深入浅出机器学习

-基于sklearn的应用实践

作者

2017-10-1

目 录

[第一章 Python语言入门 1](#_Toc497216058)

[1.1 Python简介 1](#_Toc497216059)

[1.2 安装配置Python运行环境 3](#_Toc497216060)

[1.2.1 集成开发环境 3](#_Toc497216061)

[1.2.2 Anaconda 5](#_Toc497216062)

[1.2.3 集成开放环境 8](#_Toc497216063)

[1.3 Python基本语法 8](#_Toc497216064)

[1.3.1数据类型——数字和字符串 8](#_Toc497216065)

[1.3.2 数据类型——列表list 9](#_Toc497216066)

[1.3.3数据类型——列表 11](#_Toc497216067)

[1.3.4 数据类型——字典 11](#_Toc497216068)

[1.3.5基本语句 12](#_Toc497216069)

[1.3.6小结 13](#_Toc497216070)

[1.4第三方包 14](#_Toc497216071)

[1.4.1 NumPy简介 14](#_Toc497216072)

[1.4.2 SciPy简介 14](#_Toc497216073)

[1.4.3 Matplotlib简介 15](#_Toc497216074)

[1.4.4 scikit learn简介 16](#_Toc497216075)

[1.3 Python范例 16](#_Toc497216076)

[第二章 基础知识 20](#_Toc497216077)

[2.1 线性代数基本知识 20](#_Toc497216078)

[2.1.1 第一节 20](#_Toc497216079)

[2.1.2 第二节 20](#_Toc497216080)

[2.1.3 第三节 20](#_Toc497216081)

[2.2 概率论基本知识 21](#_Toc497216082)

[2.2.1 概率、条件概率 21](#_Toc497216083)

[2.2.2 贝叶斯 21](#_Toc497216084)

[2.3 机器学习基本知识 21](#_Toc497216085)

[2.3.1 何为机器学习 21](#_Toc497216086)

[2.3.2 机器学习的主要任务 21](#_Toc497216087)

[2.3.3 机器学习的基本步骤 22](#_Toc497216088)

[第三章 scikit-learn基本知识 23](#_Toc497216089)

[3.1 sklearn结构 23](#_Toc497216090)

[3.2 加载范例数据库 23](#_Toc497216091)

[第四章 预处理 24](#_Toc497216092)

[4.1 标准化、归一化与正则化 24](#_Toc497216093)

[4.1.1 离差标准化 28](#_Toc497216094)

[4.1.2 标准差标准化 29](#_Toc497216095)

[4.1.3 规范化 32](#_Toc497216096)

[4.2 离散特征二值化 33](#_Toc497216097)

[4.2.1 特征二值化（Binarization） 39](#_Toc497216098)

[4.2.2 标签二值化（Label binarization） 40](#_Toc497216099)

[4.2.3 类别特征编码 40](#_Toc497216100)

[4.2.4 .标签编码（Label encoding） 40](#_Toc497216101)

[4.3 特征中含异常值时 40](#_Toc497216102)

[4.4 生成多项式特征 41](#_Toc497216103)

[4.5 插补处理 41](#_Toc497216104)

[4.6 独热编码 41](#_Toc497216105)

[4.7 特征选择 41](#_Toc497216106)

[4.8 降维 41](#_Toc497216107)

[第五章 一般线性模型 42](#_Toc497216108)

[5.1 线性模型概述 42](#_Toc497216109)

[5.2 线性判别分析 P60 42](#_Toc497216110)

[第六章 k-邻近算法 43](#_Toc497216111)

[6.1 k-邻近算法概述 43](#_Toc497216112)

[6.1.1 基本思想 43](#_Toc497216113)

[6.1.2 算法概述 43](#_Toc497216114)

[6.2 算法范例：k-邻近算法应用范例 43](#_Toc497216115)

[6.3 小结 43](#_Toc497216116)

[第七章 支持向量机 44](#_Toc497216117)

[7.1 概述 44](#_Toc497216118)

[7.2 支持向量机算法 44](#_Toc497216119)

[7.3 算法范例：支持向量机分类应用范例 44](#_Toc497216120)

[7.4 小结 44](#_Toc497216121)

[第八章 决策树 45](#_Toc497216122)

[8.1 简介 45](#_Toc497216123)

[8.1.1 决策树表示法 45](#_Toc497216124)

[8.2 决策树算法 45](#_Toc497216125)

[8.3 决策树学习的适用问题 45](#_Toc497216126)

[8.4 算法范例：决策树回归范例 45](#_Toc497216127)

[8.5 小结 45](#_Toc497216128)

[第九章 人工神经网络 46](#_Toc497216129)

[9.1 简介 46](#_Toc497216130)

[9.2 人工神经网络算法 46](#_Toc497216131)

[9.3 人工神经网络算法适用的问题 46](#_Toc497216132)

[9.4 算法范例：BP神经网络应用范例 46](#_Toc497216133)

[9.5 小结 46](#_Toc497216134)

[第十章 模型评价 47](#_Toc497216135)

[10.1 模型评价指标 47](#_Toc497216136)

[10.2 模型评价范例 47](#_Toc497216137)

[10.3 模型选择范例 47](#_Toc497216138)

# Python语言入门

## Python简介

### Python的诞生

Python是著名的“龟叔”Guido van Rossum在1989年圣诞节期间，为了打发无聊的圣诞节而编写的一个编程语言，而Python名称来自Guido挚爱的电视剧Monty Python’s Flying Circus。

### Python的提出背景

当你用一种语言开始作真正的软件开发时，你除了编写代码外，还需要很多基本的已经写好的现成的东西，来帮助你加快开发进度。比如说，要编写一个电子邮件客户端，如果先从最底层开始编写网络协议相关的代码，那估计一年半载也开发不出来。高级编程语言通常都会提供一个比较完善的基础代码库，让你能直接调用，比如，针对电子邮件协议的SMTP库，针对桌面环境的GUI库，在这些已有的代码库的基础上开发，一个电子邮件客户端几天就能开发出来。Python就为我们提供了非常完善的基础代码库，覆盖了网络、文件、GUI、数据库、文本等大量内容，被形象地称作“内置电池（batteries included）”。用Python开发，许多功能不必从零编写，直接使用现成的即可。除了内置的库外，Python还有大量的第三方库，也就是别人开发的，供你直接使用的东西。当然，如果你开发的代码通过很好的封装，也可以作为第三方库给别人使用。

许多大型网站就是用Python开发的，例如YouTube、Instagram，还有国内的豆瓣。很多大公司，包括Google、Yahoo等，甚至NASA（美国航空航天局）都大量地使用Python。

龟叔给Python的定位是“优雅”、“明确”、“简单”，所以Python程序看上去总是简单易懂，初学者学Python，不但入门容易，而且将来深入下去，可以编写那些非常非常复杂的程序。

总的来说，Python的哲学就是简单优雅，尽量写容易看明白的代码，尽量写少的代码。如果一个资深程序员向你炫耀他写的晦涩难懂、动不动就几万行的代码，你可以尽情地嘲笑他。

**3、Python的应用**

既然Python功能这么强大，那适合开发哪些类型的应用呢？

1. Web开发

Python定义了WSGI标准应用接口协调http服务器与基于Python的Web程序之间的沟通。

1. GUI开发

用wxPython或者PyQt来开发跨平台的桌面软件。

1. 操作系统

大多数Linux发布版以及NetBSD、OpenBSD和MacOS都集成了Python,Python标准库包含了多个调用作业系统功能的库。

1. 多媒体

可用于计算机游戏三维场景制作。

**4、Python的缺点**

任何编程语言都有缺点，Python也不例外。优点说过了，那Python有哪些缺点呢？

第一个缺点就是运行速度慢，和C程序相比非常慢，因为Python是解释型语言，你的代码在执行时会一行一行地翻译成CPU能理解的机器码，这个翻译过程非常耗时，所以很慢。而C程序是运行前直接编译成CPU能执行的机器码，所以非常快。

但是大量的应用程序不需要这么快的运行速度，因为用户根本感觉不出来。例如开发一个下载MP3的网络应用程序，C程序的运行时间需要0.001秒，而Python程序的运行时间需要0.1秒，慢了100倍，但由于网络更慢，需要等待1秒，你想，用户能感觉到1.001秒和1.1秒的区别吗？这就好比F1赛车和普通的出租车在北京三环路上行驶的道理一样，虽然F1赛车理论时速高达400公里，但由于三环路堵车的时速只有20公里，因此，作为乘客，你感觉的时速永远是20公里。

第二个缺点就是代码不能加密。如果要发布你的Python程序，实际上就是发布源代码，这一点跟C语言不同，C语言不用发布源代码，只需要把编译后的机器码（也就是你在Windows上常见的xxx.exe文件）发布出去。要从机器码反推出C代码是不可能的，所以，凡是编译型的语言，都没有这个问题，而解释型的语言，则必须把源码发布出去。

这个缺点仅限于你要编写的软件需要卖给别人挣钱的时候。好消息是目前的互联网时代，靠卖软件授权的商业模式越来越少了，靠网站和移动应用卖服务的模式越来越多了，后一种模式不需要把源码给别人。再说了，现在如火如荼的开源运动和互联网自由开放的精神是一致的，互联网上有无数非常优秀的像Linux一样的开源代码，我们千万不要高估自己写的代码真的有非常大的“商业价值”。那些大公司的代码不愿意开放的更重要的原因是代码写得太烂了，一旦开源，就没人敢用他们的产品了。

## 安装配置Python运行环境

### 集成开发环境

集成开发环境(IDE，Integrated Development Environment )是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具。集成了代码编写功能、分析功能、编译功能、调试功能等一体化的开发软件服务套。所有具备这一特性的软件或者软件套(组)都可以叫集成开发环境。如微软的Visual Studio系列，Borland的C++ Builder、Delphi系列等。该程序可以独立运行，也可以和其它程序并用。IDE多被用于开发HTML应用软件。例如，许多人在设计网站时使用IDE(如HomeSite、DreamWeaver等)，因为很多项任务会自动生成。

而Python IDE工具包括文本工具类IDE和集成工具类IDE，常见的文本工具类IDE包括IDLE和Sublime Text，常见的集成工具类IDE包括Pycharm和Spyder。下面分别简单介绍这四种IDE工具的特点：

1、IDLE具有如下特点：

* 自带
* 默认
* 常用
* 入门级

适用于Python初学者，这个功能简单直接，适合编写代码在300个之内。

2、Sublime Text是一种专业编程，适用于专业人员程序员，具有特点如下：

* 专为程序员开发的第三方专用编程工具
* 专业编程体验增强
* 多种编程风格
* 工具非注册免费试使用

3、Pycharm特点如下：

* 社区版免费
* 简单
* 适用于较复杂工程

4、Anaconda是一个集成各类Python工具的集成平台。适用于科学计算和数据分析方面，也具有如下特点：

* 开源免费
* 支持近800个第三方库
* 包含多个主流工具
* 适合数据计算领域开发

显而易见，因为Anaconda是开源免费的，包括的第三方库也是非常多，并且也是非常好用的，同时也是个跨平台工具，适用于Win/Linux/OS X，系统，在国际上尤其是科学计算，数据分析领域应用非常普遍，所以我们在讲Python语言这个系类以Anaconda平台作为我们的开发工具。

### Anaconda

为了更好的理解Anaconda，下面我们会介绍几个支撑Anaconda的工具。Anaconda来自于一个包管理和环境管理工具——conda.简单介绍一下conda,它是一个工具，用于包管理和环境管理；包管理与pip类似，管理Python第三方库；环境管理能够允许用户使用不同版本的Python，并能灵活切换。 Anaconda是一个基于conda这样一个管理工具的集合，它包括conda、某版本的Python以及一批第三方库。由于conda是一个第三方库，它将工具、第三方库、Python版本、conda都当做包来同等对待。

