第 1 章

◂ 计算机视觉与深度学习 ▸

当作者还是一个懵懂的小孩的时候，电视台播放的一部美国动画片《变形金刚》（如图 1-1 所示）激起了作者对机器人的浓厚兴趣。一句“汽车人，变形，出发！”不光是孩子，甚至于连陪同观看的大人们也会被那些懂幽默、会调侃，充满着正义、勇敢、智慧、热情、所向无敌的变形金刚人物所吸引。

图 1-1　变形金刚——霸天虎

长久以来，机器人和人工智能主题的电影、电视剧和动画片一直备受观众所喜爱，人类用对未来的无尽的想象力和炫目的特技效果构筑了一个又一个精彩的未来世界，令人陶醉。但是回归到现实，计算机科学家和工程技术人员的创造和设计能力却远远赶不上电影编剧们的想象力。动画片终究是动画片，变形金刚也不存在于这个现实世界中，要研发出一个像霸天虎一样能思考、看得到周围景物、听得懂人类语言并和人类进行流利对话的机器人，这条路还很漫长。

1.1　计算机视觉与深度学习的关系

长期以来，让计算机能看会听可以说是计算机科学家孜孜不倦的追求目标，这个目标中最基础的就是让计算机能够看见这个世界，赋予计算机一双和人类一样的眼睛，让它们也能看懂这个美好的世界，这也是激励作者及所有为之奋斗的计算机工作者的主要动力。虽然目前计算

机并不能达到动画片中变形金刚的十分之一的能力，但是技术进步是不会停止的。

1.1.1 人类视觉神经的启迪

20 世纪 50 年代，Torsten Wiesel 和 David Hubel 两位神经科学家在猫和猴子身上做了一项非常有名的关于动物视觉的实验（如图 1-2 所示）。

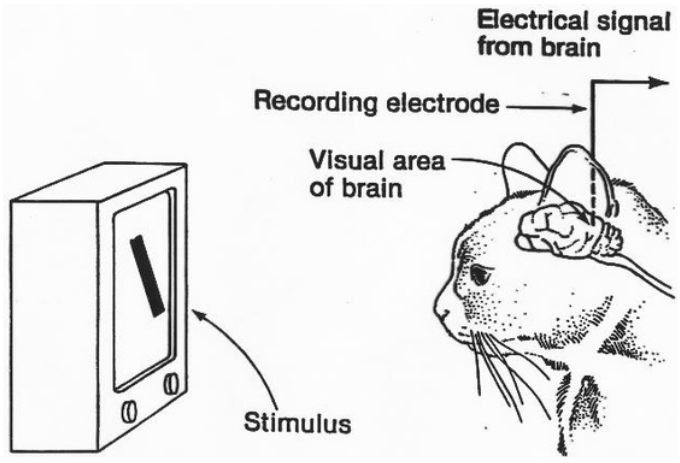

图 1-2　脑部连入电极的猫

实验中猫的头部被固定，视野只能落在一个显示屏区域，显示屏上会不时出现小光点或者划过小光条，而一条导线直接连入猫的脑部区域中的视觉皮层位置。

Torsten Wiesel 和 David Hubel 通过实验发现，当有小光点出现在屏幕上时，猫视觉皮层的一部分区域被激活，随着不同光点的闪现，不同脑部视觉神经区域被激活。而当屏幕上出现光条时，则有更多的神经细胞被激活，区域也更为丰富。他们的研究还发现，有些脑部视觉细胞对于明暗对比非常敏感，对视野中光亮的方向（不是位置）和光亮移动的方向具有选择性。

自从 Torsten Wiesel 和 David Hubel 做了这个有名的脑部视觉神经实验之后，视觉神经科学（如图 1-3 所示）正式被人们所确立。截至目前，关于视觉神经的几个广为接受的观点是：

- 大脑对视觉信息的处理是分层级的，低级脑区可能处理边度、边缘的信息，高级脑区处理更抽象的信息，比如人脸、房子、物体的运动之类的。信息被一层一层地提取出来往上传递进行处理。
- 大脑对视觉信息的处理也是并行的，不同的脑区提取出不同的信息，干不同的活，有的负责处理这个物体是什么，有的负责处理这个物体是怎么动的。
- 脑区之间存在着广泛的联系，同时高级皮层对低级皮层也有很多的反馈投射。
- 信息的处理普遍受到自上而下和自下而上的调控。也就是说，大脑可能选择性地对某些空间或者某些特征进行更加精细的加工。

图 1-3　视觉神经科学

进一步的研究发现，当一个特定物体出现在视野的任意一个范围，某些脑部的视觉神经元会一直处于固定的活跃状态。从视觉神经科学的角度解释，就是人类的视觉辨识是从视网膜到脑皮层，神经系统从识别细微细小的特征演变为目标识别。对计算机来说，如果拥有这么一个“脑皮层”对信号进行转换，那么计算机仿照人类拥有视觉就会变为现实。

1.1.2　计算机视觉的难点与人工神经网络

尽管通过大量的研究，人类视觉的秘密正在逐渐被揭开，但是相同的想法和经验用于计算机上却并非易事。计算机识别往往有严格的限制和规格，即使同一张图片或者场景，一旦光线，甚至于观察角度发生变化，那么计算机的判别也会发生变化。对于计算机来说，识别两个独立的物体容易，但是在不同的场景下识别同一个物体则困难得多。

因此，计算机视觉的核心在于如何忽略同一个物体内部的差异而强化不同物体之间的分别（如图 1-4 所示），即同一个物体相似，而不同的物体之间有很大的差别。

图 1-4　计算机视觉

长期以来，对于解决计算机视觉识别问题，大量的研究人员投入了很多的精力，贡献了很多不同的算法和解决方案。经过不懈的努力和无数次尝试，最终计算机视觉研究人员发现，**使用人工神经网络用以解决计算机视觉问题是最好的解决办法**。

人工神经网络在 20 世纪 60 年代就产生萌芽，但是限于当时的计算机硬件资源，其理论只能停留在简单的模型之上，无法得到全面的发展和验证。

随着人们对人工神经网络的进一步研究，20 世纪 80 年代人工神经网络具有里程碑意义的理论基础“反向传播算法”的发明，将原本非常复杂的链式法则拆解为一个个独立的、只有前后关系的连接层，并按各自的权重分配错误更新。这种方法使得人工神经网络从繁重的、几乎不可能解决的样本计算中脱离出来，通过学习已有的数据统计规律，对未定位的事件做出预测。

随着研究的进一步深入，2006 年，多伦多大学的 Geoffrey Hinton 在深层神经网络的训练上取得了突破。他首次证明了使用更多隐层和更多神经元的人工神经网络具有更好的学习能力。其基本原理就是使用具有一定分布规律的数据，保证神经网络模型初始化，再使用监督数据在初始化好的网络上进行计算，使用反向传播对神经元进行优化调整。

1.1.3 应用深度学习解决计算机视觉问题

受前人研究的启发，“带有卷积结构的深度神经网络（CNN）”被大量应用于计算机视觉之中。这是一种仿照生物视觉的逐层分解算法，分配不同的层级对图像进行处理（如图 1-5 所示）。例如，第一层检测物体的边缘、角点、尖锐或不平滑的区域，这一层几乎不包含语义信息；第二层基于第一层检测的结果进行组合，检测不同物体的位置、纹路、形状等，并将这些组合传递给下一层。以此类推，使得计算机和生物一样拥有视觉能力、辨识能力和精度。

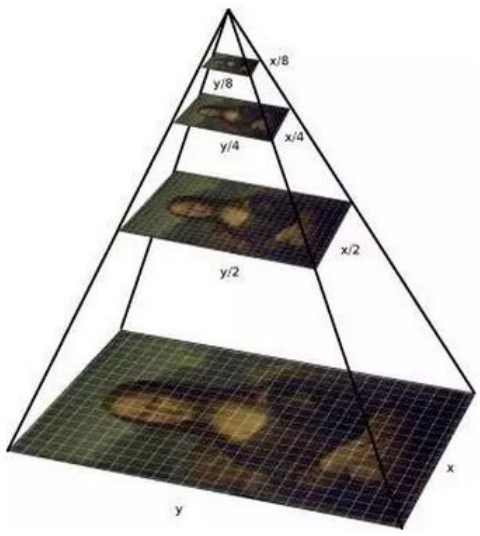

图 1-5　分层的视觉处理算法

因此 CNN，特别是其基本原理和算法被视为计算机视觉的首选解决方案，这就是深度学习的一个应用。除此之外，深度学习应用于计算机视觉上还有其他优点，主要表现如下：

- 深度学习算法的通用性很强，在传统算法里面，针对不同的物体需要定制不同的算法。相比来看，基于深度学习的算法更加通用，比如在传统 CNN 基础上发展起来的 faster RCNN，在人脸、行人、一般物体检测任务上都可以取得非常好的效果（如图 1-6 所示）。

- 深度学习获得的特征（feature）有很强的迁移能力。所谓特征迁移能力，指的是在 A 任务上学习到一些特征，在 B 任务上使用也可以获得非常好的效果。例如在 ImageNet（物体为主）上学习到的特征，在场景分类任务上也能取得非常好的效果。
- 工程开发、优化、维护成本低。深度学习计算主要是卷积和矩阵乘法，针对这种计算优化，所有深度学习算法都可以提升性能。

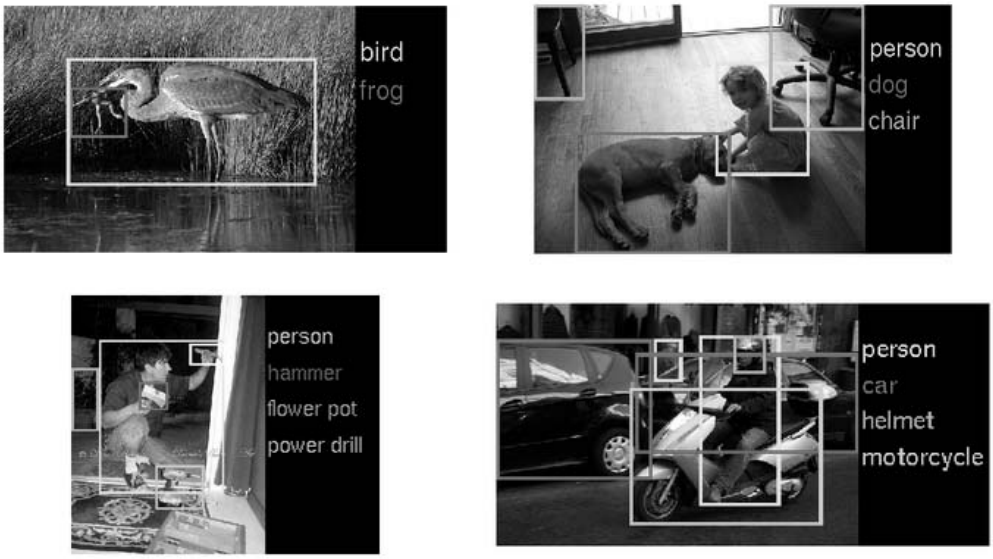

图 1-6　计算机视觉辨识图片

1.2 计算机视觉学习的基础与研究方向

计算机视觉是一个专门教计算机如何去“看”的学科，更进一步的解释就是使用机器替代生物眼睛来对目标进行识别，并在此基础上做出必要的图像处理，加工所需要的对象。

使用深度学习并不是一件简单的事，建立一项有真正识别能力的计算机视觉系统更不容易。从学科分类上来说，计算机视觉的理念在某些方面其实与其他学科有很大一部分的重叠，其中包括：人工智能、数字图像处理、机器学习、深度学习、模式识别、概率图模型、科学计算，以及一系列的数学计算等。这些领域急需相关研究人员学习其基础知识，理解并找出规律，从而揭示那些我们以前不曾注意过的细节。

1.2.1 学习计算机视觉结构图

对于相关的研究人员，可以把使用深度学习解决计算机视觉的问题归纳成一个结构关系图（如图 1-7 所示）。

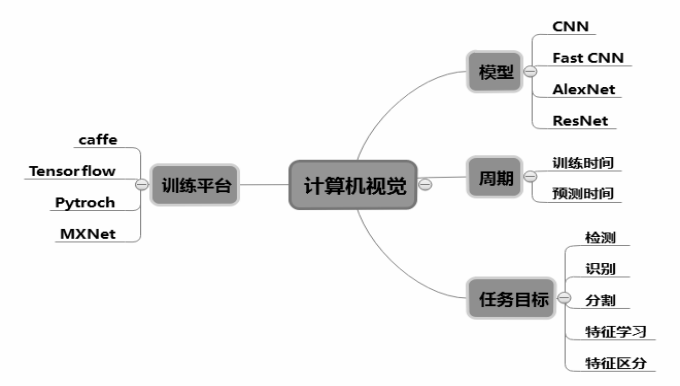

图 1-7　计算机视觉结构图

对于计算机视觉学习来说，选择一个好的训练平台是重中之重。因为对于绝大多数的学习者来说，平台的易用性以及便捷性往往决定着学习的成败。目前常用的是 TensorFlow、Caffe、PyTroch 等。

其次是模型的使用。自 2006 年深度学习的概念被确立以后，经过不断的探索与尝试，研究人员确立了模型设计是计算机视觉训练的核心内容，其中应用广泛使用的是 AlexNet、VGGNet、GoogleNet、ResNet 等。

除此之外，速度和周期也是需要考虑的一个非常重要的因素，如何使得训练速度更快，如何使用模型更快地对物体进行辨识，这是计算机视觉中非常重要的问题。

所有的模型设计和应用最核心的部分就是任务处理的对象，这里主要包括检测、识别、分割、特征点定位、序列学习 5 个大的任务，可以说任何计算机视觉的具体应用都是由这 5 个任务中的一个或者几个组合而成的。

1.2.2　计算机视觉的学习方式和未来趋势

“给计算机连上一个摄像头，让计算机描述它看到了什么。”这是计算机视觉作为一门学科被提出时就决定下来的目标，如今大量的研究人员为这个目标孜孜不倦地工作着。

拿出一张图片，上面是一只狗和一只猫，让一个人去辨识（如图 1-8 所示）。无论图片上的猫或者狗的形象与种类如何，人类总是能够精确地区分图片是猫还是狗。而把这种带有标注的图片送到神经网络模型中去学习，这种学习方式称为“监督学习”。

图 1-8　猫和狗

虽然目前来说，在监督学习的计算机视觉领域，深度学习取得了重大成果，但是相对于生物视觉学习和分辨方式的“半监督学习”和“无监督学习”，还有更多更重要的内容急需解决，比如视频里物体的运动、行为存在特定规律；在一张图片里，一个动物也是有特定的结构的，利用这些视频或图像中特定的结构可以把一个无监督的问题转化为一个有监督问题，然后利用有监督学习的方法来学习。这是计算机视觉的学习方式。

