing.load_data()
```

提示信息：

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/boston_housing.npz
57344/57026 [==============================] - 1s 23us/step
```

本地默认路径：

C:\Users\user_name\.keras\datasets
C:\Users\Administrator\.keras\datasets\boston_housing.npz

训练集合测试集

```
import tensorflow as tf
boston_housing = tf.keras.datasets.boston_housing

(train_x, train_y), (test_x, test_y) = boston_housing.load_data()
```

训练数据集

测试数据集

test_split=0.2

```
print("Training set:",len(train_x))
print("Testing set:",len(test_x))
```

运行结果：

```
Training set: 404
Testing set: 102
```

改变数据集划分比例

提取出全部数据作为训练集

```
(train_x, train_y), (test_x, test_y) = boston_housing.load_data(test_split=0)
```

```
print("Training set:",len(train_x))
print("Testing set:",len(test_x))
```

运行结果：

```
Training set: 506
Testing set: 0
```

访问数据集中的数据

```
>>>type(train_x)
numpy.ndarray
>>>type(train_y)
numpy.ndarray

>>>print("Dim of train_x:",train_x.ndim)
Dim of train_x: 2
>>>print("Shape of train_x:",train_x.shape)
Shape of train_x: (506, 13)

>>>print("Dim of train_y:",train_y.ndim)
Dim of train_y: 1
>>>print("Shape of train_y:",train_y.shape)
Shape of train_y: (506,)
```

- 访问数据集中的数据——输出train_x中的**前5行**数据

```
>>>print(train_x[0:5])
[[1.23247e+00  0.00000e+00  8.14000e+00  0.00000e+00  5.38000e-01  6.14200e+00
  9.17000e+01  3.97690e+00  4.00000e+00  3.07000e+02  2.10000e+01  3.96900e+02
  1.87200e+01]
 [2.17700e-02  8.25000e+01  2.03000e+00  0.00000e+00  4.15000e-01  7.61000e+00
  1.57000e+01  6.27000e+00  2.00000e+00  3.48000e+02  1.47000e+01  3.95380e+02
  3.11000e+00]
 [4.89822e+00  0.00000e+00  1.81000e+01  0.00000e+00  6.31000e-01  4.97000e+00
  1.00000e+02  1.33250e+00  2.40000e+01  6.66000e+02  2.02000e+01  3.75520e+02
  3.26000e+00]
 [3.96100e-02  0.00000e+00  5.19000e+00  0.00000e+00  5.15000e-01  6.03700e+00
  3.45000e+01  5.98530e+00  5.00000e+00  2.24000e+02  2.02000e+01  3.96900e+02
  8.01000e+00]
 [3.69311e+00  0.00000e+00  1.81000e+01  0.00000e+00  7.13000e-01  6.37600e+00
  8.84000e+01  2.56710e+00  2.40000e+01  6.66000e+02  2.02000e+01  3.91430e+02
  1.46500e+01]]
```

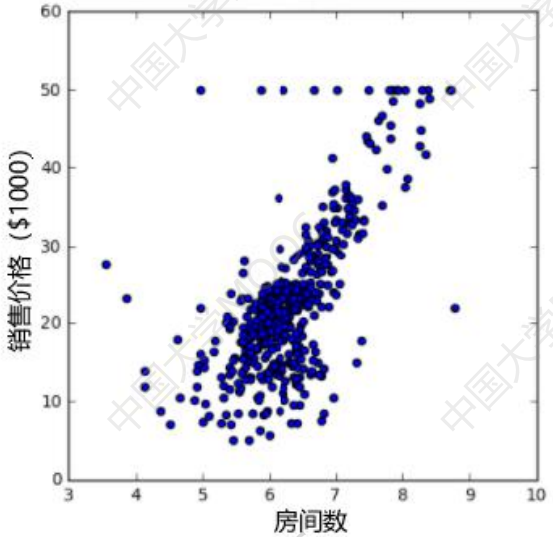
- 访问数据集中的数据——输出train_x中的**第6列**数据

```
>>>print(train_x[:, 5])
[6.142 7.61  4.97  6.037 6.376 5.708 5.536 5.468 5.628 5.019 6.404 4.628
 5.572 6.251 5.613 5.957 7.016 6.345 6.162 6.727 6.202 6.595 7.135 6.575
 5.895 6.794 6.012 7.185 5.813 5.569 6.315 6.297 6.301 5.935 7.024 6.415
 5.599 5.701 6.041 6.279 5.454 6.211 6.316 6.411 5.887 5.924 5.822 6.674
 6.842 5.713 5.968 6.461 7.358 6.565 5.88  5.87  6.348 6.193 6.854 6.546
 ......
 5.036 5.813 7.185 6.63  6.343 8.297 6.758 6.421 6.98  6.471 6.852 6.019
 6.376 6.108 6.417 6.209 5.093 5.987 6.395 6.957 6.229 5.414 6.495 6.009
 5.885 6.375 6.968 4.88  5.981 7.52  5.593 6.485 5.705 6.172 6.229 5.951
 6.593 7.061 6.03  5.884 6.897 8.259 6.812 6.122 7.333 8.78  6.273 7.802
 6.951 6.101]
```

- 访问数据集中的数据——输出train_y中的全部数据

```
>>>print(train_y)
[15.2 42.3 50.  21.1 17.7 18.5 11.3 15.6 15.6 14.4 12.1 17.9 23.1 19.9
 15.7  8.8 50.  22.5 24.1 27.5 10.9 30.8 32.9 24.  18.5 13.3 22.9 34.7
 16.6 17.5 22.3 16.1 14.9 23.1 34.9 25.  13.9 13.1 20.4 20.  15.2 24.7
 22.2 16.7 12.7 15.6 18.4 21.  30.1 15.1 18.7  9.6 31.5 24.8 19.1 22.
 14.5 11.  32.  29.4 20.3 24.4 14.6 19.5 14.1 14.3 15.6 10.5  6.3 19.3
 ......
 32.5 29.6 28.4 19.8 20.2 25.  35.4 20.3  9.7 14.5 34.9 26.6  7.2 50.
 32.4 21.6 29.8 13.1 27.5 21.2 23.1 21.9 13.  23.2  8.1  5.6 21.7 29.6
 19.6  7.  26.4 18.9 20.9 28.1 35.4 10.2 24.3 43.1 17.6 15.4 16.2 27.1
 21.4 21.5 22.4 25.  16.6 18.6 22.  42.8 35.1 21.5 36.  21.9 24.1 50.
 26.7 25. ]
```

