Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Aprendizaje Máquina y Computación Cuántica

Estudante: Samuel Magaz Romero
Dirección: Vicente Moret Bonillo

A Coruña, xuño de 2020.

A mamá, a papá, y a Jaco

Agradecimientos

Gracias a todos los familiares y amigos que durante estos meses me han apoyado y animado a seguir adelante con este proyecto, y me han aguantado barrenando sobre cuántica y otras historias, mucho más de lo que les hubiera gustado. Gracias a todos los profesores que se han esforzado porque llegase a donde estoy hoy, en especial a Vicente Moret, quien ha confiado en mí para llevar a cabo este trabajo.

Resumen

Tanto el Aprendizaje Máquina como la Computación Cuántica son materias relevantes y de creciente interés en investigación y desarrollo tecnológico en la actualidad. En este proyecto intentaremos demostrar cómo el Aprendizaje Máquina puede ser optimizado utilizando técnicas de Computación Cuántica.

El proyecto incluye una revisión exhaustiva de ambas materias, para luego buscar sinergias entre ellas y encontrar dominios de aplicación y procedimientos que mejoren el comportamiento de los algoritmos actuales de Aprendizaje Máquina y Computación Cuántica.

Posteriormente, se implementarán y probarán las aplicaciones diseñadas, evaluando sus resultados y contrastándolos con los métodos clásicos equivalentes, para comprobar su correcto funcionamiento y asegurar que las alternativas propuestas son válidas.

Abstract

Both fields, Machine Learning and Quantum Computing, are relevant subjects of growing interest on research and technological development in actuality. In this project we will attempt to prove how Machine Learning can be improved using quantum computing techniques.

The project includes an exhaustive revision of both fields, in order to look for synergies between them and to find domains of application and procedures which improve the behaviour of current Machine Learning and Quantum Computing algorithms.

Subsequently, the designed applications will be implemented and tested, evaluating their results and contrasting them with the equivalent classical methods, in order to verify their correct functioning and ensure that the proposed alternatives are valid.

Palabras clave:

- Aprendizaje Máquina
- Computación Cuántica
- Algoritmos genéticos
- Aprendizaje por refuerzo
- Qubit (bit cuántico)
- Superposición de estados
- Paralelismo cuántico

Keywords:

- Machine Learning
- Quantum Computing
- Genetic algorithms
- Reinforcement learning
- Qubit (quantum bit)
- Superposition of states
- Quantum parallelism

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se explica la motivación de este proyecto, así como los objetivos que pretende alcanzar y la estructura que se seguirá durante su desarrollo.

1.1 Motivación

Hoy en día no se puede negar la importancia, presente y futura, que tiene el Aprendizaje Máquina en casi cualquier sector sobre el que se pueda pensar. Algunos ejemplos son: salud, agricultura, ingeniería, economía... Dada la amplia gama de herramientas que el Aprendizaje Máquina propociona (clasificación de datos, toma de decisiones, modelos de regresión...), es difícil haya alguna que no pueda aplicarse a ciertos problemas que no son resolubles con otras metodologías clásicas (e.g. métodos estadísticos).

Por otra parte, la Computación Cuántica es una nueva rama de las Ciencias de la Computación, que se basa en las pecularieades de la física cuántica, como el entrelazado y la superposición de estados, para realizar cómputos.[1] Si bien la Computación Cuántica ya tiene cierto recorrido teórico (en los 80, R. Feynman hablaba de realizar simulaciones físicas en ordenadores cuánticos [2]), no ha sido hasta estos últimos años que ha empezado a coger fuerza. Google y la NASA dedican recursos, materiales y humanos, de modo creciente al desarrollo y aplicación de la Computación Cuántica.[3] Por esta razón es imprescindible utilizar las ventajas de la Computación Cuántica para resolver los problemas indicados como el aprendizaje maquina.

En este contexto, la pregunta es: ¿es posible aunar ambas disciplinas, Aprendizaje Máquina y Computación Cuántica, para obtener nuevas herramientas, más eficaces y eficientes?[4]

Si bien es cierto que ya hay artículos publicados al respecto (uno de los más interesantes publicado recientemente por Google[5]), el estado del arte se encuentra todavía en una fase embrionaria, puesto que aparecen nuevas propuestas, pero sin que surja una de ellas como superior a las demás.

La motivación de este proyecto busca revisar los fundamentos de las dos disciplinas, para identificar dónde se encuentran las limitaciones de cada una, y así estudiar nuevos métodos, que sean resultado de unir los aspectos compatibles entre ambas.

1.2 Objetivos

Una vez aclaradas las razones que motivan la realización de este proyecto, nos planteamos los siguientes objetivos:

- Describir los fundamentos del Aprendizaje Máquina
- Describir los conceptos básicos de la Computación Cuántica
- Estudiar distintas aplicaciones de la Computación Cuántica al Aprendizaje Máquina
- Desarrollar e implementar posibles aplicaciones

1.3 Estructura

El proyecto está estructurado de la siguiente forma:

- **Capítulo 1**: Motivación, objetivos y estructura del proyecto
- **Capítulo 2**: Análisis de fundamentos de Aprendizaje Máquina y de Computación Cuántica
- **Capítulo 3**: Aplicaciones de la Computación Cuántica al Aprendizaje Máquina
- **Capítulo 4**: Desarrollo e implementación de aplicaciones
- **Capítulo 5**: Experimentación con los métodos diseñados, y comparación con los resultados clásicos alcanzables
- **Capítulo 6**: Interpretación de resultados, discusión y conclusiones
- **Capítulo 7**: Trabajo futuro

Capítulo 2

Marco teórico

En este capítulo, se ofrece al lector una explicación general de los fundamentos de Aprendizaje Máquina y de Computación Cuántica. Se revisan distintos conceptos y modelos de cada una de las dos disciplinas, analizando las utilidades y capacidades de ambas, pero también sus limitaciones.

2.1 Aprendizaje Máquina

El Aprendizaje Máquina es la subdisciplina de la Inteligencia Artificial (IA) dedicada a estudiar algoritmos y modelos que son utilizados para realizar una tarea sin utilizar instrucciones explícitas, en lugar de basándonos en patrones e inferencias.

Partiendo de un conjunto de datos (conocido como ”conjunto de entrenamiento”), estos algoritmos implementan modelos matemáticos que se usarán para hacer predicciones sobre nuevos datos a posteriori.[6]

2.1.1 Definiciones

Una vez definido el Aprendizaje Máquina como disciplina, falta todavía por resolver cierta cuestión: *¿qué significa que una máquina aprenda?*

Desde hace ya años, a esta pregunta se le viene respondiendo citando a Tom M. Mitchell: *”Se dice que un programa aprende de una experiencia E respecto a un tipo de tarea T y una métrica de rendimiento P si el rendimiento en las tareas T, según se mida con P, mejora con la experiencia E”*.[6]

Una vez explicado qué significa que una máquina aprenda, surge la siguiente duda: *¿cómo se puede hacer que una máquina aprenda?* Para llevar a cabo este cometido, existen diversos métodos, clasificados en tres grandes categorías, según cómo afrontan dicha labor:

- Aprendizaje supervisado: el algoritmo construye un modelo matemático partiendo de un conjunto de datos que contiene tanto las entradas como las salidas deseadas. Nor-

malmente estos datos se separan en dos conjuntos: "entrenamiento", para construir el modelo, y "test", para evaluar su eficacia.

- Aprendizaje no supervisado: el algoritmo implementa un modelo matemático partiendo de un conjunto de datos que solo contiene las entradas, y no las salidas deseadas. El trabajo del algoritmo consiste en estructurar dichos datos, descubriendo patrones en ellos, de forma que los clasifica o agrupa (se habla de clustering[7]).

- Aprendizaje por refuerzo: dado un entorno de reglas y metas a alcanzar, un agente (el programa) recibe un feedback, positivo o negativo dependiendo de la estrategia que use para alcanzar la meta objetivo, sin caer en mínimos locales. Cada respuesta positiva refuerza la estrategia actual, mientras que las negativas le obligan a adaptarse.

2.1.2 Modelos y técnicas

Como ya ha sido expuesto, los algoritmos de Aprendizaje Máquina suponen la construcción de un modelo matemático (y su entrenamiento, en ciertos casos) para ser usado posteriormente. A continuación, se explican varios de los modelos existentes.

Redes de neuronas artificiales

Las Redes de Neuronas Artificiales (RNA) son modelos remotamente basados en las redes de neuronas biológicas que constituyen el cerebro. Estos sistemas están compuestos por una colección de neuronas artificiales, conectadas entre sí.

Cada conexión permite transmitir información (como si de la señal de una neurona biológica se tratase) entre neuronas. Estas conexiones tienen atribuídas unos pesos, los cuales se ajustan a medida que el sistema va aprendiendo. Dichos pesos simbolizan la intensidad de la conexión. Las neuronas artificiales pueden tener un umbral por el cual sólo pasan las señales en caso de ser superiores a él. Además, cada neurona artificial puede procesar la información que reciba.

En las redes de neuronas, normalmente las neuronas se agrupan en capas:

- De entrada: es la capa por la que entra la información de los patrones, y normalmente sólo distribuye dicha información a la siguiente capa.

- Ocultas: son las que realmente realizan el procesado, y las que tienen conexion con pesos ajustables durante el entrenamiento. Aunque en teoría no existe un número máximo, está demostrado que con tan solo una capa oculta se puede representar cualquier función continua.[8]

- De salida: está compuesta por las neuronas que reciben la información una vez procesada y devuelven la salida.

En esta sección también entrarían otros modelos de RNA como son los utilizados en *deep learning*, pero la explicación de éstos queda fuera del alcance de este proyecto.

Árboles de decisión

Los árboles de decisión son un modelo predictivo utilizado para ir desde observaciones sobre un objeto (representadas en las ramas) a conclusiones sobre el valor o clase del mismo (representadas en las hojas).

Los árboles en los que la variable objetivo puede tomar valores discreto se llaman árboles de clasificación, puesto que las hojas representan las clases y las ramas las conjunciones de características que determinan que un objeto pertenezca a una determinada clase. Por el contrario, si la variable objetivo puede tomar valores continuos, entonces se habla de un árbol de regresión.[9]

Máquinas de soporte vectorial

Las Máquinas de Soporte Vectorial (MSV) consisten en, dado un conjunto de datos, los cuales pertenecen a una de dos categorías, construír un modelo que permita determinar a qué clase pertenece un nuevo dato.

Más formalmente, una MSV construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta.

Para realizar dicha separación, las MSV utilizan funciones del núcleo (también conocidas como funciones del *kernel*), que les permiten operar en un espacio de características implícitas de alta dimensión sin calcular las coordenadas de los datos en ese espacio, sino simplemente calculando los productos internos entre las imágenes de todos los pares de datos.[10]

Una característica importante de las MSV, es que gracias al truco del *kernel*, permiten hacer clasificación tanto lineal como no-lineal, estableciendo un mapeado directo de los datos a espacios de más alta dimensionalidad.

