HZ BOOKS
华章IT

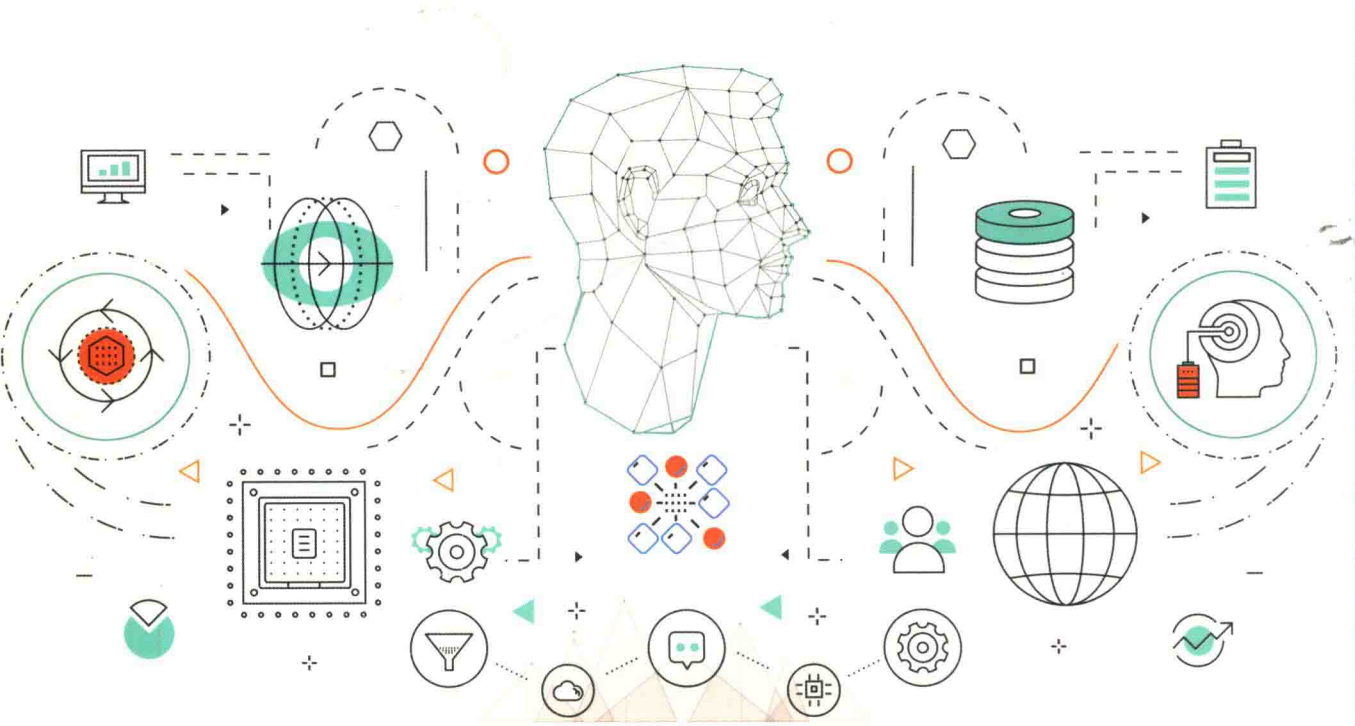
智能系统与技术丛书

深度学习一线研发人员撰写，讲解神经网络设计与PyTorch应用

以张量为基，以科学计算为线，以神经网络为面，以PyTorch为器，高效铸造AI应用

神经网络与PyTorch实战

肖智清 著



机械工业出版社
China Machine Press

智能系统与技术丛书

神经网络与PyTorch实战

肖智清 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

神经网络与 PyTorch 实战 / 肖智清著 . —北京：机械工业出版社，2018.8
（智能系统与技术丛书）

ISBN 978-7-111-60577-5

I. 神… II. 肖… III. 机器学习 IV. TP181

中国版本图书馆 CIP 数据核字（2018）第 171207 号

神经网络与 PyTorch 实战

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：高婧雅 责任校对：李秋荣
印　刷：北京市兆成印刷有限责任公司 版　次：2018 年 8 月第 1 版第 1 次印刷
开　本：186mm × 240mm 1/16 印　张：13.75
书　号：ISBN 978-7-111-60577-5 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）88379624 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

