Técnicas de Minería de Datos: Maquinas de Aprendizaje

Jose Aguilar
CEMISID, Escuela de Sistemas
Facultad de Ingeniería
Universidad de Los Andes
Mérida, Venezuela

Técnicas de Aprendizaje Automático:

- Es imposible prever todos los problemas desde el principio
- Un sistema es inteligente si es capaz de observar su entorno y aprender de él
- La autentica inteligencia reside en adaptarse, tener capacidad de integrar nuevo conocimiento, resolver nuevos problemas, aprender de los errores, etc.

Aprendizaje Automático (Machine Learning en inglés) es la rama de la Inteligencia Artificial que tiene como objetivo desarrollar técnicas que permitan a las computadoras aprender.

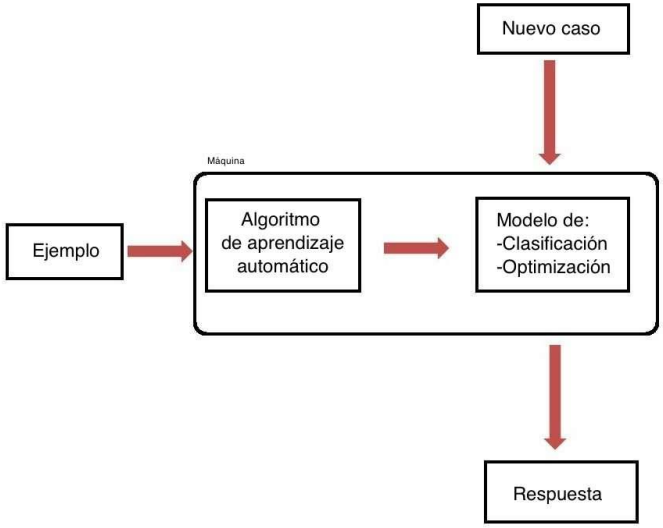
Crear algoritmos capaces de generalizar comportamientos y reconocer patrones.

Dar a los programas la capacidad de adaptarse sin tener que ser reprogramados

Inducción del conocimiento

Técnicas de Aprendizaje Automático:

- Árboles de decisión,
- Reglas de asociación,
- Redes Neuronales Artificiales,
- Tablas de decisión
- Algoritmos Evolutivos
- Y muchos más (algoritmos bio-inspirados, etc.)



Tablas de decisión

Es la forma más simple y más rudimentaria para representar la salida de una máquina de aprendizaje.

Outlook	Temperature	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

Arboles de decisión

Son unos de los algoritmos clasificadores más conocidos y usados en las tareas de Minería, ya que son una forma de representación sencilla para clasificar instancias.

Árboles de decisión son particiones de un conjunto de datos

Objetivo: Segmentar la población para encontrar grupos homogéneos según una cierta variable.

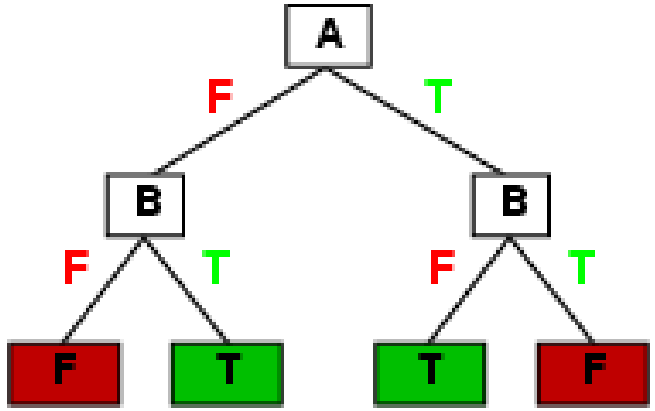
Árbol de Decisión

- Parte de un conjunto de entrenamiento

- Un árbol de decisión es consistente para cualquier conjunto de entrenamiento, cuando hay un camino a una hoja para cada uno de los miembros del conjunto

- Está basado en la idea de tablas de la verdad:

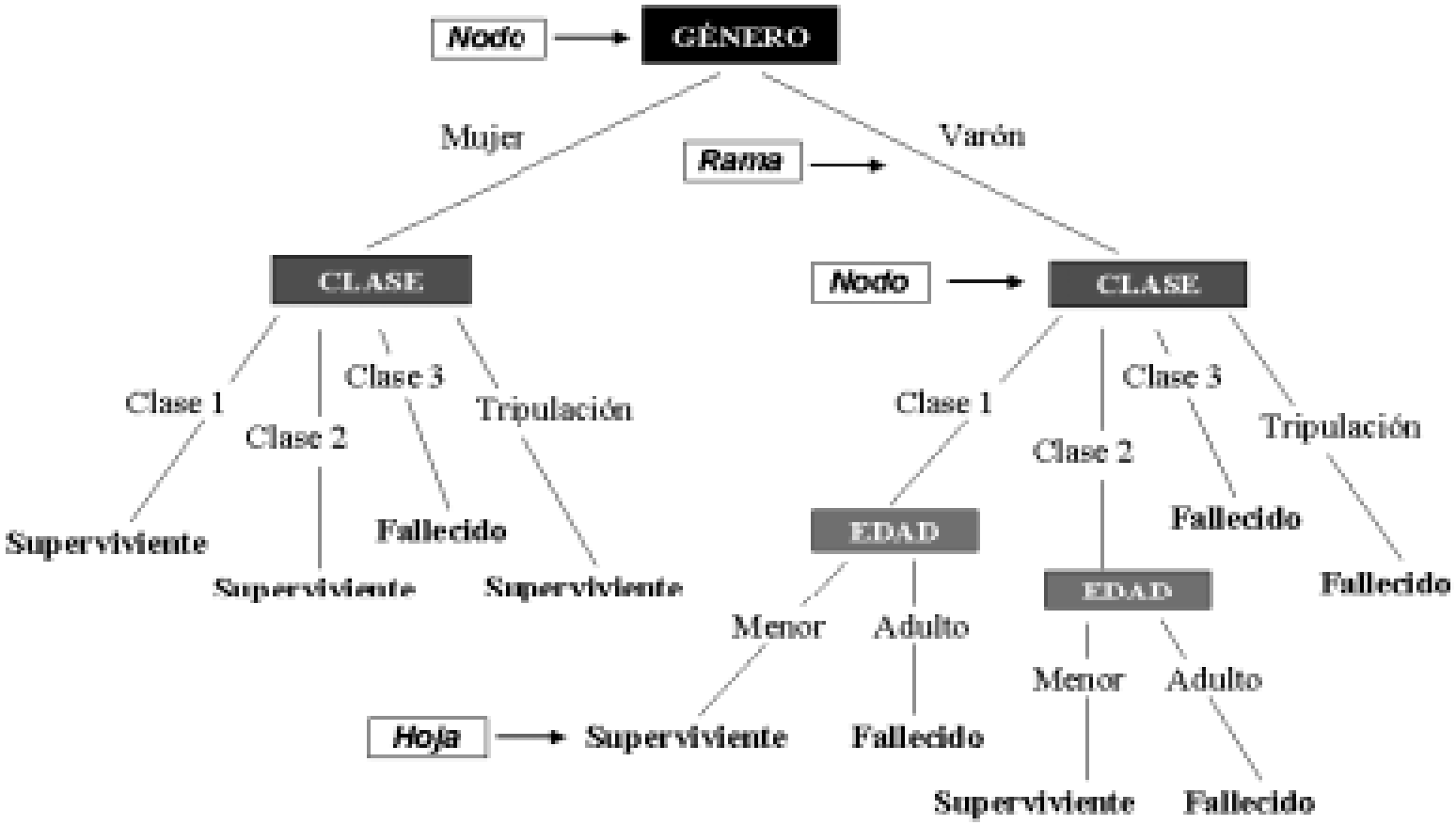
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Es una estrategia de aprendizaje inductivo

ÁRBOLES DE DECISION

Suelen ser empleados en tareas de clasificación, y también, aunque menos, en tareas de predicción



Ej. Acontecimientos relativos al hundimiento del Titanic

Construcción del Árbol de Decisión

- Idea: escoger atributo "más significativo" como raíz del (sub)-árbol

¿Cómo?

- Si hay + y - ejemplos escoger atributo que mejor los divida (mayor discriminante)
- Si hay particiones con + y -, buscar un 2do atributo para seguir partiendo

Macroalgoritmo AD(ejemplos, atributos)

```
Si ejemplos no vacios entonces
    Si ejemplos clasificados entonces
        regresar (clasificación)
    de lo contrario
        mejor: escoger_atributo(atributos, ejemplos)
        arbol: un nuevo árbol de decisión con mejor como raíz
        por cada valor Vi de mejor
            Subejemplos:ejemplos con mejor=Vi
            Subarbol: AD(Subejemplos, atributos)
            Arbol: actualizar(nueva rama con etiqueta Vi y Subarbol)
Regresa(arbol)
```

Ejemplos para construir un AD

Ejemplos

Criterios

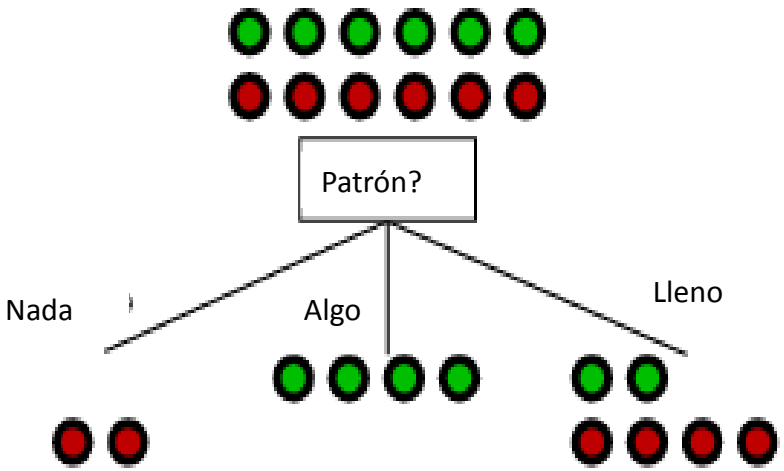
Qué aprendo?

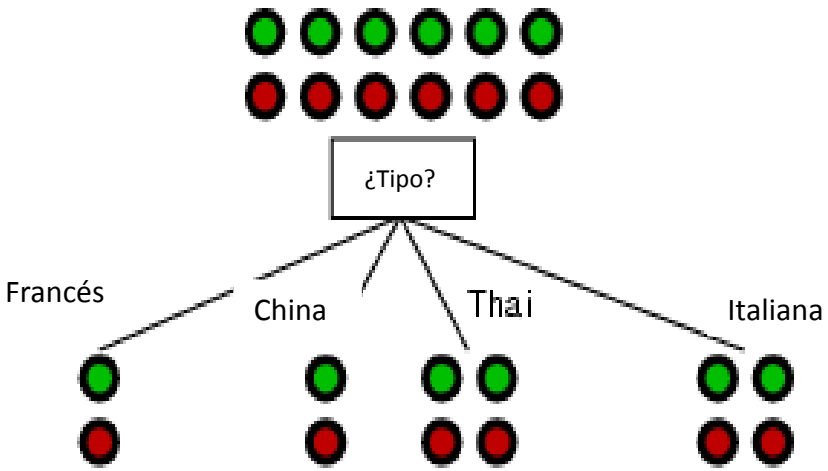
Ej	Alt	Bar	Dia	EdM	Patr	Prec	EdD	Tipo	RES	T --->	Espera
X1	S	N	N	S	Alg	$$$	N	Franc	S	0-10	S
X2	S	N	N	S	llen	$	N	Jap	S	10-15	N
X3	N	S	N	N	Alg	$	N	Hamb	N	0	S
...											
X12	S	S	S	S	llen	$	N	Hamb	N	10	S

Escoger un atributo

"más significativo"

¿*Patrón* es una mejor escogencia que *Tipo*?





Basado en concepto de *contenido de información*

Parte de $$Info(p,n) = -p\log_2(p) - n\log_2(n)$$

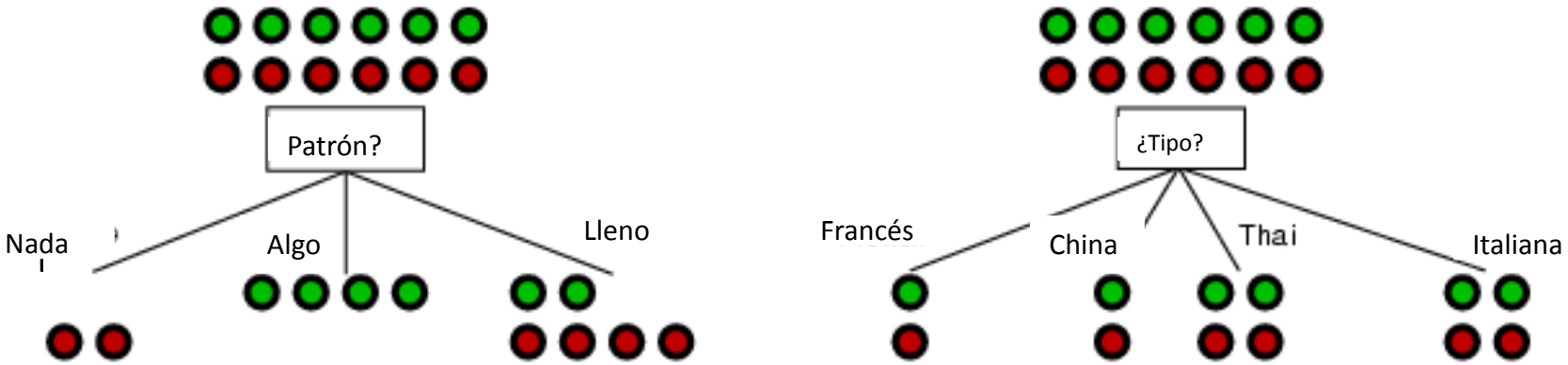
Es una medida de la entropía (grado de desorden) de los ejemplos

n: numero de ejemplos - p: numero de ejemplos +

Escoger un atributo

"más significativo"

¿*Patrón* es una mejor escogencia que *Tipo*?



Donde:

I es entropía de los ejemplos:

y $$IG(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - resto(A)$$

v: posibles valores de A

p_i y n_i? ver siguiente lamina

$$I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

$$\text{resto(A)} = \sum_{i=1}^{v} \left|\frac{p_i - n_i}{p+n}\right| I(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i})$$

Escoger un atributo

"más significativo"

¿Quién es p_i? p_i puede ser $p_i = \frac{\left|E_i^+\right|}{\left|E_i^+\right| + \left|E_i^-\right|}$

Donde E_i^+ es el porcentaje de ejemplos clasificados como + por el valor v del atributo A

Otra Formula general para escoger a los atributos:

Como hay que elegir el atributo con mayor información (menor entropía), otra posibilidad es calcular una función de merito (FM)

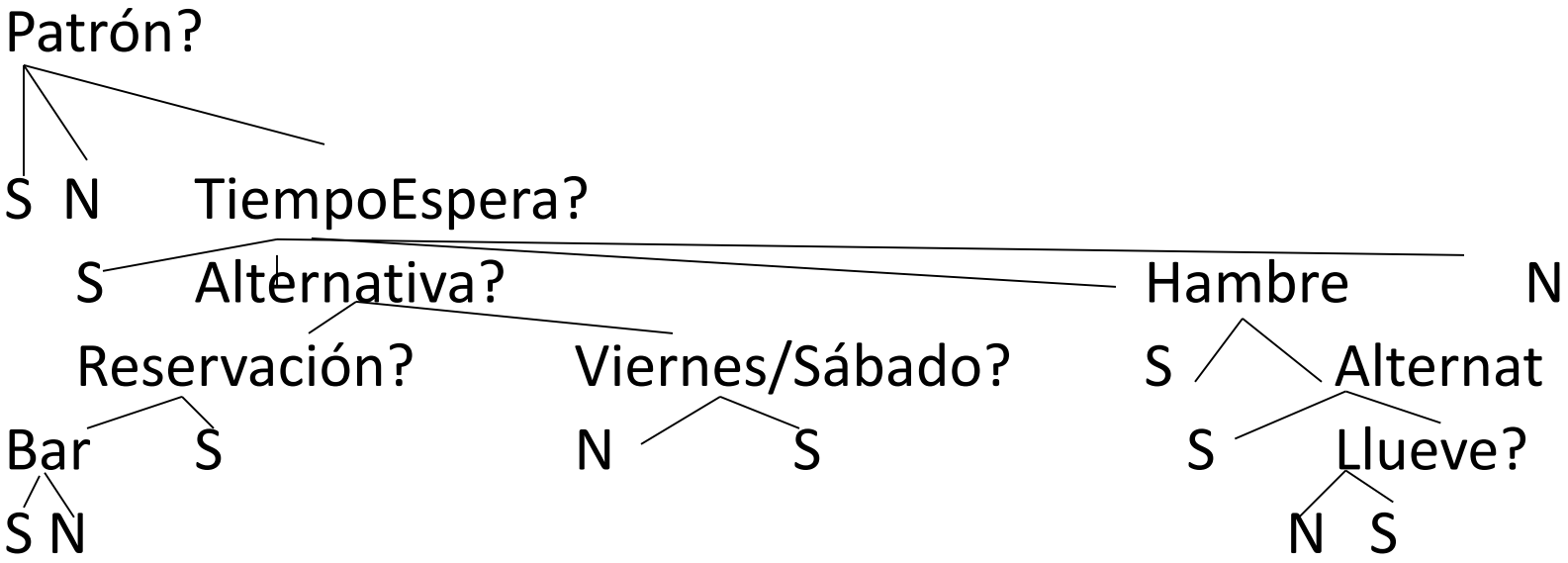
$$\mathrm{FM(A)} = \sum_{\mathrm{i=1}}^{\mathrm{v}} \mathrm{r_i}\ \mathrm{inf\ o(p_i, n_i)}$$

$p_i =$ % ejemplos clasificados como + en la rama i

$$r_i = \left|\frac{p_i - n_i}{p + n}\right|$$

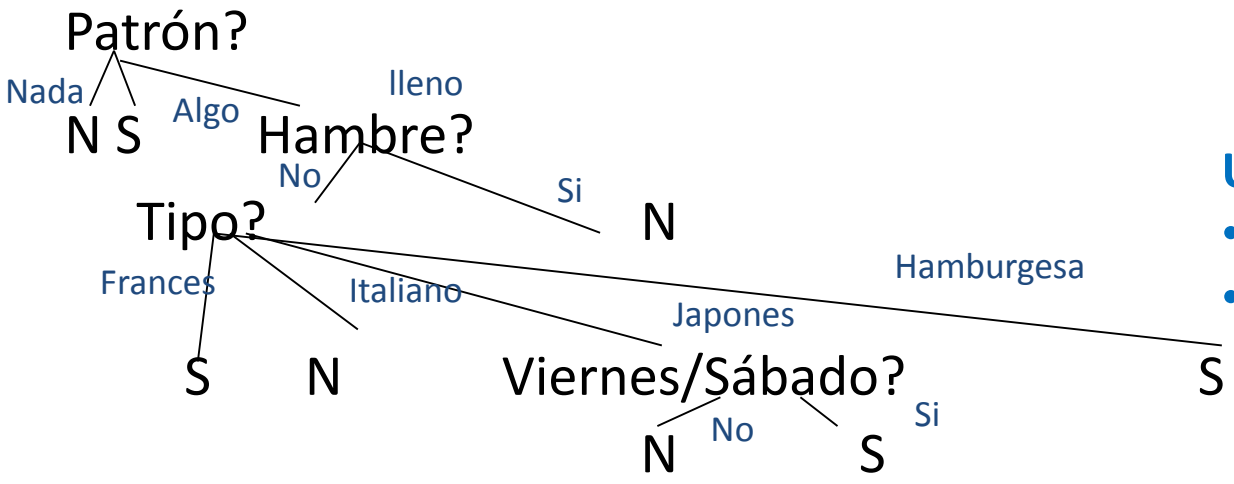
Árbol de Decisión

Para nuestro ejemplo inicial:



Arbol de Decisión y Lógica de Predicado

∀r espera(r) => Patrón(r, algo) O (Patrón(r, full) Y NoHambre(r) Y tipo(r, francés)) O (Patrón(r, full) Y NoHambre(r) Y tipo(r, hamburguesa)) O (Patrón(r, full) Y NoHambre(r) Y tipo(r, Japones) Y viernes/Sabado(r))



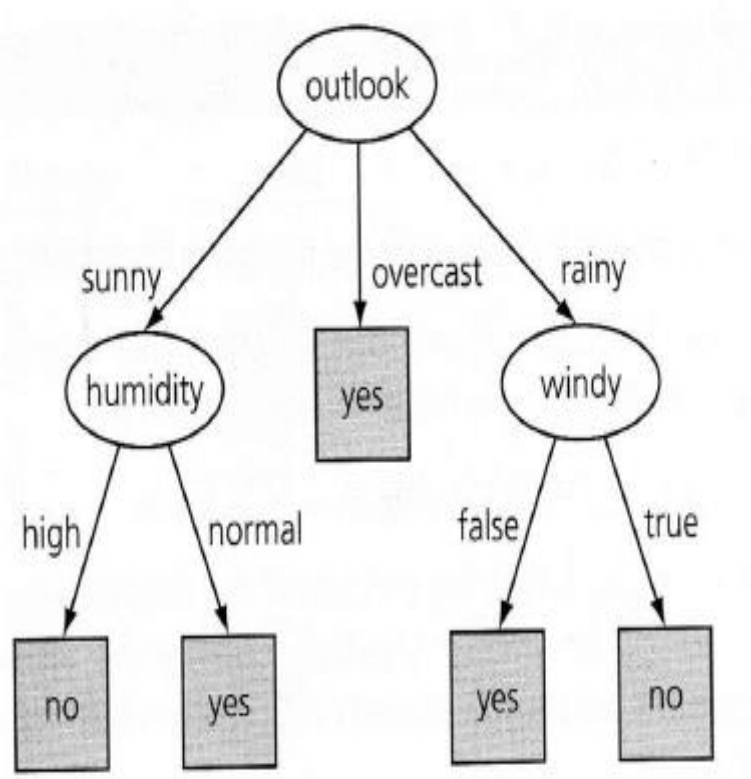
Uso de operadores:

- **Para unir ramas O**
- **Para seguir una rama Y**

Construcción de árboles de decisión

Se completa el árbol hasta cumplir un cierto compromiso:

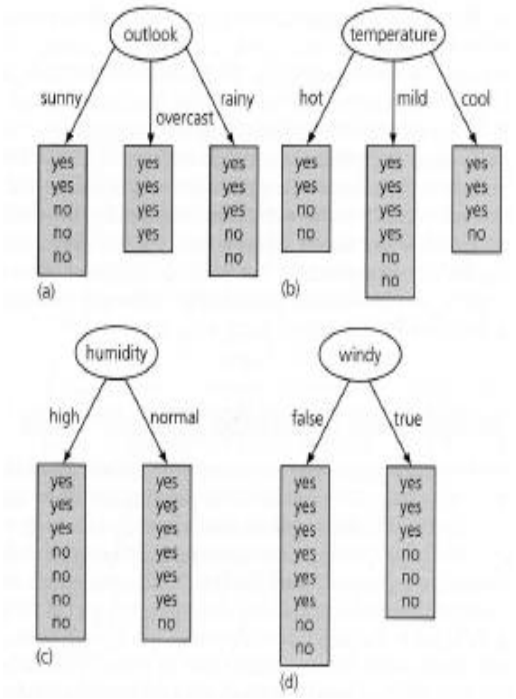
- Número mínimo de elementos en una hoja.
- **Cobertura:** Mínimo número (o porcentaje) de casos posibles cubiertos correctamente de la BD.
- **Precisión:** Error de clasificación menor de un umbral puesto. Por ejemplo: precisión del 80%. Significa, que pararemos cuando el número de clases clasificadas correctamente sea mayor o igual al 80%.



