

Apuntes de Inteligencia Artificial

Eduardo Alcaraz
ealcaraz@itmorelia.edu.mx

April 23, 2017

Contents

1 Inteligencia Artificial

La inteligencia artificial (IA) es un área multidisciplinaria que, a través de ciencias como las ciencias de la computación, la lógica y la filosofía, estudia la creación y diseño de entidades capaces de resolver cuestiones por sí mismas utilizando como paradigma la inteligencia humana .[?]

General y amplio como eso, reúne a amplios campos, los cuales tienen en común la creación de máquinas capaces de pensar. En ciencias de la computación se denomina inteligencia artificial a la capacidad de razonar de un agente no vivo.^{1 2 3} John McCarthy acuñó la expresión **inteligencia artificial** en 1956, y la definió así: {Es la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cómputo inteligentes}.

- Búsqueda del estado requerido en el conjunto de los estados producidos por las acciones posibles.
- Algoritmos genéticos (análogo al proceso de evolución de las cadenas de ADN).
- Redes neuronales artificiales (análogo al funcionamiento físico del cerebro de animales y humanos).
- Razonamiento mediante una lógica formal análogo al pensamiento abstracto humano.

También existen distintos tipos de percepciones y acciones, que pueden ser obtenidas y producidas, respectivamente, por sensores físicos y sensores

mecánicos en máquinas, pulsos eléctricos u ópticos en computadoras, tanto como por entradas y salidas de bits de un software y su entorno software. Varios ejemplos se encuentran en el área de control de sistemas, planificación automática, la habilidad de responder a diagnósticos y a consultas de los consumidores, reconocimiento de escritura, reconocimiento del habla y reconocimiento de patrones. Los sistemas de IA actualmente son parte de la rutina en campos como economía, medicina, ingeniería y la milicia, y se ha usado en gran variedad de aplicaciones de software, juegos de estrategia, como ajedrez de computador, y otros videojuegos.

2 Categorías de la Inteligencia Artificial

- Sistemas que piensan como humanos. Estos sistemas tratan de emular el pensamiento humano; por ejemplo las redes neuronales artificiales. La automatización de actividades que vinculamos con procesos de pensamiento humano, actividades como la Toma de decisiones, Resolución de problemas y aprendizaje.
- Sistemas que actúan como humanos.- Estos sistemas tratan de actuar como humanos; es decir, imitan el comportamiento humano; por ejemplo la robótica. El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor.⁷
- Sistemas que piensan racionalmente.- Es decir, con lógica (idealmente), tratan de imitar o emular el pensamiento lógico racional del ser humano; por ejemplo los sistemas expertos. El estudio de los cálculos que hacen posible percibir, razonar y actuar.⁸
- Sistemas que actúan racionalmente (idealmente).- Tratan de emular de forma racional el comportamiento humano; por ejemplo los agentes inteligentes. Está relacionado con conductas inteligentes en artefactos.⁹

3 Inteligencia artificial convencional

Se conoce también como IA simbólico-deductiva. Está basada en el análisis formal y estadístico del comportamiento humano ante diferentes problemas:

- Razonamiento basado en casos: Ayuda a tomar decisiones mientras se resuelven ciertos problemas concretos y, aparte de que son muy importantes, requieren de un buen funcionamiento.

- **Sistemas expertos:** Infieren una solución a través del conocimiento previo del contexto en que se aplica y ocupa de ciertas reglas o relaciones.
- **Redes bayesianas:** Propone soluciones mediante inferencia probabilística.
- **Inteligencia artificial basada en comportamientos:** Esta inteligencia contiene autonomía y puede auto-regularse y controlarse para mejorar.
- **Smart process management:** Facilita la toma de decisiones complejas, proponiendo una solución a un determinado problema al igual que lo haría un especialista en la dicha actividad.

4 Sistemas expertos

Los sistemas expertos son llamados así porque emulan el razonamiento de un experto en un dominio concreto, y en ocasiones son usados por éstos. Con los sistemas expertos se busca una mejor calidad y rapidez en las respuestas, dando así lugar a una mejora de la productividad del propio experto al usar este tipo de sistemas informáticos.

Es una aplicación informática capaz de solucionar un conjunto de problemas que exigen un gran conocimiento sobre un determinado tema. Un sistema experto es un conjunto de programas que, sobre una base de conocimientos, posee información de uno o más expertos en un área específica. Se puede entender como una rama de la inteligencia artificial, donde el poder de resolución de un problema en un programa de computadora viene del conocimiento de un dominio específico. Estos sistemas imitan las actividades de un humano para resolver problemas de distinta índole (no necesariamente tiene que ser de inteligencia artificial). También se dice que un SE se basa en el conocimiento declarativo (hechos sobre objetos, situaciones) y el conocimiento de control (información sobre el seguimiento de una acción).

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

- **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.
- **Adquisición de nuevos conocimientos o integrador del sistema:** son mecanismos de razonamiento que sirven para modificar los conocimientos anteriores. Sobre la base de lo anterior se puede decir que los sistemas expertos son el producto de investigaciones en el campo de la inteligencia

artificial ya que ésta no intenta sustituir a los expertos humanos, sino que se desea ayudarlos a realizar con más rapidez y eficacia todas las tareas que realiza. Debido a esto en la actualidad se están mezclando diferentes técnicas o aplicaciones aprovechando las ventajas que cada una de estas ofrece para poder tener empresas más seguras. Un ejemplo de estas técnicas sería los agentes que tienen la capacidad de negociar y navegar a través de recursos en línea; y es por eso que en la actualidad juega un papel preponderante en los sistemas expertos.

