Inteligencia Artificial

Víctor Mijangos de la Cruz

V. Aprendizaje automático

Aprendizaje automático

Antecedentes

Algunos puntos históricos relevantes en la historia del aprendizaje son:

- En el Siglo XVII, Gottfried Leibnez propone una marco teórico para llevar a cabo operaciones lógicas (*calculus ratiocinator*).
- En 1887, Charles S. Pierce habla de máquinas lógicas.
- En los 40's, McCulloch y Pitts proponen un modelo lógico para emular la actividad nerviosa.
- En los 50's, Rosenblatt plantea los fundamentos del análisis de reconocimiento de patrones.
- En los 60's, comienza el desarrollo de modelos dentro de la teoría de aprendizaje.
- En los 80's, se establecen métodos estadísticos para abordar los problemas de este tipo.

¿Qué es el aprendizaje automático?

Consideremos dos problemas y veamos qué soluciones tienen:

Tarea: Calcular la propina de un servicio

1. $x :=$ total de consumo
2. $p :=$ porcentaje deseado de propina
3. Propina por servicio: $x \cdot 0.p$

Tarea: Detectar spam

1. Depende del usuario
2. Contamos con ejemplos por cada usuario
3. ¿Qué algoritmo podemos describir para detectar spam?

No es fácil definir un algoritmo para detectar spam, pero podemos hacer que la máquina aprenda a hacerlo a partir de ejemplos.

Deducción e inferencia

La **deducción** genera conocimiento partiendo de lo general y dirigiédose a lo particular.

Generalmente, se parte de **premisas**, de los que se derivan en una **conclusión**.

Un ejemplo típico de deducción son los silogismos:

A1: Todos los hombres son mortales
A2: Sócrates es un hombre
C: Sócrates es mortal

La **inducción** parte de lo particular y se dirige a lo general.

La inducción recolecta **datos**, de los cuales extrae **patrones** que nos llevan a concluir algo sobre los datos en general.

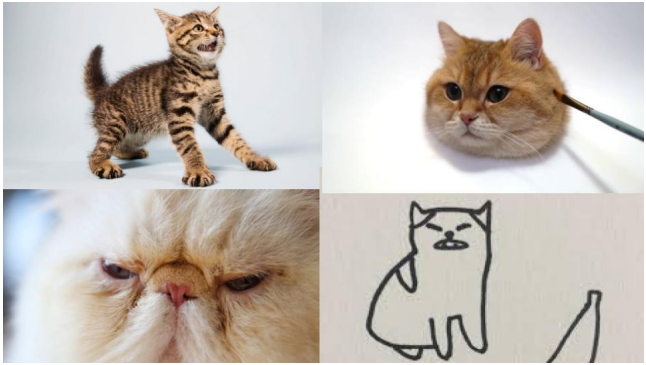
Mientras que el conocimiento deductivo es verdadero (siempre que las premisas sean verdaderas) el conocimiento inductivo es **probabilístico**.

Necesidad de métodos inferenciales

- Existen tareas en las que determinar un algoritmo resulta una tarea sencilla.
- Sin embargo, existen tareas en que no es fácil determinar un algoritmo. Causas de esto son:
  - No existe *expertise* humano.
  - La solución puede cambiar con el tiempo o responder a casos particulares.
  - Hay tareas que, a pesar de realizarse de manera cotidiana, no somos capaces de describirlas.

Necesidad de métodos inferenciales

Imagínese una tarea en la que debe clasificarse un objeto como 'es gato' y 'no es gato'. ¿Qué tipo de algoritmo podría realizar esta tarea?



Necesidad de métodos inferenciales

Una opción alternativa: Podemos dejar que la máquina observe una serie de ejemplos y extraiga las categorías relevantes.

Justificación: Los datos son abundantes y baratos, el *conocimiento* es caro y escaso.

Objetivo: Construir modelos generales a partir de ejemplos particulares (**inducción**).

Diremos, entonces, que buscamos que la máquina **aprenda** a partir de estos ejemplos.

¿Qué es el aprendizaje automático?

El **Aprendizaje Automático** nos permite inducir un resultado a partir de instancias de entrenamiento. Podemos definir aprendizaje como (Mitchell, 1997):

Aprendizaje automático

Se dice que una máquina aprende de la **experiencia** E, si su **desempeño** con respecto a una medida P en una **tarea** T mejora con la experiencia E.

Surge la pregunta:

> *¿Cómo podemos construir sistemas computacionales que mejoren en una tarea a partir de la experiencia? ¿Cuáles son las leyes fundamentales que goviernan el "aprendizaje"?*

Una aproximación al problema del aprendizaje

Se definen los siguientes elementos en el problema del aprendizaje:

1. Una tarea T (el problema al que nos enfrentamos).
2. Una medida de desempeño P (cómo evaluar qué también lo hace la máquina).
3. Experiencia E (datos, ejemplos).

Entonces, el problema de aprendizaje se puede definir como (Mithcell, 1999):

> *Un programa de computadora se dice que aprende de la experiencia E con respecto a una tarea T y una medida de desempeño P, si su desempeño con respecto a T, medido con P, mejora con la experiencia E.*

Ejemplo de Aprendizaje

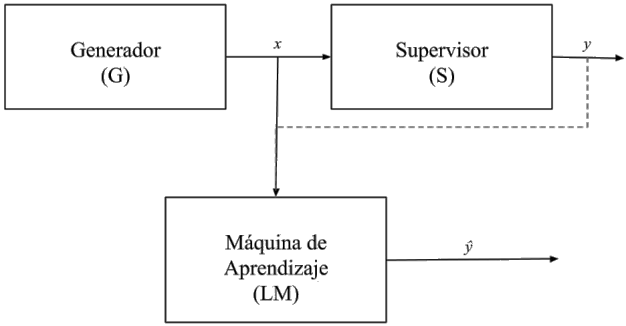
- **Diagnóstico médico**
  T: Sugerir un tratamiento dado los síntomas
  P: % de predicciones de tratamiento que fueron correctas
  E: Una base de datos, registros médicos que contengan los síntomas del paciente y la enfermedad diagnosticada.

- **Reconocimiento de escritura**

  T: Reconocer y clasificar palabras dentro de una imagen
  P: % de palabras correctamente clasificadas
  E: Una base de datos con imágenes de palabras escritas a mano, cada una asociada a una etiqueta indicando qué palabra es.

Modelo general de aprendizaje a partir de ejemplos

A partir de los elementos citados, se puede proponer un modelo general de aprendizaje a partir de ejemplos (Vapnik, 1998):



Modelo general de aprendizaje a partir de ejemplos

Este modelo cuenta con los siguientes elementos:

1. **Generador**: Se encarga de generar la experiencia (ejemplos) a partir de una distribución $p(x)$.
2. **Supervisor**: Determina la clase a la que pertenece un ejemplo. Se trata de conocimiento experto.
3. **Máquina de Aprendizaje**: Es una función que toma un ejemplo de entrada y le asigna una clase con base en una distribución $q(x)$.

Tipos de Aprendizaje

Dado un conjunto de datos $X = \{x : x \in \mathbb{R}^d\}$, la máquina de aprendizaje es una función $f : X \to \mathbb{R}$, tal que:

$$f(x) = \hat{y} \tag{1}$$

Podemos clasificar los modelos de aprendizaje como:

Supervisado: Tenemos un conjunto supervisado $\mathcal{S} = \{(x, y) : x \in \mathbb{R}^d, y \in Y\}$, tal que cada $x \in X$ se asocia a un $y \in Y$ y $\hat{y} = f(x)$ busca ser lo más cercano a cada y.

