Quantum Machine Learning:

Prospects and challenges

Iordanis Kerenidis

Why Quantum Machine Learning?

We need real-world high-impact applications of quantum computers

Reasons for ~~optimism~~ keep working on it

1. Powerful quantum tools for Linear Algebra

   Machine Learning is a lot of Linear Algebra – Matrix Multiplications, SVD, Linear Systems (neural nets, linear regression, Support Vector Machines,…)

   Quantum algorithms for Linear Algebra can offer speedups in certain cases

2. Distance Estimations

   Simple quantum circuits for estimating distances between quantum states

3. Noise resilient algorithms

   There is a lot of noise in ML data but the algorithms can deal with it

4. Multiple goals: Efficiency, Accuracy, Explainability, Energy, Trust

Reasons for ~~caution~~ keep working on it

1. Subtle quantum tools for Linear Algebra

   One needs to be very careful about when quantum algorithms can offer speedups

2. Loading classical data as quantum states

   Taking full advantage of quantum ML algorithms needs efficient quantum loaders

3. Getting classical information out of quantum algorithms

   The quantum output encodes a classical solution that needs to be extracted

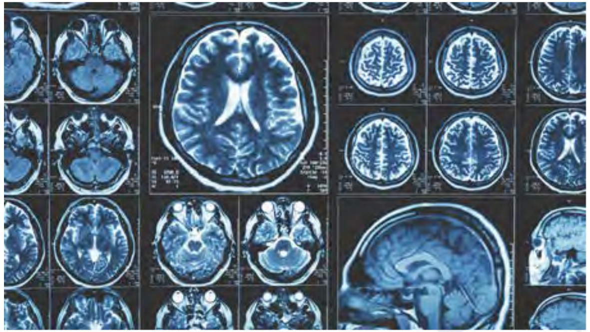
4. Benchmarking QML algorithms is difficult in the absence of hardware

   Machine Learning must work in practice! How do we test?

Supervised Learning:
a first example

Classification

Data

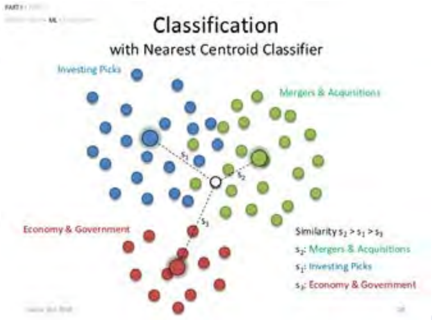


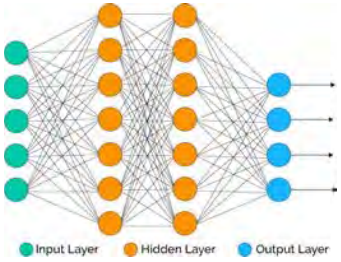
Classification

Data

→

Quantum algorithms



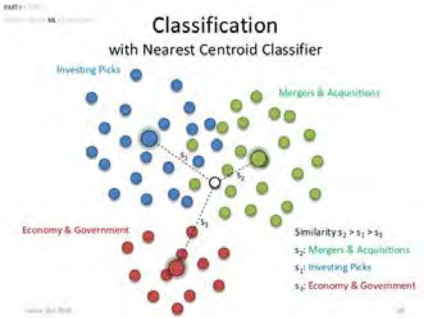


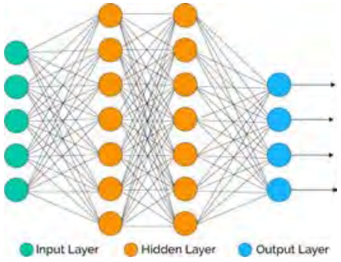
Classification

Data

→

Quantum algorithms





↓

Quantum software

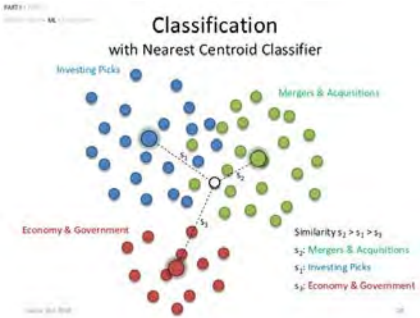
```
# let's create some synthetic data
X, y = generate_data_clusters()

# let's run the quantum classifier
qlabels = fit_and_predict(X,y=y,model='QNearestCentroid')

#import NearestCentroid from scikit-learn for benchmarking
clabels = sklearn.neighbors.NearestCentroid().fit(X,y).predict(X)

print('Quantum labels\n',qlabels)
print('Classical labels\n',clabels)

# let's plot the data (only for dimension=2)
plot(X,qlabels,'QNearestCentroid')
plot(X,clabels,'KNearestCentroid')

Quantum labels
 [2 0 0 1 0 0 0 0 1 1 2 0 1 1 0 1 2 3 2 1 2 2 2 3 3 3 3 3 0 3 3 2]
Classical labels
 [2 0 0 1 0 0 0 0 1 1 2 0 1 1 0 1 2 3 2 1 2 2 2 3 3 3 3 3 0 3 3 2]
```

