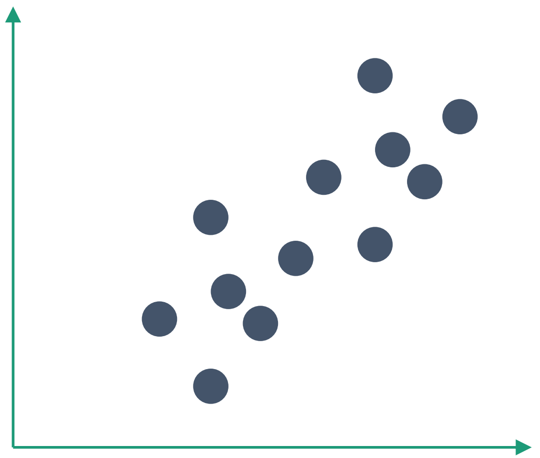


Coreset Construction for Quantum Data Science

Dr. Muhammad Usman

The University of Melbourne

Real-world Data

MCDS Quantum Data Science Workshop, December 7-10, 2021

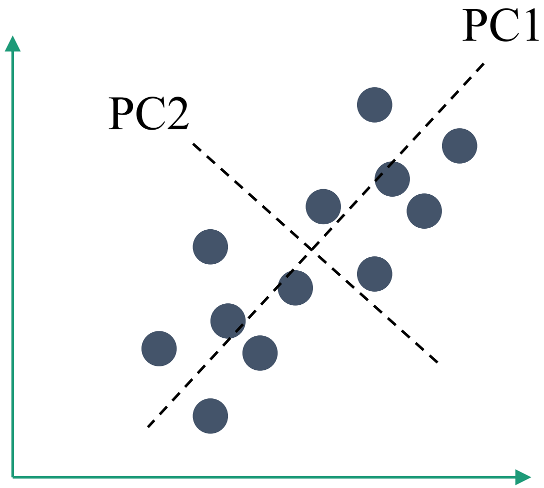
Loading of Classical Data in a Quantum Computer

Typically, data science involve large data sets

How do we define a large data set?

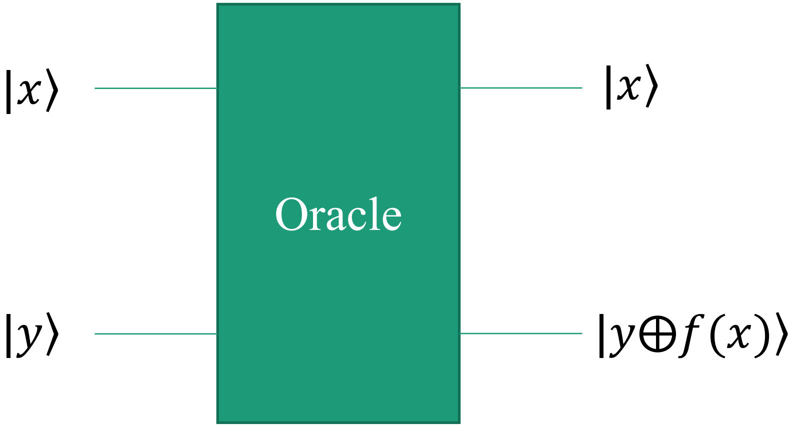
One way of thinking:

Processing becomes very hard for algorithms with cost function O(n^2), where n is the size of data

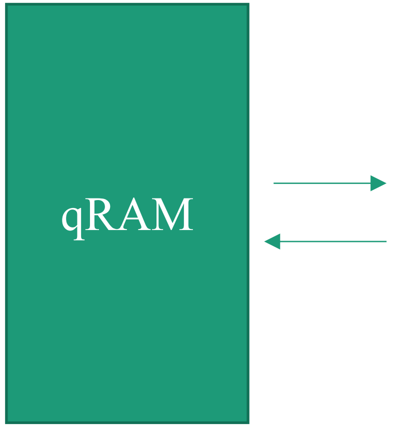


Big Question: How do we load a large data set into a quantum computer without losing potential quantum advantage?

Example: Grover's

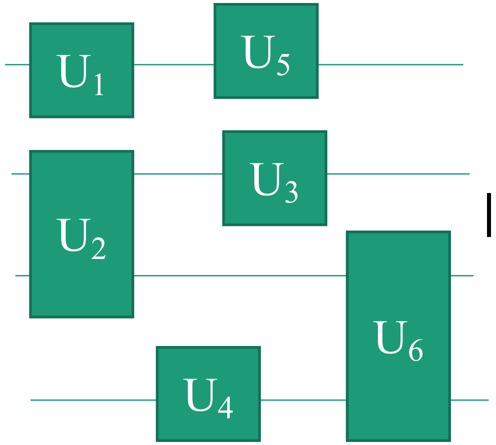


Example: HHL



$$|x\rangle = \sum_{i=1}^{N} x_i |i\rangle$$

Example: QML



$$|\psi\rangle = \sum_{i=1}^{N} \sqrt{p_i} |x_i\rangle$$

Possible Alternatives – Classical Pre-processing of Data

In NISQ era, where capabilities of quantum devices are limited i.e., a few qubits, noise, connectivity, etc.

