Contenido

TEMA 1. Introducción a la IA.

Aspectos de la inteligencia.

- La memoria.
- El pensamiento abstracto y el razonamiento.
- El lenguaje y la comunicación.
- El aprendizaje.
- La resolución del problema.
- La creatividad

Tipos de inteligencia según Howard Gardner.

- **Inteligencia Lingüística.** La que tienen los escritores, los poetas, los buenos redactores.
- **Inteligencia Lógica-matemática.** La que utilizamos para resolver problemas de lógica y matemáticas. Es la inteligencia que tienen los científicos. Se corresponde con el modo de pensamiento del hemisferio lógico.
- **Inteligencia Espacial.** Consiste en formar un modelo mental del mundo en tres dimensiones. Es la inteligencia que tienen los marineros, los ingenieros, los cirujanos, los escultores, los arquitectos o los decoradores.
- **Inteligencia Musical.** Es, naturalmente, la de los cantantes, compositores, músicos y bailarines.
- **Inteligencia Corporal-kinestésica.** Es la capacidad de utilizar el propio cuerpo para realizar actividades o resolver problemas. Es la inteligencia de los deportistas, los artesanos, los cirujanos y los bailarines.
- **Inteligencia Intrapersonal.** Es la que nos permite entendernos a nosotros mismos. No está asociada a ninguna actividad concreta.
- **Inteligencia Interpersonal.** La que nos permite entender a los demás y la solemos encontrar en los buenos vendedores, políticos, profesores o terapeutas.
- **Inteligencia Emocional.** Es formada por la inteligencia intrapersonal y la interpersonal, y juntas determinan nuestra capacidad de dirigir nuestra propia vida de manera satisfactoria.
- **Inteligencia Naturalista.** La que utilizamos cuando observamos y estudiamos la naturaleza. Es la que demuestran los biólogos o los herbolarios.
- **Inteligencia Cibernética.** La que desarrollan las personas estudiando y aprovechando la ciencia que se ocupa de los sistemas de control y telecomunicaciones.

Historia.

El encuentro conocido como la **conferencia de Dartmouth** duró dos meses y se llevó a cabo con tal éxito, que se considera esta conferencia como la **introductora del término Inteligencia Artificial** y, con él, una nueva área científica de conocimiento.

La época donde suceden **años de crítica y madurez** ocurrió en los difíciles años **70** y la **Etapa de Expansión** se produjo en los años **80.**

Fue John McCarthy quién definió **LISP** en los laboratorios de IA del MIT en 1958 siendo una gran contribución al campo de la IA.

Hofstadter.

Postuló que la inteligencia es la habilidad de responder flexiblemente a diferentes situaciones, saber aprovechar circunstancias fortuitas, dar sentido a mensajes ambiguos o contradictorios, encontrar similitudes entre situaciones diferentes y generar nuevos conceptos e ideas innovadoras. En ningún momento comenta sobre la rapidez y decía que no hay diferentes tipos de inteligencias.

Definición de Inteligencia Artificial

“El arte de construir máquinas capaces de hacer cosas que requerirían inteligencia si las hicieran los seres humanos”. (Minsky, 1986) - “El estudio de los cálculos que hacen posible percibir, razonar y actuar”. (Winston, 1992)

Tipos de inteligencias artificiales

- La IA **fuerte** pretende construir máquinas que sean **conscientes** (como una persona) de lo que hacen, que piensen como los humanos, de tal manera **que "La sala china" no pueda demostrar** que no es consciente de lo que hace.
- La IA **débil** pretende construir máquinas que **no** sean **conscientes** de lo que hacen, sino que **actúen como los humanos** en ciertas tareas, o sea máquinas **que pasen la prueba de Turing**. Se puede **comportar de manera inteligente,** pero, no por ello, de la misma manera que un ser humano.

El **efecto Flynn** es la **subida continua** de las puntuaciones de **Cociente Intelectual** a lo largo del tiempo, explica que hay aumento en el coeficiente intelectual, pero nos asegura que **no** es **simétrico**.

Entre las explicaciones que se han dado a este fenómeno, podemos encontrar una mejor nutrición y a las influencias médicas.

Deep Blue.

Supercomputadora desarrollada por **IBM** en **1997** para **jugar al ajedrez**. Fue la primera que venció a un campeón del mundo vigente, Gary Kaspárov. La definición de inteligencia es subjetiva y no podemos decir que, el hecho de que la IA ha ganado el juego de ajedrez, le hace más inteligente que su oponente.

Alan Turing.

La prueba de Alan Turing se ha desarrollado para detectar si es posible **distinguir entre una máquina y un ser humano**. A estos efectos, un ser humano habla simultáneamente con una máquina y otro ser humano, sin tener contacto visual con ninguno de los dos. Si no es posible distinguir entre hombre y máquina, es de suponer que la máquina tiene la misma capacidad intelectual que el ser humano. Durante la prueba, el ser humano no pregunta cosas especiales anteriormente definidas, por lo que, en este caso, lo que se quiere detectar no es una IA débil, sino una IA fuerte.

Searle.

La **sala china** de Searle es un **contraejemplo a la prueba de Turing**. Propone que, si Searle es encerrado en una sala con la única misión de traducir unos textos en chino, sin entender el idioma y con la única ayuda de un manual no se puede decir que el manual, que Searle y/o que la sala entienda el lenguaje chino, por tanto, un computador que traduce signos chinos no se puede decir que entienda chino ni que use la "inteligencia" para ello. **Maneja información sintáctica,** respuestas en base a una serie de reglas predefinidas **sin conciencia alguna de la propia acción**.

Conclusión.

Para Alan Turing si una maquina se comporta de manera inteligente se puede afirmar que dicha maquina posee IA débil pero, **para Searle,** aunque una maquina posea un comportamiento inteligente no se podrá decir que lo es, en tanto en cuanto, dicha maquina no posea conciencia de lo que está haciendo (IA fuerte).

Áreas de Aplicación

Problemas de percepción: Visión y habla, planificación, estrategias inteligentes, robótica, predicción financiera, aprendizaje, minería de datos, juegos, mundos virtuales, Internet, sistemas expertos.

Futuro de la IA

Orientado a abordar aquellas tareas que, ya sea por lo **incomodas, peligrosas o complicadas** que sean, conviene apoyarlas o delegarlas en sistemas inteligentes artificiales.

TEMA 2. Estrategias de búsqueda.

Sistemas de producción.

El proceso de búsqueda se puede realizar **explorando un árbol** (árbol de búsqueda) o, en general, un grafo (eliminando repeticiones de estados).

Propuesto por **POST** en 1943.

- **BH (Base de Hechos).** Conjunto de representaciones de uno o más estados por los que atraviesa el problema. Constituye la estructura de datos global.
- **RP (Reglas de Producción).** Conjunto de operadores para la transformación de los estados del problema, es decir, de la base de hechos. Cada regla tiene dos partes: **Precondiciones // Postcondiciones.**
- **EC (Estrategia de control).** Determina el conjunto de reglas aplicables mediante un proceso de *pattern-matching* y resuelve conflictos entre varias reglas a aplicar mediante el filtrado.

1. Determinar el conjunto de reglas aplicables y aplicar el filtrado.

2. Aplicar la regla seleccionada. La selección depende de la información de control.

3. Repetir hasta que se den las condiciones de terminación.

Caracterización del problema.

Identificando si se puede **descomponer el problema** ante el que nos encontramos, viendo si se pueden **ignorar** o al menos deshacer **pasos erróneos** hacia la solución, descubriendo si el **universo es predecible**, comprobando si la **bondad** de una solución es **relativa o absoluta y** si la **solución es un estado o un camino**. Además, ver el papel que desempeña **el conocimiento** a la hora de encontrar la solución.

- **Ignorables** -> Demostración de teoremas.
- **Recuperables** -> Podemos retroceder.
- **Irrecuperables** -> No se puede retroceder.

Problemas clásicos de búsqueda.

