linear regression-Least Squares Method

In [29]:
```
import numpy as np
import pandas as pd
```

波士顿房价数据集字段说明

CRIM 房屋所在镇的犯罪率
ZN 面积大于25000平方英尺住宅所占的比例
INDUS 房屋所在镇非零售区域所占比例
CHAS 房屋是否位于河边，如果位于河边，则值为1，否则值为0
NOX 一氧化氮的浓度
RM 平均房间数量
AGE 1940年前建成房屋所占的比例
DIS 房屋距离波士顿五大就业中心的加权距离
RAD 距离房屋最近的公路
TAX 财产税额度
PIRATIO 房屋所在镇师生比例
B 计算公式：1000（房屋所在镇非美籍人口所占比例-0.63）*2
LSTAT 弱势群体人口所占比例
MEDV 房屋的平均价格 （需预测值）

In [30]:
```
data = pd.read_csv(r"boston.csv")
# data.head()
# # 查看数据基本信息，是否存在缺失值（如有缺失值，需删除）
data.info()
# # 查看是否有重复值，返回False则无重复值
# data.duplicated().any()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 452 entries, 0 to 451
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     452 non-null    float64
 1   ZN       452 non-null    float64
 2   INDUS    452 non-null    float64
 3   CHAS     452 non-null    int64  
 4   NOX      452 non-null    float64
 5   RM       452 non-null    float64
 6   AGE      452 non-null    float64
 7   DIS      452 non-null    float64
 8   RAD      452 non-null    int64  
 9   TAX      452 non-null    int64  
 10  PIRATIO  452 non-null    float64
 11  B        452 non-null    float64
 12  LSTAT    452 non-null    float64
 13  MEDV     452 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 49.6 KB
```

In [31]:
```
class LinearRegression:
    """使用python实现的线性回归（最小二乘法）"""
```

```
    def fit(self,X,y):
        """根据提供的训练数据X，对模型进行训练
        Parameters
        ------
        X:类数组类型，形状：[样本数量，特征数量]
          特征矩阵，用来对模型进行训练
        y:类数组类型，形状：[样本数量]
        """
        # 如果X是数组对象的一部分，而非完整的对象数据（例如，X是由其他对象通过切片
        # 则无法完成矩阵的转换
        # 这里创建X的拷贝对象，避免转换矩阵时失败
        X = np.asmatrix(X)
        # y是一维结构（行向量或列向量），一维结构可以不用进行拷贝
        # 注意：进行矩阵运算时，需要二维结构，因此通过reshape将y转换为二维结构（列
        y = np.asmatrix(y).reshape(-1,1) # -1表示，根据y元素总数除以列数，自动进行
        # 通过最小二乘公式，求解出最佳的权重值
        self.w_ = (X.T * X).I  * X.T *y # X.T，转置X; .I, 求幂

    def predict(self,X):
        """根据参数传递的样本X，对样本数据进行预测
        Parameters
        ------
        X: 类数组类型。形状：[样本数量，特征数量]
           待预测的样本特征（属性）
        Returns
        ------
        result:数组类型
               预测的结果
        """
        # 将X转换成矩阵，注意：需要对X进行拷贝
        X = np.asmatrix(X.copy())
        result = X * self.w_
        # 将矩阵转换成ndarray数组，ravel()进行扁平化处理（多维转一维），然后返回结
        return np.array(result).ravel()
```

In [34]:

```
# 不考虑结局的情况
# 训练集和测试集分组
t = data.sample(len(data),random_state=0)
train_X = t.iloc[:400,:-1]
train_y = t.iloc[:400,-1]
test_X = t.iloc[400:,:-1]
test_y = t.iloc[400:,-1]

lr = LinearRegression()
lr.fit(train_X, train_y)
result = lr.predict(test_X)
# result
display(np.mean((result - test_y) ** 2))
# 查看模型的权重值
display(lr.w_)
```

```
10.804546509313566
```

```
matrix([[-0.32635109],
        [ 0.03994793],
        [ 0.0241933 ],
        [ 2.44585217],
        [-2.63532152],
        [ 6.26736543],
        [-0.01816585],
        [-1.04433992],
        [ 0.26789795],
        [-0.0103102 ],
        [-0.51590473],
        [ 0.01763495],
        [-0.44055414]])
```

In [37]:
```
# 考虑截距，增加一列，该列的所有值都是1
t = data.sample(len(data),random_state=0)
# 按照习惯，结局作为w0，我们为之配上一个x0，x0列放在最前面
new_columns = t.columns.insert(0,"Intercept")
# 重新安排列的顺序，如果值为空，则使用fill_value参数指定的值进行填充
t = t.reindex(columns = new_columns,fill_value = 1)
t
```

Out[37]:

	Intercept	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX
124	1	0.06417	0.0	5.96	0	0.499	5.933	68.2	3.3603	5	279
54	1	6.96215	0.0	18.10	0	0.700	5.713	97.0	1.9265	24	666
298	1	0.33045	0.0	6.20	0	0.507	6.086	61.5	3.6519	8	307
311	1	0.01501	80.0	2.01	0	0.435	6.635	29.7	8.3440	4	280
230	1	0.01096	55.0	2.25	0	0.389	6.453	31.9	7.3073	1	300
...	...	...	...	...	...	...	...	...	...	...	...
323	1	0.12650	25.0	5.13	0	0.453	6.762	43.4	7.9809	8	284
192	1	0.08707	0.0	12.83	0	0.437	6.140	45.8	4.0905	5	398
117	1	0.14932	25.0	5.13	0	0.453	5.741	66.2	7.2254	8	284
47	1	8.49213	0.0	18.10	0	0.584	6.348	86.1	2.0527	24	666
172	1	5.82115	0.0	18.10	0	0.713	6.513	89.9	2.8016	24	666

