Broadview
www.broadview.com.cn

以TensorFlow为工具，助你成为神经网络高手，深度学习专家！

Python
与神经网络实战

何宇健 编著

很丰富：包括10种算法，13个常用的真实样例
很实战：全书有25个完整实验，9539行代码
很扎实：通过822个公式，让你理解神经网络的来龙去脉
很创新：笔者原创的软剪枝技术，能应用于各种神经网络模型

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Python
与神经网络实战

何宇健 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

人工智能已成发展趋势，而深度学习则是其中最有用的工具之一。虽然科技发展速度迅猛，现在实用技术更新换代的频率已经迅速到以周来计算，但是其背后最为基础的知识却是共通的。本书较为全面地介绍了神经网络的诸多基础与进阶的技术，同时还介绍了如何利用神经网络来解决真实世界中的现实任务。本书各章的内容不仅包括经典的传统机器学习算法与神经网络的方方面面，还对它们进行了对比与创新。如果能够掌握本书所讲述的知识，相信即使具体的技术更新得再快，读者也能根据本书所介绍的知识来快速理解、上手与改进它们。

本书兼顾了理论与实践，不仅从公式上推导出神经网络的各种性质，也从实验上对它们进行了验证，比较适合初学者进行学习。同时，本书所给出的框架更能直接、简单、快速地应用在实际任务中，适合相关从业人员使用。

图书在版编目（CIP）数据

Python 与神经网络实战/何宇健编著. —北京：电子工业出版社，2018.7
ISBN 978-7-121-34238-7

Ⅰ. ①P… Ⅱ. ①何… Ⅲ. ①人工神经网络－软件工具－程序设计 Ⅳ. ①TP183

中国版本图书馆 CIP 数据核字（2018）第 106127 号

策划编辑：张月萍
责任编辑：刘　舫
印　　刷：北京京师印务有限公司
装　　订：北京京师印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　　印张：25　　字数：640 千字
版　　次：2018 年 7 月第 1 版
印　　次：2018 年 7 月第 1 次印刷
印　　数：3500 册　　定价：79.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前　言

我在写完前一本书——《Python 与机器学习实战》之后，承蒙出版社青睐，被寄予了在某个领域深入剖析并再写一本书的希望。而当时（2017 年年中）恰好是一个非常特殊的时间点——那正是人工智能概念席卷全球，成为当之无愧的“引爆点”的前夕。我在上一本书的前言里曾经说过，自从 AlphaGo 在 2016 年 3 月战胜人类围棋顶尖高手李世石后，“人工智能”“深度学习”这一类词汇就进入了大众的视野；而作为更加宽泛的一个概念——“机器学习”则多少顺势成为从学术界到工业界都相当火热的话题，这也正是我上一本书的主题为机器学习的重要原因之一。而在 2017 年年中时，由于我从方方面面的资料与新闻中都隐隐约约地感受到了深度学习的巨大潜力，所以就和出版社定好了这本书的主题——神经网络。神经网络本身是一个非常宽泛的概念，它既能代指最基础的“全连接神经网络（DNN）”（需要指出的是，DNN 原本泛指 Deep Neural Network，即泛指深层神经网络，不过简洁起见，在本书中我们统一认定它特指全连接神经网络），也能代指当今在各种领域大放异彩的“卷积神经网络（CNN）”和“循环神经网络（RNN）”。本书将主要叙述的是“全连接神经网络（DNN）”及其变体，且其中涉及的技术也能应用在“卷积神经网络（CNN）”和“循环神经网络（RNN）”中。也正因此，在本书的正文中，我们会统一认定“神经网络”代指的是 DNN；但是大家需要知道的是，在前言这里我们会用“神经网络”代指 DNN、CNN 与 RNN 的集合，而在本书以外的场合，虽然说起“神经网络”大家一般都会认为说的是 DNN，但也有可能不单单代指 DNN。

那么，为什么我选择了“神经网络”这个主题呢？简单来说，神经网络是深度学习的“前身”或说“基础”，因为深度学习往简单里说的话，其实就是“比较深的神经网络”。此外，对于本书将主要叙述的 DNN 来说，它和我上一本书中介绍的诸多传统机器学习算法也有千丝万缕的关系；通过两者之间的相互对比，想必大家能对它们都有更深刻的理解，同时也能打下更坚实的基础。

