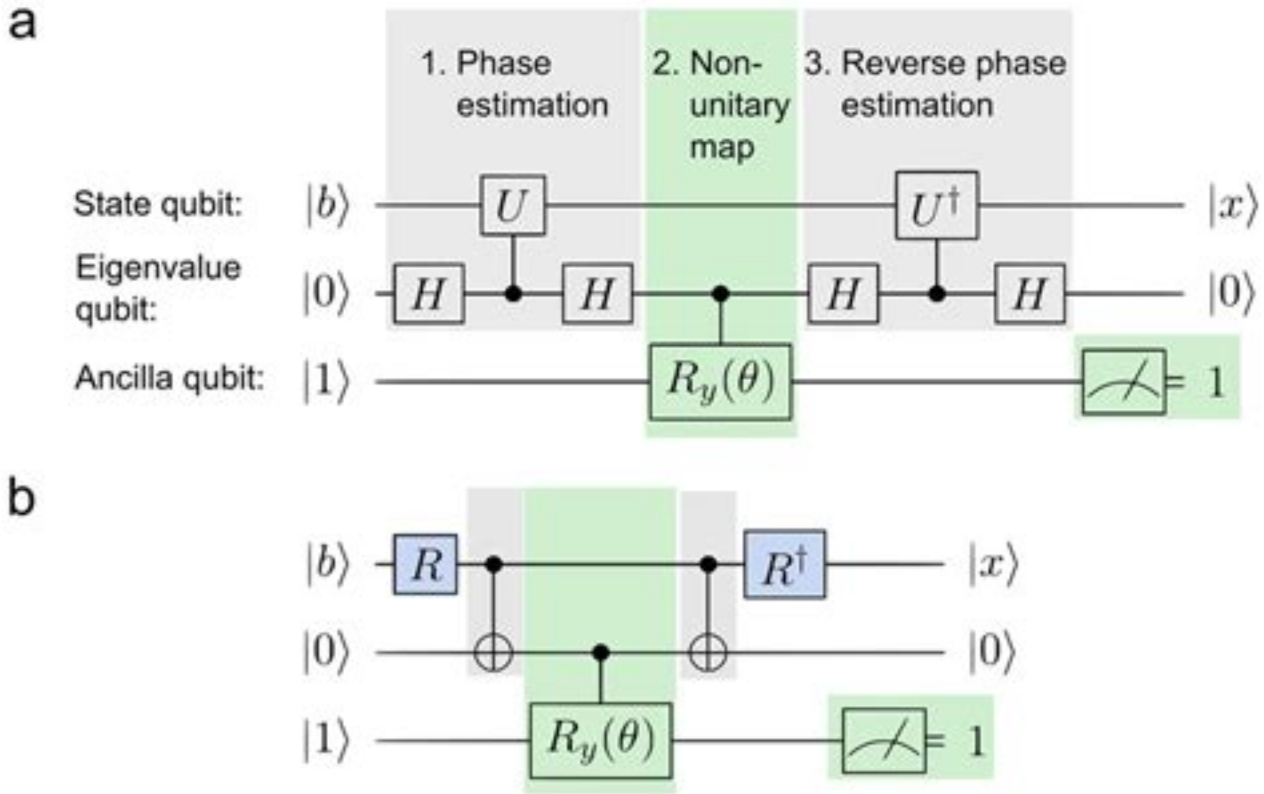
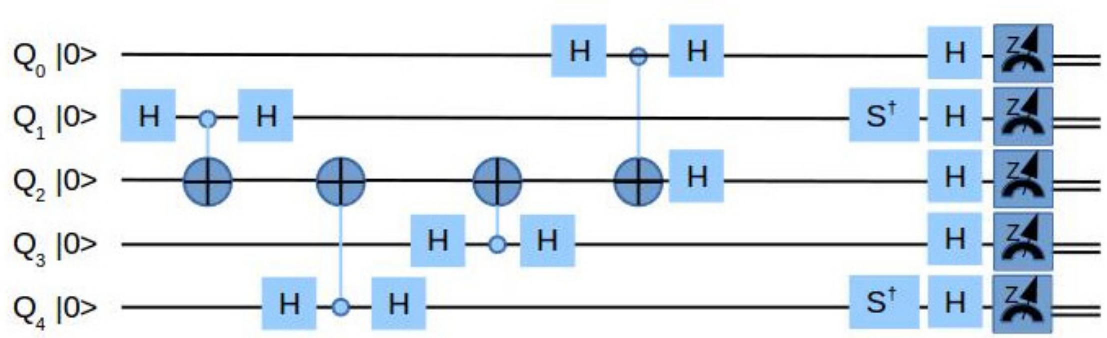
Quantum algorithms

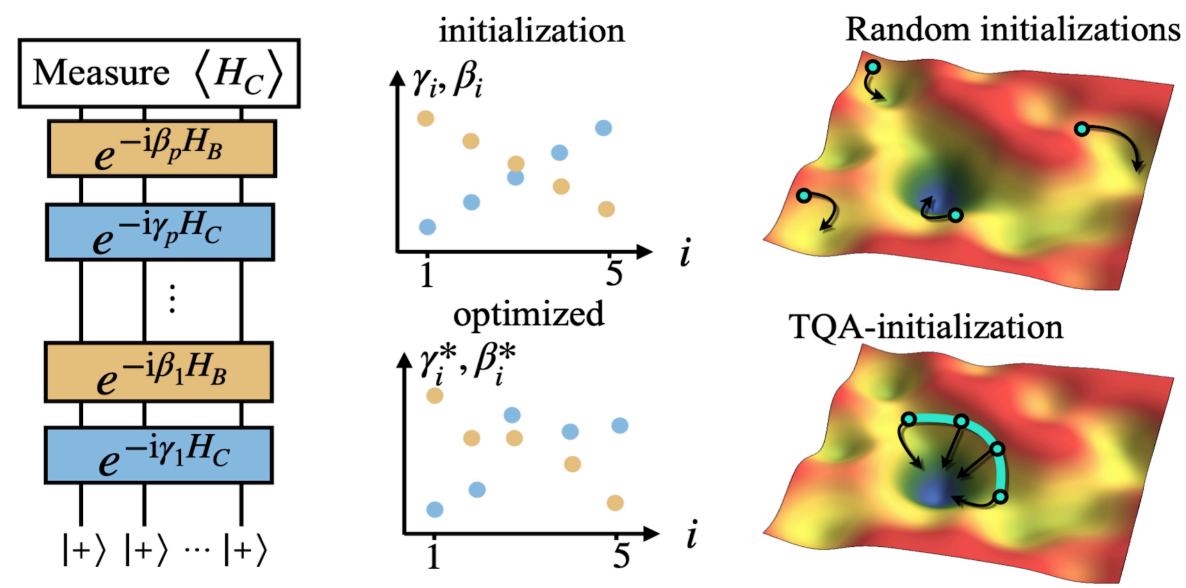
0	1	0	⋯	0
x_1	x_2	x_3		x_n

- Find if there exists i for which $x_i=1$.
- Queries: input i, output x_i.
- Classically, n queries.
- Quantum, O(√n) queries [Grover, 1996].
- Speeds up exhaustive search.



Algorithms





Different Quantum Advantages/Speedups

1. **A provable quantum speedup**: (gold standard)
   requires a proof that there can be no classical algorithm that performs as well or better than the quantum algorithm. (grover's algorithm)

2. **A strong quantum speedup**:
   compares the quantum algorithm with the best known classical algorithm. (shore's algorithm)

3. **Common quantum speedup**:
   relaxes the 'best classical algorithm' to the 'best available classical algorithm'

4. **Potential quantum speedup:**
   compares two specific algorithms and relating the speedup to this instance only

5. **Limited quantum speedup:**
   compares conceptually equivalent algorithms

The struggle for quantum speedup in machine learning

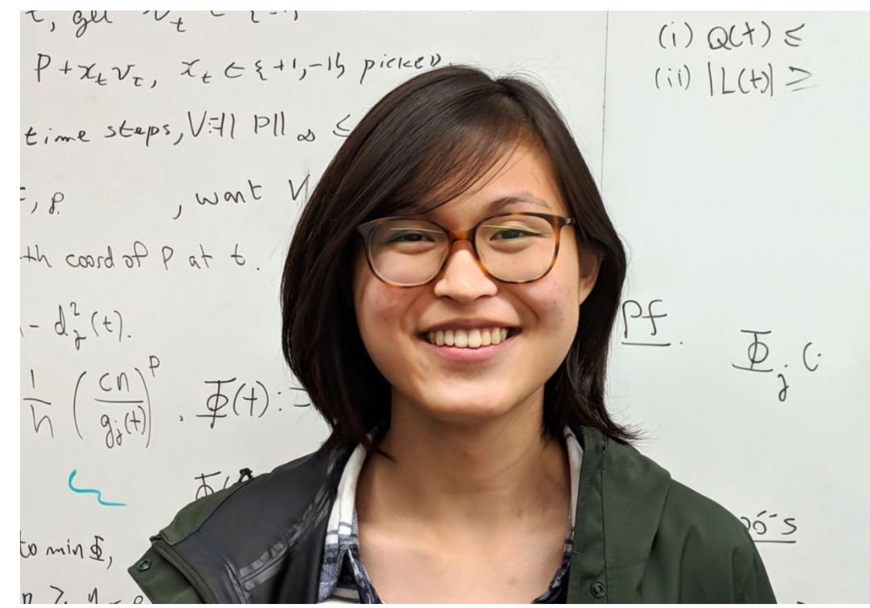
An example:

Iordanis Kerenidis and Anupam Prakash published "Quantum recommendation systems" [1603.08675], in Innovations in Theoretical Computer Science (2017), a QML algorithm claiming exponential quantum speedup over classical algorithms

