

# TRABAJO DE FIN DE GRADO



Universidad  
Zaragoza



Facultad de Ciencias  
Universidad Zaragoza

UNIVERSIDAD DE ZARAGOZA

DEPARTAMENTO DE FÍSICA DE LA MATERIA CONDENSADA

---

INTELIGENCIA ARTIFICIAL APLICADA A LA  
MECÁNICA CUÁNTICA

Curso 2021 - 2022

---

*Autor:*

Inés SERRANO MAYOR

*Tutor:*

Dr. Luis MARTÍN MORENO

*Co-tutor:*

Eduardo SÁNCHEZ BURILLO

*Mi agradecimiento a los directores de este trabajo de fin de grado por su sabia guía, en especial, a Luis Martín Moreno por su tiempo, su ayuda y por compartir conmigo su saber.*

*Es el momento también de agradecer a mi familia por ser mi principal estímulo para llegar hasta aquí y apoyarme de forma permanente e incondicional.*

# Índice

<b>1. Introducción</b>	<b>1</b>
<b>2. Presentación de las herramientas relevantes utilizadas</b>	<b>2</b>
2.1. El Modelo cuántico de Hubbard . . . . .	2
2.1.1. Interacción entre las partículas y energía . . . . .	4
2.1.2. Características del sistema . . . . .	4
2.2. Redes neuronales . . . . .	5
2.2.1. Concepto . . . . .	5
2.2.2. Funcionamiento . . . . .	8
<b>3. Resultados</b>	<b>11</b>
3.1. Entrenamiento y overfitting . . . . .	11
3.2. Reconstrucción de funciones de onda para un número de partículas distinto	15
3.3. Compresión de la función de onda . . . . .	17
3.4. Precisión en función del porcentaje de datos de entrenamiento . . . . .	18
3.5. Dependencia de la energía en función de la interacción entre partículas	20
<b>4. Conclusiones</b>	<b>21</b>

# 1. Introducción

Las redes neuronales artificiales (*Artificial Neural Networks* en inglés) son capaces de aprender de forma similar a como lo hacen las redes neuronales de los organismos vivos: el conjunto de neuronas va creando y reforzando conexiones entre ellas a partir de la experiencia para aprender algo concreto. Estas redes no necesitan que les digan cómo solucionar el problema que se les presenta, sino que aprenden simplemente observando los datos proporcionados. Son capaces de identificar patrones o tendencias difíciles de determinar por otros métodos convencionales de computación.

Este aprendizaje automático a partir de un conjunto de datos dado es sorprendente. Hoy en día las redes neuronales han conseguido lograr un rendimiento excelente en campos como la visión artificial, el reconocimiento del habla y el procesamiento del lenguaje natural. De hecho, estas técnicas están siendo implementadas a gran escala en compañías como Google, Microsoft y Facebook [1]. Sin embargo, no ha sido hasta estos últimos años cuando estas redes han prosperado y se han utilizado de manera eficiente, principalmente debido a los requerimientos de memoria y cálculo.

Además de los campos mencionados en el párrafo anterior, las redes neuronales encuentran aplicaciones en muchos otros ámbitos. La física y, más en particular, la física cuántica, no son una excepción [2].

En este trabajo usaremos una red para encontrar el estado fundamental de un sistema cuántico de muchos cuerpos interactuantes. El objetivo del trabajo es responder a la pregunta de, si a partir de una red neuronal optimizada para un número dado de partículas, esta misma red es capaz de determinar con precisión el estado fundamental del mismo sistema pero con un número diferente de partículas. Como veremos, la respuesta es afirmativa. Esto hace que dispongamos de una herramienta muy potente para convertir problemas complejos en otros más simples, ahorrando tiempo de cálculo y memoria.

Por otro lado, mostraremos cómo podemos comprimir la información de nuestro sistema, siendo necesario únicamente un número reducido de parámetros para reproducir la función de onda deseada. Además, comprobaremos que no es necesario más que un pequeño porcentaje de datos para entrenar la red de forma que esta aprenda satisfactoriamente. Por último, estudiaremos la dependencia de la energía del estado fundamental de un sistema dado en función de un parámetro asociado a la energía de interacción entre dos partículas.

## 2. Presentación de las herramientas relevantes utilizadas

### 2.1. El Modelo cuántico de Hubbard

El modelo cuántico de Hubbard es el modelo más sencillo de interacción de partículas en una red. Técnicamente es una extensión del llamado modelo *tight-binding*, donde las partículas pueden saltar de un sitio a otro de la red sin interactuar con otras. En el caso más simple, una partícula en un sitio de la red solo puede saltar a sus primeros vecinos [3]. En el caso de que las partículas sean bosones, los saltos podrán ser a sitios ya ocupados por otros bosones. El Hamiltoniano del modelo de Hubbard añade un término adicional, introduciendo una cantidad de energía  $U$  para cada par de partículas ocupando el mismo sitio de la red, representando la repulsión Coulombiana.

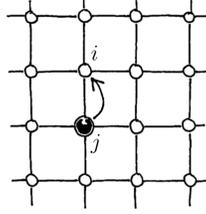


Figura 1: Salto de una partícula del sitio de la red  $j$  al  $i$  originado por el operador  $c_i^\dagger c_j$  en dos dimensiones [4]

El Hamiltoniano de Hubbard más simple en una dimensión usando la representación en segunda cuantización es de la forma [5]:

$$H = H_0 + H_{int} + H_{hop} \quad (2.1)$$

donde:

$$H_0 = \sum_{i=1}^M E_i n_i \quad (2.2)$$

$$H_{int} = \frac{U}{2} \sum_{i=1}^M n_i (n_i - 1) \quad (2.3)$$

$$H_{hop} = -t \sum_{i=1}^M c_i^\dagger c_{i+1} + c.c. \quad (2.4)$$

Donde  $E_i$  es la energía de una partícula en el sitio  $i$ ,  $-t$  es el parámetro conocido como término de "hopping", que da cuenta del salto de partículas entre sitios de la red y  $U$  es el parámetro que modula el término de interacción de Hubbard.

Por otro lado, los operadores creación y destrucción ( $c_i$  y  $c_i^\dagger$ ) nos permiten describir los saltos de las partículas a izquierda y a derecha.

Actúan de la siguiente manera sobre los estados en una dimensión:

$$c_i |n_1 n_2 \dots n_M \rangle = \sqrt{n_i} |n_1 \dots n_i - 1 \dots n_M \rangle \quad (2.5)$$

$$c_i^\dagger |n_1 n_2 \dots n_M \rangle = \sqrt{n_i + 1} |n_1 \dots n_i + 1 \dots n_M \rangle \quad (2.6)$$

siendo  $|n_1 n_2 \dots n_M \rangle$  un estado del sistema con  $M$  sitios y  $N$  partículas. Cada  $n_i$  representa el número de partículas que hay en el sitio  $i$ , de forma que:  $\sum_i n_i = N$ .

En el caso de la red unidimensional las condiciones de contorno pueden ser periódicas o no periódicas (Figura 2). En el caso de las periódicas, el salto a derecha ( $c_{i+1}^\dagger c_i$ ) de una partícula que está en la posición  $M$  (la 'última') es a la posición  $M+1$  que coincide con la 1 y el salto a izquierda ( $c_i^\dagger c_{i+1}$ ) de una partícula en la posición 1 es a la posición 'última'  $M$ .