Estructura básica de un SE Un Sistema Experto está conformado por:

- Especialistas Humanos
- Ingenieros en Conocimientos.
- Base de conocimientos (BC): Contiene conocimiento modelado extraído del diálogo con un experto.
- Base de hechos (Memoria de trabajo): contiene los hechos sobre un problema que se ha descubierto durante el análisis.
- Motor de inferencia: Modela el proceso de razonamiento humano.
- Módulos de justificación: Explica el razonamiento utilizado por el sistema para llegar a una determinada conclusión.
- oInterfaz de usuario: es la interacción entre el SE y el usuario, y se realiza mediante el lenguaje natural.

5 Tipos de SE

Principalmente existen tres tipos de sistemas expertos:

- Basados en reglas previamente establecidas.
- Basados en casos o CBR (Case Based Reasoning).
- Basados en 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.
- Aplicando redes bayesianas, basadas en estadística y el teorema de Bayes.

6 Ventajas y limitaciones de los Sistemas Expertos

6.1 Ventajas

- Permanencia: A diferencia de un experto humano un SE (sistema experto) no envejece, y por tanto no sufre pérdida de facultades con el paso del tiempo.
- Replicación: Una vez programado un SE lo podemos replicar infinitas veces.
- Rapidez: Un SE puede obtener información de una base de datos y realizar cálculos numéricos mucho más rápido que cualquier ser humano.
- Bajo costo: A pesar de que el costo inicial pueda ser elevado, gracias a la capacidad de duplicación el coste finalmente es bajo.
- Entornos peligrosos: Un SE puede trabajar en entornos peligrosos o dañinos para el ser humano.
- Fiabilidad: Los SE no se ven afectados por condiciones externas, un humano sí (cansancio, presión, etc.).
- Consolidar varios conocimientos.
- Apoyo Académico.

6.2 Limitaciones

- Sentido común: Para un Sistema Experto no hay nada obvio. Por ejemplo, un sistema experto sobre medicina podría admitir que un hombre lleva 40 meses embarazado, a no ser que se especifique que esto no es posible ya que un hombre no puede gestar hijos.
- Lenguaje natural: Con un experto humano podemos mantener una conversación informal mientras que con un SE no podemos.
- Capacidad de aprendizaje: Cualquier persona aprende con relativa facilidad de sus errores y de errores ajenos, que un SE haga esto es muy complicado.
- Perspectiva global: Un experto humano es capaz de distinguir cuales son las cuestiones relevantes de un problema y separarlas de cuestiones secundarias.
- Capacidad sensorial: Un SE carece de sentidos.
- Flexibilidad: Un humano es sumamente flexible a la hora de aceptar datos para la resolución de un problema.
- Conocimiento no estructurado: Un SE no es capaz de manejar conocimiento poco estructurado.

7 Ejemplos importantes

- Dendral
- XCon
- Dipmeter Advisor
- Mycin
- CADUCEUS
- R1
- CLIPS, Jess
- Prolog

8 Tareas que realiza un Sistema Experto

8.1 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. En el campo del mantenimiento predictivo los Sistemas Expertos se utilizan fundamentalmente como herramientas de diagnóstico. 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. Así, usando un modelo computacional del razonamiento de un experto humano, proporciona los mismos resultados que alcanzaría dicho experto.

8.2 Diseño

Diseño es el proceso de especificar una descripción de un artefacto que satisfice varias características desde un número de fuentes de conocimiento.

El diseño se concibe de distintas formas:

- El diseño en ingeniería es el uso de principios científicos, información técnica e imaginación en la definición de una estructura mecánica, máquina o sistema que ejecute funciones específicas con el máximo de economía y eficiencia.
- El diseño industrial busca rectificar las omisiones de la ingeniería, es un intento consciente de traer forma y orden visual a la ingeniería de hardware donde la tecnología no provee estas características.

Los SE en diseño ven este proceso como un problema de búsqueda de una solución óptima o adecuada. Las soluciones alternas pueden ser conocidas de antemano o se pueden generar automáticamente probándose distintos diseños para verificar cuáles de ellos cumplen los requerimientos solicitados por el usuario, ésta técnica es llamada “generación y prueba”, por lo tanto estos SE son llamados de selección. En áreas de aplicación, la prueba se termina cuando se encuentra la primera solución; sin embargo, existen problemas más complejos en los que el objetivo es encontrar la solución óptima.

8.3 Planificación

La planificación es la realización de planes o secuencias de acciones y es un caso particular de la simulación. Está compuesto por un simulador y un sistema de control. El efecto final es la ordenación de un conjunto de acciones con el fin de conseguir un objetivo global.

Los problemas que presentan la planificación mediante SE son los siguientes:

- Existen consecuencias no previsibles, de forma que hay que explorar y explicar varios planes.
- Existen muchas consideraciones que deben ser valoradas o incluirles un factor de peso.
- Suelen existir interacciones entre planes de subobjetivos diversos, por lo que deben elegirse soluciones de compromiso.
- Trabajo frecuente con incertidumbre, pues la mayoría de los datos con los que se trabaja son más o menos probables pero no seguros.
- cEs necesario hacer uso de fuentes diversas tales como bases de datos.

8.4 Control

Un sistema de control participa en la realización de las tareas de interpretación, diagnóstico y reparación de forma secuencial. Con ello se consigue conducir o guiar un proceso o sistema. Los sistemas de control son complejos debido al número de funciones que deben manejar y el gran número de factores que deben considerar; esta complejidad creciente es otra de las razones que apuntan al uso del conocimiento, y por tanto de los SE.

