## Quantum methods for Optimization and Machine Learning

Iordanis Kerenidis Paris Centre for Quantum Computing PCQC, CNRS Paris



## Quantum Algorithms for Optimisation / ML



### The HHL algorithm [Harrow, Hassidim, Lloyd 2009]

Quantum computers provide a quantum solution to a system of linear equations in certain cases exponentially faster than classical algorithms, given quantum access to the data.

## Quantum Algorithms for Optimisation / ML



The HHL algorithm [Harrow, Hassidim, Lloyd 2009]

Quantum computers provide a quantum solution to a system of linear equations in certain cases exponentially faster than classical algorithms, given quantum access to the data.

## Quantum Algorithms for Optimisation / ML



The HHL algorithm [Harrow, Hassidim, Lloyd 2009]

Quantum computers provide a quantum solution to a system of linear equations in certain cases exponentially faster than classical algorithms, given quantum access to the data.

*"It\$opens\$the\$possibility\$of\$drama4c\$speedups\$for\$machine\$learning\$tasks,\$richer\$models\$for\$data\$sets\$and\$ more natural settings for learning and inference"* Quantum Machine Learning Workshop during NIPS 2015

Remark: "Solving" systems of linear equations is BQP-complete

## Three remarks on Quantum Machine Learning PCOC



QML needs a full-scale computer with quantum access to data for "exponential savings"

## Three remarks on Quantum Machine Learning PCOC



QML needs a full-scale computer with quantum access to data for "exponential savings"

Most overhyped and underestimated field at the same time

## Three remarks on Quantum Machine Learning PCOC



QML needs a full-scale computer with quantum access to data for "exponential savings"

Most overhyped and underestimated field at the same time

(One of) the most convincing reasons to build quantum computers

## Quantum Machine Learning: the model

### Data storage and quantum access

- Data can be accessed quantumly directly 1.
- Quantum RAM : Efficient storage of classical data allowing quantum access to it  $2.$ 
	- It takes polylogarithmic time to store/update/delete an element (i,j,a<sub>ii</sub>)
	- Query in polylogarithmic time  $\sum c_{ij} |i, j, 0\rangle \rightarrow \sum c_{ij} |i, j, a_{ij}\rangle$
- 3. Other access models...

## Quantum Machine Learning: the model

#### Data storage and quantum access

- 1. Data can be accessed quantumly directly
- Quantum RAM : Efficient storage of classical data allowing quantum access to it  $2.$ 
	- It takes polylogarithmic time to store/update/delete an element (i,j, $a_{ij}$ )
	- Query in polylogarithmic time  $\sum c_{ij} |i, j, 0\rangle \rightarrow \sum c_{ij} |i, j, a_{ij}\rangle$
- 3. Other access models...

### Computation on the data

- Given quantum access to data, learn some property of the data
- Running time of quantum algorithm can be more efficient that classical



## Use-case: Recommendation Systems





# PCOC



![](_page_12_Figure_0.jpeg)

![](_page_13_Figure_0.jpeg)

![](_page_14_Figure_0.jpeg)

![](_page_15_Figure_0.jpeg)

## General quantum methods for Optimization

![](_page_16_Picture_1.jpeg)

### Iterative methods (ubiquitous in practice)

- 1. Start with an initial solution.
- 2. Update the solution according to an Update Rule
- 3. Repeat until the solution is satisfactory

### **Types of Iterative Methods**

First order - Gradient Descent Second order - Interior point methods

![](_page_16_Figure_8.jpeg)

## General quantum methods for Optimization

![](_page_17_Picture_1.jpeg)

### Iterative methods (ubiquitous in practice)

- 1. Start with an initial solution.
- 2. Update the solution according to an Update Rule
- 3. Repeat until the solution is satisfactory

### **Types of Iterative Methods**

First order - Gradient Descent Second order - Interior point methods

**Efficient Quantum Gradient Descent algorithm** for Linear Systems and Stochastic Least Squares. [Kerenidis, Prakash 2017]

![](_page_17_Figure_9.jpeg)

## General quantum methods for Optimization

![](_page_18_Picture_1.jpeg)

### Iterative methods (ubiquitous in practice)

- 1. Start with an initial solution.
- 2. Update the solution according to an Update Rule
- 3. Repeat until the solution is satisfactory

### **Types of Iterative Methods**

First order - Gradient Descent Second order - Interior point methods

**Efficient Quantum Gradient Descent algorithm** for Linear Systems and Stochastic Least Squares. [Kerenidis, Prakash 2017]

![](_page_18_Figure_9.jpeg)

Remark 1: Improved Linear Algebra

Remark 2: Great savings in QRAM

![](_page_19_Picture_1.jpeg)

### Problem:

Given matrix A and vector x, output Ax, A<sup>-1</sup>x, ...

### Problem:

Given matrix A and vector x, output Ax, A<sup>-1</sup>x, ...

## Step 1

Map A to some unitary U s.t.

- 1. The spectra of A and U are related
- 2. U is efficient to implement

 $A/\mu(A) = P \circ Q$ ,  $U = (2PP<sup>t</sup> - I)(2QQ<sup>t</sup> - I)$ 

Efфiciency via QRAM data structures

![](_page_20_Picture_9.jpeg)

### Problem:

Given matrix A and vector x, output Ax, A<sup>-1</sup>x, ...

## Step 1

Map A to some unitary U s.t.

- 1. The spectra of A and U are related
- 2. U is efficient to implement

### Step 2

Phase Estimation on U

LCU, Qubitization on U

 $A/\mu(A) = P \circ Q$ ,  $U = (2PP<sup>t</sup> - I)(2QQ<sup>t</sup> - I)$ 

Eficciency via QRAM data structures

Apply a circuit with  $O(log1/\epsilon)$  U's

![](_page_21_Picture_13.jpeg)

#### Problem:

Given matrix A and vector x, output Ax, A<sup>-1</sup>x, ...

### Step 1

Map A to some unitary U s.t.

- 1. The spectra of A and U are related
- 2. U is efficient to implement

#### Step 2

Phase Estimation on U

LCU, Qubitization on U

#### Step 3

Amplitude Amplification (VT)

 $A/\mu(A) = P \circ Q$ ,  $U = (2PP<sup>t</sup> - I)(2QQ<sup>t</sup> - I)$ 

Eficciency via QRAM data structures

Apply a circuit with  $O(log1/\epsilon)$  U's

 $O(K(A))$  iterations

![](_page_22_Picture_16.jpeg)

![](_page_23_Picture_1.jpeg)

### Problem:

Given matrix A and vector x, output Ax, A<sup>-1</sup>x, ...