First genuine real-world application for QML with advantage

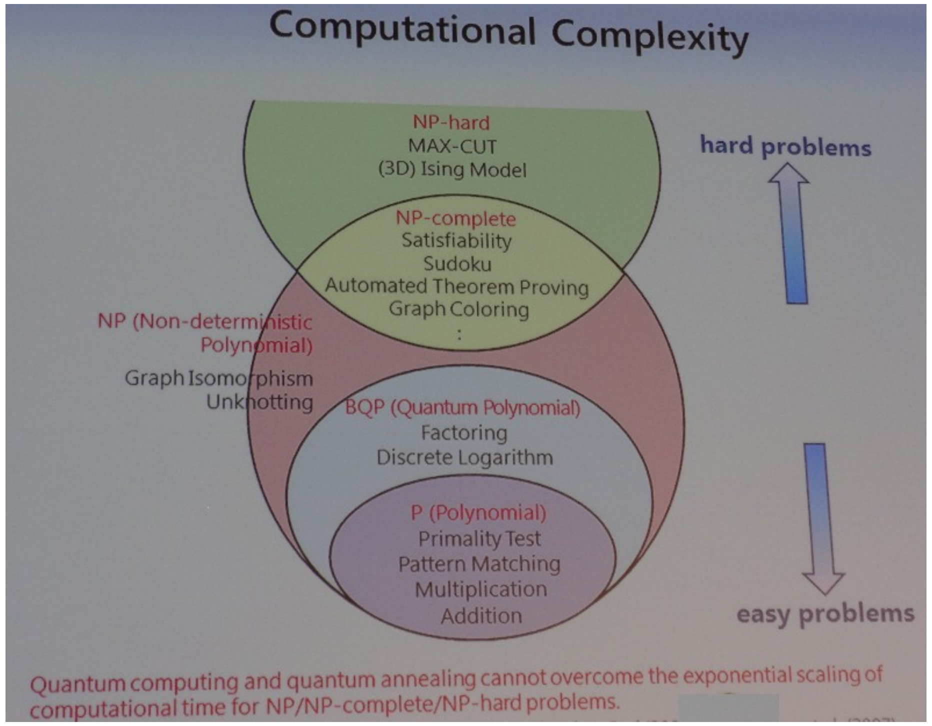
Ewing Tang, 18-year old undergrad at UT Austin debunked this claim [1807.04271]

For NNs, being highly flexible objects, and with the lack of a fully fletched mathematical algorithm describing the evolution of the network output etc, difficult to make definite statements



Have a look at https://www.quantamagazine.org/teenager-finds-classical-alternative-to-quantum-recommendation-algorithm-20180731/

Some complexity theory



P - solvable, deterministically in polynomial time $t = O(L^a)$

(considered 'easy' or 'efficiently solvable problems)

NP - non-deterministic polynomial. Solutions verifiable in polynomial time

NP-hard - hardest problem in NP class

e.g. graph colouring problem is NP-hard

NP-complete - in NP and every problem in NP is reducible to it in polynomial time. Thus, if any NP-complete problem can be solved in pol time, any NP problem can be solved in P time. e.g. traveling salesman problem

BPP - Bounded-error Probabilistic Polynomial time. Produces the correct answer with 2/3 probability for all inputs e.g. testing if number is prime with Solovay-Strassen test

BQP - Bounded-error Quantum Polynomial time. Solvable by probabilistic Turing machine in polynomial time. Correct answers with 2/3 prob. e.g. Shor's algorithm

Complexity classes of machine learning tasks

Training Complexity:

Training a neural network is considered NP-hard. As the task is to find the minimum of a non-convex optimisation problem

Prediction Complexity:

Once a model is trained, predictions are efficient (class P)

Model selection and Hyperparameter Tuning:

Considered to be in NP. 'No free lunch theorem'. High-dimensional optimisation

Feature Selection:

Some ML models require to select subset of features for training. This is considered to be NP-hard

The relation between BQP and NP is not known and topic of ongoing research

- General structure of any QC algorithm:

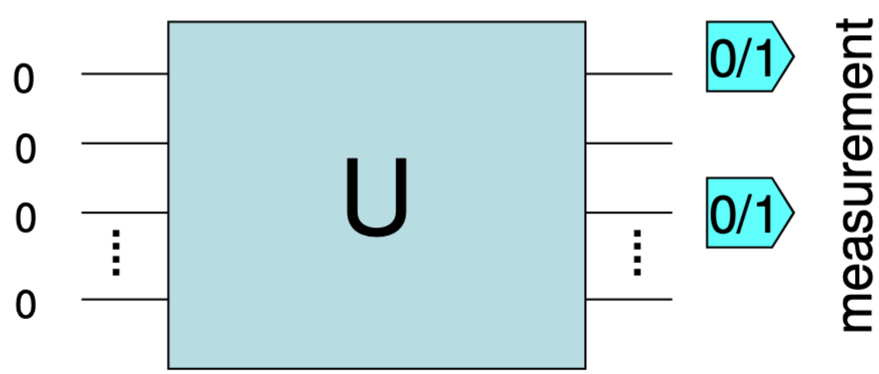
state preparation $x \rightarrow |x\rangle = S_x|0\rangle$

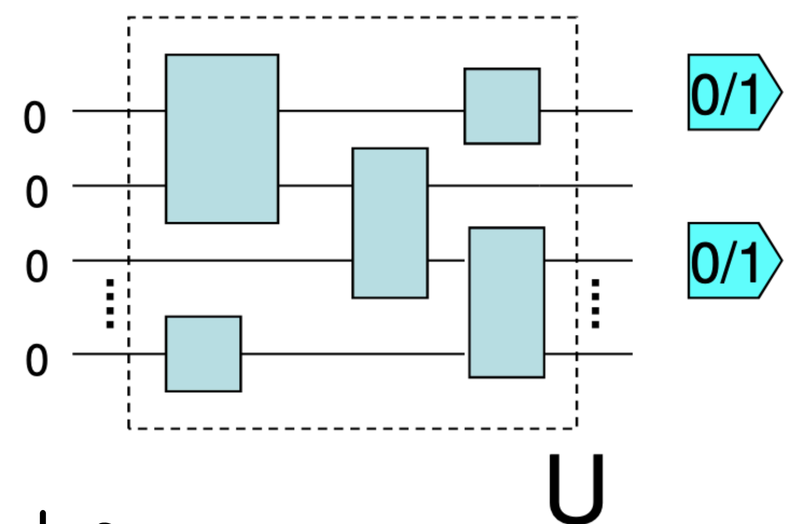
operator acts on Hilbert space states $U|x\rangle = |\Psi_1\rangle$

measurement of observable $\hat{U}$ corresponds to exp. value of operator U $\left\langle \hat{U} \right\rangle_\Psi = \frac{\langle\Psi|U|\Psi\rangle}{\langle\Psi|\Psi\rangle}$

statistical statement need to evaluate often

Need to encode Hilbert space and operator suitable for quantum system





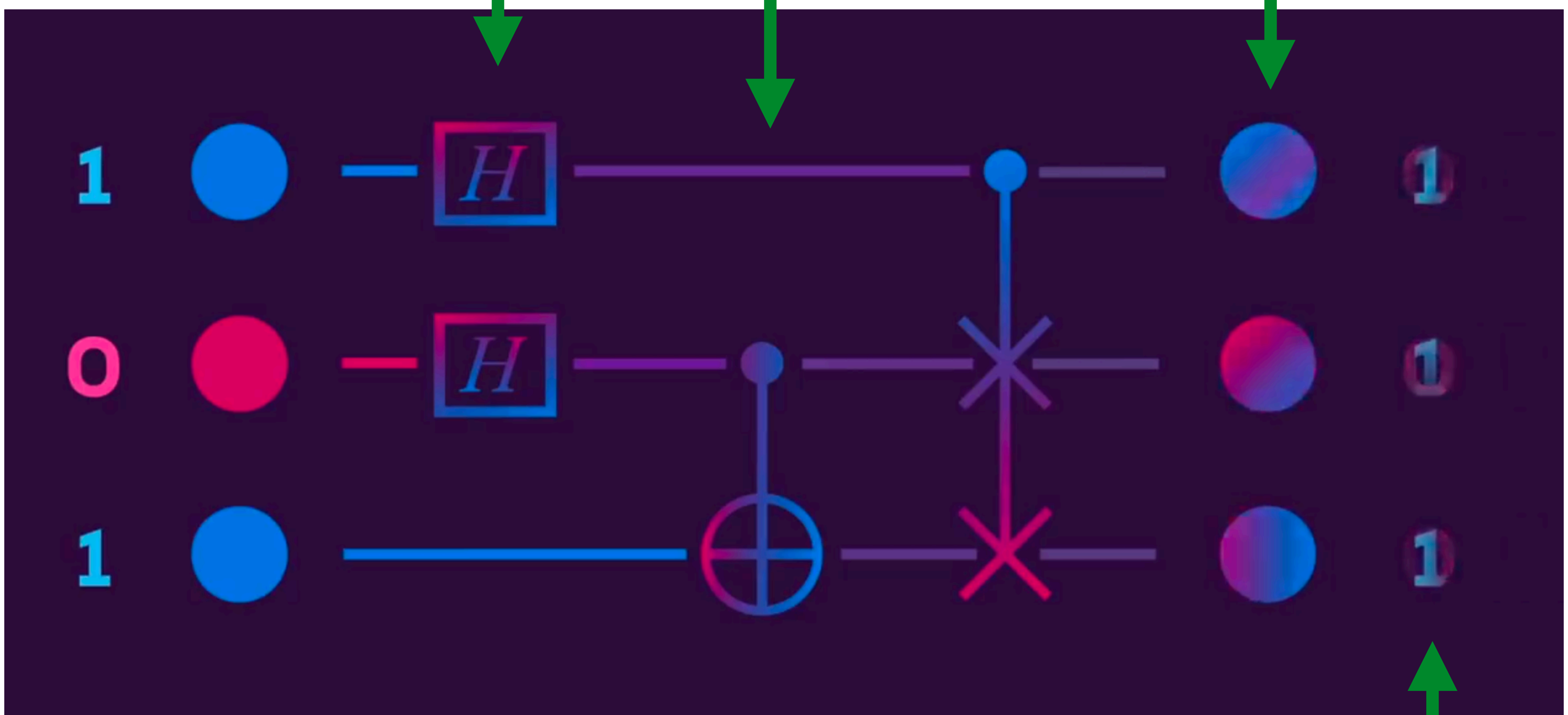
- Operator expressed in terms of individual gates

