机器学习系统设计

作者：[美]Willi Richert

版权信息

书名：机器学习系统设计

作者：Willi Richert， Luis Pedro Coelho

译者：刘峰

ISBN：978-7-115-35682-6

目录

版权声明

译者序

在眼花缭乱的互联网产品背后，你会发现总有一些东西在沙子下面闪闪发光，它们本身并不是产品，却能把令人惊艳的产品带到你我面前。机器学习技术就是这样一种宝贝。

如果在十年前，你不知道机器学习，那么可以理解，因为它还是一个科研实验室的玩具；如果在十年后的今天，作为IT从业人员的你，还没有听说过机器学习，那么你真是“奥特曼”了。

对于产品来说，机器学习技术的应用，可以给产品带来质的飞跃，提高产品的核心竞争力；对于IT从业人员来说，机器学习技术已经成为了一种必备的技能，掌握了它，可以在各大IT公司游刃有余，个人价值徒增。

《机器学习系统设计》就是一本带你在机器学习海洋中遨游的书。如果你只想学习基础理论，那么这本书或许并不适合你。它并没有深入机器学习背后的数学细节，而是通过Python这样一种广泛应用的脚本语言，从数据处理，到特征工程，再到模型选择，把机器学习解决实际问题的过程一一呈现在你的面前。这本书的最大特点在于：易上手、实践性强、贴近应用。它可以让你在很短的时间内了解机器学习的基本原理，掌握机器学习工具，然后去解决实际问题。从文字、声音到图像，从主题模型、情感分析到推荐技术，本书所教给你的都是最实际的技术，让你从一个新手迅速成长为大咖。

鉴于译者水平有限，书中难免有错误疏漏之处，欢迎读者批评指正。微博：@飞旋的世界。电子邮箱：gnefuil@gmail.com。

作者致谢

感谢我的妻子Natalie和我儿子Linus及Moritz，没有家人的支持，本书将不会写就。感谢我的现任及前任经理Andras Bode、Clemens Marschner、Hongyan Zhou和Eric Crestan，感谢他们与我进行富有成效的讨论。感谢我的同事和朋友Tomasz Marchniak、Cristian Eigel、Oliver Niehoerster和Philipp Adelt，本书很多有趣的想法大都来自于他们。如果你发现本书中的错误，记得联系我，它们都归咎于我。

——Willi Richert

我要感谢我妻子Rita的爱心和支持，还要感谢我的女儿Anna，她是我生命中最美好的存在。

——Luis Pedro Coelho

关于作者

Willi Richert

机器学习和机器人学方面的博士，目前供职于微软Bing搜索核心研发团队。他从事多种机器学习领域的研究，包括主动学习和统计机器翻译。

> **Willi Richert的致谢**
> 感谢我的妻子Natalie和我儿子Linus及Moritz，没有家人的支持，本书将不会写就。感谢我的现任及前任经理Andras Bode、Clemens Marschner、Hongyan Zhou和Eric Crestan，感谢他们与我进行富有成效的讨论。感谢我的同事和朋友Tomasz Marchniak、Cristian Eigel、Oliver Niehoerster和Philipp Adelt，本书很多有趣的想法大都来自于他们。如果你发现本书中的错误，记得联系我，它们都归咎于我。

Luis Pedro Coelho

计算生物学家（用计算机辅助理解生物系统的学者）。在这个广阔的领域中，Luis从事生物图像信息学方面的研究，致力于生物标本图像分析中机器学习技术的应用。他主要关注大规模图像数据的处理。在机器人显微镜下，每天可以获得几十万幅图像，而我们不可能做到用肉眼检查所有这些图像。

Luis从机器学习领域世界领先的卡内基-梅隆大学获得了博士学位，并发表过多篇科学论文。

Luis从1998年开始开发开源软件，他把从里斯本理工大学计算机科学课程中所学的东西应用到了实际代码中。2004年，他开始用Python进行开发，并在几个Python开源库中贡献了代码。他是mahotas（由Python编写的流行计算机视觉库）的主要开发人员，其中一些机器学习代码就出于他手。

