线性回归和梯度下降算法

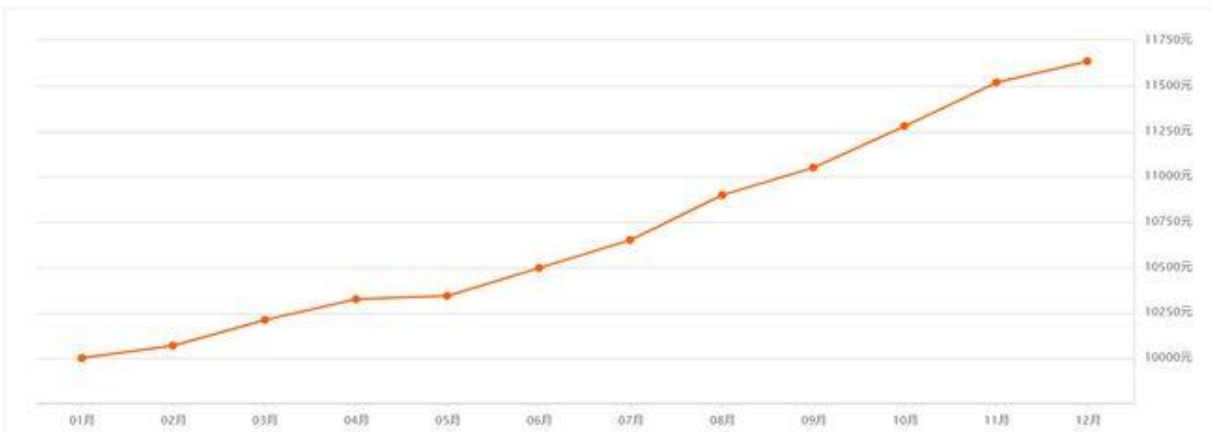
雍宾宾

yongbb@lzu.edu.cn

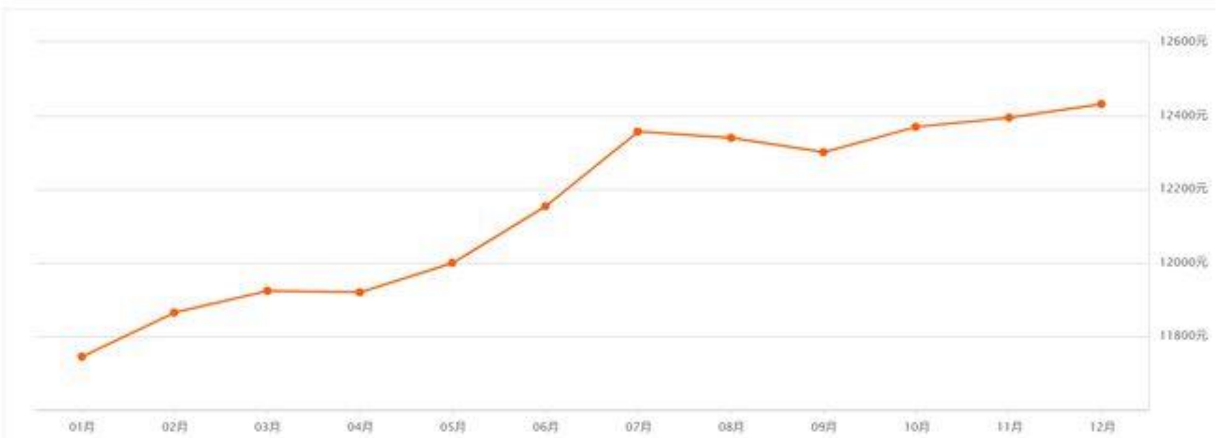
应用背景

房价