其中还有一个编写Python代码和调试代码工具Spyder，是一个非常优秀的第三方工具。

Anaconda还包含一个交互式编程环境——IPython。IPython是一个功能强大的交互式shell，适合进行交互式数据可视化和GUI相关应用。

所以掌握好Anaconda IDE几乎可以应对Python在科学计算和数据处理所有这方面的开发需求。

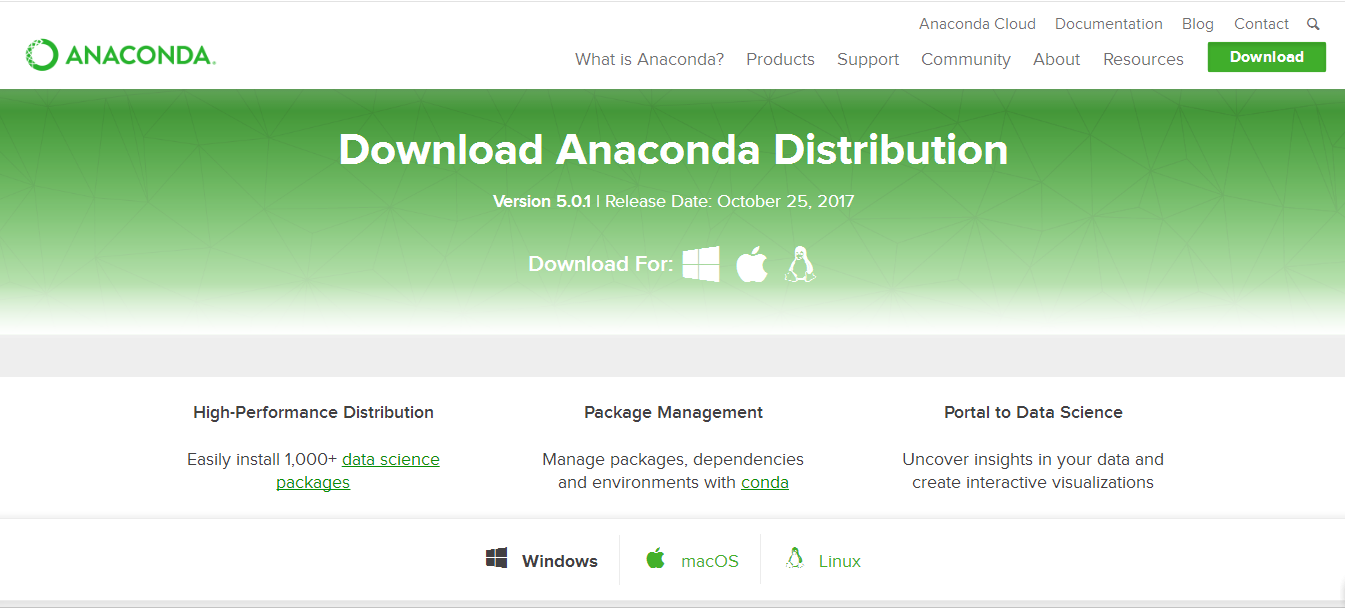
下面简单介绍Anaconda的安装过程：

我们安装Anaconda的过程可以由下面三个步骤来完成：

步骤一：下载的途径直接到Anaconda的官方网站，网址https://www.anaconda.com然后你会看到如下的一个界面:

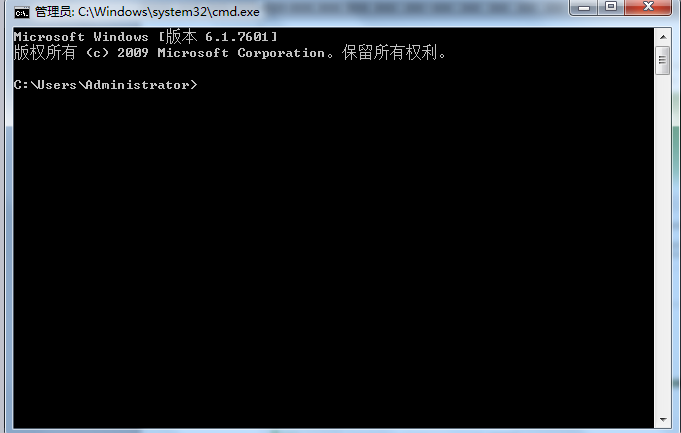


步骤二：然后点击“Download”，会弹出“Download Anaconda Distribution”的界面，直接根据你是Windows操作系统还是iOS系统，选定相对应的直接下载即可。

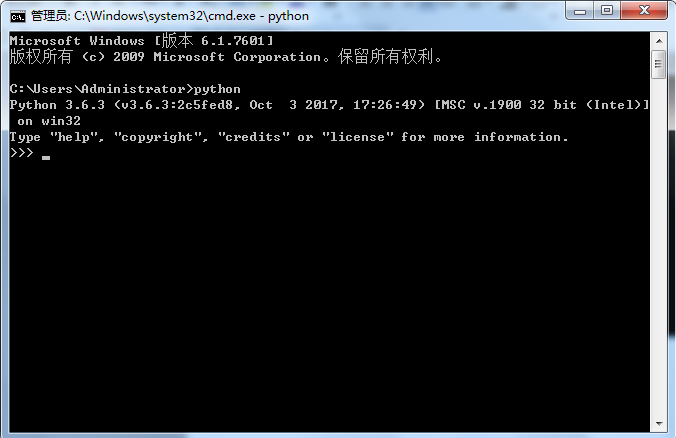


步骤三：测试Anaconda安装是否成功

在Windows下点开始，输入cmd命令，弹出如下界面：

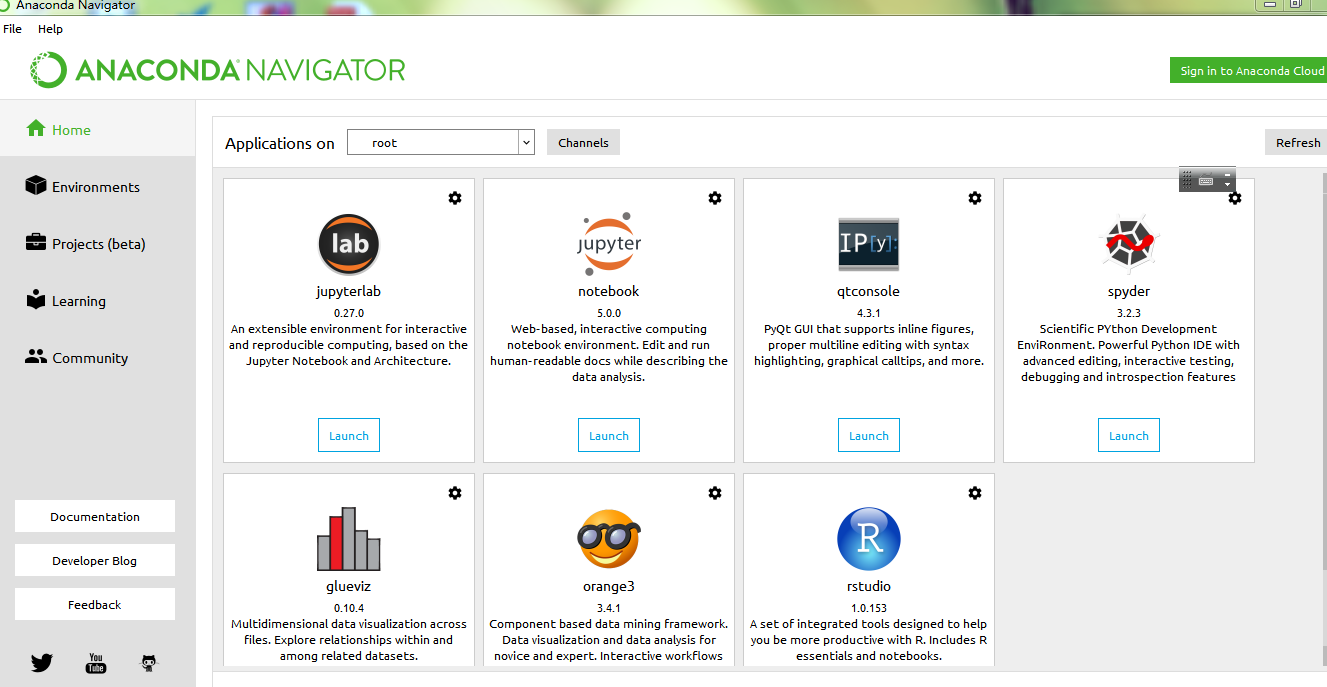


然后输入Python，弹出如下界面：



那就代表Anaconda安装成功了。

最后使用Anaconda IDE集成开发工具,点击开始，显示电脑所有程序中，点击Anaconda Navigator，来启动Anaconda,启动后界面如下：



https://www.liaoxuefeng.com/

## Python基本语法

Python是一种计算机编程语言。计算机编程语言和我们日常使用的自然语言有所不同，最大的区别就是，自然语言在不同的语境下有不同的理解，而计算机要根据编程语言执行任务，就必须保证编程语言写出的程序决不能有歧义，所以，任何一种编程语言都有自己的一套语法，编译器或者解释器就是负责把符合语法的程序代码转换成CPU能够执行的机器码，然后执行。Python也不例外。

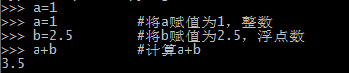
任何计算机系统，其工作的实质都在于，将输入后的数据处理后再输出，即：

输入→处理→输出

### 1.3.1数据类型——数字和字符串

计算机，顾名思义就是可以做数学计算的机器，因此，计算机程序理所当然地可以处理各种数值。但是，计算机能处理的远不止数值，还可以处理文本、图形、音频、视频、网页等各种各样的数据，不同的数据，需要定义不同的数据类型。数据类型的内容有很多，如字典、文件、列表和元组。

作为动态类型的语言，Python中使用数字无需先声明它的类型，如下所示。



在上面的例子中，我们分别将整数1和浮点数2.5赋值给了变量a和b，“=”就是赋值的意思，“+”为运算符，在Python中，除了加减乘除取余等运算符外，还有逻辑运算符如位移运算符等等。接下来直接输入公式即可完成计算。每句后面以“#”开头的文本时注释，它对于脚本的运行没有任何影响，目前可以完全被我们忽略掉——当然，在复杂脚本里面使用注释说明自己编程时的思路算法也是一个很好的习惯。

Python中的字符串是用来表示和储存文本的，通常由单引号、双引号或者三引号包围，可以包含数字、字母以及一些控制字符，如换行符（/n）、制表符（/t）、和回车（/r）等等。字符串的形式一般如下所示。

IMG_256

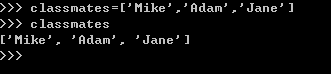
接下来用实例演示转义字符在字符串中的使用。



这个例子里面，print语句的作用是输出字符串，只有在使用print输出字符串时才能够解释字符串中的转义字符。需要注意的是，在Python3中，print的正确语法为print（），否则就会显示错误。

### 1.3.2 数据类型——列表list

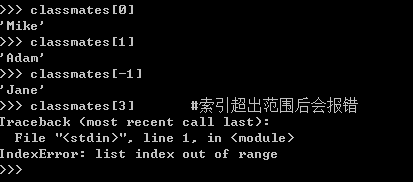
Python内置的一种数据类型是列表：list。List是一种有序的集合，可以随时添加和删除其中的元素。比如，列出班里所有同学的名字，就可以用一个list来表示：



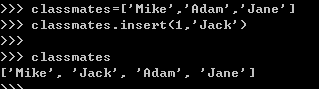
在上面的例子里，变量classmates就是一个list。我们还可以用len()函数来获取list元素的个数。

1509348425(1)

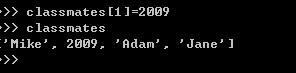
用索引来访问list中每一个位置的元素，记得索引是从0开始的。当索引超出了范围时，Python会报一个IndexError错误，所以，要确保索引不要越界，记得最后一个元素的索引是len(classmates) - 1。如果要取最后一个元素，除了计算索引位置外，还可以用-1做索引，直接获取最后一个元素。



