Computación Blanda

Soft Computing

Autor: Luis Fernando Zuluaga Torres

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

Correo-e: luis.zuluaga1@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.**

1. 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

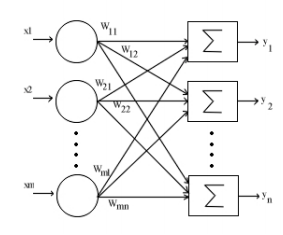
Las Redes Neuronales Artificiales (RNA) están inspiradas en la biología, esto significa que están formadas por elementos que se comportan de manera análoga a las neuronas (en las funciones más elementales) y están organizadas de una forma similar a la del cerebro, pero las analogías no son muchas más. Las características fundamentales de las RNA son:

• Aprenden de la experiencia: Las RNA pueden modificar su comportamiento como respuesta a su entorno. Dado un conjunto de entradas (quizá con las salidas deseadas), las RNA se ajustan para producir respuestas consistentes. Una amplia variedad de algoritmos de entrenamiento se ha desarrollado, cada uno con sus propias ventajas e inconvenientes.

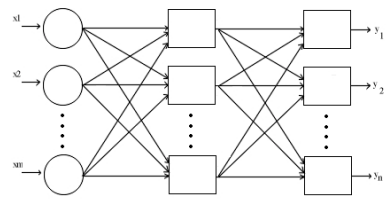
• Generalizan de ejemplos anteriores a los ejemplos nuevos: Una vez que la RNA esté entrenada, la respuesta de la red puede ser, hasta un cierto punto, insensible a pequeñas variaciones en las entradas, lo que las hace idóneas para el reconocimiento de patrones.

• Abstracción de la esencia de las entradas: Algunas RNA son capaces de abstraer información de un conjunto de entradas. Por ejemplo, en el caso de reconocimiento de patrones, una red puede ser entrenada en una secuencia de patrones distorsionados de una letra. Una vez que la red sea correctamente entrenada será capaz de producir un resultado correcto ante una entrada distorsionada, lo que significa que ha sido capaz de aprender algo que nunca había visto.

**Redes de capa simple:** A pesar de que una sola neurona puede realizar modelos simples de funciones, su mayor productividad viene dada cuando se organizan en redes. La red más simple es la formada por un conjunto de perceptrones a los que entra un patrón de entradas y proporcionan la salida correspondiente. Por cada perceptrón que tengamos en la red vamos a tener una salida, que se hallará como se hacía con un perceptrón solo, haciendo el sumatorio de todas las entradas multiplicadas por los pesos. Al representar gráficamente una red, se añade una "capa" inicial que no es contabilizada a efectos de computación, solamente sirve para distribuir las entradas entre los perceptrones. La denominaremos la capa 0. De esta manera, la representación gráfica de una red de capa simple sería la siguiente:



**Redes multicapa:** Las redes multicapa se forman por un conjunto de redes de capa simple en cascada unidas por pesos, donde la salida de una capa es la entrada de la siguiente capa. Generalmente son capaces de aprender funciones que una red de capa simple no puede aprender, por lo que ofrecen mejores capacidades computacionales. Para que este incremento en poder computacional sea tal, tiene que existir una función de activación no lineal entre las capas, por lo que generalmente se utilizará una función de activación sigmoidea en detrimento de la lineal o umbral. Para calcular la salida de una red multicapa se debe hacer de la misma manera que en las redes de capa simple, teniendo en cuenta que las salidas de una capa son las entradas de la siguiente capa.:



**Redes recurrentes:** Las redes consideradas hasta ahora no tienen conexiones entre pesos de la salida de una capa a la entrada de la misma capa o anteriores. Las redes que poseen esta característica son conocidas como redes recurrentes. Las redes recurrentes no tienen memoria, es decir, la salida solamente está determinada por las entradas y los pesos. Las capas recurrentes redireccionan previas salidas a entradas. Su salida es determinada por su entrada y sus salidas previas, por lo que se puede asemejar a la memoria a corto plazo de los seres humanos.

* 1. LÓGICA DIFUSA

La lógica difusa (fuzzy logic) permite tratar información imprecisa, como medio lista, temperatura baja o mucha fuerza, en términos de conjuntos difusos. La teoría de conjuntos difusos provee una herramienta matemática para aproximar el razonamiento de estos enunciados cuando la información disponible es incierta, incompleta, imprecisa o vaga. En Lógica Difusa, se utilizan conceptos relativos de la realidad, definiendo grados, variables de pertenencia y siguiendo patrones de razonamiento similares a los del pensamiento humano.

