DE GÖDEL AL APRENDIZAJE PROFUNDO: UNA VISIÓN PERSONAL DE LA INFORMÁTICA

Rafael Morales Bueno

Discurso de ingreso como Académico de Número, 1 de junio de 2016

AGRADECIMIENTOS

A mis padres, Rafael Morales García y María Luz Bueno Hermoso, por haberme dado la vida y enseñarme a vivirla. A mi mujer e hijos, Mariló Becerra, Carlos y Ana, son lo mejor que me ha ocurrido en la vida.

Habría muchas personas que tendría que citar, pero solo cito dos malagueños: La señorita Loli Molina y D. Juan Ruano, ejemplos de total dedicación y entusiasmo por la enseñanza. Sean las siguientes líneas un homenaje a todos ellos.

LA INFORMÁTICA EN MÁLAGA: INDUSTRIA Y TITULACIONES

Dado que soy la primera persona procedente de la ETS de Ingeniería Informática de la Universidad de Málaga que se incorpora a la Academia Malagueña de Ciencias, voy a hablar brevemente de los orígenes de la Informática en Málaga desde la perspectiva industrial y desde la académica, pues de Despeñaperros para abajo hemos sido pioneros. Comenzaré con el pasado industrial.

En 1955, Siemens abre su centro de Málaga para la fabricación de componentes (Fig. 1).

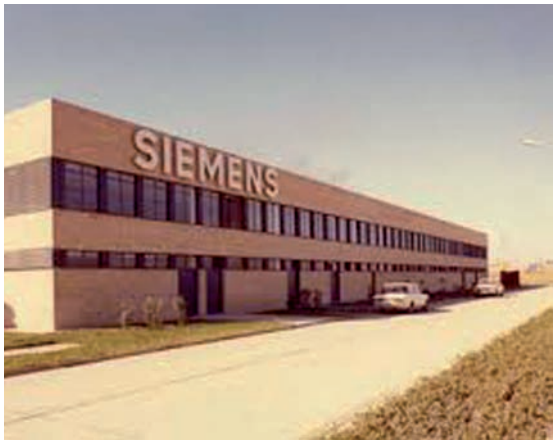

Figura 1. Fábrica de Siemens en Málaga.

En 1964 se instala una fábrica en Martiricos, la fábrica de aparatos telefónicos de CITESA (Compañía Internacional de Telecomunicación y Electrónica, S.A).

En 1975 se crea Secoinsa (Sociedad Española de Comunicaciones e Informática) bajo los auspicios del INI y con la colaboración de Telefónica. En 1977 está funcionando su fábrica en Málaga, donde también tuvo departamento de I+D (Fig. 2).

Este origen industrial se puede considerar la semilla del PTA, y ahora con el impulso dado en Málaga, nuestra ciudad está a la cabeza de las Smart cities.

Figura 2. Fábrica de Fujitsu en el Polígono del Guadalhorce, Málaga.

En este contexto de referencia industrial en Informática y Telecomunicaciones en el sur de España, la Escuela de Peritos consideró muy apropiado que se impartiera en Málaga la titulación de Diplomado en Informática (recientemente creada). Tras la correspondiente solicitud, en 1982 se crea la Escuela Universitaria Politécnica, que junto con las titulaciones de Ingeniería Técnica Industrial, impartió desde octubre de 1983 la Diplomatura de Informática (la primera al sur de Despeñaperros), posteriormente convertida en Ingeniería Técnica Informática.

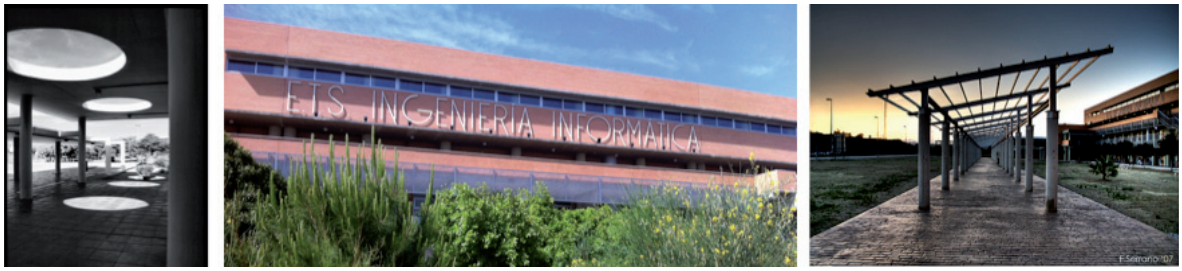

Figura 3. Escuela Técnica Superior de Ingeniería Informática, Teatinos, Málaga.

En 1988 se crea la Facultad de Informática (también la primera en el sur de España), que cambió su denominación a la actual de ETS de Ingeniería Informática en 1995 (Fig. 3).

Vaya mi recuerdo para quienes han dirigido este centro: Francisco Triguero, José María Troya, que tristemente ya no está con nosotros y al actual, aquí presente, Ernesto Pimentel.

También un afectuoso saludo a Javier López, Director del Departamento de Lenguajes y Ciencias de la Computación. Este departamento fue el origen de los diversos departamentos vinculados a la Informática en la Universidad de Málaga.

INTRODUCCIÓN

Considero a la Informática como una ciencia, que tiene sus propios principios, métodos y problemas abiertos y este es el enfoque adoptado aquí.

El siguiente párrafo está adaptado de: *The nature of computation*, Moore & Mertens, Oxford U.P. 2011.

Las ciencias que nos enseñan y nos inspiran tratan con los fundamentos. La Biología intenta comprender la naturaleza de la vida, desde las células hasta los organismos complejos. La Física busca descubrir las leyes de la naturaleza, desde la escala subatómica hasta la escala cósmica. Investigar en estas cuestiones está entre las razones que hacen que la vida merezca la pena ser vivida. Perseguir respuestas a ellas es una de las mejores cosas que los seres humanos pueden hacer.

La Informática, desde este punto de vista, intenta comprender por qué y cómo algunos problemas son fáciles y otros difíciles. Y esto no es cuestión de cómo de rápidos son nuestros ordenadores, más bien es una cuestión sobre la estructura de esos problemas, y como esas estructuras nos ayudan a resolver esos problemas o frustran nuestro intento de hacerlo. Esto nos lleva a cuestiones sobre la naturaleza de la demostración matemática, y más aún, a cuestiones relacionadas con la inteligencia y con la creatividad.

Comenzaré con un repaso histórico, en donde el hito clave es el Teorema de Gödel: antes de su desarrollo no había modelos formales que sustentaran la Informática; tras ese resultado de 1931, y en esa década del siglo XX, se consolidan los modelos formales que, por una parte, permitieron construir máquinas que fueron las realizaciones reales de esos modelos teóricos, y por otra parte, sentaron una base precisa para poder razonar, demostrar o refutar.