Algoritmos genéticos

Los algoritmos genéticos son algoritmos de búsqueda que utilizan técnicas heurísticas para imitar el proceso de selección natural, recurriendo a métodos como la mutación y el cruce para generar nuevo genotipos con la finalidad de encontrar soluciones adecuadas a un problema dado.

En estos algoritmos, los candidatos a solución tienen un conjunto de características. Dichas características son evaluadas mediante una función de ajuste (*fitness*), la cual otorga a

cada candidato una puntuación según solucione mejor o peor el problema en cuestión.

A partir de dicha evaluación, se siguen los criterios que se establezcan (mantener los X mejores canditatos, rechazar los Y peores, cruzar los candidatos en un porcentaje Z, mutar un P porcentaje de sus características, ...) con la intención de obtener una nueva generación de candidatos mejor que la anterior.

Este proceso se repite hasta que se alcance una de las condiciones límite, que afortunadamente sería la de encontrar un candidato que obtenga la puntuación máxima.

2.1.3 Aplicaciones

Si bien la cantidad de aplicaciones de los métodos y técnicas de Aprendizaje Máquina es ingente, éstas se puede agrupar, dependiendo de cómo sea la tarea que están llevando a cabo. Dentro de estas agrupaciones, algunas de las más importantes son:

- Clasificación: asignar una clase o categoría a un nuevo elemento o dato sin clasificar. Lo normal es que, para realizar esta asignación, se base en ciertas características de dicho elemento, por lo que la clasificación es un ejemplo de reconocimiento de patrones.[11]

- Regresión: aproximar la relación entre una o más variables independientes (entradas) y otras variables dependientes (salidas). Un ejemplo de regresión podría ser aproximar la función que de un entero nos lleva a su cuadrado: se alimentaría el sistema con esta información, y después se utilizaría para predecir los cuadrados de nuevos valores.[12]

- Programación de IA: entrenar agentes para que puedan actuar de forma independiente. Por ejemplo, la conducción autónoma de Vehículos Aéreos no Tripulados (del inglés, Unmaned Aerial Vehicle) (UAV).[13]

- Toma de decisiones: ante un determinado problema, decidir cuál es la forma de actuar y/o proceder. Aunque podría incluírse tanto dentro de clasificación como de programación de IA, las aplicaciones de toma de decisiones (asistentes médicos, consultores financieros, ...) son tan importantes que la mayoría de autores las separan de los dos anteriores.[14]

2.1.4 Limitaciones

Uno de los problemas más importante es la propensión al fallo a la hora de dar los resultados esperados. Esto puede ser debido a varias razones: datos escasos o incompletos, problemas de privacidad, falta de recursos, problemas de evaluación...

Otro problema, también relacionado con los datos, es la posible falta de imparcialidad que pueda presentar el modelo una vez entrenado. A nada que se razone, resulta lógico que si el

modelo sólo se entrena con patrones de los tipos A y B, no sepa qué hacer con un patrón del tipo C (suponiendo que descartar patrones no sea una opción). Esto lleva a que los modelos discriminen patrones que no deberían, lo cual puede suponer un problema dependiendo de la tarea en la que se emplee dicho modelo.[15]

La mayor limitación que afecta a este proyecto es la que viene derivada del modelo clásico de computación, y sus dificultades a la hora de enfrentarse al *big data*[16]. Las fases de entrenamiento tienden a ser las más costosas de los algoritmos de Aprendizaje Máquina, y éstas adquieren gran importancia cuando el volumen de datos con el que se trabaja aumenta. En aquellos casos en los que no existe una fase de entrenamiento (algoritmos genéticos, p. ej.), existe una función de decisión que debe ser evaluada numerosas veces, proceso que también puede resultar lento.[17]

Por lo tanto, surge la idea de si las propiedades de la Computación Cuántica podrían mejorar el rendimiento de estos algoritmos, reduciendo sus limitaciones.

2.2 Computación Cuántica

La Computación Cuántica consiste en el cómputo basado en las leyes de la Mecánica Cuántica, como la superposición y el entrelazamiento. Es un paradigma de computación distinto al de la informática clásica, puesto que al utilizar qubits en lugar de bits, se da luegar a nuevas puertas lógicas que hacen posibles nuevos algoritmos.

2.2.1 El bit cuántico o qubit

En el mundo de la Computación Cuántica, existe el análogo del bit, el cual denominamos qubit (*quantum bit*), siendo la unidad básica de información. Al igual que el bit clásico, el qubit también posee dos estados, pero éstos se comportan de una manera distinta a su análogo clásico.

Dichos estados se representan mediante la notación *bra-ket*[18]:

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

Figura 2.1: Estados $|0\rangle$ y $|1\rangle$ representados mediante *kets*

Del mismo modo que en la computación clásica un único bit no permite hacer muchas operaciones, se necesita más de un qubit para poder hacer cálculos más complejos. Por lo tanto, podemos agrupar various qubits formando un sistema de qubits (también conocido como registro cuántico):

$$|00\rangle = \begin{bmatrix}1\\0\\0\\0\end{bmatrix} \qquad |01\rangle = \begin{bmatrix}0\\1\\0\\0\end{bmatrix} \qquad |10\rangle = \begin{bmatrix}0\\0\\1\\0\end{bmatrix} \qquad |11\rangle = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

Figura 2.2: Representación de los posibles estados de un sistema de 2 qubits

2.2.2 Superposición de estados

Como ya explicamos antes, el qubit puede estar en uno de dos estados, pero también puede estar en una superposición de éstos. Por lo tanto, la forma correcta de definir un qubit sería la siguiente:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

Figura 2.3: Definición de qubit

donde $\alpha, \beta \subset \mathbb{C}$ son las amplitudes de cada estado, cuyos cuadrados representan la probabilidad de que el qubit se encuentre en cada estado, lo que implica que $|\alpha|^2 + |\beta|^2 = 1$ (teoría de la probabilidad).

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{bmatrix}1\\0\end{bmatrix} + \beta \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}\alpha\\0\end{bmatrix} + \begin{bmatrix}0\\\beta\end{bmatrix} = \begin{bmatrix}\alpha\\\beta\end{bmatrix}$$

Figura 2.4: Representación de un qubit en superposición mediante *ket*

Entendiendo la superposición de estados, una forma muy práctica de representar un qubit es mediante la esfera de Bloch. En la figura 2.5, podemos observar la representación del qubit $|\psi\rangle$ mediante coordenadas esféricas en la esfera de Bloch.

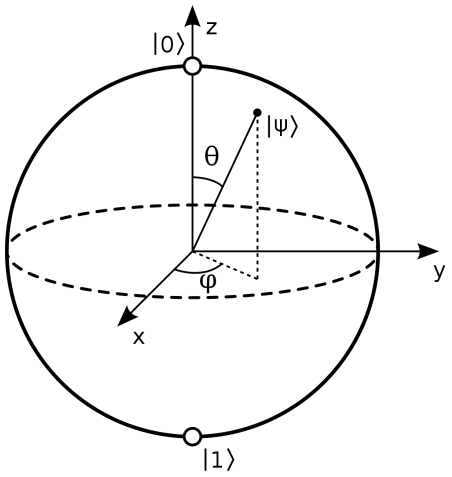


Figura 2.5: Esfera de Bloch

2.2.3 Entrelazamiento cuántico

Ahora que se han revisado los conceptos de sistema de qubits y de superposisión de estados, se puede explicar el principio de entrelazamiento cuántico, el cual no tiene un símil en la física clásica.

El entrelazamiento cuántico es un fenómeno de la mecánica cuántica (y que por lo tanto afecta a la Computación Cuántica) en el que los estados de dos o más objetos no se pueden describir de forma separada, si no que sólo se puede describir el sistema con un único estado que involucre a todos los elementos del mismo.

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Figura 2.6: Sistema de 2 qubits entrelazados

En la figura 2.6, podemos observar como los dos qubits están entrelazados. Una forma sencilla de verlo es la siguiente: sabiendo que nuestro sistema de qubits se encuentra en ese estado, si nos dicen que el primer qubit se encuentra en el estado 0, inmediatamente sabríamos que el segundo también, puesto que no hay más combinaciones posibles, de forma que los qubits están entrelazados.

Esta propiedad de la mecánica cuántica es de las más importantes en la Computación Cuántica, y se utiliza en técnicas de criptografía cuántica.

2.2.4 Decoherencia o inestabilidad del qubit

La decoherencia es el término que se utiliza en la mecánica cuántica para expresar la pérdida de coherencia cuántica, i. e., la pérdida de propiedades y comportamientos típicos del mundo cuántico, en favor de la adquisición de características propias del mundo físico clásico.

En un sistema cuántico perfectamente aislado, la coherencia se podría mantener indefinidamente, pero sería imposible poder utilizar o medir dicho sistema, por lo que resultaría inútil en este caso. Al interactuar con su entorno, el sistema pierde esta coherencia, de manera similar a un objeto que pierde energía por la fricción.

La decoherencia supone dos principales problemas para la Computación Cuántica:

- Imposibilita la capacidad de almacenar información en el tiempo (los cálculos deben realizarse con la mayor rapidez posible)
- El problema de la medida

2.2.5 El problema de la medida

Derivado de la decoherencia explicada anteriormente, el problema de la medida es uno de los que surge de la interacción con los qubits.

Si bien un qubit (o un sistema de ellos) se puede encontrar en una superposición de estados, esto es algo exclusivo del mundo cuántico: no podemos saber las amplitudes de los posibles estados de un sistema cuántico en el mundo físico, hemos de medirlo primero. Y es aquí donde surge el problema, puesto que al medir un qubit, se produce su colapso uno de los dos estados, perdiendo la información de dichas amplitudes, y siendo imposible revertir dicha medición.

Esto no tiene por qué ser negativo, puesto que obtenemos un bit clásico con el que se puede seguir trabajando. El punto clave de la cuestión reside en saber cómo trabajar con los qubits a nuestro favor, de forma que podamos aprovechar todas las propiedades que poseen, y a la vez, que al medir, el resultado nos sirva para trabajar con la información que nos proporciona.

2.2.6 Paralelismo cuántico

El paralelismo cuántico es la propiedad por la cual un sistema de qubits puede ser evaluado en todos sus estados de manera simultánea, suponiendo una mejora exponencial respecto al modelo clásico de computación. Esto se debe a la capacidad de los qubits de estar en una superposición de estados.

Dado un único bit, éste puede estar en estado 0 o 1, con el que podemos trabajar. Dado un qubit, éste puede estar en estado $|0\rangle$, $|1\rangle$ o una superposición de ambos, por lo que podemos operar sobre 2 estados a la vez. Generalizando a n elementos: n bits nos permite analizar n estados a la vez, mientras que n qubits nos permiten analizar 2^n estados de forma simultánea: he aquí la mejora exponencial que supone la Computación Cuántica.

A continuación, se revisa esta propiedad mediante un caso práctico. Esta explicación se basa en la realizada en [19].

