目录

CHAPTER 2

机器学习的简介

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。

本章的第一部分简要讨论了机器学习和准备数据集通常需要的各种步骤。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"。

第二部分介绍可能遇到的数据类型、数据集中数据可能出现的问题以及如何纠正它们。执行训练步骤时，您还将了解"留出法（hold out）"和"k-交叉验证（k-fold）"的区别。

本章的第三部分简要讨论了线性回归所涉及的基本概念。尽管线性回归是在 200 多年前发展起来的，但该技术仍然是解决（尽管很简单）统计学和机器学习问题的"核心"技术之一。事实上，在 Python 和 TensorFlow 中实现了称为"均方误差"（MSE）的技术，用于为 2D 平面（或更高尺寸的超平面）上的数据点查找最佳拟合线，目的是最大限度地减少稍后讨论的"成本"函数（cost）。

本章的第四部分包含其他代码示例，这些代码示例涉及使用 NumPy 中的标准技术进行线性回归任务。因此，如果您感到本主题很容易，则可能快速浏览本章的前两部分。第三部分演示如何使用 Keras 求解线性回归问题。

需要记住的是，我们只是提及了一些算法，而没有详细研究它们。例如，与监督学习相关的部分包含了一些算法，这些算法出现在后面与分类算法相关的章节中。列表中以粗体显示的算法是本书更为关注的算法。在某些情况下，这些算法将在下一章中更详细地讨论；或者，您可以在网上搜索有关本书中未详细讨论的算法的其他信息。

什么是机器学习？

在高级术语中，机器学习是 AI 的子集，可以解决使用"传统"编程语言不可行或过于繁琐的任务。电子邮件的垃圾邮件筛选器是机器学习的早期示例。机器学习可以大大提升准确性。

尽管机器学习算法种类繁多，但数据比所选算法更重要。数据可能会出现许多问题，例如数据不足、数据质量差、数据不正确、数据丢失、不相关数据、重复数据值等等。本章的稍后部分会介绍解决这些与数据相关的问题的技术。

如果您不熟悉机器学习术语，数据集是数据值的集合，可以是 CSV 文件或电子表格的形式。每列称为特征，每一行都是包含每个特征的特定值的数据点。如果数据集包含有关客户的信息，则每行都是一个单独的用户。

机器学习的类型

您将遇到三种主要类型的机器学习（也可能是这些的组合）：

- 监督学习
- 无监督学习

- 半监督式学习

监督学习意味着数据集中的数据点具有标识其内容的标签。例如，MNIST 数据集包含 28x28 的 PNG 文件，每个文件都包含单个手绘数字（即 0 到 9）。每个带数字 0 的图像都标有 0;每个数字为 1 的图像都有标签 1;所有其他图像都根据其图像中显示的数字进行标记。

另一个例子是，Titanic 数据集中的列是乘客的特征，例如乘客的性别、客舱等级、船票价格、乘客是否幸存下来等等。每行都包含一个乘客的信息，包括乘客幸存下来时的 1 标签。**MNIST 数据集和 Titanic 数据集涉及分类任务：目标是基于训练数据集（training dataset）训练模型，然后预测测试数据集（test dataset）中每一行的标签。**

通常，分类任务的数据集具有少量的可能值：MNIST 数据集是 0 到 9 范围内的 9 位数字之一，四种动物（狗、猫、马、长颈鹿）中的一个，两个值之一（生存与死亡、购买与未购买）。根据经验，如果结果的数量可以在下拉列表中"合理地"显示，那么它可能是一个分类任务。

对于包含房地产数据的数据集，每行都包含一个特定房屋的信息，例如卧室数量、房屋的平方英尺、浴室数量、房屋价格等。在此数据集中，房屋的价格是每一行的标签。请注意，可能的价格范围太大，无法在下拉列表中"合理地"展示。 **房地产数据集涉及回归任务：目标是基于训练数据集训练模型，然后预测测试数据集中每个房屋的价格。**

无监督学习包括未标记的数据，这是使用聚类算法（稍后讨论）的通常情况。下面列出了一些涉及聚类的重要的无监督学习算法：

- k-Means
- 层次聚类算法（HCA）(Hierarchical Cluster Analysis)

- 期望最大化（EM）（Expectation Maximization）

下面列出了一些涉及降维（dimensionality reduction）的重要无监督学习算法（稍后将详细讨论）：

- PCA（主要成分分析）（Principal Component Analysis）
- 核 PCA
- LLE（局部线性嵌入）（Locally Linear Embedding）
- t-SNE（t-distributed Stochastic Neighbor Embedding）

还有一个非常重要的无监督任务称为异常检测。此任务与欺诈检测和检测异常值相关（稍后将详细讨论）。

半监督学习是监督学习和无监督学习的组合：有些数据点被标记，有些数据点没有标记。一种技术涉及使用标记数据来对未标记的数据进行分类（也就是打上标签），之后就可以应用分类算法。

机器学习算法的类型

机器学习算法有三种主要类型：

- 回归（例如：线性回归）
- 分类（例如：k-近邻）
- 聚类（例如：k-Means）

回归是一种用于预测数值量的监督学习技术。比如预测特定股票的值。请注意，此任务不同于预测明天（或其他一些未来时间段）上特定的股票票价是否会增加或减少。回归任务的另一个示例涉及预测房地产数据集中房屋的价格。以上都是回归任务的举例。

机器学习中的回归算法包括线性回归和通用线性回归（在传统统计学中也称为多变量分析）。

分类也是一种监督学习技术，但它用于预测分类变量（就是种类）。分类任务的一个示例是检测垃圾邮件、欺诈或确定 PNG 文件中的数字（如 MNIST 数据集）。在这种情况下，数据已经被标记，因此您可以将预测的标签与 PNG 已有的标签进行比较。

机器学习中的分类算法包括以下算法列表（下一章将更详细地讨论这些算法）：

- 决策树（一个树）
- 随机森林（多棵树）
- kNN （k 最近邻居）
- 逻辑回归
- 朴素贝叶斯
- SVM（支持向量机）

某些机器学习算法（如 SVM、随机森林和 kNN）同时支持回归和分类。对于 SVM，scikit-learn 提供了两个 API：用于分类的 SVC 和用于回归的 SVR。

上述每个算法都涉及一个在数据集上训练的模型，之后该模型用于进行预测。相比之下，随机森林由多个独立的树组成（数字由您指定），每颗树对一个特征的值进行预测。如果特征为数值型，请采用均值或求众数（或执行一些其他计算），以确定"最终"预测。如果该特征是分类型，则使用众数（即最常出现的类别）作为结果。如果有多个可能的结果，可以随机选择一个。

以下链接包含有关 kNN 算法分类和回归的信息：
http://saedsayad.com/k_nearest_neighbors_reg.htm

聚类属于无监督学习，用于将相似的数据分组在一起。聚类算法在不知道数据点的性质的情况下将数据点放在不同的簇（cluster）中。数据被分成不同的簇后，可以使用 SVM（支持向量机）算法执行分类。