Running time: O(K(A)µ(A)log1/ $\varepsilon$ )

![](_page_24_Picture_1.jpeg)

Given matrix A and vector x, output Ax, A<sup>-1</sup>x, ...

Running time: O(K(A)µ(A)log1/ $\varepsilon$ )

**Open Question:** What is the optimal  $\mu(A)$ ?

![](_page_25_Picture_0.jpeg)

#### **Could QML work on real data? Represent a novel quantum Frobenius** Could QML work on real data? We will estimate *Fk*(*x*(0)) efficiently using the algorithm below. From our QRAM construction  $\mathsf{D}\mathsf{f}\mathsf{K}$  on real data for running time is logarithmic in the dimension and number of  $\mathsf{F}\mathsf{K}$

#### **Frobenius Distance Classification**<br>To the total number of points and norm of the cluster (see also Appendix A). **Frobenius Distance Classification**

QFE 4 Frobenius Distance Estimator

Require:

**QRAM** access to the matrix  $X_k$  of cluster  $k$  and to a test vector  $x(0)$ . Error parameter  $\eta \geq 0$ . Ensure: An estimate  $F_k(x(0))$  such that  $|F_k(x(0)) - F_k(x(0))| < \eta$ . *<sup>X</sup>*(0) <sup>2</sup> <sup>R</sup>*|Tk|*⇥*<sup>d</sup>* which just repeats the row *<sup>x</sup>*(0) *<sup>|</sup>Tk<sup>|</sup>* times. Then, we define  $F_k(x(0)) = \frac{\left\|X_k - X(0)\right\|_F^2}{2\left(\left\|X_k\right\|_F^2 + \left\|X_k(x)\right\|_F^2\right)}$ *F*  $2(\left\|X_k\right\|_F^2 + \left\|X(0)\right\|_F^2)$ 

we can create a superposition of all vectors in the cluster as  $\alpha$  and  $\alpha$  as  $\alpha$ 

![](_page_26_Picture_5.jpeg)

 $709 / 8543$  $7019$ 

#### Could QML work on real data? <sup>p</sup>*|Tk<sup>|</sup> i*2*T<sup>k</sup>* k*Xk*k*<sup>F</sup> i*2*T<sup>k</sup> k*2[*K*] In this section we provide a novel quantum classification algorithm, called Quantum Frobenius We will estimate *Fk*(*x*(0)) efficiently using the algorithm below. From our QRAM construction  $\mathsf{D}\mathsf{f}\mathsf{K}$  on real data for running time is logarithmic in the dimension and number of  $\mathsf{F}\mathsf{K}$

#### Frobenius Distance Classification <sup>p</sup>*N<sup>k</sup>* ⇣ **Frobenius Distance Classification Frobenius Distance Classification**

*i*2*T<sup>k</sup>* QFE 4 Frobenius Distance Estimator

 $\overline{\text{Required}}$ 

9: Output *<sup>s</sup>*

**QRAM** access to the matrix  $X_k$  of cluster  $k$  and to a test vector  $x(0)$ . Error parameter  $\eta \geq 0$ . *P*<sub>*I***</sub>** *I*<sup>*/*</sup>*i***<sub>***i***</sub>/<b>***n*<sup>*/*</sup>*ij*<sup>*/i*</sup>/*z*(*i*<sup>*/Q*)</sub>*ij*<sub>*z*</sub>(*x*(*Q*)*i*)*zx*(*x*(*d*)*i*)*zx*(*x*(*x*)*i*)*zx*(*x*(*x*)*i*)*zx*(*x*(*x*))*i*)*zx*(*x*)*ii*)*zx*(*x*)*iii*)*xx*(*x*)*ii</sub></sup>* Ensure: *<sup>X</sup>*(0) <sup>2</sup> <sup>R</sup>*|Tk|*⇥*<sup>d</sup>* which just repeats the row *<sup>x</sup>*(0) *<sup>|</sup>Tk<sup>|</sup>* times. Then, we define  $F_k(x(0)) = \frac{\left\|X_k - X(0)\right\|_F^2}{2\left(\left\|X_k\right\|_F^2 + \left\|X_k(x)\right\|_F^2\right)}$ *F*  $2(\left\|X_k\right\|_F^2 + \left\|X(0)\right\|_F^2)$ 

*i*2*T<sup>k</sup>*

An estimate  $F_k(x(0))$  such that  $|F_k(x(0)) - F_k(x(0))| < \eta$ .

**1.** Create the state  $\mathbf{r}$ :  $\mathbf{r}$ 

$$
\frac{1}{\sqrt{N_k}}\Big( \left|0\right\rangle \sum_{i\in T_k} \left| \left|x(0)\right|\right| \left|i\right\rangle \left|x(0)\right\rangle + \left|1\right\rangle \sum_{i\in T_k} \left| \left|x(i)\right|\right| \left|i\right\rangle \left|x(i)\right\rangle \Big)
$$
\n9 3 1 9 5 8 0 8 9  
\n5 2 6 8 5 8 8 9 9  
\n3 7 0 9 4 8 5 4 3

![](_page_27_Picture_8.jpeg)

#### Could QML work on real data? and QML work on real data? We will estimate *Fk*(*x*(0)) efficiently using the algorithm below. From our QRAM construction  $\mathsf{D}\mathsf{f}\mathsf{K}$  on real data for running time is logarithmic in the dimension and number of  $\mathsf{F}\mathsf{K}$

1

⇣

#### Frobenius Distance Classification <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i X <sup>p</sup>*N<sup>k</sup> |*0i *i*2*T<sup>k</sup>*  $t = t$  norms and to the total number of points and number of points and norm of the cluster (see also Appendix A). **Frobenius Distance Classification**

*i*2*T<sup>k</sup>* 5: Apply the unitary that maps QFE 4 Frobenius Distance Estimator

 $\overline{\text{Required}}$ 

**QRAM** access to the matrix  $X_k$  of cluster k and to a test vector  $x(0)$ . Error parameter  $\eta > 0$ . *<sup>X</sup>*(0) <sup>2</sup> <sup>R</sup>*|Tk|*⇥*<sup>d</sup>* which just repeats the row *<sup>x</sup>*(0) *<sup>|</sup>Tk<sup>|</sup>* times. Then, we define  $F_k(x(0)) = \frac{\left\|X_k - X(0)\right\|_F^2}{2\left(\left\|X_k\right\|_F^2 + \left\|X_k(x)\right\|_F^2\right)}$ 

