Introducción a NumPy

NumPy Introduction

Autor: juan Sebastián palacio villada y verónica palacio villada

*IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: [v.palacio@utp.edu.co](mailto:v.palacio@utp.edu.co) juansebastian.palacio@utp.edu.co

**Introducción** – en la ingeniería de sistemas y en el mundo python se ha caracterizado por ser uno de los lenguajes de programación más flexibles, utilizados y aceptados, incluso en muchas instituciones de aprendizaje informático y de todo tipo esta es la herramienta que más utilizan, esto se da a que es un lenguaje de programación sencillo de alto nivel, y es un lenguaje que no cuesta mucho tiempo aprenderlo, ya que tiene herramientas muy sencillas y esto hace que su curva de aprendizaje sea tan atractiva, python utiliza una única sintaxis de programación añadiendo a ello que podemos agregar muchas librerías que nos ayudan a desarrollar no solo funciones simples sino también complejas como programar funciones matemáticas de alto nivel . hablemos un poco sobre numpy que es una librería que le ayuda al programador a hacer funciones matemáticas complejas haciendo asi mas facil programar, algunas de las cosas que nos permite esta librería es crear, modificar, y hacer operaciones sobre ellas .

**Conceptos básicos de NumPy**

numpy es una librería de python que se encarga de hacer cálculos numéricos, para hacer uso de esta librería numpy debemos importar la librería que es llamada NumPy para que el editor pueda entender que estamos haciendo funciones de la misma

import NumPy as np

este es el reflejo de lo que vemos a la hora de importar dicha está, a la derecha de numpy podemos observar que lo acompañan dos palabras que son completamente desconocidas para nosotros y para el público, estas simplemente están abreviando la palabra numpy, esto se utiliza para que en algún momento no tengamos que llamar la función directamente por su nombre si no que podamos hacerlo un poco más rápido ya que no se tendría que escribir la palabra completa NumPy si no que escribimos simplemente las letras “np”

al explicar este concepto ya se puede empezar a crear un vector (array o arreglo) en el cual se va a poder trabajar de una buena manera, eficaz, rápida y confiable por la ayuda de esta librería

**cómo creamos un vector con la librería numpy**

primero se debe importar la librería numpy esta es la que nos da todas la lista de python

import NumPy as np que de esta habiamos hablado en los conceptos básicos de Numpy.

a continuación haremos un vector con 6 elementos

a = np.array([0,1,2,3,4,5])

el cual inicia en 0 y termina en 6 esto lo guardamos en la variable (a) y lo imprimimos en pantalla con la función

print(a, '\n')

con numpy también podemos saber el número de dimensiones que tiene el array y los elementos que lo componen

print(a.ndim '\n'), print(a.shape)

cuando pasamos a imprimir en pantalla lo que vemos es algo muy simple que sería

[0 1 2 3 4 5 ]

1

(6)

el siguiente paso a seguir es utilizar la función reshape para modificar la estructura del array la cual inicialmente tenía una dirección y ahora tiene dos, esto genera una estructura con dos dimensiones y tres elementos, luego se procede a mostrar el array con el número de elementos y dimensiones del vector (b), con shape y ndim

con numpy se puede convertir un vector en una matriz con funciones de manera simple, y rellenar los arreglo de uno por uno y logramos modificar la forma de dicho vector para hacer estos cambios se utiliza la función

b = a.reshape((3,2))

print(b, '\n')

Para saber cuántas dimensiones del tiene el arreglo utilizamos la siguiente función

print(b.ndim, '\n')

print(b.shape)

también podemos modificar en qué fila y columna podemos editar el número que lo contiene siendo el primer número la columna y el segundo la fila con la siguiente fórmula siendo (b) el arreglo que deseemos editar

b[1][0] = 77

para conservar el array o vector original que es el que anteriormente se ha mostrado se utiliza la siguiente función

c = a.reshape((3,2)).copy().

con esta función logramos hacer una reestructuración dela array , y se crea una copia del mismo, con una array definido numpy nos permite hacer iteración de los elementos que pertenecen a él, esto se hace con la función

print(a>4, '\n')

**Proyecto de Machine Learning**

Una empresa vende el servicio de proporcionar algoritmos de aprendizaje automático a través de HTTP. Con el éxito creciente de la empresa, aumenta la demanda de una mejor infraestructura para atender todas las solicitudes web entrantes. No queremos asignar demasiados recursos, ya que sería demasiado costoso. Por otro lado, perderemos dinero si no hemos reservado suficientes recursos para atender todas las solicitudes entrantes. Ahora, la pregunta es, ¿cuándo alcanzaremos el límite de nuestra infraestructura actual, que se estima en 100.000 solicitudes por hora? Nos gustaría saberlo de antemano cuando tenemos que solicitar servidores adicionales en la nube para atender todas las solicitudes con éxito sin pagar por las no utilizadas.

