QBNN: Quantum Binary Neural Network

Divye Jain, Harsh Patwari, Jack Khuu

What is a QBNN?

Binary Neural Network implemented in quantum space

What is a BNN? [1]

Neural network where the weights and activation are binary at runtime (i.e 0/1, -1/1)

Why BNN?

Reduces the amount of space needed for representation [1]

Allows many arithmetic operations to be bit operations [1]

Why quantum?

Potential improvement/speed up by replacing established classical components with quantum implementations

Opens new avenues for algorithms that would be unreasonable in classical computing

Grover's Algorithm [2]

Linear searching with near unary accuracy in sub-linear time

Well established framework for adaption

$$O\sqrt{N}$$

Grover's Algorithm [2]

1 - Uniform Superposition

2 - Marking Desired States

3 - Reflect/Amplify

4 - Measure

https://people.com/tv/viral-video-grover-dropping-f-bomb-on-sesame-street-what-do-you-hear/

Method 1: Grover Search over data [3]

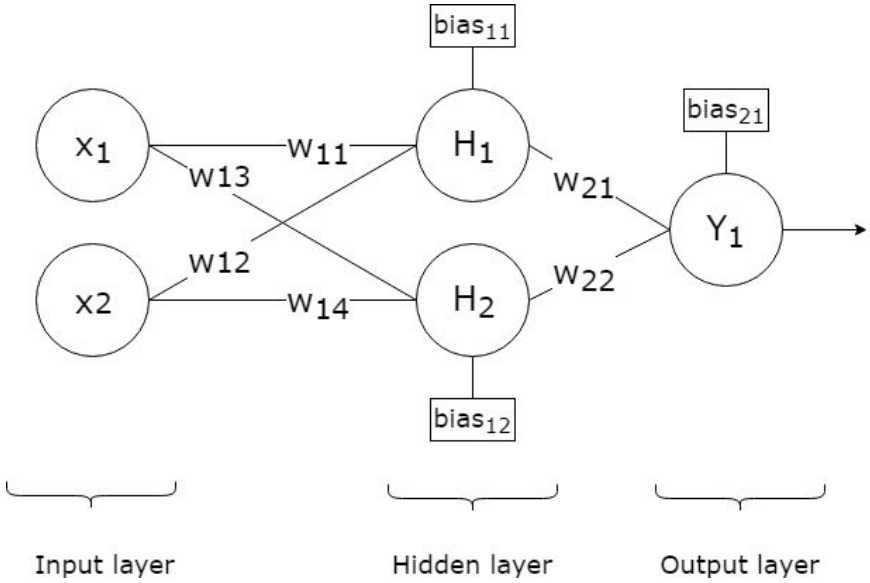
Search for data points (x, y) that are misclassified and that will update weights for each nodes

For example in classical perceptron, for misclassified input (x, y):

$$\omega_{new} \leftarrow \omega_{old} + xy$$

Method 1: Grover Search over data [3]

In a neural network:



Method 2: Grover Search for hyperplane [3]

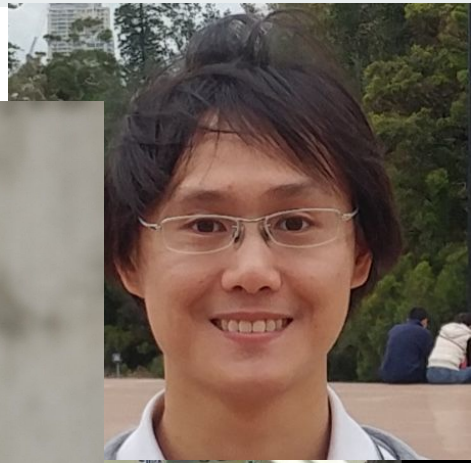
Instead of finding data to update weights, just find the weights themselves

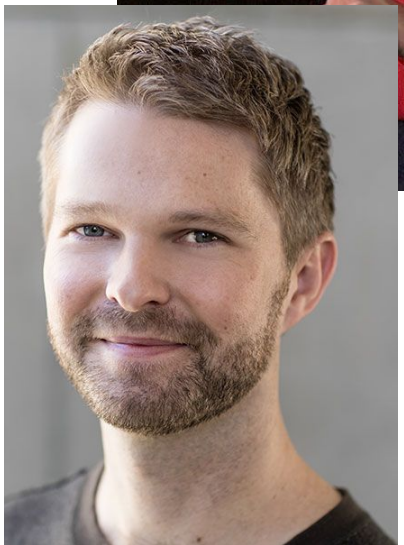
Search for a set of weights/hyperplane that separates the data (**x**, **y**)

