Fundamentos de las Redes Neuronales Profundas

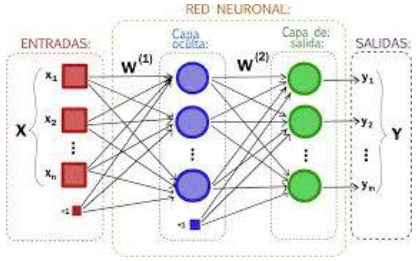
Fundamentos Matemáticos Básicos II

Clase 3

Dra. Wendy Aguilar

Modelos Generativos Profundos

UN ENFOQUE DESDE LA CREATIVIDAD COMPUTACIONAL



1. Vectores, Matrices y Tensores numéricos

Son el lenguaje en el que las redes neuronales “piensan” y realizan cálculos.

Los bloques fundamentales para representar y manipular datos en DL son:

- Vectores
- Matrices
- Tensores

Google colab

Tutorial_uso_tensores_con_numpy.ipynb
Ejercicios_vectores_matrices_tensores_con_Numpy1.ipynb
Ejercicios_vectores_matrices_tensores_con_Numpy2.ipynb

Vectorización

$$u = \begin{matrix} u_1 \\ \vdots \\ u_n \end{matrix} \qquad v = \begin{matrix} v_1 \\ \vdots \\ v_n \end{matrix}$$

$$w = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$$

```
import numpy as np
n =  1000000
u = np.random.rand(n)
v = np.random.rand(n)
```

```
import time
```

```
# 1) Con un ciclo for
inicio = time.time()
w=0
for i in range(n):
  w+= u[i]*v[i]
fin = time.time()
print("Version for: ", str(1000*(fin-inicio)) + "ms")
print(w)
```

```
# 2) Versión vectorizada
inicio = time.time()
w = np.dot(u,v)
fin = time.time()
print("Version vectorizada: ", str(1000*(fin-inicio)) + "ms")
print(w)
```

Google colab

vectorizacion.ipynb

Ejercicio 1

Vectorización

$$u = \begin{matrix} u_1 \\ \vdots \\ u_n \end{matrix} \qquad v = \begin{matrix} v_1 \\ \vdots \\ v_n \end{matrix}$$

$$w = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$$

```python
import numpy as np
n =  1000000
u = np.random.rand(n)
v = np.random.rand(n)
```

```python
import time
```

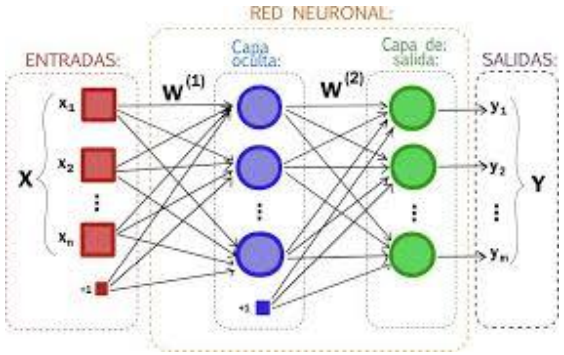
```python
# 1) Con un ciclo for
inicio = time.time()
w=0
for i in range(n):
  w+= u[i]*v[i]
fin = time.time()
print("Version for: ", str(1000*(fin-inicio)) + "ms")
print(w)
```

```python
# 2) Versión vectorizada
inicio = time.time()
w = np.dot(u,v)
fin = time.time()
print("Version vectorizada: ", str(1000*(fin-inicio)) + "ms")
print(w)
```

```
Version for:  479.0992736816406ms
249816.30215526323
Version vectorizada:  1.3687610626220703ms
249816.30215527286
```

¡Aprox. 350 veces más rápido!

Vectorización



Tiempo con método lento	Tiempo con método rápido
1 minuto	0.17 segundos
10 minutos	1.70 segundos
30 minutos	5.12 segundos
1 hora	10.23 segundos
2 horas	20.46 segundos
5 horas	51.09 segundos
10 horas	1 min 42 segundos
24 horas	4 min 5 segundos

La cantidad de operaciones con tensores que se realizan al entrenar un modelo generativo es **enorme**,
y su orden de magnitud **depende** del tipo de modelo, el tamaño de los datos, y la arquitectura.

Vectorización

Moraleja

Siempre que puedas evita usar ciclos for explícitamente

$$np.\log(v)$$
$$np.abs(v)$$
$$np.\text{maximum}(v, 0)$$
$$v ** 2$$
$$1/v$$

En NumPy, casi todas las ufuncs (universal functions) están diseñadas para operar:

- Con tensores de cualquier dimensión.
- De manera vectorizada (sin bucles explícitos).
- Usando broadcasting cuando las formas no coinciden pero son compatibles.

Vectorización

Cantidad de carbohidratos, proteínas y grasas en 100g :

	Manzanas	Carne	Huevos	Papas
Carbohidratos	56.0	0.0	4.4	68.0
Proteínas	1.2	104.0	52.0	8.0
Grasas	1.8	135.0	99.0	0.9

Calcula el porcentaje de calorías que aportan los carbohidratos, las proteínas y las grasas por cada comida.

Vectorización y Broadcasting

Cantidad de carbohidratos, proteínas y grasas en 100g :

	Manzanas	Carne	Huevos	Papas
Carbohidratos	56.0	0.0	4.4	68.0
Proteínas	1.2	104.0	52.0	8.0
Grasas	1.8	135.0	99.0	0.9
Total calorías	59.0	239.0	155.4	76.9
% Carbohidratos	56.0/59=0.94			
% Proteínas	1.2/59=0.02%			
% Grasas	1.8/59=0.03%			

Calcula el porcentaje de calorías que aportan los carbohidratos, las proteínas y las grasas por cada comida.

$$\begin{bmatrix} 0.94 & 0.0 & 0.02 & 0.88 \\ 0.02 & 0.43 & 0.33 & 0.10 \\ 0.03 & 0.56 & 0.63 & 0.01 \end{bmatrix}$$

¿Lo puedes hacer sin un ciclo *for* explícito?

Google colab

vectorizacion.ipynb
Ejercicio 2

Vectorización y Broadcasting

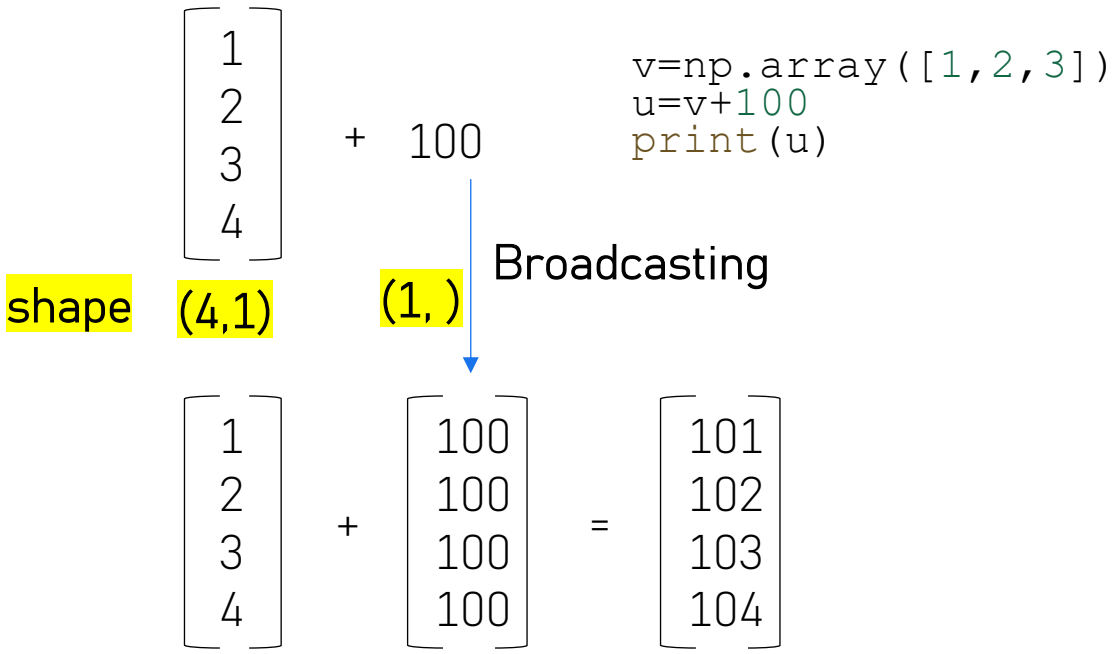
$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + 100$$

```
v=np.array([1,2,3])
u=v+100
print(u)
```

¿Se puede hacer?

