PREGUNTAS INTELIGENCIA ARTIFICAL SEGUNDO PARCIAL

1.- Componentes de un juego.

Un juego es cualquier situación de decisión con varios agentes, (jugadores) gobernada por un conjunto de reglas y con un resultado bien definido, caracterizada porque ninguno de los jugadores con su sola actuación puede determinar el resultado (interdependencia estratégica).

Formalmente, se podría definir como una clase de problemas de búsqueda con los componentes siguientes:

El **estado inicial**, que incluye la posición del tablero e identifica al jugador que mueve.

Una **función sucesor**, que devuelve una lista de pares (movimiento, estado), indicando un movimiento legal y el estado que resulta.

Un test **terminal**, que determina cuándo se termina el juego. A los estados donde el juego se ha terminado se les llaman estados terminales.

Una **función utilidad** (también llamada función objetivo o función de rentabilidad), que da un valor numérico a los estados terminales. En el ajedrez, el resultado es un triunfo, pérdida, o empate, con valores +1, -1 o 0. Algunos juegos tienen una variedad más amplia de resultados posibles: las rentabilidades en el *backgammon* se extienden desde +192 a -192**.**

Un juego se compone de las siguientes **características**:

- Número de jugadores.
- Juegos de información perfecta vs. Juegos de información imperfecta: Ajedrez vs. Póquer.
- Existencia de movimientos al azar.
- Orden de actuación de los jugadores.
- Existencia o no de pagos colaterales (equilibrio de Nash).
- Juegos de suma nula vs. Juegos de suma no nula.

En general, en un juego, cada jugador intenta conseguir el mayor beneficio para sus intereses. La solución de un juego es la determinación de una sucesión de actuaciones que indican a cada jugador qué resultado puede esperar y cómo alcanzarlo.

Por tanto, un juego puede plantearse como un problema de maximización, aunque finalmente y en muchos casos solo pueda llegarse a una satisfacción.

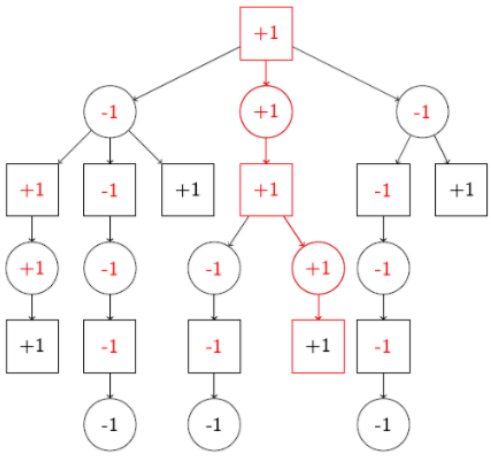
2.- ¿Qué es el factor de ramificación y cómo afecta a la complejidad de un juego? Describe en líneas generales el algoritmo minimax y el de la poda alfa-beta.

El **factor de ramificación** es el número medio de sucesores que puede tener un nodo de un grafo y es una de las tres componentes que expresan la complejidad de un grafo, y eleva la complejidad en espacio de un juego. Factores de ramificación altos hacen que los algoritmos que evalúan todas las ramas de todos los nodos, como los de búsqueda por fuerza bruta, sean más costosos computacionalmente hablando.

Minimax es un método de decisión para minimizar la pérdida máxima esperada en juegos con adversario **bipersonales**, de **suma nula** y con **información perfecta**. Minimax es un algoritmo recursivo, cuyo funcionamiento puede resumirse en cómo elegir el mejor movimiento para ti mismo suponiendo que tu contrincante escogerá el peor para ti.

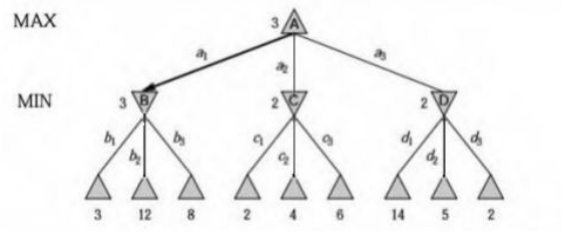
Primero nos centraremos en un **árbol completo,** es decir, los casos en los que se puede **representar todo el árbol del juego**, (juegos bipersonales, juegos con información perfecta, suma nula). Consideraremos juegos con dos jugadores, que llamaremos MAX y MIN. El algoritmo minimax busca la estrategia óptima, desde el punto de vista de un jugador, suponiendo que ambos juegan de forma imbatible.

