Electrical Engineering Department
Sharif University of Technology

Title:

Homework 1

Neural Networks

Author:
Mohammad Hossein Najafi (97103938)

Supervisor:
Dr. S. Bagheri

Spring 2024

1

Artificial neural networks (ANNs) are a type of architecture consisting of interconnected elements called neurons. They improve and grow with input from mathematicians, biologists, and psychologists, who base their work on biological evidence and mathematical principles. One important contributor to the field of ANNs is Donald Hebb, a psychologist who provided a foundational learning framework to explain how ANNs function and are modeled[1].

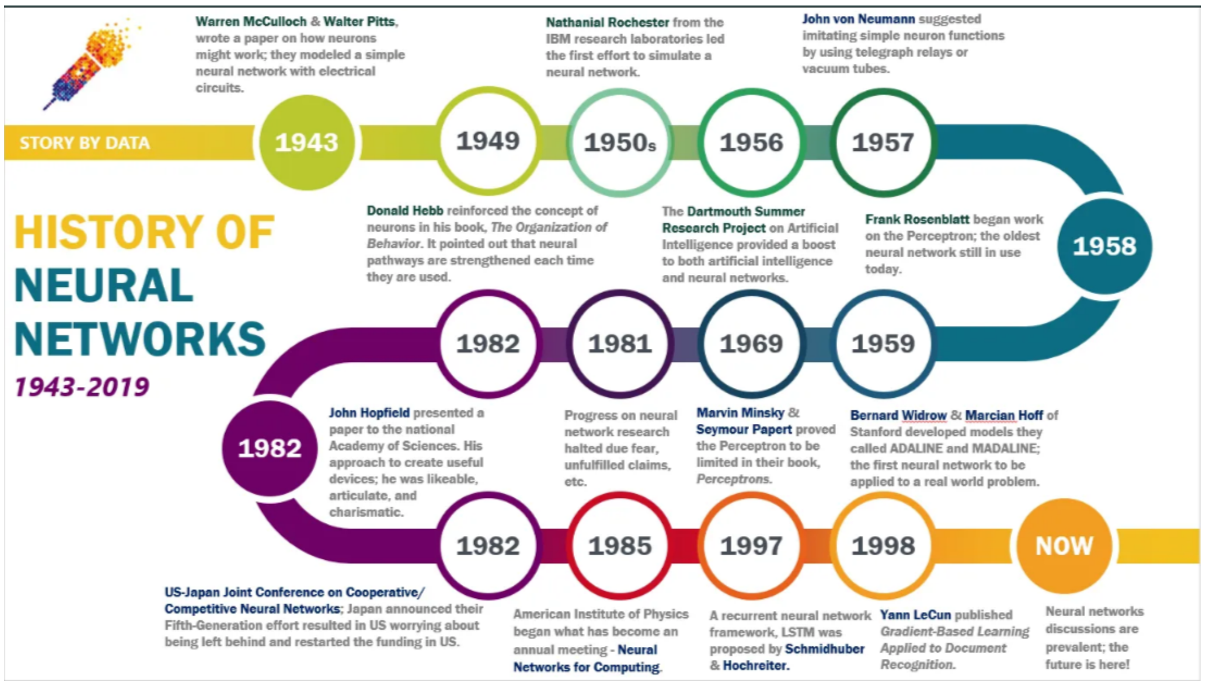
STORY BY DATA
HISTORY OF NEURAL NETWORKS
1943-2019
1943
Warren McCulloch & Walter Pitts, wrote a paper on how neurons might work; they modeled a simple neural network with electrical circuits.
1949
Donald Hebb reinforced the concept of neurons in his book, The Organization of Behavior. It pointed out that neural pathways are strengthened each time they are used.
1950s
Nathanial Rochester from the IBM research laboratories led the first effort to simulate a neural network.
1956
The Dartmouth Summer Research Project on Artificial Intelligence provided a boost to both artificial intelligence and neural networks.
1957
John von Neumann suggested imitating simple neuron functions by using telegraph relays or vacuum tubes.
1958
Frank Rosenblatt began work on the Perceptron; the oldest neural network still in use today.
1959
Bernard Widrow & Marcian Hoff of Stanford developed models they called ADALINE and MADALINE; the first neural network to be applied to a real world problem.
1969
Marvin Minsky & Seymour Papert proved the Perceptron to be limited in their book, Perceptrons.
1981
Progress on neural network research halted due fear, unfulfilled claims, etc.
1982
John Hopfield presented a paper to the national Academy of Sciences. His approach to create useful devices; he was likeable, articulate, and charismatic.
1982
1982
US-Japan Joint Conference on Cooperative/Competitive Neural Networks; Japan announced their Fifth-Generation effort resulted in US worrying about being left behind and restarted the funding in US.
1985
American Institute of Physics began what has become an annual meeting - Neural Networks for Computing.
1997
A recurrent neural network framework, LSTM was proposed by Schmidhuber & Hochreiter.
1998
Yann LeCun published Gradient-Based Learning Applied to Document Recognition.
NOW
Neural networks discussions are prevalent; the future is here!