Vectorización y Broadcasting

Es el conjunto de reglas que permiten realizar **operaciones elemento a elemento entre arreglos** (ndarrays) **de diferentes formas** (shape) sin tener que copiarlos o expandirlos explícitamente.




```
v=np.array([1,2,3])
u=v+100
print(u)
```

Las **reglas** son muy precisas y siempre se aplican en el orden de las dimensiones, **de derecha a izquierda**.

1. Alineación de dimensiones
·Compara las formas (shape) de los dos arreglos empezando por la última dimensión hacia la primera.
·Si una de las formas es más corta, se rellena con 1s a la izquierda para igualar la longitud.

2. Compatibilidad de dimensiones
En cada posición de dimensión (de derecha a izquierda), dos dimensiones son compatibles si:
·Son iguales, o
·Una de ellas es 1.

3. Expansión implícita
Si una dimensión es 1 y la otra mayor que 1, NumPy repite los datos a lo largo de esa dimensión para que coincidan (sin crear copias reales).

4. Si alguna dimensión no es compatible, se lanza un ValueError.

Vectorización y Broadcasting

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 & 20 & 30 \end{bmatrix}$$

shape (2,3) (1,3)

¿Se puede hacer?

```
v=np.array([ [1,2,3],
             [4,5,6]
           ])
u=np.array([10,20,30])
print(u+v)
```

Vectorización y Broadcasting

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 & 20 & 30 \end{bmatrix}$$

Broadcasting

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 & 20 & 30 \\ 10 & 20 & 30 \end{bmatrix} + \begin{bmatrix} 11 & 22 & 33 \\ 14 & 25 & 36 \end{bmatrix}$$

¿Se puede hacer?

```
v=np.array([ [1,2,3],
             [4,5,6]
           ])
u=np.array([10,20,30])
print(u+v)
```

Vectorización y Broadcasting

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 \\ 20 \end{bmatrix}$$

shape (2,3) (2,1)

¿Se puede hacer?

```
v=np.array([ [1,2,3],
             [4,5,6]
           ])
u=np.array([10,20])

print(u+v)
```

Vectorización y Broadcasting

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 \\ 20 \end{bmatrix}$$

Broadcasting

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 10 & 10 & 10 \\ 20 & 20 & 20 \end{bmatrix} = \begin{bmatrix} 11 & 12 & 13 \\ 24 & 25 & 26 \end{bmatrix}$$

¿Se puede hacer?

```
v=np.array([ [1,2,3],
             [4,5,6]
           ])
u=np.array([10,20]).reshape(2,1)

print(u+v)
```

Vectorización

$$u = \begin{matrix} u_1 \\ \vdots \\ u_n \end{matrix} \qquad v = \begin{matrix} v_1 \\ \vdots \\ v_n \end{matrix}$$

$$w = u_1 v_1 + u_2 v_2 + \cdots + u_n v_n$$

```
import numpy as np
n =  1000000
u = np.random.rand(n)
v = np.random.rand(n)
```

```
import time
```

¿Se ejecutó en cpu o gpu?

```
# 1) Con un ciclo for
inicio = time.time()
w=0
for i in range(n):
  w+= u[i]*v[i]
fin = time.time()
print("Version for: ", str(1000*(fin-inicio)) + "ms")
print(w)
```

```
# 2) Versión vectorizada
inicio = time.time()
w = np.dot(u,v)
fin = time.time()
print("Version vectorizada: ", str(1000*(fin-inicio)) + "ms")
print(w)
```

```
Version for:  479.0992736816406ms
249816.30215526323
Version vectorizada:  1.3687610626220703ms
249816.30215527286
```

¡Aprox. 350 veces más rápido!

- No puede ejecutar operaciones de arreglos en GPU.
- Optimizado para CPUs

- No puede ejecutar operaciones de arreglos en GPU.
- Optimizado para CPUs

Preinstalado en Google Colab

Biblioteca desarrollada por **Google Brain**, diseñada para construir, entrenar y desplegar **modelos de aprendizaje automático** y, especialmente, **redes neuronales profundas (deep learning)**.

Es una de las herramientas más poderosas y ampliamente adoptadas para crear desde prototipos hasta aplicaciones en producción.

Optimizado para GPUs y TPUs

TensorFlow incluye:

Interfaz de alto nivel para definir modelos fácilmente.

Provee herramientas para visualizar métricas durante el entrenamiento (como la pérdida y la exactitud, visualizar el grafo del modelo, ver histogramas de pesos, sesgos y otros tensores a medida que cambian con el tiempo.

Tensores en TensorFlow

```
import tensorflow as tf
```

2 tipos

Constantes

Datos (tensores) que NO cambian durante toda la ejecución:

- Datos de entrada
- Parámetros fijos del modelo.
- Hiperparámetros numéricos (ej. temperatura en softmax, coeficientes de regularización).
- Operaciones matemáticas (como sumas, multiplicaciones, etc., que generan nuevos tensores en cada paso)

Nota: Los maneja con mayor eficiencia TensorFlow

Variables

Datos (tensores) que SI cambian durante toda la ejecución:

- Pesos entrenables:.
- Parámetros que se actualizan:
- Variables de estado en capas como BatchNorm (media/varianza acumulada).
- Contadores o acumuladores que cambian entre pasos.

Tensores en TensorFlow

```
import tensorflow as tf
```

2 tipos

Creación de tensores

Constantes

```
t = tf.constant(5)

t = tf.constant([1,2,3,4,5])

t = tf.constant([[1,2],
                 [3,4]])

t = tf.constant([[[1,2],
                  [3,4],
                  [5,6]
                 ],
                 [[7,8],
                  [9,10],
                  [11,12]
                 ]
                ])
```

Variables

```
t = tf.Variable(5)

t = tf.Variable([1,2,3,4,5])

t = tf.Variable([[1,2],
                 [3,4]])

t = tf.Variable([[[1,2],
                  [3,4],
                  [5,6]
                 ],
                 [[7,8],
                  [9,10],
                  [11,12]]
                ])
```

Tensores en TensorFlow

```
import tensorflow as tf
```

2 tipos

Creación de tensores

Inter operatividad con Numpy

Constantes

```
# tf.Tensor -> np.ndarray
t = tf.constant([[1., 2.], [3., 4.]])
n = t.numpy()

# np.ndarray -> tf.Tensor
m = np.array([[5., 6.], [7., 8.]])
t2 = tf.convert_to_tensor(m)
```

Variables

```
# tf.Tensor -> np.ndarray
t = tf.Variable([[1., 2.], [3., 4.]])
n = t.numpy()

# np.ndarray -> tf.Tensor
m = np.array([[5., 6.], [7., 8.]])
t2 = tf.convert_to_tensor(m)

v = tf.Variable(m)
```

Tensores en TensorFlow

```
import tensorflow as tf
```

2 tipos

Creación de tensores

Funciones de inicialización

Constantes

```
t = tf.zeros([2,3])

t = tf.ones([2,3,4])

t = tf.fill([2,3], value=5)

t = tf.random.uniform(shape=(2, 3))

t = tf.range(start=0, limit=10, delta=2)
```

Variables

```
v = tf.Variable(m)
```

Todos los inicializadores crean tensores inmutables → si quieres que sean variables, debes envolverlos en un tf.Variable.

