Contents

[Contents 1](#_Toc525313097)

[《数论》 2](#_Toc525313098)

[<命令行中的数据科学> 5](#_Toc525313099)

[1. Linux内置命令 5](#_Toc525313100)

[2. csvkit 8](#_Toc525313101)

[3. JSON处理器，包括分片、过滤、转换等等 10](#_Toc525313102)

[4. scrape – 用XPath和CSS选择器进行HTML信息提取的工具 10](#_Toc525313103)

[5. XML2JSON xml <-> json 10](#_Toc525313104)

[6. json2csv 10](#_Toc525313105)

[python 12](#_Toc525313106)

[python 工具包 12](#_Toc525313107)

[numPy ndarray（N-dimensional array object） 12](#_Toc525313108)

[matplotlib 15](#_Toc525313109)

[graphviz 17](#_Toc525313110)

[pandas 17](#_Toc525313111)

[scikit-learn 19](#_Toc525313112)

[scipy 20](#_Toc525313113)

[SymPy 27](#_Toc525313114)

[《数据科学导论》 29](#_Toc525313115)

[《Python数据科学指南》 29](#_Toc525313116)

[《数据科学入门》 29](#_Toc525313117)

[install & upgrade 30](#_Toc525313118)

[数据集 30](#_Toc525313119)

[数据预处理 32](#_Toc525313120)

[特征创建 35](#_Toc525313121)

[特征降维 35](#_Toc525313122)

[特征选择 38](#_Toc525313123)

[异常检测 39](#_Toc525313124)

[验证 40](#_Toc525313125)

[机器学习 42](#_Toc525313126)

[NLP 47](#_Toc525313127)

[R 48](#_Toc525313128)

[install R and RStudio 48](#_Toc525313129)

[<R语言初学者指南> 49](#_Toc525313130)

[R语法 49](#_Toc525313131)

[数据获取 51](#_Toc525313132)

[数据预处理 （一般针对数据框） 51](#_Toc525313133)

[数据分析 52](#_Toc525313134)

[数据可视化 52](#_Toc525313135)

[<R and Ruby数据分析之旅> 53](#_Toc525313136)

[Spark 55](#_Toc525313137)

[《Spark快速大数据分析》 55](#_Toc525313138)

[install spark 56](#_Toc525313139)

[命令行（交互式分析数据） 56](#_Toc525313140)

[脚本（独立应用） 56](#_Toc525313141)

[RDD编程 57](#_Toc525313142)

# 《数论》

倍数，公倍数，最小公倍数(两个或多个整数的公倍数里最小的一个)

约数(因数), 公因数,最大公因数(两个数的公因数中最大的一个)

a称为b的倍数，b称为a的约数

2的倍数：　一个数的末尾是偶数（0，2，4，6，8）

3的倍数：　一个数的各位数之和是3的倍数

5的倍数：　一个数的末尾是0或5

9的倍数：　若一个整数的数字和能被9整除

合数的倍数：分解合数为质数乘积，如

4的倍数：　一个数的末两位是4的倍数

6的倍数：　一个数只要能同时被2和3整除

8的倍数：　一个数的末三位是8的倍数

10的倍数：　若一个整数的末位是0

12的倍数：　若一个整数能被3和4整除

证明：任意两个奇数的平方差是8的倍数

设任意奇数2n+1,2m+1, 其中m,n∈N

(2m+1)2 – (2n+1)2 = 4(m+n+1)(m-n)

当m,n都是奇数或都是偶数时，m-n是偶数，被2整除

当m,n一奇一偶时，m+n+1是偶数，被2整除

最大公因数与最小公倍数求法？

短除法

最大公因数 = 2\*3 （除数乘积）

最小公倍数 = 2\*3 \* 2\*3 （除数与商乘积）

欧几里得算法

假定a>b, 令b = r0

a = r0 \* q1 + r1

r0 = r1 \* q2 + r2 直至整除为止，最后一个非零余数即为最大公因数,如

8251 = 6105\*1 + 2146

6105 = 2146\*2 + 1813

2146 = 1813\*1 + 333

1813 = 333\*5 + 148

333 = 148\*2 + 37

148 = 37\*4

则最大公因数为37

性质

a\*b = (a,b) \* [a, b] 整数a,b, 最大公因子(a,b), 最小公倍数[a,b]

算术基本定理

大于1的正整数N可以**唯一分解**成有限个质数的乘积

N = p1a1 \* p2a2 \* ...\*pnan

正因数个数 = (1+a1)(1+a2)...(1+an)

全体正因数之和 = (1+p1+p12+...p1a1)(1+p2+p22+...p2a2)...(1+pn+pn2+...pnan)

当全体正因数之和=2N时，则N为完全数

质数（素数）：只有1和它本身两个因数的自然数

合数：除了1与本身以外，还有其他约数

{质数} + {合数} + {1} = 正整数

100以内共25个质数：2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97　...

哥德巴赫猜想：是否每个大于2的偶数都可写成两个素数之和？

孪生素数就是差为2的素数对，例如11和13

大于3的素数只分布在6n-1和6n+1两数列中

所有大于10的质数中，个位数只有1,3,7,9

判断是否质数？

在一般领域，对正整数n，如果用2到 之间的所有整数去除，均无法整除，则n为质数。

完全数, 亏数: 真约数之和 < 自已

数恰好等于它的真约数(除了自身以外的约数)之和

6 = 1+2+3 {1,2,3,6}

28 = 1+2+4+7+14 = 1+2+...7

496 = 1+2+4+8+16+31+62+124+248 = 1+2+...31

8128 = 1+2+...+127

所有的完全数都是三角形数

6 = 1+2+3

28 = 1+2+...7

496 = 1+2+...31

8128 = 1+2+...+127

完全数的所有约数是调和数　（即所有约数的倒数和等于2)

1/1+1/2+1/3+1/6 = 2

1/1+1/2+1/4+1/7+1/14 = 2

…

完全数(>6)可以表示成连续奇数的立方和

28 = 13 + 33

496 = 13 + 33 + 53 + 73

8128 = 13 + 33 + 53 + 73 + ...153

完全数可以表示2的连续正整数次幂和

6 = 21 + 22

28 = 22 + 23  + 24

496 = 24 + 25 + 26 + 27 + 28

8128 = 26 + 27 + 28 + … + 212

完全数都是以6或8结尾,如果以8结尾，那么就肯定是以28结尾

亲和数　(a,b)

a的真约数之和 = b

a = b的真约数之和

(220 284), (1184 1210), (2620 2924), …

python code:

**import** numpy **as** np

**from** primefac **import** factorint, isprime

**from** numba **import** jit, cuda

@jit

**def find\_amicable\_numbers**(num):

cache = np.zeros(num, dtype=long)

**for** i **in** xrange(num):

cache[i] = calc\_divisors\_sum(i)

out = np.zeros(100, dtype=long)

count = 0

**for** i **in** xrange(1, num):

**for** j **in** xrange(i + 1, num):

**if** cache[i] == j **and** i == cache[j]:

out[count] = i

count += 1

result = []

**for** i **in** xrange(count):

result.append((out[i], cache[out[i]]))

**return** result

@jit

**def calc\_divisors\_sum**(n):

divs = [1]

high = int(math.sqrt(n))+1

**for** i **in** xrange(2, high):

**if** n % i == 0:

divs.extend([i, n/i])

**return** sum(list(set(divs)))

# <命令行中的数据科学>

数据科学是一个宽广的领域，与许多领域都有交叉，例如程序设计、数据可视化以及机器学习。

数据科学含5个步骤：**数据获取、数据清洗、数据探索、数据建模和数据解释**

数据科学是OSEMN（和 awesome 相同发音），它包括获取(Obtaining)、整理(Scrubbing)、探索(Exploring)、建模(Modeling)和翻译(iNterpreting)数据。

数据获取

1. 从其他地方（如网页或服务器）下载数据
2. 从数据库或API（如MySQL and Twitter）中查询数据
3. 从其他文件（如HTML文件或电子表格）中提取数据
4. 自己生成数据（例如读取传感器或进行调查）

获取数据的格式通常是纯文本、CSV、JSON或HTML/XML

数据清洗

在所获取的数据中，缺失值、不一致、错误、怪异字符或冗余列屡见不鲜。常见的清洗操作：**行过滤，列抽取、值替换、单词提取、缺失值处理、数据格式转换**。任何数据项目中80%的工作都是数据清洗

数据探索

查看数据，从数据中推导统计量，创建有趣的可视化

数据建模

若想解释数据或预测将要发生的事情，应为数据建立统计模型。技术包括聚类、分类、回归以及降维

数据解释：从数据中得出结论、评估结果的含义

1. Linux内置命令
2. sudo apt-get install tree

tree -L 2 只查看当前第N级的目录和文件

1. seq first increment last 产生序列

seq -f "Line %g" 10

1. Bash特性：括号扩展，会将{0..100..2}变成一列由空格分隔的数字：0 2 4 … 98 100

for i in {0..100..2}

do

echo "$i^2" | bc

done | head

$echo $RANDOM 内置伪随机数，返回0~32 767之间

1. find

$find ~ -name "\*.txt" -print #在$HOME中查.txt文件并显示

$find /home -size +512k 查大于512k的文件

1. linux命令parallel实现多进程并行计算

$sudo apt-get install parallel

并行处理命令和管道

parallel -j 2 "sh a.sh" "sh b.sh"; 进程a.sh与进程b.sh并行运行

sh c.sh 等以上进程结束，才运行进程c.sh

1. 输入的字符进行替换、压缩和删除

tr(选项)(参数)

选项：

-c：取代所有不属于第一字符集的字符

-d：删除所有属于第一字符集的字符

-s：把连续重复的字符以单独一个字符表示

-t：先删除第一字符集较第二字符集多出的字符。

参数：

字符集1：指定要转换或删除的原字符集

字符集2：指定要转换成的目标字符集。

$ echo "thissss is a text linnnnnnne." | tr -s ' sn' 压缩输入中重复的字符

this is a text line.

$ tr '[:lower:]' '[:upper:]' 小写 -> 大写

$ echo $PATH | tr : '\n' | sort 显示搜索路径

$ echo ‘hello\_world!’ | tr ‘ ‘ ‘\_’

hello\_world!

1. $ sort -nrk 3 -t: sort.txt

eee:40:5.4

eee:60:5.1

ddd:20:4.2

ccc:50:3.3

bbb:10:2.5

aaa:30:1.6

AAA:BB:CC

# -n是按照数字大小排序，-r是以相反顺序，-k是指定需要爱排序的栏位，-t指定栏位分隔符为冒号

sort -u file.txt 删除重复行

1. uniq

uniq命令用于报告或忽略文件中的重复行，一般与sort命令结合使用。

uniq(选项)(参数)

选项

-c：在每列旁边显示该行重复出现的次数

-d：仅显示重复出现的行列

-f<栏位>：忽略比较指定的栏位

-s<字符位置>：忽略比较指定的字符

-u：仅显示出一次的行列

-w<字符位置>：指定要比较的字符。

参数

输入文件：指定要去除的重复行文件。如果不指定此项，则从标准读取数据；

输出文件：指定要去除重复行后的内容要写入的输出文件。如果不指定此选项，则将内容显示到标准输出设备（显示终端）。

uniq -u file.txt 只显示单一行：

sort file.txt | uniq -c 统计各行在文件中出现的次数：

sort file.txt | uniq -d 在文件中找出重复的行：

1. grep

grep（global search regular expression(RE) and print out the line，全面搜索正则表达式并把行打印出来）是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来

grep -E "[1-9]+"

grep match\_pattern file\_name

#只在目录中所有的.php和.html文件中递归搜索字符"main()"

grep "main()" . -r --include \*.{php,html}

#在搜索结果中排除所有README文件

grep "main()" . -r --exclude "README"

#在搜索结果中排除filelist文件列表里的文件

grep "main()" . -r --exclude-from filelist

1. awk '{pattern + action}' {filenames}

是一个强大的文本分析工具，相对于grep的查找，sed的编辑，awk在其对数据分析并生成报告时，显得尤为强大。简单来说awk就是把文件逐行的读入，以空格为默认分隔符将每行切片，切开的部分再进行各种分析处理。

命令行方式

awk [-F field-separator] 'commands' input-file(s)

其中，commands 是真正awk命令，[-F域分隔符]是可选的。 input-file(s) 是待处理的文件。

在awk中，文件的每一行中，由域分隔符分开的每一项称为一个域。通常，在不指名-F域分隔符的情况下，默认的域分隔符是空格。

$</etc/passwd | awk -F ':' 'BEGIN {print “start”} {print $1} END {print “end”}'

**awk工作流程是这样的：读入有'\n'换行符分割的一条记录，然后将记录按指定的域分隔符划分域，填充域，$0则表示所有域,$1表示第一个域,$n表示第n个域**。默认域分隔符是"空白键" 或 "[tab]键",所以$1表示登录用户，$3表示登录用户ip,以此类推。

1. sed

sed编辑器：可以基于输入到命令行的或是存储在命令文本文件中的命令来处理数据流中的数据。**它每次从输入中读取一行，用提供的编辑器命令匹配数据、按命令中指定的方式修改流中的数据，然后将生成的数据输出到STDOUT。**

set options script file

$echo "This is a test" | sed 's/test/big test/' 替换

$sed '3i\This is an inserted line.' data7 插入行到数据流第3行前

$sed '3a\This is an inserted line.' data7 插入行到数据流第3行后

$sed '3c\This is a changed line of text.' data7 指定行整行替换

$sed 'y/123/789/' data8 转换命令

$sed -e 's/brown/breen/; s/dog/cat/' data1 执行多个命令于文件中的每行数据上

$sed 's!/bin/bash!/bin/csh!' /etc/passwd 感叹号使路径名不需要转义

1. printf命令

printf "%-5s %-10s %-4s\n" NO Name Mark

printf "%-5s %-10s %-4.2f\n" 01 Tom 90.3456

1. cut

cut的工作就是“剪”，具体的说就是在文件中负责剪切数据用的。cut是以每一行为一个处理对象的

cut命令主要是接受三个定位方法：

第一，字节（bytes），用选项-b

第二，字符（characters），用选项-c

第三，域（fields），用选项-f

$ head -n5 /etc/passwd |cut -d : -f 1,3-5

root:0:0:root

bin:1:1:bin

**-d指定域分隔符，-f 指定要剪出哪几个域**

1. curl -s http//www.../\*.txt -o data/finn.txt

curl -u username:password ftp://host/file

curl -L http://t.co/ 重定向

1. csvkit

<https://csvkit.readthedocs.io/en/0.9.0/>

**$pip install csvkit**

1. Input

**in2csv**将csv,dbf, fixed, geojson, json, ndjson, xls, xlsx -> csv

$in2csv data/imdb.xlsx > data/imdb.csv

$head data/imdb.csv | cvscut -c Title,Year,Rating | csvlook

**sql2csv**利用python SQLAlchemy软件包，只需要使用一个工具就可以在许多不同的数据库上执行查询，包括MySQL, Oracle, PostgreSQL, SQLite, SQL Server and Sybase.

Execute an SQL query on a database and output the result to a CSV file.

