APRENDIZAJE AUTOMATICO

CLASIFICACIÓN CON K-VECINOS MÁS PRÓXIMOS (K-NN)

ENTRADA: datos entrenamiento $\mathcal{D}_{train}: \{(\underset{\sim}{x}_n, c_n),\ n=1,\dots,N\}$

ejemplos de test $\mathcal{D}_{test}$ o $\underset{\sim}{X}_{test}$

número de vecinos K

métrica o función distancia $d(\cdot,\cdot)$

SALIDA: clase predicha por K-NN para el ejemplo de test $\underset{\sim}{X}_{test}$

CÓDIGO:
1. Encontrar los K-vecinos más próximos a X_{test} en $\mathcal{D}_{train}$.
2. Encontrar la clase mayoritaria entre esos K-vecinos más próximos a X_{test}.
3. Asignar a X_{test} esa clase mayoritaria

OBSERVACIONES:

- No funciona bien si la dimensión D es muy grande. Una forma de intentar resolver esto sería seleccionar las variables más significativas.
- Si hay ruido no funciona bien, debido a que el ruido "contamina" la distancia.

TEOREMA DE BAYES EN K-NN

Denominemos: $R(X_{test})$ = elemento de volumen centrado en X_{test}.

$N(R(X_{test}))$ = ejemplos contenidos en $R(X_{test})$

$N_c(R(X_{test}))$ = ejemplos de la clase c en $R(X_{test})$.

Entonces $$\mathbb{P}(c|X_{test}) \approx \frac{N_c(R(X_{test}))}{N(R(X_{test}))}$$

MAP: $$C^*(X_{test}) = \max_c \mathbb{P}(c|X_{test}) \approx \max_c N_c(R(X_{test}))$$

r tanto, K-NN puede ser visto como un predictor MAP que liza estimaciones locales de las densidades de probabilidad levantes.

¿CUÁNTOS VECINOS? ¿VALOR DE K ÓPTIMO?

El nº de vecinos actúa como un "smoothing parameter"

K pequeño: muchas regiones y más pequeñas

K grande: menos regiones y más grandes.

El K óptimo puede ser determinado mediante algoritmos de validación cruzada:

- LOOCV: Leave One Out Cross Validation
- K-FCV: K-fold Cross Validation

CLASIFICACIÓN CON ÁRBOLES DE DECISIÓN

Buscamos la primera pregunta que mejor determine la clasificación lo mejor posible.

La mejor pregunta es la que permite obtener más información sobre la clase.

Necesitamos cuantificar la ganancia de información

$$\underset{\sim}{\mathbb{P}} = \{\mathbb{P}\}_{K=1}^{K} \qquad H(\underset{\sim}{\mathbb{P}}) = \sum_{K=1}^{K} \mathbb{P}_K \cdot \log_2(\mathbb{P}_K)$$

$\ln \rightarrow$ NATS

$\log_2 \rightarrow$ unidades BITS

ENTROPÍA

La ganancia de información aportada por la variable aleatoria X sobre la variable G se define como:

$$GI(X, G) = H(G) - H(G|X)$$

Donde:

$H(G)$ = mide el desconocimiento que tenemos de la variable G antes de saber X.

$H(G|X)$ = mide el desconocimiento sobre G después de saber el valor de X.

$$h(x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_D x_D = (w_0 \; w_1 \; \cdots \; w_D) \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_D \end{pmatrix} \text{ con } x_0 =$$

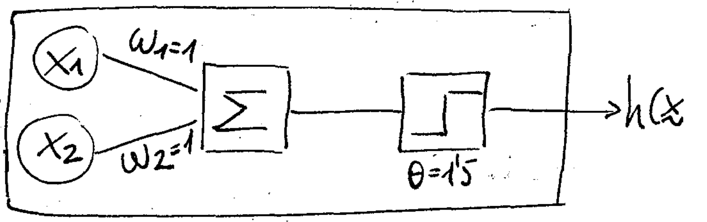
$$= \boxed{\underset{\sim}{w}^T \cdot \underset{\sim}{x}}$$