Supongamos un qubit $|\psi\rangle$ en superposición y otro qubit en estado $|0\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{2.1}$$

Construímos un sistema con ambos qubits, y a este sistema le aplicamos la siguiente transformación U_f:

$$U_f : |\psi, 0\rangle \longrightarrow |\psi, f(\psi)\rangle \tag{2.2}$$

Dado que $|\psi\rangle$ se encontraba en superposición, obtenemos el siguiente resultado:

$$U_f\,|\psi, 0\rangle = \alpha\,|0, f(0)\rangle + \beta\,|1, f(1)\rangle \tag{2.3}$$

De esta forma, hemos aplicado el operador f a ambos estados de manera simultánea. Si ahora procedemos a realizar la medición sobre ese estado cuántico, obtenemos uno de los dos estados de los que se compone, siendo más probable aquel cuya amplitud sea mayor. Es aquí

donde radica la principal dificultad de utilizar esta propiedad, en tener la capacidad de que, a la vez que se realizan las operaciones que se quieran, las amplitudes queden de tal forma que se obtenga el resultado que se desee al realizar la medición.

2.2.7 Puertas lógicas cuánticas

En la sección 2.2.1 se explicaba cómo el qubit es el análogo cuántico del bit clásico. De forma similar, existe el equivalente a las puertas lógicas clásicas, y son las puertas lógicas cuánticas. Estas puertas lógicas reciben como entrada un cojunto de qubits, y les aplican algún tipo de transformación. Aquí se explican las más importantes.

También en la sección 2.2.1 se explicaba cómo los estados de un qubit se representan mediante un vector columna. Es por esto, que las puertas lógicas se representan como matrices, y aplicar una puerta a un qubit no es más que realizar la multiplicación de éstas.

Posiblemente la mejor para empezar sea la análoga a la puerta lógica NOT, que en Computación Cuántica sería la puerta de inversión (también conocida como puerta X, o puerta Pauli-X).

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

Figura 2.7: Puerta X

Aplicando esta puerta al estado $|0\rangle$ se obtiene el estado $|1\rangle$, y viceversa. Su nombre viene dado porque es una rotación respecto al eje X (veáse fig. 2.5). Al igual que ésta, también existen las puertas Y y Z, que rotan el estado alrededor de sus respectivos ejes.

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

Figura 2.8: Inversión de $|0\rangle$ con la puerta X

Otra puerta, ésta puramente cuántica, y de las más importantes, es la puerta de Hadamard. Esta puerta se utiliza para inicializar los qubits, es decir, pasarlos del estado $|0\rangle$ a un estado de igual probabilidad para $|0\rangle$ y $|1\rangle$.

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

Figura 2.9: Puerta de Hadamard y su aplicación

De manera similar, se pueden construir puertas para aplicarlas a un sistema de qubits, en lugar de a un qubit individual. La puerta más básica de este tipo es la Controlled-NOT (CNOT) (figura 2.10), donde se utiliza un qubit de control (también denominado ancilla) y otro sobre el

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad CNOT|10\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = |11\rangle$$

Figura 2.10: Puerta *CNOT* y un ejemplo

$$CCNOT = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figura 2.11: Puerta *CCNOT*

que se trabaja. En caso de que el primer qubit esté activado, se cambia el estado del segundo; en otro caso, no se hace nada (en ninguno de los dos casos se modifica la ancilla).

Un paso más allá, utilizando 3 qubits, está la puerta Controlled-Controlled-NOT (CCNOT) o puerta de Toffoli (figura 2.11). Similar a la CNOT, esta puerta utiliza otro qubit más como ancilla, y sólo se alterna el tercer qubit en caso de que las dos ancillas estén activadas.

Si bien esta puerta puede parecer una complicación innecesaria, en realidad es extremadamente necesaria, puesto que la puerta CCNOT se puede utilizar para construir un operador NAND reversible, constituyendo así un conjunto de puertas lógicas universales, para poder construír circuitos cuánticos.

2.2.8 Circuitos cuánticos

En Computación Cuántica, existen tres grandes filosofías: adiabática, RAM Cuántica (del inglés, Quantum RAM) (QRAM) y de circuitos cuánticos.[20][21][22] Esta última será la que utilizaremos en este proyecto. La figura 2.12 ilustra un ejemplo de circuito cuántico.

Los circuitos cuánticos, al igual que los clásicos, sirven para definir operaciones y algoritmos, pero que en este caso trabajarán sobre qubits. Se pueden utilizar operadores ya establecidos (figs. 2.7, 2.10, 2.11), así como diseñar nuevos operadores para implementarlos posteriormente (algo similar al ejemplo de la sección 2.2.6).

Los distintos operadores del circuito se computan de forma secuencial. En el circuito de la figura 2.12, primero se operaría la puera X sobre q_1, luego la CNOT sobre q_0 y q_1, y así sucesivamente.

Debido al colapso que sufren los qubits al ser observados (sección 2.2.5), las mediciones deben dejarse para el final del circuito. En este caso, se mide el qubit q_2 sobre el canal c_0 (es necesario hacer las mediciones sobre un canal distinto a los que se usan para los qubits).

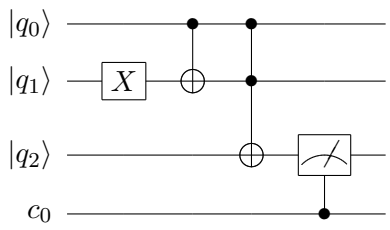


Figura 2.12: Ejemplo de circuito cuántico

2.2.9 Limitaciones

Al igual que con el Aprendizaje Máquina, procedemos a analizar las limitaciones que sufre la Computación Cuántica. Las dos más importantes son las siguientes:

- Duración en el tiempo: mencionado brevemente en la sección 2.2.4, este problema es uno que tiene difícil, sino imposible solución. La naturaleza intrínsieca de la mecánica cuántica hace imposible poder almacenar estados (y por consecuente, información) a lo largo del tiempo, por lo que no sería posible guardar información en los qubits. Este hecho hace que la Computación Clásica siga ”atada” a su clásica hermana mayor.

- Estado del arte actual: a día de hoy, la Computación Cuántica es algo propio de laboratorios y de la investigación, ya que aún no está presente de una forma consolidada en el mercado (existen opciones como las ofertadas por D-Wave Systems[23], pero no deja de ser un mercado minúsculo). Por otro lado, empresas como IBM y Google están trabajando en productos cuyo objetivo actual no es la venta al público, sino profundizar en la Computación Cuántica y seguir avanzando el campo[24][25].

Con esto concluye el marco teórico del proyecto. La sección 2.2 se basa principalmente en [1] y [26]. Se recomienda el lector acuda a dichas fuentes en caso de querer ampliar las explicaciones de los distintos conceptos aquí descritos.

Capítulo 3

Aprendizaje Máquina y Computación Cuántica

Este capítulo constituye el núcleo de este proyecto: estudiar distintas aplicaciones de la Computación Cuántica al Aprendizaje Máquina. Una vez revisadas ambas disciplinas y analizado sus fortalezas y debilidades, se han buscado soluciones y/o mejoras a aquellas deficiencias del Aprendizaje Máquina mediante herramientas propias de la Computación Cuántica. A continuación, se explica el diseño de las aplicaciones obtenidas.

3.1 Algorítmica genética cuántica

Introducidos en la sección 2.1.2, los algoritmos genéticos constituyen uno de los modelos más importantes del Aprendizaje Máquina. En la figura 3.1 se muestra un diagrama que ejemplifica la explicación de estos algoritmos.

Estos algoritmos carecen de una fase de entrenamiento como otros, por lo que resultan muy interesantes para resolver cierto tipo de problemas. En lugar de dicho entrenamiento, estos algoritmos se basan en tener un conjunto de candidatos a convertirse en solución del problema, los cuales deben ser evaluados para ver si alguno de ellos soluciona el problema de la forma más eficiente posible. En caso de que ninguno de ellos sea el candidato óptimo, se aplican distintas operaciones sobre éstos (mutación, cruce, eliminación de peores candidatos, ...) con el fin de generar nuevos candidatos, esperando que alguno de éstos se acerque más a dicha solución.

No es necesario relfexionar mucho para encontrar ciertos problemas que, si bien a día de hoy no resultan de gran importancia gracias a la potencia de los ordenadores actuales, deben tenerse en cuenta. Por ejemplo, todas las operaciones que queramos realizar sobre la generación, deben ser hechas para todos y cada uno de los miembros de la misma. Esto hace que cuanto mayor sea el tamaño de la generación, más tiempo se tarde en concluir cada uno

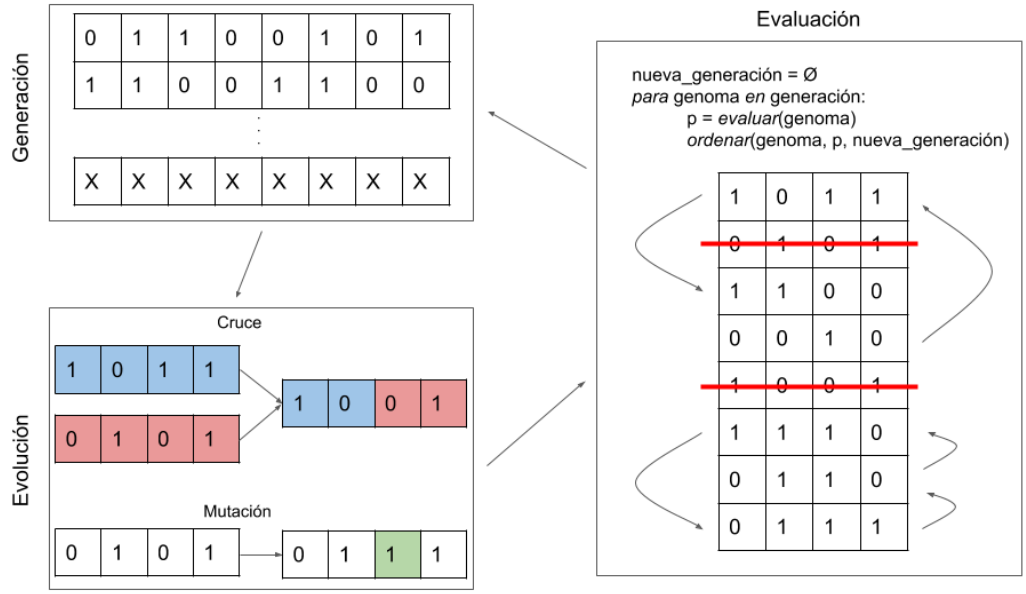


Figura 3.1: Algoritmo genético

de los pasos del algoritmo. Normalmente se intenta tener poblaciones relativamente grandes, puesto que así se originan más cruces y mutaciones, con suerte acelerando el proceso de obtención del candidato óptimo, por lo que este problema de operar con todos los genomas no es uno que se deba descartar a la ligera.