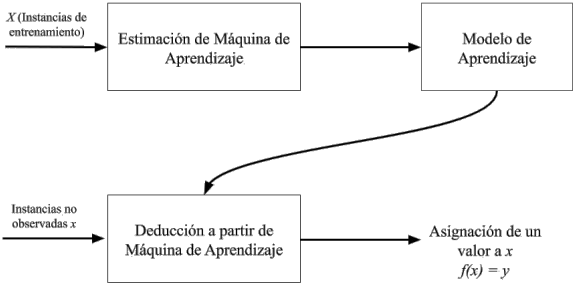
No-supervisado: La LM no conoce la supervisación. $f(x)$ es una función de agrupamiento, asigna una clase $\hat{y}$ que no se conoce previamente.

Otros tipos de aprendizaje son: aprendizaje por refuerzo, aprendizaje semi-supervisado, aprendizaje auto-supervisado.

Elementos de un problema de aprendizaje

Durante una tarea de aprendizaje, se tienen dos etapas:

1. **Entrenamiento**: Se estima la función f a partir de una muestra de ejemplos.
2. **Evaluación**: Se determina la capacidad de generalizar del modelo de aprendizaje.



Elementos de un problema de aprendizaje

Para llevar a cabo el entrenamiento y la evaluación, generalmente, se dividen los datos en 3 subconjuntos:

1. **Entrenamiento**: El subconjunto de datos que servirá para estimar el modelo de aprendizaje (70%).
2. **Validación**: Es un subconjunto de datos que se prueba constantemenete para ajustar ciertos requerimiento del modelo de aprendizaje (10-15%).
3. **Evaluación**: Es el subconjunto a partir del cuál se determina la capacidad de generalizar del modelo estimado (15-20%).

Entrenamiento y validación

El **entrenamiento** consiste en tomar un conjunto de datos empíricos $X = \{x : \text{ es un dato empírico}\}$ y obtener un modelo:

$$X \mapsto \mu$$

A este modelo se le llama **modelo de aprendizaje**. Este modelo se obtiene al minimizar una **función objetivo**.

La **validación** sirve para ajustar **hiperparámetros** del modelo.

Hiperparámetro

Un hiperparámetro es un parámetro del modelo de aprendizaje que no es aprendido, si no que es determinado por el programador.

Minimización del riesgo empírico

En general, el problema de aprendizaje se convierte en un **problema de optimización**. Para esto, se define una función $L : \Theta \to \mathbb{R}^+$, que nos diga cuánto se equivoca (Θ espacio de hipótesis), y se determina la minimización del riesgo empírico:

Minimización del riesgo empírico

Dado una función de perdida $L : \Theta \to \mathbb{R}^+$, donde Θ es el espacio de hipótesis, definimos la minimización del riesgo empírico como:

$$\arg\min_{\theta} R_E(\theta) = \arg\min_{\theta} \sum_{i=1}^{|X|} L(f(x_i), y_i; \theta) \tag{2}$$

El problema de la minimización del riesgo empírico consiste en minimizar esta función para obtener un modelo de aprendizaje a partir de la hipótesis que minimiza esta función.

Generalización

La **evaluación** consiste en determinar la **capacidad de generalización** que tiene nuestro modelo de aprendizaje.

Se utiliza una **función de desempeño** que nos dice la capacidad de generalización, y está determinadad como:

$$R_G(\theta) = \int_{\mathbb{R}} L(f(x), y; \theta) dF(x) \tag{3}$$

Esta función integra sobre el espacio del problema, lo que en la práctica no es posible cuando no conocemos el problema en general.

En la práctica, tratamos de aproximar la función de generalización a partir de evaluar **medidas de desempeño** sobre un conjunto finito de datos.

Clases predichas y clases reales

El modelo de aprendizaje, predice una clase $\hat{y} = f(x)$, llamaremos a esta la **clase predicha**.

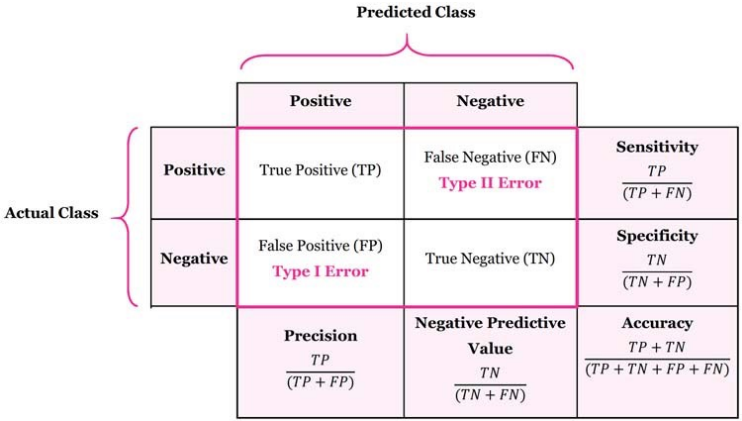
La **clase real** y es la clase real que es la clase que le pertenece al objeto x.

En base a esas clases, podemos definir los siguientes casos:

- **Verdaderos Positivos (TP)**: los casos en que x pertenecía a la clase real.
- **Verdaderos Negativos (TN)**: los casos en que se predijo que x no pertenecía a una clase en que realmente no pertenecía.
- **Falsos positivos (FP)**: los casos en que se predijo que x pertenecía a una clase que no pertenecía.
- **Falsos negativos (FN)**: los casos en que se predijo que x no pertenecía a una clase que realmente sí pertenecía.

Matriz de confusión

A partir de los casos anteriores se puede crear la **matriz de confusión** y calcular las siguientes medidas:



Ejemplo de evaluación

Supongamos que tenemos un conjunto de evaluación supervisado y una predicción de clases dada por la máquina de la siguiente forma:

Número de ejemplo	Observación	Predicción
1	1	1
2	1	1
3	1	0
4	1	1
5	1	0
6	0	0
7	0	1
8	0	1
9	0	0
10	0	1

Ejemplo de evaluación

A partir de las observaciones del cuadro anterior, obtenemos la siguiente matriz de confusión:

	Positivos	**Negativos**
Positivos	3	2
Negativos	3	2

A partir de esta matriz de confusión podemos obtener diferentes métricas de evaluación. El **Accuracy** es:

$$\begin{aligned} Acc &= \frac{VP + VN}{VP + FN + FP + VN} \\ &= \frac{3 + 2}{3 + 2 + 3 + 2} \\ &= \frac{5}{10} = 0.5 \end{aligned}$$

Ejemplo de evaluación

Para las métricas de precisión y recall tenemos:

- **Precisión:**

$$Prec = \frac{VP}{VP + FP} = \frac{3}{6} = 0.5$$

- **Recall:**

$$Rec = \frac{VP}{VP + FN} = \frac{3}{5} = 0.6$$

Ejemplo de evaluación

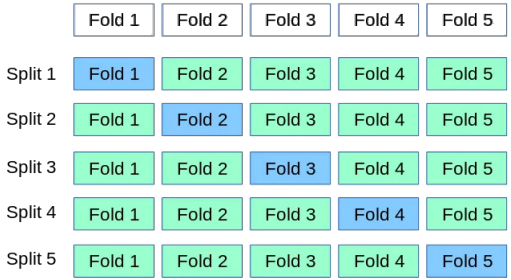
Finalmente, para la **medida $\mathbf{F_1}$** tenemos:

$$
\begin{aligned}
F_1 &= 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec} \\
&= 2 \cdot \frac{0.5 \cdot 0.6}{0.5 + 0.6} \\
&= 2 \cdot \frac{0.3}{1.1} && \approx 0.545
\end{aligned}
$$

Holdout y Validación cruzada

Al método que hemos descrito para realizar la evaluación se le conoce como **Holdout**: Sólo se evalúa el conjunto de datos una vez.