Posteriormente, trataremos sobre los problemas que son fáciles de comprobar pero difíciles de resolver. De ahí pasaremos al aprendizaje, a partir de grandes volúmenes de datos y al aprendizaje profundo.

Comenzaremos por el punto clave del nacimiento de la Informática como ciencia, con un enfoque que pueda ser comprendido por una amplia variedad de lectores.

¿SE PUEDEN AUTOMATIZAR LAS SENTENCIAS JUDICIALES?

¿Por qué planteo esta cuestión? El 26 noviembre de 2015, decía un abogado en su blog: *El martes tuve el gusto de participar en el acto de la Sección TIC, que preside Paloma Llaneza en el Colegio de Abogados de Madrid, en el cual se nos planteó una curiosa interrogante ¿Podrá la Inteligencia Artificial terminar con los abogados?*

El debate se suscita a raíz de la aparición de herramientas tecnológicas desarrolladas en IBM Watson y las opiniones fueron diversas.

En la misma dirección, el pasado viernes 13 mayo de 2016 aparecía en el diario Sur: *La*

Inteligencia Artificial ejerce como abogada. Baker & Hostetler se suman a la tecnología para resolver los litigios de sus clientes. La última contratación de la firma de abogados estadounidenses no es un prestigioso letrado, ni tampoco un recién licenciado. Conocemos su nombre: Ross. Sus padres son IBM Watson y la principal característica de este nuevo abogado es que es un sistema de inteligencia artificial.

Entonces, volviendo a si se pueden automatizar las sentencias judiciales, comenzaré preguntando al amable lector: *¿qué le pediría a una sentencia judicial*? Supongo que, tras un momento de reflexión, diría: *que sea justa*. Si quisiéramos ser un poco más explícitos, diríamos que la sentencia debe condenar a los culpables y no condenar a ningún inocente. Esto lo voy a expresar separando esos dos deseos en dos conceptos:

El sistema debe ser COMPLETO: todos los culpables de forma demostrada deben ser condenados.

El sistema debe ser CONSISTENTE (NO CONTRADICTORIO): ningún inocente debe ser condenado.

Pues bien, afirmo que un SISTEMA no puede ser a la vez CONSISTENTE y COMPLETO.

¿Qué significa esto? Lo voy a expresar como si estuviera hablado con ciudadanos de dos países.

Supongamos que un ciudadano del país PRUDENTIAL dice: En mi país se ha implantado un sistema automático de dictar sentencias judiciales, que garantiza que ningún inocente es condenado (CONSISTENTE).

Entonces yo le diría, pues puede estar seguro de que hay culpables que están en la calle (INCOMPLETO).

Por otra parte, supongamos que un ciudadano del país SAFE dice: Pues en mi país se ha implantado un sistema automático de dictar sentencias judiciales, que garantiza que todos los culpables son condenados (COMPLETO).

Entonces yo le diría, pues puede estar seguro de que hay inocentes que están en la cárcel (INCONSISTENTE). Es decir, y repito lo ya dicho, un SISTEMA, con un mínimo de complejidad, no puede ser a la vez CONSISTENTE y COMPLETO.

Y esto es en esencia lo que dice el Teorema de incompletitud de Gödel (Fig. 4). Destaco que es un teorema, que tiene su demostración, con lo cual es una verdad demostrada y por tanto irrefutable.

El ejemplo de las sentencias judiciales sirve para que se pueda entender el teorema. La correspondencia entre el ejemplo y el teorema sería la siguiente: en el teorema se parte de un sistema axiomático (en el sistema judicial serían los hechos del caso, el derecho natural y las leyes vigentes); en el teorema se hacen deducciones lógicas (en el sistema judicial se hacen deducciones razonadas); en el teorema se plantea si todas las verdades son demostrables (completo) y solo ellas (consistente). En el sistema judicial se plantea si todas las culpabilidades son condenadas (completo) y solo ellas (consistente).

Figura 4. Kurt Gödel (1906-1978).

En todo este razonamiento, la palabra clave es "automático", es decir, que se pueda hacer mediante un dispositivo (real o formal) que actúe de forma "automática" independiente del ejecutor.

Lo que es clave en el teorema de Gödel es que definió un modelo formal. Esto no lo había hecho nadie antes y sobre la base de ese modelo formal se desarrollaron una serie de modelos formales que posteriormente tuvieron su correspondencia con modelos reales de ordenadores, que nos llevan hasta los ordenadores actuales.

Situados pues en el punto central del nacimiento de la Informática (el teorema de Gödel); vamos a ir ahora desde los primeros tiempos hasta ese punto; y después avanzaremos desde Gödel hasta la época reciente.

Si el lector así lo desea, puede omitir el pasado e ir a la sección titulada "La confluencia de ideas de 1936", sin perder continuidad.

LA ANTIGÜEDAD

Se puede decir que desde los babilonios (que describen procedimientos para resolver ciertas ecuaciones cuadráticas y algunos problemas de superficie de triángulos), los científicos han estado siempre interesados en los cálculos. Han buscado caminos para hacer los cálculos más fáciles y han intentado reducir la resolución de problemas a una cuestión de realizar cálculos. También es conocido este objetivo en Egipto: el papiro de Rhind (hacia 1650 a. C.) incluye reglas para realizar operaciones entre números, soluciones de ecuaciones de primer grado y cálculos de áreas y volúmenes de figuras geométricas. Solo se dan ejemplos, sin una sistematización.

También en Grecia se utilizaron procedimientos para resolver problemas: tanto geométricos como aritméticos. Quizás el ejemplo más conocido sea el algoritmo de Euclides sobre el cual haré tres comentarios:

1. Aunque venga descrito en sus *Elementos*, libro 7, proposiciones 1 y 2, no se debe a él, pues Eudoxo, anterior a Euclides, ya lo usaba.
2. Actualmente se usa para calcular el máximo común divisor de dos números, pero su origen es geométrico y, mediante el proceso de ir cortando al segmento largo la longitud del corto, intercambiar y repetir el proceso, se llega a demostrar si una medida es conmensurable o inconmensurable respecto a otra (de este último tipo es la relación entre el lado de un cuadrado y su diagonal).
3. Euclides mostró que el algoritmo funciona para tres iteraciones (el método de inducción no se formalizó hasta el siglo XVII).

Los griegos y romanos desarrollaron métodos para realizar las operaciones elementales; en sus métodos usaban un ábaco de piedrecitas (*calculi*) de donde procede la palabra cálculo. Entre los romanos la palabra *calculator* designaba dos tipos de profesiones: de una parte a los maestros que enseñaban a usar ese ábaco, de otra a los contables que existían en las casas de los grandes patricios.

