Apuntes de Inteligencia Artificial
Clase 16/02/2024

Rolbin Méndez Brenes

Profesor: Msc. Steven Andrey Pacheco Portuguéz

I. Discusión de la tarea 1 – Comentarios

- Yann LeCun: Fue uno de los padrinos de la inteligencia artificial, ganador del premio Touring y fue uno de los que trabajo mucho en computer visión, además, participo en la creación de un dataset con muchas imágenes escritas a mano con números sumamente pixeleados, actualmente trabaja en Facebook.
- Yoshua Bengio: Fue uno de los padrinos de la inteligencia artificial y también ganador del premio Touring, además, fue un pionero en el desarrollo de modelos y algoritmos para el aprendizaje profundo.
- Geoffrey Hinton: Fue uno de los padrinos de la inteligencia artificial y también ganador del premio Touring, actualmente trabaja en Canadá dando clases, en su momento fue un pionero en el backpropagation siendo uno de los primeros en desarrollar algoritmos para entrenar redes neuronales.
- Timnit Gebru: Sus principales aportes son en el tema de ética y equidad y específicamente trabaja como defensora en la investigación sobre la equidad y responsabilidad de aprendizaje automático, de esta manera ayuda a mitigar sesgos informáticos.
- Ian Goodfellow; Contribuyo mucho en el área de seguridad de la ciencia de inteligencia artificial, además, trabajo mucho en la generación de energías adversarias, la cual consiste en tener 2 redes neuronales la cual una se encarga en por ejemplo, identificar perros o gatos y otra se encarga de identificar perros y gatos, por lo que entre ellas se entrenan mutuamente para aumentar su red neuronal.
- Sam Altman: Es considerado uno de los pioneros o padres de la CNN: Redes Neuronales Convencionales, dichos avances revolucionaron todo el avance del reconocimiento de imágenes por medio de la inteligencia artificial.

II. Redes Neuronales Convolucionales

“Estas son un subconjunto de aprendizaje automático y están en el centro de los algoritmos de aprendizaje profundo. Están compuestas de capas de nodos, que contienen una capa de entrada, una o más capas ocultas y una cada de salida. Cada nodo se conecta a otro y tiene un peso y umbral asociados. Si la salida de cualquier nodo individual está por encima del valor del umbral especificado, ese nodo se activa y envía datos a la siguiente capa de la red. De lo contrario, no se pasa ningún dato a la siguiente capa de la red.” [1]

III. Sora | ChatGPT

Sora es una inteligencia artificial proveniente de los mismos creadores de ChatGPT que le permite a los usuarios por medio de un prompt crear un video con una alta resolución e la calidad del video y con mucho sentido en cada uno de sus planos. “Sora es capaz de generar escenas complejas con múltiples personajes, tipos específicos de movimiento y detalles precisos del sujeto y el fondo. El modelo comprende no sólo lo que el usuario ha pedido en el mensaje, sino también cómo existen esas cosas en el mundo físico.” [2]

IV.

V. Programación Vectorial

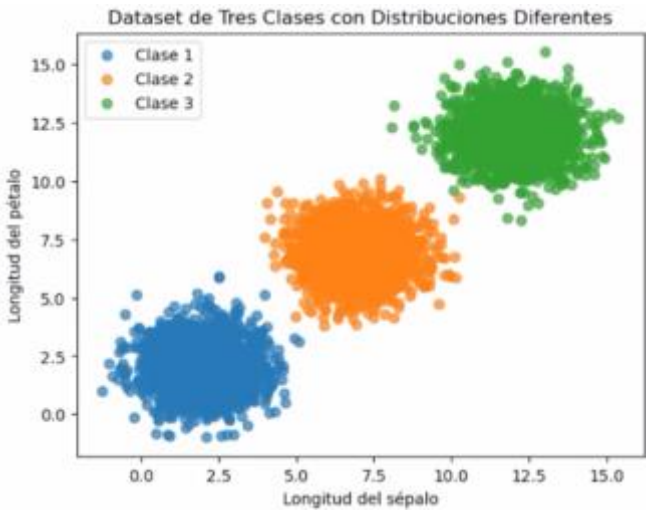
A. *Introducción a Matplotlib*

“Matplotlib es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python.” [3]

B. *Introducción a Pandas*

“Pandas es una herramienta de manipulación y análisis de datos de código abierto, rápida, potente, flexible y fácil de usar, construido sobre el lenguaje de programación Python.” [4]

Al utilizar Pandas Podemos utilizar un dataset de diferentes datos y poder mostrarlos de manera visual en un recuadro para poder observar donde se concentrar dichos elementos, si hay dispersión, entre otros, por ejemplo:



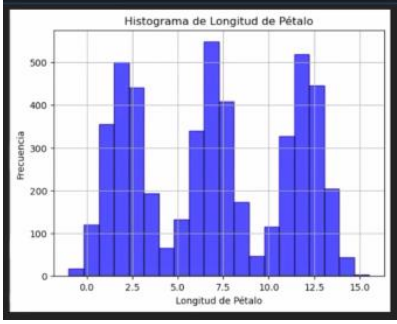
También podemos explorar los diferentes datos con solo imágenes, podemos obtener histogramas, gráficos,

entre otros plots y poder graficarlos de manera distinta para realizar análisis de diferentes datasets a lo largo del curso.

```
plt.hist(dataFrameTraining["Longitud_petalo"], bins=20, color='blue', edgecolor='black', alpha=0.7)

# Personalizar el gráfico
plt.title('Histograma de Longitud de Pétalo')
plt.xlabel('Longitud de Pétalo')
plt.ylabel('Frecuencia')
plt.grid(True)

# Mostrar el gráfico
plt.show()
```



Para ver más a fondo la librería de Pandas visitas este link

C. *Introducción a NumPy*

"NumPy es el paquete fundamental para la informática científica en Python. Es una biblioteca de Python que proporciona un objeto de matriz multidimensional, varios objetos derivados (como matrices y matrices enmascaradas) y una variedad de rutinas para operaciones rápidas en matrices, incluidas matemáticas, lógicas, manipulación de formas, clasificación, selección, E/S. , transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más." [5]

1) *Multiples Funciones*

- Arrays Multidimensionales
- Operaciones Vectorizadas
- Eficiencia Computacional
- Herramientas Matematicas y estadísticas
- Integración con otras bibliotecas
- Soporte para Algebra Lineal
- Compatibilidad con Hardware Especializado

2) *Creación de Arrays*

Para la creación de arrays definiremos un elemento de NumPy brindándole un rango de valores, o inicializándolo en cero.

Creación de Arrays

```
# Crear un array NumPy de 1D con los primeros 10 números enteros
array_1d = np.arange(10)
print("Array 1D:", array_1d)

array_1d = np.arange(5,10)
print("Array 1D:", array_1d)

array_1d = np.arange(5,10)
print("Array 1D:", array_1d)
array_1d = np.arange(5,10, 2)
print("Array 1D:", array_1d)
```

```
Array 1D: [0 1 2 3 4 5 6 7 8 9]
Array 1D: [5 6 7 8 9]
Array 1D: [5 6 7 8 9]
Array 1D: [5 7 9]
```

https://numpy.org/doc/stable/reference/generated/numpy.arange.html

```
array_1d = np.ones((10))
print("Array 1D:", array_1d)
array_1d = np.zeros((10))
print("Array 1D:", array_1d)
```

```
Array 1D: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
Array 1D: [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

3) *Creación de Matrices*

Para la creación de matrices se hará de manera muy similar que la de vectores solamente indicándole a NumPy la altura por la anchura de la matriz para crearla. Además, NumPy ofrece la función np.identity(largo) para crear una matriz identidad dándole el largo de la matriz.

Creación de Arrays: Matrices

```
array_1d = np.identity(3)
print(array_1d)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

```
array_1d = np.ones((4, 5))
print(array_1d)
```

```
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

```
array_1d = np.ones((2,4, 5))
print(array_1d)
```

```
[[[1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]]

 [[1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]
  [1. 1. 1. 1. 1.]]]
```

4) *Operaciones Básicas*

NumPy permite realizar sumar de diferentes elementos como sumas de Arrays, Restas, Multiplicaciones, Divisiones y Elevar de diferentes Arrays en diferentes dimensiones de la siguiente manera.

Operaciones básicas

```python
# Crear dos arrays NumPy de 1D con números aleatorios
lst = [2, 4, 6]
lst2 = [1, 3, 5]
# converting list to array
array_random_1 = np.array(lst)
array_random_2 = np.array(lst2)

# Realizar operaciones element-wise
suma_resultado = array_random_1 + array_random_2
resta_resultado = array_random_1 - array_random_2
multiplicacion_resultado = array_random_1 * array_random_2
division_resultado = array_random_1 / array_random_2

# Imprimir los resultados de cada operación

print("Suma:", suma_resultado)
print("Resta:", resta_resultado)
print("Multiplicación:", multiplicacion_resultado)
print("División:", division_resultado)
print("Elevar:", array_random_1**2)
```

```
Suma: [ 3  7 11]
Resta: [1 1 1]
Multiplicación: [ 2 12 30]
División: [2.         1.33333333 1.2       ]
Elevar: [ 4 16 36]
```

5) Funciones Estadisticas

Las funciones estadísticas nos permiten obtener diferentes valores de un array como la Media, la desviación estándar, el valor mínimo o el valor máximo.

Funciones estadísticas

```python
array_random = np.random.rand(100)
# Calcular la media, desviación estándar, valor máximo y mínimo
media = np.mean(array_random)
desviacion_estandar = np.std(array_random)
valor_maximo = np.max(array_random)
valor_minimo = np.min(array_random)

# Imprimir los resultados de cada operación
print("Media:", media)
print("Desviación estándar:", desviacion_estandar)
print("Valor máximo:", valor_maximo)
print("Valor mínimo:", valor_minimo)
```

```
Media: 0.4984987444949715
Desviación estándar: 0.28476451982115l1
Valor máximo: 0.975610991010298?
Valor mínimo: 0.009293305538057517
```

6) Distancia Euclidiana de una lista (Ciclos)

```python
# Función de programación por ciclos para calcular la distancia euclidi
def distancia_euclidiana_por_ciclo(vector1, vector2):
    inicio = time.time()

    resultado = 0
    # [a, b, c]
    # [1, 2, 3]
    # [(a, 1), (b, 2), (c, 3)]
    for elem1, elem2 in zip(vector1, vector2):
        resultado += (elem1 - elem2)**2

    resultado = np.sqrt(resultado)

    fin = time.time()
    tiempo_total = fin - inicio
    print(f"Tiempo por ciclos: {tiempo_total:.6f} segundos")

    return resultado
```

7) Distancia Euclidiana de una lista (Vectorial)

```python
def distancia_euclidiana_vectorial_explicita(vector1, vector2, printTime = True):
    inicio = time.time()

    # Elevar al cuadrado cada elemento de la diferencia
    diferencia_cuadrada = (vector1 - vector2)**2

    # Sumar los elementos cuadrados
    suma_cuadrada = np.sum(diferencia_cuadrada)

    # Calcular la raíz cuadrada de la suma
    resultado = np.sqrt(suma_cuadrada)

    fin = time.time()
    tiempo_total = fin - inicio
    if printTime:
        print(f"Tiempo vectorial explicito: {tiempo_total:.6f} segundos")

    return resultado
```

8) Comparación de Tiempos

Utilizando los diferentes métodos anteriores donde uno es escrito explícitamente con ciclos y otro con vectores podemos observar que la diferencia de tiempos es abismal, de esta forma muchas veces estas librerías son mucho mas eficientes que los métodos convencionales.

Comparación de tiempos

```python
# Crear dos listas de números aleatorios con un tamaño de N
lista_random_1 = np.random.rand(300000)
lista_random_2 = np.random.rand(300000)

# Utilizar las funciones con las listas y arrays aleatorios como inputs
resultado_ciclo = distancia_euclidiana_por_ciclo(lista_random_1, lista_random_2)
resultado_vectorial = distancia_euclidiana_vectorial_explicita(lista_random_1, lista_random_2)
# Imprimir los resultados completos
print("\nResultado completo por ciclos:", resultado_ciclo)
print("Resultado completo vectorial:", resultado_vectorial)
```

```
Tiempo por ciclos: 0.229363 segundos
Tiempo vectorial explicito: 0.001000 segundos

Resultado completo por ciclos: 223.60713457541607
Resultado completo vectorial: 223.60713457541158
```

D. K-Nearest Neighbor

"El algoritmo K-Nearest Neighbors (KNN) es una técnica popular de aprendizaje automático que se utiliza para tareas de clasificación y regresión. Se basa en la idea de que puntos de datos similares tienden a tener etiquetas o valores similares.

Durante la fase de entrenamiento, el algoritmo KNN almacena todo el conjunto de datos de entrenamiento como referencia." [6]

En resumen, la idea básica del KNN es clasificar un punto de datos seg[un la mayoría de las clases de sus k vecinos más cercanos basado en una métrica de distancia.

Para acceder al ejemplo de KNN de la clase visitar el cuaderno de Jupyter 3-Introducción a Numpy.ipynb.

REFERENCES

[1] "¿Qué son las redes neuronales convolucionales? | IBM". IBM in Deutschland, Österreich und der Schweiz. Accedido el 23 de febrero de 2024. [En línea]. Disponible: https://www.ibm.com/mx-es/topics/convolutional-neural-networks

[2] OpenAI. Creating video from text. [On line]. Available: https://openai.com/sora

[3] "Matplotlib — Visualization with Python". Matplotlib — Visualization with Python. Accedido el 23 de febrero de 2024. [En línea]. Disponible: https://matplotlib.org/

[4] "Pandas - Python Data Analysis Library". pandas - Python Data Analysis Library. Accedido el 23 de febrero de 2024. [En línea]. Disponible: https://pandas.pydata.org/

[5] "NumPy documentation — NumPy v1.26 Manual". NumPy -. Accedido el 23 de febrero de 2024. [En línea]. Disponible: https://numpy.org/doc/stable/

[6] "A Complete Guide to K-Nearest Neighbors (Updated 2024)". Analytics Vidhya. Accedido el 23 de febrero de 2024. [En línea]. Disponible: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/#:~:text=The%20K-Nearest%20Neighbors%20(KNN)%20algorithm%20is%20a%20popular,training%20dataset%20as%20a%20reference.