Universidad ORT Uruguay

Facultad de Ingeniería

Redes neuronales cuánticas

Entregado como requisito para la obtención del título de Ingeniera en Electrónica

Autor

Carolina Allende Amen - 233383

Tutores

André Fonseca De Oliveira

Efrain Buksman Hollander

2021

Declaración

Yo, Carolina Allende Amen, declaro que el trabajo que se presenta en esa obra es de mi propia mano. Puedo asegurar que:

- La obra fue producida en su totalidad mientras realizaba el proyecto integrador de Ingeniería en Electrónica;
- Cuando he consultado el trabajo publicado por otros, lo he atribuido con claridad;
- Cuando he citado obras de otros, he indicado las fuentes. Con excepción de estas citas, la obra es enteramente mía;
- En la obra, he acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, he explicado claramente qué fue contribuido por otros, y qué fue contribuido por mi;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Carolina Allende

05 de Febrero de 2021

Para Gilda Amen, Humberto Allende, Guillermo Pena y Emmett

Agradecimientos

A Rocío López, Paola Massonier, Lucía Sirio y Daniel López por su apoyo incansable en este trayecto

Abstract

En este trabajo se presenta una propuesta e implementación de una estructura basada en circuitos cuánticos que utiliza técnicas de redes neuronales clásicas para la resolución de un problema cuántico. El estado del arte en la computación cuántica en este momento es tal que la programación de las computadoras cuánticas actuales se basa aun en ordenar compuertas básicas. Si bien los postulados de la mecánica cuántica garantizan que cualquier operación unitaria es una operación admisible en un sistema cuántico, aún no se cuenta con un método sistemático para la traducción de estas operaciones a compuertas básicas. Mediante el uso de una red neuronal multicapa con una unidad básica formada por compuertas unitarias generales y CNOTs, este trabajo pretende ofrecer una solución al problema planteado utilizando técnicas de machine learning, especificamente una red neuronal multicapa $feedforward$ con aprendizaje supervisado mediante el algoritmo de máximo descenso. A su vez, el aprendizaje de la red está asistido por el uso de dos tipos de correlaciones cuánticas, *Mutual Information (MI)* y *Cumulative Correlation Measure (CCM)* para determinar cuántos CNOTs requiere el algoritmo, y entre qué qubits deben estar. Los resultados muestran que el modelo de la red es adecuado para la predicción de diversos algoritmos con tanta precisión como el usuario pueda desear.

Palabras clave

Computación cuántica, Redes Neuronales, Machine Learning.

Indice

Introducción

A fines del siglo XX surgió un nuevo paradigma de la computación que se basa en explotar propiedades cuánticas para lograr resolver problemas en los cuales las computadoras clásicas son ineficientes, al menos en teoría (1). Existen un puñado de algoritmos prometedores (2) (3) (4), pero esta rama de la computación todavía está en fase experimental dado a que la construcción del hardware es costosa y delicada, puesto que las tecnologías actuales requieren que la computadora se mantenga a unas milésimas de Kelvin por encima de cero absoluto.

Otra de las grandes limitantes actuales es que el estado del arte a 2021 aun se basa en programar armando cascadas de compuertas, de manera muy similar a los orígenes de la electrónica digital. Aún se están estudiando lenguajes de programación de alto nivel, y los de bajo de nivel disponibles cuentan únicamente con instrucciones para montar series de compuertas. Si bien los postulados de la mecánica cuántica que rigen la evolución de los sistemas de qubits pautan que cualquier operación unitaria puede modificar su evolución, al día de hoy, se ha estudiado poco la manera sistemática de descomponer una matriz unitaria en bloques básicos de manera de poder correr cualquier algoritmo en una computadora cuántica. Por esto, lograr traducir matrices unitarias en bloques sencillos es de gran interés y es el problema que trata este trabajo.

Aplicando técnicas de aprendizaje automático con propagación hacia adelante únicamente, se entrena a una red cuántica neuronal, a partir de conjuntos de entrada y de salida de entrenamiento, donde la neurona básica se define a partir de un conjunto de compuertas cuánticas, que son generadores universales de cualquier operación unitaria. A su vez, se explotan propiedades cuánticas, específicamente el entrelazamiento y su correspondiente cambio de correlación, para hacerla optimizar los recursos, o sea menos costosa computacionalmente.

Los resultados obtenidos son prometedores: la red logra aprender y por tanto tradu-

cir un conjunto variado de operaciones cuánticas, incluyendo al algoritmo de Deutsch (5). Además, se descubrieron casos en que la red falla y se ofrecen potenciales soluciones para el desempeño en estos casos.

Capítulo 1: Introducción a la computación cuántica

1.1 Postulados

La computación cuántica se basa en cuatro postulados. (6)

- Postulado 1: Definición de un qubit o quantum bit
- Postulado 2: Como evolucionan los qubits
- Postulado 3: Como la medida afecta a los qubits
- Postulado 4: Como los qubits se combinan para crear sistemas de varios qubits.

1.2 Postulado 1: El qubit

Primer postulado:

"Todo sistema físico aislado tiene asociado un espacio vectorial complejo con un producto interno (i.e. un espacio de Hilbert) conocido como espacio de estados del sistema. El sistema está completamente definido por su vector de estados, que es un vector unitario en el espacio de estados del sistema" [7]

La unidad básica clásica que se utiliza para representar los datos es el bit y formando arreglos de estos se crean las memorias clásicas. El bit clásico puede tomar uno de dos valores: 1 o 0. En la computación cuántica el bit tiene su análogo: el qubit (quantum bit). Desde un punto de vista físico, el qubit está asociado al estado del spin de un electrón que puede tomar valores 1 (down) o cero (up). En computación cuántica se usa una notación especial denominada notación de Dirac en honor al físico matemático del mismo apellido. Escencialmente, es una herramienta para representar vectores en el espacio de Hilbert. Por ejemplo, el estado de un qubit $|0\rangle$ está dado por:

$$|0\rangle = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} \tag{1}$$

Mientras que el $|1\rangle$ está dado por:

$$|1\rangle = \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix} \tag{2}$$

Los estados $|0\rangle$ y $|1\rangle$ son conocidos como base del espacio de estados de un qubit, es decir, cualquier vector de estados de un qubit puede ser escrito como una combinación lineal de la base, donde esta es el generador minimo del estado. A estas combinaciones lineales se las llama superposición en computación cuántica, y es esta propiedad la diferencia central

con un sistema de computación clásico. Por ejemplo, el estado presentado a continuación es un estado válido de un sistema de un qubit.

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{3}$$

donde α, $\beta \in \mathrm{C}$ y se les conoce como amplitudes.

En computación clásica el valor de los bits queda determinado de forma precisa, es decir, el dato no depende de quien lee la memoria. En cambio, en general, en computación cuántica no se puede recuperar el estado exacto: al medir un estado cuántico se destruye la superposición, y se obtiene uno de los estados de la base. Para ejemplificar, en el caso del estado $|\Psi\rangle$, al medir se obtiene el estado $|0\rangle$ con una probabilidad $|\alpha|^2$ y el estado $|1\rangle$ con una probabilidad $|\beta|^2$. Pero, generalmente, no es posible recuperar estas amplitudes y por tanto, no es posible saber el estado exacto en que se encuentra el qubit. El cuadrado de los módulos de α y β son probabilidades por lo que el estado $|\Psi\rangle$ debe ser unitario, y debe satisfacer:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{4}$$

A su vez, hay dos tipos de operaciones relevantes en los espacios de Hilbert: el producto interno y el producto externo, que la notación de Dirac permite plantear. El producto interno es el mismo que entre vectores en el álgebra clásica, y se nota como $\langle\Psi|\Psi\rangle$, que resultaría en el módulo cuadrado del estado Ψ. Por otro lado el producto externo entre dos vectores está dado por la expresión $|\Psi\rangle\,\langle\Psi|$. Este producto entre vectores resulta en una matriz, de la forma:

$$|\Psi\rangle\,\langle\Psi| = \begin{bmatrix} \alpha \\ \\ \beta \end{bmatrix} \begin{bmatrix} \alpha^* & \beta^* \end{bmatrix} = \begin{bmatrix} |\alpha|^2 & \alpha\beta^* \\ \\ \beta\alpha^* & |\beta|^2 \end{bmatrix} \tag{5}$$

Es decir, cada entrada (i,j) de la matriz resultante se obtiene multiplicando la componente i del vector columna por la componente conjugada j del vector fila.

1.3 Postulado 2: Como evolucionan los qubits

El segundo postulado:

"La evolución de un sistema cuántico ***cerrado*** *está descripto por una* ***transformación unitaria****. Esto es, el estado* $|\Psi\rangle$ *del sistema en un tiempo* t_1 *se relaciona con el estado* $|\Psi'\rangle$ *del sistema en un tiempo* t_2 *por medio de un operador unitario* $\boldsymbol{U}$ *que solo depende de los tiempos* t_1 *y* t_2*"* [8]

$$|\Psi'\rangle = \mathbf{U}\,|\Psi\rangle \tag{6}$$

Si bien no es sencillo encontrar el sistema físico asociado a la tranformación que se quiere emplear, los postulados de la mecánica cuántica garantizan que la matriz existe y es unitaria. Debe ser unitaria para preservar la norma de los estados, cuyo módulo no puede ser mayor que uno por tratarse de suma de probabilidades. Para sistemas de un qubit, existen algunas transformaciones notables, que luego pueden ser generalizadas para sistemas de varios qubits. Algunos operadores unitarios de interés son los de Pauli, dados por las matrices de las ecuación (7 a 10).

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7}$$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{8}$$

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{9}$$

$$Z = \begin{bmatrix} 1 & 0 \\ \\ 0 & -1 \end{bmatrix} \tag{10}$$

Veamos como actúan estos operadores sobre el estado Ψ definido anteriormente:

$$I\,|\Psi\rangle = \begin{bmatrix} 1 & 0 \\ \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ \\ \beta \end{bmatrix} \tag{11}$$

Donde I es el operador identidad que no modifica las amplitudes del estado de entrada.

$$X\,|\Psi\rangle = \begin{bmatrix} 0 & 1 \\ \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \\ \alpha \end{bmatrix} \tag{12}$$

Al operador X se le conoce como el análogo cuántico a la compuerta **NOT** clásica, dado a que cambia las amplitudes entre los estados de la base.

$$Z\,|\Psi\rangle = \begin{bmatrix} 1 & 0 \\ \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \\ \beta \end{bmatrix} = \begin{bmatrix} \alpha \\ \\ -\beta \end{bmatrix} \tag{13}$$

Por otro lado, al operador Z se le conoce como *Phase Flip*, puesto que altera la fase relativa a un eje.

$$Y\,|\Psi\rangle = \begin{bmatrix} 0 & -i \\ \\ i & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \\ \beta \end{bmatrix} = \begin{bmatrix} -i\beta \\ \\ i\alpha \end{bmatrix} \tag{14}$$

Por último, al operador Y se le conoce como *Bit-Phase Flip*, dado a que tiene la misma acción que aplicar los operadores X y Z en cascada.

Estos cuatro operadores notables llevan su nombre en honor del físico austríaco Wolfgang Pauli. Para este trabajo serán necesarios algunos otros operadores, como ser el de Walsh-

Hadamard dado por la matriz que se muestra a continuación junto con su acción sobre el estado $|\Psi\rangle$.

$$H|\Psi\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} \alpha + \beta \\ \alpha - \beta \end{bmatrix} \tag{15}$$

Por último, el operador C_{NOT}, o controlled-Not, operador de un sistema de al menos dos qubits, que niega el valor del segundo qubit si y solo si el primero es $|1\rangle$.

$$C_{NOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{16}$$

1.4 Postulado 3: Como la medición afecta a los qubits

El tercer postulado:

*"Las mediciones cuánticas están descriptas por un conjunto M_m de **operadores de medida**. Estos son operadores que actúan en el espacio de estados del sistema que se quiere medir. El índice **m** refiere a los posibles resultados que puedan surgir de realizar la medición."* [9]

Si el estado del sistema cuántico es $|\Psi\rangle$ inmediatamente antes de la medida, entonces al probabilidad de obtener el resultado m está dada por el producto interno:

$$p(m) = \langle\Psi|M_m^\dagger M_m|\Psi\rangle \tag{17}$$

Y el estado del sistema luego de la medida es:

$$\frac{M_m \left|\Psi\right\rangle}{\sqrt{\langle\Psi|M_m^\dagger M_m|\Psi\rangle}} \tag{18}$$

Como mencionado anteriormente, la probabilidad de obtener el resultado m está dado por la ecuación (17), y como tales deben satisfacer:

$$1 = \sum_m p(m) = \sum_m \langle\Psi|M_m^\dagger M_m|\Psi\rangle \tag{19}$$

De donde se desprende que los operadores M_m deben satisfacer la siguente relación para que la ecuación (19) se cumpla.

$$\sum_m M_m^\dagger M_m = I \tag{20}$$

Para mayor claridad analizaremos el caso de una medida en un sistema de un qubit más concretamente: analizaremos el resultado de medir el estado Ψ definido en la ecuación (3). En este caso, se tienen solo dos elementos en la base y por tanto, los operadores estan dados por $M_0 = \left|0\right\rangle \left\langle 0\right|$ y $M_1 = \left|1\right\rangle \left\langle 1\right|$, o sea:

$$M_0 = \begin{bmatrix} 1 & 0 \\ \\ 0 & 0 \end{bmatrix} M_1 = \begin{bmatrix} 0 & 0 \\ \\ 0 & 1 \end{bmatrix} \tag{21}$$

Notando que estos operadores cumplen la ecuación (20) y que el resultado de elevar cada matriz al cuadrado es sí misma, calculamos la probabilidad de medir 0 (notado $p(0)$):

$$p(0) = \langle\Psi|M_0^\dagger M_0|\Psi\rangle = \langle\Psi|M_0|\Psi\rangle \tag{22}$$

$$p(0) = |\alpha|^2 \tag{23}$$

Y el estado resultante luego de la medida es:

$$\frac{M_0 \left|\Psi\right\rangle}{|\alpha|} = \frac{\alpha \left|0\right\rangle}{|\alpha|} \tag{24}$$

Siguiendo el mismo razonamiento podemos calcular la probabilidad de obtener 1 y el estado resultante luego de la medida:

$$p(1) = |\beta|^2 \tag{25}$$

$$\frac{M_1 |\Psi\rangle}{|\beta|} = \frac{\beta |0\rangle}{|\beta|} \tag{26}$$

Mientras no se mida el estado del sistema, este es efectivamente una superposición de estados base. Luego de realizar una medición, esa superposición se rompe, y el estado colapsa a un estado de la base del operador de medida.