*i*2*T<sup>k</sup>*

 $\mathcal{L}$ 

we can create a superposition of all vectors in the cluster as  $\alpha$  and  $\alpha$  as  $\alpha$ 

Ensure:

9: Output *<sup>s</sup>*

*P*<sub>*I***</sub>** *I*<sup>*/*</sup>*i***<sub>***i***</sub>/<b>***n*<sup>*/*</sup>*ij*<sup>*/i*</sup>/*z*(*i*<sup>*/Q*)</sub>*ij*<sub>*z*</sub>(*x*(*Q*)*i*)*zx*(*x*(*d*)*i*)*zx*(*x*(*x*)*i*)*zx*(*x*(*x*)*i*)*zx*(*x*(*x*))*i*)*zx*(*x*)*ii*)*zx*(*x*)*iii*)*xx*(*x*)*ii</sub></sup>* An estimate  $F_k(x(0))$  such that  $|F_k(x(0)) - F_k(x(0))| < \eta$ .

**1.** Create the state  $\mathbf{r}$ :  $\mathbf{r}$ 

$$
\frac{1}{\sqrt{N_k}}\Big( \left|0\right\rangle \sum_{i\in T_k} \left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle + \left|1\right\rangle \sum_{i\in T_k} \left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle \Big)
$$
\n9 3 1 9 5 6 0 8 4  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n9 8 8 9 9  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n19 5 6 0 8 4

\n20 8 5 8 8 9 9

\n3 7 0 9 4 8 5 4 3

2. Apply a Hadamard to the first register to the first register to get  $\overline{6}$  Apply a Hadamard to the first register to get :. Apply a Hadamard i

$$
\frac{1}{\sqrt{2N_k}}\left|0\right\rangle\sum_{i\in T_k}\Big( \left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle+\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\Big) + \frac{1}{\sqrt{2N_k}}\left|1\right\rangle\sum_{i\in T_k}\Big( \left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle-\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\Big)\\
$$

 $\mathcal{L}$ 

⌘

*F*

![](_page_28_Picture_11.jpeg)

 $7709 / 8543$  $6470492$ 

1

⇣

#### Frobenius Distance Classification <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i X <sup>p</sup>*N<sup>k</sup> |*0i *i*2*T<sup>k</sup>*  $t = t$  norms and to the total number of points and number of points and norm of the cluster (see also Appendix A). **Frobenius Distance Classification** Classifica@on\$\$ process. The classification algorithm assigns a test point *x*(0) to the cluster *k* whose points have

*i*2*T<sup>k</sup>* 5: Apply the unitary that maps QFE 4 Frobenius Distance Estimator Let *X<sup>k</sup>* be defined as the matrix whose rows are the vectors corresponding to the *k*-th cluster, **between 1980 is the number of electron in the cluster of electron in the cluster. 5 4 7 1 9 5 6 2 / 8** 

 $\overline{\text{Required}}$ 

9: Output *<sup>s</sup>*

**QRAM** access to the matrix  $X_k$  of cluster k and to a test vector  $x(0)$ . Error parameter  $\eta > 0$ . *<sup>X</sup>*(0) <sup>2</sup> <sup>R</sup>*|Tk|*⇥*<sup>d</sup>* which just repeats the row *<sup>x</sup>*(0) *<sup>|</sup>Tk<sup>|</sup>* times. Then, we define  $F_k(x(0)) = \frac{\left\|X_k - X(0)\right\|_F^2}{2\left(\left\|X_k\right\|_F^2 + \left\|X_k(x)\right\|_F^2\right)}$ 

*i*2*T<sup>k</sup>*

 $\mathcal{L}$ 

we can create a superposition of all vectors in the cluster as  $\alpha$  and  $\alpha$  as  $\alpha$ 

*P*<sub>*I***</sub>** *I*<sup>*/*</sup>*i***<sub>***i***</sub>/<b>***n*<sup>*/*</sup>*ij*<sup>*/i*</sup>/*z*(*i*<sup>*/Q*)</sub>*ij*<sub>*z*</sub>(*x*(*Q*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf*<sub>*x*</sub>(*x*)</sub></sup> An estimate  $F_k(x(0))$  such that  $|F_k(x(0)) - F_k(x(0))| < \eta$ . Ensure:

**1.** Create the state  $\mathbf{r}$ :  $\mathbf{r}$ 

$$
\frac{1}{\sqrt{N_k}}\Big( \left|0\right\rangle \sum_{i\in T_k} \left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle + \left|1\right\rangle \sum_{i\in T_k} \left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle \Big)
$$
\n9 3 1 9 5 6 0 8 4  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n9 8 8 9 9  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n19 5 6 0 8 4

\n20 8 5 8 8 9 9

\n3 7 0 9 4 8 5 4 3

2. Apply a Hadamard to the first register to the first register to get  $\overline{6}$  Apply a Hadamard to the first register to get :. Apply a Hadamard i

$$
\frac{1}{\sqrt{2N_k}}\left|0\right\rangle \sum_{i\in T_k}\left(\left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle+\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\right)+\frac{1}{\sqrt{2N_k}}\left|1\right\rangle \sum_{i\in T_k}\left(\left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle-\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\right)
$$

 $\mathcal{L}$ 

⌘

 $2(\left\|X_k\right\|_F^2 + \left\|X(0)\right\|_F^2)$ 

*F*

*,*

it is either stored in  $\mathcal{L}_{\mathbf{A}}$  or comes directly from some quantum some quantum some  $\mathcal{L}_{\mathbf{A}}$ 

which corresponds to the average normalized squared distance between *x*(0) and the cluster *k*. Let

[KL 18]

3

 $7709 / 8543$  $6470492$ 

 $37$ 

3. Repeat and Estimate Prob[ outcome  $1$ ]=F<sub>k</sub>(x(0)) *i*2*T<sup>k</sup>* 7: Measure the first register and if the outcome is *|*1i then s:=s+1 3. Repeat and Estimate Prob[ outcome 1]=F<sub>k</sub>(x(0))