- **Jarra de agua:**

Jarra 1	Jarra 2	Regla a aplicar
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5
0	2	9
2	0	[illegible]

- **8-Puzzles:**

  La mejor heurística a la hora de encontrar solución al problema es, **la suma de las distancias** de las piezas a sus posiciones en el objetivo utilizando la **distancia Manhattan**.

- **Otros problemas:**
  - El problema del viajante de comercio.
  - El dilema del granjero.
  - El dilema de los misioneros.
  - Las torres de Hanoi.
  - El problema del puente.

Estrategias de búsqueda básica.

- **Ciclo de control básico** dentro de una estrategia de control:
  - **E1.** Exploración de la frontera.
  - **E2.** Cálculo de reglas aplicables.
  - **E3.** Resolución de conflictos.
  - **E4.** Aplicación de regla y memorización de estado.

- **Tipos de Estrategias:**

Las estrategias para considerar las podemos subdividir en:

- **Irrevocables.** Presenta la característica de que **no se permite la vuelta atrás**. Mantenemos una frontera unitaria.
- **Tentativas.** La búsqueda es multi o mono camino. Se mantienen **estados de vuelta atrás** por si el estado actual no llega a buen fin.

En todo momento se debe producir un **avance** y este debe ser **metódico y dirigido**.

Estrategia Irrevocable:

- Disponemos del suficiente conocimiento local.
- Tiene problemas de mesetas, máximos locales y crestas.
- Algoritmos voraces (no permiten la vuelta atrás).
- Las equivocaciones sólo alargan la búsqueda.
- Debemos especificar una función de evaluación *f()* que nos proporcione un mínimo (máximo) en el estado final.

Estrategia Tentativa:

Ejemplo: Una estrategia heurística para encontrar el camino a la meta en un laberinto, solo con esos datos.

Estrategias No informadas:

Son **ciegas** en el sentido de que el orden en el cual la búsqueda progresa no depende de la naturaleza de la solución que buscamos. La estrategia de búsqueda ciega es más **eficiente en pequeños problemas**.

- **Búsqueda en profundidad.** Variación del **backtracking**.
- **Búsqueda en anchura.** Mayor prioridad a los nodos de menor profundidad.
- **Coste uniforme.** Serán similar al de anchura cuando el coste de aplicación de cada regla sea unitario.

(*) Para la solución óptima **en grafos finitos** se emplea la búsqueda en **profundidad.**

(*) Puede que la búsqueda de coste uniforme se vuelva infinita, cuando el coste del nodo expandido sea cero y conduzca de nuevo al mismo estado.

Informadas:

Al contrario de las “ciegas”, las informadas sí que van a disponer de **información de lo prometedor que es un nodo**, para llegar desde él a la solución.

*Algoritmo de búsqueda A**

Los conceptos básicos de la búsqueda heurística son: **completitud, admisibilidad, dominación y optimalidad.**

Un algoritmo A1 es dominante sobre A2 si cada nodo expandido por A1 es también expandido por A2.

- **Búsqueda A*:**

A*: f*(n) = g*(n) + h*(n)

- **g*(n) = c(s, n).** Coste del camino de coste mínimo desde el nodo inicial s al nodo n. Estimada por g(n).
- **h*(n).** Coste del camino de coste mínimo de todos los caminos desde el nodo n a cualquier estado solución. Estimada por h(n).
- **f*(n).** Coste del camino de coste mínimo desde el nodo inicial hasta un nodo solución condicionado a pasar por n. Estimada por f(n).
- **C*.** Coste del camino de coste mínimo desde el nodo inicial a un nodo solución.

F*(n)=C* en cada nodo del camino óptimo.

Una **función heurística** h(n) se dice que es **admisible** cuando **garantiza** la obtención de un **camino de coste mínimo hasta un objetivo** y se cumple:

h(n) ≤ h*(n) ∀n

Cuanto más correctamente estimemos h(n), menos nodos de búsqueda generaremos.

(*) Si nuestra función heurística nos devuelve un valor superior a h*, para algún nodo, **no se puede garantizar que encontremos la solución óptima**.

A la hora de **hallar la heurística** para un problema, debemos tener en cuenta:

- Que tenga unas **buenas restricciones**, puesto que se podría quedar fuera la solución óptima.
- **Acercarla** lo máximo posible **a la heurística óptima** de ese problema, para no perder la solución óptima.

Problemas de camino mínimo.

Si una función heurística no varía (es constante o nula), no modifica la diferencia entre los valores de f(n) de cualquier nodo de búsqueda, por tanto, se buscará en los mismos que si no usamos una heurística.

Para implementar el algoritmo A* y obtener el camino óptimo utilizaremos la distancia de Manhattan, ya que no podemos movernos en diagonal, solo puede avanzar en 4 direcciones.

La **distancia de Manhattan** (h(n)) se utiliza para calcular el **camino más corto sin movimientos diagonales**, mientras que **para el 8-con tenemos diagonales**, por lo que podemos obtener un camino más corto en diagonal que si nos desplazamos sin diagonales.

Para un problema en el que nos podemos mover en las **8 direcciones** se **utilizará el cálculo de la diagonal:**

Heurística óptima: h1((x, y)) = |m-x| + |n-y|

Coste actual óptimo: h2((x,y)) = |x| + |y|

Heurística admisible: h3((x,y)) = sqrt((m-x)^2 + (n-y)^2)

Inconvenientes de mantener la admisibilidad.

Fuerza al algoritmo a **consumir mucho tiempo** en discriminar caminos cuyos costes no varían muy significativamente.

No es práctico para problemas de mayor envergadura.

En general, el nivel de información de las heurísticas permite encontrar antes la solución, pero tiene la desventaja de **requerir un mayor coste computacional** para su cálculo.

Relajación de la restricción de optimalidad.

Podemos alcanzar una **solución en menor tiempo** y, en algunos casos, **obtener la solución óptima.**

- Técnica de ajuste de pesos. El objetivo es definir una función *f()* ponderada, *fw()*, como alternativa a la utilizada en A*.

- Técnica de la admisibilidad- Ɛ. El objetivo es aumentar la velocidad de búsqueda, a costa de obtener una solución subóptima. A*Ɛ opera de forma idéntica al algoritmo A* salvo que selecciona aquel nodo de ListaFocal con menor valor de Hf(n).

- Algoritmo de estimación de coste de búsqueda A. ListaFocal es una sublista de ListaFrontera, que contiene solo nodos con f(n) menor al mejor valor de los f(n) de ListaFrontera por un factor.

(*) Lista Frontera en el algoritmo A*sirve para añadir y coger de ella los nodos a los que podemos acceder desde el nodo actual.

- Comparación de algoritmos. El algoritmo de ponderación dinámica es más sencillo, pero únicamente es aplicable a problemas donde se conoce la profundidad.

Utilizando la técnica de relajación de la restricción de optimalidad "Algoritmo de ponderación dinámica" **en los últimos niveles no utiliza ninguna búsqueda ni en anchura ni en profundidad.**

TEMA 3. Búsqueda en juegos.

Juegos como problemas de búsqueda.

- **Imposible generar todo el árbol de búsqueda**, solo se puede generar hasta un determinado nivel de profundidad.
- **Estado (N):** configuración del juego en un momento dado.
- **Árbol de juego.** Cada arista de ese árbol indica un posible movimiento.
- En cada nivel se van **alternando los jugadores**.
- **Factor de ramificación (B):** número de **posibles movimientos** que se pueden realizar.
- **Wolfgang Kempelen** crea el "ajedrecista mecánico" en el año **1760**.
- El inicio de la jugada queda definido por el análisis del árbol.

Estrategia exhaustiva: MiniMax.

- Genera todos los nodos del árbol hasta la profundidad deseada y evalúa cada nodo hoja.
- Asigna un valor al nodo raíz, el cual, siempre debe tener valor.
- Si la decisión la toma el jugador MIN se asocia, a ese nodo, el mínimo de los valores de sus hijos y el máximo, en caso de MAX.
- El método MiniMax funciona teniendo en cuenta el **mejor movimiento para ti suponiendo que el contrincante realiza el peor para ti**.
- Se nos podría presentar un problema aplicando la estrategia MiniMax, cuando necesitamos una **respuesta en muy poco tiempo**.

