北京理工大學珠海學院

BEIJING INSTITUTE OF TECHNOLOGY, ZHUHAI

# 基于多元线性回归的台北房价预测

学　　院：数理与土木工程学院

课　　程：最优化方法

班　　级：数据科学与大数据技术 2 班

姓　　名：肖宇涵　　学　　号：211205101786

中国 · 珠海

2022 年 12 月 23 日

# 基于多元线性回归的台北房价预测

## 摘 要

本文旨在利用多元线性回归模型，来预测房价。众所周知房屋的价格跟许许多多的因素有关，所以本文旨在建立一个模型通过台北的数据集去预测房价。首先，收集城市的数据，比如到最近的地铁站的距离，步行生活圈便利店的数量，收集关于房屋的一些变量，如建造的年份，和房子所在的经度和纬度。然后，利用最小二乘法和梯度下降法来进行参数优化，从而最大化模型的准确度。建立多元线性回归模型，以此来预测房价。最后，使用偏差平方和平均绝对误差来评估模型的准确性。

任务一：利用最小二乘法和梯度下降法来进行参数优化，采用传统梯度下降法跟随机梯度下降法从而最大化模型的准确度。建立多元线性回归模型，，分析出影响因素跟房价的模型从而预测出房价。

任务二：利用预测值与真实值进行拟合来评估模型的准确性,且对任务一所选择的数学模型进一步优化并展示结果。

**关键词：**多元线性回归　最小二乘法　梯度下降法

# 目 录

# 1 任务描述

在当今社会，房价与许许多多的因素有关。许多人都想知道自己所处的房屋位置的大致价格，为了能更快更好地实现这一目标，建立一个模型去求解预测。该数据集有414行和7列。它提供了从台湾新北市新店区收集的房地产估值的市场历史数据集。

任务一：利用最小二乘法和梯度下降法来进行参数优化，采用传统梯度下降法跟随机梯度下降法从而最大化模型的准确度。建立多元线性回归模型，，分析出影响因素跟房价的模型从而预测出房价。

任务二：对任务一所选择的数学模型进一步优化并展示结果。

其数据集含义如下表格展示

表 1: Beecy.

| 属性名称 | 定义 |
|---|---|
| 交易日 | 交易日期（例如2013.250=2013年3月等） |
| X2 house age | 屋龄（单位：年） |
| X3 distance to the nearest MRT station | 到最近地铁站距离（单位：米） |
| X4 number of convenience stores | 步行生活圈便利店数量 |
| X5 latitude | 地理坐标、纬度（单位：度） |
| X6 longitude | 地理坐标、经度（单位：度） |
| Y house price of unit area | 单位面积房价（10000新台币/坪） |

# 2 解决思路

任务一：利用最小二乘法和梯度下降法来进行参数优化，从而最大化模型的准确度。建立多元线性回归模型，以此来预测房价

任务二：利用预测值与真实值进行拟合来评估模型的准确性。

# 3 模型的建立与求解

## 3.1 模型的提出

### 3.1.1 模型建立

多元线性回归模型是一种用来计算两个或更多个变量间相关性的统计模型，它可以用来预测未来变量的变化趋势，并且可以帮助我们了解和控制变量之间的关系。多元线性回归模型可以描述出自变量与因变量之间的线性关系，这种线性关系可以用一个线性

方程式表示出来，其中自变量为解释变量，因变量为被解释变量。多元线性回归模型可以帮助我们预测未来的变量变化，因此可以用于房价的预测。

本文以单位面积房价作为因变量（$Y$），屋龄（单位：年）、到最近的地铁站的距离、步行生活圈便利店数量、纬度、经度作为自变量（$x_1, x_2, x_3, x_4, x_5$），探究 $x$ 对 $Y$ 的影响情况。首先建立一个多元线性回归模型，假设其函数为：

$$Y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4 + \theta_5 x_5 \tag{1}$$