Una característica de la lógica difusa es el predicado, el predicado es lo que se afirma o niega de un objeto, nos ayudan a expresar ideas y a construir enunciados. Por otro lado, El universo es el conjunto de los elementos a los que se puede aplicar un predicado.

Un predicado clásico, es aquel que, al aplicarlo a los elementos de un universo, lo divide en dos subconjuntos: el de los elementos que verifican dicho predicado, y el de los que no lo verifican. Ejemplo:

Dado el universo A = {Números naturales menores que 10} y el predicado clásico P = “ser par”, podemos realizar la división en dos conjuntos claramente diferenciados: Subconjunto de elementos de A que verifican el predicado P. Pe={2,4,6,8} Subconjunto de elementos de A que no verifican el predicado P. ∼ Pe={1,3,5,7,9}.

Predicados difusos Hay predicados P que, al aplicarlos a los elementos de un universo, no lo dividen perfectamente en dos subconjuntos, el de los que cumplen dicho predicado y el de los que no lo cumplen. A este tipo de predicados se les denomina predicados difusos. Ejemplo: Tenemos el predicado P =” joven”, y lo aplicamos al conjunto A = {Jugadores de baloncesto}. En este caso para representar el predicado” joven “hacemos uso de los colores, partimos de los tonos amarillos que representarán a un jugador de baloncesto joven, hasta llegar en la escala a los tonos azules, que representarán a un jugador de baloncesto no joven.

Un conjunto clásico es una colección de elementos. Por ejemplo, puede ser el conjunto de elementos que verifican un predicado clásico. Como sabemos dado un subconjunto clásico A de X, le podemos asociar su función característica.

Conjuntos Difusos Los conjuntos difusos son aquellos cuyos elementos no tienen por qué pertenecer (grado de pertenencia 1) o no pertenecer (grado de pertenencia 0), sino que pertenecen según un cierto grado entre 0 y 1, donde el grado esta dado por la función de pertenencia del conjunto.

La función de pertenencia de un conjunto nos indica el grado en que cada elemento de un universo dado, pertenece a dicho conjunto. Es decir, la función de pertenencia de un conjunto A sobre un universo X será de la forma: µA : X → [0, 1], donde µA(x) = r si r es el grado en que x pertenece a A. Un conjunto difuso A en X puede representarse como un conjunto de pares ordenados de un elemento arbitrario x y su función de pertenencia, del siguiente modo A = {(x, µA(x))|x ∈ X}

Las funciones de pertenencia son una forma de representar gráficamente un conjunto difuso sobre un universo. A la hora de determinar una función de pertenencia, normalmente se eligen funciones sencillas, para que los cálculos no sean complicados. En particular, en aplicaciones en distintos entornos, son muy utilizadas las triangulares y las trapezoidales.

* 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.

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 esta 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 o al conjunto de reglas.

**Componentes de un sistema experto**

Separan conocimientos (reglas y hechos) y el procesamiento; se le añade una interfaz de usuario y un componente explicativo; los siguientes componentes pueden estar estructurados de formas muy variadas.

* Base de conocimientos: Contiene el conocimiento de los hechos y las experiencias de los expertos en un dominio determinado
* Mecanismo de inferencia: Puede simular la estrategia de solución de un experto
* Componente explicativo: Explica al usuario la estrategia de solución encontrada y el porqué de las decisiones tomadas
* Interfase de usuario: Sirve para que este 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

Los sistemas expertos, salvo excepciones, no están aislados, sino que forman parte de otros sistemas, expertos o convencionales. Existen dos tipos básicos de arquitectura de integración. En la primera, el sistema basado en el conocimiento forma parte de otro sistema principal. Así, si el sistema necesita comunicarse con el sistema basado en el conocimiento, entablará una comunicación directamente o a través de una red.

En la segunda el sistema basado en el conocimiento es el sistema principal y está conectado a otros sistemas basados en el conocimiento o convencionales, que le ayudan en su operación. Como ejemplo están los subsistemas que realizan complejos cálculos matemáticos necesarios durante el proceso de razonamiento.

En ambos casos debe garantizarse una comunicación fluida de todos los sistemas, aunque estén funcionando en plataformas diferentes, ya sea directamente o a través de una red local. Este aspecto es crítico en sistemas basados en el conocimiento en tiempo real, los cuales requieren un rápido acceso a la información relevante y a las bases de datos para poder ofrecer una solución inmediata y satisfactoria.

