TEMA 1. SISTEMAS BASADOS EN EL CONOCIMIENTO

Resultados de aprendizaje

Al finalizar el tema dedicado a los sistemas basados en el conocimiento, el estudiante deberá ser capaz de:

- Describir el concepto de sistema basado en el conocimiento (SBC).
- Conocer las partes de las que consta un SBC.
- Conocer la arquitectura propia de los SBC.
- Describir el funcionamiento de un motor de inferencias hacia atrás y un motor de inferencias hacia delante.

Contenidos

1.1 Introducción

Un sistema basado en el conocimiento (SBC) es un sistema informático capaz de emular las prestaciones de un experto humano en un área concreta de conocimiento especializado. Más concretamente, el sistema experto debe ser capaz de llevar a cabo las siguientes tareas:

- Aceptar las consultas que el usuario realice acerca de una situación dada del mundo real.
- Aceptar los datos proporcionados por el usuario acerca de esta situación, y solicitar otros datos que el sistema estime relevantes.
- Procesar esta información, en busca de una respuesta a la consulta planteada.
- Emitir la respuesta hallada, que debe ser análoga en la mayor parte de los casos a la respuesta que daría un experto humano.
- Justificar la respuesta finalmente emitida, siempre que el usuario así lo solicite.

Si es posible, es interesante dotar al SBC de un *sistema de aprendizaje*, que modificaría la base de conocimientos de modo automático en función de los casos que vaya recibiendo.

El diseño de un sistema basado en el conocimiento corre a cargo del *ingeniero de conocimiento*, que es una persona que estudia la forma en que un experto en cierta materia toma decisiones y traduce esta información de forma que un ordenador pueda emular el proceso.

1.2 Partes de un sistema basado en el conocimiento

Habitualmente, un sistema experto consta de las siguientes partes:

- La base de conocimientos, que contiene el conjunto de conocimientos expertos aplicables al dominio considerado del mundo real. Esta base permanece constante a lo largo del proceso de razonamiento y también -salvo cuando interviene el módulo de adquisición o el de aprendizaje- de una sesión a otra.
- El *motor de inferencias*, o máquina lógica, que implementa un algoritmo de manipulación de los conocimientos de forma que se alcancen soluciones a los problemas propuestos. Este elemento y el anterior constituyen el núcleo del sistema.
- La *memoria de trabajo*, que contiene los datos proporcionados o los objetivos propuestos por el usuario, y los resultados intermedios (datos deducidos o sub-objetivos generados) obtenidos por el sistema en su proceso de razonamiento. Obviamente, el contenido de este archivo va siendo diferente a lo largo de cada sesión del sistema.
- La *interfaz con el usuario (o los módulos de interfaz con los usuarios)*, que se encarga de presentar de forma comprensible las respuestas del sistema y de aceptar las entradas del usuario. Una parte fundamental es el servicio de explicaciones o de justificación, que debe ser capaz de exponer al usuario el proceso de razonamiento seguido por el sistema, de forma que se explique o justifique la respuesta proporcionada en la consulta.
- El *módulo de adquisición de conocimientos*, que permite la modificación de la base de conocimientos. Esta modificación puede ser realizada por el ingeniero del conocimiento junto con el experto, o automáticamente por el sistema.

En la Figura 1.1. se representan estos elementos, así como las interacciones entre ellos.

Figura 1. 1. Arquitectura de un sistema experto

El módulo que proporciona realmente su potencia al sistema es la *base de conocimientos*. En ella se almacena lo que necesita saber el experto -humano o informático- para realizar su tarea. Cada conocimiento es un cierto fragmento de pericia en la resolución de problemas.

Los formalismos más extendidos para representar el conocimiento en los sistemas expertos en la actualidad son las reglas y las redes bayesianas. En este tema de introducción vamos a describir someramente los sistemas basados en reglas

1.3 Sistemas basados en reglas

La *representación basada en reglas* considera que el conocimiento está constituido fundamentalmente por hechos y reglas. Los hechos son afirmaciones incondicionales acerca de algún aspecto del dominio del sistema. Las reglas son afirmaciones condicionales de la forma SI <combinación de hechos del caso> ENTONCES <acciones sobre los hechos del caso>. Sin embargo, para la *representación basada en redes bayesianas* la información debe estructurarse mediante el uso de variables y mecanismos causales entre ellas.