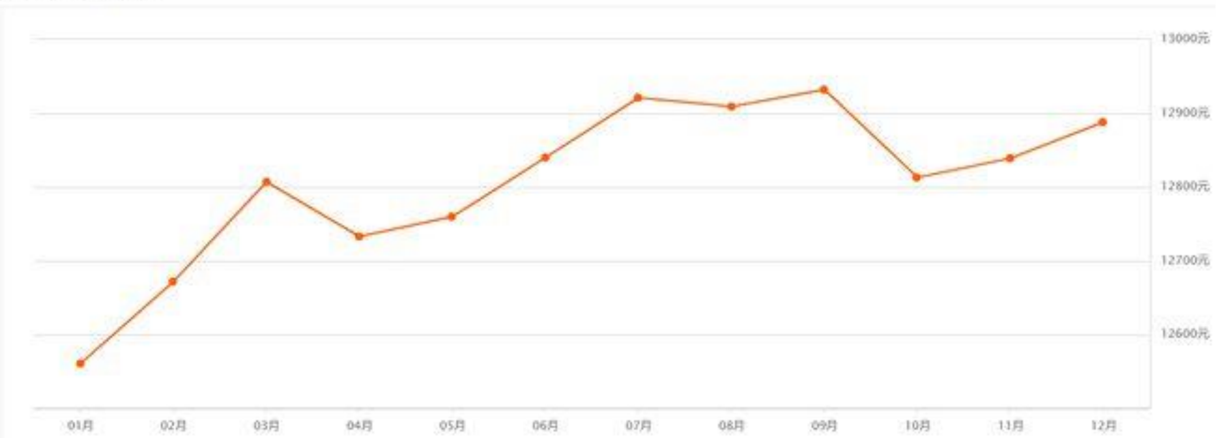
2018年兰州房价走势



2019年兰州房价走势



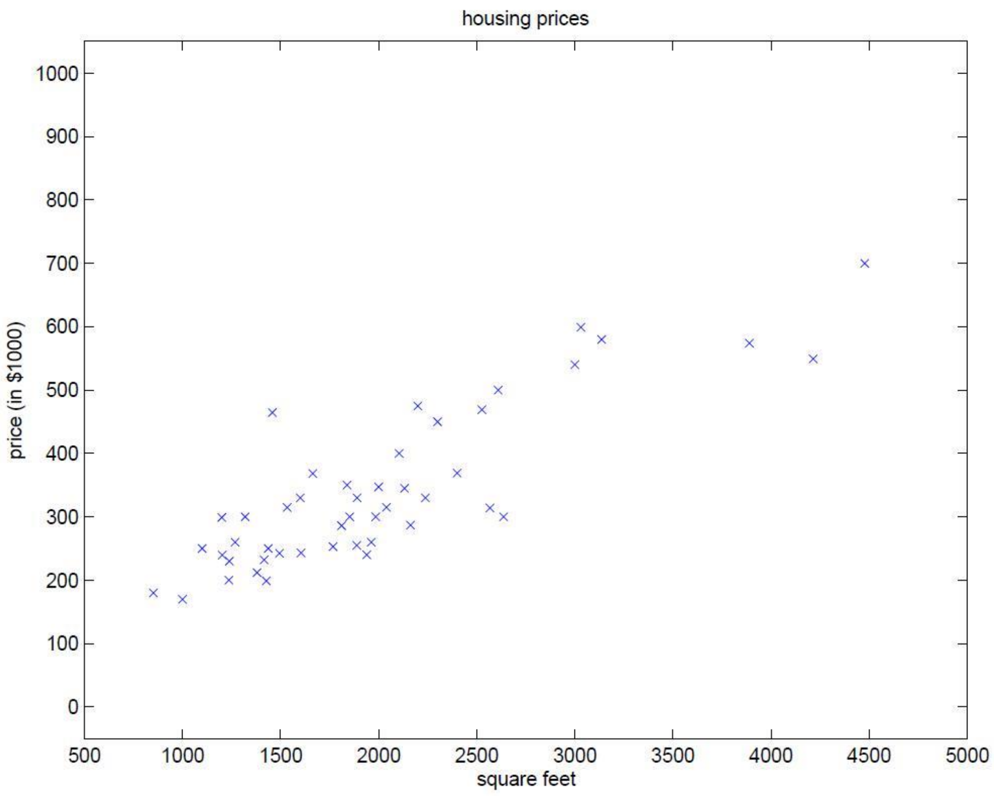
2020年兰州房价走势



举例分析