One possible answer is: data size reduction by classical pre-processing.

Coreset construction – useful for machine learning such as k-means clustering, Bayesian inference, Saddle-point approximation.

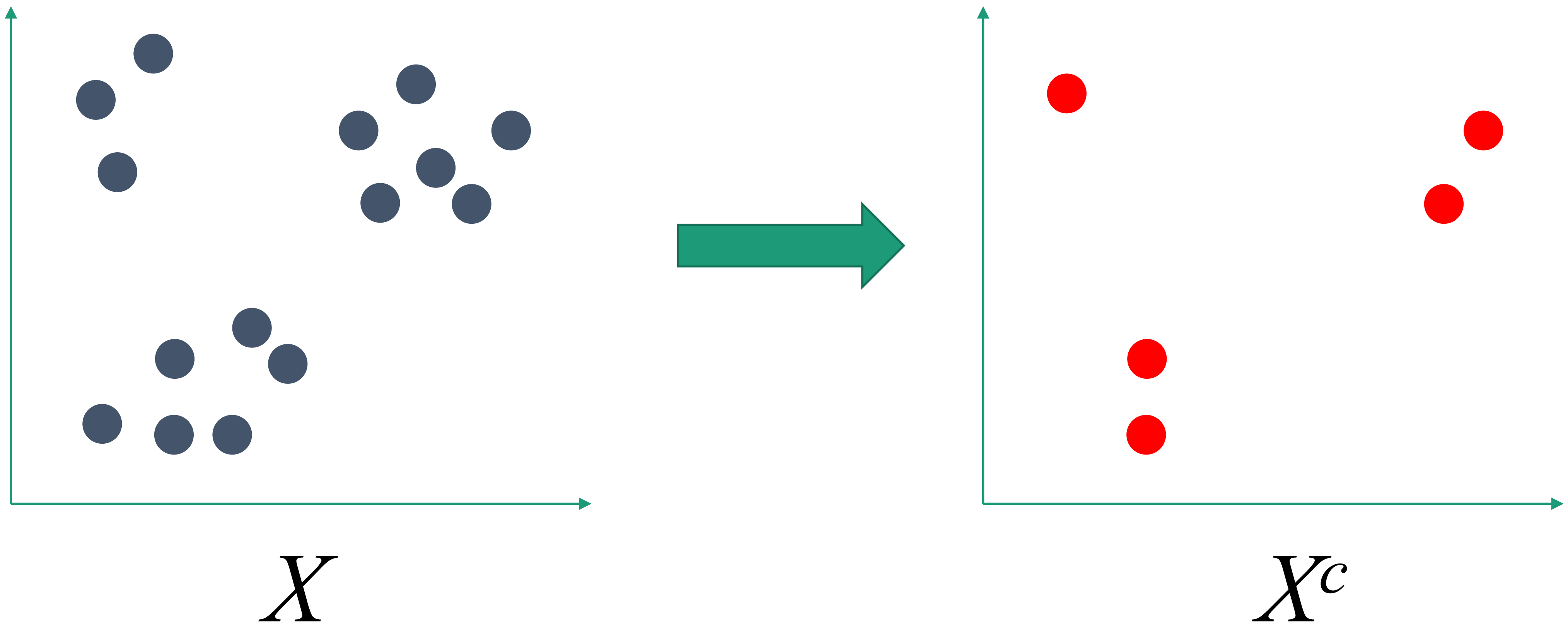
Image reduction by subpooling – Quantum convolutional neural network

A second possible answer is: Hybrid classical/quantum approaches such as quantum-classical convolutional neural network.

Can the classical data science community help?

Coreset Construction for Data Reduction

Main Idea: X -> (X^c, w), where the weighted data set X^c sufficiently summarizes the original data

