

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



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

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

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、深度学习中有许多抽象的概念、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、上海理工大学管理学院的郁明敏，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

目 录

前言	
第一部分 Python 及应用数学基础	
第 1 章 NumPy 常用操作2	
1.1 生成 ndarray 的几种方式.....3	
1.2 存取元素.....5	
1.3 矩阵操作.....6	
1.4 数据合并与展平.....7	
1.5 通用函数.....9	
1.6 广播机制.....11	
1.7 小结.....12	
第 2 章 Theano 基础13	
2.1 安装.....14	
2.2 符号变量.....15	
2.3 符号计算图模型.....17	
2.4 函数.....18	
2.5 条件与循环.....21	
2.6 共享变量.....23	
2.7 小结.....24	
第 3 章 线性代数25	
3.1 标量、向量、矩阵和张量.....25	
3.2 矩阵和向量运算.....28	
3.3 特殊矩阵与向量.....29	
3.4 线性相关性及向量空间.....31	
3.5 范数.....32	
3.6 特征值分解.....33	
3.7 奇异值分解.....34	
3.8 迹运算.....35	
3.9 实例：用 Python 实现主成分 分析.....36	
3.10 小结.....39	
第 4 章 概率与信息论40	
4.1 为何要学概率、信息论.....40	
4.2 样本空间与随机变量.....41	
4.3 概率分布.....42	
4.3.1 离散型随机变量.....42	
4.3.2 连续型随机变量.....45	
4.4 边缘概率.....47	
4.5 条件概率.....47	
4.6 条件概率的链式法则.....48	
4.7 独立性及条件独立性.....48	
4.8 期望、方差及协方差.....49	
4.9 贝叶斯定理.....52	
4.10 信息论.....53	
4.11 小结.....56	
第 5 章 概率图模型57	
5.1 为何要引入概率图.....57	

5.2	使用图描述模型结构	58
5.3	贝叶斯网络	59
5.3.1	隐马尔可夫模型简介	60
5.3.2	隐马尔可夫模型三要素	60
5.3.3	隐马尔可夫模型三个基本问题	61
5.3.4	隐马尔可夫模型简单实例	62
5.4	马尔可夫网络	64
5.4.1	马尔可夫随机场	64
5.4.2	条件随机场	65
5.4.3	实例：用 Tensorflow 实现条件随机场	66
5.5	小结	70

第二部分 深度学习理论与应用

第 6 章	机器学习基础	72
6.1	监督学习	72
6.1.1	线性模型	73
6.1.2	SVM	77
6.1.3	贝叶斯分类器	79
6.1.4	集成学习	81
6.2	无监督学习	84
6.2.1	主成分分析	84
6.2.2	k-means 聚类	84
6.3	梯度下降与优化	85
6.3.1	梯度下降简介	86
6.3.2	梯度下降与数据集大小	87
6.3.3	传统梯度优化的不足	89
6.3.4	动量算法	90
6.3.5	自适应算法	92
6.3.6	有约束最优化	95
6.4	前馈神经网络	96
6.4.1	神经元结构	97

6.4.2	感知机的局限	98
6.4.3	多层神经网络	99
6.4.4	实例：用 TensorFlow 实现 XOR	101
6.4.5	反向传播算法	103
6.5	实例：用 Keras 构建深度学习架构	109
6.6	小结	109

第 7 章 深度学习挑战与策略

7.1	正则化	110
7.1.1	正则化参数	111
7.1.2	增加数据量	115
7.1.3	梯度裁剪	116
7.1.4	提前终止	116
7.1.5	共享参数	117
7.1.6	Dropout	117
7.2	预处理	119
7.2.1	初始化	120
7.2.2	归一化	120
7.3	批量化	121
7.3.1	随机梯度下降法	121
7.3.2	批标准化	122
7.4	并行化	124
7.4.1	TensorFlow 利用 GPU 加速	124
7.4.2	深度学习并行模式	125
7.5	选择合适的激活函数	127
7.6	选择合适代价函数	128
7.7	选择合适的优化算法	129
7.8	小结	130

