# 3 线性模型

线性回归是一种有监督的学习算法，旨在采用线性方法来建模因变量和自变量之间的关系。换句话说，它的目标是拟合一条最好地捕捉数据关系的线性趋势线，并且，从这条线，它可以预测目标值可能是什么。

## IMG_256

让我们逐步了解一下线性回归是如何运作的：拟合数据（如上图所示）。

计算点之间的距离（图上的红点是点，绿线是距离），然后求平方，然后求和（这些值是平方的，以确保负值不会产生错误的值并阻碍计算）。这是算法的误差，或者更好地称为残差

存储迭代的残差基于一个优化算法，使得该线稍微“移动”，以便该线可以更好地拟合数据。

重复步骤，直到达到理想的结果，或者剩余误差减小到零。

这种拟合直线的方法称为最小二乘法。

## 3.1 基本形式

给定由d个属性描述的示例x =(x1；x2；……；xd)，其中x¡是æ在第i个属性上的取值，线性模型(linear model)试图学得一个通过属性的线性组合来进行预测的函数，即



一般用向量形式写成



其中w=(wr；w2；…；wd)w和b学得之后，模型就得以确定。

线性模型形式简单、易于建模，但却蕴涵着机器学习中一些重要的基本思想许多功能更为强大的非线性模型(nonlinear model)可在线性模型的基础上通过引入层级结构或高维映射而得此外由于 直观表达了各属性在预测中的重要性因此线性模型有很好的可解释性(comprehensibility)例如若在西瓜问题中学得“”，则意味着可通过综合考虑色泽、根蒂和敲声来判断瓜好不好，其中根蒂最要紧，而敲声比色泽更重要

## 3.2线性回归

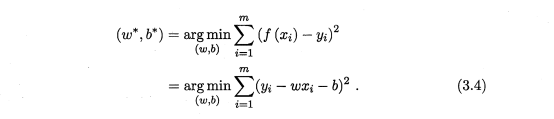
给定数据集其中“线性回归”(linear regression)试图学得一个线性模型以尽可能准确地预测实值输出标记。

我们先考虑一种最简单的情形： 输入属性的数目只有一个为便于讨论，此时我们忽略关于属性的下标，即D={(xi,yi)}其中xi∈R。对离散属性若属性值间存在“序”(order)关系可通过连续化将其转化为连续值，例如二值属性“身高”的取值“高”“矮”可转化为{1.0，0.0}，三值属性“高度”的取值“高”“中”“低”可转化为{1，0，5，0，0}；若属性值间不存在序关系，假定有个K属性值，则通常转化为K维向量例如属性“瓜”的取值“西瓜”“南瓜”“黄瓜”可转化为(0，0，1)，(0，1，0)，(1，0，0))

线性回归试图学得

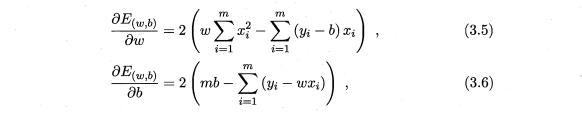


如何确定 w和b呢？显然关键在于如何确定变量X与y之间的别2.3节介绍过，均方误差(2.2)是回归任务中最常用的性能度量，因此我们可试图让均方误差最小化，即

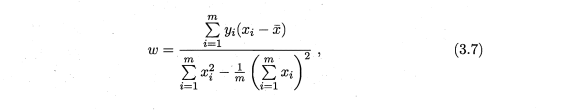


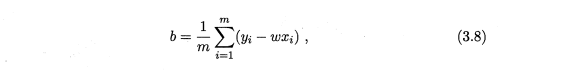
均方误差有非常好的几何意义，它对应了常用的欧几里得距离或简称“欧氏距离”(Euclidean distance)基于均方误差最小化来进行模型求解的方法称为“最小二乘法”(least square method)在线性回归中，最小二乘法就是试图找到一条直线，使所有样本到直线上的欧氏距离之和最小。

求解w和b使最小化的过为线性回归模型的最小二乘“参数估计”(parameter estimation)。 我们可将 E(w)分别对w和求导得到



然后令式(3.5)和(3.6)为零可得到w和最优解的式(closed-form)解



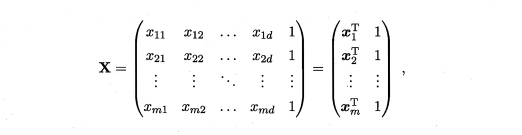


更一般的情形是如本节开头的数据集 D，样本由d个属性描述此时我们试图学得

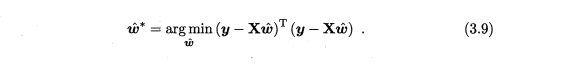


这称为“多元线性回归”(multivariate linear regression)

