Inteligencia Artificial para profesionales TIC

Técnicas ML

valgrai.eu

Valencian Graduate School and Research Network of Artificial Intelligence

Índice

1. Técnicas ML

Centrémonos en los más clásicos

Hay una gran cantidad de técnicas de ML posibles, basadas en principios bastante diferentes, con pros y contras, con **hiperparámetros** y muchas más consideraciones.

Solo vamos a cubrir los más importantes, siendo ahora bastante fácil explorar y encontrar el más adecuado para un problema específico si conocemos el contexto. Lo veremos en muchos ejemplos futuros y prácticos.

Por lo tanto, en esta unidad, vamos a cubrir las técnicas de ML centradas en la **información estructurada**, es decir, datos tabulares o conjuntos de datos basados en características.

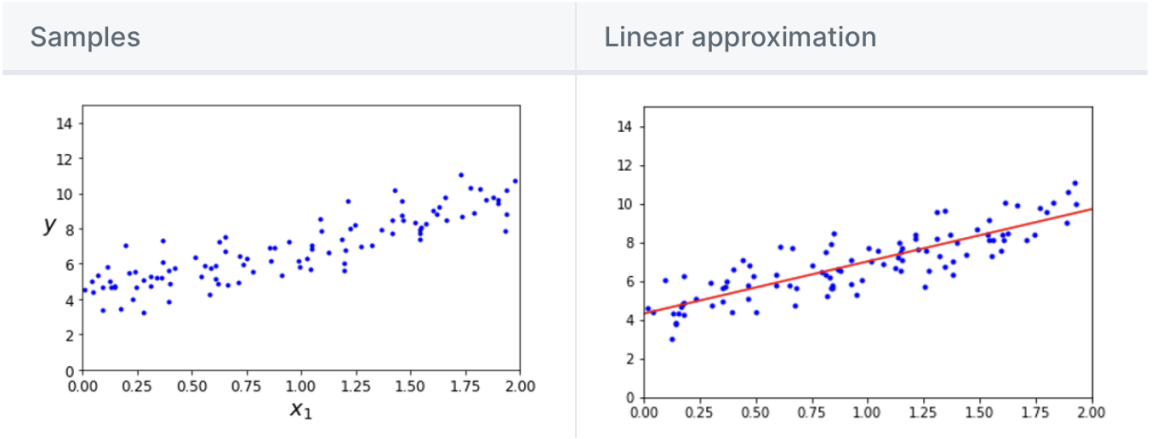
Cubriremos el aprendizaje profundo (un subconjunto del aprendizaje automático) en unidades específicas posteriors para comprender sus diferencias y ventajas.

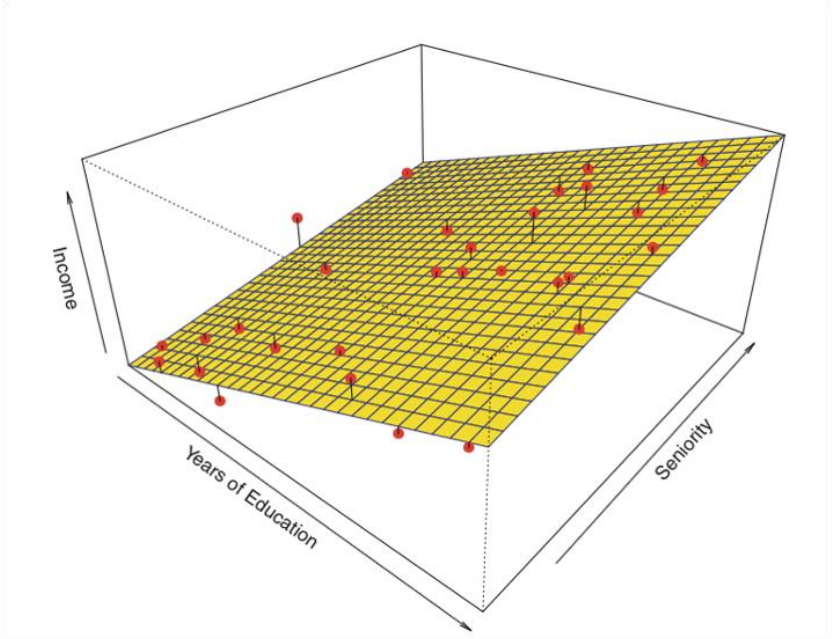
2. Regresión lineal

La estrategia de regresión clásica

La **regresión lineal** es una técnica simple de ML muy conocida cuando sabemos que el resultado de nuestro modelo se basa en una correlación lineal con las características de entrada independientes. Por lo tanto, malos resultados con no linealidad.

Con 1 característica, tenemos una línea; con un vector de n características tenemos un hiperplano de n dimensiones.





2. Linear regression

Descenso de gradiente como posible solución

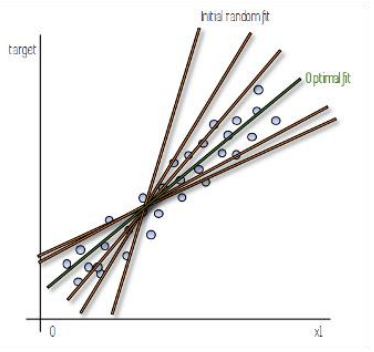
Dada una función definida por un conjunto de parámetros, el descenso de gradiente comienza con un conjunto inicial de valores de parámetros y se mueve iterativamente hacia un conjunto de valores de parámetros que minimizan la función.

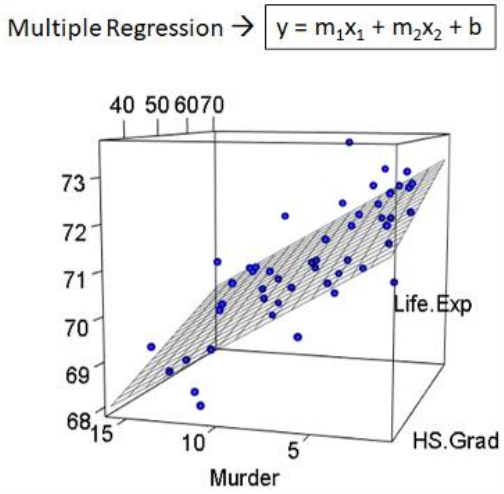
Esta minimización iterativa se logra utilizando cálculo, dando pasos en la dirección negativa del gradiente de la función (el ∇ del error de costo).

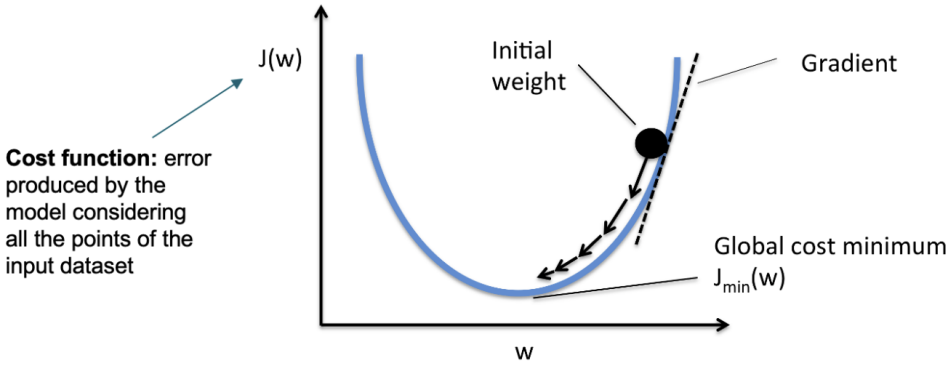
El **descenso de gradiente** reduce la función de coste J(w), por ejemplo, MSE

El tamaño de cada paso conocido como **Tasa de aprendizaje**.

