Национальный исследовательский ядерный университет
МИФИ УНЦ Квантовый Инжиниринг

Решение задачи 3
на II Всероссийском квантовом хакатоне

Выполнено командой «Квинжэссенция» в составе:

Горшков А.
Ершов Г.
Орехова К.
Пылаева Н.
Субботин Г.

Москва 2024

Постановка задачи

Методами квантового машинного обучения решить задачу классификации данных на основании содержания и эмоциональной окраски текста. На основании отзывов составить оценку мнения пользователей о различных аспектах продукта.

Векторизация данных

Для решения поставленной задачи в первую очередь необходимо представить данные в “понятном” для вычислительной машины виде. Для этого можно использовать векторизацию данных, например с помощью алгоритма TF-iDF. Кратко опишем принцип его работы:

TF-IDF (Term Frequency-Inverse Document Frequency) — это метод статистической оценки значимости слов в тексте относительно документа и коллекции документов. Целью является выделение важных терминов для анализа текста, отфильтровывая "шумовые" слова.

TF (частота термина) : измеряет, насколько часто слово встречается в документе.

IDF (обратная частота документа) : оценивает редкость слова в наборе документов, уменьшая вес часто встречающихся слов (например, "и", "в").

Значимость слов определяется как:

$$TF\_IDF(t,d) = TF(t,\ d) * IDF(t)$$

$$IDF(t) = \log\left(\frac{N}{1+m_t}\right) \qquad TF(t,d) = \frac{n_j}{\sum_k n_k}$$

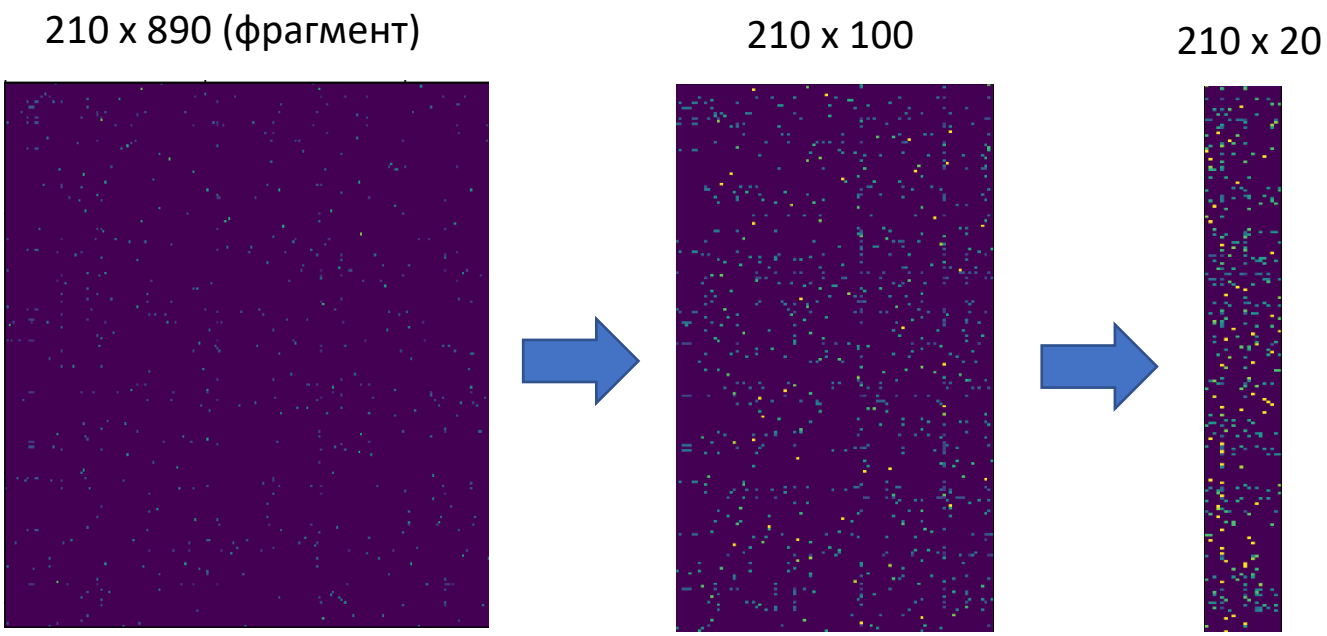
N — общее число документов, m_t — число документов, содержащих слово t, n_j – число вхождений слова t в документ, $\sum_k n_k$ – общее число слов в данном документе.

