Computación Blanda

Soft Computing

Autor: Eider De Jesus Triviño Trejos
IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: eider.trivino@utp.edu.co

***Resumen*— Este documento presenta una breve explicación e introducción a los conceptos básicos de la librería numpy hecha en python, la cual nos será muy útil a lo largo de nuestro recorrido en el aprendizaje del Machine Learning en la asignatura Computación blanda.**

Palabras clave*— *Python, Machine Learning, Código.

***Abstract*— This document presents a brief explanation and introduction to the basic concepts of the numpy library made in python, which will be very useful throughout our journey in learning Machine Learning in the Soft Computing subject.**

***Key Word*— systems, networks, artificial intelligence, software, computing, research, industry, genetic, learning.**

I. INTRODUCCIÓN

Machine Learning es una disciplina científica del ámbito de la Inteligencia Artificial que crea sistemas que aprenden automáticamente. Aprender en este contexto quiere decir identificar patrones complejos en millones de datos. La máquina que realmente aprende es un algoritmo que revisa los datos y es capaz de predecir comportamientos futuros. Automáticamente, también en este contexto, implica que estos sistemas se mejoran de forma autónoma con el tiempo, sin intervención humana. En los siguientes apartados se dará explicación del código introductorio visto en clase, Veamos cómo funciona.

```python
# Se importa la librería numpy
import numpy as np

# Se crea un vector (array) con seis elementos
a = np.array([0,1,2,3,4,5])

# Se imprime el array ... a
print(a, '\n')

# Número de dimensiones del array
print(a.ndim, '\n')

# Número de elementos del array
print(a.shape)
```

Figura 1

Primero que todo se importa la librería numpy la cual se le dará el seudónimo de np, numpy es una extensión de Python, que le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices, además se crea un vector con seis elementos para luego mostrarlos en consola junto con sus dimensiones y la cantidad de sus elementos.

```python
# Se cambia la estructura del array
b = a.reshape((3,2))
print(b, '\n')

# Se verifican los cambios
print(b.ndim, '\n')
print(b.shape)
```

Figura 2

A Continuación se cambia la estructura del array “a” con la función reshape la cual lo convierte en una matriz con la misma cantidad de elementos creados anteriormente, la nueva matriz posee dos columnas y tres filas.

```
# Se realiza una copia del array
c = a.reshape((3,2)).copy()
print(c, '\n')

# Se cambia el primer valor de c
c[0][0] = -99

# El array a no se modifica
print(a, '\n')

# El array c queda modificado
print(c)
```

Figura 3

El código mostrado en la figura 3 realiza una copia del array utilizando la función copy y cambia el valor de su primer elemento, este cambio no es efectuado en el vector "a" ya que se modificó explícitamente sólo el vector "c".

```
# Control de valores erróneos
c = np.array([1, 2, np.NAN, 3, 4])
print(c, 'n')

# Se verifica la existencia de valores nan
print(np.isnan(c), '\n')

# Se eligen todos los valores que NO son nan
print(c[~np.isnan(c)], '\n')

# Se calcula el promedio de los valores que NO son nan
print(np.mean(c[~np.isnan(c)]))
```

```
[ 1.  2. nan  3.  4.] n
[False False  True False False]

[1. 2. 3. 4.]

2.5
```

Figura 4

Para el control de valores erróneos se procede a crear un nuevo vector donde uno de sus elementos contienen un valor invalido llamado "nan", se imprime antes de su modificación para mostrar que realmente posee datos basura, después se imprimen solo los elementos que NO sean basura anteponiendo el símbolo (~) en la instrucción (np.isnan(c)) lo cual nos devuelve solos los valores no nan, Por último se promedian los datos no nan

```
# PROYECTO machine learning
Enunciado
# Enunciado: Una empresa vende el servicio de proporcionar algoritmos de apren
dizaje automático a través de HTTP.
# Con el éxito creciente de la empresa, aumenta la demanda de una mejor infrae
structura para atender todas las
# solicitudes web entrantes. No queremos asignar demasiados recursos, ya que s
ería 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 infraestruc
tura actual, que se estima en
# 100.000 solicitudes por hora?. Nos gustaría saberlo de antemano cuando tenem
os que solicitar servidores adicionales
# en la nube para atender todas las solicitudes con éxito sin pagar por las no
utilizadas.
```

