Inteligencia Artificial

grado en ingeniería informática

Resumen del temario

Autor
Carlos Sánchez Páez

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2017-2018

Índice

Índice de figuras

1. Tema 1. Introducción.

1.1. ¿Qué es la IA?

En la siguiente tabla podemos observar los cuatro enfoques que se han seguido a lo largo de la historia para definir la IA. Los de la izquierda miden el éxito en términos de la fidelidad en la forma de actuar de los humanos, mientras que los de la derecha toman como referencia la *racionalidad*. Un sistema es reacional si hace "lo correcto" en función de su conocimiento.

Sistemas que piensan como humanos	**Sistemas que piensan racionalmente**
Automatización de actividades que vinculamos con procesos de pensamiento humano: toma de decisiones, resolución de problemas, aprendizaje...	Estudio de las facultades mentales mediante el uso de modelos computacionales.
Sistemas que actúan como humanos	**Sistemas que actúan racionalmente**
Lograr que los computadores realicen tareas que, por el momento, los humanos realizamos mejor.	Estudio del diseño de agentes inteligentes.

Figura 1: Definiciones de Inteligencia Artificial

1.1.1. Comportamiento humano. Test de Turing.

El *test de Turing*, propuesto por Alan Turing en 1950 se diseñó para proporcionar una definición operacional y satisfactoria de inteligencia. En vez de proporcionar una lista larga de cualidades necesarias para obtener inteligencia artificialmente, Alan sugirió una prueba basada en la incapacidad de diferenciar entre entidades inteligentes y seres humanos. Un computador supera la prueba si un evaluador humano no es capaz de distinguir si las respuestas a una serie de preguntas son de una persona o no. Hoy por hoy podemos afirmar que para que un computador supere la prueba debe cumplir los siguientes requisitos:

- **Procesamiento de lenguaje natural** que le permita comunicarse satisfactoriamente en inglés.
- **Representación del conocimiento** para almacenar lo que se conoce o siente.
- **Razonamiento automático** para utilizar la información almacenada para responder a preguntas y extraer nuevas conclusiones.
- **Aprendizaje automático** para adaptarse a nuevas circunstancias y detectar y extrapolar patrones.

El test de Turing evita la interacción *física* directa entre el evaluador y el computador. Sin embargo, el *test global de Turing* incluye una señal de vídeo que permite al evaluador valorar la capacidad de percepción del evaluado. También permite al evaluador pasar objetos físicos "a través de una ventana". Para superar esta prueba, el computador debe poseer:

- **Visión computacional** para percibir objetos.
- **Robótica** para manipular y mover objetos.

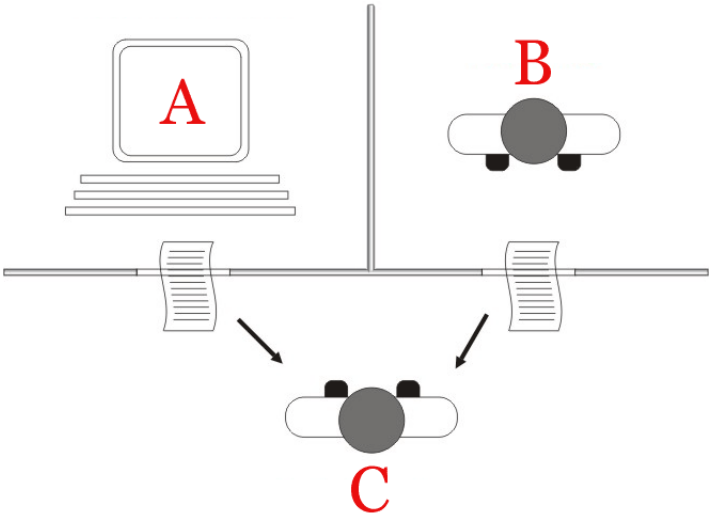

Figura 2: Test de Turing

1.1.2. Pensar como un humano: modelo cognitivo.

Primero tenemos que penetrar en la mente humana para ver cómo funciona. Podemos hacerlo mediante *introspección* (intentando atrapar nuestros propios pensamientos conforme van apareciendo) o mediante experimentos psicológicos. Una vez que contemos con una teoría precisa sobre el funcionamiento de la mente, podremos expresarla en un programa de computador. Si los datos de entrada/salida del programa y los tiempos de realización son parecidos, podremos concluir que algunos de los mecanismos del programa se pueden comparar con los que utilizamos los seres humanos.

