



## Apuntes SI - Resumen todos los temas SI

Sistemas Intelixentes (Universidade da Coruña)

## TEMA 1. INTRODUCCIÓN

### El nacimiento de la IA (1952-1956):

- La CIBERNÉTICA y las primeras RNA en 1940.
  - Investigación en Neurología: el cerebro era una red eléctrica de neuronas que se activaban en pulsos de todo o nada.
  - “Estudio de las analogías entre los sistemas de control y comunicación de los seres vivos. En particular en el dominio de las aplicaciones de los mecanismos de regulación biológica a la tecnología” (Wiener).
  - MIT -> Modelos que permitían establecer un conjunto de principios sencillos que expliquen las actividades de la mente humana.
  - Craik -> Analogías y modelos en la resolución de problemas.
  - Pitts -> Sistemas MIMO (Multiple Input Multiple Output) con RNAs.
  - Shannon -> El ordenador es un simulador de la actividad cerebral.
  - Neumann -> Arquitectura secuencial de los ordenadores.
- TRES ARTÍCULOS CLAVE:
  - Wiener, Rosenblueth y Biegelow sugieren cómo conferir fines y propósitos a las máquinas.
  - McCulloch y Pitts ponen de manifiesto de qué modo las máquinas pueden emplear la lógica y la abstracción, y demuestran que las leyes de entrada-salida pueden modelarse con RNAs.
  - Craik propone que las máquinas empleen modelos y analogías en la resolución de problemas.
- Test de Turing (1950):
  - ¿Puede pensar una máquina?
  - Incapacidad de diferenciar entre entidades inteligentes indiscutibles y seres humanos.
- Juegos e IA:
  - Damas y Ajedrez: habilidad suficiente para desafiar a un aficionado respetable.
- Logic Theorist (Newell y Herbert): Razonamiento simbólico.
- El proyecto de Investigación de verano de Inteligencia Artificial en Darmouth en 1956:
  - Interés: teoría de autómatas, redes neuronales y estudio de la inteligencia.
  - Asistentes: McCarthy, Minsky, Newell, Simon, Shannon, Samuel...

### La era de los éxitos: Los años dorados (1956-1974):

- La IA mostraba comportamiento inteligente a un nivel increíble en sus programas.
  - Razonamiento como búsqueda.
  - Lenguaje natural como interfaz.
  - Expectativas muy altas:
    - Se propone el Test de Turing.
    - Asimov escribe: I, Robot.
- Fallos en apreciar la dificultad del problema abordado.
- Desaparición de fondos de apoyo en investigación.
- Muerte del conexionismo durante más de 10 años.

### Llega el primer Invierno (1974-1980):

- Capacidad limitada de los programas de IA: Problemas juguete.
  - **Limitada capacidad de las computadoras.**
    - El exitoso trabajo de Quillian's en LN se demostró con un vocabulario de solo 20 palabras, el máximo número que cabía en memoria.
  - **La intratabilidad y la explosión combinatoria.**
  - **El conocimiento y el razonamiento de sentido común.**
  - **La paradoja de Moravec's.**
    - Las computadoras pueden derrotar a los mejores jugadores del mundo de ajedrez, pero todavía no podemos hacer que piensen como un niño de 4 años.
  - **El problema del frame y la cualificación.**
- Los informes Lighthill y ALPAC (DARPA) -> Aportación económica parada.

### Vuelve la primavera: El Boom (1980-1987):

- El conocimiento es el foco:
  - Nacen los sistemas expertos (Feigenbaum y DENDRAL).
    - La palabra de moda es “conocimiento”.
    - Millones de dólares de ROI (retorno en inversión).
  - Renace el conexionismo (Hopfield y Rumelhart).
    - Productos comerciales.

### El segundo invierno (1987-1993):

- Colapso del mercado del hardware especializado para IA.
- Mantenimiento y actualización caros en los sistemas expertos.
- Nicho estrecho.
- Corte en el apoyo financiero.
- Expectativas demasiado altas: Proyecto 5º Generación en Japón.
- Rechazo del modelo de procesado simbólico de la mente -> tesis de mente embebida en el cuerpo.

### El éxito entre bambalinas (1993-2001):

- Precaución y éxito:
  - Deep Blue, DARPA Grand Challenge, DARPA Urban Challenge...
    - Crece la capacidad de las computadoras.
    - Enfoque a problemas específicos aislados.
  - Se consiguen algunas de las viejas metas.
  - Fragmentación en subcampos que compiten:
    - Agentes Inteligentes.
    - Victoria de los *pulcros*: Herramientas matemáticas sofisticadas.
    - IA encubierta en otras aplicaciones.

### La nueva primavera (2000-actualidad):

- Explosión del Big Data.
- Aproximaciones disruptivas: *Deep Learning*.
  - Alpha Go (2015): entrenado a partir de millones de miles de partidas jugadas entre humanos.
  - Alpha Go Zero (2017): aprendizaje por refuerzo.

### Definiendo la Inteligencia:

- Para MALRAUX la inteligencia es la posesión de los medios necesarios para dominar cosas y hombres.
- Para MINSKY la inteligencia es la capacidad para resolver problemas que aún no se entienden.
- Para HASSENSTEIN la inteligencia:
  - No está condicionada por desencadenantes innatos, ni por adaptaciones a situaciones concretas.
  - Se caracteriza porque toda situación nueva, una vez percibida, se domina sin recurrir al ensayo.
  - Permite representar in mente distintas situaciones y sus interpretaciones de cara a la resolución de un problema.

- **DEFINICIÓN FENOMENOLÓGICA DE INTELIGENCIA:**
  - Los seres inteligentes se comunican.
  - Los seres inteligentes tienen conocimiento interno (autoconocimiento).
  - Los seres inteligentes tienen memoria y son capaces de procesar nuevas experiencias.
  - Los seres inteligentes tienen intencionalidad.
  - Los seres inteligentes son creativos.
  - Los seres inteligentes infieren y razonan.
- Inferencia es la comprensión de un significado en función de información relacionada.
- Razonamiento es una colección de inferencias conectadas.

### Definiciones de Inteligencia Artificial:

- Ciencia que estudia los mecanismos generales necesarios para lograr que los ordenadores hagan cosas que, por ahora, los humanos hacemos mejor.
- Disciplina encargada de diseñar máquinas que sean capaces de realizar tareas que, de ser realizadas por humanos, requerirían inteligencia.
- Rama de las ciencias de la computación que intenta encontrar esquemas generales de representación del conocimiento, y formalizar procesos de razonamiento coherentes, que permitan resolver problemas difíciles en dominios de aplicación concretos.
- Ciencia que utiliza elementos simbólicos y numéricos, conjuntos semánticos, procesos heurísticos, y mecanismos inferenciales lógicos, para emular los procesos cognoscitivos y el razonamiento de los humanos.
- El diccionario Merriam Webster:
  - Una rama de la ciencia de la computación que trata de simular comportamiento inteligente en computadoras.
  - La capacidad de una máquina para imitar comportamiento inteligente humano.

## INTRODUCCIÓN:

- Planteamiento científico de la IA:
  - Como ciencia, la IA trata de desarrollar vocabulario, conceptos y métodos que ayuden a comprender y a reproducir comportamiento inteligente.
- Planteamiento ingenieril de la IA:
  - Como ingeniería, la IA define y utiliza un conjunto de métodos que nos permiten adquirir conocimiento de alto nivel, formalizarlo, representarlo según un esquema computacionalmente eficaz, y utilizarlo para resolver problemas en dominios de aplicación concretos.
- Niveles epistemológicos de la IA:
  - Programas de IA.
  - Sistemas Basados en Conocimiento.
  - Sistemas Expertos.

### Programas de Inteligencia Artificial:

Son programas de ordenador que exhiben cierto comportamiento inteligente, fruto de la aplicación de heurísticas.

- El conocimiento heurístico es un tipo de conocimiento difícilmente formalizable, que se establece implícitamente para tratar de encontrar respuestas más o menos correctas -pero siempre válidas- a un problema concreto.
- La utilización de conocimiento heurístico no garantiza encontrar soluciones óptimas, pero sí permite el hallazgo de soluciones aceptables -si existen-.

### Sistemas Basados en Conocimiento:

Programas de IA en los que los conocimientos del dominio y las estructuras de control del conocimiento se encuentran físicamente separados.

- Arquitecturas específicas:
  - Las estructuras “control” y “conocimientos” pueden ser desarrolladas de manera independiente.
  - Una misma “base de conocimientos” puede ser manipulada por muchas estructuras de control diferentes.
  - Una misma estructura de control puede manipular a muchas “bases de conocimientos” diferentes.

### Sistemas Expertos

- Sistemas basados en conocimiento que utilizan conocimiento particular de un dominio de aplicación concreto para resolver problemas del mundo real, limitados en tamaño, pero de gran complejidad.

- Normalmente, los sistemas basados en conocimiento explicitan el conocimiento (diferencia con los sistemas inteligentes).
- Los sistemas expertos contienen conocimiento de expertos humanos, entre otras fuentes.
- Tipos de conocimiento:
  - Conocimiento Público.
    - Puede obtenerse directamente a partir de fuentes típicas, libros y manuales. Es comúnmente aceptado y reconocido.
  - Conocimiento Semipúblico.
    - Es explícito, pero no universalmente reconocido, ni comúnmente aceptado. Es el conocimiento de grupos de especialistas.
  - Conocimiento Privado.
    - No es explícito, no está universalmente reconocido, ni es comúnmente aceptado. Es de marcado carácter heurístico.

### Agentes:

- Russell y Norvig.
  - El objetivo es explicar y construir agentes que reciben percepciones del ambiente, y proceden a ejecutar acciones.
  - Cada uno de tales agentes se implanta mediante una función que correlaciona percepciones y acciones.
  - Se pueden definir diversos procedimientos útiles para representar tales funciones:
    - Sistemas de producción.
    - Agentes reactivos.
    - Planificadores lógicos.
    - Redes semánticas.
    - Sistemas lógicos de decisión.

### Estructura de los agentes. Función:

- La función agente proyecta una percepción en una acción:

$$f: \mathcal{P}^* \rightarrow \mathcal{A}$$

- Se puede presentar en forma de tabla (percepción-acción).

| <i>secuencias de percepción</i> | <i>acción a llevar a cabo</i> |
|---------------------------------|-------------------------------|
| [A, limpio]                     | Derecha                       |
| [A, sucio]                      | Aspirar                       |
| [B, limpio]                     | Izquierda                     |
| [B, sucio]                      | Aspirar                       |
| [A, limpio], [A, limpio]        | Derecha                       |
| [A, limpio], [A, sucio]         | Aspirar                       |
| ...                             | ...                           |

## Estructura de los agentes: Programa y arquitectura:

- La IA es el programa del agente:
  - Software que determina el comportamiento del agente e implementa la función del agente.
- El programa del agente se ejecuta en computadores con sensores y actuadores (hardware del agente).

$$\text{Agente} = \text{Arquitectura} + \text{Programa}$$

- La arquitectura:
  - Engloba los módulos que componen el agente.
  - Estructura el programa del agente.
  - Partes imprescindibles:
    - Componentes de percepción, facilita las percepciones de los sensores.
    - Componente de selección de acciones.
    - Componente de acción, activa los actuadores.

## Tipos básicos de agentes:

Programas para agentes se diferencian por los métodos que emplean para seleccionar las acciones.

| TIPO                 | CARACTERÍSTICAS  |
|----------------------|--|
| Reactivo simple.     | <ul style="list-style-type: none"><li>• Es el tipo de agente más sencillo.</li><li>• Selecciona la acción en base a percepciones actuales.</li><li>• Inteligencia limitada.</li></ul>  |
| Basado en modelos.   | <ul style="list-style-type: none"><li>• Mantiene un estado interno que almacena:<ol style="list-style-type: none"><li>1. Cómo evoluciona el mundo.</li><li>2. Cómo afectan al mundo las acciones ejecutadas.</li></ol></li></ul> |
| Basado en objetivos. | <ul style="list-style-type: none"><li>• Conoce una serie de objetivos a alcanzar.</li><li>• Planifica la secuencia de acciones para lograr sus metas.</li></ul>  |
| Basado en utilidad.  | <ul style="list-style-type: none"><li>• Cada estado tiene asociado una utilidad.</li><li>• Utilidad: valor numérico, representa la bondad del estado.</li><li>• El agente debe alcanzar los estados de mayor utilidad.</li></ul> |

### Agentes solucionadores de problemas:

- Los agentes inteligentes deben maximizar una medida de rendimiento (satisfacer una meta).
- Problema de decisión complejo:
  - Un agente en la ciudad de Arad (Rumanía) en viaje de vacaciones.
  - La medida de rendimiento puede considerar algunos factores: tomar el sol, mejorar el rumano, visitar monumentos...
- Problema de decisión simplificado:
  - El agente con un vuelo no reembolsable desde Bucarest al día siguiente rechaza cualquier acción que no sea llegar a Bucarest, la meta actual.
- Las metas ayudan a organizar el comportamiento limitando los objetivos.

### Formulación de metas:

- Paso 1: **Formulación de metas.**
  - Basada en la situación actual y en la medida de rendimiento del agente.
  - Meta: Conjunto de estado del mundo en los que se satisface el objetivo.
  - Tarea del agente: Encontrar la secuencia de acciones que permita alcanzar algún estado meta.

### Formulación de problemas:

- Paso 2: **Formulación del problema.**
  - A partir de una meta dada, es el proceso de decidir qué acciones y estados considerar para alcanzarla.
  - ¿Cómo es el entorno?
    - Desconocido: No es posible elegir porque no conoce su resultado (estado).
    - Conocido: ¿Cómo decidir entre varias opciones inmediatas de valor desconocido?
      - Examinar todas las secuencias posibles de acciones que nos llevan a estados de valor conocido.
      - Seleccionar la mejor entre todas las secuencias de posibles acciones que le lleven eventualmente a estados de valor conocido.

### Proceso de búsqueda y ejecución:

- La solución a cualquier problema es una secuencia fija de acciones (bajo las condiciones anteriores).
- El proceso de buscar esa secuencia se llama búsqueda.
- **Paso 3: Búsqueda:**
  - *Entrada:* un problema.
  - *Salida:* una secuencia de acciones (una solución).
- **Paso 4: Ejecución.**
  - Una vez se encuentra la solución, se llevan a cabo las acciones recomendadas.
- El diseño de agente será del tipo formular-buscar-ejecutar.

## Tipos de problemas:

- Deterministas, completamente observables -> problemas de estado único.
  - El agente sabe exactamente en qué estado estará; la solución es una secuencia.
- No-observable -> problema conformante.
  - El agente puede no tener idea de dónde está; la solución (si hay) es una secuencia.
- No deterministas y/o parcialmente observables -> problemas de contingencia, las percepciones proporcionan nueva información sobre el estado actual.
  - La solución es un plan de contingencia o una política a menudo búsqueda intercalada, ejecución.
- Espacio de estados desconocido -> problema de exploración.

## Lo que no debes olvidar:

- Un **agente** es un ente que percibe y actúa en un entorno mediante sensores y actuadores, respectivamente.
- La **función del agente** especifica la acción a realizar por un agente en base a la secuencia de percepciones recibida.
- La **medida de rendimiento** evalúa el comportamiento de agente en un entorno determinado.
- Un **agente racional** actúa siempre con el objetivo de maximizar el valor esperado dada la secuencia de percepciones hasta el momento.
- Los programas del agente implementan (algunas) funciones del agente.
- Las descripciones REAS (Rendimiento-Entorno-Actuadores-Sensores) definen entornos tarea.
- Los entornos se pueden categorizar en varias dimensiones:
  - ¿Observable, determinista, episódico, estático, discreto, agente único?
- Existen varias arquitecturas básicas de agentes:
  - Reactivo, Reactivo con estado, basado en metas y basado en utilidades.
- **Formulación de metas, formulación del problema, búsqueda** y solución son los pasos básicos de la solución de problemas.
- La formulación de problemas suele requerir **abstraer** detalles del mundo real para definir un espacio de búsqueda que pueda ser factible explorar.
- Los componentes formales de un problema de búsqueda son **estado inicial, acciones disponibles, modelo de transición, prueba de meta y función de coste** del camino. Los tres primeros definen implícitamente el **espacio de estados** del problema.
- Una **solución** es una secuencia de acciones que transforma el estado inicial en el estado meta. La **solución óptima** es la que tiene el menor coste del camino.

