# 回归与聚类算法

|  |
| --- |
|  |

# 线性回归

|  |
| --- |
|  |
| 看目标值,如果目标值是连续的,里面这个问题就是回归问题,解决这个问题的算法就是回归算法 |

# 1.线性回归原理

## 1>应用场景

|  |
| --- |
|  |

## 2.>什么是线性回归

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

## 有多个特征的回归叫做多元回归,只有一个特征的回归是单变量回归

## 线性关系和线性模型是不一样,线性关系一定是线性模型,但是线性模型不一定是线性关系

# 2.线性回归的损失和优化原理

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

## 梯度下降是比较常用的方法,正规方程由于比较复杂,所以只能使用在数据量比较小的场景

# 3.线性回归的API模块

|  |
| --- |
|  |

## sklearn.linear\_model.SGDRegressor才是梯度下降的api

# 4.案例:波士顿房价预测

|  |
| --- |
|  |
|  |
|  |
|  |
|  |
|  |
|  |

## 案例编码,我们先使用线性回归LinearRegression方法,注意sklearn1.2以上版本没有load\_boston函数,我们需要使用fetch\_openml和numpy解决问题,这里我们分别使用线性回归和梯度下降来测试,默认情况下是正规方程效果号一点点因为这个例子的数据量只有506个样本,还是比较小的

### 01\_boston-house-price-LinearRegression.py

|  |
| --- |
| """   波士顿房价预测,使用正规方程优化的LinearRegresson算法   sklearn1.2以上版本没有load\_boston方法,需要使用fetch\_openml  """  from sklearn.datasets import fetch\_openml  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import mean\_squared\_error  # 1.获取数据集  data\_x, data\_y = fetch\_openml(name="boston", version=1, as\_frame=True,                                return\_X\_y=True, parser="pandas")  # 2 划分数据集  x\_train,x\_test,y\_train,y\_test = train\_test\_split(data\_x,data\_y,random\_state=22)  # 3.特征工程,标准化  transfer = StandardScaler()  x\_train = transfer.fit\_transform(x\_train)  x\_test  = transfer.transform(x\_test)  print(x\_train)  # 4.预估器流程  estimator = LinearRegression()  estimator.fit(x\_train,y\_train)  # 5. 得出模型  print("正规方程-权重系数为：\n", estimator.coef\_)  print("正规方程-偏置为：\n", estimator.intercept\_)  # 模型评估  y\_predict = estimator.predict(x\_test)  print("正规方程-预测房价：", y\_predict)  error = mean\_squared\_error(y\_test,y\_predict) # 第一个参数是真实值,第二个参数是预测值  print("正规方程-均方误差：\n", error) |

### 效果:

|  |
| --- |
|  |

### 02\_boston-house-price-SGDRegressor.py

|  |
| --- |
| """   波士顿房价预测,使用梯度下降优化的SGDRegressor算法   sklearn1.2以上版本没有load\_boston方法,需要使用fetch\_openml  """  from sklearn.datasets import fetch\_openml  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.linear\_model import SGDRegressor  from sklearn.metrics import mean\_squared\_error  # 1.获取数据集  data\_x, data\_y = fetch\_openml(name="boston", version=1, as\_frame=True,                                return\_X\_y=True, parser="pandas")  # 2 划分数据集  x\_train,x\_test,y\_train,y\_test = train\_test\_split(data\_x,data\_y,random\_state=22)  # 3.特征工程,标准化  transfer = StandardScaler()  x\_train = transfer.fit\_transform(x\_train)  x\_test  = transfer.transform(x\_test)  print(x\_train)  # 4.预估器流程  estimator = SGDRegressor()  estimator.fit(x\_train,y\_train)  # 5. 得出模型  print("梯度下降-权重系数为：\n", estimator.coef\_)  print("梯度下降-偏置为：\n", estimator.intercept\_)  # 模型评估  y\_predict = estimator.predict(x\_test)  print("梯度下降-预测房价：", y\_predict)  error = mean\_squared\_error(y\_test,y\_predict) # 第一个参数是真实值,第二个参数是预测值  print("梯度下降-均方误差：\n", error) |

### 效果

|  |
| --- |
|  |

## 然后我们来给梯度下降调一下参数

### 02\_boston-house-price-SGDRegressor.py

|  |
| --- |
| """   波士顿房价预测,使用梯度下降优化的SGDRegressor算法   sklearn1.2以上版本没有load\_boston方法,需要使用fetch\_openml  """  from sklearn.datasets import fetch\_openml  from sklearn.model\_selection import train\_test\_split  from sklearn.preprocessing import StandardScaler  from sklearn.linear\_model import SGDRegressor  from sklearn.metrics import mean\_squared\_error  # 1.获取数据集  data\_x, data\_y = fetch\_openml(name="boston", version=1, as\_frame=True,                                return\_X\_y=True, parser="pandas")  # 2 划分数据集  x\_train,x\_test,y\_train,y\_test = train\_test\_split(data\_x,data\_y,random\_state=22)  # 3.特征工程,标准化  transfer = StandardScaler()  x\_train = transfer.fit\_transform(x\_train)  x\_test  = transfer.transform(x\_test)  # print(x\_train)  # 4.预估器流程  estimator = SGDRegressor(learning\_rate="constant",eta0=0.001,max\_iter=9000)  estimator.fit(x\_train,y\_train)  # 5. 得出模型  # print("梯度下降-权重系数为：\n", estimator.coef\_)  # print("梯度下降-偏置为：\n", estimator.intercept\_)  # 模型评估  y\_predict = estimator.predict(x\_test)  # print("梯度下降-预测房价：", y\_predict)  error = mean\_squared\_error(y\_test,y\_predict) # 第一个参数是真实值,第二个参数是预测值  print("梯度下降-均方误差：\n", error) |

### 效果就好了那么一点点

|  |
| --- |
|  |

# 5.正规方程和梯度下降对比

|  |
| --- |
|  |
|  |

## 机器学习算法使用图

|  |
| --- |
| Move mouse over image |
|  |

## 其实正规方程用的非常少,梯度下降的通用性比较好,尤其是数据比较大的时候.所以他用的比较多