Реализация алгоритма классификации точек на квантовом компьютере.

Якушев Г.А.[1]

[1]**СПб ГФМЛ №30, Санкт-Петербург, Россия**

mrdarkteslawhynot@gmail.com

Введение.

Квантовые компьютеры в последнее время становятся объектом пристального наблюдения как ученых, так и обычных людей. Не за горами тот день, когда они станут достаточно большими, чтобы выполнять нетривиальные задачи, и общедоступными настолько же, насколько сейчас распространены обычные системы.

Для того, чтобы продемонстрировать работу квантового компьютера мы выбрали задачу о классификации точек по двум множествам. Алгоритм для решения подобной задачи был описан в статье K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii [1], который и был взят за основу нашего решения.

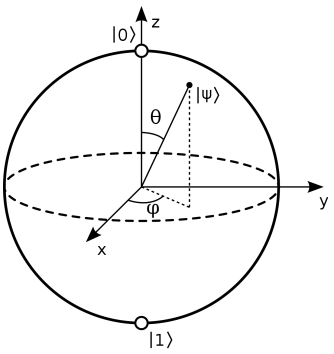

Рис 1. Сфера Блоха
Image by Smite-Meister /
CC BY-SA 3.0

Методы реализации алгоритма.

Алгоритм был реализован на Python 3 с использованием библиотеки qiskit [2]. Эта библиотека позволяет запускать написанные квантовые программы на реальных девайсах, а также симулировать из запуск компьютера на классических компьютерах.

Мы разбили точки на два класса с помощью функции $f(x)$: $[0,1]^2 \to \{0,\ 1\}$. Для обучения алгоритма мы взяли множество T :150 случайных точек из$[0,\ 1]^2$.

Наша функция классификации $\varphi(x,\ \theta)$: $[0,1]^2 \times R^n \to [0,1]$. Чем ближе число к 0, тем более уверена наша функция в том, что точка принадлежит 0 классу. Чем ближе к 1 - принадлежность к 1 классу. θ - n-мерный вектор, меняя который мы и можем обучать наш алгоритм. Такой подход очень близок к тому, что используется в классическом машинном обучении.

Для оценки точности работы нашего алгоритмы использовалась функция потерь, которая задается формулой: $L(\theta) = \sum_{t \in T} (f(t) - \varphi(t,\ \theta))^2$.

Для обучения алгоритма мы используем метод градиентного спуска.

$grad\ L_i = \sum_{t \in T} (\varphi(t,\ \theta) - f(t)) \cdot (\varphi(t,\ \theta + \delta_i) - \varphi(t,\ \theta - \delta_i))$, где δ_i- вектор, чья i-тая компонента - единственная ненулевая и равна $\frac{\pi}{2}$.

При многоразовом обучении цепи на одном множестве точек мы изменяем θ следующим образом:$\theta_{new} = \theta_{old} - \eta \cdot grad\ L$, где η - одномерный параметр, который определяет, насколько “большие” шаги мы делаем во время обучения.

Наш алгоритмы вычисления φ представляет из себя последовательное применение некоторых функций.

Для перехода от классических данных к кубитам мы преобразовали декартовы координаты данных точек в сферические на сфере Блоха. Где первая компонента x отвечает за поворот кубита вокруг оси OZ, а вторая компонента x отвечает за поворот вокруг оси OX.

В эксперименте мы использовали две разных вариации этой функции: первая поворачивала вокругOX на $[0,\ \pi]$,вокруг OZ на $[0,\ 2\pi]$, а вторая - вокругOX на $[\frac{\pi}{8},\ \frac{7\pi}{8}\ ]$,вокруг OZ на $[\frac{\pi}{8},\ \frac{15\pi}{8}]$

Далее используется Гамильтониан, который добавляет дополнительное взаимодействие между кубитами в системе.