Cabe aclarar que los sistemas de control pueden ser en lazo abierto, si en el mismo la realimentación o el paso de un proceso a otro lo realiza el operador, o en lazo cerrado si no tiene que intervenir el operador en ninguna parte del mismo. Reparación, correcta o terapia.

La reparación, corrección, terapia o tratamiento consiste en la proposición de las acciones correctoras necesarias para la resolución de un problema. Los SE en reparación tienen que cumplir diversos objetivos, como son: Reparación lo más rápida y económicamente posible. Orden de las reparaciones cuando hay que realizar varias. Evitar los efectos secundarios de la reparación, es decir la aparición de nuevas averías por la reparación.

8.5 Simulación

La simulación es una técnica que consistente en 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. Las técnicas tradicionales de simulación requieren modelos matemáticos y lógicos, que describen el comportamiento del sistema bajo estudio.

El empleo de los SE para la simulación viene motivado por la principal característica de los SE, que es su capacidad para la simulación del razonamiento de un experto humano, que es un proceso complejo.

En la aplicación de los SE para simulación hay que diferenciar cinco configuraciones posibles:

1. Un SE puede disponer de un simulador con el fin de comprobar las soluciones y en su caso rectificar el proceso que sigue.
2. Un sistema de simulación puede contener como parte del mismo a un SE y por lo tanto el SE no tiene que ser necesariamente de simulación.
3. Un SE puede controlar un proceso de simulación, es decir que el modelo está en la base de conocimiento del SE y su evolución es función de la base de hechos, la base de conocimientos y el motor de inferencia, y no de un conjunto de ecuaciones aritmético – lógicas.
4. Un SE puede utilizarse como consejero del usuario y del sistema de simulación.
5. Un SE puede utilizarse como máscara o sistema frontal de un simulador con el fin de que el usuario reciba explicación y justificación de los procesos.

8.6 Instrucción

Un sistema de instrucción realizara un seguimiento del proceso de aprendizaje. El sistema detecta errores ya sea de una persona con conocimientos e identifica el remedio adecuado, es decir, desarrolla un plan de enseñanza que facilita el proceso de aprendizaje y la corrección de errores.

8.7 Recuperación de información

Los Sistemas Expertos, con su capacidad para combinar información y reglas de actuación, han sido vistos como una de las posibles soluciones al tratamiento y recuperación de información, no sólo documental. La década de 1980 fue prolija en investigación y publicaciones sobre experimentos de este orden, interés que continua en la actualidad.

Lo que diferencia a estos sistemas de un sistema tradicional de recuperación de información es que éstos últimos sólo son capaces de recuperar lo que existe explícitamente, mientras que un Sistema Experto debe ser capaz de generar información no explícita, razonando con los elementos que se le dan. Pero la capacidad de los SE en el ámbito de la recuperación de la información no se limita a la recuperación. Pueden utilizarse para ayudar al usuario, en selección de recursos de información, en filtrado de respuestas, etc. Un SE puede actuar como un intermediario inteligente que guía y apoya el trabajo del usuario final.

9 Espacio de estados

Muchos de los problemas que pueden ser resueltos aplicando técnicas de inteligencia artificial se modelan en forma simbólica y discreta definiendo las configuraciones posibles del universo estudiado. El problema se plantea entonces en términos de encontrar una configuración objetivo a partir de una configuración inicial dada, aplicando transformaciones válidas según el modelo del universo. La respuesta es la secuencia de transformaciones cuya aplicación sucesiva lleva a la configuración deseada.

Los ejemplos más característicos de esta categoría de problemas son los juegos (son universos restringidos fáciles de modelar). En un juego, las configuraciones del universo corresponden directamente a las configuraciones del tablero. Cada configuración es un estado que puede ser esquematizado gráficamente y representado en forma simbólica. Las transformaciones permitidas corresponden a las reglas o movidas del juego, formalizadas como transiciones de estado.

Entonces, para plantear formalmente un problema, se requiere precisar una representación simbólica de los estados y definir reglas del tipo condición acción para cada una de las transiciones válidas dentro del universo modelado. La acción de una regla indica como modificar el estado actual para generar un nuevo estado. La condición impone restricciones sobre la aplicabilidad de la regla según el estado actual, el estado generado o la historia completa del proceso de solución.

El espacio de estados de un juego es un grafo cuyos nodos representan las configuraciones alcanzables (los estados válidos) y cuyos arcos explicitan las movidas posibles (las transiciones de estado). En principio, se puede construir cualquier espacio de estados partiendo del estado inicial, aplicando cada una de las reglas para generar los sucesores inmediatos, y así sucesivamente con cada uno de los nuevos estados generados (en la práctica, los espacios de estados suelen ser demasiado grandes para explicitarlos por completo).

Cuando un problema se puede representar mediante un espacio de estados, la solución computacional corresponde a encontrar un camino desde el estado inicial a un estado objetivo.

10 Ejemplo de espacio de estados

10.1 Descripción del problema

Un arriero se encuentra en el borde de un río llevando un puma, una cabra y una lechuga. Debe cruzar a la otra orilla por medio de un bote con capacidad para dos (el arriero y alguna de sus pertenencias). La dificultad es que si el puma se queda solo con la cabra la devorará, y lo mismo sucederá si la cabra se queda sola con la lechuga. ¿Cómo cruzar sin perder ninguna pertenencia?

