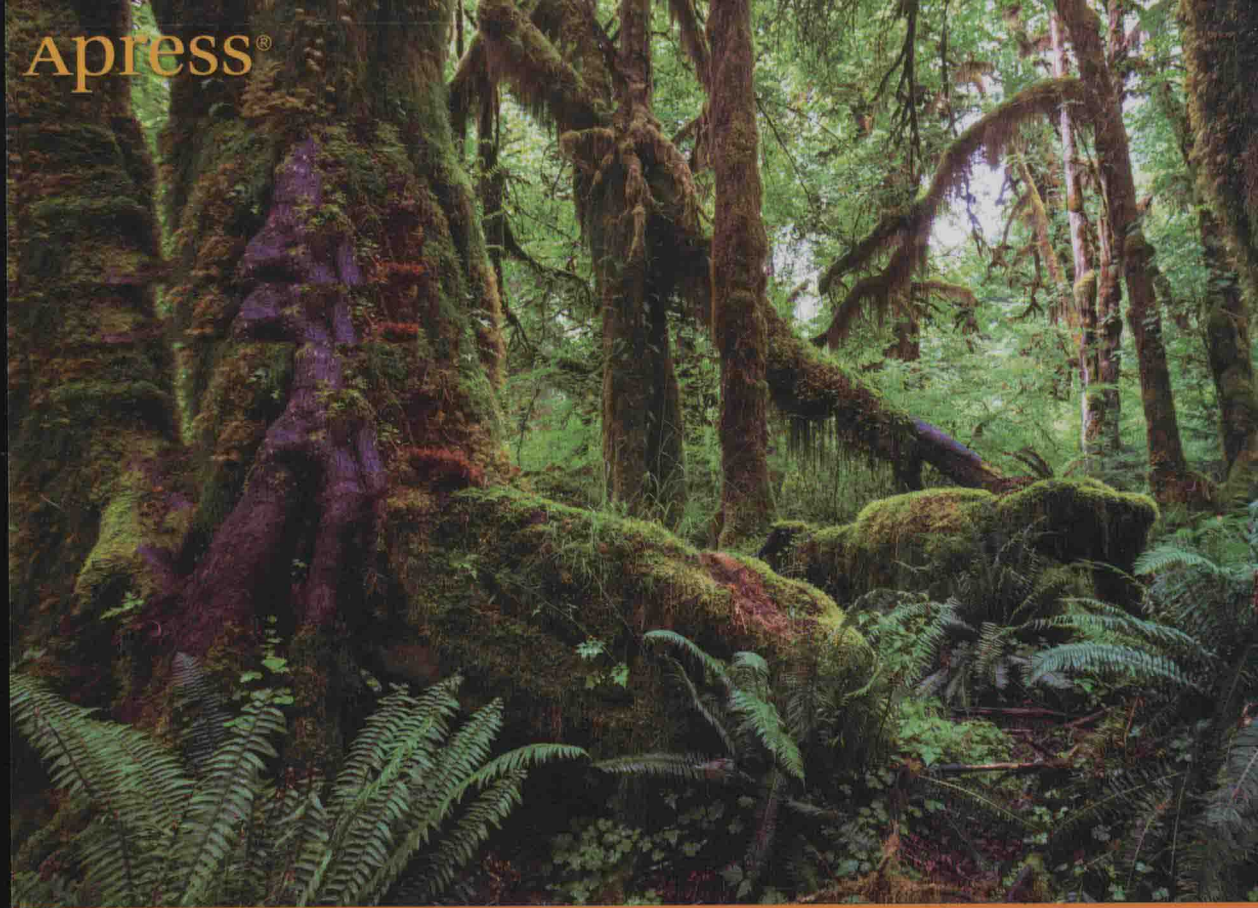

Apress®

[印] 尼基尔·盖德卡尔 著　杜长营 苏 辉 译

Python深度学习

Deep Learning with Python

清华大学出版社

Python 深度学习

[印] 尼基尔·盖德卡尔 著
杜长营 苏 辉 译

清华大学出版社
北 京

内 容 简 介

本书详细阐述了与 Python 深度学习相关的基本解决方案，主要包括深度学习介绍、机器学习基础、前馈神经网络、Theano 介绍、卷积神经网络、递归神经网络、Keras 介绍、随机梯度下降、自动求导、GPU 介绍等内容。此外，本书还提供了丰富的示例及代码，帮助读者进一步理解相关方案的实现过程。