1.5 Postulado 4: Como se combinan para crear sistemas de varios qubits

El cuarto postulado, donde $\otimes$ denota el producto tensorial:

"El espacio de estados de un sistema físico compuesto es el producto tensorial de los espacios de estados de las componentes del sistema. Además, si las componentes están numeradas desde 1 hasta n, y el sistema número i está en el estado $|\Psi_i\rangle$*, entonces el estado completo del sistema total es* $|\Psi_1\rangle \otimes |\Psi_2\rangle ... \otimes |\Psi_n\rangle$*"* [10]

Tomemos, por ejemplo, un estado de dos qubits $|\Psi\rangle = |\psi\rangle \otimes |\phi\rangle$ y $|\phi\rangle = |\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Para encontrar el espacio de estados del sistema de dos qubits descripto por ellos, basta con calcular el producto tensorial $|\psi\rangle \otimes |\phi\rangle$, el cual resultaría:

$$|\psi\rangle \otimes |\phi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{27}$$

$$|\psi\rangle \otimes |\phi\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{28}$$

1.6 Entrelazamiento cuántico

El entrelazamiento es una de las propiedades más interesantes de los sistemas cuánticos.

Conceptualmente, el entrelazamiento cuántico es el fenómeno físico que se observa cuando un grupo de partículas interactúan de manera tal que el estado cuántico de una partícula no puede ser descripto independientemente del estado de las otras partículas, ni siquiera estando separadas una gran distancia. Es un fénomeno interesante, principalmente porque no se da en la física clásica. Para su mejor entendimiento, se da un ejemplo, siendo el más sencillo el de la teleportación cuántica. La teleportación cuántica es una técnica para enviar estados cuánticos a largas distancias, sin enviar la información directamente por un canal de comunicación cuántico, que relaciona el estado recibido al del remitente.

Emmett y Marty trabajaron juntos y en ese tiempo crearon un par de qubits entrelazados, pero ahora viven en distintos puntos del planeta y cada uno tiene un qubit del par entrelazado consigo. Años más tarde, Emmett debe hacerle llegar a Marty un estado de un qubit $|\Psi\rangle$, pero él no conoce el estado del qubit que debe enviar y únicamente puede enviar información por un canal de comunicación clásica. Emmett no puede determinar el estado en el que se encuentra el qubit, puesto que solo posee una copia del mismo y aun si pudiera, describirlo tomaría una cantidad infinita de información clásica. Cuánticamente existe una solución: Emmett hace interactuar el qubit que desea enviar con su qubit del par entrelazado que comparte con Marty, y luego mide ambos qubits. Después de la medición, Emmett tiene uno de cuatro resultados posibles: 00, 01, 10 o 11. Le envía el resultado de la medida a Marty por vía clásica y dependiendo de la información recibida, Marty realiza sobre su qubit del par entrelazado una de cuatro operaciones, de manera de obtener así el estado $|\Psi\rangle$. En la figura 1 se muestra un esquema de las operaciones que realizan ambos.

El estado $|\beta_{00}\rangle$ representa el par entrelazado, de donde Emmett tiene el primer qubit y Marty el segundo. Calculando entonces los estados intermedios $|\Psi_i\rangle$, asumiendo que $|\Psi\rangle{=}\alpha\,|0\rangle{+}\beta\,|1\rangle$:

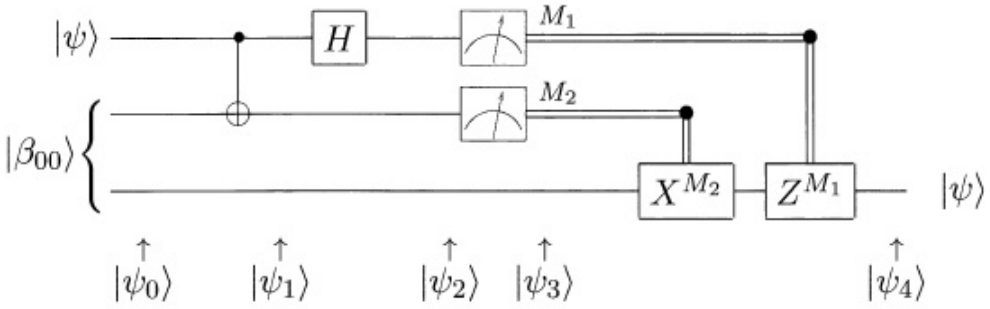

Figura 1: Esquema para la teleportación cuántica [12]

$$|\Psi_0\rangle = |\Psi\rangle \otimes |\beta_{00}\rangle \tag{29}$$

$$|\Psi_0\rangle = \frac{1}{\sqrt{2}}[\alpha\,|0\rangle\,(|00\rangle + |11\rangle) + \beta\,|1\rangle\,(|00\rangle + |11\rangle)] \tag{30}$$

Para calcular $|\Psi_1\rangle$, a la salida del C_{NOT}:

$$|\Psi_1\rangle = \frac{1}{\sqrt{2}}[\alpha\,|0\rangle\,(|00\rangle + |11\rangle) + \beta\,|1\rangle\,(|10\rangle + |01\rangle)] \tag{31}$$

Análogamente, la compuerta Hadamard afecta al qubit de Emmett del par enlazado:

$$|\Psi_2\rangle = \frac{1}{\sqrt{2}}[\alpha(|0\rangle + |1\rangle)(|00\rangle + |11\rangle) + \beta(|0\rangle - |1\rangle)(|10\rangle + |01\rangle)] \tag{32}$$

Reescribiendo la expresión $|\Psi_2\rangle$

$$|\Psi_2\rangle = \frac{1}{\sqrt{2}}[|00\rangle\,(\alpha\,|0\rangle + \beta\,|1\rangle) + |01\rangle\,(\alpha\,|1\rangle + \beta\,|0\rangle) \tag{33}$$

$$+ |10\rangle\,(\alpha\,|0\rangle - \beta\,|1\rangle) + |11\rangle\,(\alpha\,|1\rangle - \beta\,|0\rangle)]$$

Ahora Emmett realiza una medida sobre su qubit del par entrelazado.

$$|\Psi_3(00)\rangle \Rightarrow [\alpha\,|0\rangle + \beta\,|1\rangle] \tag{34}$$

$$|\Psi_3(01)\rangle \Rightarrow [\alpha\,|1\rangle + \beta\,|0\rangle] \tag{35}$$

$$|\Psi_3(10)\rangle \Rightarrow [\alpha\,|0\rangle - \beta\,|1\rangle] \tag{36}$$

$$|\Psi_3(11)\rangle \Rightarrow [\alpha\,|1\rangle - \beta\,|0\rangle] \tag{37}$$

Donde la notación $|\Psi_3(ii)\rangle$ se refiere al resultado obtenido al medir $|\Psi_3\rangle$.

De esta manera, Emmett comunica el resultado de su medida por un canal clásico y de acuerdo a lo que obtenga, Marty opera sobre su qubit del par entrelazado. Si el comunicado fue $|00\rangle$, Marty ya tiene el estado $|\Psi\rangle$. Si en cambio el comunicado es $|01\rangle$, Marty debe aplicar la compuerta X dada por la ecuación (12), si fuese $|01\rangle$, puede arreglarlo realizando una compuerta Z, ecuación (13). Por último, si Emmett mide $|11\rangle$, Marty debería aplicar primero una compuerta X y luego una Z: en todos los casos, Emmett puede obtener el estado que Marty quería transmitirle.

1.7 Representación de estados

Previamente en la sección 1.2 la notación de estados vectoriales fue introducida, y posteriormente en la sección 1.3 como los operadores afectan a estos estados. En muchos casos, no se conoce la descripción completa del sistema, solo se conoce un subsistema del mismo, como por ejemplo, si el objeto de estudio es un sistema abierto, es decir, interactúa con el ambiente. A los estados asociados a estos sistemas se les llama estados mixtos, y cuentan con su propia representación dada por el operador matriz de densidad. La matriz de densidad es un operador en un espacio de Hilbert dado por:

$$\rho = \sum_i p_j\,|\Psi_j\rangle\,\langle\Psi_j| \tag{38}$$

$$0 \leq p_j \leq 1 \tag{39}$$

$$\sum_j p_j = 1 \tag{40}$$

p_j es un elemento de una matriz que pondera peso del estado $|\Psi_j\rangle$ en el estado del sistema completo. De esta manera, se representa una mezcla estadística de estados puros.

El valor esperado de un observable A sobre este sistema, donde por observable se entiende cualquier propiedad física del sistema que pueda ser medida (energía, momento angular, etc.), está dado por las ecuaciones (41) y (42) donde Tr es el operador traza de una matriz.

$$< A >= \sum_j p_j \langle\Psi_j|A|\Psi_j\rangle = \sum_j p_j Tr(|\Psi_j\rangle \langle\Psi_j| A) = \sum_j Tr(p_j |\Psi_j\rangle \langle\Psi_j| A) \tag{41}$$

$$< A >= Tr(\sum_j p_j |\Psi_j\rangle \langle\Psi_j| A) = Tr(\rho A) \tag{42}$$

Las matrices de densidad cumplen:

- $Tr(\rho)$=1
- ρ debe ser un operador hermítico, es decir ρ=$\rho^\dagger$
- ρ debe ser definido positivo: $\langle\Psi\rho\Psi\rangle > 0 \ \forall \ \Psi$

En los casos en que el estado pueda ser representado por un vector se le denomina estado puro, y su matriz de densidad se obtiene fácilmente como $\rho_\Psi = |\Psi\rangle \langle\Psi|$

1.8 Operador traza parcial

La traza parcial es un mapeo de las matrices de densidad ρ_{AB} en el espacio compuesto $\mathscr{H}_A \otimes \mathscr{H}_B$ a matrices de densidad ρ_A en el espacio $\mathscr{H}_A$ y ρ_B en el espacio $\mathscr{H}_B$. Está definido como una extensión lineal del mapeo:

$$Tr_B : S \otimes T \rightarrow Tr(T)S \tag{43}$$

Para cualquier matriz S en $\mathscr{H}_A$ y T en $\mathscr{H}_B$. Sea $[|a_i\rangle]$ una base de $\mathscr{H}_A$ y $[|b_i\rangle]$ una ba-

se de $\mathscr{H}_B$. Cualquier matriz ρ_{AB} en $\mathscr{H}_A \otimes \mathscr{H}_B$ puede ser descompuesta como $\rho_{AB} = \sum_{ijkl} c_{ijkl} |a_i\rangle \langle a_j| \otimes |b_k\rangle \langle b_l|$ y su traza parcial con respecto a B:

$$Tr_B(\rho_{AB}) = \sum_{ijkl} c_{ijkl} |a_i\rangle \langle a_j| \otimes \langle b_k| |b_l\rangle \tag{44}$$

La cual es una matriz ρ_A en $\mathscr{H}_A$.
Un estado genérico de dos qubits ρ_{AB} puede ser escrito como combinación de los estados de la base ortonormal $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, como:

$$\rho_{AB} = \rho_{11} |00\rangle \langle 00| + \rho_{12} |00\rangle \langle 01| + \rho_{13} |00\rangle \langle 10| + ... + \rho_{44} |11\rangle \langle 11| \tag{45}$$

A la traza parcial con respecto a B, se le llama matriz reducida ρ_A del subespacio $\mathscr{H}_A$ y puede ser hallada con la ecuación (44):

$$\rho_A = (\rho_{11} + \rho_{22}) |0\rangle \langle 0| + (\rho_{13} + \rho_{24}) |0\rangle \langle 1| + (\rho_{31} + \rho_{42}) |1\rangle \langle 0| + (\rho_{33} + \rho_{44}) |1\rangle \langle 1| \tag{46}$$

1.9 Compuertas cuánticas universales

En computación clásica, cualquier operación binaria puede llevarse a cabo con una combinación finita de las compuertas básicas AND, OR y NOT. De hecho, cualquier operación puede llevarse a cabo únicamente con una cascada de compuertas $NAND$, ya que es posible implementar las tres anteriores únicamente con $NAND$s. Se dice entonces que el conjunto AND, OR y NOT son compuertas universales de la computación clásica, dado a que forman un generador de cualquier operación que se desee.

Al igual que en computación clásica, en computación cuántica también existen conjuntos de compuertas universales. Un resultado teórico importante para este fin, es el teorema de Solovay-Kitaev (13). Este teorema prueba que, si se tiene un conjunto $\mathscr{S}$ de compuertas

universales, un circuito cuántico formado por un número finito m de compuertas puede ser aproximado con un error arbitrario ε, por un circuito cuántico de orden $O(m \log(\frac{m}{\varepsilon}))$, formado únicamente por combinación de elementos de $\mathscr{S}$. Es un resultado muy importante: prueba que las computadoras cuánticas solo precisan implementar un número finito de compuertas básicas para poder implementar cualquier transformación unitaria que se pueda desear.

En particular, el conjunto formado por compuertas unitarias arbitrarias de un qubit y la compuerta C_{NOT}, presentada en la ecuación (16), son un conjunto universal de compuertas cuánticas. La prueba excede por su complejidad al contenido de este trabajo, pero puede encontrarse en (12).

1.10 Fidelidad y correlación

1.10.1 Fidelidad

Una parte central del trabajo en computación cuántica consta de comparar un resultado experimental, imperfecto, contra el resultado del modelo teórico. Para ello, es necesario algún tipo de distancia entre estados que pueda ser medida para poder cuantificar que tan parecidos son estos estados. Una medida conocida de distancia, la utilizada en este trabajo, es la que surge de la fidelidad cuántica.

