线性回归（一）

课前准备

- 下载Anaconda软件，请点击这里进行下载。

本节要点

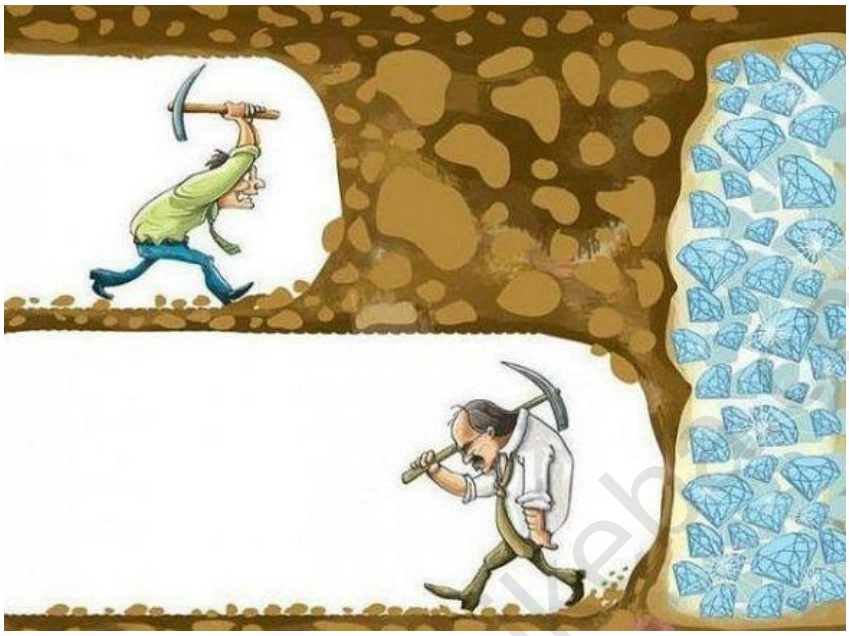
- 数据挖掘的含义。
- 模型，拟合的概念理解。
- 简单线性回归。
- 损失函数与参数求解。

何为数据挖掘

深山寻宝

江湖传闻，在神秘的深山老林中，埋藏着大量的宝藏，于是，我们整理行囊，踏上了寻宝之旅.....

然而，这是一个长年累月的过程，也许不是所有人都能坚持到最后.....



