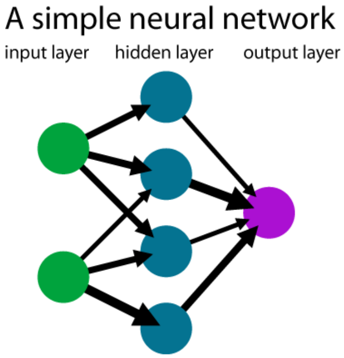
Generative Modeling using Matchgate Circuits

Peter Luo, Hong Ye Hu, Susanne F. Yelin

September 24, 2023

Neural Networks

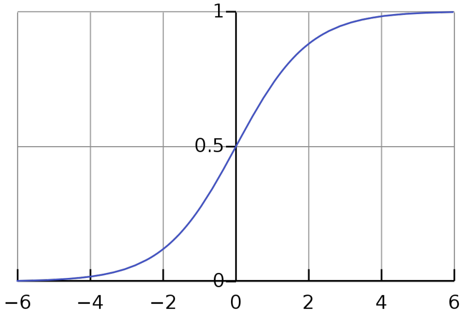
- Neural Networks are a means through which machine learning is carried out, and its architecture is inspired by neurons



- Each neuron will output

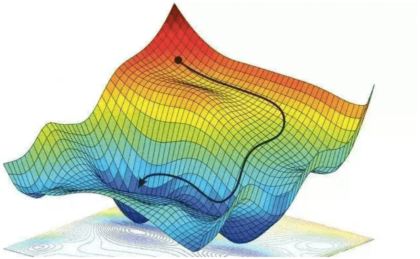
$$f\left(b+\sum w_i x_i\right),$$

where f is an "activation function", the w_i are the "weights", the b is the "bias", and the x_i being the inputs



How Neural Networks Learn

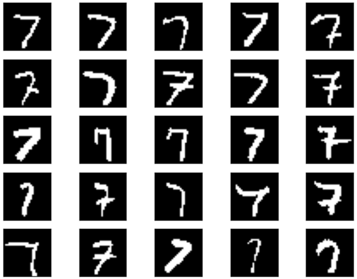
- Define a "loss function" $L(\boldsymbol{w}, \boldsymbol{b})$ which captures how close the network's output is to the desired output



- We can use optimization algorithms to adjust $\boldsymbol{w}$ and $\boldsymbol{b}$ to get closer and closer to the minimum of L

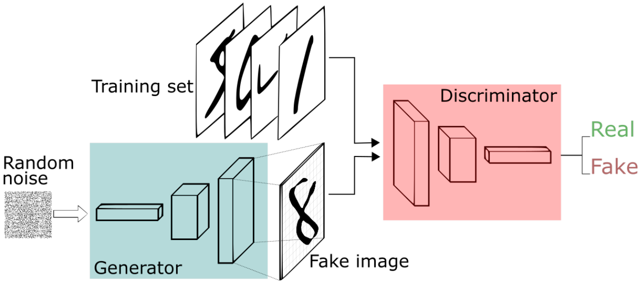
Generative Modeling

- Rather than doing a classification or prediction task, generative models strive to generate more instances of their training data
- Examples: DALL-E, Voice Generators
- Goal of this project: To use a Matchgate circuit and the MNIST dataset to generate new images of handwritten digits



Generative Adversarial Networks (GAN)

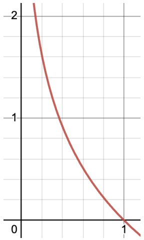
- Consists of two networks G and D with parameters $\boldsymbol{\theta}$ and ϕ and their own loss functions $L_G(\boldsymbol{\theta}, \phi)$ and $L_D(\boldsymbol{\theta}, \phi)$
- $G : \{0,1\}^{28\times 28} \to \{0,1\}^{28\times 28}$ and $D : \{0,1\}^{28\times 28} \to [0,1]$



- Assume that the real data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ comes from an underlying probability distribution p_{real}, and recall that the input to G are samples $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_n$ from p_{prior}
- The loss functions are

$$L_G(\boldsymbol{\theta}, \boldsymbol{\phi}) = - \underset{\boldsymbol{z} \sim p_{\text{prior}}}{\mathbb{E}} [\log(D(G(\boldsymbol{z})))]$$

$$L_D(\boldsymbol{\theta}, \boldsymbol{\phi}) = - \left(\underset{\boldsymbol{x} \sim p_{\text{real}}}{\mathbb{E}} [\log(D(\boldsymbol{x}))] + \underset{\boldsymbol{z} \sim p_{\text{prior}}}{\mathbb{E}} [\log(1 - D(G(\boldsymbol{z})))] \right).$$