$\uparrow$ modelo para predecir y

$$ECM(\underset{\sim}{w}) = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} \left(\underset{\sim}{w}^T \cdot \underset{\sim}{x}_n - y_n \right)^2$$

$$w^*_{LS} = \underset{\underset{\sim}{w}}{\operatorname{argmin}} \left[ECM(\underset{\sim}{w}) \right]$$ Mínimos cuadrados (Least squares)

Ejemplo: puerta lógica AND $\underset{\sim}{x} \rightarrow$



lo mismo para la puerta OR cambiando $\theta = 1'5$ por $\theta = 0'5$

Existen más soluciones.

MÉTODOS DE APRENDIZAJE

(I) MÁXIMO A PRIORI : clasificación por mayoría (maximizando priores)

$$h(\underset{\sim}{x}) = \underset{clase \in \{c_1, \dots, c_K\}}{\operatorname{argmax}} \mathbb{P}(clase)$$

(II) ML

$$h(\underset{\sim}{x}) = \underset{clase \in \{c_1, \dots, c_K\}}{\operatorname{argmax}} \mathbb{P}\left(\{(\underset{\sim}{x}_n, c_n)\}_{n=1}^{N_{train}} \,\middle|\, clase \right)$$

(III) MAP

$$h(\underset{\sim}{x}) = \underset{clase \in \{c_1, \dots, c_K\}}{\operatorname{argmax}} \mathbb{P}\left(\{(\underset{\sim}{x}_n, c_n)\}_{n=1}^{N_{train}} \,\middle|\, clase \right) \cdot \mathbb{P}(clase)$$

FEED-FORWARD MULTILAYER PERCEPTRÓN

- PARÁMETROS: pesos sinápticos de conexiones entre neuronas
- HIPERPARÁMETROS
  - → learning rate
  - → tipo y peso de regularización (TIKHONOV) [resolución de ED's, álgebra lineal] → modelos más robustos

- ARQUITECTURA: grafo de la red neuronal

Observación: el ruido regulariza → el descenso del gradiente estocástico es mejor que el descenso del gradiente normal.

REGRESIÓN LOGÍSTICA (método lineal para clasificación)

$\boxed{\text{SIGMOIDAL}}$

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad \xrightarrow{z\to-\infty} 0, \quad \xrightarrow{z\to\infty} 1 \qquad \sigma(0) = 1/2$$

LOGIT

$\mathbb{P}(C_n = 1 | \underset{\sim}{x_n})$ salida de la sigmoidal ≡ ≡ Posterior de la clase 1

$\boxed{\text{TANH}}$

$$f(z) = \qquad \xrightarrow{z\to-\infty} -1, \quad \xrightarrow{z\to\infty} 1$$

$\boxed{\text{PROBIT}}$

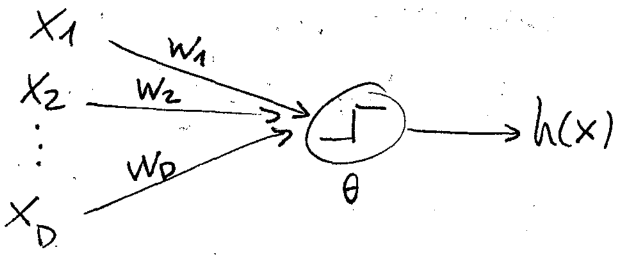
$$\text{normpdf} = \frac{e^{-z^2/2}}{\sqrt{2\pi}} \qquad N(0,1)$$

$$\text{normcdf}(z) = \int_{-\infty}^{z} \text{normpdf}(z')\,dz'$$

$$= \mathbb{P}(Z \leq z) \qquad Z \sim N(0,1)$$

REDES NEURONALES

Básica:



$$h(x) = \begin{cases} 1 & \text{si } w_1x_1 + \cdots + w_dx_d \geq \\ 0 & \text{si } w_1x_1 + \cdots + w_dx_d < \end{cases}$$

umbral: θ

$h(x) :=$ hipótesis

⌐ función de activaci

$$h(x) = \begin{cases} 1 & \text{si } \underset{\sim}{\omega}^T \underset{\sim}{x} \geq \theta \\ 0 & \text{si } \underset{\sim}{\omega}^T \underset{\sim}{x} < \theta \end{cases}$$

