ESAT INNOVATION SCHOOL | ESCUELA SUPERIOR DE ARTE Y TECNOLOGÍA

Asignatura 2019-2020

Redes Bayesianas y Lógica difusa

Nombre Asignatura: Inteligencia Artificial
Modulo Asociado: Unit 27: Artificial Intelligence (L5)
Profesor: Antonio Barella
Curso: 2019/2020
Autor/es: Lucas Merenciano y Francisco Olmos

Contenido

1. Lógica difusa

1.1 Introducción

En el sistema booleano de hoy en día, representamos que algo es verdadero con un 1 y que es falso con un 0. Sin embargo, las personas tienen la capacidad de comunicarse de una manera simple y precisa usando términos lingüísticos “vagos”.

Por ejemplo, para un ordenador un vaso podría estar vacío (representado con un 0) o lleno (representado con un 1), si tuviese agua. Mientras que, para una persona, el vaso podría estar medio lleno, casi vacío, prácticamente lleno etc. La lógica difusa le permite a un ordenador trabajar con términos y reglas lingüísticas de una manera similar a la que lo hacen los humanos.

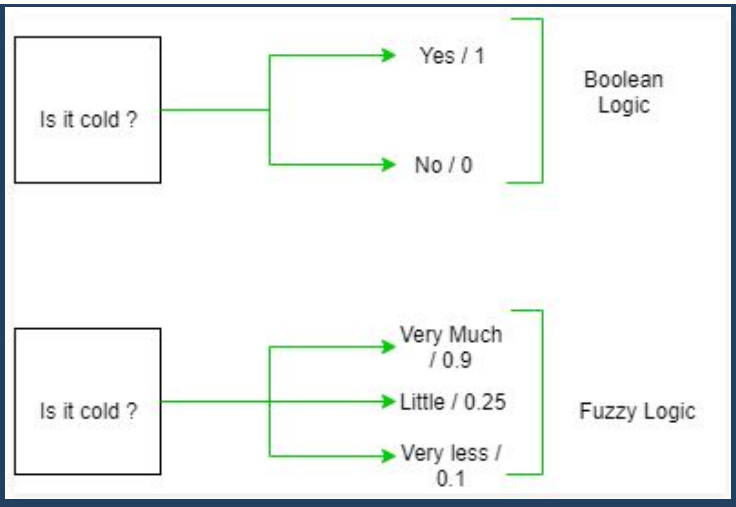

Imagen (1). Diferencia lógica difusa y lógica binaria

1.2 Conjuntos nítidos

Los conjuntos nítidos tienen límites claramente definidos, es decir, pertenecen completamente a un conjunto o no. Esto soluciona bastantes problemas ya que permite clasificar objetos con mucha precisión. Por ejemplo, un número puede ser par o impar.

Impar = {1, 3, 5, 7 …}

Par = {2, 4, 6, 8, 10 …}

Como podemos observar, el grado de pertenencia de un número a un conjunto nítido puede ser verdadero o falso. Es decir, el número 2

es 100% par y 0% impar. Es importante destacar que un número puede pertenecer a varios conjuntos nítidos, por ejemplo, el número 3 pertenece al conjunto de los impares, al conjunto de los números primos y al conjunto de los números menores a 5.

1.3 Operadores

Las operaciones más comunes que se pueden realizar con un conjunto son la unión y la interesección.

1.3.1 Unión

La unión de dos conjuntos es el resultado de juntar los dos conjuntos. Por ejemplo:

A = {1. 2. 3. 4}

B = {3, 5, 7}

A U B = {1, 2, 3, 4, 3, 5, 7}

Hacer una unión equivale a hacer un **OR**.

1.3.2 Intersección

La intersección de dos conjuntos da de resultado un conjunto que contiene los elementos comunes entre ambos. Utilizando como ejemplo los conjuntos anteriores:

A ∩ B = {3}

Hacer una intersección equivale a hacer un **AND**.

1.4 Conjuntos difusos

Los conjuntos nítidos son muy útiles, pero pueden resultar problemáticos en algunos casos. Por ejemplo, el personaje de un juego puede tener hambre o no. Abstrayéndonos a los conjuntos nítidos, el personaje estaría en el conjunto de los que tienen hambre o en el de los que no lo tienen. Lo mismo pasaría si estuviese herido. El problema de esto, es que no se puede cuantificar cuanta hambre tiene o cuan herido está.