那么，为什么我选择了这种偏基础的主题而不是一些更具体的主题（比如说图像识别、自然语言处理、推荐系统、强化学习）呢？毕竟在有了各种深度学习工具之后，DNN 作为“老前辈”一般的存在，基本只能解决一些结构化数据的问题，而且解决得通常可能还没有传统的机

器学习模型好。这主要出于两个考虑：一是我做的研究本身普遍偏基础，落实到具体的应用时也只是基础方法的具体应用，而非针对具体的应用做的优化；另一方面则是我个人认为，深度学习一旦兴旺起来，各种技术的迭代速度必定是极快的。这是因为深度学习之所以没能获得发展，相当大的原因是受了硬件设备与数据量的制约，一旦这两个制约被打破，那么至少在我看来，它就会以一个月甚至一周的迭代周期来“优胜劣汰”。在这种情况下，即使我花了比较大的力气去介绍我写书时最流行、最有效的技术，等到读者们拿到书时，这些技术很有可能已经没那么流行甚至过时了。总而言之，由于深度学习在我期望中的发展速度过快，所以我没敢向具体的技术下手。

有意思的是，时至今日（2018 年年初），深度学习果然如我所料，几乎可以算是进入了“一周一个新技术”的发展阶段，各大公司的新产品日新月异、层出不穷，这让我为当初选择了基础性的主题而感到庆幸。因为无论具体领域的具体应用技术如何发展，它们背后最本质、最基础的知识都是不会变的。如果能掌握这些基础性的知识，那么在接受各种蜂拥而至的新技术时，想必也会轻松不少吧。

当然，由于我们的目的还是想让读者能够学以致用，所以虽然本书的主题偏基础，但是我们也会花篇幅来介绍如何将这些基础性的知识进行具体的应用。事实上，我们只会在前 5 章介绍理论上的知识，而第 6 章和第 7 章，则都是在介绍如何编写能够应用于现实任务中的框架。此外，本书的代码实现都是偏工程化的，所以大部分核心代码可能会比较长。这是因为我们想要传达的是一种大规模编程下的优良习惯，比如可拓展性、可迁移性、用户友好性等，所以有些地方的实现相比起纯算法实现而言可能会显得略微冗长。不过在我看来，在实际任务中，其实一般很少能够仅仅编写纯算法实现，大部分情况下都需要融合进其他内容，所以相信本书的实现方式能够帮助大家适应今后现实中的情景。

此外需要指出的是，囿于篇幅，本书无法将所有代码悉数放出（事实上这样做的意义也不是很大），所以我们会适当地略去一些相对枯燥且和相应算法的核心思想关系不大、又或是测试代码的实现。对于这些代码以及本书展示的所有代码，我都把它们放在了 GitHub 上，大家可以参见 https://github.com/carefree0910/MachineLearning/tree/Book/_Dist/NeuralNetworks 这个目录中的相应代码。我个人的建议是，在阅读本书之前先把这个链接里面的内容都下载下来作为参照，并在需要的时候通过 Ctrl+F 组合键进行相应的检索。同时，在本书写完之后，由于我会把收笔时的代码保留在 Book 这个 Branch 中，并把后续的更新统一放在 master Branch 中，所以也可以参见 https://github.com/carefree0910/MachineLearning/tree/master/_Dist/NeuralNetworks 这个根目录中的代码以获取无法反映在本书中的、最新的更新。

最后想要说的是，与我写的上一本书类似，虽然本书会尽量避免罗列枯燥的数学公式，但是一些比较重要的公式与证明还是不可或缺的。不过考虑到数学基础因人而异，我把一些额外的、类似于“附加证明”的章节打上星号（*）。对于之前学过机器学习、基础比较扎实的读者，阅读这些带星号的章节是比较有益的，因为它们有助于更深刻地理解一些知识背后的理论；而对于零基础或仅想通过本书入门的读者，这些章节可能稍显困难，所以直接把它们跳过也是不错的选择，因为跳过它们不看并不会对理解主要内容造成很大的影响。

