




# Aprendizaje Supervisado

Matías Marenchino - Cristian Cardellino



# Temario del Curso

- ¿Qué es aprendizaje supervisado?
- Repaso general de introducción al aprendizaje automático.
  - Regresión Lineal y Polinomial, Regresión Logística, Perceptrón.
- Árboles de decisión
- Naive Bayes
- Support Vector Machines
- Ensemble learning.
  - Random Forest, Bagging, Boosting.
- Redes neuronales.
  - Perceptrón multicapa.
- Sistemas de recomendación.
  - Filtrado colaborativo, máquinas de factorización.
- Prácticas de reproducibilidad

# Primera Clase

# Temario de la Clase

- ¿Qué es aprendizaje supervisado?
- Repaso general de introducción al aprendizaje automático.
  - Regresión Lineal y Polinomial, Regresión Logística, Perceptrón.
- Árboles de decisión
- Naive Bayes
- Support Vector Machines
- Ensemble learning.
  - Random Forest, Bagging, Boosting.
- Redes neuronales.
  - Perceptrón multicapa.
- Sistemas de recomendación.
  - Filtrado colaborativo, máquinas de factorización.
- Prácticas de reproducibilidad

# Motivación

---

# Ejemplos de aprendizaje supervisado

Una compañía de tarjetas de crédito (e.g. Visa, MasterCard) recibe nuevos pedidos para acceder a una tarjeta. Cada pedido tiene información acerca del aplicante:

- Sueldo
- Edad
- Estado Civil
- Veraz
- Situación crediticia según el BCRA
- ...

**Problema:** Determinar si aceptar o rechazar el pedido.

El servicio de correo de una universidad recibe cientos de mails por día.

**Problema:** Clasificar cada mail como correo basura (spam) o correo deseado, para filtrar y aligerar el servicio.

# Descripción del problema: Aprendizaje supervisado

**Datos:** Se dispone de un conjunto de registros (o ejemplos, o instancias) descritos por  $n$  atributos:  $A_1, A_2, \dots, A_n$  y cada instancia está anotada con una etiqueta o clase (e.g. Spam / No Spam), o un valor numérico (e.g. score crediticio).

**Objetivo:** Aprender un modelo (o función) a partir de los datos, buscando predecir sus etiquetas a partir de los atributos. Este modelo puede ser utilizado para predecir las etiquetas de nuevos registros sin anotar.

# Repaso de Aprendizaje Supervisado

---

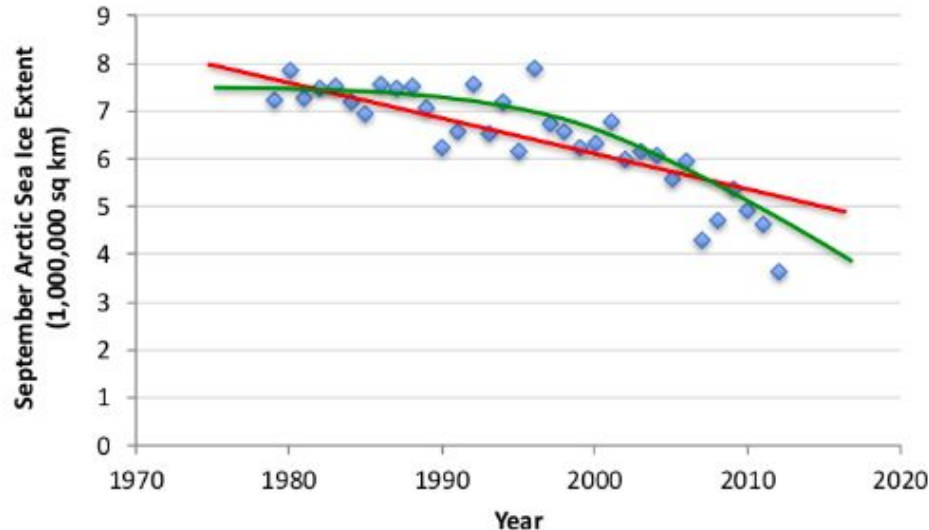


# Regresión

Dados  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

Aprender una  $f(x)$  que permita predecir  $y$  a partir de  $x$

Si  $y \in \mathbb{R}^n$ : Es un problema de **regresión**.