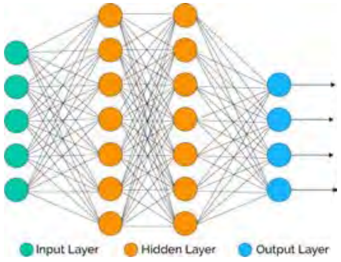
Classification

Data

→

Quantum algorithms





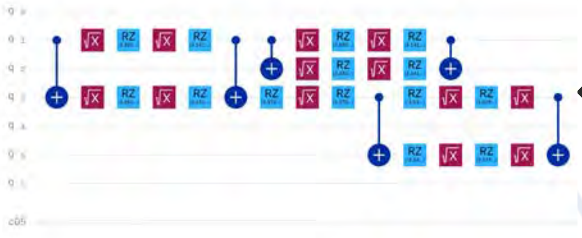
↓

Quantum software

```
# let's create some synthetic data
X, y = generate_data_clusters()

# let's run the quantum classifier
qlabels = fit_and_predict(X,y=y,model='QNearestCentroid')

#import NearestCentroid from scikit-learn for benchmarking
clabels = sklearn.neighbors.NearestCentroid().fit(X,y).predict(X)

print('Quantum labels\n',qlabels)
print('Classical labels\n',clabels)

# let's plot the data (only for dimension=2)
plot(X,qlabels,'QNearestCentroid')
plot(X,clabels,'KNearestCentroid')

Quantum labels
 [2 0 0 1 0 0 0 0 1 1 2 0 1 1 0 1 2 3 2 1 2 2 2 3 3 3 3 3 0 3 3 2]
Classical labels
 [2 0 0 1 0 0 0 0 1 1 2 0 1 1 0 1 2 3 2 1 2 2 2 3 3 3 3 3 0 3 3 2]
```

←

Quantum circuits

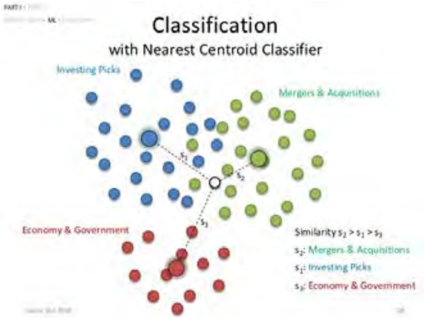


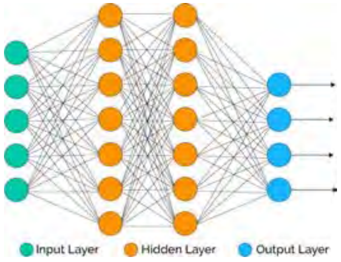
←

Classification

Data

Quantum algorithms

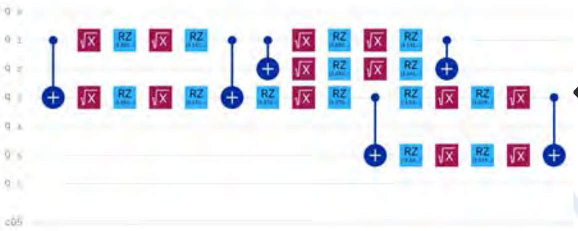




Quantum software

```
# let's create some synthetic data
X, y = generate_data_clusters()

# let's run the quantum classifier
qlabels = fit_and_predict(X,y=y,model='QNearestCentroid')

#import NearestCentroid from scikit-learn for benchmarking
clabels = sklearn.neighbors.NearestCentroid().fit(X,y).predict(X)

print('Quantum labels\n',qlabels)
print('Classical labels\n',clabels)

# let's plot the data (only for dimension=2)
plot(X,qlabels,'QNearestCentroid')
plot(X,clabels,'KNearestCentroid')

Quantum labels
 [2 0 0 1 0 0 0 0 1 1 2 0 1 1 0 1 2 3 2 1 2 2 2 3 3 3 3 3 0 3 3 2]
Classical labels
 [2 0 0 1 0 0 0 0 1 1 2 0 1 1 0 1 2 3 2 1 2 2 2 3 3 3 3 3 0 3 3 2]
```

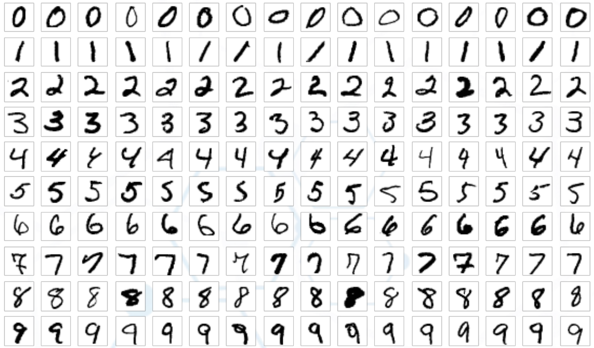
Quantum circuits



Quantum computer

Classification

Data

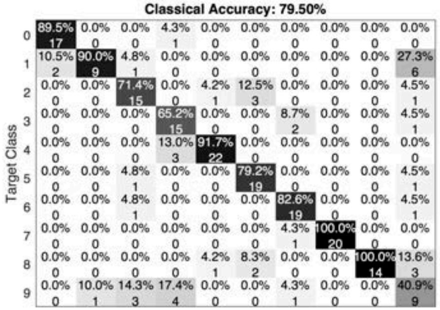


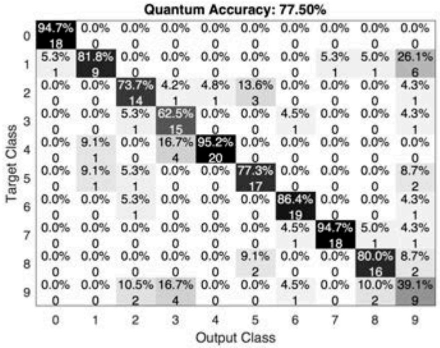
Nearest Centroid Classification on a Trapped Ion Quantum Computer