La **ciencia cognitiva** es el campo interdisciplinar en el que convergen modelos computacionales de IA y técnicas experimentales de psicología para intentar elaborar teorías precisas y verificables sobre el funcionamiento de la mente humana.

1.1.3. Pensamiento racional: "leyes del pensamiento".

Está basada en la lógica de Aristóteles. Consiste en construir sistemas inteligentes que puedan resolver problemas (resolubles) a partir de un problema descrito en lenguaje lógico.

Sus dos mayores obstáculos son la dificultad de expresar conocimiento informal en lenguaje lógico y las limitaciones físicas de computación (agotamiento de recursos).

1.1.4. Actuar de forma racional: agente racional.

Un **agente** es algo que razona. Pero se espera que los programas informáticos tengan otros atributos que los diferencien de los convencionales, como que estén dotados de controles autónomos, que se adapten a los cambios, etc. Un **agente racional** es aquel que actúa con la intención de alcanzar el mejor resultado o, cuando hay incertidumbre, el mejor resultado esperado.

En el enfoque de la IA según las "leyes del pensamiento", todo el énfasis se pone en hacer inferencias correcctas.

2. Tema 2. Agentes.

Un **agente** es cualquier cosa capaz de percibir su medioambiente con ayuda de sensores y actuar en ese medio utilizando actuadores. Un agente humano tiene ojos, oídos y otros órganos sensoriales además de manos, boca, piernas y otras partes del cuerpor para actuar. Un agente robot recibe pulsaciones del teclado, archivos de ifnormación y paquetes via red a modo de entradas sensoriales y actúa sobre el medio con mensajes en el montior, escribiendo ficheros y enviando paquetes por la red. Se trabaja con la hipótesis de que cada agente puede percibir sus propias acciones (pero no siempres sus efectos).

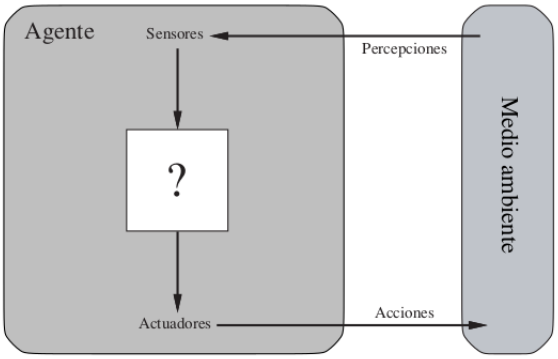

Figura 3: Funcionamiento de un agente.

La **percepción** se utiliza en este contexto para indicar que el agente puede recibir entradas en cualquier instante. La **secuencia de percepciones** de un agente refleja el historial completo de lo que ha recibido. En general, esta secuencia influirá en la decisión que tome el agente en un momento dado. En cierto modo, podemos afirmar que el comportamiento del agente viene dado por la **función del agente** que proyecta una percepción dada una acción.

La **Inteligencia Artificial** es el subcampo de la informática que se dedica a construir agentes que exhiban aspectos del comportamiento inteligente.

Un **agente inteligente** es un sistema de ordenador *situado* en un entorno que es capaz de realizar acciones de forma *autónoma* y que es *flexible* para lograr los objetivos planteados.

- **Situación**. El agente recibe entradas sensoriales de un entorno en el que está situado y realiza acciones que cambian dicho entorno.
- **Autonomía**. El sistema es capaz de actuar sin la intervención directa de los humanos y tiene control sobre sus propias acciones y estado interno.
- **Flexibilidad**
  - **Reactivo**. El agente debe percibir el entorno y responder de forma temporal a los cambios que ocurran.

