

## 2. RECONOCIMIENTO DE PATRONES

### 2.1 Generalidades

- Reconocimiento → función básica de los organismos vivos.
  - HUMANO: Estimación del parecido relativo entre datos de entrada y pobl. conocidas.
  - AUTOMÁTICO: Clasificación de datos de entrada entre pobl. mediante la búsqueda de características o atributos invariantes entre los miembros de cada población.
- Patrón → descripción de un objeto definida como una relación entre sus características
- En la mayoría de los casos el objetivo es la clasificación.

Tarea de clasificación	Datos de entrada	Salida
Detección de spam	Correo electrónico	Spam/No spam
Reconocimiento de caracteres	Imagen	Código del carácter
Reconocimiento del audio	Ondas acústicas	Palabras
Reconocimiento de voz	Ondas acústicas	Id persona
Predicciones bursátiles	Datos de mercado	Alza/Baja de acciones
Reconocimiento facial	Imagen	Id persona
Diagnóstico médico	Síntomas e historia clínica	Diagnóstico

#### Términos relacionados:

- Clasificación - Discriminación - Categorización/Predicción
- Regresión - Aproximación/Predicción
- Aprendizaje automático (machine learning)

#### Tipos de objetos reconocidos:

- Objetos concretos (reconocimiento perceptual) → como por ejemplo una forma (patrón espacial) o una secuencia (patrón temporal).
- Objetos abstractos (reconocimiento conceptual) → como un viejo argumento o la solución de un problema.

#### Herramienta: Scikit-learn

Es una librería de aprendizaje automático para Python. Es probablemente la más utilizada. Salvo deep learning, provee casi todo lo necesario para un proyecto de aprendizaje automático. Se suele complementar con Pandas y librerías para graficar.

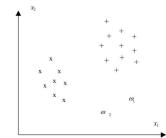
#### Problemas a resolver (etapas)

##### 1. SENSADO

**Obtención de datos:** Es el acto de medir u obtener los valores de ciertos atributos del objeto a ser reconocido. Otros atributos frecuentemente usados en distintas áreas son: Intensidades lumínicas (imágenes), Presión sonora (audio), Texto (mails, tweets, artículos), Microarrays de expresión génica.



**Representación de datos:** Los datos censados se representan mediante un vector  $x = (x_1, x_2, \dots, x_n)$ , donde  $x_i$  es el valor del atributo  $i$ . Ej: en una imagen  $\rightarrow x_i =$  intensidad de píxel  $i$ , audio  $\rightarrow x_i =$  energía en el tiempo  $i$ .



- **Variables cuantitativas:** Se representan como números del tipo que sea necesario. En una imagen, las intensidades de los píxeles se pueden representar con enteros.
- **Variables categóricas:** Pueden tomar solo un valor dentro de un conjunto limitado de valores. Pueden ser ordinales (existe una escala) o nominales (no tienen un orden).

Para variables categóricas donde no existe una relación de orden, la codificación mediante enteros no suele ser adecuada. En estos casos se suele aplicar la codificación **One-Hot**, es un método en el cual se agrega una nueva variable binaria para cada valor de categoría posible. Permite etiquetar a qué clase pertenecen los datos. Se asigna 0 a toda la dimensión, excepto 1 para la clase a la que pertenecen los datos.

Otra transformación útil para la representación de los datos es el **escalado**. Es deseable que los valores de todas las variables se encuentren en el mismo rango. Se suele aplicar una ecuación matemática para convertir valores entre 0 y 1. Luego para mostrar aplico nuevamente la ecuación.

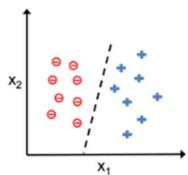
## 2. EXTRACCIÓN DE CARACTERÍSTICAS

Lo que se logra con esto es reducir la dimensión de los patrones pero con la posibilidad de perder un bajo porcentaje de la información contenida en ellos.

Dentro de esta etapa podemos encontrar las siguientes tareas:

- Creación de nuevas características
- Eliminación de características
- Reducción de la dimensión
- Otras transformaciones

## 3. CLASIFICACIÓN



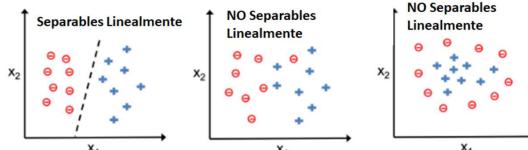
**Determinar a qué clase pertenece cada vector de entrada.** Para lograr esto, se requiere diseñar una mecanismo que, dado un conjunto de patrones de los cuales no se conoce a priori la pertenencia a la clase de cada uno de ellos, permita determinar a qué clase pertenece cada patrón.

**Generación de límites de decisión entre regiones.**  $\rightarrow$  Funciones de decisión.

Se deben generar tantas funciones de decisión como clases haya. Tomando un dato como entrada, se evalúa en las  $x$  funciones, y el valor más alto es la clase a la que pertenece. Cuando 2 funciones dan el mismo valor, el punto forma parte del límite entre las regiones.

### Tipos de problemas reales

- Linealmente separables
- No linealmente separables



Si el sistema de reconocimiento puede autoajustar ciertos coeficientes internos que definen las funciones discriminantes estaremos en presencia de un sistema adaptivo o con capacidad de aprendizaje.

## *Clasificación de los Métodos*

Definiendo a una clase como una **categoría** determinada por algunos atributos comunes y un patrón como la descripción de cualquier miembro de una categoría que representa a una clase de patrones, decimos que la teoría del Reconocimiento de Patrones se ocupa de buscar la solución al problema general de reconocer miembros de una clase dada en un conjunto que contienen elementos de muchas clases diferentes.

- Heurísticos: basados en la intuición y experiencia.
- Matemáticos:
  - Determinístico: de base estadística pero esta no está explicitada. Clasificadores entrenables.
  - Estadístico: de base estadística explícita. Ej: clasificador de Bayes.
- Sintácticos: basados en relaciones entre los objetos. Expresan una gramática. Útiles cuando los patrones no pueden ser apropiadamente descritos por mediciones (numéricamente).

## *Resumiendo las etapas*

El resultado de las etapas de sensado y extracción de características es un conjunto de vectores de características. Cada vector tiene la forma:  $\mathbf{x}=[x_1, x_2, \dots, x_n]$  donde  $x_i$  es el valor de la característica  $i$ .

- $u \rightarrow$  es un escalar.
- $\mathbf{u} \rightarrow$  (en negrita) es un vector formado por escalares  $u_i$
- $U \rightarrow$  es una matriz formada por escalares  $u_{ij}$

Después de obtener los vectores de características se realiza una de las siguientes tareas:

- **Clasificación.** Asignar cada vector de características a una clase. Por ejemplo, para detectar qué carácter aparece en una imagen.
- **Predicción o regresión.** Predecir un valor (contínuo) para cada vector de características. Por ejemplo, para calcular el valor futuro de ciertas acciones bursátiles.

*Los métodos de aprendizaje automático que vemos son clasificadores, pero con un mínimo cambio es posible adaptarlo para hacer predicciones.*

## *Entrenamiento*

Es el proceso de ajuste de los parámetros internos del método (de la función matemática) para determinar la salida.

Existen dos tipos:

- **Entrenamiento supervisado:** Los parámetros se ajustan de forma tal que la salida del clasificador se aproxime a una salida esperada conocida.
- **Entrenamiento no supervisado:** Los parámetros se ajustan para encontrar relaciones o agrupaciones entre los valores de las características.
  - **Agrupamiento:** trata de encontrar una estructura o patrón en una colección de datos no categorizados. Procesan los datos y encuentran grupos o clústeres naturales si existen en los datos.
  - **K-means:** agrupa objetos en  $k$  grupos basándose en sus características. Minimiza la suma de distancias entre cada objeto y el centroide de su grupo o cluster.

## 2.2 La función de decisión

El clasificador lineal pertenece al grupo de métodos de aprendizaje supervisado, es necesario contar con un conjunto de vectores previamente clasificados para entrenarlo. Este conjunto de vectores se llama **vectores de entrenamiento**. El clasificador lineal utiliza una función de decisión lineal  $d(x)$ . En el caso más simple, con solo dos clases de salida, cuando  $d(x)>0$ ,  $x$  pertenece una de las clases conocida, en caso contrario, a la otra.

### Función de Decisión Lineal

Si las entradas de un clasificador lineal tienen la forma  $x=[x_1, x_2, \dots, x_n]$ , la función de decisión se define como:  $d(x) = w_1x_1 + w_2x_2 + \dots + w_nx_n + w_{n+1}$

O con vector de entradas aumentado,  $x=[x_1, x_2, \dots, x_n, 1]$ :  $d(x) = w_1x_1 + w_2x_2 + \dots + w_nx_n + x_{n+1}w_{n+1}$

Como la misma cantidad es agregada a todos los patrones de todas las clases en juego, las propiedades geométricas entre las mismas se mantienen.

$w_1$
$w_2$
$w_3$
$w_4$

$X_1^1   X_2^1   X_3^1   1$	$d^1$
$X_1^2   X_2^2   X_3^2   1$	$d^2$
$X_1^3   X_2^3   X_3^3   1$	$d^3$
$X_1^4   X_2^4   X_3^4   1$	$d^4$
$X_1^5   X_2^5   X_3^5   1$	$d^5$

### Cálculo con vectores

El vector de entradas  $x$  es un vector fila. Si representamos el conjunto de coeficientes con un vector columna  $w=[w_1, w_2, \dots, w_{n+1}]^T$ , entonces  $d(x)=\mathbf{x}w$ . Notar que la salida de  $d(x)$  es un escalar.

$X_1^1   X_2^1   X_3^1   1$	$d^1$
$X_1^2   X_2^2   X_3^2   1$	$d^2$
$X_1^3   X_2^3   X_3^3   1$	$d^3$
$X_1^4   X_2^4   X_3^4   1$	$d^4$
$X_1^5   X_2^5   X_3^5   1$	$d^5$

Es común calcular la f. lineal al mismo tiempo para todos los vectores de entrada. En ese caso,  $\mathbf{d}(X)=\mathbf{X}w$ , donde  $X$  es la matriz cuyas filas son los vectores de entrada y el resultado de  $d(x)$  es un vector columna.

$X_1^1   X_2^1   X_3^1   1$	$y^1$
$X_1^2   X_2^2   X_3^2   1$	$y^2$
$X_1^3   X_2^3   X_3^3   1$	$y^3$
$X_1^4   X_2^4   X_3^4   1$	$y^4$
$X_1^5   X_2^5   X_3^5   1$	$y^5$

Para entrenar el modelo es necesario conocer la clase a la que pertenece cada uno de los vectores  $x$ . Supongamos que estas clases son  $c_1$  y  $c_2$ . Se desea que  $d(x)>0$  para  $c_1$  y

$d(x)<0$  para  $c_2$ , entonces se crea una salida esperada  $y$  para cada vector, donde  $y$  es un escalar, pero si se contempla la salida esperada para todos los vectores de entrada, se obtiene el vector columna  $y$ .

Al igual que en la regresión lineal, se buscan los coeficientes que minimizan el valor esperado del error. Para hallar el vector  $w$  óptimo se parte de la aproximación del error cuadrático medio:

$$S^2(w) = \frac{\sum_{i=1}^m (y_i - \sum_{j=1}^{n+1} w_j x_{ji})^2}{m} = E(|y - \mathbf{X}w|^2)$$

donde,  $m$ : cantidad de vectores de entrenamiento;  $n$ : cantidad de características.

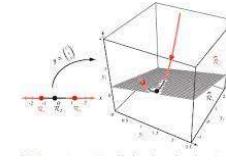
Haciendo  $\frac{\partial S^2(w)}{\partial w} = 0$ , se llega a la siguiente expresión para el cálculo de los coeficientes (para vectores de entrada definidos como filas):  $w = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T y$  donde,  $w$  : vector de pesos;  $X$  : matriz de entradas;  $y$  : vector de salida.  $w$  es la solución del sistema lineal determinado por todos los patrones de ambas clases.

La desventaja más importante es que solo puede clasificar clases linealmente separables.

La ecuación  $g(x) = 0$  define la superficie de decisión que separa puntos asignados a la categoría 1 de puntos asignados a la categoría 2. Cuando  $g(x)$  es lineal, la superficie de decisión es un **hiperplano**.  $r$  es la distancia algebraica de el punto  $x$  al hiperplano  $H$ , (la mínima distancia de un punto fijo a cualquier punto de un plano).

Partimos de un plano con 2 variables  $(x,y)$ . Para clasificarlos usamos una función  $f(x)$  que nos da un valor para cada punto de  $x_1, x_2$ . Estos valores se diagraman en una tercera dimensión que es la de la función de decisión.

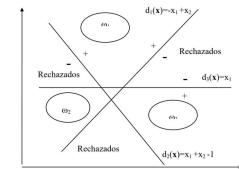
Conforme los puntos van tomando valores positivos y negativos sobre esta dimensión, se los puede clasificar (ya que estarán por encima o debajo del hiperplano).



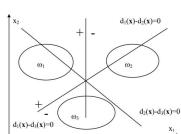
## Clasificación multiclas

### Uno contra todos (Piecewise)

- Cada clase tiene su propia función de decisión.
- Cada función se entrena para separar los vectores propios del resto.
- La salida del clasificador queda definida por la función de decisión de mayor valor para un vector determinado.



### Por pares (Pairwise)



## Funciones de Decisión Generalizadas

La función de decisión lineal para un vector de entradas  $x=[x_1, x_2, \dots, x_n]$ , definida como

$d(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_nx_n + w_{n+1}$  solo puede resolver problemas de clasificación donde las **clases son linealmente separables**. Se extiende el modelo de clasificador lineal para poder resolver problemas más complejos y difíciles.

La nueva función de decisión será:  $d(\mathbf{x}) = w_1f_1(\mathbf{x}) + w_2f_2(\mathbf{x}) + \dots + w_mf_m(\mathbf{x}) + w_{m+1}$

- Ya no se multiplica cada característica, sino por  $f_i(x)$ , que es una función que forma como parámetros el vector de características completo
- Tiene  $m$  términos, no  $n$ . No hay correspondencia con cada término como en la función lineal.
- Cada  $f_m(x)$  no tiene parámetros modificables, solo se pueden modificar los coeficientes que multiplican las  $f_m(x)$ .

El cálculo de los coeficientes  $w_i$ . Para eso es necesario una transformación en el vector de entradas.

Se define un **nuevo vector**  $\mathbf{x}^*$  cuyos componentes son las funciones:  $\mathbf{x}^* = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})]$

O, si se utiliza en vector de entradas aumentado:  $\mathbf{x}^* = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x}), 1]$

Una vez calculadas las funciones  $f_i(x)$ , el cálculo de los coeficientes es el mismo que en la función de decisión lineal. Es un espacio distinto, con nuevas características, este espacio es linealmente separable.

## Clasificador polinomial

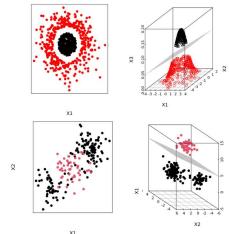
Es un caso particular de la función de decisión generalizada. El caso más simple se da para  $x^2$ , donde la función de decisión toma la siguiente forma (para un vector de entradas de dimensión 2):

$$d(\mathbf{x}) = w_{11}x_1^2 + w_{12}x_1x_2 + w_{22}x_2^2 + w_1x_1 + w_2x_2 + w_3$$

La que podría transformarse a forma lineal como:  $d(\mathbf{x}^*) = \mathbf{w}\mathbf{x}^*$  donde

$$\mathbf{x}^* = [x_1^2, x_1x_2, x_2^2, x_1, x_2, 1]$$

$$\mathbf{w} = [w_{11}, w_{12}, w_{22}, w_1, w_2, w_3]$$



- Al aumentar la cantidad de características del clasificador polinomial, se hace muy complejo y no se puede entrenar de forma directa.

## Máquinas de vectores de soporte (VSM)

- También llamados clasificadores de **margen óptimo**.
- Se consideran sólo los vectores de entrenamiento próximos a la frontera entre las clases.
  - Los valores extremos en el cálculo de la  $d(\mathbf{x})$  incorporan un sesgo a la función de decisión, por eso solo se utilizan los más próximos.
- Se maximiza la **mínima distancia** entre las clases y el hiperplano discriminante.