Sonika Johri,[1] Shantanu Debnath,[1] Avinash Mocherla,[2,3] Alexandros Singh,[2,4] Anupam Prakash,[2] Jungsang Kim,[1] and Iordanis Kerenidis[2,5]

[1] *IonQ Inc, 4505 Campus Dr, College Park, MD 20740*
[2] *QC Ware, Palo Alto, USA and Paris, France*
[3] *UCL, Centre for Nanotechnology, London, UK*
[4] *Université Sorbonne Paris Nord, France*
[5] *CNRS, University of Paris, France*

Results

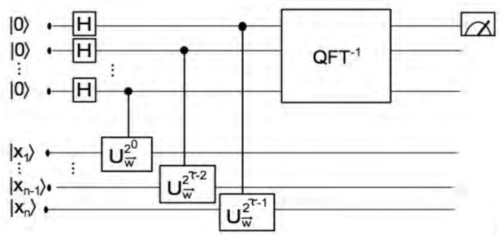


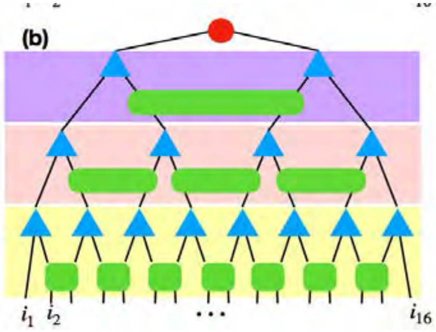


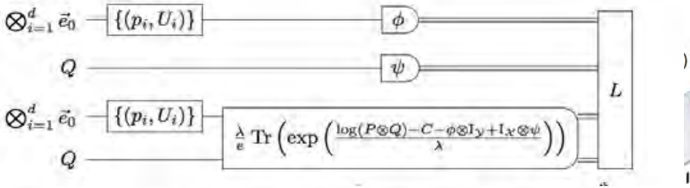
An 11-qubit QC can recognise 8 out of 10 handwritten digits

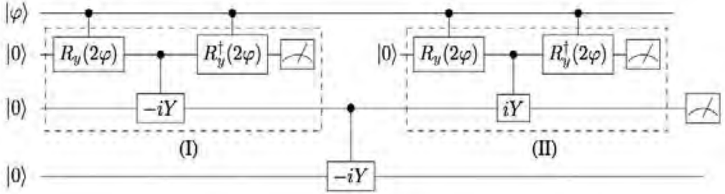
Classification: Quantum Neural Networks

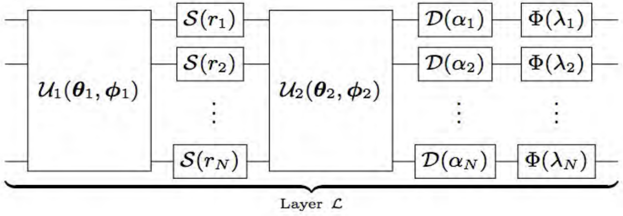
1. Many different proposals [arXiv:1412.3635, arXiv:1810.03787, arXiv:1711.11240, arXiv:1806.06871, arXiv:1806.06871, arXiv:1911.00111, arXiv:1909.12264]

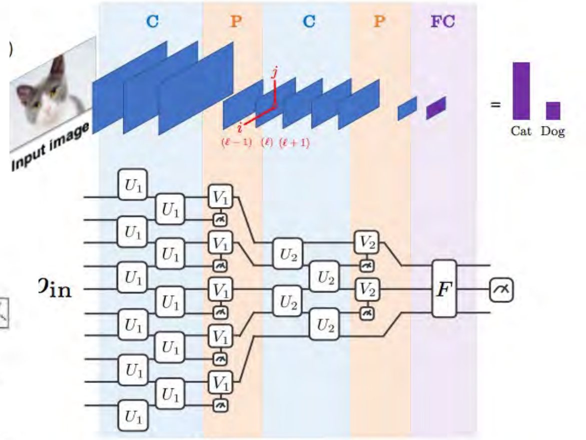


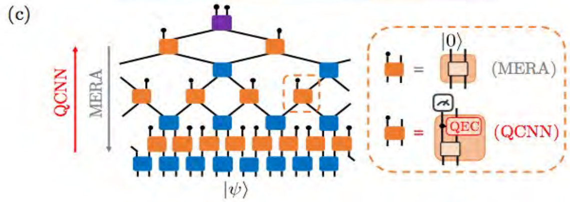






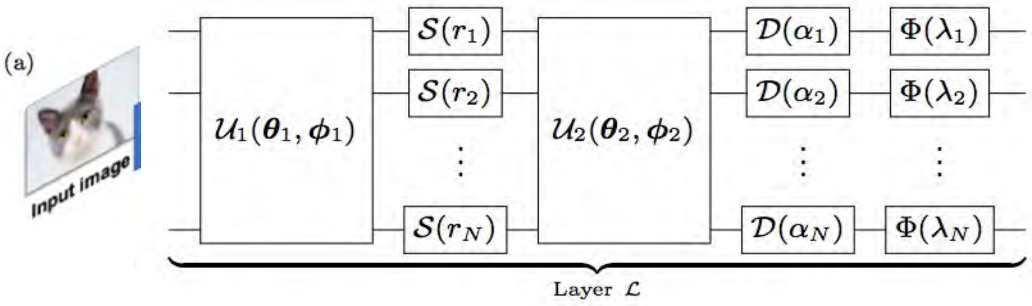






Classification: Quantum Neural Networks

1. Many different proposals [arXiv:1412.3635, arXiv:1810.03787, arXiv:1711.11240, arXiv:1806.06871, arXiv:1806.06871, arXiv:1911.00111, arXiv:1909.12264]