本书特点

- **注重基础**：本书不仅介绍了神经网络的基础知识（比如前向传导算法、反向传播算法、Dropout、Batch Normalization 等），还涵盖了传统的、经典的、基础的机器学习算法（朴素贝叶斯、决策树、支持向量机）的大意以及这些算法与神经网络在本质上的联系。
- **注重创新**：本书将会介绍许多属于新颖的、有效的技术。比如第 4 章中的转换算法，目前市面上的图书中基本没有类似的、面面俱到的介绍；而第 5 章介绍的神经网络中的软剪枝技术，则更是笔者个人的研究，可以说是“只此一家”。
- **注重应用**：本书最后两章将会着重介绍如何搭建在现实任务中确实能拿来应用的神经网络框架，该框架不仅兼顾了可扩展性，也兼顾了用户友好性，能在各个领域中发挥作用。
- **注重基础**：本书中的所有代码实现沿袭了上一本书的传统，基本都是“从零开始”。也正因此，所涉及的核心实现大多仅仅基于一些基础的第三方库（如 numpy、TensorFlow）而没有依赖更高级的第三方库，这使得我们能够通过代码实现来辅助理解算法细节。

本书的内容安排

第 1 章　绪论

本章介绍了一些基本概念与基础术语，这些内容虽然可能略显乏味，但却是跨入业界必须掌握的知识。本章的篇幅很短，内容相对浓缩，对于零基础的读者可能会需要花一些时间去消化，对于有基础的读者则是比较好的知识唤醒。

第 2 章　经典传统机器学习算法简介

本章将会精要介绍 5 个经典的传统机器学习算法——朴素贝叶斯、决策树、感知机、支持向量机和 Logistic 回归，介绍的方式偏向于直观叙述，旨在让大家更好地从原理而不是从细节上去理解这些算法。虽然本书的主题是（全连接）神经网络，但是了解这些传统的机器学习算法是有必要的，它们不仅能给我们解决实际问题的思维，也能佐证神经网络的强大。

第 3 章　神经网络入门

本章是神经网络的入门章节，旨在全面且较为深刻地叙述神经网络的各项技术要点，包括神经网络的结构、激活函数、损失函数、前向传导算法、反向传播算法、梯度下降法的各式变体、搭建 TensorFlow 模型基本框架的诸多注意事项等。如果能够将本章的所有内容都深刻理解的话，相信不仅能对神经网络有比较扎实的认知，也能对 TensorFlow 这个深度学习框架有更好的认知，这对今后的各种实际应用是大有裨益的。

第 4 章　从传统算法走向神经网络

本章通过将神经网络与第 2 章所介绍的各式传统机器学习算法做对比，以此来帮助大家加

深对这些传统机器学习算法和神经网络的理解。我们将会看到，即使是最朴素的神经网络，就已经能够自然地表达出感知机、支持向量机和 Logistic 回归这三种模型，而且只要通过定制它的权值矩阵与偏置量，甚至还能够表达出几乎任意的朴素贝叶斯模型与决策树模型。此外，这种表达还有着一定的现实意义，并不仅仅是理论上的技巧。

第 5 章　神经网络进阶

本章是神经网络的进阶章节，旨在介绍一些能够提升朴素神经网络性能的技术，包括 Dropout、Batch Normalization、Wide and Deep、Deep Neural Decision Forest、Dynamic Network Surgery 以及软剪枝技术等。这些技术并不是一个个独立的技术，在经过适当整合之后，从理论和实验效果来看都会拥有许多有趣且有用的性质。

第 6 章　半自动化机器学习框架

本章旨在介绍如何将前 5 章介绍过的技术应用到实际问题中。虽然在介绍技术理论时我们可以做出许多很强的假设，比如数据都是连续型数据、各个特征的取值都很合理，以及训练步数可以控制在最优等，但对于真实世界中的数据集而言，我们往往需要做很多预处理工作之后，才能够应用诸多（神经网络的）技术；而且由梯度下降法的特性可知，训练步数是一个理论上不可知的、但是又非常重要的超参数，究竟何时停止训练甚至是模型表现优秀与否的最关键的环节。本章会较为全面地给出针对这些问题的一套解决方案，这套解决方案能够直接应用于绝大多数非结构化数据上，实现了机器学习的“半自动化”。