机器学习中的聚类算法包括以下内容（其中一些算法是其他的变体）：

- k-means
- MeanShift（基于核密度估计的爬山算法）
- 层次聚类 （HCA）
- 期望最大化（EM）

请记住以下几点。首先，k-Means 中的 k 值是一个超参数（人为设定），通常为奇数来避免产生两个相同的类别。meanshift 算法是 k-Means 算法的变种，它不需要为 k 指定一个值。事实上，meanshift 算法决定了簇的最佳数量。但是，此算法对于大型数据集的扩展性能不好。

机器学习的任务

除非数据集已经被清洗过，否则需要检查数据集中的数据，以确保数据处于合适的状态。数据准备阶段涉及 1）检查行（"数据清洗"），以确保它们包含有效数据（可能需要域特定知识），2）检查列（特征选择或特征提取），以确定是否保留了最重要的列。

一个高级机器学习任务的执行顺序（其中一些可能不需要）如下所示：

- 获取数据集
- 数据清洗

- 特征选择
- 降维
- 算法选择
- 分离训练与测试数据
- 训练模型
- 测试模型
- 微调模型
- 获取模型的指标

首先需要取数据集。在理想情况下，此数据集已存在；否则，则需要从一个或多个数据源（例如，CSV 文件、关系数据库、非关系型数据库、Web 服务等）中获取数据。

其次，您需要执行数据清，您可以通过以下技术执行：

- 缺失值比率
- 低方差过滤器
- 高相关性过滤器

通常，数据清理需要检查数据集中的数据值，以便解决以下一个或多个问题：

- 修复不正确的值
- 解决重复值
- 解决缺失值
- 决定怎么处理异常值

如果数据集的缺失值很多，可以使用"缺失值比率（Missing Value Ratio）"技术。在极端情况下，可以删除具有大量缺失值的特征。使用"低方差滤波器（Low Variance filter）"筛选技术标识和删除那些是常量的特征。使用高相关性筛选（High Correlation filter）技术查找高度相关的特征，这会增加数据集的多重共线性（multicollinearity）：此类特征可以从数据集中删除（但在这样做之前请咨询领域专家）。

根据您的背景和数据集的性质，您可能需要与非常了解数据集内容的领域专家合作。。

例如，可以使用统计量（均值、众数等）将不正确的值替换为合适的值。重复值可以以类似的方式处理。您可以将缺失值替换为数字零、或一列中的最小值、平均值、众数，最大值。您可以将缺少的类别数据替换为类别列的众数。

如果数据集中的行包含一个异常值，则有三种选择：

- 删除行
- 保留该行
    - 将异常值替换为其他值（均值？）

当数据集包含异常值时，您需要根据数据集的领域知识做出决定。

假设数据集包含与股票相关的信息。如您所了解，1929 年有一次股市崩盘，您可以视之为离群值。这种情况很少见，但它可以包含有意义的信息。顺便说一句，20 世纪一些家庭的财富来源是基于在大萧条时期购买大量价格非常低的股票。

特征工程、选择和提取

除了创建数据集并"清理"其值外，还需要检查该数据集中的特征，以确定是否可以减少数据集的维数（即列数）。此过程涉及三种主要技术：

- 特征工程
- 特征选择
- 特征提取（也称为特征投影）

特征工程是确定基于现有特征组合产生新特征集的过程，目的是任务创建更有意义的数据集。即使在相对简单的数据集的情况下，此过程也需要领域专业知识。特征工程可能既繁琐又昂贵，在某些情况下 ， 您可能会考虑使用自动特征学习（automated feature learning）。创建数据集后，建议执行特征选择或特征提取（或两者），以确保具有 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。特征选择包括选择数据集中相关特征的子集。从本质上讲，特征选择主要是选择数据集中的"最重要"特征，有以下优势：

- 缩短训练时间
- 更简单的模型更易于解释
- 避免维度的诅咒（高维）
- 由于减少了过度拟合而实现更好的泛化（"减少变化幅度"）

特征选择技术通常用于特征众多且样本相对较少（或数据点）的领域。请记住， 没有价值的特征是冗余的或不相关的，这是两个不同的概念。例如，一个相关特征与其他强相关特征结合使用时，它可能就是冗余的。

特征选择可以包含三种策略：筛选策略（例如信息增益）、包装策略（例如，以准确性为指导的搜索）和嵌入式策略（在开发模型时用预测值得错误判断是否包括或排除一些特征）。另一个有趣的观点是，特征选择对于回归和分类任务也很有用。

特征提取从生成原始特征组合的函数创建新特征。相比之下，特征选择是指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度减少，这将是下一节的主题。

降维

降维是指减少数据集中特征数量的算法：因此称为"降维"。如您所看到的，有许多可用的技术，它们涉及特征选择或特征提取。

此处列出了使用特征选择执行降维的算法：

- 反向特征消除
- 正向特征选择
- 因素分析
- 独立组件分析

此处列出了使用特征提取来执行维数缩减的算法：

- 主要成分分析（PCA）
- 非负矩阵分解 （NMF）
- 核 PCA
- 基于图形的核 PCA
- 线性判别分析 （LDA）

- 广义判别分析
- 自编码器（Autoencoder）

以下算法结合了特征提取和降维：

- 主要成分分析
- 线性判别分析 （LDA）
- 规范相关性分析 （CCA）
- 非负矩阵分解 （NMF）

这些算法可以在预处理步骤中使用，然后再在数据集上使用聚类或其他算法（如 kNN）。

另一组算法涉及基于投影的方法，其中包括 t-Distributed Stochastic Neighbor Embedding (t-SNE)以及 UMAP。

本章讨论 PCA，您可以上网查找有关其他算法的更多信息。

PCA

主成成分是作为数据集中初始变量的线性组合的新成分。此外，这些成分不相关，最有意义或最重要的信息包含在这些新成分中。

PCA 有两个优点：1）由于特征少得多而缩短计算时间；2）当最多有三个组件时，能够绘制图。如果您有四个或五个组件，您将无法直观地显示它们，但可以选择三个组件的子集进行可视化，并且可能会获得对数据集的一些额外见解。

PCA 使用方差作为信息度量：方差越高，成分越重要。事实上，只是稍微向前跳跃：PCA 确定协方差矩阵的特征值和特征向量（稍后讨论），并构造一个新矩阵，其列是特征向量，根据最左侧

列中的最大特征值从左到右递减排序，直到最右边的特征值也具有最小的特征值。

协方差矩阵

被称为随机变量 X 的方差的统计量定义如下：

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

协方差矩阵 C 是一个 n*n 的矩阵，其主对角线上的值是变量 X1、X2、... . . . ...Xn 的方差，C 中的其他值是每对变量 Xi 和 Xj 的协方差值。

