

Clase 1 - Intro

Introducción al curso y bienvenida - Episodio 1: Machine Learning

Docentes

Machine Learning
fgama@fi.uba.ar



Fernando Gama
fgama@fi.uba.ar



Nicolás Zilberstein
nzilberstein@fi.uba.ar



Alejandro Debus
aledebus@gmail.com

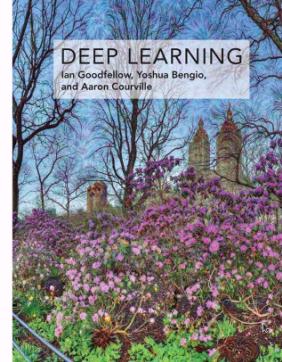
Tres trabajos prácticos:

Aprobación

Machine Learning
fgama@fi.uba.ar

- ▶ 3 trabajos prácticos
 - ⇒ TP1: Redes neuronales convolucionales (CNNs)
 - ⇒ TP2: Redes neuronales recurrentes (RNNs)
 - ⇒ TP3: Redes neuronales en grafos (GNNs)
- ▶ Informe (necesariamente en .pdf) y Código (subirlo al repositorio adecuado de GitHub)
- ▶ Entrega en parejas o grupos de tres (si impares) ⇒ Inicialmente asignadas al azar
 - ⇒ Se recomienda cambiar de parejas por cada TP (también al azar)
- ▶ No va a haber exámenes parciales ni finales, sólo los trabajos prácticos

- ▶ Libro principal
 - ⇒ I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, ser. Adaptive Comput. Mach. Learning. Cambridge, MA: The MIT Press, 2016. Capítulos 2 al 10.
- ▶ Redes neuronales en grafos
 - ⇒ F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, Convolutions, and Neural Networks: From Graph Filters to Graph Neural Networks," *IEEE Signal Process. Mag.*, vol. 37, no. 6, pp. 128–138, Nov. 2020.
- ▶ Bibliografía complementaria
 - ⇒ K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, ser. Adaptive Comput. Mach. Learning. Cambridge, MA: The MIT Press, 2012. Capítulos 1, 2, 6 y 28.



Objetivo

Objetivo

Implementar redes neuronales, determinar su funcionamiento, y poder corregirlas

- ▶ Introducir las **redes neuronales** como herramienta elemental de *deep learning*
- ▶ Comprender los elementos básicos del **aprendizaje por gradiente**
- ▶ Tomar conocimiento de diversas técnicas de **regularización**
- ▶ Reconocer el tipo de red neuronal más adecuado según el **tipo de dato**
 - ⇒ Convolucionales (imágenes), Recurrentes (secuencias), Grafos

Contenido

- ▶ Episodio 1: **Machine Learning**
Motivación, datos, aprendizaje automático
- ▶ Episodio 2: **Redes Neuronales Completas**
Minimización del riesgo empírico, arquitecturas, optimización
- ▶ Episodio 3: **Observaciones** Prácticas para el Aprendizaje
Funciones de costo, regularización
- ▶ Episodio 4: La Operación de **Convolución**
Señales y sistemas, convolución, respuesta en frecuencia, convoluciones en imágenes
- ▶ Episodio 5: Redes Neuronales **Convolucionales**
Arquitecturas, propiedades, regularizaciones
- ▶ Episodio 6: Redes Neuronales **Recurrentes**
Secuencias temporales, arquitecturas, dependencias de largo alcance, *gating*
- ▶ Episodio 7: Procesamiento de Señales en **Grafos**
Señales en grafos, filtrado y convolución, arquitecturas
- ▶ Episodio 8: Redes Neuronales en Grafos
Pooling, propiedades, aplicaciones

Motivación

Álgebra

Probabilidad

Aprendizaje Automático

→ La noción tradicional de programación es declarativa si quieres, especificamos todas las acciones que va a efectuar el programa linea tras linea, en machine learning no sucede esto. Aca viene la idea del aprendizaje, que el algoritmo mejore de manera automática

Aprendizaje

Ma
f

- ▶ Aprendizaje ⇒ Permitir que el algoritmo **mejore** al ganar experiencia
 - ⇒ Determinar una colección de **operaciones aceptables** ⇒ El algoritmo elige **cuáles usar**

- ▶ El algoritmo obtiene conocimiento desde la **experiencia para aprender**
 - ⇒ No hay necesidad de un humano que especifique formalmente todo el conocimiento necesario
- ▶ Determinar la colección de **operaciones adecuadas** para extraer conocimiento de la experiencia
 - ⇒ **Machine learning** ⇒ **Diseñar y analizar algoritmos que mejoren automáticamente**
 - ⇒ El algoritmo aprende a resolver problemas sin haber sido programado específicamente para ello

Aprendizaje automático mediante la **experiencia** ⇒ Aprende **observando datos**

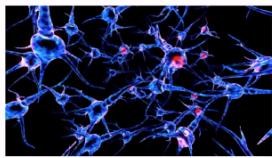
- ▶ Extrae información de los **datos** explotando su estructura
- ▶ Determinar el **modelo matemático** adecuado para procesar los datos

Necesitamos saber cual es el modelo matemático que va para cada problema. Por ejemplo si vamos a trabajar con imágenes, sabemos que vamos a tener que usar matrices. Ahí está nuestro laburo, saber cuáles son las colecciones de operaciones adecuadas.

> Repaso de álgebra

Para describir los datos, vamos a tener que recurrir a nuestra amiga el álgebra.

- ▶ Los datos pueden tener muchos **formatos distintos** y provenir de muchas fuentes distintas



Biología



Transporte



Sensado remoto

- ▶ Necesitamos **determinar operaciones aceptables** para procesar los datos
- ▶ Consideraremos que los **datos son numéricos** y vienen en un arreglo de datos (*array*)

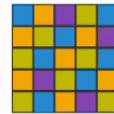
- ▶ Datos numéricos \Rightarrow Modelo matemático \Rightarrow Operaciones aceptables \Rightarrow Álgebra
- ▶ **Arreglo de datos** \Rightarrow Diferentes **interpretaciones matemáticas** prácticas
 - \Rightarrow Escalares \Rightarrow Un sólo elemento, típicamente un *float* $\Rightarrow a \in \mathbb{R}$
 - \Rightarrow Vectores \Rightarrow Un arreglo de N elementos contiguos $\Rightarrow \mathbf{x} \in \mathbb{R}^N$ (fila vs. columna)
 - \Rightarrow Matrices \Rightarrow Un arreglo bidimensional de $M \times N$ elementos $\Rightarrow \mathbf{A} \in \mathbb{R}^{M \times N}$
 - \Rightarrow Tensores \Rightarrow Un conjunto de conjuntos de matrices $\Rightarrow \mathbf{A} \in \mathbb{R}^{N_1 \times N_2 \times \dots \times N_K}$



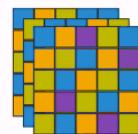
$$a \in \mathbb{R}$$



$$\mathbf{x} \in \mathbb{R}^N \text{ (columna)}$$



$$\mathbf{A} \in \mathbb{R}^{M \times N}$$



$$\mathbf{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$$

Como vemos, segun sean los datos que vamos a procesar y como los vamos a describir, tengo una forma de describirlos matemáticamente.

Operaciones que vamos a utilizar

- Suma + \Rightarrow Entre arreglos de la **misma dimensión** \Rightarrow Suma elemento a elemento
- Traspuesta T \Rightarrow Elemento en la posición (i, j) pasa a estar en la posición (j, i)
- Multiplicación \cdot \Rightarrow Se pueden combinar arreglos de **distintas dimensiones** \Rightarrow Pero con mucho cuidado
 - \Rightarrow Escalar con cualquier cosa \Rightarrow Multiplicación elemento a elemento
 - \Rightarrow Vector con vector \Rightarrow Misma dimensión \Rightarrow fila \cdot columna = escalar, columna \cdot fila = matriz
 - \Rightarrow Vector con matriz \Rightarrow Cant. de columnas igual a dimensión del vector \Rightarrow matrix \cdot vector = vector
 - \Rightarrow Matriz con matriz \Rightarrow Columnas en la izq. igual a filas en la der \Rightarrow matriz \cdot matriz = matriz
 - \Rightarrow Tensores \Rightarrow Colección de matrices \Rightarrow Multiplicar cada dimensión

- Suma + \Rightarrow Entre arreglos de la **misma dimensión** \Rightarrow Suma elemento a elemento
- Traspuesta T \Rightarrow Elemento en la posición (i, j) pasa a estar en la posición (j, i)
- Multiplicación \cdot \Rightarrow Se pueden combinar arreglos de **distintas dimensiones** \Rightarrow Pero con mucho cuidado
 - \Rightarrow Escalar con cualquier cosa \Rightarrow Multiplicación elemento a elemento
 - \Rightarrow Vector con vector \Rightarrow Misma dimensión \Rightarrow fila \cdot columna = escalar, columna \cdot fila = matriz
 - \Rightarrow Vector con matriz \Rightarrow Cant. de columnas igual a dimensión del vector \Rightarrow matrix \cdot vector = vector
 - \Rightarrow Matriz con matriz \Rightarrow Columnas en la izq. igual a filas en la der \Rightarrow matriz \cdot matriz = matriz
 - \Rightarrow Tensores \Rightarrow Colección de matrices \Rightarrow Multiplicar cada dimensión
 - \Rightarrow Distributiva $a \cdot (b + c) = a \cdot b + a \cdot c$ y **asociativa** $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ pero **no conmutativa** $a \cdot b \neq b \cdot a$
- Tener presente que en muchos **lenguajes** las operaciones pueden estar **sobre cargadas**

- Vectores \Rightarrow Describen un espacio N -dimensional $\Rightarrow \mathbf{x} \in \mathbb{R}^N$ se puede alcanzar combinando vectores
- $$\mathbb{R}^2 : \mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 = [\mathbf{v}_1 \quad \mathbf{v}_2] \mathbf{x}$$
- \Rightarrow Se necesitan (al menos) N vectores para describir $\mathbb{R}^N \Rightarrow$ Menos vectores describen un subespacio
- $$\mathbb{R}^3 : \mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \Rightarrow \mathbf{x} = x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 = \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix}$$
- **¿Cuándo es que N vectores $\{\mathbf{v}_1, \dots, \mathbf{v}_N\}$ no alcanzan para describir todo el espacio \mathbb{R}^N ?**
- \Rightarrow Cuando existen valores x_1, \dots, x_N , alguno distinto de cero tal que

$$x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 + \dots + x_N \mathbf{v}_N = \mathbf{0}$$

- Vectores \Rightarrow Describen un espacio N -dimensional $\Rightarrow \mathbf{x} \in \mathbb{R}^N$ se puede alcanzar combinando vectores

$$\mathbb{R}^2 : \mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = x_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 = [\mathbf{v}_1 \quad \mathbf{v}_2] \mathbf{x}$$

\Rightarrow Se necesitan (al menos) N vectores para describir $\mathbb{R}^N \Rightarrow$ Menos vectores describen un subespacio

$$\mathbb{R}^3 : \mathbf{v}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{v}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \Rightarrow \mathbf{x} = x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 = \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix}$$

- ¿Cuándo es que N vectores $\{\mathbf{v}_1, \dots, \mathbf{v}_N\}$ no alcanzan para describir todo el espacio \mathbb{R}^N ?