Luis Pedro Coelho的致谢

我要感谢我妻子Rita的爱心和支持，还要感谢我的女儿Anna，她是我生命中最美好的存在。

关于审校者

Matthieu Brucher在法国高等电力学院读取了工程学学位（专业是信息、信号、测量），并获得了法国斯特拉斯堡大学非监督流形学习方向的博士学位。他目前在一家石油公司担任高性能计算（HPC）软件开发员，正致力于下一代油藏模拟软件的开发。

Mike Driscoll从2006年春季开始从事Python编程，经常在博客 http://www.blog.pythonbrary.org/ 上发表关于Python的文章，偶尔也为Python软件基金会、i-Programmer和开发者论坛（Developer Zone）撰写文章。他喜爱摄影和阅读。Mike曾多次参与Packt图书的审校工作，这些图书包括：*Python 3 Object Oriented Programming*、*Python 2.6 Graphics Cookbook*和*Python Web Development Beginner's Guide*。

我要感谢我的妻子Evangeline，感谢她一直以来无尽的支持。我也要感谢我的朋友及家人，感谢他们对我的帮助。感谢耶稣对我的拯救。

Maurice HT Ling在墨尔本大学获得了分子与细胞生物学学士学位（优等），以及生物信息学博士学位。他目前在新加坡南洋理工大学担任研究员，同时还是墨尔本大学的荣誉研究员。他是*The Python Papers Anthology*的联合主编，也是新加坡Python用户组的联合创始人（自2010年起担任副主席一职）。他的研究兴趣在于生命——生物生命、人工生命以及人工智能——将计算机科学和统计学作为工具来理解生命以及它的诸多方面。个人网站：http://maurice.vodien.com。

前言

如果你手里（或者你的电子阅读器里）有这本书，可以说，这是一个幸运的巧合。毕竟，每年有几百万册图书印刷出来，供数百万读者阅读，而你恰好选择了这一本。可以说，正是机器学习算法引领你来阅读这本书（或者说是把这本书引领到你面前）。而我们作为本书的作者，很高兴看到你愿意了解更多的“怎么做”和“为什么”。

本书大部分内容都将涉及“怎么做”。例如，怎么处理数据才能让机器学习算法最大限度地利用它们？怎么选择正确的算法来解决手头的问题？

我们偶尔也会涉及“为什么”。例如，为什么正确评估很重要？为什么在特定情形下一个算法比另一个算法的效果更好？

我们知道，要成为该领域的专家还有很多知识要学。毕竟，本书只介绍了一些“怎么做”和极小一部分“为什么”。但在最后，我们希望这些内容可以帮你“启航”，然后快速前行。

本书内容

第1章通过一个非常简单的例子介绍机器学习的基本概念。尽管很简单，但也可能会有过拟合的风险，这对我们提出了挑战。

第2章讲解了使用真实数据解决分类问题的方法，在这里我们对计算机进行训练，使它能够区分不同类型的花朵。

第3章讲解了词袋方法的威力，我们可以在没有真正理解帖子内容的情况下，用它来寻找相似的帖子。

第4章让我们超越将每个帖子分配给单个簇的方式。由于真实的文本可以处理多个主题，我们可以看到如何把帖子分配到几个主题上。

第5章讲解了如何用逻辑回归判定用户的答案是好还是坏。在这个情景的背后，我们将学会用偏差-方差的折中调试机器学习模型。

第6章介绍了朴素贝叶斯的工作原理，以及如何用它对推文进行分类，来判断推文中的情感是正面的还是负面的。

第7章讨论了一个处理数据的经典课题，但它在今天仍然有意义。我们用它构建了一个推荐系统，这个系统根据用户所输入的喜欢和不喜欢的信息，为用户推荐新的商品。

第8章同时使用多种方法改进推荐效果。我们还可以看到如何只根据购物信息构建推荐系统，而不需要用户的评分数据（用户并不总会提供这一信息）。

第9章举例说明，如果有人把我们收集而成的庞大音乐库弄乱了，那么为歌曲建立次序的唯一希望就是让机器来对歌曲分类。你会发现，有时信任别人的专长比我们自己构建特征更好。