变量 X and 和 Y 的协方差公式是方差的泛化，公式如下所示：

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

注意可以翻转乘法的乘法顺序（乘法是交换性的），因此协方差矩阵 C 是对称矩阵：

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

PCA 计算协方差矩阵 A 的特征向量和特征值。

使用数据集

除了数据清理之外，还需要执行其他几个步骤，例如选择训练数据与测试数据，以及决定在训练过程中使用"hold out"（留出法）还是交叉验证。

后续部分提供了更多详细信息。

训练数据与测试数据

完成本章前面介绍的任务（即数据清理和降维）后，可以将数据集分成两部分。第一部分是训练集，用于训练模型，第二部分是

测试集，用于"推断"（另一个表示预测的术语）。确保测试集符合一下规则：

- 该集足够大来产生具有统计意义的结果
- 它代表整个数据集
- 从不对测试数据进行训练
- 从不测试训练数据

什么是交叉验证？

交叉验证的目的是在非重叠测试集上测试模型，以下列方式执行：

•步骤 1） 将数据拆分为大小相等的 k 个子集

•步骤 2） 选择一个子集进行测试，另一个子集用于训练

•步骤 3） 对其他 k-1 子集重复步骤 2

此过程称为 k 折叠交叉验证，总体误差估计值是误差估计的平均值。标准评估方法涉及 k=10 的交叉验证。广泛的实验表明，10 个子 集是获得准确估计的最佳选择。事实上，您可以重复十倍交叉验证十次并计算结果的平均值，这有助于减少方差。

下一节将讨论正则化，如果您对 TF2 代码感兴趣，这是一个重要但可选的主题。如果你打算精通机器学习，您需要了解规范化。

什么是正则化？

正则化有助于解决过度拟合问题，即当模型在训练数据上表现良好，但在验证或测试数据上表现不佳时。

正则化通过向成本函数添加一个惩罚项来解决这个问题，从而通过这个惩罚项控制模型的复杂性。

正则化通常用于：

- 大量的变量
- 变量的低比率(# observations)/(# of variables)
- 很高的多重共线性

有两种主要类型的正则化：L1 正则化（与 MAE 或差异的绝对值相关）和 L2 正则化（与 MSE 或差异的平方相关）。通常，L2 的性能优于 L1，而且计算效率很高。

ML 和特征缩放

特征缩放标准化数据的特征范围。此步骤在数据预处理步骤期间执行，部分原因是梯度下降受益于特征缩放。

假设数据符合标准正态分布，标准化就是数据减去平均值，除以标准差，从而产生（0，1）正态分布。

数据规范化与标准化

数据规范化是一种线性缩放技术。假设数据集是{X1, X2, . . . , Xn} 以及以下变量：

Minx = 最小的 Xi

Maxx = 最大的 Xi

用下面的公式计算一组新的 Xi 值：

Xi =（Xi - Minx）/[Maxx -Minx]

现在，新的 Xi 的值将介于 0 和 1 之间。

偏差-方差权衡

机器学习中的偏差可能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错过特征和目标输出之间的相关关系（拟合不足）。因为存在"嘈杂"数据、不完整的特征集或存在偏差的训练样本，所以预测结果可能出现偏差。

由于偏差而导致的错误是模型的预期（或平均）预测与要预测的正确值之间的差值。多次重复构建模型的过程，每次收集新数据，并执行分析以生成新模型。生成的模型产生一个预测范围，因为基础数据集具有一定程度的随机性。偏差可以测量那些来自正确值的模型的预测范围。

机器学习的方差是每个样本值与全体样本值的平均数之差的平方值的平均数。高方差会/可能导致算法对训练数据中的随机噪声数据进行建模，而不是对预期输出进行建模（也称为过度拟合）。

向模型添加参数会增加其复杂性，增加方差，并减少偏差。处理偏差和方差是处理拟合不足和过度拟合。

方差导致的错误是给定数据点的模型预测的可变性。与以前一样，重复整个模型构建过程，方差是给定点的预测在模型的不同"实例"之间变化的程度。

测量模型的指标

最常用的指标之一是 R 方（R-squared），它测量数据与拟合回归线（回归系数）的接近度。R 方始终为介于 0 和 100% 之间的百分比。值 0% 表示模型未解释响应数据围绕其均值的变化。值 100% 表示模型解释了数据围绕其均值的所有变化。通常，较高的 R 方值表示更好的模型。

R 方的限制

尽管高的 R 方是首选值，但它们不一定始终是好值。同样，低的 R 平方值并不总是坏的。例如，用于预测人类行为的 R 平方值通常小于 50%。而且 R 平方无法确定系数估计值和预测值之间是否有偏差。此外，R 平方值并不能代表回归模型是否足够好。因此，一个优秀的模型可能具有较低的 R 方值，或者一个拟合不佳的模型具有高的 R 方值。与残差图、其他模型统计量和主题区域知识一起评估 R 方值。

混淆矩阵

在最简单的形式中，混淆矩阵（也称为错误矩阵）是一个两行和两列的列联表，其中包含 FP，FN，TP，TN 的。2x2 的混淆矩阵中的四个条目可以如下标记：

TP: True Positive

FP: False Positive

TN: True Negative

FN: False Negative

混淆矩阵的对角线上的值是正确的，而非对角的值表示预测不正确。通常，较低的 FP 值优于 FN 值。例如，FP 表示一个健康的人被错误地诊断为患病，而 FN 表示一个不健康的人被错误地诊断为健康。

正确率与精度与召回率

2x2 的混淆矩阵有四个项，它们分别代表正确和不正确的分类的各种组合。根据上一节中的定义，正确率、精度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(accuracy)可能是不可靠的指标，因为它在不平衡的数据集中会产生误导性的结果。当不同类的观测值数量显著不同时，它同时对 FP 和 FN 给予同等的重要性。例如，宣布癌症为良性比错误地告知患者他们患有癌症更糟糕。不幸的是，准确性不会区分这两种情况。

请记住，混淆矩阵可以是 nxn 矩阵，而不仅仅是 2x2 矩阵。例如，如果一个类有 5 个可能的值，则混淆矩阵为 5x5 矩阵，主对角线上的数字是"TP"结果。

ROC 曲线

ROC（接收器操作特性曲线 receiver operating characteristic）曲线是绘制 TPR 和 TPR 的曲线，即针对 FP 率（召回率）和 FP 率（误报率）。请注意，TNR（True negative）也称为特异性（specificity）。

以下链接包含使用 SKLearn 和 Iris 数据集的 Python 代码示例，以及用于绘制 ROC 的代码：

https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

以下链接包含用于绘制 ROC 的 Python 代码的示例：

https://stackoverflow.com/questions/25009284/how-to-plot-roc-curve-in-python

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，其中一些如下：

- Rss（回归平方和）
- Tss
- R-2
- F1 score
- p-value

RSS、TSS， TSS 和 R2 的定义如下所示，其中 y 是最佳拟合线上的点的 y 坐标，y_ （y 上面一个横线）是数据集中点的 y 值的平均值：

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$$

$$ESS = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2$$

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

R^2 = 1-RSS/TSS

什么是 F1-Score？

F1-score 是测试精度的度量，它被定义为精度和召回的调和平均数。下面是相关的公式，其中 p 是精度，r 是召回：

p =（TP）/（TP+FP）

r = （TP）/（TP+FN）

F1-score = 1/[（(1/r）+（1/p)）/2] = 2*[p*r]/[p+r]

F1score 的最好值为 0，最差值为 0。记住 F1score 通常用于非数值型分类问题，而 R2 通常用于回归任务（如线性回归）。

什么是 p 值？

如果 p 值足够小（<0.005），则表示有很高的意义，同时用于拒绝零假设。零假设表示因变量（如 y）和自变量（如 x）之间没有相关性。p 的阈值通常为 1% 或 5%。

没有用于直接计算 p 值的公式，p 值始终介于 0 和 1 之间。事实上，p 值是用于评估所谓的"零假设"的统计量，它们通过 p 值表的均值或通过电子表格/统计软件进行计算。