452 rows × 15 columns

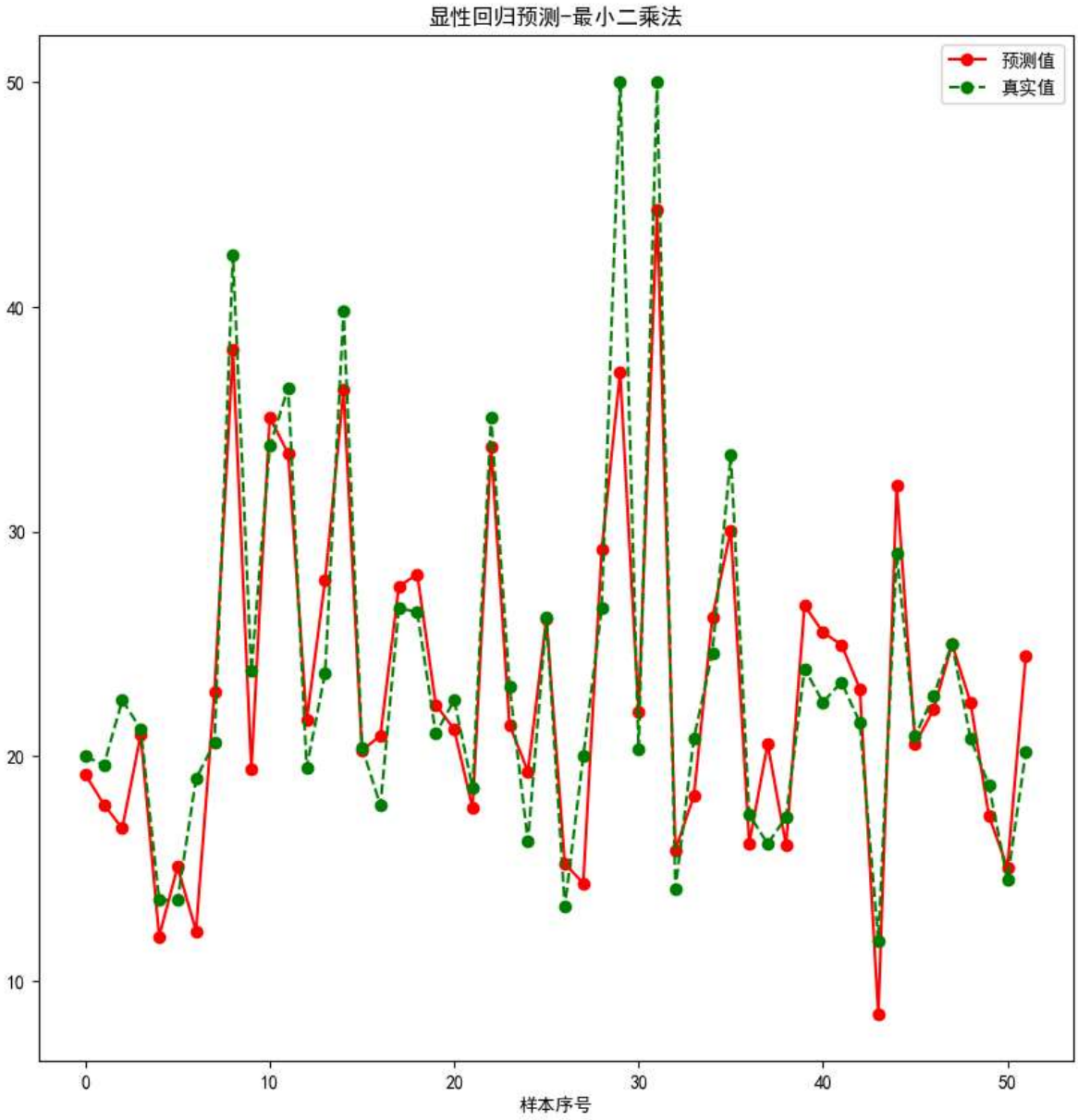
In [40]:
```
# 训练集和测试集分组
t = data.sample(len(data),random_state=0)
train_X = t.iloc[:400,:-1]
train_y = t.iloc[:400,-1]
test_X = t.iloc[400:,:-1]
test_y = t.iloc[400:,-1]

lr = LinearRegression()
lr.fit(train_X, train_y)
result = lr.predict(test_X)
# result
```

```
display(np.mean((result - test_y) ** 2))
display(lr.w_)
```

```
10.804546509313566
matrix([[-0.32635109],
        [ 0.03994793],
        [ 0.0241933 ],
        [ 2.44585217],
        [-2.63532152],
        [ 6.26736543],
        [-0.01816585],
        [-1.04433992],
        [ 0.26789795],
        [-0.0103102 ],
        [-0.51590473],
        [ 0.01763495],
        [-0.44055414]])
```

In [41]:
```
# 绘图
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rcParams["font.family"] = "SimHei"
mpl.rcParams["axes.unicode_minus"] = False
```

In [42]:
```
plt.figure(figsize=(10,10))
# 绘制预测值
plt.plot(result,"ro-",label="预测值")
# 绘制真实值
plt.plot(test_y.values,"go--",label="真实值")
plt.title("显性回归预测-最小二乘法")
plt.xlabel("样本序号")
plt.legend()
plt.show()
```


显性回归预测-最小二乘法
预测值
真实值
50
40
30
20
10
0
10
20
30
40
50
样本序号