UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Aplicaciones de Computación Cuántica a la Inteligencia Artificial

Nicolás Cobelli

Nelson Juambeltz

Javier Pérez

Melisa Techera

Ingeniería en Computación
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Junio de 2022

UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Aplicaciones de Computación Cuántica a la Inteligencia Artificial

Nicolás Cobelli

Nelson Juambeltz

Javier Pérez

Melisa Techera

Tesis de grado presentada en la Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de grado en Ingeniería en Computación.

Directores:
Sergio Nesmachnow
Alfredo Viola

Montevideo – Uruguay
Junio de 2022

Agradecimientos

Agradecemos a nuestras familias por apoyarnos en estos años de estudio y a la Facultad de Ingeniería, por proporcionarnos un buen ambiente para formarnos como profesionales y personas.

Si la física cuántica
no te conmocionó profundamente
es que aún no la has entendido

Niels Bohr

Resumen

La computación cuántica es una disciplina en auge que ofrece nuevas posibilidades a la hora de resolver diferentes problemas computacionales. En este trabajo se explora el uso de la computación cuántica y máquinas de soporte vectorial como modelos de aprendizaje automático para la resolución de tres diferentes escenarios. Un problema de clasificación binaria de células cancerosas, un problema de clasificación multi-clase sobre especies de flores y un problema de optimización de lineas de ómnibus. Las pruebas se realizaron utilizando el cluster del Centro Nacional de Supercomputación y se compararon las soluciones obtenidas mediante computación cuántica con otras obtenidas utilizando métodos de clasificación tradicionales. Los resultados obtenidos muestran un mejor desempeño de la solución cuántica en ambos problemas de clasificación y un peor desempeño en el problema de optimización.

Palabras clave— Computación Cuántica, Aprendizaje automático, Optimización, Virtual Savant, Problema de sincronización de ómnibus, Problema de clasificación de iris, Problema del cáncer de mama

Índice de siglas

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

La *computación cuántica* es un área en desarrollo de la computación en la cual se busca explotar los descubrimientos hechos por la física cuántica para poder resolver problemas complejos de una manera más eficiente. Actualmente ninguna computadora cuántica es lo suficientemente sofisticada como para resolver cálculos que una computadora clásica no pueda, pero su potencial genera un fuerte nivel de interés que ha promovido la investigación en el área [1].

El *aprendizaje automático*, por su parte, es una rama de la inteligencia artificial que busca desarrollar métodos mediante los cuales sea posible realizar predicciones. Este es un sector que ha tenido un crecimiento vertiginoso en los últimos años, con gran cantidad de recursos invertidos, tanto por la academia como por empresas privadas [2].

Dado el gran uso que tiene el aprendizaje automático hoy en día, es interesante buscar mecanismos mediante los cuales se mejoren los resultados obtenidos y la computación cuántica es un candidato para ello. En particular, existen trabajos académicos de relevancia investigando este tema tales como *Supervised learning with quantum enhanced feature maps* que investiga mecanismos para crear máquinas de soporte vectorial cuánticas y por otro lado, *Quantum machine learning in feature Hilbert spaces* describe mapas de características posibles a utilizar. Estos conceptos serán explicados en el capítulo 2.

Para estudiar el uso de la computación cuántica en el campo de la inteligencia artificial, se plantea como primer objetivo generar un relevamiento bibliográfico que permita generar un entendimiento sobre la computación cuántica y sus diferentes formas de aplicación para resolver problemas de opti-

mización. Como segundo objetivo, se propone utilizar el conocimiento generado a partir de la revisión bibliográfica para resolver dos problemas de clasificación y un problema de optimización utilizando máquinas de soporte vectorial cuánticas y clásicas y comparar los resultados.

El primer problema a estudiar trata de una clasificación binaria de células cancerosas utilizando un conjunto de datos provisto por la Universidad de Wisconsin. El segundo corresponde a un problema de clasificación multi-clase de especies de flores y por último, un problema de optimización de sincronización de lineas de ómnibus en Montevideo utilizando el paradigma *Virtual Savant*.

De los tres problemas estudiados en este trabajo, en dos de ellos la solución cuántica obtuvo mejores resultados que la clásica. Mientras que en el restante la clásica obtuvo resultados superiores.

La estructuración de la presente tesis se desarrolla de forma que el capítulo 2 contiene los aspectos teóricos generados a partir de la revisión bibliográfica que son necesarios para la resolución de los diferentes problemas planteados. El capitulo 3 corresponde a la descripción de las herramientas utilizadas para la implementación de las soluciones generadas a lo largo del trabajo, como las bibliotecas y el lenguaje de programación elegido. El capítulo 4 abarca la descripción de la evaluación experimental realizada, donde se plantean de manera formal los problemas a resolver y las técnicas utilizadas en las soluciones generadas. El capítulo 5 describe los resultados recabados en la evaluación experimental para cada solución y método empleado, presentados de forma comparativa entre el método cuántico y tradicional, junto con la descripción del ambiente de pruebas utilizado. El capítulo 6 comprende las conclusiones alcanzadas a partir de todo el trabajo generado, además de sugerencias sobre trabajo futuro que puede ser alcanzado utilizando el presente trabajo como base. Finalmente en la sección de apéndices y anexos se agrega un reporte sobre las diferentes ejecuciones realizadas a la hora de resolver el problema de sincronización de ómnibus incluyendo también ejecuciones no exitosas que pueden servir para evitar re-trabajo a la hora de iterar sobre el presente trabajo.

Capítulo 2

Fundamentos teóricos

Este capítulo define los principales conceptos teóricos de este proyecto de grado. Comienza definiendo nociones matemáticas necesarias, para luego poder desarrollar sobre la computación cuántica, el aprendizaje automático y finalmente el aprendizaje automático cuántico.

2.1. Conceptos matemáticos

En esta sección se presentan algunos conceptos matemáticos que comprenden la bases teóricas de la computación cuántica.

2.1.1. Álgebra lineal

Para comprender la computación cuántica es necesario comprender algunos conceptos previos de álgebra lineal.

2.1.1.1. Espacio vectorial y producto interno

En esta sección se presenta la definición de espacios vectoriales y de producto interno, propiedades que son de gran importancia en el campo de la computación cuántica y muy utilizadas a lo largo de este trabajo.

Definición 1 (Espacio vectorial)**.** *Un espacio vectorial V sobre un cuerpo K es un conjunto no vacío de elementos $k \in K$, que además tiene dos operaciones suma $+$ $(V \times V \mapsto V)$ y producto $\cdot$ $(K \times V \mapsto V)$ para los cuales será cerrado. La suma debe cumplir con las propiedades 1. 2, 3, 4, mientras que el producto debe cumplir con las propiedades 5, 6, 7 y 8. Los elementos de un espacio*

vectorial se les denomina vectores y la notación estándar que se les asocia en la mecánica cuántica es $|v\rangle$, también llamada ket.

Propiedad 1 (Suma - Conmutativa)**.** $u + v = v + u \ \forall u, v \in V$.

Propiedad 2 (Suma - Asociativa)**.** $u + (v + w) = v + (u + w) \ \forall u, v \in V$.

Propiedad 3 (Suma - Elemento neutro)**.** $\exists e \in V$ *tal que* $u + e = u \ \forall u \in V$.

Propiedad 4 (Suma - Elemento opuesto)**.** $\forall u \in V, \exists -u$ *tal que* $-u + u = e$.

Propiedad 5 (Producto - Asociativa)**.** $(a \cdot b) \cdot u = a \cdot (b \cdot u) \ \forall a, b \in K, \forall u \in V$.

Propiedad 6 (Producto - Elemento neutro)**.** $\exists e \in K$ *tal que* $e \cdot u = e \ \forall u \in V$.

Propiedad 7 (Producto - Distributiva respecto a suma vectorial)**.** $a \cdot (u+v) = a \cdot u + a \cdot v \ \forall a \in K, \forall u, v \in V$.

Propiedad 8 (Producto - Distributiva respecto a suma escalar)**.** $(a + b) \cdot u = a \cdot u + b \cdot u \ \forall a, b \in K, \forall u \in V$.

Definición 2 (Producto interno)**.** *El producto interno de dos vectores* $a = [a_1, a_2, ..., a_n]$ *y* $b = [b_1, b_2, ..., b_n]$*, está dado por:* $a.b = \sum_{i=1}^{n} a_i.b_i = a_1b_1 + a_2b_2 + ... + a_nb_n$.

2.1.1.2. Espacio de Hilbert

Se define a continuación el concepto de espacio de Hilbert. Para ello se introducen algunas nociones previas, para luego en la definición 6 presentarlo formalmente. Los espacios de Hilbert son muy utilizados en el campo de las máquinas de soportes vectorial.

Definición 3 (Espacio métrico)**.** *Un espacio métrico es un par (M,d) donde M es un conjunto no vacío y d*: $M \times M \to R$ *es una función real llamada distancia, tal que para cualquier* $x, y, z \in M$ *se cumplen las propiedades 9, 10 y 11.*

Propiedad 9 (Identidad)**.** $d(x, y) = 0 \Leftrightarrow x = y$.

Propiedad 10 (Simetría)**.** $d(x, y) = d(y, x)$.

Propiedad 11 (Desigualdad triangular)**.** $d(x, z) \leq d(x, y) + d(y, z)$.

Definición 4 (Sucesión de Cauchy)**.** *Dado un espacio métrico, una sucesión (x_1,x_2,x_3,....) es de Cauchy si* $\forall \epsilon > 0, \exists N \in \mathbb{N} / \forall m, n > N, d(x_m, x_n) < \epsilon$.

Definición 5 (Espacio métrico completo)**.** *Un espacio métrico (M, d) se dice completo si toda sucesión de Cauchy contenida en M converge a un elemento de M.*

Definición 6 (Espacio de Hilbert)**.** *Sea $\mathcal{H}$ un espacio vectorial con producto interno. Si además, $\mathcal{H}$ es un espacio métrico completo, se dice que $\mathcal{H}$ es un espacio de Hilbert [3].*

2.1.1.3. Operadores

En esta sección se definen los principales operadores que fueron necesarios para para implementar algoritmos cuánticos.

Definición 7 (Operador linear)**.** *Sean V y W espacios vectoriales sobre el mismo cuerpo K. Una función $F\colon V \to W$ es un operador lineal si para todo par de vectores $u, v \in V$ y para todo escalar $k \in K$ se cumplen las propiedades 12 y 13.*

Propiedad 12 (Aditividad)**.** $f(u + v) = f(u) + f(v)$.

Propiedad 13 (Homogeneidad)**.** $f(cv) = cf(v)$.

Definición 8 (Operador identidad)**.** *Dado un conjunto V, un operador identidad I es una función $I\colon V \to V$ tal que $I(x) = x$ para todo $x \in V$.*

Definición 9 (Operador adjunto)**.** *Dado un espacio de Hilbert H y un operador lineal $A\colon H \to H$ se define al operador adjunto de A, A^* como el operador lineal $A^*\colon H \to H$ que satisface $\langle Ax, y\rangle = \langle x, A^*y\rangle$ para todo $x, y \in H$.*

Definición 10 (Operador unitario)**.** *Dado un espacio de Hilbert H y un operador Identidad I, un operador unitario es un operador lineal $U\colon H \to H$ que satisface $U^*U = UU^* = I$.*

2.1.1.4. Valores y vectores propios

Los valores y vectores propios serán utilizados en la sección 3.3.2.1 para definir métodos de reducción de dimensionalidad de problemas. Sea A una matriz. Los valores propios λ de A son escalares, mientras que los vectores propios v de A son vectores distintos de 0, tales que $Av = \lambda v$

2.1.2. Notación de Dirac

Para definir los principales conceptos cuánticos, es habitual utilizar la llamada *Notación bra-ket* o *Notación de Dirac*, debiendo su nombre al matemático y físico británico Paul Dirac, uno de los fundadores de la mecánica cuántica. La notación de Dirac consta de dos partes llamadas bra y ket. Denominamos ket $|\psi\rangle$ a un un elemento de un espacio de Hilbert, mientras que bra $\langle\psi|$ representa el vector dual de $|\psi\rangle$. El concepto de vector dual es definido más adelante en la sección 2.1.3. La notación de Dirac es utilizada en álgebra lineal para espacios vectoriales y en la Definición 11 se formaliza. El producto interno de un vector y un vector dual, se puede escribir en notación de Dirac como $\langle\phi|\psi\rangle$.

Definición 11 (Bra-Ket)**.** *$\langle f|v\rangle$, con $f : V \to \mathbb{C}$ y $v \in V$ y V un espacio vectorial.*

La notación de Dirac se comprende mejor interpretando el ket de braket como un vector columna y el bra como un vector fila en el espacio vectorial. A lo largo de este proyecto de grado, se hará referencia a los estados cuánticos $|0\rangle$ y $|1\rangle$. La definición de ellos está dada por las ecuaciones 2.1 y 2.2.

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{2.1}$$

$$|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{2.2}$$

2.1.3. Vectores duales

Definición 12 (Espacio Dual)**.** *Sea $\mathcal{H}$ un espacio de Hilbert. El espacio dual $\mathcal{H}^*$ es el conjunto de transformaciones lineales $\mathcal{H} \to \mathbb{C}$*

Utilizando la notación de Dirac, el producto interno de dos vectores $|\phi\rangle \in \mathcal{H}$ y $|\psi\rangle \in \mathcal{H}^*$ puede escrbirse como $\langle\psi|\phi\rangle$

2.1.4. Producto tensorial

El producto tensorial es una operación que permite combinar matrices o espacios vectoriales. La operación de producto tensorial será representada por el símbolo $\otimes$. La definición 13 define el producto tensorial de la manera que está definido por P. Kaye, R. Laflamme, and M. Mosca (2007) [4].

El producto tensorial será utilizado en la sección 2.2 para definir algunos conceptos claves de la computación cuántica.

Definición 13 (Producto tensorial - Matrices)**.** *Sean A y B matrices de tamaño $m \times n$ y $p \times q$ respectivamente Entonces la matriz 2.3 es el producto de Kronecker a la izquierda de A y B y define el producto tensorial entre A y B*

$$A \otimes B = \begin{bmatrix} a_{11}b_{11} & \dots & a_{11}b_{1q} & \dots & \dots & a_{1n}b_{11} & \dots & a_{1n}b_{1q} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & \dots & a_{11}b_{pq} & \dots & \dots & a_{1n}b_{p1} & \dots & a_{1n}b_{pq} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{m1}b_{11} & \dots & a_{m1}b_{1q} & \dots & \dots & a_{mn}b_{11} & \dots & a_{mn}b_{1q} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & \dots & a_{m1}b_{pq} & \dots & \dots & a_{mn}b_{p1} & \dots & a_{mn}b_{pq} \end{bmatrix} \tag{2.3}$$

La ecuación 2.4 muestra la aplicación de la definición 13 a un par de vectores de dimensiones arbitrarias.

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \otimes \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_1b_1 \\ a_1b_2 \\ \vdots \\ a_1b_n \\ a_2b_1 \\ \vdots \\ a_2b_n \\ \vdots \\ a_mb_n \end{bmatrix} \tag{2.4}$$

Al estar definido sobre vectores columna en la ecuación 2.4, entonces es posible representarlas como kets en notación de Dirac. Si se tiene un vector $|A\rangle$ y otro vector $|B\rangle$, entonces se considerarán equivalentes las siguientes tres expresiones: $|A\rangle \otimes |B\rangle$, $|A\rangle\,|B\rangle$ *y* $|AB\rangle$.

Definición 14 (Producto tensorial - Espacios)**.** *Sean $\mathcal{H}_1$ y $\mathcal{H}_2$ dos espacios de Hilbert de dimensión m y n, con bases ortonormales $\{a_1, \dots, a_m\}$ y $\{b_1, \dots b_n\}$ respectivamente. $\mathcal{H}_1 \otimes \mathcal{H}_2$ es un nuevo espacio de Hilbert de dimensión $m \times n$ con base ortonormal $\{a_1 \otimes b_1, a_1 \otimes b_2, \dots, a_1 \otimes b_n, a_2 \otimes b_1, \dots, a_m \otimes b_n\}$.*

El producto tensorial además cumple con las siguientes tres propiedades.

Propiedad 14 (Producto tensorial - Asociatividad)**.** *Para todo* $c \in \mathbb{C}, |\psi_1\rangle \in \mathcal{H}_1$ *y* $|\psi_2\rangle \in \mathcal{H}_2$*,* $c(|\psi_1\rangle \otimes |\psi_2\rangle) = (c\,|\psi_1\rangle) \otimes |\psi_2\rangle = |\psi_1\rangle \otimes (c\,|\psi_2\rangle)$*.*

Propiedad 15 (Producto tensorial - Distributividad 1)**.** *Para todo* $|\psi_1\rangle, |\phi\rangle \in \mathcal{H}_1$ *y* $|\psi_2\rangle \in \mathcal{H}_2$*,* $(|\psi_1\rangle + |\phi\rangle) \otimes |\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle + |\phi\rangle \otimes |\psi_2\rangle$*.*

Propiedad 16 (Producto tensorial - Distributividad 2)**.** *Para todo* $|\psi_1\rangle \in \mathcal{H}_1$ *y* $|\psi_2\rangle, |\phi\rangle \in \mathcal{H}_2$*,* $|\psi_1\rangle \otimes (|\psi_2\rangle + |\phi\rangle) = |\psi_1\rangle \otimes |\psi_2\rangle + |\psi_1\rangle \otimes |\phi\rangle$*.*

2.1.5. Función de kernel

Las *funciones de kernel* o *de núcleo* son muy usadas en el campo de la inteligencia artificial. En particular en este trabajo, se explicará el uso que se hizo de ella en la sección 2.3.1.3. Para ello, en primer lugar se introducirán algunas definiciones previas.

