Simulating Recurrent Neural Networks on Graphic Processing Units

Summer Project

Zhangsheng Lai
Advisor: Shaowei Lin

October 4, 2017

Introduction

Introduction

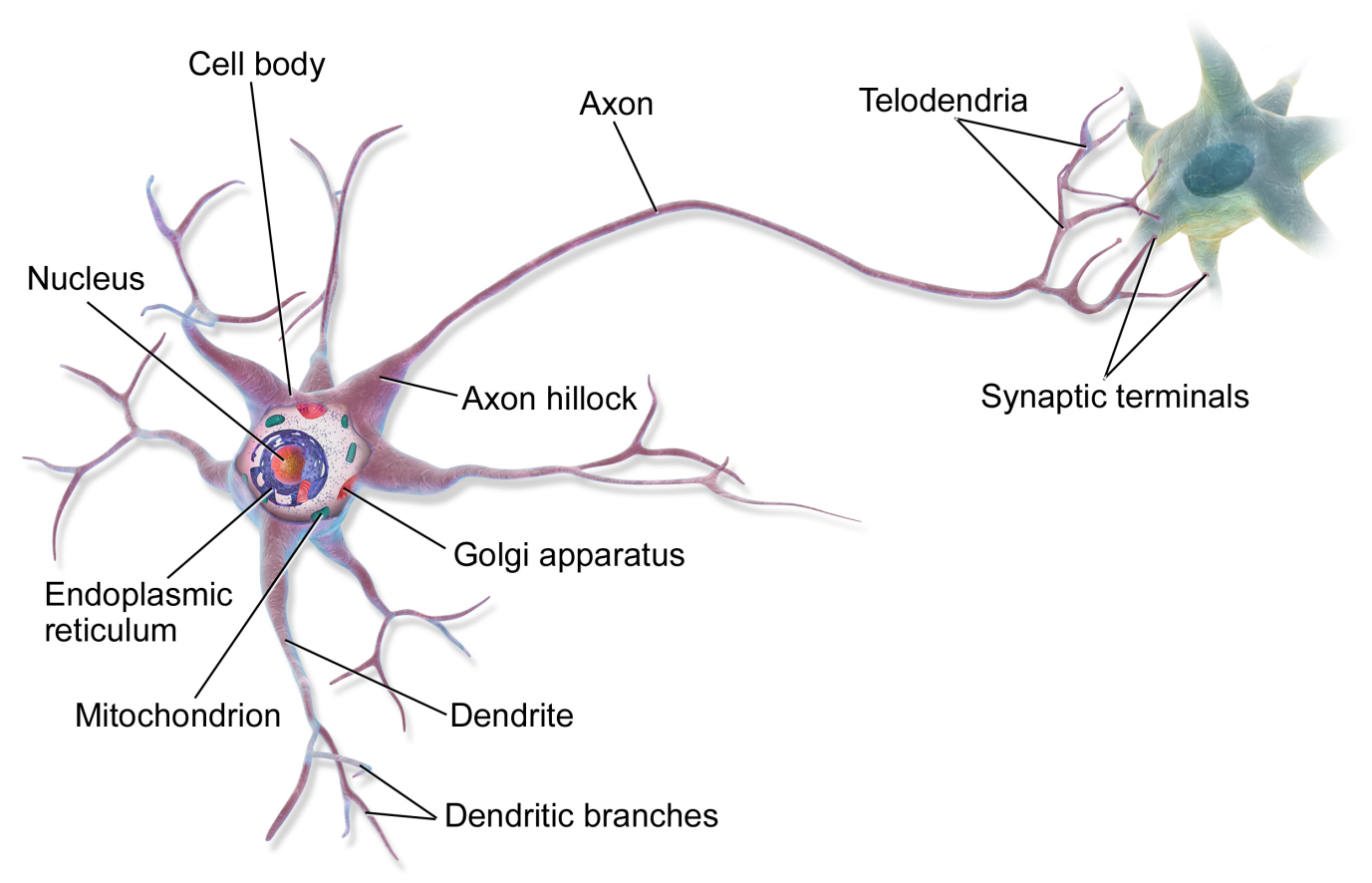


Figure 1: Anatomy of a neuron[1]

[1] By BruceBlaus - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=28761830

