UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5

"PROPUESTA DE TÉCNICAS DE APRENDIZAJE DE ELEMENTOS VIRTUALES"

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autores: Tamara Martínez Labaut

Yaima Antúnez Ojeda

Tutora: Ing. Yenifer del Valle Guevara

Asesor: Msc. Yuniesky Coca Bergolla

Ciudad de la Habana

Junio 2008

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores de este trabajo y autorizamos a la Universidad de las Ciencias Informáticas, a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Autores:

Tamara Martínez Labaut

_____

Yaima Antúnez Ojeda

_____

Tutora:

Yenifer del Valle Guevara

_____

DATOS DE CONTACTO

Tutora:

Nombre y Apellidos: Yenifer del Valle Guevara.

Institución: Universidad de las Ciencias Informáticas (UCI).

E-mail: ydelvalle@uci.cu

Graduada de la UCI en el año 2007, líder del proyecto de Realidad Virtual “Desarrollo de Elementos Virtuales Inteligentes” en la Universidad de las Ciencias Informáticas. Obtuvo el Sello Forjadores del Futuro en el 2007.

Asesor:

Nombre y Apellidos: Yuniesky Coca Bergolla

Institución: Universidad de las Ciencias Informáticas (UCI).

E-mail: ycoca@uci.cu

Graduado de Ciencias de la Computación en la Universidad Central de Las Villas en el año 2003. Profesor de la Universidad de las Ciencias Informáticas desde ese mismo año. Es Jefe de Dpto. de las asignaturas de la especialidad de la Facultad 5 “Entornos Virtuales”, es miembro del Grupo de Desarrollo de Elementos Virtuales Inteligentes. Obtuvo el Sello Forjadores del Futuro en el 2007. Defendió la maestría en el año 2007.

AGRADECIMIENTOS

A mi mamá, que siente todo lo que hago y me da fuerzas para continuar.
A mi papá, por exigirme tanto siempre para ser mejor.
A mi familia, por la preocupación y el apoyo durante estos cinco años.
A mis amigos, por cada instante a mi lado, por convertirse en la familia de la UCI.
A Ania, por acogerme como otra más de la casa.
A la decana Mayra, por sus consejos.
A Yenifer, por estar presente aprendiendo con nosotras.
A Coca, por guiarme ante cada dificultad en la tesis.
A Alexey, por su inmensa ayuda para llevar este trabajo adelante.
A todos los que en un momento determinado me brindaron su ayuda.
Gracias por estar….

Tamara

A mi mamá y a mi papá, por estar siempre conmigo y darme apoyo en todo momento.
A mi hermana, por no separarse de mí ni un instante en este curso.
A Luis Ángel, por estar en cada momento y darme fuerzas para seguir.
A mi familia, por la preocupación y ayuda de siempre.
A mi tutora Yenifer, por su ayuda incondicional.
A Alexey, por su paciencia con nosotras.
A mis amigos, profesores y a todas las personas que
me han ayudado y han estado conmigo durante la carrera.

Yaima

DEDICATORIA

A mis padres, que me dieron la vida y siempre han estado orgullosos de mis logros. Este es el mejor regalo que les puedo dar.

Tamara

A mis padres, mi hermana y mi novio para que se sientan muy orgullosos de mí. Especialmente a mi abuelo Diógenes cumpliendo su deseo de hacerme ingeniera.

Yaima

RESUMEN

Los Sistemas de Realidad Virtual han alcanzado gran auge en la actualidad, convirtiéndose en un importante campo de desarrollo e investigación. Para lograr un mayor nivel de realismo en estos sistemas, se usan técnicas de Inteligencia Artificial. En los proyectos que se realizan en la Facultad 5, es necesario alcanzar un nivel avanzado de inteligencia, pero se ve afectado por la poca utilización de estas técnicas de aprendizaje para los elementos virtuales.

En este trabajo se propone dar solución a ese problema. Para ello se hace una investigación sobre las técnicas de aprendizaje de Inteligencia Artificial, de las que se explican sus características y conceptos fundamentales. De algunas de las más aplicadas a sistemas de Realidad Virtual, se explica, además de su funcionamiento en detalles, cómo se desarrolla el aprendizaje en los elementos virtuales. Se hace una comparación entre ellas de acuerdo a las ventajas y limitaciones, llegando a la conclusión de cual sería más factible usar en un determinado caso de estudio, y se implementa su utilización. Finalmente se demuestra que las aplicaciones de realidad virtual, usando estas técnicas de aprendizaje adquieren mayor nivel de inteligencia y realismo.

PALABRAS CLAVES

Inteligencia Artificial, Realidad Virtual, técnicas, aprendizaje.

ÍNDICE

INTRODUCCIÓN

Si tuviéramos que definir la principal característica que nos separa del resto de los animales, seguramente responderíamos que es la capacidad de raciocinio. Esta capacidad nos ha permitido desarrollar una tecnología propia de tal manera que, en estos momentos, esta tecnología se orienta a descubrir su origen. ¿Cómo funciona el cerebro? ¿Se pueden construir modelos artificiales que lo emulen? ¿Se pueden desarrollar máquinas inteligentes? Todas estas preguntas han conducido a un rápido desarrollo de un campo multidisciplinar del conocimiento conocido como Inteligencia Artificial. La Inteligencia Artificial no es más que la ciencia que intenta la creación de programas para máquinas que imiten el comportamiento humano, podemos decir que se encarga de modelar la inteligencia en sistemas computacionales. La investigación en el campo de la Inteligencia Artificial se caracteriza por la producción de máquinas para la automatización de tareas que requieran un comportamiento inteligente. La Inteligencia Artificial se ha extendido a muchas áreas, algunas de sus aplicaciones son en la Lingüística Computacional, en la Minería de Datos (Data Mining), en Videojuegos, y en Mundos Virtuales.

El gran desarrollo científico obtenido en la actualidad en el campo de la informática ha estado generando una gran demanda de aplicaciones en todos los ámbitos de la sociedad, y el campo de la Realidad Virtual no está exento de esto. Un programa computacional que usa inteligencia artificial resuelve problemas en un dominio especializado que ordinariamente requiere experiencia humana, por eso se hace necesario lograr que estas aplicaciones de realidad virtual tengan un nivel de inteligencia y realismo cada vez más avanzado, para ello es necesario desarrollar técnicas de aprendizaje de los elementos virtuales en un determinado entorno.

Actualmente, son numerosos los trabajos que se realizan y publican en todas las partes del mundo, sobre todo, con el propósito de conocer más acerca del tema y de aprovechar las ventajas de su aplicación en las distintas esferas de la sociedad. Mediante estas investigaciones se ha podido determinar que existen numerosas técnicas para lograr desarrollar determinados comportamientos en los elementos de entornos virtuales.

Nuestra facultad trabaja en el área de los simuladores, y aún más reciente en los videojuegos, creándose así varios proyectos investigativos y productivos conformados por especialistas y estudiantes de diferentes años, pero la escasa aplicación de técnicas de aprendizaje avanzadas de Inteligencia Artificial en dichos proyectos han limitado el nivel de realismo e inteligencia de los mismos.

Es por ello que el **problema científico** se enfoca en:
¿Cómo lograr mayor nivel de realismo e inteligencia en las aplicaciones de Realidad Virtual de la Facultad 5?

El **objeto de estudio** se circunscribe en el proceso de desarrollo de Inteligencia Artificial en las aplicaciones de Realidad Virtual. Mientras que nuestro **campo de acción** se enmarca en la aplicación de las técnicas de aprendizaje a elementos virtuales.

Con el fin de resolver el problema científico se trazó el siguiente **objetivo**:
Proponer varias técnicas de aprendizaje de elementos virtuales que mejoren el nivel de realismo e inteligencia en las aplicaciones de Realidad Virtual.

Este trabajo está dirigido además, a defender la idea de que las aplicaciones de Realidad Virtual logran un mayor nivel de realismo e inteligencia al aplicar técnicas de aprendizaje para los elementos de un determinado entorno virtual.

Para darle solución a este problema, se desarrollan las siguientes **tareas investigativas**:

- Búsqueda bibliográfica de todo lo relacionado con las técnicas de aprendizaje de elementos virtuales.
- Estudio y análisis de las técnicas de aprendizaje de elementos virtuales.
- Explicar las características y el funcionamiento de cada técnica de aprendizaje de elementos virtuales.
- Búsqueda de las necesidades de los proyectos de la facultad para lograr un mayor realismo e inteligencia en los elementos virtuales.
- Realización de propuestas sobre cómo utilizar las técnicas de aprendizaje de elementos virtuales en los proyectos de Realidad Virtual de la facultad 5.
- Argumentación de las ventajas y desventajas y posibles resultados al aplicar nuestras propuestas.
- Implementación de una de las técnicas de aprendizaje de elementos virtuales que muestren cómo dar solución a algunos de los problemas encontrados.

Como resultado de este trabajo se pretende proponer la aplicación de técnicas de aprendizaje de elementos virtuales en las aplicaciones de realidad virtual de la facultad 5 con el fin de lograr mayor realismo e inteligencia de estos elementos.

Para darle cumplimiento a las tareas se guiará la investigación en los marcos de los métodos científicos de investigación “Teóricos y Empíricos”. Dentro de los métodos teóricos se utilizará el método “Analítico-sintético”, con éste se podrá analizar cada uno de los elementos de manera independiente, posibilitando una mayor capacidad de comprensión y de síntesis sobre los aspectos más importantes. Por su parte la utilización del método “Análisis histórico-lógico” brindará la posibilidad de analizar toda la evolución del problema que se estará estudiando. También utilizaremos el método “Inductivo-deductivo”.

Uno de los métodos empíricos que será referenciado es la “Observación” este método permite adquirir información necesaria y puede utilizarse en cualquiera de las fases de la investigación, además ofrece un gran acercamiento a la realidad y permite ver la posible solución del problema desde diferentes ángulos. También se empleará el método de la “Entrevista” para indagar sobre las necesidades de los proyectos productivos en la facultad.

El presente documento está estructurado de la siguiente manera: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones, referencias bibliográficas, bibliografía, glosario de términos y anexos.

En el *Capítulo 1: Fundamentación Teórica*, se hace un estudio donde se investiga acerca de la Inteligencia Artificial, sus características y aplicaciones, profundizando en su aplicación a la Realidad Virtual a través de las distintas técnicas de aprendizaje para los elementos de los entornos virtuales. De estas técnicas se explica su concepto, aplicaciones y su importancia en la Inteligencia Artificial.

En el *Capítulo 2: Descripción y características de las técnicas*, se plantean de las técnicas expuestas en el capítulo anterior, algunas de las más usadas en los videojuegos y demás aplicaciones de realidad virtual, de ellas se explica su descripción y características, su funcionamiento y los principales campos donde son usados, además se explica detalladamente cómo es el aprendizaje con cada una de estas técnicas.

En el *Capítulo 3: Descripción de la propuesta,* se hace una comparación entre las cuatro técnicas tratadas en el capítulo anterior en cuanto a las ventajas y desventajas de su uso, para finalmente seleccionar una de ellas con el objetivo de resolver un caso de estudio, además se hacen distintas propuestas de aplicación de estas técnicas en los proyectos productivos de la facultad que requieren de inteligencia.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

La Realidad Virtual es considerada como la manipulación de los sentidos humanos (siendo actualmente el tacto, la visión y la audición) por medio de entornos tridimensionales sintetizados por computadora (PC) en el que uno o varios participantes acoplados de manera adecuada al sistema de computación interactúan de manera rápida e intuitiva donde la computadora desaparece de la mente del usuario dejando como real el entorno generado por ella.

Es aquí donde juega un papel fundamental la Inteligencia Artificial (IA) aplicada a la Realidad Virtual, pues esta ciencia incluye características humanas tales como el aprendizaje, la adaptación, el razonamiento, la autocorrección, el mejoramiento implícito, y la percepción modular del mundo.

Precisamente sobre la Inteligencia Artificial y las técnicas de aprendizaje que propone para los elementos de un entorno virtual se hablará en este capítulo. Se definirán cada una de las técnicas a través de su concepto. Además tratará acerca de su aplicación en los distintos campos y su importancia.

1.1 La Inteligencia Artificial

Se puede decir a grandes rasgos que la IA es una ciencia que intenta la creación de programas para máquinas que imiten el comportamiento y la comprensión humana, que sea capaz de aprender, reconocer y pensar. Sencillamente, la IA busca imitar la inteligencia humana.

CARACTERÍSTICAS DE LA INTELIGENCIA ARTIFICIAL

1. Una característica fundamental que distingue a los métodos de Inteligencia Artificial de los métodos numéricos es el uso de símbolos no matemáticos, aunque no es suficiente para distinguirlo completamente.

2. El comportamiento de los programas no es descrito explícitamente por el algoritmo. La secuencia de pasos seguidos por el programa es influenciado por el problema particular presente. El programa especifica cómo encontrar la secuencia de pasos necesarios para resolver un problema dado (programa declarativo).

3. El razonamiento basado en el conocimiento implica que estos programas incorporan factores y relaciones del mundo real y del ámbito del conocimiento en que ellos operan. Los programas de IA pueden distinguir entre el programa de razonamiento o motor de inferencia y base de conocimientos dándole la capacidad de explicar discrepancias entre ellas.

4. Aplicabilidad a datos y problemas mal estructurados. Sin las técnicas de IA los programas no pueden trabajar con este tipo de problemas. Un ejemplo es la resolución de conflictos en tareas orientadas a metas como en planificación o el diagnóstico de tareas en un sistema del mundo real: con poca información, con una solución cercana y no necesariamente exacta.

TÉCNICAS DE LA INTELIGENCIA ARTIFICIAL

Para el desarrollo de las aplicaciones y sistemas que usen inteligencia artificial, es necesario el uso de distintas técnicas de aprendizaje, esta es la parte de la IA que se ocupa de conseguir que sistemas computarizados sean capaces de sintetizar el conocimiento necesario para desarrollar una tarea sin necesidad de interacción humana.

Una técnica de IA es aquella que se utiliza para lograr que un programa se comporte de forma inteligente, sin tener en cuenta la "forma de razonamiento" subyacente a los métodos que se apliquen para lograr ese comportamiento.

Cada vez más, las técnicas de IA, además de adquirir una base formal acorde con las necesidades de la materia, están dejando de ser "curiosidades académicas con mucho futuro pero poca aplicación en el presente", para pasar a ser uno de los motores que impulsan la industria de la computación, con inversiones crecientes año tras año. Estas técnicas han mostrado beneficios reales al ser aplicadas en diversos sectores empresariales, aportando ventajas competitivas, ya sea ahorrando tiempo o dinero, aumentando la eficiencia de algún proceso o aprovechando oportunidades aún no explotadas.

Técnicas como:

- Las redes neuronales.
- Los sistemas expertos.
- La lógica difusa y las redes bayesianas.
- La computación evolutiva, los algoritmos genéticos.

Han demostrado ser valiosas en tareas como:

- Minería de datos: Extracción de "conocimientos" a partir de bases de datos, identificación de patrones y correlaciones insospechadas en los datos, que nos dan poder predictivo.
- Soporte a la toma de decisiones, o Automatización de decisiones: Mediante reglas de decisión, inducidas a partir de decisiones pasadas o de entrevistas con expertos.
- Optimización: Búsqueda de soluciones, configuraciones, o diseños óptimos, en problemas con alta complejidad y cantidad de variables.
- Simulación: La simulación es diseñar y desarrollar un modelo computarizado de un sistema o proceso y conducir experimentalmente este modelo con el propósito de entender el comportamiento del sistema del mundo real o evaluar varias estrategias con las cuales pueda operar el sistema.

1.1.1 La Inteligencia Artificial aplicada a la Realidad Virtual

Desde hace algunos años, y cada vez con más interés, los entornos virtuales están comenzando a verse como un medio en el que pueden narrarse historias. Estos se basan generalmente en sistemas inteligentes capaces de dirigir automáticamente todo lo que ocurre en un entorno virtual, para adaptar la historia a los objetivos de la aplicación y a las características particulares de cada uno de los interactores.

Básicamente, la Realidad Virtual consiste en simular todas las posibles percepciones de una persona, como los gráficos para la vista, sonido, tacto e incluso sensaciones de aceleración o movimiento. Todas estas sensaciones diferentes deben ser presentadas al usuario de forma que se sienta inmerso en el universo generado por el ordenador, hasta el punto de dejar de percibir la realidad y ser engañado, sentirse transportado (al otro lado de la pantalla) como si de un universo nuevo se tratase.

Los entornos o mundos virtuales proporcionan un poderoso medio para el aprendizaje y el entrenamiento experimental, siendo casi ilimitado el conjunto de mundos que la gente puede explorar, períodos de tiempo que el usuario puede atravesar, situaciones que el usuario puede vivir, así como grados de interactividad que el usuario puede experimentar.

En este intento de simular las apariencias y la funcionalidad de la vida real, es inevitable llegar al punto de simular al propio ser humano. A menudo, los entornos virtuales incorporan agentes inteligentes con forma humana y varios grados de inteligencia, obteniendo lo que se llama Agentes Virtuales Inteligentes.

1.2 Aprendizaje Automático

Una de las ramas de la IA la constituye sin dudas el Aprendizaje Automático que tiene por objetivo desarrollar técnicas que permitan a las computadoras “aprender”. Concretamente se trata de crear programas que sean capaces de generalizar comportamientos a partir de una información que no está estructurada, suministrada en forma de ejemplos; o sea, es un proceso de inducción del conocimiento que se basa en el análisis de datos centrándose más en el estudio de la complejidad computacional de los problemas y está enfocada al diseño de las soluciones factibles a esos problemas.

Un ser inteligente es el que recoge información del mundo exterior, almacena experiencias y razona para resolver nuevas situaciones; entonces, se puede ver al Aprendizaje Automático como un intento de automatizar el comportamiento inteligente, que se caracteriza por no producir siempre los mismos resultados, pues según las circunstancias y factores objetivos y subjetivos, ante una misma situación, los seres inteligentes no toman la misma decisión.

Algunos sistemas de Aprendizaje Automático intentan eliminar toda necesidad de intuición o conocimiento experto de los procesos de análisis de datos, mientras que otros tratan de establecer un marco de colaboración entre el experto y la computadora, pero de todas formas, la intuición humana no puede ser reemplazada en su totalidad, pues el diseñador del sistema especifica la forma de representación de los datos y los métodos de manipulación y caracterización de los mismos.

Se puede decir que una máquina se considera inteligente cuando es capaz de:

a. Percibir visualmente los objetos que la rodean y reconocer sus formas.
b. “Entender” el lenguaje natural, hablado o escrito y producir respuestas en dicho lenguaje.
c. Elaborar actuaciones de acuerdo con las condiciones cambiantes del entorno llevándolas a cabo mediante los correspondientes elementos físicos.
d. Almacenar información y conocimientos que manipula a través de reglas y algoritmos para alcanzar soluciones a los problemas que plantea su funcionamiento.

Pero no siempre se persigue que las máquinas sigan lo más fielmente posible el modo de actuar del ser humano; en ocasiones, se prefiere que operen en función de una mayor eficacia respecto a la aplicación hacia la que se orientan.

Se pueden obtener en el Aprendizaje Automático tres tipos básicos de conocimientos, pero, aunque todos se efectúan en el proceso del aprendizaje, la importancia de cada tipo de conocimiento depende de las características de lo que se está tratando de aprender, estos son:

- Crecimiento. Es el que se adquiere de lo que nos rodea, el cual guarda la información en la memoria como si dejara huellas.
- Reestructuración. Al interpretar los conocimientos el individuo razona y genera nuevo conocimiento.
- Ajuste. Es el que se obtiene al generalizar varios conceptos o generando los propios.

1.2.1 Sistemas Expertos

Los Sistemas Expertos constituyen una técnica de aprendizaje de IA, de la cual son un campo preferente. Basan su funcionamiento en la simulación del razonamiento humano, que tiene para ellos un doble interés, primeramente el del análisis del razonamiento que seguiría un experto humano en la materia con el fin de codificarlo empleando un determinado lenguaje informático, por otro lado, la síntesis artificial, de tipo mecánico, de los razonamientos de tal manera que estos sean semejantes a los empleados por el experto humano en la resolución de la cuestión planteada.

Se puede decir que los Sistemas Expertos constituyen intermediarios entre un experto humano (médico, analista, empresario, etc.), que transmite sus conocimientos al sistema y el usuario de dicho sistema, que lo utiliza para resolver los problemas que se le plantean con la eficacia del especialista en la materia y que a la vez, puede aprender observando el comportamiento del sistema. Es decir, los Sistemas Expertos se pueden considerar simultáneamente como un medio de ejecución y transmisión del conocimiento.

Con los Sistemas Expertos se busca una mejor calidad y rapidez en las respuestas dando así lugar a una mejora de la productividad del experto.

Para que un Sistema Experto sea herramienta efectiva, los usuarios deben interactuar de una forma fácil, reuniendo dos capacidades para poder cumplirlo:

1. *Explicar sus razonamientos o base del conocimiento*: los Sistemas Expertos se deben realizar siguiendo ciertas reglas o pasos comprensibles de manera que se pueda generar la explicación para cada una de estas reglas, que a la vez se basan en hechos.

2. *Adquisición de nuevos conocimientos o integrador del sistema:* son mecanismos de razonamiento que sirven para modificar los conocimientos anteriores.

Asimismo, los Sistemas Expertos se definen mediante su arquitectura; obtienen, por lo tanto, una realidad palpable. Mientras que en las operaciones de programación clásicas se diferencia únicamente entre el propio programa y los datos, en el caso de los Sistemas Expertos se diferencian tres componentes principales. Son los siguientes:

- Base de conocimientos
- Base de hechos
- Motor de inferencia

La base de conocimientos aloja la totalidad de las informaciones específicas relativas al campo del saber deseado. Está escrita en un lenguaje específico de representación de los conocimientos que contiene y en el cual el experto puede definir su propio vocabulario técnico. A la inversa de lo que sucede en los programas clásicos, en la base de conocimientos las informaciones entran tal como llegan, ya que el orden no influye en los resultados obtenidos. Sucede así porque cada elemento de conocimiento es comprensible por sí mismo tomado de forma aislada y, por lo tanto, no es necesario referirse al contexto en el cual está inserto. La información se representa, generalmente, mediante reglas de producción o redes semánticas. Las reglas de producción constituyen el método más utilizado para construir bases de conocimientos en los sistemas expertos. Llamadas también implicaciones lógicas, su estructura es la siguiente: para unas ciertas causas, unos efectos; o, para determinadas condiciones, ciertas consecuencias. Junto a cada regla, se almacena también su porcentaje en forma de probabilidad. Éste indica, mediante un tanto por ciento, el grado de certeza de las consecuencias que se obtienen como resultado de la aplicación de la regla de producción. En cuanto a las redes semánticas, se trata de un método de construcción de bases de conocimientos en el cual los conocimientos se muestran mediante un grafo en el que los vértices representan los conceptos u objetos y las aristas indican las relaciones entre ellos.

Además el sistema dispone de la llamada base de hechos, que alberga los datos propios correspondientes a los problemas que se desea tratar con la ayuda del sistema. Asimismo, a pesar de ser la memoria de trabajo, la base de hechos puede desempeñar el papel de memoria auxiliar. La memoria de trabajo memoriza todos los resultados intermedios, permitiendo conservar el rastro de los razonamientos llevados a cabo. Puede, por eso, emplearse para explicar el origen de las informaciones deducidas por el sistema en el transcurso de una sesión de trabajo o para llevar a cabo

la descripción del comportamiento del propio sistema experto. Al principio del período de trabajo, la base de hechos dispone únicamente de los datos que le ha introducido el usuario del sistema, pero, a medida que va actuando el motor de inferencias, contiene las cadenas de inducciones y deducciones que el sistema forma al aplicar las reglas para obtener las conclusiones buscadas.

El último elemento, el motor de inferencias, es un programa que, mediante el empleo de los conocimientos puede resolver el problema que está especificado. Lo resuelve gracias a los datos que contiene la base de hechos del sistema experto. Por regla general, el tipo de reglas que forman la base de conocimientos es tal que, si A es válido, puede deducirse B como conclusión. En este caso, la tarea que lleva a cabo el motor de inferencias es la de seleccionar, validar y activar algunas reglas que permiten obtener finalmente la solución correspondiente al problema planteado. El sistema experto establecido se compone, por lo tanto, de dos tipos bien diferenciados de elementos, los propios del campo de los expertos relacionados con el problema concreto (es decir, la base de conocimientos y la base de hechos) y el que se puede aplicar de forma general a una gran variedad de problemas de diversos campos (como el caso del motor de inferencias). Sin embargo, el motor de inferencias no es un mecanismo universal de deducción, ya que hay dos tipos diversos: los que emplean el razonamiento aproximativo (para el cual el resultado puede ser erróneo) y aquellos que emplean un tipo de razonamiento capaz de obtener un resultado (si llegan a él), con toda seguridad, verdadero.

Sin embargo, a pesar de no existir una metodología generalmente aceptada en cuanto a la concepción de los sistemas expertos, se admite por regla general un esquema que consta de tres fases. En la primera fase, la discusión con el experto o los expertos humanos en la cual se intenta, por un lado, delimitar el problema a resolver y, por el otro, los modos de razonamiento que se emplearán para su solución. La segunda fase comprende el desglose de la expresión del conocimiento y la determinación del motor de inferencias adecuado a ella. Por último, la tercera etapa, corresponde a la creación de la base de conocimientos (en colaboración con los expertos humanos), así como a la comprobación y ajuste del funcionamiento del sistema experto mediante el empleo de ejemplos.

Existen varios tipos de Sistemas Expertos, entre los que se destacan:

- Basados en reglas.
- Basados en casos o CBR (Case Based Reasoning).
- Redes bayesianas.

En cada uno de ellos, la solución a un problema planteado se obtiene aplicando reglas heurísticas apoyadas generalmente en lógica difusa para su evaluación y aplicación, aplicando el razonamiento basado en casos, donde la solución a un problema similar planteado con anterioridad se adapta al nuevo problema o aplicando redes bayesianas, basadas en estadística y el teorema de Bayes.

Los Sistemas Expertos poseen ciertas características que los hacen muy útiles, entre ellas podemos encontrar las siguientes:

- Pueden explicar su razonamiento o decisiones sugeridas: La capacidad de explicar cómo se llegó a una decisión o solución.
- Pueden mostrar un comportamiento "inteligente": Al examinar un grupo de datos, un sistema experto puede proponer nuevas ideas o métodos para la solución del problema, o proporcionar asesoramiento en el trabajo para los trabajadores.
- Pueden obtener conclusiones de relaciones complejas: Evaluar relaciones complejas para llegar a conclusiones y solucionar problemas, por ejemplo: un Sistema Experto propuesto trabajará con un sistema de fabricación flexible para determinar la mejor utilización de las herramientas, y otro sugerirá los mejores procedimientos de control de calidad.
- Pueden proporcionar conocimientos acumulados: Se puede usar para capturar conocimientos de humanos que de lo contrario podrían perderse.
- Pueden hacer frente a la incertidumbre: Tienen capacidad para enfrentar conocimientos incompletos o inexactos en su totalidad mediante el uso de las probabilidades, las estadísticas y las heurísticas.

Pero también existen algunas características que limitan su utilidad y que debemos tener presentes:

- No se han usado o probado en forma extensa.
- Algunos sistemas expertos son difíciles de controlar y usar.
- No pueden enfrentar con facilidad conocimientos "mixtos".
- Sólo manejan áreas precisas del conocimiento.
- Tienen una capacidad limitada de aprendizaje.
- No poseen sentido común.
- Posibilidad de error.
- Dificultad de mantenimiento.

Los Sistemas Expertos realizan innumerables tareas entre las cuales podemos encontrar la monitorización que consiste en la comparación continua de los valores de las señales o datos de

entrada y unos valores que actúan como criterios de normalidad o estándares, el diseño que es el proceso de especificar una descripción de un artefacto que satisface varias características desde un número de fuentes de conocimiento, la planificación que es la realización de planes o secuencias de acciones y es un caso particular de la simulación, el control que permite conducir o guiar un proceso o sistema, la simulación que es una técnica para crear modelos basados en hechos, observaciones e interpretaciones, sobre la computadora, a fin de estudiar el comportamiento de los mismos mediante la observación de las salidas para un conjunto de entradas y la instrucción que permite realizar un seguimiento del proceso de aprendizaje para el cual el sistema detecta errores e identifica el remedio adecuado, es decir, desarrolla un plan de enseñanza que facilita el proceso de aprendizaje y la corrección de errores.