类似的，可利用最小二乘法来对 w 和进行估计为便于讨论我们把w和b吸收入向形式 =(w；b)，应把据集D为一个m x(d +1)大小的矩阵 X，其中每行对应于一个示例该行前 d个元素对应于示例的d个属性值，最后一个元素恒置为 1，即



再把标记也写成向量形式则类似(3.4)有



令对求导得到



令上式为零可得最优解的闭式解但由于涉及矩阵的计算比单变量情形要复杂一些。下面我们做一个简单的讨论。

当XTX为满秩矩阵(full-rank matrix)或正定矩阵(positive definite matrix)时，令式(3.10)为零可得



其中(XTX)是矩阵(XTX)的逆矩阵令=(1)则最终学得的多元线性回归模型为



然而，现实任务中XTX往往不是满秩矩阵例如在许多任务中我们会遇到大量的变量，其数目甚至超过样例数，导致X的列数多于行数，XTX 显然不满秩。此时可解出多个 ，它们都能使均方误差最小化选择哪一个解作为输出将由学习算法的归纳偏好决定，常见的做法是引入正则化(regularization)项。线性模型虽简单，却有丰富的变化。

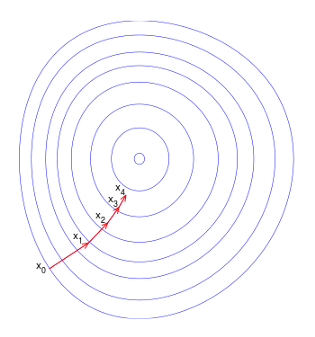
**梯度下降**

在线性回归中，梯度下降被用来更新模型参数，以使拟合直线与观测数据之间的误差最小化。通过计算损失函数对参数的梯度，并沿着梯度的反方向调整参数值，最终找到使损失函数最小化的最佳参数值，从而得到最佳拟合直线。

梯度下降是一种优化算法，用于寻找函数的最小值。在线性回归中，我们使用最小化损失函数来找到最佳拟合直线，而梯度下降是实现这一目标的常用方法之一。其基本思想是沿着损失函数的负梯度方向迭代地调整参数，使损失函数不断减小，直至收敛于最优解或接近最优解。

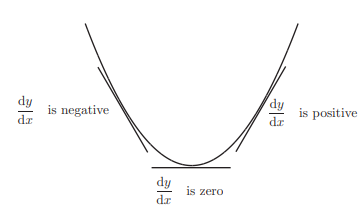


从视觉上看，该算法将执行以下操作：



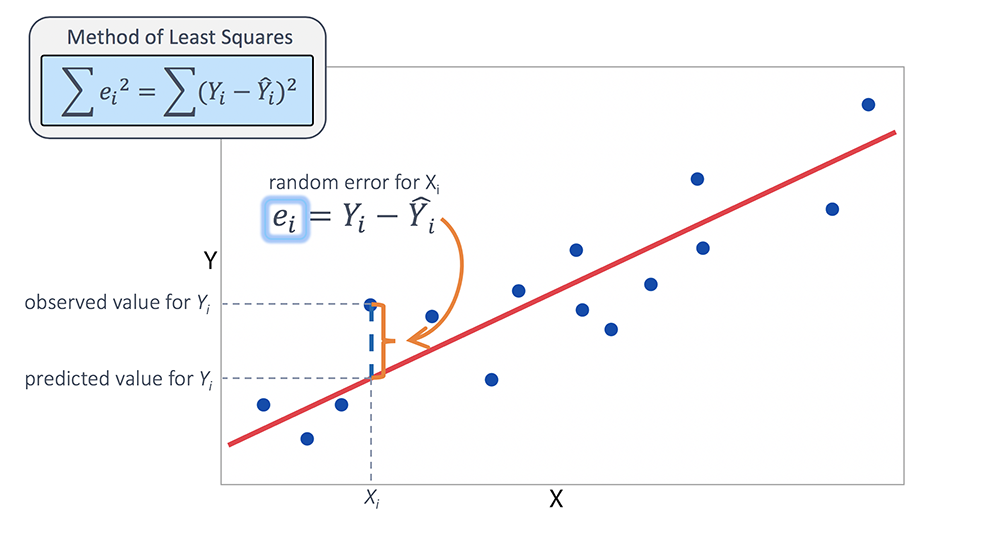
所以，在正常情况下，如图所示，山顶的坡度很高，通过不断的移动，当你到达山脚时的坡度最小，或者接近或等于零。同样的情况在数学上也适用。