Составление вектора V: компонента $V_i = \begin{cases} TF\_IDF_i\ , & \text{если } i-\text{ое слово есть в отзыве} \\ 0\ , & \text{если такого слова нет в отзыве} \end{cases}$

Размер вектора-отзыва V: $Len(V) = 1 \times M$, где M – число уникальных слов во всех отзывах

Однако полученные таким образом вектора имеют большой размер, значительно превышающий возможно вычислимый. Мы можем уменьшить его двумя путями:

Первый: просуммировать по вхождениям конкретного слова в предложение и по всем вхождениям этого слова во все предложения. При этом мы получим массив меньшего размера, состоящий из слов с большей значимостью. Однако при этом мы потеряем много данных, что критично скажется на точности. Рассмотрим, как меняется цветовая карта векторизованных данных при таком сжатии:

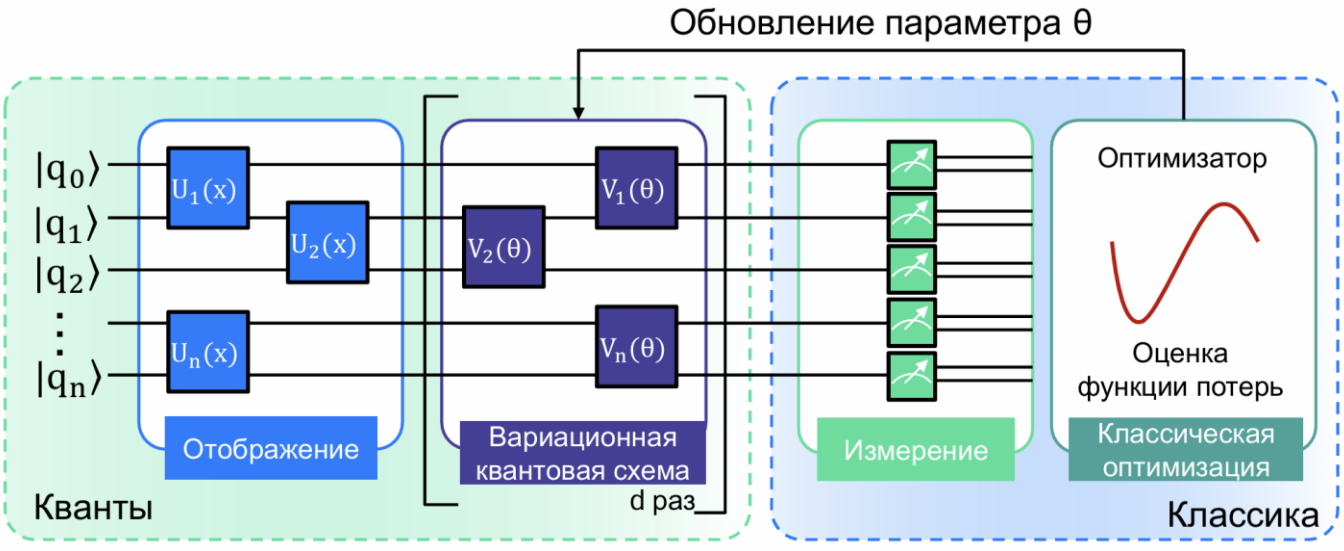


При таком размере вектора признаков уже можно запускать QVML.
Однако точность такого подхода оставляет желать лучшего и приближается к случайному выбору. При этом значительны затраты по времени обучения.

Второй способ: это слой классической нейронной сети. При этом точность решения, судя по результатам промежуточных тестов, получается большая, однако для того, чтобы это уверенно заявлять, нужно собрать больше данных.