El ábaco fue usado en China y Japón. Aunque se cree que lo inventaron los chinos y que pasó de Oriente a Europa, la constancia científica es que en fechas similares aparecen referencias a ábacos en Grecia y en China. Una obra anónima china (200-100 a. C.) de título *Jiuzhang Suanshu* (El arte del cálculo en nueve capítulos) describe un algoritmo para resolver un sistema de n ecuaciones con n incógnitas, incluyendo varios ejemplos.

EL ORIGEN DE LA PALABRA ALGORITMO

Volviendo al concepto de algoritmo vamos a ver de donde procede esa palabra. Durante el siglo X, bajo el califato de al-Ma'mun (Fig. 5) y con su patrocinio, floreció en Bagdad una especie de academia de ciencias denominada *La casa de la sabiduría.*

En este centro del saber se concentraban muchos sabios de distintas ramas de conocimiento.

(Se denomina al-Ma'mun al califa, nacido c. 13 de septiembre de 786 -muerto el 9 de agosto de 833, hijo del legendario Harun Al-Rashid, el califa intelectual y poeta que inspiró la famosa obra *Las mil y una noches* y quien diera inicio al período conocido como la Edad de Oro del islam. Wikipedia).

El miembro de *La casa de la sabiduría*, del que vamos a hablar, es el más importante desde el punto de vista de la Informática: Abu Ja'far Mohammed ibn Mûsâ al-Khowârizmî (780-850) (padre de Ja´far, Mohammed, hijo de Moisés, natural de Khowârizm, pequeña ciudad rusa de Khiva) (Fig. 6). De su nombre procede la palabra algoritmo cuya forma original era algorismo, que se vio corrompida por la influencia de la palabra aritmética (también de aquí procede la palabra guarismo = cifra).

Polifacético como sus compañeros, tiene contribuciones en astronomía, geografía e historia. Sobre métodos de cálculo escribe un libro que da origen a la palabra álgebra: *Kitab al jabr w'alal-muqabala* (Reglas de restauración y reducción).

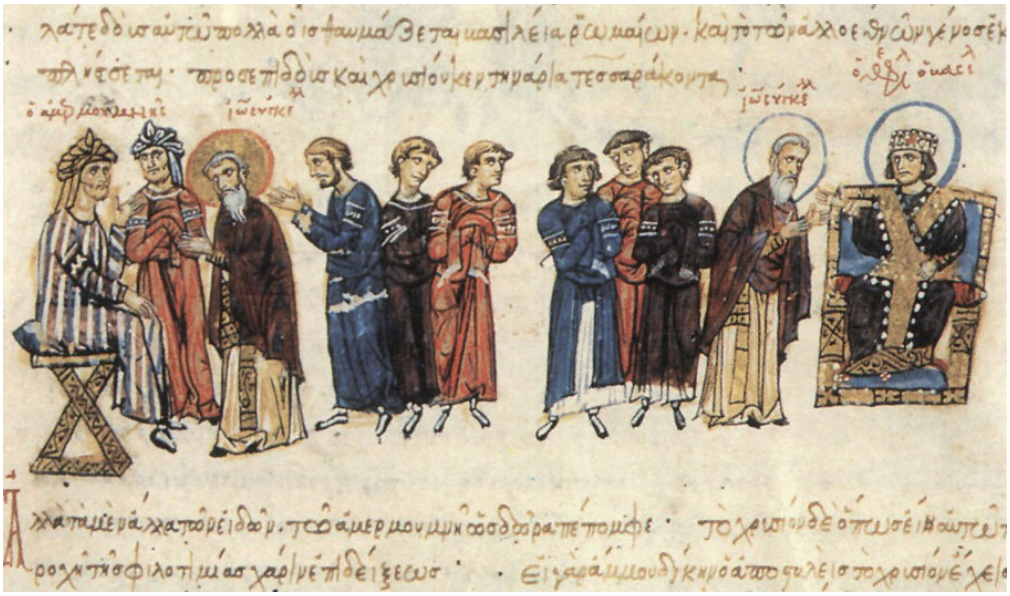

Figura 5. El califa al-Ma'mun en un manuscrito medieval. Wikipedia.

Figura 6. Sello emitido el 6 de septiembre de 1983 en la Unión Soviética conmemorando el aniversario 1200 (aprox.) del matemático persa. Wikipedia.

Como curiosidad señalo que el significado de álgebra como "restauración y reducción de miembros" era de uso común todavía en el siglo XVI en España; así, en el Quijote se usa al término algebrista en el sentido de traumatólogo (reductor de miembros).

La finalidad del Algebra de al-Khwârizmî era presentar los métodos más útiles y generales para resolver ciertos problemas algebraicos. Sus métodos para hacer las operaciones elementales son simplificaciones de métodos anteriores, pero aún así son difíciles de hacer con lápiz y papel. Dos siglos después, al-Uqlîdisî, matemático de Damasco, adaptó los algoritmos de al-Khwârizmî a métodos más apropiados para realizarlos con lápiz y papel.

RAIMUNDO LULIO

Sabemos que la matemática persa influyó en los árabes y a través de ellos llegó a España y desde aquí al resto de Europa.

Figura 7. Raimundo Lulio (ca. 1232-ca. 1316).

En concreto, los árabes inspiraron a Raimundo Lulio (ca. 1232 - ca. 1316) como sigue: Raimundo fue en peregrinación a Tierra Santa; a su vuelta decidió dedicarse a predicar la doctrina cristiana entre los árabes: pasó 9 años estudiando lengua árabe y debió entrar en contacto con la obra de los matemáticos árabes (Fig. 7).

Los métodos algorítmicos le sugirieron, hacia 1300, su obra *Ars Magna*, como un procedimiento general de base combinatoria para hallar todas las "verdades" (la motivación del libro era religiosa: la búsqueda del algoritmo universal que resolvería todos los problemas y por tanto acercaría a Dios).

Considerados desde un punto de vista sensato, los procedimientos aducidos por Lulio no tiene mucho valor. Lo importante sin embargo, es que concibió una idea ciertamente espléndida. El trabajo de Lulio tuvo una gran influencia en la matemática posterior, y así, 200 años después Cardano (1545) titula su libro sobre algoritmos algebraicos, remedando el título de la obra de Lulio: *Artis magnae seu de regulis algebraicis liber unus*.

Durante toda la Edad Media, los ábacos se usaron en Europa, pero según se extendían los métodos árabes iban apareciendo dos tendencias: abacistas y algoritmistas. Los algoritmistas reconocían la facilidad del ábaco para operaciones simples, pero su dificultad para realizar operaciones complejas: necesitaban mucho tiempo para su resolución y procedimientos sofisticados.

