Quantum machine learning (QML) là chủ đề nghiên cứu giao thoa giữa machine learning và quantum computing. Vì yêu cầu kiến thức ở cả hai lĩnh vực, nên sẽ tương đối khó nếu chúng ta bắt đầu ở project tầm cao nên mình sẽ hướng dẫn các bạn nhập môn thông qua thuật toán Quantum K - nearest neighboor (QKNN). Bài hướng dẫn bao gồm ba phần: lý thuyết cơ sở, giới thiệu và thực hành, các bạn thực hiện theo thứ tự từ trên xuống hoặc phần 1 trước, sau đó xen kẽ phần 2 và phần 3.

1. Lý thuyết cơ sở

Các khái niệm cần nắm về machine learning:

- Classification: là bài toán yêu cầu xác định label của một đối tượng chưa biết trước. Ví dụ: Tin nhắn thuộc loại bình thường, quan trọng hay spam.
- Supervised learning: học giám sát.
- Dataset: train set, test set, data point.
- Data point: data, label.
- Noise.

Tài liệu về machine learning:

- Nhập môn: cần đọc kĩ.
- kNN: cần đọc kĩ.

Các khái niệm cần nắm về quantum computing:

- Qubit, quantum state, operator.
- Superposition.
- Quantum gate: Hadamard gate, CNOT gate, SWAP gate, Fredkin gate, Rotation gate.
- Measurement cơ bản (thống kê)

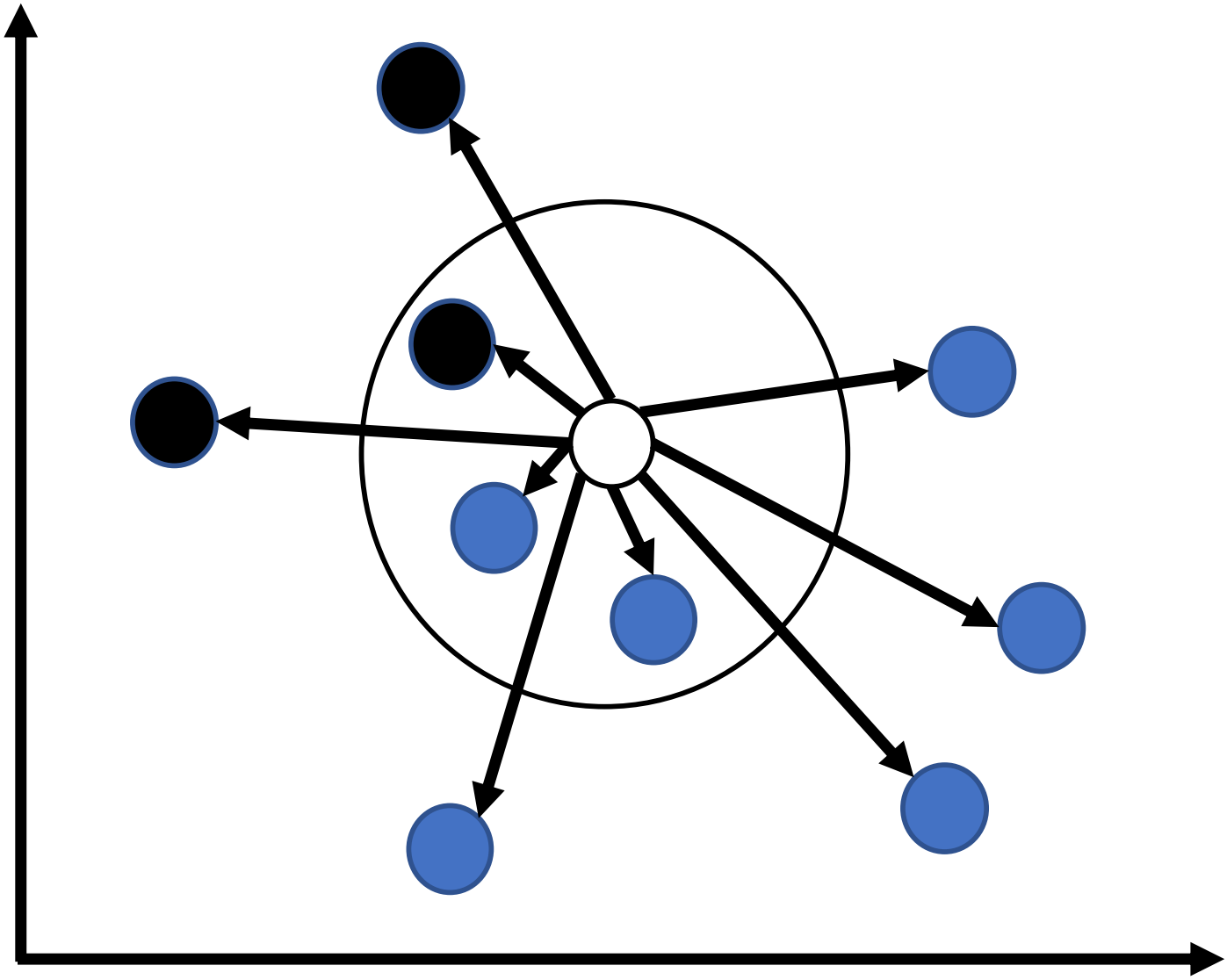
Tài liệu về quantum computing:

- Sách tiếng việt: chỉ cần đọc chương 1 - 5, bạn nào cần bản giấy có thể qua thư viện UIT hoặc thư viên trung tâm mua / mượn.

