TURING 图灵程序设计丛书

机器学习实践
测试驱动的开发方法

Thoughtful Machine Learning
A Test-Driven Approach

[美] Matthew Kirk 著
段菲 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo
O'Reilly Media, Inc.授权人民邮电出版社出版

人民邮电出版社
北　京

图书在版编目（CIP）数据

机器学习实践 : 测试驱动的开发方法 / (美) 柯克(Kirk,M.) 著 ; 段菲译. -- 北京 : 人民邮电出版社, 2015.8
（图灵程序设计丛书）
ISBN 978-7-115-39618-1

Ⅰ. ①机… Ⅱ. ①柯… ②段… Ⅲ. ①机器学习－研究 Ⅳ. ①TP181

中国版本图书馆CIP数据核字(2015)第148307号

内 容 提 要

本书主要介绍如何将测试驱动开发运用于机器学习算法。每一章都通过示例介绍了机器学习技术能够解决的有关数据的具体问题，以及求解问题和处理数据的方法。具体涵盖了测试驱动的机器学习、机器学习概述、*K* 近邻分类、朴素贝叶斯分类、隐马尔可夫模型、支持向量机、神经网络、聚类、核岭回归、模型改进与数据提取等内容。通过学习本书，你将能够利用机器学习技术解决涉及数据的现实问题。

本书适合开发人员、CTO 和商业分析师阅读。

◆ 著　[美] Matthew Kirk
译　段　菲
责任编辑　岳新欣
责任印制　杨林杰

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路11号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　印刷

◆ 开本：800×1000　1/16
印张：12.75
字数：302千字　2015年 8 月第 1 版
印数：1－3 500册　2015年 8 月北京第 1次印刷
著作权合同登记号　图字：01-2015-3679号

定价：49.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

版权声明

O'Reilly Media, Inc.介绍

O'Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O'Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名；创立了 Make 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly 现在还将先锋专家的知识传递给普通的计算机用户。无论是通过书籍出版、在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar 博客有口皆碑。”

——*Wired*

“O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——*Business 2.0*

“O'Reilly Conference 是聚集关键思想领袖的绝对典范。”

——*CRN*

“一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——*Irish Times*

“Tim 是位特立独行的商人，他不光放眼于最长远、最广阔的视野，并且切实地按照 Yogi Berra 的建议去做了：‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去，Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——*Linux Journal*

目录

前言

这是一本介绍如何解决棘手问题的书。机器学习是计算技术的一项令人叹为观止的应用，因为它要解决的很多问题都源自科幻小说。机器学习算法可用于解决语音识别、映射、推荐以及疾病检测等复杂问题。机器学习的应用领域浩瀚无垠，也正因为如此它才这样引人入胜。

然而，这种灵活性也使得机器学习技术令人望而却步。它的确可以解决许多问题，但如何知晓我们求解的是否是正确的问题，或者是否应最先求解某个问题呢？此外，令人沮丧的是，大部分学术编码标准都不够严密。

即便是在今天，人们仍未对如何编写高质量的机器学习代码给予足够的关注，这是无比遗憾的。将一种观念在整个行业广泛传播的能力取决于有效沟通的能力。如果我们编写的代码本身就质量低劣，那么恐怕不会有很多人愿意聆听我们的讨论。

本书便是我对于该问题的回答。我试图按照容易理解的方式为大家讲授机器学习。这门学科本身难度就不小，况且我们还需要阅读代码，尤其是那些极难理解的古老 C 实现，无疑更是雪上加霜。

很多读者会对本书采用 Ruby 而非 Python 感到非常困惑。我的理由是用 Ruby 编写测试程序是一种诠释你所写代码的美妙方式。本质上，这本通篇采用测试驱动方法的书讲述的是如何沟通，具体说来是如何与机器学习这个奇妙的世界沟通。

从本书可学到的知识

应该说，本书对机器学习的介绍并不全面。因此，我向你强烈推荐 Peter Flach 编著的 *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*（Cambridge University Press）[1]。如果你希望读一些数学味道较浓的书，可参阅 Tom Mitchell 的 Machine

注 1：中文版即将由人民邮电出版社出版。——编者注

Learning 系列。此外，你还可参考 Stuart Russel 和 Peter Norvig 编写的有关人工智能的名著 *Artificial Intelligence: A Modern Approach*, 3rd Edition（Prentice Hall）。

阅读完本书之后，你并不会获得机器学习的博士学位，但我希望本书能向你传授足够的知识，帮助你开始研究如何利用机器学习技术解决涉及数据的现实问题。对于求解问题的方法以及如何在基础层面上使用它们，本书会提供大量相关示例。

你还将掌握如何解决那些比普通的单元测试更为模糊的复杂问题。

本书的阅读方法

阅读本书的最佳方法是找到一些能够让你感到兴奋的例子。我力图每一章都介绍一些这样的例子，虽然有时无法尽如人意。我不希望本书过于偏重理论，而是希望通过示例向你介绍机器学习能够解决的一些具体问题，以及我处理数据的方法。

在本书的大多数章中，我都试图在一开始便引入一个商业案例，之后开始研究一个可求解的问题，直到结尾。本书是一部短篇读物，因为我希望你能够专注于理解书中的代码，并深入思考这些问题，而非沉浸在理论当中。