10.2 Representación de las configuraciones del universo del problema:

Basta precisar la situación antes o después de cruzar. El arriero y cada una de sus pertenencias tienen que estar en alguna de las dos orillas. La representación del estado debe entonces indicar en que lado se encuentra cada uno de ellos. Para esto se puede utilizar un término simbólico con la siguiente sintaxis: estado(A,P,C,L), en que A, P, C y L son variables que representan, respectivamente, la posición del arriero, el puma, la cabra y la lechuga. Las variables pueden tomar dos valores: i y d, que simbolizan respectivamente el borde izquierdo y el borde derecho del río. Por convención se elige partir en el borde izquierdo. El estado inicial es entonces estado(i,i,i,i). El estado objetivo es estado(d,d,d,d).

10.3 Definición de las reglas de transición:

El arriero tiene cuatro acciones posibles: cruzar solo, cruzar con el puma, cruzar con la cabra y cruzar con la lechuga. Estas acciones están condi-

cionadas a que ambos pasajeros del bote estén en la misma orilla y a que no queden solos el puma con la cabra o la cabra con la lechuga. El estado resultante de una acción se determina intercambiando los valores i y d para los pasajeros del bote.

10.4 Generación del espacio de estados

En este ejemplo se puede explicitar todo el espacio de estados (el número de configuraciones está acotado por 24).

estado	movidas			
	cruza solo	con puma	con cabra	con lechuga
<i>estado(i,i,i,i)</i>	problema	problema	<i>estado(d,i,d,i)</i>	problema
<i>estado(d,i,d,i)</i>	<i>estado(i,i,d,i)</i>	imposible	<i>estado(i,i,i,i)</i>	imposible
<i>estado(i,i,d,i)</i>	<i>estado(d,i,d,i)</i>	<i>estado(d,d,d,i)</i>	imposible	<i>estado(d,i,d,d)</i>
<i>estado(d,d,d,i)</i>	problema	<i>estado(i,i,d,i)</i>	<i>estado(i,d,i,i)</i>	imposible
<i>estado(d,i,d,d)</i>	problema	imposible	<i>estado(i,i,i,d)</i>	<i>estado(i,i,d,i)</i>
<i>estado(i,d,i,i)</i>	problema	imposible	<i>estado(d,d,d,i)</i>	<i>estado(d,d,i,d)</i>
<i>estado(i,i,i,d)</i>	problema	<i>estado(d,d,i,d)</i>	<i>estado(d,i,d,d)</i>	imposible
<i>estado(d,d,i,d)</i>	<i>estado(i,d,i,d)</i>	<i>estado(i,i,i,d)</i>	imposible	<i>estado(i,d,i,i)</i>
<i>estado(i,d,i,d)</i>	<i>estado(d,d,i,d)</i>	imposible	<i>estado(d,d,d,d)</i>	imposible
<i>estado(d,d,d,d)</i>	problema	problema	<i>estado(i,d,i,d)</i>	problema

Figure 1: Espacio de Estados

10.5 Problemas de los Canibales y Monjes

Se tienen 3 monjes y 3 caníbales en el margen Oeste de un río. Existe una canoa con capacidad para dos personas como máximo. Se desea que los seis pasen al margen Este del río, pero hay que considerar que no debe haber más caníbales que monjes en ningún sitio porque entonces los caníbales se comen a los monjes. Además, la canoa siempre debe ser conducida por alguien.

El espacio de estados está definido por:

$\{(Mo, Co, Me, Ce, C) \mid Mo \text{ es el número de monjes en el margen oeste con } 0 \leq Mo \leq 3$
AND $Co \text{ es el número de caníbales en el margen oeste con } 0 \leq Co \leq 3 \text{ AND}$

($Co \leq Mo$ OR $Mo=0$)

AND Me es el número de monjes en el margen este con $0 \leq Me \leq 3$

AND Ce es el número de caníbales en el margen este con $0 \leq Ce \leq 3$ AND
($Ce \leq Me$ OR $Me=0$) AND $Co+Ce=3$ AND $Mo+Me=3$ AND $C = [E|O]$ es
el margen dónde está la canoa}

El estado inicial es (3,3,0,0,O)

El estado final es (0,0,3,3,E)

Las reglas que se pueden aplicar son:

1. Viajan un monje y un caníbal de O a E: Si (Mo, Co, Me, Ce, O) AND $Mo \geq 1$ AND $Co \geq 1$ AND $Ce+1 \leq Me+1 \Rightarrow (Mo-1, Co-1, Me+1, Ce+1, E)$
2. Viajan un monje y un caníbal de E a O: Si (Mo, Co, Me, Ce, E) AND $Me \geq 1$ AND $Ce \geq 1$ AND $Co+1 \leq Mo+1 \Rightarrow (Mo+1, Co+1, Me-1, Ce-1, O)$
3. Viajan dos monjes de O a E: Si (Mo, Co, Me, Ce, O) AND $Mo \geq 2$ AND ($Mo-2=0$ OR $Co \leq Mo-2$) AND $Ce \leq Me+2 \Rightarrow (Mo-2, Co, Me+2, Ce, E)$
4. Viajan dos monjes de E a O: Si (Mo, Co, Me, Ce, E) AND $Me \geq 2$ AND ($Me-2=0$ OR $Ce \leq Me-2$) AND $Co \leq Mo+2 \Rightarrow (Mo+2, Co, Me-2, Ce, O)$
5. Viajan dos caníbales de O a E: Si (Mo, Co, Me, Ce, O) AND $Co \geq 2$ AND ($Me=0$ OR $Ce+2 \leq Me$) $\Rightarrow (Mo, Co-2, Me, Ce+2, E)$
6. Viajan dos caníbales de E a O: Si (Mo, Co, Me, Ce, E) AND $Ce \geq 2$ AND ($Mo=0$ OR $Co+2 \leq Mo$) $\Rightarrow (Mo, Co+2, Me, Ce-2, O)$
7. Viaja un monje de O a E: Si (Mo, Co, Me, Ce, O) AND $Mo \geq 1$ AND ($Mo-1=0$ OR $Co \leq Mo-1$) AND $Ce \leq Me+1 \Rightarrow (Mo-1, Co, Me+1, Ce, E)$
8. Viaja un monje de E a O: Si (Mo, Co, Me, Ce, E) AND $Me \geq 1$ AND ($Me-1=0$ OR $Ce \leq Me-1$) AND $Co \leq Mo+1 \Rightarrow (Mo+1, Co, Me-1, Ce, O)$
9. Viaja un caníbal de O a E: Si (Mo, Co, Me, Ce, O) AND $Co \geq 1$ AND ($Me=0$ OR $Ce+1 \leq Me$) $\Rightarrow (Mo, Co-1, Me, Ce+1, E)$
10. Viaja un caníbal de E a O: Si (Mo, Co, Me, Ce, E) AND $Ce \geq 1$ AND ($Mo=0$ OR $Co+1 \leq Mo$) $\Rightarrow (Mo, Co+1, Me, Ce-1, E)$

