Machine Learning

Jan Koos Assheuer [jan.assheuer@fs-students.de]
Henricus Bracht [henricus.bracht@fs-students.de]

March 22, 2022

1 Abstract

Nature has always been an inspiration for our technology. Copying properties of our brain into a machine is an interesting problem that has been investigated with Hopfield Networks that can already form memory. Our project aims to contextualize Hopfield Networks in the evolution of digital replication of "intelligence" and to draw conclusions from tweaking parameters. Furthermore, we will examine the relevance of Hopfield Networks. For these purposes, we have developed our own Hopfield Network. We found that Hopfield Networks are relevant because of their similarity to units of our brain but yet, they have computational problems and are not efficient enough to be applied.

2 Introduction

We as humans have always strived for understanding how we are able to think and reason. That a Neuron is the basic unit of our nervous system has been clear since 1891 when Heinrich Wilhelm Waldeyer wrote his paper "Über einige neuere Forschungen im Gebiete der Anatomie des Zentralnervensystems".[8] Ever since, it was of course of interest to replicate a model with neurons to also understand our brain better. McCulloch and Pitts did so in their 1943 paper "A logical calculus of the ideas immanent in nervous activity"[1]. They realized because of the neuron's "all-or-none" character that nervous activity can be described in propositional logic. They created an artificial model of a neuron called "McCulloch-Pitts-Neuron". With their neuron, it was possible to portray the logical proposition functions AND, OR and NOT. This concept was the predecessor of the Perceptron that was first published in 1957 by Frank Rosenblatt. This neuron model is the foundation of modern artificial neural networks of any kind, including the Hopfield Network that John Hopfield came up with in his paper "Neural networks and physical systems with emergent collective computational abilities"[2] in 1982. He investigated the collective properties of a system with many simple components i.e, Perceptrons. The ability of saving memories emerged. In the following treatise, we are going to first summarize the two underlying papers at hand also regarding the methodology of the research of both papers. Later on, we are going to discuss the results of both papers also regarding weaknesses in the model. Finally, we are going to mathematically prove that Energy in Hopfield Networks is monotonically decreasing and we are going to calculate the absolute value that the networks energy is decreased by. Our report will also reflect on the relevance of Hopfield Networks present-day.

3 McCulloch's and Pitt's Paper

Warren S. Culloch and Walter Pitt wanted to process how it is possible that a conglomerate of simple nervous cells called "Neurons" are capable of creating complex and abstract thought. They wanted to create a model, that makes it possible to make the brain a propositional function[1]. They defined the functionality of biological neurons as follows:

Neurons consist of Soma, the nucleus of the cell, and the axon, the “wire”. Neurons connect with the axon of Neuron 1 to the soma of Neuron 2. This connection is called Synapse. Neurons can either fire or not fire. This is what they call the “All or None” Character of the Neuron. Neurons only start to fire if their inputs exceed a certain threshold θ. Transmission speed depends on the length and the thickness of the axon. There is a signal delay at the synapses. As soon as a neuron is activated via a synapse, it turns into a state of refractoriness in which it is not easily excitable, then it returns into a state slightly more excitable in which it stays. Thus, the neuron is more easily excitable from that synapse. This process is known as learning.

Inhibition is the opposite or respectively inhibitory signals can decrease the excitedness of a neuron.

They apply this concept in their own artificial neurons that are in fact a predecessor of the perceptron. Their model sees the neuron as receiving a vector of inputs $\vec{x}$. These inputs have to exceed a threshold for the neuron for it to fire. One inhibiting signal immediately shuts the activation of the neuron.

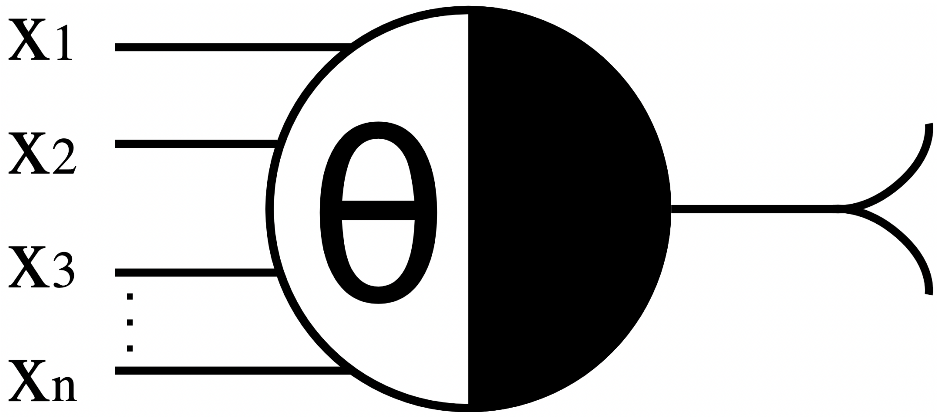

Figure 1: McCulloch-Pitts Neuron (Adrian Lange, CC BY-SA 3.0 https://creativecommons.org/licenses/by-sa/3.0, via Wikimedia Commons)

This makes it possible for us to replicate the complete Boolean algebra as it capable of the AND, OR and NOT functions. However, similarly to the Perceptron, it cannot replicate functions that are not linearly separable.