## TEMA 2. BÚSQUEDA.

### Resolución de problemas:

- La resolución de un problema en IA consiste en:
  1. La aplicación de un conjunto de técnicas conocidas, cada una de ellas definida como un paso simple en el espacio de estados.
  2. Un proceso de búsqueda, o estrategia general de exploración del espacio de estados.

Para abordar un problema desde la óptica de la inteligencia artificial tenemos que ser capaces de construir un modelo computacional del universo de discurso, o dominio del problema.

- El espacio de estados es una representación exclusivamente del dominio del problema, no indica cómo obtener la solución.
- Necesitamos también procesos de búsqueda o mecanismos generales de exploración del espacio de estados.
  - Criterios de selección y aplicación de operadores relevantes.
  - Capacidad de decisión sobre cuál será el próximo movimiento.
  - Búsqueda sistemática.
  - Tratar de generar siempre estados nuevos, no generados previamente.
- La utilización de grafos de búsqueda reduce los esfuerzos de exploración en el espacio de estados.
- La utilización de grafos de búsqueda obliga a comprobar si cada estado ha sido generado ya en pasos anteriores.
- La utilización de grafos de búsqueda demanda menos recursos de memoria, pero suponen un coste computacional mayor.
- Los árboles de búsqueda son más eficientes desde una perspectiva computacional, pero tienen mayores problemas de memoria.
- Al proceso de selección sistemática de operadores relevantes se denomina **emparejamiento**.
- **Emparejamiento literal:**
  - Búsqueda simple de operadores del conjunto O del espacio de estados.
    - A través de las precondiciones del operador si el proceso es dirigido por los datos.
    - A través de los consecuentes del operador si el proceso es dirigido por los objetivos.
  - Problemas:
    - Si O es muy grande el emparejamiento literal es muy ineficiente.
    - No es siempre evidente qué operador es aplicable.
- **Emparejamiento con variables:**
  - De naturaleza no literal.
  - Útil cuando el problema requiere una búsqueda extensa en la que haya variables involucradas.

- **Funciones Heurísticas:**

- Funciones de carácter numérico, que nos permiten estimar el beneficio de una determinada transición en el espacio de estados.
- Se utilizan para optimizar los procesos de búsqueda.
- Tratan de guiar la exploración del espacio de estados de la forma más provechosa posible.
- Sugieren el mejor camino a priori, cuando disponemos de varias alternativas.
- El estudio de las funciones heurísticas se denomina Heurética.

### Estrategias de búsqueda: Evaluación:

- La complejidad temporal y espacial se miden en términos de:
  - $b$ : factor de ramificación (número promedio de sucesores por nodo).
  - $d$ : profundidad de la solución menos costosa (o del nodo meta óptimo).
  - $m$ : máxima profundidad del espacio de estados (o de cualquier camino).
  - Complejidad Temporal -> máx. número de nodos generados.
  - Complejidad Espacial -> máx. número de nodos almacenados.
- Efectividad:

$$\text{Coste total} = \text{coste búsqueda} + \text{coste camino}$$

### Topología del proceso de búsqueda:

- Construcción de un árbol de búsqueda superpuesto al espacio de estados.
  - Raíz (estado inicial), hojas (estados sin sucesor).
  - En cada paso seleccionar una hoja y expandir.

### Lo que no debes olvidar:

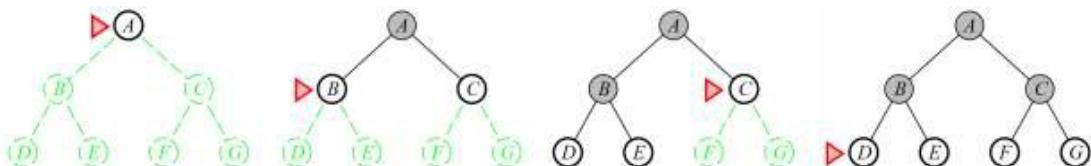
- **Formulación de metas, formulación del problema, búsqueda** y solución son los pasos básicos de la solución de problemas.
- Los componentes formales de un problema de búsqueda son **estado inicial, acciones disponibles, modelo de transición, prueba de meta y función de coste** del camino. Los tres primeros definen implícitamente el **espacio de estados** del problema.
- Una **solución** es una secuencia de acciones que transforma el estado inicial en el estado meta. La **óptima** es la que tiene el menor coste de camino.
- Los **algoritmos de búsqueda** se distinguen por el estado que seleccionan para expansión, lo cual se llama también **estrategia de búsqueda**.
- Los algoritmos de búsqueda trabajan considerando varias posibles secuencias de acciones que parten del estado inicial y forman un **árbol de búsqueda** cuya raíz es el estado inicial, las ramas son las acciones y los nodos corresponden a estados del espacio de estados del problema.
- La **búsqueda basada en grafo** evita el problema de explorar caminos redundantes. Para ellos es necesario una estructura que recuerde cada nodo previo explorado.

- Para evaluar el rendimiento de los algoritmos de búsqueda exploramos las propiedades de **completitud**, **optimización**, **complejidad en tiempo y en espacio**.

### Estrategias de búsqueda no informada:

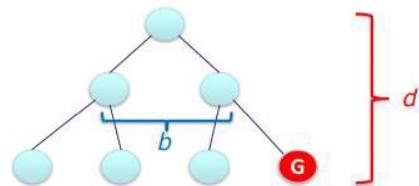
#### BÚSQUEDA PREFERENTE EN AMPLITUD (BREADTH-FIRST)

- Expandir primero el nodo raíz, luego los sucesores de nodo raíz, luego todos los sucesores de los sucesores y así sucesivamente.
- Se expanden todos los nodos de un mismo nivel antes de expandir nodos de niveles inferiores.
- Particularización del algoritmo de búsqueda general en grafo:
  - Elegir para expandir el nodo menos profundo no expandido.
  - Implementar la frontera como una cola FIFO, i.e., los nuevos nodos se insertan al final.



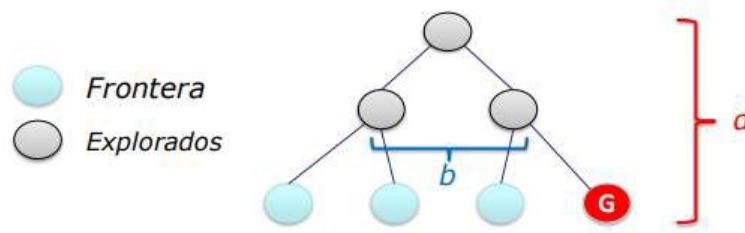
#### EVALUACIÓN DE LA BÚSQUEDA PREFERENTE EN AMPLITUD

- Complejidad temporal:
  - Si la meta se encuentra a una profundidad  $d$ , hay que generar todos los nodos de ese nivel.



$$○ b + b^2 + b^3 + \dots + b^d = O(b^d)$$

- Complejidad espacial:
  - Cada nodo generado permanece en memoria. Al nivel del nodo meta se almacena la mayor cantidad de nodos:  $O(b^{d-1})$  en el conjunto de explorados y  $O(b^d)$  en la frontera.

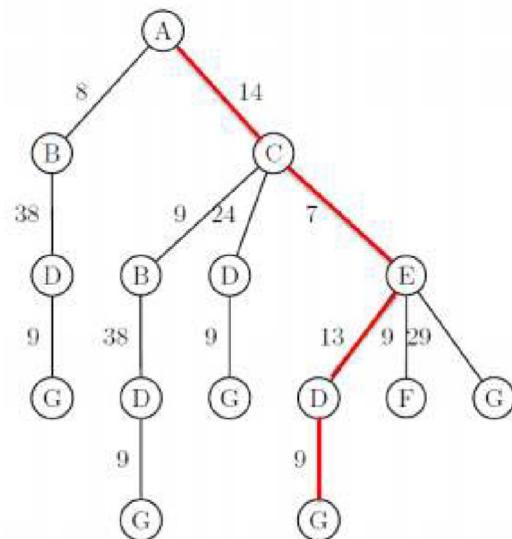


$$○ \text{ Así pues: } b + b^2 + b^3 + \dots + b^{d-1} + b^d = O(b^d)$$

## BÚSQUEDA DE COSTE UNIFORME:

- La búsqueda primero en Anchura es óptima si los costes de transición son constantes.
- Con una extensión sencilla podemos encontrar un algoritmo óptimo con cualquier función de coste por paso:
  - Expandir nodos con el menor coste de camino,  $g(n)$ .
  - Si todos los costes son iguales es idéntico a anchura.
- Implementación:
  - Frontera es una cola de prioridad ordenada por  $g$ .
  - El Test de meta se aplica al nodo cuando se selecciona para su expansión, no tras su generación.
  - Se comprueba si ya existe un camino mejor a un nodo en la frontera.
- Ejemplo:

| Paso | Frontera                   | Explorados   |
|------|----------------------------|--|
| 1    | A(0)                       | -  |
| 2    | B(8), C(14)                | A(0)   |
| 3    | C(14), D(46)               | A(0), B(8)   |
| 4    | E(21), B(23), D(38), D(46) | A(0), B(8), C(14)                                    |
| 5    | D(34), D(38), F(30), G(50) | A(0), B(8), C(14), E(21)                             |
| 6    | D(34), G(50)               | A(0), B(8), C(14), E(21), F(30)                      |
| 7    | G(50), G(43)               | A(0), B(8), C(14), E(21), F(30), D(34)               |
|      |                            | A(0), B(8), C(14), E(21), F(30), D(34), <b>G(43)</b> |

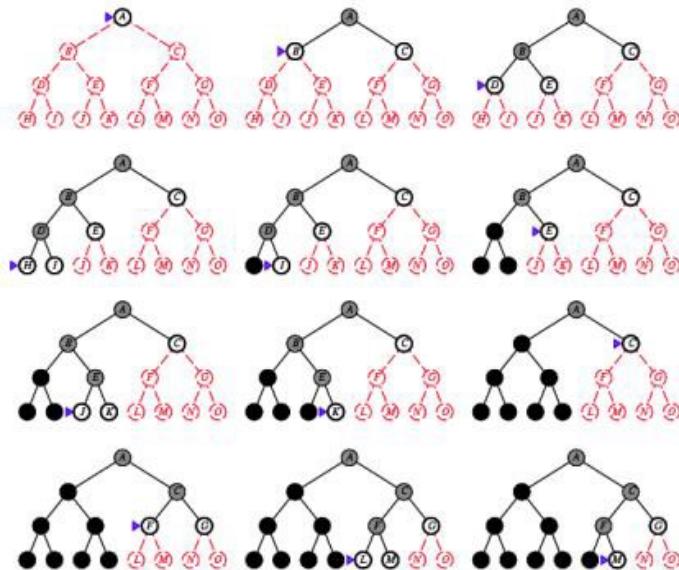


## EVALUACIÓN BÚSQUEDA DE COSTE UNIFORME

- *Completa*: Sí, si el coste de cada paso es  $\geq E$  ( $E > 0$ ).
- *Óptima*: nodos se expanden en orden creciente de coste (el primer nodo meta en ser expandido será el óptimo).
- *Complejidad*:
- *Espacio-temporal*: número de nodos con coste de camino  $\leq$  coste solución óptima  $O(b^{1+[C^*/E]})$ .
- $C^*$  coste solución óptima, y  $E$  coste de cada acción.
- Si el coste de los pasos es el mismo  $b^{1+[C^*/E]} = b^{d+1}$ .

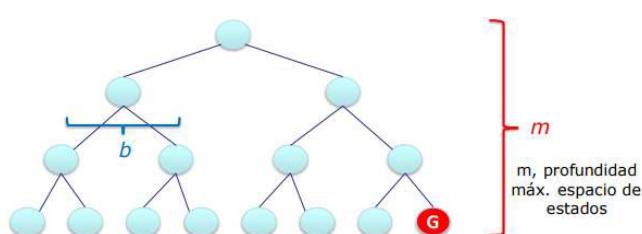
BÚSQUEDA PREFERENTEMENTE EN PROFUNDIDAD (DEPTH-FIRST)

- Expandir los nodos más profundos de la frontera actual del árbol.
  - Una vez generados, la búsqueda retrocede al siguiente nodo más profundo con sucesores sin explorar.
  - Particularización del algoritmo de búsqueda general en grafo:
    - Elegir para expandir el nodo más recientemente generado.
    - Implementar la frontera como una cola LIFO, i.e., los nuevos nodos se insertan por el principio.
  - Ejemplo:

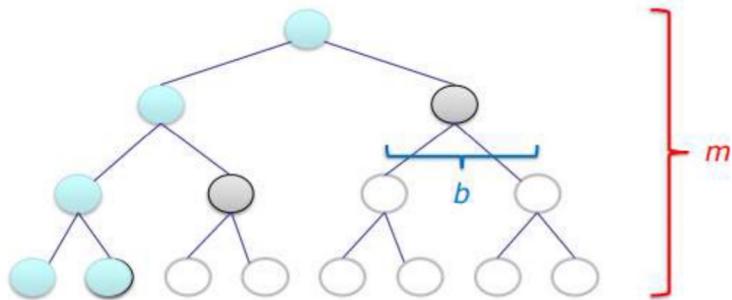


## EVALUACIÓN BÚSQUEDA PREFERENTE EN PROFUNDIDAD

- Completa:
    - Sí, si usamos búsqueda basada en grafos en espacios de estado finitos porque expandirá todos los nodos.
    - No, si usamos búsqueda basada en árboles.
    - No, en espacios de búsqueda infinitos si encuentran un camino infinito que no lleve a la meta.
  - Óptima:
    - No, puede encontrar una solución más profunda que otra en una rama no expandida.
  - Complejidad temporal:
    - Similar al tamaño del espacio de estados, tal vez infinito (búsqueda basada en grafo).
    - En el peor caso,  $b^m$  nodos:  $b^m + b^{m-1} + \dots + 1 = O(b^m)$  (búsqueda basada en árbol).

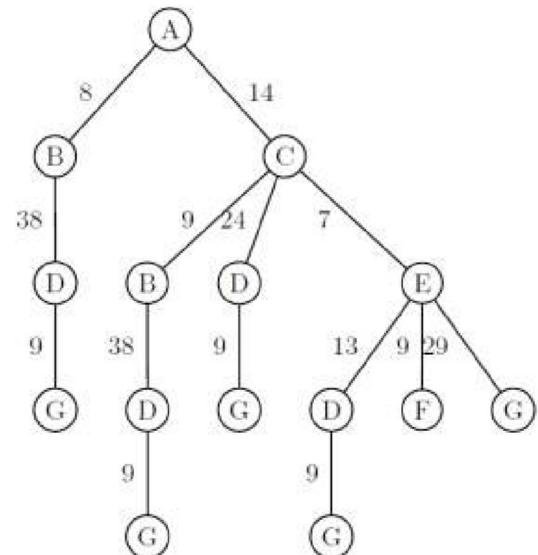


- Complejidad espacial:
  - El mayor número de nodos almacenados se alcanza en el nodo inferior de más a la izquierda.
  - Sólo almacena un camino (raíz-hoja) para búsquedas en árbol.
  - En general:  $1+b+b+\dots+b = 1+b^m = O(bm)$ .



- Ejemplo 2:

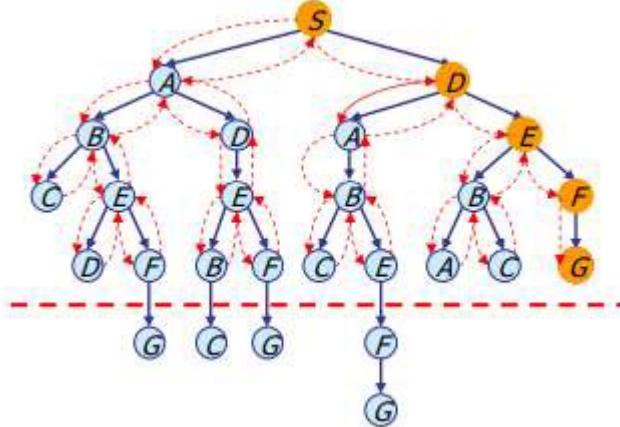
| Paso | Frontera | Explorados |
|------|----------|------------|
| 1    | A        | -          |
| 2    | B,C      | A          |
| 3    | D,C      | A,B        |
| 4    | G,C      | A,B,D      |



- Variante: Búsqueda con *backtracking*.
  - Usa menos memoria todavía.
  - Sólo se genera un sucesor de cada vez.
  - Cada nodo parcialmente expandido recuerda cuál es el siguiente sucesor a generar.
  - Complejidad espacial:  $O(m)$  en vez de  $O(b^m)$ .
  - Otro truco añadido para ahorrar memoria y tiempo:
    - Generar el sucesor modificando la descripción del estado actual.
    - Requisitos de memoria: una descripción de estado y  $O(m)$  acciones.

