CHEATSHEET-SEGUNDO-PARCIAL.pdf

Golden_Hat

Sistemas Inteligentes

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad Politécnica de Valencia

Razonamiento probabilístico:

PROBABILIDAD:

Probabilidad condicional:

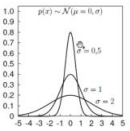
$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

$P(A|B)$ es **la probabilidad** de que **el efecto A** ocurra tras **observar que la causa B ha ocurrido**.

Independencia de variables:
Lo son si cumplen que...

$$P(x,y) = P(x) \qquad P(y)P(x|y) = P(x) \qquad P(y,x) = P(y)$$

Variables continuas y regla de bayes:



$$p(x) \sim \mathcal{N}(\mu, \sigma^2)$$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

$$P(x \in [\mu \pm 2\sigma]) = 0{,}95$$

Donde σ² = varianza y μ = mediana, media o moda...

Regla de decisión de bayes:

Elige de un set de hipótesis la de máxima probabilidad a posteriori. Es la mejor probabilidad posible a obtener:

$$c*(x) = \arg\max P(c|x)$$

La probabilidad de error de bayes es:

$$P(error\,|\,x) = 1 - P(c*(x)\,|\,x), \qquad P(c*(x)\,|\,x) = prob.\,de\,acertar$$

TEOREMA DE BAYES:

Permite actualizar el conocimiento sobre una hipótesis y después de observar una nueva evidencia x.

$$P(y|x) = \frac{P(x,y)}{P(x)} = P(y)\frac{P(x|y)}{P(x)}$$

Exista por ejemplo...

d	c	h	P
0	0	0	0,576
0	0	1	0,008
0	1	0	0,144
0	1	1	0,072
1	0	0	0,064
1	0	1	0,012
1	1	0	0,016
1	1	1	0,108
Suma:			1,000

La probabilidad de observar caries tras observar hueco es del 90%... ya que tenemos que evaluar la posibilidad de que haya también un hueco entre todas las posibilidades de que haya carie.

$$(c = 1|d = 1) = P(c = 1)\frac{P(d = 1|c = 1)}{P(d = 1)} = 0.34\frac{0.36}{0.20} = 0.61$$

El hecho de saber que hay dolor, cambia la probabilidad de que haya caries del 0.34 al 0.61.

EXAMPLE:

2023_01_17_Question 1:
Supongamos que tenemos dos cajas con 40 naranjas en la primera y 80 en la segunda. La primera caja contiene 9 naranjas Navelina y 31 Caracara. La segunda caja contiene tres veces más naranjas Navelina que Caracara. Ahora supongamos que **se elige una caja al azar (¡es decir probabilidad 50/50 entre ambas cajas!)** y **luego se elige una naranja al azar de la caja elegida**. Si la naranja elegida es Navelina, la probabilidad de que provenga de la primera casilla es:

(Observe que estamos eligiendo las primeras 2 casillas al azar... así que eso es

$$P(C = 1|T = N) = \frac{P(C = 1) * P(T = N|C = 1)}{P(C = 1) * P(T = N|C = 1) + P(C = 2) * P(T = N|C = 2)}$$

$$= \frac{1/2 \ * \ 9/40}{1/2 \ * \ 9/40 \ + \ 1/2 * 3/4} \ = \ 9/9 + 30 \ = \ 0.23$$

2022_01_27_Question 4:
Given the following table of joint frequencies of three variables of interest:

A	0	0	0	0	1	1	1	1
B	0	0	1	1	0	0	1	1
C	0	1	0	1	0	1	0	1
$N(A,B,C)$	124	28	227	175	126	222	23	75

¿Qué valor tiene $P(A = 1|B = 1, C = 0)$?

Es decir, qué probabilidad hay de que suceda A si se ha dado B y C:

$$P(A = 1|B = 1, C = 0) \ = \ \frac{23}{227 \ + \ 23}$$

Funciones discriminantes:

Un clasificador es una función definida tal que: $c(x) = \arg\max g_c(x)$.
Donde g_c **se define como la función discriminante de una clase, que determina el grado de pertenencia de un objeto x a dicha clase**.

