



Apunte 1

Nicolás Gómez Morgado

Inteligencia Artificial

11 de septiembre de 2024

Índice

1. Importante	2
2. Inteligencia Artificial	3
2.1. Introducción	3
2.2. Conceptos fundamentales	3
2.3. Python	4
2.4. Algebra lineal en Python	7
3. Matemáticas aplicadas a la IA	13
4. Deep Learning	14
5. Machine Learning	15



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.

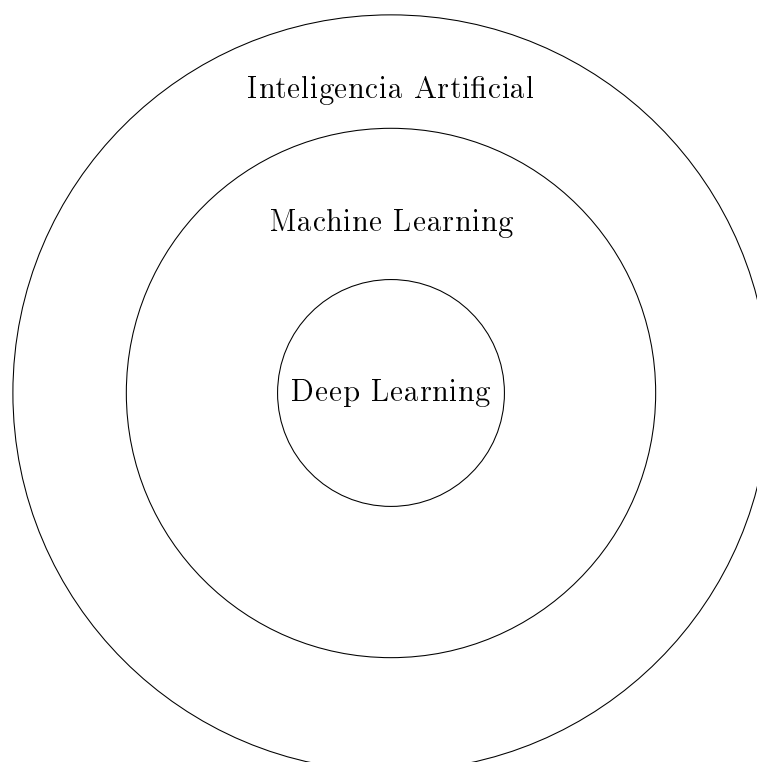
2.2. Conceptos fundamentales

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 específicas 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:

```
1 import numpy as np
2 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:

```
1 x = 29
2 s = "hola"
3 f = 15.9
4 x = False
5
6 print(x)
7 print(s)
8 print(f)
9
10 print(s+str(f))
11 print(s,f)
12
13 # Variables y Operadores
14 a = 10
15 b = 5
16 suma = a + b
17 print(f"La suma es: {suma}") # Output: La suma es: 15
18
19 # Ciclos y condiciones
20 for i in range(5):
```

```
21     if i % 2 == 0:
22         print(f"{i} es par")
23     else:
24         print(f"{i} es impar")
25
26     a = 45.7875454
27     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:

```
1     # Definiendo una función
2     def saludo(nombre):
3         return f"Hola, {nombre}!"
4
5     print(saludo("Jazna")) # Output: Hola, Jazna!
6
7     # Función con parámetro opcional
8     def f(x, y = 4):
9         return x + y
10
11     print(f(2))
12     print(f(2,8))
13
14     # Función con multiples parámetros opcionales
15     def f(x, y = 4, z = 5):
16         return x + y + z
17
18     f(3) # Un parámetro de entrada
19     f(7, z = 9) # Dos parámetros de entrada con uno opcional (x)
```