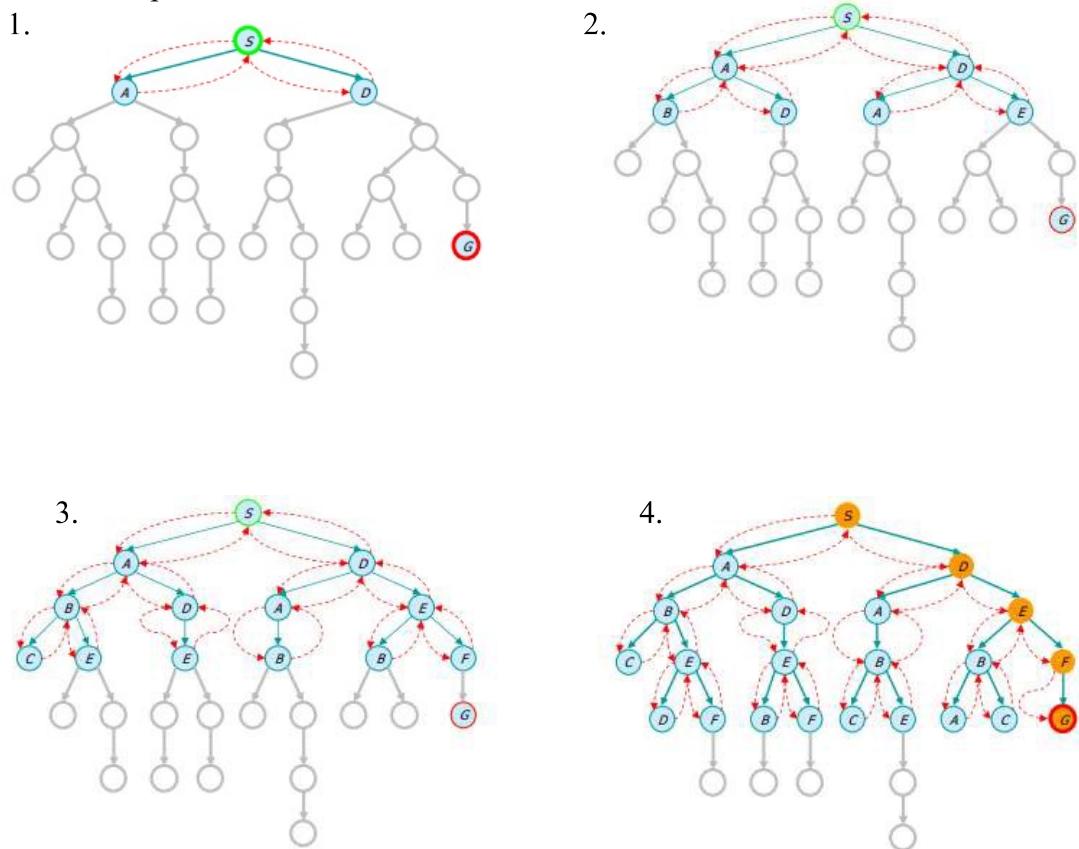
## BÚSQUEDA DE PROFUNDIDAD LIMITADA

- Fijar un límite a la profundidad máxima.
- Soluciona caminos infinitos.
- Si  $l < d$  (meta más allá del límite) incompleto.
- Si  $l > d$  no es óptimo.
- Complejidad temporal  $O(bc^l)$  y espacial  $O(b^*l)$ .



## BÚSQUEDA DE PROFUNDIDAD ITERATIVA

- Variante con profundidad iterativa.
  - Incrementar gradualmente el límite (i.e. 0, 1, 2, ...).
  - Beneficios de la búsqueda en anchura y en profundidad.
  - En espacios de búsqueda muy grandes en los que no se conoce la profundidad de la solución.



## COMPARATIVA PARA BÚSQUEDA EN ÁRBOLES

|                             | Amplitud        | Coste Uniforme                        | Profundidad | Profundidad limitada | Profundización iterativa |
|-----------------------------|-----------------|---------------------------------------|-------------|----------------------|--------------------------|
| <b>Completa</b>             | Si <sup>a</sup> | Si <sup>a,b</sup>                     | No          | Si ( $l \geq d$ )    | Si <sup>a</sup>          |
| <b>Óptima</b>               | Si (coste cte)  | Si                                    | No          | No                   | Si <sup>c</sup>          |
| <b>Complejidad Espacial</b> | $O(b^d)$        | $O(b^{1+\lceil C^*/\epsilon \rceil})$ | $O(bm)$     | $O(bl)$              | $O(bd)$                  |
| <b>Complejidad Temporal</b> | $O(b^d)$        | $O(b^{1+\lceil C^*/\epsilon \rceil})$ | $O(b^m)$    | $O(b^l)$             | $O(b^d)$                 |

b: factor de ramificación.

d: profundidad de la solución menos profunda.

m: máxima profundidad del árbol de búsqueda.

l: límite de profundidad

<sup>a</sup> completa si b es finito

<sup>b</sup> completa si coste de paso  $0 \geq \epsilon, \epsilon > 0$

<sup>c</sup> óptima si los costes de transición son iguales

## Estrategias de búsqueda informada (búsqueda heurística)

### 1. Búsqueda preferente por el mejor:

- Basados en la búsqueda en anchura.
- Examinar primero nodos que, de acuerdo con la heurística, están situados en el mejor camino hacia el objetivo.

### 2. Búsqueda local:

- Tratar de mejorar el estado actual en lugar de explorar de manera sistemática los caminos desde el estado inicial.
- El coste del camino es irrelevante y lo único importante es alcanzar el estado meta.

En ambos casos se posee conocimiento acerca de si un estado no meta es más prometedor que otro estado dado.

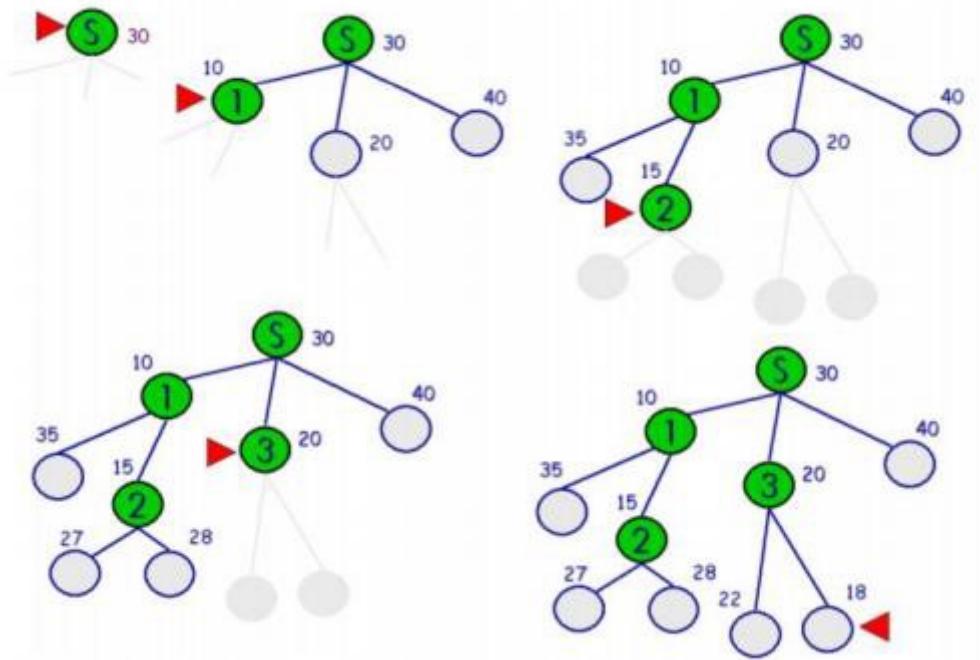
## BÚSQUEDA PREFERENTE POR EL MEJOR

- Basada en el algoritmo de búsqueda en árbol o grafo.
- El nodo a expandir se selecciona en base a la **función de evaluación, f(n)**: primero el menor (mejor) valor.
- La mayoría de estos algoritmos incluyen como componente de f una **función heurística h(n)**.
  - Coste estimado del camino menos costoso desde el estado en el nodo n hasta el estado meta.
- Se selecciona un nodo para la expansión en base a una función de evaluación heurística, f(n) mide la distancia al objetivo.
- Decide cuál es el mejor nodo a expandir.
- f(n) tomará valores pequeños en los nodos más prometedores.
- Se expande el nodo n con la evaluación más baja de f.
- Se termina el proceso cuando el nodo a expandir sea un nodo objetivo.

- Se implementa con una cola de prioridad que mantiene en orden ascendente la lista de nodos abiertos según el valor de  $f$ .
- Si  $f(n)$  sólo considera el coste mínimo estimado para llegar a una solución a partir de un nodo  $n$ ,  $h(n)$  o función heurística:
  - $f(n) = h(n)$  -> **Búsqueda avara\***.
- Si  $f(n)$  considera el coste mínimo total del camino a un nodo solución que pase por el nodo  $n$ :
  - $f(n) = g(n) + h(n)$  -> **Búsqueda A\*** (**A asterisco o estrella**).
  - $g(n)$  = coste consumido para llegar a  $n$ .
- $h(n) = 0$  si  $n$  es un nodo meta.

## BÚSQUEDA AVARA

- Ejemplo:



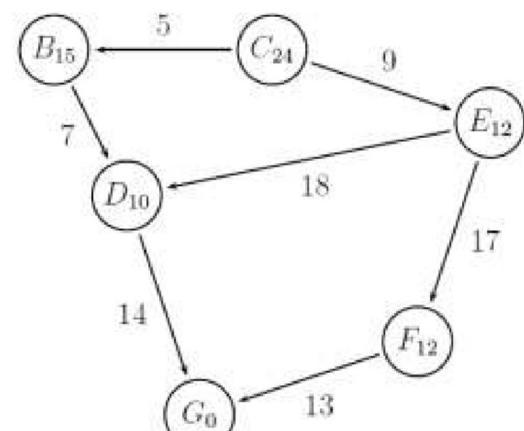
- Evaluación de la estrategia de búsqueda:

| Búsqueda avara |  |
|----------------|--|
| Óptima         | No   |
| Completa       | No, puede perderse en bucles y recorrer rutas infinitas  |
| Complejidad    | <p><i>Temporal</i><br/> <math>O(b^m)</math> siendo <math>m</math> la profundidad máxima del árbol de búsqueda</p> <p><i>Espacial</i><br/> <math>O(b^m)</math> mantienen todos los nodos en memoria</p> |

## BÚSQUEDA A\*

- Búsqueda avara:
  - Minimizar el coste de alcanzar la meta,  $h(n)$ .
  - No es óptima ni completa.
  - Eficiente.
- Búsqueda de coste uniforme:
  - Minimizar el coste del camino,  $g(n)$ .
  - Es óptima y completa.
  - A veces muy ineficiente.
- ¿Por qué no combinamos ambos métodos, en uno óptimo, completo y más eficiente?
- A\* o A-estrella es la forma más conocida de búsqueda preferente por el mejor.
- Evalúa los nodos combinando  $g(n)$  y  $h(n)$ :
  - $g(n)$ , coste real del mejor camino para alcanzar el nodo  $n$ .
  - $h(n)$ , coste estimado del camino menos costoso desde el nodo  $n$  a la meta.
  - $f(n) = g(n) + h(n)$  -> coste estimado de la solución menos costosa que pasa a través del nodo  $n$ .
- A\* puede utilizarse para encontrar:
  - Un camino de coste total mínimo
  - El camino más rápido posible  $g=1$  (menor nº pasos).
- A\* se comporta como:
  - Anchura si:
    - $g$  se incrementa en 1 y  $h=0$ .
    - Los nodos con igual  $f$  se ordenan de menos a más reciente.
  - Profundidad si:
    - $g = 0$  y  $h = 0$ .
    - Los nodos se ordenan de más a menos reciente.
- Ejemplo:

| Paso | Frontera          | Explorados                 |
|------|-------------------|----------------------------|
| 1    | C(0+24)           | -                          |
| 2    | B(5+15),E(9+12)   | C(0)                       |
| 3    | E(9+12),D(12+10)  | C(0),B(5)                  |
| 4    | D(12+10),F(26+12) | C(0),B(5),E(9)             |
| 5    | F(26+12),G(26+0)  | C(0),B(5),E(9),D(12)       |
| 6    | F(26+12)          | C(0),B(5),E(9),D(12),G(26) |



## Búsqueda A\* óptima: Admisibilidad y consistencia:

- La búsqueda A\* basada en árbol es óptima si la heurística es **admisible** (optimista), es decir, si nunca sobreestima el coste real de alcanzar la meta.
- La búsqueda A\* basada en grafo es óptima si la heurística es **consistente** (optimista).
- Si  $g(n)$  es el coste real de alcanzar  $n$  por el camino actual, y  $f(n) = g(n)+h(n)$ , entonces  $f(n)$  nunca sobreestimará el coste real de una solución por el camino actual que pase por  $n$ .
- Cuando existen distintos caminos que llevan a un mismo estado, la búsqueda A\* basada en grafo selecciona siempre el primer camino generado (puede no ser el óptimo). Soluciones:
  - Descartar el camino más costoso de los dos que llegan al mismo nodo.
  - Asegurar que el camino óptimo a cualquier estado repetido es siempre el primero que se encuentra -> CONSISTENCIA de una heurística.
- La búsqueda A\* basada en grafo es óptima si la heurística es **consistente**:
  - Si para cada nodo  $n$  y cada sucesor  $n'$  de  $n$  generado por cualquier acción  $a$ , el coste estimado de alcanzar la meta desde  $n(h(n))$  no es mayor que la suma entre el coste de llegar a  $n'$  y el coste estimado de alcanzar la meta desde  $n'$ .
- Si  $h$  es consistente, entonces los valores de  $f$  a lo largo de cualquier camino son no decrecientes -> Monotonía.
- El primer nodo meta seleccionado para expansión deberá ser una solución óptima, ya que todos los nodos posteriores serán, al menos, tan costosos.

## Evaluación búsqueda A\*

- Óptima:
  - Ningún otro algoritmo óptimo garantiza expandir menos nodos.
  - A es óptimamente eficiente para una heurística consistente dada.
- Completa:
  - Si todos los pasos exceden un  $\epsilon$  finito y  $b$  finito.
- Complejidad temporal: Exponencial.
- Complejidad espacial:
  - Mantiene todos los nodos generados en memoria.
  - No es práctico para problemas muy grandes.

## Diseño de heurísticas admisibles:

- Heurísticas basadas en la abstracción del problema:
  - La solución a un problema se construye aplicando una serie de operadores elementales sujetos a una serie de restricciones.
  - Conseguir una abstracción de un problema transformándolo en otro más sencillo eliminando ciertas restricciones del problema original (“relajándolo”).

- La función de coste de la ruta en el problema relajado será una *heurística admisible* y *consistente* en el problema original.
- Cuantas más restricciones tengamos en cuenta, más precisa será la heurística.

### Búsqueda local:

- Búsqueda no informada y búsqueda por el mejor:
  - Exploran sistemáticamente el espacio de búsqueda conservando uno o más caminos en memoria.
  - Registran qué alternativas han sido exploradas en cada punto de ese camino y cuáles no.
  - Cuando encuentran una meta, el camino a la meta constituye una solución al problema.
- En muchos problemas, el camino es irrelevante.
- Usan un único estado actual (en lugar de múltiples caminos) y se mueven sólo a sus vecinos (el camino a la meta es irrelevante).
- Los caminos no se retienen.
- No son sistemáticos, pero tienen dos ventajas:
  - Poca memoria, por lo general una cantidad constante.
  - Encuentran soluciones razonables en espacios de estados grandes o infinitos (continuos).
  - Aptos para problemas de optimización.

