HZ BOOKS
华章IT

智能系统与技术丛书

从Python和数学基础，到机器学习和TensorFlow理论，再到深度学习的应用和扩展，为深度学习提供全栈式内容解决方案

Python Deep Learning with TensorFlow

Python深度学习
基于TensorFlow

吴茂贵 王冬 李涛 杨本法 著

机械工业出版社
China Machine Press

智能系统与技术丛书

Python Deep Learning with TensorFlow

Python深度学习
基于TensorFlow

吴茂贵 王冬 李涛 杨本法 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Python 深度学习：基于 TensorFlow/ 吴茂贵等著 . —北京：机械工业出版社，2018.9
（智能系统与技术丛书）

ISBN 978-7-111-60972-8

I. P… II. 吴… III. 软件工具－程序设计 IV. TP311.561

中国版本图书馆 CIP 数据核字（2018）第 218895 号

Python 深度学习：基于 TensorFlow

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：李 艺 责任校对：张惠兰
印 刷：北京市荣盛彩色印刷有限公司 版 次：2018 年 10 月第 1 版第 1 次印刷
开 本：186mm×240mm 1/16 印 张：21.25
书 号：ISBN 978-7-111-60972-8 定 价：79.00 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）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

PREFACE

前　言

为什么写这本书

人工智能新时代学什么？我们知道，Python 是人工智能的首选语言，深度学习是人工智能的核心，而 TensorFlow 是深度学习框架中的 No.1。所以我们在本书中将这三者有机结合，希望借此把这些目前应用最广、最有前景的工具和算法分享给大家。

人工智能新时代如何学？市面上介绍这些工具和深度学习理论的书籍已有很多，而且不乏经典大作，如讲机器学习理论和算法的有周志华老师的《机器学习》；介绍深度学习理论和算法的有伊恩·古德费洛等编著的《深度学习》；介绍 TensorFlow 实战的有黄文坚、唐源编著的《 TensorFlow 实战》、山姆·亚伯拉罕等编著的《面向机器智能的 TensorFlow 实践》等。这些都是非常经典的大作，如果你对机器学习、深度学习、人工智能感兴趣的话，这些书均值得一读。

本书在某些方面或许无法和它们相比，但我觉得也会有不少让你感到满意，甚至惊喜的地方。本书的特点具体包括以下几个方面。

1. 内容选择：提供全栈式的解决方案

深度学习涉及范围比较广，既有对基础、原理的一些要求，也有对代码实现的要求。如何在较短时间内快速提高深度学习的水平？如何尽快把所学运用到实践中？这方面虽然没有捷径可言，但却有方法可循。本书基于这些考量，希望能给你提供一站式解决方案。具体内容包括：机器学习与深度学习的三大基石（线性代数、概率与信息论及数值分析）；机器学习与深度学习的基本理论和原理；机器学习与深度学习的常用开发工具（Python、TensorFlow、Keras 等），此外还有 TensorFlow 的高级封装及多个综合性实战项目等。

2. 层次安排：找准易撕口、快速实现由点到面的突破

我们打开塑料袋时，一般从易撕口开始，这样即使再牢固的袋子也很容易打开。面对深度学习这个“牢固袋子”，我们也可采用类似方法，找准易撕口。如果没有，就创造一个易撕口，通过这个易撕口，实现点到面的快速扩展。本书在介绍很多抽象、深奥的算法时

采用了这种方法。我们知道 BP 算法、循环神经网络是深度学习中的两块硬骨头，所以在介绍 BP 算法时，先介绍单个神经如何实现 BP 算法这个易撕口，再延伸到一般情况；在介绍循环神经网络时，我们也以一个简单实例为易撕口，再延伸到一般情况。希望通过这种方式，能帮助你把难题化易、把大事化小、把不可能转换为可能。

3. 表达形式：让图说话，一张好图胜过千言万语

在机器学习、深度学习中有很多抽象的概念、复杂的算法、深奥的理论，如 Numpy 的广播机制、神经网络中的共享参数、动量优化法、梯度消失或爆炸等，这些内容如果只用文字来描述，可能很难达到茅塞顿开的效果，但如果用一些图形来展现，再加上适当的文字说明，往往能取得非常好的效果，正所谓一张好图胜过千言万语。

除了以上谈到的三个方面，为了帮助大家更好理解、更快掌握机器学习、深度学习这些人工智能的核心内容，本书还包含了其他方法。我们希望通过这些方法或方式带给你不一样的理解和体验，使抽象数学不抽象、深度学习不深奥、复杂算法不复杂，这或许就是我们写这本书的主要目的。

至于人工智能（AI）的重要性，想必不用多说了。如果说 2016 年前属于摆事实论证的阶段，那么 2016 年后已进入事实胜于雄辩的阶段了，而 2018 年后应该属于撸起袖子加油干的阶段。目前各行各业都忙于 AI+，给人“忽如一夜春风来，千树万树梨花开”的感觉！