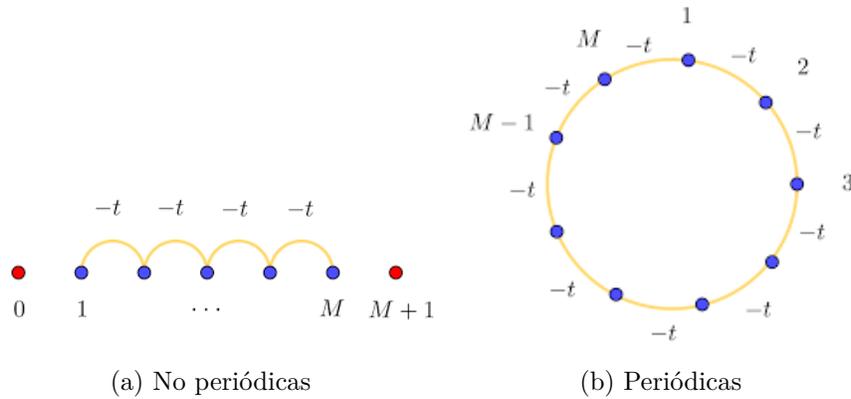


Figura 2: Salto de las partículas en una red unidimensional con condiciones de contorno (a) no periódicas y (b) periódicas.  $M$  es el número de sitios y  $-t$  el parámetro de salto.

### 2.1.1. Interacción entre las partículas y energía

Como ya hemos mencionado en el apartado anterior (2.1),  $U$  es el término de energía de interacción entre dos partículas y  $t$  es el parámetro que caracteriza el salto de una partícula de un sitio a otro de la red (posee unidades de energía). Por tanto, teniendo en cuenta estos parámetros, podemos distinguir dos situaciones: cuando la interacción entre las partículas es débil ( $U \ll t$ ) y cuando es fuerte ( $U \gg t$ ). Presentando una interacción atractiva entre las partículas cuando  $U < 0$  y repulsiva cuando  $U > 0$  [6].

El caso en el que no existe interacción entre las partículas, es decir, en el que  $U = 0$ , corresponde con el modelo de ligaduras fuertes (*tight binding* en inglés) de aproximación a primeros vecinos. Por tanto, en este caso la energía del estado fundamental de una partícula en un sistema unidimensional sigue la expresión:

$$E = E_0 - 2J\cos(ka) \quad (2.7)$$

donde  $a$  es el parámetro de red,  $k$  el vector de ondas,  $J$  es el término *hopping* ( $t$  en 2.4) y  $E_0$  es la energía de una partícula en su sitio ( $E_i$  en 2.2) [7].

### 2.1.2. Características del sistema

Utilizaremos un sistema caracterizado por el Hamiltoniano del modelo de Hubbard (2.1) con 7 sitios ( $M=7$ ) y un número de partículas fijo  $N$ . Nótese que, al trabajar con bosones, puede haber más de una partícula en cada sitio. Así, la dimensión del espacio de Hilbert crecerá rápidamente con el número de partículas  $N$ .

Para hallar las energías de nuestro sistema será necesario el cálculo del Hamiltoniano en la base de estados con el número de partículas que nos interese y así obtener la matriz hamiltoniana que podremos diagonalizar. Con esto podremos obtener las funciones de onda correspondientes a los estados de distintas energías del sistema.

Todos los cálculos se han hecho con la elección de los siguientes parámetros para el Hamiltoniano:  $E_i = 1$ ,  $U = 2$ ,  $t = 1$  y condiciones de contorno periódicas (excepto que se indique explícitamente lo contrario).

## 2.2. Redes neuronales

Como ya hemos dicho en la introducción, las redes neuronales son capaces de aprender patrones a partir de datos, sin necesidad de explicarles ninguna regla. En esta sección ahondaremos un poco más en el concepto.

### 2.2.1. Concepto

Para explicar el funcionamiento de una red neuronal definimos primero el concepto de perceptrón. Un perceptrón es una neurona artificial que admite varias entradas binarias  $x_1, x_2, \dots, x_n$  y produce una única salida binaria  $y$ .

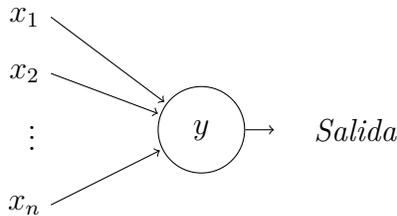


Figura 3: Esquema de funcionamiento de un perceptrón.

Para explicar cómo se decide el valor de la salida necesitamos introducir un nuevo parámetro que son los pesos,  $w_j$  (del inglés *weights*), estos indican la importancia que tienen las entradas para la salida. El valor de la salida vendrá determinado por si la suma  $\sum_j w_j x_j$  es mayor o menor que una cierta cantidad límite que llamaremos umbral. Formalmente [1]:

$$salida = \begin{cases} 0, & \text{si } \sum_j w_j x_j \leq \text{umbral} \\ 1, & \text{si } \sum_j w_j x_j > \text{umbral} \end{cases}$$

Llamando:  $w \cdot x = \sum_j w_j x_j$  y denotando el umbral o sesgo como  $-b$  (del inglés *bias*):

$$salida = \begin{cases} 0, & \text{si } w \cdot x + b \leq 0 \\ 1, & \text{si } w \cdot x + b > 0 \end{cases}$$

Esta definición puede verse como una transformación lineal con los parámetros  $w$  y  $b$  seguida de la función escalón (figura 5a), que es una función no lineal con altura 1 para valores positivos y 0 en otro caso.

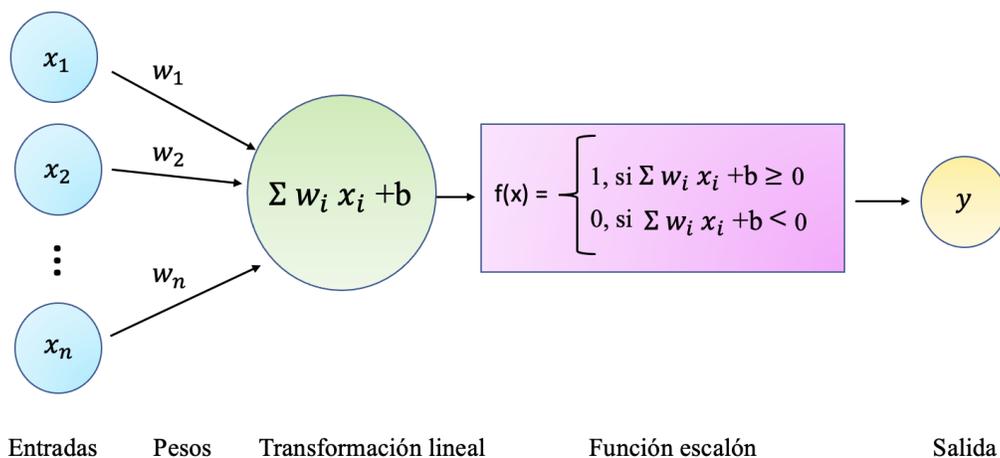


Figura 4: Esquema de las transformaciones que realiza un perceptrón desde las entradas  $x_i$  hasta la salida  $y$ . La transformación del círculo verde es lineal y la del rectángulo rosa no lineal (dada por la función escalón).