List是一个可变的有序表，所以，可以往list中追加元素到末尾，也可以把元素插入到指定的位置，要删除指定位置的元素，用pop（）方法，



要把某个元素替换成别的元素，可以直接赋值给对应的索引位置，list里面的元素的数据类型也可以不同，比如下图的这个例子。

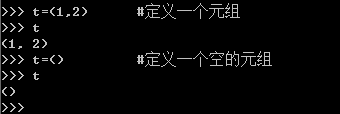


不过因为list里面可以包含不同的数据类型，这也造成它运行起来可能会比较慢，为了弥补这个缺陷，我们将来会引用数组这一个概念。

### 1.3.3数据类型——列表

接下来我们来讨论另一种有序列表，叫做元组：tuple。tuple和list非常类似，但是tuple一旦初始化就不能修改，比如同样是列出同学的名字，现在，classmates这个tuple不能变了，它也没insert()这样的方法。其他获取元素的方法和list是一样的，你可以正常地使用classmates[0]，classmates[-1]，但不能赋值成另外的元素。

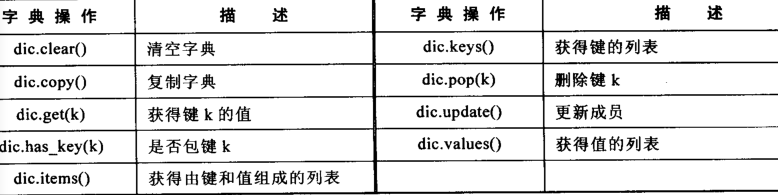
不可变的tuple有什么意义？因为tuple不可变，所以代码更安全。如果可能，能用tuple代替list就尽量用tuple。所以当你定义一个tuple的时候，tuple的元素就必须被确定下来。



需要注意的是，要定义一个只有1个元素的tuple，如果你这么定义：t=(1),定义的不是tuple，是1这个数！这是因为括号()既可以表示tuple，又可以表示数学公式中的小括号，这就产生了歧义，因此，Python规定，这种情况下，按小括号进行计算，计算结果自然是1。所以，只有1个元素的tuple定义时必须加一个逗号，来消除歧义：t=(1,)。而Python在显示只有1个元素的tuple时，也会加一个逗号，以免你误解成数学计算意义上的括号。

### 1.3.4 数据类型——字典

字典是Python中比较特别的一类数据类型，以大括号“{}”包围的数据集合。字典与列表的最大不同在于，字典是无序的，在字典中是通过键来访问成员的。字典也是可变的，可以包含任何其他类型，不过，字典中的成员位置只是象征性的，并不能通过其位置来访问该成员。字典中的成员是以“键：值”的形式来声明的，常用字典操作都在下面这张表里：



### 1.3.5基本语句

语句是Python脚本的基础，它能帮助我们根据一定的条件来执行脚本中的不同语句，从而完成不同的功能。

首先，我们来看条件语句。计算机之所以能做很多自动化的任务，因为它可以自己做条件判断。，在Python程序中，它是if语句是基本的的条件测试语句，用来判断可能遇到的不同情况，以及在不同情况下应该进行的下一步操作。它的基本形式如下：

If <条件>: #设为条件A

<语句> #要用缩进来表示语句处于if语句之中，设为语句A

elif <条件>： #设为条件B

<语句> #设为语句B

else：

<语句> #设为语句C

用简单的话来说就是，首先判断输入的数据是否满足条件A，如果满足就执行条件A，不满足的话就接着判断是否符合条件B的要求，如果满足了就执行语句B，否则就执行语句C。

第二个要介绍的是循环语句。要计算1+2+3，我们可以直接写表达式：

IMG_256

要计算1+2+3+...+10，勉强也能写出来。但是，要计算1+2+3+...+10000，直接写表达式就不可能了。为了让计算机能计算成千上万次的重复运算，我们就需要循环语句。

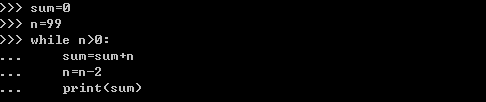
Python的循环有两种，一种是for...in循环，依次把list或tuple中的每个元素迭代出来，如下面的例子所示。

IMG_256

执行这段代码，会依次打印names的每一个元素：

IMG_256

第二种循环是while循环，只要条件满足，就不断循环，条件不满足时退出循环。比如我们要计算100以内所有奇数之和，可以用while循环实现：



上面这段例子就是在循环内部变量n不断自减，直到变为-1时，不再满足while条件，循环退出。

while语句与for语句不同的是，while语句中只有在测试条件为假时才会停止，所以在while的语句块中一定要包含改变条件的语句，以保证循环能够结束，避免死循环的出现。

### 1.3.6小结

Python的语法比较简单，采用缩进方式。

缩进有利有弊。好处是强迫你写出格式化的代码，但没有规定缩进是几个空格还是Tab。按照约定俗成的管理，应该始终坚持使用4个空格的缩进。缩进的另一个好处是强迫你写出缩进较少的代码，你会倾向于把一段很长的代码拆分成若干函数，从而得到缩进较少的代码。

缩进的坏处就是“复制－粘贴”功能失效了，这是麻烦的地方。当你重构代码时，粘贴过去的代码必须重新检查缩进是否正确。此外，IDE很难像格式化Java代码那样格式化Python代码。

最后，请务必注意，Python程序是大小写敏感的，如果写错了大小写，程序会报错。

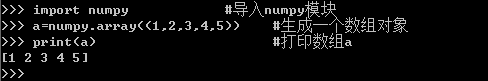
## 第三方包

Python本身就内置了很多非常有用的模块，只要安装完毕，这些模块就可以立刻使用。除了内建的模块外，Python还有大量的第三方模块。基本上，所有的第三方模块都会在PyPI - the Python Package Index上注册，只要找到对应的模块名字，即可用pip安装。而由于前面本书已经推荐大家安装了Anaconda，一些常用第三方库已经包括在里面，省去了我们很多需要自行安装的环节。

科学计算是计算机应用的主要内容之一，也是机器学习的很重要的部分，NumPy和SciPy是Python中用以实现科学计算的模块包，Matplotlib是绘制2D图形的一种模块，Sklearn包含有完善的文档和丰富的机器学习算法，这也是Python语言如此好用的原因之一。接下来我们主要介绍一下这几个第三方包。

### 1.4.1 NumPy简介

NumPy提供了Python没有提供的数组对象，使用了NumPy的数组对象就可以创建类似于C语言中的数组。使用NumPy时要先导入NumPy模块，如下图所示。

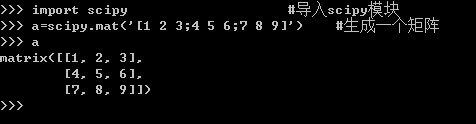


上面这个例子里，

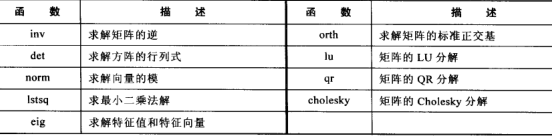
NumPy主要为后面的SciPy和sklearn提供了基础的数据结构。

### 1.4.2 SciPy简介

SciPy模块依赖于NumPy，但是SciPy提供了更多的数学工具。使用SciPy不仅可以进行矩阵运算，还可以用来求解线性方程组、进行积分运算、优化等。SciPy的功能非常接近Matlab。SciPy简单的运用如下图所示：



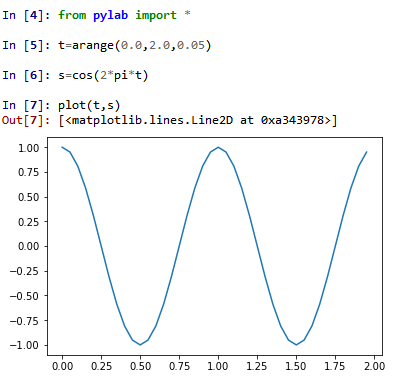
使用SciPy可以完成矩阵分解、线性方程组求解和多项式求根数学运算。SciPy中的函数名与Matlab中的函数名大部分都相同，用法也差不多。熟悉Matlab的用户可以很快的熟悉这个第三方包。下图是一些常用的函数，以后在机器学习中很多也都会被用到：



### 1.4.3 Matplotlib简介

在Python中，我们可以使用Matplotlib来绘制函数图形，它依赖于NumPy与SciPy。和SciPy一样，其中的大部分函数都和Matlab中的函数名相同。它可以绘制多种形式的图形包括普通的线图、直方图、饼图、散点图以及误差线图等。

使用Matplotlib绘制图形主要使用它的plot函数，Matplotlib还包含了一些用于设置x、y轴标签文本以及图形标题的函数。同时它还支持一部分Tex排版命令，可以较好地显示数学公式。接下来我们将在IPython交互式命令行中使用Matplotlib绘制cos图形。



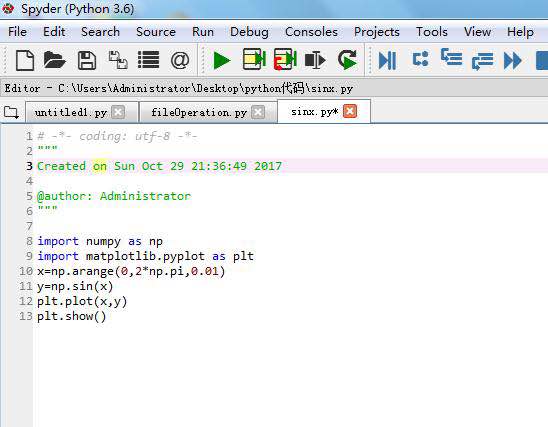
### 1.4.4 scikit learn简介

Sklearn包括了很多种机器学习的类型，比如说监督学习和非监督学习，在监督学习中，又可以分为分类学习和线性回归学习。当我们想训练我们的电脑来识别一些图片，比如猫狗猪牛等等时，这个叫做分类学习。当我们想预测房价高低时，用到的就是线性回归学习。非监督学习就是指，我们不告诉计算机你看到了什么东西，让计算机自己琢磨出来这些数据有什么不同，再画一条线区分开来这些数据的类型。从上面的大概了解可以看出来Sklearn是非常实用的。

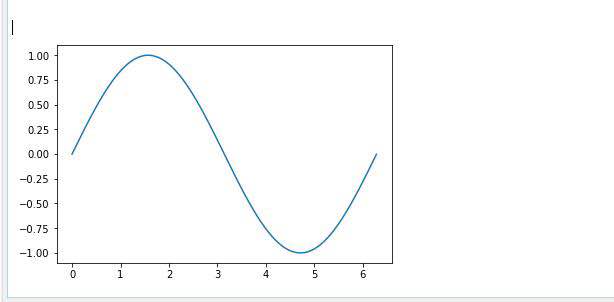
## Python范例

前面我们已经把Python安装到自己的电脑上了，并且把Python的基本语句也已经做了详细的介绍，那我们下面来实践下，本书给出了两个例子供大家参考。

打开Anaconda的代码编辑工具Spyder,在左边编辑区输入如下代码，我们来画出一个sin（x）的图像。



输入如上的代码后，点击上方Run的命令，可以得到如下的一个sinx图形



现在我们来分析一下这个脚本的内容。



首先，在第一行，import作为调用函数，因为要画正弦曲线要先设定一下x的取值范围，从0到2π，这个要用到numpy模块，用import调用函数，来导入模块numpy并以np作为别名