Colecciones de datos

En python a los arrays se les llama listas, además 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:

```
1     # Listas
2     lista = [1, 2, 3, 4, 5]
3     print(lista[0]) # Output: 1
4
5     # Diccionarios: Colección de datos que se almacenan
6     # en pares clave-valor
7     diccionario = {'nombre': 'Ana', 'edad': 25}
8     print(diccionario['nombre']) # Output: Ana
9
10    # Comprensiones de listas: Forma concisa de crear listas al usar []
11    cuadrados = [x**2 for x in lista] # Elevar al cuadrado cada elemento
12    print(cuadrados) # Output: [1, 4, 9, 16, 25]
```

Bibliotecas para ciencias de datos

```
1  ## Numpy
2
3  lista = [15,20,60,89,45,78,30]
4  print("El promedio es {:.2f}".format(np.mean(lista)))
5  print("La desviacion estandar es {:.2f}".format(np.std(lista)))
6  print("La mediana es {:.2f}".format(np.median(lista)))
7  print(f"El valor maximo es {np.max(lista)} y esta en el indice
8  {np.argmax(lista)}")
9  print(f"El valor minimo es {np.min(lista)} y esta en el indice
10 {np.argmin(lista)}")
11
12 # Crear un array
13 arreglo = np.array([1, 2, 3, 4, 5])
14 print(arreglo * 2) # Output: [ 2  4  6  8 10]
15
16 # Operaciones matematicas
17 matriz = np.array([[1, 2], [3, 4]])
18 print(np.dot(matriz, matriz)) # Output: [[7 10] [15 22]]
19
20
21 ## Pandas
22
23 # Crear un DataFrame
24 datos = {'Nombre': ['Ana', 'Luis', 'Marta'], 'Edad': [25, 32, 22]}
25 df = pd.DataFrame(datos)
26 print(df) # Output: Nombre Edad 0 Ana 25 1 Luis 32 2 ... (tabla)
27
28 # Filtrar datos
29 mayores_25 = df[df['Edad'] > 25]
30 print(mayores_25) # Output: Nombre Edad 1 Luis 32 (tabla)
31
32
33 ## Matplotlib
34 import matplotlib.pyplot as plt
35
36 # Grafico simple
37 x = [1, 2, 3, 4, 5]
38 y = [2, 4, 6, 8, 10]
39 plt.plot(x, y)
40 plt.title('Grafico de ejemplo', fontsize=16, fontweight="bold")
41 plt.xlabel('x', fontsize=14, fontweight="bold")
42 plt.ylabel('y', fontsize=14, fontweight="bold")
43 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.

```
1 from sklearn.preprocessing import StandardScaler
2
3 # Datos de ejemplo
4
5 datos = [[1, 2], [3, 4], [5, 6]]
6 scaler = StandardScaler()
7 datos_normalizados = scaler.fit_transform(datos)
8 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).

```
1 from sklearn.preprocessing import PolynomialFeatures
2
3 # Crear características polinómicas
4 poly = PolynomialFeatures(degree=2) # Grado 2 para el polinomio
5 datos = [[2, 3]]
6 print(poly.fit_transform(datos)) # Output: [[1. 2. 3. 4. 6. 9.]]
```

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

2.4. Álgebra lineal en Python

→ Producto punto entre vectores: El producto punto (o producto escalar) es una operación fundamental en álgebra lineal y se usa en muchos algoritmos de IA, como en redes neuronales y en métodos de optimización.

```
1 # Definir dos vectores
2 x_vector = np.array([1, 2, 3])
3 y_vector = np.array([4, 5, 6])
4
5 # Calcular el producto punto
6 producto_punto = np.dot(x_vector, y_vector)
7
8 print("Producto Punto:", producto_punto)
```

→ Multiplicación entre matrices: Las redes neuronales utilizan la multiplicación de matriz para realizar operaciones matemáticas. Es esencial también debido a la propagación hacia adelante y hacia atrás en redes neuronales.

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

```

5         [9, 10],
6         [11, 12]])
7
8     # Calcular la multiplicacion de matrices
9     resultado = np.dot(matriz_a, matriz_b)
10
11     print("Multiplicacion de Matrices:\n", resultado)

```

→ Vector ortogonal: Dos vectores se consideran ortogonales si su producto punto (producto escalar) es igual a cero. En otras palabras si el ángulo entre ellos es de 90 grados.