Tensores en TensorFlow

```
import tensorflow as tf
```

2 tipos

Constantes

```
a = tf.reshape(tf.range(24), (2,3,4))
print('a[0, 1]:', a[0, 1])
```

Variables

```
v = tf.Variable([[1, 2, 3],
                 [4, 5, 6]])

print(v[0, 1])

# Modificar un elemento
v[0, 1].assign(99)
```

Acceso a elementos de tensores

- **Indexado por posición** → `t[filas, columnas]`
- **Slicing** → `t[inicio:fin:paso, ...]`
- **Indices negativos** → `t[-1]` accede a la última fila
- **Selección de múltiples índices** con listas → `t[[0, 2]]`
- **Saltos** → `t[:, ::2]`

Tensores en TensorFlow

Concepto	NumPy	TensorFlow	Ejemplo NumPy	Ejemplo TensorFlow	Resultado
Tipo de objeto	`type(a)`	`type(t)`	`type(np.array([1,2,3]))`	`type(tf.constant([1,2,3]))`	`<class 'numpy.ndarray'>` / `<class 'EagerTensor'>`
Tipo de dato (dtype)	`a.dtype`	`t.dtype`	`np.array([1,2,3]).dtype`	`tf.constant([1,2,3]).dtype`	`int64` / `tf.int32`
Número de dimensiones	`a.ndim`	`t.ndim` o `len(t.shape)`	`np.array([[1,2],[3,4]]).ndim`	`tf.constant([[1,2],[3,4]]).ndim`	`2`
Forma (shape)	`a.shape`	`t.shape`	`np.array([[1,2],[3,4]]).shape`	`tf.constant([[1,2], [3,4]]).shape`	`(2, 2)`
Número total de elementos	`a.size`	`tf.size(t)` o `tf.size(t).numpy()`	`np.array([1,2,3,4]).size`	`tf.size(tf.constant([1,2,3,4])).numpy()`	`4`
Valor mínimo	`a.min()`	`tf.reduce_min(t)`	`np.array([1,2,3]).min()`	`tf.reduce_min(tf.constant([1,2,3])).numpy()`	`1`
Valor máximo	`a.max()`	`tf.reduce_max(t)`	`np.array([1,2,3]).max()`	`tf.reduce_max(tf.constant([1,2,3])).numpy()`	`3`
Promedio	`a.mean()`	`tf.reduce_mean(t)`	`np.array([1,2,3]).mean()`	`tf.reduce_mean(tf.constant([1,2,3])).numpy()`	`2.0`

- TensorFlow maneja el broadcasting igual que NumPy

Tensores en TensorFlow

```
import tensorflow as tf
```

Crear una copia de Ejercicios_vectores_matrices_tensores_con_Numpy1.ipynb y modificar para usar tensores de TensorFlow en vez de tensores de NumPy

2 tipos

Constantes

Operaciones matemáticas

Variables

```
t = tf.constant([[1., 2.], [3., 4.]])
```

Comparación de operaciones entre NumPy y TensorFlow

Operación	NumPy	TensorFlow	Ejemplo NumPy	Ejemplo TensorFlow	Resultado
Suma	`np.add(a, b)` o `a + b`	`tf.add(a, b)` o `a + b`	`np.add([1, 2], [3, 4])`	`tf.add([1, 2], [3, 4])`	`[4, 6]`
Resta	`np.subtract(a, b)` o `a - b`	`tf.subtract(a, b)` o `a - b`	`np.subtract([5, 7], [2, 3])`	`tf.subtract([5, 7], [2, 3])`	`[3, 4]`
Multiplicación elemento a elemento	`np.multiply(a, b)` o `a * b`	`tf.multiply(a, b)` o `a * b`	`np.multiply([2, 3], [4, 5])`	`tf.multiply([2, 3], [4, 5])`	`[8, 15]`
Producto punto (vectores)	`np.dot(a, b)`	`tf.tensordot(a, b, axes=1)`	`np.dot([1, 2], [3, 4])`	`tf.tensordot([1, 2], [3, 4], axes=1)`	`11`
Producto matricial	`np.matmul(A, B)` o `A @ B`	`tf.matmul(A, B)` o `A @ B`	`np.matmul([[1, 2],[3, 4]], [[5, 6],[7, 8]])`	`tf.matmul([[1, 2],[3, 4]], [[5, 6],[7, 8]])`	`[[19, 22],[43, 50]]`
Transpuesta	`np.transpose(A)` o `A.T`	`tf.transpose(A)`	`np.transpose([[1, 2],[3, 4]])`	`tf.transpose([[1, 2],[3, 4]])`	`[[1, 3],[2, 4]]`

Tensores en TensorFlow

TensorFlow hace **automatic device placement**.

Si TensorFlow detecta una GPU,
por defecto intentará colocar todos los tensores y operaciones en GPU.

Si no encuentra GPU se colocarán en CPU.

Para verificar en dónde se guardó una cierta variable:

```
print(A.device)
```

```
/job:localhost/replica:0/task:0/device:CPU:0
/job:localhost/replica:0/task:0/device:GPU:0
```

Solo una vez al arrancar tu script o notebook:

```
if tf.config.list_physical_devices('GPU'): print("GPU disponible")
else: print("No hay GPU, se usará CPU")
```

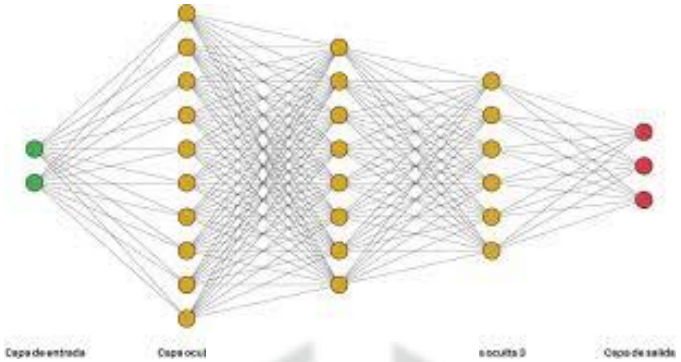
vectorizacion.ipynb
Ejercicio 3

```
N = 100_000_000
```

```
Version for:  32415.270566940308ms
NumPy dot (CPU):  96.90570831298828ms

TensorFlow: 2.19.0
Dispositivos físicos: [PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU'), PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
TF eager (CPU):  6.658554077148437ms
TF eager (GPU):  0.9670257568359375ms
```

¿Cuándo usar tensores de NymPy y cuándo de TensorFlow?



Pre-procesamiento

·**Lectura cruda e I/O:** cargar archivos (imágenes/audio/texto), parsing de JSON/CSV, tokenización, limpieza.

·**Transformaciones no diferenciables y de bajo costo**:

- ·reordenar listas, particionar train/val/test, cálculo de estadísticas del dataset.

·**Al final: convertir a tensores** (tf.convert_to_tensor, tf.data.Dataset.from_tensor_slices) o usar preprocessing layers de Keras.

Todo lo que participa en la pérdida y el gradiente:

Pesos entrenables:

- Matrices de pesos y vectores de sesgos en capas de redes neuronales,
- Parámetros de embeddings , convoluciones, atenciones, etc.

Parámetros que se actualizan:

- Variables de estado en capas como BatchNorm (media/varianza acumulada),
- Contadores o acumuladores que cambian entre pasos.

Post-procesamiento

·**Visualización y post-processing**: sample.numpy() → Matplotlib/PIL, guardar grids de imágenes, audios, etc.