在该模型中，$\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5$ 是多元线性回归模型的系数。于是求解回归模型转化为求解多元回归系数。在保证拟合程度的同时拥有较好的泛化能力。将上述式子转化为一般形式为:

$$Y^{(i)} = X^{(i)}\theta$$

其中 $\theta$ 和 $X$ 为：

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix}_{6\times 1} \qquad \mathbf{X} = \begin{bmatrix} 1 & x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}_{414\times 6}$$

### 3.1.2 模型求解

对于线性回归，我们的假设需要求的原函数为

$$f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \ldots\ldots + \theta_n x^n$$

我们用代价函数去描述多元线性回归模型的好坏，多元线性回归的代价函数如下：

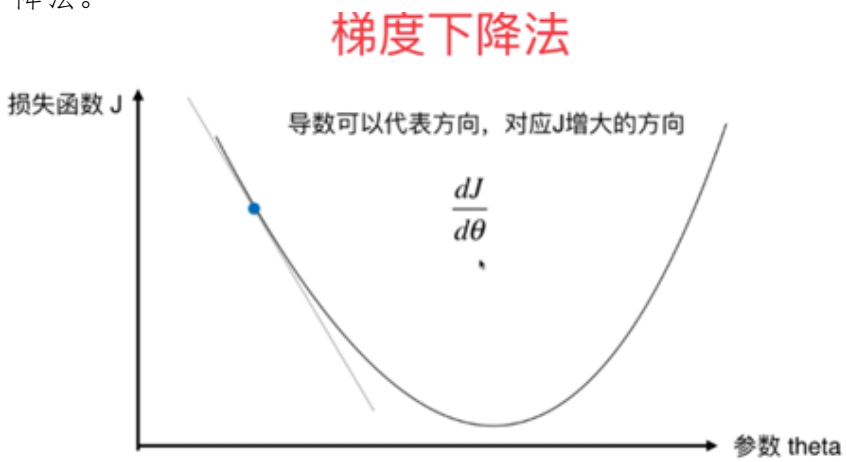
$$E(\beta) = \frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - f_\beta\left(x^{(i)}\right)\right)^2$$

在前面文字所讲述的部分，我们计算出了不同参数 $\beta$ 以及对应的 $E(\beta)$, 而想找一个 $E(\beta)$ 取其假设函数作为线性回归拟合的方程。就要用到采用梯度下降法作控制不断降低其误差值，故我们先采用传统梯度下降法来求代价函数 $E(\beta)$ 也就是梯度范数。

**首先介绍梯度下降法**

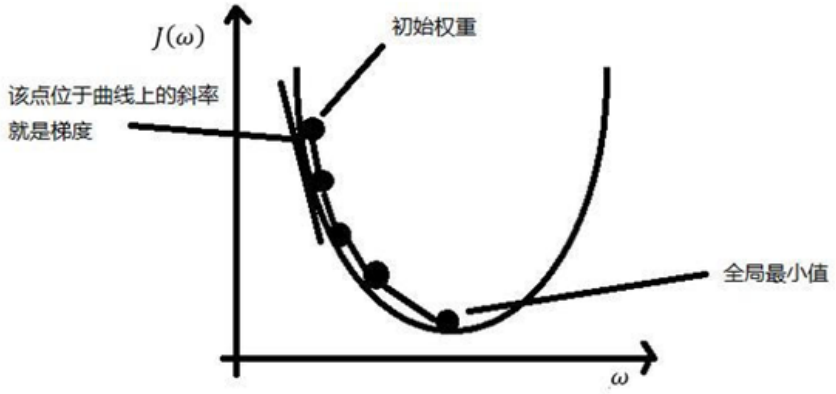
梯度下降法（Gradient descent，简称 GD）是一阶最优化算法。要使用梯度下降法找到一个函数的局部极小值，必须向函数上当前点对应梯度（或者是近似梯度）的反方向的规定步长距离点进行迭代搜索。如果相反地向梯度正方向迭代进行搜索，则会接近函数的局部极大值点，这个过程则被称为梯度上升法。梯度下降法是迭代法的一种，可以用于求解最小二乘问题 (线性和非线性都可以)。在求解机器学习算法的模型参数，即