Debemos tener cuidado al elegir la tasa de aprendizaje (se requieren muchas iteraciones demasiado pequeñas, demasiado grandes nunca podríamos alcanzar un mínimo)





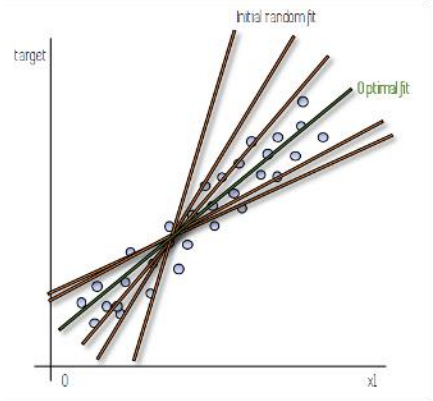


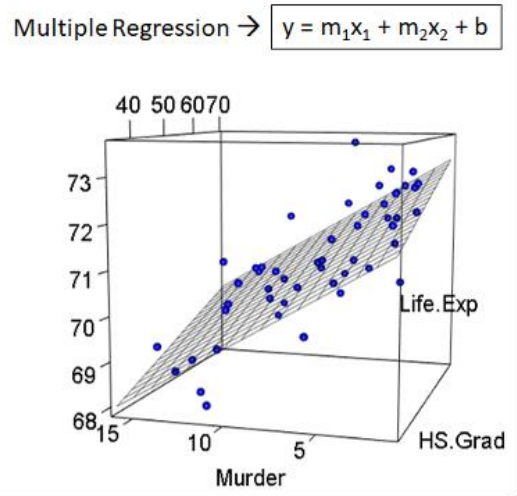
2. Regresión lineal

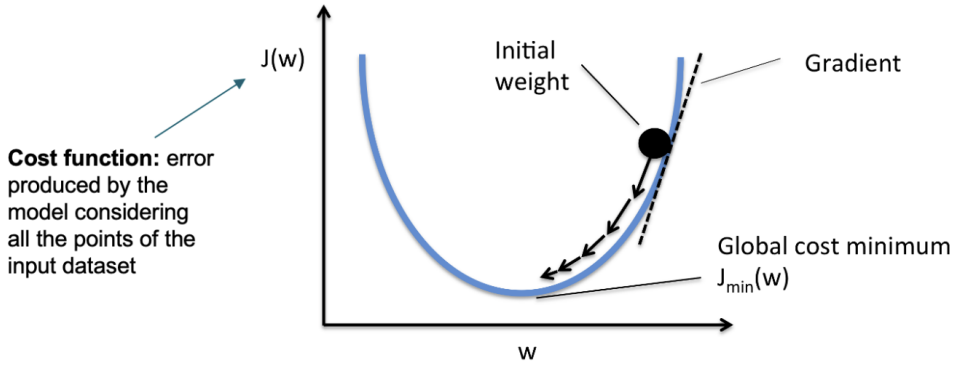
Descenso de gradiente como posible solución

Repita hasta la convergencia

m(i) = m(i) – LearningRate * Gradient(J(w), regarding m(i))







3. Regresión logística

Esto nos llevará al perceptrón y a la red neuronal artificial

La **Regresión Logística** es una técnica de ML supervisada para la **clasificación** (modelo de clasificación paramétrica además de tener 'regresión' en su definición)

Es un algoritmo ideal para tener como punto de referencia (una buena base para comparar con otras técnicas de ML 'más' avanzadas)

La Regresión Logística es un método de **clasificación** ML muy sencillo, con muy buenos resultados en varios problemas, fácil de entender e interpretar (interpretabilidad/explicabilidad)

La idea subyacente a este modelo es que en lugar de simplemente usar una ecuación lineal para dividir los datos de entrada y hacer predicciones basadas en si un nuevo dato de entrada pertenece o no a una clase específica, usaremos una función sigmoide capaz de tener como entrada de $-\infty$ a $+\infty$ y devolver de 0 a 1, que interpretaremos como una probabilidad de pertenecer a una clase específica en el proceso de predicción.

3. Regresión logística

Esto nos llevará al perceptrón y a la red neuronal artificial

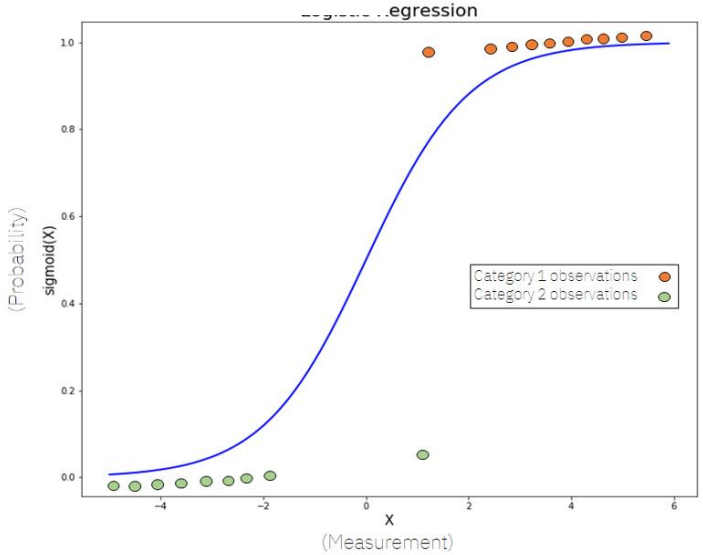
$$z = w^T x + b$$ ← Bias (value of z for x = 0)

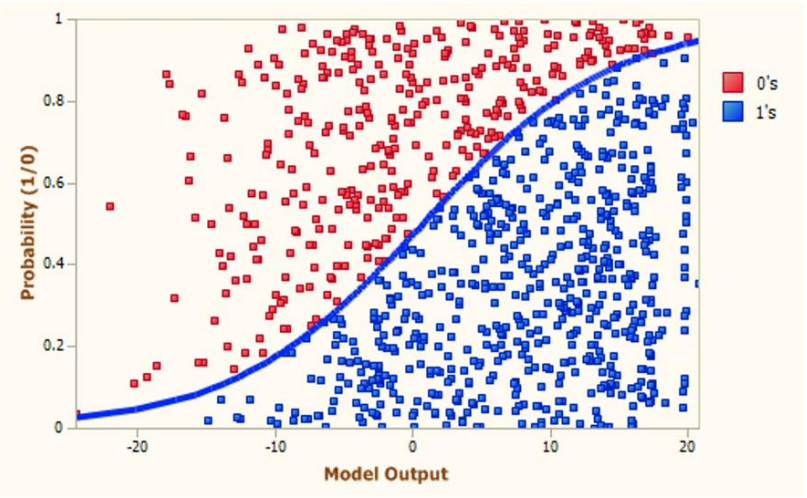
Linear weights — Vector of features

$$a = \frac{1}{1 + e^{-z}}$$

$$y_{predict} = a = sigmoid(z)$$

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$$





3. Regresión logística

Pros y cons

La principal limitación de la **Regresión Logística** es la suposición de linealidad entre la variable dependiente y las variables independientes

Los problemas no lineales no se pueden resolver con la regresión logística porque tiene una superficie de decisión lineal. Los datos separables linealmente rara vez se encuentran en escenarios del mundo real.