寻宝赋诗

实际上，数据挖掘与宝藏挖掘是一样的，只不过，宝藏挖掘，挖掘的是隐藏在泥土下面的黄金与钻石，而数据挖掘，挖掘的是隐藏在数据背后的价值。

我们可以将以上的寻宝场景，使用诗词的方式进行描述：

数年数月数青苔，
据悉宝藏深山埋。
挖地三尺金何在？
掘井及泉自会来。

数据挖掘定义

数据挖掘（data mining）是一个跨学科的计算机科学分支。它是用人工智能、机器学习、统计学和数据库的交叉方法在相对较大型的数据集中发现模式的计算过程。

——维基百科

数据挖掘的应用

数据挖掘的应用场景很多，其早已渗透于每一个角落，相信我们每个人都经历过，只不过，我们可能没有留意而已，例如：

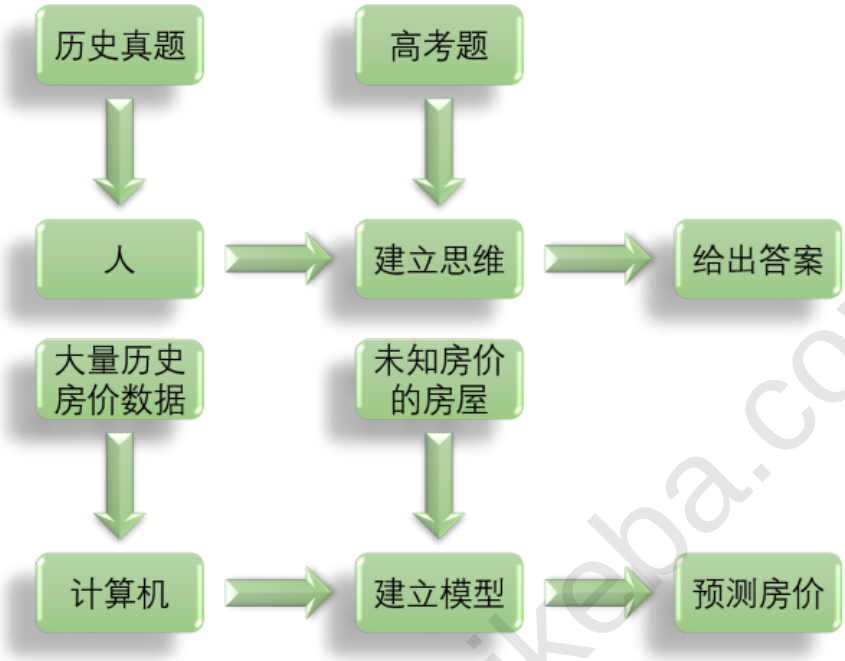
- 垃圾邮件识别。
- 贷款额度，还款预测。
- 用户流失预警。
- 关联分析，商品推荐。
- 时间序列分析。

模型

我们要实现数据挖掘，就一定要知道模型的概念，数据挖掘，最终是通过在数据上建立相关的模型来实现的。

模型含义

我们可以将模型理解为一个函数(一种映射规则）。由训练数据来确定函数的参数，当参数确定好之后，我们就可以利用该模型（函数）对未知的数据（非训练时使用的数据）进行求值，也就是进行预测。
输入模型的数据，我们称为训练数据。通过不断的进行训练，最终得到一个合适的模型，从而可以对未知的数据进行预测。而这个过程，是与我们人类学习的过程是相似的。只不过人的认知与判断是通过经验得到的，而机器的认知与判断是通过数据得到的。



数据表示

我们使用数据训练模型，例如，某城市房屋信息与价格如下：

面积	楼层	...	价格（万元）
95	6	...	125
80	3	...	102
...	...	...	...
125	10	...	158

每条历史数据（每一行），我们称为一个**样本**，每个属性（每一列），我们称为**特征**。每条数据对应的目标输出值，我们称为**标签**。

习惯上，我们使用如下的方式进行表示：

$$\begin{bmatrix} x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, \dots, x_n^{(1)}, y^{(1)} \\ x_1^{(2)}, x_2^{(2)}, x_3^{(2)}, \dots, x_n^{(2)}, y^{(2)} \\ \dots \\ x_1^{(m)}, x_2^{(m)}, x_3^{(m)}, \dots, x_n^{(m)}, y^{(m)} \end{bmatrix}$$

符号	含义
x	特征。
y	标签，即目标输出值（真实值）。
$\hat{y}$	模型的预测值。
x_j	第j个特征。
$x_j^{(i)}$	第i个样本中的第j个特征。
$y^{(i)}$	第i个样本的真实值。
$\hat{y}^{(i)}$	第i个样本的预测值。

实现预测

假设当前具有如下已知的样本数据：

x（特征）	y（标签）
1	2
2	3
3	4
4	5
5	6
6	7
7	?

当x的值为7时，y的值为多少？

A 6

B 7

C 8

D 9

我们不难找出x与y之间的关系，因此，很容易确定模型：

$$y = x + 1$$

这样，当产生新的未知数据时（没有标签的数据），我们就可以使用该模型进行预测了。例如，当新数据$x = 7$到来时，我们可以轻松的预测$y = 8$。

然而，现实中的数据不可能像上例中那么简单，模型也不可能总是通过肉眼就能观察出来的，例如，在鸢尾花数据集中，花瓣长度与花瓣宽度的部分数据如下：

花瓣长度	花瓣宽度
1.3	0.2
1.4	0.3
1.4	0.2
1.6	0.2
3.9	1.1
4.2	1.3
4.5	1.5
4.6	1.5
5.1	1.9
5.7	2.5
4.8	?