cat / dog

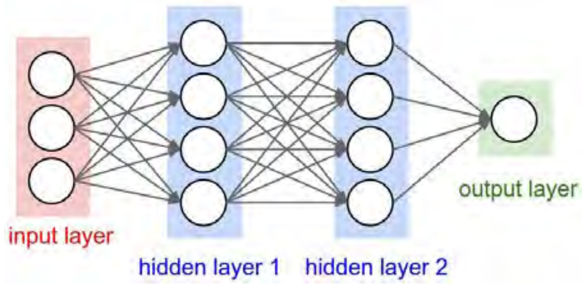
Quantum Neural Network

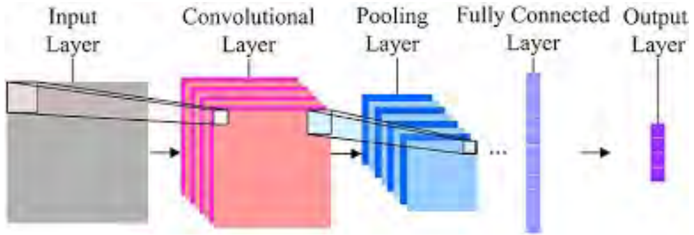
Quantum circuit with a number of parametrized gates

Input: image. Output: label

Training: Learn the gate parameters so that labels are correct

Classification: Training classical Neural Networks





Accuracy/Convergence : similar to classical Neural Networks

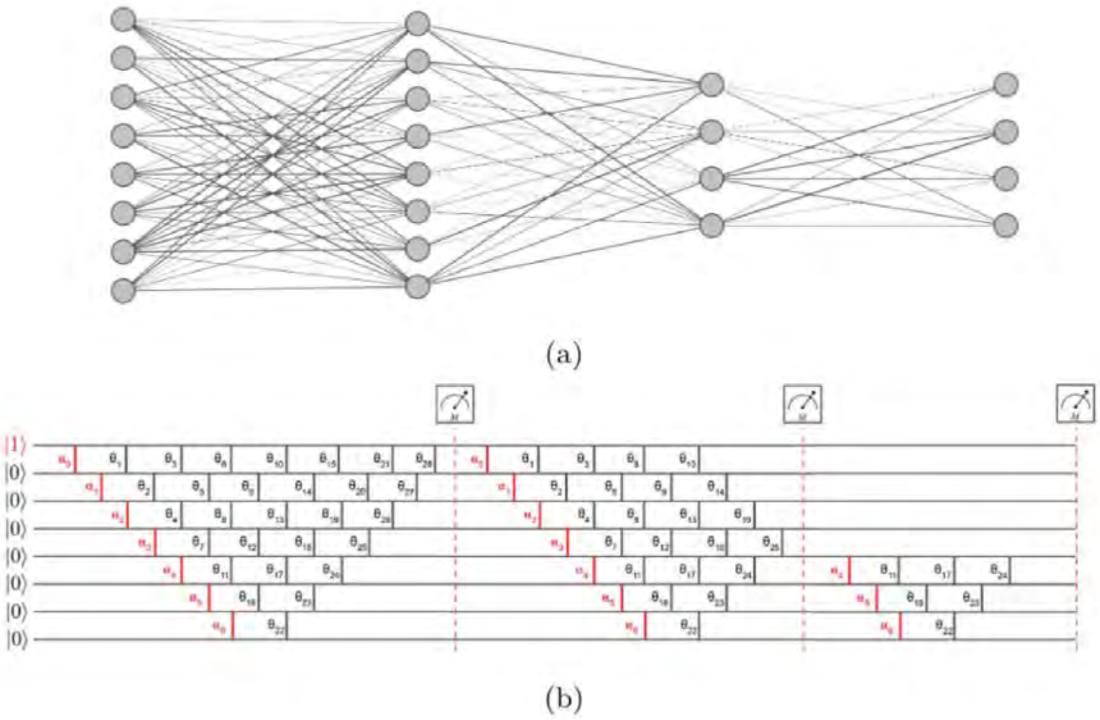
Running time: gains from IP estimation for training and evaluation

[Allcock, Hsieh, Kerenidis, Zhang ACM ToQC 20], [Kerenidis,Landman,Prakash ICLR20]

Classification: Quantum Neural Networks

Quantum Orthogonal NNs

- New classical training in $O(n^2)$
- NISQ implementations
- provable efficiency
- A new optimization landscape

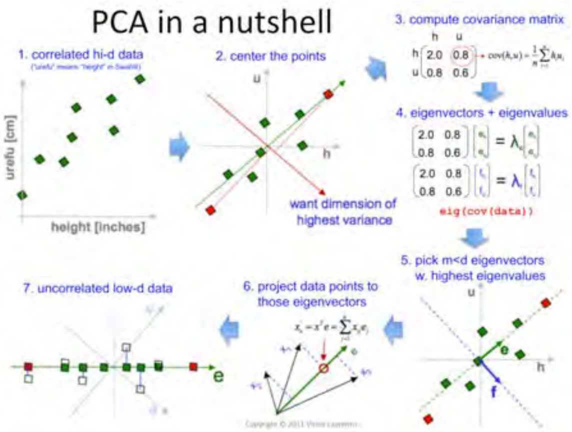


Dimensionality Reduction

Map data from R^D to a smaller space R^d where the classification is better

- Principal Component Analysis
- Linear Discriminant
- Slow Feature Analysis

Heavier Linear Algebra (SVDs, projections on sub-eigenspaces, etc.)