4 Hopfield’s Paper

In his Work “Neural networks and physical systems with emergent collective computational abilities“[2], published by J. J. Hopfield in 1982 he offers insight into how biological neurons save memories between themselves and defines the properties of a general content-addressable memory he proposes. Such a model would need to be able to store and retrieve information, be robust against changes in the model details and it should be capable of retrieving a memory item by a query using only partial information e.g. the information: "H. A. Kramers & G. H. Wannier Phys. Rev. 60, 252 (1941)." should be retrievable by a query containing just: "& Wannier, (1941)" or possibly even the slightly incorrect query "& Vanier, (1941)". If $x_1 \cdots x_N$ are coordinates in a state vector X, and the system is initialized in the close neighborhood of a local stable point X_a, X_b at $X = X_a + \Delta$ the system should then proceed until $X \approx X_a$, generating the total information from the incomplete query.
Such a network would later become to be known as "Hopfield Network" and is described as follows: Each neuron i has two states: $x_i = -1 :=$ not firing or $x_i = +1 :=$ firing.[1] Each i of the N Neurons in the Network is connected to each other Neuron j with the strength of w_{ij} where $w_{ij} = w_{ji}$ and $w_{ii} = 0$. When the network is running asynchronously, at each point in time t a random neuron is chosen and given the chance to update according to the activation function: $\sum_{j \neq i} w_{ij} x_j$. If the result is $\geq \theta_i$ the neurons state becomes $+1$, else it

[1]In his paper J.J. Hopfield described the states as 0 and 1 but for consistency we will use the annotation -1 and $+1$

becomes -1, where θ_i is a fixed threshold (by default 0). Another important property of Hopfield networks is Energy: A model's Energy is given by the function: $E = -\frac{1}{2}\sum_{i,j} w_{ij}x_i x_j - \sum_i b_i x_i$. A change in energy when Neuron i changes its state is denoted by the formula $\Delta E_i = -(x_i^{New} - x_i^{Old})(\sum_j w_{ij}x_j + b_i)^2$. Thus if the value of x_i is changed according to the activation function it can be shown that Energy is a monotonically decreasing function[3]. Changes will occur in the network until a global or local energy minimum is reached, then no more change will occur as the network will have settled into a stable state. Such a stable state can fit into one of two categories:

1. Useful states: The network settles into a learned pattern or an inverse of a learned pattern where every neuron takes on the state opposite to the one in one particular learned configuration.

2. Confused states: A mistakenly learned pattern that is a mixture of one or more saved patterns.

New patterns can continually be added to the system up to a capacity of $N!$, however the chance of settling into a useful pattern is positively correlated to the ratio $\frac{n}{N}$ where N is the Number of Neurons and n is the number of memories. The Hamming distance is defined as the Number of digits in the network that are different between several memories. If the Hamming distance between saved and new patterns is greater, more patterns can be safely added to the network, if it is small the probability that patterns are mixed up or the wrong pattern is retrieved increases.
If the network is updated synchronously, meaning that several neurons can update at the same time it is possible that the network does not settle into a stable state but fluctuates between two saved patterns e.g. $A \Rightarrow B \Rightarrow A \Rightarrow B$.... Further a chaotic wandering through random patterns with a small Hamming distance can be observed.

5 Theory and Methods

To test the functionality and limitations of Hopfield Networks we have constructed a Network in the Python programming language, modelling single Neurons through classes and storing the network's states in Pytorch Tensors. We choose to model the Hopfield network using Asynchronous Updates and drawing Neurons without replacement to be able to always reach stable states and the greatly reduced run-time complexity.

The activation function of each Neuron is seen below:

```
def activation_fn(self):
    total = 0
    for neuron in self.network.neurons:
        if neuron is not self:
            if neuron.state == 1:
                total += self.network.weights[self.n, neuron.n]
            else:
                total -= self.network.weights[self.n, neuron.n]
    return total
```

If the return value is more than or equal to 0 the neuron updates to 1, else it updates to -1 regardless of the prior state. The weights between neurons can be added or subtracted directly from the sum because they would be multiplied by either -1 or 1, only changing it´s algebraic sign.