梯度下降数学



所以，如果你看到这里的形状和这里的山是一样的。我们假设这是一条形式为y=f（x）的曲线。这里我们知道，任何一点上的斜率都是y对x的导数，如果你用曲线来检查，你会发现，当向下移动时，斜率在尖端或最小位置减小并等于零，当我们再次向上移动时，斜率会增加.

当我们向下移动时，我们会发现y值会减小，所以在这里的所有点中，我们在图的底部得到了相对最小的值。因此，我们的结论是我们总是在图的底部找到最小值（x，y）。



因为线性回归使用直线来预测连续输出.设直线y=w\*x+c

这里我们需要找到w和c，这样我们就得到了使误差最小化的最佳拟合线。所以我们的目标是找到最佳的w和c值

我们从一些随机值开始w和c，我们根据损失更新这些值，也就是说，我们更新这些权重，直到斜率等于或接近于零。

我们将取y轴上的损失函数，x轴上有w和c。

## 3.3 线性模型实现房价预测

线性回归是一种基本的机器学习算法，用于建立变量之间线性关系的模型。以下是一个使用 Python 中的 scikit-learn 库进行线性回归的典型案例，其中使用波士顿房价数据集（Boston House Prices dataset）进行房价预测。

【示例1】：

**数据特征**：数据集中包含了 13 个特征（X），这些特征用来描述影响房价的各种因素，这些特征分别是：

CRIM：城镇人均犯罪率

ZN：住宅用地超过 25,000 平方英尺的比例

INDUS：城镇非零售商业用地比例

CHAS：查尔斯河虚拟变量（边界是河流则为1，否则为0）

NOX：一氧化氮浓度（每千万）

RM：住宅平均房间数

AGE：1940 年以前建成的自住房屋的比例

DIS：到波士顿五个就业中心的加权距离

RAD：辐射性公路的接近性指数

TAX：每 10,000 美元的全值财产税率

PTRATIO：城镇师生比例

B：1000（Bk - 0.63）^ 2，其中 Bk 是城镇黑人的比例

LSTAT：较低地位人口的百分比

目标变量：数据集中还包含了一个目标变量（y），即波士顿地区房屋的中位数价格（单位：千美元）。

|  |
| --- |
| from sklearn.datasets import load\_boston  import pandas as pd  # 加载波士顿房价数据集  boston = load\_boston()  # 将数据集转换为 Pandas DataFrame  boston\_df = pd.DataFrame(boston.data, columns=boston.feature\_names)  # 查看数据集的前几行  print(boston\_df.head())  # 如果你想看到所有列的描述统计信息  print(boston\_df.describe()) |

这段代码将波士顿房价数据集转换为 Pandas DataFrame，然后使用 head() 函数可以查看数据集的前几行，而 describe() 函数可以显示每列的描述性统计信息，如均值、标准差、最小值、最大值等。这能让你更好地理解数据集中各列的值和特征。这个数据集一共有 506 个样本，是一个典型的用于回归问题的数据集。它被广泛用于机器学习和数据科学中的回归任务的练习和演示。

【线性回归示例】

|  |
| --- |
| from sklearn.datasets import load\_boston  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LinearRegression  from sklearn.metrics import mean\_squared\_error, r2\_score  import numpy as np  # 加载波士顿房价数据集  boston = load\_boston()  X = boston.data  y = boston.target  # 将数据集分为训练集和测试集  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  # 初始化线性回归模型  lin\_reg = LinearRegression()  # 在训练集上训练模型  lin\_reg.fit(X\_train, y\_train)  # 在测试集上进行预测  predictions = lin\_reg.predict(X\_test)  # 计算模型性能指标  mse = mean\_squared\_error(y\_test, predictions)  r2 = r2\_score(y\_test, predictions)  print(f"均方误差 (MSE): {mse}")  print(f"R^2 得分: {r2}") |

这个案例使用了波士顿房价数据集，将数据集分为训练集和测试集。然后使用 LinearRegression 类初始化了一个线性回归模型。模型训练后，在测试集上进行了预测，并计算了均方误差（MSE）和 R^2 得分来评估模型的性能。