Aunque es cierto que esta debilidad puede paliarse de forma relativamente sencilla (paralelizando el algoritmo entre varios núcleos de procesado, por ejemplo), surge la curiosidad de si la Computación Cuántica puede aportar una alternativa mejor.

El enfoque que se ha seguido para la solución que se explica a continuación se basa en el paralelismo cuántico (sección 2.2.6). La idea reside en reducir la población de candidatos, teniendo un único "genoma cuántico" el cual sea una superposición de todos los estados cuánticos equivalentes a todos los posibles estados de los genomas clásicos (considerando los estados como aquellas posibles combinaciones de 0 y 1 en cada uno de los genes del genoma).

Dicho genoma cuántico está compuesto por qubits (formando un sistema de qubits), cada uno de ellos representando un gen del genoma. Si bien se podría elaborar más exhaustivamente el modelo, para la aproximación que se ha podido hacer en este proyecto, las amplitudes de cada qubit no son obtenidas a partir del cálculo de la amplitud media entre todos los genomas, si no que se trabaja con un único valor. Estas posibilidades y otras son discutidas en el capítulo de trabajo futuro.

Una vez definido el genoma cuántico, el proceso que se realiza es relativamente sencillo y parecido a los algoritmos genéticos clásicos, y seguiría los pasos indicados en 3.2. Primero se inicializan los qubits mediante puertas de rotación en el eje X (puerta Rx), poniéndolos con

igual probabilidad de 0 o 1 en la primera iteración; de esta forma, la primera vez todos los genomas posibles tienen la misma probabilidad de ser medidos. Se realiza la medición de los qubits, y con los 0 y 1 obtenidos, reconstruímos el genoma correspondiente. Este genoma se evalúa, y en función del ajuste del mismo a la función de *fitness*, se alteran más o menos las aperturas de las puertas Rx del circuito: se aumentan las de aquellos qubits que colapsaron a 1 y se reducen las de los que colapsaron a 0. Dicho ajuste es propocional a la evaluación del genoma: cuanto mayor sea el resultado obtenido, más se aumentan o reducen las amplitudes.

```
procedimiento algoritmo_genetico_cuantico():
    q_circuito = inicializar_q_circuito()
    genoma = inicializar_genoma()
    mejor_genoma = genoma
    ajuste = evaluar(genoma)
    epoch = 0
    mientras ((ajuste < MAX_AJUSTE) && (epoch < MAX_EPOCHS)):
        inicializar_q_genoma(q_circuito, genoma)
        genoma = q_circuito.ejecutar()
        ajuste = evaluar(genoma)
        mejor_genoma = comparar_genomas(genoma, mejor_genoma)
        ajustar_q_circuito(q_circuito, ajuste)
        epoch++
```

Figura 3.2: Pseudocódigo de algoritmo genético con genomas cuánticos

Para poder llevar a cabo este procedimiento, es necesario saber cómo construir el circuito cuántico en el que se evaluará nuestro genoma cuántico. En primer lugar, necesitamos determinar cuántos qubits necesitamos, en total, uno por cada gen de nuestro genoma. Para inicializar estos qubits, utilizamos puertas Rx como explicamos anteriormente. Después de estas puertas, se necesita medir cada uno de los qubits, los cuales colapsarán a 0 o 1 en función de cómo hayan sido inicializados, por lo que añadimos los dispositivos de medición y los canales correspondientes. Se puede observar el circuito descrito en la figura 3.3.

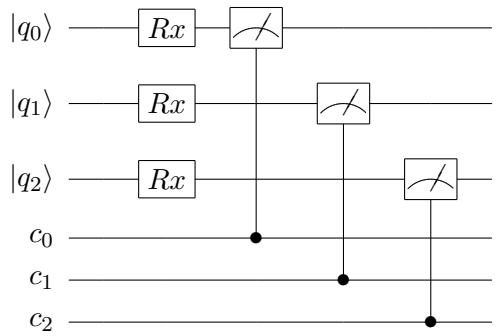


Figura 3.3: Circuito cuántico para algoritmo genético con genoma de 3 genes

De esta forma, la información que se recibe de medir el ajuste del genoma candidato en cada iteración se utiliza para evaluar todos los posibles genomas a la vez (principio de superposición), por lo que se acelera el proceso del algoritmo en sí, al reducir el número de mediciones. Aunque en un primer momento puede no parece un gran avance debido a los ejemplos que se muestran, el aumento en la velocidad se haría notable con problemas reales, donde las poblaciones son grandes y los genomas tienen muchos más genes.

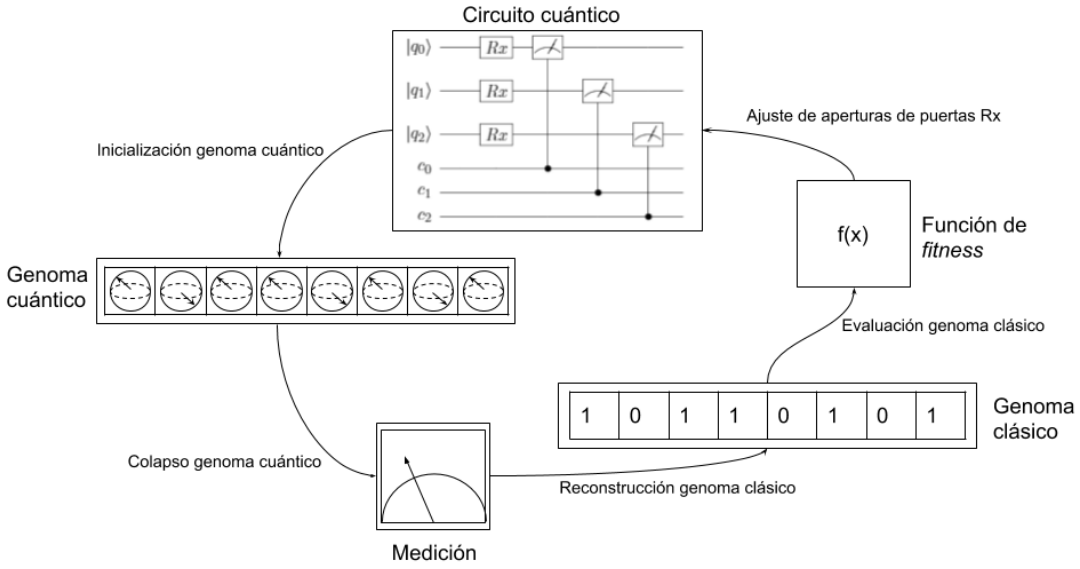


Figura 3.4: Algoritmo genético con genomas cuánticos

3.2 Aprendizaje cuántico por refuerzo

El aprendizaje por refuerzo es una de las tres grandes categorías de métodos del Aprendizaje Máquina (sección 2.1.1). Resulta de gran utilidad cuando se quiere implementar un agente que actúe de forma independiente en un entorno determinado, como por ejemplo el movimiento de robots autónomos o la Inteligencia Artificial de personajes de videojuegos.

Los métodos pertenecientes a esta categoría comparten una base común: un agente que debe aprender, se encuentra en un entorno determinado con el que puede interactuar, realizando una serie de acciones que elegirá en función de su propia estrategia. Una vez realizada la elección de entre las disponibles, la ejecuta y se lo comunica al entorno, éste le devuelve un *feedback*, el cual, dependiendo de la función que haya decidido realizar el agente, será positivo o negativo. Este *feedback* será empleado para actualizar la estrategia actual, favoreciéndola si es positivo o cambiándola en caso contrario.

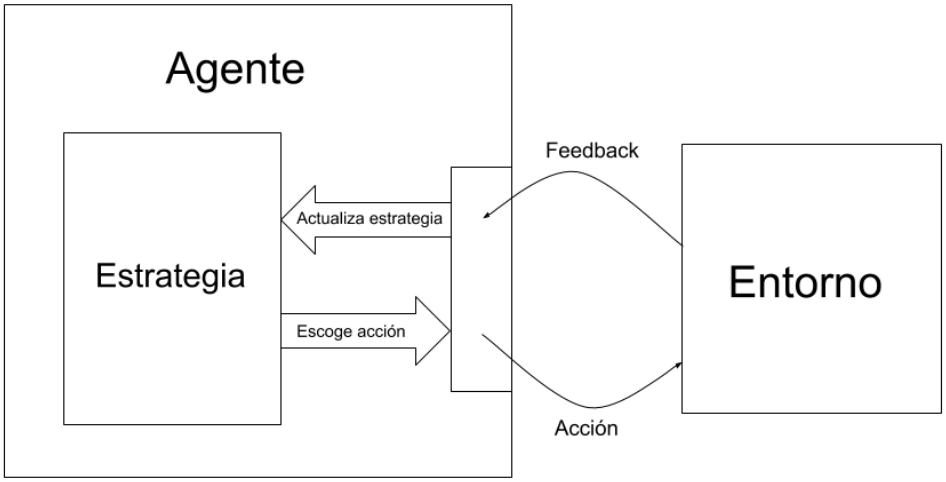


Figura 3.5: Aprendizaje por refuerzo

Esta estrategia puede verse como una tabla de probabilidades, donde según el estado en el que se encuentre el agente, ciertas acciones puedan tener mayor probabilidad de alcanzar un *feedback* positivo. El principal problema que presentan los algoritmos de este modelo es precisamente refinar esta tabla, puesto que el agente aprende de forma autónoma, a base de prueba-error, durante la fase de entrenamiento. En casos en los que el entorno en el que interactúa el agente sea grande, esta fase puede alargarse durante mucho tiempo, por lo que se buscan modelos que minimicen el tiempo de entrenamiento.

La alternativa que se presenta a continuación hace uso una vez más del pararlelismo cuántico que supone la superposición de estados (sección 2.2.6). La idea reside en utilizar dicho paralelismo para poder evaluar todas las acciones de manera simultánea. Si codificamos las acciones como estados cuánticos, podemos evaluar 2^n acciones a la vez con tan solo n qubits, reduciendo así el tiempo de entrenamiento.

En esta alternativa, la estrategia del agente pasa de ser una tabla de probabilidades a una tabla que guarda las amplitudes de los qubits que favorecen los estados que representan a cada una de las acciones, de forma que aumente la probabilidad de aquellas acciones que reportarán un *feedback* positivo (figura 3.6). De forma análoga al modelo clásico, cuando el *feedback* de una acción sea positivo, se aumentarán (según corresponda) las amplitudes de los qubits que favorecen esa acción, y se reducirán cuando el *feedback* sea negativo.

La parte de Computación Cuántica de este modelo reside en de la estrategia del agente. Como ya se explicó anteriormente, las acciones se codifican como estados cuánticos, y la estrategia se compone de las amplitudes que favorecen a dichas acciones en función de la

Estrategia clásica

Estado	P(a1)	P(a2)	P(a3)	P(a4)
0	0.25	0.40	0.25	0.10
1	0.10	0.20	0.40	0.30
2	0.00	0.33	0.33	0.33

Estrategia cuántica

Estado	R(q1)	R(q2)
0	π/4	-π/4
1	π/6	π/6
2	-π/2	-π/2

Figura 3.6: Comparación de estrategias clásica y cuántica

situación del agente dentro del entorno. De esta forma, el agente tan solo tiene que realizar una evaluación con las amplitudes correspondientes para determinar qué acción va a realizar, en lugar de buscar de entre todas las posibles cuál es la mejor opción.