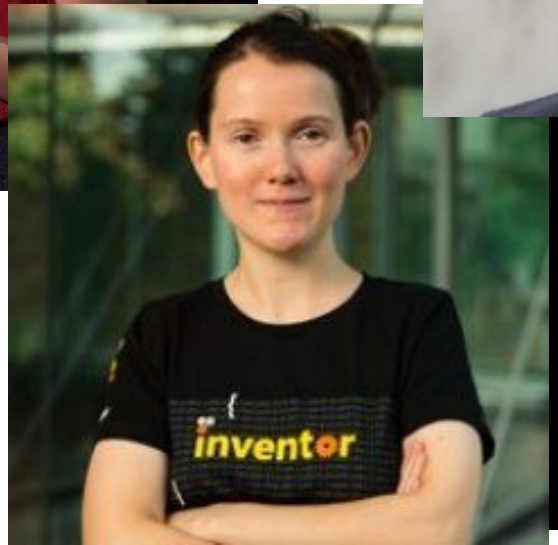
Grover's for Version Space Search

Let $\boldsymbol{w} = \{w_0, w_1, \ldots. w_n\}$, where w_i is a weight in the neural network.

1. Create an equal superposition over all possible sequences $\boldsymbol{w}$
2. Mark all $\boldsymbol{w}$ that separates the data with acceptable accuracy
3. Reflect about the mean to amplify the marked states
4. Measure the returned output






inventor

Grover's for Version Space Search

Let $\boldsymbol{w} = \{w_0, w_1, \ldots. w_n\}$, where w_i is a weight in the neural network.

1. Create an equal superposition over all possible sequences $\boldsymbol{w}$
2. Mark all $\boldsymbol{w}$ that separates the data with acceptable accuracy
3. Reflect about the mean to amplify the marked states
4. Measure the returned output

Marking Oracle

After creating a superposition over all states of the weight, the algorithm needs to mark all of the “correct weights”.

Overview: Marking Oracle

$$c_i = \begin{bmatrix} \cdot \\ \cdot \\ \cdots \\ \cdot \end{bmatrix}_{N\times 1}$$

N: The number of data points

$$c_i[j] = \begin{cases} 1 & \text{model "}w_i\text{" with input "}x_j\text{" predicted "}y_j\text{" correctly} \\ 0 & \text{model "}w_i\text{" with input "}x_j\text{" predicted "}y_j\text{" incorrectly} \end{cases}$$

Marking Oracle: Input: Qubit[] target: Qubit

$$|w_{\text{all}}\rangle = (\alpha_1|w_1\rangle + \ldots + \alpha_k|w_k\rangle)$$

$k = 2^n$; n is the length of the weight vector

Quantum Magic

$$|w_{\text{all}}\rangle = (\alpha_1|w_1\rangle|c_1\rangle + \ldots + \alpha_k|w_k\rangle|c_k\rangle)$$

Mark the target if "c" has at-least "acc"% of 1s
So if desired accuracy is 80% -> Flip target if "c" has at-least 80% 1s