FUNCIONES DE ACTIVACIÓN PRINCIPALES:

- FUNCIÓN SIGMOIDE: $\sigma(z) = \dfrac{1}{1+e^{-z}}$
  (LOGIT)

  $\sigma(-\infty) = 0$ $\quad \sigma(0) = \frac{1}{2}$

  $\sigma(\infty) = 1$

  monótona creciente

  $\sigma'(z) = \sigma(z)$

- FUNCIÓN DE ACTIVACIÓN LINEAL: $f(z) = z$

  Neurona de salida para regresión

- TANGENTE HIPERBÓLICA: $\tanh(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}$

  $\tanh(-\infty) = -1$

  $\tanh(0) = 0$

  $\tanh(\infty) = 1$

- FUNCIÓN DE ACTIVACIÓN PROBIT:
  (GAUSSIANA)

  $$\text{normpdf}(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \int_{-\infty}^{z} e^{-\frac{y^2}{2\sigma^2}} dy$$

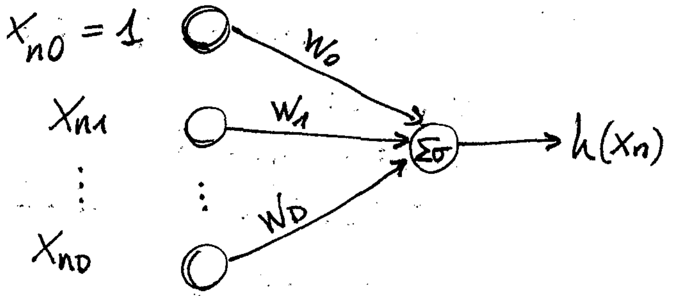
ARQUITECTURA BÁSICA

Una red neuronal es un conjunto de neuronas interconectadas.

La arquitectura de una red neuronal está determinada por la forma en que las neuronas están conectadas entre sí.

- Nodos fuente o neuronas de entrada (input layer): proporcionan la señal de entrada.
- Nodos de computación o neuronas ocultas (hidden layer): computan las señales de la capa anterior junto con sus pesos.
- Nodos de salida o neuronas de salida (output layer)

REGRESIÓN LOGÍSTICA MEDIANTE UN PERCEPTRÓN LINEAL



$w_0 x_1 + \cdots + w_D x_D = w^T x \Rightarrow$

$\Rightarrow \underset{\sim}{w}^T \underset{\sim}{x_n} = \sum_{d=0}^{D} w_d x_{nd}$

Función de activación: $\sigma(\underset{\sim}{w}^T \underset{\sim}{x_n})$

Salida: $h(x_n) = \sigma(\underset{\sim}{w}^T \underset{\sim}{x_n})$

APRENDIZAJE POR MÁXIMA VEROSIMILITUD

Datos: $\{(x_n, t_n)\}_{n=1}^{N}$ $\qquad \underset{\sim}{w}_{ML} = \underset{w}{\arg\max} \left\{ \mathcal{L}\left(\{(x_n, t_n)\}_{n=1}^{N} ; w\right)\right\}$

$= \mathbb{P}\left(\{t_n\}_{n=1}^{N} \mid \{x_n\}_{n=1}^{N}, w\right)$

Función de error: entropía cruzada

$E(\underset{\sim}{w}) = \sum_{n=1}^{N} E_n(\underset{\sim}{w})$ con $E_n(\underset{\sim}{w}) = t_n \log \underset{\underset{w}{\uparrow}}{h}(x_n) + (1-t_n)\log(1-\underset{\underset{w}{\uparrow}}{h}(x_n$

Aprendizaje ML $\equiv$ Minimizar $E(\underset{\sim}{w})$

Utilizando esto, basamos los siguientes algoritmos de aprendizaje usando redes neuronales.