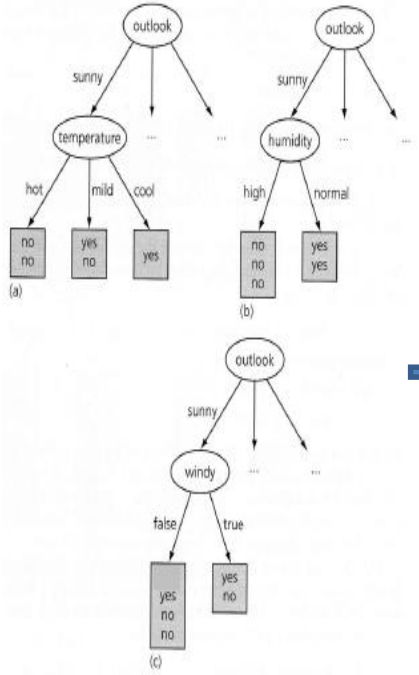
Arboles de decisión:

otros enfoques de construcción

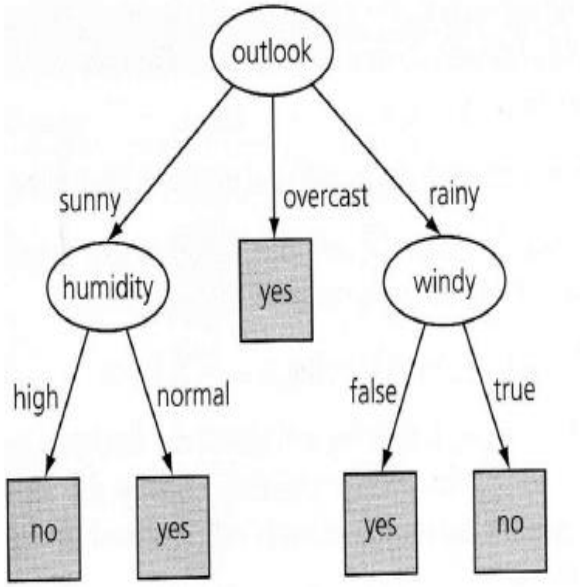
Árbol por atributo



Combinación de un atributo con el resto



Fusión de los Árboles combinados



Podado de un Árbol

¿Cómo decidir si desea reemplazar un nodo interno con una hoja?

Imaginemos que la verdadera probabilidad de error en el nodo es *q*, y que las *N* instancias son generados por un proceso de Bernoulli con parámetro *q*, de la que *E* son los errores.

El intervalo de confianza viene dado por:

$$\Pr\left[\frac{f-q}{\sqrt{q(1-q)/N}} > z\right] = c,$$

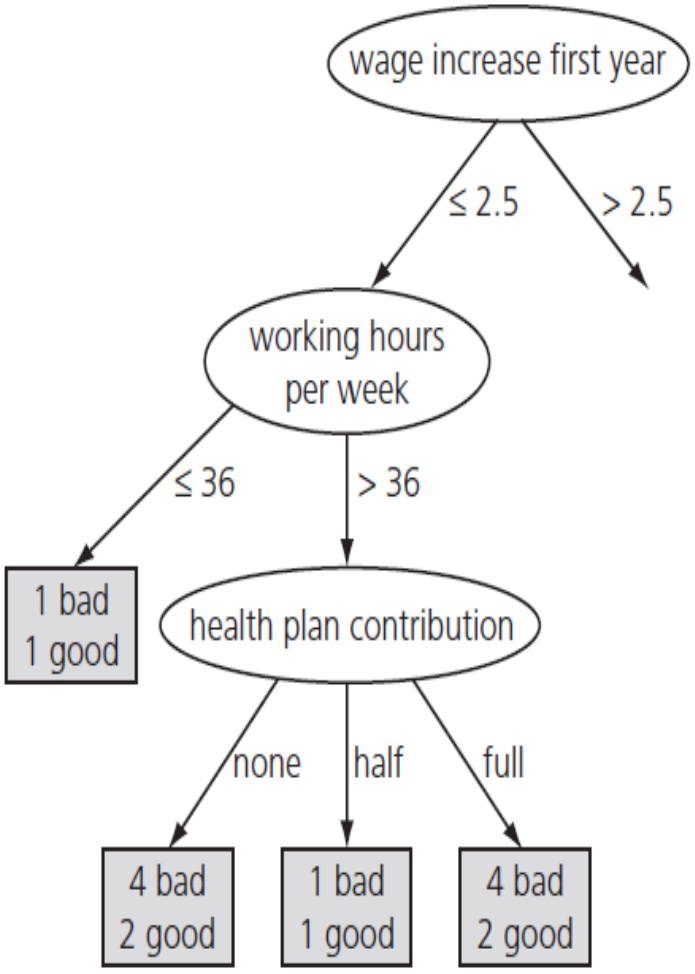
donde *N* es el número de muestras, *f=E/N* es el porcentaje de error observado, y *q* es la tasa de error.

Al igual que antes, esto conduce a un límite superior de confianza para *q*.

Ahora usamos ese límite superior de confianza como una **estimación (pesimista) para la tasa de error *e* en el nodo:**

$$e = \frac{f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}.$$

Podado de un Árbol



None: E = 2, N = 6, y por lo que f = 0,33'. e = 0,47. tasa de error de formación es del 33%, se utilizará la estimación pesimista del 47%.
Half: E = 1, N = 2, e = 0.72.
Full: Tiene el mismo valor de *e* como el primero.

El siguiente paso es combinar las estimaciones de error para estos tres hojas en la relación entre el número de ejemplos que se refieren, 6: 2: 6, lo que conduce a una estimación de error combinado de 0,51.
Health plan contribution: f = 5/14. e = 0.46. Debido a que este es menor que el error de estimación combinada de los tres niños, se podan.

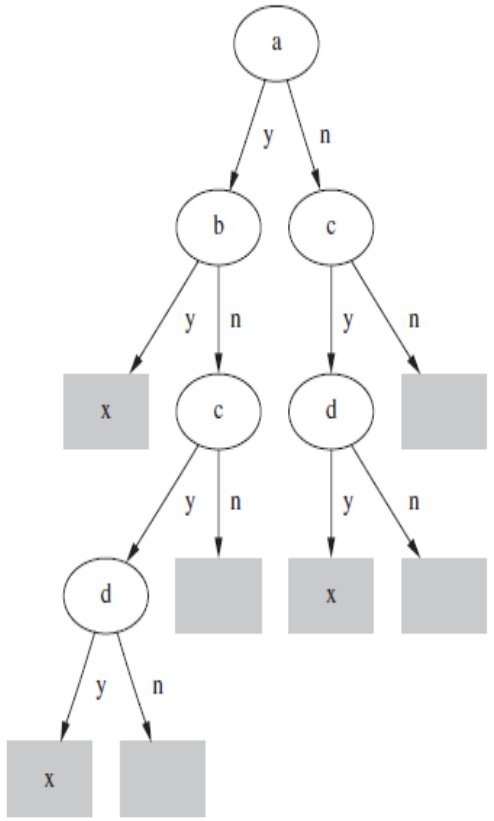
Working hours per week: La estimación de error para la primera, con E = 1 y N = 2, es e = 0,72, y para el segundo es e = 0,46. La combinación de estos, 2 : 14, conduce a un valor que es mayor que la estimación del error para el nodo de horas de trabajo, por lo que el subárbol se poda y se sustituye por un nodo hoja.

Reglas de clasificación

Las reglas de clasificación son una alternativa popular a los árboles de decisión

Por ejemplo:

```
If outlook = sunny and humidity = high then play = no
If outlook = rainy and windy = true    then play = no
If outlook = overcast                  then play = yes
If humidity = normal                   then play = yes
If none of the above                   then play = yes
```



Utilidad de una categoría

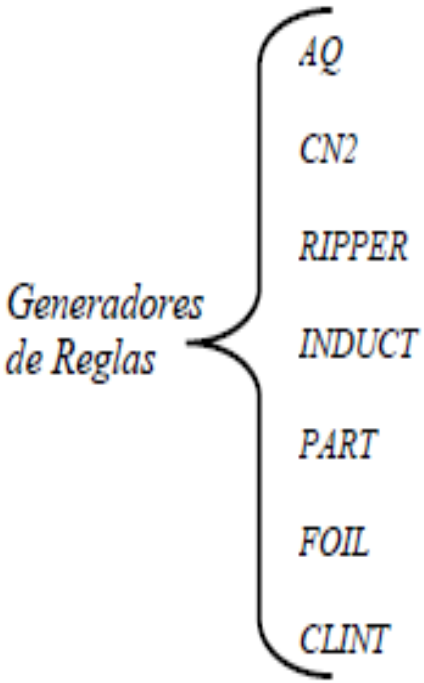
- Mide la calidad general de una partición

$$CU(C_1, C_2, \ldots, C_k) = \frac{\sum_{\ell} \Pr[C_\ell] \sum_i \sum_j \left(\Pr[a_i = v_{ij} | C_\ell]^2 - \Pr[a_i = v_{ij}]^2 \right)}{k}$$

$\Pr[a_i = v_{ij} \mid C_\ell]$ es una estimación de la probabilidad de que el atributo ai tiene un valor vij, en el grupo Cl

donde C_1, C_2,. . ., C_k son los k grupos; la suma exterior es de estos grupos; las siguientes sumas interiores de los atributos a_i , y sus posibles valores v_{i1}, v_{i2},. . . .

Algunos Sistemas de Generación de reglas



- Algunas reglas inducidas pueden derivar de la construcción de un árbol de decisión, siendo primero generado el árbol de decisión y después trasladado a un conjunto de reglas
- Otros algoritmos se basan en el uso de técnicas de aprendizaje con lógica de predicados (ILP, Inductive Logic Programming). (FOIL, FFOIL, CLINT, etc.)

Deducción de reglas rudimentarias

Relation: weather.symbolic

No.	1: outlook Nominal	2: temperature Nominal	3: humidity Nominal	4: windy Nominal	5: **play** Nominal
3	overcast	hot	high	FALSE	yes
7	overcast	cool	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
10	rainy	mild	normal	FALSE	yes
14	rainy	mild	high	TRUE	no
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes

Evaluando los atributos de los datos

	Attribute	Rules	Errors	Total errors
1	outlook	sunny → no	2/5	4/14
		overcast → yes	0/4	
		rainy → yes	2/5	
2	temperature	hot → no*	2/4	5/14
		mild → yes	2/6	
		cool → yes	1/4	
3	humidity	high → no	3/7	4/14
		normal → yes	1/7	
4	windy	false → yes	2/8	5/14
		true → no*	3/6	

Reglas

```
outlook: sunny    → no
         overcast → yes
         rainy    → yes
```

Modelización estadística

Datos de tiempo

Outlook	Temperature	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

probabilidades

Outlook			Temperature			Humidity			Windy			Play	
	yes	*no*		*yes*	*no*		*yes*	*no*		*yes*	*no*	*yes*	*no*
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

Modelización estadística

Table 4.3 A new day.

Outlook	Temperature	Humidity	Windy	Play
sunny	cool	high	true	?

Probabilidad de que sea **si:** 2/9 x 3/9 x 3/9 x 3/9 x 9/14 = 0.0053

Probabilidad de que sea **no**: 3/5 x 1/5 x 4/5 x 3/5 x 5/14 = 0.0206

Normalización

$$\text{Probability of } yes = \frac{0.0053}{0.0053 + 0.0206} = 20.5\%,$$

$$\text{Probability of } no = \frac{0.0206}{0.0053 + 0.0206} = 79.5\%.$$

sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

REGLAS DE ASOCIACION

Técnica no supervisada que permite predecir patrones de comportamientos futuros sobre **ocurrencias simultaneas** de valores de variables.

Una asociación entre dos atributos ocurre cuando la **frecuencia con la que se dan dos o más valores determinados de cada uno conjuntamente es relativamente alta**.

Las reglas de asociación intentan descubrir asociaciones o conexiones entre objetos.

Consecuencia⇐Antecedente 1 Antecedente 2 … Antecedente m.

Ejemplo, en un supermercado se analiza si los pañales y las compotas se compran conjuntamente.

Reglas de Asociación

- Pueden predecir cualquier atributo, o combinaciones de atributos.
- **La *cobertura*** de una regla de asociación es el número de instancias para las cuales ella predice correctamente **(*soporte*)**.
- **La *precisión* (*confianza*)** es el número de instancias que predice correctamente, expresado como una proporción de todas las instancias a las que se aplica.

Outlook	Temperature	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

Se utilizan para descubrir **hechos que ocurren en común** dentro de un determinado conjunto de datos

Por ejemplo, en la tabla anterior las reglas:

- If temperature = cool then humidity = normal
- If windy = false and play = no then outlook = sunny and humidity = high

RNA y Aprendizaje

INTRODUCCIÓN A LAS RNAs

¿CÓMO LA RED NEURONAL HUMANA ESTA DISEÑADA?

¿CÓMO EL CEREBRO PROCESA LA INFORMACIÓN?

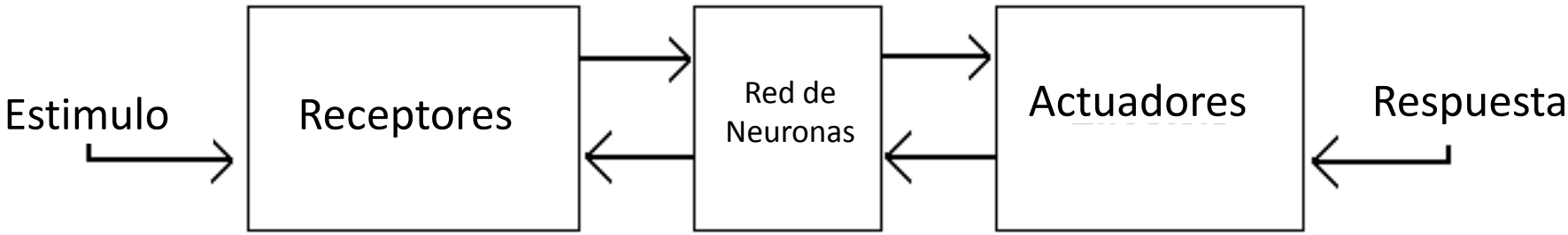
¿CON QUÉ ALGORITMOS Y ARITMÉTICA EL CEREBRO CALCULA?
¿CÓMO PUEDE EL CEREBRO IMAGINAR?

¿CÓMO PUEDE EL CEREBRO INVENTAR?

¿QUÉ ES PENSAR?

¿QUÉ ES SENTIR?

SISTEMA NERVIOSO



MODELO BIOLOGICO

SISTEMA NEURONAL

CONTROL CENTRALIZADO DE LAS FUNCIONES BIOLOGICAS

- **CEREBRO** ~ 100 MIL MILLONES DE **NEURONAS** Y 10000 CONEXIONES POR NEURONA

MODELO BIOLOGICO

- **NEURONAS**: CELULAS VIVAS

- CARACTERISTICAS:
  - ELEMENTOS SIMPLES INTERCONECTADOS
  - FUNCIONAMIENTO EN PARALELO, ASINCRÓNICA Y NO ALGORÍTMICAMENTE
  - INTERACCIONES COMPLEJAS

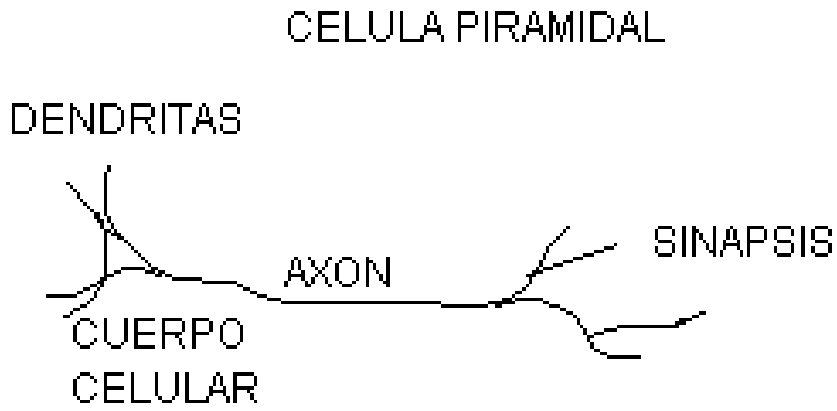
NEURONA

- UNIDAD FUNDAMENTAL DEL SISTEMA NERVIOSO ESPECIALIZADAS EN CIERTAS TAREAS

- PROCESADOR DE SEÑALES ELÉCTRICAS (DESCARGAS EN EL CUERPO CELULAR) Y BIOQUÍMICAS (NEUROTRANSMISORES)

- RECIBE Y COMBINA SEÑALES DESDE MUCHAS NEURONAS

NEURONA

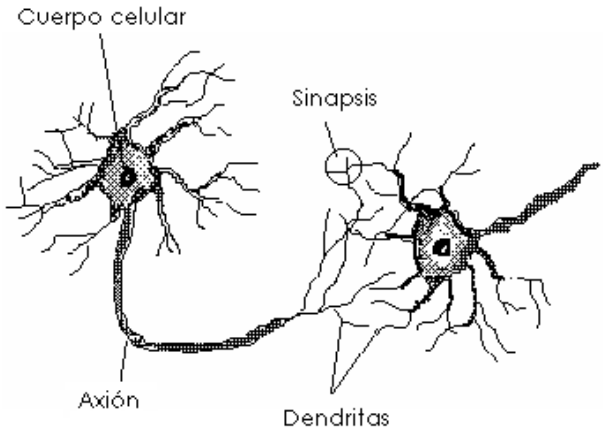


- **AXÓN:** LINEA DE TRANSMISIÓN
- **DENDRITAS:** ZONAS RECEPTORAS
- **SINAPSIS:** EXCITADORAS E INHIBIDORAS
- SEÑALES ELECTRICAS Y QUIMICAS

SINAPSIS

UNIDAD FUNCIONAL QUE INTERRELACIONA LAS NEURONAS

- **NEUROTRANSMISOR:** GENERA POLARIZACIÓN PARA LA MEMBRANA POSTSINÁPTICA
- **POTENCIAL POSTSINÁPTICO:** PUEDE SER POSITIVO (EXCITACIÓN) O NEGATIVO (INHIBICIÓN)



REDES NEURONALES

- MUCHAS **CONEXIONES PARALELAS** ENTRE NEURONAS
- MUCHAS CONEXIONES PROVEEN **MECANISMOS DE RETROALIMENTACIÓN** PARA LAS NEURONAS
- ALGUNAS NEURONAS PUEDEN **EXCITAR** UNAS NEURONAS MIENTRAS **INHIBEN** A OTRAS

REDES NEURONALES

- EJECUTAN UN PROGRAMA QUE ES **DISTRIBUIDO**

- TIENEN PARTES **PRE-HECHAS** Y OTRAS QUE **EVOLUCIONAN**

CAPACIDADES RED NEURONAL

- Procesamiento paralelo
- Adaptativa
- Asociativa
- Auto-organización
- Generalización, clasificación, extracción y optimización

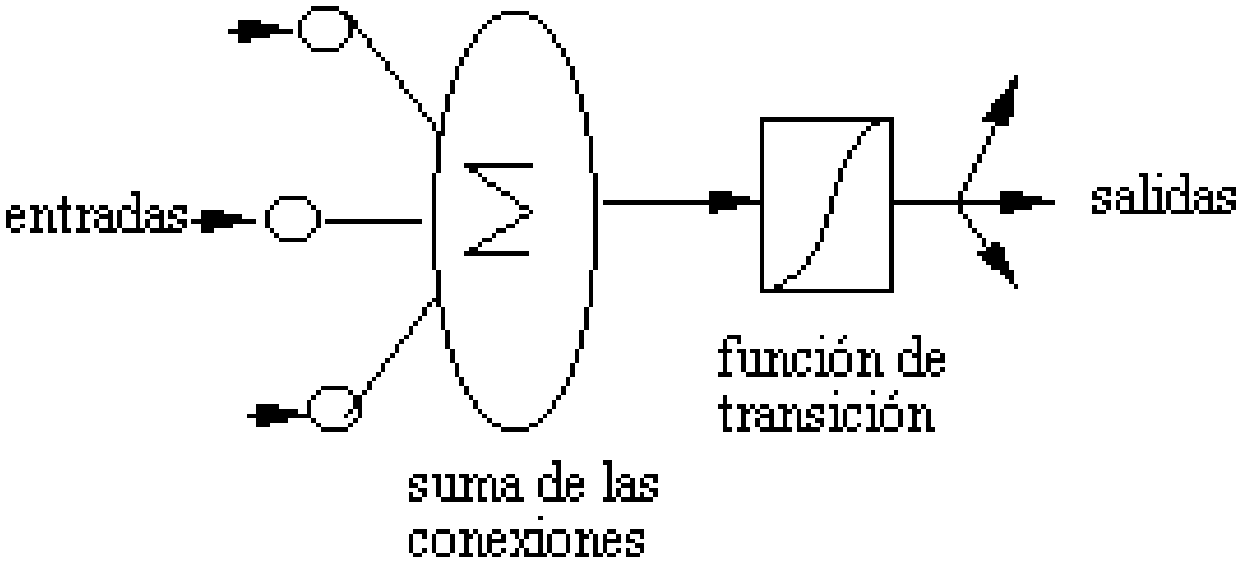
COMPARACION RED NEURONAL

Neurona Biológica	Neurona Artificial
Señales que llegan a la sinapsis	Entradas a la neurona
Carácter excitador o inhibidor de la sinapsis de entrada	Pesos de entrada
Estimulo total de la neurona	Sumatoria de pesos por entradas
Activación o no de la neurona	Función de activación
Respuesta de la neurona	Función de salida

COMPARACION RED NEURONAL

Aspectos	Computador	Cerebro Humano
Unidades de Cálculo	CPUs	10^{11} neuronas
Unidades de Almacenamiento	RAM y disco duro	10^{11} neuronas Y 10^{14} sinapsis
Ciclos	Mherz	10^{-3} segundos
Banda Ancha	Capacidad de transmisión	10^{14} conex. (bits)/segundo
Actualización/seg.	Capacidad de procesamiento paralelo	10^{14}

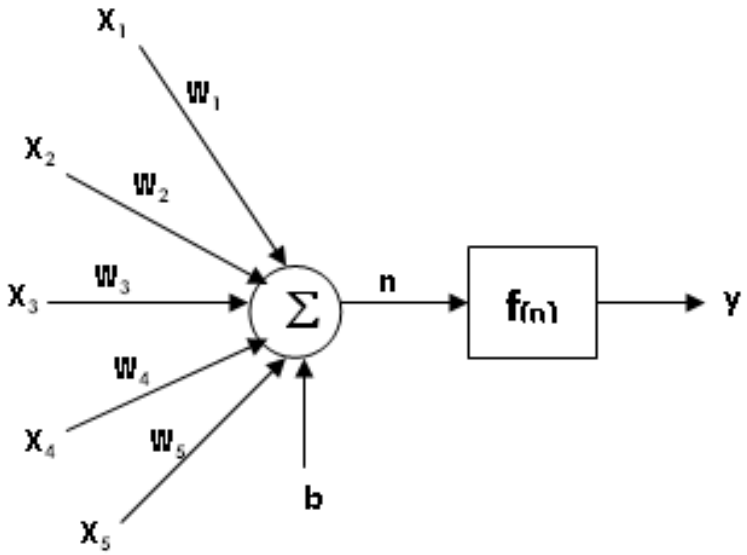
COMO TRABAJA UNA NEURONA ARTIFICIAL



COMO TRABAJA UNA NEURONA ARTIFICIAL

X1, X2,.., Xn son las **señales de entrada** y cada una pasa a través de un peso W, llamado **peso sináptico** de la conexión, cuya función es análoga a la de la **función sináptica de la neurona biológica**

El **nodo sumatorio** acumula todas las señales de entrada multiplicadas por los pesos y las pasa a la salida a través de una **función de activación o transferencia** f(n), (b es el sesgo).