Debido a las funcionalidades de los Sistemas Expertos en la actualidad se utilizan con innumerables fines, es por eso que hoy podemos encontrar este tipo de sistemas en acciones como otorgar créditos, administrar y recuperar información, evaluar el desempeño de empleados, realizar análisis de préstamos, hacer informes de mercadotecnia, detectar virus informáticos, también se emplean en los departamentos de ayuda y asistencia de diferentes aplicaciones posibilitando la reparación y el mantenimiento de las mismas, en la instalación de hospitales y en programas específicos de carácter médico y además podemos encontrar estos sistemas incorporados a productos de robótica.

APLICACIONES

Los sistemas expertos son programas basados en una forma de programación que tiene la capacidad de aprender nuevos datos o relaciones durante su ejecución. Por ejemplo, fue diseñado por el Maestro Francisco Javier Sierra Vázquez en Agosto del 2001 un sistema experto para enseñar álgebra a los alumnos, a través de un juego, el cual puede detectar que el alumno llega al resultado correcto pero no reduce las expresiones a su forma más simple, entonces el programa establece una relación con lecciones sobre reducción de expresiones o hace sugerencias al estudiante para que éste descubra cómo puede reducir sus resultados.

FISHING EXPERT V4.0

Es un juego real, que contiene patrones y técnicas de pesca como también registros de pesca. Recomendado como el mejor por los profesionales en la revista BassMasters. El usuario selecciona las condiciones en las que pesca y el programa le proporciona 400 métodos de eficacia probada para

pescar, librería, análisis condicional, recetas, bases de datos de perca, ojizarco, tilapia, trucha y control de barcos con la compra; ideal para entrenar pescadores.

El sistema se basa en investigaciones recientes sobre el comportamiento de los peces y se involucra en la búsqueda de escritos recientes en el área, para llegar a resultados de posibles condiciones de pesca y los mejores métodos para estas. La base de datos contiene cientos de posibilidades las cuales el sistema experto enlaza automáticamente. (1)

DELTA

Los sistemas expertos también pueden proporcionar conocimientos acumulados, se pueden usar para capturar conocimientos de humanos que de lo contrario podrían perderse. Ejemplo es el Sistema Experto denominado DELTA (Diesel Electronic Locomotive Troble-shooting Aid), el cual no es más que un simulador desarrollado para conservar el conocimiento de David Smith, único ingeniero competente para manejar muchas reparaciones extremadamente técnicas de esas máquinas, cuando llegó el momento de su jubilación.

XCON

Es un Sistema Experto para configuraciones de ordenadores desarrollado por la Digital Equipment Corporation. Según los deseos individuales del cliente se configuran redes de ordenadores VAX. Las funciones generales de este sistema son las siguientes: el cliente solicita los componentes y el sistema, si son compatibles y completos, los conjuga, y muestra como quedaría, de lo contrario le ofrece los más recomendables y necesarios. XCON es capaz de comprobar y completar los pedidos entrantes mucho más rápido y mejor que las personas encargadas de esa labor. El programa simula toda la configuración y si el cliente está de acuerdo, la realiza.

El sistema experto tiene una gran cantidad de casos y reglas definidos, y realiza tareas de gran complejidad. Ya que el abanico de productos que se ofrecen en el mercado es muy amplio, la configuración completa y correcta de un sistema de estas características es un problema de gran envergadura. (2)

1.2.2 Sistemas basados en Reglas

Los Sistemas basados en Reglas trabajan mediante la aplicación de reglas, comparación de resultados y aplicación de las nuevas reglas basadas en una situación modificada. También pueden trabajar por inferencia lógica dirigida, bien empezando con una evidencia inicial en una determinada situación y

dirigiéndose hacia la obtención de una solución, o bien con hipótesis sobre las posibles soluciones y volviendo hacia atrás para encontrar una evidencia existente (o una deducción de una evidencia existente) que apoye una hipótesis en particular.

Es necesario dejar claro que un Sistema basado en Reglas, aunque es un tipo de Sistema Experto no sustituye al experto en sí. En la siguiente tabla se muestran las diferencias entre un Sistema basado en Reglas y un experto humano:

	Sistema Basado en Reglas	**Experto humano**
Conocimiento	Adquirido	Adquirido + Innato
Adquisición del conocimiento	Teórico	Teórico + Práctico
Campo	Único	Múltiples
Explicación	Siempre	A veces
Limitación de capacidad	Sí	Sí, no valuable
Reproducible	Sí, idéntico	No
Vida	Infinita	Finita

Tabla 1. Diferencias entre un Sistema Experto y un Experto Humano.

Los Sistemas basados en Reglas son una expresión de los sistemas basados en el conocimiento. Su función consiste en aportar soluciones a problemas o nuevo conocimiento.

Aunque no existe una estructura común, la mayoría de los Sistemas basados en Reglas tienen unos componentes básicos:

- Base de Conocimientos. Contiene el conocimiento de los hechos y de las experiencias de los expertos en un dominio determinado.
- Base de datos. Contiene los datos iniciales del problema actual y los derivados.
- Mecanismo de Inferencia. Puede simular la estrategia de solución de un experto, emplea los distintos razonamientos para deducir nuevo conocimiento.
- Componente Explicativo. Explica al usuario la estrategia de solución encontrada y el por qué de las decisiones tomadas.

- Interfaz de Usuario. Sirve para que éste pueda realizar una consulta en un lenguaje lo más natural posible.
- Componente de Adquisición. Ofrece ayuda a la estructuración e implementación del conocimiento en la base de conocimientos.

El proceso de razonamiento de un Sistema basado en Reglas es una progresión de un conjunto de datos de partida hacia una respuesta, solución o conclusión. Teniendo en cuenta el problema que se esté tratando se debe escoger uno de los siguientes métodos:

- Razonamiento hacia delante. Se parte de los datos conocidos y se avanza de forma natural hacia el objetivo. Resulta útil cuando los datos de partida son pocos y/o existen muchas posibles conclusiones.
- Razonamiento hacia atrás. Se parte de un objetivo deseado y decide si los hechos iniciales le permiten derivar un valor para la conclusión, normalmente los hechos iniciales son nulos pues se suele partir de una base de conocimientos vacía.

Un Sistema Basado en Reglas es muy eficaz cuando tiene que analizar una gran cantidad de información, interpretándola y proporcionando una recomendación a partir de la misma, son buenos para predecir resultados futuros a partir del conocimiento que tienen.

El diseño de un Sistema Basado en Reglas requiere una enorme cantidad de conocimientos debido a que hay que tener en cuenta muchas especificaciones y restricciones. En este caso, el sistema experto ayuda al diseñador a completar el diseño de forma competente y dentro de los límites de costes y de tiempo. Por último, un Sistema Basado en Reglas puede evaluar el nivel de conocimientos y comprensión de un estudiante, y ajustar el proceso de aprendizaje de acuerdo con sus necesidades.

Los Sistemas basados en Reglas tienen ventajas y desventajas cuando se comparan con otras soluciones como el software convencional o los humanos. Entre sus ventajas se tiene la fácil modificación pues el conocimiento es explícito y accesible, la consistencia en las respuestas, la gran accesibilidad ya que trabajan las 24 horas todos los días, ellos poseen capacidad para adquirir nuevo conocimiento y perfeccionar el que poseen, además explican las soluciones y tiene capacidad de expresar amplio conocimiento. Pero en ocasiones pueden traer problemas tales como el encadenamiento infinito, la adición de nuevo y contradictorio conocimiento así como la modificación de las reglas ya existentes.

APLICACIONES

MYCIN

Es un Sistema Experto para la realización de diagnósticos, iniciado por Ed Feigenbaum y posteriormente desarrollados por E. Shortliffe y sus colaboradores. Su función es la de un simulador, que aconseja a los médicos en la investigación y determinación de diagnósticos en el campo de las enfermedades infecciosas de la sangre. A grandes rasgos, el sistema, al ser consultado por el médico, solicita primero datos generales sobre el paciente: nombre, edad, síntomas, etc. Una vez conocida esta información por parte del sistema, el Sistema Experto plantea unas hipótesis. Para poder verificarlas comprueba primero la exactitud de las premisas de la regla, esto se realiza mediante una búsqueda de enunciados correspondientes en la base de conocimientos. Estos enunciados pueden, a su vez, estar de nuevo en la parte de consulta de otra regla. También lo realiza mediante determinadas preguntas al usuario. MYCIN fue el primer Sistema basado en Reglas que se ocupó del tratamiento de la incertidumbre; sus creadores desarrollaron un modelo que, aunque inspirado en la teoría de la probabilidad, se apartaba notablemente de ella.

BUSCAMINAS

Para el clásico juego del buscaminas se implementó con Jess, una estrategia inteligente para resolver tableros del buscaminas de diferentes niveles de dificultad. Para ello se utilizó un sistema experto basado en reglas, donde una aplicación Java se encargaba de crear y almacenar toda la información del tablero y de hacer públicas una serie de funciones al sistema de reglas, estas reglas coinciden con las estrategias que puede realizar un jugador humano sobre una interfaz de usuario típico. Se llegó a la conclusión de que la aproximación basada en reglas es buena para resolver el problema, ya que, el experto humano utiliza este tipo de razonamiento al jugar. Aunque también se planteó que se podían emplear otras técnicas como el razonamiento basado en casos o razonamiento basado en el uso de probabilidades. (3)

WUMPUS

Un trabajo interesante consistió en la programación mediante reglas del control del jugador (agente) en el mundo artificial del *Wumpus* descrito en el capítulo 7 del libro "*Artificial Intelligence:A Modern Approach*". (4)

El mundo del Wumpus es una cuadrícula en la que cada casilla puede contener una bolsa de oro, un agujero o el Wumpus (el enemigo al que hay que evitar). El agente, basándose en su percepción del

entorno y en ciertas reglas de razonamiento, debe encontrar (deducir) un camino seguro hacia el oro, evitando los agujeros y el Wumpus. Una vez recogido, debe volver a la salida. Las reglas del juego pueden consultarse en (4).

Para esta implementación se podía usar cualquiera de los dos motores de inferencia más utilizados: CLIPS (5) y Jess (6), se usó Jess, y se definió la sintaxis del sistema de reglas. El trabajo con el mundo de Wumpus puso de manifiesto la necesidad de adquisición del conocimiento y de evaluación de la capacidad de resolución de problemas de un sistema basado en reglas.

1.2.3 Sistemas basados en Casos

"Un Razonador Basado en Casos resuelve problemas nuevos mediante la adaptación de soluciones previas usadas para resolver problemas similares." (7)

El Razonamiento Basado en Casos es una rama de la IA que se preocupa por el estudio de los mecanismos mentales necesarios para repetir lo que se ha hecho o vivido con anterioridad, ya sea por uno mismo, o ya sea por casos concretos recopilados en la bibliografía o en la sabiduría popular. Los diversos casos son del tipo "Si X, entonces Y" con algunas adaptaciones y críticas según las experiencias previas en el resultado de cada una de dichas reglas. (8)

La estructura de este tipo de sistemas se divide en cuatro pasos:

1. Recuperar los casos más parecidos donde un nuevo problema se aparea con casos similares guardados en la base de casos.
2. Reutiliza la solución propuesta en los casos para tratar de resolver el problema.
3. Revisar la solución propuesta.
4. Almacenar la nueva solución como parte de un nuevo caso.

Un Sistema Basado en Casos es uno de los tipos de los Sistemas Basados en Conocimiento. Ellos apoyan sus predicciones en ejemplos (casos) que se almacenan en la fase de aprendizaje. Una función de distancia o de semejanza determina los casos más semejantes al nuevo problema y las soluciones de los casos recuperados se adaptan para obtener una solución. Durante este proceso se presentan dos problemas fundamentales: lograr una representación de la memoria de casos que permita una recuperación eficiente y considerar la incertidumbre presente en el conocimiento para mejorar la eficiencia en el proceso de adaptación.

El Razonamiento Basado en Casos representa un nuevo método para resolver problemas no estructurados en los cuales el razonamiento se realiza a partir de una memoria asociativa que usa un algoritmo para determinar una medida de semejanza entre dos objetos. En este paradigma la base del comportamiento inteligente radica en recordar situaciones existentes en el pasado. Debe destacarse que es una técnica en la cual la memoria se sitúa como fundamento de la IA y más concretamente de los sistemas basados en el conocimiento.

Tipos de Sistemas Basados en Casos acorde a sus dos Componentes Principales:

1. Interpretativo. Consta de una base de casos que contiene una descripción de los problemas resueltos o no con anterioridad. Toma un problema y sus soluciones y provee argumentos para justificar sus soluciones.
2. Solucionador de Problemas. Consta de dos módulos el razonador y el asignador. Deriva soluciones para nuevos problemas utilizando soluciones antiguas para problemas similares, como punto de comienzo.

Entre las ventajas que poseen los Sistemas basados en Casos podemos encontrar que no requiere de un modelo explícito del dominio y el proceso de extracción se reduce a juntar casos históricos, su construcción consiste en identificar atributos relevantes con los cuales describir los casos, permiten dar explicaciones, usan técnicas de base de datos para manipular grandes volúmenes de información, pueden aprender adquiriendo nuevo conocimiento como casos haciendo su mantenimiento más fácil y pueden ir creciendo reflejando la experiencia acumulada.

A pesar del éxito de los sistemas basados en conocimiento, existen varios problemas:

- El proceso de extracción de conocimiento es difícil.
- Su construcción requiere de habilidades especiales.
- Normalmente son lentos e incapaces de procesar grandes cantidades de información.
- Son difíciles de mantener.

APLICACIONES

COACH (Cognitive Adaptative Computer Help):
Permite crear ayuda personalizada al usuario. Es un observador de las acciones del usuario que está aprendiendo a operar un ambiente, y en base a ellas construye un modelo adaptativo del usuario. Si bien el concepto general es aplicable para áreas diversas tales como las Interfaces Inteligentes y el soporte técnico, en particular es de interés para este trabajo ya que los dominios de prueba que

modeló corresponden al de un lenguaje y un entorno de programación (LISP y UNIX, respectivamente); así como por la prueba de adaptabilidad a distintos dominios en que probó ser efectivo.

Una de las contribuciones importantes de Coach consiste en la descripción de diversos modelos de usuarios, representados por medio de frames adaptativos; y el modelado cognitivo de variables tales como la experiencia, la latencia del conocimiento. Este sistema va aprendiendo de la experiencia adquirida en cada caso de asesoramiento a distintos usuarios. (2)

EL SISTEMA EXPERTO CASNET.
Este es un sistema basado en casos utilizado en un simulador para la medicina. Su objetivo era ayudar a los médicos en el diagnóstico y el tratamiento del glaucoma (enfermedad ocular). Los programas de diagnóstico por ordenador anteriores estaban basados principalmente en métodos estadísticos y tenían deficiencias en cuanto al razonamiento temporal (eran incapaces de seguir la evolución de la enfermedad), en cuanto al diagnóstico múltiple (la mayor parte de ellos suponía que existía un único diagnóstico que excluía a los demás) y en cuanto a la explicación de sus conclusiones (no podían justificar cómo y por qué habían obtenido los resultados).

Uno de los rasgos característicos de CASNET es la estructuración de los nodos en tres niveles:

- Observaciones: incluye los síntomas (lo que el enfermo siente), los signos (lo que el médico observa) y los resultados de las pruebas de laboratorio.
- Estados patofisiológicos: son las alteraciones que se producen en el funcionamiento de un órgano (en este caso, el ojo).
- Estados de enfermedad: dentro de este nivel, las enfermedades se encuentran clasificadas en un árbol taxonómico; los nodos inferiores corresponden a especificaciones de los nodos superiores. Cada caso permite definir y reconocer la enfermedad y los distintos tratamientos, así como diagnosticar nuevos tratamientos, en dependencia de los síntomas.

1.2.4 Redes Bayesianas

Una Red Bayesiana es un modelo probabilístico multivariado que relaciona un conjunto de variables aleatorias mediante un grafo dirigido que indica explícitamente influencia causal. Gracias a su motor de actualización de probabilidades, el Teorema de Bayes. Las redes bayesianas son una herramienta extremadamente útil en la estimación de probabilidades ante nuevas evidencias.

También se puede decir que una Red Bayesiana es un grafo acíclico dirigido en el que cada nodo representa una variable y cada arco una dependencia probabilística, en la cual se especifica la probabilidad condicional de cada variable dados sus padres. La variable a la que apunta el arco es dependiente (causa-efecto) de la que está en el origen de éste. La topología o estructura de la red nos da información sobre las dependencias probabilísticas entre las variables pero también sobre las independencias condicionales de una variable (o conjunto de variables) dada otra variable(s). Dichas independencias, simplifican la representación del conocimiento (menos parámetros) y el razonamiento (propagación de las probabilidades).

El diseño de una Red Bayesiana comienza con la definición de su grafo. Luego es necesario establecer las probabilidades condicionales de cada nodo dados los valores de sus padres. Para cada nodo sin padres, únicamente hay que definir sus probabilidades a priori. La inferencia consiste en fijar los valores de las variables observadas y calcular la probabilidad a posteriori de las variables no observadas. En el caso general, es necesario asignar a cada nodo un conjunto de probabilidades condicionales que crece exponencialmente con el número de padres del nodo. Este hecho dificulta el proceso de adquisición de los parámetros de la red, su almacenamiento y la propagación de la evidencia.

También se dice que las Redes Bayesianas constituyen una manera práctica y compacta de representar el conocimiento incierto, que permiten establecer razonamientos basados en la teoría de la probabilidad donde cada nodo está asociado a una variable aleatoria que puede tomar valores dentro de un rango discreto o continuo, siendo estos valores exclusivos y exhaustivos para cada variable de la red.

Es importante observar que la topología o estructura de la red no sólo proporciona información sobre las dependencias probabilísticas entre las variables, sino también sobre las independencias condicionales de una variable o conjunto de ellas dada otra u otras variables. Cada variable es independiente de las variables que no son descendientes suyas en el grafo, dado el estado de sus variables padre.

La inclusión de las relaciones de independencia en la propia estructura del grafo hace de las redes bayesianas una buena herramienta para representar conocimiento de forma compacta (se reduce el número de parámetros necesarios). Además, proporcionan métodos flexibles de razonamiento

basados en la propagación de las probabilidades a lo largo de la red de acuerdo con las leyes de la teoría de la probabilidad. (9)

Entre las características que poseen las Redes Bayesianas se pueden destacar que estas permiten aprender sobre relaciones de dependencia y causalidad, también posibilitan la combinación de conocimiento con datos y además, evitan el sobreajuste de los datos pudiendo manejar bases de datos inconclusas. Las Redes Bayesianas sirven para evaluar sistemáticamente el riesgo asociado a la tarea de asignar recursos. Una Red Bayesiana se puede utilizar, por ejemplo, para la cuantificación del riesgo relacionado con el desarrollo de un producto.

APLICACIONES

La teoría de los juegos poco cooperativos provee un framework normativo para analizar las interacciones estratégicas de los agentes. En algunos juegos poco cooperativos los agentes pueden estar faltos de información acerca de sus adversarios por lo que se hace necesario tomar decisiones en adversarios con información incierta. En (10) se propone la utilización de redes bayesianas para modelar la creencia en el adversario en juegos estáticos con información incompleta. En esta investigación los autores utilizan las redes bayesianas para modelar la incertidumbre del agente sobre sus adversarios ya que esta incertidumbre puede actualizarse cuando algunos acontecimientos ocurren a través de las redes.

El punto de la búsqueda en un árbol de juego es aislar de errores en la función de evaluación uno mismo. El acercamiento estándar es crear un árbol de gran anchura tan profundo como el tiempo permita y entonces evaluar el árbol como si las evaluaciones de las hojas fueran exactas. El algoritmo alfa-beta implementa esto con gran eficiencia computacional, siendo este acercamiento muy efectivo en diversos juegos. Baum y Smith (11) realizan un acercamiento bayesiano para la relevancia de jugar juegos, formando un Modelo bayesiano de incertidumbre. Los autores adoptan una función de evaluación que devuelve una distribución de probabilidad estimando la probabilidad de varios errores en evaluar cada posición. Estas estimaciones se obtiene entrenando datos, de esta forma usan información adicional en cada hoja no disponible para el acercamiento estándar. Su algoritmo se enfoca en líneas relevantes, en cuales puede, en principio, crear un árbol varias veces tan profundo como alfa-beta dada una cantidad de tiempo. Ellos prueban su acercamiento en una colección de juegos tales como Othello, Kalah, Warri y otros, mostrando que su algoritmo completo de búsqueda

juega mejor que un programa alfa-beta comparable para Othello altamente ordenado por rango, directamente aún cuando el programa alfa-beta recibió probabilidades de tiempo considerables.

En muchos contextos, los jugadores interactúan solamente con un subconjunto de la población entera, un ejemplo de esto son que interactúan en una red; los jugadores tienen una información incompleta de la estructura de la red, ellos tienen una prioridad y además conocen el número de conexiones que tienen. En su estudio sobre la convergencia de creencias en juegos con Redes bayesianas, Willemien Kets (12) coloca cuáles jugadores están localizados en una red y juegan un juego fijo con sus vecinos. Estudia la sensibilidad de las predicciones teóricas de juego para la especificación de las creencias de los jugadores mostrando que dos con prioridades similares están cerca en un sentido estratégico si y solo si le asignan probabilidades similares a todos los eventos locales, ejemplo para todos los eventos implicando los tipos de un jugador y sus vecinos, esto significa que para explorar completamente en orden el rango de posibles resultados estratégicos sería suficiente variar la distribución del tipo y la correlación entre los tipos de los jugadores.

En el libro AI for Game Developers (13), los autores proponen el uso diversas técnicas para desarrollar juegos con Inteligencia Artificial, entre ellas ejemplifican como se podrían usar las redes bayesianas en los juegos de diversas maneras. Específicamente, muestran cómo usar estas técnicas para permitirles a los NPCs (Non Player Characters) tomar decisiones cuando los estados del mundo de juego son inciertos. También muestran como un simple modelo bayesiano permite a los personajes controlados por la computadora adaptarse a cambiar situaciones. Además, se dan ejemplos para utilizar las redes bayesianas en tipos de juegos concretos que incluyen los de peleas de Kung Fu donde un jugador debe tratar de predecir los movimientos de su oponente controlado para dar una mejor respuesta, en juegos de aventuras donde los personajes pueden realizar diferentes acciones ante distintas situaciones del entorno de juego, también en juegos de guerra donde los soldados deben ajustar las estrategias defensivas y ofensivas para combatir contra los soldados controlados por la computadora a medida que aprende de batallas pasadas, todos ellos empleando las redes bayesianas para tomar las mejores decisiones.

JUEGOS DE AVENTURA

En los juegos de aventuras se presentan miles de posiciones y acciones posibles a realizar por parte de sus jugadores. Albrecht, Zukerman y Nicholson (14) proponen un Modelo Bayesiano con el fin de

lograr un acercamiento a un plan de reconocimiento de entrada para juegos de aventuras que representen características del dominio necesarias para identificar los planes de usuarios y las metas. Este modelo se aplica específicamente a un Juego de Aventuras en Calabozos para múltiples usuarios. Proponen varias estructuras de redes que representan la relación en el dominio para varias extensiones y comparan su poder para predecir la meta actual de un usuario, la siguiente acción y la siguiente posición. Las distribuciones condicionales de probabilidad para cada red son aprendidas durante una fase de entrenamiento, lo cual dinámicamente construye estas probabilidades para observar el comportamiento del usuario, este acercamiento permite usar datos incompletos y escasos durante el entrenamiento y la experimentación. Luego aplican abstracción simple y técnicas de aprendizaje para apresurar la marcha del funcionamiento de las redes bayesianas más alentadoras sin cambios significativos en la exactitud de las predicciones de la meta. Sus resultados experimentales en el dominio aplicativo mostraron un alto grado de exactitud en las predicciones.

1.2.5 Algoritmos Genéticos

Los Algoritmos Genéticos (AG) se inspiran en la evolución de los organismos vivos para hallar soluciones óptimas a problemas difíciles, pues de la misma forma en que los seres vivos compiten de manera que los más aptos sobreviven y pasan sus cualidades a sus descendientes, así varias soluciones aproximadas compiten y evolucionan para producir soluciones óptimas a los mencionados problemas.

Según Watson y Pollack (15) (16) los Algoritmos Genéticos son la técnica informática que se basa en el funcionamiento de la evolución natural para resolver determinadas situaciones, buscando, combinando y optimizando soluciones y produciendo innovación. Operan mediante simulación por una computadora, combinando aleatoriamente soluciones posibles a un problema, para producir diversidad, y seleccionando las más aptas, que se vuelven a recombinar y evaluar en sucesivas generaciones. La aptitud media de cada generación va creciendo, y al final, se obtiene una solución si no óptima, convergente con lo óptimo.

Versiones más complejas de algoritmos genéticos generan un ciclo iterativo que directamente toma a la especie (el total de los ejemplares) y crea una nueva generación que reemplaza a la antigua una cantidad de veces determinada por su propio diseño. Una de sus características principales es la de ir perfeccionando su propia heurística en el proceso de ejecución, por lo que no requiere largos períodos de entrenamiento especializado por parte del ser humano. (17)

Se definirán entonces como funciones principales de los Algoritmos Genéticos:

- Hallar de qué parámetros depende el problema.
- Codificarlos en un cromosoma.
- Aplicar los métodos de la evolución: selección, cruzamiento y mutación.

El poder de los Algoritmos Genéticos proviene del hecho de que se trata de una técnica robusta, y pueden tratar con éxito una gran variedad de problemas provenientes de diferentes áreas, incluyendo aquellos en los que otros métodos encuentran dificultades. Si bien no se garantiza que el Algoritmo Genético encuentre la solución óptima del problema, existe evidencia empírica de que se encuentran soluciones de un nivel aceptable, en un tiempo competitivo con el resto de algoritmos de optimización combinatoria. El gran campo de aplicación de los Algoritmos Genéticos se relaciona con aquellos problemas para los cuales no existen técnicas especializadas. Incluso en el caso en que dichas técnicas existan, y funcionen bien, pueden efectuarse mejoras de las mismas hibridándolas con los Algoritmos Genéticos.

Debido a sus características en la actualidad los Algoritmos Genéticos tiene grandes aplicaciones en diversas ramas como la Ingeniería Aeroespacial, en la Astronomía y la Astrofísica, la Acústica, en la Ingeniería Eléctrica, Geofísica, en las Matemáticas y Algoritmia, para reconocer patrones y en la exploración de datos, en el Diseño, la generación de melodías y se han usado hasta en el Ejército para el cumplimiento de la ley. Pero sin dudas en el campo de los videojuegos su aplicación ha alcanzado una posición importante.

APLICACIONES

En el libro AI for Game Developers (13) los autores proponen el uso diversas técnicas para desarrollar juegos con Inteligencia Artificial. Específicamente ejemplifican como se podrían usar los algoritmos genéticos en juegos de roles de diversas maneras. Los autores plantean que en un juego de roles típicamente estos se clasifican en categorías por el carácter de sus habilidades y le asignan un valor determinado a cada una. Un gnomo podría tener la oportunidad de dividir 100 puntos sobre varios atributos, como la fuerza, la habilidad mágica, la agilidad, y una resistencia mágica. En lugar de asignarle los mismos valores a cada gnomo en la población, podría ser mejor usar alguna diversidad. Por ejemplo, algunos serían físicamente más fuertes mientras otros tendrían una mayor resistencia para la magia. Variar la distribución de los valores y luego ordenar por rango la adaptabilidad de la

población ayudaría a determinar el mejor balance de dicha distribución. Las generaciones sucesivas de gnomos podrían evolucionar en adversarios más desafiantes para los jugadores.

DAMAS

En el año 2000, Chellapilla y Fogel (18) dieron una de las demostraciones más interesantes de la increíble potencia de los algoritmos genéticos. Ellos utilizaron un AG para evolucionar redes neuronales que pudieran jugar a las damas. Los autores afirman que una de las mayores dificultades en este tipo de problemas relacionados con estrategias es el problema de la asignación de crédito, en otras palabras ¿cómo escribir una función de aptitud? Se ha creído ampliamente que los criterios simples de ganar, perder o empatar no proporcionan la suficiente información para que un algoritmo genético imagine qué constituye el buen juego, en esta investigación los autores transforman esa suposición, aquí ellos dan solo las posiciones espaciales de las piezas en el tablero de ajedrez y el número total de piezas que posee cada lado, desarrollando un programa de damas capaz de jugar en un nivel competitivo como si fuera un experto humano, sin algún aporte inteligente previo en lo que se refiere a qué constituye un buen juego, incluso, a los individuos del algoritmo evolutivo no se les dieron los criterios para ganar ni los resultados de juegos anteriores.