人工智能（Artificial Intelligence，AI）正在对各行各业产生决定性的影响。神经网络（Neutral Network）作为当今人工智能的旗舰模型，将成为各行业从业人员的必备知识。PyTorch 作为简单、易用、灵活的人工神经网络库，是学习神经网络的优秀工具。在此诚邀你一起学习 PyTorch 和神经网络，拥抱人工智能的大时代。

本书特色

本书的目的是成为简单易学的神经网络中文图书，因此将理论和实践有机结合。

- **在理论方面**，本书带你从零开始入门神经网络。本书将基于“张量”这一既简单又前沿的概念，讲授必要的数学知识，与你一起轻松掌握深度学习的核心理论。
- **在实践方面**，本书带你在 Windows 系统或是 macOS 系统中轻松安装新版本的 PyTorch 开发环境，由浅入深学习 PyTorch 开发，让你在自己的电脑上轻松实现超酷的人工智能算法与应用。

读者对象

本书面向以下读者：

- 想了解人工智能、机器学习、神经网络、深度学习等热门技术的人士；
- 想运用人工智能、机器学习、神经网络、深度学习解决实际问题的人士。

你也许没有学过编程，或是英语不太好，或是好久没有用数学了，没关系——本书将以简单易懂的方式，给出所有必要的知识。只要你有兴趣，就能轻松快乐地学会神经网络。

本书主要内容

全书逻辑上分为三个部分。

- 第 1～2 章：感性介绍神经网络的基础知识，并给出一个利用 PyTorch 搭建神经网络解决实际问题的例子，使你初步了解神经网络和 PyTorch；
- 第 3～9 章：介绍基于 PyTorch 的科学计算和神经网络搭建，涵盖了几乎所有 PyTorch 基础知识，涉及了所有神经网络的常用结构，并通过例子使你完全掌握神经网络的原理和应用；
- 第 10～11 章：介绍生成对抗网络和增强学习，使你了解更多神经网络的实际用法。

代码下载与技术支持

本书的所有代码都可以在 http://github.com/zhiqingxiao/pytorch-book 上下载。笔者会实时更新代码，保证代码能够在新版本的 PyTorch 下运行。

在此推荐你加入本书学习交流 QQ 群 698847007，关注微信公众号 pytorcher。如果有任何意见、建议或者有通过网络搜索仍不能解决的问题，可以在 QQ 群里提问、在 GitHub 上提 Issue，或给笔者发邮件，笔者的电子邮箱是 xzq.xiaozhiqing@gmail.com。

致谢

在此感谢为本书出版做出贡献的所有工作人员。机械工业出版社华章公司的高婧雅对本书的写作提出了很多建设性意见。同时，还要感谢其他编辑为提升本书质量做出的大量工作，与他们合作是一个愉快的过程。最后，还要感谢我的上司、同事和亲友，特别是我的爸爸、妈妈，他们在本书写作期间给予我极大的支持。

