Computación Blanda – Agosto de 2020. Universidad Tecnológica de Pereira – Facultad de Ingenierías. Sistemas y Computación 1

Computación Blanda

Soft Computing

Autor: Marlon Deyber Restrepo Rodriguez

*IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: Marlon.restrepo@utp.edu.co

*Resumen*— Este documento presenta un resumen de las líneas clásicas de la Computación Blanda: redes neuronales, lógica difusa, sistemas expertos, algoritmos genéticos y machine learning. El objetivo del documento es brindar una panorámica general de las temáticas, mostrando su relación con las técnicas de inteligencia artificial. La diferencia entre el paradigma de Inteligencia Artificial y la computación blanda está centrada en el mecanismo de inferencia utilizado y su aplicación a la solución de problemas tomados de lo cotidiano, de las teorías de conocimiento y de su relación con ciencias afines.

*Palabras clave—* sistemas, redes, inteligencia artificial, software, computación, investigación, industria, genético, aprendizaje.

*Abstract*— This document presents a summary of the classic lines of Soft Computing: neural networks, fuzzy logic, expert systems, genetic algorithms and machine learning. The objective of the document is to provide a general overview of the topics, showing their relationship with artificial intelligence techniques. The difference between the Artificial Intelligence paradigm and soft computing is centered on the inference mechanism used and its application to the solution of problems taken from everyday life, from knowledge theories and their relationship with related sciences.

*Key Word*— systems, networks, artificial intelligence, software, computing, research, industry, genetic, learning.

# INTRODUCCIÓN

La temática de la Computación Blanda se encuentra enmarcada en el paradigma de la Inteligencia Artificial. La diferencia con dicho paradigma radica en que la Computación Blanda está centrada en la aplicación pragmática de las teorías de la Inteligencia Artificial a la solución de problemas complejos en diversos campos del conocimiento.

Las líneas derivadas de la Computación Blanda, se configuran en las siguientes tendencias: a) Redes Neuronales Artificiales,

b) Lógica Difusa, c) Sistemas Expertos, d) Algoritmos Genéticos, e) Deep Learning (Machine Learning).

En los siguientes apartados se presenta un resumen de dichas tendencias.

* 1. REDES NEURONALES

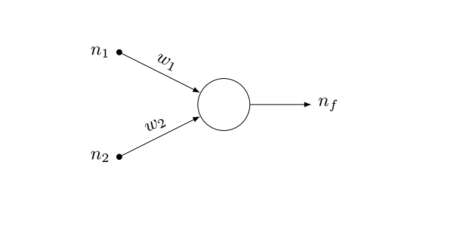
A pesar de su nombre, las redes neuronales no tienen un concepto demasiado complicado detrás de ellas. El nombre, como podéis imaginar, viene de la idea de imitar el funcionamiento de las redes neuronales de los organismos vivos: un conjunto de neuronas conectadas entre sí y que trabajan en conjunto, sin que haya una tarea concreta para cada una. Con la experiencia, las neuronas van creando y reforzando ciertas conexiones para "aprender" algo que se queda fijo en el tejido.

Ahora bien, por bonito que suene esto, el enfoque biológico no ha sido especialmente útil: las redes neuronales han ido moviéndose para tener un foco en matemáticas y estadística. Se basan en una idea sencilla: dados unos parámetros hay una forma de combinarlos para predecir un cierto resultado. Por ejemplo, sabiendo los píxeles de una imagen habrá una forma de saber qué número hay escrito, o conociendo la carga de servidores de un Centro de Procesamiento de Datos (CPD), su temperatura y demás existirá una manera de saber cuánto van a consumir, como hacía Google. El problema, claro está, es que no sabemos cómo combinarlos.

Las redes neuronales permiten buscar la combinación de parámetros que mejor se ajusta a un determinado problema.

Las redes neuronales son un modelo para encontrar esa combinación de parámetros y aplicarla al mismo tiempo. En el lenguaje propio, encontrar la combinación que mejor se ajusta es "entrenar" la red neuronal. Una red ya entrenada se puede usar luego para hacer predicciones o clasificaciones, es decir, para "aplicar" la combinación.

Para entender bien cómo funciona esta red vamos a ir con un ejemplo. Supongamos que sois alumnos de una clase en la que el profesor no ha dicho exactamente cómo va a poner las notas. Para empezar, supongamos que sólo habéis hecho dos exámenes y tenéis la nota de cada uno de ellos y la final.

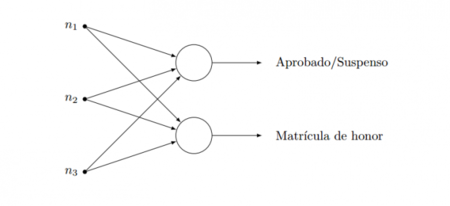
Figura 1.

La unidad básica de la red neuronal: el perceptrón. Las entradas son las dos notas, n1 y n2, cada una con su correspondiente peso wn (lo que hay que encontrar). La salida, nf, será 1 si está aprobado y 0 si se va a septiembre.

¿Cómo usamos una red neuronal para saber cuánto vale cada examen? Aquí nos bastará con la unidad fundamental de la red neuronal: el perceptrón. Un perceptrón es un elemento que tiene varias entradas con un cierto peso cada una. Si la suma de esas entradas por cada peso es mayor que un determinado número, la salida del perceptrón es un uno. Si es menor, la salida es un cero.