#Load sample data into a table using csvsql and then query it using sql2csv:

$ csvsql --db "sqlite:///dummy.db" --table "test" --insert examples/dummy.csv

$ sql2csv --db "sqlite:///dummy.db" --query "select \* from test"

1. Processing

**csvclean**

cleans a CSV file of common syntax errors. Outputs [basename]\_out.csv and [basename]\_err.csv, the former containing all valid rows and the latter containing all error rows along with line numbers and descriptions

$ csvclean examples/bad.csv

$ csvclean -n examples/bad.csv

**csvcut**

Filters and truncates CSV files. Like unix “cut” command, but for tabular data:

#Print the indices and names of all columns:

$ csvcut -n \*.csv

$ csvcut -c 1,3 \*.csv

$ csvcut -c TOTAL,"State Name" \*.csv

**csvgrep**

Filter tabular data to only those rows where certain columns contain a given value or match a regular expression:

Like the unix "grep" command, but for tabular data.

$ csvgrep -c 1 -r "^I" \*.csv

**csvjoin**

Merges two or more CSV tables together using a method analogous to SQL JOIN operation. By default it performs an inner join, but full outer, left outer, and right outer are also available via flags.

$csvjoin -c "ColumnKey,Column Key" --outer file1.csv file2.csv

This command says you have two files to outer join, file1.csv and file2.csv. The key column in file1.csv is ColumnKey, the key column in file2.csv is Column Key.

**csvsort**

sort CSV files. Like unix “sort” command, but for tabular data:

$ <countries.csv csvsort -c 2

**csvstack**

stack up the rows from multiple CSV files, optionally adding a grouping value to each row:

$ csvstack -g 2009,2010 State1.csv State2.csv

1. Output (and Analysis)

**csvformat**

convert a CSV file to a custom output format.

#convert “standard” CSV file to a pipe-delimited one

$ csvformat -D "|" examples/dummy.csv

**csvjson**

converts a CSV file into JSON or GeoJSON (depending on flags):

$ <examples/realdata/FY09\_EDU\_Recipients\_by\_State.csv csvjson -i 4

**csvlook**

render a CSV file in the console as a fixed-width table.

$ csvlook examples/testfixed\_converted.csv

**csvpy**

loads a CSV file into a csvkit.CSVKitReader object and then drops into a Python shell so the user can inspect the data however they see fit

$ csvpy --dict examples/dummy.csv -v

**csvsql**

generate SQL statements for a CSV file or create execute those statements

directly on a database.

generate a SQL CREATE TABLE statement for a CSV file.

$ createdb test

$ csvsql --db postgresql:///test --table fy09 --insert FY09\_EDU\_Recipients\_by\_State.csv

use CSVSQL to “directly” query one or more CSV files. Please note that this will create an in-memory SQL database, so it won’t be very fast:

$ csvsql --query "select m.usda\_id, avg(i.sepal\_length) as mean\_sepal\_length from iris as i join irismeta as m on (i.species = m.species) group by m.species" examples/iris.csv examples/irismeta.csv

$ <data/imdb-lists.csv csvsql --query "SELECT title trim, Year, Rating FROM stdin"

**csvstat**

Prints descriptive statistics for all columns in a CSV file. Will intelligently determine the type of each column and then print analysis relevant to that type (ranges for dates, mean and median for integers, etc.):

$ csvstat examples/realdata/FY09\_EDU\_Recipients\_by\_State.csv

1. JSON处理器，包括分片、过滤、转换等等

<https://stedolan.github.io/jq/manual/>

$apt-get install jq

$<data.json python -m json.tool 格式化json

$<data.json jq . 格式化打印并高亮

$jq '.\_id' u.json 查看某个属性

$jq '.\_id, .description' u.json 查看某2个属性

jq '.versions["1.0.3"].keywords' u.json 看看一个数组

jq '.versions["1.0.3"].keywords[1]' u.json 查看一个数组的第1个值

jq '.versions["1.0.3"] | .name, .author.name' u.json 过滤属性

1. scrape – 用XPath和CSS选择器进行HTML信息提取的工具

<https://github.com/jeroenjanssens/data-science-at-the-command-line>

将python脚本scrape -> scrape命令

$ pip install cssselect

$ curl -sL 'http://en.wikipedia.org/wiki/List\_of\_countries\_and\_territories\_by\_border/area\_ratio' > wiki.html

$ < data/wiki.html grep wikitable -A 21

$ < data/wiki.html scrape -be 'table.wikitable > tr:not(:first-child)' > table.html

$ xml2json.py -t xml2json -o data/table.json data/table.html --strip\_text

$<table.json jq -c '.html.body.tr[] | {country: .td[1], border: .td[2], surface: .td[3]}' >countries.json

$ <data/countries.json json2csv -p -k border,surface >countries.csv

1. XML2JSON xml <-> json

<https://github.com/hay/xml2json>

$ chmod u+x xml2json

$ xml2json.py -t xml2json -o data/table.json data/table.html --strip\_text

$ xml2json.py -t json2xml -o data/table1.xml data/table.json

1. json2csv

converts a stream of newline separated json data to csv format.

https://github.com/jehiah/json2csv

$sudo apt-get install golang

$gedit ~/.bashrc

export GOPATH=$HOME/go

export PATH=$PATH:$GOROOT/bin:$GOPATH/bin

$go get github.com/jehiah/json2csv

$ <data/countries.json json2csv -p -k border,surface >data/countries.csv

data/countries.json

{"surface":"0.44","border":"3.2","country":"Vatican City"}

{"surface":"2","border":"4.4","country":"Monaco"}

border,surface

3.2,0.44

4.4,2

<干净的数据-数据清洗入门与实践>

数据科学的六个步骤：（迭代）

问题陈述

数据收集与存储

数据清洗

数据分析

数据可视化

问题决议

认真地记录下曾经按什么顺序做了什么事情？工作日志，包含链接、屏幕截图、或是复制粘贴你曾经运行过的命令，并配上为什么要这样做的解释性文字

文件格式：

文本文件

CSV,TSV等分隔格式 --结构化数据

JSON格式 --半结构化数据

HTML格式 --非结构化数据

二进制文件

分隔文件，用什么字符来封闭分隔数据？

1. 采用双引号来对数据进行封闭
2. 字符转义

# python

## python 工具包

<https://docs.scipy.org/doc/numpy/reference/>

<https://matplotlib.org/gallery/index.html>

<https://docs.scipy.org/doc/scipy/reference/>

<http://scikit-learn.org/stable/>

### numPy http://old.sebug.net/paper/books/scipydoc/numpy\_intro.htmlndarray（N-dimensional array object）



创建

1. 通过Python序列对象创建数组，如果传递的是多层嵌套的序列，将创建多维数组

b = np.array((5, 6, 7, 8), dtype=np.float) # c.dtype 数组元素类型

c = np.array([[1, 2, 3, 4],[4, 5, 6, 7], [7, 8, 9, 10]], dtype=np.complex)

1. 通过functions for creating arrays

#类似python的range，指定开始值，终值和步长，注意不包括终值

np.arange(start=0, stop=1, step=0.1)

np.arange(0, 1, 0.1)[::-1].reshape(5, 2) 逆序+维数改变

np.linspace(start=0, stop=1, num=12) #等差数列，包括终值

np.logspace(start=0, stop=2, num=20, base=10.0) #等比数列

np.zeros((2,2))

np.ones((1,2))

np.full((2,2), 7)

np.eye(N=3, k=0) 正对角阵, k指定对角线，k=1指定正对角线向上平移1

np.random.randint(low=1, high=10, size(3,3)) 有限范围的随机整数

np.random.random((2,2)) [0, 1]范围内的随机数 (b-a)\*random(2,2) + a范围内

np.random.uniform(low=0, high=1, size=(3,3)) 有限范围的均匀分布

np.random.normal(loc=1.0, scale=3.0, size=(3,3)) 指定均值和标准偏差生成正态分布

#重复元素times,返回列表，若value是列表，则将列表里的元素重复times，返回列表

np.repeat(value, times)

Indexing and Slicing

a[5]

a[3:5] #通过下标范围获取数组是原始数组的视图。与原始数组共享同一块数据空间

a[:5] #by default, start is 0, end is the last and step is 1

a[:-1]

a[1:-1:2] 第三个参数表示步长

a[::-1] 数组逆序

np.may\_share\_memory(a, b) # to check if two arrays share the same memory block

aa = a[::2].copy() # force a copy

Fancy Indexing

indexed with boolean or integer arrays (masks), it creates copies not views

a[[3, 3, 1, 8]] 使用**整数序列**作为下标获得的数组不和原始数组共享数据空间

a[a>5] 使用**布尔数组**作为下标获得的数组不和原始数组共享数据空间

多维数组

NumPy采用组元(tuple)作为数组的下标, 也有整数序列或布尔数组作为下标，注意是多维

M[2:9:2, :] 取某行子集

M[2:9, 3:5] 取子矩阵

row\_index = [1,2]

col\_index = [0, 2]

M[row\_index, col\_index] (1,0), (2,2)对应的元素组成

注意：切片只是数据的观测，若要生成新数据，请加copy()

Array Shape Manipulation

c.shape 数组尺寸

c.shape = 4,3 原数组尺寸改变

c.T 转置

d = a.reshape((2, 2)) #原数组尺寸不变，同时注意a and d共享数据存储内存区域

a\_copy = a.reshape((2,2)).copy()

a.ravel() #Flattening, last dimensions ravel out “first”.

z = np.array([1, 2, 3])

z[:, np.newaxis] #结果是array([[1], [2], [3]])

z[np.newaxis, :] #结果是array([[1, 2, 3]]

a = np.arange(4\*3\*2).reshape(4, 3, 2)

b = a.transpose(1, 2, 0) #轴变换(3, 2, 4)

a = np.arange(4)

a.resize((8,)) #扩充数组，补0

数组堆叠

a = np.r\_[3,[0]\*5,-1:1:10j] row concatenation

np.c\_ column concatenation

np.vstack((dataset, sub\_dataset)) == np.stack((row1, row2)) 添加行

np.hstack((row1\_sub, row1\_sub))) 级联

np.stack((dataset, col), axis=-1) 添加列

np.insert(dataset, 3, col, axis=1) 按指定轴插入数据，这里是插入第3列

np.insert(data, 2, row, axis=0) 插入第2行

点运算： Elementwise operations

result = np.sin(a)

result = a1 + a2

+, -, \*, /, //, \*\*, %； abs(), sign(), round(), floor(), sqrt(), log(), exp()

==, >, <, np.logical\_or(), np.logical\_and(a, b)

矩阵运算,排序

np.dot(v, w) # 矢量与矢量内积，矩阵与矢量乘，矩阵与矩阵乘

a.T #矩阵a转置，is a view

b = np.sort(a, axis=1) #逐行排序

j = np.argsort(a) #数组排序，返回排序下标

聚合运算：sum(), prod(), any(), all(), mean(), median(), std(), cumsum(), …

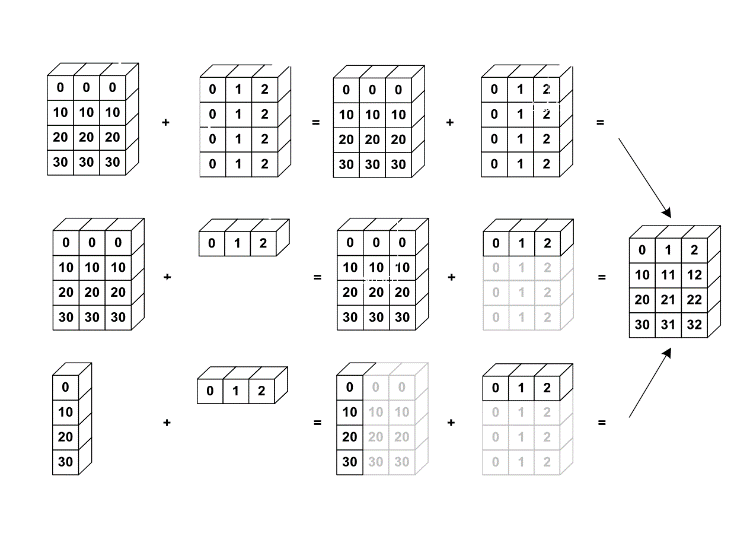
np.sum(result, axis=0) 按指定维运算 (axis=0是列，axis=1是行，axis=2是plane)

x = np.array([1, 3, 2])

x.min(), xmax(),

x.argmin(), x.argmax() #index of minimum and maximum

广播Broadcasting



a = np.arange(0, 60, 10).reshape(-1, 1)

b = np.arange(0, 5)

c = a + b

array([[ 0, 1, 2, 3, 4],

[10, 11, 12, 13, 14],

[20, 21, 22, 23, 24],

[30, 31, 32, 33, 34],

[40, 41, 42, 43, 44],

[50, 51, 52, 53, 54]])

a = np.tile(np.arange(0, 40, 10), (3, 1)) #复制3行

mileposts = np.array([0, 198, 303, 736, 871, 1175, 1475, 1544,1913, 2448])

distance\_array = np.abs(mileposts - mileposts[:, np.newaxis]) #计算两两距离

A lot of grid-based or network-based problems can also use broadcasting.