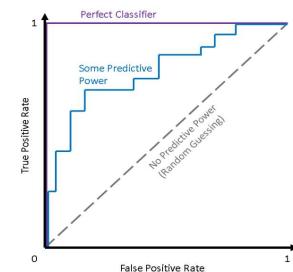
## Evaluación de modelos: Matriz de confusión

Luego de resolver con un clasificador, es importante evaluarlo, qué tan bueno es. Se arma una matriz de confusión: valores predichos en columnas y reales en filas. Se le pueden calcular métricas:

	Valor real		
Valor predicción	Verdaderos positivos	Falsos positivos	P*
	Falsos negativos	Verdaderos negativos	
Total			N*

- Factor de clasificación →  $FC = \frac{a}{b}$  donde a= Suma de elementos diagonales de M (VP+VN) y b= Suma de todos los elementos de M (VP+VP+FP+FN), porcentaje de casos bien clasif.
- Sensibilidad - Razón de Verdaderos Positivos (VPR) →  $VPR = \frac{VP}{P} = \frac{VP}{VP + FN}$  proporción de casos positivos que fueron correctamente identificadas
- Exactitud (Accuracy) →  $ACC = \frac{VP + VN}{P + N}$  lo cerca que está el resultado de una medición del valor verdadero, cantidad de predicciones positivas que fueron correctas.
- Especificidad - Razón de Verdaderos Negativos (VNR) →  $SPC = \frac{VN}{N} = \frac{VN}{FP + VN} = 1 - FPR$  casos negativos que el algoritmo ha clasificado correctamente
- Precisión (Positive predictive value) →  $Precision = \frac{VP}{VP + FP}$  dispersión del conjunto de valores obtenidos, porcentaje de casos positivos detectados.
- Razón de Falsos Positivos (FPR) →  $FPR = \frac{FP}{N} = \frac{FP}{FP + VN}$

La **curva ROC** evalúa el rendimiento de los algoritmos de clasificación binaria. Con un mismo clasificador calculo las métricas de sensibilidad y FPR para **varios umbrales**, y debería utilizar el que caiga en el punto (0,1), es decir FPR=0 y sensibilidad=1.



Ayuda a encontrar un umbral de clasificación que se adapte a nuestro problema. *El área debajo de la curva determina lo bueno de un clasificador.* En la mayoría de los casos reales existe un umbral de clasificación (costo de equivocarse en la decisión).

# 3. REDES NEURONALES

Una **red neuronal** es un modelo simplificado que emula el modo en que el cerebro humano procesa la información: Funciona simultáneamente un número elevado de unidades de procesamiento interconectadas que parecen versiones abstractas de neuronas.

## 3.1 Perceptron

Un perceptrón es una **neurona artificial**, y, por tanto, una unidad de red neuronal. Efectúa cálculos para detectar características o tendencias en los datos de entrada.

Se trata de un algoritmo para el aprendizaje **supervisado** de clasificadores binarios. Permite que las neuronas artificiales aprendan y traten los elementos de una serie de datos.

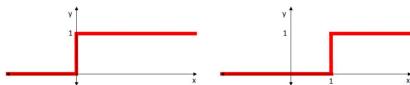
$$y = f \left( \sum_{i=1}^N w_i x_i - \theta \right)$$

La salida del perceptrón se define como:

- $x_i$  es el elemento  $i$  del vector de entradas  $x = [x_1, x_2, \dots, x_N]$
- $w_i$  es el elemento  $i$  del vector de pesos  $w = [w_1, w_2, \dots, w_N]$
- $\theta$  es el umbral de activación

- $f$  es la función umbral (también threshold o hard-lim)

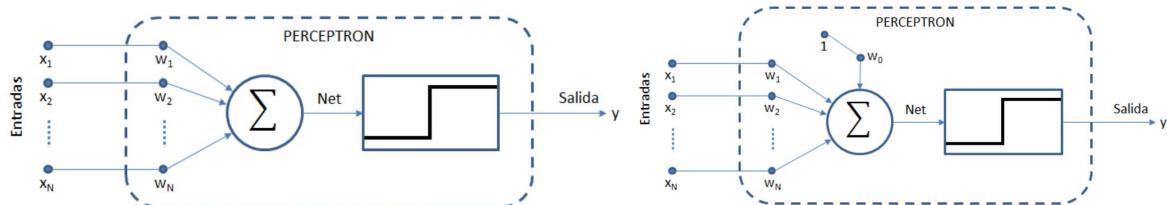
$$f(z) = \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases}$$



### Diagrama del perceptron

En definitiva se trata de una red que al recibir el vector de entrada calcula la función de decisión lineal y le aplica una función umbral que fuerza a la red a obtener solo dos valores posibles a la salida. (Usualmente 1 y 0 o 1 y -1 según como se defina F).

La diferencia fundamental está en la forma de calcular el vector de pesos  $W$ , no se hace por cálculos matriciales sino por un **método iterativo de aproximación**.



### Perceptrón con vector de entradas aumentado

$$y = f \left( \sum_{i=0}^N w_i x_i \right)$$

La salida del perceptrón se define como:

donde:  $w_0 = -\theta$  (Diagrama derecha)

En lugar de decir que expandimos el vector de entrada  $x$ , se adiciona a la neurona una entrada que siempre recibe el valor 1, que es lo mismo. Sin el término  $w_0$ , siempre pasaría por el 0 en eje, ya que es la ordenada al origen. Es el componente unitario de  $x$  suele definirse en forma separada.

En realidad el perceptrón es una **función matemática**. Los datos de entrada ( $x$ ) se multiplican por los coeficientes de peso ( $w$ ). El resultado es un valor que puede ser positivo o negativo. La neurona se activa si el peso calculado de los datos de entrada supera un umbral determinado.

El resultado predicho se compara con el resultado conocido. En caso de diferencia, el error se retropropaga para permitir ajustar los pesos.

## Entrenamiento

Para ajustar los parámetros internos (pesos) del perceptron se utiliza un método de aprendizaje supervisado, por lo tanto es necesario contar con datos en la forma:

$$\Delta_{w_i} = \alpha(y' - y)x_i$$

Los pesos se adaptan según la ley de aprendizaje:  
donde  $\alpha$  es la tasa de aprendizaje e  $y'$  es la salida esperada.

Los pasos del método son:

1. Definir el valor de  $\alpha$
2. Inicializar los pesos con valores aleatorios
3. Mientras error > 0 para algún vector de entradas, hacer:
  - a. Ejecutar ciclo completo de entrenamiento (epoch). Mientras existan vectores de entrenamiento hacer:
    - i. Tomar un vector entrada/salida del set de datos
    - ii. Calcular la salida  $y = f\left(\sum_{i=0}^N w_i x_i\right)$
    - iii. Calcular el  $error = y' - y$
    - iv. Calcular  $\Delta_{w_i} = \alpha(y' - y)x_i$
    - v. Modificar los pesos haciendo  $w_{i+1} = w_i + \Delta_{w_i}$

A menos que se defina una cantidad máxima de ciclos, el entrenamiento termina cuando **todos los vectores están bien clasificados**.

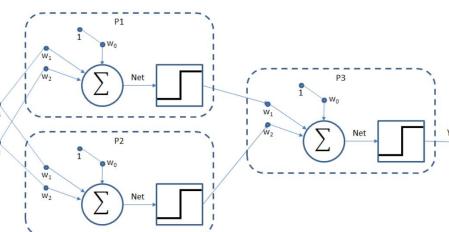
## Desventaja del perceptron

Al igual que el clasificador lineal, el perceptron no puede clasificar correctamente si las clases no son **linealmente separables**. Ejemplo: compuerta XOR.

Este problema NO se puede solucionar, el perceptrón NO puede resolver el problema, sin embargo, hay una alternativa **teórica** que involucra otro modelo y abre un camino para tratar este tipo de problemas. Conectando perceptrones entre sí, en lo que se llama perceptron multicapa, se podría resolver el problema **si supiéramos cómo adaptar los pesos**.

*¿Por qué no se puede aplicar este modelo? No sabemos cómo entrenarlo... No sabemos cómo modificar los pesos de los perceptrones de la primera capa porque no sabemos cuánto contribuyó cada uno al error de la última salida final.*

$X_0$	$X_1$	$X_2$	$X_3$	...	$X_N$	$y'$
1	0.14	0.24	0.34	...	0.55	1
1	0.11	0.89	0.33	...	0.9	1
1	0.97	0.27	0.34	...	0.53	0
1	0.93	0.36	0.72	...	0.75	1
1	0.92	0.15	0.29	...	0.97	0
1	0.58	0.31	0.52	...	0.14	0
1	0.29	0.52	0.54	...	0.88	1
1	0.58	0.59	0.51	...	0.06	1



## 3.2 Adaline

Adaline es un tipo de neurona con dos diferencias importantes respecto al perceptrón:

- Cambia el mecanismo de aprendizaje, se utiliza la regla delta, basada en el mínimo error cuadrático medio (Least Mean Squared o LMS error).
  - El aprendizaje busca encontrar el mejor vector de pesos posible en términos del criterio del error cuadrático medio.
- Tiene una función de activación lineal.
- Adaline está limitada a una única neurona de salida. Un vector  $x$  como su entrada y un número real y como salida

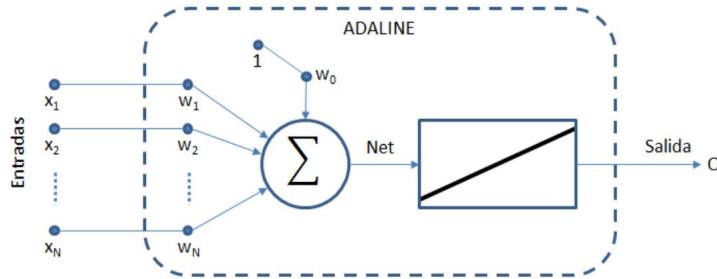
Nomenclatura:

- $O$  : salida de la red (por output). En Adaline  $O = \sum_{i=0}^N w_i x_i = Net$
- $y$  : salida esperada
- $k$  : identificador de un vector de entrenamiento

$$E = \frac{1}{2L} \sum_{k=1}^L (y_k - O_k)^2$$

El error cuadrático: donde  $L$  es la cantidad de vectores de entrenamiento.

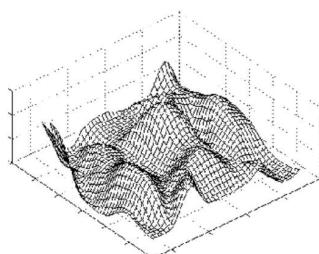
La adaptación de los pesos se hace a través de una búsqueda sobre la superficie del error. Para encontrar el mínimo de la función del error los pesos se modifican en cantidades proporcionales al gradiente decreciente de la función  $E$ .



### Ley de aprendizaje de Adaline o regla delta

La ley de aprendizaje de Adaline,  $\Delta_{w_{ik}} = \alpha(y_k - O_k)x_i$ , es igual a la del perceptrón, pero no hay que confundirse, el comportamiento no es el mismo porque el error puede tomar cualquier valor en  $\mathbb{R}$

*El mecanismo de aprendizaje conduce a actualizaciones de tipo continuo, siendo la actualización de los pesos proporcional al error cometido por la neurona.*



El **método de entrenamiento** también es similar al del perceptrón, con la diferencia de que el error nunca es cero, así que se necesita otra condición de corte. Típicamente se corta el entrenamiento cuando se llega a un umbral de error previamente definido o cuando el error se estabiliza.

Los **pesos se ajustan simultáneamente**, una tasa alta implica saltos oscilatorios, que impiden la convergencia hacia una solución; una tasa pequeña puede demorar demasiado el proceso.

### 3.3 Backpropagation

- Backpropagation es un método para entrenar redes neuronales multicapa.
- ¿Para qué sirve entrenar redes multicapa? Para clasificar datos que no sean linealmente separables.

La regla de entrenamiento backpropagation está basada en descenso de gradiente y regla de la cadena. Los pesos se inicializan aleatoriamente y cambian en la dirección que reduce el error

mediante la regla Delta:  $\Delta_{w_i} = -\alpha \delta x_i$  donde  $\delta = -(y - O)$

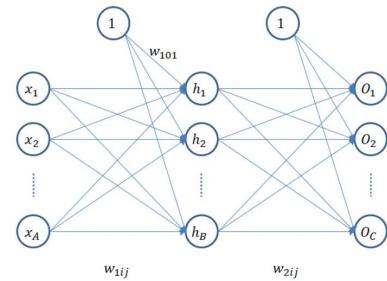
La regla delta ha sido extendida (regla delta generalizada o backpropagation) a redes con capas intermedias con **conexiones hacia adelante** (feedforward networks) y cuyas neuronas tienen **funciones de activación** continuas, no decrecientes y derivables.

#### Red multicapa con conexiones hacia adelante

El ejemplo más simple, con una sola capa oculta, donde  $i$  es la  $i$ -ésima entrada de la neurona  $j$ .

Notar que:

- No hay conexiones entre neuronas de la misma capa.
- No hay conexiones hacia neuronas de capas anteriores.
- Cada capa puede tener distintas cantidades de neuronas.
- La imagen corresponde a una red de tres capas (una oculta, la de salida y la de entrada).



#### Funciones de activación

Recordar que un Adaline, o cualquier neurona artificial antes de aplicar la función de activación, realiza una **transformación lineal**. Entonces, una red de  $n$  capas formadas por neuronas lineales se puede reemplazar por una red de una sola capa y, por lo tanto, no puede resolver problemas no linealmente separables. Por esta razón es necesario agregar no-linealidades entre las capas.

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

La función de activación no lineal más utilizada es la función sigmoidal:

$$\frac{\partial f(x)}{\partial x} = f(x)(1 - f(x))$$

La derivada de la función sigmoidal es: Entonces, por ejemplo, si las neuronas

de salida de la red anterior tienen funciones de activación sigmoidal,  $O_j = f(Net_j) = \frac{1}{1 + e^{-Net_j}}$  y la

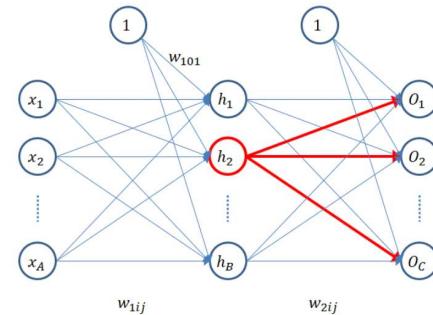
derivada de la salida con respecto a Net  $\frac{\partial O_j}{\partial Net_j} = \frac{1}{1 + e^{-Net_j}} \left( 1 - \frac{1}{1 + e^{-Net_j}} \right) = O_j(1 - O_j)$

#### Otras funciones de activación: (hay muchas más → [link](#))

Name	Plot	Equation	Derivative (with respect to $x$ )	Range	Order of continuity	Monotonic	Monotonic derivative	Approximate identity near the origin
Identity		$f(x) = x$	$f'(x) = 1$	$(-\infty, \infty)$	$C^\infty$	Yes	Yes	Yes
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$	$\{0, 1\}$	$C^{-1}$	Yes	No	No
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$	$(0, 1)$	$C^\infty$	Yes	No	No
Tanh		$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	$(-1, 1)$	$C^\infty$	Yes	No	Yes
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$	$(-\frac{\pi}{2}, \frac{\pi}{2})$	$C^\infty$	Yes	No	Yes

El nombre del método significa **retropropagación** o **propagación hacia atrás** (del error).

- El error y la modificación de los pesos de la capa de salida se calcula de la forma conocida.
- Para las capas ocultas, el error de cada neurona se calcula como la suma ponderada (por los pesos) de los errores de la capa siguiente.



## Método de entrenamiento

Cada paso (ciclo) del método tiene dos fases:

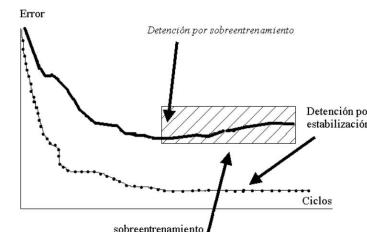
- En la primera se toma un vector de entrada y se lo propaga hacia adelante, a través de todas las capas, hasta calcular la salida.
- En la segunda fase se calcula el error de la capa de salida y se lo propaga hacia atrás para calcular el error de todas las neuronas de las capas ocultas. Una vez calculados todos los errores, se modifican los pesos.

### Pasos:

1. Definir el valor de  $\alpha$
2. Inicializar los pesos con valores aleatorios
3. Mientras error medio > error aceptado, hacer:
  - a. Ejecutar ciclo completo de entrenamiento (epoch). Mientras existan vectores de entrenamiento hacer:
    - i. Tomar un vector entrada/salida del set de datos
    - ii. Calcular la salida de cada capa hasta llegar a la capa de salida.
    - iii. Calcular el error de las neuronas de la capa de salida.
    - iv. Calcular el error de las neuronas de la última capa oculta.
    - v. Repetir el punto anterior para todas las capas ocultas (desde la salida hacia la entrada).
    - vi. Calcular los deltas de cada neurona en función de su propio error.
    - vii. Modificar los pesos.