Feedforward Neural Network

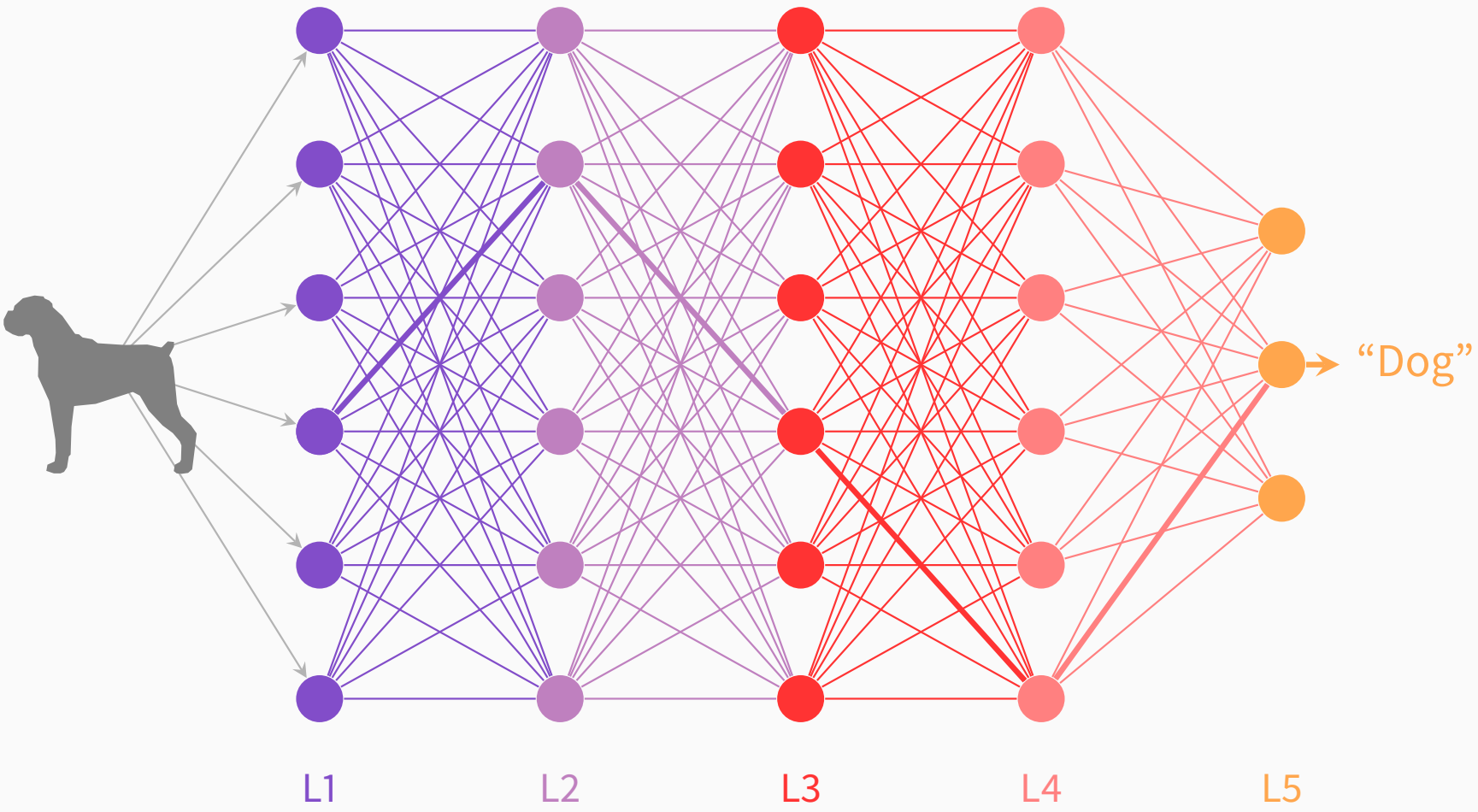
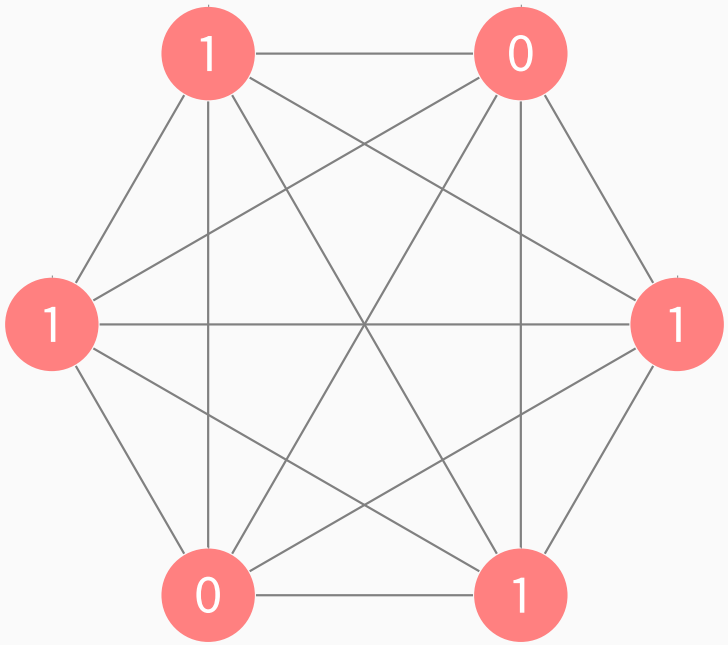


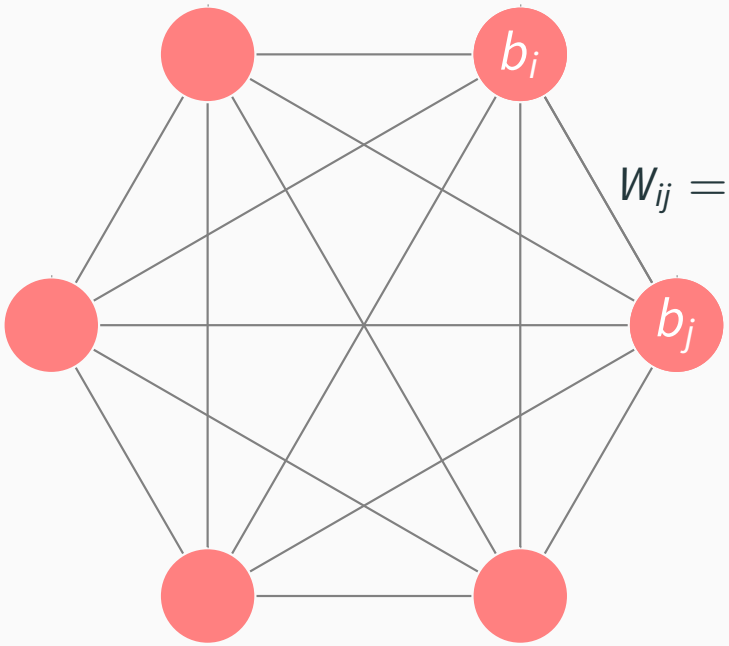
Figure 2: Feedforward Neural Network

Recurrent Neural Networks

Boltzmann Machines



Boltzmann Machines



Energy configuration, $E = -\sum_{i<j} W_{ij} x_i x_j - \sum_i b_i x_i$

Energy gap, $\Delta E_i = E(x_i = 0) - E(x_i = 1) = \sum_j W_{ij} x_j + b_i$

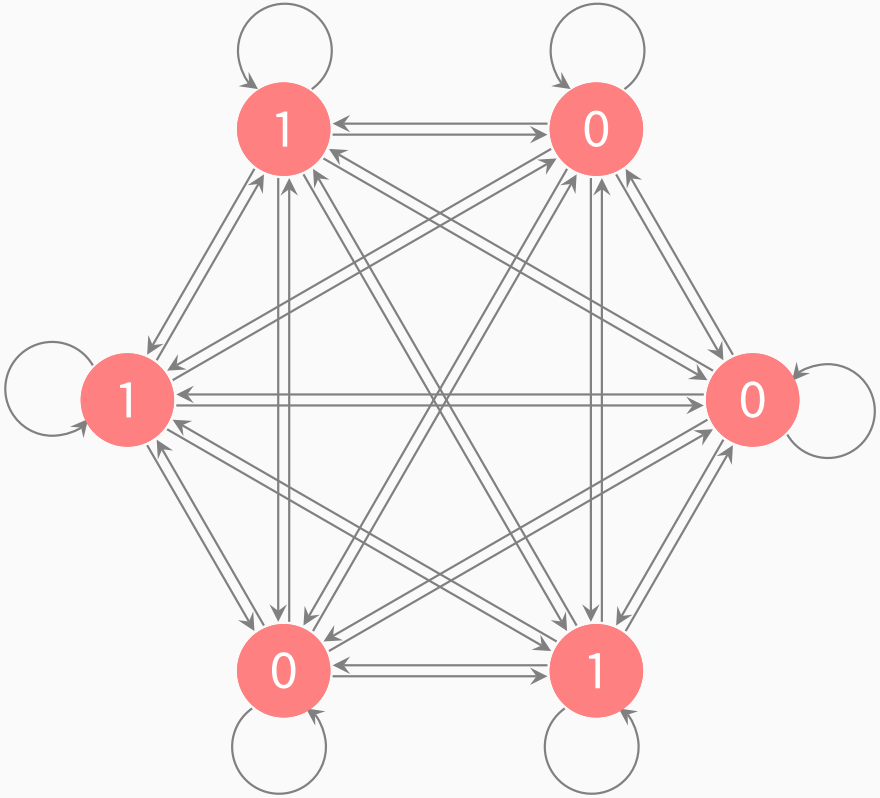
$$p_i := \mathbb{P}(x_i = 1) = \frac{1}{1 + e^{-\Delta E_i / \tau}}$$

