2

机器学习概论

本章介绍了机器学习中的许多概念，如：特征选择、特征工程、数据清洗、训练集和测试集。

本章第一节简要讨论了机器学习和准备数据集通常需要的步骤。这些步骤包括可以使用各种算法执行的“特征选择”或“特征提取”。

第二节讲述了你可能遇到的数据类型、数据问题以及如何纠正这些问题。当你执行这些步骤时，你还会了解到Holdout检验和K-fold检验的区别。

第三节简要论述了线性回归涉及的基本概念。尽管线性回归是在200多年前发展起来的，但这项技术仍然是解决统计学和机器学习中（尽管很简单）问题的“核心”技术之一。事实上，所谓的“均方误差”（MSE）技术就是用Python和TensorFlow实现的在二维平面（或更高维超平面）中寻找数据点的最佳拟合线，目的是缩小后面将会讨论到的“成本函数”。

本章的第四节为利用Numpy使用标准方法的线性回归示例代码。所以，如果你对这个话题很熟悉，你可以快速浏览本章的前两部分。第三部分展示了如何使用Keras解决线性回归问题。

要注意的一点是，部分算法没有深入研究细节。例如：监督学习部分包含一个算法列表，这些算法将在后面分类算法一节中具体介绍。造列表中以粗体显示的代码与本书关联性更大。部分粗体算法会在下一章中详细介绍。如果没有，你也可以在线搜索有关算法的其他信息，这些内容本书没有详细讨论。

什么是机器学习？

在高级术语中，机器学习是人工智能的一个子集，用来解决用传统语言不能解决的问题或过于繁琐的任务。电子邮件的垃圾邮件过滤器是机器学习的早期例子。通常，机器学习准确性高于传统算法。

尽管有各种各样的机器学习算法，但是数据比选择的算法更重要。数据有可能存在各种问题，例如：数据不足、质量低、数据不正确、数据缺失、不相关的数据、重复数据等等。在本章中，你将会了解到处理这些数据问题的方法。

在机器学习的术语中，数据集是数据值的集合，可以是CSV文件或电子表格；每列被称为一个特性，每一行为包含每个特性的一组特定值，称为数据点。例如一个数据集包含顾客的信息，则每一行都表示一个特定的顾客。

机器学习的类型

你将会遇到三种主要的机器学习类型（也可能是这些类型的组合）：

- 监督学习
- 无监督学习
- 半监督学习

*监督学习*意味着数据集中的数据点有一个标识其内容的标签。例如，MNIST数据集包含28x28个PNG文件。每个文件都包含一个手写的数字(即0-9)。每个数字为0的图像都有标签0；每个数据为1的图像都有标签1；所有的标签都是图片中显示的数字。

另一个例子是，Titanic数据集中每一列是乘客的特征，包括他们的性别、客舱等级、船票价格、是否幸存等等。每一行表示一个乘客的信息，如果乘客幸存则包含值为1,否则为0。MNIST数据集和Titanic数据集设计分类任务：目标是基于训练数据集训练模型，然后在测试集中进行测试。

一般来说，分类可能的标记值为少量的值，例如：0-9数字中的一个、四种动物（狗、猫、马、长颈鹿）之一、两个值中的一个（存活与死亡、购买与未购买）。通常，如果结果的数量可以在较短的下拉列表中显示，那么它可能是一个分类任务。

以包含房地产数据的数据集为例，每一行代表一间房屋的信息，包括卧室数量、房屋面积、浴室数量、房屋价格等等。在这个数据集中，房价为每一行的标签。但这里房价的可选范围过大，不能在较短的下拉列表中显示。*真实的房地产数据通常都包含一个回归任务：目标是基于训练数据集训练一个模型，然后预测测试集中每间房子的价格。*

*无监督学习*包括未标记的数据，典型的例子是聚类算法（之后讨论）。下面列出了一些涉及聚类的重要无监督学习算法：

- K均值
- 层级聚类分析 (HCA)
- 期望最大化

下面列出了一些涉及降维的重要无监督学习算法（之后讨论）：

- 主成分分析（PCA）
- 核主成分分析
- 局部线性嵌入（LLE）
- t分布随机邻居嵌入（t-SNE）

还有一个非常重要的无监督学习任务时异常检测。这个任务与欺诈检测和异常值检测相关（之后会详细讨论）。

*半监督学习*是监督学习和无监督学习的结合：有些数据点被标记，有些数据点没有标记。应用该方法的一种技术是使用标记数据对未标记数据进行分类（即标记），然后可以应用分类算法。

机器学习算法的类型

机器学习算法有主要有三种类型：

- 回归（如线性回归）
- 分类（如k-近邻）
- 聚类（如kMeans）

*回归*是一种用于预测数量的有监督学习技术。回归任务的一个例子是预测特定股票的价值。这个任务不同于预测特定股票第二天的几个会涨还是跌（或在未来的某个时间段）。回归任务的另一个例子是在房地产数据集中预测房屋价格。

机器学习中的回归算法包括线性回归的广义线性回归（在传统统计学中也成为了多元分析）。

分类也是一种有监督的学习技术，但是它是用来预测分类数量的。分类任务的例子有检测垃圾邮件、欺诈或识别PNG文件（如MNIST数据集）中的数字。在本例中，数据已经被标记，因此你可以将预测结果和给定标签进行比对。

机器学习中的分类算法包括以下算法列表（下一章将更详细得讨论这些算法）：

- 决策树（单棵树）
- 随机森林（多棵树）
- k-近邻（kNN）
- 逻辑回归
- 朴素贝叶斯
- 支持向量机（SVM）

一些机器学习算法（如支持向量机、随机森林和k近邻）支持回归和分类。对于支持向量机，scikit-learn库中提供了两个接口：SVC用于分类，SVR用于回归。

前面的每一种算法都涉及到一个模型，该模型是在数据集上训练的，然后使用该模型进行预测。相比之下，随机森林由多个独立的树组成（数量可以决定），每棵树都对一个特征的值进行预测。如果特征为数值型，则取平均值或模型（或其他计算）作为最终预测。如果特征为分类型，则使用最频繁的模式作为最终结果，如果存在频率相同，则随机选择一个。

下面的链接包含关于分类和回归的k-近邻算法的更多信息。

http://saedsayad.com/k_nearest_neighbors_reg.html

*聚类*是一种无监督的学习技术，用于将相似的数据分组在一起。聚类算法可以在不知道数据特性的情况下将数据点放在不同的类簇中。数据被分隔为多个类后，可以使用支持向量机算法进行分类。