Los conjuntos difusos nos permiten extender la noción de los conjuntos nítidos dándoles un valor. Entonces un personaje podría estar herido con un valor de 0.4 y estar hambriento con un valor de 0.8. En el caso de los conjuntos difusos, en lugar de pertenecer a un conjunto o ser excluido de él, todo puede pertenecer parcialmente a un conjunto, y algunas cosas pueden pertenecer a más a un conjunto que a otro. Este valor numérico que le damos a las cosas

es conocido como nivel de pertenencia. Entonces si un personaje tuviese un valor de hambre de 0.9 diríamos que pertenece al conjunto de hambre con un nivel de pertenencia de 0.9. Cuando algo está completamente en un conjunto difuso se le da un valor de 1.

1.5 Fuzzification

La lógica difusa funciona con niveles de pertenencia en los conjuntos difusos. Como no es el formato habitual en la mayoría de juegos o aplicaciones, necesita algunas conversiones. Convertir datos a niveles de pertenencia se llama *fuzzification*. Y devolverlo a su estado anterior se le llama *defuzzification*.

1.5.1 *Fuzzification* de datos numéricos

Esta técnica es la más usadas, consiste en convertir un valor numérico en un nivel de pertenencia para uno o más conjuntos difusos.

Por ejemplo, en un juego, un personaje puede tener distintos puntos de vida, los cuales tenemos que convertir a un nivel de pertenencia de los conjuntos difusos, saludable y dañado.

Para convertir los puntos de vida a un nivel de pertenencia usaremos una función de pertenencia para cada conjunto difuso. Es decir, una función que reciba, por ejemplo, el daño recibido y devuelva el nivel de pertenencia. Por ejemplo, en la siguiente figura se muestran dos funciones de pertenencia, una para el conjunto de saludable y otra para el conjunto de dañado.

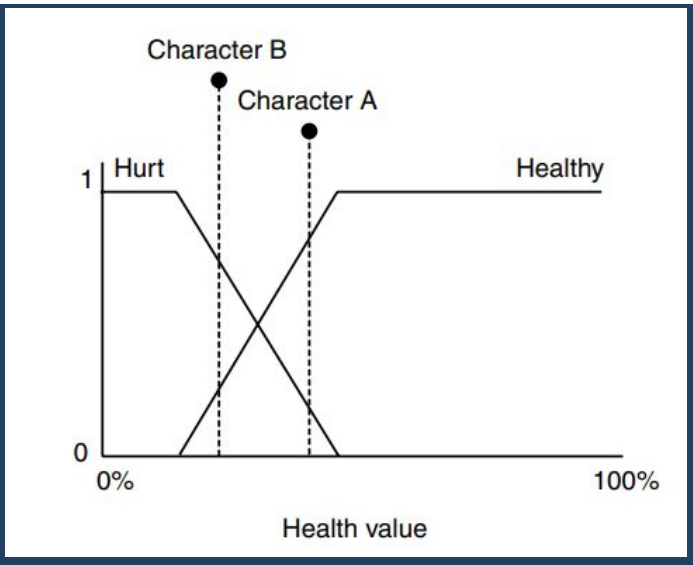

Imagen (2). Ejemplo fuzzification de datos numéricos

De estas funciones podemos leer sus niveles de pertenencia. El personaje A pertenece al conjunto saludable en un 0.8 y al de dañado en un 0.2, mientras que el personaje B pertenece al conjunto saludable con un nivel de pertenencia de 0.3 y al dañado de 0.7.

No hay límite para la cantidad de funciones de pertenencia para un mismo valor y el resultado de la suma de las funciones no tienen por qué ser siempre 1.

1.5.2 *Fuzzification* de otro tipo de datos

En algunas ocasiones tenemos que trabajar con valores que no son numéricos, como por ejemplo con booleanos o enumeraciones. Para estos casos lo más común es guardarnos previamente valores de pertenencia para cada conjunto.

Por ejemplo, un personaje puede tener un booleano que indique si ha conseguido una escopeta. En este caso, la función de pertenencia tendría guardado un valor para cuando es verdadero y otro para cuando es falso. Entonces cuando esta función reciba si el jugador tiene o no la escopeta esta función devolvería un 0 o un 1 en función de si el booleano es verdadero o falso.