LEMMINGS

Este juego es considerado como el antecesor de los juegos actuales de estrategia en tiempo real. Los personajes están basados en la creencia popular de que los lemmings se suicidan en masa en situaciones de peligro. El objetivo del juego es el de salvar a un número determinado de lemmings en cada nivel, para lo que se cuenta con ocho habilidades distintas que se pueden repartir en número limitado a cada lemming para lograr alcanzar el final de cada fase.

En (19) los autores hacen un estudio interesante donde emplean algoritmos genéticos para lograr navegar con éxito lemmings a través de mapas cada vez más difíciles, logrando que las poblaciones que heredaron genes de la mejor estrategia desarrollada del funcionamiento anterior se desarrollaran en el tiempo más corto.

AJEDREZ

El ajedrez es un juego complicado, de ahí que siempre los investigadores del campo de la IA estén en constante estudio para desarrollar aplicaciones de este juego que tengan un comportamiento cada vez más inteligente.

Aún en el final del juego, un error aparentemente menor puede cambiar el resultado del juego de una victoria o una derrota para un empate. En finales de rey y peón un simple error puede cambiar irrevocablemente la posición favorable mientras que en otros tipos de finales, numerosas maniobras inútiles pueden ser realizadas sin afectar el resultado del juego. Así, los finales de rey y peón necesitan ser cuidadosamente examinados para decidir cuál son los factores más importantes para determinar la mejor maniobra. La mayoría de los jugadores humanos determinan la mejor maniobra basándose en la experiencia. Desafortunadamente, esto es poco realista para jugadores de la computadora, especialmente aquellos con un espacio limitado de almacenamiento. Chris Wyman (20) describe un método usando algoritmos genéticos por medio del cual un programa, jugándose repetidamente, puede aprender valores para los numerosos atributos del tablero de ajedrez. Diversos experimentos usando este algoritmo muestran que con este método se puede hasta cierto punto aprender cómo jugar estos finales engañosamente simples.

LUNAR LANDER GAMES

El objetivo de este juego es que la nave espacial aterrice en la plataforma de aterrizaje. Zhangbo Liu (21) propone un algoritmo genético dirigido para la planificación de este juego. En un algoritmo genético dirigido, un componente adicional de refuerzo es introducido en el proceso evolutivo del AG, durante cada evolución el componente de refuerzo simulará una serie de acciones de un individuo antes de la prueba real y ajustará la serie de acciones según el refuerzo para tratar de mejorar la función. Esto es aplicado al Juego Lunar Lander (Aterrizador lunar) para que el módulo lunar descendente aprenda a aterrizar en la plataforma de manera segura. En esta investigación se compara la función del algoritmo genético dirigido y un algoritmo genético general así como también Q-Learning en el juego demostrando que al algoritmo genético dirigido podría asegurar alcanzar la meta y además tener una función muy superior a los otros algoritmos.

EVOLUTIONARY COMPUTING SYSTEMS LAB (ECSL)

Este grupo de desarrollo investiga sistemas que combinan la búsqueda de algoritmos genéticos con los principios de los sistemas de razonamiento basados en casos. El CIGAR aprende a tomar menos tiempo y producir mejores soluciones de calidad así como gana experiencia, llevando a un nuevo paradigma para el aprendizaje de máquinas haciendo un especial énfasis en diseño, optimización y modelado humano.

Utilizan Case Injected Genetic AlgoRithms (CIGARs) para jugar juegos de estrategia en computadora implicando un complejo rango de larga planificación con conocimientos imperfectos sobre el estado de

juego. La naturaleza dinámica de este tipo de juegos requiere que los jugadores anticipen los movimientos del adversario y se adapte a sus estrategias consecuentemente; para esto emplean algoritmos genéticos, lanzándoles como un problema de dotación de recursos, las soluciones de las cuales el mapa activa las estrategias del juego. Los resultados muestran que esta búsqueda con algoritmos genéticos se acerca a las estrategias óptimas de juego, luego desarrollan una técnica de aprendizaje construyendo sistema basado en casos de información que puede usarse para anticipar los movimientos de los adversarios. Estos métodos son desarrollados para la adquisición y la respuesta de este conocimiento, ambos después de jugar y de la observación de expertos humanos. Los resultados muestran que los algoritmos genéticos producen estrategias cercanas a lo óptimo que logran la misión de anticiparse y evitar los movimientos de los adversarios.

1.2.6 Redes Neuronales

Las Redes Neuronales no son más que un modelo artificial y simplificado del cerebro humano, que es el ejemplo más perfecto del que disponemos para un sistema que es capaz de adquirir conocimiento a través de la experiencia. Una red neuronal es "un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona". (22)

Consisten en un gran número de elementos simples de procesamiento llamados nodos o neuronas que están organizados en capas. Cada neurona está conectada con otras mediante enlaces de comunicación, cada uno de los cuales tiene asociado un peso. Los pesos representan la información que será usada por la red neuronal para resolver un problema determinado.

Así las Redes Neuronales Artificiales (RNA) son sistemas adaptativos que aprenden de la experiencia, esto es, aprenden a llevar a cabo ciertas tareas mediante un entrenamiento con ejemplos ilustrativos. Mediante un entrenamiento o un aprendizaje, las RNA crean su propia representación interna del problema, por tal motivo se dice que son auto organizadas. Posteriormente pueden responder adecuadamente cuando se les presentan situaciones a las que no habían sido expuestas anteriormente, es decir, las RNA son capaces de generalizar de casos anteriores a caso nuevos. Esta característica es fundamental ya que permite a la red responder correctamente no solo ante informaciones novedosas, sino también ante informaciones distorsionadas o incompletas.

Con las redes neuronales se intenta expresar la solución de problemas complejos, no como una secuencia de pasos, sino como la evolución de unos sistemas de computación inspirados en el cerebro

humano, y dotados por tanto de cierta “inteligencia,” los cuales no son sino la combinación de una gran cantidad de elementos simples de proceso (neuronas) interconectados que, operando de forma masivamente paralela, consiguen resolver problemas relacionados con el reconocimiento de formas o patrones, predicción, codificación, clasificación control y optimización.

Las características de las Redes Neuronales Artificiales las hacen bastante apropiadas para aplicaciones en las que no se dispone a priori de un modelo identificable que pueda ser programado, pero se dispone de un conjunto básico de ejemplos de entrada (previamente clasificados o no). Asimismo, son altamente robustas tanto al ruido como a la disfunción de elementos concretos y son fácilmente paralelizables.

Esto incluye problemas de clasificación y reconocimiento de patrones de voz, imágenes, señales. Asimismo se han utilizado para encontrar patrones de fraude económico, hacer predicciones en el mercado financiero y de tiempo atmosférico. También se pueden utilizar cuando no existen modelos matemáticos precisos o algoritmos con complejidad razonable.

Otro tipo especial de redes neuronales artificiales se ha aplicado en conjunción con los AG para crear controladores para robots. La disciplina que trata la evolución de redes neuronales mediante algoritmos genéticos se denomina Robótica Evolutiva. En este tipo de aplicación el genoma del AG lo constituyen los parámetros de la red (topología, algoritmo de aprendizaje, funciones de activación, etc.) y la adecuación de la red viene dada por la adecuación del comportamiento exhibido por el robot controlado (normalmente una simulación de dicho comportamiento).

APLICACIONES

Un grupo de investigadores de la Universidad de Manizales en el 2005 diseñó una aplicación basada en un juego online que ha tenido gran aceptación por los usuarios de Internet llamado Racing Manager diseñado por Neopoly Group y proporcionado por Yahoo, a la cual se le aplicó una red neuronal para elevar su nivel de funcionalidad con el objetivo de llevarlo a una representación más real.

Este juego consiste en la administración de una escudería de autos para competir contra los usuarios de la aplicación en diferentes partes del mundo; además contiene elementos que se aproximan a la realidad tales como las posibles configuraciones que puede tomar el bólido además del comportamiento del piloto.

La red neuronal le proporciona al piloto la característica de “aprender” con el entrenamiento realizado gracias al funcionamiento de la misma. Es esta propiedad la que permite que el juego no sea tan repetitivo y a la vez interactúe más con el usuario.

BLACK AND WHITE

Actualmente, podemos ver diversos tipos de implementación de NPCs en muchos videojuegos. Un NPC es un personaje en un juego que es controlado automáticamente por inteligencia artificial u otra técnica. Los NPCs son comunes en los juegos RPG (Juegos de Rol o Role Playing Games). Por ejemplo, el juego Black & White tiene como sus objetivos en cuanto a inteligencia artificial que las criaturas sean capaces de aprender todo tipo de cosas, del jugador y de ellas mismas mientras el juego progresa y se crea un mundo dinámico que cambia de acuerdo con las acciones del jugador.

Para ello se emplea una red neuronal, la cual se entrena mediante reglas duras, es decir, cada comportamiento debe de ser preprogramado. Este juego es en la actualidad uno de los más innovadores en cuanto a inteligencia de NPCs se refiere.

BACKGAMMON

Existen sistemas que han utilizado redes neuronales para aprender a jugar Backgammon, se ha utilizado una Red Neuronal para que el juego aprenda jugando contra sí mismo utilizando un aprendizaje supervisado y las técnicas de entrenamiento de retro propagación del error. El Backgammon se trata de un juego de carreras cuyo objetivo inmediato es sacar todas las fichas del tablero antes que el adversario. Se enfrentan dos jugadores en un tablero cuyas casillas son unos triángulos alargados repartidos en cuatro cuadrantes (6 triángulos por cuadrante).

Gerald Tesauro del centro de investigación de la IBM creó el sistema TD-Gammon el cual es una red neuronal capaz de enseñarse a sí misma a jugar Backgammon jugando en contra de ella misma. Este sistema superó las expectativas del desarrollador ya que como resultado demostró poder jugar a tal nivel que derrotó sin problemas al campeón mundial. Además, demostró diferentes estrategias nuevas que nadie había previsto.

DAMAS

Otro ejemplo de las aplicaciones de las redes neuronales es junto a otras técnicas de aprendizaje de IA, ejemplo junto a algoritmos genéticos. Una demostración de esto la presentaron Chellapilla y Fogel (18), que utilizaron un AG para evolucionar redes neuronales que pudieran jugar a las damas.

El algoritmo evolutivo comenzó con una población de 15 redes neuronales con pesos y tendencias, generados aleatoriamente, luego, cada individuo se reprodujo una vez, generando una descendencia con variaciones en los valores de la red. Luego estos 30 individuos compitieron por la supervivencia jugando entre ellos. Y así sucesivamente en un gran número de iteraciones hasta obtener la red neuronal más evolucionada, que fue la que finalmente probaron con jugadores expertos.

ROBÓTICA

En las aplicaciones de robótica también se ve el uso de estas técnicas para que los mismos aprendan determinados comportamientos inteligentes. Por ejemplo el trabajo realizado por personal de la Universidad Federico Santa María en el 2006, que implementa un simulador de un robot móvil tipo Kephera capaz de adquirir comportamientos complejos mediante aprendizaje evolutivo, que puedan ser controlados mediante una interfaz simple con el usuario. Además, el robot debe ser capaz de reconocer los entornos previamente recorridos. El robot posee poca información de su entorno. Sus acciones son consecuencia inmediata de aquello que captan sus sensores, procesado por una red neuronal, que controla la velocidad de los motores.

1.2.7 Sistemas de Lógica Difusa

Los sistemas de lógica difusa, no son más que Sistemas Expertos, basados en reglas difusas, esto se traduce en que el funcionamiento y los componentes de este sistema serán los mismos que de un sistemas basado en reglas, la diferencia radica en que las reglas que se definen son reglas difusas, y además se incorporan nuevos componentes como la interfaz de fuzzificación y la interfaz de defuzzificación que son las encargadas de convertir las variables del problema a resolver en conjuntos difusos, y convertir los conjuntos difusos en salidas numéricas, respectivamente. El uso de la lógica en estos sistemas, los hace sistemas más robustos y les permite tratar con imprecisiones.

La Lógica Difusa es un tipo de lógica que reconoce más que simples valores verdaderos y falsos, las proposiciones pueden ser representadas con grados de veracidad o falsedad. La Lógica Difusa o borrosa descansa en la idea, que en un instante dado, no es posible precisar el valor de una variable X, sino tan solo conocer el grado de pertenencia a cada uno de los conjuntos en que se ha participado el rango de variación de la variable.

Las reglas involucradas en un sistema difuso o borroso, pueden ser aprendidas con sistemas adaptativos que aprenden al 'observar' cómo operan las personas los dispositivos reales, o estas reglas pueden también ser formuladas por un experto humano. En general la lógica borrosa se aplica

tanto a sistemas de control como para modelar cualquier sistema continuo de ingeniería, física, biología o economía.

La lógica borrosa es definida también como un sistema matemático que modela funciones no lineales, que convierte unas entradas en salidas acordes con los planteamientos lógicos que usan el razonamiento aproximado. Por eso se pueden definir entre sus componentes, un sistema de inferencia difuso basado en reglas de la forma " SI..... ENTONCES..... ", donde los valores lingüísticos de la premisa y el consecuente están definidos por conjuntos borrosos, es así como las reglas siempre convierten un conjunto borroso en otro**.**

Los sistemas de lógica difusa están también muy extendidos en la tecnología cotidiana. Imitan la forma en que toman decisiones los humanos, con la ventaja de ser mucho más rápidos. Estos sistemas son generalmente robustos y tolerantes a imprecisiones y ruidos en los datos de entrada.

Cabe destacar los excelentes resultados que brinda un sistema de control basado en Lógica Difusa: ofrece salidas de una forma veloz y precisa, disminuyendo así las transiciones de estados fundamentales en el entorno físico que controle. También está la indecisión de decantarse por los expertos o por la tecnología (principalmente mediante Redes neuronales) para reforzar las reglas heurísticas iniciales de cualquier sistema de control basado en este tipo de lógica.

APLICACIONES

THE SIMS

Es importante mencionar el juego The Sims, el cual es un simulador de la vida. En este juego el jugador se encarga de diseñar las viviendas de personas para intentar darles una vida aceptable. Dentro de estas viviendas estarán varios NPCs que actuarán de acuerdo a su perfil psicológico (definido al inicio del juego mediante algunos atributos); estos NPCs son llamados Sims.

La implementación de la inteligencia de este juego se realiza mediante máquinas de estado finito difusas, o sea, basadas en sistemas de lógica difusa, lo cual es un gran avance en la aplicación de inteligencia artificial. La base de la inteligencia en este juego es su motor de comportamiento, el cual introduce comportamientos y acciones posibles asociadas con un objeto dentro del objeto en sí. Por ejemplo, el archivo para el modelo de una TV contiene todas las instrucciones para verla, encenderla, apagarla, las condiciones en que un Sim querrá o no verla, cómo debe animarse un Sim al verla, etc.

ROBÓTICA

La lógica difusa se ha convertido en una herramienta muy útil para el desarrollo de técnicas de control ya que es capaz de tratar la incertidumbre existente en el entorno. Ejemplo de ello son las siguientes aplicaciones:

La arquitectura Saphira diseñada por Saffiotti, Ruspini y Konolige (23) (24) se basa en la descomposición de las tareas complejas en comportamientos. Estos comportamientos están codificados usando una base de reglas difusas del tipo if *Antecedente* then *Consecuente*. Tanto el antecedente como el consecuente son conjuntos difusos. El valor de verdad del antecedente determina la "deseabilidad" de aplicar esa regla en el estado actual.

En el trabajo de Michaud (25) se implementa el control del robot usando un conjunto de comportamientos codificados mediante una base de reglas de difusas. También hace uso de la lógica difusa a la hora de decidir qué comportamiento es más apropiado para una situación. En la arquitectura propuesta se define un módulo de "recomendaciones" el cual está formado por dos conjuntos de reglas difusas que determinan qué comportamiento activar basándose en los impulsos externos y en las tareas que debe llevar a cabo el agente.

Pirjanian y Mataric en (26) describen el diseño de un controlador para un robot Nomad 200 en comportamientos implementados mediante una base de reglas difusas. La coordinación de comportamientos se lleva a cabo usando los conceptos de la teoría de decisión de objetivo múltiple.

En (27) Goodridge, Kay y Luo describen un sistema de control reactivo basado en comportamientos. Los comportamientos de control simples se han implementado usando la lógica difusa. Estos comportamientos se combinan mediante reglas difusas para formar comportamientos más complejos. Se ha implementado una red de control de comportamientos, usando las reglas difusas tanto implementar dichos comportamientos como para arbitrar entre las recomendaciones de control.

Saffiotti y Wesley en (28) presentan una solución al problema de la estimación de la posición basada en el uso de técnicas difusas. El algoritmo propuesto se basa en que, en cada momento, el robot tiene una idea aproximada de su posición en el mapa representada por un conjunto difuso. Además existe un conjunto de "pistas", que ayudan a la localización, también representadas por un localizador difuso.

Graves, Mollenhauer y Skubic en (29) describen el diseño y la implementación de un robot móvil que hace uso de la lógica difusa para la navegación. La implementación consta de dos módulos

(denominados Mini Boards): la unidad de inferencia difusa (FIU) y la unidad de supervisión. La unidad de inferencia difusa (FIU) se encarga de leer las señales de los sensores y la baliza del objetivo, *fuzzificar* estas señales, ejecutar un conjunto de reglas de inferencia difusas y *defuzzificar* los resultados. Las reglas de inferencia determinan para cada una de los efectores, la acción a llevar a cabo.

En (30), (31) y (32), Howard, Seraji y Tunstel exponen un método basado en lógica difusa para el análisis de las características de un terreno. Se ha diseñado un índice de *"Atravesabilidad"* (Traversability Index). Las características del terreno primero se convierten en variables difusas. Las funciones de pertenencia de los conjuntos que definen estas variables se usan en un conjunto de reglas difusas que determinan la "atravesabilidad" del terreno.

CAPÍTULO 2: DESCRIPCIÓN Y CARACTERÍSTICAS DE LAS TÉCNICAS

Una de las críticas que se oyen más a menudo respecto a la Inteligencia Artificial, es que las máquinas no se pueden considerar inteligentes hasta que no sean capaces de aprender a hacer cosas nuevas y adaptarse a las nuevas situaciones, en lugar de limitarse a hacer aquellas actividades para las que han sido programadas.

Pero se puede plantear una pequeña cuestión acerca, de si la capacidad de adaptarse a entornos nuevos para solucionar nuevos problemas realmente es una de las características importantes de las entidades inteligentes.

La adquisición de conocimiento en sí misma incluye muchas actividades diferentes. El simple almacenamiento de información computada, también conocido como Aprendizaje memorístico, constituye la actividad de aprendizaje más básica. Sin embargo muchos programas de Inteligencia Artificial son capaces de mejorar su rendimiento substancialmente a través de técnicas de aprendizaje memorístico.

Una forma de aprendizaje basada en estímulos provenientes del exterior es el Aprendizaje a partir de ejemplos. Algunas veces se aprende a clasificar las cosas sin haber recibido reglas explícitas, por tanto a veces un programa puede descubrir cosas sin la ayuda de un profesor.

Los investigadores en IA han propuesto numerosos mecanismos para llevar a cabo los tipos de aprendizaje anteriormente propuestos.

Como parte de estos mecanismos se encuentran las técnicas de aprendizaje que son las que permiten que los sistemas computarizados sean capaces de sintetizar el conocimiento necesario para desarrollar una tarea sin necesidad de interacción humana, o sea que sean capaces de aprender.

De las técnicas mencionadas, hay cuatro de ellas que se encuentran entre las que tienen mayor número de aplicaciones en el campo de Realidad Virtual y videojuegos, o sea que permiten que los elementos virtuales de un entorno sean capaces de aprender determinados comportamientos, ellas son: Redes Neuronales, Redes Bayesianas, Algoritmos Genéticos y Lógica Difusa. De ellas se explicará a continuación sus características, estructura y funcionamiento, y cómo se logra que los elementos virtuales logren un aprendizaje utilizando las mismas.

2.1 Técnicas de Aprendizaje aplicadas a la Realidad Virtual

2.1.1 Redes Neuronales

2.1.1.1 Estructura de un Sistema Neuronal Artificial

Los sistemas neuronales artificiales imitan la estructura del hardware del sistema nervioso, con la intención de construir sistemas de procesamiento de información paralelos, distribuidos y adaptativos, que puedan presentar un cierto comportamiento "inteligente".

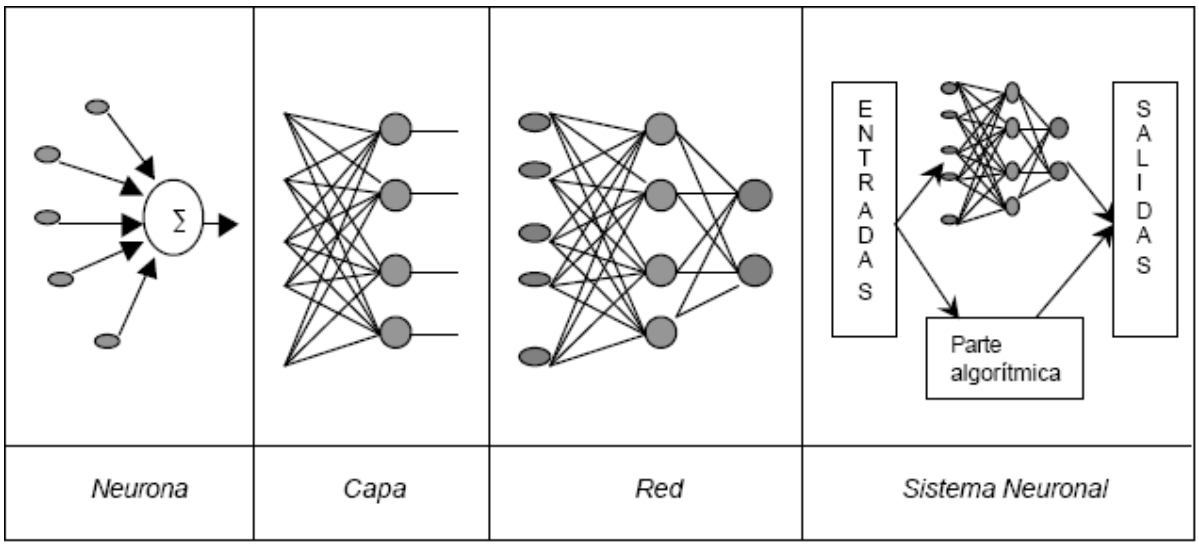


Figura 1. Estructura jerárquica de un sistema basado en Redes Neuronales Artificiales

Cada neurona realiza una función matemática. Las neuronas se agrupan en capas, constituyendo una red neuronal. Una determinada red neuronal está confeccionada y entrenada para llevar a cabo una labor específica. Finalmente, una o varias redes, más las interfaces con el entorno, conforman el sistema global.

En las redes neuronales biológicas, las neuronas corresponden a los elementos de proceso. Las interconexiones se realizan por medio de las ramas de salida (axones) que producen un número variable de conexiones (sinapsis) con otras neuronas o con otras partes. Las redes neuronales son sistemas de elementos simples de proceso muy interconectados.

Los modelos neuronales se diferencian en la función que incorpora la neurona, su organización y forma de las conexiones. Formalmente, un sistema neuronal o conexionista está compuesto de los siguientes elementos:

- Un conjunto de procesadores elementales o neuronas artificiales.
- Un patrón de conectividad o arquitectura.
- Una dinámica de activaciones.
- Una regla o dinámica de aprendizaje.
- El entorno donde opera.

2.1.1.2 Características

Debido a su constitución y fundamentos, las redes neuronales artificiales presentan un gran número de características similares a las del cerebro. Por ejemplo, son capaces de aprender de la experiencia, de generalizar de casos anteriores a nuevos casos, de abstraer características esenciales a partir de entradas que presentan información irrelevante, etc.

- Procesamiento paralelo:

El tipo de procesamiento de la información es en paralelo, en el sentido que muchas neuronas puedan estar funcionando al mismo tiempo. Aunque individualmente las neuronas sean capaces de realizar procesamientos muy simples, ampliamente interconectadas y trabajando en paralelo pueden desarrollar una actividad global de procesamiento impresionante. Esta característica resulta esencial, ya que al trabajar las neuronas en paralelo, o sea, al unísono, pueden realizar además, las tareas en menos tiempo que si lo hicieran una por una.

- Memoria Distribuida:

Mientras en un computador la información ocupa posiciones de memoria bien definidas, en los sistemas neuronales se encuentra distribuida por las sinapsis de la red, de modo que si una sinapsis se daña solamente se pierde una pequeña parte de la información.

- Aprendizaje adaptativo:

La capacidad de aprendizaje adaptativo es una de las características más atractivas de redes neuronales. Esto es, aprenden a llevar a cabo ciertas tareas mediante un entrenamiento con ejemplos ilustrativos. Como las redes neuronales pueden aprender a diferenciar patrones mediante ejemplos y

entrenamientos, no es necesario elaborar modelos a priori ni necesidad de especificar funciones de distribución de probabilidad.

Las redes neuronales son sistemas dinámicos auto adaptativos. Son adaptables debido a la capacidad de auto ajustarse que tienen las neuronas. Son dinámicos, pues son capaces de estarse adaptando constantemente a las nuevas condiciones. En el proceso de aprendizaje, los enlaces ponderados de las neuronas se ajustan de manera que se obtengan unos resultados específicos.

- Auto organización:

Las redes neuronales utilizan su capacidad de aprendizaje adaptativo para organizar la información que reciben durante el aprendizaje y/o la operación. Mientras el aprendizaje es la modificación de cada elemento procesal, la auto organización consiste en la modificación de la red neuronal completa para llevar a cabo un objetivo específico.

Esta auto organización da lugar a la generalización: facultad de responder apropiadamente cuando se les presentan datos o situaciones a los que no habían sido expuestas anteriormente. El sistema puede generalizar la entrada para obtener una respuesta. Esta característica es de especial importancia cuando se tiene que solucionar problemas para los cuales la información de entrada es poco clara o incompleta.

La neurona artificial

Las redes neuronales son modelos que intentan reproducir el comportamiento del cerebro. Como modelos, son una simplificación de lo que emulan, incorporando los elementos relevantes del sistema. Una elección adecuada de sus características, más una estructura conveniente, es el procedimiento convencional utilizado para construir redes capaces de realizar una determinada tarea.

Un modelo de red neuronal consta de dispositivos elementales de proceso: las neuronas. A partir de ellas se pueden generar representaciones específicas, de modo que un estado conjunto de ellas puede representar una letra, un número o cualquier objeto.

Generalmente se pueden encontrar tres tipos de neuronas: entradas, salidas y ocultas.

1. Aquellas que reciben estímulos externos y toman la información de entrada. Estas entradas (que son entradas a la red) pueden provenir de sensores o de otros sectores del sistema.

2. Las unidades ocultas son aquellas cuyas entradas y salidas se encuentran dentro del sistema, es decir, no tienen contacto con el exterior. Estas neuronas se ocupan del procesamiento de la información.
3. Las unidades de salida envían la respuesta del sistema. Estas pueden controlar directamente potencias u otros sistemas.

La neurona artificial pretende emular las características de las neuronas biológicas. Cada neurona i-ésima está caracterizada en cualquier instante por un valor numérico denominado valor o estado de activación ai(t). Asociado a cada unidad, existe una función de salida fi, que transforma el estado actual de activación en una señal de salida, yi.

Dicha señal es enviada a través de los canales de comunicación unidireccionales a otras unidades de la red. En estos canales la señal se modifica de acuerdo con la sinapsis (el peso sináptico, wji) asociada a cada uno de ellos según una determinada regla.

Una función de activación, F, determina el nuevo estado de activación aj (t+1) de la neurona, teniendo en cuenta la entrada total calculada y el anterior estado de activación aj (t).

Si tenemos N unidades (neuronas), podemos ordenarlas arbitrariamente y designar la j-ésima unidad como Uj. Su trabajo consiste únicamente en recibir las entradas de las neuronas vecinas y calcular un valor de salida, que es enviado a todas las neuronas restantes.

