**线性回归实验**

曹可想 2016100104008

1. **线性回归**

对一个来自维空间的数据集，尝试用其中的n维数据预测剩下的一维数据,并假设x与y之间为线性关系。

比如需要通过一个人身高、年龄、性别这三个维度的数据预测体重，在这个例子中x=（身高，年龄，性别），y=体重，并假设x与y之间存在线性关系，那么模型可以表

但是我们知道一个人的体重并不完全由（身高，年龄，性别）这三个因素决定（真实的关系不一定是线性关系），所以无论这个模型的参数如何设置，都不能完美的与真实值匹配，但是我们可以选择一组参数，使预测值与真实值尽可能接近。

下面将这个问题一般化：对一个有m个样本观测数据的数据集,求模型参数.根据“使预测值与真实值尽可能接近”的条件我们有以下表达式：

下面我们只要求出关于的函数取最小值时的值，即可确定倒数。在数据较小情况下，如果可以使用最小二乘法直接求出解析解.当数据量较大时，采取梯度下降法。

1. **梯度下降法**

梯度下降法的迭代公式为：

利用该公式给定一个初始向量代入，求得，然后把作为初始值代入公式重复以上步骤。在实数参数合适的情况下，重复足够次数后将会稳定下来，即可将其作为近似解。

这个公式每一次迭代的的几何意义是从初始值开始，每一次向梯度的反方向（即向最小值方向）前进大小的长度。由图-1可以直观的发现，只要走足够多步，总可以到达‘谷底’（当然的取值必须合适，如果太大，最终将会在‘谷底’附近徘徊，无法足够接近‘谷底’）。

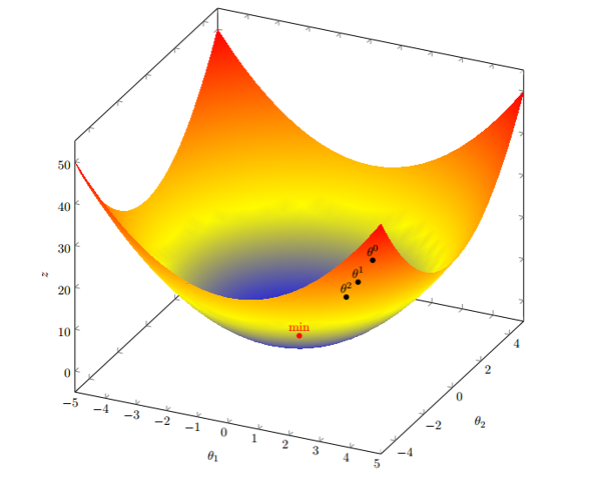


图-1

1. **Python代码及实验结果**

首先展示实验结果。在，循环次数时，测试集的预测结果与实际值结果如图-2.

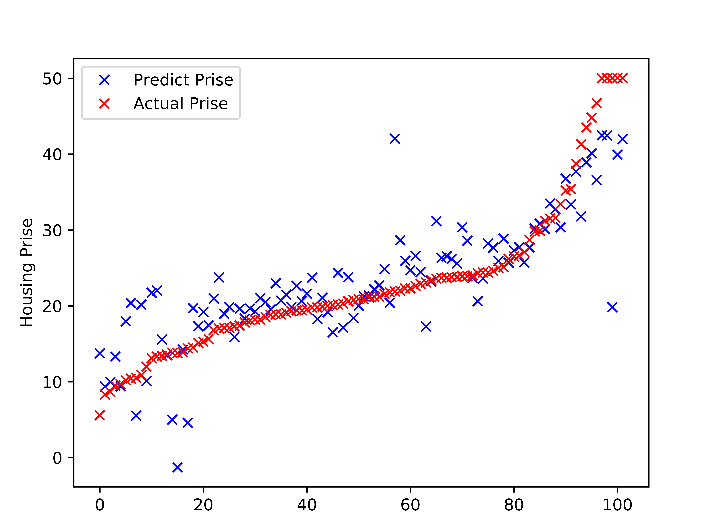
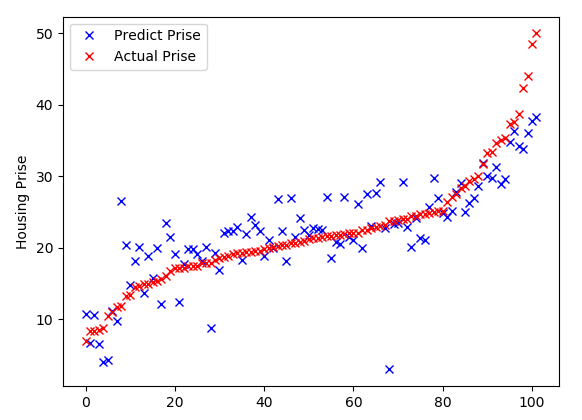


