El 7038: Introducción a la Teoría de Conjuntos Difusos y Sistemas Inteligentes

Semestre: Otoño 2024

Prof.: Claudio Held
Prof. Aux.: Leonardo Causa
Jhon Pilataxi
Ayudante: Francisco Soto
Jorge Zambrano

PROYECTO Nº 2:

SISTEMA BASADO EN CONOCIMIENTO CON ENCADENAMIENTO INVERSO

Este ejercicio recrea la estructura de un sistema de reglas con encadenamiento inverso, similar al usado en MYCIN, incluyendo grados de pertenencia (conjuntos difusos). Usaremos expresiones proposicionales, con un lenguaje de representación que ayude a simplificar el problema.

El objetivo es implementar un sencillo sistema que interactúa con el usuario, y utiliza el conocimiento de que dispone, para evaluar la validez de determinadas hipótesis que son conocidas por el sistema. Las fuentes de conocimiento del sistema son: el conjunto de reglas en su base de conocimiento, que es "conocimiento universal" (es decir, aplicable a todos los casos), y el "conocimiento particular" del caso en estudio, que se puede dividir en el conocimiento acumulado ("Hechos") sobre el caso, y consultas al usuario.

1. VARIABLES GLOBALES

1) *Base de Hechos* (Fact Base): Es el conjunto de hechos que caracteriza al actual problema, con sus condiciones iniciales (si existen), y sus conclusiones parciales e intermedias. En esta aplicación en particular, se trata de toda proposición a la que se haya asociado un valor de incerteza para el caso en estudio.

2) *Base de Reglas* (Rule Base): Es el conjunto de expresiones condicionales que definen el conocimiento en el dominio del problema.

3) *Conjunto Hipótesis* (Hypothesis Set): Son las metas de alto nivel que hay que alcanzar o demostrar.

1.1. Definiciones

La sintaxis de estas variables globales se define de la forma siguiente:

BASE DE HECHOS:

<* Base_ de_ Hechos>	:= <hecho>*	(*: indica que son elementos del conjunto)
<hecho>	:= <tripleta> < vc>	(vc: valor de certeza)
<tripleta>	:= <obj> < atr> < val>	
<obj>	:= símbolo	
<atr>	:= símbolo	
<val>	:= símbolo	
<vc>	:= número real $\in$ [-1, 1]	

BASE DE REGLAS:

<*Base_ de_ Reglas*>	:=	<regla>*
<regla>	:=	<identif> <premisa> < conclusión>
<premisa>	:=	<cláusula>*
<cláusula>	:=	<tripleta>
<conclusión>	:=	<acción>*
<acción>	:=	<tripleta>< vc>
<identif>	:=	símbolo
<tripleta>	:=	<obj> < atr> < val>
<obj>	:=	símbolo
<atr>	:=	símbolo
<val>	:=	símbolo
<vc>	:=	número real $\in$ [-1, 1]

CONJUNTO DE HIPOTESIS:

<*Conjunto _Hipótesis*>	:=	<hipótesis>*
<hipótesis>	:=	<tripleta>< vc>
<tripleta>	:=	<obj> < atr> < val>
<obj>	:=	símbolo
<atr>	:=	símbolo
<val>	:=	símbolo
<vc>	:=	número real [-1, 1]

Para ilustrar la nomenclatura anterior, se muestran dos ejemplos:

1) Un <hecho> que puede ocurrir es que “el animal tiene pelos con valor de certeza 0,5”. En el diseño propuesto, este <hecho> se expresa como "((animal tiene pelo) 0,5)" y se compone sintácticamente de una <tripleta> y un <vc>, en que <vc> = 0,5, y la <tripleta> tiene tres componentes, que son:
   <obj>: “El animal”
   <atr>: “tiene”
   <val>: “pelos”

2) Analizando la primera regla (R1) de la base de reglas de esta aplicación, de acuerdo a la sintaxis recién definida, se tiene que:
   <identif>: R1
   <premisa>: es un conjunto de cláusulas, que en la regla R1 tiene sólo un elemento:
   <cláusula>: (animal tiene pelo)
   <conclusión>: es un conjunto de acciones, que en la regla R1 tiene tres elementos:
   <acción>: ((animal es mamífero) 0.8)
   <acción>: ((animal es ave) -1,0)
   <acción>: ((animal es reptil) -1,0)

   Cada acción puede subdividirse sintácticamente de forma análoga a un <hecho>, que se muestra en el ejemplo 1.