Квантовое вариационное машинное обучение

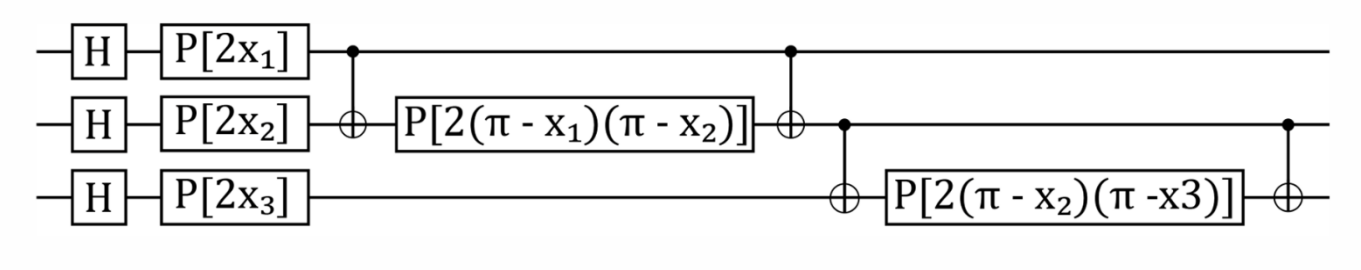
Для реализации МО мы использовали следующую схему: (arXiv:1804.11326v2)



Для отображения данных мы применяли унитарные операторы такого вида:

$$\mathcal{U}_\Phi(\vec{x}) = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n} \qquad U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} Z_i \right)$$

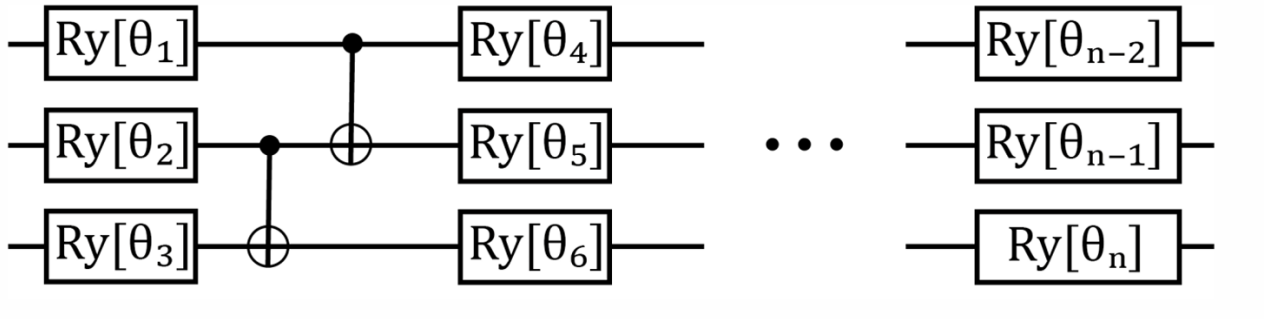
Или, что то же самое, в виде гейтов на нотном стане:



Оператор P[x] – это оператор Rz(φ), т. е. мы кодируем данные таким образом, чтобы потом измерять результат в вычислительном базисе.

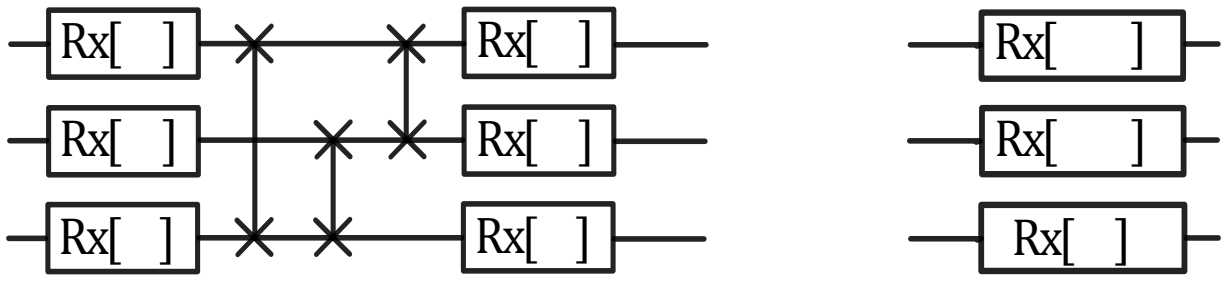
Вариационная схема может быть представлена различными анзацами:

$$U(\vec{\theta}) = U_n(\theta_n) \dots U_2(\theta_2) U_1(\theta_1)$$



Здесь можно заменить Ry на Rx, чтобы использовать вращение по всей сфере блоха. Т. е. в нашей цепи мы использовали как анзац с Ry, так и с Rx. Но при этом особого выигрыша по точности или качеству сходимости, при добавлении дополнительной оси поворота, мы не заметили.

