UNIVERSIDAD DE GRANADA

Facultad de Ciencias

GRADO EN FÍSICA

TRABAJO FIN DE GRADO

COMPUTACIÓN CUÁNTICA Y APRENDIZAJE AUTOMÁTICO

Presentado por:
D. Alberto Calzada Chávez

Curso Académico 2020/2021

A mis padres, Antonia y José.

Abstract

In this work we address the relationship between quantum computing and machine learning. We focus on a specific model of quantum computing known as the adiabatic model. Through a series of considerations on this model, we come to the concept of quantum annealing. In the second half of the study, we proceed to develop a machine learning model called the Boltzmann Machine and study how the quantum version of this algorithm takes advantage of quantum annealing, resulting in a more efficient operation than its classical analog.

Keywords: quantum computation, circuit model, universal quantum computation, adiabatic computation, quantum annealing, machine learning, Boltzmann machine, quantum Boltzmann machine

Resumen

En el presente trabajo estudiamos la relación que existe entre computación cuántica y aprendizaje automático. Nos centramos en un modelo concreto de computación cuántica conocido como modelo adiabático. Realizando una serie de consideraciones, llegamos al concepto de temple cuántico. En la segunda parte del trabajo construimos el algoritmo de aprendizaje automático conocido como Máquina de Boltzmann, y estudiamos cómo se beneficia su análogo cuántico del temple cuántico para dar lugar a un algoritmo más eficiente que su versión clásica.

Palabras clave: computación cuántica, modelo circuito, computación cuántica universal, computación adiabática, temple cuántico, aprendizaje automático, máquina de Boltzmann, máquina de Boltzmann cuántica

Índice

1 Introducción 4

2 Introducción a la computación cuántica 4
- 2.1 Modelo de puertas . . . 4
  - 2.1.1 Qubits . . . 5
  - 2.1.2 Puertas cuánticas . . . 5
  - 2.1.3 Conjunto de puertas universal . . . 7
- 2.2 Modelo adiabático . . . 8
  - 2.2.1 Teorema adiabático . . . 9
  - 2.2.2 Computación cuántica adiabática . . . 10
  - 2.2.3 Sobre la universalidad en computación cuántica adiabática . . . 11
- 2.3 Temple cuántico . . . 15
  - 2.3.1 Temple cuántico y D-Wave . . . 16

3 Breve introducción al aprendizaje automático 18
- 3.1 Aprendizaje supervisado . . . 19
- 3.2 Aprendizaje no supervisado . . . 19

4 Modelos basados en la energía 20
- 4.1 Modelo de Ising . . . 20
- 4.2 Modelo de Hopfield . . . 22
- 4.3 Máquina de Boltzmann . . . 26
  - 4.3.1 Máquina de Boltzmann restringida . . . 30

5 Máquina de Boltzmann cuántica 32

6 Conclusiones 38

Referencias 39

1 Introducción

En el presente trabajo pretendemos estudiar y comprender cómo puede ayudar la teoría cuántica en tareas de aprendizaje automático, o machine learning en inglés. Si bien estas tareas están cada vez más integradas en las distintas tecnologías que existen hoy en día, su uso a gran escala se ve, en parte, limitado por la capacidad de cómputo de los dispositivos actuales. Distintos resultados en Física, Matemáticas y Computación, han demostrado que el procesamiento de información cuántica puede llegar a ser tremendamente más eficiente a la hora de realizar tareas específicas, como factorización o búsquedas en bases de datos. Este hecho ha motivado el estudio de posibles aplicaciones de la computación cuántica para tareas de aprendizaje automático, surgiendo así el campo del *aprendizaje automático cuántico*.

La filosofía seguida en este trabajo ha sido la de partir de los conceptos fundamentales de cada una de las disciplinas involucradas: Física Cuántica, Física Estadística y Aprendizaje Automático; para llegar construir, a partir de ellos, un modelo de aprendizaje automático cuántico puntero en la investigación actual en este campo: la Máquina de Boltzmann cuántica. Es por ello que la mayor parte del esfuerzo realizado ha sido, además de la tareas de revisión bibliográfica y análisis, la organización de conceptos de distintas disciplinas en un solo cuerpo lógico. La primera parte del trabajo la dedicamos al estudio de la computación cuántica, Sec. 2. Tras introducir el modelo estándar de computación, Subsec. 2.1, conocido como modelo de puertas, y el concepto de universalidad, pasamos a explicar el modelo de computación adiabática, Subsec. 2.2, y la equivalencia entre ambos. Será el modelo adiabático en el que nos centremos ya que, mediante ciertas consideraciones que veremos, Subsec. 2.3, nos permite implementar físicamente el algoritmo de aprendizaje automático cuántico llamado *Máquina de Boltzmann cuántica*, lo que podemos hacer en la actualidad en dispositivos como los de D-Wave, Subsec. 2.3.1. En la segunda parte introducimos brevemente el concepto de aprendizaje automático, Sec. 3. A continuación, tratamos un tipo especial de modelos de aprendizaje automático: los basados en la energía, Sec. 4. Construimos entonces una serie de modelos que pertenecen a esta categoría y que culminan en la *Máquina de Boltzmann*, Subsec. 4.3, y que es el más sofisticado de ellos. Una vez explicado dicho modelo clásico, y con ayuda de todo lo estudiado en la primera parte del trabajo sobre computación cuántica, pasamos a estudiar su análogo cuántico, Sec. 5. Veremos sus limitaciones y el estado de desarrollo en que nos encontramos en la actualidad. Acabamos con una breve discusión de los conceptos estudiados y de los posibles avances que esperamos se produzcan en este campo en los próximos años, Sec. 6.

2 Introducción a la computación cuántica

Aunque no existe una definición precisa acerca del significado de computación, entendemos que al hablar de ésta nos referimos a la realización de un cálculo u operación, sea o no aritmética, siguiendo una serie de instrucciones a las que nos referimos como algoritmo.

2.1 Modelo de puertas

El modelo de puertas o modelo de circuito es el modelo estándar de computación cuántica. Esto es así por su analogía con el modelo de computación clásica. Vamos a ver en este

apartado cómo se construye y los parecidos que existen entre éste y el modelo clásico, de manera que lleguemos a ver el paso de uno a otro como algo natural.

2.1.1 Qubits

Lo primero que necesitamos para construir un modelo de computación es hablar sobre cuál es la unidad de información en dicho modelo. En el caso clásico esta unidad es el bit, el cual puede encontrarse en dos estados distintos que solemos etiquetar como 0 y 1. En el caso cuántico a la unidad de información la conocemos como bit cuántico o qubit, por *quantum bit* en inglés. En el formalismo cuántico, asociamos al sistema físico un espacio de Hilbert complejo y separable, cuyos vectores unitarios representan los posibles estados del sistema. En el caso del qubit, el espacio de Hilbert correspondiente tiene dos dimensiones y su base ortonormal estándar, conocida como *base computacional*, viene dada por los vectores $\{|0\rangle, |1\rangle\}$. Los posibles estados del sistema son los vectores unitarios de la forma:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.1}$$

que pertenecen a dicho espacio de Hilbert y donde α y β son números complejos. La condición de unitariedad implica que $|\alpha|^2 + |\beta|^2 = 1$. Como podemos ver, este sistema se encuentra en una *superposición* de los estados $|0\rangle$ y $|1\rangle$ a diferencia del caso clásico donde, como hemos visto, el bit se encuentra o en un estado o en el otro. En general, un sistema formado por N qubits tendrá asociado un espacio de Hilbert complejo y separable de dimensión 2^N y cuya base computacional viene dada por el conjunto de 2^N vectores de N componentes siguiente $\{|00\ldots0\rangle, |00\ldots1\rangle, \ldots, |11\ldots0\rangle, |11\ldots1\rangle\}$.

2.1.2 Puertas cuánticas

Como hemos dicho al comienzo de la Sec. 2, la computación se basa en la realización de operaciones y cálculos mediante algún algoritmo. Realizar operaciones significa modificar la información de forma que, tras el conjunto de operaciones, ésta se haya transformado en la respuesta a nuestro problema. Por tanto, una vez que conocemos cuál es la unidad de información en computación cuántica, nos preguntamos cómo podemos modificarla y qué operaciones podemos realizar sobre ella.

x	NOT (x)
0	1
1	0

Tabla 1: Tabla de verdad de la puerta lógica clásica NOT. En la tabla, x representa el valor de un bit, mientras que NOT(x) es el valor del bit tras la operación NOT.

Puesto que estamos construyendo el modelo cuántico en analogía al clásico estudiemos, en primer lugar, en qué consisten las operaciones sobre bits. En el caso clásico, a las operaciones sobre un bit o sobre un conjunto de ellos las conocemos como *puertas lógicas*. Algunas de ellas son la puerta NOT, la puerta OR o la puerta NAND. Estas puertas pueden actuar, como decimos, sobre uno o más bits y quedan definidas mediante su tabla de verdad. Por ejemplo, en la Tabla 1, podemos ver la tabla de verdad de la puerta lógica NOT, que actúa sobre un único bit.

En el modelo de circuito del caso cuántico también vamos a modificar la información mediante puertas, aunque existen algunas diferencias con el caso clásico. Por la naturaleza cuántica de los estados de un qubit, la operación que las distintas puertas realizan sobre ellos no puede definirse mediante una tabla de verdad. Ahora, estas operaciones vienen definidas por operadores, U, lineales

$$U\left(\alpha|0\rangle + \beta|1\rangle\right) = \alpha U|0\rangle + \beta U|1\rangle$$

y unitarios

$$U^{\dagger}U = I$$

sobre el espacio de Hilbert de estados del sistema. Esto tiene dos consecuencias inmediatas: la primera es que la probabilidad se conserva; la segunda es que, a diferencia de lo que ocurre en el caso clásico, la acción de las puertas es *reversible*. Así, por ejemplo, si actuamos sobre el estado de un solo qubit de la Ec. (2.1) con la puerta *NOT cuántica*, esperamos que el resultado sea:

$$|\psi'\rangle = \alpha|1\rangle + \beta|0\rangle. \tag{2.2}$$

El operador cuántico definido sobre el espacio de estados de un qubit que implementa esta operación es

$$X \doteq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

Se escoge esta nomenclatura para la operación NOT cuántica puesto que, como vemos, su representación en la base computacional no es más que la matriz de Pauli σ_x. Si actuamos de nuevo con la puerta NOT cuántica pero ahora sobre el estado $|\psi'\rangle$, Ec. (2.2), obtenemos el estado $|\psi\rangle$ ya que $X|\psi'\rangle = X\left(X|\psi\rangle\right) = |\psi\rangle$. Esto es lo que conocemos como *reversibilidad*. Acabamos de ver un ejemplo sencillo de puerta cuántica sobre un qubit, pero existen numerosos ejemplos de otras puertas que son también relevantes, como la puerta *Hadamard* o la Z. En el caso de puertas que actúan sobre dos qubits también existe un amplio abanico de puertas catalogadas, entre las que cabe destacar la *CNOT*. Esta puerta actúa sobre la base computacional de la siguiente forma

$$\begin{array}{l} CNOT|00\rangle = |00\rangle \\ CNOT|01\rangle = |01\rangle \\ CNOT|10\rangle = |11\rangle \\ CNOT|11\rangle = |10\rangle \end{array}\ ',$$

por lo que su representación matricial en dicha base sobre el espacio de Hilbert de dos qubits es:

$$CNOT \doteq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Las distintas operaciones que hemos visto las hemos definido sobre el espacio de Hilbert del qubit o qubits sobre el que actúa dicho operador. Sin embargo, es posible definir estos operadores sobre el espacio de Hilbert asociado a un sistema de N qubits. Así, por ejemplo, definimos la operación cuántica NOT sobre el espacio de Hilbert asociado a un sistema de N qubits y actuando sobre el qubit i-ésimo como:

$$X^i = \sigma_x^i = \overbrace{\mathbf{I} \otimes \cdots \otimes \mathbf{I}}^{i-1} \otimes \sigma_x \otimes \overbrace{\mathbf{I} \otimes \cdots \otimes \mathbf{I}}^{N-i}, \tag{2.3}$$

donde

$$\mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Puesto que el número de operadores que podemos definir sobre el espacio asociado a un cierto número de qubits es infinito, nos podemos preguntar cuáles y, más importante, cuántos de estos operadores serán suficientes para implementar un algoritmo arbitrario, si es que acaso estas preguntas tienen respuesta. Además, cabe preguntarse si existen métodos de corrección de errores que sean efectivos independientemente del número de puertas distintas que se utilicen en el cómputo. Esta pregunta viene motivada porque en la transmisión de información siempre existe cierta probabilidad de que se produzcan errores. Aunque veremos la respuesta a esta pregunta y a las anteriores en el próximo apartado, Subsec. 2.1.3, no trataremos en el presente trabajo la naturaleza de los errores en computación cuántica y sus consecuencias. Un estudio ameno y detallado de la teoría de errores puede encontrarse en el libro [Nielsen y Chuang 2010].

2.1.3 Conjunto de puertas universal

Vamos a ver a continuación que existe una serie de resultados muy importantes en el modelo de puertas de computación cuántica que permiten responder a estas preguntas y dan esperanza acerca del uso de este tipo de computación fuera de los laboratorios en un futuro próximo. Los resultados son cuatro y se pueden resumir, casi esquemáticamente, en los siguientes:

1. Toda matriz unitaria puede ser escrita, de manera exacta, como el producto de operadores unitarios que actúan únicamente de forma no trivial sobre el subespacio generado por dos vectores de la base computacional.

2. Cada uno de estos operadores unitarios puede expresarse de manera exacta mediante una combinación de puertas *NOT*, definida sobre el espacio de Hilbert correspondiente a un único qubit, y puertas *CNOT* que, como dijimos, actúan sobre el espacio correspondiente a un sistema de dos qubits.

3. Cualquier operación unitaria que actúa sobre (el espacio asociado a) un único qubit puede ser escrita con precisión arbitraria mediante el conjunto finito de puertas siguiente: *Hadamard*, *fase* y $\pi/8$.

4. El Teorema de Solovay-Kitaev. A grandes rasgos, éste nos dice que, si un conjunto finito de puertas definidas sobre el espacio de Hilbert asociado a un qubit es capaz de aproximar cualquier otra puerta definida sobre el mismo espacio, entonces es posible realizar dicha aproximación con precisión[1] $\epsilon > 0$ usando para ello tan solo $O\left(\log^c\left(1/\epsilon\right)\right)$ puertas del conjunto, donde c es una constante próxima a 2.

Combinando los dos primeros resultados tenemos que cualquier operador unitario puede escribirse como combinación de operadores unitarios que actúan sobre el espacio asociado a un único qubit y puertas *CNOT*. Si a esto le añadimos lo enunciado en el tercer punto,

[1] Definimos la precisión como el valor máximo de la distancia, D, entre los operadores U y V, es decir, $D\left(U,V\right) \leq \epsilon$. A su vez, definimos la distancia entre dos operadores como dos veces el error, E, cometido al aproximar U mediante V, que es $E\left(U,V\right) = \max_{|\psi\rangle} \|\left(U-V\right)|\psi\rangle\|$, considerando el máximo sobre todos los estados puros $|\psi\rangle$.

vemos que podemos expresar cualquier operador unitario como un producto de puertas del conjunto: *CNOT*, *Hadamard*, *fase* y $\pi/8$. El conjunto de puertas escogido en el punto 3 no es el único que existe, aunque es quizás el más representativo por ser el más utilizado. Este resultado es realmente importante ya que demuestra que existe al menos un conjunto universal de puertas cuánticas, lo que quiere decir que haciendo uso únicamente de ellas somos capaces de realizar cualquier transformación o cómputo que necesitemos de forma aproximada. Además, como se puede demostrar [Nielsen y Chuang 2010, cap. 10], los métodos de corrección de errores que conocemos, y que hacen que la computación cuántica sea en principio una tecnología prometedora, tan solo son efectivos si el conjunto de puertas a utilizar es finito. Así, podemos comprender mejor la gran importancia de este resultado.

Sin embargo, aún nos falta un punto muy importante en este desarrollo y es hablar sobre la eficiencia de una computación realizada con un conjunto finito de puertas. En primera instancia podríamos pensar que, para describir ciertos operadores por estos medios, el número de puertas necesarias para conseguir una buena aproximación es enorme, siendo la convergencia muy lenta. En concreto, el conjunto de puertas que es posible definir sobre el espacio de Hilbert asociado a un qubit forma un continuo, por lo que tratar de aproximar todas estas operaciones mediante el conjunto finito dado en el punto 3 muy probablemente podría no ser eficiente. Aquí es donde entra en juego el maravilloso resultado de Solovay y Kitaev, [Dawson y Nielsen 2006], que recogemos en el punto 4 de la lista anterior. En términos muy simples, éste asegura que el número de puertas necesarias que pertenecen a dicho conjunto finito, el cual cumple ciertas condiciones razonables, para representar un operador arbitrario sobre el espacio asociado a un único qubit es de orden polilogarítmico, por lo que dicha representación es eficiente.

En resumen, hemos estudiado cuál es la unidad de información que manejamos en computación cuántica: los qubits. Hemos visto también que en el modelo estándar de computación cuántica, o modelo de circuito, la computación se realiza actuando mediante operadores lineales y unitarios, llamados puertas lógicas cuánticas, sobre el estado de los qubits. Así mismo, se han listado una serie de resultados cuya consecuencia es la universalidad de un conjunto finito de puertas que permite, además, tanto realizar esta computación de manera eficiente como la incorporación de técnicas para su resiliencia ante los errores. Una visión más completa y precisa sobre este modelo de computación, y en concreto en lo concerniente a la teoría de errores, puede encontrarse en el texto de referencia [Nielsen y Chuang 2010], el cual ha sido también la principal guía para la elaboración de esta Subsec. 2.1.3.