Por tanto, habrá que fijarse en las posibilidades de conexión a las bases de datos más conocidas y, en especial, a aquélla que se esté utilizando como estándar en la organización, así como la posibilidad de llamar a rutinas externas en diferentes lenguajes y viceversa, o la invocación del sistema basado en el conocimiento desde otros sistemas.

Un aspecto fundamental a tener en cuenta es la portabilidad de la herramienta (herramientas multiplataforma). En este aspecto hay que valorar no sólo si la herramienta es susceptible de funcionar en diferentes plataformas sino también el coste del cambio de plataforma. Existen herramientas que permiten pasar de una plataforma a otra sin apenas esfuerzo, lo que facilitará la comercialización y el uso de los sistemas que se desarrollen.

* 1. ALGORITMOS GENÉTICOS

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 forma débil e imperfecta, cierta capacidad de solución del problema.

Estas candidatas prometedoras se conservan y se les permite reproducirse. Se realizan múltiples copias de ellas, pero estas copias no son perfectas, sino que se les introducen algunos cambios aleatorios durante el proceso de copia, a modo de las mutaciones que pueden sufrir los descendientes de una población. Luego, esta descendencia digital prosigue con la siguiente generación, formando un nuevo conjunto de soluciones candidatas, y son de nuevo sometidas a una ronda de evaluación de aptitud. Las candidatas que han empeorado o no han mejorado con los cambios en su código son eliminadas de nuevo; pero, de nuevo, por puro azar, las variaciones aleatorias introducidas en la población pueden haber mejorado a algunos individuos, convirtiéndolos en mejores soluciones del problema, más completas o más eficientes. El proceso se repite las iteraciones que haga falta, hasta que obtengamos soluciones suficientemente buenas para nuestros propósitos.

Aunque a algunos les puede parecer asombroso y anti intuitivo, los algoritmos genéticos han demostrado ser una estrategia enormemente poderosa y exitosa para resolver problemas, demostrando de manera espectacular el poder de los principios evolutivos. Se han utilizado algoritmos genéticos en una amplia variedad de campos para desarrollar soluciones a problemas tan o más difíciles que los abordados por los diseñadores humanos. Además, las soluciones que consiguen son a menudo más eficientes, más elegantes o más complejas que las que un humano produciría.

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.

Un algoritmo genético puede utilizar muchas técnicas diferentes para seleccionar a los individuos que deben copiarse hacia la siguiente generación. A continuación, mostramos una breve explicación de los más habituales (debe tenerse en cuenta que algunos son mutuamente excluyentes, mientras que otros se pueden combinar):

* **Selección elitista**: se garantiza la selección de los miembros más aptos de cada generación. (La mayoría de los AGs no utilizan elitismo puro, sino una forma modificada por la que el individuo mejor, o algunos de los mejores, son copiados hacia la siguiente generación en caso de que no surja nada mejor).
* **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. Este método puede ser útil para seleccionar más tarde, cuando todos los individuos tengan una aptitud relativamente alta y sólo les distingan pequeñas diferencias en la aptitud.
* **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. La ventaja de este método es que puede evitar que individuos muy aptos ganen dominancia al principio a expensas de los menos aptos, lo que reduciría la diversidad genética de la población y podría obstaculizar la búsqueda de una solución aceptable.
* **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 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. La ventaja de este método es que reduce el tiempo total de cálculo al utilizar una evaluación más rápida y menos selectiva para eliminar a la mayoría de los individuos que se muestran poco o nada prometedores, y sometiendo a una evaluación de aptitud más rigurosa y computacionalmente más costosa sólo a los que sobreviven a esta prueba inicial.

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:

1. 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.
2. 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 intercambian los genes que aparecen en el intervalo de 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.
   1. DEEP LEARNING

El aprendizaje profundo, también conocido cono redes neuronales profundas, es un aspecto de la inteligencia artificial (AI) que se ocupa de emular el enfoque de aprendizaje que los seres humanos utilizan para obtener ciertos tipos de conocimiento. En su forma más simple, el aprendizaje profundo puede considerarse como una forma de automatizar el análisis predictivo.

