Introducción a Machine Learning Cuántico

Máquina de Vectores de Soporte - SVM

Camilo Andrés Torres Torres

Director

Luis Daniel Benavides Navarro

Tesis presentada para adquirir el título de
Ingeniero de Sistemas

Ingeniería de Sistemas
Escuela Colombiana de Ingeniería Julio Garavito
Colombia
Enero 2018

Contenido

Resumen

En este documento se explorará cómo algunos algoritmos de aprendizaje de máquina pueden ser mejorados, cuadrática y exponencialmente, gracias a la computación cuántica. Se hace una breve introducción al mundo de los datos y por qué es importante realizar un procesamiento adecuado de los mismos, y dando las bases tanto del aprendizaje de máquina como de la computación cuántica, se describirá y se mostrará la implementación en un simulador cuántico, de un algoritmo de maquina de vectores de soporte (SVM), capaz de reconocer y clasificar imágenes de los dígitos 6 y 9, desarrollado e implementado físicamente por Zhaokai Li, Xiaomei Liu, Nanyang Xu, Jiangfeng Du, en la Universidad de Ciencia y Tecnología de China, Hefei, demostrando cómo se puede tener una mejora exponencial en complejidad temporal con respecto a su contraparte clásica.

1 Introducción

El hombre a lo largo de su historia ha sido capaz de entender, adaptar y manipular el ambiente que lo rodea para su beneficio. Es práctico afirmar que desde su concepción, el hombre ha sido capaz de extraer y utilizar los datos sensoriales para avanzar, evolucionar y construir la sociedad moderna. Actualmente, dichos datos son generados continuamente a través de medios digitales. Los datos siguen cumpliendo su papel como bloque fundamental de la civilización, y conforme aumenta el volumen de los mismos de manera exponencial, se espera que los métodos de recopilación y análisis se vuelvan más eficientes y precisos.

¿Qué hacer con tantos datos? Es bastante claro ya, que actualmente los métodos convencionales de análisis de datos no son útiles, los medios físicos ya no son suficientes, y ahí es donde la palabra digital juega un rol primordial. Afortunadamente, ha surgido el concepto de Inteligencia Artificial (AI), el cual explora el comportamiento de agentes inteligentes. En la ciencia de la computación, con fines prácticos, se define inteligencia como la capacidad de dicho agente de entender su entorno para maximizar su probabilidad de éxito en una tarea u objetivo predefinido.

Cabe resaltar que el concepto de AI es uno multidisciplinario, pues surge una gran variedad de preguntas al respecto, en temas de filosofía, biología, así como en el de nuestro actual interes, tecnología, entre otras areas del conocimiento. Entrando en detalle en éste último, surge un subconcepto enfocado al aprendizaje como tal de las máquinas. Como lo indica su nombre, el Aprendizaje de Máquinas (Machine Learning) consiste en definir modelos estadísticos capaces de "aprender", a realizar una tarea específica, sin necesidad de describir explícitamente cómo hacerlo. He ahí la importancia de los datos en un medio digital.

Antes de introducir la utilidad de la computación cuántica en éste concepto, es necesario entender que actualmente la industria emplea éstos sistemas, con el fin de automatizar sus procesos con una intervención humana cada vez menor. Por esto, las decisiones que toman dichos sistemas inteligentes debe ser lo suficientemente precisas y consolidadas sobre conjuntos masivos de datos, tan grandes como sea necesario para realizar el aprendizaje, o entrenamiento.

La magia de la computación cuántica radica en que éste nuevo modelo computacional permite rediseñar los algoritmos ya usados en la computación clásica convencional, con tiempos de ejecución polinomial, e incluso exponencialmente, más rápidos. Naturalmente, también se explora la creación de nuevos algoritmos que puedan aprovechar al máximo las propiedades de la física cuántica en el entorno de la computación. Se espera que el lector tenga un conocimiento básico sobre cómo funciona un sistema cuántico.

En éste artículo se mostrará cómo haciendo uso de la computación cuántica se puede mejorar exponencialmente el tiempo de ejecución de algunos los algoritmos empleados para Machine Learning, recopilando diferentes métodos, y dando una base sólida al lector para profundizar en cada uno de ellos. Específicamente, se mostrará en detalle la implementación del algoritmo de máquinas de vectores de soporte (SVM), adaptado a un computador cuántico.

2 Marco Teórico

2.1 Machine Learning

Arthur Lee Samuel, científico de la computación pionero en el campo de la inteligencia artificial, adoptó el término de aprendizaje de máquina (Machine Learning), como el campo de estudio enfocado en dar a los computadores la habilidad de aprender sin ser explícitamente programados para hacerlo.

Para entender mejor esta definición, se definen dos tipos de conocimiento: declarativo, e imperativo. El primero, se refiere al almacenamiento de información previamente evaluada, la cual se consulta posteriormente, con la esperanza de ya se tenga de antemano la solución esperada, como ocurre con una base de datos, o cualquier estructura llave - valor. Sin embargo, el conocimiento imperativo está más asociado a la capacidad de inferir y deducir el conocimiento a partir de los datos, con el fin de obtener modelos capaces de evaluar datos similares de los cuales no se tenía información previa, como ocurre con modelos de regresión y otros modelos estadísticos. Éste último es el campo de acción del aprendizaje de máquina.

En comparación, el paradigma de programación tradicional, como se ve en el

diagrama, espera que el computador reciba un programa F y un conjunto de datos X, de manera que el resultado de la computación sea de la forma $F : X \mapsto Y$, donde Y es el espacio de salida (output). Bajo la misma nomenclatura, en un paradigma de aprendizaje de máquina, se espera que el computador reciba unos datos de entrenamiento X', con su respectivo output Y, con el objetivo de definir un programa F, que se comporte de la manera inicialmente descrita.

Entrando más en el tema, el objetivo fundamental del aprendizaje de máquina es evaluar y reconocer patrones intrínsecos a partir de los datos de entrada, o de entrenamiento, mediante herramientas de análisis y estadística para construir un modelo capaz de adaptarse y crecer, que se ajuste de la mejor manera posible al problema y le permita inferir la manera de realizar predicciones acertadas, idealmente, sobre nuevos datos de entrada y obtener resultados satisfactorios para resolver un problema en cuestión.

A continuación, los pasos generales de un algoritmo de aprendizaje de máquina

- Definición del modelo, donde se define qué se quiere resolver y mediante qué atributos y características de los datos se espera definir el modelo.
- Preprocesamiento de los datos, donde se estandarizan los datos de entrada, probablemente de múltiples fuentes y formatos (imágenes, texto, video, etcétera), para facilitar la extracción de los atributos necesarios.
- Extracción de características (features), las cuales funcionan como datos de entrada para alimentar el modelo de entrenamiento.
- Probar y entrenar el modelo, con las características ya extraídas, donde se entrena el modelo y se ajustan ciertos parámetros para optimizar y mejorar la calidad del resultado.
- Predicción, cuando ya se ha entrenado el modelo de manera satisfactoria y es posible procesar datos nunca evaluados, con las mismas características con las que se define el modelo, para obtener un resultado acertado según lo requiere el problema.

