HZ BOOKS
华章IT

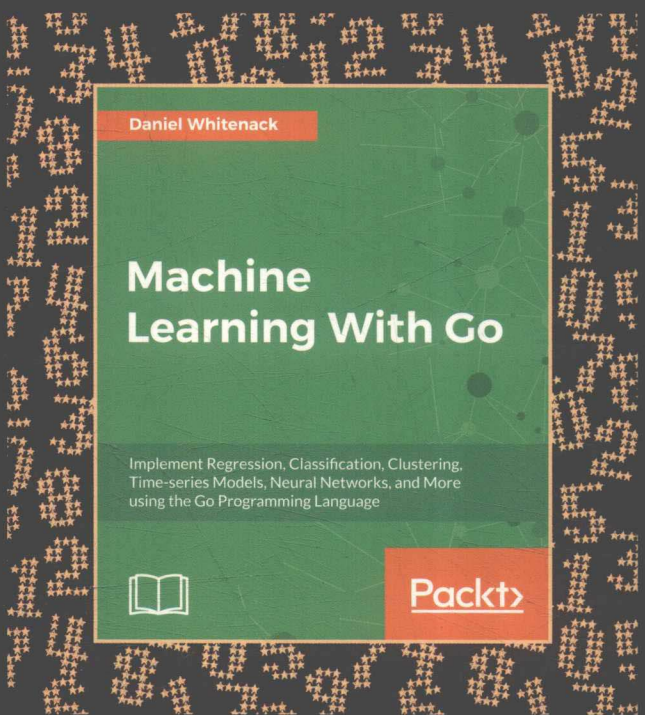
[PACKT]
PUBLISHING

数据科学与工程技术丛书

机器学习

Go语言实现

[美] 丹尼尔·怀特纳克（Daniel Whitenack） 著
谢文江 姜明魁 译



MACHINE LEARNING WITH GO

机械工业出版社
China Machine Press

数据科学与工程技术丛书

MACHINE LEARNING WITH GO

机器学习

Go语言实现

[美] 丹尼尔·怀特纳克（Daniel Whitenack） 著
谢文江 姜明魁 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

机器学习：Go 语言实现 /（美）丹尼尔·怀特纳克（Daniel Whitenack）著；谢文江，姜明魁译 . —北京：机械工业出版社，2018.9
（数据科学与工程技术丛书）
书名原文：Machine Learning With Go

ISBN 978-7-111-60979-7

I. 机… II. ①丹… ②谢… ③姜… III. 程序语言 - 程序设计 IV. TP312

中国版本图书馆 CIP 数据核字（2018）第 218894 号

本书版权登记号：图字 01-2017-7516

本书不仅深入浅出地介绍了利用 Go 语言构建预测模型的技术，还会帮助读者理解如何在真实场景中应用机器学习工作流程。第 1 章～第 3 章讲述了在机器学习流程中如何准备和分析数据；第 4 章～第 7 章详细介绍了回归、分类、集群、时间序列和异常检测等技术；第 8 章和第 9 章对神经网络、深度学习、部署、分布分析和模型进行了深入探究；附录介绍了与机器学习相关的算法 / 技术。本书适合作为对 Go 感兴趣的数据科学家、分析师、工程师和相关专业学生的参考书。

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：冯秀泳　　责任校对：殷　虹
印　　刷：北京市兆成印刷有限责任公司　　版　　次：2018 年 10 月第 1 版第 1 次印刷
开　　本：185mm×260mm　1/16　　印　　张：14.25
书　　号：ISBN 978-7-111-60979-7　　定　　价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：(010) 88378991　88361066　　投稿热线：(010) 88379604
购书热线：(010) 68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

译 者 序

机器学习是人工智能的核心，是以机器学习为手段解决人工智能中的问题。它在最近30多年已经发展为一门多领域的交叉学科，涉及概率论、统计学、计算复杂性理论等多门学科。其内容主要是设计和分析一些可以让计算机自动学习的算法，用以从繁杂的数据中自动进行分析来获得规律，并利用该规律对未知数据进行预测。

当前市面上关于机器学习的书籍举不胜举，其中大部分都是使用Python或者R。本书独辟蹊径地使用Go语言来进行机器学习，不仅克服了其他语言中容易忽视的一些问题（比如说维持高级别的数据完整性），还会利用Go的独特功能，帮助读者构建优良、高效且易维护的机器学习代码。

