Quantum Neuron

an application of Repeat-Until-Success Circuits

Moritz Schmidt

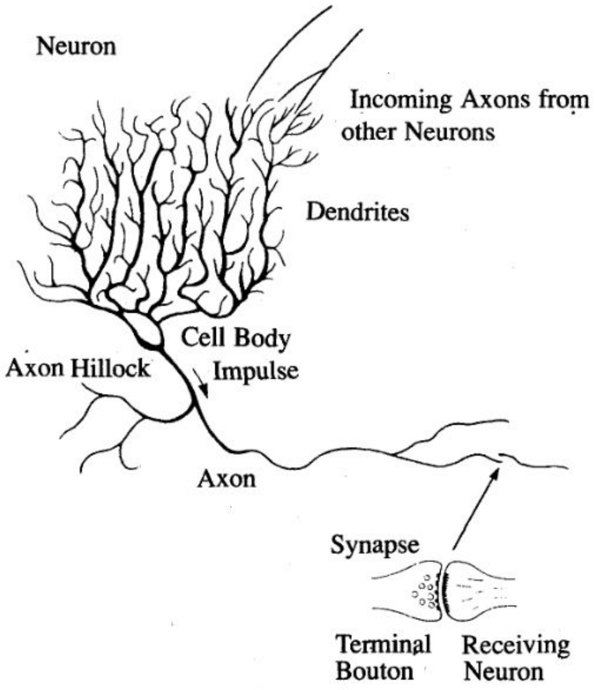
20.07.2021

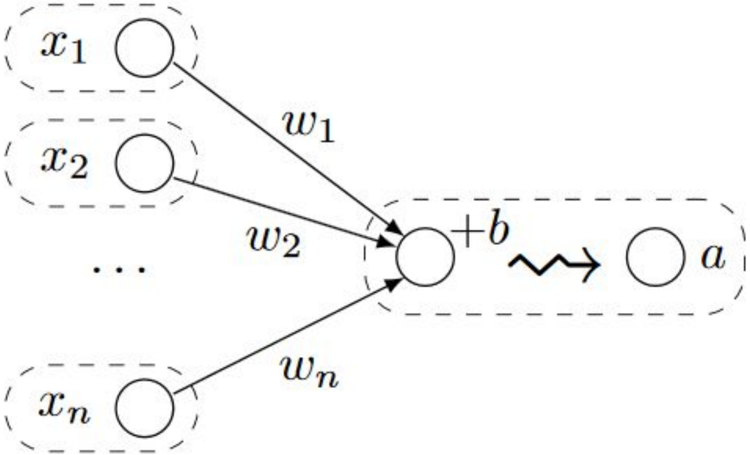
Outline

- (small) Introduction Artificial Neuron
- Quantum Artificial Neuron
- Implementation Issues In Qiskit

Classical Artificial Neuron

Neuron

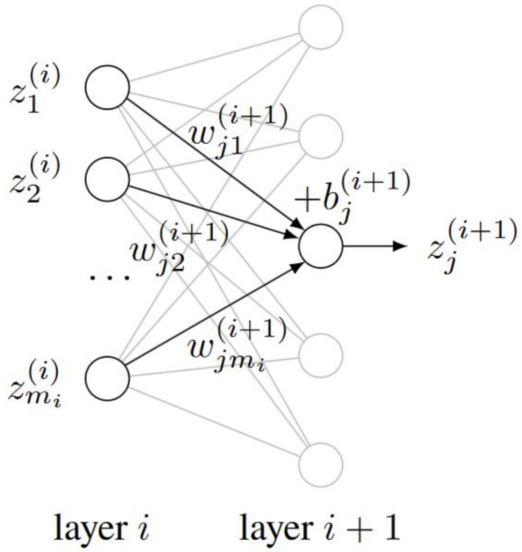




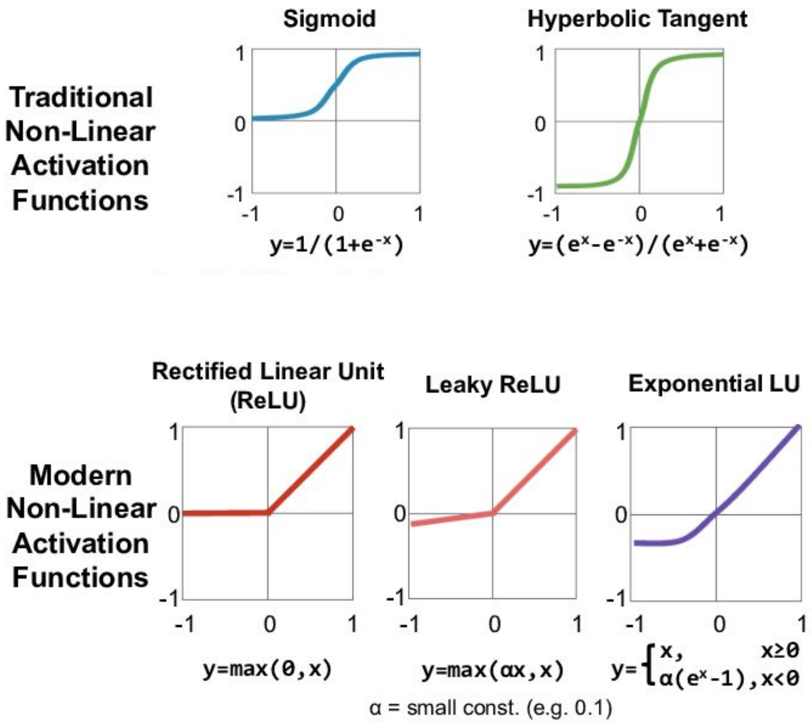
$$y = \sigma\left(\sum_{i}^{N} x_i w_i + b\right)$$

Neural Networks As Function Approximator

- Used to learn unknown functions
- Training data:
  - Input data x
  - Expected output y
- Optimize weights based on training error
- Learn to generalize for unknown test data



Activation Functions



https://www.researchgate.net/figure/Various-forms-of-non-linear-activation-functions-Figure-adopted-from-Caffe-Tutorial_fig3_315667264

Neural Network

next layer

previous layer

$$z^{(i)} = \sigma(\underbrace{W^{(i)} z^{(i-1)} + b^{(i)}}_{\text{linear combination}})$$