本书既适合作为高等院校计算机及相关专业的教材和教学参考书，也可作为相关开发人员的自学教材和参考手册。

北京市版权局著作权合同登记号 图字：01-2017-7948

图书在版编目（CIP）数据

Python 深度学习/（印）尼基尔·盖德卡尔著；杜长营，苏辉译. —北京：清华大学出版社，2018
书名原文：Deep Learning with Python
ISBN 978-7-302-51287-5

Ⅰ. ①P…　Ⅱ. ①尼…　②杜…　③苏…　Ⅲ. ①软件工具-程序设计　Ⅳ. ①TP311.561

中国版本图书馆 CIP 数据核字（2018）第 220195 号

责任编辑： 贾小红
封面设计： 刘　超
版式设计： 文森时代
责任校对： 马军令
责任印制： 李红英

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 厂： 北京密云胶印厂
经　　销： 全国新华书店
开　　本： 185mm×230mm　　**印　　张：** 8.75　　**字　　数：** 183 千字
版　　次： 2018 年 11 月第 1 版　　**印　　次：** 2018 年 11 月第 1 次印刷
定　　价： 59.00 元

产品编号：075784-01

目　录

第 1 章　深度学习介绍

本章将围绕深度学习这个主题进行一个广泛的概述和对其历史背景的介绍。同时还为读者提供了本书的学习路线图、先决条件和进一步的阅读材料，以深入探讨这一主题。

1.1　历 史 背 景

人工智能领域（AI）可以被认为是深度学习的母领域，其历史可以追溯到 1950 年。虽然我们不会详细地介绍这个历史，但是我们将继续讨论一些在这个领域的关键转折点，以便引导我们能深入地学习。

AI 早期专注的是那些可以被正式描述的任务，比如棋盘游戏或国际象棋。判断是不是能够被正式描述的任务的标准是计算机程序可不可以简单地表示这个任务。例如国际象棋游戏。国际象棋游戏的形式化描述是如何表示棋盘；如何描述每个棋子的移动；如何描述起始配置以及游戏终止的配置。

随着这些概念的形式化，对一个象棋游戏的 AI 程序建模变得相对容易，并且如果有足够计算资源的情况下，可以产生一个相对较好的象棋游戏 AI。

人工智能的第一个时代把已经获得很大成功的任务作为重点。其核心是对领域的符号表示和基于给定规则进行符号操作（越来越复杂的算法用于搜索解空间以获得解决方案）。

必须指出的是，这些规则的正式定义是手动指定的。然而，早期的人工智能系统是通用的任务/问题解决者，因为任何可以正式描述的问题都可以用通用的方法来解决。

这种系统的局限在于，国际象棋是一个相对容易的 AI 问题，因为问题的设置相对简单，而且可以很容易地形式化。人类在日常生活中遇到的许多问题并非如此。例如，考虑诊断一种疾病（如医生所做的诊断）或将人类的语音转为文本。这些任务和大多数人类能够容易掌握的任务一样，很难正式描述，并在人工智能的早期是一个巨大的挑战。

人类利用在任务/问题领域积累的大量知识来解决这些任务/问题。鉴于这一观察，随后的人工智能系统依靠一个大型的知识库捕捉有关问题/任务领域的知识。需要指出的一点是这里使用的术语是知识，而不是信息或数据。通过知识，我们利用程序/算法推理出新的数据/信息。关于这方面的一个例子是地图的图形表示，边界标有距离和交通（并不断更新），程序可以因此推理点之间的最短路径。

这种基于知识并且其中的知识由专家编写，同时以某种方式允许算法/程序推断知识的系统代表第二代 AI。其核心是使用越来越复杂的方法来表示和推理知识，并以此来帮助完

成/解决需要这些知识的任务/问题。这种复杂的例子包括使用一阶逻辑对知识进行编码和概率表示，以捕捉和推理领域固有的不确定性。

这种系统在一定程度上面临的关键挑战是如何解决许多领域固有的不确定性。人类在未知和不确定的环境中推理的能力比较强。一个重要观察是我们关于一个领域的知识不是黑色或白色，而是灰色。在这个不确定的时代，人类已经取得了很多进展。例如，在诊断疾病方面获得了一些成功，这些成功依赖于在未知和不确定的情况下利用知识进行新的推理。