COMO TRABAJA UNA RED NEURONAL

1. El conjunto de unidades de procesamiento (**neuronas** formales).

2. El **estado interno o de activación** de las neuronas.

3. Las **conexiones entre las neuronas**.

4. Las conexiones con el ambiente.

COMO TRABAJA UNA NEURONA

5. La **regla de propagación** $h_i(t)= g(w_{ij}, x_j(t))$

Ej. $h_i(t)= \sum_j w_{ij}x_j(t)$

6. La **función de activación**

$$a_i(t)= f_i(a_i(t-1), h_i(t))$$

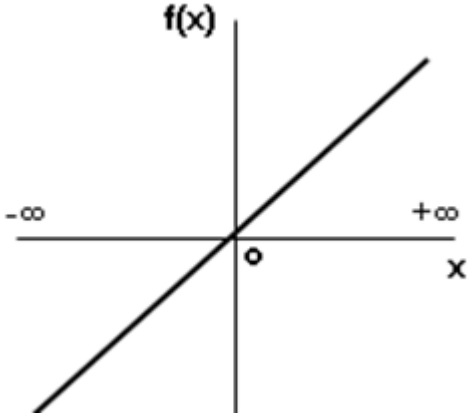
7. La **función de transición o de salida**

$$y_i(t)=F_i(a_i(t))$$

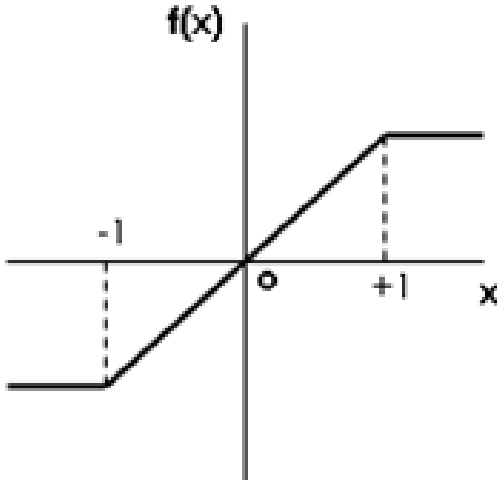
Función de activación

Función identidad o función lineal:

$$f(x) = x$$



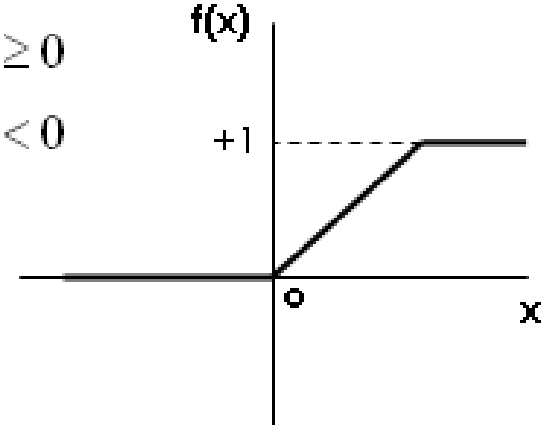
Función lineal por tramos



$$f(x) = \begin{cases} -1, x < -1 \\ x, +1 \le x \le -1 \\ +1, x > +1 \end{cases}$$

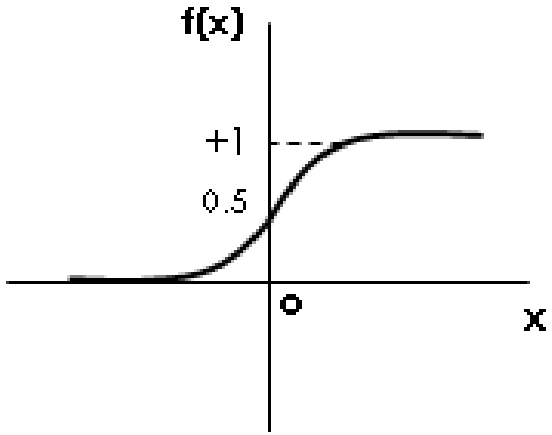
Función escalón

$$f(x) = \begin{cases} 1 & x \ge 0 \\ 0 & x < 0 \end{cases}$$



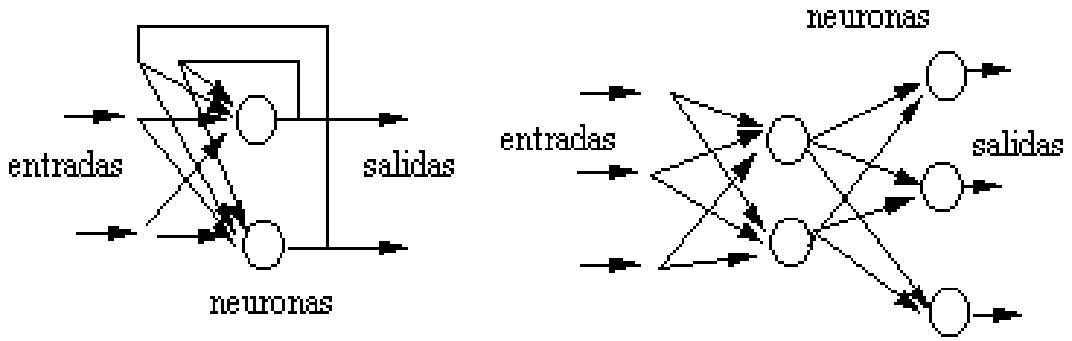
Función sigmoidal

$$f(x) = \frac{}{1+}$$



COMO TRABAJA UNA RED DE NEURONAS

8. La topología o arquitectura de la red



- **conexión total** (todas las neuronas interconectadas) o **conexión parcial** (por ejemplo, las redes de capas).
- **Realimentada** o **unidireccional**

Topologías de las RNA

Redes monocapa:

- Redes con una sola capa.
- Para unirse las neuronas crean conexiones laterales para conectar con otras neuronas de la única capa.

Redes multicapas:

- Generalización de las anteriores donde existe un conjunto de capas intermedias entre la entrada y la salida llamadas *capas ocultas.*
- Pueden ser:

    Propagación hacia adelante

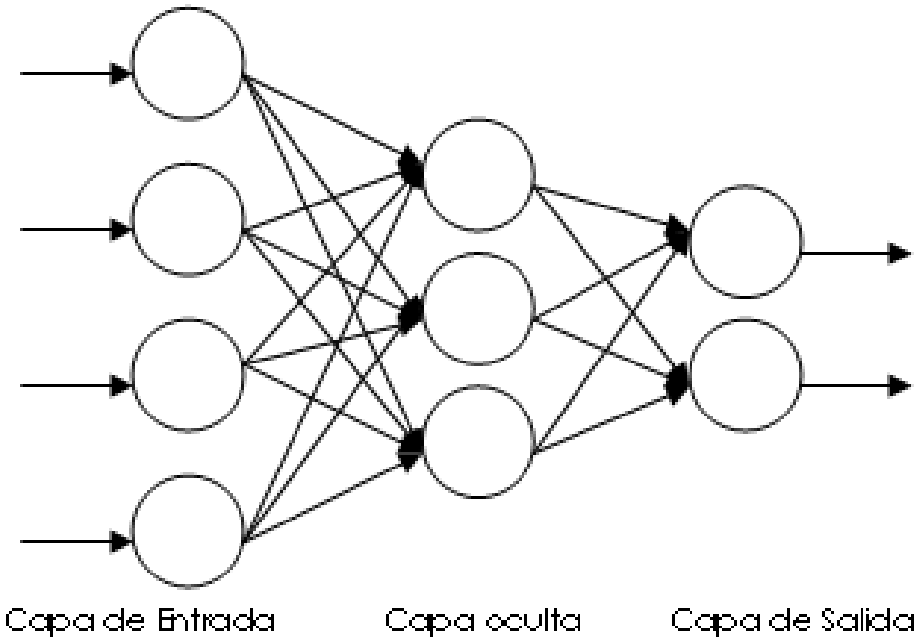
    Propagación hacia atrás

        Redes recurrentes

        Redes de alimentación lateral

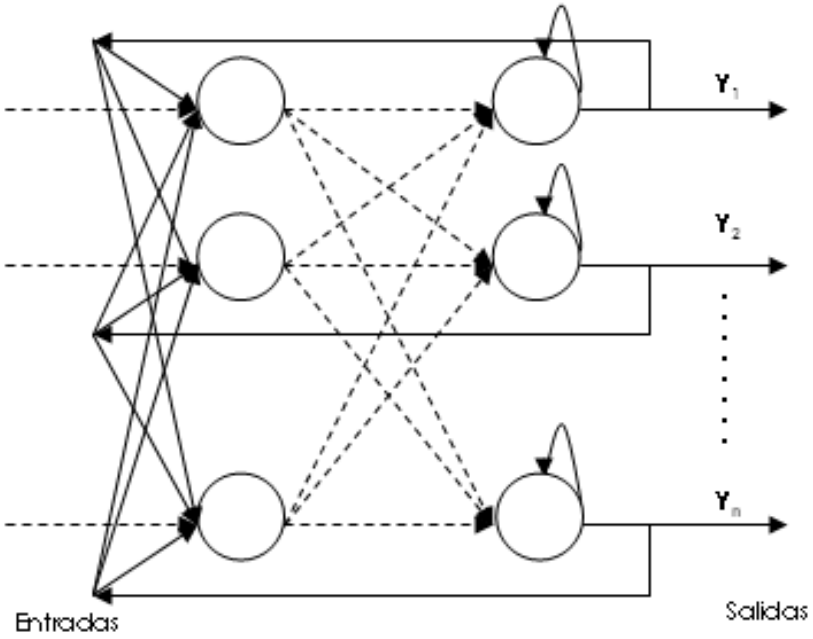
Redes Multicapas

- **Capa de Entrada:** está constituida por los nodos de entrada, que reciben directamente la información de las fuentes externas a la red.
- **Capas Ocultas:** no tienen contacto con el exterior ya que se encuentran ubicadas entre la capa de entrada y la capa de salida. La cantidad de capas ocultas dependerá del problema en estudio y deben especificarse en la arquitectura.
- **Capa de Salida:** está constituida por los nodos que transfieren la información a la salida de la red y de acuerdo al tipo de problema en estudio se determinará el número de neuronas de salida.

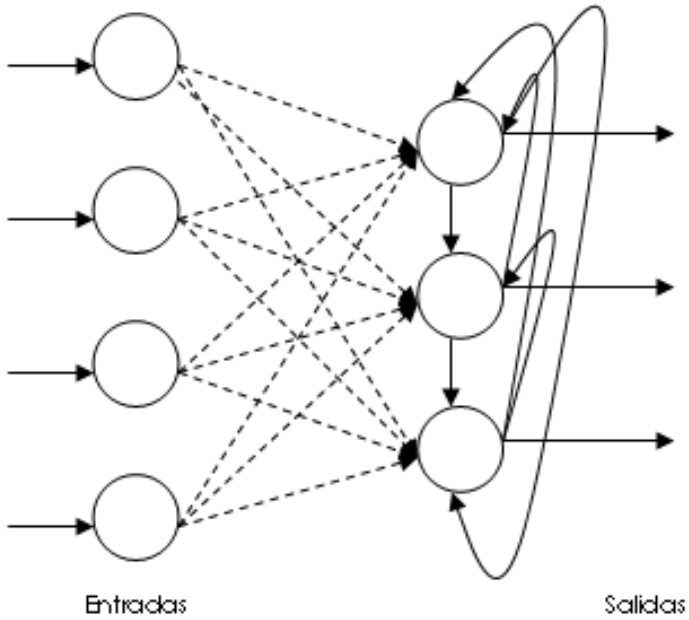


Redes Multicapas

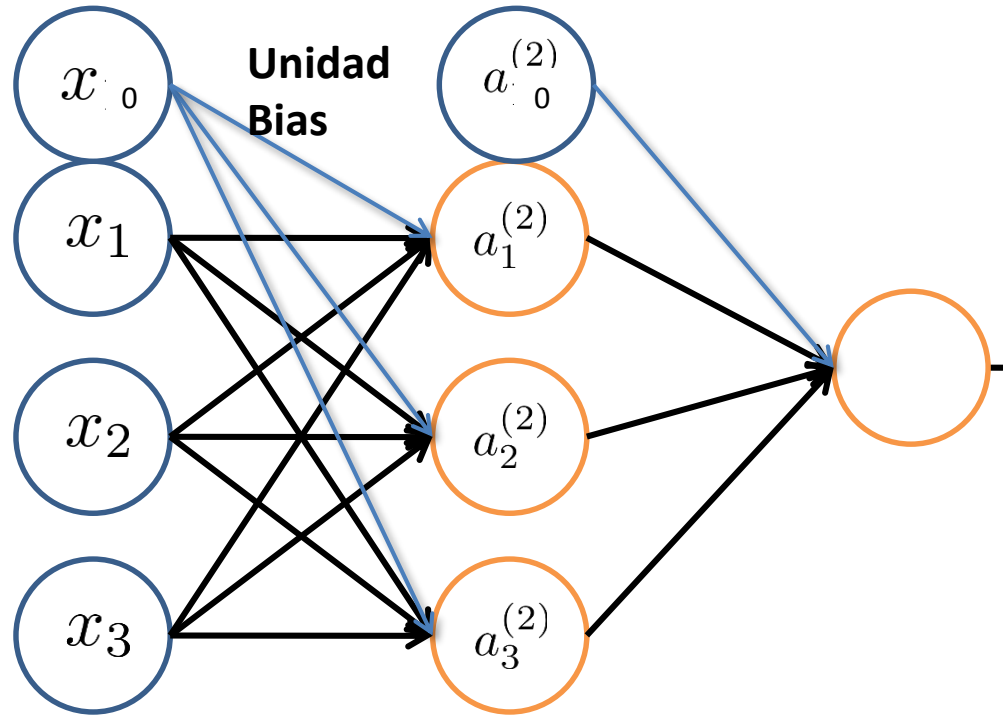
Redes recurrentes



Redes de alimentación lateral



Modelo de Redes Neuronales



$a_i^{(j)} =$ "activación" unidad i en capa j

$\Theta^{(j)} =$ Matriz de pesos entre capa j y capa $j+1$

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3) \quad \ldots$$

$$h_\Theta(x) = g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)})$$

MEMORIAS ASOCIATIVAS

- RN ALMACENAN INFORMACIÓN APRENDIDA REFLEJADA EN SUS PESOS

- AL APLICARLE UNA ENTRADA LA RNA RESPONDE CON UNA SALIDA ASOCIADA A DICHA INFORMACIÓN DE ENTRADA

ASOCIACIÓN ENTRADA/SALIDA

Auto-asociativa vs. Hetero-asociativa

Aprendizaje

en las RNs

APRENDIZAJE

- El aprendizaje de una RNA se basa en un proceso que **permite que la red aprenda a comportarse según unos objetivos específicos**.

- El aprendizaje le da la capacidad a la RNA de **cambiar su comportamiento**, es decir su proceso de entrada-salida, como resultado de los cambios en el medio.

- En particular, las reglas de aprendizaje son procedimientos que se siguen para **ajustar los parámetros de la red** a partir de un proceso de estimulación por el entorno de la red

- La mayoría de las veces consiste en **determinar un conjunto de pesos**

- El aprendizaje es esencial para la mayoría de las arquitecturas de RNA, por lo que la **elección de un algoritmo de aprendizaje** es algo de gran importancia en el diseño de una red.

APRENDIZAJE

- Al finalizar la fase de entrenamiento/aprendizaje de una RNA, se espera que la red haya aprendido lo suficiente para **resolver otro problema similar** satisfactoriamente.

- No existe en la literatura una metodología que indique la **manera de escoger el tipo o forma de aprendizaje** de la red para obtener resultados óptimos.

- Tipo de aprendizaje viene determinado por la **forma en que los parámetros** se deben adaptar

APRENDIZAJE

MODIFICAR PESOS DE LAS CONEXIONES DE LAS NEURONAS (CREAR, DESTRUIR, MODIFICAR)

$$w_{ij}(t+1)=w_{ij}(t) + \Delta w_{ij}(t)$$

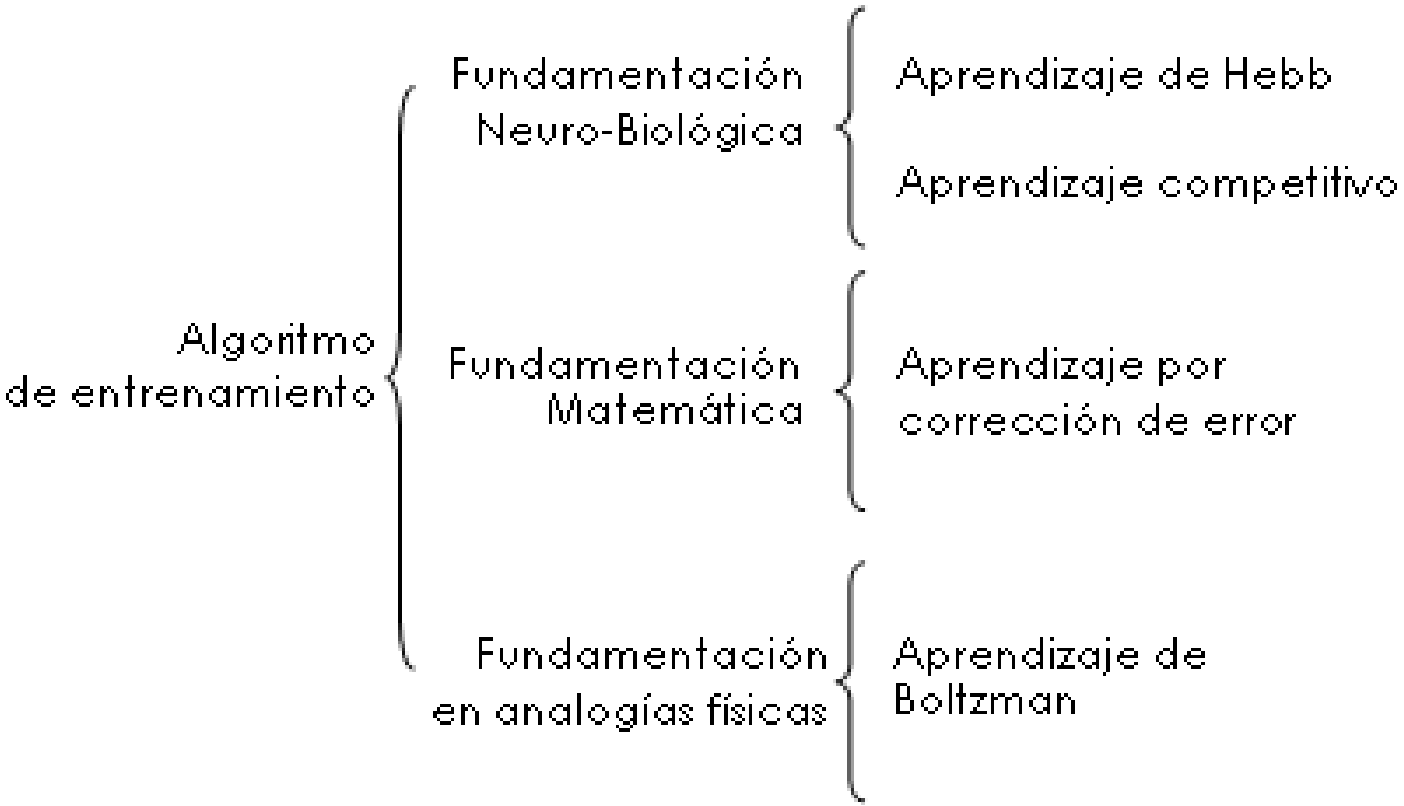
Algoritmo General de un RNA

1. Presentación de las entradas

2. Calculo de la salida actual

3. Adaptación de los pesos

APRENDIZAJE



Clasificación de los Algoritmos de Aprendizaje basados en su fundamentación conceptual

APRENDIZAJE

A. ***PARADIGMAS DE APRENDIZAJE:*** *Define como se relaciona con su entorno.* Se distinguen por el tipo de retroalimentación que se le ofrece al alumno.

- **Supervisado:** el crítico proporciona la salida correcta.
- **No supervisado:** no se proporciona retroalimentación en absoluto.
- **Basado en recompensa**: la crítica proporciona una evaluación de la calidad (el "premio") de lo hecho por el alumno.

APRENDIZAJE

B. ALGORÍTMOS DE APRENDIZAJE: *DEFINE REGLAS DE APRENDIZAJE (MODIFICACIÓN DE LOS PESOS)*

CORRECCIÓN DE ERROR
BOLTZMAN
HEBBIANO
COMPETITIVO
EVOLUTIVO

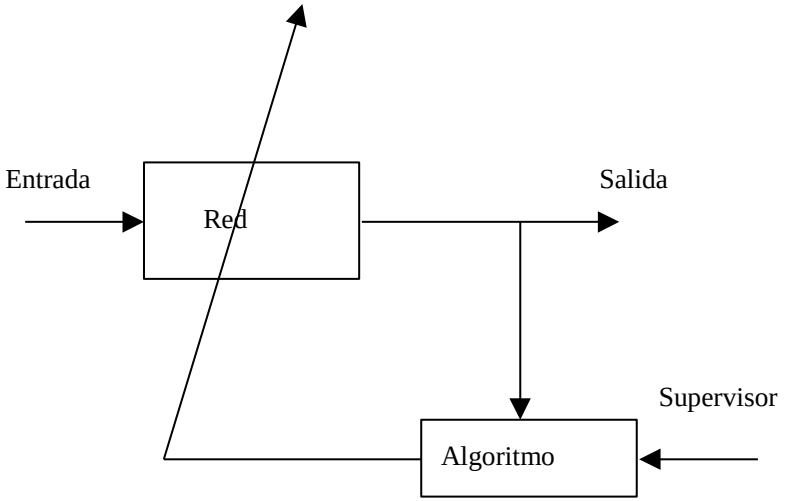
SUPERVISADO

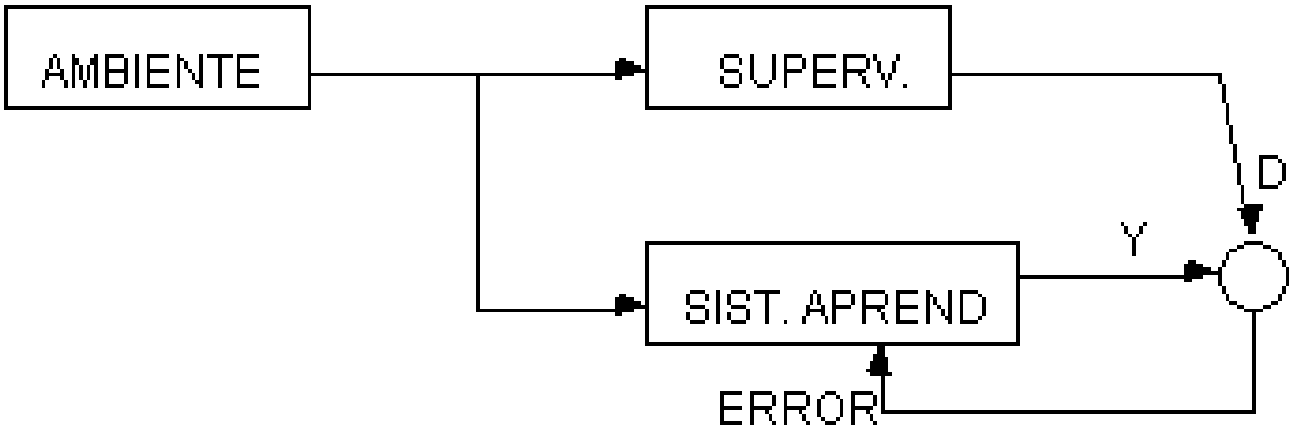
Respuesta correcta para cada ejemplo dada