### Algoritmo de escalada (“Hill-climbing”)

- Cada punto representa un estado y un valor de la función objetivo.
- Se trata de encontrar el punto más alto, máximo global.
- Inconvenientes:
  - **Máximo local:**
    - Estado puntual mejor que cualquiera de sus vecinos, pero peor que otros estados más alejados (máximo global). Cuando aparecen cerca de la solución final se llaman estribaciones.
  - **Meseta:**
    - Región del espacio de estados en la que todos los estados individuales tienen el mismo valor de la función heurística y, por lo tanto, no es posible determinar la mejor dirección para continuar.
  - **Cresta:**
    - Región del espacio de estados con estados que tienen mejores valores de la función heurística que los de regiones colindantes, pero a los que no podemos llegar mediante transiciones simples.
- Soluciones:
  - **Para los Máximos Locales:**
    - Regresar a un nodo previo e intentar una dirección diferente (backtracking).
  - **Para las Mesetas:**

- Realizar un gran salto en el espacio de búsqueda y tratar de alcanzar una región diferente del espacio de estados.
- **Para las Crestas:**
  - Aplicar más de un operador antes de realizar la prueba de meta (equivale a moverse en varias direcciones).

## TEMA 3. REPRESENTACIÓN DEL CONOCIMIENTO: DECLARATIVO Y PROCEDIMENTAL

### Métodos Estructurados:

- Los esquemas no formales de representación del conocimiento verifican las siguientes propiedades:
  - Adecuación representacional: el esquema elegido debe ser capaz de representar las distintas clases de conocimiento del dominio.
  - Adecuación inferencial: el esquema elegido debe permitir manipular conocimiento para obtener conocimiento nuevo.
  - Eficiencia inferencial: el esquema elegido debe ser versátil, utilizando información que permita optimizar el proceso inferencial.
  - Eficacia adquisicional: el esquema elegido debe suministrar vías que permitan la incorporación de información y conocimientos nuevos.

### Representación del conocimiento:

- **Métodos declarativos:**
  - El conocimiento se representa como una colección estática de hechos, para cuya manipulación se define un conjunto genérico y restringido de procedimientos.
- **Métodos procedimentales:**
  - La mayor parte del conocimiento se representa como procedimientos, lo cual le confiere al esquema de representación un carácter dinámico.

### Métodos Estructurados:

- **Ventajas de los métodos declarativos:**
  - Las verdades del dominio se almacenan una sola vez.
  - Es fácil incrementar e incorporar nuevo conocimiento sin modificar ni alterar el ya existente.
- **Ventajas de los métodos procedimentales:**
  - Mayor énfasis en las capacidades inferenciales del sistema.
  - Permiten explorar distintos modelos y técnicas de razonamiento.
  - Permiten trabajar con información de carácter probabilístico.
  - Incorporan de forma natural conocimiento heurístico.

- **Métodos declarativos:**
  - Redes semánticas:
    - Permiten describir simultáneamente acontecimientos y objetos.
  - Frames o Marcos:
    - Estructuras genéricas que permiten representar objetos complejos desde diferentes puntos de vista.
- **Métodos procedimentales:**
  - Reglas de producción:
    - Constituidas por una premisa -IF-, una conclusión -THEN- y, opcionalmente, una alternativa -ELSE-.

### Redes semánticas:

- El conocimiento se representa como un conjunto de nodos conectados entre sí por medio de arcos etiquetados.
- Los arcos representan relaciones lingüísticas entre nodos.
- Los enlaces son unidireccionales. Para establecer enlaces bidireccionales hay que tratar cada relación por separado.
- Un enlace es una relación binaria entre nodos.
- Relaciones posibles:
  1. Ocurrencia:
    - Cuando se relaciona un miembro de una categoría general con la categoría a la que pertenece ( $\in$ ).
  2. Generalización:
    - Relaciona una entidad con otra de carácter más general (ES\_UN).
  3. Agregación:
    - Se relacionan componentes de un objeto con el objeto propiamente dicho (ES\_PARTE\_DE).
  4. Acción:
    - Se establecen vínculos dinámicos entre objetos.
  5. Propiedades:
    - Relaciones entre objetos y características de los objetos.
- Elementos que permiten establecer relaciones entre distintas estructuras de conocimiento.
  1. ES\_UN (IS\_A): permite establecer relaciones entre taxonomías jerárquicas.
  2. ES\_PARTE\_DE (PART\_OF): permite establecer relaciones entre objetos y componentes de un objeto.

### Métodos estructurados:

- Herencia de propiedades:
  - Cualquier propiedad considerada cierta para una clase de elementos debe ser cierta para cualquier ejemplo de la clase (taxonomías).

- Razonamiento:
  - Por rastreo. Las relaciones pueden no ser totalmente rigurosas y, por lo tanto, las inferencias obtenidas por rastreo pueden no ser válidas (condiciones de excepción no reconocidas).
  - Emparejamiento. Construcción de fragmentos de red, algunos de cuyos nodos tienen valores definidos, pero otros no (variables). El sistema debe tratar de encontrar un fragmento de la red original que encaje con la red problema.

### Frames:

- Tratan el problema de la representación desde la óptica del razonamiento por semejanza.
- Describen clases de objetos.
- Son representaciones estructuradas de conocimiento estereotipado.
- **Estructura:**
  - Cabecera:
    - Le da nombre a la frame, y es representativa de la clase de objetos que se describen.
  - Slots:
    - Elementos que representan una propiedad o atributo del elemento genérico representado por la frame.
    - Los slots pueden anidarse sin limitación de profundidad.
    - La profundidad de un slot representa un nivel de conocimiento, y si contenido es una especialización del nivel anterior.

### Frames. Demons:

- Información procedimental (demons):
  - Procedimientos que la mayor parte del tiempo están inactivos, pero que cuando son activados desencadenan acciones concretas.
  - If\_needed, If\_added, If\_removed, ... (D\_algo).
  - Cuando un demon es activado por una entrada en la frame al nivel correspondiente, el procedimiento del demon se ejecuta, y luego el demon es eliminado.
- Los demons...
  - Proporcionan uniones procedimentales entre distintas frames.
  - Posibilitan la ejecución de rutinas externas.
  - Imprimen un cierto carácter dinámico a la representación del conocimiento con frames.
- Ventajas de los frames:
  - Permiten definir procesos de razonamiento con información incompleta.
  - Permiten inferir rápidamente hechos no representados de forma explícita.
  - Imprimen cierto carácter dinámico a la representación al definir procesos que establecen relaciones entre otras frames, y conexiones con el mundo exterior.

- Utilizan con profusión la herencia.
- Razonamiento con frames:
  1. Selección de la frame que mejor se ajuste a nuestra situación actual.
  2. Ejemplificación de dicha frame tras considerar las condiciones específicas actuales (asociar un individuo particular a una clase).

## Representación del conocimiento:

- **Reglas de producción:**
  - Son un esquema de representación del conocimiento procedimental.
  - Constan de tres partes:
    - IF Condición o premisa.
    - THEN Conclusión o acción.
    - ELSE Alternativa.
  - La premisa puede estar constituida por un conjunto de cláusulas anidadas ( $\wedge, \vee, \neg$ ).

## Tipos de reglas:

- **IFALL**
  - Todas las cláusulas de la premisa han de ser ciertas para que se ejecute la acción, o se establezca la conclusión de la parte THEN.
  - Equivale a una regla en la que todas las cláusulas estén anidadas por medio de operadores ( $\wedge$ ).
- **IFANY**
  - Todas las cláusulas de la premisa están conectadas por medio de operadores ( $\vee$ ).
  - En cuanto una cláusula es cierta se ejecuta la acción o se establece la hipótesis de la parte THEN.
  - Equivale a una búsqueda no exhaustiva de la misma regla.
- **IFSOME**
  - Todas las cláusulas de la premisa están conectadas por medio de operadores ( $\vee$ ).
  - Aunque una cláusula sea cierta, antes de ejecutar la acción o de establecer la hipótesis de la parte THEN, se investiga toda la premisa.
  - Equivale a una búsqueda exhaustiva dentro de la misma regla.

## Reglas de producción: VENTAJAS:

- Las condiciones y acciones involucradas son explícitas.
- El conocimiento es representado de forma muy modular.
- Cada regla constituye una unidad completa de razonamiento.
- Permiten almacenar y utilizar conocimiento muy específico, y de naturaleza heurística.

## TEMA 4.

### SISTEMAS DE PRODUCCIÓN

#### Introducción:

- Programas secuenciales:
  - Dependientes de los datos, condiciones iniciales, parámetros, respuestas de los usuarios, resultados de cómputos anteriores, ...
  - El flujo de control y la utilización de los datos especificados de manera rígida por el programa.
  - El defecto específico de estos programas es su secuencialidad:
    - La bifurcación sólo se efectúa en puntos y caminos explícitamente previstos en el código del programa.
  - En entornos cambiantes, la bifurcación será la norma, no la excepción, ya sea a partir de un nuevo estímulo, en función de sus propiedades y contenidos, o bien, en ausencia de nuevos estímulos, será a partir del contexto actual, historia o último estado.
- Programación basada en eventos:
  - Adecuada para situaciones en las que continuamente están surgiendo nuevos estímulos, bien originados por el entorno externo o bien generados internamente por el programa.
  - El programa responde directamente a un amplio rango de sucesos, algunos imprevistos, en lugar de hacerlo a datos esperados.
  - No usan estructuras de control inflexibles y especificadas de antemano.
  - Reconocen patrones en los datos, y seleccionan trozos de código en el sistema para que se activen.
  - Sistemas de inferencia dirigidos por patrones (SIDP).
  - Los SIDP tienen estructura modular.
  - Cada módulo es responsable de detectar las distintas situaciones posibles, reconociendo patrones en los datos, y de responder de forma adecuada a las mismas.
  - Los módulos se dividen funcionalmente en **antecedente** o lado izquierdo y **consecuente** o lado derecho.
  - El *antecedente* efectúa todos los accesos a los datos, para verificarlos y equipararlos con los patrones “plantilla” del módulo.
  - El *consecuente* efectúa la escritura o modificación de datos.
  - Tales módulos se denominan “reglas”, y los SIDP compuestos por conjuntos de reglas, Sistemas basados en Reglas (SBR) o Sistemas de Producción.

#### Sistemas de Producción:

- Definición:
  - Sistemas inteligentes basados en reglas que operan frente a una base de hechos con mecanismos de emparejamiento formando parte explícita de su arquitectura.

- Se clasifican en dos categorías según su estructura de control:
  - Sistemas dirigidos por los datos.
  - Sistemas dirigidos por los objetivos.
- **Sistemas dirigidos por los datos:**
  - Las inferencias se obtienen cuando los antecedentes de alguna (o más de una) de sus reglas de producción se emparejan con, al menos, una parte de los hechos que describen el estado actual.
  - Cuando esto ocurre, se dice que la regla en cuestión se ha activado, y está en condiciones de ser ejecutada. Su ejecución o no dependerá de la estrategia de exploración elegida.
  - Son menos específicos, porque ejecutarán todas las reglas disponibles en función de la información introducida.
- **Sistemas dirigidos por los objetivos:**
  - Tanto antecedentes como consecuentes de las reglas deben ser considerados como aserciones sobre los datos. En este caso, la activación de las reglas tiene lugar por medio de un encadenamiento regresivo, y el emparejamiento se efectúa a través de las conclusiones de las reglas.
  - Para alcanzar una determinada meta hay que configurar un proceso evocativo en el que, de forma recursiva, se van estableciendo los antecedentes de las metas como submetas de orden inferior.
  - Son más específicos, porque la ejecución lleva implícito un proceso de búsqueda.

### Base de Conocimientos:

- Características:
  - Describe el universo de discurso o dominio en el cual el sistema de producción tiene que plantear soluciones.
  - Está constituida por bases de hechos (BH) y por bases de reglas (BR).
  - Las **bases de hechos** forman el esqueleto declarativo del sistema de producción, y su misión es la de articular a todos los hechos potencialmente relevantes del dominio.
  - Las **bases de reglas** constituyen el esqueleto procedural del sistema de producción, y a través de ellas se posibilita la construcción de los circuitos inferenciales que nos van a permitir obtener conclusiones válidas.
  - La estructura de las bases de hechos y de las bases de reglas debe ser tal que ambas entidades puedan “comprenderse” entre sí.

### Memoria Activa (MA):

- Es la estructura que contiene toda la información de naturaleza estática necesaria para resolver un problema concreto. Esta información incluye:
  - Datos iniciales del problema.
  - Datos incorporados con posterioridad.
  - Hechos establecidos durante los procesos inferenciales.
  - Hipótesis de trabajo, metas o submetas que todavía no han sido establecidas.

- Almacena todos los cambios de estado de nuestro sistema, de forma que representa siempre nuestro estado actual.
- Es la responsable de interaccionar con el mundo exterior, aceptando la entrada de información de naturaleza no inferencial.
- Es el foco permanente de atención de las reglas del sistema.

### Motor de Inferencias (MI):

- Intérprete de reglas + Estrategia de control. Separación con el conocimiento.
- Funciones:
  - Examinar la memoria activa y determinar qué reglas deben ejecutarse (estrategia de búsqueda + resolución de conflictos). Encontrar conexiones entre estados iniciales del problema y estados solución.
    - Desde las premisas a las conclusiones.
    - Desde las soluciones a los datos iniciales.
    - Desde ambos simultáneamente.
  - Controlar y organizar el proceso de ejecución de las reglas seleccionadas en el paso anterior.
  - Actualizar la memoria activa cuando sea preciso (hechos, metas y submetas).
  - Asegurar el autoconocimiento del sistema (reglas activadas, reglas ejecutadas, últimos hechos incorporados a la memoria, prioridades de reglas, ...).
- El intérprete no es más que un programa secuencial cuya misión es determinar el siguiente paso a ejecutar.
- La estrategia de control es el mecanismo que examina la memoria activa y determina qué regla disparar, a través de los llamados ciclos básicos del sistema, y en función de ciertos parámetros como:
  - Criterios de activación elegidos.
  - Estrategias de búsqueda implementadas.
  - Dirección de tránsito por el espacio de estados.

### Ciclo básico de Sistemas de Producción:

- El ciclo básico está constituido por:
  - Fase de **decisión** o **selección de reglas**.
    - Restricción.
    - Equiparación.
    - Resolución de conflictos.
  - Fase de **acción** o **ejecución** de las reglas seleccionadas.

### Fase de decisión: Tareas:

- **Restricción:**
  - Trata de simplificar el proceso de equiparación.

- Elimina del foco de atención del motor de inferencias aquellas reglas que claramente no tienen nada que ver con el estado actual representado en la memoria activa del sistema.
- Ejemplo de restricción estática: los sistemas de producción suelen dividirse en varias bases de reglas y varias bases de hechos.
- Alternativa dinámica: emplea metaconocimiento (conocimiento sobre conocimiento).
- **Equiparación** (emparejamiento):
  - Se tratará de identificar qué reglas son potencialmente relevantes en el contexto del problema que queremos resolver.
  - El resultado final es la obtención del denominado *conjunto conflicto* que, incluye todas las reglas potencialmente útiles en la resolución de nuestro problema.
- **Resolución de conflictos:** decidir qué regla aplicar:
  - La decisión final está fuertemente condicionada por la estrategia genérica de búsqueda.
  - Técnicas:
    - Uso de metarreglas. Por ejemplo: las reglas proporcionadas por expertos tienen mayor prioridad que las de los novatos.
    - Asignar prioridades a las reglas (en función del orden en la base de las reglas o de valores numéricos directamente).
    - Usar la regla más específica puesto que procesa más información que una regla general.
    - Elegir la regla que use los datos más recientes, con el fin de seguir una línea de razonamiento estable y sensible con la nueva información.

#### Fase de acción: Tareas:

- Actualización de la memoria activa (nuevos hechos y/o hipótesis).
- Marcaje de las estructuras utilizadas.
- Verificación de si continuar o no el proceso cíclico.

## TEMA 5

### MODELOS DE RAZONAMIENTO

- **Modelos categóricos:**
  - Dominios de naturaleza marcadamente simbólica, con soluciones que pueden establecerse con total seguridad.
- **Modelos probabilísticos:**
  - Dominios de naturaleza estadística, soluciones no obtenibles de forma unívoca.
- Modelos de razonamiento bajo incertidumbre:
  - Dominios con incertidumbre, inherente a datos o a los propios mecanismos inferenciales.

- Modelos de razonamiento basado en conjuntos difusos:
  - Dominios en los que los elementos diferenciales incluyen matices lingüísticos.