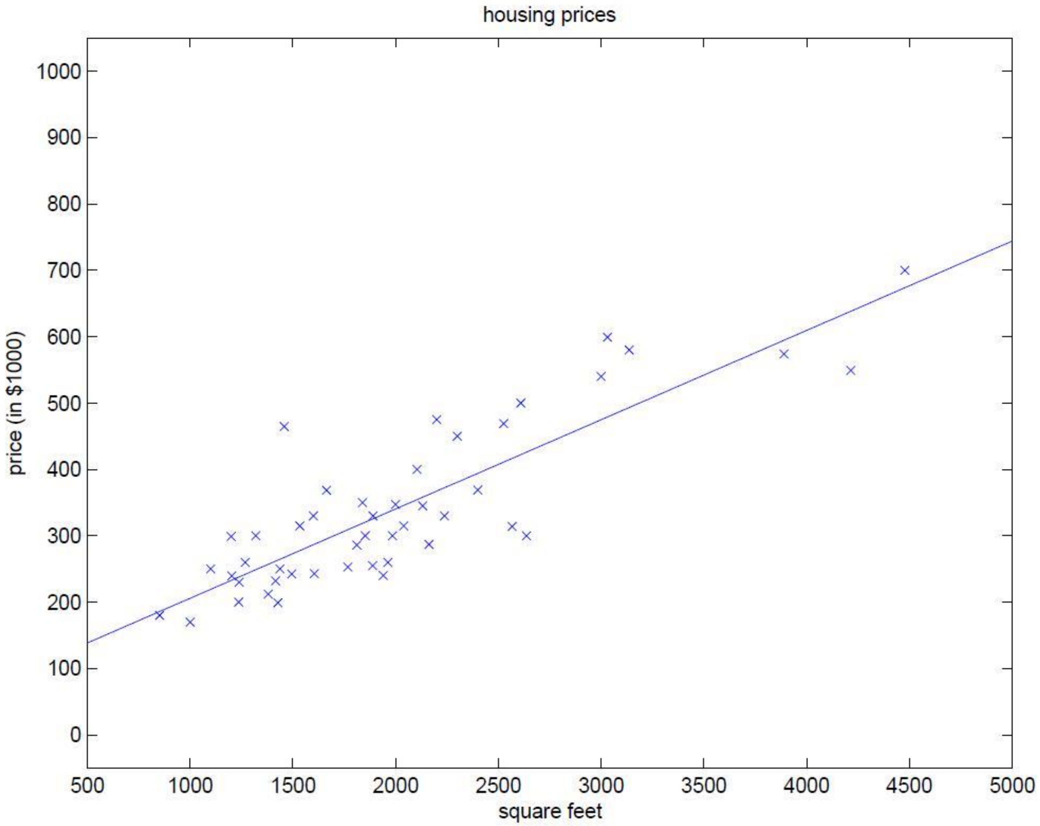
Living area (feet^2)	Price (1000\$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