1. 三维坐标空间

z, y, x = np.mgrid[0:slice, 0:height, 0:width] #directly provides matrices full of indices

distance2center = (z - zc) \*\* 2 + (y - yc) \*\* 2 + (x - xc) \*\* 2 球体坐标

1. 广播机制表达三维坐标空间

xx = np.arange(0, width).reshape(1, 1, width)

yy = np.arange(0, height).reshape(1, height, 1)

zz = np.arange(0, slice).reshape(slice, 1, 1)

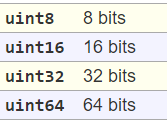
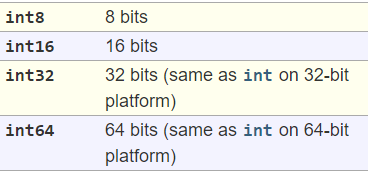
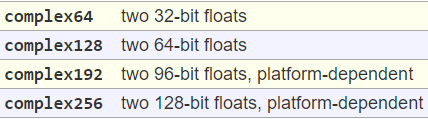
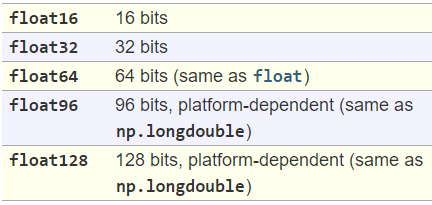
distance2center = (zz - zc) \*\* 2 + (yy - yc) \*\* 2 + (xx - xc) \*\* 2

1. ogrid返回广播，从而计算x, y网格面上各点的值，或者x, y, z网格体上各点的值

z, y, c = np.ogrid[0:slice, 0:height, 0:width] #vectors x, y, z, then broadcasting

distance2center = (z - zc) \*\* 2 + (y - yc) \*\* 2 + (x - xc) \*\* 2 球体坐标

data type and Casting

a = np.array([1, 2, 3]) #dytpe('int64')

a[0] = 1.9 #float is truncated to integer

b = a.astype(float) #forced casts

a = np.around(b) #Rounding

持久化

numpy.load和numpy.save函数以NumPy专用的二进制类型保存数据，这两个函数会自动处理元素类型和shape等信息

np.save("a.npy", a)

c = np.load( "a.npy" )

将多个数组保存到一个文件中的话，可以使用numpy.savez函数

np.savez("result.npz", a, b, sin\_array = c)

r = np.load("result.npz")

Polynomials

1-d polynomials = poly1d class (coefficients or polynomial roots to initialize a po lynomial)

manipulated in algebraic expressions, integrated, differentiated, and evaluated

from numpy import poly1d

p = poly1d([3,4,5]) 3x\*x + 4x + 5

print p.integ(k=6) 积分

print p.deriv() 微分

Vectorizing functions (vectorize)

# 定义标量函数

def addsubtract(a,b):

if a > b:

return a - b

else:

return a + b

# 矢量化标量函数

vec\_addsubtract = np.vectorize(addsubtract)

vec\_addsubtract([0,3,6,9],[1,3,5,7])

### matplotlib

%matplotlib inline for Jupyter notebook at the begining

**import matplotlib.pyplot as plt**

* 曲线绘图

plt.plot([1,2,3,4]) # array of y-axis, the default x = [0, .. len(y)-1]

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)

C, S = np.cos(X), np.sin(X)

plt.plot(X, C, 'r--', X, S, 'bs')

plt.ylabel('some numbers')

plt.show()

You can control the defaults of almost every property in matplotlib: **figure size and dpi, line width, color and style, axes, axis and grid properties, text and font properties** and so on.

plt.figure(figsize=(8, 6), dpi=80) # Create a figure of size 8x6 inches, 80 dots per inch

plt.plot(X, C, color="blue", linewidth=1.0, linestyle="-", label="cosine")

plt.plot(X, S, color="red", linewidth=2.5, linestyle="-", label="sine")

plt.legend(loc='upper left')

plt.xlim(X.min() \* 1.1, X.max() \* 1.1) # Set x limits

plt.ylim(C.min() \* 1.1, C.max() \* 1.1) # Set y limits

plt.xticks(np.linspace(-4, 4, 9, endpoint=True)) # Set x ticks

plt.yticks(np.linspace(-1, 1, 5, endpoint=True)) # Set y ticks

same to matlab, pyplot has concept of current figure and axes

matplotlib.axes.Axes = gca()

matplotlib.figure.Figure = gcf()

plt.clf()

plt.cla()

plt.close() plt.close(1) #指定figure关闭 plt.close(‘all’) #关闭所有图

xlabel(), ylabel(), title() , text(), annotate(), xscale(‘log’), yscale('log')

def f(t):

return np.exp(-t) \* np.cos(2\*np.pi\*t)

t1 = np.arange(0.0, 5.0, 0.1)

t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)

plt.subplot(211)

plt.plot(t1, f(t1), 'bo', t2, f(t2), 'k')

plt.subplot(212)

plt.plot(t2, np.cos(2\*np.pi\*t2), 'r--')

plt.show()

TeX markup in any matplotlib text

* 箱须图

以列为变量，展示四分位和outlier

plt.boxplot(x)

* 散点图

将两变量以点的形式画在一个平面上，可以帮助找出两个变量之间的关系。如果要表示分组和簇，散点图非常有效, c顔色参数，每个标签对应着一个唯一的顔色

plt.scatter(xy[:,0], xy[:,1], c=z, alpha=0.8, edgecolors='none')

* 直方图

plt.hist(x, bins=20)

* 柱状图，比较每个特征的均值和标准差

plt.barh(index, average, xerr=std)

* Pie Charts

Z = np.random.uniform(0, 1, 20)

plt.pie(Z)

* bar plot

n = 12

X = np.arange(n)

Y1 = (1 - X / float(n)) \* np.random.uniform(0.5, 1.0, n)

Y2 = (1 - X / float(n)) \* np.random.uniform(0.5, 1.0, n)

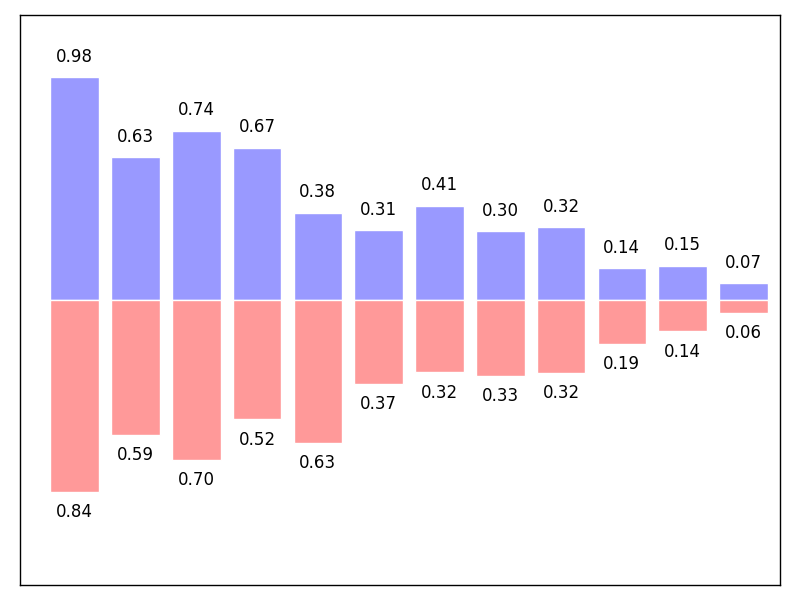
plt.bar(X, +Y1, facecolor='#9999ff', edgecolor='white')

plt.bar(X, -Y2, facecolor='#ff9999', edgecolor='white')

for x, y in zip(X, Y1):

plt.text(x + 0.4, y + 0.05, '%.2f' % y, ha='center', va='bottom')

plt.ylim(-1.25, +1.25)



* Contour plots

def f(x, y):

return (1 - x / 2 + x \*\* 5 + y \*\* 3) \* np.exp(-x \*\* 2 -y \*\* 2)

n = 256

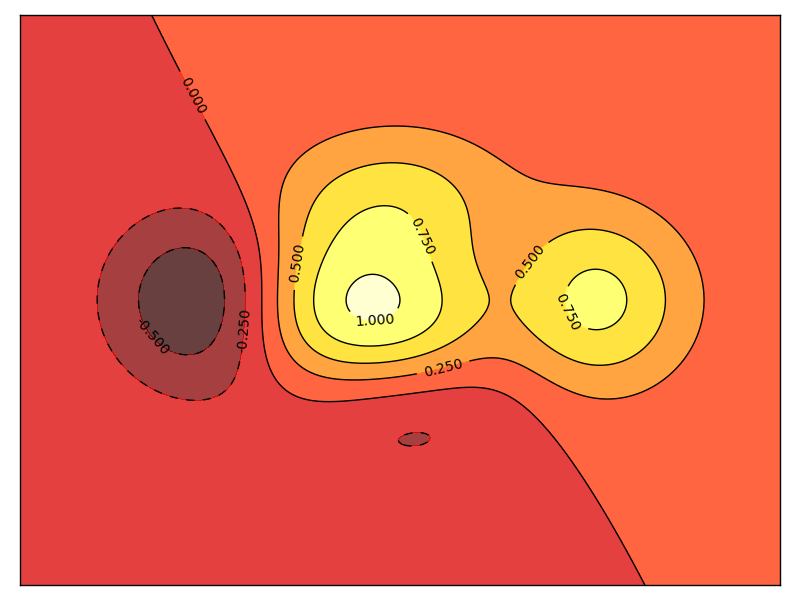
x = np.linspace(-3, 3, n)

y = np.linspace(-3, 3, n)

X, Y = np.meshgrid(x, y)

plt.contourf(X, Y, f(X, Y), 8, alpha=.75, cmap='jet')

C = plt.contour(X, Y, f(X, Y), 8, colors='black', linewidth=.5)



* Quiver plots

n = 8

X, Y = np.mgrid[0:n, 0:n]

plt.quiver(X, Y)

* 热图

x = np.random.normal(loc=0.5, scale=0.2, size=(10,10))

plt.pcolor(x)

plt.colorbar()

* 显示图像

plt.imshow(data, cmap=plt.cm.gray, interpolation='nearest')

* 画水平线x = np.random.normal(loc=0.5, scale=0.2, size=(10,10))plt.colorbar()

plt.axhline(perc\_25, label=’25th perc’, c=’r’)

* 3D Plots

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure()

ax = Axes3D(fig)

X = np.arange(-4, 4, 0.25)

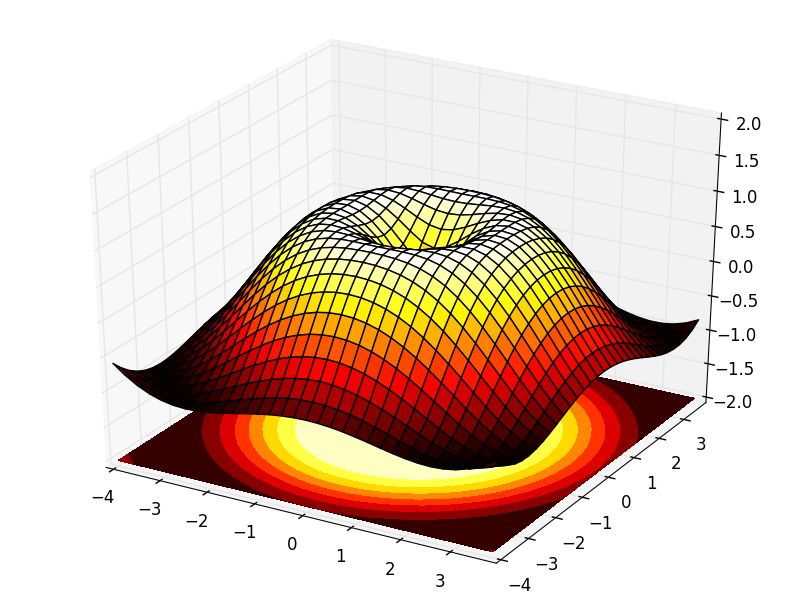
Y = np.arange(-4, 4, 0.25)

X, Y = np.meshgrid(X, Y)

R = np.sqrt(X\*\*2 + Y\*\*2)

Z = np.sin(R)

ax.plot\_surface(X, Y, Z, rstride=1, cstride=1, cmap='hot')



### graphviz

$sudo apt-get update

$sudo apt-get install graphviz

$pip install graphviz

**from** graphviz **import** Digraph

dot = Digraph(comment='The Round Table')

dot.node('A', 'YongQin')

dot.node('B', 'YinBin')

dot.node('C', 'JunBo')

dot.node('D', 'Choo')

dot.edges(['AB', 'AC', 'AD'])

dot.render('result.gv', view=True)

http://simplecv.org/

### pandas

pandas是建立在numpy基础上的，因此很容易从数据框对象提取数据，反之数据也可以自己转换成数据框

**import pandas as pd**

import numpy as np

import matplotlib.pyplot as plt

sd = pd.DataFrame({‘Col1’: range(5), ‘Col2’: ‘qzlin’}) 直接生成DataFrame数据框

dates = pd.date\_range('20130101', periods=6)

df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))

df.head()

df.tail(3)

df.index

df.columns

df.values

df.T

df[‘Col1’] = df[‘Col1’].astype(float) 数据类型转换

selection

df['A'] # select a single column 返回pandas.Series 表示一维向量

df.loc['20130102':'20130104',['A','B']] # both endpoints are included

df.loc[subRow, subCol] 子数据框

df[[‘sepal\_length’, ‘sepal\_width’]] 按列方式获取子数据框

df.at[dates[0],'A'] # 获取单元素

df[0:3] # slices the rows.

df.iloc[3:5,0:2] # By integer slices

df.iat[1,1] # 获取单元素

Boolean Indexing 掩模表明数据行是否被选中

mask\_target = iris[‘target’] == ‘Iris-virginica’ #数据行的掩模

iris.loc[mask\_target, ‘target’] = ‘New label’ 基于行列索引访问元素

df[df.A > 0]

df[df > 0]

df2[df2['E'].isin(['two','four'])]

df2[df2 > 0] = -df2 #只对df2>0的部分进行操作，别的元素不动

Missing Data

pandas primarily uses the value np.nan to represent missing data. It is **by default not included in computations**

df1.dropna(how='any') # To drop any rows that have missing data.