Definición 15 (Matriz transpuesta)**.** *Sea* $A \in M^{mxn}$*. Entonces* $A^T \in M^{nxm}$ *es matriz transpuesta de A si y sólo si* $\forall i, j$*,* $a_{i,j} = a^T_{j,i}$*, donde* $a_{i,j}$ *es el elemento de A en la posición (i,j) y* $a^T_{j,i}$ *es el elemento de* A^T *en la posición (j,i) .*

Definición 16 (Matriz conjugada transpuesta)**.** *Sea* $A \in M^{mxn}$ *una matriz de números complejos y* $A^T \in M^{nxm}$ *su matriz transpuesta. La matriz conjugada transpuesta* A^H *resulta de conjugar todos los elementos de* A^T*. Es decir,* $A^H = \overline{A^T}$*.*

Definición 17 (Matriz hermítica)**.** *Sea* $A \in M^{nxn}$*. A es hermítica si y sólo si* $A^H = A$*.*

Definición 18 (Matriz semi-definida positiva)**.** *Sea A una matriz hermítica. Entonces, A es semidefinida positiva si,* $\forall$ *vector columna z,* $z^T Az \geq 0$*.*

Definición 19 (Función simétrica)**.** *Sea f una función que toma como parámetros n números complejos. Entonces, f:* $\mathbb{C}^n \mapsto \mathbb{C}^m$ *es simétrica si y sólo si, dado un conjunto de entradas* $x_1, ..., x_n$*, la evaluación de f es igual para cualquier permutación de las entradas.*

Definición 20 (Función semi-definida positiva)**.** *Sea f una función* $f : \mathbb{R} \mapsto \mathbb{C}$*. Entonces f es semidefinida positiva si y sólo si, para cualquier* $x_1, ..., x_n \in R$ *, la matriz* $A^{nxn} = (a_{i,j})^n_{i,j=1}$*, con* $a_{i,j} = f(x_i - x_j)$ *es semi-definida positiva.*

Definición 21 (Función de kernel positivo definido)**.** *Sea X un conjunto no vacío y K una función simétrica $X \times X \mapsto \mathbb{R}$. Se dice que K es un kernel positivo definido si y sólo si, dado $n \in \mathbb{N}, \exists c_1, .., c_n \in \mathbb{R}$ $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0, \forall x_1, ..., x_n \in X$.*

2.2. Computación cuántica

En esta sección se presentan los principales conceptos téoricos de computación cuántica utilizados en este proyecto de grado.

La computación cuántica está basada en la física cuántica, la cual se diferencia de la clásica en que no es determinista, sino probabilística, con una incertidumbre inherente [5]. En 1982 Richard Feynman elaboró la idea de que la computación cuántica podría brindar una solución al problema de que las computadoras clásicas no serían capaces de simular ciertos fenómenos cuánticos [6].

La razón por la que los sistemas informáticos cuánticos son tan poderosos radica en la diferencia entre el bit, utilizado en la computación clásica, con el *quantum qubit* o *qubit*, utilizado en la computación cuántica. En un sistema clásico, un bit representa uno de dos estados posibles, cero o uno, mientras que en un sistema cuántico un *qubit* puede encontrarse en el estado $|0\rangle$, $|1\rangle$, los cuales corresponden al estado 0 o 1 clásicos, ó en una superposición entre estos dos estados.

$$|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.5}$$

La ecuación 2.5 muestra la representación de un estado cuántico. Los números α y β son números complejos y α^2 y β^2 representan la probabilidad de que el sistema $|\phi\rangle$ se encuentre en el estado $|0\rangle$ o $|1\rangle$ luego de realizar una medición. Como las probabilidades deben sumar 1, entonces se tiene que $|\alpha|^2 + |\beta|^2 = 1$ y por lo tanto el estado de un qubit se puede interpretar geométricamente como un vector unitario en un espacio vectorial de dos dimensiones.

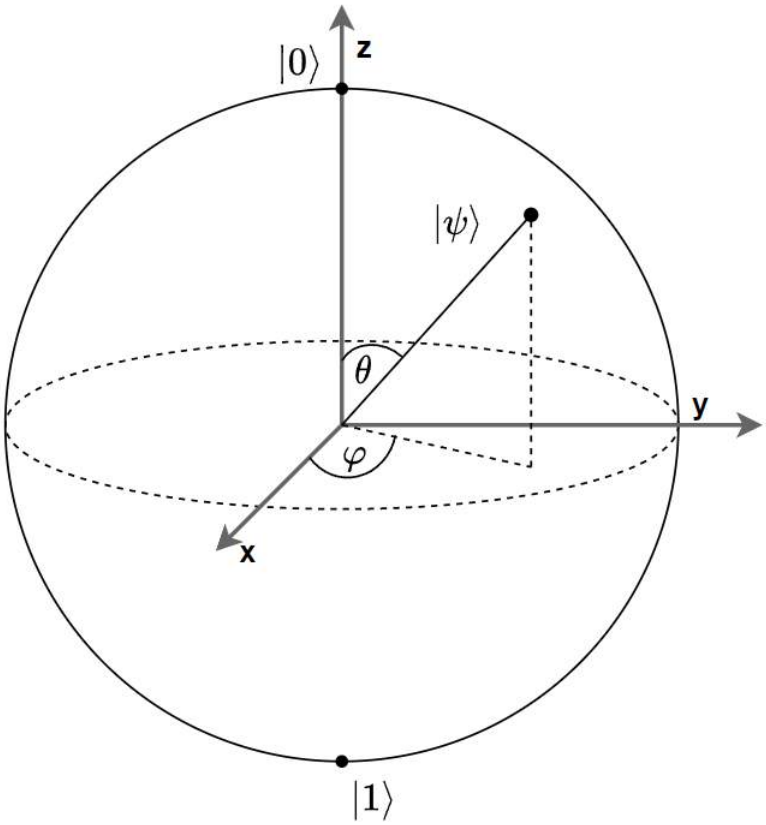


Figura 2.1: Representación de un qubit en una esfera de Bloch.

La figura 2.1 muestra la representación de un estado cuántico en una esfera de Bloch. Los valores en la superficie de la esfera representan un estado cuántico. Además, los valores antipodales (es decir, diametralmente opuestos) representan vectores de estado ortogonales.

Al interactuar con un sistema cuántico, mientras el qubit no se mide está en una superposición de ambos estados. En esta superposición, al realizar la medición para obtener el valor del qubit, $|\alpha|^2$ es la probabilidad de observarse el estado 0 y $|\beta|^2$ es la probabilidad de observarse el estado 1. El estado de un qubit es entonces la distribución de probabilidad de todas las posibles mediciones de un estado del sistema cuántico. Por lo tanto, es posible representar un bit usando un qubit, si α o β es 1.

Por otra parte, el fenómeno de *entanglement* o entrelazado es otra propiedad característica de los sistemas cuánticos. El entrelazado indica que cuando un par de qubits interactúan entre sí, se genera un estado cuántico en el cual el estado de uno no puede ser descripto sin tener en cuenta al otro. Cuando esto sucede, los estados de los qubits dependen entre sí, sin importar la distancia a

la que se encuentren separados.

La computación cuántica se basa en los postulados o axiomas 1, 2, 3 y 4 presentados a continuación. Este conjunto de postulados es explorado en más profundidad por Kaye, Laflamme y Mosca en el libro *An Introduction to Quantum Computing* [4].

El libro *Quantum Computation and Quantum Information*, de Nielsen y Chuang también explora estos postulados y brinda explicaciones acerca de su utilidad y explica que se llegó a ellos a través de un largo proceso de prueba y *pricipalmente* error. Explica además que estos postulados proveen una conexión entre el mundo físico y el formalismo matemático de la mecánica cuántica [7].

Postulado 1 (Espacio de Estados)**.** *El estado de un sistema cuántico puede ser descripto por un vector unitario en un espacio de Hilbert* $\mathcal{H}$*.*

El postulado 1 presenta una manera de describir el estado de un sistema cuántico. Por ejemplo, dado un estado con un qubit, es posible describirlo de la manera dada por la ecuación 2.5.

Postulado 2 (Evolución)**.** *La evolución temporal de un sistema cuántico cerrado puede describirse a través de un operador unitario. Esto implica que para cualquier evolución de un sistema cuántico cerrado, existe un operador unitario* U*, tal que si originalmente el sistema se encuentra en el estado* $|\psi_1\rangle$ *y posteriormente en el estado* $|\psi_2\rangle$*, entonces* $|\psi_2\rangle = U\,|\psi_1\rangle$*.*

El postulado 2 indica la manera en que un estado cuántico *evoluciona* a lo largo del tiempo. En particular, en el caso de un sistema de un qubit, cualquier operador unitario (ver definición en sección 2.1.1.3) es aplicable. Por ejemplo, en la sección 2.2.1 se definen varias compuertas en las que se aplican transformaciones a sistemas cuánticos.

Los vectores de estado de un sistema cuántico pueden representarse como vectores en un espacio de Hilbert y las compuertas cuánticas como transformaciones Hermíticas en dicho espacio [4]. Esta propiedad de los sistemas cuánticos, permitirá más adelante relacionar determinadas técnicas de computación cuántica, con el método del kernel empleado en SVMs.

Postulado 3 (Composición de Sistemas)**.** *Cuando dos sistemas cuánticos* H_1 *y* H_2 *son tratados como un único sistema, el espacio de estado de los sistemas combinados es* $H_1 \otimes H_2$*. Si el primer sistema se encuentra en el estado* $|\psi_1\rangle$ *y el segundo sistema en el estado* $|\psi_2\rangle$*, el estado combinado se encuentra en el estado* $|\psi_1\rangle \otimes |\psi_2\rangle$*.*

El postulado 3 además agrega la posibilidad de combinar sistemas y una manera de obtener el estado compuesto de ambos.

Postulado 4 (Medición)**.** *Dada una base ortonormal $B = \{|\phi_i\rangle\}$ de un espacio de estados H_A de un sistema A, es posible efectuar una medición de Von Neumann del sistema A con respecto a la base B. Esta medición, toma como entrada un estado $|\psi\rangle = \sum_i \alpha_i |\phi_i\rangle$ y retorna un valor i con probabilidad $|\alpha_i|^2$ y deja al sistema en el estado $|\phi_i\rangle$.*

A su vez, dado un estado compuesto $|\psi\rangle = \sum_i \alpha_i |\phi_i\rangle |\gamma_i\rangle$ en el espacio de estados $H_A \otimes H_B$ con $\{|\phi_i\rangle\}$ base ortonormal, pero $\{|\gamma_i\rangle\}$ unitaria no necesariamente ortonormal, una medición de Von Neumann del sistema A retornará i con una probabilidad $|\alpha_i|^2$ y deja al sistema $H_A \otimes H_B$ en el estado $|\phi_i\rangle |\gamma_i\rangle$.

En el desarrollo de los postulados 1, 2 y 3, se trabaja sobre sistemas cerrados. Sin embargo, a la hora de realizar un experimento, debe poder observarse qué está ocurriendo en ese sistema. El postulado 4 explica los efectos que una medición tiene en un sistema cuántico. La medición de un sistema cuántico, suele ser representada utilizando el símbolo 2.2.

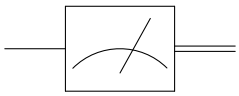


Figura 2.2: Símbolo de medición.

2.2.1. Compuertas cuánticas

Análogo al concepto de compuertas lógicas en la computación clásica, una compuerta cuántica es un circuito básico cuántico operando sobre una pequeña cantidad de qubits.

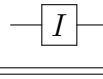
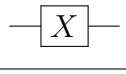
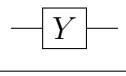
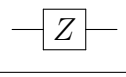
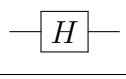
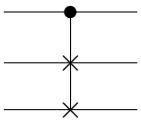
A continuación se presentan algunas de las principales compuertas cuánticas.

- Identidad: Definida para un único qubit mediante la matriz identidad, no modifica el estado cuántico.

- Compuertas de Pauli: Denifidas para un único qubit, mediante una de las matrices de Pauli (X,Y,Z). Representan una rotación sobre el eje X,Y o Z (según cuál matriz de Pauli se tome) de π radianes. En el caso de Pauli-X es el equivalente cuántico a la compuerta clásica NOT.

- Hadamard: Definida para un único qubit, aplica la matriz de Hadamard. Representa una rotación de π radianes sobre el eje $(X+Z)/\sqrt{2}$. La figura 2.3 muestra la transformación generada por esta compuerta, en el que el estado cuántico que originalmente está en posición $|1\rangle$ pasa a estar en posición $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ mientras que el $|0\rangle$ pasa a estar en posición $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$.
- CSWAP: Definida para tres qubits, aplica la matriz CSWAP. Dados los datos de ingreso (C, I_1, I_2) retorna (C, O_1, O_2) donde C es mapeado de manera directa, si C=0 entonces $I_1 = O_1$ y $I_2 = O_2$ y si C=1 entonces $I_1 = O_2$ y $I_2 = O_1$. La compuerta CSWAP entonces intercambia los valores de los qbits I_1 e I_2 si C se encuentra en el estado $|1\rangle$.

El cuadro 2.1 define las matrices y simbología de las compuertas cuánticas relevantes a este proyecto de grado.

Nombre	Matriz	Símbolo
Identidad	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	
Pauli X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	
Pauli Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	
Pauli Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	
Hadamard	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	
CSWAP	$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$	

Cuadro 2.1: Matrices y simbolos de compuertas cuánticas.

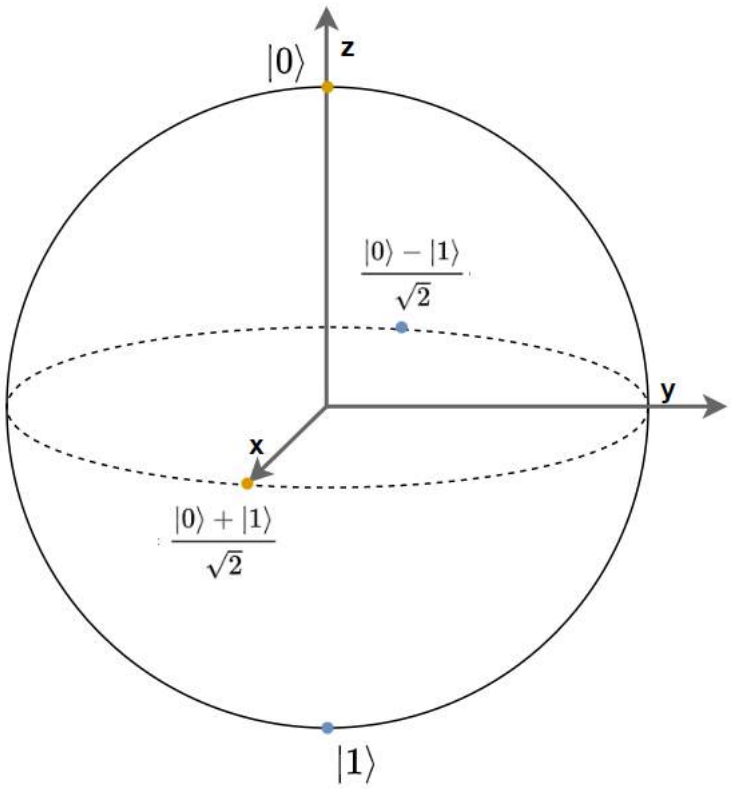


Figura 2.3: Transformación dada por Hadamard.

2.2.2. La rutina swap

La rutina swap es la rutina cuántica utilizada para calcular el producto interno entre dos estados cuánticos. En la sección 2.4.1 se utilizará la rutina swap como forma de obtener resultados de productos internos. Originalmente concebida como un método para distinguir entre dos estados cuánticos desconocidos [8], es un circuito que además puede utilizarse para estimar el producto interno entre dos vectores de estado cuánticos.

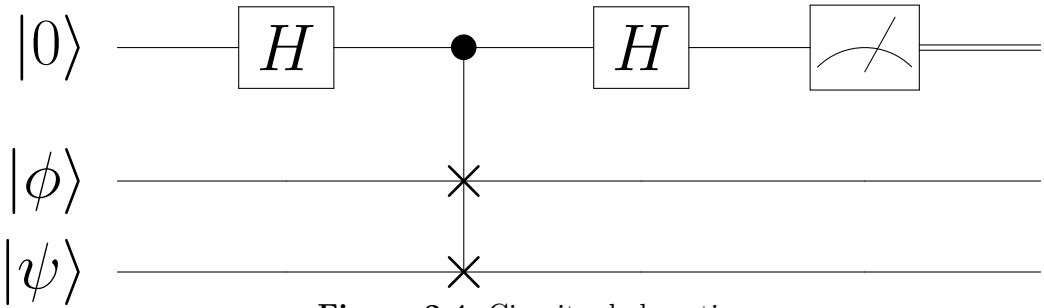


Figura 2.4: Circuito de la rutina swap.

En la figura 2.4 se observa el circuito de la rutina swap. En ella se muestra

el estado inicial del circuito y la aplicación de dos compuertas de Hadamard, una compuerta CSWAP y una medición. Los símbolos presentes en la imagen están definidos en la tabla 2.1, junto con el símbolo de medición dado por la figura 2.2.