## Otras consideraciones sobre Redes Neuronales

- Dificultad para elegir los hiperparámetros
  - Arquitectura
  - Inicialización de los pesos
  - Tasa de aprendizaje: Adición de un momento
- División del juego de datos en:
  - Datos de entrenamiento
  - Datos de validación
  - Datos de test
- Sobreentrenamiento
  - Luego de un período de entrenamiento normal, el error tiende a aumentar nuevamente



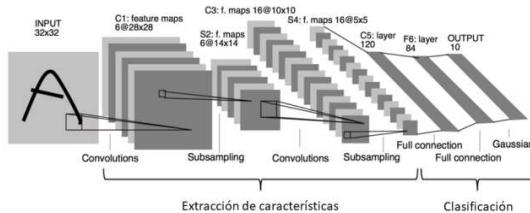
## 3.4 Deep Learning

El aprendizaje profundo es un conjunto de herramientas de aprendizaje automático que logra entrenar redes neuronales con varias capas de profundidad. También es:

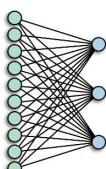
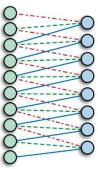
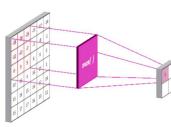
- Un tipo muy poderoso de aprendizaje automático.
- La reencarnación moderna de las redes neuronales artificiales.
- Un conjunto de funciones matemáticas simples y entrenables.
- Es el estado del arte para la mayoría de los desafíos que enfrenta el rec. de patrones.
  - Visión artificial, Reconocimiento de audio, Procesamiento del lenguaje natural (NLP), Sistemas de diagnóstico médico, Predicción de terremotos, energía, mercado, etc.

### End-to-End

- Proveer una **solución de extremo-a-extremo** es una de las grandes diferencias entre el aprendizaje profundo y el resto del aprendizaje automático. Se supone que es una ventaja.
- Los modelos de aprendizaje profundo incluyen la etapa de **extracción de características**.



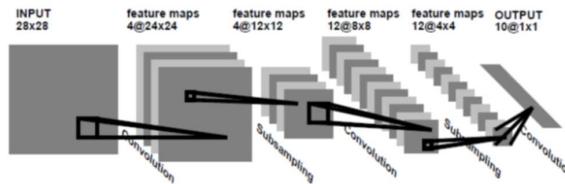
### Tipos más comunes de capas en un modelo de aprendizaje profundo

Densamente conectadas	 <p>La salida se calcula como la aplicación de la función de activación sobre la suma ponderada de las entradas. Las neuronas se conectan con todas las de la capa anterior. No hay conexiones entre neuronas de la misma capa.</p>
Convolucionales	 <p>La salida se calcula como la aplicación de la función de activación sobre la suma ponderada de las entradas. Las neuronas tienen un campo perceptivo acotado. No hay conexiones entre neuronas de la misma capa. Los pesos se comparten entre neuronas de la misma capa y kernel*</p>
Pooling	 <p>(MaxPooling, AveragePooling) Cada elemento calcula el máximo o la media de los valores en su campo receptivo.</p>
Recurrentes	<p>La salida es retroalimentada pasando por un retardo. Procesan secuencias de longitud indefinida.</p>
Long Short Term Memory (LSTM)	<p>Solucionan el problema de las dependencias Long-Term. Tienen la capacidad de añadir y eliminar información del estado de la célula.</p>

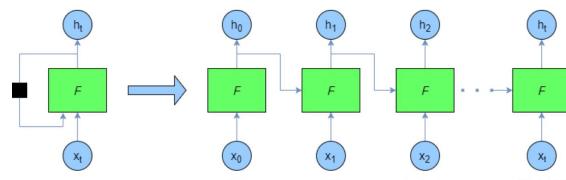
\* **Kernels de convolución:** Cada capa de convolución está compuesta por varios kernels que se especializan en detectar distintos patrones

### **Las redes convolucionales combinan capas convolucionales y de pooling.**

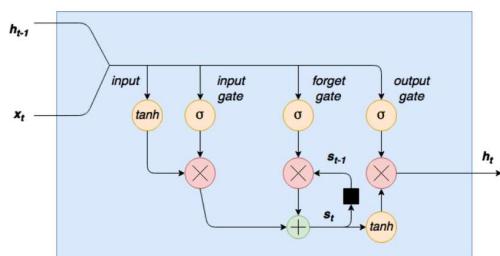
- Las capas convolucionales detectan patrones en distintos niveles.
- Las capas de pooling reducen la dimensión y hacen que la detección sea invarianta a la escala y a la traslación.



Capas Recurrentes

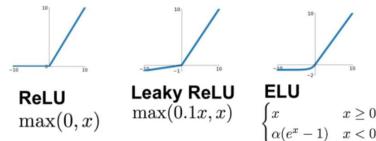


Capas LSTM



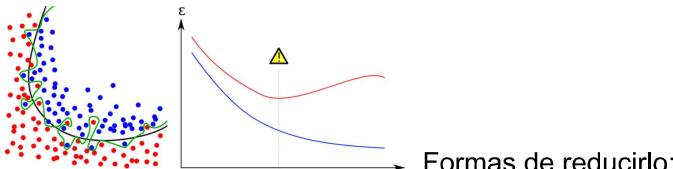
### **Complicaciones del Aprendizaje Profundo**

- Desvanecimiento del gradiente
  - Una solución es cambiar la función de activación por una función que no sature.



- Utilizar residual networks.
- Modelos más complejos (más parámetros)

- **Sobreentrenamiento:** Los modelos complejos son más propensos al este.



Formas de reducirlo:

- Early-stop
- Aumentar la cantidad de datos de entrenamiento
- Regularización
- Dropout previene la co-adaptación desmedida de las neuronas

- **Mayor necesidad de recursos**

- Modelos muy complejos (millones de parámetros)
- Por suerte son cálculos paralelizables
- La mejor solución son las GPUs (por ahora)



# Metaheurísticas

## Algoritmos heurísticos

Los algoritmos heurísticos son los más fáciles de utilizar, ya que se basan en el conocimiento de una heurística que guía el proceso de búsqueda. El conocimiento del problema usualmente ayuda a encontrar una heurística razonable que encontrará rápidamente una solución aceptable. Un algoritmo de este tipo sólo buscará dentro de un subespacio del área total a una solución buena (**que no necesariamente es la mejor**) **que satisfaga las restricciones impuestas**. La principal limitación es su incapacidad para escapar de óptimos locales (encontrar soluciones parcialmente óptimas).

## Algoritmos metaheurísticos

Una metaheurística es un proceso iterativo maestro que guía y modifica las operaciones de una heurística subordinada para producir eficientemente soluciones de alta calidad. Las metaheurísticas pueden manipular una única solución completa (o incompleta) o una colección de soluciones en cada iteración. La heurística subordinada puede ser un procedimiento de alto o bajo nivel, una búsqueda local, o un método constructivo. Entre los algoritmos metaheurísticos más conocidos están, el recocido simulado y los algoritmos genéticos.



# Metaheurísticas

## Los cuatro tipos fundamentales:

1. Las metaheurísticas **de relajación** se refieren a procedimientos de resolución de problemas que utilizan relajaciones del modelo original (es decir, modificaciones del modelo que hacen al problema más fácil de resolver), cuya solución facilita la solución del problema original.
2. Las metaheurísticas **constructivas** se orientan a los procedimientos que tratan de obtener una solución a partir del análisis y selección paulatina de las componentes que la forman.
3. Las metaheurísticas de **búsqueda** guían los procedimientos que usan transformaciones o movimientos para recorrer el espacio de soluciones alternativas y explorar las estructuras de entornos asociadas.
4. Las metaheurísticas **evolutivas** están enfocadas a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones.



# Metaheurísticas

## Metaheurísticas Evolutivas

Las metaheurísticas evolutivas establecen estrategias para **conducir** la evolución en el espacio de búsqueda de conjuntos de soluciones (usualmente llamados **poblaciones**) con la intención de acercarse a la solución óptima **con sus elementos**.

El aspecto fundamental de las heurísticas evolutivas consiste en la **interacción** entre los miembros de la población frente a las búsquedas que se guían por la información de soluciones individuales.

Las diferentes metaheurísticas evolutivas se distinguen por la forma en que **combinan** la información proporcionada por los elementos de la población para hacerla evolucionar mediante la obtención de **nuevas soluciones**.

Los algoritmos genéticos y meméticos y los de estimación de distribuciones emplean fundamentalmente **procedimientos aleatorios**, mientras que las metaheurísticas de búsqueda dispersa o de reencadenamiento de caminos (*Path-Relinking*) emplean **procedimientos sistemáticos**.



# Metaheurísticas – ALGORITMOS GENÉTICOS

## FUNDAMENTOS

En ocasiones la computación se basa en procesos observados de la naturaleza para resolver ciertos problemas: por ejemplo, las redes neuronales que replican los procesos de sinapsis entre las neuronas. En este caso, los algoritmos genéticos replican el modelo de selección natural propuesto por Darwin, y que resume la famosa frase ‘la supervivencia del más fuerte o adaptado’.

Este modelo básicamente dice que, dentro de una población, los individuos que sobreviven son aquellos que están más adaptados al medio, por lo tanto, las generaciones futuras de estos estarán mejor adaptadas ya que serán combinaciones de los mejores genes de sus antepasados. Además, esta teoría de la evolución introduce un concepto muy interesante que son las mutaciones. Una mutación es un pequeño cambio que se produce de manera aleatoria en ciertos individuos e introduce de esta manera versatilidad en las poblaciones. Habrá mutaciones que den lugar a cambios favorables y otros desfavorables.



# Metaheurísticas – ALGORITMOS GENÉTICOS

## USOS

Se utilizan para resolver problemas de Búsqueda y Optimización, ya que se basan en evolucionar poblaciones de soluciones hacia valores óptimos del problema.

## ESTRUCTURA DE UN ALGORITMO GENÉTICO

### Conceptos previos

**Individuo:** los individuos de la población son las posibles soluciones al problema que se intenta resolver.

**Población:** conjunto de individuos.

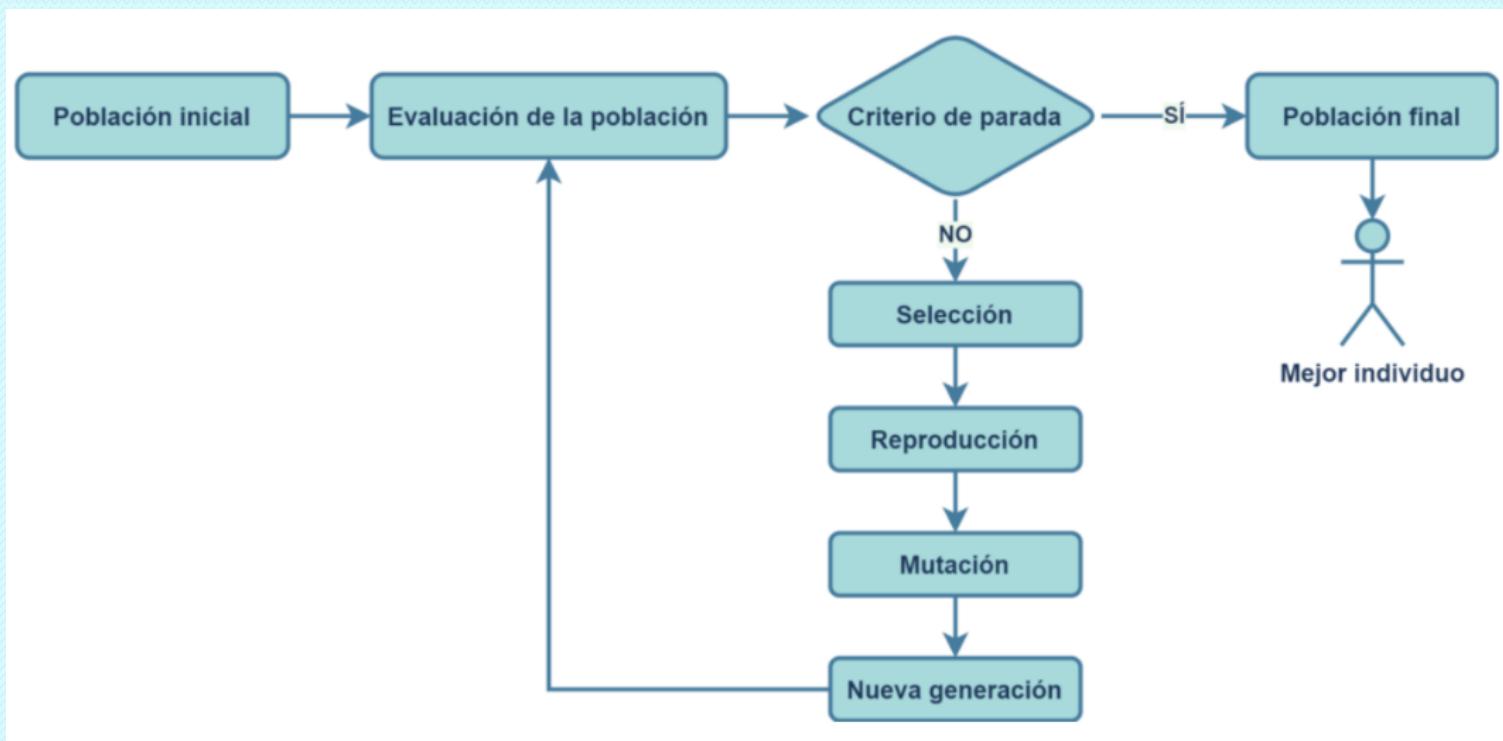
**Función fitness o de adaptación:** función que evalúa a los individuos y les asigna una puntuación en función de que tan buenas sean las soluciones para el problema.

**Función de cruce:** función que dados dos individuos genera dos descendientes a partir de la combinación de genes de sus padres, esta función depende del problema en cuestión.



# Metaheurísticas – ALGORITMOS GENÉTICOS

## ESTRUCTURA DE UN ALGORITMO GENÉTICO





# Metaheurísticas – ALGORITMOS GENÉTICOS

## ESTRUCTURA DE UN ALGORITMO GENÉTICO

- 1. Fase inicial:** se genera una población inicial de individuos (soluciones)
- 2. Fase de evaluación:** se evalúan los individuos de la población con la función fitness
- 3. Fase de selección :** se seleccionan los mejores individuos
- 4. Fase de reproducción:** se cruzan los individuos seleccionados mediante la función de cruce, dando lugar a una nueva generación que va a sustituir a la anterior
- 5. Fase de mutación:** se introducen mutaciones (pequeños cambios) en ciertos individuos de la nueva población de manera aleatoria o un entrecruzamiento cromosómico (también llamado crossover o recombinación)
- 6. Se obtuvo una nueva generación, en general, con soluciones mejores que la anterior. Se vuelve al punto 2**

**Los algoritmos genéticos pueden finalizar cuando se alcanza un número de generaciones concreto o cuando cumplen una condición de parada.**



# Metaheurísticas – ALGORITMOS GENÉTICOS

## VENTAJAS

- ✓ Se desenvuelven bien en problemas con un paisaje adaptativo complejo: aquéllos en los que la función de aptitud es ruidosa, cambia con el tiempo, o tiene muchos óptimos locales, gracias a los cuatro componentes principales de los algoritmos genéticos, paralelismo, selección, mutación y cruzamiento, los que trabajan juntos para conseguir su buen desempeño.
- ✓ Pueden explorar el espacio de soluciones en múltiples direcciones a la vez, por lo que, los algoritmos genéticos funcionan particularmente bien resolviendo problemas cuyo espacio de soluciones potenciales es realmente grande, demasiado vasto para hacer una búsqueda exhaustiva en un tiempo razonable
- ✓ Los algoritmos genéticos realizan cambios aleatorios en sus soluciones candidatas y luego utilizan la función de aptitud para determinar si esos cambios producen una mejora. Como sus decisiones están basadas en la aleatoriedad, todos los caminos de búsqueda posibles están abiertos; en contraste a cualquier otra estrategia de resolución de problemas que dependa de un conocimiento previo



# Metaheurísticas – ALGORITMOS GENÉTICOS

## DESVENTAJAS

- Si se elige mal una función de aptitud o se define de manera inexacta, puede que el algoritmo genético sea incapaz de encontrar una solución al problema, o puede acabar resolviendo el problema equivocado.
- Pueden tardar mucho en converger, o no converger en absoluto, dependiendo en cierta medida de los parámetros que se utilicen tamaño de la población, el ritmo de mutación y cruzamiento, el tipo y fuerza de la selección.
- El lenguaje utilizado para especificar soluciones candidatas debe ser robusto; es decir, debe ser capaz de tolerar cambios aleatorios que no produzcan constantemente errores fatales o resultados sin sentido. Una de las formas mas usadas es definir a los individuos como listas de números -binarios, enteros o reales- donde cada número representa algún aspecto de la solución candidata.