We implemented the networks run function like follows:

```
def run_without_replacement(self, steps):
    for i in sorted(np.arange(steps), key=lambda k: np.random.random()):
        while i >= self.n_neurons:
```

[2]See Section 6.2 for more details

[3]Refer to Section 6.1 for a mathematical proof.

```
        i -= self.n_neurons
    self.neurons[i].update()
```

The loop function ensures that the amount of unnecessary neuron updates is minimized by calling all neurons without replacement

The training function of the networks uses Hebbian learning and was implemented like this:

```
def train(self):
    for neuron_i, neuron_j in IterNeurons(self.neurons):
        hebbian_sum = 0
        for pattern in self.patterns:
            hebbian_sum += pattern[neuron_i.i, neuron_i.j] * \
                                pattern[neuron_j.i, neuron_j.j]
        hebbian_weight = hebbian_sum / len(self.patterns)
        self.weights[neuron_i.n, neuron_j.n] = hebbian_weight
        self.weights[neuron_j.n, neuron_i.n] = hebbian_weight
```

The IterNeurons class is an iterator that iterates over every neuron paired with every other neuron exactly once. Inside of the second loop the state of both neurons in every saved pattern is added to a sum which is then set as the weight between the neurons.

We can obtain patterns for the network in two ways:

1. MNIST Dataset: We can get a binary representation of a hand-drawn number from the MNIST Dataset. We resize the PIL image to the networks dimension and represent dark pixels as 1 and light pixels as -1.
2. Random Generation: To get patterns with a greater Hamming Distance we generate a random pattern with the dimension of the network using the torch.rand function.

A visualization of our Networks update cycle with two saved patterns from the MNIST dataset can be found under this link: `https://www.youtube.com/watch?v=uerMSyDq2_Y`

A Hopfield network is said to correctly represent a pattern $\{p_i^v = \pm 1 | 1 \leq i \leq N\}$ if $x_i(t+\Delta t) = x_i(t) = p_i^v(t)$ for all i. We modeled the fraction of n patterns that can be correctly retrieved by a network of N Neurons starting at a state that differentiates itself from the corresponding ground truth by the measure p[4]. Extensive computer modelling of Hopfield networks with varying values for N (The number of Neurons), n (The number of saved patterns) and p (How different the queried pattern is from the ground truth) have shown the portion of patterns that can be correctly represented to be highly varying on those parameters. Especially the ratio of $\frac{n}{N}$ as described by J.J. Hopfield is highly determining for the ratio of patterns that can be represented. Figure 2 shows on the X Axis the amount of patterns saved into a Network of N[5] Neurons and on the Y Axis the percentage share of patterns that can be correctly obtained from a query state that is 5% different to the ground truth. Here it can be seen that at a ratio $\frac{n}{N}$ of approximately 0.11 the share of solvable patterns decreases to below 95%.

[4]We measure p as the percentage value of Neurons who´s state has been flipped. As such a value of p=0 means that the state has not been changed and p=0.5 means that 50% of the Neurons in the pattern have been flipped.

[5]We have chosen relatively small values for N as to speed up the run-time of the exponentially difficult network run function. To account for random fluctuation we made 20 reruns of all models and reported the mean value, as such over 10 Million calls to Neurons update function were needed per model and run-times were becoming exorbitantly high with higher values for N.

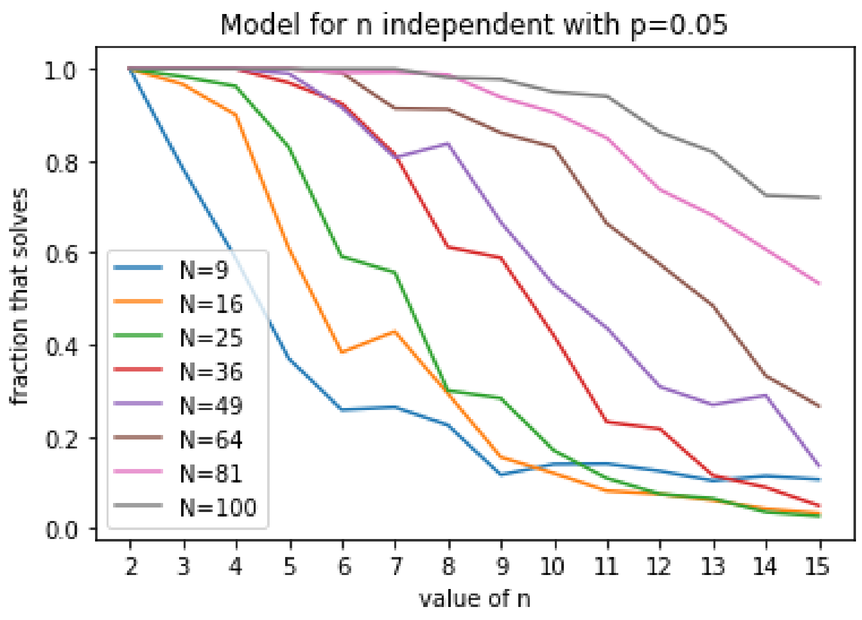

Figure 2: Model for N independent with n=5

In Figure 3 below you can see a similar graph displaying the percentage of patterns that are solvable on the Y Axis and the value of p on the X Axis. From this it is observable that the level of corruption of a networks state can reach up to 10% - 20% before the pattern becomes unrecognizable to the solver function. From there on However the networks ability to recover the original pattern decreases drastically, fully diminishing at 50% corruption or more.

