SciPy 基础

金　林
中南财经政法大学统计系
jinlin82@qq.com

2020 年 2 月 1 日

Facts

1. Initial release: Around 2001
2. Stable release: 0.18.1 / 22 September 2016
3. Website: `http://www.scipy.org`
4. History: `http://scipy.github.io/old-wiki/pages/History_of_SciPy`

What is SciPy?

1. SciPy is an open source Python library used for scientific computing and technical computing.
2. SciPy builds on the NumPy array object
3. and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy.
4. There is an expanding set of scientific computing libraries that are being added to the NumPy stack every day.
5. The NumPy stack is also sometimes referred to as the SciPy stack.

What can SciPy do?

1. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data.
2. With SciPy an interactive Python session becomes a data-processing and system-prototyping environment rivaling systems such as MATLAB, IDL, Octave, R-Lab, and SciLab.
3. The additional benefit of basing SciPy on Python is that this also makes a powerful programming language available for use in developing sophisticated programs and specialized applications.
4. Scientific applications using SciPy benefit from the development of additional modules in numerous niches of the software landscape by developers across the world.
5. Everything from parallel programming to web and data-base subroutines and classes have been made available to the Python programmer.

Sub-packages

1. constants: physical constants and conversion factors
2. cluster: hierarchical clustering, vector quantization, K-means
3. fftpack: Discrete Fourier Transform algorithms
4. integrate: numerical integration routines
5. interpolate: interpolation tools
6. io: data input and output
7. lib: Python wrappers to external libraries
8. linalg: linear algebra routines
9. misc: miscellaneous utilities (e.g. image reading/writing)

Sub-packages

10. ndimage: various functions for multi-dimensional image processing
11. optimize: optimization algorithms including linear programming
12. signal: signal processing tools
13. sparse: sparse matrix and related algorithms
14. spatial: KD-trees, nearest neighbors, distance functions
15. special: special functions
16. stats: statistical functions
17. weave: tool for writing C/C++ code as Python multiline strings

Sub-package

1. Scipy sub-packages need to be imported separately, for example:
   ```
   from scipy import stats
   ```

Intro

1. This module contains a large number of probability distributions as well as a growing library of statistical functions.
2. Each univariate distribution is an instance of a subclass of rv_continuous (rv_discrete for discrete distributions):
   1. rv_continuous([momtype, a, b, xtol, ...]) A generic continuous random variable class meant for subclassing.
   2. rv_discrete([a, b, name, badvalue, ...]) A generic discrete random variable class meant for subclassing.
3. Over 80 continuous random variables (RVs) and 10 discrete random variables have been implemented using these classes.
4. Besides this, new routines and distributions can easily added by the end user.

常见连续分布

beta	A beta continuous random variable.
cauchy	A Cauchy continuous random variable.
chi2	A chi-squared continuous random variable.
expon	An exponential continuous random variable.
f	An F continuous random variable.
gamma	A gamma continuous random variable.
ncf	A non-central F distribution continuous random variable
nct	A non-central Student’ s T continuous random variable
norm	A normal continuous random variable.
norminvgauss	A Normal Inverse Gaussian continuous random variable
pareto	A Pareto continuous random variable.
t	A Student’ s T continuous random variable.
uniform	A uniform continuous random variable.

常见离散分布

bernoulli	A Bernoulli discrete random variable.
binom	A binomial discrete random variable.
geom	A geometric discrete random variable.
hypergeom	A hypergeometric discrete random variable.
nbinom	A negative binomial discrete random variable.
poisson	A Poisson discrete random variable.
randint	A uniform discrete random variable.

常见多元分布

multivariate_normal	A multivariate normal random variable.
matrix_normal	A matrix normal random variable.
dirichlet	A Dirichlet random variable.
wishart	A Wishart random variable.
invwishart	An inverse Wishart random variable.
multinomial	A multinomial random variable.

例子：正态分布

```
from scipy import stats
import matplotlib.xyplot as plt
import numpy as np

mean, var, skew, kurt=stats.norm.stats(moments="mvsk")
x=np.linspace(-3, 3, 100)
plt.plot(x, stats.norm.pdf(x),  label='norm pdf')
plt.plot(x, stats.norm.pdf(x, 3, 2),  label='norm pdf')

### Freeze the distribution and display the frozen pdf
rv=stats.norm(3,2)
rv.ppf(0.5)
rv.pdf(3)
rv.cdf(10)

### Generate random numbers
r = stats.norm.rvs(size=1000)
plt.hist(r, density=True, histtype='stepfilled', alpha=0.2)
```

例子：F 分布

```
from scipy import stats
import matplotlib.xyplot as plt
import numpy as np

mean, var, skew, kurt=stats.f.stats(3, 5, moments="mvsk")
x=np.linspace(0.01, 6, 100)
plt.plot(x, stats.f.pdf(x, 3, 5),  label='norm pdf')
plt.plot(x, stats.f.pdf(x, 3, 2),  label='norm pdf')

### Freeze the distribution and display the frozen pdf
rv=stats.f(3,2)
rv.ppf(0.5)
rv.pdf(3)
rv.cdf(10)

### Generate random numbers
r = stats.f.rvs(3,2,size=1000)
plt.hist(r, density=True, histtype='stepfilled', alpha=0.2)
```