Nota: En referencia a la regla 3 la condición $Ce \leq Me+2$ puede intuirse como redundante. Esta condición no se cumple sólo en el caso $Ce=3$ y $Me=0$.
Pese a

que es un estado que pertenece al espacio de estados válidos, podemos intuir que nunca se llega a tener 3 caníbales y ningún monje del lado Este y la barca del lado Oeste. De todas maneras sólo se puede eliminar si podemos demostrar formalmente la imposibilidad de esta situación.

Un pasaje de estados para ir de $(3,3,0,0,O)$ a $(0,0,3,3,E)$ es el siguiente:

$(3,3,0,0,O) \Rightarrow (3,1,0,2,E) \Rightarrow (3,2,0,1,O) \Rightarrow (3,0,0,3,E) \Rightarrow (3,1,0,2,O)$
 $\Rightarrow (1,1,2,2,E) \Rightarrow (2,2,1,1,O) \Rightarrow (0,2,3,1,E) \Rightarrow (0,3,3,0,O) \Rightarrow (0,1,3,2,E)$
 $\Rightarrow (0,2,3,1,O) \Rightarrow (0,0,3,3,E)$

11 Representación de espacio de estados

La primera pregunta es, como

12 El problema del n-Puzzle

12.1 Caracterización de las búsquedas ciegas.

La búsqueda ciega o no informada sólo utiliza información acerca de si un estado es o no objetivo para guiar su proceso de búsqueda.

Los métodos de búsqueda ciega se pueden clasificar en dos grupos básicos:

- **Métodos de búsqueda en anchura:**

Son procedimientos de búsqueda nivel a nivel. Para cada uno de los nodos de un nivel se aplican todos los posibles operadores y no se expande ningún nodo de un nivel antes de haber expandido todos los del nivel anterior.

- **Métodos de búsqueda en profundidad:**

En estos procedimientos se realiza la búsqueda por una sola rama del árbol hasta encontrar una solución o hasta que se tome la decisión de terminar la búsqueda por esa dirección (por no haber posibles operadores que aplicar sobre el nodo hoja o por haber alcanzado un nivel de profundidad muy grande) . Si esto ocurre se produce una vuelta atrás (backtracking) y se sigue por otra rama hasta visitar todas las ramas del árbol si es necesario.

A partir de los dos tipos de búsqueda anteriores surgió uno nuevo, llamado método de búsqueda por profundización iterativa. El algoritmo de búsqueda más representativo de esta nueva tendencia es el DFID acrónimo de su nombre en inglés (Depth-First Iterative-Deepening).

12.2 Caracterización de las búsquedas heurísticas.

Las técnicas de búsqueda heurística se apoyan al contrario de los métodos de búsqueda ciega se apoyan en información adicional para realizar su proceso de búsqueda. Para mejorar la eficiencia de la búsqueda, estos algoritmos hacen uso de una función que realiza una predicción del coste necesario para alcanzar la solución. La función que guía el proceso toma el nombre de función heurística.

De todos los algoritmos de búsqueda heurística, uno destaca en especial: el A*. Este algoritmo, a pesar de haber sido creado entorno a los años 60, sigue en la actualidad siendo uno de los mas utilizados. Desafortunadamente, es ineficiente en cuanto al uso de memoria durante el proceso de búsqueda. Por ello, en las décadas de los 80 y 90, aparecieron algoritmos basados en el propio A*, pero que limitaban el uso de memoria. Dos de los algoritmos más representativos de esta última tendencia son el IDA* (Iterative-Deepening A*) y el SMA* (Simplified Memory-bounded A*).

13 Técnicas de Búsqueda

13.1 Solución de problemas con búsqueda.

La solución de problemas es fundamental para la mayoría de las aplicaciones de IA; existen principalmente dos clases de problemas que se pueden resolver mediante procesos computables: aquéllos en los que se utiliza un algoritmo determinista que garantiza la solución al problema y las tareas complejas que se resuelven con la búsqueda de una solución; de ésta última clase de problemas se ocupa la IA.

La solución de problemas requiere dos consideraciones:\

0.5cm]

- Representación del problema en un espacio organizado.
- La capacidad de probar la existencia del estado objetivo en dicho espacio.

Las anteriores premisas se traducen en: la determinación del estado objetivo y la determinación del camino óptimo guiado por este objetivo a través de una o más transiciones dado un estado inicial

El espacio de búsqueda, se le conoce como una colección de estados. En general los espacios de búsqueda en los problemas de IA no son completamente conocidos de forma a priori. De lo anterior ‘resolver un problema de IA’ cuenta con dos fases:\

0.5cm]

- La generación del espacio de estados
- La búsqueda del estado deseado en ese espacio.