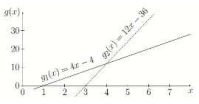
x_1	x_2	$g_1(x)$	$g_2(x)$	$g_3(x)$	$c(x)$
0	0	1.0	0.0	0.0	1
0	1	1/3	1/3	1/3	1
1	0	0.25	0.5	0.25	2
1	1	0.01	0.01	0.98	3

Puesto un ejemplo de clasificador con 3 clases... Vemos que en la columna c(x) se obtienen aquellas cuya probabilidad es **mayor**.

Clasificadores lineales: descritos en términos de función lineal.. tal que son capaces de describir una **función lineal que separe el espacio de datos**.

Están compuestos por:
Vector de pesos: W_c
Peso umbral: W_0

$$g_c(x) = \sum_d w_{cd} x_d + w_{c0} = w_c^t x + w_{c0}$$



Las fronteras de decisión surgen **al igualar las ecuaciones que definen a los clasificadores**. Sirven para definir correctamente la separación en clases del espacio de datos. El punto donde coinciden estos separadores (funciones discriminantes) es **la frontera de decisión**.

Las **regiones de decisión** son el **DOMINIO** donde el discriminante de una clase gana a los demás. En nuestro ejemplo, a partir del 4 g2, gana a g1 **y viceversa**.

Clasificador equivalente

Definido como...

$$c'(x) = \arg\max g_c'(x), \quad g_c'(x) = f(g_c(x)) + const(x)$$

Donde f es una función **estrictamente creciente**, y donde const(x) es toda y cualquier función que varíe o no con x **pero nunca con c**. Por ejemplo...

$$f(z) = log\ z, z > 0 \qquad f(z) = az + b, a > 0$$

ALGORITMO PERCEPTRÓN:

Este algoritmo toma un **dataset D** tal que $\{(x_n, y_n)\}$, donde x_n pertenece a un espacio R^D, y donde cada vector x va asociado a una clase y...

Y devuelve un **set de** $c(x) = \arg\max g_c(x)$, de la forma: $g_c(x) = W_c^t x + w_{c0}$ para todo c **expresado como set de productos escalares**. (Nótese que se utiliza en las trazas el algoritmo la notación **homogénea**, que incluye el término independiente en el vector de pesos).

Homogeneous or compact notation: $x = (1, x_1, \ldots, x_D)^t$ and $w_c = (w_{c0}, w_{c1}, \ldots, w_{cD})^t$; so, $g_c(x) = w_c^t x$

Su objetivo es **minimizar el número de errores de entrenamiento, usando para ello:**
- α : learning rate – controla el ritmo de aprendizaje.
- b: margen de tolerancia – utilizado para obtener resultados aceptables cuando las muestras no son **linealmente separables**.

Salida: $\{w_c\}^* = \arg\min_{\{w_c\}} \sum_n \left[\max_{c \neq c_n} w_c^t x_n + b > w_{c_n}^t x_n\right]$

Método:

$$|P| = \begin{cases} 1 & \text{si } P \text{ - verdadero} \\ 0 & \text{si } P \text{ - falso} \end{cases}$$

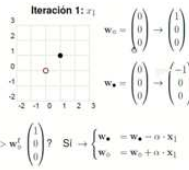
Un error se entiende cuando el valor máximo de la función discriminante de la clase que no es correcta (que no es la que está siendo procesada al momento) más el margen es mayor al valor de la función discrimintante de la clase correcta. El resultado de las [] se evalúa a 1 o 0 dependiendo de si el interior de los [] es verdadero, en cuanto a uno mayor que otro.

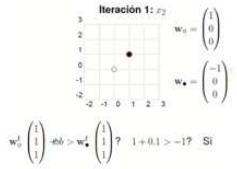
repetir
  para todo dato x_n
    $err = falso$
    para toda clase c distinta de c_n
      si $w_c^t x_n + b > w_{c_n}^t x_n$: $w_c = w_c - \alpha \cdot x_n$; $err = verdad$
    si err: $w_{c_n} = w_{c_n} + \alpha \cdot x_n$
hasta que no quedan muestras mal clasificadas

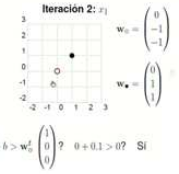
Este algoritmo **converge** si los datos son **linealmente separables**.