activation function

linear combination

$$y = \sum_{i}^{N} x_i w_i + b$$

Training

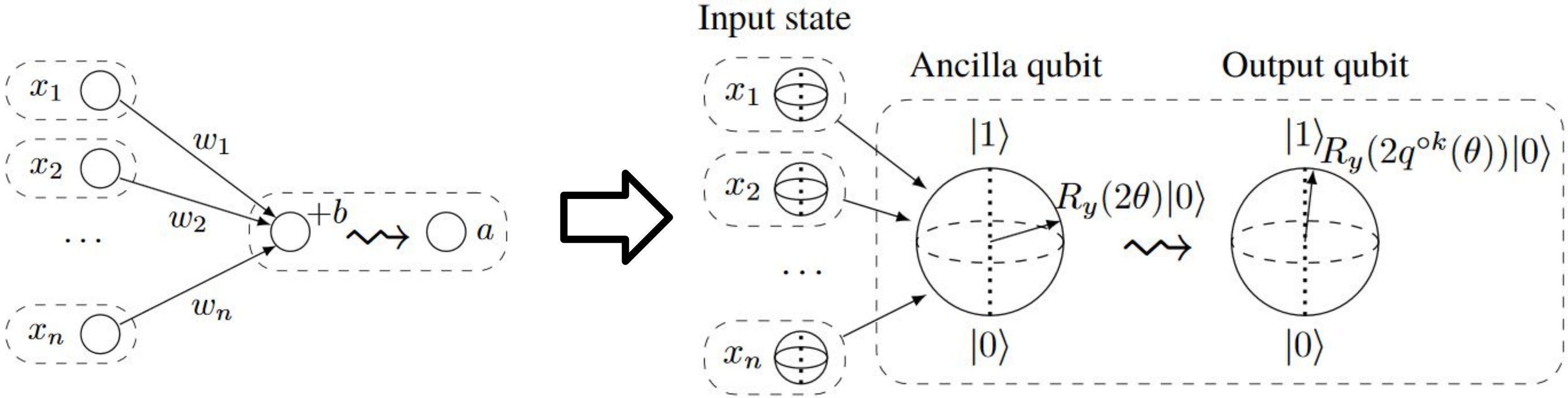
correct result output of network

- Loss Function: $$\min_{\mathbf{W},\mathbf{b}} \sum_{j=1}^{T} \|\mathbf{y}_j - f(\mathbf{x}_j)\|_2^2$$
- Often more sophisticated in practice (regularization terms,...)
- Optimization Problem!
- **Backpropagation**: Chain Rule + Gradient Descent + ‘Optimizer’
- Requires Gradient => Activation functions should be differentiable