2.2 Modelo adiabático

Una vez estudiado el modelo estándar, pasamos ahora al estudio del modelo adiabático de computación cuántica. Éste se propuso en un inicio como modelo para resolver problemas de *satisfacibilidad* [2], aunque desde entonces se ha demostrado particularmente útil en problemas combinatorios en los que estamos interesados en encontrar la solución óptima, debido a la facilidad de reformular estos problemas en términos de un Hamiltoniano.

Partimos del resultado teórico que permitió vislumbrar este modelo de computación: el Teorema Adiabático. Continuamos dando una definición formal acerca de lo que significa

[2] Un problema de *satisfacibilidad* es aquel cuya solución pasa por saber si existen valores para las variables de cierta expresión lógica que hagan que dicha expresión sea verdadera.

computación en términos del modelo adiabático. Al final vemos que los dos modelos de computación estudiados son equivalentes, para lo que realizamos una demostración parcial de dicha equivalencia, simulando eficientemente un cómputo arbitrario del modelo de circuito mediante el modelo adiabático.

2.2.1 Teorema adiabático

Como decimos, el teorema fundamental en que este modelo está basado es el *Teorema Adiabático* (TA). Dicho teorema, en términos generales, establece lo siguiente [Chirkov 2001]:

- Un sistema físico permanecerá en su autoestado instantáneo si la perturbación, $H(t)$, que actúa sobre dicho sistema varía lo suficientemente despacio y si la diferencia de energía entre este autoestado y cualquier otro del espectro del Hamiltoniano es distinta de cero durante toda la evolución del sistema.

En particular, podemos preparar el sistema en el estado propio de mínima energía, o estado fundamental, de dicha perturbación en el instante inicial. De esta forma, y si se cumple la condición de adiabaticidad (que veremos más adelante), en el instante final el sistema se encontrará en el estado fundamental de la perturbación final. En el artículo [Farhi y col. 2000] se realiza una introducción matemática sencilla al TA, que es la que vamos a seguir en las siguientes líneas.

Un sistema cuántico que se ve afectado por una perturbación dependiente del tiempo evoluciona según la ecuación de Schrödinger

$$i\frac{d}{dt}|\psi(t)\rangle = H(t)|\psi(t)\rangle, \tag{2.4}$$

siendo $|\psi(t)\rangle$ el estado del sistema y $H(t)$ la perturbación dependiente del tiempo. Consideremos, como en el citado artículo, un Hamiltoniano que varía de forma suave y depende de un único parámetro, s, que toma valores en el rango $0 \leq s \leq 1$. Entonces, si el sistema evoluciona durante un tiempo total t_f, podemos escribir $t = t_f s$. Los estados y valores propios instantáneos del Hamiltoniano se definen como:

$$H(s)|l;s\rangle = E_l(s)|l;s\rangle,$$

donde los autovalores son $E_0(s) \leq E_1(s) \leq ... \leq E_{N-1}(s)$ y donde N es la dimensión del espacio de Hilbert asociado al sistema. De todos los estados propios vamos a trabajar con el de mínima energía, $|l = 0;s\rangle$. Aunque el resultado que obtenemos es independiente del autoestado que escojamos, veremos más adelante por qué es interesante éste en particular. En el instante inicial, $s = 0$, el autoestado de mínima energía del Hamiltoniano es:

$$|\psi(s = 0)\rangle = |l = 0; s = 0\rangle$$

y, si se cumple que la diferencia de energía entre este estado y el primer estado excitado es no nula para $0 \leq s \leq 1$, condición necesaria del TA, tenemos que

$$\lim_{t_f \to \infty} \left|\langle l = 0; s = 1|\psi(t_f)\rangle\right| = 1.$$

Esto quiere decir que, si la evolución es lo suficientemente lenta, el estado final del sistema coincidirá con el estado de mínima energía del Hamiltoniano final. El *gap*[3] mínimo, $g_{\text{mín}}$, se define como

$$g_{\text{mín}} = \min_{0\leq s\leq 1} \left(E_1(s) - E_0(s)\right),$$

lo que nos lleva a la siguiente condición de adiabaticidad [Amin 2009]:

$$t_f \gg \varepsilon / g^2_{\text{mín}}, \tag{2.5}$$

donde

$$\varepsilon = \max_{0\leq s\leq 1} \left| \langle l=1; s | \frac{dH}{ds} | l=0; s \rangle \right|.$$

De esta forma,

$$\left| \langle l=0; s=1 | \psi(t_f) \rangle \right|$$

puede ser tan próximo a la unidad como se quiera.

Aunque para comprender los desarrollos posteriores que se realizan en este trabajo es suficiente con la condición vista para t_f en la Ec. (2.5), puede resultar interesante para algún lector estudiar algunos de los resultados y distintas formulaciones del Teorema Adiabático que existen. Entre ellos, mencionar el extenso trabajo de [Albash y Lidar 2018], así como el trabajo original de [Kato 1950]. En el primero de ellos se encuentran cotas precisas para t_f, aunque éstas vienen acompañadas de restricciones más fuertes sobre la forma del Hamiltoniano y su espectro. Es por ello que, en la práctica, el resultado aproximado que hemos estudiado es suficiente.

2.2.2 Computación cuántica adiabática

Una vez estudiado el Teorema Adiabático, es conveniente definir de manera formal el concepto de computación adiabática. Como vamos a ver, en esta definición no hacemos mas que explicitar la idea de computación que hemos podido intuir a partir del TA. Con este objetivo vemos, en primer lugar, el concepto de *Hamiltoniano local*.

- *Definición* (**Hamiltoniano Local en** k): Un Hamiltoniano se dice que es local en k si se puede escribir como suma de Hamiltonianos cada uno de ellos actuando de forma no trivial, como máximo, sobre el espacio de Hilbert asociado a un sistema de k partículas.

Definimos ahora el concepto de *computación adiabática*. En nuestro caso, escogemos la definición dada por [Aharonov y col. 2008], basada en la de [Farhi y col. 2000], que condensa en unas pocas líneas todas las ideas necesarias. Dicha definición es la siguiente:

- *Definición* (**Computación Adiabática**): Una computación, $CA(n, d, H_{init}, H_{final}, \epsilon)$, se dice que es adiabática y local en k si:
  - H_{init}, H_{final} son dos Hamiltonianos k locales que actúan sobre n sistemas de d estados cada uno, teniendo cada Hamiltoniano un único estado fundamental, de forma que $H(t=0) = H_{init}$ y $H(t=t_f) = H_{final}$. El Hamiltoniano $H(t)$ es la perturbación dependiente del tiempo que sufre el sistema, Ec. (2.4).

[3] El *gap* es la diferencia entre las autoenergías de dos estados propios del Hamiltoniano del sistema. En este caso, entre la del estado fundamental y la del primer nivel excitado.

- El estado fundamental de H_{init} es un estado producto tensorial.
- El estado final de la computación es ϵ-próximo según la norma l_2[4] al estado fundamental de H_{final}.

En la práctica, tal y como veremos más adelante, los sistemas que utilizamos para computación están formados por qubits, por lo que $d = 2$. En cuanto a la dependencia funcional de $H(s = t/t_f)$ con H_{init} y H_{final}, podemos escribir ésta de manera general como:

$$H(s) = A(s)H_{init} + B(s)H_{final}\,, \tag{2.6}$$

donde $A(s)$ y $B(s)$ son funciones paramétricas en s que cumplen: $A(s = 0) = 1$, $A(s = 1) = 0$; $B(s = 0) = 0$, $B(s = 1) = 1$.

2.2.3 Sobre la universalidad en computación cuántica adiabática

Una vez que hemos estudiado los fundamentos de la computación cuántica adiabática, necesitamos conocer su capacidad a la hora de resolver problemas. En el caso de la computación mediante el modelo de puertas hemos visto que, si bien en un principio parecía inabarcable la implementación de todas las posibles operaciones que se pueden realizar sobre el estado de un sistema de qubits, existe un conjunto finito de puertas que permite aproximar cualquier operación de manera eficiente. A este conjunto lo hemos llamado conjunto universal de puertas para computación cuántica. En el caso de la computación cuántica adiabática (CCA), el cómputo no se realiza mediante puertas lógicas que evolucionan el sistema, sino que la evolución de éste la rige un Hamiltoniano dependiente de tiempo con las características vistas en la Subsec. 2.2.2. El resultado del cómputo en este modelo es, idealmente, el estado fundamental de dicho Hamiltoniano en el instante final. Así, la pregunta acerca de la universalidad en CCA se traduce en conocer en qué condiciones, si las hubiera, un Hamiltoniano dependiente del tiempo permite aproximar cualquier operación sobre el estado de un sistema cuántico de manera eficiente. A dicho Hamiltoniano lo consideramos universal para computación cuántica adiabática. La demostración de que dicho Hamiltoniano existe pasa por demostrar que un cómputo mediante el modelo adiabático permite obtener, de manera eficiente, el mismo resultado que utilizando el modelo de puertas, y viceversa. La definición formal y funcional de universalidad se particulariza para este tipo de computación cuántica de la forma siguiente [Albash y Lidar 2018]:

- *Definición* (**Hamiltoniano universal para CCA**): Un Hamiltoniano, $H(t)$, dependiente del tiempo, con t en el intervalo $[0, t_f]$, es universal para computación cuántica adiabática si:
  - Dado un circuito cuántico arbitrario U que actúa sobre algún estado $|\psi\rangle$ de n partículas, de d estados cada una, y que tiene profundidad[5] L, el estado fundamental de $H(t_f)$ es igual a $U|\psi\rangle$ con probabilidad mayor que ϵ.
  - El número de partículas sobre las que actúa $H(t)$ de forma no trivial, en cada instante de tiempo, es polinómico[6] en el número de partículas, $poly(n)$ $\forall t$.

[4] Que un estado sea ϵ-próximo según la norma l_2 a otro quiere decir que la norma l_2 de la diferencia entre ambos estados es menor o igual a ϵ, con $\epsilon > 0$.

[5] Se dice que un circuito cuántico tiene profundidad L cuando el mayor número de operaciones que actúa sobre cualquiera de los qubits es L.

[6] Cuando una magnitud es de cierto orden funcional respecto a otra significa que el crecimiento de la

- t_f es polinómico tanto en el número de partículas como en la profundidad del circuito, t_f es $poly(n, L)$.

La demostración de que dicho Hamiltoniano existe pasa, como hemos dicho, por demostrar la equivalencia entre este modelo de computación y el modelo de puertas. Puesto que no queremos extendernos en exceso en este aspecto, a continuación tan solo vamos a demostrar que mediante el modelo adiabático podemos obtener eficientemente el mismo resultado que mediante el de puertas. Hacer esto nos va a permitir, además, comprender algo mejor este paradigma de computación a través de su funcionamiento.

Simulación eficiente del modelo de puertas mediante CCA

Nuestro objetivo es entonces demostrar que, partiendo de un mismo estado en los dos casos, el resultado de un cómputo mediante el modelo adiabático es el mismo que el que se obtendría si lo realizásemos mediante el modelo de puertas. El método que empleamos para realizar dicha demostración se lo debemos a [Aharonov y col. 2008] quien, utilizando un resultado previo de [A. Y. Kitaev, Shen y Vyalyi 2002], fue capaz de construir un Hamiltoniano, $H(s)$, que permite demostrar que la CCA es capaz de simular el modelo de puertas.

Supongamos que el circuito que queremos simular consta de L puertas cuánticas unitarias, $\{U_1, U_2, ..., U_L\}$, definidas sobre el espacio de Hilbert asociado a un sistema de n qubits. En primer lugar se inicializa el sistema, sin pérdida de generalidad, en cierto estado particular $|\alpha(0)\rangle$. A medida que las distintas puertas actúan sobre el sistema su estado se transforma, de manera que, tras la l-ésima puerta, su estado es $|\alpha(l)\rangle$. Es en este momento cuando utilizamos la herramienta que nombramos antes, desarrollada por Kitaev, la cual nos permite construir un Hamiltoniano cuyo estado fundamental contiene la historia de la evolución del sistema. Dicho estado fundamental es

$$|\eta\rangle = \frac{1}{\sqrt{1+L}} \sum_{l=0}^{L} |\gamma(l)\rangle \,, \tag{2.7}$$

donde

$$|\gamma(l)\rangle \equiv |\alpha(l)\rangle \otimes |1^l 0^{L-l}\rangle_c \,.$$

El estado $|1^l 0^{L-l}\rangle_c$ es el estado de un sistema al que conocemos como reloj (*clock* en inglés, de ahí el subíndice) de Feynman y cuya utilidad es la de llevar un registro del número de puertas que han actuado sobre el sistema. Dicho sistema está compuesto por L qubits, indicando cada superíndice cuántos de ellos se encuentran en el estado $|1\rangle$ y en el $|0\rangle$. De esta forma, si el estado del reloj es $|1^3 0^{L-3}\rangle_c$ quiere decir que los tres primeros qubits del reloj se encuentran en el estado $|1\rangle$ y los $L-3$ restantes en estado $|0\rangle$. En el instante inicial este reloj es inicializado en el estado $|0^L\rangle$. Cuando actúa la primera puerta, U_1, sobre el sistema, el primer qubit del registro pasa al estado $|1\rangle$ y el resto se mantienen en $|0\rangle$, siendo entonces el estado del reloj $|1^1 0^{L-1}\rangle_c$. Cada vez que actúa una puerta, el estado del primer qubit del registro por la izquierda que se encuentre en estado $|0\rangle$ pasa al estado $|1\rangle$.

Como dijimos antes, nuestro objetivo es la construcción de un Hamiltoniano cuyo estado fundamental coincida con el estado final del circuito cuántico que queremos simular. Sin

primera condicionado al de la segunda es siempre inferior o igual a de la relación funcional dada. Por ejemplo, en caso que dirige a esta nota, el número de partículas sobre las que actúa el sistema, en cada instante de tiempo, nunca será mayor que el de cualquier polinomio de n.

embargo, el Hamiltoniano que construimos mediante este método es aquel cuyo estado fundamental es el de la Ec. (2.7). ¿Tiene esto sentido? Pues de hecho sí que lo tiene ya que, si lo observamos con detenimiento, podemos darnos cuenta de que el último sumando, $\frac{1}{\sqrt{1+L}}|\gamma(L)\rangle$, de la superposición es, salvo una constante, el estado final que devuelve el circuito. Debido precisamente a que éste no es más que uno de los sumandos, la probabilidad de obtener el estado que buscamos a partir de $|\eta\rangle$ es $1/(L+1)$. Podemos comprobar que efectivamente el estado obtenido coincide con el que se obtendría mediante el modelo de circuito cerciorándonos de que el estado del registro es $|1^L\rangle_c$. La elección de este estado que contiene información sobre toda la evolución del sistema es debida a que la construcción del Hamiltoniano es sencilla en este caso y a que ya existe un desarrollo teórico que nos dice cómo hacerlo. Veamos ahora cómo se construye dicho Hamiltoniano.

En primer lugar, particularizamos el Hamiltoniano $H(s)$ dependiente del tiempo de la Ec. (2.6) para $A(s) = 1 - s$ y para $B(s) = s$, lo que queda:

$$H(s) = (1-s)H_{init} + sH_{final}, \tag{2.8}$$

El valor particular escogido para $A(s)$ y $B(s)$ no socava la generalidad de la demostración. Expresamos ahora los Hamiltonianos inicial y final como:

$$\begin{aligned} H_{init} &= H_{clockinit} + H_{input} + H_{clock} \\ H_{final} &= \frac{1}{2}\sum_{l=1}^{L} H_l + H_{input} + H_{clock}\,, \end{aligned} \tag{2.9}$$

donde vamos a definir cada uno de los términos de manera que el único estado propio de H_{init} con autovalor 0 sea $|\gamma(0)\rangle$ y el único con autovalor 0 de H_{final} sea $|\eta\rangle$. Sustituyendo cada término de la Ec. (2.9) en la expresión del Hamiltoniano dependiente del tiempo, Ec. (2.8), tenemos:

$$H(s) = H_{input} + H_{clock} + (1-s)H_{clockinit} + \frac{s}{2}\sum_{l=1}^{L} H_l\,. \tag{2.10}$$

Estudiamos a continuación la forma de cada uno de los términos:

- H_{clock}: Este término se encarga de comprobar que el estado del reloj es de la forma $|1^l 0^{L-l}\rangle$. Tiene la siguiente forma:

  $$H_{clock} = \sum_{l=1}^{L-1} |01\rangle\langle 01|^c_{l,l+1},$$

  donde el subíndice indica sobre qué qubits, del reloj en este caso, actúa el proyector. La forma de H_{clock} implica que, en caso de que uno de los términos del reloj no tenga la forma adecuada, éste tendrá una energía mayor de cero, "penalizando" entonces a dicho estado.

- $H_{clockinit}$: Su propósito es asegurarse de que el estado inicial del reloj es $|0^L\rangle_c$. Para ello nos basta comprobar que el primer qubit se encuentra en el estado cero, ya que el resto de la comprobación la realiza H_{clock}. La forma de este término es

  $$H_{clockinit} = |1\rangle\langle 1|^c_1.$$

- H_{input}: El cometido de este término es el de asegurar que en el instante inicial, para el que el estado del reloj es $|0^L\rangle_c$, el estado de los n qubits computables es $|0^n\rangle$. La forma de este término es, por tanto:

$$H_{input} = \sum_{i=1}^{n} |1\rangle\langle 1|_i \otimes |0\rangle\langle 0|_1^c$$

  donde, al igual que ocurría para el anterior término, basta con comprobar que el primer qubit del reloj se encuentra en el estado $|0\rangle$.