什么是线性回归？

线性回归的目标是找到"表示"数据集的最佳拟合线。记住两个要点。首先，最佳拟合线不一定通过数据集中的所有（甚至大部分）点。最佳拟合线的目的是最小化该线与数据集中点的垂直距离。其次，线性回归不是查找最适合的多项式：后者涉及查找通过数据集中多数点的较高次多项式。

此外，平面中的数据集可以包含位于同一垂直线上的两个或多个点，也就是说这些点具有相同的 x 值。但是，函数不能通过这样的一对点：如果两个点（x1，y1）和（x2，y2）具有相同的 x 值，则它们必须具有相同的 y 值（即 y1=y2）。另一方面，函数可以有两个或多个点位于同一水平线上。（**简单说就是函数不能 y 值一样。**）

现在考虑平面中具有许多点的散点图，这些点在拉长的云状形状中是"聚类"的：最佳拟合线可能只能经过有限的点数（事实上，最佳拟合线可能不会与所有点相交）。

需要记住的另一个例子：假设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} y 值在集合{2，4，6、...、20}.最佳拟合线的方程是 y=2*x+0。在这种情况下，所有的点是共线性的，也就是说，它们位于同一条线上。

线性回归与曲线拟合

假设数据集由 n 个数据点（x,y）组成，并且其中任何两个数据点的 x 值不同。然后根据数学中一个众所周知的结果，有一个次数小于或等于 n-1 的多项式通过这 n 个点（如果你对这个问题感兴趣，可以在网上找到这个数学证明）。例如，线是次数为 1 的多项式，它可以与平面中任何一对非垂直点相交。对于平面中的任何三角形的点（并非全部在同一条线上），有一个二次方程通过这些点。

此外，有时提供较低次数的多项式。例如，考虑 x 值等于 y 值的 个 100 点集：在这种情况下，函数 y=x（次数为一的多项式）通过所有的 100 个点。

但是，请记住，用曲线"表示"平面中的一组点的近似程度取决于这些点有多近似于这条线，由这些点的方差来衡量（方差是统计量）。这些点越有共线性，方差越小;相反，这些点越"分散"，方差越大。

解决方案何时是精确值？

尽管基于统计的解决方案为线性回归提供了闭式解（**也就是线性的，可以有表达式**）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涉及一系列近似序列，这些近似值将"收敛"到最佳值，这意味着机器学习算法生成精确值的估计值。例如，平面上一组点的最佳拟合线的斜率 m 和 y 轴上的截距 b 在统计中具有闭式解，但它们只能通过机器学习算法进行近似（会存在异常值，但它们很少出现）。

请记住，即使"传统"线性回归的闭式解为 m 和 b 提供了精确值，但有时只能使用精确值的近似值。例如，假设最佳拟合线的斜率 m 等于根号 3，b 是根号 2。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。在同一方案中，神经网络计算 m 和 b 的近似值，无论 m 和 b 的确切值是无理数、有理数还是整数。但是，机器学习算法更适合复杂、非线性、多维数据集，这超出了线性回归的范围。

作为一个简单的示例，假设线性回归问题的闭式解生成了整数或有理数的 m 和 b。具体地说，假设闭式解分别产生最佳拟合线的斜率和 y 上的截距值为 2.0 和 1.0。直线的方程如下所示：

```
y = 2.0 * x + 1.0
```

但是，训练神经网络得到的相应解可能会生成 2.0001 和 0.9997，作为最佳拟合线的 m 和 b 的值。始终牢记这一点，尤其是在训练神经网络时。

什么是多变量分析？

多变量分析将欧几里得平面中直线方程扩展到较高维度，称为超平面而不叫做直线。通用方程具有以下形式：

```
y = w1*x1 + w2*x2 + . . . + wn*xn + b
```

在二维的线性回归的情况下，您只需要查找斜率（m）和 y 上的截距（b）的值，而在多变量分析中，您需要找到 w1、w2、....，wn 的值。请注意，多变量分析是统计学中的一个术语，在机器学习中，它通常被称为"通用线性回归"。

请记住，本书中与线性回归相关的大多数代码示例都只涉及欧几里得平面中的二维的点。

其他类型的回归

线性回归找到"表示"数据集的最佳拟合线，但如果平面中的线不适合数据集，会发生什么？在使用数据集时会遇到相关的问题。

线性回归的一些替代方案包括二次方程、三次方方程或较高次数的多项式。但是，这些备选方案需要我们进行权衡，我们将稍后讨论。

另一种可能性是一种混合方法，它涉及分段线性函数，它包括一组线段。如果线段是连接在一起的，则它是一个分段线性连续函数;如果连续线段连接在一起，则它是一个线段。否则，它是一个分段线性不连续函数。

因此，给定平面中的一组点，回归涉及解决以下问题：

- 哪种类型的曲线最适合数据？我们怎么找到这条线？
- 另一种类型的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方法涉及可视化检查，但此方法对于高于两个维度的数据点不起作用。此外，这是一个主观决定，一些示例数据集将在本章的稍后部分显示。通过对数据集的可视化检

查，您可以确定二次或三次（甚至更高次数）多项式具有更适合这一数据集的潜力。但是，可视化检查可能仅限于二维或三维中的点。

让我们之后再讲非线性情况，让我们假设直线非常适合数据集。有一种著名的技术用于为此类数据集查找最佳拟合直线，该方法涉及最小化均方误差（MSE），本章稍后将对此进行讨论。

下一节提供了平面线性方程的快速回顾，以及一些说明线性方程示例的图像。

了解平面中的线（可选）

本节包含对欧几里德平面中的直线的简短回顾，因此如果您很了解这一部分则可以跳过此部分。经常被忽视的一个点是欧几里德平面中的线具有无限长度。如果选择一条线的两个不同的点，则这两个选定点之间的所有点组成线段。射线是一条"半无限"的线：当您选择一个点作为终点时，则直线另一侧的所有点都构成一条射线。

例如，平面中 y 坐标为 0 的点组成的线是 x 轴，而 x 轴上的（0，0）和（1，0）之间的点形成线段。此外，x 轴上（0，0）右侧的点形成射线，而（0，0）左侧 x 轴上的点也构成射线。