Quantum Artificial Neuron

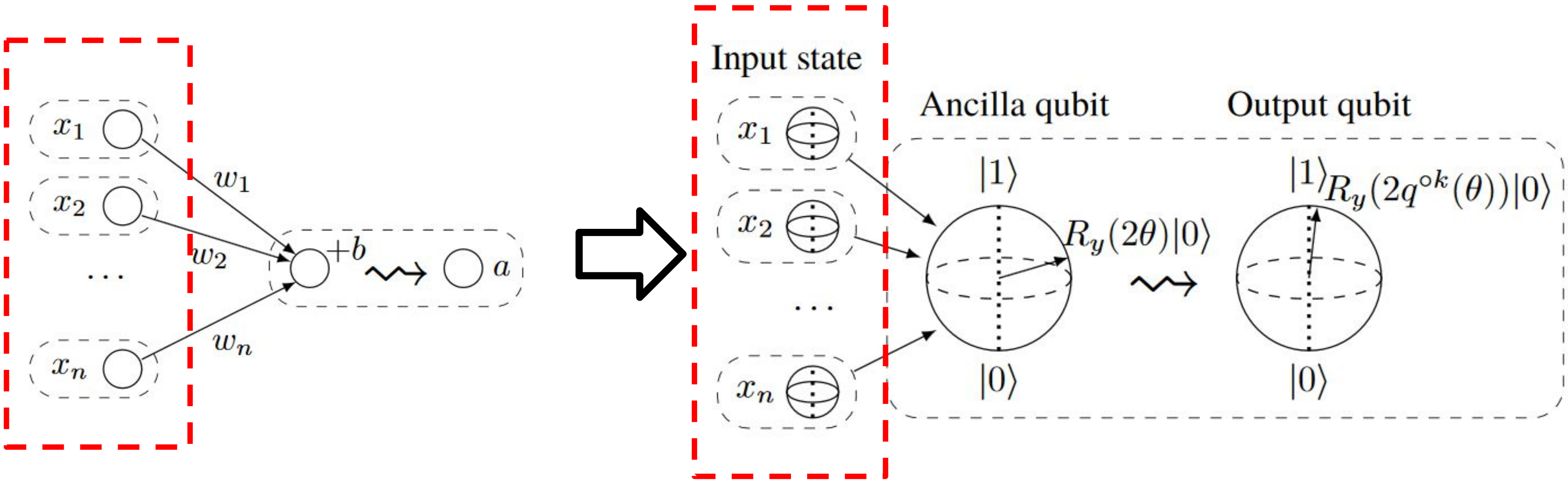
https://arxiv.org/abs/1711.11240

Quantum Perceptron: Concept



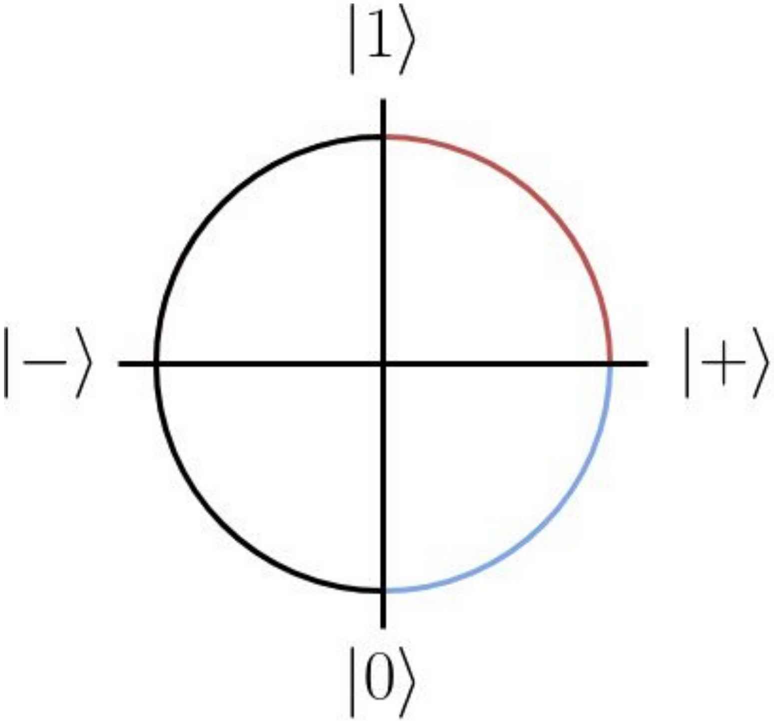
Encoding on the Bloch Sphere

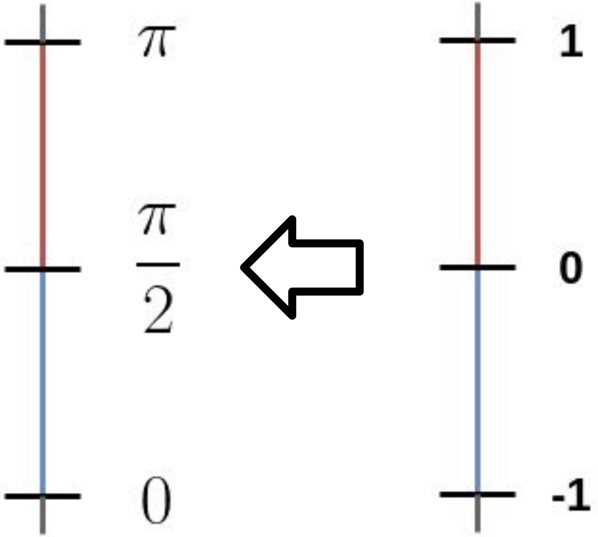
Assumption: $x \in [-1, 1]^n$



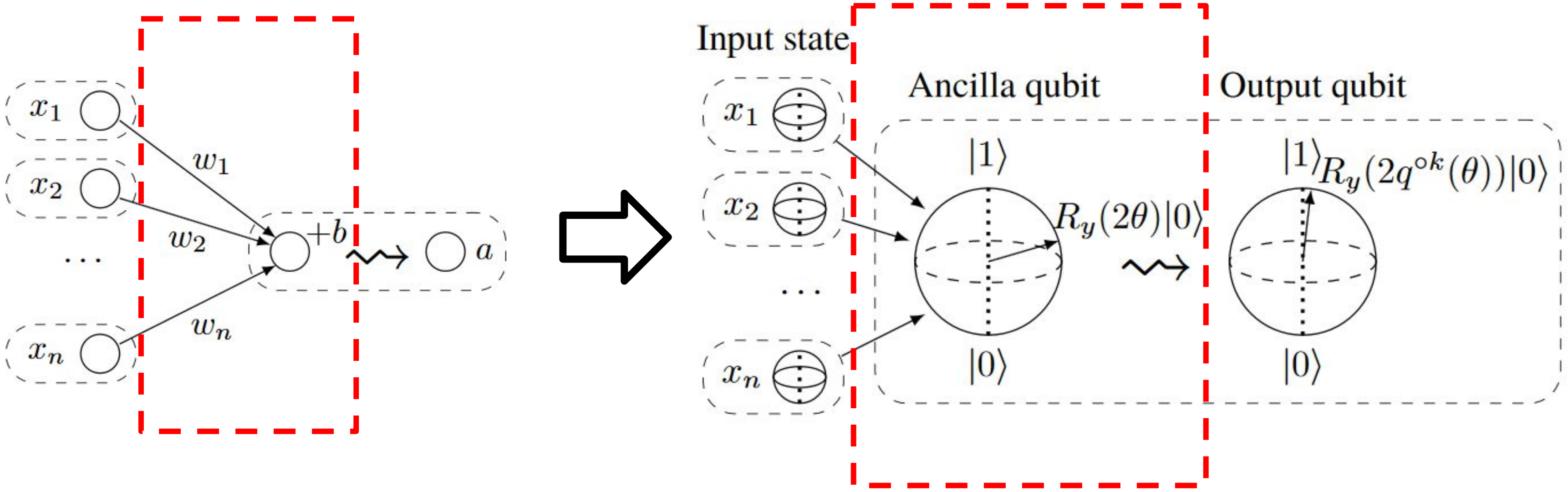
Encoding on the Bloch Sphere

Assumption: $x \in [-1, 1]^n$





Linear Combination



(1)

Linear Combination

- We want normalized data: $x \in [-1, 1]^n$
- We don't want extra constraints for Optimization: $w \in \mathbb{R}^n$
- Output has to be in bounds again! $y = \sum_{i}^{N} x_i w_i + b$

arbitrary magnitude depending on w, b

- Normalize by biggest magnitude of bias and weights

(1)

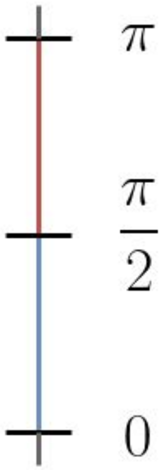
Linear Combination

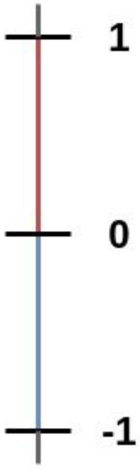
$$y = \sum_{i=0}^{N} x_i w_i$$

We start at angle $\frac{\pi}{2}$ and rotate along the Y-Axis depending on each $x_i w_i$

At Maximum rotate: $\frac{\pi}{2}$

At Minimum rotate: $-\frac{\pi}{2}$





We start at 0 and add up the terms $x_i w_i$

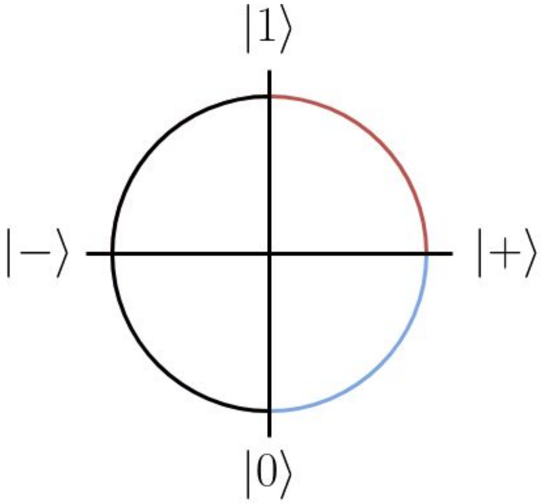
At Maximum add 1

At Minimum add -1

(1)

Linear Combination: Circuit

- Assume: $x_i \in \{-1, 1\}$
- We start in state $|+\rangle$
- We can view x_i as Rotation direction
- We can view w_i as Rotation angle
- How can we implement this as a Circuit?



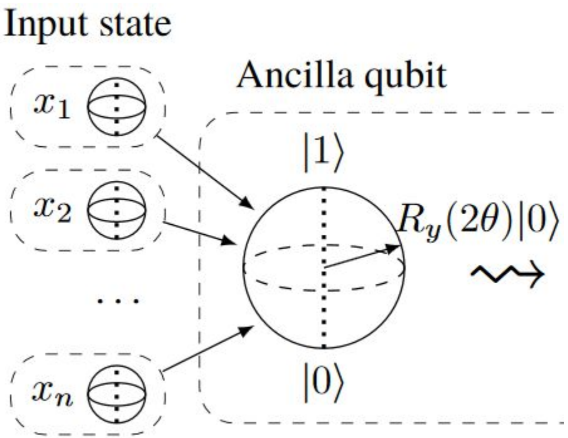
(1)

Linear Combination: Circuit

Case $x_i = 1$:

The Input Qubit is in State $|1\rangle$

Apply controlled rotation around Y-axis with with angle w_i



(1)

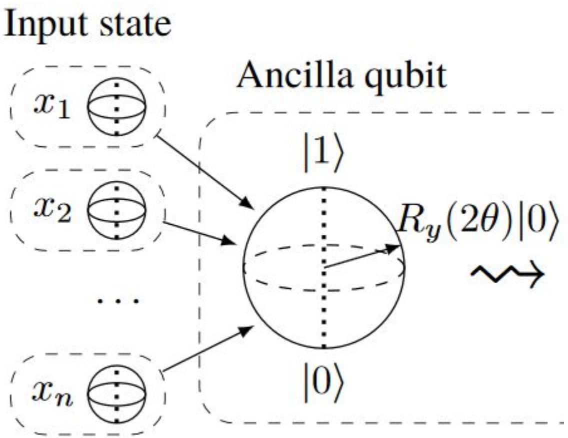
Linear Combination: Circuit

Case $x_i = 0$:

The Input Qubit is in State $|0\rangle$

We want to rotate with angle $-w_i$

But controlled rotation does not work here..

