SciPy 핵심 치트시트

SciPy 개요

SciPy는 과학적, 공학적 수치 계산을 위해 설계된 파이썬 생태계의 핵심 라이브러리입니다. NumPy를 기반으로 구축되었으며, 최적화, 선형대수, 적분, 보간, 특수 함수, FFT, 신호 및 이미지 처리, 상미분 방정식 풀이 등 다양한 고수준의 알고리즘과 도구를 제공합니다.

주요 서브패키지 구성

- `scipy.stats`: 통계 함수 및 각종 확률 분포
- `scipy.optimize`: 최적화 알고리즘 (함수 최소화, 커브 피팅 등)
- `scipy.interpolate`: 데이터 보간법
- `scipy.linalg`: NumPy보다 정밀한 고급 선형대수 루틴
- `scipy.integrate`: 수치 적분 및 상미분 방정식 해결
- `scipy.fft`: 고속 푸리에 변환
- `scipy.signal`: 신호 처리 및 필터링
- `scipy.sparse`: 대규모 희소 행렬 처리

통계 분석 (scipy.stats)

기술 통계부터 확률 분포, 가설 검정까지 폭넓은 통계 기능을 지원합니다.

- **기초 통계 요약**:

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

- **T-검정 (T-test)**:

```
# 두 집단 간 평균의 차이가 유의미한지 검정합니다 (독립 표본).
group1 = stats.norm.rvs(loc=5, scale=10, size=500)
group2 = stats.norm.rvs(loc=5, scale=10, size=500)
result = stats.ttest_ind(group1, group2)
```

- **주요 확률 분포**:
  - `stats.norm`: 정규 분포
  - `stats.binom`: 이항 분포
  - `.rvs()`: 랜덤 표본(Random Variates) 생성
  - `.pdf()` / `.pmf()`: 확률 밀도/질량 함수 계산
  - `.cdf()`: 누적 분포 함수 계산
  - `.ppf()`: 누적 분포 함수의 역함수 (백분위수 계산 시 유용)

수치 최적화 (scipy.optimize)

함수의 최솟값을 탐색하거나, 방정식의 해를 구하고 데이터를 특정 곡선에 맞추는 작업을 수행합니다.

- **함수 최솟값 찾기 (Minimize)**:

```
from scipy.optimize import minimize
def rosen(x): # 로젠브록(Rosenbrock) 함수 정의
    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])
# Nelder-Mead 알고리즘을 사용하여 최적의 x 값을 탐색합니다.
res = minimize(rosen, x0, method='nelder-mead')
```

- **커브 피팅 (Curve Fitting)**:

```
from scipy.optimize import curve_fit
def model_func(x, a, b, c):
    return a * np.exp(-b * x) + c
# 관측 데이터를 바탕으로 모델 파라미터 a, b, c의 최적값을 추정합니다.
popt, pcov = curve_fit(model_func, xdata, ydata)
```

고급 선형대수 (scipy.linalg)

표준 선형대수 연산을 더욱 빠르고 정밀하게 처리합니다.

- **역행렬**: `linalg.inv(A)`
- **행렬식**: `linalg.det(A)`
- **특이값 분해 (SVD)**: `linalg.svd(A)`
- **고유값 및 고유벡터**: `linalg.eig(A)`
- **최소 자승법**: `linalg.lstsq(A, b)`

수치 적분 (scipy.integrate)

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

데이터 보간법 (scipy.interpolate)

산재된 데이터 포인트 사이의 값을 추정하여 부드러운 곡선을 만듭니다.

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