Sean ρ y γ dos estados cuánticos que se quieren comparar, la fidelidad entre ellos esta dada por:

$$Fid(\rho, \gamma) \coloneqq Tr\sqrt{\rho^{\frac{1}{2}}\gamma\rho^{\frac{1}{2}}} \tag{47}$$

La fidelidad es una herramienta potente y útil para medir distancias entre estados cuánticos, definida como:

$$D_{Fid}(\rho, \gamma) \coloneqq \arccos\left(Fid(\rho, \gamma)\right) \tag{48}$$

Algunas propiedades relevantes de la fidelidad son:

- Está acotada: $0 \geq Fid(\rho, \gamma) \geq 1$
- Es simétrica: $Fid(\rho, \gamma) = Fid(\gamma, \rho)$,
- Si ρ (o γ) es un estado puro, $\rho = |\psi\rangle \langle\psi|$; $Fid(\rho, \gamma) = \sqrt{\langle\psi| \gamma |\psi\rangle}$

1.10.2 Correlación

Además de la información disponible en la distancia entre dos estados cuánticos, existe información útil en otro recurso: la correlación. La correlación da una medida de qué tan relacionadas están dos variables aleatorias. El análogo cuántico de la correlación clásica es particularmente útil porque entre estados puros (que son los estados de estudio tratados en este trabajo), habla de la existencia de entrelazamiento.

La correlación clásica, medida como la Información Mutua entre dos variables aleatorias X,Y viene dada por:

$$I(X : Y) = \sum_{x,y} P_{X,Y}(x, y) log \frac{P_{X,Y}(x, y)}{P_X(x) P_Y(y)} \tag{49}$$

donde $P_X(x)$,$P_Y(y)$ denotan las distribuciones marginales asociadas a las variables X e Y respectivamente. El equivalente cuántico de Correlación Mutua *Quantum Mutual Information* (MI) se define como

$$I(\rho_{AB}) = S(\rho_{AB} \| \rho_A \otimes \rho_B) = S(\rho_A) + S(\rho_B) - S(\rho_{AB}) \tag{50}$$

donde ρ_A y ρ_B son las matrices reducidas y $S(\rho)$ es la entropía cuántica de Von Neumann (11). Además de esta correlación, existen otras diversas medidas de correlación como ser las bipartitas y multiqubits. En este trabajo se utilizará también la Medida de Correlación acumulativa, *Cumulative Correlation Measure*(CCM)(14) que contabiliza y acumula información de todas las correlaciones de internas de los subespacios, definida en forma iterativa como:

$$C(\rho) = min_k [2^{N-2} S(\rho \| \rho_{Ak} \otimes \rho_{Bk}) + C(\rho_{Ak}) + C(\rho_{Bk})] \tag{51}$$

Donde el índice k indica un elemento (ρ_{Ak},ρ_{Bk}) del conjunto de todas las posibles biparticiones del estado ρ, partiendo el estado en producto de subespacios hasta llegar a qubits individuales. Si bien en el caso general, la CCM computa la correlación total, es decir, la suma de correlación clásica y cuántica, como los casos de estudio se reducen a estados puros el resultado es la correlación cuántica.

Capítulo 2: Redes neuronales y aproximadores universales

2.1 Introducción

Esencialmente, una red neuronal es una herramienta matemática que busca simular el comportamiento del cerebro acorde a un modelo particular. Están formadas por una colección de nodos conectados, donde los nodos cumplen el rol de las neuronas y las conexiones el de las sinapsis. A su vez, en cada conexión los nodos pueden transmitir y recibir información.

Estos modelos surgen originalmente en 1943 cuando Warren McCulloch y Walter Pitts (15) publican el primer modelo de neurona artificial, el cual inspira en la década de 1950 a Frank Rosenblatt a desarrollar el modelo de neurona conocida como perceptrón (16). Si bien hoy en día se utilizan otros modelos de neurona, el perceptrón sigue vigente y es un modelo relativamente sencillo.

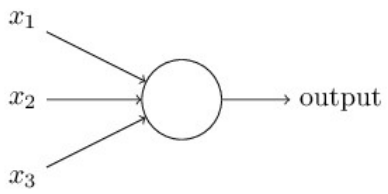

Figura 2: Perceptrón básico

Esencialmente, el perceptrón es un nodo que toma un conjunto de entradas binarias $(x_1, ..., x_n)$ y retorna una única salida de valor también binario, como se muestra en la figura 2. Para computar el valor de la salida Rosenblatt introduce el concepto de pesos sinápticos ω_i para cada conexión $i \in (1, .., n)$, los que determinan la importancia entre los

nodos. De acuerdo a estos pesos y las entradas, se calcula la suma ponderada:

$$\gamma = \sum_j \omega_{ij} x_j \tag{52}$$

Con el valor de γ computado y un valor de umbral thr fijado por el la salida puede tomar dos valores acorde a una función no lineal:

- 1, si $\gamma > thr$
- 0, si $\gamma < thr$

El umbral es simplemente un parámetro de la neurona que simula el umbral de voltaje por encima del cual una neurona real dispara, polarizando a sus vecinas. Un modelo más completo de perceptrón se presenta en la figura 3, con los pesos sinápticos y la función no lineal de mapeo entrada-salida.

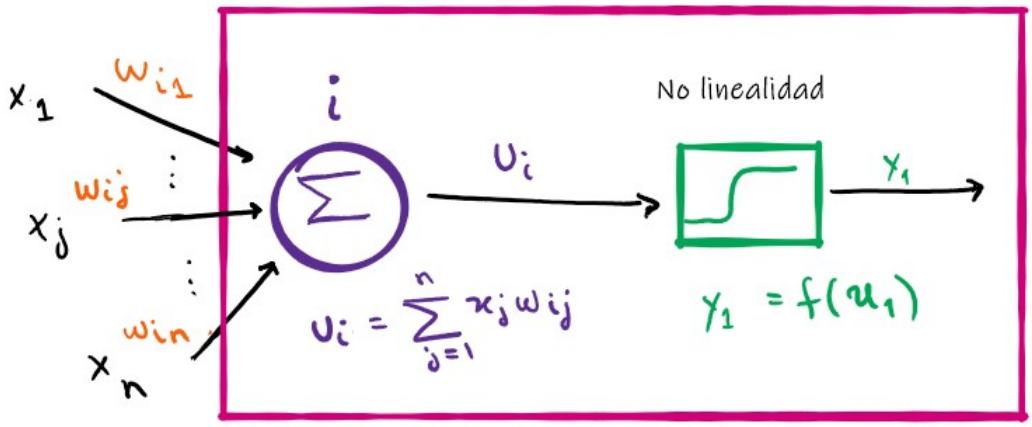

Figura 3: Perceptrón completo

Una manera de ver al perceptrón es un dispositivo que toma decisiones evaluando evidencia. El modelo es bastante sencillo, pero analizaremos un ejemplo para mayor claridad. Supongamos que una universidad ofrece un curso sobre mecánica cuántica que al lector le intriga, pero no tomó aún la decisión de tomar el curso. El modelo de perceptrón puede ser usado para modelar la toma de esta decisión. Tres factores a considerar podrían ser:

- ¿Tiene las herramientas matemáticas para tomar el curso?
- ¿Tiene la disponibilidad horaria necesaria para asistir a clase?
- ¿Tiene amigos que quieran tomar el curso con él?

En este ejemplo, los tres factores corresponden a las tres señales de entrada $x_1, x_2, x3$ de la figura 2. Por ejemplo, $x_1 = 1$ si el alumno tomo los cursos de matemática necesarios anteriormente, o $x_1 = 0$ si carece de los conocimiento necesarios. De igual manera $x_2 = 1$ si puede asistir a clase, y $x_2 = 0$ si no puede hacerlo, $x_3 = 1$ si tiene amigos que quieran tomar el curso con él, y $x_3 = 0$ si sus amigos no tienen interés en tomar el curso. Ahora, supongamos que la intriga del lector por tomar el curso es tal que no le preocupa si debe tomar el curso con desconocidos porque sus amigos no quieren tomarlo, pero tiene temor de no aprobar el curso por falta de la base previa necesaria, o por no poder asistir a clase. Un ejemplo de la asignación de los pesos sinápticos de cada factor podría ser $\omega_1, \omega_2 = 5$ y $\omega_3 = 3$: los pesos altos indican factores a los que el alumno les da mucha importancia, mientras que los bajos son los menos relevantes. Con estos pesos resta únicamente fijar el umbral, y supongamos que se fija en $thr = 9$. De esta manera, la salida del perceptrón devuelve 1 cuando el lector puede asistir a clase y cuenta con la formación previa necesaria sin importar si sus amigos toman el curso, y 0 en cualquier otro escenario.

Variando los pesos y el umbral podemos obtener distintos modelos de la toma de la decisión, por ejemplo si el umbral fuese $thr = 8$, el perceptrón decidiría si el lector debe tomar el curso cuenta con la base necesaria **o** si cuenta con la disponibilidad horaria, **y** si sus amigos tambien lo toman.

2.2 Redes neuronales como grafos dirigidos

A partir de una interconexión de neuronas como la presentada en la sección anterior es que se forman las redes neuronales. Una manera de estudiar estas redes que será útil para el análisis de este trabajo es verlas como grafos dirigidos.

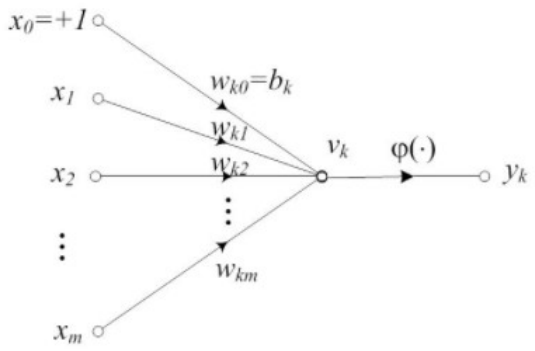

Figura 4: Neurona vista como nodo de grafo

Un grafo dirigido es una red de ramas orientadas que están interconectadas en algunos puntos llamados nodos, como se puede ver en la figura (4). Cada nodo j tiene asociada una señal x_j y cada enlace dirigido desde el nodo j hasta el nodo k tiene además asociada una función de transferencia que especifica como depende la señal y_k en el nodo k de la señal x_j en el nodo j. El flujo de las señales en la red está dictado por 3 reglas:

- **Regla 1:** Las señales fluyen únicamente en la dirección definida por la flecha en el grafo. Se distingue entre dos tipos de enlaces: los sinápticos, gobernados por una relación entrada-salida lineal, y los de activación, gobernados por una relación no lineal.

- **Regla 2:** La señal asociada a un nodo es la suma algebraica de todos las señales que entran al nodo pertinente por medio de los enlaces

- **Regla 3:** La señal en un nodo es transmitida a cada enlace de salida asociado al nodo pertinente, y su transmisión es completamente independiente de las funciones de transferencia de los enlaces por los que se transmiten.

A partir de estas tres reglas, Haykin (17) ofrece una definición matemática de red neuronal:

Üna red neuronal es un grafo orientado que consiste de nodos interconectados mediante enlaces sinápticos y de activación, caracterizado por cuatro propiedades:

- *Cada neurona se representa por un conjunto de enlaces sinápticos, un umbral aplicado externamente y un vínculo de activación. El umbral se representa por un enlace sináptico como una señal de entrada fija en -1*
- *Los enlaces sinápticos de una neurona ponderan sus respectivas señales de entrada*
- *La suma ponderada de las señales de entrada define el nivel de actividad interna total de la neurona en cuestión*
- *El enlace de activación condensa la actividad interna de la neurona para producir una señal de salida que represena la variable de estado de la neurona"*

2.3 Arquitectura de redes

La estructura que forman las neuronas en una red neuronal está íntimamente asociado al algoritmo (reglas) utilizadas para el aprendizaje de la misma. Existen varias estructuras de redes, pero en este trabajo consideraremos solo las dos que serán útiles más adelante. Estás son:

- Redes *feedforward* de una capa
- Redes *feedforward* mutilcapa

Una red neuronal usualmente está formada por un conjunto de neuronas organizadas en capas. Una red feedfoward de una capa es aquella red con una única capa de neuronas de salida, como se muestra en la figura 5.

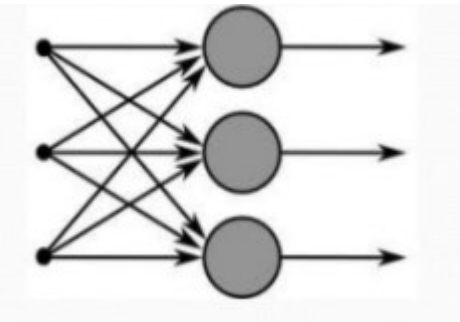

Figura 5: Red feedfoward de una capa

La capa de neuronas de entrada no se cuenta puesto que no realiza ningún tipo de computo. Un ejemplo de una red de una única capa es una memoria asociativa lineal, donde la red asocia un patrón de salida vectorial a un patrón de entrada vectorial, y la información es almacenada en la red mediante modificaciones de los pesos sinápticos. La

segunda clase de redes neuronales que analizaremos son las redes feedforward multicapa, como la que se muestra en la figura (6).

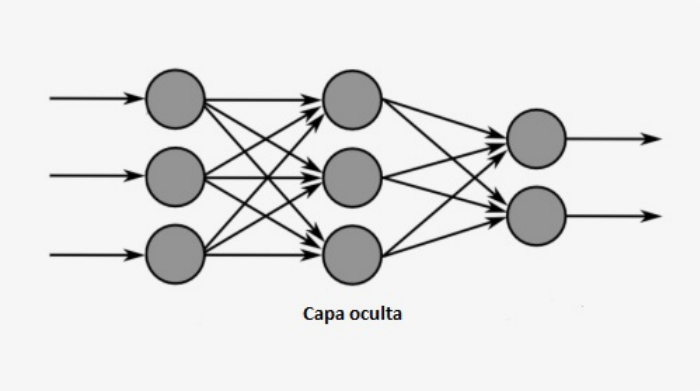

Figura 6: Red feedforward multicapa

Esta clase de redes se distingue por tener al menos dos capas de neuronas que realizan cómputos, además de la capa de entrada de neuronas que solo recepcionan las señales de entradas. A las capas intermedias que puedan existir entre la de entrada y la de salida, se les conoce como capas ocultas. La función de estas capas ocultas es la de intervenir entre las capas de entrada y salida, con el propósito de que la red obtenga una visión más global

del problema a resolver, de manera de poder reflejar sutilezas en el comportamiento que con una única capa no sería capaz de ver: agregar capas ocultas agudiza la sensibilidad de la red. Los nodos en la capa de entrada trasmiten los valores correspondientes al patrón de activación (vector de entrada) a la entrada de los nodos de la segunda capa. Luego esto se repite entre la segunda y la tercera y así sucesivamente, tantas capas como ocultas tenga la red. Típicamente, cada capa tiene tantas entradas como salida tenga la capa anterior, y el conjunto de señales que se ven a la salida de la última capa son la respuesta de la red completa al vector de entrada.