De todos modos, es una excelente primera aproximación para la clasificación binaria (rápida y fácil de interpretar)

4. Naïve Bayes

El teorema de Bayes puede predecir

Näive Bayes es una técnica de ML supervisada para la clasificación (binaria o multiclase), basada en el Teorema de Bayes.

Este clasificador utiliza las **probabilidades a priori** para **estimar eventos futuros (a-posteriori**).

Utiliza datos de entrenamiento para calcular la probabilidad observada de cada evento en función del vector de características.

Después del proceso de entrenamiento, utilizando como mecanismo el Teorema de Bayes y algunos supuestos, predecirá la clase de entradas futuras.

De alguna manera, las probabilidades que calculemos a partir de los datos de entrenamiento se utilizarán para futuras predicciones.

Es un clasificador muy básico, sin embargo, a veces da resultados bastante satisfactorios.

Es común usarlo en la clasificación de textos, detección de intrusos en la red, diagnóstico médico, etc.

4. Naïve Bayes

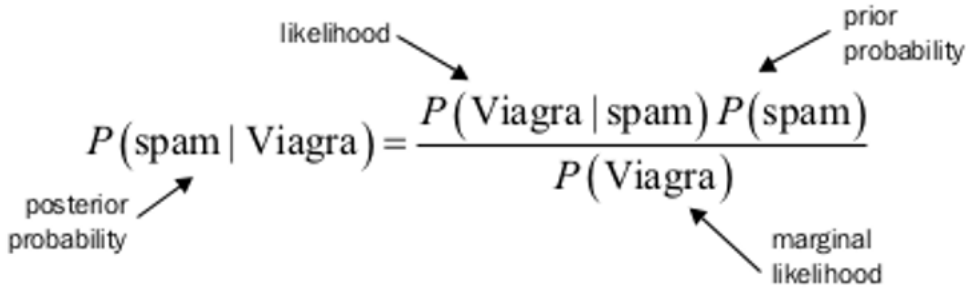
El teorema de Bayes puede predecir

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

Imaginemos que queremos estimar la probabilidad de que un correo electrónico sea spam. Sin evidencia adicional, pensemos que es 0.2 (probabilidad a priori)

¿Qué sucede si el correo electrónico contiene la palabra 'viagra'?

Aplicando el **Teorema de Bayes**, es posible calcular la probabilidad a-posterior. Si es mayor que 0.5, es más probable que este correo sea spam que ham, y debemos filtrarlo.



4. Naïve Bayes

El teorema de Bayes puede predecir

Näive Bayes va a asumir que todas las **variables son independientes e igualmente importantes** (relajamos esto para obtener también un modelo más simple y más eficiente computacionalmente, algo que también hacen otros clasificadores)

Esto no suele ser cierto. Por ejemplo, si estamos considerando quién envía el correo electrónico y el contenido del correo electrónico, la persona que envía el correo electrónico podría ser más importante que el contenido del mismo. No son igual de importantes (aunque lo vamos a suponer)

Por otro lado, en un correo electrónico donde tenemos la palabra 'viagra' (siendo un valor de las características que estamos considerando) es más probable que también aparezca la palabra 'droga' (otro valor de una característica del clasificador). **Tienen cierto grado de dependencia que vamos a ignorar en este clasificador**.

A pesar de estas suposiciones, el clasificador **funciona relativamente bien en muchos problemas**.

A partir del conjunto de datos de entrenamiento podemos construir una tabla de frecuencias, que indica el número de veces que una palabra (viagra) ha aparecido en los mensajes de spam.

Esta tabla de frecuencias se puede utilizar para calcular la tabla de probabilidad:

4. Naïve Bayes

El teorema de Bayes puede predecir

	Viagra		
Frequency	Yes	No	Total
spam	4	16	20
ham	1	79	80
Total	5	95	100

	Viagra		
Likelihood	Yes	No	Total
spam	4 / 20	16 / 20	20
ham	1 / 80	79 / 80	80
Total	5 / 100	95 / 100	100

$$P(\text{ Spam } \mid \text{ Viagra }) = \frac{P(\text{ Viagra } \mid \text{ Spam })P(\text{ Spam })}{P(\text{ Viagra })} = \frac{\left(\frac{4}{20}\right)\left(\frac{20}{100}\right)}{\left(\frac{5}{100}\right)} = 0.8$$

La probabilidad de que un correo electrónico que tenga la palabra 'viagra' sea realmente 'spam' es de 0,8 (de un 0,2 de probabilidad de tener spam, obtenemos 0,8 si la palabra 'viagra' está en el correo electrónico)

4. Naïve Bayes

Pros y cons

Pros

- Es sencillo, rápido y eficaz
- Funciona bien con ruido en los datos y valores faltantes. No requiere grandes conjuntos de datos de entrenamiento
- El cálculo de la predicción es realmente rápido

Cons

- Como contras, es asumir independencia de las variables y que todas son igual de importantes
- No es ideal con conjuntos de datos con muchos valores numéricos

5. Supported Vector Machines

SVM - Clasificación y regresión

Modelo supervisado (permite **clasificación y regresión**).

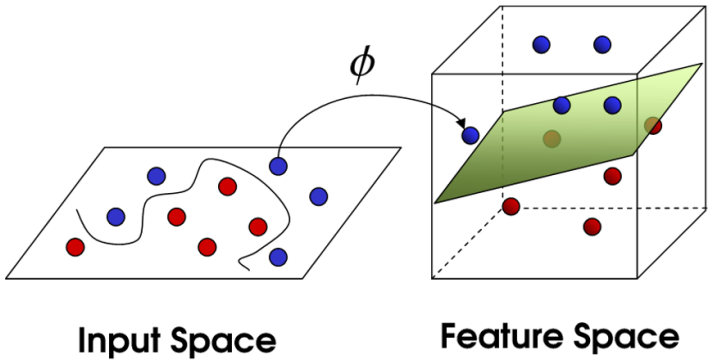
No es un modelo probabilístico.

Vladimir Vapnik introdujo esta técnica en 1963.

SVM utiliza un enfoque geométrico con una optimización con restricciones y con el objetivo de encontrar hiperplanos para dividir mejor el conjunto de entrada.

Transforma el espacio de entrada original en lo que llamamos espacio de "características" cuando el problema no se puede resolver linealmente.