- H_l: Su cometido es asegurar que la evolución del paso $l-1$ a l se realiza correctamente. La forma de este término depende del paso en el que nos encontremos. Así, tenemos que para $l = 1$:

$$\begin{aligned} H_1 = \mathbf{I} \otimes |00\rangle\langle 00|_{1,2}^c - U_1 \otimes |10\rangle\langle 00|_{1,2}^c \\ - U_1^\dagger \otimes |00\rangle\langle 10|_{1,2}^c + \mathbf{I} \otimes |10\rangle\langle 10|_{1,2}^c \,, \end{aligned}$$

  para $1 < l < L$:

$$\begin{aligned} H_{1<l<L} = \mathbf{I} \otimes |100\rangle\langle 100|_{l-1,l,l+1}^c - U_l \otimes |110\rangle\langle 100|_{l-1,l,l+1}^c \\ - U_l^\dagger \otimes |100\rangle\langle 110|_{l-1,l,l+1}^c + \mathbf{I} \otimes |110\rangle\langle 110|_{l-1,l,l+1}^c \,, \end{aligned}$$

  y para $l = L$:

$$\begin{aligned} H_L = \mathbf{I} \otimes |10\rangle\langle 10|_{L-1,L}^c - U_L \otimes |11\rangle\langle 10|_{L-1,L}^c \\ - U_L^\dagger \otimes |10\rangle\langle 11|_{L-1,L}^c + \mathbf{I} \otimes |11\rangle\langle 11|_{L-1,L}^c \,. \end{aligned}$$

De esta forma, conocemos por completo el Hamiltoniano, Ec. (2.10), que necesitamos para realizar la computación adiabática. La demostración de que, efectivamente, $|\gamma(0)\rangle$ y $|\eta\rangle$ son los estados propios, con valor propio 0, del Hamiltoniano que hemos construido en el instante inicial y final, respectivamente, es sencilla y la puede encontrar el lector que tenga interés en [Albash y Lidar 2018]. Como dijimos, la probabilidad de obtener el estado $|\alpha(L)\rangle$ a partir de $|\eta\rangle$ es $1/(L+1)$, lo que introduce un factor L en el tiempo de cómputo del caso adiabático, siendo un claro inconveniente. A pesar de ello, existen métodos relativamente sencillos, como añadir una serie de puertas identidad al final del circuito [Aharonov y col. 2008], que consiguen que $|\eta\rangle$ sea próximo a $|\alpha(L)\rangle$. De esta forma, la probabilidad de obtener el segundo a partir del primero aumenta, y la diferencia en tiempo de cómputo utilizando cada modelo se mantiene polinómica en el número de puertas, L.

A lo largo de la Subsec. 2.2, hemos introducido el modelo de computación cuántica adiabática. Para ello, hemos estudiado el Teorema Adiabático, concepto teórico fundamental sobre el que se construye ésta, lo que nos ha permitido comprender después la definición formal de CCA. Ha sido entonces cuando hemos investigado acerca de la universalidad en este modelo. A diferencia de lo que ocurría en el modelo de circuito, ahora la universalidad depende de la existencia de cierto Hamiltoniano, definido sobre el espacio de Hilbert de un sistema de qubits, que permita realizar cualquier operación sobre el estado de estos de manera eficiente. Además, el estado fundamental de dicho operador al final del proceso adiabático debe coincidir con el que obtenemos al realizar el cómputo mediante el modelo de circuito. En la Subsec. 2.2.3 demostramos que podemos construir dicho Hamiltoniano,

lo que constituye la primera parte de la equivalencia entre ambos modelos de computación. Si bien el modelo adiabático supone una nueva forma de, quizás en un futuro, conseguir un ordenador cuántico de propósito general funcional, en la actualidad cumplir la condición de adiabaticidad, Ec. (2.5), implica que el estado del sistema, al interactuar éste con el medio durante tiempos largos, deje de ser cuántico. Así, el compromiso entre ambas restricciones limita la posibilidad de implementar un computador cuántico adiabático. A pesar de esto, vamos a ver que es posible aprovechar estas condiciones en nuestro beneficio, ya que nos permiten construir un modelo capaz de resolver ciertos problemas de forma especialmente eficiente.

2.3 Temple cuántico

Una vez que hemos estudiado el modelo adiabático de computación, nos encontramos en una buena posición para comprender en qué consiste el *temple cuántico*, o *quantum annealing* en inglés. Éste no es más que una heurística para encontrar el mínimo global de cierta función, a la que conocemos como *función objetivo*.

Si bien el temple cuántico parte del fundamento teórico de la computación adiabática, éste no es un modelo de computación al uso. Esencialmente las diferencias con el modelo adiabático son dos. Por un lado relajamos la *condición de adiabaticidad* y, por otro, debemos tener en cuenta las características físicas del dispositivo con el que realizamos la computación, puesto que el sistema es, en realidad, abierto y va a interaccionar con el medio. Si recordamos lo estudiado sobre computación adiabática, en el caso en que la evolución del sistema no era lo suficientemente lenta no podíamos asegurar que el estado final del sistema fuera el estado fundamental del Hamiltoniano $H(s=1)$, con $s \in [0,1]$. Aún así, esperamos que la *rutina de ejecución* que diseñemos nos permita obtener un estado muy cercano al fundamental. Con rutina de ejecución nos referimos a la manera en que el Hamiltoniano que rige la evolución del sistema, $H(s)$, se relaciona con los Hamiltonianos inicial y final, H_{init} y H_{final} respectivamente. Para obtener el valor óptimo de la *función objetivo* necesitaremos reformular el problema [Lucas 2014] de forma que su solución se corresponda con el estado fundamental de H_{final}. Al igual que en el modelo adiabático, Ec. (2.6), la forma general de $H(s)$ en el modelo de temple cuántico es:

$$H(s) = A(s)H_{init} + B(s)H_{final},$$

donde $A(s)$ y $B(s)$ son funciones paramétricas en s tales que $A(s=0) \gg B(s=0)$ y $A(s=1) \ll B(s=1)$, siendo $0 \leq s \leq 1$. Será al tratar con el problema que queramos resolver cuando tengamos que encontrar la forma apropiada para $A(s)$ y $B(s)$, tal y como veremos al final del trabajo al estudiar la Máquina de Boltzmann cuántica.

Otra característica interesante sobre el temple cuántico tiene que ver con la clase de Hamiltoniano que gobierna la evolución del sistema. Dicha clase determina el tipo de problema, en cuanto a su complejidad, que podemos resolver. Por ejemplo, un Hamiltoniano de Ising, cuya forma estudiaremos en la Subsec. 2.3.1, permite resolver problemas de tipo NP-completo [Kempe, A. Kitaev y Regev 2006; Bravyi y col. 2008]. NP[7] y NP-completo son algunas de las categorías en que la teoría de la complejidad computacional divide los problemas. Una de las clases más relevantes de Hamiltonianos para el modelo que pretende-

[7] El conjunto de problemas NP está formado por aquellos problemas cuya solución se puede comprobar en tiempo polinómico.

mos construir la constituye la conocida como *estocuástica*[8]. La definición de Hamiltoniano estocuástico es la siguiente [Marvian, Lidar y Hen 2019]:

- *Definición* (**Hamiltoniano estocuástico**): un Hamiltoniano local, $H = \sum_{a=1}^{M} H_a$, se dice que es estocuástico respecto a cierta base B si en cada término local H_a los elementos fuera de la diagonal son no positivos.

Esta distinción surge debido a que sabemos que los *no estocuásticos* presentan un problema conocido como *problema del signo*. Éste no es más que la ineficiencia de ciertos algoritmos, como los de Monte Carlo cuánticos, al trabajar con dicha clase de Hamiltonianos. Estos algoritmos evalúan el valor medio de observables físicos, para lo que muestrean[9] el espacio de configuraciones cuánticas mediante la descomposición de la función de partición en una suma de términos más sencillos de calcular, que se interpretan como probabilidades en un proceso de Markov. En los casos en que la descomposición contiene términos negativos, estos algoritmos tienden a converger exponencialmente despacio. Además de este motivo, a día de hoy, la única realización física de un sistema capaz de llevar a cabo la heurística del temple cuántico tan solo permite trabajar con Hamiltonianos estocuásticos. Es por estas causas que dicha clasificación de Hamiltonianos es especialmente relevante. Puesto que la clase de problemas que podemos resolver se ve determinada por el Hamiltoniano que podemos utilizar, la posibilidad de una computación universal utilizando el proceso de temple cuántico se desvanece. Sin embargo, y como veremos, aún podemos tratar de resolver distintos problemas [Troyer y Wiese 2005] haciendo uso de este modelo.

Como dijimos en la Introducción, uno de los objetivos de este trabajo es estudiar la Máquina de Boltzmann cuántica. Veremos más adelante que, para su implementación, necesitamos calcular el valor esperado de ciertos operadores, lo que conseguimos mediante el muestreo sobre cierta distribución de probabilidad. En el caso de la Máquina de Boltzmann la distribución con la que trabajamos es la distribución de Boltzmann. Entre las posibilidades que ofrece el temple se encuentra la de realizar un muestreo de la distribución de estados del Hamiltoniano H_{final} utilizado. Esta posibilidad surge de manera natural en el caso del temple cuántico gracias, precisamente, a que la evolución es no adiabática. Esto implica que existe una probabilidad no nula de que el estado final del sistema no sea el fundamental de H_{final}, sino alguno de los estados excitados. Realizando el temple un número suficiente de veces, podemos caracterizar la distribución que nos interesa a partir de las muestras obtenidas y calcular así el valor esperado. En todo caso, debemos ser capaces de conseguir que la distribución de estados del Hamiltoniano al final del temple sea la correcta, ya que en otro caso las muestras que obtenemos no nos son de utilidad. Estudiaremos en mayor detalle las condiciones en las que esto ocurre, para el caso particular de una distribución de Boltzmann, en la Sec. 5.

2.3.1 Temple cuántico y D-Wave

En la tarea de implementar el temple cuántico, la empresa D-Wave Systems Inc. [s.f.] es pionera. En la actualidad cuentan con un dispositivo que implementa una arquitectura con 2000 qubits, funcionando a una temperatura de 15 mK aproximadamente.

[8] El término *estocuástico* lo he utilizado como adaptación del término inglés *estoquastic*, que se ha creado también para esta definición. Pretende unir los términos *estocástico* y *cuántico* bajo un solo nombre.

[9] Llamamos *muestrear* al proceso por el cual cierto algoritmo obtiene muestras de cierta distribución de probabilidad con el objetivo de caracterizarla.

La aproximación física que esta empresa ha seguido para la construcción de los sistemas y dispositivos necesarios es la de una electrónica de superconductores, en la que la resistencia del conductor es nula, de ahí que sea necesario mantener una temperatura tan próxima al cero absoluto. Existen otras propuestas tecnológicas para la construcción de computadores que implementen el temple cuántico, como las trampas de iones [Zhang y col. 2018], pero la electrónica de superconductores es la que lidera este campo actualmente. Los qubits los implementan mediante circuitos en forma de espira, Fig. 1, cerrados mediante uniones Josephson que permiten, con cierta probabilidad, el paso de los electrones [Johnson y col. 2011]. De esta forma, el flujo magnético que produce la corriente debida a estos electrones, y que circula en ambos sentidos por el circuito superconductor, se encuentra cuantizado en dos estados [Grant y Humble 2020], por lo que lo podemos considerar un qubit.

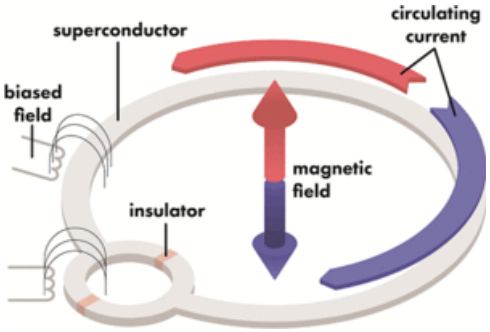

Figura 1: Representación esquemática del dispositivo físico utilizado por la empresa D-Wave como qubit. Dicho dispositivo está formado por un superconductor en forma de espira cerrado por una unión de tipo Josephson. Esto permite obtener un flujo magnético cuantizado en dos niveles. El dispositivo consta, además, de sistemas de control con los que aplicar campo externo, un sistema para medir el estado del qubit y de canales para conectar éste con otros qubits del sistema. Fuente: [Grant y Humble 2020].

El circuito dispone de un sistema para aplicar un campo magnético a cada qubit por separado, de forma que el perfil de energía del sistema de qubits puede ser externamente modificado. Así mismo, los qubits pueden afectarse entre sí mediante interferencia magnética, de manera que la interacción entre ellos es matemáticamente idéntica al entrelazamiento cuántico, al menos en lo que respecta a la computación. Por último, también dispone de un dispositivo capaz de medir el estado del qubit una vez finalizada la evolución.

Puesto que no es el objetivo de este trabajo el analizar la física del sistema en un nivel de análisis inferior al de qubit, remitimos al lector interesado en ésta al trabajo [Wendin 2017], en el que se estudia en detalle tanto la unión Josephson como otros aspectos del procesamiento de información mediante circuitos superconductores.

Como hemos expuesto en el punto anterior, la elección del tipo de Hamiltoniano que rige la evolución del sistema es de vital importancia ya que determina la clase de problemas que vamos a poder resolver. En el caso de D-Wave, el Hamiltoniano que ellos implementan físicamente es el siguiente [D-Wave 2020]:

$$H_{Ising-transversal} = -A(s)\left(\sum_i \sigma_x^i\right) + B(s)\left(\sum_i k_i\sigma_z^i + \frac{1}{2}\sum_{i,j} J_{i,j}\sigma_z^i\sigma_z^j\right), \tag{2.11}$$

donde $\sigma_{x,z}^i$ son las matrices de Pauli que actúan sobre el qubit i-ésimo, como vimos en la Ec. (2.3), k_i es el *sesgo* del qubit i-ésimo y $J_{i,j}$ el *acoplamiento* de los qubits i y j. El primer sumando es el *Hamiltoniano inicial* y el segundo el *Hamiltoniano problema* el cual, por su forma, es un Hamiltoniano de Ising. Ya vimos que entonces la clase de problemas que puede

ayudar a resolver es la de complejidad NP-completos. Además, dicho Hamiltoniano es trivialmente estocuástico lo que permite, hasta cierto punto, realizar una simulación de este cómputo con métodos Monte Carlo cuánticos y compararlos con los obtenidos mediante las máquinas de D-Wave. Hablaremos más sobre él en la segunda parte del trabajo.

Hemos visto en qué consiste el temple cuántico y en qué tipo de problemas permite resolver: problemas de optimización y aquellos de muestreo de cierta distribución de probabilidad, siendo el segundo tipo el que utilizaremos más adelante en este trabajo. Hemos estudiado también la implementación particular que realiza la empresa D-Wave.

3 Breve introducción al aprendizaje automático

La temática del presente trabajo se encuadra, como dijimos, entre dos ramas del conocimiento: la Física y la Computación. Cada una de ellas tiene algo que aportar a la discusión y ambas deben tener en cuenta los planteamientos y descubrimientos realizados por la otra. No deben entenderse como puntos de vista enfrentados sino complementarios. Por un lado, en Física se han construido una serie de modelos, como el de qubit o el modelo adiabático, con los que podemos trabajar gracias a que cuentan con un formalismo matemático/físico sólido que lo respalda, el formalismo cuántico. Por otro lado, los científicos en computación han sabido aprovechar estos resultados teóricos básicos y construir a partir de ellos computadores como los de D-Wave, dispositivos con el propósito concreto de resolver problemas de optimización o de muestreo. Ya que, hasta ahora, tan solo hemos visto el punto de vista físico, a continuación vamos a estudiar una de las utilidades más relevantes que tiene el resolver este tipo de problemas y que, desde hace unos años, ha ganado una enorme popularidad por la calidad y novedad de los resultados que encuentra: el *aprendizaje automático*.

Como todo aquello que se relaciona con la inteligencia artificial y que se vuelve tan popular, el aprendizaje automático se encuentra rodeado de cierto misticismo. Sin embargo, a pesar de haber ocupado el imaginario popular y ser transfigurado por éste, el aprendizaje automático no es más que una clase de algoritmos cuyo objetivo es el de construir un modelo matemático de un conjunto de datos.

Dentro del aprendizaje automático, la clasificación por excelencia que existe entre los algoritmos es la de *aprendizaje supervisado* y *aprendizaje no supervisado*. Realizamos esta distinción atendiendo a los recursos de los que dispone el algoritmo para construir el modelo [Burkov 2019]. En concreto, distinguimos entre aprendizaje supervisado y no supervisado dependiendo de si cada ejemplo de nuestro conjunto de datos cuenta con una etiqueta que lo identifica. Por ejemplo, si tenemos un conjunto que haga referencia a personas, cada instancia de dicho conjunto puede contar con tres *características*, como la Altura, el Peso y la Edad, y contar, o no, con una *etiqueta* que defina dicha instancia, como podría ser el Sexo de la persona. Si se dispone de la etiqueta para construir el modelo, trabajaremos con algoritmos de aprendizaje supervisado; en caso contrario, tendremos que recurrir a los de aprendizaje no supervisado. Veamos las principales características de cada clase con algo más de detalle.

3.1 Aprendizaje supervisado

Como hemos dicho, esta clase comprende a los algoritmos que trabajan con una conjunto de datos que están etiquetados. La estructura de los datos es $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$. El índice i sirve para identificar el ejemplo o *instancia* dentro del conjunto, habiendo un total de N ejemplos, Tabla 2. Además, cada uno de estos se compone de dos elementos fundamentales: $\mathbf{x}_i$ e y_i. El primero es un vector, llamado *vector de características*. Cada una de las componentes de este vector es una característica que da cierta información acerca del ejemplo i que describe. Por ejemplo, la componente j-ésima, $x^{(j)}$, puede ser la altura en cm de una persona, por seguir con el ejemplo propuesto en la introducción de la Sec. 3. El segundo elemento, y_i, es la *etiqueta* a la que nos hemos estado refiriendo. Ésta puede ser tanto un elemento de un conjunto finito de clases $\{C_1, C_2, ..., C_s\}$, siendo s el número de clases distintas, o ser un número real.