Las redes a su vez pueden clasificadas de acuerdo a su conectividad interna. Una red se dice totalmente conexa si todos los nodos de la capa i están conectados a todos los nodos de la capa $i + 1$, y parcialmente conexa alguno de estos enlaces no está presente, como se puede ver en la figura (7).

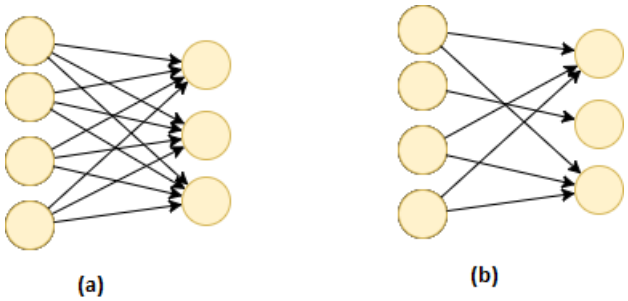

Figura 7: Red totalmente conexa (a), red parcialmente conexa (b)

2.4 Aprendizaje automático

Si bien las redes neuronales tienen varias propiedades interesantes, la propiedad de mas significativa es la capacidad de estas redes de aprender del ambiente y mejorar su desempeño a través del aprendizaje. ¿Cómo se define aprendizaje en este contexto? Haykin lo define como (18):

"El aprendizaje es el proceso por el cual los parámetros libres de una red neuronal son adaptados mediante un proceso continuo de estímulo por medio del entorno en el cual la red se encuentra sumergida. El tipo de aprendizaje está dado por la forma en que los parámetros cambian en el tiempo"

Esta definición implica la secuencia siguiente:

- La red es estimulada por su entorno
- La red cambia como reacción al estímulo
- La red responde de una nueva forma a un estímulo externo

Considerando un par de señales de nodo x_j y v_k, conectadas por un peso sináptico de ω_{jk}. La señal x_j representa la señal de salida de la neurona j y la señal v_k representa la actividad interna de la neurona k.

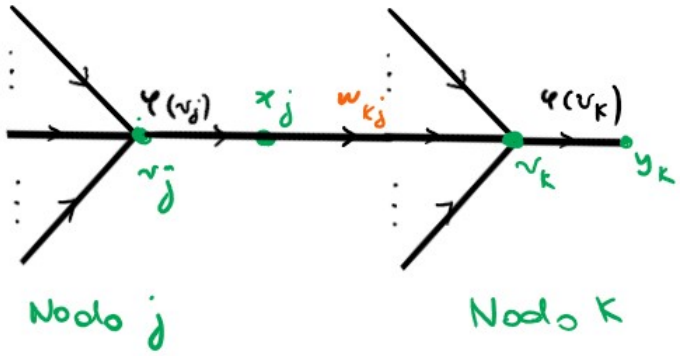

Figura 8: Cascada de neuronas

Sea $\omega_{jk}(n)$ el valor del peso sináptico ω_{jk} en el tiempo n. En ese mismo tiempo, se aplica un cambio $\Delta\omega_{jk}(n)$ al peso sináptico, el cual da como resultado $\omega_{jk}(n+1)$, según la relación 53.

$$\omega_{kj}(n+1) = \omega_{kj}(n) + \Delta\omega_{kj}(n) \tag{53}$$

A $\omega_{kj}(n)$ y $\omega_{kj}(n+1)$ se los puede ver como el valor viejo y nuevo del peso sináptico ω_{kj}, donde la ecuación 53 refleja los efectos implícitos del aprendizaje. En particular, el ajuste $\Delta\omega_{kj}(n+1)$ se computa de acuerdo al resultado del estímulo externo de acuerdo a un conjunto de reglas.

A un conjunto bien definido de reglas para resolver el problema del aprendizaje de la red se le denomina algoritmo de aprendizaje. Estos algoritmos de clasifican según el tipo de reglas que emplean y según el tipo de paradigma, modelo de aprendizaje que emplean, según la figura (9).

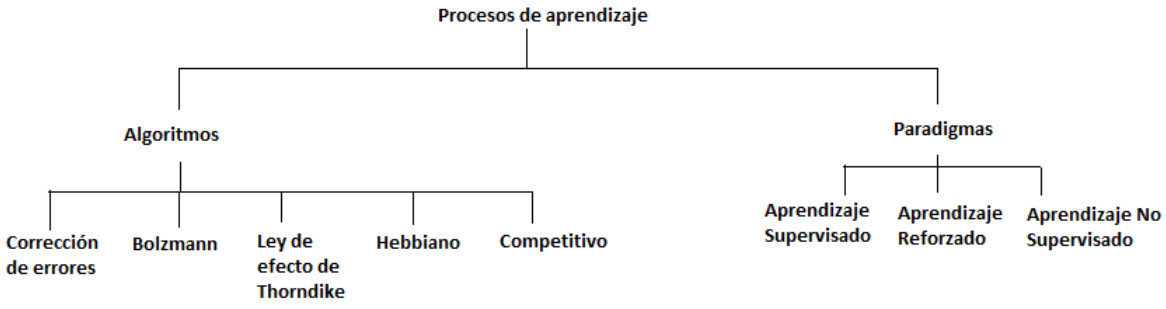

Figura 9: Clasificación de procesos de aprendizaje

En particular, se vera un algoritmo de aprendizaje supervisado: el algoritmo de corrección de errores, que fue el empleado en este trabajo.

2.5 Algoritmo de corrección de errores

Supongamos ahora entonces que se conoce un vector de entrada $x(n)$, sea $d_k(n)$ la respuesta deseada de la neurona k en el tiempo n, y sea $y_k(n)$ la respuesta real de la red. La respuesta $y_k(n)$ es producida por el estímulo $x(n)$ aplicado al inicio de la red en que la neurona k está embebida. El vector de entrada $x(n)$ y la respuesta deseada $d_k(n)$ para la neurona k constituyen un ejemplo particular del comportamiento de la red en un tiempo dado n.

Generalmente, la respuesta de la neurona k en tiempo n será diferente de la respuesta deseada $d_k(n)$ y por tanto, se puede definir una señal que se denomina error como la diferencia entre la respuesta deseada $d_k(n)$ y la respuesta real $y_k(n)$:

$$e_k(n) = d_k(n) - y_k(n) \tag{54}$$

El propósito del aprendizaje por corrección de errores es el de minimizar una función de costo basada en la señal de error $e_k(n)$, de manera que la respuesta real de la red se acerque a la respuesta ideal con el paso del tiempo. De hecho una vez fijada la función de costo, el problema de aprendizaje por corrección de errores se reduce a un problema de optimización donde se pueden aplicar todas las herramientas existentes. Un criterio comúnmente usado para la función de costo es el criterio del error cuadrático medio, definido como la media de la suma cuadrática de errores:

$$J = E[\frac{1}{2}\sum_k e_k^2(n)] \tag{55}$$

Donde E es la esperanza estadística y la suma acumula la actividad de cada neurona en la capa de salida de la red. La expresión (55) asume que los procesos subyacentes son estacionarios y la minimización de la función de costo J con respecto a los parámetros de la red lleva al llamado método del gradiente. Sin embargo, este proceso de optimización requiere conocimiento de las características estadísticas del proceso a estudiar, información que, generalmente, uno no posee. Para poder superar este obstáculo se define una solución aproximada al problema, como ser el de utilizar el valor *instantáneo* de la suma de los errores cuadráticos.

$$\mathscr{E}(n) = \frac{1}{2}\sum_k e_k^2(n) \tag{56}$$

La red se optimiza minimizando $\mathscr{E}(n)$ con respecto a los pesos sinápticos de la red. De acuerdo a la regla de aprendizaje de la corrección de errores (19), denominada regla delta usualmente, el ajuste $\Delta\omega_{kj}(n)$ que sufre el peso sináptico ω_{kj} en el tiempo n está dado por 58.

$$\Delta\omega_{kj}(n) = \eta e_k(n)x_j(n) \tag{57}$$

En esta expresión, η es una constante positiva que determina la velocidad de aprendizaje. Puesto de otra forma, el ajuste hecho a un peso sináptico es proporcional a la señal error y a la señal de entrada de la sinapsis en cuestión. Primero se computa la señal de error (54), con ella se computa la corrección según (58) y por último, la corrección es usada para actualizar el peso sináptico a $\omega_{kj}(n+1)$. En la figura 10 se muestra este comportamiento, donde el bloque z^{-1} denota un retraso unitario, donde $z^{-1}[\omega_{kj}(n+1)] = \omega_{kj}(n)$, $v_k = \sum_j x_j(n)\omega_{kj(n)}$ y $y_k(n) = \varphi(v_k(n))$.

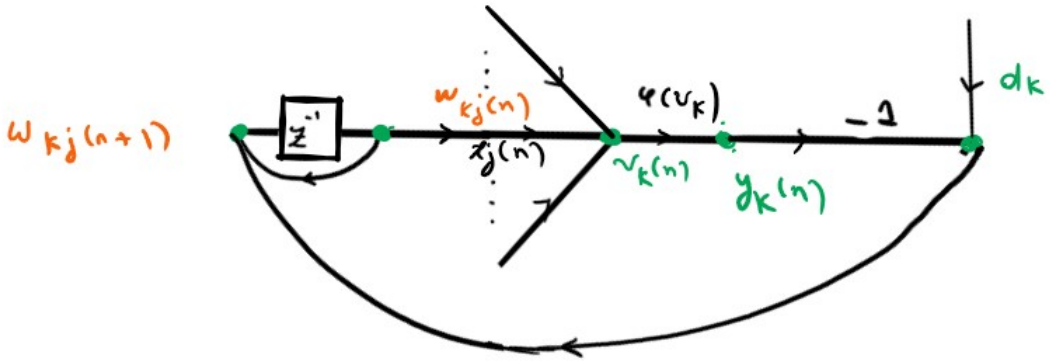

Figura 10: Grafo del aprendizaje por corrección de errores

Este grafo muestra claramente que el aprendizaje por corrección de errores se comporta como un sistema de lazo cerrado, por lo que es necesario tener cuidado a la hora de elegir el valor del parámetro de aprendizaje η de manera de asegurar la estabilidad del sistema realimentado. Este parámetro tiene un profundo impacto no solo en la velocidad de convergencia, sino en el hecho de que esta exista. Si η es muy pequeño, el proceso de aprendizaje procede sin problemas pero podría tomarle un tiempo muy largo converger a la solución. Si, por otro lado, η es muy grande, el proceso de aprendizaje se acelera pero se corre el peligro de que el proceso de aprendizaje diverga y el sistema se vuelva inestable.

Si se gráfica la función de costo J en función de los pesos sinápticos, la red neuronal consiste de una superficie multidimensional conocida como superficie de error. Dependiendo del

tipo de computo llevado a cabo en los nodos, se pueden distinguir dos tipos relevantes de superficies de error:

- Si la red consiste únicamente de nodos de procesamiento lineal, entonces la superficie de error es exactamente una función cuadrática de los pesos sinápticos y el gráfico es un paraboloide, un bowl, con un único mínimo global.

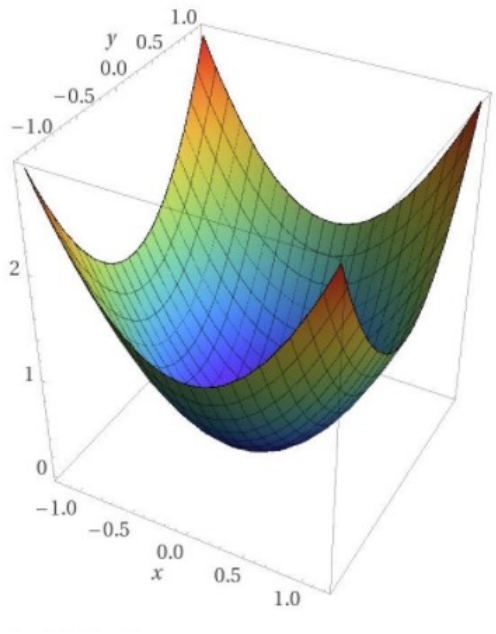

Figura 11: Superficie de error de una red lineal (20)

- Si la red posee nodos con procesamiento no lineal, el gráfico tiene al menos un mínimo global y probablemente, varios mínimos locales.

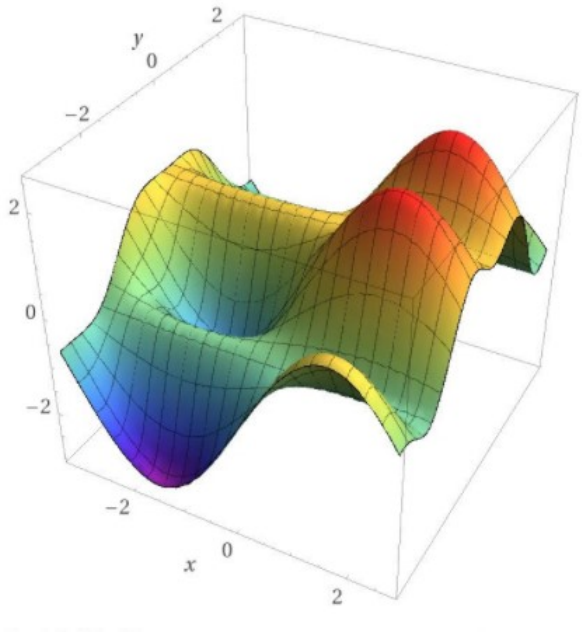

Figura 12: Superficie de error de una red no lineal (20)

En ambos casos, el objetivo del aprendizaje por corrección de errores es partir de un punto arbitrario cualquiera de la superficie de error, que está determinado por las condiciones iniciales en las que se fijan los pesos sinápticos de la red, y luego moverse en dirección al mínimo global, paso a paso. En el primer escenario el objetivo es relativamente sencillo de conseguir, mientras que en el segundo escenario no siempre es realizable, puesto que el algoritmo puede quedar atrapado en algún mínimo local de la función de costo y nunca llegar al mínimo global. Para intentar evitar que el algoritmo quede detenido en un mínimo local existen varias medidas preventivas, como ser el paso variable y el momento. Modificando la regla delta:

$$\Delta\omega_{ji}(n) = \alpha\Delta\omega_{ji}(n-1) + \eta e_k(n)x_j(n) \tag{58}$$

Es decir, ahora el cambio del peso sináptico en el paso n no depende únicamente de la señal de error y de la entrada, sino que también depende de los valores anteriores de él mismo, lo cual le da una suerte de memoria. η sigue siendo el parámetro de aprendizaje, al cual nos

referiremos como paso de aprendizaje de aquí en más, mientras que α es el momento del algoritmo. Agregar momento aumenta la velocidad de aprendizaje, evita algunos problemas de estancamiento en mínimos locales pero puede traer consigo problemas de convergencia por lo que hay que ser cuidadoso a la hora de ajustar α y η.