MIT 给机器“看电视剧”预测人类行为，MIT 的人工智能为视频配音，迪士尼研究院可以让 AI 直接识别视频里正在发生的事。除此之外，计算机视觉还可以应用在那些人类能力所限、感觉器官不能及的领域和单调乏味的工作上——在微笑瞬间自动按下快门，帮助汽车驾驶员泊车入位，捕捉身体的姿态与电脑游戏互动，工厂中准确地焊接部件并检查缺陷，忙碌的购物季节帮助仓库分拣商品，离开家时扫地机器人清洁房间，自动将数码照片进行识别分类。

或许在不久的将来（如图 1-9 所示），超市电子秤在称重的同时就能辨别出蔬菜的种类；门禁系统能分辨出是带着礼物的朋友，还是手持撬棒即将行窃的歹徒；可穿戴设备和手机帮助我们识别出镜头中的物体并搜索出相关信息。更奇妙的是，它还能超越人类双眼的感官，用声波、红外线来感知这个世界，观察云层的汹涌起伏预测天气，监测车辆的运行调度交通，甚至突破我们的想象，帮助理论物理学家分析超过三维的空间中物体的运动。

这些，似乎并不遥远。

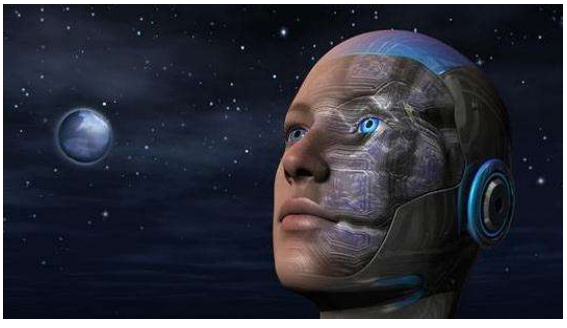

图 1-9　计算机视觉的未来

1.3 本章小结

本书在写作的时候，应用深度学习作为计算机视觉的解决方案已经得到共识，深度神经网络已经明显地优于其他学习技术以及设计出的特征提取计算。神经网络的发展浪潮已经迎面而来，在过去的历史发展中，深度学习、人工神经网络以及计算机视觉大量借鉴和使用了人类以及其他生物视觉神经方面的知识和内容，而且得益于最新的计算机硬件水平的提高，更多的数据集的收集以及能够设计更深的网络计算，使得深度学习的普及性和应用性都有了非常快的发展。充分利用这些资源进一步提高使用深度学习进行计算机视觉的研究，并将其带到一个新的高度和领域是本书写作的目的和对读者的期望。

第 2 章

◂Python的安装与使用▸

“人生苦短，我用 Python”。

这是 Python 语言在自身宣传和推广中使用的口号，针对深度学习也是这样。对于相关研究人员，最直接、最简洁的需求就是将自己的想法从纸面进化到可以运行的计算机代码，在这个过程中，所需花费的精力越小越好。

Python 完全可以满足这个需求，在计算机代码的编写和实现过程中，Python 简洁的语言设计本身可以帮助用户避开没必要的陷阱，减少变量声明，随用随写，无须对内存进行释放，这些都极大地帮助了我们使用 Python 编写出需要的程序。

其次，Python 的社区开发成熟，有非常多的第三方类库可以使用。在本章中还会介绍 NumPy、PIL 以及 threading 三个主要的类库，这些开源的算法类库在后面的程序编写过程中会起到极大的作用。

最后，相对于其他语言，Python 有较高的运行效率，而且得益于 Python 开发人员的不懈努力，Python 友好的接口库甚至可以加速程序的运行效率，而无须去了解底层的运行机制。

“人生苦短，何不用 Python。”Python 让其使用者专注于逻辑和算法本身而无须纠结一些技术细节。Python 作为深度学习以及 TensorFlow 框架主要的编程语言，更需要读者去掌握与学习。

2.1 Python 基本安装和用法

Python 是深度学习的首选开发语言，但是对于安装来说，第三方提供了集成了大量科学计算类库的 Python 标准安装包，目前最常用的是 Anaconda。

Anaconda 里面集成了很多关于 Python 科学计算的第三方库，主要是安装方便，而 Python 是一个脚本语言，如果不使用 Anaconda，那么第三方库的安装会较为困难，各个库之间的依赖性就很难连接得很好。因此，这里推荐使用集合了大量第三方类库的安装程序 Anaconda 来替代 Python 的安装。

2.1.1　Anaconda 的下载与安装

1. 第一步：下载和安装

Anaconda 的下载地址是 https://www.continuum.io/downloads/，页面如图 2-1 所示。

图 2-1　Anaconda 下载页面

目前下载的是 Anaconda 4.3.1 版本，里面集成了 Python 3.6。读者可以根据自己的操作系统进行下载。

这里作者选择的是 Windows 版本，下载之后双击运行即可安装，过程基本与其他软件一样。安装完成以后，出现程序面板，目录如图 2-2 所示。

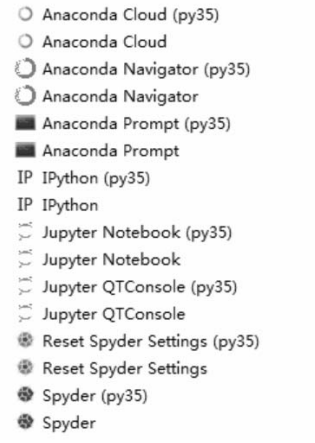

图 2-2　Anaconda 安装目录

2. 第二步：打开控制台

之后依次单击：开始→所有程序→Anaconda→Anaconda Prompt，打开窗口的效果如图 2-3

所示。这些步骤和打开 CMD 控制台类似，输入命令就可以控制和配置 Python。在 Anaconda 中最常用的是 conda 命令，这个命令可以执行一些基本操作。

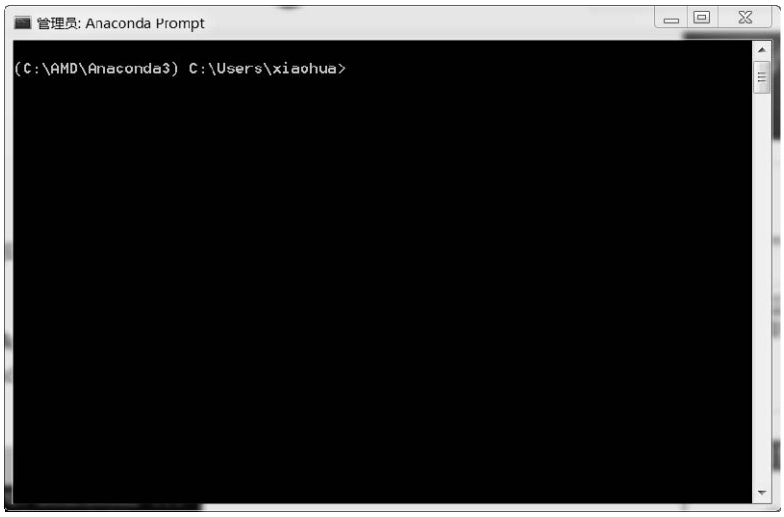

图 2-3　Anaconda Prompt 控制台

3. 第三步：验证 Python

在控制台中输入 python，会打印出版本号以及控制符号。然后在 Python 控制符号>>>后输入代码：

```
print("hello Python")
```

输入结果如图 2-4 所示。

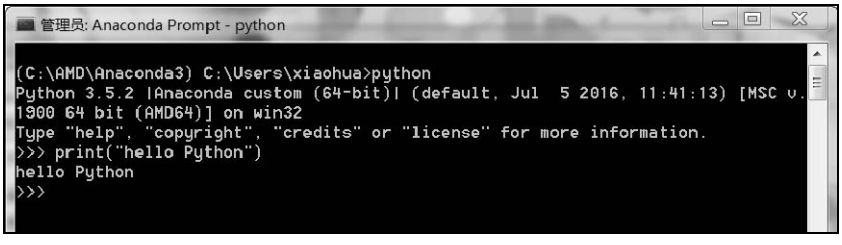

图 2-4　验证 Anaconda Python 安装成功

4. 第四步：使用 conda 命令

作者建议读者使用 Anaconda 的好处在于，它能够方便地帮助读者安装和使用大量第三方类库，查看已安装的第三方类库的命令是：

```
conda list
```

在 Anaconda Prompt 控制台中输入 exit()，或者重新打开 Anaconda Prompt 控制台后直接输入 conda list 代码，结果如图 2-5 所示。

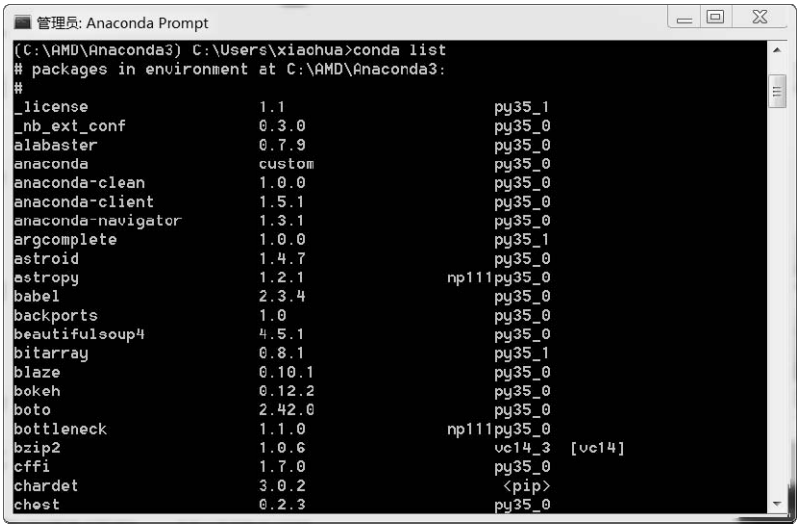

图 2-5　列出已安装的第三方类库

Anaconda 中使用 conda 进行操作的方法还有很多，其中最重要的是安装第三方类库，命令如下：

```
conda install name
```

这里的 name 是需要安装的第三方类库名，例如当需要安装 NumPy 包（这个包已经安装过），那么输入相应的命令就是：

```
conda install numpy
```

使用 Anaconda 的一个特别的好处就是可以自动安装包的依赖类库，如图 2-6 所示，这样大大减轻了使用者在安装和使用某个特定类库时碰到的依赖类库缺失的困难，使得后续工作顺利进行。

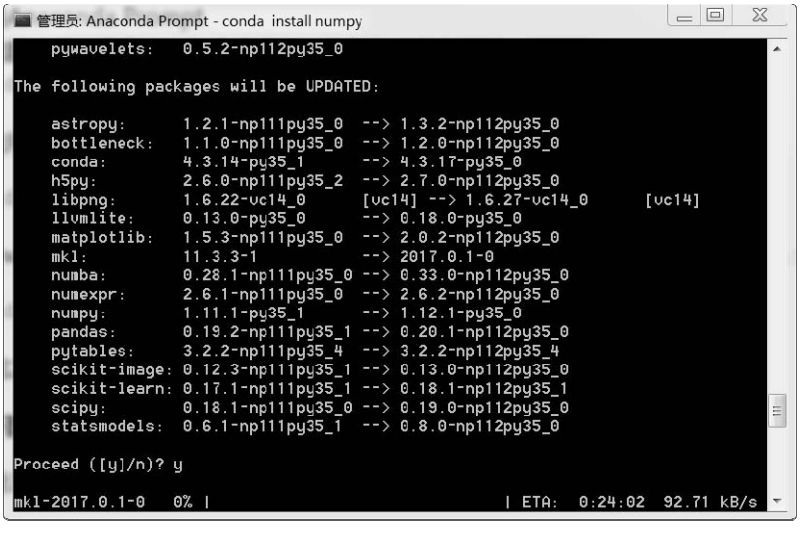

图 2-6　自动获取或更新依赖类库

2.1.2 Python 编译器 PyCharm 的安装

和其他语言类似，Python 程序的编写可以使用 Windows 自带的控制台进行。但是这种方式对于较为复杂的程序工程来说，容易混淆相互之间的层级和交互文件，因此在编写程序工程时，作者建议使用专用的 Python 编译器 PyCharm。

1.第一步：PyCharm 的下载和安装

PyCharm 的下载地址为 http://www.jetbrains.com/pycharm/。

进入 Download 页面后可以选择不同的版本，有收费的专业版和免费的社区版，如图 2-7 所示。这里作者建议读者选择免费的社区版即可。本书使用的版本为 2017.1.2。

图 2-7 PyCharm 的免费版

安装文件下载下来后，双击运行进入安装界面，直接单击 Next 按钮采用默认安装即可，如图 2-8 所示。

图 2-8 PyCharm 的安装文件

需要注意的是，在安装 PyCharm 的过程中需要对安装的位数进行选择，这里建议读者选择与所安装 Python 相同位数的文件，如图 2-9 所示。

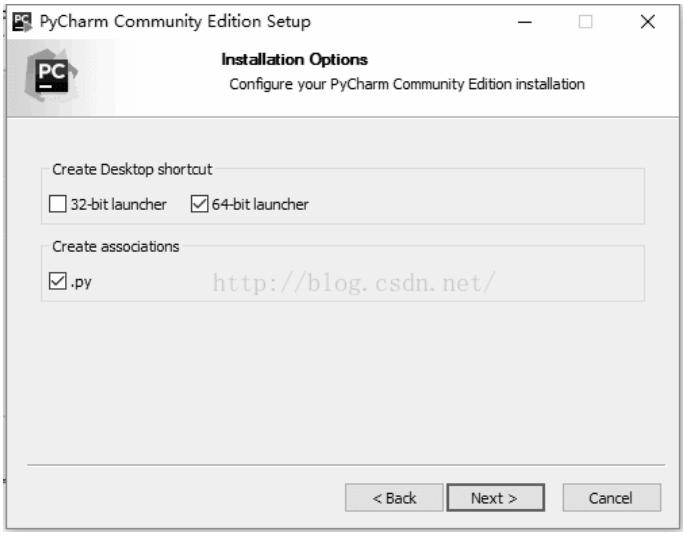

图 2-9　PyCharm 的位数选择

安装完成后出现 Finish 按钮，单击后完成安装，如图 2-10 所示。

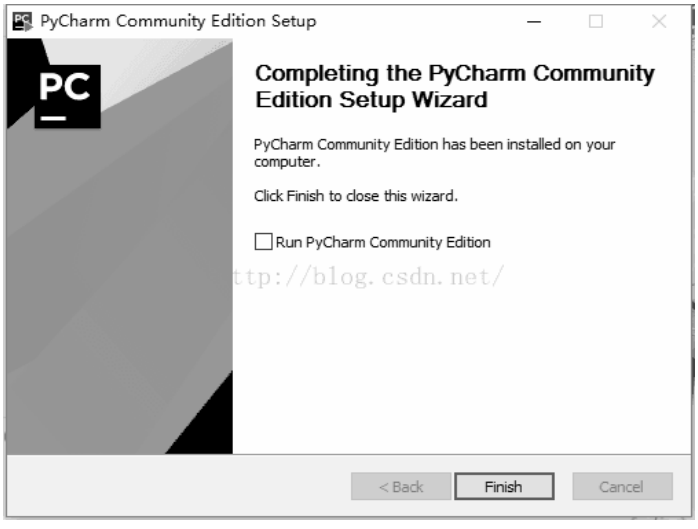

图 2-10　PyCharm 安装完成

2. 第二步：使用 PyCharm 创建程序

单击桌面上新生成的图标进入 PyCharm 程序界面，首先是第一次启动的定位，如图 2-11 所示。

图 2-11　PyCharm 启动定位

这里是对程序存储的定位，一般建议选择第二项，由 PyCharm 自动指定，单击 OK 按钮。之后单击弹出的 Accept 按钮，接受相应的协议，进入界面配置选项，如图 2-12 所示。