无约束优化问题时，梯度下降法和最小二乘法是最常采用的方法。在求解损失函数的最小值时，可以通过梯度下降法来迭代求解，得到最小化的损失函数和模型参数值。反过来，如果我们需要求解损失函数的最大值，这时就需要用梯度上升法来迭代了。在机器学习中，基于基本的梯度下降法发展了两种常用梯度下降方法，分别为随机梯度下降法和批量梯度下降法。



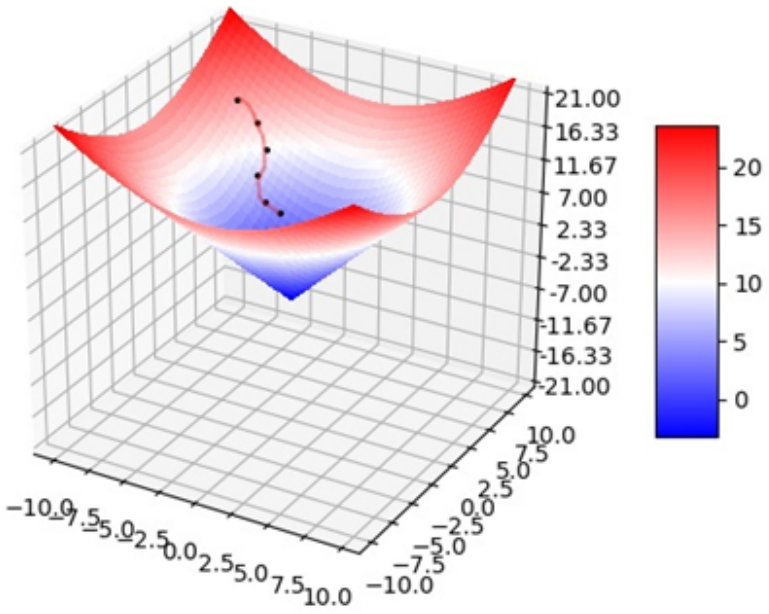

**梯度下降法的原理**

举例说明，将下山的人看作 J(w)，即表示目标函数，目的地是最低点。而中间如何到达最低点则是需要解决的问题




在当前位置求偏导，即梯度，正常的梯度方向类似于上山的方向，是使值函数增大的，下山最快需使最小，从负梯度求最小值，这就是梯度下降。梯度上升是直接求偏导，梯度下降则是梯度上升的负值。由于不知道怎么下山，于是需要走一步算一步，继续求解当前位置的偏导数。这样一步步的走下去，当走到了最低点，此时我们能得到一个近似最优解。

可以看出梯度下降有时得到的是局部最优解，如果损失函数是凸函数，梯度下降法得到的解就是全局最优解。




所以梯度下降的公式为：

$$\theta_i = \theta_i - \alpha \frac{\partial J\left(\theta_0, \theta_1, \cdots, \theta_n\right)}{\partial \theta_i}$$

求解步骤：

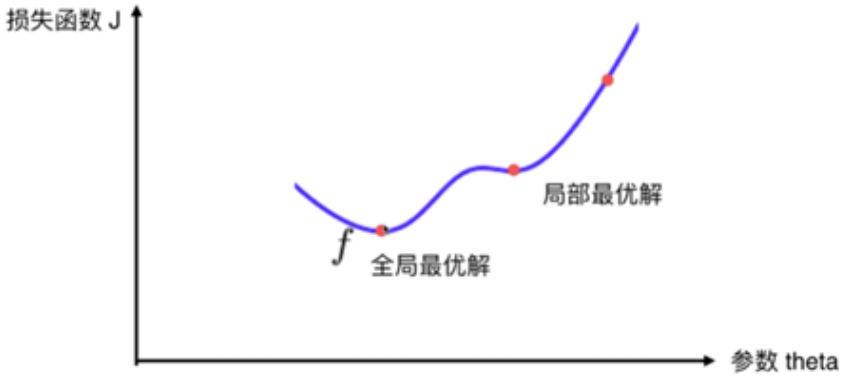
（1）确定当前位置的损失函数的梯度，对于 $\theta_i$，其梯度表达式为：$\frac{\partial J(\theta_0,\theta_1,\cdots,\theta_n)}{\partial \theta_i}$