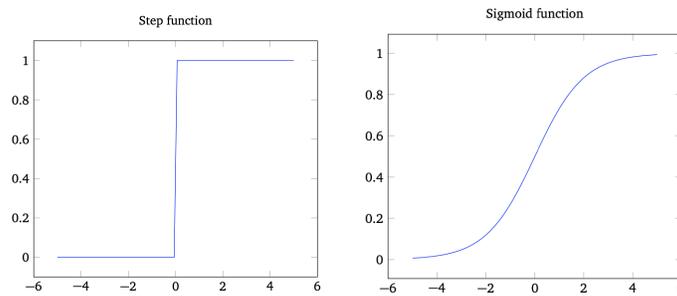
Podemos ver el perceptrón como un modelo que toma decisiones teniendo en cuenta los pros y contras. Ponemos un ejemplo para aplicar esta idea. Supongamos que este fin de semana te invitan a una fiesta, todavía no sabes si vas a ir y tienes que tomar una decisión al respecto. Para ello tendrás en cuenta tres factores: si hace buen tiempo, si tus amigos van a la fiesta y si el sitio donde se celebra está cerca de tu casa. Cada uno de estos factores los podemos representar con una variable binaria:  $x_1, x_2$  y  $x_3$  respectivamente. De forma que, por ejemplo,  $x_1 = 1$  si hace buen tiempo y  $x_1 = 0$  en caso contrario. Podemos dar ahora valores a los pesos en función de lo que consideras más o menos importante. Por ejemplo podría ser  $w_1 = 2$ ,  $w_2 = 6$  y  $w_3 = 2$  de forma que el hecho de que tus amigos vayan o no influye más que el resto de factores.

Una vez entendida esta idea, supongamos que tenemos una red de perceptrones y queremos que nuestra red sea capaz de reconocer dígitos escritos a mano. Para esto, la red tendrá que jugar con los pesos y sesgos hasta que sea capaz de clasificar los dígitos. Pero el hecho de que la salida sea binaria dificulta el aprendizaje, ya que con un pequeño cambio en los pesos o sesgos se puede cambiar completamente el output de 0 a 1, por ejemplo. Por tanto, para conseguir que un pequeño cambio en los parámetros produzca únicamente una pequeña modificación en la salida necesitamos una nueva estructura, la llamada neurona sigmoide. Estas neuronas son similares a los perceptrones, pero en su caso las entradas  $x_1, x_2, \dots, x_n$  no serán solo 0 o 1, sino que podrán tomar cualquier

valor real entre 0 y 1. En la salida, ocurrirá lo mismo y esto será gracias a la función sigmoide (figura 5b) definida como:

$$\sigma(z) = \sigma(wx + b) = \frac{1}{1 + e^{-z}}$$

que tiende a 1 cuando  $z$  se acerca a  $\infty$  y a 0 cuando  $z$  se acerca a  $-\infty$ . Podemos ver entonces que la forma de la función escalón usada para el perceptrón y la sigmoide son similares, siendo la segunda más suave.



(a) Función escalón

(b) Función sigmoide

Un conjunto de neuronas de este tipo conforma lo que llamamos red neuronal. Las redes están formadas por varias capas con varias neuronas en cada una de ellas. La estructura principal es la siguiente: una capa de entrada, una capa de salida y una o varias capas intermedias, en el último caso se llama densa (ver figura 5).

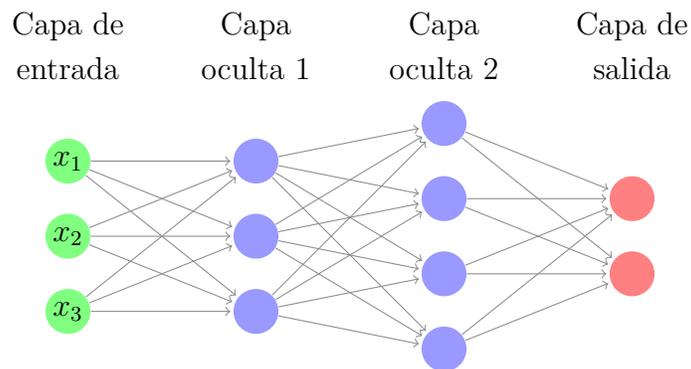


Figura 5: Estructura de una Red Neuronal Densa. En este caso la figura representa una función  $\mathbb{R}^3 \rightarrow \mathbb{R}^2$  con dos capas ocultas, la primera con 3 neuronas y la segunda con 4. Decimos en este caso que la red es del tipo 3-3-4-2. En esta arquitectura cada neurona está conectada con todas las de la capa anterior.

### 2.2.2. Funcionamiento

Existen distintos métodos de aprendizaje de la red y en este caso vamos a usar el llamado Aprendizaje Supervisado. En este método se dispone de un conjunto de datos  $x_i$  cada uno de ellos con una etiqueta  $y_i$  y la idea es que la red sea capaz de predecir la etiqueta dado un dato nuevo.

Así que la red tendrá que elaborar un algoritmo que encuentre los valores adecuados para los pesos y sesgos tal que la salida se acerque lo máximo posible a los valores de las etiquetas. Para cuantificar cómo de bien se está entrenando la red, usaremos una función de coste,  $C$ . Hay múltiples opciones para esta función, pero una de las más habituales es la del error cuadrático medio que viene dada por la expresión:

$$C = \frac{1}{2n} \sum_{i=1}^n (y_i - a_i(x_i))^2$$

siendo  $a_i(x_i)$  la predicción de la red neuronal para la variable  $y$  cuando los datos de entrada valen  $x_i$  y  $n$  el número total de entradas.

El objetivo de la red es variar  $w$  y  $b$  para minimizar la función coste,  $C$ . Esto lo haremos mediante el algoritmo de descenso de gradiente que explicaremos más adelante.

Primero veamos cómo se obtiene ese vector final  $y(x)$  de las salidas de la red. En las redes neuronales cada neurona actúa como una función donde se aplica primero una transformación lineal (a través de los parámetros  $w$  y  $b$ ) y después una no-lineal  $g(z)$ , que es la que se conoce como función de activación (de la misma manera que dijimos que funcionaba un perceptrón). Y así toda la red se construye mediante una aplicación sucesiva de funciones lineales y no-lineales. Por ejemplo, la activación  $a_j^l$  de la  $j$ -ésima neurona de la  $l$ -ésima capa está relacionada con la activación de la capa  $(l - 1)$  por la ecuación:

$$a_j^l = g \left( \sum_k w_{jk}^l a_k^{l-1} + b_j^l \right)$$

donde  $w_{jk}^l$  es el peso para la unión desde la  $k$ -ésima neurona en la capa  $(l - 1)$  a la  $j$ -ésima neurona en la capa  $l$ ,  $b_j^l$  es el *bias* de la  $j$ -ésima neurona en la capa  $l$  y la suma es sobre las  $k$  neuronas de la capa  $l$ .

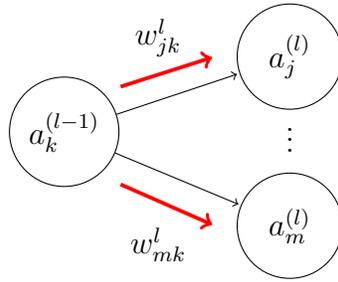


Figura 6: Notación de los parámetros en una Red Neuronal

Aunque la función de activación no lineal,  $g$ , podría ser cualquiera, hay unas pocas que son las que se usan normalmente en redes neuronales debido a los buenos resultados que proporcionan. Estas son la función sigmoide  $\sigma$  (explicada anteriormente), que se suele usar en la capa de salida, y la función RELU,  $RELU(x) = \max(0, x)$  (*Rectified Linear Unit* del inglés), o alguna similar, que se suele usar en las capas intermedias.

