第 1 章

◂ 星星之火 ▸

当笔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》（图 1-1）激起了笔者对机器人的浓厚兴趣。一句“汽车人，变形，出发！”不光是孩子，甚至于陪同观看的大人们也会被那些懂幽默会调侃，充满正义、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喜爱，人类用对未来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。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还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，这条路还很长很长。

1.1　计算机视觉与深度学习

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追求的目标，这其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，这也是激励笔者或者说激励整个为之奋斗的计算机工作者的重要力量。虽然目

前计算机并不能达到动画片中变形金刚十分之一的能力，但是进步是不会停息的。

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（图 1-2）。

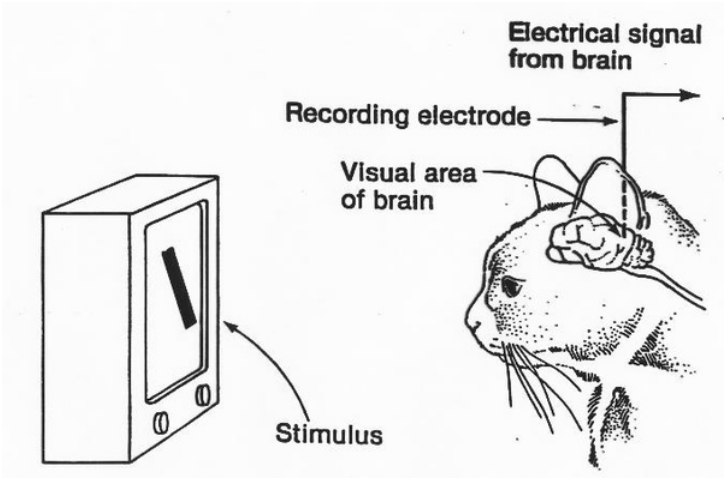

图 1-2 脑部连入电极的猫

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。

从 Torsten Wiesel 和 David Hubel 做这个有名的脑部视觉神经实验之后，视觉神经科学（图 1-3）正式被人们确立。目前为止，关于视觉神经的几个广为接受的观点是：

- 脑对视觉信息的处理是分层级的，低级脑区可能处理对边度、边缘什么的，高级脑区处理更抽象的，比如人脸、房子、运动的物体之类的。信息被一层一层抽提出来，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反馈投射。
- 信息的处理普遍受到自上而下和自下而上的注意的调控。

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时，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特征演变为目标识别。计算机如果拥有这么一个“脑皮层”，能够对信号进行转换，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，人类视觉的秘密逐渐正在被揭开，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（图 1-4）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区别，即同一个物体相似而不同的物体之间有很大的差别。

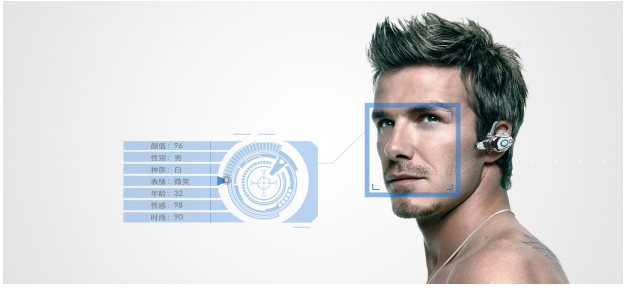

图 1-4　计算机视觉

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 年代就萌芽了，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，无法全面发展和验证。

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

受这些前人研究的启发，“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（图 1-5）。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。

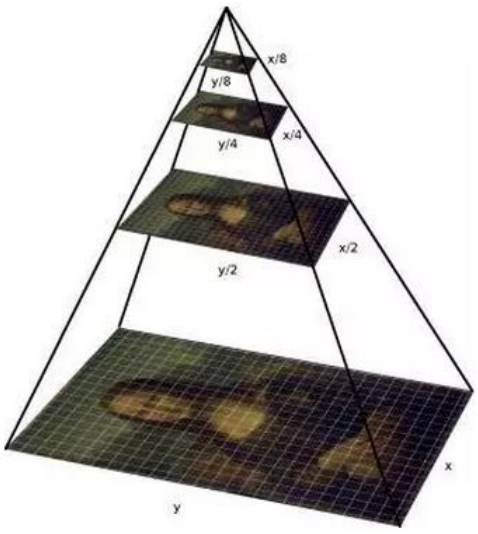

图 1-5　分层的视觉处理算法

因此，CNN，特别是基本原理和基础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，深度学习应用于解决计算机视觉的还有其他优点，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，需要针对不同的物体定制不同的算法。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（图 1-6）。
- 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，针对这种计算优化，所有深度学习算法都可以提升性能。

