Computación Blanda

# Aplicación Numpy a Machine Learning

Autor: Maria Camila Muñoz Mejía

Genny Paola Rivera Becerra

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

Correo-e: [m.munoz1@utp.edu.co](mailto:m.munoz1@utp.edu.co)

Genny.rivera@utp.edu.co

***Resumen*—** En este documento se brinda una explicación sobre el código trabajado en clase, este código permite entender y comprender cada línea de dicho algoritmo ya que cada una está explicada de tal manera que sea fácil para dicha comprensión, se explica de manera detallada que función realiza cada comando.

***Palabras clave—*** línea, código, compilar, ejecutar, números, comando, datos, estructura, función, imprimir, pantalla.

***Abstract*—** This document provides an explanation of the code worked in class, this code allows to understand and understand each line of said algorithm since each one is explained in such a way that it is easy for said understanding, it is explained in detail what function each performs command.

***KeyWord*— L**ine, code, compile, run, numbers, command, data, structure, function, to print, screen.

**INTRODUCCIÓN**

Machine learning es una derivación de inteligencia artificial que crea sistemas que aprenden de manera automatizada, es decir, identificar patrones complejos en millones de datos, predecir comportamientos todo mediante un algoritmo y que además de todo son capaces de mejorarse de manera independiente con el tiempo, por ende en este documento se dará a conocer la explicación de cada una de las líneas de la aplicación de este algoritmo.

I. APLICACIÓN NUMPY

**1.** Para importar una función en particular o el modulo completo con un nombre corto como se muestra a continuación, este nos permite trabajar con todas la funciones presentes con esta librería.

Import Numpy as np

Los datos numéricos dispuestos en una estructura similar a una matriz en Python se pueden convertir en matrices mediante el uso de la función array (). En esta línea creamos un vector (array) con sus respectivos elementos que hacen parte de sus filas y columnas.

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

La función a printf imprimirá el array definido con el carácter a.

print(a, '\n')

Esta función nos imprime el número de dimensiones que tiene el array.

print(a.ndim, '\n')

Esta función nos imprime el número de dimensiones del array en una tupla.

print(a.shape)

Después de realizar estas líneas ejecutamos el código y nos muestra lo siguiente:

[0, 1, 2, 3, 4, 5]

1

(6,)

**2.** Cambiamos la forma del array con un comando el cual lo hacemos asignando otra variable (b), Imprimimos con print la variable para poder ver el cambio de la estructura del array y lo mostramos de la siguiente manera:

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

print(b, '\n')

Verificamos lo anterior comprobando las dimensiones y dimensiones por tupla con las siguientes líneas:

print(b.ndim, '\n')

print(b.shape)

Para verificar lo anterior y comprobarlo se muestra de la siguiente manera:

[[0 1]

[2 3]

[4 5]]

2

(3,2)

**3.** Modificamos el primer elemento de la segunda fila que tenemos:

b[1][0] = 77

Verificamos el cambio y queda de la siguiente manera:

[[0 1]

[77 3]

[4 5]]

**4.** Aclaramos que debido a que el array b se construyó con base al array (a), se muestra también cambio en el vector (a) y se muestra de la siguiente manera:

[0 1 77 3 4 5]

**5.** Realizamos la copa del array para que no afecte el original que es (a), creamos una nueva variable (c), para que se cree la copia, creamos el primer valor de nuestra nueva variable el cual es -99 y queda de la siguiente manera:

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

print(c, '\n')

c[0][0] = -99

Ahora mostramos el array c modificado de la siguiente manera:

Array a, sin modificar.

[[0 1]

[77 3]

[4 5]]

[0 1 77 3 4 5]

Array c, modificado

[[-99 1]

[77 3]

[4 5]]

**6.** Se crea una nueva variable (d), en la cual las operaciones se propagar a lo largo del array, de la siguiente manera:

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

Se multiplican los elementos de la variable d por 2 de la siguiente manera:

print(d\*2, '\n')

Se imprime en pantalla y se elevan al cuadrado los elementos del array, de la siguiente manera:

print(d\*\*2)

Queda de la siguiente manera:

[2 4 6 8 10]

[1 4 9 16 25]

**7.** En este paso tenemos la nueva definición para el array a, el cual fue definido desde el principio:

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

A continuación se va a iterar sobre los elementos del array de la siguiente manera:

print(a>4, '\n')

Definimos de nuevo el array (a)

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

Esta instrucción se hace para que se ejecute la instrucción que cumple la condición de (elemento > 4), se cambia el contenido de los mayores a 4.

a[a>4] = 1

print(a, '\n')

Y los elementos que el contenido es igual a 1, reciben un nuevo valor es cual es 777, se muestra de la siguiente manera dicho proceso:

a[a==1] = 777

print(a, '\n')

Los elementos quedan de la siguiente manera:

[False False False False True]

[1 2 3 4 1]

[777 2 3 4 777]

**8.** En este paso debemos llevar un control de los valores erróneos, con las siguientes líneas:

c = np.array([1, 2, np.NAN, 3, 4])

print(c, 'n')

Verificamos si existen los valores nan

print(np.isnan(c), '\n')

Con esta línea se verifican los valores que no son nan:

print(c[~np.isnan(c)], '\n')

Se calcula el promedio de los valores que no son nan por medio del siguiente comando:

print(np.mean(c[~np.isnan(c)]))

Todo lo planteado por medio de estas líneas, se muestra de esta manera:

[1. 2. nan 3. 4.] n

[False False True False False]

[1. 2. 3. 4.]