Índice	Altura (cm)	Peso (kg)	···	Edad (años)	Sexo
1	181	84	···	34	Hombre
2	165	62	···	27	Mujer
···	···	···	···	···	···
N	176	73	···	64	Hombre

Tabla 2: Ejemplo de un posible conjunto de datos sobre el que aplicar técnicas de aprendizaje automático. Cada una de las líneas de la tabla es una instancia del conjunto. Éstas poseen una serie de características (Altura, Peso, Edad, etc., en este caso) que son las componentes del vector $\mathbf{x}$. La última columna es la etiqueta, y, asignada a cada instancia.

El objetivo de los algoritmos de aprendizaje supervisado es el de producir un modelo a partir del conjunto de datos con el que, una vez le pasemos como argumento un nuevo vector $\mathbf{x}$, sea capaz de calcular cuál es su etiqueta. El ejemplo canónico es el de un sistema de clasificación de correos electrónicos en las clases $\{spam, noSpam\}$. Pertenecen también a esta categoría los algoritmos que permiten predecir un valor numérico a partir del vector de características. Por ejemplo, si éste contiene información sobre un coche (Nº asientos, Caballos de potencia, Marca, etc.) [Alpaydin 2010], crear un modelo que prediga el precio que tendrá. El primero de los ejemplos anteriores pertenece a la clase de algoritmos que realizan tareas de *clasificación*, mientras que el segundo a la de los que realizan *regresión*. Esta última distinción no es exclusiva del aprendizaje supervisado, ya que también existen algoritmos no supervisados que realizan estas tareas.

3.2 Aprendizaje no supervisado

Dentro de esta categoría se encuentran los algoritmos que trabajan con un conjunto de datos que no tienen etiqueta. Como se puede inferir, trabajar con estos conjuntos es más complicado ya que no disponemos de ejemplos completos que describan el comportamiento deseado del modelo a construir.

Hasta ahora hemos dicho que los tipos de etiquetas que podían tener las instancias del conjunto de datos eran dos: las pertenecientes a un conjunto finito de clases y las que eran un número real. Esto ha sido así puesto que son los dos tipos más comunes y se trabaja con ellos tanto en aprendizaje supervisado como en no supervisado. Sin embargo, existe un tipo de etiquetas específico del aprendizaje no supervisado que son los vectores. Por tanto, el objetivo de estos algoritmos es el crear un modelo que transforme un vector $\mathbf{x}$ en un valor

(de una de las dos categorías vistas) o en otro vector que permita resolver un problema práctico [Géron 2017[. Alguna de las principales subcategorías en que pueden dividirse los algoritmos de esta clase son: los algoritmos de *agrupamiento*, los cuales devuelven un identificador del grupo al que pertenece cada vector de características, los algoritmos de *reducción de dimensionalidad*, que devuelven un vector de características de menor dimensión que el que recibe como entrada, o los algoritmos de *detección de anomalías*, que devuelven un número que representa lo atípica que es una instancia concreta del resto del conjunto de datos.

4 Modelos basados en la energía

Pasamos ahora a estudiar una serie de modelos para construir algoritmos de aprendizaje automático que, si bien podríamos haber desarrollado en la Sec. 3 anterior, hemos querido dedicarle una sección íntegra por su carácter esencial en el presente trabajo. Estos modelos trabajan con un sistema compuesto de distintas unidades que a menudo se conocen como *nodos*, *neuronas* o *partículas*. La característica fundamental de estos modelos es que dotan al sistema de una propiedad global, definida a partir del valor particular que cada una de las unidades que lo componen. A dicha propiedad global la conocemos como energía y es por ella que estos modelos tienen el nombre de *modelos basados en la energía*. Aquí vamos a ver los siguientes modelos: *modelo de Ising*, *modelo de Hopfield* y la *Máquina de Boltzmann* (MB). La elección de estos tres en concreto no sólo nos va a permitir estudiar conceptos generales de aprendizaje automático, sino que, además, nos va a permitir analizar y comprender un algoritmo particular de aprendizaje automático cuántico conocido como *Máquina de Boltzmann cuántica* (MBC), al que ya hicimos mención. El algoritmo cuántico se construye por analogía al clásico, por lo que nos conviene entender éste antes. A su vez, la Máquina de Boltzmann se construye como una evolución del modelo de Hopfield, el cual se basa en el modelo de Ising.

A pesar de que puede parecer un camino pesado el estudiar cada uno de estos modelos y no comenzar directamente por el estudio de la Máquina de Boltzmann, vamos a ver que el paso de uno a otro es realmente sencillo ya que comparten gran parte de sus características. Al estudiar el modelo de Ising nos centraremos en comprender los modelos basados en la energía de forma general. Después, en el estudio del modelo de Hopfield, veremos que modificando mínimamente las condiciones podemos conseguir un sistema que "aprenda" a reconocer patrones. Acabamos la sección con la Máquina clásica de Boltzmann, que va un paso más allá y nos permite aprender la distribución de estados de un conjunto de datos, al mismo tiempo que nos permite generar muestras que sigan dicha distribución.

4.1 Modelo de Ising

El modelo de Ising es un modelo propuesto por Wilhelm Lenz en 1920, quien se lo planteó luego a Ernst Ising para que lo resolviera como parte de su trabajo doctoral, lo que éste último hizo en 1925 [Ising 1925]. El modelo pretende caracterizar a un sistema ferromagnético y su solución exacta permite identificar transiciones de fase del sistema.

Vamos a considerar el sistema ferromagnético como una malla con N nodos, cada uno de ellos ocupado por un átomo con momento magnético μ. Si J es el momento angular total del átomo, esté podrá encontrarse en $(2J+1)$ orientaciones distintas en el espacio. De esta

forma, el número de configuraciones distintas en que puede encontrarse el sistema es $(2J+1)^N$. A partir de distintos estudios [Brown y Barnett 1952; Kettering y Scott 1944], se ha determinado que el valor de J que mejor se aproxima a los resultados experimentales es $J = 1/2$ [Pathria y Beale 2011], por lo que se puede inferir que el fenómeno del ferromagnetismo está ligado al espín de los electrones. Puesto que el espín del electrón es $s = 1/2$, cada punto de la malla del sistema pasa a tener únicamente dos posibles orientaciones: $s_z = \pm 1/2$. A partir de la teoría cuántica, es posible ver que la energía asociada a la interacción de dos espines vecinos es $\varepsilon_{ij} = K_{ij} \pm J_{ij}$, siendo K_{ij} la energía de Coulomb y J_{ij} a la energía de intercambio entre ambos. En esta expresión escogemos el signo positivo en el caso de que los espines sean antiparalelos ($S = 0$) y el negativo en el caso de que estos sean paralelos ($S = 1$). La diferencia de energía entre la configuración con espines paralelos y antiparalelos es, pues, $\varepsilon_{\uparrow\uparrow} - \varepsilon_{\uparrow\downarrow} = -2J_{ij}$. Atendiendo al valor del producto $\mathbf{s_i} \cdot \mathbf{s_j}$, es sencillo reescribir la energía de interacción como:

$$\varepsilon_{\uparrow\uparrow} - \varepsilon_{\uparrow\downarrow} = c - 2J_{ij} \left(\mathbf{s_i} \cdot \mathbf{s_j} \right),$$

donde tomamos un valor c arbitrario para la energía potencial puesto que éste va a depender del origen de energía. Aproximamos ahora por un valor nulo la energía de intercambio entre espines que no sean próximos, lo quiere decir que estos tan solo interactúan con sus vecinos más próximos, *v.p.* Si además reducimos el producto $\mathbf{s_i} \cdot \mathbf{s_j}$ simplemente a $s_{iz} s_{jz}$, podemos construir la energía de interacción de todo el sistema como:

$$E = C - \sum_{v.p.} J_{ij} \sigma^i \sigma^j , \tag{4.1}$$

donde σ^i, al igual que σ^j, toma el valor $+1$ si el espín es hacia arriba y el valor -1 si éste es hacia abajo. En general, son aquellos modelos que vienen descritos por la Ec. (4.1) a los que conocemos como modelos Ising.

Veamos ahora la forma de la energía del sistema para un caso particular sencillo. Escogemos, en primer lugar, un sistema de dos dimensiones, que vamos a describir mediante una red cuadrada de $N \cdot N$ nodos y escogemos también el origen de energía de manera que $C = 0$. En este ejemplo, la interacción con vecinos próximos va a significar *únicamente* con los vecinos más próximos. De esta manera, cada espín va a interaccionar con los que tiene a derecha e izquierda y con los dos más cercanos en la vertical. Modificamos ahora ligeramente la notación, de forma que sea más sencillo visualizar el modelo. Llamamos $\sigma(i,j)$ al espín del nodo situado en la posición horizontal i-ésima y en la posición vertical j-ésima de la red. Con todo, la expresión de la energía del sistema es ahora:

$$\begin{aligned} E = -\sum_{v.p.} J_{ij} \sigma^i \sigma^j = -\sum_{i=1}^{N} \sum_{j=1}^{N} \sigma(i,j) \left[J_{i,i;j,j+1}\, \sigma(i,j+1) + J_{i,i;j,j-1}\, \sigma(i,j-1) \right. \\ \left. + J_{i,i+1;j,j}\, \sigma(i+1,j) + J_{i,i-1;j,j}\, \sigma(i-1,j) \right], \end{aligned}$$

donde $J_{i,j;k,l}$ es la energía de interacción entre el espín situado en el punto (i,j) de la red y el situado en el punto (k,l). Sin embargo, aún nos queda un detalle y es que no hemos definido qué ocurre en los límites de la red. Aunque para escribir la expresión anterior lo hemos supuesto implícitamente, tomamos condiciones de contorno periódicas. Así, un nodo del extremo derecho tiene a su derecha el nodo de su misma fila pero del extremo izquierdo de la red. Las condiciones de contorno son entonces: $\sigma(i, N+1) = \sigma(i,1)$, $\sigma(i,0) = \sigma(i,N)$, $\sigma(N+1,j) = \sigma(1,j)$, $\sigma(0,j) = \sigma(N,j)$.

En el sistema que estamos considerando, tanto el número de nodos, N^2, como el volumen que estos ocupan, sea el que sea, son constantes. Si además consideramos que el sistema se encuentra en equilibrio termodinámico, la probabilidad $P(\mathbf{s})$ de que el sistema se encuentre en una configuración particular $\mathbf{s}$ es:

$$P(\mathbf{s}) = \frac{1}{Z} e^{-\beta E(\mathbf{s})}, \tag{4.2}$$

donde Z es la función de partición, $E(\mathbf{s})$ es la energía del sistema en el estado $\mathbf{s}$ y $\beta^{-1} = k_B T$, siendo k_B la constante de Boltzmann. Puesto que el sistema se encuentra en equilibrio termodinámico, se puede definir una temperatura, T, para éste. Los estados o configuraciones más probables son entonces aquellas con menor energía. Ya que estas son las configuraciones más probables, la evolución del sistema se dirige de forma natural hacia estados de menor energía. Esto no quiere decir que nunca evolucione hacia una configuración con mayor energía, tan solo que la probabilidad de que esto último ocurra es menor. En cualquier caso, esta evolución está fuertemente condicionada por el valor de la temperatura. A medida que ésta se aproxima al cero absoluto, el rango de configuraciones del sistema con una probabilidad (efectiva) distinta de cero disminuye enormemente. Únicamente las configuraciones menos energéticas tienen probabilidad no nula de realizarse. A temperaturas mayores, el rango aumenta y la evolución del sistema no se ve comprometida a ir en la dirección de configuraciones menos energéticas[10]. Es también interesante que veamos cuáles son estas configuraciones menos energéticas.

Si volvemos a la Ec. (4.1), es fácil darse cuenta de que, para que la energía sea mínima, es necesario que la suma sobre los vecinos próximos sea mínima. Ya que $J_{ij} > 0,\ \forall i,j$, y que $\sigma^i, \sigma^j = \{+1, -1\}$, dicha condición se cumplirá cuando los todos los espines sean paralelos. Es decir, el nivel de menor energía es doblemente degenerado, teniendo una de las configuraciones todos los espines hacia arriba $\left(\sigma^i = +1,\ \forall i\right)$ y la otra todos los espines hacia abajo $\left(\sigma^i = -1,\ \forall i\right)$. Este hecho, junto a la tendencia del sistema a evolucionar a configuraciones menos energéticas, se traduce en que el sistema tiende a evolucionar hacia una configuración en la que todas sus unidades son paralelas. A la hora de construir el modelo de Hopfield haremos referencia a este punto.

A lo largo de esta Subsec. 4.1 hemos estudiado el primer ejemplo de un modelo basado en energía: el *modelo de Ising*. Hemos construido la expresión de la energía que lo define y hemos podido ver cómo, a partir de las condiciones físicas impuestas, emerge un comportamiento como grupo ciertamente curioso: el sistema evoluciona hacia configuraciones en las que todos sus constituyentes son paralelos. En el próximo apartado partimos de estas ideas y vemos que aún podemos llevarlas más lejos, haciendo que el sistema evolucione hacia configuraciones particulares distintas de las dos vistas. Veremos cómo esto nos permite sentar las bases de un primer algoritmo sencillo de *aprendizaje automático*.

4.2 Modelo de Hopfield

En 1982, medio siglo después de que Lenz propusiese el modelo de Ising, John J. Hopfield publica un artículo en el que propone un novedoso modelo biológico [Hopfield 1982]. Este

[10] Una simulación sencilla del ejemplo que hemos visto la podemos encontrar aquí. El archivo $128x128\_T0.gif$ muestra la evolución de un sistema con temperatura próxima al cero absoluto, mientras que en $128x128\_T3.gif$ la temperatura tiene un valor 3. En ambos casos se toma k_B con valor unidad y T no es más que un parámetro.

modelo, que más tarde conoceríamos como *modelo de Hofield*, estudia la posibilidad de que ciertos fenómenos complejos, como la estabilidad de recuerdos o la construcción de categorías que generalizan a otras, puedan ser propiedades emergentes de la dinámica de un sistema, al igual que ocurría con los espines que se alinean en el modelo de Ising.

En este modelo consideramos un sistema de N unidades, a las que llamamos *neuronas*. Cada neurona i puede encontrarse en dos estados: $s_i = -1$ (*inactiva*) o $s_i = +1$ (*activa*), $i = 1, 2, \ldots, N$. Al contrario de lo que ocurre en el modelo de Ising, en el que cada nodo interacciona únicamente con sus vecinos próximos, ahora la interacción de cada neurona alcanza a todas las demás. A la interacción o conexión entre las neuronas i y j la llamamos w_{ij}. A estas conexiones también se las suele conocer como *pesos* o *pesos sinápticos*. El conjunto de todos los pesos forma una matriz W que, en este modelo, tiene dos particularidades. Por un lado, se considera que las neuronas no están conectadas a sí mismas, por lo que los términos de la diagonal de W son nulos: $w_{ii} = 0,\ \forall i$. Por otro, se considera que las conexiones son bidireccionales, es decir, que $w_{ij} = w_{ji},\ \forall i, j$. La matriz de pesos es entonces simétrica. Vamos a considerar, además, que cada neurona cuenta con un parámetro θ_i, al que llamamos *umbral de disparo*. Definimos entonces la *señal*, o *excitación*, e, que la neurona i recibe, como:

$$e_i = \sum_{j=1}^{N} w_{ij} s_j. \tag{4.3}$$

El valor de una neurona se actualiza según la siguiente regla [Hopfield 1982]:

$$s_i' = \begin{cases} +1 & \text{si } \sum_{j=1}^{N} w_{ij} s_j > \theta_i \\ -1 & \text{si } \sum_{j=1}^{N} w_{ij} s_j < \theta_i \end{cases}, \tag{4.4}$$

donde s_i' es el estado de la neurona i tras su actualización. Esta actualización se realiza de forma *asíncrona*. Esto quiere decir que cada neurona calcula su excitación en tiempos aleatorios, siendo nula la probabilidad de que dos neuronas se actualicen al mismo tiempo. Esta consideración permite simular el sistema de forma sencilla, eligiendo una neurona aleatoriamente en cada instante y calculado su nuevo estado según la regla anterior, Ec. (4.4).

Pasamos ahora a definir la energía del sistema, $E(\mathbf{s})$, cuando éste se encuentra en el estado $\mathbf{s}$. Dicha energía es [Feldman y Rojas 1996]:

$$E(\mathbf{s}) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} s_i s_j + \sum_{i=1}^{N} \theta_i s_i \tag{4.5}$$

donde el factor $1/2$ aparece para no contar dos veces los términos $w_{ij} s_i s_j$ y $w_{ji} s_j s_i$, que son idénticos. Cuando en una iteración se evalúa el nuevo valor de una neurona, puede ocurrir que ésta mantenga su valor o que éste cambie. En el primer caso, la energía del sistema tras la actualización es la misma. Sin embargo, si cambia el estado de la neurona, la energía del sistema también cambia. Veamos qué signo tiene este incremento de energía.