2.6 Aprendizaje supervisado

El paradigma de aprendizaje supervisado se basa en un actor esencial: el maestro. Conceptualmente, se puede pensar al maestro como una entidad que posee conocimiento del entorno, en forma de conjuntos entrada-salida de ejemplo. Sin embargo, este entorno es, de antemano, desconocido para la red neuronal de interés. Si el maestro y la red son expuestos uno de estos ejemplos de entrada, el maestro puede proveerle a la red con la salida deseada, puesto que posee el conocimiento. Entonces, lo s los parámetros de la red se ajustan bajo una influencia combinada del estimulo de entrada y la señal de error, donde la señal de error es la definida en (54). El ajuste de los parámetros se va dando por pasos, con el propósito de que la red eventualmente emule el comportamiento del maestro, y que esta emulación sea óptima en algún sentido estadístico. En resumen, el conocimiento del maestro se transfiere a la red lo más completamente posible, de forma que luego del proceso de aprendizaje, la red pueda manejar estimulos del entorno por sí misma.

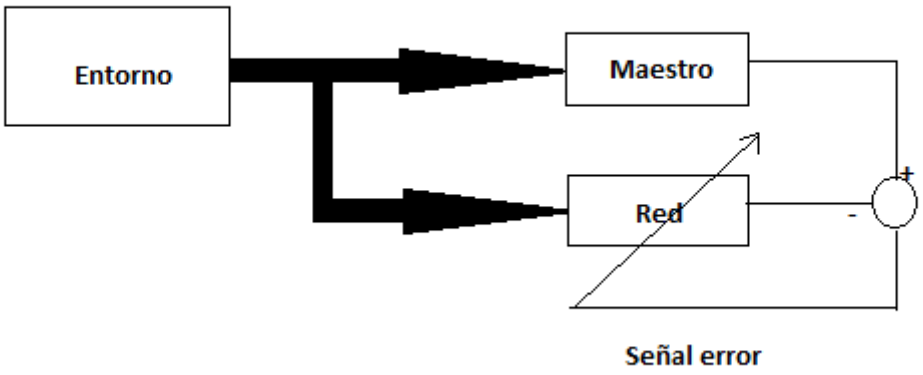

Figura 13: Modelo del aprendizaje supervisado

El proceso de aprendizaje supervisado ilustrado en (13)es en sí mismo el algoritmo de corrección de errores discutido en la sección anterior. Se trata de un sistema de lazo cerrado similar al presentado anteriormente en (10) pero en este caso, el entorno no forma parte del lazo. La medida de desempeño en este caso se puede pensar en términos de errores cuadráticos medios, definidos como función de los parámetros libres de la red. El gráfico de esta función es una superficie multidimensional de error como las presentadas anteriormente, y cualquier operación hecha por el sistema bajo la supervisión del maestro, representa un punto en esta superficie. Para que de hecho el desempeño de la red mejore con el tiempo y esta efectivamente aprenda del maestro, el punto de operación tiene que descender hacia algún mínimo, ya sea local o global. Los algoritmos de aprendizaje son capaces de lograr esta meta porque poseen información sobre el gradiente de la superficie de error, de acuerdo al punto de operación del sistema. El gradiente de la superficie de error es, en cualquier punto, un vector que apunta en la dirección en que la superficie desciende de forma más empinada, y los algoritmos de aprendizaje supervisado suelen utilizar un estimativo instantáneo de este vector. El uso de este estimativo resulta en que el punto de operación de la red se mueve siguiendo una caminata al azar, pero a pesar de esto, dado un algoritmo diseñado para minimizar una función de costo y un conjunto adecuado de patrones de entrada y salida, un sistema de aprendizaje supervisado generalmente es capaz de aproximar una función y reconocer patrones satisfactoriamente. La desventaja

grande con la que cuentan los métodos supervisados es que, sin intervención del maestro, la red no puede aprender de situaciones o patrones que no estén dentro del conjunto de entrenamiento original.

Algunos algoritmos notables de este paradigma son el famoso algoritmo de *back-propagation* (BP) y el de *least-mean-square* (LMS), que es implementado en este trabajo. El algoritmo LMS involucra un solo nodo de la red a la vez, mientras que BP involucra una conexión multicapa de nodos, de manera que LMS es un caso particular de este último.

2.7 Ecuaciones de Weiner - Hopf

Considerando un conjunto de p señales $x_1, x_2...x_p$ que son aplicadas a un junto de pesos sinápticos $\omega_1, \omega_2, ..., \omega_p$ y las señales ponderadas son luego sumadas para producir la señal de salida y. El desafío es hallar el junto de pesos sinápticos $\omega_1, \omega_2, ..., \omega_p$ óptimos de manera de minimizar la distancia cuadrática media entre la salida y del sistema y la respuesta óptima d. El sistema descripto puede ser visto como un filtro espacial, donde la relación entrada-salida está dada por:

$$y = \sum_{k=1}^{p} \omega_k x_k \tag{59}$$

Sea d la respuesta deseada del filtro, e la señal de error y J la función de costo definidas como:

$$e = d - y \tag{60}$$

$$J = \frac{1}{2} E[e^2] \tag{61}$$

La solución a este problema yace en las ecuaciones de Weiner-Hopf (21), y el filtro óptimo recibe el nombre de filtro de Weiner. De las tres ecuaciones anteriores y expandiendo, se obtiene:

$$J = \frac{1}{2} E[d^2] - E[\sum_{k=1}^{p} \omega_k x_k d] + \frac{1}{2} E[\sum_{j=1}^{p} \sum_{k=1}^{p} \omega_k \omega_j x_k x_j] \tag{62}$$

Como la esperanza es un operador lineal, (62) se puede reescribir como:

$$J = \frac{1}{2}E[d^2] - \sum_{k=1}^{p} \omega_k E[x_k d] + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} \omega_k \omega_j E[x_k x_j] \tag{63}$$

Donde se pueden reconocer 3 términos importantes:

- $E[d^2]$ es el valor cuadrático medio de la respuesta deseada, al cual se denominará r_d
- $E[dx_k]$ es la correlación cruzada entre la respuesta deseada y la señal de entrada, $r_{dx}(k)$
- $E[x_j x_k]$ es la autocorrelación de las señales de entrada, $r_x(j,k)$

Entonces se puede reescribir 63:

$$J = \frac{1}{2}r_d - \sum_{k=1}^{p} \omega_k r_{dx}(k) + \frac{1}{2}\sum_{j=1}^{p}\sum_{k=1}^{p} \omega_k \omega_j r_x(j,k) \tag{64}$$

Para determinar el punto de operación óptimo, se deriva la función de costo J con respecto a los pesos sinápticos, y luego se iguala el resultado a 0 $\forall k$. La derivada de J con respecto a w_k es exactamente el gradiente de la superficie de error con respecto al peso sináptico particular w_k:

$$\nabla_{w_k} J = \frac{\partial J}{\partial w_k}, k = 1, 2, .., p \tag{65}$$

Derivando 64:

$$\nabla_{w_k} J = -r_{dx}(k) + \sum_{j=1}^{p} \omega_j r_x(j,k) \tag{66}$$

La condición óptima satisface $\nabla_{w_k} J = 0$ por lo que, igualando (66) a 0:

$$\sum_{j=1}^{p} \omega_j r_x(j,k) = r_{dx}(k) \tag{67}$$

2.8 Método de mayor descenso y LMS

El algoritmo LMS se basa en el método de mayor descenso, que se basa en las ecuaciones de Weiner-Hopf. El método del mayor descenso tiene la tarea de continuamente buscar un mínimo de la superficie de error. Sea $\omega_k(n)$ el valor del peso ω_k en el filtro espacial calculado en la iteración n mediante este método. Razonablemente, el gradiente también se convierte en una función variante en el tiempo en sí misma:

$$\nabla_{\omega_k} J(n) = -r_{dx} + \sum_{j=1}^{p} \omega_j(n) r_x(j,k) \tag{68}$$

Los indices j, k refieren a distintos puntos en el espacio, mientras que el índice n refiere al número de iteración, al tiempo. De acuerdo a este método, la correción que se le hace al peso $w_k(n)$ en la iteración n se define como:

$$\Delta\omega_k(n) = -\eta \nabla_{\omega_k} J(n) \tag{69}$$

Y el nuevo peso $\omega_k(n+1)$ se obtiene como:

$$\omega_k(n+1) = \omega_k(n) + \Delta\omega_k(n) \tag{70}$$

Sustituyendo (68) en (69) y (70):

$$w_k(n+1) = w_k(n) + \eta[r_{dx}(k) - \sum_{j=1}^{M} \omega_j(n) r_x(j,k)] \tag{71}$$

El valor actualizado del k-ésimo peso del filtro de Weiner es igual, entonces, al valor del peso en la iteración anterior más una corrección proporcional al gradiente de la superficie de error con respecto a ese peso.

Ahora bien, si bien la expresión (71) es la formula explícita del método de máximo descenso, depende de dos parámetros no siempre conocidos: $r_x(j,k)$ y $r_{dx}(k)$. Para poder implementar la actualización a los pesos sinápticos, el algoritmo LMS utiliza estimaciones de los valores instantáneos de la función de autocorrelación $r_x(j,k)$ y la de correlación

cruzada $r_{dx}(k)$, donde las estimaciones que se usan son:

$$\hat{r}_x(j,k;n) = x_j(n)x_k(n) \tag{72}$$

$$\hat{r}_{dx}(k;n) = x_k(n)d(n) \tag{73}$$

Luego sustituyendo (72) y (73) en la expresión general del método (71):

$$\hat{\omega}_k(n+1) = \hat{\omega}_k(n) + \eta[x_k(n)d(n) - \sum_{j=1}^{p} \hat{\omega}_j(n)x_j x_k(n)] \tag{74}$$

$$\hat{\omega}_k(n+1) = \hat{\omega}_k(n) + \eta[d(n) - \sum_{j=1}^{p} \hat{\omega}_j(n)x_j]x_k(n) \tag{75}$$

Para el algoritmo LMS, $y(n) = \sum_{j=1}^{p} \hat{\omega}_j(n)x_j$, entonces:

$$\hat{\omega}_k(n+1) = \hat{\omega}_k(n) + \eta[d(n) - y(n)]x_k(n) \tag{76}$$

El algoritmo de máximo descenso se aplica a un entorno conocido, donde se conocen todos los pesos sinápticos de la red al comenzar el proceso $\omega(0)$y luego sigue una trayectoria dada en la superficie de error que culmina eventualmente en un valor óptimo ω_o. En contraste, el algoritmo LMS se aplica a un entorno desconocido donde se usa $\hat{\omega}(n)$, una estimación del vector de pesos sinápticos real, que se mueve en una trayectoria aleatoria (movimiento Browniano) en la superficie de error.

2.9 Aproximadores universales

En los estudios de redes neuronales (y sobretodo en *deep learning*, es vital el concepto de aproximador universal. Se dice que una red o, en general, una arquitectura de red, es un aproximador universal si es capaz de aproximar cualquier mapeo entrada salida al que sea sometido con tanta precisión como el usuario desee. En general, probar que una arquitectura cumple de hecho esta propiedad es engorroso, y al momento la manera de

atacar estos problemas es caso a caso: no existe un teorema general para cualquier tipo de red. En 1969 Minsky y Papert (22) prueban que una red *feedforward* de dos capas es capaz de aproximar un conjunto muy limitado de mapeos, pero luego en 1989 Hornik, Stinchcombe y White (23) prueban que una red *feedforward* multicapa cualquiera, como las que se utilizan en este trabajo, es un aproximador universal.

Capítulo 3: Modelado de la red

3.1 Introducción

En este capitulo se introduce el modelo completo implementado a partir de la teoría presentada en los capítulos anteriores, se discuten elecciones de diseño y se presentan resultados de casos de prueba.

3.2 Compuerta U_3

Para poder presentar el bloque básico de la red neuronal implementada, es necesario primero introducir una compuerta básica más. La compuerta unitaria U_3, que recibe su denominación de la plataforma IBM Q, se presenta en la ecuación 77.

$$U_3(\theta, \phi, \lambda) = \begin{bmatrix} \cos\frac{\theta}{2} & -e^{i\lambda}\sin\frac{\theta}{2} \\ e^{i\phi}\sin\frac{\theta}{2} & e^{i\lambda+i\phi}\cos\frac{\theta}{2} \end{bmatrix} \tag{77}$$

La compuerta U_3 es una compuerta unitaria de un qubit, que depende de tres parámetros. A modo de ejemplo, la compuerta X presentada en el capítulo 1 se obtiene de fijar $\theta = \pi$, $\lambda = \pi, \phi = 0$, mientras que la compuerta Z se obtiene al fijar $\theta = 0$, $\lambda + \phi = \pi$. A esta compuerta se la denomina general, ya que es capaz de generar cualquier compuerta de un qubit.

3.3 Bloque básico

La unidad básica de la red implementada consta de una capa de compuertas U_3 por cada qubit de entrada que tenga el sistema, posiblemente seguido de una compuerta CNOT y

una segunda capa de compuertas U_3 de salida. El ejemplo para una operación de 2 qubits se presenta en la figura 14.