APRENDIZAJE POR LOTES (Batch)

INPUT: datos de entrenamiento: $\{(x_n, t_n)\}_{n=1}^{N}$ donde $t_n = \begin{cases} 1 & \text{si } C_n = C_1 \\ 0 & \text{si } C_n = C_0 \end{cases}$

parámetro de aprendizaje: η

OUTPUT: parámetros del modelo (pesos de la red) $\underset{\sim}{w}_{ML}$

PSEUDOCÓDIGO:

1. Inicializa los pesos aleatoriamente $\underset{\sim}{w} \sim U[-0'5, 0'5]^{D+1}$
2. Mientras no haya convergencia (o niter < max_iter):
   - 2.1. Calcular salida de la red con los pesos actuale
     $h(x_n) = \sigma(\underset{\sim}{w}^T x_n)$ para $n = 1, 2, \ldots, N$
   - 2.2. Calcular gradiente (descenso del gradiente).
     $$\frac{\partial E(\underset{\sim}{w})}{\partial \underset{\sim}{w}} = \sum_{n=1}^{N} (h(x_n) - t_n) x_n$$
   - 2.3. Actualizar pesos:
     $$\underset{\sim}{w} = \underset{\sim}{w} - \eta \frac{\partial E(\underset{\sim}{w})}{\partial \underset{\sim}{w}}$$

APRENDIZAJE EN LÍNEA (Online)

INPUT: datos de entrenamiento $\{(x_n, t_n)\}_{n=1}^{N}$ $\quad t_n = \begin{cases} 1 & \text{si } C_n = C_1 \\ 0 & \text{si } C_n = C_0 \end{cases}$

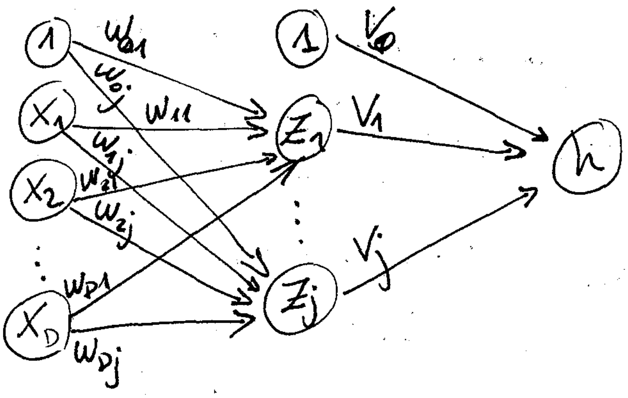
parámetro de aprendizaje : η

OUTPUT: parámetros del modelo (pesos de la red) : $\underset{\sim}{\omega}_{ML}$

PSEUDOCÓDIGO:

1. Inicializar los pesos aleatoriamente $\underset{\sim}{w} \sim U(-0'5, 0'5)^{D+1}$
2. Inicializar el contador de épocas: $K = 0$
3. Mientras no haya convergencia (o niter < max_iter):
   - 3.1. Actualizar contador de épocas : $K \mathrel{+}= 1$
   - 3.2. Calcular la salida de la red :
     $$h(x_n) = \sigma(\underset{\sim}{\omega}^T x_n) \qquad n = 1, \dots, N$$
   - 3.3. Gradiente:
     $$\frac{\partial E(\underset{\sim}{w})}{\partial \underset{\sim}{w}} = \sum_{n=1}^{N} (h(x_n) - t_n)\, x_n$$
   - 3.4. Actualizar pesos:
     $$\underset{\sim}{w} = \underset{\sim}{w} - \eta \frac{\partial E}{\partial \underset{\sim}{\omega}}(\underset{\sim}{\omega})$$

PERCEPTRÓN MULTICAPA :



$$z_j = \sigma\left(\sum_{d=0}^{D} w_{dj} x_d\right) = \sigma(\underset{\sim}{w}_j^T \underset{\sim}{x})$$