Existen dos tipos de problemas fundamentales que el paradigma de aprendizaje de máquina puede modelar: Clasificación, problema en el cual existen datos de entrenamiento previos y se conoce a qué clase pertenece cada uno de ellos. El objetivo consiste en utilizar dichos datos para entrenar la máquina y construir un modelo capaz de clasificar nuevos datos en las clases previamente definidas.

El otro problema, clusterización, consiste en el agrupamiento natural de los datos, a partir de patrones intrínsecos en los datos, y su nivel de correlación, de forma que el modelo pueda agrupar los datos que según criterios establecidos considera cercanos. Para atacar cada uno de estos problemas, se definieron dos paradigmas asociados al aprendizaje de máquina, aprendizaje supervisado

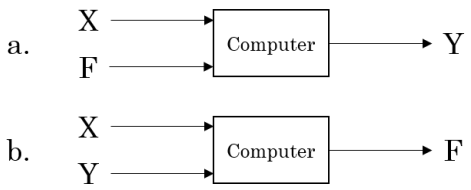

Figure 1: a. Paradigmama Tradicional. b. Paradigma con Machine Learning. Teniendo en cuenta $F : X \mapsto Y$

(clasificación) y aprendizaje no supervisado (clusterización). La gran diferencia entre ambos paradigmas, es la existencia de información previamente clasificada.

2.2 Aprendizaje Supervisado

El aprendizaje supervisado, como se mencionaba, facilita las tareas de clasificación cuando se tiene de antemano un conjunto de datos previamente clasificado, lo suficientemente robusto para generar un modelo. Formalmente, dado un conjunto de N vectores de características de la forma $(\vec{x_1}, y_1), ..., (\vec{x_N}, y_N)$ tal que y_i corresponde a la etiqueta (de clasificación) del vector $\vec{x_i}$, el enfoque de aprendizaje supervisado busca una función g, tal que $g : X \to Y$, donde X es el espacio de entrada y Y el espacio de salida. Diferentes técnicas existen para asegurarse de que la clasificación dada por g sea la correcta, como la ingeniería de características, un buen preprocesamiento, y diferentes algoritmos que ayudan a entrenar el modelo eficientemente.

Algunos algoritmos dentro de ésta categoría son, siendo los más usados comercialmente, las redes neuronales artificiales, árboles de decisión, inferencia bayesiana, y las máquinas de vectores de soporte, siendo éste último el enfoque de éste artículo.

2.2.1 SVM

Las máquinas de vectores de soporte son un algoritmo de clasificación binario. Formalmente, dado un conjunto de datos de entrenamiento X de tamaño M, tal que $X \subset \mathbb{R}^n$, cada elemento $\vec{x_i} \in X$, posee una etiqueta y, de forma que $y(\vec{x_i}) = \pm 1$, donde los elementos de $\vec{x_i}$ son sus características (atributos o features), y la etiqueta corresponde a la clase a la que pertenece. Teniendo esto en cuenta, se puede considerar cada elemento como una pareja $(\vec{x_i}, y_i), ..., (\vec{x_n}, y_n)$. Se define el espacio, preferiblemente, con las bases canónicas de $\mathbb{R}^n$, con los elementos de

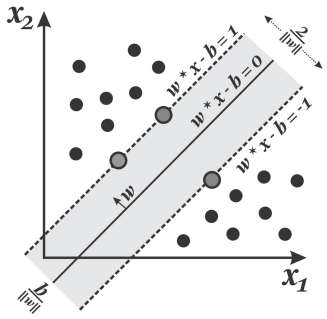

Figure 2: SVM. Los puntos en la linea punteada corresponden a los vectores de soporte

X. El objetivo del algoritmo es crear un hiperplano, definido por la expresión

$$\vec{w} \cdot \vec{x} - b = 0$$

Donde $\vec{w}$ es un vector normal con respecto al hiperplano y el parámetro $\frac{b}{||\vec{w}||}$ corresponde al ajuste (offset) con respecto al origen proyectado en $\vec{w}$. Adicionalmente, el hiperplano anterior se deriva de la construcción de dos hiperplanos de la siguiente forma

$$\vec{w} \cdot \vec{x} - b = 1 \quad \text{Para los vectores } \vec{x_i} \text{ en o sobre este límite}$$
$$\vec{w} \cdot \vec{x} - b = -1 \quad \text{Para los vectores } \vec{x_i} \text{ en o bajo este límite}$$

Los tres hiperplanos se ven de la siguiente manera [Figura 2], cada vector etiqutado correctamente, donde X tiene la propiedad de ser linealmente separables, es decir, que dentro del espacio existe un hiperplano capaz de dividir las dos clases (1 y -1). El objetivo del algoritmo SVM es encontrar el plano más efectivo en realizar dicha tarea.

Note que dentro de los límites definidos por los hiperplanos, no existe ningun vector $\vec{x_i}$, pero si existen en la traza misma del hiperplano. Esto se cumple debido a que se deben satisfacer las siguientes restricciones para todo i.

$$\vec{w} \cdot \vec{x_i} - b \geq 1, \quad \text{if} \quad y_i = 1$$
$$\vec{w} \cdot \vec{x_i} - b \leq 1, \quad \text{if} \quad y_i = -1$$

Aquellas condiciones pueden ser reescritas de la siguiente forma, haciendo uso de la etiqueta y

$$y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1$$

Los vectores sobre los cuales se definen los hiperplanos extremos se denominan vectores de soporte. La distancia entre los hiperplanos se denomina margen y se define con la expresión $\frac{2}{||\vec{w}||}$. Se desea maximizar la distancia entre estos hiperplanos para así poder encontrar el hiperplano optimo descrito por la expresión inicial. De esta forma, se busca minimizar $||\vec{w}||$ teniendo en cuenta $y_i(\vec{w} \cdot \vec{x_i} - b) \geq 1$ para todo $1 \leq i \leq n$. La solución $||\vec{w}||$ y b relacionadas a este problema determinan el clasificador $\vec{x} \mapsto \boldsymbol{sgn}(\vec{w} \cdot \vec{x} - b)$.