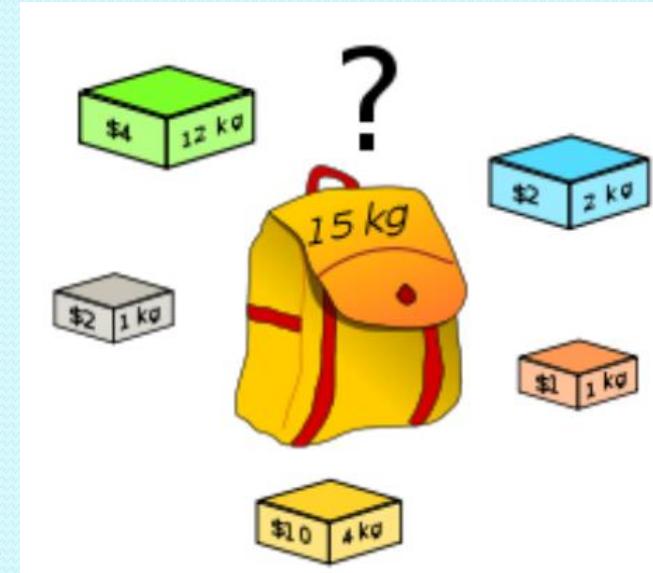


# Metaheurísticas – ALGORITMOS GENÉTICOS

## EJEMPLO PROPUESTO: EL PROBLEMA DE LA MOCHILA

Dada una mochila con una capacidad de 15 kg que se puede llenar con cajas de distinto peso y valor. ¿Qué cajas elijo de modo de maximizar mis ganancias y no exceder el peso permitido?

Una representación de una solución puede ser una matriz de bits, donde cada bit representa un objeto diferente, y el valor del bit (0 o 1) representa si el objeto está o no en la mochila. No todas las representaciones son válidas, ya que el tamaño de los objetos puede exceder la capacidad de la mochila. La aptitud de la solución es la suma de valores de todos los objetos en la mochila si la representación es válida, o 0 de lo contrario.





# Metaheurísticas – ALGORITMOS GENÉTICOS

## EJEMPLO PROPUESTO: EL PROBLEMA DE LA MOCHILA

Se tienen **n** objetos y una mochila

- El objeto **i** tiene peso **p<sub>i</sub>** y la inclusión del objeto **i** en la mochila produce un beneficio **b<sub>i</sub>**
- El objetivo es llenar la mochila, de capacidad **C**, de manera que se maximice el beneficio.

$$\text{maximizar} \quad \sum_{1 \leq i \leq n} b_i x_i$$

$$\text{sujeto a} \quad \sum_{1 \leq i \leq n} p_i x_i \leq C$$

$$\text{con} \quad x_i \in \{0,1\}, \quad b_i > 0, \quad p_i > 0, \quad 1 \leq i \leq n$$

# Metaheurísticas – ALGORITMOS GENÉTICOS

## EJEMPLO PROPUESTO: EL PROBLEMA DE LA MOCHILA

**Capacidad Mochila =15**



CUÁLES OBJETOS LLEVAR  
Binario

?

OBJETO 1	Utilidad = 4 Peso = 7
OBJETO 2	Utilidad = 5 Peso = 6
OBJETO 3	Utilidad = 6 Peso = 8
OBJETO 4	Utilidad = 3 Peso = 2

**CROMOSOMAS**

- 1 gen por variable de decisión binaria
- Cromosoma compuesto por los genes
- Función de fitness == Igual a la Función Objetivo

**ACTIVIDAD**

$$Z= 4x_1 + 5x_2 + 6x_3 + 3x_4$$

Restricción :  $7x_1 + 6x_2 + 8x_3 + 2x_4 \leq 15$

Line = 4 bits    Un bit por cada objeto [0,1]

Cromosoma:

$X_1$	$X_2$	$X_3$	$X_4$
$b_1$	$b_2$	$b_3$	$b_4$

Tamaño de la población: = 4



# Metaheurísticas – ALGORITMOS GENÉTICOS

EJEMPLO PROPUESTO: EL PROBLEMA DE LA MOCHILA

1. Población inicial aleatoria

Cromosomas

0	1	1	0
---	---	---	---

1	0	1	0
---	---	---	---

0	1	0	0
---	---	---	---

0	1	1	0
---	---	---	---

Pcr: 0,98

Pmut: 0,1

# Metaheurísticas – ALGORITMOS GENÉTICOS

## EJEMPLO PROPUESTO: EL PROBLEMA DE LA MOCHILA

Evalúa cada individuo de la población

$$Z = 4x_1 + 5x_2 + 6x_3 + 3x_4 \quad \text{Pcr: } 0,98 \quad P = 7x_1 + 6x_2 + 8x_3 + 2x_4 \leq 15 \quad \text{Pmut: } 0,1$$

Ind	x1	x2	x3	x4	Z	Peso	Probab	Prob Acum
1	0	1	1	0	11	14	0,29730	0,29730
2	1	0	1	0	10	15	0,27027	0,56757
3	0	1	0	0	5	6	0,13514	0,70270
4	0	1	1	0	11	14	0,29730	1,00000

### PRIMERA ITERACIÓN (a)

Aleat1:Padr1 0,4771 Ind 2

Aleat2:Padr2 0,9341 Ind 4

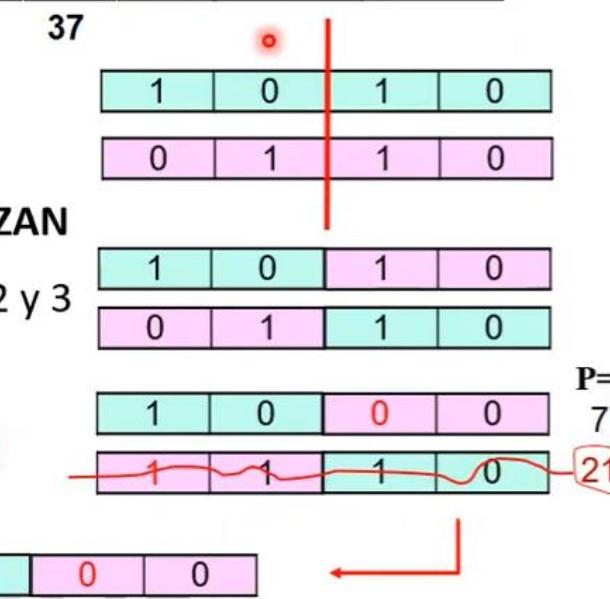
Cruzan? 0,8448 <0,98 SI CRUZAN

PuntoCorte 0,4376 Entre gen 2 y 3

Aleatorios para Mutación

0,2940	0,3741	0,0257	0,6164
0,0509	0,6250	0,4106	0,1718

<0,1



# Metaheurísticas – ALGORITMOS GENÉTICOS

EJEMPLO PROPUESTO: EL PROBLEMA DE LA MOCHILA

PRIMERA ITERACIÓN (b)

Ind	x1	x2	x3	x4	Z	Peso	Probab	Prob Acum
1	0	1	1	0	11	14	0,29730	0,29730
2	1	0	1	0	10	15	0,27027	0,56757
3	0	1	0	0	5	6	0,13514	0,70270
4	0	1	1	0	11	14	0,29730	1,00000

Aleat1: Padr1 0,8024 Ind 4



Aleat2: Padr2 0,2657 Ind 1



Cruzan? 0,0343 <0,98 SI CRUZAN



PuntoCorte 0,7402 Entre gen 3 y 4



Aleatorios para Mutación

0,0426	0,2529	0,3018	0,6218
0,3377	0,0844	0,5885	0,5171

<0,1

P=



21 8



Hijo 2





# Metaheurísticas – ALGORITMOS GENÉTICOS

EJEMPLO PROPUESTO: EL PROBLEMA DE LA MOCHILA

PRIMERA ITERACIÓN (c)

Ind	x1	x2	x3	x4	Z	Peso	Probab	Prob Acum
1	0	1	1	0	11	14	0,29730	0,29730
2	1	0	1	0	10	15	0,27027	0,56757
3	0	1	0	0	5	6	0,13514	0,70270
4	0	1	1	0	11	14	0,29730	1,00000

Aleat1:Padr1 0,6578 Ind 3

0	1	0	0
---	---	---	---

Aleat2:Padr2 0,2580 Ind 1

0	1	1	0
---	---	---	---

Cruzan? 0,9890 <0,98 NO CRUZAN

Aleatorios para Mutación

0,1674	0,3015	0,0926	0,8496
0,9182	0,9214	0,1670	0,1554

<0,1

P=

0	1	1	0
---	---	---	---

14

0	1	1	0
---	---	---	---

14

Hijo 3

0	1	1	0
---	---	---	---

Hijo 4

0	1	1	0
---	---	---	---



# Metaheurísticas – ALGORITMOS GENÉTICOS

EJEMPLO PROPUESTO: EL PROBLEMA DE LA MOCHILA

**GENERACION t+1**

Fin de primera iteración

Ind	x1	x2	x3	x4	Z	Peso	Probab	Prob Acum
1	1	0	0	0	4	7	0,12500	0,12500
2	0	0	1	0	6	8	0,18750	0,31250
3	0	1	1	0	11	14	0,34375	0,65625
4	0	1	1	0	11	14	0,34375	1,00000
					32			1

Continuar con la siguiente iteración ...



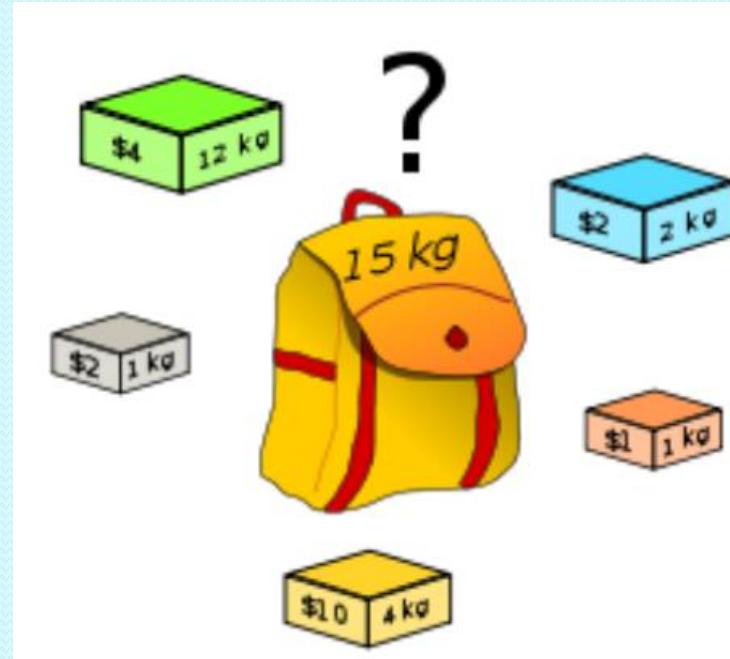
Pero si ya fuera la ultima iteración... Entonces la solución entregada sería

Llevar los objetos X2 y X3  
Con una utilidad 11  
y un peso de 14



# Metaheurísticas – ALGORITMOS GENÉTICOS

EJEMPLO PROPUESTO: EL PROBLEMA DE LA MOCHILA



[https://drive.google.com/file/d/14wGUfQhQODYXEXvepV-Xh3YghadQpkJO/view?usp=drive\\_link](https://drive.google.com/file/d/14wGUfQhQODYXEXvepV-Xh3YghadQpkJO/view?usp=drive_link)

## Redes Neuronales

Las redes neuronales son más que otra forma de emular ciertas características propias de los humanos, como la capacidad de memorizar y de asociar hechos. Si se examinan con atención aquellos problemas que no pueden expresarse a través de un algoritmo, se observará que todos ellos tienen una característica en común: la experiencia. El hombre es capaz de resolver estas situaciones acudiendo a la experiencia acumulada. Así, parece claro que una forma de aproximarse al problema consiste en la construcción de sistemas que sean capaces de reproducir esta característica humana.

En definitiva, las redes neuronales no son más que un modelo artificial y simplificado del cerebro humano, que es el ejemplo más perfecto del que disponemos para un sistema que es capaz de adquirir conocimiento a través de la experiencia. Una red neuronal es "**un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona**".

Todos los procesos del cuerpo humano se relacionan en alguna u otra forma con la (in)actividad de estas neuronas. Las mismas son un componente relativamente simple del ser humano, pero cuando millares de ellas se conectan en forma conjunta se hacen muy poderosas.

También, es bien conocido que los humanos son capaces de aprender. Aprendizaje significa que aquellos problemas que inicialmente no pueden resolverse, pueden ser resueltos después de obtener más información acerca del problema.

Por lo tanto, las Redes Neuronales consisten de unidades de procesamiento que intercambian datos o información. Se utilizan para reconocer patrones, incluyendo imágenes, manuscritos y secuencias de tiempo, tendencias financieras.

Existen numerosas formas de definir a las redes neuronales; desde las definiciones cortas y genéricas hasta las que intentan explicar más detalladamente qué son las redes neuronales. Por ejemplo:

- 1) **Un sistema de computación compuesto por un gran número de elementos simples, elementos de procesos muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas.**
- 2) **Redes neuronales artificiales son redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real del mismo modo que lo hace el sistema nervioso biológico.**

## Ventajas que ofrecen las redes neuronales

Debido a su constitución y a sus fundamentos, las redes neuronales artificiales presentan un gran número de características semejantes a las del cerebro. Por ejemplo, son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que representan información irrelevante, etc. Esto hace que ofrezcan numerosas ventajas y que este tipo de tecnología se esté aplicando en múltiples áreas. Entre las ventajas se incluyen:

**Aprendizaje Adaptativo:** capacidad de aprender a realizar tareas basadas en un entrenamiento o en una experiencia inicial.

**Auto-organización:** una red neuronal puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje.

**Tolerancia a fallos:** la destrucción parcial de una red conduce a una degradación de su estructura sin embargo, algunas capacidades de la red se pueden retener, incluso con un gran daño.

**Operación en tiempo real:** los cómputos neuronales pueden ser realizados en paralelo; para esto se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.

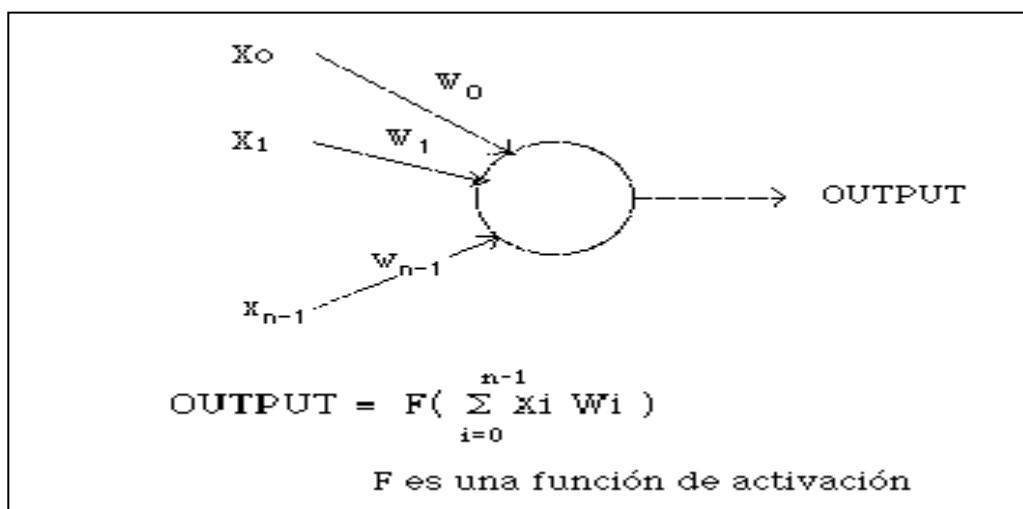
**Fácil inserción dentro de la tecnología existente:** se pueden obtener chips especializados para redes neuronales que mejoran su capacidad en ciertas tareas. Ello facilitará la integración modular en los sistemas existentes.