（2）用步长乘以损失函数的梯度，得到当前位置下降的距离，即 $\alpha \frac{\partial J(\theta_0,\theta_1,\cdots,\theta_n)}{\partial \theta_i}$

（3）确定是否所有的 $\theta_i$ 梯度下降的距离都小于 $\varepsilon$，如果小于 $\varepsilon$ 则算法终止，当前所有的 $\theta_i$ 即为最终结果。否则进入第（4）步。

（4）更新所有的 $\theta_i$，更新表达式如下，更新完成后转入步骤（1）：

$$\theta_i = \theta_i - \alpha \frac{\partial J\left(\theta_0, \theta_1, \cdots, \theta_n\right)}{\partial \theta_i}$$



所以我们现在可以得到参数的更新表达式

$$\theta j := \theta j - \sum_{i=1}^{n} \left[ f_\theta \left( x^i \right) - y^i \right] x_j^{(j)}$$

已知 $\theta_k$，下降方向 $d_k = -g_k$，求解 $\alpha_k$，其中可利用下降方向得到 $\theta_{k+1} = \theta_k + \alpha_k d_k$，进而得到函数

$$f_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \ldots\ldots + \theta_n x^n$$

编程实现时将步长设置为 0.0006，$\epsilon$ 设置为 $10 * *(-5)$, 得到的结果如下：

| | 输出结果 |
|---|---|
| 梯度范数为： | 9.648781772123521e-06 |
| 迭代次数： | 416 |
| 极值点为： | [-4.33100649e-14] |
| | [-2.25159567e-01] |
| | [-3.95064142e-01] |
| | [ 2.51773207e-01] |
| | [ 2.16862577e-01] |
| | [-8.80392413e-03] |
| 极值： | 88.76952675807274 |
| 相对误差率为： | 0.1872095467335419 |

**所以最后得到的模型为**

$$Y = -4.33100649e{+}2.25159567ex_1{+}3.95064142ex_2{-}2.51773207ex_3{-}2.16862577ex_4{+}8.80392413ex_5$$

**其代码实现如下所示**

```
import pandas as pd
import numpy as np

# 数据导入
data=pd.read_csv('F:\\资料\\学习资料\\最优化方法\\结课论文\\datasets-master
    \\Real Estate Valuation\\Real estate valuation data set.csv')

```

```
column_names = ['X2 house age','X3 distance to the nearest MRT station'
    ,'X4 number of convenience stores','X5 latitude','X6 longitude','Y
    house price of unit area']
x = np.array([data['X2 house age'],data['X3 distance to the nearest MRT
    station'],data['X4 number of convenience stores'],data['X5 latitude'],
    data['X6 longitude']]).T
y = np.array([data['Y house price of unit area']]).reshape(-1,1)

N,n = x.shape

# 数据标准化
from sklearn.preprocessing import StandardScaler
sx = StandardScaler(); x_std = sx.fit_transform(x)
sy = StandardScaler(); y_std = sy.fit_transform(y)
ones = np.ones((N,1))
X = np.hstack((ones,x_std))
Y = y_std

def fun(w,X,Y):
return 0.5*np.linalg.norm(np.dot(X,w)-Y)**2


def g(w,X,Y):
return np.dot(X.T,np.dot(X,w)-Y)

epsilon = 10**(-5); k = 0; alphak= 0.006; MaxN = 10000
theta1 = np.random.rand(6)

wk = np.array(theta1).reshape(-1,1)
while np.linalg.norm(g(wk,X,Y)) > epsilon:
dk = - g(wk,X,Y)
wk = wk + alphak*dk
k += 1
if k > MaxN: break

print('梯度范数为：',np.linalg.norm(g(wk,X,Y)),'迭代次数：',k)
print('极值点为：',wk,'极值：',fun(wk,X,Y))

def regressfun(w,X):
```

```
42          return  np.dot(X,w)
43          Y_pre =  regressfun (wk,X)
44          y_pre  = sy. inverse_transform (Y_pre)
45          print ('相对误差率为：',np.sum((abs(y_pre-y))/y)/N)
```