df1.fillna(value=5) # Filling missing data

pd.isnull(df1) # To get the boolean mask where values are nan

df.describe() # 描述性统计

df.mean(axis=0) 行平均值 .median(axis=0) 行中值

df.apply(lambda x: x.max() - x.min()) # 列处理

s = pd.Series(np.random.randint(0, 7, size=10))

s.value\_counts() # 直方图

pd.concat([df[:3], df[3:7], df[7:]])

pd.merge(left, right, on='key') # SQL style merges

df.append(s, ignore\_index=True) # Append rows to a dataframe

iris.sort\_index(by=’sepal\_length’).head() 按特征对观测值排序

df.sort\_index(axis=1, ascending=False) #sorting by an axis

df.sort\_values(by='B') #sorting by values

Grouping：

Splitting the data into groups based on some criteria

Applying a function to each group independently

Combining the results into a data structure

df.groupby('A').sum()

iris.groupby([‘target’]).mean() 以目标种类进行分组操作

iris.groupby([‘target’]).var()

Time Series

Converting between period and timestamp

rng = pd.date\_range('3/6/2012 00:00', periods=5, freq='D')

rng = pd.date\_range('1/1/2012', periods=5, freq='M')

Categoricals

df = pd.DataFrame({"id":[1,2,3,4,5,6], "raw\_grade":['a', 'b', 'b', 'a', 'a', 'e']})

df["grade"] = df["raw\_grade"].astype("category")

df["grade"].unique() 种类

df.to\_csv('foo.csv') # Writing to a csv file

pd.read\_csv('foo.csv') # Reading from a csv file

df.to\_hdf('foo.h5','df') # Writing to a HDF5 Store

pd.read\_hdf('foo.h5','df') # Reading from a HDF5 Store

df.to\_excel('foo.xlsx', sheet\_name='Sheet1') # Writing to an excel file

pd.read\_excel('foo.xlsx', 'Sheet1', index\_col=None, na\_values=['NA']) # Reading from an excel file

### scikit-learn

#Loading an example dataset

from sklearn import datasets

digits = datasets.load\_digits()

#Learning

from sklearn import svm

clf = svm.SVC(gamma=0.001, C=100.)

clf.fit(digits.data[:-1], digits.target[:-1])

#predicting

clf.predict(digits.data[-1:])

#Model persistence

from sklearn.externals import joblib

joblib.dump(clf, 'filename.pkl')

clf = joblib.load('filename.pkl')

#Generalized Linear Models

from sklearn import linear\_model

reg = linear\_model.LinearRegression() #Ordinary Least Squares

reg = linear\_model.Ridge (alpha = .5) #Ridge Regression

scikit-learn 中所有实现机器学习方法的类都来自BaseEstimator,

.fit(x, y)

.predict(x)

.predict\_proba(x) 预测的概率

.predict\_log\_proba(x) 预测概率的对数

### scipy

SciPy contains additional routines needed in scientific work: for example, routines for computing integrals numerically, solving differential equations, optimization, and sparse matrices.

Subpackage Description

io File Input and Output

special Special functions

**linalg** Linear algebra

**interpolate** Interpolation and smoothing splines

optimize Optimization and fit

**stats** Statistical distributions and functions

**integrate** Numerical Integration

**signal** Signal processing

ndimage N-dimensional image processing

**fftpack** Fast Fourier Transform routines

cluster Clustering algorithms

constants Physical and mathematical constants

odr Orthogonal distance regression

sparse Sparse matrices and associated routines

spatial Spatial data structures and algorithms

weave C/C++ integration

**File IO (scipy.io)**

import scipy.io as sio

# Load MATLAB file.

sio.loadmat(file\_name[, mdict, appendmat])

# Save a dictionary of names and arrays into a MATLAB-style .mat file.

sio.savemat(file\_name, mdict[, appendmat, ...])

# List variables inside a MATLAB file.

sio.whosmat(file\_name[, appendmat])

from scipy import misc

misc.imread(‘fname.png’) #导入图像

import matplotlib.pyplot as plt

plt.imread(‘fname.png’) #功能同上，一模一样

**Special functions (scipy.special)**

from scipy import special

airy, elliptic, bessel, gamma, beta, hypergeometric, parabolic cylinder, mathieu, spheroidal wave, struve, and kelvin

def drumhead\_height(n, k, distance, angle, t):

kth\_zero = special.jn\_zeros(n, k)[-1]

return np.cos(t) \* np.cos(n\*angle) \* special.jn(n, distance\*kth\_zero)

theta = np.r\_[0:2\*np.pi:50j]

radius = np.r\_[0:1:50j]

x = np.array([r \* np.cos(theta) for r in radius])

y = np.array([r \* np.sin(theta) for r in radius])

z = np.array([drumhead\_height(1, 1, r, theta, 0.5) for r in radius])

**Linear Algebra (scipy.linalg)**

from scipy import linalg

scipy.linalg vs numpy.linalg

numpy.matrix vs 2D numpy.ndarray

Basic routines: Inverse, Solving linear system, Determinant, norms, Solving linear least-squares problems and pseudo-inverses, Decompositions(Eigenvalues and eigenvectors), SVD(Singular value decomposition)

Singular Value Decomposition (SVD) can be thought of as an extension of the eigenvalue problem to matrices that are not square, Every matrix has a singular value decomposition. Sometimes, the singular values are called the spectrum of A

Sparse Eigenvalue Problems with ARPACK

linalg.det() function computes the determinant of a square matrix

linalg.inv() function computes the inverse of a square matrix

uarr, spec, vharr = linalg.svd(arr)

**Interpolation (scipy.interpolate)**

scipy.interpolate is useful for fitting a function from experimental data and thus evaluating points where no measure exists. The module is based on the FITPACK Fortran subroutines.

# 1-D interpolation (interp1d)

from scipy.interpolate import interp1d

x = np.linspace(0, 10, num=11, endpoint=True)

y = np.cos(-x\*\*2/9.0)

f = interp1d(x, y)

f2 = interp1d(x, y, kind='cubic')

# Multivariate data interpolation (griddata)

from scipy.interpolate import griddata

points = np.random.rand(1000, 2)

values = func(points[:,0], points[:,1])

grid\_x, grid\_y = np.mgrid[0:1:100j, 0:1:200j]

grid\_z0 = griddata(points, values, (grid\_x, grid\_y), method='nearest')

grid\_z1 = griddata(points, values, (grid\_x, grid\_y), method='linear')

grid\_z2 = griddata(points, values, (grid\_x, grid\_y), method='cubic')

# Spline interpolation in 1-d: Procedural (interpolate.splXXX)

Spline interpolation requires two essential steps: (1) a spline representation of the curve is computed, and (2) the spline is evaluated at the desired points. In order to find the spline representation, there are two different ways to represent a curve and obtain (smoothing) spline coefficients: directly and parametrically

from scipy.interpolate import splrep， splev

x = np.arange(0, 2\*np.pi+np.pi/4, 2\*np.pi/8)

y = np.sin(x)

tck = splrep(x, y, s=0)

xnew = np.arange(0, 2\*np.pi, np.pi/50)

ynew = splev(xnew, tck, der=0)

#Two-dimensional spline representation: Procedural (bisplrep)

from scipy.interpolate import bisplrep, bisplev

x, y = np.mgrid[-1:1:20j, -1:1:20j]

z = (x+y) \* np.exp(-6.0\*(x\*x+y\*y))

xnew, ynew = np.mgrid[-1:1:70j, -1:1:70j]

tck = interpolate.bisplrep(x, y, z, s=0)

znew = interpolate.bisplev(xnew[:,0], ynew[0,:], tck)

**Optimization (scipy.optimize)**

from scipy import optimize

Optimization is the problem of finding a numerical solution to a minimization or equality.

* Unconstrained and constrained minimization of multivariate scalar functions (minimize) using a variety of algorithms (e.g. BFGS, Nelder-Mead simplex, Newton Conjugate Gradient, COBYLA or SLSQP)
* Global (brute-force) optimization routines (e.g. basinhopping, differential\_evolution)
* Least-squares minimization (least\_squares) and curve fitting (curve\_fit) algorithms
* Scalar univariate functions minimizers (minimize\_scalar) and root finders (newton)
* Multivariate equation system solvers (root) using a variety of algorithms (e.g. hybrid Powell, Levenberg-Marquardt or large-scale methods such as Newton-Krylov).

from scipy.optimize import minimize **最小化**

1. multivariate scalar functions

# Unconstrained minimization of multivariate scalar functions 无约束

def rosen(x):

"""The Rosenbrock function"""

return sum(100.0\*(x[1:]-x[:-1]\*\*2.0)\*\*2.0 + (1-x[:-1])\*\*2.0)

x0 = np.array([1.3, 0.7, 0.8, 1.9, 1.2])

res = minimize(rosen, x0, method='nelder-mead', options={'xtol': 1e-8, 'disp': True})

# Constrained minimization of multivariate scalar functions (minimize) 有约束

* the Sequential Least SQuares Programming optimization algorithm (SLSQP)
* Least-squares minimization (least\_squares)

1. Univariate scalar functions

# Unconstrained minimization (method='brent')

from scipy.optimize import minimize\_scalar

f = lambda x: (x - 2) \* (x + 1)\*\*2

res = minimize\_scalar(f, method='brent')

# Bounded minimization (method='bounded')

from scipy.special import j1

res = minimize\_scalar(j1, bounds=(4, 7), method='bounded')

from scipy.optimize import root **# Root finding**

def func(x):

return x + 2 \* np.cos(x)

sol = root(func, 0.3)

#### Statistics (scipy.stats)

**from scipy import stats**

continuous random variables and discrete random variables . Over 80 continuous random variables and 10 discrete random variables

The main public methods for continuous RVs are:

rvs: Random Variates

pdf: Probability Density Function

cdf: Cumulative Distribution Function

sf: Survival Function (1-CDF)

ppf: Percent Point Function (Inverse of CDF)

isf: Inverse Survival Function (Inverse of SF)

stats: Return mean, variance, (Fisher’s) skew, or (Fisher’s) kurtosis

moment: non-central moments of the distribution

离散概率分布也称为概率质量函数（probability mass function）。离散概率分布的例子有伯努利分布（Bernoulli distribution）、二项分布（binomial distribution）、泊松分布（Poisson distribution）和几何分布（geometric distribution）等。

连续概率分布也称为概率密度函数（probability density function），它们是具有连续取值（例如一条实线上的值）的函数。正态分布（normal distribution）、指数分布（exponential distribution）和β分布（beta distribution）等都属于连续概率分布。

from scipy import stats

# 二项分布binomial distribution

n次事件中，发生概率为0.5, 恰好发生x次的概率

#计算概率值p(x=650), p(x=651), ...p(x=1200), n=1200, p=0.5

stats.binom.pmf(range(650, 1201), 1200, 0.5)

#生成随机变量x, 服从二项分布n=1200, p=0.5

data = stats.binom.rvs(n=1200, p=0.5, size=1000)

plt.hist(data, bins=24, normed=True) #可视化随机变量pmf

# 泊松分布 Poisson distribution

某路口事件发生的比率是ramda(每天发生次数), 求一段固定时间间隔内(这里是1天）发生x次事件的概率

stats.poisson.pmf(range(0, 10), 2)

#生成随机变量x, 服从泊松分布ramda=2

data = stats.poisson.rvs(mu=2, size=1000)

# 正态分布

stats.norm.pdf(np.arange(-5, 5, 0.1), loc=0, scale=1)

data = stats.norm.rvs(loc=0, scale=1, size=1000)

描述性统计

perc\_25 = np.percentile(y, 25) #数据的百分位，第25位

np.percentile(y, 50)

np.mean(col)

np.var(col)

np.std(col)

np.median(col)

stats.trim\_mean(col, 0.1) #将数据中最大和最小的10%截去，然后计算剩下的平均值

Statistical tests

#T-test to decide whether the means of two sets of observations are significantly different

a = np.random.normal(0, 1, size=100)

b = np.random.normal(1, 1, size=10)

(array(-3.17..), 0.0019...) = stats.ttest\_ind(a, b)

The T statistic value: it is a number the sign of which is proportional to the difference between the two random processes and the magnitude is related to the significance of this difference.

the p value: the probability of both processes being identical. If it is close to 1, the two process are almost certainly identical. The closer it is to zero, the more likely it is that the processes have different means

**Integration (scipy.integrate)**

help(integrate)

from scipy.integrate import quad #单变量积分

from scipy.integrate import dblquad #双变量积分

from scipy.integrate import tplquad #三变量积分

from scipy.integrate import nquad #多变量积分

N = 5

def f(t, x):

return np.exp(-x\*t) / t\*\*N

nquad(f, [[1, np.inf],[0, np.inf]])

from scipy.integrate import simps Integrating using Samples

from scipy.integrate import odeint # Ordinary differential equations 常微分方程

**Signal Processing (scipy.signal)**

Filtering: Convolution/Correlation

Time-discrete filters: FIR(finite response filters) and IIR(infinite response filters)

Other filters: Median Filter, Order Filter, Wiener filter, Hilbert filter

Spectral Analysis, spectral density

Analog Filter Design

#### image processing (scipy.ndimage)

from scipy.misc import imread, imsave, imresize

import matplotlib.pyplot as plt

img = imread('assets/cat.jpg') # Read an JPEG image into a numpy array

img\_tinted = imresize(img, (300, 300)) # Resize the image to be 300 by 300 pixels.

imsave('assets/cat\_tinted.jpg', img\_tinted) # Write the tinted image back to disk

plt.imshow(img) # Show the original image

from scipy import ndimage 具体见computervision.docx

ndimage.shift(..)

ndimage.rotate(..)

ndimage.zoom(..)

ndimage.gaussian\_filter(..)

ndimage.median\_filter(..)

ndimage.binary\_erosion(..)

Correlation and convolution

Fourier domain filters

Distance transforms

Interpolation functions

Morphology: Binary morphology, Grey-scale morphology

Segmentation and labeling

Object measurements

**Fourier analysis**

#discrete Fourier transform (DFT)， Fast Fourier Transform (FFT)

from scipy.fftpack import fft, ifft

x = np.array([1.0, 2.0, 1.0, -1.0, 1.5])

y = fft(x)

yinv = ifft(y)

#Two and n-dimensional discrete Fourier transforms

from scipy.fftpack import fft2, ifft2, fftn, ifftn

#Discrete Cosine Transforms

from scipy.fftpack import dct, idct

#Discrete Sine Transforms

from scipy.fftpack import dst, idst

**Spatial data structures and algorithms (scipy.spatial)**

Delaunay triangulations

The Delaunay triangulation is a subdivision of a set of points into a non-overlapping set of triangles,

Convex hulls

Convex hull is the smallest convex object containing all points in a given point set.