- **Pro-activo**. Los agentes no deben simplemente actuar en respuesta a su entorno, sino que deben exhibir comportamientos dirigidos a lograr objetivos y tomar la iniciativa cuando sea necesario.
- **Social**. Los agentes deben ser capaces de interactuar (cuando sea apropiado) con otros agentes artificiales o humanos para completar su propio proceso de resolución del probleam y ayudar a otros con sus actividades.

Un **sistema basado en agentes** en un sistema en el que la abstracción clave utilizada es efectivamente el agente.

2.1. Sistemas multiagente.

Un **sistema multi-agente** es un sistema diseñado e implementado con varios agentes interactuando. Son interesantes para resolver problemas que tienen múltiples perspectivas, entidades o formas de ser resueltos.

Las características principales de un sistema multiagente son:

- Cada agente tiene información incompleta o no todas las capacidades para resolver el problema, por lo que cada uno tiene un punto de vista limitado.
- No hay un sistema de control global.
- Los datos no están centralizados.
- La computación es asíncrona.

2.2. Arquitecturas de agentes.

2.2.1. Arquitecturas deliberativas.

- **Sistema de símbolos físicos**. Conjunto de entidades físicas (símbolos) que pueden combinarse para formar estructuras y que es capaz de ejecutar procesos que operan con dichos símbolos de acuerdo a conjuntos de instrucciones codificadas simbólicamente.
- **Hipótesis de sistema de símbolos físicos**. Los sistemas de símbolos físicos son capaces de generar acciones inteligentes.
- **Agente deliberativo**. Es aquel que contiene un modelo simbólico del mundo explícitamente representado y cuyas decisiones se realizan a través de un razonamiento lógico basado en emparejamientos de patrones y manipulaciones simbólicas.

Sus principales problemas son el traslado del mundo real a una descripción simbólica en un tiempo razonable y la representación de entidades complejas del mundo real en lenguaje simbólico.

2.2.2. Arquitecturas reactivas.

El comportamiento inteligente de estas arquitecturas surge como resultado de la interacción del agente con su entorno. La inteligencia está en el ojo del espectador, no es una propiedad innata ni aislada.

Los agentes reactivos seleccionan las acciones sobre la base de las percepciones actuales, ignorando el resto de percepciones históricas. Ejemplo: un robot de limpieza.

Las representaciones del mundo pueden ser fundamentalmente de dos tipos:

- **Basadas en características**. Es una representación parcial del mundo, lo que nuestro agente es capaz de percibir.
- **Icónicas**. Representación total del mundo.

Diseño de un agente reactivo EL flujo de interacción que debe seguir un agente reactivo es el siguiente:

1. Percibir su entorno a través de sensores.
2. Procesar la información percibida y realizar una representación interna de la misma.
3. Escoger una acción de entre las posibles considerando la información percibida.
4. Transformar la acción en señales para los actuadores y realizarla.

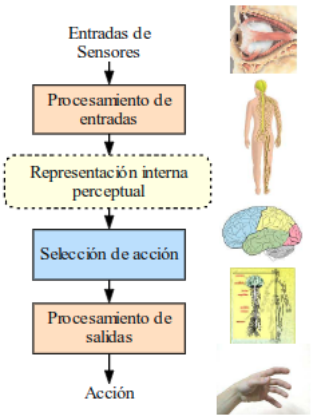

Figura 4: Flujo de interacción de un agente reactivo

Representación e implementación de funciones para seleccionar acciones.
Para realizar la selección de acciones debemos construir una función sobre el vector de características que tenga R valores de salida distintos, suponiendo que existen R posibles acciones a elegir. Podemos representar e implementar estas funciones de varias formas:

Sistemas de producción Proporcionan una representación adecuada para las funciones de selección de acciones. Una sistema de producción está formado por un conjunto de reglas (producciones). Cada producción se escribe de la siguiente manera: $c_i \rightarrow a_i$ donde c_i es la condición y a_i es la acción.