第二步由于是要画图，所以需要用到matplotlib.pyplot模块中plot方法



第三步Arange函数是numpy包中一个函数，np.pi表示π，这个表示x从0到2π，以0.01步进。



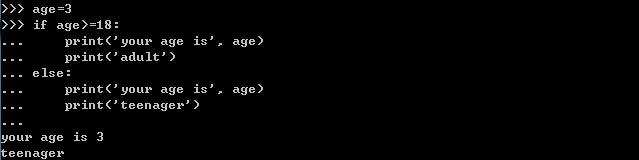
第四步就是定义下y的值



第五步就是画图呈现x轴y轴的意思



最后一步就是显示图的意思。

接下来我们利用Python来运行一个简单的if语句，打开开始菜单，输入cmd命令，弹出提示符窗口，敲入Python后，然后输入下面简单的if语句。

同样的，我们来分析一下这个脚本的内容：



这个意思就是说年龄是为3，如果年龄是大于等于18，那么就输出两行，一行是输出你的年龄，写出年龄，另外一行是输出成年人。根据Python的缩进规则，如果if语句判断是true，就把缩进的两行print语句执行了，否则，什么也不做。

也可以给if添加一个else语句，意思是，如果if判断是False,不要执行if的内容，去把else执行了,如下面的一个程序：



这个if语句进行了一个条件判断，因为之前输出的年龄是3，因为是3是小于18，那就执行else后的语句，然后同样输出两行，



一行是输出你的年龄是3，另外输出是青年人这一行。

## 小结

# 基础知识

线性代数作为数学的一个分支，广泛用于科学和工程中。同样地，线性代数对于理解和从事机器学习算法相关工作也是很有必要的，比如说，在机器学习某种事件特征我们常常会描绘成特征向量，那么线性代数所涉及到的矩阵的运算和理论方法都必将会应用进来。因此，在我们开始介绍机器学习之前，我们集中探讨一些必备的线性代数知识，如果你已经很熟悉线性代数，那么你可以轻松地跳过本章。

## 线性代数基本知识

### 向量与矩阵相关的基础知识

1、向量

把数排成一列就是向量。比如和和等，当需要强调分量个数的时候，我们分别称上面的例子为2维向量、3维向量和4维向量。我们在理解2维向量的时候可以用二维坐标去理解，3维向量我们可以用3维空间中的一点来表示。同样的，我们可以用有向线段去理解向量，其中线段的起点就是原点O，终点是该向量对应的位置，与此同时，我们就可以通过图形来理解向量的加法和数量乘法。我们可以把向量看作空间中的点，每个元素是不同的坐标轴上的点。

2、向量空间

我们可以把向量看作空间中的点，每个元素是不同的坐标轴上的点。对于这样一个附加了加法与数量乘法运算的世界，我们就称之为向量空间，也有人称之为线性空间。

3、矩阵

把数排列成长方形就是矩阵。比如或或等。需要强调矩阵规模（大小，高和宽，行数和列数）的时候，我们分别称以上例子为2×2矩阵、2×3矩阵和5×3矩阵。这个时候当行数和列数相同的时候，我们称之为方阵。我们通常会赋予矩阵粗体的大写变量名称，比如 A。矩阵A中第i行第j列的值，称之为A的（i，j）元素。为了简单起见，2×3矩阵可以写成A。

这里我们可以把矩阵理解为映射，例如：将n维向量x乘以m×n矩阵A，能得到m维向量y=Ax。也就是说，指定了矩阵A，就确定了从向量到另外一个向量的映射。这里是从n维空间到m维空间的移动。

4、逆矩阵

对于方阵A，它的逆映射对应的矩阵称之为A的逆矩阵，记为A。任意的向量x，若有Ax=y,则有Ay=x成立。反之，对于任意的向量y，若有Ay=x,则有Ax=y。更为直接一点说，如果有这样的一个映射，对于已知的目标点y,总能还原到出发点x,那么这个映射所对应的矩阵就是A。

5、矩阵的转置

对于矩阵A，将其行列互换得到的矩阵，就称为A的转置矩阵，记为A。例如：A= → A=其中，第一行变成了第一列，第二行变成了第二列。

1. 行列式

行列式，记作 det(A)，是一个将方阵 A 映射到实数的函数。行列式等于矩阵特征值的乘积。行列式的绝对值可以用来衡量矩阵相乘后空间扩大或者缩小了多少。如果行列式是 0, 那么空间至少沿着某一维完全收缩了，使其失去了所有的体积。如果行列式是 1, 那么矩阵相乘没有改变空间体积。

2.1.2矩阵的运算规则

矩阵乘法是矩阵运算中最重要的操作之一。两个矩阵 A 和 B 的矩阵乘积 是第三个矩阵 C。为了使乘法定义良好，矩阵 A 的列数必须和矩阵 B 的行数相等。如果矩阵 A 的形状是 m×n，矩阵 B 的形状是 n×p，那么矩阵 C 的形状是 m × p。矩阵乘积可以作用于两个或多个并在一起的矩阵。例如：

C = AB.

矩阵乘积运算有许多有用的性质，从而使矩阵的数学分析更加方便。比如，矩阵乘积服从分配律：A(B + C) = AB + AC.

矩阵乘积也服从结合律: A(BC) = (AB)C

但是不同于标量乘积，矩阵乘积并不满足交换律（AB = BA 的情况并非总是满足）。然而，两个向量的点积 (dot product) 满足交换律：

x ⊤ y = y ⊤ x

矩阵乘积的转置有着简单的形式：

(AB) ⊤ = B ⊤ A ⊤

由于本书的重点不是线性代数，我们并不试图展示矩阵乘积的所有重要性质，但读者应该知道矩阵乘积还有很多有用的性质。

2.1.3 线性相关、秩以及特征值、特征向量的知识

1、线性相关

如果一组向量中的任意一个向量都不能表示成其他向量的线性组合，那么这组向量被称为线性无关。如果某个向量是一组向量中某些向量的线性组合，那么我们将这个向量加入到这组向量后不会增加这组向量的所构成的空间。用数学语言来说的话，如果对于数u,...u，当成立时有u=...=u=0，则称a...a为线性无关的。

2、秩有关的概念

令A是m×n矩阵，也就是说，我们考虑把n维向量x变成m维向量y=Ax的这一映射。这里我们把A维数命名为矩阵A 的秩。也就是非零行的行数就是矩阵的秩。记为rankA。当rankA=n（秩与原空间（定义域）的维数相等）=A是单射；rankA=m（秩与目标空间（值域）的维数相等）=A是满射。

以上论述从直观上看也是显然的，如果n维原空间在映射之后的目标空间中同样保持着n维，则一定没有遇到“压缩”的情况，另外，在映射到目标空间时，如果有着和整个目标空间相同的m维，那么一定也就覆盖到了空间全体。

3、特征值与特征向量

一般而言，对于方阵A，满足：

Ap=p （p≠0）

的数和向量P分为矩阵的特征值和特征向量。

从几何意义上来讲，特征向量乘上A之后，除了长度会有伸缩变化，方向不发生变化，这里的长度变化倍率，便是特征值。

### 2.1.1 第一节 n维向量和向量空间

1、n维向量

n个数a，a，...，a构成的有序数组称为n维向量，记作（a，a，...，a)或（a，a，...，a）⊤，分别称为n维行向量或n维列向量，也就是1×n或n×1矩阵，数a称为向量的第i个分量。

2、向量空间

设v是n维向量的非空集合，且v中向量对于加法及数乘这两种运算封闭，则称v是向量空间。在这样的一个世界里，更加强调方向性，一般我们表示向量都用方向箭头来表示。其实，向量空间是我们生活的现实空间的一个缩影，也是对现实空间的抽象化。

如果W是向量空间V的一个非空子集，如W对V的加法及数乘两运算都封闭，则称W是V的子空间。

3、基及维数

设V是向量空间，若V中有r个向量线性无关，且V中任一向量都可由这r个向量来线性表出，则称这r个向量为向量空间V的一组基，r为向量空间V的维数，并称V为r维向量空间。

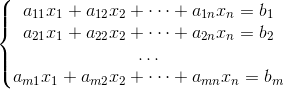
4、极大线性无关组

在向量组a，a，...，a中，如存在一个部分组a,a,...,a线性无关，且再添加进组中任一个向量a,向量组a,a,...,a，a一定线性相关，则称a,a,...,a是向量组a，a，...，a的一个极大线性无关组。

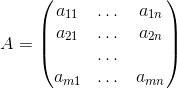
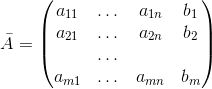
向量组a，a，...，a的极大线性无关组所含向量的个数r称为该向量组的秩，记为r(a，a，...，a)=r。

### 2.1.2 第二节 图像解方程组

### 在第一节中，我们已经介绍了线性代数有关的基础知识，了解了向量，也知道了空间的概念，培养了把矩阵看成是映射的思想。那么，从本节开始，我们一起来学习线性代数的具体知识。首先我们从解方程谈起，学习线性代数的应用之一就是求解复杂方程问题，本节核心之一也是从空间，向量的角度来求解方程。

以下是线性代数方程组的一般形式：  
为m个方程n个未知数的方程组。

其中：

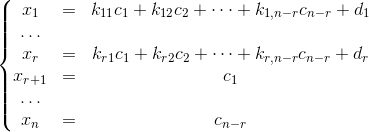
称称为系数矩阵，称为增广矩阵。

* 方程组与它的增广矩阵互相唯一决定。

判断方程组是否有解的情况：

1. 当方程个数等于未知数个数，则若≠0,方程组有唯一解；
2. 若=0且r（A）≠r(****),方程组有无解
3. 若=0且r（A）=r()，方程组由无穷多个解

接下来讨论方程组通解的情况：

1. 通解：设方程组有无穷组解，则有个自由未知量，令得  
   其中称为方程组的通解。  
   通解也可写成向量式。

2.特解：通解中的某个具体的解。

3.齐次线性方程组（右边常数项全为0）

设A是m×n矩阵，齐次方程组Ax=0有非零解的充要条件是r(A)<n，亦是A的列向量线性无关。

特别地，=0也是齐次线性方程组有非零解的充要条件。

当方程个数<未知数的个数同样为其有非零解的充要条件。

角度一：从行图像与列图像的角度解方程

1） 二维的行图像

我们首先通过一个例子来从行图像角度求解方程

〖例1〗求解方程 2x-y=0

-x+2y=3

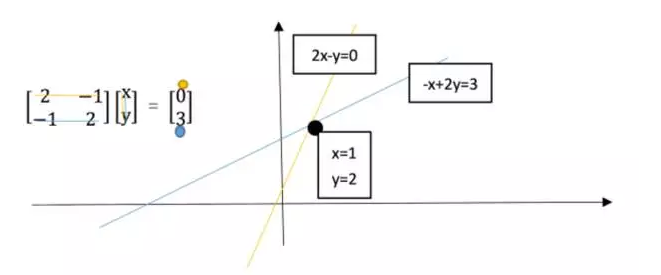
我们首先把方程写成矩阵的形式：

系数矩阵 未知向量 向量

接下来我们通过行图像来求解这个方程：

所谓行图像，就是在系数矩阵上，一次取一行构成方程，在坐标系上作图。和我们在初等数学中学习的作图求解方程的过程无异。



2）二维的列图像

从列图像的角度我们来解决这个方程组 2x-y=0

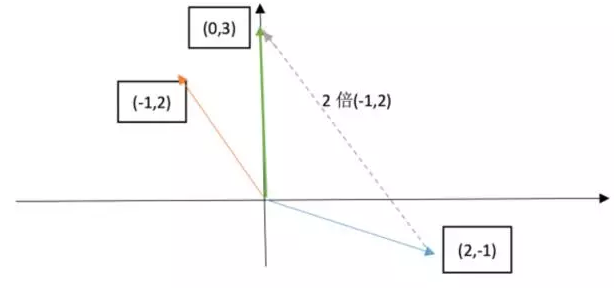
-x+2y=3

这一次我们来求解这个方程组，我们把从列提取，使用的矩阵为：

x+y=

如图，我们使用列向量构成系数矩阵，将问题化为向量和向量的线性组合，使其组合的结果为。

接下来我们使用列图像求解此方程：



即寻找合适的 x，y 使得 x 倍的(2,-1) + y 倍的(-1,2)得到最终的向量(0,3)。很明显能看出来，1 倍(2,-1) + 2 倍(-1,2)即满足条件。反映在图像上，明显结果正确。