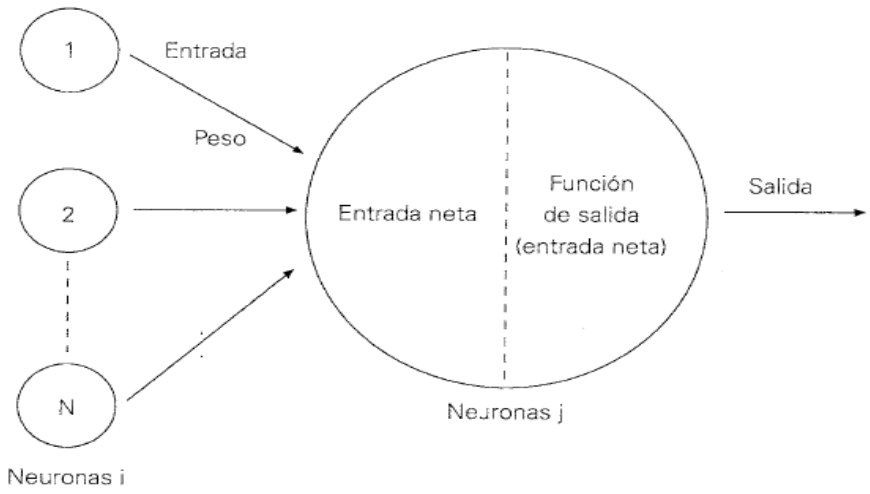


Figura 2 Funcionamiento general de una neurona artificial

Conexiones entre Neuronas

Las conexiones entre las neuronas de una red tienen asociado un peso, que es el que hace que la red adquiera conocimiento.

Tomemos el valor yi como el valor de salida de una neurona i en un instante dado. Una neurona recibe un conjunto de señales que le dan información del estado de activación de todas las neuronas con las que se encuentra conectada. Cada conexión (sinapsis) entre la neurona i y la neurona j está ponderada por un peso wji. Normalmente, como simplificación, se considera que el efecto de cada señal es aditivo, de forma que la entrada neta que recibe una neurona (potencial post sináptico) netj es la suma del producto de cada señal individual por el valor de la sinapsis que conecta ambas neuronas:

$$net_j = \sum_{i}^{N} w_{ji} \bullet y_i$$

Esta regla muestra el procedimiento a seguir para combinar los valores de entrada a una neurona con los pesos de las conexiones que llegan a esa neurona, y es conocida como regla de propagación.

2.1.1.3 Topología o arquitectura de las Redes Neuronales

Se denomina topología o arquitectura de la red a la organización y disposición de las neuronas en la red formando capas más o menos alejadas de la entrada y salida de la red. Así, los parámetros fundamentales de la red son: el número de capas (monocapa o multicapa), el número de neuronas por capa, el grado de conectividad y el tipo de conexiones entre neuronas (unidireccionales o recurrentes).

Las neuronas que componen una RNA se organizan de forma jerárquica, formando capas. Una capa o nivel, es el conjunto de neuronas, cuyas entradas de información provienen de la misma fuente (que puede ser otra capa de neuronas) y cuyas salidas de información se dirigen al mismo destino (que puede ser otra capa de neuronas) en este sentido se distinguen tres tipos de capas: la capa de entrada, recibe la información del exterior; la, o las capas ocultas son aquellas cuyas entradas y salidas se encuentran dentro del sistema, y por tanto, no tienen contacto con el exterior; por último, la capa de salida envía la respuesta de la red al exterior.

En función de la organización de las neuronas en la red, formado capas o agrupaciones podemos encontrarnos con dos tipos de arquitecturas básicas: redes multicapa y redes monocapa.

Redes Neuronales Monocapa

Las redes monocapa, están organizadas, como el propio nombre lo indica, en una sola capa de neuronas. Cada neurona está conectada con todas las demás que forman la arquitectura.

Se utilizan típicamente en tareas relacionadas con la auto-asociación; por ejemplo para regenerar informaciones de entrada que se presentan a la red incompleta o distorsionada. Para ello se almacena en los pesos de la red, ciertas informaciones mediante una etapa de entrenamiento. Posteriormente, cuando se presenta una información a la entrada de la red, esta responde proporcionando la información más parecida de las almacenadas. Por tal motivo las redes que llevan a cabo este tipo de tareas también reciben el nombre de redes auto-asociativas, ya que intenta asociar una información consigo misma.

Redes Neuronales Multicapa

Las redes multicapas, disponen de conjuntos de neuronas agrupadas en dos o más capas. En la mayoría de los casos, estas redes están formadas por una capa de entrada, una capa de salida y una o más capas intermedias u ocultas; donde la información se trasmite desde la capa de entrada, hacia la capa de salida, y donde cada neurona está conectada con todas las neuronas de la siguiente capa. Las redes multicapas se suelen utilizar en tareas denominadas, heteroasociativas. De lo que se trata es que la red aprenda parejas de datos, de forma que cuando se presenta cierta información de entrada A, deberá responder, generando la correspondiente salida asociada B. Por tal motivo las redes que llevan a cabo este tipo de tareas también reciben el nombre de redes heteroasociativas, ya que intenta asociar pares de informaciones distintas. Este tipo de redes son útiles para la clasificación de patrones, ya que en este caso se asocia el ejemplo con la clase o categoría a la que pertenece y a la aproximación de funciones, donde se asocia una información de entrada con otra información de salida.

2.1.1.4 Aprendizaje

El aprendizaje se entiende como la modificación del comportamiento inducido por la interacción con el entorno, y como resultado de experiencias conducente al establecimiento de nuevos modelos de respuesta a estímulos externos.

En las redes neuronales artificiales, el conocimiento se encuentra representado en los pesos de las conexiones entre neuronas. Todo proceso de aprendizaje implica cambios en estas conexiones, es decir, se aprende modificando los valores de los pesos de la red, en respuesta a un conjunto de ejemplos denominado grupo de entrenamiento.

Durante el proceso de aprendizaje, los pesos de las conexiones de la red sufren modificaciones, por lo tanto se puede afirmar que la red "ha aprendido" cuando los valores de los pesos permanecen estables (dwij/dt = 0).

Los modelos neuronales utilizan varios algoritmos de estimación, aprendizaje o entrenamiento para encontrar los valores de los pesos de la red y así lograr que aprenda a solucionar un determinado problema; estos criterios se denominan de forma genérica reglas de aprendizaje. Las reglas de aprendizaje consisten generalmente en algoritmos matemáticos.

El entrenamiento se realiza mediante patrones-ejemplo. De modo general, se distinguen dos tipos de aprendizaje:

- Redes Neuronales con aprendizaje supervisado.
- Redes neuronales con aprendizaje no supervisado o auto organizado.

La diferencia fundamental entre ambos tipos radica en la existencia o no de un agente externo (supervisor) que controle el proceso de aprendizaje de la red.

En cada tipo de aprendizaje, los valores de los pesos se hallan se distintas formas. En ambos casos, una vez obtenidos y guardados los pesos óptimos en la fase de entrenamiento, debemos medir la eficacia de la red de forma objetiva, mediante la presentación de casos nuevos (diferentes a los casos de entrenamiento), de forma que a la fase de entrenamiento la debe seguir una fase de test. En esta fase no se modifican los pesos, solamente se presentan casos nuevos, llamados casos de test, a la entrada de la red, y esta proporciona una salida para cada uno de ellos. Si se comprueba que se siguen obteniendo resultados dentro del margen de error deseado, se puede proceder a emplear la RNA dentro de su entorno de trabajo real.

El aprendizaje supervisado

En el aprendizaje supervisado, hay un supervisor que controla el proceso de aprendizaje de la red. El supervisor, comprueba la salida de la red en respuesta a una determinada entrada y en el caso de que

la salida no coincida con la deseada, se procede a modificar los pesos de las conexiones, con el fin de conseguir que la salida obtenida, se aproxime a la deseada.

Si la red utiliza un tipo de aprendizaje supervisado debemos proporcionarle parejas de patrones entrada-salida y la red neuronal aprende a asociarlos. En terminología estadística equivale a los modelos en los que hay vectores de variables independientes y dependientes: técnicas de regresión, análisis discriminante, modelos de series temporales, etc.

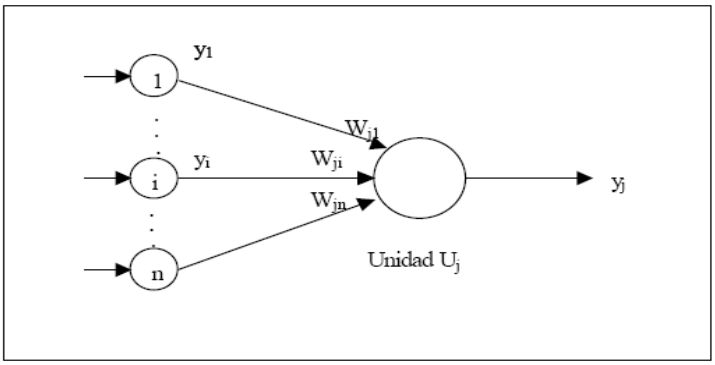
En este tipo de aprendizaje se suelen considerar tres formas: Aprendizaje por corrección de error, Aprendizaje por refuerzo y aprendizaje estocástico.

- Aprendizaje por corrección de error.

Consiste en ajustar los pesos de las conexiones de la red en función de la diferencia entre los valores deseados y los obtenidos en la salida; es decir, en función del error cometido.

Una regla o algoritmo simple de aprendizaje por error podría ser como la siguiente:

$$\Delta w_{ji} = \alpha\, y_i\, (d_j - y_j)$$



Donde:

Δwji : Variación en el peso de la conexión entre las neuronas *i* y *j*

(Δwji = wji actual – wjianterior)

yi : Valor de salida de la neurona i.

dj : Valor de salida deseado para la neurona j.

yj : Valor de salida obtenido de la neurona j.

α : Factor de aprendizaje ($0 < \alpha \leq 1$) que regula la velocidad del aprendizaje.

Un ejemplo de este tipo de algoritmo es la regla de aprendizaje del Perceptrón, utilizada en el entrenamiento de la red de dicho nombre, desarrollada por Rosenblatt en 1958. Éste presenta algunas limitaciones, como no considerar la magnitud del error global cometido durante todo el proceso de aprendizaje, sino que considera solamente los errores locales de cada información por separado.

Otro algoritmo de este tipo, pero que mejora el del Perceptrón y permite un aprendizaje más rápido y un campo de aplicación más amplio es el denominado regla delta o regla del error mínimo cuadrado (33). Se aplica en las redes ADALINE, con una sola neurona de salida, y MADALINE, con varias neuronas de salida.

Este error medio se expresa de la siguiente manera:

$$Error_{global} = \frac{1}{2P} \sum_{k=1}^{P} \sum_{j=1}^{N} (y_j^{(k)} - d_j^{(k)})^2$$

Donde:

N: Número de neuronas de salida

P: Número de informaciones que debe aprender la red

$$\frac{1}{2} \sum_{j=1}^{N} (y_j^k - d_j^k)^2$$

Error cometido en el aprendizaje de la información k- esima

Por lo tanto, se trata de encontrar unos pesos para las conexiones de la red que minimicen esta función de error. Para ello, el ajuste de los pesos de las conexiones de la red se puede hacer de forma proporcional a la variación relativa del error que se obtiene al variar el peso correspondiente:

$$\Delta w_{ji} = k \frac{\partial\, Error_{global}}{\partial\, w_{ji}}$$

Mediante este procedimiento, se llega a obtener un conjunto de pesos con los que se consigue minimizar el error medio.

Otro algoritmo de aprendizaje por corrección de error, propuesto en 1986, lo constituye el denominado regla delta generalizada o algoritmo de retropropagación del error (error back-propagation), conocido también como regla LMS(Least-Mean-Square Error) multicapa.

Se trata de una generalización de la regla delta para poder aplicarla a redes con conexiones hacia delante (feedforward) con capas o niveles ocultos de neuronas que no tienen relación con el exterior. Son redes con capa de entrada, capas ocultas y capa de salida.

Estas redes multicapa pueden utilizarse en muchas más aplicaciones que las ya mencionadas Perceptrón, ADALINE y MADALINE, pero su proceso de aprendizaje es mucho más lento debido a que durante el mismo se tiene que explorar el espacio de posibles formas de utilización de las neuronas de las capas ocultas; es decir, se debe establecer cuál va a ser su papel en el funcionamiento de la red.

- Aprendizaje por refuerzo.

Es un aprendizaje supervisado más lento que el anterior, que se basa en no disponer de un ejemplo completo del comportamiento deseado; es decir, no indicar exactamente la salida que se desea que proporcione la red ante una determinada entrada.

En este aprendizaje la función del supervisor se reduce a indicar mediante una señal de refuerzo si la salida obtenida se ajusta a la deseada (éxito = +1 ó fracaso = -1), y en función de ello se ajustan los pesos, basándose en un mecanismo de probabilidades.

Un ejemplo de algoritmo es el Linear Reward-Penality o L_{R-P} (algoritmo lineal con recompensa y penalización) presentado por Narendra y Thathacher (34) en 1974. Este algoritmo ha sido ampliado por Barto y Anandan (35), quienes en 1985 desarrollaron el denominado Associative Reward-Penality o A_{R-P} (algoritmo asociativo con recompensa y penalización), que se aplica en redes con conexiones hacia delante de dos capas cuyas neuronas de salida presentan una función de activación estocástica.

Otro algoritmo conocido es el *Adaptive Heuristic Critic*, introducido por Barto, Sutton y Anderson (36) en 1983, que se utiliza en redes feedforward de tres capas especialmente diseñadas para que una parte de la red sea capaz de generar una valor interno de refuerzo que es aplicado a las neuronas de salida de la red.

- Aprendizaje estocástico.

Este tipo de aprendizaje consiste básicamente en realizar cambios aleatorios a los pesos de las conexiones de la red y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad.

En el aprendizaje estocástico se suele hacer una analogía en términos termodinámicos, asociando la red neuronal con un sólido físico que tiene cierto estado energético. En el de caso de la red, la energía de la misma representaría el grado de estabilidad de la red, de tal forma que el estado de mínima energía correspondería a una situación en la que los pesos de las conexiones consiguen que su funcionamiento sea el que más se ajusta al objetivo deseado.

Según lo anterior, el aprendizaje consistiría en realizar un cambio aleatorio de los valores de los pesos y determinar la energía de la red. Si la energía es menor después del cambio; es decir, si el comportamiento de la red se acerca al deseado, se acepta el cambio; de lo contrario, se aceptaría el cambio en función de una determinada y preestablecida distribución de probabilidades.

El aprendizaje no supervisado o auto organizado.

Con el aprendizaje no supervisado, la red no requiere influencia de un supervisor, para ajustar los pesos de las conexiones entre las neuronas. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta. Su función consiste en encontrar las características, regularidades, correlaciones o categorías, que se pueden establecer entre los datos que se presentan en su entrada. Suele decirse que estas redes son capaces de auto organizarse.

Si el entrenamiento es no supervisado, únicamente debemos suministrar a la red los datos de entrada para que extraiga los rasgos característicos esenciales. En terminología estadística equivale a los modelos en los que sólo hay vectores de variables independientes y buscan el agrupamiento de los patrones de entrada: análisis de conglomerados, escalas multidimensionales, etc.

Puesto que no hay un supervisor que indique a la red la respuesta que debe generar una entrada concreta, cabría preguntarse precisamente por lo que la red genera en estos casos. Existen varias posibilidades en cuanto a la interpretación de las salidas, que dependen de su estructura y del algoritmo de aprendizaje empleado.

En algunos casos, la salida representa el grado de familiaridad o similitud entre las entradas y las informaciones que se le han mostrado hasta entonces (en el pasado). En otro caso podría realizar un establecimiento de categorías, indicando a la salida a qué categoría pertenece la información presentada en la entrada, siendo la propia red quien debe encontrar las categorías apropiadas a partir de correlaciones entre las informaciones presentadas.

También el aprendizaje sin supervisión permite realizar una *codificación* de los datos de entrada, generando a la salida una versión codificada de la entrada, con menos bits, pero manteniendo la información relevante de los datos.

Finalmente, algunas redes con aprendizaje no supervisado realizan un mapeo de características (feature mapping), obteniéndose en las neuronas de salida una disposición geométrica que representa un mapa topográfico de las características de los datos de entrada, de tal forma que si se presentan a la red informaciones similares, siempre sean afectadas neuronas de salida próximas entre sí, en la misma zona del mapa.

Suelen considerarse dos algoritmos de aprendizaje no supervisado:

- Aprendizaje hebbiano

Este tipo de aprendizaje se basa en el postulado formulado por Donald O. Hebb en 1949: "Cuando un axón de una celda A está suficientemente cerca como para conseguir excitar a una celda B y repetida o persistentemente toma parte en su activación, algún proceso de crecimiento o cambio metabólico tiene lugares en una o ambas celdas, de tal forma que la eficiencia de A, cuando la celda a activar es B, aumenta. Por celdas, Hebb entiende un conjunto de neuronas fuertemente conexionadas a través de una estructura compleja. La eficiencia podría identificarse por la intensidad o magnitud de la conexión, es decir, con el peso.

Se puede decir que el aprendizaje consiste en el ajuste de los pesos de las conexiones de acuerdo con la correlación (multiplicación en el caso de valores binarios +1 y –1) de los valores de activación (salidas) de las neuronas conectadas.

$$\Delta w_{ij} = y_i * y_j$$

Esta expresión responde a la idea de Hebb, puesto que si las dos unidades son activas (positivas), se refuerza la conexión; por el contrario, cuando una es activa y la otra pasiva, se debilita la conexión.

Existen muchas variaciones de dicho aprendizaje, por ejemplo, Sejnowski (37) en 1977 utilizó la correlación de covarianza de los valores de activación de las neuronas. Sutton y Barto (38) en 1981 utilizaron la correlación el valor medio de una neurona con la varianza de la otra. Klopf (39) en 1986 propuso una correlación entre las variaciones de los valores de activación en dos instantes de tiempo sucesivos, aprendizaje que denominó drive-reinforcement y que utilizó en redes del mismo nombre con topología feedforward de dos capas.

Otra versión de este aprendizaje es el hebbiano diferencial, que utiliza la correlación de las derivadas en el tiempo de las funciones de activación de las neuronas.

- Aprendizaje competitivo y cooperativo

En dicho aprendizaje suele decirse que las neuronas compiten (y cooperan) unas con otras con el fin de llevar a cabo una tarea dada.

La competición ente neuronas se realiza en todas las capas de la red, existiendo en estas neuronas conexiones recurrentes de autoexcitación y conexiones de inhibición por parte de neuronas vecinas. Si el aprendizaje es cooperativo, estas conexiones con las vecinas serán de excitación.

El objetivo de este aprendizaje es categorizar los datos que se introducen en la red. De esta forma, las informaciones similares son clasificadas formando parte de la misma categoría, y por tanto deben activar la misma neurona de salida. Las categorías deben ser creadas por la misma red, puesto que se trata de aprendizaje no supervisado, a través de las correlaciones ente los datos.

En este tipo de redes, cada neurona tiene asignado un peso total, suma de todos los pesos de las conexiones que tiene a su entrada. El aprendizaje afecta sólo a las neuronas ganadoras (activas), redistribuyendo este peso total entre todas las conexiones que llegan a la neurona vencedora y repartiendo esta cantidad por igual entre todas las conexiones procedentes de unidades activas. Por tanto, la variación del peso de una conexión entre una unidad i y otra j será nula si la neurona j no recibe excitación por parte de la neurona i y otra j será nula si la neurona j no recibe excitación por parte de la neurona i, y se modificará si es excitada por dicha neurona i.

Existe otro caso particular de aprendizaje competitivo, denominado teoría de la resonancia adaptativa, desarrollado por Carpenter y Grossberg en 1986 y utilizado en la red feedforward /feedback de dos capas conocida como ART. Esta red realiza un prototipado de las informaciones que recibe a la entrada, generando como salida un ejemplar o prototipo que representa a todas las informaciones que podrían considerarse pertenecientes a la misma categoría.

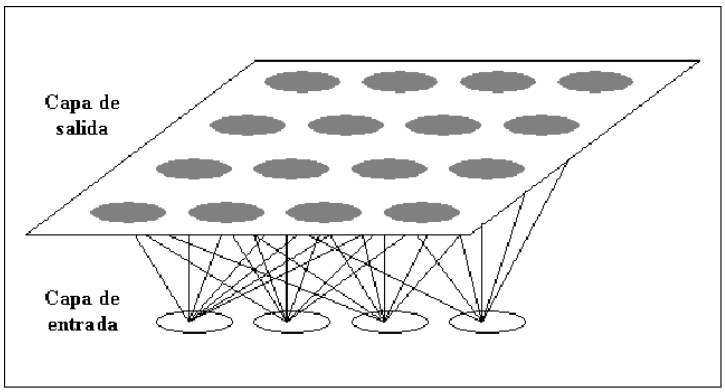


Figura 3. Arquitectura de un mapa auto organizado.

2.1.2 Sistemas de Lógica Difusa

En Inteligencia artificial, la lógica difusa o lógica borrosa se utiliza para la resolución de una variedad de problemas, principalmente los relacionados con control de procesos industriales complejos y sistemas de decisión en general, la resolución, la compresión de datos. Los sistemas basados en lógica difusa imitan la forma en que toman decisiones los humanos, con la ventaja de ser mucho más rápidos. Estos sistemas son generalmente robustos y tolerantes a imprecisiones y ruidos en los datos de entrada.

En general, un Sistema Basado en Reglas Difusas es un sistema basado en reglas en el que la Lógica Difusa puede ser empleada tanto como una herramienta para representar distintas formas de conocimiento sobre el problema a resolver como para modelar las interacciones y relaciones existentes entre las variables del mismo.

Existen distintos tipos de SBRDs dependiendo de la naturaleza de las entradas y salidas que manejan, así como la estructura de reglas difusas con la que trabajan, pudiendo distinguirse entre SBRDs puros, descriptivos y aproximativos.

Estos sistemas manejan datos de entrada-salida reales. Asimismo, proporcionan un marco natural para incluir conocimiento experto en forma de reglas lingüísticas y permiten combinarlo de forma muy sencilla con reglas obtenidas a partir de conjuntos de datos que reflejen el comportamiento del sistema.

La Lógica difusa utiliza expresiones que no son ni totalmente ciertas ni completamente falsas, es decir es la lógica aplicada a conceptos que pueden tomar un valor cualquiera de veracidad de un conjunto de valores que oscilan entre dos extremos, la verdad absoluta y la falsedad total.

La lógica difusa permite tratar información imprecisa como estatura media o temperatura baja, en términos de conjuntos borrosos que se combinan en reglas para definir acciones: si la temperatura es alta, entonces enfriar mucho. De esta manera, los sistemas de control basados en lógica difusa combinan variables de entrada definidas en términos de conjuntos difusos por medio de grupos de reglas que producen uno o varios valores de salida. Los valores de salida de las relaciones difusas no son ambivalentes, 0 ó 1, sino que puede tomar valores reales entre 0 y 1.

La lógica difusa está relacionada con probabilidades, ya que intenta cuantificar una incertidumbre. Si P es una proposición, se le puede asociar un numero v(P) en el intervalo [0; 1] tal que: si v(P) = 0, P es falso; si v(P) = 1, P es verdadero. La veracidad de P aumenta con v(P).

2.1.2.1 Conceptos básicos

La lógica difusa es una extensión del concepto clásico de conjuntos. El concepto fundamental en que se basa la lógica difusa es el de conjunto difuso, un tipo de conjunto caracterizado porque los elementos del universo de discurso en el que están definidos pueden pertenecer a el en un cierto grado, representado por una función de pertenencia, como se ve en la siguiente tabla:

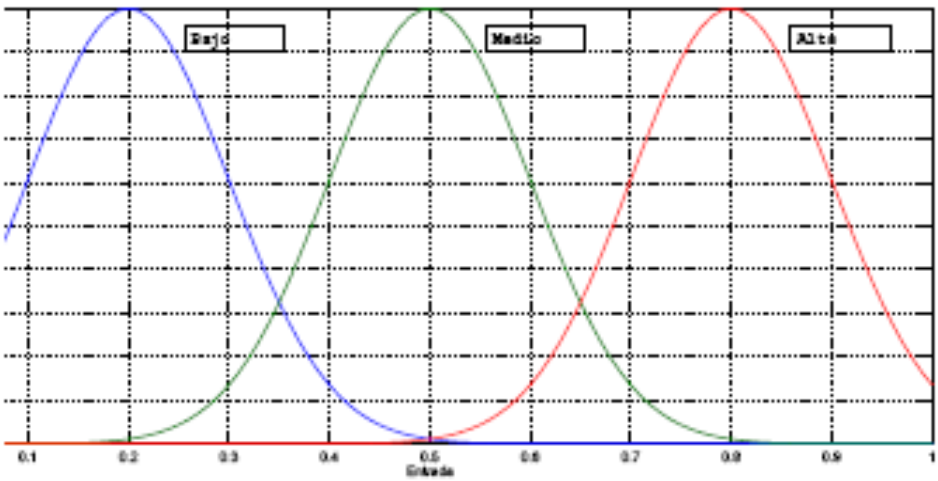


Tabla 2. Algunas funciones de pertenencia.

Las funciones de pertenencia permiten caracterizar variables lingüísticas, que puedan expresarse en términos del lenguaje natural, como *bajo*, *medio* y *alto*. En la tabla, las funciones de pertenencia caracterizan la variable *voltaje*. Se pueden utilizar diversas funciones, entre las que se destacan la gaussiana, triangular, trapezoidal, etc. La elección de la forma de la función de pertenencia es subjetiva y dependiente del contexto. Una función de pertenencia puede ser escrita como:

$$F = \{(u, \mu_F(u)) | u \epsilon U\}$$

donde u representa a cada elemento del universo de discurso U.

Por ejemplo, si en la figura X el voltaje es 0;4, el grado de pertenencia al conjunto difuso bajo es 0;14, al conjunto medio es 0;6 y al conjunto alto es 0.

Teoría de Conjuntos Difusos

Una buena estrategia para presentar la teoría de Conjuntos Difusos, consiste en recordar algunos aspectos de la teoría de conjuntos convencionales (que llamaremos conjuntos concretos), y a partir de allí hacer una extensión a los conjuntos difusos:

Un conjunto concreto se define como una colección de elementos que existen dentro de un Universo. Así, si el universo consta de los números enteros no negativos menores que 10:

U={0,1,2,3,4,5,6,7,8,9}
entonces podemos definir algunos conjuntos como, por ejemplo:
A={0,2,4,6,8}
B={1,3,5,7,9}
C={1,4,7}

Con estas definiciones hemos establecido que cada uno de los elementos del Universo pertenecen o no a un determinado conjunto. Por lo tanto, cada conjunto puede definirse completamente por una función de pertenencia, que opera sobre los elementos del Universo, y que le asigna un valor de 1 si el elemento pertenece al conjunto, y de 0 si no pertenece.

Tomando como ejemplo el conjunto C enumerado arriba, su función de pertenencia uC(x) sería de la siguiente forma:

uC(0)=0, uC(1)=1, uC(2)=0, uC(3)=0, uC(4)=1, uC(5)=0, uC(6)=0, uC(7)=1,
uC(8)=0, uC(9)=0

Ahora bien, un Conjunto Difuso se define de forma similar, con una diferencia conceptual importante: un elemento puede pertenecer parcialmente a un conjunto. De esta forma, un conjunto difuso D definido sobre el mismo universo U puede ser el siguiente:

D={20%/1,50%/4,100%/7}

La definición anterior significa que el elemento 1 pertenece en un 20% al conjunto D (y por tanto pertenece en un 80% al complemento de D), en tanto que el elemento 4 pertenece en un 50%, y el elemento 7 en un 100%.

En forma alternativa, diríamos que la función de pertenencia uD(x) del conjunto D es la siguiente:

uD(0)=0.0, uD(1)=0.2, uD(2)=0.0, uD(3)=0.0, uD(4)=0.5, uD(5)=0.0, uD(6)=0.0,
uD(7)=1.0, uD(8)=0.0, uD(9)=0.0

Las primeras diferencias que se hacen evidentes entre los Conjuntos Concretos y los Conjuntos Difusos son las siguientes:

La función de pertenencia asociada a los conjuntos concretos sólo puede tener dos valores: 1 ó 0, mientras que en los conjuntos difusos puede tener cualquier valor entre 0 y 1.

Un elemento puede pertenecer (parcialmente) a un conjunto difuso y simultáneamente pertenecer (parcialmente) al complemento de dicho conjunto. Lo anterior no es posible en los conjuntos concretos, ya que constituiría una violación al principio del tercer excluido.

Las fronteras de un conjunto concreto son exactas, en tanto que las de un conjunto difuso son, precisamente, difusas, ya que existen elementos en las fronteras mismas, y estos elementos están a la vez dentro y fuera del conjunto.

2.1.2.2 Operaciones entre conjuntos difusos

Las tres operaciones básicas entre conjuntos concretos, Unión, Intersección y Complemento, se definen también para los conjuntos difusos, intentando mantener el significado de tales operaciones.