对于这样的数据，我们是不容易发现二者之间的关系的。因此，这就需要我们通过机器学习算法来进行建模了。

回归分析

回归分析是用来评估变量之间关系的统计过程。用来解释自变量X与因变量Y的关系。即当自变量X发生改变时，因变量Y会如何发生改变。

线性回归

线性回归，是回归分析的一种，评估自变量X与因变量Y之间是一种线性关系。线性回归可以分为两种：

- 简单线性回归：具有一个自变量。
- 多元线性回归：具有多个自变量。

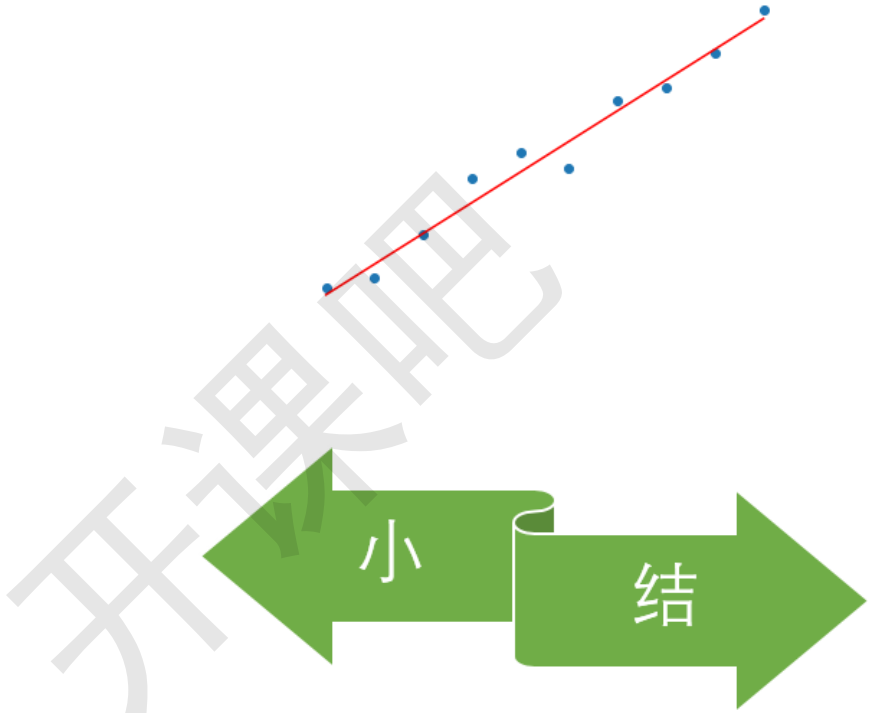
线性关系的理解：

- 画出来的图像是直的。
- 每个自变量的最高次项为1。

以下方程中，y与x哪个是线性关系?
A $y = e^x + 8x$
B $y = x^2 - 6x + 3$
C $y = 8x - 2$
D $y = sinx$

拟合

拟合，是指构建一种算法（数学函数），使得该算法的计算结果能够与真实数据相吻合。从机器学习角度讲，线性回归就是要构建一个线性函数，使得该函数与目标值之间的相符性最好。从空间的角度来看，就是要让函数的直线（面），尽可能靠近空间中所有的数据点（点到直线的平行于y轴的距离之和最短）。线性回归会输出一个连续值。



简单线性回归

模型说明

当线性回归中，仅有一个自变量时，我们称为**简单线性回归。**

这里，我们以房屋面积（x）与房屋价格（y）为例，显而易见，二者是一种线性关系，房屋价格正比于房屋面积，我们假设比例为w：

$$\hat{y} = w * x$$

- $\hat{y}$：预测值。
- w：权重。

然而，这种线性方程一定是过原点的，即当x为0时，y也一定为0。这可能并不符合现实中某些场景。为了能够让方程具有更广泛的适应性，我们这里再增加一个截距，设为w_0，即之前的方程变为：

$$\hat{y} = w_0 + w_1 * x$$

- w_0：截距（偏置），即bias（b）。
- 这里为了统一使用w参数，使用w_0表示b。

而以上方程，就是简单线性回归的模型。方程中的w（w_0与w_1），就是模型的参数。

损失函数

通过之前的介绍，我们得知，对机器学习来讲，就是从已知数据（经验）去建立一个模型，使得该模型能够对未知的数据进行预测。实际上，机器学习的过程，就是确定（学习）模型参数（即模型的权重与偏置）的过程，因为只要模型的参数确定了，我们就可以利用模型进行预测（参数模型）。