本书读者

本书面向的读者包括三个群体：开发人员、CTO 以及商业分析师。

开发人员已经熟知如何编写代码，且想更多地了解机器学习这个激动人心的领域。他们拥有在计算环境中求解问题的背景，可能使用过 Ruby 编写代码，也可能从未接触过这种语言。他们是本书的主要读者群，但本书在写作时也兼顾了 CTO 和商业分析师。

CTO 是那些真正希望了解如何利用机器学习技术提升公司业务的人。他们可能听说过 *K* 均值算法、*K* 近邻算法，但不知道这些算法的适用性。商业分析师与之相似，只是不像 CTO 那样关注技术细节。为了这两个读者群，我在每章的开头都准备了一个商业案例。

作者联系方式

如果你喜欢我之前所做的演讲，或想为某个问题寻求帮助，欢迎通过电子邮件与我联系。我的邮箱是 matt@matthewkirk.com。为了增进我们之间的联系，如果你来到西雅图地区，且我们的日程允许的话，我非常乐意请你喝一杯咖啡。

如果希望查看本书的所有代码，可从 GitHub 站点 http://github.com/thoughtfulml 免费下载。

排版约定

本书使用了下列排版约定。

- 楷体
  表示新术语或突出强调的内容。
- 等宽字体（`constant width`）
  表示程序片段，以及正文中出现的变量、函数名、数据库、数据类型、环境变量、语句和关键字等。
- 加粗等宽字体（**`constant width bold`**）
  表示应该由用户输入的命令或其他文本。

该图标表示提示或建议。

该图标表示一般注记。

该图标表示警告或警示。

该图标表示非常重要的警告，请仔细阅读。

使用代码示例

补充材料（代码示例、练习等）可以从 http://github.com/thoughtfulml 下载。

本书是要帮你完成工作的。一般来说，如果本书提供了示例代码，你可以把它用在你的程序或文档中。除非你使用了很大一部分代码，否则无需联系我们获得许可。比如，用本书

的几个代码片段写一个程序就无需获得许可，销售或分发 O'Reilly 图书的示例光盘则需要获得许可；引用本书中的示例代码回答问题无需获得许可，将书中大量的代码放到你的产品文档中则需要获得许可。

我们很希望但并不强制要求你在引用本书内容时加上引用说明。引用说明一般包括书名、作者、出版社和 ISBN。比如："*Thoughtful Machine Learning* by Matthew Kirk (O'Reilly). Copyright 2015 Matthew Kirk, 978-1-449-37406-8"。

如果你觉得自己对示例代码的用法超出了上述许可的范围，欢迎你通过 permissions@oreilly.com 与我们联系。

Safari® Books Online

Safari Books Online（http://www.safaribooksonline.com）是应运而生的数字图书馆。它同时以图书和视频的形式出版世界顶级技术和商务作家的专业作品。技术专家、软件开发人员、Web 设计师、商务人士和创意专家等，在开展调研、解决问题、学习和认证培训时，都将 Safari Books Online 视作获取资料的首选渠道。

对于组织团体、政府机构和个人，Safari Books Online 提供各种产品组合和灵活的定价策略。用户可通过一个功能完备的数据库检索系统访问 O'Reilly Media、Prentice Hall Professional、Addison-Wesley Professional、Microsoft Press、Sams、Que、Peachpit Press、Focal Press、Cisco Press、John Wiley & Sons、Syngress、Morgan Kaufmann、IBM Redbooks、Packt、Adobe Press、FT Press、Apress、Manning、New Riders、McGraw-Hill、Jones & Bartlett、Course Technology 以及其他几十家出版社的上千种图书、培训视频和正式出版之前的书稿。要了解 Safari Books Online 的更多信息，我们网上见。

联系我们

请把对本书的评价和问题发给出版社。

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）
奥莱利技术咨询（北京）有限公司

O'Reilly 的每一本书都有专属网页，你可以在那儿找到本书的相关信息，包括勘误表、示例代码以及其他信息。本书的网站地址是：

http://shop.oreilly.com/product/0636920032298.do

对于本书的评论和技术性问题，请发送电子邮件到：bookquestions@oreilly.com

要了解更多 O'Reilly 图书、培训课程、会议和新闻的信息，请访问以下网站：

http://www.oreilly.com

我们在 Facebook 的地址如下：http://facebook.com/oreilly

请关注我们的 Twitter 动态：http://twitter.com/oreillymedia

我们的 YouTube 视频地址如下：http://www.youtube.com/oreillymedia

致谢

- Mike Loukides，对于我的利用测试驱动开发来编写机器学习代码的想法，他表现出了浓厚的兴趣。
- 我的编辑 Ann Spencer，在我撰写本书的数月间，她教给了我大量编辑知识和技巧，并针对本书的设计提出了宝贵意见。

我要感谢 O'Reilly 团队的所有成员，是他们成就了本书。我尤其要感谢下面这些人。

我的审稿人：

- Brad Ediger，当我提出要编写一本关于基于测试驱动的机器学习代码的书籍时，他对于这一古怪想法兴奋不已，并对本书的初稿提出了大量极有价值的意见。
- Starr Horne，他在审阅过程中向我提供了大量真知灼见。感谢他在电话会议中与我讨论了机器学习、错误报告等。
- Aaron Sumner，他针对本书的代码结构提出了极有价值的意见。