Quantum Gate

operations on qubits

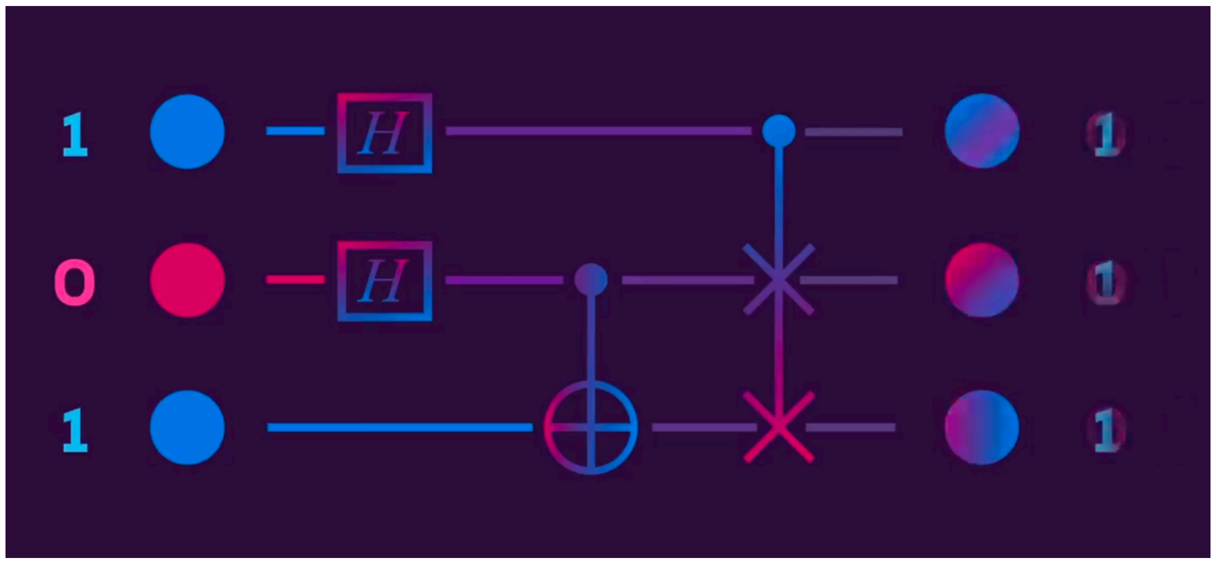
initialisation

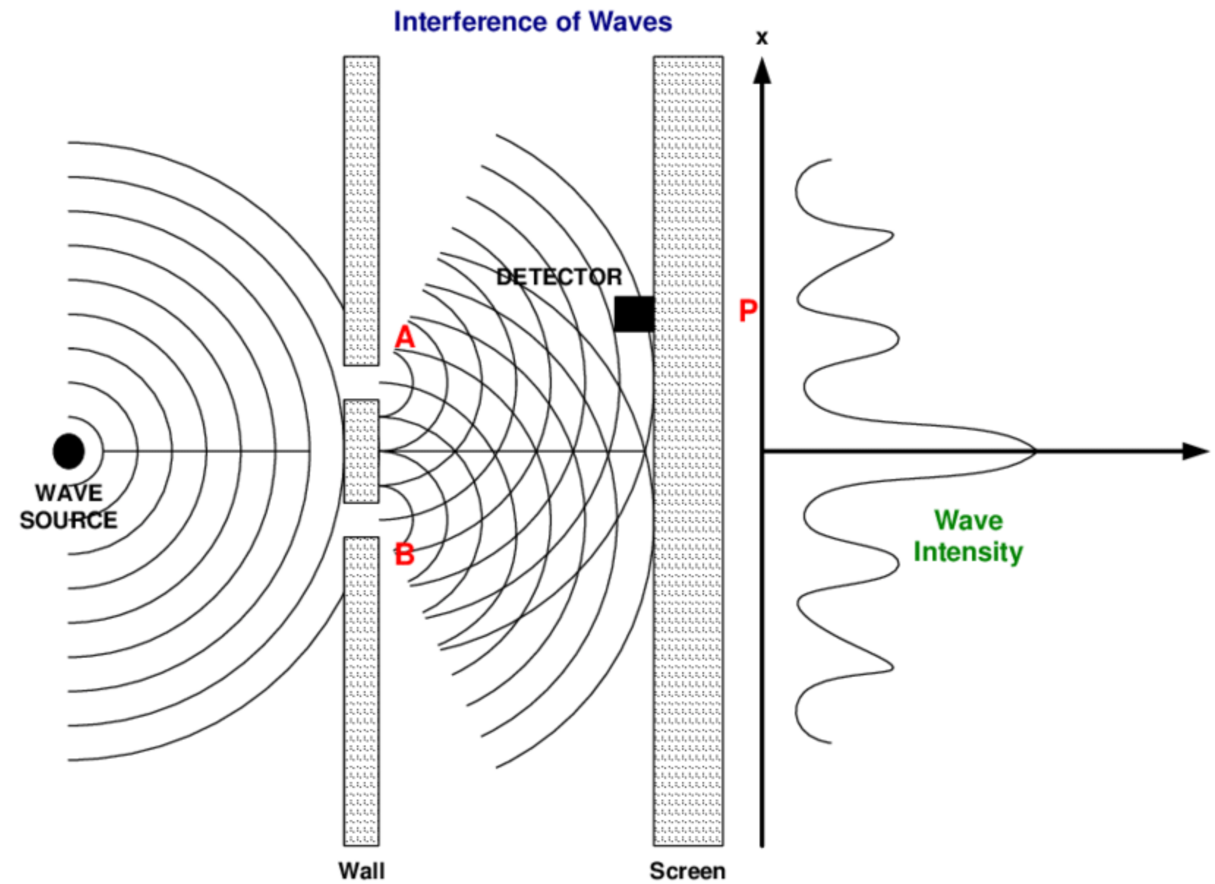
results in output of superpositions



We then measure one specific outcome. Have to repeat measurement to statistically evaluate how likely each outcome is (by calculating and measuring several times). Since we work only with probabilities, we measure only probabilities

Quantum Gate

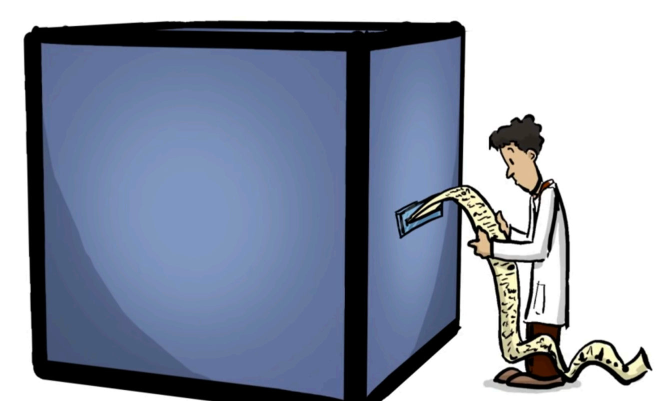




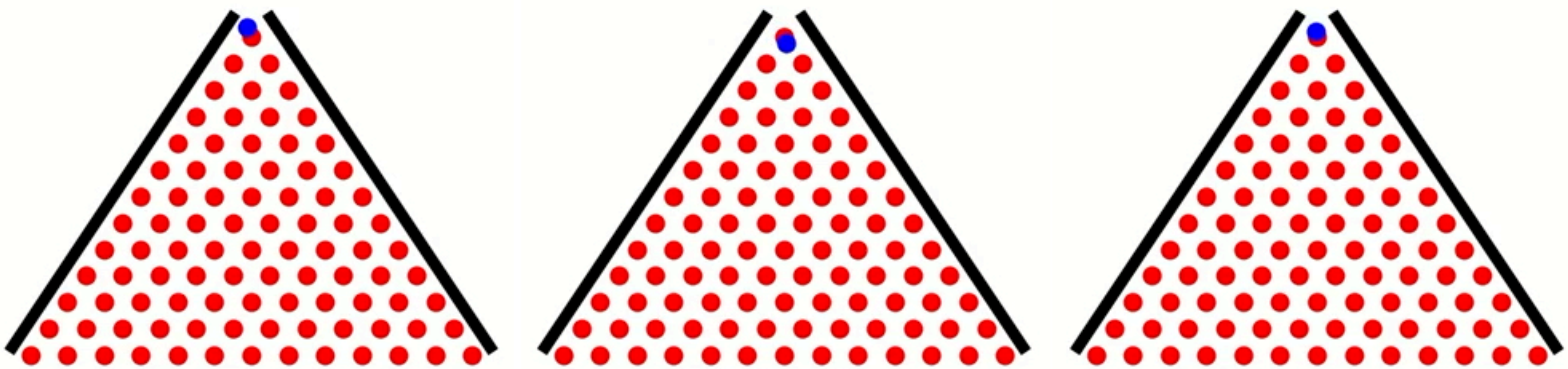
quantum gate and multi slit experiment are conceptually identical

It's a secret computation...

While operating one cannot see how the gate works. Only at the end one can measure the outcome (box is closed during operations)



Galton Board as analogy for Quantum Computer



Algorithm Zoo

Website collecting up to ~200 (until 2018) algorithms showing quantum advantage

https://quantumalgorithmzoo.org

Highlight Quantum Algorithms - used as basis for others

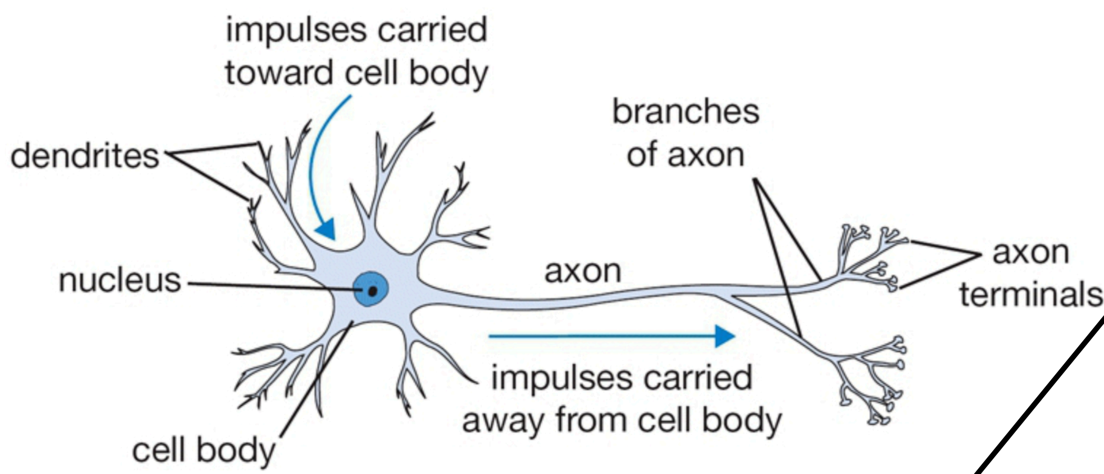
- Quantum Fourier Transformation (QFT)
- Quantum Phase Estimation (QPE)
- HHL (Harrow, Hassidim, Lloyd) algorithm
- (Gaussian) boson sampling via photonic quantum devices

