U2: RAZONAMIENTO EN AMBIENTES DETERMINISTAS

METAHEURÍSTICAS

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).

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.

Al diseñar una metaheurística se deben tener en cuenta dos **criterios contradictorios**: la exploración del espacio de búsqueda (diversificación) y la explotación de las mejores soluciones encontradas (intensificación).

Las regiones prometedoras se determinan a través de las soluciones buenas obtenidas. La **intensificación** se refiere a la exploración más a fondo de las regiones prometedoras con la esperanza de encontrar mejores soluciones. La **diversificación** implica la vista de regiones no exploradas para asegurarse que todas las regiones dle espacio de búsqueda se exploren de manera equitativa y de que la búsqueda no se limite solo a una zona reducida.

Criterios de clasificación:

- **Insipiradas en la naturaleza vs no inspiradas en la naturaleza**: muchas metaheurísticas están inspiradas en procesos naturales.
- **Con memoria vs sin memoria:** alguno algoritmos metaheurísticos no recuerdan, es decir, no utilizan información extraída durante la búsqueda. El otro grupo sí se apoya en esa información.
- **Deterministas vs estocásticos**: en métodos deterministas la solución utilizada como punto de partida lleva siempre a la misma solución final. Los métodos estocásticos tienen la posibilidad de tomar decisiones distintas para la misma situación, lo que les da mayor variabilidad en los caminos de búsqueda.
- **Basados en población vs basados en una solución**: los basados en una sola solución manipulan y transforman una sola solución durante la búsqueda, mientras que los basados en población se adapta todo un conjunto de soluciones. Las basadas en una solución están orientadas a la explotación; tiene la capacidad de intensificar la búsqueda en regiones locales. Las basadas en población estan orientas a la exploración; permiten una mejor diversificación en todo el espacio de búsqueda.

- **Iterativos vs avaros**: los iterativos comienzan con una solución completa (o población de soluciones) y la transforman en cada iteración. Los avaros parten de una solución vacía y, en cada paso, asignan una variable de decisión del problema has obtener una solución completa. La mayoria de las metaheurísticas son algoritmos iterativos.

Tipos fundamentales:

- **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.
- **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.
- **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.
- **Las metaheurísticas evolutivas:** están enfocadas a los procedimientos basados en conjuntos de soluciones que evolucionan sobre el espacio de soluciones.

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**.

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’.

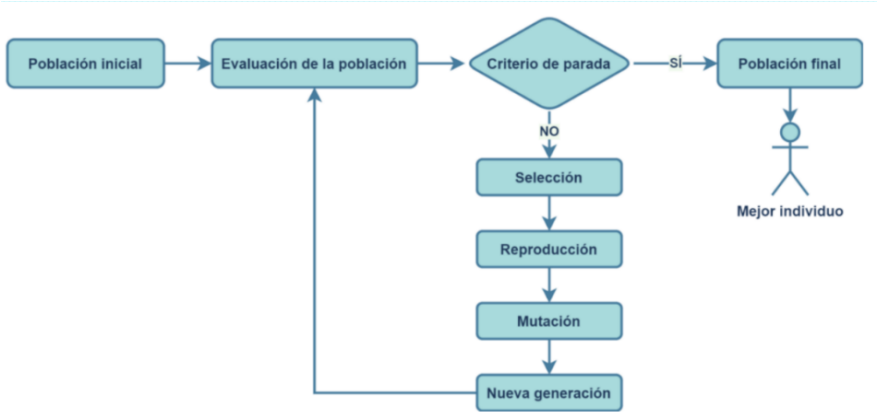
Este modelo básicamente dice que, dentro de una población, los individuos que sobreviven son aquellos que están más adaptados al medio, por lo tanto, las generaciones

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.

USOS Y ESTRUCTURA

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

- **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.



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.

VENTAJAS Y DESVENTAJAS

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.

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

U5: APRENDIZAJE AUTOMÁTICO

GENERALIDADES

Podemos distinguir dos **categorías de reconocimiento**, a saber, reconocimiento de objetos concretos y reconocimiento de objetos abstractos.

En el primer caso se trata de **reconocimiento perceptual**, como por ejemplo una forma (patrón espacial) o una secuencia (patrón temporal). **Objetos concretos**.

En el segundo caso se trata de **reconocimiento conceptual**, tal como un viejo argumento o la solución de un problema. **Objetos abstractos**.

Puede tener 2 **objetivos**:

1. Predicción: La mayoría de los casos, el objetivo es la clasificación.
2. Comprender la relación entre las variables.

Patrón: descripción de un objeto definido como una relación entre sus características. Conjunto mínimo de características comunes a un universo de datos que permite identificar dichos datos como pertenecientes a diferentes clases.

El reconocimiento de patrones está asociado al reconocimiento perceptual. Cuando una persona percibe un patrón, realiza una inferencia inductiva y asocia esta percepción con algunos conceptos generales o pistas derivados de su experiencia pasada.

El **problema de reconocimiento** puede ser concebido como el de discriminar, clasificar o categorizar la información de entrada, no entre patrones individuales sino entre poblaciones, por medio de la búsqueda de características o atributos invariantes entre los miembros de una población.

El **reconocimiento humano** es la estimación del parecido relativo entre datos de entrada y poblaciones conocidas. El **reconocimiento automático** es la clasificación de datos de entrada entre poblaciones mediante la búsqueda de características o atributos invariantes entre los miembros de cada población.