第 7 章　工程化机器学习框架

前 6 章的内容已经使我们能够用同一个模型端到端地解决许多问题，但是在现实任务中，我们不仅常常需要进行多次重复实验来更好地验证模型，而且还常常需要同时对很多模型做实验并选出其中最好的模型，这就要求我们在前 6 章的基础上进一步自动化这些功能。本章将会介绍如何利用前 6 章实现的框架来进行多次实验与参数搜索，前者可以更合理地评估特定模型在特定数据集上的性能，而后者则能够挑选出最适合特定数据集的模型。

适合阅读本书的读者

- 想要转行 AI、投身 AI 的程序员
- 想要知道 TensorFlow 编程技巧的程序员
- 想要打好基础并入门深度学习的学生、老师、在职员工等
- 想要使用神经网络来解决（非结构化数据分类预测的）实际问题的从业者

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **下载资源**：本书中部分图片的彩色版本可在下载资源处下载。
- **提交勘误**：您对书中内容的修改意见可在提交勘误处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方读者评论处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/34238

目 录

第 1 章
绪论

第一次工业革命约兴起于 17 世纪 60 年代，一直持续到 18 世纪 30 年代至 18 世纪 40 年代，人类在这段时间里使用机器取代人力、兽力，以大规模的工厂生产取代个体手工生产。第二次工业革命则紧跟着第一次工业革命之后，以电力的大规模应用为代表，新技术的重要性可谓展现得淋漓尽致。第三次工业革命常被称为科技革命、信息技术革命、数字化革命，计算机和电子数据正是在这期间普及、推广的，使得各行各业都发生了从机械和模拟电路到数字电路的变革，至今仍未结束。如果稍微总结一下的话，大抵可以这样说：第一次工业革命由机器驱动，第二次工业革命由电力驱动，第三次工业革命则由数字化与新科技驱动。

而第四次工业革命，不少人认为将会是集前三次工业革命之大成。具体一点说，不少人认为它将由人工智能驱动。

人工智能（Artificial Intelligence，常简称为 AI）在近期已经有人尽皆知的趋势，不少行业的领军企业都纷纷提出了“AI First”的策略。人工智能的魅力毋庸置疑，从小说到电影、从纪实到科幻，人工智能永远是一个热门的题材。

这也使得许多人对人工智能心生向往，但同时又可能不知如何下手。本书的目的就是为广大读者提供其中一个入手的方向——使用 Python 来搭建神经网络模型。须知神经网络乃深度学习的根基，而深度学习则是人工智能中最有名、有效的工具之一。所以在学习了神经网络相应的理论后，相信大家就能以此为基础进而开发出人工智能的应用，从而为这个 AI 社会贡献出自己的一份力量。作为本书的第 1 章，我们打算先谈谈入门 Python 神经网络所需的一些比较宽泛的知识。我们会在第 1 节进行与机器学习相关的一些说明，在第 2 节、第 3 节介绍一下 Python 和一些 Python 中常用的第三方库和小工具。

具体而言，本章主要涉及的知识点有：

- 机器学习的定义及术语

- Python 在机器学习领域的优异性
- scikit-learn 和 TensorFlow 的简介与安装
- 数据的简易预处理与进度条、计时器的使用

1.1 机器学习简介

由于人工智能在最近才火起来（不如说在以前，人们通常并不以做人工智能为荣，反而羞于说自己是做人工智能的），所以关于它的一些概念存在着一定的不统一性。本节打算对这些概念进行梳理，以方便本书后文的讨论。

首先大家可能想问：机器学习（Machine Learning，常简称为 ML）、神经网络（Neural Network，常简称为 NN）、深度学习（Deep Learning，常简称为 DL）和人工智能（AI）之间的关系到底是什么？这个问题的答案可以归结为如下三点：