La puntilla al ábaco en Europa se la dio el desarrollo de máquinas mecánicas de cálculo (los ábacos siguieron usándose en China y Japón, alcanzando tal nivel de pericia que el primer ordenador llevado a Japón tras la segunda guerra mundial fue más lento al calcular, que su competidor humano, un japonés experto en el manejo del ábaco; quizás esta fue la última vez que se produjo tal victoria).

La primera máquina mecánica de la que se tiene noticia fue inventada en 1623 por el alemán Wilhelm Schickard: hacía las cuatro operaciones aritméticas y raíces cuadradas. Todo lo que queda de ella es su descripción en una carta de Schickard a Kepler. Blas Pascal en 1639 ideó otra máquina basada en engranajes para hacer sumas y restas.

LEIBNITZ Y EL ALGORITMO UNIVERSAL

Volvamos al objetivo de los algoritmos universales.

El mismo objetivo de Raimundo Lulio de reducir a cálculos la determinación de todas las verdades (lógicas) fue perseguido por Gottfried Leibnitz (1646-1716), bien conocido por sus aportaciones al Análisis Matemático (el cálculo diferencial e integral de Leibniz, que provocó un gran debate respecto al cálculo de fluxiones y anti-fluxiones de Newton; los mismos conceptos descubiertos de forma simultánea, uno en el continente europeo y otro en la Islas Británicas) (Fig. 8).

Del estudio de los trabajos de Lulio determinó que había dos conceptos que se deberían considerar por separado: *Ars inveniendi* o procedimiento de generación y *Ars iudicandi* o procedimiento de decisión. Leibniz vio con claridad la posibilidad de confiar a una máquina la ejecución de un procedimiento general de razonamiento. Su sueño era tener un *calculus ratiocinator* para "decidir" las verdades lógicas.

Figura 8. Gottfried Leibnitz (1646-1716).

Leibnitz perseguía dos objetivos: uno, construir un lenguaje formal lo más amplio posible para expresar cualquier enunciado, incluyendo por supuesto a la aritmética. El otro

era llegar a un procedimiento tal que si una persona opina que el enunciado A es verdad y otra que el enunciado no-A es verdad, para ponerse de acuerdo solo necesitarían realizar cálculos que darían la razón a uno de los dos.

El primer objetivo fue más tarde alcanzado con los desarrollos de la Lógica y el primer lenguaje formal construido por G. Frege en 1878. El segundo objetivo se mostró inalcanzable con los trabajos de Gödel y Turing como veremos más adelante.

Otra importante contribución teórica de Leibnitz fue el desarrollo de la numeración binaria. Conviene en este punto señalar que, según Knuth, la primera discusión sobre el sistema de numeración binario aparece en 1670 en el libro *Mathesis Biceps 1*, del obispo español Juan de Caramuel (Fig. 9): en esta obra se discuten otros sistemas de numeración distintos del decimal sin dar ejemplos de operaciones aritméticas en esos sistemas. Leibniz sí presentó a la Academia de Ciencias de Paris un artículo donde introducía las cuatro operaciones básicas en binario. Recomendaba no usar este sistema para cálculos prácticos, sino para descubrir propiedades de los números que ayuden a diseñar algoritmos más eficaces.

Figura 9. Juan Caramuel (1602-1686).

El comienzo del siglo XIX presenta otro hito en el desarrollo de la Informática con Charles Babbage (1792-1871). En esa época se disponía de piezas y engranajes cada vez más precisos; Babbage concibió la idea de construir un instrumento mecánico para calcular e imprimir funciones matemáticas: *la máquina de diferencias* (para ayudar a los "computers", calculistas que hacían a mano las tablas de logaritmos, trigonométricas, astronómicas, de tiro, etc.). Sin concluirla, abandonó el proyecto e ideó *la máquina analítica*, concebida como un "computador universal" totalmente automático. Su estructura se correspondía con la que hoy se conoce como "arquitectura de Von Neumann": memoria, unidad de control, entrada y salida. Lo que se sabe del trabajo de Babbage se debe a los escritos de su ayudante Augusta Ada, condesa de Lovelace (1815-1852). Se la considera la primera programadora pues diseñó el lenguaje para la máquina analítica, llegando a escribir una rutina para el cálculo de los números de Bernouilli (Fig. 10).

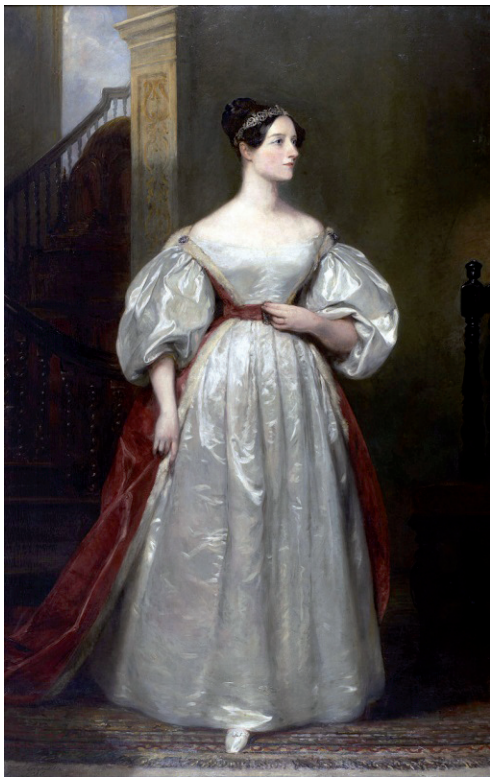

Figura 10. Augusta Ada, condesa de Lovelace (1815- 1852).

Si bien estos desarrollos tienen interés practico, los objetivos de determinar qué se puede resolver de forma algorítmica y qué se puede resolver eficientemente, necesitan de una expresiónmás precisa, que vino de la mano de la Lógica.

LA LÓGICA FORMAL

La silogística de Aristóteles fue un primer intento en esta dirección. Es en el siglo XIX cuando se produce el desarrollo de la Lógica moderna.

Los métodos de cálculo de George Boole (1815-1869) se constituyeron, en gran medida, de acuerdo con los métodos usuales del álgebra, siendo los primeros pasos hacia una formalización de la Lógica. Tras varios desarrollos a lo largo del siglo XIX, a finales de siglo, Gottlob Frege (1848-1925) desarrolla un sistema suficientemente amplio como para proceder a una formalización que es base para el desarrollo posterior. En su artículo *Begriffschrift* de 1879, presenta el que se puede considerar primer lenguaje formal, en el cual se pueden expresar enunciados tales como completitud (si todos los enunciados son demostrables) o decidibilidad (si existen algoritmos para demostrar los enunciados). B. Russell fue quién reconoció su importancia y divulgó la obra de Frege. Su libro *Principia Mathematica* está basado en la obra de Frege.