Un método que busca representar mejor la generalización es el de **validación cruzada** por **k-folds**: Se define un número k de folds. El subconjunto de evaluación se **alterna** en k iteraciones.



Evaluación no supervisada

Cuando se realiza evaluación sobre conjuntos no supervisados, generalmente se cuenta con un conjunto **gold standard** $C = \{c_1, .., c_k\}$ y clústers determinados por la máquina $\hat{C} = \{\hat{c}_1, ..., \hat{c}_k\}$.

- **Pureza:**

$$Purity(C, \hat{C}) = \frac{1}{N} \sum_i \max_j |\hat{c}_i \cap c_j|$$

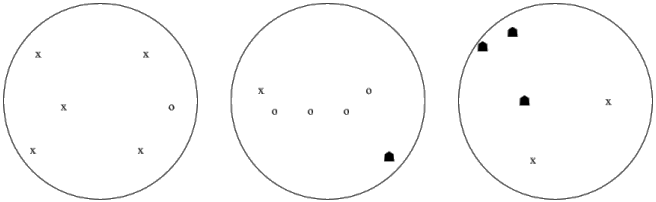
- **Mutual information:**

$$MI(C, \hat{C}) = \sum_i \sum_j p(\hat{c}_i \cap c_j) \log \frac{p(\hat{c}_i \cap c_j)}{p(\hat{c}_i) p(c_j)}$$

- **Normalized Mutual Information:**

$$NMI(C, \hat{C}) = 2 \cdot \frac{MI(C, \hat{C})}{H(C) + H(\hat{C})}$$

Evaluación no supervisada



$$
\begin{aligned}
\mathit{Purity}(C, \hat{C}) &= \frac{1}{N} \sum_i \max_j |\hat{c}_i \cap c_j| \\
&= \frac{1}{17}(4 + 4 + 3) \\
&= \frac{11}{17} \approx 0.647
\end{aligned}
$$

Aprendizaje y generalización

En resumen, tenemos los siguientes casos:

Entrenamiento: A partir de datos, se minimiza la función de riesgo para obtener un modelo de aprendizaje:

$$R_E(\theta) = \sum_{i=1}^{|X|} L(f(x_i), y_i; \theta) \tag{4}$$

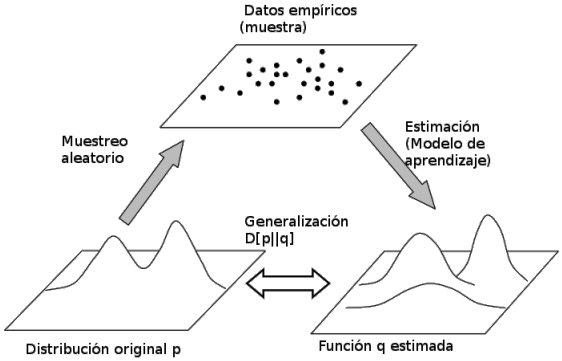
Generalización: Se busca ver la capacidad de generalización:

$$R_G(\theta) = \int_{\mathbb{R}} L(f(x), y; \theta) dF(x) \tag{5}$$

En la práctica se utilizan aproximaciones a esta función.

Aprendizaje y generalización

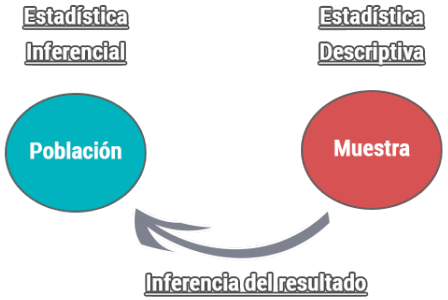
A partir de un conjunto finito de datos X, se estima una función que busca estimar un modelo de aprendizaje que corresponde a los datos.



Estadística

La estadística se divide en:

- **Estadística descriptiva:** La estadística descriptiva busca **describir** un conjunto de datos (muestra) con el fin de organizarlos y presentarlos.
- **Estadística inferencial:** Busca determinar, por medio de la inducción, propiedades de un conjunto de datos (muestra), que describan la familia de datos de forma **general**.



Aprendizaje estadístico

La teoría del aprendizaje estadístico (y sus aplicaciones) tiene sus raíces en el **análisis estadístico**.

El problema de la inferencia a partir de ejemplos puede plantearse como (Vapnik, 1998):

> *Dado una colección de datos (empíricos) originados de una dependencia funcional, inferir esta dependencia.*

Se proponen dos acercamiento a la solución de este problema:

1. **Inferencia particular (paramétrica)**: Busca estimar un número finito de parámetros que describan los datos de un problema particular. Asume una distribución (generalmente normal).
2. **Inferencia general (no paramétrica)**: Busca encontrar un método para aproximar una función a partir de los ejemplos, sin asumir una familia específica de distribuciones.

Métodos paramétricos y no paramétricos

Algunos métodos paramétricos de aprendizaje son:

- Métodos de regresión
- Bayes ingenuo
- Perceptrón

Algunos métodos no-paramétricos de aprendizaje son:

- Árboles de decisión
- k-NN
- Redes neuronales profundas

Teoría de aprendizaje estadístico

La **teoría de aprendizaje estadístico** puede entenderse como la parte teórica del aprendizaje de máquina. Se enfoca en determinar las condiciones en que se puede encontrar una buena aproximación de una **función de predicción** a partir de **datos empíricos**.

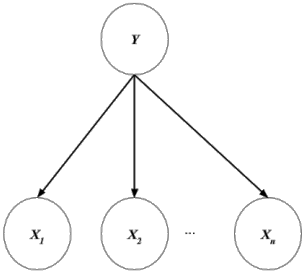
Los elementos esenciales para trabajar con esta teoría son:

- X, un conjunto de datos empíricos en $\mathbb{R}^d$.
- Y, un conjunto de clases, generalmente subconjunto de $\mathbb{R}$.
- La función $q : X \to Y$, tal que $q(x) \sim y$, $x \in X, y \in Y$.