Debido a que "todo el espacio de búsqueda" de un problema es muy grande, puede causar un bloqueo de memoria, dejando muy poco espacio para el proceso de búsqueda. Para solucionar esto, se expande el espacio paso a paso, hasta encontrar el estado objetivo.

13.2 Espacios de Estados

Muchos de los problemas que pueden ser resueltos aplicando técnicas de inteligencia artificial se modelan en forma simbólica y discreta definiendo las configuraciones posibles del universo estudiado. El problema se plantea entonces en términos de encontrar una configuración objetivo a partir de una configuración inicial dada, aplicando transformaciones válidas según el modelo del universo. La respuesta es la secuencia de transformaciones cuya aplicación sucesiva lleva a la configuración deseada. Los ejemplos más característicos de esta categoría de problemas son los juegos (son universos restringidos fáciles de modelar). En un juego, las configuraciones del universo corresponden directamente a las configuraciones del tablero. Cada configuración es un estado que puede ser esquematizado gráficamente y representado en forma simbólica. Las transformaciones permitidas corresponden a las reglas o movidas del juego, formalizadas como transiciones de estado. Entonces, para plantear formalmente un problema, se requiere precisar una representación simbólica de los estados y definir reglas del tipo condición acción para cada una de las transiciones válidas dentro del universo modelado. La acción de una regla indica como modificar el estado actual para generar un nuevo estado. La condición impone restricciones sobre la aplicabilidad de la regla según el estado actual, el estado generado o la historia completa del proceso de solución. El espacio de estados de un juego es un grafo cuyos nodos representan las configuraciones alcanzables (los estados válidos) y cuyos arcos explicitan las movidas posibles (las transiciones de estado). En principio, se puede construir cualquier espacio de estados partiendo del estado inicial, aplicando cada una de las reglas para generar los sucesores inmediatos, y así sucesivamente con cada uno de los nuevos estados generados (en la práctica, los espacios de estados suelen ser demasiado grandes para explicitarlos por completo). Cuando un problema se puede representar mediante un espacio de estados, la solución computacional corresponde a encontrar un camino desde el estado inicial a un estado objetivo.

13.2.1 Determinísticos

El espacio de estados determinísticos contienen un único estado inicial y seguir la secuencia de estados para la solución. Los espacios de estados determinísticos son usados por los sistemas expertos. Se puede describir asu vez, que un sistema es determinístico si, para un estado dado, al menos aplica una regla a él y de solo una manera.

13.2.2 No Determinísticos

El no determinístico contiene un amplio número de estados iniciales y sigue la secuencia de estados perteneciente al estado inicial del espacio. Son usados por sistemas de lógica difusa. En otras palabras, si más de una regla aplica a cualquier estado particular del sistema, o si una regla aplica a un estado particular del sistema en más de una manera, entonces el sistema es no determinístico.

13.3 Métodos de Búsqueda

13.3.1 Primero en anchura (breadthfirst)

En inglés, breadth-first search. Si el conjunto open se maneja como una lista FIFO, es decir, como una cola, siempre se estará visitando primero los primeros estados en ser generados. El recorrido del espacio de estados se hace por niveles de profundidad.

```
procedure Busqueda_en_amplitud {
  open () [estado_inicial]
  closed () {}
  while (open no esta vacia) {
    remover el primer estado X de la lista open
    if (X es un estado objetivo) return exito
    else {
      generar el conjunto de sucesores del estado X
      agregar el estado X al conjunto closed
      eliminar sucesores que ya estan en open o en closed
      agregar el resto de los sucesores al final de open
    }
  }
  return fracaso
}
```

Si el factor de ramificación es B y la profundidad a la cual se encuentra el estado objetivo más cercano es n , este algoritmo tiene una complejidad en tiempo y espacio de $O(B^n)$. Contrariamente a la búsqueda en profundidad, la búsqueda en amplitud garantiza encontrar el camino más corto.

13.3.2 Primero en profundidad (depthfirst).

En inglés, depth-first search. Si el conjunto open se maneja como una lista LIFO, es decir, como un stack, siempre se estará visitando primero los últimos estados en ser generados. Esto significa que si A genera B y C , y B genera D , antes de visitar C se visita D , que está más alejado de la raíz A , o sea más profundo en el árbol de búsqueda. El algoritmo tiene en este caso la tendencia de profundizar la búsqueda en una rama antes de explorar ramas alternativas.

```

procedure Busqueda_en_profundidad {
  open () [estado_inicial]
  closed () {}
  while (open no esta vacia) {
    remover el primer estado X de la lista open
    if (X es un estado objetivo) return exito
    else {
      generar el conjunto de sucesores del estado X
      agregar el estado X al conjunto closed
      eliminar sucesores que ya estan en open o en closed
      agregar el resto de los sucesores al principio de open
    }
  }
  return fracaso
}

```

Considerando que la cantidad promedio de sucesores de los nodos visitados es B (llamado en inglés el branching factor y en castellano el factor de ramificación), y suponiendo que la profundidad máxima alcanzada es n , este algoritmo tiene una complejidad en tiempo de $O(B^n)$ y, si no se considera el conjunto closed, una complejidad en espacio de $O(B \times n)$. En vez de usar el conjunto closed, el control de ciclos se puede hacer descartando aquellos estados que aparecen en el camino generado hasta el momento (basta que cada estado generado tenga un puntero a su padre). El mayor problema de este algoritmo es que puede "perdersse" en una rama sin encontrar la solu-

ción. Además, si se encuentra una solución no se puede garantizar que sea el camino más corto.