平均房间数与房价之间的关系

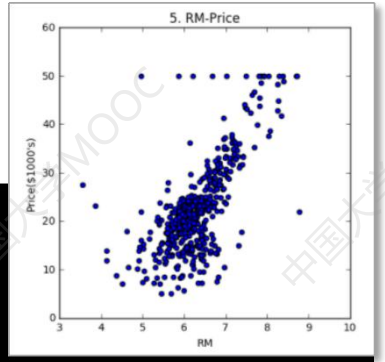


绘制散点图　　平均房间数　　房价

```
plt.scatter(train_x[:, 5],train_y)
```

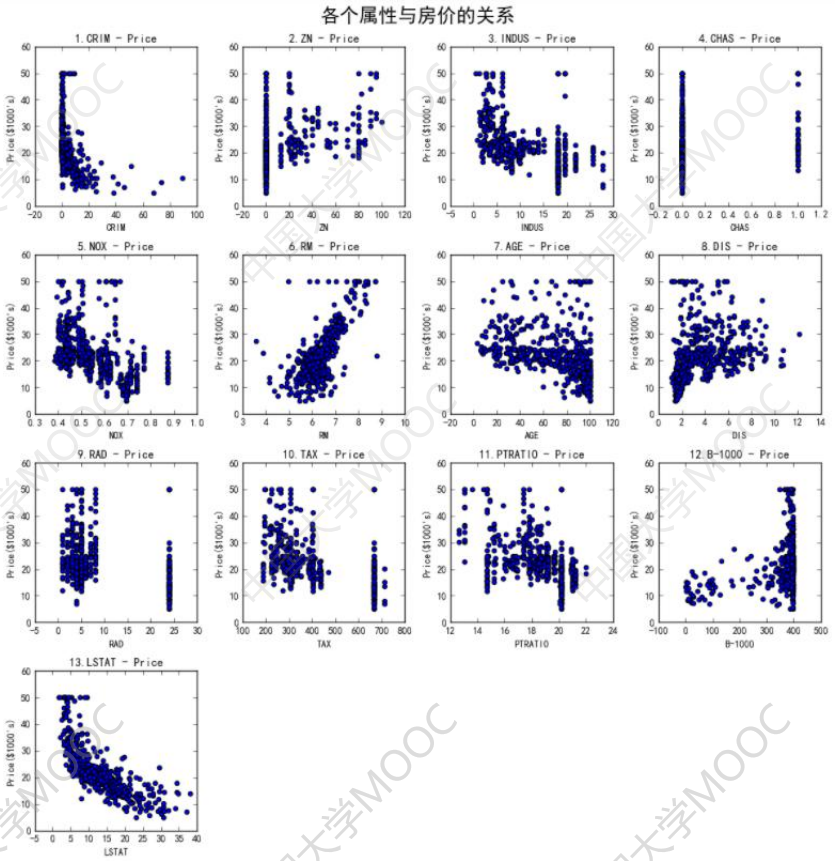

西安科技大学
计算机科学与技术学院

例：将平均房间数与房价之间的关系可视化




```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

boston_housing = tf.keras.datasets.boston_housing
(train_x, train_y), (_,_) = boston_housing.load_data(test_split=0)

plt.figure(figsize=(5,5))                    #设置绘图尺寸
plt.scatter(train_x[:, 5],train_y)           #绘制散点图
plt.xlabel("RM")                             #设置x轴标签文本
plt.ylabel("Price($1000's)")                 #设置y轴标签文本
plt.title("5. RM-Price")                     #设置标题
plt.show()                                   #显示绘图
```

例：将**所有属性**与房价之间的关系可视化



```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

boston_housing = tf.keras.datasets.boston_housing
(train_x, train_y), (_, _) = boston_housing.load_data(test_split=0)

plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False

titles = ["CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE",
          "DIS", "RAD", "TAX", "PTRATIO", "B-1000", "LSTAT", "MEDV"]

plt.figure(figsize=(12,12))

for i in range(13):
    plt.subplot(4,4,(i+1))
    plt.scatter(train_x[:,i], train_y)
    plt.xlabel(titles[i])
    plt.ylabel("Price($1000's)")
    plt.title(str(i+1)+ "." +titles[i]+" - Price")

plt.tight_layout()
plt.suptitle("各个属性与房价的关系", x=0.5, y=1.02, fontsize=20)
plt.show()
```

6.5 鸢尾花数据集可视化

6.5.1 下载鸢尾花数据集

鸢尾花

中国大学MOOC

Iris 数据集

- Anderson's Iris Data Set
- 1936，Ronald Fisher，*The use of multiple measurements in taxonomic problems*

- 加拿大的加斯帕半岛
- 同一天的同一个时间段
- 在相同的牧场上
- 由同一个人
- 使用相同的测量仪器

- 3种鸢尾花类别
- 每个类别有50个样本
- 每个样本中包括4种鸢尾花的属性特征，和鸢尾花的品种

Iris数据集

- 150个样本
- 4个属性
  - 花萼长度（Sepal Length）
  - 花萼宽度（Sepal Width）
  - 花瓣长度（Petal Length）
  - 花瓣宽度（Petal Width）
- 1个标签
  - 山鸢尾（Setosa）
  - 变色鸢尾（Versicolour）
  - 维吉尼亚鸢尾（Virginica）

Iris数据集示例

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	I. setosa
4.9	3	1.4	0.2	I. setosa
4.7	3.2	1.3	0.2	I. setosa
4.6	3.1	1.5	0.2	I. setosa
5	3.6	1.4	0.3	I. setosa
7	3.2	4.7	1.4	I. versicolor
6.4	3.2	4.5	1.5	I. versicolor
6.9	3.1	4.9	1.5	I. versicolor
6.3	3.3	6	2.5	I. virginica
5.8	2.7	5.1	1.9	I. virginica
7.1	3	5.9	2.1	I. virginica
6.3	2.9	5.6	1.8	I. virginica
6.5	3	5.8	2.2	I. virginica

get_file()函数——下载数据集

```
tf.keras.utils.get_file(fname, origin, cache_dir)
```

- 参数

  fname：下载后的文件名；

  origin：文件的URL地址；

  cache_dir：下载后文件的存储位置。C:\Users\\Administrator\.keras\datasets

  当前用户名

- 返回值：下载后的文件在本地磁盘中的**绝对路径**。

下载鸢尾花数据集 iris

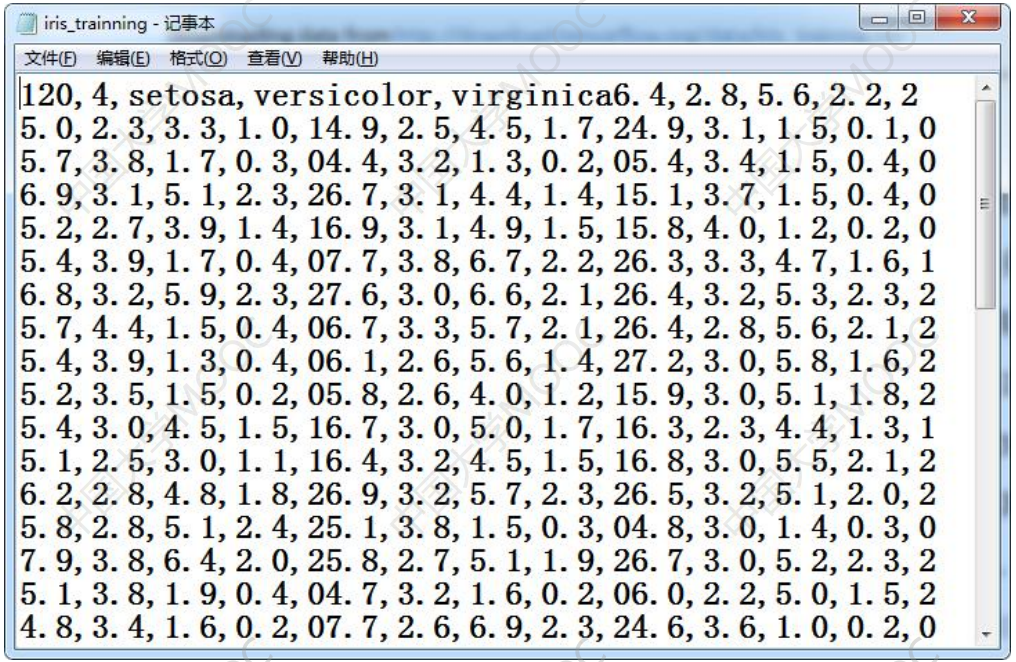
训练数据集	iris_training.csv	120条数据
测试数据集	iris_test.cvs	30条数据

```
TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
train_path = tf.keras.utils.get_file("iris_trainning.csv", TRAIN_URL)
```

```
Downloading data from http://download.tensorflow.org/data/iris_training.csv
8192/2194 [==============================================] - 1s 66us/step
```

C:\Users\Administrator\.keras\datasets\iris_training.csv

- csv文件


iris_trainning - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
120, 4, setosa, versicolor, virginica6. 4, 2. 8, 5. 6, 2. 2, 2
5. 0, 2. 3, 3. 3, 1. 0, 14. 9, 2. 5, 4. 5, 1. 7, 24. 9, 3. 1, 1. 5, 0. 1, 0
5. 7, 3. 8, 1. 7, 0. 3, 04. 4, 3. 2, 1. 3, 0. 2, 05. 4, 3. 4, 1. 5, 0. 4, 0
6. 9, 3. 1, 5. 1, 2. 3, 26. 7, 3. 1, 4. 4, 1. 4, 15. 1, 3. 7, 1. 5, 0. 4, 0
5. 2, 2. 7, 3. 9, 1. 4, 16. 9, 3. 1, 4. 9, 1. 5, 15. 8, 4. 0, 1. 2, 0. 2, 0
5. 4, 3. 9, 1. 7, 0. 4, 07. 7, 3. 8, 6. 7, 2. 2, 26. 3, 3. 3, 4. 7, 1. 6, 1
6. 8, 3. 2, 5. 9, 2. 3, 27. 6, 3. 0, 6. 6, 2. 1, 26. 4, 3. 2, 5. 3, 2. 3, 2
5. 7, 4. 4, 1. 5, 0. 4, 06. 7, 3. 3, 5. 7, 2. 1, 26. 4, 2. 8, 5. 6, 2. 1, 2
5. 4, 3. 9, 1. 3, 0. 4, 06. 1, 2. 6, 5. 6, 1. 4, 27. 2, 3. 0, 5. 8, 1. 6, 2
5. 2, 3. 5, 1. 5, 0. 2, 05. 8, 2. 6, 4. 0, 1. 2, 15. 9, 3. 0, 5. 1, 1. 8, 2
5. 4, 3. 0, 4. 5, 1. 5, 16. 7, 3. 0, 5. 0, 1. 7, 16. 3, 2. 3, 4. 4, 1. 3, 1
5. 1, 2. 5, 3. 0, 1. 1, 16. 4, 3. 2, 4. 5, 1. 5, 16. 8, 3. 0, 5. 5, 2. 1, 2
6. 2, 2. 8, 4. 8, 1. 8, 26. 9, 3. 2, 5. 7, 2. 3, 26. 5, 3. 2, 5. 1, 2. 0, 2
5. 8, 2. 8, 5. 1, 2. 4, 25. 1, 3. 8, 1. 5, 0. 3, 04. 8, 3. 0, 1. 4, 0. 3, 0
7. 9, 3. 8, 6. 4, 2. 0, 25. 8, 2. 7, 5. 1, 1. 9, 26. 7, 3. 0, 5. 2, 2. 3, 2
5. 1, 3. 8, 1. 9, 0. 4, 04. 7, 3. 2, 1. 6, 0. 2, 06. 0, 2. 2, 5. 0, 1. 5, 2
4. 8, 3. 4, 1. 6, 0. 2, 07. 7, 2. 6, 6. 9, 2. 3, 24. 6, 3. 6, 1. 0, 0. 2, 0

- iris_training.csv
  - 120个样本
  - 4个属性
    - 花萼长度（Sepal Length）
    - 花萼宽度（Sepal Width）
    - 花瓣长度（Petal Length）
    - 花瓣宽度（Petal Width）
  - 1个标签
    - 0- 山鸢尾（Setosa）
    - 1- 变色鸢尾（Versicolour）
    - 2- 维吉尼亚鸢尾（Virginica）

	A	B	C	D	E
1	120	4	setosa	versicolor	virginica
2	6.4	2.8	5.6	2.2	2
3	5	2.3	3.3	1	1
4	4.9	2.5	4.5	1.7	2
5	4.9	3.1	1.5	0.1	0
6	5.7	3.8	1.7	0.3	0
7	4.4	3.2	1.3	0.2	0
8	5.4	3.4	1.5	0.4	0
9	6.9	3.1	5.1	2.3	2
10	6.7	3.1	4.4	1.4	1
11	5.1	3.7	1.5	0.4	0
12	5.2	2.7	3.9	1.4	1
13	6.9	3.1	4.9	1.5	1
14	5.8	4	1.2	0.2	0
15	5.4	3.9	1.7	0.4	0

下载数据集

```
TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
train_path = tf.keras.utils.get_file("iris_trainning.csv", TRAIN_URL)
```

- split()函数：通过**指定的分隔符**对字符串进行**切片**，并返回一个**列表**。

```
TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
TRAIN_URL.split('/')
```

运行结果：

```
['http:', '', 'download.tensorflow.org', 'data', 'iris_training.csv']
```

获取文件名

```
TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
fname_list=TRAIN_URL.split('/')
fname_list[-1]
```

```
TRAIN_URL.split('/')[-1]
```

运行结果：

```
'iris_training.csv '
```

下载数据集

下载其他数据集时，只要改变URL地址即可

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

自动获取数据集文件名

神经网络&深度学习
TENSORFLOW
Google

6.5.2 Pandas访问csv数据集

中国大学MOOC

- **Pandas库**（**Panel Data** & **Data Analysis**）
  - 用于**数据统计**和**分析**
  - 可以高效、方便地操作大型数据集

- 导入Pandas库

```
import pandas as pd
```


中国大学MOOC

读取csv数据集文件

文件名参数——filepath_or_buffer

pd.read_csv(filepath_or_buffer, header, names)

直接使用绝对路径

```
pd.read_csv("C:/Users/Administrator/.keras/datasets/iris_training.csv")
```

```
TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
train_path = tf.keras.utils.get_file("iris_trainning.csv", TRAIN_URL)
pd.read_csv(train_path)
```

使用get_file()函数的返回值

运行结果：

	120	4	setosa	versicolor	virginica
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0
5	4.4	3.2	1.3	0.2	0
...	...	...	...	...	...
115	5.5	2.6	4.4	1.2	1
116	5.7	3.0	4.2	1.2	1
117	4.4	2.9	1.4	0.2	0
118	4.8	3.0	1.4	0.1	0
119	5.5	2.4	3.7	1.0	1

120 rows × 5 columns

二维数据表——DataFrame

是一种Pandas中常用的数据类型

```
TRAIN_URL = "http://download.tensorflow.org/data/iris_training.csv"
train_path = tf.keras.utils.get_file("iris_trainning.csv", TRAIN_URL)
df_iris=pd.read_csv(train_path)
```

```
>>>type(df_iris)
pandas.core.frame.DataFrame
```

设置列标题——header参数

```
pd.read_csv( filepath_or_buffer, header, names)
```

header 的取值是行号，行号**从0开始**

header=0，第1行数据做为列标题(默认设置)

header=None, 没有列标题

- **header=0，第1行数据作为列标题**

```
df_iris = pd.read_csv(train_path, header=0)
df_iris.head()
```

读取前5行数据

运行结果：

	120	4	setosa	versicolor	virginica
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

- **header=None**，数据文件中没有表头

```
df_iris = pd.read_csv(train_path, header=None)
df_iris.head()
```

运行结果：

	0	1	2	3	4
0	120.0	4.0	setosa	versicolor	virginica
1	6.4	2.8	5.6	2.2	2
2	5.0	2.3	3.3	1.0	1
3	4.9	2.5	4.5	1.7	2
4	4.9	3.1	1.5	0.1	0

系统自动的加上了数字序列 0,1,2,3,4 作为列标题

第一行数据被作为数据样本

设置列标题——names参数

自定义列标题，代替header参数指定的列标题

```
pd.read_csv( filepath_or_buffer, header, names)
```

- **header=0**，第1行做为列标题

	120	4	setosa	versicolor	virginica
0	6.4	2.8	5.6	2.2	2
1	5	2.3	3.3	1	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

- 设置**names参数**，指定新的列标题

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	6.4	2.8	5.6	2.2	2
1	5	2.3	3.3	1	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

中国大学MOOC

指定列标题

```
COLUMN_NAMES = ['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Species']
df_iris = pd.read_csv(train_path, names=COLUMN_NAMES, header=0)
df_iris.head()
```

运行结果：

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0

访问数据

head(n)

- **head()函数**：读取**前n行**数据

  参数为**空**时，默认读取二维数据表中的**前5行**数据

```
df_iris.head(8)
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
0	6.4	2.8	5.6	2.2	2
1	5.0	2.3	3.3	1.0	1
2	4.9	2.5	4.5	1.7	2
3	4.9	3.1	1.5	0.1	0
4	5.7	3.8	1.7	0.3	0
5	4.4	3.2	1.3	0.2	0
6	5.4	3.4	1.5	0.4	0
7	6.9	3.1	5.1	2.3	2

tail()函数：读取后n行数据

tail(n)

```
df_iris.tail(8)
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
112	5.0	3.0	1.6	0.2	0
113	6.3	3.3	6.0	2.5	2
114	5.0	3.5	1.6	0.6	0
115	5.5	2.6	4.4	1.2	1
116	5.7	3.0	4.2	1.2	1
117	4.4	2.9	1.4	0.2	0
118	4.8	3.0	1.4	0.1	0
119	5.5	2.4	3.7	1.0	1

使用索引和切片

- 读取**索引值为10-15**的行

```
df_iris[10:16]
```

	SepalLength	SepalWidth	PetalLength	PetalWidth	Species
10	5.2	2.7	3.9	1.4	1
11	6.9	3.1	4.9	1.5	1
12	5.8	4.0	1.2	0.2	0
13	5.4	3.9	1.7	0.4	0
14	7.7	3.8	6.7	2.2	2
15	6.3	3.3	4.7	1.6	1

显示统计信息

describe()方法：显示二维数据的统计信息

```
df_iris.describe()
```

		SepalLength	SepalWidth	PetalLength	PetalWidth	Species
总数	count	120.000000	120.000000	120.000000	120.000000	120.000000
平均值	mean	5.845000	3.065000	3.739167	1.196667	1.000000
标准差	std	0.868578	0.427156	1.822100	0.782039	0.840168
最小值	min	4.400000	2.000000	1.000000	0.100000	0.000000
1/4	25%	5.075000	2.800000	1.500000	0.300000	0.000000
1/2	50%	5.800000	3.000000	4.400000	1.300000	1.000000
3/4	75%	6.425000	3.300000	5.100000	1.800000	2.000000
最大值	max	7.900000	4.400000	6.900000	2.500000	2.000000

DataFrame的常用属性：ndim、size、shape

属　性	描　述
ndim	数据表的维数
shape	数据表的形状
size	数据表元素的总个数

```
>>>df_iris.ndim
2

>>>df_iris.shape
(120, 5)

>>>df_iris.shape
600
```

转化为NumPy数组

- 使用NumPy中的创建数组函数array()

```
>>>iris=np.array(df_iris)

>>>type(df_iris)
pandas.core.frame.DataFrame

>>>type(iris)
numpy.ndarray
```

- 使用DataFrame中的values方法或as_matrix()方法

```
>>>iris=df_iris.values
>>>iris=df_iris.as_matrix()
```

注：as_matrix()方法在2020年1月29日已被移除

访问数组元素——索引和切片

一维切片
读取**前6行**

```
>>>iris[0:6]
array([[6.4, 2.8, 5.6, 2.2, 2. ],
       [5. , 2.3, 3.3, 1. , 1. ],
       [4.9, 2.5, 4.5, 1.7, 2. ],
       [4.9, 3.1, 1.5, 0.1, 0. ],
       [5.7, 3.8, 1.7, 0.3, 0. ],
       [4.4, 3.2, 1.3, 0.2, 0. ]])
```