La definición de estas operaciones se hace empleando el concepto de función de pertenencia de los conjuntos.

Intersección: el resultado de efectuar la operación de Intersección entre dos conjuntos difusos A y B definidos sobre el mismo Universo, y con funciones de pertenencia uA(x) y uB(x) respectivamente, es un nuevo conjunto difuso A∩B definido sobre el mismo universo, y con función de pertenencia uA∩B (x), dada por:

uA∩B(x)= uA(x) (*) uB(x)
En donde el operador (*) debe satisfacer las siguientes propiedades:
x(*)y = y(*)x
(x(*)y)(*)z = x(*)(y(*)z)
si x<y y z<w entonces x(*)z < y(*)w
x(*)1=x

Todo operador que satisfaga las propiedades anteriores se conoce como una T-Norma, y representa la Intersección de dos conjuntos difusos. Dos de los operadores más sencillos son el mínimo y el producto clásico (en adelante se denotarán por min y * respectivamente).

Unión: el resultado de efectuar la operación de Unión entre dos conjuntos difusos A y B definidos sobre el mismo Universo, y con funciones de pertenencia uA(x) y uB(x) respectivamente es un nuevo conjunto difuso AUB definido sobre el mismo universo, y con función de pertenencia uAUB(x), dada por:

uAUB(x)= uA(x)(+)uB(x)
En donde el operador (+) debe satisfacer las siguientes propiedades:
x(+)y = y(+)x
(x(+)y)(+)z = x(+)(y(+)z)
si x<y y z<w entonces x(+)z < y(+)w
x(+)0=x

Todo operador que satisfaga las propiedades anteriores se conoce como una S-Norma, y representa la Unión de dos conjuntos difusos. Uno de los operadores más sencillo es el máximo (en adelante se denotará por max).

Complemento: el resultado de efectuar la operación de Complemento sobre un conjunto difuso A definido sobre un Universo, y con función de pertenencia uA(x) es un nuevo conjunto difuso A' definido sobre el mismo universo, y con función de pertenencia uA'(x), dada por:

uA'(x)= 1-uA(x)

2.1.2.3 Principios de Lógica Difusa

Es bien conocido que la teoría de conjuntos, el álgebra booleana y la lógica tradicional son isomorfas, bajo transformaciones adecuadas. Esto significa que tienen una estructura subyacente similar, y que por tanto las definiciones que se hagan en una cualquiera de las tres teorías se puede llevar a las otras dos, mediante transformaciones adecuadas. La siguiente tabla muestra la correspondencia de algunos operadores.

Teoría de Conjuntos	Algebra Booleana	Lógica Tradicional
Intersección	Conjunción	AND
Unión	Disyunción	OR
Complemento	Negación	NOT

Tabla3. Correspondencia entre operadores de la Teoría de Conjuntos, el Algebra Booleana y la Lógica Tradicional

Ahora bien, el razonamiento lógico consiste en la combinación de proposiciones para producir nuevas proposiciones; así, la combinación de las proposiciones "X es A" y "Y es B" mediante el operador AND da como resultado la proposición *"X es A AND Y es B"*. La tabla sugiere que puede representarse esta combinación mediante un operador análogo a la Intersección de Conjuntos.

Lo anterior es posible porque en la lógica tradicional toda proposición puede tener uno de dos valores: verdadero o falso, lo que corresponde en la teoría de conjuntos concretos a los únicos dos valores que puede tomar la función de pertenencia para cualquier conjunto: 1 ó 0.

Ahora bien, en lógica difusa una proposición puede representarse por un conjunto difuso: "X es A" corresponde a un conjunto *A* con función de pertenencia *uA(x)*, mientras que *"Y es B"* corresponde a un conjunto *B* con función de pertenencia *uB(y)*, y la combinación de estas dos proposiciones con el operador AND, es decir la proposición *"X es A AND Y es B"* corresponde a un nuevo conjunto difuso *AANDB* con función de pertenencia *uAANDB(x,y)= min(uA(x),uB(y))* en donde se ha utilizado el

operador *min* para efectuar la intersección de los dos conjuntos, pero en general podría haberse utilizado cualquier T-Norma.

Nótese que los universos de discurso sobre los cuales están definidos los conjuntos A y B no son necesariamente el mismo, son, por ejemplo U y V respectivamente, mientras que el conjunto AANDB está definido sobre el universo UxV.

En forma análoga, al operador lógico OR puede hacerse corresponder a una S-Norma, mientras que al operador lógico NOT puede hacerse corresponder el complemento.

Operador de Implicación

Un análisis especial debe hacerse con el operador lógico de implicación =>, que combina dos proposiciones con la expresión SI... ENTONCES... (IF... THEN...), y que es el fundamento de las inferencias realizadas en sistemas de lógica difusa.

Ante todo, conviene precisar que el interés por el operador => consiste en encontrar una forma de interpretar proposiciones semejantes a las utilizadas en la experiencia común para describir conocimientos. Es decir, encontrar un camino matemático para evaluar proposiciones como las siguientes: "Si las vibraciones son altas Entonces el rodamiento está desgastado", o "Si los ingresos del cliente son bajos Entonces su capacidad de endeudamiento es poca".

Ahora bien, la implicación => de la lógica tradicional tiene una tabla de verdad que se muestra en la tabla

p	q	p=>q
Verdad	Verdad	Verdad
Verdad	Falso	Falso
Falso	Verdad	Verdad
Falso	Falso	Verdad

Tabla 4. Tabla de verdad de la implicación lógica tradicional.

Esta tabla de verdad puede obtenerse también con los operadores básicos Conjunción, Disyunción y Negación, con, por lo menos, dos expresiones distintas:

(p=>q)<=>(~(p^(~q)))
(p=>q)<=>((~p)^q)

Las anteriores equivalencias permiten deducir expresiones para la implicación de la lógica difusa: Para combinar dos proposiciones "X es A" y "Y es B" en la forma "IF X es A THEN Y es B", debe representarse a cada una de dichas proposiciones por conjuntos difusos con funciones de pertenencia uA(x) y uB(y) respectivamente, y entonces la proposición combinada estará representada por un conjunto difuso A=>B, cuya función de pertenencia estará dada por

uA=>B(x,y) = 1 - min(uA(x),(1 - uB(y))) ó bien
uA>B(x,y) = max(1 - uA(x),uB(y))

No obstante, las expresiones anteriores (implicaciones lógicas o implicaciones IF-THEN) no son necesariamente las más útiles para efectuar inferencias, particularmente en aplicaciones de ingeniería. La razón puede hallarse revisando la tabla anterior: La implicación de la lógica tradicional es verdadera en tres condiciones, y sólo es falsa si la primera proposición es verdadera y la segunda es falsa, lo que puede interpretarse con la máxima "La verdad nunca implica falsedad".

La tabla de verdad de la implicación indica en qué condiciones un razonamiento es formalmente correcto, pero no necesariamente útil.

Visto lo anterior, se concluye que las expresiones de implicación que son útiles para efectuar inferencias lógicas son en realidad operadores AND, es decir, T-Normas. Al utilizar T-Normas como implicaciones, llamamos a éstas implicaciones de Ingeniería o Implicaciones AND. Nuevamente, las T-Normas más usadas como implicación son el mínimo y el producto.

Inferencia en Lógica Difusa

La Inferencia lógica consiste en la combinación de proposiciones para producir nuevas proposiciones. Así, al combinar la proposición "X es A" con la proposición "IF X es A THEN Y es B", se puede inferir la proposición "Y es B"

Una inferencia como la presentada en el párrafo anterior sólo es posible en la lógica tradicional si la primera proposición ("X es A") es idéntica a la primera parte de la segunda proposición ("(IF) X es A"); sin embargo, en la lógica difusa estas dos proposiciones no necesariamente deben ser idénticas, debido a que las fronteras de los conjuntos no son precisas. Así, al combinar la proposición "X es A*" con la proposición "IF X es A THEN Y es B", puede obtenerse la proposición "Y es B*"

2.1.2.4 Estructura de los Sistemas de Lógica Difusa

Los mecanismos de Inferencia presentados anteriormente permiten obtener Sistemas Difusos a partir de la combinación de Conjuntos difusos con reglas de la forma IF... THEN...; un Sistema de Lógica Difusa aprovecha esos mecanismos como el motor de cálculo de un sistema cuyas entradas y salidas son números concretos.

La estructura básica de un Sistema de Lógica Difusa se muestra en la figura. El sistema puede recibir varias entradas numéricas y entregar varias salidas numéricas. El bloque Difusor o Interfaz de Fuzzificación se encarga de convertir las entradas en variables difusas definidas sobre el mismo universo de discurso, que son entregados al bloque Sistema de Inferencia; este bloque, apoyado en un conjunto de reglas de la forma IF... THEN... almacenadas en la Base de Conocimientos o Base de Reglas, las cuales contienen la información del problema, interpreta los datos y produce varios conjuntos difusos para que el bloque Concresor también llamado Interfaz de Defuzzificación los tome y los convierta en salidas numéricas concretas.

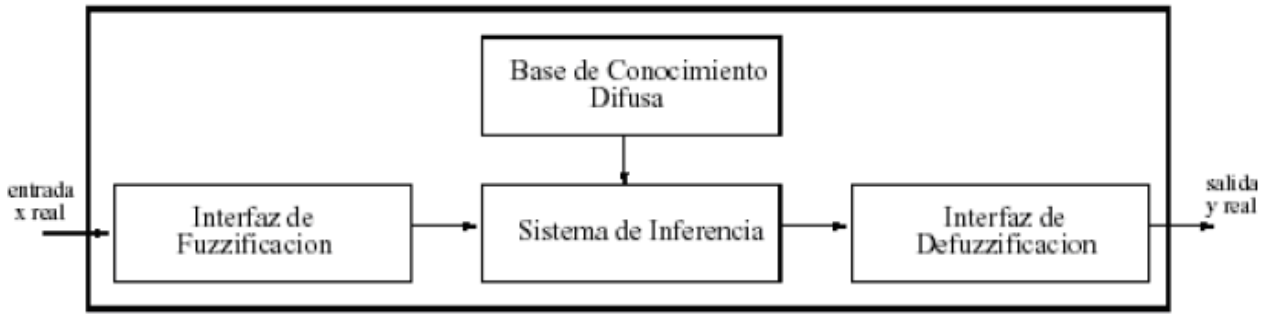


Figura 4. Estructura básica de un Sistema Basado en Reglas Difusas.

Cada una de las variables de entrada y de salida tiene una representación dentro del Sistema de Lógica Difusa en forma de Variables Lingüísticas. Una variable lingüística tiene, entre otras cosas, una colección de atributos que puede adquirir la variable, y cada atributo está representado por un conjunto difuso. Así, por ejemplo, la variable Estatura podría tener tres atributos, Bajo, Mediano y Alto, y cada uno de estos atributos estaría representado por su respectivo conjunto difuso. Estos atributos reciben el nombre de Valores Lingüísticos.

Debido a que un Sistema de Lógica Difusa puede, en general, tener varias entradas y varias salidas, la forma genérica de las reglas presentes en la Base de Reglas es la siguiente:

IF X1 es A1 AND X2 es A2 AND ... AND Xm es Am THEN Y1 es B1 AND Y2
es B2 AND... AND Yn es Bn

En estas reglas, A1,A2,..., Am,B1,B2,...,Bn son Valores Lingüísticos de las Variables Lingüísticas respectivas.

A continuación, se analizará en detalle cada uno de los componentes de un SBRDD.

Base de Conocimientos

La Base de Conocimientos es la parte esencial de los SBRDs debido a que los tres componentes restantes del sistema se ocupan de interpretar las reglas contenidas en ella y de posibilitar su utilización en problemas concretos. Está compuesta por la Base de Reglas Lingüísticas (BRL) y por la Base de Datos (BD):

La BRL está formada por un conjunto de reglas lingüísticas de tipo “SI - ENTONCES”. Ejemplo de ello es el caso de los SBRDs descriptivos con múltiples entradas y una única salida, que presentan la siguiente estructura:

R1 : SI x1 es A11 y … y xm es A1m ENTONCES Y es B1
ADEMAS
R2 : SI x1 es A21 y …y xm es A2m ENTONCES Y es B2
ADEMAS
:::
ADEMAS
Rn : SI x1 es An1 y … y xm es Anm ENTONCES Y es Bn

donde xi e Y son variables lingüísticas de entrada y salida respectivamente, y los Aij y Bi son etiquetas lingüísticas asociadas con conjuntos difusos que determinan su semántica en cada una de las reglas. La BRL está compuesta por una serie de reglas de este tipo unidas por el operador ADEMAS, lo que indica que todas las reglas pueden dispararse ante una entrada concreta.

La BD contiene la definición de los conjuntos difusos asociados a los términos lingüísticos empleados en las reglas de la BRL. Además, almacena los valores de los factores de escala que efectúan las transformaciones necesarias para trasladar los universos de discurso en los que están definidos dichos conjuntos a aquellos en que se definen las variables de entrada y salida del sistema.

En los SBRD aproximativos, sin embargo, este último elemento no es necesario puesto que las variables que se emplean en la reglas no son lingüísticas sino difusas. La razón es obvia puesto que, al tomar cada variable como valor de un conjunto difuso distinto para cada regla y no etiqueta lingüística, no tiene sentido establecer una relación entre etiquetas y conjuntos difusos. En este tipo de sistemas, la Base de Conocimientos queda reducida a una Base de Reglas Difusas (BRD) compuesta por un conjunto de reglas difusas aproximativas conectadas entre sí mediante el operador ADEMAS.

Interfaz de Fuzzificación

La Interfaz de Fuzzificación permite al SBRD trabajar con entradas y salidas reales. Tiene como tarea establecer una correspondencia entre cada valor preciso del espacio de entrada y un conjunto difuso definido en el universo de discurso de dicha entrada. Es por ello que trabaja del siguiente modo:

A ' = F (x0)

donde x0 es un valor preciso de entrada al sistema definido en el universo de discurso
U, A' es un conjunto difuso definido sobre el mismo dominio y F es un operador de fuzzificación.

Principalmente, existen dos posibilidades para la elección de F:
1. Fuzzificación puntual: Es la más empleada por su sencillez. A' se construye como un conjunto difuso puntual con soporte x0, es decir, con la siguiente función de pertenencia:

$$A'(x) = \begin{cases} 1, & si \;\; x = x_0 \\ 0, & en \;\; otro \;\; caso \end{cases}$$

2. Fuzzificación no puntual o aproximada: En este caso, A' (x0) = 1 y el grado de pertenencia de los valores restantes de U va disminuyendo según se alejan éstos de x0. Este segundo tipo de operador de fuzzificación permite el empleo de distintos tipos de funciones de pertenencia. Por ejemplo, en el caso de una función de pertenencia triangular, se puede emplear el siguiente:

$$A'(x) = \begin{cases} 1 - \dfrac{|x - x_0|}{\sigma}, & si \;\; |x - x_0| \le \sigma \\ 0, & en \;\; otro \;\; caso \end{cases}$$

Sistema de Inferencia

El Sistema de Inferencia es el componente encargado de llevar a cabo el proceso de inferencia difuso. Es así como se hace uso de principios de la Lógica Difusa para establecer una aplicación entre

conjuntos difusos definidos en U= U1x, U2x…xUn y conjuntos difusos definidos en V (donde U1,…,Un y V son los dominios en los que están definidas las variables de entrada x1,…, xn y la de salida Y, respectivamente). Este proceso de inferencia está basado en la aplicación del Modus Ponens Generalizado, extensión del Modus Ponens de la Lógica Clásica. Dicho mecanismo difuso de inferencia se pone en práctica mediante la Regla Composicional de Inferencia, que en su expresión más sencilla, toma la siguiente forma:

$$\mu_{B'}(y) = I(\mu_A(x), \mu_B(y))$$

al ser aplicada sobre una regla de tipo SI x1 es A1 y… y xm es Am ENTONCES Y es B, donde μA (x) = T(μA1 (x),…, μAn (x)), T es un operador de conjunción difuso e I es un operador de implicación difuso.

De aquí puede deducirse claramente el hecho de que el proceso de inferencia difusa se aplica a nivel de reglas individuales. De esta manera, una vez aplicada la inferencia sobre las n reglas Ri que componen la BRL, se obtienen n conjuntos difusos Bi' que representan las acciones difusas que ha deducido el SBRD a partir de las entradas recibidas.

Interfaz de Defuzzificación

Se ha comentado que si la BRL se compone de n reglas, el proceso de inferencia devuelve n conjuntos difusos. Puesto que el sistema debe proporcionar una salida precisa, la Interfaz de Defuzzificación debe asumir la tarea de agregar la información aportada por cada uno de los conjuntos difusos individuales y transformarla en un valor preciso. Para efectuar esta agregación existen dos formas de trabajo diferentes:

1. Modo A-FATI: agregar primero, defuzzificar después. En este caso, la interfaz de Defuzzificación lleva a cabo las siguientes tareas:

a) Agrega los conjuntos difusos individuales inferidos (Bi') para obtener un conjunto difuso final (B'). Para ello, emplea un operador de agregación difuso G que, modelando el operador ADEMAS; relaciona las reglas de la base:

$$\mu_{B'}(y) = G\left\{\mu_{B_i'}(y), ..., \mu_{B_n'}(y)\right\}$$

b) Mediante un método de defuzzificación D, transforma el conjunto difuso B' obtenido en un valor preciso y0, que será proporcionado como salida global del sistema:

$$y_0 = D\left(\mu_{B'}(y)\right)$$

2. Modo B-FITA: defuzzificar primero, agregar después. Este segundo modo de trabajo considera individualmente la contribución de cada conjunto difuso inferido y el valor preciso final se obtiene mediante una operación (una media, una suma ponderada o la selección de uno de ellos, entre otras) sobre un valor preciso característico de cada uno de los conjuntos difusos individuales. De este modo, se evita el cálculo del conjunto difuso final B', hecho que ahorra una gran cantidad de tiempo computacional. Este modo de operación supone una aproximación distinta al concepto representado por el operador ADEMAS.

Cuando se trabaja en Modo A-FATI los métodos de defuzzificación más habituales son: el Centro de Gravedad, el Centro de Sumas (aproximación al Centro de Gravedad computacionalmente más rápida de obtener) y la Media de los Máximos. Los operadores de agregación considerados suelen ser el máximo y el mínimo.

En caso de emplear el Modo B-FITA, los operadores de agregación más usados son la media, la media ponderada o la selección de algún valor característico de los conjuntos difusos en función del grado de importancia de la regla que los ha generado en el proceso de inferencia. Como métodos para extraer valores representativos se suelen utilizar el Centro de Gravedad y el Punto de Máximo Criterio; y como grados de importancia de la regla, el área y la altura del conjunto difuso inferido o el grado de emparejamiento de los antecedentes de la misma entrada al sistema. El operador más empleado dentro de este grupo es la media ponderada por el grado de emparejamiento, que se suele combinar con el Centro de Gravedad como valor característico del conjunto difuso.

2.1.2.5 Aprendizaje

Se puede decir que los Sistemas que usan Lógica Difusa logran un determinado aprendizaje a través de las reglas que se definen, las cuales modelan el comportamiento de un elemento, o los requisitos del problema en general, en dependencia de las características y los requerimientos del mismo. Con estas reglas difusas se realizan operaciones como la inferencia, con las cuales son capaces de

generalizar, inferir y llegar a nuevas reglas y comportamientos, de aquí surge la nueva información que aprende el sistema. Por ejemplo en un robot, donde se apliquen varias técnicas de aprendizaje, se puede usar un bloque de lógica difusa para definir la función objetivo del algoritmo que entrena la red neuronal. La ventaja de utilizar una función objetivo difusa está en que se pueden usar parámetros difusos que determinan el comportamiento que tendrá el robot. Dependiendo de estos parámetros el robot se comportará de distintas formas: moviéndose poco para ahorrar batería, moviéndose cerca del punto de partida para luego regresar, maximizando el área cubierta para realizar reconocimiento de entornos, etc. De esta forma, mediante cuatro valores reales, se determinará el comportamiento del robot.

Entrenamiento de Sistemas de Lógica Difusa

Cuando un Sistema de Lógica Difusa cuenta con un mecanismo de entrenamiento, se dice que es un Sistema Difuso Adaptativo. Los mecanismos de entrenamiento son algoritmos que le permiten al sistema cambiar su diseño para ajustarse (esto es, para adaptarse) a algunas exigencias específicas.

En general, los algoritmos de entrenamiento diseñan sólo una parte del Sistema de Lógica Difusa, generalmente la Base de Reglas, o la definición de las Variables Lingüísticas, o en algunos casos ambas cosas. Los demás parámetros los debe seleccionar el usuario.

A nivel mundial, éste es uno de los temas sobre el que más se investiga actualmente dentro de los tópicos de Lógica Difusa. Existen diversos algoritmos, y distintas estrategias dependiendo de la utilización que se le esté dando al Sistema de Lógica Difusa. Pero, qué es lo que justifica este esfuerzo a nivel mundial existen por lo menos las siguientes razones:

En primer lugar, ciertos Sistemas de Lógica Difusa son Aproximadores Universales, es decir, satisfacen una propiedad según la cual se sabe que cualquier función real continua puede ser aproximada con el grado de precisión que se desee por uno de estos Aproximadores.

Esta propiedad asegura entonces la existencia de un Sistema de Lógica Difusa con el que se puede representar, tan bien como se quiera, cualquier función no lineal continua. Sin embargo, aunque se sabe que tal Sistema existe, no se conoce un procedimiento exacto para saber cuál es. En general, los algoritmos de entrenamiento son procedimientos lógicos que intentan diseñar un Sistema de Lógica Difusa que aproxime alguna función desconocida.

En segundo lugar, un Sistema puede estar basado principalmente en el conocimiento (expresado lingüísticamente) que se tiene sobre un cierto problema. Sin embargo, en muchas ocasiones este conocimiento es insuficiente, o se encuentra acompañado de información numérica. Tal es el caso de muchas plantas industriales, donde además de un conocimiento general sobre el comportamiento de la planta, pueden existir registradores que midan y almacenen algunas de las variables del proceso.

Los algoritmos de entrenamiento son capaces de incorporar esta información numérica, junto con la información lingüística en un mismo Sistema de Lógica Difusa. Esta unión de los dos tipos de conocimiento, lingüístico y numérico, en un mismo marco conceptual, hace de los Sistemas Difusos Adaptativos algo excepcional.

Un Sistema Difuso Adaptativo puede entonces intentar diseñarse él mismo para cumplir una función específica. Esta propiedad de auto organización hace que sea sensato proponer los Sistemas de Lógica Difusa como solución a problemas complejos, en los que las representaciones matemáticas exactas no se conocen, o son lo suficientemente complicadas como para que no sea práctico emplearlas.

La investigación en algoritmos de entrenamiento es uno de los temas de mayor auge en la actualidad, hasta el momento existen dos tipos de algoritmos.

Los sistemas con Algoritmos Descriptivos, son aquellos que cuentan con un diseño fácil de entender para una persona que no haya participado en el diseño del mismo, porque la forma en que ha quedado expresado el conocimiento en forma de reglas es clara, y la definición de los valores lingüísticos corresponde también a conceptos sencillos. Este diseño es posible emplearlo para interpretar el sistema diseñado.

Otros algoritmos de entrenamiento diseñan el sistema en forma tal, que una vez concluido el diseño es virtualmente imposible entenderlo, es decir, es imposible extraer conocimiento lingüístico del sistema diseñado. Estos son los denominados Algoritmos Aproximativos.

2.1.3 Redes Bayesianas

Las redes bayesianas o probabilísticas se fundamentan en la teoría de la probabilidad y combinan la potencia del teorema de Bayes con la expresividad semántica de los grafos dirigidos; permiten

representar un modelo causal por medio de una representación gráfica de las independencias/dependencias entre las variables que forman parte del dominio de aplicación. (40)

Una red bayesiana es un grafo acíclico dirigido (las uniones entre los nodos tienen definidas una dirección) en el que los nodos representan variables aleatorias y las flechas representan influencias causales; el que un nodo sea padre de otro implica que es causa directa del mismo.

Una Red Bayesiana se puede interpretar de dos formas:

1. *Distribución de probabilidad*: Representa la distribución de la probabilidad conjunta de las variables representadas en la red.
2. *Base de reglas*: Cada arco representa un conjunto de reglas que asocian a las variables involucradas. Dichas reglas están cuantificadas por las probabilidades respectivas.

2.1.3.1 Definición

Una red bayesiana es un grafo acíclico dirigido en el que los nodos representan variables aleatorias que pueden ser continuas o discretas; en las siguientes definiciones se utilizarán letras mayúsculas para denotar los nodos (X) y las correspondientes letras minúsculas para designar sus posibles estados (x_i).

Los estados que puede tener una variable deben cumplir con dos propiedades:

1. Ser *mutuamente excluyentes*, es decir, un nodo sólo puede encontrarse en uno de sus estados en un momento dado.
2. Ser un conjunto *exhaustivo*, es decir, un nodo no puede tener ningún valor fuera de ese conjunto.

2.1.3.2 Representación del conocimiento

Una red bayesiana representa relaciones causales en el dominio del conocimiento a través de una estructura gráfica y las tablas de probabilidad condicional entre los nodos, por lo tanto el conocimiento que representa la red está compuesto por los siguientes elementos:

1. Un conjunto de nodos $\{X_i\}$ *que representan cada una de las variables del modelo*. Cada una de ellas tiene un conjunto exhaustivo de estados $\{x_i\}$ mutuamente excluyentes.

2. *Un conjunto de enlaces o arcos* (X_i, X_j) *entre aquellos nodos que tienen una relación causal*. De esta manera todas las relaciones están explícitamente representadas en el grafo.
3. *Una tabla de probabilidad condicional asociada a cada nodo* X_i *indicando la probabilidad de sus estados para cada combinación de los estados de sus padres*. Si un nodo no tiene padres se indican sus probabilidades a priori.

2.1.3.3 Estructura de una Red Bayesiana

1. Se asigna un vértice o nodo a cada variable (X_i) y se indica de qué otros vértices es una causa directa; a ese conjunto de vértices "causa del nodo X_i " se lo denota como el conjunto π_{X_i} , y se lo llamará "padres de X_i ".
2. Se une cada padre con sus hijos con flechas que parten de los padres y llegan a los hijos.
3. A cada variable X_i se le asigna una matriz $P(x_i|\pi_{X_i})$ que estima la probabilidad condicional de un evento $X_i = x_i$ dada una combinación de valores de los π_{X_i} .

Después que se ha diseñado la estructura de la red y se han especificado todas las tablas de probabilidad condicional se está en condiciones de conocer la probabilidad de una determinada variable dependiendo del estado de cualquier combinación del resto de variables de la red; para lo cual se debe calcular la probabilidad a posteriori de cada variable condicionada a la evidencia; estas probabilidades a posteriori se podrán obtener de forma inmediata a partir de la probabilidad conjunta de todas las variables $P(x_1, x_2, \dots, x_i)$. A continuación se indica cómo este proceso se ve simplificado al aplicar la propiedad de independencia condicional que permite obtener la probabilidad conjunta a partir de las probabilidades condicionales de cada nodo en función de sus padres.

Independencia condicional

La topología o estructura de una red bayesiana, además de representar explícitamente dependencias probabilísticas entre variables, describe implícitamente las independencias condicionales existentes entre ellas. La siguiente definición muestra las condiciones que deben darse para que dos variables sean condicionalmente independientes (41):

Una variable *X* es condicionalmente independiente de otra *Y* dada una tercer variable *Z,* si el conocer *Z* hace que *X* e *Y* sean independientes. Es decir, si conozco *Z, Y* no tiene influencia en *X.* Esto es: P(X | Y, Z) = P(X | Z).

Esta definición se traduce a que cada variable es independiente de todos aquellos nodos que no son sus "descendientes" una vez que se conocen sus propios nodos padres; a lo largo de este trabajo se utilizarán las palabras "nodos" y "variables" como sinónimos. Gráficamente se verifica en los casos en que los nodos X e Y están separados por Z en el grafo. Esto implica que todos los caminos para ir de X a Y pasarán necesariamente por Z. (40)

Inferencia

La inferencia es el proceso de introducir nuevas observaciones y calcular las nuevas probabilidades que tendrán el resto de las variables; por lo tanto dicho proceso consiste en calcular las probabilidades a posteriori $P(X \mid Y = y_i)$ de un conjunto de variables *X* después de obtener un conjunto de observaciones $Y = y_i$ (donde *Y* es la lista de variables observadas, y_i es la lista correspondiente a los valores observados). El fundamento matemático en el que se basan las redes probabilísticas para llevar a cabo la inferencia es el *Teorema de Bayes* que se expresa como:

$$P(y_j \mid x_i) = \frac{P(y_j)P(x_i \mid y_j)}{P(x_i)} = \frac{P(y_j)P(x_i \mid y_j)}{\sum_j P(x_i \mid y_j)P(y_j)}$$

En la práctica, esto no es viable por el tiempo necesario para llevarlo a cabo ya que incrementar el número de nodos de la red aumentaría exponencialmente el número de sumas necesarias, por ello se han desarrollado diversos algoritmos de propagación.