El planteamiento original para resolver dicho problema de optimización, implica el uso de lagrangianos. Es decir, derivadas parciales igualadas a 0 (optimización) teniendo en cuenta restricciones. El cómo se resuelven dichos operadores lagrangianos va más allá del alcance de éste artículo, sin embargo, se conoce el resultado y la expresión que se debe resolver mediante programación cuadrática. Dicho problema, dado que el conjunto de datos es linealmente separable, es convexo y converge a un minimo global.

$$\max f(\alpha_i, ..., \alpha_n) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i \alpha_i (x_i \cdot x_j) y_j \alpha_j$$

$$\text{considerando } \sum_{i=0}^{n}, \text{y } 0 \leq \alpha_i \leq \frac{1}{2n\gamma} \text{ para todo } i$$

Donde α_i es el peso asociado a cada vector $\vec{x_i}$ y γ es un parametro ajustable, definido por el programador, de tal manera que

$$\vec{w} = \sum_{i=1}^{n} \alpha_i y_i \vec{x_i}$$

de forma que $\alpha_i = 0$ si $\vec{x_i}$ se encuentra en el lado correspondiente a su clasificación según el margen, y $0 < \alpha_i < 2n\gamma^{-1}$ para los vectores de sporte. Es decir, $\vec{w}$ se describe como una combinación lineal de los vectores de soporte.

El siguiente paso es hallar b. Una vez se obtiene $\vec{w}$, se toma la expresión definida para restringir el margen y se obtiene que

$$y_i(\vec{w} \cdot \vec{x_i} - b) = 1 \iff b = \vec{w} \cdot \vec{x_i} - y$$

2.3 Computación Cuántica

Los modelos de machine learning se basan en la ejecución secuencial de ciertos algorítmos, o subrutinas, los cuales suelen implementarse de forma modular, facilitando su uso en diferentes modelos de aprendizaje. Por esta razón, se buscan constantemente maneras de optimizar dichas subrutinas, ya sea algoritmicamente o a nivel de hardware, y el múndo cuántico no es la excepción. Es necesario mostrar, a un nivel general, cómo funcionan las contrapartes cuánticas de dichas subrutinas, las cuales son empleadas por varios algoritmos discutidos más adelante. Para empezar, algunos conceptos fundamentales sobre la computación cuántica.

2.4 Conceptos

Se asume que el lector tiene conocimiento previo sobre estados cuánticos y números complejos.

2.4.1 Matriz de densidad

Antes, es necesario introducir el concepto de matriz de densidad. Esta es una herramienta que nos va a permitir representar y operar sobre un sistema cuántico, del cual no se posee toda la información. En éste contexto, existen dos tipos de probabilidades asociadas a un sistema cuántico:

- Dado un obsevable O, la probabilidad relacionada a colapsar un estado arbitrario $|\psi\rangle$, luego de realizar la medición, en un valor esperado entre los valores propios de O. Se deriva directamente de la norma cuadrada de una amplitud de probabilidad c_i. Este tipo de probabilidad es asociada a los estados puros, es decir, a aquellos sistemas donde se conoce exactamente en qué estado se encuentra el sistema. Para los estados puros, se define la matriz de densidad de la forma

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

- El otro tipo, está asociado a la probabilidad de encontrar un sistema particular en uno de sus posibles estados. Ésto significa que no conocemos el sistema en su totalidad, y se define como la probabilidad p_i de que el sistema se encuentre en su estado puro $|\psi_i\rangle$. Los llamamos estados mixtos, y su matriz de densidad se representa de la siguiente forma:

$$\rho = \sum_{i=1}^{n} p_i |\psi_i\rangle\langle\psi_i|$$

La matriz de densidad ρ es evaluada como un producto tensor. Note que el primer caso, es un caso especial de los estados mixtos, donde la probabilidad p_i de encontrar el sistema en cierto estado $|\psi_i\rangle$ es 1. Teniendo esto en cuenta, entonces se puede ver que

$$\rho = \begin{bmatrix} \psi_0 \\ ... \\ \psi_n \end{bmatrix} \begin{bmatrix} \psi_0 & ... & \psi_n \end{bmatrix} = \begin{bmatrix} \psi_0\psi_0 & ... & \psi_0\psi_n \\ ... & ... & ... \\ \psi_i\psi_0 & \psi_i\psi_j & \psi_j\psi_n \\ ... & ... & ... \\ \psi_n\psi_0 & ... & \psi_n\psi_n \end{bmatrix}$$

2.4.2 Qubit

Se define el qubit como un estado cuántico. Este estado representa la superposición entre el estado 1 y 0 referenciando al bit clásico, cada uno con una amplitud c_i y $c_i \in \mathbb{C}$, y se representa de la forma

$$c_1 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + c_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}$$

La probabilidad en números complejos denotan la amplitud de la onda de probabilidad asociada a la partícula, como se define en la mecánica cuántica. Teniendo esto en cuenta lo anterior, es posible representar N qubits en un solo estado cuántico, para esto se aplica el producto tensor, el cual se define de la forma

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bc \end{bmatrix}$$

Cabe resaltar que ésta operación aplica para vectores de cualquier tamaño.

Dado que la base de la computación cuántica es el qubit, el cual por definición tiene asociado 2 números complejos para su representación. Esto quiere decir que para poder representar N qubits, un computador clásico requeriría 2^N números complejos. Se evidencia que la cantidad de información que puede ser representada por cada qubit crece exponencialmente en base 2 y los algoritmos cuánticos aprovechan éste incremento exponencial para realizar computaciones que clásicamente no se podrían implementar con la misma complejidad temporal.

2.4.3 Compuertas Cuánticas

Son matrices que ejercen algún efecto sobre el qubit. Algunas de ellas tienen su contraparte en el mundo clásico. Por ejemplo, NOT, se espera que cambie el estado de un qubit $|1\rangle$ a $|0\rangle$ y viceversa. Dado que actua sobre un solo qubit, se requiere una matriz de 2×2 para ser representada.

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

de forma que al aplicarla sobre un qubit, por ejemplo, en estado $|1\rangle$, se tenga el sigiente efecto.

$$NOT|1\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} = |0\rangle$$

Existen compuertas especiales, las cuales se van a usar más adelante, como

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad R_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$$

donde H constrye un estado de superposición con un qubit, es decir, tanto para $|0\rangle$ como para $|1\rangle$, al aplicar H o matriz de Hadamard, existe una probabilidad del 50% de ser $|0\rangle$ o $|1\rangle$, donde $H|0\rangle = 1/\sqrt{2}(|0\rangle + |1\rangle)$ y $H|1\rangle = 1/\sqrt{2}(|0\rangle - |1\rangle)$. La compuerta de cambio de fase R_ϕ no afecta al estado $|0\rangle$ y en cambio, al aplicar sobre $|1\rangle$ se obtiene $R_\phi|1\rangle = e^{i\phi}|1\rangle$, afectando la probabilidad directamente con respecto al ángulo de rotación ϕ. Cuando $\phi = \pi/2$, la compuerta se denomina S y se usará en el circuito más adelante.

Por último, la compuerta SWAP, actúa sobre 2 qubits, por lo tanto debe ser de tamaño $2^2 = 4$. Actua de la siguiente manera

$$SWAP|01\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = |10\rangle$$

Como se puede ver, su efecto como su nombre lo indica, es alterar los valores de un estado cuántico compuesto por qubits. Existen más compuertas, y formas de construir compuertas a partir de la lógica cuántica, sin embargo, con esto es suficiente para entender el experimento.

