机器学习基础

本章我们简要介绍下机器学习（Machine Learning）的基本概念。主要介绍机器学习算法的应用，监督学习和无监督学习（supervised-unsupervised learning）的应用场景，训练和测试数据的用法，学习效果评估方式。最后，我们介绍scikit-learn及其安装方法。

自计算机问世以来，计算机可以学习和模仿人类智慧的观点，可谓“引无数英雄竞折腰”。像Arthur C. Clarke的HAL（Heuristically programmed ALgorithmic computer）(https://en.wikipedia.org/wiki/HAL_9000)和Isaac Asimov的Sonny (https://en.wikipedia.org/wiki/I,_Robot_(film))那样的人工智能已经成为共识，通过学习经验获得新知识和技能的软件程序也变得越来越多。我们用这机器学习程序发现我们会喜欢的新音乐，快速找出我们想网购的鞋子。机器学习程序让我们通过命令控制手机，让恒温器自动调节温度。比人类更准确的识别出潦草的手写邮箱地址，更安全的保护信用卡防止诈骗。从新药品调查到从网页寻找头条新闻，机器学习软件逐渐成为许多产业的核心工具。机器学习已经进入长期以来一直被认为只有人类才能胜任的领域，如杜克大学UNC(Duke)篮球队输给了北卡（UNC）的体育报道。

机器学习是设计和研究能够根据过去的经验来为未来做决策的软件，它是通过数据进行研究的程序。机器学习的基础是*归纳（generalize）*，就是从已知案例数据中找出未知的规律。机器学习的典型案例就是垃圾邮件过滤。通过对数千份已经打上是否为垃圾标签的邮件进行观察经验，对新邮件进行过滤。

人工智能研究领域的计算机科学家Arthur Samuel说，机器学习是“研究如何让计算机可以不需要明确的程序也能具有学习能力”。在20世纪五六十年代，Samuel开发了下象棋程序。程序的规则非常简单，要打败专业对手需要复杂的策略，但是通过几千局游戏的训练，程序学会了复杂的策略，可以打败很多人类棋手。

计算机科学家Tom Mitchell对机器学习的定义更正式，“一个程序在完成任务*T*后获得了经验*E*，其表现为效果*P*，如果它完成任务*T*的效果是*P* ，那么会获得经验*E*”。例如，假设你有一些图片，每个图片里是一条狗或一只猫。程序可以通过观察图片来学习，然后它可以通过计算图片正确分类比例来评估学习效果。

我们将使用Mitchell的定义来组织这一章的内容。首先，我们要介绍经验的类型，包括监督学习和无监督学习。然后，我们介绍机器学习系统可以处理的常见任务。最后，我们介绍机器学习系统效果评估方式。

从经验中学习

机器学习系统通常被看作是有无人类监督学习两种方式。监督学习问题是，从成对的已经标记好的输入和输出经验数据作为一个输入进行学习，用来预测输出结果，是从有正确答案的例子中学习。而无监督学习是程序不能从已经标记好的数据中学习。它需要在数据中发现一些规律。假如我们获取了人的身高和体重数据，非监督学习的例子就是把数据点分成组别。一种程序可能是把数据分成男人与女人，儿童与成人等不同组别。

再假设数据都标记了人的性别。那么，一种监督学习方式就是基于一个人的身高和体重数据来预测这个人是男是女。后面我们会介绍监督学习与非监督学习的算法和案例。

监督学习与非监督学习可以看作机器学习的两端。还有一些中间类型，称为半监督学习，既包含监督数据也有非监督数据，这类问题可以看作是介于监督学习与非监督学习之间的学习。半监督机器学习案例是一种增强学习(Reinforcement Learning)，问题可以通过决策来获得反馈，但是反馈与某一个决策可能没有直接关系。例如，一个增强学习程序学习玩超级玛丽游戏，让它完成一级或超过一定分数会获得奖励，如果失败一次会受到惩罚。但是，监督反馈与具体要执行的决策无关，避开板栗仔（Goombas）或者跳过火轮圈。本书讨论的半监督学习将集中于监督与非监督学习，因为这些类型包括机器学习的绝大多数问题。下一章，我们会介绍监督学习与非监督学习的更多细节。