Como ya hemos dicho anteriormente, para minimizar la función de coste,  $C(x)$ , es muy común utilizar el método llamado Descenso por Gradiente. Este método consiste en calcular el gradiente de la función coste respecto a los parámetros de la red (pesos,  $w_k$ , y sesgos,  $b_l$ ), de forma que estos parámetros se van actualizando restándoles ese gradiente de la función coste multiplicado por un cierto número  $\eta$ , denominado tasa de aprendizaje (el cual se ajusta manualmente), hasta que la función coste alcanza un mínimo.

$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}$$

$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}$$

Computacionalmente, el método de descenso de gradiente es muy costoso por la cantidad de derivadas que conlleva (una por cada parámetro). Pero gracias a la técnica de *Back Propagation* se agilizan mucho los cálculos. Esta técnica consiste en computar derivadas desde la capa de salida a la de entrada, es decir, se empieza por el cálculo de derivadas de la función coste con respecto a los parámetros de más a la derecha y de ahí hacia la izquierda se calcula el resto de derivadas de forma recursiva.

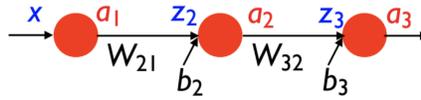


Figura 7

Si por ejemplo tenemos el esquema de la figura 7, podemos escribir:

$$z_2 = w_{21}a_1 + b_2$$

$$a_2 = \sigma(z_2)$$

$$z_3 = w_{32}a_2 + b_3$$

$$a_3 = \sigma(z_3)$$

$$c = C(a_3)$$

Para hallar la derivada de la función coste,  $C$ , con respecto a las variables  $w_{21}$  y  $b_2$ :

$$\frac{\partial C}{\partial w_{21}} = \frac{\partial C}{\partial z_2} \frac{\partial z_2}{\partial w_{21}} = a_1 \frac{\partial C}{\partial z_2}$$

$$\frac{\partial C}{\partial b_2} = \frac{\partial C}{\partial z_2} \frac{\partial z_2}{\partial b_2} = \frac{\partial C}{\partial z_2}$$

Partimos de las derivadas con respecto a los parámetros de más a la derecha y las vamos usando para calcular las siguientes, así hasta llegar a la que nos interesa:

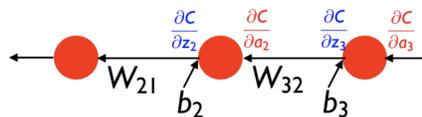


Figura 8

$$\begin{aligned}\frac{\partial C}{\partial a_3} &= a_3 - y \\ \frac{\partial C}{\partial z_3} &= \frac{\partial C}{\partial a_3} \frac{\partial a_3}{\partial z_3} = \sigma'(z_3) \frac{\partial C}{\partial a_3} \\ \frac{\partial C}{\partial a_2} &= \frac{\partial C}{\partial z_3} \frac{\partial z_3}{\partial a_2} = w_{32} \frac{\partial C}{\partial z_3} \\ \frac{\partial C}{\partial z_2} &= \frac{\partial C}{\partial a_2} \frac{\partial a_2}{\partial z_2} = \sigma'(z_2) \frac{\partial C}{\partial a_2}\end{aligned}$$

Por otro lado, grandes conjuntos de muestras pueden suponer un problema ya que puede derivar en un lento entrenamiento. En este caso se usa el descenso de gradiente estocástico. Esta técnica consiste en elegir de manera aleatoria un conjunto de  $m$  datos de entrenamiento  $X_1, X_2, \dots, X_m$  al que llamaremos *mini-batch* y estimar el gradiente  $\nabla C$  a través de el valor medio de  $\nabla C_{X_j}$ :

$$\nabla C \approx \frac{1}{m} \sum_{j=1}^m \nabla C_{X_j}$$

La suma es sobre todos los datos de entrenamiento  $X_j$  del *mini-batch*. Después se elige aleatoriamente otro *mini-batch* y se repite el proceso. Cuando ya no queden más datos de entrenamiento diremos que hemos completado una época. Y volveremos a empezar otra.

## 3. Resultados

### 3.1. Entrenamiento y overfitting

Una vez definido el concepto de red neuronal por completo, queremos ahora entrenarla para que sea capaz de construir la función de onda del estado fundamental de un sistema de partículas interactuantes. Elegimos un número de sitios  $M=7$  y un número de partículas  $N=7$  que nos proporciona un número de estados en la base suficientemente alto para poder alimentar la red. El aprendizaje de esta red va a ser supervisado. Por tanto, previamente al entrenamiento de la red escribimos un programa para obtener la base de estados de nuestro sistema en función del número de partículas  $N$  deseado y calculamos además las energías y funciones de onda de este.

La estructura de nuestra red es la siguiente: la capa de entrada, dos capas intermedias con función de activación RELU y la capa de salida con función de activación sigmoide  $\sigma$ . La notación que usaremos para indicar el número de neuronas que hay en cada capa será:  $M - n_1 - n_2 - 1$ , siendo M el número de neuronas en la entrada que es igual al número de sitios que hay en un estado, ya que van a ser los datos con los que vamos a alimentar la red. Por otro lado,  $n_1$  y  $n_2$  son el número de neuronas en las capas intermedias, que será un parámetro que tendremos que variar y elegir aquel con el que la red aprenda mejor y, por último, escribimos un '1', ya que la capa de salida estará formada por una única neurona que corresponderá con el valor de la  $i$ -ésima componente del vector función de onda del estado fundamental correspondiente al vector  $i$ -ésimo de la base de la entrada.

Escogemos una red totalmente conexas, es decir, que cada neurona de una capa está conectada a todas las neuronas de la siguiente capa.

Dividimos el conjunto de datos que le proporcionamos a la red en dos: datos de entrenamiento y datos de validación. Los datos de entrenamiento son los que se usarán para entrenar a la red y los de validación para comprobar la bondad del aprendizaje.

Inicializamos nuestra red seleccionando los parámetros según consideramos. Dibujando en una gráfica la función coste, podemos ver el aprendizaje de la red. Sin embargo, este gráfico no nos permite ver con claridad lo que ocurre cuando la función coste se acerca al 0. Para solucionar esto, representamos también la función coste a escala logarítmica. Variando los distintos parámetros de nuestra red y teniendo en cuenta estos gráficos conseguiremos entrenar la red.

