# The Memory Capacity of Neural Networks

### **Course:** Seminar/Project Machine Learning & Neuroinformatics/Brain-Computer Interfacing (708.415)

### **Author:** Christoph Rieger

### **Student number:** 01530103

### **Date:** 21.01.2023

# Introduction

This project aims to investigate the memory capacity of neural networks depending on their size. Unique input and output patterns are handed to the network to be learned. Two different network architecture, as well as two different definitions of when a pattern is learned correctly, are analyzed.

# Methods

### Data

The input vectors X and target vectors Y are binary vectors of size N. They have a sparsity s which was chosen as 0.1. This means that 10% of the bits of the input and target vectors were active with values of one and the rest of the bits were inactive with values of zero. Both vectors were generated randomly, but no duplicate vectors were allowed within the input and output matrix respectively.

### Network

The first network used was a 1-Layer fully connected network and will be called simple network from now on. It has N input neurons and N output neurons. The input and output layer are fully connected. The activation function used was the sigmoid function, which scales the outputs between zero and one. This behavior is desired, as the output vector’s values lie between those values. For an arbitrary value a it is defined as

(1)

After the application of the activation function the network’s prediction z is obtained. The prediction and the target vector are then handed over to the loss function. In this experiment the network is performing a binary classification task between y and z for each single bit. Thus, Binary cross-entropy (BCE) was chosen because it is designed for such binary classification tasks. For a batch size of one it is given by

(2)

ADAM was chosen as optimizer, because it generally performs well and was converging much faster than stochastic gradient descent.

The described network was later expanded by a recurrence between the output neurons. The training process now consisted of two steps through time, while using the same input vector. The first step was identical to the first network, the input goes to the N input neurons and is passed via the fully connected layer to the N output neurons.   
After that, in the second step, the input was again given to the input neurons of the network. Additionally, the output neurons of step 1 were connected to the output neurons of step 2, creating a recurrent layer. These recurring connections were initialized fully connected between the two output layers, but all weights of connections between the same output neurons (e.g.: output neuron 1 of step 1 to output neuron 1 of step 2) were always kept at zero. This means that the network’s prediction of the first step was used as additional input for the network in the second step.

Furthermore, the same recurrent network was trained again, but a third step was introduced, which took the prediction of step 2 as additional input. This resulted in three total steps with the same input vector.

### Loss and classification

In the evaluation phase the network’s prediction of each bit of the output had to be cast to either 1 or 0. This was done by setting all outputs greater or equal than 0.5 to 1 and all smaller than 0.5 to 0.

Two different definitions of when a pattern counts as correctly memorized by the network were analyzed. The first one only counted a pattern as memorized if the output vector is equal to the target vector.

To create a more lenient definition a second definition was introduced. It calculated the number of bits to ignore #bti as

. (3)

During each training step output neurons with the worst loss were disabled. This was done #bti times for the group of output neurons that had a target of 1 and another #bti times for the group that had a target of 0. The neurons were “disabled” by setting their loss to 0. The idea behind this procedure was that stochastically the input and output patterns could overlap many of their active bits, making it more difficult for the network to 100% correctly classify those overlapping patterns. The bits to ignore are supposed to counteract this stochastic increased difficulty of memorization.  
Due to this disabling of output neurons, and thus preventing them from learning, the validation paradigm also had to be adjusted. Instead of requiring 100% of the bits output vector to equal 100% of the target vector, #bti bits of the active bits and #bti of the inactive bits of the output vector were allowed to be not equal to the respective target bits.

### Analyzed networks

Overall, four different networks were trained and analyzed.

1. simple network
2. simple network with the custom loss function
3. recurrent network that took two steps with the same input data and with custom loss function
4. recurrent network that took three steps with the same input data and with custom loss function

### Determination of the maximum memorizable patterns

datasetSize in 100er schritten erhöhen und die nummer dort picken wo accuracy höher als 90% ist. Wenn man datasetSize weiter erhöht steigt #patternsMemorized, aber die accuracy sinkt immer weiter. Das passiert, weil es bei größerem datasetSize mehr patterns gibt die sich weniger überlappen, und das netzwerk quasi die leichtesten patterns lernt.  
Daher hab ich das mit den 90% definiert.

### Calculation of mean and standard deviation of the memorized patterns

The simple network without a recurrent layer was trained five times with each set of parameters and the mean and standard deviation of the memorized patterns was calculated over those five runs. For the more complex recurrent network only three runs were performed, to speed up the training process.  
For the generated plots the maximum number of memorized patterns was used instead of the mean, as the goal was to find the maximum number of patterns a neural network can memorize.

# Results

Network parameters needed?

Table with memorized patterns needed for each network?

Ein Bild, das Text, Reihe, Diagramm, Screenshot enthält.

Automatisch generierte Beschreibung

Figure 1: Number of patterns memorized of the 3 different networks. The datapoints of the graphs are the maximum number of memorized patterns of all runs. Marked with “x” is the mean of the runs with the standard deviation as a vertical line.

Ein Bild, das Text, Screenshot, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung

Figure 2: Number of memorized patterns per N.

## Ein Bild, das Text, Diagramm, Screenshot, Reihe enthält. Automatisch generierte Beschreibung

Figure 3: Number of memorized patterns per N squared.

# Discussion

In Figure 1 the simple network performed the worst. Its slope was constant, for each added neuron about three more patterns could be memorized.  
By introducing the custom loss function to the simple network it was able to memorize more patterns, the bits to ignore acting like a positive offset. Furthermore, in Figure 2 it can be seen that it was able to memorize more patterns per N, the higher N became.  
When looking at Figure 1 again the most complex network with custom loss and a recurrent layer performed the best. Its performance could be improved even further by performing more recurrence steps, or by adding additional layers. In Figure 2 the memorized patterns per N of the most complex network is larger than of the other two networks. However, it is declining with rising N, in contrary to the other two networks which performed better or the same with rising N. This behavior is most likely due to the more difficult training of the complex network. The loss of the two fully connected networks decreased with each training epoch, making the training process straight forward. In contrast the loss of the recurrent network increased and decreased many times during the training process. Because of that the recurrent network was much harder to train and that could explain the decreasing memorized patterns per N of it.

Something that must be considered, is the fact that with rising N not only did the network get more neurons, but also the input and target vectors got larger and more complex. Because of that the network was not able to memorize N2 more patterns with rising N, as one might expect. The observed behavior of the memorized patterns with regard to N2 can be seen in Figure 3. If the network could memorize N2 patterns the graphs would have a constant slope of one. This is not the case however and all graphs trend towards zero with rising N.