En nuestro ejemplo, las entradas serían las dos notas de los exámenes. Si la salida es uno (esto es, la suma de las notas por su peso correspondiente es mayor que cinco), es un aprobado. Si es cero, suspenso. Los pesos son lo que tenemos que encontrar con el entrenamiento. En este caso, nuestro entrenamiento consistirá en empezar con dos pesos aleatorios (por ejemplo, 0.5 y 0.5, el mismo peso a cada examen) y ver qué resultado da la red neuronal para cada alumno. Si falla en algún caso, iremos ajustando los pesos poco a poco hasta que esté todo bien ajustado.

Por ejemplo, si un alumno con muy buena nota en el segundo examen ha suspendido el curso, bajaremos el peso del segundo examen porque claramente no influye demasiado. Poco a poco acabaremos encontrando los pesos que se ajusten a las notas que puso el profesor. La idea del ajuste o retroalimentación es ir adaptando la red a la información "oculta" que tienen los datos que le pasamos para que aprenda.

Figura 1.2

Como decía antes, esto es el ejemplo simple. Quizás queramos complicarlo más, poniendo más exámenes (más nodos de entrada) o queriendo sacar más resultados, como pueda ser un perceptrón cuya salida sea uno si el alumno tiene matrícula de honor.

Multiplicando la potencia: redes multicapa

El ejemplo que he puesto antes funciona pero no se puede decir que sea demasiado potente. Pero quizás es que es demasiado simple. ¿No decíamos al principio que las redes neuronales eran un grupo de neuronas conectadas entre sí? ¿Cómo se logra esa "conexión" en las redes neuronales?

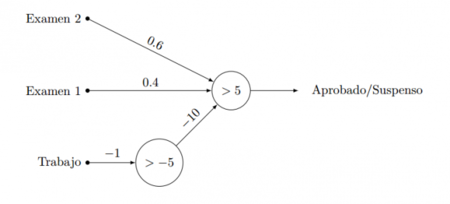
El concepto que nos falta aquí es el de las capas. Y para explicarlo vamos a seguir con nuestro ejemplo del profesor que pone notas sin decir cómo, añadiendo un trabajo que había que entregar.

Resulta que se da una situación curiosa. Hay dos alumnos que tienen la misma nota en los exámenes, dos dieces, pero uno tiene un 7 en el trabajo y otro un 4. El del 7 ha aprobado el curso, pero el del 4 no. Hay un alumno que tiene un 10 en el trabajo y 4.99 en los dos exámenes y que está suspenso.

Podemos intentar entrenar una red neuronal como la de antes todo lo que queramos en esta situación que no va a funcionar bien. Es posible que funcione en muchos casos, pero no va a ser perfecta. Y es que parece que la nota del trabajo no influye salvo que lo suspendas, en cuyo caso estás suspenso directamente. Es un filtro, un uno o un cero que tenemos que sacar en la red neuronal antes de poder dar el resultado de aprobado o suspendido en el curso... Ya deberíais ver por dónde van los tiros.

Efectivamente: necesitamos más capas. Necesitamos un perceptrón intermedio que nos diga si el trabajo está aprobado o no, y contar eso en el perceptrón de salida. Una posibilidad sería una red como la siguiente:

Figura 1.3

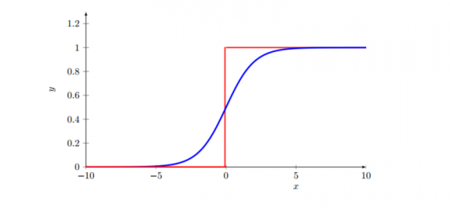


El primer perceptrón mira si la nota del trabajo multiplicada por menos uno es mayor que menos cinco (o, lo que es lo mismo, si la nota es menor que cinco). Si lo es, entonces su salida es uno. Al multiplicarla por menos diez en la entrada del segundo perceptrón, forzará siempre un suspenso. Si el trabajo está aprobado, la salida del primer perceptrón será 0 y no afectará a la media de los exámenes.

¿Qué hemos logrado con esto? O, más generalmente, ¿para qué nos sirven las capas? Lo que hemos logrado ha sido añadir información que no estaba antes. Cogemos los datos de entrada, los exploramos y sacamos las características que mejor nos ayuden a entender qué está pasando.

Lo más interesante de las capas es algo que aquí no hemos visto. En el ejemplo he puesto otra capa muy burdamente, pero lo que se suele hacer es poner varias con varios nodos, cada uno conectado a todas las entradas anteriores. Lo bueno viene cuando, durante el proceso de aprendizaje, cada capa "aprende" a encontrar y detectar las características que mejor ayudan a clasificar los datos. En nuestro ejemplo, durante el ajuste la primera capa aprendería que los alumnos con el trabajo suspenso suspenden el curso. Si cogiésemos una red para detectar números escritos a mano, puede que las capas ocultas aprendiesen a detectar trazos rectos o curvados que sirvan para decidir si estamos ante un uno o un ocho, por ejemplo.

