Ansatz HEA 2D brickwork para VQA

Universidad Nacional Autónoma de México

Laboratorio Avanzado de Procesamiento de Imágenes

Académicos: Dra. Jimena Olveres Montiel y Dr. Boris Escalante Ramírez

Alumno: Sebastián González Juárez

Resumen.

Este trabajo construye una QNN, entrenando un VQA para clasificación binaria de dígitos, partiendo de imágenes 8×8 reducidas a 4×4 y codificadas en 16 qubits. Empleamos angle encoding con rotaciones R_Y, un ansatz que cuenta con rotaciones R_Y y R_X con un HEA 2D tipo *brickwork* con acoplamientos CZ, lectura de expectativas ⟨Z⟩ y una capa logística clásica. Optimizamos con Adam una pérdida BCE regularizada sobre parámetros cuánticos y clásicos. Se obtuvo un Accuracy mayor al 98%.

Palabras clave.

- Variational Quantum Algorithm.
- Ansatz HEA 2D brickwork.
- Angle encoding.
- Optimizador Adam.
- Digits dataset.

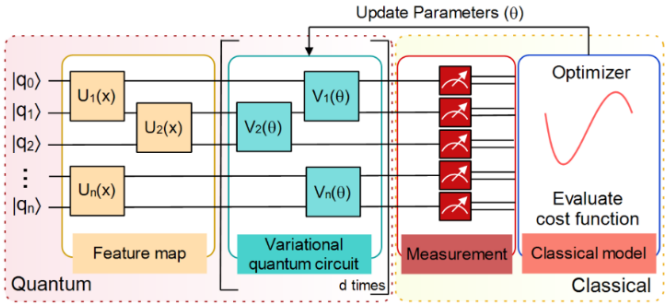
1) Introducción, Instalación e Importaciones.

1.1. ¿Qué es un VQA?

Un Variational Quantum Algorithm (VQA) es un esquema híbrido cuántico–clásico:

- Un circuito cuántico parametrizado que prepara un estado que depende de ángulos entrenables.
- Un optimizador clásico ajusta esos parámetros para minimizar una pérdida definida a partir de mediciones del circuito.

En la imagen a continuación podemos observar un diagrama que muestra la idea de un circuito VQA:



Este programa lo que hace (muy resumidamente) es:

a) Toma imágenes (digits), las reduce y normaliza, las codifica en un registro de 16 qubits (angle encoding con R_Y).

b) Aplicamos un ansatz HEA 2D brickwork.

c) Lectura de expectativas $\langle Z_i \rangle$ y usa una capa logística clásica para obtener una probabilidad de clase y entrenar con Adam.

1.2. ¿Qué necesitamos instalar?

Toda la parte de computación cuántica esta implementada con PennyLane (framework QML) y su NumPy diferenciado.

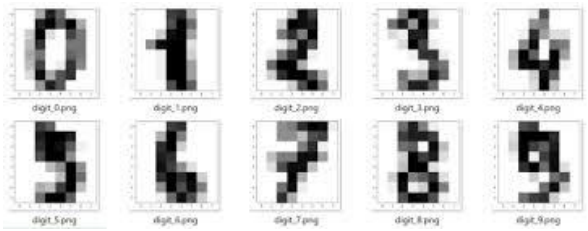
El data set es implementado por scikit-learn y además de usar este, usamos otras paqueterías clásicas.

2) Configuración y utilidades de datos. (Digits 0 vs 1, 8×8→4×4)

2.1. Data set y filtro de clases.

Este trabajo se desarrolló con el data set de digits descargado con sklearn. Este conjunto cuenta con las siguientes dimensiones:

$$D = \{(X^{(k)},\ y^{(k)})\}_{k=1}^{N},\ X^{(k)} \in \mathbb{R}^{8\times 8} \wedge y^{(k)} \in \{0,..,9\}$$



Luego definimos el subconjunto binario (clases 0 y 1):

$$D_{01} = \{(X^{(k)},\ y^{(k)})\} \in D:\ y^{(k)} \in \{0,1\}$$

2.2. Reducción espacial $8 \times 8 \rightarrow 4 \times 4$.

A continuación, hacemos una reducción espacial $8 \times 8 \rightarrow 4 \times 4$ por medio de bloques. Para una imagen $X \in \mathbb{R}^{8\times 8}$, la reducción por promedio de bloques 2×2 es un operador lineal: $\mathcal{R}: R^{8\times 8} \rightarrow R^{4\times 4}$, $Y = \mathcal{R}(X)$, definido elemento a elemento por

$$Y_{ij} = \frac{1}{4} \sum_{u \in \{2i,\, 2i+1\}} \sum_{v \in \{2j,\, 2j+1\}} X_{uv}\,, \qquad i,j = 0,1,2,3.$$

i. e. para cada elemento posición (i,j) en Y, tomamos el bloque 2×2 en X que empieza en $(2i, 2\,j)$ y lo promediamos. Por ejemplo, para $Y_{0,0}$ es el promedio de $\{X_{0,0}, X_{01}, X_{1,0}, X_{1,1}\}$.

Equivalente matricialmente, si $S \in \mathbb{R}^{4\times 8}$ promedia pares adyacentes,

$$S_{i,\,2i} = \frac{1}{2}, \quad S_{i,\,2i+1} = \frac{1}{2 \quad (i = 0,1,2,3)},$$

$$S_{i,j} = 0 \text{ en otro caso,}$$

De este modo la reducción completa es:

$$Y = \mathcal{R}(X) = SXS^{\top}$$

- SX: promedia filas de X de dos en dos $(0+1, 2+3, 4+5, 6+7)$
- $(SX)S^{\top}$: ahora promedia columnas del resultado, también de dos en dos.
  Esto reproduce exactamente el promedio por bloques 2×2.

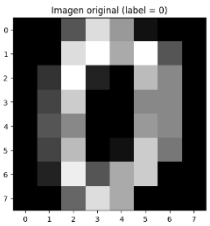
Al aplicar matrices con vec(·), se cumple:

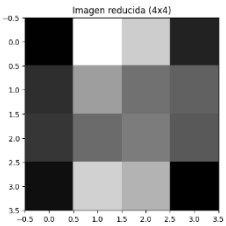
$$\text{vec}(SXS^{\top}) = (S \otimes S)\text{vec}(X)$$

Así, $\mathcal{R}$ es operador lineal con su matriz $(S \otimes S) \in \mathbb{R}^{16\times 64}$.

Es como verlo como una correlación con el kernel

$$K = \frac{1}{4}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$





2.3. Normalización global a $[0, 1]$.

Consideremos $\chi = \{X^{(k)}\}_{k=1}^{N_{01}}$, siendo sus extremos globales:

$$m_{\min} = \min_{k,u,v} X_{uv}^{(k)}\,, \qquad m_{\max} = \max_{k,u,v} X_{uv}^{(k)}$$

La normalización afín global (con $\varepsilon > 0$) es:

$$\hat{X}^{(k)} = \frac{X^{(k)} - m_{\min}\,\hat{1}}{m_{\max} - m_{\min} + \varepsilon} \in [0,1]^{8\times 8}$$

Se muestra un ejemplo a continuación:

$$X = \begin{pmatrix} 2 & 5 \\ 3 & 7 \end{pmatrix}, \qquad m_{\min} = 2, \qquad m_{\max} = 7$$

$$\hat{X} = \begin{pmatrix} 0 & 0.6 \\ 0.2 & 1 \end{pmatrix}$$

2.4. Vectorización.

Al momento tenemos las imágenes reducidas y normalizadas $Y^{(k)} \in [0,1]^{4\times 4}$, la vectorizamos a

$$\mathrm{x}^{(k)} = \text{vec}\big(Y^{(k)}\big) \in [0,1]^{16}$$

Se dé prioridad al orden en filas,

$$p = 4i + j \;:\; i,j \in \{0,1,2,3\} \;\wedge\; \mathrm{x}_p^{(k)} = Y_{ij}^{(k)}$$

2.5. Partición entrenamiento/prueba con estratificación.

Consideremos a la fracción de prueba $\rho \in (0,1)$ la fracción de prueba y π una permutación aleatoria, i.e. una semilla fija. Con $N_{tr} = \lfloor (1-\rho)N_{01} \rfloor$:

$$\mathcal{T}_{tr} = \{\mathrm{x}^{(\pi(k))}, y^{(\pi(k))}\}_{k=1}^{N_{tr}}$$
$$\mathcal{T}_{te} = \{\mathrm{x}^{(\pi(k))}, y^{(\pi(k))}\}_{k=N_{tr}+1}^{N_{01}}$$

La estratificación preserva aproximadamente la proporción de cada clase:

$$\frac{\#\{y = n \text{ en } \mathcal{T}_m\}}{N_{rt}} \approx \frac{\#\{y = n \text{ en } D_{01}\}}{N_{01}}$$

$$\mathcal{T}_m = \{\mathcal{T}_{tr}, \mathcal{T}_{te}\} \;\wedge\; n = \{0,1\}$$

3) Conectividad 2D y patrones brickwork.