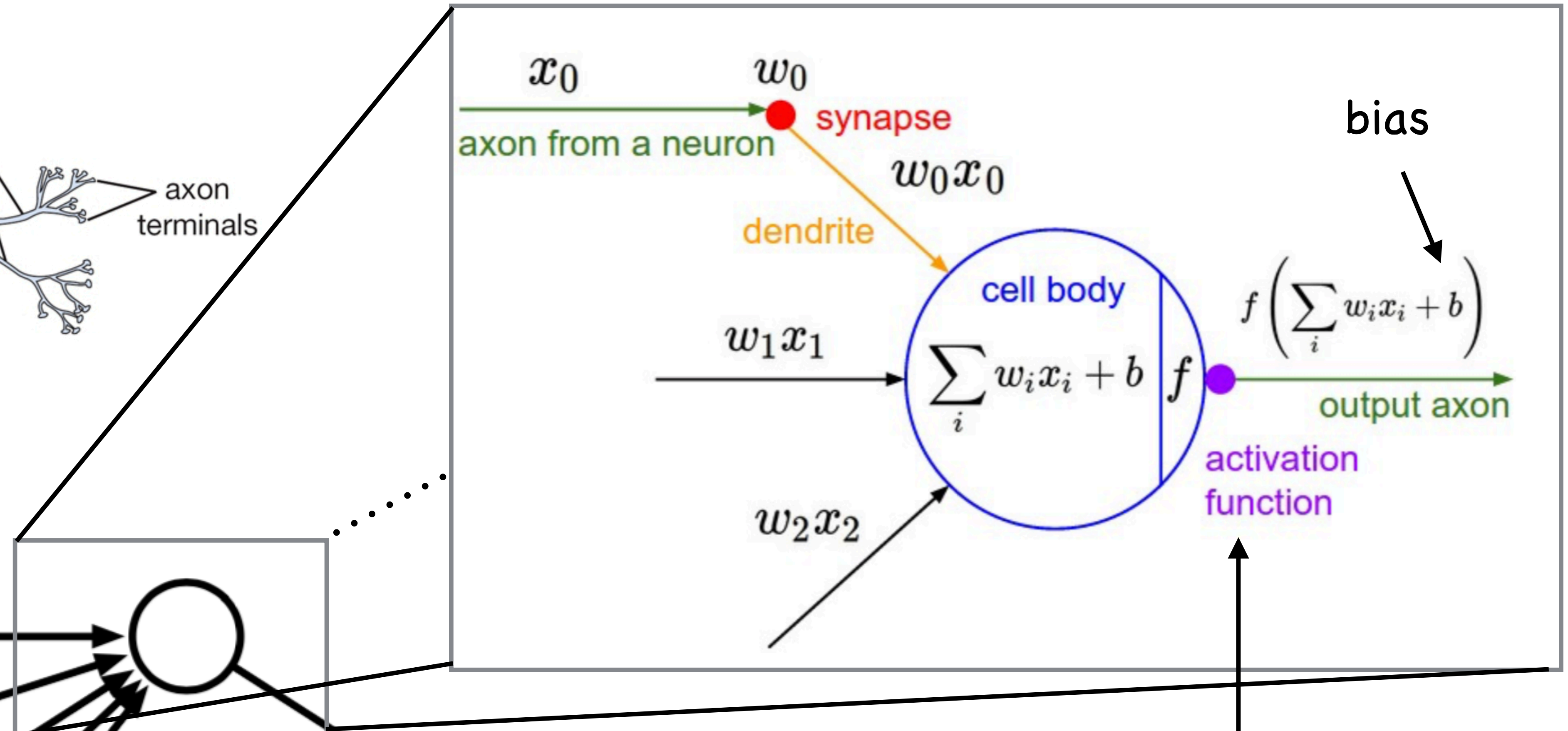
Grover's algorithm, Shore's algorithm, Deutsch algorithm, Quantum Teleportation, ...

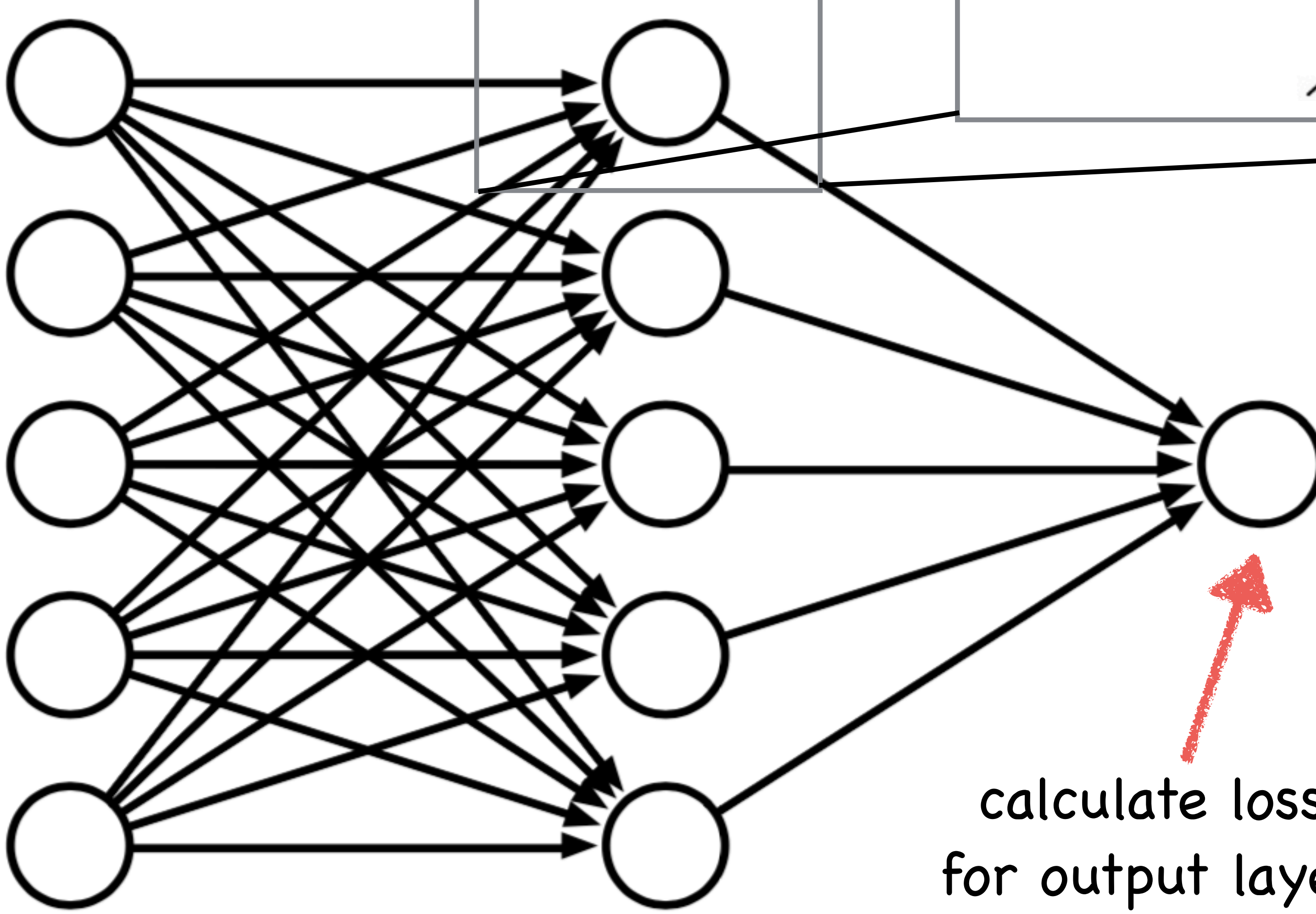
Quantum computing frameworks

Qiskit	IBM	https://qiskit.org/
Cirq	Google Quantum AI	https://quantumai.google/cirq
PENNYLANE	Frontend Ecosystem	https://pennylane.ai/
QIBO	Frontend Ecosystem	https://qibo.science/
D:WAVE	Quantum Annealing	https://www.dwavesys.com/

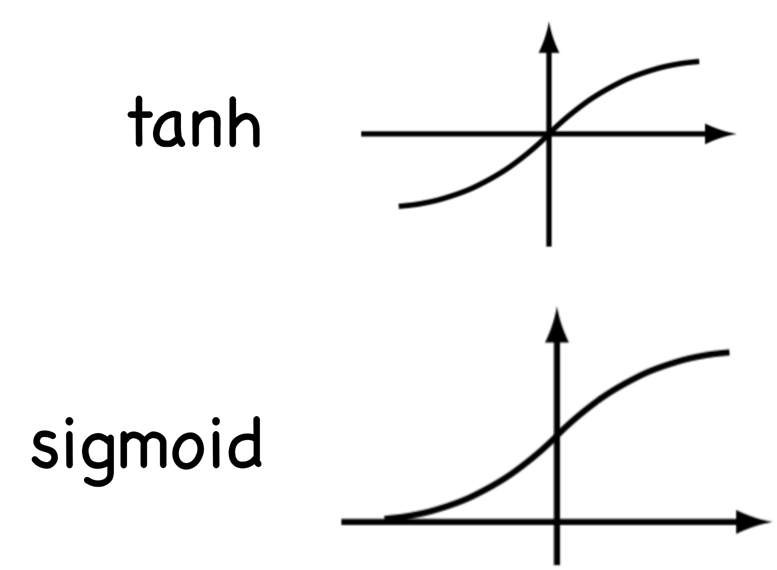
Classical Neural Network recap





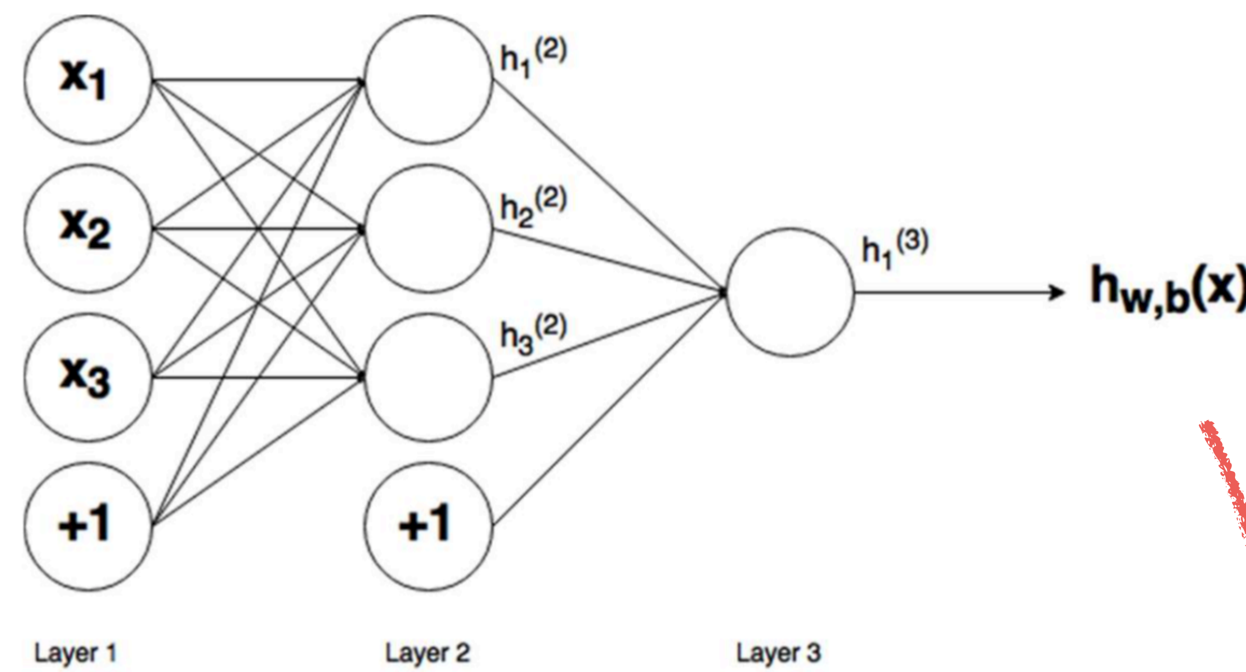


Many options for activation function f, e.g.