APLICACIONES

El reconocimiento de patrones provee una teoría general cuyas aplicaciones son múltiples. Algunas se enumeran en la tabla siguiente:

Tarea de clasificación	Datos de entrada	Salida
Reconocimiento de Caracteres	Señales ópticas o líneas	Nombre del carácter
Reconocimiento del habla	Ondas acústicas	Nombre de la palabra
Reconocimiento de voz	Voz	Nombre de quien habla
Predicción del clima	Mapas climáticos	Pronóstico climatológico
Diagnóstico médico	Síntomas	Enfermedad
Predicción de Acciones	Gráficos financieros	Predicción de alzas y bajas

PROBLEMAS A RESOLVER

El diseño de un sistema de reconocimiento automático involucra por lo general las tareas siguientes (etapas para reconocimiento de patrones):

- Sensado.
- Extracción de Características.
- Clasificación.

SENSADO

Sensado se refiere a la representación de la información obtenida mediante algún tipo de sensor sobre los objetos a ser reconocidos. Cada cantidad medida describe una característica del objeto. Esto puede evidenciarse suponiendo, por ejemplo, que se obtiene información sobre caracteres alfanuméricos.

Es la obtención de datos.

En este caso puede considerarse un **esquema de medición de grilla** como el mostrado en el lazo izquierdo de la figura.

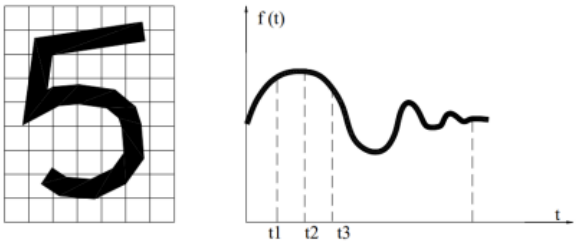


Fig. 2.1: Los objetos se representan mediante un vector patrón.

Esta grilla puede representarse mediante un vector patrón como sigue:

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad con \quad x_i = 0 \quad o \quad x_i = 1$$

A la salida es común que la veamos como una escalar, pero también puede ser un vector.

Donde se asigna a xi el valor 1 si el elemento forma parte del carácter y 0 en caso contrario. En el caso de una imagen, xi sería la intensidad del píxel i.

Variables cuantitativas: Se representan como números del tipo que sea necesario. En el primer ejemplo, peso, altura y edad son variables cuantitativas y se podrían representar con un float. En una imagen, las intensidades de los píxeles se pueden representar con enteros.

Variables categóricas o cualitativas: Pueden tomar solo un valor dentro de un conjunto limitado de valores. En el caso del nivel educativo, estos valores podrían ser "ninguno", "primario", "secundario", "terciario", "universitario o superior".

Las **variables categóricas pueden ser ordinales o nominales**. En las **ordinales** los valores que pueden tomar pertenecen a una escala (hay algunos superiores a otros, existe una jerarquía), como por ejemplo, en el caso del nivel educativo.

EXTRACCIÓN DE CARACTERÍSTICAS

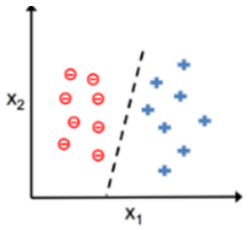
- Creación de nuevas características.
- Eliminación de características.
- Reducción de la dimensión.
- Otras transformaciones.

CLASIFICACIÓN

- Determinar a qué clase pertenece cada vector de entrada.
- Generación de límites de decisión entre regiones.
- **Clasificador**: función que permite separar objetos represnetados por patrones como pertenecientes a diferentes clases.

Toda la información medida disponible acerca de los patrones está contenida en los vectores de patrones. Cuando los vectores de patrones están **formados por números reales** es útil interpretarlos como puntos en el espacio Euclidiano n-dimensional.

En casos sencillos, el conjunto de puntos que corresponden a una misma clase resulta así agrupados en alguna región del espacio. A este agrupamiento suele llamársele **Cluster**.



Enfoque de las funciones de decisión

$d_i(\mathbf{x}) > d_j(\mathbf{x})$ para $i, j = 1, 2, \ldots, M$ y $j \neq i$

Un **problema en el área del reconocimiento de patrones** está referido a la reducción de la dimensión de los vectores de los patrones. Se suele mencionar a esta etapa como preproceso o extracción de características.

Otro problema es la determinación de un procedimiento óptimo de clasificación. Para lograr esto, se requiere diseñar un 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.

Suponiendo el sistema diseñado para reconocer **M clases diferentes**, denominadas w1, w2, …, wn puede considerarse al espacio de patrones compuesto por M regiones cada una de las cuales corresponde a los patrones de una clase.

La solución puede interpretarse como generar los límites de decisión entre las regiones, Estos pueden estar dados por las llamadas funciones de decisión o funciones discriminantes d1(x), d2(x), dn(x), las que son funciones escalares de cada vector x.

Si una función tiene mayor valor entre todas las demás, entonces el vector pertenece a la clase correspondiente a esa función, es decir si di(x) > dj(x) para i,j = 1, 2, …, M y j distinto de i, entonces el patrón x pertenece a la clase wi.

Es decir que la **pertenencia a una clase** es determinada por el mayor valor de la función discriminante di(x). Tal sistema de clasificación es esquematizado en la figura incorporando un proceso de ejecución de decisiones.