我极为出色的合作者，以及在本书编写过程中提供帮助的朋友们：Edward Carrel、Jon-Michael Deldin、Christopher Hobbs、Chris Kuttruff、Stefan Novak、Mike Perham、Max Spransy、Moxley Stratton 以及 Wafa Zouyed。

如果没有家人的支持和鼓励，我不可能顺利完成本书。

- 献给我的妻子 Sophia，是她帮助我坚持梦想，并帮助我将对本书的设想变成现实。
- 献给我的祖母 Gail，在我幼年时，是她引导我对学习产生了浓厚的兴趣。我还记得在一次公路旅行中，她心无旁骛地向我询问一些关于我正在看的那本“咖啡书”（实际上是讲 Java 的书）的问题。

- 献给我的父亲 Jay 和母亲 Carol，是他们教会了我如何分析系统，以及如何为其注入人类的情感。
- 献给我的弟弟 Jacob 以及侄女 Zoe 和 Darby，感谢他们教会了我如何通过孩子的眼光重新认识世界。

最后，我将本书献给科学和对知识不断求索的人。

第 1 章

测试驱动的机器学习

伟大的科学家既是梦想家，也是怀疑论者。在现代历史中，科学家们取得了一系列重大突破，如发现地球引力、登上月球、发现相对论等。所有这些科学家都有一个共同点，那就是他们都有着远大的梦想。然而，在完成那些壮举之前，他们的工作无不经过了周密的检验和验证。

如今，爱因斯坦和牛顿已离我们而去，但所幸我们处在一个大数据时代。随着信息时代的到来，人们迫切需要找到将数据转化为有价值的信息的方法。这种需求的重要性已日益凸显，而这正是数据科学和机器学习的使命。

机器学习是一门充满魅力的学科，因为它能够利用信息来解决像人脸识别或笔迹检测这样的复杂问题。很多时候，为完成这样的任务，机器学习算法会采用大量的测试。典型的测试包括提出统计假设、确定阈值、随着时间的推移将均方误差最小化等。理论上，机器学习算法具备坚实的理论基础，可从过去的错误中学习，并随时间的推移将误差最小化。

然而，我们人类却无法做到这一点。机器学习算法虽能将误差最小化，但有时我们可能“指挥失误”，没能令其将“真正的误差”最小化，我们甚至可能在自己的代码中犯一些不易察觉的错误。因此，我们也需要通过一些测试来发现自己所犯的错误，并以某种方式来记录我们的进展。用于编写这类测试的最为流行的方法当属*测试驱动开发*（Test-Driven Development，TDD）。这种“测试先行”的方法已成为编程人员的一种最佳实践。然而，这种最佳实践有时在开发环境中却并未得到运用。

采用驱动测试开发（为简便起见，下文中统称 TDD）有两个充分的理由。首先，虽然在主动开发模式中，TDD 需要花费至少 15%~35% 的时间，却能够排除多达 90% 的程序缺陷

（详情请参阅 http://research.microsoft.com/en-us/news/features/nagappan-100609.aspx）。其次，采用 TDD 有利于将代码准备实现的功能记录下来。当代码的复杂性增加时，人们对规格说明的需求也愈发强烈，尤其是那些需要依据分析结果制定重大决策的人。

哈佛大学的两位学者 Carmen Reinhart 和 Kenneth Rogoff 曾撰写过一篇经济学论文，大意是说那些所承担的债务数额超过其国内生产总值 90% 以上的国家的经济增长遭遇了严重滑坡。后来，Paul Ryan 在总统竞选中还多次引用了这个结论。2013 年，麻省大学的三位研究者发现该论文的计算有误，因为在其分析中有相当数量的国家未被考虑。

这样的例子还有很多，只是可能情况不像这个案例这样严重。这个案例说明，统计分析中的一处错误可能会对一位学者的学术声誉造成打击。一步出错可能会导致多处错误。上面两位哈佛学者本身都具有多年的研究经历，而且这篇论文的发表也经过了严格的同行评审，但仍然出现了这样令人遗憾的错误。这样的事情在任何人身上都有可能发生。使用 TDD 将有助于降低犯类似错误的风险，而且可以帮助这些研究者避免陷入万分尴尬的境地。

1.1 TDD的历史

1999 年，Kent Beck 通过其极限编程（extreme programming）方面的工作推广了 TDD。TDD 的强大源自其先定义目标再实现这些目标的能力。TDD 的实践步骤如下：首先编写一项无法通过的测试（由于此时尚无功能代码，因此测试会失败），再编写可使其通过的功能代码，最后重构初始代码。一些人依据众多测试库的颜色将其称为“红－绿－重构”（red-green-refactor）。红色表示编写一项最初无法通过的测试，而你需要记录自己的目标；绿色表示通过编写功能代码使测试通过。最后，对初始代码进行重构，直到自己对其设计感到满意。

在传统开发实践中，测试始终是中流砥柱，但 TDD 强调的是“测试先行”，而非在开发周期即将结束时才考虑测试。瀑布模型采用的是验收测试（acceptance test），涉及许多人员，通常是大量最终用户（而非开发人员），且该测试发生在代码实际编写完毕之后。如果不将功能覆盖范围作为考虑因素，这种方法看起来的确很好。很多时候，质量保证专业人员仅对他们感兴趣的方面进行测试，而非进行全面测试。

