Scipy

Prof.Dr. Bahadır AKTUĞ
JFM212 Python ile Mühendislik Uygulamaları

*Kaynakça bölümünde verilen kaynaklardan derlenmiştir.

Scipy

- Scipy (Scientific Python), Python’a birçok bilimsel ve mühendislik hesaplamalarında ihtiyaç duyulan matematiksel işlemler için gerekli modül ve fonksiyonları sağlamaktadır (Örneğin Bessel fonksiyonları ya da QR faktörizasyonu gibi).
- Bu yönüyle Scipy, Matlab'daki "toolbox"lar gibi modüllerden ve her bir modüldeki fonksiyonlardan oluşmaktadır.
- Söz konusu fonksiyonlar, Scipy içindeki alt modüllerde bulunmaktadır. Bu nedenle, bu alt modüller kullanılmadan önce "Scipy" yerine bu alt modüller ayrı ayrı "import" edilmelidir.
- Python ile birlikte Scipy gelmemekte, ayrıca yüklenmesi gerekmektedir.
- Scipy Numpy ile uyumlu çalışmaktadır.

Scipy

- Scipy alt modülleri

scipy.cluster	Küme analizi
scipy.constants	Matematiksel ve fiziksel sabitler
scipy.fftpack	Fourier Dönüşümü
scipy.integrate	İntegral Fonksiyonları
scipy.interpolate	Enterpolasyon (Aradeğerleme)
scipy.io	Veri Giriş ve Çıkışı
scipy.linalg	Lineer Cebir Fonksiyonları
scipy.ndimage	n-boyutlu görüntü paketi
scipy.odr	Ortogonal Mesafe Regresyonu
scipy.optimize	Optimizasyon
scipy.signal	Signal İşleme
scipy.sparse	Spars Matrisler
scipy.spatial	Mekânsal Veri Yapıları ve Algoritmalar
scipy.special	Özel matris fonksiyonları
scipy.stats	İstatistik

Scipy.io

- Scipy büyük oranda Numpy veri yapılarına dayalıdır.
- Scipy.io ise veri kaydetme ve yükleme için pratik araçlar içerir.
- Matlab *.mat dosyası formatında okuma/yazma yapılabilir:

```
from scipy import io as spio
a = np.ones((3, 3))
spio.savemat('file.mat', {'a': a})
data = spio.loadmat('file.mat', struct_as_record=True)
data['a']
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

Scipy.io

- Benzer şekilde çeşitli görüntü formatları doğrudan okunabilir:

```
from scipy import misc
misc.imread('fname.png')
array(...)
```

- Numpy'nin de çeşitli metin dosyaları okuma fonksiyonları vardır:

```
numpy.loadtxt()/numpy.savetxt()
numpy.genfromtxt()/numpy.recfromcsv()
```

- Binary Numpy formatında kaydetme/yükleme:

```
numpy.save()/numpy.load()
```

Scipy.linalg

- Scipy.linalg oldukça optimize edilmiş olan ATLAS LAPACK ve BLAS kütüphalerine dayanmaktadır.
- Numpy'nin Doğrusal Cebir fonksiyonları bulumaktadır.
- Ancak, Scipy Numpy'nin içerdiği tüm Doğrusal Cebir fonksiyonlarını içerdiği gibi ilave olarak birçok fonksiyon da sunmaktadır.
- Scipy Doğrusal Cebir kütüphanesi her zaman LAPACK ve BLAS'a dayalı olarak derlendiğinden genel olarak Numpy.linalg kütüphanesine göre daha hızlıdır.
- Bu nedenle doğrusal cebir işlemlerinde Scipy.linalg kütüphanesi, numpy.linalg kütüphanesine tercih edilir.

Scipy.linalg

- Scipy.linalg Doğrusal Cebir için oldukça kapsamlı fonksiyonlar sunar.
- Kullanımı ile ilgili örnek vermek gerekirse:

```
from scipy import linalg
arr = np.array([[1, 2],[3, 4]])
linalg.det(arr)
-2.0

iarr = linalg.inv(arr)
iarr
array([[-2. ,  1. ],
       [ 1.5, -0.5]])
```

Scipy.linalg.det

$$|\mathbf{A}| = \sum_j (-1)^{i+j} a_{ij} M_{ij}.$$

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 1 \\ 2 & 3 & 8 \end{bmatrix}$$

$$|\mathbf{A}| = 1 \begin{vmatrix} 5 & 1 \\ 3 & 8 \end{vmatrix} - 3 \begin{vmatrix} 2 & 1 \\ 2 & 8 \end{vmatrix} + 5 \begin{vmatrix} 2 & \\ 2 & 3 \end{vmatrix}$$
$$= 1\,(5 \cdot 8 - 3 \cdot 1) - 3\,(2 \cdot 8 - 2 \cdot 1) + 5\,(2 \cdot 3 - 2 \cdot 5) = -25.$$

