QIC 710: Introduction to Quantum Information Processing

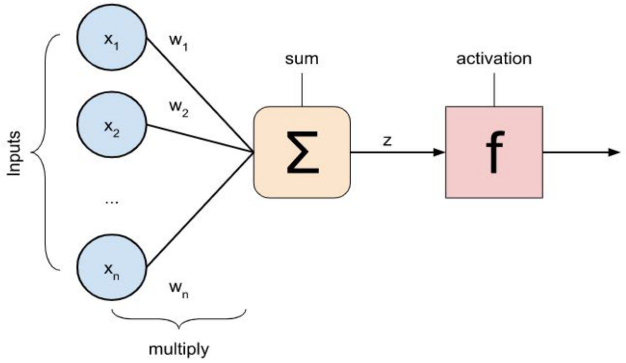
Instructor: Professor Richard Cleve

Prepared by: **Asad Raza,**
Exchange Student at the
Faculty of Mathematics,
University of Waterloo.

Quantum speed-up in global optimization of binary neural nets (arXiv: 1810.12948)

- Refresher on Classical Neural Networks and their training
- Quantum Binary Neural Networks
- Quantum Advantage in Training

Classical Neuron

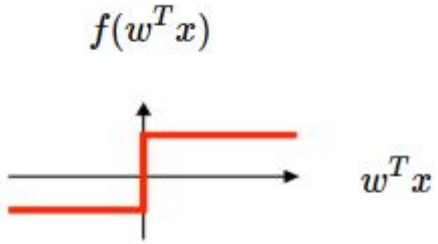


Representation of a single neuron

Image Source: Imperial College Machine Learning - Neural Networks

$$f(\sum_i w_i x_i)$$

Sign function →



Problem 1

NP-complete: No efficient algorithm to find the optimal $\vec{w}$ exists.

Number of training samples

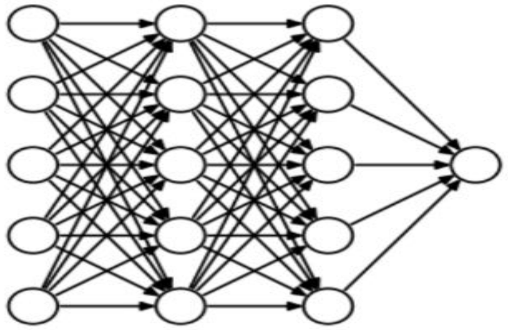
- $$C(\vec{x}_1, \vec{x}_2, \dots, \vec{x}_m, \vec{w}, \vec{y}) = \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f(\vec{x}_i, \vec{w})\right)^2$$

True Labels

Spoiler Alert!

Goal: Find the *optimal* $\vec{w}^*$ such that the cost, $C(\vec{x}_1, \vec{x}_2, \dots, \vec{x}_m, \vec{w}, \vec{y})$, is the lowest

Training Classical Neural Networks Using Gradient Descent



$$f(z) = f(\sum_i w_i x_i) = f(w^T x)$$

Problem 3

Gradient descent does not guarantee global convergence for non-convex functions

Weight of edge between neuron i in layer $l-1$ and neuron j in layer l, at $(t-1)$-st training cycle

Small learning rate

$$w_{ij}^{(l-1,l)}(t) = w_{ij}^{(l-1,l)}(t-1) - \eta \frac{\partial C}{\partial w_{ij}^{(l-1,l)}}$$

Problem 2

Negligible $w_{ij}^{(l-1,l)}(t)$ updates → No learning

But how to calculate $\frac{\partial C}{\partial w_{ij}^{(l-1,l)}}$?

$$\frac{\partial C}{\partial w_{ij}^{(L-1,L)}} = \frac{\partial C}{\partial f\left(z_j^{(L)}\right)} \frac{\partial f(z_j^{(L)})}{\partial w_{ij}^{(L-1,L)}} = \frac{\partial C}{\partial f\left(z_j^{(L)}\right)} \frac{\partial f\left(z_j^{(L)}\right)}{\partial z_j^{(L)}} \frac{\partial z_j^{(L)}}{\partial w_{ij}^{(L-1,L)}}$$

→ The derivatives for most activation functions range between 0 and 1, which are multiplied L-1 times to compute $\frac{\partial C}{\partial w_{ij}^{(L-1,L)}}$

→ $\frac{\partial C}{\partial w_{ij}^{(l-1,l)}} \approx 0$

Quantum Binary Neuron

Encode the weights, the inputs and the activations take values either +1 or -1. Dramatic Simplification (?)