Dimensionality Reduction

Map data from R^D to a smaller space R^d where the classification is better

- Principal Component Analysis
- Linear Discriminant
- Slow Feature Analysis

Heavier Linear Algebra (SVDs, projections on sub-eigenspaces, etc.)

Classification

Map points from R^D to R^d via Linear Algebra (can take $O(ND^2)$)

Perform Classification in R^d ("easier" part)

Quantum Dimensionality Reduction

Map data from R^D to a smaller space R^d where the classification is better

- Principal Component Analysis [Loyd, Mohseni, Rebentrost 13]
- Linear Discriminant [Cong, Duan 15]
- Slow Feature Analysis [Kerenidis, Luongo 18]

Heavier Linear Algebra (SVDs, projections on sub-eigenspaces, etc.)

Quantum Classification

Map points from R^D to R^d via Quantum Linear Algebra

Perform Quantum Classification in R^d

Quantum Dimensionality Reduction

Map data from R^D to a smaller space R^d where the classification is better

- Principal Component Analysis [Loyd, Mohseni, Rebentrost 13]
- Linear Discriminant [Cong, Duan 15]
- Slow Feature Analysis [Kerenidis, Luongo 18]

Heavier Linear Algebra (SVDs, projections on sub-eigenspaces, etc.)

Singular Value Estimation (Phase Estimation for non-unitaries) [Kerenidis,Prakash 17]

Given $A = \sum_{i=1}^{n} \lambda_i |v_i\rangle\langle v_i|$ and eigenvector $|v_i\rangle$, perform $|v_i\rangle|0\rangle \rightarrow |v_i\rangle|\lambda_i\rangle$

Quantum Dimensionality Reduction

Map data from R^D to a smaller space R^d where the classification is better

- Principal Component Analysis [Loyd, Mohseni, Rebentrost 13]
- Linear Discriminant [Cong, Duan 15]
- Slow Feature Analysis [Kerenidis, Luongo 18]

Heavier Linear Algebra (SVDs, projections on sub-eigenspaces, etc.)

Singular Value Estimation (Phase Estimation for non-unitaries) [Kerenidis,Prakash 17]

Given $A = \sum_{i=1}^{n} \lambda_i |v_i\rangle\langle v_i|$ and eigenvector $|v_i\rangle$, perform $|v_i\rangle|0\rangle \rightarrow |v_i\rangle|\lambda_i\rangle$

Goal: Project $|b\rangle = \sum_{i=1}^{n} b_i |v_i\rangle$ onto an eigenspace of $A = \sum_{i=1}^{n} \lambda_i |v_i\rangle\langle v_i|$ with $\lambda_i > t$

$$\sum_{i=1}^{n} b_i |v_i\rangle \rightarrow \sum_{i=1}^{n} b_i |v_i\rangle|\lambda_i\rangle \rightarrow \sum_{i:\lambda_i > t} b_i |v_i\rangle|\lambda_i\rangle|0\rangle + \sum_{i:\lambda_i < t} b_i |v_i\rangle|\lambda_i\rangle|1\rangle$$

Quantum Dimensionality Reduction

Map data from R^D to a smaller space R^d where the classification is better

- Principal Component Analysis [Loyd, Mohseni, Rebentrost 13]
- Linear Discriminant [Cong, Duan 15]
- Slow Feature Analysis [Kerenidis, Luongo 18]

