

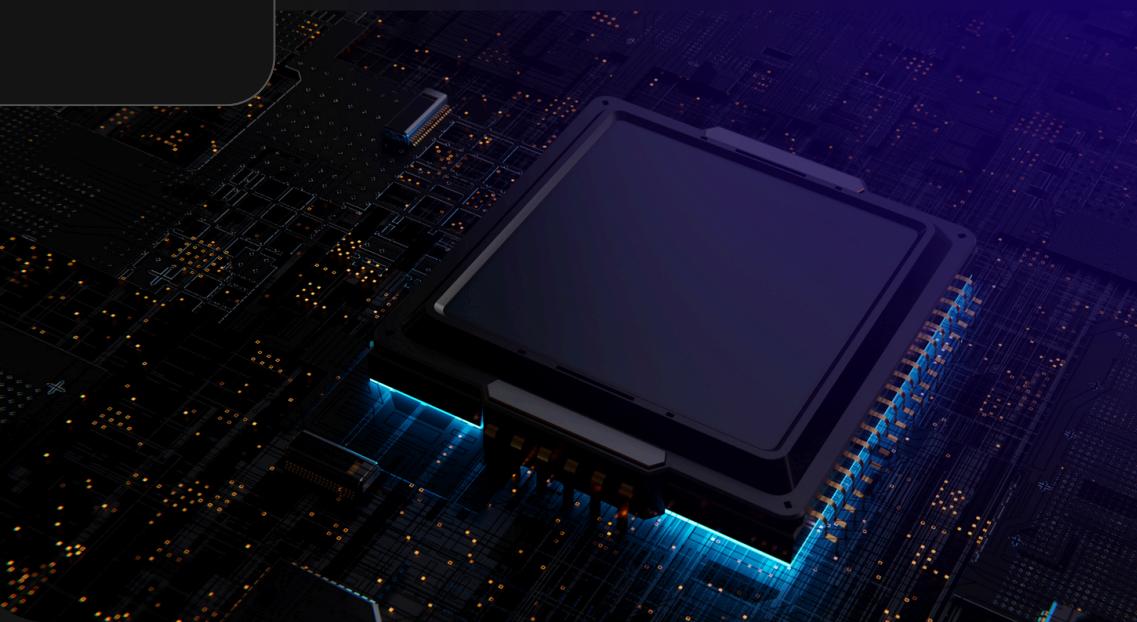
ProyectoQuantumML

Asignatura: Computación cuántica

Docente: Hans Harley Ccacyahuillca Beja

Integrantes:

- Ccapatinta Qqueccáñ Dennis Moises
- Saire Hancco Cesar Andersson
- Quispe Merma Timoteo
- Corampa Palacios Aracely Fiorela
- Huanca Alcca Jhon William
- Mamani Gabriel Bruce Maximo.



140984
141158
224874
222067
225421
182917

El Desafío - Límites de la IA Clásica y la Promesa Cuántica

Exploramos las fronteras de la Inteligencia Artificial clásica y cómo la computación cuántica, específicamente el Quantum Machine Learning (QML), emerge como la próxima evolución para resolver problemas de complejidad intratable.

IA CLÁSICA

LÍMITES

COMPUTACIÓN CUÁNTICA

La IA Clásica: Poderosa pero con Fronteras

Aunque la Inteligencia Artificial clásica ha logrado avances impresionantes en reconocimiento facial, traducción automática, conducción autónoma y diagnóstico médico, enfrenta limitaciones significativas en problemas de escala exponencial.

- Logros Impresionantes:** Reconocimiento facial (99% precisión), traducción instantánea (+100 idiomas), vehículos autónomos, diagnósticos médicos asistidos.
- Problemas que la Desafían:** Simulación molecular, optimización global de rutas, predicción climática de alta precisión, factorización de números grandes para criptografía.

La Siguiente Frontera: Quantum Machine Learning (QML)

Cuando los problemas superan la capacidad de las supercomputadoras clásicas, que tardarían años o siglos en resolverlos, surge la necesidad de un nuevo paradigma: la computación cuántica. Aquí es donde el QML ofrece una esperanza.

- Evolución:** De la computación clásica al Machine Learning, Deep Learning y, ahora, al Quantum Machine Learning.
- Definición Simple:** La integración del Aprendizaje Automático con los principios fundamentales de la mecánica cuántica, utilizando qubits y fenómenos cuánticos para procesar información.

Los Dos Pilares Cuánticos: Superposición y Entrelazamiento

1. Superposición Cuántica: Múltiples Estados Simultáneos

A diferencia de un bit clásico (0 o 1), un qubit puede ser 0 y 1 a la vez. Este fenómeno permite un procesamiento masivo en paralelo y la exploración simultánea de un vasto espacio de soluciones, revelando patrones ocultos. Un sistema de 4 qubits puede representar 16 estados a la vez, mientras que 300 qubits pueden almacenar más estados que el número de átomos en el universo.

2. Entrelazamiento Cuántico: Conexiones No-Clásicas

Cuando los qubits se entrelazan, sus estados se correlacionan de manera instantánea, sin importar la distancia. Este "acción fantasmal a distancia" (Einstein) permite capturar correlaciones complejas en los datos, ofreciendo una capacidad expresiva única para los modelos de Machine Learning. Una sola puerta cuántica puede simular el trabajo de múltiples capas clásicas.