1.6 Defuzzification

Defuzzification es el proceso inverso a la *fuzzification*, es decir, consiste en convertir un nivel de pertenencia a un valor, normalmente numérico. Existen diferentes técnicas para este proceso. Algunas de estas son:

1.6.1 Mean of maximum (MOM)

Este método consiste en calcular la media de los valores de salida que tienen los niveles de pertenencia más altos. El mayor problema con este método es que no tiene en cuenta aquellos conjuntos con un nivel de pertenencia igual a la más alta, lo que puede dar como resultado, un valor en un extremo de uno de los conjuntos.

1.6.2 Average of Maxima (MaxAV)

El valor máximo representado por un conjunto difuso es el que tiene un nivel de pertenencia de 1 en ese conjunto. Para hacer la *defuzzification* con este método haremos el sumatorio de los valores de entrada multiplicado por su nivel de pertenencia y eso lo dividiremos entre el sumatorio de los niveles de pertenencia:

Valor de salida = Σ valores de entrada * nivel de pertenencia / Σ nivel de pertenencia

Si hiciésemos el ejemplo con la siguiente tabla:

Conjunto	Valor de entrada	Nivel de pertenencia
Bajo	12.5	0.33
Medio	50.0	0.2
Alto	87.5	0.67

Altura = (12.5 * 0.33 + 50.0 * 0.2 + 87.5 * 0.67) / (0.33 + 0.2 + 0.67)

Altura = 60.625

1.6.3 Reglas difusas

Las reglas difusas relacionan el nivel de pertenencia de algunos conjuntos difusos para generar nuevos niveles de pertenencia para otros conjuntos. Es decir, en base a una condición obtendríamos una consecuencia. Por ejemplo:

Si el personaje tiene una vida menor o igual a 0 entonces el personaje pasaría a estar muerto.

Una misma regla difusa puede tener varias condiciones y el nivel de pertenencia de dicha condición influirá en el resultado. En el caso de que un sistema difuso itere en sus reglas y alguna de ellas se cumpliese entonces se ejecutaría la consecuencia y se haría el proceso de *defuzzification* para devolver un valor en el caso de que fuese necesario.

1.7 Conclusión

Como hemos visto anteriormente la lógica difusa puede tener muchos usos en la industria de los videojuegos. Generalmente podríamos utilizarla en cualquier sistema donde normalmente usaríamos la lógica booleana. Por ejemplo, para determinar las transiciones entre estados en una máquina de estados.

Actualmente donde se utiliza mas es en controladores industriales que toman decisiones en base a unos inputs. Algunos expertos las llaman máquinas de estados difusas.

2. Redes Bayesianas

Una red bayesiana consiste en un gráfico (DAG) que representa un conjunto de variables conocidas y las relaciones de dependencia entre ellas a fin de estimar la probabilidad de las variables no conocidas. Dadas sus características, este modelo resulta idóneo para la clasificación, la predicción o el diagnóstico.

2.1 DAG (grafo acíclico dirigido)

En ciencias de la computación y matemáticas un grafo acíclico dirigido o DAG [1] (del inglés Directed Acyclic Graph), es un grafo dirigido que no tiene ciclos. Esto significa que para cada vértice v, no hay un camino directo que empiece y termine en v. Podría decirse que un DAG "fluye" en solo una dirección.