Si en cierta iteración el estado de la neurona k cambia, el incremento de energía del sistema cuando éste pasa del estado $\mathbf{s}$ al estado $\mathbf{s}'$ es:

$$E(\mathbf{s}') - E(\mathbf{s}) = \left(-\sum_{j=1}^{N} w_{kj} s_k' s_j + \theta_k s_k' \right) - \left(-\sum_{j=1}^{N} w_{kj} s_k s_j + \theta_k s_k \right), \tag{4.6}$$

donde ya no es necesario el factor $1/2$. Sacando $-\left(s'_k - s_k\right)$ como factor común, la expresión se reduce a

$$E(\mathbf{s}') - E(\mathbf{s}) = -\left(s'_k - s_k\right)\left(\sum_{j=1}^{N} w_{kj}s_j - \theta_k\right). \tag{4.7}$$

Si recordamos la regla de actualización de las neuronas, Ec. (4.4), vemos que si ésta cambia del valor -1 a $+1$, tenemos que los dos factores presentes en la Ec. (4.7) son positivos. De esta forma, el incremento de energía del sistema será negativo. En caso de que pase de estado activo a inactivo, que es el evento opuesto, ambos factores tienen signo negativo, por lo que el incremento de energía del sistema vuelve a ser negativo. Esto quiere decir que, cuando una neurona del sistema se actualiza, existen dos posibilidades: o ésta mantiene su estado y no cambia la energía del sistema, o cambia su estado y la energía del sistema disminuye. La energía del sistema es entonces monótonamente decreciente con el tiempo. Esto contrasta con lo que ocurría en el modelo de Ising, donde existía cierta probabilidad, dada por la Ec. (4.2), de que el sistema se moviese hacia configuraciones de mayor energía. La ventaja es que la convergencia hacia el mínimo local de energía en el caso del modelo de Hopfield es muy rápida, en términos del número de actualizaciones necesarias para alcanzarlo. En el modelo de Ising no sólo esta convergencia es más lenta sino que además no está asegurada. Es por ello que, a pesar de que la expresión de energía en ambos modelos es análoga, utilizamos los modelos con distintos propósitos.

Sin embargo, la característica principal y más interesante del modelo de Hopfield, y la que lo convierte en un modelo tan relevante en la historia del aprendizaje automático, es que podemos escoger qué configuraciones constituyen los mínimos de energía del sistema. Esta elección la llevamos a cabo al definir los coeficientes de la matriz de pesos, W. Estos se eligen mediante lo que conocemos como *regla de Hebb* [Hebb 2005], y que podemos resumir en lo siguiente:

Las neuronas que se disparan juntas fortalecen su conexión,

aunque realmente las relaciones que impone la regla Hebbiana son más complejas que la de este simple enunciado. Veamos en qué consiste.

Supongamos que tenemos un conjunto de P estados s^μ del sistema, con $\mu = 1, 2, \ldots P$, a los que vamos a llamar *patrones*. Definimos entonces el peso sináptico w_{ij} entre las neuronas i y j como:

$$w_{ij} = \begin{cases} \frac{1}{P}\sum_{\mu=1}^{P} s_i^\mu s_j^\mu & ;\ i \neq j \\ 0 & ;\ i = j \end{cases}, \tag{4.8}$$

donde s_i^μ es el estado de la neurona i en el patrón μ. Entonces, $w_{ij} = 1$ si en todos los patrones ambas neuronas están en el mismo estado y $w_{ij} = -1$ si en todos ellos están en estados contrarios. Tenemos entonces que demostrar que, con esta regla de construcción para W, la energía del sistema es mínima cuando su configuración coincide con la cada patrón. Hacemos primero la demostración para el caso de un único patrón y generalizamos después para el caso de P patrones. Durante el curso de estas demostraciones vamos a seguir el texto [Feldman y Rojas 1996].

Si tenemos un único patrón s^1, la matriz de pesos W^1 en notación vectorial es, según la Ec. (4.8):

$$W^1 = \mathbf{s}^1\mathbf{s}^{1\intercal} - \mathbf{I}\,, \tag{4.9}$$

donde $\mathbf{s}^1$ es un vector cuya componente k-ésima coincide con el estado de la neurona k del patrón s^1 y donde $\mathbf{I}$ es la matriz identidad de dimensión N^2. Recordemos que N es el número de neuronas del sistema. El superíndice $\intercal$ indica que el vector es el transpuesto. La expresión de la energía del sistema en una configuración $\mathbf{s}$ es, por tanto,

$$E(\mathbf{s}) = -\frac{1}{2}\mathbf{s}^\intercal W^1 \mathbf{s} = -\frac{1}{2}\left(\mathbf{s}^\intercal \mathbf{s}^1 \mathbf{s}^{1\intercal}\mathbf{s} - \mathbf{s}^\intercal \mathbf{I}\mathbf{s}\right). \tag{4.10}$$

Hemos escogido, sin pérdida de generalidad [Ackley, Hinton y Sejnowski 1985], el umbral de disparo de todas las neuronas igual a cero, $\theta_i = 0,\ \forall i$. Puesto que las componentes de los vectores de estado tan solo pueden tomar los valores -1 y 1, tenemos que $\mathbf{s}^\intercal\mathbf{s} = N$. Así, la expresión (4.10) se reduce a

$$E(\mathbf{s}) = -\frac{1}{2}\left\|\mathbf{s}^\intercal \mathbf{s}^1\right\|^2 + \frac{N}{2}, \tag{4.11}$$

donde podemos ver que la energía es mínima en el caso en que $\mathbf{s} = \mathbf{s}^1$, es decir, cuando la configuración del sistema coincide con la del patrón, *q.e.d.*. Puesto que, como ya vimos, el sistema nunca pasa a configuraciones de mayor energía, se resuelve que el estado $\mathbf{s}^1$ es, además, un estado estable del sistema.

La demostración para el caso de P patrones es algo más compleja. Un estado es estable porque el valor de las neuronas no cambia cuando éstas se actualizan, lo que ocurre en este modelo cuando nos encontramos en un mínimo de energía. El que el estado de las neuronas no cambie implica que, debido a la regla de actualización (4.4), si una neurona se encuentra en estado $+1$, su *excitación* debe ser positiva. Análogamente, si la neurona se encuentra en el estado -1, su *excitación* debe ser negativa. Seguimos trabajando con $\theta_i = 0,\ \forall i$. Es decir, la condición de estabilidad se reduce a que el *signo* del estado de la neurona y el de su excitación coincidan. Matemáticamente, decimos que el estado de la neurona i es estable si:

$$sgn(e_i) = sgn\left(\sum_{j=1}^{N} w_{ij}s_j\right) = sgn(s_i), \tag{4.12}$$

donde e_i vimos que era la excitación de la neurona i y donde $sgn(x)$ es la función signo[11]. Puesto que lo que queremos demostrar es que los patrones almacenados son estados estables del sistema, generalizamos la expresión anterior para incluir a todas las neuronas:

$$sgn(\mathbf{e}) = sgn(\mathbf{s}^\mu), \tag{4.13}$$

donde $\mathbf{e}$ es un vector cuya componente i-ésima es e_i y donde $\mathbf{s}^\mu$ es un vector cuya componente k-ésima toma el valor de la neurona k en el patrón μ. Decimos entonces que un patrón μ es estable si cumple la condición (4.13). Vamos a ver que esta condición se cumple en el modelo de Hopfield, aunque con ciertas restricciones.

La matriz de pesos sinápticos en el caso de P patrones es, según la Ec. (4.8):

$$\begin{aligned} W &= \frac{1}{P}\left[\left(\mathbf{s}^1\mathbf{s}^{1\intercal} - \mathbf{I}\right) + \left(\mathbf{s}^2\mathbf{s}^{2\intercal} - \mathbf{I}\right) + \cdots + \left(\mathbf{s}^P\mathbf{s}^{P\intercal} - \mathbf{I}\right)\right] \\ &= \frac{1}{P}\left(\mathbf{s}^1\mathbf{s}^{1\intercal} + \mathbf{s}^2\mathbf{s}^{2\intercal} + \cdots + \mathbf{s}^P\mathbf{s}^{P\intercal}\right) - \mathbf{I}. \end{aligned} \tag{4.14}$$

[11] La función signo se define: $sgn(x) = +1$ si $x > 0$; $sgn(x) = -1$ si $x < 0$; $sgn(x) = 0$ si $x = 0$.

Si el sistema se encuentra en el estado correspondiente a algún patrón, por ejemplo en el estado $\mathbf{s}^1$, la excitación de las neuronas del sistema es

$$\mathbf{e} = W\mathbf{s}^1 = \frac{1}{P}\left(\mathbf{s}^1\mathbf{s}^{1\intercal}\mathbf{s}^1 + \mathbf{s}^2\mathbf{s}^{2\intercal}\mathbf{s}^1 + \cdots + \mathbf{s}^P\mathbf{s}^{P\intercal}\mathbf{s}^1\right) - \mathbf{s}^1,$$

que se reduce a

$$\mathbf{e} = \left(\frac{N}{P} - 1\right)\mathbf{s}^1 + \frac{1}{P}\sum_{\mu=2}^{P}\alpha_{\mu 1}\mathbf{s}^\mu, \quad \text{con} \quad \alpha_{\mu 1} = \mathbf{s}^{\mu\intercal}\mathbf{s}^1. \tag{4.15}$$

Para que se cumpla la condición de estabilidad serán entonces necesarios dos requisitos. El primero es que $N > P$, es decir, que el número de neuronas del sistema debe ser mayor que el número de patrones a almacenar. El segundo es que el valor absoluto de $\frac{1}{P}\sum_{\mu=2}^{P}\alpha_{\mu 1}\mathbf{s}^\mu$ debe ser menor que $\frac{N}{P} - 1$. Esto se cumple trivialmente en el caso de patrones ortogonales. Es posible demostrar que, en el caso de patrones aleatorios, el número de estos que un sistema puede almacenar es aproximadamente un 13.8 % del número de neuronas de dicho sistema [Hertz y col. 1991].

Hemos demostrado que los patrones son configuraciones que minimizan la energía del sistema y que, por las propiedades del modelo, son además estados estables. Esta es, como dijimos, la característica principal que se deriva del modelo de Hopfield. Este comportamiento hace que el modelo sea especialmente útil a la hora de realizar diversas tareas, como: *reducción de ruido*, *reconstrucción de imágenes* o, la más evidente, *identificación de patrones*. Por sus características, a este modelo en aprendizaje automático se lo categoriza como algoritmo de *memoria asociativa*, categoría distinta al aprendizaje supervisado y no supervisado.

En conclusión, tenemos un modelo basado en energía que, mediante la combinación de la regla de actualización (4.4) y la regla de Hebb, permite definir los estados estables del sistema y que éste evolucione hacia dichos estados. Esto supone un gran avance respecto al modelo de Ising, y nos permite utilizarlo como algoritmo de aprendizaje automático. Sin embargo, el número de patrones que el sistema puede recordar en función del número de neuronas es muy bajo. Esto provoca que, a día de hoy, este algoritmo apenas sea utilizado, al existir otros para los que el ratio patrones/parámetros es mucho mayor. Un ejemplo de ello son las redes neuronales convolucionales, que no trataremos aquí. A pesar de la limitación descrita, este modelo sirve como base sobre la que desarrollar la *Máquina de Boltzmann*, cuya capacidad va más allá de la simple identificación de patrones. En el siguiente apartado vamos a estudiar este algoritmo, viendo cuáles son sus características principales y con qué propósito se ha utilizado. Así mismo, su estudio supondrá el final de la Sec. 4, que hemos dedicado a modelos clásicos basados en energía.

4.3 Máquina de Boltzmann

Hasta ahora hemos visto dos modelos en los que el comportamiento emergente del sistema se caracteriza por una evolución hacia estados o configuraciones de menor energía. En el caso del modelo de Ising, este comportamiento permitió a su autor explicar hasta cierto punto el fenómeno de la magnetización de un material ferromagnético. En el caso del modelo de Hopfield, el comportamiento que manifiesta el sistema permite la identificación de patrones, la reconstrucción de imágenes, etc., por lo que entra en la categoría de algoritmo de aprendizaje automático. A pesar de que las posibilidades del segundo modelo son mucho

mayores, ambos modelos son muy parecidos, contándose entre sus principales diferencias la conectividad entre neuronas y la regla de actualización de éstas. La conectividad total[12] del modelo de Hopfield, junto a la elección inteligente de W, permiten elegir los estados estables del sistema. En cambio, en el modelo de Ising la evolución del sistema es incierta, y aunque en algún momento alcance una de sus dos configuraciones de mínima energía, el sistema puede volver a configuraciones más energéticas. Además, en este modelo no es posible codificar la solución a un problema práctico en los estados de mínima energía, por lo que no es útil como algoritmo de aprendizaje. Por contra, la evolución del sistema en el modelo de Hopfield es determinista: si conocemos el perfil de energía de los estados del sistema y el estado del que parte, podemos determinar su estado final. En el caso del modelo de Ising ya hemos visto que no es así, puesto que una mayor probabilidad de este estado menos energético no asegura que se alcance en un tiempo finito, especialmente a temperaturas que se alejan del cero absoluto. Esta última característica conlleva que el modelo de Hopfield, como ya se dijo, no sea adecuado para problemas de optimización, ya que el sistema siempre quedará atrapado en el mínimo local de la configuración de partida. Estas disquisiciones nos sirven para introducir la Máquina de Boltzmann. Como vamos a ver, este modelo combina ciertas características de los dos anteriores y permite ir mucho más allá como algoritmo de aprendizaje automático.

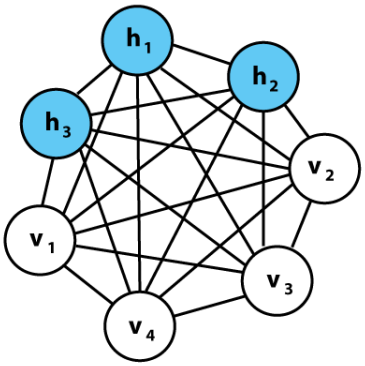

Figura 2: Esquema de la Máquina de Boltzmann. En blanco las unidades visibles y en azul las ocultas. Como podemos ver, todas las neuronas están conectadas entre sí. Fuente: [vd 2012].

La *Máquina de Boltzmann* fue propuesta en 1985 por Ackley, Hinton y Sejnowski en el artículo [Ackley, Hinton y Sejnowski 1985]. Al igual que en los otros dos modelos, el sistema está formado por un conjunto de N unidades que pueden estar en dos estados: $s_i = +1$ (*encendida*) o $s_i = -1$ (*apagada*), con $i = 1, 2, \ldots, N$. Como en el modelo de Hopfield, cada neurona consta de un parámetro adicional al que llamamos *umbral de disparo*, θ_i. Es muy común también que, dependiendo del contexto, hagamos referencia al *sesgo*, b_i, en vez de al umbral de disparo. El primero se define a partir del segundo como $b_i = -\theta_i$. Las expresiones que dependen de un parámetro u otro tan solo se distinguirán en el signo del término donde estos aparezcan. Existe, sin embargo, una diferencia en relación a las neuronas entre este modelo y los anteriores, y es que ahora éstas se dividen en dos grupos: el conjunto de *unidades visibles* y el conjunto de *unidades no visibles* u *ocultas*, Fig. 2.

Así, denotaremos por v_i el estado de la neurona visible i y por h_i al de la correspondiente neurona no visible. El estado global del sistema lo denotamos por $\mathbf{s} = (\mathbf{v}, \mathbf{h})$, con $s_k, v_i, h_j \in \{-1, +1\}$, donde $\mathbf{v}$ y $\mathbf{h}$ corresponden al estado de todas las neuronas visibles y

[12] Conectividad total en el sentido de que todas las neuronas están conectadas entre sí.

no visibles, respectivamente. Como en el modelo de Hopfield, todas las neuronas se encuentran, en principio, conectadas entre sí. El *peso sináptico* entre las neuronas i y j (del conjunto total de neuronas, dejando a un lado al grupo que pertenezcan) es w_{ij}. Estas conexiones vuelven a ser simétricas y no existe la conexión de una neurona consigo misma. La matriz de pesos W es, de nuevo, simétrica y con diagonal nula. Definimos la energía del sistema en el estado $\mathbf{s}$ como:

$$E(\mathbf{s}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}s_i s_j - \sum_{i=1}^{N} b_i s_i \,, \tag{4.16}$$

lo que coincide con la definición de la energía del modelo de Hopfield, Ec. (4.5). La regla de actualización de las neuronas recupera el carácter probabilístico que ya vimos en el modelo de Ising. La probabilidad p_i de que la neurona i-ésima pase al estado $+1$ es:

$$p_i(s_i = +1) = \frac{1}{1 + e^{-\Delta E_i}} \,, \tag{4.17}$$

donde ΔE_i es el incremento de energía del sistema cuando la neurona i pasa del estado activo al estado inactivo. Si recordamos la expresión del aumento de energía, Ec. (4.7), tenemos que

$$\Delta E_i = 2\left(\sum_{j=1}^{N} w_{ij}s_j + b_i\right) \,. \tag{4.18}$$

Si las neuronas se actualizan en orden aleatorio mediante la regla que acabamos de ver, Ec. (4.17), el sistema acaba alcanzando un estado de *equilibrio térmico* en el que la probabilidad de que el sistema se encuentre en un estado particular sigue la distribución de Boltzmann. En concreto, la probabilidad $P(\mathbf{u})$ de que el sistema se encuentre en una configuración particular $\mathbf{u}$ es:

$$P(\mathbf{u}) = \frac{e^{-E(\mathbf{u})}}{\sum_{\mathbf{s}} e^{-E(\mathbf{s})}} = \frac{e^{-E(\mathbf{u})}}{Z} \,, \tag{4.19}$$

donde hemos definido la función de partición como $Z = \sum_{\mathbf{s}} e^{-E(\mathbf{s})}$. Hasta ahora, la Máquina de Boltzmann tan solo se distingue del modelo de Hopfield en su regla de actualización para las neuronas. Como hemos visto, la principal ventaja de una regla probabilística es que dota al sistema de la capacidad para no quedar atrapado en los mínimos de energía, abriendo la posibilidad a que este modelo sea usado para resolver problemas de optimización. Sin embargo, debe existir algún otro rasgo que lo distinga de otros algoritmos de optimización y que haga uso, para ello, de su propiedad (4.19). Vamos a ver que la característica a la que nos referimos es la manera en que calculamos los pesos sinápticos, w_{ij}, entre las neuronas. Para ello, introduzcamos antes cuál es el problema para el que está especialmente bien diseñada la Máquina de Boltzmann, y que ya comentamos brevemente al final de la introducción de la presente Sec. 4.