Algorithm	Classic	Quantum	Speedup
Inferencia Bayesiana	$O(poly(NM))$	$O(poly(NM))$	$O(\sqrt{N})$
Perceptrón	$O(\exp N)$	$O(\exp N)$	$O(\sqrt{N})$
Aproximación Mínimos Cuadrados	$O(NM^2)$	$O(M \log N)$	$O(\log N)$
Máquina de Boltzmann	$O(N \exp M)$	$O(N \exp M)$	$O(\sqrt{N})$
PCA	$O(NM^2 + M^3)$	$O(M \log N + (\log M)^2)$	$O(\log N)$
SVM	$O(poly(NM))$	$O(\log(NM))$	$O(\log N)$
Aprendizaje Reforzado	$O(N \exp M)$	$O(N \exp M)$	$O(\sqrt{N})$

Figure 3: Comparación en complejidad temporal. Un speedup de $O(\log N)$ significa una mejora exponencial con respecto a la contraparte clásica. De igual forma, $O(\sqrt{N})$ representa una mejora cuadrática

3 Estado del Arte

3.1 Machine Learning Cuántico

La idea principal detrás de el procesamiento de información a nivel cuántico se basa en el siguiente supuesto: Si procesadores cuánticos son capaces de producir patrones estadísticos computacionalmente difíciles de producir por un computador clásico, entonces es muy probable que también puedan reconocer patrones sobre dicha información igualmente difíciles de encontrar clásicamente. Esta afirmación depende mucho de la capacidad de formular algoritmos de aprendizaje que usen eficientemente las propiedades de la computación cuántica sobre el análisis estadístico de la información. A continuación, una tabla que muestra una selección de algoritmos para los cuales se ha diseñado una implementación cuántica, y su mejora en complejidad temporal con respecto a sus contrapartes clásicas [Figura 3].

Estas subrutinas son principalmente 3: Transformaciones de Fourier, cálculo de valores y vectores propios (eigenvalores y eigenvectores) y sistemas de ecuaciones lineales, entre otras. Lo que éstas tienen en común, es que se requiere de un algoritmo de inversión de matrices, el cual ofrece una mejora exponencial sobre la contraparte clásica, teniendo en cuenta ciertas condiciones ($O(k^2 \log N)$, donde k corresponde al numero de condición asociado a la matriz).

3.2 Subrutinas básicas de álgebra lineal – qBLAS

Gran parte de las operaciones involucradas en el aprendizaje de máquinas involucran conceptos de algebra lineal y transformaciones matriciales, por lo tanto, diseñar un algoritmo que sea capaz de resolver ciertas operaciones que involucran matrices implica una mejora de igual proporción en el algoritmo de aprendizaje

	Clásico	Cuántico
Transformada de Fourier	$O(d\log d)$	$O((\log d)^2)$
Valores/Vectores Propios	$O(d^3)$	$O((\log d)^2)$
Sistema de ecuaciones lineales	$O(d\log d)$	$O((\log d)^3)$

Figure 4: Comparación en complejidad temporal para las subrutinas mencionadas

como tal. Como se evidencia en la tabla anterior, ésta mejora para algunos algoritmos de aprendizaje corresponde a una mejora exponencial. Esto se debe principalmente a que las subrutinas básicas de algebra lineal (BLAS), los cuales son algoritmos empleados por la gran mayoría de métodos de aprendizaje, pueden ser definidos mediante algoritmos cuánticos, los cuales representan una mejora exponencial en complejidad temporal. Para aquellos algoritmos que presentan una mejora cuadrática, se debe a que emplean otros mecanismos de aprendizaje que sacan provecho del algoritmo de búsqueda de Grover, el cual dado un arreglo desordenado de tamaño N, es capaz de encontrar un elemento cualquiera en complejidad temporal $O(\sqrt{N})$.

Los detalles de implementación están referenciados más adelante, sin embargo, una de las principales razones por las que se da este crecimiento exponencial, es la capacidad que tienen los sistemas cuánticos de representar información de una manera que resulta imposible en un sistema clásico, aprovechando la propiedad de superposición para comprimir los datos de entrada de forma exponencial. Específicamente, dado un vector de entrada $v \in C^d$, es posble codificar dicho vector en términos de amplitud en una superposición, de forma que se requieran $\log_2 d$ qubits para su representación, expresado de la forma $v_j \rightarrow |v_j\rangle = \sum_j v_j^i |i\rangle$.

3.3 Aprendizaje Supervisado

Corresponde a un grupo de métodos de aprendizaje, mediante los cuales se emplean datos de entrenamiento al sistema, y se definen las posibles categorías a clasificar, de forma que una vez la red aprenda a clasificar correctamente los datos de entrenamiento (input) a su correspondiente valor de salida (output), la red pueda clasificar un nuevo conjunto de datos nunca visto de forma correcta.

Un claro ejemplo de aprendizaje supervisado, son las machinas de vectores de soporte. La idea principal detrás de este algoritmo de clasificación y regresión, teniendo un conjunto de datos representados en un plano, se define un vector el cual es capaz de dividir el plano según los criterios establecidos, capaz de clasificar a qué grupo pertenece un nuevo dato ingresado al sistema. Adicionalmente, mediante técnicas avanzadas de algebra lineal es posible extrapolar el vector a un hiperplano, el cual clasifica los datos teniendo en cuenta múltiples parámetros o características (features, como se conoce popularmente en ML). Clásicamente,

este algoritmo tiene una complejidad de $O(\log \epsilon^{-1} poly(N, M))$ donde N es la dimensión del espacio de características, M es el número de vectores usados para entrenar el algoritmo, y ϵ corresponde al nivel de precisión esperado. Su contraparte cuántica emplea una complejidad temporal de $O(\log(NM))$.

3.4 Aprendizaje no supervisado

A diferencia del aprendizaje supervisado, el principal método es definir una red cuyo principal objetivo es aprender los criterios necesarios para clasificar los datos de entrada, analizando patrones en la información, y ser capaz de clasificar los datos en diferentes clústeres, según los criterios definidos.