我们再想一想，仅仅针对这个x+y=方程，我们任意取x和Y我们能的得到的是什么呢，可想而知，我们得到的是铺满整个平面的任意方向的向量。

3）高维行图像

我们把方程维数进行推广，从三维开始，

2x-y=0

-x+2y-z=-1,如果我们继续用行图像去解方程，我们会得到一个非常

-3y+4z=4

复杂的图像，如果绘制行图像，很明显这是一个三个平面相交得到一点，我们想直接看出这个点的性质可谓是难上加难。

比较靠谱的思路是先联立其中两个平面，使其相 交于一条直线，在研究这条直线与平面相交于哪个点，最后得到点坐标即为方程的解。这个求解过程对于三维来说或许还算合理，那四维呢？五维甚至更高维数呢？直观上很难直接绘制更高维数的图像，这种行图像受到的限制也越来越多。所以我们考虑用列图像来理解。

4）高维列图像

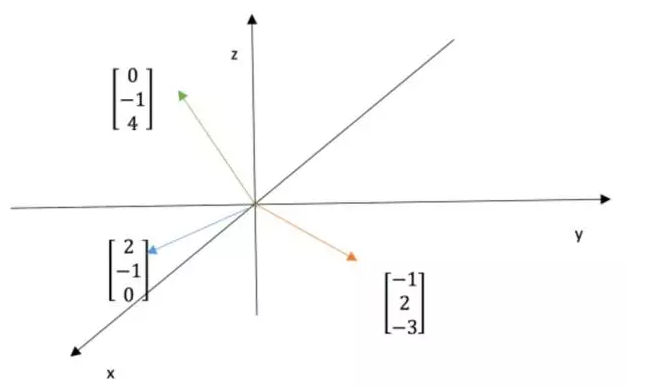
还用那个方程组 2x-y=0

-x+2y-z=-1，我们采用列图像的思路去求解这个方

-3y+4z=4

程组。上面的方程组可以化为：x+y+z=

左侧是线性组合，右侧是合适的线性组合组成的结果，这样一来思路就清晰多了，“寻找线性组合”成为了解题关键。



很明显这道题是一个特例，我们只需要取 x = 0，y = 0，z = 1。就得到了结果，这在行图像之中并不明显。

当然，之所以我们更推荐使用列图像求解方程， 是因为这是一种更系统的求解方法，即寻找线性组合，而不用绘制每个行方程的图像之后寻找那个很难看出来的点。

另外一个优势在于，如果我们改变最后的结果 b，例如本题中，

x+y+z=

那么我们 2 −1 1 0 −3 4 −3 就重新寻找一个线性组合就够了，但是如果我们使用的是行图像呢？那意味着我 们要完全重画三个平面图像，就简便性来讲，两种方法高下立判。

[https://mp.weixin.qq.com/s?\_\_biz=MzI4MDYzNzg4Mw==&mid=2247485066&idx=1&sn=7ee345dcd76e4d90d884eda7a09f148f&chksm=ebb43e5edcc3b748d1d1cc0ca29787aae4406f4682ff3b1ef245d78bd7c6b83c0ccda806b677&scene=21#wechat\_redirect](https://mp.weixin.qq.com/s?__biz=MzI4MDYzNzg4Mw==&mid=2247485066&idx=1&sn=7ee345dcd76e4d90d884eda7a09f148f&chksm=ebb43e5edcc3b748d1d1cc0ca29787aae4406f4682ff3b1ef245d78bd7c6b83c0ccda806b677&scene=21" \l "wechat_redirect)

http://www.cnblogs.com/CQBZOIer-zyy/p/5722418.html

### 2.1.3 第三节 消元法解方程组

1、消元法

对于一些“好”的系数矩阵（可逆矩阵）A 来说，我们可以使用消元法来求解方程 Ax = b，我们还是从一个例子谈起。

求解方程 x+2y+z=2

3x+8y+z=12 还是使用矩阵来计算，写成矩阵运算的形式：

4y+z=2

=

所谓矩阵的消元法，与我们初等数学中学习的解二元一次方程组的消元法其实师出同门，都是通过将不同行的方程进行消元运算来简化方程，最后能得到简化的方程组，只不过这里我们把系数单独抽出来进行运算，寻找一种矩阵情况下的普遍规律而已。

消元针对的对象是系数矩阵A**：**，首先注意，左上角的1是消元法关键，我们称之为主元1，接下来就是我们熟悉的“将一行乘倍数加到另一行”的行化简的方法将第一列中除了主元之外的元素全变为0.

第一步目标达成，我们在第一列中只留下了主元1，很好，接下来我们可以认为第一行与第一列已经“完工”了，再看去掉第一行第一列之后右下角剩下的部分：，同样，我们将左上角的2视为主元，消元第一列，使其列上（不包括第一行中元素），除此主元2之外皆为0.

这个时候第三行就只剩下5，我们直接将其处理为主元即可。由于A矩阵可逆，经过消元处理得到的上三角矩阵U=中有三个主元，至此，消元结束，得到的U即为我们想要化简的形式。

注意：并不是所有的A矩阵都可以消元处理，需要注意在我们消元过程中，如果主元位置（左上角）为0，那么意味着这个主元不可取，需要进行“换行”处理：首先看它的下一行对应位置是不是0，如果不是，就将这两行位置互换，将非零数视为主元，如果是，就看下一行，以此类推，若其下面一行都看到了，仍然没有非零数的话，那就意味着这个矩阵不可逆，消元法求出的解不唯一。

2、回带求解

其实回带求解应该和消元法同时进行，只不过本课中以及一些软件工作原理中它们是先后进行的，所以我们这里分开讨论，先介绍增广矩阵：

上文的例子中，矩阵形式=其增广矩阵的形式为：一下子就看出来了，就是把系数矩阵 A 和向量 b 拼接成一个矩阵就行了。然后像我们之前说的那样消元，但是这次要带着增广的b的那一列一起进行：

   带回方程Ax=b,变为： x+2y+z=2

x+2y+z=2

2y-2z=6 从下向上开始求解，很容易求出x.y.z的值了。

5z=-10

[https://mp.weixin.qq.com/s?\_\_biz=MzI4MDYzNzg4Mw==&mid=2247485175&idx=1&sn=b18c1df20be69abd403ee97f1e7ff5b3&chksm=ebb43e23dcc3b735c8c8029bdb5d6a22bb0745f8a5a3d7dc025c84457ac2c75ae21047e405f3&scene=21#wechat\_redirect](https://mp.weixin.qq.com/s?__biz=MzI4MDYzNzg4Mw==&mid=2247485175&idx=1&sn=b18c1df20be69abd403ee97f1e7ff5b3&chksm=ebb43e23dcc3b735c8c8029bdb5d6a22bb0745f8a5a3d7dc025c84457ac2c75ae21047e405f3&scene=21" \l "wechat_redirect)

### 2.1.4 第四节 矩阵乘法和逆矩阵

第一节我们已经介绍过了向量与矩阵之间的乘法，这节我们来介绍一下矩阵之间的乘法运算以及引进求逆的方法。

1、矩阵乘法最常见的求解方式

首先来了解矩阵之间进行乘法运算时，我们是如何求解单个元素的呢？

？ ？ = ？

A \* B = C

现在我们取C中的一个元素，就比如C中的一个元素C：

这个下标的意义就是C。它的计算也和其密切相关。即A中C对应行向量与B中C对应列向量的数量积的值即为C的值。

如图:

=

A B C

C=

推广：C=(A中第i行向量）（B中第j列向量）=

另外我们这里还要注意一下矩阵规格的问题，因为我们矩阵乘法运算是使用前一个矩阵的行向量点乘后一个矩阵的列向量，所以从向量角度看，它们必须有相同的分量。故，A的列数必须与B的行数相同，结果C矩阵的规格是A的行数，B的列数。

2、列组合与行组合方式

1）列组合的角度：还记得我们之前学习过矩阵与列向量的乘积，得到一个列向量：

 =A A A A=C

A B

B的各列，用来线性组合A的列向量

之后，每一个类似于A这样的A矩阵与向量乘积，都可以转化为矩阵A的列向量线性组合。最后将得到的列向量组合在中，即为结果C。这种方法的关键就是将右侧矩阵B看做列向量组合，将问题转化为矩阵与向量的乘法问题，也表明了矩阵C就是矩阵A中各列向量的线性组合，而B其实在告诉我们，要以什么样的方式组合A中的列向量。

2）行组合的角度：同理，我们还学习过行向量与矩阵的乘法，按照形式，这次将矩阵A看做行向量组合就行了：

==C

之后每一个类似于B这样的行向量与矩阵之积，会得到一个行向量，各行向量最后构成了C中的各行。这里注意，C中各行都来自于B形式求得的行向量，所以是矩阵B各行的线性组合组成了C的各行。

3、分块做乘法

分块乘法就是宏观上的矩阵乘法，比如现在有一个 50\*50 的矩阵与 50\*50 矩阵相乘，一个一个进行运算很麻烦，尤其是如果矩阵在某一区域上有一定的性质， 那么我们可以将其分块，如：

=

其中的A与B都是划分之后的一块块矩阵，如：得到的各个部分，而C=AB+AB,和矩阵乘法的计算步骤一样，只是这里的AB，AB都是矩阵之间的乘法而已。只要A与B分块相互匹配，就可以用这样的分块乘法求解。

4、逆矩阵求解

之前我们在基础知识的部分已经介绍过了有关的逆矩阵的知识。对于一个方阵A,如果A可逆，就有这样一个A使 AA=E=AA。

如：，如果通过行列式的角度，这个矩阵对应的行列式为0，所以矩阵不可逆，但是我们这里不从行列式的角度，我们换一个角度。我们看到这个矩阵中含有两个列向量和这两个向量，它们之间互成倍数，也就是说这两个向量之一对其线性组合无意义，那么这个A不可能有逆矩阵，换句话说：

若存在非零向量x，使得Ax=0,那么A就不可能有逆矩阵

这个是为什么呢？其实，原因很简单，如果A有逆矩阵，在Ax=0这个等式两端同时乘以A，就有 AAx=E=零向量，但是E不可能是零向量，自相矛盾，所以A没有逆矩阵。

再看看矩阵，由于两个列向量线性相关，一定有一个x，使得Ax=0。

下面将介绍求解逆矩阵的方法，其实求逆矩阵就是解方程组的过程。

例：求解的逆矩阵。

=E=

A A

从列向量的角度看来，得到两个方程：

= =直接解这个方程组就好了。

但是这个对于低阶矩阵还好，对于高阶矩阵计算量未免太大了，所以这里介绍高斯—若尔当 方法。还是用上面的例子，两个方程；

= =

这个方法就是可以同时处理两个方程组，即使用增广矩阵联系两个方程增广矩阵：



接下来进行行变换，将直线左侧消为单位矩阵E，此时右侧矩阵就是逆矩阵。

逆矩阵为： 

检验一下：AA=E 正确

所以说，高斯—若尔当 方法在求逆矩阵的过程是非常普遍使用的。

接下我们来论证它的合理性：

我们知道，上面这个过程，对进行消元处理使其变为单位矩阵E,就相当于左乘一大推消元矩阵，假设为P，就有P=E,那么P肯定就是逆矩阵A，再看直线右侧，单位矩阵经历了与同样的消元过程，最后相当于结果就是PE=A,那么直线右侧就是其逆矩阵就对了。

[https://mp.weixin.qq.com/s?\_\_biz=MzI4MDYzNzg4Mw==&mid=2247485310&idx=1&sn=72a7b0d6142c3b8cad385ff9ea239f01&chksm=ebb43faadcc3b6bce80291ab19dfd6342842a2ec658a48197d93c18b288c3033af8da0f0aba5&scene=21#wechat\_redirect](https://mp.weixin.qq.com/s?__biz=MzI4MDYzNzg4Mw==&mid=2247485310&idx=1&sn=72a7b0d6142c3b8cad385ff9ea239f01&chksm=ebb43faadcc3b6bce80291ab19dfd6342842a2ec658a48197d93c18b288c3033af8da0f0aba5&scene=21" \l "wechat_redirect)