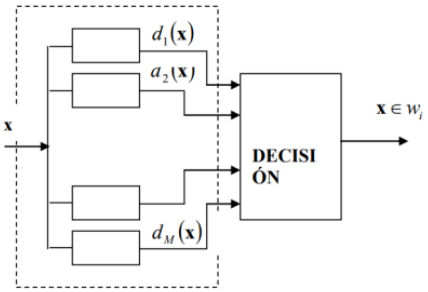


Fig. 2.3 Las funciones de decisión

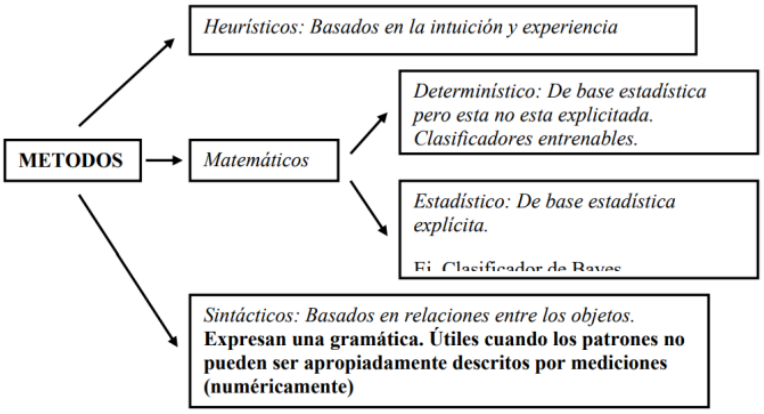
En la figura se representa esquemáticamente lo que ocurre al emplear funciones discriminantes en la clasificación. Las funciones discriminantes están representadas individualmente, esto no debe interpretarse en el sentido de que realmente es posible obtener definiciones analíticas de cada una de estas funciones si no que de alguna manera se establecen las mismas.

Por otra parte, muchas veces el proceso de establecer estas funciones se lleva a cabo en forma conjunta y simultánea y por lo tanto solo disponemos de un único sistema que generará las salidas de la manera deseada tal como se representa a la salida del bloque en línea de trazos.

En este último caso solo tendremos una **caja negra** con vectores de entrada (patrones a reconocer) y vectores de salida (patrones que representan las clases de pertenencia de los patrones).

Este último sistema constituye el núcleo del sistema de reconocimiento. A los efectos de tener una visión lo más general posible debemos considerar que el sistema debe hacerse cargo además de resolver los dos problemas anteriores (sensado y selección de características) y clasificación.

El análisis de contexto es particularmente importante en temas tales como reconocimiento del habla y escritura manual.



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

Si todas las representaciones posibles de aquel objeto que se desea clasificar, o un gigantesco número de ellas estuvieran disponibles, si toda la información contenida en el objeto pudiera ser extraída, y si el tiempo de desarrollo no tuviera límites seguramente un

sistema de fuerza bruta podría resolver el problema, pero por lo general en la práctica las condiciones de trabajo distan mucho de ser estas. Las restricciones se dan por lo común en términos de costos, tiempo y tecnología existente.

El resultado de las etapas de sensado y extracción de características es un conjunto de vectores de características. Cada vector tiene la forma:

$$x=[x1,x2,...,xn]$$

donde xi es el valor de la característica i.

Aclaración de la nomenclatura:

- *u* es un escalar.
- ***u*** (en negrita) es un vector formado por escalares ui
- ***U*** es una matriz formada por escalares *u*ij

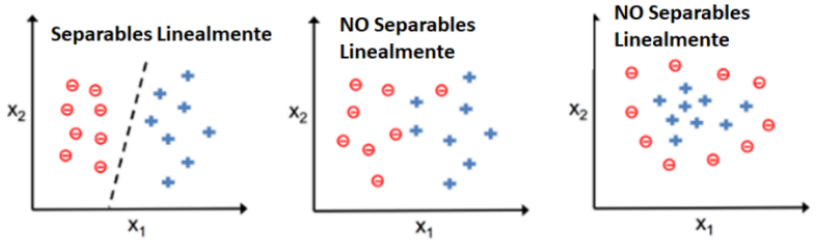
Después de obtener los vectores de características se realiza una de las siguientes tareas:

- **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 (continuo) para cada vector de características. Por ejemplo, para calcular el valor futuro de ciertas acciones bursátiles.

Los clasificadores tienen parámetros internos que determinan la salida. El proceso de ajuste de estos parámetros se llama entrenamiento. 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.

TIPOS DE PROBLEMAS REALES



El último grafico es de un problema no separable.

FUNCIÓN DE DECISIÓN LINEAL

El clasificador lineal pertenece al grupo de los **métodos de aprendizaje supervisado**, es decir, 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 a una de las clases conocidas, en caso contrario, a la otra.

La salida de d(x) es una **escalar**.

Si las entradas de un clasificador lineal tienen la forma **x = [x1, x2, …, xn],** la función de decisión, se define como:

$$d(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{n+1}$$

O con vector de entras aumentadas, x = [x1, x2, …, xn, 1],

$$d(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + x_{n+1} w_{n+1}$$

CÁLCULO CON VECTORES

El vector de entradas x es un vector fila. Si representamos el conjunto de coeficientes con un vector columna w = [w1, w2, …, wn+1]^T, entonces

$$d(\boldsymbol{x}) = \boldsymbol{x}\boldsymbol{w}$$

Es común calcular d(x) al mismo tiempo para todos los vectores de entrada. En ese caso,

$$d(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{w}$$

donde **X** es la matriz cuyas filas son los vectores de entrada y el resultado de d(x) es un vector columna.

$$\begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & 1 \\ x_1^2 & x_2^2 & x_3^2 & 1 \\ x_1^3 & x_2^3 & x_3^3 & 1 \\ x_1^4 & x_2^4 & x_3^4 & 1 \\ x_1^5 & x_2^5 & x_3^5 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} d^1 \\ d^2 \\ d^3 \\ d^4 \\ d^5 \end{bmatrix}$$

CÁLCULO DE LOS COEFICIENTES

Para entrenar el modelo es necesario conocer la clase a la que pertenece cada uno de los vectores x. Supongamos que estas clases son c1 y c2. Se desar que d(x) > 0 para c1 y d(x) < 0 para c2, entonces se crea una **salida esperada y** para cada vector, donde

$$y = \begin{cases} +1 & \text{si } \boldsymbol{x} \in c_1 \\ -1 & \text{si } \boldsymbol{x} \in c_2 \end{cases}$$

y es una escalar, pero si de contempla la salida esperada para todos los vectores de entrada, se obtiene el vector columna y

x_1^1	x_2^1	x_3^1	1
x_1^2	x_2^2	x_3^2	1
x_1^3	x_2^3	x_3^3	1
x_1^4	x_2^4	x_3^4	1
x_1^5	x_2^5	x_3^5	1

y^1
y^2
y^3
y^4
y^5

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:

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

donde, **m**: cantidad de vectores de entrenamiento y **n**: cantidad de características.