Figura 5

Aplicaremos los elementos aprendidos anteriormente para aplicarlo a un problema real de Machine Learning con el pequeño ejemplo mostrado en la figura 5.

```
# Vamos a desarrollar un programa de machine learning (básico)
# El siguiente es un paquete de datos a ser procesdos:
# La primera columna es: Número de horas
# La segunda columna es: Número de tareas ejecutadas
data = np.genfromtxt("web_traffic.tsv", delimiter="\t")
print(data[:10], '\n')

# Número de datos
print(data.shape)
```

Figura 6

Se lee el archivo "web_traffic.tsv" con ayuda de la función (genfromtxt) la cual almacenará los datos en la variable llamada data, acto seguido imprime el número de datos obtenidos los cuales son 743.

```
# Se divide el array en dos vectores columnas: x, y
x = data[:,0]
y = data[:,1]

# Se muestran los valores en x, y
print(x, '\n')
print(y, '\n')
```

Figura 7

Se divide el array en dos vectores columnas llamadas x, y

```
# Dimensión de los vectores x, y
print(x.ndim, '\n')
print(y.ndim, '\n')

# Elementos contenidos en los vectores x, y
print(x.shape, '\n')
print(y.shape)
```

Figura 8

Se evidencia que ambos vectores (x, y) tienen la misma dimensión (1) la cual equivale a la de un vector, además se comprueba que la cantidad de datos sea la misma que se mostró en un principio (743)

```
# Número de elementos en x, y, antes de ser comprimidos
print(x.shape, '\n')
print(y.shape, '\n')

# Se eliminan los elementos nan tanto de x como de y
x = x[~np.isnan(y)]
y = y[~np.isnan(y)]

# Se cuenta el número de elementos tanto de x como de y
print(x.shape, '\n')
print(x.shape, '\n')
```

```
(743,)

(743,)

(735,)

(735,)
```

Figura 9

Se eliminan los datos nan de los vectores X e Y utilizando el método visto anteriormente (Figura 4), acto seguido se imprimen el número de datos en los dos vectores pero esta vez sin contenidos basura.

```
# Se importa la librería para graficar
import matplotlib.pyplot as plt

# Dibuja los puntos (x,y) con círculos de tamaño 10
plt.scatter(x, y, s=10)

# Títulos de la gráfica
plt.title("Tráfico Web en el último mes")
plt.xlabel("Tiempo")
plt.ylabel("Accesos/Hora")


plt.xticks([w*7*24 for w in range(10)],['semana %i' % w for w in range(10)])
plt.autoscale(tight=True)

# dibuja una cuadrícula punteada ligeramente opaca
plt.grid(True, linestyle='-', color='0.75')

# Muestra el gráfico
plt.show()
```

Figura 10

Se importa una nueva librería llamada Matplotlib.pyplot dándole el seudónimo de plt, esta librería posee los métodos utilizados en el lenguaje matemático de MatLab, se dibujan los puntos (X e Y) con círculos de tamaño 10, se le dan los respectivos títulos a la gráfica, se le establece un rango a la gráfica el cual será de 10 en 10 para que el gráfico no sea muy denso en su información, después de aplicar estas configuraciones se grafica.

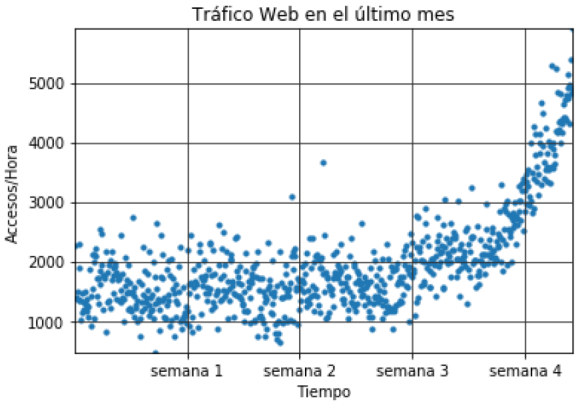

Figura 11

El gráfico obtenido representa los movimientos realizados durante un mes en el servidor, con estos datos se puede analizar si el servidor necesitará cambios favorables para su mejor rendimiento.

REFERENCIAS

Referencias en la Web:

[1] https://computerhoy.com/reportajes/tecnologia/inteligencia-artificial-469917

https://cleverdata.io/que-es-machine-learning-big-data/