神经网络&深度学习
TENSORFLOW
Google

09 回归问题

西安科技大学 牟琦
muqi@xust.edu.cn


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

9.1 机器学习基础

日晕三更雨，月晕午时风
有雨山戴帽，无雨云拦腰
朝霞不出门，晚霞行千里

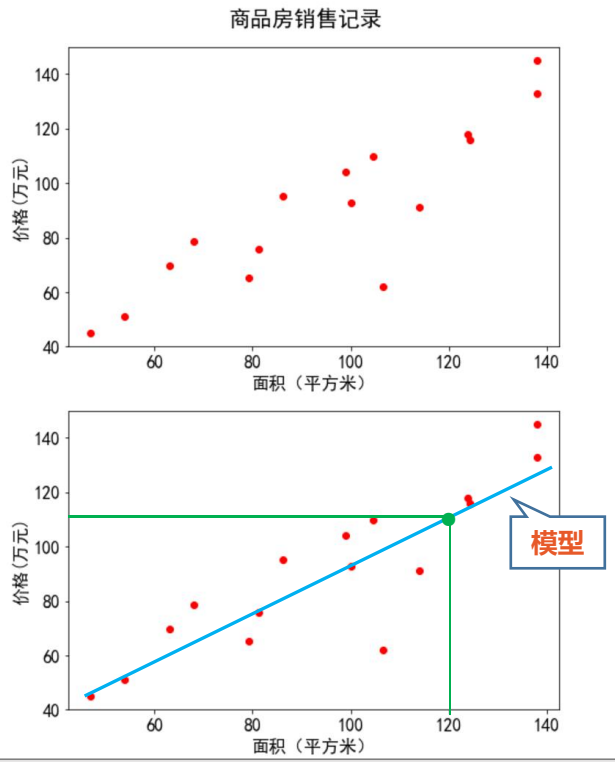
早晨东云长，有雨不过晌
早晨云挡坝，三天有雨下
早晨浮云走，午后晒死人
空中鱼鳞天，不雨也风颠
天上豆荚云，不久雨将临
天上铁砧云，很快大雨淋

机器学习：从数据中学习

- 建立模型
- 学习模型
- 预测房价

例：房屋销售记录

序号	面积（平方米）	销售价格（万元）	序号	面积（平方米）	销售价格（万元）
1	137.97	145.00	9	106.69	62.00
2	104.50	110.00	10	138.05	133.00
3	100.00	93.00	11	53.75	51.00
4	124.32	116.00	12	46.91	45.00
5	79.20	65.32	13	68.00	78.50
6	99.00	104.00	14	63.02	69.65
7	124.00	118.00	15	81.26	75.69
8	114.00	91.00	16	86.21	95.30

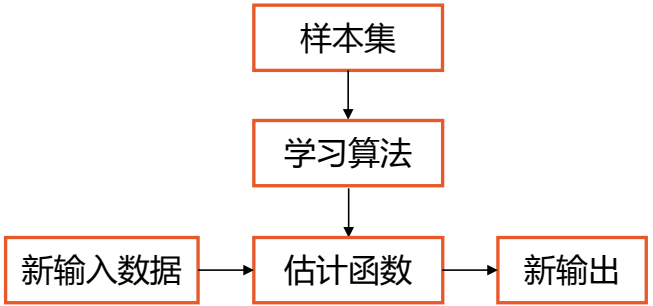


学习算法：从数据中产生模型的算法

监督学习 (Supervised Learning)

通过对**数据**的学习,寻找**属性**和**标记**之间的映射关系

- 数据集（data set）/样本集（sample set）
  - 示例（instance）/样本（sample）
  - 属性（attribute）/特征（feature）
  - 标记/标签（label）
- 模型/假设/估计函数/学习器（learner）
- 真相/真实（ground truth）



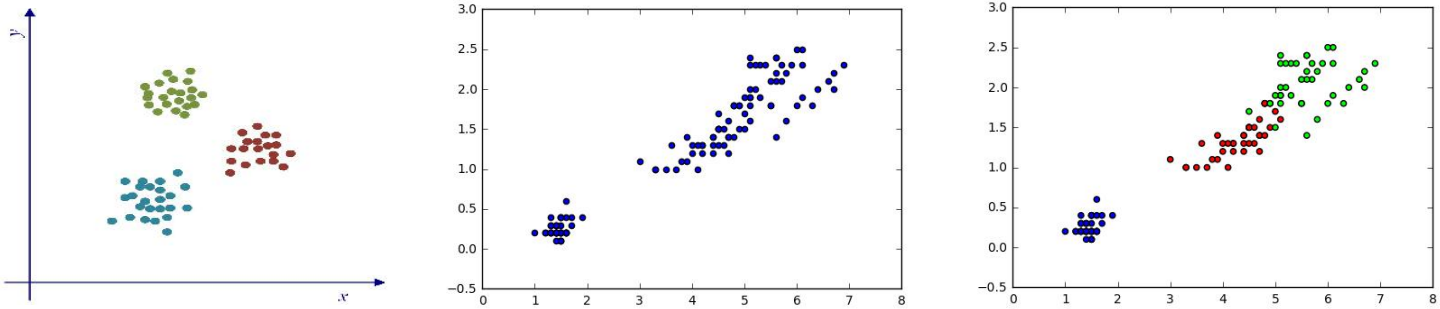
回归（regression）：预测**连续**值
分类（classification）：预测**离散**值

5

无监督学习（Unsupervised Learning）

在样本数据**没有标记**的情况下，挖掘出**数据**内部蕴含的**关系**

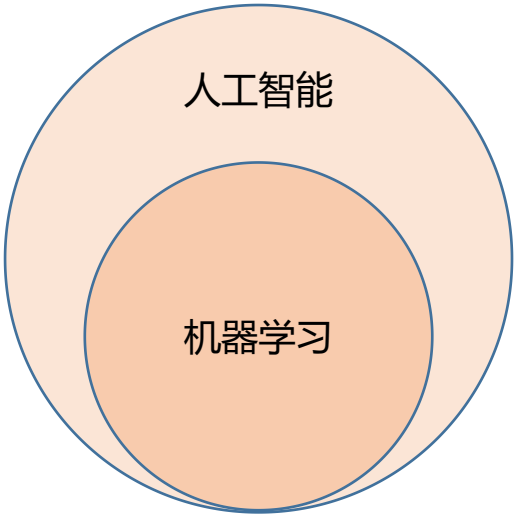
聚类：把**相似度**高的样本聚合在一起



半监督学习(Semi-Supervised Learning)

将有监督学习和无监督学习相结合，综合使用**大量的没有标记数据**和**少量有标记的数据**共同进行学习

■ 机器学习的发展和应用



神经网络&深度学习
TENSORFLOW
Google

9.2 一元线性回归

中国大学MOOC

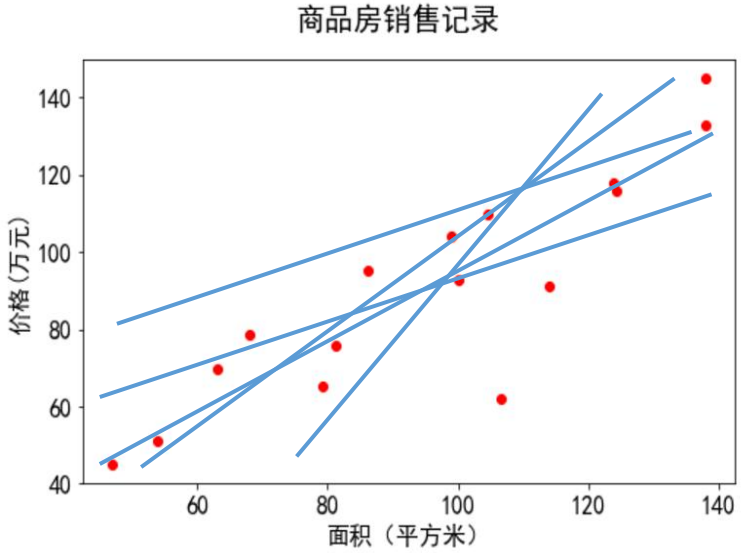
一元线性回归(Simple linear regression)