Voronoi diagrams

A Voronoi diagram is a subdivision of the space into the nearest neighborhoods of a given set of points.

**Weave (scipy.weave)**

The scipy.weave (below just weave) package provides tools for including C/C++ code within in Python code

### SymPy

Symbolic computation deals with the computation of mathematical objects symbolically

Symbolic computation systems (which by the way, are also often called computer algebra systems, or just CASs) such as SymPy are capable of computing symbolic expressions with variables.

SymPy can simplify expressions, compute derivatives, integrals, and limits, solve equations, work with matrices, it includes modules for plotting, printing (like 2D pretty printed output of math formulas, or LATEXLATEX), code generation, physics, statistics, combinatorics, number theory, geometry, logic, and more

Whenever you combine a SymPy object and a SymPy object, or a SymPy object and a Python object, you get a SymPy object, but whenever you combine two Python objects, SymPy never comes into play, and so you get a Python object.

import math

math.sqrt(8)

import sympy

sympy.sqrt(8)

易犯错误点：

x = symbols('x')

expr = x + 1

expr.subs(x, 2) 符号表达式求值

expr.subs(x, x\*y) 符号替换

expr = x\*\*3 + t\*x\*y - z

expr.subs([(x, 2), (y, 4), (z, 0)])

expr = x\*\*4 - 4\*x\*\*3 + 4\*x\*\*2 - 2\*x + 3

replacements = [(x\*\*i, y\*\*i) for i in range(5) if i%2 == 0]

expr.subs(replacements) 结果y\*\*4 - 4\*x\*\*3 + 4\*y\*\*2 - 2\*x + 3

注意：SymPy expressions are immutable, no function will change them in-place. All functions will return new expressions.

条件测试

a = (x+1)\*\*2

b = x\*\*2 + 2\*x + 1

if simplify(a - b) == 0:

if a.equals(b):

字符串转符号表达式converting strings to sympy expressions

str\_expr = "x\*\*2 + 3\*x - 1/2"

expr = sympify(str\_expr)

convert a SymPy expression to an expression

expr = sin(x)

f = lambdify(x, expr, "numpy")

f(numpy.arange(10))

Simplification

uses heuristics to determine the simplest result

>>> simplify(sin(x)\*\*2 + cos(x)\*\*2)

>>> simplify((x\*\*3 + x\*\*2 - x - 1)/(x\*\*2 + 2\*x + 1))

Polynomial/Rational Function Simplification

x, y =symbols('x y z')

>>> expand((x + 1)\*\*2)

>>> factor(x\*\*2\*z + 4\*x\*y\*z + 4\*y\*\*2\*z)

expr.coeff(x, n) gives the coefficient of x\*\*n in expr:

>>> expr = x\*y + x - 3 + 2\*x\*\*2 - z\*x\*\*2 + x\*\*3

>>> collected\_expr = collect(expr, x)

>>> collected\_expr.coeff(x, 2)

>>> trigsimp(sin(x)\*tan(x)/sec(x)) #simplify trigonometric

>>> expand\_trig(sin(x + y)) #expand trigonometric functions

By default, SymPy Symbols are assumed to be complex

Symbols can be given different assumptions by passing the assumption to symbols()

>>> x, y = symbols('x y', positive=True)

>>> a, b = symbols('a b', real=True)

Special Functions

>>> factorial(n)

>>> binomial(n, k) n choose k

>>> gamma(z)

Calculus

>>> diff(x\*\*4, x, 3) #the third derivative 微分

>>> diff(exp(x\*y\*z), x, y, 2, z, 4) #多阶偏微分

integrate(exp(x)\*sin(x) + exp(x)\*cos(x), x) 积分

integrate(sin(x\*\*2), (x, -oo, oo)) 定积分

>>> integrate(exp(-x\*\*2 - y\*\*2), (x, -oo, oo), (y, -oo, oo)) #多变量积分

limit(sin(x)/x, x, 0) 极限

>>> limit(1/x, x, 0, '+') 单边极限

>>> expr = exp(sin(x))

>>> expr.series(x, 0, 4) 泰勒级数展开

Solvers

Eq(x+1, 4) x+1 = 4 方程等式

>>> solveset(Eq(x\*\*2, 1), x) #solving algebraic equations

注：solve(x\*\*2 - 2, x)求根不被推荐

>>> f = symbols('f', cls=Function)

>>> diffeq = Eq(f(x).diff(x, x) - 2\*f(x).diff(x) + f(x), sin(x))

>>> dsolve(diffeq, f(x)) 解微分方程

Matrix([[1, 2], [2, 2]]).eigenvals()

besselj(nu, z).rewrite(jn)

latex(Integral(cos(x)\*\*2, (x, 0, pi)))

## 《数据科学导论》

## 《Python数据科学指南》

## 《数据科学入门》

数据科学 = 线性代数，统计建模，可视化，计算语言学，图形分析，机器学习，商业智能，数据存储和检索等众多学科

PyPI: Python软件的公共仓库

http://www.scipy.org/install.html

### install & upgrade

Upgrade pip

$ python -m pip install --upgrade pip

$ pip install --user **numpy scipy matplotlib pandas sympy** **scikit-learn**

安装环境

<https://www.digitalocean.com/community/tutorials/how-to-set-up-a-jupyter-notebook-to-run-ipython-on-ubuntu-16-04>

$ sudo apt-get -y install ipython ipython-notebook #install ipython

**$ sudo -H pip install jupyter # install Jupyter Notebook**

别的工具包

$ pip install --user statsmodels beautifulsoup4 networkx nltk gensim

注意：若>>>import matplotlib.pyplot as plt occus no module named “python-tk”，则需安装

$sudo apt-get install python-tk

Upgrade （安装即升级）

>>>import numpy as np

>>>numpy.\_\_version\_\_ 查看安装包的版本

$ pip install - U numpy==1.9.1 安装指定版本

$ pip install – U numpy 安装最新版本

IPython使用

$ipython

Obj? 查看对象

!pwd 运行bash命令，在命令前加一个！

$ipython notebook web浏览器方式

$ jupyter notebook #run jupyter notebook

### 数据集

Scikit-learn工具包自带数据集

from sklearn import datasets

iris = datasets.load\_iris()

数据集对象：

iris.DESCR 数据集总体描述

.data 特征空间 返回numpy.ndarray

.target 目标值 返回numpy.ndarray

.feature\_names 特征名称

.target\_names 目标名称

随机split训练集和测试集

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, train\_size=0.8)

mldata.org机器学习数据集的公共资源库

from sklearn.datasets import fetch\_mldata

earthquakes = fetch\_mldata(‘global-earthquakes’)

LIBSVM Data样本

ip\_addr = <http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

target\_page = ip\_addr + ‘binary/ala’

import urllib2

a2a = urllib2.urlopen(target\_page)

from sklearn.datasets import load\_svmlight\_file

X\_train, y\_train = load\_svmlight\_file(a2a)

直接从csv, txt, json, sql, hdfs, html, State文件加载

pandas库提供各种函数来加载以上各种文件

import numpy as np

housing = np.loadtxt(‘\*.csv’, delimiter=’,’)

loadtxt函数默认制表符为文件中数值之间的分隔符，如果分隔符是, or ; 则必须使用参数定义符进行说明; 参数skip指出从文件中的第几行开始读取数据。要求数组所有元素具有相同的类型。

import pandas as pd

iris = pd.read\_csv(‘\*.csv‘, parse\_dates=[0], error\_bad\_lines=False…)

返回pandas.DataFrame, 按照矩阵形式处理数据集，可以由不同类型的变量组成

分隔符分割变量，推断每一列的正确数据类型，转换数据，解析日期，缺失值和出错数据

parse\_dates解析日期

error\_bad\_lines,当加载的数据集有错误或坏行，忽略坏行（默认行为是停止并抛出异常）

Scikit-learn样本生成器

from sklearn import datasets

x, y = datasets.make\_classification(n\_samples=10\*\*6, n\_features=10, random\_state=101)

X, y = datasets.make\_circles()

X, y = datasets.make\_moons()

…

详细见sklearn/datasets/samples\_generator.py

处理大数据集

区块划分和加载

iris\_chunks = pd.read\_csv(filename, header=None, chunksize=10) 每次导入10行

for chunk in iris\_chunks:

print chunk.shape

以数据流方式处理

import csv

def batch\_read(filename, batch=5):

with open(filename, 'rb') as data\_stream:

batch\_output = []

for n, row in enumerate(csv.reader(data\_stream), dialect='excel')):

if n>0 and n%batch==0:

yield(np.array(batch\_output))

batch\_output = ()

batch\_output.append(row)

yield(np.array(batch\_output))

for batch\_input in batch\_read(iris\_filename, batch=3):

print batch\_input

### 数据预处理

**异质数据(pandas.DataFrame) ， 数据表格(numpy.ndarray)**

数据改写

data munge, data wrangling, data preparation

观测数据，然后在假设的设计和测试中不断循环。例如，以图形化的方式考察变量。在**描述性统计**资料的帮助下，将领域知识运用于实践，你会找到**创建新变量**的方法。**处理冗余信息和意外信息（首先是异常值）**，然后选择最有意义的变量和最有效的参数，选择一个机器学习算法进行测试

处理缺失值

若缺失比例不大，直接忽略该case

若连续变量，平均值，中位数或者最常见的数值都可以被用来替代丢失的数据。复杂点是拟合，结合类别标签进行

from sklearn.preprocessing import Imputer

imputer = Imputer(missing\_values=0, strategy=’mean’)

x\_imputer = imputer.fit\_transform(x\_t) cases平均值替代缺失数据

等价于np.mean(x\_t, axis=0) 替代某个case

Strategy可以为平均值，中位数，最常用的值

EDA (Exploratory Data Analysis) 探索性数据分析

了解数据集、检查数据集的特征和形状，验证脑海中已有的一些假设，对数据科学任务接下来的步骤有一些初步的想法

数据框对象(DataFrame)探索函数

.describe() 统计描述

.boxplot() 各个特征的box图（均值，标准差，最大/小值，分位数）

.quantile([0.1, 0.9]) 分位数

.target.unique() 目标种类

**交叉表和散列图，度量两个特征的关联性**

交叉表：两列特征均是种类，一列置于数据表左侧，一列置于数据表右侧，交叉处值表示满足行列条件的记录的汇总

pd.crosstab(iris['len']>iris['len'].mean(), iris['width']>iris['width'].mean())

len>mean(len) False True <- width>mean(width)

False 56 1

True 4 89

从以上交叉表可以看出，这两种特征（width and len）与均值的比较几乎总是同时发生

plt.scatter(iris['len'], iris['width'])

散列图，若呈现主对角(或副对角)趋势，则正相关(或负相关)

协方差矩阵：度量所有不同特征对之间相关性，若m个特征，则产生协方差矩阵m\*m，　从协方差矩阵，可以视查出强相关特征（也是可以丢弃的特征）和独立特征的数量。

cov\_data = np.corrcoef(iris.data.T)

直方图：如何选择bin数？

根据经验法则，分箱数量的初始值是观测数量的平方根，然后，需要不断修正，直到得到一个很好的概率分布形状

plt.hist(iris['width'], bins=20)

Freedman-Diaconis规则：直方图中最佳的分箱数量取决于分箱的宽度，分箱宽度可采用四分位间距IQR和观测数量进行计算

h = 2 \* IQR \* n(-1/3) 分箱宽度h

bins = (max – min) / h

数据框提供很多可视化函数进行EDA

.boxplot() 每列箱线图

iris\_df['groups'] = pd.Series([iris.target\_names[k] for k in list(iris.target)])

.boxplot(column='sepal length (cm)', by='groups') 分组箱线图 某列按groups

.plot(kind='density') 变量密度图 （每列即一个变量）

iris\_df['petal width (cm)'].plot(kind='hist', alpha=0.5) 变量直方图

散点图

colors\_palette = {0: 'red', 1: 'yellow', 2: 'blue'}

colors = [colors\_palette[c] for c in groups]

simple\_scatterplot = iris\_df.plot(kind='scatter', x=0, y=1, c=colors)

散点图矩阵

from pandas.tools.plotting import scatter\_matrix

scatter\_matrix(iris\_df, alpha=0.2, figsize=(6, 6), c=colors, diagonal='kde')

平行坐标图

from pandas.tools.plotting import parallel\_coordinates

pll = parallel\_coordinates(iris\_df, 'groups')

有用函数

np.percentile(y, 25) 第25百分位数

from scipy.stats import trim\_mean

trim\_mean(y, 0.1) 10%截尾均值，将数据最大的10%和最小的10%排队，计算剩下的平均值

输入特征进行标准化

**Z-scores标准化**：z=(x-mean)/std 均值为0,标准差为1

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train) 均值为0,标准差为1

from sklearn.preprocessing import scale

x\_s = scale(X\_train, with\_mean=True, with\_std=True, axis=0)

**归一化**: y = (x-min)/(max-min)

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit\_transform(X\_train)

线性映射：y = (x-min)/(max-min) \* (new\_max-new\_min) + new\_min

scaler.fit\_transform(X\_train) \* (new\_max-new\_min) + new\_min

哑元，虚拟编码

离散数据集编码为二进制特征，若某分类变量level=5, 则引入5个哑元变量，每个哑元变量取值0/1，每个二进制特征对应分类数据的一个等级

categorical\_feature = pd.Series(['sunny', 'cloudy', 'snowy', 'rainy', 'foggy'])

dummies = pd.get\_dummies(categorical\_feature)

对整个数据集进行虚拟编码

比如10个变量表示5张牌中的13个点数和4种花色，目标变量是0-9有序值表示不同的牌组合，使用OneHotEncoder可以将10个变量转换成(13+4)\*5=85个二值变量，有10类需要预测，需要训练10个分类器，每个分类器针对单个类相对于其他类进行预测

from sklearn.preprocessing import OneHotEncoder

hot\_encoding = OneHotEncoder(sparse=True)

X\_train = hot\_encoding.fit\_transform(X\_train.toarray())

### 特征创建

若发现特征和目标变量并不是很相关，在这种情况下，可以修改输入的数据集，应用线性或非线性变换来提高系统的精度，或者采用其他相似的方法，数据科学家负责改变数据集和输入数据，使数据更好地符合分类模型

比如房间长宽高参数与估价没有明显的关系，但长宽高经非线性变换产生新特征体积，和估计或许有明显的关系

#产生多项式特征集

#如输入２维即[a,b] ,则二阶多项式特征集[1,a,b,a^2,ab,b^2]。

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)

x\_poly = poly.fit\_transform(x)