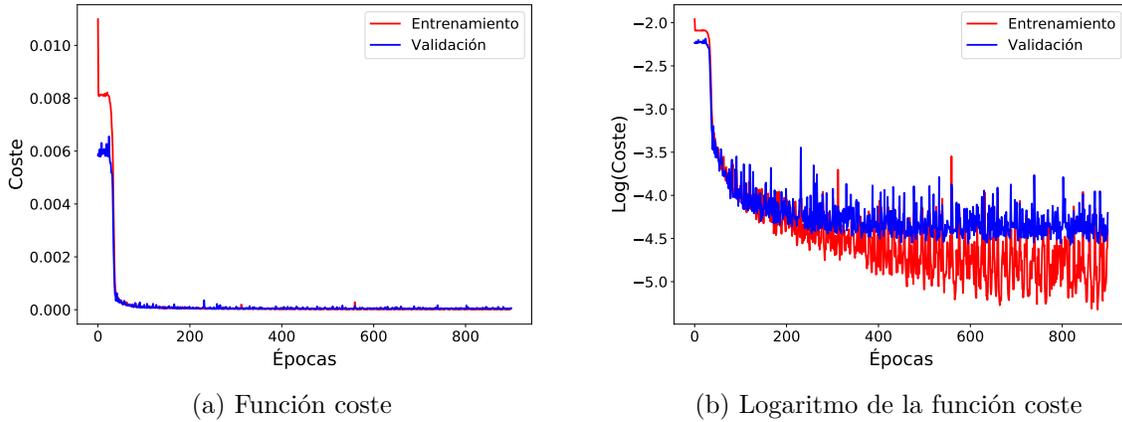


Figura 9:  $M=7$ ,  $N=7$ , red 7-60-60-1, 900 épocas,  $\eta = 0.002$ , tamaño del *mini-batch*=10, 80 % de los datos son de entrenamiento y 20 % de validación.

Observando la figura 9a, podemos pensar que los hiperparámetros están bien definidos y la red está aprendiendo. Esto es cierto, sin embargo, si nos fijamos ahora en la gráfica con la función en escala logarítmica (figura 9b), vemos que presenta un ruido considerablemente alto. Esto puede ser debido a que nuestro parámetro de aprendizaje,  $\eta$ , sea demasiado alto. Para entrenar esta red hemos usado un valor de  $\eta = 0,002$ , probamos por tanto a disminuir este parámetro.

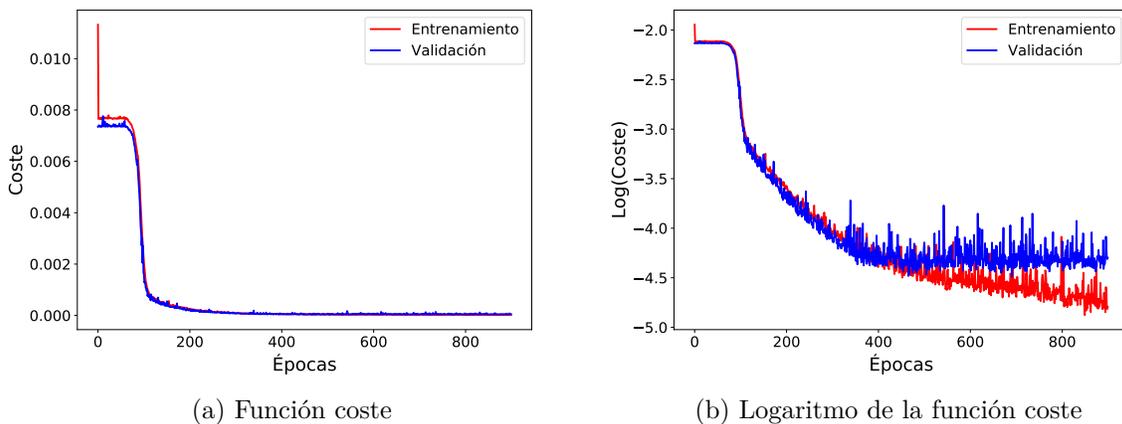


Figura 10:  $M=7$ ,  $N=7$ , red 7-60-60-1, 900 épocas,  $\eta = 0.0007$ , tamaño del *mini-batch*=10, 80 % de los datos son de entrenamiento y 20 % de validación.

Tomando entonces un valor de  $\eta = 0,0007$ , vemos en la gráfica 10b que disminuye el ruido de manera considerable. Sin embargo, observamos que la línea correspondiente a los datos de validación (azul) crece a partir de la época 400 aproximadamente y se separa de los datos del entrenamiento (roja), mientras esta sigue bajando. Estamos ante una situación llamada *overfitting* o *overtraining*.

El *overfitting* es un problema importante en las redes neuronales. Pero afortunadamente contamos con diversas técnicas para reducir los efectos de este. En general, una buena manera de reducir el *overfitting* es incrementando el número de datos con los que entrenamos la red. Sin embargo, en nuestro caso no podemos hacer esto ya que nuestros datos de entrenamiento son los estados de la base de nuestro sistema, que es un número fijo. Una cosa que sí podemos intentar es reducir el número de parámetros de nuestra red, ya que puede ser que tenga demasiados y esté aproximando muy bien los datos de entrenamiento, pero a la vez esté perdiendo la capacidad de generalización. Esto lo podemos conseguir disminuyendo el número de capas intermedias o disminuyendo el número de neuronas por capa. Elegimos la segunda opción y ponemos 35 neuronas por cada capa en lugar de 60 como teníamos anteriormente.

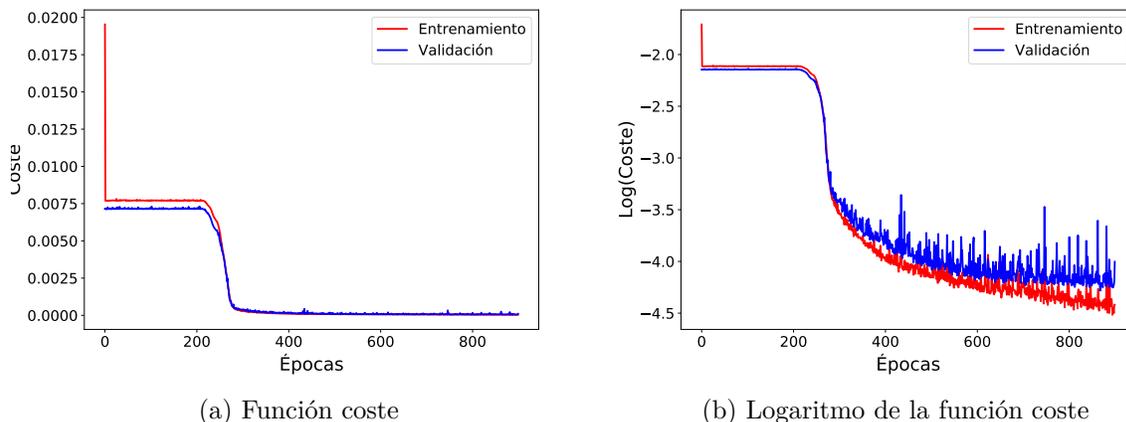


Figura 11:  $N=7$ ,  $M=7$ , red 7-35-35-1, 900 épocas,  $\eta=0.0007$ , tamaño del *mini-batch*=10, 80% de los datos son de entrenamiento y 20% de validación.

Con este último cambio conseguimos solucionar el *overfitting* que teníamos en el caso anterior, como se puede observar en la figura 11b. Si comparamos ahora las figuras 10 y 11, vemos que la función coste consigue llegar a valores más pequeños en la red de la figura 10 (7-60-60-1), lo cual indica que esta aprende mejor. Sin embargo, en este trabajo se ha escogido la red de la figura 11 (7-35-35-1) para realizar varios cálculos

ya que es más pequeña y no sesgaba los datos de entrenamiento frente a los de validación.