Existe una gran variedad de algoritmos en relación con este tipo de aprendizaje, sin embargo, uno de los más usados corresponde al análisis de componentes principales (PCA). Dado un vector de entrada, se define una matriz de covarianza sobre el mismo, de forma que se pueda evaluar la proporción por la que cambia una variable con respecto a otra. Una vez se tiene la matriz, se busca 'diagonalizar' dicha matriz realizando un cambio de base sobre sus vectores propios, de forma que la diagonal solo tenga los valores propios correspondientes a cada vector. Lo que se busca analizar es aquellos vectores que tienen un valor mucho mayor a 0, en contra de aquellos que son 0 o muy cercano. A estos se les llama componentes principales, y cada uno representa un patrón de correlación intrínseco en los datos representados por el vector de entrada. Así mismo, teniendo estos vectores es posible comprimir la información y así mismo, buscar predicciones sobre los datos. Clásicamente el algoritmo tiene una complejidad temporal de $O(d^2)$, donde d corresponde a la dimensión del vector original. Jugando un poco con la representación de los datos a nivel cuántico, es posible, como ya se vio anteriormente, representar el vector de entrada en términos de amplitudes en un estado de superposición. Una vez se tiene el vector de amplitudes, se define la matriz de densidad de probabilidad $\rho = (1/N) \sum_j |v_j\rangle\langle v_j|$, donde N es el número de vectores y un vector seleccionado al azar. Esta matriz de densidad corresponde exactamente a la matriz de covarianza del método clásico. De igual forma se procede a encontrar los valores y vectores propios utilizando versiones cuánticas de dichos algoritmos, resultando en una complejidad temporal de $O(\log(d)^2)$.

Hasta ahora se ha explorado la manera de mejorar exponencialmente los modelos ya existentes de aprendizaje automático, no obstante, existen modelos que surgen gracias a las funcionalidades ofrecidas por un modelo cuántico. Un claro ejemplo es la máquina de Boltzmann, que permite clasificar la información dado un vector de entrada, sin embargo, también es capaz de ejecutar el proceso de manera inversa. Dada la respuesta deseada, se espera recrear un modelo estadístico que pueda representar dicho output, es decir, retornar lo que se ha visto anteriormente como el vector de entrada. Sin embargo, la propuesta plantea que

es posible recrear todos los vectores de entrada que cumplan el patrón que da la respuesta seleccionada, en un estado de superposición. Sin embargo, este tipo de modelos aún es muy teórico y no se espera una pronta implementación de este.

4 Algoritmos de Aprendizaje Cuánticos

A pesar de las limitaciones encontradas en la computación clásica, los algoritmos de aprendizaje de máquina son ampliamente usados en la industria, y la forma tradicional de implementarlos en un computador clásico no cambiará pronto. Si bien la complejidad temporal de algunos algoritmos puede llegar a ser exponencial, la aceleración por hardware minimiza el efecto que dicha complejidad puede llegar a tener en tiempos reales de ejecución y a pesar de que la computación cuántica en general está tomando un mayor peso en la industria, donde incluso empresas como IBM ya han proyectado planes para construir computadores cuánticos comerciales, el percio del hardware actual con relación a los costos de implementación de un computador cuántico son considerables y limitan el avance estrictamente a aquellos con el limitado acceso a recursos cuánticos.

El objetivo de este experimento es demostrar que es posible estudiar y entender los efectos de la computación cuántica sobre los algoritmos actuales, para investigar, conocer y eventualmente crear algoritmos más poderosos que realmente justifiquen el salto a esta nueva tecnología, conforme el ambiente va creciendo y se va facilitando conforme más gente entra en el mundo cuántico.

4.1 Máquinas de Vector Soporte (qSVM)

La implementación del algoritmo está basada en el experimento descrito en [1], hecho en la Universidad de Ciencia y Tecnología de China, Hefei, el cual a su vez, se basa en la descripción [2] para clasificación de Big Data, donde se exploran técnicas para mejorar exponencialmente la complejidad del algoritmo, específicamente, con la implementación de un algoritmo para resolver el producto interno entre vectores, aprovechando el paralelismo cuántico para hallar los parámetros del hiperplano con la matriz de Kernel, y posteriormente para el proceso de clasificación.

El objetivo es construir un clasificador, en el contexto de reconocimiento óptico de caracteres (OCR), con un procesador cuántico de 4 qubits, con el fin de clasificar imágenes de los dígitos 6 y 9 en sus respectivas clases. Los detalles físicos de la implementación del experimento están fuera del alcance de este artículo, sin embargo, pueden ser encontrados en [1].

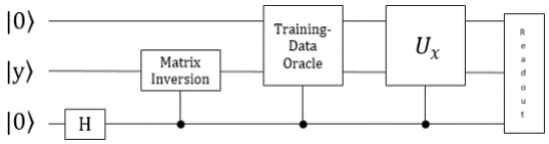

Figure 5: Vista general del algoritmo planteado

Como se vio anteriormente, la clasificación está dada por la expresión $y(\vec{x_0}) = \boldsymbol{sgn}(\vec{w}\cdot\vec{x_0}+b)$ donde $\vec{x_0}$ es el vector que se desea clasificar, $\vec{w}$ el vector normal al hiperplano y b el offset con respecto al origen del espacio. Así mismo $y(\vec{x_0}) = +1$ corresponde a una clasificación positiva (6), y $y(\vec{x_0}) = -1$ a una clasificación negativa (9) del vector $\vec{x_0}$. Las características, se toman de las imágenes $p \times p$ (pixeles) de los caracteres, donde $\vec{x_{i1}}$ y $\vec{x_{i2}}$ corresponde al ratio horizontal y vertical respectivamente de los pixeles de la imagen. Se divide el número de pixeles usados con respecto a la mitad de la imagen, para cada axis x, y. Por lo tanto, los vectores de entrenamiento $\vec{x_1}$ y $\vec{x_2}$, son respectivamente, utilizando caracteres de la fuente Times New Roman,

$$preprocess(\mathbf{6}) = \vec{x_1} = (0.987, 0.159) \quad y(\vec{x_1}) = 1$$

$$preprocess(\mathbf{9}) = \vec{x_2} = (0.354, 0.935) \quad y(\vec{x_2}) = -1$$

Como se mencionaba anteriormente, $\vec{w} = \sum_{i=1}^{M}$, en este caso $M = N = 2$. M corresponde al número de datos de entrenamiento y N el número de características tal que $x_i \in \mathbb{R}^N$.Para este caso, es posible construir una versión que trasnforma el problema de optimización cuadrática en un sistema de ecuaciones lineales, de las cuales se pueden derivar los α teniendo que $b = 0$ (es decir, solo se construirá un solo hiperplano), por lo tanto tambien $\vec{w}$. Para esto, se resuelve

$$\widetilde{F}(b, \alpha_1, \alpha_2)^T = (0, y_1, y_2)^T$$

Donde $\widetilde{F}$ es una matriz $(M+1) \times (M+1)$, y F se define de la siguiente forma, tal que

$$F \equiv (K + \gamma^{-1} I_M) \quad \widetilde{F} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & F_{00} & F_{01} \\ 1 & F_{10} & F_{11} \end{bmatrix}$$

K es la matriz de kernel asociada, la cual corresponde a

$$K_{i,j} = k(\vec{x_i}, \vec{x_j}) = \vec{x_i} \cdot \vec{x_j}$$

En este caso, la función k es un simple kernel lineal, donde el resultado es el producto punto de los dos vectores sin ningún tipo de transformación, tal como se muestra en el problema de optimización cuadrático.

