Считывание состояния сверхпроводящего кубита с использованием машинного обучения

Владимир Терентьев (группа Б02-014), Иван Соломахин (группа Б02-002), Михаил Перельштейн (ментор)
Московский физико-технический институт
(Dated: 2 февраля 2021 г.)

Возможность считать с минимальной ошибкой состояние кубита имеет существенное значение для решения фундаментальных и прикладных задач при помощи квантовых компьютеров. Считывание кубитов IBM в среднем проводит с точностью 95.12%. В данной работе проводится сбор низкоуровневых данных с квантового компьютера, которые используются для улучшения считывания состояний кубитов при помощи нейронных сетей, а также алгоритмов градиентного бустинга.

I. Введение

Квантовые компьютеры уже сейчас могут выполнять задачи, непосильные для классических вычислительных устройств. Например, в работе [1] компания Google показала квантовое превосходство с помощью сверхпроводящего процессора. Однако при попытке управлять квантовым состоянием кубита непременно возникают различные ошибки. Один класс ошибок связан с устройством схемы сверхпроводящего кубита. Такие ошибки возникают как на этапе приготовления состояний, так и на этапе их измерения. Другой – с обработкой экспериментальных данных, которая сводится к задаче бинарной классификации. Для эффективной реализации основных алгоритмов на квантовом компьютере, таких как квантовое преобразование Фурье [2], алгоритм Шора [3], алгоритм Гровера [4], необходима высокая точность считывания состояния. В этой работе для обработки данных, считанных с кубита, используются различные алгоритмы машинного обучения: нейронные сети типа «многослойный персептрон» и методы градиентного бустинга из фреймворков CatBoost и xgboost, которые потенциально могут в значительной степени улучшить точность считывания.

II. Эксперимент

Для тренировки моделей используются данные, полученные с квантового компьютера IBM Armonk. Этот компьютер состоит из одного кубита. Он поддерживает взаимодействие с помощью микроволновых импульсов, параметры которых можно настраивать, а также сбор низкоуровневых данных. Для взаимодействия с компьютером использовался Python-фреймворк Qiskit Pulse. На момент наших экспериментов компьютер имел следующие характеристики: частота кубита – 4.972 GHz, время энергетической релаксации – 169.82 μs, время фазовой релаксации – 302.3 μs.

A. Считывание данных при помощи Qiskit

Для сбора первичного датасета используется Python-фреймворк Qiskit Pulse. Данный датасет, полученный при помощи считывания состояний, представляет собой два текстовых файла, в которых раздельно хранятся комплексные числа, соответствующие измеренным состояниям $|0\rangle$ и $|1\rangle$. Действительная и мнимые части этого числа - амплитуды синусоид, различающихся по фазе на $\frac{\pi}{2}$, на которые может быть разложена исходная волна, считанная с кубита. Результатом обработки этих данных является csv-файл, в котором хранятся вещественная и мнимая часть зарегистрированной волны, а также значение $|0\rangle$ или $|1\rangle$, которому соответствует данной состояние. Диаграммы для каждого из состояний, соответствующие этим данным, представлены на Рис. 1.

B. Физические основы воздействия на кубиты

По умолчанию кубит находится в состоянии $|0\rangle$, что соответствует уровню с наименьшей энергией. Для приготовления состояния $|1\rangle$ необходимо воздействовать на кубит несколькими короткими импульсами. Данные импульсы также называются π-импульсы, поскольку соответствуют повороту вектора состояния на сфере Блоха на 180°. Кубит имеет определенную резонансную частоту, поэтому необходимо подавать импульсы именно этой частоты. Для ее определения на кубит подается несколько импульсов разной частоты и определяется пик изменения амплитуды напряжения, получаемого при считывании состояния кубита. Данный пик соответствует искомой резонансной частоте.

Для поиска амплитуды π-импульса используется аналогичный метод: на кубит, находящийся в основном состоянии, последовательно подаются импульсы различных амплитуд до тех пор, пока он не перейдет в состояние с максимальной амплитудой (соответствует состоянию $|1\rangle$). Амплитуда импульса, при которой кубит перешел в состояние $|1\rangle$ и будет являться амплитудой π-импульса. На Рис. 2 показаны данные с кубита, по которым восстановлены параметры.