2.5

II. PROYECTO 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

**1**. Se asigna a la variable data un archivo que se llama web\_traffic.tsv y este nos mostrara en pantalla 10 valores del archivo, de la siguiente manera:

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]

[5.000e+00 1.488e+03]

[6.000e+00 1.337e+03]

[7.000e+00 1.883e+03]

[8.000e+00 2.283e+03]

[9.000e+00 1.335e+03]

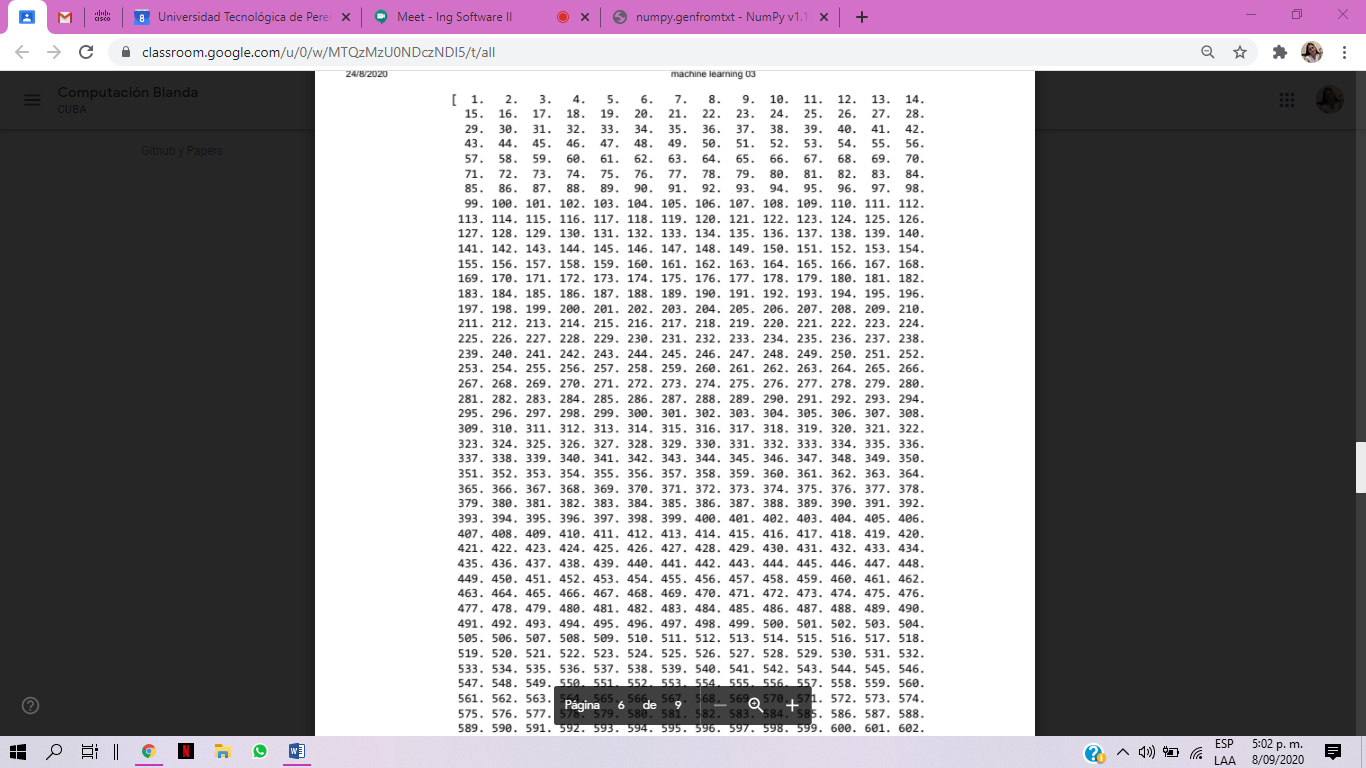
[1.000e+01 1.025e+03]]

(743, 2)

**2**. Dividimos el array en dos vectores columnas, de la siguiente manera se asignan los valores:

x = data[:,0]

y = data[:,1]

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

**3.** En esta paso 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:

1

1

(743,)

(743,)

**4.** En esta línea verificamos e investigamos el número de valores nan que contiene el vector y7, el cual se verifica de la siguiente manera:

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

Y su resultado es = 8

**5.** Con esta siguiente línea podemos ver el número de elementos en (x, y) antes de ser comprimidos, es decir que este nos permitirá ver de una manera más clara estos elementos:

print(x.shape, '\n')

print(y.shape, '\n')

Lo que realizamos es eliminar los elementos nan tanto para x como para y, con la siguiente línea:

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

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

Luego de realizar las líneas anteriores, se requiere y se debe contar con los elementos tanto de x como de y, entonces podemos visualizar lo siguiente:

print(x.shape, '\n')

print(x.shape, '\n')

Al realizar las líneas anteriores, podemos ver los comandos aplicados que nos muestran lo siguiente:

(743,)

(743,)

(735,)

(735,)

**6**. En este paso final, lo que realizamos es importar la librería para graficar:

import matplotlib.pyplot as plt

1. En este primer paso, lo que realizamos es dibujar el punto (x, y) con círculos de tamaño 10, de la siguiente manera:

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

Colocamos la siguiente línea para que salga como título en el gráfico en la parte superior:

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

En esta línea se coloca otro título pero en la parte inferior de la gráfica, de la siguiente manera:

plt.xlabel("Tiempo")

En esta línea se coloca otro título pero en la parte central izquierda de la gráfica, de la siguiente manera:

plt.ylabel("Solicitudes/Hora")

Estas líneas que se muestran a continuación agrupan los valores por semana, como muestra son 7 \* 24 que son los 7 días de la semana y las 24 horas que tiene un día, entonces en un rango de 10 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)

Como paso final dibujamos una cuadricula con su respectivo color asignado.

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

plt.show()

Y con este paso final logramos visualizar la gráfica que nos muestra como incrementa por semanas el tráfico web:

