Apunte 1

Nicolás Gómez Morgado

Inteligencia Artificial

30 de agosto de 2024

Índice

1. Importante

Evaluaciones 2,3 y 4 en grupos (máximo 2-3 personas), 1 individual.

1. 13 sept
2. 4 oct
3. 30 oct
4. 20 dic

- Herramientas:
  - Phyton
  - Google colab

Tensor: Vector con otro nombre.

2. Inteligencia Artificial

2.1. Introducción

La inteligencia artificial es una rama de la informática que se encarga de desarrollar algoritmos y programas que permiten a las computadoras realizar tareas que requieren de la inteligencia humana. En este momento ya se ha logrado incluir la inteligencia artificial en muchos aspectos de la vida cotidiana, como por ejemplo en los asistentes virtuales, en los sistemas de recomendación, en los vehículos autónomos, en la medicina, en la industria, entre otros. Por lo cual un concepto que hace no mucho parecía de ciencia ficción, hoy en día es una realidad.

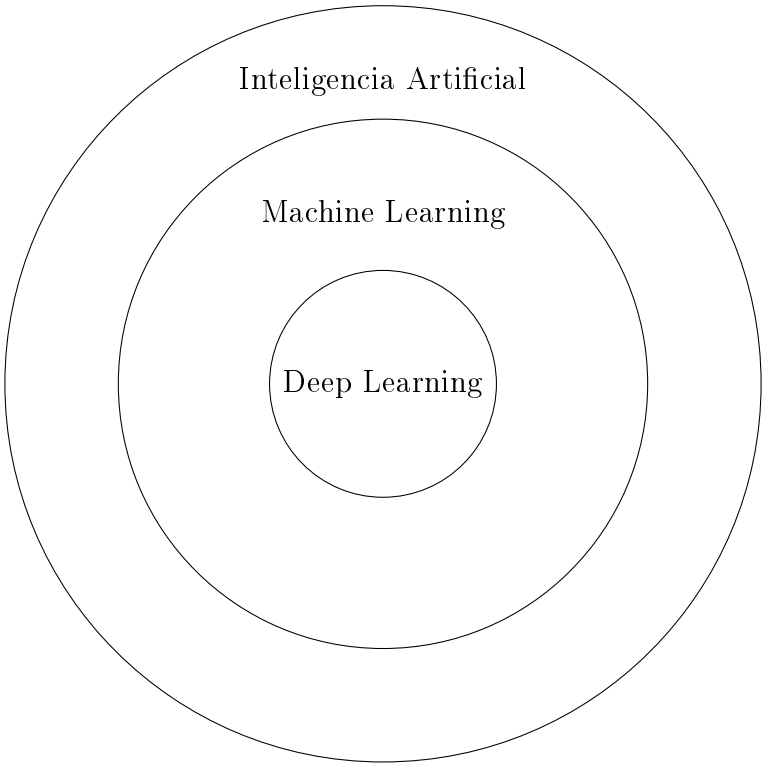
2.2. Conceptos fundamentales

Tiene como objetivo desarrollar algoritmos que permitan a las maquinas aprender de la experiencia y mejorar su rendimiento en tareas específicas. Para esto se utilizan técnicas de aprendizaje automático, que son un subconjunto de la inteligencia artificial.

Tipos de inteligencia artificial:

- **IA débil:** Realiza tareas especificas y limitadas.
- **IA fuerte:** Realiza tareas generales y complejas mas asociadas al pensar humano.

Ética en la IA: Que tipos de datos se utilizan, como se utilizan, que impacto tiene en la sociedad, etc. El objetivo es que la IA sea ética y presente algún tipo de sensibilidad social. A pesar de que la IA no tiene conciencia, si puede tener un impacto en la sociedad y siempre va a presentar errores, por lo que es importante tener en cuenta la ética en la IA.



Primeras "Inteligencias Artificiales" a crear van a estar enfocadas en Machine Learning.

