Aux 6. Introducción a la Minería de Datos

Gastón L'Huillier[1,2], Richard Weber[2]
glhuilli@dcc.uchile.cl

[1]Departamento de Ciencias de la Computación
Universidad de Chile

[2]Departamento de Ingeniería Industrial
Universidad de Chile

2010

Auxiliar 6

Árboles de Decisión

- Tipos de árboles de decisión
- Splits: Entropía, Ganancia de Información e índice de Gini
- Poda: Pre-poda y Post-poda
- Propiedades
- Uso en Rapidminer v5.0

Redes Neuronales

- Perceptron,
- Red Neuronal Perceptrón multi-capa

Recuerdo: Aprendizaje Supervisado

Dataset

$$\mathcal{X} = \begin{pmatrix} x_{1,1} & \dots & x_{1,A} \\ x_{2,1} & \dots & x_{2,A} \\ \vdots & \vdots & \vdots \\ x_{N,1} & \dots & x_{N,A} \end{pmatrix}, \mathcal{Y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$$\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{N}\ ,\ \mathbf{x_i} = (x_{i,1}, \dots, x_{i,A})\ ,\ \forall i \in \{1, \dots, N\}$$

$$\mathcal{Y} = (y_1, \dots, y_N)^T$$

Probabilidad que cliente$_j$ pague el credito

$$y = f(\mathcal{X})$$

$$\Rightarrow y_i = P(\text{ pague el credito } |\mathbf{x}_i)$$

Técnicas Aprendizaje Supervisado

- **Arboles de Decisión** (aux 6)
- **Perceptrón** (aux 6)
- **Redes Neuronales (Perceptrón Multi-capa)** (aux 6)
- Redes Bayesianas
- **Naïve Bayes** (+ adelante)
- **Support Vector Machines** (+ adelante)
- **Mutliples-clasificadores** (Boosting, Bagging, etc.) (+ adelante)
- Regressión logística
- Nearest Neighbor
- ...!

Árboles de Decisión

Representación

- Cada nodo interno es un atributo
- Cada rama corresponde a un valor del atributo
- Cada nodo hoja asigna una clasificación

Cuando deberíamos considerar esta técnica

- Cuando los objetos se pueden describir por variables binarias
- Variable objetivo toma valores discretos
- Datos de entrenamiento con potencial ruido
- e.g. Diagnósis médicas, análisis de crédito, etc.

Árboles de Decisión

Algoritmo ID3 (Iterative Dichotomiser 3)

- [Quinlan, 1986].
- Solamente permite clasificar base de datos con atributos categóricos.

Algoritmo C4.5

- [Quinlan, 1993]
- Básicamente el mismo algoritmo ID3, pero considera la posibilidad de clasificar con atributos con valores continuos.
- Se generan rangos que permiten manejar los valores continuos como valores categóricos.

Árboles de Decisión

CART

- “Classification and Regression Tree”
- [Breiman et al., 1984]
- Ramificaciones de los atributos (splits) binarias.

CHAID

- [Hartigan, 1975]
- Basado en el algoritmo “Automatic Interaction Detection” (AID) publicado el año 1963
- Chi-square Automatic Interaction Detection.

Árboles de decisión: La idea...

Age	Income	Student	Credit	Buys
$\leq$ 30	High	No	Fair	No
$\leq$ 30	High	No	Excellent	Yes
31 . . . 40	High	No	Fair	Yes
$>$ 40	Medium	No	Fair	Yes
$>$ 40	Low	Yes	Fair	Yes
$>$ 40	Low	Yes	Excellent	No
31 . . . 40	Low	Yes	Excellent	Yes
$\leq$ 30	Medium	No	Fair	No
$\leq$ 30	Low	Yes	Fair	No
$>$ 40	Medium	Yes	Fair	Yes
$\leq$ 30	Medium	Yes	Excellent	Yes
31 . . . 40	Medium	No	Excellent	Yes
31 . . . 40	High	Yes	Fair	Yes
$>$ 40	Medium	No	Excellent	No

