Sistema Experto, Lógica Difusa y un poco de Redes Neuronales

Expert System, Fuzzy Logic and a bit of Neural Networks

Christian Daniel Núñez Mejía

*Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: christian.nunez@utp.edu.co

**Resumen:** Los Sistemas Expertos son una rama de la Inteligencia Artificial cuyo propósito es simular el comportamiento de un experto humano. Para ello se establece una arquitectura cuyo corazón es un motor de inferencia soportado por una base de conocimiento, organizada en hechos, reglas y meta reglas. El motor de inferencia obedece a patrones de entrada iterando a lo largo de la base de conocimiento, encontrando hechos u objetos que resuelven en lógica de predicados de primer orden (o en el ámbito de la lógica difusa), un objetivo específico.

El Sistema Experto requiere para lograr su propósito de la utilización de una agenda temporal en donde almacena el resultado de todas y cada una de las acciones de inferencia. Globalmente, el Sistema Experto requiere disponer de un módulo de aprendizaje a través del cual mejora su sistema de inferencia con base en la experiencia.

El Motor de búsqueda de conocimiento opera en tres niveles posibles: directo, inverso e híbrido. Para el desarrollo de Sistemas Expertos se dispone de herramientas computacionales y lenguajes tanto imperativos como funcionales.

**Palabras Clave:** Sistemas Expertos, Lógica Difusa, Aprendizaje, Patrones, Inferencia, Motor de Inferencia, Agenda, Hechos, Reglas, Meta Reglas, Directo, Inverso, Híbrido, Entrada, Salida, Base de Conocimiento, Inteligencia Artificial, Agente Inteligente. Acción, Reacción, Incertidumbre, Verdadero, Falso, Red Neuronal.

**Abstract:** The Expert Systems are a branch of Artificial Intelligence whose purpose is to simulate the behavior of a human expert. To do this, the architecture is established whose heart is an inference engine supported by a knowledge base, organized into facts, rules and meta rules. The inference engine obeys to input patterns iterating along the knowledge base, finding facts or objects that solve in logic of first order predicates (or in the field of fuzzy logic), a specific objective.

The Expert System requires achieving its purpose of using a temporary agenda where it stores the result of each and every one of the inference actions. Globally, the Expert System requires having a learning module through which it improves its inference system based on experience.

The Knowledge Search Engine operates in three possible levels: direct, inverse and hybrid. For the development

of Expert Systems, computational tools and languages are available, both imperative and functional.

**Key Word**: Expert Systems, Fuzzy Logic, Learning, Patterns, Inference, Inference Engine, Agenda, Facts, Rules, Goal Rules, Direct, Inverse, Hybrid, Input, Output, Knowledge Base, Artificial Intelligence, Intelligent Agent. Action, Reaction, Uncertainty, True, False, Neural Network.

### CONTENIDO

1. **Sistemas Expertos:** Es un sistema computacional que imita la toma de decisiones de un humano experto en un tema específico. Están compuestos por una Base de Conocimientos (BC), una Base de Hechos (BH) y un Motor (o Máquina) de Inferencias (MI).  
   Estos sistemas tienen ventajas técnicas respecto a un ser humano, como la rapidez, la permanencia, la fiabilidad, el bajo costo entre otros pero presentan desventajas como la carencia de sentido común, la comunicación nula, la falta de flexibilidad, etc.
   1. **Concepto:** Sistema que busca conclusiones a partir de hechos aplicando reglas.

**Hecho:** 1) Lluvia.

2) Futbol.

3) Cine.

**Reglas:** R1. Llueve > Cine.

R2. No llueve > Futbol.

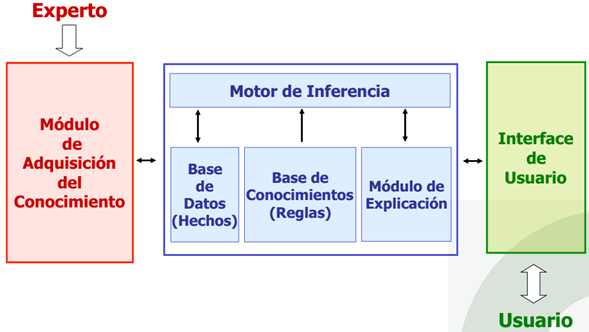
**Maquina:** Explorando la maquina buscando concordancia de hechos.

R1. Llueve > Cine.

Hecho Conclusión.

Objeto > Conclusión.