$j = 1, \dots, J$ donde $J :=$ #neuronas capa oculta

$$h(x) = \sigma\left(\sum_{j=0}^{J} v_j z_j\right) = \sigma(\underset{\sim}{v}^T \cdot \underset{\sim}{z})$$

PROPAGACIÓN HACIA ATRÁS (BACK PROPAGATION)

INPUT: datos de entrenamiento $\{(x_n, t_n)\}_{n=1}^{N}$ con $t_n = \begin{cases} 1 & \text{si } C_n = C_1 \\ 0 & \text{si } C_n = C_0 \overset{\shortparallel}{} C_2 \end{cases}$

parámetro de aprendizaje: η

OUTPUT: pesos del perceptrón multicapa $\underset{\approx}{W}_{ML}$ ← matriz (por capa)

SEUDOCÓDIGO: 1. Inicializar pesos aleatoriamente

$\underset{\sim}{W}, \underset{\sim}{V} \sim U[-0'5, 0'5]$

2. Inicializar contador de épocas: nEpoch = 0

3. Mientras no haya convergencia (o niter < max_iter):

3.1. Actualizar contador de épocas: nEpoch += 1

3.2. For $n = 1, \ldots, N$:

$z_{nj} = \sigma(V_j^T x_n) \quad j = 1, \ldots, J$

$h(x_n) = \sigma(W^T \cdot z_n)$

$\delta_n = h(x_n) - t_n$ (error de predicción)

$\Delta_{nj} = z_{nj}(1 - z_{nj}) W_j \delta_n \quad j = 1, \ldots, J$ (asigna error a ocultas)

$\underset{\sim}{W} = \underset{\sim}{W} - \eta \delta_n \underline{z_n}$ (actualizar pesos de oculta a salida)

$\underset{\sim}{V_j} = \underset{\sim}{V_j} - \eta \Delta_{nj} x_n \quad j = 1, 2, \ldots, J$

(actualizar pesos de entrada a ocultas)

APRENDIZAJE AUTOMÁTICO

$$\mathbb{E}(x^{(i)}) \equiv \text{media}\left(\{x_n^{(i)}\}_{n=1}^{N_{train}}\right) = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} x_n^{(i)}$$

$$V(\{x_n\}_{n=1}^{N_{train}}) \equiv \text{varianza}\left(\{x_n\}_{n=1}^{N_{train}}\right) = \mathbb{E}\left(\left(x^{(i)} - \mathbb{E}(x^{(i)})\right)^2\right) =$$

$$= \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} \left(x_n^{(i)} - \mathbb{E}(x^{(i)})\right)^2$$

$$\text{std}\left(\{x_n\}_{n=1}^{N_{train}}\right) = \sqrt{V(\{x_n\}_{n=1}^{N_{train}})} = \sqrt{\text{varianza}}$$

REESCALADO/NORMALIZACIÓN

- $$x_j^{(i)} \longrightarrow \frac{x_j^{(i)} - \text{mediana}\left(\{x_n^{(i)}\}_{n=1}^{N_{train}}\right)}{\text{espacio\_intercuartílico}\left(\{x_n\}_{n=1}^{N_{train}}\right)}$$

  mediana → estimación ROBUSTA del "centro" de la distribución

  espacio_intercuartílico → estimación más ROBUSTA de la escala de la DIST.

  $$\text{Rango Intercuartículo} \equiv \text{percentil}\left(\{x_n\}_{n=1}^{N_{train}}, 0'75\right) - \text{percentil}\left(\{x_n\}_{n=1}^{N_{train}}, 0'25\right)$$

  $$\text{Mediana} \equiv \text{percentil}\left(\{x_n\}_{n=1}^{N_{train}}, 0'5\right)$$

- $$\frac{x_j^{(i)} - \left(\frac{\max + \min}{2}\right)}{\left[\max\left(\{x_n\}_{n=1}^{N_{train}}\right) - \min\left(\{x_n\}_{n=1}^{N_{train}}\right)\right]}$$ aquí suponemos $\tilde{x}_n \sim U[-1,1]$