模型：$y = wx + b$

模型变量：x

模型参数

- w：权重（weights）
- b：偏置值（bias）



最佳拟合直线应该使得所有点的**残差累计值**最小

- 残差和最小

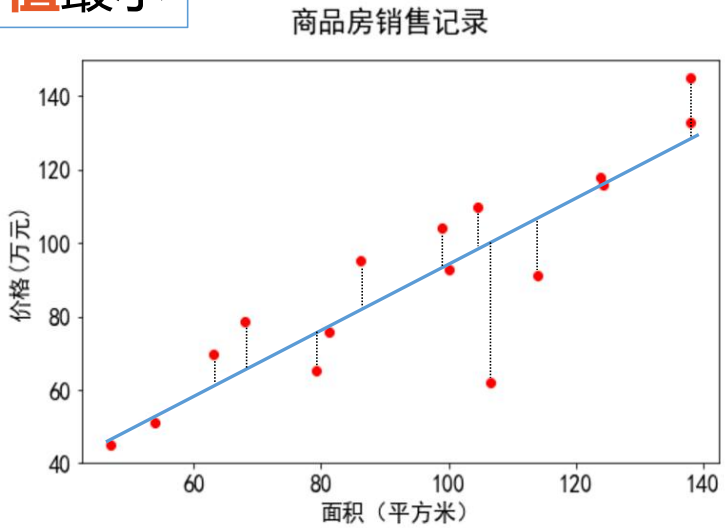
$$Loss = \sum_{i=1}^{n}(y_i - \hat{y}_i) = \sum_{i=1}^{n}(y_i - (wx_i + b))$$

- 残差绝对值和最小

$$Loss = \sum_{i=1}^{n}|y_i - \hat{y}_i| = \sum_{i=1}^{n}|y_i - (wx_i + b)$$

- 残差平方和最小

$$Loss = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - (wx_i + b))^2$$



- **损失函数/代价函数** (cost function)：估量模型的**预测值**与**真实值**的不一致程度
  - **平方损失函数** (Square Loss)

$$Loss = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - (wx_i + b))^2$$

  - **均方误差** (Mean Squqre Error)

$$Loss = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{2n}\sum_{i=1}^{n}(y_i - (wx_i + b))^2$$

- **最小二乘法**：基于**均方误差最小化**来进行模型求解的方法

$$Loss = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{2}\sum_{i=1}^{n}(y_i - (wx_i + b))^2$$

已知：样本点 (x_i, y_i)
未知：变量为 w, b

求极值问题： w, b 取何值时，损失函数最小？

极值点的偏导数为零

解析解 (Analytical solution)

封闭解 (Closed-form solution)

$$\frac{\partial Loss}{\partial w} = \sum_{i=1}^{n}(y_i - b - wx_i)(-x_i) = 0$$

$$\frac{\partial Loss}{\partial b} = \sum_{i=1}^{n}(y_i - b - wx_i)(-1) = 0$$

$$w = \frac{n\sum_{i=1}^{n}x_i y_i - \sum_{i=1}^{n}x_i \sum_{i=1}^{n}y_i}{n\sum_{i=1}^{n}x_i^2 - (\sum_{i=1}^{n}x_i)^2}$$

$$b = \frac{\sum_{i=1}^{n}y_i - w\sum_{i=1}^{n}x_i}{n}$$

$$w = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$b = \bar{y} - w\bar{x}$$

神经网络&深度学习
TENSORFLOW
Google

9.3 实例：解析法实现一元线性回归

中国大学MOOC

房屋销售记录

序号	面积（平方米）	销售价格（万元）	序号	面积（平方米）	销售价格（万元）
1	137.97	145.00	9	106.69	62.00
2	104.50	110.00	10	138.05	133.00
3	100.00	93.00	11	53.75	51.00
4	124.32	116.00	12	46.91	45.00
5	79.20	65.32	13	68.00	78.50
6	99.00	104.00	14	63.02	69.65
7	124.00	118.00	15	81.26	75.69
8	114.00	91.00	16	86.21	95.30

待评估的房屋面积

[128.15, 45.00, 141.43, 106.27, 99.00, 53.84, 85.36, 70.00]

- 加载样本数据：x, y
- 学习模型：计算w,b

$$w=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

$$b=\bar{y}-w\bar{x}$$

- 预测房价

$$\hat{y}=wx+b$$

- 导入库，设置字体
- 加载样本数据

In [1]:
```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt

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

In [2]:
```
x=tf.constant([137.97, 104.50, 100.00, 124.32, 79.20, 99.00, 124.00, 114.00,
               106.69, 138.05, 53.75,  46.91,  68.00, 63.02, 81.26,  86.21])
y=tf.constant([145.00, 110.00, 93.00, 116.00, 65.32, 104.00, 118.00, 91.00,
               62.00,  133.00, 51.00, 45.00,  78.50,  69.65, 75.69,  95.30])
```

学习模型——计算 w, b

$$w=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

$$b=\bar{y}-w\bar{x}$$

```
In [3]: meanX=tf.reduce_mean(x)
        meanY=tf.reduce_mean(y)

        sumXY=tf.reduce_sum((x-meanX)*(y-meanY))
        sumX=tf.reduce_sum((x-meanX)*(x-meanX))

        w=sumXY/sumX
        b= meanY-w*meanX

In [4]: print("权值w=",w.numpy(),"\n偏置值b=",b.numpy())
        print("线性模型:y=",w.numpy(),"*x+",b.numpy())

        权值w= 0.8945604
        偏置值b= 5.4108505
        线性模型:y= 0.8945604 *x+ 5.4108505
```

预测房价

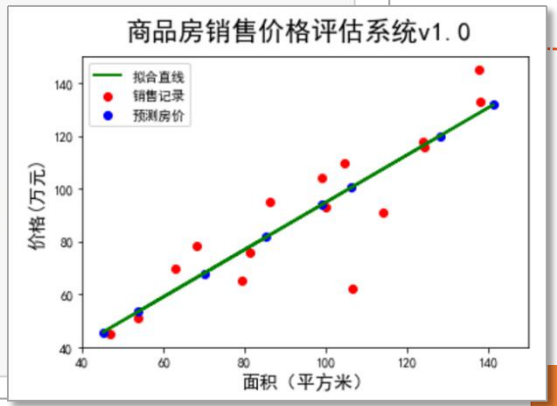
In [5]:
```
x_test=np.array([128.15, 45.00, 141.43, 106.27, 99.00, 53.84, 85.36, 70.00])
y_pred=(w*x_test+b).numpy()
```

In [6]:
```
print("面积\t估计房价")
n=len(x_test)
for i in range(n):
    print(x_test[i],"\t",round(y_pred[i],2))
```

面积	估计房价
128.15	120.05
45.0	45.67
141.43	131.93
106.27	100.48
99.0	93.97
53.84	53.57
85.36	81.77
70.0	68.03

数据和模型可视化