Juegos multijugador.

Minimax extendido a juegos de más de dos jugadores sustituyendo el valor de un nodo por un vector de valores (tantos valores como jugadores).

Estrategia de poda: α-β.

La poda alfa beta es una técnica de búsqueda que **reduce el número de nodos evaluados en un árbol de juego** por el MiniMax.

- **α** es el valor de la mejor opción hasta el momento a lo largo del camino para MAX, esto implicará, por lo tanto, la elección del **valor más alto**.
- **β** es el valor de la mejor opción hasta el momento a lo largo del camino para MIN, esto implicará por lo tanto la elección del **valor más bajo**.

El **valor MiniMax** de un nodo estará siempre acotado **α <= V(N) <= β.**

Al principio inicializamos α = -∞ y β = +∞, al no tener ninguna evidencia.

Esta búsqueda **alfa-beta** va actualizando el valor de los parámetros según se recorre el árbol. El método realizará la **poda** de las ramas restantes cuando el **valor actual** que se está examinando sea **peor que** el valor actual de **α o β para MAX o MIN, respectivamente**.

α = max[α, V (Nk, α, β)]	**β = min[β, V (Nk,α, β)]**

Uso de movimientos de libro.

- Imposible seleccionar un movimiento consultando la configuración actual del juego en un catálogo y extrayendo el movimiento correcto.
- Razonable para algunas partes de ciertos juegos. En **ajedrez**, tanto la secuencia de apertura como los finales están muy estudiados.
- El rendimiento del programa puede mejorarse si se le proporciona una **lista de movimientos** (**movimientos de libro**).
- **Se usa** el libro **en las aperturas y los finales** combinado con el procedimiento MiniMax para la parte central de la partida (Ciertas partes muy estudiadas).
- Conocimiento + Búsqueda.

Espera del reposo.

Busca evitar el efecto horizonte. Consiste en explorar nodos hasta que su valor no cambie de manera drástica después de explorar un nivel más, es decir, **el valor del nodo se estabilice** de un nivel al siguiente.

Técnica de bajada progresiva.

- **Restricciones de tiempo**: algoritmos presentados anteriormente no adecuados.
- Recorrer **nodos por niveles**.
- Al llegar la petición de jugada, devolver la solución del último nivel que se haya completado.

Poda heurística.

- Objetivo: **Reducir B** desarrollando únicamente los mejores movimientos de cada nivel.
- **g(N):** Función adicional de evaluación, de bajo coste y es una versión simplificada de f(N).
- Factor de ramificación: **Factor (Nodo) = Factor (Padre (Nodo)) – Rango (Nodo)**

Continuación heurística:

Intento de evitar el efecto horizonte provocado por la limitación en profundidad. Solo se puede tener conocimiento hasta la profundidad seleccionada.

1. Desarrollar en anchura hasta un determinado nivel.
2. Selecciona un subconjunto de nodos terminales para desarrollar búsquedas más profundas.
3. Selección dada por un conjunto de heurísticas directamente relacionadas con el juego.

TEMA 4. Búsqueda para CSP.

Problemas de satisfacción de restricciones (CSP).

Conjunto de variables definidas sobre dominios finitos y conjunto de restricciones definidas sobre subconjuntos de dichas variables.

Solución al problema es la relación n-aria **que satisface todas las restricciones del problema**. Dependiendo de los requerimientos del problema, hay que encontrar todas las soluciones o sólo una.

Redes de restricciones.

Un CSP se puede representar como un grafo de restricciones y esto puede usarse para simplificar el proceso de solución.

CSP binario.

Aquel en el que todas las **restricciones tienen** a los sumo **dos variables** respectivamente.

Ejemplos:

- Asignación de tareas para un robot.
- Coloreado de mapas.
- Generación de crucigramas.
- N-reinas.

Todo problema n-ario se puede formular como un problema binario.

Coloreado de mapas.

Datos incorrectos:

- Se tiene que resolver siempre utilizando el algoritmo AC3, ya que backtracking no nos garantiza que vaya a encontrar una solución.
- Se necesita el mismo número de colores que número de territorios fronterizos con el área que más territorios fronterizos tenga.

Generación de crucigramas.

Variables: Grupo de casillas para una palabra (slots).

Dominios: Palabras del diccionario con la longitud adecuada.

Restricciones: Misma letra en la intersección de dos palabras.

Características: CSP binario, discreto (dominios grandes).

N-reinas.

Variables: Reinas (Xi), donde la i es el número de la fila.

Dominios: Columnas posibles.

Restricciones: No colocar dos reinas en la misma columna ni en la misma diagonal.

Características: Dominios discretos y restricciones binarias.

Criptoaritmética.

Características: Dominios discretos y restricciones múltiples.

Restricciones temporales.

Variables: Sucesos.

Dominios: Intervalo temporal para cada suceso.

Restricciones: Distancia temporal permitida entre sucesos (relaciones temporales antes, después, solapado…).

Características: Dominios continuos y restricciones binarias.

Árbol de interpretaciones:

- Partimos de un **nodo raíz** que **supervisa el proceso**.
- **Cada nivel** corresponde a una **asignación** de valor para una característica de datos.
- **Cada nodo** identifica una **posibilidad de asignación**.
- La **solución** se construye de forma incremental de tal forma que **cada hoja es una interpretación**.

Métodos de resolución.

Búsqueda.

- **Generación y test.** Generar de forma sistemática y exhaustiva cada una de las posibles asignaciones a las variables y comprobar si satisfacen todas las restricciones. Hay que explorar el espacio definido por el producto cartesiano de los dominios de las variables, ya que se basa en expandir una a una todas las posibilidades del problema. Busca la solución mediante una expansión del árbol en anchura y es muy poco eficiente.

- **Backtracking.** Se trata de construir la solución de forma gradual, instanciando variables en el orden definido por la permutación dada.

- **Backjumping.** Parecido al BT, pero el retroceso no se hace a la variable instanciada anteriormente, sino a la variable más profunda que está en conflicto con la variable actual.

Eficiencia en problemas de gran tamaño: **Backjumping > Backtracking > Generación y test.**

Diferencias: Cuando encontramos un espacio de dominios vacío, Backtracking solo puede volver al nodo anterior, es decir, subir un nivel mientras que Backjumping puede saltar al nodo en conflicto.

❖ Explorar el espacio de estados hasta encontrar una solución, demostrar que no existe o agotar los recursos.

Inferencia.

Consistencia de arco, consistencia de caminos y K-consistencia.

❖ Deducir un problema equivalente que sea más fácil de resolver.

Algoritmos híbridos.

Forward Checking, Maintaining Arc Consistency y Heurísticas.

❖ Combinación de las aproximaciones anteriores. Incorporar métodos de inferencia sobre un esquema de búsqueda.

Backtracking.

Si no se puede extender: backtracking

- cronológico: se elimina la última decisión.
- no cronológico: se elimina una decisión anterior.

❖ Backtraking siempre da la solución óptima, siempre que el algoritmo con el que trabajemos devuelva todas las posibles soluciones.

Inconsistencias:

- **Trashing e inconsistencia de nodo:** Relacionado con las **restricciones unarias**. Sucede cuando un dominio contiene un valor que no satisface una restricción unaria.
- **Inconsistencia de arista:** Relacionado con las **restricciones binarias**. Sucede cuando existe una restricción binaria entre dos variables de tal forma que para un determinado valor de la primera variable no existe ninguna asignación posible para la segunda.
- **Dependencia de la ordenación:** El orden de selección de las variables es un factor crítico. Se han desarrollado diversas heurísticas de selección de variable y de valor.

Forward checking.