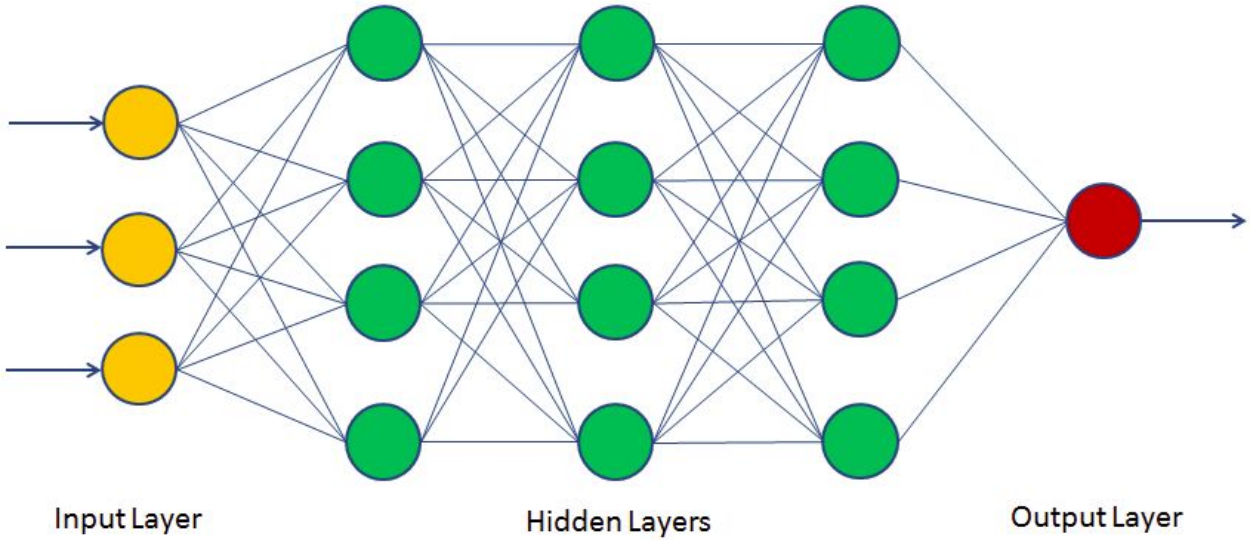
Undo Quantum Magic

Quantum Magic

$$c = \text{Qubit}[]$$
$$\text{For each data point } (x_j, y_j) :$$
$$\quad \text{Forward}(w, x_j, y_j, c[j])$$

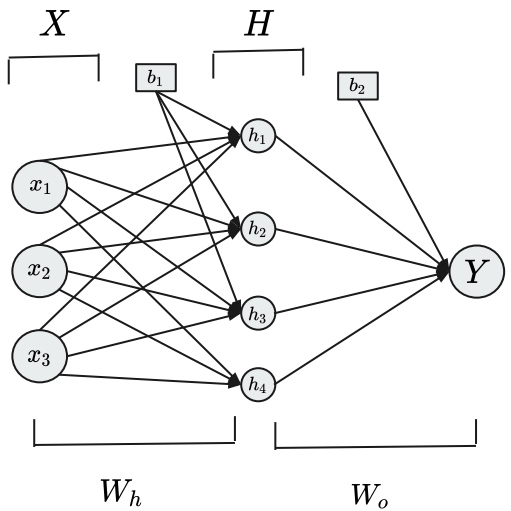
“Forward”: Our implementation of the BNN (Binary Neural Nets) forward pass with Quantum operators

Neural Net: Recap



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

How does a forward pass work?

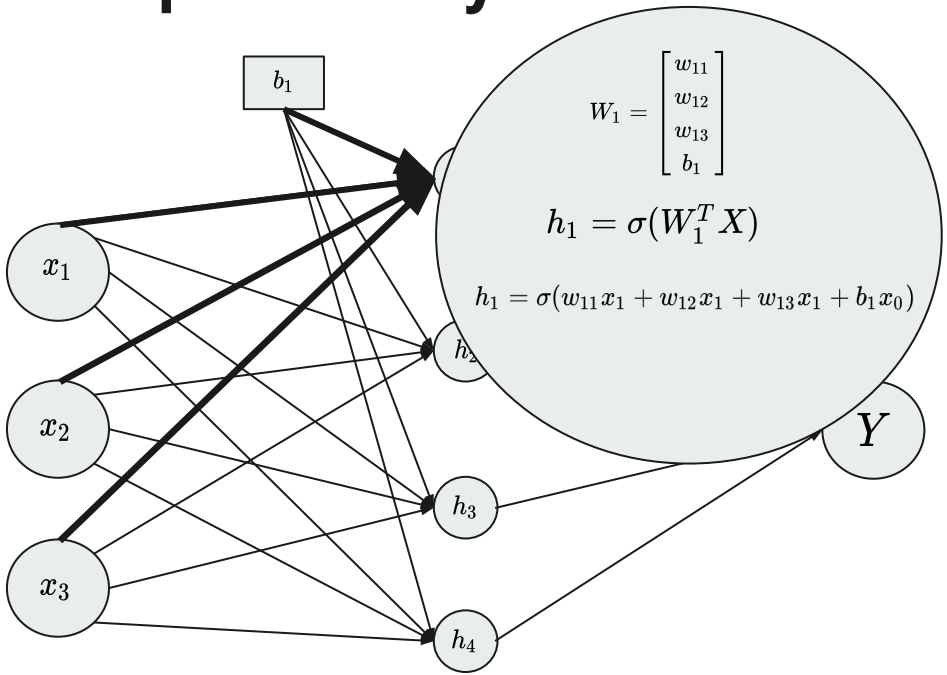


$$X \rightarrow \text{Input}$$

$$H = \sigma(W_h X + B_1)$$

$$Y = \text{Softmax}(W_o H + B_2)$$

More Specifically



$$\sigma(x) = \frac{1}{1+e^{-x}}$$

$$x_0 = 1$$

In our binary case

All weights and inputs are binary (+1 and -1)