机器学习中的聚类算法包括以下几种（其中一些为变形）：

- k-均值
- 均值偏移
- 层次聚类分析（HCA）
- 期望最大化

记住以下几点。首先，K-均值中的k值是一个超参数，它通常是一个奇数，以避免两个类之间的联系。第二，均值偏移算法是k-均值算法的一个变体，它不需要为k指定一个值。事实上，均值偏移算法可以确定最佳的分类数量。然而，对于大数据集，这种算法不能很好地调整。

机器学习任务

除非你有一个已经整理完的数据集，否则你必须要检查数据集中的数据，确保它是合适的正确的。数据准备阶段包括：第一，检查行（“数据清洗”）以确保它们包含有效的数据（这里可能需要特定领域的知识）；第二，检查列（特征选择和特征提取）以确定是否只需要保留最重要的列。

机器学习的任务流程如下所示（不一定所有都需要）：

- 获取数据集
- 数据清洗
- 特征选择
- 降维处理
- 算法选择
- 划定训练集和测试集
- 训练模型
- 测试模型

- 微调模型
- 获取模型的指标

显然，首先需要获取数据集。在理想情况下，该数据集已经存在；否则您需要从一个或多个数据源（例如：csv文件、关系数据库、非sql数据库、web服务器）中挑选数据。

其次，你需要做数据清洗，可以通过以下技术实现：

- 缺失值比例
- 低方差滤波器
- 高方差滤波器

通常，数据清洗涉及检查数据集中的数据值，为了解决以下问题：

- 修正错误值
- 解决重复值
- 解决缺失值
- 确定如何处理异常值

如果数据集缺少的值太多，请使用缺失值比率技术。在某些情况下，你可以删除具有大量缺失值的特征。使用地方插过滤技术从数据集中识别和删除具有常量值的特征。使用高方差过滤技术可以寻找高度相关的特征。这样会增加数据集中不同元素之间的相似性，从而删除部分特征。

例如：可以使用统计值（平均值、模式等）将不正确的值替换为适当的值。重复值可以用类似的方式处理，包括用零、最小值、平均值、最大值等替换掉缺失的数值。

如果数据集中的行包括离群值，则有三种选择：

- 删除行
- 保持这一行
- 将异常值替换为其他值（例如平均值）

当数据集包含异常值时，你需要结合给定数据的领域知识做出决策。

假设数据集包含股票信息。如你所知，1929年有一次股市的崩盘，你可以把这次崩盘看做一个离群值。这种情况很少发生，但是可能包含有意

义的信息。事实上，20世纪一些家庭的财富来源是建立在大萧条时期以非常低的价格购买大量股票的基础上来的。

特征工程、选择和提取

除了创建数据集并“清理”数据值外，你还需要检查该数据集中的特性，以确定时候了以减少数据集地维度（列数）。涉及到三种主要技术：

- 特征工程
- 特征选择
- 特征提取（特征投影）

*特征工程*是对于特定的任务，基于现在的特征的组合确定新的特征组合的过程。这个过程通常需要领域的专业知识，即使在相对简单的数据集的情况下也需要。特征工程是乏味和昂贵的，在某些情况下，你可以考虑使用自动匹配和特征学习。创建数据集后，最好执行选择或特征提取（或两种都执行），以确保拥有高质量的数据集。

*特征选择*也称为变量选择、属性选择或变量子集选择。特征选择实际上就是在数据集中选择相关特征的子集，这些选择的子集是“最重要的特性”。该过程存在以下优点：

- 缩短训练时间
- 更简单的模型更容易解释
- 避免维度过大
- 避免过拟合，泛化效果更好（“方差减少”）

特征选择技术通常用于特征多、样本相对较少的数据。特征可能是冗余的或无关的，这是两个不同的概念。例如：当一个相关特性与另一个强相关特性结合时，它是多余的。

特征选择可以包括三种策略：过滤策略（如信息增益）、包装策略（如精确搜索）和嵌入策略（预测误差用于确定在开发模型时是否包含或排除特征）。此外，特征选择对于重新生成和分类任务也很有用。

*特征提取*是从生成原始特征组合的函数中创建新的特征。相比之下，特征选择涉及到确定现有特征的子集。

特征选择和特征提取都会导致给定数据集的维数降低，这是下一节讨论的内容。

降维

*降维*顾名思义指的是减少数据集中特征的数量的算法。降维包括很多可用的技术，包括特征选择或特征提取。

使用特征选择执行维数缩减的算法如下所示：

- 反向特征消除
- 前向特征选择
- 因子分析
- 独立成分分析

使用特征提取执行降维的算法如下：

- 主成分分析（PCA）
- 非负矩阵分解（NMF）
- 核主成分分析
- 基于图形的核主成分分析
- 线性判别分析（LDA）
- 广义判别分析（GDA）
- 自动编码

这些算法可以在对数据集使用聚类或其他算法（如knn）之前的预处理步骤中使用。

另一组算法涉及基于投影的方法，包括t分布随机邻居嵌入（t-SNE）和UMAP。

本章讨论PCA，你可以通过在线搜索找到更多关于其他算法的信息。

主成分分析（PCA）

主成分是数据集中初始变量的线性组合的新成分。并且，这些成分是不相关的，最有意义或最重要的信息都包含在这些新的成分中。

PCA有两个优点：第一，由于特征数量较少，减少了计算时间；第二，当最多有三个成分时，能够绘制组件图。如果有四个或五个成分，就不能直观地显示，但是可以选择其中的三个子集进行可视化。

PCA使用方差度量信息：方差越大、分量越重要。PCA可以确定协方差矩阵的特征值和特征向量，并构造一个新矩阵，矩阵的每一列为特征向量，根据最左侧的列中的自大特征值从左到右排序，直到最右边的特征向量也具有最小的特征值。

协方差矩阵

首先考虑统计量随机变量方差x，定义如下：

```
variance(x) = [SUM (x - xbar)*(x-xbar)]/n
```

协方差矩阵C是一个nxn的矩阵，主对角线上的值为变量X1，X2，. . .，Xn。矩阵C上的其他值是每对变量Xi和Xj的协方差。

求变量X和变量Y的协方差公式是求方差公式的推广，公式如下所示：

```
covariance(X, Y) = [SUM (x - xbar)*(y-ybar)]/n
```

要注意到，在计算协方差时，X和Y是可交换的（因为乘法是可交换的），所以协方差矩阵C是对称矩阵：

```
covariance(X, Y) = covariance(Y,X)
```

主成分分析法计算了协方差矩阵A的特征值和特征向量。

使用数据集

除了数据清洗之外，你还需要执行其他几个步骤，例如选择训练集和测试集，以及选择在训练过程中是使用“留出法”还是交叉验证。更多的细节在之后的章节中讨论。

训练集与测试集

当你已经完成了本章前面描述的任务（如：数据清洗、降维等），你就可以将数据集分为两个部分。第一部分为训练集，用于训练模型；第二部分为测试集，用于测试（预测、推断）。测试集需要满足以下几个准则：