1.2 TDD与科学方法

TDD 之所以如此引人瞩目，部分原因在于它能够与人们及其工作方式保持良好的同步。它所遵循的“假设－测试－理论探讨”流程使之与科学方法有诸多相似之处。

科学需要反复试验。科学家在工作中也都是首先提出某个假设，接着检验该假设，最后将这些假设升华到理论高度。

我们也可将“假设－测试－理论探讨”这个流程称为“红－绿－重构”。

与科学方法一样，对于机器学习代码，测试（检验）先行同样适用。大多数机器学习践行者都会运用某种形式的科学方法，而 TDD 会强制你去编写更加清晰和稳健的代码。实际上，TDD 与科学方法的关系绝不止于相似。本质上，TDD 是科学方法的一个子集，理由有三：一是它需要构建有效的逻辑命题；二是它通过文档共享结果；三是它采用闭环反馈的工作机制。

TDD 之美在于你也可利用它进行试验。很多时候，首先编写测试代码时，我们都抱有一个信念，即最初测试中遇到的那些错误最终一定可被修正，但实际上我们并非一定要遵循这种方式。我们可以利用测试来对那些可能永远不会被实现的功能进行试验。对于许多不易解决的问题，按照这种方式进行测试十分有用。

1.2.1 TDD可构建有效的逻辑命题

科学家们在使用科学方法时，首先尝试着去求解一个问题，然后证明方法的有效性。问题求解需要创造性猜想，但如果没有严格的证明，它只能算是一种“信念”。

柏拉图认为，知识是一种被证明为正确的信念。我们不但需要正确的信念，而且也需要能够证明其正确的确凿证据。为证明我们的信念的正确性，我们需要构建一个稳定的逻辑命题。在逻辑学中，用于证明某个观点是否正确的条件有两种——必要条件和充分条件。

必要条件是指那些如果缺少了它们假设便无法成立的条件。例如，全票通过或飞行前的检查都属于必要条件。这里要强调的是，为确保我们所做的测试是正确的，所有的条件都必须满足。

与必要条件不同，充分条件意味着某个论点拥有充足的证据。例如，打雷是闪电的充分条件，因为二者总是相伴出现的，但打雷并不是闪电的必要条件。很多情形下，充分条件是以统计假设的形式出现的。它可能不够完善，但要证明我们所做测试的合理性已然足够充分了。

为论证所提出的解的有效性，科学家们需要使用必要条件和充分条件。科学方法和 TDD 都需要严格地使用这两种条件，以使所提出的一系列论点成为一个有机整体。然而，二者的不同之处在于，科学方法使用的是假设检验和公理，而 TDD 使用的则是集成和单元测试（参见表 1-1）。

表1-1：TDD与科学方法的比较

	科学方法	TDD
必要条件	公理	纯粹的功能测试
充分条件	统计假设检验	单元和集成测试

示例1：借助公理和功能测试完成证明

法国数学家费马于 1637 年提出著名的猜想[1]：对于大于 2 的任意整数 n，关于 a、b、c 的方程 $a^n + b^n + c^n$ 不存在正整数解。表面上看，这好像是一个比较简单的问题，而且据说费马声称他已经完成了证明。他在读书笔记中写道：“我确信已发现了一种美妙的证法，可惜这里空白的地方太小，写不下。”

此后的 358 年间，该猜想一直未得到彻底证实。1995 年，英国数学家安德鲁· 怀尔斯（Andrew Wiles）借助伽罗瓦（Galois）变换和椭圆曲线完成了费马大定理的最终证明。他长达 100 页的证明虽然称不上优雅，但每一步都经得起严格推敲。每一小节的论证都承前启后。

这 100 页证明中的每一步都建立在之前已被人们证明的公理和假设的基础之上，这与功能测试套件何其相似！用程序设计术语来说，怀尔斯在其证明中使用的所有公理和断言都可作为功能测试。这些功能测试只不过是以代码形式展现的公理和断言，每一步证明都是下一小节的输入。

大多数情况下，软件生产过程中并不缺少测试。很多时候，我们所编写的测试都是关于代码的随意的断言。许多情形下，为了使用以前的样例，我们只对打雷而不对闪电进行测试。即，我们的测试只关注了充分条件，而忽略了必要条件。

示例2：借助充分条件、单元测试和集成测试完成证明

与纯数学不同，充分条件只关心是否有足够的证据来支持某个因果关系。下面以通货膨胀为例来说明。自 19 世纪开始，人们已经在研究这种经济学中的神秘力量。要证明通货膨胀的存在，我们所面临的问题是并无任何公理可用。

不过，我们可以依据来自观察的充分条件来证明通货膨胀的确存在。我们观察过经济数据并从中分离出已知正确的因素，根据此经验，我们发现随着时间的推移，尽管有时也会下降，但长期来看经济是趋于增长的。通货膨胀的存在可以只通过我们之前所做的具有一致性的观察来证明。