例子：二项分布

```
from scipy import stats
import matplotlib.xyplot as plt
import numpy as np

fig, ax = plt.subplots(1, 1)

mean, var, skew, kurt = stats.binom.stats(5, 0.4, moments='mvsk')

x = np.arange(stats.binom.ppf(0.01, 5, 0.4), stats.binom.ppf(0.99, 5, 0.4))
ax.plot(x, stats.binom.pmf(x, 5, 0.4), 'bo', ms=8, label='binom pmf')
ax.vlines(x, 0, stats.binom.pmf(x, 5, 0.4), colors='b', lw=5, alpha=0.5)

rv = stats.binom(5, 0.4)
ax.vlines(x, 0, rv.pmf(x), colors='k', linestyles='-', lw=1,
label='frozen pmf')
ax.legend(loc='best', frameon=False)
```

例子：多元正态分布

```
from scipy import stats
import matplotlib.xyplot as plt
import numpy as np

x = np.linspace(0, 5, 10, endpoint=False)
y = stats.multivariate_normal.pdf(x, mean=2.5, cov=0.5); y

fig1 = plt.figure()
ax = fig1.add_subplot(111)
ax.plot(x, y)

x, y = np.mgrid[-1:1:.01, -1:1:.01]
pos = np.dstack((x, y))
rv = stats.multivariate_normal([0.5, -0.2], [[2.0, 0.3], [0.3,
    0.5]])
fig2 = plt.figure()
ax2 = fig2.add_subplot(111)
ax2.contourf(x, y, rv.pdf(pos))
```

Common Methods

rvs(loc=0, scale=1, size=1)	Random variates.
pdf(x, loc=0, scale=1)	Probability density function.
logpdf(x, loc=0, scale=1)	Log of the probability density function.
cdf(x, loc=0, scale=1)	Cumulative distribution function.
logcdf(x, loc=0, scale=1)	Log of the cumulative distributionfunction.
sf(x, loc=0, scale=1)	Survival function (also defined as 1 - cdf, but sf is some
logsf(x, loc=0, scale=1)	Log of the survival function.
ppf(q, loc=0, scale=1)	Percent point function (inverse of cdf —percentiles).
isf(q, loc=0, scale=1)	Inverse survival function (inverse of sf).
moment(n, loc=0, scale=1)	Non-central moment of order n
stats(loc=0, scale=1, moments='mv')	Mean('m'), variance('v'), skew('s'), and/or ku
entropy(loc=0, scale=1)	(Differential) entropy of the RV.
fit(data, loc=0, scale=1)	Parameter estimates for generic data.
expect(func, args=(), **kwds)	Expected value of a function (of one argument) with res
median(loc=0, scale=1)	Median of the distribution.
mean(loc=0, scale=1)	Mean of the distribution.
var(loc=0, scale=1)	Variance of the distribution.
std(loc=0, scale=1)	Standard deviation of the distribution.
interval(alpha, loc=0, scale=1)	Endpoints of the range that contains alpha percent of t

几个统计函数

describe(a[, axis, ddof, bias])	Compute several descriptive statistics of the passed array.
gmean(a[, axis, dtype])	Compute the geometric mean along the specified axis.
hmean(a[, axis, dtype])	Calculate the harmonic mean along the specified axis.
kurtosis(a[, axis, fisher, bias])	Compute the kurtosis (Fisher or Pearson) of a dataset.
kurtosistest(a[, axis])	Test whether a dataset has normal kurtosis.
mode(a[, axis, nan_policy])	Return an array of the modal (most common) value in the pa
moment(a[, moment, axis])	Calculate the nth moment about the mean for a sample.
normaltest(a[, axis, nan_policy])	Test whether a sample differs from a normal distribution.
skew(a[, axis, bias, nan_policy])	Compute the skewness of a data set.
skewtest(a[, axis, nan_policy])	Test whether the skew is different from the normal distributio
kstat(data[, n])	Return the nth k-statistic (1<=n<=4 so far).
kstatvar(data[, n])	Returns an unbiased estimator of the variance of the k-statisti
variation(a[, axis, nan_policy])	Compute the coefficient of variation, the ratio of the biased st
find_repeats(arr)	Find repeats and repeat counts.

Fitting Distributions

Building Specific Distributions

Analysing One Sample

Comparing two samples

Kernel Density Estimation

`scipy.stats.gaussian_kde`

1. Kernel density estimation is a way to estimate the probability density function (PDF) of a random variable in a non-parametric way.
2. gaussian_kde works for both uni-variate and multi-variate data.
3. It includes automatic bandwidth determination.
4. The estimation works best for a unimodal distribution; bimodal or multi-modal distributions tend to be oversmoothed.

Linear algebra functions

1. numpy.linalg for more linear algebra functions.
2. Note that although scipy.linalg imports most of them, identically named functions from scipy.linalg may offer more or slightly differing functionality.