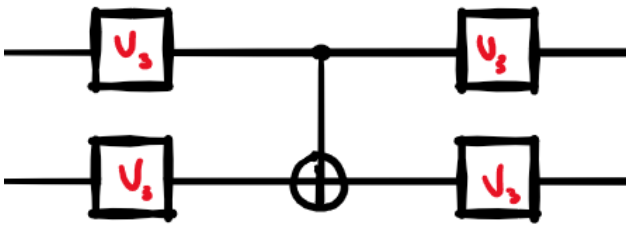

Figura 14: Bloque básico de la red

Los pesos sinápticos asociados a esta neurona son entonces los parámetros λ, ϕ y θ asociadas a U_3. Este modelo tiene un peso sináptico agregado, que es el posicionamiento de la compuerta CNOT. Al ser estas compuertas orientables por contar con un control y un target, es necesario que la red decida la correcta orientación de la compuerta en tiempo de adaptación. Como se mostró anteriormente, el conjunto de compuertas unitarias con el agregado de CNOTs son compuertas universales, y pueden representar cualquier operación unitaria arbitraria de tantos qubits como se desee.

3.4 Orientación de las compuertas CNOT

Si bien el bloque básico cuenta con una compuerta CNOT por defecto, a priori el control y el target podrían estar en cualquiera de los qubits de entrada del sistema. Esto se ilustra en la figura 15.

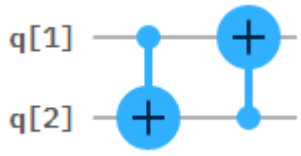

Figura 15: Orientación de CNOTs en un sistema de 2 qubits

Para que la red pueda en efecto implementar cualquier operación unitaria, el bloque tiene que poder transfigurarse en cualquiera de los dos CNOTs de la figura (15). Para poder representar el CNOT invertido, restaría con cambiar el orden de los qubits a su entrada y su salida, y de hecho la unidad básica elegida puede lograrlo, en virtud de las capas de entrada y salida de compuertas U_3, que pueden implementar una operación de $SWAP$ entre los qubits. En la tabla se muestran los parámetros asociados a las capas de entrada y salida de unitarias, de manera de poder obtener un CNOT al revés. Las primeras dos columnas corresponden a los parámetros de las unitarias de entrada para cada qubit, y las dos siguiente los asociados a las de salida. Este mismo estudio puede realizarse para comprobar que cualquier CNOT invertido es realizable, cualquiera sea el número de qubits del sistema.

	U_{3in_1}	U_{3in_2}	U_{3out_1}	U_{3out}
θ	-1.57	1.57	1.57	-1.57
ϕ	0	0	0	0
λ	0	0	0	0

Tabla 1: Parámetros de unitarias para CNOT invertida

3.5 Algoritmo de aprendizaje: Un bloque

El algoritmo de aprendizaje se divide en dos fases: una donde se elige donde y como colocar la compuerta CNOT del bloque, y una segunda fase donde se minimiza la diferencia de la fidelidad entrada-salida (función de costo) de manera de adaptar los parámetros de las compuertas unitarias. El conjunto de patrones de entrada que se utilizan consta exclusivamente de estados puros.

3.5.1 CNOTs

Como fue mencionado anteriormente, la correlación es una medida de qué tan vinculadas están dos variables aleatorias. En este caso, las variables son los resultados de la medición cada uno de los qubits del conjunto de entrada y de salida. Para esto, se computan dos tipos de correlación: la MI entre todos los pares de qubits y la CCM. Como los estados del conjunto de entrenamiento son todos puros, ambas medidas de correlación devuelven valores de entrelazamiento cuántico. La única compuerta dentro del bloque básico que es capaz de causar entrelazamiento es la CNOT, de forma que estudiando con cuidado donde surgen los cambios de correlación se puede inferir donde debe colocarse el operador CNOT en cuestión.

3.5.1.a Determinación de la cantidad de CNOTs necesarios

Si bien un bloque cuenta exclusivamente con una única compuerta CNOT, al tener que aprender algoritmos cuánticos de n qubits usualmente es necesario más de un bloque con CNOT. Más aún, si se puede estimar la cantidad de CNOTs que el algoritmo requiere, se está estimando también la cantidad de bloques mínimos necesarios que requerirá la red para aprender el algoritmo, dado a que cada uno cuenta con una única de estas compuertas.

La CCM, en virtud de las biparticiones computadas en su cálculo, es capaz de responder

esta pregunta. Esencialmente, el algoritmo computa correlaciones de subespacios de un estado, es decir, si se parte de un estado de n qubits la CCM lo quiebra progresivamente hasta llevarlo a n estados de un único qubit, en el proceso acumulando el conteo de las correlaciones que encuentra en todos los subespacios que computa. Dado el resultado, está tabulado (14) y queda acotado, cuantas compuertas CNOT son necesarias para lograr ese valor de correlación. Por tanto, primero se computan las CCM de todos los estados de entrada del sistema, y con ello se obtienen la cantidad de CNOTs necesarias.

3.5.1.b Determinación de la posición de los CNOTs

Una vez que se sabe cuántas compuertas son necesarias, surge la pregunta de dónde colocarlas. En un sistema de dos qubits hay dos orientaciones pero una única combinación control-target, conforme el sistema crece en dimensión debemos elegir por métodos numéricos donde colocar control- target y surge la necesidad de elegir dónde colocarlos por un método más refinado. Por ejemplo, en el caso de un sistema de tres qubits, los CNOTs posibles, sin contar que además pueden estar invertidos, se muestran en la figura 16.

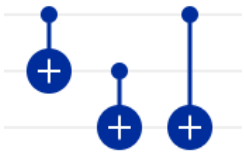

Figura 16: CNOTs posibles en un sistema de 3 qubits

Con este propósito, se construye un conjunto de entrada auxiliar combinando estados $|+\rangle$ con estados $|0\rangle$. Estos estados se permutan, empezando por un conjunto con un estado $|+\rangle$ y $n-1$ estados $|0\rangle$, luego dos estados $|+\rangle$ y $n-2$ estados $|0\rangle$, y así sucesivamente hasta generar estados con $n-1$ estados $|+\rangle$ y un único estado $|0\rangle$, donde n es el número de qubits de los estados de entrada. El estado $|+\rangle$:

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \tag{78}$$

Veamos por qué es útil definir estos conjuntos y como calcular su MI es vital para saber donde colocar los CNOTs. Sin perder generalidad, estudiaremos el caso de dos qubits. Para este caso, el conjunto auxiliar cuenta únicamente con dos elementos, puesto que $n = 2$ solo hay dos permutaciones posibles:

$$|\Psi_{aux}\rangle_1 = |+\rangle \otimes |0\rangle \tag{79}$$

$$|\Psi_{aux}\rangle_2 = |0\rangle \otimes |+\rangle \tag{80}$$

A su vez, el CNOT de un bloque de 2 qubits solo puede estar en dos posiciones como se mostró anteriormente. Veamos qué pasa si el CNOT está en su posición normal, con el control en el primer qubit y el target en el segundo, y se le ingresa el conjunto auxiliar.

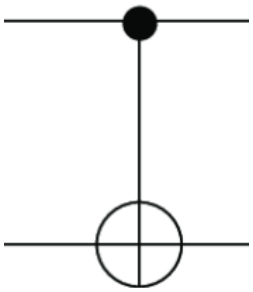

Figura 17: CNOT en su posición usual

Al ingresar $|\Psi_{aux}\rangle_1$:

$$|\Psi_{out}\rangle_1 = |+\rangle \otimes |1\rangle \tag{81}$$

Puesto que hay un $|1\rangle$ en el $|+\rangle$, este activa el control y niega e segundo qubit. Veamos que pasa al ingresar el otro estado auxiliar:

$$|\Psi_{out}\rangle_2 = |0\rangle \otimes |+\rangle \tag{82}$$

Al no haber un $|1\rangle$ en el primer qubit, el control no se activa.

Veamos entonces qué sucede en el caso que el CNOT está al revés:

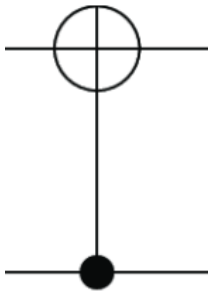

Figura 18: CNOT en su posición usual

Al ingresar el primer estado auxiliar:

$$|\Psi_{out}\rangle_1 = |+\rangle \otimes |0\rangle \tag{83}$$

Como en el control solo hay un $|0\rangle$, este no se activa.

$$|\Psi_{out}\rangle_2 = |1\rangle \otimes |+\rangle \tag{84}$$

Para una mejor visualización de los resultados:

	Derecho	Inverso
$\|\Psi_{out}\rangle_1$	$\|+\rangle \otimes \|1\rangle$	$\|+\rangle \otimes \|0\rangle$
$\|\Psi_{out}\rangle_2$	$\|0\rangle \otimes \|+\rangle$	$\|1\rangle \otimes \|+\rangle$

Entonces, como se puede apreciar, las salidas de las distintas configuraciones de CNOTs a este conjunto de entradas en particular nunca resulta en el mismo estado, y con un análisis de cada caso se puede saber donde debería estar colocada la compuerta en el bloque a implementar.

Si bien esto es válido, conforme la dimensión del sistema crece se torna crecientemente complicado. Para el caso de tres qubits, el conjunto de entrada auxiliar es:

- $|+, 0, 0\rangle$

- $|0,+,0\rangle$
- $|0,0,+\rangle$
- $|+,+,0\rangle$
- $|0,+,+\rangle$
- $|+,0,+\rangle$

En el caso genérico para n qubits, la dimensión del conjunto de estados auxiliares:

$$\#\{\Psi_{out}\} = \sum_{j=1}^{n-1} C_j^n = 2^n - 2 \tag{85}$$

3.5.2 Matrices Unitarias: Un paso de la regresión

Como se mencionó anteriormente, los pesos sinápticos a modificar en la red implementada son los parámetros de las matrices unitarias de capas de entrada y salida de cada bloque. Cabe recalcar que cada matriz cuenta con tres parámetros propios, por lo que la cantidad de pesos sinápticos que se tienen para adaptar es de:

$$\#\{\omega\} = 2(3^{n-1}) \tag{86}$$

Donde n es la cantidad de qubits del sistema.

El algoritmo implementado es el de máximo descenso, ya que se definen los valores iniciales de todos los parámetros. A su vez utiliza como función de costo la fidelidad entre la salida del sistema y la salida esperado. Si bien se trata de una optimización multivariable con posibles múltiples mínimos locales, el mínimo global que se desea alcanzar es conocido: es que la fidelidad entre los estados resultantes de la red y los de entrenamiento sea 1, puesto que esto indicaría que la salida de la red es la esperada.

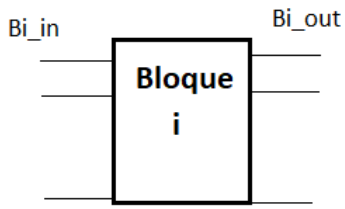

Figura 19: Un bloque

Sea el conjunto de salidas de un bloque Bi_out al igual que en la figura 19, y sea el conjunto de salidas esperadas $train\_out$. La función de costo entonces se computa como:

$$J = Fid(Bi\_out, train\_out) \tag{87}$$

Como J en la ecuación 87 es en realidad un vector, por ser Bi_out y $train\_out$ vectores, se tiene la elección de qué valor elegir del conjunto. Por defecto la red adapta con la media del vector J, pero puede modificarse de manera de trabajar con el mínimo de ser deseado, teniendo en cuenta que el uso del mínimo puede traer consigo problemas de saltos en la evolución, mientras que trabajando con el valor medio se garantiza que la evolución sea contínua. Con el valor seleccionado para entrenar el sistema computa el gradiente de J y decide hacia dónde avanzar en la superficie de error, y modifica los parámetros de las unitarias de manera acorde. Computa la nueva fidelidad entrada-salida, y si esta aumentó, modifica la celda correspondiente, y de lo contrario, retorna la misma celda que la que recibió de entrada.

3.5.3 Matrices Unitarias: Regresión completa

A partir de un paso de la regresión, la red luego se entrena en lazo de acuerdo con algunas consideraciones, el algoritmo corre hasta que se cumpla al menos una de las siguientes condiciones:

- Se llegue a la cantidad de iteraciones máximas, si la cantidad de iteraciones predefinidas no son suficientes, el programa termina y no se alcanza el resultado deseado.
- Se cumpla que la fidelidad de salida y la fidelidad media sean mayores que una fidelidad mínima. Si se cumplen estas condiciones, es porque la red logró emular dentro de la cota deseada el circuito que se buscaba: termina porque logró su cometido.
- La fidelidad se estanque y no tenga un crecimiento por encima de un mínimo en las últimas k iteraciones, con k = 10 por defecto. Si esta condición se cumple y termina el entrenamiento, es porque la fidelidad se estancó en un mínimo local y, por si solo, no puede salir de ese punto de la superficie de error.

La cantidad máxima de iteraciones, la fidelidad mínima y la derivada mínima de la fidelidad, son todos parámetros externos que el usuario define a la hora de entrenar la red. A su vez, el algoritmo cuenta con algunas medidas preventivas para intentar evitar problemas de convergencia:

- Paso variable: si la fidelidad aumentó entre el paso i y el paso $i + 1$ de la iteración, el paso al que se avanza se aumenta en una cantidad predefinida por el usuario. Si, en cambio, la fidelidad disminuye entre ambas iteraciones, el paso en la superficie de error disminuye en una cantidad predefinida. En ambos casos, el cambio del paso satura en 10 veces el paso inicial para el caso en que crece, y en 0.1 veces el paso inicial para el otro caso.
- Momento: El momento introduce un filtro para la fidelidad media de salida, dado a que pondera el cambio sostenido que tiene el vector fidelidad media en el paso i con el valor que trae de todas las iteraciones anteriores. El valor de ponderación por

defecto está fijado en 0.9, es decir, el valor del vector en los pasos anteriores tiene 90 % más peso que el valor del vector para el paso i.

Ambas medidas intentan evitar quedar atrapados en mínimos locales, así como oscilar entorno a extremos inferiores.