#### Could QML work on real data? and QML work on real data? We will estimate *Fk*(*x*(0)) efficiently using the algorithm below. From our QRAM construction  $\mathsf{D}\mathsf{f}\mathsf{K}$  on real data for running time is logarithmic in the dimension and number of  $\mathsf{F}\mathsf{K}$

1

⇣

#### Frobenius Distance Classification <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i X <sup>p</sup>*N<sup>k</sup> |*0i *i*2*T<sup>k</sup>*  $t = t$  norms and to the total number of points and number of points and norm of the cluster (see also Appendix A). **Frobenius Distance Classification**

*i*2*T<sup>k</sup>* 5: Apply the unitary that maps QFE 4 Frobenius Distance Estimator

 $\overline{\text{Required}}$ 

**QRAM** access to the matrix  $X_k$  of cluster k and to a test vector  $x(0)$ . Error parameter  $\eta > 0$ . *<sup>X</sup>*(0) <sup>2</sup> <sup>R</sup>*|Tk|*⇥*<sup>d</sup>* which just repeats the row *<sup>x</sup>*(0) *<sup>|</sup>Tk<sup>|</sup>* times. Then, we define  $F_k(x(0)) = \frac{\left\|X_k - X(0)\right\|_F^2}{2\left(\left\|X_k\right\|_F^2 + \left\|X_k(x)\right\|_F^2\right)}$ 

*i*2*T<sup>k</sup>*

 $\mathcal{L}$ 

Ensure:

9: Output *<sup>s</sup>*

*P*<sub>*I***</sub>** *I*<sup>*/*</sup>*i***<sub>***i***</sub>/<b>***n*<sup>*/*</sup>*ij*<sup>*/i*</sup>/*z*(*i*<sup>*/Q*)</sub>*ij*<sub>*z*</sub>(*x*(*Q*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf*<sub>*x*</sub>(*x*)</sub></sup> An estimate  $F_k(x(0))$  such that  $|F_k(x(0)) - F_k(x(0))| < \eta$ .

**1.** Create the state  $\mathbf{r}$ :  $\mathbf{r}$ 

$$
\frac{1}{\sqrt{N_k}}\Big( \left|0\right\rangle \sum_{i\in T_k} \left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle + \left|1\right\rangle \sum_{i\in T_k} \left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle \Big)
$$
\n9 3 1 9 5 6 0 8 4  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n9 8 8 9 9  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n19 5 6 0 8 4

\n20 8 5 8 8 9 9

\n3 7 0 9 4 8 5 4 3

2. Apply a Hadamard to the first register to the first register to get  $\overline{6}$  Apply a Hadamard to the first register to get :. Apply a Hadamard i

$$
\frac{1}{\sqrt{2N_k}}\left|0\right\rangle \sum_{i\in T_k}\left(\left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle+\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\right)+\frac{1}{\sqrt{2N_k}}\left|1\right\rangle \sum_{i\in T_k}\left(\left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle-\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\right)
$$

 $\mathcal{L}$ 

⌘

*F*

- 3. Repeat and Estimate Prob[ outcome  $1]=F_k(x(0))$ *i*2*T<sup>k</sup>* 3. Repeat and Estimate Prob[ outcome 1]=F<sub>k</sub>(x(0))
- external and if the closest cluster and if the closest register and if the outcome is  $\frac{1}{\sqrt{1+\epsilon}}$ 9: Output *<sup>s</sup>* ٠Ì <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i

![](_page_30_Picture_13.jpeg)

1

⇣

#### Frobenius Distance Classification <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i X <sup>p</sup>*N<sup>k</sup> |*0i *i*2*T<sup>k</sup>*  $t = t$  norms and to the total number of points and number of points and norm of the cluster (see also Appendix A). **Frobenius Distance Classification** Classifica@on\$\$ process. The classification algorithm assigns a test point *x*(0) to the cluster *k* whose points have

*i*2*T<sup>k</sup>* 5: Apply the unitary that maps QFE 4 Frobenius Distance Estimator Let *X<sup>k</sup>* be defined as the matrix whose rows are the vectors corresponding to the *k*-th cluster, **between 1980 is the number of electron in the cluster of electron in the cluster. 5 4 7 1 9 5 6 2 / 8** 

 $\overline{\text{Required}}$ 

**QRAM** access to the matrix  $X_k$  of cluster k and to a test vector  $x(0)$ . Error parameter  $\eta > 0$ . *<sup>X</sup>*(0) <sup>2</sup> <sup>R</sup>*|Tk|*⇥*<sup>d</sup>* which just repeats the row *<sup>x</sup>*(0) *<sup>|</sup>Tk<sup>|</sup>* times. Then, we define  $F_k(x(0)) = \frac{\left\|X_k - X(0)\right\|_F^2}{2\left(\left\|X_k\right\|_F^2 + \left\|X_k(x)\right\|_F^2\right)}$ 

*i*2*T<sup>k</sup>*

 $\mathcal{L}$ 

we can create a superposition of all vectors in the cluster as  $\alpha$  and  $\alpha$  as  $\alpha$ 

Ensure:

9: Output *<sup>s</sup>*

*P*<sub>*I***</sub>** *I*<sup>*/*</sup>*i***<sub>***i***</sub>/<b>***n*<sup>*/*</sup>*ij*<sup>*/i*</sup>/*z*(*i*<sup>*/Q*)</sub>*ij*<sub>*z*</sub>(*x*(*Q*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf*<sub>*x*</sub>(*x*)</sub></sup> An estimate  $F_k(x(0))$  such that  $|F_k(x(0)) - F_k(x(0))| < \eta$ .