14 Satisfacción de restricciones.

Los problemas pueden resolverse buscando en un espacio de estados, estos estados pueden evaluarse por heurísticas específicas para el dominio y probados para verificar si son estados meta. Los componentes del estado, son equivalentes a un grafo de restricciones, los cuales están compuestos de:

0.5cm]

Variables. Dominios (valores posibles para las variables).

Restricciones (binarias) entre las variables.

Objetivo: encontrar un estado (una asignación completa de valores a las variables) Que satisface las restricciones.

En los Problemas de Satisfacción de Restricciones (PSR), los estados y la prueba de meta siguen a una representación estándar, estructurada y muy simple.

Ejemplos:

0.5cm]

- Crucigramas
- Colorear mapas

15 Teoría de juegos.

Siendo una de las principales capacidades de la inteligencia humana su capacidad para resolver problemas, así como la habilidad para analizar los elementos esenciales de cada problema, abstrayéndolos, el identificar las acciones que son necesarias para resolverlos y el determinar cuál es la estrategia más acertada para atacarlos, son rasgos fundamentales.

Podemos definir la resolución de problemas como el proceso que partiendo de unos datos iniciales y utilizando un conjunto de procedimientos escogidos, es capaz de determinar el conjunto de pasos o elementos que nos llevan a lo que denominaremos una solución óptima o semi-óptima de un problema de planificación, descubrir una estrategia ganadora de un juego, demostrar un teorema, reconocer

Una imagen, comprender una oración o un texto son algunas de las tareas que pueden concebirse como de resolución.

Una gran ventaja que nos proporciona la utilización de los juegos es que a través de ellos es muy fácil medir el éxito o el fracaso, por lo que podemos comprobar si las técnicas y algoritmos empleados son los óptimos. En comparación con otras aplicaciones de inteligencia artificial, por ejemplo comprensión del lenguaje, los juegos no necesitan grandes cantidades de algoritmos. Los juegos más utilizados son las damas y el ajedrez.

16 Grafos

Un grafo es un conjunto de puntos (vértices) en el espacio, que están conectados por un conjunto de líneas (aristas). Otros conceptos básicos son: Dos vértices son adyacentes si comparten la misma arista. Los extremos de una arista son los vértices que comparte dicha arista. Un grafo se dice que es finito si su número de vértices es finito.

17 Tipos de grafos

Existen dos tipos de grafos los no dirigidos y los dirigidos.

- No dirigidos: son aquellos en los cuales los lados no están orientados (No son flechas). Cada lado se representa entre paréntesis, separando sus vértices por comas, y teniendo en cuenta $(V_i, V_j) = (V_j, V_i)$.

- Dirigidos: son aquellos en los cuales los lados están orientados (flechas). Cada lado se representa entre ángulos, separando sus vértices por comas y teniendo en cuenta $\langle V_i, V_j \rangle \neq \langle V_j, V_i \rangle$. En grafos dirigidos, para cada lado $\langle A, B \rangle$, A, el cual es el vértice origen, se conoce como la cola del lado y B, el cual es el vértice destino, se conoce como cabeza del lado.

18 Las redes neuronales: qué son y por qué están volviendo

Últimamente, las redes neuronales están volviendo a la actualidad por los logros que están consiguiendo. Por ejemplo, Google ha logrado derrotar a su propio reCAPTCHA con redes neuronales, en Stanford han conseguido generar pies de fotos automáticamente... Metas bastante impresionantes y que cada vez se acercan más a esa idea original de reproducir el funcionamiento del cerebro humano en un ordenador. Ahora bien, ¿en qué consisten estos modelos? ¿Cómo puede imitar un ordenador el proceso de aprendizaje y acabar desarrollando una "cosa" que funciona?.

18.1 ¿Cómo funciona una red neuronal?

A pesar de su nombre, las redes neuronales no tienen un concepto demasiado complicado detrás de ellas. El nombre, como se puede 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 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 son alumnos de una clase en la que el profesor no ha dicho exactamente cómo va a poner las calificaciones. Para empezar, supongamos que sólo hiciste dos exámenes y tienes la evaluación de cada uno de ellos y la final.

¿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 calificación en el segundo examen ha suspendido el

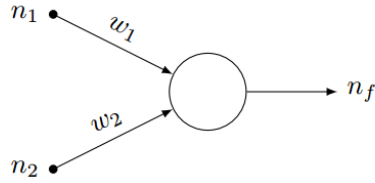


Figure 2: La unidad básica de la red neuronal: el perceptrón. Las entradas son las dos notas, n_1 y n_2 , cada una con su correspondiente peso w_n (lo que hay que encontrar). La salida, n_f , será 1 si está aprobado y 0 si se va a septiembre.

curso, bajaremos el peso del segundo examen porque claramente no influye demasiado. Poco a poco acabaremos encontrando los pesos que se ajusten a las calificaciones 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.

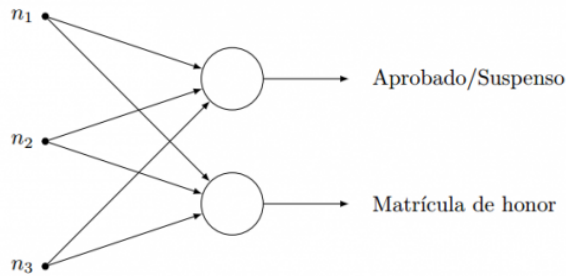


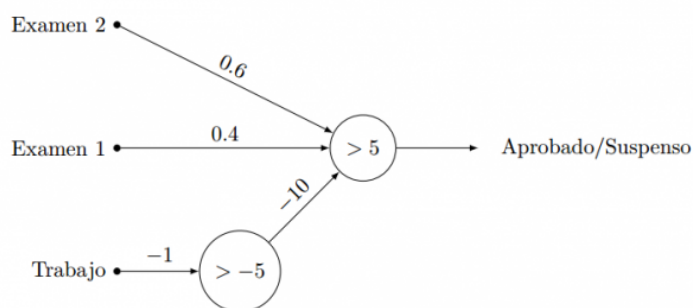
Figure 3: La unidad básica de la red neuronal: el perceptrón. Las entradas son las dos notas, n_1 y n_2 , cada una con su correspondiente peso w_n (lo que hay que encontrar). La salida, n_f , será 1 si está aprobado y 0 si se va a septiembre.