The activation function is the “sign function”

$$h_1 = \sigma(w_{11}x_1 + w_{12}x_1 + w_{13}x_1 + b_1x_0) \longrightarrow h_1 = \text{Sign}(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_1x_0)$$

$$w_{11}, w_{12}, w_{13}, b_1, x_1, x_2, x_3, h_1 \in \{-1, 1\}$$

How to do it with Quanum Operators

$$x_1 = -1, x_2 = 1, x_3 = 1$$
$$w_{11} = -1, w_{12} = -1, w_{13} = 1, b_1 = 1$$

$$h_1 = \text{Sign}\left(w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + b_1\right)$$

FlipTargetIfSame FlipTargetIfSame FlipTargetIfSame FlipTargetIfSame

0 0 1 0 1 1 1 1

$h_1 = \text{Sign}(\boxed{-1 \cdot -1} + \boxed{1 \cdot -1} + \boxed{1 \cdot 1} + \boxed{1 \cdot 1})$

$h_1 = \text{Sign}(\boxed{1} + \boxed{-1} + \boxed{1} + \boxed{1})$

FlipTargetIfMajorityOnes(1 0 1 1)

↓

$h_1 = \text{Sign}(2) = +1$

How to do the entire forward pass in Quantum?

We just saw how we can do it for one node

Repeat the process for each node in the entire layer

Now, we can use RECURSION

output of one layer ⟶ input to the other

Final layer output ⟶ Mark “c[j]” if the output is the same as y[j]

Quantum Magic

$$c = \text{Qubit}[]$$
$$\text{For each data point } (x_j, y_j) :$$
$$\quad \text{Forward}(w, x_j, y_j, c[j])$$

“Forward”: Our implementation of the BNN (Binary Neural Nets) forward pass with Quantum operators

Final Marking Oracle

$$c_i = \begin{bmatrix} \cdot \\ \cdot \\ \cdots \\ \cdot \end{bmatrix}_{N\times 1}$$

N: The number of data points

$$c_i[j] = \begin{cases} 1 & \text{model "}w_i\text{" with input "}x_j\text{" predicted "}y_j\text{" correctly} \\ 0 & \text{model "}w_i\text{" with input "}x_j\text{" predicted "}y_j\text{" incorrectly} \end{cases}$$

Marking Oracle: Input: Qubit[] target: Qubit

$$|w_{\text{all}}\rangle = (\alpha_1|w_1\rangle + \ldots + \alpha_k|w_k\rangle)$$

$k = 2^n$; n is the length of the weight vector

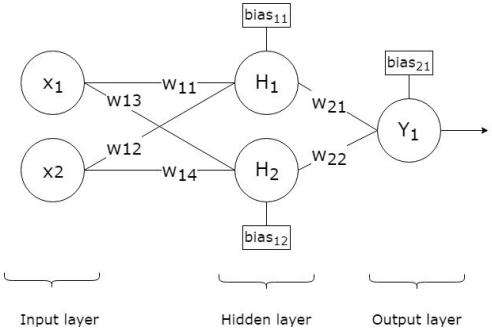
Entangle $|c\rangle$ with $|w\rangle$ by marking c[j] bit 1 if the output of the "forward pass" (from x[j]) matches the true output (y[j])

$$|w_{\text{all}}\rangle = (\alpha_1|w_1\rangle|c_1\rangle + \ldots + \alpha_k|w_k\rangle|c_k\rangle)$$

Mark the target if "c" has at-least "acc"% of 1s
So if desired accuracy is 80% -> Flip target if "c" has at-least 80% 1s