监督学习是通过一个输入产生一个带标签的输出的经验数据对中进行学习。机器学习程序中输出结果有很多名称，一些属于机器学习领域，另外一些是专用术语。本书中，我们把输出结果称为响应值（response variable），不过输出结果还有其他名称，如因变量（dependent variables），回归值（regressands），标准变量（criterion variables），测得变量（measured variables），解释变量（explained variables），结果变量（outcome variables），实验变量（experimental variables），标签（labels），和输出变量（output variables）。同理，输入变量也有很多名称。本书把输入变量作为特征（features），它们测量的现象作为解释变量（explanatory variables）。输入变量的其他名称有，预测值（predictors），解释变量（regressors），控制变量（controlled variables），操作便利（manipulated variables）和显现变量（exposure variables）。响应变量和解释变量可能需要真实的或不相关的数值。

构成监督学习经验的案例集合称为训练集（training set）。评估程序效果的案例集合称为测试集（test set）。响应变量可以看成是解释变量构成问题的答案。监督学习问题从不同问题结合中学习，就是说，监督学习程序输入是正确的答案，需要对类似的问题作出正确的反馈。

机器学习任务

常见的监督式机器学习任务就是分类（classification）和回归（regression）。分类认为需要学会从若干变量约束条件中预测出目标变量的值，就是必须预测出新观测值的类型，种类或标签。分类的应用包括预测股票的涨跌，新闻头条是政治新闻还是娱乐新闻。回归问题需要预测连续变量的数值，比如预测新产品的销量，或者依据工作的描述预算工资水平等。与分类方式类似，回归问题需要监督学习。

常见的无监督式机器学习任务是通过训练数据发现相关观测值的组别，称为类（clusters）。对应的任务称为聚类（clustering），通过一些相似性度量方法把一些观测值分成同一类。聚类常用来分析数据集。比如有一些影评数据，聚类算法可以分辨积极的和消极的影评。系统是不能给类加上“积极”或“消极”的标签的；没有监督，系统只能通过相似性度量方法把观测值分成两类。聚类分析的应用场景是用市场产品销售数据为客户分级。通过挖掘一组用户的共同属性，销售人员可以为这类客户提供定制服务。聚类还被用于互联网广播服务，比如有一些歌曲，聚类算法能够按风格流派把歌曲分组。通过不同的相似性度量方法，同样的聚类算法可能通过关键词来分组，也可能通过使用的乐器来分组。

降维（Dimensionality reduction）是另一个常见的无监督学习任务。有些问题可能包含成千上万个解释变量，处理起来非常麻烦。另外，有些解释变量属于噪音，也有些完全是无边的变量，这些影响都会降低程序的归纳能力。降维是发现对响应变量影响最大的解释变量的过程。降维可以更容易的实现

数据可视化。如不同面积房子的价格数据可视化，房子的面积可以画在x轴，其价格可以画在y轴，很容易实现可视化。再加一个解释变量，也很容易可视化房屋价格的回归问题，比如房间里卫生间的数量可以画在z轴。但是，几千个解释变量的问题是不可能可视化的。