房价与面积对应数据集



二维空间上的房价与面积对应图

举例分析



同时，分析得到的线性方程为：

$$h_\theta(x) = \theta_0 + \theta_1 x \qquad (1)$$

举例分析

如果增添了一个自变量：房间数，那么数据集可以如下所示：

Living area ($feet^2$)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

$$\theta = (X^T X)^{-1} X^T y$$

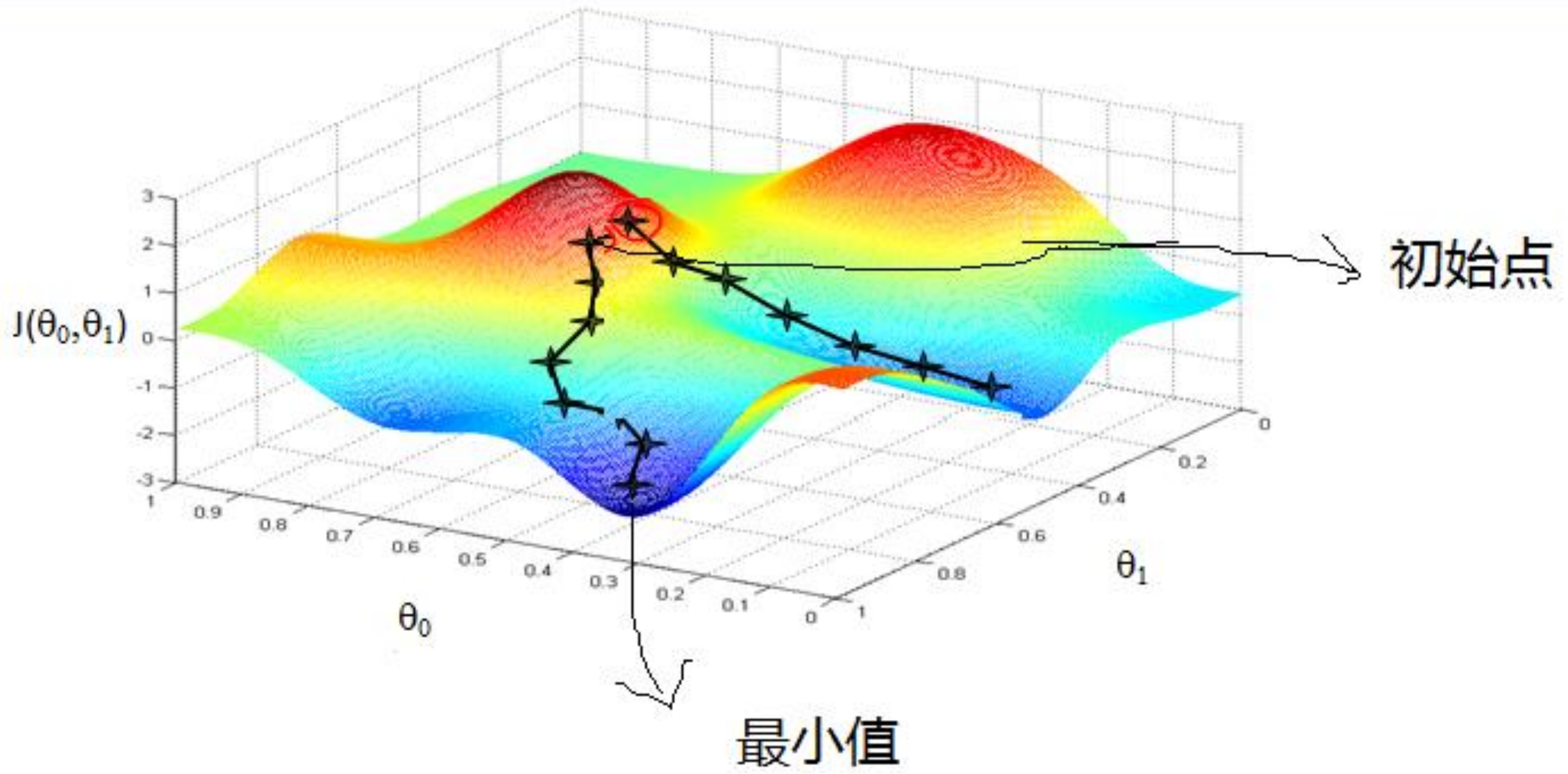
那么，分析得到的线性方程应如下所示：

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \qquad (2)$$

因此，无论是一元线性方程还是多元线性方程，可统一写成如下的格式：

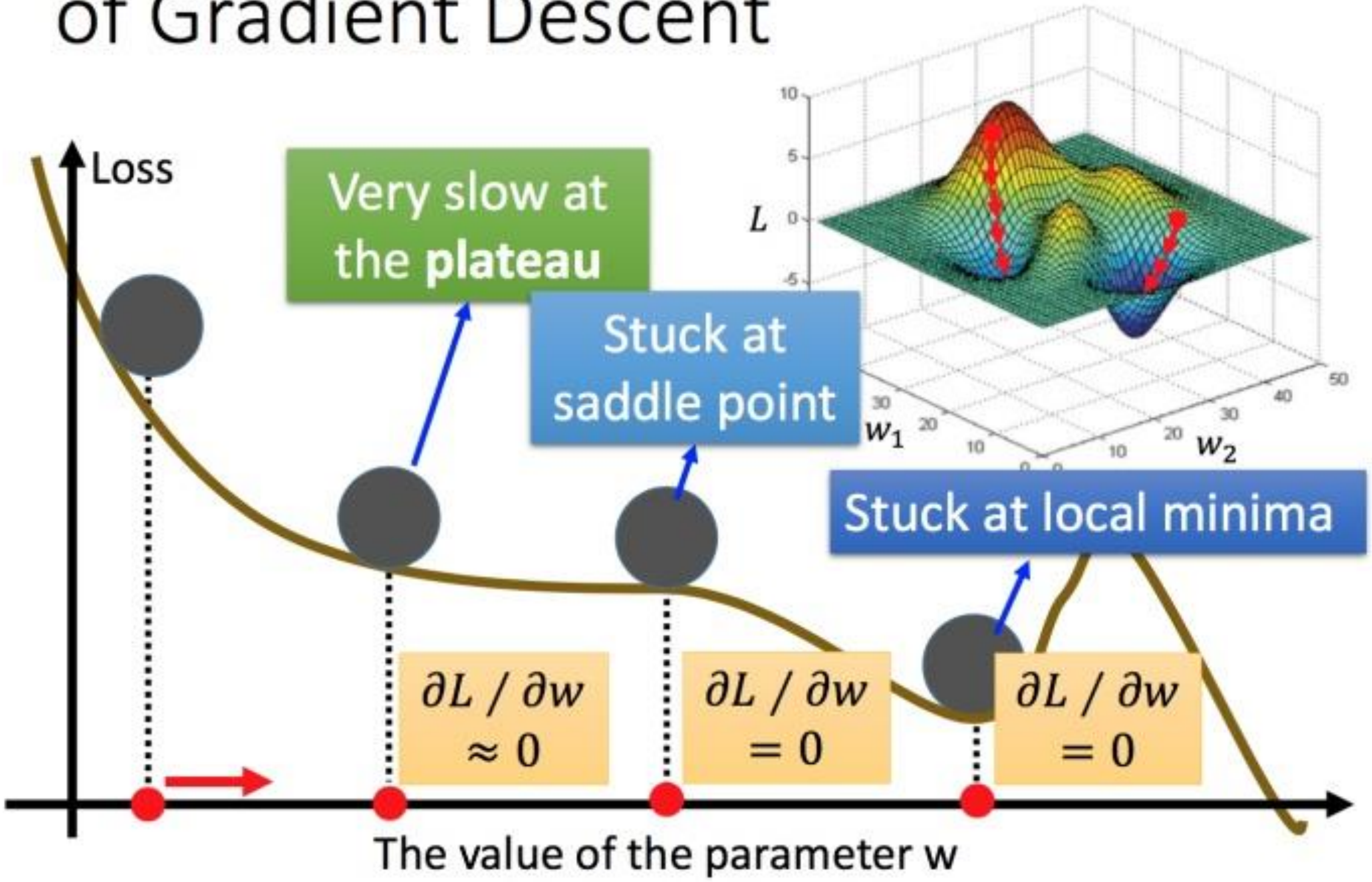
$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T X \qquad (3)$$