Algorithm 1 Boltzmann Machine Simulation.

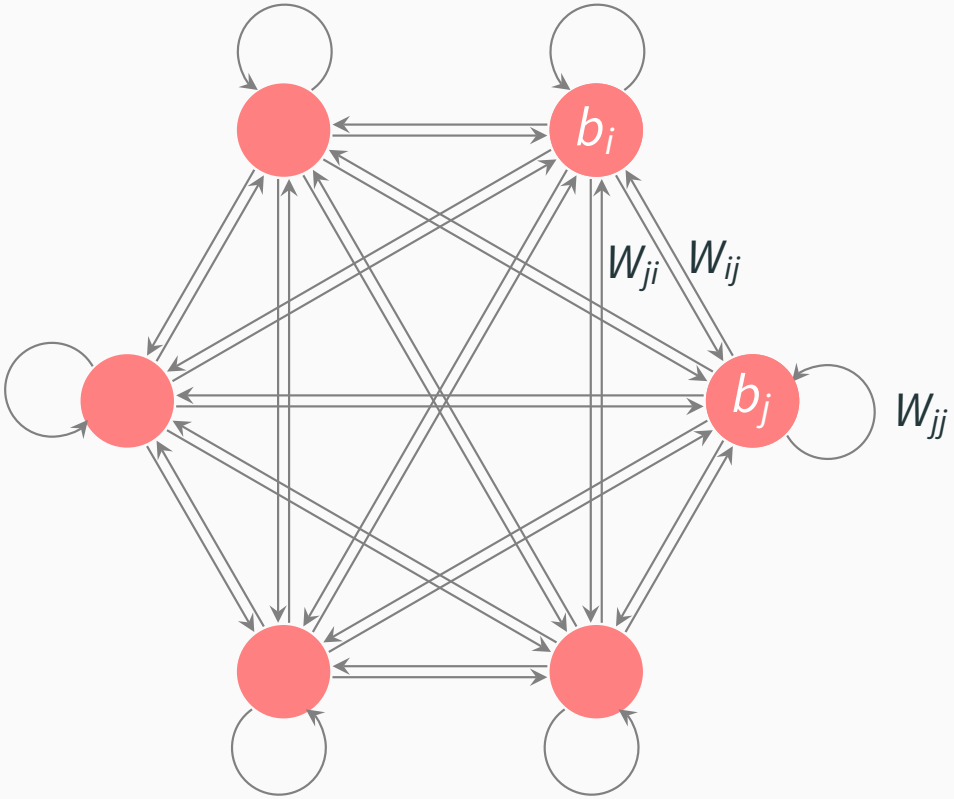
1: Initialize $\boldsymbol{W}, \boldsymbol{b}$
2: Initialize $\boldsymbol{x}^{(0)}$
3: **for** i from 1 to N **do**
4: Random $k \in \{1, \ldots, d\}$, where d is the number of neurons
5: Compute $p_k = \frac{1}{1+e^{-\Delta E_k / \tau}}$
6: Sample from the Bernoulli distribution with $p = p_k$
7: $\boldsymbol{x}^{(i)} \leftarrow \text{update}(\boldsymbol{x}^{(i-1)})$
8: **end for**

where update($x^{(i-1)}$) updates the chosen neuron k with the outcome of the sample from the Bernoulli distribution.

McCulloch-Pitts Machines



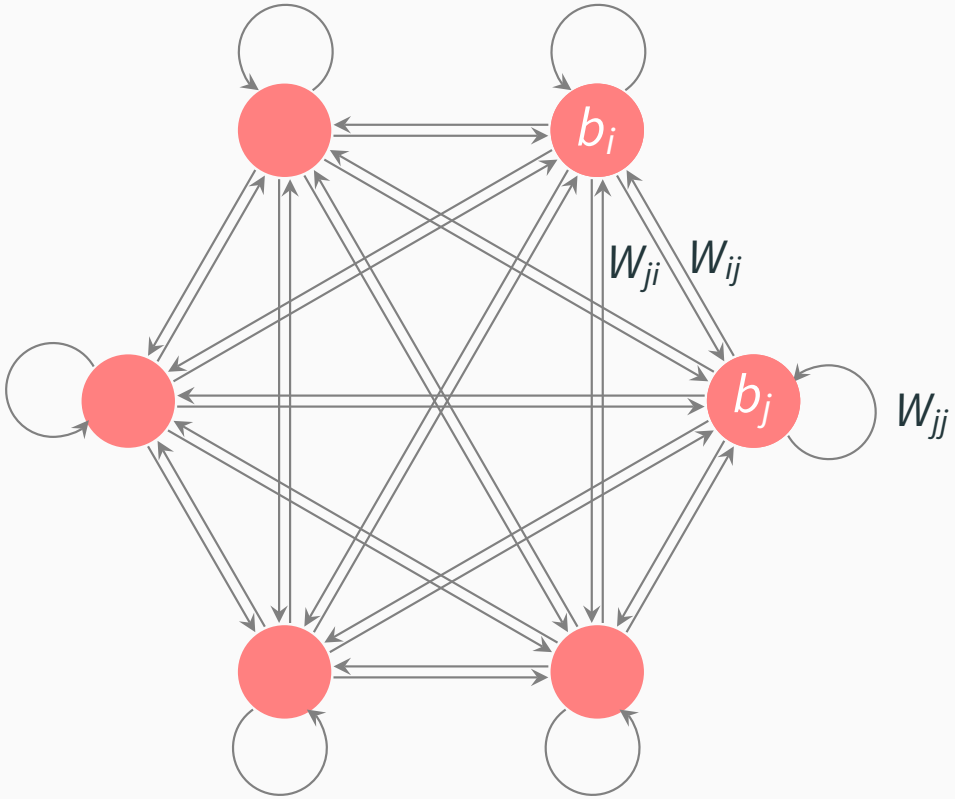
McCulloch-Pitts Machines



$$\text{Transition Energy, } E(y, x|\theta) = -\sum_{ji \in E} W_{ji} y_j x_i - \sum_{j \in V} b_j x_j - \sum_{i \in V} b_i x_i$$

$$\Gamma_{yx} = \exp\left(-\frac{1}{2\tau} E(y, x|\theta) + \frac{1}{2\tau} E(x, x|\theta)\right)$$