Teniendo en cuenta lo anterior, se procede a definir el algoritmo de manera cuántica. Para empezar, se desea calcular la matriz K haciendo uso del algoritmo para encontrar el producto punto. Se desea un oráculo capaz de producir el estado

$$|\vec{x_i}\rangle = \frac{1}{|\vec{x_i}|}\sum_{j=1}^{N}(\vec{x_i})_j|j\rangle$$

Partiendo del estado inicial

$$(\frac{1}{\sqrt{M}})\sum_{i=1}^{M}|i\rangle$$

Se espera que el oráculo de datos de entrenamiento produzca el estado $|\chi\rangle$, con el cual se espera encontrar $\boldsymbol{tr}K$, luego de descartar el registro del conjunto de entrenamiento. Esto se logra, en el caso del experimento, mediante el algoritmo de traza parcial el cual se explica más adelante.

$$|\chi\rangle = \frac{1}{\sqrt{N_\chi}}\sum_{i=1}^{M}|\vec{x_i}||i\rangle|\vec{x_i}\rangle, \quad N_\chi = \sum_{i=1}^{M}|\vec{x_i}|^2$$

Una vez se tiene la matriz K, es posible construir $\widetilde{F}$, con el objetivo de resolver el sistema para encontrar los parámetros adecuados al hiperplano. Ya existe un algoritmo cuántico para resolver sistemas de equaciones lineales con una mejora en complejidad temporal exponencial. La ecuación para encontrar los parámetros se define mediante la inversión $(b, \vec{\alpha}^T)^T = \widetilde{F}^{-1}(0, \vec{y}^T)^T$. Para empezar, se inicializa el registro cuántico en el estado $|0, \vec{y}\rangle = (1/\sqrt{N_{0,y}})(|0\rangle + \sum_{i=1}^{M} y_i|i\rangle)$. Al hacer la inversión de la matriz F, el estado cuántico se transfiere a

$$|b, \vec{a}\rangle = \frac{1}{\sqrt{N_{b,a}}}(b|0\rangle + \sum_{k=1}^{M}\alpha_i|i\rangle)$$

$N_{0,y}$ y $N_{b,a}$ simplemente son factores de normalización. Dado que la operación sobre $|b, \vec{a}\rangle$ depende de si el bit de acarreo (anc) es $|1\rangle$, la siguiente operación es condicional, es decir, se construye de forma que el bit anc sea un bit de control. Teniendo en cuenta la descripcón del SVM, se puede decir que

$$y(\vec{x_0}) = \boldsymbol{sgn}(\sum_{i=1}^{M}\alpha_i(\vec{x_i}\cdot\vec{x_0}) + b)$$

La forma en la que esto puede ser replicado cuánticamente es mediante la superposición de dos estados cuánticos $y(\vec{x_0}) = \boldsymbol{sgn}(\langle\widetilde{v_0}|\widetilde{u}\rangle)$, donde

$$|\widetilde{v}\rangle = (\frac{1}{\sqrt{N_{\widetilde{v}}}})[b|0\rangle|0\rangle + \sum_{i=1}^{M} abs(\vec{x_i})\alpha_i|i\rangle|\vec{x_i}\rangle]$$

$$|\widetilde{u}\rangle = (\frac{1}{\sqrt{N_{\widetilde{u}}}})[|0\rangle|0\rangle + \sum_{i=1}^{M} abs(\vec{x_0})|i\rangle|\vec{x_0}\rangle]$$

Se espera obtener $|\widetilde{v}\rangle$ llamando al oraculo de datos de entrenamiento sobre $|b, \vec{\alpha}\rangle$ y un registro adicional. Para obtener $|\widetilde{u}\rangle$ es necesario construir un U_{x_0} que introduzca los datos del vector de consulta $\vec{x_0}$ al circuito. Esta operación debe ser unitaria y debe satisfacer que $U_{x_0}|\widetilde{v}\rangle = |00\rangle$, de forma que $\langle 0|U_{x_0}||\widetilde{u}\rangle = \langle\widetilde{v_0}||\widetilde{u}\rangle$ Dado que las operaciones estan controladas por el bit de acarreo, el estado final es (donde $|\lambda\rangle_A$ denota el estado del bit de acarreo)

$$\psi\rangle = |\phi\rangle|1\rangle_A + |00\rangle|0\rangle_A, \quad \phi\rangle = U_{x_0}|\widetilde{u}\rangle$$

Por último, se construye un observable de la forma $O \equiv |00\rangle\langle| \otimes (|1\rangle\langle 0|)_A$, de forma que el resultado de la clasificación sea

$$y(\vec{x_0}) = \boldsymbol{sgn}(\langle 00|\phi\rangle) = \boldsymbol{sgn}(\langle\psi|O|\psi\rangle)$$

4.1.1 Implementación

El principal objetivo del circuito cuántico es construir las compuertas necesarias para obtener la contraparte cuántica de los pasos descritos del algoritmo SVM. El esquema general, muestra los tres pasos generales a seguir: el algoritmo de inversión de matrices, el oráculo de los datos de entrenamiento y la versión cuántica del vector a clasificar, seguido de la lectura en el primer registro. En la siguiente figura se muestra en detalle el circuito [Fig 3].

El circuito está diseñado sobre un computador cuántico basado en la tecnología de resonancia magnética nuclear (NMR), ampliamente usada en experimentos de este tipo. Los detalles se encuentran en [1], sin embargo, en resumen, se utiliza un compuesto a base de Carbono (C), Fluor (F) y Yodo (I) (C_2F_3I), donde el spin de los átomos asociados se utiliza como registro cuántico para realizar las operaciones. Luego de ejecutar el circuito, se utiliza un espectómetro que con la tecnología NMR, es capaz de medir la amplitud del spin, en este caso, del átomo de carbono donde se codifica la respuesta, y se mide su amplitud, magnitud y

dirección, ésta última indicando la respuesta de clasificación según sea positiva o negativa.

Adicionalmente a las matrices descritas en el marco teórico (S, Swap y H), las cuales son las más comunes en éste contexto, se introducen operaciones que tienen usos mucho más específicos. Específicamente:

$$S^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix} R_y(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{bmatrix} R_y^{-1}(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & \sin\frac{\theta}{2} \\ -\sin\frac{\theta}{2} & cos\frac{\theta}{2} \end{bmatrix}$$

Donde S^{-1} simplemente es la inversa de S, y $R_y(\theta)$ y su inversa corresponden a una rotación en la esfera de bloch sobre el eje y, o cambio de fase θ veces, diferenciando entre ellas el sentido de la rotación.