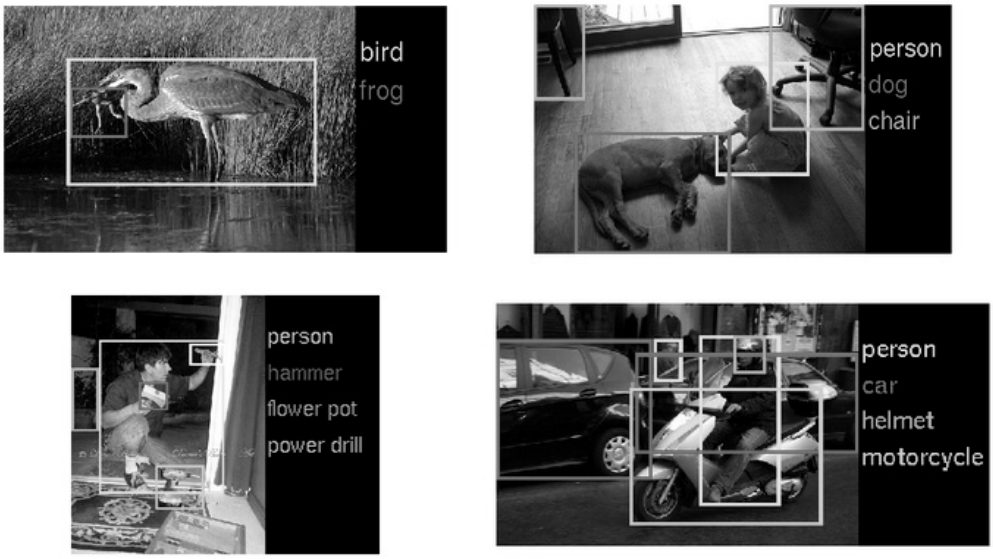

图 1-6　计算机视觉辨识图片

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会计算机如何去“看”的学科，更进一步说明就是使用机器替代生物眼睛去对目标进行识别，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。这些领域亟需相关研究人员学习其中的基础，理解并找出规律，从而揭示那些我们以前不曾注意的细节。

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（图 1-7）。

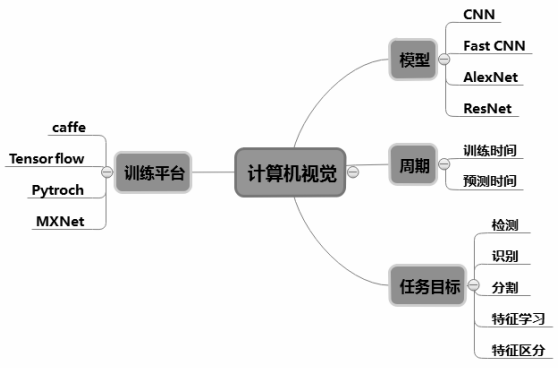

图 1-7　计算机视觉结构图

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，其中应用较为广泛的是 AlexNet、VGGNet、GoogleNet、ResNet 等。

除此之外，速度和周期也是需要考虑的非常重要的因素，如何使得训练速度更快、如何使用模型能够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，主要包括检测、识别、分割、特征点定位、序列学习五个大的任务，可以说任何计算机视觉的具体应用都是由这五个任务之一或者由其组合而成。

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什么。”这是计算机视觉作为一门学科被提出时就做出的目标。如今还有大量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，之后再拿出一张猫的图片，让一个人去辨识（图 1-8）。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。把这种带有标注的图片送到神经网络模型中去学习则称为“监督学习”。

图 1-8　猫 VS 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”，还有更多更重大的内容急待解决，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的问题，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（图 1-9），超市电子秤在称重的同时就能辨别出蔬菜的种类；门禁系统能分辨出带着礼物的朋友，或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任何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来预测天气，监测交通来调度车辆，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。

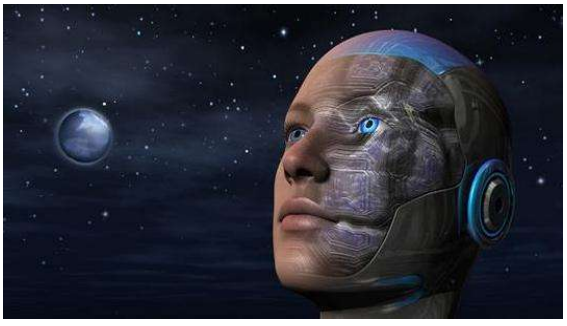

图 1-9　计算机视觉的未来

这些，似乎并不遥远。

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，更多数据集的收集以及能够设计得更深的网络计算，使得深度学习的普及性和应用性都有了非常大的发展。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 在自身宣传和推广中使用的口号，做深度学习也是这样。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，减少变量申明，随用随写，无须对内存进行释放，这些都极大地帮助 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 这三个主要的类库，这些开源的算法类库在后面的程序编写过程中会起到很大的作用。

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，很多第三方提供了集成大量科学计算类库的 Python 标准安装包，而最常用的是 Anaconda。

Anaconda 的作用就是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好。因此在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。

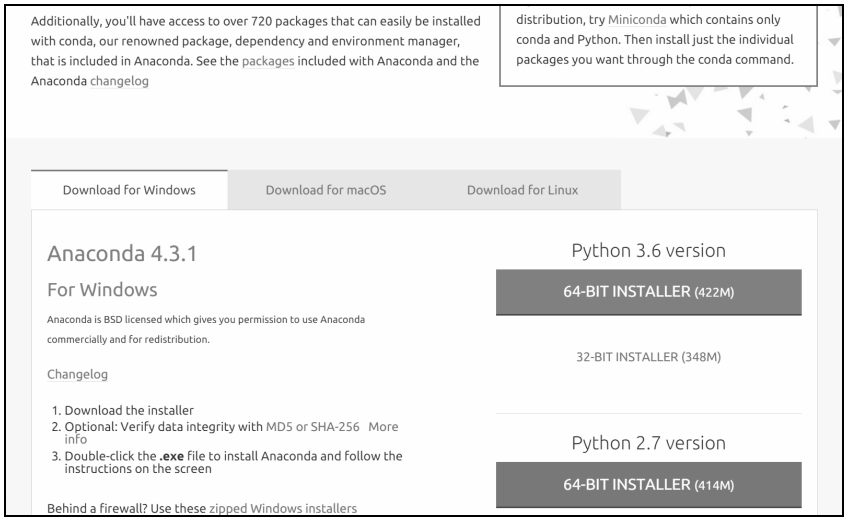

图 2-1　Anaconda 下载页面

目前提供的是 Anaconda 4.3.1 版本下载，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。

这里笔者选择的是 Windows 版本，单击运行即可安装，与普通软件一样。安装完成以后，出现程序面板，目录如图 2-2 所示。

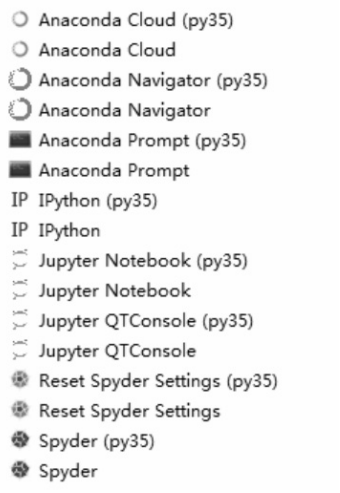

图 2-2　Anaconda 安装目录

2. 第二步：打开控制台

之后依次单击开始→所有程序→Anaconda→Anaconda Promp，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。利用这个命令可以执行一些基本操作。

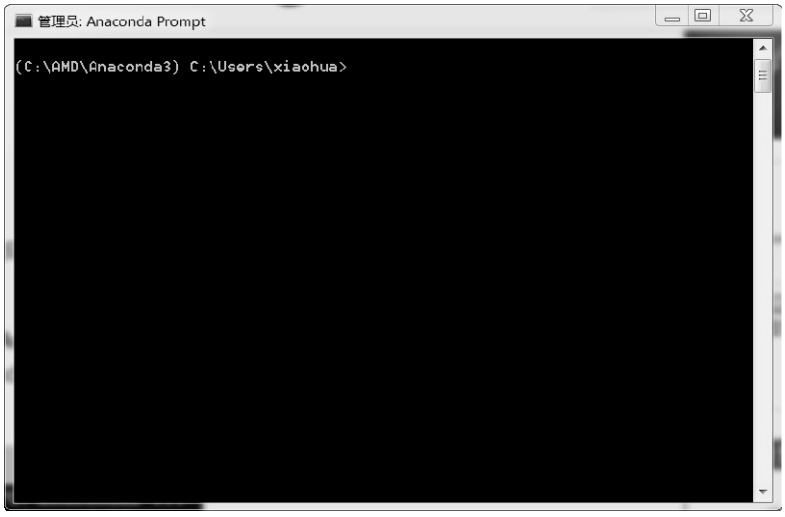

图 2-3　Anaconda Prompt 控制台

3. 第三步：验证 Python

之后在控制台中输入 Python，打印出版本号以及控制符号，并在控制符号下输入代码：

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

输出结果如图 2-4 所示，表明 Anaconda 安装成功。

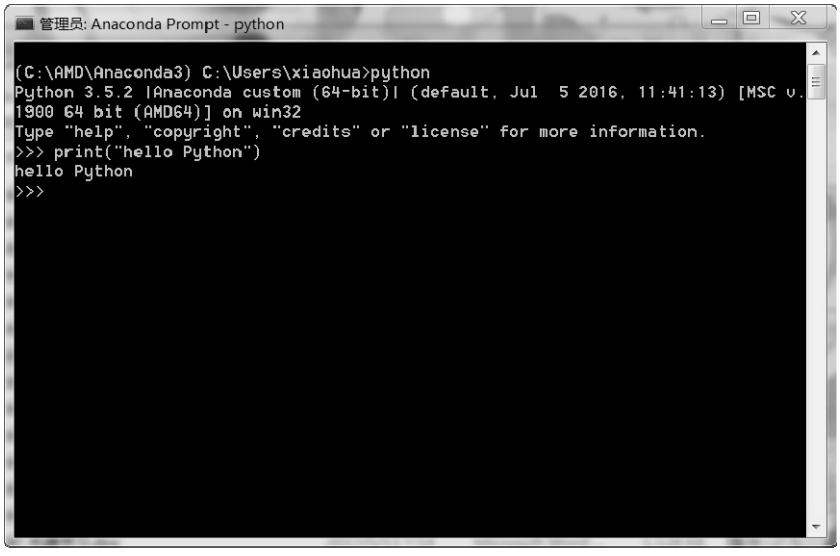

图 2-4　验证 Anaconda Python 安装成功

4. 第四步：使用 conda 命令

笔者建议读者使用 Anaconda 的好处在于，其能够极大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。