3.1. Rejilla y numeración de wires.

El nombre wires viene de usar la paquetería PennyLane. Se crea una función que mapea filas y columnas → wire (orden filas):

$$wire(r,c) = rn_{cols} + c, \qquad r,c \in \{0,\dots,3\}$$

De modo que $r = wire/n_{cols}$ y $c = wire \bmod (n_{cols})$. Recordando que tenemos rejilla en 2D tal que:

$$n_{rows} \times n_{cols} = 4 \times 4$$

El conjunto de qubits (wires): $W = \{0, 1, \dots, 15\}$.

Ejemplo para 4×4:

Fila 0:(0,0) → 0, (0,1) → 1, (0,2) → 2, (0,3) → 3
Fila 1:(1,0) → 4, (1,1) → 5, (1,2) → 6, (1,3) → 7
Fila 2: (2,0) → 8, (2,1) → 9, (2,2) → 10, (2,3) → 11
Fila 3: (3,0) → 12, (3,1) → 13, (3,2) → 14, (3,3) → 15

3.2. Aristas horizontales y verticales tipo brickwork.

La idea consiste en definir pares de qubits vecinos (aristas) que se acoplan con compuertas de 2 qubits en patrones alternados. Para un desplazamiento off ∈ {0,1}:

Horizontales.

$$E_H(\text{off}) = \{(wire(r,c), wire(r,c+1) : r \in [0..3], \\ c = \text{off}, \text{off}+2, \dots, \leq n_{cols} - 2\}$$

off=0: empareja $(0-1)$, $(2-3)$ en cada fila.

off=1: empareja $(1-2)$ en cada fila.

El conteo para nuestras $n_{col} = 4$ queda:

$$|E_H(0)| = 4 \times 2 = 8, \text{aristas (dos por fila)}$$

$$|E_H(1)| = 4 \times 1 = 4, \text{aristas (una por fila)}$$

Verticales.

$$E_V(\text{off}) = \{(wire(r,c), wire(r+1,c) : c \in [0..3], \\ r = \text{off}, \text{off}+2, \dots, \leq n_{rows} - 2\}$$

off=0: empareja $(0-1)$, $(2-3)$ en cada columna.

off=1: empareja $(1-2)$ en cada columna.

El conteo para nuestras $n_{rows} = 4$ queda:

$$|E_V(0)| = 4 \times 2 = 8, \text{aristas (dos por columna)}$$

$$|E_V(1)| = 4 \times 1 = 4, \text{aristas (una por columna)}$$

3.3. Patrón por capa (alternancia brickwork).

Par las capas $\ell = 0, 1, 2, \dots$ del ansatz:

- Usamos horizontales con $\text{off}_H(\ell) = \ell \bmod(2)$.
- Usamos verticales con $\text{off}_V(\ell) = (\ell + 1) \bmod(2)$.

Para la capa ℓ se induce el conjunto de acoplamientos:

$$E^{(\ell)} = E_H\big(\ell \bmod(2)\big) \cup E_V\big((\ell+1)\big) \bmod(2)$$

- En la capa 0: $E_H(0)$ y $E_V(1)$
- En la capa 1: $E_H(1)$ y $E_V(0)$

Con la alternación de ℓ se garantiza que todas las parejas de vecinos (H y V) se activen al menos una vez cada dos capas. Explícitamente:

Capa $\ell = 0$:	**Capa $\ell = 0$:**
$E_H(0)$ (por filas): Fila 0: (0,1), (2,3) Fila 1: (4,5), (6,7) Fila 2: (8,9), (10,11) Fila 3: (12,13), (14,15)	$E_H(1)$ (por filas): Fila 0: (1,2) Fila 1: (5,6) Fila 2: (9,10) Fila 3: (13,14)
$E_V(1)$(por columnas): Col 0: (4,8) Col 1: (5,9) Col 2: (6,10) Col 3: (7,11)	$E_V(1)$(por columnas): Col 0: (0,4), (8,12) Col 1: (1,5), (9,13) Col 2: (2,6), (10,14) Col 3: (3,7), (11,15)

Vemos que ninguno de los qubits aparece en dos pares distintos de la misma sub etapa, lo cual nos permite aplicar compuertas en paralelo.

4) Dispositivo, feature map y ansatz HEA 2D brickwork.

4.1. Dispositivo y estado inicial.

En este caso se propone una profundidad de 4 capas para nuestro ansatz, i. e. nuestro HEA 2D brickwork contará con sus 4 capas.

En PennyLane, si shots = None, nuestro simulador calculara expectativas exactas i. e. analíticas de los observables sin ruido estadístico. Podríamos también usar shots = s (entero) y la expectativa $\langle O \rangle$ se estima por promedio de S mediciones (Monte Carlo):

$$\hat{\mu} = \frac{1}{S}\sum_{s=1}^{S} Z_s, \;\; Z_s \in \{+1, -1\}, \;\; Var(\hat{\mu}) = \frac{1 - \hat{\mu}^2}{S}$$

Mayor S implicaría menor varianza lo que requeriría más tiempo de cómputo.

Finalmente creamos el dispositivo de ejecución PennyLane con los 16 qubits y además podríamos utilizar "lightning.qubit" (si fuera mayor a 20 qubits) para agilizar el tiempo en CPU o backends de hardware/simuladores externos. Para este notebook, default.qubit es suficiente y estable.

4.2. Feacture Map.

Tenemos las entradas clásicas x $\in [0,1]^{16}$ y definiremos los ángulos (no entrenables) como:

$$\theta_i = \pi x_i \in [0, \pi], \qquad i = 0, \dots, 15$$

Donde cada componente x_i es la intensidad de un píxel de 4×4 normalizado. Ej.: $x_0 = Y_{0,0}, \dots, x_{15} = Y_{3,3}$. Usamos $[0, \pi]$ y no $[0, 2\pi]$, pues $\cos(\pi x_i)$ recorre todo $[-1, 1]$ una sola vez evitando ambigüedades periódicas.

Usamos requires_grad=False para no entrenarlos.

La idea es que buscamos incrustar los datos clásicos (16 pixeles) en un estado cuántico de 16 qubits. Ahora podemos aplicar una rotación local R_Y, el cual es un operador de rotación alrededor de Y.

$$R_Y(\theta) = e^{-\frac{i\theta}{2}Y} = \begin{pmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix}$$

$$R_Y(\theta)|0\rangle = \cos\frac{\theta}{2}|0\rangle + \sin\frac{\theta}{2}|1\rangle$$

Con la probabilidad de medir los estados de:

$$\Pr(0) = \cos^2\frac{\theta}{2} \;\wedge\; \Pr(1) = \sin^2\frac{\theta}{2}$$

Valor Esperado	
$x_i = 0 \Rightarrow \theta = 0 \Rightarrow \|0\rangle$	$\langle Z\rangle = +1$
$x_i = 1 \Rightarrow \theta = \pi \Rightarrow \|1\rangle$	$\langle Z\rangle = -1$
$x_i = 1/2 \Rightarrow \theta = \pi/2 \Rightarrow \|0\rangle$	$\langle Z\rangle = 0$

Aplicando a todos los qubits, i. e. a x:

$$U_{enc}(\mathrm{x}) = \bigotimes_{i=0}^{15} R_Y(\theta_i) = \bigotimes_{i=0}^{15} R_Y(\pi x_i)$$

De modo que, el estado codificado (antes de cualquier entrelazamiento) es:

$$|\psi_{enc}(\mathrm{x})\rangle = U_{enc}(\mathrm{x})|0\rangle = \bigotimes_{i=0}^{15}\left[\cos\frac{\pi x_i}{2}|0\rangle + \sin\frac{\pi x_i}{2}|1\rangle\right]$$

En el código, requires_grad=False impide que el gradiente fluya para x, i. e. significa que θ_i se consideran constantes durante la optimización.

Observación:

- R_Z no sirve para codificar intensidades si arrancamos en $|0\rangle$, pues solo cambia la fase.

$$R_Z(\phi) = e^{-\frac{i\phi}{2}Z} = \begin{pmatrix} e^{-\frac{i\phi}{2}} & 0 \\ 0 & e^{+\frac{i\phi Z}{2}} \end{pmatrix} \Rightarrow R_Z(\phi)|0\rangle = e^{-i\phi/2}|0\rangle$$

  Solo se añade una fase global al estado $|0\rangle$ (no observable), $\Pr(1) = 0, \langle Z\rangle = +1$ para todo ϕ.