La representación basada en reglas estructura el conocimiento en *hechos y reglas*.

- Los *hechos* son afirmaciones incondicionales. Son también los componentes mínimos del sistema, o los datos con los que trabaja. Por ejemplo, "la temperatura del sensor s1 es alta".
- Las *reglas* son las estructuras fundamentales. Están formadas por dos elementos que se suelen escribir en las formas:

  <izq> → <der>

  SI <izq> **ENTONCES** <der>

  <izq> se llama también a veces *antecedente, premisa* o *condición*,
  <der> se llama *consecuente, conclusión* o *acción*.

<izq> y <der> están formados por uno o varios hechos, o bien por hechos generalizados (obtenidos sustituyendo constantes por variables o empleando operadores aritméticos y de comparación). Algunos ejemplos serían:

la temperatura es alta y la presión es alta → el peligro es grave.
la temperatura es X y X>100 → el peligro es grave.
el barómetro baja → el tiempo será lluvioso.

Las palabras NO, Y, O se emplean a menudo para agrupar los hechos del antecedente o del consecuente. No deben confundirse, sin embargo, con las correspondientes conectivas lógicas. Cada lenguaje o sistema define un significado diferente para estas palabras, implementando de esta forma diversas versiones del razonamiento por defecto, razonamiento no monótono, etc.

En los casos más sencillos, una base de conocimientos formada por hechos y reglas puede visualizarse como un hipergrafo llamado red de inferencias. Los nodos raíz corresponden a respuestas o hipótesis finales. Los nodos hoja corresponden a hechos primitivos que se consideran datos o evidencias y deben estar presentes en la memoria de trabajo o irse preguntando a lo largo de la sesión de consulta. Los restantes nodos corresponden a hechos o hipótesis intermedias que van siendo generadas por el sistema. Cada regla corresponde a un conector del hipergrafo, cuyo origen es <der> y cuyos hijos son los hechos de <izq>. Por ejemplo, consideremos una base de conocimiento con:

- 7 posibles hechos, representados por las proposiciones a, b, c, d, e, f, g.
- las tres reglas:

  a, b → c
  c → e
  b, f → g

La red de inferencia correspondiente se representa en la figura 1.2.

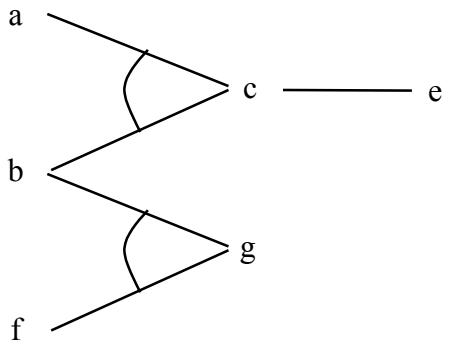

Figura 1. 2. Red de inferencias

Son respuestas finales e y g; datos o evidencias a, b y f; e hipótesis intermedia c.

1.4 Motores de inferencias en los sistemas basados en reglas

En los sistemas que representan el conocimiento mediante reglas, el motor de inferencias consiste en un algoritmo (algoritmo 1) que cíclicamente lleva a cabo estas operaciones:

- Determinar qué reglas se pueden aplicar
- Seleccionar una de ellas (resolución de conflictos)
- Ejecutar la regla seleccionada, modificando así la memoria de trabajo

Esto se repite hasta que la memoria de trabajo adopta cierta configuración que el motor de inferencias considera satisfactoria. Entonces el motor se para y se muestran los valores correspondientes. De este modo, el algoritmo para un motor de inferencias en general sería:

```
Inicializar(memoria-trabajo);
Mientras no configuración-final(memoria-trabajo)
      conjunto-reglas ← aplicables(memoria-trabajo);
      R ← resolver-conflictos(conjunto-reglas);
      memoria-trabajo ← aplicar(R, memoria-trabajo)
fin-mientras;
```