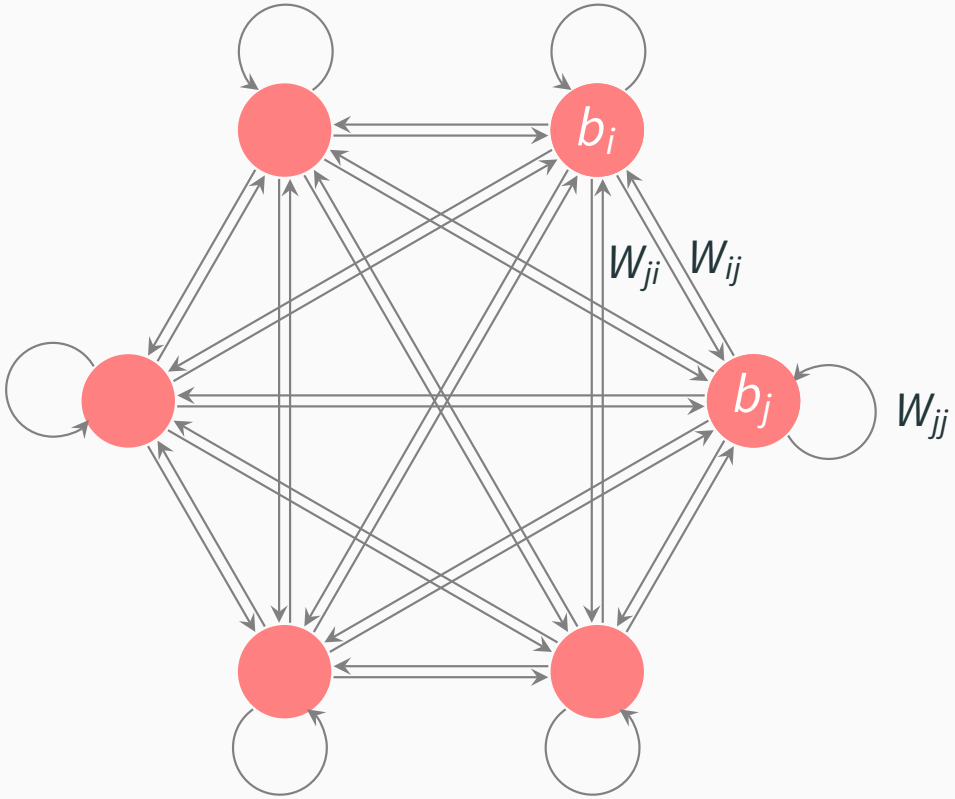
McCulloch-Pitts Machines



$$\lambda_j := \Gamma_{yx} = \exp\left(\frac{1}{2\tau} s_j z_j\right)$$

where $s_j = 1 - 2x_j$, $z_j = \sum_j W_{ji} x_i + b_j$ and x, y differ by the jth unit.

McCulloch-Pitts Machines



Transition probability from x to y, $p_{yx} = \dfrac{\lambda_j}{\sum_{j'} \lambda_{j'}}$

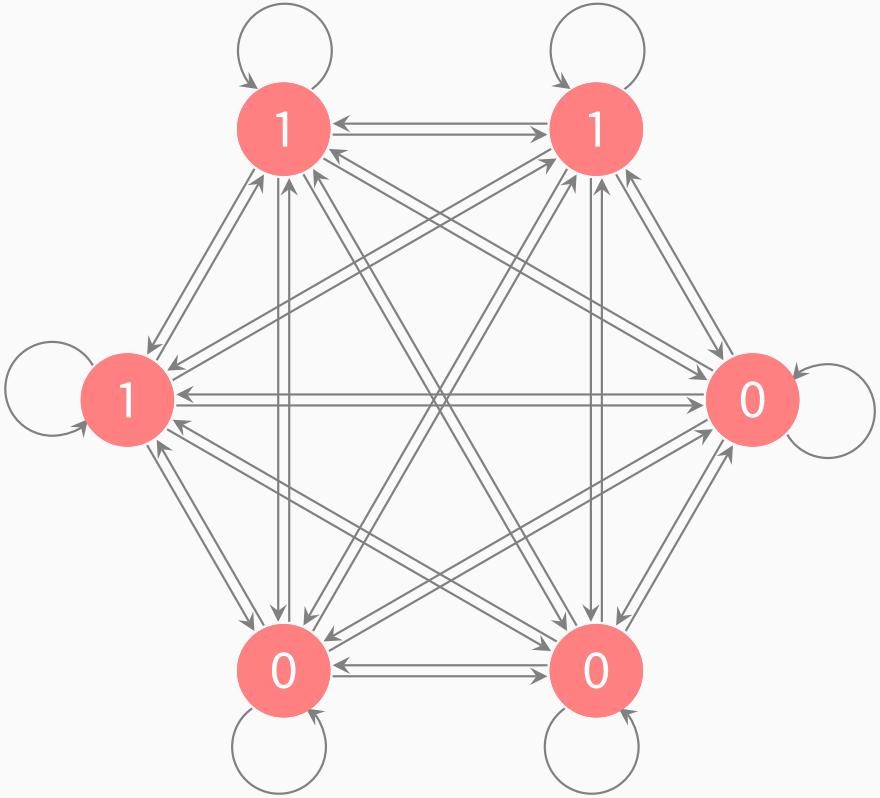
Sample holding times, $T_{yx} \sim Exp(a_x)$, where $a_x = \sum_j \lambda_j$

Algorithm 2 CTMC Simulation.