### Modelo Categórico:

- Formalmente:
  - $X = \{\text{manifestaciones}\} = \{x_1, x_2, \dots, x_n\}$
  - $Y = \{\text{interpretaciones}\} = \{y_1, y_2, \dots, y_m\}$
- Las relaciones causales entre manifestaciones e interpretaciones se formalizan a través de la función de conocimiento E.
  - $E = E(X, Y)$
- Problema lógico.
  - Dadas unas manifestaciones caracterizadas por una función -f-, encontrar la función -g- que satisface
    - $E: (f \rightarrow g)$
    - $E: (\neg g \rightarrow \neg f)$
    - $E = E(x_1, \dots, x_n, y_1, \dots, y_m)$
- Procedimiento sistemático para el modelo categórico:
  1. Identificación de M.
  2. Identificación de I.
  3. Construcción de E.
  4. Construcción del conjunto completo de complejos de manifestaciones.
  5. Construcción del conjunto completo de complejos de interpretaciones.
  6. Construcción del conjunto completo de complejos manifestación-interpretación.

|             |           |           |           |           |
|-------------|-----------|-----------|-----------|-----------|
| <b>m(1)</b> | 0         | 0         | 1         | 1         |
| <b>m(2)</b> | 0         | 1         | 0         | 1         |
|             | <b>m1</b> | <b>m2</b> | <b>m3</b> | <b>m4</b> |
| <b>i(1)</b> | 0         | 0         | 1         | 1         |
| <b>i(2)</b> | 0         | 1         | 0         | 1         |
|             | <b>i1</b> | <b>i2</b> | <b>i3</b> | <b>i4</b> |

$$M = \{ m_1, m_2, m_3, m_4 \}$$

$$I = \{ i_1, i_2, i_3, i_4 \}$$

- Construcción del conjunto completo de complejos manifestación-interpretación:
  - Base Lógica Expandida
    - $BLE = M \times I = \{ m1i1, m1i2, m1i3, m1i4, m2i1, m2i2, m2i3, m2i4, m3i1, m3i2, m3i3, m3i4, m4i1, m4i2, m4i3, m4i4 \}$
  - La solución a cualquier problema está en BLE, pero hay muchas combinaciones absurdas.
  - El papel del conocimiento -E- es eliminarlas y pasar a una base lógica reducida E : ( $BLE \rightarrow BLR$ ).

|                  | <b>m1</b> | <b>m2</b> | <b>m3</b> | <b>m4</b> |
|------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| <b>m<br/>(1)</b> | 0         | 0         | 1         | 1         | 0         | 0         | 1         | 1         | 0         | 0         | 1         | 1         | 0         | 0         | 1         | 1         |
| <b>m<br/>(2)</b> | 0         | 1         | 0         | 1         | 0         | 1         | 0         | 1         | 0         | 1         | 0         | 1         | 0         | 1         | 0         | 1         |
| <b>i<br/>(1)</b> | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 0         | 1         | 1         | 1         | 1         | 1         | 1         | 1         | 1         |
| <b>i<br/>(2)</b> | 0         | 0         | 0         | 0         | 1         | 1         | 1         | 1         | 0         | 0         | 0         | 0         | 1         | 1         | 1         | 1         |
|                  | <b>i1</b> | <b>i1</b> | <b>i1</b> | <b>i1</b> | <b>i2</b> | <b>i2</b> | <b>i2</b> | <b>i2</b> | <b>i3</b> | <b>i3</b> | <b>i3</b> | <b>i3</b> | <b>i4</b> | <b>i4</b> | <b>i4</b> | <b>i4</b> |

- $BLR = \{m1i1, m3i2, m2i3, m4i3, m3i4, m4i4\}$
- IF: (1) El conocimiento es completo
- And: (2) El dominio está bien descrito
- Then: (3) La solución a cualquier problema está en BLR.

Bayesiano:

- Las interpretaciones categóricas son poco frecuentes en el mundo real.
  - La existencia de una determinada causa no siempre conlleva la presencia de una manifestación.
  - Ante una manifestación dada, ¿podemos afirmar siempre y de forma categórica que existe una determinada causa?
- Probabilidad total y probabilidad condicional.
- La probabilidad condicional se parece a la total, pero puede ser definida como la probabilidad de las causas.
- En la probabilidad condicional aparecen involucrados dos sucesos, en donde la ocurrencia del segundo depende de la ocurrencia del primero.
- Bayes introduce una forma de razonamiento a posteriori.
  - Dadas dos posibles acciones A y B, y ante un caso tratado con una de tales acciones ¿Cuál es la probabilidad de que la acción haya sido A, si la respuesta del sistema ha sido E?

$$P(A/E) = \frac{P(E/A)P(A)}{P(E)}$$

- Obtención de la ecuación elemental del teorema de Bayes:
  - Sea una población sobre parte de cuyos elementos ha sido efectuada una acción A de un número de posibles acciones.
  - Todos los elementos de la población fueron tratados, con A o con cualquier otra acción, registrándose un cierto número de respuestas E.
- Por definición de probabilidad condicional:

$$P(E/A) = \frac{n(A \cap E)}{n(A)} = \frac{n(A \cap E)/N}{n(A)/N} = \frac{P(A \cap E)}{P(A)} \rightarrow P(A \cap E) = P(A)P(E/A)$$

- Análogamente:

$$P(A/E) = \frac{n(A \cap E)}{n(E)} = \frac{n(A \cap E)/N}{n(E)/N} = \frac{P(A \cap E)}{P(E)} \rightarrow P(A \cap E) = P(E)P(A/E)$$

$$P(A)P(E/A) = P(E)P(A/E)$$



- Obtención de una ecuación generalizada del teorema de Bayes.
  - Sea una característica cualquiera X.
  - Sea A el número de casos de una población estadística en los que X está presente.
  - Sea  $\neg A$  el número de casos de la misma población estadística en los que X está ausente.
  - Sea P una prueba potencialmente resolutiva para investigar la característica X.
  - Sea E el número de casos en los que la prueba da resultados positivos.
  - Sea  $\neg E$  el número de casos en los que la prueba da resultados negativos.

|              | <b>A</b>     | <b>-A</b>    | <b>TOTAL</b> |
|--------------|--------------|--------------|--------------|
| <b>E</b>     | <b>a</b>     | <b>b</b>     | <b>a + b</b> |
| <b>-E</b>    | <b>c</b>     | <b>d</b>     | <b>c + d</b> |
| <b>TOTAL</b> | <b>a + c</b> | <b>b + d</b> | <b>N</b>     |

- a = positivos reales
- b = falsos positivos
- c = falsos negativos
- d = negativos reales

#### ▪ Relaciones

- $P(E / A) = a / (a + c)$       *sensibilidad*
- $P(\neg E / A) = c / (a + c)$
- $P(A / E) = a / (a + b)$
- $P(\neg A / E) = b / (a + b)$
- $P(E / \neg A) = b / (b + d)$
- $P(\neg E / \neg A) = d / (b + d)$       *especificidad*
- $P(A / \neg E) = c / (c + d)$
- $P(\neg A / \neg E) = d / (c + d)$

#### ▪ Prevalencias

- $P(E) = (a + b) / N$        $P(A) = (a + c) / N$
- $P(\neg E) = (c + d) / N$        $P(\neg A) = (b + d) / N$

- Tras conocer las probabilidades a priori ¿cómo se plantea la probabilidad condicional a posteriori de la característica X dado un resultado positivo de la prueba?

$$\begin{aligned}
 Bayes \rightarrow P(A / E) &= \frac{P(E / A)P(A)}{P(E)} : P(E) = \frac{(a + b)}{N} \\
 a &= (a + c)P(E / A) : b = (b + d)P(E / \neg A) \\
 (a + c) &= N \times P(A) : (b + d) = N \times P(\neg A) \\
 a &= N \times P(A) \times P(E / A) : b = N \times P(\neg A) \times P(E / \neg A) \\
 P(E) &= P(A)P(E / A) + P(\neg A)P(E / \neg A) \\
 P(A / E) &= \frac{P(E / A)P(A)}{P(E / A)P(A) + P(E / \neg A)P(\neg A)}
 \end{aligned}$$

- La ecuación

$$P(A/E) = \frac{P(E/A)P(A)}{P(E/A)P(A) + P(E/\neg A)P(\neg A)}$$

es directamente generalizable. Así, si consideramos todos los  $A_i$  posibles obtenemos que:

$$P(A_0/E) = \frac{P(E/A_0)P(A_0)}{\sum_i P(E/A_i)P(A_i)}$$

Que es una expresión general del teorema de Bayes

## TEMA 6

### SISTEMAS CONEXIONISTAS ALIMENTADOS HACIA DELANTE

#### La Neurona Artificial:

- También llamada Elemento de Procesado (EP o PE).
- Valores de entrada y salida:
  - Las señales de e/s de una RNA son números reales.
  - Estos números deben encontrarse dentro de un intervalo.
    - Típicamente entre  $[0,1]$  o  $[-1,1]$ .
  - La codificación más simple es la binaria.
- Conexiones:
  - Las unidades son conectadas a través de conexiones.
  - **Codifican el conocimiento de la red.**
  - Las conexiones poseen valores asociados (pesos).
  - Tipos de conexiones:
    - Excitatorias  $w_{ij} > 0$
    - Inhibitorias  $w_{ij} < 0$
    - Inexistentes  $w_{ij} = 0$
- Bias:
  - Predisposición de una neurona a activarse.
  - Valor constante, similar a un peso, que no recibe entrada (o entrada = 1).
- Función de transferencia:
  - Varias disponibles.
- ¿Cómo hacer que una neurona, ante unas determinadas entradas, emita la salida que se desee?
  - Modificar los valores de los pesos de las conexiones y bias.
  - Proceso denominado **aprendizaje o entrenamiento**.
  - Se necesita un conjunto de patrones. Para cada patrón:
    - Entrada.

- Salida que debe dar la red (salida deseada).
- A partir de esos patrones se fija el valor de los pesos y bias.
- A este conjunto de patrones se denomina **conjunto de entrenamiento**.
- Ejemplo de conjunto de entrenamiento para conseguir una puerta lógica AND:

| <b>X<sub>1</sub></b> | <b>X<sub>2</sub></b> | <b>Salida deseada</b> |
|----------------------|----------------------|-----------------------|
| 0                    | 0                    | 0                     |
| 0                    | 1                    | 0                     |
| 1                    | 0                    | 0                     |
| 1                    | 1                    | 1                     |

## ADALINE:

- **Adaptative Linear Element.**
- Modelo básico de RNA.
- Widrow y Hoff propusieron un método computacionalmente eficiente denominado LMS (least mean square) para determinar parámetros del Adaline (1960).
  - También llamado Regla Delta o regla de Widrow-Hoff.
  - Similar a aplicar gradiente descendente.
  - Muy intuitivo.
- LMS o Regla Delta:
  - Algoritmo de aprendizaje supervisado por corrección de error.
    - **Aprendizaje:** fija los valores de los pesos de las conexiones y de los bias.
    - **Supervisado:** necesita un “tutor” o “supervisor” que, para cada salida, diga qué error comete con respecto a la deseada.
      - Para cada salida, se necesita saber cuál es el valor deseado, para calcular ese error.
      - Necesitamos patrones, cada uno un par de <entradas>/<salida deseada>.
    - **Por corrección de error:** minimiza el error cuadrático medio (ECM) sobre todos los patrones de entrenamiento.
      - Este ECM se calcula, para cada patrón, a partir de la salida que emite la red y la salida deseada para ese patrón.
  - Sea el conjunto de entrenamiento (X,D).
    - X: entradas (conjunto de L vectores de dimensión n).
    - D: salidas deseadas (conjunto de L vectores de dimensión I).
  - Para cada patrón k, la RNA emite una salida y<sub>k</sub>.

$$y_k = \sum_{j=0}^n w_j \cdot x_j$$

- Error para el patrón k:
  - $E_k = d_k - y_k$  (salida deseada menos salida obtenida).
- Error cuadrático (para el patrón k):

$$E_k = \frac{1}{2} (d_k - y_k)^2$$

- Error cuadrático medio (para todos los patrones):

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

- Para minimizar el error según W:
  - Se deriva con respecto a W.
  - Minimización de gradiente.
  - La superficie de error es desconocida.
  - Pero se tiene información local a través del gradiente.
- LMS o Regla Delta: Proceso:
  - La idea es realizar un cambio en los pesos proporcional a la derivada del error, medida en el patrón actual k, respecto del peso j:

$$\Delta_k w_j = -\mu \frac{\nabla E_k}{\nabla w_j}$$

$$y_k = \sum_{j=0}^n w_j \cdot x_j$$

- La regla de aprendizaje del ADALINE se justifica mediante la aplicación del gradiente descendente (derivada del error):

$$\begin{aligned} \frac{\nabla E_k}{\nabla w_j} &= \frac{\nabla (\frac{1}{2} (d_k - y_k)^2)}{\nabla w_j} = \frac{1}{2} \cdot \frac{\nabla ((d_k - y_k)^2)}{\nabla w_j} = 2 \cdot \frac{1}{2} \cdot (d_k - y_k) \frac{\nabla (d_k - y_k)}{\nabla w_j} = \\ &= -(d_k - y_k) \frac{\nabla y_k}{\nabla w_j} = -(d_k - y_k) \frac{\nabla \sum_i w_i \cdot x_{ik}}{\nabla w_j} = -(d_k - y_k) \frac{\nabla (w_j \cdot x_{jk})}{\nabla w_j} = -(d_k - y_k) x_{jk} \end{aligned}$$

- Si los pesos se mueven en dirección  $(y-d)x$ , (dada por  $\frac{\nabla E_k}{\nabla w_j}$ ) se incrementa el error.
- Por tanto, hay que moverse en dirección  $(d-y)x$  (dada por  $-\frac{\nabla E_k}{\nabla w_j}$ ) para decrementar el error.
- Por lo tanto, la modificación de pesos viene dada por:

$$\Delta_k w_j = -\mu \frac{\nabla E_k}{\nabla w_j}$$

$$\frac{\nabla E_k}{\nabla w_j} = -(d_k - y_k) \cdot x_{jk}$$


$$\Delta_k w_j = \mu \cdot (d_k - y_k) \cdot x_{jk}$$

- Modificación del peso j, para un patrón k, en el instante t:

$$w_j(t+1) = w_j(t) + \mu \cdot (d_k - y_k) \cdot x_{jk}$$

$\mu$ : tasa o velocidad de aprendizaje

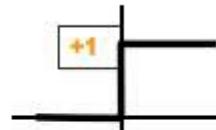
- LMS o Regla Delta: Algoritmo:
  - Se inicializan los pesos de forma aleatoria.
  - Se aplica un patrón de entrada.
  - Se computa la salida que emite la red.
  - Se calcula el error cometido para dicho patrón.
  - Se actualizan las conexiones mediante la ecuación anterior.
  - Se repiten los pasos del 2 al 5 para todos los patrones de entrenamiento.
  - Si el ECM es un valor aceptable, se termina el proceso, si no, vuelve al paso 2.
- Variante: calcular el ECM sobre todos los patrones y actualizar los pesos según ese error.
  - Es decir, se actualizan los pesos una vez introducidos todos los patrones.
  - De esta manera, el algoritmo no es dependiente del orden de introducción de los patrones.
- Aplicaciones de ADALINE:
  - Procesado de señales:
    - Diseño de filtros capaces de eliminar ruido en señales portadoras de información.
    - RNA de predicción de valores futuros en señales.
  - Conclusiones:
    - ADALINE: una sola capa de PE lineales pueden realizar aproximaciones a funciones lineales o asociación de patrones.
    - Se puede entrenar con el algoritmo LMS.
      - Si hay más de una neurona, se puede aplicar LMS sobre todas de forma sucesiva.
    - Limitaciones: sólo aproximaciones lineales.

## PERCEPTRÓN:

- Primer modelo de RNA desarrollado por Rosenblatt en 1958.
  - Anterior al ADALINE.
- Gran interés en los años 60, por su capacidad para aprender a clasificar patrones.
- Estructura: capa de entrada y un único elemento en la capa de salida.
- Diferencias con ADALINE:
  - Función de transferencia umbral (*hardlimiter*).
  - Algoritmo de entrenamiento.