**1.** Create the state  $\mathbf{r}$ :  $\mathbf{r}$ 

$$
\frac{1}{\sqrt{N_k}}\Big( \left|0\right\rangle \sum_{i\in T_k} \left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle + \left|1\right\rangle \sum_{i\in T_k} \left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle \Big)
$$
\n9 3 1 9 5 6 0 8 4  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n9 8 8 9 9  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n19 5 6 0 8 4

\n20 8 5 8 8 9 9

\n3 7 0 9 4 8 5 4 3

2. Apply a Hadamard to the first register to the first register to get  $\overline{6}$  Apply a Hadamard to the first register to get :. Apply a Hadamard i

$$
\frac{1}{\sqrt{2N_k}}\left|0\right\rangle \sum_{i\in T_k}\left(\left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle+\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\right)+\frac{1}{\sqrt{2N_k}}\left|1\right\rangle \sum_{i\in T_k}\left(\left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle-\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\right)
$$

 $\mathcal{L}$ 

⌘

 $2(\left\|X_k\right\|_F^2 + \left\|X(0)\right\|_F^2)$ 

*F*

*,*

it is either stored in  $\mathcal{L}_{\mathbf{A}}$  or comes directly from some quantum some quantum some  $\mathcal{L}_{\mathbf{A}}$ 

which corresponds to the average normalized squared distance between *x*(0) and the cluster *k*. Let

- 3. Repeat and Estimate Prob[ outcome  $1$ ]=F<sub>k</sub>(x(0)) *i*2*T<sup>k</sup>* 3. Repeat and Estimate Prob[ outcome 1]=F<sub>k</sub>(x(0)) | Remark 1
- external and if the closest cluster and if the closest register and if the outcome is  $\frac{1}{\sqrt{1+\epsilon}}$ 9: Output *<sup>s</sup>* ٠Ì <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i

Remark 1:

Classification as easy as creating the states

[KL 18]

 $\omega$ 

 $709!/8543$  $6470692$ 

 $37$ 

1

⇣

#### Frobenius Distance Classification <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i X <sup>p</sup>*N<sup>k</sup> |*0i *i*2*T<sup>k</sup>*  $t = t$  norms and to the total number of points and number of points and norm of the cluster (see also Appendix A). **Frobenius Distance Classification** Classifica@on\$\$ process. The classification algorithm assigns a test point *x*(0) to the cluster *k* whose points have

*i*2*T<sup>k</sup>* 5: Apply the unitary that maps QFE 4 Frobenius Distance Estimator Let *X<sup>k</sup>* be defined as the matrix whose rows are the vectors corresponding to the *k*-th cluster, **between 1980 is the number of electron in the cluster of electron in the cluster. 5 4 7 1 9 5 6 2 / 8** 

 $\overline{\text{Required}}$ 

9: Output *<sup>s</sup>*

**QRAM** access to the matrix  $X_k$  of cluster k and to a test vector  $x(0)$ . Error parameter  $\eta > 0$ . *<sup>X</sup>*(0) <sup>2</sup> <sup>R</sup>*|Tk|*⇥*<sup>d</sup>* which just repeats the row *<sup>x</sup>*(0) *<sup>|</sup>Tk<sup>|</sup>* times. Then, we define  $F_k(x(0)) = \frac{\left\|X_k - X(0)\right\|_F^2}{2\left(\left\|X_k\right\|_F^2 + \left\|X_k(x)\right\|_F^2\right)}$ 

*i*2*T<sup>k</sup>*

 $\mathcal{L}$ 

we can create a superposition of all vectors in the cluster as  $\alpha$  and  $\alpha$  as  $\alpha$ 

*P*<sub>*I***</sub>** *I*<sup>*/*</sup>*i***<sub>***i***</sub>/<b>***n*<sup>*/*</sup>*ij*<sup>*/i*</sup>/*z*(*i*<sup>*/Q*)</sub>*ij*<sub>*z*</sub>(*x*(*Q*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf*<sub>*x*</sub>(*x*)</sub></sup> An estimate  $F_k(x(0))$  such that  $|F_k(x(0)) - F_k(x(0))| < \eta$ . Ensure:

1. Create the state  $1<sub>z</sub>$ 

$$
\frac{1}{\sqrt{N_k}}\Big( \left|0\right\rangle \sum_{i\in T_k} \left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle + \left|1\right\rangle \sum_{i\in T_k} \left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle \Big)
$$
\n9 3 1 9 5 6 0 8 4  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n9 8 8 9 9  
\n $\mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S} \mathcal{S}$ \n19 5 6 0 8 4

\n20 8 5 8 8 9 9

\n3 7 0 9 4 8 5 4 3

2. Apply a Hadamard to the first register to the first register to get  $\overline{6}$  Apply a Hadamard to the first register to get :. Apply a Hadamard i

$$
\frac{1}{\sqrt{2N_k}}\left|0\right\rangle \sum_{i\in T_k}\left(\left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle+\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\right)+\frac{1}{\sqrt{2N_k}}\left|1\right\rangle \sum_{i\in T_k}\left(\left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle-\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle\right)
$$

 $\mathcal{L}$ 

⌘

 $2(\left\|X_k\right\|_F^2 + \left\|X(0)\right\|_F^2)$ 

*F*

*,*

it is either stored in  $\mathcal{L}_{\mathbf{A}}$  or comes directly from some quantum some quantum some  $\mathcal{L}_{\mathbf{A}}$ 

which corresponds to the average normalized squared distance between *x*(0) and the cluster *k*. Let

- 3. Repeat and Estimate Prob[ outcome  $1]=F_k(x(0))$ *i*2*T<sup>k</sup>* 3. Repeat and Estimate Prob[ outcome 1]=F<sub>k</sub>(x(0)) | Remark 2
- external and if the closest cluster and if the closest register and if the outcome is  $\frac{1}{\sqrt{1+\epsilon}}$ 9: Output *<sup>s</sup>* ٠Ì <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i

Remark 2:

Comparable accuracy to classical classifiers

[KL 18]

 $\omega$ 

 $7Q9 \neq 8543$  $647069$ 

 $37$ 

1

⇣

#### Frobenius Distance Classification <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i X <sup>p</sup>*N<sup>k</sup> |*0i *i*2*T<sup>k</sup>*  $t = t$  norms and to the total number of points and number of points and norm of the cluster (see also Appendix A). **Frobenius Distance Classification** Classifica@on\$\$ process. The classification algorithm assigns a test point *x*(0) to the cluster *k* whose points have

*i*2*T<sup>k</sup>* 5: Apply the unitary that maps QFE 4 Frobenius Distance Estimator Let *X<sup>k</sup>* be defined as the matrix whose rows are the vectors corresponding to the *k*-th cluster,

 $\overline{\text{Required}}$ 

9: Output *<sup>s</sup>*

**QRAM** access to the matrix  $X_k$  of cluster k and to a test vector  $x(0)$ . Error parameter  $\eta > 0$ . *<sup>X</sup>*(0) <sup>2</sup> <sup>R</sup>*|Tk|*⇥*<sup>d</sup>* which just repeats the row *<sup>x</sup>*(0) *<sup>|</sup>Tk<sup>|</sup>* times. Then, we define  $F_k(x(0)) = \frac{\left\|X_k - X(0)\right\|_F^2}{2\left(\left\|X_k\right\|_F^2 + \left\|X_k(x)\right\|_F^2\right)}$ 