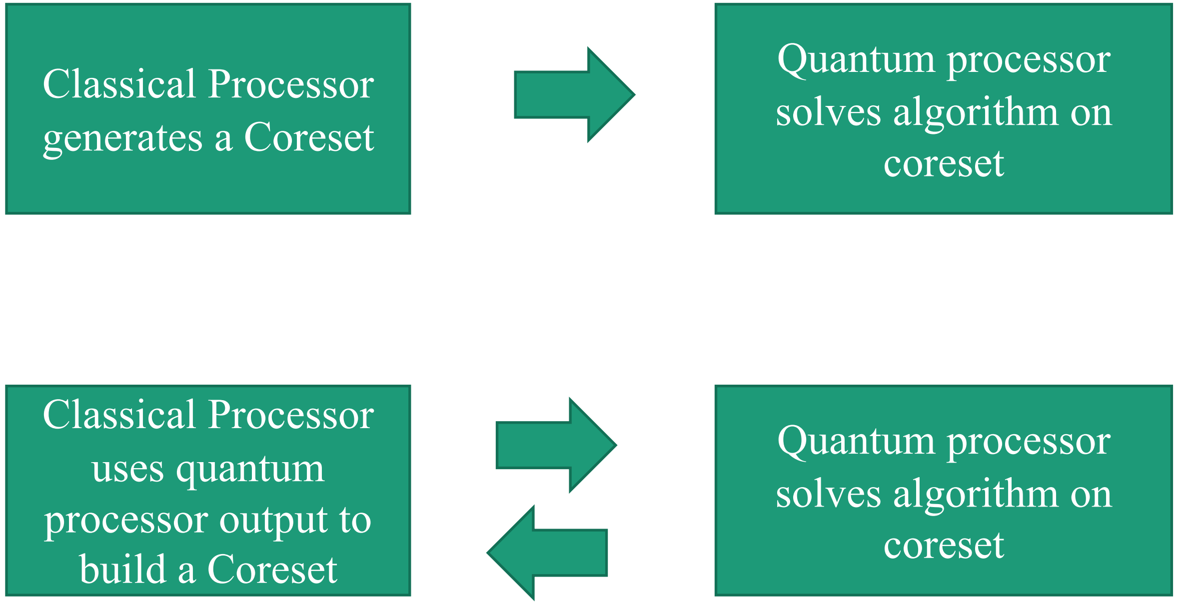


Here X^c is a coreset of X if $\sum_{x \in X} f(x) \approx \sum_{x \in X^c} f(x)$, where $f(x)$ is the function to be evaluated on dataset.

Coreset Construction for Data Reduction

Here X^c is a coreset of X if $\sum_{x \in X} f(x) \approx \sum_{x \in X^c} f(x)$, where *f(x)* is the function to be evaluated on dataset.

Question: what is a good coreset?



A good coreset is an ϵ-coreset such that

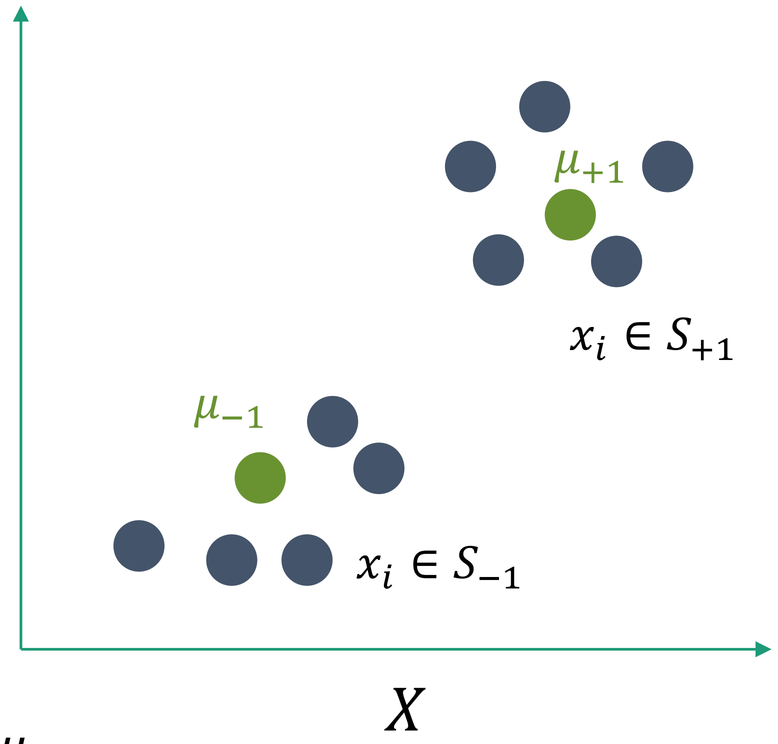
$|Cost(f(X)) - Cost(f(X^c,w))| \leq \epsilon |Cost(f(X))|$

Types of Coresets:

- Non-adaptive coresets, applications for k-means clustering, max-cut, maximum likelihood estimation
- Adaptive coresets, applications are Bayesian inference, Saddle-point optimisation

2-means Clustering problem

To demonstrate the working of the coreset technique, we will look at a very simple problem – 2-means clustering



Problem Statement:

Input : A data set $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$
Output : Cluster centers $\boldsymbol{\mu}_{-1}, \dots, \boldsymbol{\mu}_{+1}$ which approximately minimize

$$\sum_{i \in [n]} \min_{j \in \{\boldsymbol{\mu}_{-1}, \boldsymbol{\mu}_{+1}\}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_j\|^2$$

For 2-means clustering problem, we will have only two cluster centers μ_{+1} and μ_{-1}

The problem can be redefined as partitioning the data set into two subsets S_{-1} and S_{+1}, such that it minimizes the squared-distance to the closest cluster centers:

$$\sum_{i \in S_{-1}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_{-1}\|^2 + \sum_{i \in S_{+1}} \|\boldsymbol{x}_i - \boldsymbol{\mu}_{+1}\|^2$$