2. Giới thiệu

2.1. Thuật toán kNN cổ điển

kNN được xếp vào dạng thuật toán machine learning đơn giản nhất, với ý tưởng có thể tóm gọn như sau: "hùa theo số đông, bên cạnh mình như thế nào thì mình sẽ theo vậy".



Hình 1. Train set có những data point màu xanh và màu đen, Test set có data point chưa có màu, hỏi data point này nên có màu gì?

Input: chúng ta sẽ được cho trước dataset D, có chia thành hai tập nhỏ hơn là train set D_{train} và test set D_{test}. Trong đó giả định những data point d_i trong train set có đủ (data, label) còn test set thì data point d_j chỉ có (data).

Output: xác định các label của data point trong test set.

Thuật giải:

B1:

FOR d_j IN D_{test}:
___FOR d_i IN D_{train}:
_____$dist_{i,j} = distanceFunc(d_i, d_j)$.

Chú thích: $distanceFunc$ là hàm tính khoảng cách giữa hai data point bất kỳ. Tuy vào định nghĩa mà đây có thể là hàm Euclidean hoặc cosin.

B2:

FOR d_j IN D_{test}:
___$dist_j = [dist_{0,j}, dist_{1,j}, ..., dist_{n_{train},j}]$ với n_{train} là số lượng data point của D_{train}.
___Sắp xếp $dist_j$ theo thứ tự tăng dần và lấy k data point thuộc D_{train} gần với d_j nhất.
___Lấy ra k label tương ứng, chọn ra label có số lượng nhiều nhất (major vote).
___$d_j.label = majorVote(label_1, label_2, ..., label_k)$

Như hình 1, nếu $k = 3$ thì label sẽ là màu xanh.

B3: Đánh giá hiệu quả của kNN, bước này sẽ được hướng dẫn sau.

Nhận xét: chúng ta sẽ chú ý đến B1, thực thi hàm $distanceFunc$ là bước tốn nhiều tài nguyên nhất và toàn bộ B1 có độ phức tạp $\mathcal{O}(n_{train} * n_{test} * n)$ với n_{train}, n_{test} và n lần lượt là số lượng data point trong D_{train}, D_{test} và kích thước data point. Nếu thực thi $distanceFunc$ trên máy tính lượng tử, chúng ta sẽ giảm độ phức tạp xuống được $\mathcal{O}(n_{train} * n_{test} * \log n)$, cực kì có lợi trong trường hợp n lớn.

Câu hỏi:

- Chọn k như thế nào cho hợp lý?
- Chọn $distanceFunc$ như thế nào cho hợp lý?
- Điều kiện của n_{train} và n_{test} như thế nào thì thuật toán mới tiến hành được?

2.2. SWAP - test circuit.

Hiểu nôm na, đây là một thiết kế mạch có chức năng tính toán khoảng cách giữa hai data point. Phần này khá dài nên các bạn đọc trong paper này.

Input: d_i và d_j.
Output: là một số thực, khoảng cách giữa d_i và d_j.