  Podríamos usar una Hadamard junto a R_Z, sin embargo, esto sería equivalente a usar R_X o R_Y.

- R_X codifica bien, pero con fase imaginaria, pero se podría usar también:

$$R_X(\theta)|0\rangle = \cos\frac{\theta}{2}|0\rangle - i\sin\frac{\theta}{2}|1\rangle$$

4.3. Ansatz - Rotaciones locales entrenables (bloque variacional de 1 qubit).

Por capa $\ell = 0, \ldots, L-1$ $(L = n\_layers)$, asignamos dos parámetros por qubit: $\alpha_{\ell,i}, \beta_{\ell,i} \in \mathbb{R}$. De modo que:

$$\alpha^{(\ell)} \equiv (\alpha_{\ell,0}, \ldots, \alpha_{\ell,N-1}) \;\wedge\; \beta^{(\ell)} \equiv (\beta_{\ell,0}, \ldots, \beta_{\ell,N-1}) \in \mathbb{R}^N$$

$$U_{\mathrm{loc}}^{(\ell)}(\alpha^{(\ell)}, \beta^{(\ell)}) = \bigotimes_{i=0}^{N-1} U_{\mathrm{loc},i}^{(\ell)} = \bigotimes_{i=0}^{15}\left(R_Z(\beta_{\ell,i})\, R_Y(\alpha_{\ell,i})\right).$$

Esto quiere decir que a cada qubits le aplicaremos su correspondiente $R_Y(\alpha_{\ell,i})$ y después $R_Z(\beta_{\ell,i})$, donde

Lo que modificamos al entrenar son los escalares $\alpha_{\ell,i}$ y $\beta_{\ell,i}$.

4.4. Ansatz - Entrelazamiento tipo brickwork (bloques de 2 qubits).

Usamos compuertas CZ entre pase de vecinos definidos por los conjuntos de aristas de la rejilla 4×4, recordando:

- Horizontales con desplazamiento off $\in \{0, 1\}$:

$$E_H(\mathrm{off}) = \{(wire(r,c), wire(r,c+1) : r \in [0..3],\; c = \mathrm{off}, \mathrm{off}+2, \ldots, \leq 2\}$$

- Verticales con desplazamiento off $\in \{0, 1\}$:

$$E_V(\mathrm{off}) = \{(wire(r,c), wire(r+1,c) : c \in [0..3],\; r = \mathrm{off}, \mathrm{off}+2, \ldots, \leq 2\}$$

Por capa ℓ alternamos:

$$\mathrm{off}_H(\ell) = \ell \bmod 2 \;\wedge\; \mathrm{off}_V(\ell) = (\ell+1) \bmod 2$$

Como dentro de cada conjunto los pares son disjuntos, los CZ conmutan entre sí y se pueden aplicar en paralelo:

$$U_H^{(\ell)} = \prod_{(a,b)\in E_H(\ell \bmod 2)} CZ_{a,b}$$

$$U_V^{(\ell)} = \prod_{(a,b)\in E_V((\ell+1) \bmod 2)} CZ_{a,b}$$

Con $CZ = \mathrm{diag}(1,1,1,-1)$, la cual añade fase -1 solo al estado $|11\rangle$. Entrelaza porque viene rodeada de rotaciones R_Y y R_X que no conmutan con Z. El hecho de que los CZ conmuten (las aristas son disjuntas) nos permite aprovechar el paralelismo cuántico dentro de un mismo conjunto E_H y E_V.

4.5. Unitario de una capa y unitario total del ansatz.

De este modo, estamos aplicando: rotaciones locales → horizontales → verticales. El unitario de la capa ℓ es:

$$U_{layer}^{(\ell)}\left(\alpha^{(\ell)},\beta^{(\ell)}\right)=U_V^{(\ell)}U_H^{(\ell)}U_{\text{loc}}^{(\ell)}\left(\alpha^{(\ell)},\beta^{(\ell)}\right)$$

Para L capas, el ansatz total es:

$$U_{\text{ans}}(\Theta)=\prod_{\text{l}=0}^{L-1}U_{\text{layer}}^{(\ell)}\left(\alpha^{(\ell)},\beta^{(\ell)}\right)$$

Finalmente, el estado antes de medir para una entrada x es:

$$|\psi(\text{x},\Theta)\rangle=U_{\text{ans}}(\Theta)U_{enc}(\text{x})|0\rangle$$

5) QNode (multi-readout) y probabilidad de clase 1.

5.1. Observables de medición y estado del circuito.

Una vez armado nuestro circuito, recordemos que tenemos $N=16$ qubits, para una entrada $\text{x}\in[0,1]^{16}$ y parámetros Θ. Definimos el observable Pauli-Z en el qubit i como:

$$Z_i=I^{\otimes i}\otimes Z\otimes I^{\otimes(N-i-1)},\qquad Z=\begin{pmatrix}1&0\\0&-1\end{pmatrix}$$

Pues los observables Z_i conmutan entre sí $\left([Z_i,Z_j]=0\right)$, por lo que son simultáneamente medibles en la base computacional. La expectativa (valor medio o esperado) de Z_i que devuelve QNode para ese qubit en el estado final es:

$$z_i(\text{x},\Theta)=\langle\psi(\text{x},\Theta)|Z_i|\psi(\text{x},\Theta)\rangle$$

Lo cual es justamente nuestra función de costo, la cual hemos visto antes, pero con la expansión:

$$z_i(\text{x},\Theta)=\langle 0|U_{enc}(\text{x})U_{\text{ans}}(\Theta)Z_iU_{\text{ans}}(\Theta)U_{enc}(\text{x})|0\rangle$$

Por otro lado, también sabemos que:

$$z_i(\text{x},\Theta)=\langle\psi(\text{x},\Theta)|Z_i|\psi(\text{x},\Theta)\rangle=\text{Tr}(\rho(\text{x},\Theta)Z_i)\in[-1,1]$$

(demostrado en el anexo)

Donde: $|\psi(\text{x},\Theta)\rangle=U_{\text{ans}}(\Theta)U_{enc}(\text{x})|0\rangle=U(\text{x},\Theta)|0\rangle$

$$\Rightarrow\rho(\text{x},\Theta)=|\psi(\text{x},\Theta)\rangle\langle\psi(\text{x},\Theta)|$$

Siendo la matriz de densidad: $\rho=U\rho_0U^\dagger,\ \rho_0=|0\rangle\langle 0|$

Así, para todas las entradas:

$$z(\text{x},\Theta)=(z_0,z_1,\dots,z_{N-1})^\top\in[-1,1]^N$$

$$\Rightarrow\ z(\text{x},\Theta)\in[-1,1]^{16}$$

Donde $\top$ indica la transpuesta, i. e., escribiéndolo como vector columna. Con cada componente z_i siendo el valor esperado del observable de Pauli en el qubit i.

La interpretación resumida es la siguiente:

- Si $z_i\approx+1$, el qubit i "está más cerca" de $|0\rangle$.
- Si $z_i\approx-1$, el qubit i "está más cerca" de $|1\rangle$.
- Si $z_i\approx 0$, el qubit i está en una superposición casi equiprobable.

Todo esto se hace con la línea: [qml.expval(qml.PauliZ(i)) for i in range(n_qubits)].

5.2. Capa de lectura clásica. (lineal + sigmoide)

Para esta capa trabajaremos ya con parámetros clásicos los cuales será entrenables:

$$\text{vector w}_{\text{read}}\in\mathbb{R}\ \wedge\ \text{sesgo}b_{\text{read}}\in\mathbb{R}$$

Definimos el logit lineal:

$$t(\text{x},\Theta,\text{w}_{\text{read}},b_{\text{read}})=-\text{w}_{\text{read}}^\top z(\text{x},\Theta)+\ b_{\text{read}}$$

Donde recordemos la salida cuántica del QNode:

$$z(\text{x},\Theta)=(\langle Z_0\rangle,\dots,\langle Z_{N-1}\rangle)^\top\in[-1,1]^N$$

En este punto podemos empezar a visualizar hacia a donde nos dirigimos, pues ya son datos clásicos y es la base de una CNN.

El signo negativo proviene de una convención, pues como sabemos $\langle Z\rangle=+1$ significa "más alineado con $|0\rangle$" y $\langle Z\rangle=-1$ "más alineado con $|1\rangle$", en tareas donde la clase positiva se asocia a $|1\rangle$ en ciertos qubits, conviene que valores grandes de z_j (cercanos a $+1$) disminuyan el logit (restando evidencia de clase 1). Se podría absorber en w_{read}.

A continuación, aplicamos una función de activación sigmoide (logística):

$$\sigma(t)=\frac{1}{1+e^{-t}},\qquad p(\text{x})=\sigma\left(t(\text{x},\Theta,\text{w}_{\text{read}},b_{\text{read}})\right)\in(0,1)$$

Esta función de activación convierte el logit en una probabilidad calibrada para clasificación binaria.

6) Pérdida, exactitud y mini-batches.

6.1. Pérdida. (BCE + L_2)

Una vez ya tenemos nuestra probabilidad logística $\left(p(\text{x})\right)$, utilizamos Binary Cross-Entropy (BCE):

$$l(\text{x},\text{y})=-\left[y\log\left(p(\text{x})\right)+(1-y)\log\left(1-p(\text{x})\right)\right]$$

Para un mini batch $B=\left\{\left(\text{x}^{(\text{j})},y^{(j)}\right)\right\}_{j=1}^{|B|}$, la perdida de datos promedio es:

$$\text{L}_{\text{data}}(B)=\frac{1}{|B|}\sum_{j=1}^{|B|}l\left(\text{x}^{(\text{j})},y^{(j)}\right)$$

Para evitar log 0, se recorta a p a $[\varepsilon, 1-\varepsilon]$, con $\varepsilon = 10^{-9}$.

Luego de esto agregaremos L_2, una regularización que funcionara como penalización cuadrática sobre todos los parámetros entrenables:

$$\mathrm{R}(\Theta, \mathrm{w}_{\mathrm{read}}, b_{\mathrm{read}}) = \|\Theta\|_2^2 + \|\mathrm{w}_{\mathrm{read}}\|_2^2 + b_{\mathrm{read}}^2$$

La pérdida total será:

$$\mathrm{L}(B) = \mathrm{L}_{\mathrm{data}}(B) + \lambda \mathrm{R}(\Theta, \mathrm{w}_{\mathrm{read}}, b_{\mathrm{read}})$$

Donde $\lambda \geq 0$ es el hiperparámetro de regularización L_2.

6.2. Exactitud. (accuracy)

Definimos la predicción binaria con umbral 0.5:

$$\hat{y}(\mathrm{x}) = 1\{p(\mathrm{x}) \geq 0.5\}$$

La accuracy en un conjunto S es:

$$Acc(S) = \frac{1}{|S|} \sum_{(\mathrm{x},y)\in S} 1\{\hat{y}(\mathrm{x}) = y\}$$

6.3. Mini-batches (SGD) y barajado.

Tenemos el conjunto de entrenamiento:

$$\mathcal{T}_{tr} = \left\{\mathrm{x}^{(k)}, y^{(k)}\right\}_{k=1}^{N_{tr}}$$

Un mini-batch es un subconjunto $B \subset \mathrm{T}_{tr}$ de tamaño $|B|$.

$$B = \left\{\left(\mathrm{x}^{(\mathrm{j})}, y^{(j)}\right)\right\}_{j=1}^{|B|}$$

Podemos utilizarlos pues el gradiente de un mini-batch es un estimador insesgado del gradiente completo (si el muestreo es uniforme):

$$E_B[\nabla \mathrm{L}(B)] = \nabla \mathrm{L}(\mathcal{T}_{tr})$$

Con menor costo por iteración y mayor varianza, lo que implica reducir con $m = |B|$ y con las épocas.

7) Inicialización, optimizador y entrenamiento.

7.1. Hiperparámetros.

En esta sección comenzaremos escogiendo los hiperparámetro:

- batch_size = 64. Es el núm. de ej. usados para estimar el gradiente en cada paso (mini-batch). Escogemos 64 pues cumple las siguientes 2 cosas necesarias: Nos es lo demasiado pequeño como par causar ruido al gradiente, ni grande para ser demasiado lento por poder de cómputo.
- epochs = 10. El número de pasadas que completara el conjunto de entramiento. Para este caso hemos visto que con 10 épocas es suficiente, un mayor número de épocas significara mayor tiempo de ejecución y para un entorno de ejecución de TPU prestado por colab, aproximadamente tardamos entre 10 a 11 min. Por época.
- stepsize = 0.03 (Adam). Es la tasa base α del optimizador Adam. Como usamos shots=None (expectativas analíticas) el gradiente es menos ruidoso. La idea sería ver si hay oscilaciones o "diverge", bajar a 0.01; si todo es muy lento, probar con 0.05.
- l2_lambda=1e-4. Es el coeficiente de regularización L2 el cual habíamos visto que penaliza a $\mathrm{R}(\Theta, \mathrm{w}_{\mathrm{read}}, b_{\mathrm{read}})$. Al ser un valor muy "suave" ayuda a evitar tanto ángulos grandes como pesos enormes. La idea sería ver si hay overfitting lo subiríamos a 1e-3 y si el los no baja, lo modificaríamos a 1e-5 o directamente a 0.

7.2. Inicialización de pesos cuánticos: (n_layers, n_qubits, 2) para RY y RZ.

Tenemos:

- n_layers = L: número de capas del ansatz.
- n_qubits = N: número de qubits (16 en tu rejilla 4×4).
- 2: dos parámetros por qubit y por capa (recordemos α para R_Y y β para R_Z)

Es decir:

weights[l, i, 0] $\equiv \alpha_{l,i}$ (ángulo de R_Y del qubit i en la capa l).

weights[l, i, 1] $\equiv \beta_{l,i}$ (ángulo de R_Z del qubit i en la capa l).

Usamos qnp es pennylane.numpy lo cual hace que PennyLane envuelva a NumPy para que sea diferenciable con autograd. Así que convertimos nuestro ndarray de Numpy en un tensor diferencibale manejado por PennyLane.

Una vez con esto, requires_grad=True le dice a PennyLane/autograd que rastree gradientes con respecto a este tensor. Si fuese False, estos pesos se tratarían como constantes y no se actualizarían en el entrenamiento.

Esto es lo que nos permite que opt.step(...) (Adam) calcule $\nabla_{weights}$ mediante parameter-shift en el QNode.

Veamos que usamos rng, lo cual es un generador de NumPy creado para rng = np.random.default_rng(SEED), declarado al inicio. Y luego aplicamos una estandarización normal donde cada entrada $\mathcal{N}(0,1)$ nos dice que la media es 0 y la varianza es 1.

En otras palabras, rng.standard_normal(shape) es que estamos muestreando "Independent and Identically Distributed" (Independientes e Idénticamente Distribuidos).

Y que si todos los parámetros empezaran iguales (p. ej., todo 0), qubits/capas tenderían a aprender lo mismo. Una distribución aleatoria con media 0 rompe esa simetría desde el paso 1. Además, como la media es 0, entonces no hay sesgo inicial hacia rotar a favor o en contra (equiprobable). Finalmente es i. i. d. ya que cada aúngo empieza independientemente y así se evitan correlaciones artificiales entre R_Y y R_Z o entre qubits.

Con la escala pequeña (0.01), hace que cada entrada sea ahora $\mathcal{N}(0{,}0.01^2)$ lo que se interpreta en una desviación típica ≈ 0.01 rad (~0.57°).

Mantiene el ansatz muy cerca de la identidad, con pequeñas rotaciones aleatorias que rompen la simetría, pero no "revuelven" el estado.

7.3. Inicialización de la capa de lectura. (clásica)

Buscamos crear a w_read (vector de 16 ceros) y b_read (0) como parámetros entrenables que se actualizarán con el Adam.

Así que np.zeros(n_qubits, dtype=np.float32) nos crea el arreglo de 16 qubits y como vimos, con qnp.array(..., requires_grad=True) convertiremos ese array a pennylane.numpy (qnp), que soporta autograd y así poder calcular los gradientes respecto a este tensor entrenados con Adam.

Recordemos que w_read es el vector de pesos de la capa lineal de lectura:

$$t(\mathrm{x}, \Theta, \mathrm{w}_{\mathrm{read}}, b_{\mathrm{read}}) = -\mathrm{w}_{\mathrm{read}}^{\top} z(\mathrm{x}, \Theta) + b_{\mathrm{read}}$$

De forma análoga podemos entender la línea de código para b_read que desplaza el logit. Inicializamos estos valores en 0, pues implicaría que $t = 0$ para cualquier $z(\mathrm{x}, \Theta)$, lo que nos lleva a que $p = \sigma(0) = 0.5$.