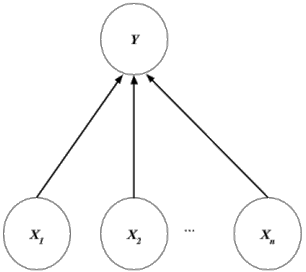
Modelos generativos y discriminativos

Generalmente la función de predicción q puede verse como una función de probabilidad entre $x \in X$ y $y \in Y$. A partir de la forma de estimar esta probabilidad, se tienen dos familias de modelos:

1. **Modelos generativos:** Estiman: $q(y, x) = q(x|y)q(y)$
2. **Modelos discriminativos:** Estiman: $q(y|x)$



Bayes naïve (ingenuo)



Perceptrón

Función de riesgo

Podemos expresar la función de riesgo (tanto empírico como de generalización) como:

$$R(q) = \mathbb{E}_p[L(X, q(X))] \tag{6}$$

En la teoría de aprendizaje estadístico se reconocen dos clases de **problemas**:

1. **Problema de regresión:** Está determinado por

$$L = ||y - q(x)||^2$$

2. **Problema de clasificación:** Está determinado por

$$L = \begin{cases} 1 & \text{si } q(x) \neq y \\ 0 & \text{si } q(x) = y \end{cases}$$

Parte de la teoría de aprendizaje

La teoría de aprendizaje se puede dividir en las siguientes partes (Vapnik, 1998):

1. **Teoría de consistencia:** Busca encontrar las condiciones para que el riesgo de entrenamiento sea igual (o lo más cercano) al riesgo de generalización.
2. **Teoría de cotas:** Busca obtener las cotas en la habilidad de generalización de las máquinas de aprendizaje.
3. **Teoría de control de generalización:** Busca encontrar los mejores métodos que permitan determinar la capacidad de generalizar de una ML a partir de datos empíricos.
4. **Teoría de algoritmos:** Se ocupa de desarrollar algoritmos de máquinas de aprendizaje.

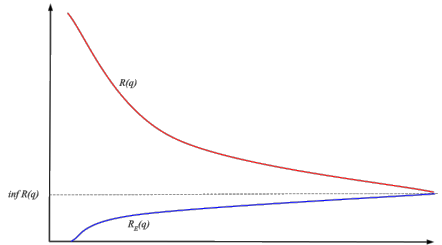
Consistencia

Consistencia

Se dice que un proceso de aprendizaje es consistente si el riesgo de entrenamiento y generalización convergen al ambos a un valor mínimos. Esto es:

$$\inf_q R_E(q) \xrightarrow{P} \inf_q R(q) \tag{7}$$

Cuando $N \to \infty$, siendo N el número de ejemplos.



Convergencia por datos empíricos

Un problema al que nos enfrentamos es determinar qué tantos datos son necesarios para qué se dé la convergencia que pedimos.

Si bien no es factible determinar esto, tenemos resultados que nos dan información sobre el poder de generalización:

$$P(\sup |R(q) - R_E(q)| > \epsilon) \leq 2e^{-2\epsilon^2 N} \tag{8}$$

Esta desigualdad nos hace ver que buscar minimizar el error de evaluación (en base al de entrenamiento) requiere que el número de ejemplos crezca.

Overfiiting

Un problema común en el aprendizaje es el del overfitting.

Overfiting

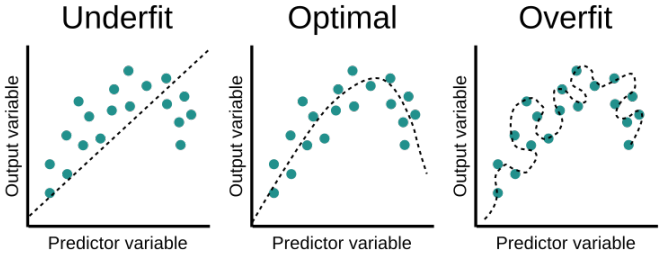
Se da overfitting cuando, en el entrenamiento, se estima una función q que se sobreajusta a los datos ($R_E(q)$ bajo), de tal forma que no es capaz de generalizar ($R(q)$ alto).

Cuando la estimación de la ML es pobre, se habla de **underfitting**.

Overfiiting

El overfitting (y underfitting) dependen en gran medida de la **capacidad** de una ML para tener una buena generalización.

- Una ML con 'poca potencia' no podra representar los datos adecuadamente (undefitting).
- Una ML con 'mucha potencia' requerirá de muchos datos para no sobre representar la muestra de entrenamiento (overfitting).

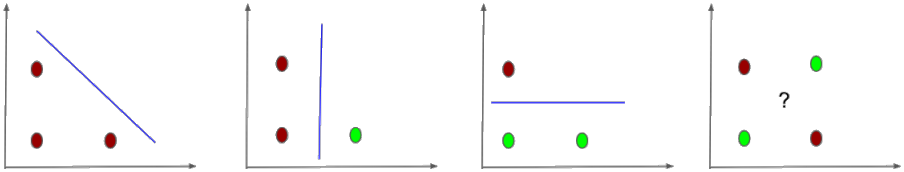


Por tanto, requerimos de un concepto de **capacidad** o potencia.

Dimensión de Vapnik-Chervonenski

La dimensión de Vapnik-Chervonenski o dimensión VC nos dice la "capacidad" que tiene una ML.

La dimensión VC es el máximo número de datos $x_1, ..., x_h$ que la máquina de aprendizaje es capaz de clasificar correctamente sin importar su organización.



Desigualdad de Vapnik-Chervonenski

Un resultado importante sobre la teoría de aprendizaje es la siguiente desigualdad:

$$R(q) \leq R_E(q) + \sqrt{\frac{h[\log(2N/h) + 1]}{N}} \tag{9}$$

donde h es la dimensión de Vapnik-Chervonenski y N el número de ejemplos.

Un algoritmo con mayor capacidad (dimensión VC grande) requiere de un mayor número de datos. De otra forma, puede presentarse overfitting.

Los elementos de un modelo de aprendizaje

Los elementos que conformarán nuestro modelo de aprendizaje son los siguientes:

1. **Conjunto de datos**: Dos tipos de conjuntos:
   1. Conjunto supervisado: $\mathcal{S} = \{(x, y) : x \in \mathbb{R}^d, y \in Y\}$.
   2. Conjunto no-supervisado: $\mathcal{U} = \{x : x \in \mathbb{R}^d\}$.
2. **Algoritmo de aprendizaje**: Determinado por:
   1. Una arquitectura (o función) que depende del tipo de datos; por ejemplo, las funciones de la forma $f(x) = wx + b$, con w y b parámetros de la función.
   2. Una función de riesgo $R_E(f)$ para estimar los parámetros que mejor se ajusten a los datos.
3. **Un método de evaluación**. Determinar una métrica y un procedimiento de evaluación.

Aprendizaje supervisado

Aprendizaje supervisado

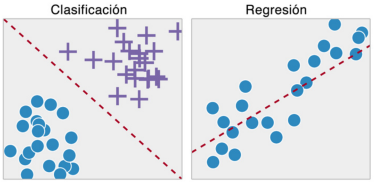
El aprendizaje supervisado se puede categorizar según los valores que toma la función en:

Regresión: Los valores son continuos, esto es, $Y \subseteq \mathbb{R}$. Por tanto:

$$f : X \to \mathbb{R}$$

Clasificación: Los valores son discretos, Y representa un conjunto de clases. En este caso:

$$f : X \to \{0, 1, 2, ...\}$$



Conjunto supervisado

El aprendizaje supervisado se puede caracterizar por medio del conjunto de datos de entrenamiento; es decir por medio del conjunto que utiliza para aprender el modelo de aprendizaje:

Conjunto supervisado

Un conjunto supervisado se define como el conjunto de datos:

$$\mathcal{S} = \{(x, y) : x \in \mathbb{R}^d, y \in Y\}$$

Donde x son los vectores que representan los datos y Y es la supervisión, que pueden ser clases o los valores de regresión.

De esta forma, la distinción entre regresión y clasificación queda caracterizada como:

- **Regresión:** $Y = \mathbb{R}$ (conjunto continuo y no necesariamente delimitado).
- **Clasificación:** $Y \subseteq \mathbb{Z}$ (conjunto finito y discreto).