- $$\frac{x_j^{(i)} - \mathbb{E}\left(\{x_n\}_{n=1}^{N_{train}}\right)}{\text{std}\left(\{x_n\}_{n=1}^{N_{train}}\right)}$$ $\tilde{x}_n \sim N(0,1)$

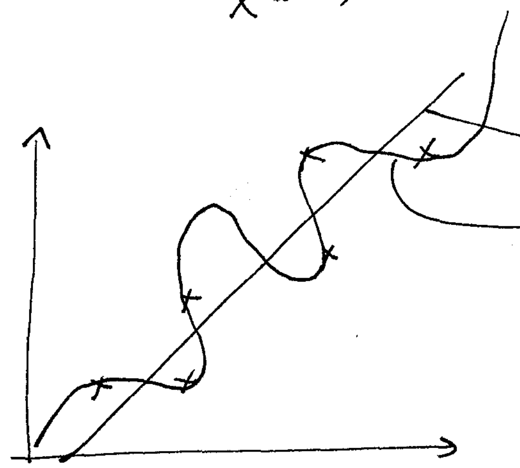
$$\mathcal{D}_{train} = \{(x_n, c_n)\}_{n=1}^{N_{train}} \xrightarrow{\text{Inducción}} \text{PREDICTOR} \qquad \underset{\sim}{x_n} = \begin{pmatrix} x_{n1} \\ x_{n2} \\ \vdots \\ x_{nd} \end{pmatrix} = \begin{pmatrix} x_n^{(1)} \\ x_n^{(2)} \\ \vdots \\ x_n^{(d)} \end{pmatrix}$$

$n = 1, \dots, N_{train}$
$d = 1, \dots, D$

VECINOS PRÓXIMOS

INPUT: $\mathcal{D}_{train}$
distancia
K: # vecinos
$x^{(test)}$

- No funciona bien si la dimensión D es muy grande
  ↳ ¿selección de variables?
- Si hay ruido no funciona bien, debido a que el ruido "contamina la distancia.



→ recta (modelo lineal) Error > 0

→ polinomio de alto nvl Error = 0

Capacidad de generalización: estimar mediante la tasa de acierto en $\mathcal{D}_{train}$

El polinomio complejo es un ej. de SOBREAJUSTE (OVERFITTING)
La recta suele SUBAJUSTAR (UNDERFITTING)

BOOTSTRAP (remuestreo):

CROSS-VALIDATION (validación-cruzada)
- loo: leave one out
- validación simple
- K-fold cross validation

Para K-NN suele ser
$K < \sqrt{N_{train}}$

Recordar el REESCALADO:

$$x_n^{(d)} \longrightarrow \frac{x_n^{(d)} - \text{centro}}{\text{escala}}$$

centro	escala	
media	varianza	$X \sim N(0,$
mediana	iqr	
$\frac{\max+\min}{2}$	$\frac{\max-\min}{2}$	$X \sim U(0,$

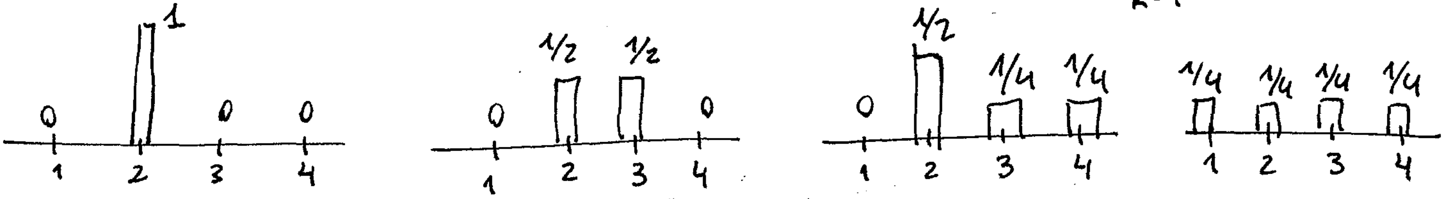
Más usada → media
Más robusta → mediana
valores entre 0 y 1 → $\frac{\max+\min}{2}$