Así que, durante el entrenamiento, en este caso autograd calcularа $\frac{\partial \mathrm{L}}{\partial w}$ y $\frac{\partial \mathrm{L}}{\partial b}$, Adam actualizará ambos junto con los pesos cuánticos.

7.4. Optimizador Adam.

Utilizaremos opt = qml.AdamOptimizer(stepsize=stepsize) para crear una instancia del optimizador Adam de PennyLane. Adam (Adaptive Moment Estimation) es un optimizador que actualiza los parámetros aprendiendo de sus gradientes y suavizando/promediando su historial.

En otras palabras, calcula el promedio del gradiente (tendencia) y el promedio del cuadrado del gradiente (qué tan variable/ruidoso es). Este ajustara automáticamente el tamaño de paso por parámetro:

- Inestables → pasos más pequeños.
- Estables → pasos más grandes.

Por otro lado, stepsize = 0.03 significa que cada actualización mueve los parámetros aproximadamente un 3% del vector de corrección que propone el optimizador.

Lo usaremos en el siguiente punto, para poder minimizar la perdida (BCE+L2) ajustando:

- Los ángulos cuánticos weights.
- La capa de lectura clásica $\mathrm{w}_{\mathrm{read}}$ y b_{read}.

7.5. Bucle de entrenamiento.

Démonos un intervalo para recordar nuestro conjunto de datos y sus notaciones:

- Train: $D_{tr} = \left\{\left(X^{(k)},\ y^{(k)}\right)\right\}_{k=1}^{N_{tr}}$, con $y^{(k)} \in \{0, 1\}$.
- En una época e, se baraja D_{tr} y se parte en mini-lotes $B_{e,t}$ de tamaño $m = batch\_size$.
- Lectura cuántica:

$$z(\mathrm{x}, \Theta) = (\langle Z_0 \rangle, \ldots, \langle Z_{15} \rangle)^{\top} \in [-1, 1]^{16}$$

- Capa lineal + sigmoide:

$$t(\mathrm{x}, \Theta, \mathrm{w}_{\mathrm{read}}, b_{\mathrm{read}}) = -\mathrm{w}_{\mathrm{read}}^{\top} z(\mathrm{x}, \Theta) + b_{\mathrm{read}}$$

$$\sigma(t) = \frac{1}{1 + e^{-t}}, \qquad p(\mathrm{x}) = \sigma\big(t(\mathrm{x}, \Theta, \mathrm{w}_{\mathrm{read}}, b_{\mathrm{read}})\big) \in (0,1)$$

- BCE puntual:

$$l(\mathrm{x}, \mathrm{y}) = -\big[y \log\big(p(\mathrm{x})\big) + (1 - y) \log\big(1 - p(\mathrm{x})\big)\big]$$

- Regularización:

$$\mathrm{R}(\Theta, \mathrm{w}_{\mathrm{read}}, b_{\mathrm{read}}) = \|\Theta\|_2^2 + \|\mathrm{w}_{\mathrm{read}}\|_2^2 + b_{\mathrm{read}}^2$$

Fijando el objetivo del mini-batch $B_{e,t} = \left\{\left(\mathrm{x}^{(j)},\ y^{(j)}\right)\right\}_{j=1}^{m}$:

$$\begin{aligned} \mathrm{L}_{\mathrm{B_{e,t}}}(\Phi) &= \frac{1}{m} \sum_{j=1}^{m} l\big(\mathrm{x}^{(\mathrm{j})}, y^{(j)}\big) + \lambda \mathrm{R}(\Phi), \Phi = (\Theta, \mathrm{w}_{\mathrm{read}}, b_{\mathrm{read}}) \\ &= \frac{1}{m} \sum_{j=1}^{m} \big[-y^{(j)} \log\big(p^{(j)}\big) - \big(1 - y^{(j)}\big) \log\big(1 - p^{(j)}\big)\big] \\ &\quad + \lambda(\|\Theta\|_2^2 + \|\mathrm{w}_{\mathrm{read}}\|_2^2 + b_{\mathrm{read}}^2) \end{aligned}$$

La idea es minimizar la perdida empírica regularizada sobre el conjunto de entrenamiento de los varios mini-batch, i.e.

$$\begin{aligned} \min_{\Phi} \mathrm{L}(\Phi) = \frac{1}{N_{tr}} \sum_{k=1}^{N_{tr}} BCE\left(y^{(k)}, \sigma\big(-\mathrm{w}^{\top} \mathrm{z}\big(\mathrm{x}^{(k)}, \Theta\big) + b\big)\right) \\ + \lambda(\|\Theta\|_2^2 + \|\mathrm{w}_{\mathrm{read}}\|_2^2 + b_{\mathrm{read}}^2) \end{aligned}$$

Lo que implica:

$$\nabla \mathrm{L}(\Phi) \approx \nabla \mathrm{L}_{\mathrm{B_t}}(\Phi) = \frac{1}{m}\sum_{j=1}^{m} \nabla l\left(\mathrm{x}^{(j)}, y^{(j)}\right) + \nabla \lambda \mathrm{R}(\Phi)$$

Gradientes para los parámetros clásicos $(\mathbf{w_{read}}, \boldsymbol{b_{read}})$

a) Derivadas internas (una muestra).

Para una muestra (quite el super índice $^{(j)}$ para aligerar la notación.

- $p = \sigma(t) = \frac{1}{1+e^{-t}},\ t = -\mathrm{w}^{\top}\mathrm{z} + b$
- $l = -y \log p - (1-y)\log(1-p)$

Derivadas:

$$\frac{\partial l}{\partial p} = -y\frac{1}{p} - (1-y)\frac{1}{1-p}(-1) = -\frac{y}{p} + \frac{1-y}{1-p}$$
$$\frac{\partial p}{\partial t} = \sigma(t)\big(1-\sigma(t)\big) = p(1-p)$$
$$\frac{\partial l}{\partial t} = \frac{\partial l}{\partial p}\frac{\partial p}{\partial t} = \left(-\frac{y}{p} + \frac{1-y}{1-p}\right)p(1-p)$$
$$= -y(1-p) + (1-y)p$$
$$= -y + yp + p - yp = p - y$$

Además,

$$\frac{\partial t}{\partial b} = 1 \qquad \Big| \qquad \frac{\partial t}{\partial \mathrm{w}} = -\mathrm{z} \in \mathbb{R}^N$$

Así las cadenas finales quedan como:

$$\mathrm{w}\text{:}\ \frac{\partial l}{\partial \mathrm{w}} = \frac{\partial l}{\partial t}\frac{\partial t}{\partial \mathrm{w}} = (p-y)(-\mathrm{z}) = -(p-y)\mathrm{z} \in \mathbb{R}^N$$

$$b\text{:}\ \frac{\partial l}{\partial b} = \frac{\partial l}{\partial t}\frac{\partial t}{\partial b} = (p-y)(1) = p - y$$

Los cuales son los resultado para una muestra sin L_2.

b) Cadena total y promedio en el mini-batch.

Recordemos que para el mini-batch definimos:

$$\mathrm{L}_{data} = \frac{1}{m}\sum_{j=1}^{m}\left[l^{(j)}, p^{(j)}\right],\ \ p^{(j)} = \sigma\left(t^{(j)}\right),\ \ t^{(j)} = -\mathrm{w}^{\top}\mathrm{z} + b$$

Las derivadas promedio (linealidad de la derivada + promedio):

$$\frac{\partial \mathrm{L}_{data}}{\partial \mathrm{w}} = \frac{1}{m}\sum_{j=1}^{m}\left[-\left(p^{(j)} - y^{(j)}\right)\mathrm{z}^{(j)}\right]$$
$$\frac{\partial \mathrm{L}_{data}}{\partial b} = \frac{1}{m}\sum_{j=1}^{m}\left[p^{(j)} - y^{(j)}\right]$$

Con regularización sobre los parámetros clásicos w, b:

$$\mathrm{R_c}(\mathrm{w_{read}}, b_{\mathrm{read}}) = \|\mathrm{w_{read}}\|_2^2 + b_{\mathrm{read}}^2 = \sum_i w_i^2 + b^2$$

Las derivadas de λR_c son:

$$\frac{\partial}{\partial \mathrm{w}}(\lambda R_c) = \lambda 2\mathrm{w}, \qquad \frac{\partial}{\partial b}(\lambda R_c) = \lambda 2b$$