Más allá de perceptrones: sigmoides, redes profundas y redes convolucionales

Figura 1.4

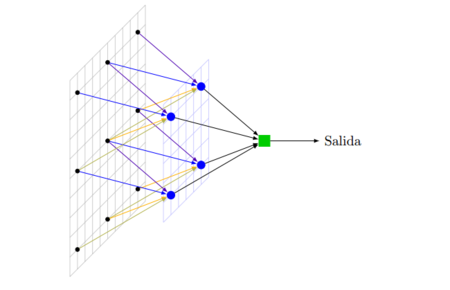
En rojo, la función "escalón". En azul, la sigmoide, una aproximación más suave pero con la misma idea.

Como siempre, hasta ahora nos hemos centrado en simplificaciones para entender bien los conceptos de redes neuronales. En la realidad, las cosas se complican bastante. Por ejemplo, se dejan de usar perceptrones para usar otras "neuronas" con un comportamiento más suave, usando funciones como la sigmoide. La idea es que pequeños cambios en los pesos provoquen pequeños cambios en la salida de la red, para así poder hacer más "fácil" el aprendizaje.

Las capas también se complican. Nos podemos encontrar varias capas intermedias con varias neuronas cada una, llegando a lo que llaman "redes neuronales profundas". La idea es que con más capas con más neuronas cada una se pueden mejorar las predicciones en conjuntos de datos más complicados. Este artículo, por ejemplo, explica desde un punto de vista visual y matemático cómo afectan las multiples capas y unidades al funcionamiento de la red neuronal.

El siguiente paso son redes convolucionales, que están funcionando muy bien en reconocimiento de voz y procesamiento de imágenes. En una red neuronal como las que hemos visto antes, pondríamos una neurona para cada píxel de una imagen y después pondríamos varias capas con varias neuronas, todas conectadas entre sí, para tratar de encontrar un número en una foto, por ejemplo. El problema es que no es demasiado efectivo (imaginaos todos los pesos que habría computar para una red que acepte imágenes de 1920x1080)

Figura 1.5



La idea de las redes convolucionales es tratar de buscar características locales en pequeños grupos de entradas (en el caso de las imágenes, de píxeles), como puedan ser bordes o colores más o menos homogéneos. Es la misma idea que comentábamos cuando introducíamos las capas unos párrafos más arriba, pero con una variación: buscamos características no en toda la imagen sino sólo en pequeñas regiones. Además, buscamos siempre detectar la misma característica en todos los grupos, por lo que podemos repetir esa estructura y reducir los ajustes que tenemos que hacer.

Para llevar a cabo esta idea, ponemos un mismo grupo de neuronas por cada grupo de entradas (por ejemplo, un cuadrado de 3x3 píxeles en una imagen o una secuencia de 4 mediciones en un archivo de sonido). La idea es que todos los elementos que metamos en la capa (llamada capa de convolución) tienen los mismos pesos por cada entrada, y se reduce considerablemente el número de parámetros. Si metemos más capas, la red neuronal podrá descubrir más y más complejas características de la imagen: se puede empezar por colores o bordes orientados y acabar con capas que se activan con formas circulares o cuadradas, por poner un ejemplo.

Después de las capas de convolución se suele poner otra red neuronal "tradicional", que ahora tendrá más fácil el trabajo: no tiene que valorar cada píxel por separado sino que mira a un conjunto de características de alto nivel de la imagen. Ya no se trata de decidir si la imagen es un coche sabiendo que el píxel 1208 es amarillo y el 1209 es verde, sino quizás sabiendo que hay una forma rectangular en la imagen con dos formas circulares en la parte inferior. De nuevo, se trata de extraer la información "oculta" en la entrada para tratar de encontrar qué es lo que define esos datos.

* 1. LÓGICA DIFUSA

La Lógica Difusa proporciona un mecanismo de inferencia que permite simular los procedimientos de razonamiento humano en sistemas basados en el conocimiento. La teoría de la lógica difusa proporciona un marco matemático que permite modelar la incertidumbre de los procesos cognitivos humanos de forma que pueda ser tratable por un computador. En este primer capítulo se describirán los fundamentos y características de este mecanismo de representación de la incertidumbre.

La lógica difusa (fuzzy logic, en inglés) se adapta mejor al mundo real en el que vivimos, e incluso puede comprender y funcionar con nuestras expresiones, del tipo «hace mucho calor», «no es muy alto», «el ritmo del corazón está un poco acelerado», etc.

La clave de esta adaptación al lenguaje se basa en comprender los cuantificadores de cualidad para nuestras inferencias (en los ejemplos de arriba, «mucho», «muy» y «un poco»).

En la teoría de conjuntos difusos se definen también las operaciones de unión, intersección, diferencia, negación o complemento, y otras operaciones sobre conjuntos (véase también subconjunto difuso), en los que se basa esta lógica.

Para cada conjunto difuso, existe asociada una función de pertenencia para sus elementos, que indica en qué medida el elemento forma parte de ese conjunto difuso. Las formas de las funciones de pertenencia más típicas son trapezoidal, lineal y curva.

Se basa en reglas heurísticas de la forma SI (antecedente) ENTONCES (consecuente), donde el antecedente y el consecuente son también conjuntos difusos, ya sea puros o resultado de operar con ellos. Sirvan como ejemplos de regla heurística para esta lógica (nótese la importancia de las palabras «muchísimo», «drásticamente», «un poco» y «levemente» para la lógica difusa):

SI hace muchísimo frío. ENTONCES aumento drásticamente la temperatura.