Justificación y Contexto del Proyecto: Por Qué Quantum Machine Learning Ahora

Este proyecto se sitúa en la vanguardia de la tecnología, respondiendo a un crecimiento exponencial en QML, la disponibilidad de hardware cuántico, la creciente demanda laboral y nuestra contribución a la investigación de vanguardia.

CAMPO EMERGENTE

HARDWARE CUÁNTICO

FUTURO DEL TRABAJO

Campo Emergente en Explosión Exponencial

El QML está experimentando un crecimiento sin precedentes. Los papers científicos han aumentado 40 veces desde 2015, con inversiones globales superiores a los \$30 mil millones de USD. Empresas como IBM, Google y Amazon lideran la innovación, marcando el inicio de una nueva era tecnológica.

Preparación para el Futuro del Trabajo

La demanda de talento en computación cuántica está en auge, con salarios competitivos para ingenieros y científicos. Habilidades en programación cuántica (Qiskit, Cirq) y algoritmos variacionales serán cruciales en los próximos 5-10 años, haciendo de esta una inversión de carrera estratégica.

De Teoría a Realidad: Hardware Cuántico Disponible

Ya no es ciencia ficción. IBM Quantum ofrece acceso gratuito a sus procesadores de 127 qubits (Eagle) en la nube, mientras que Google y Amazon Braket proporcionan acceso a otros QPUs. Esto significa que la investigación y experimentación con QML es accesible hoy mismo.

Contribución a Investigación de Vanguardia

Este proyecto busca ofrecer una implementación reproducible de VQC en español, una comparación justa con baselines clásicos y un análisis honesto de las limitaciones. Contribuye a llenar la brecha de recursos educativos y desmitifica el "hype" cuántico con datos reales.

Tecnologías Híbridas: El Futuro Es Colaboración Cuántica-Clásica

El mito de que los ordenadores cuánticos reemplazarán a los clásicos es eso, un mito. La realidad es que trabajarán juntos, formando arquitecturas híbridas que combinan lo mejor de ambos mundos para resolver problemas complejos.

1

Preprocesamiento Clásico

La CPU/GPU procesa datos iniciales, extrae features relevantes y normaliza.

2

Procesamiento Cuántico

La QPU resuelve subproblemas específicos, usando superposición y entrelazamiento.

3

Post-procesamiento Clásico

La CPU interpreta los resultados, ajusta hiperparámetros y visualiza.

Contexto de Nuestro Experimento: Dataset Iris y Stack Tecnológico

Dataset: Iris (Filtrado a 2 Clases)

Elegimos el dataset Iris por su simplicidad y familiaridad, ideal para validar conceptos y facilitar comparaciones. Filtrado a 2 clases (Setosa vs Versicolor) para optimizar el rendimiento en simuladores cuánticos actuales, que son intrínsecamente lentos.

- **Datos:** 100 muestras (50 por clase), 4 features.

Stack Tecnológico Completo

- **Lenguaje Base:** Python 3.8+
- **Frameworks Cuánticos:** Qiskit, Qiskit Machine Learning, Qiskit Algorithms.
- **Frameworks Clásicos:** Scikit-Learn, NumPy, Pandas, Matplotlib/Seaborn.
- **Plataforma:** Google Colab con StatevectorSampler (simulador ideal).

Nuestro Proyecto: VQC vs. SVM Clásico en Qiskit

Este proyecto busca diseñar e implementar un clasificador cuántico (VQC) y compararlo rigurosamente con un método clásico bien establecido (SVM), evaluando su desempeño y potencial en problemas específicos.

VQC SVM QISKIT

1

Objetivo General del Proyecto

Nuestra misión es construir un clasificador cuántico funcional, compararlo con un modelo clásico equivalente y medir su valor real, explorando sus limitaciones actuales y el potencial futuro del Quantum Machine Learning.

- Construir un clasificador cuántico funcional.
- Compararlo con un modelo clásico (SVM).
- Medir la eficiencia y el valor del QML.

2

Los Dos Contendientes: SVM Clásico vs. VQC Cuántico

Analizamos las características y configuraciones de cada modelo para una comparación justa. El SVM, con su solidez matemática, contrasta con el VQC, que explota las propiedades cuánticas para una exploración de soluciones única.

- **SVM**: Algoritmo clásico, kernel RBF, Scikit-Learn. Rápido, confiable, accesible, pero limitado por la arquitectura clásica.
- **VQC**: Modelo híbrido, combina Feature Map, Ansatz entrenable y Optimizador Clásico (COBYLA). Aprovecha superposición y entrelazamiento, con potencial en hardware cuántico real.

1

1. Comprender Fundamentos del QML

Adquirir una base teórica sólida en Quantum Machine Learning, circuitos cuánticos variacionales, qubits y puertas cuánticas. Distinguir el QML del ML clásico antes de la implementación práctica.

2

2. Implementar Modelo Cuántico Concreto

Desarrollar un VQC utilizando Qiskit Machine Learning, configurando un ZZFeatureMap con 4 qubits y un Ansatz RealAmplitudes. Integrar el optimizador COBYLA para asegurar un funcionamiento de principio a fin.