Para poder realizar dicha evaluación, es necesario construir un circuito cuántico en el que se puedan operar los qubits necesarios. En primer lugar, se determina cuantos qubits son necesarios para codificar las acciones que pueda realizar el agente (n qubits permiten trabajar con 2^n acciones). A cada qubit se le pone una puerta de Hadamard (sección 2.2.7) de forma que todos los estados (posibles acciones) tengan la misma probabilidad. Posteriormente, se añade a cada qubit una puerta Rx, las cuales se inicializarán en cada evaluación con las aperturas correspondientes. Finalmente, se añaden los dispositivos de medición para obtener la acción que realizará el agente. Se muestra el circuito aquí descrito en la figura 3.7.

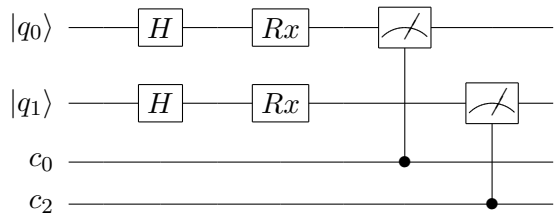


Figura 3.7: Circuito cuántico para estrategia de aprendizaje por refuerzo de agente con 4 posibles acciones

Con esto sería suficiente para implementar un modelo de aprendizaje por refuerzo con estrategia cuántica. Para aclarar su funcionamiento, se explica a continuación un ejemplo.

Supóngase que se quiere entrenar un agente para que se mueva en un plano de coordenadas. Los movimientos que puede realizar el agente son desplazarse hacia adelante, hacia atrás, hacia la izquierda o hacia la derecha. Como en total son 4 acciones, se podrían codificar con 2 qubits:

$$|00\rangle \equiv \uparrow \qquad |01\rangle \equiv \rightarrow$$
$$|10\rangle \equiv \downarrow \qquad |11\rangle \equiv \leftarrow$$

Como el agente todavía no realizado ninguna acción, todas tienen la misma probabilidad de ser ejecutadas. Se evalúa el circuito para las amplitudes correspondientes y se obtiene una medición, por ejemplo $|01\rangle$. Esto implica un movimiento hacia la derecha. En caso de que esta acción devuelva un *feedback* positivo, el agente actualizará su estrategia, modificando las amplitudes del estado anterior para que favorezcan que al realizar la evaluación, sea más probable obtener otra vez el estado $|01\rangle$ que cualquier otro (si hubiera sido negativo, las modificaría a la inversa). Suponiendo que las recompensas tan solo indican si ha sido una buena o mala acción (no indican cuánto de buena o mala), modificaríamos las aperturas con un factor determinado previamente, en este caso será de $\pi/6$.

Como se tiene que favorecer el estado $|01\rangle$, se reduce la apertura del primer qubit (favoreciendo que colapse a 0) y se aumenta la del segundo (favoreciendo que colapse a 1). Con esto también se están favoreciendo los estados $|00\rangle$ y $|11\rangle$, pero en menor medida:

Estrategia para la posición (0,0) en el instante 0:

$$R(q_0) = 0 \quad R(q_1) = 0$$
$$P(\uparrow) = 1/4 \quad P(\rightarrow) = 1/4 \quad P(\downarrow) = 1/4 \quad P(\leftarrow) = 1/4$$

Estrategia para la posición (0,0) en el instante 1:

$$R(q_0) = -\pi/6 \quad R(q_1) = \pi/6$$
$$P(\uparrow) = 8/36 \quad P(\rightarrow) = 16/36 \quad P(\downarrow) = 4/36 \quad P(\leftarrow) = 8/36$$

Destacar que estos son los resultados obtenidos para la codificación de las acciones que se ha escogido. Si, por ejemplo, el agente va a acabar favoreciendo moverse hacia la derecha y hacia abajo, convendría cambiar dicha codificación, puesto que los estados $|01\rangle$ y $|10\rangle$ se contradirían. Además, recordar que las rotaciones que se guardan en la estrategia se aplican después de una puerta de Hadamard, por lo que la rotación final del qubit será la de la estrategia más $\pi/2$.

Capítulo 4

Desarrollo e implementación

En este capítulo se explica cómo ha sido el desarrollo del proyecto, desde las decisiones que se han tomado durante su realización hasta qué herramientas se han empleado y por qué. También se exponen los detalles de implementación y de evaluación de los distintos algoritmos diseñados durante el proyecto.

4.1 Metodología de trabajo

La selección de la metodología para este proyecto ha sido realizada teniendo en cuenta la naturaleza investigadora del mismo. Se diferencian dos fases de trabajo dentro de este: una primera, en la que el esfuerzo se centra en el análisis de los métodos clásicos del Aprendizaje Máquina y las propiedades de la Computación Cuántica que pueden mejorarlos, y una segunda, orientada a la implementación de los nuevos métodos obtenidos para poder corroborarlos.

Debido a las necesidades intrínsecas de cada una de estas fases, se ha seguido una metodología apropiada en cada momento, optando por el método científico para la primera, y un modelo en espiral para la segunda.

4.1.1 Método científico e investigación sinérgica

El método científico[27] es un método empírico que ha caracterizado las ciencias naturales desde el s. XVII, el cual consiste en la observación sistemática, medición y experimentación, así como en la formulación, comprobación y modificación de hipótesis.

Este método se adapta perfectamente a nuestras necesidades durante la primera etapa del proyecto, puesto que nos permite observar las propiedades de ambas disciplinas, realizar hipótesis sobre posibles sinergias entre ellas, y realizar experimentos para comprobar la validez de las mismas.

Además, el método científico acepta la posibilidad de que los experimentos refuten la hipótesis realizada, pudiendo reformurlar esta para probarla de nuevo. Esta manera de proceder

encaja con la metodología en espiral que se sigue en la segunda fase del proyecto.

4.2 Modelos de desarrollo

Un modelo de desarrollo software es un secuencia estructurada y bien definida de las etapas para desarrollar un producto software deseado. Algunos de los más comunes son:

- Modelo en cascada: constituye una secuencia lineal de las fases de análisis, diseño, implementación y evaluación. Cada sección se aborda completamente antes de pasar a la siguiente, por lo que es un modelo poco flexible.
- Modelo en V: similar al de cascada, pero "mejorado". Hay dos tipos de actividades, de desarrollo y de prueba, entre las que existe comunicación, de forma que las pruebas se adaptan a cada fase del desarrollo en función de cómo haya sido. Permite mayor flexibilidad que el modelo en cascada, pero también es bastante rígido.
- Modelos evolutivos: estos modelos se caracterizan por abordar el software como un sistema más complejo y evolutivo, de forma que se acomodan a los distintos cambios que pueda sufrir el software durante su proceso de desarrollo. Dentro de estos modelos se encuentra el modelo en espiral, utilizado en este proyecto.

4.2.1 Modelo en espiral

Descrito originalmente por Barry Boehm en el año 1986, el modelo en espiral[28] es un modelo del proceso de desarrollo software basado en distintas actividades agrupadas en iteraciones. Es un modelo de proceso evolutivo que permite combinar la naturaleza interactiva de construcción de prototipos con los aspectos controlados y sistemáticos de otros modelos más rígidos, como el modelo en cascada. Además, pone especial énfasis en el análisis de los riesgos para reducir su impacto.

Como se aprecia en la figura 4.1, dentro de cada iteración existen diversas actividades, las cuales se pueden clasificar en las siguientes fases:

1. Determinación de objetivos, alternativas y restricciones: los primeros ciclos se centran más en la viabilidad del sistema, mientras que los siguientes se orientan al análisis de requisitos.
2. Evaluación de alternativas y análisis de riesgos: se establece qué y cómo se va a hacer en la iteración actual, teniendo en cuenta las distintas posibilidades, y cuales de ellas poseen una mayor viabilidad y un menor riesgo.

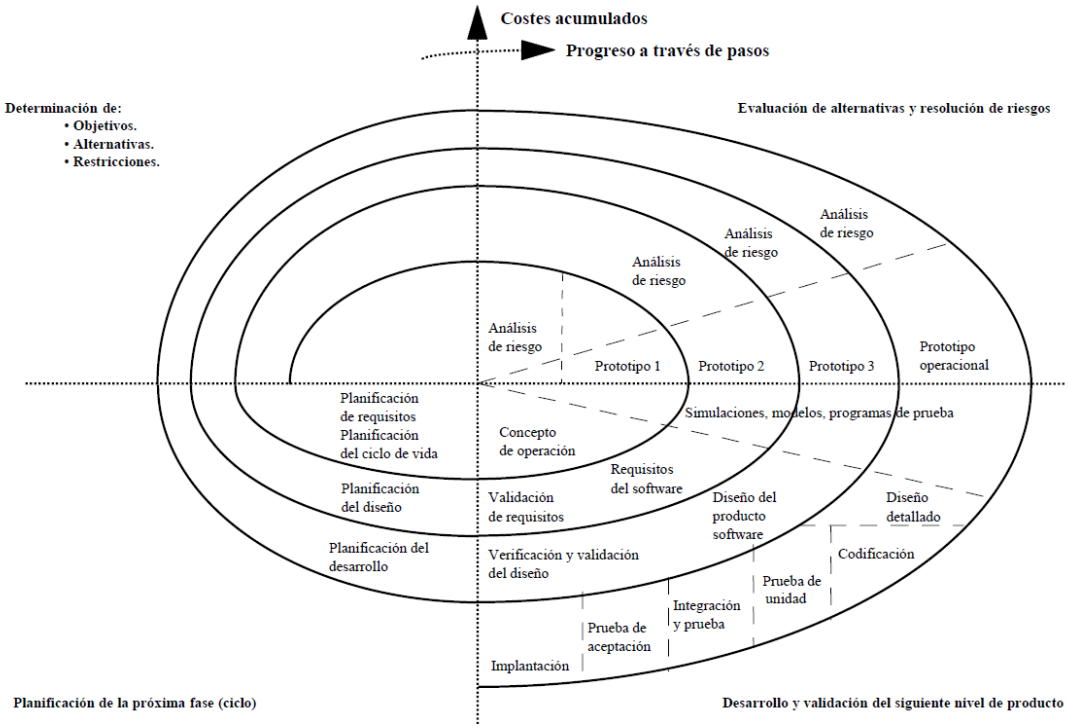


Figura 4.1: Diagrama del modelo en espiral

3. Desarrollo y validación del siguiente nivel del producto: se diseña, codifica y prueba un producto acorde a los análisis realizados previamente, validando y verificando que se cumplen los requisitos establecidos anteriormente.

4. Planificación de la próxima iteración: en caso de que se quieran implementar nuevas funcionalidades al producto o mejorar aquellas que ya posee, se planifica la siguiente iteración del modelo y se comienza otra vez con la primera fase.