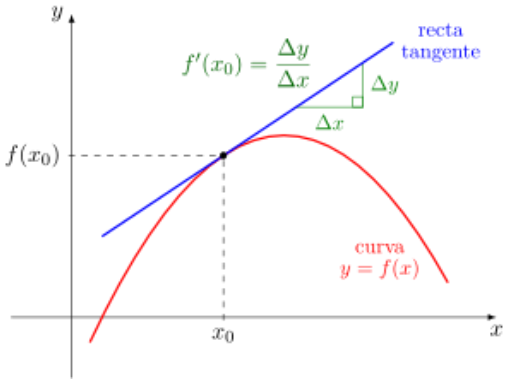
·**Métricas de desempeño no diferenciables o análisis offline**: FID/IS calculados como post-hoc, estadísticas, tablas, reportes.

·**Serialización fuera del grafo:** escribir a disco formatos propios (aunque para producción conviene SavedModel, tf.io).

- No tiene *automatic differentiation*, por lo que si quieres calcular gradientes necesitas hacerlo manualmente.

- Usando **tf.GradientTape**, calcula derivadas de manera automática y optimiza la ejecución.

¿Cuáles son los fundamentos matemáticos básicos que necesito saber para crear modelos generativos profundos?



2. Derivadas y Gradientes

Nos proporcionan el marco matemático necesario para comprender cómo ajustar los parámetros de los modelos de DL con el fin de minimizar los errores y mejorar su desempeño.

Derivada 1D

Definición intuitiva

La derivada de una función de una sola variable nos dice **qué tan rápido y en qué dirección está cambiando el valor de una función en un punto dado**.

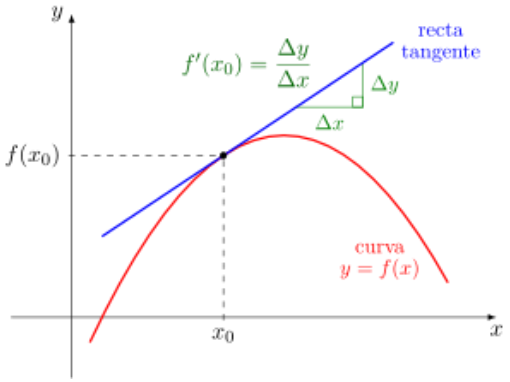
Geométricamente, representa la **pendiente de la línea tangente a la gráfica** de la función en un punto específico.

Notación

$$f'(x) \qquad \frac{dy}{dx} \quad \text{o} \quad \frac{df(x)}{dx}$$

Derivada de *y* con respecto a *x*

Cómo cambia *y* cuando varía *x*



2. Derivadas y Gradientes

Nos proporcionan el marco matemático necesario para comprender cómo ajustar los parámetros de los modelos de DL con el fin de minimizar los errores y mejorar su desempeño.

Derivada 1D

Definición intuitiva

La derivada de una función de una sola variable nos dice **qué tan rápido y en qué dirección está cambiando el valor de una función en un punto dado**.

Geométricamente, representa la **pendiente de la línea tangente a la gráfica** de la función en un punto específico.

Notación

$$f'(x) \qquad \frac{dy}{dx} \quad \text{o} \quad \frac{df(x)}{dx}$$

Derivada de *y* con respecto a *x*

Cómo cambia *y* cuando varía *x*

Interpretación en DL

Función de pérdida:

- Mide qué tan bien coinciden las predicciones de nuestro modelo con los valores reales en los datos de entrenamiento.
- Nuestro objetivo es **minimizar** esta función de pérdida.
- Los **pesos y sesgos** en la RN son los parámetros que podemos ajustar para lograr esta minimización.
- La **derivada** de la función de pérdida con respecto a un peso (o sesgo) nos indica **cuánto cambiaría la pérdida si hiciéramos un cambio pequeño en ese peso específico**.
  - **Derivada positiva**: Aumentar ligeramente el peso aumentaría la pérdida, y disminuirlo reduciría la pérdida.
  - **Derivada negativa**: Aumentar ligeramente el peso reduciría la pérdida, y disminuirlo la aumentaría.
  - **Derivada cercana a cero**: Cambiar el peso tendría un impacto mínimo en la pérdida de ese punto. Esto podría indicar un **mínimo local** o una **región plana** en el paisaje de la función de pérdida.

Cómo cambia *el error* cuando varía *el peso*.

2. Derivadas y Gradientes

Nos proporcionan el marco matemático necesario para comprender cómo ajustar los parámetros de los modelos de DL con el fin de minimizar los errores y mejorar su desempeño.

Gradiente (derivada multivariable)

Definición intuitiva

En una función con múltiples variables de entrada, el gradiente es un vector de todas las derivadas parciales de la función con respecto a cada una de sus variables de entrada.

Notación

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]$$

$$\nabla \text{ f(x,y,z,...)=[}\partial\text{x}\partial\text{f,}\partial\text{y}\partial\text{f,}\partial\text{z}\partial\text{f,...]}$$

2. Derivadas y Gradientes

Nos proporcionan el marco matemático necesario para comprender cómo ajustar los parámetros de los modelos de DL con el fin de minimizar los errores y mejorar su desempeño.

Gradiente (derivada multivariable)

Definición intuitiva

En una función con múltiples variables de entrada, el gradiente es un vector de todas las derivadas parciales de la función con respecto a cada una de sus variables de entrada.

Notación

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]$$

$$\nabla\ f(x,y,z,\ldots)=[\partial x\partial f,\partial y\partial f,\partial z\partial f,\ldots]$$

Interpretación en DL

Función de pérdida:

- Suele ser de múltiples variables (todos los pesos y bias de la RN).
- Sea W un vector de todos los pesos y biases, Entonces el gradiente de la función de pérdida con respecto a estos parámetros es un vector que contiene las derivadas parciales de la pérdida con respecto a cada peso individual y sesgo.

$$\nabla\ W,bL=[\partial w1\partial L,\partial w2\partial L,\ldots,\partial b1\partial L,\partial b2\partial L,\ldots]$$

- El gradiente apunta en la dirección del mayor ascenso de la función de pérdida.
- Esto significa que, si diéramos un pequeño paso en la dirección del gradiente, la pérdida aumentaría hacia el máximo,
- Por lo tanto, necesitamos movernos en la dirección opuesta al gradiente.

Descenso por gradiente en DL

2. Derivadas y Gradientes

Nos proporcionan el marco matemático necesario para comprender cómo ajustar los parámetros de los modelos de DL con el fin de minimizar los errores y mejorar su desempeño.

Gradiente (derivada multivariable)

- **Autoencoder variacional (VAE) pequeño:**
  decenas de miles a algunos millones de parámetros.
- **GAN básica (DCGAN, 64x64 imágenes):**
  unos 10–50 millones de parámetros.
- **StyleGAN2 / BigGAN:**
  cientos de millones (~200–400M).
- **Modelos de difusión (ej. Stable Diffusion v1):**
  ~890 millones de parámetros.
- **Modelos de difusión grandes (Stable Diffusion XL, Imagen, DALL·E 3):**
  1–3 **billones** (miles de millones) de parámetros.
- **LLMs generativos tipo GPT-4:**
  se estima entre **hundreds of billions** hasta **1T+ parámetros**.

$$\nabla f(x,y,z,\ldots)=[\partial x\partial f,\partial y\partial f,\partial z\partial f,\ldots]$$

$$\nabla W,bL=[\partial w1\partial L,\partial w2\partial L,\ldots,\partial b1\partial L,\partial b2\partial L,\ldots]$$