### **Algunas aplicaciones de redes neuronales artificiales.**

- Reconocimiento de caracteres impresos
- Reconocimiento de caracteres manuscritos.
- Memorias asociativas.
- Reconocimiento de voz
- Control de robots
- Toma de decisiones
- Reconocimiento de enfermedades cardiacas
- Reconocimiento de señales de radio
- Generación de reglas para sistemas expertos

### **El Neuron Artificial**

Es un elemento procesador al que se aplican un conjunto de entradas, cada una representando la salida de otro neurón. Estas entradas que podemos llamar  $X_i$ , se multiplican por un peso (número real) asociado a ellas, que podemos llamar  $W_i$



$$\text{NET} = \sum_{i=0}^{n-1} x_i w_i$$

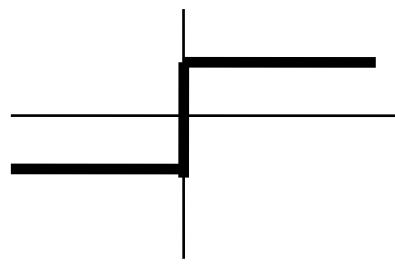
La suma ponderada se "filtra" por medio de una función, llamada función de activación:

$$\text{OUT} = F(\text{NET})$$

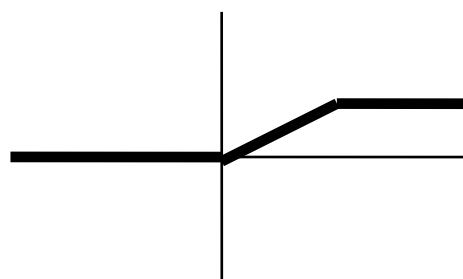
Se dice que un neurón "enciende" si su valor de salida es mayor o igual a un valor umbral, el cual es determinado por los posibles valores de la función de activación.

Las funciones de activación pueden ser las siguientes:

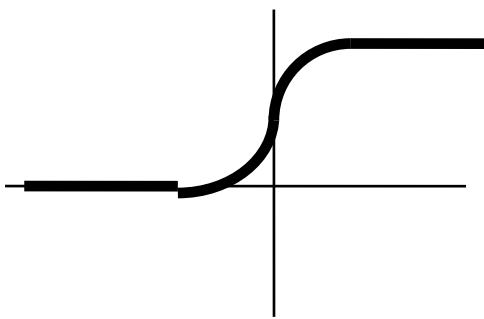
a) escalón



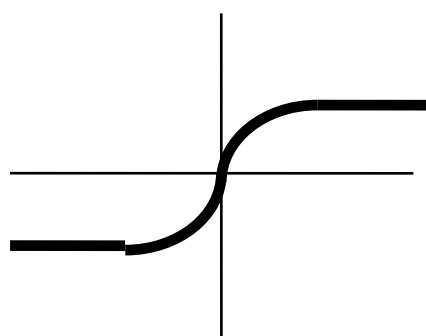
b) lineal con umbral



c) sigmoide



d) tangente hiperbólica



### Características de las redes neuronales

Existen cuatro aspectos que caracterizan una red neuronal: su topología, el mecanismo de aprendizaje, tipo de asociación entre la información de entrada y de salida, y la forma de representación de estas informaciones.

#### Topología : Monocapa y Multicapa

Consiste en la organización y disposición de las neuronas en la red formando capas o agrupaciones de neuronas. Los parámetros fundamentales de la red son: número de capas, número de neuronas por capa, grado de conectividad y tipo de conexión entre neuronas.

Al hacer una clasificación topológica de las RNAs se suelen distinguir:

**Redes monocapa** : se establecen conexiones laterales entre las neuronas que pertenecen a la única capa que constituye la red. Ejemplos red de HOPFIELD. Las redes monocapa se utilizan típicamente en tareas relacionadas con lo que se conoce como autoasociación; por ejemplo, para regenerar informaciones de entrada que se presenta como incompleta o distorsionada.

**Redes multicapa** : disponen las neuronas agrupadas en varios niveles. Dado que este tipo de redes disponen de varias capas, las conexiones entre neuronas pueden ser del tipo feedforward (conexión hacia adelante) o del tipo feedback (conexión hacia atrás).

## **Que es el aprendizaje? :**

El aprendizaje es el proceso por el cual una red neuronal modifica sus pesos en respuesta a una información de entrada. Los cambios que se producen durante la etapa de aprendizaje se reducen a la destrucción (el peso de la conexión toma el valor 0), modificación y creación (el peso de la conexión toma un valor distinto de 0) de conexiones entre las neuronas.

Podemos considerar que el proceso de aprendizaje ha terminado:

- a) Cuando los valores de los pesos permanecen estables
- b) Mediante un número fijo de ciclos
- c) Cuando el error es menor que el error inicialmente establecido

Un aspecto importante es determinar los criterios de la regla de aprendizaje; cómo se van a modificar los pesos.

## **Mecanismos de aprendizaje**

**Aprendizaje supervisado**  
**Aprendizaje NO supervisado**

La diferencia entre ambos tipos estriba en la existencia o no de una agente externo que controle todo el proceso.

Otro criterio para diferenciar las reglas de aprendizaje se basa en considerar si la red puede aprender durante su funcionamiento (aprendizaje ON LINE) o requiere de una fase previa de entrenamiento (aprendizaje OFF LINE). En este último debe existir un conjunto de datos de entrenamiento y un conjunto de datos de test o prueba; igualmente los pesos de las conexiones no se modifican después de terminar la etapa de entrenamiento de la red. En la red ON LINE los pesos varían dinámicamente cada vez que se presente una nueva información al sistema.

### **Redes con aprendizaje supervisado:**

Se caracteriza porque el proceso de aprendizaje se realiza mediante un entrenamiento controlado por un agente externo (supervisor, maestro) que determina la respuesta que debería generar la red a partir de una entrada determinada. El supervisor comprueba la salida generada por el sistema y en el caso de que no coincida con la esperada, se procederá a modificar los pesos de las conexiones. En este tipo de aprendizaje se suelen distinguir a su vez tres formas de llevarlo a cabo:

- **Por corrección de error**
- **Por refuerzo**
- **Estocástico**

### **Aprendizaje por corrección de error :**

Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida.

Algoritmos que utilizan este tipo de aprendizaje son:

Regla de aprendizaje del perceptrón: utilizada en la red PERCEPTRON

Regla delta generalizada: utilizada en redes multicapa

### **Aprendizaje por refuerzo :**

Este tipo de aprendizaje es más lento que el anterior y se basa en la idea de no disponer de un ejemplo completo del comportamiento deseado; es decir, de no indicar durante el entrenamiento la salida exacta que se desea que proporcione la red ante una determinada entrada. Aquí la función del supervisor se reduce a indicar mediante una señal de refuerzo si la salida obtenida en la red se ajusta a la deseada (éxito = +1 o fracaso = -1) y en función de ello se ajustan los pesos basándose en un mecanismo de probabilidades.

### **Aprendizaje estocástico :**

Consiste básicamente en realizar cambios aleatorios en los valores de los pesos y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad. Un red que utiliza este tipo de aprendizaje es la red Boltzman Machine, ideada por Hinton, Ackley y Sejnowski en 1984 y la red Cauchy Machine desarrollada por Szu en 1986.

### **Redes con aprendizaje NO supervisado**

No requieren de influencia externa para ajustar los pesos de las conexiones entre sus neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta; son capaces de autoorganizarse.

Estas redes deben encontrar las características, regularidades, correlaciones o categorías que se pueden establecer entre los datos de la entrada.

Pero, ¿qué genera la red en la salida?. Existen varias posibilidades en cuanto a interpretación:

- La salida representa el grado de familiaridad o similitud entre la información de entrada y las informaciones mostradas con anterioridad.
- Clusterización o establecimiento de categorías, indicando la red a la salida a qué categoría pertenece la información de entrada, siendo la propia red la que debe establecer las correlaciones oportunas.
- Codificación de los datos de entrada, generando a la salida una versión codificada con menos bits, pero manteniendo la información relevante de los datos.
- Mapeo de características, obteniéndose una disposición geométrica que representa un mapa topográfico de las características de los datos de entrada.

### **Tipo de asociación entre las informaciones de entrada y salida**

Las redes neuronales son sistemas que almacenan cierta información aprendida; esta se registra de forma distribuida en los pesos asociados a las conexiones entre neuronas. Hay que establecer cierta relación o asociación entre la información presentada a la red y la salida ofrecida por esta. Es lo que se conoce como memoria asociativa.

Existen dos formas primarias de realizar esta asociación entrada/salida y que generan dos tipos de redes:

### **Redes heteroasociativas :**

La red aprende parejas de datos  $[(A_1, B_1), (A_2, B_2), \dots, (A_n, B_n)]$ , de tal forma que cuando se le presente determinada información de entrada  $A_i$  responda con la salida correspondiente  $B_i$ . Al asociar informaciones de entrada con diferentes informaciones de salida, precisan al menos de 2 capas, una para captar y retener la información de entrada y otra para mantener la salida con la información asociada. Si esto no fuese así se perdería la información inicial al obtenerse la salida asociada; es necesario mantener la información de entrada puesto que puede ser necesario acceder varias veces a ella, por lo que debe permanecer en la capa de entrada. El aprendizaje de este tipo de redes suele ser **supervisado**.

### **Redes autoasociativas :**

La red aprende ciertas informaciones  $A_1, A_2, \dots, A_n$  de forma que cuando se le presenta una información de entrada realizará una autocorrelación, respondiendo con uno de los datos almacenados, el más parecido al de entrada. Este tipo de redes pueden implementarse con una sola capa de neuronas. El tipo de aprendizaje utilizado habitualmente es el **no supervisado** y suelen utilizarse en tareas de filtrado de información para la reconstrucción de datos, eliminando distorsiones o ruido, explorar relaciones entre informaciones similares para facilitar la búsqueda por contenido en bases de datos y para resolver problemas de optimización

## **Representación de la información de entrada y salida**

### **Redes continuas :**

En un gran número de redes, tanto los datos de entrada como de salida son de naturaleza analógica (valores reales continuos y normalmente normalizados, por lo que su valor absoluto será menor que la unidad). En este caso las funciones de activación de las neuronas serán también continuas, del tipo lineal o sigmoidal.

### **Redes discretas :**

Por el contrario, otras redes sólo admiten valores discretos  $[0,1]$  a la entrada, generando también en la salida respuestas de tipo binario. La función de activación en este caso es del tipo escalón.

### **Redes híbridas :**

La información de entrada es continua pero a la salida ofrecen información binaria.

# Inteligencia Artificial

UTN FRC 2023



Mario Alejandro García  
September 18, 2023

# 10 Modelos bayesianos

Las redes bayesianas, también llamadas redes de creencia, redes causales y diagramas de influencia, son estructuras gráficas para representar las relaciones probabilísticas entre un gran número de variables y para realizar inferencias probabilísticas sobre esas variables.

El contenido de este capítulo completa la Unidad 4, razonamiento bajo incertidumbre.

El material de lectura principal es el libro [RNR04]. Se debe leer:

- Secciones 13.1 a 13.6
- Secciones 14.1 a 14.4 hasta “Inferencia por enumeración” inclusive.

Un buen libro para profundizar es [Nea04], donde además se presentan varios métodos para que los agentes inteligentes puedan adquirir de forma automática el conocimiento que se representa en redes bayesianas.

En el resto del capítulo haremos un recorrido, a modo de repaso, por conceptos importantes de Probabilidad y después veremos el uso de este conocimiento en redes bayesianas, siempre como un complemento al libro y haciendo foco en explicar con ejemplos algunas ideas que podrían no estar suficientemente claras en el libro.

Una aclaración sobre el libro. En la página 550 de [RNR04], donde dice

$$\mathbf{P}(Dolor - de - muelas|Caries)\mathbf{P}(Infectarse, Caries)\mathbf{P}(Caries),$$

debería decir

$$\mathbf{P}(Dolor - de - muelas|Caries)\mathbf{P}(Infectarse|Caries)\mathbf{P}(Caries).$$

## 10.1 Algunos conceptos de repaso

### Probabilidad *a priori*

La probabilidad *a priori* o incondicional asociada a una proposición  $a$  es el grado de creencia que se le otorga en **ausencia de cualquier otra información**; y se escribe como  $P(a)$ . Por ejemplo, dada la variable  $X$  con dominio (posibles valores que puede tomar) es  $\langle x_1, x_2 \rangle$ , si la probabilidad *a priori* de que  $X = x_1$  es 0,3, deberíamos escribir

$$P(X = x_1) = 0,3 \quad \text{o} \quad P(x_1) = 0,3.$$

### Probabilidad condicional

Una vez que el agente obtiene alguna evidencia que afecta a la variable  $X$ , las probabilidades *a priori* ya no son aplicables a  $X$ . En su lugar usamos probabilidades *a posteriori* o condicionales. La probabilidad condicional del evento  $a$  dada la ocurrencia del evento  $b$  se escribe como  $P(a|b)$ . Por ejemplo, si la probabilidad de que  $X = x_1$  dada la evidencia de que  $Y = y_1$  es 0,7, deberíamos escribir

$$P(X = x_1|Y = y_1) = 0,7 \quad \text{o} \quad P(x_1|y_1) = 0,7.$$

Las probabilidades condicionales pueden definirse en términos de probabilidades no condicionales. La ecuación que la define es

$$P(a|b) = \frac{P(a \wedge b)}{P(b)},$$

donde  $P(a \wedge b)$  es la probabilidad conjunta de  $a$  y  $b$ , es decir, la probabilidad de que  $a$  y  $b$  ocurran al mismo tiempo. Recordar que muchos textos se utiliza  $P(a \cap b)$  en lugar de  $P(a \wedge b)$ . Si bien el significado es el mismo, pensarlo desde la teoría de conjuntos e imaginar el diagrama de Venn suele ser un aporte valioso para facilitar la comprensión de estos conceptos.

De la ecuación anterior se obtiene

$$P(a \wedge b) = P(a|b)P(b), \tag{10.1}$$

conocida como la **regla del producto**. Esta última ecuación es más intuitiva, ya que expresa que la probabilidad de que ocurran  $a$  y  $b$  es igual a la probabilidad de que ocurra  $a$  dado  $b$ , siempre y cuando ocurra  $b$ , o sea, por la probabilidad de  $b$ .

La probabilidad condicional es una herramienta muy útil para representar información causal de la forma  $P(\text{efecto}|\text{causa})$ .

### Regla de Bayes

Como  $P(a \wedge b) = P(b \wedge a)$ , usando la ecuación 10.1 podemos decir que

$$P(b \wedge a) = P(b|a)P(a).$$

Entonces,

$$P(b|a)P(a) = P(a|b)P(b)$$

$$P(b|a) = \frac{P(a|b)P(b)}{P(a)}. \quad (10.2)$$

La ecuación 10.2 es la regla o teorema de Bayes. Permite calcular la probabilidad condicional de una variable dada otra cuando se cuenta con la información en el sentido inverso. Por ejemplo, si una alarma se dispara (por la razón que sea) con probabilidad  $P(\text{alarma})$ , los robos ocurren con probabilidad  $P(\text{robo})$  y sabemos que la probabilidad de que nuestra alarma se dispare cuando hay un robo es  $P(\text{alarma}|\text{robo})$ , con la regla de Bayes podemos calcular la probabilidad de que esté ocurriendo un robo cuando suena la alarma,  $P(\text{robo}|\text{alarma})$ .