\Rightarrow Cuando existen valores x_1, \dots, x_N , alguno distinto de cero tal que

$$x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 + \dots + x_N \mathbf{v}_N = \mathbf{0}$$

- Los vectores $\{\mathbf{v}_1, \dots, \mathbf{v}_K\}$ son **linealmente independientes** si

$$x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 + \dots + x_K \mathbf{v}_K = \mathbf{0} \iff x_1 = x_2 = \dots = x_K = 0$$

- Un conjunto de vectores determina los **puntos que podemos alcanzar** al combinarlos

\Rightarrow Dados los vectores $\{\mathbf{v}_1, \dots, \mathbf{v}_K\}$ podemos escribir la combinación lineal como

$$x_1 \mathbf{v}_1 + x_2 \mathbf{v}_2 + \dots + x_K \mathbf{v}_K = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_K] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix} = \mathbf{V}\mathbf{x}$$

\Rightarrow Analizar las características de la matriz \mathbf{V} nos da información sobre el espacio alcanzable

" Al mirar las columnas de una matriz, puedo tener noción del alcance que puede generar mi Matriz" osea, la base. Puedo saber que subespacio me va a generar. Esta hablando básicamente de base del subespacio columna de la matriz. A esto en inglés se lo conoce como range.

- Un conjunto de vectores determina los **puntos que podemos alcanzar** al combinarlos
 ⇒ Dados los vectores $\{\mathbf{v}_1, \dots, \mathbf{v}_K\}$ podemos escribir la combinación lineal como

$$x_1\mathbf{v}_1 + x_2\mathbf{v}_2 + \dots + x_K\mathbf{v}_K = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \dots \quad \mathbf{v}_K] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix} = \mathbf{V}\mathbf{x}$$

- ⇒ Analizar las características de la matriz \mathbf{V} nos da información sobre el espacio alcanzable
- **Alcance** ⇒ Subespacio que puede alcanzar una matriz ⇒ Subespacio generado por las columnas
- **Rango** ⇒ $\min\{\text{filas lineal. indep., cols. lineal. indep.}\} \Rightarrow \mathbf{A} \in \mathbb{R}^{M \times N} \Rightarrow \text{rango}(\mathbf{A}) \leq \min\{M, N\}$

- Dos matrices **cuadradas** con nombre propio que son muy importantes
- Matriz **identidad** ⇒ Matriz \mathbf{I} tal que $\mathbf{A} \cdot \mathbf{I} = \mathbf{A}$ ⇒ **Sólo para matrices cuadradas**

$$\mathbf{A} \cdot \mathbf{I} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \mathbf{A}$$

- Matriz **inversa** ⇒ Matriz \mathbf{A}^{-1} tal que $\mathbf{A} \cdot \mathbf{A}^{-1} = \mathbf{I}$ ⇒ **Sólo para matrices cuadradas**

$$\mathbf{A} \cdot \mathbf{A}^{-1} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \mathbf{A}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Importante: } \mathbf{A}^{-1} \neq \begin{bmatrix} 1/a_{11} & 1/a_{12} & 1/a_{13} \\ 1/a_{21} & 1/a_{22} & 1/a_{23} \\ 1/a_{31} & 1/a_{32} & 1/a_{33} \end{bmatrix}$$

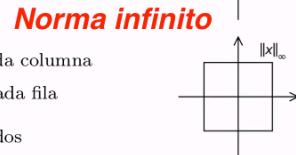
⇒ Calcular la inversa de una matriz no es fácil ⇒ Costo $O(N^3)$ (métodos de $O(N^{2.373})$)

⇒ No todas las matrices cuadradas tienen inversa ⇒ La inversa no siempre existe ⇒ $\text{rango}(\mathbf{A}) = N$

invertir una matriz es una operación computacionalmente costosa. Escala con el cubo de la cantidad de elementos de la matriz.

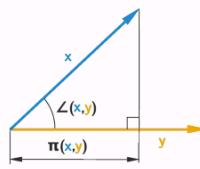
> Algo que vamos a querer a hacer es determinar el tamaño de los datos. (De la matriz) vamos a utilizar la **norma**

- ▶ Determinar el **tamaño de un arreglo de datos** (escalar, vector, matriz) \Rightarrow Norma $\rho : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}_{\geq 0}$
 - \Rightarrow Desigualdad triangular: $\rho(\mathbf{x} + \mathbf{y}) \leq \rho(\mathbf{x}) + \rho(\mathbf{y})$
 - \Rightarrow Homogeneidad absoluta: $\rho(c\mathbf{x}) = \rho_{\mathbb{R}}(c)\rho(\mathbf{x})$
 - \Rightarrow No-negatividad: $\rho(\mathbf{x}) \geq 0$ y $\rho(\mathbf{x}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$
- ▶ Escalares $c \in \mathbb{R} \Rightarrow \rho(c) = |c|$
- ▶ Vectores $\mathbf{x} \in \mathbb{R}^N$
 - \Rightarrow Norma Euclídea $\rho(\mathbf{x}) = \|\mathbf{x}\|_2 = \left(\sum_{i=1}^N |x_i|^2 \right)^{1/2}$
 - \Rightarrow Norma Manhattan $\rho(\mathbf{x}) = \|\mathbf{x}\|_1 = \sum_{i=1}^N |x_i|$
 - \Rightarrow Norma infinito $\rho(\mathbf{x}) = \|\mathbf{x}\|_\infty = \max_{i=1, \dots, N} |x_i|$
- ▶ Matrices $\mathbf{A} \in \mathbb{R}^{M \times N}$
 - $\Rightarrow \|\mathbf{A}\|_2 = \sup_{\mathbf{x}} \|\mathbf{Ax}\|_2 / \|\mathbf{x}\|_2$
 - $\Rightarrow \|\mathbf{A}\|_1 = \max_{1 \leq j \leq N} \sum_{i=1}^M |a_{ij}| \Rightarrow$ El máximo de la suma de cada columna
 - $\Rightarrow \|\mathbf{A}\|_\infty = \max_{1 \leq i \leq M} \sum_{j=1}^N |a_{ij}| \Rightarrow$ El máximo de la suma de cada fila
 - $\Rightarrow \|\mathbf{A}\|_F = \left(\sum_{i=1}^M \sum_{j=1}^N |a_{ij}|^2 \right)^{1/2} \Rightarrow$ Suma de todos los cuadrados



> Otra cosa que vamos a querer hacer es comparar los datos. Para esto vamos a usar el **producto interno**:

- ▶ Determinar una forma de **comparar entre datos** \Rightarrow **Producto interno** $\pi : \mathbb{R}^{N \times N} \times \mathbb{R}^{N \times N} \rightarrow \mathbb{R}$
 - \Rightarrow Linealidad $\Rightarrow \pi(\mathbf{x} + \mathbf{y}, \mathbf{z}) = \pi(\mathbf{x}, \mathbf{z}) + \pi(\mathbf{y}, \mathbf{z})$, $\pi(a\mathbf{x}, \mathbf{y}) = a\pi(\mathbf{x}, \mathbf{y})$
 - \Rightarrow Simetría $\Rightarrow \pi(\mathbf{x}, \mathbf{y}) = \pi(\mathbf{y}, \mathbf{x})$
 - \Rightarrow No-negatividad $\Rightarrow \pi(\mathbf{x}, \mathbf{x}) \geq 0$ y $\pi(\mathbf{x}, \mathbf{x}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{0}$
- ▶ Escalares $c_1, c_2 \in \mathbb{R} \Rightarrow \pi(c_1, c_2) = c_1 c_2$
- ▶ Vectores $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N \Rightarrow \pi(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$
- ▶ Matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{M \times N} \Rightarrow \pi(\mathbf{A}, \mathbf{B}) = \sum_{i=1}^M \sum_{j=1}^N a_{ij} b_{ij}$
- ▶ **El producto interno define una norma** $\pi(\mathbf{x}, \mathbf{x}) = \rho(\mathbf{x})^2$ (no es cierto a la inversa)
- ▶ **Distancia entre datos** $\rho(\mathbf{x} - \mathbf{y})$ para alguna norma adecuada
- ▶ **Ángulo entre datos** $\angle(\mathbf{x}, \mathbf{y}) \Rightarrow \cos(\angle(\mathbf{x}, \mathbf{y})) = \frac{\pi(\mathbf{x}, \mathbf{y})}{\rho(\mathbf{x})\rho(\mathbf{y})}$
 - \Rightarrow Dos vectores son ortogonales si $\pi(\mathbf{x}, \mathbf{y}) = 0$



El producto interno me define una geometría del espacio, me permite determinar o definir distancias, tamaños y ángulos.