本书不仅深入浅出地介绍了机器学习的理论，还使用了大量的Go语言实例。从基础的数据收集和整理，到常见机器学习技术介绍，再到最后的神经网络和深度学习，所有这些内容均有详细的Go语言实现。

不仅如此，本书所使用的软件包、部署工具以及工作流程在业内已经广泛使用，读者可以轻松地将其整合到自己的机器学习和数据分析流程中。

关于作者

Daniel Whitenack（@dwhitena）博士是一位工作在 Pachyderm（@pachydermIO）的资深数据科学家，开发出了革命性的、分布式的数据流水线，其中包括预测模型、数据可视化、统计分析等，在世界各地的会议（GopherCon、JuliaCon、PyCon、ODSC、Spark Summit 等）上做过演讲。他目前在普渡大学（@LifeAtPurdue）教授数据科学 / 工程，同时与 Ardan 实验室（@ardanlabs）共同维护着 Jupyter 项目的 Go 内核，并积极为各种开源数据科学项目做出贡献。

在这里我想感谢我的妻子，感谢她在我撰写此书时给予的无限耐心和支持。

同时我还想感谢许多热心人和数据科学家对我的指导、合作以及鼓励。他们是 Bill Kennedy、Brendan Tracey、Sebastien Binet、Alex Sanchez、Pachyderm 的整个团队（包括 Joey Zwicker 和 Joe Doliner）、Chris Tava、Mat Ryer、David Hernandez、Xuanyi Chew、Minio 的团队（包括 Anand Babu Periasamy 和 Garima Kapoor），以及其他许多帮助过我的人。

Soli Deo Gloria

关于审校者

Niclas Jern，自从 Go 1.0 起就一直使用 Go 语言解决大量有趣的问题。他在埃博学术大学获得计算机工程的硕士学位，专业是软件工程。

他喜爱用 Go 以及其他编程语言去解决问题，尤其是数据处理和机器学习领域的问题。他的爱好是散步、在健身房举铁健身，偶尔在 http://www.njern.co 发言，以及同家人享受天伦之乐。

Niclas 目前就职于 Walkbase，这是他与埃博学术大学的同学联合创办的一家公司。在公司中他主要领导工程团队，解决来自革命性的零售分析中的数据处理问题。

Richard Townsend，还在华威大学攻读学士学位的时候，就成了 2014 年 GoLearn 的最高贡献者（这也是 GoLearn 有许多奇怪行为的原因）。自那时起，他便在英国顶级科技公司工作，工作范围包括从嵌入式系统到安卓操作系统框架的所有内容，目前正致力于网络浏览器的优化工作。他花费了大量的时间从事情感分析工作（在情感分析方面合著了两篇论文），以及其他自然语言处理任务，例如使用最新的深度学习技术进行词性标注。

前　言

在位于科技前沿的公司以及越来越多的大型企业中，机器学习和人工智能似乎都风靡一时。数据科学家正在尝试使用机器学习来完成任何事情，从驾驶汽车到画猫。但是，如果关注数据科学界的动态，很可能会看到类似于 Python 和 R 用户之间的*语言战争*。这些语言在机器学习交流中占据主导地位，而且往往被当成将机器学习集成到组织中的唯一选择。本书中将探索第 3 个选项：由 Google 创造的开源编程语言，Go。

Go 的独特功能以及 Go 程序员的思维方式，可以帮助数据科学家克服他们遇到的一些常见困难。尤其是数据科学家（很不幸地）被公认为会产生糟糕、低效和不易维护的代码。本书将解决这个问题，并将清楚地阐明，如何在机器学习方面富有成效，同时还能生产出保持高级别完整性的应用程序。书中还会帮助读者应对在现有工程学组织中有关完整性分析和机器学习代码的常见挑战。

本书将使读者成为富有成效的、有创新精神的数据分析师，帮助读者利用 Go 来构建强大而有价值的应用程序。为此，本书会清楚地介绍在 Go 世界中机器学习的技术和编程方面的内容，同时也将指导读者理解现实分析工作中合理的工作流程和理念。