那么，模型的参数该如何求解呢？对于监督学习来说，我们可以通过建立损失函数来实现。**损失函数**，也称**目标函数**或**代价函数**，简单的说，就是关于误差的一个函数。损失函数用来衡量模型预测值与真实值之间的差异。机器学习的目标，就是要建立一个损失函数，使得该函数的值最小。

也就是说，损失函数是一个关于模型参数的函数（以模型参数w作为自变量的函数），自变量可能的取值组合通常是无限的，我们的目标，就是要在众多可能的组合中，找到一组最合适的自变量组合（值），使得损失函数的值最小。

损失函数我们习惯使用J来表示，例如，$J(w)$则表示以w为自变量的函数。

参数估计

最小二乘法

$$J(w) = \sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$$
$$= \sum_{i=1}^{m}(y^{(i)} - w_0 - w_1 * x^{(i)})^2$$

- m：样本个数。
- $x^{(i)}$：第i个样本的输入特征值。
- $\hat{y}^{(i)}$：第i个样本的预测值。
- $y^{(i)}$：第i个样本的真实值。

对w_0求偏导

由平方函数的特性可知，该函数是一个凸函数，具有唯一的极值点（极小值），我们只需要对该函数求导（偏导），令导函数为0即可求得参数。

$$\frac{\partial J(w)}{\partial w_0} = 0 \quad \Rightarrow$$
$$\sum_{i=1}^{m} 2 * (y^{(i)} - w_0 - w_1 * x^{(i)}) * (-1) = 0 \quad \Rightarrow$$
$$\sum_{i=1}^{m}(y^{(i)} - w_0 - w_1 * x^{(i)}) = 0 \quad \Rightarrow$$
$$\sum_{i=1}^{m} y^{(i)} - \sum_{i=1}^{m} w_0 - \sum_{i=1}^{m} w_1 * x^{(i)} = 0 \quad \Rightarrow$$
$$\sum_{i=1}^{m} y^{(i)} - m * w_0 - w_1 * \sum_{i=1}^{m} x^{(i)} = 0 \quad \Rightarrow$$
$$m * w_0 = \sum_{i=1}^{m} y^{(i)} - w_1 * \sum_{i=1}^{m} x^{(i)} \quad \Rightarrow$$
$$w_0 = \bar{y} - w_1 * \bar{x} \quad (1)$$

对w_1求偏导

$$\frac{\partial J(w)}{\partial w_1} = 0 \quad \Rightarrow$$
$$\sum_{i=1}^{m} 2 * (y^{(i)} - w_0 - w_1 * x^{(i)}) * (-x^{(i)}) = 0 \quad \Rightarrow$$
$$\sum_{i=1}^{m}(y^{(i)} - w_0 - w_1 * x^{(i)}) * x^{(i)} = 0 \quad (2)$$

我们将（1）式带入（2）中：

$$\sum_{i=1}^{m}(y^{(i)} - \bar{y} + w_1 * \bar{x} - w_1 * x^{(i)}) * x^{(i)} = 0 \quad \Rightarrow$$
$$\sum_{i=1}^{m}(y^{(i)} * x^{(i)} - \bar{y} * x^{(i)} + w_1 * \bar{x} * x^{(i)} - w_1 * (x^{(i)})^2) = 0 \quad \Rightarrow$$
$$\sum_{i=1}^{m}(y^{(i)} * x^{(i)} - \bar{y} * x^{(i)}) + w_1 * \sum_{i=1}^{m}(\bar{x} * x^{(i)} - (x^{(i)})^2) = 0 \quad \Rightarrow$$
$$w_1 = \frac{\sum_{i=1}^{m}(y^{(i)} * x^{(i)} - \bar{y} * x^{(i)})}{\sum_{i=1}^{m}((x^{(i)})^2) - \bar{x} * x^{(i)})} \quad (3)$$