第 8 章 安装 TensorFlow

8.1	TensorFlow CPU 版的安装	131
-----	---------------------	-----

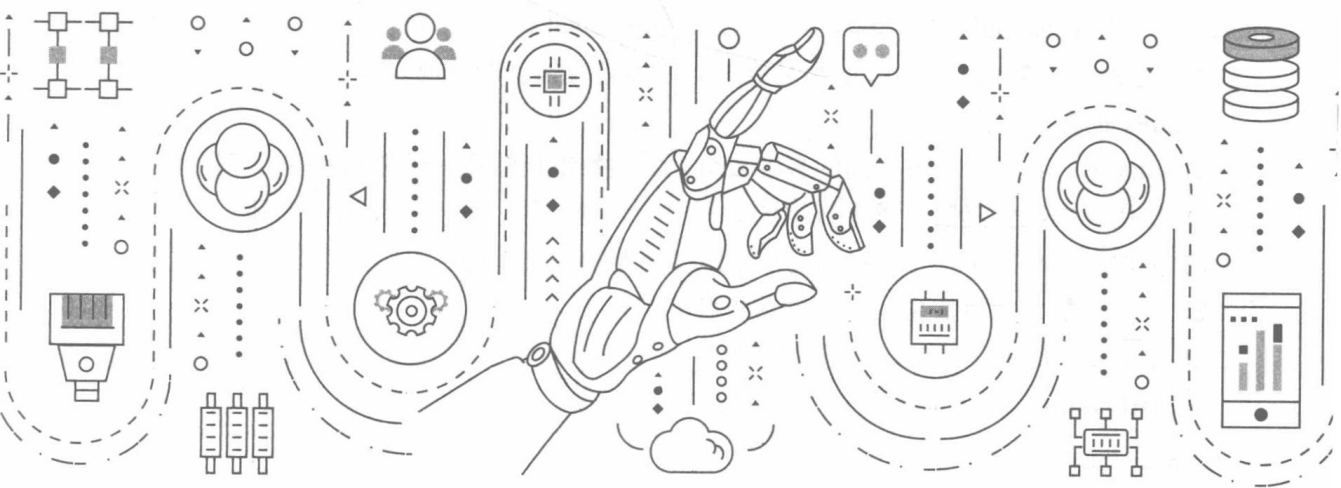
8.2 TensorFlow GPU 版的安装·····	132	10.6.3 创建迭代器·····	174
8.3 配置 Jupyter Notebook·····	136	10.6.4 从迭代器中获取数据·····	174
8.4 实例: CPU 与 GPU 性能比较·····	137	10.6.5 读入输入数据·····	175
8.5 实例: 单 GPU 与多 GPU 性能比较·····	138	10.6.6 预处理数据·····	175
8.6 小结·····	140	10.6.7 批处理数据集元素·····	176
		10.6.8 使用高级 API·····	176
第 9 章 TensorFlow 基础·····	141	10.7 小结·····	177
9.1 TensorFlow 系统架构·····	141	第 11 章 TensorFlow 神经元函数·····	178
9.2 数据流图·····	143	11.1 激活函数·····	178
9.3 TensorFlow 基本概念·····	144	11.1.1 sigmoid 函数·····	179
9.3.1 张量·····	144	11.1.2 tanh 函数·····	179
9.3.2 算子·····	145	11.1.3 relu 函数·····	180
9.3.3 计算图·····	146	11.1.4 softplus 函数·····	181
9.3.4 会话·····	146	11.1.5 dropout 函数·····	181
9.3.5 常量·····	148	11.2 代价函数·····	181
9.3.6 变量·····	149	11.2.1 sigmoid_cross_entropy_ with_logits 函数·····	182
9.3.7 占位符·····	153	11.2.2 softmax_cross_entropy_ with_logits 函数·····	183
9.3.8 实例: 比较 constant、 variable 和 placeholder·····	154	11.2.3 sparse_softmax_cross_ entropy_with_logits 函数·····	184
9.4 TensorFlow 实现数据流图·····	156	11.2.4 weighted_cross_entropy_ with_logits 函数·····	184
9.5 可视化数据流图·····	156	11.3 小结·····	185
9.6 TensorFlow 分布式·····	158	第 12 章 TensorFlow 自编码器·····	186
9.7 小结·····	160	12.1 自编码简介·····	186
第 10 章 TensorFlow 图像处理·····	162	12.2 降噪自编码·····	188
10.1 加载图像·····	162	12.3 实例: TensorFlow 实现 自编码·····	188
10.2 图像格式·····	163	12.4 实例: 用自编码预测信用卡 欺诈·····	191
10.3 把图像转换为 TFRecord 文件·····	164	12.5 小结·····	197
10.4 读取 TFRecord 文件·····	165		
10.5 图像处理实例·····	166		
10.6 全新的数据读取方式—— Dataset API·····	170		
10.6.1 Dataset API 架构·····	170		
10.6.2 构建 Dataset·····	171		