- 神经网络是深度学习的“根基”。
- 深度学习是机器学习的一个分支。
- 机器学习是人工智能领域的一种技术。

换句话说，我们可以使用机器学习技术来做人工智能，而深度学习则是众多机器学习技术中效果最好的技术之一（围棋界的 Master 是其中最具代表性的存在；当然，Master 所用的技术并不全是深度学习，但核心技术大都是深度学习）。深度学习比较严谨的定义可以在维基百科（Wikipedia）上找到——它是试图使用包含复杂结构或由多重非线性变换构成的多个处理层对数据进行高层抽象的、基于对数据进行表征学习的一种算法。不过通俗地说，它其实就是各种花式神经网络的集合。事实上维基百科中也说了，“深度学习”已成为类似术语，或者说是神经网络的品牌重塑。

神经网络单单凭借其名字及效果，其实就已经使不少人或多或少地对它产生了兴趣与敬畏。其看上去很复杂的数学公式，更是使得许多初学者望而却步。但是如果我们耐下心来一步步踏实地走下来，就会发现神经网络其实并没有想象中那么可怕。为了更好地理解神经网络，对传统机器学习算法进行介绍（第 2 章）并用神经网络与它们进行对比（第 4 章）是有必要的，为此我们需要先知道机器学习的基本思想和其中一些常见的术语。

1.1.1 什么是机器学习

机器学习听上去非常玄妙，但它本质上没有多么复杂。要理解这个词语，我们一般会把它拆成“机器”与“学习”这两部分来理解。其中，“机器”是拥有“没有生命”“能够进行运算”“服从人类指令”等基本属性的物件，比如，我们的电脑或服务器等硬件设备。而所谓的“学习”，和我们人类所说的学习有相近之处，但却又不完全一致。人类在学习时，常常是通过五感（视觉、听觉、触觉、嗅觉和味觉）与外界进行交互来接收信息，然后结合过往的经验来理解、吸收。而且我们在年龄尚小、自我监督能力不够强时，常常还需要一定的激励才能很好地将学

习继续下去（比如考试拿到一定的分数就能获得想要的礼物之类的）。而对于机器来说，它与外界的交互可以说只有一个媒介——那就是数据。同时，虽然我们有时需要设计一种包含激励的算法（比如说强化学习），但机器不会因为不具有激励就不进行学习。只要我们将学习的指令下达给机器，机器就会不知疲倦地执行相应的步骤。

所以对于机器学习而言，核心的内容其实只有两个：算法（赋予“学习能力”的指令）和数据（“学习”的素材）。在传统的机器学习模型中，算法往往是比较关键的部分；数据虽然也至关重要，但我们往往会着重要求它的“质量”而非“数量”。不过对于本书将主要讨论的算法——神经网络而言，数据的“数量”往往是最为重要的一环（当然，质量也很重要），这也是为什么不少人称之为“数据驱动的模型”。具体的细节我们会在第3章第6节的最后给出一个解释，此处就暂时按下不表。

值得一提的是，在笔者写下这段话时，刚出了一个很厉害的东西，叫作“AlphaGo Zero”，它是没有利用任何数据、从头学习的围棋 AI，但其最终性能却比以往所有利用了数据的围棋AI都要强。然而笔者认为，AlphaGo Zero的成功虽然确实说明了数据在特定领域并不是必需的，但却并不能说明数据的重要性降低了。事实上我们在讨论围棋时，常常以其“规则简单却博大精深”来赞扬它，而这个“规则简单”很可能恰恰是AlphaGo Zero无须人工数据的关键。引用周志华教授的看法之一就是：“别幻想什么无监督学习，监督信息来自精准规则，非常强的监督信息”。当我们面对一个实际问题时，规则常常是模糊且充满不确定性的（比如股票交易问题），此时大量的数据是不可或缺的。

1.1.2 机器学习常用术语

正如物理学中有势能、动能、惯性、功率等专有名词，化学中有元素周期表作为根基，数学更是有完备的符号体系来进行表述，机器学习领域也有一套术语方便同行间的沟通、交流与讨论。这些术语乍一听可能会觉得非常“高大上”——“样本空间”“交叉验证集”“假设空间”“泛化能力”……然而我们需要知道的是，这些术语实际上都是可以用非常直观的方式来理解的。虽然它们背后的数学背景可能会五花八门（比如概率论、数理统计、实变与泛函等），但是即使我们撇开过于细节的知识，也不会对实际应用乃至研发和创新带来很大的影响（当然，如果真的想做到顶尖的话，相应的数学背景还是要补上的，不过那就超出本书的讨论范围了）。