Paquete de datos a ser procesados:

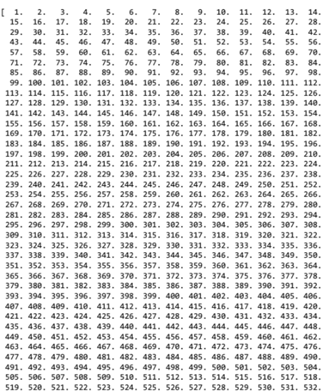
1. Columna: Número de horas
2. Columna: Número tareas ejecutadas

Se asigna a la variable de los datos del archivo que tiene el nombre de “web\_traffic.tsv” este nos muestra en la pantalla los 10 valores que contiene el archivo con la siguiente función :

data=np.genfromtxt("web\_traffic.tsv",delimiter="\t")

print(data[:10], '\n')

Nos muestra los datos de la siguiente manera:

print(data.shape)

[[1.000e+00 2.272e+03]

[2.000e+00 nan]

[3.000e+00 1.386e+03]

[4.000e+00 1.365e+03]

(743, 2)

tendremos que dividir el vectores en dos columnas, con la siguiente función:

x = data[:,0]

y = data[:,1]

Mostramos los valores en x, y los cuales se generan de la siguiente manera:

En esta pasó lo que se realiza es la muestra de la dimensión de los vectores x, y de la siguiente manera:

print(x.ndim, '\n')

print(y.ndim, '\n')

Verificamos los elementos que contienen los vectores x, y este nos muestra los siguientes datos:

(743 , 1)

(743 , 1)

En la siguiente línea se cuentan el número de valores “nan” que contiene el vector “y”, el cual se verifica con la siguiente función:

print(np.sum(np.isnan(y)))

la cual nos imprime en pantalla:

8

en la siguiente línea podemos ver el número de elementos que contiene (x, y) antes de que puedan ser comprimidos, con la siguiente función:

print(x.shape, '\n')

print(y.shape, '\n')

al imprimilo nos muestra los siguientes valores:

(743,)

(743,)

ahora lo que realizaremos es la eliminación de los elementos “nan” tanto para x como para y, con la siguiente función:

x = x[~np.isnan(y)]

y = y[~np.isnan(y)]

print(x.shape, '\n')

print(x.shape, '\n')

al imprimilo nos muestra los siguientes valores:

(735)

(735)

En este último paso, lo que realizamos es la importación de la librería para graficar:

import matplotlib.pyplot as plt

lo primero es realizar un dibujo de el punto (x, y) con círculos de tamaño 10, con la siguiente función:

plt.scatter(x, y, s=10)

Colocamos la siguiente función para que salga un título en el gráfico en la parte superior:

plt.title("Trafico Web del último mes")

En esta con esta función se le coloca otro título en la parte inferior del gráfica:

plt.xlabel("Tiempo")

con esta función se coloca otro título pero en la parte central izquierda de la gráfica:

plt.ylabel("Solicitudes/Hora")

Estas funciones que se muestran agrupan los valores por semana ya que los datos lo tenemos en horas pasaremos los mismos en semanas (7 \* 24 ) qué es: los 7 días de la semana y las 24 horas que tiene un día, con un rango de 10 se pone a iterar w y se le asigna valores a w (1, 2, 3, 4,....) y así representa en la gráfica con puntos como evoluciona los días si aumenta o disminuye.

plt.xticks([w\*7\*24 for w in range(10)],

['semana %i' % w for w in range(10)])

plt.autoscale(tight=True)

Por último se dibuja una cuadrícula con su color asignado para este ejercicio y ejecutamos el código con la última función.

plt.grid(True,linestyle='-',color='0.75')

plt.show()

este gráfico nos muestra cómo incrementa por semanas el tráfico web