*i*2*T<sup>k</sup>*

 $\mathcal{L}$ 

we can create a superposition of all vectors in the cluster as  $\alpha$  and  $\alpha$  as  $\alpha$ 

*P*<sub>*I***</sub>** *I*<sup>*/*</sup>*i***<sub>***i***</sub>/<b>***n*<sup>*/*</sup>*ij*<sup>*/i*</sup>/*z*(*i*<sup>*/Q*)</sub>*ij*<sub>*z*</sub>(*x*(*Q*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf<sub><i>x*</sub>(*x*(*d*))*i*<sub>*z*</sub> *zf*<sub>*x*</sub>(*x*)</sub></sup> An estimate  $F_k(x(0))$  such that  $|F_k(x(0)) - F_k(x(0))| < \eta$ . Ensure:

1. Create the state  $1<sub>z</sub>$ 

$$
\frac{1}{\sqrt{N_k}}\Big( \left|0\right\rangle \sum_{i\in T_k}\left\|x(0)\right\|\left|i\right\rangle\left|x(0)\right\rangle +\left|1\right\rangle \sum_{i\in T_k}\left\|x(i)\right\|\left|i\right\rangle\left|x(i)\right\rangle \Big)
$$

2. Apply a Hadamard to the first register to the first register to get  $\overline{6}$  Apply a Hadamard to the first register to get :. Apply a Hadamard i

$$
\frac{1}{\sqrt{2N_k}}\left|0\right>\sum_{i\in T_k}\Big(\left\|x(0)\right\|\left|i\right>\left|x(0)\right>+ \left\|x(i)\right\|\left|i\right>\left|x(i)\right>\Big)+\frac{1}{\sqrt{2N_k}}\left|1\right>\sum_{i\in T_k}\Big(\left\|x(0)\right\|\left|i\right>\left|x(0)\right>- \left\|x(i)\right\|\left|i\right>\left|x(i)\right>\Big)\Big)
$$

 $\mathcal{L}$ 

⌘

 $2(\left\|X_k\right\|_F^2 + \left\|X(0)\right\|_F^2)$ 

*F*

*,*

 $\frac{3}{2}$ 

it is either stored in  $\mathcal{L}_{\mathbf{A}}$  or comes directly from some quantum some quantum some  $\mathcal{L}_{\mathbf{A}}$ 

which corresponds to the average normalized squared distance between *x*(0) and the cluster *k*. Let

- 3. Repeat and Estimate Prob[ outcome  $1]=F_k(x(0))$ *i*2*T<sup>k</sup>* 3. Repeat and Estimate Prob[ outcome 1]=F<sub>k</sub>(x(0)) | Remark 2
- external and if the closest cluster and if the closest register and if the outcome is  $\frac{1}{\sqrt{1+\epsilon}}$ 9: Output *<sup>s</sup>* ٠Ì <sup>p</sup>*N<sup>k</sup>* ⇣ *|*0i X k*x*(0)k *|i*i + *|*1i

Remark 2:

Comparable accuracy to classical classifiers

Accuracy

[KL 18]

and *|Tk|* is the number of elements in the cluster. For the test point *x*(0), define the matrix

67.5%\$

 $7709 / 8$ 

Dimensionality Reduction: Slow Feature Analysis Classification

![](_page_34_Picture_2.jpeg)

#### **Dimensionality Reduction: Slow Feature Analysis** *A* is usually approximated with a small fraction of all the possible derivatives, roughly linear (and derivative matrix to be just double the number of data points without compromising the accuracy.

 $\ddotsc$   $\ddotsc$   $\ddotsc$   $\ddotsc$ SFA - Algorithm 1 (Classical) Slow Feature Analysis

#### Require:

Input  $X \in \mathbb{R}^{n \times d}$  (normalized and polynomially expanded), and  $K < d \in \mathbb{N}$ Ensure:

 $\dot{Y} = ZW$ , where  $Z = AB^{-2}$  is the whitehed input signal, and  $W \in \mathbb{R}^{m \times m}$ <br>eigenvectors of the matrix  $A = \dot{Z}^T \dot{Z}$  corresponding to the smallest eigenvalues *Y* = *ZW*, where  $Z = XB^{-1/2}$  is the whitened input signal, and  $W \in \mathbb{R}^{d \times (K-1)}$  are the  $K-1$ 

This will include the data will infact allow us to which a quantum procedure. In procedure, the matrix  $\mathcal{L}$ 

1: Whiten the signal:  $Z := XB^{-1/2}$ , and create  $\dot{Z}$  from  $Z$ .

2: Perform PCA on the derivative covariance matrix  $A = \dot{Z}^T \dot{Z}$  of the whitened data.

3: Return  $Y = ZW$ , the projection of whitened data onto *W*, the  $K-1$  slowest eigenvectors of *A* 

![](_page_35_Picture_9.jpeg)

 $\mathbf{r}$ 5 6  $0661$ 3 3 9  $59$ 8365  $9158084$  $\overline{\mathbf{3}}$  $5626858899$  $3770948543$  $\bm{O}$ 

#### Could QML work on real data? This will include the data will infact allow us to which a quantum procedure. In procedure, the matrix  $\mathcal{L}$ Classification *A* is usually approximated with a small fraction of all the possible derivatives, roughly linear (and **Dimensionality Reduction: Slow Feature Analysis** [KL 18] derivative matrix to be just double the number of data points without compromising the accuracy. 562  $SFA - Algorithm$ Dist 2 9  $0664$  $\mathcal{O}$ 5  $\overline{\phantom{a}}$ Require:  $637$ 3  $\text{Input } X \in \mathbb{R}^{n}$  **Feat Vec 1 Feat Vec 2 Feat Vec 3** ৩ Ensure: *<sup>Y</sup>* <sup>=</sup> *ZW*, where *<sup>Z</sup>* <sup>=</sup> *XB*1*/*<sup>2</sup> is the whitened input signal, and *<sup>W</sup>* <sup>2</sup> <sup>R</sup>*<sup>d</sup>*⇥(*K*1) are the *<sup>K</sup>* <sup>1</sup> 3  $\dot{x} = ZW$ , where the matrix  $\dot{x} = \dot{z}W$  and  $\dot{z} = \dot{z}W$  corresponding to the smallest eigenvectors of  $\mathbf Q$  $\mathcal{S}$ 3 5 6  $9158$  $\overline{\mathcal{E}}$  $\mathcal{O}$ ىتى 1: Whiten the sig  $56$  $6858899$  $\mathbf{2}$  $\mathbf{a}$   $\rightarrow$   $\mathbf{b}$ 2: Perform PCA **on the derivative covariance matrix** *A*  $\overline{Z}$ **<sup>***Z***</sup><b>***ZZZZZ***<sup>***Z***</sup><b>***ZZZZ***</del>***ZZ***</del>**  $7709!/8543$ 3: Return  $Y = Z$  **W**, the projection of  $A$  $64706$ ∙ \$\$\$\$ 3 Quantum algorithms for machine learning

#### **Dimensionality Reduction: Slow Feature Analysis** *A* is usually approximated with a small fraction of all the possible derivatives, roughly linear (and derivative matrix to be just double the number of data points without compromising the accuracy.

