神经网络&深度学习 TENSORFLOW Google


# 10.2 梯度下降法求解线性回归

中国大学MOOC


# 10.2.1 梯度下降法求解线性回归

## 梯度下降法求解**一元线性回归**问题

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

**极值问题**: $\arg\min_{w,b} Loss(w,b)$

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

$$= \underline{Aw^2 + Bb^2 + Cwb + Dw + Eb + F}$$

凸函数

$$w^{(k+1)} = w^{(k)} - \eta \boxed{\frac{\partial Loss(w,b)}{\partial w}}$$

$$b^{(k+1)} = b^{(k)} - \eta \boxed{\frac{\partial Loss(w,b)}{\partial b}}$$

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

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

$$w^{(k+1)} = w^{(k)} - \eta\sum_{i=1}^{n} x_i(wx_i + b - y_i)$$

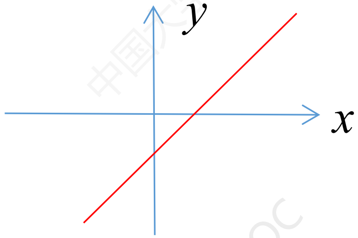
$$b^{(k+1)} = b^{(k)} - \eta\sum_{i=1}^{n}(wx_i + b - y_i)$$

# 10.2.1 梯度下降法求解线性回归



## 一元线性回归问题

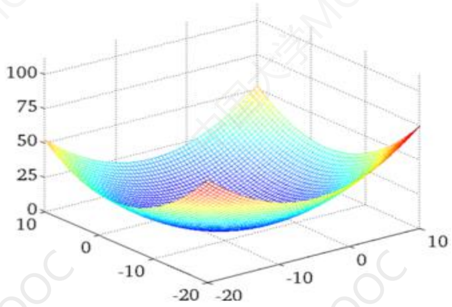
$$y = wx + b$$




## 二元求极值问题

$$\underset{w,b}{\arg\min}\, Loss(w,b)$$

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




$$w^{(k+1)} = w^{(k)} - \eta \frac{\partial Loss(w,b)}{\partial w}$$

$$b^{(k+1)} = b^{(k)} - \eta \frac{\partial Loss(w,b)}{\partial b}$$



西安科技大学
计算机科学与技术学院

## 平方损失函数

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

模型参数更新算法：

$$w^{(k+1)} = w^{(k)} - \eta\sum_{i=1}^{n}x_i(wx_i + b - y_i)$$

$$b^{(k+1)} = b^{(k)} - \eta\sum_{i=1}^{n}(wx_i + b - y_i)$$

## 均方差损失函数

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

模型参数更新算法：

$$w^{(k+1)} = w^{(k)} - \frac{\eta}{n}\sum_{i=1}^{n}x_i(wx_i + b - y_i)$$

$$b^{(k+1)} = b^{(k)} - \frac{\eta}{n}\sum_{i=1}^{n}(wx_i + b - y_i)$$

## 梯度下降法求解**多元线性回归**问题

$$\hat{Y} = XW$$

$$Loss = \frac{1}{2}(Y - \hat{Y})^2 = \frac{1}{2}(Y - XW)^2$$

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

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

权值更新算法：

$$W^{(k+1)} = W^{(k)} - \eta \frac{\partial Loss(W)}{\partial W}$$

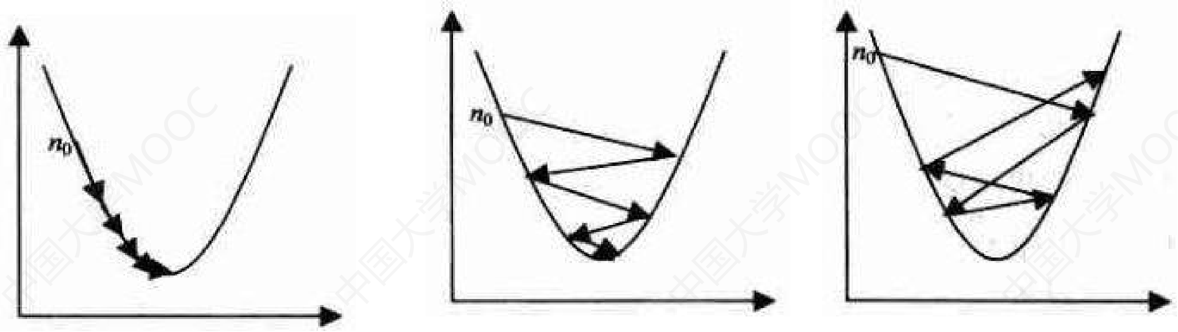
$$\frac{\partial Loss}{\partial W} = X^T(XW - Y)$$