* 1. **Arquitectura:**



* 1. **Ejemplo Base**

**Modelo (MYCIN).**

**Hechos:**

|  |  |  |
| --- | --- | --- |
| **01** | **Fiebre** | **Síntoma** |
| **02** | **Tos** | **Síntoma** |
| **03** | **Malestar** | **Síntoma** |
| **04** | **Gripa** | **Enfermedad** |
| **05** | **Faringitis** | **Enfermedad** |
| **06** | **Migraña** | **Enfermedad** |
| **07** | **Dolor de Cabeza** | **Síntoma** |

**Regla:**

**R1: Síntoma > enfermedad.**

**Peso**

**R2:** Fiebre > Gripa. 0.6

**R3:** Tos > Gripa . 0.3

**R4:** Malestar > Gripa. 0.9

**R5:** Dolor de Cabeza > Gripa. 0.2

**R6:** Fiebre > Faringitis. 0.4

**R7:** Tos > Faringitis. 0.1

**R8:** Malestar > Faringitis. 0.7

**R11: Diagnostico > Tratamiento**.

**R12:** Tiene Gripa > Acetaminofén 1.0

**R9:** Malestar > Migraña 0.3

**R10:** Dolor de Cabeza > Migraña 1.0

**Motor de Inferencia:**

**I: Fiebre** R2, R6………Agenda

**Gripa [**0.6**]**

**---------------------------------------**

**Faringitis [**0.4**]**

**II: Tos** R3, R7…………Agenda

**Gripa [**0.6+(1-0.6) \*0.3**]**

**Gripa [**0.72**]**

**----------------------------------------**

**Faringitis [**0.4+(1-0.4)-0.1**]**

**Faringitis [**0.46**]**

1. En algún Lenguaje, Programarlo.
2. En SQL.
3. En CLIPS.
   1. **CLIPS.**

**Evolución Histórica.**

**Primeras versiones:**

El primer prototipo de CLIPS fue desarrollado durante la primavera (boreal) de 1985, y tomó poco más de dos meses.

Era compatible con todos los desarrollos hechos por la sección

de Inteligencia Artificial, y su sintaxis estaba basada en la sintaxis de ART (otra herramienta para el desarrollo de sistemas expertos). Sin embargo, CLIPS fue desarrollado sin tener acceso o haber conocido previamente el código fuente de ART.

Inicialmente, con el desarrollo de CLIPS se buscaba tener mayor conocimiento sobre la construcción de sistemas expertos y sentar las bases de un lenguaje para reemplazar las herramientas comerciales que estaban siendo usadas. La versión 1.0 de CLIPS demostró que eso era posible. Después de un desarrollo adicional, se vio que el costo de CLIPS sería significativamente menor al de otras herramientas y que sería ideal para entrenamiento. Otro año de desarrollo y de uso interno sirvió para mejorar portabilidad, desempeño, funcionalidad y documentación de soporte. A mediados de 1986, CLIPS v3.0 estuvo disponible para grupos fuera de la NASA.

**Desarrollo y Expansión:**

Otras funcionalidades agregadas transformaron CLIPS; ya no era una herramienta para entrenamiento sobre construcción de sistemas expertos, sino que ahora servía también para el desarrollo y ejecución de los mismos. Las versiones CLIPS v4.0 y v4.1 (1987) tenían significativas mejoras en cuanto a desempeño, integración con otros lenguajes, y capacidad de ejecución. CLIPS v4.2 (1988) fue una completa re-escritura

del código fuente con el fin de hacerlo más modular. Esta versión también incluyó un manual detallado de la arquitectura de CLIPS y una aplicación de ayuda para la verificación y validación de programas basados en reglas. Nuevas funcionalidades vinieron con la CLIPS v4.3 (1989). Originalmente, CLIPS era un lenguaje de reglas basado en el

Algoritmo Rete (Programación Lógica). CLIPS v5.0 (1991) introdujo dos nuevos paradigmas de programación:

Programación Imperativa y Programación Orientada a Objetos

(POO). El lenguaje POO dentro de CLIPS es llamado COOL

(CLIPS Object-Oriented Language, o Lenguaje Orientado a

Objetos de CLIPS). CLIPS v5.1 (1991) ya soportaba las recientemente desarrolladas o mejoradas interfaces X Windows, MS-DOS y Macintosh. CLIPS v6.0 (1993) tenía nuevas funcionalidades relacionadas con el reconocimiento de patrones en objetos/reglas y soporte a Ingeniería de Software basada en reglas. CLIPS v6.1 (1998) soportaba compiladores C++, aunque ya no soportaba los viejos compiladores C no ANSI. También se agregaron comandos para llevar control del tiempo de desarrollo y para funciones definidas por el usuario. CLIPS v6.2 es soportada por diversos sistemas operativos, y tiene mejoras en su interfaz de desarrollo para Windows 95/98/NT y MacOS. Actualmente, CLIPS es mantenido fuera de la NASA como software de dominio público.