- 集合足够大，可以产生有统计意义的结果
- 可以代表整个数据集
- 切勿用测试集进行训练
- 测试数据从来没有被测试过

什么是交叉验证

交叉验证的目的是用不重叠的测试集测试模型，其执行方式如下：

- 1：将数据拆分为大小相等的k个子集
- 2：选择一个子集进行测试，其他子集用于训练
- 3：对其他k-1个子集重复步骤2

这个过程称为K折交叉验证，总体误差估计为其平均值。评估的标准方法包括誓词交叉验证。大量的实验表明，10个子集是获得准确估计的最佳选择。实际上吗，你可以重复十次交叉验证的过程，计算平均值，有利于减小方差。

下一节讨论正则化，如果你想要精通机器学习，你需要学习正则化。

什么是正则化

正则化有助于解决过拟合问题，当模型在训练集上表现良好，但在验证集上表现不好时，就会出现这个问题。正则化通过在代价函数中增加一个惩罚因子控制模型的复杂性。

正则化通常用于：

- 大量变量
- 观察项/变量数比例过小
- 高多重共线性

有两种重要的正则化类型：L1正则化（与平均绝对误差或差分的绝对值相关）和L2正则化（与均方误差或差分平方有关）。一般来说，L2的性能比L1好，而且计算是高效的。

机器学习和特征缩放

特征缩放标准化了数据的特征范围。这一步是在数据预处理步骤中执行的。

假设数据符合标准正态分布，标准化包括减去没个数据点的平均值并除以标准差，从而得到N(0,1)的正态分布。

数据规范化与标准化

特征缩放标准化了数据的特征范围。这一步是在数据预处理步骤中执行的。数据规范化是一种线性缩放技术。假设一个数据集有{X1,X2,X3……Xn}以及以下两个量：

```
Minx = minimum of Xi values

Maxx = maximum of Xi values
```

计算一组新的Xi如下所示：

```
Xi = (Xi - Minx)/[Maxx - Minx]
```

这里得到而Xi就是缩放后的结果，它是介于0和1之间的。

偏差-方差权衡

*偏差*是由于学习算法中的错误假设造成的。高偏差可能导致算法欠拟合。预测的偏差可能是由于噪声数据、不完整的特征及或有偏差的训练样本引起的。

偏差导致的错误是模型的预测值与准确值之间的差异。多次重复模型构建过程，每次收集 新的数据，并执行分析并生成新的模型。由于基础数据集具有一定程度的随机性，因此得到的模型具有一系列预测值。偏差可以衡量这些预测值正确的程度。

机器学习中的方差是偏离平均值平方的期望值。高方差可能导致算法在训练数据中对噪声建模，而不是对语气数据建模（也就是过拟合）。

向模型中添加参数会增加模型的复杂性、增加方差并减少偏差。处理偏差和方差就是处理过拟合和欠拟合的过程。

偏差-方差权衡

*偏差*是由于学习算法中的错误假设造成的。高偏差可能导致算法欠拟合。预测的偏差可能是由于噪声数据、不完整的特征及或有偏差的训练样本引起的。

偏差导致的错误是模型的预测值与准确值之间的差异。多次重复模型构建过程，每次收集 新的数据，并执行分析并生成新的模型。由于基础数据集具有一定程度的随机性，因此得到的模型具有一系列预测值。偏差可以衡量这些预测值正确的程度。

度量模型的度量标准

*偏差*是由于学习算法中的错误假设造成的。高偏差可能导致算法欠拟合。预测的偏差可能是由于噪声数据、不完整的特征及或有偏差的训练样本引起的。最常见的度量之一是R平方，它预测数据域拟合回归线的接近成都。R平方值为介于0和1之间的百分数。0表示模型无法解释相应数据在其平均值附近的任何可变性。值100%表示模型解释了平均值附近所有的可能性。一般来说，R平方值越高越好。

R平方的局限性

虽然高R平方值是好的，但是不代表它总是正确的值。同样，低的R平方不是一定代表坏的。例如，预测人类行为的R平方值通常小于50%.此外，R平方值不能确定系数的估计和预测是否存在偏差。才外，R平方值并不表示回归模型是否充分。因此，对于一个好的模型，R平方值可能低也可能高。还要结合残差图、其他模型统计数据和领域知识综合评价。

混淆矩阵

虽然高R平方值是好的，但是不代表它总是正确的值。同样，低的R平方不是一定代表坏的。例如，预测人类行为的R平方值通常小于50%.此外，R平方值不能确定系数的估计和预测是否存在偏差。才外，R平方值并不表示回归模型是否充分。因此，对于一个好的模型，R平方值可能低也可能高。还要结合残差图、其他模型统计数据和领域知识综合评价。

混淆矩阵是一种列联表，具有两行两列，其中包含正确接受、正确拒绝、错误接受、错误拒绝，记录如下：

TP: 正确接受

FP: 错误接受

TN: 正确拒绝

FN: 错误拒绝

混淆矩阵的对角线上的值是正确的，而非对角线上的值是错误的预测。通常，较低的FP值比FN值好。例如，FP表示健康人被错误地诊断为疾病，而FN表示不健康的人被错误地诊断为健康。

准确率、精确率、召回率

2x2混淆矩阵有四个项，它们代表正确和不正确分类的各种组合。根据上一节中的定义，准确率、精确率和召回率的定义由以下公式给出：

```
precision = TP/(TN + FP)
accuracy  = (TP + TN)/[P +N]
recall    = TP/[TP + FN]
```

准确性可能是一个不可靠的指标，因为它会在不平衡的数据集中产生误导性的结果。当不同类别的观测值显著不同时，假阳性和假阴性分类的重要性是相同的。例如：宣布癌症是良性的比误判更糟糕。但是准确度不能区分这两个量。

混淆矩阵不仅仅是2x2的矩阵，还可以是其他nxn的矩阵。例如：如果一个类有5个可能的值，那么混淆矩阵是一个5x5的矩阵。主对角线上的数字是TP的结果。

其他有用的统计术语

机器学习依赖于许多统计量来评估模型的有效性，其中一些统计量如下所示：

- RSS
- TSS
- R^2
- F1 score
- p-value

部分计算式如下：

```
RSS = 残差平方和 (y - y^)**2

TSS = 总平方和(y - y_)**2

R^2 = 1 - RSS/TSS

p = (# of correct positive results)/(# of all
       positive results)

    r = (# of correct positive results)/(# of all
    relevant samples)
    F1-score  = 1/[((1/r) + (1/p))/2]
              = 2*[p*r]/[p+r]
```