为了简单和方便起见，在本书中，我们将交替使用术语"线"和"线段"，现在让我们来深入了解欧几里德平面中的线的详细信息。以防你在细节上有点模糊，下面是欧几里德平面中（非垂直）线的方程： `y = m * x + b`

m 的值是线的斜率，b 的值是 y 截距（即线与 y 轴相交的位置）。

如果需要，您可以使用一个更通用的方程，该方程也可以表示垂直线，如下所示：

a*x + b*y + c = 0

但是，我们不会使用垂直线，因此我们将坚持使用第一个公式。

图 2.1 显示了三条水平线，其方程（从上到下）分别是 y = 3、y = 0 和 y = -3。

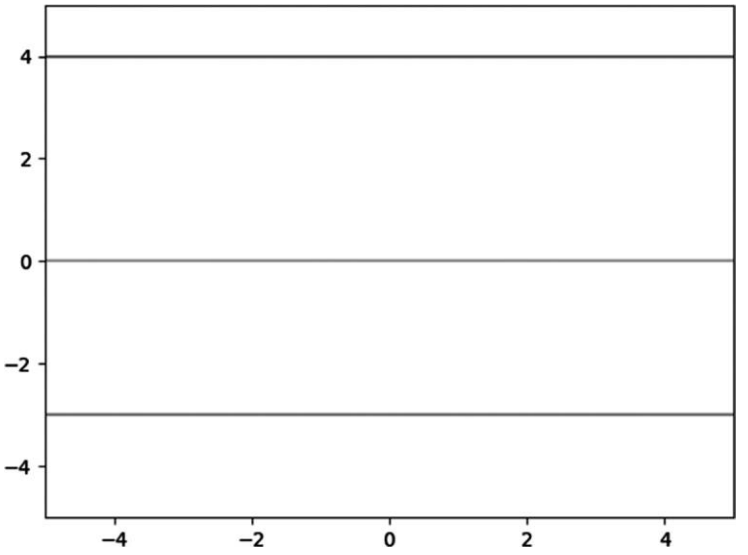

图 2.1 三条水平线段的图形。

图 2.2 显示两个倾斜线，其方程分别是 y = x 和 y = -x。

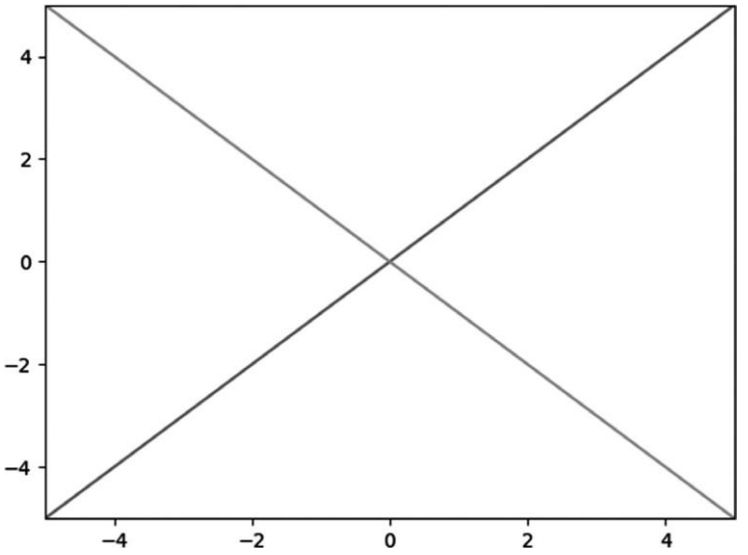

图 2.2 两个对角线段的图形。

图 2.3 显示了两条倾斜的平行线，其方程分别是 y = 2*x 和 y = 2*x+3。

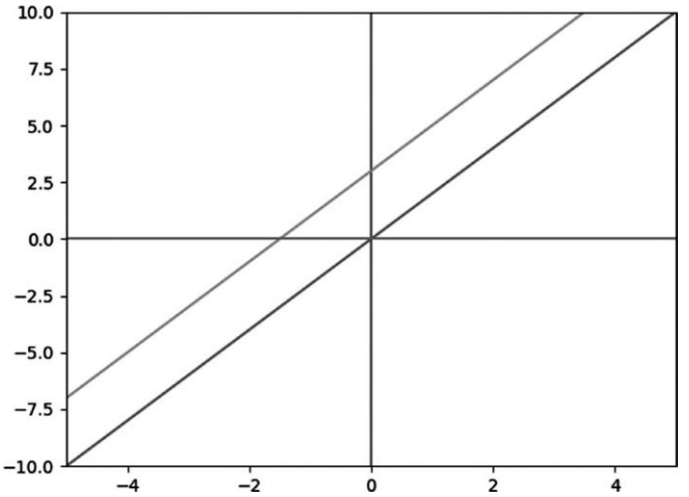

图 2.3 两个倾斜平行线段的图形。

图 2.4 显示了由连续的线段组成的分段线性图。

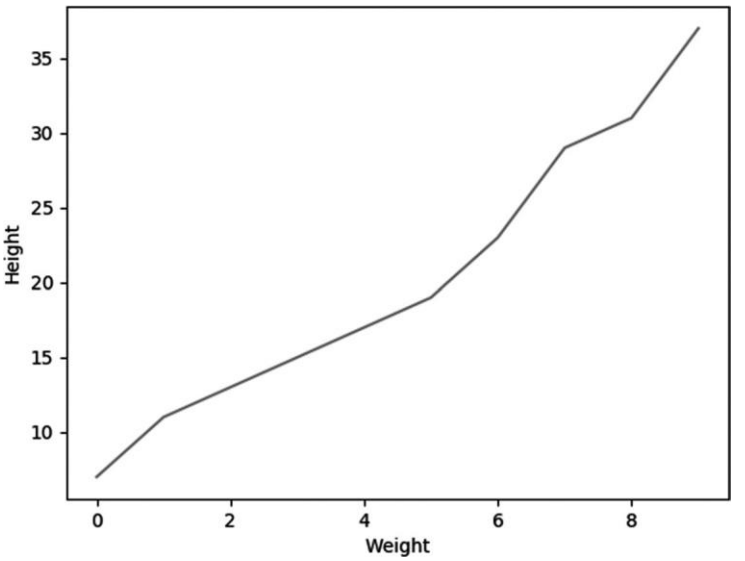

图 2.4 线段组成的分段线性图。

现在，让我们将注意力转向使用 NumPy API 生成准随机数据，然后使用 Matplotlib 绘制数据。

用 Numpy 和 Matplotlib 散点图 （1）

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

需要注意的一个细节是，所有相邻的水平值的等间距的，而垂直值基于线性方程加上"扰动"值。这种"扰动技术"（这不是标准术语）用于本章中的其他代码示例，以便在绘制点时添加一点随机的效果。这种技术的优点是 m 和 b 的最佳拟合值是预先知道的，因此我们不需要猜测它们的值。

列表 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()
```

列表 2.1 包含两个 import 语句，然后初始化数组变量 x，x 包括 15 个介于 0 和 1 之间的随机数。

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 模拟一组近似线段的值。

此技术用于模拟线段的代码样例，然后训练部分用于近似最佳拟合线的 m 和 b 的值。显然，我们已经知道最佳拟合线的方程：此技术的目的是将斜率 m 和截距 b 的训练值与已知值（本例中为 2.5 和 5）进行比较。列表 2.1 的部分输出如下：

```
      x：  [-1.42736308]
[  0.09482338]

       [-0.45071331]

       [ 0.19536304]

       [-0.22295205]


      y: [1.12530514]

      [5.05168677]

       [3.93320782]

       [5.49760999]