La condición de una relga puede ser cualquier función booleana definida sobre el vector de características.

Redes Consiste en implementar los sistemas de producción por medio de circuitos electrónicos (red de puertas lógicas). Las redes están formadas por elementos con umbral o cualquier otro elemento que aplique una función no lineal sobre una suma no ponderada de sus entradas.

Las funciones booleanas que se pueden implementar mediante una unidad lógica con umbral (ULU) se denominan *funciones linealmente separables*.

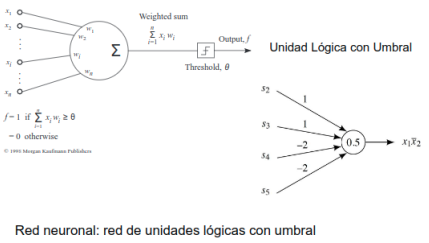

Figura 5: Unidad lógica con umbral (ULU).

Arquitectura de subsunción Consiste en hacer que el comportamiento global del agente descanse sobre un conjunto de módulos de comportamiento. Cada uno de ellos recibirá información directamente de las entradas sensoriales. Si estas entradas cumplen una precondición específica de cada módulo, se ejecutará un programa específico. Además, un módulo de comportamiento puede ser subsumido en otro.

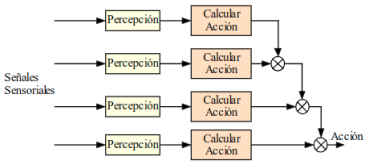

Figura 6: Arquitectura de subsunción.

Agentes reactivos con memoria Surgen para paliar nas limitaciones del sistema sensorial de un agente y mejorar la precisión teniendo en cuenta el historial sensorial.

En estos agentes, al representación de un estado en el instante $t + 1$ depende de la representación del estado en el instante t y la acción seleccionada en es momento.

Los agentes reactivos se diseñan completamente, por lo que es necesario anticipar todas las posibles reacciones para todas las situaciones. Realizan pocos cálculos y lo almacenan todo en memoria.

3. Tema 3. Búsqueda en espacios de estados.

3.1. Diseño de un agente deliberativo.

El agente dispone de varios elementos:

- Un modelo del mundo en el que habita.
- Un modelo de los efectos de sus acciones sobre el mundo.

Nuestro agente será capaz de razonar sobre esos dos modelos para decidir qué hacer para conseguir un objetivo.

3.2. La búsqueda en un espacio de estados.

Un espacio de estados es una representación del conocimiento a través de las acciones del agente.
La búsqueda en el espacio de estados consiste en resolver el problema mediante la proyección de las distintas acciones que puede realizar nuestro agente.

Una estructura de **grafo dirigido** nos puede ser muy útil para buscar sencuencias de acciones que nos lleven al objetivo final. Un nodo representa un estado del sistema y una arista una posible acción. Esta estructura se denomina **grafo de estados**.

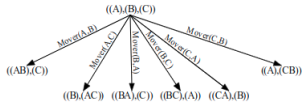

Figura 7: Grafo de estados.

3.3. Descripción de un problema.

Un problema queda definido por los siguientes elementos:

- Un estado inicial e_i
- Un estado final e_f
- Una lista de posibles acciones a_i
- El costo de estas acciones c_i

3.4. Procedimiento de búsqueda.

El procedimiento que tiene que seguir un algoritmo de búsqueda es el siguiente:

1. $DATOS \leftarrow$ base de datos inicial.
2. Mientras que $DATOS$ no satisfaga la condición final:
   a) Seleccionar un $A \in ACCIONES$ que sea aplicable a $DATOS$.
   b) $DATOS \leftarrow$ resultado de aplicar A a $DATOS$
3. Fin.