Otro parámetro que podemos tener en cuenta para ver la precisión de la función de onda del estado fundamental obtenida con la red optimizada es la fidelidad,  $f$ , definida como:

$$f = \frac{|\sum_n \psi^*(\mathbf{n})\psi_{exact}(\mathbf{n})|^2}{\sum_n |\psi(\mathbf{n})|^2}$$

donde  $\psi$  es la función de onda generada por la red neuronal y  $\psi_{exact}$  es la obtenida diagonalizando el hamiltoniano. Cuanto más cercano a 1 sea el valor de  $f$ , mayor será la precisión. Con el último ejemplo de entrenamiento de los datos el valor de la fidelidad obtenido es  $f = 0,9986$ , así que definitivamente podemos decir que nuestra red ha aprendido a reconstruir la función de onda buscada con una fidelidad excelente.

### **3.2. Reconstrucción de funciones de onda para un número de partículas distinto**

Una vez tenemos optimizada la red para un número de bosones  $N=7$ , queremos ver si ahora esta red es capaz de obtener también los estados fundamentales para un número de partículas  $N'$  distinto. Para comprobar si es capaz de reconstruir estas funciones nos fijaremos en la fidelidad.

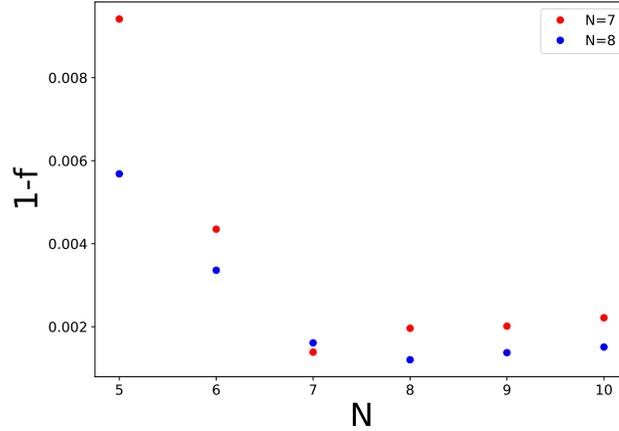


Figura 12: Representación de la fidelidad  $f$  en función del número de bosones  $N$ . En el caso  $N=7$ : red 7-35-35-1, 900 épocas,  $\eta = 0.0007$ , tamaño del *mini-batch*=10, 80% de los datos son de entrenamiento y 20% de validación. En el caso de  $N=8$ : red 7-50-50-1, 900 épocas,  $\eta = 0.0007$ , tamaño del *mini-batch*=10, 80% de los datos son de entrenamiento y 20% de validación. En ambos casos  $M=7$ .

Observando la figura 12 vemos que el valor de  $1-f$  es siempre menor que 0.01, lo que nos indica que esta red es capaz de encontrar la función de onda del estado fundamental para sistemas con distinto número de bosones. Sin embargo, la precisión de los estados para los  $N$ 's distintos para el que ha sido optimizado la red es siempre menor que la de este. Por ejemplo, en el caso en el que la red ha sido entrenada para  $N=7$  bosones, este tiene el valor más alto de la fidelidad. Pero, además, el valor de la fidelidad es alto para  $N=6$  o  $N=8$ . Estos resultados implican que una red optimizada para un número de parámetros puede generar estados fundamentales de un sistema de muchos cuerpos con alta precisión para un número distinto de parámetros. Este método nos permite extrapolar estados cuánticos.

A pesar de haber obtenido muy buenos resultados, nos podríamos preguntar qué hubiese pasado si en lugar de escoger una red totalmente conexa, hubiésemos usado una convolucional (en esta, cada neurona de una capa no tiene por qué estar conectada a todas las neuronas de la capa siguiente). De hecho, en [8] enfocaron este mismo problema para ambos casos y obtuvieron incluso mejores resultados con la red convolucional.

### 3.3. Compresión de la función de onda

La red neuronal usa una serie de parámetros para entrenarse. Este número de parámetros es generalmente menor a los que se necesitan para reproducir una función de onda. Por ejemplo, para generar el estado fundamental de nuestro sistema con 7 bosones necesitaríamos 1716 parámetros (dimensión del espacio de Hilbert de 7 bosones en 7 sitios). Sin embargo, cuando hemos entrenado la red en el apartado (3.1), esta ha usado 1576 parámetros para construirla. Esto implica que gracias a las redes neuronales podemos comprimir la información de manera eficiente.

Por tanto, lo interesante ahora es ver cuánto podemos comprimir la información, es decir, cuál es el límite de parámetros que necesitamos de forma que la red sea capaz de reconstruir la función de onda. Para esto vamos a jugar con el número de parámetros de la red cambiando el número de neuronas por capa. Hacemos este proceso tanto para una red con dos capas intermedias como para una red con una sola capa intermedia.

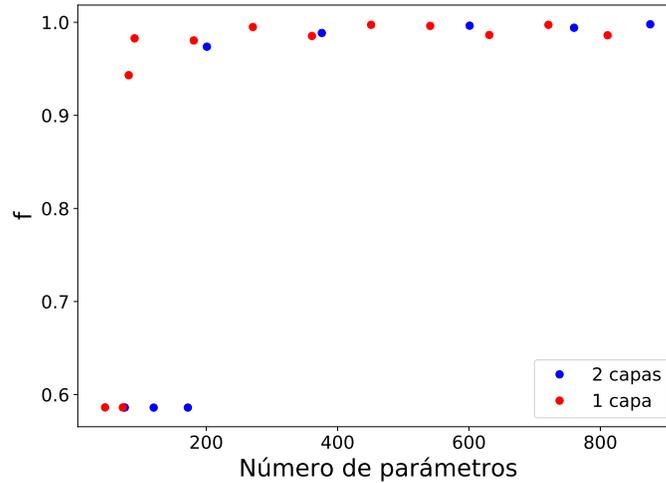


Figura 13: Fidelidad  $f$  en función del número de parámetros de la red.  $M=7$ ,  $N=7$ , 900 épocas,  $\eta=0.0009$ , tamaño del *mini-batch*=10, 80% de los datos son de entrenamiento y 20% de validación

En los resultados obtenidos (figura 13) vemos que el número de parámetros con el que la red de dos capas deja de aprender es 201 (correspondiente a 15 neuronas en cada capa). Al poner 171 parámetros (correspondiente a 9 neuronas en cada capa) la red aprende peor. Por otro lado, en el caso de la red con una única capa intermedia vemos

que con 91 parámetros (10 neuronas en la capa) la red sigue aprendiendo de forma considerablemente buena. Cuando disminuimos el número de parámetros a 82 (9 neuronas en la capa) la red ya no aprende tan bien. Por tanto, concluimos que la red neuronal necesita mucho menos parámetros de los que necesitaríamos para determinar la función de onda sin ella. Esto nos permite comprimir bastante la información necesaria. Por otro lado, los resultados también nos dicen que se necesitan menos parámetros para el caso en el que la red es más sencilla y contiene una sola capa que para cuando es más compleja y contiene dos capas.

### 3.4. Precisión en función del porcentaje de datos de entrenamiento

Como ya hemos comentado previamente, los datos están divididos en datos de entrenamiento y datos de validación. La red aprende las correlaciones de la función de onda a partir de los datos que le damos para el entrenamiento. Así que otra pregunta que nos podemos hacer ahora es qué porcentaje de los datos necesita la red que le enseñemos para entrenarse y que la precisión del resultado sea buena. Para responder a esta pregunta calculamos la fidelidad de los estados obtenidos tras entrenar la red para un porcentaje distinto de datos de entrenamiento con respecto a los datos totales, usando el resto para la parte de validación.