### Ejemplo de uso de la regla de Bayes

Un ejemplo para comprender la importancia del teorema de Bayes. Supongamos que Juan tiene un examen médico, que incluye una radiografía de tórax, como rutina de ingreso para su nuevo trabajo en un banco y que en la radiografía hay un hallazgo compatible con el cáncer de pulmón. A esto último lo representamos como  $\text{Rad} = \text{positivo}$ . Al recibir el resultado, Juan piensa que tiene la enfermedad ( $\text{Enf} = \text{verdadero}$ ) y se preocupa mucho, pero... ¿debería hacerlo? Sin conocer la exactitud del test, Juan realmente no puede saber qué tan probable es que tenga cáncer de pulmón. Cuando se entera que el test no es absolutamente concluyente, decide investigar y descubre que este tiene una

tasa de falsos negativos de 0,4 y de falsos positivos 0,02. De los datos podemos deducir que:

$$P(Rad = \text{positivo} | Enf = \text{verdadero}) = 0,6$$

$$P(Rad = \text{positivo} | Enf = \text{falso}) = 0,02$$

Dadas estas probabilidades, Juan se siente un poco mejor. Sin embargo, nota que todavía no sabe cuál es la probabilidad de qué él tenga cáncer de pulmón. La probabilidad de que Juan tenga cáncer de pulmón es  $P(Enf = \text{verdadero} | Rad = \text{positivo})$ , y esta no es una de las probabilidades listadas recién. Juan finalmente recuerda el teorema de Bayes y se da cuenta de que todavía necesita otra probabilidad para determinar la que le interesa. La probabilidad faltante es  $P(Enf = \text{verdadero})$ , que representa la probabilidad de que él tenga cáncer de pulmón antes de conocer el resultado de la radiografía. El otro dato útil que tiene Juan es que pertenece al grupo de personas que se realizó un examen preocupacional, no un examen motivado por síntomas. Entonces, cuando se entera de que solo 1 de cada 1000 nuevos empleados tiene cáncer de pulmón, asigna 0.001 a  $P(Enf = \text{verdadero})$ . Ahora sí, Juan aplica la regla de Bayes:

$$\begin{aligned} & P(Enf = \text{verdadero} | Rad = \text{positivo}) \\ &= \frac{P(Rad = \text{positivo} | Enf = \text{verdadero})P(Enf = \text{verdadero})}{P(Rad = \text{positivo})} \\ &= \frac{P(Rad = \text{positivo} | Enf = \text{verdadero})P(Enf = \text{verdadero})}{P(Rad = \text{pos} | Enf = \text{ver})P(Enf = \text{ver}) + P(Rad = \text{pos} | Enf = \text{fal})P(Enf = \text{fal})} \\ &= \frac{0,6 \times 0,001}{0,6 \times 0,01 + 0,02 \times 0,999} \\ &= 0,029 \end{aligned}$$

Entonces, ahora Juan sabe que su probabilidad de tener la enfermedad es cercana a 0.03 y se relaja un poco mientras espera el resultado de otros estudios.

Supongamos que otra persona, Pedro, tiene el mismo diagnóstico de radiografía de tórax que tuvo Juan ( $Rad = \text{positivo}$ ). Sin embargo, Pedro se hizo el estudio porque ha trabajado en minas durante 20 años, y sus empleadores se han preocupado porque notaron que cerca del 10% de sus trabajadores desarrollaron cáncer de pulmón después de trabajar varios años en las minas. ¿Cuál es la probabilidad de que Pedro tenga cáncer de pulmón? Basado en la información que tenemos sobre Pedro antes de que se realizará el test, le asignamos una probabilidad *a priori*  $P(Enf = \text{verdadero}) = 0,1$ . Repitiendo los cálculos de la regla de Bayes para este valor, podemos concluir que  $P(Enf = \text{verdadero} | Rad = \text{positivo}) = 0,769$  para Pedro, no muy alentador.

### Distribución conjunta completa

La distribución conjunta completa es la distribución de probabilidad conjunta que considera el conjunto completo de variables. Es decir, que contiene la probabilidad de ocurrencia de cada una de las combinaciones posibles entre los valores que puede tomar cada variable. La tabla 10.1 muestra la distribución conjunta completa para el caso sencillo del ejemplo de la radiografía de tórax de Juan.

Notar que la última fila y la última columna de la tabla 10.1 contienen la probabilidad de ocurrencia de cada valor de  $Enf$  y  $Rad$  respectivamente. Estas probabilidades ( $P(x_i)$ ) se llaman marginales y se calculan sumando los valores de la fila o columna donde se encuentran, es decir sumando todas las probabilidades conjuntas donde la variable toma el valor  $x_i$ .

La tabla 10.2 muestra la distribución conjunta completa para el ejemplo de Pedro. Podríamos unir los ejemplos, pero para eso deberíamos definir una nueva variable *Trabajo* con dominio  $\langle en\_las\_minas, en\_otro\_lado \rangle$  y conocer la probabilidad de que una persona trabaje en las minas. Si esto fuera así, la tabla tendría todas las combinaciones de las tres variables.

Table 10.1: Distribución conjunta completa para el ejemplo de Juan.

		<i>Enf</i>		$P(Rad)$
		<i>verdadero</i>		
<i>Rad</i>	<i>positivo</i>	0,0006	0,01988	0,02058
	<i>negativo</i>	0,0004	0,97902	0,97942
$P(Enf)$		0,001	0,999	1

Table 10.2: Distribución conjunta completa para el ejemplo de Pedro.

		<i>Enf</i>		$P(Rad)$
		<i>verdadero</i>		
<i>Rad</i>	<i>positivo</i>	0,06	0,018	0,078
	<i>negativo</i>	0,04	0,882	0,922
$P(Enf)$		0,1	0,9	1

### Independencia

Dos eventos  $a$  y  $b$  son independientes si se cumple alguna de las siguientes condiciones:

1.  $P(a|b) = P(a)$  y  $P(a) \neq 0$ ,  $P(b) \neq 0$ .

2.  $P(a) = 0$  o  $P(b) = 0$ .

Es decir, si  $P(a)$  y  $P(b)$  no son nulas,  $a$  y  $b$  son independientes cuando  $P(a|b) = P(a)$ . En ese caso, claramente la probabilidad de ocurrencia del evento  $a$  no cambia si ocurre o no ocurre  $b$ .

Solo si  $a$  y  $b$  son independientes, partiendo de  $P(a|b) = P(a)$  y usando la regla del producto (ecuación 10.1) se puede ver que  $P(a \wedge b) = P(a)P(b)$ .

Es importante tener en mente que la dependencia entre dos variables o eventos no implica que uno sea la causa del otro.

### **Independencia condicional**

Dos eventos  $a$  y  $b$  son condicionalmente independientes dado  $c$ , si  $P(c) \neq 0$  y se cumple alguna de las siguientes afirmaciones:

1.  $P(a|b \wedge c) = P(a|c)$  y  $P(a|c) \neq 0$ ,  $P(b|c) \neq 0$ .
2.  $P(a|c) = 0$  o  $P(b|c) = 0$ .

Un ejemplo para comprender esta propiedad. Supongamos que en un pueblo existen dos vecinos que no tienen ningún tipo de interacción entre ellos. Cada vecino tiene cierta probabilidad de salir de su casa con paraguas cuando hay pronóstico de lluvias. Llamemos  $p_1$  y  $p_2$  a los eventos “salir con paraguas” para el vecino 1 y para el vecino 2 respectivamente. Naturalmente,  $P(p_1)$  y  $P(p_2)$  son altas en caso de pronóstico positivo. Cuando uno de los vecinos sale con paraguas es más probable que el otro lo haga también, es decir,  $p_1$  y  $p_2$  no son independientes. Esto no significa que se influyan mutuamente en el mundo real. El comportamiento se debe a que ambos eventos tienen la misma causa. Entonces, para un observador que no conoce el pronóstico, los eventos son dependientes. Si el observador conoce el pronóstico, los eventos se vuelven independientes dado el pronóstico. Es decir, si sabemos que va a llover, la probabilidad de ocurrencia del evento  $p_1$  es condicionalmente independiente de la ocurrencia del evento  $p_2$  y se escribe así:  $P(p_1|lluvia \wedge p_2) = P(p_1|lluvia)$ , donde  $lluvia$  es el evento que representa al pronóstico positivo de lluvia.

La independencia condicional tiene particular importancia en la utilización de las redes bayesianas porque, como veremos más adelante, permiten representar toda la información necesaria mediante un conjunto reducido de probabilidades condicionales.

## 10.2 Redes bayesianas

Una red bayesiana es una estructura de datos que representa las dependencias entre variables. Muestra una descripción compacta de cualquier distribución de probabilidad conjunta completa. Es un grafo dirigido en el que cada nodo contiene información probabilística cuantitativa. Especificación completa:

1. Un conjunto de variables aleatorias forman los nodos de la red. Las variables pueden ser discretas o continuas.
2. Un conjunto de arcos dirigidos conectan pares de nodos. Si hay un arco de un nodo  $X$  a un nodo  $Y$ , se dice que  $X$  es un *padre* de  $Y$ .
3. Cada nodo  $X_i$  tiene una distribución de probabilidad condicionada  $P(X_i|Padres(X_i))$  que cuantifica el efecto de los padres del nodo.
4. El grafo no tiene ciclos dirigidos, entonces es un un grafo acíclico dirigido, o GAD.

La topología de la red especifica las relaciones de independencia condicional que existen en el dominio. El significado intuitivo de un arco que sale de  $X$  y apunta a  $Y$  es, habitualmente, que  $X$  tiene una influencia directa sobre  $Y$ . Es generalmente sencillo para un experto del dominio decidir qué influencias directas existen en el área. Una vez que la topología de la red bayesiana está diseñada, necesitamos sólo especificar una distribución de probabilidad condicional para cada variable dados sus padres. La combinación de la topología y las distribuciones condicionales son suficientes para definir la distribución conjunta completa para todas las variables.

### 10.2.1 Inferencia en redes bayesianas

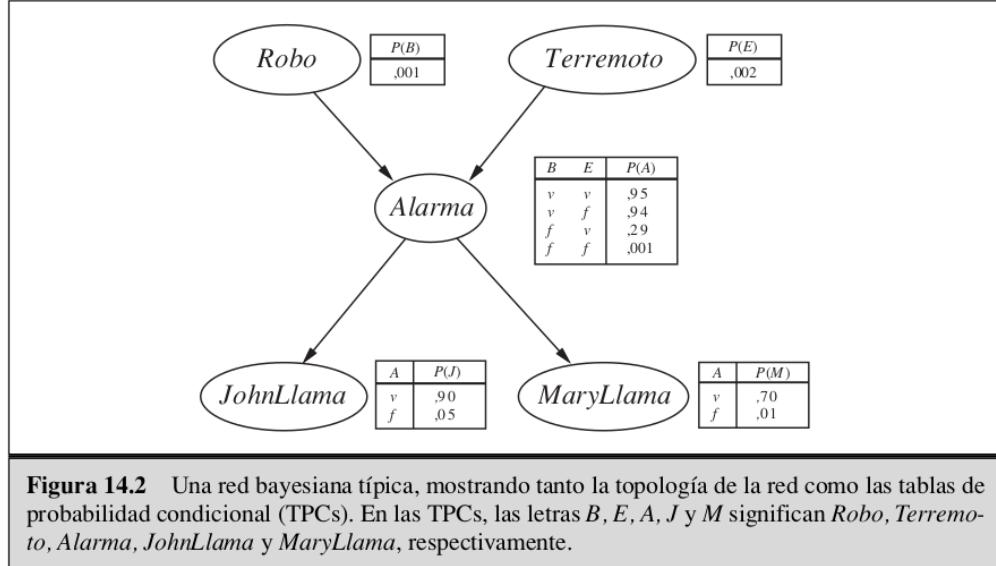
La tarea básica de cualquier sistema de inferencia probabilista es calcular la distribución de probabilidad *a posteriori* para un conjunto de variables pregunta, dado algún evento observado (esto es, alguna asignación de valores para un conjunto de variables evidencia). Notación en este contexto:  $X$  es la variable pregunta;  $\mathbf{E}$  es el conjunto de variables evidencias  $E_1, \dots, E_m$ , y  $e$  es un evento observado particular;  $\mathbf{Y}$  denotará las variables no evidencia  $Y_1, \dots, Y_l$  (a veces llamadas variables ocultas). Así, el conjunto completo de variables es  $\mathbf{X} = X \cup \mathbf{E} \cup \mathbf{Y}$ . Una pregunta típica pide la distribución de probabilidad *a posteriori*  $\mathbf{P}(X|e)$ .

Planteamos una pregunta sobre el ejemplo del robo (figura 10.1)en [RNR04]. En la red del robo, podemos observar el evento en el que *JohnLlama* = cierto y *MaryLlama* = cierto. Podríamos entonces preguntarnos por la probabilidad de que haya ocurrido un

robo:

$$\mathbf{P}(Robo|JohnLlama = cierto \wedge MaryLlama = cierto)$$

Para hacer más cortas las expresiones, reemplazemos los nombres de las variables por



**Figura 14.2** Una red bayesiana típica, mostrando tanto la topología de la red como las tablas de probabilidad condicional (TPCs). En las TPCs, las letras  $B$ ,  $E$ ,  $A$ ,  $J$  y  $M$  significan *Robo*, *Terremoto*, *Alarma*, *JohnLlama* y *MaryLlama*, respectivamente.

Figura 10.1: Red bayesiana del ejemplo del robo en [RNR04].

la primera letra, por ejemplo  $R$  para *Robo*. Como las variables de este ejemplo son binarias, vamos a usar la misma letra (en minúscula) para indicar el caso verdadero y la letra negada para el caso falso, por ejemplo  $\langle r, \neg r \rangle$  para  $R$ . Entonces, la pregunta se puede escribir como  $\mathbf{P}(R|j \wedge m)$  si queremos conocer tanto la probabilidad de que el robo haya ocurrido como la de que el robo no haya ocurrido (devuelve dos valores) y  $P(r|j \wedge m)$  si sólo nos interesa la probabilidad de que el robo sea cierto.

### Usando las distribuciones conjuntas totales

Si bien es el método más simple y directo para realizar inferencias sobre el dominio, técnicamente no es un método de inferencia sobre **redes bayesianas** porque requiere que las probabilidades estén expresadas como probabilidades conjuntas.

Almacenar probabilidades conjuntas no es eficiente. Se podrían calcular a partir de las probabilidades condicionales, pero esto no mejoraría la eficiencia. Además, aunque se partiera de las probabilidades conjuntas el método tiene complejidad algorítmica muy alta.

A pesar de estas desventajas, es importante comprender cómo y por qué funciona, ya

que es la base para el resto de los métodos. La explicación se omite en este documento porque está muy clara en la sección 13.4 de [RNR04]. Leer del libro.

### Con la regla de Bayes

Respondamos la pregunta usando la regla de Bayes:

$$P(r|j \wedge m) = \frac{P(j \wedge m|r)P(r)}{P(j \wedge m)} \quad (10.3)$$

Sabemos que  $P(r) = 0,001$ , pero no conocemos  $P(j \wedge m|r)$  ni  $P(j \wedge m)$ . Tenemos que calcularlas. Empecemos por  $P(j \wedge m)$ , para lo cual es necesario conocer  $P(a)$  porque

$$P(j \wedge m) = 0,9 \times 0,7 \times P(a) + 0,05 \times 0,01 \times P(\neg a). \quad (10.4)$$

Entonces, calculamos  $P(a)$  como

$$\begin{aligned} P(a) &= 0,95P(r)P(t) + 0,94P(r)P(\neg t) + 0,29P(\neg r)P(t) + 0,001P(\neg r)P(\neg t) \\ &= 0,95 \times 0,001 \times 0,002 + 0,94 \times 0,001 \times 0,998 + 0,29 \times 0,9990,002 + \\ &\quad 0,001 \times 0,999 \times 0,998 \\ &\approx 0,002516442. \end{aligned}$$

Ahora sí (recordando que  $P(\neg a) = 1 - P(a)$ ),

$$P(j \wedge m) \approx 0,002084100239. \quad (10.5)$$

El siguiente paso es calcular  $P(j \wedge m|r)$  como

$$P(j \wedge m|r) = 0,9 \times 0,7 \times P(a|r) + 0,05 \times 0,01 \times P(\neg a|r). \quad (10.6)$$

Notar que el cálculo es parecido al de 10.4, pero en este caso usamos  $P(a|r)$  porque suponemos que es cierto que hubo un robo, entonces no debemos usar la probabilidad *a priori*  $P(a)$ .

No conocemos todavía  $P(a|r)$ . Para calcularla suponemos que el robo existió, pero no sabemos nada sobre el terremoto, así que debemos tener en cuenta las dos posibilidades

(con y sin terremoto):

$$\begin{aligned}
 P(a|r) &= P(a|r \wedge t)P(t) + P(a|r \wedge \neg t)P(\neg t) \\
 &= 0,95 \times 0,002 + 0,94 \times 0,998 \\
 &\approx 0,94002.
 \end{aligned}$$

Con lo anterior calculado, volvemos a la ecuación 10.6:

$$\begin{aligned}
 P(j \wedge m|r) &= 0,9 \times 0,7 \times 0,94002 + 0,05 \times 0,01 \times (1 - 0,94002) \\
 &\approx 0,59224259
 \end{aligned}$$

Finalmente, podemos responder la pregunta inicial completando el proceso iniciado en la ecuación 10.3:

$$\begin{aligned}
 P(r|j \wedge m) &= \frac{P(j \wedge m|r)P(r)}{P(j \wedge m)} \\
 &\approx \frac{0,59224259 \times 0,001}{0,002084100239} \\
 &\approx 0,284171835364393
 \end{aligned}$$

La probabilidad de que haya ocurrido un robo, dado que Mary y John llamaron para avisar, es cercana a 0.28.