1.2 Inicialización de las variables globales

Base de Hechos: inicialmente vacía.

Base de Reglas: contiene las siguientes reglas:

(R1: ((animal da leche)) (((animal es mamífero)1,0) ((animal es ave) -1,0) ((animal es reptil) -1,0))) (R2: ((animal tiene pelo)) (((animal es mamífero) 0,8) ((animal es ave) -1,0) ((animal es reptil) -1,0))) (R3: ((animal pone huevos) (animal tiene piel dura)) (((animal es mamífero) -1,0) ((animal es ave) -1,0) ((animal es reptil) 1,0))) (R4: ((animal pone huevos) (animal puede volar)) (((animal es ave) 1,0) ((animal es reptil) -1,0))) (R5: ((animal tiene plumas)) (((animal es mamífero) -1,0) ((animal es ave) 1,0) ((animal es reptil)- 1,0))) (R6: ((animal tiene garras)) (((animal es carnívoro) 0,8))) (R7: ((animal come carne)) (((animal es carnívoro) 1,0))) (R8: ((animal es mamífero) (animal es rumiante)) (((animal es ungulado) 0,75))) (R9: ((animal es mamífero) (animal tiene pezuñas)) (((animal es ungulado) 1,0))) (R10: ((animal vive con personas)) (((animal es doméstico) 0,9))) (R11: ((animal vive en zoológico)) (((animal es doméstico) -0,8)))	(R12: ((animal es mamífero) (animal es carnívoro) (animal tiene manchas oscuras)) (((animal es cheetah) 0,9))) (R13: ((animal es mamífero) (animal es carnívoro) (animal tiene rayas negras)) (((animal es tigre) 0,85))) (R14: ((animal es mamífero) (animal es carnívoro) (animal es doméstico) (((animal es perro) 0,9))) (R15: ((animal es reptil) (animal es doméstico)) (((animal es tortuga) 0,7))) (R16: ((animal es mamífero) (animal es ungulado) (animal tiene cuello largo)) (((animal es jirafa) 1,0))) (R17: ((animal es mamífero) (animal es ungulado) (animal tiene rayas negras)) (((animal es cebra) 0,95))) R18: ((animal es mamífero) (animal puede volar (animal es feo)) (((animal es murciélago) 0,9))) (R19: ((animal es ave) (animal vuela bien)) (((animal es gaviota) 0,9))) (R20: ((animal es ave) (animal corre rápido)) (((animal es avestruz) 1,0))) (R21: ((animal es ave) (animal es parlanchín)) (((animal es loro) 0,95))) (R22: ((animal es mamífero) (animal es grande) (animal es ungulado) (animal tiene trompa)) (((animal es elefante) 0,9)))

Conjunto Hipótesis: contiene las hipótesis siguientes:

((animal es perro)	0,0)
((animal es cheetah)	0,0)
((animal es tigre)	0,0)
((animal es elefante)	0,0)
((animal es jirafa)	0,0)
((animal es cebra)	0,0)
((animal es murciélago)	0,0)
((animal es tortuga)	0,0)
((animal es avestruz)	0,0)

((animal es gaviota) 0,0)
((animal es loro) 0,0)

2. ENCADENAMIENTO INVERSO

El objetivo del sistema es demostrar las hipótesis del *Conjunto_Hipótesis. El algoritmo de encadenamiento inverso (AEI) (Backward Chainer) comenzará con la primera hipótesis y seguirá con las siguientes, en el orden predeterminado, hasta que una hipótesis se demuestre verdadera o todas las hipótesis hayan sido evaluadas. Cada hecho tiene asociado un valor de certeza vc en el rango [-1, 1]. Los extremos del intervalo indican certeza completa de falsedad y verdad, respectivamente. El valor intermedio 0 indica ignorancia.