Además, cada nodo hoja se etiqueta con 1, -1, 0, si es ganador para MAX, para MIN o empate y el objetivo es encontrar un conjunto de movimientos accesible que dé como ganador a MAX. Se propagan los valores de las jugadas terminales de las hojas hasta la raíz. Una “estrategia ganadora para MAX es un subárbol en el que todos los nodos terminales son ganadores. Incluso un juego simple como tic-tac-toe es demasiado complejo para dibujar el árbol de juegos entero. Los juegos complejos no se pueden resolver ya que es imposible la exploración total hasta la terminación, por lo que se busca un nuevo objetivo, el cual es encontrar una buena jugada inmediata. Por esta razón es tan importante la heurística en el proceso.



Por lo que ahora nos centraremos en casos donde **no se pueden representar todo el árbol del juego** (que son la mayoría). Considerando un **árbol de juegos**, la estrategia óptima puede determinarse examinando el valor minimax de cada nodo. Este valor es la heurística de la búsqueda (Valor-Minimax) de estar en el estado correspondiente asumiendo que ambos jugadores juegan óptimamente desde allí hasta el final del juego. El valor minimax de un nodo terminal es solamente su utilidad. Además, hay que tener en cuenta que MAX preferirá moverse a un estado de valor máximo, mientras que MIN prefiere un estado de valor mínimo. Entonces los pasos a seguir son:

- Primero tenemos en cuenta:
    - El valor V(J) de un nodo J de la frontera de búsqueda es igual al de su evaluación estática; en otro caso
    - Si J es un nodo MAX, entonces su valor V(J) es igual al máximo de los valores de sus nodos sucesores
    - Si J es un nodo MIN, entonces su valor V(J) es igual al mínimo de los valores de sus nodos sucesores.
- Para determinar el valor minimax, V(J) de un nodo J, hacer lo siguiente:
- Si J es un nodo terminal, devolver V(J)=f(J); en otro caso
- Para k=1, 2, ..., b, hacer:
    - Generar J_k, el k-ésimo sucesor de J
    - Calcular $V(J_k)$
    - Si k=1, hacer $AV(J) \leftarrow V(J_1)$; en otro caso, para k>=2,
    - hacer $AV(J) \leftarrow \max \{AV(J), V(J_k)\}$ si J es un nodo MAX o
    - hacer $AV(J) \leftarrow \min \{AV(J), V(J_k)\}$ si J es un nodo MIN
- Devolver V(J)=AV(J)



Por ejemplo, en el árbol de arriba, el algoritmo primero va hacia abajo a los tres nodos izquierdos, y utiliza la función heurística para descubrir que sus valores son 3, 12 y 8 respectivamente. Entonces toma el mínimo de estos valores, 3, y lo devuelve como el valor del nodo B. Un proceso similar devuelve hacia arriba el valor de 2 para C y 2 para D. Finalmente, tomamos el máximo de 3, 2 y 2 para conseguir el valor de 3 para el nodo de raíz. Por tanto, podemos ver cómo el algoritmo minimax realiza una exploración primero en profundidad completa del árbol de juegos.

$$\text{VALOR-MINIMAX}(n) = \begin{cases} \text{UTILIDAD}(n) & \text{si } n \text{ es un estado terminal} \\ \max_{s \in \text{Sucesores}(n)} \text{VALOR-MINIMAX}(s) & \text{si } n \text{ es un estado MAX} \\ \min_{s \in \text{Sucesores}(n)} \text{VALOR-MINIMAX}(s) & \text{si } n \text{ es un estado MIN} \end{cases}$$

El algoritmo poda alfa-beta se basa en el uso recursivo del valor minimax como el algoritmo anterior, pero se apoya en dos parámetros: alfa y beta para reducir la complejidad espacial del grafo podando las ramas que son irrelevantes para la solución.