### 特征降维

1. 缺失值比率(missing values ratio)

若数据列包含太多缺失值，则包含有用信息的可能性较少。可以将数据列缺失值大于某个阈值的列去掉

1. **低方差滤波(low variance filter)**

若数据列变化非常小，则包含的信息少。所以数据列方差小的列被移除。注意方差与数据范围相关，因此前提：数据归一化处理

1. 高相关滤波(high correlation filter)

**若两数据列高度相关（即变化趋势相似），则包含信息也相同，那么使用一列即可**。注意类别相关系数是通过计算皮尔逊方值来表示。相关系数对范围敏感，所以前提：数据归一化处理

1. PCA

新的特征是现有特征的线性组合（即旋转），输入空间经过旋转后，输出集合的第一个向量包含信号的大部分能量（即方差）。第二个向量与第一个向量正交，它包含剩余的大部分，第三个向量又与前两人个向量正交，并包含剩余能量的大部分，以此类推

主要算法：奇异值分解Singular value Decompositions

通过情况下，输出集合的基数是保证输入能量的95%（有时候是99%)

注：**PCA变换对正交向量的尺度敏感，解决方案：对信号进行白化。在这种情况下，各维度特征向量的方差强制为１。白化会去除部分信息，但是，有时候它会提高机器学习算法的精度**

PCA变换会丧失数据的解释性，若数据的解释能力对分析很重要，那么就不适合用PCA

#信号白化，然后PCA,保留能量（方差）>９５%,　从而知道维数和具体方差

pca\_95pc = PCA(n\_components=0.95, whiten=True)

X\_pca\_95pc = pca\_95pc.fit\_transform(iris.data)

print pca\_95pc.explained\_variance\_ratio\_.sum()

print X\_pca\_95pc.shape

若降至两维，画散列图，看类别是否可分

plt.scatter(X\_pca\_1cw[:, 0], X\_pca\_1cw[:, 1], c=iris.target, alpha=0.8, edgecolors='none')；plt.show()

PCA奇异值分解算法进行满秩计算，计算量很大，可以采用随机SVD(Randomized SVD)进行近似的迭代计算，速度更快，常用于大数据的PCA

from sklearn.decomposition import RandomizedPCA

rpca\_2c = RandomizedPCA(n\_components=2)

X\_rpca\_2c = rpca\_2c.fit\_transform(iris.data)

附加知识：

**通过正交变换将一组可能存在相关性的变量转换为一组线性不相关的变量,前提是尽可能多地反映原来变量的信息**

转换后的这组变量叫主成分

从原始变量中导出少数几个主成分，使它们尽可能多地保留原始变量的信息

应用场合：特征降维（比如人脸识别，200\*200图像，将灰度值作为特征，即40000维，邻近像素相关性高，可以用PCA降维，保存识别所需要的信息）

数据变化的主方向u1就是协方差矩阵的主特征向量，而u2是次特征向量。

白化的目的就是降低输入的冗余性,即如下：

1. 特征之间相关性较低；(ii)所有特征具有相同的方差。

白化是一种数据预处理方法。 事实证明这也是一种生物眼睛(视网膜)处理图像的粗糙模型。具体而言，当你的眼睛感知图像时，由于一幅图像中相邻的部分在亮度上十分相关，大多数临近的“像素”在眼中被感知为相近的值。因此，如果人眼需要分别传输每个像素值（通过视觉神经）到大脑中，会非常不划算。取而代之的是，视网膜进行一个与ZCA中相似的去相关操作 (这是由视网膜上的ON-型和OFF-型光感受器细胞将光信号转变为神经信号完成的)。由此得到对输入图像的更低冗余的表示，并将它传输到大脑。

算法：m个case, n个特征

x\_s = scale(x, with\_mean=True, with\_std=True, axis=0) m\*n

x\_c = np.corrcoef(x\_s.T) n\*n

eig\_val, eig\_vec = scipy.linalg.eig(x\_c) n\*n

w = eig\_vec[, :2] 选择前两个特征向量 n\*2

x\_pca = x\_s.dot(w) 降维 m\*2

1. Factor Analysis （因子分析）

**从特征变量群中提取共性因子**的统计技术。如一个学生的英语，语文，数学成绩都很好，那么潜在的共性因子可能是智力水平高。

要求：变量之间具有强相关性（这样才能找到共性特征）

主要算法：PCA,　主轴因子法，极大似然法，最小二乘法，a因子提取法等

from sklearn.decomposition import FactorAnalysis

fact\_2c = FactorAnalysis(n\_components=2)

X\_factor = fact\_2c.fit\_transform(iris.data)

1. LDA(Linear Discriminant Analysis)

线性判别分析是一种分类器，也经常用于降维

条件：有监督，只能用于分类，不能用于回归分析

基本思想：将高维特征空间投影到最佳子空间，使类间距最大，类内距最小

对于两类，即找到最佳投影轴，使两类样本在该轴投影之间的距离尽可能远，而每一类样本的投影尽可能紧。

from sklearn.lda import LDA

lda\_2c = LDA(n\_components=2)

X\_lda\_2c = lda\_2c.fit\_transform(iris.data, iris.target)

受限玻耳兹曼机(Restricted Boltzmann Machine RBM)

常用于降维（隐层少一点），学习特征（隐层输出就是特征）

from sklearn.neural\_network import BernoulliRBM

rbm = BernoulliRBM(n\_components=64, learning\_rate=0.01, n\_iter=100)

rbm.fit(X)

rbm.components\_

7.核PCA

**PCA特点**：输入的数据集必须要有相关联的列，若输入的变量间没有相关性，PCA无效。**只适合线性分布的数据**

**核PCA:非线性降维，输入数据→核空间→PCA**

核函数k(x,y)：相似度函数, 计算x, y之间的相似度

高维空间点积可以由低维空间点积获得

输入向量x是n维空间，核映射成phi(x), m维空间, m>>n

如：x=[x1, x2], 核映射phi(x) = [x12, x1x2, x2x1, x22]

y=[y1, y2], 核映射phi(y) = [y12, y1y2, y2y1, y22]

高维空间的点积<phi(x), phi(y)> = x12 y12 + 2\*x1x2 y1y2 + x22 y22

<x,y>2 = (x1 y1 + x2 y2)2 = x12 y12 + 2\*x1x2 y1y2 + x22 y22

所以<phi(x), phi(y)> = <x,y>2

RBF高斯核：K(X1, x2) = exp(gamma\*||x1-x2||2) 值(0, 1)

gamma = -1/(2\*sig2)

from sklearn.datasets import make\_circles

from sklearn.decomposition import KernelPCA

x, y = make\_circles(n\_samples=400, factor=.2, noise=.02)

kpca = KernelPCA(kernel=’rbf’, gamma=10, n\_components=1)

x\_kpca = kpca.fit\_transform(x)

8. SVD (Singular Value Decomposition)奇异值分解

SVD不需要一个协方差或者相关矩阵，它直接作用于原始数据矩阵

A = U\*S\*VT  m\*n = (m\*k) \* (k\*k) \* (n\*k)T

左奇异向量\*奇异值对角阵\*右奇异向量

from scipy.linalg import svd

x\_s = scale(x, with\_mean=True, with\_std=True, axis=0) m\*n

U, S, V = svd(x\_s, full\_matrices=False)

**x\_t = U[:, :2] 选择前两个奇异值近似原始的矩阵 m\*2**

SVD经常被应用在文本挖掘应用里，用来发现词和文档，文档与文档以及词与词之间隐藏的关联(语义关联)

文本语料的近似表示方法（潜在语义分析）

词－文档矩阵 －》　SVD　－》　新的语义空间，同义词，多义词距离非常接近

9. 随机映射

据Johnson-Linden Stauss定理：**存在高维至低维的映射(欧几里得空间),点到点的距离保持在一个sigma方差内。 即高维空间点对距离是d, 低维空间点对距离在(1-sigma, 1+sigma)\*d范围内**

from sklearn.random\_projection import GaussianRandomProjection

from sklearn.metrics import euclidean\_distances

gauss\_proj = GaussianRandomProjection(n\_components=100)

x\_t = gauss\_proj.fit\_transform(x) m\*n → m\*100

abs(euclidean\_distances(x), euclidean\_distances(x\_t))

10 用NMF(Non-negative Matrix Factorization)分解特征矩阵

协同过滤算法的非负矩阵分解

A = Adash \* H m\*n = (m\*d) \* (d\*m)

优化问题：

min|A - Adash \* H|2

from sklearn.decomposition import NMF

nmf = NMF(n\_components=2)

A\_dash = nmf.fit\_transform(A) m\*n → m\*2

### 特征选择

单特征选择

#选择最优25%特征

Xbin = Binarizer().fit\_transform(scale(X))

Selector\_chi2 = SelectPercentile(chi2, percentile=25).fit(Xbin, y)

chi\_scores = Selector\_chi2.get\_support()

print sum(chi\_scores)

Selector\_f\_classif = SelectPercentile(f\_classif, percentile=25).fit(X, y)

f\_classif\_scores = Selector\_f\_classif.get\_support()

print sum(f\_classif\_scores)

最优特征子集　（见page111)

from sklearn.feature\_selection import RFECV

selector = RFECV(estimator=classifier, step=1, cv=10, scoring='accuracy')

selector.fit(X\_train, y\_train)

print 'Optimal number of features: %d' % selector.n\_features\_

X\_train\_selected = selector.transform(X\_train)

**若变量数>训练集，除了降维，特征选择，有些算法使用正则化限制系数的权重，从而防止过拟合，并在不失去预测能力的情况下选择最相关的变量**。特别是L1正则化lasso,它根据集合的正则化强度把许多变量的系数都设置为0.

### 异常检测

单变量异常检测

异常点：数据源中远离其他点的那些点，**评估数据集的中心及其分布将有助于检测异常点**。可以采用平均值和中位值作为数据集的大致中心，采用标准差和绝对中位差作为数据集的分布

1. 超出３倍绝对中位差

中位值作为评估值比较健壮 ，它是在升序排序的多个观察值中位于中间的观察值，要想彻底改变中位值，我们要替换掉远离中位值的一半观察值

mad = 1.4826\*median(|x – median(x)|) 绝对中位差

(lower\_limit, upper\_limit) = (median-3\*mad, median+3\*mad)

1. 超出3倍绝对标准差

平均值和标准差容易受异常点干扰，评估会发生巨大偏差

(lower\_limit, upper\_limit) = (mean-3\*std, mean+3\*std)

理论依据：对于高斯分布，68.27%的数据落在一个标准差范围内，95.45%的数据落在两个标准差范围内，**99.73%的数据落在三个标准差范围内**。

boxplot(), 最小值，下四分位，中位数，上四分位，最大值，　其余的为outlier

多变量异常检测

1. EllipticEnvelope算法

**检查每个观测量与总均值的距离**

# Robust Covariance Estimate

from sklearn.covariance import EllipticEnvelope

robust\_covariance\_est = EllipticEnvelope(store\_precision=False, assume\_centered=False, contamination=.05)

robust\_covariance\_est.fit(normalized\_data)

detection = robust\_covariance\_est.predict(normalized\_data)

outliers = np.where(detection==-1)

regular = np.where(detection==1)

1. OneClassSVM

一类SVM(异常检测器）

指training data只有一类positive or negative的data,而没有另外一类

若数据集二分类，那么经典svm寻找两类的最大间隔

若只有一类，那么**OneClassSVM通过学习该类的特征或服从某类分布。若新的样本不符合以前的数据分布，那么标记为新奇样本**。

若两类极度不平衡，若用经典SVM会造成分类过于偏向数目多的样本类别，使train出来的model有bias,可以考虑用OneClassSVM

from sklearn import svm

outliers\_fraction = 0.02

nu\_estimate = 0.95 \* outliers\_fraction + 0.05

machine\_learning = svm.OneClassSVM(kernel='rbf', gamma=1.0/len(normalized\_data), degree=3, nu=nu\_estimate)

machine\_learning.fit(normalized\_data)

detection = machine\_learning.predict(normalized\_data)

outliers = np.where(detection==-1)

regular = np.where(detection==1)

1. LOF(Local Outlier Factor)局部异常因子

### 验证

case与case之间的度量：距离，相似度

任何数据集都是一系列的点，属于某个特定空间。如欧氏空间，点是实数向量，向量的长度是它的维度数

距离：两点间最短路径的长度

对于不同数据类型组成的空间，选择合适的距离度量

from sklearn.metrics import pairwise

pairwise.paired\_distance(X, Y, metric=’euclidean’)

computes the distances between (X[0], Y[0]) …

Lr-Norm: 注意：欧氏空间

d(x, y) = ( sum(x-y)r )1/r

当r=2, 欧氏距离

当r=1, manhattan距离

cosine: 两个向量之间的夹角的余弦 注意：欧氏空间，整数或布尔值

np.dot(x, y) / np.sqrt( np.dot(x,x) \* np.dot(y,y) )

Jaccard距离：1 – 两个向量的交集/并集 注意：集合

Hamming距离：两个向量间不同的位的数量 注意：位类型

相似度： 1 – 距离

随机分解数据集

from sklearn.cross\_validation import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=.3, random\_state=1)

data = np.column\_stack([X, y]) 列合并

随机分解类别分布一致的训练集和测试集

from sklearn.cross\_validation import StratifiedShuffleSplit

stratified\_split = StratifiedShuffleSplit(y, test\_size=0.2, n\_iter=1)

for train\_idx, test\_idx in stratified\_split:

train = data[train\_idx]

test = data[test\_idx]

随机采样

所采的样本应该能够很好地代表整体数据集，具有和原数据基本一致的特征。如平均值特征，样本的平均值必须和原数据的平均值尽量接近

x\_sample\_idx = np.random.choice(range(x.shape[0]), 10, replace=True)

x\_t[x\_sample\_idx, :] 随机采样10个case

replace参数控制采样是有放回还是无放回

混淆矩阵：confusion matrix

每类被错分为其他类情况的表。理想情况下，一个完美的分类，其混淆矩阵所有非对角线上的元素都应该为0

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

准确率Accuracy = 正确分类的样本数/总样本数

若类极不平衡，如google抓取感兴趣页面100，共索引1000,0000万页面，若把所有页面归为非感兴趣页面，则accuracy = 99.999%

假定感兴趣的为正类，不感兴趣的为负类

若正类判定为正类，称为true positive；若正类判定为负类，称false negative 即去真

若负类判定为负类，称为true negative；若负类判定为正类，称false positive即存伪

精确度Precision = 正确被检索的item(TP) / 所有被检索到的item(TP+FP)