Not as bad as it sounds. *See Bengio et al. (arXiv: 1602.02830)*

Scheme: Represent -1 as $|1\rangle$ and 1 as $|0\rangle$

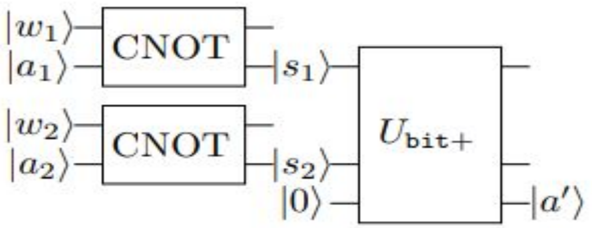


Figure 5. *Functioning of QBN:* the input data is encoded into quantum states. The multiplication succeeds by the CNOT gate and outputs the states $|s_1\rangle$, $|s_2\rangle$. Finally, the Toffoli gate, which has known decompositions into elementary gates, executes $U_{\mathtt{bit+}}$, leading to the output state $|a'\rangle$.

$$V(|w_1\rangle_1|a_1\rangle_2|w_2\rangle_3|a_2\rangle_4|0\rangle_5) = |w_1\rangle_1|s_1\rangle_2|w_2\rangle_3|s_2\rangle_4|a'\rangle_5$$

Generalize to Quantum Binary Feedforward Neural Network (**QBFNN**)

Inputs

All network weights

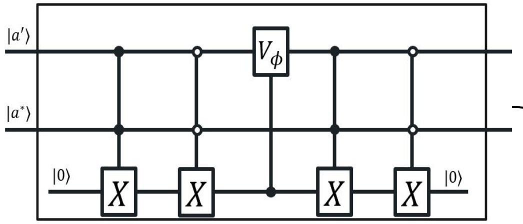
QBFNN outputs, encoded by ancillas

$$U\left(|a_1\rangle|a_2\rangle \dots |a_n\rangle|w_1\rangle|w_2\rangle \dots |w_N\rangle|0\rangle^{\otimes p}\right) = |\widetilde{a_1}, \widetilde{a_2}, \dots, \widetilde{a_n}, w_1, w_2, \dots, w_N, \text{rest}, a'_1, a'_2, \dots, a'_k\rangle$$

Saved values to achieve unitarity

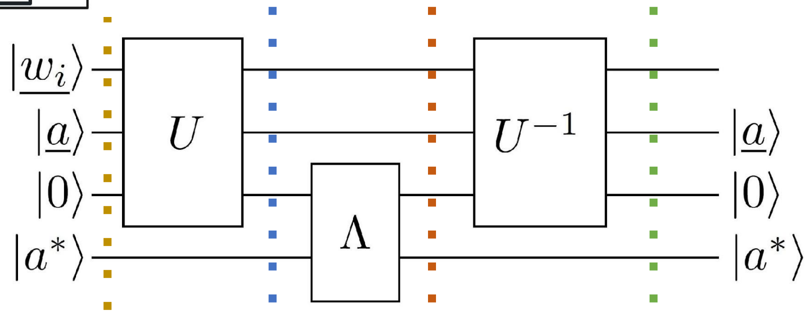
Training QBFNN: Single Weight Configuration

Pick **one pair** of training data and call it $(\vec{a}, a^*)$, **and one possible** weight configuration $\vec{w}_i$ (out of 2^N)



3. After oracle Λ :
$\Lambda(|\underline{w_i}, \underline{\tilde{a}}, a_i'\rangle|a^*\rangle) =$
$e^{i\Delta\theta_{\vec{w}_i}}|\underline{w_i}, \underline{\tilde{a}}, a_i'\rangle|a^*\rangle$

4. Uncomputation to disentangle:
$U^\dagger\left(e^{i\Delta\theta_{\vec{w}_i}}|\underline{w_i}, \underline{\tilde{a}}, a_i'\rangle|a^*\rangle\right) =$
$e^{i\Delta\theta_{\vec{w}_i}}|\underline{w_i}\rangle|\underline{a}\rangle|0\rangle|a^*\rangle$



1.Input
$|\underline{w_i}\rangle|\underline{a}\rangle|0\rangle|a^*\rangle$

2. After neuron/NN action:
$U(|\underline{w_i}\rangle|\underline{a}\rangle|0\rangle|a^*\rangle) =$
$|\underline{w_i}, \underline{\tilde{a}}, a_i'\rangle|a^*\rangle$

$e^{i\Delta\theta_{\underline{w_i}}}\,|\underline{w_i}\rangle$ → Reuse this weight state, $e^{i\Delta\theta_{\vec{w}_i}}|\vec{w}_i\rangle$, for the next training sample