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

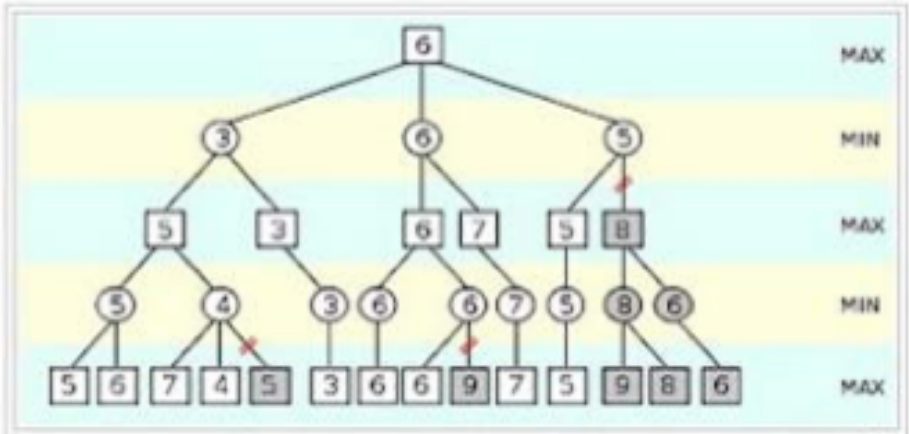
La búsqueda alfa-beta actualiza el valor de α y β según se va recorriendo el árbol y termina la recursión cuando encuentra un nodo peor que el actual valor α o β correspondiente. Al principio del proceso de búsqueda alfa toma el valor -infinito, mientras que beta toma el valor +infinito. Desde un nodo MAX, iremos actualizando el valor de alfa a medida que vayamos explorando cada arco que cuelgue de ese nodo.

Dicha actualización consiste en comparar el valor obtenido por cada arco con el valor actual de alfa. Si el valor obtenido es mayor, ése será el nuevo valor de alfa. Por tanto, alfa almacena el mejor valor que hemos encontrado desde el nodo en que estamos. Sólo nos interesará proseguir la búsqueda desde ese nodo si podemos mejorar alfa. En caso de que estemos en un nodo MIN, iremos actualizando el valor de beta a medida que vayamos explorando cada arco que cuelgue de ese nodo. Dicha actualización consiste en comparar el valor obtenido por cada arco con el valor actual de beta. Si el valor obtenido es menor, ése será el nuevo valor de beta. Por tanto, beta almacena el peor valor (para MAX) encontrado por MIN hasta el momento. Sólo interesará que la búsqueda prosiga si MIN pudiera reducir beta.

El intervalo (alfa, beta) contiene los valores que se puede seguir consiguiendo en el proceso de búsqueda. Cuando alfa crece tanto que sobrepasa a beta o cuando beta disminuye tanto que se hace menor que alfa, se puede hacer una poda y no seguir explorando el subárbol que cuelga desde el nodo en que nos encontramos, prosiguiendo la búsqueda en profundidad desde su nodo padre.

Algoritmo ALFA-BETA. Para calcular el valor V(J, alfa, beta), hacer lo siguiente:

- Si J es un nodo terminal, devolver V(J)=f(J). En otro caso, sean J_1, ..., J_k, ..., J_b los sucesores de J. Hacer k ← 1 y, si J es un nodo MAX ir al paso 2; si J es un nodo MIN ir al paso 5.
- Hacer alfa ← max (alfa, V(J_k, alfa, beta)).
- Si alfa >= beta devolver beta; si no, continuar
- Si k=b, devolver alfa; si no, hacer k ← k+1 y volver al paso 2.
- Hacer beta ← min (beta, V(J_k, alfa, beta)).
- Si beta <= alfa devolver alfa; si no, continuar
- Si k=b, devolver beta; si no, hacer k ← k+1 y volver al paso 5.