Algunos tipos de clasificadores supervisados

Dentro de los clasificadores supervisado podemos distinguir los siguientes tipos:

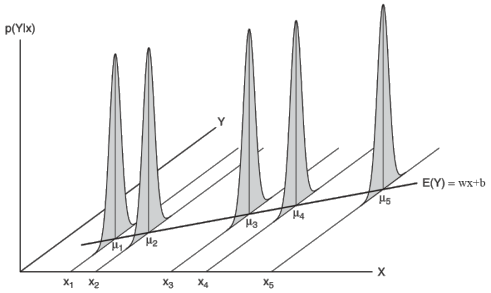
- **Regresión:**
  - Regresión lineal
  - Regresión logística
- **Clasificación:**
  - Clasificación lineal: perceptrón, máquinas de soporte vectorial
  - Clasificacón particional: árboles de decisión, bosques aleatorios, *k*-nearest neighbors

Modelo lineal y estadística

Los modelos lineales asumen que la media de una distribución es un factor lineal; dada una variable $Y \sim N(\mu, \sigma)$, la media, se estima como:

$$\mu = \mathbb{E}(Y) = \sum_i w_i x_i + b$$

Gráficamente, dado un conjunto de puntos, se estima el valor esperado de esos puntos:



Modelo general lineal

Los **modelos lineales** buscan aproximar una variable dependiente Y a partir de funciones lineales. En general, se basan en la función:

$$f(x) = wx + b = b + \sum_{i=1}^{d} w_i x_i \tag{10}$$

En general, cuando el modelo es de regresión, lo que se estima es la media de los valores en Y; es decir:

$$\hat{y} = wx + b$$

Error

Cuando se estima un modelo de **regrsión lineal** como $wx + b$, es decir, el valor esperado, se espera tener un error que ajuste el valor en cada punto. Sea este **error** ϵ_y. Por tanto, el valor exacto sería:

$$y = wx + b + \epsilon_y$$

Si despejamos el error podemos obtener que:

$$\epsilon_y = y - wx + b$$

Función de riesgo en regresión lineal

El objetivo del aprendizaje ne la regresión es minimizar el riesgo; para esto, se pueden definir al menos dos formas de minimizar el error:

- **Least-absolute value (LAV)**:

$$R(w, b) = \sum_{(x,y)\in\mathcal{S}} |y - wx + b|$$

- **Least-squares**:

$$R(w, b) = \frac{1}{2} \sum_{(x,y)\in\mathcal{S}} (y - wx + b)^2$$

De manera general se usa el método de least-squares, pues es más fácil de derivar.

Otra derivación del error cuadrático

Si asumimos que la función de riesgo está definida como:

$$R(w, b) = -\sum_{(x,y)} \ln p(y|x)$$

Y tomamos asumimos que $Y \sim N(\mu, \sigma = 1)$, donde $\mu = wx + b$, tenemos:

$$\begin{aligned}
\arg\min_{w,b} R(w, b) &= \arg\min_{w,b} -\sum_{(x,y)} \ln\Big(\frac{1}{\sqrt{2\pi}} e^{\frac{1}{2}\left(y-(wx+b)\right)}\Big) \\
&= \arg\min_{w,b} \sum_{(x,y)} \frac{1}{2}\big(y - (wx + b)\big) - \ln\frac{1}{\sqrt{2\pi}} \\
&= \arg\min_{w,b} \sum_{(x,y)} \frac{1}{2}\big(y - (wx + b)\big)
\end{aligned}$$

Obtención de los parámetros

Para obtener los parámetros en la regresión podemos derivar e igualar a 0:

$$\nabla_w \frac{1}{2}||Y - Xw||^2 = X^T Y - X^T X w$$

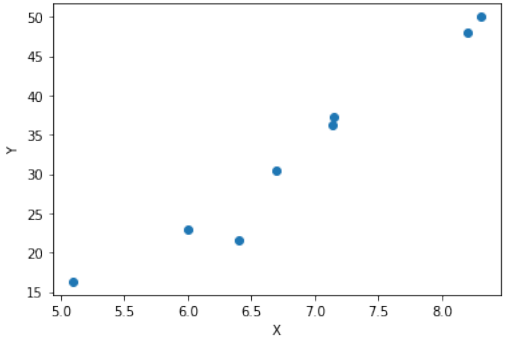
De donde, igualando a 0, tenemos que:

$$\arg\min_w \frac{1}{2}||Y - Xw||^2 = (X^T X)^{-1} X^T Y$$

Ejemplo de regresión

Podemos considerar los siguientes datos:

	X	Y
Casa 1	6	22.9
Casa 2	7.14	36.2
Casa 3	6.4	21.6
Casa 4	6.7	30
Casa 5	5.1	16.3
Casa 6	7.15	37.3
Casa 7	8.3	50
Casa 8	8.2	48



Ejemplo de regresión

Podemos usar el 70% del dataset para **entrenar**. Tomemos los siguientes 5 ejemplos:

	X	**Y**
Casa 1	6	22.9
Casa 4	6.7	30
Casa 5	5.1	16.3
Casa 6	7.15	37.3
Casa 8	8.2	48

Además a X le agregamos un 1 que representa el bias. Tenemos que: $X^T Y = \begin{pmatrix} 1081.825 & 154.5 \end{pmatrix}$
Mientras que:

$$(X^T X)^{-1} = \begin{pmatrix} 0.18 & -1.21 \\ -1.21 & 8.22 \end{pmatrix}$$

De tal forma que el resultado es:

$$w = 10.49, b = -38.67$$

Ejemplo de regresión

Podemos **evaluar** con el dataset restante, tomando la fórmula:

$$f(x) = 10.49x - 38.67$$

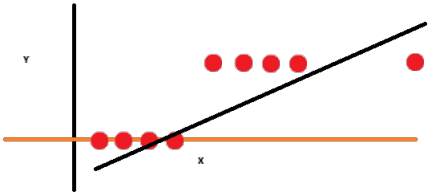
De tal forma que obtenemos:

	$\mathbf{X}$	$\mathbf{f(x)}$	$\mathbf{Y}$	**Error**
Casa 2	7.14	36.22	36.2	0.02
Casa 3	6.4	28.46	21.6	6.5
Casa 6	7.15	36.33	37.3	0.96
Casa 7	8.3	48.39	8.2	1.6

El error promedio es de 2.27, por lo que el método obtenido aproxima bastante bien los valores esperados de y.

Clasificación con modelos lineales

Supóngase ahora que se cuenta con un problema de clasificación binaria; si intentamos aproximar los datos con una recta, notamos que existe un **umbral** b, tal que si $wx \leq b$, $w \in \mathbb{R}^d$, pertenecen a la clase 0, y si $wx > b$ pertenecen a la clase 1.



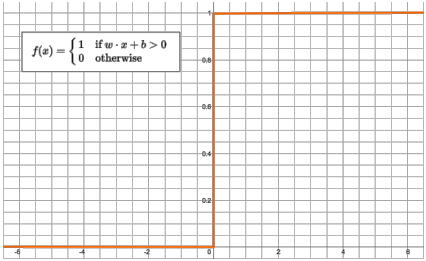
Perceptrón

Perceptrón

El perceptrón es un clasificador lineal binario que está determinado por la función $f: \mathbb{R}^d \to \{0, 1\}$, dada por:

$$f(x) = \begin{cases} 1 & \text{si } wx + b > 0 \\ 0 & \text{si } wx + b \leq 0 \end{cases}$$

Donde $w \in \mathbb{R}^d$ es un vector de pesos y $b \in \mathbb{R}$ es el bias.