Figure 1: History of Neural Networks[2]

Hebbian learning, or Hebbian theory, suggests that neurons process incoming signals to generate desired responses. These neural units are interconnected via inputs (x1, x2, x3...xn), connection weights (w1, w2, w3...wn), and activation functions (functions that determine the output of a node).

Now, delving into the explanation of the Hebb network: "When the activity of cell A's axon repeatedly triggers cell B's firing, resulting in consistent excitation, metabolic changes occur in one or both cells, enhancing A's effectiveness in activating B."

In this scenario, if two linked neurons are simultaneously active, adjustments to their synaptic connections can strengthen their association. The Hebbian weight update rule is expressed as follows:

$$w_{\text{new}} = w_{\text{old}} + x \times y$$
$$bias_{\text{new}} = bias_{\text{old}} + y$$

Learning Algorithm

1. First step: Set all weights and biases to zero.
2. Second step: Introduce the learning pattern: $x = s$, $y_{\text{out}} = t$.

3. Third step: Calculate changes in weights and bias: $\Delta w = s \cdot t$, $\Delta \text{bias} = t$.
4. Fourth step: Update weights and bias.
5. Fifth step: If all patterns are learned, stop; otherwise, return to the second step.

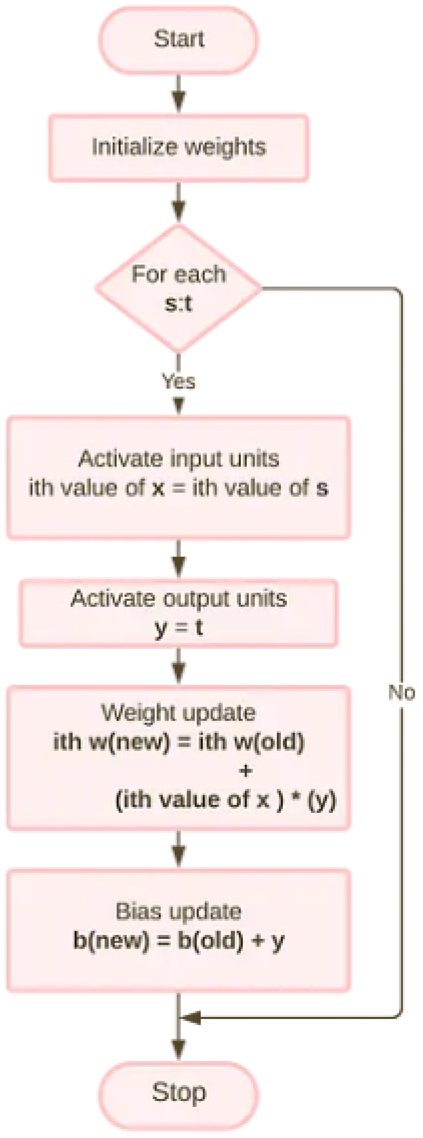
Start
Initialize weights
For each s:t
Yes
No
Activate input units ith value of x = ith value of s
Activate output units y = t
Weight update ith w(new) = ith w(old) + (ith value of x) * (y)
Bias update b(new) = b(old) + y
Stop