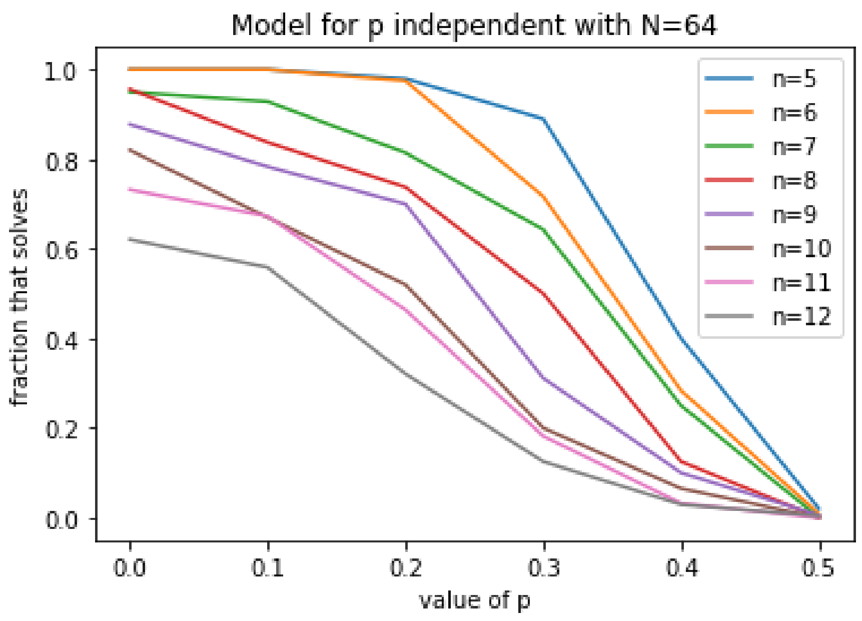

Figure 3: Model for p independent with N=64

6 Results

An advantage of the Hopfield Network is the cheap training: The time complexity of training n patterns into a network of N Neurons is given by $T(n, N) = O(n \times N)$.

A major limitation of the model we have observed is the exponentially increasing computational complexity with a rising number of N Neurons. This behavior can be attributed to two kinds of properties of the model:

1. Neuron activation:
   For any neuron to update its activation must be called. In this function the strength and state of every connection is added up and returned. As such the time complexity of the activation function is $T(N) = O(N)$ and the time complexity for updating N Neurons is $T(N) = O(N^2)$

2. Neuron selection:
   When the Network attempts to update, it chooses a random neuron with replacement and calls its activation function. The chance for the network to choose each Neuron is $\frac{1}{N}$ and as such at the end of a pattern retrieval process when most neurons are stable and only a few are left to be updated the network mostly chooses already stable neurons, calls their activation function and then does not change them. This behavior however can be circumvented by drawing the Neurons without replacement, this changes the average time complexity of drawing one specific Neuron to 1. However, this can influence which patterns are retrieved by a query to the network in rare cases, especially with a small number of neurons.

As such the time complexity of updating N neurons in a Network with replacement is $T(N) = O(N^3)$ and without replacement it is $T(N) = O(N^2)$. Thus, a mayor shortcoming of the network is revealed to be the computation time of settling into a stable state.

Something that we considered unexpected is the high amount of stable states that can exist within the network using such a simple learning rule (Up to 0.15 - 0.25 times the number of Neurons can be stable as saved patterns, depending on hamming distance and the number of Neurons). Likewise surprising we found the fact that with patterns that have a small Hamming-Distance, only very few can be stored in the Network before states become unrecognizable. Furthermore, we were impressed by the difference in run-time complexity resulting from choosing neurons with or without replacement.

6.1 Proof of Decreasing Energy

We are given the Energy formula:

$$E = -\frac{1}{2}\sum_{i,j} w_{ij}x_i x_j - \sum_i b_i x_i \tag{1}$$

We assume $\forall i\ x_i \in \{-1, 1\}$, $w_{ij} = w_{ji}$. Now, we want to take a look at what is happening in case we update x_i. The change in Energy is given by:[6]

$$\Delta E = -(x_i^{New} - x_i^{Old})(\sum_j w_{ij}x_j + b_i) \tag{2}$$

$\frac{1}{2}$ can be removed because before it was only serving the purpose of compensating the addition of unnecessary terms because Matrix W is symmetric. This meant that we summed double the amount we wanted. Since now we only regard each connection to neuron i, we have to remove $\frac{1}{2}$.
Then, we have 4 different cases to investigate.