3

3. Comparar Resultados Rigurosamente

Medir métricas clave como precisión, tiempo de cómputo, precision, recall y F1-Score. Utilizar el mismo dataset (Iris binario), split train/test (70/30) y preprocesamiento (MinMax scaling) para una comparación científica.

4

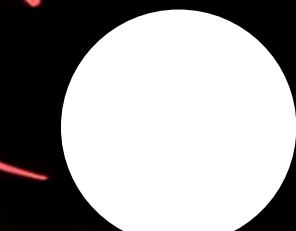
4. Analizar Limitaciones Actuales

Investigar las razones por las que el VQC puede ser más lento que el SVM, los desafíos del QML actual y cuándo tiene sentido usar computación cuántica. Ofrecer una perspectiva realista del estado del arte.

Fundamentos de Computación Cuántica

Bit Clásico

0



1

Poder Exponencial

4

qubits

16 estados

10

qubits

1024 estados

Qubit

0



1

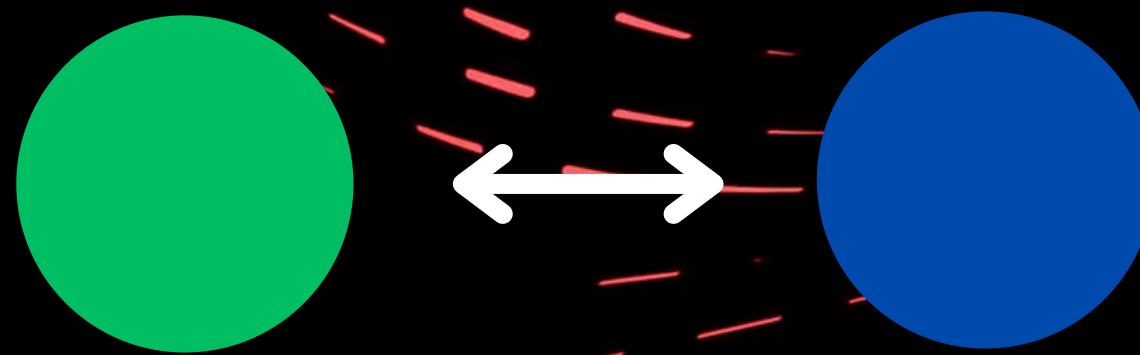
300

qubits

>átomos
del universo

Entrelazamiento cuántico

Sistema Clásico



Sistema Cuántico



correlacionados instantáneamente

"Acción fantasmal a distancia"

- Albert Einstein

Permite capturar correlaciones complejas entre datos

Puertas Cuánticas

H
Hadamard

Crea superposición
 $|0\rangle \rightarrow (|0\rangle + |1\rangle)/\sqrt{2}$

⊕

CNOT

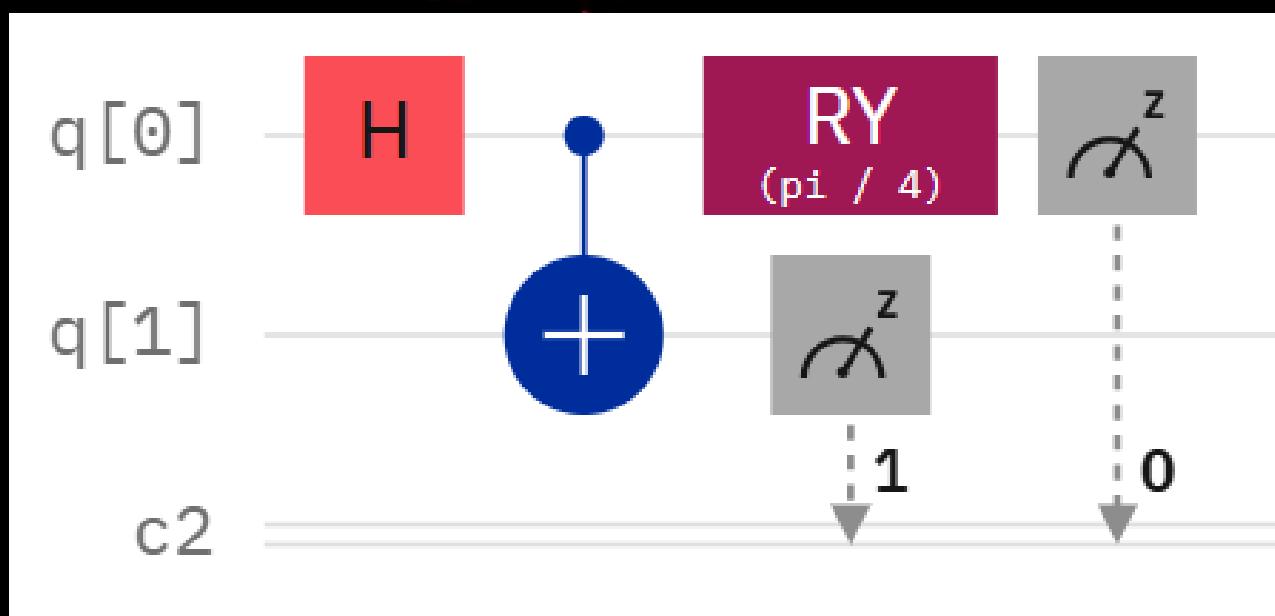
Crea entrelazamiento
Control → Target

R θ

Rotaciones

Parámetros que se
entrenan
 $R_x(\theta), R_y(\theta), R_z(\theta)$

Ejemplo de circuito cuántico



Variational Quantum Classifier (VQC)

Es un modelo de Machine Learning híbrido que combina circuitos cuánticos parametrizados con optimización clásica para realizar clasificación de datos.