- Matrices **diagonales** ⇒ Los únicos elementos no-nulos están en la diagonal

$$\mathbf{D} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix} = \text{diag}(\mathbf{d}), \quad \mathbf{d} = [d_1 \quad d_2 \quad d_3]$$

- Matrices **simétricas** ⇒ La matriz y su traspuesta son idénticas $\mathbf{S} = \mathbf{S}^T$

$$\mathbf{S} = \begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}$$

- Matrices **ortogonales** ⇒ La matriz multiplicada por su traspuesta da la identidad $\mathbf{V}^T \mathbf{V} = \mathbf{V} \mathbf{V}^T = \mathbf{I}$

$$\mathbf{V} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3], \quad \mathbf{V}^T \mathbf{V} = \begin{bmatrix} \mathbf{v}_1^T \mathbf{v}_1 & \mathbf{v}_1^T \mathbf{v}_2 & \mathbf{v}_1^T \mathbf{v}_3 \\ \mathbf{v}_2^T \mathbf{v}_1 & \mathbf{v}_2^T \mathbf{v}_2 & \mathbf{v}_2^T \mathbf{v}_3 \\ \mathbf{v}_3^T \mathbf{v}_1 & \mathbf{v}_3^T \mathbf{v}_2 & \mathbf{v}_3^T \mathbf{v}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

⇒ Vectores ortonormales en las columnas $\|\mathbf{v}_i\|_2^2 = \mathbf{v}_i^T \mathbf{v}_i = 1$ y $\mathbf{v}_i^T \mathbf{v}_j = 0$ cuando $i \neq j$

⇒ La inversa es la traspuesta $\mathbf{V}^{-1} = \mathbf{V}^T$

Para las ortogonales tamos piola, porque no nos jode "el costo de calcular la inversa", porque para ellas es la traspuesta.

> Autovalores y autovectores

- Algunas matrices cuadradas admiten una **descomposición en el producto de tres matrices**

$$\mathbf{A} = \mathbf{V} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}^{-1}$$

con $\mathbf{V} = [\mathbf{v}_1 \quad \cdots \quad \mathbf{v}_N] \in \mathbb{R}^{N \times N}$ y $\boldsymbol{\lambda} = [\lambda_1 \quad \cdots \quad \lambda_N] \in \mathbb{R}^N$ tal que $\mathbf{A} \mathbf{v}_i = \lambda_i \mathbf{v}_i$

⇒ Esta descomposición permite **interpretaciones espectrales** de la matriz

⇒ **Facilita algunas operaciones:** $f(\mathbf{A}) = \mathbf{V} \text{diag}(f(\boldsymbol{\lambda})) \mathbf{V}^{-1}$ para f analítica i.e., $f(x) = \sum_{k=0}^{\infty} f_k x^k$

$$\mathbf{A}^{-1} = \mathbf{V} \text{diag}(\boldsymbol{\lambda}^{-1}) \mathbf{V}^{-1}, \quad \|\mathbf{A}\|_2^2 = \lambda_{\max}\{\mathbf{A}^T \mathbf{A}\}$$

⇒ Sirve para **determinar propiedades numéricas** de la matriz ⇒ Número de condicionamiento

⇒ La **operación es computacionalmente costosa** $O(N^3)$

Descomponer en ava's nos permite ahorrar costos computacionales. El tema es que diagonalizar también es una operación costosa.

- Las matrices **simétricas** tienen una diagonalización en una **base ortonormal** $\mathbf{V} : \mathbf{V}^T \mathbf{V} = \mathbf{V} \mathbf{V}^T = \mathbf{I}$

$$\mathbf{S} = \mathbf{V} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}^T$$

$$\Rightarrow \|\mathbf{S}\|_2^2 = \lambda_{\max}\{\mathbf{S}^T \mathbf{S}\} = \lambda_{\max}\{\mathbf{V} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}^T \mathbf{V} \text{diag}(\boldsymbol{\lambda}) \mathbf{V}^T\} = \lambda_{\max}\{\mathbf{V} \text{diag}(\boldsymbol{\lambda}^2) \mathbf{V}^T\} = (\lambda_{\max}\{\mathbf{S}\})^2$$

> Tengo una banda de data de una matriz, como puedo resumirlo en un numerito que me hable de los datos sin tener que mirar todos?

- Obtener un escalar que, de alguna forma, resuma la información en la matriz (Ejemplo: Normas)
 - ⇒ Algunos casos especiales de uso frecuente para matrices cuadradas
- **Traza** ⇒ Suma de los elementos en la diagonal ⇒ $\text{traza}[\mathbf{A}] = \sum_{i=1}^N a_{ii} = \sum_{i=1}^N \lambda_i(\mathbf{A})$
 - ⇒ Norma de Frobenius: $\|\mathbf{A}\|_F = \sqrt{\text{traza}[\mathbf{A}^\top \mathbf{A}]}$, Producto interno: $\pi(\mathbf{A}, \mathbf{B}) = \text{traza}[\mathbf{A}^\top \mathbf{B}]$
 - ⇒ Propiedades: $\text{traza}[\mathbf{ABC}] = \text{traza}[\mathbf{CAB}] = \text{traza}[\mathbf{BCA}]$ y $\mathbf{x}^\top \mathbf{y} = \text{traza}[\mathbf{yx}^\top]$
 - ⇒ Fácil de calcular en una computadora
- **Determinante** ⇒ $\det(\mathbf{A}) : \mathbb{R}^{N \times N} \rightarrow \mathbb{R}$ tal que $\det(\mathbf{A}) = \prod_{i=1}^N \lambda_i(\mathbf{A})$
 - ⇒ Propiedades: $\det(\mathbf{A}^\top) = \det(\mathbf{A})$, $\det(\mathbf{AB}) = \det(\mathbf{A}) \det(\mathbf{B})$, $\det(\mathbf{A}^{-1}) = 1/\det(\mathbf{A})$
 - ⇒ Computacionalmente costoso $O(N!)$ (se puede lograr en $O(N^3)$)

El determinante es mucho mas costoso de calcular. En general de nociiones de volumen de la matriz de datos.

> Repaso de Probabilidad

En general asociamos la probabilidad de los juegos de azar. Tienen repetibilidad y ademas incertidumbre, nunca podemos saber a ciencia cierta cual va a ser el resultado que vamos a obtener.

- Típicamente asociamos probabilidades a juegos de azar



- ¿Qué tienen los juegos de azar?
 - ⇒ Repetibilidad bajo las mismas condiciones (aproximadamente)
 - ⇒ Imposibilidad de describir exactamente qué va a pasar

Acá nos interesa la probabilidad porque nosotros en DL no podemos saber de ante mano todo el espacio de datos que vamos a llegar a procesar. La probabilidad nos permite simplificar los modelos de mis datos. Nos da un marco teórico que nos permite trabajar con datos inciertos.

- ▶ Hay una **imposibilidad práctica de describir** exactamente todos los datos que queremos procesar
 - ⇒ **Simplificar** los modelos incorporando teoría de probabilidades
 - ⇒ **Cuantificar** lo que no sabemos acerca de los datos
- ▶ **Teoría de probabilidades** ⇒ Marco teórico para trabajar con **datos inciertos**
 - ⇒ Cómo describimos la incertezza de los datos
 - ⇒ Cómo operamos con entidades que son inciertas
- ▶ Que un fenómeno sea aleatorio no implica que todos los eventos posibles tienen la misma probabilidad



> Sobre variables aleatorias

Como trabajar con conjuntos es muy engorroso porque hay operaciones que no están definidas, trabajamos con variables aleatorias, los transformamos en vectores. Lo que nos lleva a las distribuciones de probabilidad

- ▶ **Espacio de probabilidad** (Ω, \mathcal{A}, P) ⇒ Abstracción matemática para describir fenómenos aleatorios
 - ⇒ Ω : eventos elementales ⇒ “unidades” aleatorias del fenómeno
 - ⇒ \mathcal{A} : álgebra de eventos ⇒ cuáles combinaciones de los eventos elementales son posibles
 - ⇒ P : función de probabilidad ⇒ asigna probabilidades a los eventos $P : \mathcal{A} \rightarrow \mathbb{R}$
- ▶ **Variables aleatorias** $\mathbf{X} : \Omega \rightarrow \mathbb{R}^N$ ⇒ Mapean los eventos elementales en un vector
 - ⇒ \mathcal{A} : álgebra de eventos ⇒ Rectángulos $\{[\mathbf{X}(\omega)]_i \leq x_i\}_{i=1}^N$ con $\omega \in \Omega$ ($\mathbf{x} \in \mathbb{R}^N$ determina el evento)
 - ⇒ $F_x : \mathbb{R}^N \rightarrow [0, 1]$: distribución de probabilidades ⇒ $F_x(\mathbf{x}) = P(\mathbf{X} \leq \mathbf{x}) = P(\{[\mathbf{X}(\omega)]_i \leq x_i\}_{i=1}^N)$
- ▶ Variables aleatorias **discretas** ⇒ Ω es numerable ⇒ $f_x(\mathbf{x}) = P(\mathbf{X} = \mathbf{x})$
- ▶ Variables aleatorias **continuas** ⇒ Ω es no numerable ⇒ **densidad** $f_x(\mathbf{x}) = \frac{\partial^N F_x(\mathbf{x})}{\partial x_1 \partial x_2 \cdots \partial x_N}$

