Computación Blanda

Soft Computing

Autor: Marlon Deyber Restrepo Rodriguez Y Kevin Alexander Rodriguez Bedoya
IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia
Correo-e: marlon.restrepo@utp.edu.co
Kevinalexander.rodriguez@utp.edu.co

***Resumen*— Este documento presenta algunos ejemplos de los conceptos básicos de la librería numpy para aplicarse en el tema de machine learning.**

***Palabras clave*— numpy, machine learning, Manejo de datos, Graficacion de datos.**

***Abstract*— This document presents some examples of the basic concepts of the numpy library for applied in the subject of machine learning.**

***Key Word*— numpy, machine learning, data management, data graphing.**

I. INTRODUCCIÓN

El Machine Learning puede ser una herramienta increíblemente beneficiosa para descubrir información y predecir tendencias futuras.

Antes de poder tener dichas predicciones a la tendencia de los datos, es necesario acudir a algunas funcionalidades basadas en manejo de arreglos y graficas las cuales en este caso serán utilizadas a partir de la librería numpy de python.

I.1 ARRAYS

```
# 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)
```

En este apartado se define un arreglo de 6 elementos
Y se muestra en pantalla el arreglo con las siguientes propiedades del mismo:

- Número de elementos
- Numero de dimensiones

```
# 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)
```

En la variable **b** de tipo array se crea y guarda la estructura modificada del arreglo **a**, en este caso se crearon 3 filas y 2 columnas, seguidamente de ello verificamos mostrando el arreglo b en pantalla con su respectiva dimensión y numero de filas y columnas.

```
# Se modifica el primer elemento de la segunda fila
b[1][0] = 77

# Se verifica el cambio
print(b)
```

```
# Para evitar afectar el array original, 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)
```

```
# Las operaciones se propagan a lo largo del array
d = np.array([1,2,3,4,5])

# Se multiplican los elementos por 2
print(d*2, '\n')

# Se elevan al cuadrado los elementos del array
print(d**2)
```

Al momento de cambiar el primer elemento del **arreglo b**, el **arreglo a** se ve afectado, por lo tanto se crea un arreglo auxiliar (**arreglo c**) para usarlo como una copia y evitar la modificación no deseada del **arreglo a**, en la última de las imagines se puede notar como las operaciones sobre el **arreglo d** afecta a todos los elementos.

```
# Nueva definición para el array a
a = np.array([1,2,3,4,5])

# A continuación se va a iterar sobre todos los elementos del array
print(a>4, '\n')

# Se ejecuta la instrucción para los elementos que cumplan la condicion: "elem
ento > 4"
# Se cambian el contenido de los elmentos mayores a 4
a[a>4] = 1
print(a, '\n')

# Los elementos cuyo contenido es igual a 1, reciben como nuevo valor el númer
o 777
a[a==1] = 777
print(a, '\n')
```

En las instrucciones anteriores básicamente se comprueba una condición Con cada elemento del arreglo, los que la cumplen, son remplazados por El valor 1 y luego los elementos con el valor 1 son remplazados por el Valor 777, este tipo de cambios puede llegar a ser útil

```
# 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)]))
```

Para controlar y excluir la cantidad de valores erróneos sobre un arreglo

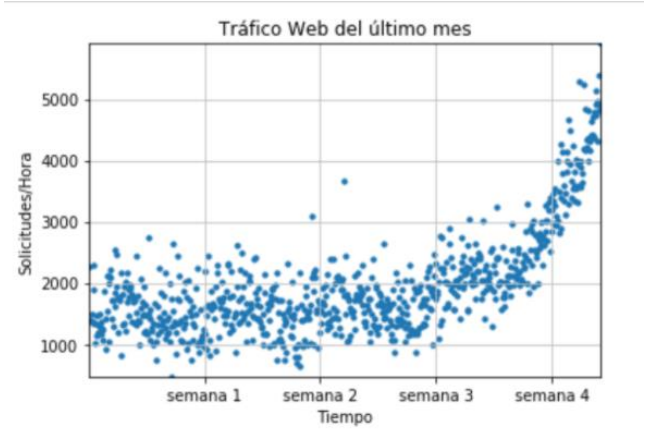
```
# Vamos a desarrollar un programa de machine learning (básico)
# El siguiente es un paquete de datos a ser procesdos:
#
# La primera columna es: Número de horas
# La segunda columna es: Número de tareas ejecutadas
data = np.genfromtxt("web_traffic.tsv", delimiter="\t")
print(data[:10], '\n')

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

```
# 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')
```

```
# Investigamos el número de valores nan que contiene el vector y
print(np.sum(np.isnan(y)))
```

```
# 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')
```

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

# Dibuja el punto (x,y) con círculos de tamaño 10
plt.scatter(x, y, s=10)
plt.title("Tráfico Web del último mes")
plt.xlabel("Tiempo")
plt.ylabel("Solicitudes/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')
plt.show()
```



PROYECTO machine learning

ENUNCIADO:

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.

En primeria instancia se busca separar los datos erróneos, para hacer un conteo
de los datos sobrantes usados en la gráfica de “Trafico web del último mes”