训练数据和测试数据

训练集里面的观测值构成了算法用来学习的经验数据。在监督学习问题中，每个观测值都由一个响应变量和若干个解释变量组成。

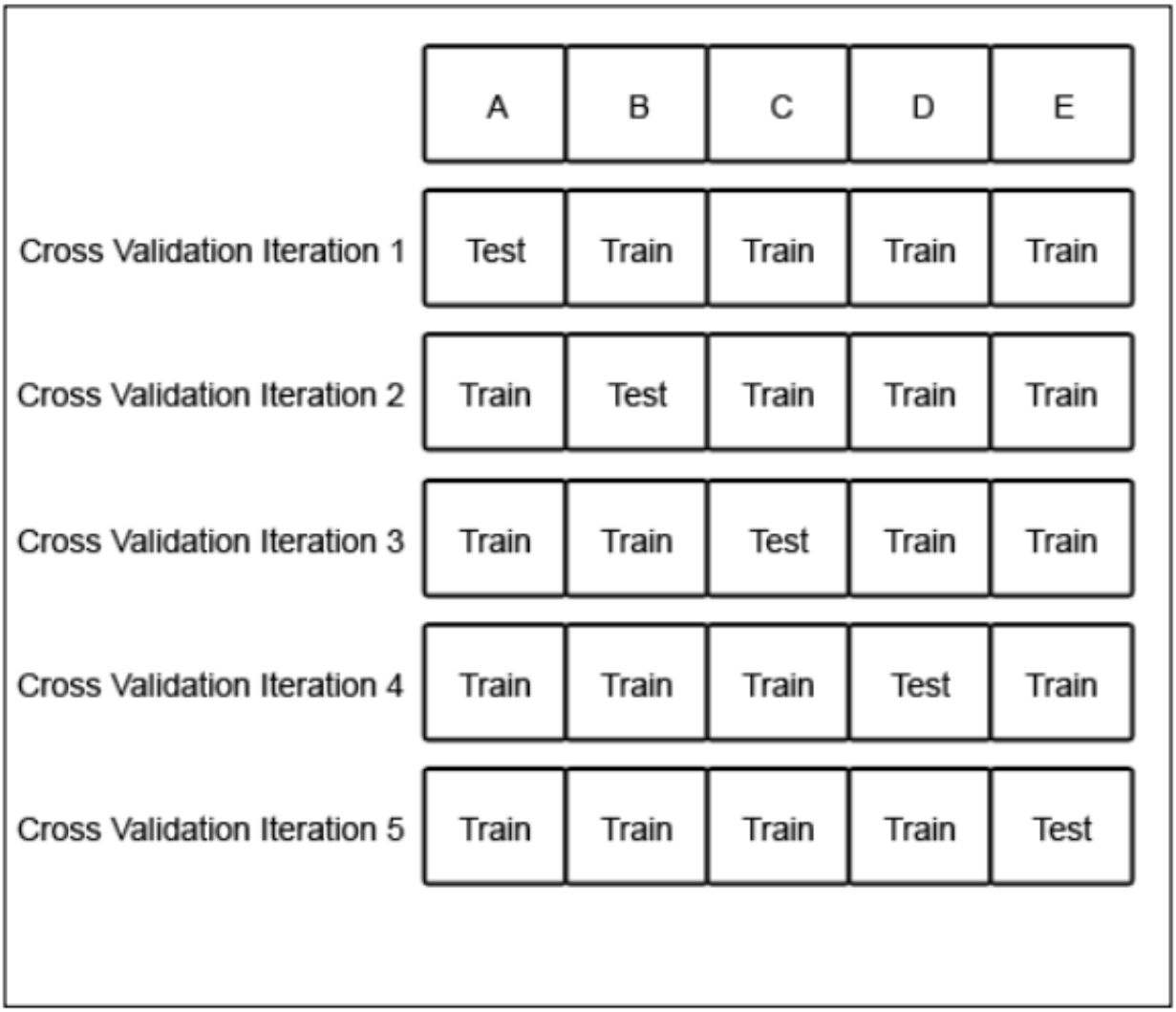
测试集是一个类似的观测值集合，用一些度量标准来评估模型的运行效果。需要注意的是，测试集的数据不能出现在训练集中。否则，很难评价算法是否从训练集中学到了归纳能力，或者仅仅只是简单的记录了结果。归纳很好的程序能够用新数据有效地完成任务。相反，一个通过记忆训练数据来学习复杂模型的程序，可能通过训练集准确预测响应变量的值，但是在处理新问题的时候由于没有归纳能力会预测失败。

训练集的记忆称为过度拟合（over-fitting）。一个记住了观测值的程序不一定能够很好的完成工作，因为它在记忆关系和结果的时候，把噪声也同时记住了。平衡记忆能力与归纳能力，或者说是过度拟合与拟合不够，是许多机器学习算法面对的共同问题。后面的章节，我们会介绍正则化（regularization），可以用来减轻许多模型的过度拟合程度。

除了训练集和测试集，还有一个观测值集合称为验证集（validation set或 hold-out set），有时候需要用到。验证集用来调整超参数（hyperparameters）变量，这类变量控制模型是如何学习的。这个程序也通过测试集来评估其真实的效果，验证集的效果不能用于评估其真实的效果，由于程序参数已经用验证数据调整过了。通常会把监督学习的观测值分成训练、验证和测试集三部分。各部分的大小没有要求，按实际观测值的规模来定。一般把50%以上的数据作为训练集，25%的数据做测试集，剩下的作为验证集。

有的训练集只包含几百个观测值，有的可能有几百万。随着存储成本越来越便宜，网络连接范围不断扩大，内置传感器的智能手机的普及，以及对隐私数据态度的转变都在为大数据新动力，千万甚至上亿级别的训练集成为可能。本书不会涉及这类需要上百个机器并行计算才能完成的任务，许多机器学习算法的能力会随着训练集的丰富变得更强大。但是，机器学习算法也有句老话“放入的是垃圾，出来的也是垃圾”。一个学习了一大堆错误百出的教材的学生不会比只读一点好书的学生考得好。同理，对一堆充满噪声、没有关联、或标签错误的数据进行学习的算法，也不会比只学习一小部分更有代表性的训练集的算法效果更好。