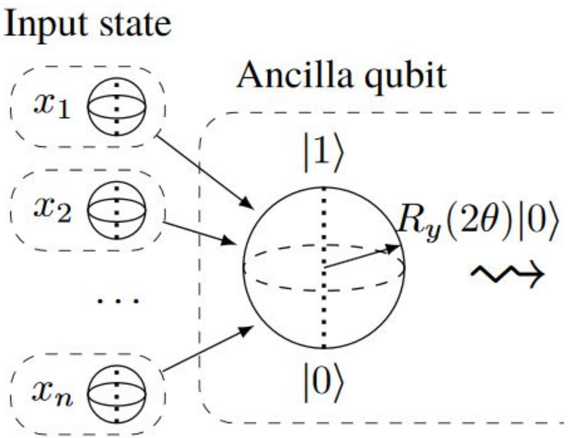


(1)

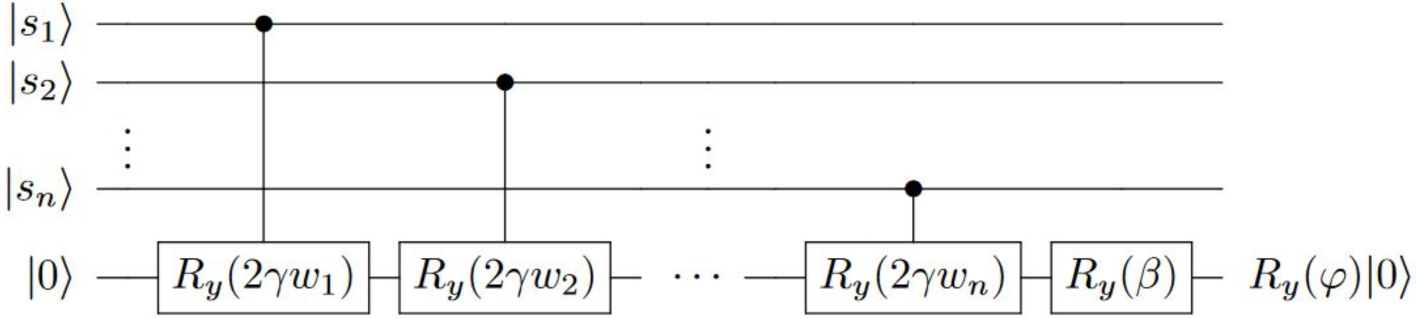
Linear Combination: Circuit

Solution:

- Rotate with angle $-w_i$ by default
- In case $x_i = 1$ rotate with angle $2w_i$
    - first rotation negates the default rotation
    - second rotation is the actual rotation



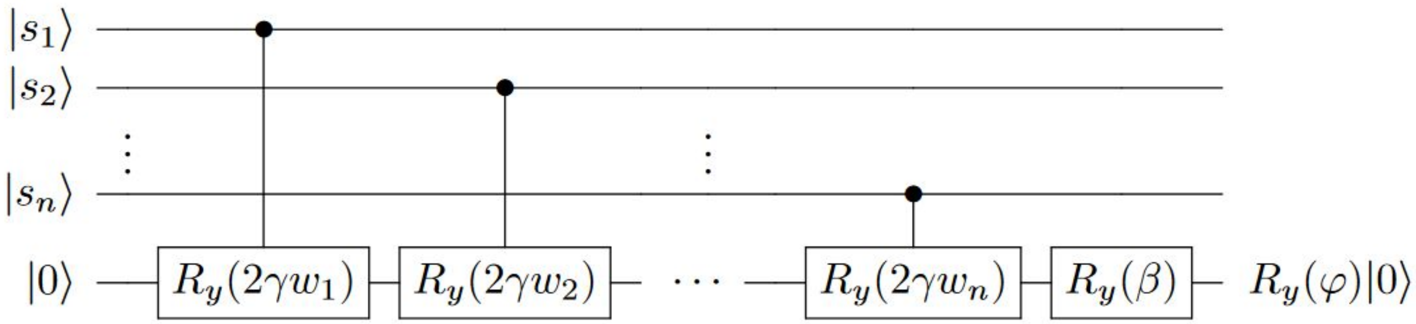
Linear Combination: Circuit



$$\gamma = \frac{\pi}{2} \cdot \frac{1}{n \cdot w_{max} + b_{max}} \qquad \beta = \frac{\pi}{2} + \gamma(b - w_1 - w_2 - \cdots - w_n)$$

(1)

Linear Combination: Circuit

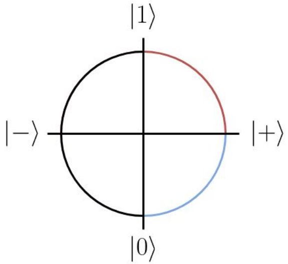


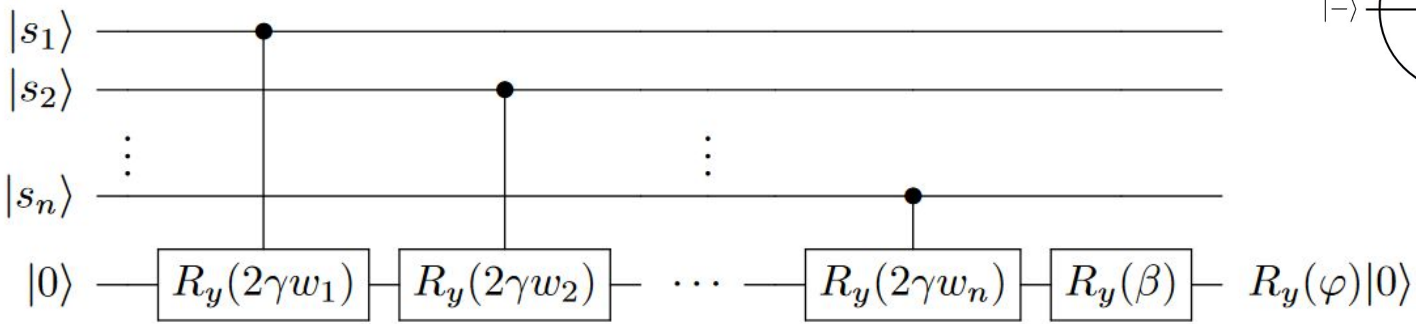
Normalization Factor

$$\gamma = \frac{\pi}{2} \cdot \frac{1}{n \cdot w_{max} + b_{max}} \qquad \beta = \frac{\pi}{2} + \gamma(b - w_1 - w_2 - \cdots - w_n)$$

(1)

Linear Combination: Circuit



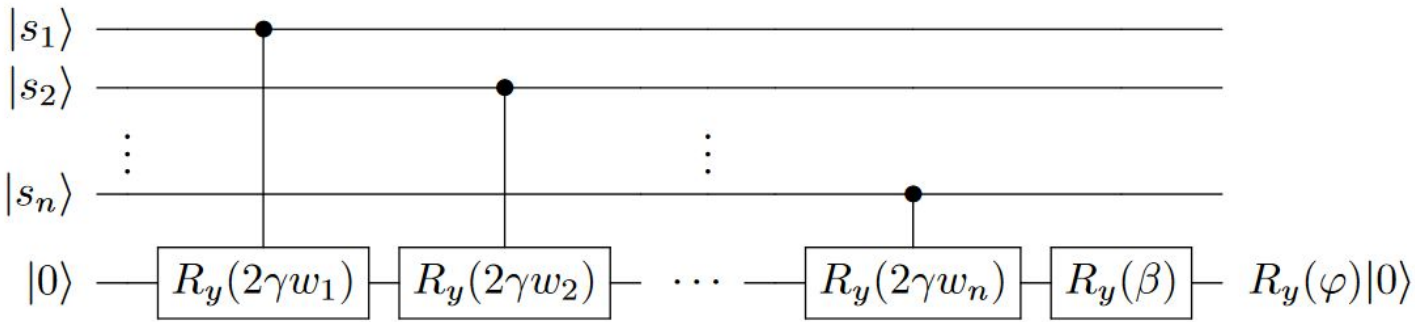


Initialize to $|+\rangle$

$$\gamma = \frac{\pi}{2} \cdot \frac{1}{n \cdot w_{max} + b_{max}} \qquad \beta = \frac{\pi}{2} + \gamma(b - w_1 - w_2 - \cdots - w_n)$$