       [4.46994978]
```

图 2.5 显示基于 x 和 y 的值的散点图。

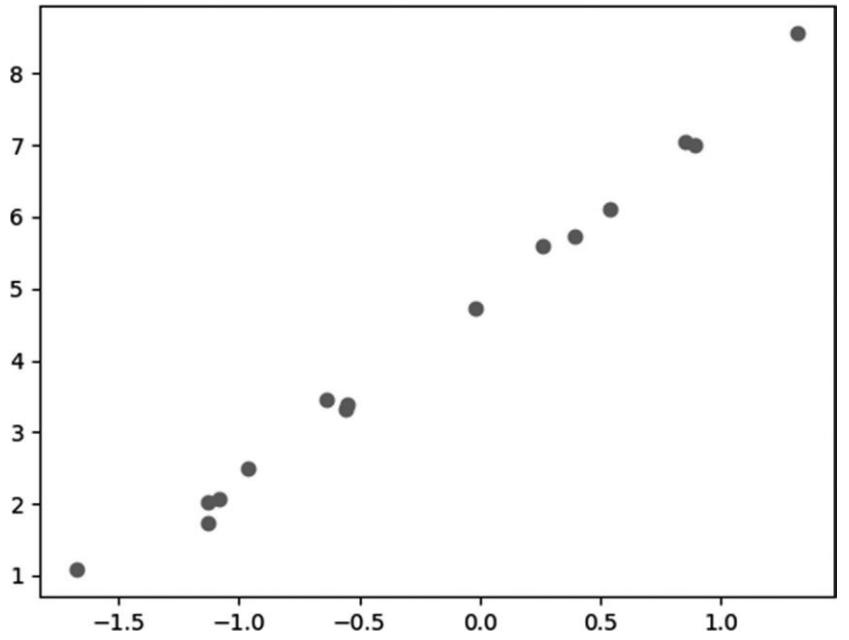

图 2.5 线段的散点图。

为什么"扰动技术"有用

您已经了解如何使用"扰动技术"，通过比较，考虑一个数据集，该数据集具有 Python 数组变量 X 和 Y：

```
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]
```

如果需要为上面的数据集找到最佳拟合线，如何猜测斜率 m 和截距 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"晃动"一条线上的点，其斜率 m 的值（以及可选的 b 值）是预先指定的。

请记住，"扰动技术"仅在引入小随机值时才有效，这些值不会导致 m 和 b 产生不同的值。

使用 NumPy 和 Matplotlib 的散点图 （2）

列表 2.1 中的代码将随机值分配给变量 x，而硬编码值（4）分配给斜率 m。 y 值是 x 值乘以硬编码（4），再加上通过"扰动技术"计算的随机值。但是我们不知道 b 的值。

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

本示例中的代码仅打印 trainX 和 trainY 的值，这些值对应于欧几里德平面中的数据点。列表 2.2 显示了 np_plot2.py 的内容，说明如何在 NumPy 中模拟线性数据集。

列表 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)
```

列表 2.6 通过 NumPy linspace（）API 初始化 NumPy 的数组变量 trainX，然后 trainY 的定义由两部分组成。 第一部分是线性变量 4*trainX，第二部分涉及随机生成的"扰动技术"。列表 2.6 的输出是：

```
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列表 2.2 的示例，它使用相同的"扰动技术"来生成一组近似二次方程（而不是线段）的点。

使用 NumPy 和 Matplotlib 的二次散点图

列表 2.3 显示了 np_plot_quadratic.py 的内容，该内容说明了如何在平面中绘制二次函数

列表 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()
```

列表 2.3 使用通过 np.linspace（） API 初始化数组变量 x，在这种情况下，生成的是一组 100 个在-5 和 5 之间等间距的浮点数。请注意 x 的初始化中的片段[:，None]，这将导致数组中每个元素都是一个由单个数字组成的数组。

数组变量 y 由两部分组成：第一部分是二次方程-0.5+2.2*x +0.3*x**2，第二部分是基于随机数的"扰动"值（类似于列表 2.1

中的代码）。因此，数组变量 y 模拟一组近似二次方程的值。；列表 2.3 的输出是：

```
x:
[[-5. ]
 [-4.8989899 ]
 [-4.7979798 ]
 [-4.6969697 ]
 [-4.5959596 ]
 [-4.49494949]
 // values omitted for brevity
 [ 4.8989899 ]
 [ 5. ]]
```

图2.6显示基于x 和 y的值的散点图，它们具有近似于二次方程的形状。

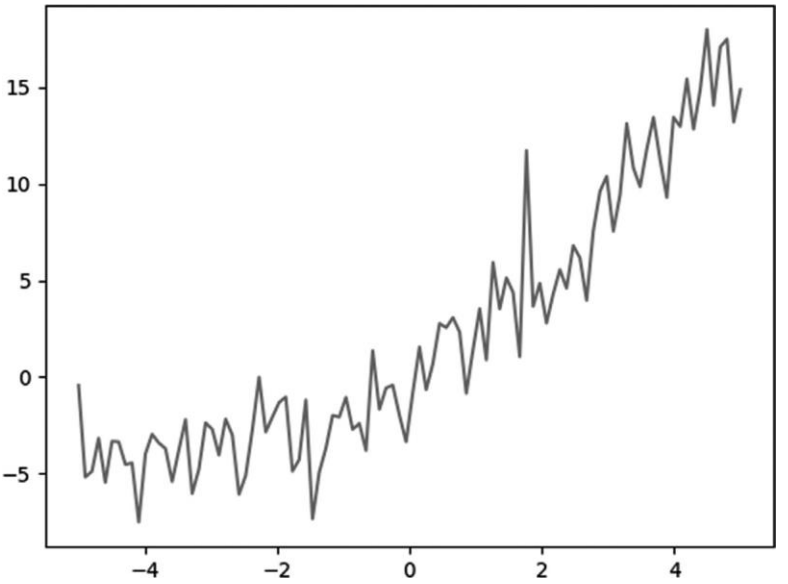

图 2.6 二次方程的散点图。

平均平方误差（MSE）公式

在英语中，MSE 是实际 y 值和预测 y 值之间差值的平方和，然后除以点数。请注意，预测 y 值是每个点在最佳拟合线上的 y 值。

尽管 MSE 在线性回归中很流行，但还有其他类型的误差可用，其中一些类型将在下一节中简要讨论。

错误类型列表

尽管本书中我们将只讨论线性回归的 MSE，但还有其他类型的公式可用于线性回归，其中一些公式如下：

- MSE

- RMSE
- RMSPROP
- MAE

MSE 是上述错误类型的基础。例如，RMSE 是"根均方误差"，它是 MSE 的平方根。

另一方面，MAE 是"平均绝对误差"，它是 y 的差值的绝对值的总和 （不是 y 的差的平方），然后除以点数。

RMSProp 优化器利用最近梯度的大小来规范化梯度。具体地说，RMSProp 在 RMS（根均值平方）梯度上保持平均移动，然后除以当前梯度。

尽管计算 MSE 的导数更容易，但 MSE 更容易受到异常值的影响，而 MAE 受到异常值的影响较小。原因很简单：平方项可以明显大于绝对值。例如，如果差值为 10，则 100 将被添加到 MSE，而仅向 MAE 添加 10。同样，如果差值为-20，则将平方项 400 添加到 MSE，而仅向 MAE 添加 20（即-20 的绝对值）。