图 2-12　PyCharm 界面配置

在配置区域可以选择自己的使用风格对 PyCharm 的界面进行配置，如果对其不熟悉的话，直接单击 OK 按钮使用默认配置即可。

最后就是创建一个新的工程，如图 2-13 所示。

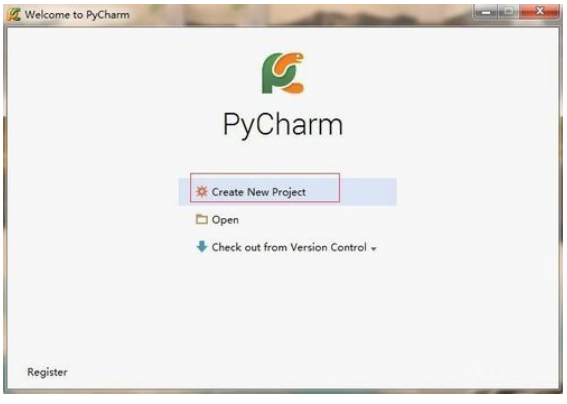

图 2-13　PyCharm 工程创建界面

在这里，建议读者新建一个 PyCharm 的工程文件，如图 2-14 所示。

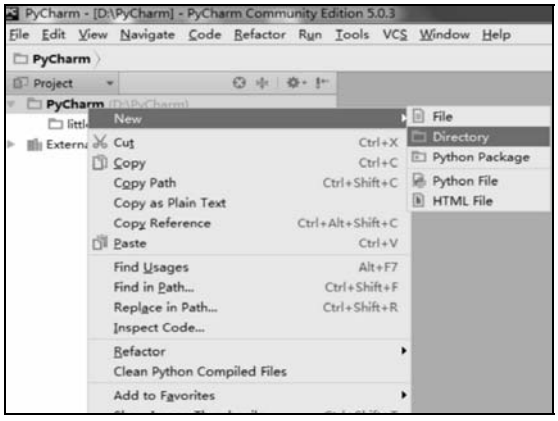

图 2-14　PyCharm 新建文件界面

之后右击新建的工程名“PyCharm”，在弹出的菜单中单击“new”|“Python File”新建一个“helloworld”文件，内容如图 2-15 所示。

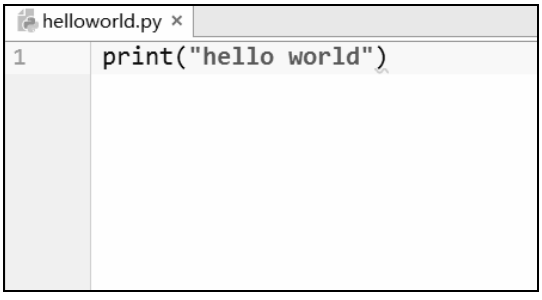

图 2-15　新建的 helloworld 文件

输入代码后，单击 run|run…菜单开始运行，或者直接右击“helloworld.py”后，在弹出的菜单中选择 run。如果成功输出“hello world”，那么恭喜你，Python 与 PyCharm 的配置就成功了！

2.1.3　使用 Python 计算 softmax 函数

对于 Python 科学计算来说，最简单的想法就是可以将数学公式直接表达成程序语言，可以说，Python 满足了这个想法。本小节将使用 Python 实现和计算一个深度学习中最为常见的函数——softmax 函数。至于这个函数的作用，现在暂时不做说明，作者只是带领读者尝试实现其程序的编写。

首先 softmax 计算公式如下所示：

$$S_i = \frac{e^{V_i}}{\sum_{0}^{j} e^{V_i}}$$

其中 V_i 是长度为 j 的数列 V 中的一个数，带入 softmax 的结果其实就是先对每一个 V_i 取 e

为底的指数计算变成非负，然后除以所有项之和进行归一化，之后每个 V_i 就可以解释成观察到的数据 V_i 属于某个类别的概率，或者称作似然（Likelihood）。

softmax 用以解决概率计算中概率结果大占绝对优势的问题。例如函数计算结果中的 2 个值 a 和 b，且 a>b，如果简单地以值的大小为单位衡量的话，那么在后续的使用过程中，a 永远被选用，而 b 由于数值较小而不会被选择，但是有时也需要数值小的 b 被使用，那么 softmax 就可以解决这个问题。

softmax 按照概率选择 a 和 b，由于 a 的概率值大于 b，在计算时 a 经常会被取得，而 b 由于概率较小，取得的可能性也较小，但是也有概率被取得。

公式 softmax 的代码如下所示：

```
import numpy
def softmax(inMatrix):
m,n = numpy.shape(inMatrix)
outMatrix = numpy.mat(numpy.zeros((m,n)))
soft_sum = 0
for idx in range(0,n):
    outMatrix[0,idx] = math.exp(inMatrix[0,idx])
    soft_sum += outMatrix[0,idx]
for idx in range(0,n):
    outMatrix[0,idx] = outMatrix[0,idx] / soft_sum
return outMatrix
```

可以看到，当传入一个数列后，分别计算每个数值所对应的指数函数值，之后将其相加后计算每个数值在数值和中的几率。

```
a = numpy.array([[1,2,1,2,1,1,3]])
```

结果如下所示：

```
[[ 0.05943317  0.16155612  0.05943317  0.16155612  0.05943317  0.05943317
   0.43915506]]
```

2.2 TensorFlow 类库的下载与安装（基于 CPU 模式）

对于 TensorFlow 的安装来说，由于在前面已指导读者使用 Anaconda 进行 Python 环境的

配置。TensorFlow 的安装就非常简便了。

首先打开 Anaconda 安装目录中的 Anaconda Prompt，如图 2-16 所示。

图 2-16　Anaconda Prompt 控制台

之后直接输入如下：

```
pip install tensorflow
```

之后将自动根据你所安装的 Anaconda 环境，安装对应的最新 TensorFlow 类库。等待提示安装成功即可。

安装完成以后输入如下命令行：

```
conda list
```

在显示的目录中找到 tensorflow，可以查看对应的版本号，如图 2-17 所示。

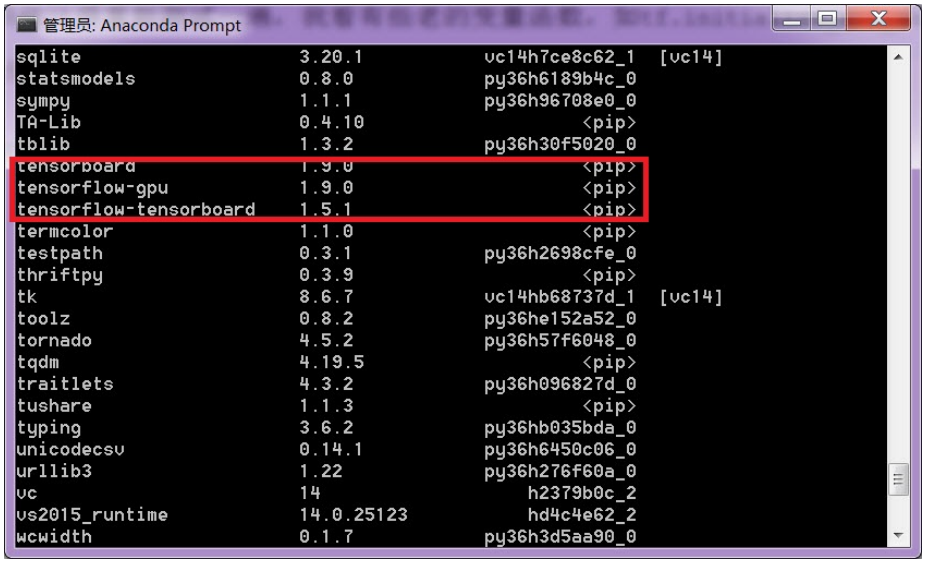

图 2-17　测试程序 tensorflow 安装目录

需要验证 TensorFlow 安装的情况，首先打开 PyCharm，新建一个 hello tensorflow 的 Python 文件，代码如程序 2-1 所示。

【程序 2-1】

```
import tensorflow as tf
hello = tf.constant("hello tensorflow")
sess = tf.Session()
```

```
print(sess.run(hello))
```

打印结果：b'hello tensorflow' 。

2.3 TensorFlow 类库的下载与安装（基于 GPU 模式）

2.2 节中安装的是基于 CPU 模式的 TensorFlow 类库，这也是一般默认安装的 TensorFlow 模式，而往往在进行大规模数据计算时需要安装基于 GPU 模式的 TensorFlow，输入如下代码：

```
pip install tensorflow-gpu
```

等待提示成功后即可认为基于 GPU 模式的 TensorFlow 安装完毕。但是如果需要真正使用 GPU 模式对数据进行处理，除了安装 tensorflow-gpu 库包以外，还需要安装 CUDA 与 cuDNN，这是 NVIDIA 为了使用 GPU 进行程序运算专门提供的工具包。

2.3.1 CUDA 配置

由于本书使用的是最新的 tensorflow-gpu 版本，其对应 cuda 9.0.dll，因此就要下载 cuda 9.0 对应 Windows 版本的安装文件。

（1）下载地址：https://developer.nvidia.com/cuda-90-download-archive。

（2）下一步是选择下载的版本（如图 2-18 所示），这里 NVIDIA 提供了多种版本，请读者自行选择对应的操作系统以及版本号。

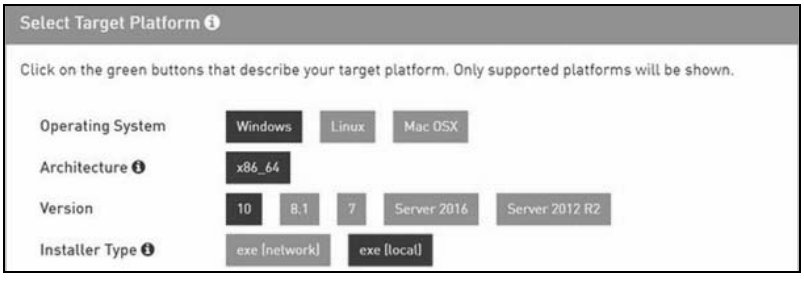

图 2-18　选择的版本号

（3）还需注意最后的 Installer Type 选项，exe（network）是在线安装版（如图 2-19 所示），也就是执行这个安装程序需要联网。exe（local）是离线安装版（如图 2-20 所示），这个文件比较大。选完后，单击下面的 Download 按钮就可以下载。

图 2-19　在线安装程序

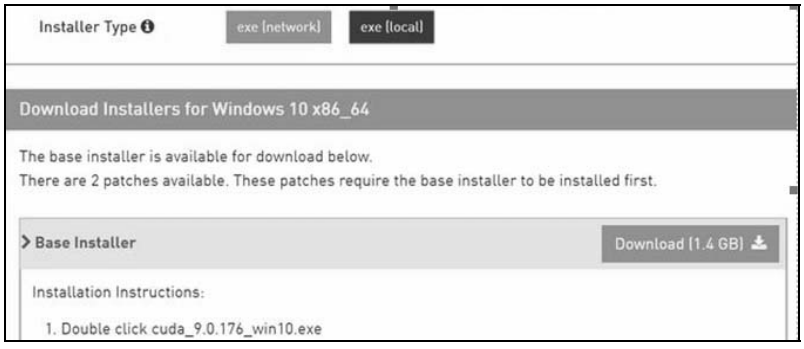

图 2-20　离线下载程序

（4）下载完成后，双击运行文件然后单击 OK 按钮，等进度条走完，就会进入安装界面，如图 2-21 所示。

图 2-21 进入安装界面

（5）之后继续下一步，进入加载界面，如图 2-22 所示。

图 2-22　进入加载界面

（6）检查系统兼容性，如果检测通过了，那么恭喜你，你的显卡可以安装 CUDA，如果没有通过，只能抱歉地告诉你，只能 pip uninstall tensorflow-gpu，然后执行 pip install tensorflow，这种情况是你的电脑显卡不支持 tensorflow-gpu 加速。

（7）之后是软件许可协议，如图 2-23 所示，单击“同意并继续”按钮。

图 2-23　软件许可协议

（8）此时出现安装选项，如图 2-24 所示。选中“精简”单选按钮，然后单击“下一步”按钮，之后等待安装完成即可。

图 2-24　选择安装模式

（9）完成后，在环境变量检查 PATH 路径。在计算机桌面上的“计算机”图标上右击，打开属性→高级系统设置→环境变量，发现已经有 CUDA_PATH 和 CUDA_PATH_V9_0 两个环境变量。

CUDA_PATH 是 C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA9.0，仅仅如此是不够的，还需要在环境变量里的 PATH 全局变量中加入 bin 和 lib\x64 目录的路径：

```
CUDA_SDK_PATH = C:\ProgramData\NVIDIA Corporation\CUDA Samples\v9.0
CUDA_LIB_PATH = %CUDA_PATH%\lib\x64
CUDA_BIN_PATH = %CUDA_PATH%\bin
CUDA_SDK_BIN_PATH = %CUDA_SDK_PATH%\bin\win64
CUDA_SDK_LIB_PATH = %CUDA_SDK_PATH%\common\lib\x64
```

打开 cmd，输入$ nvcc　-V 可以验证 CUDA 的安装是否成功。

2.3.2　cuDNN 配置

对于 TensorFlow 而言，真正实现加速的是 cuDNN，cuDNN 调用的是 CUDA 显卡驱动。所以最后我们要配置 cuDNN 这个模块。

cuDNN 的全称为 NVIDIA CUDA Deep Neural Network library，是 NVIDIA 专门针对深度神经网络（Deep Neural Networks）中的基础操作而设计的基于 GPU 的加速库。cuDNN 为深度神经网络中的标准流程提供了高度优化的实现方式，例如 convolution、pooling、normalization 以及 activation layers 的前向以及后向过程。

cuDNN 只是 NVIDIA 深度神经网络软件开发包中的其中一种加速库。想了解 NVIDIA 深度神经网络加速库中的其他包，请访问链接 https://developer.nvidia.com /deep-learning-software。

下面我们说一下正确安装 cuDNN 的方式，其实按官方安装说明进行安装就可以了。

（1）从 https://developer.nvidia.com/cudnn 上下载 cuDNN 相应版本的压缩包（可能需要注册或登录）。

（2）如果这个压缩包不是.tgz 格式的，把这个压缩包重命名为.tgz 格式。解压当前的.tgz 格式的软件包到系统中的任意路径，解压后的文件夹名为 CUDA。文件夹中包含三个子文件夹：一个为 include；一个为 lib；还有一个是 bin。