Selection of Coreset Data Points

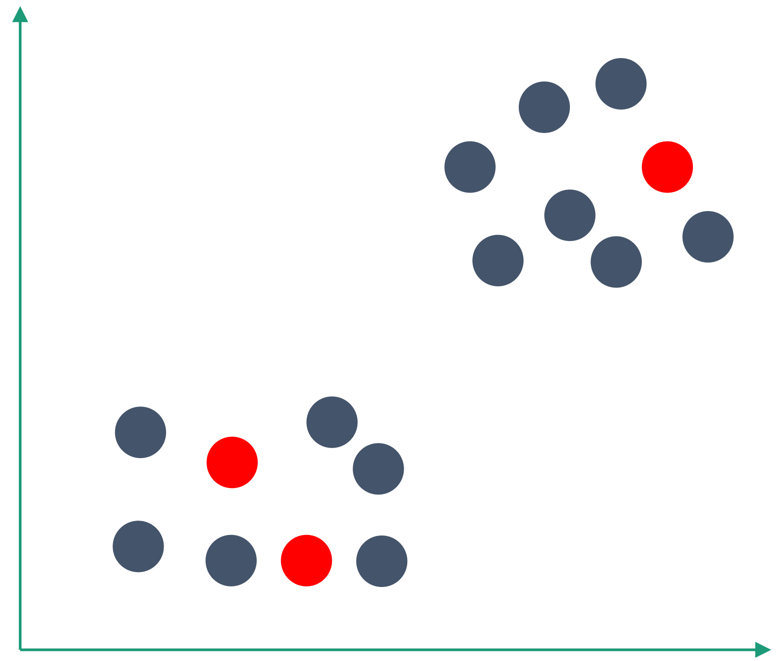
Generally, the size of coreset is a variable and it is optimized to make a best selection

However, due to limited number of qubits, we can keep the coreset size fixed, where each data point in coreset is represented by a qubit

For example: Coreset size = 3 can be implemented using 3 qubits

Methods to construct coresets:

- Random sampling
- Braverman method (arXiv:1612.00889, 2016)



X = Full Data w, X^c = Coreset(X)

For 2-means clustering problem:

$$(\mu_{-1}, \mu_{+1}) = Cluster(X)$$

$$(\mu_{-1}, \mu_{+1}) = Cluster(X^c, w)$$

$$Cost = \sum_{x \in X \text{ or } x \in X^c} \min(|x - \mu_{-1}|^2, |x - \mu_{+1}|^2)$$

Construction of Coresets

How do we compute X^c ?

Case of k-Clustering

Algorithm 1 D^p-SAMPLING

Require: data set $\mathcal{X}$, k, p

Uniformly sample $x \in \mathcal{X}$ and set $B = \{x\}$.
for $i \leftarrow 2, 3, \ldots, k$ **do**
  Sample $x \in \mathcal{X}$ with probability $\frac{\mathrm{d}(x,B)^p}{\sum_{x' \in \mathcal{X}} \mathrm{d}(x',B)^p}$ and add it to B.
return B

Step 1: Calculation of Centroids using K-Means++

Step 2: Calculation of Coreset Points

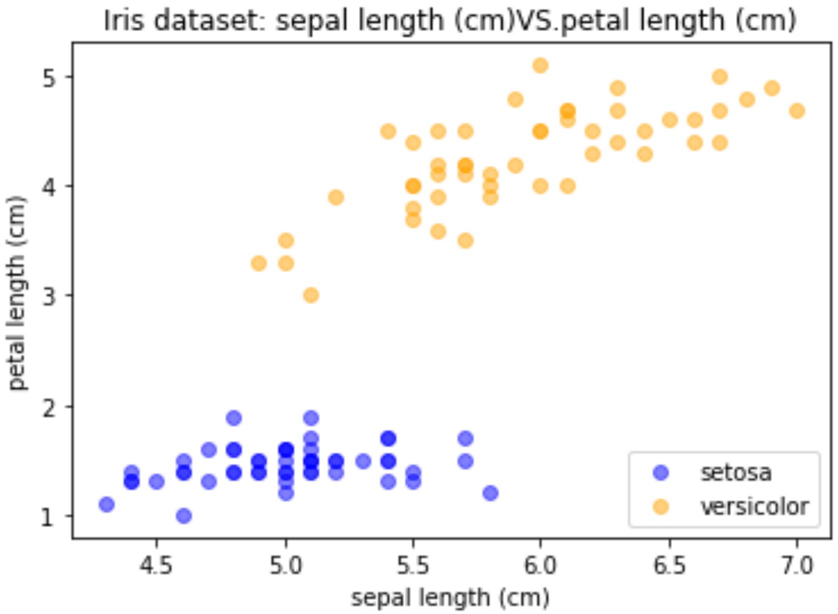
Algorithm 2: CORESET$(P, w, \mathcal{B}, m)$