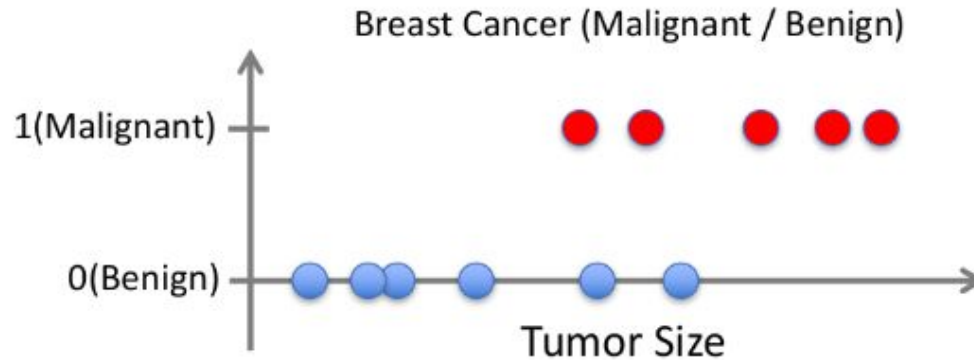
Filminas del  
curso previo

# Clasificación

Dados  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

Aprender una  $f(x)$  que permita predecir  $y$  a partir de  $x$

Si  $y$  es categórica: Es un problema de **clasificación**.

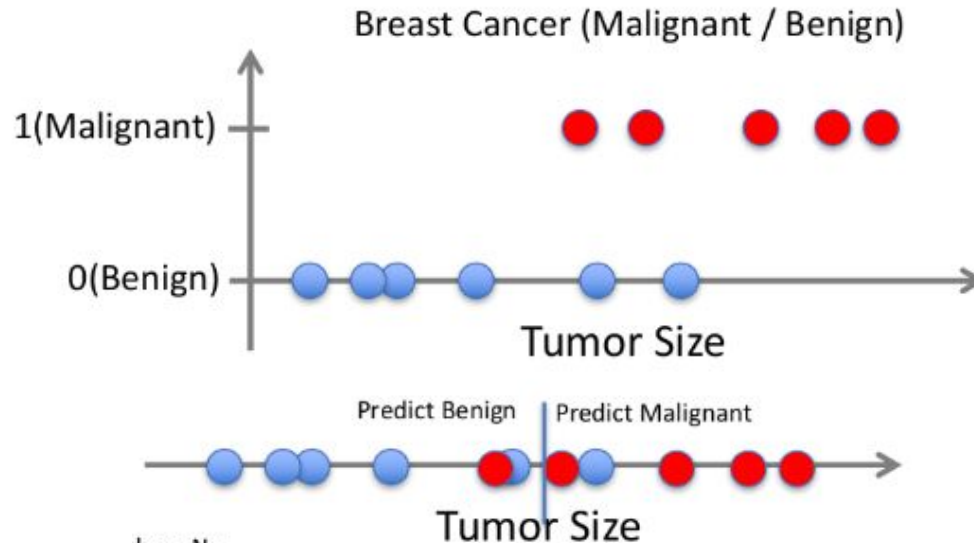


# Clasificación

Dados  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$

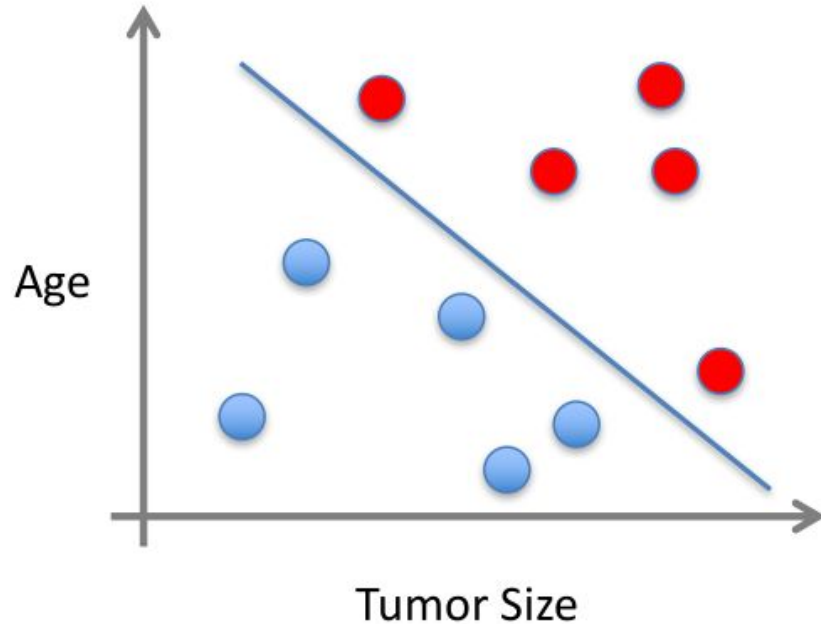
Aprender una  $f(x)$  que permita predecir  $y$  a partir de  $x$

Si  $y$  es categórica: Es un problema de **clasificación**.