putting things together

input hidden output



Notation:

layer l

weight → $w_{ij}^{(l)}$

node number of connection in layer l+1

node number of connection in layer l

bias of layer l is connected to all nodes in layer l+1, thus $b_i^{(l)}$

$b_i^{(l)}$ and $w_{ij}^{(l)}$ have to be calculated during the learning phase of the ANN

The feedforward pass:

activation function

$$
\begin{aligned}
h_1^{(2)} &= f(w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + w_{13}^{(1)} x_3 + b_1^{(1)}) \\
h_2^{(2)} &= f(w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 + w_{23}^{(1)} x_3 + b_2^{(1)}) \\
h_3^{(2)} &= f(w_{31}^{(1)} x_1 + w_{32}^{(1)} x_2 + w_{33}^{(1)} x_3 + b_3^{(1)}) \\
h_{W,b}(x) &= h_1^{(3)} = f(w_{11}^{(2)} h_1^{(2)} + w_{12}^{(2)} h_2^{(2)} + w_{13}^{(2)} h_3^{(2)} + b_1^{(2)})
\end{aligned}
$$

We have managed the first feedforward pass, now we need to evaluate the loss/cost function

- The cost/loss function evaluates the performance of the learning outcome (forward pass) of the ANN, i.e. how well did the NN approximate the training data

NN output of final layer = prediction of NN given input x

$$
\begin{aligned}
J(w,b,x,y) &= \frac{1}{2} \parallel y^z - h^{(n_l)}(x^z) \parallel^2 \\
&= \frac{1}{2} \parallel y^z - y_{pred}(x^z) \parallel^2
\end{aligned}
$$

Here the sum of squared errors, or L2 norm of the errors

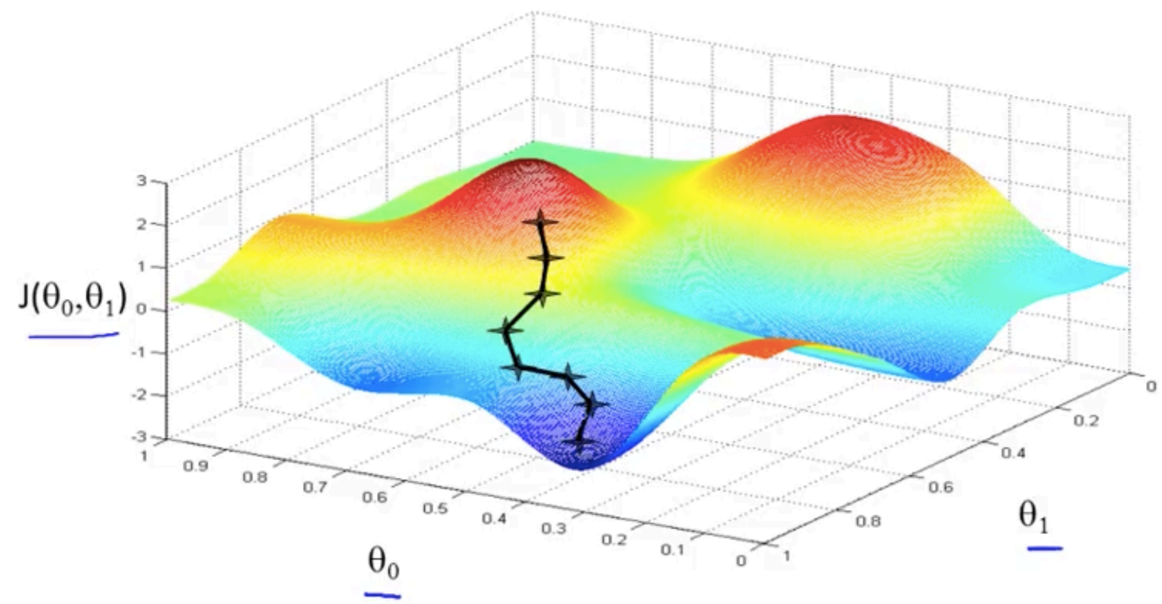
Many loss functions possible. When fitting more useful is the mean-square error (MSE)

$$
\begin{aligned}
J(w,b) &= \frac{1}{m} \sum_{z=0}^{m} \frac{1}{2} \parallel y^z - h^{(n_l)}(x^z) \parallel^2 \\
&= \frac{1}{m} \sum_{z=0}^{m} J(W,b,x^{(z)},y^{(z)})
\end{aligned}
$$

where m runs over all trainings pairs

We have evaluated the loss, so how does the network learn?

gradient descent and backpropagation



The loss function establishes a hypersurface for which we try to find a minimum using gradient descent

Gradient descent for every weight $w_{ij}^{(l)}$ and every bias $b_i^{(l)}$ in the NN looks like:

$$w_{ij}^{(l)} = w_{ij}^{(l)} - \alpha \frac{\partial}{\partial w_{ij}^{(l)}} J(w,b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(w,b)$$

in short:

$$w_{new} = w_{old} - \alpha * \nabla error$$

where α is the learning rate

Learning via backpropagation

- Backpropagation is method to compute the partial derivative of the loss function E(y,y'). It is about determining how changing the weights impact the overall loss in the NN
  - variation of loss with respect to weight w_k of NN is

$$\frac{dE}{dw_k} = \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k}$$ with

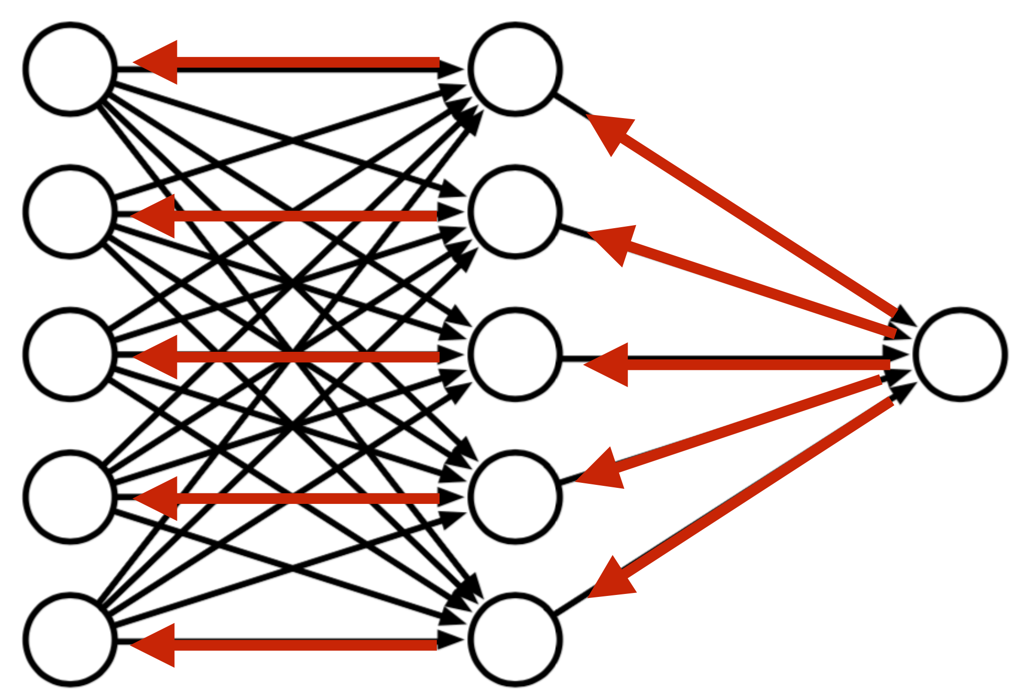
comb of weights $s = \sum_k w_k h_k$

activation function y

- weights of network adjusted by learning rate μ

$$\Delta w_k = -\mu \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k}$$

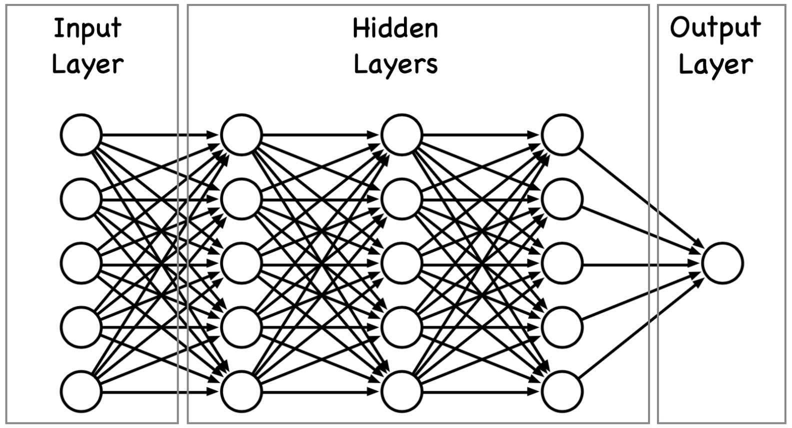
- New network weights reduce value of loss function