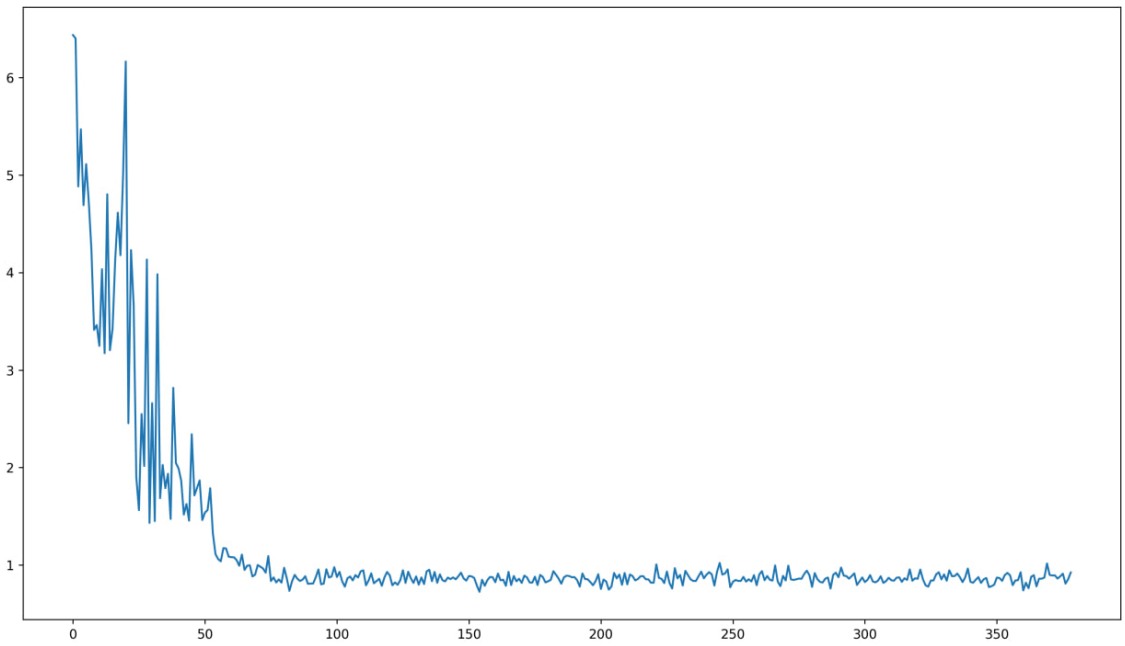
Так же мы пробовали реализовать анзац с SWAP вместо CNOT. И он также оказался работоспособным.



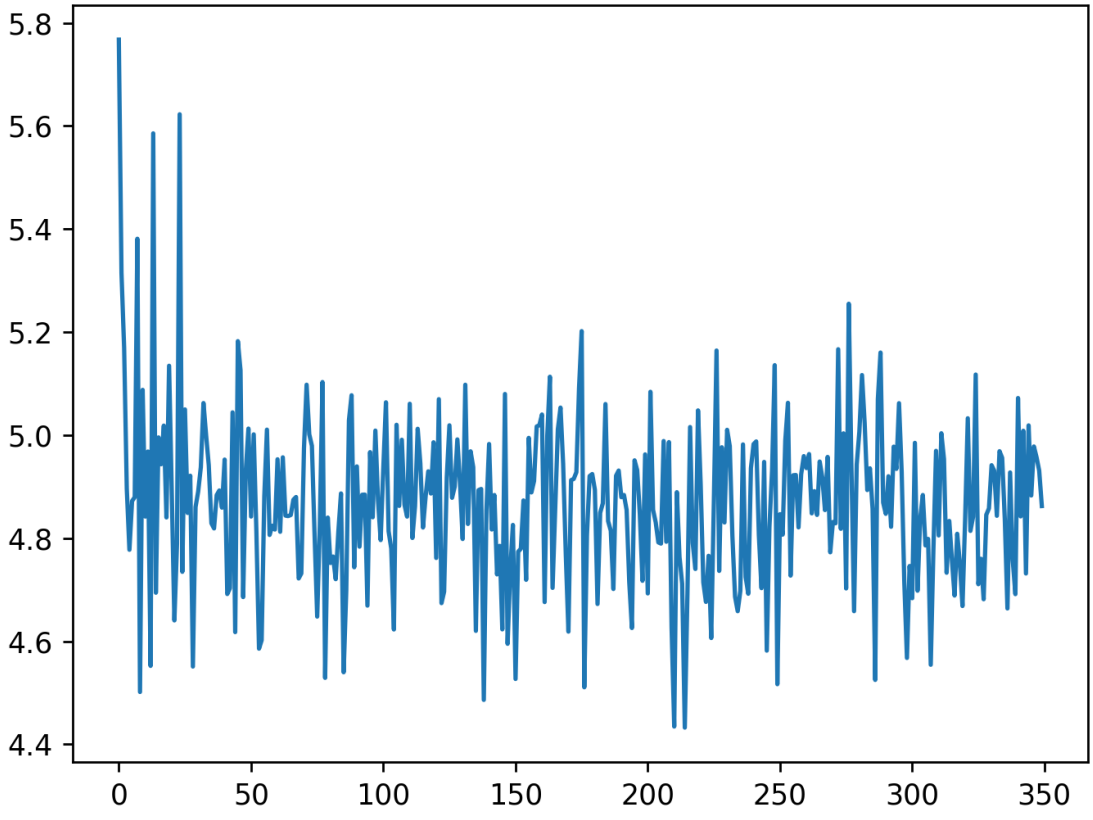
По сходимости обучения нужно отметить важность выбора начальных значений параметров. Так как, вероятнее всего, функция ошибки почти везде плоская и лишь в некотором месте у неё можно корректно вычислить градиент. Для оптимизации использовался SciPy minimize метод COBYLA.