（3）复制上述 3 个文件夹到 CUDA_PATH 指定的路径下面（见图 2-25）。检查一下环境变量中是否有 lib/x64 文件夹的配置，这一步很重要。

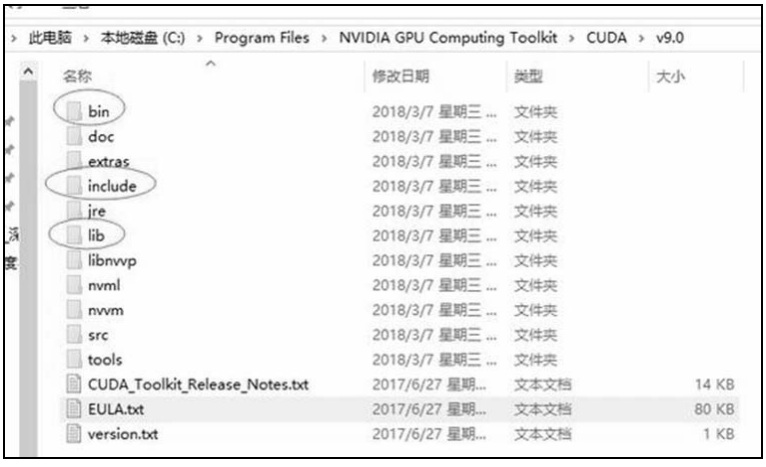

图 2-25　解压后的 cuDNN 文件

（4）之后仿照上一节的代码对其进行验证。这里需要注意的是，第一次使用 tensorflow-gpu 模式进行处理的时候，由于需要对显卡进行甄别，加载的速度较慢，同时打印的内容也较多，如图 2-26 所示。

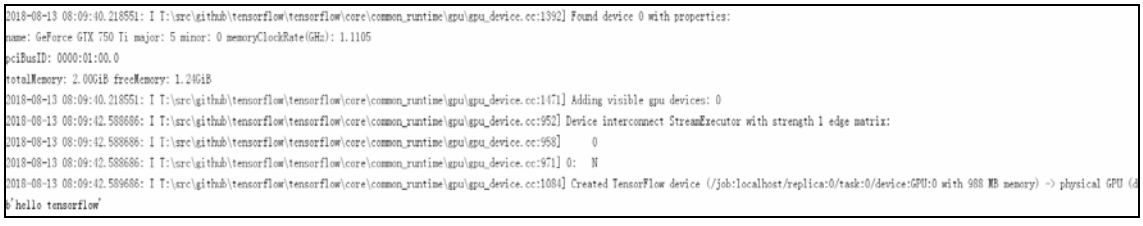

图 2-26　第一次加载 tensorflow-gpu 模式

2.4 OpenCV 类库的下载与安装

OpenCV 是专属 Python 的视觉程序包，OpenCV 的安装较为复杂，从一般的资料上看，可能读者需要先安装其他附属的工具包，但是实际上并不需要如此，对于一般使用者来说，最好的方法就是直接安装其他程序设计人员编译好的 whl 安装文件。

（1）首先下载编译好的 OpenCV 类库安装文件，下载地址为：http://www.lfd.uci.edu/~gohlke/pythonlibs。之后使用 Ctrl+F 组合键，以 OpenCV 为关键词进行搜索，可以搜索到已编译好的、以 whl 为后缀的 OpenCV 安装文件，这是 pip 使用的安装文件，如图 2-27 所示。

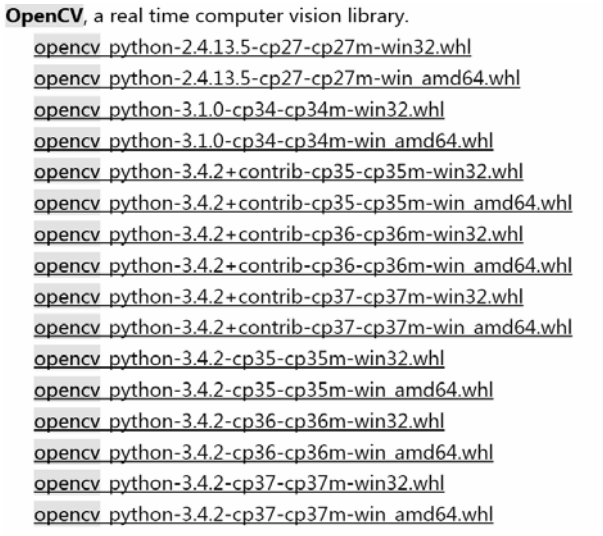

图 2-27　选择 OpenCV 的类库包

（2）在这里，whl 文件为编译好的 Python 类库安装文件，可以根据读者安装的 Python 版本与 OpenCV 的最新版本下载对应的 whl 文件。

安装了 Anaconda 的读者，可以直接打开 Anaconda Prompt 输入安装 whl 文件的命令，代码如下：

```
pip install C://XXX/CCC/OpenCV_python-3.4.2-cp36-cp36m-win_amd64.whl
```

后面的地址是下载的 whl 文件在本地计算机存储的地址，即通过命令行形式使用 pip 安装本地的 whl 文件，读者可以自行选择安装地址。

（3）安装结束后，打开 PyCharm，新建一个名为 Opencv_TEST 的文件，如图 2-28 所示。

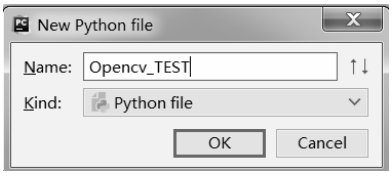

图 2-28　新建 OpenCV 的测试程序

输入如下代码进行测试：

【程序 2-2】

```
import cv2
```

```
jpg = cv2.imread("1.jpg")
cv2.imshow("test.jpg",jpg)
cv2.waitKey()
```

其中“1.jpg”是保存在程序文件同一目录下的图片，而 imshow 函数是 cv2 显示图片的函数，其作用是将以矩阵形式保存的图片显示出来，waitKey 函数是等待函数，只有等待键盘触及后才能关闭显示的图片。

具体结果请读者自行完成。

2.5 Python 常用类库中的 threading

除了前面介绍的本书必须使用的两个类库 ThensorFlow 与 OpenCV，Python 还提供了多种多样的用于不同目的和方向的类库。

Python 常用类库参见表 2-1。

表 2-1　Python 常用类库

分 类	名 称	库 用 途
科学计算	Matplotlib	用 Python 实现的类 matlab 的第三方库，用以绘制一些高质量的数学二维图形
	SciPy	基于 Python 的 matlab 实现，旨在实现 matlab 的所有功能
	NumPy	基于 Python 的科学计算第三方库，提供了矩阵、线性代数、傅立叶变换等的解决方案
GUI	PyGtk	基于 Python 的 GUI 程序开发 GTK+库
	PyQt	用于 Python 的 QT 开发库
	WxPython	Python 下的 GUI 编程框架，与 MFC 的架构相似
	Tkinter	Python 下标准的界面编程包，因此不算是第三方库
其他	BeautifulSoup	基于 Python 的 HTML/XML 解析器，简单易用
	PIL	基于 Python 的图像处理库，功能强大，对图形文件的格式支持广泛
	MySQLdb	用于连接 MySQL 数据库
	cElementTree	高性能 XML 解析库，Py2.5 应该已经包含了该模块，因此不算是一个第三方库
	PyGame	基于 Python 的多媒体开发和游戏软件开发模块
	Py2exe	将 Python 脚本转换为 Windows 上可以独立运行的可执行程序
	pefile	Windows PE 文件解析器

表 2-1 给出了 Python 中常用类库的名称和说明，到目前为止，Python 中已经有 7000 多个可以使用的类库可供计算机工程人员以及科学研究人员使用。

2.5.1　threading 库的使用

对于希望充分利用计算机性能的程序设计者来说，多线程的应用是必不可少的一个重要技能。多线程类似于使用计算机的一个核心执行多个不同任务。多线程的好处如下：

- 使用线程可以把需要使用大量时间的计算任务放到后台去处理。
- 减少资源占用，加快程序的运行速度。
- 在传统的输入输出以及网络收发等普通操作上，后台处理可以美化当前界面，增加界面的人性化。

本节将详细介绍 Python 中操作线程的模块：threading，相对于 Python 既有的多线程模块 thread，threading 重写了部分 API 模块，对 thread 进行了二次封装，从而大大提高了执行效率。

2.5.2　threading 模块中最重要的 Thread 类

Thread 是 threading 模块中最重要的类之一，可以使用它来创造线程。其具体使用方法是创建一个 threading.Thread 对象，在它的初始化函数中将需要调用的对象作为初始化参数传入，具体代码如程序 2-3 所示。

【程序 2-3】

```
#coding = utf8
import threading,time
count = 0
class MyThread(threading.Thread):
    def __init__(self,threadName):
        super(MyThread,self).__init__(name = threadName)

    def run(self):
        global count
        for i in range(100):
            count = count + 1
            time.sleep(0.3)    print(self.getName() , count)

for i in range(2):
    MyThread("MyThreadName:" + str(i)).start()
```

在上面定义的 MyThread 类中，重写了从父对象继承的 run 方法，run 方法中，将一个全局变量逐一增加，在接下来的代码中，创建了 5 个独立的对象，分别调用其 start 方法，最后将结果逐一打印。

可以看到在程序中，每个线程被赋予了一个名字，然后设置每隔 0.3 秒打印输出本线程的计数，即计数加 1。而 count 被人为地设置成全局共享变量，因此在每个线程中都可以自由地对其进行访问。

程序运行结果如图 2-29 所示。

```
MyThreadName:0 2
MyThreadName:1 2
MyThreadName:1 4
MyThreadName:0 4
MyThreadName:1 6
MyThreadName:0 6
MyThreadName:1 8
MyThreadName:0 8
```

图 2-29　程序运行结果

通过上面的结果可以看到，每个线程被起了一个对应的名字，而在运行的时候，线程所计算的计数被同时增加，这样可以证明，在程序运行过程中，2 个线程同时对一个数进行操作，并将其结果进行打印。

其中的 run 方法和 start 方法并不是 threading 自带的方法，而是从 Python 本身的线程处理模块 Thread 中继承来的。run 方法的作用是在线程被启动以后，执行预先写入的程序代码。一般而言，run 方法所执行的内容被称为 Activity，而 start 方法是用于启动线程的方法。

2.5.3　threading 中的 Lock 类

虽然线程可以在程序的执行过程中极大地提高程序的执行效率，但是其带来的影响却难以忽略。例如在上一个程序中，由于每隔一定时间打印当前的数值，应该逐次打印的数据却变成了 2 个相同的数值被打印出来，因此需要一个能够解决这类问题的方案出现。

Lock 类是 threading 中用于锁定当前线程的锁定类，顾名思义，其作用是对当前运行中的线程进行锁定，只有被当前线程释放后，后续线程才可以继续操作。

```
import threading
lock = threading.Lock()
lock.acquire()
lock.release()
```

类中主要代码如上所示。acquire 方法提供了确定对象被锁定的标志，release 在对象被当前线程使用完毕后将当前对象释放。修改后的代码如程序 2-4 所示。

【程序 2-4】

```
#coding = utf8
import threading,time,random

count = 0
class MyThread (threading.Thread):

    def __init__(self,lock,threadName):
        super(MyThread,self).__init__(name = threadName)
```

```
        self.lock = lock

    def run(self):
        global count
        self.lock.acquire()
        for i in range(100):
            count = count + 1
            time.sleep(0.3)
            print(self.getName() , count)
        self.lock.release()

    lock = threading.Lock()
    for i in range(2):
        MyThread (lock,"MyThreadName:" + str(i)).start()
```

可以看到 Lock 被传递给 MyThread，并在 run 方法中人为锁定当前的线程，必须等当前线程执行完毕后，后续的线程才可以继续执行。程序执行结果如图 2-30 所示。

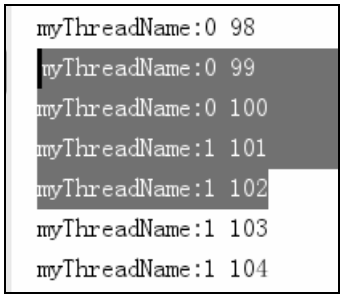

图 2-30　程序运行结果

可以看到，其中变色的部分，线程 2 只有等线程 1 完全结束后，才执行后续的操作。本程序中，Thread1 等到 Thread0 完全结束后，才执行自己的操作。

2.5.4　threading 中的 join 类

join 类是 threading 中用于堵塞当前主线程的类，其作用是阻止全部的线程继续运行，直到被调用的线程执行完毕或者超时。具体代码如程序 2-5 所示。

【程序 2-5】

```
import threading, time
def doWaiting():
    print('start waiting:', time.strftime('%S'))
    time.sleep(3)
    print('stop waiting', time.strftime('%S'))
    thread1 = threading.Thread(target = doWaiting)
    thread1.start()
```

```
    time.sleep(1)                                    #确保线程 thread1 已经启动
    print('start join')
    thread1.join()                                   #将一直堵塞，直到 thread1 运行结束
    print('end join')
```

程序的运行结果如图 2-31 所示。

```
start waiting: 29
start join
stop waiting 32
end join
```

图 2-31　程序运行结果

其中的 time 方法设定了当前的时间，当 join 启动后，堵塞了调用整体进程的主进程，而只有当被堵塞的进程执行完毕后，后续的进程才继续执行。

除此之外，对于线程的使用，Python 还有很多其他的方法，例如 threading.Event 以及 threading.Condition 等，这些都是在程序设计时能够极大地帮助程序设计人员编写合适程序的工具。限于篇幅，这里不再一一进行介绍，读者可以参考相关图书，在后续的使用过程中，作者会带领读者了解和掌握更多的相关内容。

2.6 本章小结

本章介绍了 Python 的基本安装和编译器的使用。在这里推荐读者使用 PyCharm 免费版作为 Python 编辑器，这有助于更好地安排工程文件的配置和程序的编写。本章还介绍了全书最重要的两个类库：TensorFlow 和 OpenCV 的下载和安装。

同时，本章还介绍了最常用的一些类库，这里只是对线程类做了详细的介绍，线程类是 Python 最为重要的一个类库，在后面的代码编写中会频繁遇到。

本章是 Python 最基础的内容，后面的章节还将以 Python 使用为主，并且还会介绍更多的 Python 类库，希望读者能够掌握相关内容。

第 3 章

◂Python数据处理及可视化▸

前面章节中对 Python 的安装做了一个基本的介绍，并且建议读者使用 PyCharm 免费版作为使用 Python 编写程序的编译器。相对于使用控制台或自带的编译器，可以更加直观和明晰化地对所构建的工程做出层次安排。

本章将使用 Python 对数据的处理和可视化做出介绍，主要向读者介绍 Python 的使用，并对第 3 章中深度学习使用的一些算法做出复写，同时也向读者介绍第三方类库的使用，对于大多数的 Python 程序设计，建议读者使用已有的类库来解决问题，而不是自行编写相应的代码。这是初学者非常易犯的错误，对于 Python 来说，大多数的类库都是在底层使用效率更高的 C 语言实现，并且由经验丰富的程序设计人员编写，因此不建议读者自行设计和完成相应的程序。