La figura muestra un ejemplo de la poda alfa-beta. Cada nivel representa la jugada de los jugadores MAX y MIN, que tendrán que definir un valor α o β respectivamente.

3.- ¿Que problemas plantea el cálculo de predicados en la resolución de problemas de IA?

Problemas semánticos

1. Es imposible de expresar todo en formulas heurísticas (ordenando los predicados o reglas frecuentes), metaconocimiento, jerarquía (subconjunto) y herencia (cumple las propiedades básicas), igualdad (ligado con el razonamiento acerca del predicado) y sentido común (relacionado con las heurísticas, metaconocimiento).
2. Razonamiento temporal: Es un problema la variable de tiempo, se maneja de forma poco satisfactoria. El paso del tiempo provoca cambios y el cambio en el mundo, supone que un predicado que era verdadero se vuelve falso y uno que era falso se vuelve verdadero.
3. Razonamiento acerca de predicados. No podemos introducir información incompleta o imprecisa.
4. Información Incompleta (Faltan datos o ES VERDAD o ES FALSA) y/o imprecisa (vaguedad o imprecisión, probabilidad o desconocimiento o priori, Aplicando el principio de Laplace: Da la misma probabilidad a todo). Por ejemplo, hay diferentes tipos tonos de colores por lo tanto ni es cierto ni falso.
5. Excepción: Relacionado con la herencia y la jerarquía. Existe una forma que se denomina la circunscripción, la cual trata de reducir los predicados a los ámbitos de mi alrededor.
6. Monotonía: impide la revisión de las demostraciones al actualizar las Bases de Conocimiento.

Problemas computacionales

1. Consistencia: Solidez (que tenga la garantía de la verdad absoluta). Lo que se demuestra como verdadera lo es realmente. Se suele demostrar por refutación por resolución.
2. Completitud: (Solo se da en la refutación por resolución), si una cosa es verdadera puede demostrarse, pero si no la puedo demostrar no puedo decir que sea falsa. Semidecidible.
3. Complejidad computacional: (tratabilidad). No tienen un algoritmo en tiempo polinomial. Es exponencial, conforme el número de axiomas aumenta la complejidad en tiempo y en espacio de razonamiento aumenta exponencialmente con el número de axiomas.

4.- ¿Modelos de conocimiento heredable ¿Qué tipo de conocimiento organizan las redes semánticas? Describir en líneas generales el concepto de "frame".

Los modelos de conocimiento heredable son estructuras jerárquicas que permiten representar el conocimiento de manera incremental de manera que facilitan la representación del conocimiento humano. Se obtiene relacionando clases con otras clases de carácter más general, lo que permite reutilizar el conocimiento general en las nuevas clases y heredar de manera total o parcial el conjunto de atributos que tienen definidos. Dentro de los modelos de conocimiento heredable se encuentran las redes asociativas y los marcos (frames).

REDES ASOCIATIVAS:

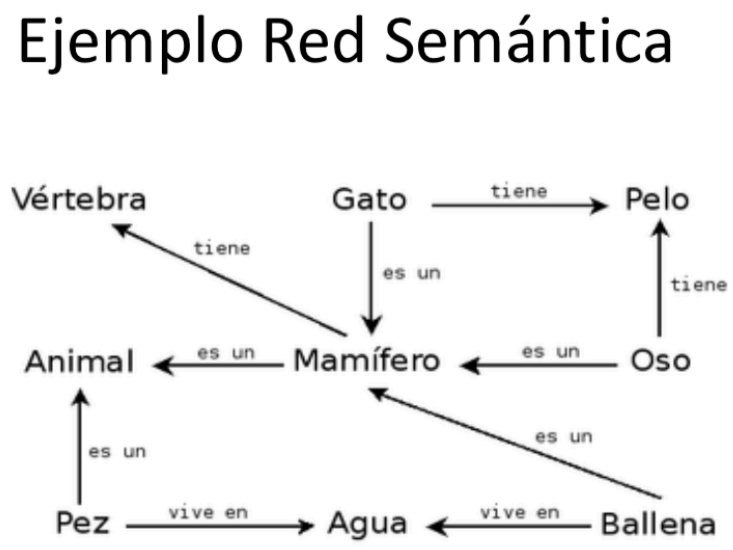
Cada nodo representa un concepto (o una proposición) y los enlaces corresponden a relaciones (inclusión, pertenencia, causalidad) o a categorías gramaticales (verbo principal, sujeto, objeto, complementos, etc). Entre ellas se conocen:

- Redes semánticas: son las destinadas a representar o a comprender el lenguaje natural.
- Redes de clasificación: es exactamente lo que su nombre indica, una clasificación de objetos o conceptos con sus características propias (herencia y demás).
- Redes causales: son las que llevan asociadas, junto a sus nodos que representan variables, una relación de influencia, representada por los enlaces.

Las redes semánticas son una forma de lógica para representar el lenguaje natural y la notación que proporcionan para cierta clase de sentencias es a menudo más conveniente, pero si se deja de lado la <<interfaz humana>>, los conceptos base (objetos, relaciones, cuantificación, etc.) son los mismos.

Existen diversas variantes, pero todas son capaces de representar objetos individuales, categorías de objetos y relaciones entre objetos. Se representa como un grafo dirigido etiquetado constituido por:

- Nodos: representan conceptos (un objeto individual o una clase de objetos).
- Arcos: representan relaciones binarias entre los conceptos.



MARCOS:

Un marco es una estructura de datos para representar una situación estereotipada, como encontrarse en un cierto tipo de sala de estar o asistir a un cumpleaños infantil.

Cada marco se caracteriza por un conjunto de campos o slots que se asocian en general a atributos, y que en conjunto sirven para identificar los marcos.

Los marcos están especialmente concebidos para tareas de reconocimiento: la información recibida hace que se activen unos marcos y esta a su vez provoca la activación de otros marcos conectados con los primeros, dando lugar así a una red de activación, cuyo objetivo es predecir y explicar la información que se va a encontrar en esa situación. Este reconocimiento suele denominarse herencia o más generalmente reconocimiento descendente.

Ejemplo Marcos:

- FRAME: pájaro
    - is_a: animal
    - forma_moverse: volar
    - activo_durante: día
- FRAME: pingüino
    - is_a: pájaro
    - color: blanco_y_negro
    - forma_moverse: andar
    - activo_durante: noche
    - tamaño: mediano
- FRAME: pepe
    - is_a: pingüino

La información (propiedades) específica al concepto representado por un marco es representado mediante atributos o slots. Los atributos ofrecen un medio de representar las propiedades de objetos individuales o clases de objetos.

Una faceta es considerada como una propiedad asociada a un atributo:

- faceta valor: es la más común y referencia el valor real del atributo.
- faceta valor por defecto: denota el valor inicial del atributo en caso de que no se
- especifique lo contrario.
- faceta tipo valor: especifica el tipo de datos del valor del atributo.
- faceta cardinalidad: especifica si se trata de un atributo uni o multi-valuado.
- faceta máxima cardinalidad: solo es válida para atributos multivaluados y especifica
- el máximo número de valores asociados al atributo.
- facetas demonio: permiten la integración de conocimiento declarativo y procedural.
- Un demonio o valor activo es un procedimiento que es invocado en un momento
- determinado durante la manipulación del atributo donde ha sido especificado.
- faceta tipo atributo: especifica si se trata de un atributo heredable o no heredable.
- faceta herencia: especifica el tipo de herencia del atributo.

5.- Estructura y componentes de un sistema experto.

Un sistema experto incorpora conocimiento sobre ámbitos específicos del conocimiento humano, tales como la medicina o los negocios. Los principales componentes del sistema son la base de conocimiento, el motor de inferencia y la base de hechos.

La base de conocimiento está compuesta por hechos y reglas (obtenidas de los expertos) del cálculo de predicados acerca del ámbito de la aplicación (aunque también podría componerse de conocimiento representado por alguna variante sintáctica del cálculo de predicados).