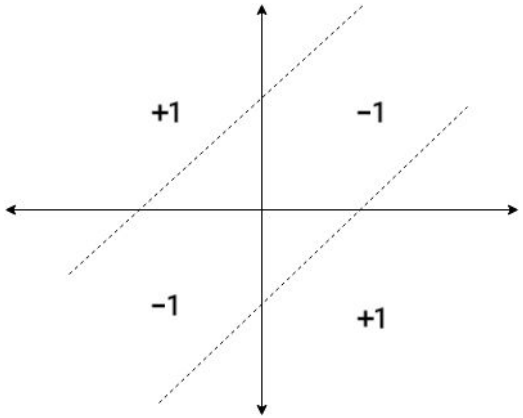
Un-Entangle $|c\rangle$ with $|w\rangle$ by marking c[j] bit 1 if the output of the "forward pass" (from x[j]) matches the true output (y[j])

Resources required

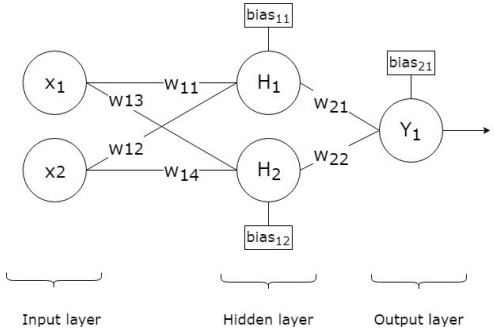


Let us take the example of the circuit for the XNOR gate:

XNOR is a simple example of a circuit that cannot be predicted with a classical single layer perceptron



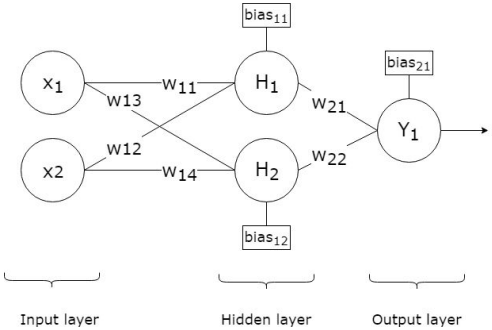
Resources required



Let us take the example of the circuit for the XNOR gate:

Total number of qubits required = 40

Resources required

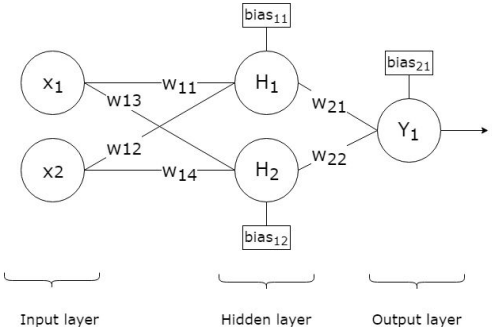


Let us take the example of the circuit for the XNOR gate:

data points = 4, # features = 2
true labels = 4

# qubits to represent the input	=	(4 x 2) + 4	= 12
# predicted labels			= 4
# qubits to represent the weights including the bias			= 9
Total			= **25**

Resources required



Let us take the example of the circuit for the XNOR gate:

Total number of qubits required by adder = $(3 \times (\log_2 4 + 3)) = 15$

Optimizations

Process inputs in batches

In this case, process a single data point and a single label at a time.

data points / batch = 1, # features = 2
true labels / batch = 1

qubits to represent the input = (1 x 2) + 1 = 3

Optimizations

Use recursion to replace adder

We use the adder for computing the majority and to check if the choice of weights gives us the desired accuracy.

Using a recursive approach to create a boolean array and do a ControlledOnBitString operation saves us 15 qubits.

Optimizations

Total number of qubits required after optimization = 25

Metric	Sum
CNOT	14119044
QubitClifford	4009671
R	0
Measure	3774
T	9883188
Depth	221340
Width	25
BorrowedWidth	0

Demo

Impact

- We can potentially create any random quantum circuit, given just the input and the expected output

- Our network doesn’t need to be differential!! Since we are no longer doing back prop

Works Cited

[1] Courbariaux, Matthieu and Bengio, Yoshua. Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1. arXiv preprint arXiv:1602.02830, 2016.

[2] Figgatt, C. et al. Complete 3-qubit grover search on a programmable quantum computer. Preprint at https://arxiv.org/abs/1703.10535 (2017)

[3] Wiebe, N., Kapoor, A. & Svore, K. M. Quantum perceptron models. Adv. Neural Inform. Process. Syst. 29, 3999–4007 (2016)

[4] John and Jakub for the QFT Adder :D

Thank You

Questions? Comments? T-shirts?