Inicializando el qubit ancilla en $|0\rangle$, el estado inicial del sistema es $|0\phi\psi\rangle$. Luego de la primera compuerta Hadamard, el estado pasa a ser $\frac{1}{\sqrt{2}}(|0\phi\psi\rangle + |1\phi\psi\rangle)$. Aplicando la compuerta CSWAP, el estado pasa a ser $\frac{1}{\sqrt{2}}(|0\phi\psi\rangle + |1\psi\phi\rangle)$. Finalmente, aplicando la última compuerta Hadamard, el estado final del sistema es el dado por la ecuación 2.6. Esta ecuación por su parte puede reescribirse como la ecuación 2.7, utilizando la propiedad de asociatividad de la multiplicación de vectores.

$$\frac{1}{2}(|0\phi\psi\rangle + |1\phi\psi\rangle + |0\psi\phi\rangle - |1\psi\phi\rangle) \tag{2.6}$$

$$\frac{1}{2}|0\rangle(|\phi\psi\rangle + |\psi\phi\rangle) + \frac{1}{2}|1\rangle(|\phi\psi\rangle - |\psi\phi\rangle) \tag{2.7}$$

De la ecuación 2.7 utilizando el axioma de medición, se deduce que realizando una medición en la base computacional, la probabilidad de medir 0 es la dada por la ecuación 2.8.

$$\mathbb{P}(0) = \frac{1}{2}(\langle\phi\psi| + \langle\psi\phi|)\frac{1}{2}(|\phi\psi\rangle + |\psi\phi\rangle) \tag{2.8}$$

Teorema 1. *Sea K el número de ejecuciones de la rutina swap sobre un par de estados $|\phi\rangle, |\psi\rangle$ preparados independientemente para cada ejecución y M_i el resultado de la i-ésima medición en la base computacional. Entonces, $|\langle\psi|\phi\rangle|^2 = \lim_{K\to\infty} 1 - \frac{2}{K} \times \sum_{i=1}^{K} M_i$.*

Demostración. Sea $Ev = 0 \times \mathbb{P}(0) + 1 \times \mathbb{P}(1) = \mathbb{P}(1)$ el valor esperado de una medición del sistema cuántico luego de la ejecución de la rutina swap.

$$\begin{aligned}
&Ev = \mathbb{P}(1) \\
\Rightarrow &Ev = 1 - \mathbb{P}(0) \\
\Rightarrow &Ev = 1 - \frac{1}{2}(\langle\phi,\psi| + \langle\psi\phi|)\frac{1}{2}(|\phi\psi\rangle + |\psi\phi\rangle) \quad \text{(ecuación 2.8)} \\
\Rightarrow &Ev = 1 - \frac{1}{2}(\langle\phi|\langle\psi| + \langle\psi|\langle\phi|)\frac{1}{2}(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle) \\
\Rightarrow &Ev = 1 - (\frac{1}{4}(\langle\phi|\langle\psi||\phi\rangle|\psi\rangle + \langle\phi|\langle\psi||\psi\rangle|\phi\rangle + \langle\psi|\langle\phi||\phi\rangle|\psi\rangle + \langle\psi|\langle\phi||\psi\rangle|\phi\rangle))) \quad \text{(distributiva)} \\
\Rightarrow &Ev = 1 - (\frac{1}{4}(\langle\phi|\langle\psi||\phi\rangle|\psi\rangle + 1 + 1 + \langle\psi|\langle\phi||\psi\rangle|\phi\rangle)))
\end{aligned}$$

$$\Rightarrow Ev = 1 - (\frac{1}{4}\langle\phi|\langle\psi|\phi\rangle|\psi\rangle + \frac{1}{4}\langle\psi|\langle\phi|\psi\rangle|\phi\rangle + \frac{1}{4} + \frac{1}{4})$$
$$\Rightarrow Ev = 1 - (\frac{1}{4}\langle\phi|\langle\psi|\phi\rangle|\psi\rangle + \frac{1}{4}\langle\psi|\langle\phi|\psi\rangle|\phi\rangle + \frac{1}{2})$$
$$\Rightarrow Ev = 1 - (\frac{1}{4}\langle\psi|\phi\rangle\langle\phi|\psi\rangle + \frac{1}{4}\langle\phi|\psi\rangle\langle\psi|\phi\rangle + \frac{1}{2})$$
$$\Rightarrow Ev = 1 - (\frac{1}{4}\langle\psi|\phi\rangle\langle\phi|\psi\rangle + \frac{1}{4}\langle\phi|\psi\rangle\langle\psi|\phi\rangle + \frac{1}{2})$$
$$\Rightarrow Ev = 1 - (\frac{1}{4}\langle\phi|\psi\rangle\langle\phi|\psi\rangle + \frac{1}{4}\langle\phi|\psi\rangle\langle\psi|\phi\rangle + \frac{1}{2}) \quad \text{(prop conmutativa)}$$
$$\Rightarrow Ev = 1 - (\frac{1}{4}|\langle\phi|\psi\rangle|^2 + \frac{1}{4}|\langle\phi|\psi\rangle|^2 + \frac{1}{2})$$
$$\Rightarrow Ev = 1 - (\frac{1}{2}|\langle\psi|\phi\rangle|^2 + \frac{1}{2}) \quad \text{(sumando)}$$
$$\Rightarrow Ev = 1 - \frac{1}{2}|\langle\psi|\phi\rangle|^2 - \frac{1}{2}$$
$$\Rightarrow 2Ev = 1 - |\langle\psi|\phi\rangle|^2 \quad \text{(sumando y multiplicando por 2)}$$
$$\Rightarrow |\langle\psi|\phi\rangle|^2 = 1 - 2Ev$$
$$\Rightarrow |\langle\psi|\phi\rangle|^2 = 1 - 2 \times \lim_{K\to\infty} \frac{1}{K}\sum_{i=1}^{K} M_i \quad \text{(ley fuerte de los grandes números)}$$
$$\Rightarrow |\langle\psi|\phi\rangle|^2 = \lim_{K\to\infty} 1 - \frac{2}{K} \times \sum_{i=1}^{K} M_i$$

□

Del teorema 1 se deduce que el producto interno entre dos estados cuánticos arbitrarios puede aproximarse utilizando la rutina swap. Con múltiples ejecuciones de la rutina, se puede calcular una mejor aproximación a la probabilidad de realizar una medición de 1 en el qubit ancilla y esta probabilidad puede utilizarse para calcular el producto interno entre los estados cuánticos.

2.3. Aprendizaje automático

En esta sección se presentan los conceptos teóricos de aprendizaje automático utilizados en este trabajo.

El aprendizaje automático es una rama de la inteligencia artificial. Desde el punto de vista metodológico, se tiene un conjunto de datos de entrenamiento a partir de los cuales un algoritmo de aprendizaje automático construye un modelo. Una vez construído el modelo, podrá hacer predicciones o tomar decisiones sobre datos nuevos, estén o no estén en el conjunto de entrenamiento.

El aprendizaje puede ser supervisado o no supervisado. La diferencia radica en que en el aprendizaje supervisado se parte de un conjunto de datos de entrenamiento previamente etiquetado.

2.3.1. Máquinas de soporte vectorial

Las Máquinas de soporte vectorial (SVM) son un método de aprendizaje automático supervisado. Una SVM es un clasificador binario, que construye a partir de los datos brindados, un hiperplano en un espacio de dimensiones arbitrarias [9].

Se tiene como conjunto de entrenamiento n observaciones en un espacio p-dimensional, $x^1 = (x_1^1, ..., x_p^1)$,..., $x^n = (x_1^n, ..., x_p^n)$ donde cada una de ellas puede ser clasificada en $y^1, ..., y^n \in \{-1, 1\}$. Entonces, un SVM es un clasificador que si se le provee una observación x^*, de p dimensiones, desconocida, lo clasifica en una de los dos categorías posibles. Por tanto, se trata de hallar el hiperplano de mayor margen que divida el grupo de puntos en los que $y_i = 1$ de los que $y_i = -1$. Los hiperplanos pueden ser expresados como $w^T x - b = 0$ siendo w la normal al hiperplano y b una constante.

Si los datos son linealmente separables, es posible hallar un hiperplano en el que se busca que la distancia entre el hiperplano y los datos de cada lado sea la mayor posible. Para prevenir errores además se toman dos márgenes dados por las ecuaciones 2.9 y 2.10. Por lo tanto, se obtiene que $w^T x - b >= 1$, si y_i=1 y $w^T x - b <= -1$, si y_i=-1. Esto se puede reescribir como $y_i(w^T x - b) >= 1$.

$$w^T x - b = 1 \tag{2.9}$$

$$w^T x - b = -1 \tag{2.10}$$

Geométricamente, es posible demostrar que la distancia entre ambos márgenes es $\frac{2}{\|w\|}$. No se incluye esa demostración dado que escapa el alcance de este trabajo. Al mismo tiempo, se quiere *maximizar* la distancia entre ambos hiperplanos (2.9 y 2.10). Finalmente, el problema de optimización es *minimizar* $\|w\|$ sujeto a $y_i(w^T x_i - b) >= 1$.

En este trabajo no se profundiza en los métodos para obtener el hiperplano separador, pero si se mencionará es que la SVM toma el producto interno como medida de qué tan próximo dos vectores x, y se encuentran entre sí.

En la figura 2.5 se observa el funcionamiento básico de una SVM. Los puntos

verdes y naranjas representan los datos de dos clases de datos diferentes (A y B), que son linealmente separables. Entre ellos, se encuentran los vectores de soporte que generan la frontera de decisión.

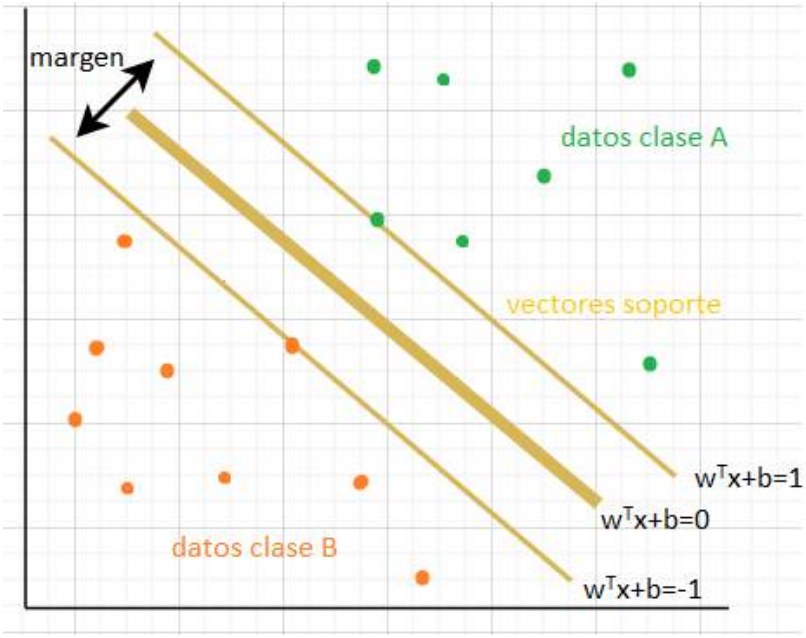


Figura 2.5: Representación gráfica de una SVM.

2.3.1.1. Extensiones multiclase

Un clasificador SVM es binario. Sin embargo, es habitual querer clasificar problemas que deben ser clasificados en más de dos categorías. Para ello, se busca transformar el problema en un problema multiclase usando alguna de las siguientes estrategias.

2.3.1.1.1. All Pairs Dado un problema de clase k, se entrenan $k\frac{(k-1)}{2}$ clasificadores binarios a partir de un conjunto de datos de entrenamiento correspondiente a dos clases .A momento de realizar la predicción, todos los clasificadores clasifican y para cada categoría se van sumando los resultados de los distintos clasificadores. El clasificador total se queda con la categoría que tenga la suma más alta.

2.3.1.1.2. One Against Rest Dado un problema de clase k, construir k clasificadores, donde el clasificador í-ésimo representa a la clase i, $\forall i \in \{1, .., n\}$. El clasificador que genere el valor más alto es seleccionado como el "ganador" y se selecciona a la clase dada.

2.3.1.2. Mapas de características

En la definición 22 se define formalmente el concepto de mapa de características.

Definición 22 (Mapa de características)**.** *Sea $\mathcal{F}$ un espacio de Hilbert, al que se llama espacio de características, $\mathcal{X}$ un conjunto de entradas y x un elemento de $\mathcal{X}$. Se define mapa de características como una función $\phi : \mathcal{X} \longrightarrow \mathcal{F}$ y se llama vectores de características a los vectores $\phi(x) \in \mathcal{F}$.*

Si bien el problema puede estar originalmente definido en un espacio de dimensiones finitas, puede ocurrir que sus datos no sean linealmente separables. Es por esto, que los atributos del problema son transformados a *características* en un *espacio de características*, un espacio de mayores dimensiones $\mathcal{F}$. De esta forma, se trabaja sobre las características de los atributos en lugar de los atributos en sí mismos y puede ser posible encontrar fronteras de decisión no lineales más complejas.

En la figura 2.6 se bosqueja el funcionamiento básico de un mapa de características. Se tienen datos de dos clases, una verde y otra amarilla. Estos datos no son linealmente separables $\mathbb{R}$, pero se le aplica una transformación mediante mapa de características que los transforma en un espacio de mayores dimensiones ($\mathbb{R}^2$, el espacio de características) en el que es posible encontrar un hiperplano separador (la línea naranja) en el que los valores de la clase verde estén por encima del hiperplano mientras que los valores de la clase amarilla están por debajo. Por lo tanto un problema que no era linealmente separable se transformó en uno que sí lo es.

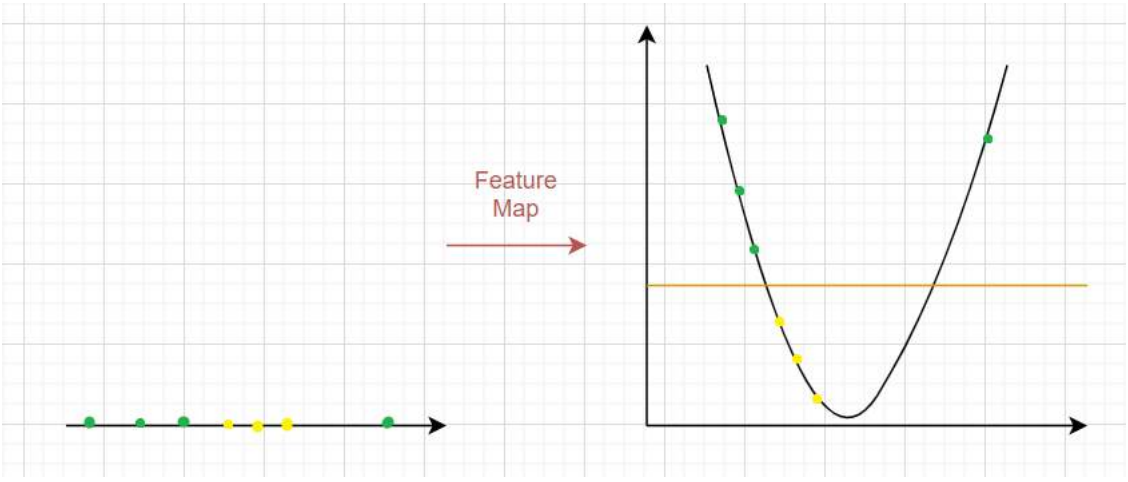


Figura 2.6: Transformación dada por un mapa de características.

2.3.1.3. Truco del kernel

El espacio de características elegido puede estar en un espacio vectorial de muy altas dimensiones, en el que computar el producto interno $\langle \phi(x), \phi(y) \rangle$ puede ser computacionalmente muy costoso. Por lo tanto, se introduce el *Truco del Kernel* o *Truco del Núcleo* en español. El truco del Kernel evita operar directamente en el espacio de características. El espacio de características está *implícito*. Esto se logra dado un kernel κ reemplazando la operación del producto interno $\langle \phi(x), \phi(y) \rangle$ por $\kappa(x, y)$ durante la ejecución del algoritmo SVM. La justificación formal de por qué reemplazar el producto interno por una función de Kernel es posible está presentada en el teorema 2.

Teorema 2. *Dado un mapa de características $\phi : \mathcal{X} \longrightarrow \mathcal{F}$, la función $\kappa : \mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{C}$ definida como $\kappa(x, y) = \langle \phi(x), \phi(y) \rangle_{\mathcal{F}}$, donde $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ es un producto interno definido sobre $\mathcal{F}$, es una función de kernel.[10].*

Como se mencionó en la subsección 2.3.1, SVM toma el producto interno entre dos vectores como medida de qué tan próximos son. Entonces como resultado del teorema 2 se tiene que utilizando una función de kernel sobre valores en $\mathcal{X}$, es posible tener una medida de qué tan próximos son los elementos en el espacio de características. Dado que un producto interno siempre es positivo-definido, entonces el kernel también lo es y por lo tanto kernel está definido según lo presentado en la sección 2.1.5.

Las definiciones 23, 24, 25 y 26 presentan algunos de los kernels más utilizados.

Definición 23 (Función de base radial - RBF)**.** *Sea x e y dos vectores en cierto espacio de entrada $\mathcal{X} = \mathcal{R}^p$, se define el kernel RBF como $K(x, y) = \exp\left(\frac{d(x,y)^2}{2\theta^2}\right)$ donde $d(x, y)$ es la distancia euclidiana y θ un parámetro de entrada [11].*

Definición 24 (Kernel polinomial)**.** *Sea x e y dos vectores en cierto espacio de entrada $\mathcal{X} = \mathcal{R}^p$ y d el grade del polinomio, se define el kernel polinomial como, $K(x, y) = (\gamma x^T y + c_0)^d$ donde c_0 y γ son parámetros [12].*

Definición 25 (Kernel Lineal)**.** *Se define el kernel lineal como un caso especial del kernel polinomial donde $d = 1$, $c_0 = 0$ y $\gamma = 1$, por lo tanto, $K(x, y) = (x^T y)$ [12].*

Definición 26 (Kernel del Coseno)**.** *Sea x e y dos vectores en cierto espacio de entrada $\mathcal{X} = \mathcal{R}^p$, se define el kernel del coseno como, $K(x, y) = \frac{xy^T}{\|x\|\|y\|}$ [12].*

2.3.2. Virtual Savant

Virtual Savant (VS) es un paradigma que se vale de técnicas del aprendizaje automático para resolver un problema determinado del cual puede no conocerse su comportamiento. Tiene ese nombre por el síndrome *savant*, una condición por la cual una persona tiene habilidades en un área (cálculos rápidos o memoria por ejemplo) muy superiores a la media, por medio del reconocimiento de patrones.

Basándose en un conjunto de instancias referentes a dicho problema y sus soluciones computados por un algoritmo de referencia, VS es capaz de resolver de manera eficiente nuevas instancias haciendo uso de los grandes beneficios que ofrece el paralelismo [13].

El uso de VS puede conseguir en algunos casos una mejor precisión que el algoritmo original cuando el problema a resolver es escalado, sin necesidad de realizar un entrenamiento adicional [14]. Este algoritmo de referencia es utilizado para entrenar un clasificador. Una vez entrenada, la inteligencia artificial es capaz de a partir de nuevas instancias del problema, conseguir soluciones con una precisión que dependerá del entrenamiento realizado.

VS consiste de dos importantes etapas, la clasificación y la mejora de la precisión de los resultados obtenidos. La etapa de clasificación consiste en utilizar múltiples instancias en paralelo de la inteligencia artificial entrenada previamente para conseguir una solución cuya estructura y datos dependerá del problema a resolver. Una vez completada esta fase, la segunda etapa es la encargada de procesar la solución obtenida anteriormente mediante el uso de heurísticas para conseguir un resultado con la más alta precisión posible.

2.3.3. Algoritmos Evolutivos

Los Algoritmos Evolutivos (AE) son algoritmos basados en los principios de la teoría de la evolución. Para ello, se tiene un conjunto de soluciones del problema que constituyen una población. A cada individuo de la población se le mide el fitness establecido con una función de fitness, se seleccionan los más aptos (es decir, los que tienen mejor fitness). Luego, se genera una nueva generación mediante la cruza de los más aptos y la mutación que introduce variaciones aleatorias y se elimina a los elementos de la generación anterior que tenían peor fitness. Se itera este proceso de manera que en el largo plazo la población mejore su fitness.