### Algunas consideraciones

Si bien el desarrollo anterior no está en el libro, es fundamental para comprender cómo se lleva a cabo el razonamiento sobre las redes bayesianas. Entenderlo es necesario tanto para tomar decisiones sobre la aplicabilidad de este tipo de soluciones a un problema real, como para la comprensión de los métodos que se utilizan en inferencia y aprendizaje automáticos sobre redes de Bayes.

En las redes grandes que pueden aparecer en casos reales, como el de la figura 10.2, hay aspectos muy importantes a tener en cuenta:

- Llevar a cabo un proceso de deducción y análisis similar al del ejemplo anterior es prácticamente imposible.
- La representación de las probabilidades conjuntas totales tendría un tamaño inmanejable, por eso usamos las probabilidades condicionales.
- Los mecanismos de inferencia automáticos se deben enfocar en la eficiencia, hasta el punto donde, para casos de gran tamaño, el cálculo exacto es demasiado ambicioso.
- No es posible crear estas redes de forma manual. Tanto el conocimiento del dominio para definir la topología, como las probabilidades conocidas, no son suficientes. En estos casos, se deben utilizar métodos automáticos para *aprender* las redes.

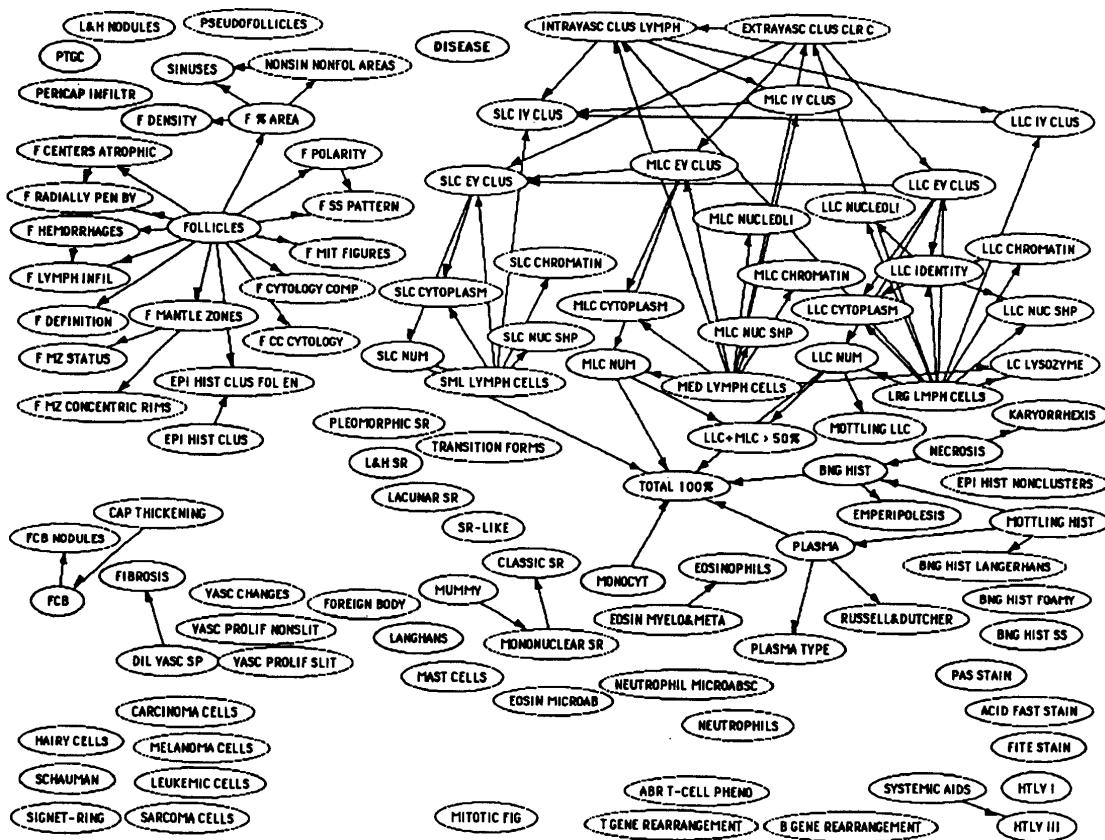


Figura 10.2: Red bayesiana de patologías de ganglios linfáticos en el sistema de decisión del proyecto Pathfinder [HHN90].

### Inferencia automática por enumeración

La inferencia por enumeración también está muy bien explicada en el libro, por lo tanto no se ofrece acá una explicación completa. Ver sección 14.4 de [RNR04].

Sí vamos a ver los detalles de los cálculos para el ejemplo del robo para mejorar la comprensión. Partimos de la expresión

$$\mathbf{P}(\mathbf{X}|\mathbf{e}) = \alpha \mathbf{P}(\mathbf{X}, \mathbf{e}) = \alpha \sum_y \mathbf{P}(\mathbf{X}, \mathbf{e}, \mathbf{y}),$$

donde  $\alpha$  es la constante de normalización  $\frac{1}{P(\mathbf{e})}$ .

Para el caso del robo, tal como se puede ver en el libro,

$$P(r|j, m) = \alpha \sum_{t^* \in \langle t, \neg t \rangle} \sum_{a^* \in \langle a, \neg a \rangle} P(r)P(t^*)P(a^*|r, t^*)P(j|a^*)P(m|a^*), \quad (10.7)$$

donde  $\alpha = 1/P(j \wedge m) \approx 1/0,002084100239$  según la ecuación 10.5.

Notar que la “P” de  $P(r|j, m)$  en la ecuación 10.7 no está en negrita, pero en el libro sí. Esto es un error del libro, porque se está calculando una sola probabilidad, no un conjunto. Además, también con respecto al libro, se cambió la nomenclatura a los subíndices de las sumatorias para darle mayor claridad.

Las variables  $T$  y  $A$  pueden tomar dos valores cada una,  $\langle t, \neg t \rangle$  y  $\langle a, \neg a \rangle$  respectivamente. Entonces, cálculo se realiza en cuatro ciclos, uno para cada combinación posible de las variables, tal como lo indican las sumatorias de la ecuación 10.7. A continuación se calcula el resultado de las sumatorias para cada ciclo  $c_i$ .

$$\begin{aligned}
c_1 &= P(r)P(t)P(a|r, t)P(j|a)P(m|a) \\
&= 0,001 \times 0,002 \times 0,95 \times 0,9 \times 0,7 \\
&\approx 0,000001197
\end{aligned}$$

$$\begin{aligned}
c_2 &= P(r)P(t)P(\neg a|r, t)P(j|\neg a)P(m|\neg a) \\
&= 0,001 \times 0,002 \times 0,05 \times 0,05 \times 0,01 \\
&\approx 5 \times 10^{-11}
\end{aligned}$$

$$\begin{aligned}
c_3 &= P(r)P(\neg t)P(a|r, \neg t)P(j|a)P(m|a) \\
&= 0,001 \times 0,998 \times 0,94 \times 0,9 \times 0,7 \\
&\approx 0,0005910156
\end{aligned}$$

$$\begin{aligned}
c_4 &= P(r)P(\neg t)P(\neg a|r, \neg t)P(j|\neg a)P(m|\neg a) \\
&= 0,001 \times 0,998 \times 0,06 \times 0,05 \times 0,01 \\
&\approx 0,00000002994
\end{aligned}$$

Finalmente usamos los  $c_i$  en la ecuación 10.7:

$$\begin{aligned}
P(r|j, m) &= \alpha(c_1 + c_2 + c_3 + c_4) \\
&\approx \frac{0,000001197 + 5 \times 10^{-11} + 0,0005910156 + 0,00000002994}{0,002084100239} \\
&\approx 0,284171835364393
\end{aligned}$$

Si comparamos este método con el utilizado antes, podemos ver que el resultado es el mismo, que se realizaron menos pasos y de forma automática; y utilizando (salvo por el cálculo de  $\alpha$ ) solo los valores originales de las tablas de probabilidad condicional.

Tal como se puede ver en el libro, la eficiencia de este método se puede mejorar y existen otros métodos que, si bien están fuera del alcance de la cátedra, deben ser considerados en caso de aplicación en problemas reales.



# Metaheurísticas – ALGORITMOS GENÉTICOS

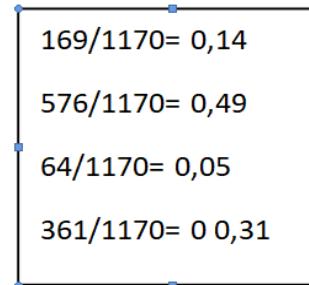
## RESOLUCIÓN DE EJEMPLOS

Ejercicio 2: Resolución problema de la función  $f(x) = x^2$

Población inicial generada aleatoriamente: 01101, 11000, 01000, 10011

Primera iteración

Cromosoma	x	f(x)	Prob.	Copias
01101	13	169	0.14	1
11000	24	576	0.49	2
01000	8	64	0.06	0
10011	19	361	0.31	1
		1170		



En este caso de acuerdo a las probabilidades los dos pares de padres se entrecruzan y solamente los primeros hijos mutan

Primeros padres      Hijos      mutan

01|101                  01000    01100

11|000                  11101    11001

Segundos padres

11|000                  11011

10|011                  10000



# Metaheurísticas – ALGORITMOS GENÉTICOS

## RESOLUCIÓN DE EJEMPLOS

### Segunda iteración

Cromosoma	X	f(x)	Prob.	Copias
11011	27	729	0.41	2
10000	16	256	0.15	1
01100	12	144	0.08	0
11001	25	625	0.36	1
		1754		

En este caso de acuerdo a las probabilidades los dos pares de padres se entrecruzan y solamente los primeros hijos de cada pares de padres mutan

Primeros padres      Hijos      mutan

11|011                  11000          11001

10|000                  10011          10011

Segundos padres

11|011                  11001          11101

11|001                  11011          11011



# Metaheurísticas – ALGORITMOS GENÉTICOS

## RESOLUCIÓN DE EJEMPLOS

### Tercera iteración

Cromosoma	X	f(x)	Prob.	Copias
11101	29	841	0.33	2
11011	27	729	0.29	1
11001	25	625	0.24	1
10011	19	361	0.14	0
		2556		

En este caso de acuerdo a las probabilidades los dos pares de padres se entrecruzan y solamente el segundo hijo de los primeros padres muta

Primeros padres      Hijos      mutan

11|101                  11011      11011

11|011                  11101      11001

Segundos padres

11|101                  11001      11001

11|001                  11101      11101

### Cuarta iteración

Cromosoma	X	f(x)	Prob.	Copias
11011	27	729	0.26	2
11001	25	625	0.22	0
11001	25	625	0.22	0
11101	29	841	0.30	2
		2820		



# Metaheurísticas – ALGORITMOS GENÉTICOS

## RESOLUCIÓN DE EJEMPLOS

Ejercicio 3: [https://drive.google.com/file/d/1coD0-q8Xf9Zf724\\_j6CQErb9wFAWVOME/view?usp=drive\\_link](https://drive.google.com/file/d/1coD0-q8Xf9Zf724_j6CQErb9wFAWVOME/view?usp=drive_link)

Ejercicio 4: [https://drive.google.com/file/d/14wGUfQhQODYXEXvepV-Xh3YghadQpkJO/view?usp=drive\\_link](https://drive.google.com/file/d/14wGUfQhQODYXEXvepV-Xh3YghadQpkJO/view?usp=drive_link)



# Metaheurísticas – ALGORITMOS GENÉTICOS

## RESOLUCIÓN DE EJEMPLOS

Ejercicio 5: Resolución Problema del carguero.

	C1	C2	C3	C4
Peso	100	155	50	112
Beneficio	20	35	22	5

Población inicial generada aleatoriamente  
1010, 0101, 1100, 0100

Primera iteración

Cromosoma	Peso	Beneficio	Prob.	Prob. Acum
1010	150	42	0.25	0.25
0101	267	40	0.23	0.48
1100	255	55	0.32	0.80
0100	155	35	0.20	1
		172		

Probabilidad de cruce 0.98 Probabilidad de mutación 0.1

Probabilidad de primer padre 0.43 el primero que la supera es 0101 (ver probab. acumulada)

Probabilidad de segundo padre 0.69 el primero que la supera es 1100

Probabilidad de cruce 0.84 < 0.98 se cruzan

Probabilidad de corte 0.44

Probabilidad de mutación 0.1

0.29 0.37 0.03 0.62

0.05 0.63 0.41 0.17



# Metaheurísticas – ALGORITMOS GENÉTICOS

## RESOLUCIÓN DE EJEMPLOS

Padres	Hijos	mutan	Peso	Beneficio
--------	-------	-------	------	-----------

01 01	0100	0110	205	57
-------	------	------	-----	----

11 00	1101	0101	267	40
-------	------	------	-----	----

Probabilidad de primer parento 0.70 el primero que la supera es 1100

Probabilidad de segundo parento 0.35 el primero que la supera es 0101

Probabilidad de cruce 0.26 < 0.98 se cruzan

Probabilidad de corte 0.74

Probabilidad de mutación 0.1

0.23 0.07 0.36 0.62

0.52 0.16 0.05 0.71



# Metaheurísticas – ALGORITMOS GENÉTICOS

## RESOLUCIÓN DE EJEMPLOS

Padres    Hijos    mutan    Peso    Beneficio

110 0	1101	1001	212	25	Hasta aquí respetan el peso (Se descartaría si no respeta la carga máxima)
010 1	0100	0110	205	57	

## NUEVA POBLACIÓN

Cromosoma	Peso	Beneficio	Prob.	Prob. Acum
0101	267	40	0.22	0.22
0110	205	57	0.32	0.54
1001	212	25	0.14	0.68
0110	205	57	0.32	1
		179		

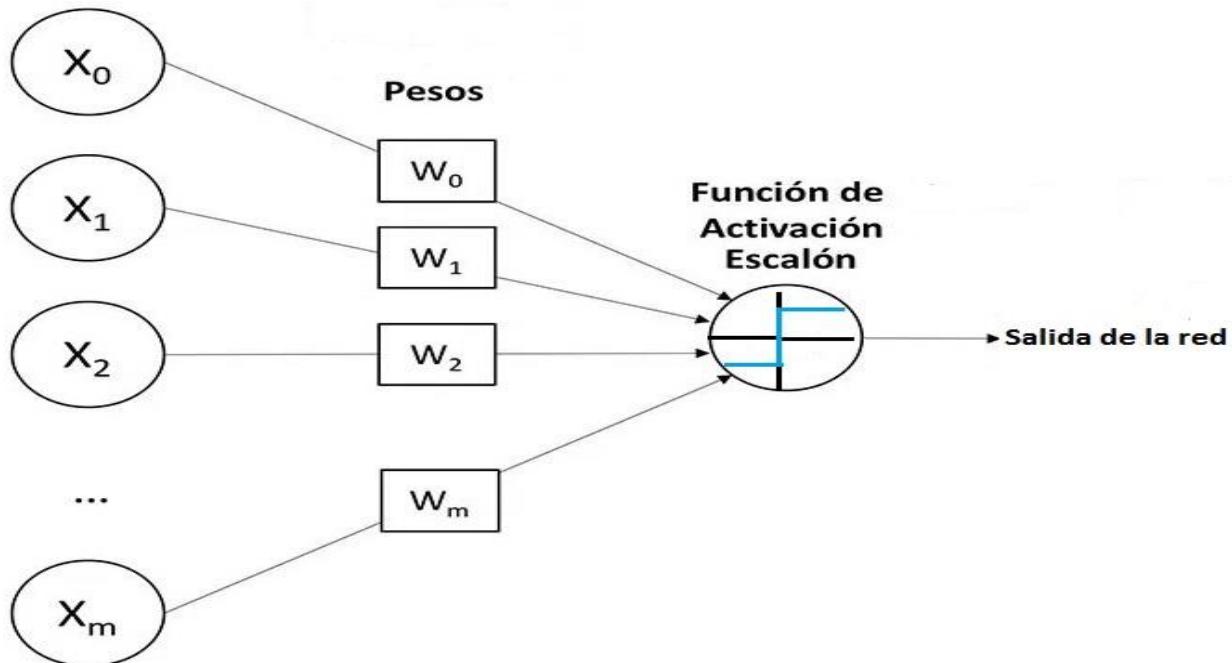


## Algoritmos de Redes Neuronales

### Algoritmo de Red Perceptrón

Red monocapa con conexiones hacia adelante con aprendizaje supervisado.