二维切片
读取**前6行**中的**前4列**

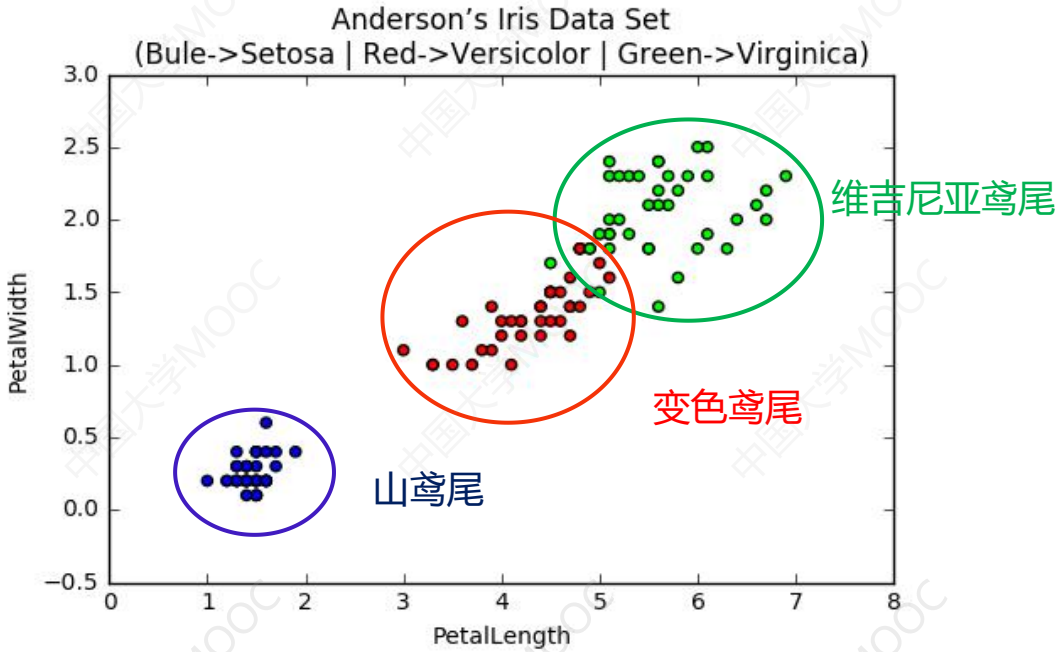
```
>>>iris[0:6,0:4]
array([[6.4, 2.8, 5.6, 2.2],
       [5. , 2.3, 3.3, 1. ],
       [4.9, 2.5, 4.5, 1.7],
       [4.9, 3.1, 1.5, 0.1],
       [5.7, 3.8, 1.7, 0.3],
       [4.4, 3.2, 1.3, 0.2]])
```

二维切片
所有行中的
鸢尾花种类

```
>>>iris_y=iris[:,4]
array([2., 1., 2., 0., 0., 0., 0., 2., 1., 0., 1., 1., 0., 0., 2., 1., 2.,
       2., 2., 0., 2., 2., 0., 2., 2., 0., 1., 2., 1., 1., 1., 1., 1., 2.,
       2., 2., 2., 2., 0., 0., 2., 2., 2., 0., 0., 2., 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., 1., 0., 2., 1., 0., 0., 2., 0., 0., 2., 1., 0., 0., 1.,
       0., 1., 0., 0., 0., 0., 1., 0., 2., 1., 0., 2., 0., 1., 1., 0., 0.,
       1.])
```

6.5.3 鸢尾花数据集可视化

鸢尾花数据散点图


Anderson's Iris Data Set
(Bule->Setosa | Red->Versicolor | Green->Virginica)
PetalWidth
PetalLength
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0
1
2
3
4
5
6
7
8
维吉尼亚鸢尾
变色鸢尾
山鸢尾

- 花瓣长度

```
>>>iris[:,2]
array([5.6, 3.3, 4.5, 1.5, 1.7, 1.3, 1.5, 5.1, 4.4, 1.5, 3.9, 4.9, 1.2,
       1.7, 6.7, 4.7, 5.9, 6.6, 5.3, 1.5, 5.7, 5.6, 1.3, 5.6, 5.8, 1.5,
       4. , 5.1, 4.5, 5. , 4.4, 3. , 4.5, 5.5, 4.8, 5.7, 5.1, 5.1, 1.5,
       1.4, 6.4, 5.1, 5.2, 1.9, 1.6, 5. , 1.6, 6.9, 1. , 6. , 1.4, 4.4,
       4. , 1.2, 4.7, 4.8, 6.1, 5.1, 5.4, 3.5, 3.9, 5.6, 5. , 5.5, 4.5,
       6.3, 1.3, 6.1, 5.5, 1.5, 1.3, 4.6, 1.3, 6.1, 4.9, 1.5, 3.8, 4.2,
       4.5, 5.3, 1.5, 4.7, 4.6, 4.2, 5.6, 1.5, 4.8, 4.5, 5.1, 1.3, 5.2,
       4.7, 1.4, 1.5, 5.8, 1.4, 1.4, 6.7, 4.8, 1.6, 1.4, 3.3, 1.3, 4.1,
       1.6, 1.4, 1.5, 1.4, 3.6, 1.6, 4.9, 4.1, 1.6, 6. , 1.6, 4.4, 4.2,
       1.4, 1.4, 3.7])
```

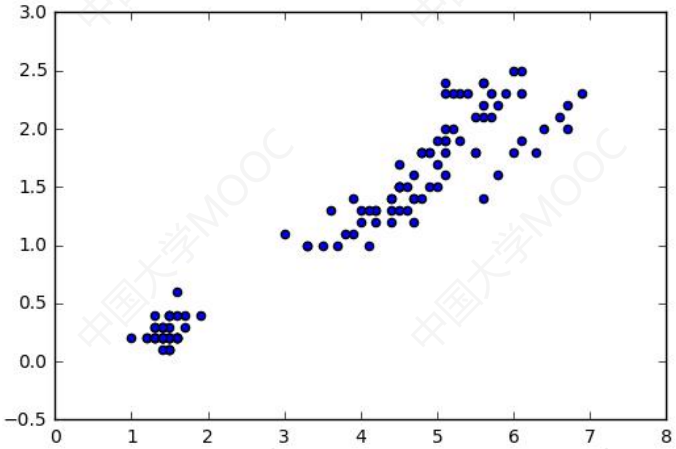
- 花瓣宽度

```
>>>iris[:,3]
array([2.2, 1. , 1.7, 0.1, 0.3, 0.2, 0.4, 2.3, 1.4, 0.4, 1.4, 1.5, 0.2,
       0.4, 2.2, 1.6, 2.3, 2.1, 2.3, 0.4, 2.1, 2.1, 0.4, 1.4, 1.6, 0.2,
       1.2, 1.8, 1.5, 1.7, 1.3, 1.1, 1.5, 2.1, 1.8, 2.3, 2. , 2.4, 0.3,
       0.3, 2. , 1.9, 2.3, 0.4, 0.2, 1.5, 0.2, 2.3, 0.2, 1.8, 0.2, 1.4,
       1.3, 0.2, 1.4, 1.8, 1.9, 1.9, 2.3, 1. , 1.1, 2.4, 1.9, 1.8, 1.5,
       1.8, 0.2, 2.5, 1.8, 0.2, 0.2, 1.3, 0.2, 2.3, 1.8, 0.1, 1.1, 1.3,
       1.5, 1.9, 0.2, 1.4, 1.5, 1.3, 2.4, 0.1, 1.4, 1.3, 1.6, 0.3, 2. ,
       1.2, 0.3, 0.2, 2.2, 0.3, 0.2, 2. , 1.8, 0.2, 0.2, 1. , 0.3, 1. ,
       0.4, 0.2, 0.2, 0.2, 1.3, 0.2, 1.8, 1.3, 0.2, 2.5, 0.6, 1.2, 1.2,
       0.2, 0.1, 1. ])
```