Figure 2: Flowchart of Hebb training algorithm[1]

2

No, a multilayer Hebbian neural network cannot be imagined. The fundamental principle of Hebbian learning, known as ”cells that fire together, wire together,” implies that synaptic connections strengthen between neurons that are simultaneously activated. In a multilayer network, the learning process becomes more complex, as neurons in different layers may not directly interact during training. Additionally, Hebbian learning lacks a mechanism to adjust synaptic connections based on error feedback, which is essential for training deep neural networks. Therefore, while Hebbian learning is suitable for single-layer networks or simple associative learning tasks, it does not apply to multilayer architectures commonly used in modern neural networks.

3

Letters with simple shapes or those that look the same on both sides, like ”B”, ”C”, and ”D”, may not be learned well because the model learns patterns where parts of the letter are similar. Letters with straight lines, such as ”L” and ”N”, might also be learned quickly since the model can grasp these simpler shapes easily. However, letters like ”B” and ”D” might be a bit trickier because they have both straight lines and curves, making it harder for the model to figure out the right connections between different parts of the letter.

4

Initially, within the code, we define a function to display fonts, establish a fonts array in a 63-cell array (7×9) resolution in binary, and generate the pattern and target arrays. Subsequently, we define a model class and allocate 5 neurons to it. We then input the pattern along with the target into the model, allowing it to learn. Afterward, we assess its predictions, yielding the following results:

- Error rate for ’B’: 80.0%
- Error rate for ’C’: 80.0%
- Error rate for ’D’: 80.0%
- Error rate for ’L’: 80.0%
- Error rate for ’N’: 80.0%

Total error rate: 80.0%
As observed, the error rate stands at 80%, indicating suboptimal targets. This suggests that for all fonts, each neuron’s target is consistently 1, which is unfavorable.

5

For bipolar encoding, we first define a function that replaces 0 cells with -1 cells and provide the patterns and targets for this function. Subsequently, we train the model with bipolar encoding arrays, resulting in the following outcomes:

- Error rate for 'b': 13.33%
- Error rate for 'c': 0.0%
- Error rate for 'd': 20.0%
- Error rate for 'l': 13.33%
- Error rate for 'n': 0.0%

Total error rate: 9.33%
As observed, the error rate decreases significantly, indicating a much better prediction performance compared to binary encoding.

6

6.1 10% noise

To insert 10% noise, we randomly change about 6 cells from each pattern array.
The resulting error rates are as follows for binary encoding:

- Error rate for 'b': 80.0%
- Error rate for 'c': 80.0%
- Error rate for 'd': 80.0%
- Error rate for 'l': 80.0%
- Error rate for 'n': 80.0%

Total error rate for noisy patterns: 80.0%

For bipolar encoding, the error rates are as follows:

- Error rate for 'b': 13.33%
- Error rate for 'c': 0.0%
- Error rate for 'd': 20.0%
- Error rate for 'l': 20.0%
- Error rate for 'n': 0.0%

Total error rate for noisy bipolar patterns: 10.67%

As observed, binary encoding remains 100% ineffective under 10% noise, whereas bipolar encoding exhibits some robustness.

6.2 25% noise

For binary encoding with 25% noise, we randomly change about 16 cells from each pattern array. The resulting error rates are as follows:

- Error rate for 'b': 80.0%
- Error rate for 'c': 80.0%
- Error rate for 'd': 80.0%
- Error rate for 'l': 80.0%
- Error rate for 'n': 80.0%

Total error rate for noisy patterns: 80.0%

For bipolar encoding with 25% noise, the error rates are as follows:

- Error rate for 'b': 13.33%
- Error rate for 'c': 0.0%
- Error rate for 'd': 20.0%
- Error rate for 'l': 13.33%
- Error rate for 'n': 13.33%

Total error rate: 12.0%

As observed, binary encoding remains ineffective even with 25% noise, while bipolar encoding still exhibits robustness under these conditions.