(1)

Linear Combination: Circuit

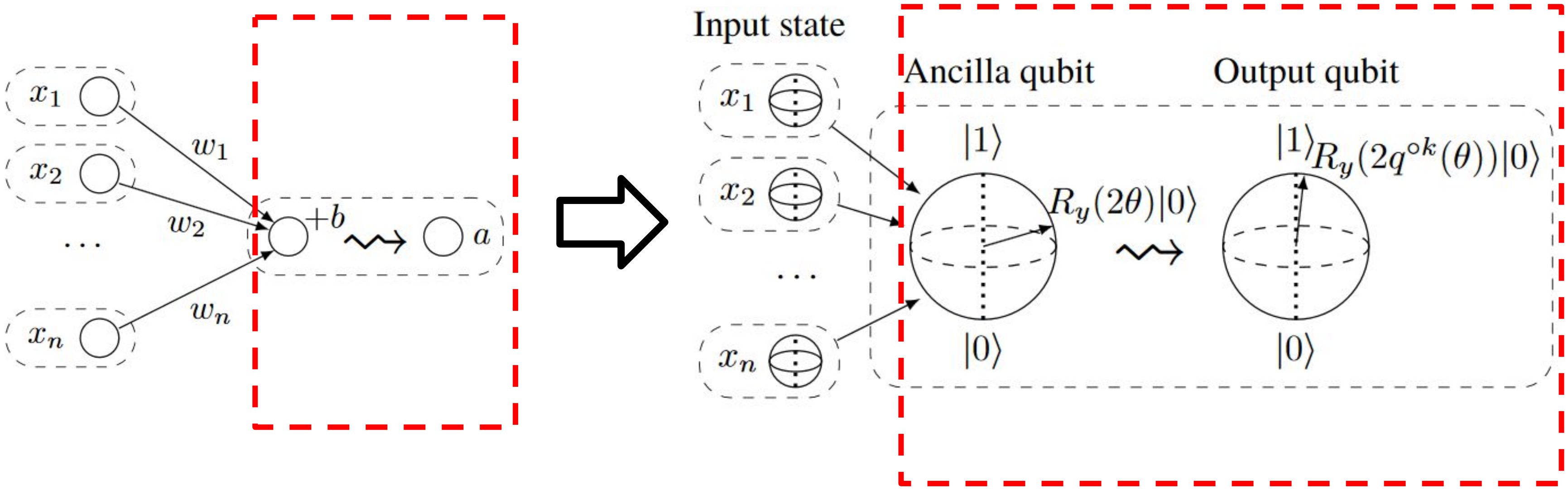


Bias + default rotations

$$\gamma = \frac{\pi}{2} \cdot \frac{1}{n \cdot w_{max} + b_{max}} \qquad \beta = \frac{\pi}{2} + \gamma(b - w_1 - w_2 - \cdots - w_n)$$

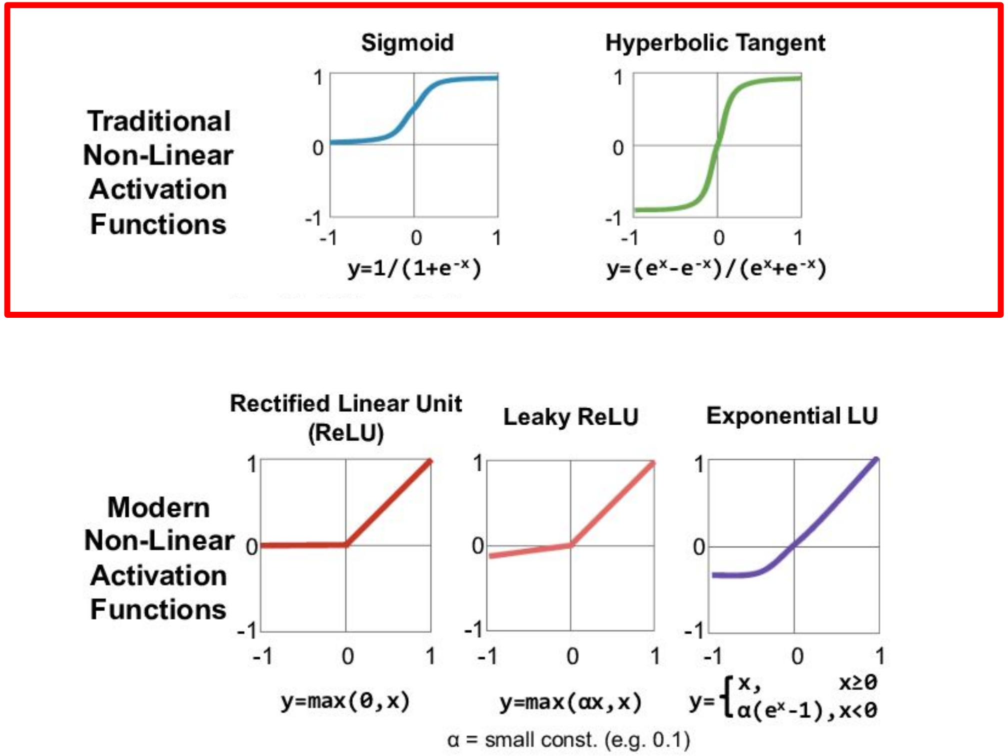
(1)

Activation Function



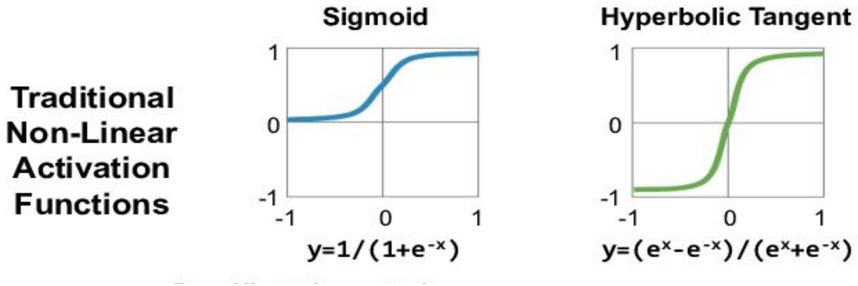
(1)

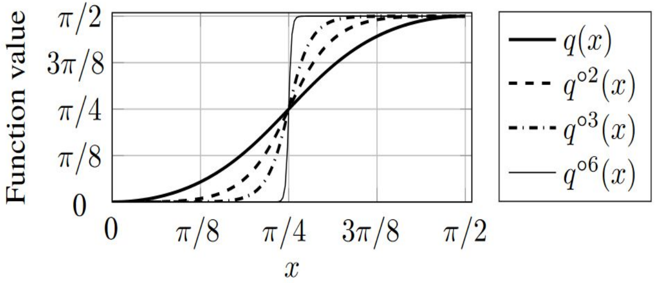
Quantum Neuron: Activation Function



https://www.researchgate.net/figure/Various-forms-of-non-linear-activation-functions-Figure-adopted-from-Caffe-Tutorial_fig3_315667264