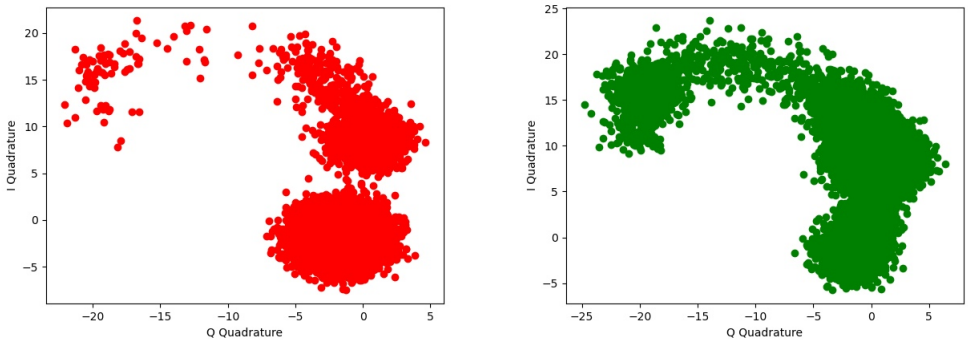

Рис. 1. IQ-диаграммы для состояний $|0\rangle$ и $|1\rangle$ соответсвенно

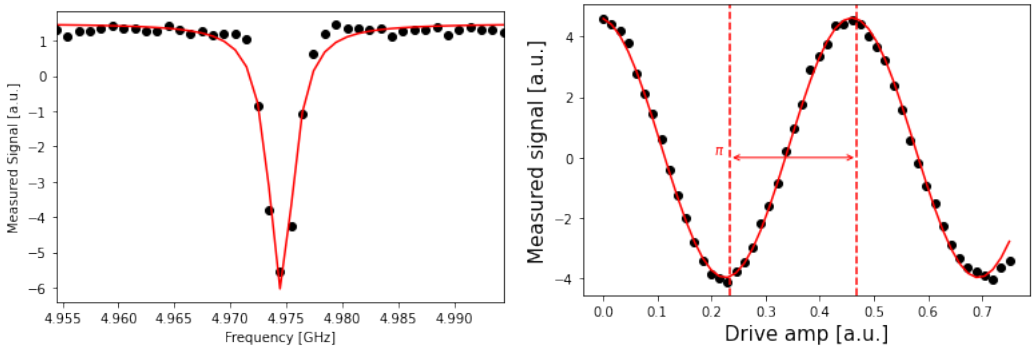

Рис. 2. Поиск резонансной частоты кубита и амплитуды π-импульса

III. Анализ экспериментальных данных

A. Нейронная сеть

Одним из возможных методов достижения высокой точности считывания состояний кубитов является использование нейронной сети для интерпретации полученного значения. В данной работе использовались нейронные сети типа «многослойный персептрон». Для удобства классификации состояний каждое комплексное число, переданное на вход, представлялось в виде действительной и мнимой части, а затем подавалось на 2 нейрона в 1 слое. Исходные данные были разбиты на два наборы: тренировочный, состоящий из 90% данных и тестовый из 10% данных. Для классификации состояний кубитов использовались нейронные сети различной глубины и с различным количеством нейронов в скрытых слоях. Были использованы сети со следующими параметрами слоев 2-10-1, 2-20-1, 2-8-6-1, 2-10-8-6-1, 2-20-8-6-1. Здесь через запятую перечислены количества нейронов в нейросети. Например, на Рис. 3 можно видеть нейросеть, имеющую архитектуру 2-4-4-1, аналогичную той, что было использована в данной работе. Последний слой выдавал два результата – «0» или «1», соответствующие состояниям измеряемых кубитов. Для

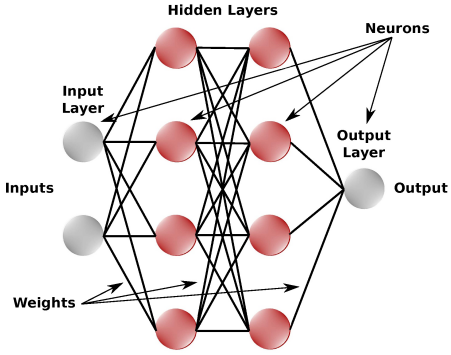

Рис. 3. Схема нейронной "сети типа многослойный персептрон¨

этого в последнем слое использовалась функция активации $\sigma(Z)$ (Sigmoid). Для внутренних слоев использовалась функция активации $ReLU(Z)$ (Rectified Linear Unit)

$$\sigma(Z) = \frac{1}{1 + e^{-Z}} \tag{1}$$

$$ReLU(Z) = \max(0, Z) \tag{2}$$

$$Z = W \cdot X + b \tag{3}$$

W – весовой коэффициент, X – входные данные (для 1 слоя-тренировочные данные, для последующих-выходные значения предыдущего слоя), b – коэффициент смещения

В следующей таблице приведены значения Accuracy (отношение верных предсказаний к общему количеству предсказаний), достигнутые при помощи описанных выше нейронных сетей.