许多监督学习的训练集都是手工准备的，或者半自动处理。建一个海量监督数据集需要耗费许多资源。好在scikit-learn有些数据集，可以让开发者直接验证自己的模型。在开发阶段，尤其是训练集不够的时候，交叉验证（cross-validation ）的方法可以用相同的数据对算法进行多次训练和检验。在交叉验证中，训练数据是分成N块的。算法用N-1块进行训练，再用最后一块进行测试。每块都被算法轮流处理若干次，保证算法可以在训练和评估所有数据。下图就是5块数据的交叉验证方法：

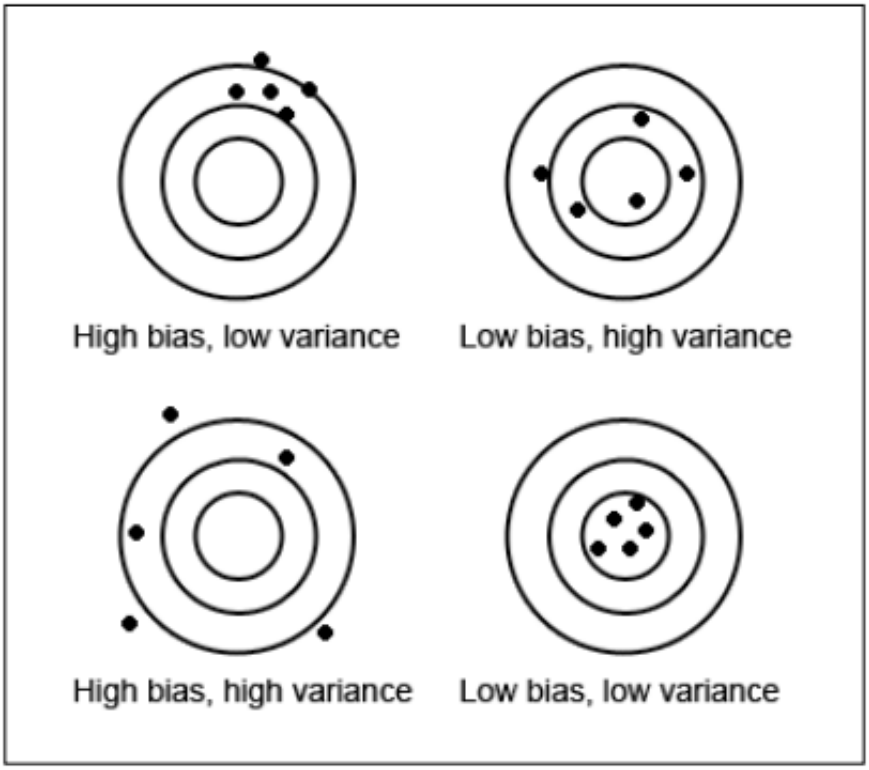


数据集被等分成5块，从A标到E。开始的时候，模型用B到E进行训练，在A上测试。下一轮，在A，C，D和E上训练，用B进行测试。依次循环，直到每一块都测试过。交叉验证为模型的效果评估提供了比只有一个数据集更准确的方法。

效果评估，偏差，方差

许多度量方法可以用于评估一个程序是否学会了有效处理任务。在监督学习问题中，很多效果度量标准用来评估预测误差。有两种基本的预测误差：模型的偏差（bias）和方差（variance）。假设你有很多训练集都是不一样的，但是都具有代表性。一个高偏差的模型会产生类似的误差，无论它使用哪个训练集。模型偏离自己对真实关系假设的误差超过了模型在训练集训练的结果。模型有高偏差是固定不变的，但是模型有高方差可能是灵活的，因为模型发觉了训练集里面的噪音部分。就是说，高方差的模型是过度拟合了训练集数据，而一个模型有高偏差的时候，其实是拟合不够的表现。

偏差和方差就像飞镖射到靶子上。每个飞镖就是从不同数据集得出的预测结果。高偏差、低误差的模型就是把飞镖扔到了离靶心很远的地方，但是都集中在一个位置。而高偏差、高误差的模型就是把飞镖扔到了靶子上，但是飞镖离靶心也很远，而且彼此间很分散。低偏差、高误差的模型就是把飞镖扔到了离靶心很近的地方，但是聚类效果不好。最后就是低偏差、低误差的模型，把飞镖扔到了离靶心很近的地方，聚类效果也很好。如下图所示：