本书内容

在机器学习流程中准备和分析数据：

- 第 1 章，*数据的收集和组织*，包含了收集、组织和解析数据，这些数据来自本地和远端，或者需要保存到本地和远端。阅读完本章，读者会理解如何与存储在多个位置、不同格式的数据做交互，如何解析和清洗数据，如何输出清洗过、解析过的数据。
- 第 2 章，*矩阵、概率论和统计学*，涵盖了将数据组织成矩阵和矩阵运算。阅读完本章，读者将了解如何在 Go 程序中生成矩阵，以及如何利用这些矩阵执行各种类型的矩阵运算。本章会介绍统计度量和日常数据分析工作的操作关键。

读者读完本章之后，还将了解如何执行可靠的汇总数据分析、描述和可视化分布、量化假设以及转换数据集（例如降维）。

- 第 3 章，*评估和验证*，本章主要介绍评估和验证，评估和验证是衡量机器应用性能和确保其通用化的关键。一旦读者完成本章，将会了解评估模型性能的各种度量指标，以及各种用以更通用地验证模型的技术。

机器学习的技术：

- 第 4 章，*回归*，本章解释了回归是一种广泛使用的连续变量建模技术，同时也是其他模型的基础。回归生成能立刻解释的模型。因此当开始在机构中引入预测功能时，回归可以提供一个很好的起点。
- 第 5 章，*分类*，分类作为一种与回归不同的机器学习技术，其目标变量通常是分类的或标记的。例如，分类模型可以将电子邮件分类为垃圾邮件或者非垃圾邮件，或者将网络数据分类为欺诈性或非欺诈性的数据等。
- 第 6 章，*集群*，集群是一种用于形成样本分组的无监督机器学习技术。在本章的最后，读者将能够自动形成数据点的分组，用以更好地理解其结构。
- 第 7 章，*时间序列和异常检测*，本章介绍了用于对时间序列数据进行建模的技术，如股票价格、用户事件等。阅读本章后，读者将会理解如何评估时间序列中的各种术语，建立时间序列模型，并检测时间序列中的异常情况。

机器学习探究：

- 第 8 章，*神经网络和深度学习*，介绍了利用神经网络进行回归、分类和图像处理的技术。阅读本章之后，读者将了解如何以及何时应用这些更复杂的建模技术。
- 第 9 章，*部署、分布分析和模型*，使读者能够将整个课程中开发的模型应用部署到生产环境，并对生产规模数据进行分布处理。本章说明了如何利用本书使用的现有代码轻松完成这两件事情，而不用对代码进行重大修改。
- 附录，*与机器学习相关的算法 / 技术*，会在本书全书中被引用，并且将提供与机器学习工作流程相关的算法、优化和技术的信息。

阅读本书需要的准备

为了运行本书中的示例并尝试使用本书中涵盖的技术，通常需要以下内容：

- 可以访问一个类似 bash 的 shell。
- 一个完整的 Go 环境，包括 Go、一个编辑器以及相关的默认或定制的环境变量。例如，可以按照在 https://www.goinggo.net/2016/05/installing-go-and-your-

workspace.html 上的指南进行操作。

- 各种 Go 的依赖包。这些可以通过命令 `go get` 得到。

然后，为了运行与一些高级的机器学习主题（如数据流水线和深度学习）相关的示例，需要准备一些额外的内容：

- 需要安装或部署 Pachyderm。可以按照链接中的文档来获取 Pachyderm，并在本地或在云上运行：http://pachyderm.readthedocs.io/en/latest/。
- 一个可运行的 Docker（https://www.docker.com/community-edition#/download）。
- 安装一个 TensorFlow。要在本地安装 TensorFlow，可以在 https://www.tensorflow.org/install/ 上阅读安装指南。

谁适合读本书

本书对于以下类型的读者会有所帮助：

1. 对机器学习和数据分析感兴趣的 Go 程序员
2. 对 Go 感兴趣的数据科学家 / 分析师 / 工程师，并希望将其整合到他们的机器学习和数据分析工作流程中

约定

许多 Go 代码片段不包括 `package main` 和 `func main(){}` 主函数。除非特别说明，Go 代码片段都是在一个必要的基础架构中编译。在本书中将假定代码示例被编译在一个叫做 `myprogram` 的目录中，因此会被编译为一个名为 `myprogram` 的二进制文件。然而，读者将认识到，代码可以被复制到 `GOPATH/src` 目录下的任何文件夹中，或者被编译成根据读者的喜好命名的二进制文件。