Aprendizaje en el perceptrón

El perceptrón, en tanto algoritmo supervisado depende, para **entrenar**, de un conjunto supervisado:

$$\mathcal{S} = \{(x, y) : x \in \mathbb{R}^d, y \in \{0, 1\}\}$$

Su entrenamiento depende de los errores que cometa:

- Si $f(x) = 1$, pero $y = 0$, quiere decir que $wx + b$ es positivo, cuando debe ser negativo. Por tanto los pesos deben **decaer**.
- Si $f(x) = 0$, pero $y = 1$, quiere decir que $wx + b$ es negativo, cuando debe ser positivo. Por tanto los pesos deben **crecer**.

Aprendizaje en el perceptrón

En general, en el perceptrón se tiene:

$\mathbf{f(x)}$	$\mathbf{y}$	$\mathbf{f(x) - y}$
1	0	1
0	1	-1

El factor de $-(f(x) - y)$ puede ser un factor de cambio. En general, tenemos la regla para actualizar los pesos definida como:

$$w_i \leftarrow w_i - \eta\big(f(x) - y\big)x_i$$

El factor η controla el tamaño del cambio, y multiplicarlo por x_i implica sólo actualizar los pesos relevantes (si $x_i = 0$, no se modifican los pesos).

Algoritmo del perceptrón

Algorithm Algoritmo de aprendizaje en el Perceptrón

procedure FitPerceptron($\mathcal{S}, T, \eta$)
    Inputs: $\mathcal{S}$ conjunto supervisado; $T > 0$ número de iteraciones; η rango de aprendizaje.
    Inicializa $w \in \mathbb{R}^d, b \in \mathbb{R}$ aleatoriamente
    for $x, y \in \mathcal{S}$ **do**
        $f(x) = sgn(wx + b)$ Forward
        $w_i \leftarrow w_i - \big(f(x) - y\big)x_i$ Backward
    end for
    Detener si $\sum_x \big(f(x) - y\big)^2 = 0$ o después de T iteraciones.
end procedure

Perceptrón para problemas lógicos

El perceptrón puede solucionar **problemas lógicos**; en particular aquellos que tienen que ver con los operadores AND, OR y NOT.

p_1	p_2	AND	OR	NOT(p_1)
F	F	F	F	V
F	V	F	V	V
V	F	F	V	F
V	V	V	V	F

El perceptrón trabaja las condiciones de verdad y falsedad como entradas de un vector $V = 1$ y $F = 0$; de tal forma, que la entrada es un vector 2-dimensional:

x_1	x_2	AND	OR	NOT(x_1)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0
Pesos	$w = (1, 1)$	$w = (1, 1)$	$w = (-1, 0)$	
Bias	$b = -0.5$	$b = -1.5$	$b = 0.5$	

Problemas con la decisión del perceptrón

El modelo lineal, en el que se basa el perceptrón, sin embargo, muestra limitaciones:

- La convergencia a una solución no es única. Dado un problema pueden existir infinitas soluciones.
- Minsky (1969) muestra que el perceptrón es un clasificador que sólo soluciona problemas limitados.

Principalmente, la última observación terminó por limitar las aplicaciones de los perceptrones y el interés en utilizarlos.

Ya en los 90s nuevas perspectivas surgieron, una de ellas son las **SVMs**.

Máquinas de Soporte Vectorial

Las máquinas de soporte vectorial (SVM) son similares al perceptrón, pero estiman un hiperplano óptimo tomando en cuenta vectores de soporte para cada una de las clases. Estos se definen:

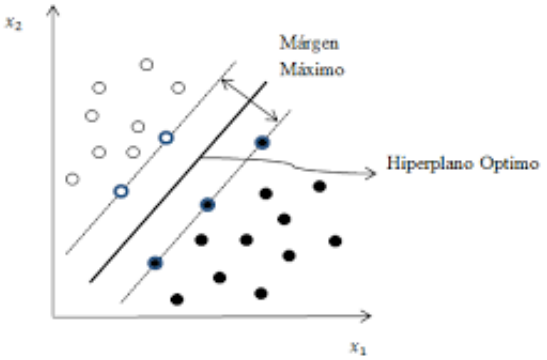
1. Para clase 1: $\{x : wx + b = 1\}$
2. Para clase 0: $\{x : wx + b = -1\}$

De esta forma, se encuentra el hiperplano que separa los datos justamente en medio de los dos conjuntos que representan las clases.

A esta forma de separar los datos se le conoce como **margen duro**.

Máquinas de soporte vectorial

En las SVMs, se obtiene el hiperplano que está a distancia $\frac{1}{||w||}$ de cada uno de los vectores de soporte.



Clasificación en las SVMs

En las SVMs, la clasificación binaria suele hacerse en las clases -1 (equivalente a 0) y 1. De esta forma, el conjutno supervisado es:

$$\mathcal{S} = \{(x, y) : x \in \mathbb{R}, y \in \{-1, 1\}\}$$

De tal forma que el margen duro puede reescribirse como:

$$y(wx + b) \geq 1$$

Y la función de desición se define como:

$$\hat{y} = sgn(wx + b) = \begin{cases} 1 & \text{si } wx + b > 0 \\ -1 & \text{si } wx + b < 0 \end{cases}$$

Máquinas de soporte vectorial lineales

Máquina de soporte vectorial

Una máquina de soporte vectorial (SVM) es un clasificador (binario) definido por la función de decisión:

$$f(x) = sgn(wx + b)$$

donde $w \in \mathbb{R}^d$, $b \in \mathbb{R}$ son parámetros. Esta función está restringida a cumplir:

$$y(wx + b) \geq 1$$

Las máquinas de soporte vectorial así definida supone que los datos son linealmente separables, de tal forma que se puede encontrar el margen adecuado que separe los datos.

Entrenamiento en las SVMs

La optimización de la SVM, dado el conjunto de datos $\mathcal{S} = \{(x, y) : x \in \mathbb{R}^d, y \in \{-1, 1\}\}$, busca encontrar los parámetros w, b que clasifiquen los datos sujetos a las restricciones.

La optimización que definiremos estará sujeta a una restricción; entonces tenemos que buscamos:

$$\begin{aligned} &\min \frac{1}{2}||w||^2 \\ &\quad \mathbf{s.t.}\ y(wx + b) \geq 1 \end{aligned}$$

Utilizando multiplicadores de Lagrange podemos definir la siguiente función de riesgo ($\lambda_i \geq 0$):

$$\begin{aligned} R(w; \lambda) &= \frac{1}{2}||w||^2 - \sum_i \lambda_i \big(y_i(wx_i + b) - 1\big) \\ &= \frac{1}{2}||w||^2 - \sum_i \lambda_i y_i(wx_i + b) + \sum_i \lambda_i \end{aligned}$$

Entrenamiento en las SVMs

Derivando la función de riesgo sobre los parámetros, tenemos:

$$\nabla_w R(w;\lambda) = w - \sum_i \lambda_i y_i x_i$$

$$\nabla_b R(w;\lambda) = \sum_i \lambda_i y_i$$

Igualando a 0 ambas derivadas obtenemos los siguientes valores:

$$w = \sum_i \lambda_i y_i x_i$$

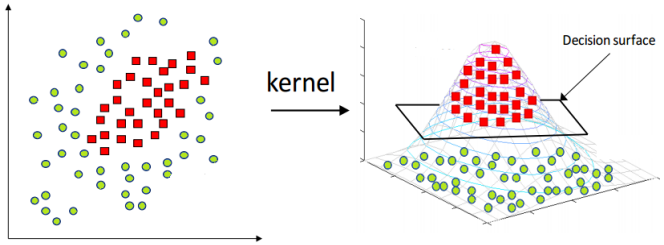
$$0 = \sum_i \lambda_i y_i$$

Problemas no linealmente separables

Si bien las SVMs lidian con el problema de encontrar la solución óptima en la clasificación, están planteadas para solucionar únicamente problemas de clasificación linealmente separables.