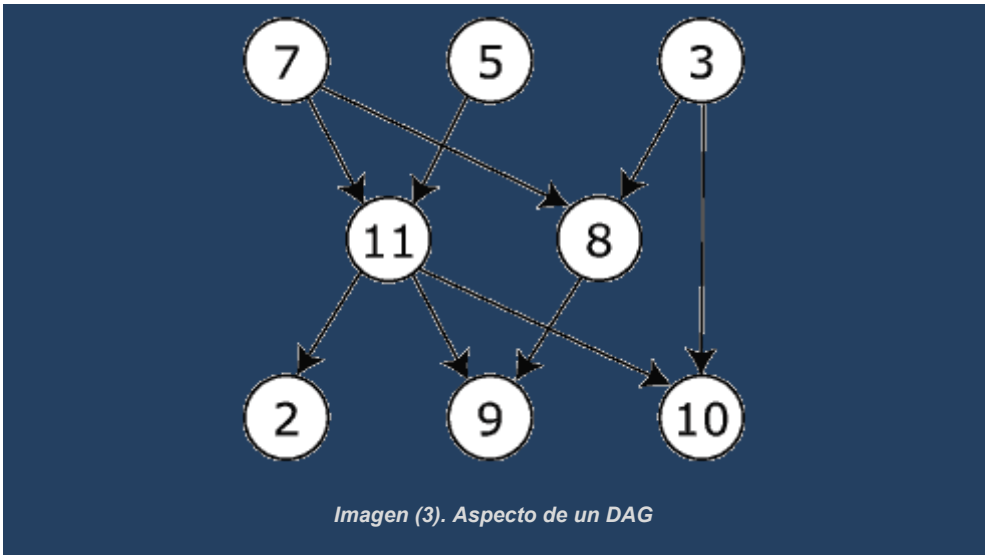

Imagen (3). Aspecto de un DAG

2.2 Inferencia de las redes bayesianas

A partir de una red ya construida, y dados los valores concretos de algunas variables de una instancia, podrían tratar de estimarse los valores de otras variables de la misma instancia aplicando razonamiento probabilístico.

El razonamiento probabilístico sobre las redes bayesianas consiste en propagar los efectos de las evidencias (variables conocidas) a través de la red para conocer las probabilidades a posteriori de las variables desconocidas. De esta forma se puede determinar un valor

estimado para dichas variables en función de los valores de probabilidad obtenidos.

2.3 Ejemplo de red bayesiana simple

Un ejemplo muy utilizado de red bayesiana es la relación entre la lluvia, un aspersor y la humedad de la hierba, suponiendo que hay un 20% de probabilidad de que llueva y un 40% de probabilidad de que el aspersor esté encendido si no llueve.

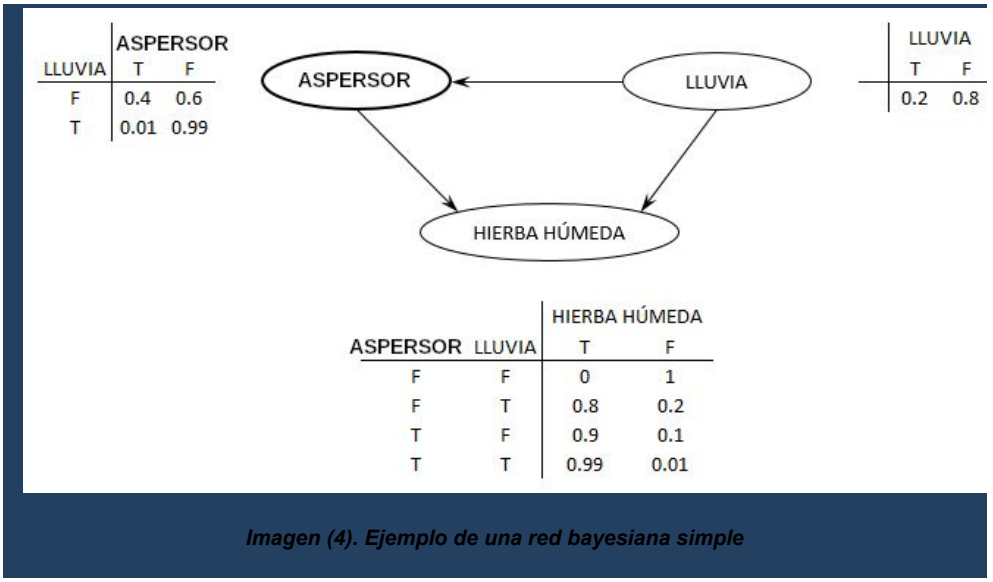

Imagen (4). Ejemplo de una red bayesiana simple

Utilizando este modelo y aplicando diferentes fórmulas, podemos responder a preguntas como:

- ¿Cuál es la probabilidad de que esté lloviendo dado que la hierba está húmeda?

- ¿Cuál es la probabilidad de que lloverá dado que la hierba está húmeda?

- ¿Qué impacto tendrá encender el aspersor?

Las fórmulas necesarias para resolver estas preguntas se encuentran explicadas en este artículo [2].