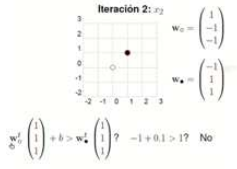
El efecto de alfa > 0 → convergencia independiente del valor determinado, pero lenta, si es muy próxima a 0.
Efecto de margen b >= 0 → converge con margenes centrados si el valor es cercano al que permite la clasificación lineal. Si es demasiado grande, no converge.

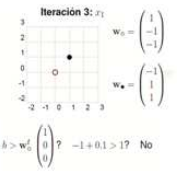
Veamos un ejemplo, incializado a alfa 1 y a beta 0.1

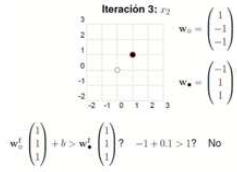






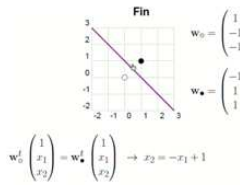






Como puede verse, el algoritmo perceptrón intenta ajustar poco a poco el vector de pesos para poder crear una frontera de decisión.

Esta frontera es el resultado de igualar lo de la derecha, que no es más que la expresión homogénea de una función discriminante, con lo de la izquierda.

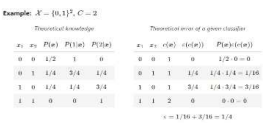


ESTIMACIÓN DEL ERROR:

Error a **posteriori**: $\varepsilon(c(x)) = 1 - P(c(x)|x)$

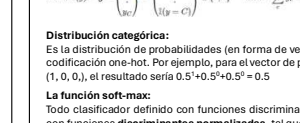
Error a **priori con χ:** $: E_x[\varepsilon(c(x))] = \sum_{x \in X} P(c(x)|x)$

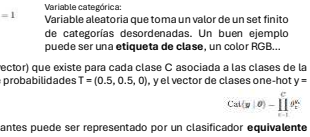
Example: $\mathcal{X} - \{0,1\}^2$, $C = 2$



Error del **clasificador de bayes:**
$\varepsilon(c*(x)) = 1 - P(c(x)|x) = 1 - max_c P(c|x)$

Es decir, no es más que un clasificador a posteriori que selecciona aquella clase con **mayor probabilidad a posterori para cada objeto x y calcula su error posible**.





Calcular el error teórico exacto es imposible.
Holdout:

$$\varepsilon_{N,M} = \frac{1}{M}\sum_{m=1}^{M} [c_N(x_m) \neq c_m] = \frac{\text{número de errores}}{M}$$

Resustitución:

$$\varepsilon_N^r = \frac{1}{N}\sum_{n=1}^{N} [c_N(x_n) \neq c_n] = \frac{\text{número de errores}}{N}$$

Podemos modelar la probabilidad de error de mediante una distribución normal donde la media es la prob de error media para el error de clasificador y varianza tal que...

Si $Var(\varepsilon_N)$ es despreciable y M es grande, podemos asumir que:

$$\varepsilon_{N,M} \sim \mathcal{N}\left(\mathbb{E}(\varepsilon_N), \frac{\mathbb{E}(\varepsilon_N)(1 - \mathbb{E}(\varepsilon_N))}{M}\right)$$

y podemos construir un ***intervalo de confianza al 95 %*** para ε_N,

$$P(\varepsilon_N \in I) = 0.95 \quad \text{amb} \quad I = \left[\varepsilon_{N,M} \pm 1.96\sqrt{\frac{\varepsilon_{N,M}(1 - \varepsilon_{N,M})}{M}}\right]$$

Regresión logística:

Codificación one-hot y variables categóricas:
Establece un vector de clases identificador por **índices** tal que la clase activa/representada se señala en su índice correspondiente con un 1, y todas las demás, con un 0.

$$\text{one-hot}(y) = \boldsymbol{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_C \end{pmatrix} = \begin{pmatrix} \mathbb{1}(y=1) \\ \vdots \\ \mathbb{1}(y=C) \end{pmatrix} \in \{0,1\}^C \quad \text{with} \quad \sum_c y_c = 1$$

Variable categórica:
Variable aleatoria que toma un valor de un set finito de categorías desordenadas. Un buen ejemplo puede ser una **etiqueta de clase**, un color RGB...