Cabe destacar que en el modelo en espiral no existen fases fijas, si no que el modelo se adapta a las necesidades de cada proyecto. Esto es una ventaja frente a otros modelos, como el modelo en cascada o el modelo en V, puesto que aporta mayor flexibilidad durante todo el proceso de desarrollo, facilitando la adaptación frente a complicaciones no previstas.

4.3 Software de desarrollo[1]

Pese a que la mayor parte de este trabajo es de índole teórica, es importante exponer qué herramientas se han utilizado a la hora de realizar la parte práctica, i.e., la implementación de las soluciones propuestas y sus respectivas pruebas.

[1]Todos los derechos de los logos pertenecen a sus respectivas organizaciones.

Python

Creado por Guido van Rossum en 1991, Python[29] es un lenguaje interpretado de alto nivel de propósito general. Su filosofía de diseño enfatiza la legibilidad del código, lo cual, junto con sus constructos y su enfoque orientado a objetos, provoca que se genere código limpio con una lógica muy clara y fácilmente comprensible. Además, al ser un lenguaje interpretado, facilita la tarea de cambiar el código existente para hacer pruebas de una manera rápida y sencilla, sin necesidad de compilar tras cada cambio.

Estas características hacen de Python la primera opción para muchas personas que necesitan programar, pero no están ampliamente familiarizadas con el sector de la informática (véase matemáticos, físicos, otros ingenieros, etc.), por lo que este lenguaje cuenta también con una amplia gama de librerías del ámbito científico y matemático, muy útiles para resolver problemas de la naturaleza los de este proyecto, como algunas de las que se mencionan a continuación.

NumPy

NumPy[30] añade a Python soporte para grandes vectores y matrices multi-dimensionales, además de una amplia colección de funciones matemáticas de alto nivel para operar con los mismos. NumPy es un software *open-source* publicado bajo la licencia BSD.

SymPy

SymPy[31] es una librería de Python para matemática simbólica. Su objetivo es convertirse en un sistema de álgebra computacional con todas las funciones, manteniendo el código lo más simple posible para que sea comprensible y fácilmente extensible. SymPy está escrito completamente en Python.

Cirq

Cirq[32] es un framework para escribir, manipular y optimizar circuitos cuánticos y luego ejecutarlos contra ordenadores cuánticos y simuladores. Cirq intenta exponer los detalles del hardware, en lugar de abstraerlos, porque, en el Régimen Cuántico Ruidoso de Escala Intermedia (del inglés, Noisy Intermediate-Scale Quantum Regime) (NISQ), estos detalles determinan si es posible o no ejecutar un circuito.

4.4 Implementación del sistema de aprendizaje cuántico

A la hora de implementar las soluciones propuestas, se encontró la dificultad de que no existe ninguna librería ni framework que permita realizar algo semejante a lo que en este proyecto se pretende.

Existen soluciones que facilitan la implementación de distintas técnicas del Aprendizaje Máquina, pero no modificarlas para incluir una parte cuántica en ellas. Es por esto que se han tenido que implementar soluciones partiendo de cero, para garantizar la mayor similitud en ambas versiones de los algoritmos (clásica y cuántica). Teniendo este hecho presente, las principales consideraciones que se han tenido para la implementación de estos algoritmos han sido:

- Ambas versiones deben compartir entrada y salida de datos, para facilitar su comparación.
- Debe favorecerse la modularidad de la implementación, de forma que ambas versiones compartan la mayor parte del código posible.
- Por la naturaleza probabilística de la Computación Cuántica, es necesario llevar a cabo varias ejecuciones de los circuitos cuánticos, con el fin de minimizar el error que dicha naturaleza pueda provocar. Esta característica hará que la versión cuántica tarde más en ejecutarse, por lo que debe tratarse de reducir al máximo posible este impacto.

4.5 Validación y evaluación

El objetivo de este proyecto no es otro si no aportar soluciones o mejoras mediante herramientas propias de la Computación Cuántica a aquellas carencias que sufren algunos métodos del Aprendizaje Máquina. Por lo tanto, las pruebas que se realicen sobre las alternativas propuestas deben evaluar precisamente este aspecto, centrándose en:

- Que se realizan correctamente los pasos del algoritmo.
- Que se comportan de forma idéntica o equivalente a los algoritmos originales, es decir, que devuelve la salida correcta.

Estos dos conceptos se conocen como verificación y validación del software[33], y son los que se deben usar para asegurar que el producto satisface su propósito. Las pruebas para evaluar los algoritmos, expuestas en la tabla 4.1, han sido diseñadas con dicho objetivo.

Prueba	Resultado esperado
Maximización de funciones mediante algoritmos genéticos	Ambas versiones devuelven el mismo resultado
Entrenamiento del movimiento de un agente en un plano	Ambas versiones devuelven una política similar

Tabla 4.1: Pruebas diseñadas para la evaluación de los algoritmos

Capítulo 5

Experimentación

En este capítulo se exponen las pruebas realizadas para las soluciones propuestas, así como los resultados obtenidos tras realizar las mismas.

5.1 Pruebas realizadas

Las pruebas que se han realizado para comprobar el correcto funcionamiento de los algoritmos son las diseñadas en la sección 4.5. Debido a la naturaleza probabilística de la Computación Cuántica, cada prueba se ha ejecutado un total de 64 veces, por lo que los resultados aquí reflejados son un promedio del total de las ejecuciones.

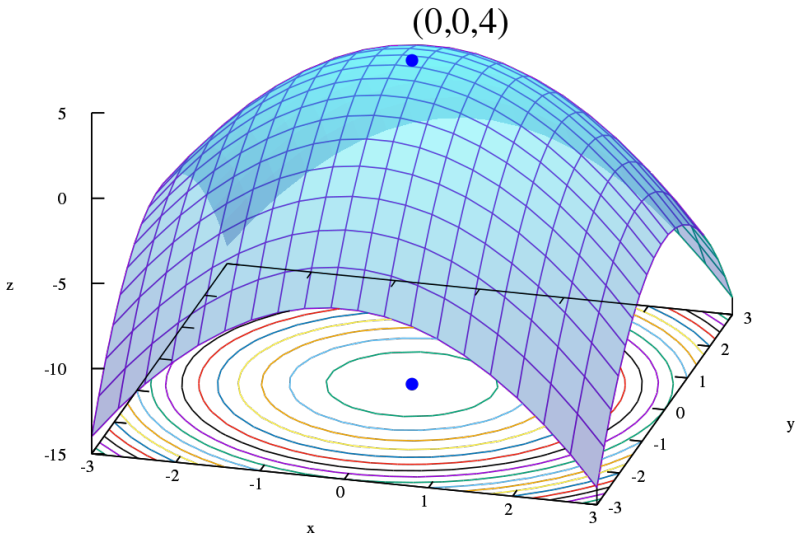
5.1.1 Maximización de funciones mediante algoritmos genéticos

Como se muestra en la tabla 4.1, se ha diseñado una prueba de maximización de funciones para evaluar la solución propuesta de algoritmos genéticos. Se ha optado por esta opción debido a dos motivos principales:

- Es una prueba que encaja perfectamente con los algoritmos genéticos y su función de *fitness*.
- Resulta sencillo programar la parte cuántica para esta prueba.

La maximización o minimización matemática (también denominada optimización) consiste en la selección del mejor elemento dentro de un conjunto, de acuerdo con un determinado criterio. En el caso más sencillo, un problema de optimización consiste en maximizar o minimizar una función real escogiendo valores de entrada para la misma y evaluando su valor de salida. La figura 5.1 ilustra este concepto.

Para los algoritmos genéticos, los candidatos representarán los valores de entrada de las funciones, siendo estos números codificados en binario, de forma que, para cada genoma, cada



La tupla $(x, y) = (0, 0)$ maximiza la función $f(x, y) = -(x^2 + y^2) + 4$ con un valor de $z = 4$

Figura 5.1: Ejemplo de maximización

uno de sus genes será un dígito de dicho número. Con esta codificación, cuando se quieran evaluar los candidatos, se pasan a base decimal y se les aplica la función, y cuando haya que pasárselos a la parte de evolución de la población, se revierte el cambio para realizar las operaciones genéticas sobre los genomas.

De esta manera, se puede utilizar la propia función a maximizar como función de *fitness*, dado que cuanto mejor sea el candidato, mayor será el valor que devuelva la función.

Aunque la solución propuesta podría trabajar con funciones reales, en este caso se ha optado por utilizar funciones enteras en pos de relizar un código más sencillo, sin tener la necesidad de codificar números reales en binario para los genomas. Por este mismo motivo, los valores de entrada a tener en cuenta se encuentran en el intervalo $[0, 7]$, los números naturales que se pueden codificar con 3 bits, de forma que el valor que se determine como máximo caerá dentro de dicho rango.

5.1.2 Entrenamiento del movimiento de un agente en un plano

En el caso de la alternativa propuesta para el aprendizaje por refuerzo, se ha escogido como prueba el entrenamiento de un agente que debe moverse sobre un plano, con el objetivo de llegar a una meta. Es un ejemplo clásico del aprendizaje por refuerzo, y parte del trabajo ya se hizo en el ejemplo de la sección 3.2.

El concepto de la prueba es relativamente simple. El agente se encuentra posicionado en una casilla de un plano. Para moverse hacia otras casillas, dispone de cuatro movimientos (arriba, abajo, izquierda y derecha). Dado que su objetivo es llegar a una determinada meta moviéndose por el plano, las recompensas que recibirá se determinan en función de si con el movimiento realizado, se acerca o aleja a su objetivo. En este caso, el entorno se compone de 9 casillas dispuestas en forma de cuadrado por las que es posible moverse. El agente comenzará situado en la casilla superior izquierda, y la meta se encontrará en la casilla inferior derecha. Para esta prueba se ha optado por que en el entorno no haya ningún otro impedimento (por ejemplo, obstáculos). La figura 5.2 ayuda a ilustrar esta prueba.