Classical Neural Network recap

Very powerful principle which NNs are designed to exploit

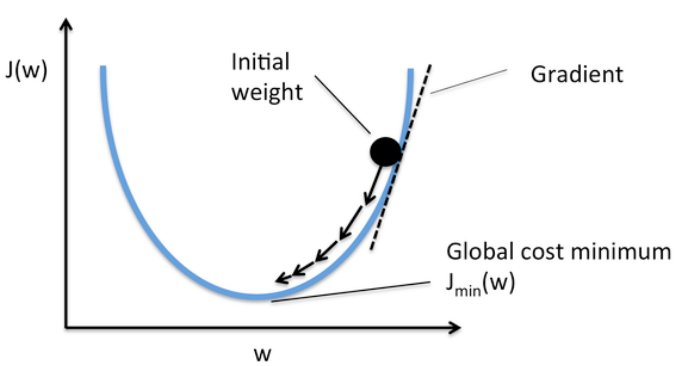
1. an adaptable complex system that allows approximating a complicated function



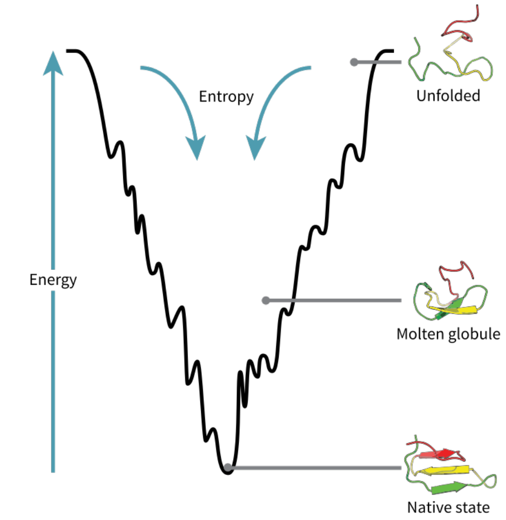
2. the calculation of a loss function in the output layer which is used to define the task the NN algorithm should perform by minimising this function

$$E(y,y') = \frac{1}{2}|y-y'|^2$$

3. a way to update the network continuously while minimising the loss function, e.g. backpropagation



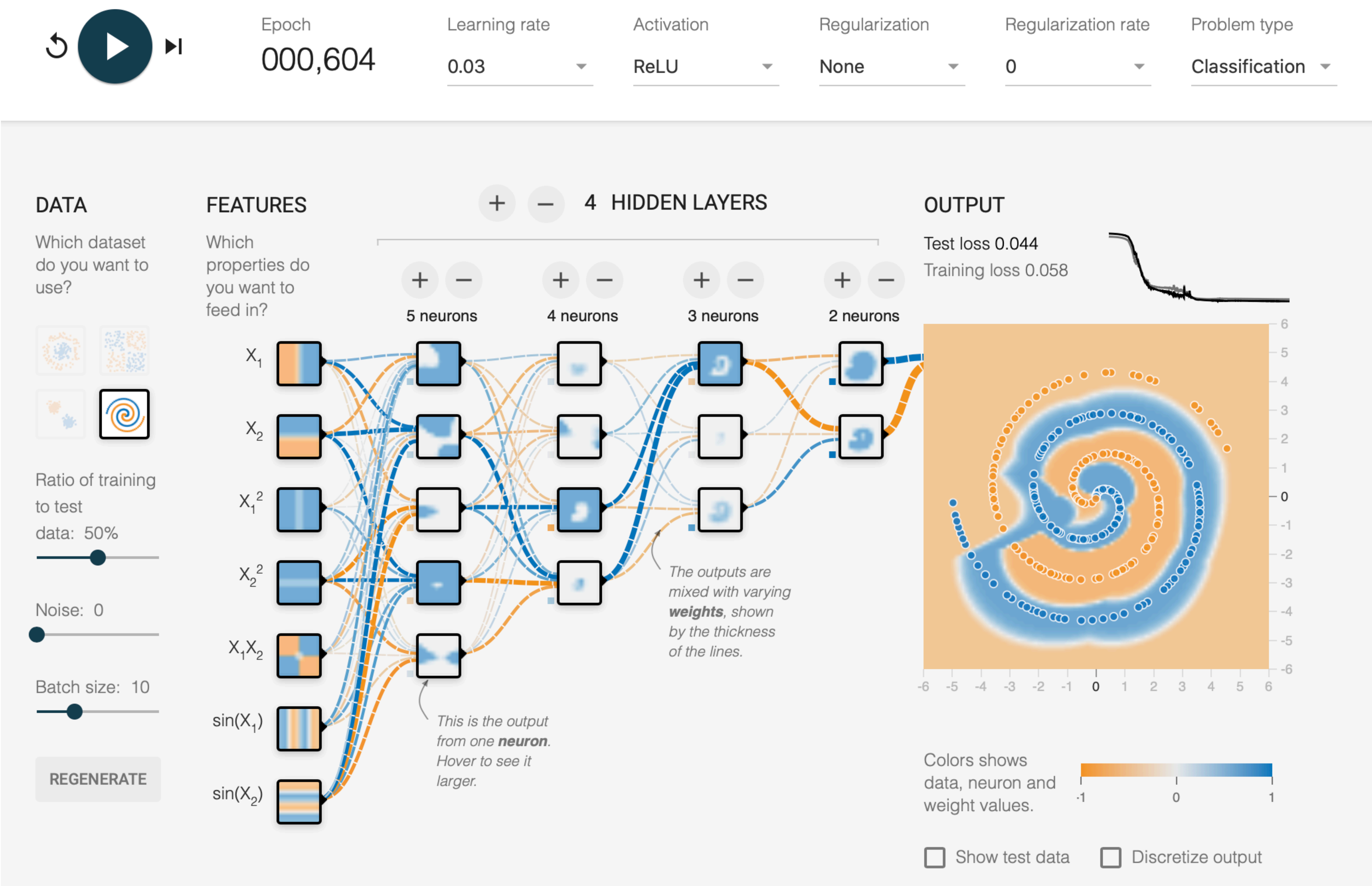
=



Difficult to keep all in quantum system - but not impossible? stay tuned!

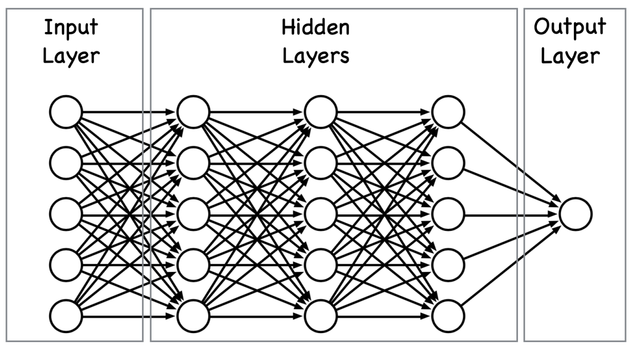
Visualisation of supervised learning using

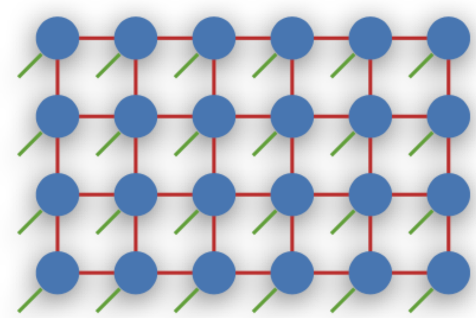
http://playground.tensorflow.org/

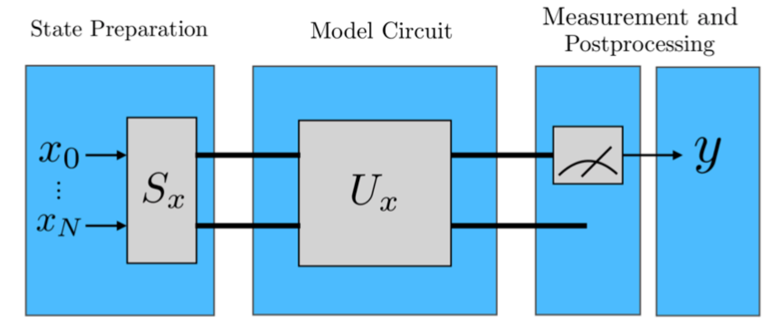


Classical ML Algorithms | Tensor Networks | Quantum Computing

1. an adaptable complex system that allows approximating a complicated function







2. the calculation of a loss function used to define the task the method

$$E(y, y') = \frac{1}{2}|y - y'|^2$$

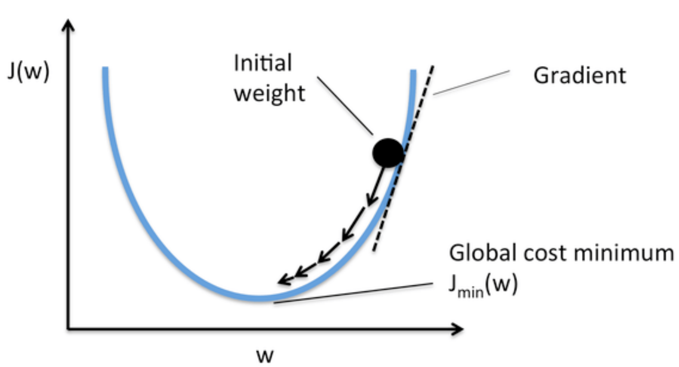
$$B^{s_2}{}_{p_1 p_2}\Gamma^{l p_1 p_2}{}_{s_2} = f^l(\mathbf{x}^{(\mathbf{n})})$$

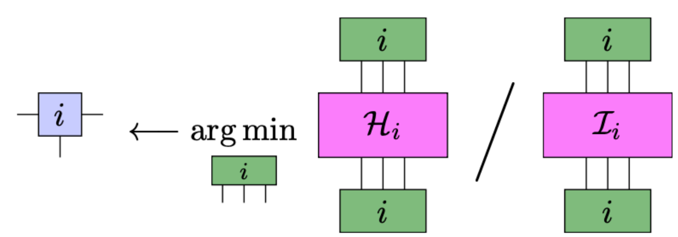
$$\mathcal{L} = L\left(p(l, \mathbf{x}),\ l^{truth}\right)$$

ground state

$$|\Gamma\rangle := \underset{|\psi\rangle \in \mathcal{D}}{\arg\min} \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle}$$

3. a way to update 1. while minimising the loss function





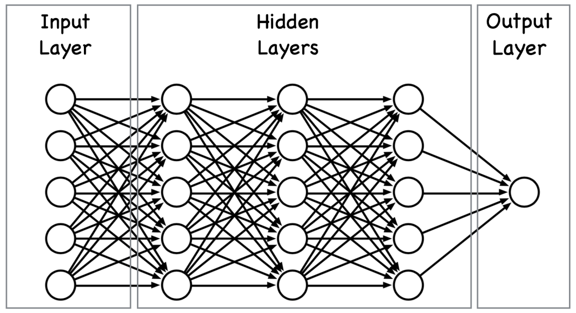
quantum: annealing

hybrid: classical opti.