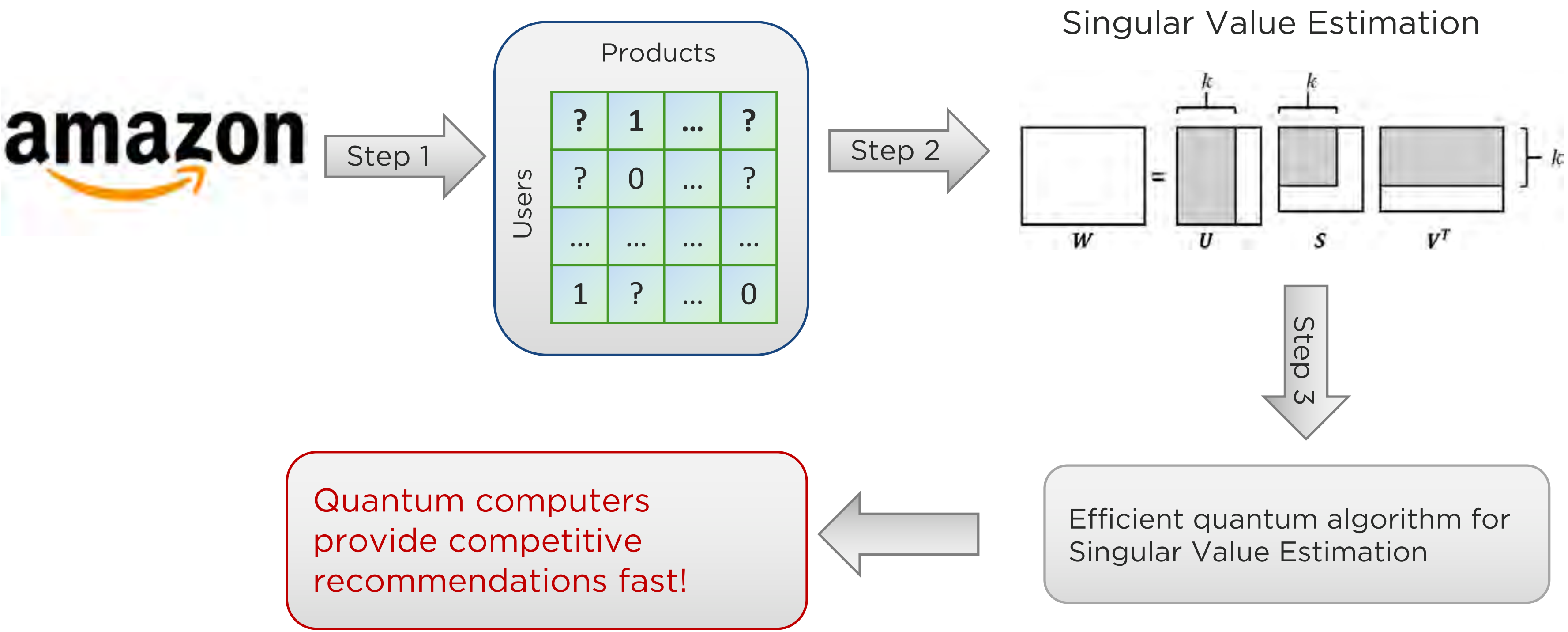
Heavier Linear Algebra (SVDs, projections on sub-eigenspaces, etc.)

Singular Value Estimation (Phase Estimation for non-unitaries) [Kerenidis,Prakash 17]

Given $A = \sum_{i=1}^{n} \lambda_i |v_i\rangle\langle v_i|$ and eigenvector $|v_i\rangle$, perform $|v_i\rangle|0\rangle \rightarrow |v_i\rangle|\lambda_i\rangle$

Running time parameters: $O(\kappa, \mu, 1/\varepsilon)$

Recommendation Systems [Kerenidis, Prakash, ITCS 17]



Recommendation Systems [Kerenidis, Prakash, ITCS 17]

amazon

Algorithms	Theory	Parameters Users: 10^8 Products: 10^8 types: 100
Quantum	types* log (users*products)	~ 10^3
CUR-based [2002]	$(types)^2$ * products	~ 10^{12}
FKV-based [Tang 2018, CGLLTW19]	$(types)^8$ * log (users*products)	~ 10^{17}

On the Input

Loading data for Quantum Machine Learning

In cases, quantum inputs can be very interesting for applications

In cases, it is easy to construct the input: Reinforcement Learning, NN for PDES

In cases, quantum input coming from Dim. Reduction / kernels / etc.

Classical: Heavy Linear Algebra to produce input states
Quantum: Loading cost of initial points is subsumed

In cases, one will need specific quantum data loaders

Quantum circuits of size O(d) and depth log(d)

1. Quantum data loaders

Goal: Load N-dim classical data onto quantum computer

Solution:

1. Map data point to gate parameters in linear time
2. Build quantum circuit to run in $logN$ steps

$$\mathrm{RBS}(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
T  : |0|1|2|3|

q0 : -X-B-B-B-
        | | |
q1 : ---|-|-S-
        | |
q2 : ---|-S-B-
        |   |
q3 : ---|---S-
        |
q4 : ---S-B-B-
          | |
q5 : -----|-S-
          |
q6 : -----S-B-
            |
q7 : -------S-

T  : |0|1|2|3|
```

1. Quantum data loaders

Data loaders

	qubits (Q)	depth (D)	multiqubit gates	Remarks
multiplexer	$logN$	$O(N)$	$O(NlogN)$	Impractical depth
QRAM hardware [Lloyd]	$O(N)$?	$O(logN)$?	Light-matter interaction gates	New hardware is needed
QRAM circuit [Mosca et al]	$O(N)$	$O(N)$	$O(NlogN)$	Impractical depth
Our parallel data loader	N	$logN$	$(N-1)$ 2-q gates	Unary encoding
Our optimized data loader	$2\sqrt{N}$	$\sqrt{N}logN$	1.5N 2-q gates	Any values s.t. $Q * D = O(NlogN)$

2. Quantum distance estimation

Goal: Given data points x and y, find their distance

Solution:

1. Build circuit [loader(x)+loader(y)†] to estimate distance in $2logN$ steps

```
T  : |0|1|2|3|4|5|6|

q0 : -X-B-B-B-B-B-B-
        | | | | | |
q1 : ---|-|-S-S-|-|-
        | |     | |
q2 : ---|-S-B-B-S-|-
        |   | |   |
q3 : ---|---S-S---|-
        |         |
q4 : ---S-B-B-B-B-S-
          | | | |
q5 : -----|-S-S-|---
          |     |
q6 : -----S-B-B-S---
            | |
q7 : -------S-S-----