- $x_i^{New} = -1$, $x_i^{Old} = -1$:

$$\Delta E = -(-1 - (-1))(\sum_j w_{ij}x_j + b_i) = 0 \tag{3}$$

[6] See appendix for more information

$$\Delta E = -(0)(\sum_j w_{ij}x_j + b_i) = 0 \tag{4}$$

- $x_i^{New} = 1$, $x_i^{Old} = 1$:

$$\Delta E = -(1-1)(\sum_j w_{ij}x_j + b_i) = 0 \tag{5}$$

$$\Delta E = -(0)(\sum_j w_{ij}x_j + b_i) = 0 \tag{6}$$

- $x_i^{New} = -1$, $x_i^{Old} = 1$:

$$\Delta E = -(-1-1)(\sum_j w_{ij}x_j + b_i) = 2\sum_j w_{ij}x_j + b_i \tag{7}$$

$$x_i^{New} = 0 \Rightarrow a_i = \sum_j w_{ij}x_j + b_i \leq 0$$

$$\Delta E \leq 0$$

- $x_i^{New} = 1$, $x_i^{Old} = -1$:

$$\Delta E = -(1-(-1))(\sum_j w_{ij}x_j + b_i) = -2(\sum_j w_{ij}x_j + b_i) \tag{8}$$

$$x_i^{New} = 1 \Rightarrow a_i = \sum_j w_{ij}x_j + b_i \geq 0$$

$$-2(\sum_j w_{ij}x_j + b_i) = \Delta E \leq 0$$

For all cases $\Delta E \leq 0$
Quod erat demonstrantum

6.2 Energy difference of Neuron

As we have already found out in section 6.1, the change in Energy of the in case Neuron i changes can be described as:

$$\Delta E_i = -(x_i^{New} - x_i^{Old})(\sum_j w_{ij}x_j + b_i)$$

Also, we have already investigated both cases that are either a switch of state of Neuron i from -1 to 1 or vice versa from 1 to -1, the calculations are shown above.

Henceforth, we found out that the change of Energy is always $-|2(\sum_j w_{ij}x_j + b_i)|$.

7 Discussion

Hopfield Networks have highly interesting properties and are thus worthwhile to investigate. They established a relationship between Neural Networks and statistical mechanics. Restricting a recurrent Neural Network to symmetric connection led to the occurrence of stable states. The Energy function made it possible to analyze convergence properties. The Energy Function is also analogous to the Energy function in the Ising-Model[9] that describes Ferromagnetism mathematically, again relating to statistical mechanics. More importantly, training a

Hopfield Network is easy because it is based on Hebbian Learning. In contrast, modern (Deep) Neural Networks have highly complex training algorithms such as Backpropagation. More importantly, it is quite similar to learning in Human brains, since Hebbian Learning is inspired from that. That makes it possible to have "One-Shot-Learning" i.e., to recognize a pattern, the Network must not have seen a huge amount of training samples. In comparison, they can therefore reconstruct corrupted patterns such as Autoencoders but with less training effort, also because Hopfield Networks learn unsupervised. There have also been studies suggesting that structures that are similar to Hopfield Networks can be found in our brain within the Hippocampus.[5] As we could observe, implementing a Hopfield Network does not require big effort, although modern Machine Learning Frameworks do not offer a Hopfield Network Feature.

However, we need to acknowledge that Hopfield Networks are not applied outside of other models for a reason. They have notable limits that make it diffult to apply them nowadays.[6] As McCulloch and Pitts already stated, understanding the logic is difficult for recurrent networks. In addition, it is computationally expensive since it can only store around $0.15N$[2] patterns with N neurons while being properly accurate. Thus, the more patterns you have, the more Neurons you need. Henceforth, it could be necessary to reduce the size of of optical components. Furthermore, it could not profit from the resurgence of Deep Neural Networks since the Hopfield Network just works differently. Hopfield Networks as such just keep being building blocks for more advanced principles such as Boltzmann Machines and Deep Belief Networks.

8 Conclusion

To conclude our paper, we can definitely say that Hopfield Networks are an interesting technology that still have a relevance in todays work, having in mind it is still applied in modern models[7]. Not only because of this but also because Hopfield Networks are interesting in biological research, since they replicate biological functions in the Hippocampus. Other models like Convolutional Neural Networks have been further developed because they can solve specific tasks. With further research on Hopfield Networks, we might be capable to understand our own intelligence better. It is a pity that research has not gone further investigating on Hopfield Networks. Still, computational problems arise in storing memories because of the exponential growth of complexity with having more Neurons. For further developement, we have to restrict neural connections in a Hopfield Network without it losing its properties.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," The Bulletin of Mathematical Biophysics, vol. 5, no. 4, pp. 115–133, 1943.

[2] J. J. Hopfield, "Neural networks and physical systems with emergent collective com- putational abilities," Proceedings of the National Academy of Sciences of the United States of America, vol. 79, no. 8, pp. 2554–2558, 1982.

[3] Liviu, S. (2019, 10. September). Hopfield nets and the brain - Serban Liviu. Medium. Abgerufen am 10. März 2022, von https://medium.com/@serbanliviu/hopfield-nets-and-the-brain-e5880070cdba#:%7E:text=Hopfield%20nets%20learn%20by%20using,the%20values%20of%20those%20neurons.