SI voy a llegar un poco tarde. ENTONCES aumento levemente la velocidad.

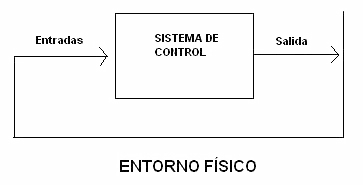
Los métodos de inferencia para esta base de reglas deben ser sencillos, versátiles y eficientes. Los resultados de dichos métodos son un área final, fruto de un conjunto de áreas solapadas entre sí (cada área es resultado de una regla de inferencia). Para escoger una salida concreta a partir de tanta premisa difusa, el método más usado es el del centroide, en el que la salida final será el centro de gravedad del área total resultante.

Las reglas de las que dispone el motor de inferencia de un sistema difuso pueden ser formuladas por expertos o bien aprendidas por el propio sistema, haciendo uso en este caso de redes neuronales para fortalecer las futuras tomas de decisiones.

Los datos de entrada suelen ser recogidos por sensores que miden las variables de entrada de un sistema. El motor de inferencias se basa en chips difusos, que están aumentando exponencialmente su capacidad de procesamiento de reglas año a año.

Un esquema de funcionamiento típico para un sistema difuso podría ser de la siguiente manera:

**Figura 2**



En la figura, el sistema de control hace los cálculos con base en sus reglas heurísticas, comentadas anteriormente. La salida final actuaría sobre el entorno físico, y los valores sobre el entorno físico de las nuevas entradas (modificado por la salida del sistema de control) serían tomadas por sensores del sistema.

Por ejemplo, imaginando que nuestro sistema difuso fuese el climatizador de un coche que se autorregula según las necesidades: Los chips difusos del climatizador recogen los datos de entrada, que en este caso bien podrían ser la temperatura y humedad simplemente. Estos datos se someten a las reglas del motor de inferencia (como se ha comentado antes, de la forma SI... ENTONCES... ), resultando un área de resultados. De esa área se escogerá el centro de gravedad, proporcionándola como salida. Dependiendo del resultado, el climatizador podría aumentar la temperatura o disminuirla dependiendo del grado de la salida.

I.2.1 LOGICA DIFUSA COMPENSATORIA (LDC)

La LDC es un modelo lógico multivalente que permite la modelación simultánea de los procesos deductivos y de toma de decisiones. El uso de la LDC en los modelos matemáticos permite utilizar conceptos relativos a la realidad siguiendo patrones de comportamiento similares al pensamiento humano. Las características más importantes de estos modelos son: La flexibilidad, la tolerancia con la imprecisión, la capacidad para moldear problemas no lineales y su fundamento en el lenguaje de sentido común. Bajo este fundamento se estudia específicamente cómo acondicionar el modelo sin condicionar la realidad.

La LDC utiliza la escala de la LD, la cual puede variar de 0 a 1 para medir el grado de verdad o falsedad de sus proposiciones, donde las proposiciones pueden expresarse mediante predicados. Un predicado es una función del universo X en el intervalo [0, 1], y las operaciones de conjunción, disyunción, negación e implicación, se definen de modo que restringidas al dominio [0, 1] se obtenga la Lógica Booleana.

Las distintas formas de definir las operaciones y sus propiedades determinan diferentes lógicas multivalentes que son parte del paradigma de la LD. Las lógicas multivalentes se definen en general como aquellas que permiten valores intermedios entre la verdad absoluta y la falsedad total de una expresión. Entonces el 0 y el 1 están asociados ambos a la certidumbre y la exactitud de lo que se afirma o se niega y el 0,5 a la vaguedad y la incertidumbre máximas. En los procesos que requieren toma de decisiones, el intercambio con los expertos lleva a obtener formulaciones complejas y sutiles que requieren de predicados compuestos. Los valores de verdad obtenidos sobre estos predicados compuestos deben poseer sensibilidad a los cambios de los valores de verdad de los predicados básicos.

Esta necesidad se satisface con el uso de la LDC, que renuncia al cumplimiento de las propiedades clásicas de la conjunción y la disyunción, contraponiendo a éstas la idea de que el aumento o disminución del valor de verdad de la conjunción o la disyunción provocadas por el cambio del valor de verdad de una de sus componentes, puede ser “compensado” con la correspondiente disminución o aumento de la otra. Estas propiedades hacen posible de manera natural el trabajo de traducción del lenguaje natural al de la Lógica, incluidos los predicados extensos si éstos surgen del proceso de modelación.

En la LDC, el operador conjunción, expresado como c (and) es la media geométrica.

I.2.2 APLICACIONES

La lógica difusa se utiliza cuando la complejidad del proceso en cuestión es muy alta y no existen modelos matemáticos precisos, para procesos altamente no lineales y cuando se envuelven definiciones y conocimiento no estrictamente definido (impreciso o subjetivo).

En cambio, no es una buena idea usarla cuando algún modelo matemático ya soluciona eficientemente el problema, cuando los problemas son lineales o cuando no tienen solución.