什么是线性回归？

线性回归的目标是找到“代表”数据集的最佳拟合线。记住两个关键点。首先，最佳拟合线不一定要通过数据集中的所有（甚至大部分）点。最佳拟合直线的目的是使该直线与数据集中的点之间的垂直距离最小。其次，线性回归并不能确定最佳拟合多项式：后者涉及到找到一个经过数据集中许多点的高次多项式。

此外，平面上的数据集可以包含两个或多个位于同一垂直线上的点，也就是说这些点具有相同的x值。但是，函数不能通过这样一对点：如果两个点（x1，y1）和（x2，y2）具有相同的x值，那么它们必须具有相同的y值（即y1=y2）。另一方面，一个函数可以有两个或多个点位于同一条水平线上。

现在考虑一个散点图，其中平面上有许多点类似于一个细长的云状形状："最佳拟合线"可能只与有限数量的点相交（事实上，最佳拟合线可能不会与任何点相交）。

另一个需要记住的场景是：假设一个数据集包含一组位于同一条线上的点。例如，假设x值在集合{1,2,3，…，10}中，y值在集合中{2,4,6，…，20}。则最佳拟合线的方程为y=2*x+0。在这个场景中，所有的点都是共线的，也就是说它们位于同一条线上。

线性回归与曲线拟合

假设一个数据集由n个形式为（x，y）的数据点组成，并且这些数据点中没有两个具有相同的x值。然后根据一个著名的数学结果，有一个多项式的次数小于或等于n-1，通过这n个点（如果你真的感兴趣，你可以在网上的文章中找到这种说法的数学证明）。例如，一条直线是一次多项式，它可以与平面上任何一对非垂直点相交。对于平面上的任何三重点（不都在同一直线上），有一个二次方程通过这些点。

此外，有时可以使用低次多项式。例如，考虑一组100个点，其中x值等于y值：在这种情况下，直线y=x（它是一次多项式）穿过所有100个点。

但是，请记住，一条直线"代表"平面上的一组点的程度取决于这些点与一条直线的接近程度，直线是用点的方差来衡量的（方差是一个统计量）。点越共线，方差越小；反之，点越"分散"，方差越大。

什么时候解释精确解？

虽然基于统计的解决方案为线性回归提供了闭合形式的解决方案，但神经网络提供了近似的解决方案。这是因为线性回归的机器学习算法包含一系列近似值，这些近似值"收敛"到最优值，这一点意味着机器学习算法产生精确值的估计值。例如，二维平面上一组点的最佳拟合线的斜率m和y截距b在统计学上有一个闭合形式的解，但它们只能通过机器学习算法来近似（确实存在例外，但这种情况很少见）。

请记住，即使"传统"线性回归的闭合形式解提供了m和b的精确值，但有时您只能使用精确值的近似值。例如，假设最佳拟合线的斜率m

等于3的平方根，y轴截距b是2的平方根。如果计划在源代码中使用这些值，则只能使用这两个数字的近似值。对于一个有理性的网络，不管m值是否是精确的，不管m值是否是有理数。然而，机器学习算法更适合于复杂、非线性、多维的数据集，这超出了线性回归的能力。一个简单的例子，假设线性回归问题的闭式解产生m和b的整数值或有理数。具体地说，假设一个闭式解分别产生最佳拟合线的斜率和y截距的值2.0和1.0。直线方程如下：

y=2.0*x+1.0

然而，训练神经网络得到的相应解可能会产生斜率m和y截距b的值分别为2.0001和0.9997，作为最佳拟合线的m和b值。要时刻记住这一点，尤其是在训练神经网络时。

什么是多元分析？

多元分析将欧几里德平面上直线的方程推广到更高的维，它被称为超平面而不是直线。广义方程的形式如下：

y=w1*x1+w2*x2+。+wn*xn+b

在二维线性回归的情况下，你只需要找到斜率（m）和y-截距（b）的值，而在多元分析中，你需要找到w1，w2。，西北。请注意，多元分析是统计学中的一个术语，在机器学习中，它通常被称为“广义线性回归”

请记住，本书中大多数与线性回归相关的代码示例都涉及欧几里得平面中的二维点。

其他类型的回归

线性回归可以找到“代表”数据集的最佳拟合线，但是如果平面中的直线不适合数据集，会发生什么情况？在处理数据集时，这是一个相关的问题。

线性回归的一些替代方法包括二次方程、三次方程或高次多项式。然而，这些替代方案涉及到权衡，我们将在后面讨论。

另一种可能性是一种混合方法，它涉及分段线性函数，它包含一组线段。如果连续线段是连通的，那么它是分段线性连续函数；否则它是分段线性间断函数。

因此，给定平面上的一组点，回归涉及到以下问题：

- 什么样的曲线适合数据？我们怎么知道？
- 另一种曲线是否更适合数据？
- “最适合”是什么意思？

检查一条线是否适合数据的一种方法涉及到视觉检查，但这种方法不适用于高于二维的数据点。此外，这是一个主观的决定，一些样本数据集将在本章后面显示。通过对数据集的可视化检查，您可能会决定二次或三次（甚至更高阶）多项式有可能更适合数据。然而，目视检查可能仅限于二维平面或三维中的点。

让我们推迟非线性场景，假设一条线很适合数据。对于这样一个数据集，有一种众所周知的寻找“最佳拟合”线的技术，它涉及最小化均方误差（MSE），我们将在本章后面讨论。

下一节将快速回顾平面中的线性方程，以及一些图解线性方程示例的图像。

用 NumPy和Matplotlib实现的散点图（1）

2.1显示了np_plot1.py的内容，说明了如何使用Numpy randn（）API生成数据集，然后使用Matplotlib中的scatter（）API来绘制数据集中的点。

需要注意的一个细节是，所有相邻的水平值间距相等，而垂直值基于线性方程加上“扰动”值。在本章的其他代码示例中使用了这种“扰动技术”（这不是一个标准术语），以便在绘制点时添加一个稍微随机化的效果。这项技术的优点是预先知道m和b的最佳拟合值，因此我们不需要猜测它们的值。

例2.1:np_plot1.py

```
import numpy as np
import matplotlib.pyplot as plt

x = np.random.randn(15,1)
y = 2.5*x + 5 +0.2*np.random.randn(15,1)

print("x:",x)
print("y:",y)

plt.scatter(x,y)
plt.show()
```