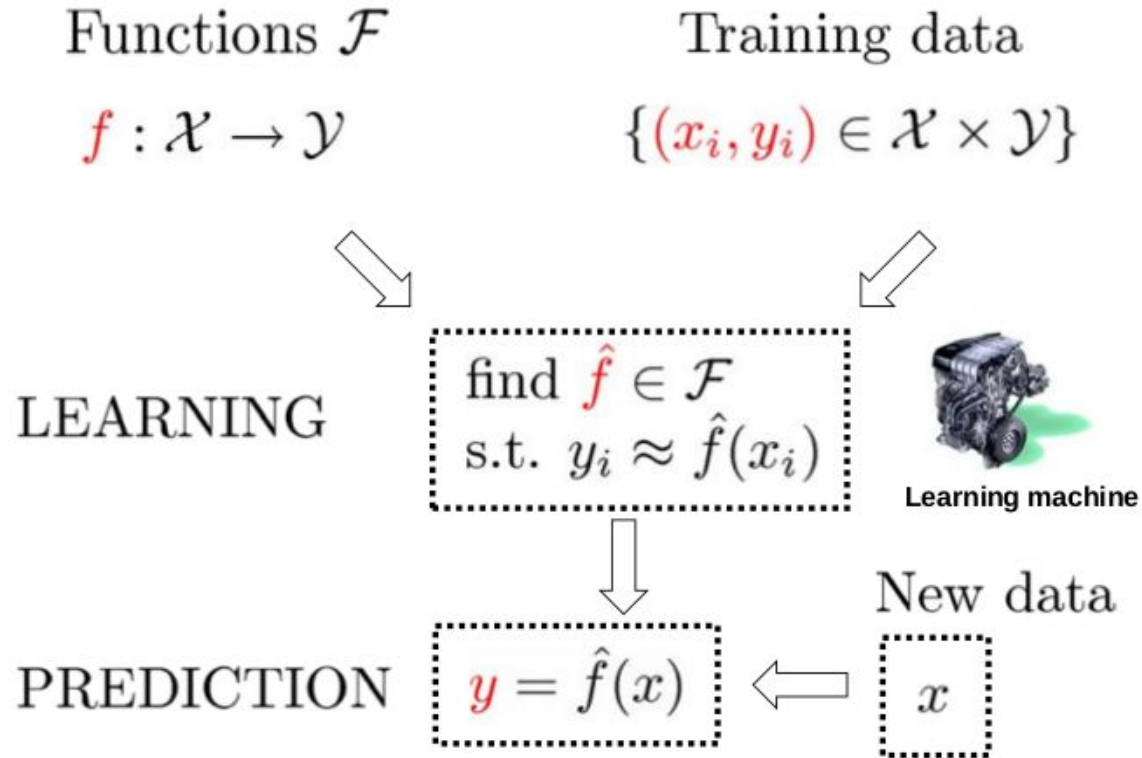
Filminas del  
curso previo

# Aprendizaje Supervisado



- La variable  $x$  puede ser multidimensional.
- Cada dimensión corresponde a un atributo:
  - Edad del paciente
  - Tamaño del tumor
  - Uniformidad en la forma de la célula
  - Etcétera
- La **regresión** busca “acercar” los datos a una función (lineal, polinomial, etc.)
- La **clasificación** busca separar los datos mediante ciertos “bordes”.