optimisation → Data Analysis (Classification, anomaly, regression, fitting, ...)
→ Simulation of field theories (Groundstate, tunnelling, Real-time...)
→ Calculation of differential equations, etc etc

How can QNN be superior to NN

1. an adaptable complex system that allows approximating a complicated function

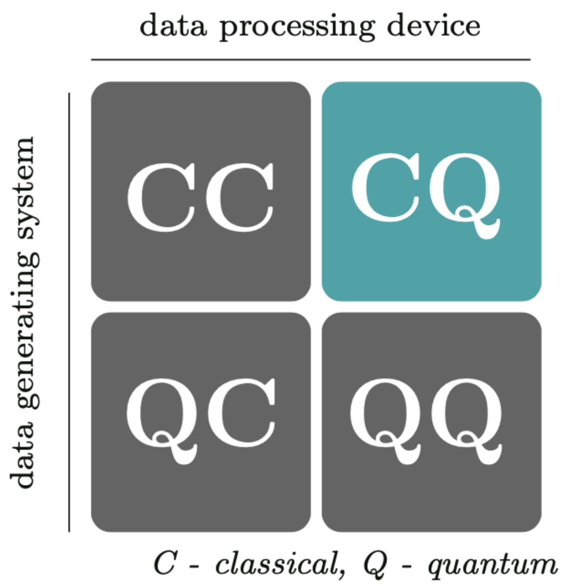


2. loss function

$$E(y, y') = \frac{1}{2}|y - y'|^2$$

- Input to QML can be quantum state [Huang et al '21]

  proven exponential advantage on noisy device over classical algorithm of any size



- QML more expressive

  [Eisert, Cramer, Plenio '08] [Alcazar, Leyton-Ortega, Perdomo-Ortiz '20] [Araz, MS '22]

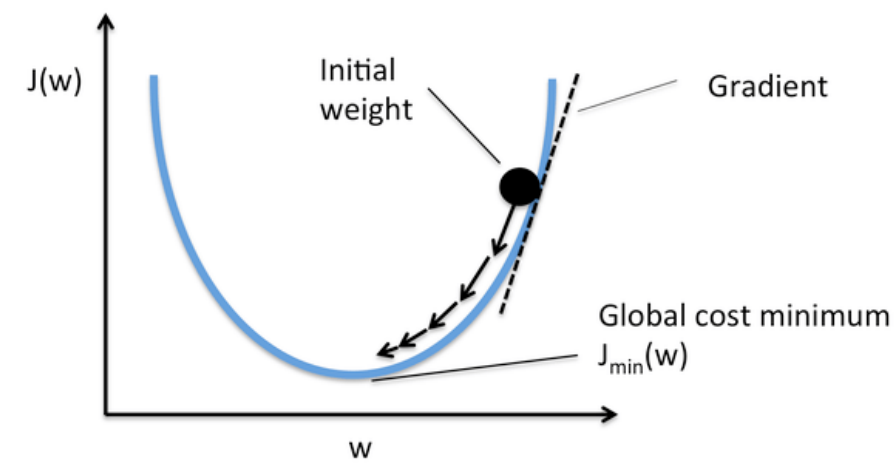
- Hybrid model possible - combination of classical and quantum nodes

- Exploit geometry of quantum loss function

  $\longrightarrow$ Faster learning [Stokes, et al '20] [Blance, MS '20]

How can QNN be superior to NN

3. a way to update the network continuously while minimising the loss function, e.g. backpropagation



- Quantum sampling of loss function/energy function
- Faster learning, i.e. faster groundstate finding of loss function
- More reliable in finding the global minimum of the loss function

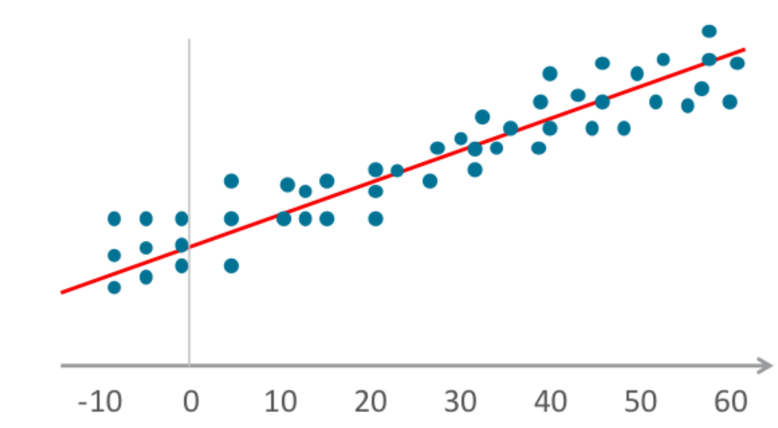
→ Potentially: Less sensitive to Barren Plateaus

Learns faster and from less data

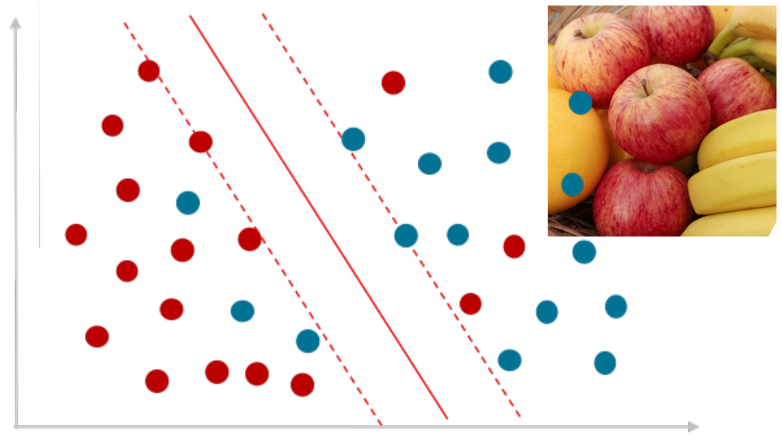
Doesn't get stuck in local minima
(less random in outcomes -> more interpretable)

Supervised

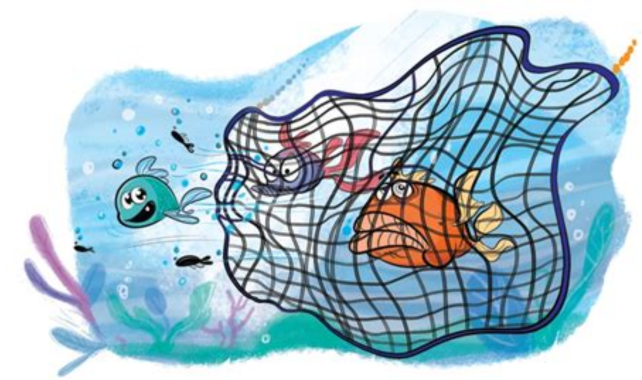
Regression



Classification

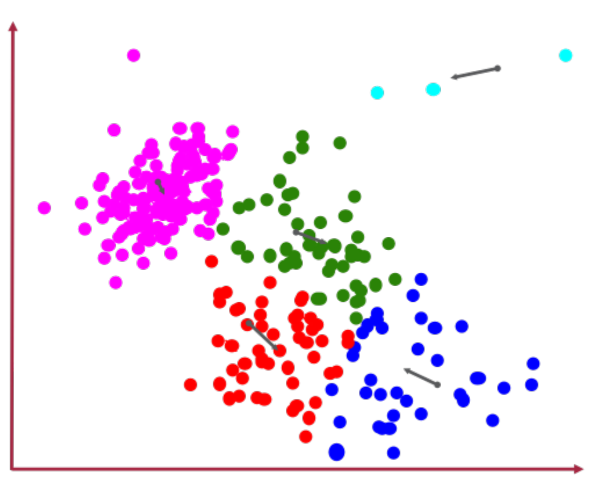


Fine-grained small net

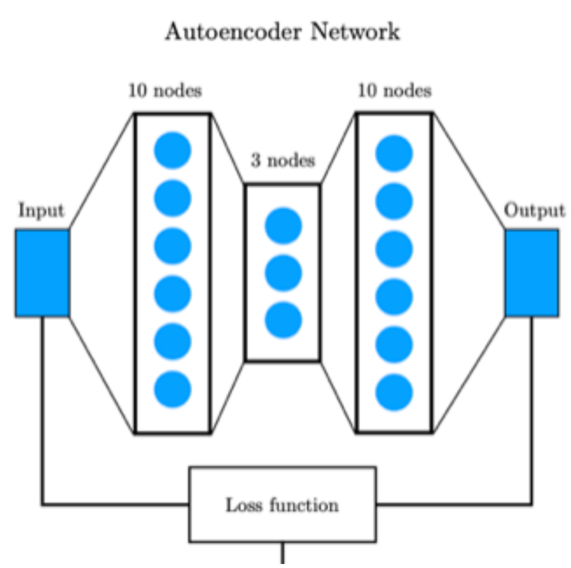


Unsupervised

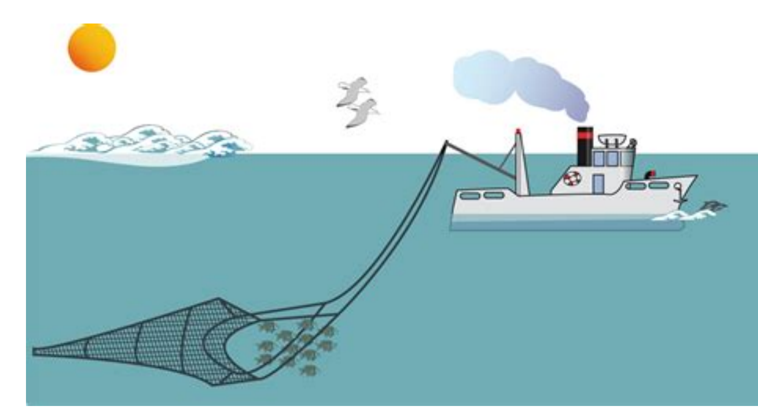
Clustering



Autoencoder



Large net



for quantum continuous variable algorithm see [Blance, MS '21]

Toy example

[Schuld, Petruccione '21]

- Squared distance classifier with quantum interference via Hadamard gate

- Titanic data set:

Input data

⟶ 2 x 2D vectors

inspired by Kaggle.com

	Raw data		Preprocessed data		Survival
	Price	Cabin	Price	Cabin	
Passenger 1	8,500	0910	0.85	0.36	1 (yes)
Passenger 2	1,200	2105	0.12	0.84	0 (no)
Passenger 3	7,800	1121	0.78	0.45	?

feature 1 feature 2 classify? label (1 or 0)