```
import numpy as np
from scipy import linalg
arr = np.array([[1, 3,5],[2,5,1],[2,3,8]])
linalg.det(arr)
-25.0
```

Scipy.linalg.inv

$$\mathbf{A} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 1 \\ 2 & 3 & 8 \end{bmatrix} \qquad \mathbf{A}^{-1} = \frac{1}{25}\begin{bmatrix} -37 & 9 & 22 \\ 14 & 2 & -9 \\ 4 & -3 & 1 \end{bmatrix} = \begin{bmatrix} -1.48 & 0.36 & 0.88 \\ 0.56 & 0.08 & -0.36 \\ 0.16 & -0.12 & 0.04 \end{bmatrix}$$

```
import numpy as np
from scipy import linalg
arr = np.array([[1, 3,5],[2,5,1],[2,3,8]])
iarr = linalg.inv(arr)
iarr
array([[-1.48 , 0.36, 0.88 ],
        [ 0.56, 0.08, -0.36])
        [ 0.16, -0.12, 0.04]])
```

Scipy.linalg

- Scipy.linalg Doğrusal Cebir için oldukça kapsamlı fonksiyonlar sunar:

```
arr = np.arange(9).reshape((3, 3)) + np.diag([1, 0, 1])
uarr, spec, vharr = linalg.svd(arr)
```

- Burada, sadece SVD, QR ve Eigen Decomposition açıklanmakla birlikte, bunların dışında;
  - LU
  - Cholesky
  - Schur matris faktörizasyon yöntemleri de scipy.linalg kütüphanesi içinde bulunmaktadır.

Scipy.linalg

$$\begin{aligned} x + 3y + 5z &= 10 \\ 2x + 5y + z &= 8 \\ 2x + 3y + 8z &= 3 \end{aligned}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 1 \\ 2 & 3 & 8 \end{bmatrix}^{-1} \begin{bmatrix} 10 \\ 8 \\ 3 \end{bmatrix} = \frac{1}{25} \begin{bmatrix} -232 \\ 129 \\ 19 \end{bmatrix} = \begin{bmatrix} -9.28 \\ 5.16 \\ 0.76 \end{bmatrix}$$

- Yukarıdaki doğrusal denklem sistemini scipy.linalg ile çözmek istersek:

```
import numpy as np
from scipy import linalg
A = np.array([[1, 3,5],[2,5,1],[2,3,8]])
b = np.array([[10], [8],[3]])
linalg.inv(A).dot(b)
array([[-9.28], [5.16],[ 0.76]])
```

Scipy.linalg

$$\begin{aligned} x + 3y + 5z &= 10 \\ 2x + 5y + z &= 8 \\ 2x + 3y + 8z &= 3 \end{aligned}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 5 & 1 \\ 2 & 3 & 8 \end{bmatrix}^{-1} \begin{bmatrix} 10 \\ 8 \\ 3 \end{bmatrix} = \frac{1}{25} \begin{bmatrix} -232 \\ 129 \\ 19 \end{bmatrix} = \begin{bmatrix} -9.28 \\ 5.16 \\ 0.76 \end{bmatrix}$$

- veya daha hızlı bir çözüm için "solve" fonksiyonu tercih edilmelidir