参数化简

$$\sum_{i=1}^{m}(\bar{x} * y^{(i)}) = \bar{x} * m * \frac{1}{m} * \sum_{i=1}^{m} y^{(i)}$$
$$= m * \bar{x} * \bar{y}$$
$$= \sum_{i=1}^{m}(\bar{x} * \bar{y}) \quad (4)$$

$$\sum_{i=1}^{m}(\bar{x} * x^{(i)}) = \bar{x} * m * \frac{1}{m} * \sum_{i=1}^{m} x^{(i)}$$
$$= m * \bar{x} * \bar{x}$$
$$= m * \bar{x}^2$$
$$= \sum_{i=1}^{m} \bar{x}^2 \qquad (5)$$

根据（4）与（5），对（3）式进行调整：

$$w_1 = \frac{\sum_{i=1}^{m}(y^{(i)} * x^{(i)} - \bar{y} * x^{(i)})}{\sum_{i=1}^{m}((x^{(i)})^2 - \bar{x} * x^{(i)})}$$
$$= \frac{\sum_{i=1}^{m}(y^{(i)} * x^{(i)} - \bar{y} * x^{(i)} - \bar{x} * y^{(i)} + \bar{x} * \bar{y})}{\sum_{i=1}^{m}((x^{(i)})^2 - \bar{x} * x^{(i)} - \bar{x} * x^{(i)} + \bar{x}^2)}$$
$$= \frac{\sum_{i=1}^{m}(x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum_{i=1}^{m}(x^{(i)} - \bar{x})^2}$$

示例

波士顿房价数据集

我们以波士顿房价数据集为例，来演示简单线性回归的使用。波士顿房价数据集说明如下：

特征	说明
CRIM	房屋所在镇的犯罪率。
ZN	面积大于25000平方英尺住宅所占比例。
INDUS	房屋所在镇非零售区域所占比例。
CHAS	房屋位于河边，值为1，否则值为0。
NOX	一氧化氮浓度。
RM	平均房间数。
AGE	1940年前建成房屋所占比例。
DIS	房屋距离波士顿五大就业中心的加权距离。
RAD	距离房屋最近公路编号。
TAX	财产税额度。
PTRATIO	房屋所在镇师生比。
B	计算公式为 1000 * （房屋所在镇非美籍人口所在比例 - 0.63）** 2。
LSTAT	弱势群体人口所在比例。
MEDV	房屋的平均价格。

浏览数据

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_boston
import matplotlib.pyplot as plt

plt.rcParams["font.family"] = "SimHei"
plt.rcParams["axes.unicode_minus"] = False
plt.rcParams["font.size"] = 12

# 载入波士顿房价数据集。
boston = load_boston()
# 获取房源数据与目标值（房价）。
X, y = boston.data, boston.target
# 将房源数据X与目标值y拼接，便于整体上显示数据。
df = pd.DataFrame(np.concatenate([X, y.reshape(-1, 1)], axis=1),
        columns=boston.feature_names.tolist() + ["MEDV"])
df.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MI
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36

切分数据集

我们的目的，不应该只是让模型在现有的数据中表现优秀，更重要的是，模型也应该能够适用于未知的数据，即模型在未知数据中，同样能够具有不错的预测能力，我们将这种行为称为模型的泛化能力。

为了能够验证模型的泛化能力，我们需要将现有的数据集分成两个部分：

- 训练集：用于训练模型，求解出最佳参数值（参数估计）。
- 测试集：用于验证模型的泛化能力。

```
from sklearn.model_selection import train_test_split

# 仅使用RM（平均房间数）一个特征，实现简单线性回归。
x, y = df["RM"], df["MEDV"]
# 将数据集划分为训练集与测试集，返回切分之后的数据。
# test_size：测试集大小。
# random_state：随机种子，可用来产生相同的随机数序列。
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
```

关于随机种子，说法正确的是（ ）。【不定项】
A 随机种子相同，产生的随机数一定相同。
B 随机种子不同，产生的随机数一定不同。
C 要产生随机数，就必须要存在随机种子。
D 如果需要设置随机种子，则其值是多少，并不重要，只要记住该值就可以。