1. Feature Map

Codifica datos clásicos en estados cuánticos

$$x_1, x_2, x_3, x_4 \rightarrow |\psi\rangle$$

2. Ansatz

Circuito variacional con parámetros $\theta_1, \theta_2, \dots, \theta_n$

3. Medición

Extrae predicción del estado cuántico
 $|\psi\rangle \rightarrow$ Clase 0 ó 1

Aprovecha la superposición y el entrelazamiento para explorar espacios de características de alta dimensión

Componentes de Nuestro VQC

ZZFeatureMap

Transforma datos clásicos (Iris) a estados cuánticos usando entrelazamiento ZZ
`feature_map = ZZFeatureMap(feature_dimension=4, reps=2)`

RealAmplitudes Ansatz

Círculo variacional con rotaciones Ry y entrelazamiento CNOT
`ansatz = RealAmplitudes(num_qubits=4, reps=3)`

Optimizador COBYLA

Optimización clásica libre de gradientes, ideal para sistemas con ruido
`optimizer = COBYLA(maxiter=100)`

Dataset y Preprocesamiento de Datos

Preparando los datos para el aprendizaje cuántico

DATASET: IRIS

- Dataset clásico de Machine Learning con 150 muestras de flores
- Contiene 4 características: longitud y ancho de sépalos y pétalos
- Originalmente tiene 3 clases (especies de iris)

FILTRADO PARA CLASIFICACIÓN BINARIA

- Seleccionamos solo 2 clases: Setosa y Versicolor
- Resultado: 100 muestras totales
- Razón: Simplificar la prueba de concepto y reducir carga computacional en el simulador cuántico

PREPROCESAMIENTO CRÍTICO

- Normalización con MinMaxScaler: rango [0, 1]
- ¿Por qué es crucial?
 - Las puertas cuánticas operan con ángulos y amplitudes
 - Valores grandes causan inestabilidad en el circuito
 - El feature map cuántico requiere datos normalizados

DIVISIÓN DE DATOS

- 70% entrenamiento (70 muestras)
- 30% prueba (30 muestras)
- División estratificada para mantener proporción de clases

Stack Tecnológico y Herramientas

Combinando lo mejor del mundo clásico y cuántico

Python 3.x

Lenguaje base del proyecto

NumPy & Pandas

Manipulación de datos
y operaciones numéricas

Matplotlib & Seaborn

Visualización de resultados
y análisis comparativo

Qiskit 1.x

Framework cuántico de IBM

- Feature Maps
- Circuitos variacionales
- Simulación cuántica

Qiskit Machine Learning

Algoritmos de QML

- VQC (Variational Quantum Classifier)
- Optimizadores cuánticos

Scikit-Learn

Modelo de referencia

- SVM con kernel RBF
- Métricas de evaluación
- Validación cruzada

Benchmark clásico para
comparar con modelo cuántico

Nota:

- 💡 Enfoque Híbrido: Combinamos algoritmos cuánticos con optimización clásica para aprovechar lo mejor de ambos mundos

Flujo de Trabajo del Proyecto

Contenido - Diagrama de Flujo (Vertical)

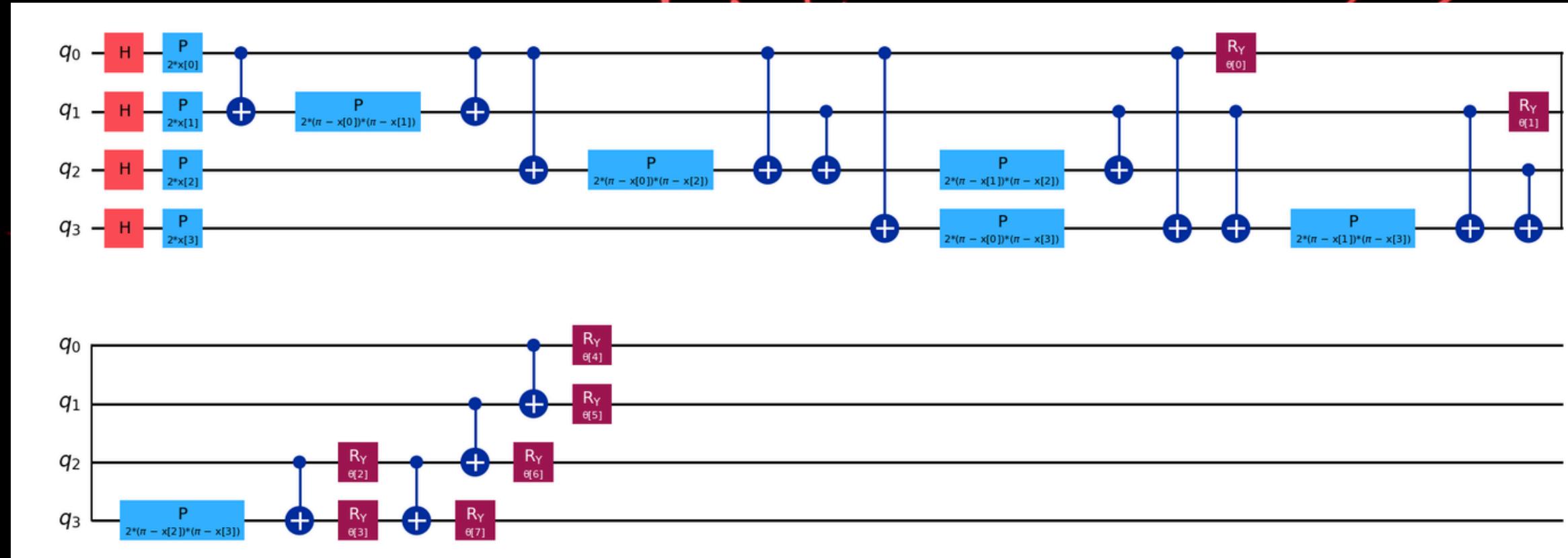
Características Clave:

- Reproducibilidad: Semilla aleatoria fija (42)
- Validación: Conjunto de test independiente
- Comparación justa: Mismo dataset para ambos modelos
- Simulación cuántica: StatevectorSampler de Qiskit



Arquitectura del modelo VQC (Clasificador Cuántico Variacional)

Arquitectura del circuito cuántico



Para construir este modelo, no usamos neuronas tradicionales, sino un circuito cuántico dividido en tres etapas fundamentales que ven aquí en pantalla:

- Primero, el Feature Map: Como las computadoras cuánticas no entienden nuestros datos clásicos directamente, usamos el ZZFeatureMap. Este componente toma los datos del dataset Iris y los 'proyecta' a un espacio cuántico de alta dimensión utilizando el fenómeno del entrelazamiento. Esto es clave porque nos permite encontrar patrones que modelos lineales simples no verían.
- Segundo, el Ansatz: Esta es la parte que realmente 'aprende'. Usamos una estructura llamada RealAmplitudes. Imaginen esto como las capas ocultas de una red neuronal; tiene compuertas con parámetros rotacionales que iremos ajustando.
- Y tercero, el Optimizador Clásico: Usamos COBYLA. Dado que las computadoras cuánticas actuales tienen ruido, este optimizador es ideal porque no requiere calcular gradientes complejos, haciendo el entrenamiento más estable.

Implementación en Código VQC

Implementación con Qiskit Machine Learning

1. Configuración Modular

(Feature Map + Ansatz + Optimizador)

```
# 1. Feature Map
feature_map = ZZFeatureMap(feature_dimension=num_features, reps=1)
print(f"\n  Feature Map configurado:")
print(f"    - Dimensión: {num_features}")
print(f"    - Repeticiones: 1")
print(f"    - Tipo: ZZ (interacción entre pares de qubits)")

# 2. Ansatz (circuito variacional)
ansatz = RealAmplitudes(num_qubits=num_features, reps=1)
num_params = ansatz.num_parameters
print(f"\n  Ansatz configurado:")
print(f"    - Qubits: {num_features}")
print(f"    - Repeticiones: 1")
print(f"    - Parámetros entrenables: {num_params}")

# 3. Optimizador
optimizer = COBYLA(maxiter=100)
print(f"\n  Optimizador configurado:")
print(f"    - Tipo: COBYLA (Constrained Optimization BY Linear Approximation)")
print(f"    - Máx iteraciones: 100")
print(f"    - Sin gradientes (gradient-free)")
```

2. Ensamblaje del Modelo

(Unificación en un solo objeto)

```
# 5. Crear VQC
vqc = VQC(
    sampler=sampler,
    feature_map=feature_map,
    ansatz=ansatz,
    optimizer=optimizer
)
```

3. Ejecución Híbrida

(Compatible con Scikit-Learn))

```
# Entrenar
vqc.fit(X_train, y_train)
```

Pasando al código, utilizamos la librería Qiskit Machine Learning para implementar la arquitectura que vimos en el diagrama anterior. Lo dividimos en tres componentes configurables:

- Primero: Definimos el ZZFeatureMap (arriba), que se encarga de transformar nuestros datos clásicos al mundo cuántico.
- Segundo: Configuramos el Ansatz (centro), que es el circuito variacional que contiene los parámetros que la IA va a entrenar.
- Tercero: Inicializamos COBYLA (abajo), nuestro optimizador clásico libre de gradientes, esencial para navegar el ruido del sistema.

Una vez configuradas las piezas, las ensamblamos en el bloque número 2.

Aquí creamos la instancia del VQC (Variational Quantum Classifier). Simplemente le pasamos el mapa de características, el circuito ansatz y el optimizador. Qiskit se encarga de conectar todo internamente.

Y finalmente, el paso más importante: el entrenamiento.

vqc.fit(X_train, y_train).

Si alguno ha trabajado con Inteligencia Artificial clásica en Python, reconocerá este comando. Es idéntico a como se entrena un modelo en Scikit-Learn.

Esto es fundamental porque demuestra que la tecnología cuántica ya está lista para integrarse en flujos de trabajo de ciencia de datos tradicionales sin que tengamos que aprender un lenguaje de programación completamente nuevo desde cero.