梯度下降算法



J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1
初始点
最小值

More Limitation of Gradient Descent

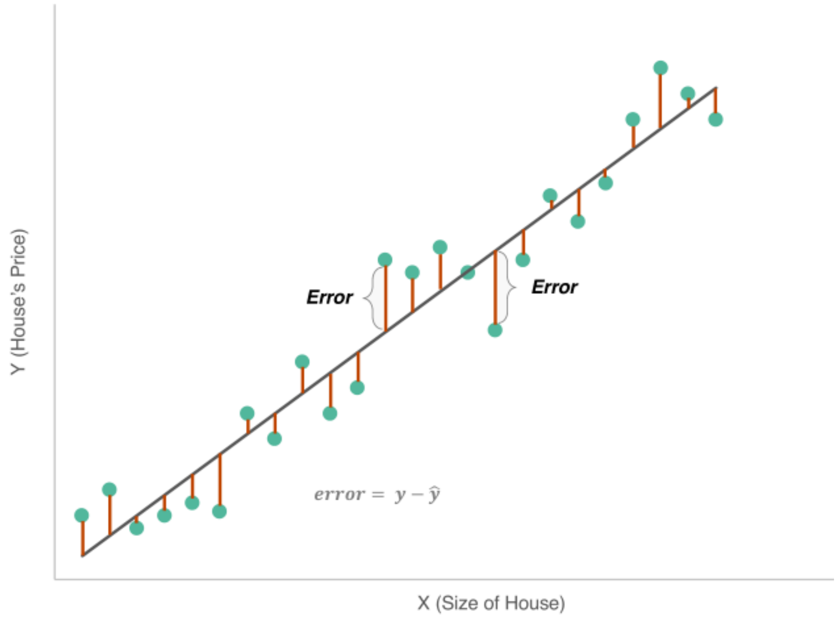


梯度下降算法

- 为了得到目标线性方程，我们只需确定公式（3）中的Θ。
- 使用一个损失函数(loss function) 来评估h(x)函数的好坏。

$$h(x)=\sum_{i=0}^{n}\theta_i x_i=\boxed{\theta^T}X \qquad (3)$$

$$J(\theta)=\frac{1}{2}\sum_{i=1}^{m}\left(\boxed{h_\theta\left(x^i\right)}-y^i\right)^2 \qquad (4)$$



梯度下降算法

由之前所述，求ΘT的问题演变成了求J(Θ)的极小值问题，这里使用梯度下降法。而梯度下降法中的梯度方向由J(Θ)对Θ的偏导数确定，由于求的是极小值，因此梯度方向是偏导数的反方向。

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \qquad (5)$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2$$

$$= 2 \cdot \frac{1}{2} (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y)$$

$$= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - \mathrm{y}\right)$$

$$\boxed{= (h_\theta(x) - y) x_j}$$

梯度下降算法

所以公式(5)就演变成： $$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)} \qquad (6)$$

当样本数量m不为1时，

$$\mathrm{J}(\theta) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^i\right) - y^i\right)^2 \qquad (4)$$

$$\theta_j := \theta_j + \alpha\sum_{i=1}^{m}(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)} \qquad (7)$$

梯度下降算法

当样本集数据量m很大时，批量梯度下降算法每迭代一次的复杂度为O(mn),复杂度很高。因此，为了减少复杂度，当m很大时，我们更多时候使用**随机梯度下降算法**(stochastic gradient descent),算法如下所示：

Loop{

For i=1 to m{

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)x_j^{(i)} \qquad \text{(for every j)}$$