$$\boxed{W^{(k+1)} = W^{(k)} - \eta X^T(XW - Y)}$$

## 学习率

$$w_i^{(k+1)} = w_i^{(k)} - \boxed{\eta} \frac{\partial Loss(\boldsymbol{w})}{\partial w_i}$$

对于**凸函数**，只要学习率设置的足够小，可以保证**一定收敛**




**超参数**：在开始**学习之前**设置，不是通过训练得到的


# 10.2.2 梯度下降法求解一元线性回归

## ——NumPy实现

## 梯度下降法求解一元线性回归

平方损失函数：

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

模型参数更新算法：

$$w^{(k+1)} = w^{(k)} - \eta\sum_{i=1}^{n}x_i(wx_i + b - y_i)$$

$$b^{(k+1)} = b^{(k)} - \eta\sum_{i=1}^{n}(wx_i + b - y_i)$$

均方差损失函数：

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

模型参数更新算法：

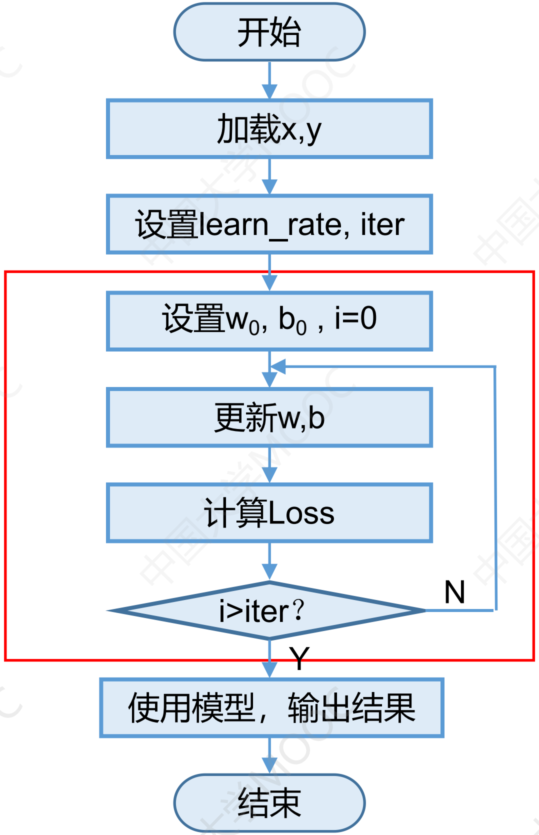
$$w^{(k+1)} = w^{(k)} - \frac{\eta}{n}\sum_{i=1}^{n}x_i(wx_i + b - y_i)$$

$$b^{(k+1)} = b^{(k)} - \frac{\eta}{n}\sum_{i=1}^{n}(wx_i + b - y_i)$$

## 商品房销售记录

| 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 |

- 加载样本数据 x, y
- 设置超参数 学习率，迭代次数
- 设置模型参数初值 $w_0, b_0$
- 训练模型 w, b
- 结果可视化



## 加载数据

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

```
In [2]: x = 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])
        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])
```

## 设置超参数

```
In [3]: learn_rate=0.00001
        iter=100

        display_step=10
```

## 设置模型参数初值

```
In [4]: np.random.seed(612)
        w = np.random.randn()
        b = np.random.randn()
```

## 训练模型

```
In [5]: mse=[]

        for i in range(0, iter+1):

            dL_dw=np.mean(x*(w*x+b-y))
            dL_db=np.mean(w*x+b-y)

            w=w-learn_rate*dL_dw
            b=b-learn_rate*dL_db

            pred=w*x+b
            Loss= np.mean(np.square(y-pred))/2
            mse.append(Loss)

            if i % display_step == 0:
                print("i: %i, Loss:%f, w: %f, b: %f" % (i,mse[i], w, b))
```