2. Derivadas y Gradientes

Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse

$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejemplo 1

$$y = \underbrace{(x^2 + 1)}_{u}{}^3$$

$$y = u^3 \qquad u = x^2 + 1$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = (3u^2)(2x) = 6u^2x = \boxed{6x(x^2 + 1)^2}$$

$$Para\ x = 5$$

$$\frac{dy}{dx} = 6(5)(5^2 + 1)^2 = 30(25 + 1)^2 = 30 * 676 = \boxed{20{,}280}$$

2. Derivadas y Gradientes

Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse

$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 1

$$y = (3x^2 - 7x + 1)^5$$

$$\frac{dy}{dx} =$$

$$Para\ x = 2$$

2. Derivadas y Gradientes

Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse

$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 1

$$y = \underbrace{(3x^2 - 7x + 1)^5}_{u}$$

$$y = u^5 \qquad u = 3x^2 - 7x + 1$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = (5u^4)(6x - 7)$$

$$= (5(3x^2 - 7x + 1)^4)(6x - 7)$$

$$Para\ x = 2$$

$$\frac{dy}{dx} = (5(3 * 2^2 - 7 * 2 + 1)^4)(6 * 2 - 7)$$

$$= (5(12 - 14 + 1)^4)(12 - 7)$$

$$= (5(-1)^4)(5) = 5 * 5 = 25$$

2. Derivadas y Gradientes

Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse

$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 2

$$y = (x^2 - 3x + 8)^3$$

$$\frac{dy}{dx} =$$

$$Para\ x = 1$$

2. Derivadas y Gradientes

Regla de la cadena

Definición intuitiva

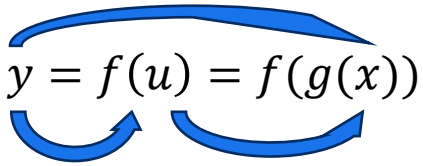
Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse



$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 2

$$y = (\underbrace{x^2 - 3x + 8}_{u})^3$$

$$y = u^3 \qquad u = x^2 - 3x + 8$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = (3u^2)(2x - 3)$$

$$= (3(x^2 - 3x + 8)^2)(2x - 3)$$

$$Para\ x = 1$$

$$= (3(1 - 3 + 8)^2)(2 - 3)$$

$$= (3(6)^2)(-1) = (3 * 36)(-1) = -108$$

2. Derivadas y Gradientes

Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse

$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 3

$$y = (8x - 7)^{-5}$$

$$\frac{dy}{dx} =$$

$$Para\ x = 3$$

2. Derivadas y Gradientes

Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse

$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 3

$$y = \underbrace{(8x-7)}_{u}^{-5}$$

$$y = u^{-5} \qquad u = 8x - 7$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = (-5u^{-6})(8)$$

$$= (-5(8x-7)^{-6})(8)$$

$$= -40(8x-7)^{-6}$$

$$Para\ x = 3$$

$$= -40(8*3-7)^{-6} = -40(17)^{-6}$$

$$= -40\frac{1}{17^6} = -4\frac{1}{24{,}137{,}569} = -1.657$$

2. Derivadas y Gradientes

$$\frac{dy}{dx} sen(x) = \cos(x)$$

Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse

$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 4

$$y = sen\left((3x^2 + 1)^4\right)$$

$$\frac{dy}{dx} =$$

$$Para\ x = 1$$

2. Derivadas y Gradientes

$$\frac{dy}{dx} sen(x) = \cos(x)$$

Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse

$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 4

$$y = sen\big(\overbrace{(3x^2+1)^4}^{v}\big)$$

u

$$y(v) = sen(v)$$

$$v(u) = u^4$$

$$u(x) = 3x^2 + 1$$

$$\frac{dy}{dx} = \frac{dy}{dv}\frac{dv}{du}\frac{du}{dx} = \cos(v)\, 4u^3 6x$$

$$= cos\big((3x^2+1)^4\big)4\big((3x^2+1)^3\big)6x$$

$$= 24x\big((3x^2+1)^3\big)\, cos\big((3x^2+1)^4\big)$$

$Para\ x = 1$

$$= 24\big((3+1)^3\big)\, cos\big((3+1)^4\big)$$

$$= 24(64)\, cos(256) = 1536 * -0.0397 =$$

$$= -61.1186$$

2. Derivadas y Gradientes

$$\frac{dy}{dx} sen(x) = \cos(x)$$

$$\frac{dy}{dx} \ln(x) = \frac{1}{x}$$

Regla de la cadena

Definición intuitiva

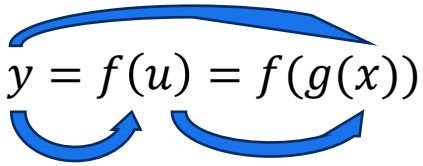
Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse



$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 5

$$y = \sqrt{ln(sen(3x))}$$

2. Derivadas y Gradientes

$$\frac{dy}{dx} sen(x) = \cos(x)$$

$$\frac{dy}{dx} \ln(x) = \frac{1}{x}$$

Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Formulación matemática

Si f y g son dos funciones tales que:

$$y = f(u) \text{ y } u = g(x)$$

y si $g(x)$ está en el dominio de f, entonces puede escribirse

$$y = f(u) = f(g(x))$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = f'(u)g'(x)$$

Ejercicio 5

$$y = \sqrt{ln(sen(3x))}$$

x = w, v = $\sqrt{\ln(sen(3x))}$, u = $ln(sen(3x))$

$$u(x) = 3x$$

$$v(u) = sen(u)$$

$$w(v) = \log(v)$$

$$y(w) = \sqrt{w}$$

$$\frac{dy}{dx} = \frac{dy}{dw}\frac{dw}{dv}\frac{dv}{du}\frac{du}{dx} = \left(\frac{1}{2\sqrt{w}}\right)\left(\frac{1}{v}\right)(\cos(u))(3)$$

$$= \left(\frac{1}{2\sqrt{\ln(sen(3x))}}\right)\left(\frac{1}{sen(3x)}\right)(\cos(3x))(3)$$

Diferenciación con TensorFlow

API central: tf.GradientTape

```python
# Definir la variable x
x = tf.Variable(5.0)

with tf.GradientTape() as tape:
    y = (x**2 + 1)**3   # y = (x^2 + 1)^3

# Calcular la derivada dy/dx
dy_dx = tape.gradient(y, x)

print("Valor de y en x=5:", y.numpy())
print("dy/dx en x=5:", dy_dx.numpy())
```

```
Valor de y en x=5: 17576.0
dy/dx en x=5: 20280.0
```

Ejemplo 1

$$y = \underbrace{(x^2 + 1)}_{u}{}^3$$

$$y = u^3 \qquad u = x^2 + 1$$

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = (3u^2)(2x) = 6u^2x = \boxed{6x(x^2 + 1)^2}$$

$$Para\ x = 5$$

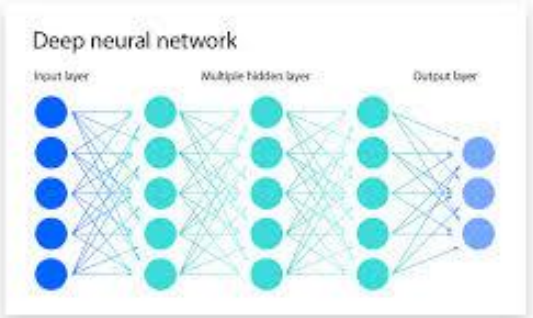
$$\frac{dy}{dx} = 6(5)(5^2 + 1)^2 = 30(25 + 1)^2 = 30 * 676 = \boxed{20{,}280}$$

Google colab

ejercicios_diferenciacion_TensorFlow.ipynb

Ejercicios del 1 al 4

2. Derivadas y Gradientes



Regla de la cadena

Definición intuitiva

Nos permite calcular la derivada funciones compuestas.

Generalization: The chain rule can be extended to more complex compositions of functions. For example, if y=f(u), u=g(v), and v=h(x), then:

$$dxdy=dudy \cdot dvdu \cdot dxdv$$

Papel en DL

- Las RN son en esencia composiciones complejas de funciones.
- Cada capa ejecuta una transformación de su entrada.
- Estas transformaciones se apilan.
- La función de pérdida es una función de la salida de la red, la cual a su vez es una función de los pesos y biases en cada capa.
- Cuando queremos encontrar cómo un cambio en un peso en particular de la red afecta a la pérdida final, necesitamos usar la regla de la cadena para propagar las derivadas a través de las capas.