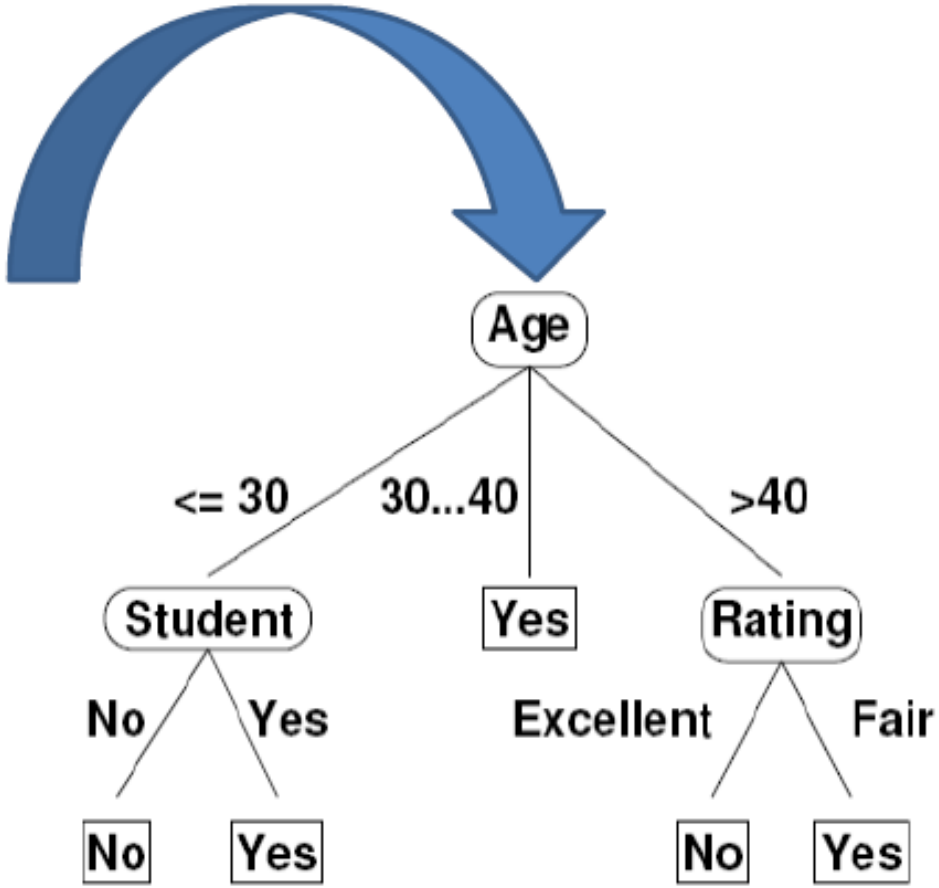


Figura: Idea principal de los árboles de decisión

Algoritmo ID3

ID3(Muestas, Atributos, AtributoObjetivo) /* AtributoObjetivo = {C_1 , …, C_k }
*/
{
Se crea un nodo **Raiz** para el árbol
IF todas los muestras son de tipo C_i,
Retornar nodo **Raiz**, con etiqueta C_i
IF la cantidad de atributos restantes es vacía,
Retornar nodo **Raiz** con etiqueta
= valor más común de los AtributoObjetivo en la muestra /*mayoría de votos*/
ELSE se inicia,
A = Atributo que mejor clasifica a los datos. /*Mejor clasifica → Ganancia de Información*/
Raiz = **A**.
foreach s_i, de **A**, /* s_i , i = 1,..n, con n = numero de ***splits*** posibles*/
Crear una nueva rama bajo la raíz,
E (s_i) = subconjunto de la muestra que representan el valor s_i de **A**
IF **E**(s_i) es vacío
Bajo la rama se agrega un nodo con etiqueta
= los valores del AtributoObjetivos más común en la muestra
ELSE bajo la rama se agrega el sub-árbol **ID3(E(s_i), Atributos – {A}, AtributoObjetivo)**
FIN
RETURN **Raiz**
}

Figura: Algoritmo ID3

Algoritmo ID3

ID3(Muestas, Atributos, AtributoObjetivo) /* AtributoObjetivo = {C_1 , …, C_k }
*/
{
Se crea un nodo **Raiz** para el árbol
IF todas los muestras son de tipo C_i,
Retornar nodo **Raiz**, con etiqueta C_i
IF la cantidad de atributos restantes es vacía,
Retornar nodo **Raiz** con etiqueta
= valor más común de los AtributoObjetivo en la muestra /*mayoría de votos*/
ELSE se inicia,
A = Atributo que mejor clasifica a los datos. /*Mejor clasifica → Ganancia de Información*/
Raiz = **A**.
foreach s_i, de **A**, /* s_i , i = 1,..n, con n = numero de ***splits*** posibles*/
Crear una nueva rama bajo la raíz,
E (s_i) = subconjunto de la muestra que representan el valor s_i de **A**
IF **E**(s_i) es vacío
Bajo la rama se agrega un nodo con etiqueta
= los valores del AtributoObjetivos más común en la muestra
ELSE bajo la rama se agrega el sub-árbol **ID3(E(s_i), Atributos – {A}, AtributoObjetivo)**
FIN
RETURN **Raiz**
}