感谢你选择本书。祝你学习快乐！

推荐阅读

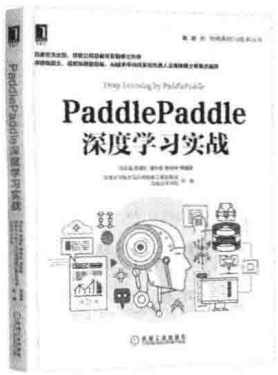

深度学习
核心技术、工具与案例解析

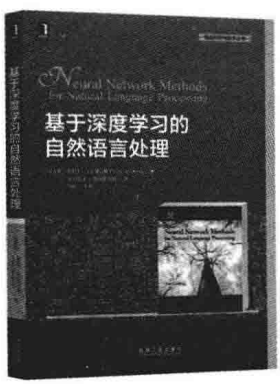

深度学习
主流框架和编程实战


自然语言处理与深度学习
通过C语言模拟


深度学习
Java语言实现


O'REILLY
TensorFlow
学习指南
深度学习系统构建详解

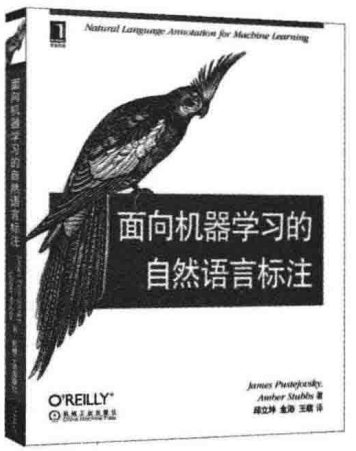

PaddlePaddle
深度学习实战

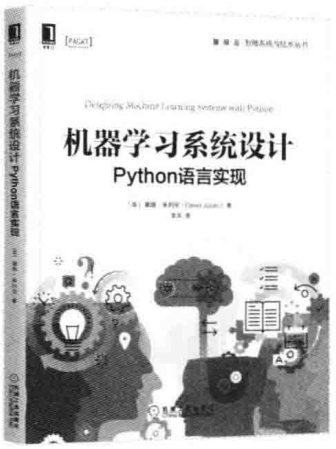

Neural Network Methods for Natural Language Processing
基于深度学习的
自然语言处理

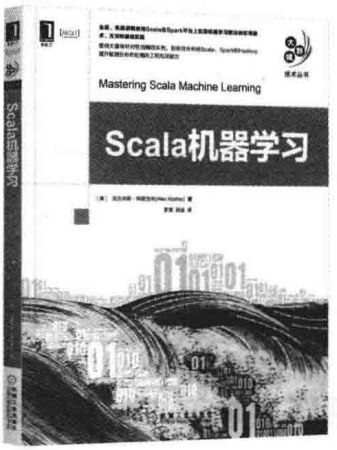

Python自然语言处理实战
核心技术与算法

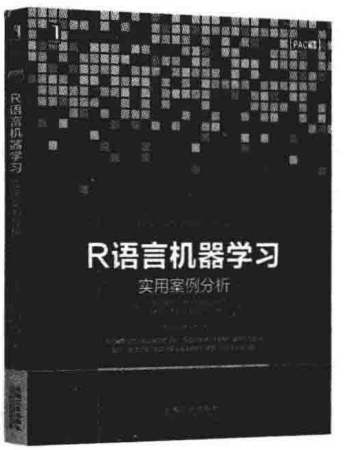

深度卷积网络
原理与实践

推荐阅读









Python机器学习

作者：Sebastian Raschka，Vahid Mirjalili ISBN：978-7-111-55880-4 定价：79.00元

机器学习：实用案例解析

作者：Drew Conway，John Myles White ISBN：978-7-111-41731-6 定价：69.00元

面向机器学习的自然语言标注

作者：James Pustejovsky，Amber Stubbs ISBN：978-7-111-55515-5 定价：79.00元

机器学习系统设计：Python语言实现

作者：David Julian ISBN：978-7-111-56945-9 定价：59.00元

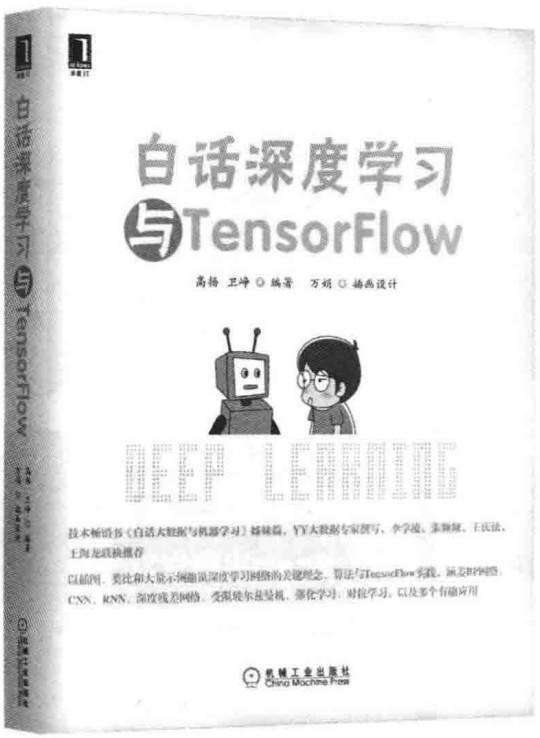
Scala机器学习

作者：Alexander Kozlov ISBN：978-7-111-57215-2 定价：59.00元

R语言机器学习：实用案例分析

作者：Dipanjan Sarkar，Raghav Bali ISBN：978-7-111-56590-1 定价：59.00元

推荐阅读



白话深度学习与TensorFlow

书号：978-7-111-57457-6 作者：高扬 定价：69.00元

《白话大数据与机器学习》姊妹篇，YY大数据专家多年实践沉淀之作，简单易懂，轻松入门

CONTENTS

目　录

CHAPTER 1

第 1 章

初识神经网络

本章将以在围棋比赛中战胜人类世界冠军的 AlphaGo 为例，介绍什么是神经网络，使你初步认识神经网络的强大。本章还介绍人工智能、机器学习、神经网络等热门概念的区别和联系。