召回率Recall = 正确被检索的item(TP) / 应该被检索到的item(TP+FN)

F1 Score = 精确度和召回率的调和平均 = 2 / (1/P + 1/R) = 2\*PR/(P+R)

from sklearn.metrics import classification\_report

print classification\_report(y\_test, y\_pred, target\_names=iris.target\_names)

交叉验证

交叉验证只是用于优化目的，而不是为了性能评估，交叉验证只是从平均最佳的角度，指出模型最好的算法和参数选择。为了得到模型性能的无偏估计，更应该使用测试集

k fold　=> mean(模型总体性能的公正估计) and std(模型的方差)

from sklearn import svm

model = svm.SVC(kernel='rbf', gamma=0.001, C=1.0)

scores = cross\_validation.cross\_val\_score(model, X, y, cv=10, scoring='accuracy', n\_jobs=-1)

np.mean(scores), np.std(scores)

超参数优化：　通过搜索空间，自动搜索最佳参数，并使用预定义或自定义的评分函数对结果进行评分

from sklearn import grid\_search, svm

search\_grid = [

{'C': np.logspace(start=-2, stop=3, num=6, base=10.0), 'kernel': ['linear']},

{'C': np.logspace(start=-2, stop=3, num=6, base=10.0), 'kernel': ['rbf'], 'gamma': [0.001, 0.0001]},

]

search\_func = grid\_search.GridSearchCV(estimator=svm.SVC(), param\_grid=search\_grid, scoring='accuracy', n\_jobs=-1, iid=False, refit=True, cv=10)

search\_func.fit(X, y)

print search\_func.best\_estimator\_

print search\_func.best\_params\_

print search\_func.best\_score\_

### 机器学习

#### 获取模型信息

print model.get\_params()

model = build\_model()

model.fit(X\_train, y\_train)

y\_predicted = model.predict(X\_test)

回归是函数逼近

回归缺点：线性回归采用的系数值上限和下限无法控制，对给定的数据，回归容易过拟合，

泛化能力不佳，解决方法是正则化方法

#### 线性回归

目标是找到一个好的权重列向量W, 当与观测向量相乘时能很好地近似目标值

即X \* W = Y，求W使代价函数sum(|y – x\*w|2)最小

**from sklearn.linear\_model import LinearRegression**

**from sklearn.metrics import mean\_squared\_error**

model = LinearRegression(normalize=True, fit\_intercept=True)

model.fit(X\_train, y\_train)

model.coef\_, model.intercept\_ #回归系数和截距

y\_predicted = model.predict(X\_test)

mean\_squared\_error(y, y\_predicted) #均方误差

多项式特征建模

from sklearn.preprocessing import PolynomialFeatures

poly\_features = PolynomialFeatures(2)

X\_train\_poly = poly\_features.fit\_transform(X\_train)

X\_test\_poly = poly\_features.fit\_transform(X\_test)

model.fit(X\_train\_poly)

多项式特征，即模型的属性数量增加，即模型的复杂度提高，模型的训练集误差减少，即意味着低偏差，但泛化能力退化，如何解决？

正则化方法

在训练线性回归模型时，有的权重很大，导致模型不稳定 ，正则化对高权重进行惩罚，从而减少系数的值

代价函数sum(|y – x\*w|2) + alpha\*sum(wTw) L2

参数alpha决定了缩减的幅度，alpha越大，缩减幅度越大，系数的值越趋于0

线性回归模型对噪声很敏感，即使加入少量的噪声，线性回归的权重会变化很大，而正则化在噪音数据环境下相对稳定

from sklearn.linear\_model import Ridge

model = Ridge(normalize=True, alpha=0.5)

model.fit(X\_train\_poly, y\_train)

mean\_squared\_error(y\_test, model.predict(X\_test\_poly))

LASSO: Least absolute shrinkage and selection operator

代价函数sum(|y – x\*w|2) + alpha\*sum|w| L1

L2对相关联的变量的权重相同，而L1倾向于spare的权重，即大多数权重缩减为0，对于相关联的变量，只保留其中的一个。LASSO可以用来特征选择

from sklearn.linear\_model import Lasso

model = Lasso(normalize=True, alpha=0.22)

model.fit(x, y)

idx = [i for i, coef in enumerate(model.coef\_) if abs(coef) > 0.0]

x\_selected = x[:, idx] 特征选择

因为只保留最重要的变量，LASSO避免了过拟合，但LASSO引起预测能力损失，对于变量间相互关联的情形，LASSO只会选择其中的一个，而L2则缎带每个相应变量分配相同的权重。因此L2比L1预测能力要高一些

交叉验证优化超参数

def optimize\_model(x, y):

kfold = KFold(y.shape[0], 5)

model = Ridge(normalize=True)

alpha\_range = np.logspace(-5, 5, 11)

grid\_param = {"alpha": alpha\_range}

grid = GridSearchCV(estimator=model, param\_grid=grid\_param, cv=kfold, scoring='mean\_squared\_error')

grid.fit(x, y)

for i, param\_result in enumerate(grid.grid\_scores\_):

print 'Fold %d mse %0.2f' % (i, abs(param\_result[1])), param\_result[0]

print grid.best\_params\_

return grid.best\_estimator\_

model = optimize\_model(X\_train\_poly, y\_train)

print 'test set: mse = %f' % model\_worth(y\_test, model.predict(X\_test\_poly))

#### 逻辑回归

线性回归模型 + 01非线性映射

简单采用硬阈值函数：sign(X\*W) = Y

为了平滑硬阈值函数并且预测样本属于某个类别的概率，逻辑回归模型使用了logit函数。它的输出是一个0~1之间的实数，表示观测值属于类别1的概率

Prob(yi=1|Xi) = logistic(Xi\*W)

logistic(a) = 1/(1+e-a)

**from sklearn.linear\_model import LogisticRegression**

clf = LogisticRegression()

clf.fit(X\_train, y\_train)

from sklearn.metrics import classification\_report

print classification\_report(y\_test, clf.predict(X\_test))

#### native bayes (多类分类器)

后验概率通过类的先验概率乘以似然函数再除以证据值

**from sklearn.naive\_bayes import GaussianNB**

clf = GaussianNB()

clf.fit(X\_train, y\_train)

print classification\_report(y\_test, clf.predict(X\_test))

#### KNN (多类分类器, 回归)

from sklearn.neighbors import KneighborsClassifier

用于分类，预测值是近邻的k个目标值的投票表决

clf = KNeighborsClassifier(3)

clf.fit(X\_train, y\_train)

print classification\_report(y\_test, clf.predict(X\_test))

from sklearn.neighbors import KneighborsRegressor

用于回归，预测值是近邻的k个目标值的平均

#### decision tree (多类)

优点：易于解释

缺点：容易过拟合，类别不平衡严重

如何寻找特征测试条件，从而将数据集合分割成多个纯的子集？

信息增益 ＝ 父节点熵 － 子节点平均熵， 度量哪个属性对于分类最有用

假定父节点

主演 奥斯卡得奖 票房 观看

Y Y N Y

Y N Y N

N N Y Y

N Y Y Y

父节点熵 = - sum( p(Y|cases)\*log(p(Y|cases) )

= - (¾ log ¾ + ¼ log ¼) = 0.811

属性：主演

主演＝Y => 子节点1

主演=N => 子节点2

子节点1, -( ½ log½ + ½ log½)

子节点2, -( 2/2 log2/2 + 0 )

信息增益 ＝ 父节点熵 － (子节点1+子节点2)/2 ＝ 0.311

重复计算属性为奥斯卡得奖，为票房的信息增益，比较结果，具有最高信息增益的属性成为决策树的特征测试条件，将父节点划分为比较纯的子集，然后针对各子集，重复以上过程

from sklearn import tree

dtree = tree.DecisionTreeClassifier(criterion=’entropy’)

dtree.fit(X, y)

# 查看决策树

def output\_dtree(dtree, output="iris.pdf"):

import pydot

from sklearn.externals.six import StringIO

dotfile = StringIO()

tree.export\_graphviz(dtree, out\_file=dotfile)

graph = pydot.graph\_from\_dot\_data(dotfile.getvalue())

graph[0].write\_pdf(output)

#### SVM (多类分类器，回归)

SVM对不同尺度的变量和大数值很敏感，需要归一化

不平衡类，算法倾向于频繁的类别（即该类包含较多的样本数），除了重采样，还可以采用根据类别出现的频率设置相应的惩罚参数Ｃ

from sklearn.svm import SVC

clf = SVC(kernel='rbf', degree=2, random\_state=101)

clf.fit(X\_train, y\_train)

print classification\_report(y\_test, clf.predict(X\_test))

from sklearn.svm import SVR

hypothesis = SVR()

hypothesis.fit(X\_train, y\_train)

from sklearn.metrics import mean\_absolute\_error

print 'MAE', mean\_absolute\_error(y\_test, hypothesis.predict(X\_test))

#### ensemble组合策略

from sklearn.ensemble import BaggingClassifier

from sklearn.neighbors import KNeighborsClassifier

hypothesis = BaggingClassifier(KNeighborsClassifier(n\_neighbors=1), max\_samples=0.7, max\_features=0.7, n\_estimators=100)

### NLP

词频统计

from collections import Counter

sentence = “I am qzlin”

words = sentence.split()

word\_count = Counter(words)

在程序中使用json在模块间传送信息，字典就是最适合的数据类型。Json ↔ dictionary

在机器学习程序开发中，常常有不同模块生成不同的特征值，此时使用元组串联是一个很好的选择

直方图

from collections import Counter

c = Counter([0, 1, 2, 0])

# c = {0: 2, 1: 1, 2: 1}

word\_counts = Counter(document)

for word, count in word\_counts.most\_common(10):

print word, count

文本分词化

段落级别, 语句级别, 词级别

sent\_tokenize加载一个预置的分词化模型，包里还针对不同的语言预先训练了多种分词器实例

word\_tokenize函数使用了正则表达式来将句子分解为词

from nltk.tokenize import sent\_tokenize, word\_tokenize

from collections import defaultdict

sents = sent\_tokenize(doc)

word\_dict = defaultdict(list)

for idx, sent in enumerate(sents):

word\_dict[i].extend(word\_tokenize(sent))

删除停用词

所有的文本挖掘应用都必须找出文本中的关键短语，一个信息检索应用也需要找出关键词来优化检索、评估检索结果，文本分类系统也需要关键短语来作为分类器所需的特征

from nltk.corpus import stopwords

stopwords.words(‘english’)

文本标准化：

Import nltk.stem

porter = stem.porter.PorterStemmer() 词提取

p\_words = [porter.stem(w) for w in input\_words]

wordnet\_lemm = stem.WordNetLemmatizer() 词形还原

wn\_words = [wordnet\_lemm.lemmatize(w) for w in input\_words]

文本表示法

将文本转化为数值特征

|  |  |  |  |
| --- | --- | --- | --- |
|  | Word1 | Word2 | Word3 |
| Doc1 | value |  |  |
| Doc2 |  |  |  |

Value一般为如下之一：

二进制值，表示这个词在给定的文档中存在或不存在

频率，表示这个词在给定的文档中出现的总次数

TFIDF： 评估一个词对于文档集或语料库的一份文档的重要程度 TF-IDF = -TF\*log(DF)

词频TF = 频率/某文档总词数

文档频率DF = 含该词的文档数/所有文档总数

from sklearn.feture\_extraction\_text import CountVectorizer, TfidfTransformer

from nltk.corpus import stopwords

count\_v = CountVectorizer(stop\_words= stopwords.words(‘english’))

tdm = count\_v.fit\_transform(doc\_list) 文档-特征词矩阵

tdm.get\_shape() 行列维数

tfidf = TfidfTransformer()

tdm\_tfidf = tfidf.fit\_transform(tdm)

# R

统计计算和统计制图

CRAN: Comprehensive R Archive Network

## install R and RStudio

/etc/apt/sources.list

deb https://mirrors.tuna.tsinghua.edu.cn/CRAN/bin/linux/ubuntu trusty/

sudo apt-get update

sudo apt-get install r-base

//install RStudio

apt-get install libjpeg62

wget http://download1.rstudio.org/rstudio-0.98.1062-amd64.deb

sudo dpkg -i \*.deb

rm \*.deb

$which R R命令行方式

$R

$which Rscript R脚本运行方式

helloworld.r

#!/usr/bin/Rscript

print("Hello World")

$chmod u+x helloworld.r

$./helloworld.r

$which rstudio R IDE

$rstudio 启动RStudio环境

## <R语言初学者指南>

### R语法

>help(”boxplot“)

>library(MASS) 载入包

>q() 退出

>getwd()

>setwd(file="C:/.../") 改变当前目录

>rm(list = ls(all=TRUE) ) 删除当前所有变量

一般来说，R中的变量名最好使用大写字母开头，这样可以避免将它和一些内部函数名混淆

在R中使用内部函数前最好查看一下相应的帮助文件，以确保你知道这个函数如何处理缺失值

#表示注释

<-表示赋值

向量

Wing <- c(59, 55, 53.5, 55) c()表示连接(Concatenate)

Wing[1]

Wing[1:3]

Id <- rep(c(1, 2, 3, 4), each=8) 生成8个1,然后是8个2,...