- SE DAN DATOS DE ENTRADA Y SALIDA OBJETIVO

- SALIDA RED DEBE CONCORDAR CON LA DESEADA

SUPERVISADO





CORRECCIÓN DE ERROR

CONOCIDO TAMBIEN COMO **DESCENSO DE GRADIENTE**

$$E_k(t) = D_k(t) - Y_k(t)$$

D_k: respuesta deseada

Y_k: respuesta neurona k $\qquad Y_k=F(X_k)$

X_k: entrada neurona k

$$\Delta W_{ij}(t) = \alpha E_i(t) X_j(t)$$

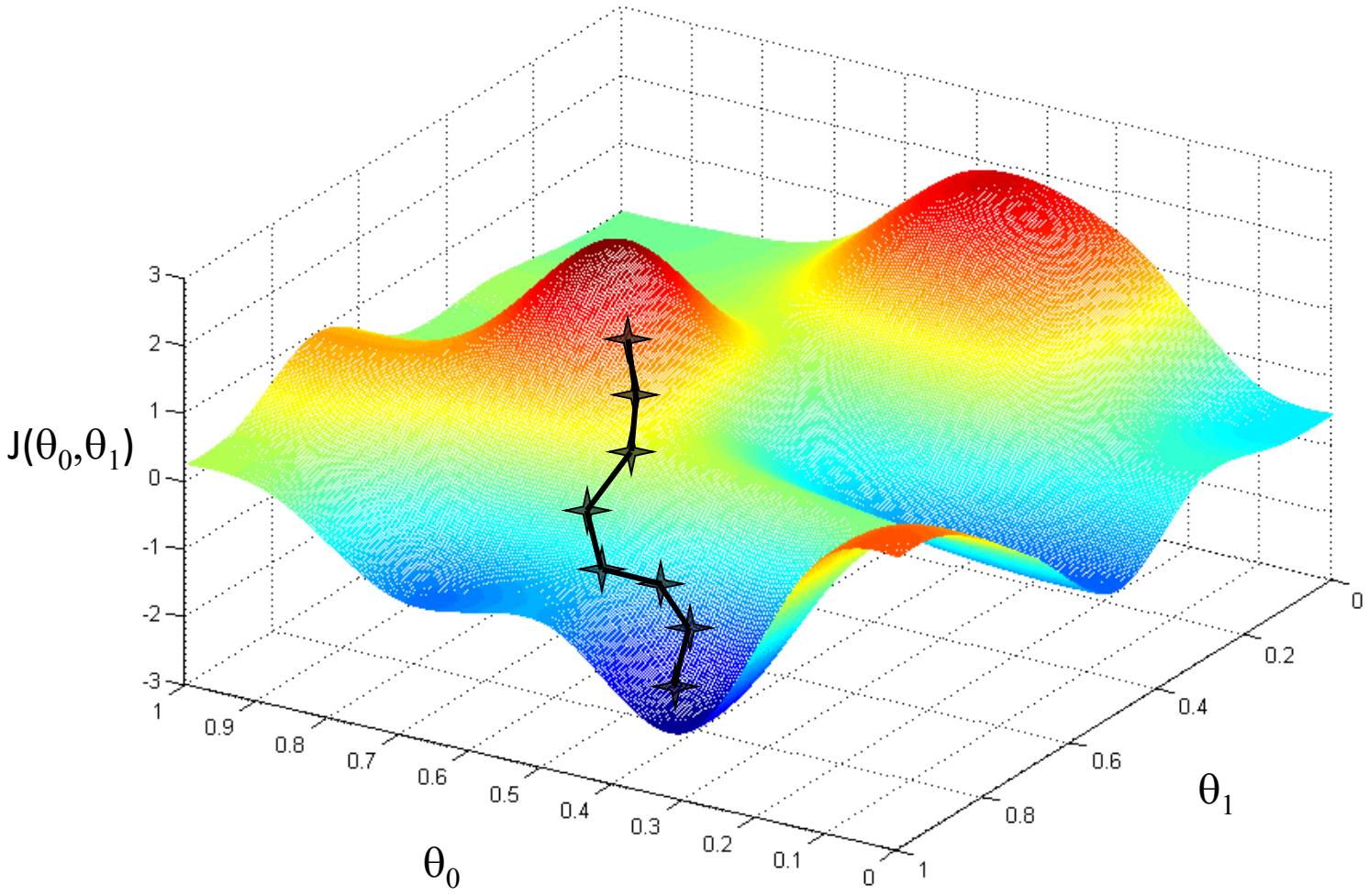
α: taza de aprendizaje

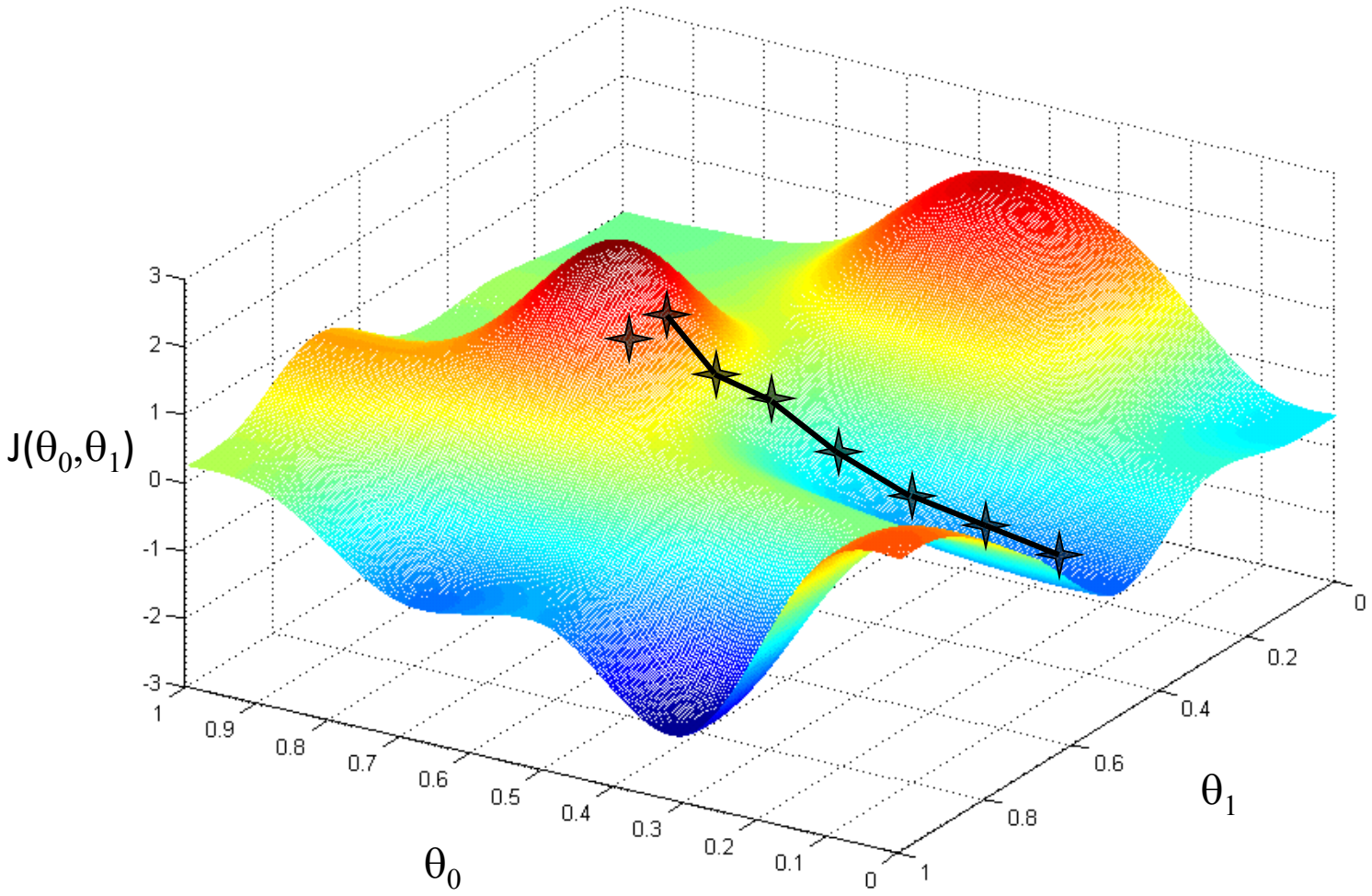
CORRECCIÓN DE ERROR

ALGORITMO

1. CALCULAR EDO. DE LA RED (Y_i)
2. CALCULAR ERROR (E_i)
3. AJUSTAR PESOS

$$w_{ij}(t+1)=w_{ij}(t) + \Delta w_{ij}(t)$$


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1


J(θ0,θ1)
3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ0
0
0.2
0.4
0.6
0.8
1
θ1

Algoritmo de un RNA

1. Inicialización de los pesos y umbral

2. Fase de entrenamiento
   1. Presentación de las entradas y salida deseada
   2. Adaptación de los pesos

3. Fase de Reconocimiento
   1. Presentación de una entrada dada
   2. Salida reconocida

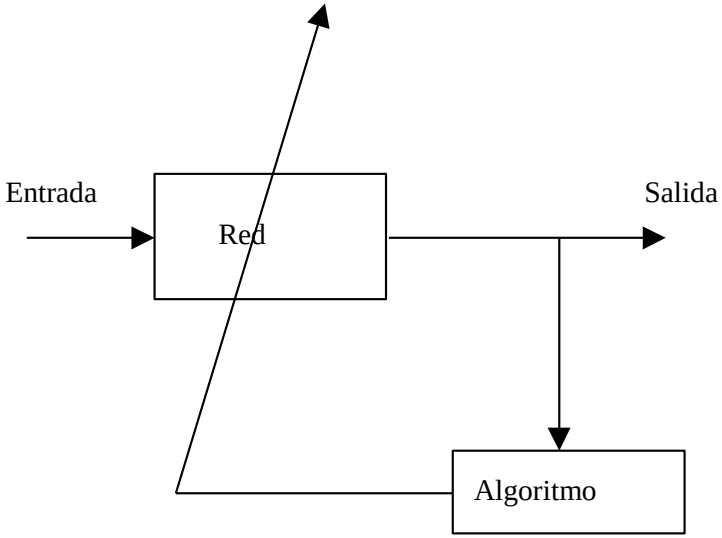
NO SUPERVISADO (AUTOORGANIZADO)

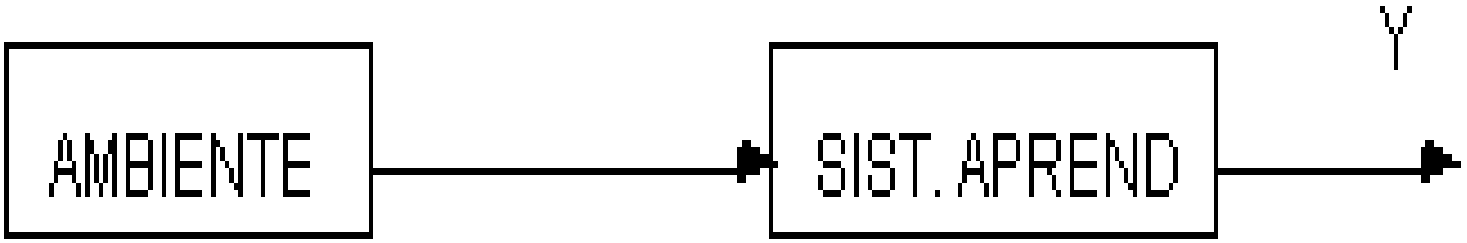
- NO RECIBE INFORMACIÓN DE SU ENTORNO (Se reciben patrones sin la respuesta deseada)

- CON LOS DATOS SE BUSCAN CORRELACIONES O REGULARIDADES EN EL CONJUNTO DE ENTRADAS:
  - EXTRAER RASGOS
  - AGRUPAR PATRONES SEGÚN SU SIMILITUD

- MAPAS AUTOORGANIZADOS

NO SUPERVISADO (AUTOORGANIZADO)





HEBBIANO

- MÁS VIEJO

- DOS O MAS NEURONAS ACTIVADAS SIMULTANEAMENTE

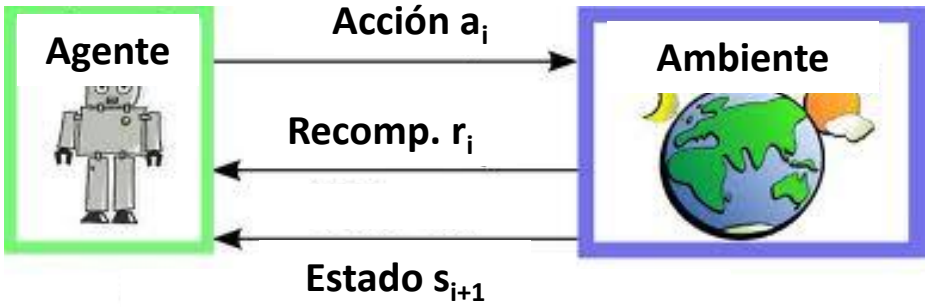
=> REFORZAR LA CONEXIÓN ENTRE ELLAS

$$\Delta W_{ij} = \alpha Y_i Y_j$$

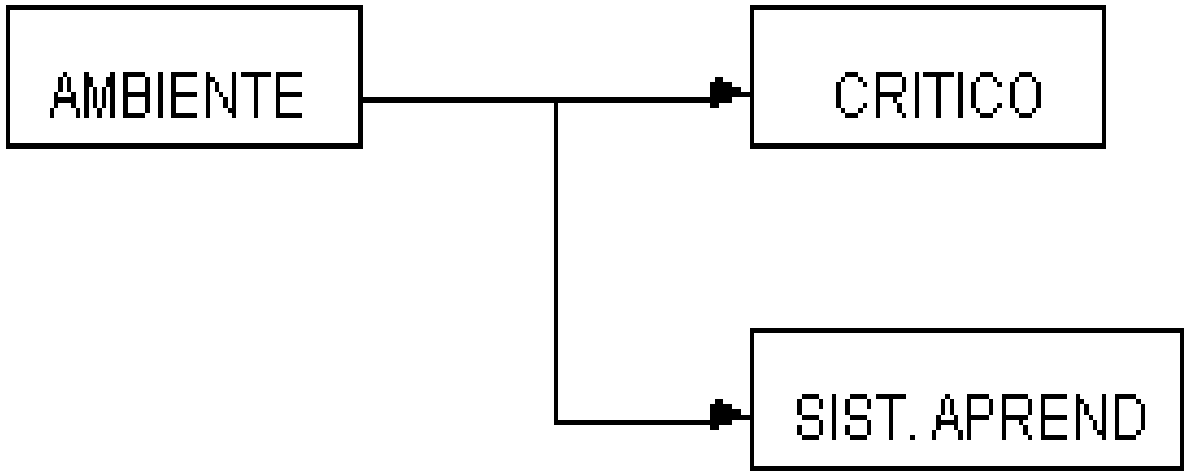
REFORZADO

Recompensa ocasional

- SUPERVISOR INDICA SI SALIDA SE AJUSTA A LO DESEADO O NO (que bien o mal se esta haciendo, no si es la salida deseada!!)
- SUPERVISOR HACE PAPEL DE CRÍTICO MÁS QUE DE MAESTRO (premio-castigo)



REFORZADO



REFORZADO

- Particularmente útiles en los ámbitos en los que exista información de reforzamiento (expresado como penalizaciones o recompensas) proporcionada después de una secuencia de acciones realizadas en el ambiente.

- Métodos comunes: Q-Learning y diferencia temporal- (TD)
  - **Q-Learning:** aprende la utilidad de llevar a cabo acciones que me lleven a ciertos estados,
  - **TD** aprender la utilidad de estar en ciertos estados.

REFORZADO

- Todos los métodos de aprendizaje por refuerzo están inspirados en
  - fórmulas de actualización de la utilidades esperadas
  - exploración del espacio de estados.

- La actualización es a menudo una suma ponderada de:
  - valor actual utilidad,
  - refuerzo obtenido al realizar una acción y
  - utilidad esperada por el siguiente estado alcanzado, después se realiza la acción.

Tareas de Aprendizaje

- **Aproximación**
- **Asociación**
  - **Autoasociativa**
  - **Heteroasociativa**
- **Clasificación**
- **Predicción**
- **Control** planta:u(t),y(t) modelo:r(t),d(t) lim|d(t)-y(t)|=0
- **Filtraje**

Modelos Neuronales

Clasificación por tipo de aprendizaje y arquitectura

Híbridos: RBF (RADIAL BASIC FUNCTION)

Supervisados

Realimentados: feed-propagation

Unidireccionales: PERCEPTRON, MNA, BOLTZMAN, backpropagation,

No supervisados

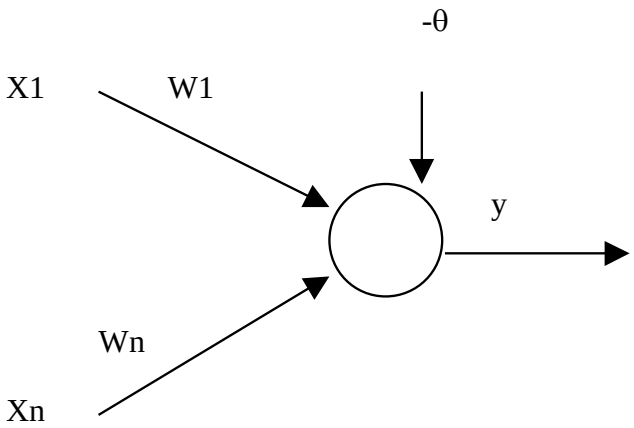
Realimentados: ART, HOPFIELD

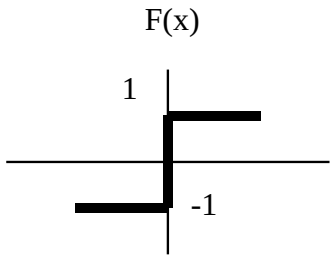
Unidireccionales: KOHONEN

Reforzados

PERCEPTRÓN

- 1ER MODELO DE RED DE NEURONAS ARTIFICIALES (ROSEMBLATT 1958)
- APRENDE PATRONES SENCILLOS (2 CLASES)
- 1 NEURONA





$Y=F(\sum WiXi-\theta)$

PERCEPTRÓN

- REGIONES QUE INDICA A QUE PATRÓN PERTENECE CADA CLASE SEPARADAS POR UN HIPERPLANO

  => PATRONES SEPARABLES GEOMÉTRICAMENTE

  => DOS ENTRADAS LINEA RECTA $X2=W1X1/W2+\theta/W2$

  => TRES ENTRADAS PLANO

- NO RESUELVE OR-EXCLUSIVO

PERCEPTRÓN

- APRENDIZAJE: SUPERVISADO

- ALGORÍTMO:

1. INICIAR PESO Y UMBRAL

2. PRESENTAR PAR ENTRADA-SALIDA

3. CALCULAR SALIDA ACTUAL
$Y(t)$

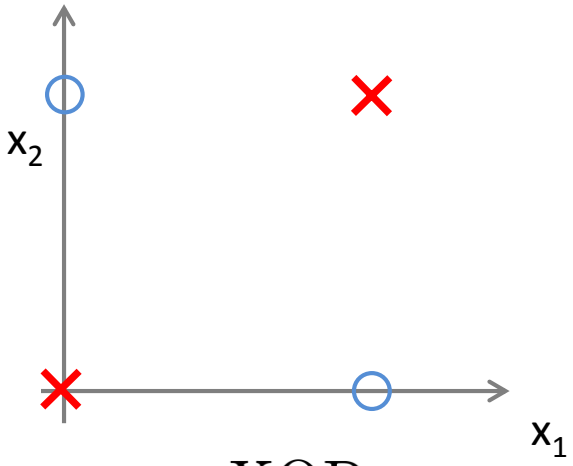
4. ADAPTAR LOS PESOS
$Wi(t)=Wi(t)+\alpha[d(t)-Y(t)]Xi(t)$

HASTA QUE $d(t)-y(t)^2$ valor pequeño

5. REGRESAR AL PASO 2

Ejemplo de clasificación no lineal: XOR/XNOR

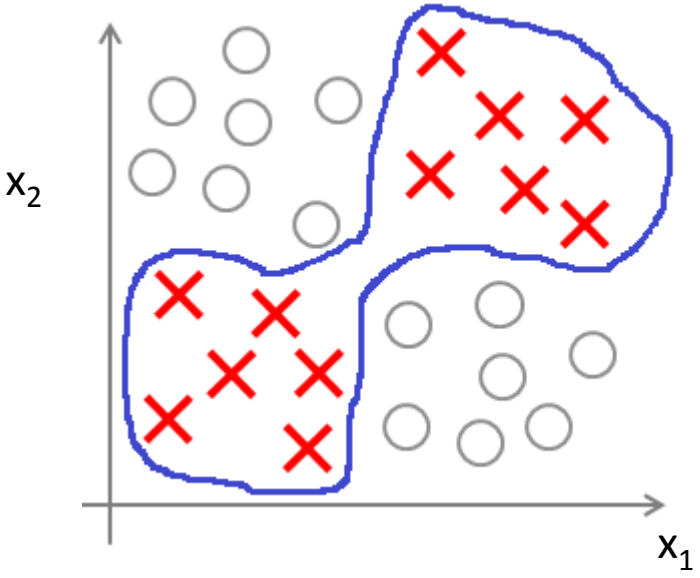
x_1, x_2 are binary (0 or 1).



$y = x_1 \text{ XOR } x_2$

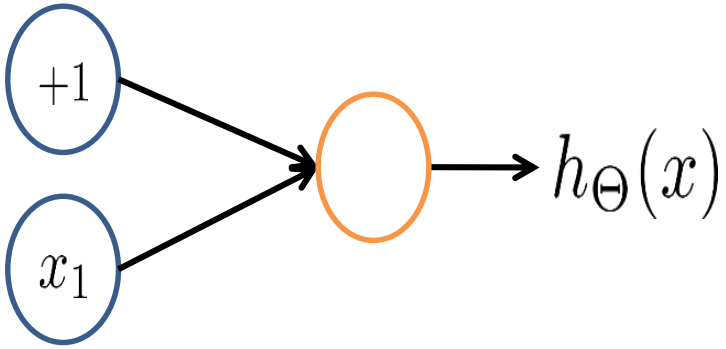
$x_1 \text{ XNOR } x_2$

$\text{NOT } (x_1 \text{ XOR } x_2)$



Negation

X_1 es binario (0 o 1).

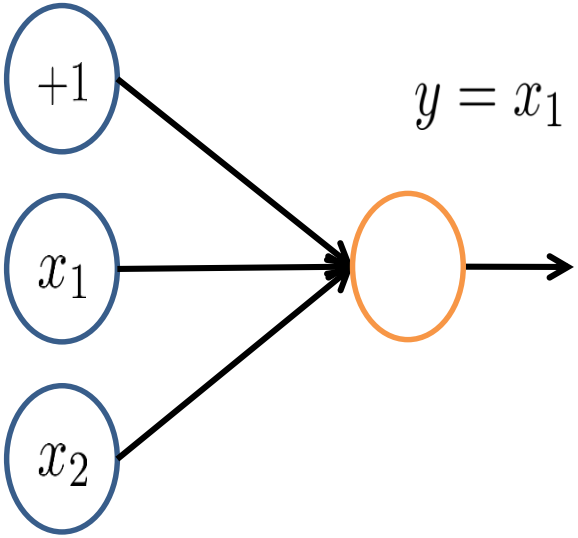


x_1	$h_\Theta(x)$
0	1
1	0

$$h_\Theta(x) = g(10 - 20x_1)$$

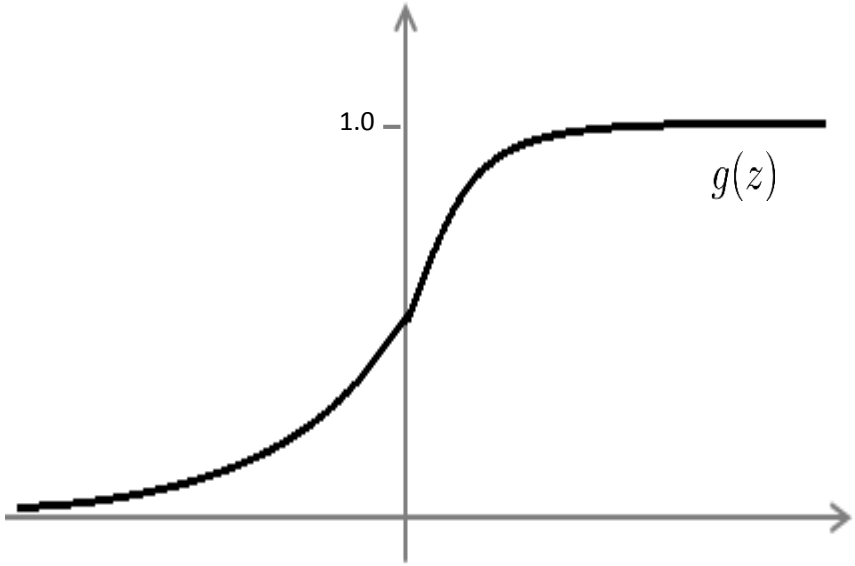
AND

$x_1, x_2 \in \{0, 1\}$



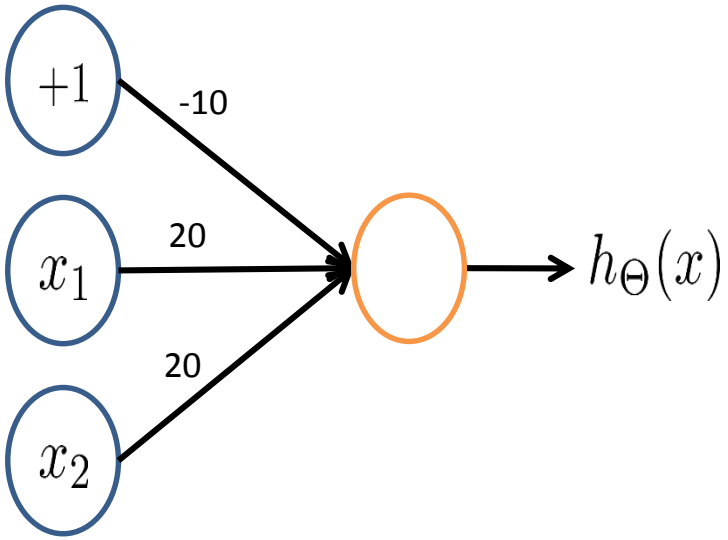
$y = x_1$ AND x_2

$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$



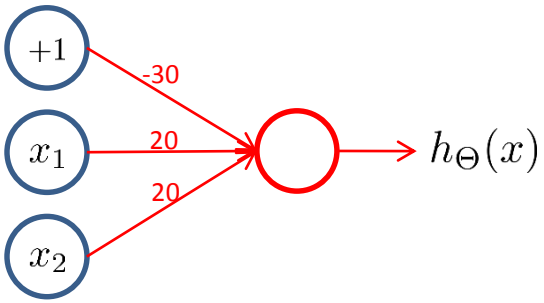
x_1	x_2	$h_\Theta(x)$
0	0	0
0	1	0
1	0	0
1	1	1