Gradientes totales:

$$\frac{\partial \mathrm{L}_{data}}{\partial \mathrm{w}} = \frac{1}{m}\sum_{j=1}^{m}\left[-\left(p^{(j)} - y^{(j)}\right)\mathrm{z}^{(j)}\right] + 2\lambda \mathrm{w}$$
$$\frac{\partial \mathrm{L}_{data}}{\partial b} = \frac{1}{m}\sum_{j=1}^{m}\left[p^{(j)} - y^{(j)}\right] = 2\lambda b$$

O en su forma matricial:

$$\nabla_{\mathrm{w}}\mathrm{L}_{data} = -\frac{1}{m}\mathrm{Z}^{\top}(\mathrm{p} - \mathrm{y}) + 2\lambda \mathrm{w}$$

$$\nabla_{\mathrm{b}}\mathrm{L}_{data} = \frac{1}{m}1^{\top}(\mathrm{p} - \mathrm{y}) + 2\lambda \mathrm{b}$$

Gradientes para los parámetros cuánticos $(\mathbf{w_{read}}, \boldsymbol{b_{read}})$

a) Derivadas internas (una muestra).

$$\frac{\partial l}{\partial \Theta} = \frac{\partial l}{\partial t}\frac{\partial t}{\partial \Theta} = (p-y)\frac{\partial t}{\partial \Theta}(-\mathrm{w}^{\top}\mathrm{z}) = (p-y)\left(-\mathrm{w}^{\top}\frac{\partial z}{\partial \Theta}\right)$$

b) Cadena total y promedio en el mini-batch.

$$\frac{\partial \mathrm{L}_B}{\partial \Theta} = \frac{1}{m}\sum_{j=1}^{m}\left[\left(p^{(j)} - y^{(j)}\right)\left(-\mathrm{w}^{\top}\frac{\partial z^{(j)}}{\partial \Theta}\right)\right] + 2\lambda\theta$$

c) Regla parameter-shift para R_Y y R_Z.

Sea un ángulo θ (por ejemplo $\alpha_{\ell,i}$ o $\beta_{\ell,i}$) que aparece en una puerta de la forma: $U(\theta) = e^{-i\theta G}$, con G un generadpr de espectro $\pm\frac{1}{2}$. Donde justamente $R_Y(\theta) = e^{-i\theta Y/2}$ y $R_Z(\theta) = e^{-i\theta Z/2}$ valen y se satisface:

$$\frac{\partial}{\partial \theta}\langle O\rangle_\theta = \frac{1}{2}\left(\langle O\rangle_{\theta+\frac{\pi}{2}} - \langle O\rangle_{\theta-\frac{\pi}{2}}\right)$$

Donde, $\langle O\rangle_\theta = \langle\psi(\theta)|O|\psi(\theta)\rangle$ es el valor esperado del observable O, en nuestro caso O son los Z_i de salida y $|\psi(\theta)\rangle$ es el estado resultante del circuito con ese θ.

d) Aplicando a nuestro vector z.

Cada componente $z_i = \langle Z_i\rangle$ cumple, para $\forall\ \theta \in \Theta$:

$$\frac{\partial z_i}{\partial \theta} = \frac{1}{2}\left(z_i^{(j)}\Big|_{\theta+\frac{\pi}{2}} - z_i^{(j)}\Big|_{\theta-\frac{\pi}{2}}\right)$$

Donde $z_i^{(j)}\Big|_{\theta\pm\frac{\pi}{2}}$ es la misma lectura $\langle Z_i\rangle$ pero calculada con ese único parámetro corridos $\theta \pm \frac{\pi}{2}$ (el resto de los parámetros queda igual).

Sustituyendo en la ecu. anterior:

$$\frac{\partial L_B}{\partial \Theta} = \frac{1}{m}\sum_{j=1}^{m}\left(p^{(j)} - y^{(j)}\right)\left(-\sum_{j=1}^{15} w_i \frac{1}{2}\left(z_i^{(j)}\Big|_{\theta+\frac{\pi}{2}} - z_i^{(j)}\Big|_{\theta-\frac{\pi}{2}}\right)\right) + 2\lambda\theta$$

8) Evaluación y Resultados.

8.1. Accuracy final.

Los accuracy para el conjunto de entrenamiento y de prueba final, fueron:

- **Train**: 0.972 → 97.2% de aciertos sobre el conjunto de entrenamiento.
- **Test**: 0.986 → 98.6% de aciertos sobre el conjunto de prueba.

8.2. Matriz de confusión. (test)

La matriz de confusión del conjunto de prueba quedo como:

	Pred (0)	Pred (1)
True (0)	35	1
True (1)	0	36

Donde podemos observar:

- Verdaderos negativos (TN) = 35
- Falsos positivos (FP) = 1 (clase 0 mal predicha como 1)
- Falsos negativos (FN) = 0 (clase 1 mal predicha como 0)
- Verdaderos positivos (TP)=36

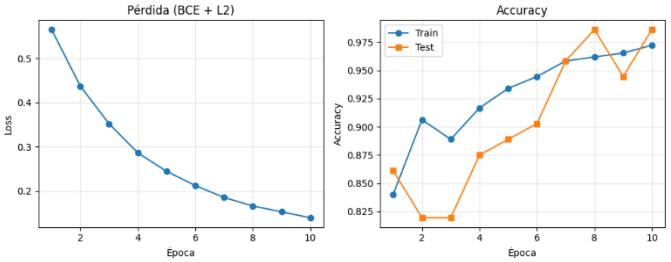
8.3. Reporte de clasificación.

	precision	recall	f1-score	support
0	1.000	0.972	0.986	36
1	0.973	1.000	0.986	36
accuracy			0.986	72
macro avg	0.986	0.986	0.986	72
weighted avg	0.986	0.986	0.986	72

Como hay clases balanceadas (36/36), los promedios macro y ponderado coinciden (0.986).

8.4. Curvas de entrenamiento.

En las siguientes imágenes podemos observar el comportamiento de nuestra función de perdida y del accuracy:



- Pérdida (BCE+L2): desciende de forma monótona y suave → entrenamiento estable, sin señales de divergencia.
- Accuracy train/test: ambas suben y permanecen muy cercanas; la brecha de generalización es mínima → no se observa sobreajuste.

Los pequeños dientes de sierra en el test son normales, pues podemos ver claramente la tendencia ascendente.

9) Anexo

9.1. Gradiente de un Observable (función de costo)

En un algoritmo cuántico variacional (VQA) elegimos un ansatz $U(\vec{\theta})$ que actúa sobre un estado de referencia $|0\rangle$. Para evaluar qué tan "buena" es la elección de $\vec{\theta}$, definimos una función de costo como el valor esperado de algún observable $\hat{M}$:

$$C(\vec{\theta}) = \langle 0|U^{\dagger}(\vec{\theta})\hat{M}U(\vec{\theta})|0\rangle$$

- $|0\rangle$: estado de referencia (por ejemplo, todos los qubits en $|0\rangle$). Es el punto de partida del hardware.
- $U(\vec{\theta})$: ansatz donde se llevan a cabo las rotaciones y compuertas, prepara el estado:

$$|\psi(\vec{\theta})\rangle = U(\vec{\theta})|0\rangle$$

Recordando que interpretamos el valor esperado como el promedio de una variable aleatoria.

Pues de este modo: $C(\vec{\theta}) = \langle\psi(\vec{\theta})|\hat{M}|\psi(\vec{\theta})\rangle = \langle\hat{M}\rangle_{\vec{\theta}}$, lo cual en cuántica es medir el valor esperado de un observable de $\hat{M}$. i.e. Medir muchas veces $\hat{M}$ tras preparar $U(\vec{\theta})|0\rangle$, la media de los resultados (ponderada por sus probabilidades,) tendera a $C(\vec{\theta})$.

Para optimizar la función de costo:

$$C(\vec{\theta}) = \langle 0|U^{\dagger}(\vec{\theta})\hat{M}U(\vec{\theta})|0\rangle$$

Buscamos minimizarlo con: $\vec{\theta}_{opt} = \arg\min_{\vec{\theta}} C(\vec{\theta})$

Podemos realizar lo mismo que se hacen en el descenso gradiente clásico, que actualiza los parámetros como:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla C\left(\theta^{(t)}\right)$$

Supongamos que nuestra función de coste es: $f(\theta) = \langle 0|U^\dagger(\theta)\hat{B}U(\theta)|0\rangle$, donde $\hat{B}$ es el observable que medimos.