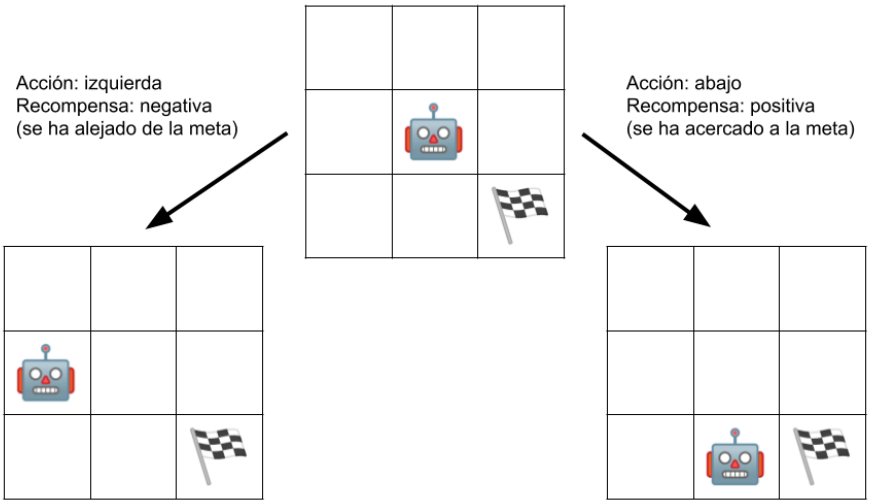


Figura 5.2: Ejemplo de acciones del agente durante la fase de entrenamiento

Para esta prueba, la principal diferencia entre la versión clásica y la versión cuántica del algoritmo se halla en el propio agente, puesto que en ambos casos el entorno en el que se encuentran y con el que interactúan es clásico. Dicho entorno también indica en qué estado se encuentra en cada momento, siendo éste determinado únicamente por la posición del agente en el plano. Por lo tanto, los estados se identifican mediante un número, del 0 al 8, empezando por arriba a la izquierda y aumentando de izquierda a derecha y de arriba abajo.

Como se explicó en la sección 3.2, la diferencia entre ambas versiones se encuentra en la

forma que tiene cada una de implementar su estrategia y, por consecuente, cómo toma las decisiones. En la versión clásica, la estrategia se compone de una tabla donde se almacenan las probabilidades de realizar cada acción en un cada estado, siendo la más probable aquella que más favorezca al objetivo (en este caso, la que acerque más al agente a la meta).

La codificación de las acciones que se ha empleado para el entrenamiento del agente cuántico es la siguiente:

$$|00\rangle \equiv \uparrow \qquad |01\rangle \equiv \leftarrow$$
$$|10\rangle \equiv \downarrow \qquad |11\rangle \equiv \rightarrow$$

Con esta codificación, y el problema que presenta la prueba, en todos los estados se deberían favorecer los movimientos derecha (estado $|10\rangle$) y abajo (estado $|11\rangle$).

5.2 Resultados obtenidos

A continuación se exponen los resultados obtenidos para cada una de las pruebas realizadas.

5.2.1 Resultados para algoritmos genéticos

La tabla 5.1 muestra los resultados de las pruebas ejecutadas con algoritmos genéticos. Ambas versiones se han evaluado con las mismas funciones, y como se explicaba al principio del capítulo, cada prueba ha sido ejecutada 64 veces, mostrando aquí los resultados promedio.

Función	Resultado versión clásica	Resultado versión cuántica
$-5x + 10$	0	0
$2x$	7	7
$3x - 5$	7	7
$-(x-2)^2 + 3$	2	2
$-4x + 7$	0	0

Tabla 5.1: Resultados de la prueba de maximización de funciones mediante algoritmos genéticos

5.2.2 Resultado para aprendizaje por refuerzo

Tal como se explicó en el apartado 5.1.2, las versiones clásica y cuántica del agente tienen las estrategias codificadas de formas diferentes. En la tabla 5.2 se muestran los valores obtenidos para el agente cuántico. Se recuerda que estos valores son los que se utilizan para inicializar las puertas Rx del circuito que determina qué acción se realizará mediante la

medición de sus qubits. Los valores asociados a cada estado indican qué acciones serán más probables en el mismo.

Estado	$Rx(q_0)$	$Rx(q_1)$
0	1.57	0.00
1	1.57	-0.78
2	1.57	-1.57
3	1.57	1.57
4	1.57	0.39
5	1.57	-1.57
6	1.57	1.57
7	1.57	1.57
8	1.57	-1.57

Tabla 5.2: Resultado de la estrategia del agente cuántico

Se puede observar como en todos los estados se concluye que el primer qubit deba colapsar a 1, de forma que siempre se obtengan los estados $|10\rangle$ o $|11\rangle$. Por ejemplo, en el caso del estado 0, por caer en la diagonal de las casillas, favorece de forma exactamente equitativa a ambos estados.

Para facilitar la comparación, las tablas 5.3 y 5.4 muestran las probabilidades de realizar cada acción en cada estado de la versión clásica y la versión cuántica, respectivamente.

Estado	P($\uparrow$)	P($\rightarrow$)	P($\downarrow$)	P($\leftarrow$)
0	0.05	0.05	0.85	0.05
1	0.15	0.55	0.15	0.15
2	0.05	0.05	0.85	0.05
3	0.12	0.64	0.12	0.12
4	0.12	0.64	0.12	0.12
5	0.00	0.05	0.90	0.05
6	0.25	0.25	0.25	0.25
7	0.00	0.05	0.90	0.05
8	0.20	0.00	0.00	0.80

Tabla 5.3: Resultado del entrenamiento del agente clásico

Estado	P(↑)	P(→)	P(↓)	P(←)
0	0.00	0.50	0.50	0.00
1	0.00	0.25	0.75	0.00
2	0.00	0.00	1.00	0.00
3	0.00	1.00	0.00	0.00
4	0.00	0.63	0.37	0.00
5	0.00	0.00	1.00	0.00
6	0.00	1.00	0.00	0.00
7	0.00	1.00	0.00	0.00
8	0.00	0.00	1.00	0.00

Tabla 5.4: Resultado del entrenamiento del agente cuántico

Capítulo 6

Interpretación, discusión y conclusiones

Este capítulo expone la interpretación de los resutlados obtenidos, la discusión sobre el proceso de realización del proyecto y las conclusiones que se han obtenido del mismo.

6.1 Interpretación de resultados

Para la alternativa de algoritmos genéticos, se puede observar en la tabla 5.1 como ambas versiones coinciden en el resultado, por lo que la versión cuántica del algoritmo podría sustituír a la versión clásica, obteniendo las ventajas expuestas en la sección 3.1.

Respecto a la solución propuesta de aprendizaje por refuerzo, analizando las tablas de resultados 5.3 y 5.4 se puede concluír que ambas estrategias son válidas para modelar el movimiento del agente en el entorno diseñado, corroborando que la alternativa cuántica de aprendizaje por refuerzo es válida, y puede ser utilizada en lugar del método clásico.

6.2 Discusión de resultados

Al comienzo de esta memoria se explicaban las motivaciones detrás de este proyecto, así como su objetivo principal: obtener nuevas versiones de los métodos clásicos del Aprendizaje Máquina, utilizando herramientas de la Computación Cuántica para subsanar las carencias de éstos.

Tras la primera fase del proyecto, en la que se analizaban los fundamentos de ambas disciplinas, se diseñaron dos nuevas alternativas a algoritmos clásicos. En primer lugar, algoritmos genéticos sustituyendo las generaciones de candidatos por un único candidato cuántico, y en segundo lugar, un método de aprendizaje por refuerzo basando la estrategia del agente en un circuito cuántico, y no en una tabla de probabilidades. Ambas alternativas hacen uso de una

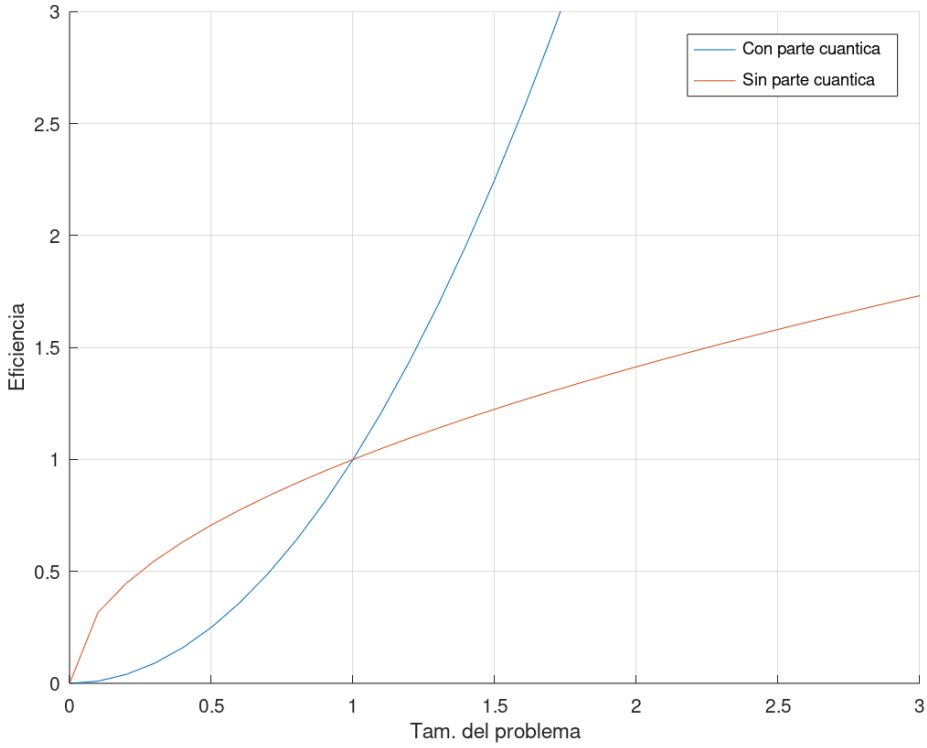


Figura 6.1: Comparación entre los métodos clásicos y cuánticos

de las propiedades más potentes de la Computación Cuántica, el paralelismo cuántico, el cual permite aumentar de forma exponencial el número de operaciones que se pueden realizar de forma simultánea.

Con las nuevas alternativas ya diseñadas, se elaboraron pruebas para comprar si efectivamente eran válidas o no. Mientras que los resultados obtenidos demostraron que si lo eran, en ninguno de los dos casos resultaron ser más rápidos que los métodos clásicos. Este problema puede deberse a que la complejidad que añaden las partes cuánticas de los métodos propuestos sea demasiado grande para problemas peuqeños como los que abordaban las pruebas. Se ha de tener en cuenta que añadir una parte cuántica al algoritmo implica realizar acciones como inicializar los circuitos cuánticos y simularlos, lo cual puede consumir más tiempo en comparación a la parte cláisca equivalente cuando esta sea relativamente pequeña.

La gráfica de la figura 6.1 ilustra el comportamiento esperado de ambos métodos según aumente el tamaño del problema. De todas formas, este estudio queda fuera del alcance de este proyecto.

Al margen de este hecho, en general se obtienen los resultados esperados, que no eran más que sinergias entre el Aprendizaje Máquina y la Computación Cuántica las cuales favorecieran

los métodos clásicos.

6.3 Conclusiones

Tras la realización de este proyecto, se ha llegado a ciertas conclusiones respecto al Aprendizaje Máquina, a la Computación Cuántica, y al propio trabajo de investigación en sí.

Ambas disciplinas suponen herramientas realmente potentes y útiles a la hora de resolver cierto tipo de problemas. El Aprendizaje Máquina ofrece métodos que ya se emplean en infinidad de situaciones a día de hoy, pero no por ello debe darse asumirse que se encuentren ya en su máximo potencial. Con la aparición de nuevas disciplinas en la informática como con el *big data* y la computación de alto rendimiento, la capacidad de cálculo aumenta de forma exponencial año tras año, permitiendo el uso de modelos más avanzados, que trabajen mejor y con más datos.