Figura: “Atributo que mejor clasifica” ⇒ Mejor “Split”

Evaluación de los Splits

- Lo ideal es encontrar los splits que generen nuevas ramas cuyos nodos tengan sean lo más limpios posible (o menor impureza posible)
- Ganancia de Información Ratio (porción)
- Test Chi^2
- Entropía (Ganancia de Información)
- Índice de Gini

Gini Split

Indice de Gini

$$\text{Gini}(n) = 1 - \sum_{x \in \mathcal{X}} P(n,x)^2$$

- $P(n,x) =$ Probabilidad de ocurrencia de la clase x en el nodo n.
- Gini(n) es la probabilidad de no sacar dos registros de la misma clase del mismo nodo.
- Mientras menor el índice de Gini, mejor es la pureza del split.

Gini Split

Dado un split $\mathcal{S}_n = \{S_1, \ldots, S_k\}$ del nodo n, donde $|n|$ es la cardinalidad de elementos en el nodo n.

$$\text{GiniSplit}(n, \mathcal{S}) = \sum_{s \in \mathcal{S}_n} \frac{|s|}{|n|} \text{Gini}(s)$$

$\Rightarrow$ Seleccionamos para ramificar, aquel split con menor GiniSplit.

Ejemplo Gini Split

Outlook	Temp.	Humidity	Windy	Play
Sunny	Hot	High	FALSE	No
Sunny	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Rainy	Mild	High	FALSE	Yes
Rainy	Cool	Normal	FALSE	Yes
Rainy	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Sunny	Mild	High	FALSE	No
Sunny	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	FALSE	Yes
Sunny	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Mild	High	TRUE	No

Figura: Base de datos “Weather”

Gini Split

Ejemplo

$$\text{Gini}(\text{sunny}) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0{,}48$$

$$\text{Gini}(\text{overcast}) = 1 - \left(\frac{4}{4}\right)^2 - \left(\frac{0}{4}\right)^2 = 0{,}0$$

$$\text{Gini}(\text{rainy}) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0{,}48$$

$$\text{GiniSplit}(\text{Outlook}) = \left(\frac{5}{14}\right) \times 0{,}48 + \left(\frac{4}{14}\right) \times 0{,}0 + \left(\frac{5}{14}\right) \times 0{,}48 = 0{,}343$$

Entropía Split

Entropía

$$\text{Entropia}(n) = -\sum_{x \in \mathcal{X}} P(n,x) log_2 \left(P(n,x)^2 \right)$$

- La entropía mide la impureza de los datos S (mide la información (número de bits) promedio necesaria para codificar las clases de los datos en el nodo)
- El criterio de selección con la mayor ganancia de información (Gain)

Gain Split

Dado un split $\mathcal{S}_n = \{S_1, \ldots, S_k\}$ del nodo n, donde $|n|$ es la cardinalidad de elementos en el nodo n.

$$\text{Gain}(n, \mathcal{S}) = \text{Entropia}(n) - \sum_{s \in \mathcal{S}_n} \frac{|s|}{|n|} \text{Entropia}(s)$$

$\Rightarrow$ Seleccionamos para ramificar, aquel split con menor GiniSplit.

Gain Split

Ejemplo

$$\text{Entropy}(\text{Outlook}) = -\left(\frac{9}{14}\right) log_2 \left(\frac{9}{14}\right) - \left(\frac{5}{14}\right) log_2 \left(\frac{5}{14}\right) = 0{,}94$$

$$\text{Entropy}(\text{sunny}) = -\left(\frac{2}{5}\right) log_2 \left(\frac{2}{5}\right) - \left(\frac{3}{5}\right) log_2 \left(\frac{3}{5}\right) = 0{,}97$$

$$\text{Entropy}(\text{overcast}) = -\left(\frac{4}{4}\right) log_2 \left(\frac{4}{4}\right) - \left(\frac{0}{4}\right) log_2 \left(\frac{0}{4}\right) = 0{,}0$$

$$\text{Entropy}(\text{rainy}) = -\left(\frac{3}{5}\right) log_2 \left(\frac{3}{5}\right) - \left(\frac{2}{5}\right) log_2 \left(\frac{2}{5}\right) = 0{,}97$$

$$\text{Gain}(\text{Outlook}) = 0{,}94 - \left(\left(\frac{5}{14}\right) \times 0{,}97 + \left(\frac{4}{14}\right) \times 0{,}0 + \left(\frac{5}{14}\right) \times 0{,}97\right) = 0{,}25$$

Sobre-Ajuste del Modelo

Pre-poda

- Parar la construcción de un árbol antes que se termine de construir.s
- Entropía
- Índice de Gini, Twoing
- TestChi^2
- No expandir según algún criterio:
- Detener el crecimiento del árbol dado un número mínimo de muestras presentes en un nodo.

