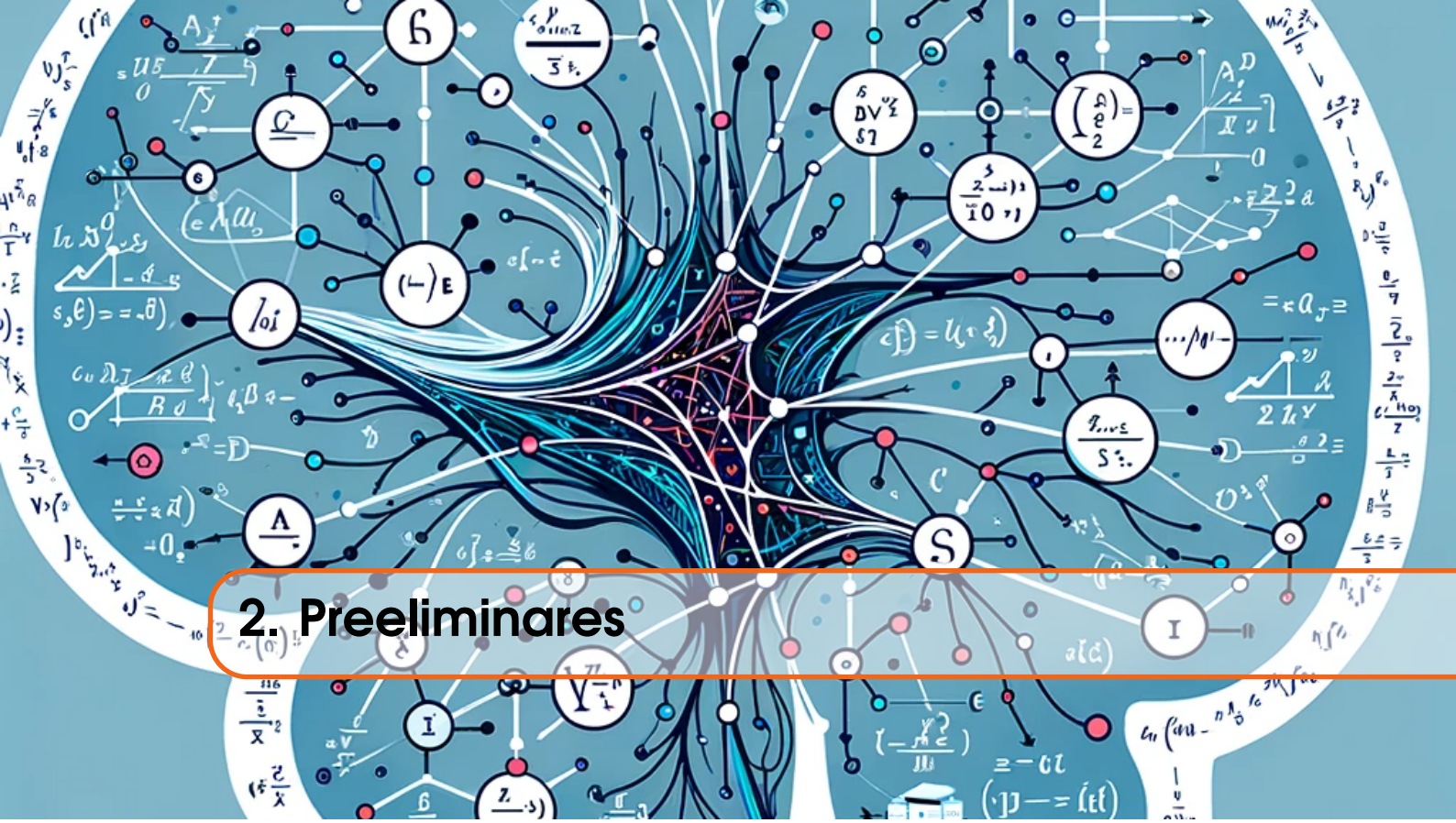
Parte Uno: Fundamentos



2. Preeliminares

2.1 Introducción

En este capítulo, nos adentraremos en los cimientos matemáticos que son esenciales para comprender y trabajar efectivamente con redes neuronales. A menudo, el poder y la eficacia de las redes neuronales en tareas de aprendizaje automático pueden parecer casi mágicos. Sin embargo, este "truco de magia" se basa firmemente en principios matemáticos bien establecidos. Aquí, exploraremos estos principios con el fin de desmitificar cómo las redes neuronales aprenden a partir de datos y realizan predicciones o clasificaciones asombrosamente precisas.

Los temas incluidos en este capítulo son fundamentales en la ciencia de datos y la inteligencia artificial: álgebra lineal, cálculo diferencial, probabilidad y estadística. Cada uno de estos temas contribuye de manera única al funcionamiento de las redes neuronales.

La álgebra Lineal es el lenguaje en el que se expresan las redes neuronales. Entender conceptos como vectores, matrices, y operaciones como la multiplicación de matrices, es esencial para comprender cómo las redes procesan y transforman los datos.