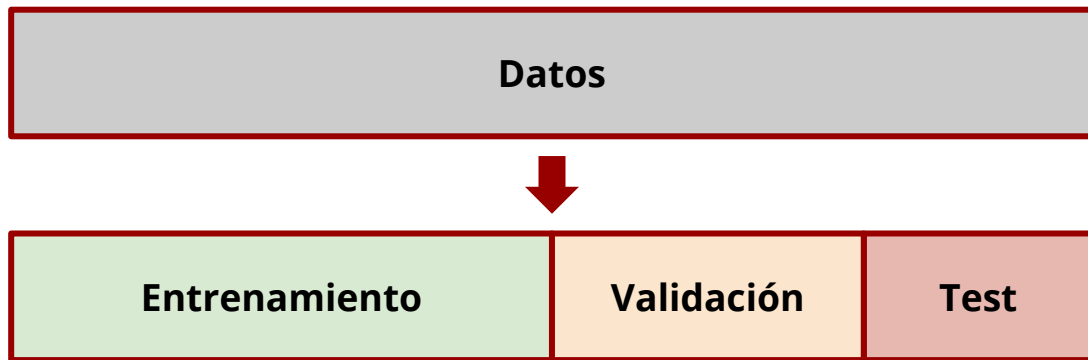
# Aprendizaje supervisado



# Elección de *hiperparámetros*

Dividir el conjunto total de ejemplos en tres subconjuntos

- **Entrenamiento:** aprendizaje de variables del modelo
- **Validación:** ajuste/elección de hiperparámetros
- **Test:** estimación final de la performance del modelo entrenado (y con hiperparámetros elegidos adecuadamente)



# Regresión Lineal y Polinomial

---

# Regresión Lineal

Busca ajustar los datos de entrenamiento mediante una función que sea una recta.

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \sum_{j=0}^d \theta_j x_j$$

Los valores  $\theta$  son los pesos de los atributos (o *features* en inglés).

Se entrena minimizando la suma del error cuadrado.

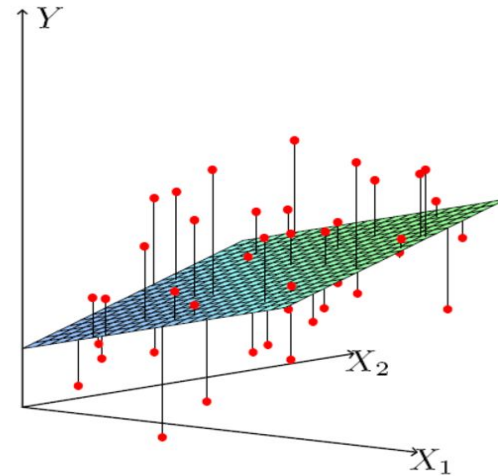
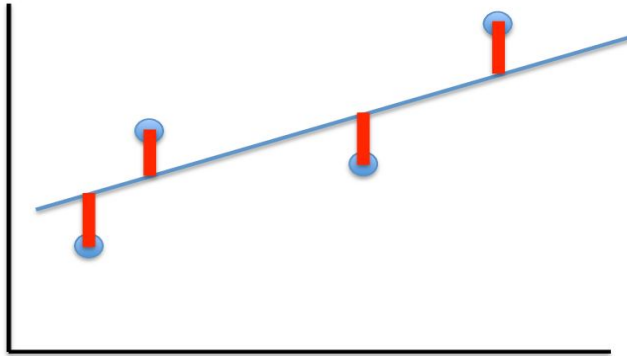


# Regresión Lineal: Función de costo

Función objetivo:

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left( h_{\boldsymbol{\theta}} \left( \mathbf{x}^{(i)} \right) - y^{(i)} \right)^2$$

Se resuelve mediante:  $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$



# Regresión Polinomial

Busca ajustar los datos de entrenamiento mediante una **función polinomial**:

$$y(x, \mathbf{w}) = w_0 + w_1x + w_2x^2 + \dots + w_Mx^M = \sum_{j=0}^M w_jx^j$$

Mientras **más alto el grado del polinomio**, más se ajusta a los datos (pero se vuelve más complejo y **tiende a sobreajustar**).

Demo Time (demo1)

# Regresión Logística

---

# Regresión Logística

Usa un enfoque probabilístico.