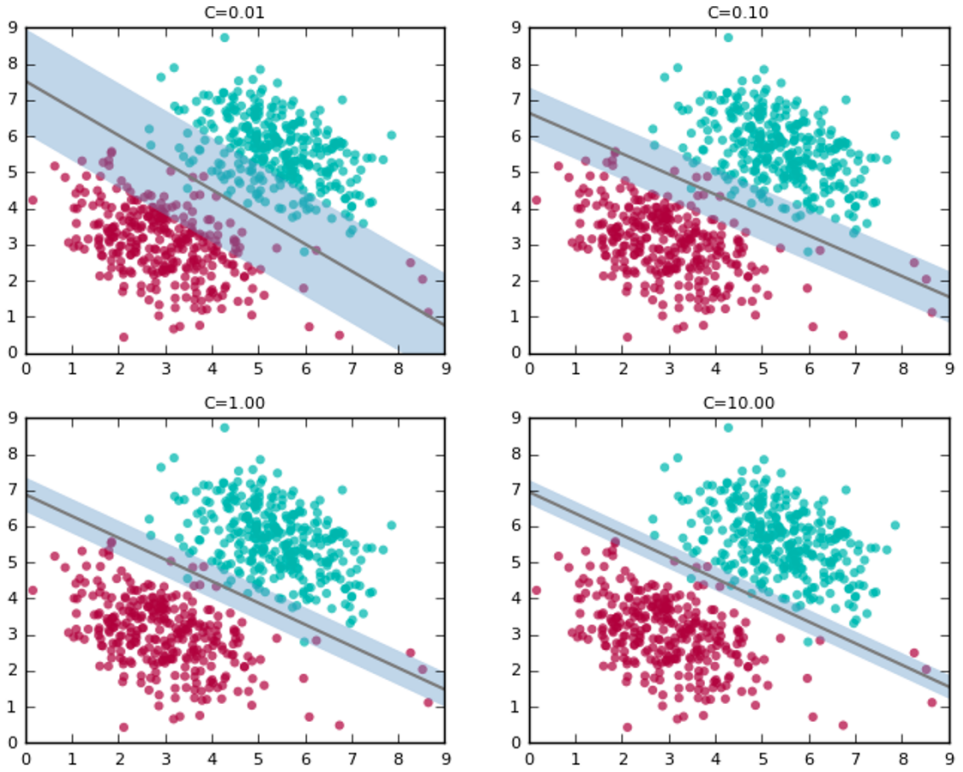
Utilizando algunas funciones posibles, se crea un nuevo espacio de características (más dimensiones) donde se pueden realizar divisiones con hiperplanos.



5. Supported Vector Machines

Soft margin classifier

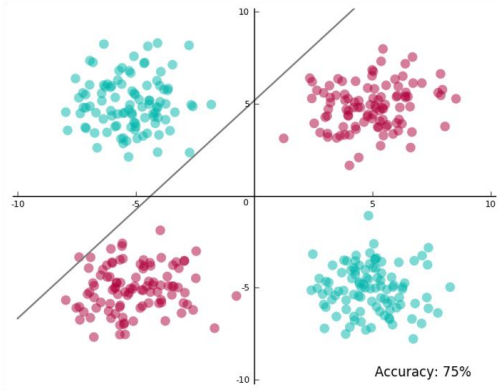
C como parámetro de costo o regularización.



Cross validations and comparing metrics will give as a good C value for our model

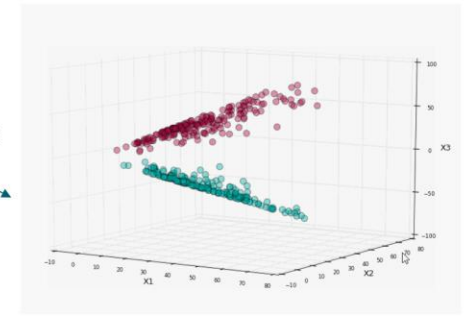
5. Supported Vector Machines

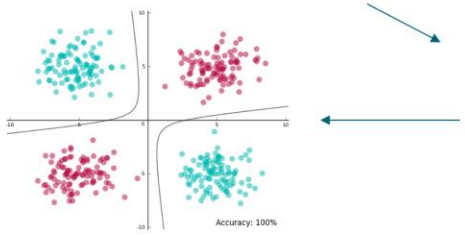
No linealidad

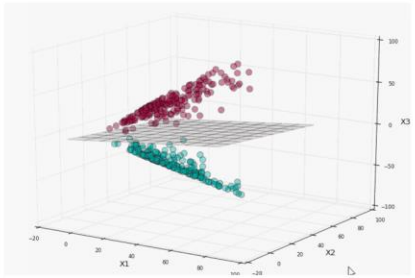


Feature expansion:

$$X_1 = x_1^2$$
$$X_2 = x_2^2$$
$$X_3 = \sqrt{2}x_1x_2$$







5. Supported Vector Machines

SVM - Clasificación y regresión

¿Qué funciones probamos para la expansión de características y cómo podemos lidiar con un proceso de costos más computacionales? **Kernel tricks** (basados en estudiar la similitud entre cada par de puntos del conjunto de datos de entrenamiento en el espacio expandido y "asignar" esto al espacio de características original)

La biblioteca Scikit-learn nos permite explorar funciones polinómicas, de base radial y sigmoide con hiperparámetros en cuanto a su complejidad.

5. Supported Vector Machines

Pros y cons

Pros

Aunque hemos presentado SVM con ejemplos de **clasificación**, se han extendido para trabajar también en problemas de **regresión**.

Es muy estable con los datos de ruido y no suele generar sobreajuste.

Cons

Como contra, requiere una búsqueda inicial de los kernels e **hiperparámetros** más apropiados, lo que hace que el proceso de entrenamiento sea especialmente intensivo con grandes conjuntos de datos y muchas características.

Los valores de las características mejor escalados (de hecho, con cualquier técnica de ML basada en distancias euclidianas).

Es un modelo de caja negra, bastante difícil de interpretar.

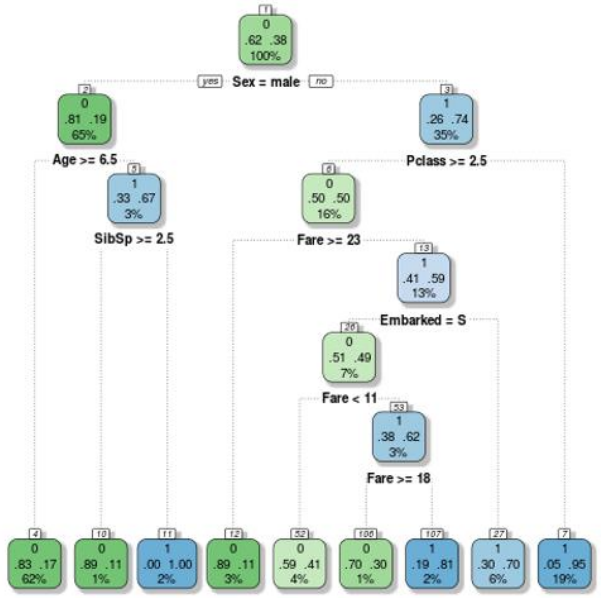
6. Árboles de decisión

Bien conocido, pero ahora creado por inducción

Los árboles de decisión se utilizan en muchos contextos como una herramienta de apoyo para la toma de decisiones, o como bocetos preliminares de programas o estrategias complejas.

La idea de usarlos en ML es poder generarlos a partir de datos de entrenamiento (aprendizaje).