Quantum Computing

- Leverages principles from quantum mechanics to do computations
- Consequently, it can solve certain problems exponentially faster than classical computers, reducing runtime from millions of years to just a few minutes
- It can also break some commonly used encryption methods, such as RSA or ECC
- It can simulate molecular interactions efficiently, which has implications in drug discovery and the material sciences

Limitations

- Quantum computers need to be kept at very cold temperatures, thus requiring a large amount of energy
- Real quantum devices are noisy
- Number of qubits that we're able to run on the best quantum computers is relatively small (on the order of 10^2)
- A lot is known about what is theoretically possible, but not a lot is known about their actual performance
- Temporary solution: classical simulation

Quantum Computing

Classical computing:

- Consists of logic gates (e.g. AND, NOT, OR)
- One classical bit: a 0 or a 1

Quantum computing:

- Consists of quantum gates, represented by unitary matrices with complex entries
- One quantum bit: a vector of two complex numbers, which describes the probability of being measured in the 0 or 1 state

Quantum Bits

- One qubit looks like

$$a\,|0\rangle + b\,|1\rangle \to \begin{pmatrix} a \\ b \end{pmatrix}$$

- $|a|^2$ and $|b|^2$ are the probabilities so we require $|a|^2 + |b|^2 = 1$
- For two qubits,

$$a\,|00\rangle + b\,|01\rangle + c\,|10\rangle + d\,|11\rangle \to \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

- The number of complex numbers we need to keep track of scales exponentially with the number of qubits
- Quantum computers are hard to simulate classically

Quantum Gates

- An n-qubit gate is a $2^n \times 2^n$ unitary matrix with complex entries
- A matrix U is unitary if $UU^\dagger = I$, where $\dagger$ means the conjugate transpose.
- 1-qubit examples:

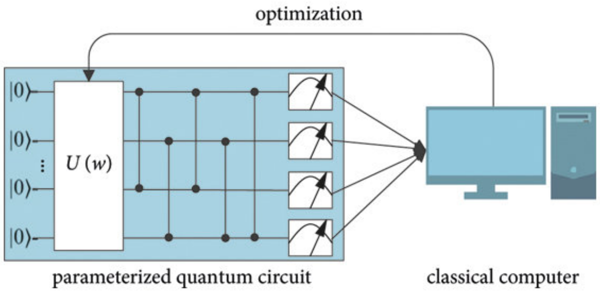
$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \qquad \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$$

- Example:

$$H\left|0\right\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix} \to \frac{1}{\sqrt{2}}\left|0\right\rangle + \frac{1}{\sqrt{2}}\left|1\right\rangle$$

Parametrized Quantum Circuit

- Quantum gates can be parametrized
- By replacing the neural network with a parametrized quantum circuit and then measuring its output, we obtain a hybrid quantum-classical system compatible with machine learning procedures

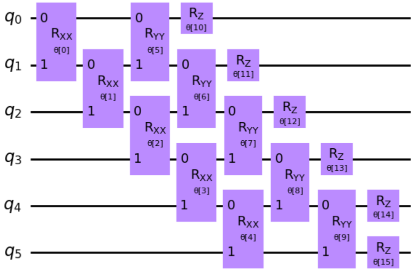


Matchgates

- Matchgates are a restricted class of parametrized quantum gates, generated by

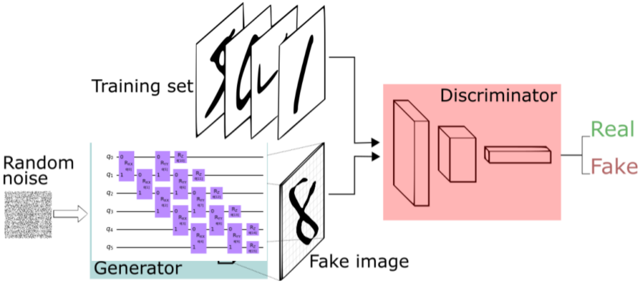
$$e^{-iX\otimes X\theta/2}, \quad e^{-iY\otimes Y\theta/2}, \quad e^{-iX\otimes Y\theta/2}, \quad e^{-iY\otimes X\theta/2}, \quad e^{-iZ\theta/2}$$

- Matchgates are differentiable with respect to θ and can be simulated in polynomial time



Setup

1. Use a parametrized quantum circuit made of matchgates as our G
2. Use a classical neural network as D
3. Train it just like you would train a GAN, i.e. use the same loss functions



- Goal: Understand the performance of QML on a real dataset with a large number of qubits

Acknowledgements

- Thanks to Hong-Ye Hu and Susanne Yelin
- Thanks to other Yelin group undergrads for keeping me sane
- Thanks to PRISE

Questions?