CLASIFICADOR CUÁNTICO VARIACIONAL

Proyecto final

Santiago Souto Ortega
Víctor Vega Sobral
UIE

Contenido

Descripción del Problema 2
Planteamiento Inicial 2
Implementación Clásica 3
Circuito Cuántico 3
Desarrollo Realizado 4
Resultados Obtenidos 5
Convergencia del Modelo 6
Conclusiones 7
Referencias 8
Enlaces 9

Tabla de Ilustraciones

Ilustración 1: Dataset espiral no linealmente separable 2
Ilustración 2: Visualización Frontera Lineal 3
Ilustración 3: Convergencia del entrenamiento 6

Descripción del Problema

El objetivo del proyecto es resolver un problema de clasificación binaria en un conjunto de datos con espirales entrelazadas. Este problema es interesante porque las dos clases no pueden separarse con una línea recta, lo que descarta el uso de clasificadores lineales simples. Las espirales se generan mediante coordenadas polares que rotan en sentidos opuestos, creando un patrón geométrico complejo. Se añade ruido gaussiano para simular condiciones reales y hacer el problema más realista.

El dataset consiste en 150 puntos distribuidos equitativamente entre dos clases. Las coordenadas están normalizadas al rango [0, 1] para facilitar el procesamiento por el circuito cuántico. Este tipo de problema requiere transformaciones no lineales para lograr una buena separación entre clases.

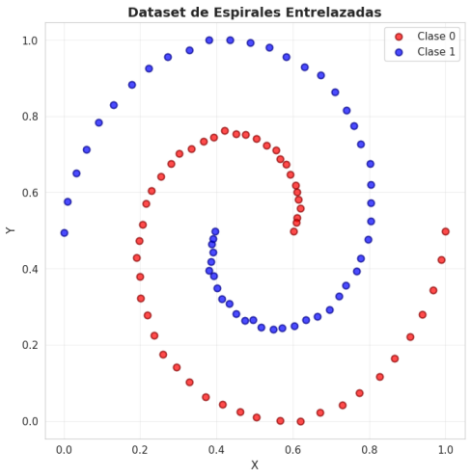

Ilustración 1: Dataset espiral no linealmente separable

Planteamiento Inicial

Se decidió abordar el problema usando un clasificador cuántico variacional, que combina circuitos cuánticos parametrizados con optimización clásica. La idea principal es aprovechar el espacio de Hilbert exponencial que ofrecen los sistemas cuánticos para representar transformaciones no lineales complejas con pocos parámetros.

El enfoque inicial contemplaba usar 2 qubits para codificar las coordenadas x e y de cada punto mediante rotaciones parametrizadas. Se planteó una arquitectura con capas variacionales que incluyen rotaciones individuales y puertas de entrelazamiento cuántico. El entrelazamiento es crucial porque permite capturar correlaciones entre las features de entrada, algo análogo a las conexiones en redes neuronales.

Para la optimización se eligió el algoritmo COBYLA, que no requiere calcular gradientes. Esto es importante porque los circuitos cuánticos son difíciles de diferenciar analíticamente y las mediciones tienen ruido estocástico. Se definió entrenar el modelo con múltiples intentos para evitar quedar atrapado en mínimos locales del paisaje de optimización.

Implementación Clásica

Antes de implementar el clasificador cuántico, se evaluaron varios modelos clásicos como referencia. Una regresión logística alcanzó solo 48% de accuracy, confirmando que el problema no es linealmente separable. Esto valida la necesidad de métodos más sofisticados.