### 3.2 模型优化

小批量梯度下降算法是每次从训练样本集上随机抽取一个小样本集，在抽出来的小样本集上采用全梯度下降算法迭代更新权重。被抽出的小样本集所含样本点的个数称为 $batchsize$，通常设置为 2 的幂次方，更有利于 GPU 加速处理。特别的，若 $batchsize = 1$，则变成了随机梯度下降算法；若 $batchsize$ =n，则变成了全梯度下降算法。其迭代形式为：

$$\theta = \theta - \eta \cdot \nabla_\theta J\left(\theta; x^{(i:i+n)}; y^{(i:i+n)}\right)$$

其中迭代次数虽有增加，但是模型的拟合程度更优了，优化后的模型相对误差率为 0.18843110770048596，拟合程度更优了。

## 4 总结与评价

### 4.1 总结

本文通过对房地产的各个因素进行分析，用梯度下降法对多元线性回归进行了求解。之后为了提高模型的优化程度，利用小批量梯度下降法对模型又一次进行了分析和求解，使得模型的误差减小了，拟合度增加了。

### 4.2 评价

优点：利用对多个因变量进行了回归分析，经过进一步优化使得模型的拟合度更好了，也减小了相对误差和绝对误差。适用范围广，能用于多个领域进行相关因素的分析。
缺点：在考虑拟合程度和误差大小的情况下，不可避免地增加了代码的运算时间，在运算效率上还有欠缺。

# 参 考 文 献

[1] 黄永昌.《scikit-learn 机器学习：常用算法原理及编程实战》.

[2] 姜启源.《数学模型》（第三版）. 高等教育出版社.