- Regresión o clasificación supervisada
- Modelo de caja blanca
- Permiten características numéricas y categóricas (binarias y multiclase)
- Suelen utilizarse como elementos básicos para modelos más complejos (boosting, bagging)
- Dividir o segmentar el espacio de predicción de variables (características de los elementos)
- Para predecir una nueva observación, utilizan el promedio o el modo de regresión
- El conjunto final de reglas sobre las variables de predicción se puede resumir como un árbol de decisión



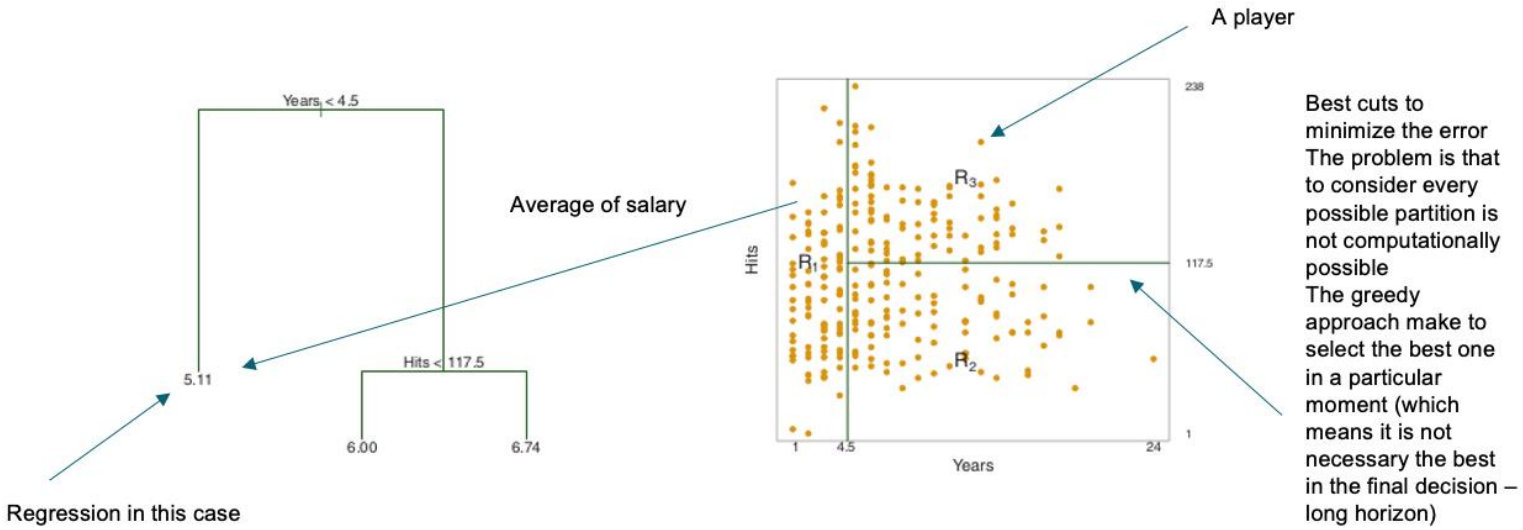
6. Árboles de decisión

Estrategia básica

Aunque existen varios algoritmos, uno de los más comunes es el algoritmo ID3 (y variantes), y este es el enfoque que describiremos aquí.

Dividen los valores del espacio de características en cuadros de alta dimensión que minimizan el error de predicción.

Usaremos un enfoque voraz (greedy) de arriba hacia abajo conocido como división binaria recursiva.



6. Árboles de decisión

Estrategia básica

Intentaremos los **mejores cortes** para minimizar el error.

El problema es que **considerar todas las particiones posibles no es posible desde el punto de vista computacional.**

El enfoque voraz hace que se **seleccione el mejor en un momento determinado o a corto plazo** (lo que significa que no es necesario el mejor en la decisión final, a largo plazo).

La división binaria recursiva se refiere al hecho de que elegiremos primero una variable/característica que divida mejor los conjuntos, y así sucesivamente (veremos cómo consideramos la 'mejor división').

En el nodo raíz, se elige la variable más predictiva, generando una división con diferentes muestras de entrada para cada clase/etiqueta

El algoritmo sigue la misma estrategia dividiéndose en nodos con este "criterio" de la mejor variable/característica hasta que alcanza una condición de parada:

- Todas (o casi todas) las muestras de entrenamiento de un nodo son de una clase única (etiqueta)
- No hay más variables capaces de distinguir las muestras de entrada
- El árbol ha alcanzado una profundidad predefinida

6. Árboles de decisión

Mejor corte

Si los segmentos de una división tienen valores de una sola clase/etiqueta/categoría, consideramos que se trata de una división pura.

Existen varias métricas de "**pureza**" para identificar este criterio de corte.

Muchos algoritmos DT utilizan **entropía**.

Un valor de entropía igual a 0 significa que el conjunto es totalmente homogéneo, sin embargo un valor de 1 significa desorden total o caos.

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2 (p_i)$$

6. Árboles de decisión

Mejor corte

Por ejemplo, si después de decidir una característica y un valor para hacer el corte con la información de casas de San Francisco y Nueva York, uno de los segmentos que creamos finalmente tiene un 60% de elementos de una categoría/etiqueta/clase y un 40% de la otra (en un ejemplo binario como San Francisco / Nueva York)

-0.60 * log2(0.60) – 0.40 * log2(0.40) = **0.9709506 (Desorden/caos casi total)**

El algoritmo tiene que decidir con qué característica tenemos que ver el corte (y también, el umbral de corte)

Es habitual utilizar la **entropía** con el fin de calcular o medir el cambio resultante para realizar el corte con una característica concreta (y umbral, siempre el que nos da la entropía mínima)

Lo que sigue el algoritmo es una medida llamada **Ganancia de Información (IG),** que intentaremos maximizar, que se define como la diferencia entre la entropía del segmento antes de la división (S1) y de la partición resultante después de la división (S2)

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

Entropy at the current node

Entropy of the result after the cut

6. Árboles de decisión

Poda

El algoritmo voraz de arriba hacia abajo puede generar buenas predicciones en el conjunto de datos de entrenamiento, pero **tiende a provocar un alto sobreajuste**.

Si esto sucede, es porque el árbol es demasiado complejo (profundidad).

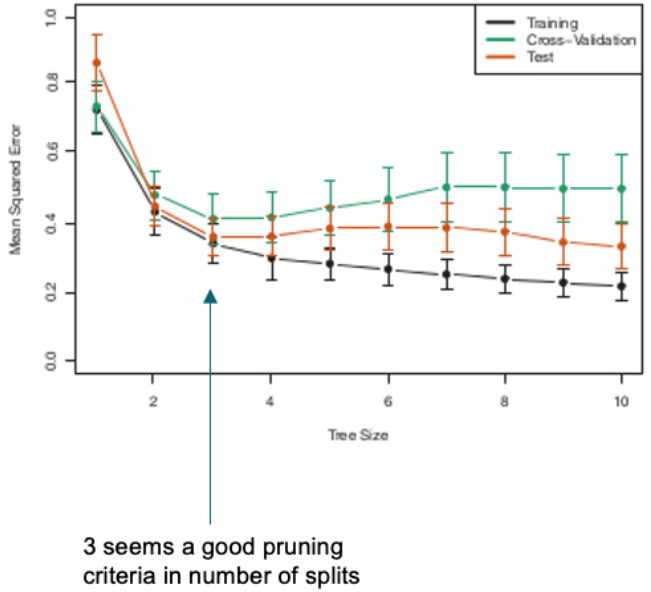
Un árbol más pequeño puede darnos una menor varianza y mejores capacidades de predicción (generalización) con el consiguiente sesgo.

Una buena estrategia es: **generar un árbol profundo y luego podarlo para obtener un subárbol (post-poda)**

¿Cuándo seleccionamos el subárbol? Usando el que nos da la menor tasa de error (las mejores métricas) usando validación cruzada o hold-out.