Dado que AE no lleva a una solución óptima, es necesario tener algún

criterio por el cual frenar la evolución. Para hacer esto hay dos enfoques, en primer lugar, frenar la iteración después de un tiempo de ejecución, mientras que la otra opción consiste en definir un mínimo de mejora entre generaciones.

En este trabajo no se implementará ningún AE, pero si se tomarán los resultados del AE implementado en el trabajo de doctorado *Learning for Optimization with Virtual Savant* [15].

2.4. Aprendizaje automático cuántico

El aprendizaje automático cuántico es un área de la computación que busca mejorar los algoritmos de aprendizaje automático mediante el uso de la computación cuántica. En este proyecto de grado sólo se discuten las máquinas de soporte vectorial cuánticas.

2.4.1. Máquinas de soporte vectorial cuánticas

Las SVM tienen el problema de que, cuando el espacio de características es muy grande, la función de kernel puede tornarse muy costosa en computar [16]. Es por ello que surgen las Máquinas de soporte vectorial cuánticas (QSVM), que son la versión cuántica de las SVM.

Las QSVM buscan capitalizar las propiedades cuánticas mencionadas en la subsección 2.2, como el entrelazamiento y la superposición, buscando obtener soluciones menos costosas computacionalmente que las obtenidas mediante el uso de SVMs clásicas.

Algo a tener en cuenta, sin embargo, es que para que una QSVM tenga un mejor desempeño que una SVM, el kernel debe ser difícil de computar. En caso contrario, el uso de una QSVM no presenta ventajas [16].

2.4.1.1. Métodos para implementar QSVMs

Existen dos maneras de implementar QSVMs según el trabajo *Supervised learning with quantum enhanced feature spaces* [16].

2.4.1.1.1. VQC Está basado en los circuitos cuánticos variacionales (VQC - Variational Quantum Circuits), un modelo universal de computación cuántica [17]. Este modelo utiliza circuitos cuánticos parametrizables y mediante mediciones del circuito, utiliza un algoritmo de optimización clásico para ajus-

tar estos parámetros y obtener el hiperplano que mejor separa a los datos de entrenamiento. No se profundizará sobre este método en este trabajo.

2.4.1.1.2. Estimación cuántica del kernel Consiste en utilizar una rutina cuántica que permita estimar el producto interno entre dos estados cuánticos, para lograr una estimación de la matriz de kernel. Esto permite utilizar un algoritmo de optimización cuántico para resolver el problema de optimización de una SVM convencional. La rutina swap, en conjunto con el teorema 1 da una forma de calcular dichos productos internos. Este es el método en el que se profundiza en este proyecto de grado.

2.4.1.2. Espacio de características cuántico

En las subsecciones 2.3.1.3 y 2.3.1.2 se definió kernel, mapa de características y espacio de características. En el caso de mapa de características cuánticos, el espacio de vectores F es un espacio de Hilbert en el cual sus vectores son estados cuánticos. El mapa de características entonces transforma los datos de entrada x a $|\psi_1\rangle$, un elemento cuántico del espacio de características.

El trabajo *Quantum machine learning in feature Hilbert spaces* estudia varias posibilidades sobre cómo implementar mapa de características para QSVMs y fue tomado como base para la investigación de mapa de características en la presente tesis [10].

En este trabajo se utilizarán dos vectores de características, Pauli y Product Enconding, que se introducirán en los párrafos 2.4.1.2.1 y 2.4.1.2.2. Adicionalmente, se consideraron los mapa de características Basis Encoding, Amplitude Encoding y Copies of Quantum States [10]. Basis encoding fue descartado por utilizar n qubits para representar una única característica, siendo n la cantidad de dígitos en binario de la característica representada en dicho sistema. Amplitude Encoding y consiguiente Copies of Quantum States, fue descartada por la necesidad de utilizar $\mathcal{O}(2^{n+1})$ compuertas cuánticas para su implementación, siendo n la cantidad de características a codificar [18].

2.4.1.2.1. Mapa de características de Pauli Dado un vector de entrada x, Pauli le aplica la transformación dada por la ecuación 2.11. En ella, la variable P_i denota una matriz de Pauli (ver el cuadro 2.1), mientras que el índice S denota conectividades entre diferentes qubits.

$$U\phi_x = \exp\left(i \sum_{S \leqslant [n]} \prod_{i \in S} P_i \phi_s(x)\right) \tag{2.11}$$

2.4.1.2.2. Mapa de características Product Encoding Dado un vector de entrada $x \in \mathbb{R}^N$ le aplica la transformación dada por la ecuación 2.12. El uso de este mapa de características implica el kernel del coseno (ver definición 26). Este mapa de características lo que hace es codificar tensorialmente cada característica del vector de entrada x según las amplitudes de un qubit separado [10].

$$U\phi_x = \begin{pmatrix} cosx_1 \\ senx_1 \end{pmatrix} \otimes \begin{pmatrix} cosx_2 \\ senx_2 \end{pmatrix} \otimes ... \otimes \begin{pmatrix} cosx_n \\ senx_n \end{pmatrix} \tag{2.12}$$

Capítulo 3

Implementación

En este capítulo se presenta la implementación de la solución generada en Python para la ejecución de los diferentes problemas a estudiar, junto a una introducción a las bibliotecas, técnicas y ambiente utilizados. Así mismo, se describen algunas técnicas de normalización de datos y se realiza una selección de los kernels y vectores de características a utilizar.

3.1. Ambiente y bibliotecas

En esta sección se menciona cómo se generó el código, con el uso de cuáles bibliotecas y el ambiente en el que se ejecutó.

3.1.1. Ambiente utilizado

El lenguaje de programación utilizado en el trabajo es Python y con el objetivo de garantizar que el código se ejecutara de manera correcta, se utilizó el manejador de paquetes y ambientes para Python llamado Conda. De esta manera, es posible crear ambientes con todas las dependencias a utilizar en una versión especifica. Así mismo, buscando automatizar la instalación de paquetes, se creó un archivo de tipo bash que crea el ambiente e instala los paquetes necesarios en la versión requerida.

3.1.2. Bibliotecas utilizadas

En esta sección se describirán las principales bibliotecas utilizadas.

- Numpy es una biblioteca de Python que provee varias operaciones multidimensionales en arreglos y matrices, fue utilizada para la manipulación

de datos [19].

- Qiskit corresponde a un entorno de trabajo de código abierto desarrollado por IBM para operar en computación cuántica [20]. Se destaca a su vez, por permitir el acceso a la IBM Quantum Experience. En este proyecto fue utilizado para generar las QSVM, en particular su componente Aqua, que permite desarrollar aplicaciones cuánticas sin tener que realizar implementaciones a bajo nivel.

- Scikit-learn es una biblioteca especializada en algoritmos de aprendizaje automático, fue utilizada en este proyecto de grado para la implementación de las SVMs clásicas y el kernel del coseno con *cosine-similarity*, así como también para importar y manipular los conjuntos datos [21]. A su vez, se utilizó para las diferentes implementaciones realizadas, una herramienta que forma parte de Scikit-learn llamada Standard Scaler, que permite normalizar los valores de cada una de las variables, así también llamadas *características*, de los problemas resueltos. Dichas variables se normalizan de forma independiente antes de aplicar el método de SVM, eliminando la media y escalando los datos de forma que su varianza sea igual a uno.

- Multiprocessing es una biblioteca que se utilizó para implementar el paralelismo en Python como parte del método de Virtual Savant para el problema de sincronización de líneas de ómnibus descripto en la sección 4.3. La misma permite que la predicción de los tiempos para cada nodo se realice de forma paralela mediante un *pool* con una cantidad determinada de procesadores [22].

3.1.3. Paralelización en Python

Como se mencionó en la subsección 3.1.2, en este trabajo se utilizó la técnica de multiprocesamiento para la paralelización. Por lo general, para paralelizar existen dos estrategias posibles: crear hilos o utilizar técnicas de multiprocesamiento.

En el caso de Python, dado el uso de GIL [23] (lock del interpretador global por sus siglas en inglés), el cual sólo permite que un hilo pueda estar ejecutando al mismo tiempo, el uso de hilos no mejora el desempeño. Por lo tanto, la única opción disponible era utilizar técnicas de multiprocesamiento. De esta manera,

para cada procesador asignado de la CPU se ejecuta la función que se encarga de predecir los resultados de la SVM o de la QSVM según corresponda.

3.2. Simulación de circuitos cuánticos

En esta sección se presentan los aspectos técnicos que fueron considerados para implementar circuitos cuánticos.

Qiskit provee varias clases para la ejecución de un sistema cuántico llamados *backends*. Existen dos interfaces para los backends, Aer e IBMQ. Aer permite simular circuitos cuánticos y ejecutarlos en la máquina local mientras que IBMQ permite acceso a simuladores en la nube provistos por IBM. En este proyecto de grado se utilizaron backends provistos por la interfaz Aer de manera de poder realizar ejecuciones locales.

Para este proyecto se evaluó el uso del backend *qasm_simulator*, un simulador que automatiza múltiples ejecuciones de circuitos cuánticos y provee distintos modelos de fuentes de errores. Es importante notar que durante la ejecución de los experimentos, no se utilizó la funcionalidad de modelado de errores del simulador, por no ser este el objetivo del trabajo.

La principal razón para la elección del simulador *qasm_simulator* fue la facilidad para multiples ejecuciones de circuitos cuánticos, con el fin de obtener mejores aproximaciones a un resultado ideal. Duarante el testeo de los distintos prototipos se utilizó el backend *statevector_simulator*, por ser el simulador más veloz para la ejecución de una QSVM, ya que es un simulador de única ejecución de los circuitos cuánticos. El backend *statevector_simulator* presenta como desventaja que utiliza vectores de estado de tamaño 2^N donde N es el número de qubits. Debido a que la dimensión del vector de estado crece exponencialmente, no se utilizó este backend.

Dado que se trabajó con simulaciones que tienen componentes pseudoaleatorios, es necesario definir una semilla aleatoria para poder garantizar la reproducibilidad de los experimentos. En particular, qiskit provee formas de inicializar las semillas aleatorias del simulador y transpilador de circuitos cuánticos. Estas semillas serán referidas como *simulator_seed* y *transpiler_seed* respectivamente. En casos donde se utilizó una semilla aleatoria con otros propósitos, se referirá simplemente como *semilla*.

A lo largo del proyecto de grado se utiliza el término *shot* (ejecución), para denotar una ejecución de un circuito cuántico, por lo que el número de shots de un experimento es el número de ejecuciones independientes de los circuitos

cuánticos del experimento.

3.3. Manipulación de datos

A lo largo del presente proyecto, se volvió necesaria la utilización de técnicas de manipulación y escalamiento de datos para cada uno de los problemas estudiados. En esta sección se describen las técnicas que fueron utilizadas en el presente proyecto de grado junto con las razones para su utilización.

3.3.1. Normalización de datos

Las máquinas del estilo SVM y QSVM asumen que los datos operan en un rango estándar, por ejemplo, que todos los datos se encuentran en el rango -1 y 1. Sin embargo, esta suposición no representa la realidad de los datos con los que se operan en los problemas estudiados, lo cual puede provocar que si una característica del vector tiene valores muy diferentes a los demás, la SVM le asigne demasiada (o demasiada poca) importancia. Por lo tanto, se aplican técnicas de normalización de datos para escalar los datos y normalizarlos.

3.3.2. Reducción de la dimensionalidad

Dependiendo del mapa de características elegido, existen limitaciones a la cantidad de qubits según la dimensión del problema. En el caso particular de utilizar el kernel del coseno, la cantidad de qubits debe ser igual a la dimensión del problema. Por lo tanto, se vuelve necesario implementar técnicas de reducción de dimensionalidad para poder trabajar con instancias de problemas de grandes dimensiones con pocos qubits. En particular, se aplica la técnica de análisis de componentes principales, o así también llamada por sus siglas, PCA.

3.3.2.1. Análisis de componentes principales

Análisis de componentes principales (PCA), es un método de reducción de dimensionalidad de un problema concentrándose en analizar cuáles son sus componentes principales, a partir de las covarianzas que presentan. De este manera, un problema que originalmente tenía n dimensiones, puede trabajarse en m dimensiones con $m < n$ [24].

Para definir PCA, es necesario definir previamente algunos conceptos estadísticos, presentados en las definiciones 27, 28, 29 y 30. Además, se utilizaron conceptos de álgebra lineal introducidos en 2.1.1.4.

Definición 27 (Esperanza)**.** *La esperanza E[X] es una generalización del promedio ponderado. Representa el valor esperado de una ejecución. En el caso discreto, dada una variable aleatoria X con función de probabilidad* $P[X = x_i]$*, entonces* $E[X] = \sum_{i=1}^{n} x_i P[X = x_i]$*.*

Definición 28 (Varianza)**.** *La varianza mide la dispersión de una variable aleatoria. Se calcula como* $Var(X) = E[X^2] - E[X]^2$*.*

Definición 29 (Covarianza)**.** *Sean X e Y dos variables aleatorias. Se define la covarianza entre ellas como* $Cov(X, Y) = E[XY] - E[X]E[Y]$*.*

Definición 30 (Matriz de covarianza)**.** *Sea X un vector aleatorio, tal que* $X = [X_1, ..., X_n]$*, donde la í-ésima entrada de X es una variable aleatorio con varianza finita. Entonces, la matriz de covarianza* $\sum$ *es la dada por la ecuación 3.1.*

$$\sum = \begin{bmatrix} Var(X_1) & Cov(X_1, X_2) & \dots & Cov(X_1, X_n) \\ Cov(X_2, X_1) & Var(X_2) & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_n, X_1) & Cov(X_n, X_2) & \dots & Var(X_n) \end{bmatrix} \tag{3.1}$$

En primer lugar, deben ser definidos los valores y vectores propios a utilizar los componentes principales. El objetivo de PCA es comprimir la información en los primeros componentes. Es decir, si se tiene un conjunto de datos inicial con dimensión 8, se tendrán 8 componentes principales, pero PCA buscará condensar la información en la primera. De esta manera es posible descartar las últimas y así reducir la dimensionalidad.

Dado que PCA busca condensar la información en la primera componente, PCA define como primer componente a aquella que presente la mayor varianza. Es por esta razón que previo a reducir la dimensionalidad de un conjunto mediante de datos utilizando PCA sea necesario normalizar los datos.

Dado que los vectores propios de la matriz de covarianza representan la dirección de los ejes en los que existe mayor varianza, es necesario calcularlos. Los valores propios de la matriz de covarianza representan la cantidad de

varianza contenida en cada componente principal. Entonces, al ordenar los vectores propios según sus valores propios asociados se obtienen los componentes principales ordenados.

Una vez obtenidos los vectores propios, se construye una matriz de características en la que las columnas son los vectores propios de la matriz de covarianza. En esta matriz se elige la nueva dimensionalidad que tendrán los datos al descartar algunos de los vectores propios. Finalmente, sea D el nuevo conjunto de datos, N el conjunto de datos original normalizado y M la matriz de características, entonces $D = M^T \times N^T$. En este proyecto de grado no se implementó PCA, si no que se tomó la implementación de la biblioteca sklearn.

Capítulo 4

Evaluación experimental

En este capítulo se plantean tres diferentes problemas con el objetivo de poder comparar el desempeño de SVM y de QSVM. En primer lugar, se proponen dos problemas de clasificación, de manera tal de generar prototipos de SVM y QSVM. Luego se procede a evaluar el desempeño de ambas estrategias para un problema de optimización complejo sobre la sincronización de líneas de ómnibus mediante el uso del paradigma Virtual Savant.

4.1. Clasificación de tumores mamarios

El primer problema a estudiar en este proyecto de grado es un problema de clasificación de tumores mamarios para detectar si los tumores son benignos o malignos, por lo que en esta sección se presenta una descripción del problema y de la metodología de evaluación.

4.1.1. Descripción

La clasificación de tumores mamarios es un problema basado en datos reales de cáncer de mama publicados por la Universidad de Wisconsin [25] y que la bilbioteca sklearn provee.

El cáncer de mama es el principal cáncer que ocurre en mujeres, por lo cual es de interés común encontrar maneras de prevenirlo. En particular en este caso, el doctor Wolberg de la Universidad de Wisconsin, tomó muestras de fluidos de 370 mujeres para obtener características celulares de cada una de ellas. A partir de allí, detectó que había una correlación importante entre algunas características celulares y si la persona tenía un tumor benigno o maligno.

Las características del núcleo celular a partir de la cual se realiza la clasificación son: radio, textura (medida como la desviación estándar de valores en escala de grises), perímetro, área, suavidad, compacidad, concavidad, cantidad de puntos cóncavos, simetría y dimensión fractal.

Dado que el problema de clasificación de tumores mamarios consta de clasificar en benignos o malignos, es un problema de clasificación binario. Por lo tanto, es posible resolverlo mediante el uso de una SVM [26].

4.1.2. Metodología de evaluación

Para la evaluación experimental se utilizó un conjunto de entrenamiento de 30 instancias y un conjunto de validación de 110 instancias. Para dividir entre datos de validación y entrenamiento, se utilizó la función *train_test_split* de SkLearn, con valor de semilla 109 para garantizar la reproducibilidad. La SVM clásica se ejecutó utilizando el kernel lineal y el kernel del coseno, mientras que la cuántica se ejecutó con el mapa de características de Pauli y el mapa de características Product Encoding. La QSVM se ejecutó con varias semillas S y T, definidas en la sección 3.2 de manera de obtener veinte resultados con diferentes semillas, para luego analizar el mejor caso, el peor caso y el caso promedio y compararlos con los obtenidos por la SVM.

Para resolver el problema de clasificación de tumores mamarios, se utilizó el conjunto de datos provisto por sklearn y luego se procedió a realizar el escalamiento de los datos para poder manipularlos. Se utilizó el método StandardScaler de sklearn para poder normalizar los datos previo al uso de PCA. Luego se realizó una reducción de dimensionalidad de instancias utilizando la técnica PCA para reducir a dos características y se procedió a utilizar *MinMaxScaler* para normalizar y que todos los valores del problema estuvieran entre -1 y 1. Una vez normalizados los datos, se procedió a reemplazar las etiquetas de las instancias del problema, dejando cero en el caso de tumor maligno y un uno en el caso de un tumor benigno.