## 2.1.5 第五节 特征分解

许多数学对象可以通过将它们分解成多个组成部分，或者找到它们的一些属性而更好地理解，这些属性是通用的，而不是由我们选择表示它们的方式引起的。

例如，整数可以分解为质数。我们可以用十进制或二进制等不同方式表示整数12，但质因数分解永远是对的 12 = 2 × 3 × 3。从这个表示中我们可以获得一些有用的信息，比如 12 不能被 5 整除，或者 12 的倍数可以被 3 整除。

正如我们可以通过分解质因数来发现整数的一些内在性质，我们也可以通过分解矩阵来发现矩阵表示成数组元素时不明显的函数性质。特征分解是使用最广的矩阵分解之一，即我们将矩阵分解成一组特征向量和特征值。

方阵 A 的特征向量是指与 A 相乘后相当于对该向量进行放缩的非零向量 P：

AP = P

数值 被称为这个特征向量对应的特征值。（类似地，我们也可以定义左奇异向量p⊤A = p⊤ ，但是通常我们更关注右奇异向量。

如果p是A的特征向量，那么任何放缩后的向量 sp（≠0）也是A的特征向量。此外，sp和 p有相同的特征值。基于这个原因，通常我们只考虑单位特征向量。

假设矩阵 A 有 n 个线性无关的特征向量 {v (1) ,...,v (n) }，对应着特征值

{λ ,...,λ}。我们将特征向量连接一个矩阵，使得每一列是一个特征向量：V =

[v (1) ,...,v (n) ]. 类似地，我们也可以将特征值连接成一个向量 λ = [λ 1 ,...,λ n ] ⊤ 。因此 A 的特征分解可以记作

A = Vdiag(λ)V

我们已经看到了构建具有特定特征值和特征向量的矩阵，能够使我们在目标方向上延伸空间。然而，我们也常常希望将矩阵分解 (decompose) 成特征值和特征向量。这样可以帮助我们分析矩阵的特定性质，就像质因数分解有有助于我们理解整数。

不是每一个矩阵都可以分解成特征值和特征向量。在某些情况下，特征分解会涉及到复数，而非实数。幸运的是，在本书中我们通常只需要探讨一类有简单分解的矩阵。具体地，每个实对称矩阵都可以分解成实特征向量和实特征值：

A = QΛQ⊤

其中 Q 是 A 的特征向量组成的正交矩阵，Λ 是对角矩阵。特征值 Λi,i 对应的特征向量是矩阵 Q 的第 i 列，记作 Q i 。因为 Q 是正交矩阵，我们可以将 A 看作是沿方向 v (i) 延展 λi 倍的空间。

虽然任意一个实对称矩阵 A 都有特征分解，但是特征分解可能并不唯一。如果两个或多个特征向量拥有相同的特征值，那么这组特征向量生成子空间中，任意一组正交向量都是该特征值对应的特征向量。因此，我们可以使用任意一组正交向量作为 Q 的特征向量。按照惯例，我们通常按降序排列 Λ 的条目。在该约定下，特征分解唯一当且仅当所有的特征值都是唯一的。

矩阵的特征分解给了我们很多关于矩阵的有用信息。矩阵是奇异的当且仅当含有零特征值。实对称矩阵的特征分解也可以用于优化二次方程 f(x) = x⊤Ax，其中限制 =1。当 x 等于 A 的某个特征向量时，f 将返回对应的特征值。在限制条件下，函数 f 的最大值是最大特征值，最小值是最小特征值。

所有特征值都是正数的矩阵被称为正定；所有特征值都是非负数的矩阵被称为半正定。同样地，所有特征值都是负数的矩阵被称为负定 ；所有特征值都是非正数的矩阵被称为半负定。半正定矩阵受到关注是因为它们保证 ∀x,x⊤Ax ≥ 0。此外，正定矩阵还保证 x ⊤Ax = 0 ⇒ x = 0。

https://github.com/exacity/deeplearningbook-chinese

## 概率论基本知识

### 概率、条件概率

### 贝叶斯

## 机器学习基本知识

### 何为机器学习

* 什么是学习

孔子云：“学而时习之，不亦说乎。”本意为：学了之后及时、经常地进行温习和实习不是一件很愉快的事吗？而自古以来“学”意为自学或有人教你学，偏重于知识理论的学习；“习”意为巩固知识、技能的行为，偏重于复习行动的实践；“学习”则意为自我概念的变化、价值与潜能的实现。既是概念的变化、又是潜能的实践，所以“学而时习之，不亦说乎”的新解就是：“学习”不仅要学习理论知识，还要经常动手实践，将理论和实践结合在一起不是一件很有成就感的事吗？温菲尔德也说过：“学习是由练习或经验引起的行为或知识的较持久的变化。”可见，“学习”的本质其实就是一个从陌生到熟悉再到掌握的过程、学习理论和实践巩固相结合的过程。

* 什么是机器学习

定义： 对于某类任务 T 和性能度量 P，如果一个计算机程序在 T 上以 P 衡量的性能随着经验 E 而自我完善，那么我们称这个计算机程序在从经验 E。

### 机器学习的主要任务

* 分类
* 回归
* 聚类

### 机器学习的基本步骤

* 数据的采集
* 数据的前处理
* 机器学习算法的处理
* 算法的对比
* 结果评价

# scikit-learn基本知识

## sklearn结构

起源

主要模块

## 加载范例数据库

# 预处理

## 标准化、归一化与正则化

数据的标准化（normalization）是将数据按照一定规则缩放，使之落入一个小的特定区间。这样去除数据的单位限制，将其转化为无量纲的纯数值，便于不同单位或量级的指标能够进行比较和加权。

前数据标准化方法有多种，归结起来可以分为直线型方法(如极值法、标准差法)、折线型方法(如三折线法)、曲线型方法(如半正态性分布)。不同的标准化方法，对系统的评价结果会产生不同的影响，然而不幸的是，在数据标准化方法的选择上，还没有通用的法则可以遵循。

* + 归一化的目标主要有两个：

1、把数变为（0，1）之间的小数

主要是为了数据处理方便提出来的，把数据映射到0～1范围之内处理，更加便捷快速，应该归到数字信号处理范畴之内。

2、把有量纲表达式变为无量纲表达式

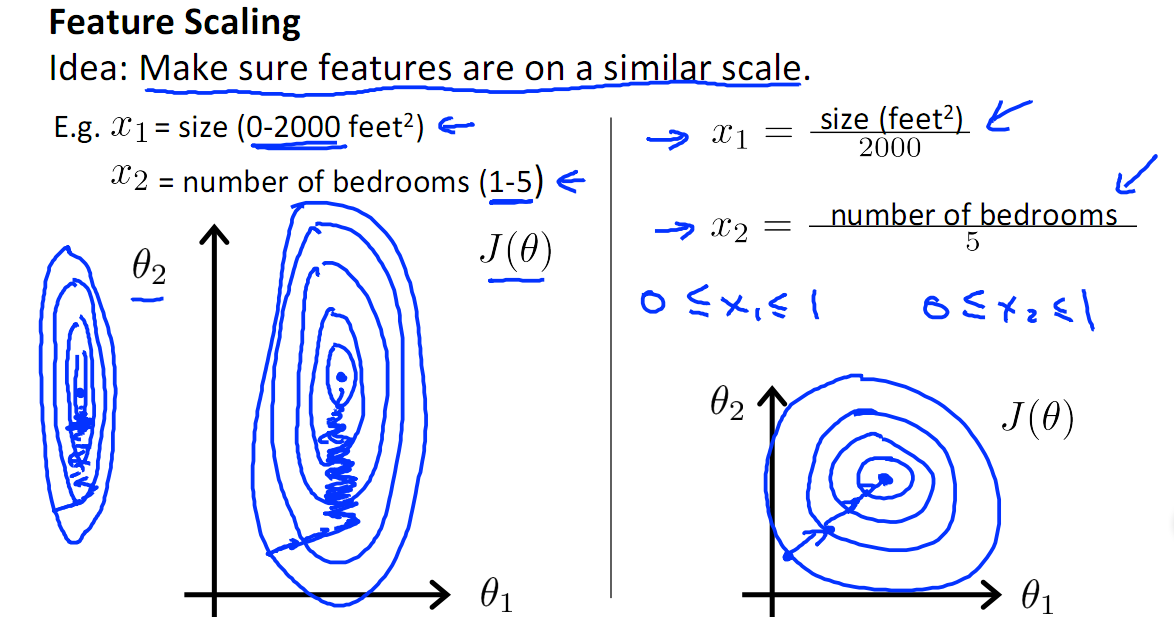
化为无量纲的表达式，成为纯量。 比如，复数阻抗可以归一化书写：Z = R + jωL = R(1 + jωL/R) ，复数部分变成了纯数量了，没有量纲。

另外，微波之中也就是电路分析、信号系统、电磁波传输等，有很多运算都可以如此处理，既保证了运算的便捷，又能凸现出物理量的本质含义。

* + 归一化后有两个好处

1. 提升模型的收敛速度

如下图，x1的取值为0-2000，而x2的取值为1-5，假如只有这两个特征，对其进行优化时，会得到一个窄长的椭圆形，导致在梯度下降时，梯度的方向为垂直等高线的方向而走之字形路线，这样会使迭代很慢，相比之下，右图的迭代就会很快（理解：也就是步长走多走少方向总是对的，不会走偏）

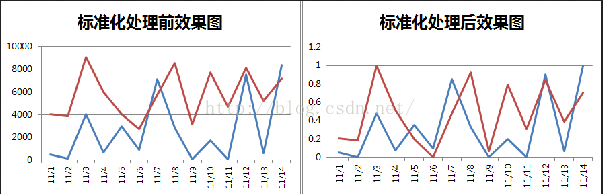


2.提升模型的精度

**P1137 25.2 Importance of Feature Scaling**

归一化的另一好处是提高精度，这在涉及到一些距离计算的算法时效果显著，比如算法要计算欧氏距离，上图中x2的取值范围比较小，涉及到距离计算时其对结果的影响远比x1带来的小，所以这就会造成精度的损失。所以归一化很有必要，他可以让各个特征对结果做出的贡献相同。

在多指标评价体系中，由于各评价指标的性质不同，通常具有不同的量纲和数量级。当各指标间的水平相差很大时，如果直接用原始指标值进行分析，就会突出数值较高的指标在综合分析中的作用，相对削弱数值水平较低指标的作用。因此，为了保证结果的可靠性，需要对原始指标数据进行标准化处理。



在数据分析之前，我们通常需要先将数据标准化，利用标准化后的数据进行数据分析。数据标准化也就是统计数据的指数化。数据标准化处理主要包括数据同趋化处理和无量纲化处理两个方面。数据同趋化处理主要解决不同性质数据问题，对不同性质指标直接加总不能正确反映不同作用力的综合结果，须先考虑改变逆指标数据性质，使所有指标对测评方案的作用力同趋化，再加总才能得出正确结果。数据无量纲化处理主要解决数据的可比性。经过上述标准化处理，原始数据均转换为无量纲化指标测评值，即各指标值都处于同一个数量级别上，可以进行综合测评分析。

从经验上说，归一化是让不同维度之间的特征在数值上有一定比较性，可以大大提高分类器的准确性。

**归一化是把数据映射到（0，1）区间内，一般方法是(x-min(x))/(max(x)-min(x)),而标准化是一种统计的处理，基于正态分布的假设，但是即使数据不服从正态分布，也可以用此法，区别是标准化后的数据可正可负，但是一般绝对值不会太大。具体的办法是(x-mean(x))/std(x)。标准化后的数据的均值＝0，标准差＝1**