Используемые слои	Тренировочный набор, %	Тестовый набор, %
2-10-1	93.89805	93.94763
2-20-1	93.86100	93.93528
2-8-6-1	93.89119	93.95998
2-10-8-6-1	93.90354	93.95998
2-20-8-6-1	93.89256	93.95998

Таблица 1. Эффективность конфигураций нейронных сетей

Более глубокие нейронные сети справляются с задачей лучше, чем небольшие сети, однако обучение достаточно крупной сети сталкивается с проблемой исчезающего градиента, что приводит к крайне медленному обучению.

B. Градиентный бустинг

Градиентный бустинг является альтернативным способом решения задачи бинарной классификации. Современные алгоритмы позволяют превзойти нейронные сети в решении данного класса задач.

Градиентный бустинг является сильным алгоритмом машинного обучения. Суть метода заключается в построении ансамбля слабых моделей - деревьев принятия решений, в которых модели строятся последовательно, то есть следующее дерево учится на ошибках предыдущего, затем этот процесс повторяется, наращивая количество слабых моделей. Таким образом получается сильная модель, имеющая большую эффективность.

В настоящей работе использовались следующие модели градиентного бустинга: CatBoostClassifier и xgboost.XGBClassifier.

Фреймворк	Тренировочный набор, %	Тестовый набор, %
CatBoostClassifier	94.00373	93.79940
xgboost.XGBClassifier	95.80027	91.42786

Таблица 2. Используемые алгоритмы градиентного бустинга и их Accuracy

Таким образом, после предварительной тренировки модель достигла уровня, близкого к эффективности нейросети. В результате дальнейших тренировок есть возможность достичь лучших результатов.

IV. Обсуждение результатов и дальнейшие планы исследований

Благодаря использованию нейронной сети и градиентного бустинга удалось достичь результатов близких к значениям, полученным IBM. С помощью нейросети на тренировочном наборе, состоящем из 90% данных, была достигнута точность 93.90354%, на тестовом наборе из 10% данных – 93.95998%. Использование градиентного бустинга дало близкий к данному результат: на тренировочном наборе – 94.00373%, на тестовом – 93.79941%. Данные результаты являются предварительными и, скорее всего, будут улучшены в следующей работе. Применение градиентного бустинга в данной работе кажется более эффективным, чем использование нейросети, поэтому в следующей работе планируется сделать акцент на градиентном бустинге. Кроме того, для выбора лучшей модели мы собираемся сравнить эффективность различных фреймворков, в которых реализован алгоритм бинарной классификации. Также планируется продолжить тренировать нейросети для достижения лучших результатов. Применение градиентного бустинга в данной работе кажется более эффективным, чем использование нейросети, поэтому в следующей работе планируется сделать акцент на использовании градиентного бустинга. Кроме того, мы собираемся сравнить эффективность различных фреймворков, в которых реализован данных алгоритм. Достижению более высокой точности мешает не только несовершенство обработки информации, полученной от квантового компьютера, но и ошибки, связанные с приготовлением состояний. Однако даже при неидеальном приготовлении состояний возможно достижение более высокой точности с помощью совершенствования алгоритмов обработки данных. Для более быстрых тренировок нейронных сетей можно использовать графические процессоры, позволяющие проводить параллельные вычисления. По этой причине планируется переписать код, написанный с использованием numpy, на tensorflow или на аналогичном фреймворке, поддерживающем обучение на графическом процессоре. Для улучшения работы градиентного бустинга можно применить стекинг, то есть использование нескольких алгоритмов обучения (ансамбля алгоритмов) вместе для достижения большей производительности. Это возможно благодаря тому, что решение о классификации каждого обучающего и тестового экземпляра принимается на основе решений нескольких алгоритмов методом «голосования» (для выбора между $|0>$ и $|1>$ алгоритмы «голосуют», и состояние, имеющее больше половины голосов, выбирается как результат работы ансамбля алгоритмов). Таким образом, ансамбль алгоритмов допускает ошибку только если ошибается больше половины алгоритмов. Также мы планируем опробовать созданные нами модели на данных с других процессоров IBM, когда они получат поддержку Qiskit Pulse.