}

}

即每读取一条样本，就迭代对ΘT进行更新

梯度下降算法

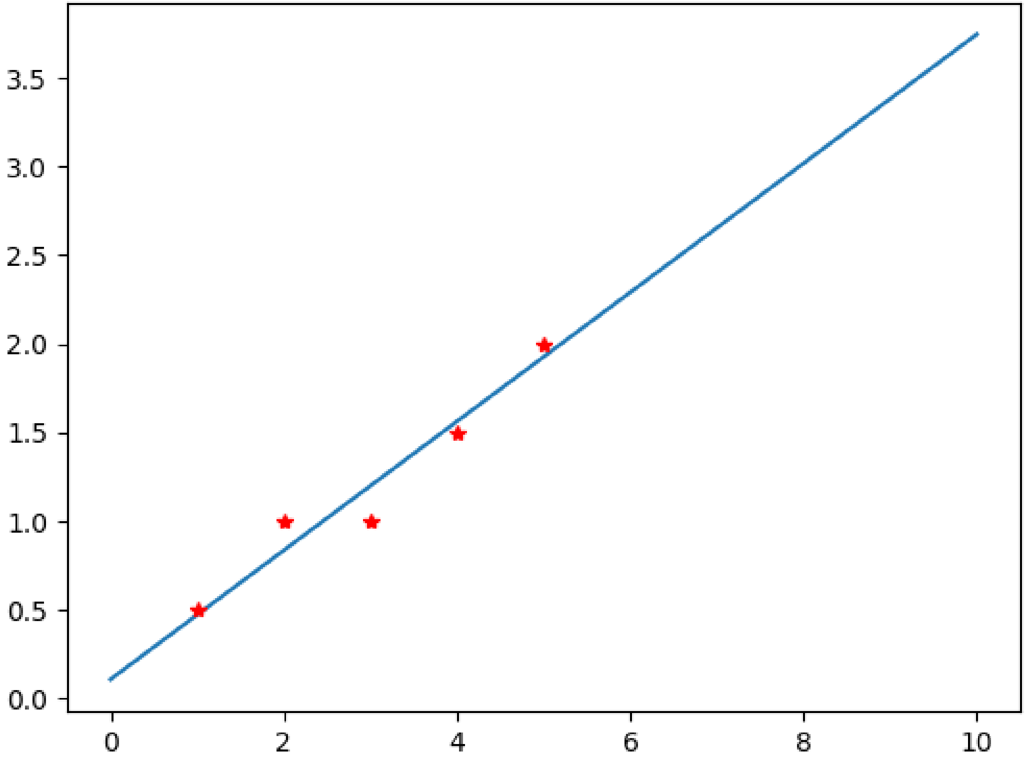
梯度下降算法的优缺点

1. 全样本学习

2. 单样本学习

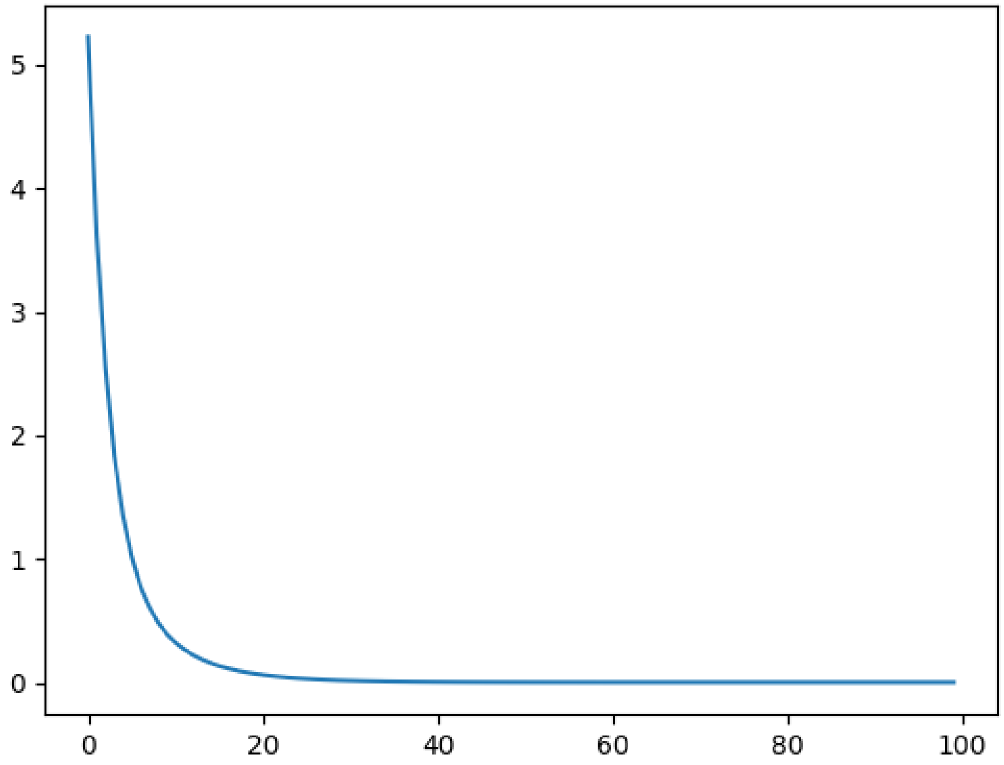
3. 小批量梯度下降(mini-batch gradient decent)