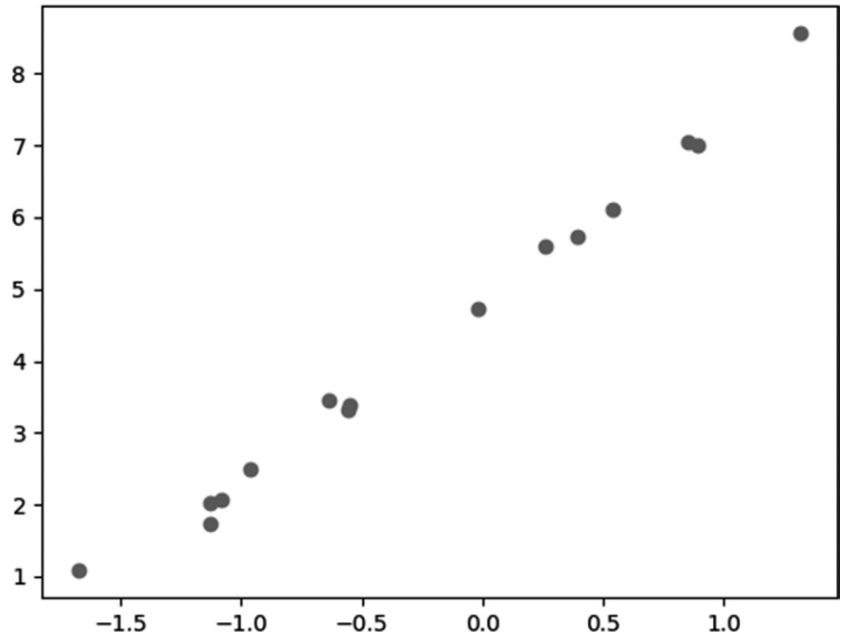
清单2.1包含两个import语句，然后用15个0到1之间的随机数初始化数组变量x. 接下来，数组变量y分两部分定义：第一部分是线性方程2.5*x+5，第二部分是基于随机数的“扰动”值。因此，数组变量y模拟一组近似于线段的值。

该技术用于模拟线段的代码样本中，然后训练部分对最佳拟合直线的m和b值进行近似。显然，我们已经知道最佳拟合线的方程：该技术的目的是将斜率m和y截距b的训练值与已知值（在本例中为2.5和5）进行比较。

例2.1的部分输出如下：

```
x: [[-1.42736308]
 [ 0.09482338]
 [-0.45071331]
 [ 0.19536304]
 [-0.22295205]
 // values omitted for brevity
y: [[1.12530514]
 [5.05168677]
 [3.93320782]
 [5.49760999]
 [4.46994978]
 // values omitted for brevity
```

图2.5显示了基于x和y值的散点图。



扰动技术的作用

您已经了解了如何使用“扰动技术”，并通过比较，考虑一个数据集，其中包含在Python数组变量X和Y中定义的以下点：

```
X =[0,0.12,0.25,0.27,0.38,0.42,0.44,0.55,0.92,1.0]
Y = [0,0.15,0.54,0.51, 0.34,0.1,0.19,0.53,1.0,0.58]
```

如果您需要为前面的数据集找到最佳拟合线，您将如何猜测斜率m和y轴截距b的值？在大多数情况下，您可能无法猜测它们的值。另一方面，“扰动技术”使您能够“抖动”一条直线上的点，其斜率m的值（可选地y轴截距b的值）是预先指定的。

请记住，“扰动技术”只有在引入小随机值时才有效，这些随机值不会导致m和b的不同值。

用 NumPy和Matplotlib实现的散点图（2）

例2.1中的代码将随机值赋给变量x，而硬编码值赋给斜率m。y值是x值的硬编码倍数，加上通过“扰动技术”计算的随机值。因此我们不知道y截距b的值。

在本节中，trainX的值基于np.linspace API，trainY的值涉及上一节中描述的“扰动技术”。

本例中的代码只是打印trainX和对应于欧几里得平面上的数据点。

Listing 2.2: np_plot2.py

```
import numpy as np

trainX = np.linspace(-1, 1, 11)
trainY = 4*trainX + np.random.randn(*trainX.shape)*0.5

print("trainX: ",trainX)
print("trainY: ",trainY)
```

例2.6通过NumPy linspace（）API初始化NumPy数组变量trainX，后由两部分定义的数组变量trainY。第一部分是线性项4*trainX，第二部分涉及“扰动技术”，即随机生成的数字。2.6的输出如下：

```
trainX: [-1. -0.8 -0.6 -0.4 -0.2 0. 0.2  0.4
          0.6  0.8  1. ]
trainY:  [-3.60147459 -2.66593108 -2.26491189
          -1.65121314 -0.56454605  0.22746004
          0.86830728  1.60673482  2.51151543
          3.59573877  3.05506056]
```

下一节包含一个类似于2.2的示例，使用相同的“扰动技术”生成一组近似于二次方程而不是线段的点。

带有numpy 和 matplotlib的二次散射图

例2.3显示了np_plot_quadratic.py这说明了如何在平面上绘制二次函数。

例2.3: np_plot_quadratic.py

```
import numpy as np
import matplotlib.pyplot as plt

#see what happens with this set of values:
#x = np.linspace(-5,5,num=100)

x = np.linspace(-5,5,num=100)[:,None]
y = -0.5 + 2.2*x +0.3*x**2 + 2*np.random.
    randn(100,1)
print("x:",x)

plt.plot(x,y)
plt.show()
```

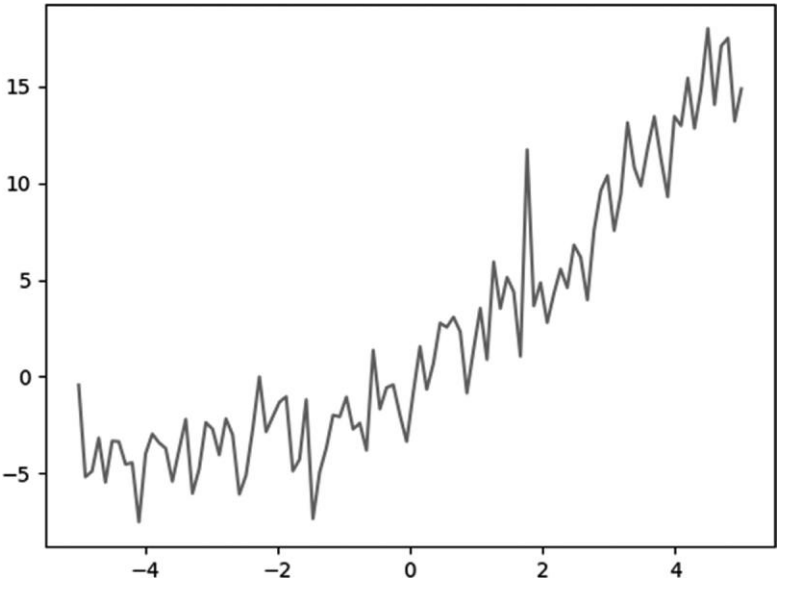
例2.3使用np.linspace()接口，在本例中是一组100个等距的十进制数，介于-5和5之间。主义在初始化x时的代码[:,None]，这会产生一个数组，每个元素都是由一个单个数字组成的。