- Los valores de las variables futuras ,que son inconsistentes con la asignación actual, son temporalmente eliminados de sus dominios.
- Si el dominio de una variable futura se queda vacío, la instanciación de la variable actual se deshace y se prueba con un nuevo valor.

(*) Forward Checking no soluciona la dependencia de la ordenación del Backtraking.

Propagación de Restricciones.

- Transformar el problema en otro **más sencillo** sin inconsistencias de arco.
- Propiedad de consistencia de arista: Una arista dirigida c(ep) = <Vi, Vj> es consistente si, y sólo si, para todo valor asignable a Vi existe al menos un valor en Vj que satisface la restricción asociada a la arista.
- Obtendremos una, ninguna o varias soluciones.
- Un CSP puede transformarse en una red consistente mediante un algoritmo sencillo (**AC3**) que examina las aristas, eliminando los valores que causan inconsistencia del dominio de cada variable.

Algoritmo AC3.

Dos situaciones:

- Cuando un dominio queda vacío se queda inconsistente y sin solución.
- Si el grafo es consistente puede tener una solución o más.

(*) Tras la utilización del algoritmo AC3 hemos logrado eliminar todos los valores que causan inconsistencia del dominio de cada variable, usaremos backtracking únicamente cuando la eliminación de inconsistencias de lugar a más de una solución.

(*) En el algoritmo AC3 de búsqueda por CSP, la variable Q: contiene todas las restricciones binarias del problema en ambos sentidos.

TEMA 5. Sistemas Expertos Difusos.

Lógica Difusa.

Con lógica de **primer orden**/**multivaluada**:

- Es **monotónica**.
- Tiene **dificultad** de **representar** el **conocimiento** real.

Con lógica **difusa**:

- **Representación** del **conocimiento** de forma más **natural**.
- Intentan **imitar** la forma del **racionamiento** de los **humanos**.
- **Reconoce** valores que representa grados de **veracidad** o **falsedad**.
- Dice cuanto **porcentaje** de cada conjunto le pertenece a una variable.
- Intenta **disminuir** las **transacciones entre estados combinando reglas para decidir entre conjunto de estos aplicando** el **cumplimiento** de dichas **reglas** y **generando soluciones para cada una** de ellas. **Dependiendo** del **grado** de **cumplimiento se escoge** el **resultado** de **una de ellas.**

Desventajas:

- Tiene **múltiples definiciones** de operadores y reglas de inferencia difusa.
- Ante un problema que tiene solución mediante un modelo matemático, obtenemos **peores resultados**.

Conjuntos difusos.

En un conjunto difuso no se especifican los elementos que forman parte del conjunto dependiendo de si cumplen o no unas propiedades, sino que se especifica una función de pertenencia que indica si un elemento pertenece al conjunto dado.

La **función de pertenencia** se establece de una **manera arbitraria** (triangular, gaussiana, trapezoidal).

Operaciones entre conjuntos.

$$\mu_{A\cap B}(x) = \min(\mu_A(x), \mu_B(x))$$
$$\mu_{A\cup B}(x) = \max(\mu_A(x), \mu_B(x))$$
$$\mu_{\neg A}(x) = 1 - \mu_A(x)$$

El operador de conjunto difuso unión, *A ∪ B : f A∪B (x) = max[fA(x), fB (x)]*, **cumple la propiedad de Morgan:** *¬(A ∪ B) = ¬A ∩ ¬B.*

Por lo que, la siguiente expresión es correcta: **A U (¬B∩C) = (A U ¬B) ∩ (A U C)**

Variables lingüísticas.

- Es una variable cuyos valores son palabras o sentencias en un lenguaje natural o sintético.

Las siguientes variables son aptas como variables lingüísticas: Edad, distancia y temperatura.

La **suma** de los **factores de pertenencia** de un conjunto difuso para un determinado valor **puede ser cualquier valor**.

Al **enmarcar** una variable lingüística tendremos en cuenta tanto el **universo como** los **valores lingüísticos**.

Modificadores lingüísticos.

- Un modificador lingüístico permite modificar el significado de un conjunto difuso.

Por ejemplo, *u_poco(x) = x^y, y<0* es un modificador que aplica el significado muy poco a un conjunto.

Modificador lingüístico **Muy:** $\mu(x) = \mu(x)^2$

Ejemplos de modificadores lingüísticos: "no", "muy", "algo", "casi".

Sistemas Expertos Difusos.

- Los sistemas expertos son una rama de la IA que hace uso del conocimiento especializado para resolver problemas como un especialista humano y suelen darse situaciones de incertidumbre. Estos nacen por la necesidad de interpretar acciones en el mundo real.

Debe ser capaz de explicar paso a paso cómo se obtuvo la respuesta. Un sistema experto representa y usa conocimiento y puede operar con información incompleta.

El **sistema clásico** representa y usa **datos**, frente al **experto** que representa y usa **conocimiento**.

Existen varias formas para crear programas que actúen como **sistemas expertos**, los primeros y más utilizados son los sistemas **basados en reglas**:

- Se genera un consecuente para cada regla en función del grado de cumplimiento de cada una.
- Se obtiene una salida numérica a partir de todos los consecuentes obtenidos.
- Se comparar los valores numéricos de entrada al sistema con las funciones de pertenencia asociadas a los términos lingüísticos de la parte del antecedente de la regla asociada a esa entrada.

En los sistemas expertos, el conocimiento se representa mediante lógica (proposiciones y predicados).

Respecto a las **diferencias** entre un sistema experto y uno clásico, podemos afirmar que el sistema experto puede funcionar con pocas reglas.

Características: Alto desempeño (implica la capacidad de mantenerse activo y en funcionamiento durante largos periodos de tiempo, sin descanso), tiempo de respuesta adecuado, confiabilidad, comprensible, flexibilidad y que tenga una representación explicita del conocimiento.

Los términos **probabilidad** y **pertenencia** se diferencian en que el **conjunto** de **probabilidades** siempre suma uno y el de la pertenencia no necesariamente suma uno. No podemos establecer el valor de los conjuntos de pertenencia sin conocer su función correspondiente.

El **orden** a aplicar las reglas de un sistema difuso es: Fuzzyficación, aplicar operador fuzzy, implicador fuzzy, combinación de las reglas, defuzzyficación.

De que **partes** consta un sistema experto difuso: De una entrada de datos, una base de conocimientos, una fuzzyficacion, una toma de decisiones, una defuzzyficacion.

En la "**fuzzyficación**" es falso que los antecedentes de las reglas difusas no tienen por qué cumplirse siempre y que se disparan las reglas cuyo consecuente tiene un cierto grado de cumplimiento.

En la **toma de decisiones** de un Sistema Experto, las reglas se ejecutan de forma **paralela**.

Parámetros para establecer en el SE.

Se pueden **combinar varias reglas mediante** varios posibles métodos de **agregación**. La agregación en un sistema difuso es el proceso que genera el conjunto de salida a cierta variable.

Los parámetros por establecer en el SE son: el **AND/OR** a utilizar, el método de agregación para los conjuntos de variables a defuzzycar, el método de activación y el método de defuzzyficación.

En Defuzzyficación se obtiene una salida numérica a partir de todos los consecuentes obtenidos. Podemos utilizar el cálculo de centro de masas para la obtención del resultado. Ejemplo: **COG**.

operator **OR**		operator **AND**	
keyword for **Algorithm**	**Algorithm**	keyword for **Algorithm**	**Algorithm**
MAX	Max ($\mu_1(x)$, $\mu_2(x)$)	MIN	Min($\mu_1(x)$, $\mu_2(x)$)
ASUM	$\mu_1(x)+\mu_2(x)$ - $\mu_1(x)$ $\mu_2(x)$	PROD	$\mu_1(x)$ $\mu_2(x)$
BSUM	Min(1, $\mu_1(x)$ + $\mu_2(x)$)	BDIF	Max (0, $\mu_1(x)$ + $\mu_2(x)$ -1)

Operación booleana "AND" (x AND y): **minimum**(truth(x), truth(y))

FCL.