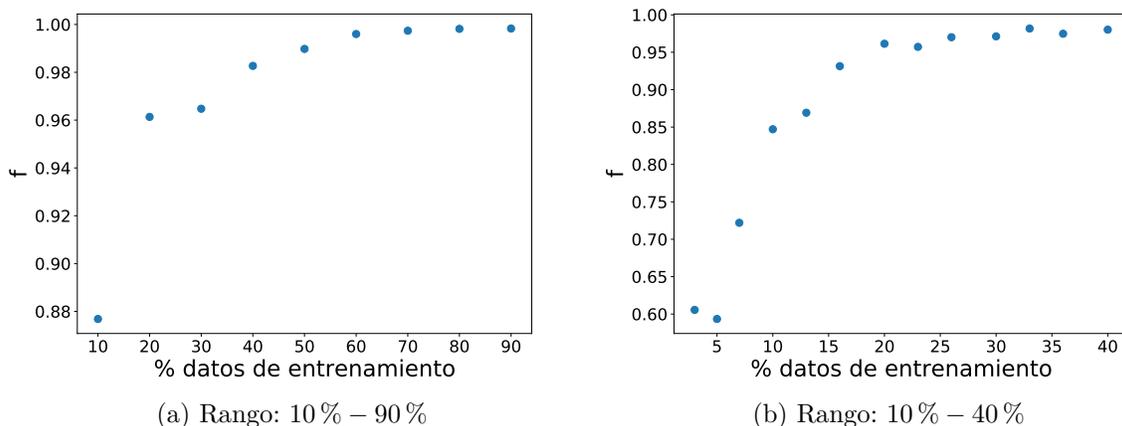


Figura 14:  $N=7$ ,  $M=7$ , red 7-25-25-1, 900 épocas,  $\eta = 0.0009$ , tamaño del *mini-batch*=10.

Hemos comenzado usando un 90% de los datos para entrenamiento y un 10% para

validación, y se ha ido bajando el porcentaje de los datos de entrenamiento con el objetivo de encontrar el punto en el que empeora la fidelidad. Según los datos obtenidos, recogidos en la figura 14a, ese punto empieza aproximadamente cuando se toma el 40 % de los datos para entrenamiento, aunque sigue presentando una fidelidad muy alta. Para observar bien esta transición nos centramos en el intervalo 0 % – 40 %, figura 14b. En este gráfico se aprecia mejor que es a partir del 20 % cuando la fidelidad comienza a bajar de forma considerable hasta un valor de 0.6, para un 2,5 %, aproximadamente.

A continuación repetimos este proceso para un número distinto de bosones,  $N=8$ .

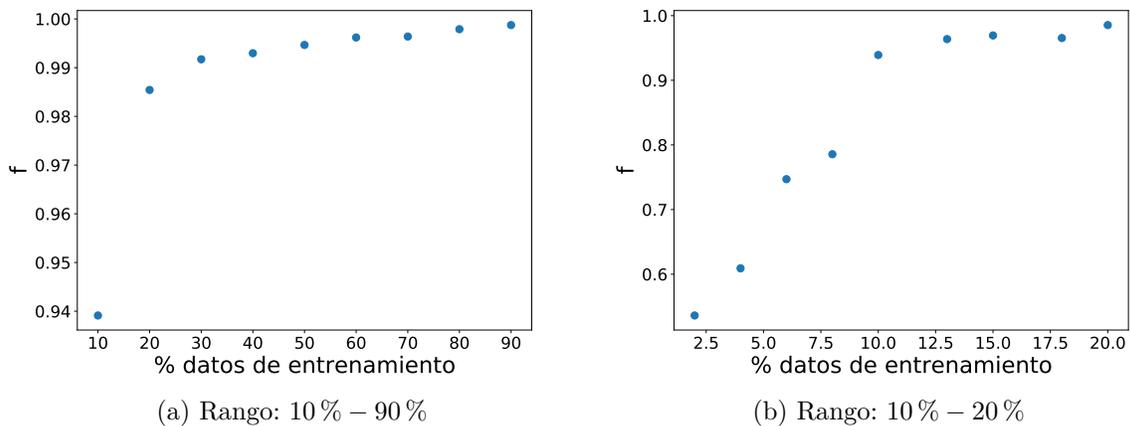


Figura 15:  $N=8$ ,  $M=7$ , red 7-50-50-1, 900 épocas,  $\eta = 0.0007$ , tamaño del *mini-batch*=10.

En este caso, mirando la figura 15a, vemos que el valor en el que la fidelidad comienza a disminuir es en torno al 20 %. De la misma manera que hemos hecho anteriormente, nos centramos en el intervalo de porcentajes en el que la fidelidad disminuye, en este caso es 10 % – 20 %, figura 15b. A partir de estos resultados vemos que en 10 % el valor de la fidelidad comienza a bajar hasta llegar a casi un valor de 0.5, para un 2,3 %.

A partir de estos resultados vemos como la red no necesita un porcentaje elevado en los datos de entrenamiento para sacar resultados satisfactorios.

### 3.5. Dependencia de la energía en función de la interacción entre partículas

Acabamos nuestro trabajo con un estudio de la dependencia de la energía del estado fundamental de un sistema en función del término de interacción entre las partículas.

Para ello, calculamos la energía del nivel fundamental,  $E$ , de la siguiente manera:

$$E = \langle \psi | H | \psi \rangle \quad (3.1)$$

donde  $|\psi\rangle$  es la función de onda correspondiente al nivel fundamental predecida por la red para un valor fijo de  $U$  y  $H$  es el Hamiltoniano del modelo de Hubbard (2.1) para un sistema con condiciones de contorno periódicas sobre una base de estados formados por  $M=7$  huecos y  $N$  partículas.

Por tanto, dado un  $N$  entrenamos la red para cada valor de  $U$ , obteniendo así la función de onda  $|\psi\rangle$  para calcular la energía  $E$  correspondiente.

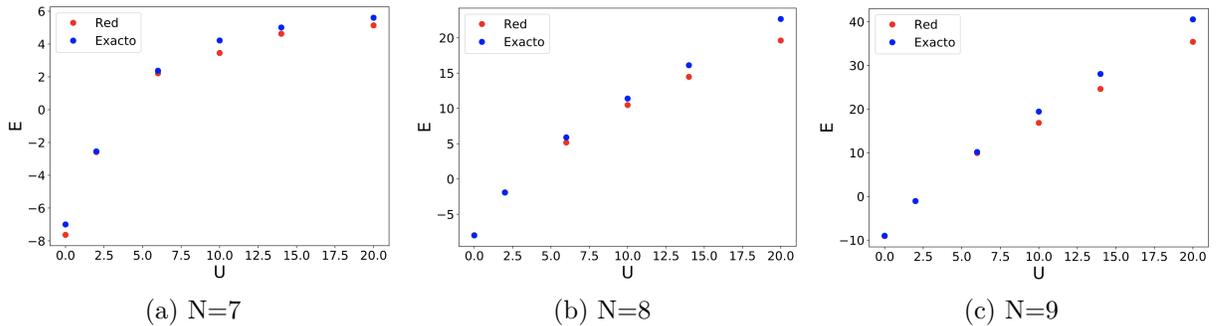


Figura 16: Energía del estado fundamental  $E$  frente al parámetro de interacción  $U$  para (a) 7 bosones, (b) 8 bosones y (c) 9 bosones.