Backpropagation: el corazón del aprendizaje

3. Probabilidad y Teoría de la Información

Nos proporcionan un marco crucial para comprender y trabajar con incertidumbre en DL.
Son esenciales para tareas como: interpretar las salidas de los modelos, diseñar funciones de pérdida y técnicas de regularización efectivas.

3. Probabilidad y Teoría de la Información

Nos proporcionan un marco crucial para comprender y trabajar con incertidumbre en DL.
Son esenciales para tareas como: interpretar las salidas de los modelos, diseñar funciones de pérdida y técnicas de regularización efectivas.

Conceptos básicos de probabilidad

La probabilidad se encarga de cuantificar la posibilidad de que ocurran eventos.

P(A) - Probabilidad de que ocurra el evento A.

$$P(A) = \frac{\text{número de casos favorables de X}}{\text{número total de casos posibles}}$$

Es un número real entre 0 y 1.

0: imposible
1: con certeza ocurrirá

Ejemplo:

Si lanzamos una moneda justa, la probabilidad de obtener cara es p(cara)= ½ = 0.5=50%,

3. Probabilidad y Teoría de la Información

Nos proporcionan un marco crucial para comprender y trabajar con incertidumbre en DL.
Son esenciales para tareas como: interpretar las salidas de los modelos, diseñar funciones de pérdida y técnicas de regularización efectivas.

Conceptos básicos de probabilidad

La probabilidad se encarga de cuantificar la posibilidad de que ocurran eventos.

P(A) - Probabilidad de que ocurra el evento A.

$$P(A) = \frac{\text{número de casos favorables de X}}{\text{número total de casos posibles}}$$

Es un número real entre 0 y 1.

0: imposible
1: con certeza ocurrirá

Ejemplo:

Si lanzamos una moneda justa, la probabilidad de obtener cara es p(cara)= ½ = 0.5,

$P(A \cap B)$ = $P(A) * P(B)$, para A y B independientes
El resultado de uno no afecta al otro

Probabilidad conjunta

Es la probabilidad de que ocurran simultáneamente tanto el evento A como el B.

Ejemplo:

Si lanzamos dos dados justos de seis lados, la probabilidad de obtener un 3 en el primer dado (evento A) y un 4 en el segundo dado (evento B) es P(A)*P(B) = 1/6 * 1/6 = 1/36.

3. Probabilidad y Teoría de la Información

Nos proporcionan un marco crucial para comprender y trabajar con incertidumbre en DL.
Son esenciales para tareas como: interpretar las salidas de los modelos, diseñar funciones de pérdida y técnicas de regularización efectivas.

Conceptos básicos de probabilidad

La probabilidad se encarga de cuantificar la posibilidad de que ocurran eventos.

P(A) - Probabilidad de que ocurra el evento A.

$$P(A) = \frac{\text{número de casos favorables de X}}{\text{número total de casos posibles}}$$

Es un número real entre 0 y 1.

0: imposible
1: con certeza ocurrirá

Ejemplo:

Si lanzamos una moneda justa, la probabilidad de obtener cara es p(cara)= ½ = 0.5,

P(A ∩ B) = P(A) * P(B), para A y B independientes
El resultado de uno no afecta al otro

Probabilidad conjunta

Es la probabilidad de que ocurran simultáneamente tanto el evento A como el B.

Ejemplo:

Si lanzamos dos dados justos de seis lados, la probabilidad de obtener un 3 en el primer dado (evento A) y un 4 en el segundo dado (evento B) es P(A)*P(B) = 1/6 * 1/6 = 1/36.

$$\mathbf{P(A|B)} = \frac{P(A \cap B)}{P(B)}$$, para P(B)>0

Probabilidad condicional

Es la probabilidad de que ocurra el evento A dado que el evento B ya ocurrió.
Se lee "la probabilidad de A dado B".

Ejemplo:
Se lanza una moneda y, de manera independiente, se lanza un dado de seis caras.

$P(A) = \frac{1}{2}$ (cara o cruz).
$P(B) = \frac{3}{6} = \frac{1}{2}$ (2, 4 o 6 en el dado).

$$P(A \cap B) = P(A)\,P(B) = \tfrac{1}{2} \cdot \tfrac{1}{2} = \tfrac{1}{4}.$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{\frac{1}{4}}{\frac{1}{2}} = \tfrac{1}{2}.$$

P(A) - Probabilidad de que ocurra el evento A.

$$P(A) = \frac{\text{número de casos favorables de X}}{\text{número total de casos posibles}}$$

P(A ∩ B) = P(A) * P(B), para A y B independientes

Probabilidad conjunta

P(A|B) $= \frac{P(A \cap B)}{P(B)}$, para P(B)>0

Probabilidad condicional

Teorema de Bayes

Reorganiza los conceptos fundamentales de la probabilidad para inferir información "al revés" de cómo normalmente la observamos.

$$P(A \mid B) = \frac{P(A) * P(B \mid A)}{P(B)} \text{ para } P(B)>0$$

Nota: No requiere saber si A y B son dependientes o independientes.

Ejemplo:

F = Una persona tiene fiebre
G = La misma persona tiene gripe

$$P(F \mid G) = \frac{P(\ ) \cdot P(\ \ \ \ \ )}{P(\ \ )}$$

Tipo de probabilidad	Significado	Papel en Bayes
Condicional $P(A \mid B)$	Probabilidad de que ocurra A dado que ya ocurrió B	**Es lo que queremos estimar**
Inversa $P(B \mid A)$	Probabilidad de observar B si sabemos que A ocurrió	**Es la que observamos** (modelo generativo)
Prior $P(A)$	Probabilidad previa de A, antes de observar B	**Creencia inicial**
Evidencia $P(B)$	Probabilidad total de B (marginal)	**Sirve para normalizar la ecuación**

Un modelo generativo **sin condiciones** aprende $P(B)$: generar datos realistas de manera "incondicional".

Un modelo generativo **condicional** aprende $P(B|A)$: controlar *qué* se genera a partir de un contexto. Esto permite **control creativo:** decirle al modelo *qué* queremos (ejemplo: *"dibuja un gato azul"* → se modela $P(\text{imagen gato azul}|\text{prompt})$).

P(A) - Probabilidad de que ocurra el evento A.

$$P(A) = \frac{\text{número de casos favorables de X}}{\text{número total de casos posibles}}$$

P(A ∩ B) = P(A) * P(B), para A y B independientes

Probabilidad conjunta

P(A|B) $= \frac{P(A \cap B)}{P(B)}$, para P(B)>0

Probabilidad condicional

Teorema de Bayes

Reorganiza los conceptos fundamentales de la probabilidad para inferir información "al revés" de cómo normalmente la observamos.

$$P(A \mid B) = \frac{P(A) * P(B \mid A)}{P(B)} \text{ para } P(B)>0$$

Nota: No requiere saber si A y B son dependientes o independientes.

Ejemplo:

F = Una persona tiene fiebre
G = La misma persona tiene gripe

$$P(F \mid G) = \frac{P(F) \cdot P(G \mid F)}{P(G)}$$

Tipo de probabilidad	Significado	Papel en Bayes
Condicional $P(A \mid B)$	Probabilidad de que ocurra A dado que ya ocurrió B	**Es lo que queremos estimar**
Inversa $P(B \mid A)$	Probabilidad de observar B si sabemos que A ocurrió	**Es la que observamos** (modelo generativo)
Prior $P(A)$	Probabilidad previa de A, antes de observar B	**Creencia inicial**
Evidencia $P(B)$	Probabilidad total de B (marginal)	**Sirve para normalizar la ecuación**

Un modelo generativo **sin condiciones** aprende $P(B)$: generar datos realistas de manera "incondicional".