Quantum Neuron: Activation Function



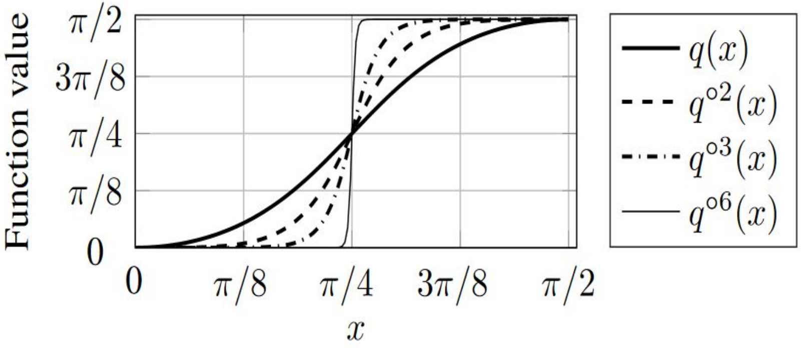


$$q(\varphi) = \arctan(\tan^2 \varphi)$$

https://www.researchgate.net/figure/Various-forms-of-non-linear-activation-functions-Figure-adopted-from-Caffe-Tutorial_fig3_315667264

Quantum Neuron: Activation Function

- Until now: Angle on **Bloch Sphere**
- Here: Actual Angle
- (0 and 1 State are orthogonal)
- Applying q multiple times ⇨ Step Function



$$q(\varphi) = \arctan(\tan^2 \varphi)$$

Activation Function: Interpretation

$$\varphi \in [0, \frac{\pi}{2}] \Longleftrightarrow |\varphi\rangle = cos(\varphi)|0\rangle + sin(\varphi)|1\rangle$$

Activation Function: Interpretation

$$\varphi \in [0, \frac{\pi}{2}] \Longleftrightarrow |\varphi\rangle = cos(\varphi)|0\rangle + sin(\varphi)|1\rangle$$

$$q(\varphi) = arctan(tan^2(\varphi))$$

$$tan^2(\varphi) = \frac{sin^2(\varphi)}{cos^2(\varphi)}$$

Activation Function: Interpretation

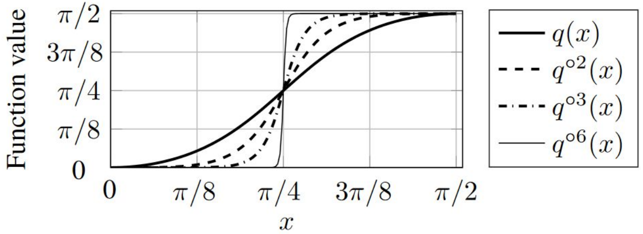
$$\varphi \in [0, \frac{\pi}{2}] \Longleftrightarrow |\varphi\rangle = cos(\varphi)|0\rangle + sin(\varphi)|1\rangle$$

$$q(\varphi) = arctan(tan^2(\varphi)) \Rightarrow |q(\varphi)\rangle = \frac{cos^2(\varphi)}{z}|0\rangle + \frac{sin^2(\varphi)}{z}|1\rangle$$

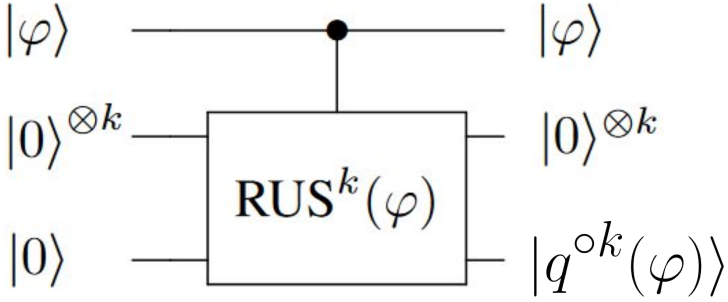
Normalization Term

(cancels out in tangens)

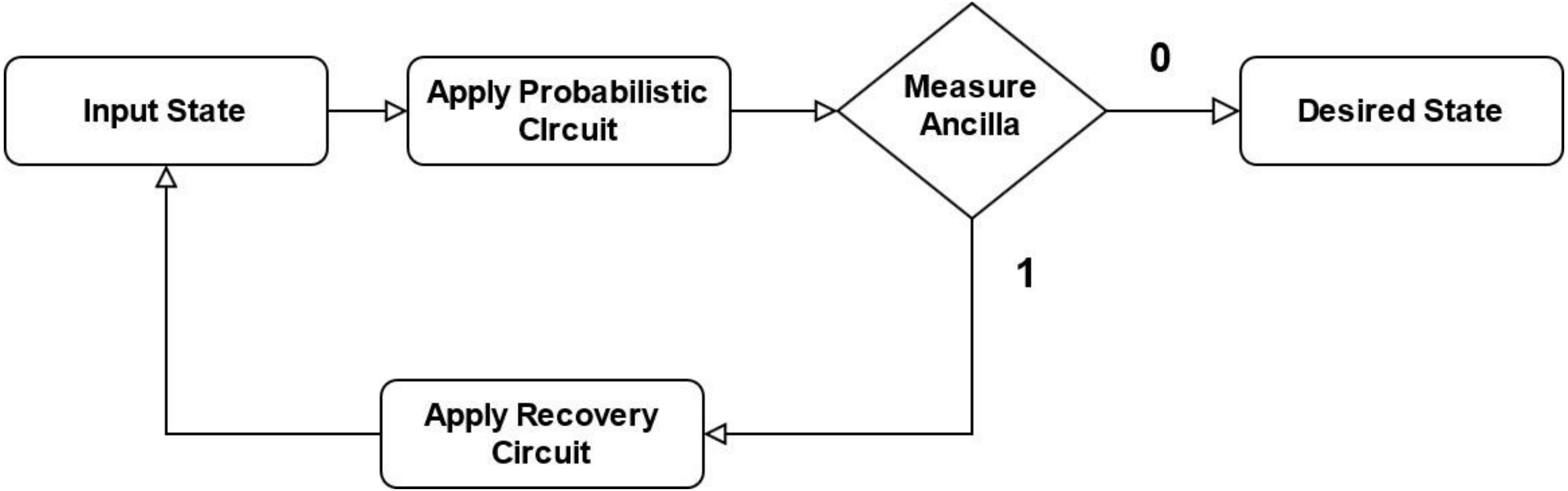
Activation Function



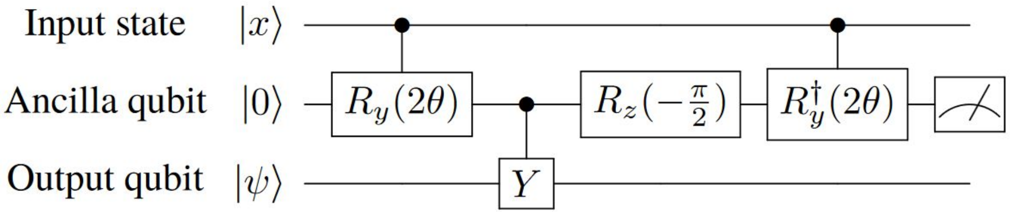
$$q(\varphi) = \arctan(\tan^2 \varphi)$$