Post-poda

- Remover las ramas de un árbol de decisión completo.
- Se debe utilizar un conjunto de datos distinto que el de entrenamiento para decidir cual es el mejor “Árbol podado”. Al plantear el problema como un modelo de optimización, por el momento solo se deben considerar Heurísticas dado que es un problema **NP-hard**.

En resumen... [1]

- Son modelos de tipo supervisado **fáciles de entender e interpretar**. Es un modelo transparente, en el cual se pueden seguir cada clasificación obtenida mediante las reglas lógicas.

- Pueden clasificar utilizando tanto **datos categóricos como contínuos**. No existe una restricción de los tipos de datos necesarios para poder construir el árbol de decisión.

- Son **fácilmente convertidos en un conjunto de reglas lógicas**, útiles para implementar al momento que se necesitan reutilizar las reglas obtenidas para un determinado árbol de decisión.

En resumen... [2]

- Se pueden generar las reglas lógicas obtenidas en **código SQL**, de manera de integrarlas en los sistemas de bases de datos de una empresa. A diferencia de otros modelos, permite una **fácil integración en sistemas de información** de la empresa.

- Permiten identificar aquellos atributos que permiten entregar una mayor cantidad de información al modelo. **"Embedded Feature Selection"**.

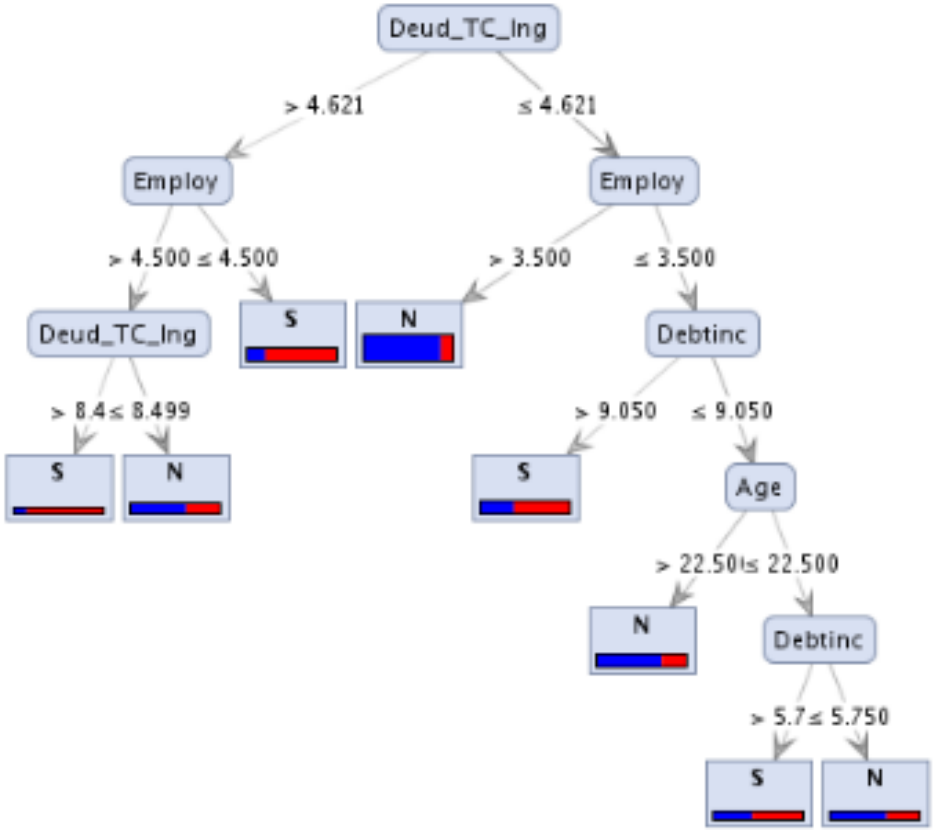
- Es necesaria una **cantidad representativa de datos** del conjunto que se desea modelar, para poder entrenar un árbol de decisión con significancia estadística. Si se tiene una muestra que se desea modelar para clasificar en tres categorías, donde dos categorías tienen un porcentaje sumamente mayor de datos al de la tercera categoría (47 %, 50 % y 3 % respectivamente), se tienen problemas para que el árbol pueda generar las reglas lógicas que permitan discriminar la tercera categoría.