Distribución categórica:
Es la distribución de probabilidades (en forma de vector) que existe para cada clase C asociada a las clases de la codificación one-hot. Por ejemplo, para el vector de probabilidades T = (0.5, 0.5, 0), y el vector de clases one-hot y = (1, 0, 0,), el resultado sería $0.5^1 + 0.5^0 + 0.5^0 = 0.5$

$$\text{Cat}(\boldsymbol{y} \mid \boldsymbol{\theta}) = \prod_{c=1}^{C} \theta_c^{y_c}$$

La función soft-max:
Todo clasificador definido con funciones discriminantes puede ser representado por un clasificador **equivalente** con funciones **discriminantes normalizadas**, tal que...

$$c(\boldsymbol{x}) = \underset{c}{\text{argmax}}\ a_c = \underset{c}{\text{argmax}}\ e^{a_c} = \underset{c}{\text{argmax}}\ \frac{e^{a_c}}{\sum_{c'} e^{a_{c'}}}$$

La función softmax **transforma** un vector de **logits en un vector de probabilidades** *xd ¿?*

$$\mathcal{S}(\boldsymbol{a}) = \left[\frac{e^{a_1}}{\sum_c e^{a_c}}, \ldots, \frac{e^{a_C}}{\sum_c e^{a_c}}\right] \quad \text{satisfying} \quad 0 \leq \mathcal{S}(\boldsymbol{a})_c \leq 1 \quad \text{and} \quad \sum_c \mathcal{S}(\boldsymbol{a})_c = 1$$

$$p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \text{Cat}(\boldsymbol{y} \mid \mathcal{S}(f(\boldsymbol{x}; \boldsymbol{\theta}))) = \prod_c [\mathcal{S}(f(\boldsymbol{x}; \boldsymbol{\theta}))_c]^{y_c}$$

En vez de predecir una sola clase (la más probable), se predicen todas desde una función predictora logit, governada por un parámetro que en este caso es la **distribución categórica**

Veamos un ejemplo de perceptrón con los siguientes parámetros:

$$C = D = 2, \quad a_1 = g_1(x_1, x_2) = -x_1 - x_2 + 1, \quad a_2 = g_2(x_1, x_2) = x_1 + x_2 - 1$$

De los que se concluyen...

$$\boldsymbol{a} = f(\boldsymbol{x}; \mathbf{W}) = \mathbf{W}^t \boldsymbol{x} \quad \text{with} \quad \mathbf{W}^t = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \boldsymbol{x} = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

Siguiendo la fórmula:

$$p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}) = \text{Cat}(\boldsymbol{y} \mid \mathcal{S}(f(\boldsymbol{x}; \boldsymbol{\theta}))) = \prod_c (\mathcal{S}(f(\boldsymbol{x}; \boldsymbol{\theta}))_c)^{y_c}$$

$\boldsymbol{x}^t$	$\boldsymbol{a}^t$	$\mu_1 = \mathcal{S}(\boldsymbol{a})_1$	$\mu_2 = \mathcal{S}(\boldsymbol{a})_2$
$(1,0,0)$	$(1,-1)$	$\frac{e^1}{e^1+e^{-1}} = 0.8808$	$\frac{e^{-1}}{e^1+e^{-1}} = 0.1192$
$(1,1,1)$	$(-1,1)$	$\frac{e^{-1}}{e^{-1}+e^1} = 0.1192$	$\frac{e^1}{e^{-1}+e^1} = 0.8808$
$(1,0.5,0.5)$	$(0,0)$	$\frac{e^0}{e^0+e^0} = 0.5000$	$\frac{e^0}{e^0+e^0} = 0.5000$