Algoritmos de propagación.

Existen varios métodos computacionales que aprovechan la estructura gráfica para propagar los efectos que las observaciones del mundo real tienen sobre el resto de las variables de la red; las diferencias entre ellos se basan principalmente en la precisión de los resultados y en el consumo de recursos durante el tiempo de ejecución. Los algoritmos de propagación se dividen inicialmente en "exactos" o "aproximados" según cómo calculen los valores de las probabilidades. Los métodos exactos calculan los valores por medio del Teorema de Bayes mientras que los aproximados utilizan técnicas iterativas de muestreo en las que los valores se aproximarán más o menos a los exactos dependiendo del punto en que se detenga el proceso.

Los algoritmos de propagación dependen del tipo de estructura de la red bayesiana, existiendo las siguientes tres topologías de red:

- Árboles
- Poliárboles
- Redes multiconectadas

2.1.3.4 Aprendizaje

El Aprendizaje de Redes Bayesianas resulta de gran importancia porque es práctico y además provee un enfoque de comprensión y diseño de otros algoritmos.

Características**:**

- Cada nuevo ejemplo puede aumentar o disminuir la estimación de una hipótesis (flexibilidad - incrementalidad).
- Conocimiento a priori se puede combinar con datos para determinar la probabilidad de las hipótesis.
- Da resultados con probabilidades asociadas.
- Puede clasificar combinando las predicciones de varias hipótesis.
- Sirve de estándar de comparación de otros algoritmos.

Problemas:

- Se requieren conocer muchas probabilidades.
- Es computacionalmente caro (depende linealmente del número de hipótesis).

Lo que normalmente se quiere saber en aprendizaje es cuál es la mejor hipótesis (más probable) dados los datos.

Si denotamos P(D) como la probabilidad a priori de los datos (ej. cuales datos son más probables que otros), P(D | h) la probabilidad de los datos dada una hipótesis, lo que queremos estimar es: P(h | D), la probabilidad posterior de h dados los datos. Esto lo podemos estimar con Bayes.

Teorema de Bayes:

$$P(h \mid D) = \frac{P(D \mid h)P(h)}{P(D)}$$

Para estimar la hipótesis más probable o MAP (máximum a posteriori hypothesis):

$$h_{MAP} = argmax_{h \in H} (P(h \mid D))$$
$$= argmax_{h \in H} \left(\frac{P(D|h)P(h)}{P(D)} \right)$$
$$= argmax_{h \in H} (P(D \mid h)P(h))$$

Ya que *P(D)* es una constante independiente de *h*.

Si asumimos que todas las hipótesis son igualmente probables, entonces nos queda la hipótesis de máxima verosimilitud o ML (máximum likelihood):

$$h_{ML} = argmax_{h \in H} (P(D \mid h))$$

Proceso de aprendizaje.

El aprendizaje en la redes bayesianas consiste en definir la red probabilística a partir de datos almacenados en bases de datos en lugar de obtener el conocimiento del experto. Este tipo de aprendizaje ofrece la posibilidad de inducir la estructura gráfica de la red a partir de los datos observados y de definir las relaciones entre los nodos basándose también en dichos casos; según Pearl (40) a estas dos fases se las puede denominar respectivamente *aprendizaje estructural* y *aprendizaje paramétrico*. A continuación se resume cada una de estas dos fases:

- **Aprendizaje estructural**: obtiene la estructura de la red bayesiana a partir de bases de datos, es decir, las relaciones de dependencia e independencia entre las variables involucradas. Las técnicas de aprendizaje estructural dependen del tipo de estructura o topología de la red (árboles, poliárboles o redes multiconectadas). Otra alternativa es combinar conocimiento subjetivo del experto con aprendizaje, para lo cual se parte de la estructura dada por el experto y se la valida y mejora utilizando datos estadísticos.
- **Aprendizaje paramétrico**: dada una estructura y las bases de datos, obtiene las probabilidades a priori y condicionales requeridas.

Algunos estudios sobre el aprendizaje de redes bayesianas indican que el requisito principal para poder realizar la tarea de aprendizaje a partir de datos es disponer de bases de datos muy extensas en las que esté especificado el valor de cada variable en cada uno de los casos. Por otro lado, el

aprendizaje a partir de bases de datos incompletas generalmente consiste en inferir de alguna manera los datos ausentes para completar la base de datos, algunas investigaciones (42) estiman los límites del conjunto de datos ausentes y obtienen un único punto de estimación que es modificado con diferentes pesos dependiendo del patrón que supuestamente siguen dichos datos, luego se construye una base de datos completa y se procede como se indico anteriormente.

Aprendizaje Paramétrico.

El aprendizaje paramétrico consiste en encontrar los parámetros asociados a una estructura dada de una red bayesiana. Dichos parámetros serían las probabilidades a priori de los nodos raíz y las probabilidades condicionales de las demás variables dados sus padres; si se conocen todas las variables es fácil obtener las probabilidades requeridas ya que las probabilidades previas corresponden a las marginales de los nodos raíz y las condicionales se obtienen de las conjuntas de cada nodo con su(s) padre(s). Para que se actualicen las probabilidades con cada caso observado, éstas se pueden representar como razones enteras y actualizarse con cada observación.

Datos completos

El aprendizaje de los parámetros es simple cuando todas las variables son completamente observables en el conjunto de entrenamiento. El método más común es el llamado *estimador de máxima verosimilitud*, que consiste sencillamente en estimar las probabilidades deseadas a partir de la frecuencia de los valores de los datos de entrenamiento, de forma análoga a como se hace en Naive Bayes.

La calidad de estas estimaciones dependerá de que exista un *número suficiente de datos* en la muestra. Cuando esto no es posible se puede cuantificar la incertidumbre existente representándola mediante una distribución de probabilidad, para así considerarla explícitamente en la definición de las probabilidades. Habitualmente se emplean distribuciones Beta en el caso de variables binarias, y distribuciones Dirichlet para variables multivaluadas. Esta aproximación es útil cuando se cuenta con el apoyo de expertos en el dominio de la aplicación para concretar los valores de los parámetros de las distribuciones.

Si existen variables de tipo continuo la estrategia más habitual es aplicar discretizarlas antes de construir el modelo estructural. Existen algunos modelos de redes bayesianas con variables continuas, pero están limitados a variables gaussianas relacionadas linealmente. Es posible también efectuar la

discretización mientras se construye el grafo de la red, si éste se aprende utilizando el principio MDL como medida de ajuste.

Datos incompletos

Aparecen mayores dificultades cuando los datos de entrenamiento no están completos. Pueden plantearse dos tipos de información incompleta:

- Valores faltantes: faltan algunos valores de uno o varias variables en algunos ejemplos.
- Nodo oculto: faltan todos los valores de una variable.

El primer caso es más sencillo, y existen varias alternativas, entre ellas:

- Eliminar los ejemplos con valores ausentes.
- Considerar un nuevo valor adicional para la variable: 'desconocido'.
- Considerar el valor más probable a partir de los datos de la misma en las demás instancias.
- Considerar el valor más probable en base a las demás variables.

Las dos primeras opciones son habituales en problemas de aprendizaje, y válidas siempre y cuando se cuente con un número elevado de datos completos. La tercera opción viene a ignorar las posibles dependencias de la variable con las demás, cuando ya se cuenta con la estructura que las describe en el grafo; no suele proporcionar los mejores resultados.

La cuarta técnica se sirve de la red ya conocida para inferir los valores desconocidos. Primero se rellenan las tablas de parámetros usando todos los ejemplos completos. Después, para cada instancia incompleta, se asignan los valores conocidos a las variables correspondientes en la red y se propaga su efecto para obtener las probabilidades a posteriori de las no observadas. Entonces se toma como valor observado el más probable y se actualizan todas las probabilidades del modelo antes de procesar la siguiente instancia incompleta.

La aparición de nodos ocultos requiere un tratamiento más complejo. Existen diferentes técnicas para estima las probabilidades faltantes en este caso. Una habitual es la aplicación del algoritmo EM (Expectation Maximization), cuya aplicación ya se ha estudiado en la asignatura en el contexto de las técnicas de agrupamiento. Su aplicación al aprendizaje de parámetros se traduce en lo siguiente:

- Asignar valores aleatorios (o basados en conocimiento experto, si se dispone de él) a las probabilidades desconocidas de la red.

- Utilizar los datos conocidos para estimar desconocidos infiriéndolos sobre el modelo con las probabilidades actuales.
- Completar el conjunto de datos con los valores estimados y volver a calcular las probabilidades de la red a partir de ellos.
- Repetir los dos pasos anteriores hasta que no haya cambios significativos en las probabilidades.

Aprendizaje Estructural.

El aprendizaje estructural conlleva explorar un espacio de grafos. Esta tarea es muy compleja. A poco que se incrementa el número de variables (nodos), el número de posibles grafos a construir con ellas se dispara. Por eso en muchas ocasiones se restringe el espacio de búsqueda a grafos con características concretas. Existen muchos algoritmos específicos para el aprendizaje de redes donde G se limita a un árbol, o a un poliárbol, o a otras estructuras menos generales.

No obstante, existen técnicas para aprender redes con estructuras generales. Trabajar sin restricciones debería permitir, idealmente, construir redes que ajusten mejor al conjunto de entrenamiento, por complejas que sean las dependencias entre los atributos.

Hay dos aproximaciones básicas al aprendizaje de redes sin restricciones. La primera de ellas reúne métodos que exploran las relaciones de dependencia existentes entre pares, tripletas u otros subconjuntos de variables para elegir la forma en que deben conectarlas. El estudio de esas relaciones requiere establecer un criterio cuantitativo para medir la dependencia entre variables, y es dicho criterio el que guía la construcción de la red. Un ejemplo de algoritmo que se engloba en esta familia de técnicas es el de construcción de TAN de Friedman y Goldszmidt. (43)

Dicho algoritmo crea una red con una topología restringida, pero el principio que guía la construcción es el mismo, una medida de información mutua que cuantifica la relación entre las variables.

La otra aproximación habitual al aprendizaje de redes consiste en realizar una búsqueda guiada por una medida global de calidad. Nótese que en la aproximación anterior el criterio guía es local, se aplica a subconjuntos reducidos de variables, no a toda la red. En esta otra aproximación, la operación general consiste en generar distintos grafos mediante un algoritmo de búsqueda y aplicar a cada uno de ellos una función de medida de calidad para decidir qué grafo conservar en cada paso.

Existen muchos algoritmos que siguen esta técnica, definidos a partir de la combinación de dos elementos:

- Algoritmo de búsqueda
- Medida global de ajuste

Es habitual emplear *algoritmos de búsqueda* heurística. Intentar una búsqueda exhaustiva por todo el espacio de grafos es sencillamente intratable. Algunas posibilidades son las técnicas de ascenso de colinas (*hill climbing)*, algoritmos genéticos, búsquedas bidireccionales, etc. Otra opción es aplicar una búsqueda voraz. Se comienza con una red vacía y se aplican sucesivas operaciones locales mejorando de forma maximal la medida de ajuste hasta que se encuentra un óptimo local. Las operaciones aplicadas incluyen la adición, borrado e inversión de arcos.

También hay muchas *medidas de ajuste*. Dos habituales son la medida bayesiana y el principio de mínima longitud de descripción. La medida bayesiana trata de maximizar la probabilidad de la estructura dados los datos de entrenamiento P(Bs | D). Como el objetivo de la medida es comparar el valor obtenido para distintas estructuras *i* y *j*, es habitual recurrir al cociente:

$$P(B_{Si} \mid D) \,/\, P(_{BSj} \mid D) = P(B_{Si}, D) \,/\, P(B_{Sj}, D)$$

Considerando variables discretas y datos independientes, las probabilidades conjuntas del segundo cociente se pueden estimar utilizando las predicciones hechas por cada estructura ante los datos de entrenamiento.

Por su parte, el *principio MDL* caracteriza el aprendizaje en términos de compresión de los datos. El objetivo del aprendiz es encontrar un modelo que facilite la obtención de la descripción más corta posible de los datos originales. La longitud de esta descripción toma en cuenta:

- La descripción del propio modelo, penalizando la complejidad del mismo.
- La descripción de los datos que usan el modelo, alentando su verosimilitud.

En el contexto de las redes bayesianas, el modelo es la red. Dicha red B describe la probabilidad condicional PB sobre las instancias que aparecen en los datos. Usando esta distribución, puede construirse y codificarse un esquema que asigne palabras de código más cortas a las instancias más probables. De acuerdo con el principio MDL, debería escogerse una red B tal que la longitud combinada de la descripción de la red y los datos codificados (con respecto a PB) sea mínima.

A partir de este punto, distintos autores definen diferentes formas de medir cada elemento de la descripción a partir del esquema general:

MDL (B | D) = complejidad (B) – verosimilitud (D), en el que un valor menor MDL (B | D) de es mejor.

Por ejemplo, en (43) se define:

$$MDL(B \mid D) = \frac{|B| \cdot \log(N)}{2} - N \cdot \sum_{i=1}^{N} \log(P_B(d_i))$$

El primer término (complejidad (B)) representa la longitud de una descripción de la red en la que se emplean *½ logN bits* para cada parámetro (N es el tamaño de D), siendo |B| el número total de éstos. El segundo término mide cuántos bits se necesitan para describir todos los elementos d*i* en D asignando las longitudes de código en función de la distribución de probabilidad P*B*.

Tanto la medida bayesiana como MDL son bien conocidas y están bien estudiadas. Ambas funciones son asintóticamente equivalentes cuando aumenta el tamaño de la muestra, y además asintóticamente correctas: con probabilidad igual a uno la distribución aprendida converge a la distribución subyacente a medida que el número de muestras aumenta.

2.1.4 Algoritmos Genéticos.

Los Algoritmos Genéticos, como ya se ha dicho anteriormente, son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. Las poblaciones evolucionan en la naturaleza, a lo largo de las generaciones, acorde con los principios de la selección natural y la supervivencia de los más fuertes, postulados por Darwin. Imitando este proceso, los Algoritmos Genéticos son capaces de ir creando soluciones para problemas del mundo real. La evolución de dichas soluciones hacia valores óptimos del problema depende en buena medida de una adecuada codificación de las mismas.

Un algoritmo genético consiste en una función matemática o una rutina de software que toma como entradas a los ejemplares y retorna como salidas cuáles de ellos deben generar descendencia para la nueva generación.

2.1.4.1 Funcionamiento

Los Algoritmos Genéticos usan una analogía directa con el comportamiento natural. Trabajan con una población de individuos, cada uno de los cuales representa una solución factible a un problema dado.

A cada individuo se le asigna un valor ó puntuación, en dependencia de la bondad de dicha solución. En la naturaleza esto equivaldría al grado de efectividad de un organismo para competir por unos determinados recursos. Cuanto mayor sea la adaptación de un individuo al problema, mayor será la probabilidad de que el mismo sea seleccionado para reproducirse, cruzando su material genético con otro individuo seleccionado de igual forma. Este cruce producirá nuevos individuos, descendientes de los anteriores los cuales comparten algunas de las características de sus padres. Cuanto menor sea la adaptación de un individuo, menor será la probabilidad de que dicho individuo sea seleccionado para la reproducción, y por tanto de que su material genético se propague en sucesivas generaciones. De esta manera se produce una nueva población de posibles soluciones que reemplaza a la anterior y verifica la interesante propiedad de que contiene una mayor proporción de buenas características en comparación con la población anterior. Así a lo largo de las generaciones las buenas características se propagan a través de la población. Favoreciendo el cruce de los individuos mejor adaptados, van siendo exploradas las áreas más prometedoras del espacio de búsqueda. Si el Algoritmo Genético ha sido bien diseñado, la, población convergerá hacia una solución óptima del problema.

El siguiente segmento de seudo-código simula el proceso (44):

```
Generar población inicial, G (0);
Evaluar G(0);
t:=0;
Repetir
t:=t+1;
Generar G(t) usando G(t-1);
Evaluar G(t);
Hasta encontrar una solución;
```

Inicialmente la población se genera de manera aleatoria, y estará compuesta por un conjunto de cromosomas, pudieran ser caracteres que representen las posibles soluciones de un problema. En caso de no hacerlo de forma aleatoria, es importante garantizar que dentro de la población inicial, se tenga una representación de la mayor parte de la población posible o al menos evitar la convergencia prematura. A cada miembro de la población se le aplicará una función de aptitud con el propósito de saber que tan buena es la solución que se está codificando, o sea para ver su comportamiento. Posteriormente se procede a hacer la selección de los cromosomas que van a cruzarse en la próxima generación, como es lógico se escogerán los mejores. El cruzamiento es el principal operador genético, representa la reproducción sexual, opera sobre dos cromosomas a la vez para generar dos

descendientes donde se combinan las características de ambos cromosomas padres. De este cruzamiento surgirá el nuevo conjunto de cromosomas a los que se le aplicará el proceso anterior. También existe un operador de mutación que cambiará de manera aleatoria un alelo ("un bit de cadena representativa") de un cromosoma. Esto garantiza la inserción de nuevo material cromosómico y la interconexión total del espacio de búsqueda. El AG se ejecuta una cantidad determinada de generaciones o hasta que se estabilice la población (cuando todos o la mayoría de los individuos tengan la misma actitud).

Características

- Son algoritmos estocásticos. Dos ejecuciones pueden dar dos soluciones diferentes.
- Son algoritmos de búsqueda múltiple de los cuales se obtienen varias soluciones como respuesta.
- Son algoritmos que hacen una barrida mayor al subespacio de posibles soluciones válidas.
- En los algoritmos genéticos la convergencia del algoritmo es poco sensible a la población inicial si esta se escoge de forma aleatoria y es lo suficientemente grande, a diferencia de otros algoritmos donde la convergencia y el resultado final dependen fuertemente de la posición inicial.
- Debido a su grado de penetración casi nulo, la curva de convergencia asociada al algoritmo presenta una convergencia excepcionalmente rápida al principio, que casi enseguida se bloquea.
- Es una búsqueda paramétricamente robusta, es decir, para que no converja hay que escoger realmente mal los parámetros del algoritmo. Con tasas razonables, va a converger en una solución razonablemente buena si la representación es la adecuada.
- Los algoritmos genéticos son intrínsecamente paralelos, o sea, independientemente de lo que se haya implementado de forma paralela o no, buscan en diferentes puntos del espacio de soluciones de forma paralela. Ese paralelismo intrínseco permite que sea fácil modificar el código para que se ejecute simultáneamente en varios procesadores.

2.1.4.2 Métodos de representación

Primeramente, para que un algoritmo genético pueda resolver un problema determinado, es necesario algún método que codifique las soluciones potenciales de manera tal que puedan ser procesadas por una computadora. A continuación se enumeran tres de los métodos existentes para este fin cuya virtud es que facilitan la definición de operadores que causen los cambios aleatorios en las soluciones

seleccionadas (cambiar un 1 por un 0 o viceversa, sumar o restar al valor de un número una cantidad elegida al azar, o cambiar una letra por otra.):

1. Un enfoque común es codificar las soluciones como cadenas binarias: secuencias de 0s y 1s donde el dígito de cada posición representa el valor de algún aspecto de la solución.
2. Similar al anterior, consiste en codificar las soluciones como cadenas de enteros o números decimales, donde cada posición representa algún aspecto particular de la solución permitiendo una mayor precisión y complejidad que el método comparativamente restringido de utilizar sólo números binarios.
3. Otro método consiste en representar a los individuos de un Algoritmo Genético como cadena de letras, donde cada letra representa un aspecto especifico de la solución.

2.1.4.3 Operadores Genéticos

Selección:

El proceso de selección natural permite determinar cuáles individuos llegarán a reproducirse, por tanto los que tengan rasgos menos válidos para realizar sus tareas tendrán pocas probabilidades de hacerlo por lo que su patrimonio genético podría desaparecer para siempre.

Existen diversas técnicas para seleccionar a los individuos que deben copiarse hacia la siguiente generación que puede utilizar un algoritmo genético, algunas de ellas son mutuamente excluyentes pero en cambio otras, pueden usarse en combinación, algo que se hace frecuentemente. Las más comunes se listan a continuación (45):

- *Selección elitista*: se garantiza la selección de los miembros más aptos de cada generación.
- *Selección proporcional a la aptitud*: los individuos más aptos tienen más probabilidad de ser seleccionados, pero no la certeza.
- *Selección por rueda de ruleta*: una forma de selección proporcional a la aptitud en la que la probabilidad de que un individuo sea seleccionado es proporcional a la diferencia entre su aptitud y la de sus competidores.
- *Selección escalada*: al incrementarse la aptitud media de la población, la fuerza de la presión selectiva también aumenta y la función de aptitud se hace más discriminadora.

- *Selección por torneo*: se eligen subgrupos de individuos de la población, y los miembros de cada subgrupo compiten entre ellos. Sólo se elige a un individuo de cada subgrupo para la reproducción.
- *Selección por rango*: a cada individuo de la población se le asigna un rango numérico basado en su aptitud, y la selección se basa en este ranking, en lugar de las diferencias absolutas en aptitud.
- *Selección generacional*: la descendencia de los individuos seleccionados en cada generación se convierte en toda la siguiente generación. No se conservan individuos entre las generaciones.
- *Selección por estado estacionario*: la descendencia de los individuos seleccionados en cada generación vuelven al acervo genético preexistente, reemplazando a algunos de los miembros menos aptos de la siguiente generación. Se conservan algunos individuos entre generaciones.
- *Selección jerárquica*: los individuos atraviesan múltiples rondas de selección en cada generación. Las evaluaciones de los primeros niveles son más rápidas y menos discriminatorias, mientras que los que sobreviven hasta niveles más altos son evaluados más rigurosamente.

Cruzamiento y Mutación:

Una vez que han sido seleccionados los mejores individuos, éstos deben ser utilizados para crear la siguiente generación con la esperanza de mejorar su aptitud. Para lograr dicho propósito existen dos estrategias básicas:

La primera es el cruzamiento que no es más que el intercambio de material genético entre dos cromosomas. Implica elegir a dos individuos para que intercambien segmentos de su código, produciendo una ``descendencia'' artificial cuyos individuos son combinaciones de sus padres. Este proceso pretende simular el proceso análogo de la recombinación que se da en los cromosomas durante la reproducción sexual. La forma de realizarlo depende de la representación elegida para los cromosomas o individuos.

Existen varias formas para realizar el cruzamiento, las más comunes incluyen al *cruzamiento de un punto*, en el que se establece un punto de intercambio en un lugar aleatorio del genoma de los dos individuos, y uno de los individuos contribuye todo su código anterior a ese punto y el otro individuo contribuye todo su código a partir de ese punto para producir una descendencia, y al *cruzamiento*

uniforme, en el que el valor de una posición dada en el genoma de la descendencia corresponde al valor en esa posición del genoma de uno de los padres, teniendo los dos la misma probabilidad de que esto ocurra.

También puede hacerse el *cruzamiento teniendo en cuenta más de un punto de cruce*. La implementación de esta otra vía de cruzamiento, garantiza que el cruce basado en dos puntos, representa una mejora, mientras que añadir más puntos de cruce no beneficia el comportamiento del AG. La principal ventaja de hacerlo de esta manera, radica en que el espacio de búsqueda puede ser explorado más fácilmente y la principal desventaja sería que aumenta la probabilidad de ruptura de buenos esquemas.

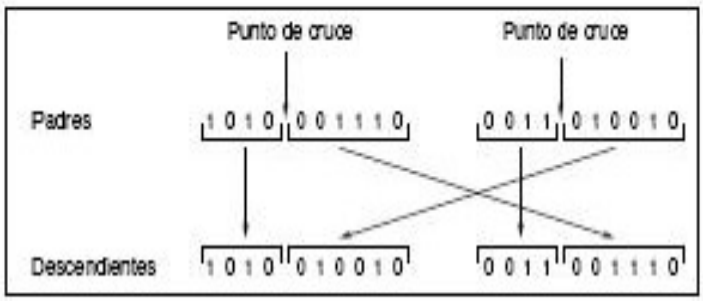


Figura 5. Operador de cruce basado en un punto.

La segunda y más sencilla se llama mutación. Al igual que una mutación en los seres vivos cambia un gen por otro, una mutación en un algoritmo genético también causa pequeñas alteraciones en puntos concretos del código de un individuo y por último, invierte el orden de una sección contigua del cromosoma, recolocando de esta manera el orden en el que se almacenan los genes.

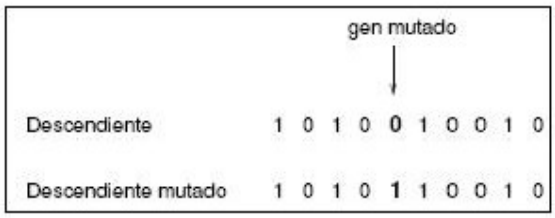


Figura 6. Operador de mutación.

No es conveniente abusar de la mutación. Es cierto que es un mecanismo generador de diversidad, y, por tanto, la solución cuando un algoritmo genético está estancado, pero también es cierto que reduce el algoritmo genético a una búsqueda aleatoria. Siempre es más conveniente usar otros mecanismos de generación de diversidad, como aumentar el tamaño de la población, o garantizar la aleatoriedad de la población inicial.

En resumen, para resolver un problema particular mediante un algoritmo genético, debemos tener las siguientes componentes:

- una codificación o representación genética de las soluciones potenciales del problema (cadenas binarias en el caso del algoritmo genético simple).
- un modo de crear la población inicial (aleatoria en el algoritmo genético simple).
- una función de evaluación que juega el papel del entorno permitiendo clasificar a los individuos en términos de su aptitud.
- operadores genéticos de reproducción, cruce y mutación que simulan la evolución de las poblaciones.
- valores de los parámetros del algoritmo: longitud de los individuos, tamaño de la población, número de generaciones y probabilidad de mutación.

2.1.4.4 Aprendizaje

Los Algoritmos Genéticos constituyen una de las técnicas inteligentes que más se emplean en la actualidad y esto se debe principalmente a sus notables resultados prácticos y su versatilidad. El Aprendizaje Artificial es uno de los temas que más atención ha recibido por parte de la comunidad científica que experimenta con tales técnicas, esta disciplina estudia el proceso de aprendizaje en general, con énfasis en la traslación a las máquinas de lo hasta ahora conocido sobre dicho proceso en los seres vivos. El problema fundamental es que los agentes artificiales con frecuencia deben operar en ambientes no deterministas, dinámicos, y en general, bastantes complejos, por lo que no resulta factible incorporar a priori todo el conocimiento requerido para un rendimiento aceptable; en ocasiones, tal conocimiento ni siquiera se encuentra disponible con antelación. Por ende, resulta imperativo que el agente aprenda sobre el ambiente, utilizando las nuevas percepciones y el conocimiento que posee. Según los mecanismos de inferencia a los que recurra el agente, el proceso se puede catalogar como deductivo, inductivo, o abductivo. En este contexto, el Algoritmo Genético aporta su gran potencial en el manejo de poblaciones de deducciones, observaciones e hipótesis, con las ventajas del paralelismo implícito, y del refinamiento y descubrimiento de información.

Los Algoritmos Genéticos constituyen un conjunto de modelos matemáticos inspirados en la teoría Darwinista de la evolución. Concretamente se dice que un algoritmo genético es un método de búsqueda de soluciones donde su funcionamiento consiste en *evolucionar* una población de potenciales soluciones de un problema con el objetivo de que las soluciones mejor adaptadas prevalezcan y transmitan sus características a soluciones hijas, y luego de una sucesión suficiente de generaciones se debería disponer de soluciones bastante adaptadas al problema particular. Según Mitchell (46): Un agente artificial *aprende* de la experiencia E respecto a alguna clase de tareas T y medida de rendimiento P, si su desempeño en las tareas de T, según la medida P, mejora con la experiencia E.