El motor de inferencia se compone de todos los procesos que manipulan la base de conocimiento para deducir la información perdida por el usuario; por ejemplo, resolución, encadenamiento hacia atrás o hacia delante (en algunos casos, la base de conocimiento y los mecanismos de inferencia pueden estar estrechamente relacionados).

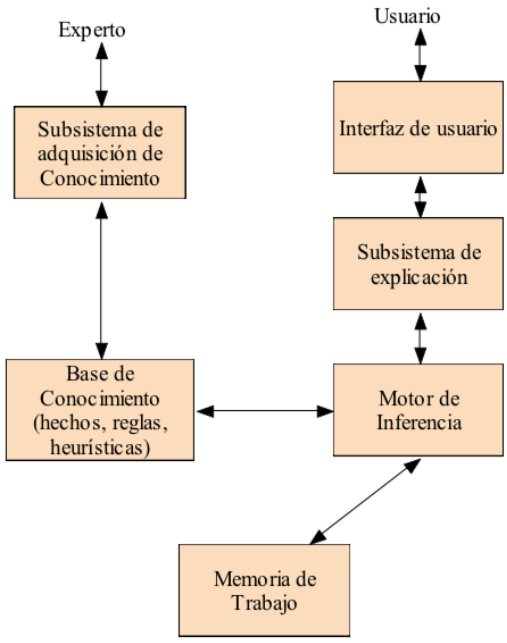
La Base de Hechos, también conocida como memoria de trabajo o base de datos global, es la que contiene los datos de partida y los criterios de parada, la misma se va actualizando durante la ejecución del sistema.

Además de los componentes anteriores los sistemas expertos necesitan interactuar con el usuario y con el experto.

Interfaz con el usuario, es la que facilita el diálogo con el usuario, permite hacerle preguntas al sistema e incluso obtener conocimientos análogos a los del experto. Las explicaciones pueden ser obtenidas de la base de hechos donde se almacenan los pasos para llegar a la solución.

Interfaz con el experto, permite al experto consultar los conocimientos almacenados en la base de conocimientos y da la posibilidad de incluir nuevos conocimientos. Su objetivo es que el experto pueda introducir directamente sus conocimientos en la máquina sin necesidad de ver al ingeniero que desarrolló el sistema.

La memoria de trabajo contiene la información relevante que el Motor de Inferencia está usando para razonar las respuestas para el usuario.



Los métodos para extraer el conocimiento se estudian en el área de aprendizaje automático dentro de la IA (extracción de reglas o aprendizaje de reglas).

Además, normalmente, la percepción del mundo por parte de un agente no es perfecta.

Del mismo modo, es posible que una regla no sea aplicable siempre (aunque sí en un gran número de casos). Este hecho no permite que la regla sea admitida en un sistema de cálculo proposicional o de predicados, dado que daría lugar a sistemas inconsistentes.

Se hace necesario establecer mecanismos para tratar con incertidumbre.

6.- Paradigmas de Aprendizaje Automático.

El aprendizaje al igual que la inteligencia es un concepto difícil de definir porque involucra a un amplio abanico de procesos y mecanismos, sin embargo, es una capacidad esencial de la inteligencia humana. De ahí su importancia para la IA, por razones teóricas o científicas (manifestación de inteligencia) y por razones de eficacia técnica, especialmente cuando existen patrones o conocimiento difíciles de describir o descubrir. En el campo del aprendizaje automático encontramos, principalmente, los siguientes paradigmas de aprendizaje:

- **Aprendizaje memorístico**: los datos se almacenan sin tratar de comprenderlos o de inferirlos a partir de otros ya conocidos y de forma repetitiva. Por ejemplo, si un sistema recibe, ante una determinada acción en cierta circunstancia, una respuesta desventajosa, puede 'memorizar' que esa acción da malos resultados y, si esos malos resultados se repiten en el tiempo, concluir (aprender) que esa no es una acción adecuada para dicha situación.
- **Aprendizaje deductivo**: es la obtención de nuevo conocimiento a partir de conocimientos que ya se poseen. También es el razonamiento artificial (dar una explicación a un evento observado a partir de conocimientos previos y crear así nuevo conocimiento).
- **Aprendizaje analítico**: Basado en explicaciones: Construir una explicación para cada ejemplo en relación con un concepto dado y generalizar la explicación de modo que pueda emplearse en el futuro.
- **Aprendizaje analógico**: se buscan soluciones a problemas nuevos encontrando similitudes con problemas ya conocidos y adaptando sus soluciones. Se basa en la idea de que, si dos situaciones son similares en algún aspecto, entonces también pueden serlo en otros.
- **Aprendizaje inductivo**: es el paradigma más ampliamente estudiado dentro del aprendizaje. Se trata de aprender un concepto o una clasificación a partir de ejemplos y contraejemplos. Se formulan hipótesis mediante la búsqueda de regularidades en unos ejemplos de 'entrenamiento' observados previamente (ejemplos) y se aceptan o rechazan dichas hipótesis con la aparición de nuevos ejemplos.
- **Tipos de aprendizaje según el conocimiento utilizado**. Uno de los puntos clave para el aprendizaje es el tipo de realimentación disponible en el proceso:
    - Aprendizaje Supervisado: Para cada entrada, se dispone de un profesor/supervisor que proporciona una salida deseada, ya sea una clase o un valor a aproximar (clasificación vs regresión).
    - Aprendizaje no supervisado: No se dispone de una salida deseada cada entrada, sino que se busca agrupar/clasificar los datos en función de ciertas características (medida de distancia)
    - Aprendizaje por refuerzo: Se aprende a partir de la información obtenida al realizar procesos de ensayo error en los que se obtienen "señales" de beneficio/coste.

El aprendizaje inductivo combinado con el analítico es uno de los métodos más potentes.

7.- Describir el problema del ruido y el del sobreajuste en aprendizaje automático.

Uno de los problemas a los que se enfrentan los sistemas de aprendizaje, y que provocan el sobreajuste, es cuando los ejemplos de entrenamiento contienen ruido:

- Valores de atributos erróneos, subjetivos
- Clasificación equivocada
- Valores desconocidos

Si hay dos o más ejemplos con la misma descripción (en términos de los atributos), pero diferentes clasificaciones, el algoritmo APRENDIZAJE-ÁRBOL-DECISIÓN fallará en la búsqueda de un árbol de decisión que sea consistente con todos los ejemplos.

Esto ocurre a la hora de elegir atributos para test en la toma de decisiones. Si no quedan atributos, pero sí ejemplos positivos y negativos, tenemos un problema, ya que los ejemplos que quedan tienen exactamente la misma descripción, pero clasificaciones diferentes. Esto sucede cuando alguno de los datos es incorrecto; diremos que hay ruido en los datos.

También ocurre cuando los atributos no proporcionan suficiente información para describir completamente la situación, o cuando el dominio no es determinista. Una forma sencilla de resolver el problema es utilizar el voto de la mayoría. Si no quedan atributos, pero sí ejemplos positivos y negativos, tenemos un problema, ya que los ejemplos que quedan tienen exactamente la misma descripción, pero clasificaciones diferentes. Esto sucede cuando alguno de los datos es incorrecto; diremos que hay ruido en los datos.

También ocurre cuando los atributos no proporcionan suficiente información para describir completamente la situación, o cuando el dominio no es determinista. Una forma sencilla de resolver el problema es utilizar el voto de la mayoría. La solución consiste en tener en cada nodo hoja la clasificación de la mayoría de su conjunto de ejemplos, si se requiere una hipótesis determinista.

En general, existen dos métodos para manejar ruido (basados en la condición de terminación):

- Pruning (o pre-pruning): Cambiar el criterio de paro del árbol de decisión para “podar” ramas.
- Post-pruning: “Podar” ramas una vez construido el árbol.

Estas operaciones de «poda» y de aumento de la tolerancia al ruido nos ayudarían a minimizar el riesgo de sobreajuste.