Ejemplo: Sea un modelo de regresión para un problema de clasificación en 3 clases y para vectores de D = 2

$$p(y \mid \boldsymbol{x}; \boldsymbol{\theta}) = \text{Cat}(y \mid \mathcal{S}(\mathbf{W}^t \boldsymbol{x} + \boldsymbol{b})) \quad \text{with} \quad \mathbf{W}^t = \begin{pmatrix} 1 & 1 \\ -1 & 1 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{C \times D} \quad \text{and} \quad \boldsymbol{b} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

Con ello, la probabilidad de que P(x) pertenezca a la clase 1, cuando x =(0.5, 0.5) es de...

$$p(y=1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = \mathcal{S}\left(\begin{pmatrix} 1 & 1 \\ -1 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}\right)_1 = \mathcal{S}((1,0,0)^t)_1 = \frac{e}{e+2} = \frac{1}{1+2/e} = 0.5761$$

Apredizaje por máxima probabilidad:
Intenta establecer un criterio para aprender W de un dataset de entrenamiento, tal que siendo D nuestro dataset D = {(x_n, y_n)}...

Se busca la LL(W) (Log-likelihood of W given D and W).

$$\text{LL}(\mathbf{W}) = \log p(\mathcal{D} \mid \mathbf{W}) = \log \prod_{n=1}^{N} p(\boldsymbol{y}_n \mid \boldsymbol{x}_n, \mathbf{W})$$

$$= \sum_{n=1}^{N} \log \text{Cat}(\boldsymbol{y}_n \mid \boldsymbol{\mu}_n) \quad \text{with} \quad \boldsymbol{\mu}_n = \mathcal{S}(\boldsymbol{a}_n) \quad \text{and} \quad \boldsymbol{a}_n = \mathbf{W}^t \boldsymbol{x}_n$$

$$= \sum_{n=1}^{N} \log \prod_{c=1}^{C} \mu_{nc}^{y_{nc}} = \sum_{n=1}^{N} \sum_{c=1}^{C} y_{nc} \log \mu_{nc}$$

Example (cont.): log-likelihood of $\mathbf{W}^t = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix}$ with $\mathcal{D} = \{((1,0,0)^t, (1,0)^t), ((1,1,1)^t, (0,1)^t)\}$

$$\text{LL}(\mathbf{W}) = y_{11} \log \mu_{11} + y_{12} \log \mu_{12} + y_{21} \log \mu_{21} + y_{22} \log \mu_{22}$$
$$= \log \mu_{11} + \log \mu_{22}$$
$$= \log 0.8808 + \log 0.8808 = -0.1269 - 0.1269 = -0.253$$

Considerándolo un problema de minimización...
Hablamos de neg-log-likelihood, que es lo mismo pero con el signo cambiado y normalizado por el número de samples.

Example (cont.): neg-log-likelihood of $\mathbf{W}^t = \begin{pmatrix} 1 & -1 & -1 \\ -1 & 1 & 1 \end{pmatrix}$ with $\mathcal{D} = \{((1,0,0)^t, (1,0)^t), ((1,1,1)^t, (0,1)^t)\}$

$$\text{NLL}(\mathbf{W}) = -\frac{1}{2}\text{LL}(\mathbf{W}) = 0.1269$$

Descenso de gradiente:
Algoritmo iterativo que intenta minimizar un objetivo de un set de distribución categórica inicial dado.

Utiliza los parámetros que veremos en el ejemplo en la derecha.

Learning factor: $\eta_i > 0$ plays the same role as Perceptron; we can choose a sn constant value, $\eta_i = \eta$

Direction of steepest descent: $-\nabla \mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}_i}$ is the neg -gradient of the objecti evaluated at $\boldsymbol{\theta}_i$

Convergence: if η is not very large and the objective is convex (bowl-shaped), converges to a (global) minimum

Example: $\mathcal{L}(\theta) = \theta^2$, $\theta_0 = 9$, $\eta_t = 0.2$, $\frac{d\mathcal{L}}{d\theta} = 2\theta$ and tolerance 0.01

Ya que **no entiende esto ni *peter***, intentaremos entender el ejemplo.