La Máquina de Boltzmann pretende capturar la distribución de estados de cierto conjunto de datos de manera que, una vez finalizado el *aprendizaje*[13], pueda generar estados que sigan, aproximadamente, dicha distribución. La estructura del conjunto de datos es

[13] En este contexto, llamamos *aprendizaje* al método por el que los parámetros del sistema se ajustan para conseguir el objetivo para el que se diseña el algoritmo. En este caso, el *aprendizaje* consiste en ajustar los parámetros hasta que la distribución *aprendida* por el algoritmo sea lo suficientemente parecida a la del conjunto de datos.

la misma que estudiamos en la Subsec. 3.1, aunque este modelo no necesita de una etiqueta para las instancias pues, como vamos a ver, se trata de un modelo de aprendizaje no supervisado. Cada *instancia* del conjunto de datos es un vector $\mathbf{x}$ cuyas componentes pueden tomar los valores -1 y $+1$. A la distribución de estados del conjunto de datos la llamamos $P^{datos}(\mathbf{x})$. Esta distribución, que nos es desconocida, es la que queremos que sigan los estados generados por el modelo. La Máquina de Boltzmann, una vez elegidos de forma adecuada todos sus parámetros, funciona de la siguiente manera:

1. Se inicia el sistema en un estado $\mathbf{s}$ aleatorio y se deja evolucionar, de forma asíncrona y según la regla (4.17), durante un cierto tiempo. Cuando éste pasa, la distribución de los estados del sistema será independiente del estado inicial y corresponderá a la distribución de Boltzmann definida por los parámetros del sistema.

2. Cada uno de los estados por los que pasa el conjunto visible se considera un estado generado por el sistema. Es aquí donde cobran sentido los términos *visibles* y *ocultas* para nombrar a las neuronas del sistema, ya que son las primeras las únicas que consideramos accesibles.

Como resultado, en nuestro modelo los estados del sistema siguen una distribución de Boltzmann mientras que los estados del subconjunto visible seguirán, idealmente, la misma distribución que los del conjunto de datos. Sin embargo, aún necesitamos saber de qué forma escoger los parámetros del sistema de manera adecuada para que esto ocurra. Vamos a ver que la técnica de *descenso del gradiente*, que definiremos en un momento, nos indica una forma óptima de escoger estos valores.

Estudiamos, en primer lugar, cuál es probabilidad $P^{modelo}(\mathbf{v})$ de observar el estado particular $\mathbf{v}$ para el conjunto visible. Esta probabilidad viene dada por la distribución de Boltzmann, Ec. (4.19), sumada sobre los estados del conjunto *no visible*:

$$P^{modelo}(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{s})}, \quad \text{con} \quad \mathbf{s} = (\mathbf{v}, \mathbf{h}), \tag{4.20}$$

que no es más que una *distribución marginal*. Ya que queremos que $P^{modelo}(\mathbf{v})$ sea lo más *parecida* posible a $P^{datos}(\mathbf{x})$ que, como dijimos, es la distribución de estados del conjunto de datos, necesitamos definir una distancia que nos dé información acerca de cuán parecidas son. Usualmente se utiliza la *divergencia de Kullback − Leibler*[14], aunque aquí utilizaremos la *vverosimilitud logarítmica negativa*, $\mathcal{L}$, que se define como:

$$\mathcal{L} = -\sum_{\mathbf{v}} P^{datos}(\mathbf{v}) \log P^{modelo}(\mathbf{v}), \tag{4.21}$$

y que será con la que trabajemos también en el caso cuántico de este modelo. Una vez que hemos definido la distancia, estamos en condiciones de explicar en qué consiste la técnica de *descenso del gradiente*. Ésta se basa en calcular el gradiente de la magnitud a optimizar, en nuestro caso la verosimilitud logarítmica negativa, en función de los parámetros del sistema, lo que nos permite modificar estos en la dirección opuesta al gradiente:

$$\delta\lambda = -\epsilon\partial_\lambda \mathcal{L}, \tag{4.22}$$

[14]La divergencia de Kullback-Leibler, KL, se define como: $KL = \sum_{\mathbf{v}} P^{datos}(\mathbf{v}) \log \frac{P^{datos}(\mathbf{v})}{P(\mathbf{v})}$, por lo que únicamente se diferencia de la distancia definida por la verosimilitud logarítmica en el origen escogido. Así, cuando las distribuciones coinciden $KL = 0$, mientras que $\mathcal{L}$ será mínima pero no nula.

donde $\lambda \in \{w_{ij}, b_i\}$ y donde ϵ es la *tasa de aprendizaje*. Al calcular $\partial_\lambda \mathcal{L}$ obtenemos:

$$\begin{aligned} \partial_\lambda \mathcal{L} &= \sum_{\mathbf{v}} P^{datos}(\mathbf{v}) \frac{\sum_{\mathbf{h}} \partial_\lambda \left[E(\mathbf{s})\right] e^{-E(\mathbf{s})}}{\sum_{\mathbf{h}} e^{-E(\mathbf{s})}} - \frac{\sum_{\mathbf{s}'} \partial_\lambda \left[E(\mathbf{s}')\right] e^{-E(\mathbf{s}')}}{\sum_{\mathbf{s}'} e^{-E(\mathbf{s}')}} \\ &= \overline{\langle \partial_\lambda E(\mathbf{s}) \rangle_{\mathbf{v}}} - \langle \partial_\lambda E(\mathbf{s}') \rangle , \end{aligned} \tag{4.23}$$

donde $\langle \cdots \rangle$ y $\langle \cdots \rangle_{\mathbf{v}}$ son el valor esperado sin fijar y fijando el valor de las neuronas visibles, respectivamente, y donde $\overline{\langle \cdots \rangle_{\mathbf{v}}} \equiv \sum_{\mathbf{v}} P^{datos}(\mathbf{v}) \langle \cdots \rangle_{\mathbf{v}}$. Haciendo uso de la Ec. (4.16), podemos expresar el gradiente en función de los parámetros del sistema, lo que queda:

$$\delta b_i = \epsilon \left(\overline{\langle s_i \rangle_{\mathbf{v}}} - \langle s_i' \rangle \right), \tag{4.24}$$

$$\delta w_{ij} = \epsilon \left(\overline{\langle s_i s_j \rangle_{\mathbf{v}}} - \langle s_i' s_j' \rangle \right). \tag{4.25}$$

Calcular los distintos valores esperados nos permite modificar los parámetros del sistema de forma que $\mathcal{L}$ disminuya, lo que significa que la distribución de los estados generados por el modelo se parecerá más a la de los estados del conjunto de datos, que es nuestro objetivo último. Sin embargo, el cálculo de estos valores esperados no es algo sencillo de realizar en términos computacionales, véase [Montúfar 2018]. Es por ello que se creó una versión simplificada de la MB para la que sí se han desarrollado algoritmos que hacen el cálculo de forma eficiente. Esta versión simplificada se conoce como *Máquina de Boltzmann restringida*, (*MBR*). Vamos a ver en qué consiste y cuáles son las técnicas que se utilizan para calcular estos valores.

4.3.1 Máquina de Boltzmann restringida

Este modelo se distingue de la versión no restringida en la conexión que las neuronas tienen entre sí. Si antes la conexión era total, ahora no existen conexiones entre las unidades de un mismo conjunto (recordemos que las habíamos dividido en un conjunto de *unidades visibles* y en otro de *unidades ocultas*), Fig. 3.

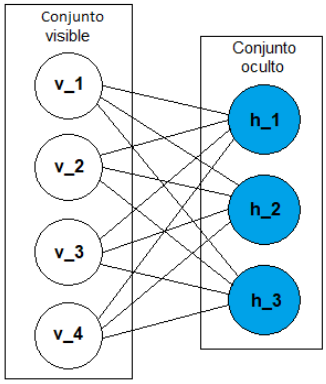

Figura 3: Esquema de la Máquina de Boltzmann restringida. En blanco las unidades visibles y en azul las ocultas. Como podemos ver no existen conexiones entre las unidades de un mismo conjunto.

Teniendo esto en cuenta, definimos la energía $E(\mathbf{s})$ asociada a una configuración $\mathbf{s}$ del sistema como:

$$E(\mathbf{s}) = -\frac{1}{2}\sum_{i=1}^{N_v}\sum_{j=1}^{N_h} w_{ij}v_i h_j - \sum_{i=1}^{N_v} a_i v_i - \sum_{j=1}^{N_h} b_j h_j, \tag{4.26}$$

donde $N_{v,h}$ es el número de neuronas en el conjunto visible y oculto, respectivamente, a_i es el sesgo de la neurona i del conjunto visible, b_j es el sesgo de la neurona j del conjunto no visible y donde w_{ij} es el peso sináptico entre la neurona i del conjunto visible y la j del no visible. Con la nueva función de energía del sistema, la Ec. (4.25) queda:

$$\delta w_{ij} = \epsilon\left(\overline{\langle v_i h_j\rangle_{\mathbf{v}}} - \left\langle v_i' h_j'\right\rangle\right). \tag{4.27}$$

El doble valor esperado $\overline{\langle v_i h_j\rangle_{\mathbf{v}}}$ es ahora sencillo de calcular. Al no existir conexiones entre las neuronas del conjunto no visible, si escogemos arbitrariamente un estado $\mathbf{x}$ del conjunto de datos para el estado del conjunto visible, $\mathbf{v} = \mathbf{x}$, la probabilidad de que la unidad j del conjunto no visible se encuentre en el estado $h_j = +1$ es [Hinton 2012]:

$$p\left(h_j = +1\,|\mathbf{v}\right) = \frac{1}{1+e^{-\Delta E_{h_j}}} = \sigma\left(\Delta E_{h_j}\right), \tag{4.28}$$

con

$$\Delta E_{h_j} = 2\left(\sum_{i=1}^{N_v} w_{ij}v_i + b_j\right), \tag{4.29}$$

y donde σ es la función sigmoide, $\sigma(x) = {}^{1}/_{(1+e^{-x})}$. Notemos que ahora la suma se realiza únicamente sobre las N_v neuronas del conjunto visible. Fijando el valor de h_j según la probabilidad (4.28), tenemos que el producto $v_i h_j$ es una muestra independiente para calcular el promedio. Repitiendo la operación varias veces más pero sin cambiar el estado del conjunto visible calculamos $\langle v_i h_j\rangle_{\mathbf{v}}$. Si repetimos todo el proceso para distintos estados del conjunto visible hallamos $\overline{\langle v_i h_j\rangle_{\mathbf{v}}}$.

Tras haber estudiado cómo calcular el primer sumando de la Ec. (4.27), llamado *fase positiva*, pasamos a indicar cómo se puede realizar el cálculo del segundo término, llamado *fase negativa*. Éste es muy costoso debido a la capacidad de cómputo que requiere, al ser un promedio sobre todos los estados $\mathbf{s}$ del sistema. Es por ello que la MB no fue apenas utilizada hasta algunos años después de que se crease el modelo. En el año 2002, de nuevo Hinton, propone en [Hinton 2002] un algoritmo que permite calcular el segundo sumando de la Ec. (4.27) de manera eficiente. Este algoritmo se conoce como *divergencia contrastante*, y se puede resumir en los siguientes pasos:

1. Tomar una de las instancias $\mathbf{x}$ del conjunto de datos como estado de las unidades visibles: $\mathbf{v} = \mathbf{x}$.

2. Calcular el estado del conjunto de neuronas no visibles a partir del estado de las visibles, utilizando para ello la Ec. (4.28). Puesto que las neuronas no visibles no están conectadas entre sí, este cálculo se puede llevar a cabo en paralelo. Notemos que hasta aquí el procedimiento es el mismo que para calcular la fase positiva.

3. A partir del estado generado para las no visibles *reconstruir* el estado de las visibles. Para ello, utilizar la expresión, análoga a la Ec. (4.28), siguiente:

$$p\left(v_i = +1\,|\mathbf{h}\right) = \sigma\left[2\left(\sum_{j=1}^{N_h} w_{ij}h_j + a_i\right)\right], \tag{4.30}$$

de forma que activamos cada neurona visible según esta probabilidad.

4. El producto $v_i h_j$, donde v_i es ahora el valor generado para la neurona i tras la reconstrucción del estado visible, es una muestra independiente con la que calcular $\langle v_i h_j \rangle$.

Al igual que antes, basta repetir la operación para obtener distintas muestras con las que hallar el valor esperado. Todo los pasos seguidos para el cálculo de δw_{ij} son aplicables para calcular tanto δa_i y δb_j, aunque las expresiones serán más sencillas. El aprendizaje consiste en realizar sucesivas actualizaciones de los parámetros del sistema, de la forma que hemos descrito, y optimizar así $\mathcal{L}$.

A lo largo de esta Subsec. 4.3, hemos conseguido formular un modelo capaz de aproximar la distribución de estados de un conjunto de datos. Hemos definido sus características básicas: los posibles estados de sus unidades, la relación entre ellas, sus sesgos y la energía del sistema. Tras esto hemos asignado una distribución de Boltzmann a los estados del sistema, de forma que la probabilidad de que el sistema se encuentre en uno de ellos se calcula a partir su energía. Con el objetivo de que el modelo reproduzca la distribución de los datos, hemos definido una distancia, la *verosimilitud logarítmica negativa*, que nos permita determinar cuán parecidas son la distribución objetivo y la que el modelo genera. Puesto que son los parámetros del sistema los que, en última instancia, controlan la distribución que genera el modelo (debido a la relación entre la distribución de probabilidad y la energía y, a su vez, de ésta con los parámetros), estudiamos de qué forma es necesario modificarlos para que el modelo se acerque a la distribución objetivo. Calculamos entonces el gradiente de la distancia, que depende de una serie de valores esperados. Puesto que no existe una manera conocida de calcularlos eficientemente, pasamos a describir un modelo especial de MB, la Máquina de Boltzmann restringida. Tras caracterizar este modelo a partir del modo particular en que están conectadas sus neuronas, definimos su energía y expresamos el gradiente de $\mathcal{L}$ en términos de las nuevas variables del sistema. Tras ello se explica cómo calcular el primero de los sumandos del gradiente y se indica, después, cómo hallar el segundo haciendo uso de la técnica de *divergencia contrastante*[15]. Al concluir la fase de aprendizaje, podemos utilizar el modelo para generar estados que sigan la misma distribución que los del conjunto de datos, tal y como se indicó al comienzo de la Subsec. 4.3.

5 Máquina de Boltzmann cuántica

Una vez visto el modelo de MB, así como su versión restringida, estamos en condiciones de estudiar su análogo cuántico: la Máquina de Boltzmann cuántica (MBC). Este modelo ha sido propuesto muy recientemente en el trabajo [Amin y col. 2018] y, aunque aún no existen dispositivos físicos capaces de aprovechar todas sus posibilidades, los resultados indican que se comporta mejor que su análogo clásico. Será al final de la presente Sección cuando hablemos sobre su implementación física, lo que nos permitirá enlazar con lo visto en la primera parte del trabajo puesto que dicha implementación pasa por utilizar el temple cuántico. En concreto, los estudios realizados hasta la fecha utilizan los dispositivos de la

[15] El lector interesado en conocer cómo se implementa este algoritmo puede consultar aquí una versión en Python de éste.

empresa D-Wave Inc., por lo que se encuentran limitados por las posibilidades que estos ofrecen. En cualquier caso, veamos antes en qué consiste la MBC.