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

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

$$w^{(k+1)} = w^{(k)} - \eta\frac{\partial Loss}{\partial w}$$

$$b^{(k+1)} = b^{(k)} - \eta\frac{\partial Loss}{\partial b}$$

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

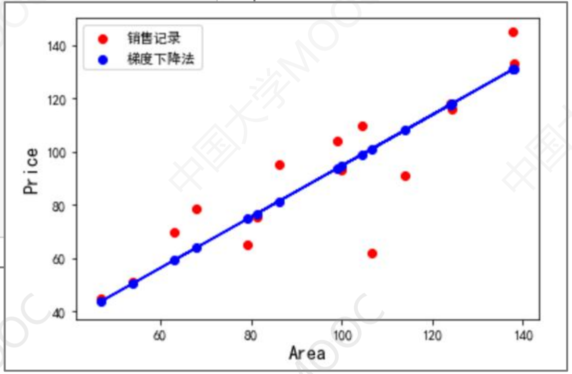
## 训练模型

```
In [5]: mse=[]

        for i in range(0, iter+1):

            dL_dw=np.mean(x*(w*x+b-y))
            dL_db=np.mean(w*x+b-y)

            w=w-learn_rate*dL_dw
            b=b-learn_rate*dL_db

            pred=w*x+b
            Loss= np.mean(np.square(y-pred))/2
            mse.append(Loss)

            if i % display_step == 0:
                print("i: %i, Loss:%f, w: %f, b: %f" % (i,mse[i], w, b))
```

```
i: 0, Loss:3874.243711, w: 0.082565, b: -1.161967
i: 10, Loss:562.072704, w: 0.648552, b: -1.156446
i: 20, Loss:148.244254, w: 0.848612, b: -1.154462
i: 30, Loss:96.539782, w: 0.919327, b: -1.153728
i: 40, Loss:90.079712, w: 0.944323, b: -1.153435
i: 50, Loss:89.272557, w: 0.953157, b: -1.153299
i: 60, Loss:89.171687, w: 0.956280, b: -1.153217
i: 70, Loss:89.159061, w: 0.957383, b: -1.153156
i: 80, Loss:89.157460, w: 0.957773, b: -1.153101
i: 90, Loss:89.157238, w: 0.957910, b: -1.153048
i: 100, Loss:89.157187, w: 0.957959, b: -1.152997
```