[3] 高等教育出版社. https://blog.csdn.net/weixin_38100489/article/details/78175928

[4] 高等教育出版社. https://blog.csdn.net/weixin_38100489/article/details/78175928

# 附　录

## 附录 1：小批量梯度下降法的程序

```
import pandas as pd
import numpy as np

# 数据导入
data=pd.read_csv('F:\\资料\\学习资料\\最优化方法\\结课论文\\datasets-master\\Real Estate Valuation\\Real estate valuation data set.csv')

column_names = ['X2 house age','X3 distance to the nearest MRT station','X4 number of convenience stores','X5 latitude','X6 longitude','Y house price of unit area']
#x = np.array([data['age'], data['bmi']]).T
x = np.array([data['X2 house age'],data['X3 distance to the nearest MRT station'],data['X4 number of convenience stores'],data['X5 latitude'],data['X6 longitude']]).T
#y = np.array([data['charges']]).reshape(-1,1)
y = np.array([data['Y house price of unit area']]).reshape(-1,1)

N,n = x.shape

# 数据标准化
from sklearn.preprocessing import StandardScaler
sx = StandardScaler(); x_std = sx.fit_transform(x)
sy = StandardScaler(); y_std = sy.fit_transform(y)
ones = np.ones((N,1))
X = np.hstack((ones,x_std))
Y = y_std

def fun(w,X,Y):
    return 0.5*np.linalg.norm(np.dot(X,w)-Y)**2


def g(w,X,Y):
    return np.dot(X.T,np.dot(X,w)-Y)

epsilon = 10**(-5); k = 0; alphak= 0.001; MaxN = 100000 #整体迭代最佳步长0.0009
theta1 = np.random.rand(6)
```

```
32
33 wk = np.array (theta1).reshape(-1,1)
34
35 import random
36 N_batch = 100 # 取N为整体迭代，取1为单点迭代，取其他为批量迭代中一批的样本数
37 Index_batch = random.sample(range(0,N),N_batch)
38 X_batch = X[Index_batch]
39 Y_batch = Y[Index_batch]
40
41 while np.linalg.norm(g(wk,X_batch,Y_batch)) > epsilon:
42     Index_batch = random.sample(range(0,N),N_batch)
43     X_batch = X[Index_batch]
44     Y_batch = Y[Index_batch]
45     dk = - g(wk,X_batch,Y_batch)
46     wk = wk + alphak*dk
47     k += 1
48     print('梯度范数为：',np.linalg.norm(g(wk,X,Y)))
49     if k > MaxN: break
50
51 print('梯度范数为：',np.linalg.norm(g(wk,X,Y)),'迭代次数：',k)
52 print('极值点为：',wk,'极值：',fun(wk,X,Y))
53
54 def regressfun(w,X):
55     return np.dot(X,w)
56 Y_pre = regressfun(wk,X)
57 y_pre = sy.inverse_transform(Y_pre)
58 print('相对误差率为：',np.sum((abs(y_pre-y))/y)/N)
59
60
61 from sklearn.linear_model import LinearRegression
62 import matplotlib.pyplot as plt
63
64 z = list(range(len(y)))
65 plt.plot(z,Y,label = 'true')#实际的y
66 plt.plot(z,Y_pre,label = 'pred')#预测的y
67 plt.legend()#展示下图例
68 plt.show()
69
70
```

```
#%%
import pandas as pd
import numpy as np

# 数据导入
data=pd.read_csv('F:\\资料\\学习资料\\最优化方法\\结课论文\\datasets-master\\Real Estate Valuation\\Real estate valuation data set.csv')

column_names = ['X2 house age','X3 distance to the nearest MRT station','X4 number of convenience stores','X5 latitude','X6 longitude','Y house price of unit area']
#x = np.array([data['age'], data['bmi']]).T
x = np.array([data['X2 house age'],data['X3 distance to the nearest MRT station'],data['X4 number of convenience stores'],data['X5 latitude'],data['X6 longitude']]).T
#y = np.array([data['charges']]).reshape(-1,1)
y = np.array([data['Y house price of unit area']]).reshape(-1,1)

N,n = x.shape

# 数据标准化
from sklearn.preprocessing import StandardScaler
sx = StandardScaler(); x_std = sx.fit_transform(x)
sy = StandardScaler(); y_std = sy.fit_transform(y)
ones = np.ones((N,1))
X = np.hstack((ones,x_std))
Y = y_std

def fun(w,X,Y):
    return 0.5*np.linalg.norm(np.dot(X,w)-Y)**2


def g(w,X,Y):
    return np.dot(X.T,np.dot(X,w)-Y)

epsilon = 10**(-5); k = 0; alphak= 0.00001; MaxN = 10000 #整体迭代最佳步长0.0009
theta1 = np.random.rand(6)

wk = np.array(theta1).reshape(-1,1)
```

```
while np.linalg.norm(g(wk,X,Y)) > epsilon:
    dk = -g(wk,X,Y)
    wk = wk + alphak*dk
    k += 1
    print('梯度范数为：',np.linalg.norm(g(wk,X,Y)))
    if k > MaxN: break

print('梯度范数为：',np.linalg.norm(g(wk,X,Y)),'迭代次数：',k)
print('极值点为：',wk,'极值：',fun(wk,X,Y))

def regressfun(w,X):
    return np.dot(X,w)
Y_pre = regressfun(wk,X)
y_pre = sy.inverse_transform(Y_pre)
print('相对误差率为：',np.sum((abs(y_pre-y))/y)/N)

from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

z = list(range(len(y)))
plt.plot(z,Y,label = 'true')#实际的y
plt.plot(z,Y_pre,label = 'pred')#预测的y
plt.legend()#展示下图例
plt.show()
```