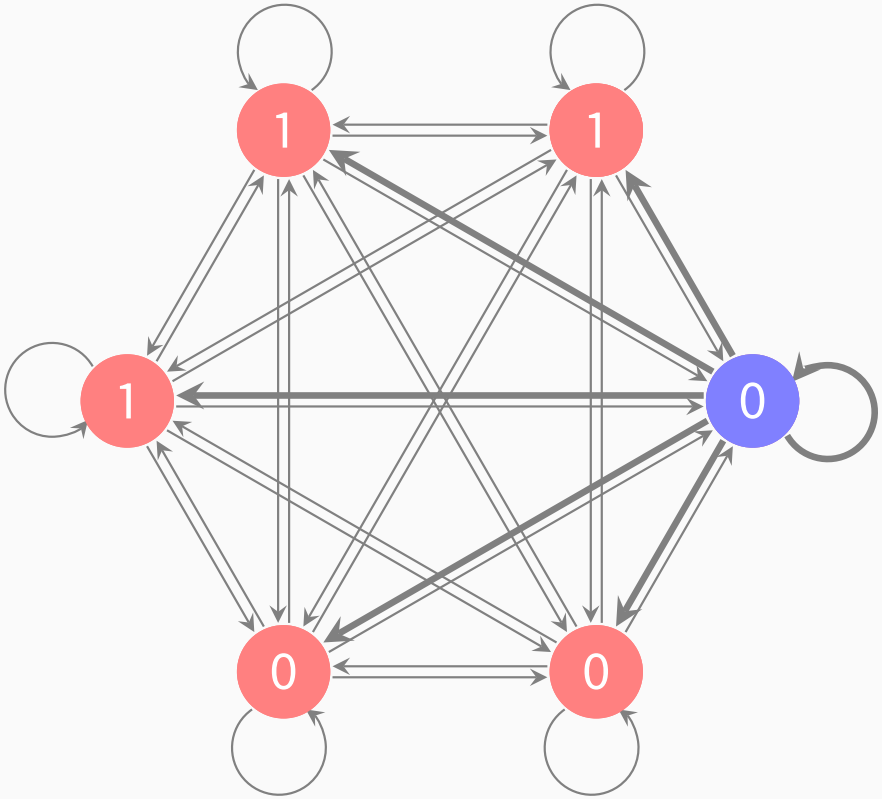
1: Initialize $\boldsymbol{W}, \boldsymbol{b}$
2: Initialize $\boldsymbol{x}^{(0)}$
3: **for** i from 1 to N **do**
4: Compute Γ_{yx}, p_{yx} for each $\boldsymbol{y} \neq \boldsymbol{x}$
5: Compute $a_x = \sum \Gamma_{yx}$
6: $\boldsymbol{x}^{(i)} \leftarrow \text{flip}(\boldsymbol{x}^{(i-1)})$
7: Sample holding time $T_i \sim \text{Exp}(a_x)$
8: **end for**

where $\text{flip}(x^{(i-1)})$ flips the state of the transiting neuron.