**Desarrollador:**

Gary Riley tiene más de treinta años de experiencia en el campo de los sistemas expertos. Mientras trabajaba en el Centro Espacial Johnson de la NASA, fue responsable del diseño y desarrollo de los componentes basados en reglas de CLIPS. Desde que dejó la NASA, ha continuado desarrollando y manteniendo de forma independiente una versión de dominio público de CLIPS. Es el coautor del libro de texto ‘Sistemas expertos: Principios y programación’, ahora en su cuarta edición. Además de crear herramientas de sistemas expertos, también ha sido un usuario experto de herramientas de sistemas, desarrollando y manteniendo tres sistemas expertos durante un período de diez años.

**Características Principales:**

**Representación del Conocimiento:** CLIPS permite manejar una amplia variedad de conocimiento, soportando tres paradigmas de programación: el declarativo, el imperativo, y el orientado a objetos. La programación lógica basada en reglas permite que el conocimiento sea representado como reglas heurísticas que especifican las acciones a ser ejecutadas dada una situación. La POO permite modelar sistemas complejos como componentes

modulares. La programación imperativa permite ejecutar algoritmos de la misma manera que en C, Java, LISP y otros lenguajes.

**Portabilidad:** CLIPS fue escrito en C con el fin de hacerlo más portable y rápido, y ha sido instalado en diversos sistemas operativos (Windows 95/98/NT, MacOS X, Unix) sin ser necesario modificar su código fuente. CLIPS puede ser ejecutado en cualquier sistema con un compilador ANSI de C, o un compilador de C++. El código fuente de CLIPS puede ser modificado en caso que el usuario lo considere necesario, con el fin de agregar o quitar funcionalidades.

**Integrabilidad:** CLIPS puede ser embebido en código imperativo, invocado como una sub-rutina, e integrado con lenguajes como C, Java, FORTRAN y otros. CLIPS incorpora un completo lenguaje orientado a objetos (COOL) para la elaboración de sistemas expertos. Aunque está escrito en C, su

interfaz más próxima se parece a LISP. Pueden escribirse extensiones a CLIPS sobre C, y, al contrario, CLIPS puede ser llamado desde C. CLIPS puede ser extendido por el usuario mediante el uso de protocolos definidos.

**Desarrollo Interactivo:** La versión estándar de CLIPS provee un ambiente de desarrollo interactivo y basado en texto; este incluye herramientas para la depuración, ayuda en línea, y un editor integrado. Las interfaces de este ambiente tienen menús,

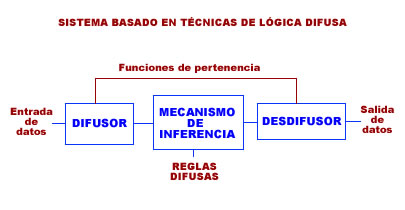
editores y ventanas que han sido desarrollados para MacOS, Windows 95/98/NT, X Windows, entre otros.

**Verificación/Validación:** CLIPS contiene funcionalidades que permiten verificar las reglas incluidas en el sistema experto que está siendo desarrollado, incluyendo diseño modular y particionalmente de la base de conocimientos del sistema, chequeo de restricciones estático y dinámico para funciones y algunos tipos de datos, y análisis semántico de reglas para prevenir posibles inconsistencias.

**Documentación:** En la página web oficial de CLIPS se encuentra una extensa documentación que incluye un Manual de Referencia y una Guía del Usuario.

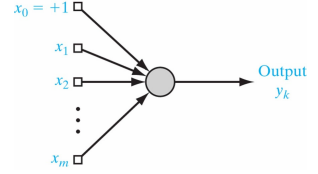
**Bajo Costo:** CLIPS es un software de dominio público.

1. **Lógica Difusa:** La lógica difusa se basa en lo relativo de lo observado como posición diferencial. Este tipo de lógica toma dos valores aleatorios, pero contextualizados y referidos entre sí. Se basa en reglas heurísticas de la forma SI (Antecedente) ENTONCES (Consecuente), siendo el antecedente y el consecuente conjuntos difusos. Este tipo de lógica presenta valores intermedios, no solo 0 y 1.
   1. **Concepto:** En general la lógica difusa imita como una persona toma decisiones basada en información con las características mencionadas.
   2. **Modelo.**



**3. Algoritmo de aprendizaje del perceptrón**

Un umbral (positivo) es equivalente a un sesgo/bias (negativo), por lo que podemos evitar tratar de forma separada el umbral añadiendo una entrada fija x0=+1. De esta forma, aprendemos el umbral como si fuese un peso más.