第10章讲解了如何在处理图像这个特定情景下应用分类方法。这个领域又叫做模式识别。

第11章告诉我们还有其他什么方法可以帮我们精简数据，使机器学习算法能够处理它们。

第12章讲解了不断膨胀的数据规模，以及这为何会为数据分析造成难题。在本章中，我们利用多核或计算集群，探索了一些更大规模数据的处理方法。另外，我们还介绍了云计算（将亚马逊的Web服务当做云计算提供商）。

附录A罗列了一系列机器学习的优质资源。

阅读需知

本书假定读者了解Python，并且知道如何利用`easy_install`或`pip`安装库文件。我们并不依赖于任何高等数学知识，如微积分或矩阵代数。

总体而言，本书将使用以下版本的软件，不过如果你使用任何新近版本，也没有问题。

- Python 2.7
- NumPy 1.6.2
- SciPy 0.11
- Scikit-learn 0.13

读者对象

本书适合想通过开源库来学习机器学习的Python程序员阅读参考。我们会通过示例概述机器学习的基本模式。

本书也适用于想用Python构建机器学习系统的初学者。Python是一个能够快速构建原型系统的灵活语言，它背后的算法都是由优化过的C或C++编写而成。因此，它的代码运行快捷，并且十分稳健，完全可以用在实际产品中。

排版约定

当你阅读本书时，会发现书中有各式各样的文本，它们用来区分不同类型的信息。下面是这些样式文本的示例以及相应说明。

正文中的代码是这样的：“我们可以通过使用`include`命令将其他内容包含进来。”

代码段采用如下形式：

```
def nn_movie(movie_likeness, reviews, uid, mid):
    likes = movie_likeness[mid].argsort()
  # 逆序排列，使最受喜爱的电影排在前面
    likes = likes[::-1]
  # 返回最相似电影的打分
    for ell in likes:
        if reviews[u,ell] > 0:
            return reviews[u,ell]
```

如果我们想让你注意代码段的特定部分，就会用粗体表示相应代码行或条目：

```
def nn_movie(movie_likeness, reviews, uid, mid):
    likes = movie_likeness[mid].argsort()
  # 逆序排列，使最受喜爱的电影排在前面
    likes = likes[::-1]
  # 返回最相似电影的打分
    for ell in likes:
        if reviews[u,ell] > 0:
            return reviews[u,ell]
```

新的术语以及**重要文字**采用楷体字。你在屏幕（如菜单或者对话框）中见到的文字这样出现在正文中：“点击**Next**按钮以进入下一界面。”

> **注意**　这里给出重要的注意事项。
>
> **提示**　提示和技巧则会在这里出现。

读者反馈

我们一贯欢迎读者的反馈意见。请告诉我们你对本书的看法，喜欢哪些部分，不喜欢哪些部分。这些反馈对于协助我们创作出真正对读者有所裨益的内容至关重要。

如果给我们反馈一般性信息，你可以发送电子邮件到feedback@packtpub.com，并在邮件标题中注明书名。如果你是某一方面的专家并愿意参与撰稿，请访问www.packtpub.com/authors参阅我们的作者指南。

客户支持

现在你已经拥有了某本由Packt出版的书，为了让你的付出得到最大的回报，我们还为你提供了其他许多方面的服务，请注意以下信息。

下载代码

如果你是通过http://www.packtpub.com的注册账户购买的图书，可以从该账户中下载相应Packt图书的示例代码[1]。如果你是从其他地方购买的本书，可以访问http://www.packtpub.com/support并进行注册，我们将会为你发送一封附有示例代码文件的电子邮件。