1.1 例说神经网络

1.1.1 从围棋和 AlphaGo 说起

围棋（Go）是一种历史悠久的棋类游戏。如图 1-1 所示，围棋的棋盘是一个19×19 的网格。下棋时，双方选手轮流每人下一步棋。每下一步棋，棋手可以将一枚自己的棋子放在围棋棋盘的某一个网格上。直接或间接占有最多棋盘网格的选手获胜。长期以来，人们普遍认为在短期内计算机程序不可能在围棋上战胜人类职业选手，所以围棋也被视为人类智慧的瑰宝。谷歌（Google）旗下产品阿尔法狗（AlphaGo）于 2016 年 3 月战胜围棋世界冠军李世石，于 2017 年 5 月战胜排名世界第一的围棋世界冠军柯洁，引起了全社会的关注。

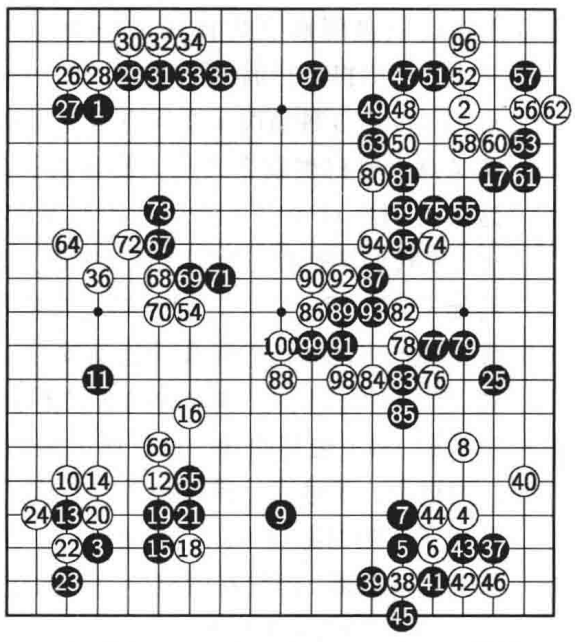


图 1-1 围棋的棋谱。其中实心圆圈表示黑棋棋子，空心圆圈表示白棋棋子。圆圈中的数字表示这个棋子是在第几步被放置在棋盘上的（图片改编自 D. Silver, et al. Mastering the game of Go without human knowledge, Nature, 2017）

在 AlphaGo 诞生之前，就出现过一些围棋程序。最早的棋类程序是根据预

定的固定策略来下棋的（如 1914 年 L. Quevedo 实现的自动下棋机）。在这类棋类程序的设计过程中，人类根据自己的对弈经验，抽象出一系列明确的对弈策略，并将这些策略用程序实现。这类程序的逻辑是这样的：首先判断当前棋局是不是满足第一种策略的适用条件。如果满足第一种策略的适用条件，则根据第一种策略决定棋子的放置位置；如果不满足，则判断当前棋局是不是满足第二种策略的适用条件。如果满足第二种策略的适用条件，则根据第二种策略决定棋子的放置位置；以此类推，直到棋子的方法被确定。具体流程如图 1-2 所示。这种方法能够利用人类已经知道的围棋对弈知识，达到一定的效果。