V. Заключение

В данной работе были исследованы различные методы улучшения точности считывания состояний кубитов. Было выяснено, что нейронные сети и градиентный бустинг позволяют достичь достаточно высокой производительности. По этой причине мы полагаем, что использование подобных алгоритмов машинного обучения позволит достаточно хорошо исправлять ошибки квантовых компьютеров, что позволит реализовать на них алгоритмы, требующие высокой точности считывания.

VI. Приложение

Для более детального изучения проделанной нами работы мы прилагаем к статье архив, содержащий все использованные программы. Кроме того, программы можно найти на нашем GitHub: `https://github.com/TerentevVSs/Qubits-readout-result-v-1/`. Ниже приводится инструкция по использованию кода.

A. Нейронные сети

В папках, названия которых состоят из чисел и знака «-» (например «2-10-8-6-1») хранятся данные о использованных нейронных сетях. Числа в данных названиях обозначают количество нейронов в слоях. Файлы вида "parameter parameter_name.txt"(например «parameter W1.txt» «parameter b1.txt») содержат определенные параметры нейросети. Цифра в названии параметра относится к слою, в котором используются из этого файла. «W» обозначает, что параметр является весовым коэффициентом, а «b» обозначает параметр смещения. Файл "Training examples.csv"содержит данные для тренировок и тестов. Первый столбец-вещественная часть полученной волны, второй-комплексная часть, третий-исходное значение (0 или 1).
Parameters = L_layer_model(train_x, train_y, layers_dims, num_iterations=0, learning_rate=0.5, print_cost=True, lambd=0, rate="continue"). Здесь
«train_x» — тренировочные входные данные,
«train_y» — правильные метки для входных данных,
«layers_dims» – архитектура нейросети вида [2, 10, 8, 6, 1],
«num_iterations» – количество итераций алгоритма,
«learning_rate» – шаг обучения,
«print_cost» – если «True», показывает значение функции потерь во время обучения,
«lambd» – параметр для регуляризации,
«rate» – если «True», продолжает обучение сохраненной модели, иначе начинает обучать заново) Для запуска модели можно поставить num_iterations=0 и запустить программу. При этом обучение не будет продолжено и сразу будет выведен результат работы данной модели.

B. Градиентный бустинг

В папке Gradient Boosting содержатся файлы моделей градиентного бустинга. Результат каждой из моделей при некоторых конфигурациях можно увидеть в конце программы в виде комментариев, а также при запуске соответствующей архитектуры модели. В данной работе использовалось 2 модели: catboost.CatBoostClassifier и xgboost.XGBClassifier.

1. *catboost.CatBoostClassifier*

Чтобы запусить определенную модель нужно заменить 36 строку: cat = CatBoostClassifier(iterations=40000, task_type='GPU', devices='0-3', depth=4, learning_rate=0.1, loss_function = 'CrossEntropy') на соответствующую строку из журнала в конце программы. Здесь
«interations» – количество итераций обучения,
«task_type='GPU'» – обучение на графическом процессоре,
«devices='0-3'» – обучение на нескольких графических процессорах с индексами 0-3,
«depth» – глубина решающего дерева,
«learning_rate» – шаг обучения алгоритма,
«loss_function» – функция потерь модели.

2. *xgboost.XGBClassifier*

Эта модель на данный момент не имеет функции сохранения параметров и поэтому запуск возможен только с начального этапа обучения. В следующей работе планируется улучшить интерфейс работы с данной моделью.

[1] Arute, F., Arya, K., Babbush, R. et al. Quantum supremacy using a programmable superconducting processor. Nature 574, 505–510 (2019). https://doi.org/10.1038/s41586-019-1666-5

[2] Hales, L.; Hallgren, S. (November 12–14, 2000). "An improved quantum Fourier transform algorithm and applications". Proceedings 41st Annual Symposium on Foundations of Computer Science: 515–525. CiteSeerX

10.1.1.29.4161. doi:10.1109/SFCS.2000.892139. ISBN 0-7695-0850-2. S2CID 424297.

[3] Shor, P.W. (1994). "Algorithms for quantum computation: discrete logarithms and factoring". Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Comput. Soc. Press: 124–134. doi:10.1109/sfcs.1994.365700. ISBN 0818665807.

[4] Grover L.K.: A fast quantum mechanical algorithm for database search, Proceedings, 28th Annual ACM Symposium on the Theory of Computing, (May 1996) p. 212