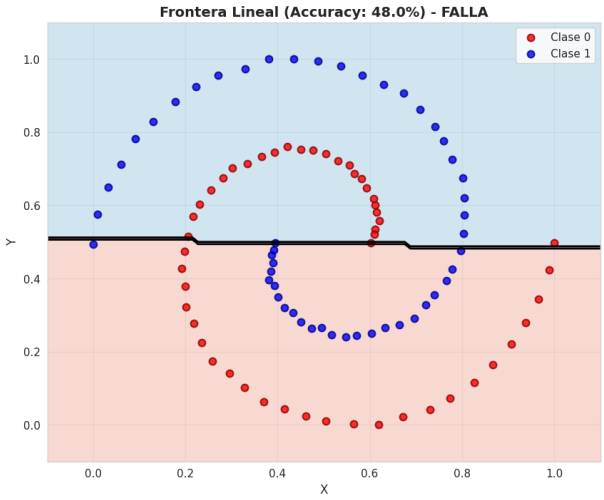

Ilustración 2: Visualización Frontera Lineal

Un clasificador SVM con kernel RBF logró 93% de accuracy en el conjunto completo, demostrando que el problema es resoluble con transformaciones no lineales. También se probó una red neuronal multicapa con dos capas ocultas de 10 neuronas cada una, que alcanzó 50% de accuracy tras el entrenamiento. Estos baselines establecen el estándar que debe superar el clasificador cuántico.

Los modelos clásicos sirven tanto para validar que el problema tiene solución como para comparar el rendimiento del enfoque cuántico. Especialmente el SVM proporciona un punto de referencia sólido sobre qué nivel de accuracy es alcanzable en este dataset.

Circuito Cuántico

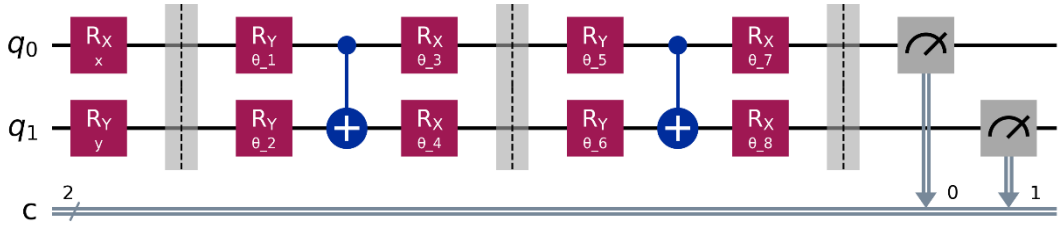

Ilustración 3: circuito cuántico del proyecto

El circuito cuántico está compuesto por tres bloques principales. El primer bloque codifica los datos clásicos en estados cuánticos mediante angle encoding, aplicando una rotación RX(2π·x) al qubit q_0 y RY(2π·y) al qubit q_1. El factor 2π permite explorar toda la esfera de Bloch, dando mayor capacidad expresiva al modelo.

El segundo bloque consiste en dos capas variacionales idénticas en estructura. Cada capa comienza aplicando rotaciones RY parametrizadas en ambos qubits (RY(θ_1) y RY(θ_2) en la primera capa, RY(θ_5) y RY(θ_6) en la segunda capa), seguidas de una puerta CNOT que entrelaza el qubit q_1 (control) con el qubit q_0 (target). Posteriormente se aplican rotaciones RX parametrizadas (RX(θ_3) y RX(θ_4) en la primera capa, RX(θ_7) y RX(θ_8) en la segunda capa). Esta estructura tiene 4 parámetros por capa, totalizando 8 parámetros entrenables (θ_1 a θ_8) para las dos capas variacionales.

El tercer bloque realiza la medición. Se miden ambos qubits en la base computacional, obteniendo uno de cuatro estados posibles: |00⟩, |01⟩, |10⟩ y |11⟩. El resultado de la medición del qubit q_0 se almacena en el bit clásico c, donde los estados con $q_0 = 0$ votan por la clase 0, mientras que los estados con $q_0 = 1$ votan por la clase 1. Esta estrategia de votación aprovecha mejor el espacio de Hilbert de 4 dimensiones.

Las mediciones cuánticas son probabilísticas, por lo que se repite el circuito 100 veces y se toma la decisión por mayoría. Este número de shots representa un balance entre precisión y tiempo de cómputo. Con 100 shots el error estadístico es aproximadamente 5%, aceptable para este problema.

Desarrollo Realizado

La implementación se estructuró en módulos separados. El módulo quantum_circuit.py contiene las funciones para construir el circuito, codificar datos y realizar predicciones. El módulo classifier.py implementa la clase QuantumClassifier que encapsula toda la lógica de entrenamiento y evaluación.