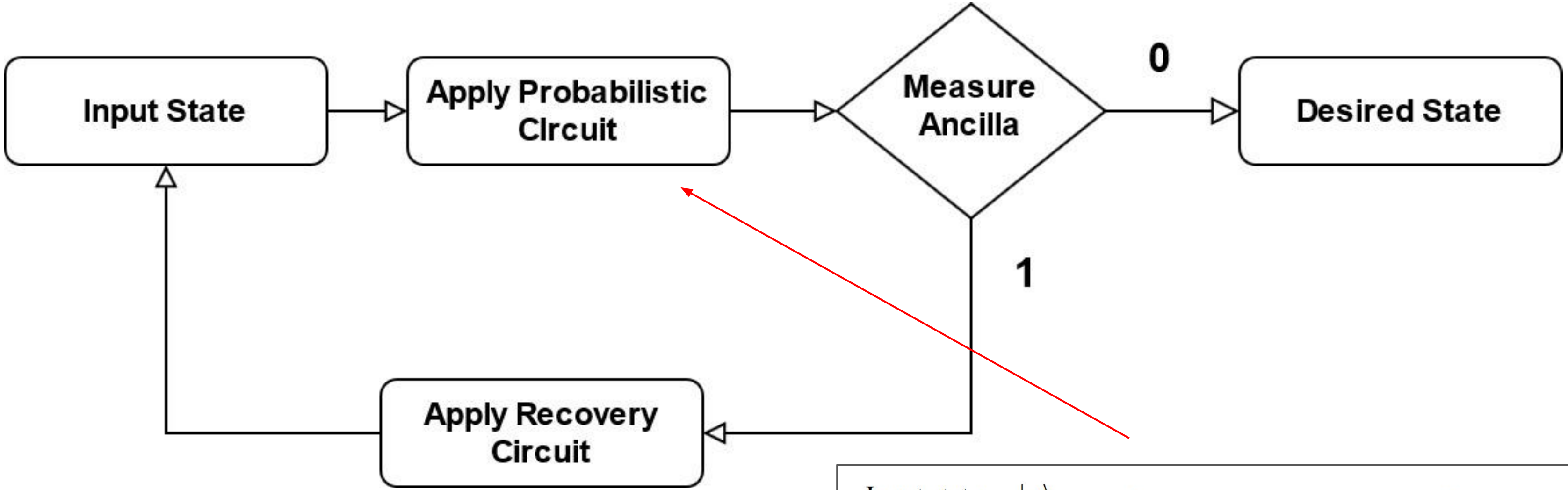
Repeat-Until-Success Circuit

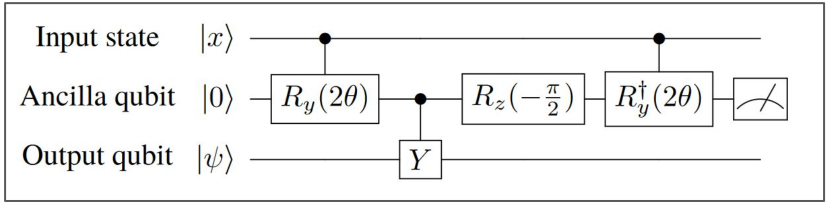


Activation Function: RUS (k=1)