数组Y分两部分定义：第一部分是二次方程，第二部分是基于随机数的扰动值。因此，数组变量Y模拟了彝族近似二次方程的值。2.3的输出如下：

```
x:
[[-5.        ]
 [-4.8989899 ]
 [-4.7979798 ]
 [-4.6969697 ]
 [-4.5959596 ]
 [-4.49494949]
 // 省略了部分值
 [ 4.8989899 ]
 [ 5.        ]]
```

图2.6 展示了基于x和y值的图，具有二次方程的形状。

FIGURE 2.6 二次方程的散点图



均方误差公式（MSE）

简单地说，MSE是实际y值与预测y值之差的平方和除以点数。请注意，预测的y值是如果每个点实际位于最佳拟合线上，则每个点都将具有的y值。

虽然MSE在线性回归中很流行，但也有其他类型的误差，下一节将简要讨论其中的一些类型。

错误类型列表

除去MSE其他类型的公式也可以用线性回归：

- MSE
- RMSE
- RMSPROP
- MAE

MSE是上述错误类型的基础。例如，RMSE是“均方根误差”，是MSE的平方根。

MAE是“平均绝对误差”，它是y项差值的绝对值之和（不是y项差值的平方），然后除以项数。

RMSProp优化器利用最近渐变的大小来规范化渐变。具体来说，RMSProp在RMS（均方根）梯度上保持一个移动平均值，然后将该项除以当前梯度。

虽然计算MSE的导数更容易，但MSE更容易受到异常值的影响，而MAE则不太容易受到异常值的影响。原因很简单：平方项可以明显大于项的绝对值。例如，如果一个差项是10，那么将100的平方项加到MSE中，而只有10加到MAE中。类似地，如果差项为-20，则平方项400被添加到MSE，而只有20（即-20的绝对值）被添加到MAE。

非线性最小二乘法

在预测房价时，如果数据集包含范围很广的值，线性回归或随机森林等技术会导致模型过度拟合具有最高值的样本，以减少数量，如平均绝对误差。

在这种情况下，您可能需要一个误差度量，例如相对误差，以降低用最大值拟合样本的重要性。这种技术被称为非线性最小二乘法，它可以使用基于对数的标签和预测值的转换。

下一节包含几个代码示例，第一个示例涉及手动计算MSE，然后是一个使用NumPy公式执行计算的示例。

手动计算均方误差

本节包括两个线性回归的图。

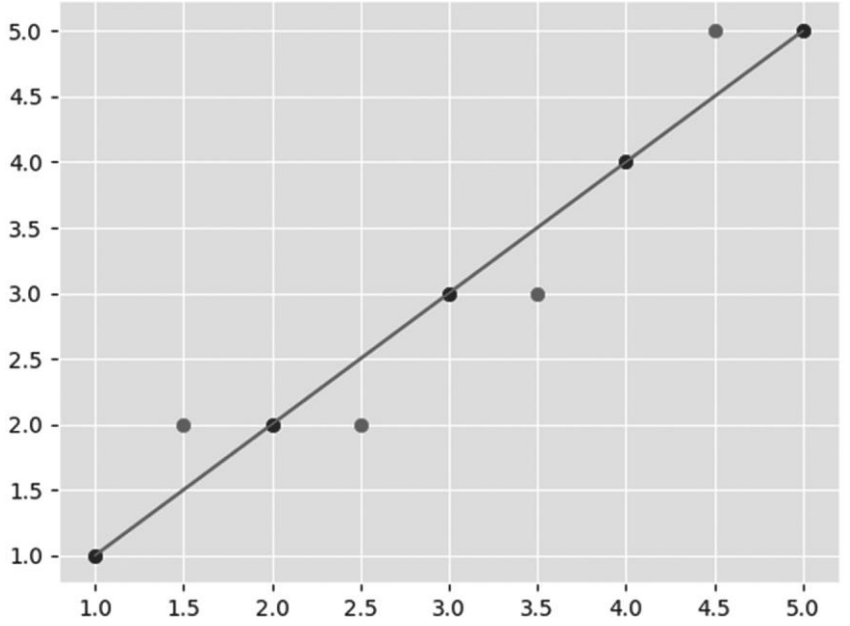


FIGURE 2.7 近似散点图中点的线图

图2.7显示了一个近似散点图的线段。均方误差的计算式子如下：

```
MSE = (1*1 + (-1)*(-1) + (-1)*(-1) + 1*1)/7 = 4/7
```

图2.8均方误差的计算式子如下：

```
MSE = ((-2)*(-2) + 2*2)/7 =8/7
```

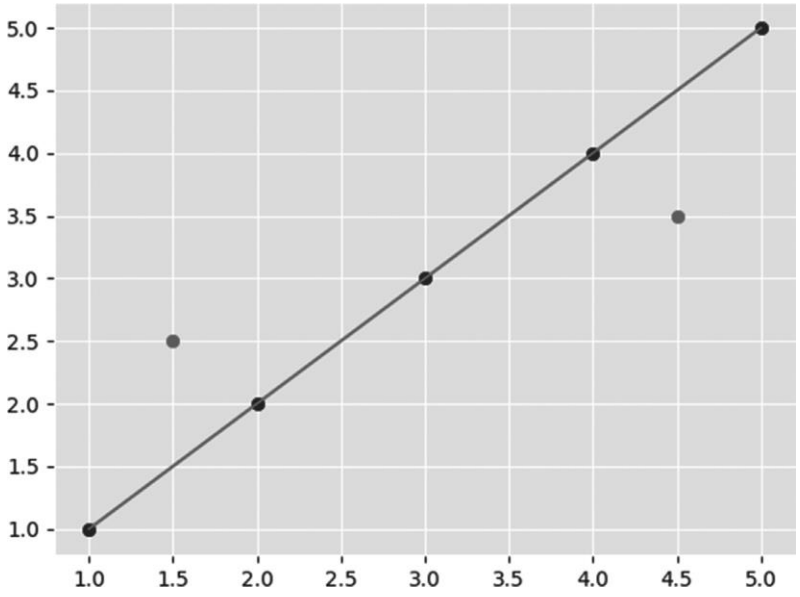


FIGURE 2.8 一种近似散点图点的线图

因此，图2.7中的线比2.8中的线具有更小的均方误差。

在这两个图中，我们计算很快，但是在实际问题中，例如在欧几里得平面上画出10个与直线不紧密吻合的点，计算就会比较复杂。

更好的解决方法为使用numpy库计算，例如调用np.linspace()接口。

np.linspace()处理近似线性数据

例2.4展示了利用np.linspace()API 与扰动技术结合的方法。

Listing 2.4: np_linspace1.py

```
import numpy as np

trainX = np.linspace(-1, 1, 6)
trainY = 3*trainX+ np.random.randn(*trainX.
         shape)*0.5

print("trainX: ", trainX)
print("trainY: ", trainY)
```