Elementos para el aprendizaje de la IA: La IA requiere de fuentes de datos para entrenarse a si misma. Algunas fuentes de datos son:

- kaggle
- sklearn
- visualdata.io

2.3. Python

Python es un lenguaje de programación de alto nivel, interpretado y orientado a objetos. Es un lenguaje muy versátil y fácil de aprender, por lo que es muy utilizado en el desarrollo de aplicaciones web, en la ciencia de datos, en la inteligencia artificial, entre otros. Python es un lenguaje de programación muy popular en la actualidad, por lo que es muy probable que ya hayas escuchado hablar de él.

Librerías de Python

Python cuenta con una gran cantidad de librerías que facilitan el desarrollo de aplicaciones en diferentes áreas. La forma de importar una librería en Python es la siguiente:

```
import numpy as np
import pandas as pd
```

Uso de tipos de datos

Python es un lenguaje de programación que cuenta con varios tipos de datos, como enteros, flotantes, cadenas, booleanos, entre otros. A continuación se muestra un ejemplo de cómo se pueden utilizar estos tipos de datos en Python:

```
x = 29
s = "hola"
f = 15.9
x = False

print(x)
print(s)
print(f)

print(s+str(f))
print(s,f)

# Variables y Operadores
a = 10
b = 5
suma = a + b
print(f"La suma es: {suma}")  # Output: La suma es: 15

# Ciclos y condiciones
```

```
    for i in range(5):
        if i % 2 == 0:
            print(f"{i} es par")
        else:
            print(f"{i} es impar")

    a = 45.7875454
    print("El valor es {:.3f}".format(a))
```

Funciones y uso de parámetros opcionales

En python se pueden definir funciones de las cuales existen dos tipos de parámetros, los obligatorios y los opcionales. Los parámetros opcionales son aquellos que tienen un valor por defecto y no es necesario pasarlos al llamar la función. A continuación se muestra un ejemplo de cómo se pueden definir funciones en Python:

```
    # Definiendo una función
    def saludo(nombre):
        return f"Hola, {nombre}!"

    print(saludo("Jazna"))  # Output: Hola, Jazna!

    # Función con parámetro opcional
    def f(x, y = 4):
        return x + y

    print(f(2))
    print(f(2,8))

    # Función con multiples parámetros opcionales
    def f(x, y = 4, z = 5):
        return x + y + z

    f(3) # Un parámetro de entrada
    f(7, z = 9) # Dos parámetros de entrada con uno opcional (x)
```

Colecciones de datos

En python a los arrays se les llama listas, ademas de estos existen los diccionarios y las comprensiones de listas. A continuación se muestra un ejemplo de cómo se pueden utilizar estas colecciones de datos en Python:

```
    # Listas
    lista = [1, 2, 3, 4, 5]
    print(lista[0])  # Output: 1

    # Diccionarios: Colección de datos que se almacenan
    # en pares clave-valor
    diccionario = {'nombre': 'Ana', 'edad': 25}
    print(diccionario['nombre'])  # Output: Ana

    # Comprensiones de listas: Forma concisa de crear listas al usar []
    cuadrados = [x**2 for x in lista] # Elevar al cuadrado cada elemento
    print(cuadrados)  # Output: [1, 4, 9, 16, 25]
```

Bibliotecas para ciencias de datos

```
    ## Numpy

    lista = [15,20,60,89,45,78,30]
    print("El promedio es {:.2f}".format(np.mean(lista)))
    print("La desviacion estandar es {:.2f}".format(np.std(lista)))
    print("La mediana es {:.2f}".format(np.median(lista)))
    print(f"El valor maximo es {np.max(lista)} y esta en el indice
    {np.argmax(lista)}")
    print(f"El valor minimo es {np.min(lista)} y esta en el indice
    {np.argmin(lista)}")

    # Crear un array
    arreglo = np.array([1, 2, 3, 4, 5])
    print(arreglo * 2)  # Output: [ 2  4  6  8 10]

    # Operaciones matematicas
    matriz = np.array([[1, 2], [3, 4]])
    print(np.dot(matriz, matriz)) # Output: [[7 10] [15 22]]


    ## Pandas

    # Crear un DataFrame
    datos = {'Nombre': ['Ana', 'Luis', 'Marta'], 'Edad': [25, 32, 22]}
    df = pd.DataFrame(datos)
    print(df) # Output: Nombre Edad 0 Ana 25 1 Luis 32 2 ... (tabla)

    # Filtrar datos
    mayores_25 = df[df['Edad'] > 25]
    print(mayores_25) # Output: Nombre Edad 1 Luis 32 (tabla)


    ## Matplotlib
    import matplotlib.pyplot as plt

    # Grafico simple
    x = [1, 2, 3, 4, 5]
    y = [2, 4, 6, 8, 10]
    plt.plot(x, y)
    plt.title('Grafico de ejemplo', fontsize=16, fontweight="bold")
    plt.xlabel('x', fontsize=14, fontweight="bold")
    plt.ylabel('y', fontsize=14, fontweight="bold")
    plt.show() # Output: Grafico de ejemplo
```