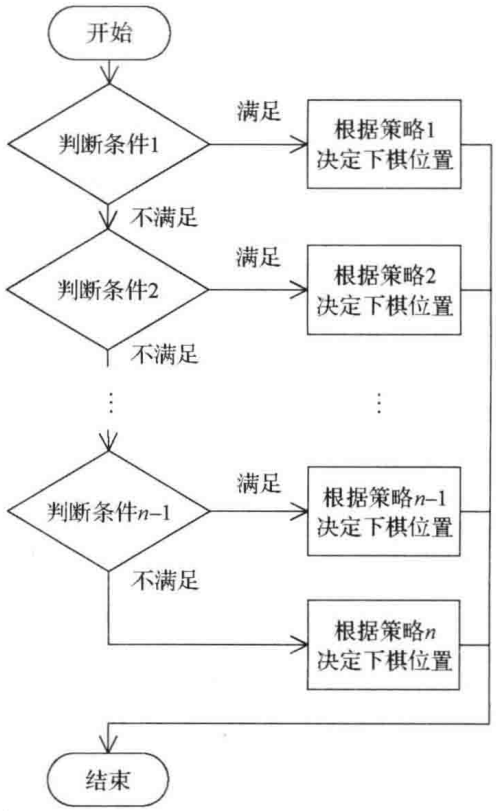


图 1-2　基于预定策略的围棋程序

但是，这些基于预定策略的围棋程序明显弱于人类职业棋手的围棋水平。基于预定策略的围棋程序无法战胜人类职业棋手，主要原因是围棋的最优决策非常复杂。围棋的复杂性具体体现为以下几点。

- 围棋的棋盘网格上共有 19×19 = 361 个交叉点。在围棋进行的某个时刻，每个交叉点都可能有 3 种情况（交叉点上有黑棋、交叉点上有白棋或交叉点上没有棋子）。这使得围棋对弈中可能遇到的状态非常多。宋代的沈括在《梦溪笔谈》中称："大约连书万字四十三个，即是局之大数"，即棋局的可能状态数是要通过连续写 43 个代表 10^4 的"万"字，即 10^{172}（这可以通过 $3^{19\times19}$ 计算得到）。如果扣除一些不满足围棋规则的状态，剩下的状态（约占总状态数的 1%）也有约 10^{170} 种。这比棋子数目有限的象棋、国际象棋等大得多。
- 在围棋中，很难通过棋子的数量和棋子的绝对位置判断形势的好坏。这与象棋这样的棋子有限的游戏不同。在象棋中，我们可以通过评估各类棋子的数目给出一个形势的简易判断。但是在围棋中，棋子的数量和形势的好坏往往没有特别明显的关系。这会对计算机进行形势判断造成障碍。
- 在围棋进行的过程中，双方交替下棋，每一步可以在某一个网格交叉点上放置一颗自己的棋子，而这 361 个网格交叉点都有可能是放置的位置。即使在某步确定棋子位置的时候只考虑未来的 8 步，也大概有 5×10^{20} 种可能组合。按照当今最快的超级计算机之一"天河二号"的理论峰值运算能力计算，也需要 4 个小时来穷举来这些组合。

- 围棋中的每一步都可能影响未来几百步，在决策某一步时只评估未来的若干步远远不够。而穷举未来所有可能的计算量过大，其必胜记忆法超过10^{600}。这个数字远远超过了可观测宇宙的原子总数（约10^{75}）。

围棋程序经过数十年的改进，已经越来越强。目前，AlphaGo 等围棋程序已经超过人类顶尖职业高手的水平。那么，为什么 AlphaGo 能够克服上述的种种困难，达到令人惊叹的围棋水平呢？这其中的关键在于，AlphaGo 不再使用人工抽象的诸多规则，而使用了人工神经网络来进行决策。人工神经网络是受生物大脑运作机理的启发而设计的一种数学模型。通过神经网络，可以做出非常复杂的决策。在下面的章节中，我们将从生物的神经系统出发，介绍人工神经网络的基本原理。

1.1.2　人的神经系统

人类的神经系统可以做非常复杂的决策，而这些决策离不开神经系统中广泛存在的基本单元——神经元（neutron）。神经元的简化模型如图 1-3 所示。每一个神经元都有一个胞体、一个或多个树突和一个轴突。对于某个神经元，其他神经元发来的信号通过树突传递到神经元内，然后在胞体中进行非线性的处理。处理过的信号再通过轴突传出。一个轴突可以有多个轴突末梢，将相同的信号传递给其他神经元。