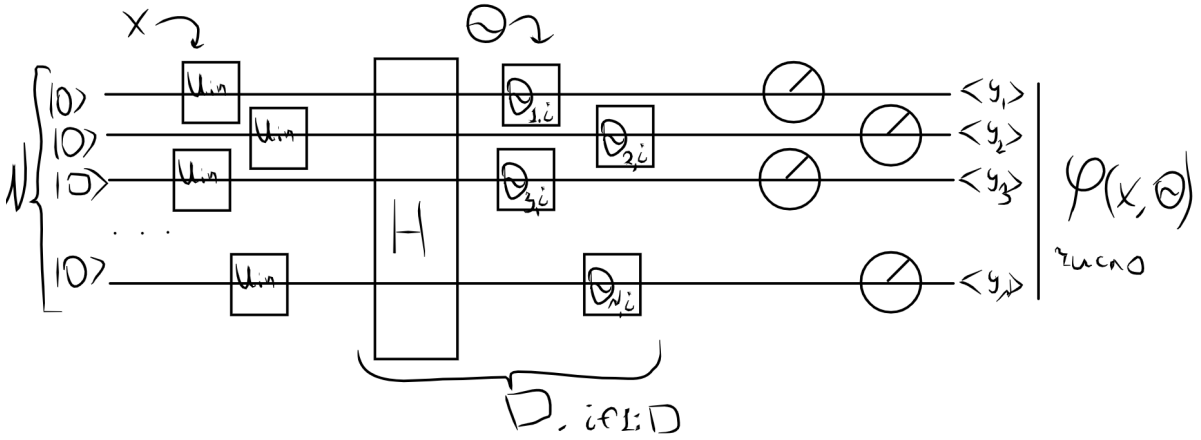

Рис 2

Схема квантовой функции

Затем будем обозначать числом N кол-во кубитов, D кол-во слоев, содержащее в себе N кубитов. Для того, чтобы значение нашей функции зависело от θмы используем поворот на произвольный угол, т е поочередные повороты вокруг осей $OX,\ OZ,\ OX$ на соответствующие θ. Эти повороты различны для каждого кубита каждого слоя, поэтому theta - вектор размерности $n\ =\ N\cdot D\ \cdot 3$.

После чего мы измеряем состояние кубита и возвращаемся к классическому формату данных, битам. Таких измерений проводится достаточное кол-во (1000), так как измерение состояния - вероятностный процесс.

Результаты.

Оптимизация и поиск θбыли проведены в симуляции, а обсчет точек был произведен на реальных квантовых компьютерах IBM[3].

В качестве параметров мы взяли: в массиве T 150 точек с координатами x_0, x_1, $\eta\ =\ 0.02$, при увеличении этого значения иногда невозможно попасть в подходящую θ из-за слишком большого шага, а при малом значении не хватает скорости, чтобы найти решение в адекватное время.

Мы провели несколько экспериментов с различными начальными условиями. В первую очередь мы классифицировали точки разделенные “пополам” (рис 3.1)

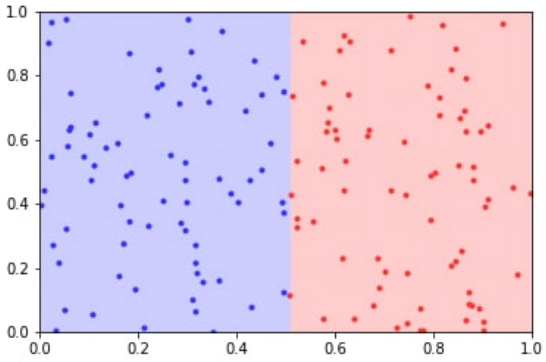

Рис 3.1
Исходная функция классификации

В эксперименте использовались $N = 2, D = 2$ - количество кубитов в слое и количество слоев соответственно. А также были испробованы два алгоритма задания начального состояния (перевод классических данных в заданные кубитам) описанные выше.Полученные результаты представлены на рисунках 3.2 (зависимостьL от итерации), 3.3 (итоговый результат).

Точки на краях картинки могут слипаться из-за того, что углы на сфере равны с точностью до 2π, а следовательно наш алгоритм плохо различает верх/низ и лево/право.

Для решения данной проблемы, во втором варианте задания начального состояния мы решили добавить разделение в повороте на начальный угол в размере $\frac{\pi}{8}$ с каждого края.