- Capacidad de representación bastante limitada: un único EP en la capa de salida.
- Capaz de resolver problemas linealmente separables.
  - Entradas bidimensionales: separación a través de una línea.
  - Entradas tridimensionales: separación a través de un plano.
  - Entradas n-dimensionales: separación a través de un hiperplano.
- Ejemplo: función OR:

| $x_1$ | $x_2$ | $y$ |
|-------|-------|-----|
| 0     | 0     | 0   |
| 0     | 1     | 1   |
| 1     | 0     | 1   |
| 1     | 1     | 1   |

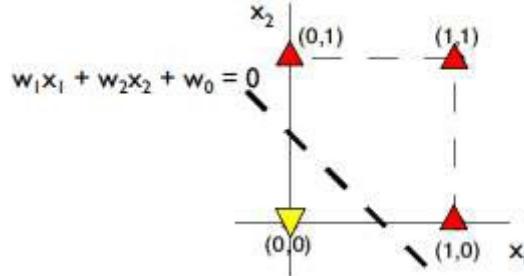


Salida de la red:  $y = f(w_0 + \sum_i w_i x_i) = f(w_0 + w_1 x_1 + w_2 x_2)$

Si  $w_0 + w_1 x_1 + w_2 x_2 > 0$  la salida será 1

En caso contrario, la salida será 0

- El sumatorio que se le pasa como entrada a la función de transferencia representa la recta que separará las dos clases.
  - $w_1 x_1 + w_2 x_2 + w_0 = 0$
- Representados los patrones según las entradas:
  - 4 patrones: (0,0) (0,1) (1,0) (1,1).



- Reglas para la actualización de pesos:

1.  $w_i(t+1) = w_i(t)$  cuando la salida es correcta

$$w_i(t+1) = w_i(t) + x_i(t) \text{ si la salida es } -1 \text{ y debería ser } 1.$$

$$w_i(t+1) = w_i(t) - x_i(t) \text{ si la salida es } 1 \text{ y debería ser } -1.$$

2.  $w_i(t+1) = w_i(t)$  cuando la salida es correcta.

$$w_i(t+1) = w_i(t) + \mu x_i(t) \text{ si la salida es } -1 \text{ y debería ser } 1.$$

$$w_i(t+1) = w_i(t) - \mu x_i(t) \text{ si la salida es } 1 \text{ y debería ser } -1.$$

3. La más utilizada es la regla delta:

$$w_i(t+1) = w_i(t) + \mu(d(t) - y(t))x_i(t).$$

- Con esta regla de aprendizaje se obtiene una convergencia finita si el conjunto de entrenamiento es **linealmente separable**.

## PERCEPTRÓN MULTICAPA:

- Perceptrón multicapa (*Multilayer Perceptron*, MLP).
  - Se conectan más neuronas a la salida de otras.
    - Cada neurona recibe entradas, y computa su salida mediante los pesos y la función de transferencia.
  - Se organiza en capas.
  - Arquitectura general:
    - Capa de entrada:
      - Neuronas de entrada: no computan nada, solo almacenan las entradas para pasarlas a la siguiente capa.
      - Una neurona de entrada por cada entrada.
    - Capa(s) oculta(s):
      - Formadas por neuronas ocultas.
      - No más de dos capas ocultas.
      - No hay método para determinar cuantas neuronas en cada capa oculta.
    - Capa de salida:
      - Emiten la salida de la RNA.
      - Una neurona por cada salida deseada.
    - Conocimiento en la red:
      - Reside en los pesos de las conexiones y bias.
      - No centralizado, distribuido.
      - Ventajas:
        - Autoorganización.
        - Tolerancia a fallos.
      - Desventajas:
        - Imposible de explicar su funcionamiento.
- Generalmente, cada capa está solamente conectada con la capa siguiente.
- Funciones de transferencia:
  - No están limitadas a ser umbral o lineales.
  - Se asume que es la misma para todas las neuronas de la RNA.
  - Funciones más típicas:
    - Escalón o umbral:
      - Para problemas de clasificación.
      - Salidas 0/1 o -1/1.
    - Lineal y lineal mixta:
      - Para problemas de regresión y ajuste de curvas.
- Aproximadores universales:
  - Varios investigadores (Funahashi en 1989 y Hornik, Stinchcombe y White en 1989) demostraron que un perceptrón multicapa con suficientes unidades no lineales **puede aprender cualquier tipo de función o relación continua entre un grupo de variables de entrada y salida.**
  - Esta propiedad convierte a los perceptrones multicapa en herramientas de propósito general, flexibles y no lineales.

- Aplicaciones:
  - Ajuste de funciones y curvas:
    - Reproducir una función matemática desconocida.
  - Clasificación:
    - Sistema que, ante un nuevo patrón, lo clasifique en una clase entre varias posibles.
  - Regresión:
    - Sistema que, ante un nuevo patrón, le da la salida esperada y coherente con los patrones usados en el entrenamiento.
- Entrenamiento:
  - No se puede usar la Regla Delta: está pensada para redes sin capas ocultas.
  - La Regla Delta Generalizada o *Backpropagation* o algoritmo de retroprogramación del error.
    - Generalización de la Regla Delta para RR.NN.AA de múltiples capas y funciones de transferencia no lineales y diferenciables.
- Características del algoritmo de *backpropagation*:
  - Entrenamiento supervisado por corrección de error.
  - Aprendizaje off-line.
  - Capacidad de generalización.
- Notación:
  - n: número de entradas de cada patrón.
  - h: capa oculta o capa de salida.
  - p: patrón.
  - j: PE j en la capa oculta h.
  - k: PE k en la capa de salida.
  - $w_{ji}^h$ : conexión PE i (capa h-1) con PE j (capa h).
  - $i_{pj}^h$ : salida PE j en la capa oculta h para el patrón p.
  - $o_{pj}$ : salida PE j en la capa de salida para el patrón p.

$$i_{pj}^h = f_j^h(neta_{pj}^h) \quad neta_{pj}^h = \sum_{i=0}^n w_{ji}^h x_{pi} \quad o_{pj} = f_j^o(neta_{pj}^o)$$

- En la capa de salida puede haber más de un PE.
  - No basta con calcular un único valor de error:
 
$$\delta_{pk} = (y_{pk} - o_{pk})$$
  - En cambio, el error que hay que minimizar es la suma de los cuadrados de los errores de todas las unidades de salida.

$$E_p = \frac{1}{2} \sum_{k=1}^n \delta_{pk}^2 = \frac{1}{2} \sum_{k=1}^n (y_{pk} - o_{pk})^2$$

- Es posible modificar los pesos de la capa de salida o (conexiones de o-1 a la capa o).
- En las capas ocultas no hay valores de salidas deseadas.
  - No se puede calcular el error ni, por lo tanto, modificar los pesos.
- Solución: propagar el error hacia atrás.
  - El error en la capa de salida  $E_p$  está relacionado con la salida de los PE en la capa anterior (o-1).
  - Las actualizaciones de los pesos en esa capa dependen de todos los términos de errores de la capa de salida.
  - A esto se refiere el término de retropropagación del error o *backpropagation*.
- Comparación de términos de modificación de pesos:

|                        | <b>Capa oculta (h) – neurona j</b>  | <b>Capa de salida (o) – neurona k</b>  |
|------------------------|---|--|
| Término de error       | $\delta_{pj}^h$ se calcula a partir de la salida que emite el PE j y el error de salida retropropagado hacia esa capa<br>$\delta_{pj}^h = \sum_k \delta_{pk}^o w_{kj}^o$    | $\delta_{pk}^o$ se calcula a partir de la salida que emite el PE k ( $f_k^o(neta_{pk}^o)$ ) y el error de salida $\delta_{pk}$<br>$\delta_{pk}^o = \delta_{pk} f_k^o(neta_{pk}^o)$ |
| Variación de los pesos | La variación de los pesos se calcula a partir de ese $\delta_{pj}^h$ y de las entradas que recibe la neurona<br>$w_{ji}^h(t+1) = w_{ji}^h(t) + \mu \delta_{pj}^h i_i^{h-1}$ | La variación de los pesos se calcula a partir de ese $\delta_{pk}^o$ y de las entradas que recibe la neurona<br>$w_{kj}^o(t+1) = w_{kj}^o(t) + \mu \delta_{pk}^o i_{pj}^{o-1}$     |

- **Entrenamiento:**
  - Se pueden emplear todos los datos disponibles para entrenar la red.
  - Lo que se necesita es un subconjunto de datos que cubran todo el espacio de los mismos.
    - **Importante que sean representativos.**
    - Estos patrones que se usan para entrenar guían el proceso de entrenamiento, y la RNA aprenderá en función de ellos.
  - La RNA admite la Generalización:
    - Dados varios vectores de entrada (no pertenecientes al conjunto de entrenamiento), similares a patrones existentes en el conjunto de entrenamiento, la red conocerá las similitudes entre dichos patrones.
    - Patrones que no ha visto el entrenamiento.
  - Si la red se entrena mal o insuficientemente, las salidas pueden ser imprecisas.
    - También, si el conjunto de entrenamiento no fue bien escogido.

- Región de incertidumbre: (entrenado con A,B).
  - Red con 2 unidades ocultas (1 capa oculta).
  - Al minimizar el error los planos que se generan se alinean tan cerca de los patrones de entrenamiento como sea posible.
  - Patrones mal escogidos.
- La capacidad de una red para resolver un problema está ligada a los ejemplos utilizados durante el aprendizaje.
- El conjunto de entrenamiento debe:
  - Ser significativo:
    - Número suficiente de ejemplos.
    - Si el conjunto de entrenamiento es reducido, la red no será capaz de resolver el problema de forma eficaz.
  - Ser representativo:
    - Ejemplos diversos: todas las regiones del espacio de estados deben estar suficientemente cubiertas
  - Si un conjunto de aprendizaje contiene muchos más ejemplos de un tipo que del resto, la red se especializará en dicho subconjunto y no será de aplicación general.
- Entradas: valor real arbitrariamente grande o pequeño.
  - Importante normalizar las entradas para que estén entre 0 y 1 o entre -1 y 1 (depende del problema) **cada entrada por separado.**
  - Dos formas de normalización:
    - Entre máximo y mínimo.
      - Si todos los valores están acotados:
$$X = \frac{x - \min}{\max - \min}$$
    - Usando media y desviación típica:
      - Si los valores son el resultado de una medición y algún valor puede salirse y ser muy alto o muy bajo.
- Salidas:
  - Es necesario normalizar también los valores de las salidas deseadas de la BD para entrenar la red, **cada salida por separado.**
  - Una vez entrenada, cuando se esté utilizando la red, para saber el valor real se desnormaliza el valor que devuelve la red.
- Control de convergencia:
  - Posibilidad de convergencia hacia alguno de los mínimos locales.
    - No se puede asegurar en ningún momento que el mínimo que se encuentre sea global.
      - **Proceso no determinístico.**

- Una vez que la red se asienta en un mínimo, sea local o global, cesa el aprendizaje, aunque el error siga siendo demasiado alto, si se ha alcanzado un mínimo local.
  - Algoritmo de *backpropagation* clásico.
  - Algoritmos más recientes (variantes) son capaces de saltar mínimos locales.
- Cada vez que se entrena da un resultado distinto!
  - Se inicializan los pesos en un punto distinto (aleatorio).
- Si se alcanza un mínimo local y el error es satisfactorio, el entrenamiento ha sido un éxito.
  - Si no sucede así, puede realizarse varias acciones para solucionar el problema:
    - Cambio de arquitectura (más capas ocultas o más PE).
    - Modificación de parámetros de aprendizaje.
    - Emplear un conjunto de pesos iniciales diferentes (es decir, entrenar otra vez).
    - Modificar el conjunto de entrenamiento o presentar los patrones en distinta secuencia.
- El problema del **sobreentrenamiento**:
  - Buen error de entrenamiento pero mala generalización.
  - La red ha aprendido los patrones de entrenamiento no las características que los definen.
  - Si los patrones de entrenamiento tienen ruido, la red lo ha aprendido como si fuera una característica de estos patrones.
  - Causas:
    - Topología demasiado compleja.
    - Entrenamiento durante demasiado tiempo.
      - No parar el entrenamiento a tiempo.
    - No comprobar la generalización de la red mientras entrena.

### Sobreentrenamiento:

- Los patrones se distribuyen en el espacio.
  - Situación ideal: poder visualizarlos y saber cuántas neuronas se necesitan para separarlos.
    - En general, tienen N dimensiones: imposible visualizar.
- El mundo real:
  - No se pueden visualizar los patrones.
  - Estos tienen ruido.
    - Los patrones se han movido de la situación que deberían tener.

- Si se usa una RNA con 4 neuronas en la primera capa y una capa de salida, se podría generar esto:
    - 100% de precisión en el conjunto de **entrenamiento**.
    - Sin embargo, el sistema no generaliza bien: ha aprendido el ruido.
    - Con nuevos patrones, el sistema se comportará mal.
- Primera causa del sobreentrenamiento:
  - Red demasiado compleja para el problema a resolver.
    - A priori no se sabe cuál es la complejidad idónea.
- Segunda causa del sobreentrenamiento:
  - Forzar la red a aprender durante demasiados ciclos.
    - Habitualmente, el sobreentrenamiento se da por ambas causas a la vez.
- Cómo evitar el sobreentrenamiento:
  - Utilizar un conjunto de validación.
    - No interfiere en el entrenamiento pero lo supervisa y lo para si es necesario.
    - Habitualmente, este conjunto se obtiene dividiendo el conjunto de entrenamiento en entrenamiento y validación.
  - A la vez que se entrena la red, se va evaluando con el conjunto de validación.
    - Se realiza una “estimación” de cómo generaliza la red, pero no interviene en el entrenamiento.
      - Se espera que el conjunto de test tendrá un error similar al del entrenamiento.
    - Si el error en el conjunto de validación aumenta, la red se está sobreentrenando.
      - Porque aumenta el error con patrones que no intervienen en el entrenamiento.
  - Parar el entrenamiento:
    - Establecer un nuevo parámetro: número de ciclos sin mejorar la validación.
      - Si transcurre este número de ciclos sin mejorar el error de validación, se para el entrenamiento.
    - Entrenar durante un número de ciclos muy alto y, durante el entrenamiento, memorizar la red con mejor resultado de validación.
      - Cuando finalice el entrenamiento, se devuelve esa red y no la red final entrenada (posiblemente sobreentrenada).
    - Etc.
- Para entrenar correctamente hacen falta 3 conjuntos de patrones:
  - Entrenamiento.
    - Guía el proceso de entrenamiento.
  - Validación.
    - No participa directamente en el entrenamiento.

- Supervisa el entrenamiento y lo para si es necesario.
- Dictamina cuál es la red a devolver cuando finaliza el entrenamiento.
- Test.
  - No interviene en nada en el entrenamiento.
  - Se evalúa una vez haya finalizado el entrenamiento, con la red resultante (posiblemente la de mejor validación).
  - Analiza la capacidad de generalización de la red.
  - Es la única manera de saber si la red está bien entrenada.
    - El conjunto de entrenamiento guía el entrenamiento y el de validación, de alguna manera, también participa.
    - Es necesario este conjunto puesto que no interviene en nada.
  - Generalmente este conjunto se tiene antes de entrenar, así que, a medida que se entrena, se va calculando ese error de test.
    - Pero este valor de error NO se utiliza para tomar ninguna decisión.

## Aplicaciones de las RNA:

- Se usan para el reconocimiento de patrones.
- Problemas donde más importante es el patrón que los datos exactos.
- ¿Cuándo usar una RNA o aplicar programación convencional?
  - RNA:
    - No se sabe cómo resolver el problema.
    - No se puede escribir un algoritmo que resuelva el problema.
    - No se puede explicitar el conocimiento.
    - El sistema hallado no requiere explicación.
- Aplicaciones:
  - Clasificación:
    - Clasifica objetos en un número finito de clases, dadas sus propiedades (entradas).
    - Busca una función de mapeo que permita separar la clase 1 de la clase dos y esta de la clase 3...
    - El número de clases es finito y conocido.
    - Se conoce la pertenencia a la clase de cada patrón.
    - Número de salidas de la RNA:
      - Si solo hay dos clases: 1 única salida.
      - Si hay más de dos clases: 1 neurona de salida por clase.
  - Predicción:
    - Intenta determinar la función que mapea un conjunto de variables de entrada en una (o más) variables de salida.
    - Es básicamente numérica.
    - Está basada en supuestos estadísticos.
    - Ejemplo:
      - Monitoreo la reserva de plazas en empresas de aviación.

- *Clustering* (Clasificación no supervisada):
  - Intenta agrupar una serie de objetos en grupos.
  - Cada objeto es representado por un vector de atributos n-dimensional.
  - Los objetos que forman cada grupo deben ser similares.
  - La similaridad es medida del grado de proximidad.
  - Luego cada grupo es etiquetado.
  - No se conoce a priori el número de clases o *clusters*.
  - No se conoce a qué *cluster* pertenece cada patrón.
    - Aprendizaje no supervisado.
    - **No para perceptrones multicapa.**
- Aproximación de curvas (*fitting*):
  - Reconocer la curva que subyace tras un conjunto de patrones  $(x_i, y)$ .
- Regresión:
  - Genérico.
  - A partir de una BD o conjunto de patrones, intentar encontrar una RNA que modelice la relación entre dos conjuntos de ellos.
    - Patrones: (entradas, salidas deseadas).
    - Relación desconocida.
    - Modelo debe de ser generalizable, es decir, que ante nuevos patrones desconocidos devuelva unas salidas coherentes.
- Control.