Esta técnica se ha empleado con bastante éxito en la industria, principalmente en Japón, extendiéndose sus aplicaciones a multitud de campos. La primera vez que se usó de forma importante fue en el metro japonés, con excelentes resultados. Posteriormente se generalizó según la teoría de la incertidumbre desarrollada por el matemático y economista español Jaume Gil Aluja.

A continuación se citan algunos ejemplos de su aplicación:

* Sistemas de control de acondicionadores de aire
* Sistemas de foco automático en cámaras fotográficas
* Electrodomésticos familiares (frigoríficos, lavadoras...)
* Optimización de sistemas de control industriales
* Sistemas de escritura
* Mejora en la eficiencia del uso de combustible en motores
* Sistemas expertos del conocimiento (simular el comportamiento de un experto humano)
* Tecnología informática
* Bases de datos difusas: Almacenar y consultar información imprecisa. Para este punto, por ejemplo, existe el lenguaje FSQL.
* ...y, en general, en la gran mayoría de los sistemas de control que no dependen de un Sí/No.
  1. SISTEMAS EXPERTOS

Sistemas expertos(ES, siglas del término Expert System) es un sistema de información basado en el conocimiento que usa su conocimiento de un área de aplicación compleja y específica a fin de actuar como un consultor experto para los usuarios finales. Los sistemas expertos proporcionan respuestas sobre un área problemática muy específica al hacer inferencias semejantes a las humanas sobre los conocimientos obtenidos en una base de conocimientos especializados.

Es una rama de la Inteligencia artificial; son sistemas informáticos que simulan el proceso de aprendizaje, de memorización, de razonamiento, de comunicación y de acción en consecuencia de un experto humano en cualquier rama de la ciencia.

I.3.1 CARACTERISTICAS

Para que un sistema actúe como un verdadero experto, es deseable que reúna, en lo posible, lo más importante de las características de un experto humano, esto es:

* Habilidad para adquirir conocimiento.
* Fiabilidad, para poder confiar en sus resultados o apreciaciones.
* Solidez en el dominio de su conocimiento.
* Capacidad para resolver problemas.

Dada la complejidad de los problemas que usualmente tiene que resolver un sistema experto, puede existir cierta duda en el usuario sobre la validez de respuesta obtenida. Por este motivo, es una condición indispensable que un sistema experto sea capaz de explicar su proceso de razonamiento o dar razón del por qué solicita tal o cual información o dato.

Estas características le permiten almacenar datos y conocimiento, sacar conclusiones lógicas, tomar decisiones, aprender de la experiencia y los datos existentes, comunicarse con expertos humanos, explicar el porqué de las decisiones tomadas y realizar acciones como consecuencia de todo lo anterior. Técnicamente un sistema experto, contiene una base de conocimientos que incluye la experiencia acumulada de expertos humanos y un conjunto de reglas para aplicar ésta base de conocimientos en una situación particular que se le indica al programa. Cada vez el sistema se mejora con adiciones a la base de conocimientos.

I.3.2 TIPOS DE SISTEMAS EXPERTOS

Existen tres tipos de sistemas expertos:

Basados en reglas: Aplicando reglas heurísticas apoyadas generalmente en lógica difusa para su evaluación y aplicación.

Basados en casos CBR (Case Based Reasoning): Aplicando el razonamiento basado en casos, donde la solución a un problema similar planteado con anterioridad se adapta al nuevo problema.

Basados en redes: Aplicando redes bayesianas, basadas en estadística y el teorema de Bayes.

I.3.3 ¿PORQUE UTILIZAR UN SISTEMA EXPERTO?

Con su ayuda, personas con poca experiencia pueden resolver problemas que requieren un "conocimiento formal especializado". Se pueden obtener conclusiones y resolver problemas de forma más rápida que los expertos humanos. Estos sistemas razonan pero en base a un conocimiento adquirido y no tienen sitio para la subjetividad. Se ha comprobado que tienen al menos, la misma competencia que un especialista humano.

Su uso es especialmente recomendado en las siguientes situaciones:

Cuando los expertos humanos en una determinada materia son escasos.

En situaciones complejas, donde la subjetividad humana puede llevar a conclusiones erróneas.

Cuando es muy elevado el volumen de datos que ha de considerarse para obtener una conclusión.

Diferencias entre sistemas expertos y los tradicionales

Sistemas expertos:

Toman Decisiones

Calculan Resultados

Basados en Heurísticas

Dan Explicaciones de los Resultados

Usan Reglas de Inferencia

Accedan Bases de Conocimientos (Deductivas)

Centrados en el Experto y el Usuario

Manejan Conocimiento Impreciso, Contradictorio o Incompleto

Usan Datos y Lenguajes Simbólicos

Sistemas tradicionales:

Calculan resultados

Basados en Algoritmos

Dan Resultados sin Explicaciones

Usan Secuenciación, Ciclos y Condicionales

Acceden a Bases de Datos

Centrados en el Analista y el Programador

Conocimientos Precisos, Completos y Exactos

Usan Datos Numéricos y Lenguajes Procedurales.

I.3.4 VENTAJAS

Producción y productividad mayores. Pueden trabajar más rápido que lo humanos. Están disponibles ininterrumpidamente de día y noche, ofreciendo siempre su máximo desempeño. Pueden duplicarse ilimitadamente, i.e. tener tantos de ellos como se requieran.