非线性最小二乘法

在预测房价时（数据集包含广泛的值）时，线性回归或随机森林等技术可能导致模型过度拟合具有最高值的样本，以减少误差（如平均绝对误差）。

在这种情况下，您可能想要一个误差指标（如相对误差）来降低使用最大值拟合样本的重要性。此技术称为非线性最小二乘法，它可能使用基于日志的标签和预测值的变换。

下一节包含多个代码示例，其中第一个示例涉及手动计算 MSE，然后是使用 NumPy 公式来计算。最后，我们将查看用 TensorFlow 计算 MSE 的示例。

手动计算 MSE

本节包含两个图，每个图都包含一条线近似于散点图中的一组点。

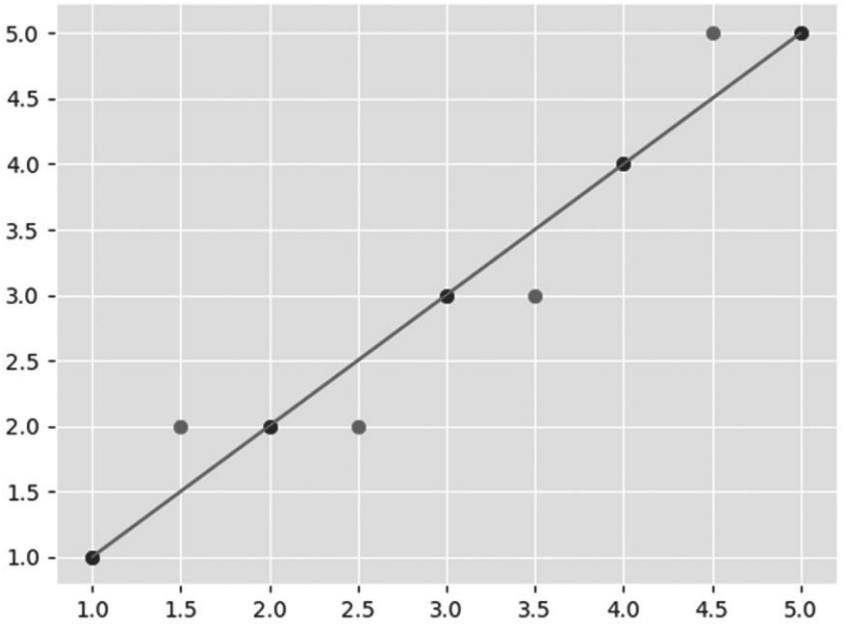

图 2.7 近似于散点图的点的线。

图 2.7 显示了一条近似线段（其中一些与线段相交）。图 2.7 中直线的 MSE 计算如下：

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

图 2.8 显示了一组点和一条，这条线是数据的最佳拟合线的潜在候选项。图 2.8 中直线的 MSE 计算如下：

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

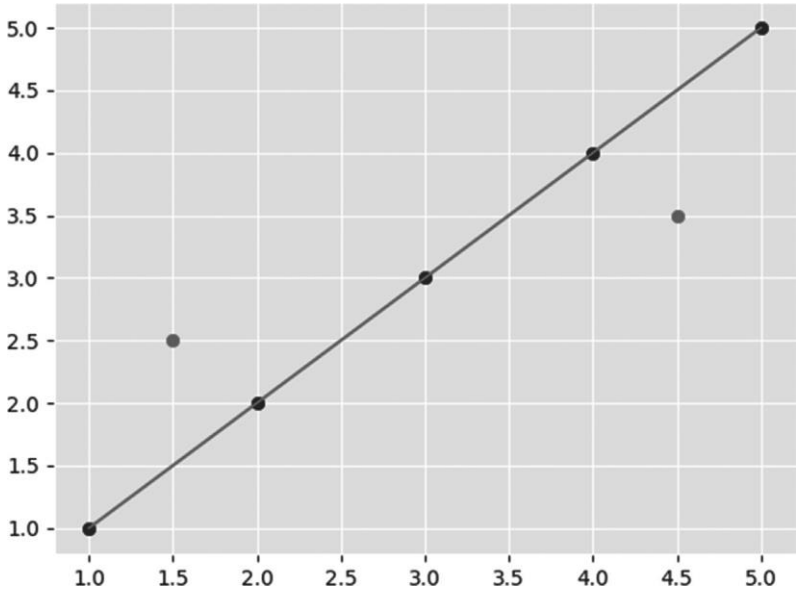

图 2.8 近似散点图中点的线。

因此，图 2.7 中的直线的 MSE 小于图 2.8 中的直线，这可能会令您感到惊讶（或者您猜对了？

在这两个数字中，我们轻松、快速地计算了 MSE，但总的来说，它更加困难。例如，如果我们在欧几里德平面中绘制 10 个远离直线的点，这些点的值都不是整数，则可能需要一个计算器。

更好的解决方案涉及 NumPy 函数，如 np.linspace（）API，下一节会讨论。

使用 np.linspace （）近似线性数据

列表 2.4 显示了 np_linspace1.py 的内容，该内容说明了如何使用 np.linspace（） API 与"扰动技术"一起生成一些数据。

列表 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)
```

此代码示例的目的只是生成和显示一组随机生成的数字。本章的稍后部分，我们将使用此代码作为实际线性回归任务的起点。

列表 2.4 从通过 np.linspace（） API 初始化数组变量 trainX 开始。接下来，数组变量 trainY 通过您在以前的代码示例中看到的"扰动技术"进行定义。列表 2.4 的输出是：

```
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]
```

现在，我们知道如何为线性方程生成（x，y），下面让我们学习如何计算 MSE，下一节将讨论这一点。

下一个示例使用 np.linspace（）和 np.random.randn（）生成一组数据值，以便在数据点中引入一些随机性。

使用 np.linspace()计算 MSE

本节中的代码示例与本章中的许多早期代码示例不同：它使用硬编码的数组 X 和 Y。因此，您将不知道斜率和截距的正确值（并且可能无法猜测它们的正确值）。列表 2.5 中 plain_linreg1.pyplain_ linreg1.py 演示如何使用模拟数据计算 MSE。

列表 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()
```

列表 2.5 使用硬编码值初始化数组变量 X 和 Y，然后初始化标量 W 和 b。列表 2.5 的下一部分包含一个 for 循环，该循环迭代 100 次。每次循环迭代后，Y_pred，Loss_error 和 cost 的值被算出来。接下来，根据数组 Y_pred Y_pred-Y 中的数据的总和以及 Y_pred-y 和 X 的内积计算 dw 和 db 的值。

请注意如何更新 W 和 b：它们的值分别由 0.01*dw 和 0.01*db 进行减少。此计算应该看起来有点熟悉：代码以编程方式计算 W 和 b 的梯度的近似值，这两个值都乘以学习速率（硬编码值 0.01），并且生成的变量从 W 和 b 的当前值中减去，以便生成 W 和 b 的新近似值。虽然这种技术非常简单，但它确实计算了 W 和 b 的合理值。