2.1. Casos determinan procedencia de acción

El AEI es un procedimiento recursivo que busca encontrar el valor de certeza de una determinada proposición mediante tres procedimientos alternativos, que se aplican siempre en el orden establecido que se muestra a continuación:

1. **Revisar la base de hechos:** Si la hipótesis H (o conclusión intermedia) fue deducida previamente para el caso actual, está ahora en la base de hechos. El AEI buscará en la base de hechos. Dado que cada hecho tiene asociado un vc, es necesario definir en qué caso una hipótesis puede ser considerada un conocimiento relevante. Para ello definimos un valor umbral β. Un hecho será considerado con relevancia adecuada sólo si $|vc(hecho)| \geq \beta$. Se usa el valor absoluto para tomar en cuenta tanto las certidumbres positivas como negativas (hecho se considera verdadero o falso).

   Si H $\in$ *Base_de_Hechos* se pasa a la etapa siguiente, de lo contrario este caso falla. El resolver a H en la base de hechos puede ser descrito como $F(H) \neq \varnothing$, en que

   $$F(H) = \{hecho \mid hecho \in *Base\_de\_Hechos*, H = hecho, |vc(hecho)| \geq \beta \}.$$

2. **Cuestionar la base de reglas:** La base de reglas está ordenada de un cierto modo (por ejemplo; según el número de regla, o la secuencia en que fueron creadas). Si $F(H) = \varnothing$ (es decir, el caso 1 falló), pero la hipótesis (o conclusión intermedia) puede ser inferida (demostrada) por al menos una regla en la base de reglas, el AEI se concentra en el subconjunto de reglas cuya **conclusión** contenga a la hipótesis (o conclusión intermedia) H. Como las reglas pueden tener la conclusión H con diferentes grados de certeza, debe definirse un umbral que determine cuándo la fuerza conclusiva de la regla es suficientemente grande como para ser considerada dentro del subconjunto de reglas útiles. La utilidad de la regla para cada caso estará dada por el grado de membresía de las premisas y el grado de implicación de la regla. Si las premisas están completamente satisfechas, el valor de certeza de la inferencia depende solamente del grado de la implicancia. Si decimos que ε es el umbral de mínimo nivel de implicancia aceptable, podemos definir el subconjunto R(H) de reglas útiles como

   $$R(H) = \{ regla \mid regla \in *Base\_de\_Reglas*, H = acción \in conclusión(regla), |vc(acción)| \geq \varepsilon \}.$$

   Nótese que R(H) no cambia durante la ejecución del programa. Por lo tanto, se puede determinar el soporte lógico de cada hipótesis durante el tiempo de compilación, para mejorar la velocidad del programa.
   Si $R(H) = \varnothing$, el caso 2 es infructuoso, de lo contrario el AEI tomará la conjunción de cláusulas de las premisas y tratará de probarlas en el orden en que aparecen, es decir profundidad-primero

(depth first). Como están relacionadas por conjunción, si una cláusula de una regla falla, las demás cláusulas de la premisa correspondiente se abandonan. Una cláusula fallará si su valor de certeza es menor al umbral β predefinido anteriormente.

El procedimiento continúa hasta que H se demuestra con una certidumbre satisfactoria, o hasta que todas las reglas en R(H) se han probado. Si H se demuestra, entonces el AEI anotará H (y sólo H) en la *Base_de_Hechos* con su valor de certeza computado, y pasará a la etapa siguiente.

3. **Preguntar al usuario:** Si fallan los casos 1 y 2, el AEI le preguntará al usuario el vc para H.

Nótese que el caso 2 no falla si hay reglas que podrían demostrar H, es decir $R(H) \neq \emptyset$, pero ocurre que todas las reglas en R(H) fallan. El caso 2 sólo falla si no hay reglas que puedan potencialmente probar H, es decir, si $R(H) = \emptyset$. Si H es una hipótesis de alto nivel, el AEI no preguntará al usuario. Podrá desplegar un mensaje diciendo que esta hipótesis no puede ser demostrada.

2.2. Operadores

En este ejemplo utilizaremos los siguientes operadores para propagar los valores de certidumbre:

1) Conjunción: Para calcular el vc de una premisa, los vc de las cláusulas de la premisa se agregan aplicando la función **min** modificada (ver nota).

2) Propagación: Se utiliza para calcular el vc de una conclusión de la regla gatillada. Una vez obtenido el vc de la premisa (agregada), se aplica el grado de implicancia (es decir, el vc de la regla misma) utilizando el **producto**. Esta función sólo se usa con vc (premisa) $\geq \delta$ (definida más adelante) y vc regla $\geq \varepsilon$.