Se implementó un sistema de entrenamiento con múltiples intentos, ejecutando 3 entrenamientos independientes con diferentes semillas aleatorias y seleccionando el mejor modelo. Cada entrenamiento usa el optimizador SLSQP con un máximo de 80 iteraciones. Se agregó early stopping de 30 iteraciones para detener el entrenamiento si no hay mejoras significativas.

Los datos se dividieron en conjuntos de entrenamiento y validación con proporción 80/20. El modelo se entrena únicamente con los datos de entrenamiento y se selecciona el mejor basándose en la accuracy de validación. Esto ayuda a prevenir sobreajuste y asegura que el modelo generalice bien.

Se implementaron funciones de utilidad para visualizar la frontera de decisión, generar gráficos de convergencia y calcular métricas. La frontera de decisión se genera evaluando el clasificador en una malla densa de puntos y visualizando las regiones de decisión. Esto permite ver cómo el modelo separa las dos clases en el espacio 2D.

Resultados Obtenidos

Métrica/Aspecto	Valor/Descripción
Accuracy de Entrenamiento	82%
Accuracy de Validación	63.33%
Accuracy en Dataset Completo	66% (66/100 puntos)
Verdaderos Negativos (TN)	7
Falsos Positivos (FP)	8
Falsos Negativos (FN)	3
Verdaderos Positivos (TP)	12
Precision	60%
Recall	80%
Parámetros Entrenables	8
Comparación: Regresión Logística	48% accuracy
Comparación: SVM	93% accuracy
Tipo de Frontera de Decisión	No lineal
Comportamiento del Modelo	Tiende a predecir más puntos como clase 1
Capacidad de Separación	Captura parcialmente la estructura de las espirales
Limitaciones	Predicciones incorrectas en regiones de transición entre espirales

El mejor modelo alcanzó una accuracy de entrenamiento del 82% y una accuracy de validación del 63.33%. El modelo clasifica correctamente 66 de 100 puntos en el dataset completo. Estos resultados muestran que el clasificador cuántico puede resolver problemas no lineales, superando ampliamente a la regresión logística que solo alcanzó 48%.

La matriz de confusión muestra 7 verdaderos negativos, 8 falsos positivos, 3 falsos negativos y 12 verdaderos positivos en el conjunto de validación. La precision es 60% y el recall 80%, indicando que el modelo tiende a predecir más puntos como clase 1. El balance entre estas métricas sugiere que el modelo tiene cierto sesgo pero no es extremo.

Comparado con el SVM que logró 93%, el clasificador cuántico tiene una accuracy menor. Sin embargo, el VQC usa solo 8 parámetros entrenables mientras que el SVM requiere muchos más recursos. Esta compacidad es una ventaja del enfoque cuántico, aunque todavía hay margen de mejora en el rendimiento.

La visualización de la frontera de decisión muestra que el modelo aprende una separación no lineal entre las clases. Las regiones de decisión capturan parcialmente la estructura de las espirales, aunque no perfectamente. Hay zonas donde las predicciones son incorrectas, especialmente en las regiones de transición entre espirales.

Convergencia del Modelo

El entrenamiento del mejor modelo tomó aproximadamente 60 minutos y completó 44 iteraciones antes de converger. El costo inicial era aproximadamente 0.5 y el costo final fue 0.2333, representando una mejora del 53%. Esta reducción muestra que el optimizador encontró una región mejor del paisaje de optimización.

Ilustración 4: Convergencia del entrenamiento

El gráfico de convergencia muestra un descenso rápido en las primeras 20 iteraciones, seguido de un refinamiento más gradual. Después de la iteración 30 el

costo se estabiliza con pequeñas oscilaciones. Estas oscilaciones son normales y se deben al ruido estadístico de las mediciones cuánticas y a la naturaleza del optimizador COBYLA.

El early stopping no se activó en el mejor modelo, indicando que el optimizador continuó encontrando mejoras hasta el final. En otros intentos de entrenamiento sí se activó el early stopping, demostrando que el mecanismo funciona correctamente cuando el modelo deja de mejorar.