Por otra parte, las redes neuronales aprenden ajustando sus parámetros para mejorar su rendimiento. Este proceso de aprendizaje es guiado por el cálculo diferencial, en particular por conceptos como derivadas y gradientes. Comprender estos conceptos es crucial para entender cómo se optimizan las redes durante el entrenamiento.

En otro canal igual de importante, podemos ver que las redes neuronales, en su núcleo, realizan tareas de inferencia estadística. Entender la teoría de probabilidad y las técnicas estadísticas nos permite comprender cómo las redes neuronales modelan la incertidumbre y hacen predicciones basadas en datos. Pero sobre todo nos dan las herramientas para desmitificar las cajas negras y comprender a las redes como procesos estocásticos que modelan problemas del mundo real.

2.2 Algebra lineal

El álgebra lineal es una rama de las matemáticas que estudia conceptos tales como vectores, matrices, espacio dual, sistemas de ecuaciones lineales y en su enfoque de manera más formal, espacios vectoriales y sus transformaciones lineales. Esta rama de las matemáticas es esencial en

una amplia gama de campos, incluyendo la ciencia, la ingeniería, la economía, las finanzas y en especial la inteligencia artificial.

2.2.1 Vectores y espacios vectoriales

En matemáticas, un vector es un objeto que tiene magnitud y dirección. Se puede representar gráficamente como un segmento de recta orientado, o de forma algebraica como una tupla de números.

$$\overrightarrow{v} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ n \end{bmatrix}$$

La notación para describir en que espacio se encuentra un vector, v, puede ser algún conjunto de números como los reales, $\mathbb{R}$, y este puede tener un número n de dimensiones, teniendo así:

$$\overrightarrow{v} \in \mathbb{R}^n$$

La magnitud de un vector es su longitud, y se puede medir en unidades de longitud, como metros, kilómetros o pies. La dirección de un vector es la dirección en la que está apuntando, y se puede especificar por un ángulo o por un punto cardinal.

Los vectores se pueden sumar, restar, multiplicar por escalares.

- La suma de dos vectores es otro vector que apunta en la misma dirección que los dos vectores originales, pero con una magnitud que es la suma de las magnitudes de los dos vectores originales.
- La resta de dos vectores es otro vector que apunta en la dirección opuesta a la de los dos vectores originales, pero con una magnitud que es la diferencia de las magnitudes de los dos vectores originales.
- La multiplicación de un vector por un escalar es otro vector que tiene la misma dirección que el vector original, pero con una magnitud que es el producto del escalar por la magnitud del vector original.
- El producto escalar de dos vectores es un número real que representa el producto de las magnitudes de los vectores originales y el coseno del ángulo entre ellos.

Suma de vectores

La suma de dos vectores se puede calcular algebraicamente utilizando las componentes de los vectores. Si los vectores tienen componentes (x_1, y_1) y (x_2, y_2), entonces su suma es el vector $(x_1 + x_2, y_1 + y_2)$.

Resta de vectores

La resta de dos vectores se puede calcular algebraicamente utilizando las componentes de los vectores. Si los vectores tienen componentes (x_1, y_1) y (x_2, y_2), entonces su resta es el vector $(x_1 - x_2, y_1 - y_2)$.

Multiplicación de un vector por un escalar

La multiplicación de un vector por un escalar, a, también se puede calcular algebraicamente utilizando las componentes del vector. Si el vector tiene componentes (x, y) y el escalar es a, entonces el producto del vector por el escalar es el vector (ax, ay).

Producto escalar de dos vectores

Si los vectores tienen componentes (x_1, y_1) y (x_2, y_2), entonces su producto escalar es:

$$(x_1, y_1) \cdot (x_2, y_2) = x_1 \cdot x_2 + y_1 \cdot y_2$$

2.2.2 Matrices

Las matrices son arreglos de filas y columnas, útiles para hacer operaciones numéricas en aplicaciones como en las matemáticas. Es común que este arreglo sea de m-filas y n-columnas, creando así un arreglo con tamaño $m \times n$:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \tag{2.1}$$

misma que es posible reescribir como $A = [a_{ij}]$.

Suma de matrices

Si tenemos dos matrices A y B de $m \times n$, el resultado de la suma de ambas será una matriz $m \times n$, $A + B$:

$$A + B = (a_{ij} + b_{ij}) = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{bmatrix} \tag{2.2}$$

es importante recordar que para realizar la operación de suma ambas matrices deben tener las mismas dimensiones de lo contrario será imposible realizar esta operación ya que como se observa la suma se realiza elemento a elemento.