“人生苦短，我用 Python！编程复杂，请用类库！”

3.1 从小例子起步——NumPy 的初步使用

从小例子起步，本节将介绍 NumPy 的基础使用。

3.1.1 数据的矩阵化

对于机器学习来说，数据是一切的基础。一切数据又不是单一的存在，其构成往往由很多的特征值所决定。表 3-1 是用以计算回归分析的房屋面积与价格对应表，这里加上了每个房屋中地下室的有无。

表 3-1 某地区房屋面积与价格对应表

价格/千元	面积/平方米	卧室/个	地下室
200	105	3	无
165	80	2	无
184.5	120	2	无
116	70.8	1	无
270	150	4	有

表 3-1 是数据的一般表示形式，但是对于机器学习的过程来说，这是不可辨识的数据，因

此需要对其进行调整。

常用的机器学习表示形式为数据矩阵，即可以将表 3-1 表示为一个专门的矩阵形式，见表 3-2。

表 3-2　某地区房屋面积与价格计算矩阵

ID	Price	Area	Bedroom	Basement
1	200	105	3	False
2	165	80	2	False
3	184.5	120	2	False
4	116	70.8	1	False
5	270	150	4	True

从表 3-2 中可以看到，一行代表一个单独的房屋价格和对应的特征属性。第一列是 ID，即每行的标签。标签是独一无二的，一般不会有重复出现。第二列是价格，一般被称为矩阵的目标。目标可以是单纯的数字，也可以是布尔变量或者一个特定的表示。表 3-2 中的标签是房屋的价格，是一个数字标签。第 2、3、4 列是属性值，也是标签所对应的特征值，根据此特征值的不同，每行所对应的目标也是有所不同的。

不同的 ID 用于表示不同的目标。一般来说，机器学习的最终目的就是使用不同的特征属性对目标进行区分和计算。已有的目标是观察和记录的结果，而机器学习的过程就是创建一个可进行目标识别的模型的过程。

建立模型的过程称为机器学习的训练过程，其速度和正确率主要取决于算法的选择，而算法是目标和属性之间建立某种一一对应的关系的过程。这点在前面介绍机器学习过程的时候已经有所介绍。

继续回到表 3-2 的矩阵中。通过观察可知，矩阵中所包含的属性有两种，分别是数值型变量和布尔型变量。其中第 2、3、4 列是数值变量，这也是机器学习中最常使用和辨识的类型。第 5 列是布尔型变量，用以标识对地下室存在的判定。

这样做的好处在于，机器学习在工作时是根据采用的算法进行建模的，算法的描述只能对数值型变量和布尔型进行处理，而对于其他类型的变量处理相对较少。即使后文有针对文字进行处理的机器学习模型，其本质也是将文字转化成矩阵向量进行处理，这一点将在后文继续介绍。

当机器学习建模的最终目标是求得一个具体数值时，即目标是一个数字，那么机器学习建模的过程基本上可以被转化为回归问题。差别在于是逻辑回归还是线性回归。

当目标为布尔型变量时，问题大多数被称为分类问题，而常用的建模方法是第 4 章中介绍的决策树方法。一般来说，当分类的目标是两个的时候，问题被转化为二元分类；而分类的结果多于两个的时候，分类称为多元分类。

许多情况下，机器学习建模和算法的设计是由程序设计和研究人员所选择的，而具体采用何种算法和模型也没有一定的要求。回归问题可以被转化为分类问题，而分类问题往往也可以由建立的回归模型解决。这点没有特定的要求。

3.1.2　数据分析

对于数据来说，在进行机器学习建模之前，需要对数据进行基本的分析和处理。

从图 3-1 可以看到，对于数据集来说，在进行数据分析之前，需要知道很多东西。首先需要知道的是一个数据集的数据多少和每个数据所拥有的属性个数，对于程序设计人员和科研人员来说，这些都是简单的事；但是对于机器学习的模型来说，是必不可少的内容。

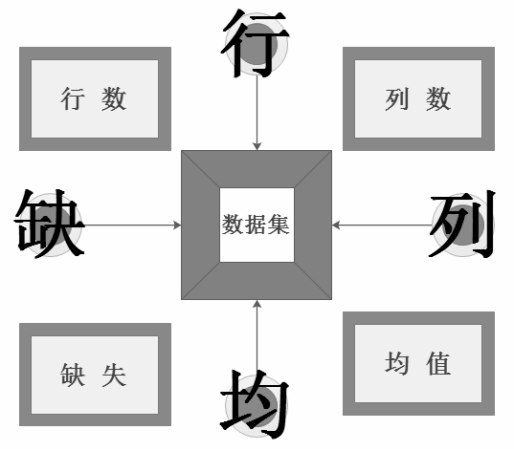

图 3-1　数据分析的要求

除此之外，对于数据集来说，缺失值的处理也是一个非常重要的操作。最简单的处理方法是对有缺失值的数据进行整体删除。问题在于，机器学习的数据往往来自于现实社会中，因此可能数据集中大多数的数据都会存在某些特征属性缺失，解决的办法往往是采用均值或者与目标数据近似的数据特征属性替代。有些情况下替代方法是可取的，有些情况下替代或者采用均值的办法处理缺失值是不可取的，因此要根据具体情况具体处理。

首先从一个小例子开始。以表 3-2 的矩阵为例，建立一个包含有数据集的数据矩阵，之后可以利用不同的方法对其进行处理。第一个代码如程序 3-1 所示。

【程序 3-1】

```
import numpy as np
data = np.mat([[1,200,105,3,False],[2,165,80,2,False],
           [3,184.5,120,2,False],[4,116,70.8,1,False],[5,270,150,4,True]])
row = 0
for line in data:
   row += 1
print( row  )
print( data.size)
```

程序 3-1 第一行引入了 Anaconda 自带的一个数据矩阵化的包。对于 NumPy，读者只需要知道 NumPy 系统是 Python 的一种开源数值计算扩展。这种工具可用来存储和处理大型矩阵，比 Python 自身的嵌套列表（nested list structure）结构要高效得多。

第一行代码的意思是引入 NumPy 将其重命名为 np 使用，第二行使用 NumPy 中的 mat()

方法建立一个数据矩阵，row 是引入的计算行数的变量，使用 for 循环将 data 数据读出到 line 中，每读一行就将 row 的计数加一。data.size 是计算数据集中全部数据的数据量，一般与行数相除则为列数。最终打印结果请读者自行打印测试。

需要说明的是，NumPy 将数据转化成一个矩阵的形式进行处理，其中具体的数据可以通过二元的形式读出，如程序 3-2 所示。

【程序 3-2】

```
import numpy as np
data = np.mat([[1,200,105,3,False],[2,165,80,2,False],
            [3,184.5,120,2,False],[4,116,70.8,1,False],[5,270,150,4,True]])

print( print( data[0,3]))
print( print( data[0,4] ))
```

最终打印结果如下：

```
3.0
0.0
```

细心的读者可能已经注意到，[0,3]对应的是矩阵中第 1 行第 4 列数据，其数值为 3，而打印结果为 3.0，这个没什么问题。对于[0,4]数据，矩阵中是 False 的布尔类型，而打印结果是 0。这点涉及 Python 的语言定义，其布尔值可以近似地表示为 0 和 1。即读者需要注意：

```
True = 1.0
False = 0
```

如果需要打印全部的数据集，即可调用如下方法：

```
Print( data)
```

将全部的数据以一个数据的形式进行打印，请读者自行打印测试。

3.1.3 基于统计分析的数据处理

除了最基本的数据记录和提取外，机器学习还需要知道一些基本数据的统计量，例如每一类型数据的均值、方差以及标准差等。当然在本书中并不需要手动或者使用计算器去计算以上数值，NumPy 提供了相关方法。程序如下所示。

【程序 3-3】

```
import numpy as np
data = np.mat([[1,200,105,3,False],[2,165,80,2,False],
            [3,184.5,120,2,False],[4,116,70.8,1,False],[5,270,150,4,True]])


col1 = []
for row in data:
   col1.append(row[0,1])
```

```
print( np.sum(col1))
print( np.mean(col1)    )
print( np.std(col1))
print( np.var(col1))
```

在上面的代码中，col1 生成了一个空的数据集，之后采用 for 循环将数据集进行填充。在程序 3-3 中第一列数据被填入 col1 数据集中，这也是一个类型数据的集合，之后依次计算数据集的和、均值、标准差以及方差，这些对于机器学习模型的建立有一定的帮助。

3.2 图形化数据处理——Matplotlib 包的使用

对于单纯的数字来说，光从读数据的角度并不能直观反映数字的偏差和集中程度，因此需要采用另外一种方法更好地分析数据。对于数据来说，没有什么能够比用图形来解释更为形象和直观的了。

3.2.1　差异的可视化

继续回到表 3-2 的数据，第二列是各个房屋的价格，其价格并不相同，因此直观地查看价格的差异和偏移程度是较为困难的一件事。

研究数值差异和异常的方法是绘制数据的分布程度，相对于合适的直线或曲线，其差异程度如何，以便帮助确定数据的分布。

【程序 3-4】

```
import numpy as np
import pylab
import scipy.stats as stats

data = np.mat([[1,200,105,3,False],[2,165,80,2,False],
            [3,184.5,120,2,False],[4,116,70.8,1,False],[5,270,150,4,True]])


col1 = []
for row in data:
   col1.append(row[0,1])

stats.probplot(col1,plot=pylab)
pylab.show()
```

结果如图 3-2 所示。

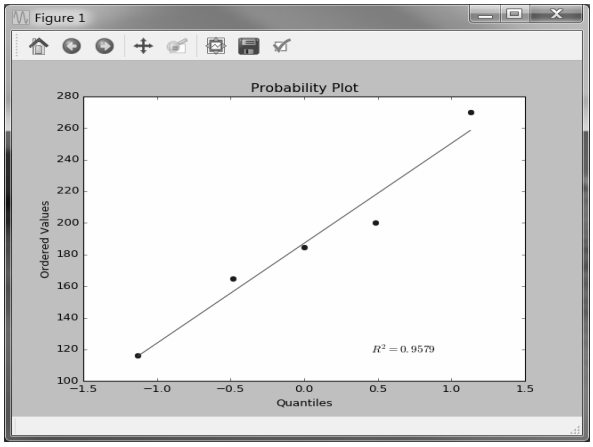

图 3-2　房屋价格的偏离展示

程序 3-4 展示了一个对价格的偏离程度的代码实现，col1 集合是价格的合集，scipy 是专门的机器学习的数据处理包，probplot 计算了 col1 数据集中数据在正态分布下的偏离程度。从图 3-2 可以看到，价格围绕一条直线上下波动，有一定的偏离，但是偏离情况不太明显。

其中 R（为 0.9579）指的是数据拟合的相关性，一般 0.95 以上就可以认为数据拟合程度比较好。

3.2.2　坐标图的展示

通过上文第一个对回归的可视化处理可以看到，可视化能够让数据更加直观地展现出来，同时可以对数据的误差表现得更为直观。

图 3-3 展示了一个横向坐标图，用以展示不同类别所占的比重。系列 1、2、3 可以分别代表不同的属性，类别 1~6 可以看作 6 个不同的特例。通过坐标图的描述可以非常直观地看到，不同的类别中不同的属性所占的比重如何。

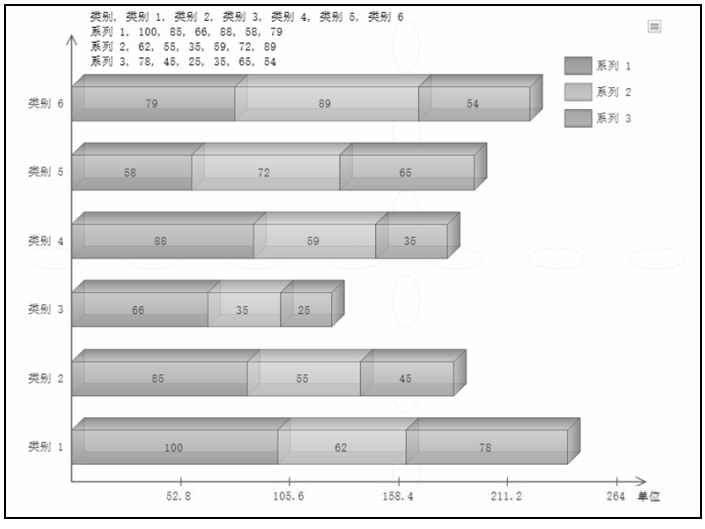

图 3-3　横向坐标图

可以看到，一个坐标图能够对数据进行展示，其最基本的要求是可以通过不同的行或者列表现出数据的某些具体值，不同的标签使用不同的颜色和样式以展示不同的系统关系。程序 3-5 展示对于不同目标的数据提取不同的行进行显示的代码。

【程序 3-5】

```
   import pandas as pd
   import matplotlib.pyplot as plot
   rocksVMines = pd.DataFrame([[1,200,105,3,False],[2,165,80,2,False],

[3,184.5,120,2,False],[4,116,70.8,1,False],[5,270,150,4,True]])

   dataRow1 = rocksVMines.iloc[1,0:3]
   dataRow2 = rocksVMines.iloc[2,0:3]
   plot.scatter(dataRow1, dataRow2)
   plot.xlabel("Attribute1")
   plot.ylabel(("Attribute2"))
   plot.show()

   dataRow3 = rocksVMines.iloc[3,0:3]
   plot.scatter(dataRow2, dataRow3)
   plot.xlabel("Attribute2")
   plot.ylabel("Attribute3")
   plot.show()
```

从图 3-4 可以看出，通过选定不同目标行中不同的属性，可以对其进行较好的衡量并比较两行之间的属性关系以及属性之间的相关性。不同的目标，即使属性千差万别，也可以构建相互关系图。

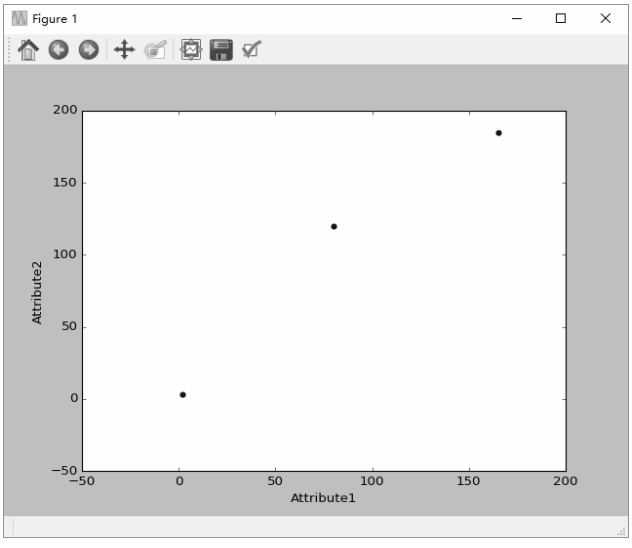

图 3-4　不同目标属性之间的关系

顺带说一句，本例中采用的数据较少，一般随着数据增加，属性之间会呈现一种正态分布，这一点可以请读者自行验证。

程序 3-5 可以得到两幅图，第一幅图请读者自行查看，建议与第一幅进行比较。

3.2.3 玩个大的数据集

现在开始玩个大的数据集。

对于大规模数据来说，涉及的目标比较多，并且属性特征值比较多，对其查看会是非常复杂的。因此，为了更好地理解和掌握大数据的分布，将其转化成可视性较强的图形显然是更好的做法。