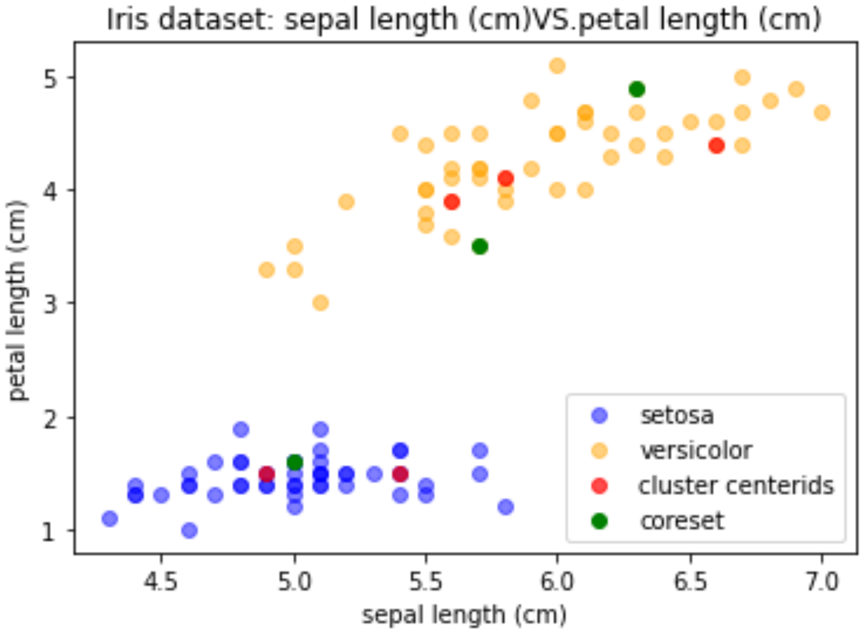
Input: A weighted set (P, w) where $P \subseteq X$ and (D, X) is a ρ-metric space, (α, β)-approximation $\mathcal{B} : P \to B$, and sample size $m \geq 1$.

Output: A pair (S, u) that satisfies Theorem 6.6.

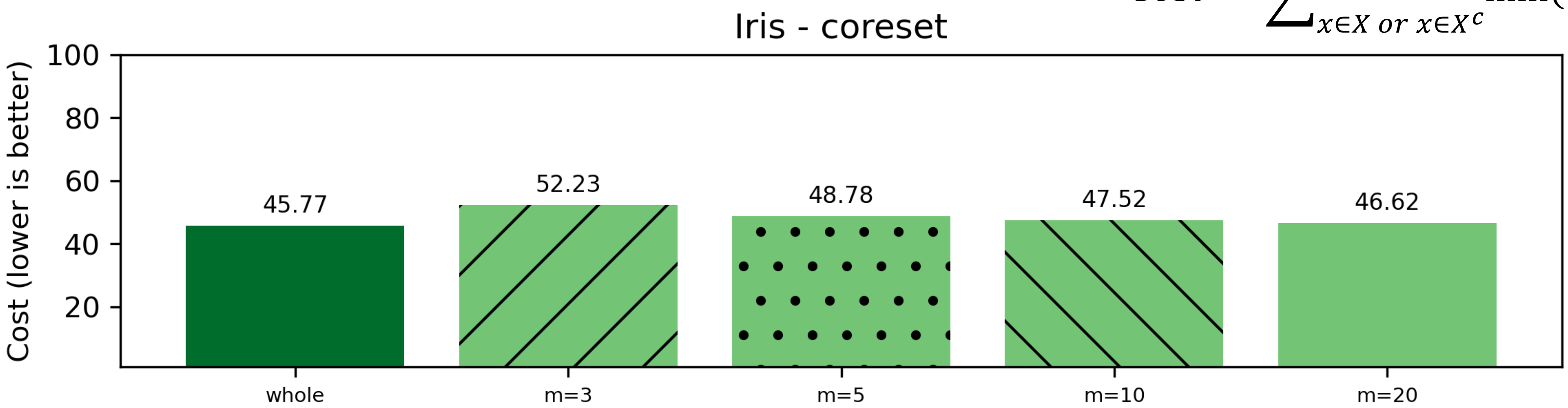
for *each* $b \in B$ **do**
  Set $P_b \leftarrow \{p \in P \mid \mathcal{B}(p) = b\}$
for *each* $b \in B$ *and* $p \in P_b$ **do**
  Set $\mathrm{Prob}(p) \leftarrow \dfrac{w(p)D(p, \mathcal{B}(p))}{2\sum_{q \in P} w(q)D(q, \mathcal{B}(q))} + \dfrac{w(p)}{2|B|\sum_{q \in P_b} w(q)}$.
Pick a sample S of at least m points from P such that for each $q \in S$ and $p \in P$ we have $q = p$ with probability $\mathrm{Prob}(p)$.
for *each* $p \in P$ **do**
  Set $u(p) \leftarrow \frac{w(p)}{|S| \cdot \mathrm{Prob}(p)}$.
Set $u(p) \leftarrow 0$ for each $p \in P \setminus S$. `/* Used only in the analysis. */`
return (S, u)