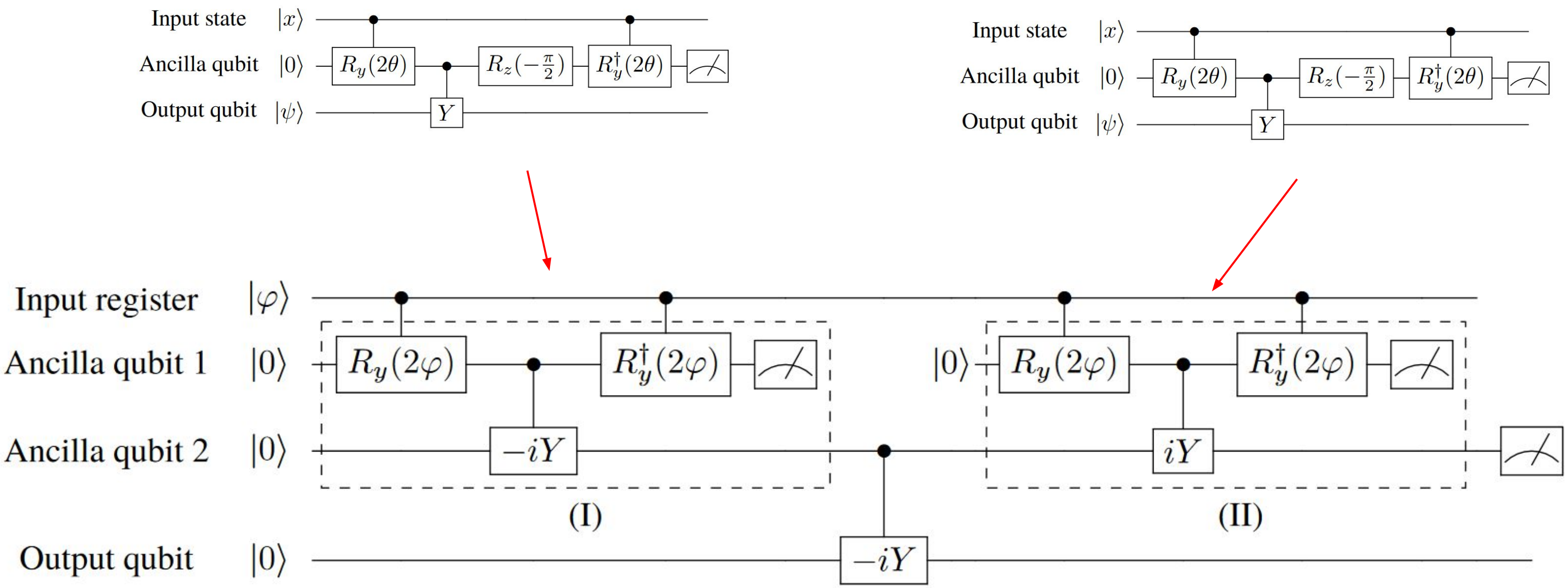


Repeat-Until-Success Circuit

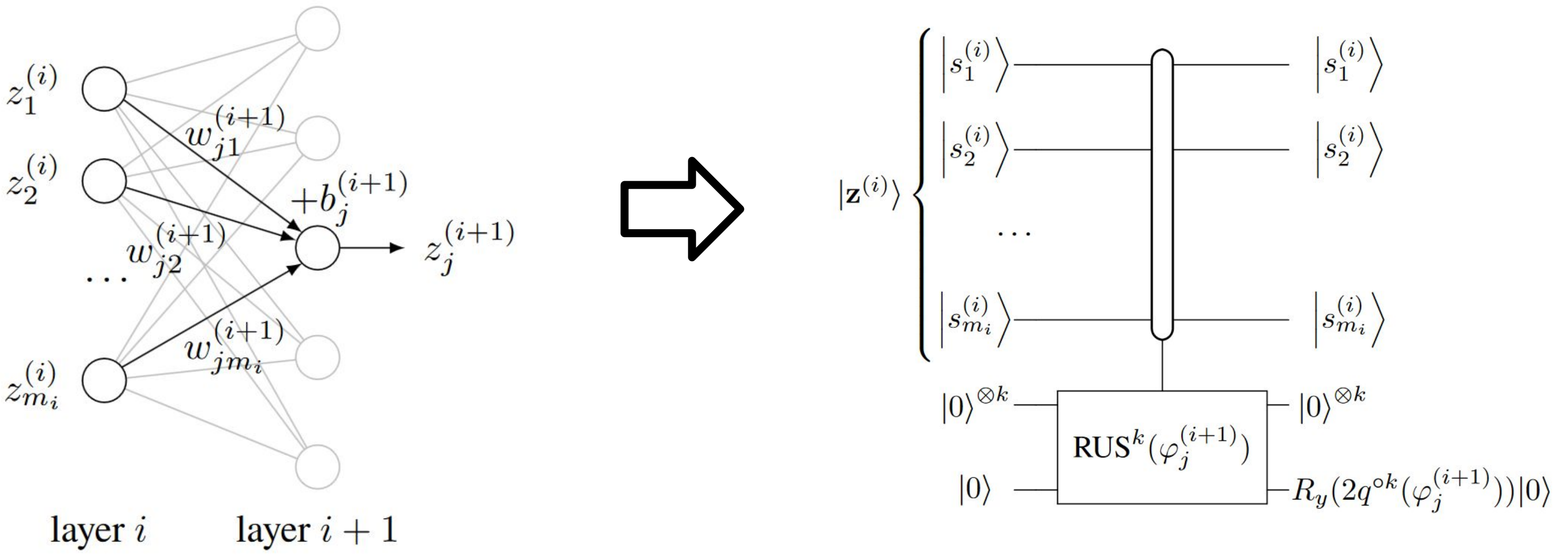




Activation Function: RUS (k=2)



From Single Neuron to Neural Network



(1)

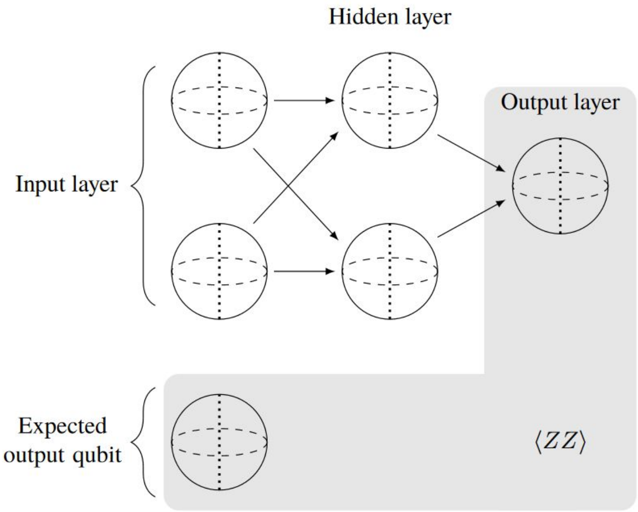
Comparing To Classical Neural Networks

- Model allows to simulate a classical neural network computation!
- But worse runtime then classically..
- And we only did forward pass...
- Entire optimization still has to be done classically….

Superposition As Input

Superposition of input data

Superposition of expected results



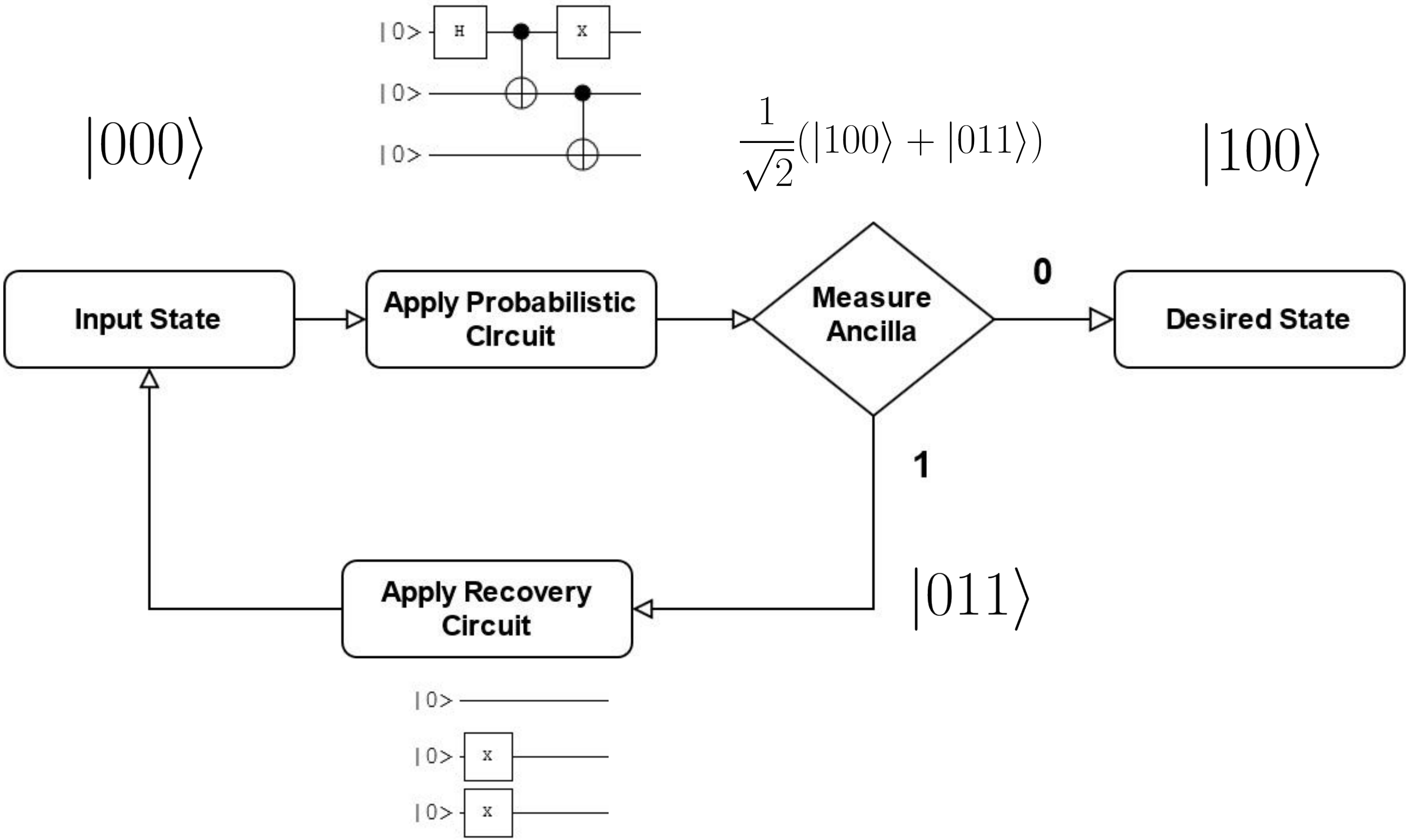
(1)

Training Of Superposition

- Training again only classically,
- Gradients??..
- gradient-free optimization used in paper
- Testing: Each data point separately
- Still: In times of “Big Data”, evaluate entire training data in one go!

Questions?

RUS Implementation Qiskit



$|000\rangle$
|0>
H
X
|0>
|0>
$\frac{1}{\sqrt{2}}(|100\rangle + |011\rangle)$
$|100\rangle$
Input State
Apply Probabilistic Circuit
Measure Ancilla
0
Desired State
1
$|011\rangle$
Apply Recovery Circuit
|0>
|0>
X
|0>
X