前面对小数据集进行了图形化查阅，现在对现实中的数据进行处理。

数据来源于真实的信用贷款数据，从 50000 个数据记录中随机选取 200 个数据进行计算，而每个数据又有较多的属性值。大多数情况下，数据是以 csv 格式进行存储的，pandas 包同样提供了相关读取程序。具体代码见程序 3-6。

【程序 3-6】

```
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c://dataTest.csv")
dataFile = pd.read_csv(filePath,header=None, prefix="V")

dataRow1 = dataFile.iloc[100,1:300]
dataRow2 = dataFile.iloc[101,1:300]
plot.scatter(dataRow1, dataRow2)
plot.xlabel("Attribute1")
plot.ylabel("Attribute2")
plot.show()
```

从程序 3-6 可以看出，首先使用 filePath 创建了一个文件路径，用以建立数据地址。之后使用 pandas 自带的 read_csv 读取 csv 格式的文件。dataFile 是读取的数据集，之后使用 iloc 方法获取其中行的属性数据，scattle 是做出分散图的方法，对属性进行画图。最终结果如图 3-5 所示。

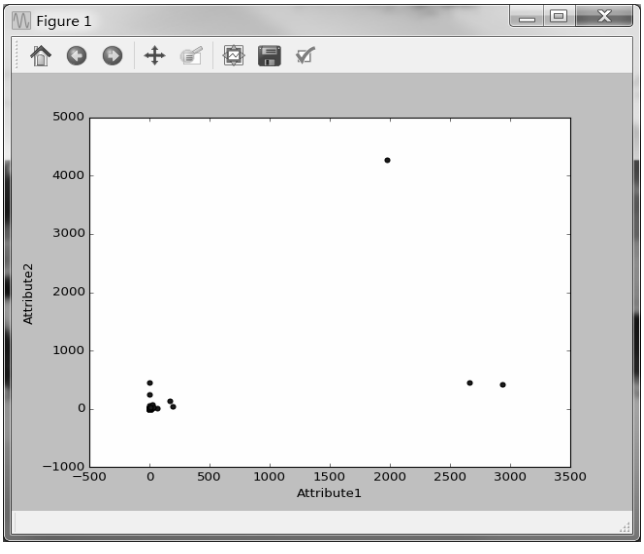

图 3-5　大数据集中不同目标属性之间的关系

这样可以看出，数据在 (0,0) 的位置有较大的集合，表明属性在此位置的偏离程度较少，而几个特定点是偏离程度较大的点。这可以帮助读者对离群值进行分析。

程序 3-6 出现了两幅图，第一个图请读者自行分析。

下面继续对数据集进行分析。程序 3-5 和程序 3-6 让读者看到了对数据同一行中不同的属性进行处理和现实的方法。如果是要对不同目标行的同一种属性进行分析，那么要如何做呢？请读者参阅程序 3-7。

【程序 3-7】

```
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c://dataTest.csv")
dataFile = pd.read_csv(filePath,header=None, prefix="V")

target = []
for i in range(200):
    if dataFile.iat[i,10] >= 7:
        target.append(1.0)
    else:
        target.append(0.0)

dataRow = dataFile.iloc[0:200,10]
plot.scatter(dataRow, target)
plot.xlabel("Attribute")
plot.ylabel("Target")
plot.show()
```

程序 3-7 对数据进行处理，提取了 200 行数据中的第 10 个属性，并对其进行判定，单纯的判定规则是根据均值对其区分的，之后计算判定结果，如图 3-6 所示。

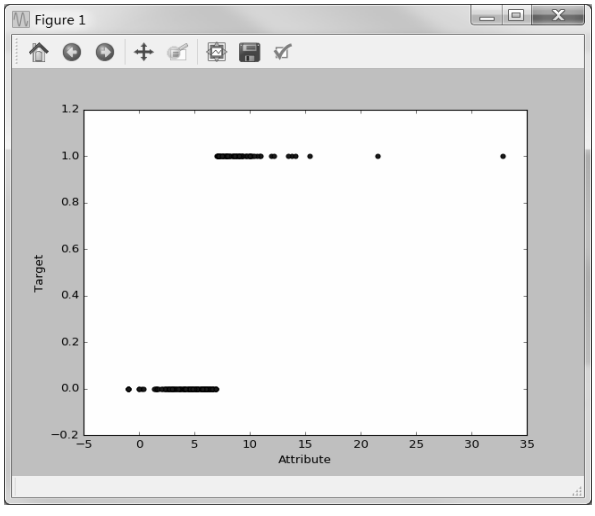

图 3-6　大数据集中不同行相同属性之间的关系

通过图 3-6 可以看出，属性被人为地分成两部分，数据集合的程度也显示了偏离程度。如果下一步需要对属性的离散情况进行反映，则应该使用程序 3-8。

【程序 3-8】

```
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c://dataTest.csv")
dataFile = pd.read_csv(filePath,header=None, prefix="V")

target = []
for i in range(200):
    if dataFile.iat[i,10] >= 7:
        target.append(1.0 + uniform(-0.3, 0.3))
    else:
        target.append(0.0 + uniform(-0.3, 0.3))
dataRow = dataFile.iloc[0:200,10]
plot.scatter(dataRow, target, alpha=0.5, s=100)
plot.xlabel("Attribute")
plot.ylabel("Target")
plot.show()
```

在此段程序中，离散的数据被人为地加入了离散变量，具体显示结果请读者自行完成。

读者可以对程序的属性做出诸多的抽取，并尝试使用更多的方法和变量进行处理。

3.3 深度学习理论方法——相似度计算

我们从上一节的内容可以得知，不同目标行之间由于其属性的不同，画出的散点图也是千差万别的，而对于机器学习来说则需要一个统一的度量，即需要对其相似度进行计算。

相似度的计算方法很多，这里选用最常用的两种，即欧几里得相似度和余弦相似度计算。如果读者对此不感兴趣，可以跳过本节继续学习。

3.3.1 基于欧几里得距离的相似度计算

欧几里得距离（Euclidean distance）是常用的计算距离的公式，用来表示三维空间中两个点的真实距离。

欧几里得相似度计算是一种基于用户之间直线距离的计算方式。在相似度计算中，不同的物品或者用户可以将其定义为不同的坐标点，而特定目标定位坐标原点。使用欧几里得距离计算两个点之间的绝对距离。欧几里得公式距离如公式 3-1 所示。

【公式 3-1】

$$d=\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$$

从公式 3-1 可以看到，作为计算结果的欧式值显示的是两点之间的直线距离，该值的大小表示两个物品或者用户差异性的大小，即用户的相似性如何。如果两个物品或者用户距离越大，那么相似度越小；反之，距离越小则相似度越大。

由于在欧几里得相似度计算中最终数值的大小与相似度成反比，因此在实际中常常使用欧几里得距离的倒数作为相似度值，即 1/d+1 作为近似值。

请参看一个常用的用户-物品推荐评分表的例子（见表 3-3）。

表 3-3　用户与物品评分对应表

	物品 1	物品 2	物品 3	物品 4
用户 1	1	1	3	1
用户 2	1	2	3	2
用户 3	2	2	1	1

表 3-3 是 3 个用户对物品的打分表，如果需要计算用户 1 和其他用户之间的相似度，通过欧几里得距离公式可以得出：

$$d_{12}=\sqrt{(1-1)^2+(1-2)^2+(3-3)^2+(1-2)^2}\approx 1.414$$

从结果可以得知，用户 1 和用户 2 的相似度为 1.414。用户 1 和用户 3 的相似度是：

$$d_{13}=\sqrt{(1-2)^2+(1-2)^2+(3-1)^2+(1-1)^2}\approx 2.287$$

d_{12} 分值小于 d_{13} 的分值，因此可以得到用户 2 更加相似于用户 1（距离越小，相似度越大）。

3.3.2　基于余弦角度的相似度计算

与欧几里得距离类似，余弦相似度也将特定目标（物品或者用户）作为坐标上的点，但不是坐标原点，是与特定的被计算目标进行夹角计算。具体如图 3-7 所示。

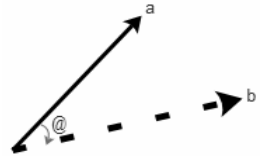

图 3-7　余弦相似度示例

从图 3-7 可以很明显地看出，两条直线分别从坐标原点出发，引出一定的角度。如果两个目标较为相似，那么其线段形成的夹角较小。如果两个用户不相近，那么两条射线形成的夹角较大。因此在使用余弦度量的相似度计算中，可以用夹角的大小来反映目标之间的相似性。余弦相似度的计算如公式 3-2 所示。

【公式 3-2】

$$\cos@ = \frac{\sum(x_i \times y_i)}{\sqrt{\sum x_i^2} \times \sqrt{\sum y_i^2}}$$

余弦值一般为[-1,1]，这个值的大小与余弦夹角的大小成正比。如果用余弦相似度计算表 3-3 中用户 1 和用户 2 之间的相似性，结果如下：

$$d_{12} = \frac{1\times1+1\times2+3\times3+1\times2}{\sqrt{1^2+1^2+3^2+1^2}\times\sqrt{1^2+2^2+3^2+2^2}} = \frac{14}{\sqrt{12}\times\sqrt{18}} \approx 0.789$$

用户 1 和用户 3 的相似性如下：

$$d_{13} = \frac{1\times2+1\times2+3\times1+1\times1}{\sqrt{1^2+1^2+3^2+1^2}\times\sqrt{2^2+2^2+1^2+1^2}} = \frac{8}{\sqrt{12}\times\sqrt{10}} \approx 0.344$$

从计算可得，相对于用户 3，用户 2 与用户 1 更为相似（两个目标越相似，其线段形成的夹角越小）。

3.3.3 欧几里得相似度与余弦相似度的比较

欧几里得相似度是以目标绝对距离作为衡量的标准，而余弦相似度是以目标差异的大小作为衡量标准的，其表述如图 3-8 所示。

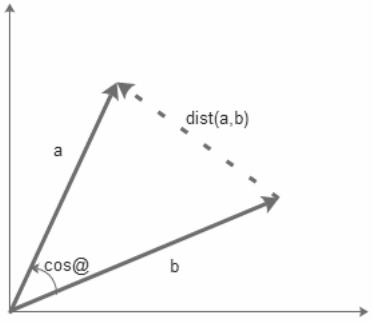

图 3-8　欧几里得相似度与余弦相似度

从图 3-8 中可以看到，欧几里得相似度注重目标之间的差异，与目标在空间中的位置直接相关。而余弦相似度是不同目标在空间中的夹角，更加表现在前进趋势上的差异。

欧几里得相似度和余弦相似度具有不同的计算方法和描述特征。一般来说欧几里得相似度

用以表现不同目标的绝对差异性，从而分析目标之间的相似度与差异情况。而余弦相似度更多的是对目标从方向趋势上区分，对特定坐标数字不敏感。

举例来说，两个目标在不同的两个用户之间的评分分别是（1,1）和（5,5），这两个评分在表述上是一样，但是在分析用户相似度时，更多的是使用欧几里得相似度而不是余弦相似度。余弦相似度更好地区分了用户分离状态。

3.4 数据的统计学可视化展示

在 3.3 节中，读者对数据（特别是大数据）的处理有了一个基本的认识，通过数据的可视化处理，对数据的基本属性和分布有了较为直观的理解。但是对于机器学习来说，这里的数据需要更多的分析处理，需要用到更为精准和科学的统计学分析。

本节将使用统计学分析对数据进行处理。

3.4.1 数据的四分位

四分位数（Quartile）是统计学中分位数的一种，即把所有数据由小到大排列并分成四等份，处于三个分割点位置的数据就是四分位数。

- 第一四分位数（Q1）又称“下四分位数”，等于该样本中所有数据由小到大排列后第 25%的数据。
- 第二四分位数（Q2）又称“中位数”，等于该样本中所有数据由小到大排列后第 50%数据。
- 第三四分位数（Q3）又称“上四分位数”，等于该样本中所有数据由小到大排列后第 75%的数据。

第三四分位数与第一四分位数的差距又称四分位距（InterQuartile Range，IQR）。

首先要确定四分位数的位置，若用 n 表示项数，则分位数的位置分别为：

- Q1 的位置= $(n+1) \times 0.25$
- Q2 的位置= $(n+1) \times 0.5$
- Q3 的位置= $(n+1) \times 0.75$

使用图形表示，如图 3-9 所示。

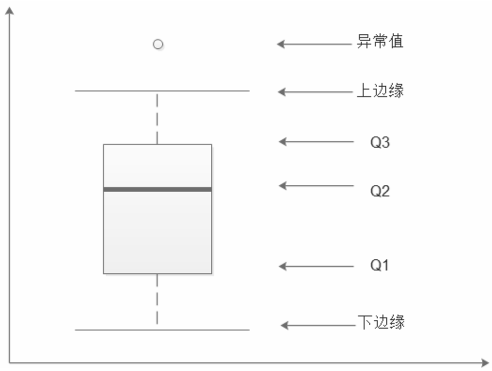

图 3-9　四分位的计算

从图 3-9 可以看到，四分位在图形中根据 Q1 和 Q3 的位置绘制了一个箱体结构，即根据一组数据 5 个特征绘制的一个箱子和两条线段的图形。这种直观的箱线图反映出一组数据的特征分布，还显示了数据的最小值、中位数和最大值。

3.4.2　数据的四分位示例

现在回到数据处理中来，这里依旧使用 3.2.3 小节中的数据集进行数据处理。

首先回顾一下本例中的数据集。本数据集来源是真实世界中某借贷机构对申请小额贷款的贷款人的背景调查，目的是根据不同借款人的条件，分析判断借款人能否按时归还贷款。一般来说，借款人能否按时归还贷款是所有借贷最为头疼的问题，其中的影响因素很多，判别相对麻烦，判断错误后果也较为严重。通过机器学习，可以较为轻松地将其转化成一个回归分类问题。