Sea un modelo de regresión logística en notación compacta para un problema de clasificación en 3 clases, y datos representados en un espacio de 3 dimensiones.

$$p(\boldsymbol{y} \mid \boldsymbol{x}; \mathbf{W}) = \text{Cat}(\boldsymbol{y} \mid \mathcal{S}(\mathbf{W}^t \boldsymbol{x})) \quad \text{with} \quad \mathbf{W}^t = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \in \mathbb{R}^{}$$

Actualicemos el valor de W por valor de una iteración de **descenso de gradiente** con el set de datos D={x(1,1,1), y=1} y **con un learning factor tal que 0.1.**

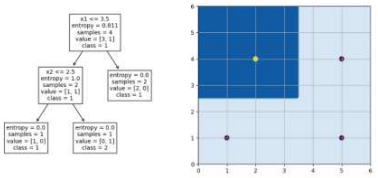
$$\boldsymbol{a} = \mathbf{W}^t \boldsymbol{x} = \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \qquad \boldsymbol{\mu} = S(\boldsymbol{a}) = \frac{1}{1+2e} \begin{pmatrix} 1 \\ e \\ e \end{pmatrix} = \begin{pmatrix} 0.1554 \\ 0.4223 \\ 0.4223 \end{pmatrix}$$

$$\mathbf{W} = \mathbf{W} - \eta \boldsymbol{x}(\boldsymbol{\mu} - \boldsymbol{y})^t$$
$$= \begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} - 0.1 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} (-0.8446, 0.4223, 0.4223)$$

$$= \begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} -0.0845 & 0.0422 & 0.0422 \\ -0.0845 & 0.0422 & 0.0422 \\ -0.0845 & 0.0422 & 0.0422 \end{pmatrix} = \begin{pmatrix} 1.0845 & -0.0422 & 0.9578 \\ -0.9155 & 0.9578 & -1.0422 \\ 0.0845 & -0.0422 & 0.9578 \end{pmatrix}$$

Árboles de clasificación:

Son una estructura utilizada para clasificar correctamente los objetos:



¿Cómo se construye un árbol de clasificación? Se sigue este algoritmo →

Árbol($S = \{(\boldsymbol{x}_n, c_n)\}$) // S es un conjunto de aprendizaje
$(\mathcal{C}, L, R, \Delta\mathcal{I}) =$ **Dicotomiza**(S)
si $\Delta\mathcal{I} < \epsilon$ **devuelve Nodo(Moda**($\{c_n\}$), $-$, $-$, $-$)
si no devuelve Nodo($-$, $\mathcal{C}$, **Árbol(**L**)**, **Árbol(**R**))**

Un árbol se crea partiendo de S donde S se entiende como un conjunto de aprendizaje {(xn, cn)}. Siendo C un criterio de partición, y L(conjunto de datos del lado izquierdo) y R(conjunto de datos del lado derecho) los conjuntos sobre los que se divide el conjunto C tras la dicotomización... se consigue un decremento de la impureza delta I. tras la misma.

Si el decremento de la impureza es menor a la tau determinada, que es un umbral de impureza, se considera que no vale la pena dicotomizar, por lo que dicho nodo a dicotomizar se convierte en nodo terminal y su etiqueta de clase es la clase más frecuente de los datos de ese nodo.

En caso contrario, se dicotomiza, y se crea un nodo interno sin etiqueta de clase que divide las clases según el criterio C, creando nodos hijos izquierdo y derecho.