En resumen... [3]

- Es necesario tener presente el sobre ajuste y podar (pre y post poda) un árbol de decisión. Distintos valores para los **parámetros relacionados con la post-poda**, pueden generar distintas alturas en los árboles de decisión, pero **no reordenar el orden de las reglas lógicas**, y la jerarquía de las variables presentes en los nodos del árbol.

- Dependiendo de los índices de medidas de ganancia de información **(criterios de pre-poda)**, se pueden obtener árboles de decisión completamente distintos, **reordenando el orden de la jerarquía de las variables** presentes en los nodos del árbol.

Arbol de Decisión en RapidMiner v5.0



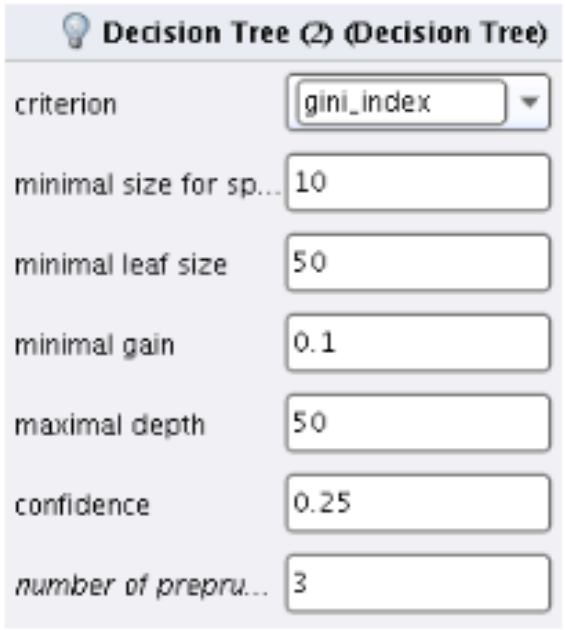


Figura: Árboles en RapidMiner (v5.0)

Auxiliare 6

Árboles de Decisión

- Tipos de árboles de decisión
- Splits: Entropía, Ganancia de Información e índice de Gini
- Poda: Pre-poda y Post-poda
- Propiedades
- Uso en Rapidminer v5.0

Redes Neuronales

- Perceptron
- Red Neuronal Perceptrón multi-capa

Conceptos generales

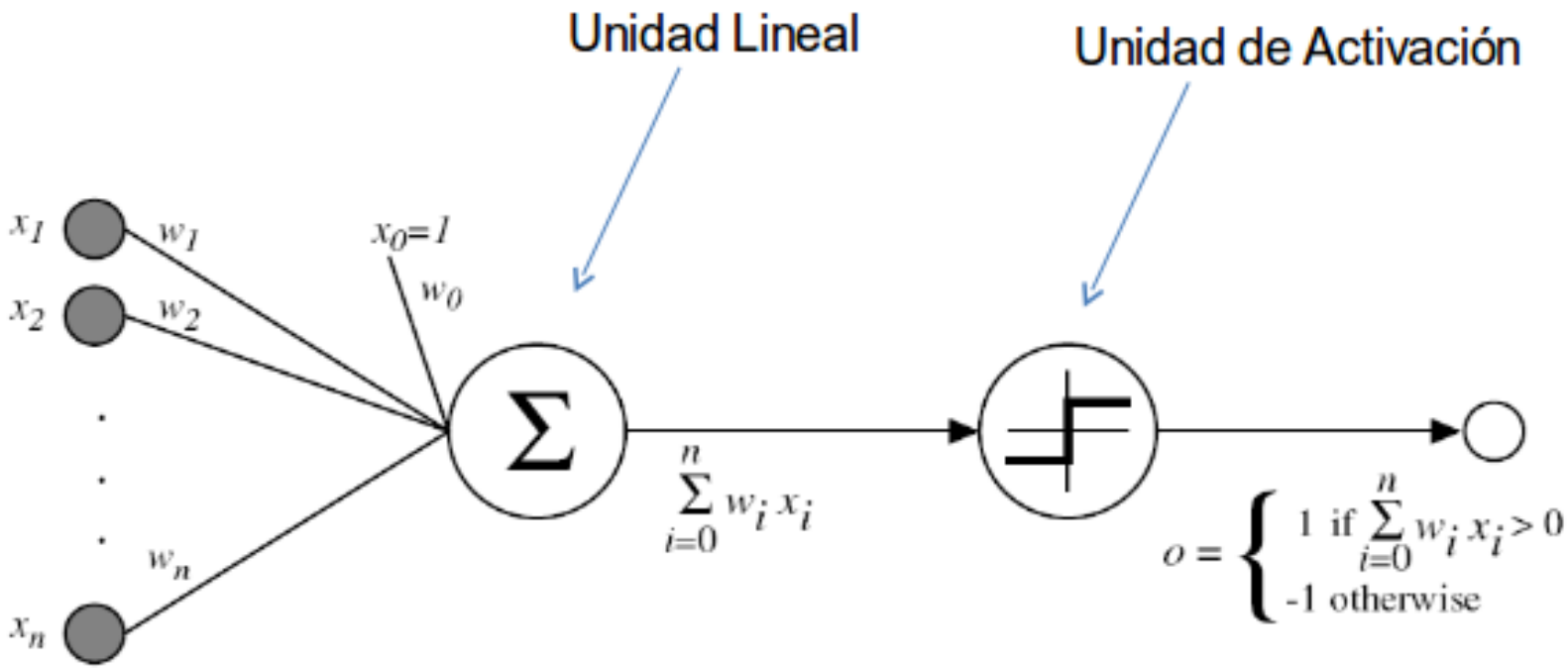
- Método de **regresión y clasificación** de aprendizaje supervisado.
- Las redes neuronales pertenecen al conjunto de herramientas de clasificación y regresión **no lineal**.
- Se ha demostrado que es un “aproximador” universal: **Cualquier función continua se puede aproximar por una red neuronal** (en particular utilizando un perceptrón multicapa MLP).
- La habilidad de la red neuronal de **aprender a partir de un conjunto de ejemplos** es un modelo adecuado para abordar un gran número de problemas dado que puede aproximar funciones no lineales, filtrar ruido en los datos, entre otras aplicaciones.