```
In [7]: plt.figure()

        plt.scatter(x,y,color="red",label="销售记录")
        plt.scatter(x_test,y_pred,color="blue",label="预测房价")
        plt.plot(x_test,y_pred,color="green",label="拟合直线",linewidth=2)

        plt.xlabel("面积（平方米）", fontsize=14)
        plt.ylabel("价格(万元)", fontsize=14)

        plt.xlim((40,150))
        plt.ylim((40,150))

        plt.suptitle("商品房销售价格评估系统v1.0",fontsize=20)

        plt.legend(loc="upper left")
        plt.show()
```

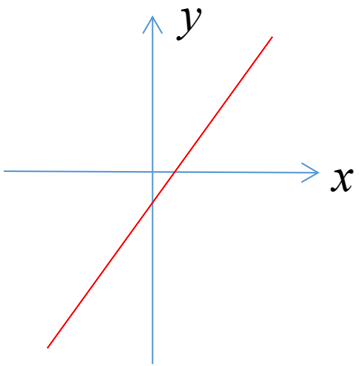



tensorflow / tensorflow
TensorFlow: Open sou
Clone this repo:
git clone https://tensor
Branches
master

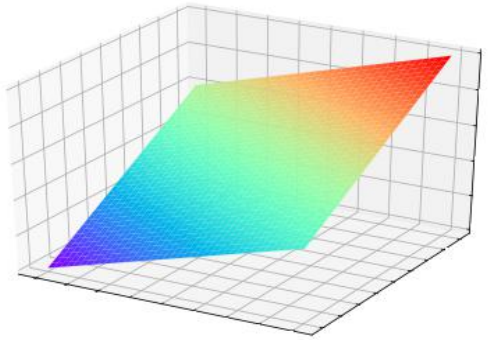
9.4 多元线性回归

- **多元回归**（Multivariate Regression）：回归分析中包括**两个或两个以上**的**自变量**
- **多元线性回归**（Multivariate Linear Regression）：因变量和自变量之间是**线性**关系

$$y = wx + b$$



$$y = w_1x_1 + w_2x_2 + b$$



$$y = w_1x_1 + \ldots + w_mx_m + b$$

超平面 (Hyperplane)：

直线在**高维空间**中的推广

模型：$\hat{y} = w_1 x^1 + \ldots + w_m x^m + b$

令$b=w_0$，$x^0=1$

$x^1, x^2, \ldots, x^m$	样本属性
$w_1, w_2, \ldots, w_m$	权值

x^1: 面积，x^2: 房间数，x^3: 楼层数

w_1=0.6，w_2=0.3，w_3=0.1

向量形式：$\hat{y} = w_0 x^0 + w_1 x^1 + \ldots + w_m x^m = W^T X$

$$W = (w_0, w_1, \ldots, w_m)^T$$

$$X = (x^0, x^1, \ldots, x^m)^T$$

n 个样本 (X_i, y_i) （$i=1, 2, \ldots, n$）

$$\hat{y}_i = W^T X_i$$

损失函数：$Loss = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - W^T X_i)^2$

损失函数：$$Loss=\sum_{i=1}^{n}(y_i-\hat{y}_i)^2=\sum_{i=1}^{n}(y_i-W^TX_i)^2$$

$$Loss=(Y-XW)^T(Y-XW)$$

$$X=(X_1,X_2,...,X_n)^T \quad Y=(y_1,y_2,...,y_n)^T$$

$$X_i=(x_i^0,x_i^1,...,x_i^m)$$

极值问题：参数向量W取何值时，*Loss*函数达到最小？

函数$f(x)$最小化时自变量x的取值 $\arg\min_x f(x)$

函数$f(x)$最大化时自变量x的取值 $\arg\max_x f(x)$

$$\arg\min_w (Y-XW)^T(Y-XW)$$

求解模型参数

$$Loss = (Y - XW)^T (Y - XW)$$

$$\frac{\partial Loss}{\partial W} = \frac{\partial((Y - XW)^T (Y - XW))}{\partial W}$$

$$= 2X^T (XW - Y)$$

$$= 2X^T XW - 2X^T Y = 0$$

$$\Rightarrow X^T XW = X^T Y$$

$$\Rightarrow W = (X^T X)^{-1} X^T Y \quad (X^T X\text{为满秩矩阵})$$

线性方程组

$$
\begin{aligned}
&w_0 + w_1 x_1^1 + \ldots + w_j x_1^j + \ldots + w_m x_1^m = \hat{y}_1 \\
&w_0 + w_1 x_2^1 + \ldots + w_j x_2^j + \ldots + w_m x_2^m = \hat{y}_2 \\
&\ldots \\
&w_0 + w_1 x_i^1 + \ldots + w_j x_i^j + \ldots + w_m x_i^m = \hat{y}_i \\
&\ldots \\
&w_0 + w_1 x_n^1 + \ldots + w_j x_n^j + \ldots + w_m x_n^m = \hat{y}_n
\end{aligned}
$$

n 个样本

m 个属性

$$
\begin{bmatrix} 1 & x_1^1 & \ldots & x_1^m \\ 1 & x_2^1 & \ldots & x_2^m \\ \ldots & \ldots & \ldots & \ldots \\ 1 & x_n^1 & \ldots & x_n^m \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \ldots \\ w_m \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \ldots \\ \hat{y}_n \end{bmatrix}
$$

矩阵形式： $XW = \hat{Y}$

损失函数：

$$Loss = (Y - \hat{Y})^2 = (Y - XW)^2$$

$$\frac{\partial Loss}{\partial W} = 2X^T(XW - Y) = 0$$

$$W = (X^T X)^{-1} X^T Y$$

n维向量：向量中的**元素个数**为n

$(1,2,3)^T$：3维向量

$(1,2,3,4,5)^T$：5维向量

$(1,2,...,n)^T$：n维向量

$X=(x^0,x^1,...,x^m)^T$ ：m+1维向量

m×n矩阵：由 **m×n 个数**排成的m行n列的数表

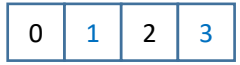
$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ \cdots & \cdots & & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

$$\begin{bmatrix} 1 & x_1^1 & \cdots & x_1^m \\ 1 & x_2^1 & \cdots & x_2^m \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_n^1 & \cdots & x_n^m \end{bmatrix}$$

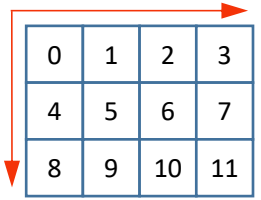
$n\times(m+1)$

多维数组 (TensorFlow/NumPy...)

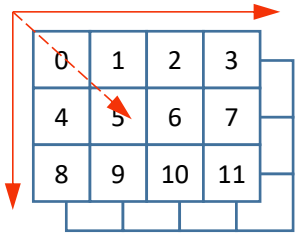
rank=1



rank=2



rank=3



神经网络&深度学习
TENSORFLOW
Google

9.5 实例：解析法实现多元线性回归

中国大学MOOC

商品房销售记录

序号	面积（平方米）	房间数	销售价格（万元）	序号	面积（平方米）	房间数	销售价格（万元）
1	137.97	3	145.00	9	106.69	2	62.00
2	104.50	2	110.00	10	138.05	3	133.00
3	100.00	2	93.00	11	53.75	1	51.00
4	124.32	3	116.00	12	46.91	1	45.00
5	79.20	1	65.32	13	68.00	1	78.50
6	99.00	2	104.00	14	63.02	1	69.65
7	124.00	3	118.00	15	81.26	2	75.69
8	114.00	2	91.00	16	86.21	2	95.30
	x_1	x_2	y		x_1	x_2	y

- 加载样本数据
- 数据处理
- 学习模型：计算W

$$W = (X^T X)^{-1} X^T Y$$

- 预测房价

■ 加载样本数据

In [1]:
```
import numpy as np
```

In [2]:
```
x1=np.array([137.97, 104.50, 100.00, 124.32, 79.20, 99.00, 124.00, 114.00,
             106.69, 138.05, 53.75, 46.91, 68.00, 63.02, 81.26, 86.21])
x2=np.array([3, 2, 2, 3, 1, 2, 3, 2, 2, 3, 1, 1, 1, 1, 2, 2])
y=np.array([145.00, 110.00, 93.00, 116.00, 65.32, 104.00, 118.00, 91.00,
            62.00, 133.00, 51.00, 45.00, 78.50, 69.65, 75.69, 95.30])
```

In [3]:
```
x1.shape, x2.shape, y.shape
```

Out[3]:
```
((16,), (16,), (16,))
```

■ 数据处理

$$Y = X W$$
$$n\times(m+1) \quad (m+1)\times 1$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & \cdots & x_1^m \\ 1 & x_2^1 & \cdots & x_2^m \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_n^1 & \cdots & x_n^m \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \cdots \\ w_m \end{bmatrix}$$

16×3

n=16, m=2

