Preguntas segundo parcial IA Curso 18/19

1. Componentes de un juego.

Un juego es un entorno compuesto por varios agentes (llamados **jugadores**) y un **conjunto de reglas** que pueden usar los jugadores para generar **estrategias** con el fin de llevarles a obtener un cierto **beneficio**.

En un juego también podemos encontrar los siguientes componentes:

- El número de jugadores y el orden de actuación de los mismos
- La existencia o no de movimientos de **azar** que hace que un juego sea o no **determinista**
- La **información** disponible para los jugadores (**perfecta o imperfecta**), cuando hablamos de información perfecta hablamos de aquellos juegos en los que los jugadores disponen de toda la información sobre el juego, un ejemplo sería el ajedrez. Sin embargo, al tratar con información imperfecta, hablamos de juegos en los que un jugador no tiene toda la información sobre el estado del juego, por ejemplo el póker.
- La existencia o no de **pagos colaterales** (equilibrio de Nash), el equilibrio de Nash es una situación en la que los jugadores están satisfechos con el estado actual del juego. Un ejemplo de equilibrio de Nash se da en un regateo, cuando los jugadores llegan a un acuerdo común acaba el regateo y se alcanza este equilibrio.
- El **reparto de beneficios** que deriva en juegos de suma nula o juegos de suma no nula. Los juegos de suma nula son aquellos en los que el beneficio de un jugador supone la pérdida de otro, es decir, se gana exactamente lo que pierde el oponente. Por otro lado, los juegos de suma no nula son aquellos en los que la ganancia de un jugador no deriva estrictamente en la pérdida de otro.

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 en un juego hace referencia al número medio de nodos hijo en cada nodo.En caso de que este valor no sea uniforme, se puede calcular el factor de ramificación medio. Cuanto mayor sea este factor más complicado, computacionalmente hablando, será encontrar una solución ya que el número de nodos a expandir y explorar aumenta exponencialmente en cada nivel.

El algoritmo **minimax** es un algoritmo recursivo que avanza en profundidad hasta llegar a un nodo hoja o hasta el límite de exploración y devuelve los valores de ese nodo hacia arriba. En cada nivel tomaremos el mínimo o el máximo de los sucesores, lo que nos garantiza que tomamos la mejor decisión suponiendo que el oponente también busca obtener el máximo beneficio. Los pasos para la ejecución del algoritmo son los siguientes:

1. Generación del árbol de juego. Se generarán todos los nodos hasta llegar a un estado terminal.
2. Cálculo de los valores de la función de utilidad para cada nodo terminal.
3. Calcular el valor de los nodos superiores a partir del valor de los inferiores. Dependiendo del nivel, si es MAX o MIN, se elegirán los valores mínimos y máximos representando los movimientos del jugador y del oponente, de ahí el nombre de minimax.
4. Elegir la jugada valorando los valores que han llegado al nivel superior.

El algoritmo de **poda alfa-beta** se comporta de la misma forma que el algoritmo **minimax**, pero en este caso introducimos dos variables auxiliares (alfa y beta) que iremos modificando tras recorrer el árbol y que dependiendo de los valores obtenidos en los nodos nos ayudará a decidir si explorarlos o no.

- 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.

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

Resolver una problema de inteligencia artificial requiere de un proceso de análisis del dominio del problema y un estudio sobre la elección del vocabulario y codificación de los axiomas que necesita para inferir. Dependiendo del problema puede presentar problemas en la semántica debido a la complejidad de representar todo el conocimiento con fórmulas lógicas.

Es complicado hacer razonamientos sobre predicados, ya que estos se aplican a objetos o funciones, pero nunca a otros predicados.

Es complicado expresar que dos cosas sean iguales, ya que deben tener las mismas propiedades en todos los sentidos. Hay propiedades que no se pueden conocer, por tanto, no podemos decir que sean las mismas.

Otro concepto complicado de representar es el tiempo, ya que en cada instante puede cambiar el estado del conocimiento. El tiempo es un componente continuo, cosa que hace que sea imposible trabajar con él, y a pesar de poder tratarlo como una variable discreta deberíamos añadir acciones que nos permitan operar con él, lo complica la implementación.

No es recomendable intentar resolver problemas en los que no dispongamos de información completa o bien que esta sea imprecisa. Dentro de esta incertidumbre podemos diferenciar la probabilidad y la vaguedad. La probabilidad nos dice la dificultad de predecir algo de forma exacta aún teniendo toda la información, la vaguedad, por otro lado, hace referencia a la falta de información precisa sobre algo. Ambas son difíciles de representar usando cálculo de predicados.