Producto

El producto entre dos matrices solo es posible si satisface una condición y esta requiere que la primera matriz tenga un tamaño $m \times n$ y la segunda $n \times p$ (nótese que ambas comparten en sus dimensiones el valor n), dando como resultado a una nueva matriz con dimensiones $m \times p$; es decir, si se tienen dos matrices AyB con dimensiones $2 \times 3y3 \times 4$ respectivamente, el resultado será una nueva matriz cuyas dimensiones serán 2×4.

$$AB = (a_{ij}b_{ij}) = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1p} \\ c_{21} & c_{22} & \cdots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mp} \end{bmatrix} \tag{2.3}$$

y para calcular los elementos de cada una de las filas desplazaremos de acuerdo al elemento que estemos calculado de la matriz resultante, dado por el subíndice de c_{ij} donde i será la fila correspondiente de A y j la columna correspondiente de B, para un primer elemento tendremos:

$$c_{11} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ \vdots \\ b_{n1} \end{bmatrix} = a_{11}b_{11} + a_{12}b_{21} + \cdots + a_{1n}b_{n1} \tag{2.4}$$

y de esta misma forma se calcula cada uno de los elementos correspondientes de la matriz resultante de la operación de producto.

Matriz inversa

La inversa de una matriz cuadrada, A, es otra matriz cuadrada, A^{-1}, que satisface la siguiente ecuación $AA^{-1} = I$, siendo I la matriz identidad; es decir una matriz inversa es aquella que multiplicada por la matriz original da como resultado a la matriz identidad. Para que una matriz tenga inversa el determinante de esta debe ser distinto de cero.

Algunos métodos para calcular la matriz inversa son:

- Método del adjunto
- Método de Gauss-Jordad
- Método de Cramer

Matriz identidad

La matriz identidad, I, es una matriz de dimensiones $n \times n$, es decir es una matriz cuadrada en la que sus elementos de la diagonal principal tienen como valor unos y el resto tienen valor cero.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

La inversa de la matriz identidad siempre será la misma matriz identidad.

Determinante

El determinante de una matriz cuadrada, A, es un numero real y este es de utilidad para calcular la distancia entre dos vectores, determinar si es que una matriz es invertible, resolver sistemas de ecuaciones lineales. Y lo podemos encontrar representado como:

$$det(A) = |A|$$

Algunos métodos para calcular el determinante son: la regla de Sarrus, Teorema de Laplace o método de cofactores

2.2.3 Ejercicios

Exercise 2.1 Dadas las siguientes matrices,

$$A = \begin{bmatrix} -8 & -5 & -7 \\ -3 & 9 & 5 \end{bmatrix}, B = \begin{bmatrix} 4 & -5 \\ -8 & -3 \end{bmatrix}, C = \begin{bmatrix} 7 & 5 & 8 \\ -6 & 5 & -4 \end{bmatrix},$$
$$D = \begin{bmatrix} -2 \\ 4 \end{bmatrix}, E = \begin{bmatrix} 9 & 0 & 8 \\ -3 & 8 & 4 \\ -8 & -9 & 5 \end{bmatrix}, F = \begin{bmatrix} 10 & -5 \\ 6 & -4 \\ 0 & -15 \end{bmatrix},$$
$$G = \begin{bmatrix} 1 & 6 \\ 9 & 2 \end{bmatrix}, H = \begin{bmatrix} 2 & -8 & -1 \\ -8 & 4 & 11 \\ 6 & 5 & 4 \end{bmatrix}$$

Calcular

1. A+C
2. C-A
3. EH
4. GD

■

Exercise 2.2 Usando las matrices del ejericio anterior. Hallar la transpuesta de las matrices B y F. ■

2.3 Cálculo

El cálculo es una rama de las matemáticas que se enfoca en el estudio de las tasas de cambio y las acumulaciones infinitas [19]. Esta áreas se divide en dos: el cálculo diferencial, que trata sobre cómo cambian las cantidades, y el cálculo integral, que se ocupa de la acumulación de cantidades y de la determinación de áreas bajo curvas, entre otras aplicaciones. El cálculo diferencial permite analizar la tasa de cambio de una función, mientras que el cálculo integral permite encontrar la cantidad total acumulada de algo. Estos conceptos se desarrollan a través de herramientas matemáticas como límites, derivadas e integrales, que permiten modelar y resolver problemas complejos en una variedad de disciplinas.

En el campo del aprendizaje automático, el cálculo tiene varias aplicaciones. Por ejemplo, el cálculo diferencial es esencial en la optimización de funciones, que es un componente clave en la mayoría de los algoritmos de aprendizaje automático. Al entrenar un modelo, como una red neuronal, se busca minimizar una función de pérdida, que mide cuán lejos está el modelo de realizar predicciones precisas. Para lograrlo, se utilizan métodos como el descenso de gradiente, que emplean derivadas para determinar la dirección y magnitud de los cambios necesarios en los parámetros del modelo para reducir la pérdida.

El cálculo integral también juega un papel en el aprendizaje automático, especialmente en la estimación de probabilidades y en técnicas como la integración numérica, que se utiliza para aproximar valores en modelos probabilísticos.

2.3.1 Calculo diferencial

De manera intuitiva, la derivada mide cómo cambia el valor de una función cuando la variable independiente experimenta un pequeño cambio. De forma general nos permite entender cómo varía una cantidad en relación con otra.