 $\ddotsc$   $\ddotsc$   $\ddotsc$   $\ddotsc$ SFA - Algorithm 1 (Classical) Slow Feature Analysis

#### Require:

Input  $X \in \mathbb{R}^{n \times d}$  (normalized and polynomially expanded), and  $K < d \in \mathbb{N}$ Ensure:

 $\dot{Y} = ZW$ , where  $Z = AB^{-2}$  is the whitehed input signal, and  $W \in \mathbb{R}^{m \times m}$ <br>eigenvectors of the matrix  $A = \dot{Z}^T \dot{Z}$  corresponding to the smallest eigenvalues *Y* = *ZW*, where  $Z = XB^{-1/2}$  is the whitened input signal, and  $W \in \mathbb{R}^{d \times (K-1)}$  are the  $K-1$ 

This will include the data will include the data with a quantum procedure. In procedure, the matrix  $\mathcal{L}$ 

1: Whiten the signal:  $Z := XB^{-1/2}$ , and create  $\dot{Z}$  from  $Z$ .

2: Perform PCA on the derivative covariance matrix  $A = \dot{Z}^T \dot{Z}$  of the whitened data.

3: Return  $Y = ZW$ , the projection of whitened data onto *W*, the  $K-1$  slowest eigenvectors of *A* 

![](_page_37_Picture_9.jpeg)

 $\mathbf{r}$ 5 6  $0661$ 3 3 9  $59$ 8365  $9158084$  $\overline{\mathbf{3}}$  $5626858899$  $3770948543$  $\bm{O}$ 

#### **Dimensionality Reduction: Slow Feature Analysis** *A* is usually approximated with a small fraction of all the possible derivatives, roughly linear (and derivative matrix to be just double the number of data points without compromising the accuracy.

 $\ddotsc$   $\ddotsc$   $\ddotsc$   $\ddotsc$ SFA - Algorithm 1 (Classical) Slow Feature Analysis

#### Require:

Input  $X \in \mathbb{R}^{n \times d}$  (normalized and polynomially expanded), and  $K < d \in \mathbb{N}$ Ensure:

 $\dot{Y} = ZW$ , where  $Z = AB^{-2}$  is the whitehed input signal, and  $W \in \mathbb{R}^{m \times m}$ <br>eigenvectors of the matrix  $A = \dot{Z}^T \dot{Z}$  corresponding to the smallest eigenvalues  $Y = ZW$ , where  $Z = XB^{-1/2}$  is the whitened input signal, and  $W \in \mathbb{R}^{d \times (K-1)}$  are the  $K-1$ 

*<sup>i</sup>*2[*n*] *<sup>x</sup><sup>i</sup> <sup>|</sup>i*i*.*

This will include the data will include the data with a quantum procedure. In procedure, the matrix  $\mathcal{L}$ 

2: Perform PCA on the derivative covariance matrix  $A = \dot{Z}^T \dot{Z}$  of the whitened data. 3: Return  $Y = ZW$ , the projection of whitened data onto *W*, the  $K-1$  slowest eigenvectors of *A* 1: Whiten the signal:  $Z := XB^{-1/2}$ , and create  $\dot{Z}$  from  $Z$ .

## a Guantum Slow Feature Analysis For Machine learning

Definition 1. *The vector state <sup>|</sup>x*<sup>i</sup> *for <sup>x</sup>* <sup>2</sup> <sup>R</sup>*<sup>n</sup> is defined as* <sup>1</sup> *(Matrix Multiplication, Inversion, Projection)* Proposition 1. *(Phase estimation [Kit96]) Let U be a unitary operator, with eigenvectors |v<sup>j</sup>* i Efficient Quantum Linear Algebra

We start by stating some known results that we will use in the following sections.

![](_page_38_Picture_9.jpeg)

[KL 18]

#### **Dimensionality Reduction: Slow Feature Analysis** *A* is usually approximated with a small fraction of all the possible derivatives, roughly linear (and derivative matrix to be just double the number of data points without compromising the accuracy.

 $\ddotsc$   $\ddotsc$   $\ddotsc$   $\ddotsc$ SFA - Algorithm 1 (Classical) Slow Feature Analysis

#### Require:

Input  $X \in \mathbb{R}^{n \times d}$  (normalized and polynomially expanded), and  $K < d \in \mathbb{N}$ Ensure:

 $\dot{Y} = ZW$ , where  $Z = AB^{-2}$  is the whitehed input signal, and  $W \in \mathbb{R}^{m \times m}$ <br>eigenvectors of the matrix  $A = \dot{Z}^T \dot{Z}$  corresponding to the smallest eigenvalues  $Y = ZW$ , where  $Z = XB^{-1/2}$  is the whitened input signal, and  $W \in \mathbb{R}^{d \times (K-1)}$  are the  $K-1$ 

This will include the data will include the data with a quantum procedure. In procedure, the matrix  $\mathcal{L}$ 

2: Perform PCA on the derivative covariance matrix  $A = \dot{Z}^T \dot{Z}$  of the whitened data. 3: Return  $Y = ZW$ , the projection of whitened data onto *W*, the  $K-1$  slowest eigenvectors of *A* 1: Whiten the signal:  $Z := XB^{-1/2}$ , and create  $\dot{Z}$  from  $Z$ .