3) Disyunción: Si una misma conclusión H es obtenida por más de una regla, el vc de H se obtendrá aplicando **max** modificado (ver nota)

Nota: En esta aplicación surgen valores de certidumbre positivos y negativos. La función **max** es sesgada hacia información positiva, y la función min hacia información negativa. Esto significa que en una disyunción una información negativa más fuerte (por ejemplo, con vc = -0,9) es superada (anulada) por una información más débil (por ejemplo, con vc = -0,4). Este resultado no es siempre adecuado para aplicaciones de búsqueda de conocimiento, y debemos considerar modificar las funciones **max** y **min** para que se tome en cuenta el **grado de certidumbre** (es decir, utilizar funciones modificadas que apliquen adecuadamente valores absolutos).

3. CONSIDERACIONES DE CONTROL

3.1 Hipótesis

El propósito de AEI es demostrar una o más hipótesis del conjunto dado (*Conjunto_Hipótesis*). Si las hipótesis son mutuamente excluyentes, es útil usar este conocimiento en forma explícita para evitar interferencias innecesarias. Se puede establecer un parámetro de diseño (d1) que indique si se desea que el AEI se detenga cuando una hipótesis H_i se ha demostrado en forma "satisfactoria". Esta condición se puede establecer como un umbral α para el vc calculado para la hipótesis, que es el valor mínimo de aceptación de la hipótesis. Es decir, si d1=V, y $vc(H_i) \geq \alpha$, se detendrá el proceso.

3.2 Conclusiones intermedias

Es importante controlar los recursos que se utilizan en la tarea de probar conclusiones intermedias. Una forma sencilla de detener esta tarea es comparar el valor absoluto del vc con un umbral γ. Si $|vc(concl.inter.)| \geq \gamma$, el AEI retornará el vc obtenido; de lo contrario debe tratar otras reglas que puedan probar la conclusión intermedia y mejorar el vc calculado.

3.3 Parámetros de Control

Cuando el AEI obtiene el vc de cada cláusula, es necesario recalcular el vc acumulado de la premisa para decidir si es aún útil seguir adelante a la cláusula siguiente, o gatillar la regla si ya se consideró toda la premisa. El vc acumulado se compara con el umbral δ, que indica el valor mínimo de significancia para la premisa de una regla. $|vc\ (premisa)| \geq \delta$. Este valor en general dependerá de vc(regla). Nótese que δ no es un parámetro independiente, sino que está relacionado con β y ε. Si

$$|vc\ (H\ inferido)| = |\ vc\ (premisa) * vc\ (regla\ )| \geq \beta,$$

entonces

$$|vc\ (premisa)| \ \geq\ \beta\ /\ |vc\ (regla)|$$

Por otra parte, una regla útil satisface $|vc\ (regla)| \geq \varepsilon$, y por consistencia, $\varepsilon \geq \beta$. Por lo tanto, δ(regla) está limitado por

$$\beta \ \leq\ \delta(regla)\ \leq\ \beta\ /\ \varepsilon\ \leq\ 1.$$

En resumen, los umbrales pueden ordenarse de la forma siguiente:

-1	-γ	-ε	-β	0	β	ε	α	γ	1

en que

α = parámetro para determinar cuando una hipótesis de alto nivel ha sido demostrada en forma satisfactoria, de manera que no sea necesario analizar otra hipótesis (suponiendo que se trata de hipótesis mutuamente excluyentes).

β = parámetro para determinar cuando un hecho (o su negación) tiene un grado de certidumbre suficiente como para ser útil en una inferencia.

γ = parámetro para determinar cuando un hecho (o su negación) tiene un grado de certidumbre suficiente como para no requerir mayor esfuerzo para mejorar su grado de certidumbre.

δ = parámetro para determinar cuando una premisa tiene un grado de certidumbre suficiente como para gatillar una regla. $\delta(regla) \in [\beta\ ,\ \beta/\varepsilon]$.

ε = parámetro para determinar cuando una regla tiene un grado de certidumbre suficiente como para inferir una hipótesis (o conclusión intermedia) dada, suponiendo que la premisa esté totalmente satisfecha.