En paralelo a Frege, Guiseppe Peano (1858-1932) escribió *Arithmeticae Principia. Nova methodo exposita*, en donde axiomatiza la aritmética y de donde se deduce toda la teoría de números naturales.

Al final de esta época está bien establecido que gran parte de la matemática puede ser deducida con la ayuda de un cálculo lógico.

Observamos como todos estos desarrollos perseguían reducir los procesos de demostración a meros cálculos o procedimientos mecánicos que no dependan del ejecutor.

HILBERT Y EL ENTSCHEIDUNGSPROBLEM

Pasemos ahora a reseñar los éxitos y fracasos posteriores. Para situarnos en el entorno científico de la época, tras la gran cantidad de desarrollos de la Matemática en los siglos anteriores, y la creación de la teoría de conjuntos, aparecen tres grandes escuelas de pensamiento sobre la fundamentación de la Matemática: la logicista, encabezada por los matemáticos ingleses Russell y Whitehead, la intuicionista, con el holandés Brower a la cabeza y la escuela formalista o axiomática de Hilbert.

La tesis logicista es que la Matemática es una rama de la Lógica. Las nociones matemáticas han de ser definidas en términos de las nociones lógicas. Los teoremas de la matemática han de ser demostrados como teoremas de Lógica.

La escuela intuicionista defiende la no aceptación de la ley del tercero excluido para conjuntos infinitos. Así por ejemplo, si en un dominio infinito D, supuesto que todo miembro de D no tiene la propiedad P, se alcanza una contradicción, se puede deducir que existe un elemento de D que tiene la propiedad P; este razonamiento de tipo existencial no es aceptado por la escuela intuicionista.

En tercer lugar, el programa de la escuela formalista fue esbozado por David Hilbert en 1904 (Fig. 11), donde se propone: *formular la matemática clásica como una teoría axiomática formal, y deberá demostrarse que esta teoría es consistente, esto es, libre de contradicción*.

Figura 11. David Hilbert (1862-1943).

Este programa fue acometido por Hilbert y su equipo desde 1920. El problema más grave era la demostración de la no contradicción. (El método habitualmente utilizado era proporcionar un modelo, con lo cual solo se alcanzaba una demostración de consistencia relativa: si el modelo era consistente, la teoría también lo era. Este método no ofrecía soluciones para casos como la teoría clásica de números o la teoría de conjuntos).

En este ámbito se sitúa el problema de la decisión o *Entscheidungsproblem* como uno de los más ambiciosos de la meta matemática. En la Matemática, hay cuestiones generales tales que cualquier instancia particular se puede responder mediante un método efectivo, que aplicado a ese caso nos dará una respuesta SI o NO. Así por ejemplo, hay métodos para saber, dados dos polinomios con coeficientes enteros, si uno es factor del otro; basta realizar la división y observar si el resto es cero. Esto se puede hacer de manera mecánica y en un número finito de pasos. Un método de este tipo se denomina un procedimiento de decisión. El problema de la decisión es determinar si existen tales métodos para saber si una fórmula (o expresión) es válida (es decir, verdadera) o no, en un lenguaje formalizado. Este problema se asocia habitualmente con Hilbert por los grandes esfuerzos que dedicó a su resolución, pero aparece anteriormente en Schroeder y en Löwenheim.

Y ahora ya nos vamos acercando a darle forma a los algoritmos y a definir sustitutos formales en la década de 1930, lo que llevaría a la construcción de ordenadores en la década de 1940.

Hay que tener en cuenta que "método" o "procedimiento" es aún un concepto informal. En esa época había cierto consenso en lo que se podía considerar como método, en el sentido de usar un espacio finito, usar un tiempo finito y no dependencia de quien aplica el método. Siguiendo con el programa de trabajo, Hilbert y Ackermann delimitaron, hacia 1928, de un modo claro, la lógica de primer orden y presentaron un cálculo deductivo para ella. Se plantearon la cuestión de si ese cálculo deductivo era semánticamente suficiente, esto es, si permitía deducir todas las fórmulas válidas. En su libro *Elementos de Lógica Teórica* (HILBERT & ACKERMANN 1928) manifiestan explícitamente que aún no habían encontrado respuesta a esa pregunta.

Dos años después de la publicación del libro de Hilbert y Ackermann, el austriaco Kurt Gödel (1906-1978) publica su trabajo sobre *La suficiencia de los axiomas del cálculo lógica de primer orden* (GÖDEL 1930), en donde demuestra que, mediante los axiomas y las reglas de inferencia de la lógica, en el ámbito de la lógica de primer orden, toda fórmula válida es deducible. Con esto el programa de Hilbert obtenía un primer y esperanzador éxito.

Sin embargo Gödel continuó sus trabajos y al año siguiente publicó su trabajo: *Sobre las sentencias formalmente indecidibles de los Principia Mathematica y sistemas afines* (GÖDEL 1931) en el cual demostraba la imposibilidad de llevar a cabo el programa de Hilbert. (Recuérdese que el objetivo era desarrollar sistemas formales y demostrar su consistencia). Gödel demostró que todos los sistemas formales de la Matemática clásica (incluidos el sistema de los Principia Mathematica, la aritmética de Peano, la teoría axiomática de conjuntos, y en general, cualquier sistema formal que cumpliese ciertas condiciones de aceptabilidad) son incompletos o contradictorios (como lo hemos explicado de manera informal, con el ejemplo de las sentencias judiciales al inicio de este discurso).

La demostración de Gödel se basa en una modificación de la paradoja del mentiroso en la que se sustituye "ser verdadero" por "ser demostrable": Se especifica una fórmula F que depende del sistema de reglas y axiomas dado, y tal que la semántica de F puede ser interpretada como que "F no es deducible en el sistema". F no contiene variables libres, por lo cual F o ¬F es válida. Si F es válida, entonces F no es deducible y por tanto el sistema es incompleto. Si ¬F es válida y el sistema es completo, entonces ¬F debe ser deducible, luego F no es deducible, con lo cual, por definición de F, es válida. Esto es, F y ¬F son ambas válidas, con lo cual el sistema es contradictorio.

Además esta incompletitud no tiene remedio: por muchos axiomas que añadamos, el sistema sigue siendo incompleto.

También demostró que es imposible probar la consistencia de un sistema formal dentro del propio sistema (se puede probar desde una teoría más potente que el propio sistema formal, pero eso es de dudosa utilidad).

En este trabajo, Gödel define y utiliza las funciones recursivas primitivas (que él llama recursivas) de las que afirma que: *... tienen la importante propiedad de que para cada conjunto dado de argumentos el valor de la función puede computarse mediante un procedimiento finito*. (Nótese la referencia a procedimiento; ya nos vamos acercando a la formalización del concepto de algoritmo).