这种系统的关键局限是需要专家手动编写有关领域的知识。收集、编辑和维护这些知识库使这些系统不切实际。在某些领域，甚至收集和编辑这些知识（例如，将语音转换成文本或将文档从一种语言翻译成另一种语言）都非常困难。虽然人类可以很容易地学会完成这样的工作，但是手工编写和编码相关的知识（例如，英语的语法知识，口音和主题）则非常具有挑战性。

人类通过获取关于任务/问题领域的知识来完成这些任务，这个过程被称为学习。鉴于这一观察，在这 10～20 年，AI 性能的提升归功于数据质量的提高。这个领域的重点是开发算法来获得给定数据的任务/问题领域的相关知识。需要注意的是，这种知识获取依赖于数据本身和人为定义的标记数据的方法。例如，考虑诊断某种疾病的问题。对于这样一个任务，一个人类专家会收集很多病例，例如病人有没有得过这种疾病。然后，人类专家将确定一些有助于进行预测的特征，例如患者的年龄、性别以及来自诸如血压、血糖等一些诊断测试的结果。人类专家将编译所有这些数据，并以适当的方式表示，例如对数据进行缩放/标准化等。一旦准备好这些数据，机器学习算法就可以通过从标记数据中进行归纳来学习如何推断患者是否患有该疾病。请注意，标记数据由有疾病和没有疾病的患者组成。所以，ML 算法本质上是将输入（年龄、性别、来自诊断测试的数据等特征）映射到正确结果（有疾病或没有疾病）的数学函数。找到这个最简单的数学函数并以一定的精度预测输出是 ML 领域的核心。学习任务需要多少例子或者算法的时间复杂度等具体问题，ML 在其领域已经提供了理论上的答案。如果有足够的数据，足够的计算资源和人力资源来设计特征，那么这个领域已经成熟到可以解决一大类问题。

主流 ML 算法的局限在于，将它们应用到一个新的领域需要大量的特征工程。例如，考虑识别图像中对象的问题。使用传统的 ML 技术，这样的问题将需要大量的特征工程。其中专家指定并给出 ML 算法需要使用的特征。从某种意义上说，真正的智能在于识别特征，而 ML 算法正在做的是简单地学习如何结合这些特征来得出正确的答案。在应用 ML 算法之前，领域专家的特征识别或数据表示是 AI 概念上和实用层面的“瓶颈”。

它是概念上的“瓶颈”，是因为如果领域专家识别特征，ML 算法只是学习结合并从中得出结论，这真的是 AI 吗？它是实用层面的“瓶颈”，是因为传统 ML 构建模型的过程受到特征工程的限制。在这个问题上可以投入的人力有限度。

人类从原始数据开始学习概念。例如，向一个孩子展示一种特定动物（例如猫）的一些例子/实例，他就能很快学会识别猫。学习过程不涉及父母给出猫的特征，如它有胡须，或有毛皮或有尾巴。人类学习从原始数据到得到结论没有明确的步骤，其中特征是自己学到的。从某种意义上说，人类从数据本身学习出数据表示。此外，他们将概念组织成一个层次结构，用复杂的概念表达原始的概念。

深度学习的领域主要侧重于学习合适的数据表示，以便这些数据可以用来得出结果。深度学习中的“深度”这个词是指从原始数据直接学习概念层次的思想。对于深度学习来说，一个技术上更为合适的术语是表示学习；更实用的术语则是自动特征工程。

1.2　相关领域的进展

重要的是要注意其他领域的进步，这些领域对深度学习的近期兴趣点和其成功起着关键的作用。有以下几点需要注意。

（1）在过去的十年中，收集、存储和操作大量数据的能力大大提高（例如，Apache Hadoop 生态系统）。

（2）生成有监督数据的能力（指带有标签的数据——例如标注了所描述的对象图片）随着众包服务的普及（如亚马逊劳务分包平台（Amazon Mechanical Turk，AMT））已经得到大幅度提高。