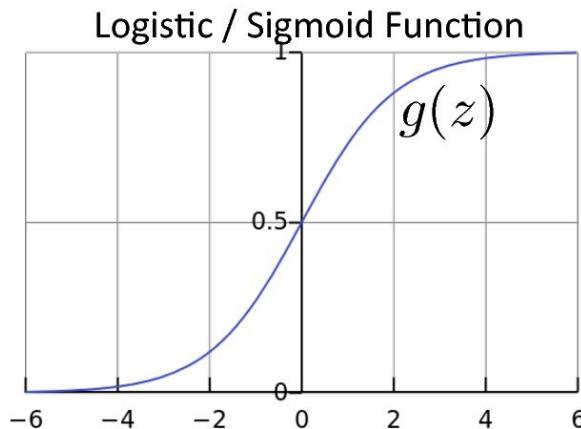
$h_{\theta}(\mathbf{x})$  debería ser  $p(y = 1 \mid \mathbf{x}; \theta)$

Modelo de regresión logística

$$h_{\theta}(\mathbf{x}) = g(\theta^{\top} \mathbf{x})$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

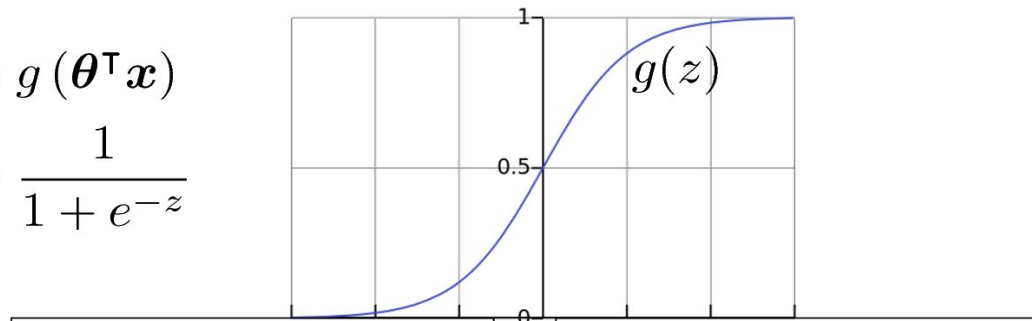
$$h_{\theta}(\mathbf{x}) = \frac{1}{1 + e^{-\theta^{\top} \mathbf{x}}}$$



# Regresión Logística

$$h_{\theta}(\mathbf{x}) = g(\theta^T \mathbf{x})$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

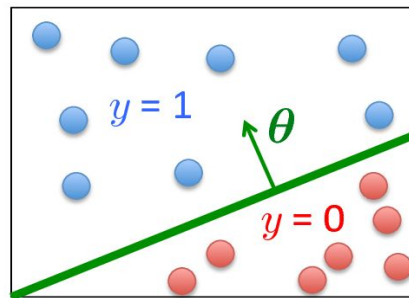


$\theta^T \mathbf{x}$  debería tener valores **negativos** grandes para instancias negativas y valores **positivos** grandes para instancias positivas.

Definir un umbral y...

Predecir  $y = 1$  si  $h_{\theta}(\mathbf{x}) \geq 0.5$

Predecir  $y = 0$  si  $h_{\theta}(\mathbf{x}) < 0.5$



# Regresión Logística: Función de costo

$$J(\boldsymbol{\theta}) = - \sum_{i=1}^n \left[ y^{(i)} \log h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) \right]$$