OR

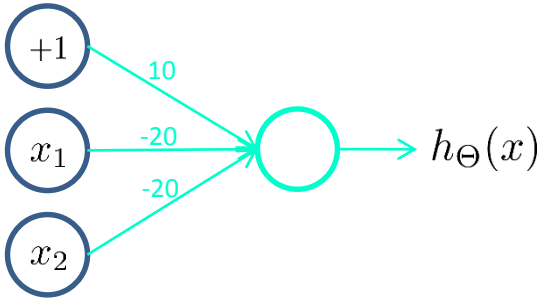


x_1	x_2	$h_\Theta(x)$
0	0	0
0	1	1
1	0	1
1	1	1

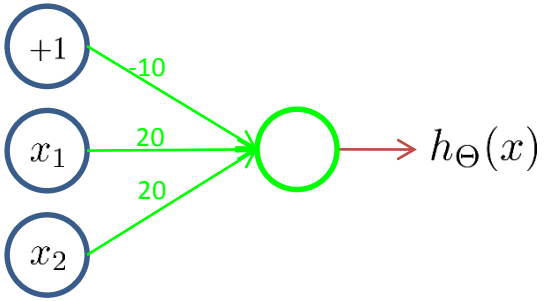
x_1 XNOR x_2



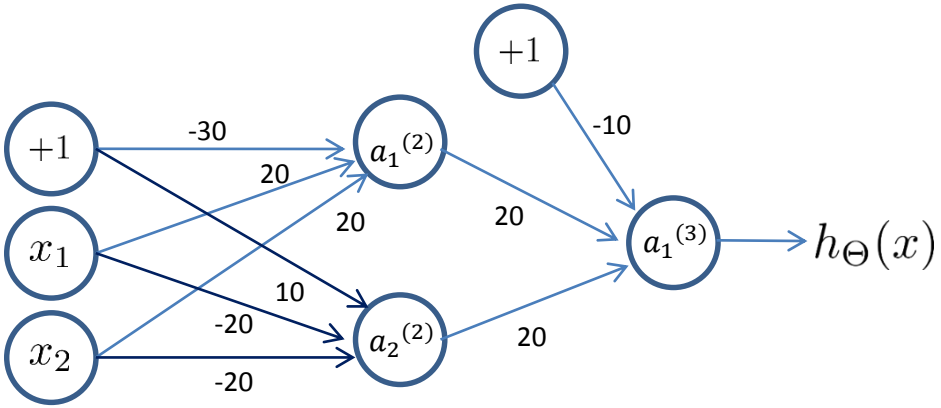
x_1 AND x_2



(NOT x_1) AND (NOT x_2)



x_1 OR x_2



x_1	x_2	$a_1^{(2)}$	$a_2^{(2)}$	$h_\Theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

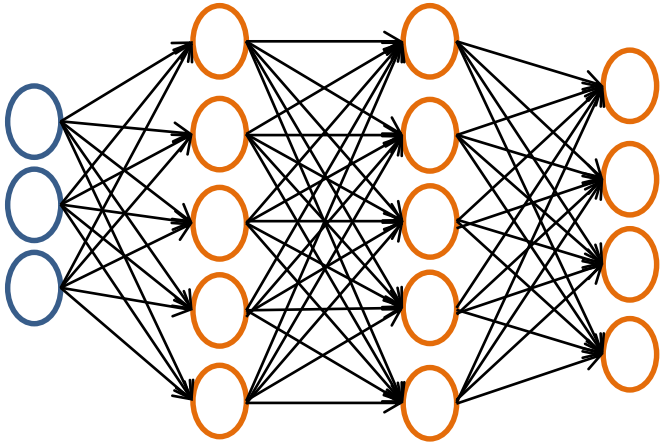
Clasificación multi-clase

Peatón

Carro

Motocicleta

Camión

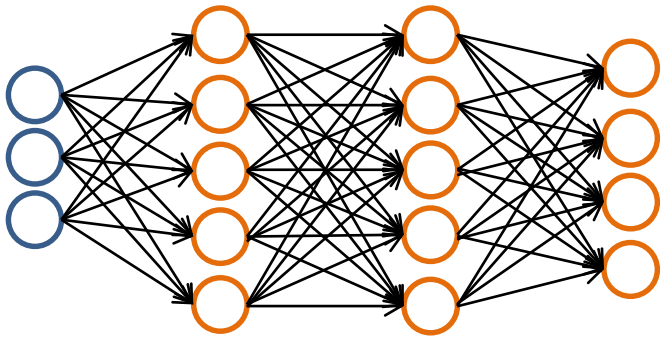


$$h_\Theta(x) \in \mathbb{R}^4$$

Será $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ etc.

peatón **carro** **motocicleta**

Clasificación multi-clase



$h_\Theta(x) \in \mathbb{R}^4$

Se quiere $h_\Theta(x) \approx \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $h_\Theta(x) \approx \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, etc.

cuando peatón carro motocicleta

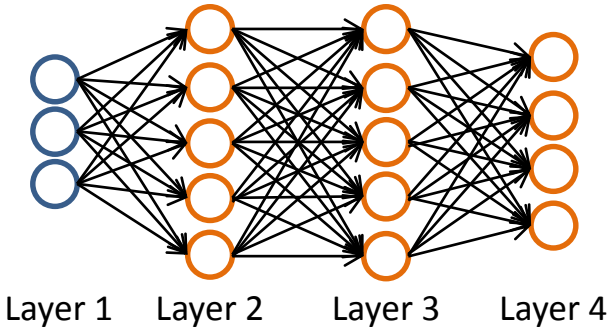
Conj. Entrenam. $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})$

$y^{(i)}$ será $\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

peatón carro motocicleta camión

Función de costo

Redes Neuronales (Clasificación)



$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

$L =$ total de # de capas en la red

$s_l =$ # de unidades (sin contar las bias) en la capa l

Clasificación binaria

$y = 0 \text{ or } 1$

1 salida

Clasificación Multi-clase(K clases)

$y \in \mathbb{R}^K$ E.g. $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

peatón carro motocicleta camión

K salidas

Función de costo

Regresión lógica:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \theta_j^2$$

Redes neuronales:

$$h_\Theta(x) \in \mathbb{R}^K \quad (h_\Theta(x))_i = i^{th} \text{ output}$$

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(h_\Theta(x^{(i)}))_k + (1 - y_k^{(i)}) \log(1 - (h_\Theta(x^{(i)}))_k)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_{ji}^{(l)})^2$$

Gradiente

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log h_\theta(x^{(i)})_k + (1 - y_k^{(i)}) \log(1 - h_\theta(x^{(i)})_k)\right]$$
$$+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}(\Theta_j^{(l)})^2$$

$$\min_{\Theta} J(\Theta)$$

Se calcula:

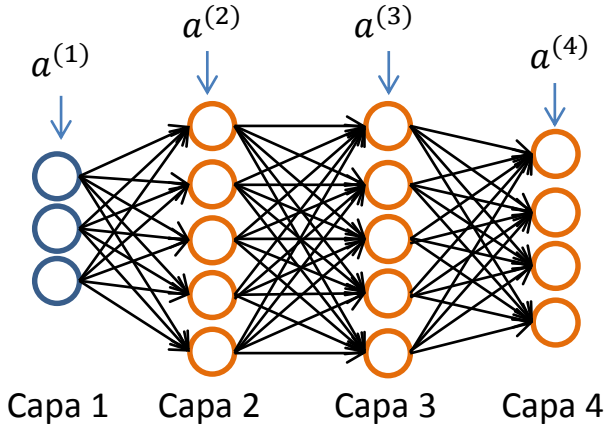
- $J(\Theta)$
- $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$

Gradiente

Dado ejemplo de entrenamiento (x , y):

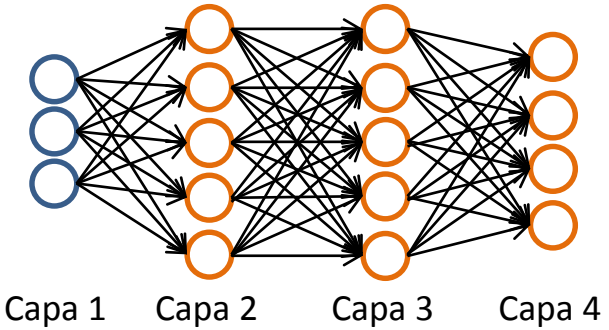
Propagación hacia adelante:

$$
\begin{aligned}
a^{(1)} &= x \\
z^{(2)} &= \Theta^{(1)} a^{(1)} \\
a^{(2)} &= g(z^{(2)}) \quad (\text{add } a_0^{(2)}) \\
z^{(3)} &= \Theta^{(2)} a^{(2)} \\
a^{(3)} &= g(z^{(3)}) \quad (\text{add } a_0^{(3)}) \\
z^{(4)} &= \Theta^{(3)} a^{(3)} \\
a^{(4)} &= h_\Theta(x) = g(z^{(4)})
\end{aligned}
$$



Gradiente: Backpropagation

$\delta_j^{(l)}$ = "error" de nodo j in capa l.



Por cada salida (capa L = 4)

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

$$\delta^{(3)} = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$$

$$\delta^{(2)} = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^{(2)})$$

Algoritmo Backpropagation

Training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Set $\triangle_{ij}^{(l)} = 0$ (for all l, i, j).

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$

Using $y^{(i)}$, compute $\delta^{(L)} = a^{(L)} - y^{(i)}$

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots, \delta^{(2)}$

$$\triangle_{ij}^{(l)} := \triangle_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$$

$$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} \text{ if } j \neq 0$$

$$D_{ij}^{(l)} := \frac{1}{m} \triangle_{ij}^{(l)} \text{ if } j = 0$$

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

REDES BAYESIANAS

Redes Bayesianas

Las redes bayesianas son grafos dirigidos acíclico cuyos nodos representan variables aleatorias en el sentido de Bayes

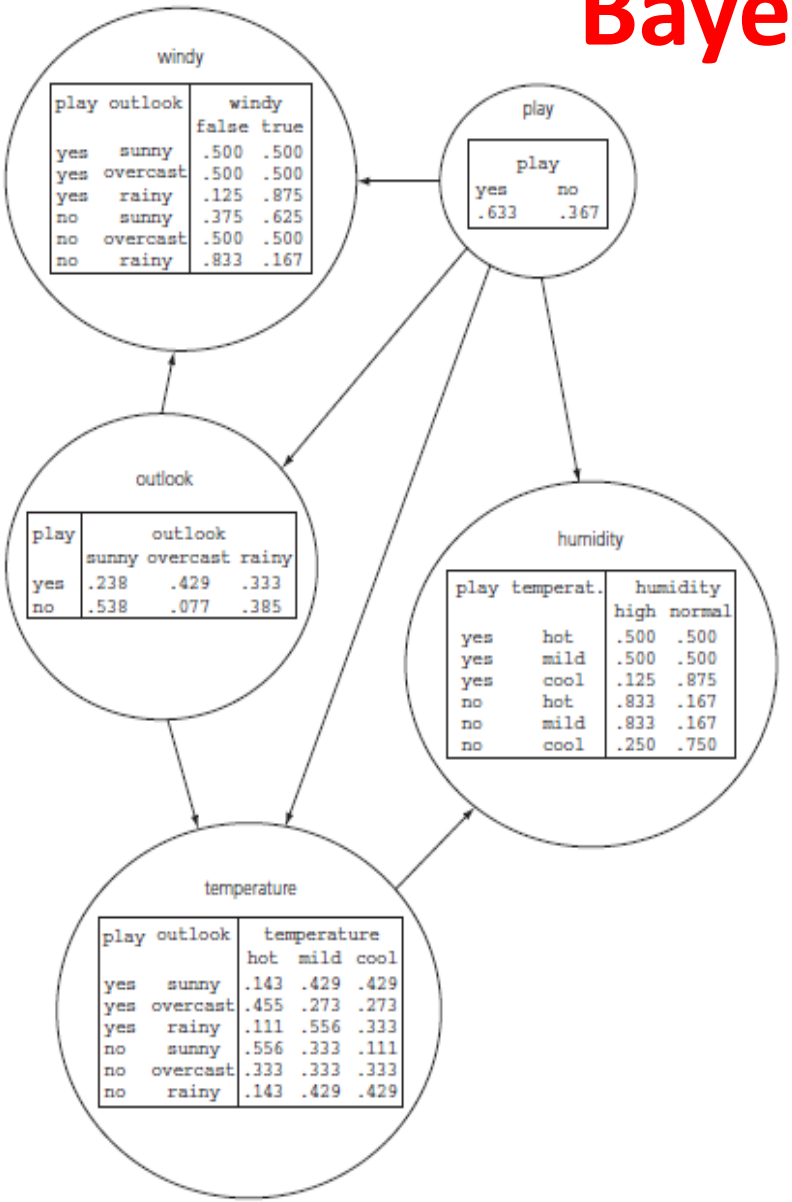
En el teorema de Bayes se expresa la probabilidad condicional de un evento aleatorio A dado B en términos de la distribución de probabilidad condicional del evento B dado A y la distribución de probabilidad marginal de sólo A. Pueden ser cantidades observables, variables latentes, parámetros desconocidos o hipótesis.

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

Redes Bayesianas

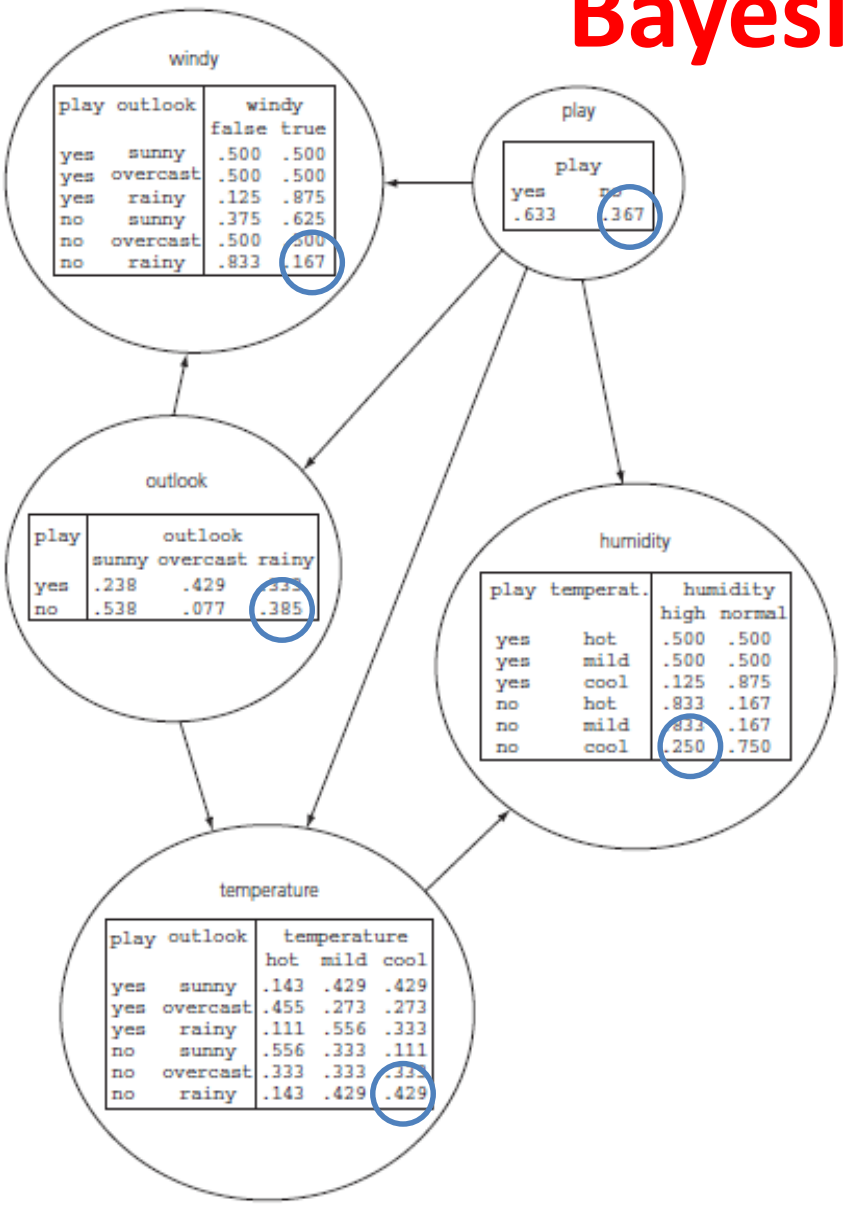
- Las aristas representan dependencias condicionales
- Los nodos que no se encuentran conectados representan variables las cuales son condicionalmente independientes de las otras.
- Cada nodo tiene asociado una función de probabilidad que toma como entrada un conjunto particular de valores de las variables padres del nodo y devuelve la probabilidad de la variable representada por el nodo.

Haciendo predicciones con Redes Bayesianas



Por ejemplo, considerar la posibilidad de una instancia con valores **perspectivas = lluvias, temperatura = frío, humedad = alto, y con viento = true.**

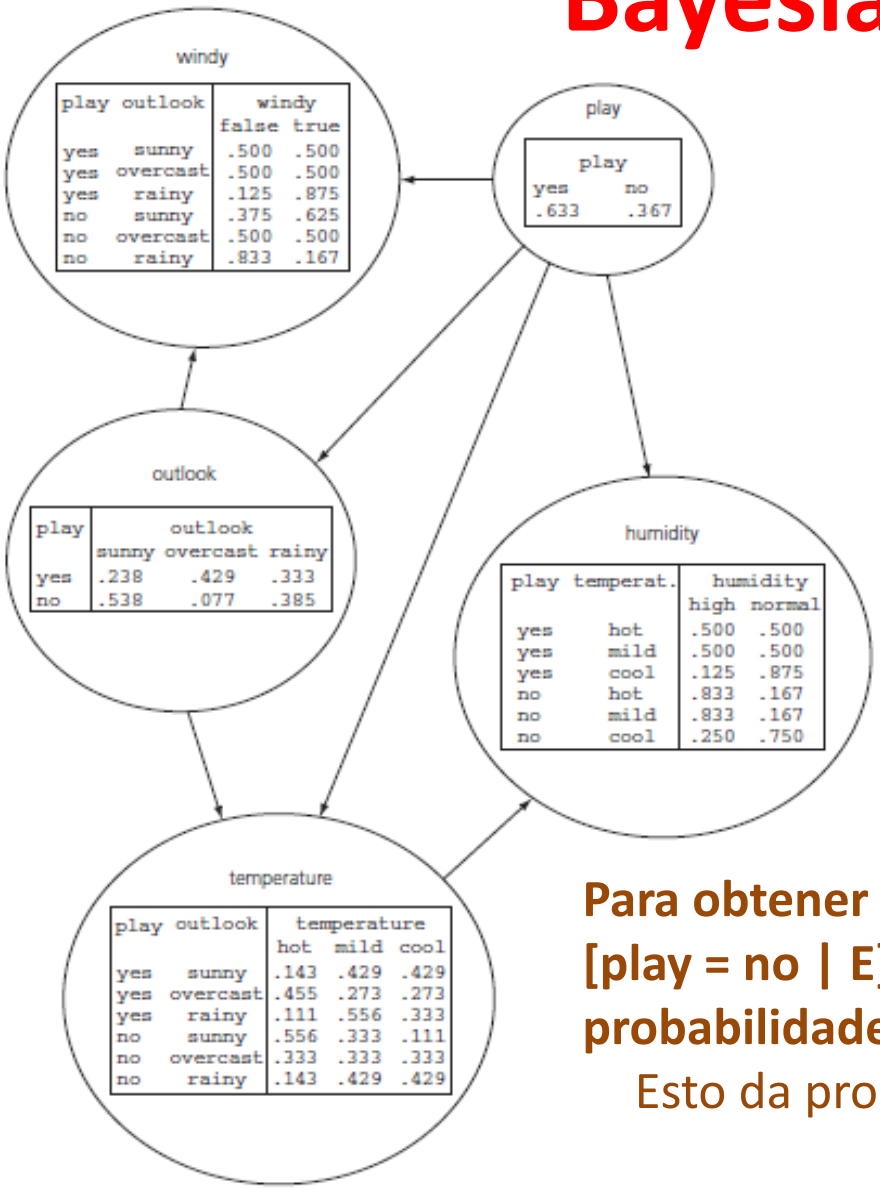
Haciendo predicciones con Redes Bayesianas



Para calcular **la probabilidad para jugar = no**, en la red da la probabilidad:

- 0.367 desde el nodo Jugar,
- 0.385 desde perspectiva,
- 0.429 desde temperatura,
- 0.250 desde humedad, y
- 0.167 desde viento

Haciendo predicciones con Redes Bayesianas



El producto es 0,0025. El mismo cálculo para el **juego = Si** es 0.0077.

Sin embargo, estos no son la respuesta final:

las probabilidades finales deben sumar 1,

Para obtener las probabilidades condicionales Pr [play = no | E] y Pr [play = yes | E], normalizar las probabilidades conjuntas dividiéndolas por su suma. Esto da probabilidad **0,245 para jugar = no y 0.755 para jugar = si**

Aprendizaje de Redes Bayesianas

Consiste en inducir un modelo, estructura y parámetros asociados, a partir de datos.

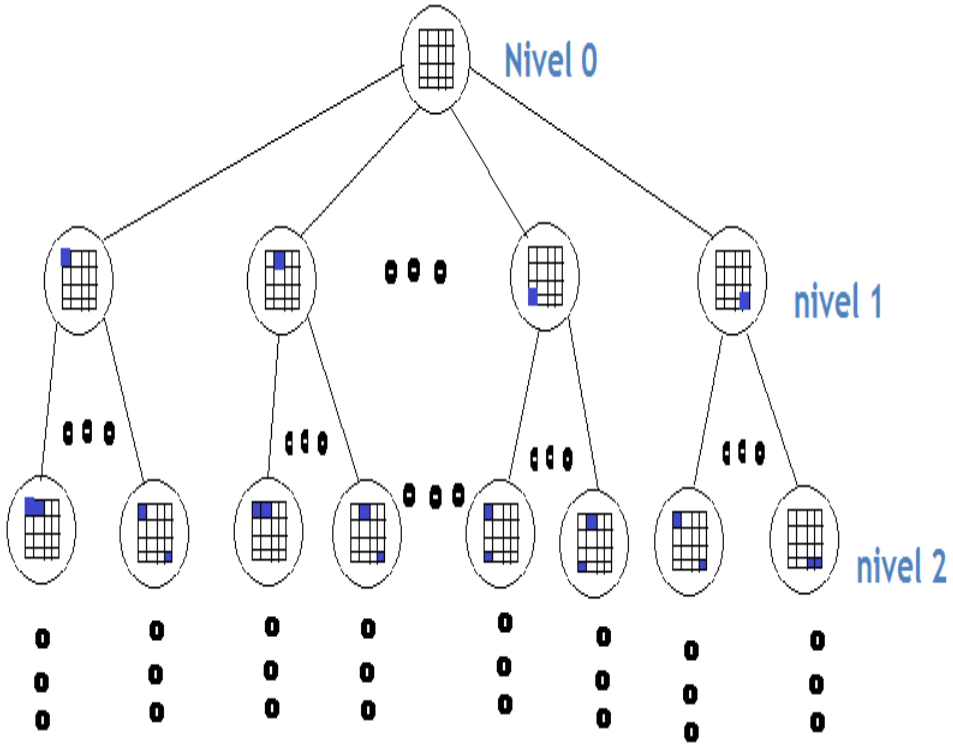
Este puede dividirse naturalmente en dos partes:

- **Aprendizaje estructural**. Obtener la estructura o topología de la red.
- **Aprendizaje paramétrico**. Dada la estructura, obtener las probabilidades asociadas.

Aprendizaje paramétrico

- Aprendizaje por máxima verosimilitud
- Aprendizaje por método de maximización de la expectativa
- Aprendizaje por conteo

Aprendizaje paramétrico



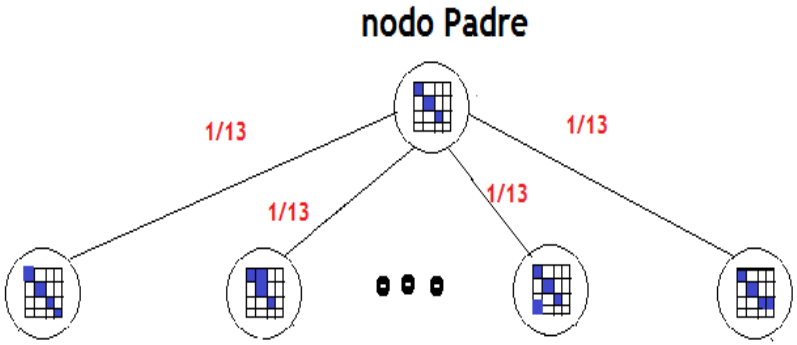
Red bayesiana para el manejo de incertidumbre

- $MUE = \max(U(a_i) * \sum_{i=0}^{n} P_i(^{a_i}/_{P_i}))$

Caso juego: Según la función MUE la mejor acción será aquella en la cual la razón dada entre la utilidad y la probabilidad de que el oponente obtenga una mala jugada sea máxima.

Modelo Matemático de Aprendizaje

Se tiene el siguiente Árbol con 13 nodos



Red bayesiana en su estado de máxima confusión

Según acción del adversario sea buena o no, la rama debe ser premiada (o penalizada) y las del resto de hermanos inversamente modificadas (aprendizaje reforzado)

Para actualizar las ramas se pueden usar los siguientes valores:

- $\mathbf{p_{obj}}$= 6/10 se suma (resta) a la rama evaluada para premiar (castigar)
- $\mathbf{p_{resto}}$=2/10 se resta (suma) al resto de ramas para penalizar (premiar)

Aprendizaje Paramétrico

Estimador de máxima verosimilitud : El método consiste en encontrar el valor que maximice el logaritmo de la verosimilitud (L).

