

- Árboles de decisión

+ Conceptos básicos

Un árbol de decisión es un grafo etiquetado que representa un concepto

* Nodos interiores: Attributos

* Arcos: valores del nodo origen

* Hojas: valor de calificación (+ ó -)

+ ID3

* Entropía: La entropía de un conjunto de ejemplos D .

$$\text{Ent}(D) = - \frac{|P|}{|D|} \cdot \log_2 \frac{|P|}{|D|} - \frac{|N|}{|D|} \cdot \log_2 \frac{|N|}{|D|}$$

Siendo P los casos positivos y N los negativos

Se denota como $\text{Ent}([p^+, n^-])$

Este valor nos permite calcular la homogeneidad del conjunto y tiene algunos casos interesantes como

- $\text{Ent}([K^+, K^-]) = 1$ (ausencia de homogeneidad)
- $\text{Ent}([K^+, v]) = 0$ (homogeneidad total)

* Entropía esperada: Buscamos los nodos con menor entropía, ya que, al elegir nos devolverá un árbol con mayor homogeneidad y por lo tanto éste es menor

$$\sum_{\forall \text{Valores}(A)} \frac{\text{IDval}}{|D|} \cdot \text{Ent}(Dv)$$

Siendo Dv el subconjunto de los elementos de D con el valor del atributo A igual a v

* Ganancia: Ganancia de información esperada después de usar un atributo A

$$\text{Ganancia}(D, A) = \text{Ent}(D) - \sum_{\forall \text{Valores}(A)} \frac{\text{IDvi}}{|D|} \cdot \text{Ent}(Dv)$$

Para el algoritmo ID3, tomamos el nodo con mayor ganancia

* Procedimiento algorítmico

1º Calculamos la entropía total del conjunto inicial

2º Calculamos la ganancia teniendo en cuenta el conjunto y cada atributo a estudiar (cada atributo es una columna de la tabla).

Las fórmulas de la entropía esperada nos permiten hacerlo por cada valor, por ejemplo, si la columna es Edad y tiene valores Roja, Azul y Verde, habrá que hacer el sumatorio de cada valor

3º Tomamos el valor con mayor ganancia y empezamos el árbol con esa categoría y un arco por valor.

4º Si hay arcos sin cerrar, repetimos el proceso, teniendo en cuenta que el conjunto total ha disminuido tras la primera clasificación, además solo se estudiarán las categorías no tomadas anteriormente

Sobreganancia: Se ajustan tanto los resultados al entrenamiento que no interpretan bien nuevos valores.

* Medida del rendimiento del aprendizaje

Para medir el rendimiento del modelo se utilizan conjuntos de entrenamiento y prueba (test). El conjunto de entrenamiento se usa para calibrar el modelo y el conjunto de prueba para medir el rendimiento (proporción de ejemplos bien clasificados).

Estratificación: cada clase correctamente representada.

A veces se utiliza un tercer modelo de validación.

Si no se tienen suficientes ejemplos se utilizan validación cruzada, esto consiste en dividir el modelo en K grupos y K aprendizajes; y realizar una media de los rendimientos.

* Sobreajuste

Una hipótesis H es un posible modelo al aprender (ej: árboles de decisión).

Una hipótesis $h \in H$ sobreajusta los ejemplos de entrenamiento, si existe una hipótesis h' que se ajusta peor que h (menos desarrollada) pero mejora la distribución completa.

Potibles causas

- Ruido (errores en los datos)
- Atributos irrelevantes
- Conjunto de entrenamiento pequeño

Maneras de evitarlo

- Evitar modelos complejos (penalizarlos)
- Medir el rendimiento sobre conjuntos de validación independientes

Evitar sobreajuste en árboles

Early stopping

Profundidad máxima

Parar en nodos con un porcentaje bajo de datos

Parar en nodos con mayoría de una clase

Podar a posteriori

Mirar ejemplos de podar

* Centinela o balanceo

Cuando los valores son numéricos, tomamos los valores frontera y creamos un atributo que contendrá los valores con mayor ganancia, iteraremos el proceso hasta eliminar los mínimos y valores fronteros.

- Reglas

+ Aprendizaje de reglas

* Reglas cubiertas: Una regla cubre un ejemplo si cumple las condiciones; la cubre correctamente si la conclusión es concluyente

Para estudiar las reglas se usa la frecuencia relativa (p_t), siendo t los ejemplos cubiertos y p los correctamente cubiertos

Las reglas están formadas por unas condiciones con conjunciones de atributos y un resultado (+ o -)

* Algoritmo de reglas

- 1º Dada una conclusión o resultado sumar la frecuencia relativa de todos los atributos y sus valores
 - 2º Seleccionamos el valor con mayor frecuencia y añadimos el atributo a la regla.
 - 3º Restamos a componer la frecuencia relativa con los atributos restantes y los valores actualizados.
 - 4º Iteramos hasta llegar a una frecuencia relativa igual a 1
 - 5º Una vez llegados a una regla que no tenga ejemplos sin cubrir, eliminamos los ejemplos que cubre y empezamos de nuevo hasta cubrir correctamente todo el resultado buscado
- # Si tienen la misma frecuencia relativa se toma la que produzca mayor ganancia
- $$\{P^* = \log_2 \left(\frac{P_t}{P_f} - \log_2 \frac{P_t}{P_f} \right)\}; P_t: \text{frecuencia relativa después de añadir la condición}; P_f: \text{antes de añadir}$$

- Basado en instancias: KNN (k nearest neighbors)

+ Conceptos básicos

Dado un conjunto de entrenamiento y un ejemplo nuevo, devuelven la categoría mayoritaria de los K ejemplos del conjunto de entrenamiento más cercanos al ejemplo que se quiere clasificar.