Una vez completada la etapa de manipulación de datos, se procedió a trabajar con la SVM y la QSVM. En una primera instancia se las entrenó con los datos definidos como de entrenamiento y al finalizar el entrenamiento se pasó a la fase de predicción.

En la figura 4.1 se presenta el conjunto de datos del problema de los tumores mamarios con su dimensionalidad reducida de la forma descripta anteriormente y con sus datos normalizados entre -1 y 1.

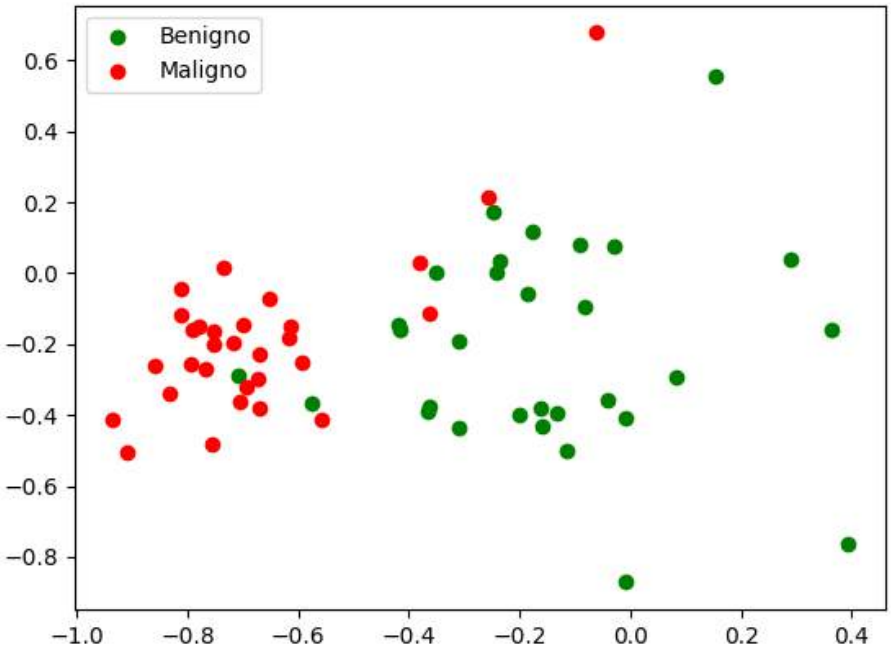


Figura 4.1: Conjunto de datos del problema de tumores mamarios con dimensionalidad reducida.

4.2. Problemas de clasificación de flores iris

Esta sección describe el problema de clasificación de flores iris. Se describe el problema y la metodología de evaluación.

4.2.1. Descripción

Las flores iris son un conjunto de aproximadamente 300 especies de plantas floreadas. En este conjunto de datos, se intenta clasificar los datos de tres tipos diferentes de flores iris. Para cada una de las flores, se tienen las siguientes variables: largo del pétalo en centímetros, ancho del pétalo en centímetros, largo del sépalo en centímetros, ancho del sépalo en centímetros. El conjunto de datos consta de ciento cincuenta flores, cincuenta de cada especie (iris setosa, iris virginica e iris versicolor) y los datos de largo y ancho de sus pétalos y sus sépalos

Si bien el problema de clasificación de flores iris es un problema con un conjunto de datos más reducido que el problema de clasificación de tumores

mamarios, al ser categorizable en tres categorías, es un problema de clasificación multiclase. Dado que SVM y QSVM son técnicas de clasificación binarias, para clasificar las flores iris es necesario utilizar extensiones multiclase definidas en la sección 2.3.1.1.

4.2.2. Metodología de evaluación

Para la evaluación experimental del problema de clasificación de flores iris se utilizó un conjunto de entrenamiento de 50 instancias y un conjunto de validación de 100 instancias. Para dividir entre datos de validación y entrenamiento, se utilizó la función *train_test_split* de SkLearn, con valor de semilla 109 para garantizar la reproducibilidad. Para el caso de la SVM se predijo una vez y se calculó la tasa de acierto entre los valores predichos y los verdaderos. En el caso de la QSVM, dada su pseudoaleatoriedad, se ejecutó con varios valores de semilla para obtener varios resultados diferentes. Luego, se calculó la media, mediana y desviación estándar de los resultados y se los comparó con los resultados obtenidos por la SVM.

La evaluación experimental del problema de clasificación de flores iris se realizó en el caso de SVM con cuatro kernels diferentes (lineal, coseno, RBF y polinomial de grado tres) y en el caso de QSVM con dos mapas de características, Pauli-Z y Product Encoding.

Para resolver el problema de clasificación de flores iris, se importó el conjunto de datos provisto por sklearn. El conjunto de datos es escalado en primer lugar, ya que para aplicar PCA es recomendado que los datos sean normalizados previamente como se mencionó en la subsección 3.3.2.1. Luego, se realizó una reducción de la dimensionalidad mediante el uso de la técnica PCA, para poder utilizar una menor cantidad de qubits en la QSVM.

Mediante el uso de MinMaxScaler provisto por sklearn, se transformaron los datos para que sus valores estuvieran entre -1 y 1. De esta manera, los datos tuvieron su dimensionalidad reducida, de manera tal de poder trabajar con menos qubits en la QSVM y fueron normalizados, de manera tal de poder ser clasificados por una SVM y QSVM. Además, se modificaron las etiquetas que tienen los vectores, dejando 0 para *iris setosa*, 1 para *iris versicolor* y 2 para *iris virginica*.

Una vez realizada la manipulación de los datos, se procedió a entrenar la SVM y la QSVM. Dado que el problema de clasificación de flores iris es un problema de clasificación en tres categorías, es necesario utilizar extensio-

nes multiclase como las introducidas en la subsección 2.3.1.1 , ya que SVM y QSVM son clasificadores binarios. En el caso de clasificación de flores iris, se definió utilizar la estrategia AllPairs ya que al ser un problema de una dimensión pequeña. Al contar con solo 4 atributos y aproximadamente 150 ejemplos en el conjunto de datos, no es demasiado costoso construir más clasificadores. Una vez entrenada la QSVM y SVM, se procede a la clasificación.

En la figura 4.2 se presenta el conjunto de datos del problema de clasificación de flores iris con su dimensionalidad reducida de la manera descripta en los anteriores párrafos y con sus datos normalizados entre -1 y 1.

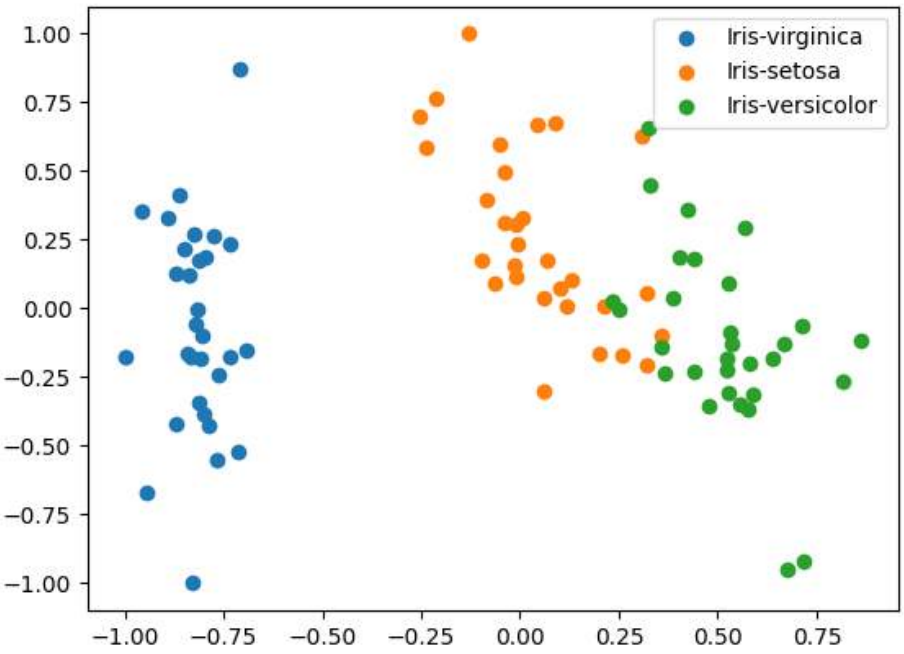


Figura 4.2: Conjunto de datos del problema de clasificación de flores de iris con dimensionalidad reducida.

4.3. Problema de sincronización de líneas de ómnibus

Esta sección describe el Problema de sincronización de líneas de ómnibus (BSP). Se describe el problema y la metodología de evaluación.

4.3.1. Descripción

El último problema abordado en este proyecto de grado es una versión modificada del problema de sincronización de líneas de ómnibus [15], así tambien llamado BSP, que consiste en coordinar los horarios de los ómnibus con el objetivo de optimizar el *headway* o tiempo entre salidas consecutivas de ómnibus de una línea y así disminuir los tiempos de espera de pasajeros. Se plantea como objetivo hallar los headways de cada línea, es decir, el tiempo entre dos viajes consecutivos de una línea. Para este problema se asume que todas las paradas de ómnibus son potencialmente estaciones de trasbordo y que un trasbordo se puede hacer en más de una parada, con el pasajero caminando entre una y la otra. Por lo tanto, las estaciones de trasbordo son zonas de trasbordo a las cuales se les asigna el nombre de nodos de sincronización.

En este proyecto de grado se tomará como referencia el trabajo de doctorado *Learning for Optimization with Virtual Savant* [15].

4.3.1.1. Formulación matemática

El problema [15] consta de los siguientes elementos:

- Un conjunto $I = \{i_1, i_2, ..., i_n\}$ de líneas de ómnibus y un conjunto $J(i)$ de líneas con las que se puede hacer trasbordo.
- Un conjunto $B = \{b_1, b_2, ..., b_n\}$ de nodos de sincronización. Cada nodo $b \in B$ es una tupla triple $\langle i, j, d_b^{i,j} \rangle$ que indica que en el nodo b se puede hacer trasbordo entre las líneas i y j y que las paradas de cada línea están separadas por una distancia $d_b^{i,j}$.
- Un período de planeamiento $[0, T]$ y el número de viajes de cada línea i, f_i, necesario para satisfacer la demanda.
- Una función de tiempo de viaje $TT : I \times I \times B \rightarrow Z, TT_b^i = TT(i, b)$, que indica el tiempo que los ómnibus de la línea i demoran en alcanzar el nodo de sincronización b desde el punto de inicio de la línea.
- Una función de tiempo caminado $WT : I \times I \times B \rightarrow N, WT_B^{i,j} = WT(i, j, b)$, que indica el tiempo que un pasajero camina de una parada a otra a una velocidad ws.
- Una tiempo de espera máxima $W_b^{i,j}$ para cada nodo de sincronización, que indica el tiempo máximo que un pasajero estaría dispuesto a esperar

por la línea j luego de haber bajado de la línea i y caminado a la parada de la línea j.

- Un rango válido de tiempos de espera o *headways* que miden el tiempo entre dos viajes consecutivos de una línea. El rango válido para la línea i está definido por $[h_i, H^i]$.
- La hora de partida del viaje r de la línea i se representa como X_i^r
- La capacidad máxima de transferencia de los ómnibus se representa como C
- Se utiliza una variable binaria $Z_{i,j}^{r,s,b}$ que indica si el viaje r de la línea i y el viaje s de la línea j están sincronizados o no en el nodo b.

Por lo tanto, se propone como función objetivo la función descrita en la ecuación 4.1a. La función objetivo busca maximizar el número de transferencias sincronizadas, ponderando por la demanda de transferencias de cada viaje en cada nodo de sincronización. La demanda a su vez está dividida entre los f_i viajes de la línea i, lo cual es razonable siempre y cuando la demanda no varíe demasiado en el período de planeamiento T. Además, el número de pasajeros está acotado por la capacidad de transferir de los ómnibus C.

$$\text{maximizar: } \sum_{b\in B}\sum_{i\in I}\sum_{j\in J(i)}\sum_{r=1}^{f_i}\sum_{s=1}^{f_j} Z_{r,s,b}^{i,j} \times \min(\frac{P_b^{i,j}}{f_i}, C) \tag{4.1a}$$

$$\text{sujeto a: } X_1^i \leqslant H^i \tag{4.1b}$$

$$T - H^i \leqslant X_{f_i}^i \leqslant T \tag{4.1c}$$

$$h^i \leqslant X_{r+1}^i - X_r^i \leqslant H^i \tag{4.1d}$$

$$(X_s^j + TT_b^j) - (X_r^i + TT_b^i) > WT_b^{i,j} \text{ si } Z_{r,s,b}^{i,j} = 1 \tag{4.1e}$$

$$(X_s^j + TT_b^j) - (X_r^i + TT_b^i) \leqslant W_b^{i,j} + WT_b^{i,j} \text{ si } Z_{r,s,b}^{i,j} = 1 \tag{4.1f}$$

$$X_r^i - X_{r-1}^i = X_s^i - X_{s-1}^i \forall r, s : r > 1, s > 1 \tag{4.1g}$$

$$X_r^i \in \{0, \ldots, T\}, Z_{r,s,b}^{i,j} \in \{0, 1\} \tag{4.1h}$$

En cuanto a las restricciones, la ecuación 4.1b indica que el primer viaje de cada línea comienza antes que la cota mayor de headways de la línea. En tanto, la restricción dada por la ecuación 4.1c indica que el último viaje debe terminar antes que T. La ecuación 4.1d garantiza que el headway computado de cada línea esté dentro del rango de headways válidos para esa línea. Las restricciones

dadas por las ecuaciones 4.1e y 4.1f aseguran que el viaje r de la línea i y el viaje s de la línea j están sincronizadas en el nodo b si los pasajeros son capaces de realizar trasbordos, considerando el tiempo de caminar entre las paradas y el tiempo máximo que los pasajeros están dispuestos a esperar. La ecuación 4.1g indica que los headways para una línea de ómnibus son constantes en el período de planeamiento. Por último, la ecuación 4.1h refleja el dominio de variables de decisión del problema [15].

4.3.1.2. Generación de instancias

Las instancias utilizadas en este proyecto de grado fueron tomadas del trabajo de doctorado *Learning for Optimization with Virtual Savant* [15]. En esta sección se describe cómo fueron generadas.

Las instancias fueron generadas utilizando datos reales. Las fuentes utilizadas fueron los datos de ventas de boletos con tarjeta STM de 2015 durante el horario 12:00-14:00 e información abierta de líneas de ómnibus, paradas de ómnibus y horarios.

Las instancias fueron definidas con tres tamaños posibles, según la cantidad de nodos de sincronización. Los tres tamaños de instancia generados fueron treinta, setenta y cientodiez. Los nodos de sincronización fueron elegidos de entre los cientosetenta nodos con mayor demanda y seleccionados al azar con distribución uniforme. La demanda de cada nodo de sincronización $P_b^{i,j}$ fue asignada según la demanda real en mayo de 2015, la capacidad máxima de transferencia C fue fijada en cinco en todas las instancias. El tiempo de caminata entre paradas $WT_b^{i,j}$ se calculó según la distancia de paradas y asumiendo una velocidad constante de caminata de $6km/h$. El tiempo de viaje de las líneas al nodo de sincronización TT_b^i, TT_b^j fueron calculados utilizando los horarios publicados de las líneas de ómnibus. El rango de headways válidos $[h^i, H^j]$ fue calculado usando los horarios publicados de las líneas de ómnibus también. El tiempo de espera máximo para cada nodo de sincronización fue definido según el headway máximo para la línea de salida $WT_b^{i,j} = \lambda H^j$, con $\lambda \in \{\, 0,7; 0,9; 1,0 \,\}$[15].

La figura 4.3 muestra la información topológica de la instancia trabajada. En naranja se muestran las paradas de ómnibus y en verde los nodos de sincronización. Esta imagen está basada en una imagen similar presente en el trabajo de doctorado *Learning for Optimization with Virtual Savant* [15].

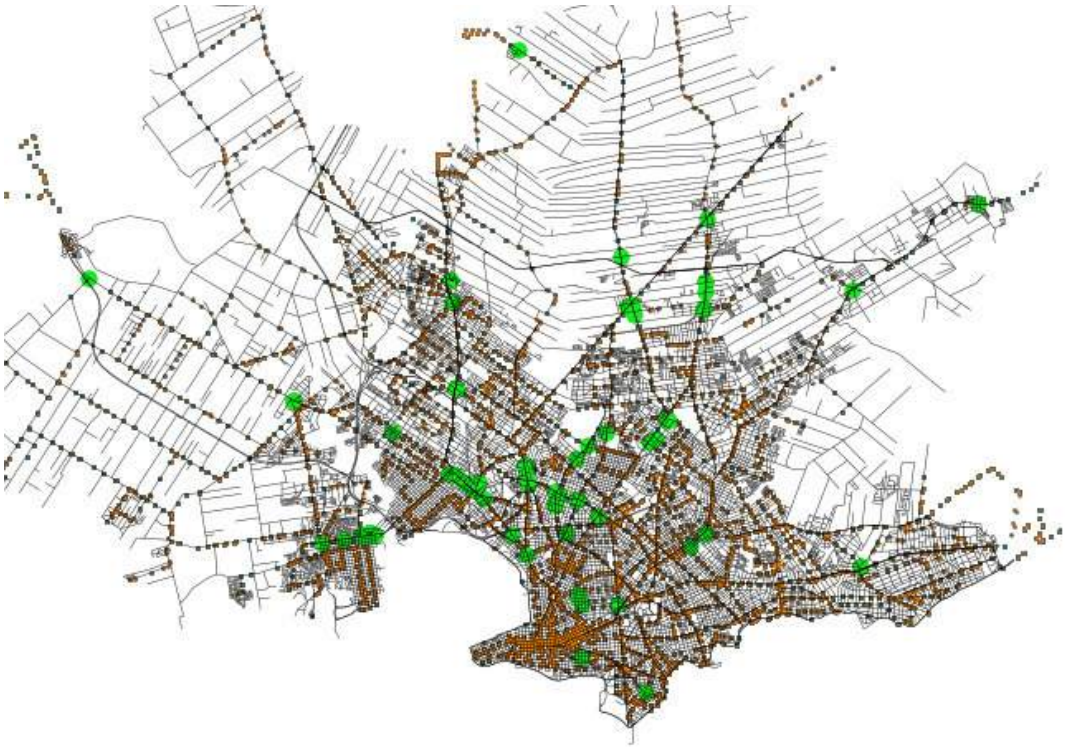


Figura 4.3: Mapa de paradas de ómnibus de Montevideo.