绘制散点图

```
plt.scatter(iris[:,2],iris[:,3])
plt.show()
```

运行结果：



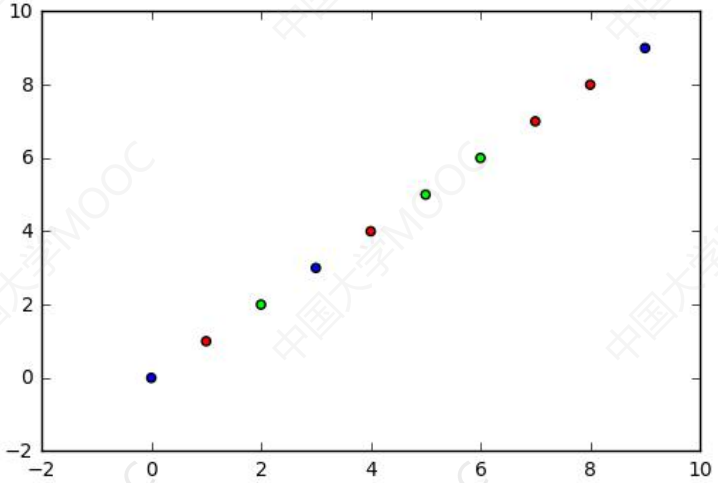
- **色彩映射**：将**参数c**指定为一个**列表或数组**，所绘制图形的颜色，可以随这个列表或数组中元素的值而**变换**，变换所对应的颜色由**参数cmap**中的颜色所提供。

colormap

```
plt.scatter(x, y, c, cmap )
```

```
x=np.arange(10)
y=np.arange(10)

dot_color=[0,1,2,0,1,2,2,1,1,0]

plt.scatter(x,y,c=dot_color,cmap='brg')
plt.show()
```

0 1 2

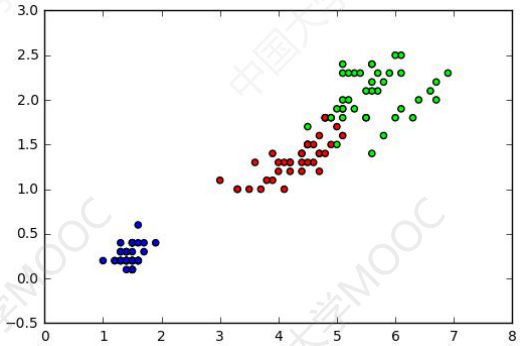


- 鸢尾花种类

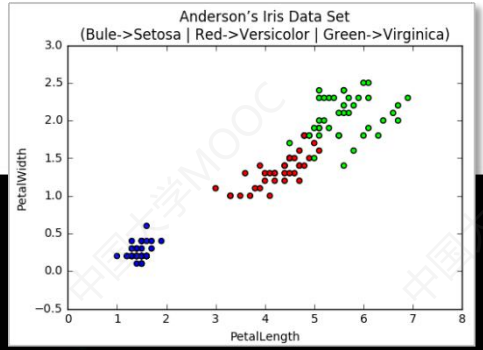
```
>>>iris_y=iris[:,4]
array([2., 1., 2., 0., 0., 0., 0., 2., 1., 0., 1., 1., 0., 0., 2., 1., 2.,
       2., 2., 0., 2., 2., 0., 2., 2., 0., 1., 2., 1., 1., 1., 1., 1., 2.,
       2., 2., 2., 2., 0., 0., 2., 2., 2., 0., 0., 2., 0., 2., 0., 2., 0.,
       1., 1., 0., 1., 2., 2., 2., 2., 1., 1., 2., 2., 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., 1., 0., 2., 1., 0., 0., 2., 0., 0., 2., 1., 0., 0., 1.,
       0., 1., 0., 0., 0., 0., 1., 0., 2., 1., 0., 2., 0., 1., 1., 0., 0.,
       1.])
```

```
plt.scatter(iris[:,2],iris[:,3],c=iris[:,4],cmap='brg')
plt.show()
```