3.5.4 Adaptación de una red

Con el algoritmo de adaptación de un bloque funcionando, se define el algoritmo de adaptación de una red. Para ello, veamos como se concatenan los bloques:

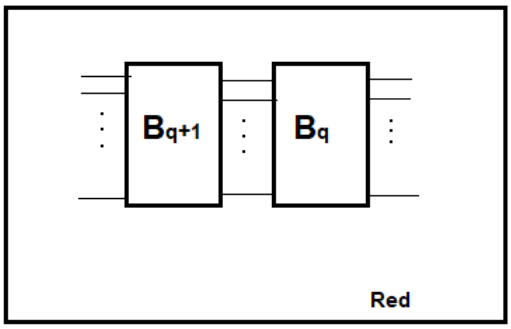

Figura 20: Cascada de bloques

Supongamos que conocemos los conjuntos de entrenamiento de entrada y salida de la red completa, y estos se denominan *train_in* y *train_out*.El primer bloque que se coloca en la red, numerado bloque 1, es de hecho el último bloque de la red puesto que se concatenan hacia atrás. En la figura 20 los bloques q y $q+1$ son dos bloques cualquiera contiguos de una red, supongamos que el bloque q ya fue adaptado a su máximo potencial pero este no alcanza para cumplir con la fidelidad mínima de salida del sistema, y que su matriz asociada, es decir, la matriz de n qubits que mapea la entrada y la salida del bloque q fue calculada y denominada B_q. Luego, se agrega un nuevo bloque vacío $q+1$, y los estados de entrada y salida con los que se entrena denominados B^{in}_{q+1} y B^{out}_{q+1} respectivamente:

$$B_{q+1}^{in} = train\_in \tag{88}$$

$$B_{q+1}^{out} = B_q^{\dagger} train\_out \tag{89}$$

El conjunto de salida se calcula de esta forma dado a que como las matrices asociadas a operadores cuánticos son hermíticas cumplen que $U^{-1} = U^{\dagger}$. De esta manera, al colocarlos en cascada, si en algún bloque se alcanza la fidelidad óptima entre entrada y salida de ese bloque, también se la alcanza entre entrada y salida de la red.

Luego para entrenar una red además de los parámetros anteriormente discutidos, se introducen un nuevo asociados a la red: la cantidad máxima de bloques. El crecimiento máximo de la red está pautado por el usuario, quien debe decidir de antemano cuantos bloques deberá tener como máximo la red. El algoritmo puede cortarse antes de lograr la fidelidad deseada, en cuyo caso el usuario deberá aumentar la cantidad de bloques de no haber sido suficientes y volver a entrenar la red.

3.6 Casos de prueba

En esta sección se presentan los resultados del desempeño de la red. La desviación máxima δ_{max} será utilizada como una herramienta más para medir el error total al completar el algoritmo, y se define como:

$$\delta_{max} = max[U_{teórica} - U_{red}] \tag{90}$$

Donde $_{teórica}$ es la matriz real del algoritmo, y U_{red} la resultante al culminar la adaptación de la red.

En los gráficos presentados, la curva azul representa la curva real de desempeño de la fidelidad conforme se itera, y la roja es un filtro de la curva anterior, a efectos de mitigar posibles oscilaciones.

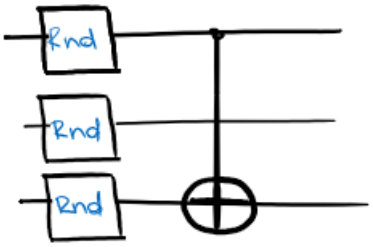

Figura 21: Prueba 1

- δ_{max} = 12 x 10^{-3}. Fidelidad de salida 1, 501 iteraciones, un bloque.

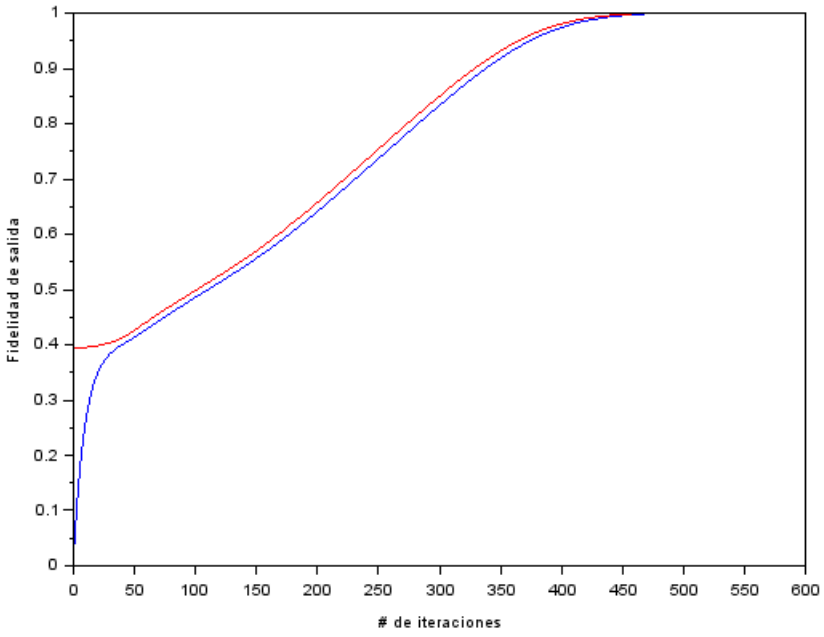

Figura 22: Evolución de la fidelidad para aprender un CNOT 1-3 con una capa de unitarias de un qubit aleatorias

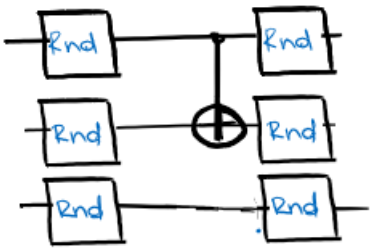

Figura 23: Prueba 2

- δ_{max} = 18 x 10^{-3}. Fidelidad de salida 1, 551 iteraciones, un bloque.

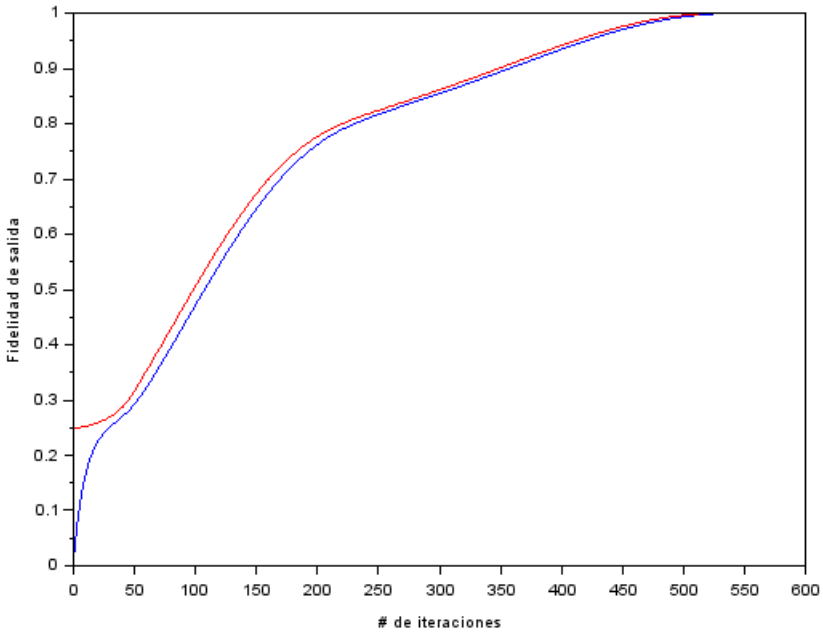

Figura 24: Evolución de la fidelidad para aprender un CNOT 2-3 con dos capas de unitarias de un qubit aleatorias

- Algoritmo de Deutsch de dos qubits - Función balanceada

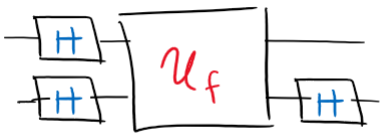

Figura 25: Prueba 3

$\delta_{max} = 38$ x 10^{-3}, 301 iteraciones, un bloque.

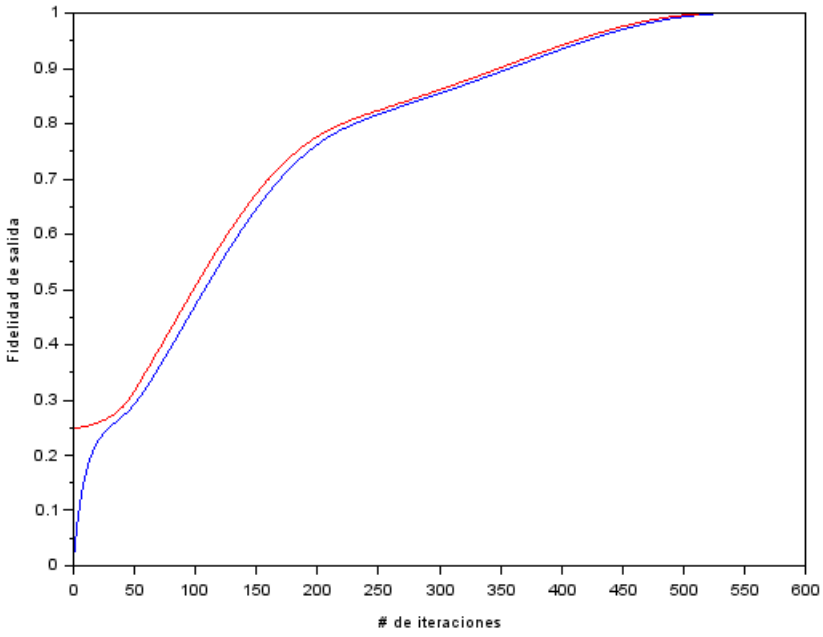

Figura 26: Evolución de la fidelidad para aprender el algoritmo de Deutsch con una función balanceada

- Dos CNOTs en cascada, el segundo al revés

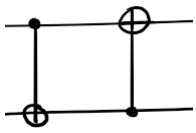

Figura 27: Prueba 4

δ_{max} = 18 x 10^{-2}, 1202 iteraciones, dos bloques

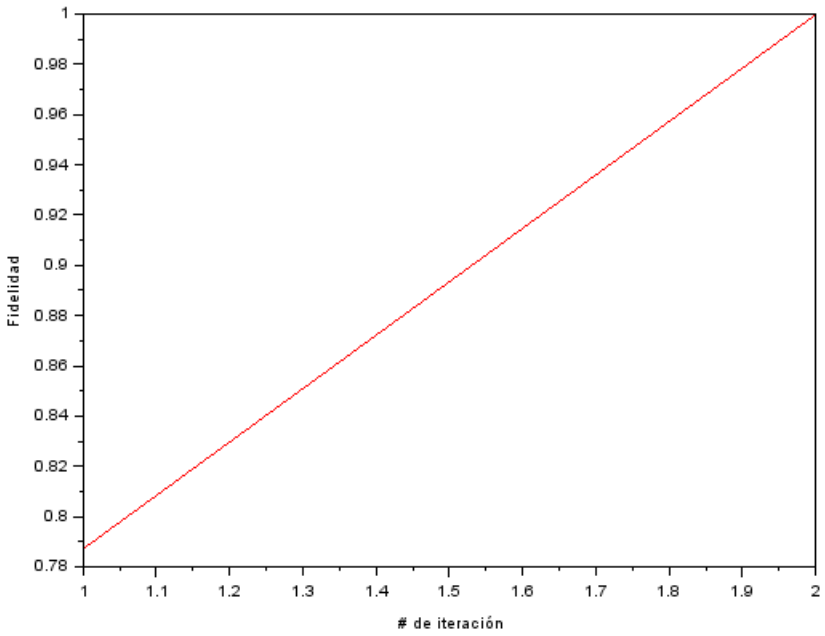

Figura 28: Evolución de la fidelidad para aprender dos CNOTs

Capítulo 4: Conclusiones

Si bien el algoritmo implementado funciona razonablemente bien, tiene algunos problemas a resolver. La idea de este trabajo, desde su comienzo, era la de poder lograr que la red aprendiera una compuerta Toffoli la cual podría verse como una especie de CNOT con dos controles, donde ambos deben activarse para negar el tercer qubit:

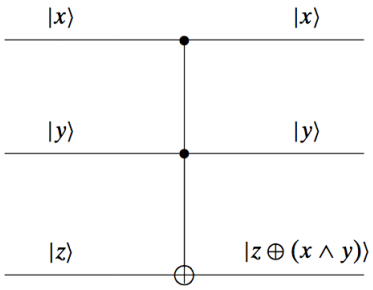

Figura 29: Compuerta Toffoli

Dado a que implementa un doble control, la cantidad de bloques de la red necesarios para emularla no son conocidos a priori, pero se le permitió a la red crecer hasta un estimado de 11 bloques sin lograr que esta diera con la compuerta correcta. Como estos, surgieron otros problemas de convergencia donde el algoritmo queda atrapado en un mínimo local a pesar del paso variable y del momento, lo cual evidencia que el algoritmo de aprendizaje seleccionado no es el óptimo. En un futuro, la primera idea a implementar sería la de una optimización que tenga en consideración la interacción de los bloques. ¿Qué implicaría eso para este trabajo? Entre otras cosas, que cuando se agrega un bloque nuevo, una capa nueva, se ajusten todos los parámetros de todos los bloques de nuevo, en vez de ajustar uno único y dejar los demás fijos. Es probable que el tiempo de cómputo aumente, pero que le sea más sencillo descubrir más algoritmos.

Otra idea sería modificar el algoritmo de aprendizaje por uno del tipo $back-propagation$, donde se propaga hacia adelante la entrada modificando los pesos sinápticos, pero una

vez que se tiene el error salida real-salida deseada, se propaga hacia atrás y se vuelven a modificar los pesos sinápticos, la idea se ilustra en la imagen a continuación.

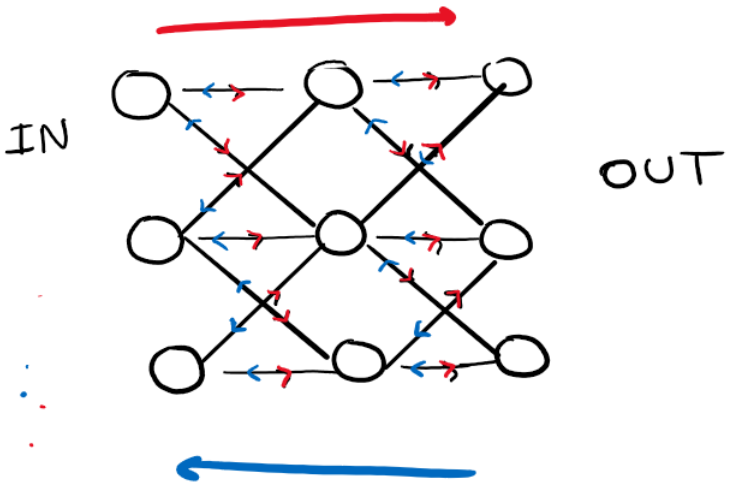

Figura 30: Propagación en ambos sentidos

Este algoritmo es una aproximación de primer orden del método de máximo descenso, y al igual que su antecesor tiene su pro y su contra. Algunos beneficios:

- Es sencillo de computar localmente
- Realiza una caminata estocástica de descenso de gradiente por la superficie de error
- No es tan inestable frente a cambios de momento y paso como el algoritmo de máximo descenso.