Para lidiar con problemas **no linealmente** separables, se ha propuesto el uso de **truco del kernel**.

La idea del truco del kernel es transformar los datos a un espacio de mayor dimensión, donde sean separables.



Kernels

Kernel

Un kernel es una función $k : X \times X \to \mathbb{R}$ que consiste en el producto punto de dos vectores transformados a un espacio de mayor dimensión, esto es:

$$k(x, x') = \phi(x) \cdot \phi(x')$$

donde $\phi : \mathbb{R}^d \to \mathbb{R}^m$, $m > d$, es una transformación de los datos a un espacio de mayor dimensión.

De esta forma, en lugar de computar la transformación ϕ, las SVMs pueden trabajar con los kernels, teniendo que:

$$f(x) = sgn\Big(\sum_i w_i y_i k(x, x'_i) + b\Big)$$

con $x'_i, y_i \in \mathcal{S}$.

Kernels

Algunos kernels que se pueden definir, sin hacer explícito ϕ son:

- Polinomial:

$$k(x, x') = (x \cdot x' + 1)^p$$

- Radial Basis Function (RBF):

$$k(x, x') = exp\{-\frac{1}{2\sigma^2}||x - x'||^2\}$$

- Tangente hiperbólica:

$$k(x, x') = \tanh(\gamma x \cdot x' + b)$$

Vecinos más cercanos

En los métodos de aprendizaje los datos están representados en un espacio vectorial, por lo que los datos de entrenamiento son vectores de rasgos que representan lo documentos.

Algunos métodos de aprendizaje se basan en la **proximidad** que tienen los datos con respecto a los grupos de una clase.

Entre más cerca esté de puntos de una clase, el algoritmo clasificará a un punto nuevo como esa clase.

Vecino

Decimos que un punto x de un espacio vectorial es un vecino de otro punto x', si existe una vecindad que los contiene a ambos; es decir, si para alguna r se cumple:

$$||x - x'||_p \leq r$$

Métricas

La vecindad depende de la **topología del espacio**; generalmente, esta topología se define por la métrica.

Métrica

Una métrica es una función $d : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$, que cumple:

1. $d(x, x') \geq 0$ y sólo 0 sii $x = x'$.
2. Para todo x, x', se cumple $d(x, x') = d(x', x)$.
3. para todo x, x', x'' se cumple $d(x, x') \leq d(x, x'') + d(x'', x')$

Una familia muy común de métricas son las llamadas métricas de Minkowski, definidas por un parámetro $1 \leq p \leq \infty$:

$$d(x, x') = ||x - x'||_p = \left(\sum_i |x_i - x'_i|^p \right)^{\frac{1}{p}}$$

k-Vecinos más cercanos

k-vecinos más cercanos

El algoritmo de k vecinos más cercanos (k-NN) es un algoritmo no paramétrico para clasificación supervisado que estima la clase de un dato x en base a los vecinos más cercanos a este; es decir:

$$\hat{y} = \arg\max_{y \leftarrow x'} \{||x - x^{(i)}||_p\}_{i=1}^{k}$$

donde se cumple la condición que los ejemplos x' son los k primeros ejemplos que cumplen $||x - x^{(1)}||_p \leq ||x - x^{(2)}||_p \leq \cdots \leq ||x - x^{(N)}||_p$

Los parámetros del algoritmo de k-NN son k, el número de vecinos a considerar, y p que determina el tipo de métrica a usar.

Entrenamiento en k-NN

En el algoritmo de k-NN el entrenamiento es prácticamente nulo, pues lo único que se haces es:

- Guardar los ejemplos $x^{(i)}$ en el espacio vectorial.
- Asociar a cada ejemplo su clase y.

La evaluación suele ser más costosa, pues en ella se calcula la métrica del punto de entrada con todos los ejemplos de entrenamiento, para quedarse únicamente con los k más cercanos y de allí ver cuál es la clase que más se repite.

Algoritmo de k-NN

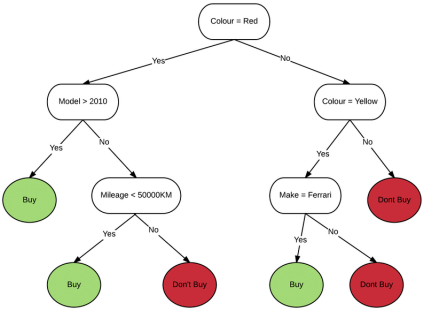
Algorithm Algoritmo de k-NN

procedure FIT-KNN($\mathcal{S}$)
    $(x^{(i)}, y_x) \leftarrow \mathcal{S}$
end procedure
procedure PREDICT-KNN(x, k, p)
    for $x^{(i)}, y_x \in \mathcal{S}$ **do**
        $dist \leftarrow (||x - x^{(i)}||_p, y_x)$
        $kNN \leftarrow \text{SORT}(dist)[:k]$
        $\hat{y} \leftarrow \arg\max_y \text{COUNT}(kNN)$
    end for
    return $\hat{y}$
end procedure

Árboles de decisión

Árbol de decisión

Un árbol de decisión es un árbol cuyos nodos representan valores de los rasgos de los datos; los hijos de los nodos responden a decisiones sobre estos valores y las hojas del árbol son clases a las que pueden pertenecer los datos.



Construcción del árbol

El problema se vuelve **construir el árbol óptimo**: es decir, el árbol más pequeño posible que clasifique adecuadamente a todos los datos (del entrenamiento).

Este es un problema NP-completo, por lo que se determina un algoritmo que busque un árbol adecuado dado los datos.

- Se observan los valores que tienen los rasgos en las clases: si son **categóricos** se ve que datos cumplen el valor y cuáles no. Si es **numérico** se determina los datos que superan el valor y aquellos que no lo superan ($x \geq \phi$).
- En base a los valores de los rasgos, se **biparticionan** los datos: El nodo padre responde a esta bipartición, los hijos responden a los datos que si cumplen el rasgo y aquellos que no.
- Se toma como nodo el rasgo que maximice una función de **ganancia de información**.
- Este proceso se repite hasta que los hijos de un nodo pertenezcan todos a la misma clase. Se crea una **hoja con esa clase**.