Resultados y Comparación

Comparación: Clásico VS Cuántico

```

1 print("=" * 80)
2 print("蝶 COMPARACIÓN DIRECTA: CLÁSICO vs CUÁNTICO")
3 print("=" * 80)
4
5 # Crear tabla comparativa
6 comparison_data = {
7     'Métrica': ['Accuracy (Train)', 'Accuracy (Test)', 'Precision', 'Recall', 'F1-Score', 'Tiempo (s)'],
8     'SVM Clásico': [
9         f'{acc_classic_train:.4f}',
10        f'{acc_classic_test:.4f}',
11        f'{precision_classic:.4f}',
12        f'{recall_classic:.4f}',
13        f'{f1_classic:.4f}',
14        f'{classic_train_time:.4f}'
15    ],
16    'VQC Cuántico': [
17        f'{acc_quantum_train:.4f}',
18        f'{acc_quantum_test:.4f}',
19        f'{precision_quantum:.4f}',
20        f'{recall_quantum:.4f}',
21        f'{f1_quantum:.4f}',
22        f'{quantum_train_time:.4f}'
23    ]
24}
25
26 df_comparison = pd.DataFrame(comparison_data)
27 print("\n", df_comparison.to_string(index=False))
28
29 # Análisis de diferencias
30 print(f"\n蝶 Análisis de Diferencias:")
31 print(f"  • Diferencia en Accuracy (Test): {abs(acc_classic_test - acc_quantum_test):.4f}")
32 print(f"  • Factor de tiempo (Quantum/Classic): {quantum_train_time/classic_train_time:.2f}x más lento")
33
34 if acc_quantum_test >= acc_classic_test:
35     print(f"蝶 El modelo cuántico logró IGUAL o MEJOR accuracy")
36 else:
37     print(f"⚠️ El modelo clásico logró mejor accuracy (diferencia: {(acc_classic_test - acc_quantum_test)*100:.2f}%)")

```

```

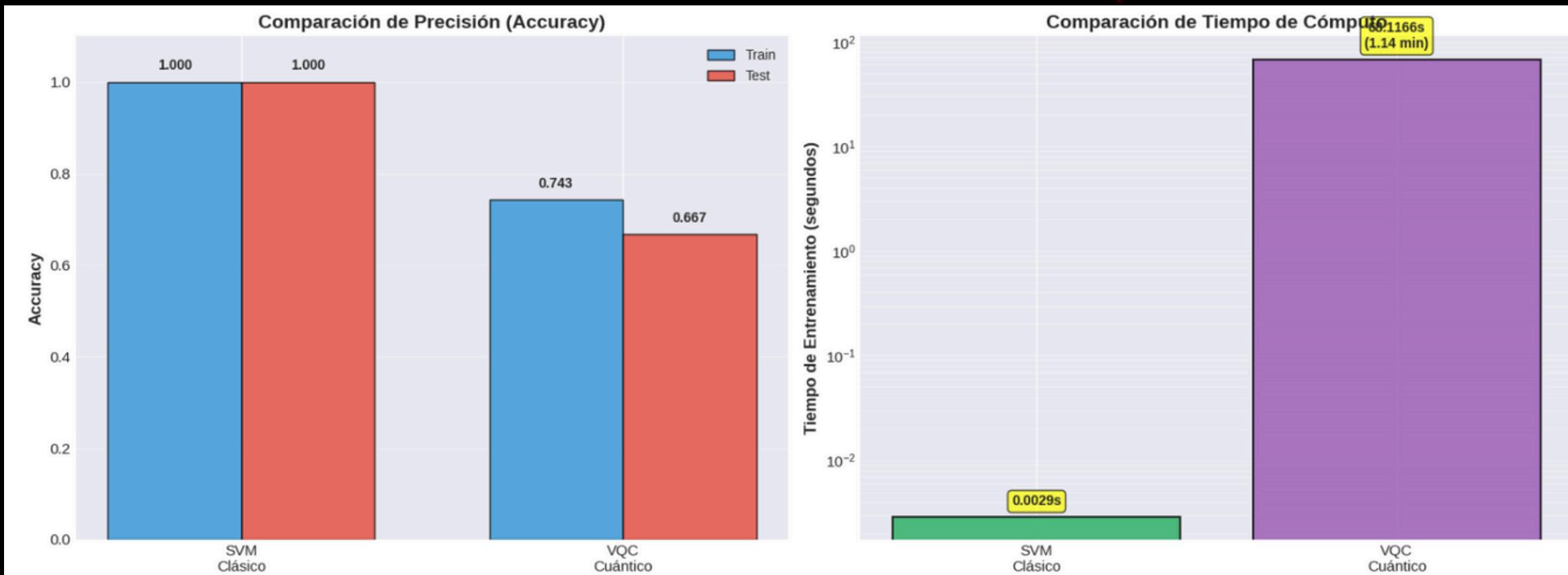
=====
蝶 COMPARACIÓN DIRECTA: CLÁSICO vs CUÁNTICO
=====

Métrica SVM Clásico VQC Cuántico
Accuracy (Train) 1.0000 0.7429
Accuracy (Test) 1.0000 0.6667
Precision 1.0000 0.6667
Recall 1.0000 0.6667
F1-Score 1.0000 0.6667
Tiempo (s) 0.0029 68.1166

🔍 Análisis de Diferencias:
• Diferencia en Accuracy (Test): 0.3333
• Factor de tiempo (Quantum/Classic): 23535.86x más lento
⚠️ El modelo clásico logró mejor accuracy (diferencia: 33.33%)

```

Gráficas



Precisión (Accuracy)

En la Figura de Comparación de Precisión, se observa que:

- El SVM clásico alcanzó:
 - 100% en entrenamiento
 - 100% en prueba
- El VQC cuántico obtuvo:
 - 74.29% en entrenamiento
 - 66.67% en prueba

Aunque el modelo cuántico no iguala la precisión del modelo clásico, sí demuestra capacidad de aprendizaje real. El VQC logra capturar patrones del dataset y clasificar correctamente una proporción significativa de los datos de prueba, lo que confirma que el pipeline cuántico funciona y no es aleatorio.