参数求解

根据之前推导的公式，来求解最佳参数。

$$w_0 = \bar{y} - w_1 * \bar{x}$$

$$w_1 = \frac{\sum_{i=1}^{m}(x^{(i)} - \bar{x})(y^{(i)} - \bar{y})}{\sum_{i=1}^{m}(x^{(i)} - \bar{x})^2}$$

```
# 计算w1的分子部分。
numerator = np.sum((x_train - x_train.mean()) * (y_train - y_train.mean()))
# 计算w1的分母部分。
denominator = np.sum((x_train - x_train.mean()) ** 2)
w1 = numerator / denominator
w0 = y_train.mean() - w1 * x_train.mean()
print(w1)
print(w0)
```

```
9.312949225629247
-36.18099264633921
```

实现预测

当回归方程确定了，模型也就确定了，一切都迎刃而解。对于未知价格的房屋（使用测试集来模拟），我们就可以通过回归方程（模型）来实现预测。

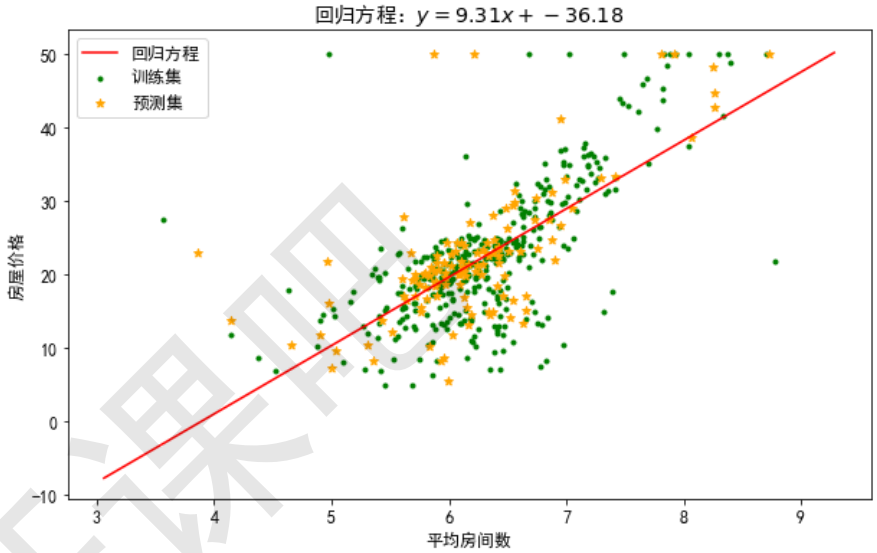
```
y_hat = x_test * w1 + w0
print(y_hat.iloc[:10])
```

```
329    22.797915
371    21.708300
219    23.170433
403    13.633973
78     21.857307
15     18.150753
487    18.811973
340    19.398688
310    10.132304
102    23.468447
Name: RM, dtype: float64
```