Estos resultados negativos se vieron completados en el trabajo de Turing de 1936,

en el cual junto con la definición de la máquina que lleva su nombre, de la que hablaremos en el punto siguiente, demuestra que el cálculo de predicados es indecidible.

Observamos que para poder demostrar la indecidibilidad del cálculo de predicados era necesario conocer formalmente el significado de la palabra "decidible".

Pues bien, alrededor de los años 30 estas cuestiones estaban suficientemente maduras como para que en 1936 aparecieran una serie de artículos que convergían en el concepto de algoritmo.

¿HAY ALGÚN ESPAÑOL EN ESTOS INICIOS DE LA INFORMÁTICA?

En este punto me voy a referir a Leonardo Torres-Quevedo, que fue un ingeniero español que construyó un dispositivo electromecánico llamado El Ajedrecista, en 1912 (Fig. 12).

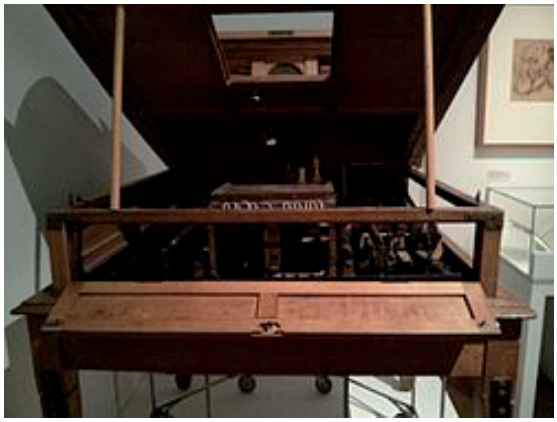

Figura 12. El ajedrecista construído por Leonardo Torres Quevedo en 1912.

El ajedrecista hizo su debut durante la Feria de París de 1914, generando gran expectación en aquellos tiempos y hubo una extensa primera mención en Scientific American como *Torres and His Remarkable Automatic Device* (Torres y su Extraordinario Dispositivo Automático) el 6 de noviembre de 1915.

También es poco conocido que el transbordador aéreo que cruza las cataratas del Niágara se debe a él, pues su diseño ganó el concurso internacional convocado para ese fin (Fig. 13).

Pues bien, junto a la base del transbordador hay una placa con una inscripción que dice que Leonardo Torres-Quevedo fue el que construyó el primer computador.

Figura 13. Transbordador aéreo que atraviesa las cataratas del Niágara.

LA CONFLUENCIA DE IDEAS DE 1936

Gödel en su trabajo, usó unas funciones de las que decía: *... tienen la importante propiedad de que para cada conjunto dado de argumentos el valor de la función puede computarse mediante un procedimiento finito*. Está ya dando una formulación de lo que después serían las funciones recursivas.

En abril de 1935, Alonzo Church presentó una comunicación en la reunión de la AMS, donde formalizaba las funciones recursivas en términos del λ-cálculo y enunció la equivalencia entre su clase de funciones y las calculables por procedimientos efectivos (hoy conocida como tesis de Church-Turing). Cuando Church expuso su tesis, Stephen C. Kleene (posteriormente estudiante de doctorado suyo) se propuso demostrar su falsedad mediante diagonalización sobre toda la clase de las funciones definibles por medio del λ-cálculo. Al ver la imposibilidad de conseguirlo, cambió de opinión y fue Kleene quién le dio el nombre de tesis de Church (Fig. 14).

En septiembre de 1935, Kleene presentó en una comunicación los resultados fundamentales de su tesis doctoral en donde, siguiendo las funciones que usó Gödel en su demostración del teorema de incompletitud, definió las funciones recursivas primitivas y las funciones recursivas generales.

Figura 14. Alonzo Church (1903-1995).

Demostraba, entre otros resultados, el teorema de la Forma Normal que lleva su nombre (este teorema en terminología actual se podría enunciar como sigue: cada programa admite uno equivalente usando bucles FOR y un solo WHILE).

En 1936, las dos comunicaciones de Church y Kleene se publicaron en forma de artículos. En ambos casos dieron caracterizaciones formales de la noción de función efectivamente calculable y demostraron la existenca de problemas indecidibles pero sin exhibir ninguno. La presentación explícita de un problema indecidible se debe a Alan Turing (Fig. 15), que en 1936 definió la que hoy se conoce como máquina de Turing (él la llamó A-machine, la "A" de automática). Demuestra la universalidad de su máquina, la indecidibilidad del problema de la parada y del problema de la decisión por métodos constructivos; y, en un trabajo posterior, demuestra la equivalencia de su modelo con los otros modelos ya citados. También definió máquina con oráculo y la reducibilidad entre problemas para comparar dificultades de resolución.

Una característica importante de la máquina de Turing como modelo es que define el concepto de paso de cómputo, lo cual será clave para el desarrollo de la teoría de la complejidad.

El mismo año de 1936, Emil Post define el modelo que lleva su nombre. Muy parecido al modelo de Turing, lo publicó en un artículo con pocos meses de diferencia, pero escrito de manera totalmente independiente.

Todos estos conceptos intentan capturar la idea intuitiva de algoritmo; así, por ejemplo, Turing explica en su artículo que desea simular el comportamientode un calculador humano (los "computers" de la Inglaterra del siglo XIX que calculaban tablas de logaritmos y trigonométricas, etc.), que a partir de unos datos y unas reglas de operación escribe resultados intermedios, borra, consulta estos resultados hasta alcanzar un resultado final y durante este proceso va evolucionando a través de diversos estados que reflejan cómo evoluciona el proceso desde el punto de partida hasta un punto final.

Figura 15. Alan Turing (1912-1954).

ESTUDIA INFORMÁTICA Y GANA UN MILLÓN

Llegados a este punto, tenemos un modelo formal (máquina de Turing) asociado a un concepto informal (algoritmo), y además, ese modelo formal se ha convertido en un dispositivo real: ordenador.

Entonces, lo que puede hacer una máquina de Turing es lo que puede hacer un ordenador y lo que no puede hacer una máquina de Turing, no lo puede hacer un ordenador.

De manera que sabemos que hay problemas que un ordenador no puede resolver. Y además esos problemas forman una jerarquía (hay unos más difíciles que otros).

También hay problemas para lo que es fácil encontrar la solución y otros para los que es fácil comprobar la solución (y no sabemos si es fácil encontrar la solución).

Primero vamos a aclarar que queremos decir con fácil y después nos centraremos en el segundo tipo (los problemas fáciles de comprobar la solución).