[4] Marsalli, M. (o. D.). McCulloch-Pitts Neurons. The Mind Project. Abgerufen am 3. März 2022, von https://mind.ilstu.edu/curriculum/mcp_neurons/index.html

[5] Piccinini, G. (2004). The First Computational Theory of Mind and Brain: A Close Look at Mcculloch and Pitts's "Logical Calculus of Ideas Immanent in Nervous Activity". Synthese, 141(2), 175–215. https://doi.org/10.1023/b:synt.0000043018.52445.3e

[7]Most notably, Hopfield networks find usage in modern Boltzmann machines, which are famously used by Netfilx´s recomendation system.

[6] Rojas, R. (1996). The Hopfield Model. Neural Networks, 335–369. https://doi.org/10.1007/978-3-642-61068-4_13

[7] Shaurya, M. (2021, 7. Dezember). McCulloch-Pitts Neuron vs Perceptron model - Manu Shaurya. Medium. Abgerufen am 15. März 2022, von https://medium.com/@manushaurya/mcculloch-pitts-neuron-vs-perceptron-model-8668ed82c36

[8] Waldeyer, W. (1891). Ueber einige neuere Forschungen im Gebiete der Anatomie des Centralnervensystems (Fortsetzung aus No. 44.). DMW - Deutsche Medizinische Wochenschrift, 17(45), 1244–1246. https://doi.org/10.1055/s-0029-1206842

[9] Wikipedia-Autoren. (2004, 21. Mai). Hopfield-Netz. Wikipedia. Abgerufen am 22. Februar 2022, von https://de.wikipedia.org/wiki/Hopfield-Netz

[10] Wikipedia-Autoren. (2005, 26. Januar). McCulloch-Pitts-Zelle. Wikipedia. https://de.wikipedia.org/wiki/McCulloch-Pitts-Zelle

Appendices

Derivation to Energy Difference formula

$$E = -\frac{1}{2}\sum_{i,j} x_i x_y w_{ij} - \sum_i x_i b_i \tag{1}$$

$\frac{1}{2}$ can be removed because before it was only serving the purpose of compensating the addition of unnecessary terms because Matrix W is symmetric.

$$E_i = x_i(-\sum_j x_j w_{ij} - b_i) \tag{2}$$

$$\Delta E_i = x_i^{New}(-\sum_j x_j w_{ij} - b_i) - x_i^{Old}(-\sum_j x_j w_{ij} - b_i) \tag{3}$$

$$\Delta E_i = (x_i^{New} - x_i^{Old})(-\sum_j x_j w_{ij} - b_i) \tag{4}$$

$$\Delta E_i = -(x_i^{New} - x_i^{Old})(\sum_j x_j w_{ij} + b_i) \tag{5}$$

Further sensitivity analysis

Below you can see further graphs of modeled parameters where:
N = Number of Neurons in the Network
n = Number of patterns in the Network
p = Percentage value by which queried patterns are different.