在软件开发领域，经济学中的这类充分条件对应集成测试。集成测试旨在测试一段代码的首要行为。集成测试并不关心代码中微小的改动，而是观察整个程序，看所期望的行为是否能够如期发生。同样，如果将经济视为一个程序，则我们可断言通货膨胀或通货紧缩是存在的。

注 1：即我们熟知的费马大定理。——译者注

1.2.2 TDD要求你将假设以文字或代码的形式记录下来

学术机构通常要求教授发表其研究成果。虽然有很多人抱怨各大学过于重视发表文章，但其实这样做是合理的：发表是一种使研究成果成为永恒的方法。如果教授们决定独自研究并取得重大突破，却不将其成果发表，那么这种研究将无任何价值。

TDD 同样如此：测试在同行评审中能够发挥重要的作用，也可作为一个版本的文档。实际上很多时候，在使用 TDD 时，文档并不是必需的。由于软件具有抽象性，且总处在变化之中，因此如果某人没有将其代码文档化或对代码进行测试，将来它便极有可能被修改。如果缺乏能够保证这些代码按特定方式运行的测试，则当新的编程人员参与到该软件的开发和维护工作中时，将无法保证他不会改动代码。

1.2.3 TDD和科学方法的闭环反馈机制

科学方法和 TDD 均采用闭环反馈的机制。当某人提出一项假设，并对其进行检验（测试）时，他会找到关于自己所探索问题的更多信息。对于 TDD，也同样如此；某个人对其所想进行测试，之后当他开始编写代码时，对于如何进行便可以做到心中有数。

总之，TDD 是一种科学方法。我们提出一些假设，对其进行检验（测试），之后再重新检视。TDD 践行者们遵循的也是相同的步骤，即首先编写无法通过的测试，接着找到解决方案，然后再对这个解决方案进行重构。

示例：同行评审

许多领域，无论是学术期刊、图书出版，还是程序设计领域，都有自己的同行评审，且形式各异。原因编辑（reason editor）之所以极有价值，是因为他们对于一部作品或一篇文章而言是第三方，能够给出客观的反馈意见。在科学界，与之对应的则是对期刊文章的同行评审。

TDD 则不同，因为第三方是一个程序。当某人编写测试时，程序以代码的形式表示假设和需求，而且是完全客观的。在其他人查看代码之前，这种反馈对于程序开发人员检验其假设是很有价值的。此外，它还有助于减少程序缺陷和功能缺失。

然而，这并不能缓解机器学习或数学模型与生俱来的问题，它只是定义了处理问题和寻求足够好的解的基本过程。

1.3 机器学习中的风险

对于开发过程而言，虽然使用科学方法和 TDD 可提供良好的开端，但我们仍然可能遇到一些棘手的问题。一些人虽然遵循了科学方法，但仍然得到了错误的结果；TDD 可帮助我们创建更高质量的代码，且更加客观。接下来的几个小节将介绍机器学习中经常会遇到的几个重要问题：

- 数据的不稳定性
- 欠拟合
- 过拟合
- 未来的不可预测性

1.3.1 数据的不稳定性

机器学习算法通过将离群点最少化来尽量减少数据中的不稳定因素。但如果错误的来源是人为失误，该如何应对？如果错误地表示了原本正确的数据，最终将使结果偏离真实情况，从而产生偏倚。

对我们所拥有的不正确信息的数量予以考虑，这是一个重要的现实问题。例如，如果我们使用的某个应用程序编程接口（API）将原本表示二元信息的 0 和 1 修改为 –1 和 +1，则这种变化对于模型的输出将是有害的。对于时间序列，其中也可能存在一些缺失数据。这种数据中的不稳定性要求我们找到一种测试数据问题的途径，以减少人为失误的影响。

1.3.2 欠拟合

如果模型未考虑足够的信息，从而无法对现实世界精确建模，将产生欠拟合（underfitting）现象。例如，如果仅观察指数曲线上的两点，我们可能会断言这里存在一个线性关系（如图 1-1 所示）。但也有可能并不存在任何模式，因为只有两个点可供参考。