Formante, si $y = f(x)$, entonces la derivada se define como la tasa de incremento en la variable observada con respecto a un intervalo muy pequeño. Es decir,

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{2.5}$$

alternativamente, las siguientes notaciones para la derivada son válidas.

$$f'(x) = y' = \frac{df}{dx} = \frac{dy}{dx} = \frac{d}{dx}(f(x)) \tag{2.6}$$

2.3.2 Derivadas parciales

Las derivadas parciales son una extensión del concepto de derivada para funciones de varias variables. Mientras que la derivada ordinaria mide la tasa de cambio de una función respecto a una sola variable, manteniendo todas las demás constantes.

Suponiendo una función de varias variables, $y = f(x_1 \dots x_n)$, la derivada parcial con respecto de una variable, x_i, es decir $\frac{\partial f}{\partial x_i}$ se calcula como:

$$\frac{\partial f}{\partial x} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x, y) - f(x,y)}{\Delta x} \tag{2.7}$$

2.3.3 Vector gradiente

2.3.4 Ejercicios

Exercise 2.3 Determine la derivada de las siguientes funciones. ■

Exercise 2.4 Encuentre el gradiente de la función f, evalúe el gradiente en el punto indicado p_0, encuentre la razón de cambio de la función en el punto p_0 pero usando el vector u.

$$f(x,y) = 7xy^2 - 2x^2y$$

$$p_0 = (1,1)$$

$$u = [3/13, 10/13]$$

■

2.3.5 Lecturas recomendadas

- Capítulos 2 y 3. Stewart, James. Cálculo De Una Variable: Trascendentes Tempranas. 4a. ed. México D.F.: Thomson, 2001.

2.4 Probabilidad y estadística

La probabilidad y la estadística son fundamentales en el aprendizaje profundo ya que proporcionan un marco teórico para modelar la incertidumbre inherente a los datos y a los modelos de inferencia, permitiendo la estimación de parámetros, la validación rigurosa de modelos, la evaluación de su rendimiento y la realización de inferencias fundamentadas sobre los fenómenos subyacentes.

2.4.1 Probabilidad

La probabilidad es una medida numérica que representa la posibilidad o verosimilitud de que ocurra un evento específico dentro de un conjunto de eventos posibles. Siguiendo la escuela frecuentista, es la relación entre el número de resultados favorables (eventos que satisfacen cierta condición) y el

número total de resultados posibles en un experimento o situación aleatoria. Se expresa típicamente como un valor en el intervalo $[0,1]$, donde 0 indica que el evento es imposible de ocurrir y 1 indica que es seguro que ocurra. Valores intermedios representan grados de certeza o incertidumbre en la ocurrencia del evento.

El conjunto de todos los posibles resultados de un experimento estadístico se llama espacio muestral y se representa por S [21]. Es como el "universo" de posibilidades en el que se desarrolla un evento probabilístico. Por ejemplo, si estás lanzando un dado, el espacio muestral sería $\{1,2,3,4,5,6\}$, ya que esos son todos los resultados posibles. Un evento es un subconjunto del espacio muestral que describe un resultado específico o una combinación de resultados que nos interesa analizar en un experimento probabilístico. Puede ser tan simple como un solo resultado o tan complejo como una combinación de múltiples resultados. Por ejemplo, si estás lanzando un dado y te interesa obtener un número par, el evento sería $\{2,4,6\}$. Dos eventos A y B son mutuamente excluyentes, o disjuntos, si $A \cap B = \emptyset$, es decir, si A y B no tienen elementos en común. Cada punto muestral de un evento tiene asociado un peso que se define como las veces que puede ocurrir dentro del total de elementos en el espacio muestral.

La probabilidad de un evento A se define como la suma de los pesos de todas las muestras en A. Por lo tanto, $0 \leq P(A) \leq 1$, $P(\emptyset) = 0$, y $P(S) = 1$. Además, si $A_1, A_2, A_3, \ldots$ es una secuencia de eventos mutuamente excluyentes, entonces la probabilidad de que ocurra uno de ellos es $P(A_1 \cup A_2 \cup A_3 \cup \ldots) = P(A_1) + P(A_2) + P(A_3) + \ldots$.

A menudo es más fácil calcular la probabilidad de algún evento a partir de las probabilidades conocidas de otros eventos siguiendo algunas reglas. Si A y B son dos eventos, entonces $P(A \cup B) = P(A) + P(B) - P(A \cap B)$. Si A y B son mutuamente excluyentes, entonces $P(A \cup B) = P(A) + P(B)$.

2.4.2 Probabilidad condicional

2.4.3 Regla de Bayes

2.4.4 Variable aleatoria

Una variable aleatoria es aquella variable que puede tomar diferentes valores aleatoriamente y esta es una función que asigna un valor numérico a cada resultado de un experimento aleatorio. Las variables aleatorias se pueden clasificar en dos tipos:

- Variables aleatorias continuas: Son variables aleatorias que pueden tomar cualquier valor dentro de un intervalo.
- Variables aleatorias discretas: Son variables que pueden tomar valores de un numero finito o infinito numerable.