→ Matriz ortogonal: Una matriz cuadrada es ortogonal si su transpuesta es igual a su inversa:

$$A^T \cdot A = 1$$

→ Vector propio: Un vector propio (o autovector) de una matriz es un vector no nulo que cambia de magnitud pero no de dirección cuando se aplica la matriz a él. Es decir, si se tiene una matriz \mathbf{A} y un vector \mathbf{v} , el vector \mathbf{v} es un vector propio de \mathbf{A} si cumple con la siguiente ecuación:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$$

donde:

- \mathbf{A} es una matriz cuadrada.
- \mathbf{v} es el vector propio.
- λ es un valor propio (o autovalor), que es un escalar.

El valor propio asociado a un vector propio indica en qué medida se escala el vector propio cuando se aplica la matriz \mathbf{A} .

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

```

1     # Crear una matriz
2     matriz = np.array([[1, 2, 3],
3                        [4, 5, 6],
4                        [7, 8, 9]])
5
6     # Aplicar SVD
7     U, S, Vt = np.linalg.svd(matriz)
8
9     print("Matriz U:\n", U)
10    print("Valores singulares S:\n", S)
11    print("Matriz V traspuesta:\n", Vt)

```

→ Tensores y escalares: Nombre técnico de los vectores y matrices en el mundo de la IA.

- Manejo de datos en varias dimensiones.

- Reconocimiento de imágenes (3 canales de colores=3dimensiones).
- Tensor de rango 0: Escalar.
- Tensor de rango 1: Vector.
- Tensor de rango 2: Matriz.
- Tensor de rango 3: Tensor 3D (hablamos de profundidad).

```
1 # Escalar (Tensor de rango 0)
2 escalar = 29
3
4 # Vector (Tensor de rango 1)
5 vector = np.array([1, 2, 3])
6 print("Vector\n", vector)
7
8 # Matriz (Tensor de rango 2)
9 matriz = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
10 print("Matriz\n", matriz)
11
12 # Tensor de rango 3
13 tensor_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]],
14                       [[9, 10], [11, 12]]])
15 print("Tensor 3D\n", tensor_3d)
```

→ Operaciones entre vectores(tensores de rango 1):

- Se pueden sumar, restar, multiplicar y dividir.
- Se pueden realizar cálculos con escalares(tensores de rango 0).

→ Funciones universales: Funciones que se aplican a todos los elementos de un tensor.

- **np.exp(tensor)**: Calcula el exponencial de cada elemento del tensor.
- **np.log(tensor)**: Calcula el logaritmo natural de cada elemento del tensor.
- **np.sin(tensor)**: Calcula el seno de cada elemento del tensor.
- **np.cos(tensor)**: Calcula el coseno de cada elemento del tensor.
- **np.tan(tensor)**: Calcula la tangente de cada elemento del tensor.
- **np.sqrt(tensor)**: Calcula la raíz cuadrada de cada elemento del tensor.
- **np.abs(tensor)**: Calcula el valor absoluto de cada elemento del tensor.
- **np.sum(tensor)**: Suma todos los elementos del tensor.
- **np.sum(tensor, axis= n)**: Suma los elementos de cada **columna n** del tensor.
- **np.mean(tensor)**: Calcula el promedio de todos los elementos del tensor.
- **np.std(tensor)**: Calcula la desviación estándar de todos los elementos del tensor.
- **np.max(tensor)**: Encuentra el valor máximo del tensor.
- **np.min(tensor)**: Encuentra el valor mínimo del tensor.
- **np.argmax(tensor)**: Encuentra el índice del valor máximo del tensor.

- **np.argmax(tensor)**: Encuentra el índice del valor mínimo del tensor.

- Y así muchas otras funciones.