第 13 章 TensorFlow 实现	
Word2Vec	198
13.1 词向量及其表达	198
13.2 Word2Vec 原理	199
13.2.1 CBOW 模型	200
13.2.2 Skim-gram 模型	200
13.3 实例: TensorFlow 实现 Word2Vec	201
13.4 小结	206
第 14 章 TensorFlow 卷积神经网络	207
14.1 卷积神经网络简介	207
14.2 卷积层	208
14.2.1 卷积核	209
14.2.2 步幅	211
14.2.3 填充	212
14.2.4 多通道上的卷积	213
14.2.5 激活函数	214
14.2.6 卷积函数	215
14.3 池化层	216
14.4 归一化层	217
14.5 TensorFlow 实现简单卷积神经网络	218
14.6 TensorFlow 实现进阶卷积神经网络	219
14.7 几种经典卷积神经网络	223
14.8 小结	224
第 15 章 TensorFlow 循环神经网络	226
15.1 循环神经网络简介	226
15.2 前向传播与随时间反向传播	228
15.3 梯度消失或爆炸	231
15.4 LSTM 算法	232
15.5 RNN 其他变种	235
15.6 RNN 应用场景	236
15.7 实例: 用 LSTM 实现分类	237
15.8 小结	241
第 16 章 TensorFlow 高层封装	242
16.1 TensorFlow 高层封装简介	242
16.2 Estimator 简介	243
16.3 实例: 使用 Estimator 预定义模型	245
16.4 实例: 使用 Estimator 自定义模型	247
16.5 Keras 简介	252
16.6 实例: Keras 实现序列式模型	253
16.7 TFLearn 简介	255
16.7.1 利用 TFLearn 解决线性回归问题	256
16.7.2 利用 TFLearn 进行深度学习	256
16.8 小结	257
第 17 章 情感分析	258
17.1 深度学习与自然语言处理	258
17.2 词向量简介	259
17.3 循环神经网络	260
17.4 迁移学习简介	261
17.5 实例: TensorFlow 实现情感分析	262
17.5.1 导入数据	262
17.5.2 定义辅助函数	267
17.5.3 构建 RNN 模型	267
17.5.4 调优超参数	269
17.5.5 训练模型	270
17.6 小结	272

第 18 章 利用 TensorFlow 预测 乳腺癌	273	20.3 实施步骤	307
18.1 数据说明	273	20.3.1 数据准备	307
18.2 数据预处理	274	20.3.2 预处理数据	307
18.3 探索数据	276	20.3.3 训练模型	309
18.4 构建神经网络	279	20.3.4 测试模型	313
18.5 训练并评估模型	281	20.4 小结	316
18.6 小结	283		
第 19 章 聊天机器人	284	第三部分 扩展篇	
19.1 聊天机器人原理	284	第 21 章 强化学习基础	318
19.2 带注意力的框架	286	21.1 强化学习简介	318
19.3 用 TensorFlow 实现聊天 机器人	289	21.2 强化学习常用算法	320
19.3.1 接口参数说明	290	21.2.1 Q-Learning 算法	320
19.3.2 训练模型	293	21.2.2 Sarsa 算法	322
19.4 小结	302	21.2.3 DQN 算法	322
 		21.3 小结	324
第 20 章 人脸识别	303	第 22 章 生成式对抗网络	325
20.1 人脸识别简介	303	22.1 GAN 简介	325
20.2 项目概况	306	22.2 GAN 的改进版本	327
		22.3 小结	329

第一部分

Python 及应用数学基础



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？这节课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。

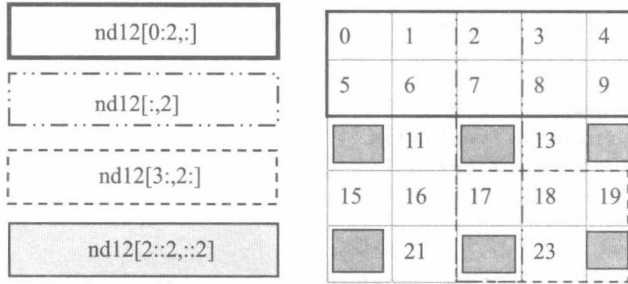


图 1-1 获取多维数组中的元素

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