3.5. Estrategias de control.

3.5.1. Estrategias irrevocables.

En estas estrategias, el grafo explícito está formado por un único nodo en cada momento. Este nodo incluye la descripción completa del sistema. El algoritmo es el siguiente:

1. Se selecciona una acción A.
2. Se aplica sobre E para obtener $E' = A(E)$.
3. $E \leftarrow E'$.

3.5.2. Estategias tentativas.

Pueden ser de dos tipos:

- Estrategias retroactivas.
- Búsqueda en grafos.

Estrategias retroactivas. Se basan en guardar en memoria únicamente un hijo de cada estado, manteniendo el camino desde el estado inicial al actual. Por tanto, el grafo explícito es realmente una lista. El proceso se detiene cuando llegamos al objetivo y no deseamos encontrar más soluciones o bien cuando ya no podemos aplicar más operadores al nodo raiz.

La vuelta atrás se produce si se cumple alguna de estas condiciones:

- Encontramos una solución pero deseamos encontrar otra alternativa.
- Llegamos a un límite en el nivel de profundidad explorado o en el tiempo de exploración de una misma rama.
- Generamos un estado que ya existía en el camino.
- No existen reglas aplicables al último nodo de la lista (último nodo del gráfico explícito).

Búsqueda en grafos. En memoria guardaremos todos los estados (nodos que hayamos generado hasta el momento). De esta forma, la búsqueda se realizaría de la siguiente forma:

1. Seleccionar un estado E del grafo.
2. Seleccionar un operador A aplicable sobre E.
3. Aplicar A para obtener un nodo $A(E)$.
4. Añadir el arco $E \rightarrow A(E)$ al grafo.
5. Repetir.

3.6. Medidas del comportamiento de un sistema de búsqueda.

- **Completitud**. Hay garantía de encontrar la solución en el caso de que exista.
- **Optimalidad**. Hay garantía de encontrar la solución óptima.
- **Complejidad en el tiempo**. Tiempo que requiere para encontrar la solución.
- **Complejidad en el espacio**. Memoria que requiere para realizar la búsqueda.

3.7. Búsqueda sin información.

3.7.1. Búsqueda en anchura.

1. Creamos una cola de nodos *abiertos* (por visitar).
2. Creamos un set de nodos *cerrados* (ya visitados).
3. Metemos el nodo de origen en *abiertos*.
4. Mientras que haya elementos el *abiertos* y no encontremos solución:
   a) Sacamos el primer elemento de abiertos.
   b) Si es solución, la devolvemos.
   c) En caso contrario, metemos el nodo en cerrados y calculamos sus nodos hijos.
   d) Si el hijo no está ni en abiertos ni en cerrados y es viable, lo metemos en abiertos.
5. Fin

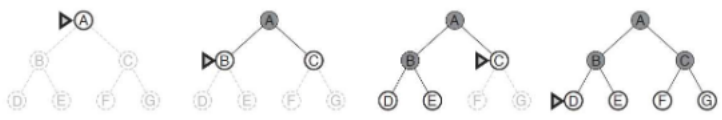

Figura 8: Búsqueda en anchura.

Características

- **Completitud**. Encuentra la solución si existe y el factor de ramificación es finito en cada nodo.
- **Optimalidad**. Si todos los operadores tienen el mismo coste, encuentra la solución óptima.
- **Complejidad en el tiempo**. Buena si las metas son cercanas.
- **Complejidad en el espacio**. Consume memoria exponencialmente.

3.7.2. Búsqueda en profundidad.

Es igual que la búsqueda en anchura, sólo tenemos que sustituir la cola (FIFO) por una pila (LIFO).

En cada paso se genera un solo sucesor, es decir, cada vez que obtenemos un hijo, se le aplica un operador obteniendo un sucesor nuevo (y así sucesivamente). Para evitar que este proceso sea infinito, establecemos una profundidad límite a partir de la cual se dejarán de generar sucesores.

Como podemos ver, solo tenemos que almacenar la parte del árbol que contiene el camino que está siendo explorado, por lo que el consumo de memoria crece de forma lineal. Sin embargo, si encontramos el objetivo, no podremos asegurar que sea el situado a menor profundidad.