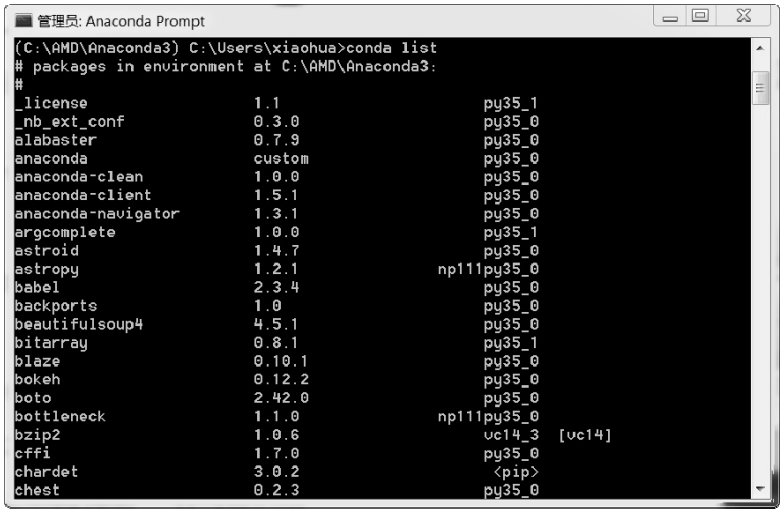

图 2-5　列出已安装的第三方类库

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，例如需要安装 NumPy 包（这个包已经安装过），那么输入的相应命令就是：

```
conda install numpy
```

使用 Anaconda 的好处就是可以自动安装所安装包的依赖类库，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的情况下造成的依赖类库的缺失的困难，使得后续工作顺利进行。

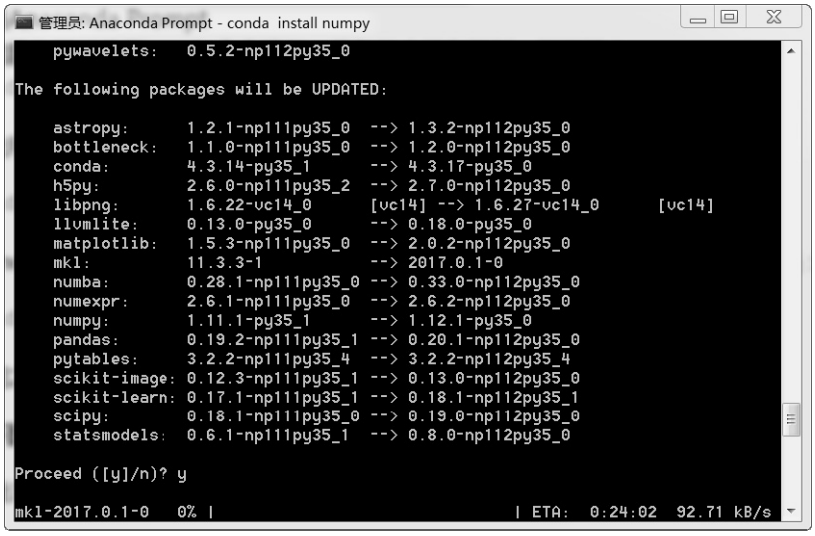

图 2-6　自动获取或更新依赖类库

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 自带的控制台完成。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，笔者建议使用专用的 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，收费的专业版和免费的社区版，如图 2-7 所示。这里笔者建议读者选择免费的社区版。