Entendemos por "fácil" que el esfuerzo sea proporcional al tamaño del problema. Pongamos un ejemplo, es "fácil" repartir caramelos entre niños, solo que el esfuerzo está relacionado con la cantidad de caramelos y la cantidad de niños: repartir 20 caramelos entre 5 niños, requiere un esfuerzo proporcional a los valores 20 y 5. Repartir 1 millón de caramelos entre 500.000 niños, requiere un esfuerzo proporcional a esos valores.

Pero hay otros problemas que no tienen esta característica. Veamos dos ejemplos:

1. Colorear el mapa de las provincias de España: seguro que lo hemos hecho alguna vez. La condición es que provincias con frontera común no tengan el mismo color, pues si usamos el mismo color para Málaga y Córdoba, no sabríamos donde acaba Málaga y dónde empieza Córdoba. Cualquiera considera este problema una cuestión infantil y sin interés.

2. Asignar invitados a las mesas de una celebración. Esto es un problema serio. Todos de forma directa o indirecta habremos vivido situaciones de debate y discusión del tipo *no se te ocurra sentar a Armando en la misma mesa que Venancio, que ya sabes lo que pasó…*

Primero vamos a ver la conexión entre los dos problemas. Alguien podría decir que el problema 1 no plantea ninguna dificultad; si el número de provincias es alrededor de 50, se toma la caja de 60 colores y un color para cada provincia. Problema resuelto (Fig. 16).

¿Y si hiciéramos lo mismo en el segundo? Entonces diríamos: no hay problema: 50 invitados, pues 50 mesas; una para cada invitado; así no se pelea nadie. Problema resuelto. ¿Y que nos diría nuestro entorno familiar? ¡Te das cuenta de la tontería que acabas de decir!; si cada mesa es de 10, queremos usar 5 mesas o a lo sumo 6. Ah!, entonces queremos el mínimo de mesas; pues la coherencia dice que en el coloreado del mapa se debería usar el mínimo de colores.

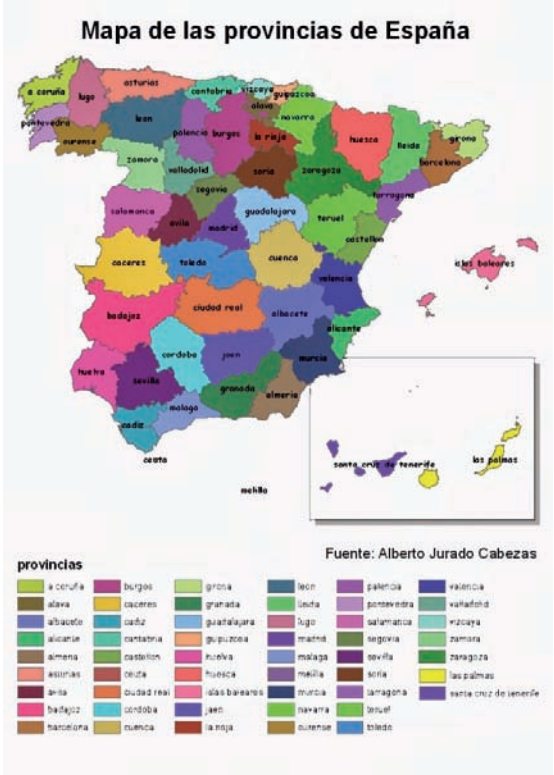

Figura 16. Mapa coloreado de las provincias de España.

Los dos problemas son el mismo: los colores son las mesas y las provincias fronterizas son los invitados que se llevan mal y no puede estar en la misma mesa (no pueden tener el mismo color).

Vamos a pensar en el problema de las mesas: ¿Es fácil comprobar una solución? Sí.

Si a los responsables de la celebración, les dan una asignación de invitados a mesas, de un vistazo dirán "esta mesa bien; esta también,..., de acuerdo está bien".

Entonces, un enfoque directo sería: vamos a generar todas las posibles formas de sentar a los 50 invitados en 5 mesas y que los organizadores comprueben una a una. Para esas cifras, haciendo 1000 comprobaciones por segundo, tardarían más que el tiempo que se estima existe el Universo.

A esto me refiero con problemas fáciles de comprobar (y que no se sabe si son difíciles de resolver).

En el estudio de estos problemas se centraron los científicos Stephen Cook y Leonid Levin. Y volvió a ocurrir que descubrieron los

mismos resultados de forma independiente en 1972: el primero en Estados Unidos, publicando sus resultados en inglés; el segundo en la antigua Unión Soviética, publicando sus resultados en ruso. Al cabo de los años se ha conocido esa coincidencia.

Pues bien, si a alguien se lo ocurre un programa que resuelva de manera "fácil" uno de estos dos problemas (y esto que digo es aplicable a otros muchos) entonces ganará un millón de dólares. También ganará el millón de dólares si demuestra que alguno de estos problemas no admite un programa "fácil".

Y esto está patrocinado por una fundación y se puede ver en su web: http://www.claymath.org/millennium-problems

El problema número 3 trata de esto.

De momento nadie ha sido capaz de resolverlo. Y mientras tanto, ¿qué podemos hacer?

Los seres humanos, enfrentados a este tipo de problemas, son capaces de resolverlos, no en el tiempo de vida del Universo, sino en unos días o semanas.

¿Cómo? Pues razonando de forma inteligente, en lugar de explorar todas las soluciones. Y aquí entramos en terreno de la Inteligencia Artificial.

Los seres humanos aprenden. Si pudiéramos enseñar a los ordenadores a aprender, nos podríamos acercar a la forma de razonamiento humano.

En este punto se abre en dos rutas, ambas vinculadas a resolver este tipo de problemas mediante el aprendizaje:

Una, que veremos de inmediato, relativa a descubrir conocimiento a partir de las enormes cantidades de datos que se generan cada día y otra relativa al aprendizaje simulando al cerebro humano.

EL EMPLEO DEL FUTURO: DATA SCIENTIST

Desde la primera perspectiva, esa enorme cantidad de datos que se generan cada día es mayor, y más aún desde la existencia de Internet; además, están disponibles para su consulta y su explotación. Entender esos datos y hacer uso de ellos para algún fin requiere de programas de ordenador y de personas que sepan trabajar con ellos y de ahí surge el empleo del futuro.

El 28 de enero de este año 2016, en el diario Sur remitida por Europa Press, aparecía una noticia sobre: *Los 25 empleos con más futuro en 2016* y en primer lugar situaba Data Scientist, con un salario anual de 108.027 €.

Ese campo de trabajo e investigación se denomina genéricamente Aprendizaje Automático y es uno de los temas en los que trabajamos el grupo de investigación de la Universidad de Málaga "Investigación y Aplicaciones en Inteligencia Artificial", que dirige el Prof. Ricardo Conejo. Hay una serie de palabras vinculadas a este campo que enumero ahora en inglés y entre paréntesis en español cuando es posible, pues en muchos casos se usa directamente el término inglés: *Computational learning* (aprendizaje automático), *Knowlegde Discovery* (descubrimiento del conocimiento), *Data Mining* (Minería de datos), *Big Data, Business intelligence* (inteligencia de negocio).