Haciendo $\frac{\partial S^2(\boldsymbol{w})}{\partial \boldsymbol{w}} = 0,$ se llega a la siguiente expresión para el cálculo de los coeficientes (para vectores de entrada definidos como filas):

$$\boldsymbol{w} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y}$$

donde,

$\boldsymbol{w}$: vector de pesos

$\boldsymbol{X}$: matriz de entradas

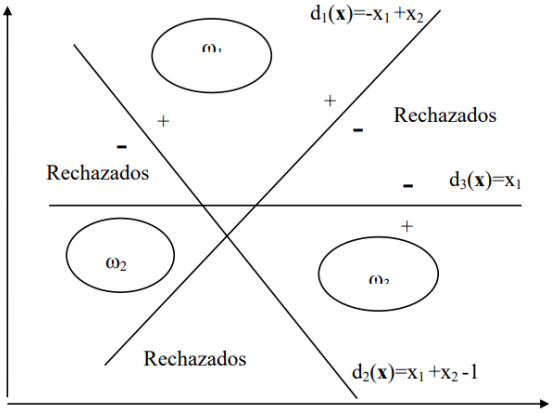
$\boldsymbol{y}$: vector de salida

La **desventaja** más importante radica en que solo puede clasificar clases linealmente separables.

CLASIFICACIÓN MULTICLASE

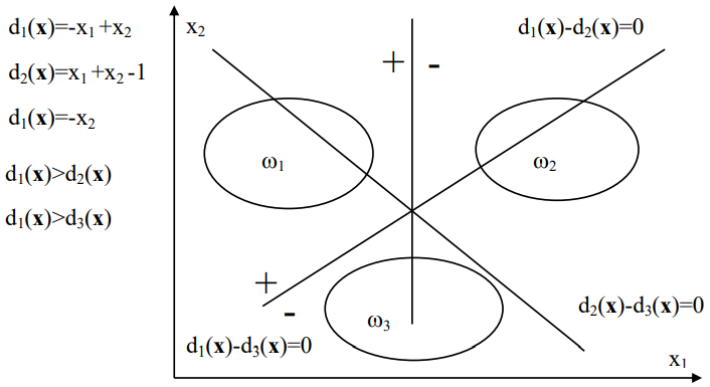
Uno contra todos (Piecewise)

- Cada clase tiene su propia función de decisión.
- Cada función se entrena para separar los vectores propios del resto.
- La **salida** queda definida por la función de decisión de mayor valor para un vector determinado.
- Las fronteras quedan definidas por las intersecciones entre las funciones de decisión.



Clasificador por pares (Pairwise)

- Hay una función de decisión por cada par de clases (N (N - 1) / 2).
- Cada función se entrena para separar los vectores del par correspondiente.
- La **salida** del clasificador queda definida por la salida con frecuencia más alta entre todas las funciones de decisión.



FUNCIÓN DE DECISIÓN GENERALIZADA

La función de decisión lineal para un vector de entradas x = [x1, x2, …, xn], definida como

$$d(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{n+1}$$

solo puede resolver problemas de clasificación donde las clases son linealmente separables. Un enfoque para extender este clasificador a problemas más complejos es el de la función de decisión generalizada, de la forma

$$d(\boldsymbol{x}) = w_1 f_1(\boldsymbol{x}) + w_2 f_2(\boldsymbol{x}) + \ldots + w_m f_m(\boldsymbol{x}) + w_{m+1}$$

donde **fi(x)** es una **función real simple** del vector de entradas x.

La última ecuación representa un número infinito de funciones de decisión. La capacidad de reconocimiento y la forma de la frontera de decisión dependerá de la elección de las funciones fi(x) y de la cantidad de términos en d(x).

Una vez elegidas las funciones fi(x), el **cálculo de los coeficientes wi** se puede realizar la misma forma que en el clasificador lineal. Para eso es necesario una transformación en el vector de entradas. Se define un nuevo vector **x*** cuyos componentes son las funciones fi(x)

$$\boldsymbol{x}^* = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x})]$$

o si se utiliza un vector de entradas aumentadas

$$\boldsymbol{x}^* = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}), 1]$$

Una vez calculadas las funciones fi(x), el cálculo de los coeficientes es el mismo que en la función de decisión lineal.

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:

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

Lo que podría transformarse a forma lineal como

$$d(\boldsymbol{x}^*) = \boldsymbol{w}\boldsymbol{x}^*$$

donde

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

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

Consideraciones:

- Al aumentar la cantidad de características, el clasificador se hace muy complejo y no se puede entrenar de forma directa.
- Existe una relación entre la evolución de los modelos neuronales y los obetnidos mediante la función de decisión generalizada.

MÁQUINAS DE VECTORES DE SOPORTE (SVM)

- **Clasificadores de márgen óptimo**: se maximiza la mínima distancia entre las clases y el hiperplano discriminante.
- **Clasificadores de vectores de soporte**: se consideran solo los vectores de entrenamiento que están muy próximos a la frontera entre las clases.
- **Máquinas de vectores de soporte**: se reemplaza el kernel lineal (producto escalar) por otros no lineales.