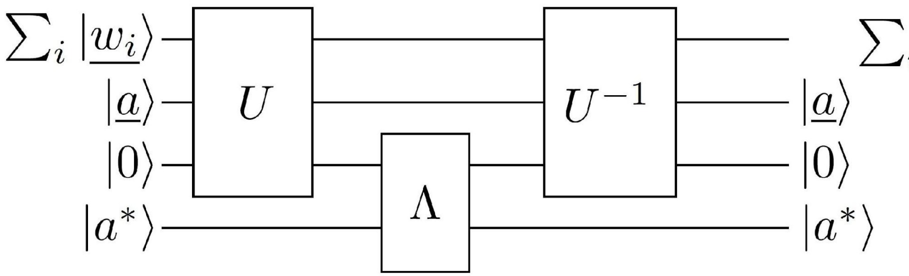
After looping through n data samples, the final weight state will read:

$e^{i\pi\frac{N_i}{n}}|\vec{w}_i\rangle$

$N_i \in \{0,1,\ldots,n\}$ is the number of correct predictions

Training QBFNN: Superposition of Weights

$$\frac{1}{\sqrt{2^N}}\sum_{i=1}^{2^N}|\underline{w_i}\rangle \longleftrightarrow$$



$$\sum_i e^{i\Delta\theta_{\underline{w_i}}}|\underline{w_i}\rangle \longleftrightarrow \frac{1}{\sqrt{2^N}}\sum_{i=1}^{2^N} e^{i\pi\frac{N_i}{n}}|\underline{w}$$

ONE loop instead of 2^N loops for each weight configuration

Since we loop through all the n training samples coherently, we can represent n rounds of $U^\dagger \Lambda U$ as one gigantic unitary $\tilde{\boldsymbol{U}}$

$$\left(\frac{1}{\sqrt{2^N}}\sum_{i=1}^{2^N}|\vec{w}_i\rangle\right)\otimes|\vec{a}_1\rangle|\vec{a}_2\rangle\ldots|\vec{a}_n\rangle|0\rangle^{\otimes p}|a_1^*\rangle|a_2^*\rangle\ldots|a_n^*\rangle$$

$$\xrightarrow{\tilde{U}}\left(\frac{1}{\sqrt{2^N}}\sum_{i=1}^{2^N}e^{i\pi\frac{N_i}{n}}|\vec{w}_i\rangle\right)\otimes|\vec{a}_1\rangle|\vec{a}_2\rangle\ldots|\vec{a}_n\rangle|0\rangle^{\otimes p}|a_1^*\rangle|a_2^*\rangle\ldots|a_n^*\rangle$$

Phase Estimation

Observe that $|\psi_i\rangle = |\vec{w}_i\rangle(\otimes_{j=1}^{n}|\vec{a}_j\rangle \otimes |a_j^*\rangle) \otimes |\underline{0}\rangle$, $i = 1,2,\ldots,2^N$ are **eigenvectors** of $\widetilde{U}$ with **eigenvalues** $e^{i\pi\frac{N_i}{n}}$ since $\widetilde{U}|\psi_i\rangle = e^{i2\pi\frac{N_i}{2n}}|\psi_i\rangle$

Estimate $\varphi^{(i)} = \frac{N_i}{2n}$ using phase estimation to get the t-bit binary encoding $\varphi^{(i)} = |\varphi_1^{(i)}\varphi_2^{(i)}\ldots\varphi_t^{(i)}\rangle$

Since phase estimation works for a superposition of eigenvectors, $\sum_i|\vec{w}_i\rangle \otimes (\otimes_{j=1}^{n}|\vec{a}_j\rangle \otimes |a_j^*\rangle) \otimes |0\rangle \otimes |0\rangle^{\otimes t}$

$\overset{PE}{\rightarrow} \sum_i|\vec{w}_i\rangle \otimes \left|\varphi_1^{(i)}\varphi_2^{(i)}\ldots\varphi_t^{(i)}\right\rangle(\otimes_{j=1}^{n}|\vec{a}_j\rangle \otimes |a_j^*\rangle) \otimes |0\rangle$

But what is a "good" N_i?

Fix a threshold count $N_t \in \mathbb{N}^+$. If $N_i \geq N_t$, N_i classifies as a "good enough" configuration.

