YAPAY ZEKA VE PYTHON PROGRAMLAMA

SciPy
Emir Öztürk

SCI-PY

- Matematiksel, bilimsel ve mühendislik işlemleri için bir kütüphane
- Numpy + Scipy library + Matplotlib + IPython + SymPy + pandas
- pip3 install numpy scipy matplotlib ipython jupyter pandas sympy

Sci-py kütüphanesi birden fazla paketin bir araya getirilmesi ile oluşur. Paketin amacı bilimsel hesaplama ve çözümlerin yapılabilmesidir.
Sci-py paketini yüklemek için pip paket yöneticisi ile slaytta verilen paketler yüklenebilir.

SABİTLER

- from scipy import constants
- print(dir(constants))
  - Metric (SI)
  - Binary
  - Kütle
  - Açı
  - Zaman
  - Uzunluk
  - Basınç
  - Alan
  - Hacim
  - ...

https://www.w3schools.com/python/scipy_constants.asp

sci-py kütüphanesi altında bilimsel hesaplamalar için farklı kategorilerde sabitler bulunmaktadır. Bu sabitlerin ekrana gösterilmesi dir ile yapılabilir. Sabitlerin kullanılması için constants.sabitadı kullanılabilir.

KÖK BULMA

- from scipy.optimize import root
  - def esitlik(x):
    - return x + cos(x)
  - root(esitlik,0)

Herhangi bir fonksiyonun kökünü bulabilmek adına root metodu kullanılabilmektedir. Bir fonksiyon tanımı yapıldıktan sonra root fonksiyonu ilk tahmin ile (ikinci parametre) çalıştırılır.

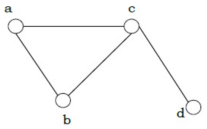
MINIMA - MAXIMA

- minimize()
- Metotlar
  - CG, BFGS, Newton-CG, L-BFGS-B, TNC, COBYLA, SLSQP
- from scipy.optimize import minimize
- def esitlik(x):
  - return x**2 + x + 2
  - minimum = minimize(esitlik,tahminikok,method='BFGS')

bir metodun minimum, maximum değerlerini belirli yaklaşım yöntemleri ile bulma işlemlerini gerçekleştirmek için minimze ve maximize kullanılabilir. Bu metotların yaklaşım yöntemleri parametre olarak verilebilir. Seçilen bir iterasyon yöntemi ile maksimum veya minimum değer yakalanmaya çalışılır.

GRAFLAR

- Bitişiklik matrisi
- Graf düğümleri satır ve sütunlar
- Graf kenarları değerler
- $\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

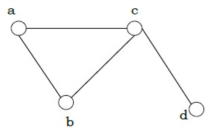


Graflar düğümler ve bu düğümlerin arasındaki bağlantılardan oluşur. Düğüm ve bağlantıları (vortex - edge) python içerisinde tanımlamak için bitişiklik matrisi kullanılabilir. Bu matrislerde satır ve sütunlar graftaki düğümleri ifade ederken, değerler ise bu düğümlerin arasında bağlantı olup olmadığını ve ağırlıklı graflar için ağırlık değerini verir. Daha sonra bu matris belirli sezgisel yöntemlerle gezilebilmektedir.

GRAFLAR – BFS – DFS

- depth_first_order(csr matrix,başlangıç elemanı)
- breadth_first_order(csr matrix,başlangıç elemanı)

```
from scipy.sparse.csgraph import depth_first_order
from scipy.sparse import csr_matrix
import numpy as np
arr = np.array([
 [0, 1, 1, 0],
 [1, 0, 1, 0],
 [1, 1, 0, 1],
 [0, 0, 1, 0]])
graph = csr_matrix(arr)
print(graph)
x,y = depth_first_order(graph,0, return_predecessors=True)
print(x)
```



Grafları gezmek için derinlik öncelikli (depth first) veya genişlik öncelikli (breadth first) yaklaşımlar kullanılabilir. Başlangıç ve bitiş düğümleri belirlendikten sonra aradaki yolu tespit etmek için bu yöntemler kullanılabilir. Ayrıca en kısa yolu bulmak için dijkstra algoritması da aynı paket içerisinde mevcuttur.

INTERPOLATION

- İki aralık arası sayı oluşturma
- from scipy.interpolate import interp1d
- X değerleri ve Y değerleri
- np.arange()
- interpolasyonFonksiyonu = interp1d(x,y)
- yeniXler = np.arange(0,9,0.1)
- yeniYler = interpolasyonFonksiyonu(yeniXler)

İnterpolasyon yöntemleri de sci-py içerisinde mevcuttur. Önceden verilen x ve y değerleri için bir interpolasyon fonksiyonu tanımlanır. Daha sonra bu fonksiyona yeni x değerleri verildiğinde buna uygun y değerlerini üretir.

SYMPY

- Sembolik çözüm
  - Mathematica
  - Maple
  - Matlab

https://scipy-lectures.org/packages/sympy.html

Sympy ile sembolik çözüm yapılabilmektedir. Mathematica, Maple, Matlab ile de kullanılabilen bu yönteme bir python alternatifi sunmaktadır. Amaç, değişkene sahip denklemlerin değer verilmeden kullanılabilmesi, sadeleştirilmesi, çözümlenmesi, türev veya integral uygulanması gibi işlemlerin gerçekleştirilmesidir.

SYMPY - SEMBOLLER

- import sympy as sym
- x = sym.Symbol('x')
- y = sym.Symbol('y')
- x + y + x - y ?
- (x + y) ** 2 ?

Sembol tanımı yapmak için sym.Symbol kullanılabilmektedir. Daha sonra python dilinde yazılan ifadeler sembolik olarak yazdırılabilirler.

SYMPY - GENİŞLETME VE BASİTLEŞTİRME

- sym.expand((x+y) ** 2)
- sym.expand((x+y)**3)
- sym.simplify((x + x*y) / x)

Bu formüllerin zaman zaman genişletme veya sadeleştirme işlemlerine tabi tutulması gerekmektedir. Genişletme için expand, sadeleştirme için simplify kullanılabilir.

SYMPY - LIMIT

- sym.limit(sym.sin(x) / x, x, 0) ?
- sym.limit(1 / x, x, sym.00) ?
- sym.limit(x ** x, x ,0) ?

Sembolik metot üzerinde limit işlemi

DIFFERENTIATION

- sym.diff(sym.sin(x),x) ?
- sym.diff(sym.sin(2*x),x) ?
- sym.diff(sym.tan(x), x)
- Türev
- sym.diff(sym.sin(2*x),x,1)
- sym.diff(sym.sin(2*x),x,2)
- sym.diff(sym.sin(2*x),x,3)

Sembolik metot üzerinde türev işlemi

INTEGRATION

- sym.integrate(6 * x ** 5,x) ?
- sym.integrate(sym.sin(x),x) ?
- Sınırlı
- sym.integrate(x**3, (x,-1,1))
- sym.integrate(sym.sin(x),(x, 0,sym.pi /2))

Sembolik metot üzerinde integrasyon. İkinci parametre olarak hangi değişkene göre integral alınacağı belirtilir. Eğer ikinci parametre parantez içerisinde sınır değerleri tanımlanarak verilirse sınırlı integral almak da mümkündür.

PANDAS

- Veri analizi
- Veri düzenleme
- Verileri yeniden boyutlandırma
- Veri setlerinin birleştirilmesi
- Zaman serileri ile ilgili hesaplar