图-2

在，循环次数时，测试集的预测结果与实际值结果如图-3.



从结果的观测上可以看到模型的预测价格存在虚高的情况。

主体代码

|  |
| --- |
| **import** numpy **as** np  **from** sklearn**.**model\_selection **import** train\_test\_split  **from** linear\_regression **import** linear\_reg  **import** matplotlib**.**pyplot **as** plt  **if** \_\_name\_\_ **==** '\_\_main\_\_'**:**  data **=** np**.**loadtxt**(**'housing.data'**)** # 读取数据存储为二维数组，自动去除空格  **(**row**,** col**)** **=** data**.**shape  data **=** np**.**c\_**[**np**.**ones**((**row**,** 1**)),** data**]**  col **+=** 1  # 划分测试集和训练集  **(**X**,** Y**)** **=** **(**data**[:,** **:-**1**],** data**[:,** **-**1**])**  train\_X**,** test\_X**,** train\_Y**,** test\_Y **=** train\_test\_split**(**X**,** Y**,** test\_size**=**0.2**)**  # print(train\_X.shape)  # 切分后的train\_Y,test\_Y默认为一维数组，无法转置，需要将其变为二维数组  train\_Y **=** np**.**array**([**train\_Y**]).**T  test\_Y **=** np**.**array**([**test\_Y**]).**T  # 生成初始的theta  theta **=** np**.**random**.**random**(**size**=(**col**-**1**,** 1**))**  # 梯度下降法求解theta  theta **=** linear\_reg**(**train\_X**,** train\_Y**,** theta**)**  # print(theta)  pre\_Y **=** np**.**dot**(**test\_X**,** theta**)**  # 绘图  com **=** np**.**c\_**[**test\_Y**,** pre\_Y**]**  com **=** com**[**com**[:,** 0**].**argsort**()]**  plt**.**figure**()**  plt**.**plot**(**com**[:,** 1**],** 'bx'**,** label**=**'Predict Prise'**)**  plt**.**plot**(**com**[:,** 0**],** 'rx'**,** label**=**'Actual Prise'**)**  plt**.**ylabel**(**'Housing Prise'**)**  plt**.**legend**()**  plt**.**savefig**(**'Housing\_Prise.png'**,** dpi**=**1000**)**  plt**.**show**()** |

函数代码

|  |
| --- |
| **import** numpy **as** np  **def** linear\_reg**(**X**,** Y**,** theta**):**  m **=** 10000000  alpha **=** 0.000001  X\_X **=** np**.**dot**(**X**.**T**,** X**)**  X\_Y **=** np**.**dot**(**X**.**T**,** Y**)**  **for** i **in** range**(**0**,** m**,** 1**):**  grad **=** np**.**dot**(**X\_X**,** theta**)** **-** X\_Y  theta **=** theta **-** alpha **\*** grad**/**4000  **return** theta |

1. **问题**

在迭代过程中的取值很关键，如果过小需要的迭代次数很长才能得到满意的结果，如果过大就会导致迭代结果趋于正无穷，因此需要测试很多次，希望可以找到选择合适的方法。