（3）由图形处理器单元带来的计算能力的巨大改进。

（4）自动微分的理论和软件实现方面的进展（如 Theano）。

虽然这些进步是深度学习之外的，但它们在促进深度学习方面发挥了重要作用。

1.3　先 决 条 件

阅读本书的关键先决条件是 Python 的知识，以及有关线性代数、微积分和概率的一些课程。建议读者在进行需要涵盖这些先决条件时参考以下内容。

Pyhton：Mark Pilgrim 编写的 *Dive Into Python*。

线性代数：Gilbert Strang 编写的 *Linear Algebra*。

微积分：Gilbert Strang 编写的 *Calculus*。

概率：Larry Wasserman 编写的 *All of Statistics*。

1.4　后续章节概述

下面为读者提供后续章节的整体概述。要注意每一章都强调一些深度学习的概念或技能，以便引起读者足够的重视。强烈建议读者不仅阅读相关章节，还要推导数学细节（使用笔和纸），并使用每章提供的源代码。

（1）第 2 章介绍机器学习的基础知识。本章的关键在于概括了测试集的例子、过拟合和欠拟合训练数据的思想、模型的能力以及正则化的概念。

（2）第 3 章介绍前馈神经网络，并作为整本书的基础概念。神经网络的整体结构、输入、隐藏层和输出层、损失函数及最大似然原理的基础等概念是本章的重要概念。

（3）第 4 章提供了 Theano 函数库的实践介绍。它涵盖了如何将网络定义为计算图，自动求导复杂网络的梯度，并训练神经网络。

（4）第 5 章介绍卷积神经网络，这也许是深度学习最成功的应用。

（5）第 6 章介绍递归神经网络和长短时间记忆（LSTM）网络，这是深度学习的又一次成功应用。

（6）第 7 章提供了对 Keras 库的实践介绍。Keras 库是对 Theano 库的高级抽象，它可能是构建深度学习应用程序的理想工具。

（7）第 8 章向读者介绍随机梯度下降（SGD），这是训练神经网络最重要的部分。本章还介绍了随机梯度下降的不足之处，以及对于解决这些不足的一些变种。

（8）第 9 章向读者介绍自动微分（通常称为反向传播），这是一种用于求导任意复杂网络的梯度（SGD）的标准技术。

（9）第 10 章向读者介绍了图形处理器（GPU）和基于 GPU 的计算，这些都是深度学习的关键技术。

1.5　安装所需函数库

读者需要安装一些库才能运行这些章节中例子的源代码。建议读者安装 Anaconda Python Distribution（https://www.continuum.io/downloads），这会让安装所需软件包的过程非常简单（使用 conda 或 pip）。读者需要的软件包列表包括 Scikit-learn、Theano、Autograd、Keras 和 PyOpenCL。

第 2 章　机器学习基础

深度学习是机器学习的一个分支，本章将介绍机器学习的基础知识。虽然机器学习作为一门学科本质上偏数学，但是我们将所需要的数学知识控制在需要掌握的最低限度。学习本章涵盖的主题的先决条件是线性代数、多变量微积分和基本概率论。

2.1　直　　觉

作为人类，我们直觉上可以意识到学习是什么：它意味着在一段时间内能够在某项任务上做得更好。这项任务可以是物理意义上的（如学习驾驶汽车）或智力层面的（如学习一门新的语言）。机器学习的核心是开发出可以像人类一样学习的算法；即随着时间的推移，通过得到更多的经验，结果也会随着时间的推移而变得更好。

也许很多人想知道如何根据现有经验改进自己的算法。毕竟有很多算法是为了解决现实世界中的问题而开发和实现，它们由人类开发并在软件中实现。从银行业务到电子商务，从汽车导航系统到月球上的航天器，算法无处不在，而且大多数算法不会随着时间的推移而改进。这些算法简单地执行计划任务，并且需要经常进行一些维护。所以为什么我们需要机器学习？

这个问题的答案是，对于某些任务来说，开发从经验中学习并改进其性能的算法比手动开发完整的算法更容易。虽然这对读者来说可能看起来并不直观，但我们将在本章中对此做出解释。