Ahora voy a dar unas pinceladas relativas a lo que puede hacer un Data Scientist, relacionándolo con algunas situaciones reales de aprendizaje a partir de grandes conjuntos de datos.

Debo decir que las situaciones reales citadas son casos en los que hemos hecho aportaciones desde el grupo de investigación (IA)2 (Investigación y Aplicaciones en Inteligencia Artificial).

Consideremos que deseamos obtener conocimiento a partir de expedientes de incapacidad laboral. Tenemos una cantidad enorme de expedientes anónimos que incluyen datos de edad, peso, estatura, profesión, pruebas médicas vinculadas a patologías comunes y profesionales, etc., y una valoración en términos de incapacidad si/no, total, parcial, etc.

Entonces diremos que estamos en una situación de datos "etiquetados": cada expediente, ya valorado tiene una etiqueta que es la valoración de su incapacidad.

Esos datos están en papel. Entonces comienza la primera fase de preparación de los datos: por una parte definir una forma de codificación y por otra pasarlos a formato electrónico.

En algunos datos la codificación la define el Data Scientist, en otros hay codificaciones (por ejemplo, para las enfermedades hay unas codificaciones internacionales de 9 símbolos, CIE-9, o de 11 símbolos, CIE-11.

Otro dato importante para el Data Scientist es que estos datos son estáticos (un expediente se analiza en un momento dado).

El objetivo puede ser explicar: ciertas profesiones tiene vinculadas ciertas patologías (en este punto acaba el trabajo del Data Scientist; después vendrán los expertos para hacer propuestas para reducir los perjuicios de esas patologías). El otro objetivo puede ser predecir: dado un nuevo expediente sin valorar, usando lo que hemos aprendido de los miles de expedientes ya estudiados, construir un programa que "sugiera" una valoración para ese expediente. Esa sugerencia puede ser útil para el interesado, de manera que tenga una estimación de la valoración de su expediente; también puede ser útil para el médico valorador, pues la estimación estará en coherencia con los cientos de miles de valoraciones ya realizadas. Por supuesto, un sistema de este tipo sugiere y la decisión final es del experto humano.

Veamos ahora un segundo ejemplo de datos no etiquetados: contratación de seguros o "la cesta de la compra". Supongamos que una persona tiene una familia y una empresa, tiene una variedad de seguros de hogar, automóviles, responsabilidad civil, en las fábricas, oficinas, etc. Ese conjunto de seguros se puede ver como una cesta de la compra en la que lleva diversidad de productos.

¿Qué hacen en el Supermercado con la ayuda del Data Scientist? Pues, por ejemplo, ver productos que en las facturas aparecen muchas veces juntos y ponerlos en estanterías cercanas; para que el cliente compre unos y otros.

¿Y con los seguros? Pues si en esas cantidades enormes de datos, hay muchos asegurados con grupos de seguros iguales, y un cliente tiene un grupo parecido y le falta uno, pues entonces, el comercial de la empresa de seguros llama a ese cliente y le dice: tiene usted los seguros A, B y C. ¿No le interesaría contratar el D, ya que muchas personas en su misma situación, que tienen A, B C y también el D?

¿Quién le descubre a la empresa de seguros a qué cliente llamar? El Data Scientist.

EL APRENDIZAJE PROFUNDO

Y llegamos ahora a la simulación del cerebro humano, en relación al segundo enfoque que comentamos antes: problemas que un ordenador necesitaría cientos o miles de años para resolver, y que las personas los resuelven en un tiempo razonable. Quizás las personas no llegan a la solución óptima, pero si a una solución razonable.

Ese es el enfoque del aprendizaje profundo: programar los ordenadores para que aprendan a partir de diversas representaciones de datos, en la forma que lo hace el cerebro humano.

El cerebro está formado por miles de millones de neuronas que se organizan en capas. Cada capa produce salidas que pasan como datos de entrada a la capa siguiente, que genera a su vez salidas y así sucesivamente.

En el caso de las redes neuronales artificiales, las primeras capas identifican patrones muy simples, que pasan como entrada a la siguiente capa que combina esos patrones simples para obtener patrones más complejos.

También el ser humano aprende por refuerzo, de manera que una decisión que le resulta satisfactoria o le reporta beneficios la repetirá, mientras que en caso contrario la evitará de manera que se refuerzan los estímulos positivos.

En tercer lugar, en ausencia de información o ausencia de experiencias, intentamos una experimentación exploratoria, a ver qué decisión resulta más prometedora.

Estos modelos, cuyos algoritmos estaban inventados a finales del siglo pasado, necesitaban de la potencia de cálculo de los ordenadores actuales y su ejecución en paralelo en miles de ordenadores.

Como ejemplo muy reciente por su éxito en una competición de un juego de mesa, citaré a AlphaGO, programa desarrollado por Google DeepMind en Londres, para jugar al GO.

GO es un juego de mesa de origen chino con unos 4000 años de antigüedad. Se juega sobre un tablero de 19 x 19 y el número de situaciones que se pueden dar es muy superior a juegos conocidos como el cubo de Rubik o el ajedrez.

El programa AlphaGo tiene una red neuronal con 13 capas ocultas. Para el procesamiento se han usado casi 2000 unidades de computación y 300 de procesamiento gráfico.

La red se ha entrenado en tres fases con las que ha experimentado en total unos 30 millones de movimientos y 160.000 partidas.

Hay que tener en cuenta que su contrincante humano, Lee Sedol, si hubiera jugado una partida al día desde los 10 años, dado que la duración de una partida es de 4 ó 5 horas, habría jugado unas 8500 partidas.

La diferencia fundamental es que con solo ese entrenamiento los seres humanos llegan a niveles muy altos.

Es decir, con un ejemplo más sencillo, un programa de aprendizaje profundo, tras ver miles de fotos, diciéndole en cuales hay un perro y en cuales no, llega a identificar perros con pocos errores.

¿Cuántos perros necesita ver un niño para identificarlos correctamente? UNO, solo UNO. Esa es una diferencia fundamental.

Como conclusión, quiero señalar que todos estos experimentos en situaciones reales, generan buenos resultados y se perciben buenas expectativas para el futuro, por lo que los ordenadores ayudarán cada vez más al avance de la civilización, pero no hay demostraciones formales de su calidad y en muchos casos no las habrá porque en la base está el teorema de incompletitud de Gödel, que en el fondo dice que hace falta la intervención humana, y aun así nunca llegaremos a demostrar y a saber todas las verdades.