T  : |0|1|2|3|4|5|6|
```

Properties

- Shallow and noise robust circuits
- N qubits, $2logN$ depth
- Can use any optimized data loader
- Can be combined with Amplitude Estimation

Unsupervised Learning

Clustering

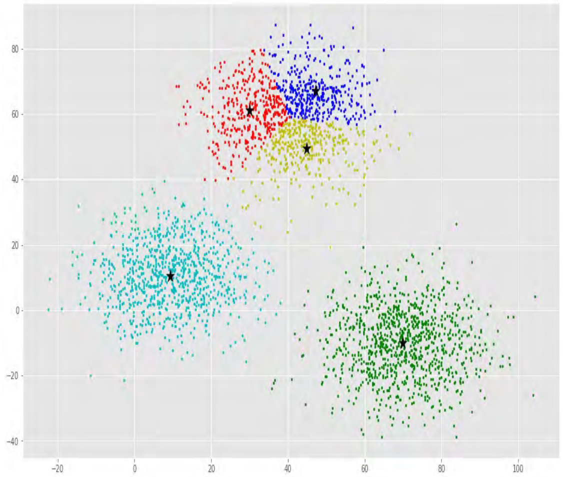
k-means++

Input: N points in d-dimensions
Output: K clusters/centroids

1. Start with some initial centroids (e.g. ++-method)

Repeat until convergence

2. For each point
   estimate distances to centroids and
   assign to closest cluster
3. Update the centroids



Clustering

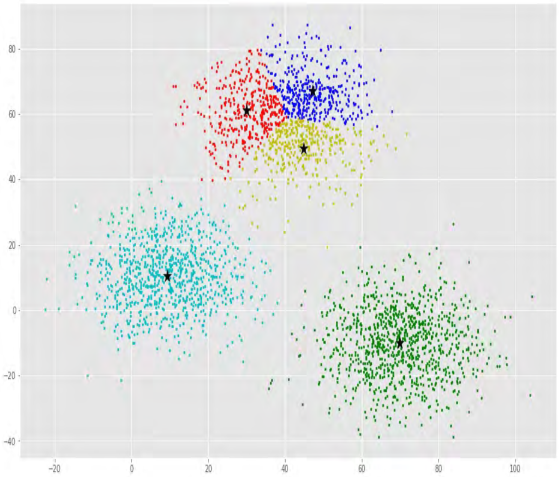
q-means++ [Kerenidis,Landman,Luongo, Prakash NeurIPS 2019]

Input: N points in d-dimensions (quantum access)
Output: K clusters/centroids

1. Start with some initial centroids (e.g. ++-method)

Repeat until convergence

  2. For all points in superposition
     estimate distances to centroids and
     assign to closest cluster
  3. Update the centroids
     i. Quantum linear algebra to find new centroid
     ii. Tomography to recover classical description



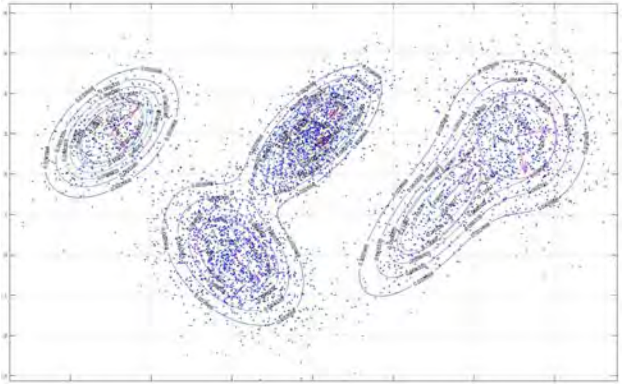
Clustering

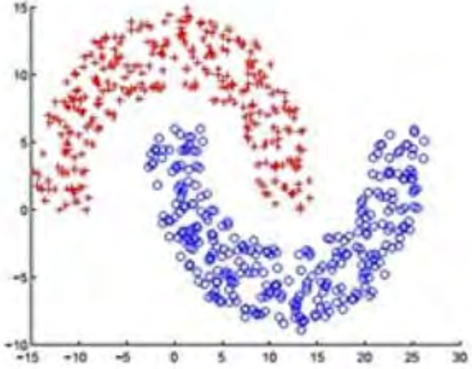
Extensions

Expectation Maximization for Gaussian Mixture Models
[Kerenidis,Luongo,Prakash ICML 2020]

Spectral Clustering [Kerenidis,Landman PRA 2021]

Map points to the low eigenspace of the Laplacian, then apply k-means





(b) Spectral Clustering

Reinforcement Learning

Quantum Policy Iteration [Cherrat, Kerenidis,Prakash 2021]

Input:
states S, actions A, transition matrix P, Reward function R

Output: policy π, such that π(s)=a

Policy Iteration