图 2-7　PyCharm 的免费版

双击运行后进入安装界面，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

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

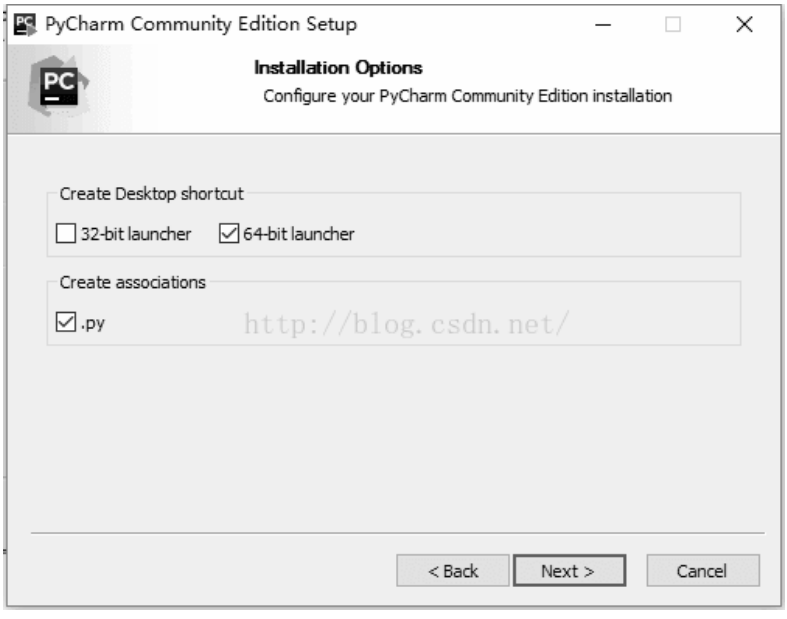

图 2-9　PyCharm 的位数选择

安装完成后出现 Finish 按钮，单击后安装完成，如图 2-10 所示。

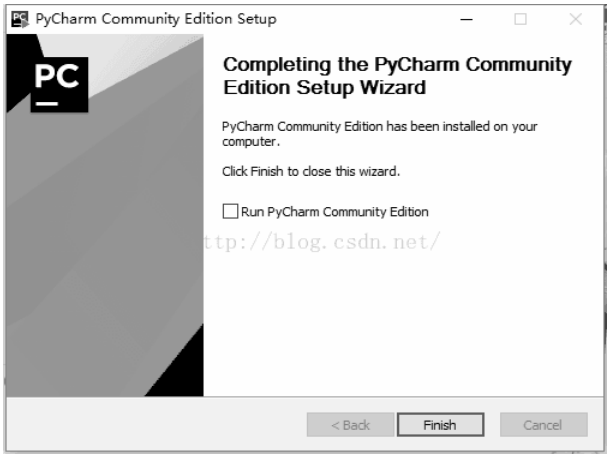

图 2-10　PyCharm 安装完成

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。

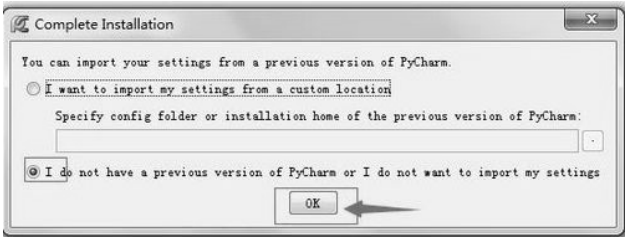

图 2-11　PyCharm 启动定位

这里是对程序存储的定位，一般建议选择第二个由 PyCharm 自动指定。之后单击 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 界面进行配置。如果对其不熟悉，直接单击 OK 按钮使用默认设置即可。

最后就是创建一个新的工程。单击 Create New Project 按钮，即可新建一个 PyCharm 工程，如图 2-13 所示。

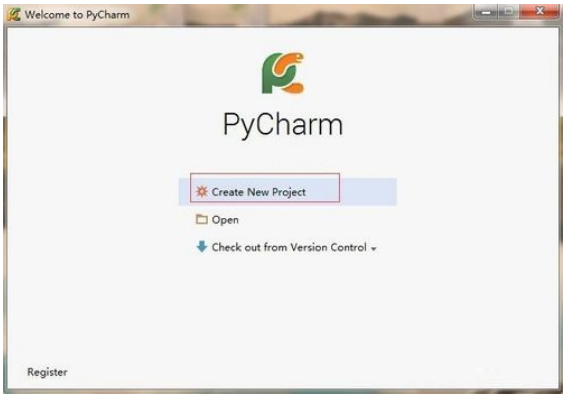

图 2-13　PyCharm 工程创建界面

在这里，笔者建议读者新建一个 PyCharm 的工程文件。右击新建的工程名“PyCharm”，在弹出的菜单中单击 New | Python File 命令（图 2-14），新建一个 helloworld 文件，内容如图 2-15 所示。