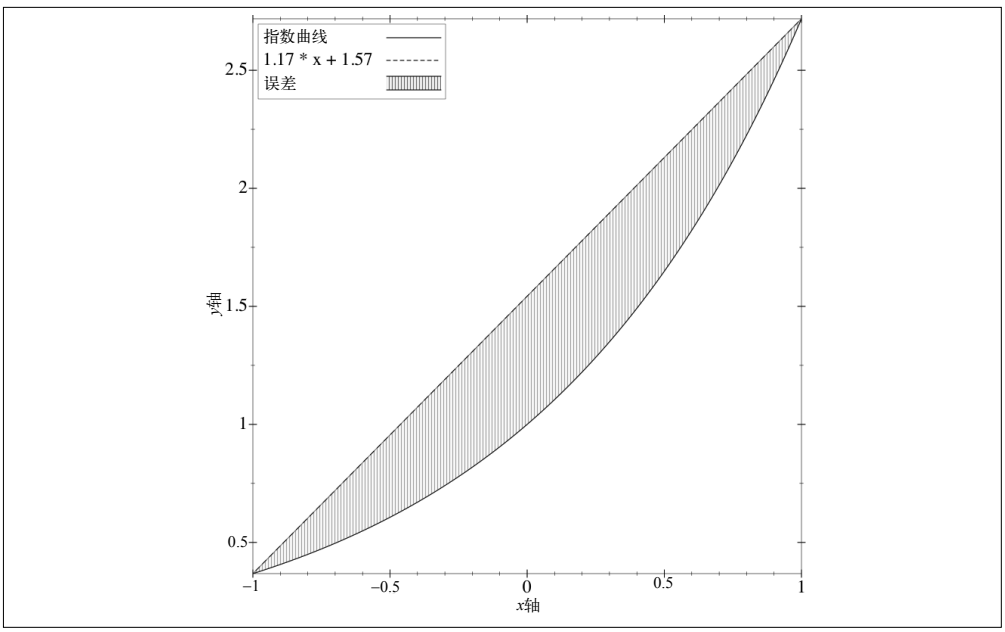

图 1-1：在 [–1,+1] 区间内，直线可对指数曲线取得良好的逼近效果

不幸的是，如果对该区间（[−1,+1]）进行扩展，将无法看到同样的逼近效果，取而代之的是显著增长的逼近误差（如图 1-2 所示）。

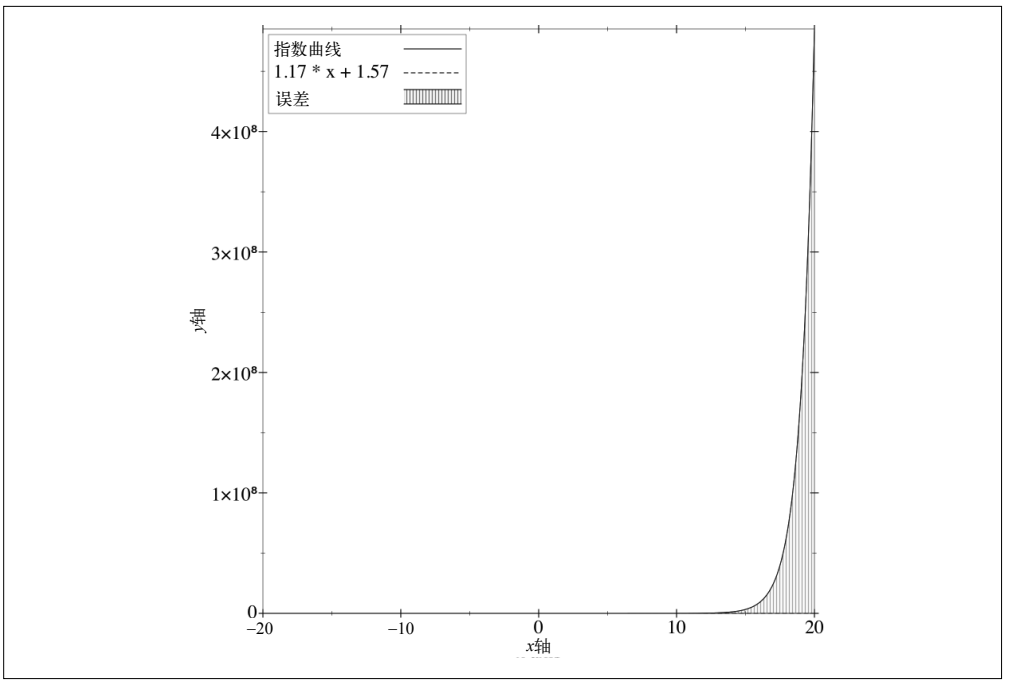

图 1-2：在 [−20,20] 区间内，直线将无法拟合指数曲线

统计学中有一个称为 power 的测度，它表示无法找到一个假负例（false negative）的概率。当 power 的值增大时，假负例的数量将减少。然而，真正影响该测度的是样本规模。如果样本规模过小，将无法获取足够的信息，从而无法得到一个良好的解。

1.3.3 过拟合

样本数太少是很不理想的一种情形，此时还存在对数据产生过拟合（overfitting）的风险。仍以相同的指数曲线为例，比如共有来自这条指数曲线的 30 000 个采样点。如果我们试图构建一个拥有 300 000 个算子的函数，便是对指数曲线过拟合，其实际上是记忆了全部 30 000 个数据点。这是有可能出现的，但如果有一个新数据点偏离了那些抽样，则这个过拟合模型对该点将产生较大的误差。

表面看来缓解模型欠拟合的最佳途径是为其提供更多的信息，但实际上这本身可能就是一个难以解决的问题。数据越多，通常意味着噪声越多，问题也越多。使用过多的数据和过于复杂的模型将使学习到的模型只能在该数据集上得到合理的结果，而对其他数据集将几乎完全不可用。

1.3.4 未来的不可预测性

机器学习非常适合不可预测的未来，因为大多数算法都需要从新息（即新的信息）中学习。但当新息到来时，其形式可能是不稳定的，而且会出现一些之前未预料到的新问题。我们并不清楚什么是未知的。在处理新息时，有时很难预测我们的模型是否仍能正常工作。

1.4 为降低风险应采用的测试

既然我们面临着若干问题，如不稳定的数据、欠拟合的模型、过拟合的模型以及未来数据的不确定性，到底应如何应对？好在有一些通用指导方针和技术（被称为启发式策略）可循，若将其写入测试程序，则可降低这些问题发生的风险。