## Conclusiones:

- Ventajas del uso de RR.NN.AA.:
  - Debido a su constitución y a sus fundamentos, las RR.NN.AA. presentan un gran número de características semejantes a las del cerebro, por ejemplo:
    - Aprender de la experiencia.
    - Generalización de casos anteriores a nuevos casos.
    - Abstraer características esenciales a partir de entradas con información irrelevante.
  - Esto hace que se apliquen en múltiples áreas, por las ventajas que ofrecen:
    1. Aprendizaje Adaptativo.
      - Capacidad de aprender a realizar tareas basadas en un entrenamiento o una experiencia inicial.
    2. Autoorganización.
      - Las redes neuronales usan su capacidad de aprendizaje adaptativo para organizar la información que reciben durante el aprendizaje y/o la operación.

- Una RNA puede crear su propia organización o representación de la información que recibe mediante una etapa de aprendizaje.
  - Esta autoorganización permite a las redes neuronales de responder apropiadamente cuando se les presentan datos o situaciones a los que no habían sido expuestas anteriormente.
3. Tolerancia a fallos: Dos aspectos distintos:
- Pueden aprender a reconocer patrones con ruido, distorsionados, o incompleta.
  - Pueden seguir realizando su función (con cierta degradación) aunque se destruya parte de la red.
  - Comparados con los sistemas computacionales tradicionales, los cuales pierden su funcionalidad en cuanto sufren un pequeño error de memoria, en las redes neuronales, si se produce un fallo en un pequeño número de neuronas, aunque el comportamiento del sistema se ve influenciado, sin embargo, no sufre una caída repentina.
    - La razón por la que las redes neuronales son tolerantes a fallos es que tienen su información distribuida en las conexiones entre neuronas, con cierto grado de redundancia.
    - En cambio, en la mayoría de los ordenadores algorítmicos y sistemas de recuperación de datos se almacena cada pieza de información en un estado único, localizado y direccionable.
4. Procesamiento en paralelo.
5. Operación en tiempo real.
- Los computadores neuronales pueden ser realizados en paralelo, y se diseñan y fabrican máquinas con hardware especial para obtener esta capacidad.
6. Fácil inserción dentro de la tecnología existente.
- Debido a que una red puede ser rápidamente entrenada, comprobada, verificada y trasladada a una implementación hardware de bajo costo, es fácil insertar RR.NN.AA. para aplicaciones específicas dentro de sistemas existentes (chips, por ejemplo).
  - De esta manera, las redes neuronales se pueden utilizar para mejorar sistemas de forma incremental, y cada paso puede ser evaluado antes de acometer un desarrollo más amplio.
- Desventajas del uso de RR.NN.AA.:
    1. Implementación deficiente como sistemas paralelos.
      - Si se implementan en máquinas en serie, solo se ejecuta una instrucción a la vez.
      - Por ello, modelar procesos paralelos en máquinas serie puede ser un proceso que consume mucho tiempo.

2. No existencia de una regla para construir una RNA para un problema dado.
  - Hay muchos factores a tener en cuenta: el algoritmo de aprendizaje, la arquitectura, el número de neuronas por capa, el número de capas, la representación de los datos, etc.
3. Imposibilidad de explicar el funcionamiento de una RNA.
  - Una RNA funciona como una “caja negra”: ante unas entradas, genera unas salidas, pero no es posible comprender la relación entre ellas.
  - Cuando se modela estadísticamente se puede ver qué variables forman parte del modelo o cuáles de las que finalmente se utilizaron para modelar fueron seleccionadas y las relaciones entre las mismas.
  - Esto provoca que en algunos entornos en los que es necesario explicar el funcionamiento, no se puedan utilizar, por ejemplo, en medicina.
4. Cuanto más flexible se requiere que sea una RNA, más información habrá que enseñarle.
  - Más patrones de entrenamiento.

# TEMA 7

## OTROS MODELOS DE

### SISTEMAS CONEXIONISTAS

#### Redes Autoorganizativas:

##### Introducción:

- La autoorganización en Biología: proceso a través del cual el comportamiento global de un sistema se obtiene solamente como resultado de la interacción local entre los componentes que integran el sistema.
- La **autoorganización** en Sistemas Conexiónistas consiste en la modificación repetida de los pesos de las conexiones en respuesta a modos de activación y siguiendo unas reglas preestablecidas, hasta el desarrollo final de la estructura o sistema.
  - No existen observadores globales, no existe un “jefe” que determine el comportamiento, no se cuenta con la *salida deseada*.
  - Comportamiento emergente: características surgen de forma inesperada a partir de la interacción entre los componentes del sistema.
  - La red crea su propia representación de la información que recibe mediante la etapa de aprendizaje.
  - La información relevante debe ser localizada en los propios datos de entrada (redundancia), sin redundancia sería imposible encontrar características en los datos.

##### Problemas a resolver:

- El reconocimiento automático, descripción y agrupamiento de **patrones** son actividades importantes en una gran variedad de disciplinas científicas, como biología, psicología, medicina, inteligencia artificial, etc.
- **Patrón**: entidad representada por un conjunto de propiedades y las relaciones entre ellas (vector de características). Por ejemplo, un patrón puede ser:
  - Una señal sonora y su vector de características, el conjunto de coeficientes espectrales extraídos de ella (espectrograma).
  - Una imagen de una cara humana, de la cual se extrae el vector de características formado por un conjunto de valores numéricos calculados a partir de la misma.
- Es importante conocer qué tipo de proceso puede realizar la red autoorganizativa, qué representa la salida de la red y qué problemas puede resolver.

#### SOM – Mapas autoorganizativos:

- La estructura anatómica e histológica cerebral revela que la **ubicación espacial** de las células nerviosas tiene gran importancia.

- En la corteza auditiva primaria las neuronas se distribuyen en un mapa según la frecuencia temporal o la tonalidad a la que responden.
  - En la corteza visual las neuronas se organizan en columnas, y en cada columna presentan características similares: selectividad ante la orientación del estímulo.
- Una de las propiedades importantes del cerebro es el significativo **orden de sus unidades de proceso**.
  - Este orden hace que células estructuralmente idénticas rengan una diferente funcionalidad debida a parámetros internos que evolucionan de forma diferente según dicha ordenación.
  - Propiedad fundamental para la representación de imágenes visuales, abstracciones, etc.
- La posibilidad de que la representación del conocimiento en una categoría particular pueda **asumir la forma de un mapa de características** organizado geométricamente sobre la corteza del cerebro, ha motivado una serie de investigaciones que han dado lugar a nuevos modelos de redes neuronales artificiales.
- **La localización** de la neurona en la red, especifica un orden topológico que describe la relación de similitud entre los patrones de entrada.
- “*El supuesto del procesado inteligente de la información puede ser visto como la creación de imágenes simplificadas del mundo real con diferentes niveles de abstracción, en relación a un subconjunto particular de datos observables*” [T. Kohonen].
- El científico finlandés Teuvo Kohonen diseñó en 1982 un modelo llamado **mapa autoorganizativo de características** que consiste en una red de neuronas de 2 capas: capa de entrada y capa de competición (salida).
- Modelo de red neuronal con capacidad para formar mapas de características, simulando los mapas topológicos de los fenómenos sensoriales existentes en el cerebro, a través de una organización matricial de neuronas artificiales, que permite:
  - Conseguir un modelo simplificado de los datos de entrada.
  - Obtener un mapa que muestra gráficamente las relaciones existentes entre los datos, preservando su topología.
  - Proyectar datos altamente dimensionales a un esquema de representación de baja dimensión: representar conjuntos de datos de gran número de atributos en mapas 2D.
  - Encontrar similitudes en los datos: visualmente podemos detectar de forma rápida cómo quedan agrupados patrones con valores próximos entre sí (Ej. Color rojo).
- Características principales:
  - Son redes no supervisadas.
  - **Aprendizaje competitivo**: las células compiten por aprender (por modificar sus pesos), solo hay una célula ganadora:
    - Se actualizan los pesos de la neurona ganadora para acerclarlos al patrón de entrada.

- Así se van asociando a cada neurona de la capa competitiva, un grupo de patrones de entrada similares, generando *clusters* o grupos.
  - Los pesos de las neuronas serán los prototipos, centros o centroides de los *clusters*.
- Presentan una topología predefinida de neuronas en el mapa, el aprendizaje conserva la **relación topológica** (orden topográfico).
- La **vecindad** entre neuronas preserva las relaciones topológicas: las neuronas cercanas responden ante patrones similares.
  - Se puede definir para cada célula de la capa competitiva el conjunto de células próximas que serán sus vecinas (según arquitectura uni, bi o n-dimensional de la capa).

### **SOM – Mapas autoorganizativos de Kohonen:**

- Se trata de categorizar los datos de entrada, agrupar los datos similares:
  - Se persigue que patrones parecidos hagan reaccionar a las mismas neuronas.
  - Cada neurona se especializa en determinados patrones de entrada.
  - Cada grupo de patrones estará representado por un prototipo (pesos de la neurona).
  - El sistema debe relacionar cada prototipo con los patrones de entrada que representa.
  - Cada prototipo será utilizado, una vez entrenada la red, para clasificar patrones de datos nuevos y desconocidos.
- En el aprendizaje competitivo:
  - Si un patrón nuevo pertenece a una categoría reconocida previamente, entonces la inclusión de este nuevo patrón a esta categoría o clase matizará la representación de la misma.
  - Si el nuevo patrón no pertenece a ninguna de las categorías reconocidas anteriormente, entonces la estructura y los pesos de la red neuronal serán ajustados para reconocer a la nueva categoría.
- La capa de entrada recibe la señal de entrada a la red.
  - La **dimensión** de la entrada depende del número de atributos que tengan los patrones de entrada, si  $n$  atributos, entonces:  $e = \{e_1, \dots, e_n\}$ .
  - Cada neurona de la capa de entrada **se conecta con todas** las neuronas de la capa de competición (no hay conexiones entre las neuronas de competición).
  - Si hay  $m$  neuronas en la capa de competición, los pesos de las conexiones entre las dos capas se definen mediante una matriz  $(n, m)$ .
  - El **vector de pesos** de cada neurona de la capa de competición tendrá el mismo nº de componentes que el vector de entrada.
  - Al tener la misma dimensión, se pueden **comparar entre sí** los dos vectores, definiendo una función de distancia entre ellos.

- Los mapas de Kohonen utilizan usualmente la **distanzia euclídea**, la salida de las neuronas de la capa de competición se calcula aplicando dicha función de distancia:

$$d = \sqrt{\sum_{i=1}^n (e_i - \mu_{ij})^2}$$

•  $t_j$  – salida de la neurona  $j$   
 •  $e_i$  – entrada  $i$   
 •  $\mu_{ij}$  - peso de  $e_i$  a neurona  $j$

- **Funcionamiento del modelo – FASE APRENDIZAJE:**

- Objetivo: acercar los pesos a los vectores de entrada.
- Los pesos de las conexiones se inicializan aleatoriamente al principio de la fase de aprendizaje.
- Se introducen todos los patrones de entrenamiento un número suficiente de veces.
- Cada vez que la red recibe un patrón o vector de entrada, se calcula la salida de todas las células de la capa de competición (cálculo de distancias).
- Gana la célula de **menor distancia** con ese vector de entrada. Dicha célula tendrá salida 1 y el resto salida 0.
- Por tanto, los **pesos de la neurona ganadora** se modifican siguiendo la ecuación siguiente, donde  $\alpha$  es la tasa de aprendizaje, que habitualmente decrece con el tiempo (primero tasa elevada y luego baja) hasta llegar a 0 o al final de las iteraciones de entrenamiento (finaliza el aprendizaje):

$$\begin{aligned}\mu_{ij}(t+1) &= \mu_{ij}(t) + \Delta\mu_{ij} \\ \Delta\mu_{ij} &= \alpha(e_i - \mu_{ij})\end{aligned}$$

- **Funcionamiento del modelo – FASE APRENDIZAJE (vecindad):**

- Vecindad:** indica qué otras células aprenden cuando se activa una neurona.
- Los pesos de las conexiones de las células vecinas se modifican en función de la distancia, con lo que  $\alpha$  se dividirá por una función distancia de vecindario cuando se aplique a las vecinas la modificación de los pesos, siendo  $c_i$  la célula ganadora y  $c_j$  la vecina:

$$\Delta\mu_{ij} = \frac{\alpha}{d(c_i, c_j)} (e_i - \mu_{ij})$$

- La **vecindad** también se modifica con el tiempo o el número de iteraciones, inicialmente es mayor y se va reduciendo para estabilizar el aprendizaje.
- La **vecindad** se modifica con el tiempo o el número de iteraciones, inicialmente es mayor y se va reduciendo para estabilizar el aprendizaje.
- Vecindad:** regulada por 3 parámetros:
  - Radio de vecindad:** amplitud del alcance de las neuronas afectadas por vecindad.
  - Topología de la vecindad:** neuronas que se consideran vecinas inmediatas.

- Función de vecindad: cuantificación del grado de vecindad proporcional a distancia a la ganadora (centro).
- Las neuronas pertenecientes a la región de vecindad o interés del ganador son también modificadas para conseguir que la red cree **regiones** que respondan a valores muy próximos al del patrón de entrenamiento.
- Como consecuencia, patrones de entrada que no se hayan usado para el entrenamiento serán similares a algún vector de pesos de una región y serán correctamente agrupadas -> **demuestra la Generalidad de esta estructura**.
- 2 ideas centrales en las que se basa Kohonen para desarrollar sus estructuras usando aprendizaje autoorganizativo y competitivo: “*El proceso de adaptación de pesos y el concepto de geometría topológica de elementos de proceso*”.
- **Funcionamiento del modelo – FASE DE APRENDIZAJE (algoritmo).**  
*Tendencia del proceso de aprendizaje:* cada neurona tiende a colocarse en el centroide de aquellos grupos de ejemplos de entrada para los cuales es la neurona ganadora.
  1. Inicializar pesos.
    - Asignar a los pesos valores pequeños aleatorios.
  2. Presentar una entrada.
    - El conjunto de aprendizaje ( $\xi$ ) se presenta repetidas veces hasta llegar a la convergencia de la red.
    - Actualizar  $\alpha$  (ir reduciendo su valor).
  3. Propagar el patrón de entrada hasta la capa de competición.
    - Obtener los valores de salida (distancias) de las neuronas de dicha capa.
  4. Seleccionar la neurona ganadora (BMU – *best-matching unit*).
    - La de menor distancia al patrón.
  5. Actualizar conexiones entre capa de entrada y la BMU.
    - Actualizar también los pesos de sus vecinas según el grado de vecindad.
    - 1. Fase Ordenación: valores altos de  $\alpha$  y radio vecindario. 2. Fase Convergencia: valores bajos.
  6. Si  $\alpha$  se mantiene por encima del umbral de parada (o no se ha llegado a T iteraciones), volver a 2, en caso contrario FIN.
- **Funcionamiento del modelo – Validar calidad del SOM.**
  - Se repite el proceso de entrenamiento N veces, cada una de ellas con una configuración inicial diferente de los vectores de pesos sinápticos.
  - Para determinar la calidad del SOM y ayudar en la selección de sus parámetros de tamaño y aprendizaje adecuados se emplean el *error de cuantización medio* y *medidas de preservación de la topología*.
    - Se calcula el error de representación para cada uno de los mapas obtenidos y se selecciona el de menor error.
    - Este error de cuantización medio, evalúa el grado de adaptación del mapa SOM a los datos de entrada, siendo mejor el que menor

error obtenga entre las conexiones de las BMU y los vectores (patrones de entrada) a los que representa.