Hay situaciones en las que una condición no es cierta aunque haya una regla para ello, por ejemplo. "Los pájaros vuelan" es una regla útil ya que la mayoría de los pájaros vuelan. Las excepciones hacen que sea complicado representar la información correctamente.

El cálculo de predicados presenta también varios problemas computacionales.

Consistencia: Hay predicados que mostramos pero no podemos decir que son verdaderos.Por ejemplo las paradojas.

Completitud: Algo puede ser verdadero y no poder demostrarse. No siempre se podrá demostrar excepto si tenemos los axiomas adecuados. Partiendo de un conjunto de axiomas si queremos llegar a una verdad podemos concluir con que es verdadero, pero también podemos obtener que los axiomas no son los adecuados y no podremos afirmar nada.

Es muy pesado computacionalmente, para cada demostración añadimos un axioma. Conforme se van obteniendo clausulas el dominio va creciendo exponencialmente.

En resumen, el cálculo de predicados puede ofrecer buenos resultados cuando tratamos con un dominio del problema que no va a cambiar y del cual disponemos de toda la información y esta es precisa. Dependiendo del problema puede que sea más complejo semántica y computacionalmente hablando.

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

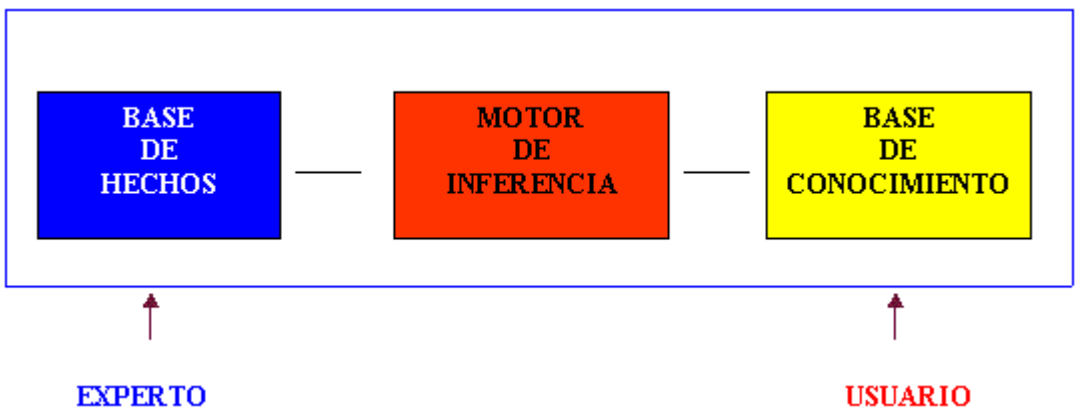
Los modelos de conocimiento heredable se acercan mucho más a la forma en la que los humanos almacenamos el conocimiento de lo que lo hace el cálculo de predicados.

Las redes semánticas son redes asociativas que buscan representar el lenguaje natural. Podemos representarlas como un grafo dirigido etiquetado en el que los nodos representan instancias y los arcos relaciones. Usan la herencia como mecanismo de razonamiento, de esta forma un concepto puede heredar las propiedades de otro concepto más alto en la jerarquía. Al tener que representar un sistema bastante complejo las redes se vuelven inmanejables, lo que dio lugar a la creación de los frames.

Un frame es una estructura de datos que nos permite representar el conocimiento de forma que un ordenador pueda operar con él. Cada frame se caracteriza por un conjunto de campos que normalmente están asociados a atributos, y que en conjunto sirven para identificar los marcos. Podemos discernir frames clase, que representan conocimiento de clases de objetos y frames instancia, que representan conocimiento de objetos individuales. Además de la parte declarativa un frame puede añadir una procedimental, que mediante funciones y métodos ayudan a calcular soluciones y reducir el coste de la inferencia.

5. Estructura y componentes de un sistema experto

Un sistema experto es un sistema basado en el conocimiento que se comporta como un experto (humano) en un determinado dominio de actividad. Es decir resuelve un problema y puede ser consultado de manera que este justifica su razonamiento. Un sistema experto tiene los mismos componentes que un sistema basado en el conocimiento, pero añadiendo un subsistema de explicación que es el que justifica su comportamiento. Podemos ver los siguientes componentes:



Base de conocimiento: Contiene el conocimiento que se extrae del experto codificado en la base de datos. Generalmente se representa mediante reglas.