Getting the "quality" weight vector

Define another **oracle** $O_{\pm 1}$, which acts on $\left|\varphi_1^{(i)}\varphi_2^{(i)} \dots \varphi_t^{(i)}\right\rangle$ as follows:

$$O_{\pm 1}\left|\varphi_1^{(i)}\varphi_2^{(i)} \dots \varphi_t^{(i)}\right\rangle = \begin{cases} -\left|\varphi_1^{(i)}\varphi_2^{(i)} \dots \varphi_t^{(i)}\right\rangle, if\ N_i \geq N_t \\ \left|\varphi_1^{(i)}\varphi_2^{(i)} \dots \varphi_t^{(i)}\right\rangle, if\ N_i < N_t \end{cases}$$

$$|\widehat{W}\rangle = \frac{1}{\sqrt{2^N}}\sum_i (-1)^{N_i \geq N_t}|\underline{w_i}\rangle$$

Started off with $\sum_i |\vec{w}_i\rangle \otimes \left(\otimes_{j=1}^{n}|\vec{a}_j\rangle \otimes |a_j^*\rangle\right) \otimes |0\rangle \otimes |0\rangle^{\otimes t}$

$\overset{PE}{\rightarrow} \sum_i |\vec{w}_i\rangle \otimes \left|\varphi_1^{(i)}\varphi_2^{(i)} \dots \varphi_t^{(i)}\right\rangle \left(\otimes_{j=1}^{n}|\vec{a}_j\rangle \otimes |a_j^*\rangle\right) \otimes |0\rangle$

$\overset{O_{\pm 1}}{\longrightarrow} \sum_i (-1)^{N_i \geq N_t}|\vec{w}_i\rangle \otimes \left|\varphi_1^{(i)}\varphi_2^{(i)} \dots \varphi_t^{(i)}\right\rangle \left(\otimes_{j=1}^{n}|\vec{a}_j\rangle \otimes |a_j^*\rangle\right) \otimes |0\rangle \overset{PE^{-1}}{\longrightarrow} \left(\sum_i (-1)^{N_i \geq N_t}|\vec{w}_i\rangle\right) \left(\otimes_{j=1}^{n}|\vec{a}_j\rangle \otimes |a_j^*\rangle\right) \otimes |0\rangle \otimes |0\rangle^{\otimes t}$

Problem: Entanglement between weights and phases!

Solution: $PE^{\dagger}$

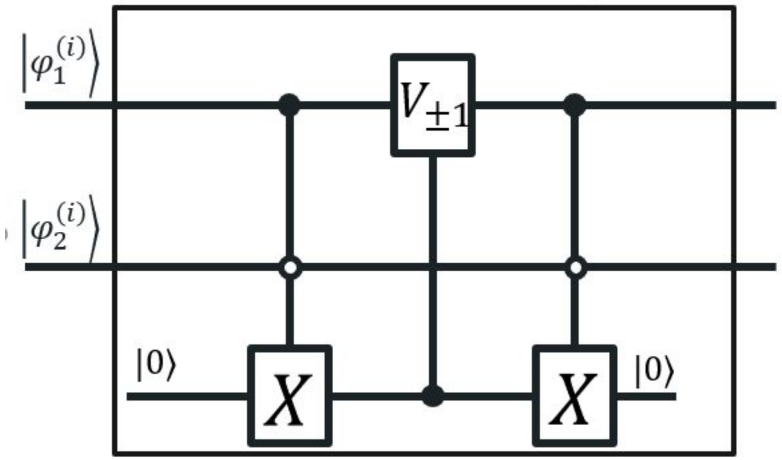
Uncompute phase estimation!

Applying Grover's to find the "good" weights

Example: $n = 2$

$N_t = n \rightarrow \frac{N_i}{2n} = \frac{1}{2}$

$\varphi^{(i)} = \frac{N_i}{2n}$, the encoding $|\varphi_1^{(i)}\varphi_2^{(i)}\rangle$ will be correspondingly $|10\rangle$



After $PE^\dagger$, we had the weight superposition $|\widehat{W}\rangle = \frac{1}{\sqrt{2^N}} \sum_i (-1)^{N_i \geq N_t} |\underline{w_i}\rangle$

A Grover state!

The Grover oracle acted as

$$O|x\rangle = \begin{cases} -|x\rangle, if\ x\ is\ solution\ to\ search \\ |x\rangle, else \end{cases}$$

$O(\sum_i |x_i\rangle) = \sum_i (-1)^{f(x_i)} |x_i\rangle$, where $f(x_i) = 1$ if x is solution and 0 else

Iterate for $k^* = \sqrt{\frac{2^N}{M}} \frac{\pi}{4}$ times to get the optimal weight vector, $\vec{w}^*$

What have we achieved?

- To ensure globally optimal set of weight, need $\approx 2^N$ steps (≈ Problem 1)
- Vanishing/Exploding gradient issues for gradient based trainings (Problem 2)
- Gradient descent almost never finds globally optimal solutions. Difference between local and global optimum can be **huge** (Problem 3)

- Find globally optimal set of weights in $\approx \sqrt{2^N}$ steps
- No vanishing gradients (Problem 2 resolved). Even better, no gradients involved.
- Guaranteed to find globally optimal weight configuration (Problem 3 resolved)

Thank you for listening!
Questions?