Las últimas ramas tienen en cuenta los detalles específicos del conjunto de datos de entrenamiento.

La regularización es generalmente la forma en que llamamos a ML cada vez que tomamos una decisión para mejorar la generalización de nuestro modelo, haciéndolo normalmente más pequeño o menos poderoso (menos centrado en los detalles del conjunto de datos de entrenamiento y más capaz de darnos buenas predicciones futuras)



6. Árboles de decisión

Pros y cons

Pros

- Modelo de propósito general con buen comportamiento general
- Utiliza solo las características más importantes (selección automática de características)
- Se puede utilizar con conjuntos de datos pequeños/grandes
- Muy fácil de interpretar (IA explicable). Capacidad para trabajar con características numéricas y categóricas a la ve.z
- La multicolinealidad no afecta a la calidad del algoritmo
- Eficiente desde el punto de vista computacional en las predicciones

Cons

- No tiene buenas métricas con algunos problemas (donde no podemos dividir el espacio en las cajas)
- Propenso al sobreajuste
- Un pequeño cambio en los datos puede dar lugar a un árbol muy distinto
- Cálculo intensivo cuando se entrena si los datos son similares → necesita varias preguntas para separar los datos.

7. Métodos de ensemble

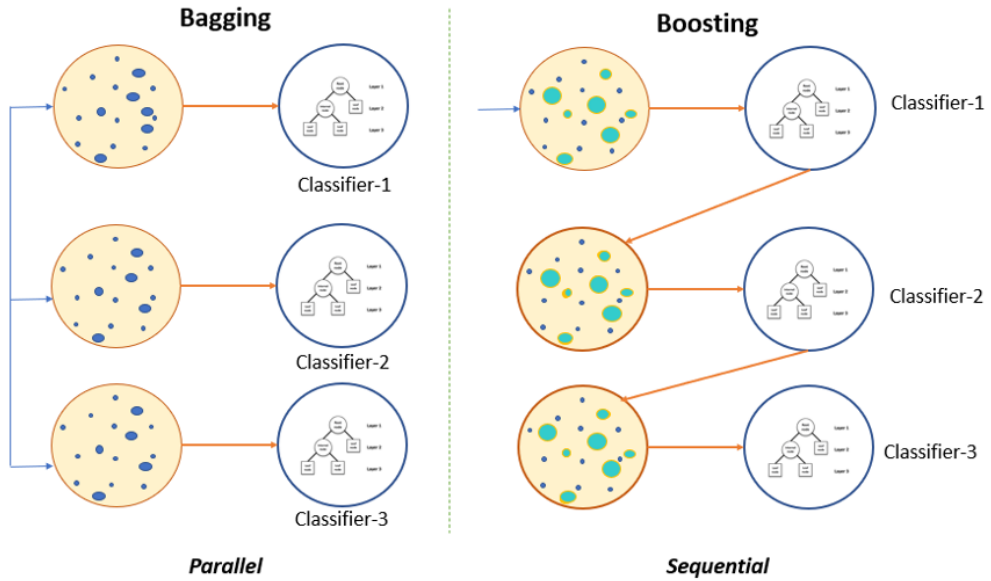
Muchos modelos en conjunto

Existen diferentes técnicas cuando queremos combinar los resultados de diferentes modelos (generados con diferentes algoritmos o no), que nos pueden dar mejores resultados finales.

Esta combinación o agregación se puede realizar de diferentes maneras.

Un enfoque directo y sencillo es una media ponderada de los resultados de cada modelo (promediar en regresión, el más votado en clasificación, dar o no diferentes pesos a los modelos, etc.) en lo que se denomina **bagging**.

Otra posibilidad, es utilizar varios modelos en secuencia (**boosting**)



7. Métodos de ensemble

Random forest

El modelo de conjunto de bagging más popular, aunque los nuevos métodos de boosting están dando mejores métricas.

Combina el bagging con árboles de decisión y con la selección aleatoria de características para agregar diversidad al conjunto final de árboles de decisión.

Por lo general, utiliza una cantidad importante de árboles, por lo que cada característica puede aparecer en varios modelos.

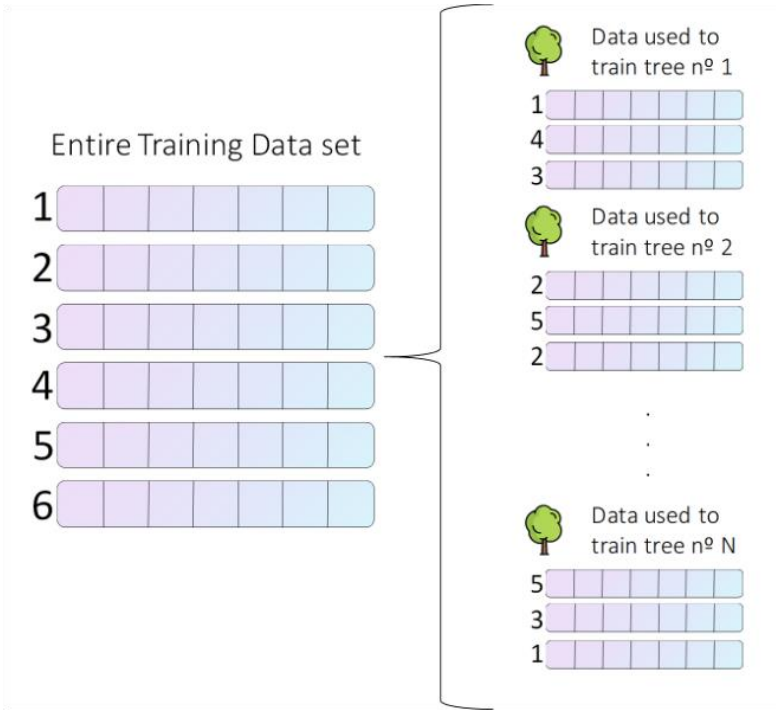
Al construir los árboles del bosque aleatorio, las divisiones se realizan con una selección aleatoria de m características del total de p características (generalmente m = sqrt(p))

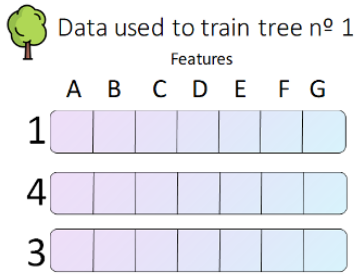
El resultado de los árboles se combina. Este resultado es el voto de la mayoría (clasificación) o del promedio (regresión)

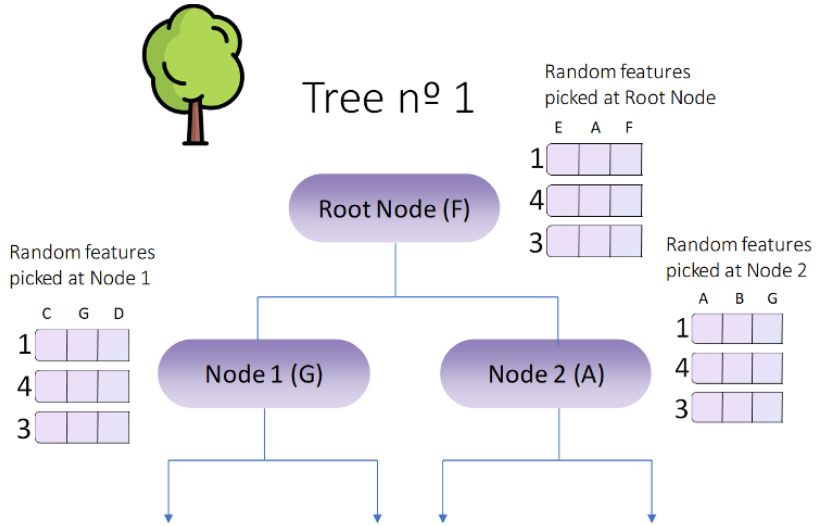
Bootstrapping, selección aleatoria de características (m < p)