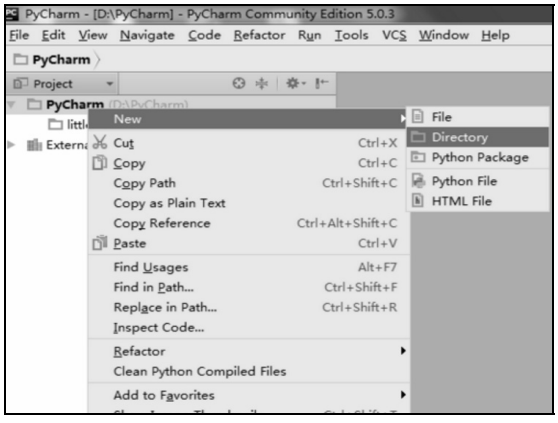

图 2-14　PyCharm 新建文件

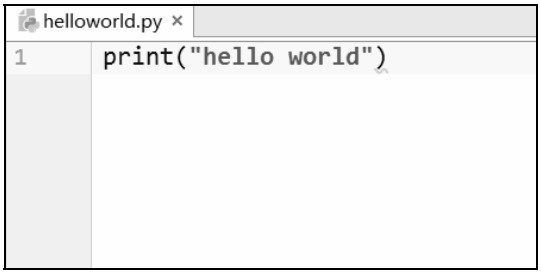

图 2-15　PyCharm 工程创建界面

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

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不加以说明，笔者只是带领读者尝试实现其程序的编写。

softmax 计算公式如下：

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

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，函数计算结果中有两个值 a 和 b，且 a>b。如果简单地以值的大小为单位衡量，那么在后续的使用过程中 a 永远被选用，而 b 由于数值较小则不会被选择。但是有时也需要数值小的 b 被使用，此时 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，因此在计算时 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：

```
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
```

可以看出，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概率。

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

结果如下：

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

2.2 Python 常用类库中的 threading

如果说 Python 的简单易用奠定了 Python 的发展，那么丰富的第三方类库就是 Python 不断前进的动力。随着科技前沿的发展，Python 应用越来越丰富，更多涉及不同种类的第三方类库被加入 Python 之中。

Python 常用类库可参见表 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，Python 2.5 应该已经包含该模块，因此不算第三方库
	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供计算机工程人员以及科学研究人员使用。

2.2.1 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；并且重写了更为方便的 API 来处理线程。

2.2.2 threading 模块中最重要的 Thread 类

Thread 是 threading 模块中的重要类之一，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 2-1 所示。

【程序 2-1】

```
#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()
```

在笔者定义的 MyThread 类中，重写了从父对象继承的 run 方法。在 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。

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 2-16 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 2-16　程序运行结果

通过上面的结果可以看出，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，在程序运行过程中两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。

2.2.3　threading 中的 Lock 类

虽然线程可以在程序的执行过程中提高程序的运行效率，但是其带来的影响却难以忽略。例如，在上一个程序中，每隔一定时间就要打印当前的数值，应该逐次打印的数据却变成了两个相同的数值，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，只有当前线程被释放后，后续线程才可以继续操作。

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

类中的主要代码如上所示。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 2-2 所示。

【程序 2-2】

```
#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()
```

Lock 被传递给 MyThread，并在 run 方法中人为锁定当前的线程，必须等线程执行完毕后，后续的线程才可以继续执行。程序执行结果如图 2-17 所示。

```
myThreadName:0 98
myThreadName:0 99
myThreadName:0 100
myThreadName:1 101
myThreadName:1 102
myThreadName:1 103
myThreadName:1 104
```

图 2-17　程序运行结果

从变色的部分可以看出，线程 2 只有等线程 1 完全结束后才执行后续的操作。在本程序中，Thread1 等到 Thread0 完全结束后才执行第二个操作。

2.2.4　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 2-3 所示。

【程序 2-3】

```
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 2-18 所示。

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

图 2-18　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，只有当被堵塞的进程执行完毕后，后续的进程才可以继续执行。

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。在后续的使用过程中，笔者会带领读者了解和掌握更多的相关内容。

2.3 本章小结

本章介绍了 Python 的基本安装和编译器的使用。在这里笔者推荐读者使用 PyCharm 免费版作为 Python 编辑器，有助于更好地安排工程文件的配置和程序的编写。

同时，本章还介绍了一些常用的类库。这里只是把线程类做了一个详细的介绍。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。