MATRIZ DE CONFUSIÓN

Luego de resolver con un clasificador, es importante evaluarlo, que tán bueno es. Se la matriz de confusión con los valores predicos en columnas y los reales en las filas.

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

FACTOR DE CLASIFICACIÓN

Porcentaje de casos bien clasificados.

$$FC = \frac{a}{b}$$

donde

- a = suma de los elementos diagonales de M (VP + VN).
- b = suma de todos los elementos de M (VP + VN + FP + FN)

SENSIBILIDAD - RAZÓN DE VP

Proporción de casos positivos que fueron correctamente identificados

$$VPR = \frac{VP}{P} = \frac{VP}{VP + FN}$$

P = todos los positivos.

RAZÓN DE FP

$$FPR = \frac{FP}{N} = \frac{FP}{FP + VN}$$

N = todos los negativos.

EXACTITUD

Lo cerca que está el resultado de una medición del valor esperado, cantidad de predicciones positivas que fueron correctas.

$$ACC = \frac{VP + VN}{P + N}$$

ESPECIFICIDAD - RAZÓN DE VN

Casos negativos que el algoritmo ha clasificado correctamente.

$$SPC = \frac{VN}{N} = \frac{VN}{FP + VN} = 1 - FPR$$

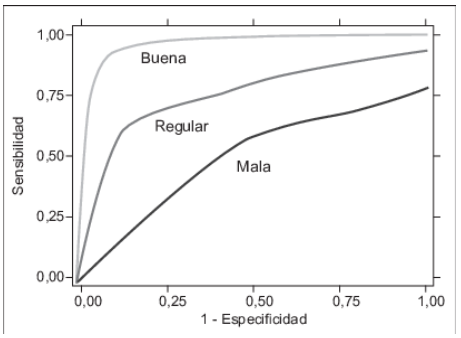
PRECISIÓN

Dispersión del conjunto de valores obtenidos. Porcentaje de casos positivos detectados.

$$Precisión = \frac{VP}{VP + FP} = \frac{VP}{P^*}$$

donde P* = todos los clasificados como positivos.

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 **FRP = 0** y **sensibilidad = 1**.

Ayuda a encontrar el umbral de clasificación que se adapte a nuestro problema. El **área debajo de la curva** determina lo bueno de un clasificador.

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.

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".

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:

- 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.

- **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.

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.

CARACTERÍSTICAS FUNDAMENTALES

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

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.

Redes monocapa : se establecen conexiones laterales entre las neuronas que pertenecen a la única capa que constituye la red. Ejemplos red de HOPPFIELD. 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).

Mecanismo de 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**:

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

Mecanismos de aprendizaje: Aprendizaje supervisado y 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 ONLINE) o requiere de una fase previa de entrenamiento (aprendizaje OFFLINE). **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 ONLINE** 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 perceptron:

- Regla de aprendizaje del perceptron: 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 la información 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 [(A1,B1), (A2,B2),....(An,Bn)], de tal forma que cuando se le presente determinada información de entrada Ai responda con la salida correspondiente Bi. 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 A1, A2, .. ,An 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

Forma de representación de las informaciones

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 continúa pero a la salida ofrecen información binaria.

PERCEPTRÓN

Un perceptrón es una neurona artificial, y por lo 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.

La **salida** del perceptrón se define como:

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

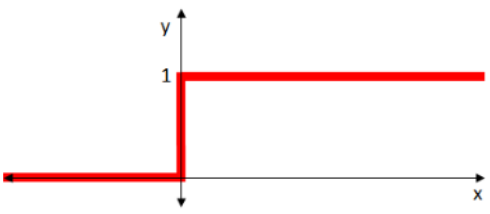
donde:

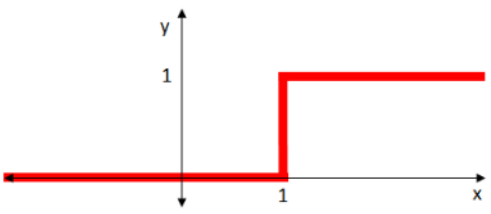
x_i es el elemento i del vector de entradas $\boldsymbol{x} = [x_1, x_2, \ldots, x_N]$

w_i es el elemento i del vector de pesos $\boldsymbol{w} = [w_1, w_2, \ldots, w_N]$

θ es el umbral de activación.

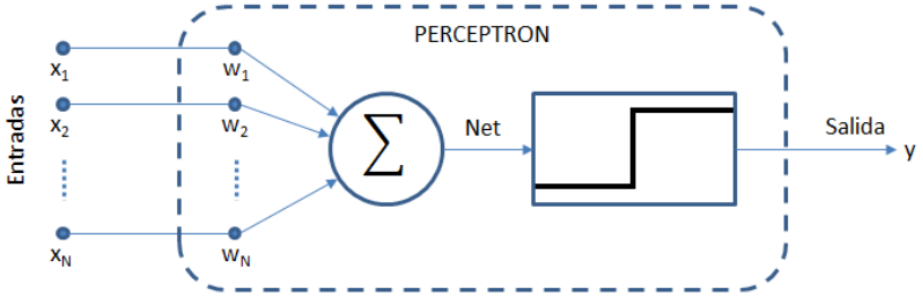
f es la función umbral (también *threshold* o *hard-lim*). $f(z) = \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases}$





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.

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.