► Discretas

Nombre	Soporte	Parámetros	$f_x(\mathbf{x}) = P(\mathbf{X} = \mathbf{x})$
Bernoulli	$\mathbf{X} \in \{0, 1\}$	$p \in [0, 1]$	$f_x(x) = p^x(1-p)^{1-x}$
Binomial	$\mathbf{X} \in \{0, 1, \dots, M\}$	$M \in \mathbb{N}, p \in [0, 1]$	$f_x(x) = \binom{M}{x} p^x(1-p)^{M-x}$
Uniforme	$\mathbf{X} \in \{1, \dots, M\}$	$M \in \mathbb{N}$	$f_x(x) = \frac{1}{M}$
Poisson	$\mathbf{X} \in \{0, 1, 2, \dots\}$	$\lambda \in \mathbb{R}, \lambda \geq 0$	$f_x(x) = \frac{\lambda^x}{x!} e^{-\lambda}$
Multinomial	$\mathbf{X} \in \{0, \dots, M\}^N, \sum_{i=1}^N x_i = M$	$M \in \mathbb{N}, \mathbf{p} \in [0, 1]^N$	$f_x(\mathbf{x}) = \frac{M!}{x_1! \cdots x_N!} p_1^{x_1} \cdots p_N^{x_N}$

► Continuas

Nombre	Soporte	Parámetros	densidad $f_x(\mathbf{x})$
Exponencial	$\mathbf{X} \in \mathbb{R}, x \geq 0$	$\lambda \in \mathbb{R}, \lambda \geq 0$	$f_x(x) = \lambda e^{-\lambda x}$
Normal/Gaussiana	$\mathbf{X} \in \mathbb{R}$	$\mu \in \mathbb{R}, \sigma \in \mathbb{R}, \sigma > 0$	$f_x(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{\sigma^2}}$
Uniforme	$\mathbf{X} \in [a, b]$	$a, b \in \mathbb{R}, a \leq b$	$f_x(x) = \frac{1}{b-a}$
Normal Multivariante	$\mathbf{X} \in \mathbb{R}^N$	$\boldsymbol{\mu} \in \mathbb{R}^N, \boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}, \boldsymbol{\Sigma} > \mathbf{0}$	$f_x(\mathbf{x}) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})}}{\sqrt{(2\pi)^N \det(\boldsymbol{\Sigma})}}$

De la manera en la que nosotros vamos a modelar los datos es como si fueran un conjunto de variables aleatorias, vamos a tener una distribución conjunta de los datos, me va a decir de cierta manera cual es la probabilidad de haber obtenidos esos datos.

- Conjunto de variables aleatorias $\mathbf{X}_1, \dots, \mathbf{X}_M \Rightarrow$ Distribución conjunta $f_{x_1, \dots, x_M}(\mathbf{x}_1, \dots, \mathbf{x}_M)$
 \Rightarrow Distribuciones marginales $\Rightarrow f_{x_i}(\mathbf{x}_i) = \int f_{x_1, \dots, x_M}(\mathbf{x}_1, \dots, \mathbf{x}_M) d\mathbf{x}_1 \cdots d\mathbf{x}_{i-1} d\mathbf{x}_{i+1} \cdots d\mathbf{x}_M$
- Ejemplo: dos variables aleatorias \mathbf{X}, \mathbf{Y}
 \Rightarrow Conjunta $f_{x,y}(\mathbf{x}, \mathbf{y})$, Marginal $f_x(\mathbf{x}) = \int f_{x,y}(\mathbf{x}, \mathbf{y}) d\mathbf{y}$
- Distribución condicional $\mathbf{X}|\mathbf{Y} \Rightarrow f_{x|y}(\mathbf{x}) = \frac{f_{x,y}(\mathbf{x}, \mathbf{y})}{f_y(\mathbf{y})}$
- Regla de Bayes: Relacionar $\mathbf{X}|\mathbf{Y}$ con $\mathbf{Y}|\mathbf{X}$

$$f_{x|y}(\mathbf{x}) = \frac{f_{x,y}(\mathbf{x}, \mathbf{y})}{f_y(\mathbf{y})} = \frac{f_{y|x}(\mathbf{y})f_x(\mathbf{x})}{\int f_{y|x}(\mathbf{y})f_x(\mathbf{x})d\mathbf{x}}$$

- Independencia: $\mathbf{X}|\mathbf{Y} \sim \mathbf{X} \Rightarrow f_{x|y}(\mathbf{x}) = f_x(\mathbf{x}) \Rightarrow \mathbf{Y} \text{ no me aporta información sobre } \mathbf{X}$
- En general, la distribución conjunta de variables aleatorias $\mathbf{X}_1, \dots, \mathbf{X}_M$ mutuamente independientes

$$f_{x_1, \dots, x_M}(\mathbf{x}_1, \dots, \mathbf{x}_M) = f_{x_1}(\mathbf{x}_1)f_{x_2}(\mathbf{x}_2) \cdots f_{x_M}(\mathbf{x}_M)$$

Si X es una V.A, estará completamente caracterizada por su distribución de probabilidades. Es una función que el asigna probabilidades a todos los valores posibles que puede tomar mi V.A. es un montón de info que va a ser difícil de almacenar, por eso vamos a querer algún tipo "de resume" de esa info sobre mis VAs, como lo puedo condensar en un numerito. Aca entran los momentos de mis VAs, media esperanza, etc.

- **\mathbf{X}** está completamente caracterizada por su distribución de probabilidades $F_x(\mathbf{x}) = P(\mathbf{X} \leq \mathbf{x})$
 - ⇒ La distribución de probabilidades es una función $F_x : \mathbb{R}^N \rightarrow [0, 1]$
 - ⇒ Mucha información para almacenar, especialmente si la distribución no es paramétrica
- Calcular resúmenes de la variable aleatoria que sean escalares
 - ⇒ Definición general: Sea $h : \mathbb{R}^N \rightarrow \mathbb{R}^M$ una función arbitraria

$$\mathbb{E}_x[h(\mathbf{X})] = \int h(\mathbf{x}) f_x(\mathbf{x}) d\mathbf{x}$$

- Resúmenes más usados: Momentos de primer y segundo orden
 - ⇒ **Media** $\mathbb{E}_x[\mathbf{X}] = \boldsymbol{\mu}$ (en este caso $h(\mathbf{X}) = \mathbf{X}$ es la identidad)
 - ⇒ **Covarianza** $\mathbb{E}_x[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^\top] = \boldsymbol{\Sigma}$ que es una matriz en $\mathbb{R}^{N \times N}$ semidefinida positiva
- Variables aleatorias unidimensionales
 - Momentos de orden p : $\mathbb{E}_x[\mathbf{X}^p] = \int x^p f_x(x) dx$
 - Bajo ciertas condiciones, conocer todos los momentos $\{\mathbb{E}_x[\mathbf{X}^p], p = 1, 2, \dots\}$ permite reconstruir f_x

> Entrando en lo que es estadística

Conocer una VA X quiere decir conocer su distribución. Tradicionalmente esto se hacia mediante un modelo. Por ejemplo caras de un dado, asumiendo que todas las caras son igualmente probables. Trabajando con datos, no voy a saber cuales son los valores probables, pero si vamos a tener realizaciones de esa variable aleatoria. Repetir el experimento muchas veces y observe el resultado, voy a usar eso que observe como una manera de obtener cual era la distribución real. Básicamente estadística ;)

- Conocer una variable aleatoria \mathbf{X} implica conocer su distribución $F_x(\mathbf{x}) = P(\mathbf{X} \leq \mathbf{x})$
- A veces, esto es posible gracias a conocimientos del problema (modelización)
- Muchas otras veces, sólo tenemos acceso a **realizaciones** de la variable aleatoria
 - ⇒ Una realización de una v.a. es una instancia de esa v.a. $\mathbf{X} \rightsquigarrow \mathbf{x}$
 - ⇒ Como el **experimento es repetible**, uno puede tener muchas realizaciones $\mathbf{X} \rightsquigarrow \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$
- La estadística se usa para **inferir modelos probabilísticos** a través de una muestra de realizaciones
 - ⇒ Distribución de probabilidades $\hat{F}_x(\mathbf{x}) = M^{-1} \sum_{m=1}^M 1\{\mathbf{x}_m \leq \mathbf{x}\}$, densidades: **histogramas**
 - ⇒ **Media**: $\mathbb{E}_x[\mathbf{X}] = \boldsymbol{\mu} \approx \hat{\boldsymbol{\mu}} = M^{-1} \sum_{m=1}^M \mathbf{x}_m$
 - ⇒ **Covarianza**: $\mathbb{E}_x[(\mathbf{X} - \boldsymbol{\mu})(\mathbf{X} - \boldsymbol{\mu})^\top] = \boldsymbol{\Sigma} \approx \hat{\boldsymbol{\Sigma}} = M^{-1} \sum_{m=1}^M (\mathbf{x}_m - \hat{\boldsymbol{\mu}})(\mathbf{x}_m - \hat{\boldsymbol{\mu}})^\top$
 - ⇒ **Ley de los grandes números**: $\mathbb{E}_x[h(\mathbf{X})] \approx M^{-1} \sum_{m=1}^M h(\mathbf{x}_m)$
 - ⇒ **Máxima verosimilitud**: $\hat{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta}} \prod_{m=1}^M f_x(\mathbf{x}_m; \boldsymbol{\theta})$ (distribuciones paramétricas)
- Saber cuándo un estimador es bueno, cuándo converge, cuántas muestras se necesitan, etc.

Yo puedo inferir modelos probabilísticos a través de la muestra de una población. Acá lo que asumimos que es clave es que las relaciones de la VA son independientes, que siempre siguen una misma distribución.

> Que es el Machine Learning?