Lenguaje que solo tiene características propias de lógica difusa, no es un lenguaje de lógica difusa totalmente completo.

En Fuzzy Control Lenguaje (FCL) la **declaración** de las **variables de entrada** se hace en el apartado *VAR_INPUT*.

En cuanto a la sintaxis para definir sus reglas y sus algoritmos son **incorrectas**:

- RULE 1: THEN condition IF conclusion;
- ASUM: u_1(x) - u_2(x) + u_1(x) u_2(x)

TEMA 6. Árboles de Decisión.

Es interesante aprenderlos a partir de un **conjunto de vectores** (Clasificación, inducción…).

Establecen el **orden** para testar los atributos y conseguir la **clasificación** del vector de entrada. Para componer dicho orden se eligen primero aquellos atributos que mejor ganancia de información prometen, a efectos de descubrir la clase del vector de entrada.

Los árboles de decisión usan una **estrategia irrevocable** de descenso por gradiente (**método voraz**), lo que quiere decir que **poseen conocimiento local y buscan la mejor solución** a partir de esa información local.

Los arboles de decisión nos **sirven para determinar si una acción/operación se debe realizar o no**.

La **evaluación** de un árbol de decisión se realiza siempre **de derecha a izquierda**. Las variables cuyo valor se conoce antes de tomar la decisión han de aparecer a la izquierda del nodo en el desarrollo del árbol y las que no se conocen aparecerán a la derecha.

En un árbol de decisión **se seleccionan solo aquellos atributos que nos ofrecen mejor ganancia de información**.

Las **variables continuas** deben **categorizarse antes de** poder **ser usadas** en un árbol de decisión y deben ser **discretizadas**, aunque posteriormente no se seleccionen sus atributos durante el proceso de decisión.

Entropía y Ganancia de Información.

Entropía: Medida del **grado de incertidumbre asociado a una distribución de probabilidad**. Te da un grado de incertidumbre, en vez de uno de probabilidad.

En una **distribución uniforme**, todos los valores son igualmente probables Pi = 1/N y por tanto la entropía es máxima, lo cual indica **máxima incertidumbre (mínima información)**.

En una **distribución pico** en la que Pi = 1 y Pj = 0, para todo j≠i la entropía es mínima lo cual indica **mínima incertidumbre,** o sea, **máxima información.**

Al no ser un valor de probabilidad, cuanto **más cercano a 0** sea el valor**, menos incertidumbre** existe, pero **el valor puede ser mayor que 1**, existen más de 2 valores en la distribución de probables.

Tener la **mínima incertidumbre** (0) significa que tenemos la **máxima información** sobre ese caso, **y no se puede dar el caso contrario** bajo ningún motivo.

En un sistema con mismas probabilidades, la entropía es alta, ya que no sabemos cual puede ser el resultado.

Fórmula: $$E(S) = \sum_{i \in C} -p_i \log_2 p_i$$

Entropía condicionada: Entropía de la distribución de **Y condicionada a X**. Es una extensión del concepto de entropía de la información a procesos donde **intervienen varias variables aleatorias no necesariamente independientes**.

Si es una **entropía condicionada menor que E(Y**) indica que el **conocimiento de X mejora** la información que se dispone sobre Y.

Fórmula: $E(Y \mid X) = \sum_j Prob(X = v_j)\, E(Y \mid X = v_j)$

Si E(Y|X) = 0, se puede determinar el **valor** que tendrá **Y dado** el valor de **X**.

Ganancia de información: Es la medida de cuanto ayuda el **conocer** el valor de una **variable aleatoria X para conocer el verdadero valor de otra** Y. La alta ganancia implica que el atributo X permite reducir la incertidumbre de la clasificación del ejemplo de entrada.

Fórmula: $IG(Y \mid X) = E(Y) - E(Y \mid X)$

Algoritmo ID3.

Cada nodo del árbol representa cada uno de los **atributos** (Antigüedad, Moroso, Ingresos). Los **arcos** representan los valores que el nodo puede tomar (1200) y las **hojas** representan la decisión final que pueden ser positivos o negativos (Conceder, No conceder).

La **complejidad** de ID3 **crece linealmente** con el número de ejemplos de entrenamiento **y** crece **exponencialmente** con el número de atributos.

Se calcula la ganancia de información de todos los atributos y se escoge el de mayor resultado.

Es utilizado en los arboles de decisión, a partir de un conjunto de ejemplos genera unas reglas.

Consideración de atributos numéricos

ID3 **solo** trabaja con **atributos discretos**. Si se usan atributos continuos hay que descomponerlos en rangos. Para ello, se ordenan los ejemplos según el valor y se toman como puntos límite los puntos medios de aquellos en que se cambie de clase.

- **Atributos con un gran número de valores**

Se forman grupos pequeños de ejemplos que pueden ser **homogéneos** por casualidad.

Formula de la Ganancia Normalizada: $G_N(S,A) = \frac{G(S,A)}{\sum v_i \in V(A) - p_{v_i} \log_2 p_{v_i}}$

TEMA 7. Redes Bayesianas.

Probabilidad como media de incertidumbre.

- **Aproximación frecuencial:**

Utiliza los **eventos pasados** para **predecir los presentes**.

La probabilidad P de un evento a, P(a) se define por la frecuencia de a basada en las observaciones pasadas.

- **Aproximación bayesiana:**

Trata de razonar sobre creencias en condiciones de incertidumbre.

(*) Sí existir la consistencia interna y es la fiabilidad en sentido estricto.

- **Axiomas de la probabilidad** (hay 3):
  I. P(a) debe ser un nº entre [0,1]
  II. Si a es un evento cierto, entonces P(a)=1
  III. Si a y b son mutuamente exclusivos entonces P(a v b) = P(a) + P(b)

  Resultado: P(a + ¬a) = 1 (por el 2º axioma)

Sucesos independientes: P(A|B) = P(A)

Teorema de Bayes.

Constante de normalización: P(B).

Nos permite obtener la P(A|B) en términos de la P(B|A) siendo P(B) > 0.

Regla de Bayes

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} = \alpha \cdot P(B \mid A)P(A)$$

Redes Bayesianas.

Una red bayesiana es un **grafo acíclico dirigido** para representar dependencias entre variables y mostrar una descripción escueta de cualquier distribución de probabilidad conjunta completa.

Una red bayesiana permite especificar la distribución conjunta de un grupo de variables aleatorias.

Formada por:

- Un conjunto de variables aleatorias que forman los nodos de la red. Cada nodo X tendrá adjunta una distribución P(X|Padres(X)).
- Un conjunto de enlaces que determinan la influencia (dependencia) entre nodos. Si X se conecta con Y, se dice que X influencia a Y.

Su **finalidad principal** es calcular la distribución conjunta de las variables nodo.

Su distribución de probabilidad se puede expresar como $P(x1, ..., xn) = i \prod P(xi /pa(xi))$.

$P(x1, ..., xn) = i \prod P(xi /xi+1,...,xn)$

Interferencia media redes Bayesianas.

A partir de la distribución conjunta podemos contestar cualquier pregunta relativa a la red.

Tipos:

- **Interferencia Exacta**.

Funciona para todas las RR.BB. Las redes de una sola conexión son tratables utilizando inferencia exacta.

Tiene mucha complejidad y es posible contestar cualquier pregunta relativa a la red.

Regla de inferencia general: $P(B \mid C) = \alpha \cdot \sum_{D} P(B, D, C)$

- **Interferencia Casos Especiales.**

El modelo de Kim y Pearl: Método de inferencia para redes bayesianas, solo aplicable a un poliárbol.

Se basa en el paso de dos tipos de mensajes entre nodos.

Sirven para actualizar la credibilidad e introducir nuevas evidencias.

No existe más de un camino entre cada pareja de nodos.

- **Interferencia Aproximada.**

Son métodos estocásticos, basados en muestreos que simulan las distribuciones de probabilidad de la red.