Orígen de las Redes Neuronales

Perceptrón (Rosemblatt, 1958)

- Modelo de clasificación que **define un límite de decisión** (hiperplano) para clases $y \in \{+1, -1\}$.
- Puede representar cualquier límite de **decisión lineal**.
- Sólo permite **entradas binarias**.
- Sólo puede clasificar datos que sean **linealmente separables**, ya que el algoritmo mediante el cuál se entrena el modelo sólo converge si se tiene tal propiedad (teorema de convergencia del perceptrón).

Perceptrón [1]



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Figura: Estructura general del perceptron.

Perceptrón [2]

Función AND

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

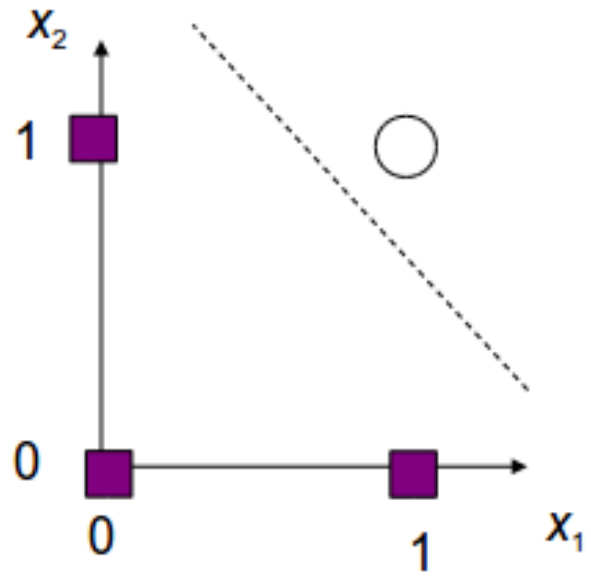


Figura: Función lógica AND.

Perceptrón [3]

Función OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

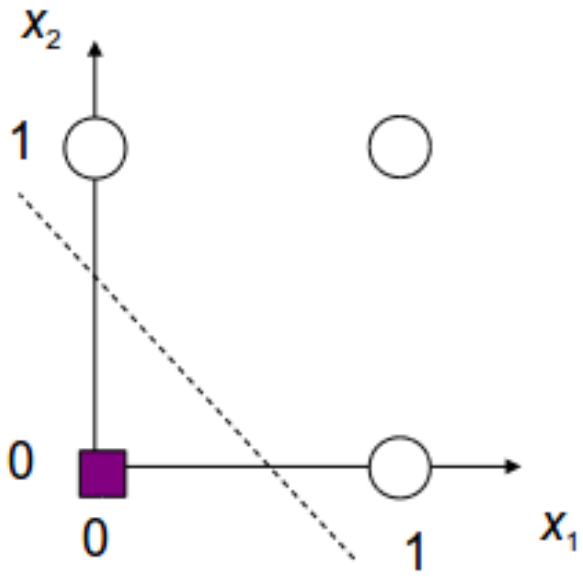


Figura: Función lógica OR.