Mayor calidad: Dan la probabilidad de aumentar la calidad proporcionando asesoría consistente y reduciendo las tasas de error.

Operación en entornos peligrosos. Muchas tareas requieren que los seres humanos operen en entornos hostiles y peligrosos.

Captación de experiencia escasa y su dimensión. Uno de los principales beneficios de los sistemas expertos es su facilidad de trasmitir experiencia a través de fronteras internacionales.

Siempre se ajustan a las normas establecidas y son consistentes en su desempeño, i.e. no desarrollan apreciaciones subjetivas, tendenciosas, irracionales o emocionales.

Confiabilidad. Los sistemas expertos son confiables.No padece de olvido, fatiga, dolor o comete errores de cálculo.

No requiere un sueldo, promociones, seguros médicos, incapacidades.

Accesibilidad al conocimiento y escritorios de vida. Hacen accesible el conocimiento (y la información) a mucha gente en diversos lugares. Siempre están dispuestos a dar explicaciones, asistir o enseñar a la gente, así como a aprender.

Pueden tener una vida de servicio ilimitada. Funciones incrementadas de otros sistemas expertos. La integración de un sistema experto con otros sistemas expertos hacen que estos últimos se vuelvan más eficientes, los sistemas integrados abarcan más aplicaciones, trabajan más rápido y producen resultados de mayor calidad.

Capacidad para trabajar con información incompleta o inconcreta. En contraste con los sistemas de cómputo convencionales, un sistema experto puede trabajar con información incompleta al igual que los expertos humanos.

Impartición de capacitación. Es factible que un sistema experto facilite la capacitación. El personal nuevo que trabaja con un sistema experto se vuelve más experimentado. La fusión de explicación puede servir como un dispositivo de enseñanza y de ese modo puede efectuar apuntes que tal vez se inserten en la base de conocimiento.

Mejoramiento de las funciones para resolver problemas. Un sistema experto mejora la solución de problemas permitiendo la integración de juicios de expertos de primera línea en el análisis. De este modo, un sistema experto tiene la posibilidades resolver problemas cuyo enlace y conocimiento supera a los de cualquier individuo.

Reducción del tiempo para la toma de decisiones. Con el empleo de la recomendación del sistema, un ser humano puede tomar decisiones mucho más rápido.

Reducción del tiempo fuera de servicio. Muchos sistemas expertos operacionales se emplean para diagnosticar malos funcionamientos y prescribir reparaciones.

I.3.5 DESVENTAJAS

Tienen una noción muy limitada acerca del contexto de problema, i.e. no pueden percibir todas las cosas que un experto humano puede apreciar de un situación.

Pueden existir decisiones que sólo son de competencia para un ser humano y no una máquina.

No saben como subsanar sus limitaciones, e.g. no son capaces de trabajar en equipo o investigar algo nuevo.

Son muy costosos de desarrollar y mantener.

Tareas que realiza un Sistema Experto Monitorización.

La monitorización es un caso particular de la interpretación, y 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. Se trata de que el programa pueda determinar en cada momento el estado de funcionamiento de sistemas complejos, anticipándose a los posibles incidentes que pudieran acontecer.

I.4 ALGORITMOS GENETICOS

un **algoritmo genético** (**AG**) es una técnica de resolución de problemas que imita a la evolución biológica como estrategia para resolver problemas, englobándose dentro de lo que antes hemos denominado técnicas basadas en poblaciones. Dado un problema específico a resolver, la entrada del AG es un conjunto de soluciones potenciales a ese problema, codificadas de alguna manera, y una métrica llamada **función de aptitud**, o **fitness**, que permite evaluar cuantitativamente a cada solución candidata. Estas candidatas pueden ser soluciones que ya se sabe que funcionan, con el objetivo de que el AG las mejore, pero se suelen generar aleatoriamente.

A partir de ahí, AG evalúa cada candidata de acuerdo con la función de aptitud. Por supuesto, se debe tener en cuenta que estas primeras candidatas generadas aleatoriamente, tendrán una eficiencia mínima con respecto a la resolución del problema, y la mayoría no funcionarán en absoluto. Sin embargo, por puro azar, unas pocas pueden ser prometedoras, pudiendo mostrar algunas características que muestren, aunque sólo sea de una froma débil e imperfecta, cierta capacidad de  solución del problema.

I.4.1 METODOS DE REPRESENTACION

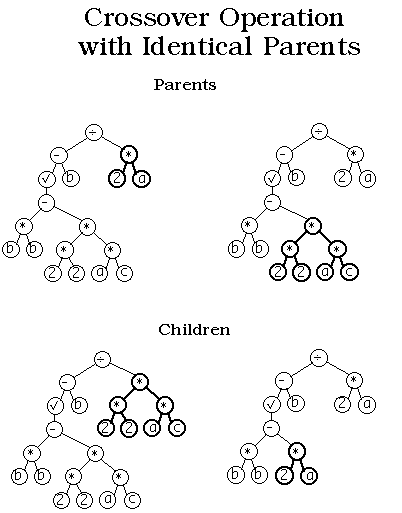
Antes de que un algoritmo genético pueda ponerse a trabajar en un problema, se necesita un método para codificar las soluciones potenciales del problema de forma que el propio algoritmo pueda procesarlas aplicando las operaciones que le permiten evolucionar. Un enfoque común es codificar las soluciones como cadenas binarias: secuencias de 11's y 00's, donde el dígito de cada posición representa el valor de algún aspecto de la solución.