在理想情况下，模型具有低偏差和低误差，但是二者具有背反特征，即要降低一个指标的时候，另一个指标就会增加。这就是著名的偏差-方差均衡(Bias-Variance Trade-off)。后面我们会介绍很多模型的偏差-方差均衡特点。

无监督学习问题没有误差项要评估，其效果的是评估数据结构的一些属性。

大多数效果评估方法只能用于具体的任务。机器学习系统应该可以这样评估：用系统在真实世界中发生错误的代价来表示效果评估方法。这看起来很明显，下面例子描述的是适用于一般任务而不只是具体任务的效果评估方法。

有一个对肿瘤数据进行观察的机器学习系统分类任务，需要预测肿瘤是恶性的（malignant）还是良性的（benign）。准确度，或者是正确分类的比例，就是对程序效果评价的直观度量方法。准确度能够评价程序效果，不过它不能区分出，误把良性肿瘤分为恶性肿瘤，和误把恶性肿瘤分为良性肿瘤的效果差异。在一些应用里，发生不同类型错误的代价是相同的。但是，在这个问题里面，没有识别出恶性肿瘤的代价要比误把良性肿瘤分为恶性肿瘤的代价要大的多。

我们可以通过对每一种可能的预测结果进行评估来建立分类系统效果的不同评价方法。当系统正确地识别出一个恶性肿瘤，这个预测叫真阳性(True positive)；如果把一个良性肿瘤分成了一个恶性肿瘤，叫假阳性(False positive）；正确地识别出一个良性肿瘤叫真阴性(True negative)；把一个恶性肿瘤分成了一个良性肿瘤，叫假阴性(False negative）。这四个结果可以用来计算分类系统效果的评价体系，包括准确率（accuracy），精确率（precision）和召回率（recall）三项指标。

准度率计算公式如下，*TP*是真阳性统计结果，*TN*是真阴性统计结果，*FP*是假阳性统计结果，*FN*是假阴性统计结果：

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}$$

精确率是被判断为恶性肿瘤中，真正为恶性肿瘤统计结果所占比例：

$$P = \frac{TP}{TP + FP}$$

召回率是指真正为恶性肿瘤被分类系统判断出来的比例：

$$R = \frac{TP}{TP + FN}$$

在这个例子中，精确率是评估被系统判断为恶性肿瘤中的肿瘤里面，确实为恶性肿瘤的比例。而召回率是评估真实的恶性肿瘤被系统正确判断出来的比例。

从精确率和召回率评估指标可以看出，高准确率的分类系统实际没有发现出所有的恶性肿瘤。如果绝大多数肿瘤都是良性的，那么分类器没有预测出恶性肿瘤也可以获得极高的准确率。而一个具有低准确率和高召回率的分类系统反而更好，因为它能够识别出更多恶性肿瘤。

许多其他效果评估方法都可以用于分类方法中，后面我们会介绍一些，包括多标签分类问题的评价标准。下一章，我们会介绍一些回归问题的常用评价标准。

scikit-learn简介

自2007年发布以来，scikit-learn已经成为最给力的Python机器学习库（library）了。scikit-learn支持的机器学习算法包括分类，回归，降维和聚类。还有一些特征提取（extracting features）、数据处理（processing data）和模型评估（evaluating models）的模块。

作为Scipy库的扩展，scikit-learn也是建立在Python的NumPy和matplotlib库基础之上。NumPy可以让Python支持大量多维矩阵数据的高效操作，matplotlib提供了可视化工具，SciPy带有许多科学计算的模型。

scikit-learn文档完善，容易上手，丰富的API，使其在学术界颇受欢迎。开发者用scikit-learn实验不同的算法，只要几行代码就可以搞定。scikit-learn包括许多知名的机器学习算法的实现，包括LIBSVM和LIBLINEAR。还封装了其他的Python库，如自然语言处理的NLTK库。另外，scikit-learn内置了大量数据集，允许开发者集中于算法设计，节省获取和整理数据集的时间。

scikit-learn可以不受任何限制，遵从自由的BSD授权。许多scikit-learn的算法都可以快速执行而且可扩展，除了海量数据集以外。最后，scikit-learn稳定性很好，大部分代码都可以通过Python的自动化测试（mock，nose等）。