Las compuertas de la forma $e^{i\pi F}$ y similares, son matrices exponenciales (de la forma e^{tA} donde A es una matriz $n \times n$), dado que F es la matriz de Kernel ajustada con la corrección γ. Su discusión está fuera del alcance de este texto, sin embargo, existen múltiples herramientas que facilmente ayudan a resolver éste tipo de expresiones. El resultado corresponde a la siguiente serie:

$$e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k$$

El primer paso para resolver el circuito, es encontrar la matriz Kernel, mediante el circuito descrito por el oráculo de entrenamiento. Este circuito, recibe como parámetros los thetas asociados a los caracteres de entrenamiento, donde $\theta_i = arccot((\vec{x_i}_1)/(\vec{x_i}_2))$. Este valor corresponde a la operación de arco cotangente asociada a los atributos del caracter (razón horizontal y vertical). $\vec{x_1}$ corresponde a 6, y $\vec{x_2}$ corresponde a 9, luego

$$\theta_1 = arccot(\frac{0.987}{0.159}) = 9.15^o \quad \theta_2 = arccot(\frac{0.354}{0.935}) = 69.26^o$$

Por lo tanto, las matrices de rotación asociadas son, aproximadamente

$$R_y(\theta_1) = \begin{bmatrix} 1 & -0.08 \\ 0.08 & 1 \end{bmatrix} \quad R_y(\theta_2) = \begin{bmatrix} 0.82 & -0.56 \\ 0.56 & 0.82 \end{bmatrix}$$

Teniendo en cuenta que ambas compuertas están bajo control, donde el circulo blanco significa que se aplicará la rotación si el qubit se encuentra en estado $|0\rangle$, y viceversa para el circulo negro con el estado $|1\rangle$. Para entender qué efecto tienen éstos qubits de control, así se ven las versiones controladas de las

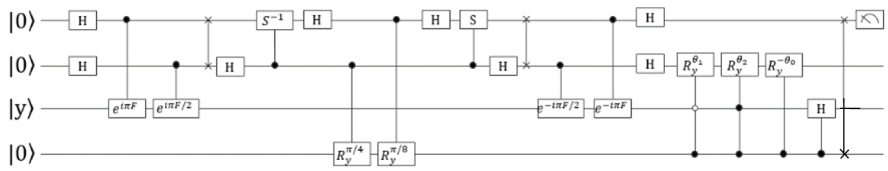

Figure 6: Vista detallada del algoritmo planteado

matrices anteriores, donde los prefijos $ac\_$ y $c\_$ corresponden a anti control y control respectivamente.

$$ac\_R_y(\theta_1) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -0.08 \\ 0 & 0 & 0.08 & 1 \end{bmatrix} \quad ac\_R_y(\theta_2) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0.82 & -0.56 \\ 0 & 0 & 0.56 & 0.82 \end{bmatrix}$$

Teniendo claro todos los componentes del circuito para encontrar la matriz Kernel, el proceso para resolverlo es el siguiente [Figure 7]

$$c\_R_y(\theta_2) \times ac\_R_y(\theta_1) \times (H \otimes I) \times |0,0\rangle$$

El resultado es el siguiente estado mixto. Adicionalmente, su matriz de densidad.

$$|\psi\rangle = \begin{bmatrix} 0 \\ 0.70 \\ 0.54 \\ 0.44 \end{bmatrix} \quad \rho(|\psi\rangle) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0.50 & 0.38 & 0.31 \\ 0 & 0.38 & 0.30 & 0.24 \\ 0 & 0.31 & 0.24 & 0.20 \end{bmatrix}$$

Una vez se tiene la matriz de densidad del estado, se aplica el algoritmo de traza parcial para descartar el segundo registro y obtener la matriz de densidad equivalente a la del primer registro, la cual según el experimento es la misma matriz $K/\boldsymbol{tr}K$ que se usa para hallar la matriz F. El algoritmo de traza parcial para matrices 4×4 es el siguiente

$$\boldsymbol{Tr}_A = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} = \begin{bmatrix} a+f & c+h \\ i+n & k+p \end{bmatrix}$$

Comparando los resultados, se puede evidenciar que el resultado es el esperado, dentro de cierto margen de error, el cual puede ser atribuido a las diferencias entre la experimentación con un computador cuántico real y una simulación, además del manejo de números flotantes en procesadores clásicos.

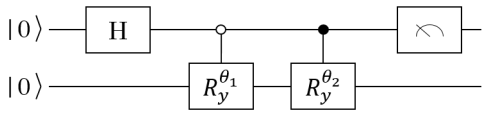

Figure 7: Algoritmo para encontrar la matriz de kernel (Entrenar el modelo)

$$\rho(|\psi\rangle) = \begin{bmatrix} 0.5 & 0.31 \\ 0.31 & 0.5 \end{bmatrix} \quad K/\boldsymbol{tr}K = \begin{bmatrix} 0.5065 & 0.2425 \\ 0.2425 & 0.4935 \end{bmatrix}$$

Habiendo demostrado el algoritmo y su funcionamiento, se usa la versión del experimento original para continuar con el cálculo, ya que es la más exacta, a pesar de la minima diferencia. Dado que F se define como $F \equiv (K + \gamma^{-1}\boldsymbol{I}_M)$, la cual equivale aproximadamente a $\boldsymbol{tr}K$ para $\gamma = 2$ (valor seleccionado en el experimento). Cabe resaltar que γ es un parámetro especificado por el usuario y mientras más alto es su valor, menor es la varianza del modelo aumentando el cesgo en la clasificación.

Teniendo F, ahora es posible hallar las matrices exponenciales relacionadas al proceso de inversión de matrices, las cuales manipulan el tercer registro con la información de las etiquetas de los datos de entrenamiento (y_i y y_2, 1 y -1 respectiva y aproximadamente) de la forma $|y\rangle = \frac{\binom{y_1}{y_2}}{\sqrt{2}}$.

$$e^{i\pi F} = \begin{bmatrix} -0.7234 - 0.0185i & 0 - 0.6902i \\ 0 - 0.6902 & -0.7234 + 0.0185i \end{bmatrix}$$

$$e^{i\pi \frac{F}{2}} = \begin{bmatrix} -0.01 + 0.9283i & -0.3718 \\ -0.3718 & 0.01 + 0.9283i \end{bmatrix}$$

Las matrices $e^{-i\pi F}$ y $e^{-i\pi F/2}$ simplemente corresponden a las inversas de la respectiva matriz exponencial.