1.4.1 利用接缝测试减少数据中的不稳定因素

在其著作 *Working Effectively with Legacy Code*（Prentice Hall 出版）中，Michale Feathers 在处理遗留代码时引入了*接缝测试*（testing seams）这个概念。接缝是指一个代码库的不同部分在集成时的连接点。在遗留代码中，很多时候都会遇到这样的代码：其内部机制不明，但当给定某些输入时，其行为可预测。机器学习算法虽不等同于遗留代码，但二者有相似之处。对待机器学习算法，也应像对待遗留代码那样，将其视为一个黑箱。

数据将流入机器学习算法，然后再从中流出。可通过对数据输入和输出进行单元测试来检验这两处“接缝”，以确保它们在给定误差容限内的有效性。

示例：对神经网络进行接缝测试

假设你准备对一个神经网络模型进行测试。你知道神经网络的输入数据取值需要介于 0 和 1 之间，且你希望所有数据的总和为 1。当数据之和为 1 时，意味着它相当于一个百分比。例如，如果你有两个小玩具和三个陀螺，则数据构成的数组将为 (2/5,3/5)。由于我们希望确保输入的信息为正，且和为 1，因此在测试套件中编写了下列测试代码：

```
it 'needs to be between 0 and 1' do
  @weights = NeuralNetwork.weights
  @weights.each do |point|
      (0..1).must_include(point)
  end
end

it 'has data that sums up to 1' do
  @weights = NeuralNetwork.weights
  @weights.reduce(&:+).must_equal 1
end
```

接缝测试是一种定义代码片段之间接口的好方法。虽然这个例子非常简单，但请注意，当

数据的复杂性增加时，这些接缝测试将变得更加重要。新加入的编程人员接触到这段代码时，可能不会意识到你所做的这些周密考虑。

1.4.2 通过交叉验证检验拟合效果

交叉验证是一种将数据划分为两部分（训练集和验证集）的方法，如图 1-3 所示。训练数据用于构建机器学习模型，而验证数据则用于验证模型能否取得期望的结果。这种策略提升了我们找到并确定模型中潜在错误的能力。

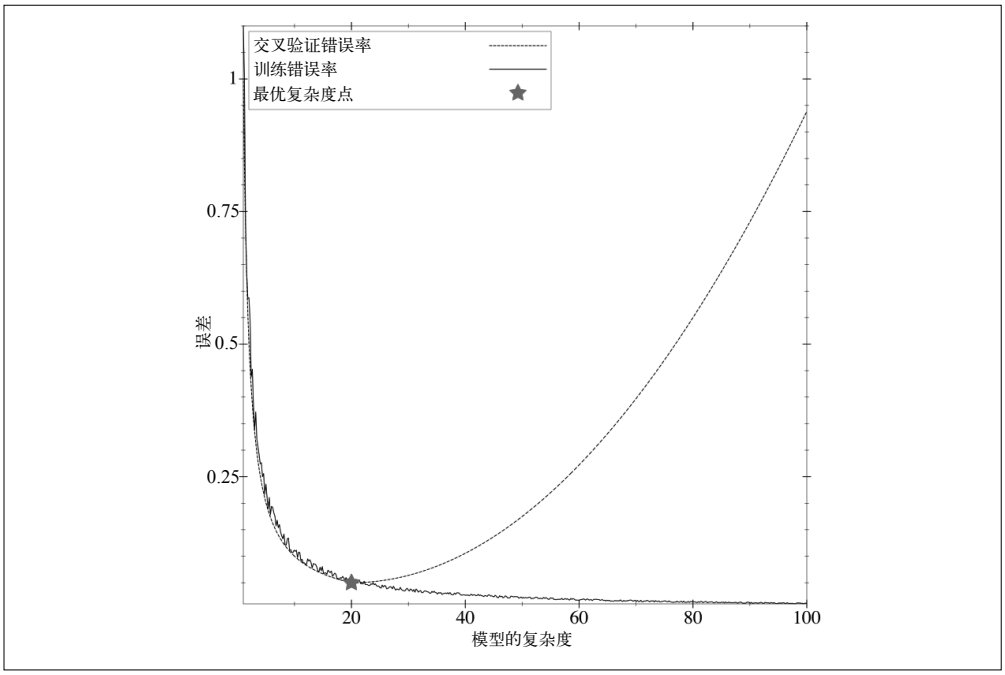

图 1-3：我们真正的目标是将交叉验证错误率或真实错误率最小化

训练专属于机器学习世界。由于机器学习算法的目标是将之前的观测映射为结果，因此训练非常重要。这些算法会依据人们所收集到的数据进行学习，因此如果缺少用于训练的初始数据集，该算法将百无一用。

交换训练集和验证集，有助于增加验证次数。你需要将数据集一分为二。第一次验证中，将集合 1 作为训练集，而将集合 2 作为验证集，然后将二者交换，再进行第二次验证。根据拥有的数据量，你可将数据划分为若干更小的集合，然后再按照前述方式进行交叉验证；如果你拥有的数据足够多，则可在任意数量的集合上进行交叉验证。