En este modelo, nuestro sistema está formado por un conjunto de N qubits. Por lo visto en la Subsec. 2.1, sabemos que el espacio de Hilbert asociado a un sistema de N qubits tiene dimensión 2^N y que su base está formada por las 2^N configuraciones del sistema. Las unidades del sistema vuelven a dividirse en dos grupos: el conjunto de unidades *visible* y el conjunto de unidades *no visible*. Para conocer el estado clásico s_i de una neurona del sistema, medimos el operador σ_z^i asociado a dicha neurona. Este operador no es más que:

$$\sigma_z^i = \overbrace{\mathbf{I}\otimes\cdots\otimes\mathbf{I}}^{i-1}\otimes\sigma_z\otimes\overbrace{\mathbf{I}\otimes\cdots\otimes\mathbf{I}}^{N-i}, \tag{5.1}$$

donde

$$\mathbf{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Como podemos ver, se trata de un operador definido sobre el espacio de Hilbert del sistema, actuando únicamente de forma no trivial sobre el subespacio de dos dimensiones asociado al i-ésimo qubit mediante la matriz Z de Pauli. Por tanto, los posibles estados clásicos de la neurona i-ésima son $s_i \in \{-1, +1\}$. Cuando sea necesario diferenciar si una neurona corresponde al conjunto visible o al no visible, nos referiremos al estado de ésta como v_i o h_j, respectivamente. Al igual que en el caso clásico, cada neurona tendrá asociado un parámetro b_i que llamamos *sesgo*. Las neuronas vuelven a estar relacionadas dos a dos mediante los pesos sinápticos w_{ij}, que recordemos relacionan las neuronas i y j del sistema, de la misma forma que en la MB. Sobre el espacio de Hilbert del sistema definimos también el operador Hamiltoniano siguiente:

$$\mathcal{H} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\sigma_z^i\sigma_z^j - \sum_{i=1}^{N} b_i\sigma_z^i, \quad \text{con } \{b_i, w_{ij} \in \mathbb{R} \,|\, w_{ij} = w_{ji};\ w_{ii} = 0\}\ \forall i, j, \tag{5.2}$$

expresión análoga a la Ec. (4.16) pero sustituyendo el estado de las neuronas $s_{i,j}$ por los operadores $\sigma_z^{i,j}$, respectivamente. Los elementos de la diagonal del Hamiltoniano son los posibles estados de energía del sistema. Denotamos por $|\mathbf{s}\rangle = |\mathbf{v}, \mathbf{h}\rangle$, con $|\mathbf{s}\rangle = |s_1, s_2, \ldots, s_N\rangle$ y $|\mathbf{v}, \mathbf{h}\rangle = |v_1, v_2, \ldots, v_{N_v}, h_1, h_2, \ldots, h_{N_h}\rangle$ y donde $N_{v,h}$ es el número de neuronas visibles y ocultas, respectivamente, a los estados propios de este operador. Para describir la probabilidad de que el sistema se encuentre en cualquiera de sus 2^N estados vamos a utilizar el formalismo cuántico de matriz de densidad, ρ. Puesto que queremos que estos estados sigan una distribución de Boltzmann, definimos la matriz de densidad de la siguiente forma:

$$\rho = \frac{e^{-\mathcal{H}}}{\mathrm{Tr}\left[e^{-\mathcal{H}}\right]} = \frac{e^{-\mathcal{H}}}{Z}, \tag{5.3}$$

donde $e^{-\mathcal{H}}$, al igual que $\mathcal{H}$, es una matriz diagonal de dimensión 2^N y donde Tr indica que tomamos la traza. Los elementos de la diagonal, $e^{-E(\mathbf{s})}$, corresponden a la probabilidad de que al medir el estado del sistema éste se encuentre en el estado $|\mathbf{s}\rangle$. En la ecuación anterior hemos definido también la función de partición como $Z = \mathrm{Tr}\left[e^{-\mathcal{H}}\right]$. A partir de la Ec. (5.3) podemos obtener la probabilidad de que el conjunto visible se encuentre en el estado $|\mathbf{v}\rangle$, que es:

$$P^{modelo}(\mathbf{v}) = \mathrm{Tr}\left[\Lambda_{\mathbf{v}}\rho\right], \tag{5.4}$$

donde $\Lambda_{\mathbf{v}} = |\mathbf{v}\rangle\langle\mathbf{v}| \otimes \mathbf{I_h}$. Este operador tiene la función de limitar la traza a únicamente aquellos estados propios de la matriz densidad en que el conjunto visible se encuentre en el estado $|\mathbf{v}\rangle$. El operador $|\mathbf{v}\rangle\langle\mathbf{v}|$ es el proyector sobre el subespacio de unidades visibles en estado $|\mathbf{v}\rangle$, mientras que $\mathbf{I_h}$ es la matriz identidad definida sobre el subespacio de Hilbert correspondiente al conjunto no visible. La Ec. (5.4) es equivalente a la Ec. (4.20) del modelo clásico en el caso en que $\mathcal{H}$ y, por tanto, ρ sean diagonales.

Sin embargo, el formalismo estudiado permite también trabajar en el caso en que estos operadores no sean diagonales. Podemos introducir términos sencillos fuera de la diagonal utilizando para ello el operador σ_x^i, cuya descomposición es la misma que vimos en la Ec. (2.3). El Hamiltoniano del sistema introduciendo estos nuevos términos no diagonales lo podemos escribir como:

$$H = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}\sigma_z^i\sigma_z^j - \sum_{i=1}^{N} b_i\sigma_z^i - \sum_{i=1}^{N}\Gamma_i\sigma_x^i = \mathcal{H} - \sum_{i=1}^{N}\Gamma_i\sigma_x^i, \tag{5.5}$$

donde Γ_i es un número real correspondiente a la "intensidad" del campo transversal del qubit i-ésimo. El nuevo término no diagonal del Hamiltoniano representa las componentes transversales del estado de los qubits. Los estados propios del nuevo Hamiltoniano serán superposiciones de estados $|\mathbf{s}\rangle$, mientras que las Ecs. (5.3) y (5.4) siguen siendo válidas. Es muy importante notar que H se trata de un Hamiltoniano de Ising transversal y que, como estudiamos en la Subsec. 2.3.1, es la clase de Hamiltonianos que pueden ser implementados en los dispositivos de D-Wave. Como vamos a ver, este hecho es de importancia fundamental a la hora de realizar físicamente la MBC.

Como ya vimos en el modelo clásico, nuestro objetivo es conseguir que la distribución $P^{modelo}(\mathbf{v})$ de estados del conjunto visible generados por el modelo sea lo más parecida posible a la distribución $P^{datos}(\mathbf{x})$ de estados $\mathbf{x}$ de cierto conjunto de datos dado. Para ello, volvemos a utilizar la verosimilitud logarítmica negativa, $\mathcal{L}$, como medida del parecido entre ambas distribuciones. A partir de la Ec. (4.21) llegamos fácilmente a que en el modelo cuántico esta medida pasa a ser:

$$\mathcal{L} = -\sum_{\mathbf{v}} P^{datos}(\mathbf{v}) \log \frac{\mathrm{Tr}\left[\Lambda_{\mathbf{v}} e^{-H}\right]}{\mathrm{Tr}\left[e^{-H}\right]}, \tag{5.6}$$

donde hemos utilizado que la distribución de los datos es la dada por la Ec. (5.4) y que $\mathrm{Tr}\left[\Lambda_{\mathbf{v}}\rho\right] = \frac{\mathrm{Tr}\left[\Lambda_{\mathbf{v}} e^{-H}\right]}{\mathrm{Tr}\left[e^{-H}\right]}$. Como vimos, para escoger de forma apropiada los parámetros del sistema utilizamos la técnica de descenso del gradiente, optimizando así la medida escogida. El gradiente de $\mathcal{L}$ respecto a los parámetros del sistema, $\lambda \in \{w_{ij}, b_i\}$, es:

$$\partial_\lambda \mathcal{L} = \sum_{\mathbf{v}} P^{datos}(\mathbf{v}) \left(\frac{\mathrm{Tr}\left[\Lambda_{\mathbf{v}} \partial_\lambda e^{-H}\right]}{\mathrm{Tr}\left[\Lambda_{\mathbf{v}} e^{-H}\right]} - \frac{\mathrm{Tr}\left[\partial_\lambda e^{-H}\right]}{\mathrm{Tr}\left[e^{-H}\right]} \right). \tag{5.7}$$

Nuestra esperanza es que los dos sumandos de esta expresión se reduzcan, como en el caso clásico, al cálculo de una serie de valores esperados que sean sencillos, computacionalmente, de calcular. Sin embargo, como demuestran en [Amin y col. 2018], el cálculo del primer sumando requiere una estimación numérica que se vuelve muy costosa, puesto que es necesario realizarla para cada uno de los vectores del conjunto de datos. Para solucionar este problema, los autores del estudio proponen realizar el gradiente no de la propia $\mathcal{L}$ sino de

una cota superior de ésta. Encuentran, a partir de la desigualdad de Golden-Thompson[16], la siguiente cota inferior para la probabilidad $P^{modelo}(\mathbf{v})$:

$$P^{modelo}(\mathbf{v}) = \frac{\mathrm{Tr}\left[\Lambda_{\mathbf{v}} e^{-H}\right]}{\mathrm{Tr}\left[e^{-H}\right]} \geq \frac{\mathrm{Tr}\left[e^{-H_{\mathbf{v}}}\right]}{\mathrm{Tr}\left[e^{-H}\right]}, \tag{5.8}$$

donde $H_{\mathbf{v}} = \langle \mathbf{v}|H|\mathbf{v}\rangle$ es el Hamiltoniano del sistema cuando los qubits del conjunto visible se encuentran en el estado clásico $|\mathbf{v}\rangle$. Si introducimos esta cota para $P^{modelo}(\mathbf{v})$ en la Ec. (5.6), obtenemos una cota superior, $\widetilde{\mathcal{L}}$, para la distancia entre distribuciones:

$$\mathcal{L} \leq \widetilde{\mathcal{L}} \equiv -\sum_{\mathbf{v}} P^{datos}(\mathbf{v}) \log \frac{\mathrm{Tr}\left[e^{-H_{\mathbf{v}}}\right]}{\mathrm{Tr}\left[e^{-H}\right]}. \tag{5.9}$$

Calculando ahora el gradiente de $\widetilde{\mathcal{L}}$ respecto a los parámetros del sistema, tenemos que

$$\begin{aligned} \partial_\lambda \widetilde{\mathcal{L}} &= \sum_{\mathbf{v}} P^{datos}(\mathbf{v}) \frac{\mathrm{Tr}\left[e^{-H_{\mathbf{v}}} \partial_\lambda H_{\mathbf{v}}\right]}{\mathrm{Tr}\left[e^{-H_{\mathbf{v}}}\right]} - \frac{\mathrm{Tr}\left[e^{-H} \partial_\lambda H\right]}{\mathrm{Tr}\left[e^{-H}\right]} \\ &= \overline{\langle \partial_\lambda H_{\mathbf{v}} \rangle_{\mathbf{v}}} - \langle \partial_\lambda H \rangle, \end{aligned} \tag{5.10}$$

donde $\langle \cdots \rangle_{\mathbf{v}} = \frac{\mathrm{Tr}\left[e^{-H_{\mathbf{v}}} \cdots\right]}{\mathrm{Tr}\left[e^{-H_{\mathbf{v}}}\right]}$ y donde, de nuevo, $\overline{\langle \cdots \rangle_{\mathbf{v}}} \equiv \sum_{\mathbf{v}} P^{datos}(\mathbf{v}) \langle \cdots \rangle_{\mathbf{v}}$. Entonces, según la expresión de descenso del gradiente, Ec. (4.22), la expresión de la variación de cada uno de los parámetros del sistema es:

$$\delta b_i = \epsilon \left(\overline{\langle \sigma_z^i \rangle_{\mathbf{v}}} - \langle \sigma_z^i \rangle \right), \tag{5.11}$$

$$\delta w_{ij} = \epsilon \left(\overline{\langle \sigma_z^i \sigma_z^j \rangle_{\mathbf{v}}} - \langle \sigma_z^i \sigma_z^j \rangle \right), \tag{5.12}$$

donde recordemos que ϵ era la tasa de aprendizaje. Si recordamos el modelo clásico, para realizar el cálculo de estos valores esperados introdujimos la MB restringida. En el modelo cuántico, tal y como H. Amin y col. demuestran en su trabajo, una versión de la MB en la que se anula la conexión únicamente entre las neuronas del conjunto no visible permite calcular, de manera exacta, el valor esperado $\langle \sigma_z^i \rangle_{\mathbf{v}}$ que aparece en la Ec. (5.11). A la MB con este tipo de relación entre sus neuronas la conocemos como *Máquina de Boltzmann cuántica semirestringida* (MBCR). En el caso de que se trate de una neurona del conjunto visible tenemos trivialmente que $\langle \sigma_z^i \rangle_{\mathbf{v}} = v_i$, siendo v_i la componente i-ésima del estado $|\mathbf{v}\rangle$ en el que se encuentra el conjunto visible. Recordemos que, durante el proceso de actualización de los parámetros del sistema, los estados del conjunto visible los escogemos al azar de entre los estados del conjunto de datos. Veamos qué ocurre en el caso de que el valor esperado corresponda a una neurona no visible. Utilizando el modelo de la MBCR encontramos que, para un estado particular $|\mathbf{v}\rangle$ del conjunto visible, $H_{\mathbf{v}}$ es:

$$H_{\mathbf{v}} = -\sum_{j=1}^{N_h} \left[\Gamma_j \sigma_x^j + b_j^{\mathrm{eff}}(\mathbf{v}) \sigma_z^j \right],$$

con $b_j^{\mathrm{eff}}(\mathbf{v}) = b_j + \sum_{i=1}^{N_v} w_{ij} v_i$. La expresión analítica del valor esperado que buscamos calcular es entonces:

$$\langle \sigma_z^j \rangle_{\mathbf{v}} = \frac{b_j^{\mathrm{eff}}}{D_j} \tanh D_j, \tag{5.13}$$

[16] La desigualdad de Golden-Thompson enuncia lo siguiente: $\mathrm{Tr}\left[e^{A} e^{B}\right] \geq \mathrm{Tr}\left[e^{A+B}\right]$, donde A y B son matrices hermíticas.

siendo $D_j = \sqrt{\Gamma_j^2 + \left(b_j^{\text{eff}}\right)^2}$. Ahora, para calcular $\overline{\langle \sigma_z^j \rangle_{\mathbf{v}}}$, tan solo es necesario que repitamos el cálculo anterior para distintos estados del conjunto visible. La Ec. (5.13) también nos permite calcular la fase positiva de la Ec. (5.12). Puesto que el estado del conjunto visible es el estado clásico que escogemos del conjunto de datos, $\langle \sigma_z^i \sigma_z^j \rangle_{\mathbf{v}}$ se reduce trivialmente a

$$\langle \sigma_z^i \sigma_z^j \rangle_{\mathbf{v}} = \begin{cases} \langle v_i \sigma_z^j \rangle_{\mathbf{v}} = v_i \langle \sigma_z^j \rangle_{\mathbf{v}} & ;\ i{\in}\textit{visibles},\ j{\in}\textit{ocultas} \\ v_i v_j & ;\ i{\in}\textit{visibles},\ j{\in}\textit{visibles} \end{cases}. \tag{5.14}$$

El caso en el que las dos neuronas sean ocultas no se contempla puesto que no existe conexión entre unidades del conjunto oculto en el modelo de MBCR. De esta forma, vemos que el cálculo de este valor esperado se reduce a conocer el de las neuronas del conjunto no visible, para lo que contamos con la Ec. (5.13). A pesar de que este modelo restringido nos permite calcular las fases positivas, no nos dice nada sobre cómo calcular las negativas. Es en este momento en el que hacemos uso de los conceptos estudiados al comienzo de este trabajo y, en concreto, del temple cuántico.

Como explicamos en la Subsec. 2.3, el hecho de que en el temple cuántico la condición de adiabaticidad se relaje implica que, tras un proceso de temple, existe una probabilidad no nula de que el estado final del sistema no sea el estado fundamental del *Hamiltoniano problema*[17], que identificamos como $\mathcal{H}$ al comparar las Ecs. (2.11) y (5.5). Esto permite que, si somos capaces de controlar la dinámica del sistema, el proceso de temple cuántico se convierta en un proceso de muestreo de cierta distribución de probabilidad. En nuestro caso, el de la MBC, la distribución de la que queremos obtener muestras es la de Boltzmann, Ec. (5.3). Es por ello que la dificultad es entonces doble. Por un lado, necesitamos implementar físicamente un sistema cuyo Hamiltoniano sea el de la Ec. (5.5) para realizar el temple y, por otro, necesitamos que la distribución de estados propios del Hamiltoniano final sea una distribución de Boltzmann. Construir el modelo de MBC significa entonces construir un modelo que cumpla con estas dos exigencias.

La primera de ellas es sencilla de satisfacer, puesto que se reduce a una reparametrización. Como hemos visto, podemos implementar el Hamiltoniano de la MBC, Ec. (5.5), en los dispositivos de D-Wave. Para ello es necesario encontrar la correspondencia entre sus parámetros y los de la Ec. (2.11). Identificando cada uno de los términos, vemos que la reparametrización necesaria es la siguiente [Amin y col. 2018]:

$$\begin{aligned} \Gamma_i &= \Gamma = \beta A(s^*) \\ b_i &= \beta B(s^*) k_i \\ w_{ij} &= \beta B(s^*) J_{ij}, \end{aligned}$$

donde $\beta = \kappa_B T$, siendo κ_B la constante de Boltzmann y T la temperatura del sistema durante el temple, y donde s^* es un instante del proceso de temple conocido como *tiempo de congelación* y cuyo significado explicaremos en un momento. Puesto que tanto k_i como J_{ij} son parámetros del sistema que podemos controlar, tan solo necesitamos saber cómo controlar también s^*.

Pasamos ahora a ver cómo resolver la segunda condición: la distribución de estados del sistema al final del temple debe ser una distribución de Boltzmann. Para que esto ocurra deben darse una serie de condiciones, tal y como se estudia en [Amin 2015].

[17] Recordemos que en la Subsec. 2.3.1 vimos que llamábamos *Hamiltoniano problema* a aquel correspondiente al sistema al final del proceso de temple. Matemáticamente, éste es el factor que acompaña al término $B(s)$ en la Ec. (2.11) y que tiene la misma forma que el de la Ec. (5.2).

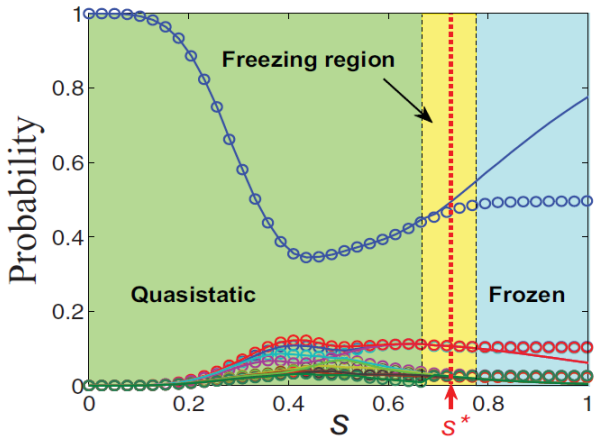

Figura 4: Ejemplo de un proceso de temple. (Círculos) Probabilidad de ocupación durante un proceso de temple cuántico de los 12 estados de menor energía para un sistema de 16 qubits. (Líneas) Distribución de Boltzmann para el mismo sistema. En el proceso de temple, la temperatura T del sistema se escogió como $T = 40$ mK y tiempo de templado $t_a = 20$ μs. Fuente: [Amin 2015].