Antes de continuar, vale la pena mencionar que el orden del bit de control para las compuertas controladas afecta como se construye la matriz. Ya se mostró anteriormente cuando el qubit de control es el primero, pero qué pasa cuando el qubit de control se encuentra abajo, como ocurre con las compuertas del oráculo y de U_{x_0}. A continuación la matriz de control inversa de Hadamard y $R_y(\theta_1)$ (teniendo en cuenta solo el segundo y tercer registro) para mostrar respectivamente la forma de la matriz con un qubit de control y anti-control.

$$c\_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 & 1 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \quad ac\_R_y(\theta_1) = \begin{bmatrix} 1 & 0 & -0.08 & 0 \\ 0 & 1 & 0 & 0 \\ 0.08 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Una vez claros los parámetros del hiperplano, se procede a clasificar un nuevo dato $\vec{x_0} = (0.147, 0.989)$, el cual corresponde a un caracter 9. Para esto se definen θ_0 y $R_y^{-1}(\theta_0)$

$$\theta_0 = arccot(\frac{0.147}{0.989}) = 1.4232 \quad R_y(\theta_0) = \begin{bmatrix} 0.7573 & 0.6530 \\ -0.6530 & 0.7573 \end{bmatrix}$$

Todas las compuertas del circuito se ajustan al tamaño $2^4 \times 2^4$ teniendo en cuenta el número de registros. El estado final correspondiente al circuito, con esto datos de entrada y teniendo en cuenta los parámetros del hiperplano, es el siguiente

$$(SWAP_1 6)\times$$
$$(I_4 \otimes c\_H)(I \otimes c\_R_y^{-1}(\theta_0))\times$$
$$(I \otimes c\_c\_R_y(\theta_2))(c\_ac\_R_y(\theta_1))\times$$
$$(H \otimes H \otimes I_4)(c\_e^{-i\pi F} \otimes I_2)(I_2 \otimes e^{-i\pi \frac{F}{2}} \otimes I_2)\times$$
$$(SWAP_4 \otimes I_8)(I \otimes H \otimes I_4)\times$$
$$(c\_S \otimes I_8)(H \otimes I_8)\times$$
$$(c\_R_y(\frac{\pi}{8}))(I \otimes c\_R_y(\frac{\pi}{4}))\times$$
$$(H \otimes I_8)(c\_S^{-1} \otimes I_4)\times$$
$$(I \otimes H \otimes I_4)(SWAP_4 \otimes I_4)\times$$
$$(I \otimes e^{i\pi \frac{F}{2}} \otimes I)\times$$
$$(H \otimes H \otimes I_4)\times$$
$$|0,0,0,0\rangle$$

Luego de evaluar tan larga expresión, se obtiene un resultado final $|\psi\rangle$. El truco de la traza parcial solo funciona con dos registros y bajo ciertas restricciones, y no es posible construir el observable que corresponde a la lectura en el espectómetro con tegnología (NMR). Sin embargo, teniendo en cuenta que la lectura se realiza sobre el primer registro, y que el valor esperado corresponde

a positivo para 6, y negativo para 9, se entiende que el estado final es un entrelazamiento entre todos los registros usados en el circuito, y es posible simular un observable que permita realizar la clasificación. El estado final $|\psi\rangle$.

$$|\psi\rangle = \begin{bmatrix} (1.1047728180439458 + 0.06381888831075151i) \\ (-0.1258039462847832 + 0.22888916056370265i) \\ (-0.43593398064112043 - 0.018736902692599144i) \\ (0.2949903145878561 + 0.01389692917327686i) \\ (-0.21431560278431863 - 0.27756308404158275i) \\ (0.12851417804573315 - 0.007270634151512745i) \\ (0.21864619145876216 + 0.01736212886306917i) \\ (-0.08127307303992118 - 0.05142116056686084i) \\ (0.15497272437095622 - 0.0667192866461708i) \\ (-0.017883154381689515 + 0.16199251771484846i) \\ (0.2706116170790217 + 0.07860870931905856i) \\ (0.014946098882381763 - 0.09968059949171401i) \\ (-0.0898342389582101 - 0.26068620967028855i) \\ (0.16917638815629304 - 0.0589313728714254i) \\ (-0.035606421172640984 - 0.4339226777897079i) \\ (-0.045861440316261035 + 0.05431758563807792i) \end{bmatrix}$$

Teniendo en cuenta este resultado, se observa que el estado tiene una tendencia negativa, es decir, existen más elementos positivos. Es una especulación, sin embargo para los demás resultados también se muestra el mismo comportamiento (Más positivo para 6, Más negativos para 9). De esto se puede concluir que la simulación funciona correctamente, salvo por la imposibilidad de construir un observable adecuado. Con esto se concluye que el resultado de la clasificación para $\vec{x_0}$ es un 9 ($y(\vec{x_0}) = -1$).

5 Conclusiones

Si bien es mucho más complicado realizar simulaciones sobre circuitos cuánticos en computadores clásicos, hay que entender que dichas simulaciones representan algoritmos cuánticos reales sobre procesadores reales, que tienen la capacidad de ejecutar dicho algoritmo de manera intrínseca, dada la naturaleza de los procesadores cuánticos. Hay que recordar que los computadores cuánticos representan en algunos casos una mejora exponencial en complejidad temporal y que a pesar de la confusión que puede generar la computación cuántica y la forma de realizar los algoritmos, hay que entender que el grado de dificultad se mantendrá constante, y este es el futuro. Además, empresas como IBM, Google, Microsoft, grandes universidades en diversos países, particularmente China, que está actualmente a la vanguardia tecnológica, apuntan a esta tecnología y buscan la manera de hacerla más accesible a la academia. Así como la

computación clásica está basada en el modelo deterministico del materialismo y la física clásica, la computación cuántica planta sus bases en la probabilidad y la naturaleza estocástica del mundo microscópico, de forma que es entendible que se desee explorar las diversas y posibles ventajas que podrá traer ésta tecnología al mundo de la computación.

References

[1] Zhaokai Li, Xiaomei Liu, Nanyang Xu, Jiangfeng Du.
Experimental Realization of a Quantum Support Vector Machine.
University of Science and technology of China, Hefei, China. 2015.
https://doi.org/10.1103/PhysRevLett.114.140504

[2] Patrick Rebentrost, Masoud Mohseni, Seth Lloyd.
Quantum Support Vector Machine for Big Data Classification.
MIT, Massachusetts, USA. 2014.
https://doi.org/10.1103/PhysRevLett.113.130503

[3] Dawid Kopzcyk
Quantum Machine Learning for Data Scientists.
Quantee Limited, Manchester, UK. 2018.
arXiv:1804.10068

[4] Seth Lloyd, Masoud Mohseni, Patrick Rebentrost.
Quantum Principal Component Analysis.
MIT, Massachusetts, USA. 2013.
10.1038/nphys3029

[5] Noson S. Yanofsky, Mirco A. Mannucci.
Quantum Computing for Computer Scientists.
Cambridge. 2013.

[6] Vladimir Vapnik, Corinna Cortes.
Suppoert Vector Networks.
Ktuwer Academic Publishers, Boston, USA. 1995.

[7] Seth Lloyd.
Quantum Machine Learning Lecture.
Keio University Minato, Tokyo, Japon. 2018.
https://www.youtube.com/watch?v=Lbndu5EIWvI

[8] Eric Grimson.
Introduction to Machine Learning.
MIT Open Courseware. 2016.
https://www.youtube.com/watch?v=h0e2HAPTGF4

[9] Patrick Winston.
Learning: Support Vector Machines.
MIT Open Courseware.
https://www.youtube.com/watch?v=_PwhiWxHK8o