4. PARAMETROS A EMPLEAR

d1 = V (detener AEI con primera hipótesis cumplida, ver 3.1).
α = 0,7
β = 0,2
γ = 0,85
ε = 0,5
δ = 0,2 / vc (regla)

5. OPCIONALES

En los códigos implementados y en el informe explicite claramente cada uno de los opcionales que realizó. Compare estos resultados con el sistema sin los opcionales.

5.1. Precalificador de Reglas

El AEI descrito puede mejorarse agregando un precalificador de reglas. En la implementación básica, la premisa de cada regla (incluida en R(H)) es probada secuencialmente, y la verificación de cada cláusula de la premisa genera una profundidad-primero. Puede ocurrir que se gasten recursos (búsqueda automática, consulta al usuario, etc.) para verificar la primera cláusula de la premisa, para luego descubrir que otra cláusula de la misma premisa ya se sabía que es falsa. Se dedicó esfuerzo inútil a una regla que no se puede gatillar.

El precalificador de reglas resolvería este problema, utilizando una búsqueda ancho-primero, evaluando primero que todas las cláusulas de la premisa sean verdaderas o bien aún desconocidas.

5.2. Marcador de conclusiones

Supongamos que el AEI falló con los casos 1 y 2 con una conclusión intermedia dada P (ver 2.1). Aplicando el tercer caso, el usuario manifestó su ignorancia (vc = 0). El sistema anotará P en la *Base_de_Hechos* con vc = 0. Si se necesita P más tarde en el AEI, el sistema planteado no considerará al dato de P en la *Base_de_Hechos* como un dato útil (el caso 1 falla). A continuación, P no puede probar por reglas (la base de reglas no ha cambiado = caso 2 falla), y el sistema le hará la misma pregunta nuevamente al usuario (vuelve el caso 3). Una posible solución es usar un campo especial para marcar esta condición (y otras también).

5.3 Representación

Compilar la base de reglas en un reticulado, indicando las dependencias lógicas de reglas, hipótesis, y conclusiones intermedias.

5.4 Ampliar el universo de preguntas

Reemplazar el predicado implícito en la cláusula de cada premisa (correspondiente a "¿Es cierto que...?") por predicados explícitos que deben ser evaluados, como por ejemplo, "Es falso que...?", "¿Es desconocido que...?","¿...está en este rango...?”**,** **“**¿...está en este conjunto...?", etc. Esta modificación permitiría expresar otras preguntas además de igualación de símbolos.

5.5 Interfaz gráfica

Implementar una interfaz gráfica (por ejemplo, usando App Designer de MATLAB) en la cual el sistema interactúe con el usuario a través de barras deslizantes, botones, etc. y que sea capaz de desplegar en pantalla una foto del animal determinado (en caso de que el sistema seleccione algún animal del conjunto Hipótesis).

6. SE PIDE

El sistema computacional implementado por Ud. debe tener capacidad de “diálogo amistoso” con el usuario, y de explicar las conclusiones a las que arribe.

El proyecto se puede realizar en grupo con un máximo de 2 integrantes. Debe entregar un informe que incluya un resumen al comienzo (abstract), una corta descripción del proyecto, resultados de diversos casos ilustrativos, y una sección de discusión y conclusiones. No entregue tablas sin ningún tipo de análisis, los gráficos, figuras y tablas deben ser numerados y llevar un título autoexplicativo. Recuerde que se evalúa la calidad y no la cantidad. No incluir códigos fuentes en el cuerpo del informe, incluir esta información como parte de un anexo.

Puede utilizar cualquier lenguaje de programación para implementar el proyecto: Matab, R, Python, etc. Entregue una copia del código fuente y también un ejecutable (en caso de ser necesario). Asegúrese de que su programa funciona correctamente. Si es necesario incluya un README con las instrucciones necesarias para la ejecución y/o instalación.

El plazo de entrega vence el jueves **13 de junio de 2024** a las 23:59:59 hrs. a través de Ucursos. Se descontarán 2 puntos (de 100 totales) por cada día de atraso (sábado y domingo se consideran como 1 día de atraso). La parte opcional equivale a 30 puntos extras, los que se suman al puntaje de la parte obligatoria. El puntaje final del proyecto se calcula como: puntaje parte obligatoria + puntaje parte opcional (si se realiza) – descuentos por atraso.

CHB, Adaptado de P.P. Bonissone, "Expert Systems in Computer Engineering”, Oct. 29, 1990.