Para obtener $\nabla f(\theta)$ hay que aplicar regla del producto:

$$\begin{aligned}\nabla f(\theta) &= \nabla\langle 0|U^\dagger(\theta)\hat{B}U(\theta)|0\rangle \\ &= \langle 0|\nabla U^\dagger(\theta)\hat{B}U(\theta)|0\rangle + \langle 0|U^\dagger(\theta)\hat{B}\nabla U(\theta)|0\rangle \\ &= \left\langle 0\middle|\left(\frac{dU^\dagger(\theta)}{d\theta}\right)\hat{B}U(\theta)\middle|0\right\rangle + \left\langle 0\middle|U^\dagger(\theta)\hat{B}\left(\frac{dU(\theta)}{d\theta}\right)\middle|0\right\rangle\end{aligned}$$

Para poder desarrollar más debemos conocer nuestro ansatz $U(\theta)$, para ejemplificar propongamos un caso de un solo qubit: $U(\theta) = e^{-\frac{i}{2}\theta\hat{P}_j}$

Además, por lo general para una puerta unitaria suele escribirse como la exponencial de un generador Hermítico: $U(\theta) = e^{-i\theta G/2}$, con G un operador Hermítico. Pero en particular para este caso como trabajamos con un solo qubit, las puertas de rotación son los generadores de Pauli: $(\sigma_x, \sigma_y, \sigma_z)$. $\sigma_x = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$, $\sigma_x = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$, $\sigma_z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$

Regresando, tenemos:

$$\frac{dU(\theta)}{d\theta} = -\frac{i}{2}\hat{P}_j U(\theta), \qquad \frac{dU^\dagger(\theta)}{d\theta} = \frac{i}{2}U^\dagger(\theta)\hat{P}_j$$

Sustituyendo,

$$\begin{aligned}\nabla f(\theta) &= \left\langle 0\middle|\left(\tfrac{i}{2}U^\dagger(\theta)\hat{P}\right)\hat{B}U(\theta)\middle|0\right\rangle \\ &\qquad + \left\langle 0\middle|U^\dagger(\theta)\hat{B}\left(-\tfrac{i}{2}\hat{P}U(\theta)\right)\middle|0\right\rangle \\ &= \frac{i}{2}\langle 0|U^\dagger(\theta)\hat{P}\hat{B}U(\theta)|0\rangle - \frac{i}{2}\langle 0|U^\dagger(\theta)\hat{B}\hat{P}U(\theta)|0\rangle \\ &= \frac{i}{2}\langle 0|U^\dagger(\theta)\left(\hat{P}\hat{B} - \hat{B}\hat{P}\right)U(\theta)|0\rangle\end{aligned}$$

En mecánica cuántica y álgebra lineal, el conmutador entre dos operadores $\hat{A}$ y $\hat{B}$ se define como: $[\hat{A}, \hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A}$

Por ende: $\nabla f(\theta) = \frac{i}{2}\langle 0|U^\dagger(\theta)[\hat{P}, \hat{B}]U(\theta)|0\rangle$

Ahora, haciendo uso de la siguiente identidad matemática para conmutadores que involucran operadores de Pauli:

$$[\hat{P}_j, \hat{B}] = -i\left(U^\dagger\left(\frac{\pi}{2}\right)\hat{B}U\left(\frac{\pi}{2}\right) - U^\dagger\left(-\frac{\pi}{2}\right)\hat{B}U\left(-\frac{\pi}{2}\right)\right)$$

Ahora sustituyendo en la expresión anterior obtenemos el gradiente de: $f(\theta)$

$$\begin{aligned}&\nabla f(\theta) = \tfrac{i}{2}\langle 0|U^\dagger(\theta)[\hat{P}, \hat{B}]U(\theta)|0\rangle \\ &= \frac{i}{2}\left\langle 0\middle|U^\dagger(\theta)\left(-i\left(U^\dagger\left(\frac{\pi}{2}\right)\hat{B}U\left(\frac{\pi}{2}\right) - U^\dagger\left(-\frac{\pi}{2}\right)\hat{B}U\left(-\frac{\pi}{2}\right)\right)\right)U(\theta)\middle|0\right\rangle \\ &= -\frac{i^2}{2}\left\langle 0\middle|U^\dagger(\theta)\left(U^\dagger\left(\frac{\pi}{2}\right)\hat{B}U\left(\frac{\pi}{2}\right) - U^\dagger\left(-\frac{\pi}{2}\right)\hat{B}U\left(-\frac{\pi}{2}\right)\right)U(\theta)\middle|0\right\rangle \\ &= \frac{1}{2}\left\langle 0\middle|U^\dagger(\theta)U^\dagger\left(\frac{\pi}{2}\right)\hat{B}U\left(\frac{\pi}{2}\right)U(\theta)\middle|0\right\rangle \\ &\qquad - \frac{1}{2}\left\langle 0\middle|U^\dagger(\theta)U^\dagger\left(-\frac{\pi}{2}\right)\hat{B}U\left(-\frac{\pi}{2}\right)U(\theta)\middle|0\right\rangle \\ &= \frac{1}{2}\left\langle 0\middle|U^\dagger\left(\theta+\frac{\pi}{2}\right)\hat{B}U\left(\theta+\frac{\pi}{2}\right)\middle|0\right\rangle \\ &\qquad - \frac{1}{2}\left\langle 0\middle|U^\dagger\left(\theta-\frac{\pi}{2}\right)\hat{B}U\left(\theta-\frac{\pi}{2}\right)\middle|0\right\rangle\end{aligned}$$

Recordemos que: $f(\theta) = \langle 0|U^\dagger(\theta)\hat{B}U(\theta)|0\rangle$

Por lo tanto, $\nabla f(\theta) = \frac{1}{2}\left(f\left(\theta+\frac{\pi}{2}\right) - f\left(\theta-\frac{\pi}{2}\right)\right)$

Lo cual es la Parameter-Shift Rule (PSR) en su forma más usada. La ventaja de PSR es que permite usar el optimizador de Descenso de Gradiente para circuitos variacionales.

9.2. Demostraciones

Demostración 1.

$$z_i(\mathrm{x}, \Theta) = \langle\psi(\mathrm{x}, \Theta)|Z_i|\psi(\mathrm{x}, \Theta)\rangle = \mathrm{Tr}(\rho(\mathrm{x}, \Theta)Z_i)$$

Sea $\{|e_k\rangle\}_{k=1}^d$ una base ortonormal de $\mathcal{H}$ (con $d = 2^N$ para N qubits). Por definición de traza:

$$\mathrm{Tr}(\rho A) = \sum_{k=1}^{d}\langle e_k|\rho A|e_k\rangle$$

Para $\rho = |\psi\rangle\langle\psi|$,

$$\mathrm{Tr}(\rho A) = \sum_k \langle e_k||\psi\rangle\langle\psi|A|e_k\rangle$$

Reordenando los términos y usando la resolución de la identidad $\sum_k |e_k\rangle\langle e_k| = I$:

$$\begin{aligned}\mathrm{Tr}(\rho A) &= \sum_k \langle\psi|A|e_k\rangle\langle e_k|\psi\rangle = \langle\psi|A(\textstyle\sum_k|e_k\rangle\langle e_k|)|\psi\rangle \\ &= \langle\psi|A|\psi\rangle\end{aligned}$$

Como la base es arbitraria, el resultado es base-independiente. Tomando $A = Z_i$: $\mathrm{Tr}(\rho Z_i) = \langle\psi|Z_i|\psi\rangle$. ■

Demostración 2.

Identidad de la "Regla de Desplazamiento de Parámetros (PSR)"

Sea $U(\phi) = e^{-\frac{i}{2}\phi\hat{P}_j}$ una puerta parametrizada,

Donde $\hat{P}_j$ es un operador de Pauli hermítico y unitario por construcción, lo cual nos indica que: $\hat{P}_j^2 = \hat{I}$ y $\hat{P}_j^\dagger = \hat{P}_j$.

Partamos de la serie de potencias de la exponencial cerrada:

$$e^{\pm ia\hat{P}_j} = \cos a\,\hat{I} \pm i\sin a\,\hat{P}_j$$

Aplicado a nuestro operador:

$$U(\phi) = e^{-\frac{i}{2}\phi\hat{P}_j} = \cos\frac{\phi}{2}\hat{I} - i\sin\frac{\phi}{2}\hat{P}_j$$
$$U^{\dagger}(\phi) = e^{\frac{i}{2}\phi\hat{P}_j} = \cos\frac{\phi}{2}\hat{I} + i\sin\frac{\phi}{2}\hat{P}_j$$

Recordemos que nos interesa saber quién es la siguiente expresión:

$$-i\left(U^{\dagger}\left(\frac{\pi}{2}\right)\hat{B}U\left(\frac{\pi}{2}\right) - U^{\dagger}\left(-\frac{\pi}{2}\right)\hat{B}U\left(-\frac{\pi}{2}\right)\right)$$

Consideremos a $\hat{B}$ como cualquier operador sin ser necesariamente de Pauli.