El algoritmo de caminata estocástica minimiza las perdidas de información con respecto al algoritmo de máximo descenso implementado en este trabajo, puesto computa el gradiente de la superficie de error de forma recursivamente hacia atrás, a partir de la regla de la cadena. La caminata estocástica podría ayudar a evitar que quede estancado en un mínimo local, pero a la vez sufre de ser de lenta convergencia. Este algoritmo también permite el uso del momento y del paso variable, dos cualidades que podrían ayudar considerablemente a palear los efectos sobre la velocidad de convergencia.

Por otro lado, probando el funcionamiento de la red frente a diversos algoritmos se detectó problemas de funcionamiento para operaciones con mayor cantidad de CNOTs que de qubits,puesto que las correlaciones no pueden estimar correctamente la cantidad de estas compuertas necesarias y por tanto hacen fallar al algoritmo completo. Sería deseable buscar alguna herramienta que pueda complementar el uso de las correlaciones para detectar cantidad y posición de CNOTs o volver a una fase anterior de este trabajo, donde todas las compuertas del bloque eran elegidas por la regresión. Esto se cambió originalmente para mejorar el desempeño del algoritmo, pero quizás sea buena idea volver a explorar esa versión para ver si puede resolver estos casos en particular.

Otra mejora inmediata que habría que implementar es el armado del traductor entre código SciLab y el Python especializado que utiliza IBM, qiskit. IBMq, la plataforma web de computación cuántica de IBM, utiliza su propio Python modificado para que investigadores de todo el mundo programen sus computadoras cuánticas en línea. La programación se reduce a concatenar operaciones, pero la virtud de este trabajo es que los bloques de la red están formados completamente por operaciones disponibles en la plataforma. Con el circuito descompuesto en bloques del tipo introducido en este trabajo, se podría programar un traductor entre lenguajes que convierta un tipo de código en otro, de manera que el usuario simplemente deba copiar y pegar el código en la plataforma de IBM para poder correrlo en una computadora cuántica real.

La primera versión del sistema QAN, como denominamos a la biblioteca desarrollada a lo largo de este trabajo, funciona razonablemente bien aprendiendo diversos algoritmos cuánticos, sin embargo considero que puede ser mejorado con algunas de las ideas propuestas en esta última sección. Es mi intención seguir trabajando en este proyecto para mejorarlo fuera del marco de este trabajo de fin de carrera.

Anexo: Manual de usuario

Funciones auxiliares útiles

```
[comb_lst, cant_comb] = comb2qbs(nqbs)
```

Toma como argumento de entrada el número de qubits del sistema y devuelve una lista de todas las combinaciones tomadas de a dos de qubits del sistema, y cuantas son en total

```
[sts, comb_lst, cant] = GeneraEstadosCeroMas(nqbs)
```

Función que genera el conjunto de estados auxiliar para asignación de CNOTs. Toma como argumento de entrada el número de qubits del sistema y devuelve una lista de todas las combinaciones de a dos qubits, los estados generados y el cardinal de este conjunto.

```
[sts, cant] = GeneraEstadosGiros(nqbs, cant_giros)
```

Función que genera estados como giros entorno a los ejes X e Y. Toma como argumento de entrada el número de qubits del sistema y la cantidad de giros deseada, y devuelve una lista con los estados generados y el cardinal de este conjunto.

Funciones asociadas a la correlación

```
[max_ccm, ind_max, v_ccm] = CalculaCCM(sts)
```

Función que toma como entrada una lista de estados vectoriales, computa su CCM y devuelve un vector v_ccm con esas correlaciones para cada vector del conjunto de entrada, cuál es máxima, max_ccm, y en qué indice se da dicho máximo, ind_max.

```
cant_CNots = CantidadCNots(val_ccm)
```

Esta función toma como entrada un valor de CCM y a partir de él, calcula la cantidad de CNOTs necesarios

```
[med_corr, corr_res, corr_max, ind_max] = OutqbsCorr(sts, comb_lst)
```

Esta función toma como entrada una lista de estados vectoriales y una lista de combinaciones dos a dos, computa la correlación del tipo mutual information para cada combinación, para cada estado, y lo devuelve en la matriz corr_res. Computa luego la media para cada qubit y la retorna en el vector med_corr, calcula cual es el máximo, corr_max y dónde se da ind_max.

Funciones asociadas a las matrices unitarias

```
U = qb_QAN_Unit3(parsU3)
```

Esta función toma como entrada un vector de parámetros ordenados θ, ϕ, λ y devuelve la matriz unitaria asociada a esos parámetros.

```
[parsU] = qb_QAN_Inv_Unit3(U)
```

Esta función es la inversa a qb_QAN_Unit3. Toma una matriz unitaria U y retorna los parámetros ordenados θ, ϕ, λ en un vector.

```
[CNot_lst] = qb_QAN_CreateCNots(nqbs, comb_lst)
```

Esta función toma el número de qubits y la lista de combinaciones dos a dos posibles, y retorna una lista de todos los CNOTs posibles del sistema

Funciones asociadas a los bloques

```
QC = qb_QAN_QCell_create(nqbs, sts_out, comb_lst, Cnot_lst)
```

Esta función toma como entrada el número de qubits, sts_out los estados de salida de la función GeneraEstadosCeroMas, la lista de combinaciones dos a dos comb_lst y la lista Cnot_lst salida de qb_QAN_CreateCnots. Retorna un objeto tipo QCell donde:

- QC(1): nqbs
- QC(2): ind_Cnot, el par de qubits donde está el CNOT del bloque
- QC(3): U_Cnot, la matriz asociada al CNOT
- QC(4): Parámetros de unitarias de entrada, de largo 3nqbs
- QC(5): Parámetros de unitarias de salida, de largo 3nqbs
- QC(6): Matriz de la celda evaluada

```
[QC, UQC] = qb_QAN_QCell_U(QC)
```

Esta función toma como entrada un bloque QCell y retorna el mismo bloque y la matriz unitaria asociada al bloque.

```
[QC, inc_fid, fid_out, vec_grad] = qb_QAN_QCell_AdaptU1(QC, sts_in, sts_out,
vec_grad, parsU1, calc_mean)
```

Esta función toma como entrada un bloque QC, un conjunto de estados de entrada sts_in y uno de salida sts_out. Luego puede tomar algunos parámetros extra, pero son opcionales, de no ser ingresados el programa les asigna un valor por defecto. vec_grad es un vector de tamaño $2(3^{nqbs})$ con los parámetros iniciales de las matrices unitarias, parsU1 es un vector de cuatro entradas: los primeros dos valores se usan para estimar el gradiente, el tercero es para estimar el paso de gradiente y el cuarto, para ponderar al hacer método de momento. El último parámetro es calc_mean, bandera que que indica si la adaptación se hace estudiando la fidelidad media o la mínima. Computa un paso de la regresión de LMS para adaptar los parámetros de las unitarias. Retorna el bloque QC, el incremento de fidelidad del paso inc_fid, la fidelidad de salida del paso fid_out y el nuevo valor de los parámetros de las matrices unitarias, vec_grad.

```
[QC_out, vec_fid, med_vec_fid] = qb_QAN_QCell_AdaptU(QC, sts_in, sts_out, pars_opt)
```

Esta función toma como entrada un bloque QC, un conjunto de estados de entrada sts_in, un conjunto de salida sts_out, y un vector de parámetros pars_opt dónde:

- pars_opt(1): Valor para estimar el gradiente
- pars_opt(2): Valor para estimar el gradiente
- pars_opt(3): Tamaño inicial del paso (gradiente)
- pars_opt(4): Momento (gradiente)
- pars_opt(5): Parámetro para aumentar el paso
- pars_opt(6): Parámetro para bajar el paso
- pars_opt(7): Cantidad máxima de iteraciones
- pars_opt(8): Porcentaje de estados de entrada para el entrenamiento
- pars_opt(9): Parámetro para detener: fidelidad mínima admisible
- pars_opt(10): Parámetro para detener: cambio mínimo de la fidelidad
- pars_opt(11): Parámetro para detener: diferencia máxima entre fidelidad y filtro.
- pars_opt(12): Parámetro para el filtro de la fidelidad

Retorna el vector de fidelidades de salida vec_fid, la media de este vector med_vec_fid y el bloque de salida adaptado QC_out.

```
QN = qb_QAN_QNet_create(nqbs)
```

Esta función toma como entrada el número de qubits y retorna un objeto QNet, QN donde:

- QN(1): nqbs
- QN(2): comb_lst
- QN(3): Lista de CNots

- QN(4): Lista de QCells
- QN(5): Matriz de la red evaluada

```
QN = qb_QAN_QNet_insertCell(QC, sts_out)
```

Esta función toma como entrada un bloque tipo QCell, QC, y un conjunto de estados que se suponen como los de salida de la función GeneraEstadosCeroMas. Retorna una red con el bloque insertado.

```
[QN, net_fid, UQN] = qn_QAN_QNet_adapt(QN, sts_in_cell, sts_out_cell,

sts_in_train, sts_out_train, pars_opt_cell, pars_opt_net)
```

Esta función toma como entrada una red QN, vacía, un conjunto de estados de entrada y de salida de la primer celda, un conjunto de estados de entrada y salida para entrenar, pars_opt_cell son los mismos parámetros que qb_QAN_QCell_AdaptU, y pars_opt_net es la cantidad máxima de celdas. Retorna una red QN, la fidelidad de la red, net_fid y la matriz asociada a la red, UQN.

Referencias

[1] Arute, F., Arya, K., Babbush, R. et al. "Quantum supremacy using a programmable superconducting processor". Nature 574, 505–510, 2019.

[2] Shor P. W. et al. "Quantum Error Correction and Orthogonal Geometry". Physical Review Letters 78.3, 405–408, 1997.

[3] Shor, Peter W. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer", Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, 20–22 de Noviembre, 1994.

[4] Grover, Lov K. " A fast quantum mechanical algorithm for database search"Proceedings, 28th Annual ACM Symposium on the Theory of Computing (STOC), paginas 212-219, 1996.

[5] David Collins, K. W. Kim, W. C. Holton,"Deutsch-Jozsa algorithm as a test of quantum computation", American Physical Society (APS), Physical Review A Vol. 58, 1998.

[6] Allende Amen, C. "Modelado de la propagación de errores en algoritmos cuánticos", 2020. [Online] https://sisbibliotecas.ort.edu.uy/cgi-bin/koha/opac-retrieve-file.pl?id=84a62abf8ab65c629fde40679412e580

[7] Nielsen, Michael A. Chuang, Isaac L. "Quantum computation and Quantum Information", Cambridge University Press. ISBN 0521635039. Página 80. 2000. Traducido del inglés.

[8] Nielsen, Michael A. Chuang, Isaac L. "Quantum computation and Quantum Information", Cambridge University Press. ISBN 0521635039. Página 81. 2000. Traducido del inglés.

[9] Nielsen, Michael A. Chuang, Isaac L. "Quantum computation and Quantum Information", Cambridge University Press. ISBN 0521635039. Página 85. 2000. Traducido del inglés.

[10] Nielsen, Michael A. Chuang, Isaac L. “Quantum computation and Quantum Information”, Cambridge University Press. ISBN 0521635039. Página 94. 2000. Traducido del inglés.

[11] Nielsen, Michael A. Chuang, Isaac L. “Quantum computation and Quantum Information”, Cambridge University Press. ISBN 0521635039. 2000. Traducido del inglés.

[12] Nielsen, Michael A. Chuang, Isaac L. “Quantum computation and Quantum Information”, Cambridge University Press. ISBN 0521635039. Páginas 191-194. 2000. Traducido del inglés.

[13] Dawson, C. M., Nielsen, M. A., ”The Solovay-Kitaev Algorithm”, Quantum Information and Computation, Vol. 0, No. 0 (2005).

[14] Fonseca De Oliveira, A. L., Buksman, E., López De Lacalle, J., Çumulative measure of correlation for multipartite quantum states”, International Journal Of Modern Physics B, Vol. 28, 2013.

[15] McCulloch, W.S., Pitts, W. .A logical calculus of the ideas immanent in nervous activity”. Bulletin of Mathematical Biophysics 5, 115–133, 1943. https://doi.org/10.1007/BF02478259

[16] Rosenblatt, F. ”The perceptron: A probabilistic model for information storage and organization in the brain”, Psychological Review, Vol. 65 No. 6, 1958

[17] Haykin, S. ”Neural Networks: A Comprehensive Foundation”, Macmillan College Publishing Company, ISBN 0023527617. 1994. Traducido del inglés

[18] Haykin, S. ”Neural Networks: A Comprehensive Foundation”, Macmillan College Publishing Company, ISBN 0023527617, pp 45. 1994. Traducido del inglés

[19] Widrow, B., Hoff M.E, “Adaptive Switching Circuits,” 1960 IRE WESCON Convention Record, 1960, pp. 96-104.

[20] Kathuria, A. Ïntro to optimization in deep learning: Gradient Descent”, 2018. [Online] https://blog.paperspace.com/intro-to-optimization-in-deep-learning-gradient-descent/

[21] N. Wiener, E. Hopf, Üeber eine Klasse singulärer Integralgleichungen"Sitzungber. Akad. Wiss. Berlin, 1931, pp. 696–706

[22] Minsky, M. Papert, S. "Perceptrons: An Introduction to Computational Geometry", The M.I.T. Press, Cambridge, Massachusets, 1969.

[23] Hornik, K. , Stinchcombe, M., White, H. , "Multilayer Feedforward Networks are Universal Approximators", Neural Networks, ISSN 0893-6080, Vol. 2, 1989.