McCulloch-Pitts Machines

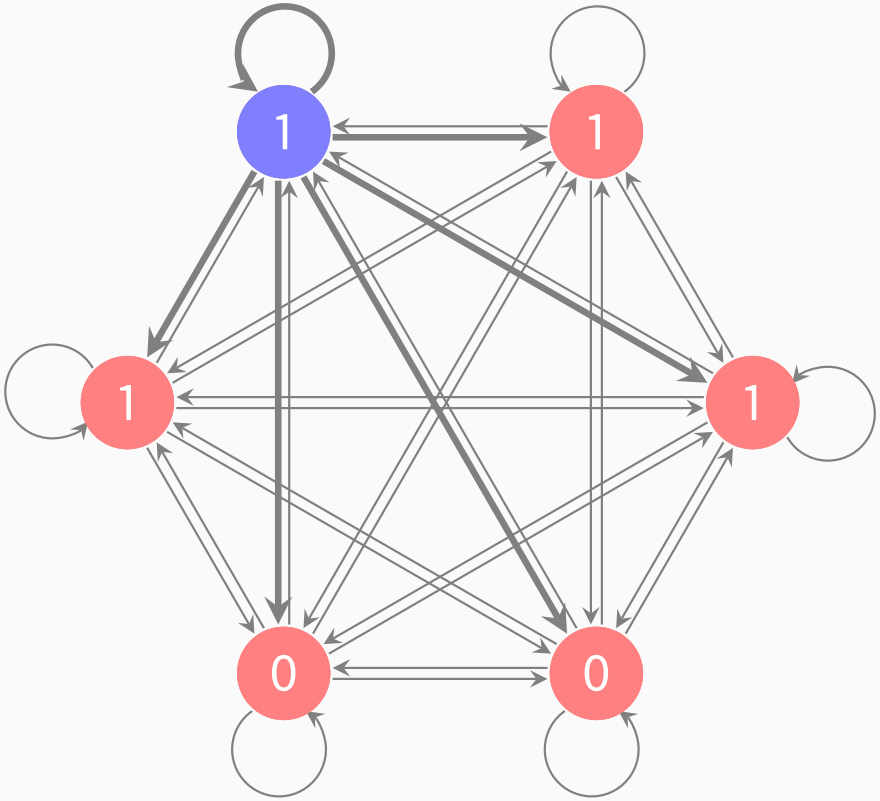


McCulloch-Pitts Machines



$(T_0, (1, 0, 0, 0, 1, 1))$

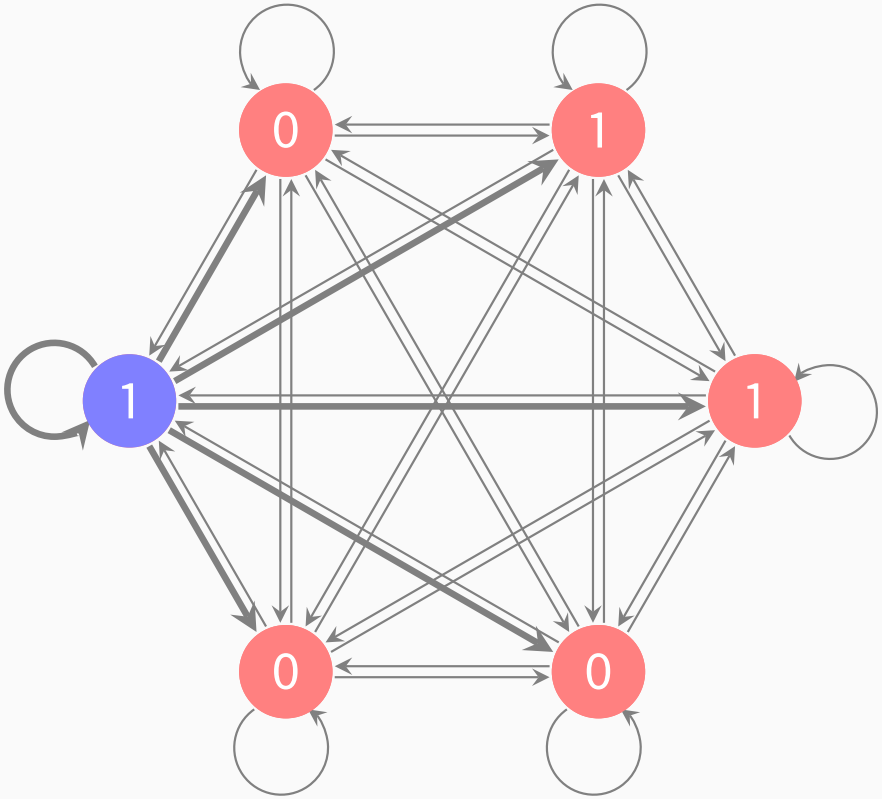
McCulloch-Pitts Machines



$(T_0, (1, 0, 0, 0, 1, 1))$

$(T_1, (1, 0, 0, 1, 1, 1))$

McCulloch-Pitts Machines



$$(T_0, (1, 0, 0, 0, 1, 1))$$

$$(T_1, (1, 0, 0, 1, 1, 1))$$

$$(T_2, (1, 0, 0, 1, 1, 0))$$

Simulating on GPUs

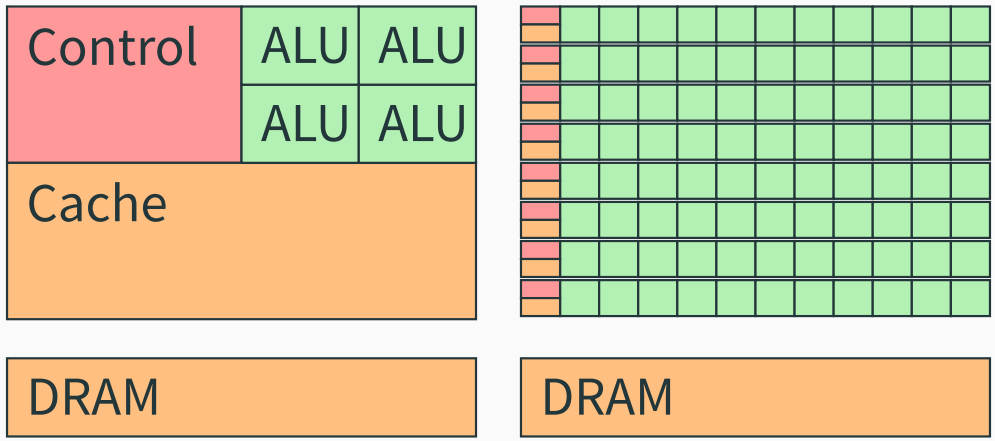
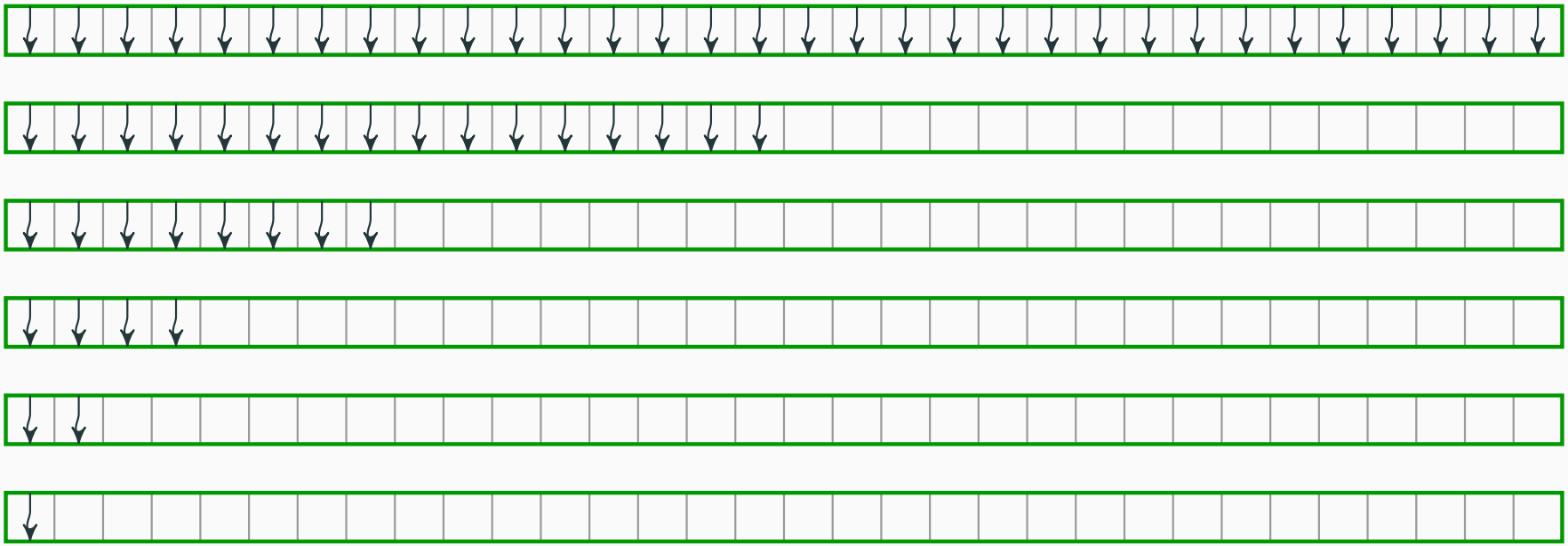


Figure 3: Comparison between the amount of transistors devoted to different functions inside a CPU and a GPU.