我们在1.1.1节刚说过，神经网络常常被称为“数据驱动的模型”，下面就来看看和模型、数据相关的一些术语和相应的默认符号。它们都是非常基础且重要的，是需要被牢牢记住并掌握的。

- 模型（Model），它是某个机器学习算法所导出的、能够完成训练与预测等任务的物件，我们常常用 G 来指代它。我们的任务就是使用数据来训练 G，并让 G 能够为我们解决一系列特定的事情。而模型本身又可以衍生出两个相关的概念：
    - 参数（Parameter），它是决定模型行为的东西，我们的训练目的就是把参数训练到一个能够使模型表现最好的值。一般而言，参数会用θ来代指，从而我们的模型

G 常常可以写成：

$$G(\boldsymbol{x}) = G(\boldsymbol{x}|\theta)$$

从直观上来说，$G(\boldsymbol{x}|\theta)$表示的就是“G 在参数θ下的行为”。

- 超参数（Hyper Parameter），它是决定模型结构或训练行为的东西。与参数不同的是，超参数一般是不能被训练而只能被“选择”的。换句话说，我们可以通过选择不同的超参数来搭建出同一套算法下结构和训练方式不一的模型，但我们在训练的过程中，超参数常常是保持不变的。超参数可以用$\tilde{\theta}$来代指，不过需要单独指明超参数的场景很少，我们一般会将超参数和参数视为一个整体，并把这个整体用Θ（大写的θ）来表示，从而我们的模型 G 常常可以写成：

$$G(\boldsymbol{x}) = G(\boldsymbol{x}|\Theta)$$

从直观上来说，$G(\boldsymbol{x}|\Theta)$表示的就是“G 在参数和超参数的集合Θ下的行为”。如果没有进行特殊说明，后文中我们会直接称Θ为参数，而不一一仔细区分参数与超参数。

- 空间（Space），它常作为后缀出现，如果没有特别指出，那么它表示的就是前面的词语“可能存在的取值”。比如，我们说“模型空间”时，指的就是“所有可能的模型”；当我们说“参数空间”时，指的就是“所有可能的参数”；当我们说“样本空间”（常用$\mathcal{D}$代指）时，指的就是“所有可能的样本”（样本的定义见下）。
- 数据集（Data Set），顾名思义，它是数据的集合。
- 样本（Sample），它是数据集中的每一条单独的数据。如果没有特殊说明，我们会默认数据集中有 N 个样本，并用符号：

$$D = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_N, y_N)\}$$

来表示数据集。不难看出，其中的

$$(\boldsymbol{x}_i, y_i), \qquad i = 1,2, \dots, N$$

就是一个样本，它包含以下两个部分。

- 特征向量（Feature Vector），它是样本中的$\boldsymbol{x}_i$，常常是模型输入的来源。我们既可以把特征向量直接输入模型（$\boldsymbol{x}_i \to G$），也可以先对特征向量中的各个“特征（Feature）”进行相应“预处理（Preprocess）”，然后把预处理后的特征向量作为模型的输入（$\boldsymbol{x}_i \xrightarrow{Preprocess} \boldsymbol{x}_i^* \to G$）。对特征向量进行预处理这一步通常被称为“数据预处理（Preprocessing）”。此外，一般而言，我们会假设特征向量是 n 维的列向量，即一个特征向量中会有 n 个特征（Feature）：

$$\boldsymbol{x}_i = \left(x_i^{(1)}, \dots, x_i^{(n)}\right)^{\mathrm{T}}$$

而对于特征自身而言，则大体上可以分为两种：离散型特征和连续型特征。其中，离散型特征的取值是离散的，它一般只有有限个取值，比如，“是否已经工作”这个特征的取值就只有“是”和“否”这两个取值。而连续型特征的取值就是连