```
import numpy as np
from scipy import linalg
A = np.array([[1, 3,5],[2,5,1],[2,3,8]])
b = np.array([[10], [8],[3]])
np.linalg.solve(A, b)
array([[-9.28], [5.16],[ 0.76]])
```

Scipy.linalg Matris/Vektör Normları

$$\|\mathbf{x}\| = \begin{cases} \max|x_i| & \text{ord} = \inf \\ \min|x_i| & \text{ord} = -\inf \\ \left(\sum_i |x_i|^{\text{ord}}\right)^{1/\text{ord}} & |\text{ord}| < \infty. \end{cases}$$

$$\|\mathbf{A}\| = \begin{cases} \max_i \sum_j |a_{ij}| & \text{ord} = \inf \\ \min_i \sum_j |a_{ij}| & \text{ord} = -\inf \\ \max_j \sum_i |a_{ij}| & \text{ord} = 1 \\ \min_j \sum_i |a_{ij}| & \text{ord} = -1 \\ \max \sigma_i & \text{ord} = 2 \\ \min \sigma_i & \text{ord} = -2 \\ \sqrt{\text{trace}\left(\mathbf{A}^H\mathbf{A}\right)} & \text{ord} = \text{'fro'} \end{cases}$$

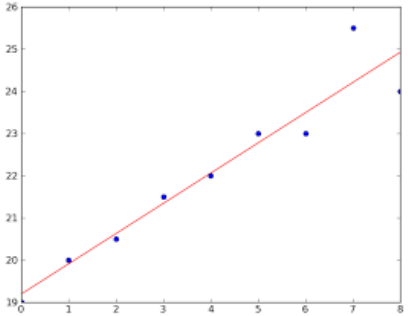
```
import numpy as np
from scipy import linalg
A=np.array([[1,2],[3,4]])
linalg.norm(A) →5.4772255750516612
linalg.norm(A,'fro') →5.4772255750516612
linalg.norm(A,1) → 6
linalg.norm(A,-1) → 4
linalg.norm(A,np.inf) → 7
```

Scipy.linalg En Küçük Kareler Çözümü

$$\|(w_1 x_i + w_2) - y_i\|^2$$

$$\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ . & 1 \\ . & 1 \\ . & 1 \\ x_n & 1 \end{pmatrix} \mathbf{w} = \begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{pmatrix}$$

```
from numpy import arange,array,ones,linalg
xi = arange(0,9)
A = array([ xi, ones(9)])
y = [19, 20, 20.5, 21.5, 22, 23, 23, 25.5, 24]
w = linalg.lstsq(A.T,y)[0] → doğrunun parametreleri
## w, resid, rank, sigma = linalg.lstsq(....)
```



Scipy.linalg Genelleştirilmiş Matris Tersi

- Scipy.linalg kütüphanesi içinde genelleştirilmiş matris tersi olarak Moore-Penrose tersi kullanılmaktadır.
- Moore-Penrose tersi için iki adet fonksiyon vardır:
  - pinv ve pinv2
- Bunlardan pinv2 Tekil Değer Ayrıştırılması (Singular Value Decomposition) kullanarak genelleştirilmiş matrisi tersini bulmaktadır.

```
A = floor(random.rand(4,4)*20-10)
b = floor(random.rand(4,1)*20-10) → Ax=b
pinv = linalg.pinv(A)
xPinv = dot(pinv,b)
```

Scipy.linalg Tekil Değer Ayrıştırılması (SVD)

- Tekil Değer Ayrıştırılması (Singular Value Decomposition/SVD) da Doğrusal Cebirdeki güçlü yöntemlerden biridir.
- SVD kullanılarak Moore-Penrose tersi alınabilir.
- Önceki En Küçük Kareler problemi SVD ile çözülmek istenirse:

```
A = floor(random.rand(4,4)*20-10)
b = floor(random.rand(4,1)*20-10) → Ax=b
U,s,V = linalg.svd(A) # A'nın SVD Ayrıştırılması
pinv_svd = dot(dot(V.T,linalg.inv(diag(s))),U.T)
```

Scipy.linalg QR Faktörizasyonu

- QR Matrix Ayrıştırması (Factorization) da doğrusal denklem sistemlerinin çözümü için güçlü bir yöntemdir.
- Önceki En Küçük Kareler problemi QR ile çözülmek istenirse:

$$\|Ax - b\|_2 \qquad QRx = b \qquad Q^T QRx = Q^T b \qquad Rx = Q^T b$$

```
A = random.rand(5,3)
b = random.rand(5,1)
Q,R = linalg.qr(A)
Qb = dot(Q.T,b)
x_qr = linalg.solve(R,Qb)
```

Scipy.linalg Özdeğer/Özvektör

- Eigenvalue Decomposition da diğer bir Matrix Ayrıştırması (Factorization) yöntemidir.