GPU Algorithm: Reduction



Reduction

```
mod = SourceModule("""
        __global__ void reduce_kernel(float *d_out, float *d_in)
        {
            int myId = threadIdx.x + blockDim.x * blockIdx.x;
            int tid = threadIdx.x;
            // do reduction in global memory
            for (unsigned int s = blockDim.x / 2; s > 0; s >>= 1)
            {
                if (tid < s)
                {
                    d_in[myId] += d_in[myId + s];
                }
                __syncthreads(); // make sure all adds at one stage are
                    done
            }
            // only thread 0 writes result for this block back to global
                memory
            if (tid == 0)
            {
                d_out[blockIdx.x] = d_in[myId];
            }
        }
        """, arch='sm_60')
```

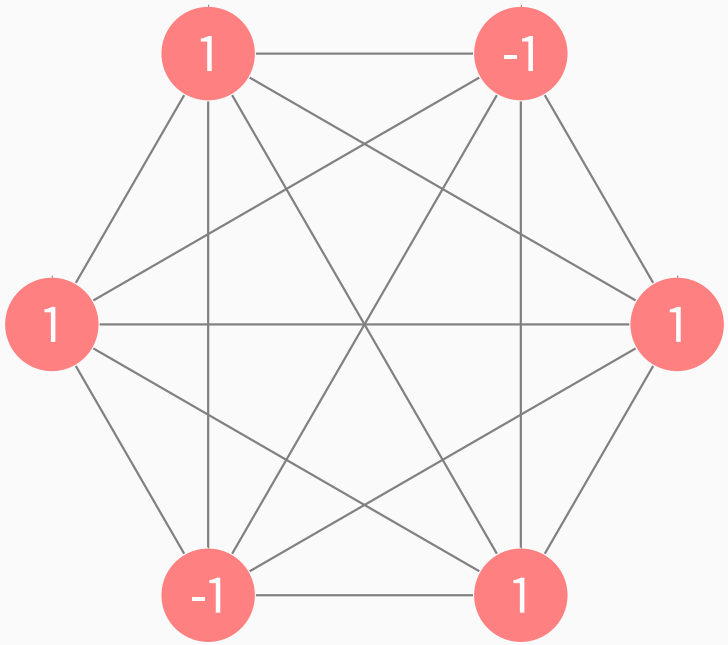
Importance to Simulating on GPUs

- Faster matrix multiplication
- Larger neural networks
- Larger function space
- Energy efficiency

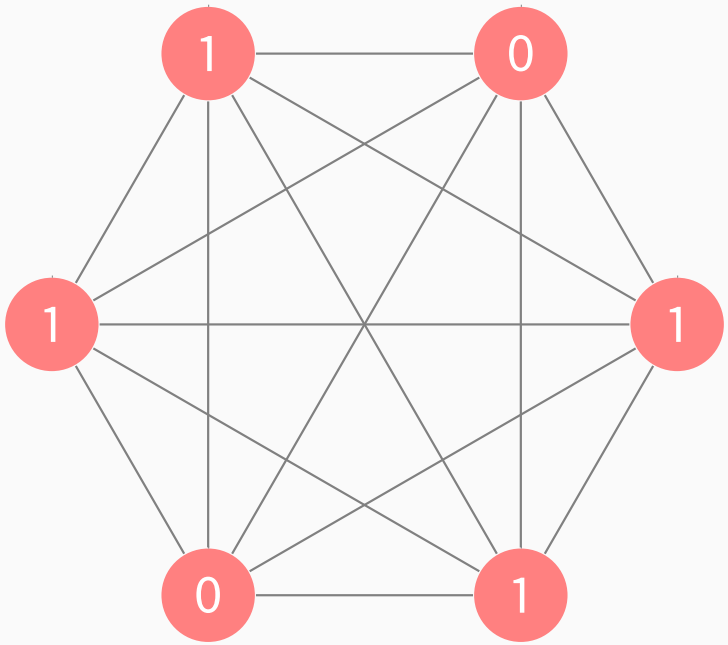
References

- 2010 IEEE Sensor Array and Multichannel Signal Processing Workshop, 41-44
  Modeling neuron firing pattern using a two state Markov chain.
  http://ieeexplore.ieee.org/document/5606761/
- CSC321: Introduction to Neural Networks and machine Learning
  Hopfield nets and simulated annealing.
  https://www.cs.toronto.edu/ hinton/csc321/notes/lec16.pdf
- CSC321: Introduction to Neural Networks and machine Learning
  Boltzmann Machines as Probabilistic Models.
  https://www.cs.toronto.edu/ hinton/csc321/notes/lec17.pdf
- CUDA C Programming Guide
  From Graphics Processing to General Purpose Parallel Computing.
  http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html
- S. Lin.
  Biological Plausible Deep Learning for Recurrent Spiking Neural Networks