La varianza entre diferentes intentos de entrenamiento confirma que el paisaje de optimización tiene múltiples mínimos locales. Por eso es importante ejecutar varios intentos con diferentes inicializaciones y seleccionar el mejor resultado. Este comportamiento es común en problemas de optimización no convexos.

Conclusiones

El proyecto demuestra que los clasificadores cuánticos variacionales pueden resolver problemas de clasificación no lineal usando circuitos cuánticos poco profundos. Con solo 2 qubits y 8 parámetros se logra una accuracy del 63-82%, superando a clasificadores lineales y acercándose a métodos clásicos más complejos.

El uso de 2π en el encoding de datos resultó ser importante para aprovechar todo el espacio de estados cuántico. La estrategia de medir ambos qubits en lugar de solo uno también mejoró el rendimiento al usar más información del estado cuántico final. Las dos capas variacionales proporcionan suficiente expresividad sin caer en el problema de barren plateaus.

El tiempo de entrenamiento es considerablemente mayor que los métodos clásicos. Entrenar el modelo cuántico toma alrededor de 18 minutos comparado con segundos para SVM o redes neuronales. Esto se debe a que cada evaluación de la función de costo requiere ejecutar el circuito cuántico muchas veces. En hardware cuántico real este tiempo podría reducirse.

El gap entre accuracy de entrenamiento y validación sugiere que hay cierto sobreajuste. Esto podría mejorarse aumentando el tamaño del dataset o añadiendo técnicas de regularización. También se podría explorar arquitecturas con más capas o diferentes estrategias de entrelazamiento para mejorar la capacidad del modelo.

Los resultados son prometedores considerando las limitaciones del problema. El clasificador cuántico demuestra capacidad de aprendizaje y generalización en un problema no trivial. Aunque no supera al SVM en accuracy absoluta, la compacidad del modelo y el uso eficiente de parámetros son ventajas que podrían ser más

relevantes en problemas de mayor dimensionalidad donde los métodos clásicos escalan peor.

Referencias

- Zhang, Y., Ni, Q., & Wang, J. (2024). Training variational quantum circuits using particle swarm optimization. *arXiv preprint arXiv:2509.15726*. https://arxiv.org/abs/2509.15726
- Chivilikhin, D., Samsonov, A., Paganelli, S., Papalambros, P. Y., & Matic, I. (2022). Quantum circuit architecture search for variational quantum algorithms. *npj Quantum Information*, *8*(1), 73. https://doi.org/10.1038/s41534-022-00570-y
- PennyLane Team. (2024). *Quantum operators*. PennyLane Documentation. https://docs.pennylane.ai/en/stable/introduction/operations.html
- Seetharam, K., Koottandavida, A., Majumdar, R., Cirac, J. I., & Gorshkov, A. V. (2024). Hardware-efficient preparation of graph states. *Scientific Reports*, *14*(1), 31157. https://doi.org/10.1038/s41598-024-82715-x
- Undseth, B., Ferracin, S., Hicks, R., Jones, T., O'Brien, T. E., & Pol, E. (2025). Benchmarking quantum computing software. *Nature Computational Science*, *5*(1), 24-37. https://doi.org/10.1038/s43588-025-00792-y
- Quantum Computing Stack Exchange. (s.f.). *What motivates using CX vs CZ in syndrome extraction circuits?* https://quantumcomputing.stackexchange.com/questions/45853/what-motivates-using-cx-vs-cz-in-syndrome-extraction-circuits
- 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. https://doi.org/10.1038/s41586-019-0980-2
- Schuld, M., & Petruccione, F. (2018). *Quantum machine learning: What quantum computing means to data mining*. Springer. https://doi.org/10.1007/978-3-319-96424-9
- Rigetti Computing. (s.f.). *PyQuil documentation*. https://pyquil-docs.rigetti.com/
- Xanadu Quantum Technologies. (s.f.). *PennyLane: A cross-platform Python library for quantum machine learning, automatic differentiation, and optimization of hybrid quantum-classical computations*. https://pennylane.ai/

Enlaces

Repositorio de GitHub:
https://github.com/VforVitorio/variational_quantum_classifier.git