Para generar en Python un número flotante aleatorio entre 0.0 y 1.0, puedes usar la función random() del módulo random:

```
import random

numero_aleatorio = random.random()
print(numero_aleatorio)
```

2.4.5 Distribución de probabilidad

La distribución de probabilidad es una función que asigna una probabilidad a cada valor posible de una variable aleatoria; es decir, describe como se distribuyen los valores de una variable aleatoria.

Distribución de probabilidad discreta

A la distribución de probabilidad sobre variables aleatorias discretas se le puede describir haciendo uso de una función de masa de probabilidad (PMF, por sus siglas en ingles). Esta función se representa con la letra P y es la encargada de mapear de un estado de la variable aleatoria a la

probabilidad de que esa variable toma en ese estado, la probabilidad de una variable x es denotada por $P(x)$. Una PMF debe satisfacer las siguientes condiciones:

- El dominio de P debe ser el conjunto de todos los posible estados de x.
- Un evento imposible tiene probabilidad cero y ningun estado puede ser menos probable que eso.
- La suma de las probabilidades de todos los estados es igual a uno. Algunos se le refieren como ser normalizada.

Distribución de probabilidad contínua

A la distribución de probabilidad sobre variables aleatorias continuas se le puede describir haciendo uso de una función de densidad de probabilidad, p,(PDF, por sus siglas en ingles). Esta funcion no describe la probabilidad de un estado directamente, sino la probabilidad de que esta caiga en una region infinitesimal con volumen δx es dada por $p(x)\delta x$. Una PDF debe satisfacer las siguientes condiciones:

- El dominio de p debe ser el conjunto de todos los posible estados de x.
- Todos los estados tienen una probabilidad mayor o igual a cero y no en necesario que $p(x)$ sea menor o igual a uno.
- Su integral en todo el rango de valores posibles de la variable aleatoria debe ser igual a uno

Distribución uniforme: esta distribución hace que cada uno de los estados tenga una probabilidad igual.

2.4.6 Probabilidad marginal

Algunas veces se conoce la distribución de probabilidad sobre todo el conjunto de estados y necesitamos conocer la distribución de probabilidad solo sobre un subconjunto de estados. A la probabilidad sobre un subconjunto se le conoce como distribución de probabilidad marginal. En otras palabras, la probabilidad marginal es la probabilidad de que ocurra un evento específico, sin tener en cuenta la probabilidad de que ocurran otros eventos en conjunto con él. La probabilidad marginal se puede calcular sumando, o integrando el volumen (según sea el caso), las probabilidades conjuntas de todos los eventos que pueden ocurrir en conjunto con el evento específico.

2.4.7 Ejercicios

Exercise 2.5 Considerando la fecha actual. Suponga un sistema que es capaz de predecir el año de nacimiento de una persona que máximo tiene 120 años. Describa el espacio muestral del sistema. ■

Exercise 2.6 Supongamos que en una población, el 7% de las personas tienen influenza. Se sabe que la prueba utilizada para detectarla tiene una tasa de falsos positivos del 2% y una tasa de falsos negativos del 1%. Juan se realiza la prueba y el resultado es positivo. ¿Cuál es la probabilidad de que realmente tenga la enfermedad? Tome en cuenta que lo anterior implica: $P(\text{Enfermedad}) = 0.07$, $P(\text{No Enfermedad}) = 0.93$, $P(\text{Positivo}|\text{No Enfermedad}) = 0.02$ y $P(\text{Negativo}|\text{Enfermedad}) = 0.01$. ■

2.5 Programación en Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, famoso por su legibilidad y sintaxis clara, que favorece un código más limpio y comprensible. Desde su creación en 1991 por Guido van Rossum, Python se ha expandido más allá de sus humildes comienzos para convertirse en uno de los pilares de la programación moderna, especialmente en los campos de la ciencia de datos y la inteligencia artificial (IA).

La importancia de Python en la ciencia de datos y la IA se debe a varias razones clave. En primer lugar, su sintaxis sencilla y legible hace que sea accesible para profesionales de diversos campos, no solo para programadores experimentados. Esto ha permitido a un amplio espectro de científicos, investigadores y analistas de datos adoptar Python para el análisis y la visualización de datos, sin tener que invertir tiempo considerable en aprender las complejidades de la programación.

Además, Python se ha beneficiado enormemente de una comunidad activa y comprometida que ha desarrollado una vasta colección de bibliotecas y frameworks especializados. En el ámbito de la ciencia de datos, bibliotecas como NumPy, pandas, Matplotlib y SciPy han simplificado el manejo de grandes conjuntos de datos, el análisis numérico y la visualización de datos. En el terreno de la IA y el aprendizaje automático, bibliotecas como TensorFlow, Keras y PyTorch ofrecen herramientas potentes y flexibles para la construcción y el entrenamiento de modelos de IA, desde redes neuronales simples hasta sistemas complejos de aprendizaje profundo.