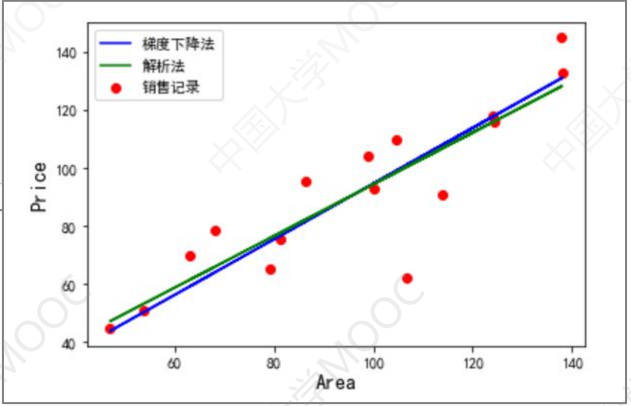
## ■ 结果可视化——数据和模型

```
In [6]: plt.rcParams['font.sans-serif'] = ['SimHei']

        plt.figure()

        plt.scatter(x,y,color="red",label="销售记录")
        plt.scatter(x,pred,color="blue",label="梯度下降法")
        plt.plot(x,pred,color="blue")

        plt.xlabel("Area",fontsize=14)
        plt.ylabel("Price",fontsize=14)

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



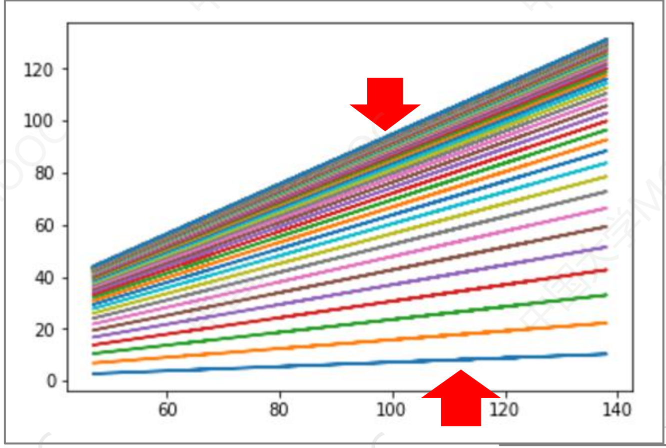
**解析解**：w=0.8945604, b=5.4108505

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

        plt.scatter(x,y,color="red",label="销售记录")
        plt.plot(x,pred,color="blue",label="梯度下降法")
        plt.plot(x,0.89*x+5.41,color="green",label="解析法")

        plt.xlabel("Area",fontsize=14)
        plt.ylabel("Price",fontsize=14)

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



# 10.2.2 梯度下降法求解一元线性回归——NumPy实现

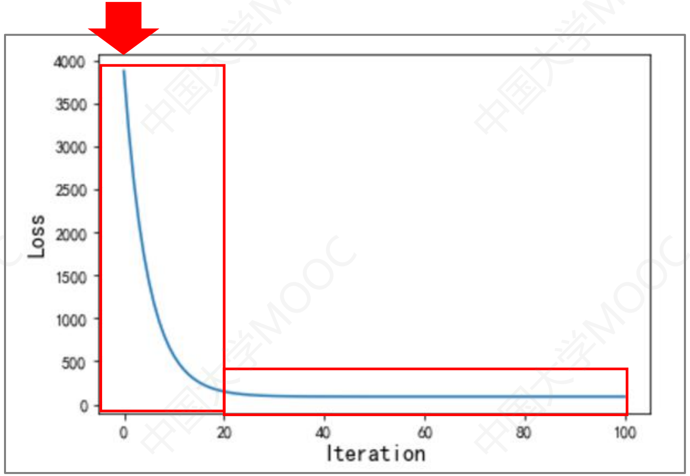





In [5]:

```
mse=[]

for i in range(0, iter+1):

    dL_dw=np.mean(x*(w*x+b-y))
    dL_db=np.mean(w*x+b-y)

    w=w-learn_rate*dL_dw
    b=b-learn_rate*dL_db

    pred=w*x+b
    Loss= np.mean(np.square(y-pred))/2
    mse.append(Loss)

    plt.plot(x,pred)

    if i % display_step == 0:
        print("i: %i, Loss:%f, w: %f, b: %f" % (i,mse[i], w, b))
```

## ■ 结果可视化——损失变化

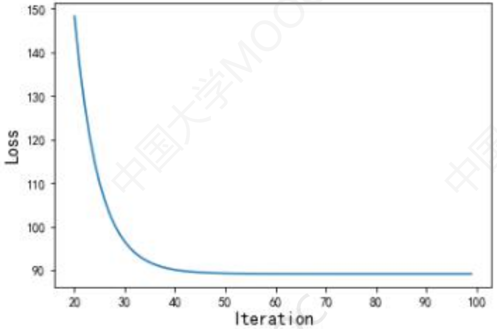



In [8]:

```
plt.figure()

plt.plot(mse)

plt.xlabel("Iteration",fontsize=14)
plt.ylabel("Loss",fontsize=14)

plt.show()
```

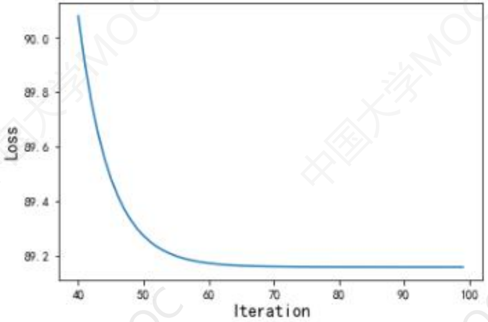
In [9]:

```
plt.figure()

plt.plot(range(20, 100), mse[20:100])

plt.xlabel("Iteration", fontsize=14)
plt.ylabel("Loss", fontsize=14)

plt.show()
```

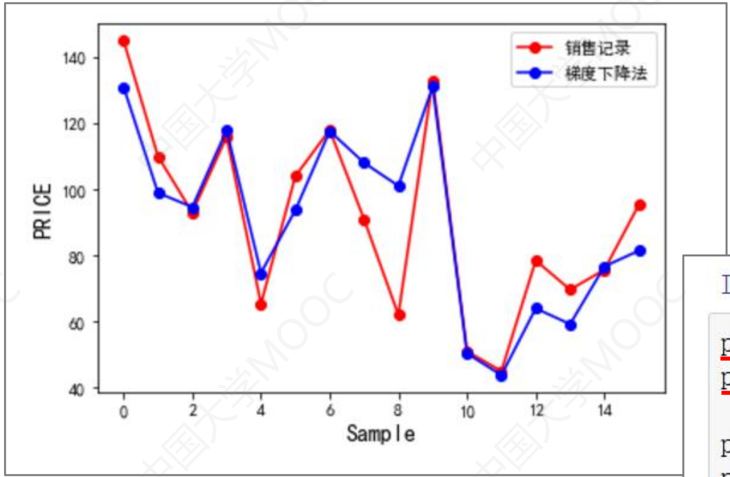



In [10]:

```
plt.figure()

plt.plot(range(40, 100), mse[40:100])

plt.xlabel("Iteration", fontsize=14)
plt.ylabel("Loss", fontsize=14)

plt.show()
```



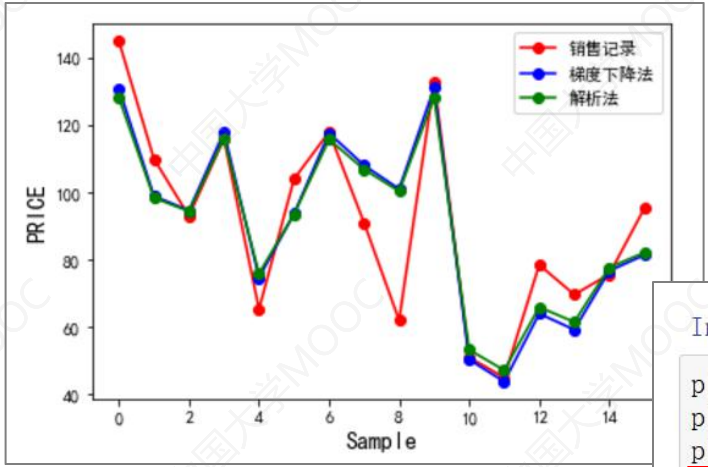
## ■ 结果可视化——估计值 & 标签值




In [11]:

```
plt.plot(y,color="red",marker="o",label="销售记录")
plt.plot(pred,color="blue",marker="o",label="梯度下降法")

plt.legend()
plt.xlabel("Sample",fontsize=14)
plt.ylabel("PRICE",fontsize=14)
plt.show()
```

# 10.2.2 梯度下降法求解一元线性回归——NumPy实现



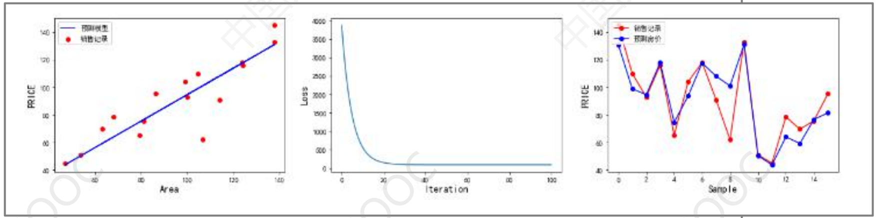

In [12]:

```
plt.plot(y,color="red",marker="o",label="销售记录")
plt.plot(pred,color="blue",marker="o",label="梯度下降法")
plt.plot(0.89*x+5.41,color="green",marker="o",label="解析法")

plt.legend()
plt.xlabel("Sample",fontsize=14)
plt.ylabel("PRICE",fontsize=14)
plt.show()
```

# 10.2.2 梯度下降法求解一元线性回归——NumPy实现



In [13]:
```python
plt.rcParams['font.sans-serif'] = ['SimHei']

plt.figure(figsize=(20,4))

plt.subplot(1,3,1)
plt.scatter(x,y,color="red",label="销售记录")
plt.plot(x,pred,color="blue",label="预测模型")
plt.xlabel("Area",fontsize=14)
plt.ylabel("PRICE",fontsize=14)
plt.legend(loc="upper left")

plt.subplot(1,3,2)
plt.plot(mse)
plt.xlabel("Iteration",fontsize=14)
plt.ylabel("Loss",fontsize=14)

plt.subplot(1,3,3)
plt.plot(y,color="red",marker="o",label="销售记录")
plt.plot(pred,color="blue",marker="o",label="预测房价")
plt.legend()
plt.xlabel("Sample",fontsize=14)
plt.ylabel("PRICE",fontsize=14)
plt.legend(loc="upper left")

plt.show()
```




# 10.2.3 梯度下降法求解多元线性回归
## ——NumPy实现

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 |

- **归一化 / 标准化**：将数据的**值**限制在**一定的范围**之内

使所有属性处于同一个**范围**、同一个**数量级**下

更快**收敛**到最优解

提高学习器的**精度**

**线性归一化，标准差归一化，非线性映射归一化**

- **线性归一化**：对原始数据的线性变换

$$x^* = \frac{x - \min}{\max - \min}$$

**等比例**缩放

所有的数据都被映射到[0,1]之间

- **标准差归一化**：将数据集归一化为**均值为0**，**方差为1**的标准正态分布

$$x^* = \frac{x - \mu}{\sigma}$$

- **非线性映射归一化**：对原始数据的**非线性**变换

指数、对数、正切

## □ 线性归一化

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

In [2]:
```
area=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])
room=np.array([3, 2, 2, 3, 1, 2, 3, 2, 2, 3, 1, 1, 1, 1, 2, 2])
```

In [3]:
```
x1=(area -area.min())/(area.max()-area.min())
x2=(room -room.min())/(room.max()-room.min())
```

In [4]:
```
x1, x2
```

Out[4]:
```
(array([0.99912223, 0.63188501, 0.58251042, 0.84935264, 0.3542901 ,
        0.57153829, 0.84584156, 0.73612025, 0.65591398, 1.        ,
        0.07504937, 0.        , 0.23140224, 0.17676103, 0.37689269,
        0.43120474]),
 array([1. , 0.5, 0.5, 1. , 0. , 0.5, 1. , 0.5, 0.5, 1. , 0. , 0. , 0. ,
        0. , 0.5, 0.5]))
```

## 使用**梯度下降法**求解**多元线性回归**

- 加载样本数据 area, room, price
- 数据处理 归一化，X, Y
- 设置超参数 学习率，迭代次数
- 设置模型参数初值 $W_0$ $(w_0,w_1,w_2)$
- 训练模型 W

$$W^{(k+1)} = W^{(k)} - \eta X^T (XW - Y)$$

- 结果可视化

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

$$W^{(k+1)} = W^{(k)} - \eta \frac{\partial Loss(W)}{\partial W}$$

## 加载样本数据

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

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

In [2]: area=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])
        room=np.array([3,2,2,3,1,2,3,2,2,3,1,1,1,1,2,2])
        price = 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])
        num = len(area)
```

## 数据处理

```
In [3]: x0 = np.ones(num)

        x1=(area -area.min())/(area.max()-area.min())
        x2=(room -room.min())/(room.max()-room.min())

        X =np.stack((x0,x1,x2), axis = 1)
        Y= price.reshape(-1,1)

In [4]: X.shape,Y.shape

Out[4]: ((16, 3), (16, 1))
```