Sobre la inferencia exacta en la aproximada, decimos que las redes con conexión múltiple son intratables utilizando la exacta.

Inferencia utilizando algoritmos de muestreo aleatorio (Monte Carlo).

El **método Montecarlo** se basa en la realización de muestreo de cada probabilidad en las sucesivas iteraciones, almacenando cada resultado, para obtener finalmente una solución basada en aproximaciones.

Los algoritmos de muestreo aleatorio son:

- **Muestreo Directo.** Para responder cualquier pregunta dentro de la red bayesiana, empleando el algoritmo de este modelo. Debemos contar las apariciones en *s[]* de las evidencias y después debemos dividir por suficientes muestras.
- **Muestreo por Rechazo.**
- **Muestreo por Gibbs (MCMC).**

TEMA 8. Aprendizaje Bayesiano.

Basado en el teorema de Bayes
Permite **combinar** los datos de **ejemplo con conocimiento** a priori.
Usado como clasificador puede obtener **probabilidades de pertenecer a cada clase**.
Posibilidad de construir representaciones más complejas

Aprendizaje automático.

Consiste en programar una computadora para que **mejore** en la realización de una **tarea** a partir de **datos** de ejemplo o de la **experiencia**.
Las situaciones o tareas donde el **algoritmo** que se requiere **debe adaptarse** a circunstancias particulares o los algoritmos difíciles de programar “a mano” son necesidades que **impulsan** el **aprendizaje automático**.

Ejemplos de áreas de aplicación.

Un ejemplo del área de aplicación es la **minería de datos** (Data mining).

Tipos de aprendizaje.

Existen 3 tipos: **supervisado**, **no supervisado** y por **refuerzo**.

- **Supervisado.**

  - Aprendemos a partir de **ejemplos conocidos** (etiquetados según su clave). Reconocimiento de **patrones.**
  - La regresión estima una **función continua**.
  - Conocemos la **clase** a la que pertenece cada ejemplo.
  - Aleatoriamente se parte el conjunto inicial de ejemplos en **dos grupos**. (Entrenamiento y validación)
  - El **conjunto de validación** se emplea para estimar el **error de generalización.**
  - Se entrena hasta **alcanzar el error de validación** mínimo.
  - El **error de validación** sirve para comprobar la capacidad de generalización del proceso y **evitar el sobre entrenamiento**.
  - Cuando los **conjuntos de entrenamiento no son suficientemente grandes** utilizamos la **técnica de validación cruzada**,
  - k-fold corss validation, que consiste en **coger un conjunto y lo divide en k conjuntos del mismo tamaño** y hace un conjunto para validación y los otros para entrenamiento, cambiándose los papeles para que todos sean validadores y de entrenamiento.

- **No Supervisado.**

  - Tiene un conjunto de datos que **se agrupan en clusters**.
  - No tenemos etiquetas de a qué pertenecen los datos.

- **Refuerzo.**
  - **No** parte de un conjunto de **datos de ejemplo**.
  - **No** se utiliza en la **minería** de datos.
  - Tenemos una medida de lo **bien o mal que está funcionando el algoritmo**, pero **no sabemos** exactamente **qué falla**.
  - Se utiliza en programación con un **robot** mecánico que juegue al **ping pong**.

Fundamentos.

- **Teorema de Bayes.**

Es utilizado para **detectar** si un correo es spam o no mediante **palabras guardadas** en un vocabulario.
Coste computacional **alto**. En el caso general es lineal con el número de hipótesis candidatas.
Basado en el teorema: $(P(h|D) = P(D|h)\ P(h)\ /P(D))$

- **MAP y ML.**

Permite conocer el máximo a posteriori MAP. $(hMAP = argmaxP(h|D))$
Necesidad de un **conocimiento** a **priori**. **Si no** se tiene este conocimiento estas probabilidades han de ser **estimadas**.
Un **conocimiento** a **priori** puede ser combinado con los datos observados para **determinar** la probabilidad de una **hipótesis**. Implica **máxima verosimilitud** debido a que **P(h) = cte**.

Clasificador bayesiano.

En el ejemplo el cual pertenece a la hipótesis está caracterizado como **tuplas de atributos**.
Las hipótesis son las **clases** a las que **puede pertenecer un ejemplo**.

Estimar las probabilidades.

Los **métodos bayesianos** pueden acomodar las hipótesis que hacen **predicciones** probabilísticas.
Suponemos que los **valores** de los atributos **son condicionalmente independientes** para una clase dada (Naive).

Simplificaciones (Naive).

Cuando dispongamos de **conjuntos de entrenamiento de tamaño medio o grande**.
Los **atributos** que describen a los ejemplos sean **independientes entre sí** con respecto al concepto que se pretende aprender.
No importa el orden de aparición de los **elementos**, sólo importa si estos elementos están presentes.
En caso de clasificar un texto **la posición** de las palabras en el texto **importa** o no **dependiendo del objetivo**.

Fases.

- **Fase 1:**

Aparece 1 en el numerador y |Voc| (significa **conjunto de palabras en X**) en el denominar **para evitar que la probabilidad salga 0** si la palabra no ha salido nunca. Se **añade** el **1** a nk (nk+1) **para que** ese valor **0 no influya** demasiado en el resultado final. (P(wk|cj) = (nk+1)/(n+|Voc|))

Calcula el producto de **probabilidades** de todos los atributos del **ejemplo** a clasificar **condicionado** a la **clase** que se está probando.

Durante la fase 1 tras tomar un conjunto de ejemplos debemos calcular la probabilidad a priori de cada clase:

P(cj) = nº de ejemplos etiquetados con cj / nº total de ejemplos
nº de ejemplos de la clase actual / nº total de ejemplos

- **Fase 2:**

En un clasificador bayesiano **dado un documento** x=w1,w2,...,wn, nos **quedamos** con las **posiciones** de palabras que están contenidas en Voc y **devolvemos** la estimación **map**.
Un **algoritmo** de **aprendizaje** es **consistente** si obtiene una **hipótesis** que no comete **ningún error** sobre los ejemplos de **entrenamiento**.

TEMA 9. Redes Neuronales.

Neuronas.

- Son dispositivos de "**todo o nada**".
- Las **salidas** que provoca pueden **afectar** al **resultado** final.
- Si en la neurona **biológica** las entradas son **dendritas**, en la **neurona computacional** las entradas son **número reales**.
- La **inicialización** de **pesos** en una red neuronal debe realizarse de forma **arbitraria**.
- El modelo computacional **se asemeja** a una neurona biológica.
- Una neurona puede decir si va a llover o no (**elección si o no**).
- En una computacional la integración es la **suma** ponderada(net) por los pesos sinápticos seguida de **una** función de **activación** f(**net**).
- Para que una neurona dispare salida es necesario **superar** un umbral.

Dendritas, nº reales; Axón, suma ponderada; **Soma, resultado**. Siendo las primeras de cada pareja las partes de una neurona biológica y la segunda las de una neurona computacional.

Neuronas artificiales (Red neuronal).

Se basan en el **aprendizaje supervisado**.
Converge cuando **el error de validación se mantiene bajo** y los ejemplos de entrenamiento **no** provocan cambios significativos en los pesos de la red.
Puede sufrir sobre entrenamiento y arrojar resultados erróneos debido al **exceso de flexibilidad o rigidez**.
El sobre entrenamiento se puede detectar cuando el error de **validación sube** después de haber alcanzado niveles mínimos. El tiempo de entrenamiento es lento, y el tiempo de respuesta una vez entrenada puede ser lento.

Multicapa.

- Con perceptrones multicapa si no se produce procesamiento nos encontramos en la **capa de entrada**.
- El error δ: **una neurona** de una capa intermedia **contribuye** en los δ de la **capa siguiente**.
- No usar entrenamiento multicapa puede provocar el problema de la no-separabilidad lineal.
- Para resolver **problemas de paridad** es preciso **incorporar una capa adicional** de neuronas.
- Se buscan **funciones derivables** con forma **similar** al escalón del **perceptrón** de **una** sola **capa.**
- Se puede aplicar el algoritmo de **entrenamiento multicapa** si la función de activación es **derivable**.