在本书中，为了区分不同类型的信息，采用了不同的文本样式。下面是这些样式的一些例子以及它们的含义说明。Go 语言中的文本、数据库表名、文件夹名、文件名、文件扩展名、路径名、虚拟 URL、用户输入和 Twitter 句柄中的代码如下段所示：“接下来的代码行读取链接并将其分配给 `BeautifulSoup` 功能。”代码块如下所示：

```
#import packages into the project
from bs4 import BeautifulSoup
from urllib.request import urlopen
import pandas as pd
```

当需要提醒读者注意代码块的特定部分时，相关行或项目以粗体显示：

```
 [default] exten => s,1,Dial(Zap/1|30)
 exten => s,2,Voicemail(u100)
 exten => s,102,Voicemail(b100)
exten => i,1,Voicemail(s0)
```

命令行输入或输出如下所示：

```
C:\Python34\Scripts> pip install -upgrade pip
C:\Python34\Scripts> pip install pandas
```

新术语和**重要词汇**以粗体显示。在屏幕上看到的单词，例如菜单或对话框，在本书中会如下所示：“为了下载新模块，我们将转到：**文件**|**设置**|**项目名称**|**项目解释器**”。

警告或重要的注意事项使用这样的图标。

提示和技巧使用这样的图标。

下载示例代码及彩色图像

本书的示例源码及所有截图和样图，可以从 http://www.packtpub.com 通过个人账号下载，也可以访问华章图书官网 http://www.hzbook.com，通过注册并登录个人账号下载。

目　录

第 1 章
数据的收集和组织

调查结果显示，数据科学家至少有 90% 的时间是用于收集、组织和清洗数据，而不是用于训练或调试复杂的机器学习模型。为什么会这样呢？难道机器学习不才是最有趣的部分吗？为何会如此关注所收集数据的状态呢？首先，如果没有数据，机器学习模型将无法学习。虽然这是显而易见的事情，但需注意的是，构建模型的强大与否，有一部分是取决于所给予的数据。常言道，“输入是垃圾，产出也会是垃圾”。只有确定了收集的数据是相关并且干净的，机器学习的模型才能变得强大。也只有这样，才能如预想般操作数据，并产出有价值的结果。

当使用一种特定模型时，不是所有类型的数据都是合适的。例如，高维数据（例如文本数据）在一些特定模型中运行得就不是很好。另外一些模型假设变量是正态分布的，这种假设并非总是成立的。因此在收集需要的数据时应格外小心，并确保知道数据和模型是如何交互的。

数据科学家花如此多时间收集和组织数据还有另一个原因，那就是数据通常是凌乱的并且很难聚合。在大多数的组织中，数据经常以多种格式保存在多种系统中，并且拥有多种访问权限。通常情况下，很难像指定一个文件路径那样简单地为模型提供训练数据集合。

为了形成一个训练 / 测试集合，或者提供变量给一个模型以用于预测，可能需要处理各种各样的数据类型，例如 CSV、JSON、数据库中的表，等等。除此之外还可能需要转换其中一些数值。常见的转换包括解析时间格式、转化分类数据为数字数据、数值归一化、使用一些函数交叉处理数据。然而，不能总是假设一个特定变量的所有可能取值都会出现，或者能被同一种方式解析。

数据经常有数值缺失、类型混合、数值损坏的情况发生，如何处理这些场景将直接影响所构建模型的质量，因此需要仔细地收集、组织和理解所采用的数据。

虽然本书大部分内容的重点是各种各样的建模技术，但是读者仍应当将数据的收集、解析和组织当作是数据科学项目成功的一个（甚至是最）关键的组成部分。如果在机器学习的项目开发中，这部分数据没有保持高级别的完整性，那么从长远来看将会导致各种麻烦。

1.1 数据处理-Gopher 方式

和其他用于数据科学 / 分析的处理语言相比，Go 提供了强大的数据操作和解析能力。虽然其他语言（例如 Python 或者 R）能让用户很快地进行数据交互，但是这些语言经常倡导打破完整性的便利。如果在代码中经常采用动态和交互性数据探索，会导致行为变得奇怪。

以这个简单的 CSV 文件为例：

```
1,blah1
2,blah2
3,blah3
```

可以迅速写出一些 Python 代码去解析这个 CSV 文件，然后在无须了解数据是什么类型的情况下，输出整数列的最大值。

```
import pandas as pd

# Define column names.
cols = [
 'integercolumn',
 'stringcolumn'
 ]

# Read in the CSV with pandas.
data = pd.read_csv('myfile.csv', names=cols)

# Print out the maximum value in the integer column.
print(data['integercolumn'].max())
```