Mirando los 3 gráficos (figura 16) comprobamos que la energía del nivel fundamental aumenta conforme aumenta el parámetro de interacción  $U$ .

Para entender los resultados obtenidos, nos fijamos primero en el caso de  $N=8$  bosones. Si el parámetro  $U$  es muy grande con respecto del parámetro  $t$  (en nuestro caso  $t = 1$ ), el término del hamiltoniano (2.1) correspondiente a este último (2.4) se puede tratar como una perturbación y, por tanto, su contribución a la energía es prácticamente despreciable. Al tener  $U$  un valor elevado, las partículas tenderán a estar lo más alejadas posibles unas de otras con el objetivo de minimizar la energía [6]. Así, para  $U$

grande, el nivel de menor energía (fundamental) en este sistema con 7 sitios y 8 bosones será el de un bosón en cada sitio a excepción de un sitio que tendrá dos bosones. Es decir, la energía del nivel fundamental  $E$  será aproximadamente  $U$ , correspondiente a la energía de interacción de un par de bosones. Esto se comprueba en el gráfico 16b, por ejemplo, para  $U = 20$  la energía es  $E \approx 20$ .

Con este mismo razonamiento, en el caso de 9 bosones en 7 sitios tendremos dos pares de bosones con energía  $U$  cada uno. Si  $U$  es grande, la energía será aproximadamente  $2U$ . Lo comprobamos en el gráfico 16c donde para  $U = 20$  la energía es  $E \approx 40$ .

Por último, en el caso de 7 bosones, habrá 0 pares de bosones ya que los 7 bosones se pueden colocar en cada uno de los 7 sitios. Por tanto, en este caso la energía,  $E$ , será bastante menor que  $U$  ya que no hay pares de bosones.

En el caso de  $U = 0$ , como hemos explicado en el apartado (2.1.1), la energía para una partícula viene dada por la expresión (2.7). Por tanto, con los parámetros que hemos tomado en nuestro sistema ( $E_0 = 1$  y  $t = J = 1$ ), la energía será -1. La energía para  $U = 0$  será por tanto aproximadamente -7 en el caso de 7 bosones, -8 en el caso de 8 y -9 en el de 9, como se comprueba en la figura 16.

## 4. Conclusiones

En este trabajo hemos hecho uso de la inteligencia artificial en el campo de la física cuántica. Hemos conseguido que una red neuronal aprenda a reconstruir la función de onda del nivel fundamental de un sistema de bosones dado aprendiendo a partir de los datos que le hemos proporcionado (los estados de la base). La red ha sido, por tanto, capaz de identificar los patrones necesarios para aprender de forma automática.

El primer objetivo de este trabajo era comprobar si con una red entrenada para un número de bosones  $N$ , podíamos también obtener resultados para un número de bosones distinto  $N'$ . Para ello, tras haber aprendido el funcionamiento de estas redes y a identificar los problemas que podían dificultar un buen entrenamiento, hemos conseguido entrenar la red para un sistema concreto con  $N$  bosones. Después, cambiando el número de bosones y utilizando esta misma red entrenada hemos comprobado cómo de buenos eran estos resultados a través de la fidelidad. De forma sorprendente, hemos obtenido resultados excelentes, pudiendo afirmar que la red sí es capaz de aprender las funciones de onda del nivel fundamental cuando le cambiamos el número de partículas.

Sin embargo, debido a las limitaciones tecnológicas, solamente lo hemos podido hacer para sistemas 'pequeños'. Por ejemplo, hemos visto que, si entrenábamos la red para  $N=7$ , esta red nos servía para  $N=6$  o  $N=8$ , pero ¿y para  $N=1000$ ? Para responder a esto necesitaríamos un ordenador mucho más potente y, si la respuesta fuese afirmativa, esto sería de extraordinario valor.

Por otro lado, aprovechando nuestra red neuronal construida, hemos querido ponerla a prueba. Primero, al ver que la red utiliza un número de parámetros menor para construir la función de onda del que necesitaríamos si no tuviésemos la red, hemos querido ver cuál era ese límite inferior de parámetros necesarios. Partiendo de un punto en el que sabemos que la red aprende, hemos ido disminuyendo el número de parámetros de la red jugando con las capas y el número de neuronas por capa. Tras este proceso, hemos observado que la red necesita muy pocos parámetros para funcionar bien. Lo que da lugar a una compresión de la información que puede llegar a ser muy útil. Además, esto nos lleva a darnos cuenta de que no es necesario una red muy grande para obtener buenos resultados.

La segunda prueba a la que hemos sometido a la red ha sido ir quitándole un cierto porcentaje de los datos totales que usa para el entrenamiento con el objetivo de encontrar cuál es este límite. Para ello, hemos comenzado con un porcentaje alto de datos de entrenamiento y lo hemos ido disminuyendo poco a poco. Se ha observado que no necesita gran porcentaje de datos para conseguir reconstruir la función de onda.

Por último, hemos hecho un estudio de la variación de la energía de este estado fundamental en función del parámetro de interacción entre las partículas. Hemos comprobado como los resultados coinciden con los esperados según los fundamentos físicos de estos sistemas. El estudio de esta dependencia se podría haber hecho sin el uso de las redes neuronales (de hecho, también lo hemos realizado de esta manera), puesto que, al no ser las dimensiones demasiado grandes, se puede diagonalizar la matriz hamiltoniana fácilmente. Sin embargo, si hubiésemos escogido un sistema con unas dimensiones mayores, el cálculo exacto a través de la diagonalización no hubiese sido posible y solo habríamos podido recurrir a las redes neuronales para la obtención de estos resultados de la energía.

Tras la realización de este trabajo se puede ver lo potente que es esta herramienta incluso en esta disciplina tan misteriosa para muchos como es la física cuántica. El desarrollo de estas redes va en aumento y estos algoritmos ya están revolucionando la tecnología.

## Referencias

- [1] M. A. Nielsen, *Neural Networks and Deep Learning*. (Determination Press, 2018). URL <http://neuralnetworksanddeeplearning.com/>.
- [2] V. Dunjko and H. J. Briegel, Machine learning & artificial intelligence in the quantum domain: a review of recent progress, *Reports on Progress in Physics*. **81** (7), (2018). URL <https://doi.org/10.1088/1361-6633/aab406>.
- [3] The hubbard model at half a century, *Nature Physics*. **9**(9), (2013). URL <https://doi.org/10.1038/nphys2759>.
- [4] T. Lancaster and S. J. Blundell, *Quantum Field Theory for the Gifted Amateur*. (Oxford University Press, Apr. 2014). URL <https://doi.org/10.1093/acprof:oso/9780199699322.001.0001>.
- [5] N. Majlis, *Quantum Theory Of Magnetism, The (2nd Edition)*. (World Scientific Publishing Company, 2007). URL <https://books.google.es/books?id=uA08DQAAQBAJ>.
- [6] D. I. Khomskii, *Basic aspects of the quantum theory of solids: order and elementary excitations*. (Cambridge university press, 2010).
- [7] M. Roy, The tight binding method, *Rutgers University*. **5**, (2015).
- [8] H. Saito and M. Kato, Machine learning technique to find quantum many-body ground states of bosons on a lattice, *Journal of the Physical Society of Japan*. **87** (1), (2018). URL <https://doi.org/10.7566/jpsj.87.014001>.