4.3.2. Metodología de evaluación

Dada la mayor complejidad del problema BSP en comparación con el problema de clasificación de flores iris y la clasificación de tumores mamarios, se recurrió a diversas técnicas de evaluación de resultados. Las instancias del problema están divididas según su tamaño (30, 70 y 110) y para cada tamaño se ejecutó la SVM y la QSVM.

El problema BSP presenta una mayor complejidad en comparación con el problema de clasificación de flores iris y con el problema de clasificación de tumores mamarios. Por ello, no se hizo un análisis de resultados de la misma instancia con diferente semilla, sino que se comparó para cada instancia el fitness obtenido y la precisión una sola vez y luego se realizó un análisis de la solución promediando los resultados.

4.3.2.1. Función de fitness

Para medir la calidad de la solución generada se utiliza una función de fitness. Previo a calcular el fitness se necesita obtener para cada nodo de sincronización sus características, es decir: tiempo de viaje para líneas entrantes

y salientes (TT_b^i, TT_b^j), tiempo caminando entre paradas ($WT_b^{i,j}$), demanda de pasajeros ($P_b^{i,j}$)y tiempo de espera máximo $W_b^{i,j}$. Asimismo, deben obtenerse los headways para todas las líneas entrantes y salientes e inicializarse el fitness en 0.

Una vez obtenidas todas las características, para calcular el fitness se itera sobre cada par de líneas entrantes y salientes y para cada uno se calcula el tiempo de espera. El tiempo de espera está definido como la diferencia de tiempo entre que la línea entrante llega al nodo y la línea saliente se va del nodo, menos el tiempo que demora la persona en llegar caminando a la parada. En el peor de los casos el usuario deberá esperar en el nodo de sincronización el tiempo completo de headway de la línea. Dos líneas se consideran sincronizadas si el tiempo calculado es menos al tiempo máximo de espera. En el caso de que las líneas estén sincronizadas, se suma al resultado del fitness la demanda acumulada según la función objetivo, definida por la ecuación 4.1a. En el algoritmo 1 se presenta una implementación en pseudocódigo del algoritmo utilizado para calcular el fitness descripto anteriormente.

4.3.2.2. Algoritmos de comparación

Con el objetivo de comparar los resultados obtenidos utilizando QSVM, se ejecuta el problema utilizando otros algoritmos. Para ello, fueron utilizados un algoritmo de tipo greedy y una SVM clásica. Además se cuenta con resultados de un algoritmo evolutivo, que se utiliza como referencia y se calcula el gap porcentual entre los resultados de fitness dado por la SVM, la QSVM y el algoritmo Greedy en comparación con el AE. [15].

4.3.2.3. Validación cruzada

Para la evaluación se utilizó como técnica de validación la validación cruzada con tres iteraciones, la cual garantiza que los resultados de la evaluación experimental sean independientes de la partición entre datos de entrenamiento y datos de validación utilizados. Por esta razón, se llevaron a cabo tres muestras eligiendo diferentes particiones de las entradas como datos de validación y dejando el resto para entrenamiento. Las diferentes particiones se generan de forma pseudo-aleatoria y para garantizar la reproducibilidad de los experimentos, se utilizó otra semilla, la cual siempre tomó el valor 1234.

Algoritmo 1: Cálculo del fitness para evualuar BSP.

función `calcular_fitness(`*instancia,solución*`):`
  T = obtener_período_planeamiento(instancia);
  C = obtener_capacidad_ómnibus(instancia);
  nodos = obtener_nodos_sincronización(instancia);
  fitness = 0;
  para cada nodo n **en** nodos **hacer**
    TT_b^i,TT_b^j,$WT_b^{i,j}$,$P_b^{i,j}$,$W_b^{i,j}$ = obtener_características(n) ;
    y_b^i, y_b^j = obtener_headways(solución, b);
    para cada n **en** T **step** y_b^i **hacer**
      `/* Itera sobre líneas entrantes */`
      para cada n **en** T **step** y_b^j **hacer**
        `/* Itera sobre líneas salientes */`
        tiempo_de_espera = $(n + TT_b^i) - (m + TT_b^j) - WT_b^{i,j}$;
        si *tiempo_de_espera* $> y_b^j$ **entonces**
          `/* Como máximo usuario espera headway completo */`
          tiempo_de_espera = y_b^j
        fin
        si *tiempo_de_espera* > 0 *&* *tiempo_de_espera* $<= W_b^{i,j}$ **entonces**
          `/* función objetivo multiplicada por T, para evitar división en` $f_i = T/y_b^i$ `*/`
          $fitness = fitness + min(P_b^{i,j} \times y_b^i, C \times T)$
        fin
      fin
    fin
  fin
fin

4.3.3. Aspectos técnicos

El vector de características del problema elegido para el entrenamiento es el dado por el vector definido por la ecuación 4.2. Este vector es correspondiente a la configuración C4 de características presentado en el trabajo de doctorado *Learning for Optimization with Virtual Savant* [15]. Fue elegido por presentar el mejor desempeño en el trabajo de doctorado. Los nodos de sincronización también fueron tomados de *Learning for Optimization with Virtual Savant* de

manera tal de poder generar resultados comparables.

$$\langle h^i, H^i, h^j, H^j, WT_b^{i,j}, P_b^{i,j}, W_b^{i,j}, TT_b^i, TT_b^j \rangle \tag{4.2}$$

4.3.3.1. Escalamiento de datos

En el caso de QSVM, no se realizó una reducción de la dimensionalidad mediante PCA como se hizo en los otros problemas. Debido a ello, se necesitaron diez qubits para representar el vector definido por la ecuación 4.2. Para implementar SVM sí fue necesario realizar un escalamiento de los datos. Para ello, se utilizó StandardScaler con el objetivo de normalizar los datos. Además de manipular los vectores de características, fue necesario manipular las etiquetas de los datos. Para esto, se utlizó el método LabelEncoder de sklearn que normaliza los valores de las etiquetas.

4.3.3.2. Virtual Savant

En la implementación de VS se tomó como base el trabajo de referencia, que utiliza clasificadores random forest como forma de predecir y se lo modificó para que tome SVMs y QSVMs.

Durante la ejecución, se entrenan dos clasificadores, uno para predecir los headways de las líneas entrantes y otro para predecir los headways de las lineas salientes. En los datos de entrenamiento se tienen las características dos veces pero etiquetadas de manera diferente según sean entrantes o salientes. Los vectores de características fueron definidos por la ecuación 4.2 y sus valores y etiquetas fueron también tomadas del trabajo de referencia.

Cuando la SVM o QSVM realiza la clasificación, puede ocurrir que haya múltiples predicciones para una línea de ómnibus o que alguna predicción no sea válida, dado que una línea de ómnibus puede pertenecer a varios nodos de sincronización. Dado que existe un rango de headways válidos, puede ocurrir que la predicción del headway para una línea de ómnibus no esté dentro de este rango. Las predicciones con headway fuera de rango son descartadas. Por lo tanto, una vez que la SVM o QSVM finalizan de clasificar, se agrega un paso posterior. Para cada línea de ómnibus puede ocurrir uno de los tres casos detallados a continuación:

- Hay una sola predicción válida para la línea. Se toma esta predicción.
- Hay más de una predicción válida para la línea. Se toma una de ellas al azar con distribución uniforme.

- No hay ninguna predicción válida para la línea. Se asigna una al azar dentro del rango de headways posibles con distribución uniforme.

Luego de realizada la clasificación, se pasa a la etapa de *mejora* de la solución mediante alguna heurística. En el trabajo de referencia, la heurística tomada para mejorar la predicción consiste en una búsqueda local que selecciona una línea de ómnibus [15]. Para la línea seleccionada, le cambia su headway asignado de manera aleatoria con distribución uniforme en el rango de headways válidos. El cambio es aceptado si mejora el fitness de la solución. Este procedimiento es ejecutado una cantidad parametrizable de veces. En este proyecto no se implementó la etapa de mejora, para reducir los tiempos de ejecución del algoritmo y poder estudiar de manera exclusiva el desempeño de las máquinas de soporte vectorial. El hecho de no haber implementado la etapa de mejora, implica que los resultados necesariamente sean peores a los obtenidos con AE o con Greedy. Esto es por la manera en que los algoritmos construyen la solución, Greedy es un algoritmo de tipo constructivo mientras que AE es un algoritmo constructivo con búsqueda local. Dado que no es el objetivo del proyecto de grado, no se explicará en mayor detalle por qué los algoritmos constructivos obtienen mejores resultados.

Capítulo 5

Resultados

En esta sección se presentan los resultados experimentales de la resolución de los problemas descriptos en el capítulo anterior. Los programas fueron ejecutados en la plataforma computacional del Centro Nacional de Supercomputación, ClusterUY.

Según la documentación oficial, ClusterUY cuenta con la siguientes núcleos de cómputo: 11200 núcleos de cómputo CPU Antel Xeon-Gold 6138 2.00GHz, 96 núcleos núcleos de cómputo CPU AMD EPYC 7642 2.30GHz. Cuenta con una memoria RAM de 3,80 TB y con 100.352 núcleos de cómputo GPU Nvidia Tesla P100 con 12Gb de memoria interconectados por una red de alta velocidad Ethernet de 10 Gbps [27].

5.1. Clasificación de tumores mamarios

En esta sección se presentan los resultados obtenidos en la ejecución de la SVM y la QSVM para el problema de clasificación de tumores mamarios.

En la ejecución del problema de tumores mamarios se reporta una precisión de 0,91 en el caso de kernel lineal y de 0,79 en el caso de kernel del coseno. Asimismo, en la tabla 5.1 se reportan los resultados de la ejecución de la QSVM tomando diferentes valores de semillas y utilizando en mapa de características de Pauli.

Semilla T	Semilla S	Precisión
737371	383638	0,81
514396	447126	0,78
641097	216407	0,76
237128	334758	0,78
361687	512771	0,85
881389	696518	0,80
701343	381212	0,72
950615	339180	0,84
657847	652324	0,81
340286	795502	0,79
364584	133984	0,82
594836	379325	0,78
532803	104027	0,70
972536	351707	0,79
155201	62068	0,81
124133	67435	0,82
395705	162543	0,80
654945	881968	0,76
221560	164733	0,84
837097	809660	0,75

Cuadro 5.1: Resultados de QSVM con mapa de características de Pauli.

Con el objetivo de analizar los resultados, se calculó la media, mediana y desviación estándar de los resultados brindados por la QSVM. En el caso del mapa de características de Pauli, se tiene que la ejecución tuvo un resultado de precisión medio de 0,79; mediano de 0,79 con una desviación estándar de 0,04.

Los resultados del mapa de características Product Encoding se reportan en la tabla 5.2. Se reporta que el mejor resultado obtenido fue 0,92, el peor 0,71 con una media de 0,84, una mediana de 0,85 y una desviación estándar de 0,06.

Semilla T	Semilla S	Precisión
401112	347899	0,87
832637	720977	0,73
588398	390286	0,84
439459	695328	0,90
988557	760328	0,86
922471	987784	0,90
999035	744041	0,89
18294	53809	0,85
205780	984537	0,71
726608	195898	0,84
944340	640950	0,77
575452	975929	0,92
713998	101965	0,87
671910	403167	0,83
170224	638478	0,71
579802	391994	0,83
525386	251231	0,88
126810	732227	0,87
721049	424075	0,82
312423	158075	0,81

Cuadro 5.2: Resultados de QSVM con mapa de características Product Encoding.

El gráfico 5.1 presenta los resultados de la ejecución de la QSVM con mapa de características de Pauli, comparado con ambas ejecuciones de SVM. Se aprecia que los resultados oscilan alrededor del resultado de la SVM con kernel del coseno, lo cual es consistente con el análisis estadístico previo indicó una media de resultados de 0,79, valor idéntico al obtenido en la SVM utilizando el kernel del coseno. Asímismo, los resultados reportados con QSVM con mapa de características de Pauli, figuran en todos los casos por debajo de los obtenidos con SVM kernel lineal.

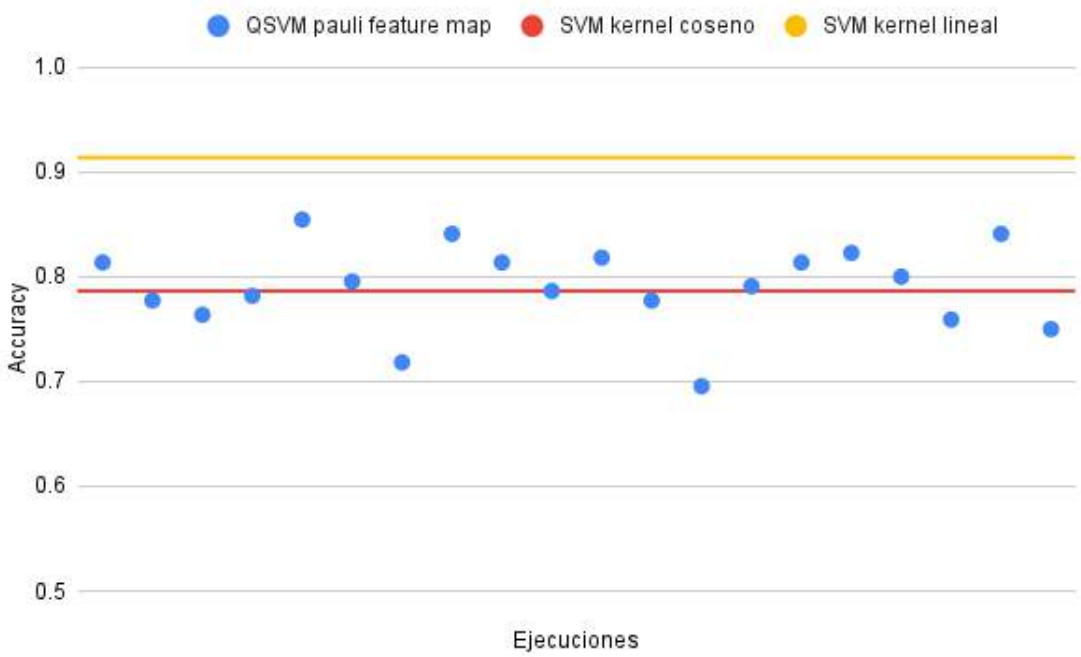


Figura 5.1: Resultados de precisión de clasificación de tumores con QSVM de Pauli.

El gráfico 5.2 presenta los resultados de la ejecución de la QSVM con mapa de características Product Encoding, comparado con ambas ejecuciones de SVM. Los resultados presentan una gran variación, llegando a tener un mejor desempeño que la SVM lineal en una ocasión y un peor desempeño que la SVM del coseno en cuatro ocasiones. En promedio su desempeño está entre el desempeño de ambas SVMs.

El mejor desempeño de la QSVM con Product Encoding sobre la SVM con kernel del coseno es sumamente interesante, pues como fue mencionado en la definición de Product Encoding (ver párrafo 2.4.1.2.2), su uso lleva implícito al kernel del coseno.

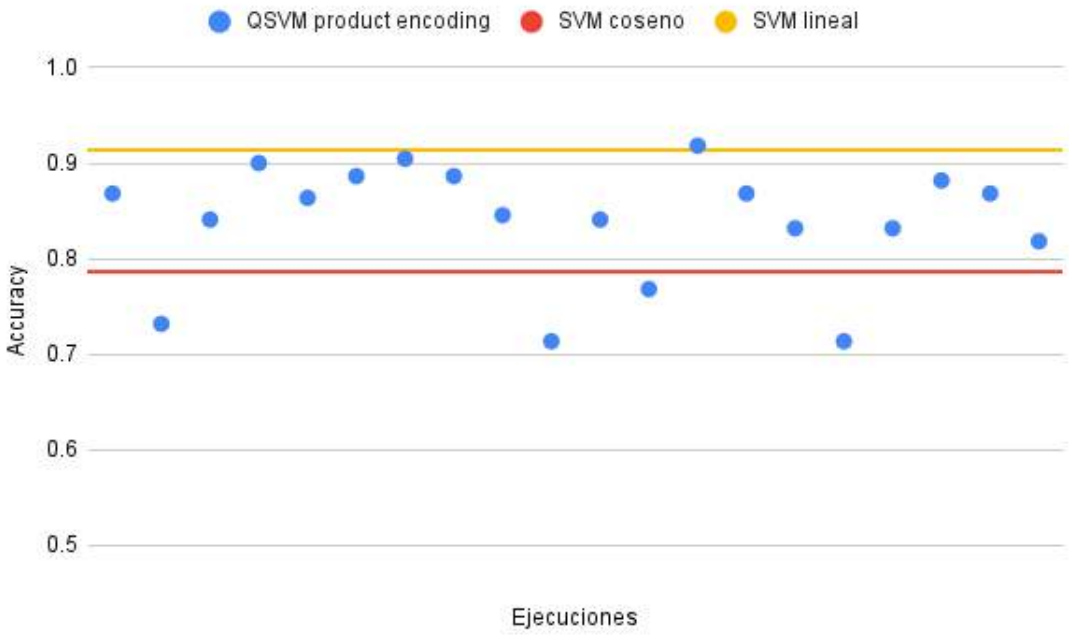


Figura 5.2: Resultados de precisión de clasificación de tumores con QSVM Product Encoding.

5.2. Clasificación de flores iris

Dado que este es un conjunto de datos relativamente pequeño, con sólo 150 instancias, se utilizó para probar el funcionamiento de QSVM y SVM con varios kernels y mapa de características.

En la tabla 5.3 se presentan los resultados de la ejecución de SVM para el problema iris. El kernel lineal tuvo una tasa de acierto del 100 %. Los demás kernels estuvieron entre el 80 y el 93 %.

Kernel	Precisión
Lineal	1,00
Coseno	0,80
RBF	0,93
Polinomial	0,80

Cuadro 5.3: Resultados de SVM para problema de iris.