2.2　二 元 分 类

为了进一步讨论这个问题，我们需要对之前使用的一些术语（如任务、学习、经验和改进）进行精确的描述。下面将从二元分类的任务开始。

考虑一个抽象的问题，我们有如下的数据形式。

$$D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$$

其中 $x\in\mathbb{R}^n$ 并且 y=±1。我们无法访问所有数据，只能访问其中的一部分 $S\in D$。使用 S，我们的任务是生成一个计算过程表示函数 $f:x\to y$，这样对于测试集的数据 x_i,y_i，用函数 f 来预测 $f(x_i)=y_i$。用 $S\in D$ 表示测试集的数据，即 $(x_i,y_i)\notin S$ 和 $(x_i,y_i)\in U$。我们利用在训练集的数据上的表现作为性能的衡量标准，

$$E(f,D,U)=\frac{\sum_{(x_i,y_i)\in U}[f(x_i)\neq y_i]}{|U|}$$

现在对当前任务做一个精确的定义，基于训练集的数据集 S 得到函数 f 来把数据分成两类(y=±1)。使用误差 $E(f, D, U)$表示在测试集的数据 U 上衡量性能（以及改善性能）。可见数据的大小$|S|$是经验值。基于这个定义，我们想要开发生成函数 f（通常被称为模型）的算法。一般来说，机器学习领域研究的是这种算法的开发，这些算法产生的模型可以在测试集的数据上做出预测，还有一些其他正式的任务（本章后面我们将介绍多个这样的任务）。请注意，x 通常被称为输入/输入变量，y 被称为输出/输出变量。

和其他计算机科学一样，这种算法的可计算性是一个重要的方面，但除此之外，我们也希望有一个模型 f，以尽可能小的$|S|$达到一个小的误差 $E(f, D, U)$。

现在我们把这个抽象而精确的定义与现实世界的问题联系起来，使抽象有基础。比如说一个电子商务网站想要为注册用户定制登录页面，向他们展示用户可能有兴趣购买的产品。该网站有用户的历史数据，并希望作为一个功能以增加销售。那么这个现实世界的问题是如何映射到我们之前描述的二元分类的抽象问题的。

人们可能会注意到的第一件事是，给定一个特定的用户和一个特定的产品，人们想要预测用户是否会购买该产品。因为这是要预测的值，所以它映射到 y=±1，我们把 y=±1 的值表示预测用户是否将购买产品，y=−1 是预测用户不会购买该产品。请注意，选择这个值没有特别的理由。我们交换这两个值（让 y=+1 表示未购买和 y=−1 表示购买）也没有区别。我们只是用 y=±1 来表示数据分类的两类。接下来，假设可以以某种方式将产品的属性、用户的购买和浏览历史记录表示为 $x\in\mathbb{R}^n$，这一步在机器学习中被称为特征工程将在本章后面介绍。目前来说，我们可以得到这样的映射就足够。我们把用户浏览、购买的历史数据、产品的属性及用户是否购买了该产品映射为 $\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$。基于这些数据，我们希望生成一个函数或模型 $f:x\to y$，使用它来确定特定用户将购买哪些产品，并使用它来填充用户的登录页面。通过填充用户的登录页面，查看他们是否购买产品，并评估错误 $E(f, D, U)$来衡量模型对于测试集的数据产生的影响。

2.3　回　　归

机器学习中另一个任务即回归。对于数据 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$，其中 $x\in\mathbb{R}^n$ 和 $x\in\mathbb{R}$。目标是生成一个计算过程来实现函数 $f:x\to y$。请注意，函数 f 和二元分类不同，预测的结果不是二元类标签 y=±1，而是实值的预测。将未知数据上的均方根误差（RMSE）作为衡量该任务的性能标准，

$$E(f,D,U)=\left(\frac{\sum_{(x_i,y_i)\in U}(y_i-f(x_i))^2}{|U|}\right)^{\frac{1}{2}}$$

注意　RMSE 只是简单地把预测值和实际值区分开，将它们平方取均值可以让正负值被一样处理，然后把平均值集中在所有测试集的数据上，最后取平方根来平衡平方的操作。

与抽象的回归任务对应的现实世界问题是基于个人的财务历史预测其信用评分，信用卡公司可以根据该评分来增大其信用额度。

2.4　泛　　化

机器学习中最重要的一个直觉，这是开发一个在测试集的数据上也有良好表现的模型所必要的。为了做到这一点，将先介绍一个回归任务的示例数据集。

为了生成示例数据集，我们通过在-1 和 1 之间等距地生成 100 个值作为输入变量(x)，根据 $y=2+x+2x^2+\epsilon$ 生成输出变量（y），其中 $\epsilon\sim\mathcal{N}(0,0.1)$ 是从均值为 0、标准差为 0.1 的正态分布中采样的噪声（随机变化），代码如清单 2-1 所示，数据如图 2-1 所示。