本书特色

要说特色的话，就是上面谈到的几点，概括来说就是：把理论原理与代码实现相结合；找准切入点，从简单到一般，把复杂问题简单化；图文并茂使抽象问题直观化；实例说明使抽象问题具体化。希望通过阅读本书，能给你带来新的视角、新的理解。

读者对象

- 对机器学习、深度学习感兴趣的广大在校学生、在职人员。
- 对 Python、TensorFlow 感兴趣，并希望进一步提升的在校学生、在职人员。

如何阅读本书

本书共 22 章，按照“基础→应用→扩展”的顺序展开，分为三个部分。

第一部分（第 1 ～ 5 章）为 Python 和应用数学基础部分：第 1 章介绍 Python 和 TensorFlow 的基石 Numpy；第 2 章介绍深度学习框架的鼻祖 Theano；第 3 ～ 5 章介绍机器

学习、深度学习算法应用数学基础，包括线性代数、概率与信息论、概率图等内容。

第二部分（第 6 ～ 20 章）为深度学习理论与应用部分：第 6 章为机器学习基础，也是深度学习基础，其中包含很多机器学习的经典理论和算法；第 7 章为深度学习理论及方法；第 8~10 章介绍 TensorFlow 基于 CPU、GPU 版本的安装及使用，TensorFlow 基础，TensorFlow 的一些新 API，如 Dataset API、Estimator API 等（基于 TensorFlow1.6 版本）；第 11 ～ 15 章为深度学习中神经网络方面的模型及 TensorFlow 实战案例；第 16 章介绍 TensorFlow 的高级封装，如 Keras、Estimator、TFLearn 等内容；第 17 ～ 20 章为 TensorFlow 综合实战案例，包括图像识别、自然语言处理等内容。

第三部分（第 21 ～ 22 章）为扩展部分：介绍强化学习、生成式对抗网络等内容。

勘误和支持

书中代码和数据的下载地址为 http://www.feiguyunai.com。由于笔者水平有限，加之编写时间仓促，书中难免出现错误或不准确的地方，恳请读者批评指正。你可以通过 QQ（1715408972）给我们反馈，也可以加入 QQ 交流群（763746291）进行交流，非常感谢你的支持和帮助。

致谢

在本书编写过程中，得到很多同事、朋友、老师和同学的支持！感谢张粤磊、张魁、刘未昕等负责后台环境的搭建和维护工作。感谢博世的王红星、上海理工管理学院的郁明敏，在百忙中挤出时间帮忙审稿；感谢上海交大慧谷的程国旗老师、东方易通的杨易老师、容大培训的童金浩老师、赣南师大的许景飞老师等对我们的支持和帮助！