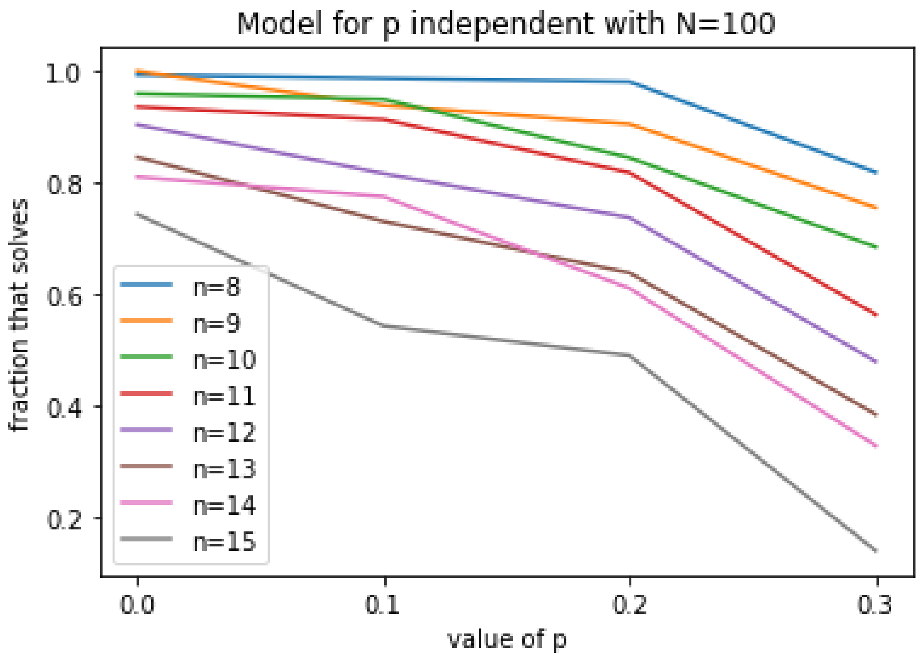

Figure 4: Model for p independent with N=100

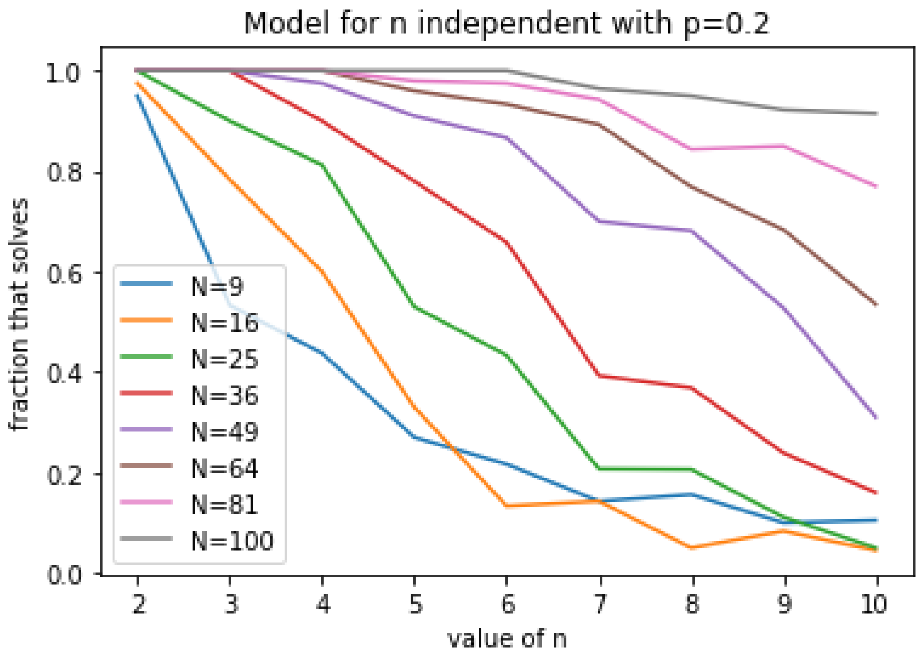

Figure 5: Model for n independent with p=20%

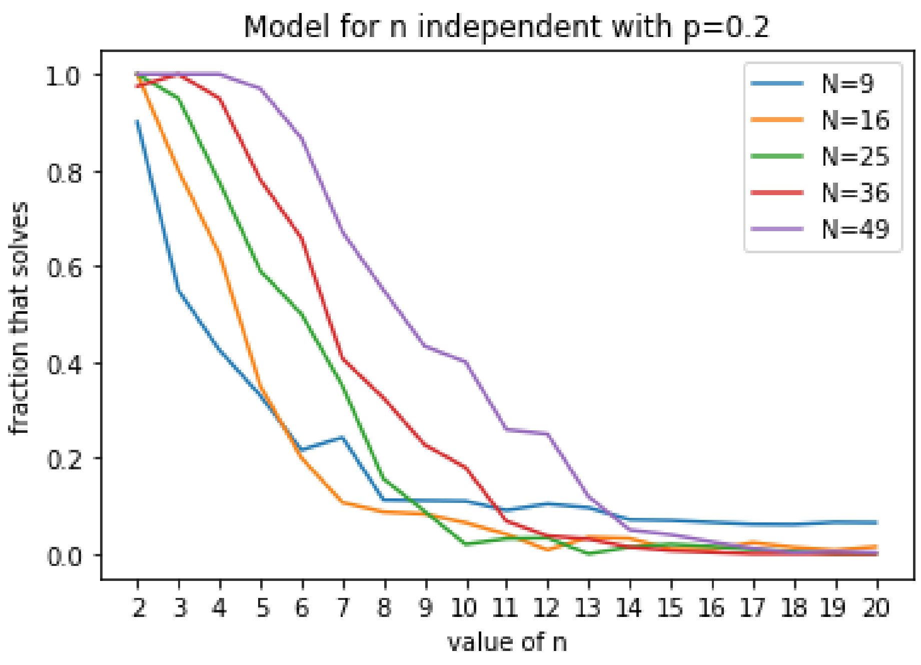

Figure 6: Model for n independent with p=20%