2.4 Utilidades de las redes bayesianas

En la actualidad, las redes bayesianas poseen muchas aplicaciones, sobre todo cuando la cantidad de datos manipulados aumenta a rápidamente, haciéndose necesario procesarlos e interpretarlos de

forma que sea posible extraer el conocimiento preciso para una adecuada toma de decisiones.

Son muy utilizadas en el campo de la biomedicina, para la predicción de la supervivencia en pacientes con cáncer de mama, entre muchas otras aplicaciones.

Se utilizan también en investigaciones policiales, por ejemplo, para determinar la probabilidad de asaltos a casas en función de un conjunto de variables.

También son utilizadas en muchas empresas para cosas como reconocimiento de voz, pero sobretodo para el diagnóstico de fallos. Microsoft y HP las utilizan para el diagnóstico de problemas de impresora y Intel para el diagnóstico de fallos de procesadores, por ejemplo

También en el campo de la psicología, entre otras cosas para tratar y prevenir trastornos del estado de ánimo.

Incluso la Nasa utiliza este tipo de redes como ayuda a la decisión de misiones espaciales.

2.5 Uso en los videojuegos

A pesar de ser una herramienta muy interesante, las redes bayesianas no son muy utilizadas en la industria del videojuego.

Podrían ser muy útiles a la hora de dotar de comportamientos “humanos” a los enemigos controlados por la máquina (bots), ya que estos podrían decidir cómo actuar en base a la información que tienen.

El “problema” es que en videojuegos la falta de información no es un problema, por lo tanto, no es tan útil deducir información si en el propio código del videojuego ya está toda, por lo tanto el propio “bot” no necesita deducir nada.

Por lo tanto, en el mundo de los videojuegos, a la hora de simular inteligencia, se opta por acceder directamente a toda la información que necesite el enemigo, causando que el enemigo sea extremadamente eficiente (lo cual lo deshumaniza), por ese motivo se le programa para que cometa fallos, simulando un comportamiento más humano.

En ámbitos más académicos, se ha demostrado su utilidad a la hora de programar inteligencias artificiales para que hagan de rival para el

jugador, adaptando su dificultad según la partida. Un claro ejemplo es el BayesChess [3].

2.6 Conclusión

Las redes bayesianas han demostrado ser extremadamente útiles y son utilizadas en una gran variedad de áreas, destacando la toma de decisiones.
Su punto fuerte consiste en la deducción de información de la cual no se dispone a partir de la disponible.
En videojuegos no son muy utilizadas debido a que su capacidad deductiva no es tan útil ya que se dispone de toda la información necesaria del estado del juego.
Pero ha demostrado potencial a la hora de simular comportamientos y errores humanizados y seguramente cada vez aparecerán más videojuegos que la utilicen.

3. Referencias Bibliográficas

3.1 Referencias

[1] Weisstein, E., 2020. *Acyclic Digraph -- From Wolfram Mathworld*. [online] Mathworld.wolfram.com. Available at: <https://mathworld.wolfram.com/AcyclicDigraph.html> [Accessed 18 May 2020].

[2] En.wikipedia.org. 2020. *Bayesian Network*. [online] Available at: <https://en.wikipedia.org/wiki/Bayesian_network> [Accessed 18 May 2020].

[3] Fernández, A. and Salmerón, A., 2007. *(PDF) Bayeschess: A Computer Chess Program Based On Bayesian Networks*. [online] Academia. Available at: <https://www.academia.edu/10522250/BayesChess_A_computer_chess_program_based_on_Bayesian_networks> [Accessed 18 May 2020].

3.2 Bibliografía

Matt Buckland, G. (2005). Programming Game AI by Example.

Ian Millington, G. (2006). Artificial Intelligence for Games.

https://www.geeksforgeeks.org/

http://www.lcc.uma.es/~eva/aic/apuntes/fuzzy.pdf

http://casanchi.com/mat/intrologicadifusa01.pdf

3.3 Fuente de imágenes

(1) https://www.geeksforgeeks.org/fuzzy-logic-introduction/

(2) Ian Millington, G. (2006). Artificial Intelligence for Games.

(3) https://es.wikipedia.org/wiki/Grafo_ac%C3%ADclico_dirigido

(4) https://es.wikipedia.org/wiki/Red_bayesiana