SciPy 치트시트

1. SciPy란?

SciPy는 과학 및 공학 계산을 위한 Python 생태계의 핵심 라이브러리 중 하나입니다. NumPy를 기반으로 하며, 최적화, 선형대수, 적분, 보간, 특수 함수, FFT, 신호 및 이미지 처리, 상미분 방정식 풀이 등 다양한 고급 과학 계산 기능을 제공합니다.

2. 주요 서브패키지

- `scipy.stats`: 통계 함수 및 확률 분포
- `scipy.optimize`: 최적화 알고리즘 (최소화, 커브 피팅 등)
- `scipy.interpolate`: 보간법
- `scipy.linalg`: 선형대수 루틴
- `scipy.integrate`: 수치 적분
- `scipy.fft`: 고속 푸리에 변환
- `scipy.signal`: 신호 처리
- `scipy.sparse`: 희소 행렬

3. 통계 (`scipy.stats`)

기술 통계, 확률 분포, 통계 검정 등 광범위한 기능을 제공합니다.

- 기술 통계:

```
from scipy import stats
import numpy as np
data = np.array([1, 2, 3, 4, 5])
stats.describe(data) # 개수, 최소/최대, 평균, 분산, 왜도, 첨도
```

- T-검정 (T-test):

```
# 독립 표본 t-검정
group1 = stats.norm.rvs(loc=5, scale=10, size=500)
group2 = stats.norm.rvs(loc=5, scale=10, size=500)
stats.ttest_ind(group1, group2)
```

- 정규성 검정 (Normality Test): `stats.shapiro(data)`
- 확률 분포:
  - `stats.norm`: 정규 분포
  - `stats.uniform`: 균등 분포
  - `stats.binom`: 이항 분포
  - `.rvs()`: 랜덤 표본 생성
  - `.pdf()`: 확률 밀도 함수
  - `.cdf()`: 누적 분포 함수
  - `.ppf()`: 누적 분포 함수의 역함수 (백분위수)

4. 최적화 (`scipy.optimize`)

함수의 최솟값을 찾거나, 방정식의 근을 구하거나, 커브 피팅을 수행합니다.

- 함수 최소화:

```
from scipy.optimize import minimize
def rosen(x): # 로젠브록 함수
    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')
print(res.x) # 최솟값을 만드는 x
```

- 커브 피팅 (Curve Fitting):

```
from scipy.optimize import curve_fit
def func(x, a, b, c):
    return a * np.exp(-b * x) + c
xdata = np.linspace(0, 4, 50)
ydata = func(xdata, 2.5, 1.3, 0.5) + 0.2 *
np.random.normal(size=len(xdata))
popt, pcov = curve_fit(func, xdata, ydata)
# popt: 최적의 매개변수 [a, b, c]
```

5. 선형대수 (`scipy.linalg`)

NumPy의 `np.linalg`보다 더 많은 고급 기능을 제공합니다.

- 역행렬: `linalg.inv(A)`
- 행렬식: `linalg.det(A)`
- 특이값 분해 (SVD): `linalg.svd(A)`
- 고유값 문제: `linalg.eig(A)`
- 최소 자승 문제: `linalg.lstsq(A, b)`

6. 수치 적분 (`scipy.integrate`)

- 함수 적분: `integrate.quad(lambda x: x**2, 0, 4)`
- 상미분 방정식(ODE) 풀이: `integrate.solve_ivp(fun, t_span, y0)`

7. 보간법 (`scipy.interpolate`)

알려진 데이터 포인트를 기반으로 새로운 데이터 포인트를 추정합니다.

```
from scipy.interpolate import interp1d
x = np.linspace(0, 10, num=11)
y = np.cos(-x**2/9.0)
f = interp1d(x, y, kind='cubic') # 3차 스플라인 보간
x_new = np.linspace(0, 10, num=41)
y_new = f(x_new)
```