清单 2-1　泛化 vs 机器学习

```
# 生成示例数据集
import pylab
import numpy
x = numpy.linspace(-1,1,100)
signal = 2 + x + 2 * x * x
noise = numpy.random.normal(0,0.1,100)
y = signal + noise
pylab.plot(signal,'b');
pylab.plot(y,'g')
pylab.plot(noise,'r')
pylab.xlabel("x")
pylab.ylabel("y")
pylab.legend(["Without Noise", "With Noise", "Noise"], loc = 2)
x_train = x[0:80]
y_train = y[0:80]

# 模型度为 1
pylab.figure()
degree = 2
```

```
X_train = numpy.column_stack([numpy.power(x_train,i) for i in xrange(0,degree)])
model = numpy.dot(numpy.dot(numpy.linalg.inv(numpy.dot(X_train.transpose(),X_train)),
    X_train.transpose()),y_train)
pylab.plot(x,y,'g')
pylab.xlabel("x")
pylab.ylabel("y")
predicted = numpy.dot(model, [numpy.power(x,i) for i in xrange(0,degree)])
pylab.plot(x, predicted,'r')
pylab.legend(["Actual", "Predicted"], loc = 2)
train_rmse1 = numpy.sqrt(numpy.sum(numpy.dot(y[0:80] - predicted[0:80], y_train -
    predicted[0:80])))
test_rmse1 = numpy.sqrt(numpy.sum(numpy.dot(y[80:] - predicted[80:], y[80:] -
    predicted[80:])))
print("Train RMSE (Degree = 1)", train_rmse1)
print("Test RMSE (Degree = 1)", test_rmse1)

# 模型度为 2
pylab.figure()
degree = 3
X_train = numpy.column_stack([numpy.power(x_train,i) for i in xrange(0,degree)])
model = numpy.dot(numpy.dot(numpy.linalg.inv(numpy.dot(X_train.transpose(),X_train)),
    X_train.transpose()),y_train)
pylab.plot(x,y,'g')
pylab.xlabel("x")
pylab.ylabel("y")
predicted = numpy.dot(model, [numpy.power(x,i) for i in xrange(0,degree)])
pylab.plot(x, predicted,'r')
pylab.legend(["Actual", "Predicted"], loc = 2)
train_rmse1 = numpy.sqrt(numpy.sum(numpy.dot(y[0:80] - predicted[0:80],
    y_train - predicted[0:80])))
test_rmse1 = numpy.sqrt(numpy.sum(numpy.dot(y[80:] - predicted[80:],
    y[80:] - predicted[80:])))
print("Train RMSE (Degree = 2)", train_rmse1)
print("Test RMSE (Degree = 2)", test_rmse1)

# 模型度为 8
pylab.figure()
degree = 9
X_train = numpy.column_stack([numpy.power(x_train,i) for i in xrange(0,degree)])
model = numpy.dot(numpy.dot(numpy.linalg.inv(numpy.dot(X_train.transpose(),X_train)),
    X_train.transpose()), y_train)
pylab.plot(x,y,'g')
pylab.xlabel("x")
```

```
pylab.ylabel("y")
predicted = numpy.dot(model, [numpy.power(x,i) for i in xrange(0,degree)])
pylab.plot(x, predicted,'r')
pylab.legend(["Actual", "Predicted"], loc = 3)
train_rmse2 = numpy.sqrt(numpy.sum(numpy.dot(y[0:80] - predicted[0:80],
    y_train - predicted[0:80])))
test_rmse2 = numpy.sqrt(numpy.sum(numpy.dot(y[80:] - predicted[80:],
    y[80:] - predicted[80:])))
print("Train RMSE (Degree = 8)", train_rmse2)
print("Test RMSE (Degree = 8)", test_rmse2)

# 输出
Train RMSE (Degree = 1) 3.50756834691
Test RMSE (Degree = 1) 7.69514326946
Train RMSE (Degree = 2) 0.91896252959
Test RMSE (Degree = 2) 0.446173435392
Train RMSE (Degree = 8) 0.897346255079
Test RMSE (Degree = 8) 14.1908525449
```