Algoritmo 1. Motor de inferencias

La descripción anterior es muy general. Para diseñar un motor de inferencias, es necesario especificar cuándo una regla es aplicable a la memoria de trabajo (*rule triggering*), cuál es el criterio de selección en caso de varias posibilidades (*resolución de conflictos*) y en qué consiste la ejecución de una regla (*rule firing*). Según se haga esto, tendremos dos grandes clases de motores:

- *con encadenamiento hacia delante o dirigidos por los datos* (*forward-chaining*, *data-driven*), en los que una regla se selecciona para su aplicación cuando su antecedente figura en la memoria de trabajo. A partir de ese momento, el proceso de razonamiento continúa, buscando soluciones del problema. Este esquema resulta apropiado para problemas cuyos dominios conlleven síntesis, como diseño, configuración, planificación, etc.
- *con encadenamiento hacia atrás o dirigidos por los objetivos* (*backward-chaining, goal-driven),* en los que se selecciona un objetivo y el sistema trata de comprobar su validez encontrando evidencias que lo apoyen. Este esquema encaja perfectamente para problemas de diagnóstico, que tienen un número pequeño de conclusiones que pueden ser extraídas, pero gran cantidad de datos iniciales. El sistema sólo pedirá datos en el momento que los necesite, y, cuando encuentre la respuesta, parará la ejecución o se dirigirá a otro objetivo. Es el esquema que se implementó en MICYN y E-MYCIN.

Los motores hacia delante (algoritmo 2) parten de ciertos hechos iniciales y consideran que una regla es aplicable cuando los hechos ya encontrados satisfacen su antecedente. Al ejecutar una regla, añaden a la memoria de trabajo, como nuevos hechos probados, lo que figura en su consecuente (que puede ser la modificación o supresión de un hecho de la memoria de trabajo). El proceso acaba cuando se añade a los hechos del caso un nuevo hecho considerado como final, o cuando ya no hay más reglas aplicables.

```
Mem-trabajo ← hechos-iniciales;
Mientras no configuración-final(mem-trabajo)
      conjunto-reglas ← match(mem-trabajo,antecedentes);
      R ← resolver-conflictos(conjunto-reglas);
      C ← consecuente(R);
      mem-trabajo ← mezclar(C, mem-trabajo)
fin-mientras;
```

Algoritmo 2. Algoritmo motor de inferencias hacia delante

Por el contrario, los motores hacia atrás (algoritmo 3) parten de un objetivo inicial y consideran que una regla está en correspondencia con la memoria de trabajo cuando su consecuente está compuesto por sub-objetivos presentes en ella. Al ejecutar una regla, sustituyen estos sub-objetivos por los que figuran en el antecedente. El proceso acaba cuando todos los sub-objetivos presentes son datos proporcionados por el usuario o hechos de la base de conocimientos.

```
mem-trabajo ← obj-iniciales;
Mientras no configuración-final(mem-trabajo)
      obj ← seleccionar(mem-trabajo);
      conjunto-reglas ← match(obj, consecuentes);
      R ← resolver-conflictos(conjunto-reglas);
      A ← antecedente(R);
      mem-trabajo ← mezclar(A, mem-trabajo)
fin-mientras;
```

Algoritmo 3. Motor de inferencias hacia atrás

Es posible diseñar motores de inferencias que funcionen parcialmente hacia delante y parcialmente hacia atrás. De este modo, y, partiendo de los datos del caso, se puede buscar aquellas hipótesis que son plausibles (con encadenamiento hacia delante). Una vez seleccionadas las hipótesis, y, mediante el encadenamiento hacia atrás, intentaremos demostrar cuáles de ellas son válidas pidiendo más información al usuario.