**归一化（Normalization）**

        1.把数据变为（0，1）之间的小数。主要是为了方便数据处理，因为将数据映射到0～1范围之内，可以使处理过程更加便捷、快速。

        2.把有量纲表达式变换为无量纲表达式，成为纯量。经过归一化处理的数据，处于同一数量级，可以消除指标之间的量纲和量纲单位的影响，提高不同数据指标之间的可比性。

        主要算法：

        1.线性转换，即min-max归一化（常用方法）

        y=(x-min)/(max-min)

        2. 对数函数转换

        y=log10(x)

        3.反余切函数转换

        y=atan(x)\*2/PI

**标准化（Standardization）**

        数据的标准化是将数据按比例缩放，使之落入一个小的特定区间。

        主要方法：

        1.z-score标准化，即零-均值标准化（常用方法）

        y=(x-μ)/σ

        是一种统计的处理，基于正态分布的假设，将数据变换为均值为0、标准差为1的标准正态分布。但即使数据不服从正态分布，也可以用此法。特别适用于数据的最大值和最小值未知，或存在孤立点。

        2.小数定标标准化

        y=x/10^j  （j确保max(|y|)<1）

        通过移动x的小数位置进行标准化

        3.对数Logistic模式

        y=1/(1+e^(-x))

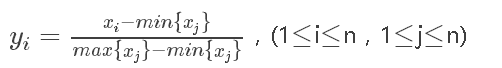
**正则化（Regularization）**

        用一组与原不适定问题相“邻近”的适定问题的解，去逼近原问题的解，这种方法称为正则化方法。如何建立有效的正则化方法是反问题领域中不适定问题研究的重要内容。通常的正则化方法有基于变分原理的Tikhonov 正则化、各种迭代方法以及其它的一些改进方法。

**总的来说**，归一化是为了消除不同数据之间的量纲，方便数据比较和共同处理，比如在神经网络中，归一化可以加快训练网络的收敛性；标准化是为了方便数据的下一步处理，而进行的数据缩放等变换，并不是为了方便与其他数据一同处理或比较，比如数据经过零-均值标准化后，更利于使用标准正态分布的性质，进行处理；正则化而是利用先验知识，在处理过程中引入正则化因子(regulator)，增加引导约束的作用，比如在逻辑回归中使用正则化，可有效降低过拟合的现象。

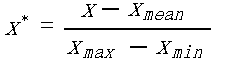
### 离差标准化

通常意义上的归一化，也称**Min-Max标准化，**离差标准化是将某变量中的观察值减去该变量的最小值，然后除以该变量的极差。即



其中max{xj}为样本数据的最大值，min{xj}为样本数据的最小值。

如果想要将数据映射到[-1,1]，则将公式换成：



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。**

当然，在构造类对象的时候也可以直接指定最大最小值的范围：feature\_range=(min, max)，此时应用的公式变为：

X\_std=(X-X.min(axis=0))/(X.max(axis=0)-X.min(axis=0))

X\_scaled=X\_std/(max-min)+min

经过离差标准化后，各种变量的观察值的数值范围都将在〔0，1〕之间，并且经标准化的数据都是没有单位的纯数量。离差标准化是消除量纲（单位）影响和变异大小因素的影响的最简单的方法。有一些关系系数（例如绝对值指数尺度）在定义时就已经要求对数据进行离差标准化，但有些关系系数的计算公式却没有这样要求，当选用这类关系系数前，不妨先对数据进行标准化，看看分析的结果是否为有意义的变化。

将属性缩放到一个指定的最大和最小值（通常是1-0）之间，这可以通过preprocessing.MinMaxScaler类实现。

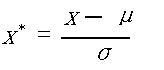
使用这种方法的目的包括：

1、对于方差非常小的属性可以增强其稳定性。

2、维持稀疏矩阵中为0的条目。

### 标准差标准化

也叫Z-score 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：



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。

经过 Z-score 标准化后，各变量将有约一半观察值的数值小于0，另一半观察值的数值大于0，变量的平均数为0，标准差为1。经标准化的数据都是没有单位的纯数量。它是当前用得最多的数据标准化方法。如果特征非常稀疏，并且有大量的0（现实应用中很多特征都具有这个特点），Z-score 标准化的过程几乎就是一个除0的过程，结果不可预料。

大于0说明高于平均水平，小于0说明低于平均水平。

z-score标准化方法适用于属性A的最大值和最小值未知的情况，或有超出取值范围的离群数据的情况。

示例：

Sklearn的StandardScaler提供了对数据进行**标准差标准化处理**的功能，下面我们用PCA算法来演示标准化处理的影响，数据我们选用的是UCI数据集中的Wine数据集，该数据集的特征值分别代表了葡萄酒的酒精度、苹果酸等的含量，因此取值范围差异较大，例如酒精度的取值范围在（，）之间，苹果酸的取值范围为（，）之间。代码如下：

from \_\_future\_\_ import print\_function

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.naive\_bayes import GaussianNB

from sklearn import metrics

import matplotlib.pyplot as plt

from sklearn.datasets import load\_wine

from sklearn.pipeline import make\_pipeline

print(\_\_doc\_\_)

RANDOM\_STATE = 42

FIG\_SIZE = (10, 7)

features, target = load\_wine(return\_X\_y=True)

# Make a train/test split using 30% test size

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, target,

test\_size=0.30,

random\_state=RANDOM\_STATE)

# Fit to data and predict using pipelined GNB and PCA.

unscaled\_clf = make\_pipeline(PCA(n\_components=2), GaussianNB())

unscaled\_clf.fit(X\_train, y\_train)

pred\_test = unscaled\_clf.predict(X\_test)

# Fit to data and predict using pipelined scaling, GNB and PCA.

std\_clf = make\_pipeline(StandardScaler(), PCA(n\_components=2), GaussianNB())

std\_clf.fit(X\_train, y\_train)

pred\_test\_std = std\_clf.predict(X\_test)

# Show prediction accuracies in scaled and unscaled data.

**print**('**\n**Prediction accuracy for the normal test dataset with PCA')

**print**('{:.2%}**\n**'.format(metrics.accuracy\_score(y\_test, pred\_test)))

**print**('**\n**Prediction accuracy for the standardized test dataset with PCA')

**print**('{:.2%}**\n**'.format(metrics.accuracy\_score(y\_test, pred\_test\_std)))

# Extract PCA from pipeline

pca = unscaled\_clf.named\_steps['pca']

pca\_std = std\_clf.named\_steps['pca']

# Show first principal componenets

**print**('**\n**PC 1 without scaling:**\n**', pca.components\_[0])

**print**('**\n**PC 1 with scaling:**\n**', pca\_std.components\_[0])

# Scale and use PCA on X\_train data for visualization.

scaler = std\_clf.named\_steps['standardscaler']

X\_train\_std = pca\_std.transform(scaler.transform(X\_train))

# visualize standardized vs. untouched dataset with PCA performed

fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=FIG\_SIZE)

**for** l, c, m **in** zip(range(0, 3), ('blue', 'red', 'green'), ('^', 's', 'o')):

ax1.scatter(X\_train[y\_train == l, 0], X\_train[y\_train == l, 1],

color=c,

label='class %s' % l,

alpha=0.5,

marker=m

)

**for** l, c, m **in** zip(range(0, 3), ('blue', 'red', 'green'), ('^', 's', 'o')):

ax2.scatter(X\_train\_std[y\_train == l, 0], X\_train\_std[y\_train == l, 1],

color=c,

label='class %s' % l,

alpha=0.5,

marker=m

)

ax1.set\_title('Training dataset after PCA')

ax2.set\_title('Standardized training dataset after PCA')

**for** ax **in** (ax1, ax2):

ax.set\_xlabel('1st principal component')

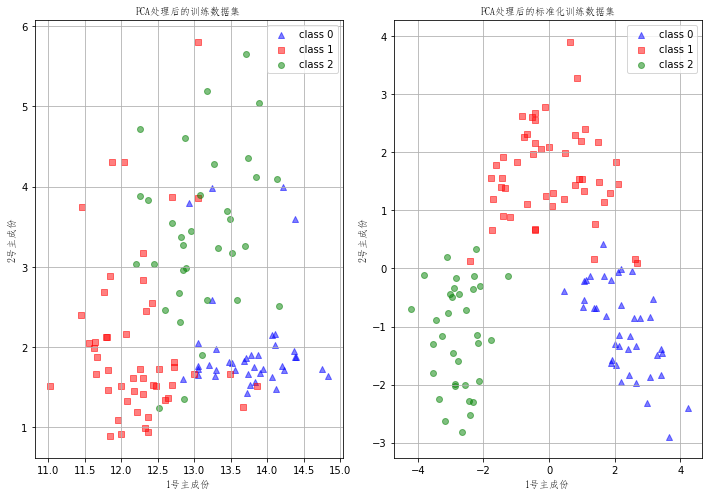
ax.set\_ylabel('2nd principal component')

ax.legend(loc='upper right')

ax.grid()

plt.tight\_layout()

plt.show()



结果是显著的，我们可以看出第一个主成份在非标准化数据集

### 规范化

规范化是将不同变化范围的值映射到相同的固定范围，常见的是[0,1]，此时也称为归一化。《机器学习》周志华

正则化的过程是将每个样本缩放到单位范数（每个样本的范数为1），如果后面要使用如二次型（点积）或者其它核方法计算两个样本之间的相似性这个方法会很有用。

将每个样本变换成unit norm。Normalization主要思想是对每个样本计算其p-范数，然后对该样本中每个元素除以该范数，这样处理的结果是使得每个处理后样本的p-范数（l1-norm,l2-norm）等于1。

|  |  |
| --- | --- |
| 1  2 | X = [[ 1, -1, 2],[ 2, 0, 0], [ 0, 1, -1]]  sklearn.preprocessing.normalize(X, norm='l2') |

得到：

|  |  |
| --- | --- |
| 1 | array([[ 0.40, -0.40, 0.81], [ 1, 0, 0], [ 0, 0.70, -0.70]]) |

可以发现对于每一个样本都有，0.4^2+0.4^2+0.81^2=1,这就是L2 norm，变换后每个样本的各维特征的平方和为1。类似地，L1 norm则是变换后每个样本的各维特征的绝对值和为1。还有max norm，则是将每个样本的各维特征除以该样本各维特征的最大值。

在度量样本之间相似性时，如果使用的是二次型kernel，需要做Normalization。

Scaling inputs to unit norms is a common operation for text classification or clustering for instance. For instance the dot product of two l2-normalized TF-IDF vectors is the cosine similarity of the vectors and is the base similarity metric for the Vector Space Model commonly used by the Information Retrieval community。P1999

**sklearn.preprocessing.Normalizer**

## 离散特征二值化

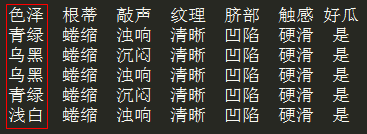
scikit-learn中离散特征二值化

【http://blog.csdn.net/u012328159/article/details/71617381?utm\_source=itdadao&utm\_medium=referral】

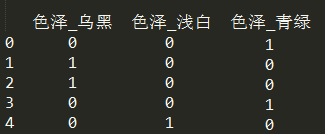
最近在看西瓜书用scikit-learn中的CART去跑西瓜数据集，结果遇到麻烦了，西瓜数据集特征不光离散的，而且还是中文的。。（PS：其实我们的数据集中特征值常常是离散的类别，这个很正常），但在scikit-learn中不支持这种离散的类别特征作为输入，这点不得不说weka的人性化，直接输入原始数据集就可以了。。为了解决这个问题，就要用到独热编码（One-Hot Encoding），下面来说下这个One-Hot Encoding：

* **One-Hot Encoding**

关于one-hot编码的定义，简单的说就是用n位bit（0和1）表示n个状态，并且任意时候只有一位有效数字。直接看例子，我觉得举例子比较好理解。用西瓜数据集（只选了部分样本）举例子吧，西瓜数据集如下：



就拿“色泽”这个属性来看看one-hot encoding，显然色泽有“青绿，乌黑，浅白”三个不同的离散值（类别），那么就需要三位二进制数来表示这个特征，则边玩码后应该是这样的:



来讲下上图中数据是怎样产生的，即one-hot encoding是怎样编码的。显然“色泽”这个属性的三个离散值按照[乌黑，浅白，青绿]排列的，来看第一个样本（图中index为0的）色泽=青绿，因此和[乌黑，浅白，青绿]一比对，对应的编码就是001，依次类推2,3,4,5个样本色泽这个属性的编码分为为100,100,001,010。其实说白了就是把一个属性变成了“三个”，因为模型才不关心你几个特征呢，它只管你给他他能计算的，能挖掘出信息的。这就是one-hot编码的大概内容。下面回到scikit-learn中的实现：自然而言想到sklearn的 的Preprocessing模块中的OneHotEncoder函数，这个函数的具体细节大家去看文档[链接](http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html" \l "sklearn.preprocessing.OneHotEncoder" \t "_blank)。但这个函数最坑的地方在于没法对字符串类型的变量进行编码。。这个是最让人无语的地方，不知道后续的更新版本中会不会支持字符串类型的变量。下面只能造个例子来展示下OneHotEncoder怎么用：

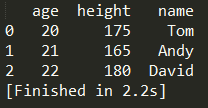
* 造的数据集

**[python]** [view plain](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "copy)

[print](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "print)[?](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "?)