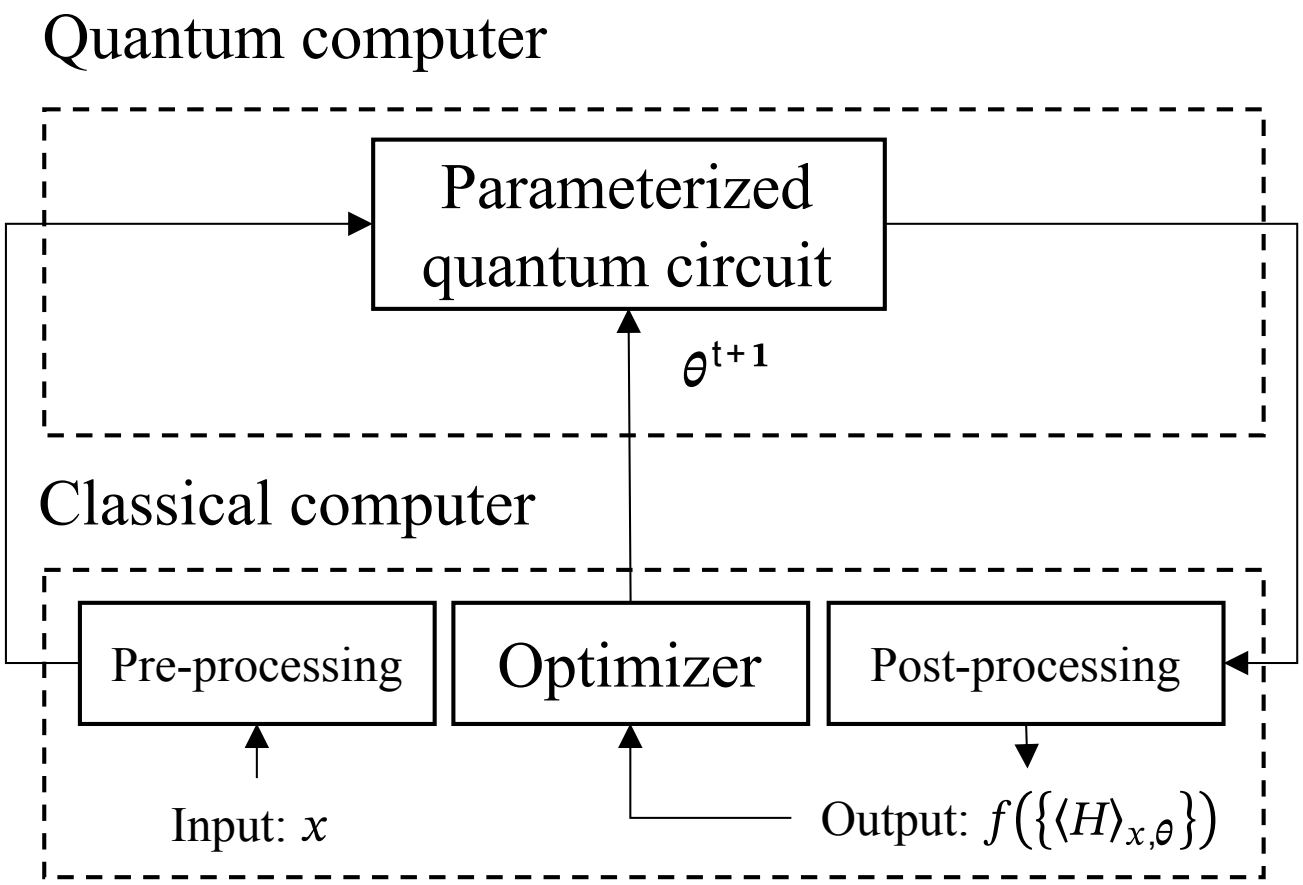
Câu hỏi:

- Làm sao để đưa d_i và d_j vào mạch lượng tử?
- SWAP-test hoạt động như thế nào (viết chuỗi phương trình biến đổi trạng thái).
- Làm sao để mạch lượng tử trả về kết quả (viết phương trình toán tử đo tác động lên trạng thái).

2.3. Quantum kNN

Nếu như các bạn đã hiểu hai phần 2.1 và 2.2 thì phần này đơn giản là chúng ta thay thế B1 trong phần 2.1 bằng mạch lượng tử trong phần 2.2.

Lúc này, nếu nhìn tổng quan, mô hình của chúng ta sẽ có kiến trúc như sau:



Hình 2. Mô hình hybrid quantum - classical.

Đây chính là một trong những mô hình phổ biến trong quantum machine learning [1], chúng ta sẽ tìm hiểu bám theo mô hình này trong khoảng thời gian về sau.

3. Thực hành

Yêu cầu trước khi tiến hành code:

- Cài dặt Python 3: nếu máy Windows các bạn cài Python bằng Microsoft Store, nếu máy Mac các bạn cài Python bằng các download từ trang chủ.
- Cài đặt các package: sklearn, qiskit, matplotlib thông qua pip.
- Sử dụng editor bất kì để code (khuyến nghị Visual Studio Code).

Code mẫu QKNN:

- Folder base: chứa các hàm cơ bản
- Folder jupyter: chưa các file notebook
  -- knn.ipynb: demo thuật toán kNN cổ điên.
  -- ist_circuit.ipynb: demo SWAP-test circuit.
  -- qknn.ipynb: demo thuật toán QKNN.

Ở trong code mẫu, chúng ta sẽ sử dụng tập dữ liệu Iris và thử nghiệm trên tập này. Trong quá trình code, các bạn nên code mọi thứ lại từ đầu.

Hướng dẫn quy trình thí nghiệm:

B1. Học cách sử dụng terminal để gọi thực thi file.

B2. Tổ chức code thành danh sách các hàm và lớp. Sử dụng jupyter notebook *.ipynb để kiểm thử tính chính xác của code mình viết nhanh hơn.

B3. Sau khi đảm bảo hàm / lớp chính xác, tổ chức chúng vào các file *.py để dễ debug.

B4. Nhớ lưu các kết quả thí nghiệm bằng hình ảnh (sử dụng matplotlib) hoặc file csv (sử dụng numpy / pandas).

B5. Commit code lên repository Github để lưu lại toàn bộ lịch sử.

Tham khảo

[1] M. Schuld and F. Petruccione, Machine learning with quantum computers (Springer, 2021).