```
>>> import numpy as np
>>> from scipy import linalg
>>> A = np.array([[1, 2], [3, 4]])
>>> la, v = linalg.eig(A)
>>> l1, l2 = la
>>> print(l1, l2)   # özdeğerler
(-0.372281323269+0j) (5.37228132327+0j)
>>> print v[:, 0]   # ilk özvektör
[-0.82456484  0.56576746]
```

Scipy.fftpack

- Diğer önemli bir Scipy modülü ise Fast Fourier Dönüşümü (FFT) kütüphanesidir:

$$y[k] = \sum_{n=0}^{N-1} e^{-2\pi j \frac{kn}{N}} x[n] \qquad x[n] = \frac{1}{N} \sum_{n=0}^{N-1} e^{2\pi j \frac{kn}{N}} y[k]$$

```
>>> from scipy.fftpack import fft, ifft
>>> x = np.array([1.0, 2.0, 1.0, -1.0, 1.5])
>>> y = fft(x)
>>> y
array([ 4.50000000+0.j        ,  2.08155948-1.65109876j,
       -1.83155948+1.60822041j, -1.83155948-1.60822041j,
        2.08155948+1.65109876j])
>>> yinv = ifft(y)
>>> yinv
array([ 1.0+0.j,  2.0+0.j,  1.0+0.j, -1.0+0.j,  1.5+0.j])
```

Scipy.fftpack

- Scipy.fftpack, Fast Fourier Dönüşümü için gerekli diğer fonksiyonları da sunar:

```
from scipy import fftpack
time_step = 0.02
period = 5.
time_vec = np.arange(0, 20, time_step)
sig = np.sin(2 * np.pi / period * time_vec) + \
        0.5 * np.random.randn(time_vec.size)
sample_freq = fftpack.fftfreq(sig.size, d=time_step)
sig_fft = fftpack.fft(sig)
```

Scipy.optimize

- Scipy.optimize BFGS gibi yerel optimizasyon fonksiyonları sunar:

```
from scipy import optimize
def f(x):
    return x**2 + 10*np.sin(x)

optimize.fmin_bfgs(f, 0)
Optimization terminated successfully.
         Current function value: -7.945823
         Iterations: 5
         Function evaluations: 24
         Gradient evaluations: 8
array([-1.30644003])
```

Scipy.optimize

- Scipy.optimize bazı global optimizasyon araçları da içermektedir:

```
optimize.basinhopping(f, 0)
                nfev: 1725
 minimization_failures: 0
                 fun: -7.9458233756152845
                   x: array([-1.30644001])
             message: ['requested number of basinhopping
iterations completed successfully']
                njev: 575
                 nit: 100
```

Scipy.interpolate

- Aradeğerleme için Scipy.interpolate kütüphanesi kullanılabilir:

```
from scipy.interpolate import interp1d
measured_time = np.linspace(0, 1, 10)
noise = (np.random.random(10)*2 - 1) * 1e-1
measures = np.sin(2 * np.pi * measured_time) + noise
linear_interp = interp1d(measured_time, measures)
```

Scipy.interpolate

- Aradeğerleme «Doğrusal» yerine «Kübik» olarak da yapılabilir:

```
from scipy.interpolate import interp1d
measured_time = np.linspace(0, 1, 10)
noise = (np.random.random(10)*2 - 1) * 1e-1
measures = np.sin(2 * np.pi * measured_time) + noise
cubic_interp = interp1d(measured_time, measures,
kind='cubic')
cubic_results = cubic_interp(computed_time)
```

Kaynakça

1 Wentworth, P., Elkner, J., Downey, A.B., Meyers, C. (2014). How to Think Like a Computer Scientist: Learning with Python (3nd edition).
2 Pilgrim, M. (2014). Dive into Python 3 by. Free online version: DiveIntoPython3.org ISBN: 978-1430224150.
3 Summerfield, M. (2014) Programming in Python 3 2nd ed (PIP3) : - Addison Wesley ISBN: 0-321-68056-1.
4 Jones E, Oliphant E, Peterson P, et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/.
5 Millman, K.J., Aivazis, M. (2011). Python for Scientists and Engineers, Computing in Science & Engineering, 13, 9-12.
6 John D. Hunter (2007). Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, 9, 90-95.
7 Travis E. Oliphant (2007). Python for Scientific Computing, Computing in Science & Engineering, 9, 10-20.
8 Goodrich, M.T., Tamassia, R., Goldwasser, M.H. (2013). Data Structures and Algorithms in Python, Wiley.
9 http://www.diveintopython.net/
10 https://docs.python.org/3/tutorial/
11 http://www.python-course.eu
12 https://developers.google.com/edu/python/
13 http://learnpythonthehardway.org/book/