- La definición no es matemática, y por lo tanto, no es taxativa

Definición de Aprendizaje (Mitchell, 1997)

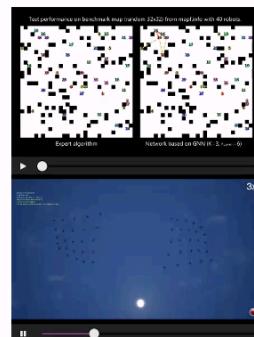
Se dice que **un algoritmo aprende** de una experiencia E con respecto a una dada **clase de tareas T** y según **una medida de desempeño P** , si su desempeño sobre las tareas T **mejoró**, **medido según P** , al considerar la experiencia E

- El objetivo es **utilizar los datos** (experiencia E) para que **el algoritmo mejore**
 - ⇒ El diseño de algoritmos de aprendizaje se reduce a **cómo mejor utilizar los datos**
 - ⇒ Y las nociones de qué es mejor, vienen dados por la **tarea a resolver** y la **medida de desempeño**

Si bien el libro pone un montón de ejemplos (como 15) de problemas de ML, Fernando dice que para él se resumen perfectamente en dos:

Tareas a Resolver

- Clasificación
 - ⇒ El algoritmo toma los datos
 - ⇒ Los mapea en un **conjunto discreto de valores**
 - ⇒ $f : \mathbb{R}^N \rightarrow \{1, \dots, C\}$
 - ⇒ $c \in \{1, \dots, C\}$ corresponde a una clase
- Regresión
 - ⇒ El algoritmo toma los datos
 - ⇒ Los mapea en **valores continuos**
 - ⇒ $f : \mathbb{R}^N \rightarrow \mathbb{R}^D$
 - ⇒ Cada punto $y \in \mathbb{R}^D$ puede ser cualquier valor



- Típicamente, queremos tener la menor cantidad de errores posibles
- Cómo se contabilizan los errores depende de la tarea en sí
- Clasificación ⇒ Todos los errores valen lo mismo → acierto o no acierto. colta
 - ⇒ Los errores se cuantifican de manera binaria: cometí un error o no
- Regresión ⇒ Los errores están determinados por qué tan lejos quedás el valor objetivo
 - ⇒ Equivocarse, pero equivocarse por poco, es mejor que equivocarse por mucho

 **La magnitud del error SI importa.**

- Aprendizaje supervisado ⇒ Tenemos datos, y también tenemos el mapeo correcto para esos datos
 - ⇒ Los datos tienen la forma $\{(\mathbf{x}_m, \mathbf{y}_m)\}_{m=1}^M$
 - ⇒ Queremos una función f tal que se minimice $J(\mathbf{y}_m, f(\mathbf{x}_m))$ para alguna medida de desempeño J
- Aprendizaje no-supervisado ⇒ Tenemos sólo los datos, pero no el mapeo
 - ⇒ Los datos tienen la forma $\{\mathbf{x}_m\}$
 - ⇒ Se busca la función f que minimiza $J(f(\mathbf{x}_m))$ para alguna medida de desempeño J
- Típicamente, los problemas de clasificación implican la existencia de datos supervisados
 - ⇒ Pero no tiene por qué ser necesariamente así ⇒ Los algoritmos de clustering

> Como puedo pensar un problema de regresión lineal en términos de aprendizaje automático?

Es importante que nosotros vamos a tener que determinar el modelo a utilizar. Si bien los pesos W los va a aprender el modelo, nosotros tenemos que ser capaces de elegir el modelo. Aca proponemos uno lineal por ejemplo.

- Consideremos un conjunto de muestras $\{(\mathbf{x}_m, y_m)\}$
 - ⇒ (i) \mathbf{x}_m : mediciones de velocidad y posición, y_m : aceleración
 - ⇒ (ii) \mathbf{x}_m : valor de las acciones de algunas compañías, y_m : valor de la acción de otra compañía
- Elegimos un algoritmo lineal $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x}$ ⇒ Es fácil de implementar
- Queremos usar los datos para aprender \mathbf{w}
- La medida de desempeño es

$$J(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2 = (y - \mathbf{w}^T \mathbf{x})^2$$

- Notemos que si existe \mathbf{w} es tal que $y_m = \mathbf{w}^T \mathbf{x}_m$, entonces $J(y_m, \mathbf{w}^T \mathbf{x}_m) = 0$
- Si esto se cumple para todo m , entonces ese valor de \mathbf{w} determina el algoritmo perfecto
- Si existe al menos un valor de m que no cumple que $y_m = \mathbf{w}^T \mathbf{x}_m$, entonces no podemos obtener $J = 0$
- Buscamos el algoritmo que minimiza J ⇒ Buscamos el vector \mathbf{w} tal que

$$\mathbf{w}^* = \arg \min_{\mathbf{w}} \frac{1}{M} \sum_{m=1}^M J(y_m, \mathbf{w}^T \mathbf{x}_m)$$

Vamos a calcular la derivada de la función costo respecto de mi vector \mathbf{W} , ya que queremos encontrar el \mathbf{W} que la hace mínima, es decir minimizar con respecto a \mathbf{W}

- Reescribimos cada término del costo como
- $$J(y_m, \mathbf{w}^T \mathbf{x}_m) = (y_m - \mathbf{w}^T \mathbf{x}_m)^2 = y_m^2 - 2y_m(\mathbf{w}^T \mathbf{x}_m) + (\mathbf{w}^T \mathbf{x}_m)^2$$
- ⇒ Para minimizar esto hay que encontrar el valor donde la derivada se hace cero
 - Si no están familiarizados con derivadas de vectores, esto se escribe como

$$J_m(\mathbf{w}) = y_m^2 - 2y_m \sum_{i=1}^N \mathbf{w}_i x_{im} + \left(\sum_{i=1}^N \mathbf{w}_i x_{im} \right)^2$$

- De manera que la derivada se puede tomar para cada elemento de \mathbf{w}

$$\frac{\partial J_m}{\partial \mathbf{w}_i} = -2y_m x_{im} + 2 \left(\sum_{i=1}^N \mathbf{w}_i x_{im} \right) x_{im}$$

- Y, en notación vectorial, esto se convierte en

$$\frac{\partial J_m}{\partial \mathbf{w}} = -2y_m \mathbf{x}_m^T + 2(\mathbf{w}^T \mathbf{x}_m) \mathbf{x}_m$$

Vamos a minimizar en el promedio de todas las muestras:

Ejemplo: Regresión Lineal – Obtención

Machine Learning
fgama@fi.uba.ar

- Ahora volvemos a la función original que queremos minimizar y le tomamos la derivada

$$\frac{\partial}{\partial \mathbf{w}} \left\{ \frac{1}{M} \sum_{m=1}^M J(y_m, \mathbf{w}^\top \mathbf{x}_m) \right\} = \frac{1}{M} \sum_{m=1}^M \frac{\partial J_m}{\partial \mathbf{w}}$$

- Y usando la derivada $\partial J_m / \partial \mathbf{w} = -2y_m \mathbf{x}_m + 2(\mathbf{w}^\top \mathbf{x}_m) \mathbf{x}_m$ nos queda

$$\frac{1}{M} \sum_{m=1}^M (-2y_m \mathbf{x}_m + 2(\mathbf{w}^\top \mathbf{x}_m) \mathbf{x}_m) = -\frac{2}{M} \sum_{m=1}^M y_m \mathbf{x}_m + \frac{2}{M} \sum_{m=1}^M (\mathbf{w}^\top \mathbf{x}_m) \mathbf{x}_m$$

- Ahora, notemos que $(\mathbf{w}^\top \mathbf{x}_m)$ es un escalar, y además $\mathbf{w}^\top \mathbf{x}_m = \mathbf{x}_m^\top \mathbf{w}$, entonces

$$(\mathbf{w}^\top \mathbf{x}_m) \mathbf{x}_m = \mathbf{x}_m (\mathbf{w}^\top \mathbf{x}_m) = \mathbf{x}_m (\mathbf{x}_m^\top \mathbf{w}) = (\mathbf{x}_m \mathbf{x}_m^\top) \mathbf{w}$$

⇒ En el último paso usamos la asociatividad del producto de matrices ⇒ $\mathbf{x}_m \mathbf{x}_m^\top \in \mathbb{R}^{N \times N}$

¶

Ejemplo: Regresión Lineal – Obtención

Machine Learning
fgama@fi.uba.ar

- Con lo que finalmente obtenemos que la derivada se puede escribir como

$$\frac{\partial}{\partial \mathbf{w}} \left\{ \frac{1}{M} \sum_{m=1}^M J(y_m, \mathbf{w}^\top \mathbf{x}_m) \right\} = -\frac{2}{M} \sum_{m=1}^M y_m \mathbf{x}_m + \frac{2}{M} \sum_{m=1}^M (\mathbf{x}_m \mathbf{x}_m^\top) \mathbf{w}$$

- Obtenemos el valor óptimo de \mathbf{w} como aquel que anula la derivada

$$-\frac{2}{M} \sum_{m=1}^M y_m \mathbf{x}_m + \frac{2}{M} \sum_{m=1}^M (\mathbf{x}_m \mathbf{x}_m^\top) \mathbf{w}^* = 0$$

- Dividiendo por $2/M$, obtenemos que

$$\left[\sum_{m=1}^M \mathbf{x}_m \mathbf{x}_m^\top \right] \mathbf{w}^* = \left[\sum_{m=1}^M y_m \mathbf{x}_m \right]$$

⇒ Si $M \geq N$ y $[\sum_{m=1}^M \mathbf{x}_m \mathbf{x}_m^\top]$ es invertible, conseguimos el valor óptimo de \mathbf{w}

Ejemplo: Regresión Lineal – Análisis

Machine Learning
fgama@fi.uba.ar

- Elegimos un algoritmo lineal y usamos los datos para encontrar el mejor algoritmo

$$\mathbf{w}^* = \left[\sum_{m=1}^M \mathbf{x}_m \mathbf{x}_m^\top \right]^{-1} \left[\sum_{m=1}^M y_m \mathbf{x}_m \right]$$

⇒ Aprendimos \mathbf{w} usando el conjunto de datos (\mathbf{x}_m, y_m)

- La obtención del mejor algoritmo puede ser engorrosa ⇒ Usar programación autograd
- Puede ser que la función a optimizar no sea diferenciable ⇒ Episodio 2
- La elección de un algoritmo lineal es un poco arbitraria ⇒ Episodio 2

¶

Así encontramos el mejor \mathbf{W} según la medida de desempeño que elegí, en este caso la distancia.

Pero claro, todo esto es engorroso, tuve que derivar, igualar etc etc. Tambien puede ser que la funcion que quiera optimizar no sea diferenciable y que se un bardo calcular la derivada, etc ,etc ,etc.