Y es precisamente una de estas nuevas disciplinas, la Computación Cuántica, que abre un gran abanico de posibilidades, donde el paralelismo cuántico permitirá resolver problemas que son irresolubles a día de hoy. No obstante, es necesario destacar la dificultad que supone desarrollar algoritmos cuánticos. Citando a Peter Shor: *"[...] los ordenadores cuánticos funcionan de una manera tan diferente a los ordenadores clásicos que nuestras técnicas para diseñar algoritmos y nuestras intuiciones para comprender el proceso de computación ya no funciona."*[34].

Si se quiere explotar todo el potencial que puede proporcionar la Computación Cuántica, es necesario apoyar su investigación e impulsar el estudio de la misma. Y es que la conclusión final de este proyecto no es otra sino la gran importancia que tiene en el ámbito de la informática la investigación, especialmente en comparación a la que realmente se le da. Todavía existen muchas técnicas y herramientas que no han sido descubiertas, y es nuestra labor encontrarlas y darles el mejor uso posible.

Capítulo 7

Trabajo futuro

En este último capítulo se exponen algunas posibles líneas futuras de trabajo, tanto de los algoritmos diseñados aquí como en general sobre la aplicación de la Computación Cuántica al Aprendizaje Máquina.

Respecto a los algoritmos genéticos con genomas cuánticos, la versión desarrollada en este proyecto es relativamente sencilla, especialmente en cuanto a la parte de evolución. Se pueden introducir conceptos como la mutación y el cruce entre genomas. Ambas operaciones pueden implementarse mediante operadores cuánticos que ya existen.

En el caso de la mutación, se puede emplear una puerta X, la cual invierte las probabilidades de colapsar a cada uno de los estados, de manera similar a como la mutación clásica cambia el valor de un gen. La operación de cruce se puede realizar con una puerta *SWAP* (en caso de que se quiera hacer siempre determinado cruce), o una puerta Fredkin, también conocida como Controlled-SWAP (CSWAP) (si se requiere controlar cuándo se realiza el cruce).

Para el aprendizaje por refuerzo resulta más difícil definir una línea de trabajo a seguir puesto que tiene una alta cohesión con el problema que se quiera resolver. De forma genérica, hay ciertos aspectos que sí se pueden evaluar para obtener una versión mejorada de la solución propuesta.

Por ejemplo, en este caso todas las soluciones estaban codificadas y evaluadas en el mismo circuito. Puede ser que se obtengan mejores resultados separando esta codificación, ya sea en grupo más pequeños o de forma individual. Otro factor que no se ha implementado en esta versión es el ratio de aprendizaje. Para este caso, las recompensas simplemente eran positivas o negativas (carecían de magnitud). Se puede intentar implementar un coeficiente de aprendizaje, de manera similar a [35].

En general, se alenta a todo aquel y toda aquella que vea una posibilidad de mejorar los métodos ya existentes del Aprendizaje Máquina mediante la Computación Cuántica. No solo por obtener nuevas alternativas, si no por ahondar en la investigación de ambas disciplinas, donde puede que se encuentre el siguiente gran avance en la historia de la computación.

Apéndices

Apéndice A

Planificación

A.1 Material

El material utilizado para la realización de este proyecto ha sido:

- Intel Core i5-7600K @ 3.80GHz
- 8GB RAM
- Debian GNU/Linux 10 (buster)
- Python 3.7.3

A.2 Planificación

La planificación del proyecto se puede observar en el diagrama de Gantt de la figura A.1.

A.3 Costes

La tabla A.1 muestra el desglose y el coste total del proyecto.

		Recursos
		Ingenier@ informático
Costes	€/año	25.000€
	€/hora	15€
	total	350h x 15€/h = 5.250€

Tabla A.1: Costes del proyecto

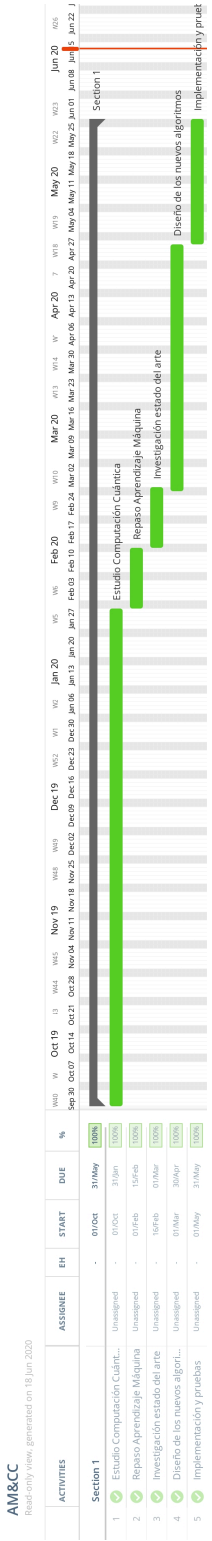


Figura A.1: Diagrama de Gantt con la planificación del proyecto

Lista de acrónimos

CCNOT Controlled-Controlled-NOT. 12

CNOT Controlled-NOT. 11, 12

CSWAP Controlled-SWAP. 39

IA Inteligencia Artificial. 3, 6, 18

MSV Máquinas de Soporte Vectorial. 5

NISQ Régimen Cuántico Ruidoso de Escala Intermedia (del inglés, Noisy Intermediate-Scale Quantum Regime). 26

QRAM RAM Cuántica (del inglés, Quantum RAM). 12

RNA Redes de Neuronas Artificiales. 4, 5

UAV Vehículos Aéreos no Tripulados (del inglés, Unmaned Aerial Vehicle). 6

Glosario

clustering Agrupación de una serie de elementos de acuerdo con un criterio. 4

feedback Devolución de una señal modificada a su emisor. 4

puerta Rx Puerta cuántica de rotación en el eje X. 16

puerta Fredkin Puerta de intercambio controlado de qubits. 39

qubit Unidad de información básica de la Computación Cuántica. 7

Bibliografía

[1] N. S. Yanofsky and M. A. Mannucci, *Quantum Computing for Computer Scientists*. Cambridge University Press, 2008.

[2] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6–7, p. 467–488, Jun 1982.

[3] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, and et al., "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, p. 505–510, Oct 2019.

[4] V. Moret-Bonillo, "Can artificial intelligence benefit from quantum computing?" *Progress in Artificial Intelligence-Springer*, 09 2014.

[5] M. Broughton, G. Verdon, T. McCourt, A. J. Martinez, J. H. Yoo, S. V. Isakov, P. Massey, M. Y. Niu, R. Halavati, E. Peters *et al.*, "Tensorflow quantum: A software framework for quantum machine learning," *arXiv preprint arXiv:2003.02989*, 2020.

[6] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[7] M. Aldenderfer and R. Blashfield, *Cluster Analysis*. SAGE Publications, Inc., 1984. [Online]. Available: https://doi.org/10.4135/9781412983648

[8] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989. [Online]. Available: https://doi.org/10.1007/bf02551274

[9] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[10] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, vol. 36, no. 3, pp. 1171–1220, Jun. 2008. [Online]. Available: https://doi.org/10.1214/009053607000000677

[11] E. Alpaydin, *Introduction to Machine Learning*, ser. Adaptive Computation and Machine Learning series. MIT Press, 2014. [Online]. Available: https://books.google.es/books?id=7f5bBAAAQBAJ

[12] D. Freedman, *Statistical Models: Theory and Practice*. Cambridge University Press, 2005. [Online]. Available: https://books.google.es/books?id=hmt1tFDdFqEC

[13] A. K. Yadav and P. Gaur, "Ai-based adaptive control and design of autopilot system for nonlinear uav," *Sadhana*, vol. 39, no. 4, pp. 765–783, 2014.

[14] M. L. Littman, *Algorithms for sequential decision making*. Brown University Providence, RI, 1996.

[15] J. Dastin, "Amazon scraps secret AI recruiting tool that showed bias against women," https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G, 2018, accedido el 18/04/2020.

[16] P. Zikopoulos, C. Eaton *et al.*, *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

[17] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, Oct. 2014. [Online]. Available: https://doi.org/10.1080/00107514.2014.964942

[18] P. A. M. Dirac, "A new notation for quantum mechanics," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 35, no. 3, p. 416–418, 1939.

[19] D. Dong, C. Chen, H. Li, and T. Tarn, "Quantum reinforcement learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 5, pp. 1207–1220, 2008.

[20] R. Barends, A. Shabani, L. Lamata, J. Kelly, A. Mezzacapo, U. Las Heras, R. Babbush, A. G. Fowler, B. Campbell, Y. Chen *et al.*, "Digitized adiabatic quantum computing with a superconducting circuit," *Nature*, vol. 534, no. 7606, pp. 222–226, 2016.

[21] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Physical review letters*, vol. 100, no. 16, p. 160501, 2008.

[22] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell *et al.*, "Superconducting quantum circuits at the surface code threshold for fault tolerance," *Nature*, vol. 508, no. 7497, pp. 500–503, 2014.

[23] D-Wave Systems, "Meet D-Wave | D-Wave Systems," https://www.dwavesys.com/our-company/meet-d-wave, 2020, accedido el 25/04/2020.

[24] Google Research, "Quantum | Google Research," https://research.google/teams/applied-science/quantum/, 2020, accedido el 25/04/2020.

[25] IBM, "IBM | Quantum Computing," https://www.ibm.com/quantum-computing/, 2020, accedido el 25/04/2020.

[26] V. Moret-Bonillo, *Adventures in Computer Science*. Springer, 2017.

[27] Oxford English Dictionary, "Scientific Method | Definition of Scientific Method by Oxford Dictionary on Lexico," https://www.lexico.com/definition/scientific_method, 2020, accedido el 15/06/2020.

[28] B. Boehm, "A spiral model of software development and enhancement," *SIGSOFT Softw. Eng. Notes*, vol. 11, no. 4, p. 14–24, Aug. 1986. [Online]. Available: https://doi.org/10.1145/12944.12948

[29] Python, "Welcome to Python.org," https://www.python.org, 2020, accedido el 15/06/2020.

[30] NumPy, "NumPy," https://https://numpy.org/, 2020, accedido el 15/06/2020.

[31] SymPy Development Team, "SymPy," https://www.sympy.org/en/index.html, 2020, accedido el 15/06/2020.

[32] The Cirq Developers, "Cirq 0.8.0 documentation," https://cirq.readthedocs.io/en/stable/, 2020, accedido el 15/06/2020.

[33] Porject Management Institute, *A guide to the project management body of knowledge (PMBOK guide)*. Newtown Square, PA: Project Management Institute, 2017.

[34] P. W. Shor, "Why haven't more quantum algorithms been found?" *Journal of the ACM*, vol. 50, no. 1, pp. 87–90, Jan. 2003. [Online]. Available: https://doi.org/10.1145/602382.602408

[35] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, May 1992. [Online]. Available: https://doi.org/10.1007/bf00992698