CON VECTOR DE ENTRADAS AUMENTADO

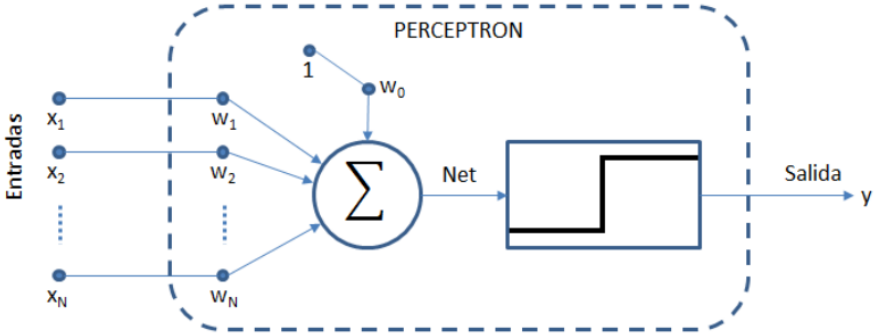
La salida del perceptrón se define como:

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

donde:

$w_0 = -\theta$

En lugar de decir que expandemos el vector de entrada x, se adiciona a la neurona una entrada que siempre recibe el valor 1. Sin el término w0, siempre pasaría por el 0 en el eje, ya que es la ordenada al origen.



ENTRENAMIENTO

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

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

donde **y'** es la **salida esperada**. Los pesos se adaptan según la **ley de aprendizaje** (función recursiva que permite alcanzar el valor apropiado para la matriz de coeficientes):

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

donde **α** es la **tasa de aprendizaje**.

Pasos (el proceso es iterativo, la condición de corte es que no haya error (y' - y) o que el error sea 0):

1. Definir el valor de α.
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 persos haciendo $w_{i_{t+1}} = w_{i_t} + \Delta_{w_i}$

El **entrenamiento termina** cuando todos los vectores están bien clasificados.

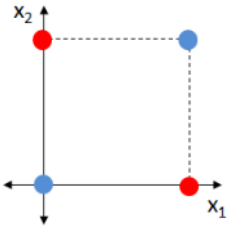
La **desventaja**, es que al igual que el clasificador lineal, el perceptrón no puede clasificar correctamente si las clases no son linealmente separables.

PROBLEMA DE LA XOR

Este problema **NO se puede solucionar**, el perceptron 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.

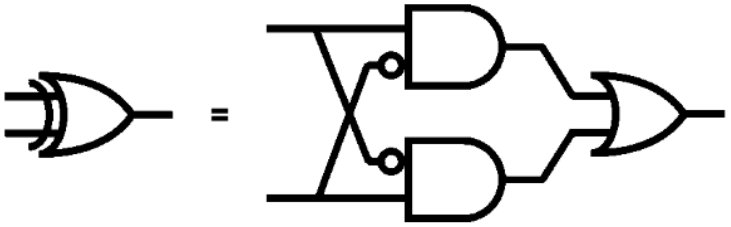
Conectando perceptrones entre sí, en lo que se llama **perceptron multicapa**, se podría resolver el problema si supiéramos cómo adaptar los pesos. El valor de este modelo es únicamente educativo.

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



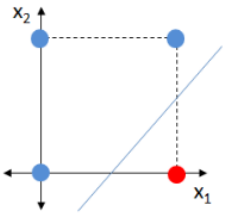
Con perpectrón multicapa

Una compuerta XOR se puede construir con otras compuertas. El **enfoque** para tratar el problema con perceptrones es entrenar percetrones individuales para que funcionen como cada una de las compuertas del circuito y después conectarlos.



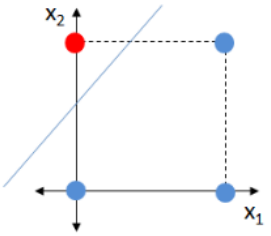
Primero hacemos que el perceptron P1 funcione como la compuerta C1 (AND con la segunda entrada negada).

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



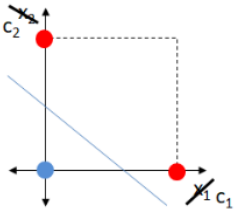
Lo mismo con el perceptron P2, lo entrenamos para que funcione como C2 (AND con la primera entrada negada).

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



Por último, entrenamos el perceptron P3 para que funcione como la compuerta C3 (OR).

x_1	x_2	c_1	c_2	y'
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0



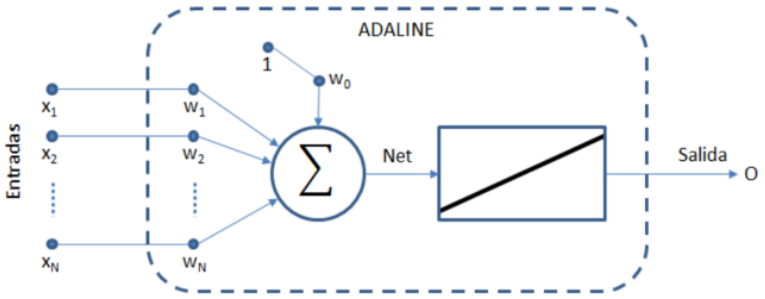
Lo importante a comprender es que el último perceptron resuelve correctamente el problema porque trabaja en un espacio distinto. Este nuevo espacio c1 - c2 fue creado por los dos perceptrones de la primera capa.

¿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.

ADALINE

Es un tipo de neurona con dos **diferencias importantes respecto a perceptrón**:

- Cambia el mecanismo de aprendizaje, se utiliza la **regla delta**, basada en el mínimo error cuadrático medio (Least Mean Squared o LMS error).
- Adaline tiene una **función de activación lineal**.



donde:

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

El **error cuadrático medio** se define como:

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

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.

La **regla delta** queda definida como

$$\Delta_{w_{ik}} = -\alpha\delta\, x_i$$

$$= -\alpha\left(-(y_k - O_k)\right) x_i$$

$$= \alpha(y_k - O_k)x_i$$

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.