1.读者还可免费注册iTuring.cn，至本书页面下载——编者注

勘误

虽然我们会全力确保本书内容的准确性，但错误仍在所难免。如果你发现了本书中的错误（包括文字和代码错误），而且愿意向我们提交这些错误，我们感激不尽。这样一来，不仅可以减少其他读者的疑虑，也有助于改进本书后续版本。要提交你发现的错误，请访问http://www.packtpub.com/submit-errata，选择相应图书，点击**errata submission form**（提交勘误表[2]），登记你的勘误详情。勘误通过验证之后将上传到Packt网站，或添加到已有的勘误列表中。任何图书当前的勘误都可以通过http://www.packtpub.com/support来查看。

2.中文版的勘误请注册iTuring.cn，至本书页面提交。——编者注

举报盗版

对所有媒体来说，互联网盗版都是一个棘手的问题。Packt很重视版权保护。如果你在互联网上发现我们公司出版物的任何非法复制品，请及时告知我们相关网址或网站名称，以便我们采取补救措施。

如果发现可疑盗版材料，请通过copyright@packtpub.com联系我们。

对你帮助我们保护作者权益、确保我们持续提供高品质图书的行为表示敬意。

疑难解答

如果你就本书存有疑问，请发送电子邮件到questions@packtpub.com，我们会尽力解决。

第1章　Python机器学习入门

机器学习（ML）就是教机器自己来完成任务，就这么简单。复杂性源于细节，而这很可能就是你要读这本书的原因。

也许你现在拥有过多的数据，却对这些数据缺少理解，你希望机器学习算法可以帮助解决这个难题。于是你随机找了一些算法开始钻研，但过了一段时间就感到困惑了：在无数的算法中应该选择哪一个呢？

或许你笼统地对机器学习感兴趣，也阅读过相关的博客和文章。机器学习中的任何东西看起来都那么不可思议、那么酷，所以你开始进行探索，把一些简单的数据放入一个决策树或者一个支持向量机。但是，成功将它应用到一些其他数据之后，你又心生疑惑：所有的设置都正确吗？你得到最优的结果了吗？怎么知道有没有更好的算法？或者，你的数据是否就是“正确的”？

欢迎加入机器学习的行列！我们作为本书的作者，也曾处在这个阶段，寻找过机器学习理论教材背后的真实故事。我们发现，很多东西都是标准教材中通常不会讲到的“魔术”。所以，从某种意义上说，我们在把这本书写给年轻的自己。它不仅是机器学习的快速入门书，而且还会把我们积累的经验教训传授给你。我们希望它还可以让你更顺畅地走进计算机科学中最令人兴奋的一个领域。

1.1　梦之队：机器学习与Python

机器学习的目标就是通过若干示例（怎样做或不做一个任务）让机器（软件）学会完成任务。假设每天早上当你打开电脑，都会做同样的事情：移动电子邮件，把属于某一

特定主题的邮件放入同一个文件夹。过了一段时间，你感到厌烦了，开始琢磨是否可以让这种琐事自动完成。一种方法是分析你的大脑，将整理电子邮件时大脑思考过程中的规则记录下来。然而，这种方式相当麻烦，而且总不完美。你会漏掉一些规则，同时又会对另一些规则细致过头。另一种更好的、更加面向未来的方法是将这个过程自动化，即选择一组电子邮件元数据信息和邮件正文/文件夹名对，让算法据此选出最好的规则集。这些数据对就是你的训练数据，而生成的规则集（也叫做模型）以后能够应用到新的电子邮件上。这就是最简单的机器学习。

当然，机器学习（也常称作数据挖掘或预测分析）本身并不是一个全新的领域。正相反，它这些年来的成功可以归因于务实地采用了已经验证了的坚实技术，以及借鉴其他成功领域的真知灼见，例如统计学。统计学的目的是通过学习更多的潜在模式和关联关系，来帮助人类深入理解数据。对机器学习的成功应用了解得越多（你已经查看过kaggle.com了吧？），越会发现应用统计学是机器学习专家经常研究的一个领域。