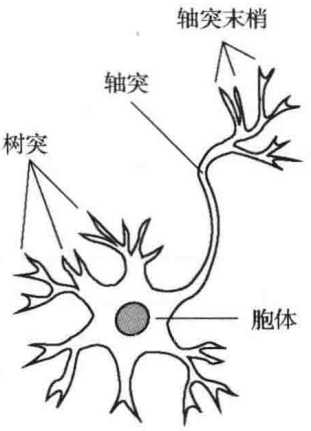


图 1-3　神经元的简化模型（图片改编自 http://en.wikipedia.org/wiki/Neuron）

初看之下，神经元模型只是将简单叠加过的输入信号进行了预设的非线性处理，其输入和输出关系并不复杂。那么，如此简单的模型是为什么有能力完成日常中多样而又复杂的决策呢？原因在于，人的神经系统中神经元的数量非常多，神经元间又通过树突和轴突以复杂的拓扑关系互相传递信号。这些神经元联合起来组成神经网络，能实现非常复杂的输入/输出关系。在后续的两节中，我们将用简化的定量模型来佐证这一点。

1.1.3　人工神经元

在 AlphaGo 等软件使用的人工神经网络中，计算机使用人工神经元来模拟人的神经元。人的神经元是这样的一种运算：假设人工神经元有 N 个输入 $(x[0],x[1],...,x[m-1])$，每个输入 $x[i]$（$0\leqslant i<m$）都是一个实数，那么人工神经元的输出 y 可以由 $m+1$ 个权值 $(w[0],w[1],w[2],...,w[m-1],w[m])$ 和 1 个非线性函数 f 确定，确定的方法为

$$y=f\left(w[0]x[0]+w[1]x[1]+w[2]x[2]+\cdots+w[m-1]x[m-1]+w[m]\right)$$

图 1-4 表示了这样的输入/输出关系。在这样的人工神经元中，将每个输入乘以权重，就相当于人的神经元中树突的功能；将各输入加起来再做非线性变换 f，就相当于人的神经元中胞体的功能；将非线性变换的结果输出出去，就相当于轴突的功能。

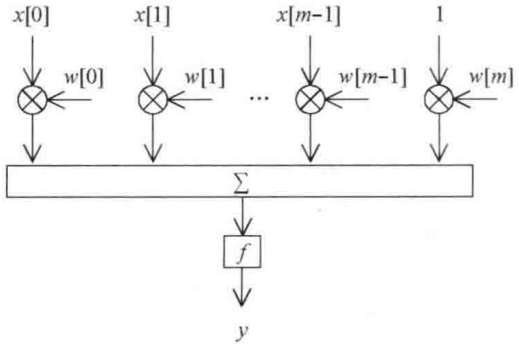


图 1-4　人工神经元的运算

下面来看一个人工神经元的例子。某个神经元有 $N=2$ 个输入，它的权值分别为 $w[0]=1$、$w[1]=1$、$w[2]=-1$，并且非线性函数 f 为函数输入值和 0 的最大值 $f(\cdot)=\max(\cdot,0)$。这个人工神经元输入和输出之间的关系是

$$y=\max(x[0]+x[1]-1,0)$$

值得一提的是，这个神经元可以完成逻辑运算中的“与”（and）运算。与运算是一种二元运算，它的输入和输出都是取自二元集合 $\{0,1\}$ 中的元素。当两个输入 $(x[0],x[1])$ 满足 $x[0]=x[1]=1$ 时，与运算的输出为 1；在其他情况下，与运算的输出为 0。我们可以列出所有可能的输入，通过逐一验证所有可能的输出来证明这个神经元实现了与运算（见表 1-1）。

表 1-1　与运算中输入和输出之间的关系

输入 $x[0]$	输入 $x[1]$	输出 y
0	0	0
0	1	0
1	0	0
1	1	1

实际上，在非线性函数 f 固定的情况下，选择不同的权重，神经元可以完成不同的运算。例如，在刚才的例子中，即使保持函数 $f(\cdot)=\max(\cdot,0)$ 和输入范围为 $x[0],x[1]\in\{0,1\}$ 不变，只要将权重改为 $w[0]=2$、$w[1]=-1$、$w[2]=-1$，神经元就变成下面的函数：

$$y=\max(2x[0]-x[1]-1,0)=\begin{cases}1, & x[0]=1\text{且}x[1]=0\\ 0, & \text{其他}\end{cases}$$