La versatilidad de Python también juega un papel crucial en su popularidad. Puede ser utilizado para desarrollar prototipos rápidos y también en aplicaciones de producción a gran escala, lo que lo hace ideal para la investigación y el desarrollo en ciencia de datos e IA, donde la experimentación y la iteración rápida son comunes. La capacidad de Python para integrarse con otros lenguajes y tecnologías facilita la implementación de soluciones complejas que requieren la combinación de bibliotecas y herramientas específicas del dominio.

En resumen, Python se ha convertido en un lenguaje fundamental para la ciencia de datos y la inteligencia artificial debido a su simplicidad, su amplio ecosistema de bibliotecas especializadas, y su comunidad activa y de apoyo. Estos factores han colaborado para que Python sea no solo accesible para los principiantes, sino también lo suficientemente poderoso para los profesionales y investigadores que buscan empujar los límites de lo posible en sus campos.

2.5.1 Numpy

NumPy, que es la abreviatura de Numerical Python, es un paquete fundamental en el ecosistema de Python, especialmente diseñado para realizar operaciones numéricas y matemáticas de manera eficiente. Este paquete se ha convertido en la piedra angular de muchas aplicaciones científicas y de ingeniería debido a su capacidad para trabajar con arreglos y matrices de datos a gran velocidad, superando con creces las capacidades de las listas de Python estándar en términos de rendimiento y funcionalidad.

El corazón de NumPy es el objeto ndarray, que representa una colección homogénea y multi-dimensional de elementos (todos del mismo tipo), permitiendo realizar operaciones matemáticas complejas sobre grandes volúmenes de datos con relativa facilidad y eficiencia. A diferencia de las listas en Python, que pueden contener elementos de diferentes tipos y tienen una flexibilidad inherente, los arreglos de NumPy están optimizados para cálculos numéricos pesados y operaciones vectorizadas que aplican una operación sobre todos los elementos del arreglo sin la necesidad de bucles explícitos.

Además de su poderoso objeto de arreglo, NumPy ofrece un vasto conjunto de funciones matemáticas que permiten realizar desde operaciones aritméticas básicas hasta cálculos matemáticos más complejos como transformadas de Fourier, álgebra lineal, y generación de números aleatorios. Esta amplia gama de herramientas matemáticas, junto con la capacidad de integrarse sin problemas con otras bibliotecas científicas de Python, hace de NumPy un componente esencial en el análisis de datos, la modelización científica, la simulación, y mucho más.

Funcionalidad común

En NumPy, un array se define como una estructura de datos central que permite almacenar y manipular grandes conjuntos de datos numéricos de manera eficiente. Esta estructura es conocida como ndarray (abreviatura de "N-dimensional array"), que representa una colección homogénea de elementos; es decir, todos los elementos deben ser del mismo tipo. Los arrays de NumPy son

n-dimensionales, lo que significa que pueden representar no solo vectores y matrices (1D y 2D respectivamente), sino también estructuras de datos de mayor dimensionalidad.

Para definir un array en NumPy, primero se debe importar la biblioteca NumPy, típicamente con el alias np para facilitar su uso. Luego, se puede utilizar la función np.array() para crear un array, pasando una lista (o una lista de listas para dimensiones superiores) de elementos como argumento. Esta lista define los contenidos del array, y NumPy automáticamente determina el tipo de datos más adecuado para los elementos contenidos.

Por ejemplo, para crear un simple array unidimensional, se pasaría una única lista de números. Para un array bidimensional, que podría representar una matriz matemática, se pasaría una lista de listas, donde cada sublista representa una fila de la matriz. NumPy ofrece una gran flexibilidad en la definición de arrays, permitiendo especificar explícitamente el tipo de datos de los elementos mediante el argumento dtype si se desea un control más fino sobre el comportamiento del array.

Además de np.array(), NumPy proporciona una serie de funciones para generar arrays de formas particulares y con contenidos predefinidos. Por ejemplo, np.zeros() crea un array lleno de ceros, np.ones() un array de unos, y np.arange() genera un array con una secuencia de números dentro de un rango especificado. Estas funciones son especialmente útiles para crear arrays que sirven como punto de partida para cálculos más complejos.

Una vez definido, el array de NumPy se puede manipular mediante una amplia gama de operaciones matemáticas y lógicas, permitiendo realizar desde simples sumas hasta operaciones más complejas como multiplicaciones de matrices o transformadas de Fourier, todo ello de manera muy eficiente gracias a la optimización interna de NumPy y a su capacidad para operar sobre todos los elementos del array de manera simultánea (un proceso conocido como vectorización).