Formalmente, la red bayesiana U es un par $B = \langle G, \Theta \rangle$.

- G es el grafo acíclico en el cual sus vértices corresponden a las variables aleatorias $X_1, \ldots\ldots, X_n$, y cuyos arcos representan dependencias directas entre las variables. El grafo G codifica las suposiciones de independencia de cada variable X_i.
- Θ representa el conjunto de parámetros que cuantifican la red. El mismo contiene un parámetro $\theta_{x_{ij}\pi_{ij}} = P_B(x_{ij}|\pi_{ij})$ para cada posible valor x_{ij} de X_i el cual ocurre según la condición π_{ij},
- B define una única distribución de probabilidad conjunta:

$$P_B(X_1, X_2, \ldots\ldots, X_n) = \prod_{i=1}^{n} \prod_{j=1}^{k_i} \theta_{X_{ij}|\pi_{ij}}$$

Donde ki es el número de valores posibles de la variable X_i.

Aprendizaje Paramétrico

Estimador de máxima verosimilitud : El método consiste en encontrar el valor que maximice el logaritmo de la verosimilitud (L).

Este método requiere una colección de datos de entrenamiento compuesta por *m* casos, que en general se asumen independientes.

Dada la colección de datos de entrenamiento $\Sigma = \{y_1 \dots \dots y_m\}$, en donde cada y_l tiene definido un valor dado para cada X_i que se corresponde a uno de sus valores posibles x_{ij}, tal que $y_l = (x_{l1} \dots \dots x_{ln})^T$, y la colección de parámetros para cada variable $\Theta = (\theta_1 \dots . \theta_n)^T$, donde θ_i es el vector de parámetros de la distribución condicional de la variable X_i,

el logaritmo de la verosimilitud de la colección de datos de entrenamiento es :

$$\log\ \mathrm{L}(\Theta, \Sigma) = \sum_{\mathrm{l=1\ y}}^{\mathrm{m}} \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{Log\ P}(x_{il} | \pi_{\mathrm{il}} \theta_{\mathrm{i}})$$

Aprendizaje Estructural

- Aprendizaje de Arboles
- Aprendizaje NaiveBayes

Aprendizaje NaiveBayes

1. Crear un grafo β vacio con η como el conjunto de vértices.
2. Calcular la información mutua entre todos los pares de variables. Para ello se utiliza la ecuación

$$I(X_i, X_j) = \sum_{X_i, X_j} P(X_i, X_j) Log \frac{P(X_i, X_j)}{P(X_i)P(X_j)}$$

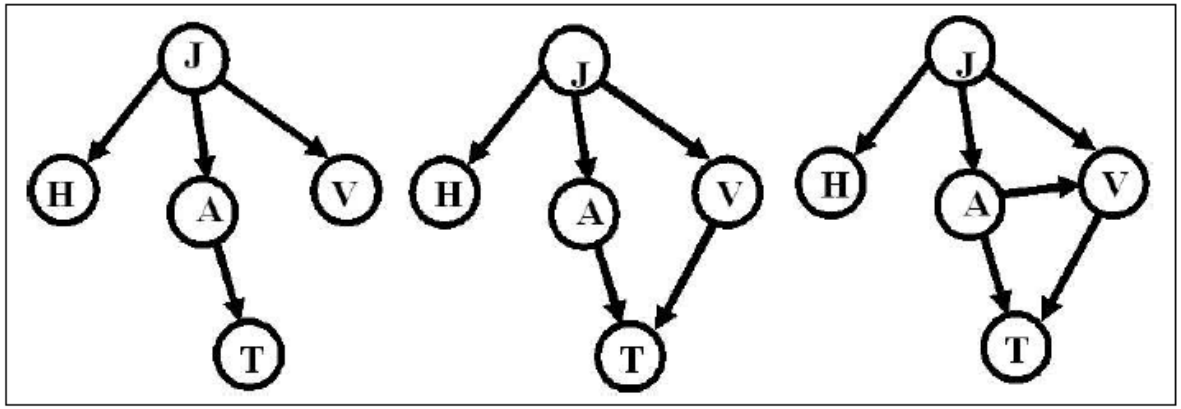
3. Ordenar las informaciones mutuas de mayor a menor.
4. Se agrega el arco con el mayor valor de la información mutua al grafo β.
5. Agregar el siguiente arco mientras no forme un ciclo, de lo contrario, desechar.
6. Repetir los pasos 3, 4 y 5 hasta que se cubran todas las variables η.

Probabilidad conjunta de **n** variables se expresa por la ecuación:

$P(X_1, X_2, \ldots . X_n) = \prod_{i=1}^{n} P(X_i | X_{j(i)})$ En donde $X_{j(i)}$ es el padre de X_i

Aprendizaje basado en medidas globales

1. Generar una estructura inicial - árbol.
2. Calcular la medida de calidad de la estructura inicial.
3. Agregar / invertir un arco en la estructura actual.
4. Calcular la medida de calidad de la nueva estructura.
5. Si se mejora la calidad, conservar el cambio; si no, dejar la estructura anterior.
6. Repetir 3 a 5 hasta que ya no haya mejoras.



Optimización (minimizar) una medida de la diferencia entre la distribución real *P* y la aproximada *P**

$$DI(P,P^*) = \sum_X P(x) Log \frac{P(x)}{P^*(x)}$$

Algoritmos Evolutivos

COMPUTACION EVOLUTIVA

- ENFOQUES ALTERNATIVOS DE BUSQUEDA Y APRENDIZAJE BASADOS EN MODELOS COMPUTACIONALES DE PROCESOS EVOLUTIVOS
- *IDEA:* BUSQUEDA ESTOCASTICA EVOLUCIONANDO A UN CONJUNTO DE *ESTRUCTURAS* Y SELECCIONANDO DE MODO ITERATIVO A LAS *MAS APTAS*

FINALIDAD: SUPERVIVENCIA DEL MAS APTO

MODO: ADAPTACION AL ENTORNO

COMPUTACION EVOLUTIVA

- FENOMENOS NATURALES SIMULADOS:
  - HERENCIA GENETICA
  - LUCHA POR LA SUPERVIVENCIA
- INSPIRACIONES TEORICAS:
  - EVOLUCION DE DARWIN
  - SELECCIÓN DE WEISMANN
  - GENETICA DE MENDELL

COMPUTACION EVOLUTIVA

- EMULACION DE PROCESOS EVOLUTIVOS:
  - POBLACION DE POSIBLES SOLUCIONES => INDIVIDUOS
  - PROCESO DE SELECCIÓN => APTITUD DE LOS INDIVIDUOS
  - PROCESO DE TRANSFORMACION => GENERACION DE NUEVOS INDIVIDUOS

Características de la CE

El propósito genérico de los algoritmos en la CE es guiar una búsqueda estocástica haciendo evolucionar un conjunto de estructuras, seleccionando de modo iterativo las mas adecuadas.

- Se trabaja con la codificación de un conjunto de parámetros y no con los parámetros en si.
- Se usa una función objetivo para definir lo que se desea optimizar. Esta función permite evaluar la calidad de los individuos en cada generación
- Se trabaja con una población del problema, no con un punto simple.
- Se usan reglas de transición probabilisticas, no reglas deterministas

COMPUTACION EVOLUTIVA

- NIVEL DE ABSTRACCION:
  - GENOTIPO => ESTRUCTURAS SON CROMOSOMAS Y GENES. OPERADORES GENETICOS CAMBIAN ESTRUCTURAS (AG)
  - FENOTIPO => ESTRUCTURAS SON COMPORTAMIENTOS. OPERADOR MUTACION CAMBIA ESTRUCTURA (EE,PE).

COMPUTACION EVOLUTIVA

- OBJETIVOS CONFLICTIVOS QUE SE SIGUEN:
  - EXPLOTAR LAS MEJORES SOLUCIONES
  - EXPLORAR EL ESPACIO DE BUSQUEDA
- PARADIGMAS:
  - ALGORITMOS GENETICOS (HOLLAND)
  - PROGRAMAS EVOLUTIVOS (MICHALEWICZ)
  - PROGRAMACION GENETICA (KOZA)
  - ESTRATEGIAS EVOLUTIVAS (RECHENBERG SCHWEFEL)
  - PROGRAMACION EVOLUTIVA (FOGEL)

COMPUTACION EVOLUTIVA

- MACROALGORITMO:

  1.- POBLACION INICIAL

  2.- EVALUACION DE LOS INDIVIDUOS

  3.- REPRODUCCION INICIAL

  4.- REEMPLAZO

  5.- CONDICION DE FINALIZACION O REGRESA A PASO 2

COMPUTACION EVOLUTIVA

- ASPECTOS DE IMPLEMENTACION:
  - REPRESENTACION DE LOS INDIVIDUOS
  - MEDIDAS DE ADAPTACION (FUNCION ADAPTATIVA)
  - MECANISMOS DE SELECCIÓN Y REMPLAZO
  - INTERPRETACION DE LOS RESULTADOS
  - PARAMETROS Y VARIABLES QUE CONTROLAN EL PROCESO
  - OPERADORES GENETICOS A UTILIZAR

Representación de los Individuos

- Generalidades
- Tipos
  - Cadenas binarias => 2^l soluciones en vector de longitud l
  - Vector de valores reales: incluye vector de desviacion standard
  - Elementos del problema
  - Estados Finitos
  - Arboles

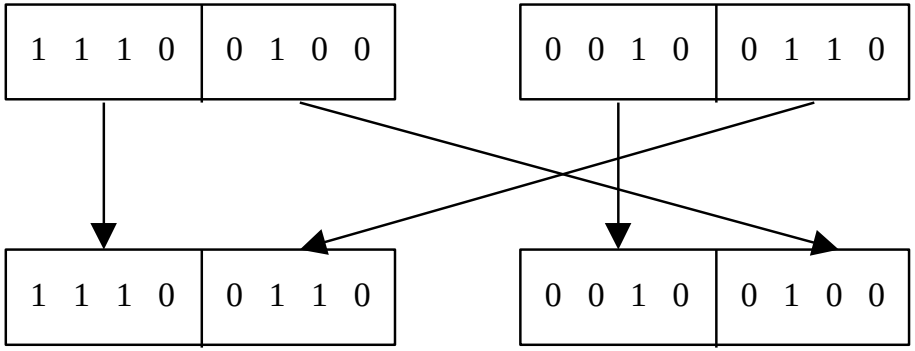
Operadores Genéticos

- Mecanismos de transformación que se utilizan para recorrer el espacio de las soluciones de un problema.

  realizan los cambios de la población durante la ejecución de un algoritmo evolutivo.

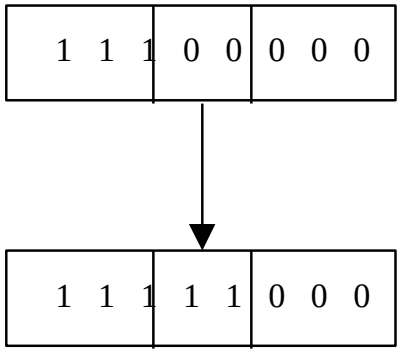
- En general, se basan en los operadores genéticos biológicos.
- Operadores genéticos clásicos:
  - Mutación
  - Cruce

Operador Cruce



- Algunos Tipos:
  - Cruce multipunto
  - Cruce segmentado

Operador de Mutación



- Tipos
  - Mutaciones sobre genes
  - Mutaciones no estacionarias
  - Mutaciones no uniformes

Operadores Avanzados

- Dominación
- Segregación
- Inversión
- Duplicación

Mecanismos de funcionamiento de los algoritmos evolutivos

- Estrategias de selección de los individuos susceptibles a reproducirse.
- Estrategia de apareamiento de los individuos en la fase de reproducción.
- Estrategia de reemplazo.

Mecanismos de Selección

- Políticas empleadas para la conformación de los progenitores.
- Tipos:
  - elitesca
  - aleatoria
  - Por sorteo
  - universal (por ruleta)
  - Por torneos

Mecanismos de Reemplazo

- Tipos:
  - Padre por hijo, o Reemplazo directo
  - Por similitud
  - Por inserción (,), p.e., peores por mejores
  - Por inclusión (+)

Estrategia de Apareamiento

- Que individuos con que individuos aparear en la fase de reproduccion

- Tipos:
  - Aleatorio,
  - Autofertilizacion,
  - etc.

ALGUNOS PARAMETROS Y VARIABLES QUE CONTROLAN EL PROCESO

- Probabilidades de uso de los operadores
- Criterios de parada
  - Por ejemplo, Número de generaciones
- Tamaño y diversidad de la población

Problema del agente viajero

"Visitar todas las ciudades importantes de Vzla. por lo menos una vez, comenzando y terminando en Mérida"

- Este problema es un "Touring Problem": requiere más información sobre el espacio de estados. Además de la ubicación del agente, un registro de las ciudades visitadas.
- El test objetivo consistiría en verificar si el agente está en Mérida y se ha visitado todas las ciudades UNA SOLA VEZ.
- El objetivo es determinar cuál es el recorrido más corto, es un problema de complejidad NP.
- Un uso de los algoritmos que lo resuelven es en la planificación de los movimientos de los taladros automáticos para circuitos impresos.

Ejemplos de Aplicación

Problema de optimización combinatoria

- El *Agente Viajero:* Dadas N ciudades, el viajero de comercio debe visitar cada ciudad una vez teniendo en cuenta que el costo total del recorrido debe ser mínimo.

G=(N, A)

N={1, ...,n}: conjunto de *n* nodos (vértices)

A={a_{ij}}: matriz de adyacencia.

d_{ij} = { ∞ Si $a_{ij} = 0$

Lij Si $a_{ij} = 1$

Lij: distancia entre las ciudades i y j.

Ejemplos de Aplicación

- Suponiendo que las ciudades son numeradas desde 1 hasta n, una solución al problema puede expresarse a través de una matriz de estado *E*

e_{ij} = { 1 Si la ciudad *j* fue la i^{esima} ciudad visitada

0 En otro caso

- La matriz E permite definir un arreglo unidimensional V de dimensión *n*;

$v_j = i$ Si la ciudad i fue la j^{esima} ciudad visitada

Ejemplos de Aplicación

- Función objetivo

$$F1 = \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n} L_{ik} e_{ij} e_{kj+1}$$

$$F2 = C\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\left(e_{ij} - n\right) + \sum_{i=1}^{n}\sum_{j=1}^{n}\left(e_{ij} - 1\right) + \sum_{j=1}^{n}\sum_{i=1}^{n}\left(e_{ij} - 1\right)\right)$$

$$FC = F1 + F2$$

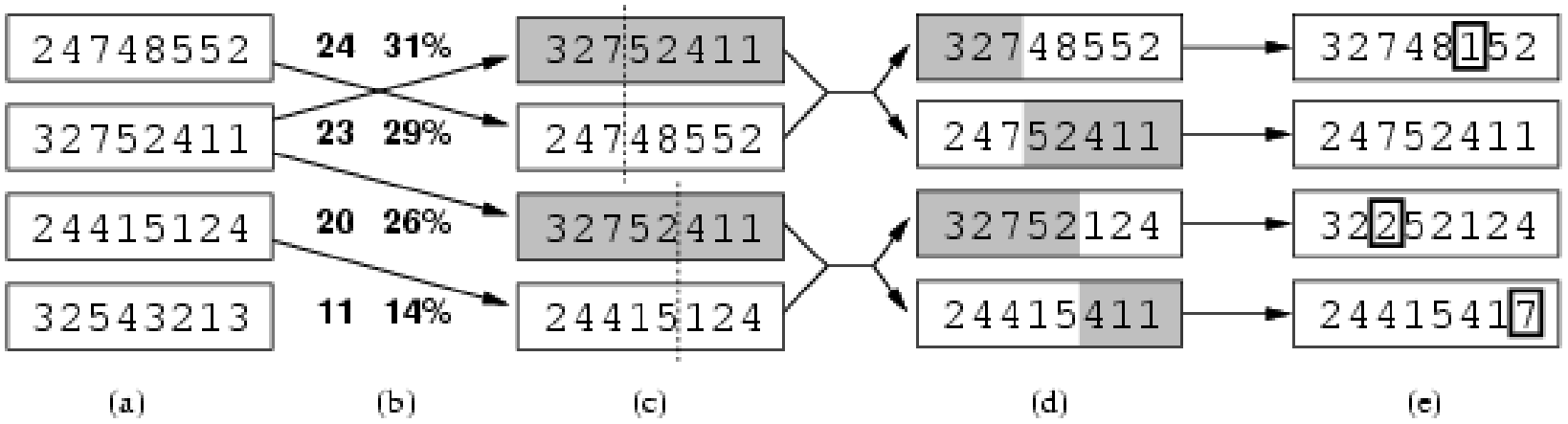
C = n * Max(L_{ik}) factor de penalización.

Ejemplos de Aplicación

- Codificación de los individuos:
  - Una solución viene dada como un recorrido que cumple con las restricciones del problema.
  - Se puede realizar por medio del arreglo unidimensional V de longitud *n*
- Función Objetivo: F1 o FC
- Operadores genéticos:

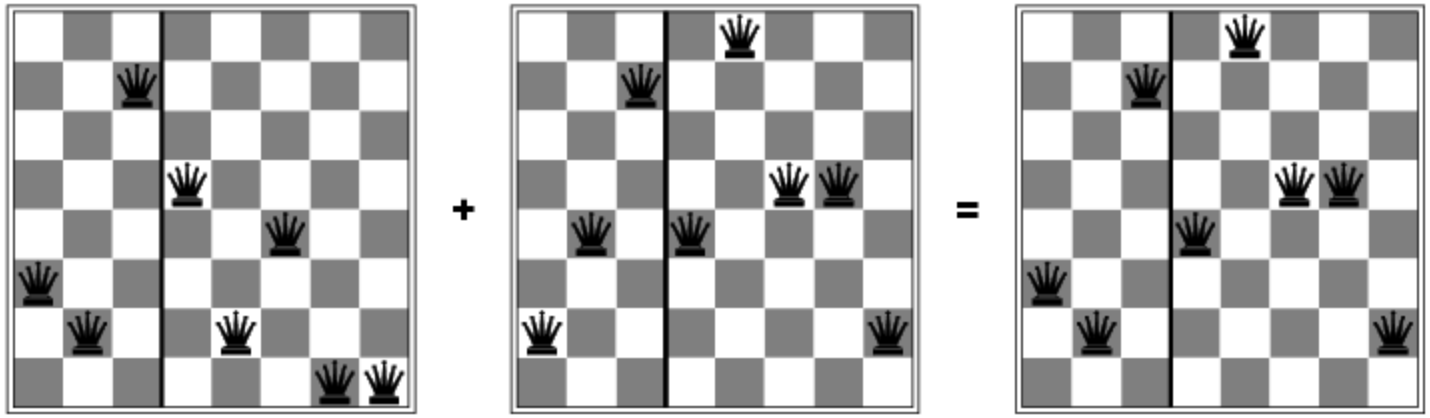
Si es F1 => Inversión.

Algoritmos Genéticos (AGs)



Población Inicial Función de Aptitud Selección Cruce Mutación

Algoritmos Genéticos (AGs)



Puedo tener operadores particulares a un problema dado

=> Cumplen con las restricciones del problema

COMPUTACION EVOLUTIVA

- OPTIMIZACION
- ADAPTACION
- MAQUINAS INTELIGENTES
- BIOLOGIA
- CONSTRUCCION DE PATRONES

Algoritmos Genéticos

Algoritmos Genéticos (AGs)

METODO ESTOCASTICO DE BUSQUEDA INFORMADA DE SOLUCIONES CUASI-OPTIMAS. SE MANTIENE UNA POBLACION QUE REPRESENTA POSIBLES SOLUCIONES, LA CUAL ES SOMETIDA A CIERTAS TRANSFORMACIONES PARA OBTENER NUEVOS CANDIDATOS, Y A UN PROCESO DE SELECCIÓN SESGADO A FAVOR DE LAS MEJORES SOLUCIONES

Algoritmos Genéticos (AGs)

- Dicha técnica es la más conocida.
- Razones:
  - Reúnen, de manera natural, todas las ideas fundamentales de la CE.
  - Poseen la mayor base teórica, la cual es sencilla y con mayores posibilidades de ampliación.
  - Requieren menor conocimiento especifico del problema, lo que les da mayor versatilidad.

Características de los AGs

- Búsqueda Informada.
- Búsqueda Basada en Población.
- Operadores Aleatorios.

Implementación de los AGs

- Seleccionar la Codificación: Si suponemos que un individuo *Y* posee *m* genes se requieren para la codificación L bits

$$L = \sum_{i=0}^{m} L_i \qquad L_i = \log_2 a_i$$

gen *i* tiene a_i posibles valores (alelos).

- Seleccionar la población inicial y su tamaño.

Lo mas variada posible.

Implementación de los AGs

- Definir criterio de parada
  - *numero de iteraciones* (generaciones).
  - *similitud de las estructuras*
  - *similitud de contenido*.
- Funcion de Evaluación

  De manera ideal, la función de evaluación coincide con el objetivo a maximizar o minimizar
- Manejar los individuos no factibles
  - *penalización*,
  - *reparación o corrección*,
  - *decodificación*

Ejemplos de Aplicación

Búsqueda en un espacio de soluciones

- Supongamos que se desea maximizar la función: $F(x) = Sen(x\pi/256)$ en el intervalo [0,255].
- *Codificación*. Como las soluciones varían entre 0 y 255, se pueden utilizar cadenas binarias de longitud 8. Por ejemplo, 00010100.

Ejemplos de Aplicación

- *Función de evaluación*. La función de evaluación es la misma función dada, y se utilizar a como función de aptitud la diferencia entre el valor máximo (1) y el valor para la función dada: g(x)=1-F(x).
- *Población inicial*.

Individuo	**x**	**F(x)**
10111101	189	0.733
11011000	216	0.471
01100011	99	0.937
11101100	236	0.243

LIMITACIONES

- FUNCION DE APTITUD SIEMPRE POSITIVA
- NUMERO DE GENERACIONES FINITO
- CODIFICACION BINARIA ES LA COMUN
- NO CONSIDERA RESTRICCIONES (se pueden incorporar en la función de costo)
- PROBLEMAS DE DIVERSIDAD: SUPERINDIVIDUOS, CONVERGENCIA PREMATURA, PRESION SELECTIVA, MECANISMOS DE ESCISION, CONTROLANDO EDADES, ETC.
- PROBLEMAS DE REPRESENTACION: COMPLETITUD, COHERENCIA, UNIFORMIDAD, ETC.