В зависимости от начальных значений параметров, при обучении на одних и тех же данных сходимость могла выглядеть различно.

Случай попадания в область «спуска» функции ошибки:



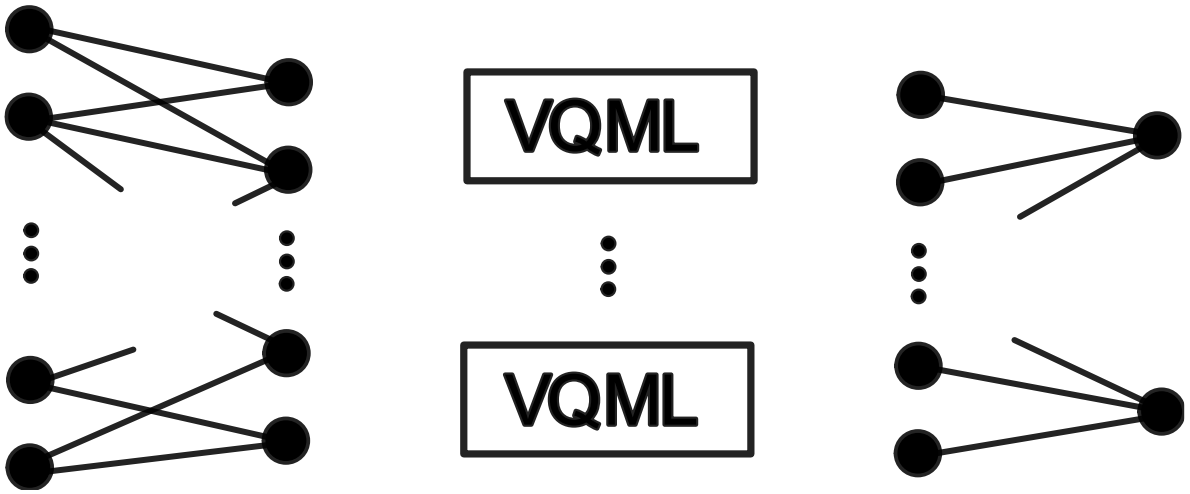
Случай блуждания по плоскости:



Проблематика решения

Для достижения приемлемой точности необходимо обучать нейросеть на большом количестве входных параметров, однако для квантовой реализации в настоящий момент это критично. Ситуацию не спасает даже уменьшения размерности входного вектора, с использованием слоя классической нейронной сети, ибо при таком сильном уменьшении (до порядка 20 параметров, далее вычисления становятся трудными), сильно теряется точность.

Как возможное решение этой проблемы, мы можем предложить использование гибридной нейронной сети с разделенным квантовым слоем:



В таком случае мы сможем уменьшить степень сжатия данных и при этом не затрачивать столько времени на эмуляцию квантовых цепей большой размерности. Как проблему такого подхода, можно выделить потерю связи между вершинами первого слоя при переходе к квантовому, так как отдельные вариационные цепи не взаимодействуют между собой. Но и эту проблему можно решить, введя запутывание между отдельными QVML с помощью телепортации состояний между ядрами квантового вычислителя. (arXiv:2408.01424v1)