An example: 2-Means Clustering Problem on Iris dataset





$$Cost = \sum_{x \in X \ or \ x \in X^c} \min(|x - \mu_{-1}|^2, |x - \mu_{+1}|^2)$$



m = size of coreset

Implementation on a quantum computer

$$\mu_{-1} = \frac{\sum_{i \in S_{-1}} w_i x_i}{W_{-1}}$$

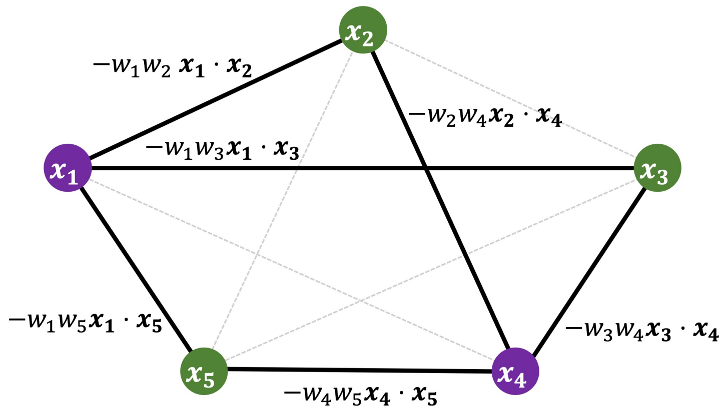
$$\mu_{+1} = \frac{\sum_{i \in S_{+1}} w_i x_i}{W_{+1}}$$

$$Cost\ Function: \sum_{i \in S_{-1}} w_i |x_i - \mu_{-1}|^2 + \sum_{i \in S_{+1}} w_i |x_i - \mu_{+1}|^2$$

Minimizing the above equation is in fact same as maximizing:

$$W_{+1} W_{-1} \|\mu_{+1} - \mu_{-1}\|^2 \xrightarrow{\text{If } W_{-1} = W_{+1}} \sum_{i \in S_{-1} j \in S_{+1}} -w_i w_j\, x_i . x_j$$

This problem is easy to interpret in terms of quantum optimisation problem (for a coreset size m=5):



$$\xleftrightarrow{\text{Hamiltonian for max-cut problem}} \quad H_{max-cut} = \sum_{i<j} w_i w_j\, x_i . x_j Z_i Z_j$$

Implementation on Quantum Approximate Optimisation Algorithm (QAOA)

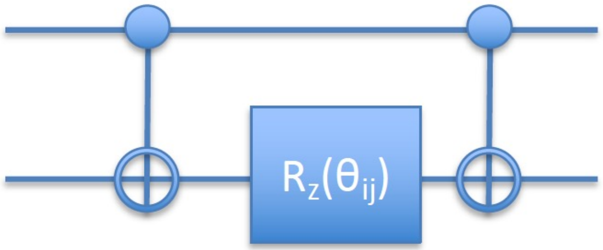
$$\boxed{H = \sum_{i \neq j} J_{ij} Z_i Z_j + \sum_i B_i Z_i}$$

For <u>every Z term</u> in the Hamiltonian:

$$H = \ldots + B_i \, Z_i + \ldots$$

$$\theta_{\mathrm{i}} = -\frac{B_i \alpha}{2}$$

For <u>every ZZ</u> term in the Hamiltonian:



$$H = \ldots + J_{ij} \, Z_i Z_j + \ldots$$

$$\theta_{ij} = -\frac{J_{ij} \alpha}{2}$$

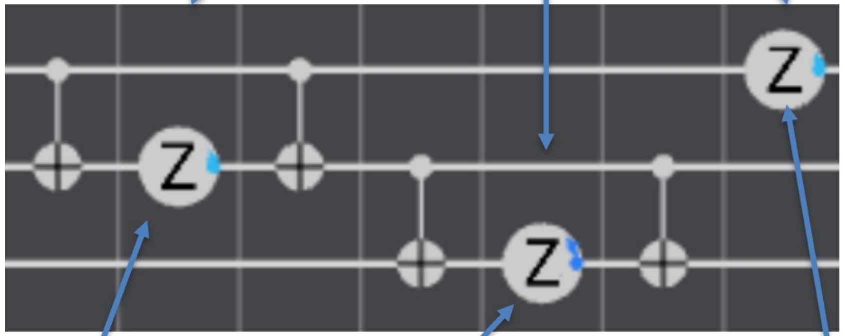
Angles all proportional to their term in the Hamiltonian.

Implementation on Quantum Approximate Optimisation Algorithm (QAOA)

Each term's sequence can be placed consecutively. Order of terms does not matter.

$$H = Z_1 Z_2 + 2Z_2 Z_3 + Z_1$$

Each term in the cost function gets a gate sequence.