→ Manipulación de forma:

- **np.shape**: Retorna los valores de la cantidad de filas y columnas en una tupla.
- **np.reshape(tensor, (filas, columnas))**: Cambia la forma de un tensor. (fila = n, columna = m para cambiar las cantidades).
- **np.transpose(tensor)**: Transpone un tensor.
- **np.flatten(tensor)**: Convierte un tensor de cualquier rango en un vector (tensor de rango 0).

→ Indexación y selección:

- Un tensor de rango 1 no puede ser indexado como un tensor de rango 2 siempre que la cantidad de elementos sea prima.
- **tensor[0]**: Retorna el primer elemento del tensor.
- **tensor[-1]**: Retorna el último elemento del tensor.
- **tensor[0:3]**: Retorna los elementos desde el índice 0 hasta el 2 (uno menos el último).
- **tensor[2:]**: Retorna los elementos desde el índice 2 hasta el final.
- **tensor[:3]**: Retorna los elementos desde el inicio hasta el índice 2.
- **tensor[1,2]**: Retorna el elemento en la fila 1 y columna 2.
- **tensor[tensor == 5]**: Retorna los elementos que cumplen con la condición (iguales a 5).

Probabilidad

→ Distribución normal: Técnica de probabilidad que describe la distribución de los datos.

- Campana de Gauss.

→ Teorema de Bayes: Clasificación de datos en base a la probabilidad de que ocurran ciertos eventos basándose en otros eventos.

- **P(A|B)**: Probabilidad de que ocurra el evento A dado que ocurrió el evento B.
- **P(B|A)**: Probabilidad de que ocurra el evento B dado que ocurrió el evento A.
- **P(A)**: Probabilidad de que ocurra el evento A.
- **P(B)**: Probabilidad de que ocurra el evento B.

Capítulos del libro a estudiar ():

HASTA AQUÍ LA EVALUACIÓN DEL 13 DE SEPTIEMBRE + CAPÍTULOS DEL LIBRO

Medidas de tendencia central

Nos ayuda a entender lo que tenemos para crear modelos de calidad.

→ Promedio o media aritmética: Es la suma de todos los valores dividida por la cantidad de valores.

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

- Es una medida sensible a los valores atípicos (outlayers).
- Paradoja de Bil Gates: Si Bill Gates entra a un bar, el promedio de riqueza de los presentes aumenta.
- Valores atípicos (outlayers): Valores que se alejan mucho de la media o de los datos en general.
- Para casos con valores atípicos es preferible utilizar la mediana para describir los datos.

→ Varianza: Es la media de los cuadrados de las diferencias entre cada valor y la media.

$$\begin{aligned} \text{Población: } \sigma^2 &= \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n} \\ \text{Muestra: } s^2 &= \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1} \end{aligned}$$

→ Desviación estándar: Es la raíz cuadrada de la varianza.

$$\begin{aligned} \text{Población: } \sigma &= \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}} = \sqrt{\sigma^2} \\ \text{Muestra: } s &= \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}} = \sqrt{s^2} \end{aligned}$$

- **np.mean(tensor)**: Calcula el promedio (valor mas común) de todos los elementos del tensor.
- **np.median(tensor)**: Calcula la mediana (valor central) de todos los elementos del tensor.
- **np.varimuestra(tensor)**: Calcula la varianza de todos los elementos del tensor.
- **np.std(tensor)**: Calcula la desviación estándar de todos los elementos del tensor.
- **np.sqrt(np.varimuestra(tensor))**: Calcula la desviación estándar de todos los elementos del tensor.
- En una situación perfecta para una gráfica de gauss la media y la mediana son iguales. En otro caso el gráfico se sesga.



- $\mu - \Sigma$ = Media - Desviación estándar = 68 % de los datos.
- $\mu + \Sigma$ = Media + Desviación estándar = 68 % de los datos.

Medidas de posición



3. Matemáticas aplicadas a la IA



4. Deep Learning



5. Machine Learning