这个简单的程序将输出正确的结果：

```
$ python myprogram.py
3
```

如下所示，现在移除其中一个整数用于模拟数值丢失的情况。

```
1,blah1
2,blah2
,blah3
```

Python 程序随后就会打破数据结果的完整性，明确来讲，上述程序仍然会保持运

行，不会告诉用户有任何不一样的地方，仍然会产生数值，但却是一个不同类型的数值。

```
$ python myprogram.py
2.0
```

这种情况是无法接受的，一个整数数值可以在用户察觉不到任何变化的时候凭空消失。这种很难追踪的行为会对建模产生深远的影响。一般来说，一旦选择了动态类型和抽象的便利性，就需要接受这种行为上的变化。

重点并不是说 Python 无法处理这种行为，因为 Python 专家很快会认识到其实是有办法解决这个问题的。重点是这种便利性在默认情况下并没有提升完整性，因此很容易搬起石头砸自己的脚。

从另一方面来说，可以利用 Go 语言的静态类型和明确的错误处理方式，来确保数据的解析跟期望的一致。在这个简单例子中，可以毫不费力地写一些 Go 代码去解析的 CSV（现在暂不用关心具体的技术细节）：

```
// Open the CSV.
f, err := os.Open("myfile.csv")
if err != nil {
    log.Fatal(err)
}

// Read in the CSV records.
r := csv.NewReader(f)
records, err := r.ReadAll()
if err != nil {
    log.Fatal(err)
}

// Get the maximum value in the integer column.
var intMax int
for _, record := range records {

    // Parse the integer value.
    intVal, err := strconv.Atoi(record[0])
    if err != nil {
        log.Fatal(err)
    }

    // Replace the maximum value if appropriate.
    if intVal > intMax {
        intMax = intVal
    }
}

// Print the maximum value.
fmt.Println(intMax)
```

如果这个 CSV 文件的所有整型数值都存在，它将会产生同样正确的结果：

```
$ go build
$ ./myprogram
3
```

但是和之前的 Python 代码相比，Go 代码在 CSV 输入遇到非预期的情况下（例如在移除数值 3 的情况下），会给予用户提示：

```
$ go build
$ ./myprogram
2017/04/29 12:29:45 strconv.ParseInt: parsing "": invalid syntax
```

以上 Go 语言的程序维持了数据完整性，用户可以毫无顾虑地用一种适合自己的方法来处理数值丢失情况。

1.2 Go 语言收集和组织数据的最佳实践

正如前文所提及的，在数据收集、解析和组织的过程中，Go 语言可以维持高级别的数据完整性。在任何为机器学习工作流程准备数据的时候，都应该确保采用了 Go 语言的独特特性。

一般来说，使用 Go 语言的数据科学家 / 分析师在收集和组织数据的时候，应遵循以下最佳实践。这些最佳实践致力于维护应用中数据的完整性，并且使用户可以重现任何分析数据：

1. **检查并强制指定期望的类型**：这看起来可能是显而易见的，但是当使用动态类型语言的时候，却经常被忽视。所以虽然有些烦琐，但是明确地将数据解析为期望类型和明确地处理相关错误能解决使用过程中大量令人头疼的问题。
2. **标准化和简化数据输入 / 输出**：有许多第三方的程序包可用于处理特定类型的数据，或者同特定类型的数据源进行交互（本书会包含其中一部分）。但如果将与数据源的交互方式标准化，尤其是要以 `stdlib` 的使用为中心，就可以开发出可预测的模型，并且在团队中维持一致性。举例来说，应该选择 `database/sql` 同数据库进行交互，而不是选择各种各样第三方的 API 和 DSL。
3. **数据的版本控制**：不同的训练数据、选择的参数和输入的数据，会让机器学习模型产生极其不同的结果。因此，在没有对代码和数据进行版本控制的情况下，是不可能重现结果的。本章后面将会讨论数据版本管理的恰当技术。

一旦开始偏离这些常见的原则，就应该立刻停止手头的工作。因为可能会因为便利性而牺牲完整性，而这是一条危险的道路。在本书后续章节中涉及各式各样的数据格式 / 源时，会严格遵守这些原则。