#### Entradas



1) Inicialización de pesos  $W_i \quad i = 1 \text{ a } m$

Se asignan valores aleatorios a cada uno de los pesos  $W_i$  de las conexiones

2) Presentación de un nuevo par (entrada, salida esperada)

Presentar un patrón de entrada  $X_p \ (x_1, x_2, \dots, x_m)$  junto con la salida esperada  $d_p(t)$

3) Cálculo de la salida actual

$$Y_p(t) = f [ \sum w_i(t) x_i(t) ] \quad i = 1 \text{ a } m$$

siendo  $f [ ]$  la función de activación en este caso la **función escalón**

4) Adaptación de los pesos

**error**

$$W_i(t+1) = W_i(t) + \alpha [ d_p(t) - Y_p(t) ] x_i(t)$$

5) Volver al paso 2

El proceso se repite hasta que el

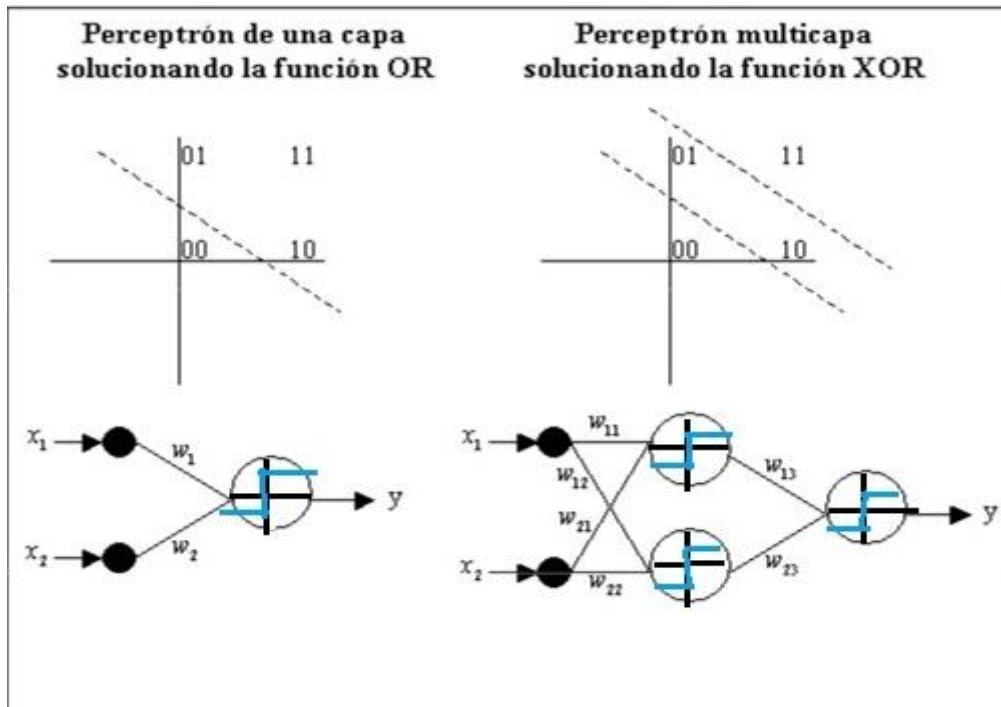
**error = salida esperada  $d(t)$  – salida de la red  $y(t)$**   
sea 0 para todas las entradas.



### A tener en cuenta:

La red perceptron utiliza la función de activación escalón sólo tiene en cuenta si se ha equivocado o no.

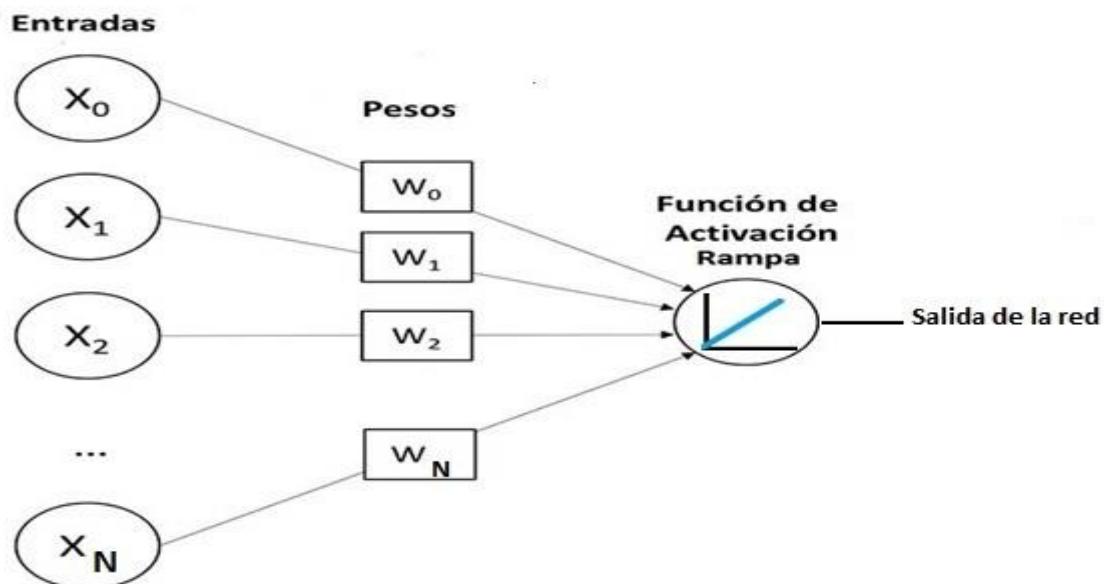
## Ejemplo de red Perceptron mono y multicapa





## Algoritmo de la red Adaline (ADAptive LINear Element)

Red monocapa con conexiones hacia adelante con aprendizaje supervisado.



$$X_k = (x_1, x_2, \dots, x_N) \quad K = 1 \text{ a } L \quad L \text{ es la cantidad de entradas.}$$

1) Inicialización de pesos Se asignan **valores aleatorios** a cada uno de los pesos  $W_j$  de las conexiones  $j = 1 \text{ a } N$

2) Presentación de un nuevo par (entrada, salida esperada) Presentar un patrón de entrada  $X_k$  con  $k = 1 \text{ a } L \quad L \text{ es la cantidad de entradas.}$

3) Cálculo de la salida actual

$S_k = f [\sum w_j x_{kj}] \quad \text{de } j = 1 \text{ a } N \quad (j \text{ es la cantidad de componentes de cada entrada})$   
siendo  $f [ ]$  la función de activación en este caso la **función rampa**

$$e_k = (d_k - S_k)$$

3) Se actualizan los pesos

$$W_j(t+1) = w_j(t) + \alpha [e_k x_{kj}]$$

Siendo  $\alpha$  la tasa de aprendizaje

4) Se repiten los pasos 1 al 3 con todos los vectores de entrada L

5) Si el error cuadrado medio:

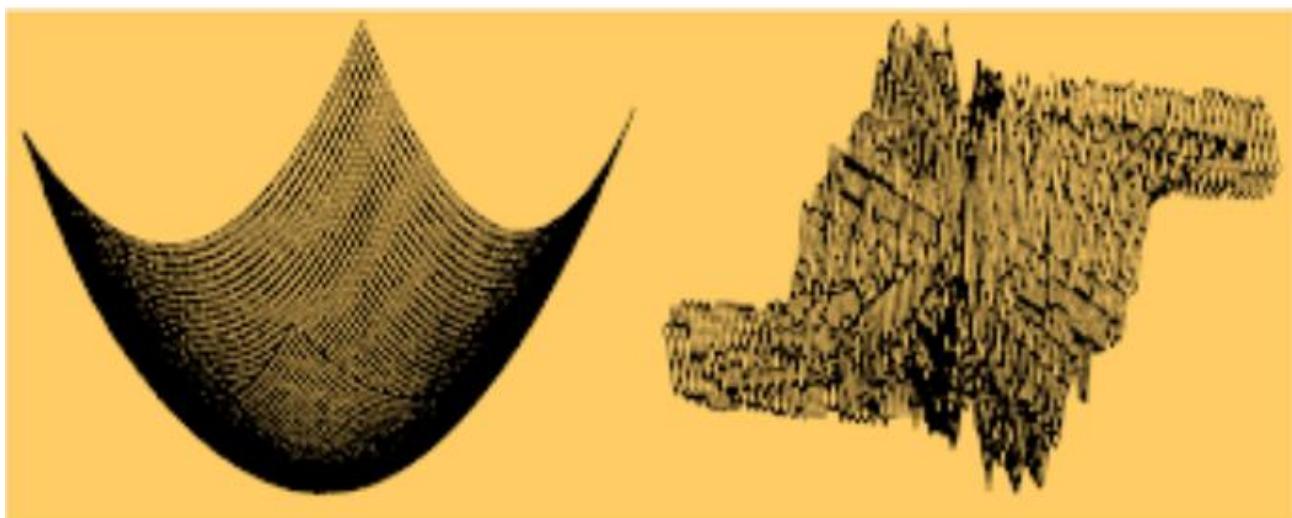
$$\langle E^2 \rangle = (1 / 2L) \sum_k e_k^2 \quad k = 1 \text{ a } L, \quad L \text{ es la cantidad de entradas.}$$

es un valor aceptable termina el algoritmo, sino se vuelve al paso 1.



Hay una gran diferencia entre el cálculo del error en la red perceptron y en adaline ya que el error en la red perceptron se calcula como la diferencia entre la salida deseada y la salida de la red siendo esta última binaria con lo cual sólo tiene en cuenta si se ha equivocado o no.

En el caso de adaline el error es un valor numérico y permite medir **cuanto se ha equivocado** la red, siendo el cálculo del error a través del error cuadrado medio.

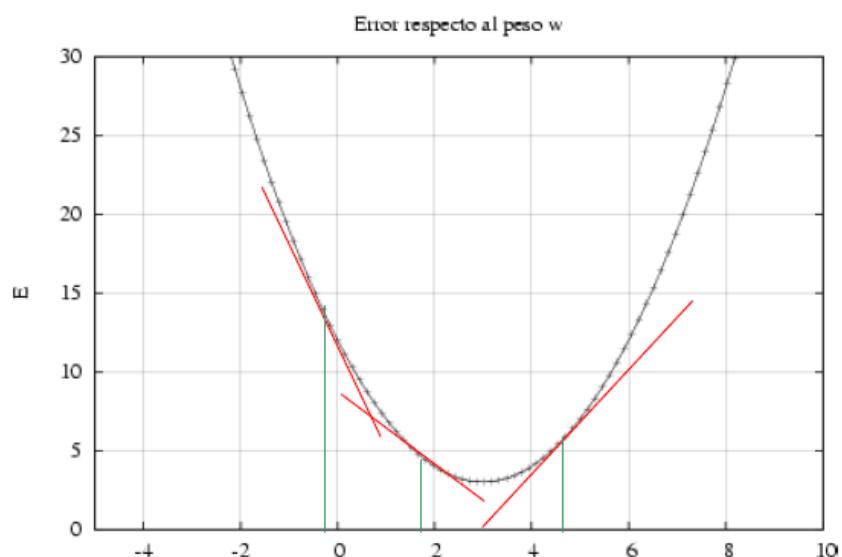


Superficie del error Red Adaline

Superficie del error Red Perceptron

La regla de aprendizaje de ADALINE es la regla Delta, que busca el conjunto de pesos que minimiza la función de error, la idea es realizar un **cambio en cada peso proporcional a la derivada del error**, medida en el patrón actual, respecto del peso:

$$\Delta_p w_j = -\gamma \frac{\partial E^p}{\partial w_j}$$

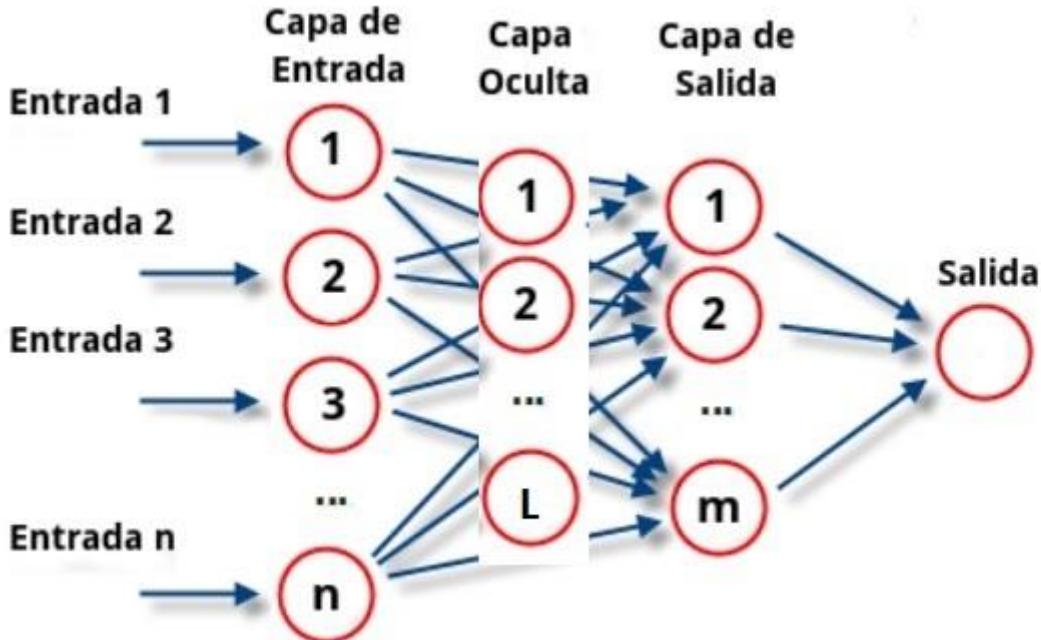


Aplicando la regla de la cadena queda:

$$\Delta_p w_j = \gamma(d^p - y^p)x_j$$



**Algoritmo de red backpropagation** red multicapa con conexiones hacia atrás con aprendizaje supervisado



- 1) Establecer el error aceptable para cada entrada
- 2) Inicializar los pesos de la red con valores pequeños aleatorios
- 3) Presentar la entrada  $X_p = (X_{p1}, X_{p2}, \dots, X_{pn})$   $i = 1 \text{ a } n$ , y especificar la salida deseada que debe generar la red:  $d_1, d_2, \dots, d_m$
- 4) Calcular la salida de la red, primero las salidas de la **capa oculta**

$$net^h = \sum_{pj} w^{hj} x_{pi} \quad i = 1 \text{ a } n \quad (\text{cantidad de componentes de cada entrada}) \\ j = 1 \text{ a } L \quad (\text{cantidad de neuronas ocultas})$$

**h** número de capa oculta    **p** p-esimo vector de entrada    **j** j-esima neurona oculta

$$y_{pj} = f^h (net^h)$$

Se realizan los mismos cálculos para obtener las salidas de las neuronas de la **capa de salida**

$$net^o = \sum_{pk} w^{ok} y_{pj} \quad j = 1 \text{ a } L \quad (\text{cantidad de neuronas ocultas})$$



$$y_{pk} = f^o \circ (\text{net}^o) \quad k=1 \text{ a } m \quad (\text{cantidad de las neuronas de salida})$$

5) Calcular el error para la **capa de salida**

$$\delta^o = (d_{pk} - y_{pk}) f^{o'} (\text{net}^o) \quad k=1 \text{ a } m$$

Calcular el error de la **capa oculta**

$$\delta^h = f^{h'} (\text{net}^h) \sum_{pj} \delta^o w^o_{pj} \quad k=1 \text{ a } m \quad (\text{cantidad de neuronas de salida})$$

El error en una neurona oculta es **proporcional a la suma de los errores conocidos** que se producen en las neuronas de salida ponderada por el peso.

6) Actualización de los pesos

Para las neuronas de la **capa de salida**:

$$W^o(t+1) = W^o(t) + \alpha \delta^o y_{pj} \quad k=1 \text{ a } m \quad (\text{cantidad de neuronas de salida})$$

y para los pesos de las neuronas de la **capa oculta**

$$W^h(t+1) = W^h(t) + \alpha \delta^h x_{pi} \quad j=1 \text{ a } L \quad (\text{cantidad de neuronas ocultas})$$

6) El proceso se repite hasta que el término de error

$$E_p = \frac{1}{2} \sum_{pk} (\delta^o)^2 \quad \text{para } k=1 \text{ a } m$$

resulta aceptable pequeño para cada entrada

Esta red backpropagation utiliza la regla delta de aprendizaje generalizada.