大多数情况下，人们会选择将验证数据和训练数据分为两部分，一部分用于训练模型，而

另一部分用于验证训练结果在真实数据上的表现。例如，假设你正在训练一个语言模型，利用隐马尔可夫模型（Hidden Markov Model，HMM）对语言的不同部分进行标注，则你会希望将该模型的误差最小化。

示例：对模型进行交叉验证

依据我们训练好的模型，训练错误率大致为 5%，但是当我们引入训练集之外的数据时，错误率可能会飙升到 15%。这恰恰说明了使用经划分的数据集的重要性；好比复式记账之于会计，对于机器学习而言这一点是极为必要的。例如：

```
def compare(network, text_file)
  misses = 0
  hits = 0

  sentences.each do |sentence|
    if model.run(sentence).classification == sentence.classification
      hits += 1
    else
      misses += 1
    end
  end

  assert misses < (0.05 * (misses + hits))
end

def test_first_half
  compare(first_data_set, second_data_set)
end

def test_second_half
  compare(second_data_set, first_data_set)
end
```

首先将数据划分为两个子集，这个方法消除了可能由机器学习模型中不恰当的参数引起的一些常见问题。这是在问题成为任何代码库的一部分之前，找到它们的绝佳途径。

1.4.3 通过测试训练速度降低过拟合风险

奥卡姆剃刀准则（Occam's Razor）强调对数据建模的简单性，并且认为越简单的解越好。这直接意味着“避免对数据产生过拟合”。越简单的解越好这种观点与过拟合模型通常只是记忆了它们的输入数据存在一些联系。如果能够找到更简单的解，它将发现数据中的一些模式，而非只是解析之前记忆的数据。

一种可间接度量机器学习模型复杂度的指标是它所需的训练时长。例如，假设你为解决某个问题，对两种不同的方法进行了测试，其中一种方法需要 3 个小时才能完成训练，而另一种方法只需 30 分钟。通常认为花费训练时间越少的那个模型可能越好。最佳方法可能是将基准测试包裹在代码周围，以考察它随着时间的推移变得更快还是更慢。

许多机器学习算法都需要设置最大迭代次数。对于神经网络，你可能会将最大 epoch 数设为 1000，表明你认为如果模型在训练中不经历 1000 次迭代，便无法获得良好的质量。epoch 这种测度所度量的是所有输入数据的完整遍历次数。

示例：基准测试

更进一步，你也可使用像 MiniTest 这样的单元测试框架。这类框架会向你的测试套件增加一定的计算复杂性和一个 IPS（iteration per second，每秒迭代次数）基准测试，以确保程序性能不会随时间而下降。例如：

```
it 'should not run too much slower than last time' do
  bm = Benchmark.measure do
    model.run('sentence')
  end
  bm.real.must_be < (time_to_run_last_time * 1.2)
end
```

这里，我们希望测试的运行时间不超过上次执行时间的 20%。

1.4.4 检测未来的精度和查全率漂移情况

精度（precision）和查全率（recall）是度量机器学习实现性能的两种方式。精度是对真正例的比例（即真正率）的度量[1]。例如，若精度为 4/7，则意味着所预测的 7 个正例中共有 4 个样本是真正例。查全率是指真正例的数目与真正例和假负例数目之和的比率。例如，若有 4 个真正例和 5 个假负例，则相应的查全率为 4/9。

为计算精度和查全率，用户需要为模型提供输入。这使得学习流程成为一个闭环，并且由于数据被误分类后所提供的反馈信息，随着时间的推移，系统在数据上的表现也会得到提升。例如，网飞（Netflix）公司[2]会依据你的电影观看历史来预测你对某部影片的星级评价。如果你对该系统预测的评分不满意，并按照自己的意志重新评分，或者表明你对该部影片不感兴趣，则网飞再将你的反馈信息输入模型，以服务于将来的预测。

1.5 小结

机器学习是一门科学，并且需要借助客观的方法来解决问题。像科学方法一样，TDD 也有助于问题的解决。TDD 和科学方法之所以相似，是因为二者具有下列三个共同点。

- 二者均认为解应当符合逻辑，且具有有效性。

注 1：精度（precision）并不等同于真正率（true positive rate）。真正率是指实际正例中被预测正确的样本比例，而精度则是在所预测的正例中实际正例所占的比例。此外，精度也不同于准确率（accuracy），后者是指被正确分类的样本在整个测试集中所占的比例。——译者注

注 2：网飞公司是一家在线影片租凭提供商，拥有极为优秀的影片自动推荐引擎。——译者注

- 二者均通过文档共享结果，且可持续不断地工作。
- 二者都有闭环反馈的工作机制。

虽然科学方法和 TDD 有许多相似之处，但机器学习仍有其特有的问题：

- 数据的不稳定性
- 欠拟合
- 过拟合
- 未来的不可预测性

好在，借助表 1-2 所示的启发式策略，可在一定程度上缓解这些挑战。

表1-2：降低机器学习风险的启发式策略

问题/风险	启发式策略
数据的不稳定性	接缝测试
欠拟合	交叉验证
过拟合	基准测试（奥卡姆剃刀准则）
未来的不可预测性	随着时间的推移追踪精度和查全率

美妙的是，在真正开始编写代码之前，你可以编写或思考所有这些启发式策略。像科学方法一样，测试驱动开发也是求解机器学习问题的一种极有价值的方法。