El momento adecuado para parar el proceso de razonamiento depende tanto del dominio en cuestión como del modelo de tratamiento de la incertidumbre que estemos usando y el efecto que queramos conseguir. Por ejemplo, en un dominio en que se pretenda clasificar un objeto y en el que el conjunto de clases posibles sea exhaustivo y excluyente, no tendrá sentido continuar, puesto que una vez asignado un objeto a una clase no puede pertenecer a otras. Sin embargo, si el sistema está determinando probabilidades de pertenecer a clases puede que solamente sea sensato parar cuando determinada clase alcance una probabilidad superior a cierto umbral prefijado. En dominios de diagnóstico, en los que posiblemente haya más de una avería, puede que tenga sentido continuar el proceso de inferencias hasta el final (o puede que no, si la hipótesis alcanzada se considera lo suficientemente plausible). Para decidir cuándo debe parar el motor, lo que hay que definir es qué se entiende por `configuración-final(memoria_trabajo)`. Podemos por ejemplo decir que es aquella que contiene al menos una hipótesis final (si queremos que el razonamiento pare en el momento que se alcance una conclusión), o aquella que contiene al menos una

hipótesis final con un grado que consideremos aceptable (un factor de certeza de 0.95; una probabilidad de 0.9), o aquella que hace que la agenda esté vacía. Esta definición determinará el momento en el que el motor de inferencias se detiene.

Ejemplo 1. *Funcionamiento de los motores de inferencia*

El funcionamiento de los motores de inferencia se explica mejor con un ejemplo sencillo. Supongamos un sistema que intenta diagnosticar una avería en un coche, y que dispone de las siguientes reglas:

SI el motor no se enciende y el motor recibe gasolina,
ENTONCES el problema es de las bujías
SI el motor no enciende y las luces no se encienden,
ENTONCES el problema es de la batería
SI el motor no se enciende y las luces encienden,
ENTONCES el problema es del arranque
SI hay gasolina, ENTONCES el motor recibe gasolina

Nuestro problema es averiguar qué le ocurre al coche, dados ciertos hechos que son directamente observables. Hay tres problemas posibles con el coche: bujías, batería, arranque. Supongamos que el conjunto de los datos iniciales es vacío.

Un sistema dirigido por los objetivos trataría de comprobar cada posible problema del coche. Supongamos que primero intenta ver si el problema es de las bujías. Entonces, OBJ = {problemas con la bujía?}. Buscaríamos en la base de conocimientos qué reglas hay que tengan "problemas con la bujía" en su consecuente. Solamente está la regla 1, así que los hechos contenidos en su antecedente se convierten en nuestros nuevos objetivos, es decir, OBJ = {motor recibe gasolina? motor enciende?}. Supongamos que el sistema intenta probar primero si el motor recibe gasolina. Este hecho es consecuente de la regla 4, luego el nuevo objetivo es entonces el antecedente de la regla 4, OBJ = {hay gasolina?}. Dado que no es el consecuente de ninguna regla, el sistema preguntará al usuario:

¿Hay gasolina?

Supongamos que la respuesta es que sí. El motor de inferencia añadiría al conjunto de datos del caso, como hecho ya probado, que hay gasolina, de forma que esta pregunta no vuelva a plantearse de nuevo.

Una vez comprobado este objetivo, el conjunto de objetivos a probar es OBJ={motor enciende?}. Dado que no es consecuente de ninguna regla, el sistema preguntará al usuario:

¿Enciende el motor?

Supongamos que la respuesta es que no. Tenemos entonces que el antecedente de la regla 1 se satisface, y por tanto se considera probado que el coche tiene un problema de batería. Dependiendo de cómo se haya implementado el sistema, algunos podrían paren en ese momento, mientras que otros podrían continuar razonando. Supongamos que hemos definido que la configuración final de la memoria de trabajo se alcanza cuando la agenda está vacía. En ese caso, intentaríamos probar la siguiente hipótesis, "problemas con la batería", que es el consecuente de la regla 2. Sus antecedentes se convierten en los nuevos objetivos. Como en los datos del caso figura que el motor no enciende, el objetivo ahora es probar si las luces encienden o no. Se preguntaría al usuario:

¿Encienden las luces?