Para abordar a los Algoritmos Genéticos en cuanto a su relación con el aprendizaje artificial se deben juntar ambas nociones, esto es lo que suele llamarse Aprendizaje Genético (47). Resulta conveniente aclarar que los individuos codifican conocimiento sobre el ambiente en forma de hipótesis o hechos inciertos, la noción de certidumbre siempre está en relación con la base de conocimientos y el procedimiento de demostración del agente. Según esta relación el proceso de aprendizaje se puede clasificar como:

- **Deductivo**

La deducción se expresa (48):

> Si se aceptan p(1) y $\forall X(p(X) \supset q(X))$, entonces, por *deducción*, concluir q(1).

En los Algoritmos Genéticos, la transmisión de la información a los hijos por parte de los progenitores constituye un proceder deductivo, por cuanto el conocimiento codificado en los progenitores es recibido parcial o íntegramente.

Este tipo de inferencia va de una generalización a una particularización, no aporta información semántica a la base de conocimientos, por eso frecuentemente se encuentra asociada a otros paradigmas de aprendizaje. Algunos operadores como el de clonación constituyen una reexpresión de la sentencia deductiva $A \supset A$, una forma simple de probar algo ya que permite preservar el conocimiento de una generación a la otra. Y, si sólo una parte de la información es transmitida al hijo, entonces se estaría siguiendo un enfoque descendente en el que de la generalización del padre se arriba a información específica (parcial) del hijo. Por otro lado, algunas investigaciones recurren a las facultades deductivas de los algoritmos genéticos con fines de modelado de la deducción humana (49).

El aprendizaje genético deductivo se utiliza para diversas aplicaciones tales como la generación evolutiva de rostros, con aplicaciones inmediatas en la Criminalística (50) donde se trata de obtener nuevas imágenes a partir de la combinación de la información más general representada por las imágenes de la población inicial, igualmente se utiliza para la interpretación de datos sísmicos al construir imágenes deductivamente tras incorporar restricciones geológicas y geométricas a los operadores genéticos.

También, en el ámbito de la Lingüística Computacional, Bianchi y Delmonte (51) emplean el aprendizaje deductivo para resolver la cuestión de la ambigüedad de palabras en textos en lenguaje italiano, y para la conversión de un texto en una secuencia de categorías gramaticales.

Otra perspectiva distinta de modelado es desarrollada por Chakraborti (49) cuya investigación reconoce en primer lugar que la noción de *Validez* constituye un nodo central de todas las ciencias, además se plantea que el razonamiento deductivo puede concebirse como la búsqueda evolutiva de la conclusión a partir de las premisas, demostrando que los algoritmos genéticos representan una herramienta excelente para la deducción.

- **Inductivo**

Se habla de aprendizaje genético inductivo cuando el algoritmo genético se usa con el fin de aprender teorías a partir de hechos observados.

En este caso, lo que se busca es obtener individuos que resulten más generales que sus predecesores dependiendo de la información que codifiquen, donde la relación de generalidad estará dada por el problema en particular. El proceso de inducción se define formalmente (49):

> Si se observan los hechos p(1), p(2), p(3), y los únicos objetos(constantes) del mundo son 1, 2 y 3, entonces, por *inducción*, concluir $\forall X p(X)$.

El aprendizaje genético inductivo ha sido empleado para el desarrollo de diversas aplicaciones y estudios en disciplinas tales como la Lingüística Computacional, Echizenya (52) presenta un estudio para la traducción al inglés de textos en japonés donde, a diferencia de los enfoques tradicionales en los que las instancias de aprendizaje se definen previamente, el algoritmo genético crea las instancias para la inducción de las reglas de traducción.

El Aprendizaje Genético Difuso es una combinación de los algoritmos genéticos con otras técnicas de Computación Inteligente para extender el alcance del proceso de aprendizaje donde se conjuntan reglas difusas con la inferencia evolutiva; se han desarrollado diversos estudios pero el aprendizaje inductivo en sistemas de este tipo se dificulta debido a la aparición de un elevado numero de características que incrementan la dimensión del problema. Con un enfoque más práctico se han creado también sistemas clasificadores genéticos difusos para la detección de intrusos en una red con altas tasas de reconocimiento y autonomía.

La combinación de Redes Neuronales Artificiales con Algoritmos Genéticos es conocida como Neuroevolución, y el problema típico es la determinación del número de capas y nodos, tarea donde el algoritmo genético puede desenvolverse con una buena heurística. En este sentido el aprendizaje genético inductivo ha servido para realizar diversas investigaciones. Por ejemplo, se ha utilizado para el reconocimiento de imágenes satelitales, para obtener las mejores topologías para una red neuronal y entrenarlas, para refinar Redes Neuronales Artificiales expertas donde no sólo se computan las respuestas sino también un estimado de lo correcta que es dicha respuesta, entre otras aplicaciones.

En el área de la Robótica también se emplean las técnicas evolutivas. Augustsson (53) presentan una aplicación novedosa de las técnicas evolutivas, al incorporar un sistema de aprendizaje genético a un robot volador. Con este tipo de máquinas, uno de los principales requisitos es un aprendizaje en línea, pues se desean reacciones lo más inmediatas posibles. Lo que la investigación demuestra es la posibilidad de incorporar a un pequeño robot volador un sistema evolutivo lineal capaz de aprender a desempeñarse correctamente a partir de percepciones aerodinámicas. También resalta el trabajo de DiPietro (54) sobre la resolución del problema *Keepaway*, referido a la inducción de la mejor respuesta de atacantes ante defensores en el soccer robótico. Esta investigación comprueba las cualidades de los algoritmos genéticos en labores de planificación.

- **Abductivo**

A diferencia de la deducción, la causa se infiere aquí a partir del efecto:

> Si se sabe que $\forall X\,(p(X) \supset q(X))$, al observar q(1), puede proponerse la hipótesis p(1) para explicar la ocurrencia de q(1).

En este caso, la capacidad de exploración del algoritmo genético se conjuga con la explotación de nuevas hipótesis explicativas de las observaciones. Generalmente, esto implica que el sistema de

aprendizaje genético construye hipótesis donde su validez es comprobada por un evaluador. Es el razonamiento de los diagnósticos, que tratan de proporcionar alguna explicación a los síntomas o acontecimientos del ambiente.

El aprendizaje genético abductivo se aplica en innumerables campos. Por ejemplo se emplea en el ámbito musical en el cual las composiciones se abordan bajo la figura de hipótesis (preferentemente musicales). En algunos estudios se han combinado la deducción para crear nuevas piezas musicales (construcciones) a partir de las existentes. En (55) se presenta un ambiente virtual complejo, habitado por agentes sónicos, con capacidad para oír y generar sonidos. El sistema mantiene retroalimentación con una audiencia humana, y los agentes evolucionan para conservar lo más que puedan el interés de dicha audiencia. El resultado inmediato de la evolución es la hipótesis que cada agente ofrece sobre las preferencias musicales de la audiencia, codificadas bajo la forma de reglas muy complejas.

Basado en aprendizaje abductivo se han creado además sistemas de detección de intrusos de nueva generación, con una detección satisfactoria de ataques de denegación de servicios pues el algoritmo genético opera sobre poblaciones de hipótesis, con una función de evaluación difusa, que considera como mejores individuos a aquellos que proporcionen una buena explicación sobre el ataque que experimenta la red.

Por otra parte, Hamel combina Programación Lógica Inductiva, Aprendizaje de Conceptos, Programación Genética y Lógica de Ecuaciones, para definir una nueva área, la Programación Lógica Ecuacional Inductiva. Estos sistemas proceden a partir de una población de teorías candidatas, cuya adaptación se encuentra supeditada a lo bien que expliquen los ejemplos. Es decir, con el enfoque evolutivo se promueve la subsistencia de las mejores hipótesis.

CAPÍTULO 3: DESCRIPCIÓN DE LA PROPUESTA

Las técnicas de aprendizaje de elementos virtuales han alcanzado gran auge en nuestros días, pues se han convertido en un interesante campo de investigación de la Inteligencia Artificial. Esto se debe fundamentalmente a que estas técnicas son de gran utilidad en un sinnúmero de aplicaciones, tanto empresariales, como industriales, domésticas, en la medicina, etc.

Pero no solo en aplicaciones industriales o equipos se usan estas técnicas, existen otras áreas donde han tenido gran desarrollo. Ejemplo de ello es el área de Realidad Virtual y simuladores. La programación de videojuegos y otras aplicaciones de Realidad Virtual está convirtiéndose hoy en día en un creciente campo donde se investigan y aplican tanto técnicas gráficas como complicados algoritmos que provienen de campos tales como la Inteligencia Artificial. En la programación de juegos se usan técnicas avanzadas, fundamentalmente, en los videojuegos que requieren de cierta inteligencia para poder realizar un determinado comportamiento. En estas áreas la función de las técnicas de aprendizaje consiste en desarrollar mayor nivel de inteligencia y realismo, tanto en los elementos virtuales, como robots y simuladores, pues permiten que los elementos aprendan y puedan desarrollar determinados comportamientos inteligentes.

Pero cada una de estas técnicas es más útil en un determinado campo, en dependencia de sus características, ventajas y desventajas. En este capítulo se realiza una comparación entre estas técnicas, con el fin de seleccionar una para aplicarla en un caso de estudio. Además se hace una propuesta de cómo aplicar estas técnicas en cada uno de los proyectos productivos de la facultad que las necesiten.

3.1 Comparación entre las técnicas de aprendizaje.

3.1.1 Algoritmos Genéticos

Ventajas

Lo más sorprendente y útil de los AG es que no es necesario disponer de un conocimiento profundo del problema a resolver, sino que se parte de estructuras simples que interactúan, dejando que sea la evolución quien haga el trabajo.

Un importante punto es que los algoritmos genéticos son intrínsecamente paralelos. La mayoría de los otros algoritmos son en serie y sólo pueden explorar el espacio de soluciones hacia una solución en una dirección al mismo tiempo. Sin embargo, ya que los AGs tienen descendencia múltiple, pueden producir múltiples soluciones igualmente buenas, al mismo problema, dándoles una mayor probabilidad en cada ejecución de encontrar la solución óptima.

Otra ventaja notable de los algoritmos genéticos es que se desenvuelven bien en problemas con un paisaje adaptativo complejo -aquéllos en los que la función de aptitud es discontinua, ruidosa, cambia con el tiempo, o tiene muchos óptimos locales. Los algoritmos evolutivos, han demostrado su efectividad al escapar de los óptimos locales y descubrir el óptimo global incluso en paisajes adaptativos muy escabrosos y complejos

Finalmente, una de las cualidades de los algoritmos genéticos que, a primera vista, puede parecer un desastre, resulta ser una de sus ventajas: los AGs no saben nada de los problemas que deben resolver. En lugar de utilizar información específica conocida a priori para guiar cada paso, realizan cambios aleatorios en sus soluciones candidatas y luego utilizan la función de aptitud para determinar si esos cambios producen una mejora. Como sus decisiones están basadas en la aleatoriedad, todos los caminos de búsqueda posibles están abiertos teóricamente a un AG; en contraste, cualquier estrategia de resolución de problemas que dependa de un conocimiento previo, debe inevitablemente comenzar descartando muchos caminos a priori, perdiendo así cualquier solución novedosa que pueda existir.

Desventajas

Un problema muy conocido que puede surgir con un AG se conoce como convergencia prematura, Si un individuo que es más apto que la mayoría de sus competidores emerge muy pronto en el curso de la ejecución, se puede reproducir tan abundantemente que merme la diversidad de la población demasiado pronto, provocando que el algoritmo converja hacia el óptimo local que representa ese individuo. Esto es algo que no debe ocurrir, pues existen varios métodos para evitarlo, los cuales implican controlar la fuerza selectiva, para no proporcionar tanta ventaja a los individuos excesivamente aptos

El problema de cómo escribir la función de aptitud debe considerarse cuidadosamente para que se pueda alcanzar una mayor aptitud y verdaderamente signifique una solución mejor para el problema dado. Si se elige mal una función de aptitud o se define de manera inexacta, puede que el algoritmo

genético sea incapaz de encontrar una solución al problema, o puede acabar resolviendo el problema equivocado

Finalmente, no es aconsejable utilizar algoritmos genéticos en problemas resolubles de manera analítica. No es que los algoritmos genéticos no puedan encontrar soluciones buenas para estos problemas; simplemente es que los métodos analíticos tradicionales consumen mucho menos tiempo y potencia computacional que los AGs y, a diferencia de los AGs, a menudo está demostrado matemáticamente que ofrecen la única solución exacta.

3.1.2 Redes Neuronales

Ventajas

Hay muchas buenas razones para el uso de redes neuronales y los avances en este campo incrementarán su popularidad. Son excelentes como clasificadores/reconocedores de patrones, y pueden ser usadas donde las técnicas tradicionales no funcionan. Las redes neuronales pueden manejar excepciones y entradas de datos anormales, muy importante para sistemas que manejan un amplio rango de datos (sistemas de radar y sonar, por ejemplo). Sus ventajas fundamentales radican en:

- Las redes neuronales son aproximadores universales. Esto significa que no existe ninguna función por extraña que sea que no puedan aproximar, dado un número suficiente de capas ocultas y de neuronas
- Son rápidas de ejecutar: una red entrenada no tarda casi en calcular unas salidas dadas unas entradas
- Son capaces de generalizar: una red bien entrenada aprende el problema en general, y es capaz de resolverlo aunque le demos información dañada o con ruido (hasta cierto límite).
- Tolerancia a fallos: la destrucción parcial de una red conduce a una degradación de su estructura; sin embargo, algunas capacidades de la red se pueden retener, incluso sufriendo un gran daño.

Desventajas

Sin embargo, los sistemas neuronales presentan ciertos inconvenientes. Uno importante es que habitualmente realizan un complejo procesamiento que supone millones de operaciones, por lo que no es posible seguir paso a paso el razonamiento que les ha llevado a extraer sus conclusiones.

Otros problemas con las redes neuronales son la falta de reglas definitorias que ayuden a construir una red para un problema dado, hay muchos factores a tomar en cuenta: el algoritmo de aprendizaje, la arquitectura, el número de neuronas por capa, el número de capas, la representación de los datos y mucho más y otro de los problemas es la llamada caja negra el problema es que cuando modelamos estadísticamente somos capaces de ver que variables forman parte del modelo o cuales de las que finalmente se utilizaron para modelar fueron seleccionadas por los algoritmos para predecir o clasificar, podemos ver sus pesos y la ecuación final, cosa que no es posible en las redes neuronales.

Cuando se usa una red neuronal, también existe una disyuntiva o discusión respecto a que es mejor si un modelo estadístico de probabilidad o una red neuronal, esto depende del problema, sin embargo, la complejidad de modelamiento en ambos casos requiere de conocimientos sólidos que permitan llegar a resultados satisfactorios y válidos, lo cual no es necesario si se usan otras técnicas.

Además con respecto al entrenamiento, son lentas de entrenar: cuanto más grande es una red, más le cuesta entrenarse y aprender y son difíciles de entrenar: cuanto más complejo el problema, más difícil es entrenar una red.

Necesitan bastante prueba y error: aún habiendo programado bien el algoritmo y elegido bien los patrones de entrenamiento, la red puede no funcionar si los parámetros de inicialización no son los correctos. Hay formas que ayudan a aproximar esos valores, pero al final termina siendo un proceso de prueba y error.

3.1.3 Redes Bayesianas

Ventajas

Las redes bayesianas poseen una sólida teoría probabilista que les permite dar una interpretación objetiva, permite el manejo de incertidumbre y permite realizar un proceso de inferencia, por lo que son capaces de trabajar con datos incompletos, y dadas las dependencias de los temas es sencillo construir el modelo.

También se puede decir que el Aprendizaje de Redes Bayesianas resulta de gran importancia porque es práctico y además provee un enfoque de comprensión y diseño de otros algoritmos. Este tipo de aprendizaje ofrece la posibilidad de inducir la estructura gráfica de la red a partir de los datos observados y de definir las relaciones entre los nodos basándose también en dichos casos, o sea que tan solo con los datos observados se puede definir una red, por lo que no se necesitan datos

adicionales. Por otro lado, las redes bayesianas pueden desarrollar el aprendizaje a partir de bases de datos incompletas, realizan estimaciones, para inferir de alguna manera, los datos ausentes, para completar la base de datos.

Otras ventajas de este aprendizaje son:

- Cada nuevo ejemplo puede aumentar o disminuir la estimación de una hipótesis (flexibilidad - incrementalidad).
- Conocimiento a priori se puede combinar con datos para determinar la probabilidad de las hipótesis.
- Da resultados con probabilidades asociadas.
- Puede clasificar combinando las predicciones de varias hipótesis.
- Sirve de estándar de comparación de otros algoritmos.

Desventajas

Sin embargo, algunos estudios sobre el aprendizaje de redes bayesianas indican que el requisito principal para poder realizar la tarea de aprendizaje a partir de datos es disponer de bases de datos muy extensas en las que esté especificado el valor de cada variable en cada uno de los casos, lo cual resulta muy costoso e inseguro.

Las redes bayesianas, en general, tienen dos inconvenientes principales, necesitan gran cantidad de probabilidades numéricas y la dificultad de los cálculos debido a la presencia de bucles en estas redes bayesianas. Es computacionalmente caro (depende linealmente del número de hipótesis). Además, no es sencillo de implementar el proceso de inferencia, podría requerir gran espacio de memoria a medida que se va complicando la red. Y por último es difícil determinar los tiempos estándar de revisión de los temas. Es necesario hacer estudios estadísticos para determinarlos.

3.1.4 Sistemas de Lógica Difusa

Ventajas

La lógica difusa es una solución más abierta que la lógica clásica, permite dar mejores soluciones a problemas que tengas variables imprecisas, o que puedan tomar valores variados en el tiempo. Estos sistemas son generalmente robustos y tolerantes a imprecisiones y ruidos en los datos de entrada. Algunos sistemas de Lógica Difusa tienen la propiedad de *auto organización*, lo que hace que sea sensato proponer los Sistemas de Lógica Difusa como solución a problemas complejos, en los que las

representaciones matemáticas exactas no se conocen, o son lo suficientemente complicadas como para que no sea práctico emplearlas. En esto radica su principal ventaja, en la capacidad que tienen para resolver problemas matemáticamente muy complejos.

Principalmente, miraremos la aptitud del control difuso en términos generales. El empleo de los sistemas de control difuso es recomendable:

- Para procesos muy complejos, cuando no hay un modelo matemático simple.
- Para procesos altamente no lineales.
- Si el procesamiento del (lingüísticamente formulado) conocimiento experto puede ser desempeñado.

Desventajas

Pero precisamente en esto radica también su desventaja fundamental, ya que al resolver problemas difíciles y sobre todo que tengan variables imprecisas, si son usados en otro tipo de problema que no es adecuado, esto conllevaría una solución muy complicada, y trabajosa, o sea, que un problema que tendría una solución más sencilla, se convertiría en un problema matemático difícil de resolver e implementar. Por eso lo más importante a la hora de implementar un problema usando un sistema de lógica difusa es saber si seleccionar esta técnica o no para resolverlo, o sea, escoger cuidadosamente las aplicaciones que se van a resolver usando la misma.

El empleo del control difuso no es una buena idea si:

- El control convencional teóricamente rinde un resultado satisfactorio.
- Existe un modelo matemático fácilmente soluble y adecuado.
- El problema no es soluble.

3.1.5 Conclusiones de la comparación

Al comparar estas técnicas, se puede llegar a la conclusión de que resuelven complejos problemas de la vida cotidiana, sin embargo es muy importante saber en qué tipo de aplicaciones se deben usar, pues cada una de ellas es más adecuado emplearlas en determinados tipos de aplicaciones en dependencia de sus ventajas y desventajas, además de los tipos de problemas que resuelven. En el área específica de Realidad Virtual, son usadas, en dependencia del tipo de aplicación, por ejemplo en el área de la robótica son muy empleadas, las redes neuronales y los sistemas de lógica difusa; en los videojuegos, también se emplean estas dos técnicas, al igual que las redes bayesianas y los algoritmos genéticos. En los simuladores son muy empleados los sistemas expertos. Lo importante es

saber seleccionar cuál es la técnica más adecuada, en dependencia de las características del problema o aplicación, de los datos que se conozca, de las variables que se tengan, etc.

De acuerdo a las ventajas y desventajas analizadas en cada una de ellas, se ha podido determinar que una de las técnicas que presenta grandes ventajas a la hora de desarrollar un videojuego, o una aplicación de realidad virtual, son los algoritmos genéticos.

Los algoritmos genéticos, son una técnica muy potente que además tiene un amplio campo de aplicación en las soluciones de nuestros días, desde implementaciones sencillas en juegos, pasando por otras más complejas como aplicaciones médicas, electrónicas, planeación de obras civiles, entre otras. En los campos de realidad virtual y videojuegos, son usadas frecuentemente.

Mientras el poder de la evolución gana reconocimiento cada vez más generalizado, los algoritmos genéticos se utilizan para abordar una amplia variedad de problemas en un conjunto de campos sumamente diverso, demostrando claramente su capacidad y su potencial.

La gran sencillez de los AG los hace muy atractivos para los programadores, además del gran número de ventajas que poseen con respeto a la aplicación de otras técnicas. Es por ello que se decidió aplicarlos al caso de estudio.

3.2 Propuesta de la aplicación de las técnicas en los proyectos de la Facultad

3.2.1 Proyecto Juegos de Consola

En el proyecto Juegos de Consola se está desarrollando una aplicación de realidad virtual que consiste en un juego de carreras de autos para varios jugadores donde uno de sus requerimientos es que permita al usuario realizar carreras contra varios autos controlados por la PC, estos tendrían que tener un cierto nivel de inteligencia para poder darle al jugador la mayor sensación de realismo posible.

Para lograr que los automóviles controlados por la computadora se comporten de manera natural en el entorno simulado se pueden utilizar algunas de las técnicas de aprendizaje descritas en este trabajo.

Una de las técnicas que se puede aplicar para lograr que los automóviles controlados por la PC tengan un comportamiento inteligente son las Redes Neuronales. De forma simultánea a esta investigación se desarrolló un módulo de redes neuronales para definir el comportamiento de los carros autónomos en

el videojuego “Rápido y Curioso” desarrollado por el proyecto. Se empleó un Perceptrón Multicapa con tres capas de neuronas, la capa de entrada que recibe los parámetros de retroalimentación del entorno virtual (distancia al centro de la pista, ángulo entre la dirección del carro y la pista, distancia del carro a la próxima curva, ángulo de la próxima curva); una oculta, en la que tiene lugar la mayor parte del procesamiento de la red; y otra de salida (velocidad deseada, ángulo de giro del timón), que decide cómo controlar el auto.

También se pudieran aplicar los AG para lograr un comportamiento inteligente, por ejemplo en una situación donde el auto de carreras controlado por la PC esté en la misma dirección que otro automóvil pero se encuentren en sentidos opuestos de forma tal que su aproximación provocaría una colisión, una posible respuesta que pudiera dar el auto controlado por la PC sería cambiar de senda o frenar. Así se podrían ir codificando las respuestas a los distintos escenarios que pueda tener desarrollados este juego permitiendo que estos automóviles se comporten siempre de una manera lógica. La función de aptitud pudiera estar determinada por el tiempo de carrera, la posición final de auto, la cantidad de automóviles que eliminó de la competencia o las veces que tuvo que incorporarse nuevamente a la carrera, entre otras.

3.2.2 Proyecto Desarrollo de Elementos Virtuales Inteligentes.

Este proyecto se dedica a incorporar elementos inteligentes y otras estrategias de la IA en todos los proyectos de Realidad Virtual de la Facultad 5, respondiendo a las necesidades de los mismos.

También trabaja en la realización de una biblioteca de inteligencia artificial la cual incluirá técnicas básicas de IA como las que están reflejadas en este trabajo de manera genérica.

El proyecto no cuenta con la bibliografía adecuada para poder cumplir con las demandas que tiene sobre las técnicas aquí expuestas, además necesita de un estudio de las mismas para poder seleccionar la forma óptima y genérica de implementarlas en dicha biblioteca.

Es por eso que en este proyecto las técnicas de aprendizaje tratadas en este trabajo son ampliamente aplicables y su utilización estará dada en dependencia de los problemas concretos a resolver por el proyecto.

3.2.3 Proyecto Herramienta de Desarrollo para Sistemas de Realidad Virtual (HDRSV)

El proyecto HDSRV se dedica a la elaboración de una herramienta base para todos los proyectos de Realidad Virtual de la facultad 5, la cual permita generar sistemas de Realidad Virtual.

Un tipo de productos que genera dicho proyecto es la realización de demos para probar las potencialidades de la herramienta, entre los cuales están los demos con inteligencia artificial que incluyen la aplicación de una o más de las técnicas expuestas en este trabajo.

Uno de los demos desarrollados por este proyecto es para un futuro simulador de tiro, donde uno de sus requerimientos consiste en crearle a un jugador un escenario donde varios soldados controlados por la PC traten de llegar a él para eliminarlo. Se puede ir aumentando la capacidad de reacción de los elementos que van apareciendo al usuario, que vayan surgiendo nuevos soldados con características más adecuadas para evitar el disparo del jugador. Por ejemplo, uno o varios soldados controlados por la PC para tratar de eliminar al jugador pudieran hacerlo moviéndose entre obstáculos que habrían entre ellos, una situación pudiera ser, que el jugador tenga un blanco en la mira, comience a disparar y el soldado controlado por la PC se esconda detrás del obstáculo más próximo a él, para evitar los disparos. En este caso la función de aptitud pudiera determinarse por factores como: distancia que logró aproximarse al jugador, cantidad de disparos acertados, etc.

Otro ejemplo concreto es el caso de estudio que se describe a continuación.

3.3 Descripción del caso de Estudio

Para demostrar que es posible aplicar la técnica de Algoritmo Genético en el desarrollo de aplicaciones de Realidad Virtual que demanden un alto nivel de inteligencia se tomará como ejemplo un pequeño Demo de Tiro donde existen un grupo de soldados controlados por la computadora que deben responder ante diversas situaciones provocadas por el usuario. Para lograr esto, se puebla el mundo del juego con diversas soluciones posibles a tener en cuenta de las cuales el AG debe ser capaz de determinar las mejores a ejecutar por los soldados ante las acciones del adversario, claro, las soluciones no serán siempre iguales para cada soldado con el objetivo de lograr mayor diversidad, elevando el nivel de inteligencia en el juego. O sea, se pretende usar un AG para crear jugadores controlados por la PC que respondan con los comportamientos adecuados para combatir las acciones de un jugador.

A continuación se describe detalladamente el ejemplo. En este juego, el jugador puede realizar una serie de acciones distintas para atacar a los soldados controlados por la computadora, y estos tendrán que responder a cada acción con el comportamiento adecuado.

Los posibles escenarios que tiene el juego se muestran a continuación:

- Atacar con disparo cerca.
- Explotar el tanque cerca.
- Ataque Aéreo.
- Activar la alarma de combate.

De esta manera es como quedarían descritas las situaciones a las cuales la población de soldados tendría que responder con comportamientos diferentes. La respuesta a cada una de estas situaciones será almacenada en un cromosoma.

Por supuesto, un juego verdadero tendría una lista mucho más extensa y detallada de los escenarios posibles debido al grado de complejidad y la diversidad que estos presentan.

Seguidamente se listan los posibles comportamientos que podrían tener los soldados controlados por la computadora ante los escenarios ya descritos.

- Correr.
- Saltar.
- Agacharse.
- Combatir.
- Caminar.

3.3.1 Definición de la estructura “GeneticAlgorithm”

Como propuesta de implementación de la estructura “GeneticAlgorithm”, será creada una clase base llamada “Creature”, utilizando como lenguaje de programación C++. Esta clase tendrá los posibles comportamientos en los distintos escenarios del juego, o sea, tendría un conjunto de cromosomas. Cada elemento de este conjunto representará un comportamiento para un soldado controlado por la computadora en un escenario específico, el tamaño de este conjunto será 4 ya que esta es la cantidad de escenarios posibles. Esta clase tendrá como atributos significativos a *id*, *suma*, *aptitud* y los principales métodos serán su constructor, los métodos para obtener los valores de los atributos y para actualizarlos, el método para determinar la aptitud “Aptitud” y el método “Crear_Individuo”.

Si se tiene en cuenta que los mejores individuos serán aquellos que reaccionen con el mejor comportamiento ante cada situación, es necesario entonces darle una puntuación específica a cada respuesta por escenario dependiendo de lo efectiva que sea dicha respuesta ante la acción que se realizó, estos valores serán guardados en variables para utilizarlas luego para obtener la aptitud del individuo.

El método “Aptitud” tendrá como funcionalidad calcular el valor de la aptitud, el resultado estará dado por la sumatoria total de los puntos acumulados por el individuo al responder ante cada acción específica, este valor se almacenará en el atributo *aptitud*. Así quedaría definida la “función de aptitud”.