Otro método similar consiste en codificar las soluciones como cadenas de enteros o números decimales, donde cada posición, de nuevo, representa algún aspecto particular de la solución. Este método permite una mayor precisión y complejidad que el método comparativamente restringido de utilizar sólo números binarios, y a menudo está intuitivamente más cerca del espacio de soluciones del problema. Esta técnica se utilizó, por ejemplo, en el trabajo de Steffen Schulze-Kremer, que escribió un algoritmo genético para predecir la estructura tridimensional de una proteína, basándose en la secuencia de aminoácidos que la componen.

El objetivo de cualquier método es que facilite la definición de operadores que causen los cambios aleatorios en las soluciones candidatas seleccionadas: cambiar un 00 por un 11 o viceversa, sumar o restar al valor de un número una cantidad elegida al azar, o cambiar una letra por otra.

Otra estrategia, desarrollada principalmente por John Koza, de la Universidad de Stanford, y denominada **programación genética**, representa a los programas como estructuras de datos ramificadas llamadas **árboles**. En este método, los cambios aleatorios pueden generarse cambiado el operador, alterando el valor de un cierto nodo del árbol, o sustituyendo un subárbol por otro.

**Figura 3**

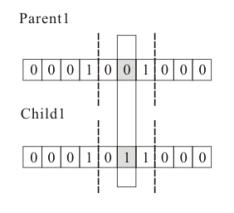


I.4.2 METODOS DE CAMBIO

Una vez que la selección ha elegido a los individuos aptos, éstos deben ser alterados aleatoriamente con la esperanza de mejorar su aptitud para la siguiente generación. Existen dos estrategias básicas para realizar esta tarea:

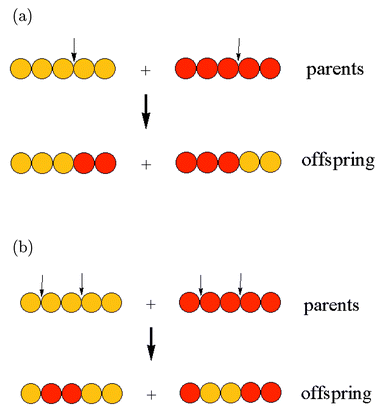
La primera y más sencilla es la que se conoce como **mutación**. Al igual que una mutación en los seres vivos cambia un gen por otro, una mutación en un algoritmo genético también causa pequeñas alteraciones en puntos concretos de la codificación del individuo.

**Figura 3.1**



El segundo método se llama **cruzamiento**, y consiste en seleccionar a dos individuos para que intercambien segmentos de su código genético, 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. Las formas comunes de cruzamiento 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, al **cruzamiento en dos puntos**, en el que se intercembian los genes que aparecen en el intervalo dee genes delimitados por dos puntos, 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 o al valor en esa posición del genoma del otro padre, elegido con un 50% de probabilidad.

**Figura 3.2**

****

I.4.3 DESVENTAJAS Y LIMITACIONES

A pesar de las bondades apuntadas anteriormente para este tipo de algoritmos, debemos mencionar algunas de las limitaciones que presentan:

* Como ocurre en muchos problemas de optimización, si tiene una alta complejidad entonces **la función de evaluación puede resultar demasiado costosa** en términos de tiempo y recursos.
* Hay casos en los que, dependiendo de los parámetros que se utilicen para la evaluación, **el algoritmo puede no converger a una solución óptima**, o bien terminar en una convergencia prematura con resultados no satisfactorios (es decir, devolviendo un óptimo local o incluso un punto arbitrario).
* **No poseen una buena escalabilidad** con la complejidad. En sistemas en los que intervienen muchas variables, componentes o elementos el espacio de búsqueda asociado puede crecer de manera exponencial debido, entre otras cosas, a las relaciones no lineales que puedan surgir entre los subconjuntos de variables.
* La mejor solución lo es solo en comparación a otras soluciones, por lo que no se tiene demasiado claro un criterio de cuándo detenerse ya que no se cuenta con una solución específica.
* No es recomendable utilizarlos para problemas que buscan respuesta a problemas con soluciones simples como Sí/No o problemas de decisión, ya que el algoritmo difícilmente convergerá y el resultado será casi aleatorio.
* El diseño de la función de aptitud (fitness) y la selección de los criterios de mutación entre otros, necesitan de cierta pericia y **conocimiento previo del problema** para obtener buenos resultados.

I.5 DEEP LEARNING

es un conjunto de aprendizaje automático que intenta modelar abstracciones de alto nivel en datos usando arquitecturas computacionales que admiten transformaciones no lineales múltiples e iterativas de datos expresados en forma matricial o tensorial.

El aprendizaje profundo es parte de un conjunto más amplio de métodos de aprendizaje automático basados en asimilar representaciones de datos. Una observación (por ejemplo, una imagen) puede ser representada en muchas formas (por ejemplo, un vector de pixeles), pero algunas representaciones hacen más fácil aprender tareas de interés (por ejemplo, "¿es esta imagen una cara humana?") sobre la base de ejemplos, y la investigación en esta área intenta definir qué representaciones son mejores y cómo crear modelos para reconocer estas representaciones.