感谢机械工业出版社的杨福川老师、李艺老师给予本书的大力支持和帮助。

最后，感谢我的爱人赵成娟，在繁忙的教学之余帮助审稿，提出不少改进意见或建议。

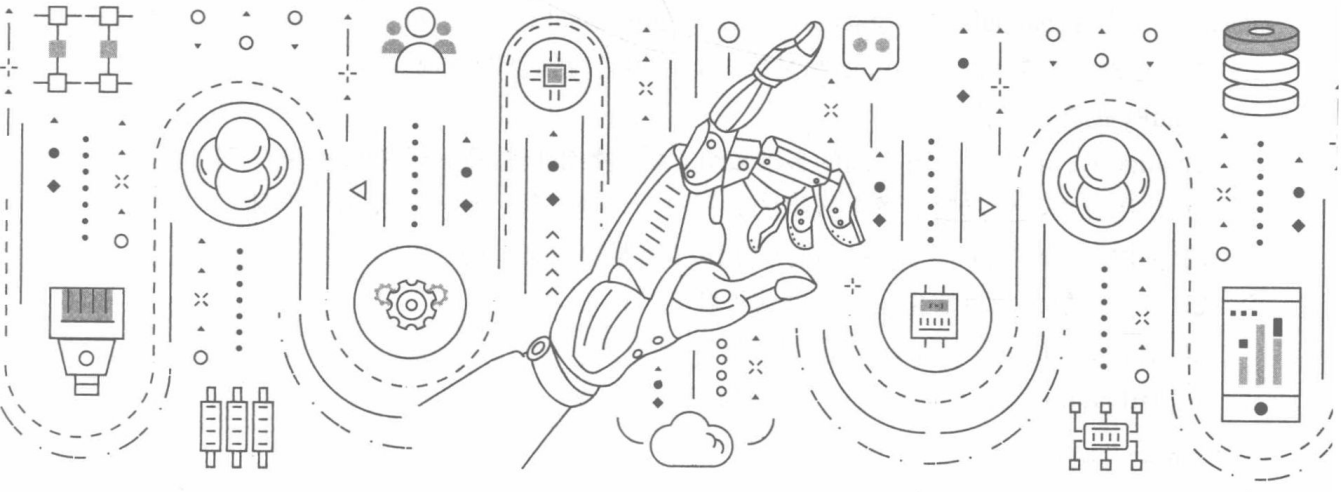
吴茂贵

CONTENTS

目　录

第一部分

Python 及应用数学基础



CHAPTER 1

第 1 章

NumPy 常用操作

NumPy 是 Python 的基础，更是数据科学的通用语言，而且与 TensorFlow 关系密切，所以我们把它列为第一章。

NumPy 为何如此重要？实际上 Python 本身含有列表（list）和数组（array），但对于大数据来说，这些结构有很多不足。因列表的元素可以是任何对象，因此列表中所保存的是对象的指针。这样为了保存一个简单的 [1,2,3]，都需要有 3 个指针和 3 个整数对象。对于数值运算来说，这种结构显然比较浪费内存和 CPU 计算时间。至于 array 对象，它直接保存数值，和 C 语言的一维数组比较类似。但是由于它不支持多维，也没有各种运算函数，因此也不适合做数值运算。

NumPy（Numerical Python 的简称）的诞生弥补了这些不足，它提供了两种基本的对象：ndarray（N-dimensional array object）和 ufunc（universal function object）。ndarray 是存储单一数据类型的多维数组，而 ufunc 则是能够对数组进行处理的函数。

NumPy 的主要特点：

- ndarray，快速，节省空间的多维数组，提供数组化的算术运算和高级的广播功能。
- 使用标准数学函数对整个数组的数据进行快速运算，而不需要编写循环。
- 读取 / 写入磁盘上的阵列数据和操作存储器映像文件的工具。
- 线性代数，随机数生成，以及傅里叶变换的能力。
- 集成 C、C++、Fortran 代码的工具。

在使用 NumPy 之前，需要先导入该模块：

```
import numpy as np
```

本章主要内容如下：

- 如何生成 NumPy 的 ndarray 的几种方式。
- 如何存取元素。
- 如何操作矩阵。
- 如何合并或拆分数据。

- 简介 NumPy 的通用函数。
- 简介 NumPy 的广播机制。

1.1　生成 ndarray 的几种方式

NumPy 封装了一个新的数据类型 ndarray，一个多维数组对象，该对象封装了许多常用的数学运算函数，方便我们进行数据处理以及数据分析，那么如何生成 ndarray 呢？这里我们介绍生成 ndarray 的几种方式，如从已有数据中创建；利用 random 创建；创建特殊多维数组；使用 arange 函数等。

1. 从已有数据中创建

直接对 python 的基础数据类型（如列表、元组等）进行转换来生成 ndarray。

（1）将列表转换成 ndarray

```
import numpy as np
list1 = [3.14,2.17,0,1,2]
nd1 = np.array(list1)
print(nd1)
print(type(nd1))
```

打印结果：

```
[ 3.14  2.17  0.    1.    2.  ]
<class 'numpy.ndarray'>
```

（2）嵌套列表可以转换成多维 ndarray

```
import numpy as np
list2 = [[3.14,2.17,0,1,2],[1,2,3,4,5]]
nd2 = np.array(list2)
print(nd2)
print(type(nd2))
```

打印结果：

```
[[ 3.14  2.17  0.    1.    2.  ]
 [ 1.    2.    3.    4.    5.  ]]
<class 'numpy.ndarray'>
```

如果把（1）和（2）中的列表换成元组也同样适合。

2. 利用 random 模块生成 ndarray

在深度学习中，我们经常需要对一些变量进行初始化，适当的初始化能提高模型的性能。通常我们用随机数生成模块 random 来生成，当然 random 模块又分为多种函数：random 生成 0 到 1 之间的随机数；uniform 生成均匀分布随机数；randn 生成标准正态的随机数；normal 生成正态分布；shuffle 随机打乱顺序；seed 设置随机数种子等。下面我们列举几个

简单示例。

```
import numpy as np

nd5 = np.random.random([3,3])
print(nd5)
print(type(nd5))
```

打印结果：

```
[[ 0.88900951  0.47818541  0.91813526]
 [ 0.48329167  0.63730656  0.14301479]
 [ 0.9843789   0.99257093  0.24003961]]
<class 'numpy.ndarray'>
```

生成一个随机种子，对生成的随机数打乱。

```
import numpy as np

np.random.seed(123)
nd5_1 = np.random.randn(2,3)
print(nd5_1)
np.random.shuffle(nd5_1)
print("随机打乱后数据")
print(nd5_1)
print(type(nd5_1))
```

打印结果：

```
[[-1.0856306   0.99734545  0.2829785 ]
 [-1.50629471 -0.57860025  1.65143654]]
```