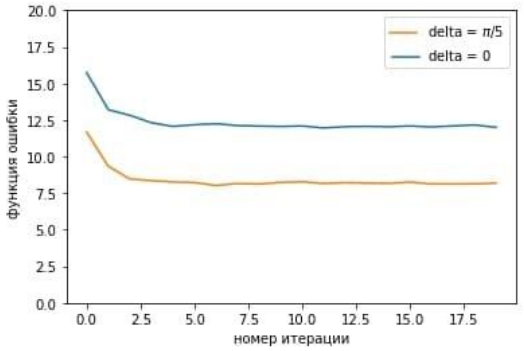

Рис 3.2
Функция ошибки от итерации

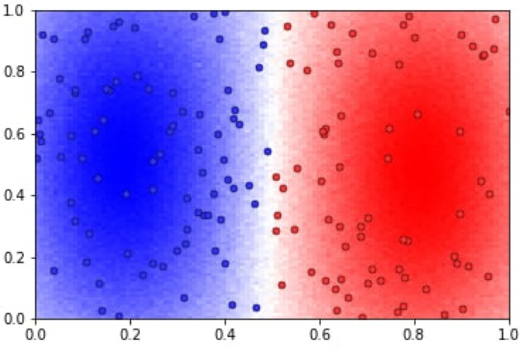

Рис 3.3
Классификация с разделением границ

Второй задачей была задача о классификации точек на круге. (уже разделенные края)

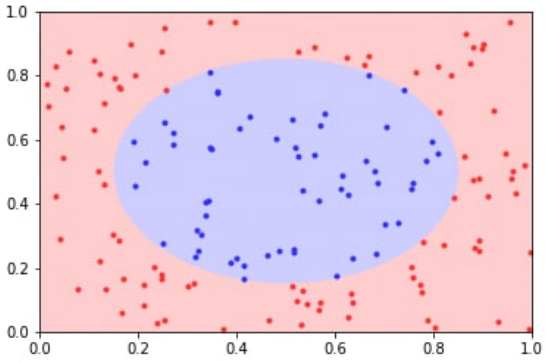

Рис 4.1
Исходная функция классификации

Изначально мы попробовали запустить эту задачу на такой же структуре: $N = 2, D = 2$. Результат виден на рис 4.2. Но этого явно не хватает для точного распределения точек по классам. Поэтому мы увеличили число кубитов и слоев до $N = 4, D = 5$. После чего получили удовлетворительный результат (рис 4.3).

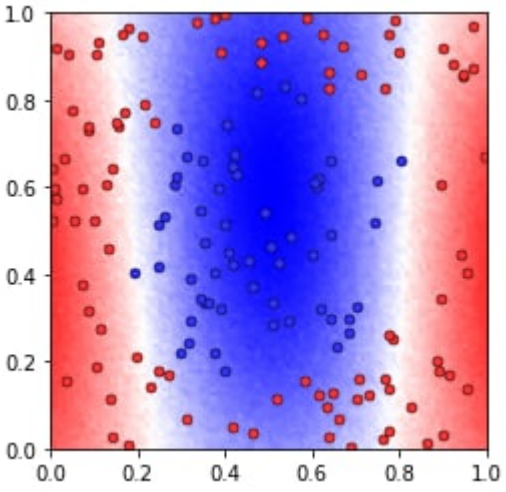

Рис 4.2
Классификация круга на малом числе кубит

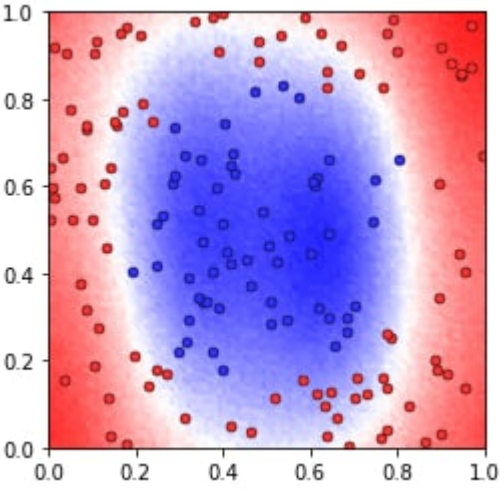

Рис 4.3
Классификация круга на большом числе кубит

По итогу мы реализовали алгоритм классификации точек по двум множествам для простейших случаев на квантовам компьютере и попробовали различные способу улучшения точности получаемого результата.

Ссылки на литературу и статьи.

[1]Mitarai K. et al. Quantum circuit learning //Physical Review A. – 2018. – Т. 98. – №. 3. – С. 032309.
[2]Aleksandrowicz G. et al. Qiskit: An open-source framework for quantum computing //Accessed on: Mar. – 2019. – Т. 16.
[3]IBM Quantum. (2021) (https://quantum-computing.ibm.com/) Дата обращения: 24.03.2021