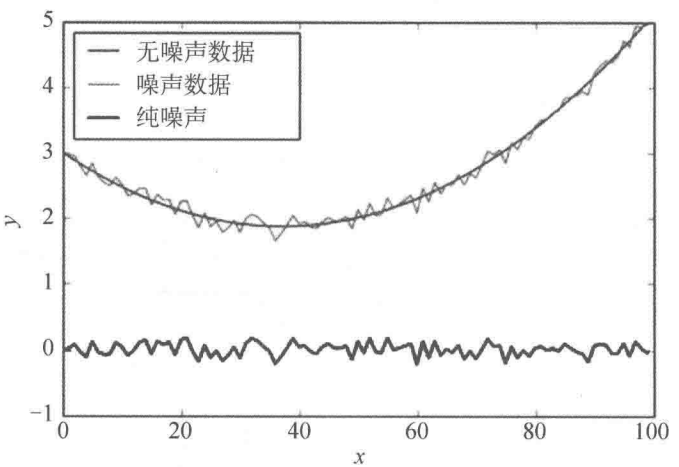

图 2-1　生成用于回归问题的示例数据集

为了模拟训练集和测试集的数据，我们使用前 80 个数据点作为训练集的数据，将其余数据点看作测试集的数据。即只用前 80 个数据点建立模型，剩下的数据点用来评估模型。

接下来，我们使用一个非常简单的算法来生成一个模型，通常称为最小二乘。给定一个形式为 $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$ 的数据集，其中 $x\in\mathbb{R}^n$，$y\in\mathbb{R}$，最小二乘模型采取形式为 $y=\beta x$，其中 β 是向量，目标是使得 $\|X\beta-y\|_2^2$ 最小化。X 矩阵的每一行就是 x，

因此 $X \in \mathbb{R}^{m \times n}$，$m$ 是样本的数量（在例子中是 80）。β 的值可以根据闭式 $\beta = (X^T X)^{-1} X^T y$ 得到。我们正在讨论最小二乘法的许多重要细节，其中更重要的是我们如何将输入变量转换为合适的形式。在第一个模型中，我们将x变换为值$[x^0, x^1, x^2]$的向量。即如果 $x = 2$，它将被转换为[1, 2, 4]。接着我们可以使用上面描述的公式生成一个最小二乘模型 β。我们使用二阶多项式（度=2）方程近似给定的数据，最小二乘算法可以简单地进行曲线拟合或为$[x^0, x^1, x^2]$的每个元素生成系数。

我们可以使用 RMSE 方法来评估模型在测试集数据上的表现。当然也可以在训练集数据上计算 RMSE 值。实际值和预测值如图 2-2 所示，清单 2-1 展示了生成模型的源代码。

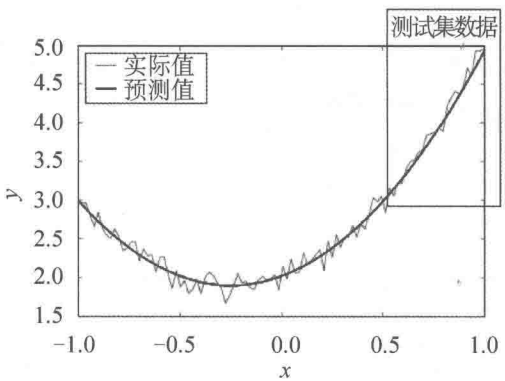

图 2-2　度为 2 的模型的实际值和预测值

接下来，用最小二乘算法生成另一个模型，将 x 转换成$[x^0, x^1, x^2, x^3, x^4, x^5, x^6, x^7, x^8]$。即，用度为 8 的多项式近似给定的数据。实际值和预测值如图 2-3 所示，清单 2-1 显示了生成模型的源代码。作为最后一步，生成一个度为 1 的模型。

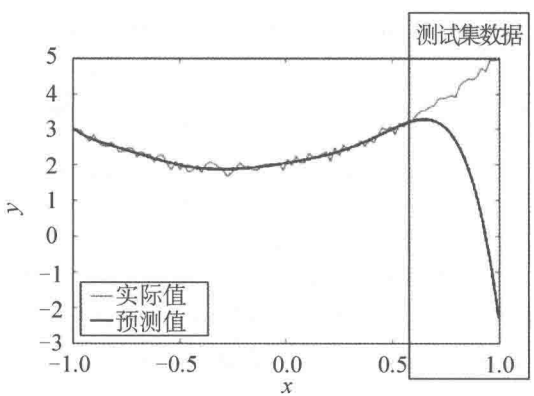

图 2-3　度为 8 的模型的实际值和预测值