随机打乱后数据为：

```
[[-1.50629471 -0.57860025  1.65143654]
 [-1.0856306   0.99734545  0.2829785 ]]
<class 'numpy.ndarray'>
```

3. 创建特定形状的多维数组

数据初始化时，有时需要生成一些特殊矩阵，如 0 或 1 的数组或矩阵，这时我们可以利用 np.zeros、np.ones、np.diag 来实现，下面我们通过几个示例来说明。

```
import numpy as np

# 生成全是 0 的 3x3 矩阵
nd6 = np.zeros([3,3])
# 生成全是 1 的 3x3 矩阵
nd7 = np.ones([3,3])
# 生成 3 阶的单位矩阵
nd8= np.eye(3)
# 生成 3 阶对角矩阵
print (np.diag([1, 2, 3]))
```

我们还可以把生成的数据保存到磁盘，然后从磁盘读取。

```
import numpy as np
nd9 = np.random.random([5,5])
np.savetxt(X=nd9,fname='./test2.txt')
nd10 = np.loadtxt('./test2.txt')
```

4. 利用 arange 函数

arange 是 numpy 模块中的函数，其格式为：arange([start,] stop[, step,], dtype=None)。根据 start 与 stop 指定的范围，以及 step 设定的步长，生成一个 ndarray，其中 start 默认为 0，步长 step 可为小数。

```
import numpy as np

print(np.arange(10))
print(np.arange(0,10))
print(np.arange(1, 4,0.5))
print(np.arange(9, -1, -1))
```

1.2　存取元素

上节我们介绍了生成 ndarray 的几种方法，数据生成后，如何读取我们需要的数据？这节我们介绍几种读取数据的方法。

```
import numpy as np
np.random.seed(2018)
nd11 = np.random.random([10])
# 获取指定位置的数据，获取第 4 个元素
nd11[3]
# 截取一段数据
nd11[3:6]
# 截取固定间隔数据
nd11[1:6:2]
# 倒序取数
nd11[::-2]
# 截取一个多维数组的一个区域内数据
nd12=np.arange(25).reshape([5,5])
nd12[1:3,1:3]
# 截取一个多维数组中，数值在一个值域之内的数据
nd12[(nd12>3)&(nd12<10)]
# 截取多维数组中，指定的行，如读取第 2,3 行
nd12[[1,2]]  # 或 nd12[1:3,:]
## 截取多维数组中，指定的列，如读取第 2,3 列
nd12[:,1:3]
```

如果你对上面这些获取方式还不是很清楚，没关系，下面我们通过图形的方式说明如何获取多维数组中的元素，如图 1-1 所示，左边为表达式，右边为对应获取元素。

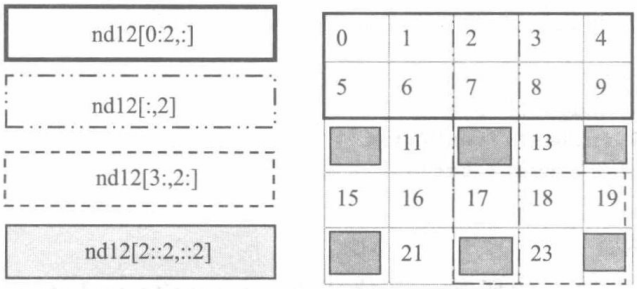


图 1-1　获取多维数组中的元素

获取数组中的部分元素除通过指定索引标签外，还可以使用一些函数来实现，如通过 random.choice 函数从指定的样本中进行随机抽取数据。

```
import numpy as np
from numpy import random as nr

a=np.arange(1,25,dtype=float)
c1=nr.choice(a,size=(3,4))  #size 指定输出数组形状
c2=nr.choice(a,size=(3,4),replace=False)  #replace 缺省为 True，即可重复抽取
# 下式中参数 p 指定每个元素对应的抽取概率，默认为每个元素被抽取的概率相同
c3=nr.choice(a,size=(3,4),p=a / np.sum(a))
 print("随机可重复抽取")
print(c1)
print("随机但不重复抽取")
print(c2)
print("随机但按制度概率抽取")
print(c3)
```

打印结果：

```
随机可重复抽取
[[  7.  22.  19.  21.]
 [  7.   5.   5.   5.]
 [  7.   9.  22.  12.]]
随机但不重复抽取
[[ 21.   9.  15.   4.]
 [ 23.   2.   3.   7.]
 [ 13.   5.   6.   1.]]
随机但按制度概率抽取
[[ 15.  19.  24.   8.]
 [  5.  22.   5.  14.]
 [  3.  22.  13.  17.]]
```

1.3　矩阵操作

深度学习中经常涉及多维数组或矩阵的运算，正好 NumPy 模块提供了许多相关的计算