> Aca lo hicimos con lapiz y papel, vamos a ver como se puede hacer esto con ML para que se haga solo.

> Pero como sabemos que la función que elegimos va a servir para el resto de los datos que no conoce? Como sabemos que generaliza bien?

Generalización

Machine Learning
fgama@fi.uba.ar

- ▶ ¿Cómo sabemos que el algoritmo que aprendió de ciertos datos va a servir cuando tenga nuevos datos?

El algoritmo debe tener un **buen desempeño en muestras que no hemos visto**

- ▶ El aprendizaje estadístico (*statistical learning*) pretende dar respuesta a este problema
- ▶ Asumimos que los datos de interés son v.a. tienen cierta distribución $\mathbf{X} \sim f_x(\mathbf{x})$
- ▶ Asumimos que las muestras que observamos son realizaciones de esa distribución

$$\mathbf{X} \rightsquigarrow \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_M\}$$

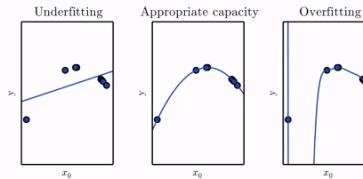
- ▶ Optimizamos sobre los datos observados \Rightarrow Sirve para los no observados \Rightarrow Misma distribución

La suposición fundamental es que los datos vienen de la misma distribucion y que son IID independientes. Por lo tanto, la generalización q obtuve va a servir para generalizar datos que no observe aun siempre y cuando estos sigan la misma distribución.

Overfitting & Underfitting

Machine Learning
fgama@fi.uba.ar

- ▶ En la práctica, **particionamos los datos** disponibles en dos
 \Rightarrow Muestras de **entrenamiento** (80%) y muestras de **prueba** (20%) de manera aleatoria
- ▶ Las muestras de entrenamiento se usan para **encontrar el mejor algoritmo**
- ▶ Las muestras de prueba se usan para confirmar que **el algoritmo funciona bien**
- ▶ Según los errores en las muestras de entrenamiento y en las de prueba surgen dos problemas
 \Rightarrow **Underfitting** si el **error de entrenamiento** es muy grande
 \Rightarrow **Overfitting** si el **error de entrenamiento** es pequeño pero **el de prueba es grande**
- ▶ Ejemplo: Dados los puntos azules
- ▶ Algoritmo 1: lineal
- ▶ Algoritmo 2: cuadrático
- ▶ Algoritmo 3: polinomio de alto orden



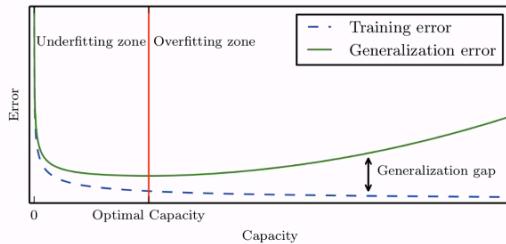
El set de validacion por ahora lo vamos a entender como un subconjunto del set de train, mas adelante lo vemos en detalle para regularizar. Pero básicamente nos permite tomar decisiones en real time de manera estadísticamente

independiente sobre nuestro aprendizaje.

Capacidad

Machine Learning
fgama@fi.uba.ar

- ▶ Para arreglar el problema de *underfitting* (*overfitting*), se ajusta la **capacidad** del algoritmo
 - ⇒ Habilidad del algoritmo para ajustar distintos modelos a los datos observados (se puede calcular)



- ▶ Algoritmos más simples van a generalizar mejor ⇒ Puede que tengan **mucho** error de entrenamiento
- ▶ Algoritmos más complejos van a minimizar el error de entrenamiento ⇒ Pero **no** van a generalizar
- ▶ Existen muchas otras técnicas para solucionar el *overfitting* ⇒ Episodios siguientes

La capacidad es algo que nosotros tenemos que elegir como parte del diseño.

Nada Es Gratis

Machine Learning
fgama@fi.uba.ar

- ▶ Teorema 'Nada es Gratis' (teorema *No Free Lunch*) [Wolpert, 1996]
 - ⇒ Considerando todas las distribuciones de los datos, todos los algoritmos tienen el mismo desempeño
- ▶ Ningún algoritmo es universalmente mejor que ningún otro

No existe un algoritmo único que es bueno para todos los datos

⇒ Hay que diseñar algoritmos específicos para cada tarea

¶

- El aprendizaje automático es una especie de **estadística aplicada**
 - ⇒ Usar el **poder computacional** y la **gran cantidad de datos** para **aprender funciones difíciles**
 - ⇒ No necesariamente otorgar garantías (intervalos de confianza, varianza) sobre lo aprendido

- Preguntas sin respuesta
 - ⇒ ¿Se puede aprender cualquier cosa?
 - ⇒ ¿Cuándo fallan los algoritmos de aprendizaje automático?
 - ⇒ ¿Por qué fallan? ¿Por qué funcionan?
 - ⇒ ¿Cuál es el mejor algoritmo para usar dados unos datos y una tarea?

k

La gran diferencia con la estadística es que el ML no se enfoca en entregar garantías matemáticas de lo que aprendí. Sino que se enfoca en aprender funciones matemáticas difíciles sin necesariamente tener una garantía matemática.

> La clase que viene vamos a ver...

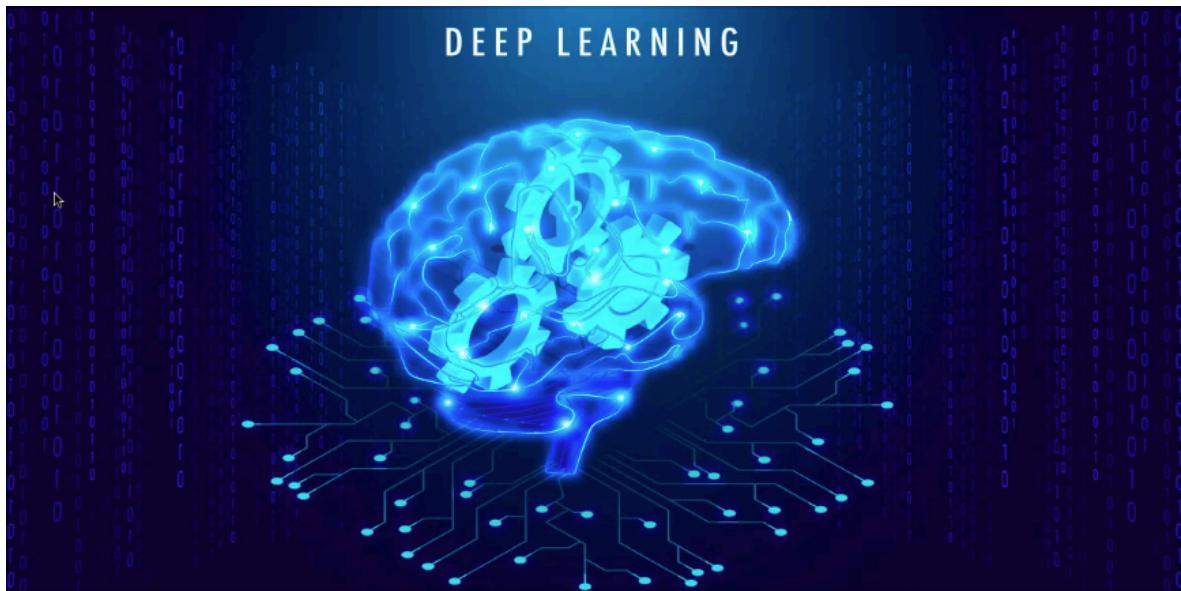
- ¿Cómo definimos formalmente un problema de aprendizaje?
 - ⇒ **Minimización del riesgo empírico**

- ¿Qué tipo de algoritmos vamos a usar? ¿Por qué?
 - ⇒ **Redes Neuronales:** Definición y propiedades

- ¿Cómo hacemos para encontrar el mejor algoritmo?
 - ⇒ **Optimización**

k

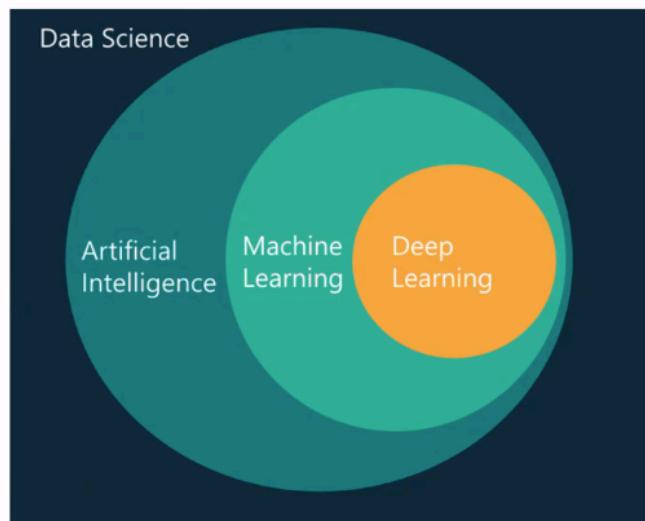
————— > Práctica



Programa tentativo

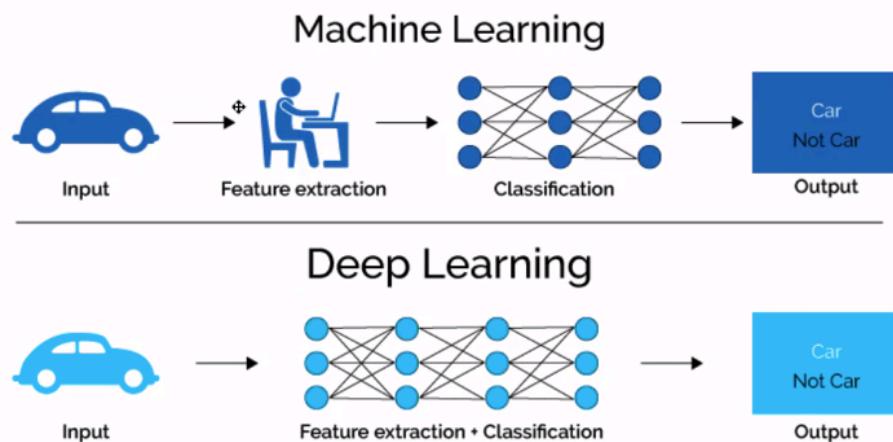
- 0 - ¿Qué es deep learning? (y codificación de un perceptrón simple en Python)
- 1 - Introducción al deep learning (TP opcional)
- 2 - Redes neuronales convolucionales (TP obligatorio, 28 de julio)
- 3 - Redes neuronales recurrentes (TP obligatorio, 11 de agosto)
- 4 - Redes neuronales de grafos (TP obligatorio, 25 de agosto)

¿Qué es deep learning?