运行结果：



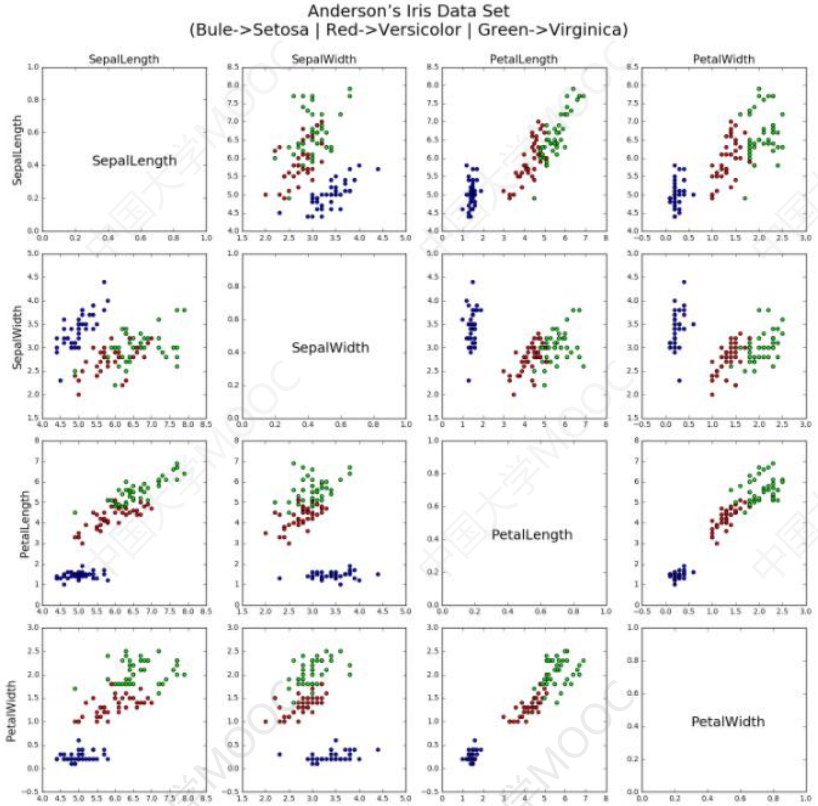
鸢尾花数据散点图 —— 2个属性



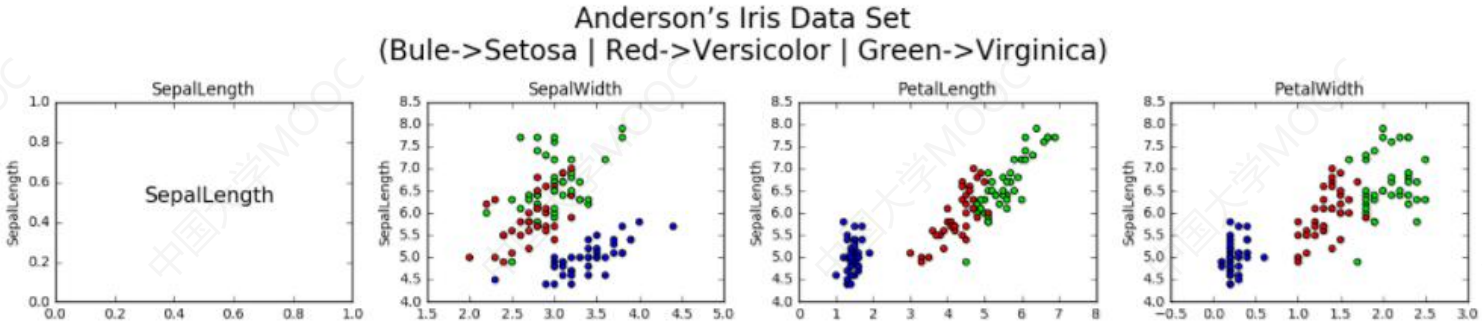

```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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

COLUMN_NAMES = ['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Species']
df_iris = pd.read_csv(train_path, names=COLUMN_NAMES, header=0)

iris=np.array(df_iris)

plt.scatter(iris[:,2],iris[:,3],c=iris[:,4],cmap='brg')
plt.title("Anderson's Iris Data Set\n(Bule->Setosa | Red->Versicolor | Green->Virginica)")
plt.xlabel(COLUMN_NAMES[2])
plt.ylabel(COLUMN_NAMES[3])
plt.show()
```

6.5.3 鸢尾花数据集可视化

	Sepal Length	Sepal Width	Petal Length	Petal Width
Sepal Length	X	O	O	O
Sepal Width	O	X	O	O
Petal Length	O	O	X	O
Petal Width	O	O	O	X



6.5.3 鸢尾花数据集可视化



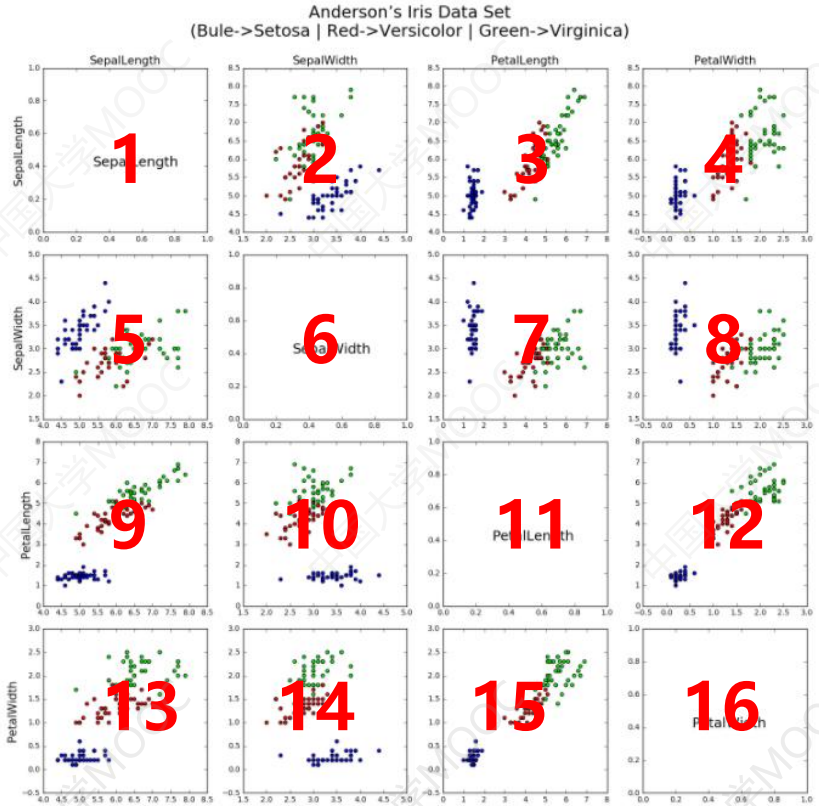

```
for i in range(4):
    plt.subplot(1, 4, i + 1)

    if(i==0):
        plt.text(0.3,0.5,COLUMN_NAMES[0],fontsize=15)
    else:
        plt.scatter(iris[:,i], iris[:,0], c=iris[:,4], cmap='brg')

    plt.title(COLUMN_NAMES[i])
    plt.ylabel(COLUMN_NAMES[0])
```

```
fig = plt.figure('Iris Data', figsize=(15, 3))

fig.suptitle("Anderson's Iris Data Set\n(Bule->Setosa | Red->Versicolor | Green->Virginica)")

for i in range(4):
    plt.subplot(1, 4, i + 1)
    if(i==0):
        plt.text(0.3,0.5,COLUMN_NAMES[0],fontsize=15)
    else:
        plt.scatter(iris[:,i], iris[:,0], c=iris[:,4], cmap='brg')

    plt.title(COLUMN_NAMES[i])
    plt.ylabel(COLUMN_NAMES[0])

plt.tight_layout(rect=[0,0,1,0.9])

plt.show()
```