## 设置超参数

```
In [5]: learn_rate=0.001
        iter=500

        display_step=50
```

## 设置模型参数初识值

```
In [6]: np.random.seed(612)
        W = np.random.randn(3,1)
```

## 训练模型

In [7]:
```
mse=[]

for i in range(0, iter+1):

    dL_dW= np.matmul(np.transpose(X),np.matmul(X,W)-Y)
    W=W-learn_rate*dL_dW

    PRED=np.matmul(X,W)
    Loss= np.mean(np.square(Y-PRED))/2
    mse.append(Loss)

    if i % display_step == 0:
        print("i: %i, Loss:%f" % (i,mse[i]))
```

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

$$W^{(k+1)}=W^{(k)}-\eta\frac{\partial Loss(W)}{\partial W}$$

X W

$(16,3)$ $(3,1) \rightarrow (16,1)$

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

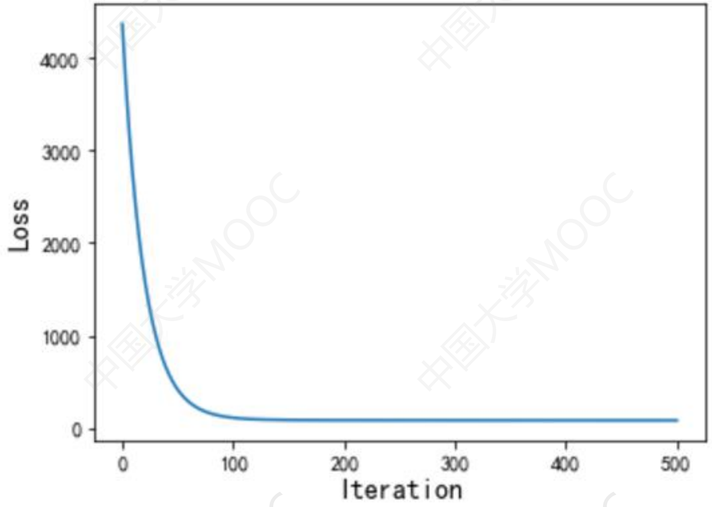
```
i: 0,   Loss:4368.213908
i: 50,  Loss:413.185263
i: 100, Loss:108.845176
i: 150, Loss:84.920786
i: 200, Loss:82.638199
i: 250, Loss:82.107310
i: 300, Loss:81.782545
i: 350, Loss:81.530512
i: 400, Loss:81.329266
i: 450, Loss:81.167833
i: 500, Loss:81.037990
```

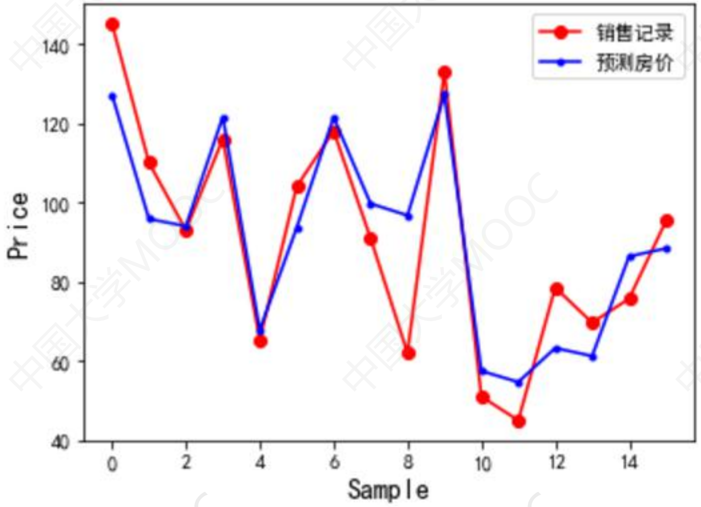
## 结果可视化

In [8]:
```
plt.figure(figsize=(12,4))

plt.subplot(1,2,1)
plt.plot(mse)
plt.xlabel("Iteration",fontsize=14)
plt.ylabel("Loss",fontsize=14)

plt.subplot(1,2,2)
PRED=PRED.reshape(-1)
plt.plot(price,color="red",marker="o",label="销售记录")
plt.plot(PRED,color="blue",marker=".",label="预测房价")
plt.xlabel("Sample",fontsize=14)
plt.ylabel("Price",fontsize=14)

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

# 10.2.3 梯度下降法求解多元线性回归——NumPy实现








西安科技大学
计算机科学与技术学院