Perceptrón [4]

Función XOR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

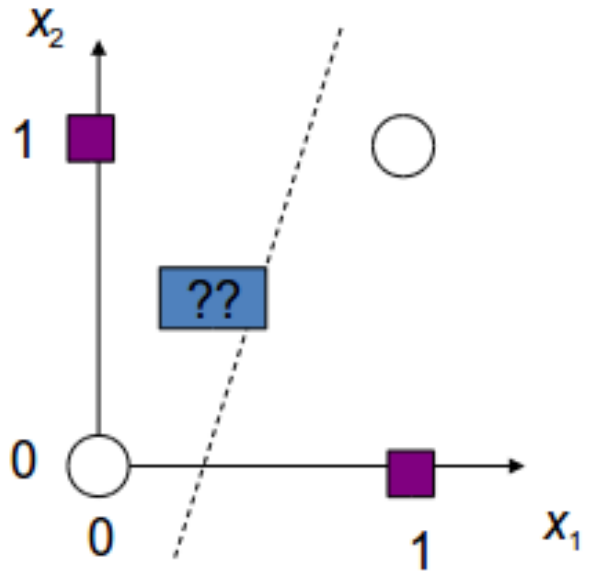


Figura: Función lógica XOR.

Perceptrón [5]

Perceptrón (Algoritmo)

- Se inicializa el algoritmo con todos los pesos $w = 0$.
- Se detiene cuando se obtiene un error mínimo e, o se alcanza el límite de épocas fijadas inicialmente.
- Sea $\eta > 0$ una tasa de aprendizaje.

$$t \in \{+1, -1\},$$

$$o((b, \mathbf{w}), \mathbf{x}) = b + \mathbf{w} \cdot \mathbf{x} = b + \sum_i w_i x_i$$

$$E(t, o) = \text{máx}(0, 1 - t \cdot o)$$

$$\frac{\partial E(t, o)}{\partial w_i} = -t \cdot x_i$$

Perceptrón [6]

1. initialize $\mathbf{w}$
2. **while** stopping criterion is false
3. **for** each $\Delta w_i \in \Delta \mathbf{w}$
4. $\Delta w_i \leftarrow 0$
5. **for** each $(\mathbf{x}_j, t_j) \in D$
6. **for** each $\Delta w_i \in \Delta \mathbf{w}$
7. $\Delta w_i \leftarrow \Delta w_i - \eta * \partial E(t_j, o(\mathbf{w}, \mathbf{x}_j))/\partial w_i$
8. **for** each $w_i \in \mathbf{w}$
9. $w_i \leftarrow w_i + \Delta w_i$
10. **return** $\mathbf{w}$

Figura: Algoritmo Perceptrón.

Perceptrón [7]

	Inputs							Weights				
	x_1	x_2	x_3	x_4	t	o	E	b	w_1	w_2	w_3	w_4
								0	0	0	0	0
t = 0	0	0	0	1	-1	0	1	-1	0	0	0	-1
	1	1	1	0	1	-1	2	0	1	1	1	-1
	1	1	1	1	1	2	0	0	1	1	1	-1
	0	0	1	1	-1	0	1	-1	1	1	0	-2
...	0	0	0	0	1	-1	2	0	1	1	0	-2
	0	1	0	1	-1	-1	0	0	1	1	0	-2
	1	0	0	0	1	1	0	0	1	1	0	-2
	1	0	1	1	1	-1	2	1	2	1	1	-1
t = 9	0	1	0	0	-1	2	3	0	2	0	1	-1

Figura: Evaluación del perceptrón.

Red Neuronal - Perceptrón Multicapa

Artificial Neural Net MultiLayer Perceptron

- La arquitectura de la red neuronal se caracteriza porque tiene sus neuronas agrupadas en capas de diferentes niveles.
- Cada una de las **capas esta formada por un conjunto de neuronas** y se distinguen entre ellas en 3 niveles de capas distintas:
  - la **capa de entrada**: se encargan de recibir las señales o patrones que proceden del exterior y propagar las señales a todas las otras neuronas de la siguiente capa
  - las **capas ocultas**: son las que tienen la misión de realizar el procesamiento no lineal de los patrones recibidos.
  - la **capa de salida**: actúa como salida de la red, proporcionando al exterior la respuesta de la red, para cada uno de los patrones de entrada.