1. **import** pandas as pd
2. **from** sklearn.preprocessing **import** OneHotEncoder
3. **from** sklearn.preprocessing **import** LabelBinarizer
4. **from** sklearn.feature\_extraction **import** DictVectorizer
6. data = pd.DataFrame({'name':['Tom','Andy','David'],'age':[20,21,22],'height':[175,165,180]})
7. **print**(data)

运行结果如下：



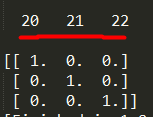
下面使用OneHotEncoder类来对特征进行onehot编码：

**[python]** [view plain](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "copy)

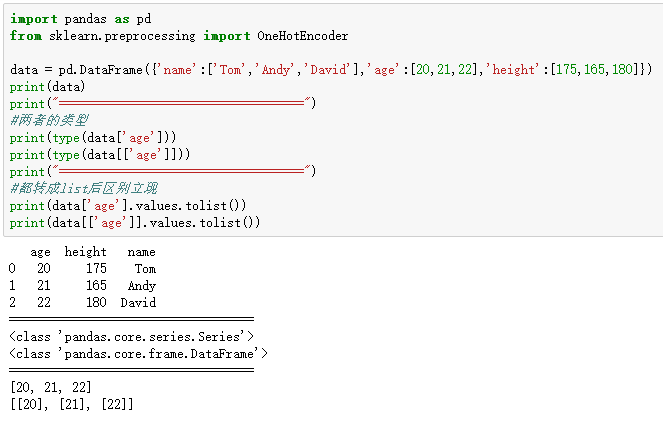
[print](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "print)[?](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "?)

1. #新版本不允许传入一维数组，因此传入data[['age']],sparse=False为不使用稀疏矩阵表示
2. arr = OneHotEncoder(sparse = False).fit\_transform(data[['age']])
3. **print**(arr)

输出结果为（OneHotEncoder会按属性排序，形成有序值，因此下面三列是20,21,22）：



讲到这个地方，要插一个插曲，也是遇到的坑，记录一下，就是当时OneHotEncoder(sparse = False).fit\_transform(data['age'])参数传的是data['age']，然后有提示就是新版不再允许传入一维数组，一直没搞懂data['age']和data[['age']]的区别，请教石神后，方知data['age']返回的是series，data[['age']]因为传入的是列表['age']所以返回的是dataframe，知道这点区别，为了更加弄明白本质区别是什么，做了实验，相信看完下面的对比，还有和我一样不明白的同学能够有一个清晰的认识：



一目了然，转成列表后，一个是一维的一个是二维的。

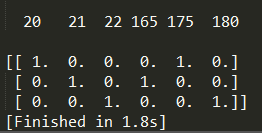
回到正题，还可以传入多个特征一次性处理完：

**[python]** [view plain](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "view plain) [copy](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "copy)

[print](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "print)[?](http://blog.csdn.net/u012328159/article/details/71617381?utm_source=itdadao&utm_medium=referral" \t "_blank" \o "?)

1. arr = OneHotEncoder(sparse = False).fit\_transform(data[['age','height']])
2. **print**(arr)

结果为（注意：列名20 21.....是我为了方便大家看加的，并不是实际输出结果）：



* **LabelBinarizer()**

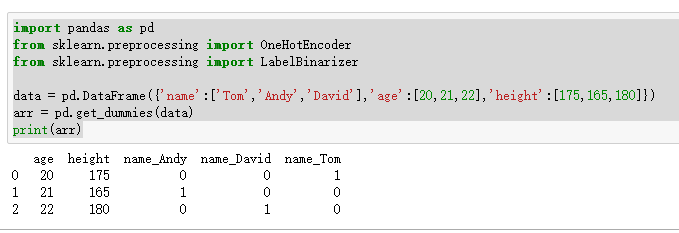
OneHotEncoder()千般好万般好，唯一的不好就是这货没法处理字符串类型的特征值。。比如我们处理‘name’属性就会报错。。提示不能处理字符串类型。。没办法生活还得继续啊，只能曲线救国了，查看文档后发现了LabelBinarizer()类，大家看名字也能看出是干嘛的，对，就是用于类别（label）编码的，上代码：



发现完全可以解决问题，但 新问题是LabelBinarizer()只能传入一维数组处理，不接受二维的，也就是说如果你想一次对多个特征处理，不能全部传入。。。博主试图用pandas的apply（）解决，但无奈水平不到家，没解决，有解决的可以在下方评论贴出，只求别太麻烦。

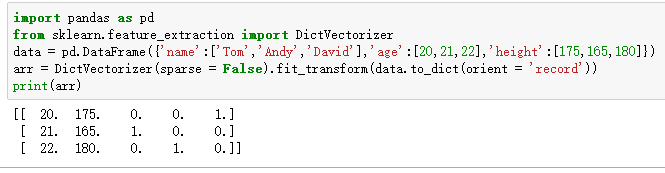
* 新大陆1(get\_dummies())

经过探索发现pandas的get\_dummies()函数可以完美解决问题，这个函数的功能用官方文档的话就是Convert categorical variable into dummy/indicator variables，把类别型的变量转换成dummy variables（用0和1表示类别的出现）。上代码看效果吧：



* **新大陆2(**DictVectorizer**)**

翻了参考资料,发现sklearn的feature\_extraction模块中提供了DictVectorizer类，即把字典向量化，采用0/1二值向量化。上代码：



### 特征二值化（Binarization）

给定阈值，将特征转换为0/1

|  |  |
| --- | --- |
| 1  2 | binarizer = sklearn.preprocessing.Binarizer(threshold=1.1)  binarizer.transform(X) |

### 标签二值化（Label binarization）

lb = sklearn.preprocessing.LabelBinarizer()

### 类别特征编码

有时候特征是类别型的，而一些算法的输入必须是数值型，此时需要对其编码。

|  |  |
| --- | --- |
| 1  2  3 | enc = preprocessing.OneHotEncoder()  enc.fit([[0, 0, 3], [1, 1, 0], [0, 2, 1], [1, 0, 2]])  enc.transform([[0, 1, 3]]).toarray() #array([[ 1., 0., 0., 1., 0., 0., 0., 0., 1.]]) |

上面这个例子，第一维特征有两种值0和1，用两位去编码。第二维用三位，第三维用四位。

### .标签编码（Label encoding）

le = sklearn.preprocessing.LabelEncoder()

le.fit([1, 2, 2, 6])

le.transform([1, 1, 2, 6]) #array([0, 0, 1, 2])

#非数值型转化为数值型

le.fit(["paris", "paris", "tokyo", "amsterdam"])

le.transform(["tokyo", "tokyo", "paris"]) #array([2, 2, 1])

## 特征中含异常值时

|  |  |
| --- | --- |
| 1 | sklearn.preprocessing.robust\_scale |

此Scaler删除中位数并根据分位数范围缩放数据（默认为**四分位距**：Interquartile Range）。 IQR是第一个四分之一（第25个分位数）和第三个四分位数（第75个分位数）之间的范围。

数据集的标准化是许多机器学习估计器的常见要求。 通常，这是通过去除平均值和缩放到单位方差来完成的。 然而，异常值常常会影响样本平均值/方差为负。 在这种情况下，中位数和四分位数范围往往会给出更好的结果。

## 生成多项式特征

这个其实涉及到特征工程了，多项式特征/交叉特征。

|  |  |
| --- | --- |
| 1  2 | poly = sklearn.preprocessing.PolynomialFeatures(2)  poly.fit\_transform(X) |

原始特征：[http://img.blog.csdn.net/20160203140705738](http://img.blog.csdn.net/20160203140705738)

转化后：[http://img.blog.csdn.net/20160203140725613](http://img.blog.csdn.net/20160203140725613)

## 插补处理

preprocessing.Imputer

## 独热编码

**sklearn.preprocessing.OneHotEncoder**

## 特征选择

## 降维

# 一般线性模型

## 线性模型概述

## 线性判别分析 P60

# k-邻近算法

## k-邻近算法概述

### 基本思想

### 算法概述

## 算法范例：k-邻近算法应用范例

## 小结

# 支持向量机

## 概述

## 支持向量机算法

## 算法范例：支持向量机分类应用范例

## 小结

# 决策树

## 简介

### 决策树表示法

## 决策树算法

## 决策树学习的适用问题

## 算法范例：决策树回归范例

## 小结

# 人工神经网络

## 简介

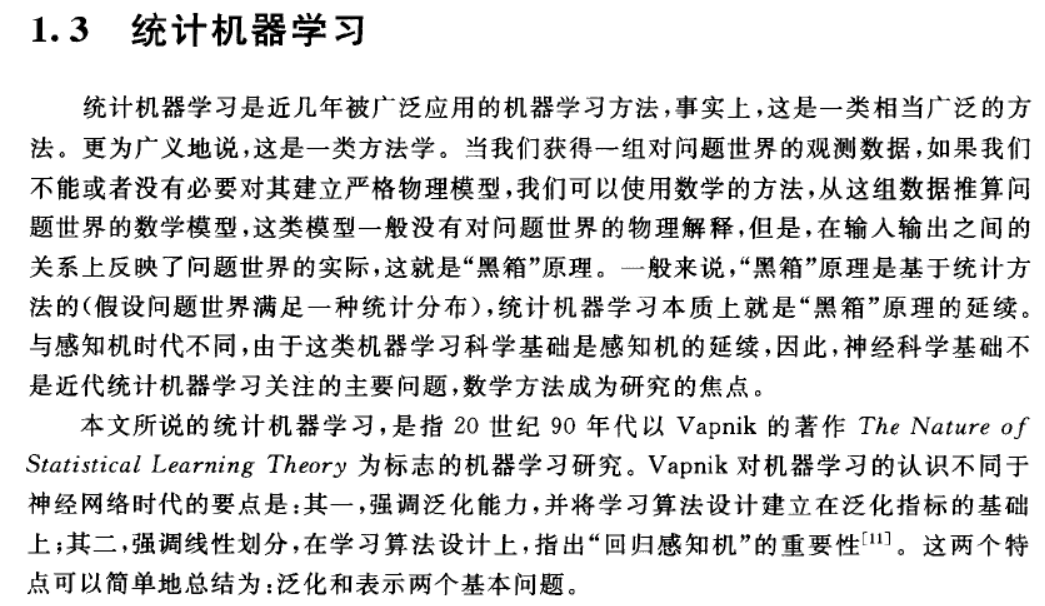
## 人工神经网络算法

## 人工神经网络算法适用的问题

## 算法范例：BP神经网络应用范例

## 小结

素材



# 模型评价

## 模型评价指标

## 模型评价范例

## 模型选择范例