- Nearest neighbour to classify
  <-> need distance measure

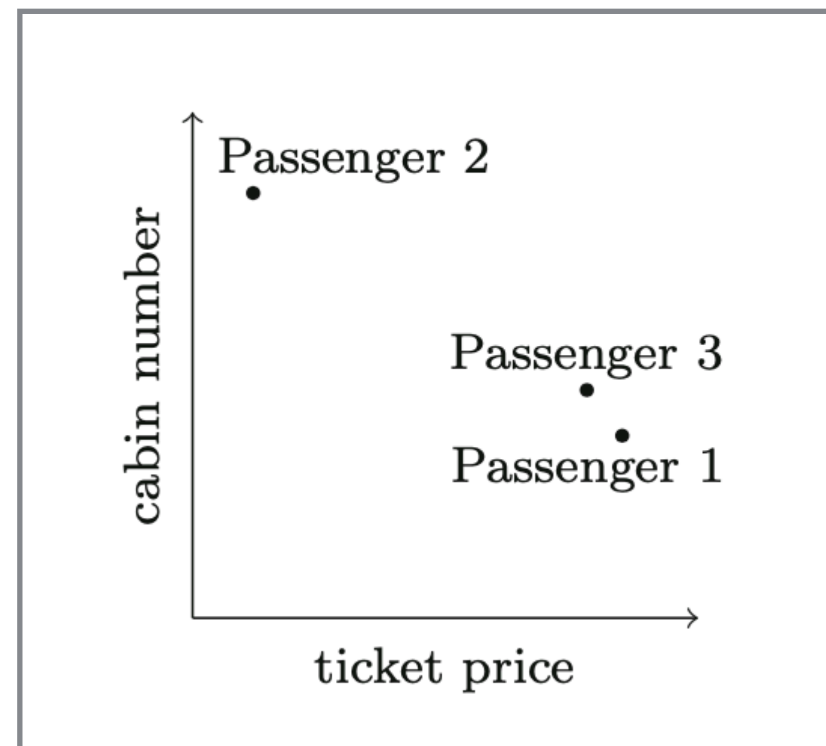
- Define probability by squared distance classifier:

$$p_{\tilde{x}}(\tilde{y}=1) = \frac{1}{\chi}\frac{1}{M_1}\sum_{m|y^m=1}\left(1-\frac{1}{c}|\tilde{x}-x^m|^2\right)$$

tot prob. normalised to 1

sum of weights of all training inputs

prediction depends most on data points closest to test point



How to use Quantum Computer to calculate this classification

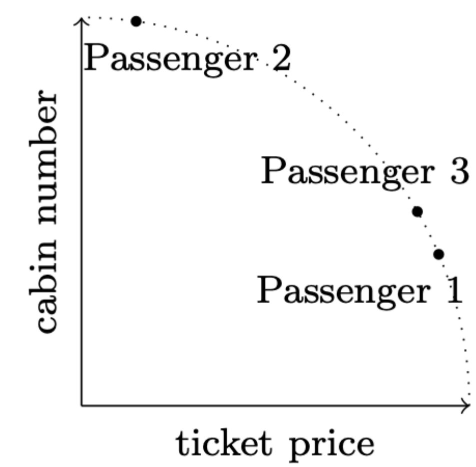
Step A: Data processing and inputing

Normalise length of input vector to 1

-> Project data onto unit-sphere, only angles remain

	Raw data	
	Price	Cabin
Passenger 1	8,500	0910
Passenger 2	1,200	2105
Passenger 3	7,800	1121

	price	room	survival
Passenger 1	0.921	0.390	yes (1)
Passenger 2	0.141	0.990	no (0)
Passenger 3	0.866	0.500	?



Step B: Data encoding (here amplitude encoding)

Data vector

Passenger 1 Passenger 2 Passenger 3 padding

$$\alpha = \frac{1}{\sqrt{4}} (0.921, 0.39, 0.141, 0.99, 0.866, 0.5, 0.866, 0.5)^T$$

|alpha|^2 = 1

length=2^3

Extend the state by 4th qubit -> 2^4 components

$$\alpha_{\text{init}} = \frac{1}{\sqrt{4}}(0, 0.921, 0, 0.39, 0.141, 0, 0.99, 0, 0, 0.866, 0, 0.5, 0.866, 0, 0.5, 0)^T$$

Vector component represent subamplitudes

for each feature encoded in an amplitude (e.g. 001), q4 is in the state that corresponds to the label of feature vector

Amplitudes assignment somewhat random, but does job…

q_1	q_2	q_3	q_4	Step B $\alpha_{\text{init}} \rightarrow$	
0	0	0	0	0	Passenger 1 (has label 1)
0	0	0	1	$\frac{1}{\sqrt{4}}0.921$	
0	0	1	0	0	
0	0	1	1	$\frac{1}{\sqrt{4}}0.390$	
0	1	0	0	$\frac{1}{\sqrt{4}}0.141$	Passenger 2 (has label 0)
0	1	0	1	0	
0	1	1	0	$\frac{1}{\sqrt{4}}0.990$	
0	1	1	1	0	
1	0	0	0	0	Passenger 3 (assuming label 1)
1	0	0	1	$\frac{1}{\sqrt{4}}0.866$	
1	0	1	0	0	
1	0	1	1	$\frac{1}{\sqrt{4}}0.500$	
1	1	0	0	$\frac{1}{\sqrt{4}}0.866$	Passenger 3 (assuming label 0)
1	1	0	1	0	
1	1	1	0	$\frac{1}{\sqrt{4}}0.500$	
1	1	1	1	0	

Step C: Apply Hadamard transformation on q1 $H_n^{(q_1)} = \frac{1}{\sqrt{2}} \begin{pmatrix} \mathbb{1} & \mathbb{1} \\ \mathbb{1} & -\mathbb{1} \end{pmatrix}$

Qubit state				Transformation of amplitude vector	
q_1	q_2	q_3	q_4	Step B $\boldsymbol{\alpha}_{\text{init}}$	Step C $\boldsymbol{\alpha}_{\text{inter}}$
0	0	0	0	0	0
0	0	0	1	$\frac{1}{\sqrt{4}}0.921$	$\frac{1}{\sqrt{8}}(0.921 + 0.866)$
0	0	1	0	0	0
0	0	1	1	$\frac{1}{\sqrt{4}}0.390$	$\frac{1}{\sqrt{8}}(0.390 + 0.500)$
0	1	0	0	$\frac{1}{\sqrt{4}}0.141$	$\frac{1}{\sqrt{8}}(0.141 + 0.866)$
0	1	0	1	0	0
0	1	1	0	$\frac{1}{\sqrt{4}}0.990$	$\frac{1}{\sqrt{8}}(0.990 + 0.500)$
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	$\frac{1}{\sqrt{4}}0.866$	$\frac{1}{\sqrt{8}}(0.921 - 0.866)$
1	0	1	0	0	0
1	0	1	1	$\frac{1}{\sqrt{4}}0.500$	$\frac{1}{\sqrt{8}}(0.390 - 0.500)$
1	1	0	0	$\frac{1}{\sqrt{4}}0.866$	$\frac{1}{\sqrt{8}}(0.141 - 0.866)$
1	1	0	1	0	0
1	1	1	0	$\frac{1}{\sqrt{4}}0.500$	$\frac{1}{\sqrt{8}}(0.990 - 0.500)$
1	1	1	1	0	0

$\frac{N}{2} \times \frac{N}{2}$ where $N = 2^n$

$$H \cdot \alpha_{\text{init}} = \alpha_{\text{inter}}$$

Superposition q1=0/1 states

-> connects training data (0) with new/testing data (1)

-> just one computational operation but acts on all subamplitudes

Step D: Measure the first qubit and **only accept if in state 0**

Introduces an if statement into quantum algorithm

-> similar to rejection sampling

-> q1 has to be in 0

-> zero all amplitudes with q1=1 and renormalise such that total probability is 1 (see chi)

Qubit state				
q_1	q_2	q_3	q_4	Step D $\boldsymbol{\alpha}_{\text{final}}$
0	0	0	0	0
0	0	0	1	$\frac{1}{\sqrt{8\chi}}(0.921+0.866)$
0	0	1	0	0
0	0	1	1	$\frac{1}{\sqrt{8\chi}}(0.390+0.500)$
0	1	0	0	$\frac{1}{\sqrt{8\chi}}(0.141+0.866)$
0	1	0	1	0
0	1	1	0	$\frac{1}{\sqrt{8\chi}}(0.990+0.500)$
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

$$\chi = \frac{1}{8}(|0.921+0.866|^2 + |0.390+0.500|^2 + |0.141+0.866|^2 + |0.990+0.500|^2) = 0.902$$

Step E: Measure the last qubit with probability $p(q_4)$

measurement = sampling from prob distribution

Interpret probability for fourth qubit to be classifier output

We have:

$$p(q_4 = 0) = \frac{1}{4\chi}\left(|0.141 + 0.866|^2 + |0.990 + 0.500|^2\right) \approx 0.448$$

$$p(q_4 = 1) = \frac{1}{4\chi}\left(|0.921 + 0.866|^2 + |0.390 + 0.500|^2\right) \approx 0.552$$

which is exactly the squared distance classifier $p_{\tilde{x}}(\tilde{y} = 1) = \frac{1}{\chi}\frac{1}{M_1}\sum_{m|y^m=1}\left(1 - \frac{1}{c}|\tilde{x} - x^m|^2\right)$.

$\boxed{c = 4}$

$$p(q_4 = 0) = \frac{1}{\chi}\left(1 - \frac{1}{4}(|0.141 - 0.866|^2 + |0.990 - 0.500|^2)\right) \approx 0.448,$$

$$p(q_4 = 1) = \frac{1}{4\chi}\left(1 - \frac{1}{4}(|0.921 - 0.866|^2 + |0.390 - 0.500|^2)\right) \approx 0.552$$

with $\chi = \frac{1}{4}(|0.921 + 0.866|^2 + |0.390 + 0.500|^2 + |0.141 + 0.866|^2 + |0.990 + 0.500|^2)$

Crux, after data encoding only **one computational operation** and **two simple measurements** needed Irrespective of size of input vector or dataset.

Some takeaway observations from example

- Data encoding often very important for quantum machine learning - especially for classical data
  influences runtime, the principles of algorithm etc
- The quantum algorithm imposes preprocessing requirements on classical data (e.g. regularisation and normalisation of data).
- Result of QML algorithm results from a measurement process. Thus, we need to run experiment several times
- Often QML algorithms are inspired by classical algorithms
- The way quantum computers work may require adaptations to classical models. Here, we used squared distance because it suited quantum formalism.