安装scikit-learn

目前scikit-learn的稳定版本是0.16.1。用这个版本可以保证本书的代码可以正常运行，如果你之前装过，可以检查一下版本：

In [3]:

```
import sklearn
sklearn.__version__
```

Out[3]:

```
'0.16.1'
```

如果你还没安装，你可以通过pip安装，也可以从源代码安装。下面我们简单介绍一下Windows，Linux和Mac OS系统分别安装的过程，具体内容可以从官方网站 (http://scikit-learn.org/dev/install.html)找到。Python版本需要是Python (>= 2.6 or >= 3.3)，NumPy (>= 1.6.1)，SciPy (>= 0.9)。

Windows系统安装

强烈建议使用miniconda (http://conda.pydata.org/miniconda.html)安装，可以根据Windows系统版本和想要的Python版本选择下载安装。

miniconda会自动安装scikit-learn的依赖包，如NumPy和SciPy等。双击安装文件安装miniconda，之后在命令行工具输入下列指令：

```
conda install scikit-learn
```

确实之后一会儿就安装完了，如果要更新版本，直接输入：

```
conda update scikit-learn
```

如果已经装好了Python，Numpy和Scipy，可以用pip安装（Python 2.7.9和Python3.4自带）：

```
pip install -U scikit-learn
```

如果想从源代码编译scikit-learn，请参阅官方最新文档，可以从GitHub下载最新代码 (https://github.com/scikit-learn/ scikit-learn)。

Linux系统安装

Linux系统安装由你的发行版决定。最好的方法是用pip安装，也可以通过源代码编译安装。下面以Ubuntu 14.10为例：

pip安装很简单：

```
sudo pip install -U scikit-learn
```

源代码安装需要一堆依赖：

```
sudo apt-get install python-dev python-numpy python-numpy-dev pyth
onsetuptools python-numpy-dev python-scipy libatlas-dev g++
```

然后执行命令：

```
python setup.py install
```

Mac OS系统安装

直接用pip很简单：

```
pip install -U numpy scipy scikit-learn
```

安装版本检验

装完之后，就可以像前面那样检验一下版本：

In [3]:

```
import sklearn
sklearn.__version__
```

Out[3]:

```
'0.16.1'
```

要执行scikit-learn的单元测试，首先要安装nose库，然后运行：

```
nosetests -v sklearn
```

这样scikit-learn就安装好了。

安装pandas和matplotlib库

本书还会用到pandas (http://pandas.pydata.org/pandas-docs/stable/index.html)库，是一个开源Python数据分析工具。Windows，Linux和Mac OS系统都可以用pip安装：

```
pip install pandas
```

如果安装了miniconda，也可以用conda命令安装。

在Debian-和Ubuntu-系统上也可以通过apt-get命令安装：

```
apt-get install python-pandas
```

matplotlib (http://matplotlib.org/)是一个图形库，可以很容易的创建点图，直方图，曲线图等不同样式的图形。matplotlib的依赖关系和pandas一样，都需要Numpy。前面已经装过了，一样可以通过pip或conda安装，在Debian-和Ubuntu-系统上也可以通过apt-get命令安装：

```
apt-get install python-matplotlib
```

Windows和Mac OS系统系统上也有可执行文件可以安装。

总结

本章，我们把机器学习定义成一种程序的设计和研究过程，其可以建立一种从一件任务的过往经验中学习并改善处理能力的程序。我们讨论了经验监督的范围。一端是监督学习，程序从打上标签的输入和输出数据中学习。另外一种是无监督学习，程序需要发现没有标签数据的内置结构。半监督学习同时使用有标签和无标签的训练数据。

我们通过案例介绍了机器学习的常见问题。在分类任务中，程序需要从解释变量预测出响应变量的离散数值。在回归任务中，程序从解释变量预测出响应变量的连续数值。无监督学习任务包括聚类和降维，聚类是将观测值通过相似度评价方法分成不同的类，降维是将解释变量集合缩减为一个合成特性集合，同时尽可能的保留数据的信息。我们还介绍了偏差-方差均衡和不同机器学习任务的效果评价方法。

最后，我们介绍了scikit-learn的历史，目标和优点，以及scikit-learn和相关开发工具的安装过程。下一章，我们就详细的介绍回归问题，用scikit-learn建立本书的第一个模型。