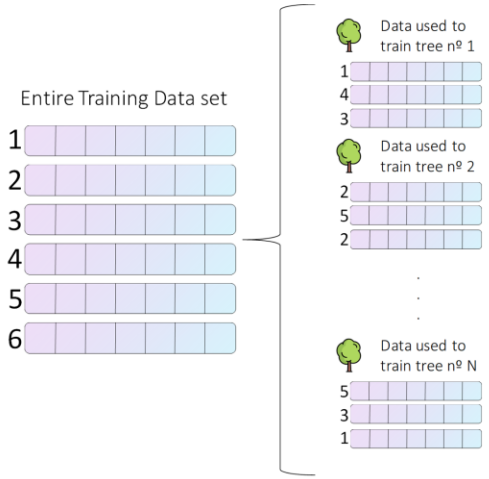
7. Métodos de ensemble

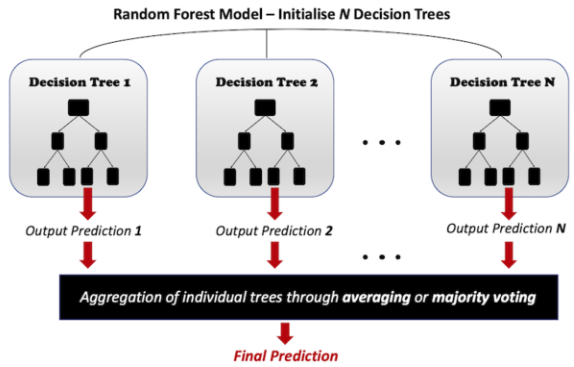
Random forest











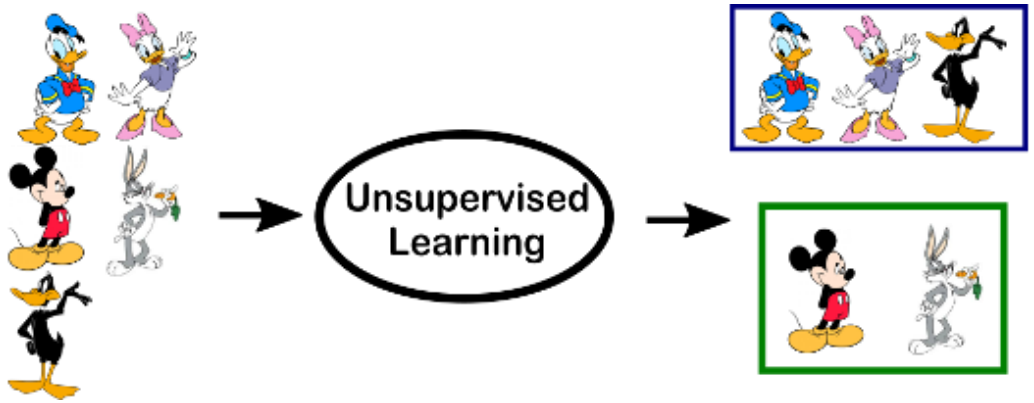
8. Aprendizaje no supervisado

Agrupamiento - clustering

El **aprendizaje no supervisado** trata de obtener patrones, grupos o cualquier estructura de un conjunto de datos no etiquetado; siendo una tarea muy difícil frente a conjuntos de datos sin contexto ni información previa.

El aprendizaje automático no supervisado es la tarea de aprendizaje automático de inferir una función para describir una estructura oculta a partir de datos "no etiquetados" (no se incluye una clasificación o categorización en las observaciones). Entre los escenarios comunes para el uso de algoritmos de aprendizaje no supervisado se incluyen:

- Exploración de datos
- Detección de valores atípicos
- Reconocimiento de patrones



8. Aprendizaje no supervisado

Clustering

La técnica no supervisada más conocida es lo que llamamos **clustering**.

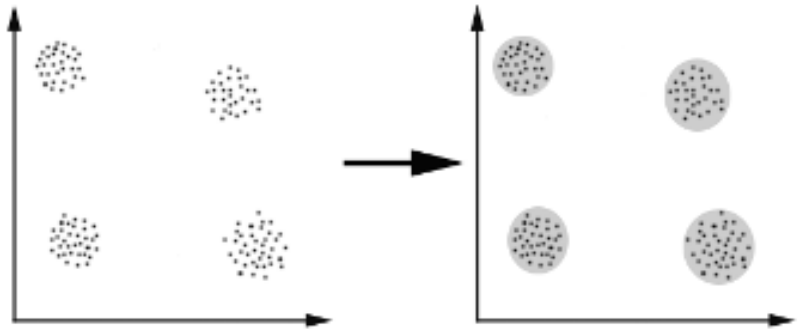
La agrupación en clústeres es una técnica de aprendizaje automático no supervisada que intenta dividir los datos en clústeres o grupos similares, sin necesidad de caracterizar las propiedades comunes de los clústeres.

Las instancias dentro de un clúster deben ser muy similares, pero diferentes entre los clústeres.

Por lo tanto, la **similitud** es un concepto básico en la agrupación.

Una definición vaga de agrupamiento podría ser "el proceso de organizar objetos en grupos cuyos miembros son similares de alguna manera"

Un clúster es, por lo tanto, una colección de objetos que son "similares" entre ellos y son "diferentes" a los objetos que pertenecen a otros clústeres.



8. Aprendizaje no supervisado

K-means

El algoritmo comienza asignando a todos los n elementos del conjunto de datos a uno de los k clústeres, donde k debe ser un número dado.

El objetivo es minimizar las diferencias entre los elementos de un clúster (utilizando la similitud) y maximizar las diferencias entre los clústeres.

A menos que los valores de k y n sean realmente pequeños, no es posible calcular todos los grupos óptimos posibles a partir de todas las combinaciones posibles del conjunto de datos de entrada.

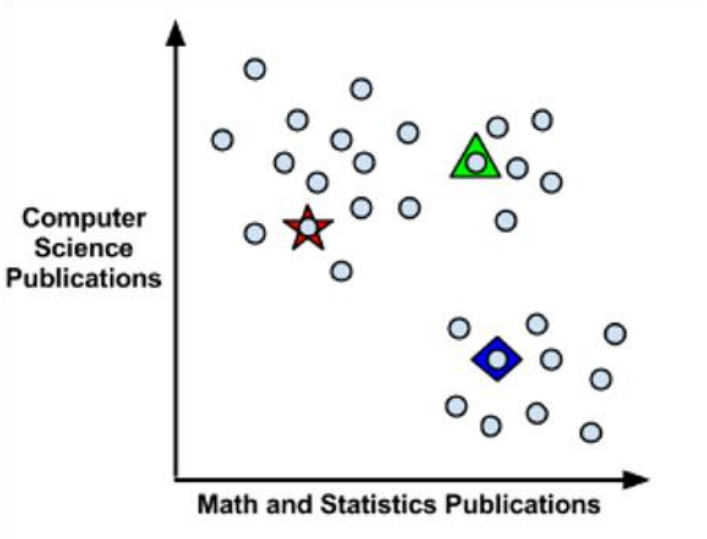
Para resolverlo, la k-media utiliza un **enfoque heurístico** para encontrar la solución óptima.