Supongamos que la respuesta es sí. Se añade a los datos del caso, y se considera que no se ha podido demostrar si hay un problema de batería o no. El sistema intentaría ahora comprobar si hay problemas con el arranque, pero dados los datos del caso (el motor no funciona y las luces no encienden), el sistema no puede inferir nada acerca del sistema de arranque. La interacción completa con este sistema tan simple sería:

Sistema ¿Hay gasolina?
Usuario. Sí.
Sistema ¿Enciende el motor?
Usuario. No.
Sistema ¿Encienden las luces?
Usuario. Sí
Sistema. Creo que el problema es de las bujías.

Nótese que, en general, resolver problemas utilizando el encadenamiento hacia atrás conlleva un proceso de búsqueda en el espacio de las posibles formas de resolver un problema, comprobándolas todas sistemáticamente.

Otro enfoque consiste en utilizar el encadenamiento hacia delante. Se trabaja a partir de los datos iniciales del caso. Supongamos en nuestro ejemplo que los datos iniciales son: {hay gasolina, no enciende el motor, no encienden las luces}.

Se intenta ver qué reglas tienen estos hechos en sus antecedentes. En principio, la única regla aplicable es la regla 4, ya que es la única cuyo antecedente está en el conjunto de datos iniciales. Tras la aplicación de la regla 4, añadimos motor_recibe_gasolina a los datos del caso. Nos encontramos ahora con que es aplicable la regla 1, indicando que el problema es de las bujías.

1.5 Resolución de conflictos en sistemas basados en reglas

Al determinar las reglas aplicables, es posible que varias reglas se satisfagan simultáneamente en un paso del proceso. El motor debe decidir entonces cuál de estas reglas se ejecutará. Es la llamada *resolución de conflictos*. Si consideramos que cada combinación de hechos que se presenta en la memoria de trabajo representa un estado del problema que se resuelve, y que las reglas representan los operadores de transición entre estados, la cuestión se reduce a determinar la estrategia de búsqueda en el espacio de estados.

A) CRITERIOS ESTÁTICOS. La prioridad de cada regla queda determinada cuando se codifica la base de conocimientos. Esto se puede hacer de varias formas:

- *Orden textual de las reglas*. Las reglas anteriores en la base tienen prioridad sobre las posteriores.
- *Utilidad explícita de reglas*. Cada regla tiene asociado un valor numérico. Tienen prioridad las reglas con mayor valor.
- *Utilidad explícita de hechos*. Cada hecho tiene asociado un valor numérico. Tienen prioridad las reglas que añadan o modifiquen hechos de mayor valor. Si el valor numérico se refiere al tiempo que llevan los hechos que activan la regla en la lista de hechos, tenemos las estrategias LEX (menos tiempo) o MEA (más tiempo).

- *Especificidad*. Si el antecedente de R1 subsume al de R2, entonces R2 tiene prioridad sobre R1. O bien si el consecuente de R1 subsume al de R2, entonces R2 tiene prioridad.
- *Generalidad*. Lo contrario a lo anterior.

B) CRITERIOS DINÁMICOS U OPORTUNÍSTICOS. La prioridad se determina en cada paso del motor. Algunos de los criterios más comunes son:

- *Mínima espera.* Tienen prioridad las reglas que llevan menos tiempo en el conjunto de reglas aplicables.
- *Máxima espera.* Tienen prioridad las reglas que llevan más tiempo en el conjunto de reglas aplicables.

C) CRITERIOS DINÁMICOS MANIPULABLES. Los parámetros que sirven para determinar la prioridad se pueden modificar en cada paso del motor. Ello se consigue mediante el uso de meta-reglas, que se aplican en cada paso antes de cualquier regla. Por ejemplo, la meta-regla:

SI (s1 temperatura NO-CALCULADA) y (s2 temperatura alta)
ENTONCES UTILIDAD(s1 temperatura) $\leftarrow$ 100.

establece un valor muy alto para la utilidad del parámetro "temperatura" del sensor s1 siempre que en el sensor s2 la temperatura sea alta.

