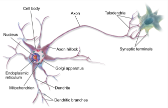


Figure 1: Anatomy of a neuron[1]

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

- the features that define a neuron are electrical excitability, where a neuron spikes and discharge electrical signals through the synapses, which are complex membrane junctions that transmit signals to other neurons
- there are approximately 10^{14} neurons in the human brain
- artifical neuron networks are inspired by these biological neurons

Feedforward Neural Network

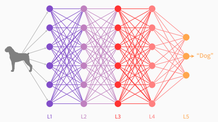
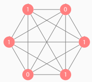


Figure 2: Feedforward Neural Network

- for example we have feedforward neural networks where connections between the units do not form a cycle
- we have managed to use feedforward neural networks, to classify images very well
- however the connections between the neurons in our brain are much more complex than those in the feedforward neural networks
- however, the methodology used to do classification is based on learning parameters of the model and then do matrix multiplication to obtain a probability of it being classified as a particular class.

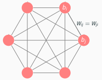
- recurrent neural networks are artificial neural network where connections between units form a directed cycle.
- these neural networks are the more popular and mainstream ones, but today we are going to look at RNNs and how to simulate them.
- one of the more popular RNN is Long short term memory (LSTM), and they are able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame, but the neurons in LSTMs communicate with real values, which is different from the way neurons communicate in our brain
- we will look at some RNNs where their architecture is closer to our brains and by building such neural networks with the neurons matching the number of neurons in the brain, we hope to possibly arrive at some learning theories that is close to how learning is done in the brain, if not as good as the brain
- to construct such a big network of neurons, we have to rely on hardware that are more suitable to dealing with large numbers of computation, thus we would want to simulate these RNNs on GPUs
- today I'm going to talk about 2 types of RNNs, Boltzmann machines and McCulloch-Pitts machines
- their main differences is BM is discrete time and MPM is continuous time, their similarities is that they both have the spiking characteristic in them when we simulate these machines

Boltzmann Machines



- composed of primitive computing elements called units
- units has two states, on or off, represented by $\{1, 0\}$
- weights can take on any real value
- connected to each other by bi-directional links
- link weights are symmetric, having the same strength in both directions

Boltzmann Machines



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

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

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

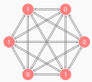
- the neurons are binary stochastic units
- when $\Delta E_i > 0 (< 0)$, $p_i > 0.5 (< 0.5)$
- temperature variable controls the amount of noise; higher temperature means more noise and also gives us a higher probability of transiting to a higher energy state and hence avoids local minimum
- when $\tau \to 0$ we get Hopfield network
- for $\tau_1 > \tau_2$, we are less likely to go to a lower energy state compared to in τ_1 compared to τ_2, i.e. more likely to go to a higher energy state when the temperature is higher. This allows us to escape from local minimum and arrive at the global minimum

2017-10-03

Simulating Recurrent Neural Networks on Graphic Processing Units

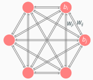
└─Recurrent Neural Networks

└─McCulloch-Pitts Machines



- state 1 is the refractory state, the neuron just fired and is unable to fire till it recovers
- state 0 is the armed state, the neuron just recovered and is waiting to fire
- here we model the units with the Nossenson-Messer neuron model, which explains biological firing rates in response to external stimuli

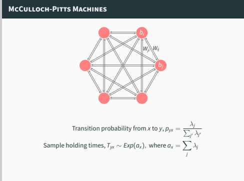
McCulloch-Pitts Machines



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

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

- here the W matrix need not be symmetrical with zero diagonals like what we had in the Boltzmann machine model
- we define a transition as a state that is one hop away from the current state, i.e. differs by one bit
- we shall think of x as the current state and y to be any state that is one hop away
- transition energy requires the current and the future state that it is transiting to
- for each $y \neq x$, start a Poisson process with rate $\Gamma_{yx} = \lambda_j$, hence for d neurons, we start d Poisson Processes
- the neuron chosen to transit is the neuron whose Poisson Process has the smallest interarrival time, which uniquely determines the new state
- we store the smallest interarrival time; this is the holding time for state x; time that the system stays in state x
- as such, we can talk about the interarrival timings of the Poisson process and our simulation of the McCulloch-Pitts machine not only gives us a binary tuple, but also the time taken from it to transit from its earlier state



- when doing the updates we can just update the linear responses z_j and apply softmax on the λ_j's to get the probability distribution of the transitions.
- it seems counter-intuitive to think of 0 as armed and 1 as refractory, but it is in fact the most natural thinking
- a transition from $0 \longrightarrow 1$ is the firing process and a transition from $1 \longrightarrow 0$ is the recovery process
- when a neuron transit from $0 \longrightarrow 1$, it changes the value of the linear response; for a transiting neuron i, if $W_{ji} > 0$, then such a transition increases the linear response of neuron j and if $W_{ji} < 0$ it decreases the linear response of neuron j
- the sign s depends on the state of the neuron, it preserves the sign of the linear response if it is armed and flips the sign of the linear response if it is refractory

Simulating on GPUs

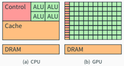


Figure 3: Comparison between the amount of transistors devoted to different functions inside a CPU and a GPU.

- To simplify quite a bit, think of a GPU as a factory and a CPU as Steven Hawking. Factory workers, each represented by a core, can complete lots of easy, similar tasks with incredible efficiency?tasks like geometry and shading. On the other hand Mr. Hawking, while incredibly smart and only occasionally baffled, is just one man. His skill set is better used on singular, complex problems like artificial intelligence.
- DRAM: dynamic random access memory, ALU: arithmetic logic unit,Cache, Control
- trade off control for compute in the form of lots of simple compute units
- GPUs have an explicit programming model; we have to write programs in the way that we utilise as much of the parallel processing as much as possible
- GPUs optimize for throughput, not latency; they are willing to accept increase latency of any single individual computation in exchange for more computation being performed per second, the computation performed per second is measured by floating point operations per second (FLOPS)
- GPUs are good at efficiently launching lots of threads and running them in parallel

Importance to Simulating on GPUs

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

- train larger neural networks
- learning from a larger function space
- GPUs are more energy efficient than CPUs; they are optimized for throughput and performance per watt and not absolute performance