Criterios de partición:

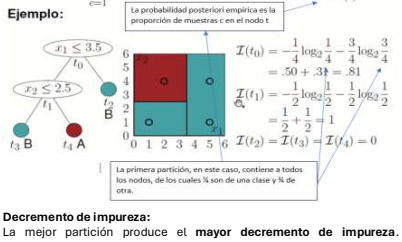
Un criterio de partición usual consiste en elegir un par variable umbral (d, r) y dividir los datos {(xn, cn)} de la siguiente manera.

$$L = \{(\boldsymbol{x}_n, c_n) : x_{nd} \leq r\} \quad \text{y} \quad R = \{(\boldsymbol{x}_n, c_n) : x_{nd} > r\}$$

Evaluación de la impureza de un nodo:
Suele evaluarse como la incertidumbre sobre la clase de los objetos en t, conocida como la entropía de la distribución empírica de la probabilidad posteriori de clases en t.

$$\mathcal{I}(t) = -\sum_{c=1}^{C} \hat{P}(c \mid t) \log_2 \hat{P}(c \mid t) \quad \text{donde} \quad \hat{P}(c \mid t) = \frac{N_c(t)}{N(t)}$$

Ejemplo:



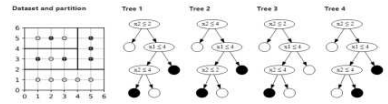
Decremento de impureza:
La mejor partición produce el **mayor decremento de impureza**. Recordamos que cuando la impureza de un nodo sea menor a la umbral, **dejaremos de particionar el nodo.**

$$(d^*, r^*) = \arg\max_{d,r} \Delta\mathcal{I}(d, r)$$

$$\Delta\mathcal{I}(d, r) = \mathcal{I}(t) - \frac{N(L(t))}{N(t)} \mathcal{I}(L(t)) - \frac{N(R(t))}{N(t)} \mathcal{I}(R(t))$$

EJEMPLOS:

The figure below shows a two-class 2d dataset along with a partition of the space into 4 regions, as well as four possible classification trees. Which of the four is consistent with the data and partition represented?



Solution: Tree 1.

2022_01_27_Question 1: Given the following 3 nodes of a classification tree with samples belonging to 3 classes:

c	1	2	3
n_1	2/12	5/12	5/12
n_2	3/11	4/11	4/11
n_3	5/11	2/11	3/11

where each row indicates the "posterior" probability of each class at the node. Which of the following inequalities is true?

1. $\mathcal{I}(n_1) < \mathcal{I}(n_3) < \mathcal{I}(n_2)$
2. $\mathcal{I}(n_3) < \mathcal{I}(n_2) < \mathcal{I}(n_1)$
3. $\mathcal{I}(n_1) < \mathcal{I}(n_2) < \mathcal{I}(n_3)$
4. $\mathcal{I}(n_2) < \mathcal{I}(n_3) < \mathcal{I}(n_1)$

Solution: option 1.

2023_01_26_Question 3: Consider the classification tree learning algorithm applied to a four-class problem, $c = 1, 2, 3, 4$. The algorithm has reached a node t with the following data: 2 from class 1, 16 from 2, 8 from 3 and 256 from 4. The impurity of t, $\mathcal{I}(t)$, measured as the entropy of the class posterior probability given by the empirical distribution in t, is:

1. $0.00 \leq \mathcal{I}(t) < 0.50$
2. $0.50 \leq \mathcal{I}(t) < 1.00$
3. $1.00 \leq \mathcal{I}(t) < 1.50$
4. $1.50 \leq \mathcal{I}(t)$

Solution: option 2.

$$\mathcal{I}(t) = -\sum_{i=1}^{4} \hat{P}(c \mid t) \log_2 \hat{P}(c \mid t)$$
$$= -\frac{2}{282}\log_2\frac{2}{282} - \frac{16}{282}\log_2\frac{16}{282} - \frac{8}{282}\log_2\frac{8}{282} - \frac{256}{282}\log_2\frac{256}{282} = 0.56$$

CLUSTERING: Algoritmo de K-medias:

El aprendizaje no supervisado, **o clustering** es un problema clásico del aprendizaje automático. La aproximación más usual es la del clustering particional, que basa en crear clusters de datos basado en las agrupaciones naturales de los datos en el espacio de representación.