AVANCES

- FUNCIONES MULTIOBJETIVOS

  =>conjunto de Paretos
- AG PARALELOS
- META-ALGORITMOS GENETICOS

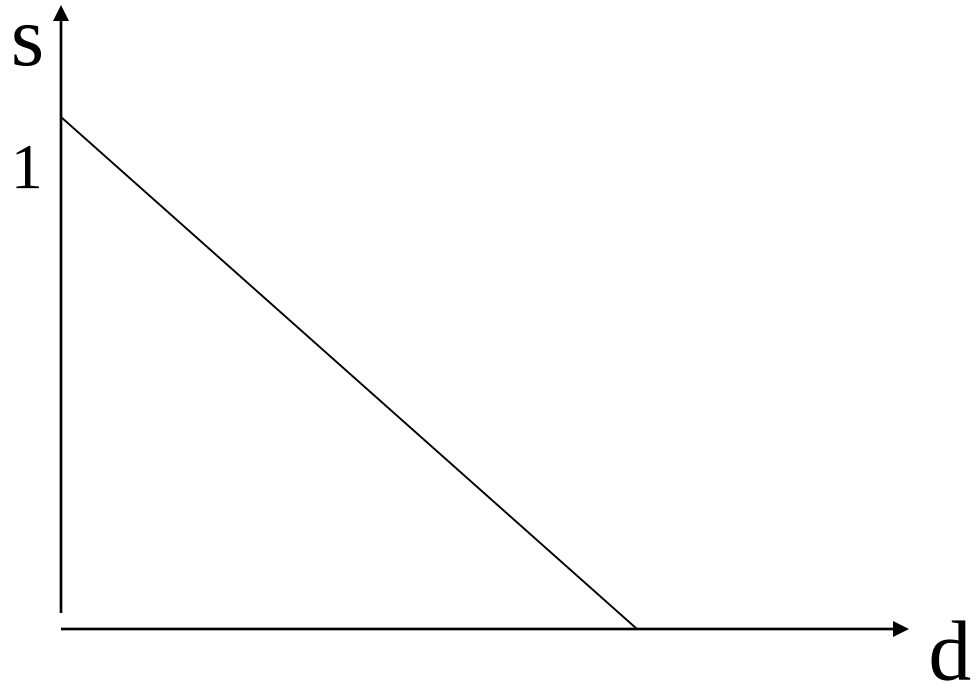
  =>APRENDIZAJE
- NICHOS Y ESPECIES

NICHOS Y ESPECIES

- Diferencia entre ambiente y organismo
- Estables subpoblaciones: especies
- Sirven a diferentes subdominios de una función: nichos
- Metafora de compartir: vecinos

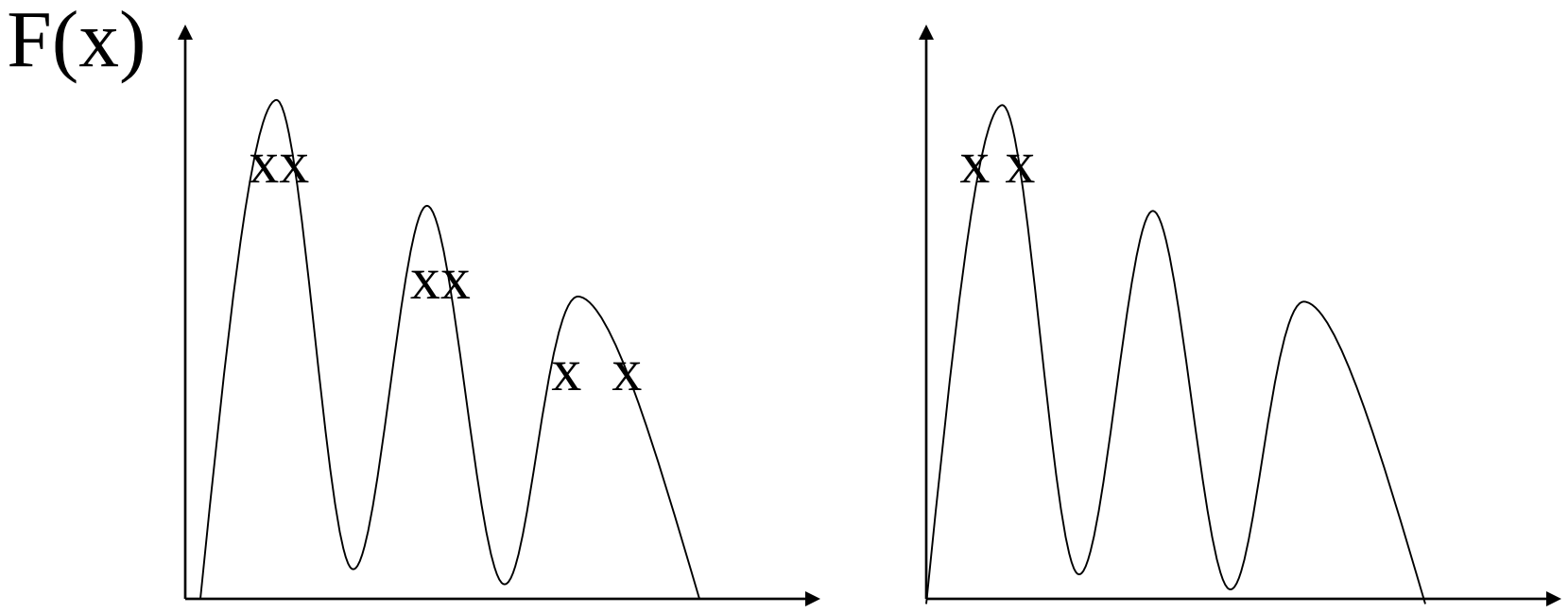
$$Fs(x_i)=F(x_i)/\sum_j s(d(x_i,x_j))$$

NICHOS Y ESPECIES



- Cruce entre individuos semejantes

NICHOS Y ESPECIES



Nicho

Especie

PROBLEMAS MULTIOBJETIVOS

- Muchos problemas de optimización se reducen a la consideración de un solo criterio, mientras que en otros problemas es necesario evaluar varios criterios.

  problemas multicriterios o multiobjetivos

- Las técnicas de optimización convencional, tales como los métodos basados en gradiente, y unos menos convencionales, tales como recocido simulado, son difíciles de extender a casos verdaderamente multiobjetivos, ya que estos no fueron diseñados para evaluar esto.

PROBLEMAS MULTIOBJETIVOS

En el mundo real, la mayor parte de los problemas tienen varios objetivos (posiblemente en conflicto entre sí) que se desea sean satisfechos de manera simultánea.

Muchos de estos problemas suelen convertirse a mono-objetivo transformando todos los objetivos originales, menos uno, en restricciones adicionales.

Hay tres tipos de situaciones que pueden presentarse en un problema multiobjetivo:

- Minimizar todas las funciones objetivo.
- Maximizar todas las funciones objetivo.
- Minimizar algunas funciones y maximizar otras.

PROBLEMAS MULTIOBJETIVOS

- Ejemplo: fabrica de cauchos que deben minimizar dos costos:
  - Número de accidentes y
  - Costo de producir cauchos

- Escenarios:

  a = (2,10)

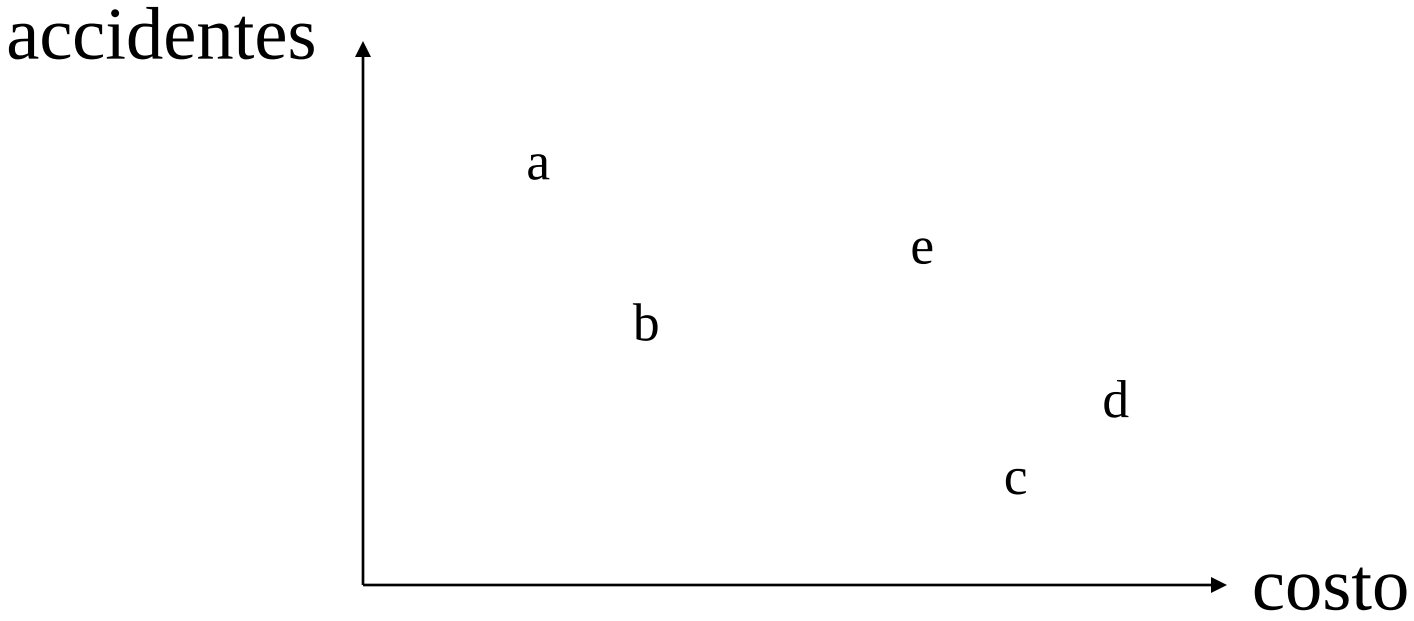
  b = (4,6)

  c = (8,4)

  d = (9,5)

  e = (7,8)

PROBLEMAS MULTIOBJETIVOS



Individuos no dominados: a, b, c!!!

Problemas de Localización Multiobjetivo

DEFINICIÓN:

Escoger un conjunto de sitios para situar varias subsidiarias que prestarán su servicio a un conjunto de clientes con el objetivo de optimizar uno o más criterios, algunos económicos o de otra índole.

- La función objetivo general es el costo de operación (fijo y variable)
- En otros contextos es necesario considerar otras funciones objetivo: distancia, cobertura, tiempo de respuesta, equidad, etc.
- Criterios en conflicto
  - Costo
  - Cobertura

Problemas de Localización Multiobjetivo

Información para el modelo

I,i : Conjunto lugares de localización de las instalaciones, m instalaciones

J,j : Conjunto e índice de los lugares de demanda, n clientes

f_i : Costo fijo de operar una instalación en el lugar i.

c_{ij} : Costo de atender toda la demanda del lugar j desde la instalación i.

d_j : Demanda del cliente j.

h_{ij} : Distancia entre i y j

Componentes del problema de Cobertura

D_{max} : Distancia máxima de cobertura.

Q_j : $\{i: h_{ij} \leq D_{max}\}$, conjunto de instalaciones que pueden atender la demanda del nodo j cumpliendo con la distancia máxima de cobertura.

Problemas de Localización Multiobjetivo (Modelo matemático)

Variables de decisión

$y_i = \begin{cases} 1 & \text{si se abre sitio i} \\ 0 & \text{de lo contrario} \end{cases}$

$x_{ij} = \begin{cases} 1 & \text{si el cliente j es atendido por sitio i} \\ 0 & \text{de lo contrario} \end{cases}$

Función objetivo y restricciones

$$\text{mín} \quad z_1 = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} x_{ij} + \sum_{i \in \mathcal{I}} f_i y_i$$

$$\text{máx} \quad z_2 = \sum_{j \in \mathcal{J}} d_j \sum_{i \in Q_j} x_{ij}$$

$$\sum_{i \in \mathcal{I}} x_{ij} = 1 \quad , \quad j \in \mathcal{J}$$

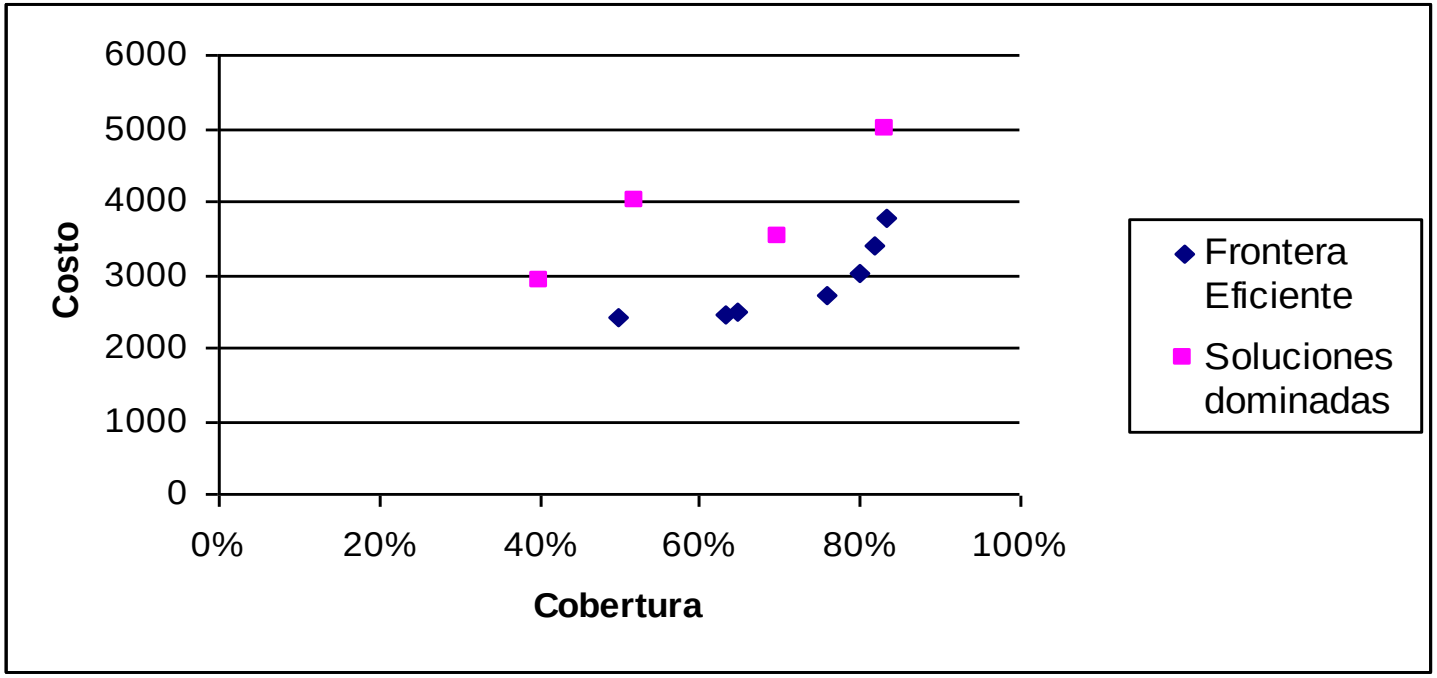
$$x_{ij} \leq y_i \quad , \quad j \in \mathcal{J}, \quad i \in \mathcal{I}$$

$$x_{ij} \in \{0,1\}, \quad j \in \mathcal{J}, \quad i \in \mathcal{I}$$

$$y_i \in \{0,1\}, \quad i \in \mathcal{I}$$

Generalidades Optimización Multiobjetivo

- Los objetivos considerados están en conflicto.
- Se busca un conjunto de soluciones eficientes, en las cuales no sea posible mejorar el valor de una de las funciones objetivo sin deteriorar el desempeño de la otra.



PROBLEMAS MULTIOBJETIVOS

TEORIA DE PARETO

- A diferencia de la optimización de un único objetivo, la solución a un problema multiobjetivo no es un único punto, sino una familia de puntos conocida como el "Conjunto Optimo de Pareto".
- Cada punto en esa superficie es óptimo en el sentido de que no pueden realizarse mejoras en un componente del vector de costo que no conduzca a la degradación en al menos uno de los componentes restantes.
- La noción de optimalidad de Pareto es sólo un primer paso hacia la solución práctica de un problema multiobjetivos; también envuelve la escogencia de una única solución compromiso del "Conjunto Optimo de Pareto" de acuerdo a alguna información de preferencia.

PROBLEMAS MULTIOBJETIVOS

TEORIA DE PARETO

Decimos que un vector de variables de decisión $x^* \in \alpha$ (α es la zona factible) es un **óptimo de Pareto** si no existe otro $\mathbf{x \in \alpha}$ **tal que** $\mathbf{f_i(x) \leq f_i(x^*)}$ $\forall \mathbf{i = 1, \ldots, k}$

x^* es un **óptimo de Pareto** si no existe ningún vector factible de variables de decisión $x^* \in \alpha$ que decremente algún criterio sin causar un incremento simultáneo en al menos un criterio.

Desafortunadamente, este concepto casi siempre produce no una solución única sino un conjunto de ellas a las que se les llama **conjunto de Pareto**.

Los vectores x^* correspondientes a las soluciones incluidas en el conjunto de óptimos de Pareto son llamados **no dominados**.

Las funciones objetivo cuyos vectores no dominados se encuentran en el conjunto de óptimos de Pareto se denominan **frente de Pareto**.

ESQUEMAS RESOLUCION PROBLEMAS MULTIOBJETIVOS

- No basadas en Pareto
  - Ordenamiento lexicográfico
  - Suma lineal de pesos
  - VEGA
  - Método ε-constraint
  - Teoría de juegos

- Basadas en Pareto
  - Jerarquización de Pareto “pura”
  - MOMGA (Multi-objective Messy Genetic Algorithm),
  - PAES (Pareto Archived Evolution Strategy),
  - PESA (Pareto Envelope-based Selection Algorithm),
  - NSGA (Nondominated Sorting Genetic Algorithm),
  - SPEA (Strength Pareto Evolutionary Algorithm).

ESQUEMAS RESOLUCION PROBLEMAS MULTIOBJETIVOS

VEGA

1. Repetir Hasta (# Iteraciones dada o Población Converge)
   1.1.- Dividir población en *n* subpoblaciones S_1,…, S_n. (hay n objetivos)
   1.2.- Evaluar una FO distinta en c/u.
   1.3.- Unir mejores individuos de las subpoblaciones Si, i = 1,...,n. (shuffling)
   1.4 Aplicar operadores genéticos.

PESA (Pareto Envelope-based Selection Algorithm),

1. Generar una inicial población
2. Evaluar la población con la función multi-objetivos
3. Crear el conjunto de individuos no-dominados con los que lo cumplan
2. Repetir Hasta (# Iteraciones dada o Población Converge)
   2.1.- Aplicar operadores genéticos sobre el conjunto de individuos no-dominados
   2.2.- Evaluar nuevos individuos con la función multi-objetivos
   2.3.- Insertar en el conjunto de individuos no-dominados a los que lo cumplan
3. Regresar el conjunto de individuos no-dominados

PROBLEMAS MULTIOBJETIVOS

ENFOQUES BASADOS EN LA DIVISIÓN DE LA POBLACIÓN

1.- Dividir población en *n* subpoblaciones S_1,…, S_n.

% se va a iterar un AG con una FO distinta en c/u.

2.- Definir vector de prioridad para las FOs.

3.- Repetir Hasta (Evaluar todas las FOs)

3.1 Para cada S_i, $\forall$ i = 1,...,*n*, usar la FO correspondiente según el vector de prioridad.

3.1.1 Iterar el AG para esa FO_i según un número *y* dado de iteraciones en cada S_i.

3.1.2 Reemplazar *X* % de individuos en cada población S_i que cumplen mal la FO actual.

PROBLEMAS MULTIOBJETIVOS

ENFOQUES BASADOS EN LA DIVISIÓN DE LA POBLACIÓN (VEGA)

1.- Dividir población en *n* subpoblaciones S_1,…, S_n.

2.- Iterar un AG con una FO distinta en c/u.

3.- Unir mejores individuos de las subpoblaciones Si, i = 1,...,n.

4.- Repetir Hasta (# Iteraciones dada o Población Converge)

4.1 Seleccionar FO_i según probabilidad de escogencia.

4.2 Disminuir probabilidad de escogencia de la FO_i seleccionada.

4.3 Aplicar AG a población global usando FO_i seleccionada. Hacer un reemplazo parcial.

PROBLEMAS MULTIOBJETIVOS

ENFOQUES BASADOS EN POBLACIÓN TOTAL

1.- Inicio.

2.- Repetir Hasta (Que c/FO sea escogida un mínimo número de veces ó población converja).

2.1 Escoger una FO (aleatoriamente o cíclicamente).

2.2 Iterar el AG en la población usando la FO escogida.

ENFOQUES BASADOS EN LA TEORIA DE PARETO

Enfoque en dos fases: en la primera se clasifican los individuos en no-inferiores o no (no-dominados o dominados), para cada una de las FOs que conforman la FMO. En la segunda, se utilizan los individuos no-dominados como los padrotes para la fase de reproducción.

Fase de Clasificación :

1.- Generar población inicial.

2.- Evaluar los individuos para c/FO y establecer su clasificación en c/FO.

3.- Seleccionar individuos no-dominados (Según c/u de las FOs).

4.- Asignar rango 1 a los individuos no-dominados y remover de la disputa, luego se encuentra un nuevo conjunto de individuos no-dominados con rango 2, y así sucesivamente. Esto se hace para c/FOs.

ENFOQUES BASADOS EN LA TEORIA DE PARETO

Fase de Optimización :

1.- Seleccionar los individuos para reproducción por algún mecanismo clásico (ruleta, elitesco).

2.- Reproducir con operadores clásicos, usando los individuos no-dominados como los padrotes.

Macro Algoritmo:

1.- Generar población.

2.- Repetir Hasta que población generada sea igual a la anterior o sea homogénea.

2.1.- Fase de Clasificación: Agrupar individuos en dominados y no-dominados para c/FO.

2.2.- Fase de Optimización: Generar nueva población según un AG clásico.

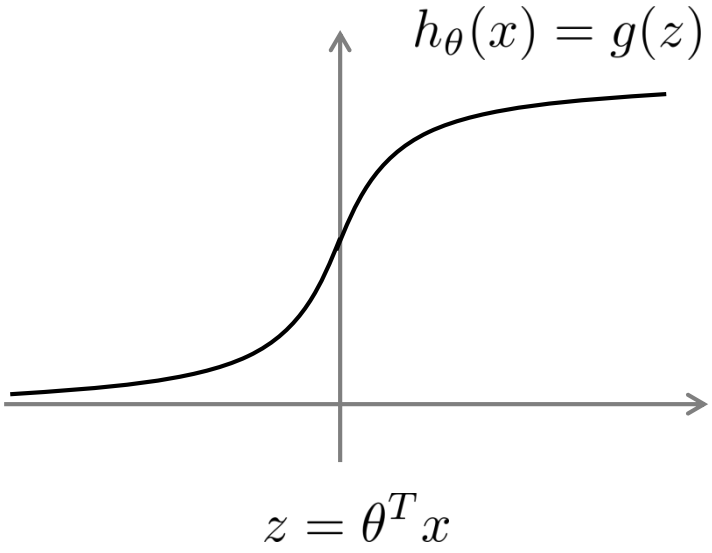
Maquinas de Soporte Vectorial

Maquinas de Soporte Vectorial

- Son un conjunto de algoritmos de aprendizaje supervisado
- Las MVS han sido desarrolladas como una técnica robusta para clasificación y regresión aplicado a grandes conjuntos de datos complejos con ruido
- La MVS mapean los puntos de entrada a un espacio de características de una dimensión mayor, para luego encontrar el hiperplano o conjunto de hiperplanos en ese espacio que los separe y maximice el margen entre las clases.
- Usa una función núcleo o kernel para ello, con una gran capacidad de generalización.
- La diferencia más notable con respecto a otros algoritmos de aprendizaje, es la aplicación de un nuevo principio inductivo, que busca la minimización del riesgo estructural,

Regresión lógica

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$



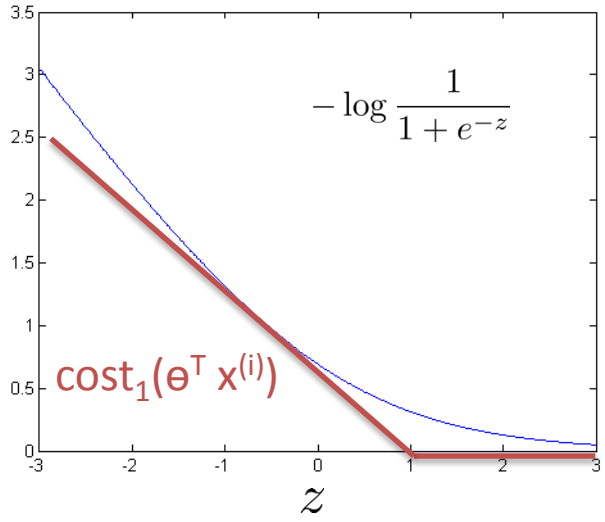
If $y = 1$, queremos $h_\theta(x) \approx 1$, $\theta^T x \gg 0$

If $y = 0$, queremos $h_\theta(x) \approx 0$, $\theta^T x \ll 0$

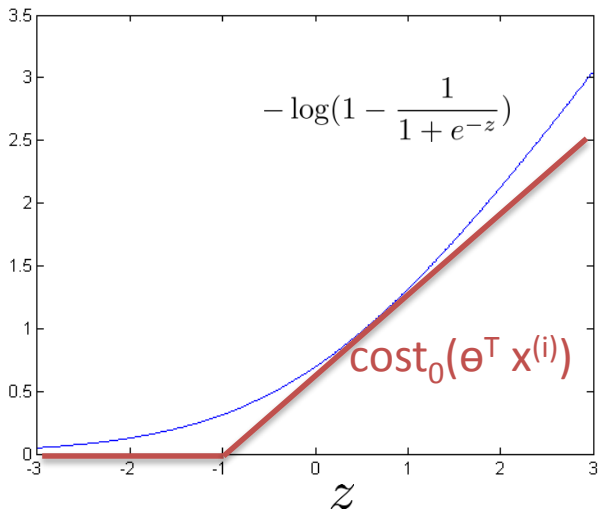
Regresión lógica

$$\text{Costo} = -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y)\log(1 - \frac{1}{1+e^{-\theta^T x}})$$

Si $y = 1$ (para $\theta^T x \gg 0$):



Si $y = 0$ (para $\theta^T x \ll 0$):



Máquina de soporte vectorial

Regresión Lógica:

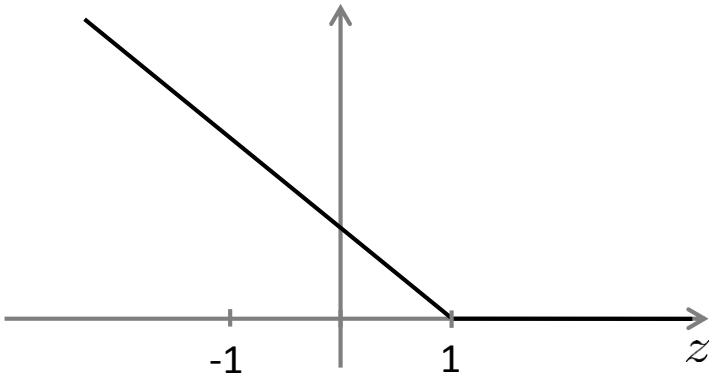
$$\min_{\theta} \frac{1}{m} \left[\sum_{i=1}^{m} y^{(i)} \left(-\log h_\theta(x^{(i)}) \right) + (1 - y^{(i)}) \left((-\log(1 - h_\theta(x^{(i)})) \right) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

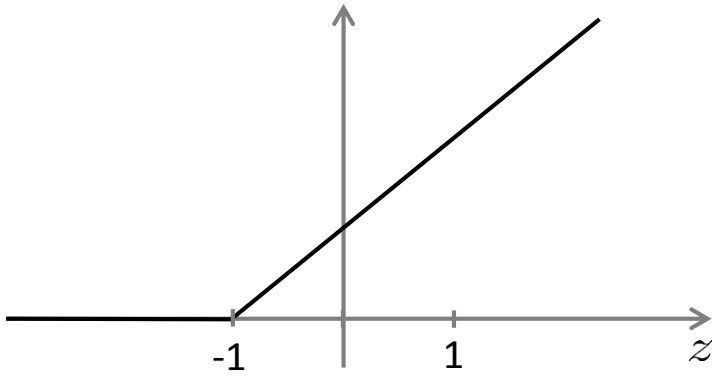
MSV:

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

Máquina de soporte vectorial

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$



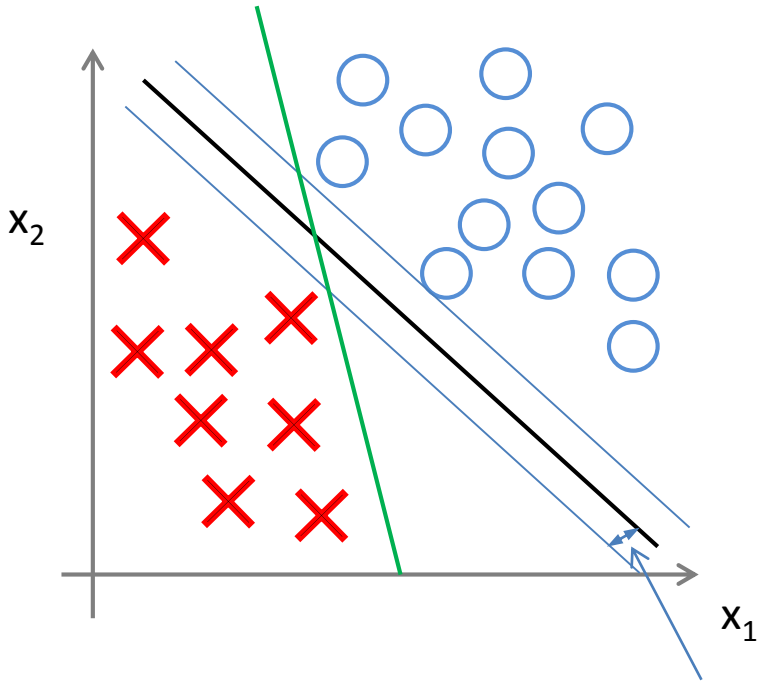


Si $y = 1$, queremos $\theta^T x \geq 1$ (no solo ≥ 0)

Si $y = 0$, queremos $\theta^T x \leq -1$ (no solo < 0)

Barrera de decision SVM

SVM Barrera de decisión : Caso lineal

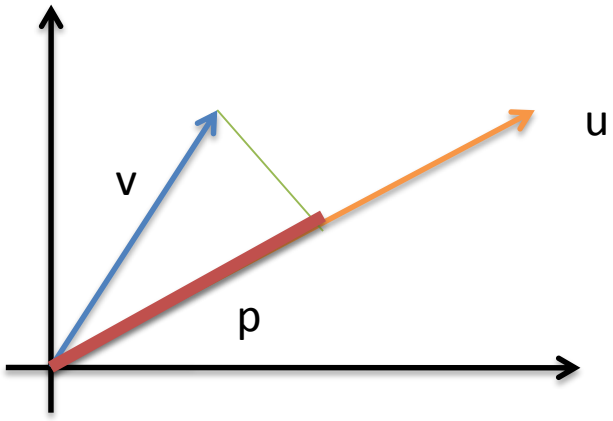


Margen de clasificación

Barrera de decisión no lineal

Matemáticas detrás de SVM

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$



p = proyección de v en u

$(U^T V) = p \cdot ||u||$

$||u|| = \sqrt{(u_{(1)}^2 + u_{(2)}^2)}$

Barrera de decisión no lineal

Matemáticas detrás de SVM

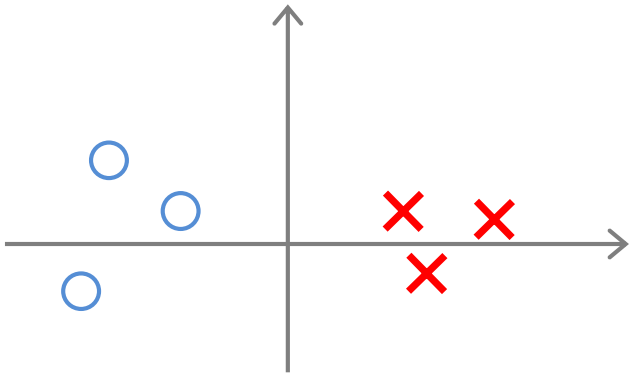
$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

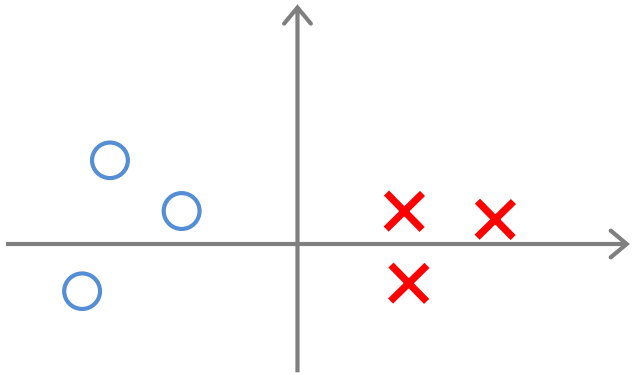
s.t. $p^{(i)} \cdot \|\theta\| \geq 1 \quad$ if $y^{(i)} = 1$

$p^{(i)} \cdot \|\theta\| \leq -1 \quad$ if $y^{(i)} = 1$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplification: $\theta_0 = 0$





Barrera de decisión no lineal

Matemáticas detrás de SVM

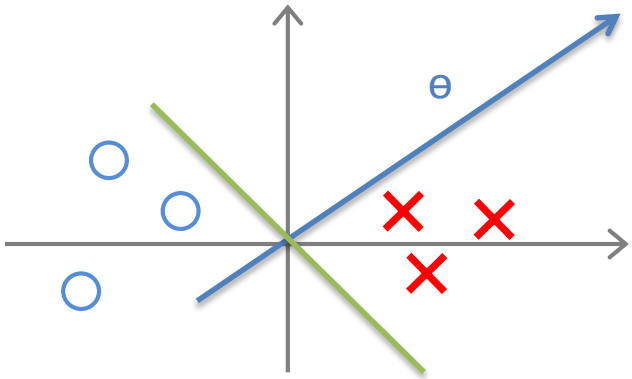
$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

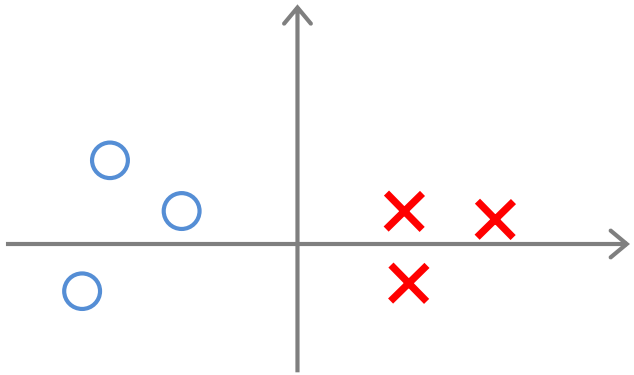
$$\text{s.t.} \quad p^{(i)} \cdot \|\theta\| \geq 1 \qquad \text{if } y^{(i)} = 1$$

$$p^{(i)} \cdot \|\theta\| \leq -1 \qquad \text{if } y^{(i)} = 1$$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplification: $\theta_0 = 0$





Barrera de decisión no lineal

Matemáticas detrás de SVM

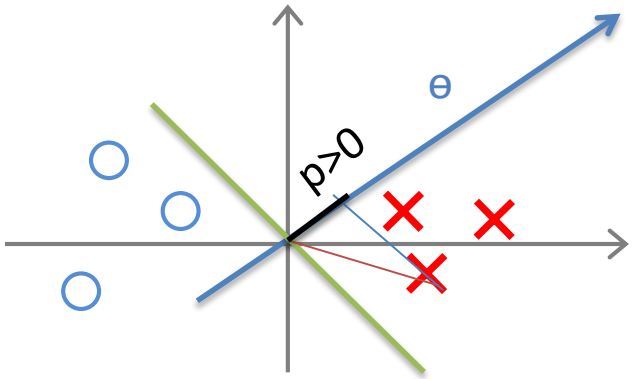
$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

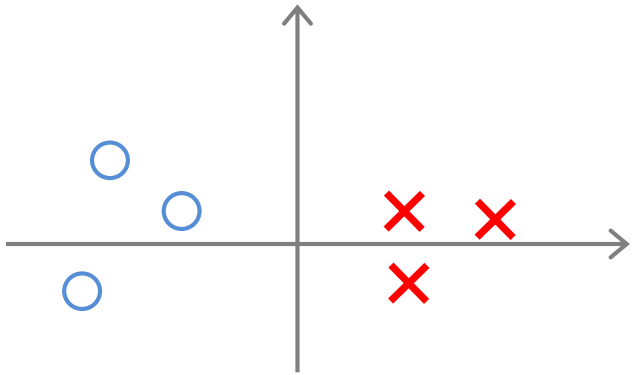
$$\text{s.t.} \quad p^{(i)} \cdot \|\theta\| \geq 1 \qquad \text{if } y^{(i)} = 1$$

$$p^{(i)} \cdot \|\theta\| \leq -1 \qquad \text{if } y^{(i)} = 1$$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplification: $\theta_0 = 0$





Barrera de decisión no lineal

Matemáticas detrás de SVM

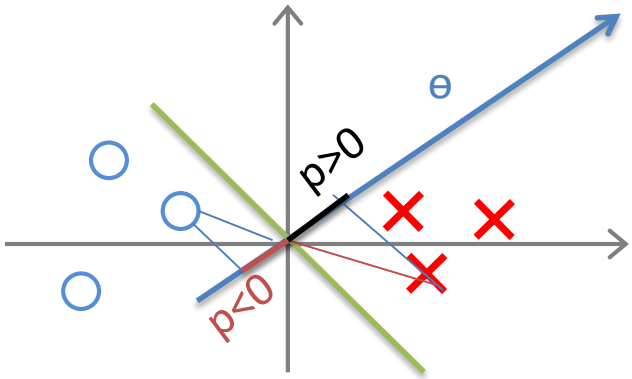
$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

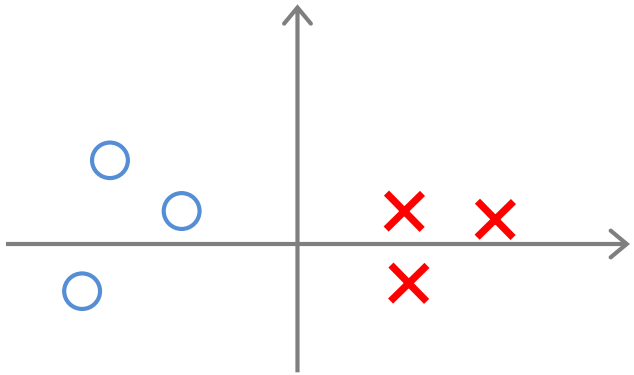
$$\text{s.t.} \quad p^{(i)} \cdot \|\theta\| \geq 1 \qquad \text{if } y^{(i)} = 1$$

$$p^{(i)} \cdot \|\theta\| \leq -1 \qquad \text{if } y^{(i)} = 1$$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplification: $\theta_0 = 0$





Barrera de decisión no lineal

Matemáticas detrás de SVM

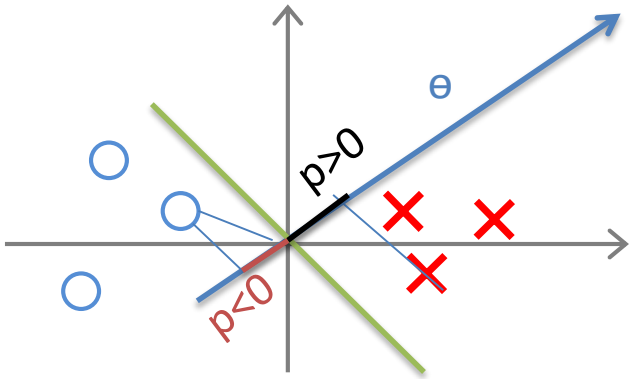
$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

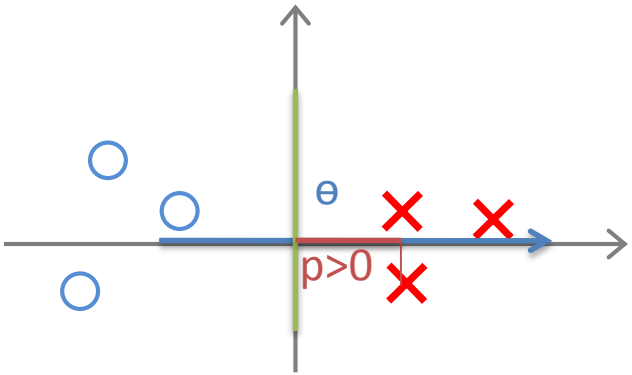
$$\text{s.t.} \quad p^{(i)} \cdot \|\theta\| \geq 1 \qquad \text{if } y^{(i)} = 1$$

$$p^{(i)} \cdot \|\theta\| \leq -1 \qquad \text{if } y^{(i)} = 1$$

where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplification: $\theta_0 = 0$





Barrera de decisión no lineal

Matemáticas detrás de SVM

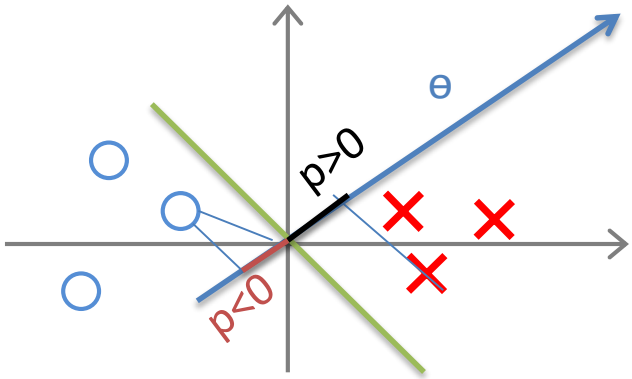
$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

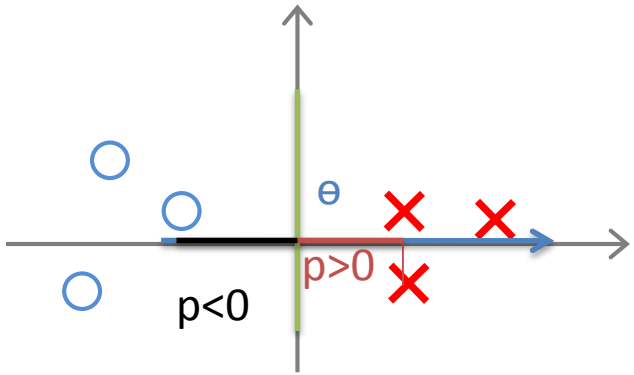
$$\text{s.t.} \quad p^{(i)} \cdot \|\theta\| \geq 1 \qquad \text{if } y^{(i)} = 1$$

$$p^{(i)} \cdot \|\theta\| \leq -1 \qquad \text{if } y^{(i)} = 1$$

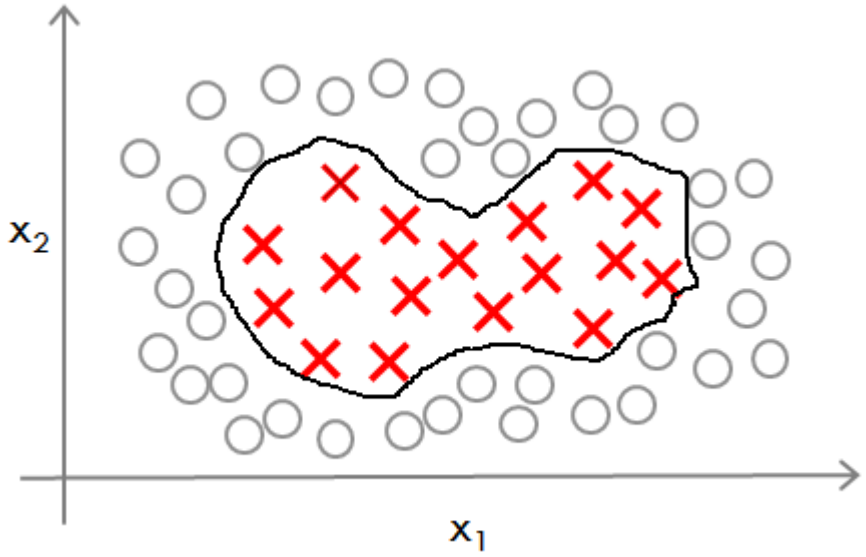
where $p^{(i)}$ is the projection of $x^{(i)}$ onto the vector θ.

Simplification: $\theta_0 = 0$





Barrera de decisión no lineal



Predice $y = 1$ Si

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots \geq 0$$

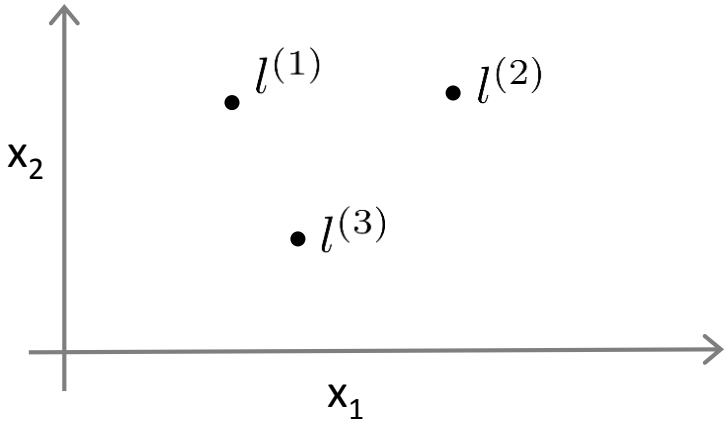
$$f_1 = x_1$$
$$f_2 = x_2$$
$$f_3 = x_1 x_2$$

$$\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \cdots \geq 0$$

Hay una diferente/mejor escogencia de las características $f_1, f_2, f_3, \ldots$?

Barrera de decisión no lineal

Kernel



Dado x, calcular las nuevas características dependiendo de la proximidad con los puntos de referencia $l^{(1)}, l^{(2)}, l^{(3)}$

$$f_1 = \text{similarity}(x, l^{(1)}) = exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

$$f_2 = \text{similarity}(x, l^{(2)}) = exp\left(-\frac{\|x - l^{(2)}\|^2}{2\sigma^2}\right)$$

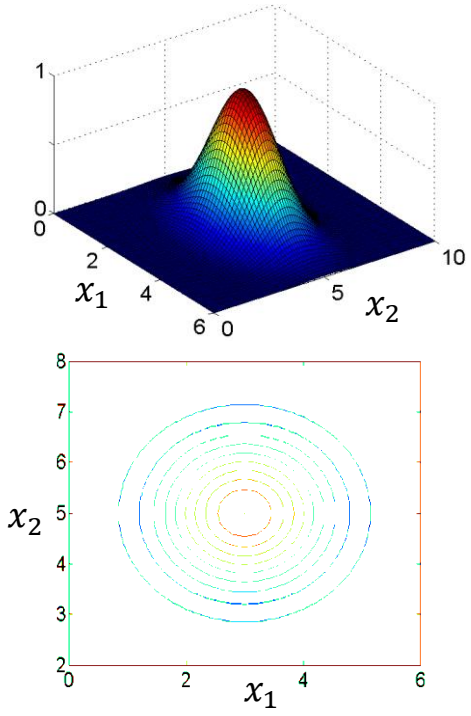
$\ldots$

Gaussian Kernel

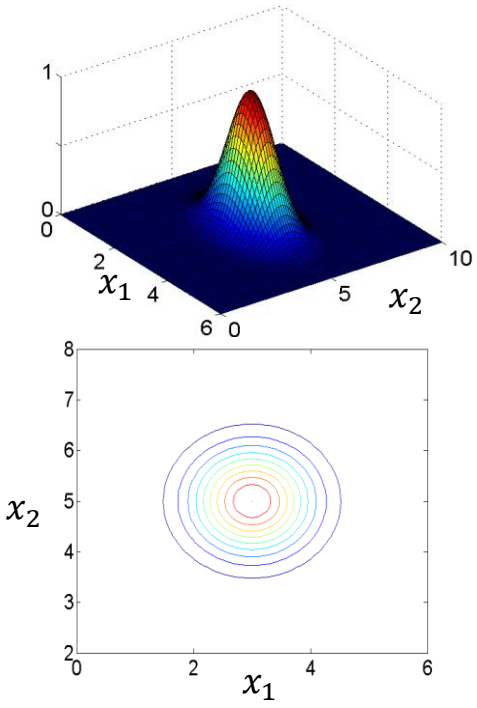
Ejemplo:

$$l^{(1)} = \begin{bmatrix} 3 \\ 5 \end{bmatrix}, \quad f_1 = \exp\left(-\frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$$

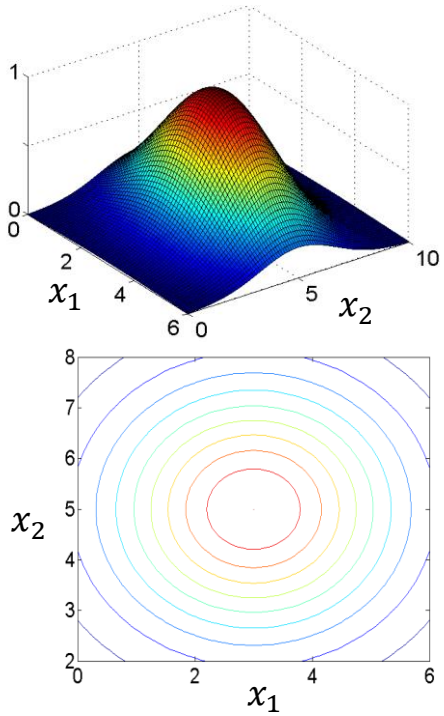
$\sigma^2 = 1$



$\sigma^2 = 0.5$

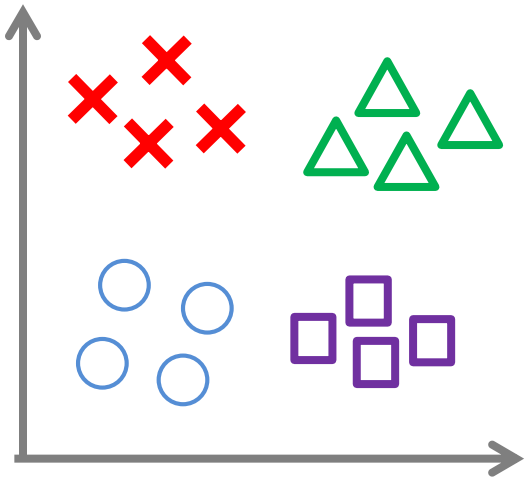


$\sigma^2 = 3$



Usar un SVM

Clasificación multiclase



$$y \in \{1, 2, 3, \ldots, K\}$$

Muchos paquetes de SVM ya tienen funcionalidades para realizar clasificación multi-clase.

Otra manera, use el método de one-vs.-all. (Entrenar K SVMs, uno para distinguir $y = i$ del resto, para $i = 1, 2, \ldots, K$), obtener $\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$

Escoger la clase i con el valor mas grande de $(\theta^{(i)})^T x$