数据集中的数据如图 3-10 所示。


```
20001,6.15,7.06,5.24,2.61,0,4.36,0,5.76,3.83,6.94,5.86,0,9.15,6.09,1.02,3.47,4.52,11,5.35,7.5
7,0.22,12.69,0,3.55,4.58,8.02,6.59,5.16,7.45,13.04,0,0,4.35,0.25,0,7.17,5.27,4.48,0.02,0.48,6
.67,6.29,3.58,8.82,0,1.6,4.81,0.33,0.95,1.36,4.89,4.72,5.51,3.87,2.02,3.31,9.02,5.73,8.02,0,1
.72,0.86,0,0,4.35,2.17,4.35,0,0,9.02,5.72,6.82,0.07,1.05,6.67,0.47,0,1.58,2.33,0.24,8.2,2.57,
3.47,3.52,0.51,1.55,0,7.95,4.25,2.71,0,9.17,5.16,4.58,9.17,0.29,0,2.17,4.35,0,0,4.35,0,0,10.8
7,9.03,7.51,0,4.71,6.29,0,7.25,14.09,7.57,2.12,0,6.12,2.54,8.53,4.43,8.91,6.81,0.6,7.48,5.52,
2.42,0.64,5.63,3.29,0.03,7.33,4.55,0,5.73,3.72,0.57,11.17,2.01,0.29,6.52,15.22,4.35,23.91,0,5
.99,9.8,5.04,7.35,7.67,2.24,7.35,0.64,3.19,0.3,6.59,6.89,1.8,2.4,4.49,0,5.88,8.09,4.41,1.47,5
.15,0,8.18,6.85,4.28,0.1,9.27,7.67,4.47,0,6.39,5.09,9.28,5.39,5.99,5.69,6.89,0.6,8.82,5.88,8.
09,0.19,0,8.66,4.76,7.14,8.85,7.8,3.9,0,4.28,2.38,0.29,7.61,4.15,5.75,0,0,0,6.89,4.49,6.29,0.
9,4.49,0.6,6.89,0.74,6.62,7.35,5.15,1.47,2.21,0.74,2.94,3.68,1.05,0,0.95,1.9,0,0,8.31,5.75,11
.5,2.88,2.88,2.56,0.6,0.3,2.1,5.09,2.21,8.09,0.74,2.21,8.09,0,0,0,0,0,764,0,0,0,0,0,0,0,0,5,0
,701,0,0,0,0,0,0,0,0,0,1,0,121,0,0,4,-1,0,0,0,0,0,0,0,0,0,0,0,-
1,49,0,0,1,1244.33,0,0,11.33,0,0,1,-1,-
1,0,0,12,1267,5.78,77,1,1,1,1,1,1,1,1,0,0,0,1,1,1,5,0.38,1,5,1,1,1,35.4025,116.58031,0,13,0,1
,0,182,51,200,123,598,379,358,0.94,0.88,1.48,22.74,11742.05,10.42,5.33,9.23,78.89,11.71,4.2,1
223.99,65.53,1.5,107.99,44.21,6375,34.68,5367.05,7.66,9.61,30.59,226,23.11,8.51,0.61,358,283,
4,6,5,9,0,0,0,0,40,12,641,18,49,0,0,0,0,0,4,48.12,407,14,0,0,3,0,405,36,0,10,0,0,3,24,387,3
71,16,26.87,21.24,5,1000,29,14,2,4,2,1,0,0,0,0,0,4,0,1,0.16,15.76,42,3.29,0.36,27.21,0.16,6.2
7,6.12,140,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-
1,0,4,27,0,3,50,0,1,7,0,3,0,0.23,11,0,0,0,0,97,88,252,82,0,27,92,2867,0,185,334,12500,0,0,0,2
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,6,1,1318,0,0,0,0,0,0,0,0,4,370,0,0,0,0,0,0,0,0,0,16,
0,1,0,0,0,0,0,0,0,0,0,0,49,-1,19,159,157,3,16,-
1,91,37,4,6074,754,1732,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,11,0,21,21,5,84,1,0,0,0,0,1,9,6
70,1,0,0,0,0,0,0,0,0,0,0,0,0,122,0,0,0,0,51,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2281,0,
0,0,0,0,0,0,0,0,0,0,42,-1,-1,-1,-1,-
1,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,174,0,0,0,4,1,0,6,5645,1212,1060,0,37,0
,0,0,0,0,0,0,0,43,0,2,0,0,0,0,0,0,0,0,0,0,0,0,13,0,0,0,0,0,0,0,619,115,0,0,0,0,0,0,93,-
1,37,0,1,115,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,16,1358,90,2,0,3494,0,244,
0,17,17,0,0,0,0,2,101,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,5,0,0,0,0,0,0,0,0
,0,0,0,-1,-
```


图 3-10　小贷（小额贷款）数据集

这个数据集的形式是每一行为一个单独的目标行，使用逗号分隔不同的属性；每一列是不同的属性特征。不同列的含义在现实中至关重要，这里不做解释。具体代码如程序 3-9 所示。

【程序 3-9】

```
from pylab import *
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c://dataTest.csv")
dataFile = pd.read_csv(filePath,header=None, prefix="V")

print(dataFile.head())
print((dataFile.tail())

summary = dataFile.describe()
print(summary)

array = dataFile.iloc[:,10:16].values
boxplot(array)
plot.xlabel("Attribute")
plot.ylabel(("Score"))
show()
```

首先来看数据的结果：

```
     V0    V1    V2    V3    V4    V5    V6    V7    V8    V9  ...    V1129  \
0  20001  6.15  7.06  5.24  2.61  0.00  4.36  0.00  5.76  3.83  ...        7
1  20002  6.53  6.15  9.85  4.03  0.10  1.32  0.69  6.24  7.06  ...        6
2  20003  8.22  3.23  1.69  0.41  0.02  2.89  0.13 10.05  8.76  ...        1
3  20004  6.79  4.99  1.50  2.85  5.53  1.89  5.41  6.79  6.11  ...        3
4  20005 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00 -1.00  ...     7

   V1130  V1131  V1132  V1133  V1134  V1135  V1136  V1137  V1138
0      6      1      2      5      7      3      6      8     12
1      7     15      2      6      7      1      8      1     24
2      8      3      1      1      8      8      1      7      6
3      6     20      1      6      8      1      6      5     12
4      8      1      1      8      8      1      8      8      1

[5 rows x 1139 columns]
       V0    V1    V2    V3    V4    V5    V6    V7    V8     V9  ...    \
196  20197  3.59  5.63  6.21  5.24  1.88  1.65  4.74  3.73   7.19  ...
197  20198  7.27  5.31  9.35  2.77  0.00  1.37  0.74  5.77   4.64  ...
198  20199  6.18  5.05  6.43  6.05  1.93  2.58  3.75  7.32   4.19  ...
199  20200  6.12  7.45  1.05  1.03  0.16  1.44  0.32  6.49  10.79  ...
```

```
200  20201  5.60  6.29  6.11  2.64  0.11  4.08  2.44  7.04   5.60  ...

     V1129  V1130  V1131  V1132  V1133  V1134  V1135  V1136  V1137  V1138
196      6      6      1      1      6      8      9      8      4     28
197      7      1      1      1      1      8     24      7      8     14
198      3      7      1      2      7      7      3      3      7      4
199      7      8      1      2      4      7      6      8      7     12
200      7      7      3      1      7      8      1      2      7     23

[5 rows x 1139 columns]
              V0         V1         V2         V3         V4  \
count   201.000000  201.000000  201.000000  201.000000  201.000000
mean  20101.000000    5.266219    6.447015    6.156020    3.319303
std      58.167861    2.273933    2.443789    2.967566    3.134570
min   20001.000000   -1.000000   -1.000000   -1.000000   -1.000000
25%   20051.000000    4.130000    5.190000    4.660000    1.200000
50%   20101.000000    5.240000    6.410000    6.000000    2.830000
75%   20151.000000    6.590000    7.790000    7.640000    4.570000
max   20201.000000   13.150000   13.960000   16.620000   28.440000

             V5         V6         V7         V8         V9     ...      \
count  201.000000  201.000000  201.000000  201.000000  201.000000     ...
mean     0.907662    2.680149    2.649254    5.149055    5.532736     ...
std      1.360489    2.292231    2.912611    2.965096    2.763270     ...
min     -1.000000   -1.000000   -1.000000   -1.000000   -1.000000     ...
25%      0.020000    1.270000    0.320000    3.260000    3.720000     ...
50%      0.300000    2.030000    1.870000    4.870000    5.540000     ...
75%      1.390000    3.710000    4.140000    6.760000    7.400000     ...
max      8.480000   12.970000   18.850000   15.520000   13.490000     ...

          V1129       V1130       V1131       V1132       V1133       V1134  \
count  201.000000  201.000000  201.000000  201.000000  201.000000  201.000000
mean     6.054726    6.039801    7.756219    1.353234    4.830846    7.731343
std      1.934422    2.314824    9.145232    0.836422    2.161306    0.444368
min      1.000000    1.000000    1.000000    1.000000    1.000000    7.000000
25%      6.000000    5.000000    1.000000    1.000000    3.000000    7.000000
50%      7.000000    7.000000    1.000000    1.000000    6.000000    8.000000
75%      7.000000    8.000000   15.000000    2.000000    7.000000    8.000000
max      8.000000    8.000000   35.000000    7.000000    8.000000    8.000000

          V1135       V1136       V1137       V1138
count  201.000000  201.000000  201.000000  201.000000
mean    10.960199    5.631841    5.572139   16.776119
```

```
std      9.851315    2.510733    2.517145    8.507916
min      1.000000    1.000000    1.000000    1.000000
25%      3.000000    3.000000    4.000000   11.000000
50%      8.000000    7.000000    7.000000   17.000000
75%     18.000000    8.000000    7.000000   23.000000
max     36.000000    8.000000    8.000000   33.000000
```

这一部分打印出来的是计算后的数据头和尾部，为了节省空间，我们只选择了前 6 个和尾部的 6 个数据。第一列是数据的编号，对数据目标行进行区分，其后是每个不同的目标行的属性。

dataFile.describe()方法对数据进行统计学估计，count、mean、std、min 分别求得每列数据的计数、均值、方差以及最小值等。最后的几个百分比是求得四分位的数据，具体图形如图 3-11 所示。这里的 6 列数据是从整个数据集中随机选取的 6 列做的数据描述结果，即对随机选择的 6 列数据做的四分位分布计算；竖的每个四分位图是随机选择的每个数据列做的数据可视化描述，主要是用来展示离群点。

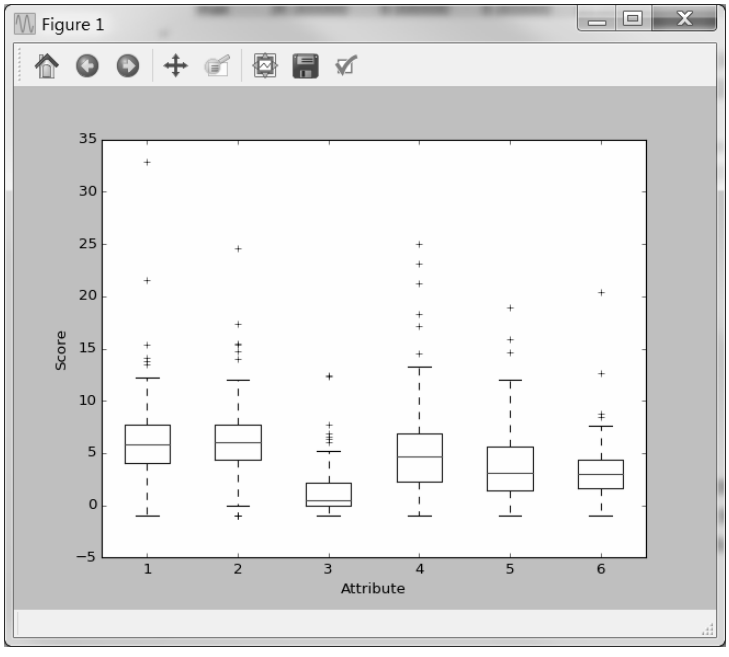

图 3-11　小贷数据集的四分位显示

在程序中选择了第 11~16 列的数据作为分析数据集，从图 3-11 可以看出，不同的数据列做出的箱体四分位图也是不同的。部分不在点框体内的数据被称为离群值，一般被视作特异点加以处理。

读者可以多选择不同的目标行和属性点进行分析。

四分位图是一个以更好、更直观的方式来识别数据中异常值的方法，与数据处理的其他方

式相比，能够更有效地让分析人员判断离群值。

3.4.3 数据的标准化

继续对数据进行分析。读者在进行数据选择的时候可能会遇到某一列的数值过大或者过小的问题，即数据的显示若超出其他数据部分较大时则会产生数据图形失真的问题，如图 3-12 所示。

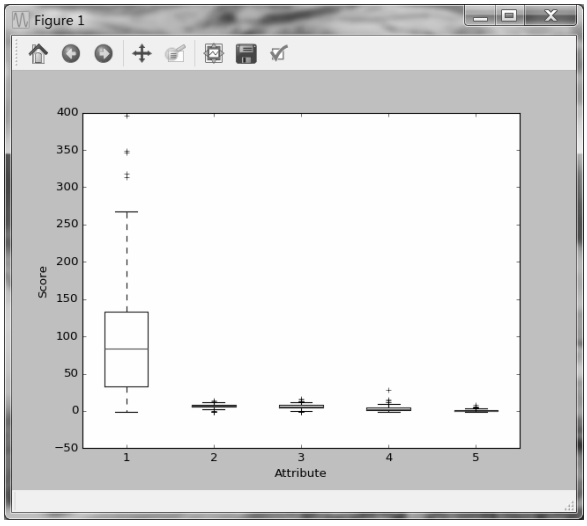

图 3-12　数据超预期的四分位图

对其来说，需要一个能够处理数据，使其具有共同计算均值的方法，即数据的标准化处理。

顾名思义，数据的标准化是将数据根据自身一定比例进行处理，落入一个小的特定区间，一般为(-1,1)。这样做的目的是去除数据的单位限制，将其转化为无量纲的纯数值，使得不同单位或量级的指标能够进行比较和加权，其中最常用的就是 0-1 标准化和 Z-score 标准化。

1. 0-1 标准化（0-1 normalization）

0-1 标准化也叫离差标准化，是对原始数据的线性变换，使结果落到[0,1]区间，转换函数如下：

$$X = \frac{X - \min}{\max - \min}$$

其中，max 为样本数据的最大值，min 为样本数据的最小值。这种方法有一个缺陷，就是当有新数据加入时，可能导致 max 和 min 的变化，需要重新定义。

2. Z-score 标准化（zero-mean normalization）

Z-score 标准化也叫标准差标准化，经过处理的数据符合标准正态分布，即均值为 0，标准差为 1，其转化函数为：

$$X = \frac{x - \mu}{\sigma}$$

其中，μ 为所有样本数据的均值，σ 为所有样本数据的标准差。

一般情况下，通过数据的标准化处理后，数据最终落在(-1,1)的概率为 99.7%，在(-1,1)之外的数据被设置成-1 和 1，以便处理。

【程序 3-10】

```
from pylab import *
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c://dataTest.csv")
dataFile = pd.read_csv(filePath,header=None, prefix="V")

summary = dataFile.describe()
dataFileNormalized = dataFile.iloc[:,1:6]
for i in range(5):
    mean = summary.iloc[1, i]
    sd = summary.iloc[2, i]

dataFileNormalized.iloc[:,i:(i + 1)] = (dataFileNormalized.iloc[:,i:(i + 1)]
- mean) / sd
array = dataFileNormalized.values
boxplot(array)
plot.xlabel("Attribute")
plot.ylabel(("Score"))
show()
```