7

In the conducted analysis, two encoding methods, binary and bipolar, were assessed for their effectiveness in pattern recognition tasks, particularly under noisy conditions. Binary encoding consistently yielded high error rates of 80.0%, indicating poor recognition performance across various patterns ('B', 'C', 'D', 'L', 'N'). Even when subjected to 10% and 25% noise levels, binary encoding failed to improve its accuracy. Conversely, bipolar encoding exhibited superior performance with lower error rates, demonstrating its resilience to noise. With 10% noise, bipolar encoding achieved a total error rate of 9.33%, showcasing its ability to maintain accuracy even in noisy environments. Furthermore, when noise increased to 25%, bipolar encoding still outperformed binary encoding with a total error rate of 12.0%. These findings underscore the robustness and effectiveness of bipolar encoding for pattern recognition tasks, highlighting its advantages over binary encoding, particularly in scenarios involving noisy data.

8

The prediction made in section 3 regarding the model's learning of different letters based on their shapes appears to be consistent with the obtained results. The analysis suggested that letters with simpler shapes or symmetrical features, such as ”C” and ”N”, may be learned more effectively by the model. Indeed, in the results obtained, these letters showed lower error rates or were not affected by noise as much. Conversely, letters with more complex shapes, such as ”B” and ”D”, were predicted to be trickier for the model to learn due to their combination of straight lines and curves. This prediction aligns with the higher error rates observed for these letters in the results, indicating that the model indeed struggled more with recognizing them accurately, especially as the noise level increased. Therefore, the prediction regarding the model's learning behavior based on letter shapes appears to be consistent with the obtained results.

9

In my opinion, both increasing and decreasing the resolution can have significant impacts on the results of pattern recognition tasks.

Increasing Resolution: Increasing the resolution typically involves capturing more detail in the patterns, which can potentially lead to more accurate recognition. With higher resolution, the model can distinguish finer features within the patterns, resulting in better learning and recognition performance. This increased detail can help the model differentiate between similar patterns and reduce ambiguity, ultimately leading to higher accuracy. However, increasing resolution also comes with computational costs, as more data points need to be processed, potentially leading to longer training times and increased complexity.

Decreasing Resolution: On the other hand, decreasing the resolution reduces the amount of detail captured in the patterns. While this can lead to faster processing times and reduced computational complexity, it may also result in the loss of important information. With lower resolution, the model might struggle to differentiate between similar patterns or capture subtle variations, potentially leading to decreased accuracy in recognition tasks. Additionally, reducing resolution excessively may result in patterns becoming too abstract or indistinct, making them difficult for the model to learn effectively.

Impact on Results: The resolution directly affects the amount of information available to the model during training and inference. Higher resolution can provide more detailed information, potentially improving accuracy but at the cost of increased computational requirements. Conversely, lower resolution sacrifices detail for efficiency but may lead to decreased accuracy due to loss of information. Therefore, the optimal resolution depends on the specific requirements of the pattern recognition task, balancing the need for accuracy with computational constraints.

In conclusion, both increasing and decreasing resolution have their respective advantages and disadvantages in pattern recognition tasks. The choice of resolution should be carefully considered based on factors such as the complexity of patterns, computational resources available, and the desired level of accuracy.

References

[1] J. shah, “Hebb network,” https://medium.com/analytics-vidhya/hebb-network-c38596e1a7a1, 2024, accessed: March 2024.

[2] K. Strachnyi, “Brief history of neural networks,” https://medium.com/analytics-vidhya/brief-history-of-neural-networks-44c2bf72eec, 2024, accessed: March 2024.