```
In [4]: x0=np.ones(len(x1))

In [5]: X=np.stack((x0,x1,x2), axis = 1)
        X

Out[5]: array([[  1.  , 137.97,   3.  ],
               [  1.  , 104.5 ,   2.  ],
               [  1.  , 100.  ,   2.  ],
               [  1.  , 124.32,   3.  ],
               [  1.  ,  79.2 ,   1.  ],
               [  1.  ,  99.  ,   2.  ],
               [  1.  , 124.  ,   3.  ],
               [  1.  , 114.  ,   2.  ],
               [  1.  , 106.69,   2.  ],
               [  1.  , 138.05,   3.  ],
               [  1.  ,  53.75,   1.  ],
               [  1.  ,  46.91,   1.  ],
               [  1.  ,  68.  ,   1.  ],
               [  1.  ,  63.02,   1.  ],
               [  1.  ,  81.26,   2.  ],
               [  1.  ,  86.21,   2.  ]])
```

(16,)
(16, 3)

■ 数据处理

$$Y = X \quad W$$
$$n\times(m+1) \quad (m+1)\times 1$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1^1 & \cdots & x_1^m \\ 1 & x_2^1 & \cdots & x_2^m \\ \cdots & \cdots & \cdots & \cdots \\ 1 & x_n^1 & \cdots & x_n^m \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ \cdots \\ w_m \end{bmatrix}$$

16×1

```
In [6]: Y= np.array(y).reshape(-1, 1)
        Y

Out[6]: array([[145.  ],
               [110.  ],
               [ 93.  ],
               [116.  ],
               [ 65.32],
               [104.  ],
               [118.  ],
               [ 91.  ],
               [ 62.  ],
               [133.  ],
               [ 51.  ],
               [ 45.  ],
               [ 78.5 ],
               [ 69.65],
               [ 75.69],
               [ 95.3 ]])
```

■ 求解模型参数

$$W = (X^T X)^{-1} X^T Y$$

功 能	函 数
矩阵相乘	np.matmul( )
矩阵转置	np.transpose( )
矩阵求逆	np.linalg.inv( )

```
In [7]: Xt=np.transpose(X)                          #计算X'
        XtX_1 = np.linalg.inv(np.matmul(Xt,X))      #计算(X'X)-1
        XtX_1_Xt= np.matmul(XtX_1,Xt)               #计算(X'X)-1X'
        W= np.matmul(XtX_1_Xt,Y)                    #W=((X'X)-1)X'Y

In [8]: W

Out[8]: array([[11.96729093],
               [ 0.53488599],
               [14.33150378]])

In [9]: W=W.reshape(-1)
        W

Out[9]: array([11.96729093,  0.53488599, 14.33150378])

In [10]: print("多元线性回归方程：")
         print( "Y=",W[1],"*x1+" ,W[2],"*x2+",W[0])

         多元线性回归方程：
         Y= 0.5348859949724747 *x1+ 14.331503777673149 *x2+ 11.967290930535732
```

■ 预测房价

In [11]:
```
print("请输入房屋面积和房间数，预测房屋销售价格：")
x1_test=float(input("商品房面积:"))
x2_test=int(input("房间数："))

y_pred = W[1]*x1_test+W[2]*x2_test+W[0]
print ("预测价格：" ,round(y_pred,2),"万元")
```

```
请输入房屋面积和房间数，预测房屋销售价格：
商品房面积:140
房间数：3
预测价格： 129.85 万元
```

```
请输入房屋面积和房间数，预测房屋销售价格：
商品房面积:140
房间数：4
预测价格： 144.18 万元
```

NumPy数组运算函数

功 能	函 数
数组堆叠	np.stack( )
改变数组形状	np.reshape( )
矩阵相乘	np.matmul( )
矩阵转置	np.transpose( )
矩阵求逆	np.linalg.inv( )

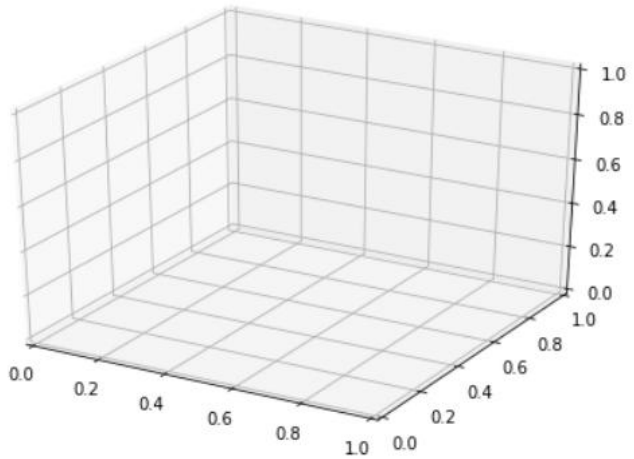

tensorflow / tensorflow
TensorFlow: Open sou
Clone this repo:
git clone https://tensor
Branches
master

9.6.1 三维数据可视化

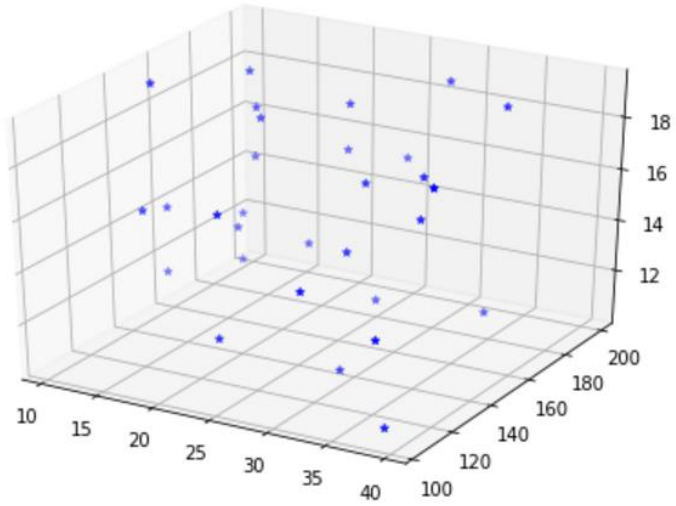
mplot3d工具集

- 绘制三维图形
- 内置于Matplotlib
- Figure对象
- Axes3d对象