Vamos a trabajar primero con: $U^{\dagger}(\phi)\hat{B}U(\phi)$:

$$U^{\dagger}(\phi)\hat{B}U(\phi) = \left[\cos\frac{\phi}{2}\hat{I} + i\sin\frac{\phi}{2}\hat{P}_j\right]\hat{B}\left[\cos\frac{\phi}{2}\hat{I} - i\sin\frac{\phi}{2}\hat{P}_j\right]$$
$$= \left[\cos\frac{\phi}{2}\hat{B} + i\sin\frac{\phi}{2}\hat{P}_j\hat{B}\right]\left[\cos\frac{\phi}{2}\hat{I} - i\sin\frac{\phi}{2}\hat{P}_j\right]$$
$$= \cos\frac{\phi}{2}\hat{B}\cos\frac{\phi}{2}\hat{I} - \cos\frac{\phi}{2}\hat{B}i\sin\frac{\phi}{2}\hat{P}_j + i\sin\frac{\phi}{2}\hat{P}_j\hat{B}\cos\frac{\phi}{2} - i\sin\frac{\phi}{2}\hat{P}_j\hat{B}i\sin\frac{\phi}{2}\hat{P}_j$$
$$= \cos^2\frac{\phi}{2}\hat{B} - i\cos\frac{\phi}{2}\sin\frac{\phi}{2}\hat{B}\hat{P}_j + i\cos\frac{\phi}{2}\sin\frac{\phi}{2}\hat{P}_j\hat{B} + \sin^2\frac{\phi}{2}\hat{P}_j\hat{B}i\hat{P}_j$$
$$= \cos^2\frac{\phi}{2}\hat{B} + \sin^2\frac{\phi}{2}\hat{P}_j\hat{B}i\hat{P}_j + i\cos\frac{\phi}{2}\sin\frac{\phi}{2}\left(\hat{P}_j\hat{B} - \hat{B}\hat{P}_j\right)$$
$$= \cos^2\frac{\phi}{2}\hat{B} + \sin^2\frac{\phi}{2}\hat{P}_j\hat{B}i\hat{P}_j + i\cos\frac{\phi}{2}\sin\frac{\phi}{2}\left[\hat{P}_j, \hat{B}\right]$$

Evaluemos en $\phi = \frac{\pi}{2}$:

$$U^{\dagger}\left(\frac{\pi}{2}\right)\hat{B}U\left(\frac{\pi}{2}\right) = \cos^2\frac{\pi}{4}\hat{B} + \sin^2\frac{\pi}{4}\hat{P}_j\hat{B}i\hat{P}_j + i\cos\frac{\pi}{4}\sin\frac{\pi}{4}\left[\hat{P}_j, \hat{B}\right]$$
$$= \left(\frac{\sqrt{2}}{2}\right)^2\hat{B} + \left(\frac{\sqrt{2}}{2}\right)^2\hat{P}_j\hat{B}i\hat{P}_j + i\left(\frac{\sqrt{2}}{2}\right)^2\left[\hat{P}_j, \hat{B}\right]$$
$$= \frac{1}{2}\hat{B} + \frac{1}{2}\hat{P}_j\hat{B}i\hat{P}_j + i\frac{1}{2}\left[\hat{P}_j, \hat{B}\right]$$

Evaluemos en $\phi = -\frac{\pi}{2}$:

$$U^{\dagger}\left(-\frac{\pi}{2}\right)\hat{B}U\left(-\frac{\pi}{2}\right)$$
$$= \cos^2-\frac{\pi}{4}\hat{B} + \sin^2-\frac{\pi}{4}\hat{P}_j\hat{B}i\hat{P}_j + i\cos-\frac{\pi}{4}\sin-\frac{\pi}{4}\left[\hat{P}_j, \hat{B}\right]$$
$$= \left(\frac{\sqrt{2}}{2}\right)^2\hat{B} + \left(-\frac{\sqrt{2}}{2}\right)^2\hat{P}_j\hat{B}i\hat{P}_j + i\left(\frac{\sqrt{2}}{2}\right)\left(-\frac{\sqrt{2}}{2}\right)\left[\hat{P}_j, \hat{B}\right]$$
$$= \frac{1}{2}\hat{B} + \frac{1}{2}\hat{P}_j\hat{B}i\hat{P}_j - i\frac{1}{2}\left[\hat{P}_j, \hat{B}\right]$$

Sustituyamos:

$$U^{\dagger}\left(\frac{\pi}{2}\right)\hat{B}U\left(\frac{\pi}{2}\right) - U^{\dagger}\left(-\frac{\pi}{2}\right)\hat{B}U\left(-\frac{\pi}{2}\right)$$
$$= \frac{1}{2}\hat{B} + \frac{1}{2}\hat{P}_j\hat{B}i\hat{P}_j + i\frac{1}{2}\left[\hat{P}_j, \hat{B}\right] - \frac{1}{2}\hat{B} - \frac{1}{2}\hat{P}_j\hat{B}i\hat{P}_j + i\frac{1}{2}\left[\hat{P}_j, \hat{B}\right]$$
$$= i\frac{1}{2}\left[\hat{P}_j, \hat{B}\right] + i\frac{1}{2}\left[\hat{P}_j, \hat{B}\right] = i\left[\hat{P}_j, \hat{B}\right]$$

Por lo tanto,

$$-i\left(U^{\dagger}\left(\frac{\pi}{2}\right)\hat{B}U\left(\frac{\pi}{2}\right) - U^{\dagger}\left(-\frac{\pi}{2}\right)\hat{B}U\left(-\frac{\pi}{2}\right)\right)$$
$$= -i\left(i\left[\hat{P}_j, \hat{B}\right]\right) = -i^2\left[\hat{P}_j, \hat{B}\right] = -(-1)\left[\hat{P}_j, \hat{B}\right] = \left[\hat{P}_j, \hat{B}\right] \quad \blacksquare$$

Demostración 3.

$$p = \frac{1}{1 + e^{-t}}$$
$$\frac{dp}{dt} = \frac{e^{-t}}{(1 + e^{-t})^2} = \frac{1}{1 + e^{-t}}\left(1 - \frac{1}{1 + e^{-t}}\right) = p(1 - p) \quad \blacksquare$$

9.3. Algunas Variables:

$\boldsymbol{X} \in \mathbb{R}^{\mathbf{8\times 8}}$: la imagen original (matriz de 8 filas × 8 columnas de intensidades).

$\boldsymbol{Y} \in \mathbb{R}^{\mathbf{4\times 4}}$: la imagen reducida (resultado de bajar resolución).

$\boldsymbol{\mathcal{R}} \in \mathbb{R}^{\mathbf{8\times 8}}$: el operador de reducción (la función que convierte X en Y).

$\boldsymbol{i}, \boldsymbol{j} \in \{\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}\}$: índices de fila y columna en la imagen reducida Y.

$\boldsymbol{u}, \boldsymbol{v}$: índices en X que caen dentro del bloque 2×2 que corresponde a la celda (i, j) de Y.

$\boldsymbol{S} \in \mathbb{R}^{\mathbf{4\times 8}}$: matriz que promedia pares adyacentes. Hace "mitad y mitad" entre filas (o columnas) contiguas.

$\mathbf{vec}(\cdot)$: vectoriza una matriz (la apila en un vector).

$\otimes$: producto de Kronecker (construye matrices bloque para representar operaciones sobre filas y columnas a la vez).

$\boldsymbol{\chi} = \boldsymbol{D_{01}} = \{\boldsymbol{X^{(k)}}\}_{\boldsymbol{k=1}}^{\boldsymbol{N_{01}}}$: son todas las imágenes del subconjunto binario (clases 0 y 1).

$\boldsymbol{K} = \frac{\mathbf{1}}{\mathbf{4}}\begin{bmatrix}\mathbf{1} & \mathbf{1}\\ \mathbf{1} & \mathbf{1}\end{bmatrix}$: filtro promediado (kernel de suavizado).

$\hat{\mathbf{1}}$: una matriz de unos del mismo tamaño que $X^{(k)}$.

$\boldsymbol{\rho} \in (\mathbf{0}, \mathbf{1})$: fracción destinada a prueba (ρ=0.2 → 20% test).

$\boldsymbol{N_{tr}}$ y $\boldsymbol{N_{01}}$: tamaño de entrenamiento y número total de ejemplos 0/1.

$\boldsymbol{\mathcal{T}_{tr}}$ y $\boldsymbol{\mathcal{T}_{te}}$: particiones de train y test.

9.4. Circuito cuántico completo.