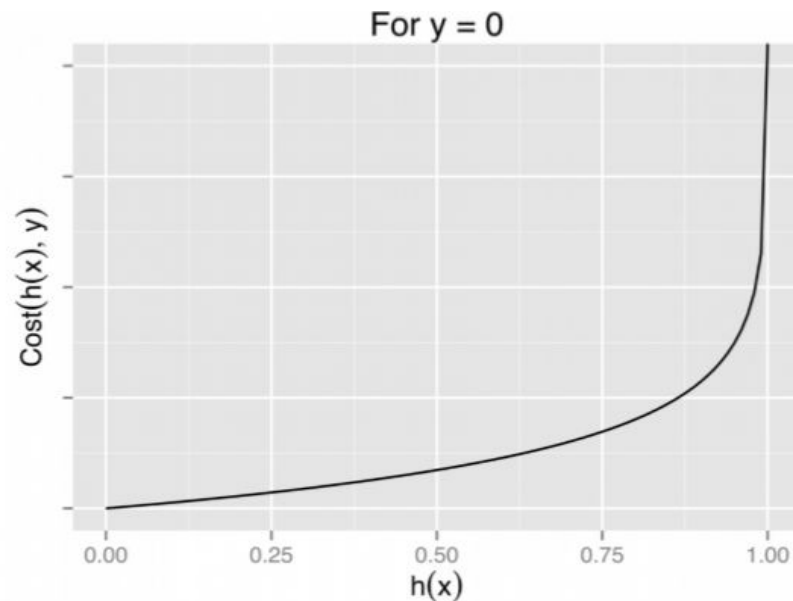
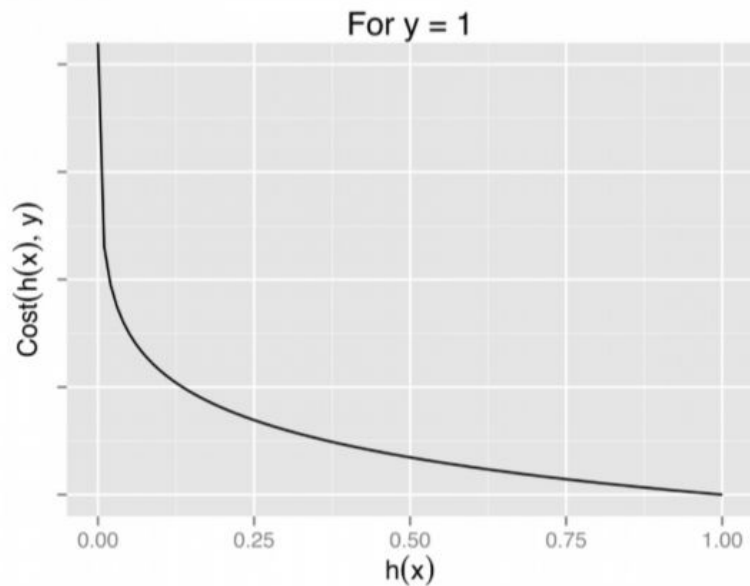
Costo de una sola instancia de los datos

$$\text{cost}(h_{\boldsymbol{\theta}}(\mathbf{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\mathbf{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\mathbf{x})) & \text{if } y = 0 \end{cases}$$

Se reescribe la función de coste como

$$J(\boldsymbol{\theta}) = \sum_{i=1}^n \text{cost}(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), y^{(i)})$$

# ¿Por qué esta función de costo?





Demo Time (demo2)

# Algoritmo del perceptrón

---

# El algoritmo del "perceptrón"

Propuesto por Roseblatt en 1958

El objetivo es encontrar un hiperplano de separación

Sólo encuentra la solución si los datos son linealmente separables

Es un algoritmo *online* (procesa un ejemplo a la vez)

# El algoritmo del "perceptrón"

Entrada:

- Una conjunto de entrenamiento  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Una tasa de aprendizaje  $r$

Algoritmo:

- Inicializar  $w^{(0)} \in \mathbb{R}^n$
- Para cada ejemplo  $(x_i, y_i)$ 
  - Predecir  $y_i' = \text{signo}(w^T x_i + w_0)$
  - Si  $y_i' \neq y_i$ :
$$w^{(t+1)} \leftarrow w^{(t)} + r (y_i x_i)$$

# El algoritmo del "perceptrón"

Entrada:

- Una conjunto de entrenamiento  $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$
- Una tasa de aprendizaje  $r$

Algoritmo:

- Inicializar  $w^{(0)} \in \mathbb{R}^n$
- Para cada ejemplo  $(x_i, y_i)$ 
  - Predecir  $y_i' = \text{signo}(w^T x_i + w_0)$
  - Si  $y_i' \neq y_i$ :
$$w^{(t+1)} \leftarrow w^{(t)} + r (y_i x_i)$$

Actualiza solo cuando comete un error

Error en positivos:

$$w^{(t+1)} \leftarrow w^{(t)} + r x_i$$

Error en negativos:

$$w^{(t+1)} \leftarrow w^{(t)} - r x_i$$

Si  $y_i w^T x_i \leq 0 \rightarrow \text{error}$

Filminas del  
curso previo

Demo Time (demo3)

# Árboles de decisión

# Árboles de decisión

Aprende a diferenciar los datos en base a reglas de decisión.

Los nodos del árbol representan las reglas. Las hojas asignan la clase o el valor.

El árbol se particiona recursivamente. Para obtener el resultado, simplemente se siguen los nodos de decisión de acuerdo a los datos y se asigna la clase final.

Es un algoritmo de “caja blanca”, ya que puede visualizarse fácilmente e interpretarse por los humanos (a diferencia de un algoritmo de “caja negra” como son las redes neuronales).

Son buenos con datos de mucha dimensionalidad (high dimensional data)



# Árboles de decisión: Algoritmo

Selecciona el mejor atributo de acuerdo a alguna métrica (e.g. ganancia de información).