这个代码示例的目的仅仅是生成和显示一组随机的数字。在本章后面，我们将使用此代码作为实际线性回归任务的起点。例2.4从数组变量trainx的定义开始，该变量通过np.linspace（）接口。接下来，数组变量trainy是通过扰动技术定义的。2.4的输出如下：

```
trainX:  [-1.  -0.6 -0.2  0.2  0.6  1. ]
trainY:  [-2.9008553  -2.26684745 -0.59516253
          0.66452207  1.82669051  2.30549295]
trainX:  [-1.  -0.6 -0.2  0.2  0.6  1. ]
trainY:  [-2.9008553  -2.26684745 -0.59516253
          0.66452207  1.82669051  2.30549295]
```

既然我们知道了如何为一个线性方程生成（x,y）的值，让我们学习如何计算MSE，这将在下一节中讨论。

利用 np.linspace()计算均方误差

本节中的代码示例不同于本章前面的许多代码示例:它使用了X和Y的硬编码值数组，而不是“干扰”技术。因此，您将不知道斜率和y轴截距的cor- rect值(并且您可能无法猜出它们的正确值)。例2.5显示了plain_linreg1.py的内容，它演示了如何用模拟数据计算MSE。

Listing 2.5: plain_linreg1.py

```
import numpy as np
import matplotlib.pyplot as plt

X = [0,0.12,0.25,0.27,0.38,0.42,0.44,0.55,0.92,1.0]
Y = [0,0.15,0.54,0.51,
   0.34,0.1,0.19,0.53,1.0,0.58]

costs = []
#Step 1: Parameter initialization
W = 0.45
b = 0.75

for i in range(1, 100):
  #Step 2: Calculate Cost
  Y_pred = np.multiply(W, X) + b
  Loss_error = 0.5 * (Y_pred - Y)**2
  cost = np.sum(Loss_error)/10

  #Step 3: Calculate dW and db
  db = np.sum((Y_pred - Y))
  dw = np.dot((Y_pred - Y), X)
  costs.append(cost)

  #Step 4: Update parameters:
  W = W - 0.01*dw
  b = b - 0.01*db

  if i%10 == 0:
    print("Cost at", i,"iteration = ", cost)
```

(Continued)

```
#Step 5: Repeat via a for loop with 1000 iterations

#Plot cost versus # of iterations
print("W = ", W,"& b = ",  b)
plt.plot(costs)
plt.ylabel('cost')
plt.xlabel('iterations (per tens)')
plt.show()
```

例2.5用硬编码的值初始化数组变量X和Y，然后初始化标量变量W和b。每次循环迭代后，计算变量Y_pred、Loss_error和cost。接下来，根据数组Y_pred-Y中的项之和以及Y_pred-Y和X的内积，计算dw和db的值。

注意W和b是如何更新的：它们的值分别按0.01*dw和0.01*db递减。计算W和b梯度的近似值，这两个值都乘以学习率（硬编码值0.01），并从W和b的当前值中删除得到的项，以便生成W和b的新近似值。虽然这项技术非常简单，但它确实能计算出W和b的合理值。

例2.5中的最后一段代码显示了W和b的中间方法，以及成本（纵轴）与迭代次数（横轴）的关系图。2.5的输出如下：

```
Cost at 10 iteration =  0.04114630674619492
Cost at 20 iteration =  0.026706242729839392
Cost at 30 iteration =  0.024738889446900423
Cost at 40 iteration =  0.023850565034634254
Cost at 50 iteration =  0.0231499048706651
Cost at 60 iteration =  0.02255361434242207
Cost at 70 iteration =  0.0220425055291673
Cost at 80 iteration =  0.021604128492245713
Cost at 90 iteration =  0.021228111750568435
W = 0.47256473531193927 & b = 0.19578262688662174
```