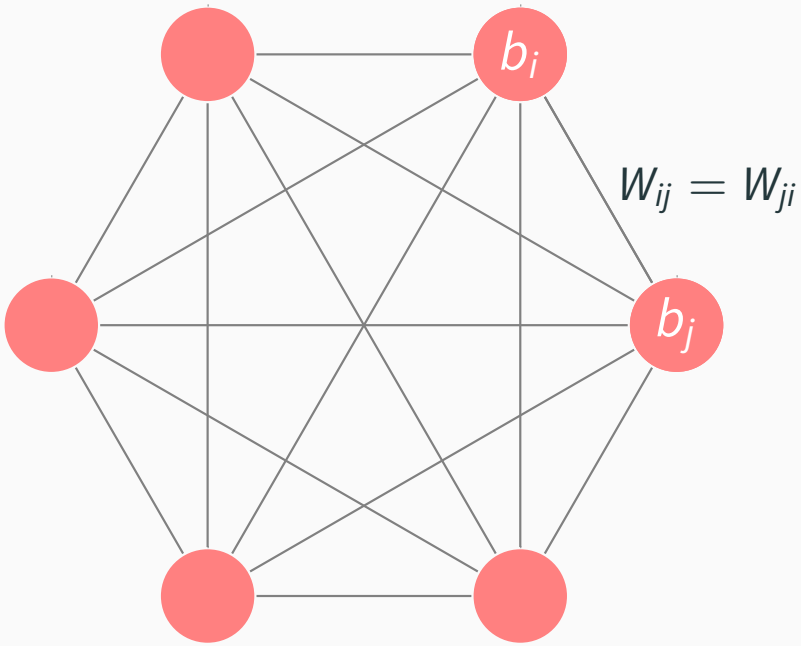
Hopfield Networks



Hopfield Networks



Hopfield Networks

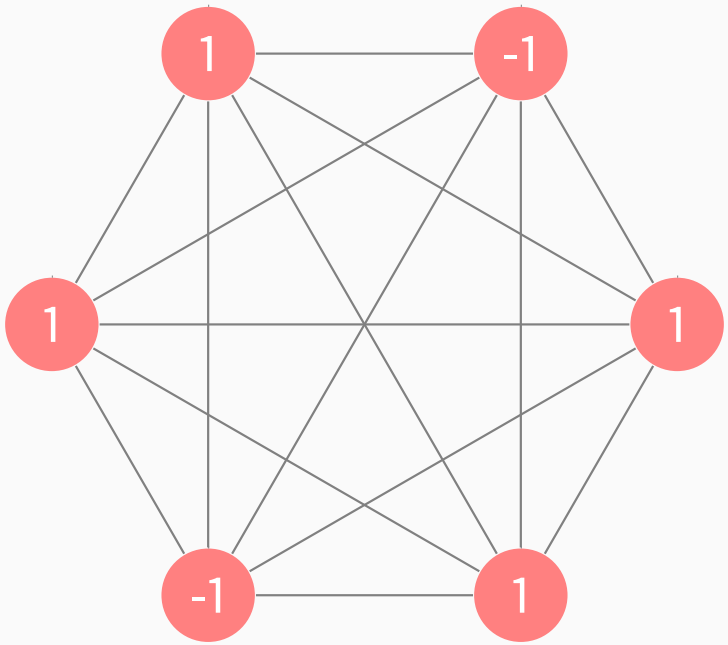


$$\text{Energy configuration, } E = -\sum_{i<j} W_{ij} x_i x_j - \sum_i b_i x_i$$

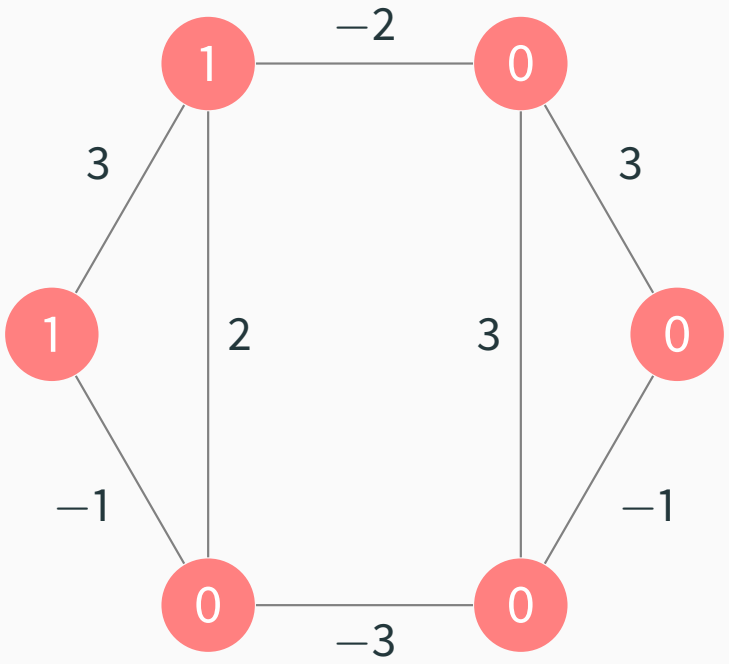
$$\text{Energy gap, } \Delta E_i = E(x_i = 0) - E(x_i = 1) = \sum_j W_{ij} x_j + b_i$$

$$\text{Update rule, } x_i := \begin{cases} 1 & \sum_j W_{ij} x_j + b_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Hopfield Networks

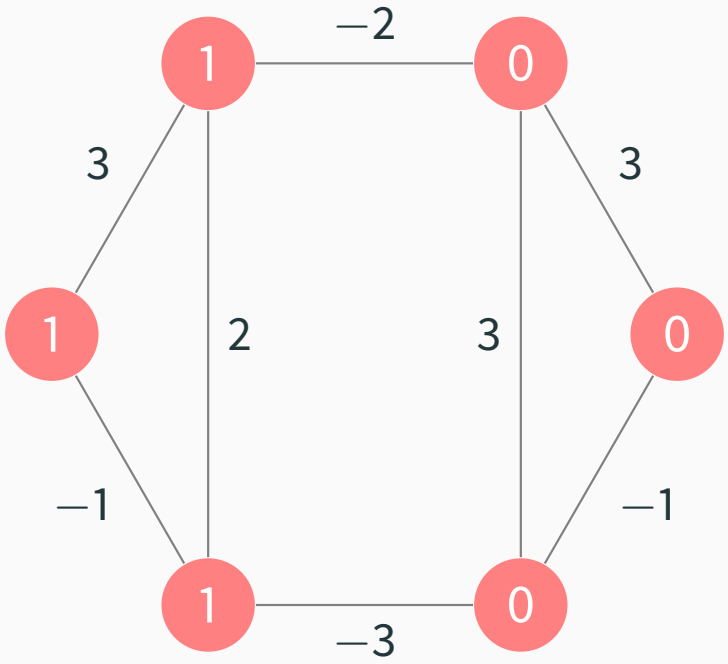


Hopfield Networks



(1, 0, 0, 0, 0, 1)

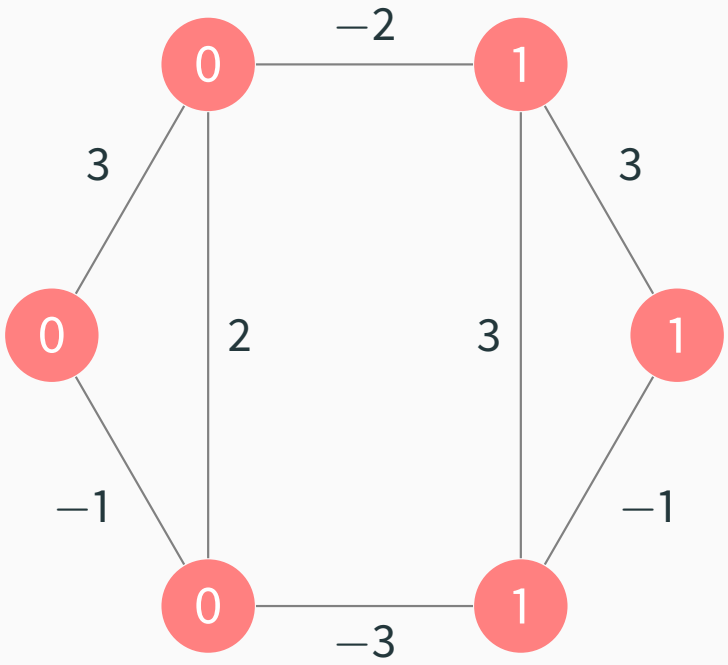
Hopfield Networks



(1, 0, 0, 0, 0, 1)

(1, 1, 0, 0, 0, 1)

Hopfield Networks



(1, 0, 0, 0, 0, 1)

(1, 1, 0, 0, 0, 1)

(0, 0, 1, 1, 1, 0)