鸢尾花数据集可视化

子图序号：

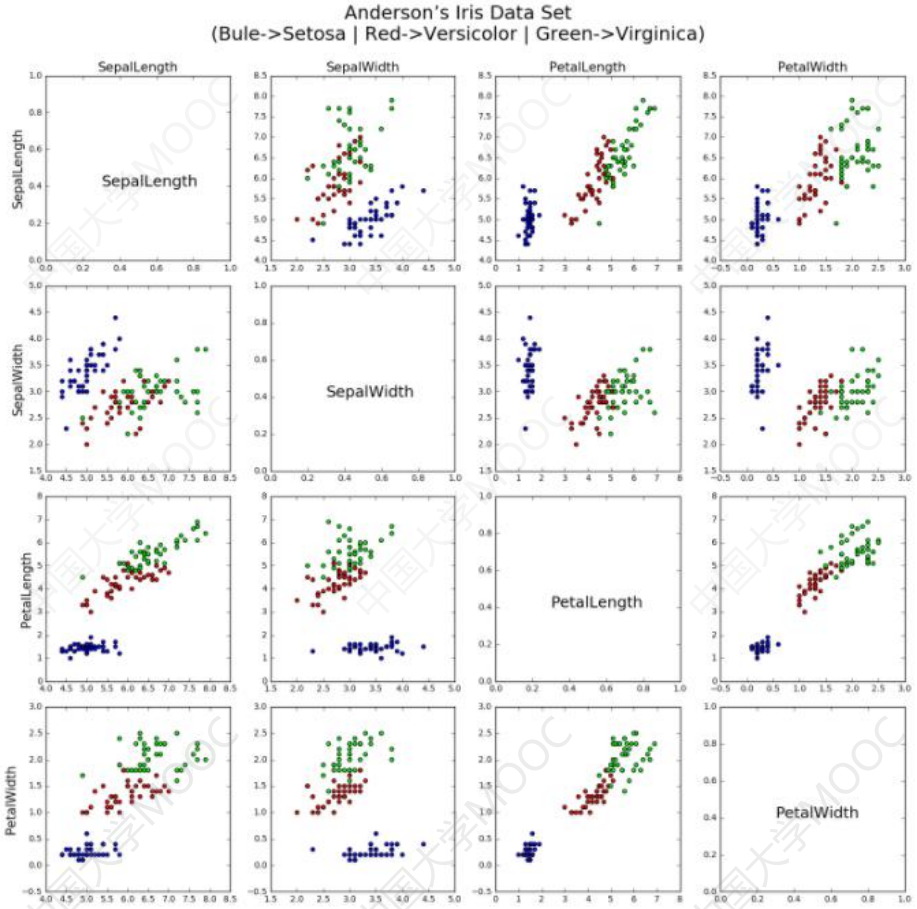
4×i + (j+1)

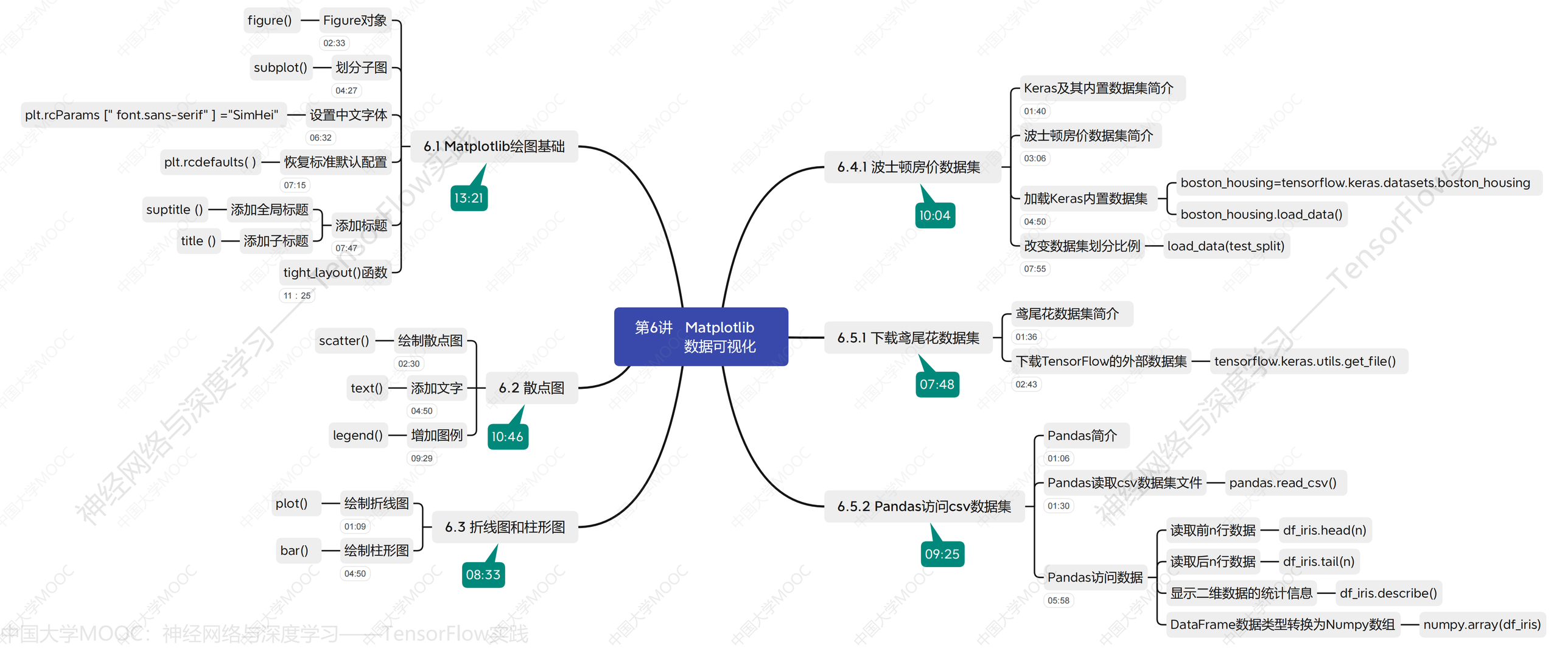


6.5.3 鸢尾花数据集可视化


```
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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

COLUMN_NAMES = ['SepalLength', 'SepalWidth', 'PetalLength', 'PetalWidth', 'Species']
df_iris = pd.read_csv(train_path, names=COLUMN_NAMES, header=0)
iris=np.array(df_iris)

fig = plt.figure('Iris Data', figsize=(15, 15))
fig.suptitle("Anderson's Iris Data Set\n(Bule->Setosa | Red->Versicolor | Green->Virginica)",fontsize=20)

for i in range(4):
    for j in range(4):
        plt.subplot(4, 4, 4*i + (j + 1))
        if(i==j):
            plt.text(0.3,0.4,COLUMN_NAMES[i],fontsize=15)
        else:
            plt.scatter(iris[:,j], iris[:,i], c=iris[:,4], cmap='brg')
        if(i == 0):
            plt.title(COLUMN_NAMES[j])
        if(j == 0):
            plt.ylabel(COLUMN_NAMES[i])

plt.show()
```

- 鸢尾花数据集可视化


Anderson's Iris Data Set
(Bule->Setosa | Red->Versicolor | Green->Virginica)
SepalLength
SepalWidth
PetalLength
PetalWidth



神经网络与深度学习——TensorFlow实践
中国大学MOOC：神经网络与深度学习——TensorFlow实践