Un modelo generativo **condicional** aprende $P(B|A)$: controlar *qué* se genera a partir de un contexto. Esto permite **control creativo:** decirle al modelo *qué* queremos (ejemplo: *"dibuja un gato azul"* → se modela $P(\text{imagen gato azul}|\text{prompt})$).

P(A) - Probabilidad de que ocurra el evento A.

P(A ∩ B) = P(A) * P(B), para A y B independientes

Probabilidad conjunta

P(A|B) $= \frac{P(A \cap B)}{P(B)}$, para P(B)>0

Probabilidad condicional

Teorema de Bayes

$$P(A \mid B) = \frac{P(A) * P(B \mid A)}{P(B)} \text{ para } P(B)>0$$

Ejemplo:

En el consultorio del Dr. Gómez, el 40% de los pacientes fingen tener una enfermedad (para obtener un descanso médico). Además, el 10% de los pacientes del consultorio son hombres. La probabilidad de que un paciente finja una enfermedad dado que es hombre es del 50%. Calcular la probabilidad de que un paciente sea hombre dado que finge una enfermedad.

F = 40% = 0.4 probabilidad de que un paciente finja enfermedad
H = 10% = 0.1 probabilidad de que un paciente sea hombre
P(F|H) = 50% = 0.5 probabilidad de que finja dado que es hombre
P(H|F) =

$$P(F \mid G) = \frac{P(\ ) \cdot P(\ \ \ \ \ )}{P(\ \ )}$$

Nota: Los eventos "ser hombre" y "fingir enfermedad" **son dependientes**, porque conocer uno **altera** la probabilidad del otro.

P(A) - Probabilidad de que ocurra el evento A.

P(A ∩ B) = P(A) * P(B), para A y B independientes

Probabilidad conjunta

P(A|B) $= \frac{P(A \cap B)}{P(B)}$, para P(B)>0

Probabilidad condicional

Teorema de Bayes

$$P(A \mid B) = \frac{P(A) * P(B \mid A)}{P(B)} \text{ para } P(B)>0$$

Ejemplo:

En el consultorio del Dr. Gómez, el 40% de los pacientes fingen tener una enfermedad (para obtener un descanso médico). Además, el 10% de los pacientes del consultorio son hombres. La probabilidad de que un paciente finja una enfermedad dado que es hombre es del 50%. Calcular la probabilidad de que un paciente sea hombre dado que finge una enfermedad.

F = 40% = 0.4 probabilidad de que un paciente finja enfermedad
H = 10% = 0.1 probabilidad de que un paciente sea hombre
P(F|H) = 50% = 0.5 probabilidad de que finja dado que es hombre
P(H|F) =

$$P(H \mid F) = \frac{P(H) \cdot P(F \mid H)}{P(F)}$$

$$= \frac{0.1 * 0.5}{0.4} = 0.125 = 12.5\%$$

Nota: Los eventos "ser hombre" y "fingir enfermedad" **son dependientes**, porque conocer uno **altera** la probabilidad del otro.

P(A) - Probabilidad de que ocurra el evento A.

P(A ∩ B) = P(A) * P(B), para A y B independientes

Probabilidad conjunta

P(A|B) $= \frac{P(A \cap B)}{P(B)}$, para P(B)>0

Probabilidad condicional

Teorema de Bayes

$$P(A \mid B) = \frac{P(A) * P(B \mid A)}{P(B)}$$ para P(B)>0

Ejemplo: Confiabilidad de una prueba médica

Calcula la probabilidad de que una persona realmente esté enferma dado que la prueba médica dio positiva: P(enferma | test positivo).

P(E) = P(enferma) = **0.01** (conocida a priori)
P(TP | E) = P(test positivo | enferma) = **0.95** (probabilidad de que un test de positivo dado que la persona está enferma - sensibilidad).
P(TP |~E) = P(test positivo | no enfermo) = **0.05** (conocida a priori – false positive rate)
P(~E) = P(no enfermo) = 1 – P(enfermo) = **0.99**

P(E | TP) = $\frac{P(\quad) \cdot P(\quad)}{P(\quad)}$

P(A) - Probabilidad de que ocurra el evento A.

P(A ∩ B) = P(A) * P(B), para A y B independientes

Probabilidad conjunta

P(A|B) $= \frac{P(A \cap B)}{P(B)}$, para P(B)>0

Probabilidad condicional

Teorema de Bayes

$$P(A \mid B) = \frac{P(A) * P(B \mid A)}{P(B)} \text{ para } P(B)>0$$

Ejemplo: Confiabilidad de una prueba médica

Calcula la probabilidad de que una persona realmente esté enferma dado que la prueba médica dio positiva: P(enferma | test positivo).

P(E) = P(enferma) = **0.01** (conocida a priori)
P(TP | E) = P(test positivo | enferma) = **0.95** (probabilidad de que un test de positivo dado que la persona está enferma - sensibilidad).
P(TP |~E) = P(test positivo | no enfermo) = **0.05** (conocida a priori – false positive rate)
P(~E) = P(no enfermo) = 1 – P(enfermo) = **0.99**

$$P(E \mid TP) = \frac{P(E) \cdot P(TP \mid E)}{P(TP)}$$

$$= \frac{0.01 * 0.95}{P(TP)}$$

P(A) - Probabilidad de que ocurra el evento A.

P(A ∩ B) = P(A) * P(B), para A y B independientes
Probabilidad conjunta

P(A|B) $= \frac{P(A \cap B)}{P(B)}$, para P(B)>0
Probabilidad condicional

Teorema de Bayes

$$P(A \mid B) = \frac{P(A) * P(B \mid A)}{P(B)} \text{ para } P(B)>0$$

Ejemplo: Confiabilidad de una prueba médica

Calcula la probabilidad de que una persona realmente esté enferma dado que la prueba médica dio positiva: P(enferma | test positivo).

P(E) = P(enferma) = **0.01** (conocida a priori)
P(TP | E) = P(test positivo | enferma) = **0.95** (probabilidad de que un test de positivo dado que la persona está enferma - sensibilidad).
P(TP |~E) = P(test positivo | no enfermo) = **0.05** (conocida a priori – false positive rate)
P(~E) = P(no enfermo) = 1 – P(enfermo) = **0.99**

La probabilidad total de una prueba positiva.

P(TP) = P(TP | E) * P(E) + P(TP | ~E) * P(~E)
= 0.95 * 0.01 + 0.05 * 0.99 = 0.0095 + 0.0495 = **0.059**

$$P(E \mid TP) = \frac{P(E) \cdot P(TP \mid E)}{P(TP)}$$

$$= \frac{0.01 * 0.95}{0.059} = 0.161 = 16.1\%$$

P(A) - Probabilidad de que ocurra el evento A.

P(A ∩ B) = P(A) * P(B), para A y B independientes
Probabilidad conjunta

P(A|B) $= \frac{P(A \cap B)}{P(B)}$, para P(B)>0
Probabilidad condicional

Teorema de Bayes

$$P(A \mid B) = \frac{P(A) * P(B \mid A)}{P(B)} \text{ para } P(B)>0$$

Ejemplo: Confiabilidad de una prueba médica

Calcula la probabilidad de que una persona realmente esté enferma dado que la prueba médica dio positiva: P(enferma | test positivo).

P(E) = P(enferma) = **0.01** (conocida a priori)
P(TP | E) = P(test positivo | enferma) = **0.95** (probabilidad de que un test de positivo dado que la persona está enferma - sensibilidad).
P(TP |~E) = P(test positivo | no enfermo) = **0.05** (conocida a priori – false positive rate)
P(~E) = P(no enfermo) = 1 – P(enfermo) = **0.99**

La probabilidad total de una prueba positiva.