2.5.2 Ejercicios

Exercise 2.7 Escriba un programa usando numpy que obtenga la transpuesta de una matriz sin usar una función preestablecida. ■

2.6 Optimización

La optimización es el proceso de encontrar los valores más efectivos o eficientes para ciertas variables bajo un conjunto de restricciones, con el objetivo de maximizar o minimizar una función específica. Esta función podría representar diversos objetivos, como minimizar el costo, maximizar la eficiencia, o reducir el error.

En el contexto de las redes neuronales, la optimización se centra en ajustar los parámetros de la red para mejorar su rendimiento en tareas específicas, como la clasificación o la predicción. Cada red neuronal se compone de múltiples capas y conexiones, y cada conexión tiene un peso asociado que determina su importancia en el cálculo de la salida de la red. El papel de la optimización es encontrar el conjunto de pesos que minimice la diferencia entre las predicciones del modelo y los valores reales, es decir, minimizar la función de pérdida.

Este proceso de ajuste se realiza comúnmente a través de algoritmos como el descenso del gradiente, donde los pesos se ajustan iterativamente en la dirección que más reduce la función de pérdida. A medida que el algoritmo de optimización procesa los datos de entrenamiento, los pesos se van refinando de manera que el modelo se vuelve cada vez más preciso en sus predicciones, lo que efectivamente "entrena" a la red para realizar su tarea de forma más eficaz. Este proceso de entrenamiento y optimización es fundamental para que las redes neuronales funcionen correctamente y cumplan con las expectativas en aplicaciones prácticas.

2.6.1 Conceptos de optimización

La función de interés se denomina función objetivo y usualmente se representa con la letra f.

Definition 2.6.1 — Punto estacionario. Un punto estacionario de una función es aquel en el cual la derivada (o el gradiente en el caso de varias variables) es igual a cero. Es decir, en ese punto la función no tiene cambios instantáneos en su valor (no crece ni decrece localmente). Para f un punto estacionario es aquel a donde:

$$f'(a) = 0$$

Definition 2.6.2 — Mímino local. Un mínimo local es aquel punto estacionario cuya evaluación es la menor en su vecindario. Es decir, a es un mínimo local si:

$$f(a) < f(a \pm \Delta)$$

donde $\Delta > 0$ y Δ toma valores dentro del vecindario delimitado por u, es decir $\Delta \leq u$.

Definition 2.6.3 — Mínimo global. Un mínimo global es un punto estacionario en el cual el valor de la función es menor al valor de la función en cualquier punto del dominio. Es decir, a es un mínimo global si:

$$f(a) < f(b) \forall b \neq a$$

Tanto el mínimo local como el mínimo global son conceptos análogos a el máximo global y local; en ambos casos, los máximos son puntos que tienen valores mayores a sus vecinos. Tales definiciones no se formalizan en este libro dado que en área de aprendizaje automático, el principal objetivo es la minimización del error.

2.7 Descenso por gradiente

El descenso por gradiente es un algoritmo de optimización utilizado en el campo del aprendizaje automático y en otros campos relacionados con la optimización de funciones. Su objetivo es encontrar los valores de los parámetros de una función que minimicen (o maximicen) su valor.

Supongamos que deseamos minimizar una función objetivo f. Es decir, queremos encontrar el valor del parámetro x^* que nos provea la mínima evaluación:

$$x^* = \arg\min f(x) \tag{2.8}$$

El descenso por gradiente especifica que de forma iterativa podemos hallar el valor del parámetro a través de actualizar un valor candidato. Formalmente,

Definition 2.7.1 — Actualización con descenso por gradiente. Sea x_{t-1} un valor candidato, η un factor de escala (en el ámbito del aprendizaje, se denomina tasa de aprendizaje) y $\nabla_x f(x)$ el gradiente de la función $f(x)$ con respecto de x. El nuevo valor candidato se calcula como

$$x_t = x_{t-1} - \eta \nabla_x f(x) \tag{2.9}$$

En el caso particular donde solo tenemos un parámetro, el gradiente se puede definir como la derivada unidimensional,

$$\nabla_x f(x) = \frac{\delta}{\delta x} f(x) \tag{2.10}$$

Para las situaciones donde se tienen más parámetros, el gradiente se calcula con las derivadas parciales con respecto de cada parámetro. Como se verá después en la aplicación a las redes neuronales. Para comprender mejor el método partiremos de un ejemplo.

■ **Example 2.1 — Descenso por gradiente a una iteración.** Suponiendo la siguiente función objetivo a minimizar:

$$f(x) = x^2 \tag{2.11}$$

Determine el valor del parámetro x, a una iteración del descenso por gradiente usando un valor candidato, $x = -2$, y una tasa de aprendizaje, $\eta = 0.1$.
Solucion: Para esta función, el gradiente es:

$$\frac{\delta}{\delta x} f(x) = 2x \tag{2.12}$$

De esta forma, la actualización iterativa, ecuación (2.9), se escribe como:

$$x_t = x_{t-1} - \eta 2x_{t-1} \tag{2.13}$$

Utilizando el valor candidato, $x = -2$, y la tasa de aprendizaje, $\eta = 0.1$, el nuevo valor se determina como:

$$x_t = (-2) - 0.1(2(-2)) = -1.6 \tag{2.14}$$

De lo anterior podemos observar que el nuevo valor es mejor dado que disminuye la evaluación de la función. Es decir, $f(-1.6) < f(-2)$. ■

■ **Example 2.2 — Descenso por gradiente a varias iteraciones.** Partiendo la ecuación x^2 del ejemplo 2.1. Suponga ahora que el valor inicial es $x = 5$ y que la tasa de aprendizaje $\eta = 0.1$.