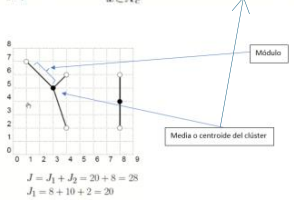
Asumamos la existencia de una función criterio J que evalúa la calidad de cualquier partición de N datos en C clusters.

$$J(\Pi) \; : \; \Pi = \{X_1, \ldots, X_C\} \qquad \Pi^* = \arg\min_{\Pi = \{X_1, \ldots, X_C\}} J(\Pi)$$

El problema del clústering se reduce entonces a lo siguiente: **encuéntrese la partición pi estrella que minimice el criterio J**.

Para ello, se selecciona una función **criterio que permite** encontrar este **pi estrella**. Esto se hace meidante el algoritmo SEC (Suma de errores cuadráticos), y se define tal que *nc* es la talla, es decir, el número de puntos de una clase c, y ||x|| la distancia euclidea (que como está elevada al cuadrado, se debe dejar únicamente con los componentes al cuadrado y sumados, ya que nos cargamos la raíz cuadrada!)

$$J(\Pi) = \sum_{c=1}^{C} J_c \qquad J_c = \sum_{\boldsymbol{x} \in X_c} \|\boldsymbol{x} - \boldsymbol{m}_c\|^2, \quad \boldsymbol{m}_c = \frac{1}{n_c} \sum_{\boldsymbol{x} \in X_c} \boldsymbol{x}$$



$J = J_1 + J_2 = 20 + 8 = 28$
$J_1 = 8 + 10 + 2 = 20$
$J_2 = 4 + 4 = 8$

Algoritmo de C-medias de duda y hart:

Dada una partición, el incremento de la SEC debido a la transferencia de un dato del clúster i al j es...

$$\triangle J = \frac{n_j}{n_j + 1} \|\boldsymbol{x} - \boldsymbol{m}_j\|^2 - \frac{n_i}{n_i - 1} \|\boldsymbol{x} - \boldsymbol{m}_i\|^2$$

La transferencia será provechosa **si delta J es menor que 0.**

- ***Entrada:*** una partición inicial, $\Pi = \{X_1, \ldots, X_C\}$
- ***Salida:*** una partición optimizada, $\Pi^* = \{X_1, \ldots, X_C\}$
- ***Método:***

Calcular medias y J
repetir
  para todo dato x
    Sea i el clúster en el que se encuentra x
    Hallar un $j^* \neq i$ que minimice $\triangle J$ al transferir x de i a j^*
    Si $\triangle J < 0$, transferir x de i a j^* y actualizar medias y J
hasta no encontrar transferencias provechosas

Este algoritmo recorrería todos los datos del espacio y haría transferencias al momento en función de si encuentra provechoso mover un dato de un set a otro.

C-medias convencional:

$$\|\boldsymbol{x} - \boldsymbol{m}_j\|^2 < \|\boldsymbol{x} - \boldsymbol{m}_i\|^2$$

Algoritmo de Lloyd's:

easy; each data sample is assigned to its nearest centroid.

$$J(\boldsymbol{m}_1, \ldots, \boldsymbol{m}_K) = \sum_{n=1}^{N} \min_{k=1,\ldots,K} \|\boldsymbol{x}_n - \boldsymbol{m}_k\|_2^2$$

Evaluación del clustering:

Se hace con el **índice Rand**: mide la **similaridad** entre 2 particiones de un set X de N samples de datos, una partición refernicia, y otra partición predicción:

$$\text{RI} = \frac{a + b}{a + b + c + d}$$

A → número de pares de X que están en el mismo cluster que R y que P
B → número de pares de X que están en el clusters diferentes que R y que P
C → número de pares de X que están en el mismo cluster que R y en diferente P.
D → número de pares de X que están en diferente cluster que R y en el mismo P.

Example: $N = 4$ data samples with reference labels $R = (0, 0, 1, 1)$ and prediction $P = (2, 2, 0, 1)$

$\{i, j\}$	1	2	3	4
1		a	b	b
2			b	b
3				c

$\Rightarrow \quad \text{RI} = \frac{1+4}{6} = 0.83$