Los conjuntos de entrenamiento son vectores numéricos con categoría asignada

+ Distancias

* Euclídea = $\sqrt{\sum_{i=1}^n (x_i - y_i)^2}$ (para medir propiedades similares)

* Manhattan = $\sum_{i=1}^n |x_i - y_i|$ (Propiedades distintas)

* Hamming = Número de componentes en los que se difieren (vectores no numéricos)

Normalizar componentes de misma magnitud (restar media y dividir por la desviación típica).

+ Elección de K

Conocimiento específico del problema

Resultado de n niveles en conjuntos más pequeños

Si la clasificación es lineal: preferiblemente usar una K impar

+ Algoritmo K-NN con pesos

Consideraremos los K vecinos más cercanos $\{x_1, \dots, x_K\}$ al objeto x que queremos clasificar.

A cada uno de los K vecinos más cercanos se les asignará un peso:

$$w_i = \frac{1}{\text{dist}(x_i, x)}$$

Se suman los pesos de los distintos clasificadores y a "x" se le asignará la clasificación con mayor peso total.

+ Algoritmo K-NN NCC (Nearest Centroid classifier)

Dado un conjunto de entrenamiento sobre R^n junto a su clasificación y un nuevo punto x , NCC le aplicará al nuevo punto la clasificación cuyo centroide está más cercano al punto.

Calculamos el centroide de los puntos de cada clasificación y aplicamos K-NN sobre el conjunto de centroides con $K=1$

+ Algoritmo K-NN con rechazo

Para este, además del conjunto de entrenamiento D , el valor K y el objeto x , necesitaremos un umbral u .

El algoritmo toma los K puntos del conjunto más cercanos a x y devuelve la clasificación solo si la clasificación con más puntos supera el umbral.

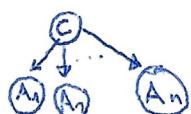
- Naive Bayes

+ Conceptos básicos

Poniendo de un conjunto de atributos A_1, \dots, A_n con C valores finitos cada clase, tenemos un conjunto de entrenamiento D , con una serie de tuplas de valores para cada atributo, con su clasificación, queremos aprender un clasificador para cada nueva instancia.

+ Superficie de Naive

Tanto los atributos como el valor de clasificación se consideran variables aleatorias. Dado el valor de clasificación, los atributos son condicionalmente independientes entre sí (c d-separa A_i, A_j)



+ Clasificadores Naive Bayes

Teniendo en cuenta las independencias condicionales entre atributos, podemos obtener el clasificador así:

$$c_{NB} = \arg \max_{c_j \in C} \prod_{i=1}^n P(A_i | c_j)$$

$$\text{Siendo } P(c_j) = \frac{n(c=c_j)}{N} \text{ y } P(A_i | c_j) = \frac{n(A_i = a_i, c=c_j)}{n(c=c_j)}$$

N : n° total de ejemplos

$n(c=c_j)$: n° ejemplos clasificados como c_j

$n(A_i = a_i, c=c_j)$: n° ejemplos de c_j que su valor de atributo A_i es a_i

+ Algoritmo Naïve Bayes

$$\hat{C}_{NB} = \operatorname{argmax}_{c_j \in \{+, -\}} P(c_j) P(\text{attr}_1 | c_j) P(\text{attr}_2 | c_j) \dots$$

Hay que estudiar el caso en el que c_j es positivo y condicionar los atributos como positivos; y por otro lado el caso en el que c_j es negativo y condicionar los atributos como negativos.

$P(c_j)$: contamos las veces positivas o negativas entre el total de ejemplos.

$P(\text{attr}_i | c_j)$: contamos las veces en las que se da el atributo i y se da c_j positivo o negativo, entre las veces en las que se da el atributo i y no se da c_j .