Mientras que los algoritmos tradicionales de aprendizaje automático son lineales, los algoritmos de aprendizaje profundo se apilan en una jerarquía de creciente complejidad y abstracción. Para entender el aprendizaje profundo, imagine a un niño cuya primera palabra es "perro". El niño aprende lo que es (y lo que no es) un perro señalando objetos y diciendo la palabra "perro". El padre dice "Sí, eso es Perro" o "No, eso no es un perro". Mientras el niño continúa apuntando a los objetos, se vuelve más consciente de las características que poseen todos los perros. Lo que el niño hace, sin saberlo, es aclarar una abstracción compleja (el concepto de perro) construyendo una jerarquía en la que cada nivel de abstracción se crea con el conocimiento que se obtuvo de la capa precedente de la jerarquía.

Los programas informáticos que utilizan el aprendizaje profundo pasan por el mismo proceso. Cada algoritmo en la jerarquía aplica una transformación no lineal en su entrada y utiliza lo que aprende para crear un modelo estadístico como salida. Las iteraciones continúan hasta que la salida ha alcanzado un nivel de precisión aceptable. El número de capas de procesamiento a través de las cuales los datos deben pasar es lo que inspiró la etiqueta de profundidad ("deep").

En el aprendizaje tradicional de las máquinas, el proceso de aprendizaje es supervisado y el programador tiene que ser muy, muy específico al decirle a la computadora qué tipos de cosas debe buscar para decidir si una imagen contiene un perro o no contiene un perro. Este es un proceso laborioso llamado extracción de características y la tasa de éxito de la computadora depende totalmente de la capacidad del programador para definir con precisión un conjunto de características para "perro". La ventaja del aprendizaje profundo es que el programa construye el conjunto de características por sí mismo sin supervisión. Esto no es sólo más rápido, sino que por lo general es más preciso.

Inicialmente, el programa de computadora podría ser provisto de datos de entrenamiento, un conjunto de imágenes para las cuales un humano ha etiquetado cada imagen "perro" o "no perro" con meta etiquetas. El programa utiliza la información que recibe de los datos de entrenamiento para crear un conjunto de características para el perro y construir un modelo predictivo. En este caso, el modelo que la computadora crea por primera vez podría predecir que cualquier cosa en una imagen que tenga cuatro patas y una cola debería estar etiquetada como "perro". Por supuesto, el programa no es consciente de las etiquetas "cuatro patas" o "cola", simplemente buscará patrones de píxeles en los datos digitales. Con cada iteración, el modelo predictivo que crea el equipo de cómputo se vuelve más complejo y más preciso.

Debido a que este proceso imita el pensamiento humano, el aprendizaje profundo a veces se conoce como aprendizaje neuronal profundo o redes neuronales profundas. A diferencia del niño pequeño, que tardará semanas o incluso meses en comprender el concepto de "perro", un programa informático que utiliza algoritmos de aprendizaje profundo puede mostrar un conjunto de entrenamiento y ordenarlo a través de millones de imágenes, identificando con precisión qué imágenes tienen perros en tan solo unos minutos.

Con el fin de lograr un nivel aceptable de precisión, los programas de aprendizaje profundo requieren acceso a inmensas cantidades de datos de entrenamiento y poder de procesamiento, ninguno de los cuales estaba fácilmente disponible para los programadores hasta la era de los grandes datos y la computación en nube. Debido a que la programación del aprendizaje profundo es capaz de crear modelos estadísticos complejos directamente a partir de su propia salida iterativa, es capaz de crear modelos predictivos precisos a partir de grandes cantidades de datos no etiquetados y no estructurados.

Esto es importante a medida que el internet de las cosas (IoT) continúa haciéndose más penetrante, porque la mayoría de los datos que los seres humanos y las máquinas crean están desestructurados y no están etiquetados. Los casos de uso de hoy para el aprendizaje profundo incluyen todos los tipos de aplicaciones de análisis de big data, especialmente aquellos enfocados en el procesamiento del lenguaje natural (NLP), traducción de idiomas, diagnóstico médico, señales de comercio bursátil, seguridad de redes e identificación de imágenes.

REFERENCIAS

Referencias en la Web:

[1]

<https://computerhoy.com/reportajes/tecnologia/inteligencia-artificial-469917>

[2]

<http://www2.ulpgc.es/hege/almacen/download/38/38584/practica_ia_2.pdf>

[3]

<https://www.uaeh.edu.mx/docencia/P_Presentaciones/icbi/asignatura/logicaDifusa.pdf>

[4]

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

[5]

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

[6]

<https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-profundo-deep-learning>