```
In [1]: import matplotlib.pyplot as plt
        from mpl_toolkits.mplot3d import Axes3D

In [2]: fig=plt.figure()
        ax3d = Axes3D(fig)
        plt.show()
```



绘制散点图——随机点



In [3]:

```
import numpy as np
```

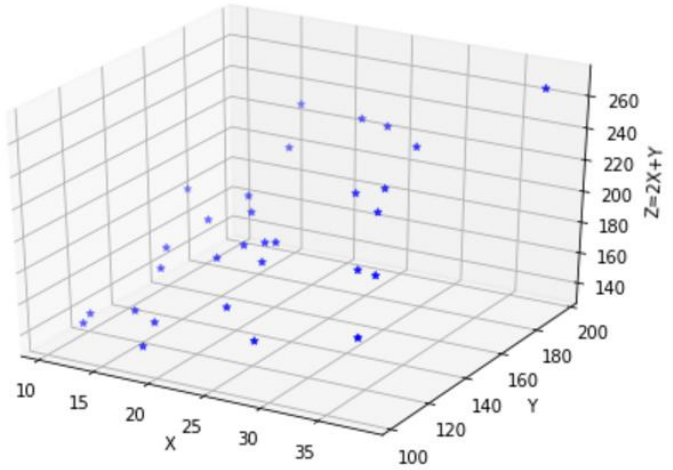
In [4]:

```
x=np.random.uniform(10,40,30)
y=np.random.uniform(100,200,30)
z=np.random.uniform(10,20,30)
```

In [5]:

```
fig = plt.figure()
ax3d = Axes3D(fig)
ax3d.scatter(x,y,z,c='b',marker="*")
plt.show()
```

绘制散点图——z=2x+y



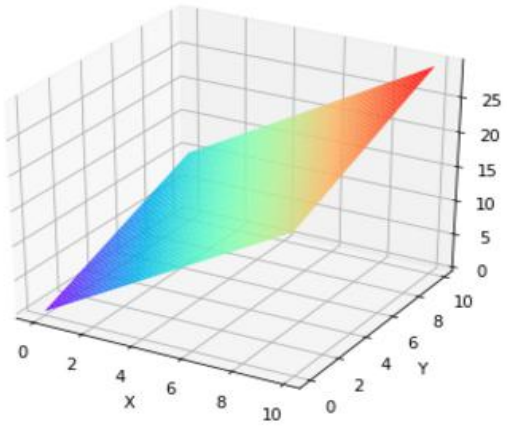
In [6]:

```
x=np.random.uniform(10,40,30)
y=np.random.uniform(100,200,30)
z=2*x+y
```

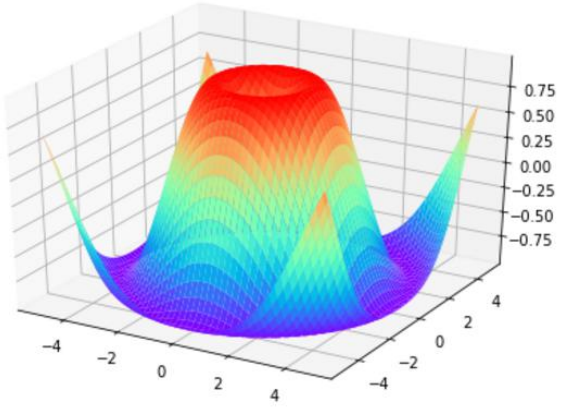
In [7]:

```
fig = plt.figure()
ax3d = Axes3D(fig)

ax3d.scatter(x,y,z,c='b',marker="*")

ax3d.set_xlabel('X')
ax3d.set_ylabel('Y')
ax3d.set_zlabel('Z=2X+Y')

plt.show()
```

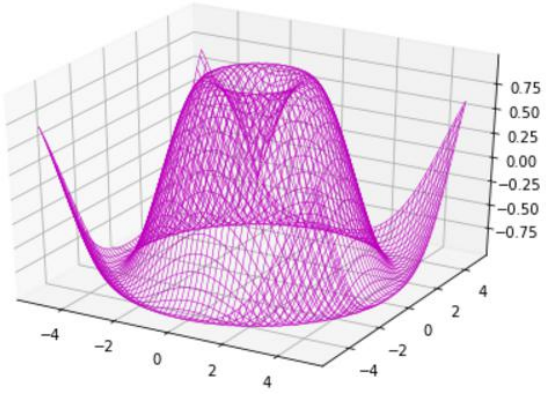
平面图


25
20
15
10
5
0
0
2
4
6
8
10
X
0
2
4
6
8
10
Y

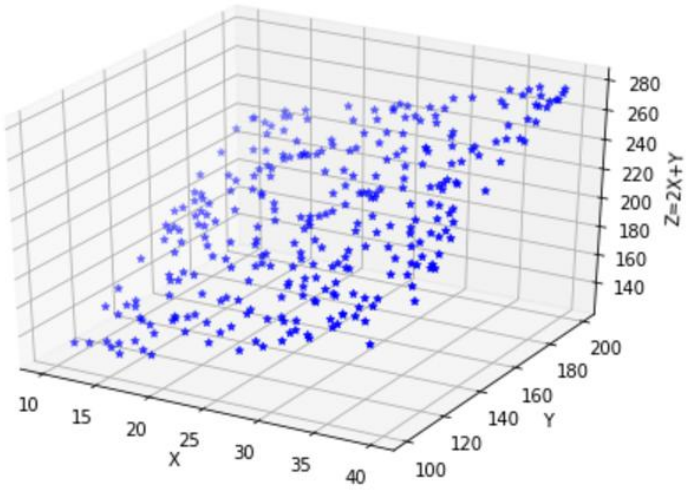
曲面图


0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−4
−2
0
2
4
−4
−2
0
2
4

线框图


0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−4
−2
0
2
4
−4
−2
0
2
4

绘制散点图——z=2x+y



In [8]:

```
x=np.random.uniform(10,40,300)
y=np.random.uniform(100,200,300)
z=2*x+y
```

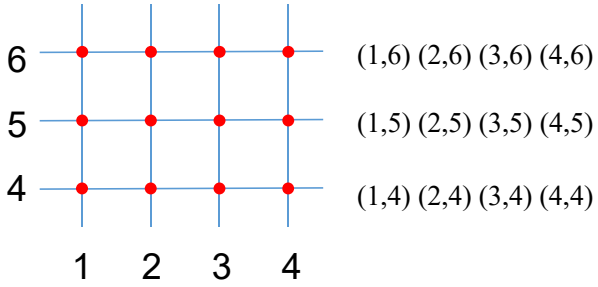
In [9]:

```
fig = plt.figure()
ax3d = Axes3D(fig)

ax3d.scatter(x,y,z,c='b',marker="*")

ax3d.set_xlabel('X')
ax3d.set_ylabel('Y')
ax3d.set_zlabel('Z=2X+Y')

plt.show()
```

网格点坐标矩阵

X=(1, 2, 3, 4)

Y=(4, 5, 6)



np.meshgrid()：生成网格点坐标矩阵

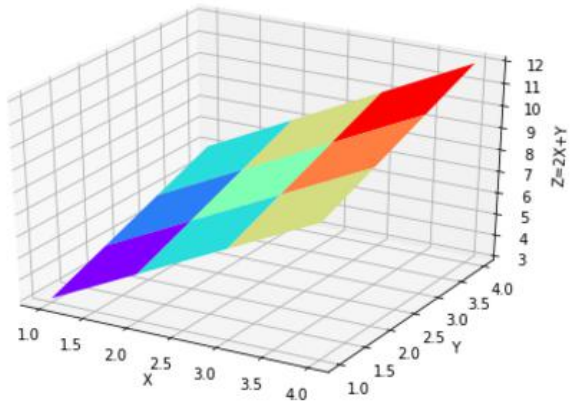
```
In [10]: x=[1, 2, 3, 4]
         y=[4, 5, 6]
         X, Y=np.meshgrid(x, y)

In [11]: X
Out[11]: array([[1, 2, 3, 4],
                [1, 2, 3, 4],
                [1, 2, 3, 4]])

In [12]: Y
Out[12]: array([[4, 4, 4, 4],
                [5, 5, 5, 5],
                [6, 6, 6, 6]])
```

绘制平面图——z=2x+y

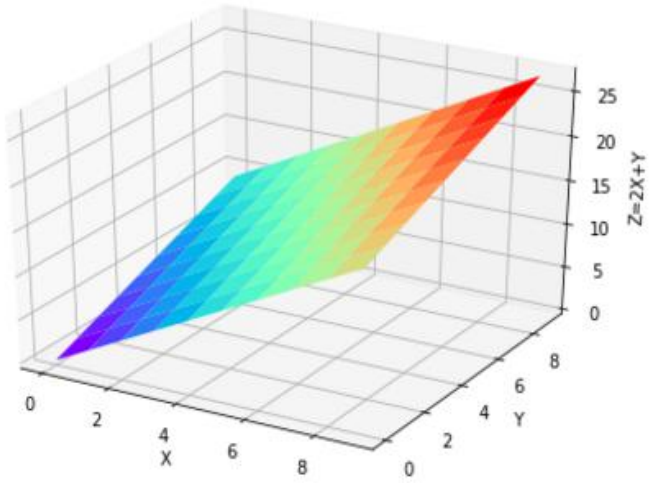
```
In [13]: x=np.arange(1,5)
         y=np.arange(1,5)
         X,Y=np.meshgrid(x,y)

In [14]: X.shape,Y.shape
Out[14]: ((4, 4), (4, 4))

In [15]: Z=2*X+Y

In [16]: Z.shape
Out[16]: (4, 4)
```

plot_surface()：绘制平面/曲面图

```
In [17]: fig = plt.figure()
         ax3d = Axes3D(fig)

         ax3d.plot_surface(X,Y,Z,cmap="rainbow")

         ax3d.set_xlabel('X')
         ax3d.set_ylabel('Y')
         ax3d.set_zlabel('Z=2X+Y')
         plt.show()
```



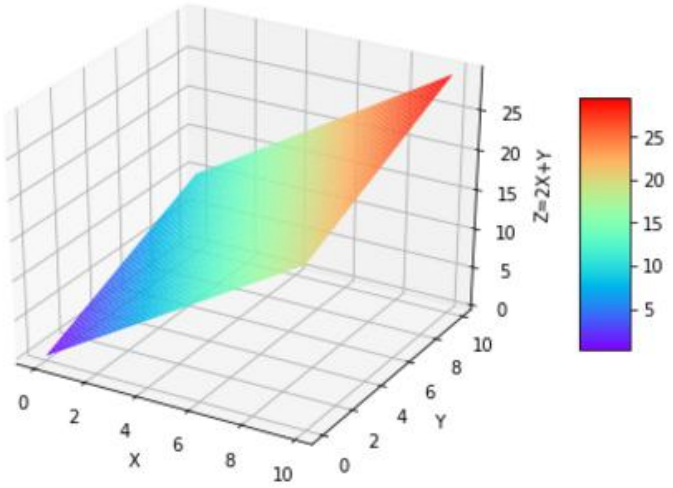
绘制平面图——z=2x+y



In [18]:
```
x=np.arange(0,10)
y=np.arange(0,10)
X,Y=np.meshgrid(x,y)
Z=2*X+Y
```

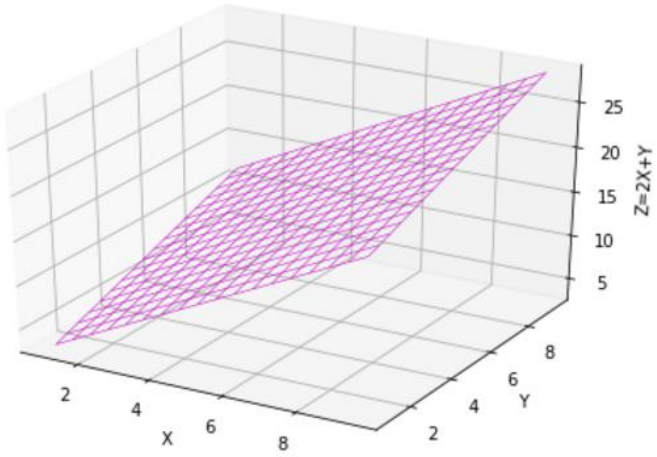
In [19]:
```
fig = plt.figure()
ax3d = Axes3D(fig)

ax3d.plot_surface(X,Y,Z,cmap="rainbow")

ax3d.set_xlabel('X')
ax3d.set_ylabel('Y')
ax3d.set_zlabel('Z=2X+Y')
plt.show()
```

绘制平面图——z=2x+y




```
In [20]: x=np.arange(0,10,0.1)
         y=np.arange(0,10,0.1)
         X,Y=np.meshgrid(x,y)
         Z=2*X+Y

In [21]: Z.shape

Out[21]: (100, 100)

In [22]: fig = plt.figure()
         ax3d = Axes3D(fig)

         surf=ax3d.plot_surface(X,Y,Z,cmap="rainbow")
         fig.colorbar(surf, shrink=0.5, aspect=5)

         ax3d.set_xlabel('X')
         ax3d.set_ylabel('Y')
         ax3d.set_zlabel('Z=2X+Y')
         plt.show()
```

绘制线框图——z=2x+y



plot_wireframe()：绘制线框图

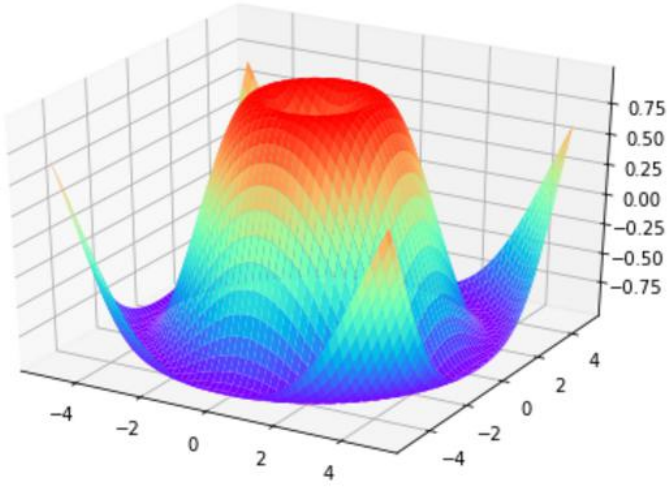
In [23]:

```
x=np.arange(1,10,0.5)
y=np.arange(1,10,0.5)
X,Y=np.meshgrid(x,y)
Z=2*X+Y
```

In [24]:

```
fig = plt.figure()
ax3d = Axes3D(fig)

ax3d.plot_wireframe(X,Y,Z,color="m",linewidth=0.5)

ax3d.set_xlabel('X')
ax3d.set_ylabel('Y')
ax3d.set_zlabel('Z=2X+Y')
plt.show()
```

绘制曲面图——$z=\sin\sqrt{x^2+y^2}$



In [25]:

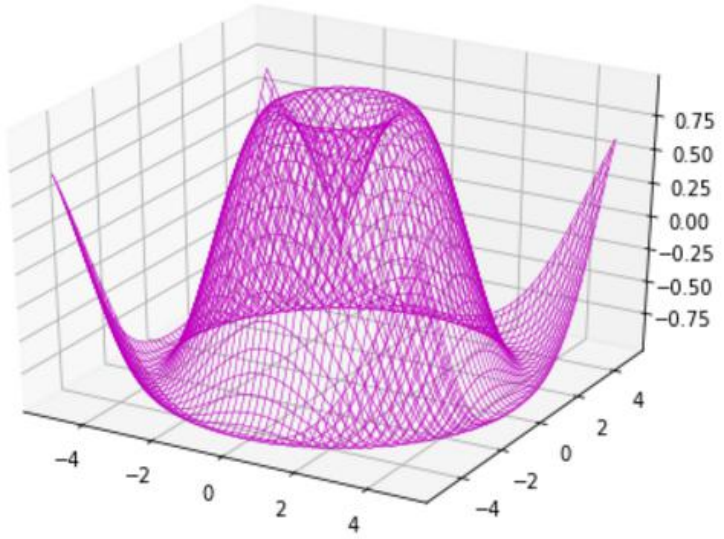
```
x=np.arange(-5,5,0.1)
y=np.arange(-5,5,0.1)
X,Y=np.meshgrid(x,y)
Z=np.sin(np.sqrt(X** 2+Y** 2))
```

In [26]:

```
fig = plt.figure()
ax3d = Axes3D(fig)

ax3d.plot_surface(X,Y,Z,cmap="rainbow")

plt.show()
```

绘制曲面的线框图

```
In [27]: fig = plt.figure()
         ax3d = Axes3D(fig)

         ax3d.plot_wireframe(X,Y,Z,color="m",linewidth=0.5)

         plt.show()
```

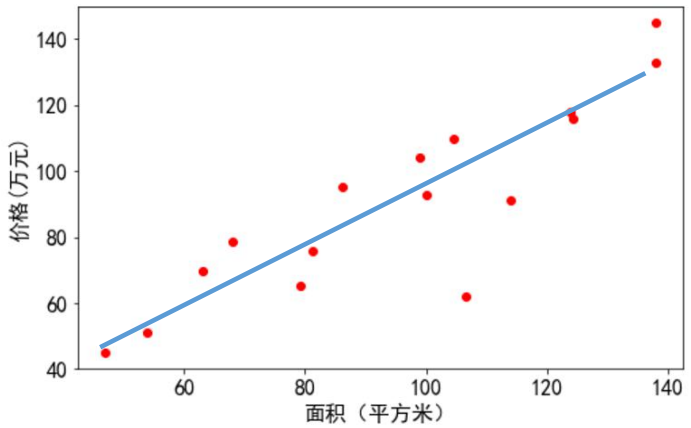


神经网络&深度学习
TENSORFLOW
Google

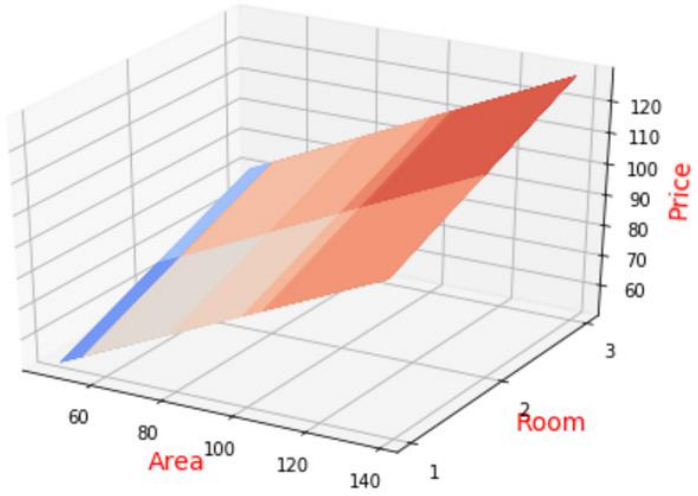
9.6.2 实例：线性回归模型可视化

中国大学MOOC

一元线性回归



二元线性回归



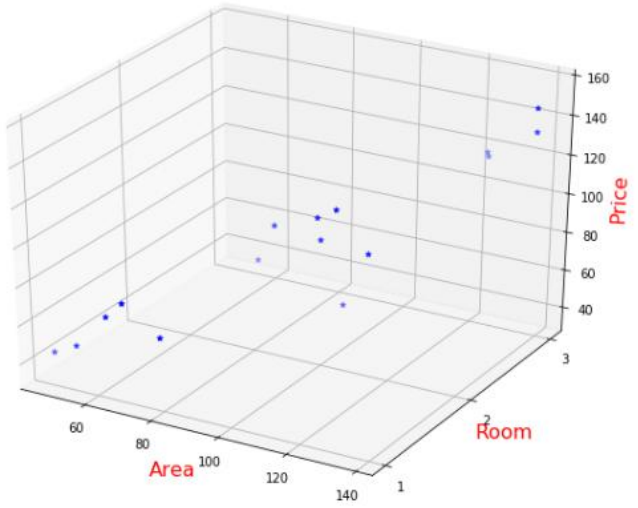
加载数据

In [1]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
```

In [2]:
```
x1=np.array([137.97,104.50,100.00,124.32,79.20,99.00,124.00,114.00,
             106.69,138.05,53.75,46.91,68.00,63.02,81.26,86.21])
x2=np.array([3,2,2,3,1,2,3,2,2,3,1,1,1,1,2,2])
y=np.array([145.00,110.00,93.00,116.00,65.32,104.00,118.00,91.00,
            62.00,133.00,51.00,45.00,78.50,69.65,75.69,95.30])
```

In [3]:
```
W=np.array([11.96729093,  0.53488599,  14.33150378])
y_pred=W[1]*x1+W[2]*x2+W[0]
```

绘制散点图

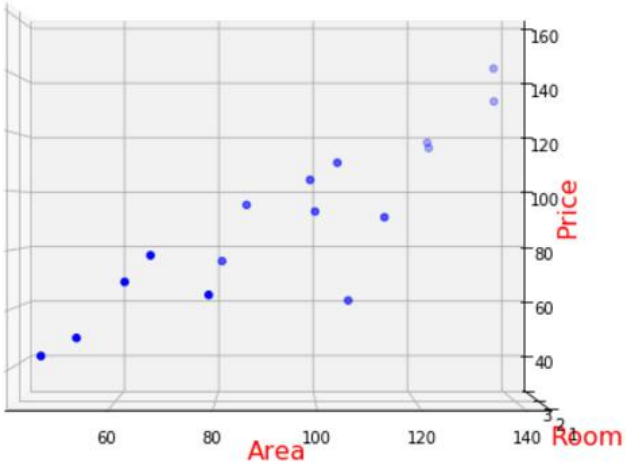


In [4]:

```
fig= plt.figure(figsize=(8,6))
ax3d = Axes3D(fig)

ax3d.scatter(x1,x2,y,color="b",marker="*")

ax3d.set_xlabel('Area',color='r',fontsize=16)
ax3d.set_ylabel('Room',color='r',fontsize=16)
ax3d.set_zlabel('Price',color='r',fontsize=16)
ax3d.set_yticks([1,2,3])
ax3d.set_zlim3d(30,160)

plt.show()
```