Aunque haya 15 ejemplos si hay 7 con atributo $+$, $P(\text{attr}_1 | c_j)$ como máxima sera $7/7$ y como mínima $0/7$, valores entre $[0, 1]$

+ Log-probabilidades

Para evitar que las probabilidades sean excesivamente bajas debido a un conjunto pequeño de entrenamiento o que estén sean 0 y lograr que las probabilidades de probabilidad sean 0, se plantean las probabilidades logarítmicas.

$$\hat{C}_{NB} = \operatorname{argmax}_{c_j \in C} [\log(P(c_j)) + \sum_i \log(P(\text{attr}_i | c_j))] \quad \# \text{Además de introducir log, cambiamos el producto por un sumatorio}$$

+ Suavizado

Para evitar los problemas anteriores y el sobreajuste; el suavizado consiste en añadir m ejemplos cuyos valores se distribuyen tecnicamente a priori. La estimación suavizada de una probabilidad se calcularía como:

$$\text{Est. suav.} = \frac{n+m \cdot p}{n+m} \quad \begin{array}{l} n: \text{ejemplos favorables y totales} \\ p: \text{estimación a priori de la probabilidad} \\ m: \text{constante (ejemplos adicionales)} \end{array}$$

+ Suavizado Laplace

Puntiendo del concepto anterior, podemos usar esta estimación:

$$\hat{P}(\text{attr}_i | c_j) = \frac{n(A_i = a_i, c = c_j) + K}{n(c = c_j) + K |A_i|} \quad \begin{array}{l} K: \text{constante fija} \\ |A_i|: n^{\circ} \text{ posibles valores de } A_i \text{ (atributo)} \end{array}$$

Normalmente se utiliza $K=1$

- Clustering

+ Conceptos básicos

Consiste en dividir un conjunto de entradas en subconjuntos (clusters), de forma que entre estos comparten un patrón o características o criterio deseado.

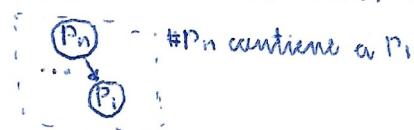
+ Partición estricta

Dada un conjunto D con vectores metricables como ejemplo, buscar una partición del conjunto D en K clusters, $P = \{C_1, \dots, C_K\}$ con $K \in \mathbb{N}$ tal que

$$\left\{ \begin{array}{l} C_i \neq \emptyset \text{ para } i \in \{1, \dots, K\} \text{ (No pueden haber clusters vacíos)} \\ \bigcup_{i=1}^{i=K} C_i = D \text{ (La unión de los clusters es el conjunto } D) \\ C_i \cap C_j = \emptyset \text{ siendo } i \neq j, \text{ para } i, j \in \{1, \dots, K\} \text{ (No puede haber un elemento en más de un cluster)} \end{array} \right.$$

+ Jerárquico

Dada un conjunto D como el anterior, buscar construir un conjunto de particiones enridadas con estructura de árbol, de forma que si $C_i \in P_m$ y $C_j \in P_l$, siendo $m > l$, entonces $C_i \subseteq C_j$ o $C_i \cap C_j = \emptyset$.



Mientras la partición estricta no tiene ningún tipo de estructura a los niveles de dividir el conjunto, la jerárquica crece de manera recursiva cada cluster desde uno general a un cluster que contiene un único elemento.

+ Proximidad

Clustering consiste en agrupar instancias según su proximidad, para ello, se usan como en clasificadores anteriores las distancias Euclídea, Manhattan y Hamming

+ Algoritmo K-medias

1º Asignamos K centros de forma aleatoria o con heurísticas (m_1, \dots, m_K)

2º Asignar al punto x_i centro m_x , el cluster m_x , cuya distancia sea menor

3º Asignar \bar{m}_x al cluster m_x la media aritmética de los ejemplos de este.

4º Iterar

La selección de los centros iniciales tienen gran importancia

+ Jerárquico (extra)

* Aglomerativo: De menos a más, partimos de un individuo y vamos agrupando hasta llegar a un cluster general

* Divisivo: contrario al aglomerativo

* Algoritmo jerárquico

1º Calculamos la matriz de proximidad para los N clusters (en un inicio todos los individuos)

2º Buscamos la menor distancia entre clusters en la matriz, combinamos en un mismo cluster los individuos que están a distancia mínima.

3º Actualizamos la matriz con los nuevos más grandes clusters

4º Iteramos

+ basado en densidad

Agrupamiento por regiones en función de la densidad de puntos, no es necesario dar el número de clusters de inicio

* DBSCAN

Parámetros

- └ minPts : mínimo de puntos por región
- └ ϵ : distancia

Clasificación de puntos

- └ p es núcleo si hay minPts o una distancia menor que ϵ
- └ q es frontera si hay menos minPts o una distancia menor que ϵ , pero está a menos de ϵ de distancia de un núcleo.
- └ r es ruido, ni núcleo ni frontera
- └ q es directamente alcanzable desde p si p es núcleo y la distancia es menor que ϵ
- └ Un punto es alcanzable desde p si hay una secuencia de p (núcleo) hasta dicho punto, donde haber alguna q en la secuencia.

Algoritmo (Entrada: Nube D, ϵ , minPoints)

- 1º Determinar núcleos, fronteras y ruido en función de ϵ y minPts
- 2º Crear un grafo conectando los p (núcleos) si su distancia es menor que ϵ
- 3º Conectar las fronteras a un p con distancia menor que ϵ
- 4º Desarrollar cada componente conexa como cluster