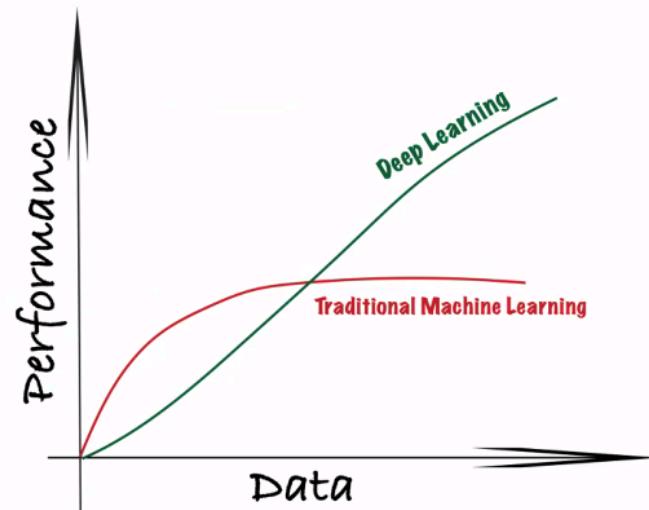


En el deep learning, el modelo aprende solo las características, mientras que en el deep learning no.

¿Qué es deep learning?

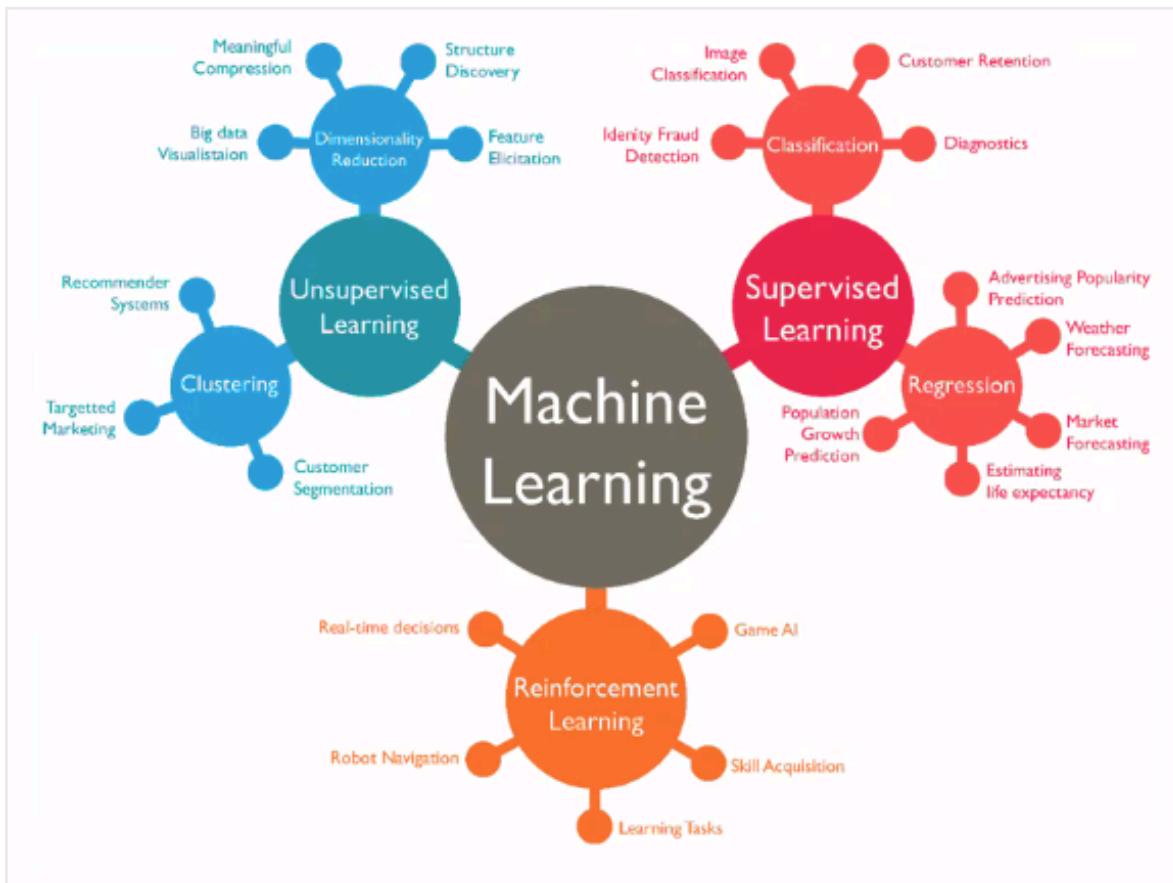


Performance deep learning vs machine learning



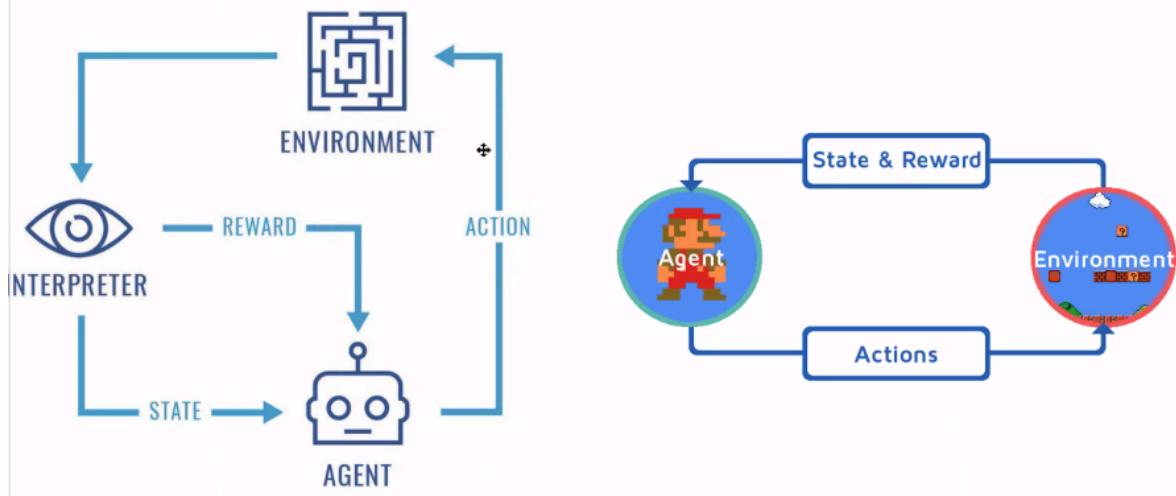
Por lo general un modelo de deep learning va a performa (a cierto nivel de datos) mucho mejor que uno de ML pero vamos a necesitar muchísimos mas datos. Se estanca el ML ponele. Esto no implica que si tengo una banda de data gane el DL, va a depender el tipo de dato y su naturaleza que modelo sea mejor.

Por eso es importante entender cuales son las herramientas correctas para los problemas.



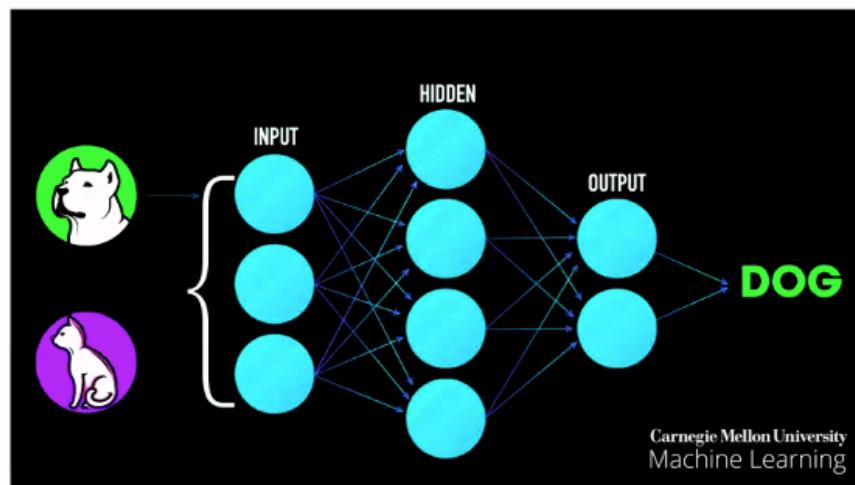
\$\$\$ Ojo, tenemos que tener en cuenta el costo de desarrollo e implementación de un modelo DL. Si bien ponele me puedo ahorrar teka en nose, operaciones fraudulentas de un banco, tengo que analizar si no me sale mas caro desarrollar, contratar, deployear y mantener un modelo de DL.\$\$\$\$