Se seleccionan ejemplos del conjunto de entrenamiento utilizando cualquier política que garantice que todos los ejemplos de entrenamiento se acabarán escogiendo:

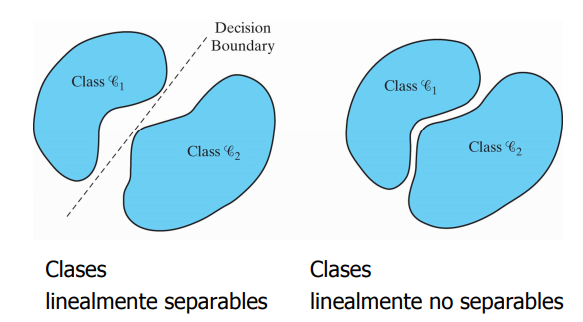
Si la salida es correcta, se dejan los pesos tal cual.

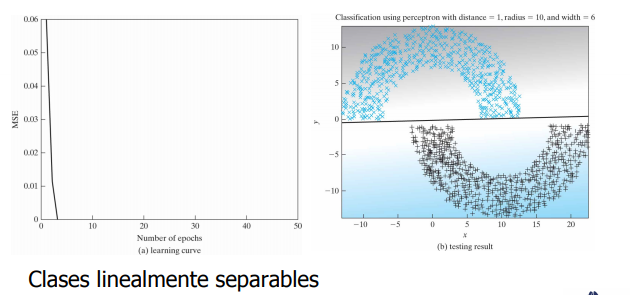
Si la unidad de salida incorrectamente da un cero, se añade el vector de entrada al vector de pesos.

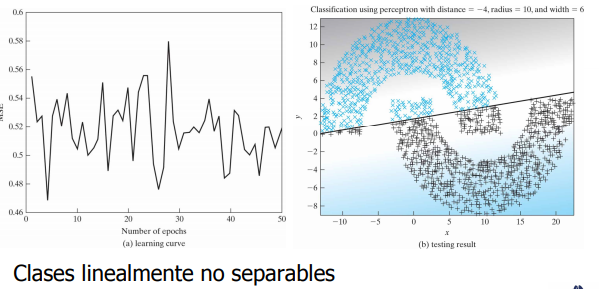
Si la unidad de salida incorrectamente da un uno, se resta el vector de entrada del vector de pesos.

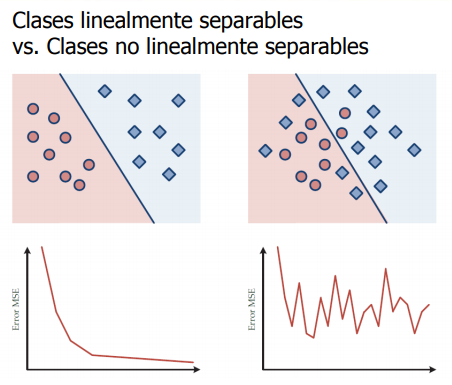
El algoritmo de aprendizaje del perceptrón garantiza encontrar un conjunto de pesos que proporcione la respuesta correcta si tal conjunto existe.

El perceptrón es un modelo de clasificación lineal, por lo cual será capaz de clasificar correctamente los ejemplos de entrada siempre que las clases sean linealmente separables.



****

****

****

1. CONCLUSIONES

Un sistema experto puede emular la heurística humana sí

puede emular la heurística hasta el punto en el que la

obtención de nuevo conocimiento no involucre instintos

humanos. Los tópicos abordados serían la obtención de

conocimientos basados en la experiencia, un sistema de

aprendizaje y un motor de inferencia.

REFERENCIAS

[1] Sistemas Expertos. ECURED. (2018) Disponible en:

<https://www.ecured.cu/Sistemas_expertos>

[2] Sebastián Badaró, Leonardo Javier Ibañez y Martín Jorge

Agüero. “Sistemas Expertos: Fundamentos, Metodologías y

Aplicaciones”. (2013) Disponible en:

<https://www.palermo.edu/ingenieria/pdf2014/13/CyT_13_24.pdf>

[3] ARQUITECTURA DE UN SISTEMA EXPERTO.

(2011) Disponible en:

<https://maricelamaldonado.wordpress.com/2011/03/23/arquitectura-de-un-sistema-experto>

[4] APL. Disponible en:

<https://es.wikipedia.org/wiki/APL>

[5] Lógica Difusa. Disponible en:

<https://es.wikipedia.org/wiki/L%C3%B3gica_difusa>

[6] CLIPS. Disponible en:

<https://es.wikipedia.org/wiki/CLIPS>

[7] About the Developer (Gary Riley) Disponible En:

<http://www.clipsrules.net/AboutTheDeveloper.html>