El sobreajuste es el efecto que se produce al sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado. El algoritmo de aprendizaje debe alcanzar un estado en el que será capaz de predecir el resultado en otros casos a partir de lo aprendido con los datos de entrenamiento, generalizando para poder resolver situaciones distintas a las acaecidas durante el entrenamiento. Sin embargo, cuando un sistema se entrena demasiado (se sobreentrena) o se entrena con datos extraños, el algoritmo de aprendizaje puede quedar ajustado a unas características muy específicas de los datos de entrenamiento que no tienen relación con la función objetivo.

Se dice que una hipótesis h se sobre ajusta al conjunto de entrenamiento si existe alguna otra hipótesis h' tal que el error de h es menor que el de h' sobre el conjunto de entrenamiento, pero es mayor sobre la distribución completa de ejemplos del problema (entrenamiento + test). Por lo que es preferible una hipótesis que se ajuste regularmente a los datos de

entrenamiento y a los de test, que una que se ajuste muy bien a entrenamiento, pero muy mal a los test.

Para afrontar este problema se usa una la técnica poda del árbol de decisión. La poda funciona impidiendo divisiones recursivas sobre atributos que no son claramente relevantes, incluso cuando los datos en ese nodo del árbol no estén clasificados de forma uniforme.

8.- ¿Qué son y como se construyen los arboles de decisión?

Un árbol de decisión es un modelo de predicción muy similar a los sistemas de predicción basados en reglas, que sirve para representar y categorizar una serie de condiciones que ocurren de forma sucesiva para la resolución de un problema.

Toma como entrada un objeto o una situación descrita a través de un conjunto de atributos y devuelve una "decisión", el valor previsto de la salida dada la entrada. Los atributos de entrada pueden ser continuos o discretos, a su vez el valor de salida puede ser continuo o discreto.

Un árbol de decisiones desarrolla una secuencia de test para poder alcanzar una decisión. Cada nodo interno del árbol corresponde con un test sobre el valor de una de las propiedades, y las ramas que salen del nodo están etiquetadas con los posibles valores de dicha propiedad. Cada nodo hoja del árbol representa el valor que ha de ser devuelto si dicho nodo hoja es alcanzado.

Para construir los árboles de decisiones se deben coger ejemplos positivos donde un predicado meta es verdadero y ejemplos negativos en los que el mismo predicado meta es falso. El conjunto de ejemplos completo se denomina conjunto de entrenamiento.

El problema de encontrar un árbol de decisión que sea consistente con el conjunto de entrenamiento puede parecer difícil, pero, de hecho, existe una solución trivial. El problema del árbol trivial es que memoriza exactamente las observaciones y no extrae ningún patrón a partir de los ejemplos, por lo tanto, no se puede esperar que sea capaz de extrapolar a ejemplos que no le han sido proporcionados. La manera de óptima de inferir el árbol compatible con todas las instancias es la navaja de Ockham pero es inviable computacionalmente. Se recurre a una solución pseudo-óptima, seleccionando el atributo en cada nivel del árbol en función de la calidad de la división que produce.

Para la elección de los atributos del test se debe elegir primero los atributos que proporcionen una clasificación lo más exacta posible de los ejemplos y así minimizar la profundidad del árbol final. Un atributo perfecto divide los ejemplos en conjuntos que contienen solo ejemplos positivos o negativos. Para saber si un atributo es bastante adecuado, la medida debe tener su valor máximo cuando el atributo es perfecto, y su valor mínimo cuando el atributo es inadecuado. Una medida adecuada es la ganancia de información proporcionada por el atributo, donde se utiliza el término de información en su sentido matemático.

Ejemplo

Temperatura	Nivel de vibraciones	Horas de funcionamiento	Meses desde revisión	Probabilidad de fallo
ALTA	ALTO	< 1000	> 1 MES	fallará
BAJA	BAJO	< 1000	< 1 MES	no fallará
ALTA	BAJO	>1000	> 1 MES	no fallará
ALTA	BAJO	< 1000	> 1 MES	no fallará
BAJA	ALTO	< 1000	> 1 MES	no fallará
BAJA	ALTO	>1000	> 1 MES	fallará
ALTA	ALTO	< 1000	< 1 MES	fallará