En la Figura 4, podemos ver las características fundamentales de un proceso de temple cuántico en los dispositivos de D-Wave. En ella podemos comparar la distribución de Boltzmann (líneas) con la probabilidad de ocupación de los estados (círculos) para cada instante de tiempo. Durante la primera parte del temple (zona verde), la evolución del sistema es *cuasiestática*. Decimos que un sistema sigue una *evolución cuasiestática* si la dependencia de su dinámica con el tiempo es muy lenta en comparación con el tiempo de relajación de éste. En tales condiciones, el sistema permanece en un estado de casi equilibrio. La distribución de estados coincide entonces con la de Boltzmann. Sin embargo, al encontrarnos aún lejos del final del temple, la energía de estos estados no coincide con la de los valores propios de $\mathcal{H}$. A continuación, el tiempo de relajación del sistema comienza a aumentar (región amarilla), de forma que la ocupación de estados se aleja de la distribución de Boltzmann. Al instante de tiempo en el que esta desviación se hace evidente lo conocemos como *punto de congelación*, s^*. Es a partir de este momento, siendo aún la ocupación de niveles energéticos parecida a la distribución que buscamos, cuando dicha probabilidad se "congela", manteniéndose prácticamente constante durante el resto del temple. Con todo, necesitaremos entonces que prácticamente toda la evolución ocurra en la zona cuasiestática del temple, siendo las otras dos regiones lo más breves posibles, y que $A(s^*)$, la amplitud del término transversal del Hamiltoniano en el punto de congelación, sea lo más pequeña posible. Lo primero permite que la distribución de estados obtenida sea parecida a la de Boltzmann, mientras que lo segundo permite que esta distribución corresponda al espectro de energías de $\mathcal{H}$.

Actualmente, las limitaciones de los dispositivos de D-Wave hacen muy difícil controlar estos dos aspectos del proceso de temple. En general dependen del problema, por lo que el uso del temple como un generador de muestras para una distribución de Boltzmann no es aún de aplicabilidad general. Sin embargo, resultados preliminares como [Adachi y Henderson 2015; Amin y col. 2018] realizados para casos en que se cumplen las anteriores condiciones, demuestran que este modelo consigue aproximar la distribución objetivo igual de bien que el modelo clásico con un menor número de actualizaciones de sus parámetros.

A lo largo de este apartado hemos estudiado el modelo de Máquina de Boltzmann

cuántica. Hemos estudiado cómo la energía asignada a cada estado en el modelo clásico pasa ahora a ser un Hamiltoniano y la forma de éste. Tras ello, describimos el sistema mediante su matriz densidad y encontramos la probabilidad de que el conjunto visible se encuentre en un estado $|\mathbf{v}\rangle$ particular. Más tarde generalizamos el Hamiltoniano del sistema para incluir términos no diagonales, lo que hacemos mediante los términos de espín transversal. La forma del Hamiltoniano del sistema coincide entonces con la de los Hamiltonianos que podíamos implementar en los dispositivos de D-Wave. Definimos, como en el caso clásico, la distancia entre la distribución de estados visibles generada por el modelo y la del conjunto de datos y calculamos el gradiente en función de los parámetros del sistema. Sin embargo, $\mathcal{L}$ es ahora computacionalmente muy costosa de calcular. Encontramos la solución a este problema haciendo uso de una cota superior de la distancia. Con ella, la expresión para la variación de los parámetros es análoga a la del caso clásico, dependiendo de una serie de valores esperados sobre una distribución de Boltzmann. Para realizar el cálculo de la fase positiva utilizamos un modelo simplificado de la MBC, la MBC semirrestringida. Con este modelo podemos calcular de manera exacta los valores esperados $\langle \cdots \rangle_{\mathbf{v}}$, mientras que el cálculo de $\overline{\langle \cdots \rangle_{\mathbf{v}}}$ se reduce a calcular el primero con distintos $|\mathbf{v}\rangle$. Para calcular la fase negativa hacemos uso del temple cuántico, estudiado en la primera mitad del presente trabajo. Bajo una serie de condiciones, vemos que el proceso de temple puede ser utilizado como generador de muestras de una distribución de Boltzmann, si bien en la actualidad aún no es una posibilidad general. Con dichas muestras calculamos los valores esperados de las fases negativas. Al saber cómo actualizar los parámetros hemos conseguido construir la MBC.

6 Conclusiones

Una vez terminado nuestro estudio, podemos realizar ciertas reflexiones acerca de los resultados que hemos visto. En primer lugar, debemos advertir que aún nos encontramos en los inicios del aprendizaje automático cuántico, tan solo tenemos que fijarnos en que la mayor parte de los resultados presentados en este trabajo han sido publicados en los últimos 20 años. A pesar de ello, y tal y como hemos podido comprobar, los resultados son prometedores, por lo que cabe esperar que en los próximos años la comunidad investigadora dirija cada vez más su atención y esfuerzo a este área del conocimiento. El ejemplo de la Máquina de Boltzmann cuántica es muy interesante y permite adivinar que la investigación los próximos años va a ser realmente emocionante. Recordemos que el proceso físico que permite la implementación de este algoritmo es el temple cuántico, el cual no es un modelo de computación universal y, a pesar de ello, el tipo de cálculo que nos permite realizar es de una tremenda utilidad. Así, vemos que no será necesario disponer de un computador cuántico universal para utilizar el potencial de la computación cuántica en nuestro día a día. Los computadores cuánticos de propósito específico, como podríamos clasificar, por ejemplo, el de D-Wave, probablemente no estén lejos de ser utilizados en tareas de aprendizaje automático [Biamonte y col. 2017]. Esperamos que en los próximos años las nuevas arquitecturas de los dispositivos de D-Wave Systems [Dattani, Szalay y Chancellor 2019], y de otras compañías que nazcan, atajen la enorme brecha que aún existe entre computación cuántica y clásica, al menos en lo que a estos problemas específicos se refiere. Para ello, estas nuevas arquitecturas tendrán que salvar las numerosas limitaciones de los dispositivos actuales, entre las que se encuentran: el escaso número de qubits disponibles, la conexión parcial entre ellos o el control reducido de los parámetros del temple, tal y como hemos

comprobado en el ejemplo estudiado. Sin embargo, superar estas limitaciones no permite resolver un mayor número de tipos de problemas (en cuanto a su clase de complejidad) ya que, como vimos, estos estaban limitados por la clase de Hamiltoniano con que podemos realizar el temple. En todo caso, lidiar con este problema no es aún una prioridad pues existen numerosos problemas de tipo NP-completo cuya resolución eficiente ya supondría un enorme logro debido al impacto positivo que tendría para nuestra sociedad.

Por último, esperamos que surjan nuevos y más potentes algoritmos de aprendizaje automático capaces de aprovechar las posibilidades que ofrecen los dispositivos cuánticos de computación. Pensamos que sería muy interesante, por ejemplo, tratar de conseguir una versión cuántica del algoritmo de aprendizaje conocido como *Red de Creencia Profunda* [Hinton, Osindero y Teh 2006]. Éste utiliza la Máquina de Boltzmann como bloque fundamental, aunque su potencial es mucho mayor que el de éstas, por lo que parece un paso lógico a dar en la investigación en aprendizaje automático cuántico.

Referencias

(s.f.). D-Wave Systems Inc. URL: https://www.dwavesys.com/.

Ackley, D., Geoffrey E. Hinton y T. Sejnowski (1985). "A Learning Algorithm for Boltzmann Machines". En: Cogn. Sci. 9, págs. 147-169.

Adachi, Steven H. y Maxwell P. Henderson (2015). "Application of Quantum Annealing to Training of Deep Neural Networks". En: ArXiv abs/1510.06356.

Aharonov, Dorit y col. (2008). "Adiabatic Quantum Computation Is Equivalent to Standard Quantum Computation". En: SIAM Review 50.4, págs. 755-787. DOI: 10.1137/080734479. URL: https://doi.org/10.1137/080734479.

Albash, Tameem y Daniel A. Lidar (ene. de 2018). "Adiabatic quantum computation". En: Rev. Mod. Phys. 90 (1), pág. 015002. DOI: 10.1103/RevModPhys.90.015002. URL: https://link.aps.org/doi/10.1103/RevModPhys.90.015002.

Alpaydin, Ethem (2010). Introduction to Machine Learning. 2nd. The MIT Press. ISBN: 026201243X.

Amin, Mohammad H. (jun. de 2009). "Consistency of the Adiabatic Theorem". En: Phys. Rev. Lett. 102 (22), pág. 220401. DOI: 10.1103/PhysRevLett.102.220401. URL: https://link.aps.org/doi/10.1103/PhysRevLett.102.220401.

— (nov. de 2015). "Searching for quantum speedup in quasistatic quantum annealers". En: Phys. Rev. A 92 (5), pág. 052323. DOI: 10.1103/PhysRevA.92.052323. URL: https://link.aps.org/doi/10.1103/PhysRevA.92.052323.

Amin, Mohammad H. y col. (mayo de 2018). "Quantum Boltzmann Machine". En: Phys. Rev. X 8 (2), pág. 021050. DOI: 10.1103/PhysRevX.8.021050. URL: https://link.aps.org/doi/10.1103/PhysRevX.8.021050.

Biamonte, Jacob y col. (sep. de 2017). "Quantum machine learning". En: Nature 549.7671, págs. 195-202. ISSN: 1476-4687. DOI: 10.1038/nature23474. URL: https://doi.org/10.1038/nature23474.

Bravyi, Sergey y col. (mayo de 2008). "The Complexity of Stoquastic Local Hamiltonian Problems". En: Quantum Info. Comput. 8.5, págs. 361-385. ISSN: 1533-7146.

Brown, Sheldon y S. J. Barnett (ago. de 1952). "Carriers of Electricity in Metals Exhibiting Positive Hall Effects". En: Phys. Rev. 87 (4), págs. 601-607. DOI: 10.1103/PhysRev.87.601. URL: https://link.aps.org/doi/10.1103/PhysRev.87.601.

Burkov, A. (2019). The Hundred-Page Machine Learning Book. Andriy Burkov. ISBN: 9781999579517. URL: https://books.google.es/books?id=0jbxwQEACAAJ.

Chirkov, A. G. (feb. de 2001). "On the adiabatic theorem in quantum mechanics". En: Technical Physics Letters 27.2, págs. 93-96. ISSN: 1090-6533. DOI: 10.1134/1.1352758. URL: https://doi.org/10.1134/1.1352758.

D-Wave (2020). Getting Started with the D-Wave System. User Manual. URL: https://docs.dwavesys.com/docs/latest/doc_getting_started.html.

Dattani, Nikesh S., S. Szalay y N. Chancellor (2019). "Pegasus: The second connectivity graph for large-scale quantum annealing hardware". En: ArXiv abs/1901.07636.

Dawson, Christopher M. y Michael A. Nielsen (ene. de 2006). "The Solovay-Kitaev Algorithm". En: Quantum Info. Comput. 6.1, págs. 81-95. ISSN: 1533-7146.

Farhi, Edward y col. (2000). "Quantum computation by adiabatic evolution". En: arXiv preprint quant-ph/0001106.

Feldman, J. y R. Rojas (1996). Neural Networks: A Systematic Introduction. Springer Berlin Heidelberg. ISBN: 9783540605058. URL: https://books.google.es/books?id=txsjjYzFJS4C.

Géron, Aurélien (2017). Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. 1st. O'Reilly Media, Inc. ISBN: 1491962291.

Grant, Erica K. y Trav Humble (jul. de 2020). Adiabatic Quantum Computing and Quantum Annealing. DOI: 10.1093/acrefore/9780190871994.013.32. URL: https://oxfordre.com/physics/view/10.1093/acrefore/9780190871994.001.0001/acrefore-9780190871994-e-32.

Hebb, Donald Olding (2005). The organization of behavior: A neuropsychological theory. Psychology Press.

Hertz, J. y col. (ene. de 1991). Introduction To The Theory Of Neural Computation. Vol. 44. DOI: 10.1063/1.2810360.

Hinton, Geoffrey E. (ago. de 2002). "Training Products of Experts by Minimizing Contrastive Divergence". En: Neural Comput. 14.8, págs. 1771-1800. ISSN: 0899-7667. DOI: 10.1162/089976602760128018. URL: https://doi.org/10.1162/089976602760128018.

— (2012). "A Practical Guide to Training Restricted Boltzmann Machines". En: Neural Networks: Tricks of the Trade: Second Edition. Ed. por Grégoire Montavon, Geneviève B. Orr y Klaus-Robert Müller. Berlin, Heidelberg: Springer Berlin Heidelberg, págs. 599-619. ISBN: 978-3-642-35289-8. DOI: 10.1007/978-3-642-35289-8_32. URL: https://doi.org/10.1007/978-3-642-35289-8_32.

Hinton, Geoffrey E., Simon Osindero y Yee-Whye Teh (jul. de 2006). "A Fast Learning Algorithm for Deep Belief Nets". En: Neural Comput. 18.7, págs. 1527-1554. ISSN: 0899-7667. DOI: 10.1162/neco.2006.18.7.1527. URL: https://doi.org/10.1162/neco.2006.18.7.1527.

Hopfield, John (mayo de 1982). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". En: Proceedings of the National Academy of Sciences of the United States of America 79, págs. 2554-8. DOI: 10.1073/pnas.79.8.2554.

Ising, Ernst (feb. de 1925). "Beitrag zur Theorie des Ferromagnetismus". En: Zeitschrift für Physik 31.1, págs. 253-258. ISSN: 0044-3328. DOI: 10.1007/BF02980577. URL: https://doi.org/10.1007/BF02980577.

Johnson, M. W. y col. (mayo de 2011). "Quantum annealing with manufactured spins". En: Nature 473.7346, págs. 194-198. ISSN: 1476-4687. DOI: 10.1038/nature10012. URL: https://doi.org/10.1038/nature10012.

Kato, Tosio (1950). "On the Adiabatic Theorem of Quantum Mechanics". En: Journal of the Physical Society of Japan 5.6, págs. 435-439. DOI: 10.1143/JPSJ.5.435. eprint: https://doi.org/10.1143/JPSJ.5.435. URL: https://doi.org/10.1143/JPSJ.5.435.

Kempe, J., A. Kitaev y O. Regev (2006). "The Complexity of the Local Hamiltonian Problem". En: SIAM J. Comput. 35, págs. 1070-1097.

Kettering, C. F. y G. G. Scott (nov. de 1944). "Inertia of the Carrier of Electricity in Copper and Aluminum". En: Phys. Rev. 66 (9-10), págs. 257-267. DOI: 10.1103/PhysRev.66.257. URL: https://link.aps.org/doi/10.1103/PhysRev.66.257.

Kitaev, A. Yu., A. H. Shen y M. N. Vyalyi (2002). Classical and Quantum Computation. USA: American Mathematical Society. ISBN: 0821832298.

Lucas, Andrew (2014). "Ising formulations of many NP problems". En: Frontiers in Physics 2, pág. 5. ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005. URL: https://www.frontiersin.org/article/10.3389/fphy.2014.00005.

Marvian, Milad, Daniel A. Lidar e Itay Hen (abr. de 2019). "On the computational complexity of curing non-stoquastic Hamiltonians". En: Nature Communications 10.1, pág. 1571. ISSN: 2041-1723. DOI: 10.1038/s41467-019-09501-6. URL: https://doi.org/10.1038/s41467-019-09501-6.

Montúfar, Guido (2018). "Restricted Boltzmann Machines: Introduction and Review". En: Information Geometry and Its Applications. Ed. por Nihat Ay, Paolo Gibilisco y František Matúš. Cham: Springer International Publishing, págs. 75-115. ISBN: 978-3-319-97798-0.

Nielsen, Michael A. e Isaac L. Chuang (2010). Quantum Computation and Quantum Information: 10th Anniversary Edition. Cambridge University Press. DOI: 10.1017/CBO9780511976667.

Pathria, R.K. y Paul D. Beale (2011). "12 - Phase Transitions: Criticality, Universality, and Scaling". En: Statistical Mechanics (Third Edition). Ed. por R.K. Pathria y Paul D. Beale. Third Edition. Boston: Academic Press, págs. 401-469. ISBN: 978-0-12-382188-1. DOI: https://doi.org/10.1016/B978-0-12-382188-1.00012-8. URL: http://www.sciencedirect.com/science/article/pii/B9780123821881000128.

Troyer, Matthias y Uwe-Jens Wiese (mayo de 2005). "Computational Complexity and Fundamental Limitations to Fermionic Quantum Monte Carlo Simulations". En: Phys. Rev. Lett. 94 (17), pág. 170201. DOI: 10.1103/PhysRevLett.94.170201. URL: https://link.aps.org/doi/10.1103/PhysRevLett.94.170201.

vd, Sunny (nov. de 2012). URL: https://commons.wikimedia.org/wiki/File:Boltzmannexamplev1.png#filelinks.

Wendin, G (sep. de 2017). "Quantum information processing with superconducting circuits: a review". En: Reports on Progress in Physics 80.10, pág. 106001. DOI: 10.1088/1361-6633/aa7e1a. URL: https://doi.org/10.1088%2F1361-6633%2Faa7e1a.

Zhang, Jie y col. (nov. de 2018). "Realizing an adiabatic quantum search algorithm with shortcuts to adiabaticity in an ion-trap system". En: Phys. Rev. A 98 (5), pág. 052323. DOI: 10.1103/PhysRevA.98.052323. URL: https://link.aps.org/doi/10.1103/PhysRevA.98.052323.