本书后面将会介绍，构想出一个合适的机器学习（ML）方法，从来都不是一个瀑布式的过程。相反，你需要反复分析，在各色各样的机器学习算法中尝试不同版本的输入数据。这种探索方式非常适合Python。作为一门解释性高级编程语言，Python似乎就是专为尝试不同事物而设计的。更重要的是，用它进行这些尝试非常迅捷。无疑，它比C语言或其他类似的静态类型编程语言要慢一点。然而，它有着大量易用的库，而这些库往往是用C语言编写的，因此你不必为了敏捷性而牺牲速度。

1.2 这本书将教给你什么（以及不会教什么）

本书将全面展示不同应用领域正在使用的各种机器学习算法，以及使用它们时应当注意什么。然而，根据亲身经验，我们知道做这些很“酷”的事——使用和调整机器学习算法，比如支持向量机（**SVM**）、最邻近搜索（**NNS**），或者同时支持两者——其实只需要耗费一位优秀机器学习专家的一点儿时间。看看下面这个典型的工作流程，你就会发现绝大部分时间将花费在一些相当平凡的任务上：

1. 读取和清洗数据；
2. 探索和理解输入数据；
3. 分析如何最好地将数据呈现给学习算法；
4. 选择正确的模型和学习算法；
5. 正确地评估性能。

在探索和理解输入数据的时候，我们需要一点统计学和基础数学知识。但当这样做的时候，你会发现，这些数学课上似乎十分枯燥的知识，用来处理有趣的数据时，其实真的很令人兴奋。

解读数据标志着旅程的开始。你面对诸如无效值或缺失值的问题时，会发现这更像是一种技艺而非一门精确的科学。这是一种非常有益的技艺，因为如果这部分做得正确，那么你的数据就能够适应更多的机器学习算法，从而成功的可能性大大提高。

数据在程序的数据结构中就绪之后，你要清楚自己正在跟何方神圣打交道。你有足够的数据来回答自己的问题吗？如果没有，也许应当考虑通过额外的途径来获取一些。或许你的数据过多？那么你可能要考虑怎样最有效地从中抽取样本。

你通常不会直接将数据输入机器学习算法，而是在训练前对部分数据进行提炼。很多时候，使用机器学习算法会让你得到性能提升的回报。一个简单算法在提炼后数据上的表现，甚至能够超过一个非常复杂的算法在原始数据上的效果。这部分机器学习流程叫做**特征工程**（feature engineering），通常是一个非常令人兴奋的有意思的挑战。你有创意和智慧，便会立即看到结果。

选择正确的学习算法并不只是尝试一下工具箱中的三四个算法那么简单（工具箱中会有很多的算法）。它更需要的是深思熟虑，来权衡性能和功能的不同需求。你是否会为了快速得到结果而牺牲质量，还是愿意投入更多的时间来得到最好的结果？你是否对未来的数据有一个清晰的认识，还是应该在这方面更保守一点？

最后，性能评估是怀有远大抱负的机器学习初学者最常犯错误的地方。有一些简单的错误，比如使用了与训练相同的数据来测试你的方法。但还有一些比较难的，例如，你使用了不平衡的训练数据。再说一次，数据决定了你的任务是成功还是失败。

我们看到，只有第4点是关于那些花哨的算法的。虽然如此，希望这本书可以使你相信，另外4个任务并不是简单的杂务，它们同等重要，或许还更加令人兴奋。我们希望读过本书之后，你可以真正爱上数据，而非学到的算法。

最后，我们并不想让机器学习算法的理论把你压垮，因为这方面已经有很多优秀的著作了（可以在附录A中找到我们的推荐）。相反，我们会在各节中直观地介绍各种基础方法——这对于你大致理解其中的思想已经足够了，并且能够确保你走好第一步。因此，这本书并不是机器学习“权威指南”，而更像是初学者的工具。我们希望它能够激发你的好奇心，并足以让你保持渴望，不断探索这个有趣的领域。

在本章的余下部分，我们将着手介绍Python的基础库NumPy和SciPy，并且使用Scikit-learn进行第一个机器学习训练。同时我们将介绍基本的ML概念，它们稍后将贯穿于全书。本书余下的各章会详细讲述之前介绍的5个步骤，同时突出介绍使用Python的机器学习方法在各种应用场景中的不同方面。