Tiempo de Cómputo (Punto Crítico)

En la gráfica de tiempo de entrenamiento, la diferencia es clara y contundente:

- SVM clásico:
 - ~0.0029 segundos
- VQC cuántico:
 - ~68.12 segundos (~ 1.14 minutos)

Esto implica que el modelo cuántico fue aproximadamente:

- 23,535 veces más lento que el modelo clásico.

¿Por qué ocurre esta diferencia tan grande?

Porque el VQC se ejecuta sobre un simulador cuántico clásico, lo cual implica:

- Simular vectores de estado cuánticos de dimensión exponencial.
- Evaluar múltiples circuitos por iteración del optimizador.
- Combinar optimización clásica + simulación cuántica.

Resultados y Comparación

Matrices de Confusión Comparativas

```

1 # Matriz de Confusión: Lado a Lado
2 fig, axes = plt.subplots(1, 2, figsize=(14, 6))
3
4 # Matriz Clásica
5 cm_classic = confusion_matrix(y_test, y_pred_classic_test)
6 sns.heatmap(cm_classic, annot=True, fmt='d', cmap='Blues', ax=axes[0],
7             cbar_kws={'label': 'Cantidad de Muestras'},
8             xticklabels=['Setosa', 'Versicolor'],
9             yticklabels=['Setosa', 'Versicolor'])
10 axes[0].set_xlabel('Predicción', fontsize=12, fontweight='bold')
11 axes[0].set_ylabel('Valor Real', fontsize=12, fontweight='bold')
12 axes[0].set_title(f'Matriz de Confusión - SVM Clásico\n(Accuracy: {acc_classic_test:.3f})', fontsize=13, fontweight='bold')
13
14 # Matriz Cuántica
15 cm_quantum = confusion_matrix(y_test, y_pred_quantum_test)
16 sns.heatmap(cm_quantum, annot=True, fmt='d', cmap='Purples', ax=axes[1],
17             cbar_kws={'label': 'Cantidad de Muestras'},
18             xticklabels=['Setosa', 'Versicolor'],
19             yticklabels=['Setosa', 'Versicolor'])
20 axes[1].set_xlabel('Predicción', fontsize=12, fontweight='bold')
21 axes[1].set_ylabel('Valor Real', fontsize=12, fontweight='bold')
22 axes[1].set_title(f'Matriz de Confusión - VQC Cuántico\n(Accuracy: {acc_quantum_test:.3f})', fontsize=13, fontweight='bold')
23
24 plt.tight_layout()
25 plt.show()
26
27 # Análisis de errores
28 print("\nAnálisis de Matrices de Confusión:")
29 print("\nSVM Clásico:")
30 print("• Verdaderos Positivos (Clase 1 correcta):", cm_classic[1,1])
31 print("• Verdaderos Negativos (Clase 0 correcta):", cm_classic[0,0])
32 print("• Falsos Positivos (Clase 0 predicha como 1):", cm_classic[0,1])
33 print("• Falsos Negativos (Clase 1 predicha como 0):", cm_classic[1,0])
34
35 print("\nVQC Cuántico:")
36 print("• Verdaderos Positivos (Clase 1 correcta):", cm_quantum[1,1])
37 print("• Verdaderos Negativos (Clase 0 correcta):", cm_quantum[0,0])
38 print("• Falsos Positivos (Clase 0 predicha como 1):", cm_quantum[0,1])
39 print("• Falsos Negativos (Clase 1 predicha como 0):", cm_quantum[1,0])

```

Análisis de Matrices de Confusión:

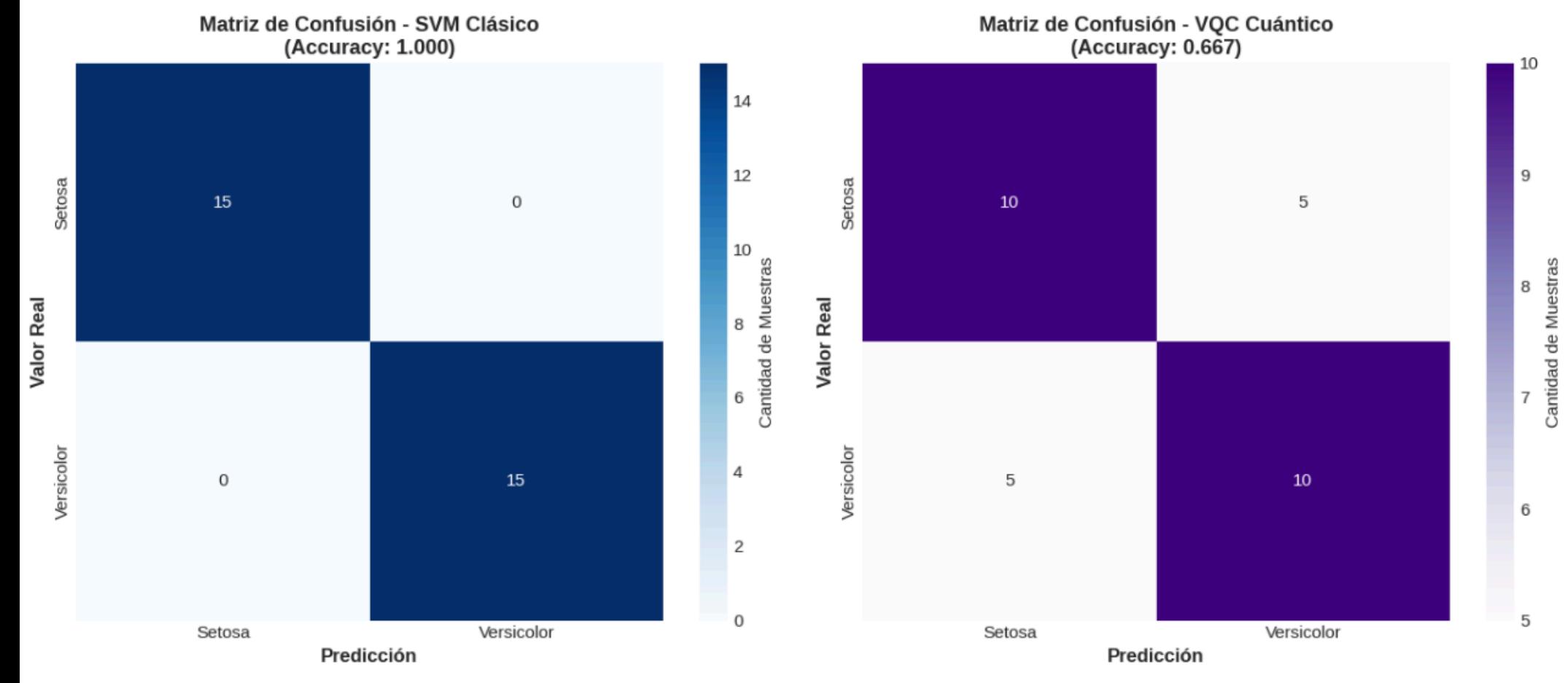
SVM Clásico:

- Verdaderos Positivos (Clase 1 correcta): 15
- Verdaderos Negativos (Clase 0 correcta): 15
- Falsos Positivos (Clase 0 predicha como 1): 0
- Falsos Negativos (Clase 1 predicha como 0): 0

VQC Cuántico:

- Verdaderos Positivos (Clase 1 correcta): 10
- Verdaderos Negativos (Clase 0 correcta): 10
- Falsos Positivos (Clase 0 predicha como 1): 5
- Falsos Negativos (Clase 1 predicha como 0): 5

Gráficas



SVM Clásico – Clasificación Perfecta

La matriz de confusión del SVM clásico muestra un desempeño ideal:

- Verdaderos Positivos (VP): 15
- Verdaderos Negativos (VN): 15
- Falsos Positivos (FP): 0
- Falsos Negativos (FN): 0

El modelo clásico clasificó correctamente el 100% de las muestras de prueba, sin cometer ningún tipo de error.

Esto indica que:

- Las clases son linealmente separables en el espacio de características.
- El SVM encontró un hiperplano óptimo.
- El modelo generaliza perfectamente en este escenario.

Interpretación:

El modelo cuántico clasifica correctamente 20 de 30 muestras, lo que corresponde a una accuracy del 66.7%.

Los errores están simétricamente distribuidos entre ambas clases:

- 5 instancias de Setosa fueron clasificadas erróneamente como Versicolor.
- 5 instancias de Versicolor fueron clasificadas erróneamente como Setosa.

Qué nos dice esto:

- El modelo sí aprende, pero no logra separar completamente las clases.
- No existe sesgo hacia una clase específica.
- El límite de decisión cuántico es menos expresivo que el clásico en este caso.

Conclusiones

En conclusión, el porcentaje de precision obtenido para el modelo quantico hibrido VQC nos indica que es un modelo que puede aprender, este resultado se demostró en los datos obtenidos en la matriz de confusión, donde ambos modelos obtuvieron resultados favorables en las mismas condiciones.

Y si bien los resultados del modelo clásico fueron mejores que el cuantico en cuanto a precision, esto dependerá tambien de la complejidad y del tipo de problema y datos que se utilizaran, ya que en entornos más complejos se obtendrían resultados más favorables para el modelo VQC, ademas de que el tiempo de ejecución siempre sera menor en el modelo VQC.

Bibliografía

- Benedetti, M., Lloyd, E., Sack, S., & Fiorentini, M. (2019). Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4), 043001. <https://doi.org/10.1088/2058-9565/ab4eb5>
- Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747), 209–212.
- Abbe, E. (2018). Community detection and stochastic block models. *Journal of Machine Learning Research*, 18(177), 1–86. <https://www.jmlr.org/papers/v18/16-480.html>
- Fortunato, S., & Hric, D. (2016). Community detection in networks: A user guide. *Physics Reports*, 659, 1–44. <https://doi.org/10.1016/j.physrep.2016.09.002>

Muchas Gracias