- Una vez entrenada la red, se vuelven a procesar todos los patrones de entrenamiento, obteniendo para cada uno de ellos la neurona BMU, y se calcula el error de cuantificación medio como:

$$E = \frac{1}{N} \sum_{i=1}^N \|e_i - w_{bm}\|^2 \quad \text{siendo } N \text{ el número de vectores de entrada.}$$

- Además, se emplean **Medidas de preservación de la topología**:
  - El error de cuantización indicado, no tiene en cuenta la estructura del mapa de la capa competitiva de la SOM, la información sobre las relaciones existentes en los patrones de entrenamiento.
    - Por lo que también es necesario medir la adaptación de la topología de la red a las características topológicas del espacio de datos de entrenamiento.
  - Cuando una red SOM converge, la distribución de la estructura regular definida en la capa de salida se suele distorsionar, pudiendo aparecer efectos adversos (neuronas que representan a datos de entrada similares pero que están alejadas, etc.).
    - En este caso el error de cuantización puede ser bueno, pero a pesar de ellos existen problemas.
  - Es necesario emplear además **medidas de preservación de la topología para validar la calidad** de la red entrenada que permitan comprobar que:
    - Los prototipos de las neuronas que se encuentran próximas en el mapa también se encuentren próximos en el espacio de entrada de datos.
    - Los datos de entrada similares quedan representados por neuronas próximas en el mapa.

- **Funcionamiento del modelo – MODO OPERACIÓN:**

- Se introduce un patrón de entrada para conocer su prototipo.
- Cada neurona de salida calcula la similitud entre el vector de entrada y su vector de pesos.
- Vence la neurona con mayor similitud: será la categoría seleccionada por la red para agrupar ese patrón de entrada o el prototipo seleccionado.

### **Mapas auto-organizativos:**

- Una demostración visual del funcionamiento de la red la ofrecen los *mapas topológicos bidimensionales de Kohonen*.
  - Distribuyen los elementos de la red (puntos que representan a los pesos) a lo largo de un conjunto de patrones de entrada.
  - Cuando las entradas son de 2 atributos o 3 (2D y 3D), los pesos representados por puntos coordenadas en 2D se van aproximando a los

- datos de entrada que están ahí representados, en un cuadrado 2D, por ejemplo.
- Se representan los puntos (pesos de las neuronas) conectados con los de sus vecinas, no es una conexión física, sino una relación entre neuronas a efectos de vecindario (conexión topológica).
    - Grado de vecindad: número de células que habrá que recorrer para llegar de una célula a otra.
  - *Mapas topológicos bidimensionales de Kohonen:*
    - A medida que se introducen los patrones de entrada, la red los va aprendiendo, varía el valor de sus conexiones, trasladándose hacia las entradas.
    - Donde hay mayor densidad de entradas, habrá más densidad de “puntos conexiones – neuronas”.
    - Se comprueba si la red está comportándose correctamente si no se cruzan las líneas.
  - **Aplicaciones:** Tipos de problemas que pueden resolver los SOM, condicionarán cómo se interprete lo que representa la salida de un SOM.
    - **Agrupamiento de patrones (clustering):** A partir de un conjunto de entrada se desea determinar si se puede dividir ese conjunto en diferentes grupos o *clusters* con un centro o patrón del *cluster*.
      - Permite separar datos en grupos (a priori no sabemos cuántos existen).
      - Facilita indicar a qué grupo pertenece cada dato de entrada determinando su neurona ganadora.
      - Permite caracterizar cada grupo mediante los pesos de la neurona que representa ese grupo.
    - **Prototipado:** Similar al anterior, en lugar de interés por los grupos, interesa obtener un prototipo del grupo al que pertenece cada patrón de entrada.
      - Usa los pesos de la ganadora para determinar ese prototipo.
    - **Análisis de componentes principales:** Se trata de detectar qué elementos/atributos del conjunto de entrada caracterizan en mayor grado ese conjunto de datos. Seleccionar las entradas de la red realmente necesarias, sin pérdida de información – supone reducción de dimensionalidad.
      - Las demás entradas podrán eliminarse sin una pérdida significativa de información.
    - **Extracción y relación de características:** Se pretende organizar los vectores de entrada en un mapa topológico.
      - A partir de la red entrenada, patrones parecidos producirán respuestas similares en neuronas cercanas.
      - Si existe una organización global de patrones de entrada, se verá reflejada en la salida de la red.
      - Se ubican entradas parecidas y/o relacionadas en zonas próximas de la red.

- Estas categorías no son disjuntas. Ej.: componentes principales para reducir la dimensionalidad y después aplicar clustering.
- **Ventajas de SOM:**
  - Transparentes a los datos de entrada (se limitan a comparar vectores).
  - Adaptación local de los vectores de referencia a la densidad de probabilidad de los datos.
  - Facilidad de visualización gracias a la malla que conforma la topología de la capa de salida.
  - Facilidad de integración con otras técnicas.
- **Limitaciones o inconvenientes de SOM:**
  - Necesidad de determinar la arquitectura exacta de la red antes de someterla al entrenamiento, así como la imposibilidad de modificar su diseño durante el mismo tiempo.
    - Dimensión de la red (se tiene que fijar a priori – más neuronas > tiempo entrenamiento).
    - Velocidad y vecindad del aprendizaje.
  - Algunas neuronas pueden no ser entrenadas: pueden existir vectores de pesos muy distanciados de las entradas, con lo cual, nunca ganarán (este tipo de neurona es común cuando se aplica el algoritmo a redes y conjuntos de datos muy grandes).
    - Patrones cercanos activan neuronas distantes.
    - Neuronas vecinas pueden ser activadas por patrones distantes.
  - No se puede medir hasta qué punto es buena la neurona vencedora, la neurona que gana es la más cercana a la entrada, pero no podemos cuantificar si está cerca o lejos de los pesos adecuados.
  - Importante en estos sistemas: la inicialización de pesos.
  - Son caros computacionalmente cuando se incrementa la dimensión de los datos.

## **Otros modelos autoorganizativos: Crecimiento de Redes.**

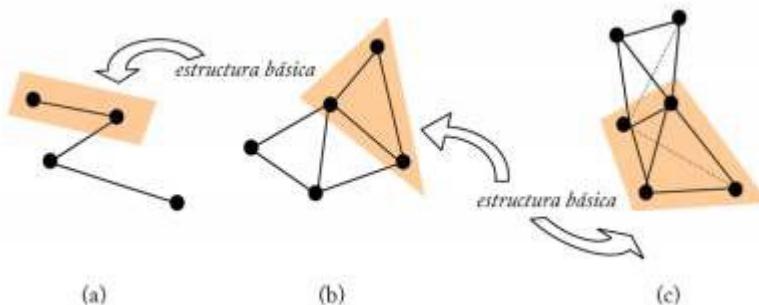
- A pesar de las interesantes características del SOM de Kohonen, en la década de los años 90 se representaron nuevos modelos de mapas autoorganizativos que pretendían solucionar los inconvenientes.
- Al trabajar con SOM hay que definir previamente la estructura de la red.
  - Esta definición suele ser bastante compleja por la falta de información que normalmente se tiene sobre el espacio de entrada.
  - La topología estática de los SOM limita a veces su capacidad para ajustarse a los datos.
- Para solucionar este problema aparece una nueva filosofía a la hora de aplicar redes de neuronas a problemas de agrupamiento: **Crecimiento de Redes**.
  - Se trata de iniciar el aprendizaje con un conjunto pequeño de elementos de proceso (EP) y que sea la propia red la que determine de forma autónoma su estructura, a partir de los datos de entrenamiento se generan los EP necesarios y las conexiones entre ellos.

- Esta característica permite obtener modelos simplificados de los datos **con mejores medidas de la preservación de la topología** que las redes de Kohonen.
- Se construyen partiendo de estructuras básicas (*incluso hipertetraedros de dimensión k*).
- Para llegar a la estructura final de la red se realiza un proceso donde se van añadiendo y borrando EP y conexiones, al tiempo que se sigue un proceso de autoorganización similar al que se produce en el modelo propuesto por Kohonen.
- Incluso pueden existir mallas de vecindad separadas en la capa de salida.
- Mantienen la capacidad de ofrecer un mapa para visualizar las relaciones existentes entre los patrones de entrada.
- Inserción de Neuronas:
  - Para poder determinar dónde y cómo se ha de incluir una nueva neurona se introduce un nuevo concepto, **valor resource** (error de cuantización): valor asignado a cada neurona que medirá lo lejos que se encuentra de la estructura ideal (se obtiene del error con respecto a los patrones de entrada).
    - Este valor irá variando a medida que la red vaya evolucionando.
  - Siempre después de un nº constante de “modificaciones de pesos” sobre la red actual, se analizará la necesidad o no de añadir nuevas neuronas.
    - Añadir un nuevo elemento a la red conlleva el adecuar las conexiones entre los elementos vecinos para que la nueva estructura sea consistente.
  - Nueva neurona: tanto su vector de pesos como su valor *resource*, se calcularán a partir de los parámetros de sus neuronas vecinas.
- Borrado de Neuronas:
  - Cuando el conjunto de patrones o vectores de entrada que pueden activar una determinada neurona tienen una probabilidad de aparición muy baja, podemos estimar que el coste de mantener esa neurona dentro de la estructura de la red no compensa y, por lo tanto, se elimina.

### **Growing Cell Structures (GCS):**

- Entre los modelos de crecimiento de redes, se creó el modelo conocido como Growing Cell Structures (GCS), planteado por B. Fritzke en 1994.
- La red GCS ofrece como principal ventaja frente al modelo de Kohonen la dinamicidad de la arquitectura de la capa de salida de la red durante la fase de entrenamiento de la misma.
  - Orientada a cuantificación vectorial – clustering (no supervisada).
  - La adición de neuronas (EP) se realiza en las regiones con más patrones de entrenamiento.
  - Su estructura de la capa de salida es simple y está formada por **estructuras k-dimensionales básicas**.

- El número de parámetros de entrenamiento y la complejidad de su configuración son normalmente bajos, aunque a veces difíciles de interpretar (inconveniente).
  - El objetivo es conseguir una estructura cuyos vectores de entrenamiento se distribuyan de acuerdo a la distribución (desconocida) de los vectores de entrada.
    - Patrones/vectores de entrada similares se agruparán en neuronas topológicamente próximas.
    - Neuronas vecinas agruparán vectores de entrada similares.
    - Regiones del espacio de entrada con una densidad alta serán representadas por un número elevado de neuronas y viceversa.
  - Durante el proceso de adaptación de pesos de la red, se insertan y eliminan neuronas/conexiones de vecindad asegurando que la capa de salida queda compuesta por **estructuras k-dimensionales básicas**.
  - Se muestran estructuras básicas para k igual a 1 (a – segmento), 2 (b – triángulo) y 3 (c – tetraedro).
  - k dimensiones (hipertetraedros). Existirán:  $k+1$  vértices y  $(k+1)*k/2$  aristas.



- Exceptuando la topología de la capa de salida, el resto de la arquitectura de una red GCS es idéntica a la del modelo de Kohonen.
  - respecto a la dinámica de procesamiento de vectores, el entrenamiento es competitivo, pero cada neurona tiene además de vecinas y vector de pesos, un **contador** que almacena el **error de cuantización**: cuadrado de la distancia euclídea al patrón de entrada cuando dicha neurona es ganadora (BMU):
    - Se utilizan 2 tasas de aprendizaje: una para modificar los pesos de la BMU y otra para los de sus vecinas inmediatas.

## **Growing Neural Gas (GNG):**

- En 1995 Fritzke planteó el modelo Growing Neural Gas (GNG) Gas Neuronal Creciente.
  - Es otro modelo de red autoorganizativa dinámica, pero a diferencia de la GCS no mantiene en el proceso de inserción y eliminación de neuronas ninguna estructura k-dimensional estricta o fija en la capa de salida de la red.

- Se adapta a diferentes dimensiones y puede ir variando la dimensión de la estructura dimensional que emplea en función de los datos de entrada.
  - A diferencia de la GCS, la GNG trabaja más con **aristas** que con triángulos.
  - Con la inserción y borrado de aristas la red trata de adaptarse a los vectores de entrada.
  - Los parámetros introducidos son constantes en el aprendizaje.
  - En los mapas de Kohonen, las conexiones de vecindad son laterales formando una cruz en cada unidad, en GNG, una unidad puede tener mucho más de cuatro vecinos, generando diversas figuras geométricas.
  - Se trata, por tanto, de una red con mayor capacidad de aprendizaje.
- Se adapta a diferentes dimensiones y puede ir variando la dimensión de la estructura dimensional que emplea en función de los datos de entrada.

## TEMA 8

### COMPUTACIÓN EVOLUTIVA

¿Qué es la evolución?

- Proceso de optimización natural.
- Historia:
  - Lamarck
    - Herencia de caracteres adquiridos.
  - Darwin “El origen de las especies”
    - Selección natural.
  - Mendel
    - Mecanismo de herencia.
  - De Vries
    - Mutación. Cambios abruptos.
  - Neo-darwinismo
    - Suma de las anteriores.
  - Teoría Neutralista de evolución molecular
    - Deriva genética.

Ideas desde la Naturaleza.

- Aplicaciones desde la naturaleza:
  - Leonardo da Vinci
    - Vuelo de pájaros y desarrollo posterior de aviones (año 1500).
- Computación evolutiva (CE).
  - Definir un problema sin solución algorítmica.
  - Población de individuos o posibles soluciones.
  - Operadores genéticos:
    - Selección.
    - Cruce y mutación.
    - Sustitución.

Orígenes:

Basada en la teoría de la evolución de Darwin.

- John Holland, “planes reproductivos”: técnica que incorpora la selección natural en un programa de computadora.
- Su **objetivo**: que las computadoras aprendieran por sí mismas.
- John Holland se preguntaba cómo lograba la naturaleza crear seres cada vez más perfectos.  
A principios de los 60 aprendió que la evolución era una forma de adaptación más potente que el simple aprendizaje, y tomó la decisión de aplicar estas ideas para desarrollar programas bien adaptados para un fin determinado.
- David Goldberg conoció a Holland y se convirtió en su estudiante. Fue uno de los primeros que trató de aplicar los algoritmos genéticos a problemas industriales.

## Investigadores e innovadores:

- 1948 Turing: propuso la búsqueda por medios evolutivos o genéticos.
- 1962 Bremermann: propone la optimización a través de la recombinación y la evolución.

1964 Rechenberg: introdujo los conceptos de estrategias de evolución.

- 1965 L. Fogel, Owens and Walsh: introdujo la programación evolutiva.
- 1975 Holland: Realmente es el creador de la teoría en 1975 y reedita su libro en 1992.
- 1989 D. Goldberg: Promueve el uso de algoritmos genéticos para aprendizaje y optimización.
- 1992 Koza: introduce la programación genética.

## Orígenes: ¿Qué es un Algoritmo Genético?

- ¿Podemos crear un algoritmo con la misma filosofía que emplea la naturaleza?
  - En respuesta a esto nacieron los algoritmos genéticos.
- Los algoritmos genéticos establecen una analogía entre el conjunto de posibles soluciones de un problema y el conjunto de individuos de una población natural.

## Orígenes: Fundamentos de los AG:

- La naturaleza utiliza potentes medios para impulsar la evolución satisfactoria de los organismos, el proceso de selección natural.
- Los organismos que son poco aptos para un determinado ambiente mueren, en tanto que los que están bien adaptados para vivir, **se reproducen**, y transmiten sus caracteres.
- Ocasionalmente se producen **mutaciones** al azar, y aunque implican la pronta muerte del individuo mutado, algunas mutaciones dan como resultado nuevas y satisfactorias especies.

## Algoritmos Genéticos.

Los algoritmos Genéticos (AG) son algoritmos de búsqueda inspirados en procesos de selección natural.

Se aplican principalmente en problemas de optimización. Se comportan de un modo muy eficaz en problemas de superficie compleja, con múltiples mínimos locales y grandes espacios de búsqueda.

Está justificado su uso en aquellos problemas cuya complejidad no permita una solución directa. Por ejemplo los no resolubles polinómicamente (NP-duros).

## Componentes de un AG:

Generalmente se acepta que un Algoritmo Genético debe tener 5 componentes:

1. Una representación genética de las soluciones del problema, es decir, representación de las variables que intervienen en cadenas de bits/caracteres.

2. Una forma de crear una población inicial de soluciones.
3. Una función de evaluación en términos de conveniencia o adaptación de la solución evaluada.
4. Operadores genéticos que cambien la composición de los descendientes.
5. Valores para los parámetros utilizados por los Algoritmos Genéticos (N, Pc, Pm, ...).

### Población y Evaluación de individuos:

- Se genera aleatoriamente la población inicial.
- A cada uno de los individuos o cromosomas generados se le aplicará la **función de aptitud o evaluación**.
- Esta función que debe ser capaz de “castigar” a las malas soluciones, y de “premiar” a las buenas, de forma que sean estas últimas las que se propaguen con mayor rapidez.
- Base para determinar qué soluciones tienen mayor o menor probabilidad de sobrevivir.