Como el objetivo es que la población creada sea bien diversa no se puede inicializar cada individuo con un sistema de constantes predeterminadas pues no sería efectivo para encontrar la mejor solución, por eso es necesario asignar los comportamientos de manera aleatoria siendo esta la principal función del método “Crear_Individuo”. En este método también se le da la puntuación a cada comportamiento según el escenario.

Para describir lo que haría este método tomaremos como ejemplo el caso “ATACADO_POR_DISPARO_CERCA”, a este cromosoma se le asigna uno de manera aleatoria: “COMBATIR”, “CORRER”, “AGACHARSE”, “SALTAR” o “CAMINAR” con una puntuación definida en dependencia de su efectividad en este escenario, de esta forma se podría determinar cuál de ellas es la más ventajosa cuando el soldado controlado por la computadora es “ATACADO_POR_DISPARO_CERCA", en este caso sería “COMBATIR”. Para los demás cromosomas se realizaría un proceso semejante.

Ahora se creará la población, comenzando por la propuesta de la implementación de la estructura “GeneticAlgorithm” que tendrá como atributo principal una lista de tipo “Creature”, esta lista constituirá la población sobre la cual se realizarán todos los procesos del algoritmo genético, también tendrá otros atributos como son “cant”, “cantSeleccionados” y “cantNuevos”. Como métodos principales estarán: “Selección”, “Cruzamiento”, “Mutación” y ”NuevaPoblación” además de su constructor.

Selección Natural

La selección de los miembros de la población que pasarán a realizar el cruzamiento constituye una de las principales y más importantes tareas del Algoritmo Genético, el método “Selección” es el responsable de cumplir este objetivo. Este método se encarga de escoger a los dos individuos que

tengan el mayor valor de aptitud, de esta forma se castigarían las malas soluciones y se le daría la oportunidad a las mejores de ser seleccionadas para realizarle el proceso de Cruzamiento y Mutación. Cuando se termine el proceso se retornan todos los individuos que fueron seleccionados.

Cruzamiento y Mutación

Cuando ya se han seleccionados los individuos que se cruzarán, el siguiente paso es realizar el cruzamiento para lograr una descendencia. El llamado “Cruzamiento” es el método que se encarga de esta tarea. Para lograrlo, en el método se aplicará la técnica de “cruce por un punto”, en el cual un padre aporta sus genes que están antes del punto de cruce y el otro padre sus genes que están después de dicho punto.

En el método “Cruzamiento” se pasa como parámetro una lista que debe ser la que contenga a los individuos que se seleccionaron para realizar el cruzamiento, luego para cada iteración se genera un punto de cruce con un valor aleatorio entre uno y dos porque cada individuo solo tiene 4 cromosomas y no se realizaría el cruzamiento de escogerse como punto de cruce a cero o tres. Después de realizado el proceso con los dos padres anteriormente seleccionados se obtiene como resultado una lista con dos nuevos individuos que será retornada por dicho método.

Para obtener una generación de individuos la estructura “GeneticAlgorithm” puede apoyarse además en el método “Mutación” que será el encargado de aplicarle la operación de mutación a un individuo de la nueva generación, es decir después que se crean nuevos individuos es posible que estos sean sometidos a la mutación para garantizar la inserción de nuevo material cromosómico.

Este proceso se simula de manera parecida a la forma en que ocurre en la vida real, con un bajo grado de probabilidad de ocurrencia, es por eso que el método “Mutación” reasignará los cromosomas aleatoriamente donde cada rasgo tiene una probabilidad del 5% de mutar.

El método “Mutación” recibe una lista, que debe ser la que contenga a los nuevos individuos y se selecciona de manera aleatoria a cualquiera de ellos para realizarle el proceso, también se determina aleatoriamente el gen que se va a mutar y se retorna la lista. Es necesario tener en cuenta que durante este proceso es probable que ningún individuo sufra alteración genética.

Nueva Población

Después que es generada una nueva descendencia, esta debe ser insertada en la población actual, en la estructura "GeneticAlgorithm" el método que será el encargado de realizar dicho proceso es "NuevaPoblación".

Para insertar la nueva descendencia en la población se tiene en cuenta que la misma es menor que la población actual, por lo que sólo serán eliminados los dos individuos que contengan los menores valores de aptitud en la población y se insertarán por cada uno de ellos a los individuos de la nueva descendencia.

Para determinar cuáles son los individuos menos aptos se organiza la población de manera descendente según su valor de "aptitud" utilizando método de ordenamiento Burbuja, que tiene como ventaja su sencillez, fácil implementación y eficacia, pero consume bastante tiempo, ya que siempre hace la misma cantidad de comparaciones incluso cuando la lista ya está ordenada.

De esta manera quedaría conformada la estructura "GeneticAlgorithm", capaz de determinar el mejor comportamiento que pudiera tener un soldado controlado por la computadora frente a diferentes escenarios con el objetivo de tener un buen desempeño ante la acción de cualquier jugador para tratar de demostrar que la aplicación de los Algoritmos Genéticos en soluciones a problemas de Inteligencia Artificial es una opción confiable y que pueden alcanzar resultados satisfactorios en este tipo de problemas.

3.3.2 Resultados.

Para tener mayor comprensión del funcionamiento del algoritmo descrito anteriormente a continuación se ejemplificará con un juego de datos. Es importante recordar que la población inicial se genera de forma aleatoria y luego, a medida que ocurre el proceso evolutivo del algoritmo, los individuos irán evolucionando hasta alcanzar un comportamiento bastante cercano al óptimo.

Cada escenario ocupa una posición en la cadena de cromosomas, por lo que quedarían dispuestos de la siguiente forma:

0-----Atacar con disparo cerca.

1-----Explotar el tanque cerca.

2-----Ataque Aéreo.

3-----Activar la alarma de combate.

Ahora, ante cada escenario las respuestas poseen un valor diferente de importancia en dependencia de lo óptima que esta sea.

Respuesta/Valor por Escenario	0	1	2	3
0-----Correr	4	5	3	3
1-----Agacharse	3	4	5	1
2-----Saltar	1	1	1	5
3-----Combatir	5	3	4	4
4-----Caminar	2	2	2	2

Tabla 5. Valor de las respuestas ante cada escenario.

Teniendo en cuenta los escenarios descritos y los valores de cada respuesta ante cada uno de ellos, a continuación se muestran las iteraciones que tuvo este algoritmo para una población de cuatro individuos.

Población Inicial

De esta población inicial, al calcularle la aptitud después de responder a todos los escenarios son seleccionados los dos individuos que mejor respondieron, o sea, los dos individuos con mayor valor de aptitud.

Id del Individuo	Cromosomas	Valor de cada respuesta ante cada escenario				Aptitud	Seleccionados para el cruce
0	2234	1	1	4	2	8	-
1	0111	4	4	5	1	14	X
2	0414	4	2	5	2	13	X
3	1140	3	4	2	3	12	-
Totales		**12**	**11**	**16**	**8**	**47**	

Tabla 6. Población inicial.

Luego de realizarles el proceso de cruce por un punto a los individuos 1 y 2, se obtienen dos nuevos individuos (4 y 5) con un valor de aptitud superior, por lo tanto al escoger a la población que formaría la

primera generación quedaría conformada por los individuos 1, 2,4 y 5 puesto que ellos son los que presentan mayor aptitud.

Id del Individuo	Cromosomas	Valor de cada respuesta ante cada escenario				Aptitud	Seleccionados para el cruce
1	0111	4	4	5	1	14	X
2	0414	4	2	5	2	13	-
4	0114	4	4	5	2	15	X
5	0411	4	2	5	1	12	-
Totales		16	12	20	6	**54**	

Tabla 7. Primera generación.

Luego de realizarles el proceso de cruce por un punto a los individuos 1 y 4, se obtienen dos nuevos individuos (6 y 7) con igual valor de aptitud que sus padres, por lo tanto, al escoger a la población que formaría la siguiente generación quedaría conformada por los individuos 1, 4,6 y 7 puesto que ellos son los que presentan mayor aptitud.

Id del Individuo	Cromosomas	Valor de cada respuesta ante cada escenario				Aptitud	Seleccionados para el cruce
1	0111	4	4	5	1	14	-
4	0114	4	4	5	2	15	X
6	0114	4	4	5	2	15	X
7	0111	4	4	5	1	14	-
Totales		16	16	20	6	**58**	

Tabla 8. Segunda generación.

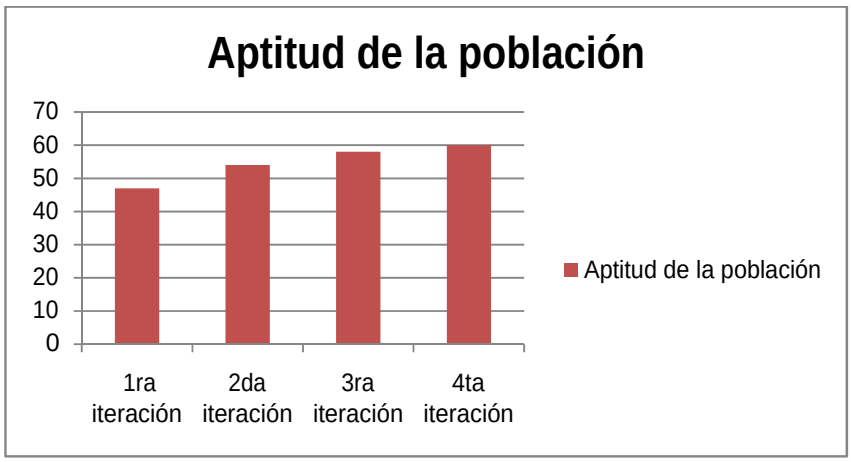
En este caso, luego del proceso de cruzamiento de los individuos 4 y 6, se obtienen dos nuevos individuos (8 y 9) con exactamente los mismos cromosomas que sus padres, y al seleccionar la población quedaría compuesta por los individuos 4, 6, 8 y 9 que presentan todos los mismos

cromosomas, o sea, el algoritmo ha llegado a su punto de convergencia ya que la población no evolucionará más.

Id del Individuo	Cromosomas	Valor de cada respuesta ante cada escenario				Aptitud	Seleccionados para el cruce
4	0114	4	4	5	2	15	-
6	0114	4	4	5	2	15	-
8	0114	4	4	5	2	15	-
9	0114	4	4	5	2	15	-
Totales		16	16	20	8	**60**	

Tabla 9. Población Final.

Independientemente que para este caso el algoritmo solamente llegó a tres iteraciones debido a los pocos escenarios y respuestas implementados y la población tan pequeña que evita mayor diversidad a la hora de escoger los cromosomas cuando son creados los individuos se puede ver cómo el valor de la aptitud de la población fue aumentando a medida que se obtenía una nueva generación de individuos por lo que podemos decir que hubo un proceso de aprendizaje de los mejores comportamientos por parte de las generaciones siguientes demostrando que es factible aplicar los AG a los elementos virtuales en aplicaciones que requieran cierto grado de inteligencia. La siguiente gráfica muestra los resultados obtenidos.



Gráfica 1. Aptitud general de la población.

Para lograr los resultados alcanzados se tuvo en cuenta que el problema tuviera un espacio de búsqueda delimitado, o sea, en el ejemplo el espacio de búsqueda consistió en los escenarios que se definieron y en los posibles comportamientos que podía tener el soldado. Para determinar cuáles eran las mejores respuestas, se definió una función de aptitud bastante sencilla, solo estaba determinada por una suma. También las posibles soluciones fueron codificadas con constantes, esto hizo más fácil la implementación de la estructura y su comprensión.

En conclusión al desarrollo del ejemplo se propone que siempre se tengan en cuenta elementos tan importantes como los descritos anteriormente en la aplicación de un Algoritmo Genético en cualquier problema a solucionar en la Inteligencia Artificial de un entorno virtual, para lograr un funcionamiento correcto del mismo y la solución satisfactoria del problema planteado.

CONCLUSIONES

Con el desarrollo del presente trabajo se han logrado cumplir los objetivos al proponer el uso de técnicas de aprendizaje de elementos virtuales para mejorar el nivel de realismo e inteligencia en las aplicaciones de Realidad Virtual.

Se realizó una búsqueda de las necesidades de los proyectos de Realidad Virtual de la facultad con el objetivo de analizar qué tipo de técnicas aplicar en aquellos que requieren un comportamiento inteligente.

Además a través de una comparación entre estas técnicas, se argumentaron sus ventajas y desventajas y se concluyó usando un Algoritmo Genético en una aplicación demostrativa, donde se prueba cómo los elementos virtuales son capaces de aprender mediante esta técnica.

Como resultado de la investigación realizada se proponen las técnicas de aprendizaje de Inteligencia Artificial: Redes Neuronales, Redes Bayesianas, Lógica Difusa y Algoritmos Genéticos, para ser utilizadas en los proyectos de Realidad Virtual, de las cuales se logró explicar cómo se desarrolla este proceso de aprendizaje, adquiriendo una amplia visión y comprensión del funcionamiento de las mismas.

RECOMENDACIONES

Se recomienda:

- Utilizar el presente trabajo como punto de partida para profundizar sobre estos temas en futuras investigaciones de Inteligencia Artificial.

- Aplicar las técnicas que se proponen en los distintos proyectos de Realidad Virtual para lograr mejores resultados en cuanto al nivel de realismo e inteligencia en los mismos.

- Que se tenga en cuenta esta investigación para que en los nuevos proyectos de Realidad Virtual que surjan, se apliquen estas técnicas, y se seleccione la más adecuada en dependencia de las características del proyecto.

REFERENCIAS BIBLIOGRÁFICAS

1. Free Download Manager. *Fishing Expert (Experto en Pesca) (Fishing Expert) 4.0a.* [En línea] 1 de Enero de 2001. http://www.freedownloadmanager.org/es/downloads/Experto_de_Pesca_7326_p/.

2. **Meda B, Josue y Jiménez, Gloria L.** Ingeniería del Conocimiento. *Ejemplos de Sistemas Expertos.* [En línea] 2004.
http://www.te.ipn.mx/catalogo/site/cursos_linea/conocimiento/Ingenieria_conocimiento/creditos.htm.

3. **Gómez M, Marco A y Díaz A, Belén.** *Evaluación de estrategias de razonamiento para sistemas basados en reglas.* Facultad de Informática,Universidad Complutense de Madrid : s.n., 2006.

4. **Russell, S. J. y Norvig, P.** *Artificial Intelligence:A Modern Approach.* s.l. : 2nd ed,Prentice Hall, 2002.

5. **Riley, D.** *CLIPS, a tool for building expert systems.*

6. **Friedman-Hill, J.** *Jess, the Expert System Shell for the Java Platform.* 2003.

7. **Riesbeck, Christopher K, y otros.** AICOM. *Case-Based REasoning: Foundational Issues, Methodological Variations, and System Approaches.* s.l. : Inside Case-Based Reasoning Lawrence Erlbaum Associates, Publishers, 1989. Vol. 7, 1, March 1994.

8. **Posada Toledo, Nidia.** *CBR y toma de decisiones en el contexto de un GIS: caso del volcán Popocatépetl.* Puebla, México : Universidad de las Américas ,Escuela de Ingeniería,Departamento de Ingeniería en Sistemas Computacionales, 2001.

9. **Velasco Villanueva, David A.** Inteligencia Artificial II. *Redes Bayesianas.* 2007.

10. **Wang, Xiao-Feng, y otros.** Machine Learning and Cybernetics. *Using Bayesian networks to model the belief in the opponent in static game with incomplete information.* Proceedings of 2004 International Conference on Volume 1, Issue, 26-29 Aug. 2004 : s.n., 2004.

11. **Smith, Warren D y Baum, Eric B.** *A Bayesian approach to relevance in game playing.* Artificial Intelligence 97,195-242 : s.n., 1997.

12. **Kets, Willemien.** *Convergence of Beliefs in Bayesian Network Games.* 2007.

13. **Bourg, David M y Seeman, Glenn.** *AI for Game Developers.* s.l. : O´Reilly, 2004.

14. **Albrecht, D.W, Zukerman, I y Nicholson, E.** *Bayesian Models for Keyhole Plan Recognition in an Adventure Game.* 1998. Vol. 8, 1-2.

15. **Watson y Pollack.** *How symbiosis can guide evolution.* s.l. : Springer-Verlag, 1999. Proceedings of the Fifth European Conference on Artificial Life, pp. 29-38.

16. **Watson y Pollack.** *Symbiotic Combination as an Alternative to Sexual Recombination in Genetic Algorithms.* s.l. : Springer-Verlag, 2000. Proceedings of Parallel Problem Solving from Nature, pp. 425-434.

17. **Srinivas, M y Patnaik.** *Genetic Algorithms: A Survey.* Junio 1998,2000. IEEE Computer, pp. 17-26.

18. **Chellapilla, K y Fogel, D.B.** Evolutionary Computation. 2000. Proceedings of the 2000 Congress on Volume 2, Page(s):857 - 863.

19. **Kendall, G y Spoerer, K.** *Scripting the Game of Lemmings with a Genetic Algorithm.* 2004.

20. **Wyman, Chris.** *Using Genetic Algorithms to Learn Weights for Simple King-Pawn Chess Endgames.* Machine Learning, CS 6350, Final Project.

21. **Liu, Zhangbo.** *A Guided Genetic Algorithm for the Planning in Lunar Lander Games.* 2006. Department of Computer Science. University of British Columbia.

22. **Catalina Gallego, Alfredo.** *Introducción a las redes neuronales artificiales.* [En línea] http://www.gui.uva.es/login/login/13/redesn.html.

23. **Konolige, K y Myers, K.** The Saphira architecture for autonomous mobile robots. *In Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems.* 1998. Chapter 9, The MIT Press, 211-242.

24. **Saffiotti, A, Konolige, K y Ruspini, E.** *A multivalued logic approach to integrating planning and control.* 1995. Artificial Intelligence, 76, 481-526.

25. **Michaud, F.** *Selecting behaviors using fuzzy logic.* Barcelona, España : s.n., 1997. In Proc. IEEE Int. Conf. on Fuzzy Systems,585-592.

26. **Pirjanian, P y Mataric, M.** *A decision-theoretic approach to fuzzy behavior coordination.* 1999. IEEE International Symposium on Computational Intelligence in Robotics and Automation.

27. **Goodridge, S. G, Kay, M. G y Luo, R. C.** *Multi-layered fuzzy behavior fusion for reactive control of an autonomous mobile robot.* Barcelona, España : s.n., 1997. In Proc. IEEE Int. Conf. on Fuzzy Systems, 579-584.

28. **Saffiotti, A y Wesley, L.P.** *Perception-based self-localization using fuzzy locations.* Berlín : Springer-Verlag, 1996. Reasoning with Uncertainty in Robotics, 368-385.

29. **Graves, S, Mollenhauer, J y Skubic, M.** *The FuzzBug Mobile Robot.* Mayo 1993. Technical Report 93-041, Department of Computer Science, Texas A&M University.

30. **Howard, Ayanna y Seraji, Homayoun.** *A Real-Time Autonomous Rover Navigation System.* 2000. In Proc.of the World Automation Congress.

31. **Howard, Ayanna, y otros.** *Enhancing Fuzzy Robot Navigation Systems by Mimicking Human Visual Perception of Natural Terrain Traversability.* Vancouver, Canad : s.n., Julio 2001. Joint 9th IFSA World Congress/20th NAFIPS International Conferencea.

32. **Howard, Ayanna, Seraji, Homayoun y Tunstel, Edward.** *A Rule-Based Fuzzy Traversability Index for Mobile Robot Navigation.* Seoul (Korea) : s.n., Mayo 2001. IEEE International Conference on Robotics and Automation.

33. **Widrow, B y Hoff, M. E.** *Adaptive switching circuits.* 1960. In IRE WESCON, pages 96-104, New York. Convention Record.

34. **Narendra, K. S y Thathachar, M. A.** *Learning Automata: A Survey.* 1974. Vol. 14. Transactions on Systems, Man, and Cibernetic. p 323-334.

35. **Barto, A. G y Anandan, P.** *Pattern recognizing stochastic learning automata.* 1985. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 15, pp. 360--375.

36. **Barto, A.G, Sutton, R.S y Anderson, C.** *Neuron-like adaptive elements that can solve difficult learning control problems.* 1983. IEEE Transactions on Systems, Man, and Cybernetics, SMC-13: 834-846.

37. **Sejnowski, TJ.** *Storing covariance with nonlinearly interacting neurons.* 1977. J Math Biol 4:303–321.

38. **Sutton, R.S y Barto, A.G.** *Toward a modern theory of adaptive networks: Expectation and prediction.* 1981. Psychological Review 88:135-140.

39. **Klopf, A. Harry.** *A drive-reinforcement model of single neuron function: An alternative to hebbian neuronal model.* 1986. In A.I.P. Conference Proceedings 151, Neural Networks for Computing. American Institute of Physics.

40. **Pearl, J.** *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Palo Alto : s.n., 1988. Morgan Kaufmann Publishers.

41. **Felgaer, P.E.** *Optimización de redes Bayesianas basado en Técnicas de Aprendizaje por Inducción.* Febrero 2005. Laboratorio de Sistemas Inteligentes. Facultad de Ingeniería. Universidad de Buenos Aires.

42. **Ramoni, Marco y Sebastiani, Paola.** *Learning Bayesian Networks from Incomplete Databases.* Febrero 1997.

43. **Friedman, Nir, Goldszmidt, Moises y Geiger, Dan.** *Bayesian Network Classifiers.* [En línea] http: / /www.cs.huji.ac.il /~ni r /Paper s /FrGG1.pdf.

44. **Buckles, B. P y Petry, F. E.** *Genetic Algorithms.* 1992. IEEE Computer Society Press.

45. **Marczyk, Adam.** *Algoritmos genéticos y computación evolutiva.* 2004.

46. **Mitchell, T.** *Machine Learning.* s.l. : McGraw-Hill, 1997. pp. 1-3.

47. **Kim, S y Zhang, B.** *Web document retrieval by genetic learning of importance factors for html tags.* Australia : Proc. Int. Workshop Text Web Mining, 2000. pp. 13-23.

48. **Muggleton, S y De Raedt, L.** Inductive Logic Programming: Theory and Methods. *Journal of Logic Programming.* 1994.

49. **Chakraborti, C.** Human Deductive Reasoning, Genetic Algorithms: A Proposed Model. *Journal of Indian Council of Philosophical Research .* 2001, Vol. 18, pp. 61-90.

50. **Hancock, P y Frowd, C.** *Evolutionary generation of faces.* 2002. Creative Evolutionary Systems, Morgan Kaufmann.

51. **Bianchi, D y Delmonte, R.** *Tecniche di apprendimento applicate al problema del tagging: una prima valutazione per l'Italiano.* 2002. Convención IA.

52. **Echizenya, H.** *Application of Genetic Algorithms for Example-Based Machine Translation Method Using Inductive Learning and Its Effectiveness.* 2001. IPSJ JOURNAL Vol. 37.

53. **Augustsson, P, Wolff, K y Nordin, P.** *Creation of a Learning, Flying Robot by means of Evolution.* 2002. Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1279-1285.

54. **DiPietro, A, While, L y Barone, L.** *Learning In RoboCup Keepaway Using Evolutionary Algorithms.* 2002. Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1065-1072.

55. **McCormack, J.** *Evolving Sonic Ecosystems.* 2002. Kybernetes Vol. 32 (1/2).

BIBLIOGRAFÍA

Brito Boadas, José Alejandro. Agosto 2004. *Aprendizaje Deductivo, Inductivo, y Abductivo con Algoritmos Genéticos.* Agosto 2004. Revista Ingeniería Informática, Edición Número 10.

Buckland, Mat. 2005. *Programming Game AI by Example.* s.l. : Wordware Publishing, 2005. ISBN:1556220782.

Buckland, Mat y Collins, Mark. 2002. *AI Techniques for Game Programming .* s.l. : Thomson Course Technology, 2002. ISBN:193184108X.

Champandard, Alex J. 2003. *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors.* s.l. : New Riders Publishing, 2003. ISBN : 1-5927-3004-3 .

Conocimientos-La divisa del nuevo milenio. *¿Qué es el Aprendizaje? Desde la Inteligencia Artificial.* [En línea] http://conocimientosweb.net/portal/article150.html.

Luis, Joe y Guevara, Omar E. Junio 2007. *Algoritmos Genéticos en Entornos Virtuales.* Universidad de las Ciencias Informáticas.La Habana : s.n., Junio 2007.

Rabin, Steve. 2002. *AI Game Programming Wisdom.* s.l. : Charles River Media, 2002. ISBN:1584500778.

Rangel, Norman y Santana, Jorge L. 1999. Realidad Virtual y sus aplicaciones en Internet. *Realidad Virtual.Conceptos Fundamentales.* [En línea] Octubre de 1999. http://web.archive.org/web/20040603134439/www.utp.ac.pa/seccion/topicos/realidad/cap1.htm.

Samper Márquez, Juan J. 1997. REDcientífica. *Introducción a los Sistemas Expertos .* [En línea] 1997. http://www.redcientifica.com/doc/doc199908210001.html.

Santaella Vallejo, Juan. *Un Sistema Experto: MYCIN.* 5º de Ing. De Telecomunicación.Inteligencia de Redes de Comunicación.

Schwab, Brian. 2004. *AI Game Engine Programming.* s.l. : Charles River Media, 2004. ISBN:1584503440.

GLOSARIO DE TÉRMINOS

-A-

Algoritmo: Un conjunto de reglas bien definidas para la solución de un problema en un número finito de pasos.

Aleatorio: Al azar, que no sigue un patrón, secuencia u orden determinado.

Aplicación: En informática las aplicaciones son los programas con los cuales el usuario final interactúa, es decir, son aquellos programas que permiten la interacción entre el usuario y la computadora. Esta comunicación se lleva a cabo cuando el usuario elige entre las diferentes opciones o realiza actividades que le ofrece el programa.

Arco: Es la unión entre dos nodos y representa la dependencia entre dos variables.

-E-

Entornos Virtuales: Pueden ser descritos como un espacio conceptual en el que un usuario establece una comunicación (interacción), en condiciones de tiempo y espacio posiblemente distintas, con otros usuarios (o su representación), o con elementos propios del entorno**.**

Entrenamiento: Adiestramiento, técnica que se realiza para perfeccionar el ejercicio de una actividad, para mejorar un comportamiento.

Estocástico: Un proceso estocástico es una sucesión de variables aleatorias indexadas por una variable continua o discreta. Cada una de las variables aleatorias del proceso tiene su propia función de distribución de probabilidad.

-H-

Heurística: Regla que permite orientar un algoritmo hacia la solución de un problema. Técnica de programación que permite a un sistema la creación gradual de un valor óptimo para una variable específica por medio del registro de los valores obtenidos en operaciones anteriores. Técnica empleada en los sistemas de inteligencia artificial.

-I-

Inferencia: Una inferencia es una evaluación que se realiza entre conceptos que, al interactuar, muestran sus propiedades de forma discreta, que permitirá trazar una línea lógica de causa-efecto, entre los diferentes puntos inferidos en la resolución del problema.

Iteraciones: Número determinado de veces que se realiza un proceso.

-M-

Métodos Adaptativos: Son aquellos que modifican su conducta durante su ejecución, atendiendo a los cambios que se producen en su entorno o en el propio programa.

Modelo probabilístico: Es un modelo que se basa en el cálculo matemático de probabilidades, o sea en el cálculo o determinación cuantitativa de la posibilidad de que se verifique un suceso.

-N-

Nodo: Un nodo es una variable aleatoria que puede tener varios estados, cada nodo será una estructura o registro que dispondrá de varios campos.

-P-

Programación evolutiva: La programación evolutiva (PE) es una rama de la computación evolutiva (o algoritmos evolutivos). La programación evolutiva es prácticamente una variación de los algoritmos genéticos, donde lo que cambia es la representación de los individuos. En el caso de la PE los individuos son tripletas cuyos valores representan estados de un autómata finito. Cada tripleta está formada por: *El valor del estado actual*un símbolo del alfabeto utilizado*El valor del nuevo estado*.

-S-

Simulador: Aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los simuladores reproducen sensaciones que en realidad no están sucediendo.

-T-

Técnica: Una técnica es un procedimiento o conjunto de estos, (reglas, normas o protocolos), que tienen como objetivo obtener un resultado determinado.

Topología: Es la organización y disposición de las los elementos de una determinada estructura formando otras estructuras.

ANEXOS

Figura 6. Respuesta de los individuos aplicando el Algoritmo Genético.

Figura 7. Resultado del proceso evolutivo.