可视化

我们绘制训练集，测试集与回归方程，回归方程的y值就是模型的预测值。

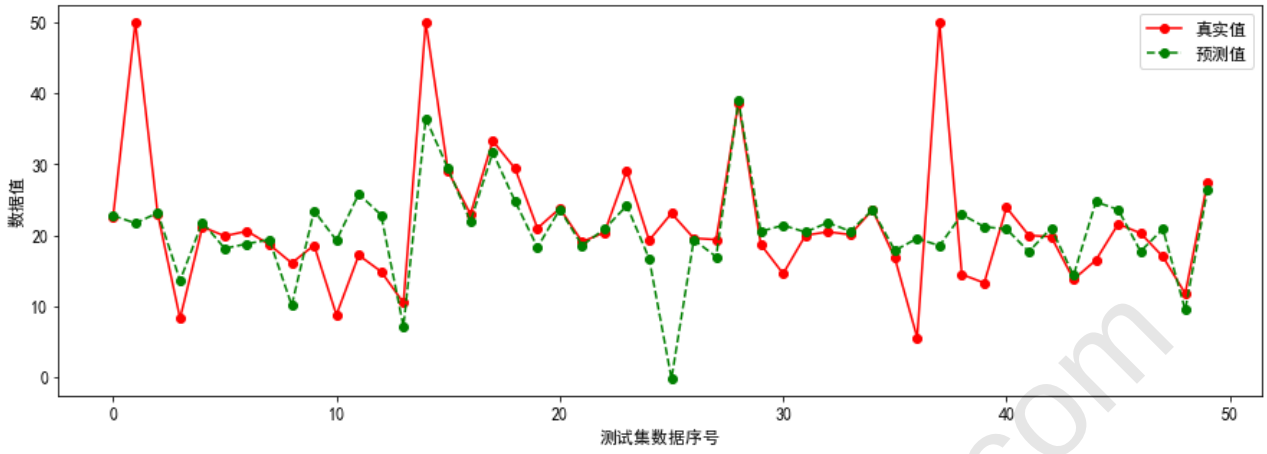
```
plt.figure(figsize=(10, 6))
plt.scatter(x_train, y_train, c="g", marker=".", label="训练集")
plt.scatter(x_test, y_test, c="orange", marker="*", label="预测集")
# 让回归方程x的区间比数据集中x的区间大一些。
t = np.linspace(x_train.min() - 0.5, x_train.max() + 0.5, 100)
t2 = t * w1 + w0
plt.plot(t, t2, "r-", label="回归方程")
plt.legend()
plt.xlabel("平均房间数")
plt.ylabel("房屋价格")
plt.title(f"回归方程：$y = {w1:.2f}x + {w0:.2f}$")
plt.show()
```



我们也可以绘制在测试集上，真实值与预测值的对比，因为测试集数据较多，这里我们只绘制其中的一部分进行比较。

```
num = 50
plt.figure(figsize=(15, 5))
index = np.arange(num)
plt.plot(index, y_test.iloc[:50], label="真实值", color="r", marker="o")
plt.plot(index, y_hat.iloc[:50], label="预测值", ls="--", color="g", marker="o")
plt.xlabel("测试集数据序号")
plt.ylabel("数据值")
plt.legend()
```

```
<matplotlib.legend.Legend at 0x24a8121d648>
```



使用scikit-learn

我们通过简单线性回归的公式，自行计算出模型（直线方程），不过，sklearn中，提供了线性回归的实现，我们可以直接通过该库提供的相关类来实现，这样会更加简便。

```
from sklearn.linear_model import LinearRegression

# 创建线性回归类的对象。
lr = LinearRegression()
# 我们提供数据（x）与数据对应的标签（y），fit方法用来训练（拟合）模型，求解参数。
# 需要注意，在fit方法中，x要求为二维数组类型。
X_train = x_train.values.reshape(-1, 1)
X_test = x_test.values.reshape(-1, 1)
lr.fit(X_train, y_train)
# 当训练模型后，就可以获取参数，即w1与w0。
print("权重：", lr.coef_)
print("偏置：", lr.intercept_)
y_hat = lr.predict(X_test)
print(y_hat[:10])
```

```
权重： [9.31294923]
偏置： -36.180992646339185
[22.7979148  21.70829974 23.17043277 13.63397276 21.85730693 18.15075314
 18.81197253 19.39868833 10.13230385 23.46844714]
```

关于w_1与w_0确定的方程（直线），以下说法正确的是（ ）。【不定项】
A 应该使得所有数据（点）到直线的距离（平行于y轴的距离）平方和最小。
B 线性回归拟合的方程需要穿过空间中的每个点（每条数据）。
C 对于相同的输入数据，方程预测的结果可能是不同的。
D 线性回归模型输出的是连续值。

扩展点

- 随机种子的概念与内幕。

总结

- 数据挖掘的基本概念与应用。
- 简单线性回归及公式推导。
- scikit-learn实现线性回归。

作业

- 在简单线性回归中，如果拟合出的参数w_1值为0，则此时为什么情况?
- 根据家庭电流数据集，使用全局有用功率（x）与全局电流（y）建立简单线性回归。
  - global_active_power：全局有用功率。
  - Global_intensity：全局电流。
  - 注意：数据中存在缺失值（?表示缺失）。