梯度下降算法



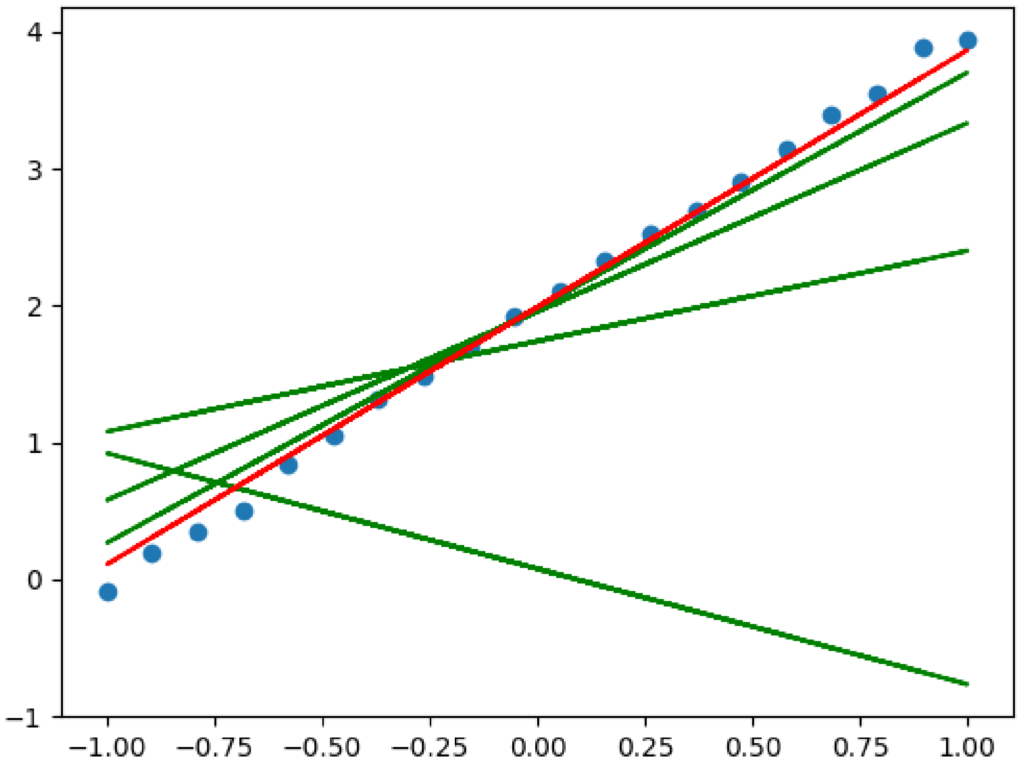

```python
import matplotlib.pyplot as plt
import numpy as np
def J(x,y,a):
    Jtemp = 0
    for xtemp, ytemp in zip(x, y):
        Jtemp += (a[0] + a[1] * xtemp - ytemp) ** 2
    return Jtemp/2

def gradient_decent(x,y):
    a=[0,0]
    for i in range(1000):
        print(J(x,y,a))
        for xtemp, ytemp in zip(x, y):
            a[1] = a[1] - 0.01*(a[0]+a[1]*xtemp-ytemp)*xtemp
            a[0] = a[0] - 0.01*(a[0]+a[1]*xtemp-ytemp)*1
    return a


x = np.array([1, 2, 3, 4, 5])
y = np.array([0.5, 1, 1, 1.5, 2])
for x0,y0 in zip (x,y):
    plt.plot(x0, y0, 'r*')

a = gradient_decent(x,y)

xp = np.linspace(0, 6, 2)
yp = a[0]+ a[1] * xp
plt.plot(xp, yp)
plt.show()
```

梯度下降算法




```
#encoding:utf8
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras.layers import Dense
from keras.models import Sequential
from keras.optimizers import SGD
def sgd(x_train, y_train):
    plt.scatter(x_train, y_train)
    model = Sequential()
    model.add(Dense(1, activation='linear', input_shape=(1,)))
    model.summary()
    sgd = SGD(lr=0.1)
    model.compile(loss='mse', optimizer=sgd)
    history = model.fit(x_train, y_train, epochs=100)
    y_predict = model.predict(x_train)
    plt.plot(x_train, y_predict, 'r')
    plt.show()
    plt.plot(history.epoch, history.history['loss'], label='train loss')
    plt.show()

x = np.linspace(-1, 1, 20)
np.random.shuffle(x)
y = 2*x + 2 + np.random.normal(0, 0.05, (20, ))
sgd(x,y)
```

梯度下降算法




```
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras.layers import Dense
from keras.models import Sequential
from keras.optimizers import SGD
def sgd2(x_train, y_train):
    plt.scatter(x_train, y_train)
    model = Sequential()
    model.add(Dense(1, activation='linear', input_shape=(1,)))
    model.summary()
    sgd = SGD(lr=0.02)
    model.compile(loss='mse', optimizer=sgd)
    for i in range(200):
        cost = model.train_on_batch(x_train, y_train)
        print(cost)
        y_predict = model.predict(x)
        if 199 == i:
            plt.plot(x, y_predict, 'r')
        if i%50 == 0:
            plt.plot(x, y_predict, 'g')
    plt.show()

x = np.linspace(-1, 1, 20)
np.random.shuffle(x)
y = 2*x + 2 + np.random.normal(0, 0.05, (20, ))
sgd2(x,y)
```

Thank you! Questions?