1. Calcule los valores del parámetro para 5 iteraciones consecutivas. SOLUCION: Los valores se encuentran escritos en la tabla siguiente.

x_t	$f(x_t)$	x_{t+1}
5	25	4
4	16	3.2
3.2	10.2	2.56
2.56	6.55	2.04
2.04	4.19	1.63

2. Implemente un programa en python y grafique los resultados del método para 10 iteraciones. SOLUCIÓN: Observe la figura 2.1

■

2.7.1 Implementación

El método de descenso por gradiente consiste en ejecutar repetidamente la ecuación (2.9). Para esta implementación estamos suponiendo que la función solo tiene una variable, por lo cual solo se usa una derivada univariable. Por lo anterior, la implementación de la ecuación (2.9) se realiza en una función de Python mostrada en el código 2.1. La función recibe el valor actual del parámetro y la tasa de aprendizaje.

La función de actualización se prueba en el código 2.2 para una función objetivo cuadrática.

2.7.2 Ejercicios

Exercise 2.8 Calcule de forma iterativa 6 valores del parámetro x para minimizar la función:

$$f(x) = 2x + x^2$$

Use el parámetro inicial $x = 10$ y una tasa $\eta = 0.2$ ■

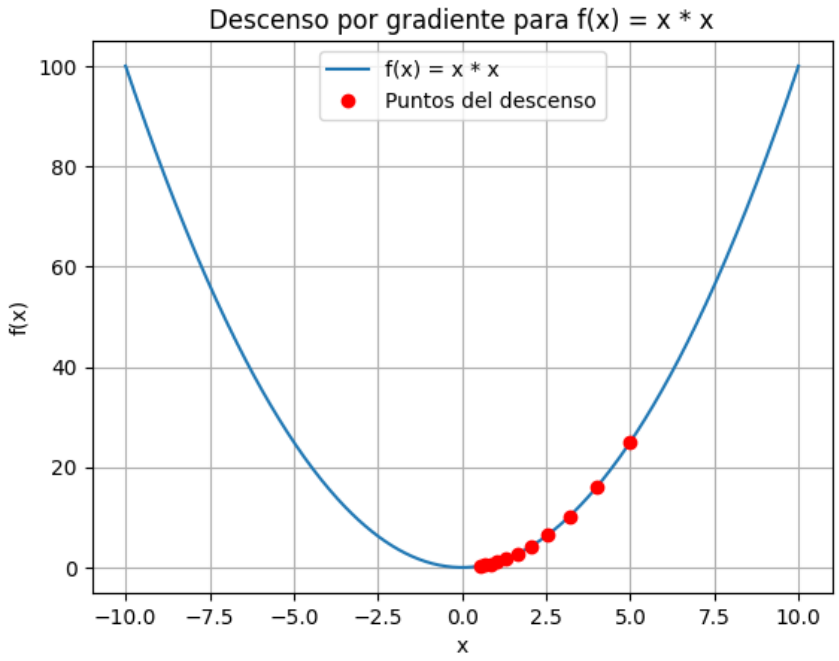


Figure 2.1: Valores obtenidos del descenso por gradiente.

```
# descenso por gradiente:
def descenso_por_gradiente(parametro, tasa):
    parametro = parametro - tasa * df(parametro)
    return parametro
```

Table 2.1: Implementación del descenso por gradiente unidimensional en Python.

Exercise 2.9 Modifique el código 2.2 para que se minimice la función descrita en el ejercicio 2.8. Grafique los valores calculados para 20 iteraciones del método. ■

```python
#funcion objetivo:
def f(x):
    return x**2

# derivada de la funcion objetivo:
def df(x):
    return 2*x

# funcion principal:
def main(iterations):
    print("Introduccion al descenso por gradiente")

    parametro = 5
    tasa = 0.1

    print(f"Valor inicial x: {parametro}")
    print(f"Valor de f(x): {f(parametro)}")

    for i in range(iterations):
        parametro = descenso_por_gradiente(parametro, tasa)
        print(f"\nIteracion: {i+1}")
        print(f"Valor de x: {parametro}")
        print(f"Valor de f(x): {f(parametro)}")


if __name__ == "__main__":
    iterations = int(input("Introduce la cantidad de iteraciones: "))
    if iterations <= 0:
        print("El numero de iteraciones debe ser mayor a 0")
    else:
        main(iterations)
```

Table 2.2: Prueba de la implementación del descenso por gradiente.