Base de hechos: Contiene la información que se genera tras una consulta. A una información dada el sistema la empareja con su conocimiento y genera una salida.

Motor de inferencia: Es el encargado de trabajar con la información de la base de conocimiento y la base de hechos para deducir nuevos hechos.

Interfaz de usuario: La interacción entre un sistema experto y un usuario se realiza en lenguaje natural. La interfaz de usuario es la encargada de facilitar al usuario la comunicación con el sistema.

Los sistemas expertos además incluyen un módulo de explicaciones/justificación que permite desde explicar cómo se ha llegado a una solución hasta la justificación de la necesidad de datos adicionales.

6. Paradigmas de Aprendizaje Automático.

Dentro del aprendizaje automático podemos diferenciar los distintos paradigmas de aprendizaje:

- **Aprendizaje memorístico:** En un humano equivale al aprendizaje común, aprendemos cosas y las almacenamos en el cerebro. El equivalente en un ordenador sería el de crear una base de conocimiento y añadir información nueva.
- **Aprendizaje deductivo:**Utiliza la deducción lógica para inferir nuevo conocimiento. Es un razonamiento artificial.
- **Aprendizaje analítico, basado en explicaciones:** Se construye una explicación para cada ejemplo en relación con un concepto dado y generalizamos la explicación de modo que pueda emplearse en el futuro.
- **Aprendizaje analógico:** Entender y resolver una situación por su parecido con otras anteriormente resueltas.
- **Aprendizaje inductivo:** Es el más ampliamente estudiado dentro del aprendizaje. Se trata de aprender un concepto o una clasificación a partir de ejemplos y contraejemplos.

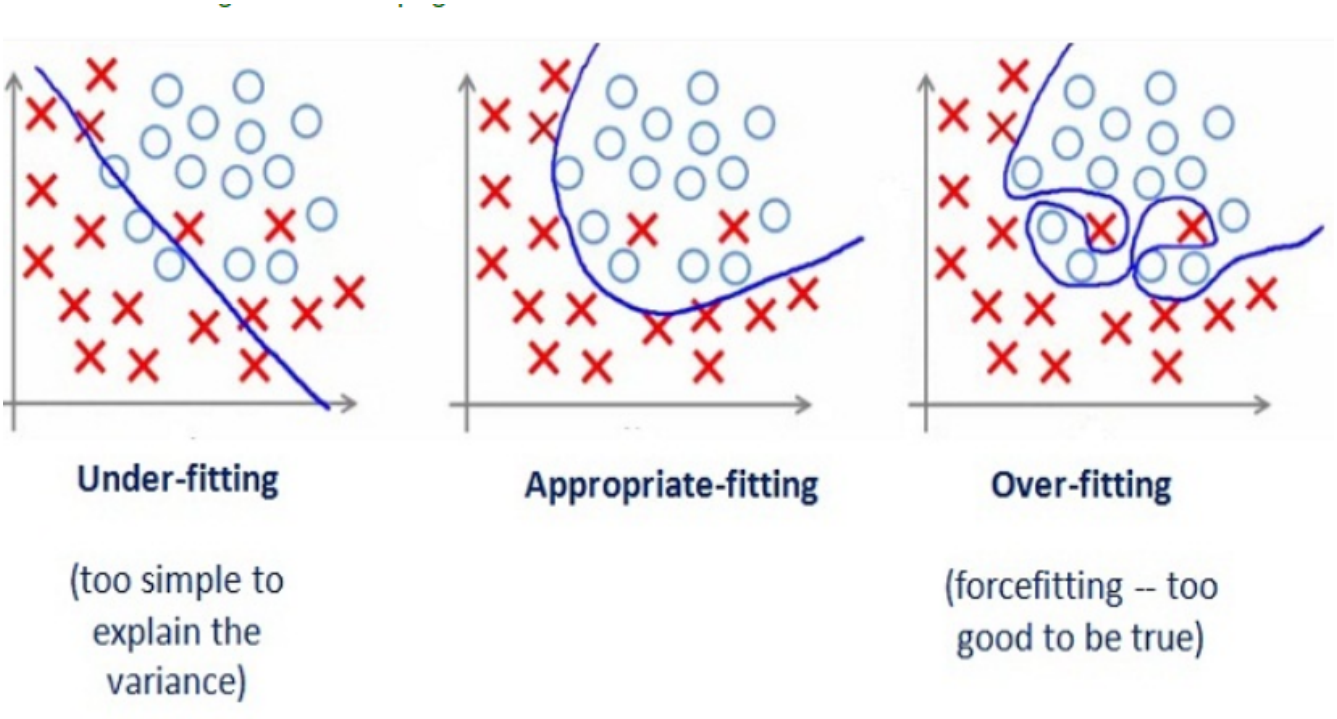
Dependiendo del conocimiento usado podemos diferenciar los siguientes tipos:

- **Aprendizaje supervisado:** El aprendizaje supervisado busca inferir una función determinada dados unos datos de aprendizaje. Es decir, teniendo una **x** y una **y** deducir una **f** tal que **f(x) = y**. Podemos diferenciar clasificación,cuando la salida es categórica, y regresión cuando la salida es cuantitativa.
- **Aprendizaje no supervisado:** En el aprendizaje no supervisado sólo tenemos datos de entrada, por lo que se busca modelar una distribución que nos permita distribuir los datos. Podemos diferenciar los siguientes algoritmos:
    - Agrupación: el objetivo es descubrir agrupaciones en los datos
    - Análisis de asociaciones: queremos ir y descubrir las reglas que mejor describen la mayor parte de los datos posibles
    - Estimación cuantil
- **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.

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

Podemos decir que tenemos ruido cuando dos ejemplos con la misma descripción (en término de atributos) se clasifican de forma distinta. Este problema se puede dar debido a una mala medición de los datos o al no tener en cuenta todos los atributos a la hora de clasificar, es decir, si de un conjunto de datos con 20 atributos tenemos en cuenta sólamente 2, es muy probable que al encontrar datos cuyos 2 atributos escogidos sean iguales se clasifiquen de forma distinta. El ruido siempre afectará a la tasa de aciertos y errores, podemos solucionar esto tomando una muestra mayor de datos (suponiendo que la mayoría están bien medidos) o teniendo en cuenta más atributos, para así minimizar la tasa de errores.

En cuanto al sobreajuste, el aumentar la complejidad polinómica puede hacer que reduzcamos el error de entrenamiento y creamos que tenemos una solución muy buena, después al testear la solución descubriremos que no es la óptima. Al sobreajustar tendemos a aprender ciertos patrones que presentan los datos que se usan en el entrenamiento, por lo que la solución no va a ser apta para nuevos datos que no hayan sido explorados previamente. Para solucionar el sobreajuste podemos evitar usar funciones muy complejas que tiendan a ajustarse perfectamente al conjunto de entrenamiento (Navaja de Ockham: elegir la hipótesis más simple consistente con los datos) o bien, entrenar usando pequeños conjuntos de entrenamiento y evaluando con otros. Tras repetir el proceso con varios conjuntos distintos se calcula la media aritmética obtenida en las medidas de evaluación.



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

Un árbol de decisión toma como entrada un objeto o una situación descrita a través de un conjunto de atributos y devuelve una decisión, el valor previsto de salida dada esa entrada. Podemos tener atributos discretos y continuos, que proporcionarán una salida discreta (clasificación) o una continua (regresión).

Podemos inferir un árbol de distintas formas:

- **Trivial:** Se crea una ruta del árbol por cada instancia de entrenamiento. Esto ocasiona árboles excesivamente grandes y no funciona bien en caso de tratar con instancias nuevas.
- **Óptima:** Es el árbol más pequeño posible compatible con todas las instancias. Para esto se deben crear todas las posibles soluciones, cosa que no es viable computacionalmente.
- **Pseudo-óptima:** En cada nivel se selecciona un atributo en función de la división que produce.

En un árbol de decisión contamos con situaciones en los nodos intermedios del árbol que pueden estar conectadas con soluciones finales u otras decisiones mediante arcos. Al crear un árbol de decisión buscamos poner los nodos cuyos arcos nos generan más divisiones al comienzo y conforme vayamos explorando los niveles nos acerquemos a una solución más específica.

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

- ¿Qué atributo elegir para el primer nodo?

ATRIBUTO	VALORES	CLASE	
		fallará	*no fallará*
Temperatura	Alto	2	2
	Bajo	1	2
Nivel de vibraciones	Alto	3	1
	Bajo	0	3
Horas defuncionamiento	< 1000	2	3
	>1000	1	1
Meses desde revisión	> 1 mes	2	3
	< 1 mes	1	1