再进一步，如果输入范围不仅限于 $\{0,1\}$，这个神经元采用不同的权重后，还可以表示其他很多函数，例如 $\max(x[0]-4.2x[1]+\sqrt{3},0)$ 等。

但是，仅仅通过改变权重，并不能实现所有的运算关系。一个不能实现的例子是“或”（or）运算。或运算是一个定义在 $\{0,1\}$ 上的二元运算，它在 $x[0]=x[1]=0$ 的情况下输出 0，在其他情况下输出 1，即

$$\text{or}(x[0],x[1])=\begin{cases}0, & x[0]=x[1]=0\\ 1, & \text{其他}\end{cases}$$

可以证明，不存在权重 $(w[0],w[1],w[2])$，使得神经元的输入和输出关系为或运算。用反证法证明：反设权重为 $(w[0],w[1],w[2])$ 时输入/输出关系为或运算，则权重应满足① $\max(w[2],0)=0$；② $\max(w[0]+w[2],0)=\max(w[1]+w[2],0)=\max(w[0]+w[1]+w[2],0)=1$。由②可得 $w[0]+w[2]=w[1]+w[2]=w[0]+w[1]+w[2]=1$，进而 $w[0]=w[1]=0$，$w[2]=1$，这与①矛盾。得证。

通过刚才的例子，我们了解到一个人工神经元可以表示多种输入/输出关系，但是不能表示所有可能的输入/输出关系，而实际的问题（比如围棋）远远比这样的形式复杂得多。这时候，就需要同时使用多个神经元，将它们连成网络。

1.1.4　人工神经网络

人工神经网络是由多个神经元组成的网络。在神经网络中，某些神经元的输出会作为另外一些神经元的输入。

我们来看一个例子。图 1-5 是一个由两个神经元组成的神经网络。第 1 个神经元就是上节提到的与运算神经元，它可以将输入 $(x[0],x[1])$ 转化为输出 $\max(x[0]+x[1]-1,0)$。第 2 个神经元有 3 个输入，前 2 个输入 $x[0]$、$x[1]$ 直接取自第 1 个神经元的输入，第 3 个输入 $x[2]$ 是第 1 个神经元的输出。第 2 个神经元可以将输入 $(x[0],x[1],x[2])$ 转化为输出 $\max(x[0]+x[1]-x[2],0)$。这两个神经元组成的神经网络可以将输入 $(x[0],x[1])$ 转化为输出

$$\max(x[0]+x[1]-\max(x[0]+x[1]-1,0),0)$$

仿照之前对与运算的分析，我们可以列表验证，当输入 $x[0],x[1]\in\{0,1\}$ 时，输入和输出如表 1-2 所示。这就是 1.1.3 节提到的或运算。在该节中，我们知道了单个神经元不可能实现或运算，但是现在，我们用两个权重不同的神经元搭成的神经网络实现了或运算。

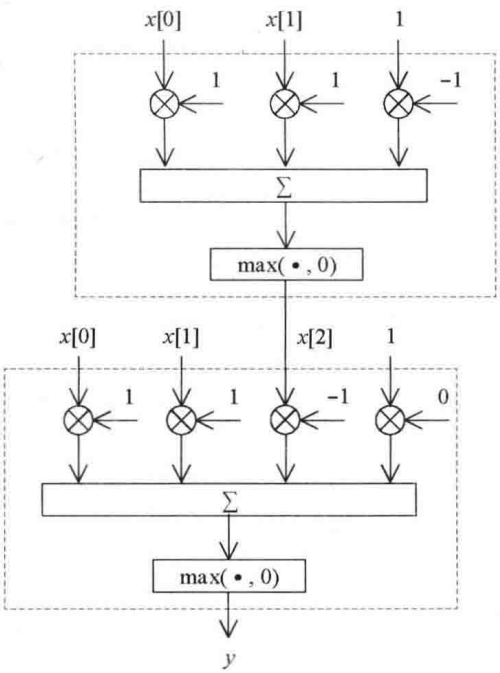


图 1-5　两个神经元组成的神经网络

表 1-2　或运算中输入和输出之间的关系

输入 $x[0]$	输入 $x[1]$	输出 y
0	0	0
0	1	1
1	0	1
1	1	1

上面的例子告诉我们：多个神经元可以实现单个神经元不能完成的运算。这已经暗示了