Interpretación geométrica.

El ajuste de hiperplanos se forma, dado **dos conjuntos** de ejemplos correspondientes a dos clases, buscaremos su separación por un hiperplano.
Un perceptrón puede aproximar cualquier función si **posee dos capas ocultas**.
La interpretación geométrica de la función de activación "umbral" es un **hiperplano**.

Backpropagation.

- Itera hasta que el error **baje** de un umbral.
- Es necesaria una **función derivable**.
- En la convergencia con valores **muy pequeños** hay **convergencia lenta**.
- Las fases son: **Entrada, integración y salida**. (Neuronas **biológicas y computacional)**

Mide lo que **contribuye cada neurona** al error obtenido: $\delta_k = (d_k - y_k) * f'(net_k)$

Aprendizaje estocástico (Gradiente).

La inicialización de los pesos es **arbitraria y aleatoria** y el problema de la inicialización en los descensos por gradiente **tiene** solución. (D**istintas** inicializaciones)
En una inicialización de red Arbitraria y Aleatoria, son los **mínimos locales**.
La solución aportada a dicho problema es **entrenar la red** desde las distintas inicializaciones.

Ajuste de la red.

El **overshooting** ocurre cuando nos **saltamos el máximo**.
En la **inicialización de red** se pretende **evitar** el **mínimo local**.
El problema XOR es un problema de **paridad**.

Regla Delta.

- **Permite ajustar** iterativamente el hiperplano/**plano**.
- Se asume que el **incremento** de los pesos es proporcional a la disparidad entre la **salida** observada y la salida deseada.
- Se utiliza en el aprendizaje **supervisado**.
- Si el peso utilizando la regla delta es demasiado grande se **reduce** el peso de las **aristas**.
- En el entrenamiento de perceptrones de una neurona si todos los ejemplos están bien etiquetados después de un número de iteraciones diremos que los **conjuntos** de ejemplos son **linealmente separables**.

En la interpretación geométrica es **falso** que los **problemas** con regiones de decisión más **complejas no requieren** distintas **estrategias** de **separación**.
En los perceptores multi-capa es **falso** que la "interpretación geométrica" se desarrolló sobre un algoritmo que **permite encontrar los pesos asociados a todas las neuronas**.

TEMA 10. Boosting y Adaboost.

Un poco de notación.

Están compuestos por: Patrones, Clases, Conjunto de entrenamiento**, función aprendida,** clasificador**.**

Combinar clasificadores "débiles".

- Los clasificadores débiles son moderadamente **precisos**, **simples** y **funcionan** al menos **mejor** que una clasificación aleatoria.
- Los **árboles** de **decisión** y las **redes neuronales** son **clasificadores débiles/inestables**.
- A más clasificadores que se añadan **no** se mejorará el aprendizaje de datos.
- Si **combinamos** un conjunto de **clasificadores** obtendremos **uno** más **fuerte** que los mismos por separado.
- Combinando **muchos** clasificadores **débiles** obtendremos un **clasificador fuerte**.

Boosting vs Bagging.

Un **clasificador** más **fuerte** se puede formar por el conjunto de Votos ponderados (**clasificadores**) y Muestreo ponderado (**ejemplos**). Los **clasificadores ponderados** se utilizan en los clasificadores con el **mismo peso** en el voto.

- **Boosting no** combina los clasificadores con el mismo peso en el voto.
- En boosting cada nuevo modelo está **influenciado** por el **comportamiento** de los **anteriores**.
- **Boosting** puede dañar performance en **datasets** ruidosos.
- El boosting se entrena hasta que consigue **clasificar bien** el máximo de ejemplos posibles.
- Boosting se lleva a cabo con adaboost y **escoge un valor** de confianza **αt**.
- Boosting concede **menor** peso a las **muestras** que están clasificados **correctamente** y el más **alto** a los **mal clasificados**.
- En boosting los **votos** son ponderados **unos mejores que otros**.
- En boosting los ejemplos más **cercanos** a la **frontera** de decisión reciben los pesos más **altos**.
- **Boosting** realiza un **muestreo** ponderado mientras que **Bagging no**.
- Los **votos ponderados** en boosting se realizan con clasificadores **débiles**.
- Boosting se usa a día de hoy en las cámaras (**detención de rostro**).
- Boosting es la técnica para **entrenar** varios **clasificadores débiles** para encontrar un clasificador mejor.

- **Bagging** ayuda a mejorar Redes neuronales y árboles de decisión.
- En Bagging los elementos del conjunto de entrenamiento clasificados por h(t) se pueden usar en h(t+1) y la hipótesis final no se selecciona el clasificador que mejor haya evaluado el conjunto de entrenamiento.

- **Diferencias entre Boosting y Bagging:** el **Boosting asigna pesos** a **cada registro** de entrenamiento y **Bagging** elige aleatoriamente los registros para **formar** los **subconjuntos**.
- Tanto boosting como bagging **mejoran clasificadores inestables** como por ejemplo las redes neuronales.
- **Boosting** pondera y da **más peso** a los ejemplos que **más cuestan** de **clasificar** y **bagging entrena un clasificador débil** con el subconjunto cogido T veces.

- Los modelos o clasificadores tienen **mismos pesos** en la formación de la **hipótesis final**.
- En el muestreo ponderado los ejemplos más **cercanos** a la **frontera** de **decisión** son los más **difíciles** de clasificar y recibirán los **pesos** más **altos**.

Adaboost.

- Es falso que "**i**" indexa **clasificadores** (**débiles**), mientras que "**t**" indexa **ejemplos**.
- Es un algoritmo utilizado para construir **clasificadores sólidos** utilizando la combinación lineal de clasificadores simples.
- Persigue **mantener** un peso en cada uno de los ejemplos de entrenamiento.
- Si obtenemos ϵt = 0 tenemos una **frontera perfecta**.
- Es un algoritmo adaptativo porque los subsecuentes **clasificadores** son **construidos** y **mejorados** en favor de los clasificadores **anteriores mal clasificados**.
- Pasos necesarios: **Calcular** error del **modelo** en el **set** de **entrenamiento** y **ponderar** todos los **sets** de entrenamiento de **igual** forma.
- **Boosting y Adaboost** están basados en **aprendizaje múltiple**.
- Ponderar el conjunto de clasificadores inicial de forma que en su conjunto podamos clasificar de forma correcta los ejemplos de nuestro entrenamiento.

Construyendo y usando Dt.

- El valor de confianza depende del error que se cometió en la **clasificación débil**.
- En Adaboost entrenamos un clasificador débil usando Dt para obtener ht.
- Cuando se actualiza la distribución D inicialmente, **cuando T>1** todos los **ejemplos** son **igualmente probables**.
- Usando Dt, **εt** es el **error** asociado a **ht**.
- El valor de αt surge de intentar optimizar el error asociado a ht, εt, y es ($\alpha t = 1/2 \ln (1 - \epsilon t) / \epsilon t$)
- **Primero entrenar un clasificador débil usando Dt** y obtener ht, escoger un valor de confianza αt y por último actualizar la distribución D.

TEMA 11. Percepción Visual Artificial.

Definiciones.

El modelo RGB supone **un problema** en el campo de sistemas inteligentes porque existen colores muy similares a simple vista que tienen una **representación RGB** muy **diferente** (alejada), lo cual puede suponer **un alto coste** computacional.

Cuando se **reduce la resolución** de una imagen, al perder píxeles promedia entre los **píxeles eliminados**.
En cuanto a los modelos de color **el tipo de imagen** que podemos obtener es una imagen **binaria**, valores 0 y 1.

Un método de **procesamiento robusto ante el ruido** de una imagen es bueno cuando genera los **mismos resultados** con o sin ruido. (**Google night mode 😆)**

El **error** que puede **degradar** la **imagen** suelen ser el **ruido**.
En **ampliación** de una imagen se **repiten píxeles**.