1.3 遇到困难的时候怎么办

本书中，我们会试图讲清楚每一个必要的想法，保证你能重现各个步骤。虽然如此，你仍然可能会遇到困难。其原因可能是软件包版本的古怪组合，可能是简单的拼写错误，也可能是理解上的问题。

在这种情况下，可以通过很多不同的途径来获取帮助。很有可能，你想问的问题早已有人提出，而且下面这些优质的问答网站已经给出了答案。

- http://metaoptimize.com/qa
  这个问答网站专注于机器学习主题。几乎所有的问题都会得到机器学习专家的高水平解答。即使你并没有问题，不时地翻阅这些问答也是一个很好的习惯。
- http://stats.stackexchange.com
  这个问答网站又叫交叉验证（Cross Validated），和MetaOptimized相似，但它更专注于统计方面的问题。
- http://stackoverflow.com
  这个问答网站与前面的相似，但还会更宽泛地讨论一些常规的编程主题。例如，一些软件包的问题，这些我们也会在本书中提到（SciPy和Matplotlib）。
- Freenode的#machinelearning频道
  这个互联网中转聊天（IRC）频道专门讨论机器学习主题。这是个机器学习方面的专业社区，虽然很小，但是非常活跃，十分有用。
- http://www.TwoToReal.com
  这是由本书作者制作的一个即时问答网站，来为你解答不适于上述任何网站的问题。如果你提交了一个问题，我们将会收到一条即时消息；只要我们当中有人在线，就会与你交谈解决。

正如一开始所述，本书试图帮助你快速开始机器学习之旅。因此，我们鼓励你构建自己的机器学习相关博客的列表，并且定期查阅。这是去了解什么可行、什么不可行的最佳方式。

在这里，唯一要着重指出的博客是http://blog.kaggle.com。这是举办过很多次机器学习比赛的Kaggle公司维护的博客（在附录A里可以找到更多链接）。通常，他们鼓励比赛优胜选手写文章，详细介绍他们是怎样着手解决难题的，什么样的策略不可行，以及他们是怎样想出获胜的策略的。如果你不想读其他的东西，没问题，但这个必须读。

1.4 开始

如果你已经安装了Python（2.7或更高版本），那么还需要安装NumPy和SciPy来处理数据，并需要安装Matplotlib对数据进行可视化。

1.4.1 NumPy、SciPy和Matplotlib简介

在讨论具体的机器学习算法之前，必须说一下如何最好地存储需要处理的数据。这很重要，因为多数高级学习算法，如果运行永远不会结束，对我们毫无用处。这可能仅仅是因为数据访问太慢了，也可能是因为这些数据的表示方式迫使操作系统一直做数据交换。再加上Python是一种解释性语言（尽管是高度优化过的），和C或者Fortran相比，这类语言对很多重数值算法来说运行缓慢。所以或许应该问一问究竟为什么有这么多科学家和公司，甚至在高度计算密集型领域内豪赌Python。

答案就是，在Python中很容易把数值计算任务交给下层的C或Fortran扩展包。这也正是NumPy和SciPy要做的事情（http://scipy.org/install.html）。在NumPy和SciPy这个组合中，NumPy提供了对高度优化的多维数组的支持，而这正是大多数新式算法的基本数据结构。SciPy则通过这些数组提供了一套快速的数值分析方法库。最后，用Python来绘制高品质图形，Matplotlib（http://matplotlib.org/）也许是使用最方便、功能最丰富的程序库了。

1.4.2 安装Python

幸运的是，所有主流操作系统，如Windows、Mac和Linux，都有针对NumPy、SciPy和Matplotlib的安装程序。如果你对安装过程不是很清楚，那么可能就需要安装Enthought Python发行版（https://www.enthought.com/products/epd_free.php）或者Python(x,y)（http://code.google.com/p/pythonxy/wiki/Downloads），而这些已经包含在之前提到过的程序包里了。