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 númerico y permite medir cuanto se ha equivocado la red, siendo el cálculo del error a través del error cuadrado medio.

MÉTODO DE ENTRENAMIENTO

El método de entrenamiento también es similar al del perceptron, 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.

BACKPROPAGATION

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

Regla delta:

Recordar que

$$\Delta_{w_i} = -\alpha\delta x_i$$

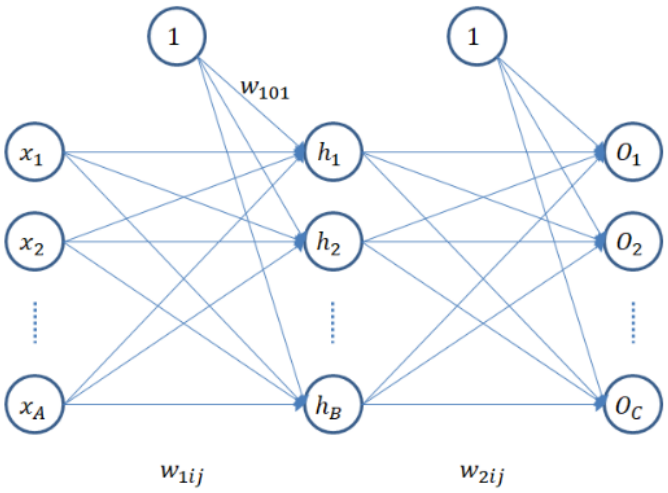
$$\delta = \frac{\partial E_k}{\partial O_k}\frac{\partial O_k}{\partial Net_k} = -(y - O)\frac{\partial O_k}{\partial Net_k}$$

En Adaline $O_k = f(Net_k) = Net_k$, entonces $\delta = -(y - O)$

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

RED MULTICAPA CON CONEXIONES HACIA ADELANTE

Con una solo capa oculta:



donde i es la i-ésima entrada de la neurona j.

- 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 capa oculta, la capa de salida y la de entrada).

FUNCIONES DE ACTIVACIÓN

Recordar que un Adaline, o cualquier neurona artificial (de las que vimos) 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 razon es necesario agregar no-linealidades entre las capas.

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

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

La derivada de la función sigmoidal es:

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

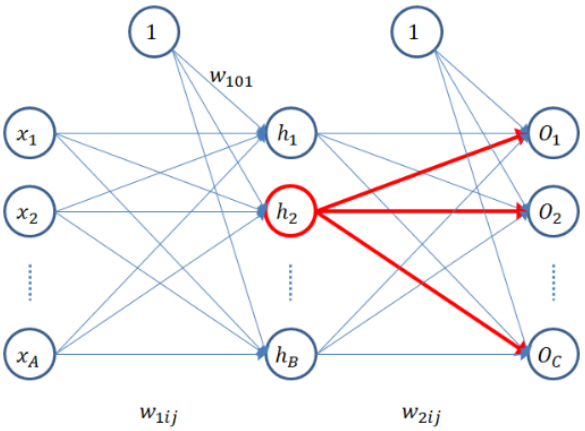
Entonces, por ejemplo, si las neuronas de salida de la red anterior tienen funciones de activación sigmidal,

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

y la derivada de la salida con respecto a Net:

$$\frac{\partial O_j}{\partial Net_j} = \frac{1}{1+e^{-Net_j}}\left(1-\frac{1}{1+e^{-Net_j}}\right) = O_j(1-O_j)$$

El nombre del método significa retro-propagación 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 erorr de todas las neuronas de las capas ocultas. Una vez calculados todos los errores, se modifican los pesos.

Pasos:

1. Definir el valor de α.
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 fución de su propio error.
        vii. Modificar los pesos.

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.

OTRAS CONSIDERACIONES SOBRE REDES NEURONALES

- Dificultad para elegir los hiperparámetros:
    - Arquitectura.
    - Inicialización de los pesos.
    - Tasa de aprendizaje.

- Adición de un momento

$$\Delta w_{i(t+1)} = \alpha \delta x_i + \beta \Delta w_{i(t)}$$

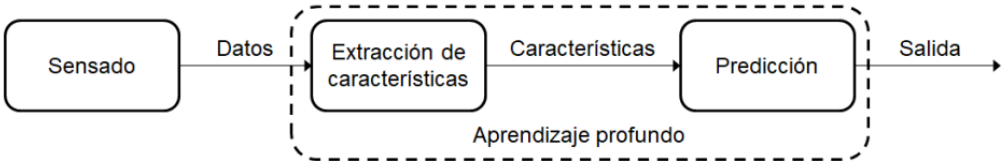
- Sobreentrenamiento.

APRENDIZAJE PROFUNDO

Es un conjunto de **herramientas de aprendizaje automático** que logra entrenar redes neuronales con varias capas de profundidad.

- 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 reconocimiento de patrones.
  - Visión artificial
  - Reconocimiento de audio
  - Procesamiento del lenguaje natural (NLP)
  - Sistemas de diagnóstico médico
  - Predicción de terremotos, de uso de energía, de valores de mercado, etc.
- También "amenaza" otras áreas de la IA.
- Es una solución end-to-end.

Una de las particularidades que distingue al aprendizaje profundo, y que probablemente sea una de las causas de su buen desempeño, es la capacidad para aprender representaciones complejas de caracter´ısticas y ajustar los par´ametros del clasificador al mismo tiempo.