FALSO: Si ampliamos o reducimos una imagen el número de píxeles sigue siendo igual.

- **Convolución:**

El **nuevo** valor del **píxel** es el resultado de **sumar** la **multiplicación** de los **valores** de la máscara por los valores de los píxeles.

En el **proceso** de **captura** de una imagen mediante un **sensor** se produce una **discretización**.

Histograma.

- En histogramas podemos saber en una imagen qué número de **píxeles** comparten el **mismo valor**.
- Es **incorrecto** que la **ecualización** del histograma **consiste** en **comprimirlo** para mejorar el contraste.
- Si **aumentamos** el **valor** de todas las posiciones de la matriz de una imagen **aumentará** el **brillo** de la imagen.
- En una imagen con **poco contraste** podemos "**expandir**" el histograma.
- Si a una imagen se le **baja contraste y** se le **sube** el **brillo**, el histograma de la imagen **será más comprimido** y desplazado hacia la **derecha**.
- Para **aumentar** o **reducir** el **brillo** en una imagen tenemos que **aumentar** o **reducir** el **valor** de cada **píxel**.

La ecuación utilizada es:

- **Paso 1:** Se calcula el histograma de la imagen **h(k)**.

- **Paso 2:** Se calcula el histograma acumulado $Ha(k)$ = ∑$(desde\ j = 0\ hasta\ k)\ h(j)$.

- **Paso 3:** se produce un mapeo con la siguiente fórmula l'(x,y) = $(Ha(l(x,y)) - \boldsymbol{minv}) * (L-1)/(totalpix - \boldsymbol{minv})$

El proceso de binarización:

- Consiste en poner a **1** los píxeles que estén por **encima** de un **umbral** o entre dos umbrales y el **resto** a **0**.

Aristas (Edges).

- Son puntos de **alta derivada** en valor absoluto.
- Los tipos de **detectores** para **puntos** de **esquina** o **corners** están basados en "**edges**" o en "niveles de gris".
- En los filtros sobre ímagenes hay que **tener** en **cuenta el tamaño de la máscara** porque **puede tener** un **efecto no deseado** en el **resultado**, aunque hay veces que es lo que se pretende.

Detector de Canny.

Criterios (formalización):

- Buena **detección**: minimizar el número de falsos positivos y falsos negativos.
- Buena **localización**.
- **Respuesta única**.

Supuestos:

Ruido **gaussiano** y aristas tipo "**escalón**".

Resumen:

- En el **último paso** del trabajo el **algoritmo** realiza la unión de los píxeles, incorporando "**candidatos débiles**" que están 8-conectados a los píxeles "**probables**".
- El **filtro** que se usa para **suavizar** una imagen es el **Filtro Gaussiano**.
- El **valor** de **máximo** de la máscara de convolución aparece en el píxel central y disminuye hacia los extremos.
- Es el más efectivo a la hora de detectar **bordes** debido a su **eficacia**.
- Utilizando **trade-off** si **aumentamos sigma reducimos el ruido**, pero difuminamos los bordes y perdemos calidad en la localización.

Detector Nitzberg-Harris.

Para la construcción del detector de **Nitzberg-Harris** alguno de sus pasos son: **Calcular gradientes horizontal y vertical** para cada uno de los píxeles de la vecindad.

La **matriz A(x, y)** captura la estructura de intensidad de la vecindad local.

TEMA 12. Percepción automática.

En percepción automática en los modelos de color **el RGB presenta un problema importante** a la hora de segmentar por color, dos colores similares pueden aparecer lejos en el espacio de representación del modelo.

Transformada de Hough.

Motivación:

- Desarrollar técnicas que permitan **identificar primitivas** geométricas **sencillas** en una imagen.

Mecanismo:

- **Espacio paramétrico**. Los valores de los parámetros de la ecuación paramétrica **definen unívocamente** a **cada primitiva**. Por cada parámetro tenemos una dimensión en el espacio paramétrico y cada dimensión se "discretiza" en celdas.
- **Cada punto** de la imagen participa en un **proceso** de **votación**.

Análisis:

- **Ajuste de sensibilidad**: aunque es un factor de menor importancia, el ajusta del tamaño de celda y del número mínimo de votos es clave para evitar un elevado número de falsos positivos.

- Los puntos críticos son: **ineficiencia**, **ajuste de sensibilidad** y **solamente aplicable** a **primitivas sencillas**.

- La complejidad **espacial** y **temporal** de la ineficiencia son: O(N!) y O(N-1)! Respectivamente.

- En los puntos críticos el **método exhaustivo** solamente es aplicable a primitivas sencillas, haciéndose necesaria una discretización ligera para satisfacer los requerimientos de memoria.

FALSO: Emplear la transformada de Hough para determinar el reconocimiento de patrones geométricos inherentes a la imagen es suficiente para resolver problemas de iluminación.

Características SIFT.

SIFT trabaja de forma correcta sobre imágenes con **diferente iluminación**, ángulo de captura, etc.

Devuelve las **características** encontradas en la **imagen**.

Su orden es:

- Localización del punto. Escala. Orientación. Cálculo del descriptor. ->

Encontrar la posición de los puntos, calcular el descriptor.

Tiene dos pasos:

- **Buscar** los puntos característicos de una imagen y **calcular** sus descriptores.

FALSO: encontrar para cada punto el centroide más cercano.

En la localización:

- La **multiescala** se consigue con una **diferencia gaussiana** (DoG).
- Los **puntos relevantes** son: **máximos** y **mínimos**.

Segmentación de imágenes.

Es el proceso de extraer zonas de la imagen con el **mismo color**/**nivel** de **gris**/**textura** para identificarlas automáticamente.

Algoritmo de las K-medias.

- Es un método de agrupamiento heurístico con **número** de **clases conocido** (**K**).
- En su modo probabilístico, se puede aplicar a un **número reducido de distribuciones**.
- Termina cuando ya no hay cambios en las pertenencias o se han realizado el **máximo** de **iteraciones** fijadas como parámetro.
- **Asignar** a cada clúster el **centroide más cercano no** es un **paso** de K-medias.
- Las **medias distribuciones** (**clústeres**) se pueden encontrar en el algoritmo.
- Se **inicializa** asignando las **medias** de manera **aleatoria**.
- Es necesario conocer el número de distribuciones distintas existentes.
- Los **clústeres representan** las **medias de las distribuciones** a partir de las cuales podemos clasificar una imagen con un conocimiento previo.
- Se **buscan k puntos** (medias) y **es necesario indicar** explícitamente el valor de **k**.
- **No es un punto crítico**: Una **mala inicialización** puede llevar más tiempo.

FALSO: cuando inicializamos el método de las K-medias no podemos redistribuir las medias de manera uniforme.

FALSO: la transformada de Hough usa coordenadas polares para la identificación de primitivas geométricas sencillas de imagen. Por los que el algoritmo de las K-medias se puede usar también.

Segmentación basada en regiones

La **técnica** de **segmentación** basada en regiones en general trabaja mejor en **imágenes con ruido**.

En el **método** de **Crecimiento** de **regiones** podemos empezar por cualquier píxel de la imagen, pero es mejor lanzar varios puntos de partida "**semillas**".
El **objetivo** es encontrar **regiones** de la imagen **homogéneas** según algún criterio.

Conlleva una complejidad **mayor**, al tener que manejar **alguna estructura** de datos adicional.

Existen **dos maneras** de hacerlo:

Crecimiento de regiones: La manera en la que se empieza con regiones pequeñas y después se hacen crecer o se mezclan siguiendo un criterio de similaridad.
Se aplica un **detector** de **aristas**. Los puntos cuyo valor de magnitud de gradiente estén próximos a cero serán "**valles**", es decir, **puntos** dentro de **regiones**. Usaremos estos puntos como "**semillas**".

Partición de regiones: Empezamos con **regiones grandes** y las **vamos dividiendo**, usando un criterio de **homogeneidad**.