图2.9 显示了例2.5的结果。

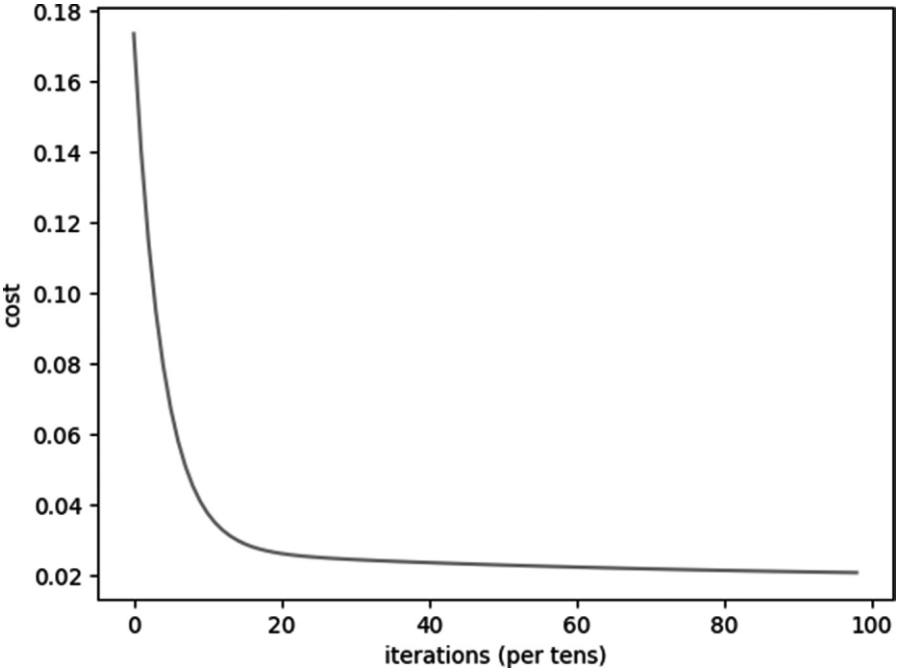


FIGURE 2.9 线性回归的均方误差

代码 plain_linreg2.py和例2.5类似：不同的地方是这个代码需要让外循环执行100次，在迭代过程中，内循环也执行100次。

运用keras进行线性回归

本节中的代码示例主要包含Keras代码，用于执行线性回归。如果您已经阅读了本章前面的示例，那么本节将更容易理解，因为线性回归的步骤是相同的。

例2.6keras_linear_regression.py说明了如何在Keras中执行线性回归。

Listing 2.6: keras_linear_regression.py

```
##################################################
   ###############
#Keep in mind the following important points:
#1) Always standardize both input features and
    target variable:
#doing so only on input feature produces incorrect
   predictions
#2) Data might not be normally distributed: check
    the data and
#based on the distribution apply StandardScaler,
   MinMaxScaler,
#Normalizer or RobustScaler
##################################################
   ###############

import tensorflow as tf
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_
   split

df = pd.read_csv('housing.csv')
X  = df.iloc[:,0:13]
y  = df.iloc[:,13].values

mmsc = MinMaxScaler()
X  = mmsc.fit_transform(X)
y  = y.reshape(-1,1)
y  = mmsc.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_
   split(X, y, test_size=0.3)
```

```
# this Python method creates a Keras model
def build_keras_model():
  model = tf.keras.models.Sequential()
  model.add(tf.keras.layers.Dense(units=13,
    input_dim=13))
  model.add(tf.keras.layers.Dense(units=1))

  model.compile(optimizer='adam',loss='mean_
    squared_error',metrics=['mae','accuracy'])
  return model

batch_size=32
epochs = 40

# specify the Python method 'build_keras_model'
   to create a Keras model
# using the implementation of the scikit-learn
   regressor API for Keras
model = tf.keras.wrappers.scikit_learn.
        KerasRegressor(build_fn=build_
        keras_model, batch_size=batch_
        size,epochs=epochs)

# train ('fit') the model and then make
   predictions:
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
#print("y_test:",y_test)
#print("y_pred:",y_pred)

# scatter plot of test values-vs-predictions
fig, ax = plt.subplots()
ax.scatter(y_test, y_pred)
ax.plot([y_test.min(), y_test.max()], [y_test.
   min(), y_test.max()], 'r*--')
ax.set_xlabel('Calculated')
ax.set_ylabel('Predictions')
plt.show()
```

例2.6从多个import语句开始，然后用CSV文件的内容初始化housing.csv.注意，训练集X是用数据集前13列的内容初始化的向量，变量y包含数据集最右边的列。

下一部分使用MinMaxScaler类计算平均值和标准偏差，然后调用fit_trans-form（）方法更新X值和y值，使它们的平均值为0，标准偏差为1。

接下来，build_keras_mode函数创建keras的模型，其中包含两个全连接层。请注意，输入层的大小为13，即数据帧X中的列数。下一个代码片段使用adam优化器，MSE loss函数编译模型，并指定度量的MAE和精度。然后将编译后的模型返回给调用者。

下一部分将batch_size变量初始化为32，epochs变量初始化为40，并在创建模型的代码片段中指定它们，如下所示：

```
model =
tf.keras.wrappers.scikit_learn.
   KerasRegressor(build_fn=build_keras_model,
   batch_size=batch_size,epochs=epochs)
```

例2.6的下一部分调用fit()函数来训练模型，然后对X_test测试数据调用predict()函数计算预测值，并用这些预测值初始化变量y_pred.

最后一部分为绘制散点图，其中水平轴是test的值（来自csv文件housing.csv）纵轴是一组预测值。

图2.5展示了基于测试值和预测值的散点图。

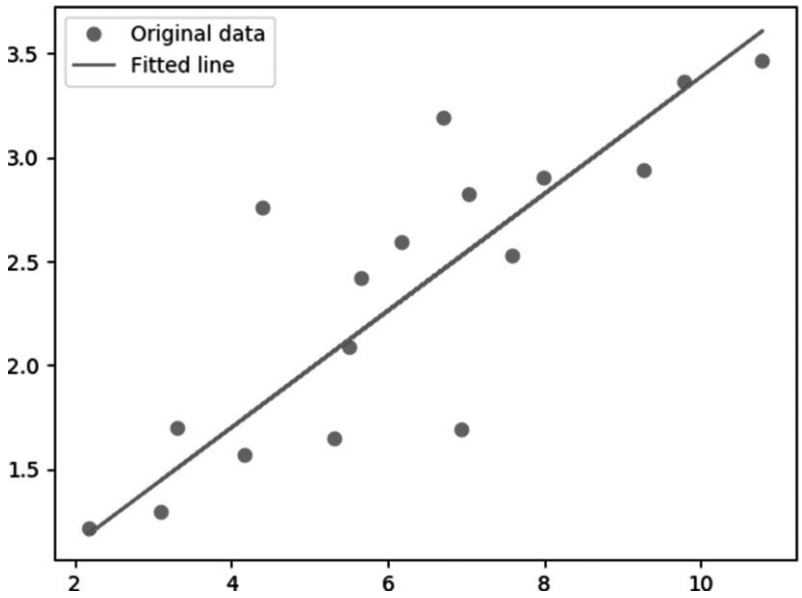


图 2.10 散点图和最佳拟合线

2.7展示了csv文件的前四行。

Listing 2.7: housing.csv

```
0.00632,18,2.31,0,0.538,6.575,65.2,4.09,1,296,15.3,
   396.9,4.98,24
0.02731,0,7.07,0,0.469,6.421,78.9,4.9671,2,242,17
   .8,396.9,9.14,21.6
0.02729,0,7.07,0,0.469,7.185,61.1,4.9671,2,242,17
   .8,392.83,4.03,34.7
0.03237,0,2.18,0,0.458,6.998,45.8,6.0622,3,222,18
```

总结

本章向您介绍了机器学习和诸如特征选择、特征工程、数据清理、训练集和测试集等概念。接下来你学习了有监督、无监督和半监督学习。然后，您学习了回归任务、分类任务和聚类，以及准备数据集通常需要的步骤。这些步骤包括可以使用各种算法执行的“特征选择”或“特征提取”。然后，您了解了数据集中的数据可能出现的问题，以及如何纠正这些问题。

此外，您还学习了线性回归，并简要介绍了如何为欧几里得平面中的值数据集计算最佳拟合线。您看到了如何使用NumPy执行线性回归，以便用数据值初始化数组，以及引入一些y值随机性的“扰动”技术。此技术很有用，因为您将知道最佳拟合线的坡度和y轴截距的正确值，然后可以将其与训练的值进行比较。

然后，您学习了如何在涉及Keras的代码示例中执行线性回归。此外，您还了解了如何使用Matplotlib来显示最佳拟合的线图，以及在与训练相关的代码块中显示成本与迭代次数的关系图。