Ganancia de información

La ganancia de información se determina como una diferencia entre la información de los nodos hijos del árbol y la información particionando dado un rasgo ϕ:

$$GI(X, \phi) I(X) - \mathbb{E}_{p \sim \phi} I(X|\phi)$$

Donde X son los datos en el nodo del árbol y ϕ es el rasgo que se está considerando. Se pueden usar diferentes medidas de información ($X|\phi$ es la bipartición de X considerando ϕ):

- Entropía: $H(X_k) = \sum_{y \in X|\phi} p(y) \log p(y)$
- Gini Impurity: $G(X_k) = \sum_{y \in X|\phi} p(y)\big(1 - p(y)\big)$

Algoritmo de Árboles de decisión

Algorithm Algoritmo de Árboles de decisión

procedure BuildDecisionTree($\mathcal{S}$)
    $I \leftarrow I(X)$
    $\Phi \leftarrow$ Features(X)
    $Best \leftarrow 0$
    for $\phi \in \Phi$ **do**
        $IG \leftarrow I - \mathbb{E}_{p \sim \phi} I(X|\phi)$
        if $IG > Best$ **then**
            $Best \leftarrow IG$
            $parent \leftarrow \phi$
            $positive\_child \leftarrow$ BuildDecisionTree($X|\phi$)
            $negative\_child \leftarrow$ BuildDecisionTree($X|\neg\phi$)
        end if
    end for
end procedure

Aprendizaje no-supervisado

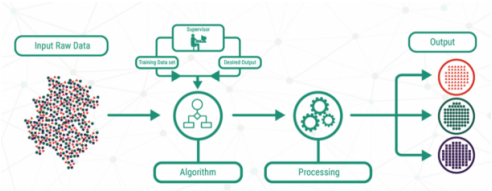
Aprendizaje no supervisado

El aprendizaje no supervisado se basa en determinar características de los datos sin que se cuente con una supervisión que indique cuál es el resultado de la tarea.

En este sentido, el aprendizaje no-supervisado encuentra patrones en los datos que puede explotar de diferentes formas, por ejemplo:

Agrupamiento: Determina grupos de datos en base a similitudes que la máquina encuentra en estos.

Reducción de dimensionalidad: Representa los datos en espacios vectoriales de baja dimensión, buscando preservar la mayor información posible.



Conjunto supervisado

El aprendizaje no supervisado se puede caracterizar por medio del conjunto de datos de entrenamiento; es decir por medio del conjunto que utiliza para aprender el modelo de aprendizaje:

Conjunto no supervisado

Un conjunto no supervisado se define como el conjunto de datos:

$$\mathcal{U} = \{x \in \mathbb{R}^d\}$$

Donde x son los vectores que representan los datos. No existe una supervisión que guíe la tarea que resolverá la máquina.

El objetivo del aprendizaje no supervisado es obtener información relevante de los datos, sin ningún tipo de ayuda.

Algunos algoritmos no supervisado

Algunos de los métodos de aprendizaje no supervisado son:

- **k-means**: Algoritmo de agrupamiento de datos en k clústers.
- **AutoEncoder:** Un AutoEncoder es una red neuronal no supervisada para distintas tareas como reducción de dimensionalidad.
- Otros métodos de agrupamiento, como métodos de agrupamiento jerárquico, espectral, etc.

El problema del agrupamiento

El **problema del agrupamiento** se puede plantear como sigue:

Problema del agrupamiento

Dado un conjunto de datos $X \subseteq \mathbb{R}^d$, y un número k de clústers y una función R que evalúa la calidad de los clústers, encontrar una función de agrupamiento $f : X \to \{1, ..., k\}$ que optimice la función R.

La función de agrupamiento f, asigna a cada $u \in X$ un clúster o grupo:

$$f(u) = j$$

donde $j \in \{1, ..., k\}$. Entonces el clúster j puede verse como:

$$f^{-1}(j) \subseteq X$$

Agrupamiento radial

En el agrupamiento se busca crear **clústers** o grupos que contengan datos que guarden similitud entre sí, mientras que los datos fuera del clúster no tengan (o tengan poca) similitud con los datos del clúster.

Para hacer esto, se puede tomar la distancia de los vectores en el espacio vectorial.

En particular podemos usar un **métrica** del espacio vectorial definida por las funciones $(1 \leq p \leq \infty)$

$$||u - v||_p = \left(\sum_i |u_i - v_i|^p \right)^{\frac{1}{p}}$$

Generalmente $p = 2$, que define la métrica euclideana.

k medias

Dado el dataset X, y un número de clústers k dado, podemos determinar una media (valor esperado) para cada clúster, c_j, como:

$$\mu_j = \frac{1}{|c_j|} \sum_{u \in c_j} u$$

En este caso se suman las $u \in X$ que pertenecen al clúster c_j, por lo que μ_j es un vector que representa la media (o valor esperado) para ese clúster.

En este sentido, se asume que X está determinada por un conjunto de **distribuciones normales** $\mathcal{N}(\mu_j, \sigma_j)$ y el objetivo es estimar los parámetros de la distribución.

Estimación de parámetros

Los valores de las medias se estiman como el promedio de los clústers. Pero también podemos estimar la **varianza** en los clústers de la siguiente forma:

$$\sigma_j^2 = \frac{1}{|c_j|} \sum_{u \in c_j} ||u - \mu_j||^2$$

Para minimizar la varianza, necesitamos derivar:

$$\begin{aligned} \nabla_{\mu_j} \frac{1}{|c_j|} \sum_{u \in c_j} ||u - \mu_j||^2 &= \frac{2}{|c_j|} \sum_{u \in c_j} -(u - \mu_j) \\ &= 2\mu_j - \frac{2}{|c_j|} \sum_{u \in c_j} u \end{aligned}$$

Igualando a 0 y despejando μ_j obtenemos $\mu_j = \frac{1}{|c_j|} \sum_{u \in c_j} u$.

Función de riesgo

La función de riesgo que debemos minimizar, entonces, puede estar definida en términos de la varianza:

$$R(\mu_1, .., \mu_k) = \frac{1}{|c_j|} \sum_{j=1}^{k} \sum_{u \in c_j} ||u - \mu_j||^2$$

En general, minimizar esta función, equivale a encontrar las medias más apropiadas para cada clúster, que minimicen la varianza dentro de estos. Lo que responde a la hipótesis del clúster.

Para minimizar esta función, definiremos un algoritmo iterativo.

Algoritmo de k-means

La idea del algoritmo de *k*-means se resume de la siguiente forma:

1. Generar medias μ_j (o centroides) de manera aleatoria.
2. Asignar cada punto $u \in X$ en la colección, al clúster cuya media este más cercana al documento.
3. Recalcular las medias en base a los puntos que están en el clúster.
4. Repetir los pasos anteriores hasta que ya no haya más reasignaciones.

Algoritmo de k-means

Algorithm Algoritmo de k-means

procedure kMeans(X; k)
  $\mu_j \leftarrow$ Random(X, k), $j = 1, ..., k$
  for j **from** 1 **to** k **do**
    $\hat{c}_j \leftarrow \{\}$
    for $u \in X$ **do**
      $l \leftarrow \arg\min_j ||u - \mu_j||^2$
      $\hat{c}_l \leftarrow \hat{c}_l \cup \{u\}$
      for j **from** 1 **to** k **do**
        $\mu_j \leftarrow \frac{1}{|\hat{c}_j|} \sum_{u \in \hat{c}_j} d_i$
      end for
    end for
  end for
  return $\hat{c}_1, ..., \hat{c}_k$
end procedure

Textos recomendados

Russel, S. y Norvig, P. (2021). "V. Machine Learning". *Artifitial Intelligence: A Modern Approach*, Pearson, pp. 430-478.

Joshi, P. (2017). "2. Classification and Regression Using Supervised Learning". *Artificial Intelligence with Python*, Packt, pp. 33-68.

Joshi, P. (2017). "4. Detecting Patterns with Unsupervised Learning". *Artificial Intelligence with Python*, Packt, pp. 33-68.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill.

Alpaydin, E. (2020). *Introduction to Machine Learning*. MIT Press.