Reinforcement learning



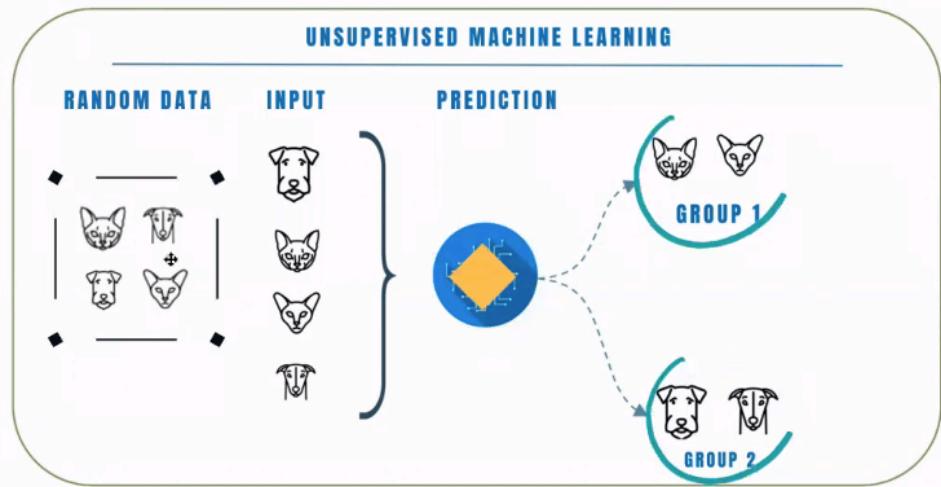
El reinforcement learning es mas tiempo real, le damos recompensas que simulan la dopamina en el cerebro. No lo vemos en el curso :(

Aprendizaje supervisado



[LINK - TensorFlow Playground](#)

Aprendizaje no supervisado



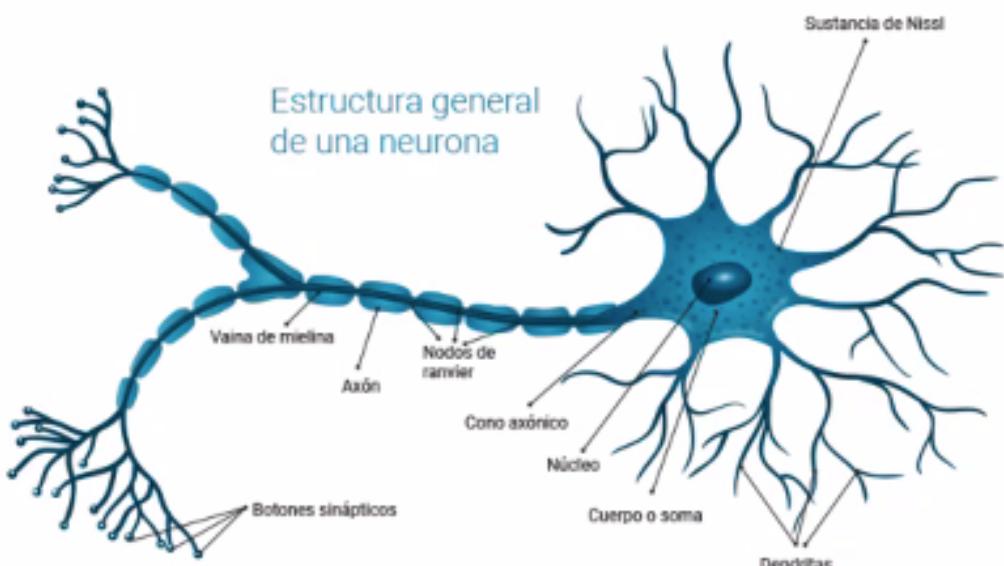
Tambien se usan redes neuronales para este tipo de clustering en DL.

> El perceptron simple

Es la unidad básica de una red neuronal. Las RN estan inspiradas en el cerebro, y el perceptron simple se inspira en una neurona.

Neurona biológica

- Corteza cerebral: 10^{11} neuronas
- Neurona biológica: soma, dendritas, axón
- Fisiología de la neurona:
 - Sinapsis, neurotransmisores
 - Despolarización
 - Propagación del impulso
 - Refuerzo de las sinapsis -> aprendizaje



- Las **dendritas**, que son la vía de entrada de las señales que se combinan en el cuerpo de la neurona. De alguna manera la neurona elabora una señal de salida a partir de ellas.
- El **axón**, que es el camino de salida de la señal generada por la neurona.
- Las **sinapsis**, que son las unidades funcionales y estructurales elementales que median entre las interacciones de las neuronas. En las terminaciones de las sinapsis se encuentran unas vesículas que contienen unas sustancias químicas llamadas **neurotransmisores**, que ayudan a la propagación de las señales electroquímicas de una neurona a otra.
- La neurona es estimulada o excitada a través de sus entradas y cuando se alcanza un cierto umbral, la neurona se activa, pasando una señal hacia el axón.

Modelo simplificado

El perceptrón ocupa un lugar especial en el desarrollo histórico de las redes neuronales: fue la primera red neuronal descrita algorítmicamente. Propuesto por Rosenblatt (1962), un psicólogo, que inspiró a ingenieros, físicos y matemáticos por igual a dedicar su esfuerzo de investigación a diferentes aspectos de las redes neuronales en las décadas de 1960 y 1970.

- Modelo unidimensional.
- Este modelo neuronal básico consiste en una combinación lineal de entradas (x) con pesos sinápticos (w) seguido de una función de activación.
- La función de activación "verifica" si la ponderación de entradas supera un determinado umbral.
- Todas las entradas llegan de forma simultánea.

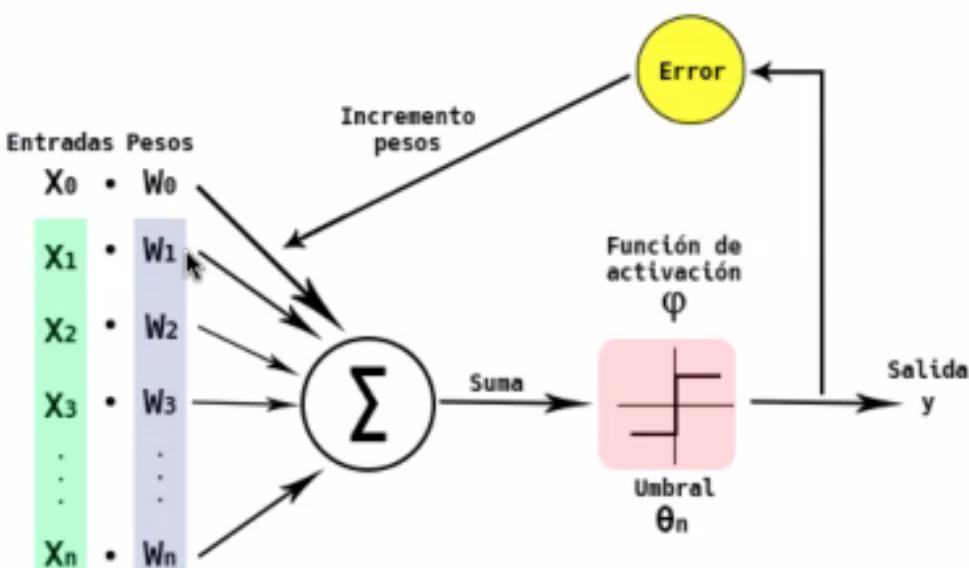
Características de la neurona

- Alta NO linealidad.
- Alto paralelismo.
- Aprenden a partir de los datos.
- Generalización y adaptabilidad.
- Robustez.

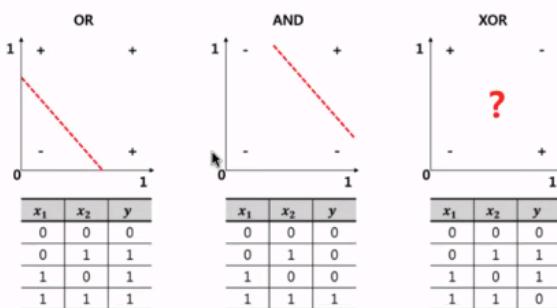
Función que mapea un vector de entrada a un valor de salida binario

$$f(x) = \begin{cases} 1 & \text{si } w \cdot x - u > 0 \\ 0 & \text{en otro caso} \end{cases}$$

- Donde w es un vector de pesos reales y $w \cdot x$ es el producto escalar. u es el umbral que representa el grado de inhibición de la neurona. Es un término constante que no depende del valor que tome la entrada.
- Espacialmente, el umbral (bias) altera la posición (aunque no la orientación) del límite de decisión.



- El algoritmo de aprendizaje del perceptrón no termina si el conjunto de aprendizaje no es linealmente separable .
- Si los vectores no son linealmente separables, el aprendizaje nunca llegará a un punto en el que todos los vectores se clasifiquen correctamente. El ejemplo más famoso de la incapacidad del perceptrón para resolver problemas con vectores linealmente no separables es el problema o exclusivo booleano



El perceptron hace uso del aprendizaje Hebbiano. Es básicamente ir

ponderando cada señal a través de los pesos.

Aprendizaje

El perceptrón hace uso del aprendizaje Hebbiano, es decir implica que los pesos sean ajustados de manera que cada uno de ellos represente la mejor relación entre ellos posibles (teniendo en cuenta las entradas y salida deseada).

Δw_i , proporcional al producto de una entrada x_i y de una salida y_i de la neurona.

Es decir, $\Delta w_i = \epsilon x_i y_i$ donde a $0 < \epsilon < 1$ se le llama tasa de aprendizaje

```
[ ]:  
1. Inicialización al azar  
2. Para cada ejemplo de datos de entrenamiento x[n] | y[n]  
    * Activación:  
        z = VInp.dot(pesos_sinapticos, x) + b) donde V es la función de activación utilizada  
    * Calcular error  
        error = y - z  
    * Actualización del vector de pesos sinápticos  
        w(t+1) = w(t) + ε * error * x           # ε es la tasa de aprendizaje  
        b(t+1) = b(t) + ε * error  
3. Volver a 2 hasta satisfacer algún criterio de fin.
```

Cada una de las iteraciones de aprendizaje se llama Epoca o Epoch en inglés. Ojo, el ejemplo del print de arriba NO es una implementación del descenso del gradiente. Es aprendizaje hebbiano. Otra cosa.