改变视角

view_init(elev,azim)

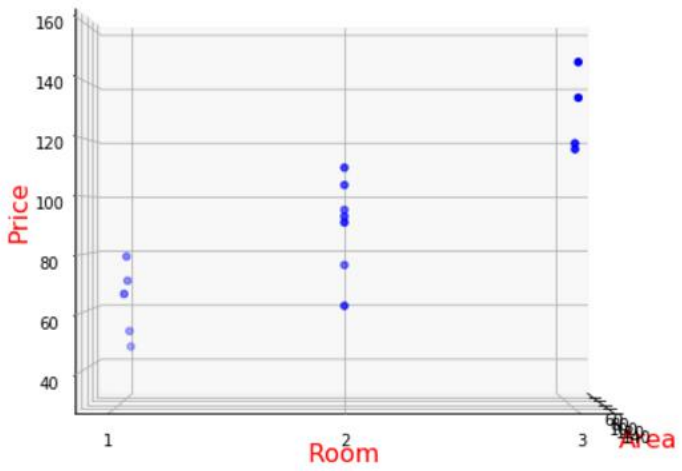


In [5]:

```
fig= plt.figure(figsize=(8,6))
ax3d = Axes3D(fig)
ax3d.view_init(elev=0,azim=-90)

ax3d.scatter(x1,x2,y,color='b')

ax3d.set_xlabel('Area',color='r',fontsize=16)
ax3d.set_ylabel('Room',color='r',fontsize=16)
ax3d.set_zlabel('Price',color='r',fontsize=16)
ax3d.set_yticks([1,2,3])
ax3d.set_zlim3d(30,160)

plt.show()
```

改变视角

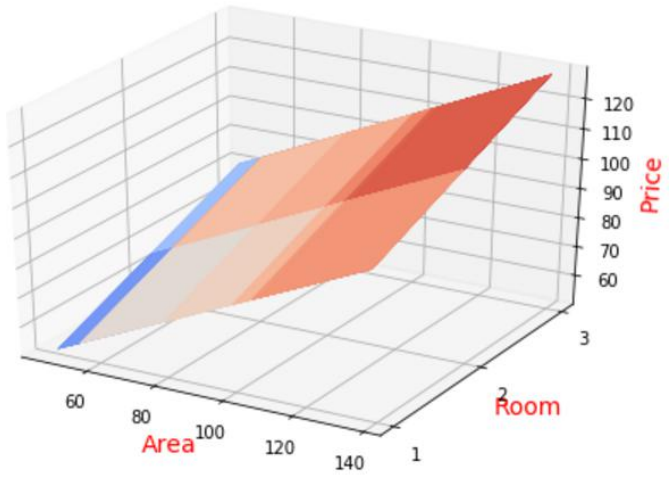
view_init(elev,azim)



In [6]:

```
fig= plt.figure(figsize=(8,6))
ax3d = Axes3D(fig)
ax3d.view_init(elev=0,azim=0)

ax3d.scatter(x1,x2,y,color='b')

ax3d.set_xlabel('Area',color='r',fontsize=16)
ax3d.set_ylabel('Room',color='r',fontsize=16)
ax3d.set_zlabel('Price',color='r',fontsize=16)
ax3d.set_yticks([1,2,3])
ax3d.set_zlim3d(30,160)

plt.show()
```

绘制平面图



In [7]:

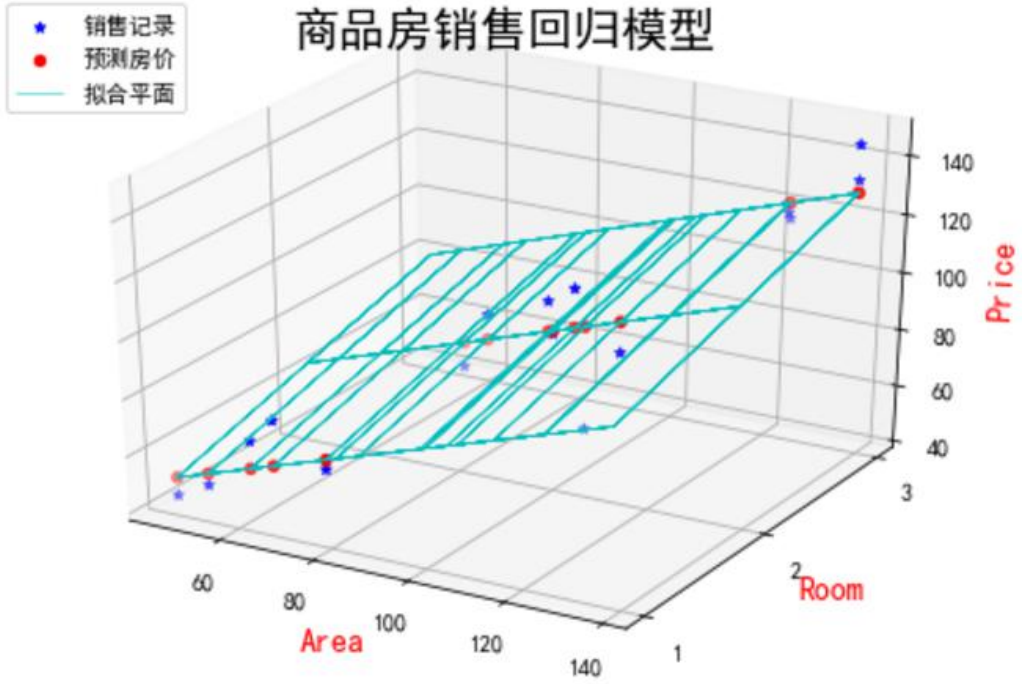
```
X1,X2=np.meshgrid(x1,x2)
Y_PRED=W[0]+W[1]*X1+W[2]*X2
```

In [8]:

```
fig= plt.figure()
ax3d = Axes3D(fig)

ax3d.plot_surface(X1,X2,Y_PRED,cmap="coolwarm")

ax3d.set_xlabel('Area',color='r',fontsize=14)
ax3d.set_ylabel('Room',color='r',fontsize=14)
ax3d.set_zlabel('Price',color='r',fontsize=14)
ax3d.set_yticks([1,2,3])

plt.show()
```

绘制散点图和线框图

In [9]:
```
plt.rcParams['font.sans-serif'] = ['SimHei']

fig= plt.figure()
ax3d = Axes3D(fig)

ax3d.scatter(x1,x2,y,color='b',marker="*",label="销售记录")
ax3d.scatter(x1,x2,y_pred,color='r',label="预测房价")
ax3d.plot_wireframe(X1,X2,Y_PRED,color="c",linewidth=0.5,label="拟合平面")

ax3d.set_xlabel('Area',color='r',fontsize=14)
ax3d.set_ylabel('Room',color='r',fontsize=14)
ax3d.set_zlabel('Price',color='r',fontsize=14)
ax3d.set_yticks([1,2,3])

plt.suptitle("商品房销售回归模型", fontsize=20)
plt.legend(loc="upper left")
plt.show()
```


销售记录
预测房价
拟合平面
商品房销售回归模型
Price
140
120
100
80
60
40
Area
60
80
100
120
140
Room
1
2
3