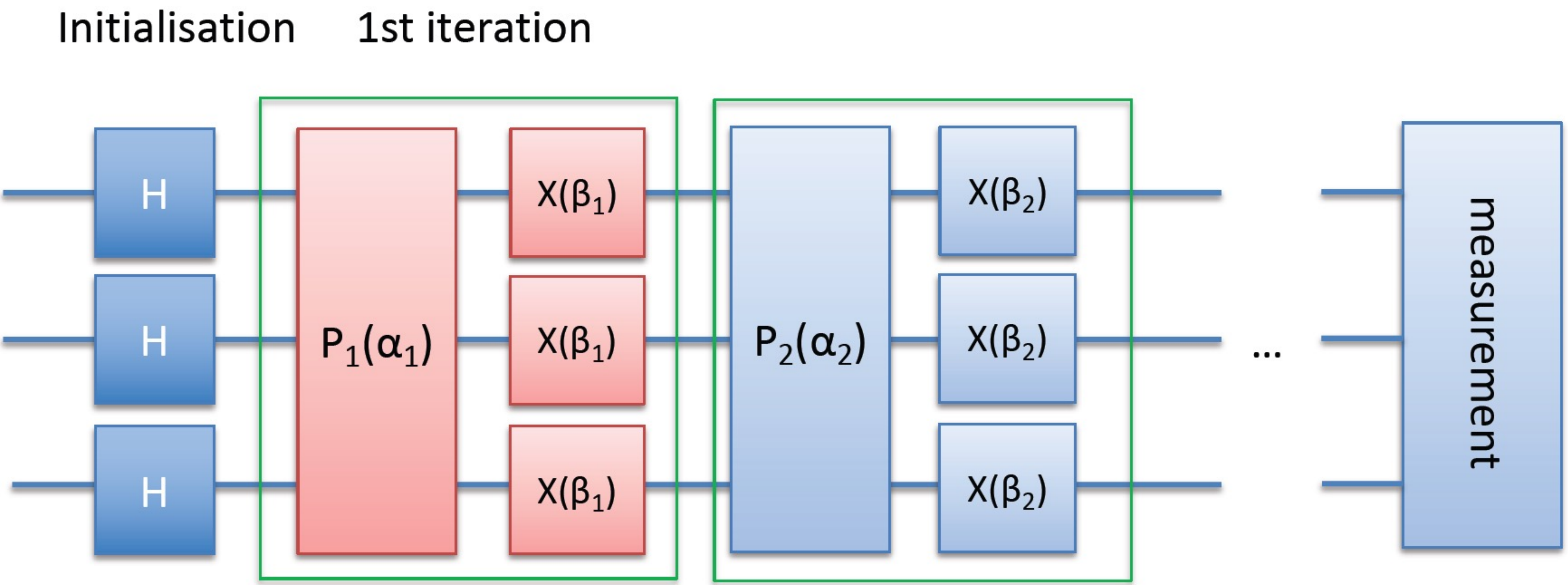
Each rotation angle is proportional to the energy term.

$$\theta_{12} = -\frac{\alpha}{2}$$

$$\theta_{23} = -\frac{2\alpha}{2} = -\alpha$$

$$\theta_1 = -\frac{\alpha}{2}$$

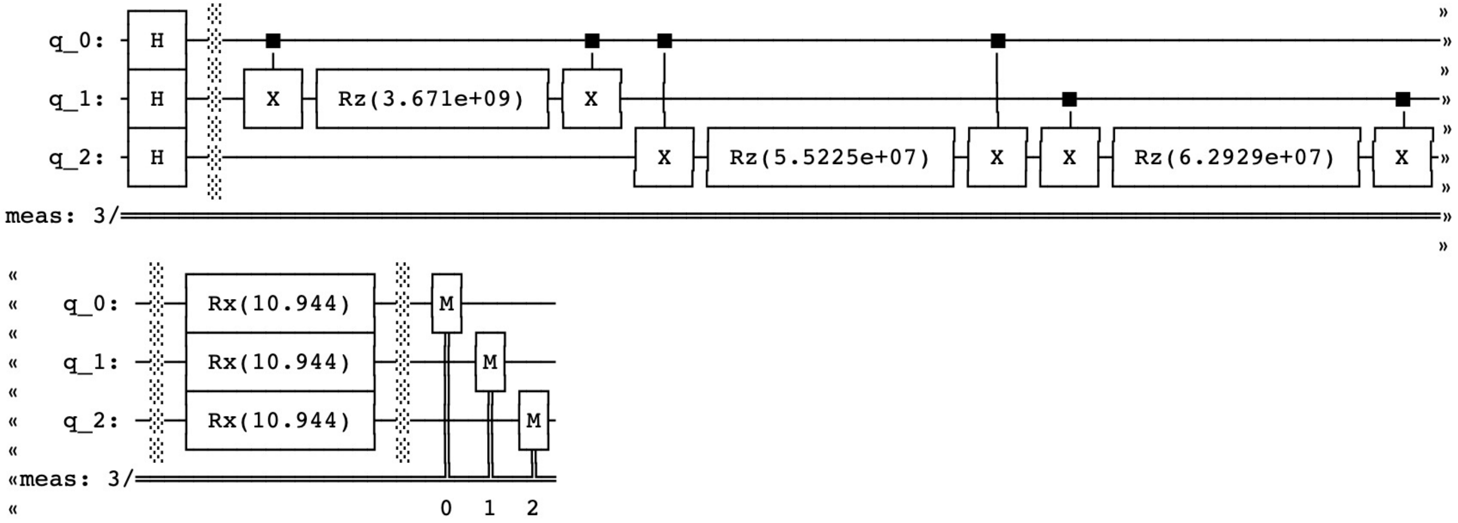
Implementation on Quantum Approximate Optimisation Algorithm (QAOA)