列表 2.5 中的最后代码块显示 W 和 b 的中间近似值，以及成本（垂直轴）与迭代次数（水平轴）的图。列表 2.5 的输出是：

10 次迭代的成本 = 0.04114630674619492

20 次迭代的成本 = 0.026706242729839392

30 次迭代的成本 = 0.024738889446900423

40 次迭代 的成本 = 0.023850565034634254

50 次迭代的成本 = 0.0231499048706651

60 次迭代的成本 = 0.02255361434242207

70 次迭代的成本 = 0.0220425055291673

80 次迭代的成本 = 0.021604128492245713

90 次迭代的成本 = 0.02122811750568435

W = 0.47256473531193927 & b = 0.1957826268862174

图 2.9 显示了列表 2.5 中代码生成的点的散点图。

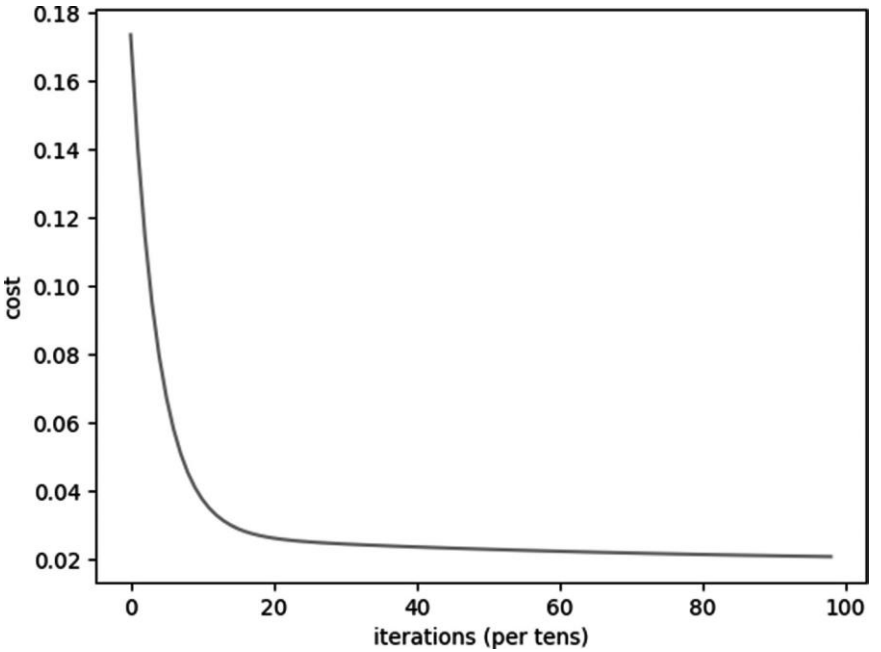

图 2.9带线性回归的 MSE 值。

代码示例 plain-linreg2.py 与列表 2.5 中的代码类似：区别在于，与具有 100 次迭代的单个循环不同，而是有一个执行 100 次的外部循环，在外部循环的每个迭代中，内部循环也执行 100 次。

Keras 上的线性回归

本节中的代码示例主要包含执行线性回归的 Keras 代码。如果您已经阅读了本章中的示例，则本节将更易于理解，因为线性回归的步骤是相同的。

列表 2.6 显示了 keras_linear_regression.py 的内容，说明如何在 Keras 中执行线性回归。

列表 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()
```

列表 2.6 从多个 import 语句开始，然后使用 CSV 文件 housing.csv 的内容初始化 dataframe df（其中一部分在列表 2.7 中）。请注意，训练集 X 使用数据集 housing.csv 的前 13 列的内容初始化，变量 y 包含数据集 housing.csv 的剩余部分。

列表 2.6 中的下一节使用 MinMaxScaler 类计算均值和标准差，然后调用 fit_ trans transform()方法来更新 X 值和 y 值，以便使它们具有平均值 0 和标准差 1。

接下来，python 中的 build_keras_mode（）方法创建有两个全连接层（dense layer）的基于 Keras 的模型。请注意，输入层的大小为 13，即 dataframe X 中的列数。下一个代码段使用 adam 优化器，MSE 损失函数来编译模型，并指定 MAE 和准确性（accuracy）作为指标。然后将编译后的模型返回给调用方。

列表 2.6 的下一部分将 batch_size 初始化为 32，将 epochs 初始化为 40，并用它们在创建模型的代码段中给相应的变量赋值，如下所示：

```
model =

tf.keras.wrappers.scikit_learn.

KerasRegressor(build_fn=build_keras_model,

batch_size=batch_size,epochs=epochs)
```

列表 2.6 中显示的简短注释块解释了前面代码段的用途，该代码段构建了我们的 Keras 模型。

列表 2.6 的下一部分调用 fit()方法来训练模型，然后在 X_test 数据上调用 predict()方法来计算一组预测值，并用这些预测值初始化变量 y_pred。

列表 2.6 的最后一部分显示了一个散点图，其中水平轴是 y_test 中的值（housing.csv 中的实际值），而垂直轴是一组预测值。

图 2.5 显示了测试值和这些测试值的预测值的散点图。

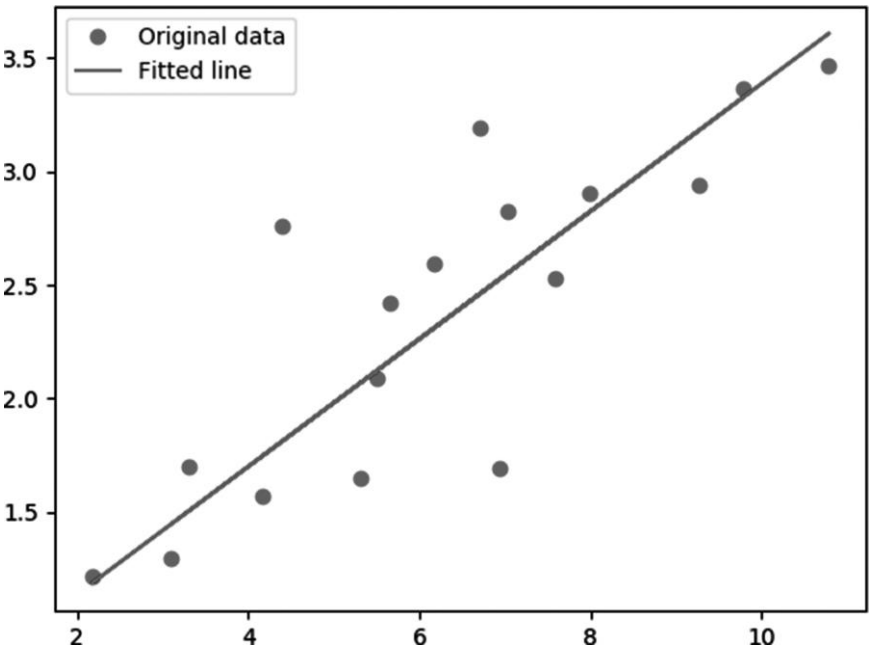

图 2.10 散点图和最佳拟合线。

列表 2.7 显示了列表 2.6 中 Python 代码使用的 CSV 文件 housing.csv 的前四行。

列表 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
   .7,394.63,2.94,33.4
```

总结

本章介绍了机器学习和特征选择、特征工程、数据清理、训练集和测试集等概念。接下来，您了解了监督学习、无监督和半监督学习。然后，您了解了回归任务、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，您还通过如何计算欧几里德平面中数据集的最佳拟合线的简要说明了解了线性回归任务。您看到了如何使用 NumPy 初始化数组，以及给 y 值引入一些随机性"扰动"技术。此技术很有用，因为您将知道最佳拟合线的斜率和截距的真实值，然后可以与训练值进行比较。

然后，您学习了如何在涉及 Keras 的代码样本中执行线性回归。此外，您还看到了如何使用 Matplotlib 来显示最佳拟合曲线，以及如何显示成本与训练相关的代码块的迭代次数的关系。