P(TP) = P(TP | E) * P(E) + P(TP | ~E) * P(~E)
= 0.95 * 0.01 + 0.05 * 0.99 = 0.0095 + 0.0495 = **0.059**

$$P(E \mid TP) = \frac{P(E) \cdot P(TP \mid E)}{P(TP)}$$

$$= \frac{0.01 * 0.95}{0.059} = 0.161 = 16.1\%$$

Entonces, aunque la prueba tiene una alta sensibilidad (95 %), P(TP | E),

la probabilidad de que una persona con un resultado positivo realmente tenga la enfermedad es de solo aproximadamente 16.1 %, P(E | TP).

Modelos Generativos Profundos

UN ENFOQUE DESDE LA CREATIVIDAD COMPUTACIONAL

Ejercicios de probabilidad y teoría de la información

Ejercicios 1,2, 4 y 5

3. Probabilidad y Teoría de la Información

Nos proporcionan un marco crucial para comprender y trabajar con incertidumbre en DL.
Son esenciales para tareas como: interpretar las salidas de los modelos, diseñar funciones de pérdida y técnicas de regularización efectivas.

Conceptos básicos de teoría de la información

Entropía

Cuantifica la **cantidad de incertidumbre o aleatoriedad** en una distribución de probabilidad.

Alta entropía: Distribución más impredecible.

Ocurre cuando las probabilidades de diferentes resultados son muy parecidas o son la misma.

P. ej., un lanzamiento de una moneda justa tiene una entropía alta porque P(cara) = 0.5 y P(cruz) = 0.5, haciendo que la probabilidad del resultado sea igualmente incierto.

Baja entropía: Distribución más predecible.

Ocurre cuando la probabilidad está concentrada en unos cuantos resultados.

P. ej., un lanzamiento de una moneda truqueada que siempre cae en cara tiene poca entropía porque P(cara) = 1 y P(cruz) = 0, haciendo que el resultado sea perfectamente predecible.

3. Probabilidad y Teoría de la Información

Nos proporcionan un marco crucial para comprender y trabajar con incertidumbre en DL.
Son esenciales para tareas como: interpretar las salidas de los modelos, diseñar funciones de pérdida y técnicas de regularización efectivas.

Conceptos básicos de teoría de la información

Entropía

Cuantifica la **cantidad de incertidumbre o aleatoriedad** en una distribución de probabilidad.

Alta entropía: Distribución más impredecible.

Ocurre cuando las probabilidades de diferentes resultados son muy parecidas o son la misma.

P. ej., un lanzamiento de una moneda justa tiene una alta entropía porque P(cara) = 0.5 y P(cruz) = 0.5, haciendo que la probabilidad del resultado sea igualmente incierto.

Baja entropía: Distribución más predecible.

Ocurre cuando la probabilidad está concentrada en unos cuantos resultados.

P. ej., un lanzamiento de una moneda truqueada que siempre cae en cara tiene poca entropía porque P(cara) = 1 y P(cruz) = 0, haciendo que el resultado sea perfectamente predecible.

Fórmula: La entropía *H(X)* de una variable aleatoria *X* con valores posibles (x1, x2, ..., xn} y una distribución de probabilidad *P(X)* se define como:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

La base del logaritmo normalmente es 2, dando como resultado una entropía medida en bits.

En DL, se usa frecuentemente el logaritmo natural (base e), con una medida en nats (natural units).

Ejemplo. La entropía del lanzamiento de una moneda justa es:

Dos posibles resultados: **cara (C)** y **cruz (K)**.
Ambas con probabilidad $p = 0.5$.

$$H(X) = -[0.5 \log_2(0.5) + 0.5 \log_2(0.5)]$$

$$H(X) = -[0.5 \cdot (-1) + 0.5 \cdot (-1)] = -[-0.5 - 0.5] = 1 \text{ bit}$$

3. Probabilidad y Teoría de la Información

Nos proporcionan un marco crucial para comprender y trabajar con incertidumbre en DL.
Son esenciales para tareas como: interpretar las salidas de los modelos, diseñar funciones de pérdida y técnicas de regularización efectivas.

Conceptos básicos de teoría de la información

Entropía

Cuantifica la **cantidad de incertidumbre o aleatoriedad** en una distribución de probabilidad.

Alta entropía: Distribución más impredecible.

Ocurre cuando las probabilidades de diferentes resultados son muy parecidas o son la misma.

P. ej., un lanzamiento de una moneda justa tiene una alta entropía porque P(cara) = 0.5 y P(cruz) = 0.5, haciendo que la probabilidad del resultado sea igualmente incierto.

Baja entropía: Distribución más predecible.

Ocurre cuando la probabilidad está concentrada en unos cuantos resultados.

P. ej., un lanzamiento de una moneda truqueada que siempre cae en cara tiene poca entropía porque P(cara) = 1 y P(cruz) = 0, haciendo que el resultado sea perfectamente predecible.

Fórmula: La entropía H(X) de una variable aleatoria X con valores posibles (x1, x2, ..., xn} y una distribución de probabilidad P(X) se define como:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

Ejemplo. La entropía del lanzamiento de la moneda truqueada:

3. Probabilidad y Teoría de la Información

Nos proporcionan un marco crucial para comprender y trabajar con incertidumbre en DL.
Son esenciales para tareas como: interpretar las salidas de los modelos, diseñar funciones de pérdida y técnicas de regularización efectivas.

Conceptos básicos de teoría de la información

Entropía

Cuantifica la **cantidad de incertidumbre o aleatoriedad** en una distribución de probabilidad.

Alta entropía: Distribución más impredecible.

Ocurre cuando las probabilidades de diferentes resultados son muy parecidas o son la misma.

P. ej., un lanzamiento de una moneda justa tiene una alta entropía porque P(cara) = 0.5 y P(cruz) = 0.5, haciendo que la probabilidad del resultado sea igualmente incierto.

Baja entropía: Distribución más predecible.

Ocurre cuando la probabilidad está concentrada en unos cuantos resultados.

P. ej., un lanzamiento de una moneda truqueada que siempre cae en cara tiene poca entropía porque P(cara) = 1 y P(cruz) = 0, haciendo que el resultado sea perfectamente predecible.

Fórmula: La entropía H(X) de una variable aleatoria X con valores posibles (x1, x2, ..., xn} y una distribución de probabilidad P(X) se define como:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_2 P(x_i)$$

Ejemplo. La entropía del lanzamiento de la moneda truqueada (siempre sale sol):

$$H(X) = -(1 \cdot \log_2(1) + 0 \cdot \log_2(0))$$

- $\log_2(1) = 0$
- $0 \cdot \log_2(0) = 0$ (por convención, $0 \log 0 = 0$)

$$H(X) = -(1 \cdot 0 + 0) = 0 \text{ bits}$$

3. Probabilidad y Teoría de la Información

Nos proporcionan un marco crucial para comprender y trabajar con incertidumbre en DL.
Son esenciales para tareas como: interpretar las salidas de los modelos, diseñar funciones de pérdida y técnicas de regularización efectivas.

Conceptos básicos de teoría de la información

Divergencia KL (Kullback-Leibler Divergence)

Es una medida que indica **qué tanto una distribución de probabilidad difiere de otra**.

Cuantifica la pérdida de información cuando una distribución de probabilidad Q se usa para aproximar otra distribución P.

También se conoce como **entropía relativa**.

Fórmula

Para las distribuciones de probabilidad P(x) y Q(x) sobre el mismo conjunto de eventos, la medida KL-divergencia de Q respecto de P es:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}.$$

Propiedades

No es simétrica:

$$D_{KL}(P \parallel Q) \neq D_{KL}(Q \parallel P)$$

Siempre es no-negativa.

$$D_{KL}(P \parallel Q) = 0 \quad \text{si y sólo si} \quad P(x) = Q(x) \quad \text{para todo } x.$$

Modelos Generativos Profundos

UN ENFOQUE DESDE LA CREATIVIDAD COMPUTACIONAL

Ejercicios de probabilidad y teoría de la información

Ejercicios 3, 6 y 7