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.

18.2 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 calificaciones 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 suspenso en el curso... Ya puedes 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:



El primer perceptrón mira si la calificación 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.

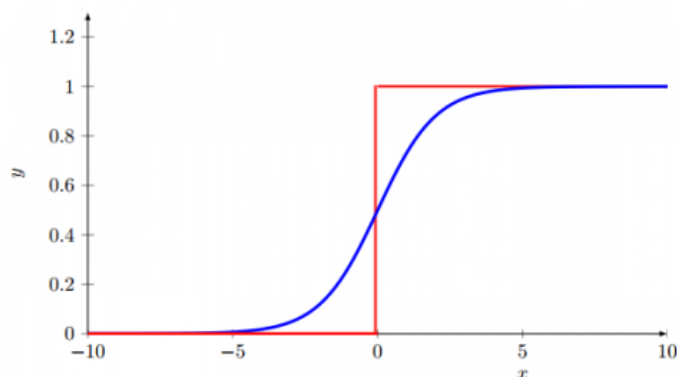


Figure 4: En rojo, la función "escalón". En azul, la sigmoide, una aproximación más suave pero con la misma idea.

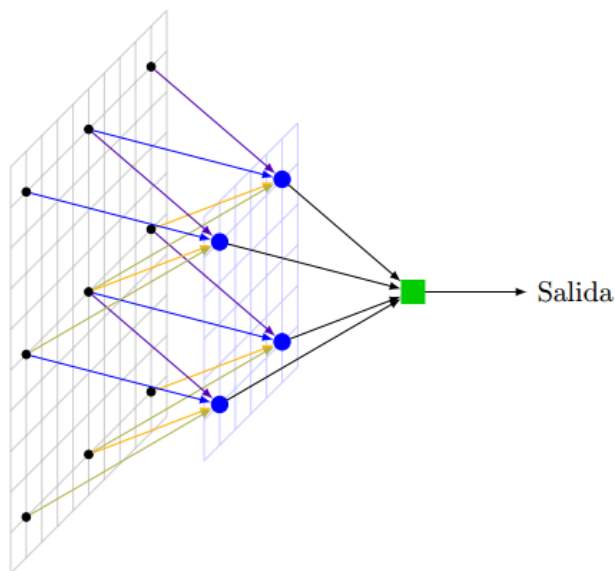
18.3 Más allá de perceptrones: sigmoides, redes profundas y redes convolucionales

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 múltiples 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 píxeles).



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.

18.4 ¿Una nueva época dorada para redes neuronales?

Las redes neuronales no son una idea nueva. Datan de los años 40 y 50, cuando se empezaron a publicar los primeros conceptos. Sin embargo, nunca tuvieron un gran éxito, más que nada porque se necesita una cantidad importante de recursos de un ordenador para entrenar y ejecutar una red neuronal con buenos resultados.

En los últimos años se han conseguido grandes avances gracias a la mejora de los ordenadores y al uso de GPUs para este tipo de computaciones. Hace poco os hablábamos en Xataka de los pies de foto generados por ordenador gracias a una red neuronal de convolución (para el reconocimiento de imagen) junto con una red neuronal recurrente para formar las frases. Los investigadores de Stanford usaron tarjetas GPU para poder entrenar y ejecutar este tipo de redes neuronales en un tiempo razonable.

Algo parecido montó Google con Street View: una red neuronal convolucional que lograba una precisión del 96% a la hora de reconocer números de

calle en las imágenes que toman sus coches. Los de Mountain View están bastante enamorados de las redes neuronales, de hecho: también las usaron para mejorar el reconocimiento de voz de Android o para ahorrar electricidad en sus centros de datos.

Las redes neuronales parece que incluso podrían acabar dominando uno de los juegos que se les resiste a los ordenadores: el juego de Go. En la Universidad de Edimburgo, unos investigadores han logrado usar redes convolucionales para detectar patrones en los tableros y tratar de sacar el mejor movimiento con una efectividad considerable: 90% de juegos ganados contra GNU Go y 10% contra Fuego, dos de los programas que mejor juegan a Go. Aunque pueda parecer poco, hay que tener en cuenta que ambos exploran un buen número de movimientos posibles para ver cuál da más ventaja. La red neuronal sólo mira al estado actual del tablero y emite un veredicto en muchísimo menos tiempo.

Por supuesto, estas redes tampoco son la panacea. A modo de curiosidad, unos investigadores usaron una red neuronal para generar imágenes que engañaban a otra red neuronal diseñada para reconocer objetos. Así, lo que a nosotros nos parece una imagen aleatoria, para la red neuronal es un bikini o un armadillo. Es parte del problema del sobreajuste: redes que se comportan muy bien para los datos de ejemplo o parecidos, pero que con datos muy distintos dan resultados absurdos.

Sea como sea, es un campo muy interesante y que promete bastantes avances a corto plazo sobre todo en reconocimiento de imagen y de sonido.