从代码中可以看到数据被处理为标准差标准化的方法。dataFileNormalized 被重新计算并定义，大数值被人为限定在(-1,1)，请读者自行运行验证。

程序 3-10 中所使用的数据被人为修改，请读者自行修改验证，这里作者不再进行演示。此外，读者可以对数据进行处理，验证更多的标准化方法。

3.4.4　数据的平行化处理

从 3.4.2 小节可以看到，对于每种单独的数据属性来说，可以通过数据的四分位法进行处理、查找和寻找离群值，从而对其进行分析处理。

对于属性之间的横向比较，即每个目标行属性之间的比较，使用四分位法则较难判断。为了描述和表现每一个不同目标行之间的数据差异和不同，需要另外一种处理和展示方法。

平行坐标（Parallel Coordinates）是一种通常的可视化方法，用于对高维几何和多元数据的可视化。平行坐标为了表示在高维空间的一个点集，在 N 条平行线的背景下（一般这 N 条线都竖直且等距），一个在高维空间的点被表示为一条拐点在 N 条平行坐标轴的折线，在第 K 个坐标轴上的位置就表示这个点在第 K 维的值。

平行坐标是信息可视化的一种重要技术。为了克服传统的笛卡儿直角坐标系容易耗尽空间、难以表达三维以上数据的问题，平行坐标将高维数据的各个变量用一系列相互平行的坐标轴表示，变量值对应轴上位置。为了反映变化趋势和各个变量间的相互关系，往往将描述不同变量的各点连接成折线。所以平行坐标图的实质是将欧式空间的一个点 Xi(xi1,xi2,...,xim) 映射到二维平面上的一条曲线。

平行坐标图可以表示超高维数据。平行坐标的一个显著优点是具有良好的数学基础，其射影几何解释和对偶特性使它很适合用于可视化数据分析。

【程序 3-11】

```
from pylab import *
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c://dataTest.csv")
dataFile = pd.read_csv(filePath,header=None, prefix="V")

summary = dataFile.describe()
minRings = -1
maxRings = 99
nrows = 10
for i in range(nrows):
    dataRow = dataFile.iloc[i,1:10]
    labelColor = (dataFile.iloc[i,10] - minRings) / (maxRings - minRings)
    dataRow.plot(color=plot.cm.RdYlBu(labelColor), alpha=0.5)
plot.xlabel("Attribute")
plot.ylabel("Score")
show()
```

从代码可以看出，本例首先计算总体的统计量，之后设置计算的最大值和最小值（本例中设置-1 为最小值、99 为最大值）。为了计算简便，选择了前 10 行作为目标行数，使用 for 循环对数据进行训练。

最终图形结果如图 3-13 所示。

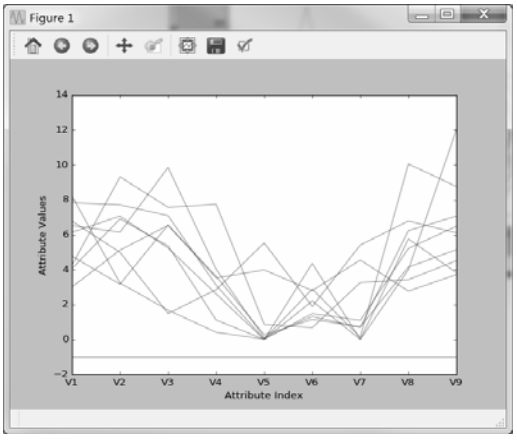

图 3-13　属性的图形化展示

从图 3-13 中可以看出，由于属性不同而画出了 10 条不同的曲线。这些曲线是根据不同的属性得出的不同的运行轨迹。

可以选择不同的目标行和不同的属性进行验证，观察更多的数据展示结果有何不同。

3.4.5　热点图-属性相关性检测

前面小节中，作者对数据集中数据的属性分别进行了横向和纵向的比较，现在请读者换一种思路，如果要对数据属性之间的相关性进行检测，那该怎么办？

热点图是一种判断属性相关性的常用方法，根据不同目标行数据对应的数据相关性进行检测。程序 3-12 展示了对数据相关性进行检测的方法，根据不同数据之间的相关性做出图形。

【程序 3-12】

```
from pylab import *
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c://dataTest.csv")
dataFile = pd.read_csv(filePath,header=None, prefix="V")

summary = dataFile.describe()
corMat = DataFrame(dataFile.iloc[1:20,1:20].corr())

plot.pcolor(corMat)
plot.show()
```

最终结果如图 3-14 所示。

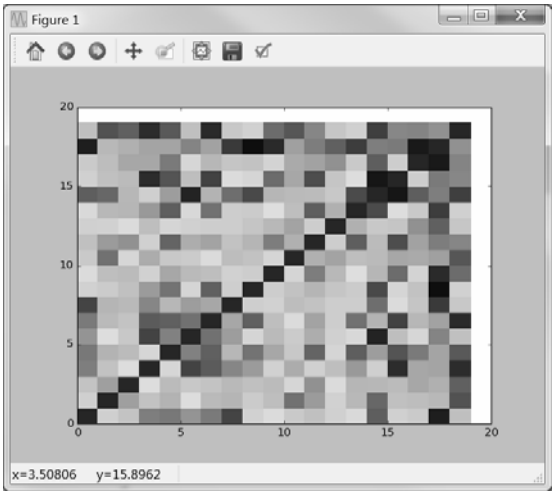

图 3-14　属性之间的相关性图

不同颜色之间显示了不同的相关性，彩色的深浅（参看下载报中的相关文件）显示了相关

性的强弱程度。读者可以通过打印相关系数来直观地显示数据：

```
print(corMat)
```

代码中选择了前20行中的前20列数据属性进行计算，读者可以对其进行更多的验证和显示处理。

3.5 Python 数据分析与可视化实战——某地降水的关系处理

上面的章节对数据属性间的处理做了一个大致的介绍，本节将使用这个处理方法解决一个实际问题。农业灌溉用水主要来自于天然降水和地下水。随着中原经济区的发展和城镇化水平的提高，城市用水日趋紧张。下面提供河南省降水量的变化及分布规律，为合理调度和利用水资源提供决策。数据集名为 rain.csv，记录了从 2000 年开始到 2011 年之间的每月降水量数据。本节将以降水量进行统计计算，找出规律并进行分析。

3.5.1 不同年份的相同月份统计

对于不同年份，每月的降水量也是不同的，一般情况下，降水量会随着春夏秋冬的交替呈现不同的状态，横向是一个过程。对于不同的年份来说，每月的降水量应该在一个范围内浮动，而不应偏离均值太大。

【程序 3-13】

```
from pylab import *
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c://rain.csv")
dataFile = pd.read_csv(filePath)

summary = dataFile.describe()
print(summary)

array = dataFile.iloc[:,1:13].values
boxplot(array)
plot.xlabel("month")
plot.ylabel(("rain"))
show()
```

打印结果如下所示。

```
                0           1           2           3           4
count    12.000000   12.000000   12.000000   12.000000   12.000000
mean   2005.500000  121.083333   67.833333  102.916667  263.416667
```

```
std      3.605551  103.021144   72.148626  137.993714  246.690258
min   2000.000000    0.000000    0.000000    0.000000   70.000000
25%   2002.750000   17.750000    9.750000    3.000000  136.250000
50%   2005.500000  125.000000   39.500000   51.500000  155.000000
75%   2008.250000  204.500000  123.250000  150.000000  232.500000
max   2011.000000  295.000000  192.000000  437.000000  833.000000

                5           6           7           8           9
count   12.000000   12.000000   12.000000   12.000000   12.000000
mean   1134.583333  2365.666667  2529.000000  1875.500000  1992.416667
std     618.225240   705.323180  1120.231226   603.135821   670.834414
min     218.000000   766.000000   865.000000   746.000000   621.000000
25%     685.500000  2117.000000  1770.250000  1723.500000  1630.000000
50%     951.500000  2440.500000  2023.500000  1943.500000  1961.000000
75%    1599.000000  2723.750000  3603.000000  2321.750000  2231.750000
max    2134.000000  3375.000000  4163.000000  2508.000000  3097.000000

               10          11          12
count   12.000000   12.000000   12.000000
mean   1219.250000  159.333333   38.333333
std     743.534938  124.611639   34.494620
min     328.000000    0.000000    0.000000
25%     612.250000   64.000000   18.750000
50%    1208.500000  123.000000   25.500000
75%    1672.250000  278.250000   46.250000
max    2561.000000  357.000000  100.000000
```

从打印结果可以看到，程序对平均每个月份的降水量进行了计算，获得了其偏移值、均值以及均方差的大小。

通过四分位的计算，可以获得一个波动范围，具体结果如图 3-15 所示。

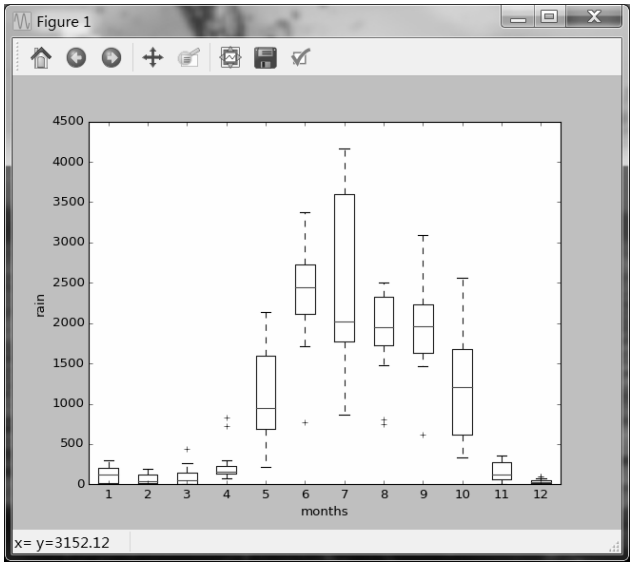

图 3-15　降水量的四分位图

从图 3-15 中可以直观地看到，不同月份之间的降水量有很大的差距，1~4 月降水量明显较少，从 5 月份开始降水量有明显增多，到 7 月份达到顶峰后回落，之后逐渐减少，12 月达到最低的降水量。

同时，有几个月份的降水量有明显的偏移，即出现离群值，可能跟年度情况有关，需要继续进行分析。

3.5.2 不同月份之间的增减程度比较

正常情况下，每年降水量都呈现一个平稳的增长或者减少的过程，其下降的坡度（趋势线）应该是一样的。程序 3-14 展示了这种趋势。

【程序 3-14】

```
from pylab import *
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c://rain.csv")
dataFile = pd.read_csv(filePath)

summary = dataFile.describe()
minRings = -1
maxRings = 99
nrows = 11
for i in range(nrows):
    dataRow = dataFile.iloc[i,1:13]
    labelColor = (dataFile.iloc[i,12] - minRings) / (maxRings - minRings)
    dataRow.plot(color=plot.cm.RdYlBu(labelColor), alpha=0.5)
plot.xlabel("Attribute")
plot.ylabel(("Score"))
show()
```

最终打印结果如图 3-16 所示。

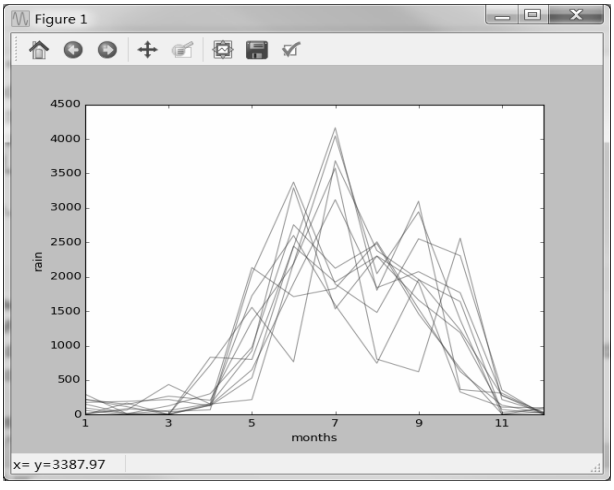

图 3-16　降水量的趋势图

从图中可以明显地看到，降水的月份并不是一个规律的上涨或下跌，而是呈现一个不规则的浮动状态，增加最快的为 6~7 月，下降最快的为 7~8 月，之后有一个明显的回升过程。

3.5.3　每月降水是否相关

每月的降水量理论上来说应该是相互独立的，即每月的降水量和其他月份没有关系。但是实际上是这样的吗？

【程序 3-15】

```
from pylab import *
import pandas as pd
import matplotlib.pyplot as plot
filePath = ("c:// rain.csv")
dataFile = pd.read_csv(filePath)

summary = dataFile.describe()
corMat = DataFrame(dataFile.iloc[1:20,1:20].corr())

plot.pcolor(corMat)
plot.show()
```

通过计算，最终结果如图 3-17 所示。

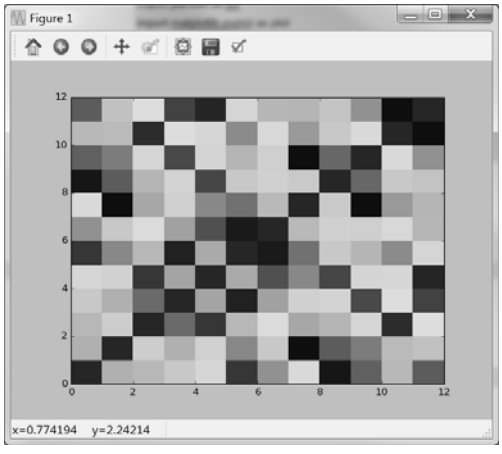

图 3-17　月份之间的相关性显示

从图 3-17 可以看出，颜色分布比较平均，表示并没有太大的相关性，因此可以认为每月的降水是独立行为，即每个月的降水量和其他月份没有关系。

3.6 本章小结

上面的章节已经对数据属性间的处理做了一个大致的介绍，并使用了数据分析的方法对其进行分析和整理。本章从直观的观察开始，深入介绍和研究了数据集和分析工具，了解了使用 Python 类库进行数据分析的基本方法。数据分析从最基本的矩阵转换开始，直到对数据集特征值的分析和处理，对掌握和了解简单的数据分析打下基础。

使用相应的类库进行深度学习程序设计是本章的重点，也是希望读者掌握的内容。再重复一次，请读者在程序设计时尽量使用已有的 Python 去进行程序设计。在数据的可视化展示过程中，作者通过多种数据图形向读者演示了可以通过使用不同的类库非常直观地进行数据分析。希望本章中提供的不同研究方法和程序设计思路能够帮助读者掌握基本数据集描述性和统计值之间的关系，以有利于对数据的掌握。

本章是机器学习的基础，虽然内容简单，但是非常重要，希望读者能够使用不同的数据集进行处理并演示得到更多的结果。