ENTROPÍA: ¿Cuán diferente es una distribución de probabilic[idad] de la distribución UNIFORME?

$$\underset{\sim}{\mathbb{P}} = \{\mathbb{P}\}_{k=1}^{K} \qquad H[\underset{\sim}{\mathbb{P}}] = -\sum_{k=1}^{K} \mathbb{P}_k \log \mathbb{P}_k$$

base e NATS
base 2 BITS

$$0 \leq \mathbb{P}_k \leq 1 \qquad \sum_{k=1}^{K} \mathbb{P}_k = 1$$



0 bits   1 bit   1'5 bits   2 bits

ENTROPÍA : $H\left(\{\mathbb{P}_k\}_{k=1}^{K}\right)$ $\qquad 0 \leq \mathbb{P}_k \leq 1; \quad k = 1, 2, \dots, K$

$$\sum_{k=1}^{K} \mathbb{P}_k = 1$$

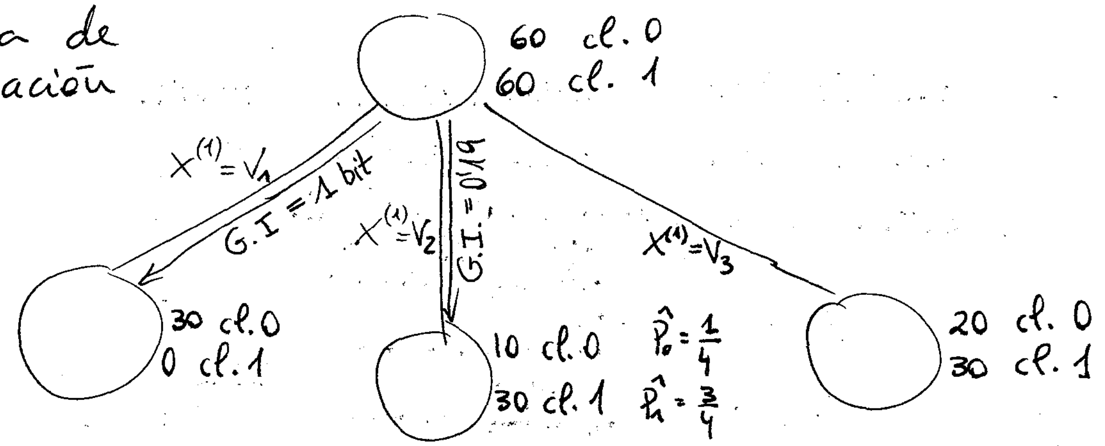
$$H\left(\{\mathbb{P}_k\}_{k=1}^{K}\right) = -\sum_{k=1}^{K} \mathbb{P}_k \log_2 \mathbb{P}_k \quad \text{(bits)} \qquad (\ln \to \text{NATS})$$

ENTROPÍA BINARIA: $H(\mathbb{P}) = -\mathbb{P}\log_2 \mathbb{P} - (1-\mathbb{P})\log_2(1-\mathbb{P})$

prob. clase 1   prob. clase 0

Ejemplo: $\mathcal{D}_{train} = \{(x_n, c_n)\}_{n=1}^{N_{train}} \quad c_n \in \{0,1\} \quad N_{train} = 120 \quad$ #clase 0 = 60, #clase 1 = 60

G.I := ganancia de información



$\hat{p}_0 = 1$, $\hat{p}_1 = 0$ $\qquad \hat{p}_0 = \frac{1}{4}$, $\hat{p}_1 = \frac{3}{4}$ $\qquad \hat{p}_0 = \frac{2}{5}$, $\hat{p}_1 = \frac{3}{5}$

$H(0) = -0\log_2 0 - 1\log_2 1 = 0$ bits $\qquad H\left(\frac{1}{4}\right) = 0'81$ bits $\qquad H\left(\frac{2}{5}\right) = H\left(\frac{3}{5}\right) = \dots$

Árboles de decisión (ID3)