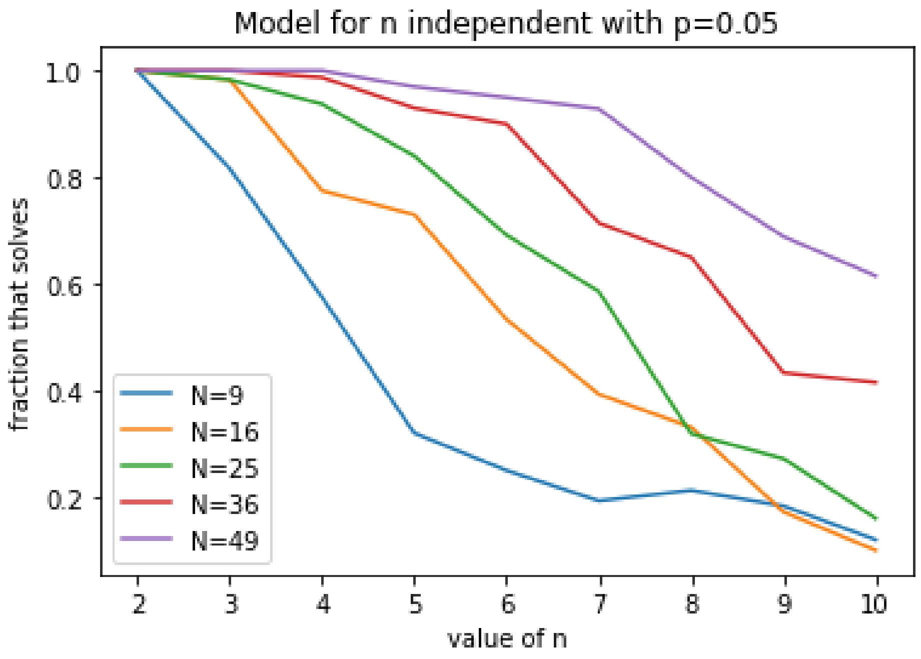

Figure 7: Model for n independent with p=5%

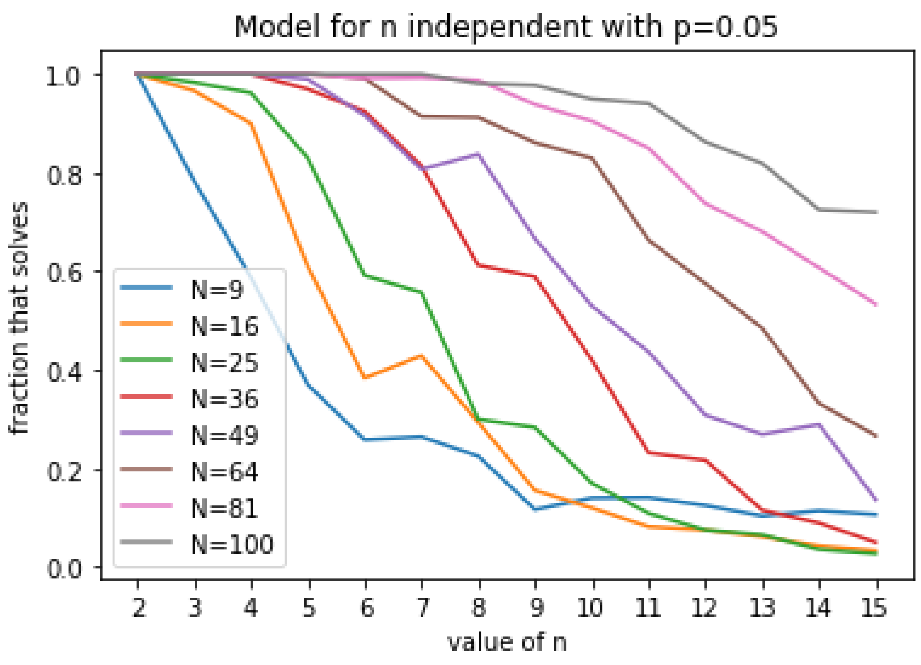

Figure 8: Model for n independent with p=5%

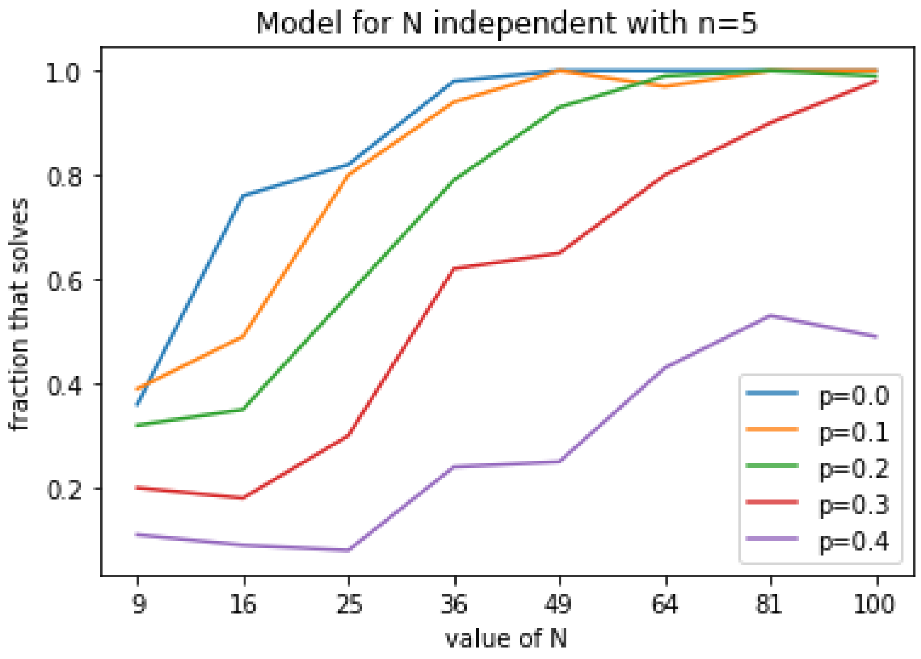

Figure 9: Model for N independent with n=5