Hacer un nodo de decisión con ese atributo, que particione los datos en subconjuntos de menor tamaño.

Repetir recursivamente el procedimiento para cada nodo hijo.

El algoritmo se detiene si:

- Todos los ejemplos del subconjunto son de la misma clase.
- Todos los elementos del subconjunto son constantes con respecto al atributo/s de interés del nodo actual.

# Árboles de decisión: Métodos de decisión

Definen la forma de particionar el conjunto de datos (es una heurística).

Buscan rankear cada atributo de acuerdo a ciertos parámetros.

Los criterios más populares son:

- **Information Gain:** calcula la diferencia entre la entropía del conjunto antes de dividirse y la entropía media luego de dividir el conjunto de datos, para cada atributo.
- **Gini Index:** Es una métrica que mide cuánto un elemento del subconjunto elegido al azar sería identificado incorrectamente. Busca los atributos que tienen menor índice de Gini.
- **Mean Squared Error:** Este método sirve para casos de regresión (en lugar de clasificación). Busca minimizar el error cuadrático medio en los nodos hijos a la hora de particionar los datos.

Demo Time (demo4)

# Naive Bayes

---

# Naive Bayes

Es un clasificador basado en el teorema de Bayes, con una asunción “naive” sobre los datos.

Es muy sencillo de programar y entender.

Sirve mucho como baseline y, aunque simplista, puede tener resultados que sobrepasan a algoritmos mucho más complejos.

Es rápido de entrenar y funciona con datos de mucha dimensionalidad (e.g. es muy útil a la hora de clasificar documentos).

# Teorema de Bayes

El algoritmo de “Naive Bayes” está fuertemente ligado al teorema de Bayes.

El Teorema de Bayes establece:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

El teorema establece que se puede encontrar la probabilidad de **A** (e.g. una clase objetivo) dada la ocurrencia de **B** (e.g. un conjunto de features). Es decir, **B** es la evidencia y **A** es la hipótesis.

# Naive Bayes: Asunciones

Bajo el teorema de Bayes, la principal asunción es que los atributos son independientes entre sí.

La presencia de un feature no afecta a los otros. Esta asunción es “naive”, por eso el nombre del algoritmo.

Una segunda asunción, es que todos los atributos tienen el mismo efecto en la salida del algoritmo.

# Naive Bayes: Utilizando el Teorema de Bayes

En base a lo establecido, se puede utilizar el teorema de Bayes para calcular la probabilidad de una clase  $y$  de la siguiente manera:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

Donde  $y$  representa la clase y  $X$  representa el vector de atributos:

$$X = (x_1, x_2, x_3, \dots, x_n)$$



# Naive Bayes: Utilizando el Teorema de Bayes

Utilizando sustitución en la fórmula anterior obtenemos:

$$P(y|x_1, \dots, x_n) = \frac{P(x_1|y)P(x_2|y)\dots P(x_n|y)P(y)}{P(x_1)P(x_2)\dots P(x_n)}$$

Dado que el denominador es estático para todas las entradas del conjunto de datos se puede ignorar y se establece una proporcionalidad:

$$P(y|x_1, \dots, x_n) \propto P(y) \prod_{i=1}^n P(x_i|y)$$

# Naive Bayes: Predicción de la clase

En base a lo visto previamente, la predicción de la clase objetivo es sencillamente aquella con mayor probabilidad:

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y)$$

# Naive Bayes: Algoritmo

- Calcular la probabilidad para cada clase

$$P(y) / \forall y \in Y$$

- Calcular la probabilidad condicional de cada atributo dada cada clase:

$$P(x_i|y) / 0 \leq i \leq n, \forall y \in Y$$

- Calcular la clase de acuerdo a la que maximice la probabilidad (o, en este caso la proporción):

$$y = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y)$$

# Naive Bayes: Tipos de algoritmos

**Bernoulli Naive Bayes:** Para casos donde los atributos son variables booleanas (e.g. si una palabra ocurre o no en un documento).

**Multinomial Naive Bayes:** Para casos donde los atributos representan frecuencias (e.g. la cantidad de veces que una palabra ocurre en un documento).

**Gaussian Naive Bayes:** Para casos donde los atributos toman valores continuos, se asume que los valores son muestras de una distribución gaussiana (esto se usa para calcular las probabilidades condicionales en el algoritmo).



Fin de la primera clase