1.6 Ejercicios propuestos

Ejercicio 1.1. Supongamos un sistema con las siguientes reglas:

R1: SI el motor no se enciende y el motor recibe gasolina, ENTONCES el problema es de las bujías,
R2: SI el motor no enciende y las luces no se encienden, ENTONCES el problema es de la batería,
R3: SI el motor no se enciende y las luces encienden, ENTONCES el problema es del arranque,
R4: SI hay gasolina, ENTONCES el motor recibe gasolina.

Supongamos que los datos del caso son {hay gasolina, motor no enciende, luces encienden}. Describe el funcionamiento del motor de inferencias bajo los siguientes supuestos:

- Supuesto 1. Motor de inferencias hacia delante, resolución conflictos orden de reglas, configuración final cuando se demuestre una hipótesis
- Supuesto 2. Motor de inferencias hacia delante, resolución conflictos orden inverso de reglas, configuración final cuando no se puedan ejecutar más reglas
- Supuesto 3. Motor de inferencias hacia atrás, configuración final cuando se demuestre una hipótesis.

Ejercicio 1.2. (Adaptado de Winston, 1992). Consideremos las siguientes reglas:

R1: SI un animal tiene pelo o da leche, ENTONCES es mamífero
R2: SI un animal tiene plumas o vuela y pone huevos, ENTONCES es un ave
R3: SI un animal es mamífero y come carne, ENTONCES es carnívoro
R4: SI un animal tiene dientes puntiagudos y tiene garras y tiene ojos saltones ENTONCES es carnívoro
R5: SI un animal es mamífero y tiene pezuñas ENTONCES es un ungulado
R6: SI un animal es mamífero y rumia ENTONCES es un ungulado

R7: SI un animal es mamífero y es carnívoro y tiene color leonado y tiene manchas oscuras ENTONCES es un leopardo
R8: SI un animal es mamífero y es carnívoro y tiene color leonado y tiene rayas negras ENTONCES es un tigre
R9: SI un animal es ungulado y tiene cuello largo y tiene piernas largas y tiene manchas oscuras ENTONCES es una jirafa
R10: SI un animal es un ungulado y tiene rayas negras ENTONCES es una cebra
R11: SI un animal es ave y no vuela y tiene el cuello largo y tiene piernas largas y tiene color blanco y negro ENTONCES es un avestruz
R12: SI un animal es ave y no vuela y nada y tiene color blanco y negro, ENTONCES es un pingüino
R13: SI es un ave y vuela bien, ENTONCES es un albatros

Describe cómo funcionará el proceso de inferencias si:

a) Tenemos un motor de inferencias hacia delante y en la memoria de trabajo tenemos un animal llamado Robbie que vuela, pone huevos, y tiene cuello largo
b) Tenemos un motor de inferencias hacia detrás y en la memoria de trabajo tenemos un animal llamado Jimmy, que tiene pelo, dientes puntiagudos, garras, ojos saltones, rayas negras, color leonado, e intentamos probar que Jimmy es un tigre.

Ejercicio 1.3. (Tomado del libro de Castillo, Gutiérrez y Hadi, tema 2). Cuatro agentes secretos, Alberto, Luisa, Carmen y Tomás, están en uno de los cuatro países: Egipto, Francia, Japón y España. Se han recibido los siguientes telegramas de los agentes:

- De Francia: Luisa está en España.
- De España: Alberto está en Francia.
- De Egipto: Carmen está en Egipto.
- De Japón: Carmen está en Francia.

El problema radica en que no se sabe quién ha enviado cada uno de los mensajes, pero es conocido que Tomás miente (¿es un agente doble?) y que los demás agentes dicen la verdad. ¿Quién está en cada país?

1.7 Bibliografía

Castillo, E., Gutiérrez, J., & Hadi, A. (1997). *Sistemas Expertos y Modelos de Redes Probabilísticas.* Monografías de la Academia Española de Ingeniería.
Gómez, A., Juristo, N., Montes, C., & Pazos, J. (1997). *Ingeniería del Conocimiento.* Editorial Centro de Estudios Ramón Areces, S.A.