1.4.3 使用NumPy和SciPy智能高效地处理数据

让我们快速浏览一下NumPy的基础示例，然后看看SciPy在NumPy之上提供了哪些东西。在这个过程中，我们将开始使用Matplotlib这个非凡的工具包进行绘图。

你可以在http://www.scipy.org/Tentative_NumPy_Tutorial上找到NumPy所提供的更多有趣示例。

你也会发现由Ivan Idris所著的《Python数据分析基础教程：NumPy学习指南（第2版）》非常有价值。你还可以在http://scipy-lectures.github.com上找到辅导性质的指南，并到http://docs.scipy.org/doc/scipy/reference/tutorial访问SciPy的官方教程。

在本书中，我们使用1.6.2版本的NumPy和0.11.0版本的SciPy。

1.4.4 学习NumPy

让我们引入NumPy，并小试一下。对此，需要打开Python交互界面。

```
>>> import numpy
>>> numpy.version.full_version
1.6.2
```

由于我们并不想破坏命名空间，所以肯定不能做下面这样的事情：

```
>>> from numpy import *
```

这个`numpy.array`数组很可能会遮挡住标准Python中包含的数组模块。相反，我们将会采用下面这种便捷方式：

```
>>> import numpy as np
>>> a = np.array([0,1,2,3,4,5])
>>> a
array([0, 1, 2, 3, 4, 5])
>>> a.ndim
1
>>> a.shape
(6,)
```

这里只是采用了与在Python中创建列表相类似的方法来创建数组。不过，NumPy数组还包含更多关于数组形状的信息。在这个例子中，它是一个含有5个元素的一维数组。到目前为止，并没有什么令人惊奇的。

现在我们将这个数组转换到一个2D矩阵中：

```
>>> b = a.reshape((3,2))
>>> b
array([[0, 1],
       [2, 3],
       [4, 5]])
>>> b.ndim
2
>>> b.shape
(3, 2)
```

当我们意识到NumPy包优化到什么程度时，有趣的事情发生了。比如，它在所有可能之处都避免复制操作。

```
>>> b[1][0]=77
>>> b
array([[ 0, 1],
       [77, 3],
       [ 4, 5]])
>>> a
array([ 0, 1, 77, 3, 4, 5])
```

在这个例子中，我们把`b`的值从2改成77，然后立刻就会发现相同的改动已经反映在`a`中。当你需要一个真正的副本时，请记住这个。

```
>>> c = a.reshape((3,2)).copy()
>>> c
array([[ 0, 1],
       [77, 3],
       [ 4, 5]])
>>> c[0][0] = -99
>>> a
array([ 0, 1, 77, 3, 4, 5])
>>> c
array([[-99, 1],
[ 77, 3],
[ 4, 5]])
```

这里，`c`和`a`是完全独立的副本。

NumPy数组还有一大优势，即对数组的操作可以传递到每个元素上。

```
>>> a*2
array([ 2, 4, 6, 8, 10])
>>> a**2
array([ 1, 4, 9, 16, 25])
Contrast that to ordinary Python lists:
>>> [1,2,3,4,5]*2
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
>>> [1,2,3,4,5]**2
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for ** or pow(): 'list' and
```

```
'int'
```

当然，我们在使用NumPy数组的时候会牺牲Python列表所提供的一些敏捷性。像相加、删除这样的简单操作在NumPy数组中会有一点麻烦。幸运的是，这两种方式都可以使用。我们可以根据手头上的任务来选择最适合的那种。

1. 索引

NumPy的部分威力来自于它的通用数组访问方式。

除了正常的列表索引方式，它还允许我们将数组本身当做索引使用。

```
>>> a[np.array([2,3,4])]
array([77, 3, 4])
```

除了判断条件可以传递到每个元素这个事实，我们得到了一个非常方便的数据访问方法。

```
>>> a>4
array([False, False, True, False, False, True], dtype=bool)
>>> a[a>4]
array([77, 5])
```