Por otra parte, en la tabla 5.4 se presentan los resultados de la ejecución de QSVM con el mapa de características Product Encoding. Los resultados estuvieron en un rango similar a los resultados obtenidos con SVM, siendo

0,80 el peor resultado y 1,00 el mejor. Además, los resultados presentaron una media de 0,91 y una mediana de 0,93, con una desviación estándar de 0,07.

Semilla S	Semilla T	Precisión
5678	5678	0,93
41317	7768	0,8
32046	42531	0,93
97741	18137	0,93
90588	15270	1,00
10469	90738	0,80
57603	11939	0,93
23991	50750	1,00
88271	14978	0,87
51989	35305	0,93
20484	56992	0,80
21321	33691	1,00
11946	46057	0,93
96890	76295	1,00
14013	57220	1,00
32336	91704	0,87
7396	48588	1,00
19672	3403	0,80
95693	59828	0,80
25427	23406	0,93

Cuadro 5.4: Resultados de QSVM con mapa de características Product Encoding para el problema iris.

En la tabla 5.5 se presentan los resultados se reportan los resultados del mapa de características de Pauli. En este caso los valores oscilan entre un 0,67 en el peor caso y un 0,93 en el mejor caso. Además, los resultados presentaron una media de 0,80 y una mediana de 0,80, con una desviación estándar de 0,07. Se reporta un desempeño marcadamente inferior del mapa de caracterìsticas de Pauli comparado con el uso del mapa de características Product Encoding, destacándose que el resultado medio reportado utilizando el mapa de características de Pauli es apenas el peor resultado reportado utilizando el mapa de características Product Encoding.

Semilla S	Semilla T	Precisión
39837	26857	0,80
8358	96668	0,80
27028	57593	0,87
791	96546	0,80
5473	60574	0,73
12413	83281	0,80
845	91221	0,87
77492	79703	0,73
49614	62360	0,87
53843	32665	0,93
33174	87197	0,80
93751	67132	0,73
91327	49997	0,93
48902	63972	0,87
88290	45258	0,73
31821	55299	0,80
491	22851	0,80
38372	12104	0,73
76547	29626	0,80
9871	64241	0,67

Cuadro 5.5: Resultados de QSVM con mapa de características de Pauli para el problema iris.

La figura 5.3 muestra una comparación entre la ejecución de la QSVM con mapa de características Product Encoding y la SVM con kernel del coseno. Se aprecia que la QSVM obtiene iguales o mejores para todas las ejecuciones.

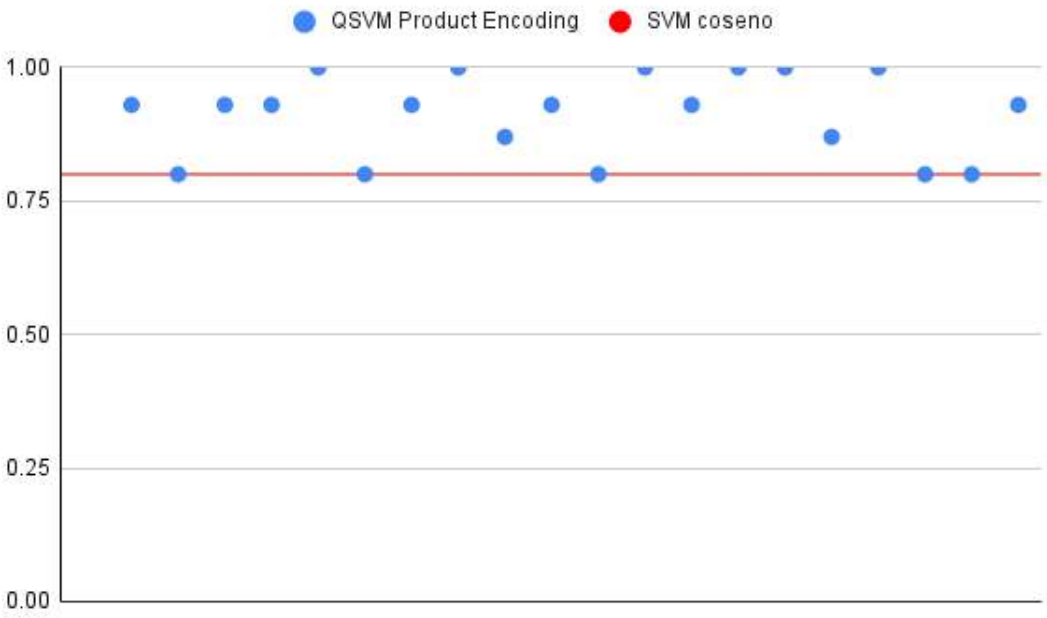


Figura 5.3: Resultados de precisión de clasificación de flores iris con QSVM Product Encoding.

5.3. Problema de sincronización de líneas de ómnibus

En esta sección se presenta la parametrización de las ejecuciones de la ejecución de Virtual Savant QSVM y SVM para el problema de sincronización de líneas de ómnibus y sus resultados.

5.3.1. Parametrización de las ejecuciones

Las instancias varían en su dimensionalidad y pueden ser de tamaño de 30, 70 o 110, como se explicó en la sección 4.3.1.2. Asimismo, son configurables también la cantidad de ejecuciones (o shots) del circuito cuántico y se presentan tres tipos de semilla (semilla de prueba, semilla transpilador y semilla simulador) para garantizar la reproducibilidad (ver secciones 3.2 y 4.3.2.3).

Para la ejecución de la QSVM y de la SVM para el problema de sincronización de líneas de ómnibus se utilizaron instancias de tamaño 30, 512 shots. Además, se toma valor de semilla de prueba 377559759, valor de semilla transpilador 79531 y de simulador 585896.

A lo largo de la evaluación experimental se ejecutó la QSVM con varios valores de semilla, de tamaño de instancia y de shots. Debido a la complejidad de las operaciones necesarias para ejecutar VS QSVM, no siempre fue posible converger a una solución. En el cuadro 5.6 se presenta el listado de ejecuciones, indicando los valores tamaño de instancia, cantidad de shots y semillas y se indica para cada partición si el algoritmo convergió con una C o si no convergió como NC.

Para las instancias de mayor tamaño no fue posible lograr que la ejecución de VS QSVM convergiera a una solución. Se tienen, sin embargo, los resultados de la ejecución de Virtual Savant SVM pero no fueron reportados porque no es posible compararlos con la ejecución de QSVM, que es el objetivo de este proyecto de grado.

Debido a la utilización de técnicas de validación cruzada para poder evaluar los resultados, las instancias fueron divididas en particiones y se reportan las predicciones divididas en 3 diferentes particiones. La técnica de validación cruzada se explica con más detalle en la subsección 4.3.2.3.

La elección del kernel y del mapa de características para la ejecución de SVM y QSVM, en las ejecuciones de los problemas de clasificación de tumores mamarios y de flores iris se reportó que el mapa de características Product Encoding obtuvo mejores soluciones que el de Pauli. Por lo tanto se tomó este mapa de características para ejecutar la QSVM. Dado que el kernel del coseno es comparable con el mapa de características Product Encoding, se lo utilizó para implementar la SVM.

Tamño.	Shots	Sem.	Sem. T	Sem. S	Part. 0	Part. 1	Part. 2
30	512	1234	524103	485796	NC	C	NC
30	512	758185	518714	98754	C	C	NC
30	512	377559759	79531	585896	C	C	C
70	512	377559759	79531	585896	NC	NC	NC
70	512	973159	91503	71093	NC	NC	NC
70	1024	959595	272727	424242	NC	NC	NC
70	1024	9999232	18911891	1891	NC	NC	NC
70	2048	999777	333	555	NC	NC	NC
70	512	25	20	15	NC	NC	NC
70	512	55	50	45	NC	NC	NC
70	512	19191912	5500	7	NC	NC	NC
70	512	71	11111	9	NC	NC	NC
70	512	512005897	77777	515	NC	NC	NC
70	512	3	7	11	NC	NC	NC
110	2048	19	17	15	NC	NC	NC
110	2048	9500	700	5	NC	NC	NC

Cuadro 5.6: Listado de ejecuciones de QSVM por tamaño, shot y semilla.

5.3.2. Cálculo de la precisión para cada algoritmo

Los resultados de precisión de los valores de los headways se reportan en las tablas 5.7, 5.8 y 5.9. Se presentan los resultados divididos por instancia, partición y algoritmo ejecutado. Dado que se entrenó un clasificador para líneas entrantes entrena una SVM (o QSVM) y otro para líneas salientes, se los presentan por separado.

Instancia	QSVM E	QSVM S	SVM E	SVM S
BS.12-14.30.40.120.70.4	0,10	0,17	0,30	0,27
BS.12-14.30.37.120.90.1	0,03	0,10	0,27	0,27
BS.12-14.30.41.120.90.2	0,13	0,13	0,23	0,13
BS.12-14.30.40.120.100.0	0,03	0,13	0,20	0,13
BS.12-14.30.37.120.100.1	0,03	0,10	0,07	0,27
BS.12-14.30.40.120.90.4	0,13	0,10	0,20	0,30
BS.12-14.30.42.120.70.3	0,23	0,23	0,37	0,33
BS.12-14.30.40.120.70.0	0,10	0,13	0,50	0,30
BS.12-14.30.40.120.100.4	0,10	0,20	0,13	0,17
BS.12-14.30.37.120.70.1	0,08	0,10	0,27	0,33

Cuadro 5.7: Precisión de la partición 0.

Instancia	QSVM E	QSVM S	SVM E	SVM S
BS.12-14.30.41.120.70.2	0,13	0,17	0,33	0,23
BS.12-14.30.40.120.90.0	0,07	0,13	0,23	0,10
BS.12-14.30.42.120.100.3	0,07	0,20	0,33	0,30
BS.12-14.30.42.120.90.3	0,03	0,27	0,20	0,33
BS.12-14.30.41.120.100.2	0,13	0,07	0,37	0,23
BS.12-14.30.40.120.90.4	0,13	0,03	0,23	0,37
BS.12-14.30.42.120.70.3	0,17	0,13	0,20	0,17
BS.12-14.30.40.120.70.0	0,07	0,10	0,23	0,17
BS.12-14.30.40.120.100.4	0,13	0,10	0,27	0,17
BS.12-14.30.37.120.70.1	0,07	0,03	0,33	0,33

Cuadro 5.8: Precisión de la partición 1.

Instancia	QSVM E	QSVM S	SVM E	SVM S
BS.12-14.30.41.120.70.2	0,03	0,17	0,30	0,20
BS.12-14.30.40.120.90.0	0,03	0,10	0,23	0,23
BS.12-14.30.42.120.100.3	0,10	0,06	0,43	0,43
BS.12-14.30.42.120.90.3	0,20	0,03	0,33	0,20
BS.12-14.30.41.120.100.2	0,13	0,13	0,23	0,17
BS.12-14.30.40.120.70.4	0,03	0,13	0,43	0,33
BS.12-14.30.37.120.90.1	0,07	0,10	0,40	0,10
BS.12-14.30.41.120.90.2	0,23	0,07	0,3	0,17
BS.12-14.30.40.120.100.0	0,13	0,17	0,33	0,37
BS.12-14.30.37.120.100.1	0,13	0,07	0,30	0,27

Cuadro 5.9: Precisión de la partición 2.

5.3.2.1. Análisis

En el cuadro 5.10 se reporta que tanto la implementación clásica como la cuántica tienen una desviación estándar próxima a 0 y además que el promedio de la SVM es ampliamente superior a la QSVM. Los resultados de precisión representan una muy baja fiabilidad en los tiempos de headways obtenidos. En el caso de la SVM se llega sólo una vez a un valor de 0,5 mientras que en el caso de QSVM nunca se llega a superar el 0,3.

Igualmente, al ser BSP un problema no determinista, para evaluar el desempeño de los distintos algoritmos se tomaron los resultados de la función de fitness.

Algoritmo	Media	Mediana	Desviación estandar
QSVM E	0,10	0,10	0,06
QSVM S	0,12	0,10	0,06
SVM E	0,29	0,28	0,09
SVM S	0,25	0,25	0,09

Cuadro 5.10: Análisis estadístico de la precisión.

En el gráfico 5.4 se grafican los resultados de precisión para las líneas entrantes, donde se observa el bajo desempeño en cuanto a precisión que presenta la QSVM, con resultados próximos a 0 en gran cantidad de instancias.

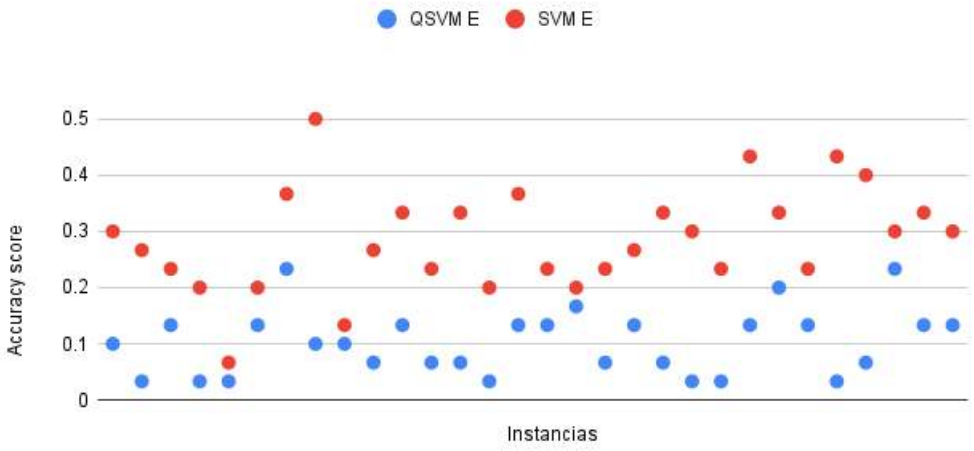


Figura 5.4: Resultados de precisión para las líneas entrantes.

El gráfico 5.5 muestra una representación gráfica de los resultados de precisión para las líneas salientes. La SVM obtuvo mejores resultados que la QSVM.

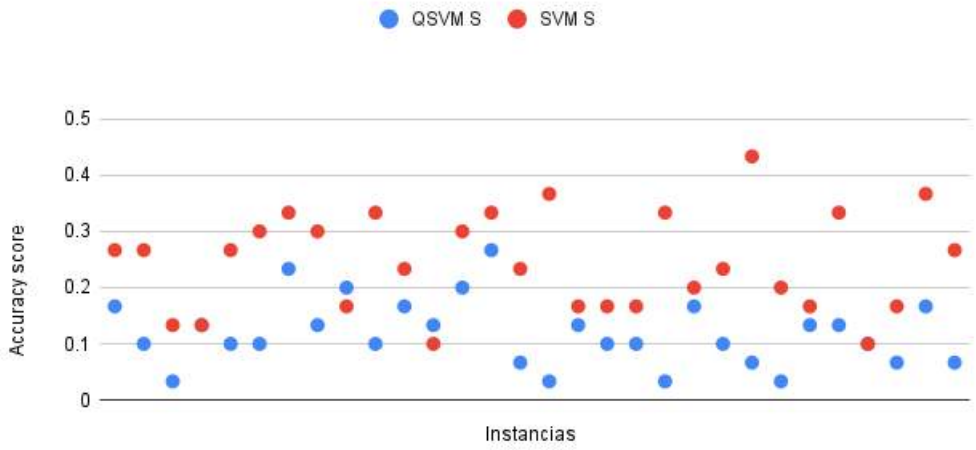


Figura 5.5: Resultados de precisión para las líneas entrantes.

5.3.3. Cálculo del fitness para cada algoritmo

En esta sección se detalla el resultado del fitness para la ejecución de VS utilizando SVM y QSVM y cada uno de los algoritmos de referencia, además del gap porcentual con el AE de cada uno de ellos.

Los resultados de fitness para cada instancia y partición se reportan en los cuadros 5.11, 5.12 y 5.13

Instancia	QSVM	SVM	Greedy	AE
BS.12-14.30.40.120.70.4	26436	26577	29801	30484
BS.12-14.30.37.120.90.1	31919	34265	34946	35351
BS.12-14.30.41.120.90.2	30498	31583	31624	32920
BS.12-14.30.40.120.100.0	26383	26324	27615	27979
BS.12-14.30.37.120.100.1	33630	33956	34995	35351
BS.12-14.30.40.120.90.4	27307	28391	30107	30528
BS.12-14.30.42.120.70.3	28358	29320	30072	31422
BS.12-14.30.40.120.70.0	23255	25569	27484	27931
BS.12-14.30.40.120.100.4	28410	26561	30137	30528
BS.12-14.30.37.120.70.1	31848	33922	34178	35290

Cuadro 5.11: Fitness de la partición 0.

Instancia	QSVM	SVM	Greedy	AE
BS.12-14.30.41.120.70.2	28353	31328	32090	32916
BS.12-14.30.40.120.90.0	25937	27256	27127	27949
BS.12-14.30.42.120.100.3	28221	30228	31068	31458
BS.12-14.30.42.120.90.3	28728	30474	30920	31458
BS.12-14.30.41.120.100.2	30360	31647	32141	32934
BS.12-14.30.40.120.90.4	27645	26805	30107	30528
BS.12-14.30.42.120.70.3	27501	27507	30072	31422
BS.12-14.30.40.120.70.0	24869	23588	27484	27931
BS.12-14.30.40.120.100.4	27143	29479	30137	30528
BS.12-14.30.37.120.70.1	32359	32534	34178	35290

Cuadro 5.12: Fitness de la partición 1.

Instancia	QSVM	SVM	Greedy	AE
BS.12-14.30.41.120.70.2	29451	31245	32090	32916
BS.12-14.30.40.120.90.0	24517	26295	27127	27949
BS.12-14.30.42.120.100.3	29003	30591	31068	31458
BS.12-14.30.42.120.90.3	29064	30767	30920	31458
BS.12-14.30.41.120.100.2	30095	31685	32141	32934
BS.12-14.30.40.120.70.4	26999	28513	29801	30484
BS.12-14.30.37.120.90.1	32233	34427	34946	35351
BS.12-14.30.41.120.90.2	30419	32218	31624	32920
BS.12-14.30.40.120.100.0	25757	27073	27615	27979
BS.12-14.30.37.120.100.1	33786	34749	34995	35351

Cuadro 5.13: Fitness de la partición 2.

5.3.3.1. Gap porcentual respecto al AE

Dado que no es posible saber el resultado óptimo para cada ejecución, es decir el máximo fitness posible para una instancia dada, se toma el resultado dado por el algoritmo AE como referencia y de los demás resultados se reporta el gap porcentual con el resultado generado por el AE de referencia.