Cómo funciona

Como el algoritmo es heurístico (con una selección aleatoria de elementos en la primera iteración), los resultados de su aplicación con el mismo conjunto de datos pueden ser diferentes.

Como hemos estado considerando, cada elemento o muestra del conjunto de datos de entrada va a ser un vector de características (puntos multidimensionales)

El algoritmo comienza seleccionando aleatoriamente k puntos iniciales (otra posibilidad podría ser asignar arbitrariamente cada muestra a un clúster)



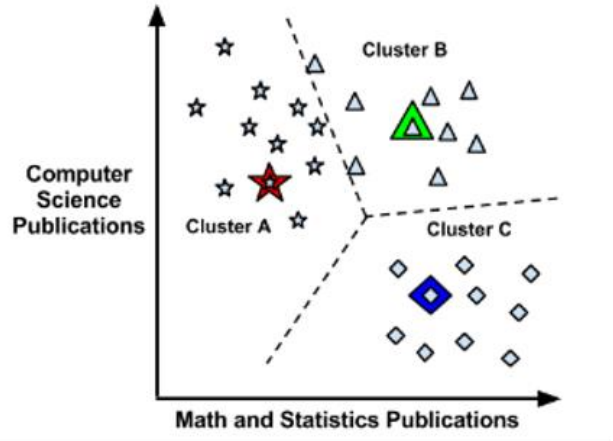
8. Aprendizaje no supervisado

K-means

Después de la selección de estos k puntos iniciales, estos puntos iniciales se consideran como los "centroides" iniciales de cada k clúster, y todo el resto de muestras del conjunto de datos de entrada se asignan al centroide más cercano basado, generalmente, en la distancia euclidiana.

$$\mathrm{dist}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Esto significa que debemos t[illegible] ás, es una buena práctica 'normalizar' o 'escalar' sus rangos antes de usar este algoritmo.



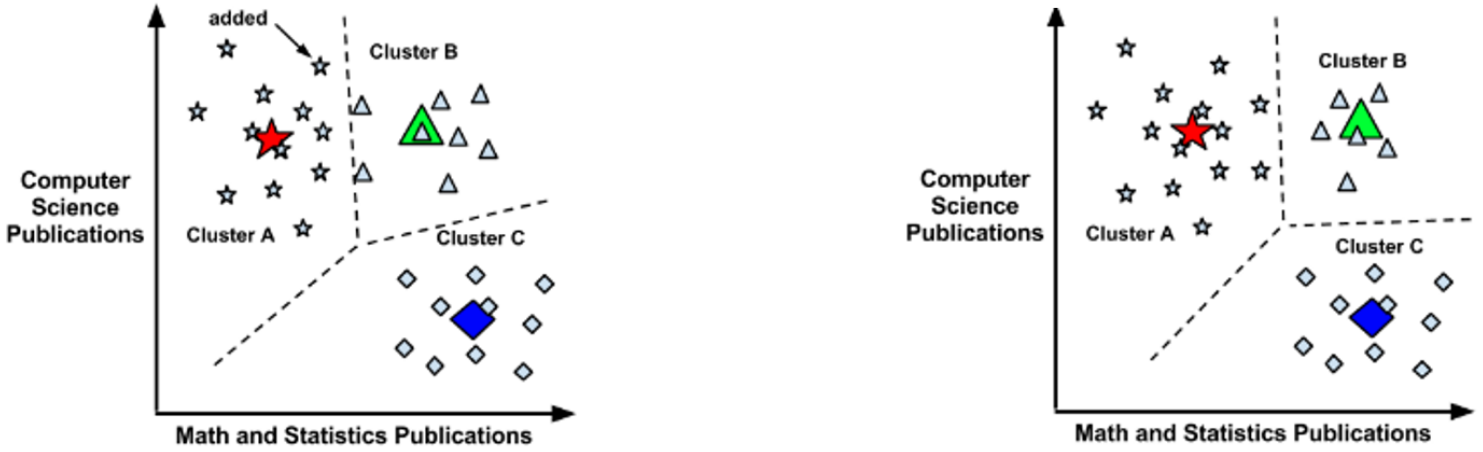
8. Aprendizaje no supervisado

K-means

Ahora, se calcula **un nuevo 'centroide' real** para cada uno de los k clústeres como la media de todas las muestras asignadas a este clúster.

A continuación, las muestras se reasignan al clúster cuyo centroide está más cerca.

Este proceso se repite hasta que las métricas de similitud converjan.



8. Aprendizeja no supervisado

K-means

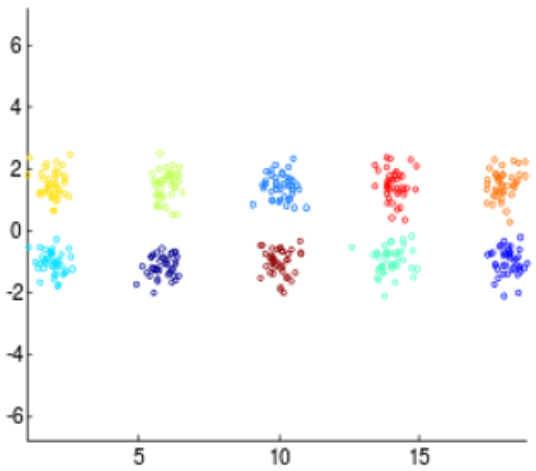
La **selección de k es realmente difícil**.

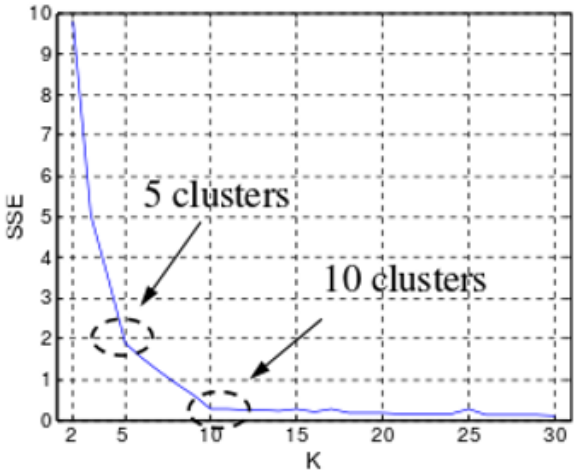
Un valor final alto de k puede producir un sobreajuste.

También es interesante cualquier conocimiento previo que podamos tener del contexto del conjunto de datos y si podemos tener una estimación de k.

Sin conocimientos previos, un valor de bien general es k = sqrt(n/2).

Es común utilizar lo que llamamos 'el método del codo', donde queremos ver la evolución de la métrica (por ejemplo, MSE) y k:





Valencian Graduate School and Research Network of Artificial Intelligence

valgrai.eu

valgrai.eu

Valencian Graduate School and Research Network of Artificial Intelligence