Red Neuronal - Perceptrón Multicapa

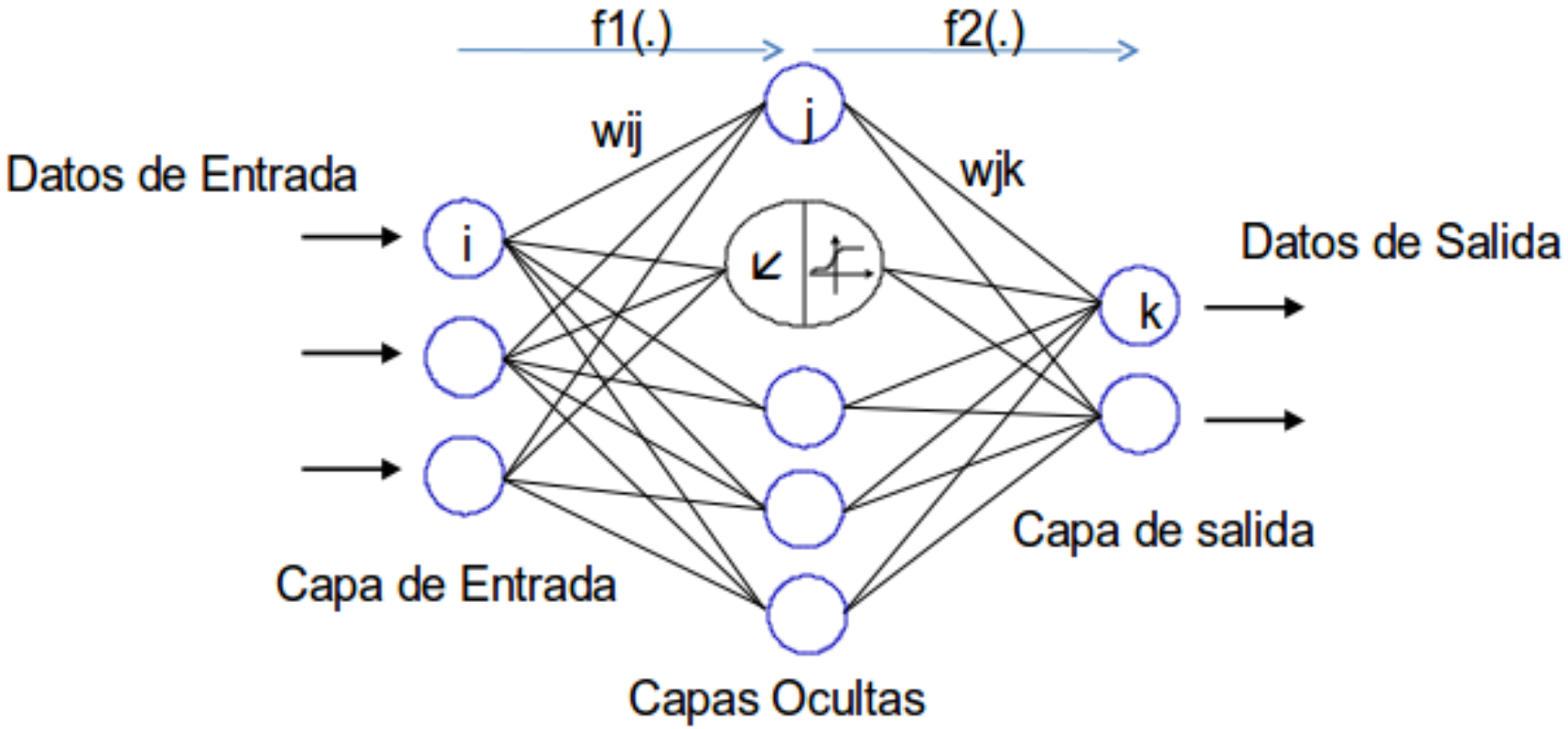


Figura: Diagrama Red Neuronal.

References I


Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984).
Classification and Regression Trees.
Chapman and Hall/CRC, 1 edition.


Hartigan, J. (1975).
Clustering Algorithms.
Wiley, New York.


Quinlan, J. R. (1986).
Induction of decision trees.
Mach. Learn., 1(1):81–106.


Quinlan, J. R. (1993).
C4.5: programs for machine learning.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.