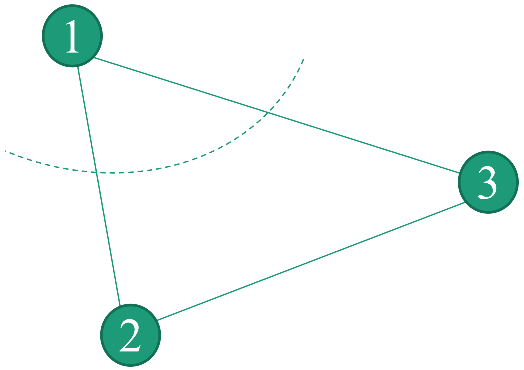


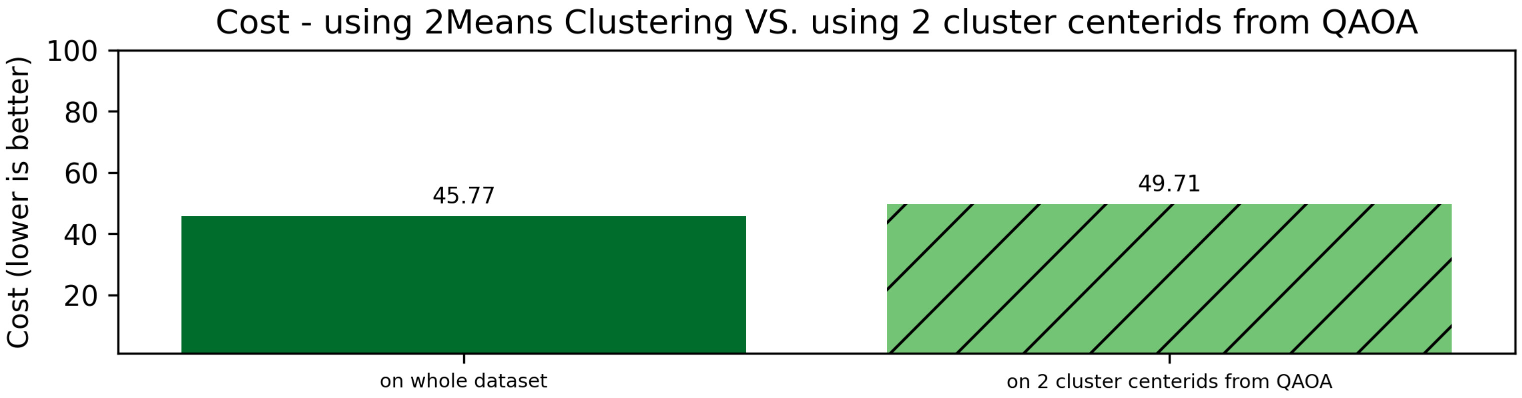
extract E value
re-set
parameters
α_i and β_i
repeat until min
is found

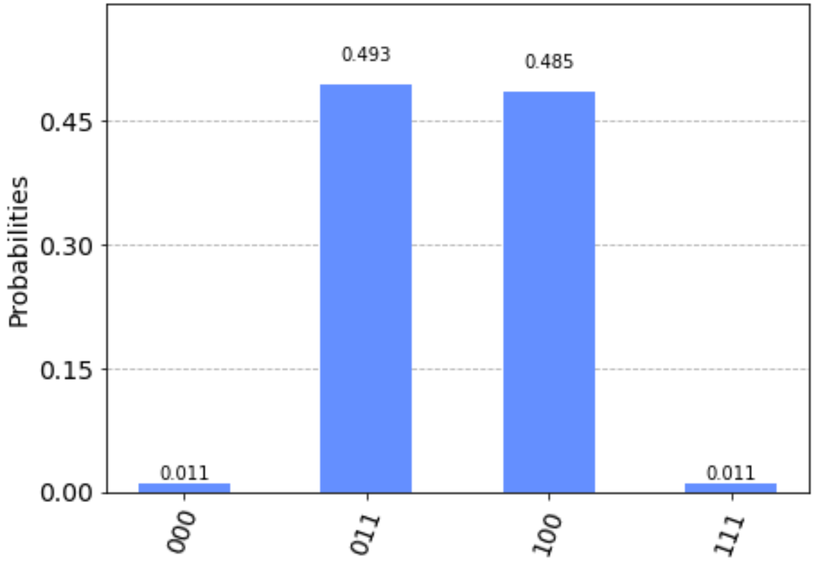
Implementation on QAOA

$$H_{max-cut} = \sum_{i<j} w_i w_j \, x_i . x_j Z_i Z_j \longrightarrow \text{H} = 511802775.7\ \text{ZZI} + 7699359.5\ \text{ZIZ} + 8773443.4\ \text{IZZ}$$









Open Questions

- How to find efficient coresets for a given dataset with non-trivial distribution ?
- Adaptive coreset construction (quantum/classical)?
- Coresets for other machine learning applications?
- Noise mitigation for quantum computing implementation?