END-TO-END

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

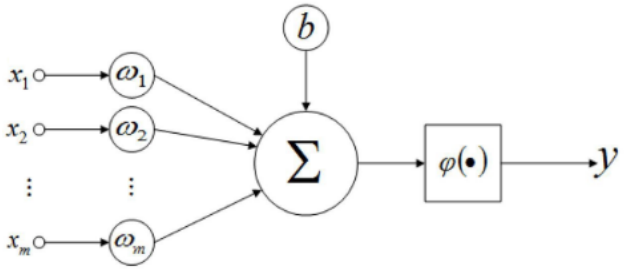
TIPOS MÁS COMUNES DE CAPAS

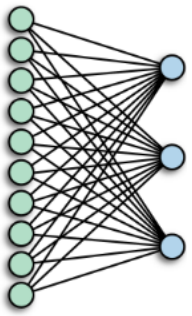
- Densamente conectadas
- Convolucionales
- Pooling (MaxPooling, AveragePooling)

- Recurrentes
- LSTM

Capas densas (dense, feed forward, fully connected)

- 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.

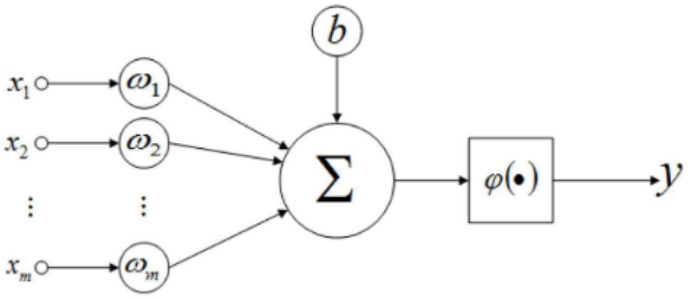




Capas convolucionales

- Cada neurona de una capa de convolución, al igual que las neuronas de las capas densas, realiza una transformación lineal de sus entradas antes de la aplicación de la función de activación.
- La **salida** se calcula como la aplicación de la función de activación sobre la suma ponderada de las entradas.

$$s_j = f(\sum_{i=0}^{N} w_{ij}\ x_i)$$



- 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

Kernels de convolución (filters, features maps)

Cada capa de convolución está compuesta por varios kernels que se especializan en detectar distintos patrones.

Capas de pooling

Las capas de agrupación o pooling tienen una mecánica similar a las capas de convolución, en el sentido de que cada elemento tiene su propio campo receptivo y realiza la misma operación que los elementos vecinos. Estas capas no se ajustan, no tienen pesos y su funcionamiento no cambia durante el entrenamiento. La **salida** de cada elemento se calcula como el valor máximo (max pooling) o el valor medio (average pooling) del campo receptivo.

A diferencia de las capas de convolución, donde usualmente los campos receptivos de una neurona se forman sobre todos los mapas de características de la capa anterior, en las capas de pooling generalmente existe un mapa de características por cada mapa de características de la capa anterior.

Mientras que la **función de la capa de convolución** es detectar conjunciones locales de características de la capa anterior, la **función de la capa de pooling** es fusionar varias características semánticamente similares en una.

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 invariante a la escala y a la traslación.

Neuronas recursivas

- La **salida** es retroalimentada pasando por un retardo.
- Procesan secuencias de longitud indefinida.

Long Short Term Memory (LSTM)

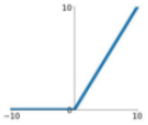
- Solucionan el problema de las dependencias Long-Term.
- Tienen la capacidad de añadir y eliminar información del estado de la célula.

COMPLICACIONES DEL APRENDIZAJE PROFUNDO

- Desvanecimiento del gradiente.
- Modelos más complejos (más parámetros).
    - Sobreentrenamiento.
    - Mayor necesidad de recursos.

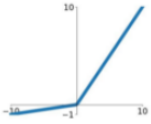
Desvanecimiento del gradiente

Una solución es cambiar la función de activación por una función que no sature.



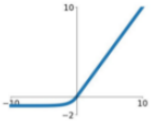
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

Sobreentrenamiento

Existe un comportamiento propio de las redes de retropropagación cuya manfiestación es que luego de un período de entrenamiento normal, el error tiende a aumentar nuevamente.

Esta particularidad puede ser empleada para detener el proceso de entrenamiento una vez que se ha detectado que se ha iniciado la fase de sobreentrenamiento.

Los modelos complejos son más propensos al sobreentrenamiento.

Algunas **formas de reducir el sobreenrenamiento**:

- **Early-stop**: En lugar de tomar la configuración final de los pesos, se puede obtener un modelo con un mejor error para los datos de validación (y por lo tanto, con suerte, un mejor error para el conjunto de test) volviendo a la configuración de parámetros en el momento donde el error de validación es mínimo. El método es simple, cada vez que mejora el error en el conjunto de validación, se almacena una copia de los parámetros del modelo. Cuando termina el proceso de entrenamiento, se devuelven estos parámetros, en lugar de los últimos parámetros. El algoritmo finaliza cuando ningún parámetro ha mejorado con respecto al mejor error de validación registrado para un número preestablecido de iteraciones.
- Aumentar la cantidad de datos de entrenamiento.
- **Regularización**: Cualquier modificación que podemos hacer sobre un algoritmo de aprendizaje con el objetivo de reducir su error de generalización pero no su error de entrenamiento.

$$L_1 : L(x, y) = E^2(x, y) + \lambda \sum_{i=1}^{n} |w_i|$$

$$L_2 : L(x, y) = E^2(x, y) + \lambda \sum_{i=1}^{n} w_i^2$$

- **Dropout**: previene la co-adaptación desmedida de las neuronas. Es una estrategia potente de regularización, donde se eliminan al azar algunas neuronas de las capas ocultas durante el entrenamiento. En cada ciclo, con una probabilidad previamente definida (generalmente por capas), se eligen ciertas neuronas y se anulan sus salidas