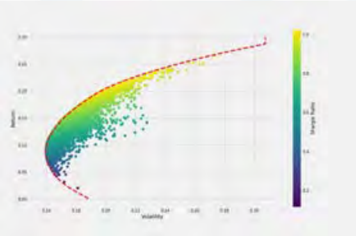
start with π_0
solve $(I - \gamma P^{\pi}) Q = R$
update π from Q

Remarks

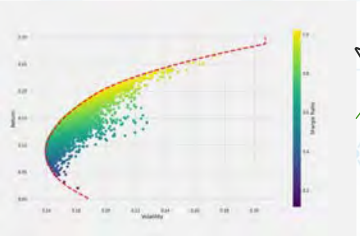
"No input" / Well-conditioned / ℓ^{∞} guarantees

Quantum Optimization

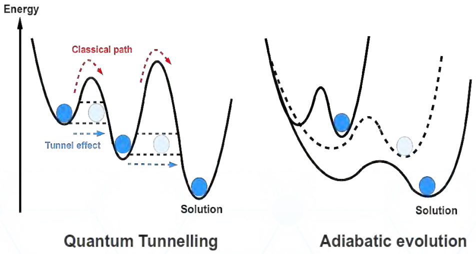
Optimization



Optimization

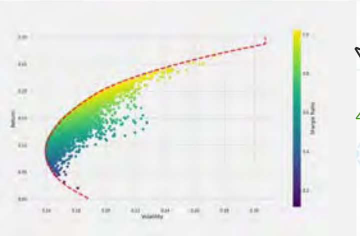


Analog Combinatorial opt



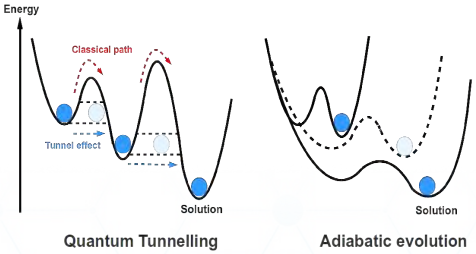
→

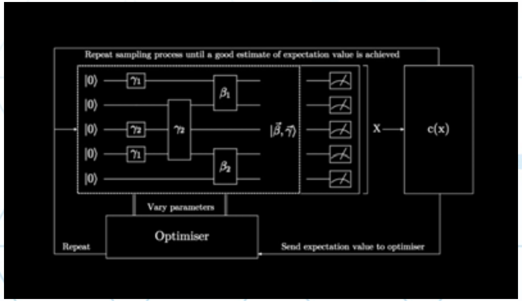
Optimization

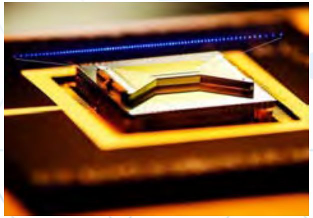


Analog Combinatorial opt

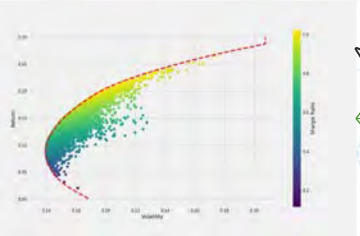
Discrete Combinatorial opt







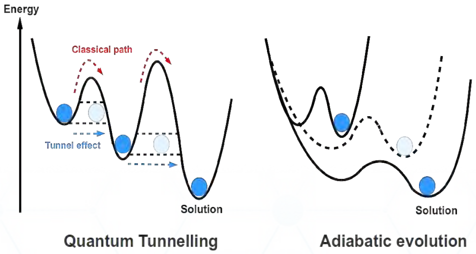
Optimization

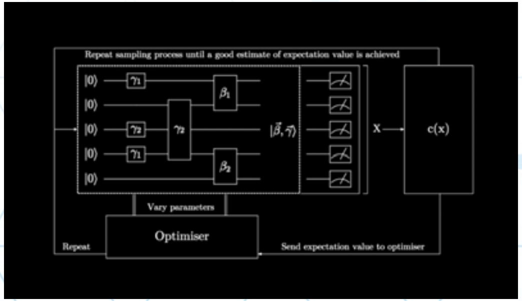


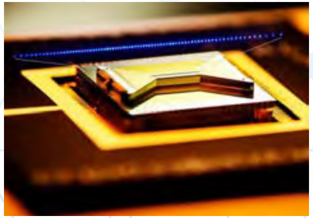
Analog Combinatorial opt

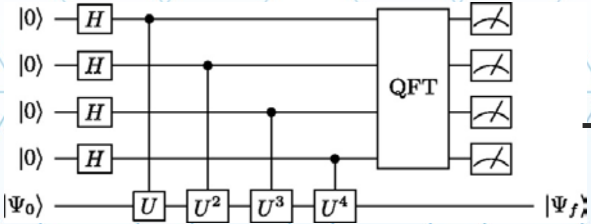
Discrete Combinatorial opt

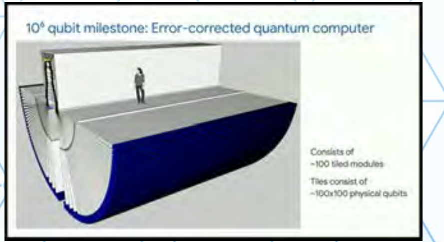
Convex opt











[Kerenidis,Prakash 2018]
[Kerenidis,Prakash,Szilágyi 2020]

Challenges - Prospects

Powerful yet subtle quantum tools

Power: Linear Algebra, Distance Estimations, Tomography, etc.
Subtleties: Input, Output, running time parameters

Promising directions

Heavy Linear Algebra algorithms (Dim. Reduction, Kernels, Spectral Clustering)
Reinforcement Learning (no classical data, well-conditioned systems, ℓ^∞ guarantees)
Quantum Neural Networks

Final Remarks

ML is about practical solutions to real-world problems.

It’s a long, arduous way till we see QML applications, but certainly worth pursuing

Thank you!

Iordanis Kerenidis

jkeren@irif.fr, iordanis.kerenidis@qcware.com

cnrs dépasser les frontières

QCWARE