Varias arquitecturas de aprendizaje profundo, como redes neuronales profundas, redes neuronales profundas convolucionales, y redes de creencia profundas, han sido aplicadas a campos como visión por computador, reconocimiento del habla y reconocimiento de señales de audio y música, y han mostrado producir resultados de vanguardia en varias tareas.

I.5.1COMO FUNCIONA DEPP LEARNING

La mayor parte de los métodos de aprendizaje emplean arquitecturas de **redes neuronales**, por lo que, a menudo, los modelos de aprendizaje profundo se denominan **redes neuronales profundas**.

El término “profundo” suele hacer referencia al número de capas ocultas en la red neuronal. Las redes neuronales tradicionales solo contienen dos o tres capas ocultas, mientras que las redes profundas pueden tener hasta 150.

Los modelos de Deep Learning se entrenan mediante el uso de extensos conjuntos de datos etiquetados y arquitecturas de redes neuronales que aprenden directamente a partir de los datos, sin necesidad de una extracción manual de características.

Uno de los tipos más populares de redes neuronales profundas son las conocidas como [redes neuronales convolucionales](https://es.mathworks.com/discovery/convolutional-neural-network.html)**(CNN** o**ConvNet)**. Una CNN convoluciona las características aprendidas con los datos de entrada y emplea capas convolucionales 2D, lo cual hace que esta arquitectura resulte adecuada para procesar datos 2D, tales como imágenes.

Las CNN eliminan la necesidad de una [extracción de características](https://es.mathworks.com/discovery/feature-extraction.html) manual, por lo que no es necesario identificar las características utilizadas para clasificar las imágenes. La CNN funciona mediante la extracción de características directamente de las imágenes. Las características relevantes no se entrenan previamente; se aprenden mientras la red se entrena con una colección de imágenes. Esta extracción de características automatizada hace que los modelos de Deep Learning sean muy precisos para tareas de visión artificial, tales como la clasificación de objetos.

I.5.2¿DIFRENCIA ENTRE MACHINE LEARNING Y DEEP LEARNIG?

El Deep Learning es una forma especializada de aprendizaje automático. Un flujo de trabajo de Machine Learning empieza con la extracción manual de las características relevantes de las imágenes. Estas características se utilizan entonces para crear un modelo que categoriza los objetos de la imagen. Con un flujo de trabajo de Deep Learning, las características relevantes se extraen directamente de las imágenes. Además, el aprendizaje profundo realiza un “aprendizaje completo”, es decir, se proporcionan datos sin procesar y una tarea que realizar, como puede ser una clasificación, a una red, la cual aprende cómo hacerlo automáticamente.

Otra diferencia clave es que con los algoritmos de Deep Learning la escala aumenta con los datos, mientras que, en el caso del aprendizaje superficial, existe convergencia. El aprendizaje superficial hace referencia a los métodos de aprendizaje automático que llegan a un punto muerto en cierto nivel de rendimiento cuando se agregan más ejemplos y datos de entrenamiento a la red.

Una ventaja fundamental de las redes de aprendizaje profundo es que suelen seguir mejorando a medida que aumenta el tamaño de los datos.

I.5.4 VENTAJAS DEL DEPP LEARNING

En el ámbito laboral reduce los costos y salarios adicionales. Por resultar un atractivo, conlleva a generar más ingresos. Se han desarrollado aplicaciones que realizan tareas que el hombre nunca hubiera podido realizar debido a su complejidad. Puede predecir situaciones a largo plazo. Reduce el tiempo que consume realizar cierta actividad.  
  
I.5.5 DESVENTAJAS DEL DEEP LEARNIG

Por ser SW requiere de constantes actualizaciones (mantenimiento).  
 Realizar estos sistemas expertos requiere de mucho tiempo y dinero.  
Crear maquinas que sean autosuficientes.  
 El hombre se siente superado cuando una maquina lo supera.

REFERENCIAS

R referencias en la Web:

[1]

h [ttps://computerhoy.com/reportajes/tecnologia/intel](https://computerhoy.com/reportajes/tecnologia/inteligencia-artificial-469917) i [gencia-artificial-469917](https://computerhoy.com/reportajes/tecnologia/inteligencia-artificial-469917)

[2]

<https://www.xataka.com/robotica-e-ia/las-redes-neuronales-que-son-y-por-que-estan-volviendo>

[3]

<https://www.esi.uclm.es/www/cglez/downloads/docencia/2011_Softcomputing/LogicaDifusa.pdf>

[4]

<https://es.wikipedia.org/wiki/L%C3%B3gica_difusa#:~:text=La%20l%C3%B3gica%20difusa%20(tambi%C3%A9n%20llamada,contextualizados%20y%20referidos%20entre%20s%C3%AD.>

[5]

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

[6]

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

[7]

<http://www.cs.us.es/~fsancho/?e=65>

[8]

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

[9]

<https://www.grapheverywhere.com/deep-learning/>

[10]

<https://es.mathworks.com/discovery/deep-learning.html>

Materia: Computación Blanda. Segundo Semestre de 2020. Docente: José Gilberto Vargas Cano.