Manipulación de datos

→ Preprocesamiento: Hace referencia a la limpieza y transformación de los datos para que puedan ser utilizados por los algoritmos de aprendizaje automático.

```
from sklearn.preprocessing import StandardScaler

# Datos de ejemplo

datos = [[1, 2], [3, 4], [5, 6]]
scaler = StandardScaler()
datos_normalizados = scaler.fit_transform(datos)
print(datos_normalizados)
```

Estandarización

$$\text{Valor estandarizado} = \frac{\text{Original} - \bar{x}}{\text{STD}}$$

→ Ingeniería de características: Consiste en la creación de nuevas características a partir de las características existentes para mejorar el rendimiento de los algoritmos de aprendizaje automático (machine learning).

```
from sklearn.preprocessing import PolynomialFeatures

# Crear caracteristicas polinomicas
poly = PolynomialFeatures(degree=2) # Grado 2 para el polinomio
datos = [[2, 3]]
print(poly.fit_transform(datos))  # Output: [[1. 2. 3. 4. 6. 9.]]
```

$$\text{Características polinómicas} = [\text{bias}, x_1, x_2, {x_1}^2, x_1 \cdot x_2, {x_2}^2]$$

Algebra lineal

→ Producto cruz entre vectores

```
# Definir dos vectores
x_vector = np.array([1, 2, 3])
y_vector = np.array([4, 5, 6])

# Calcular el producto punto
producto_punto = np.dot(x_vector, y_vector)

print("Producto Punto:", producto_punto)
```

→ Multiplicación entre matrices: Las redes neuronales utilizan la multiplicación de matrices para realizar operaciones matemáticas.

```
# Definir dos matrices
matriz_a = np.array([[1, 2, 3],
                     [4, 5, 6]])
matriz_b = np.array([[7, 8],
                     [9, 10],
                     [11, 12]])

# Calcular la multiplicacion de matrices
resultado = np.dot(matriz_a, matriz_b)

print("Multiplicacion de Matrices:\n", resultado)
```

→ Descomposición en valores singulares: Reducción de la dimensionalidad para utilizarse en métodos como PCA y tratamientos de datos de la IA. Su uso es mas común en Deep Learning.

```
    # Crear una matriz
    matriz = np.array([[1, 2, 3],
                       [4, 5, 6],
                       [7, 8, 9]])

    # Aplicar SVD
    U, S, Vt = np.linalg.svd(matriz)

    print("Matriz U:\n", U)
    print("Valores singulares S:\n", S)
    print("Matriz V traspuesta:\n", Vt)
```

→ Tensores y escalares:

```
    # Escalar (Tensor de rango 0)
    escalar = 29

    # Vector (Tensor de rango 1)
    vector = np.array([1, 2, 3])
    print("Vector\n", vector)

    # Matriz (Tensor de rango 2)
    matriz = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
    print("Matriz\n", matriz)

    # Tensor de rango 3
    tensor_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]],
                          [[9, 10], [11, 12]]])
    print("Tensor 3D\n", tensor_3d)
```

→ Operaciones entre vectores:
→ Funciones unviersales:
→ Manipulacion de forma:

3. Matemáticas aplicadas a la IA

4. Deep Learning

5. Machine Learning