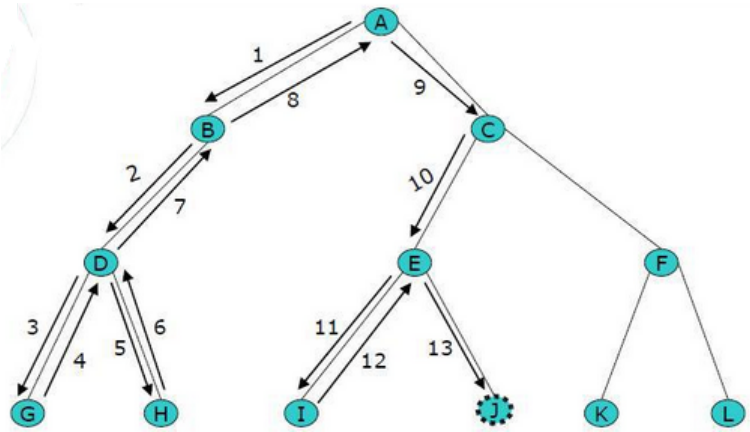

Figura 9: Búsqueda en profundidad.

Características

- **Completitud**. No asegura encontrar solución.
- **Optimalidad**. No asegura encontrar la solución óptima.
- **Eficiencia**. Buena si las metas están lejos o hay problemas de memoria.
- No es un buen algoritmo cuando hay ciclos.

3.7.3. Búsqueda con costo uniforme.

Igual que la búsqueda en anchura sustituyendo la cola por una cola con prioridad con el costo del camino como factor de prioridad. Es decir, elegimos siempre el nodo que tenga menor coste. Es un algoritmo voraz (*Greedy*).

3.7.4. Descenso iterativo.

Consiste en realizar varias búsquedas en profundidad aumentando cada vez la profundidad límite hasta que encontremos el objetivo. De esta forma garantizamos que el objetivo encontrado es el más cercano al nodo inicial.

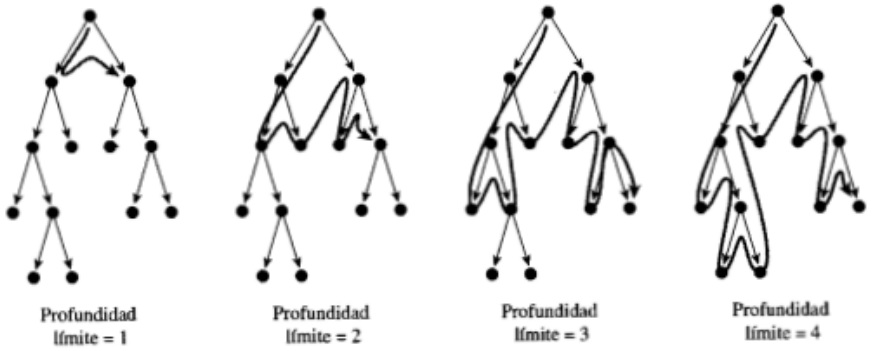

Figura 10: Descenso iterativo.

3.7.5. Búsqueda bidireccional.

Consiste en realizar búsquedas desde los dos extremos: desde el inicio intentando encontrar el destino y desde el destino intentando encontrar el inicio.

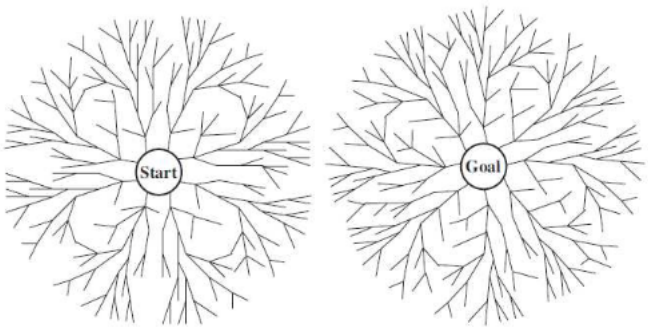

Figura 11: Búsqueda bidireccional.