C4.5 ó C5.0

CART ← criterio de Gini (otra entropía)

VALIDACIÓN PARA SELECCIÓN DE → ARQUITECTURA / → MODELO

$\boxed{D_{test}}$

D_{train}: D_I / D_{II} / D_{III}

Entrenamiento	Validación
$D_{II} \cup D_{III}$	$\text{Error}(D_I)$

- VALIDACIÓN SIMPLE: selecciona el modelo que tenga el menor error en el conjunto de validación.

$$\underset{\text{MODELO}}{\text{argmin}}\ \text{Error}_{\text{MODELO}}(D_I)$$

Entrena con $D_I \cup D_{II} \cup D_{III}$ (e.d. D_{train})

- VALIDACIÓN CRUZADA DE 3 BLOQUES (3-FOLD CROSS VALIDATION)

$$\underset{\text{MODELO}}{\text{argmin}}\ \text{Error}^{3\text{-CV}}_{\text{MODELO}}$$

$$\text{Err}^{3\text{-CV}}_{\text{MODELO}} = \frac{\text{Err}_{\text{MODELO}}(D_I) + \text{Err}_{\text{MODELO}}(D_{II}) + \text{Err}_{\text{MODELO}}(D_{III})}{3}$$

Entr	Val
$D_{II} \cup D_{III}$	$\text{Err}(D_I)$
$D_I \cup D_{II}$	$\text{Err}(D_{III})$
$D_I \cup D_{III}$	$\text{Err}(D_{II})$

MODELO	ARQUITECTURA	HIPERPARÁMETROS	PARÁM.
NN	# capas ocultas # neuronas en capa oculta	• Cte. aprendizaje • Cte. regularización • f. de activación	ω
K-NN	—	• Función distancia • K • pesos para votos	—
ÁRBOLES DE DECISIÓN	ID3 / C4.5 / CART	• Poda • # mínimo ejemplos para división	Divisiones en nodos interiores
SVM	—	• Kernel • Anchura del Kernel	ω

APRENDIZAJE POR MÁXIMA VEROSIMILITUD

$$\mathbb{P}(C_1|\underset{\sim}{x}, \underset{\sim}{w}) = h(\underset{\sim}{x})$$

Datos: $\{(x_n, t_n) \; ; \; n = 1, \dots, N_{train}\}$ $\qquad t_n = \begin{cases} 1 & \text{si } C_n = C_1 \\ 0 & \text{si } C_n = C_2 \end{cases}$

Verosimilitud:

$$\mathcal{L}\left(\underset{\sim}{w} \; ; \; \{(x_n, t_n)\}_{n=1}^{N_{train}}\right) \equiv \mathbb{P}\left(\{t_n\}_{n=1}^{N_{train}} \,\middle|\, \{x_n\}_{n=1}^{N_{train}} ; \underset{\sim}{w}\right) =$$

$$= \prod_{n=1}^{N_{train}} \left[\mathbb{P}(C_1|\underset{\sim}{x_n}, \underset{\sim}{w})\right]^{t_n} \cdot \left[1 - \mathbb{P}(C_1|\underset{\sim}{x_n}, \underset{\sim}{w})\right]^{(1-t_n)} =$$

$$= \prod_{n=1}^{N_{train}} \left[h(\underset{\sim}{x_n})\right]^{t_n} \cdot \left[1 - h(\underset{\sim}{x_n})\right]^{(1-t_n)}$$

Función de error

$$E(\underset{\sim}{w}) = -\log \mathcal{L}\left(\underset{\sim}{w} \; ; \{(x_n, t_n)\}_{n=1}^{N_{train}}\right) = \sum_{n=1}^{N_{train}} E_n(\underset{\sim}{w})$$

donde $E_n(\underset{\sim}{w}) \equiv -t_n \log h(\underset{\sim}{x_n}) - (1-t_n) \log(1 - h(\underset{\sim}{x_n}))$

Aprendizaje ML = minimizar $E(\underset{\sim}{w})$

"embeddings" en espacios de Hilbert con núcleo reproductor