这还可用于修剪异常值。

```
>>> a[a>4] = 4
>>> a
array([0, 1, 4, 3, 4, 4])
```

鉴于这是一个经常碰到的情况，所以这里有一个专门的修剪函数来处理它。如下面的函数调用所示，它将数组值超出某个区间边界的部分修剪掉。

```
>>> a.clip(0,4)
array([0, 1, 4, 3, 4, 4])
```

2. 处理不存在的值

当我们预处理刚从文本文件中读出的数据时，NumPy的索引能力就派上用场了。这些数据中很可能包含不合法的值，我们像下面这样用`numpy.NAN`做标记，来表示它不是真实数值。

```
c = np.array([1, 2, np.NAN, 3, 4]) # 假设已经从文本文件中读取了数据
>>> c
array([ 1., 2., nan, 3., 4.])
>>> np.isnan(c)
array([False, False, True, False, False], dtype=bool)
>>> c[~np.isnan(c)]
array([ 1., 2., 3., 4.])
>>> np.mean(c[~np.isnan(c)])
2.5
```

3. 运行时行为比较

让我们比较一下NumPy和标准Python列表的运行时行为。在下面这些代码中，我们将会计算从1到1000的所有数的平方和，并观察这些计算花费了多少时间。为了使评估足够准确，我们重复做了10 000次，并记录下总时间。

```
import timeit
normal_py_sec = timeit.timeit('sum(x*x for x in xrange(1000))',
                              number=10000)
naive_np_sec = timeit.timeit('sum(na*na)',
                             setup="import numpy as np; na=np.arange(1000)",
                             number=10000)
good_np_sec = timeit.timeit('na.dot(na)',
                            setup="import numpy as np; na=np.arange(1000)",
                            number=10000)

print("Normal Python: %f sec"%normal_py_sec)
print("Naive NumPy: %f sec"%naive_np_sec)
print("Good NumPy: %f sec"%good_np_sec)

Normal Python: 1.157467 sec
Naive NumPy: 4.061293 sec
Good NumPy: 0.033419 sec
```

我们观察到两个有趣的现象。首先，仅用NumPy作为数据存储（原始NumPy）时，花费的时间竟然是标准Python列表的3.5倍。这让我们感到非常惊奇，因为我们原本以为既然它是C扩展，那肯定要快得多。对此，一个解释是，在Python中访问个体数组元素是相当耗时的。只有当我们在优化后的扩展代码中使用一些算法之后，才能获得速度上的提升。一个巨大的提升是：当使用NumPy的`dot()`函数之后，可以得到25倍的加速。总而言之，在要实现的算法中，应该时常考虑如何将数组元素的循环处理从Python中移到一些高度优化的NumPy或SciPy扩展函数中。

然而，速度也是有代价的。当使用NumPy数组时，我们不再拥有像Python列表那样基本上可以装下任何数据的不可思议的灵活性。NumPy数组中只有一个数据类型。

```
>>> a = np.array([1,2,3])
>>> a.dtype
dtype('int64')
```

如果尝试使用不同类型的元素，NumPy会尽量把它们强制转换为最合理的常用数据类型：

```
>>> np.array([1, "stringy"])
array(['1', 'stringy'], dtype='|S8')
>>> np.array([1, "stringy", set([1,2,3])])
array([1, stringy, set([1, 2, 3])], dtype=object)
```

1.4.5 学习SciPy

在NumPy的高效数据结构之上，SciPy提供了基于这些数组的算法级应用。本书中任何一个数值分析方面的重数值算法，你都可以在SciPy中找到相应的支持。无论是矩阵运算、线性代数、最优化方法、聚类、空间运算，还是快速傅里叶变换，都囊括在这个工具包中了。因此在实现数值算法之前先查看一下SciPy模块，是一个好习惯。

为了方便起见，NumPy的全部命名空间都可以通过SciPy访问。因此从现在开始，我们会在SciPy的命名空间中使用NumPy的函数。通过比较这两个基础函数的引用，很容易就可以进行验证，例如：

```
>>> import scipy, numpy
>>> scipy.version.full_version
0.11.0
>>> scipy.dot is numpy.dot
True
```

各种各样的算法被分组到下面这个工具包中：

SciPy工具包	功能