En las tablas 5.14, 5.15 y 5.16 se reporta el gap porcentual del fitness respecto al dado por el algoritmo de referencia. Es de destacar que el algoritmo Greedy es un algoritmo de tipo constructivo y AE es un algoritmo constructivo con búsqueda local. Al no haberse implementado la fase de mejora en VS QSVM y VS SVM, es de esperar que obtengan peores resultados que los resultados de referencia.

Instancia	QSVM	SVM	Greedy	AE
BS.12-14.30.40.120.70.4	86,7 %	87,2 %	98,0 %	100,0 %
BS.12-14.30.37.120.90.1	90,3 %	96,9 %	99,0 %	100,0 %
BS.12-14.30.41.120.90.2	92,6 %	95,9 %	96,1 %	100,0 %
BS.12-14.30.40.120.100.0	94,3 %	94,1 %	98,7 %	100,0 %
BS.12-14.30.37.120.100.1	95,1 %	96,1 %	99,0 %	100,0 %
BS.12-14.30.40.120.90.4	89,4 %	93,0 %	98,6 %	100,0 %
BS.12-14.30.42.120.70.3	90,2 %	93,3 %	95,7 %	100,0 %
BS.12-14.30.40.120.70.0	83,3 %	91,5 %	98,4 %	100,0 %
BS.12-14.30.40.120.100.4	93,1 %	87,0 %	98,7 %	100,0 %
BS.12-14.30.37.120.70.1	90,2 %	96,1 %	96,8 %	100,0 %

Cuadro 5.14: Porcentaje de fitness en comparación al algoritmo benchmark para la partición 0.

Instancia	QSVM	SVM	Greedy	AE
BS.12-14.30.41.120.70.2	86,1 %	95,2 %	97,5 %	100,0 %
BS.12-14.30.40.120.90.0	92,8 %	95,5 %	97,1 %	100,0 %
BS.12-14.30.42.120.100.3	89,7 %	96,1 %	98,5 %	100,0 %
BS.12-14.30.42.120.90.3	91,3 %	96,9 %	98,3 %	100,0 %
BS.12-14.30.41.120.100.2	92,2 %	96,1 %	97,6 %	100,0 %
BS.12-14.30.40.120.90.4	90,6 %	98,9 %	98,6 %	100,0 %
BS.12-14.30.42.120.70.3	87,5 %	95,7 %	95,7 %	100,0 %
BS.12-14.30.40.120.70.0	89,0 %	84,5 %	98,4 %	100,0 %
BS.12-14.30.40.120.100.4	88,9 %	96,6 %	98,7 %	100,0 %
BS.12-14.30.37.120.70.1	91,7 %	92,2 %	96,8 %	100,0 %

Cuadro 5.15: Porcentaje de fitness en comparación al algoritmo benchmark para la partición 1.

Instancia	QSVM	SVM	Greedy	AE
BS.12-14.30.41.120.70.2	89,4 %	94,9 %	97,5 %	100,0 %
BS.12-14.30.40.120.90.0	87,7 %	94,1 %	97,1 %	100,0 %
BS.12-14.30.42.120.100.3	92,2 %	97,2 %	98,8 %	100,0 %
BS.12-14.30.42.120.90.3	92,4 %	98,3 %	98,3 %	100,0 %
BS.12-14.30.41.120.100.2	91,4 %	96,2 %	97,6 %	100,0 %
BS.12-14.30.40.120.70.4	88,6 %	93,5 %	97,8 %	100,0 %
BS.12-14.30.37.120.90.1	91,2 %	97,4 %	98,9 %	100,0 %
BS.12-14.30.41.120.90.2	92,4 %	97,9 %	96,1 %	100,0 %
BS.12-14.30.40.120.100.0	92,1 %	96,8 %	98,7 %	100,0 %
BS.12-14.30.37.120.100.1	95,6 %	98,3 %	99,0 %	100,0 %

Cuadro 5.16: Porcentaje de fitness en comparación al algoritmo benchmark para la partición 2.

5.3.3.2. Análisis

En el cuadro 5.17 se brinda un análisis estadístico sobre la calidad de resultados de cada ejecución en comparación con el algoritmo de referencia. En primer lugar, se observa que el algoritmo Greedy tiene un desempeño inferior al AE, con un gap porcentual medio de 97,9 %. La implementación con SVM obtuvo un gap porcentual medio de 96 % mientras que QSVM tuvo un gap medio de 90,9 %. En las implementaciones de VS no se implementó la etapa de mejora con búsqueda local. Si se hubiera implementado, es esperable que los resultados fueran aún mejores. Los resultados además presentan niveles de desviación estándar de 2,7 % en la implementación cuántica de VS y de 3,4 % en la implementación clásica.

Para explicar el peor desempeño de la implementación cuántica sobre la clásica en el problema BSP, el equipo de trabajo especula con que al tratarse de una simulación de computación cuántica y un problema de gran complejidad, las aproximaciones realizadas introducen errores que empeoran la solución. En particular, la estimación del producto interno mediante el uso de la rutina swap puede introducir errores. El equipo de trabajo también considera que esta es la razón por la que en instancias de dimensionalidad elevada no fue posible converger a una solución.

Algoritmo	Media	Mediana	Desviación estándar
QSVM	90,6 %	90,9 %	2,7 %
SVM	94,8 %	96,0 %	3,4 %
Greedy	97,9 %	98,3 %	1,0 %

Cuadro 5.17: Análisis estadístico del fitness respecto al benchmark.

En el gráfico 5.6 se presentan los resultados de las ejecuciones realizadas, donde se ve que los resultados obtenidos mediante QSVM son inferiores a los demás en casi todas las ejecuciones.

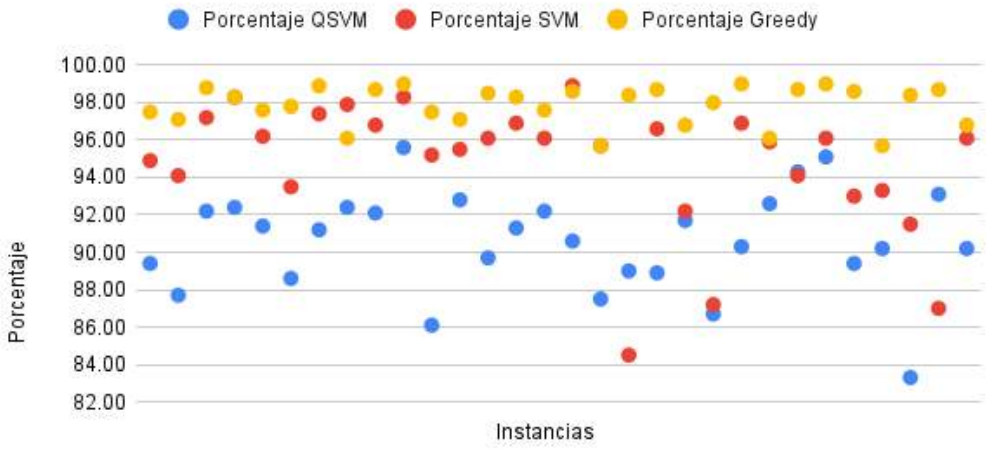


Figura 5.6: Porcentaje de fitness como gap porcentual del AE.

En el gráfico 5.7 se muestra la distribución de los resultados respecto al AE en una curva de campana. Se observa que el algoritmo Greedy tiene un pico en el 98 %, mientras que los resultados de los demás algoritmos están más distribuidos a lo largo de la curva y además tienen su pico bastante antes.

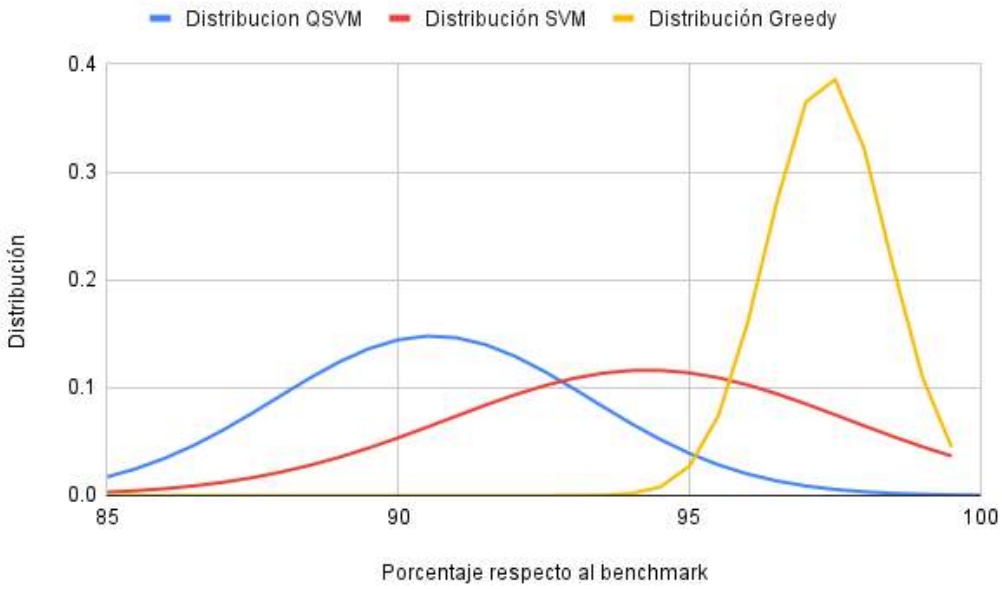


Figura 5.7: Distribución de fitness como gap porcentual del AE.

Capítulo 6

Conclusiones y trabajo futuro

A lo largo del proyecto de grado se propuso la utilización de la computación cuántica en conjunto con la inteligencia artificial para la resolución de problemas de clasificación y optimización. En particular, se buscó la forma de abordar los problemas de clasificación de flores iris y tumores mamarios, mediante el uso de SVM y QSVM para finalmente abordar un problema de optimización de líneas de ómnibus en Montevideo utilizando el paradigma *Virtual Savant*.

Como primer objetivo, se realizó una exhaustiva revisión bibliográfica que permitió un entendimiento de los principales conceptos de la computación cuántica. Seguidamente se implementó una máquina de soporte vectorial para problemas de clasificación, que luego junto con la técnica de Virtual Savant, se utilizó para revolver un problema de optimización en versiones clásicas y cuánticas, buscando poder comparar los resultados obtenidos mediante ambas metodologías. En el caso del problema BSP se cuenta con los resultados de la ejecución de un algoritmo *Greedy* y de un AE tomadas del trabajo de doctorado *Learning for Optimization with Virtual Savant*. Estos resultados fueron comparados a los obtenidos mediante las implementaciones realizadas en este proyecto de grado, tomándose el AE como algoritmo de referencia.

Para comparar el desempeño de la aplicación de VS utilizando SVM y QSVM en el problema BSP, se utilizaron los resultados obtenidos con el kernel del coseno en el caso clásico y el mapa de características Product Encoding en el caso cuántico, ya que estos son comparables entre sí. Allí se observa un mejor desempeño de la solución cuántica tanto en el problema de clasificación de tumores mamarios como en el de clasificación de flores iris. Mientras que en el problema BSP se observa un mejor desempeño de la solución clásica.

En los problemas de clasificación de tumores mamarios los resultados ob-

tenidos en la ejecución clásica con kernel lineal fueron superiores a los demás. La elección del kernel depende de los datos del problema y el hecho de que un kernel tenga mejores resultados para un problema no quiere decir que sea así para todos. Sin embargo, ello muestra la importancia de la elección del kernel y del mapa de características. En posibles trabajos futuros sería interesante estudiar los resultados del problema BSP con otros mapas de características.

En el problema BSP ambas implementaciones de VS obtuvieron resultados inferiores que el AE y el algoritmo greedy. En el caso clásico se obtuvo un gap de resultados medio de 94,8 % en comparación con el AE, mientras que en la implementación cuántica fue de 90,6 %, lo cual era esperable dada la forma en que ambos algoritmos construyen sus resultados. Los algoritmos Greedy son algoritmos de búsqueda constructiva que realizan muchas iteraciones para alcanzar soluciones a los problemas y los AE son algoritmos constructivos con búsqueda local. Por su parte, VS es un paradigma mas complejo y estocástico que depende de simulaciones y por lo tanto es importante notar que una forma de mejorar los resultados obtenidos es implementando la fase de mejora.

Los resultados obtenidos en los problemas de clasificación de tumores mamarios y de clasificación de flores iris indicaron un mejor desempeño de la implementación cuántica en comparación con la implementación clásica. Esto no es así en el problema BSP. Para explicar el mejor desempeño de la implementación en comparación con la clásica, el equipo de trabajo especula que al trabajar con una aproximación de la matriz de kernel, puede haber errores en las aproximaciones numéricas que generen un hiperplano separador que haya favorecido la clasificación de algunos de los casos usados para la validación. Para el caso del BSP, al ser un problema más complejo con un mayor espacio de soluciones y más clases, resulta menos probable que las aproximaciones numéricas en el cálculo del hiperplano separador puedan tener un efecto positivo en los casos de validación. Esta teoría parece estar apoyada por el hecho de que el BSP fue el único problema que presentó dificultades para hallar semillas para que el algoritmo convergiera. En la literatura actual no se ha realizado un análisis cualitativo del error del hiperplano separador obtenido utilizando una QSVM. Este análisis debe basarse en la cantidad de ejecuciones de la rutina swap utilizada para calcular los productos internos entre estados cuánticos y las cotas de error [8]. Para instancias grandes del problema BSP no fue posible converger a una solución. Como principal línea de trabajo futuro, se propone investigar maneras de reducir la probabilidad de no convergencia en tales casos, para poder obtener resultados en instancias grandes del problema.

Finalmente, es importante mencionar que las soluciones generadas en el presente trabajo para los diferentes problemas abordados pueden ser utilizadas como base para futuros trabajos de investigación. En particular existe la posibilidad de profundizar en el análisis de la diferencias entre la solución cuántica comparada a la clásica para el problema BSP, así como en la resolución de instancias más grandes de dicho problema.

Bibliografía

[1] S. Gamble, *Quantum Computing: What It Is, Why We Want It, and How We're Trying to Get It*. USA: National Academies Press, 2019.

[2] D. Zhang, N. Maslej, E. Brynjolfsson, J. Etchemendy, T. Lyons, J. Manyika, H. Ngo, J. C. Niebles, M. Sellitto, E. Sakhaee, Y. Shoham, J. Clark, and R. Perrault, "2021 ai index report," tech. rep., Universidad de Stanford, diciembre 2020.

[3] W. Rudin, *Real and Complex Analysis, 3rd Ed.* USA: McGraw-Hill, Inc., 1987.

[4] P. Kaye, R. Laflamme, and M. Mosca, *An Introduction to Quantum Computing*. USA: Oxford University Press, Inc., 2007.

[5] E. Grumbling and M. Horowitz, eds., *Quantum Computing: Progress and Prospects*. Washington, DC: The National Academies Press, 2019.

[6] M. Ying, "Quantum computation, quantum theory and ai," *Artificial Intelligence*, vol. 174, no. 2, pp. 162–176, 2010. Special Review Issue.

[7] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. The Edinburgh Building, Cambridge CB2 8RU, UK: Cambridge University, 2010.

[8] H. Buhrman, R. Cleve, J. Watrous, and R. de Wolf, "Quantum fingerprinting," *Physical Review Letters*, vol. 87, setiembre 2001.

[9] C. Cortez and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[10] M. Schuld and N. Killoran, "Quantum machine learning in feature hilbert spaces," *Physical Review Letters*, vol. 122, febrero 2019.

[11] J. Vert, K. Tsuda, and B. Schölkopf, *A Primer on Kernel Methods*, pp. 35–70. Cambridge, MA, USA: MIT Press, 2004.

[12] scikit-learn developers, "6.8. pairwise metrics, affinities and kernels — scikit-learn 0.24.1 documentation." `https://scikit-learn.org/stable/modules/metrics.html#polynomial-kernel`. Consultado el 11/04/2021.

[13] R. Massobrio, B. Dorronsoro Díaz, and S. Nesmachnow Cánovas, "Virtual savant for the knapsack problem: learning for automatic resource allocation," *Proceedings of the Institute for System Programming of the RAS*, vol. 31, pp. 21–32, junio 2019.

[14] F. Pinel, B. Dorronsoro, and P. Bouvry, "The virtual savant: Automatic generation of parallel solvers," *Information Sciences*, vol. 432, pp. 411–430, 2018.

[15] R. Massobrio, *Learning for Optimization with Virtual Savant*. PhD thesis, Universidad de Cádiz, Universidad de la República, 2021.

[16] V. Havlicek, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum enhanced feature spaces," *Nature*, vol. 567, junio 2018.

[17] J. Biamonte, "Universal variational quantum computation," *Phys. Rev. A*, vol. 103, p. L030401, marzo 2021.

[18] M. Möttönen, J. Vartiainen, V. Bergholm, and M. Salomaa, "Transformation of quantum states using uniformly controlled rotations," *Quantum Information & Computation*, vol. 5, pp. 467–473, setiembre 2005.

[19] The NumPy community, "Numpy v1.21 manual." `https://numpy.org/doc/stable/`. Consultado el 12/07/2021.

[20] IBM Corporation, "Qiskit." `https://qiskit.org/`. Consultado el 12/07/2021.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] Python developers, "multiprocessing — process-based parallelism — python 3.10.1 documentation." https://docs.python.org/3/library/multiprocessing.html. Consultado el 04/01/2022.

[23] Python Developers, "Globalinterpreterlock - python wiki." https://wiki.python.org/moin/GlobalInterpreterLock, 2020. Consultado el 20/01/2022.

[24] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, p. 417–441, 1933.

[25] W. H. Wolberg, W. N. Street, and O. L. Mangasarian, "Breast cancer wisconsin (diagnostic) data set." datos provistos por UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic), 1995.

[26] W. H. Wolberg and O. L. Mangasarian, "Multisurface method of pattern separation for medical diagnosis applied to breast cytology.," *Proceedings of the National Academy of Sciences of the United States of America*, diciembre 1990.

[27] S. Nesmachnow and S. Iturriaga, "Cluster-uy: Collaborative scientific high performance computing in uruguay," in *Supercomputing* (M. Torres and J. Klapp, eds.), (Cham), pp. 188–202, Springer International Publishing, 2019.