Id <- rep(1:4, each=8)

Id <- rep(seq(from=1, to=4, by=1), each=8)

Head <- c(33, 22, 11, 44)

向量嵌套向量

Z <- cbind(Wing, Head) 以列的方式结合变量

Z[, 1] 第一列

Z[1:8, 1] 子数组

Z[2, ] 第二行

Size <- dim(Z) 返回维数(行，列）

Z2 <- rbind(Wing, Head) 以行的方式结合变量

矩阵

Dmat <- matrix(nrow=8, ncol=4) 矩阵

colnames(Dmat) <- c("Wing", "Tarsus", "Head", "Wt") 给矩阵加列名

Dmat[3:5, 2:3] 子矩阵

Dmat2 <- as.matrix(cbind(Wing, Head)) 生成矩阵

多维数组

a <- array(1:24, c(2, 3, 4))

4组两行三列

a[1, 2, 3] 访问第3组，第1行第2列元素

数组框

Dfrm <- data.frame(WC=Wing, TS=Tarsus, HD=Head, W=Wt, Wsq=sqrt(Wt)) 数据框

每一行表示一个样本的不同观察值

Dfrm$WC 访问列

Dfrm[ c(‘WC’, ‘W’) ] 访问数据框某几列

列表

Y <- list(x1 = c(1, 2,3),

x2 = c("a", "b"),

x3 = matrix(nrow=2, ncol=2)) 同python and javascript的list

Y$x1 访问元素

引入list函数的原因是因为几乎所有R中的函数（比如线性回归，广义线性模型，t-检验等等）的输出结果都是保存在列表中

条件指令

if (a) {

x < -1

} else {

x <- 2

}

ifelse (a, x <- 1, x < -2) 相当于a ? x=1 : x=2

循环

for (i in 1:27) {

YourFileName <- paste(i, ".jpg", sep="") 将变量连接为字符串

jpeg(file=YourFileName) 打开一个jpg文件

plot(x, y, ...)

dev.off() 关闭jpg文件

}

函数

函数中最后一行代码的结果将是返回信息, 也可以用return提前返回

Index.function <- function(Spec, Choice1="Sannon") {

if (Choice1 == "Richness") {

Index <- rowSums(Spec>0, na.rm=TRUE)

}

if (Choice1 == "Shannon") {

RS <- rowSums(Spec, na.rm=TRUE)

prop <- Spec / RS

Index <- -rowSums(prop\*log10(prop), na.rm=TRUE)

}

list(Index=Index, MyChoice=Choice1)

}

### 数据获取

以NA表示缺失值，一般最好以Excel中的第一列来识别样本单元，第一行作为变量名

Squid <- read.table(file="C:/RBook/squid.txt", header=TRUE) 将csv读入数据框中

write.table(Squid, file="Squid.txt", sep=",", quote=TRUE, append=FALSE, na="NA") 写csv

### 数据预处理 （一般针对数据框）

names(Squid) 返回变量名（列名）

str(Squid) 显示变量的属性

unique(Squid$Sex) 等价于levels(Squid$Sex)

attach(Squid)

Squid3 <- Squid[Sex==1 | Location==2, ] 过滤子集

Squid4 <- Squid[order(Sex), ] 排序

deattach(Squid)

合并数据框，Sample是两个数据框的相同变量

SquidMerged <- merge(Sq1, Sq2, by="Sample", all=TRUE)

变换列产生新列

Squid$fLocation <- factor(Squid$Location)

Squid$fSex <- factor(Squid$Sex, levels=c(1,2), labels=c("M", "F"))

在做任何统计分析之前，你应该将数据可视化

最后一部分data=Squid告诉R,变量在数据框 Squid中，从而lm()函数参数可以使用使用变量GSI,Location,...

有些函数没有data数据框，可以用Squid$GSI访问变量，若嫌麻烦，可以用attach and deattach限定上下文

M1 <- lm(GSI~factor(Location)+factor(Year), data=Squid)

attach(Squid）

...直接使用变量GSI, Location,...

deattach(Squid)

### 数据分析

tapply(y, x, FUN=mean) 根据x的不同水平对y使用FUN函数

sapply(y, FUN=mean)

lapply(y, FUN=mean) 对y的每一个变量使用FUN的函数

summary(y) 计算基本信息

y可以是一个变量，cbind命令的输出，或数据框

table(x,y) 计算列联表

### 数据可视化

plot(y, x) y对x的图形

不建议写成plot(Y~X, data=Veg)，因为Y~X符号是用来告诉R，Y作为X的函数。但并不是每个涉及变量的散点图都有一个因果关系

lines(x, y) 在已存在图形上添加线

order(x) 排序

M <- loess(y ~ x) 使用LOESS平滑

fitted(M) 得到拟合值

生成具有4个面板的图形窗口， mar选项指定每个图形周围空白的大小

op <- par(mfrow=c(2,2), mar=c(3,3,2,1))

pie(Cases) 饼图

pie3D(Cases)

par(op)

barplot(x) 条形图

boxplot(y) 盒形图

pairs(x) 生成多面板散点图

arrows(x1,y1, x2,y2) 绘制箭头

text(x,y, "hello") 在图形上添加文本

points(x, y) 在图形上添加点

legend("topleft", MyText, lty=c(1, 2, 3)) 添加图例

title(MyText) 添加标题

高维数据绘图

library(lattice)

xyplot(SAL ~ MyTime | factor(Station), ...) 散点图

~连接的变量分别表示y轴和x轴

|后条件变量是为了生成多面板

条件变量通常是一个因子，也可以是连续变量，当使用连续的变量作为条件变量时，它的每一个值在默认情况下都被理解为一个离散值，然而，此类变量通常具有很多的值，此时我们就需要将其分割为一些区间。

bwplot(SAL ~ factor(Month) | Area, ...) 盒形图

dotplot(factor(Month) ~ SAL | Station, ...) 点图

histogram(~ SAL | Station, ...) 直方图

barchart()

contourplot()

levelplot()

...

## <R and Ruby数据分析之旅>

$R

>install.packages('tree') 安装R程序包

>installed.packages() 查看已安装程序包

>library(tree) 使用程序包

因子factor

值可以分为名义型nominal, 顺序型ordinal 和连续型continuous

在R中，名义值和顺序值都用因子来表示

colors <- c('green', 'red', 'blue')

factor(colors) nominal

默认情况下，对于字符数据，因子各水平的排序是根据字母顺序创建的。但是，我们也可以在创建因子的时候指定各水平的顺序

results <- c('poor', 'average', 'good')

factor(results, order=TRUE, levels=results) ordinal

数据框

按列构造

league <- data.frame(team, home\_wins, home\_draws, home\_losses, away\_wins, away\_draws, away\_losses)

league[ c('team', 'home\_wins') ]

league$team

league$team[league$home\_wins > 8]

league$team[ league$away\_wins > league$home\_wins ]

等价于

with(league, team[away\_wins > home\_wins] ) with上下文管理

league <- merge(league, points, by='team') 合并数据框（添加列方式）

league[order(-league$pts, ]) 按字段pts降序

league <- rbind(league, league\_another) 合并数据框（添加行方式）

R包自带数据

install.packages('ggplot2')

library(ggplot2)

mpg

str(mpg)

summary(mpg)

从csv文件导入数据

epl <- read.csv('\*.csv')

从MySQL数据库导入数据

install.packages( c('DBI', 'RMySQL') )

con <- dbConnect(MySQL(), host='localhost', dbname='eps', user='root', password='root')

league <- dbGetQuery(con, 'select \* from league')

dbDisconnect(conn)

绘图

如果你将图表输出到文件，你可以使用pdf(), png(), jpeg(), bmp(), win.metafile()及其他函数来设置输出文件的格式。

pdf('some\_file\_name.pdf')

#do some chart plotting

dev.off()

R中绘图是逐层进行的。在绘制了图形之后（通过plot()函数），你可以继续添加新的图层，在其中包含新的信息

install.packages('ggplot2')

程序包ggplot2基本思想：复合多个图层，各层之是相互堆叠起来。

统计绘图实际上是从数据到几何对象的审美属性的一个映射过程，而这些几何对象应当可以进行统计变换，并且可以在坐标系统内画出。审美属性包括颜色、形状、大小，而几何对象类型可以是点、线条或柱形等性质

具体见p52

# Spark

## 《Spark快速大数据分析》

**Spark：批处理，迭代算法，交互式查询，流处理**



Spark Core实现了Spark的基本功能，包含任务调度，内存管理，错误恢复，与存储系统交互，RDD(Resilient distributed dataset:弹性分布式数据集)等

Spark SQL是Spark用来操作结构化数据的程序包，Spark SQL还支持开发者将SQL和传统的RDD编程的数据操作方式相结合

Spark Streaming是Spark提供的对实时数据进行流式计算的组件

MLib: 机器学习库

GraphX: 用来操作图的程序库，可以进行并行的图计算

数据科学家就是主要负责分析数据并建模的人。数据科学家有可能具备SQL、统计、预测建模（机器学习）等方面的经验，以及一定的使用python, matlab or R语言进行编程的能力

数据科学家在使用R或Pandas等**传统数据分析工具时所能处理的数据集受限于单机**，而有了**Spark**,就能**处理更大数据**规模的问题。

对工程师来说，Spark为开发用于集群并行执行的程序提供了一条捷径。通过封装，Spark不需要开发者关注如何在分布式系统上编程这样的复杂问题，也无需过多关注网络通信和程序容错性。Spark已经为工程师提供了足够的接口来实现常见的任务，以及对应用进行监视、审查和性能调优

**Hadoop MapReduce在迭代计算和交互计算的任务上表现得效率低下**。因此，**Spark从一开始就是为交互式查询和迭代算法设计**的，同时还支持内存和高效的容错机制

Shell工具类似R,Python, Scala, matlab, Bash等，只能使用单机的硬盘和内存来操作数据，而Spark shell可用来与分布式存储在许多机器的内存或者硬盘上的数据进行交互，并且处理过程的分发由Spark自动控制完成

在Spark中，我们通过对**分布式数据集**的操作来表达我们的计算意图，这些计算会自动地在集群上并行进行。这样的数据集被称为RDD, RDD是Spark对分布式数据和计算的基本抽象

驱动器程序Spark shell启动时自动创建SparkContext对象，即sc变量，从而可以创建RDD,在RDD上执行操作

驱动器程序一般要管理多个执行器节点，比如我们在集群上运行count()，那么不同的节点会统计文件的不同部分的行数。Spark会自动将函数如filter里的fun函数发到各个执行器节点上。这样你就可以在单一的驱动器程序中编程，并且让代码自动运行在多个节点上

### install spark

$wget <http://d3kbcqa49mib13.cloudfront.net/spark-2.0.0-bin-hadoop2.7.tgz>

$tar -xf spark…tgz

$cd spark…

### 命令行（交互式分析数据）

启动Spark中Python的shell

**$bin/pyspark**

// actions, return values

lines = sc.textFile("README.md")

lines.count()

lines.first()

//transformation, return a new RDD

pythonlines = lines.filter(lambda line: "Python" in line)

textFile.map(lambda line: len(line)).reduce(lambda a,b: a if (a>b) else b)

wordCounts = textFile.flatMap(lambda line: line.split()).

map(lambda word: (word, 1)).

reduceByKey(lambda a, b: a+b)

wordCounts.collect()

### 脚本（独立应用）

pi.py

from random import random

from pyspark import SparkContext

def sample(p):

x, y = random(), random()

return 1 if x\*x + y\*y < 1 else 0

NUM\_SAMPLES = 100000

sc = SparkContext("local", "Pi App") 通过集群URL和应用名初始化sc,然后同shell一样操作

count = sc.parallelize(xrange(0, NUM\_SAMPLES)).map(sample).reduce(lambda a,b: a+b)

print "Pi is roughly %f" % (4.0 \* count / NUM\_SAMPLES)

**$bin/spark-submit** pi.py

spark-submit脚本会帮我们引入python程序的spark依赖。这个脚本为Spark的PythonAPI配置好了运行环境

查看spark自带的范例

$tree examples/src/main/python

$cat examples/src/main/python/pi.py

$bin/spark-submit examples/src/main/python/pi.py

### RDD编程

RDD其实就是分布式的元素集合。在Spark中，对数据的所有操作不外乎**创建RDD、转化已有RDD以及调用RDD操作进行求值**。而在这一切背后，Spark会自动将RDD中的数据分发到集群上，并将操作并行化执行

Spark中的RDD就是一个不可变的分布式对象集合。每个RDD都被分为多个分区，这些分区运行在集群中的不同节点上

Spark只会惰性计算这些RDD。它们只有第一次在一个行动操作中用到时，才会真正计算

默认情况下，Spark的RDD会在你每次对它们进行行动操作时重新计算。可以使用RDD.persist()让Spark把这个RDD缓存下来，Spark会把RDD的内容保存到内存中（以分区方式存储到集群中的各机器上）。在实际操作中，你会经常用persist()来把数据的一部分读取到内存中，并反复查询这部分数据

每个Spark程序或shell会话都按如下方式工作：

1. 从外部数据创建出输入RDD
2. 使用转化操作对RDD进行转化，以定义新的RDD
3. **对需要被重用的中间结果RDD执行persist()操作**
4. 使用行动操作触发一次并行计算

创建ADD

lines = sc.textFile("README.md") 读取一个外部数据集

list = ["pands", "i like pandas"]

lines = sc.parallelize(list) 将已有集合转RDD

RDD转换操作

返回一个新的RDD，

惰性求值，

不会改变已有的数据，会返回一个全新的RDD

rdd.map(func)

rdd.flapMap(func)

rdd.filter(func)

rdd.distinct(func) 去重

rdd.sample(withReplacement, fraction, [seed]) 采样

rdd.union(other)

rdd.intersection(other)

rdd.subtract(other)

rdd.cartesian(other) 笛卡尔积

RDD行动操作 (触发并行计算，返回值)

rdd.collect() 返回RDD中的所有元素

rdd.count() RDD中的元素个数

rdd.countByValue() 各元素在RDD中出现的次数

rdd.take(num) 从RDD中返回num个元素

rdd.top(num) 从RDD中返回前num个元素

rdd.takeOrdered(num)(myOrdering)

rdd.takeSample(withReplacement, num, [seed])

rdd.reduce(func) 整合RDD中所有数据

rdd.fold(initValue)(func) 同reduce,只是需要初始值

rdd.aggregate(initValue)(seqOp, combOp)

rdd.foreach(func)

持久化（缓存）

如果简单地对RDD调用行动操作，Spark每次都会重算RDD以及它的所有依赖。这在迭代算法中消耗大，因为迭代算法常常会多次使用同一组数据

为了避免多次计算同一个RDD，可以让Spark对数据进行持久化。当我们让Spark持久化存储一个RDD时，计算出RDD的节点会分别保存它们所求出的分区数据

RDD不同的持久化级别：在Scala and Java中，默认情况下persist()会把数据以序列化的形式缓存在JVM的堆空间中。在Python中，我们会始终序列化要持久化存储的数据，所以持久化级别默认值就是以序列化后的对象存储在JVM堆空间中。

如果要缓存的数据太多，内存中放不下，Spark会自动利用最近最小使用LRU的缓存策略把最老的分区从内存中移除