Proposition 1. *(Phase estimation [Kit96]) Let U be a unitary operator, with eigenvectors |v<sup>j</sup>* i

### a Guantum Slow Feature Analysis For Machine learning

**E**<br>
(Matrix Multiplication, Inversion, Projection) Efficient Quantum Linear Algebra

#### We start by stating some known results that we will use in the following sections. The following sections. Remark:

*<sup>i</sup>*2[*n*] *<sup>x</sup><sup>i</sup> <sup>|</sup>i*i*.* Classification only needs quantum states

![](_page_39_Picture_11.jpeg)

## Quantum Classifier

Input:  $X$ , a new vector  $x(0)$ 

- 1. Do QSFA to quantumly project X and  $x(0)$  to Y and  $y(0)$
- 2. Use Frobenius Distance Classification on Y,  $y(0)$

![](_page_40_Picture_5.jpeg)

## Quantum Classifier

Input:  $X$ , a new vector  $x(0)$ 

- 1. Do QSFA to quantumly project X and  $x(0)$  to Y and  $y(0)$
- 2. Use Frobenius Distance Classification on Y,  $y(0)$

### Accuracy **Accuracy**

 $\frac{3}{5}$  and  $\frac{3}{5}$  a \$\$\$\$ We simulate the quantum procedures including errors in an HPC machine and test it on the 10000 test digits of MNIST for different parameters

![](_page_41_Picture_7.jpeg)

## **Quantum Classifier**

Input:  $X$ , a new vector  $x(0)$ 

- 1. Do QSFA to quantumly project X and  $x(0)$  to Y and  $y(0)$
- 2. Use Frobenius Distance Classification on Y, y(0)

### **Accuracy**

We simulate the quantum procedures including errors in an HPC machine and test it on the 10000 test digits of MNIST for different parameters

## **Running time**

Classical: O(n  $d^2$ )  $\approx$  = 10<sup>13</sup> (1 hour on 6Tb RAM HPC)

![](_page_42_Picture_9.jpeg)

## **Quantum Classifier**

Input:  $X$ , a new vector  $x(0)$ 

- 1. Do QSFA to quantumly project X and  $x(0)$  to Y and  $y(0)$
- 2. Use Frobenius Distance Classification on Y, y(0)

### Accuracy

We simulate the quantum procedures including errors in an HPC machine and test it on the 10000 test digits of MNIST for different parameters

### **Running time**

Classical:  $O(n d^2) \approx 10^{13}$  (1 hour on 6Tb RAM HPC) Quantum: O( $\kappa$ ,  $\mu$ , 1/ $\theta$ , 1/ $\delta$ , 1/ $\eta$ , K, polylog(n, d, 1/ $\epsilon$ ),...)

![](_page_43_Picture_9.jpeg)

## **Quantum Classifier**

Input:  $X$ , a new vector  $x(0)$ 

- 1. Do QSFA to quantumly project X and  $x(0)$  to Y and  $y(0)$
- 2. Use Frobenius Distance Classification on Y, y(0)

### Accuracy

We simulate the quantum procedures including errors in an HPC machine and test it on the 10000 test digits of MNIST for different parameters

### **Running time**

Classical:  $O(n d^2) \approx 10^{13}$  (1 hour on 6Tb RAM HPC) Quantum: Ο(κ, μ, 1/θ, 1/δ, 1/η, Κ, polylog(n, d, 1/ε),...) ~= 10<sup>7</sup>

![](_page_44_Picture_9.jpeg)

### Main question:

Better accuracy by increasing the dimension, keeping efficient time? Quantum time: O( $\kappa$ ,  $\mu$ , 1/ $\theta$ , 1/ $\delta$ , 1/ $\eta$ , K, polylog(n, d, 1/ $\epsilon$ ),...)

![](_page_45_Picture_3.jpeg)

### Main question:

Better accuracy by increasing the dimension, keeping efficient time? Quantum time: Ο( $\kappa$ , μ, 1/θ, 1/δ, 1/η, K, polylog(n, d, 1/ε),...)

![](_page_46_Figure_3.jpeg)

![](_page_46_Figure_4.jpeg)

![](_page_46_Picture_5.jpeg)

### Main question:

Better accuracy by increasing the dimension, keeping efficient time? Quantum time: Ο( $\kappa$ , μ, 1/θ, 1/δ, 1/η, Κ, polylog(n, d, 1/ε),...)

![](_page_47_Figure_3.jpeg)

![](_page_47_Figure_4.jpeg)

![](_page_47_Picture_5.jpeg)

### Main question:

Better accuracy by increasing the dimension, keeping efficient time? Quantum time: O( $\kappa$ ,  $\mu$ , 1/ $\theta$ , 1/ $\delta$ , 1/ $\eta$ , K, polylog(n, d, 1/ $\epsilon$ ),...)

## Hope (and some evidence):

Quantum classification algorithms can handle bigger dimensions (hence be more accurate), since their running time scales much more favourably with the dimension.

![](_page_48_Picture_5.jpeg)

## **Unsupervised Classification: Q-means [KLLP 18]**

![](_page_49_Picture_1.jpeg)

#### K-means

Input: M N-dimensional points, K clusters

Start with some random points as centroids  $1.$ 

Repeat until convergence

- 2. For each point compute distances to the centroids and assign to closest cluster O(KMN)
- 3. Recompute the centroids O(MN)

## Unsupervised Classification: Q-means [KLLP 18]

![](_page_50_Picture_1.jpeg)

Input: M N-dimensional points, K clusters

 $\frac{3}{5}$  with some random points a 1. Start with some random points as centroids

Repeat until convergence

- 2. For each point compute distances to the centroids and assign to closest cluster O(KMN)
- 3. Recompute the centroids O(MN)

## Q-means

 $\frac{1}{2}$   $\frac{1}{2}$  Input: M N-dimensional points with quantum access, K clusters

1. Start with some random points as centroids

Repeat until convergence

- 2. For all points in superposition compute distances to centroids and assign to closest cluster  $O(K \log(MN))$
- 2. Use Matrix Multiplication and tomography to recompute the centroids  $O(KN \log(MN))$

## Summary and open questions

### Summary

QML is (one of) the best reason to build quantum computers

- Use case: Quantum recommendation systems
- General Methods: Quantum gradient descent for linear gradients
- Benchmarking: Classification of MNIST dataset
- ML data has some hidden structure (e.g. low rank approximations)
- ML is very robust to errors

### **Open Questions**

Build quantum computers and QRAMs

Find new quantum methods (Interior point methods, fully quantum methods,...)

Find more real-world applications

Benchmark hardware via applications