Comparison of Artificial Neural Networks and Human Brains on Solving Number Series

Report of the practical part of the lecture Cognitive Modelling (WS11/12)

Tina Kämmerer Ioulia Kalpakoula Michael Kleber Robert Terbach

Contents

1 Introduction

1.1 Short motivation

In our experiment with artificial neural networks we want to compare the performance of a network with the performance of the human brain. As artificial neural networks are build to somehow resemble the human brains neuronal behaviour, we expect the experiment to be quite interesting. Of course we cannot model the human brain with artificial neural networks since the complexity is far beyond computable yet not even really graspable. But our experiment focuses on a small problem that doesn't need the complete brain to be solved (though still the brains capabilities that are used are way larger than the networks we will build), so our model might get us a little insight to human thinking processes at solving number series. Maybe the faster computational abilities the computer has give our artificial neural network an advantage in calculating number series, but we rather expect that it does not. The complexity in our task is not in calculating numbers but in finding the function to calculate. Solutions are not simple addition or multiplication but a combination of different operations of different complexity. Finding these combinations is the real problem. And because solving number series can hardly be done in a formalized way but more with a try-and-error-like approach, we assume to loose the computational advantage.

1.2 General empirical question: Human ability of solving number sequence problems by inductive reasoning

Our challenge was to study the human ability of solving number sequence problems by the means of inductive reasoning and to design a system which solves the series. There were four different approaches for this task: genetic algorithms, rule-based heuristic systems, inductive programming (e.g. MagicHaskeller, Igor) and artificial neural networks. The assignment itself was divided into three parts. First we selected five number series which had a challenging aspect to them and which we would like our later program to solve. To have a small insight into the human approach to solving number series, we designed and executed a small experiment in order to gain some knowledge which might help us design our program. Afterwards we created our own neural network. In the following we would like to provide background knowledge about neural networks and their design as well as explain our experiment setup and the findings we gained during the execution of the test. Then the configuration of our artificial neural network is introduced and we discuss our overall results.

2 Theoretical background

2.1 A short introduction to artificial neural networks

The general idea behind artificial neural networks arose from the psychologist Donald O. Hebbs famous theory from 1949:
"When an axon of cell A is near enough to excite cell B or repeatedly or persistently takes part in firing it, some growth process

or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased" (Hebb, 1949)

We see this rule in neural networks learning algorithms. The concept of neurons is directly transferred to artificial neural networks. The axons are the connections between neurons and what Hebb called near translates to the weights of these connections. The ability of a neuron to fire when multiple other neurons try to excite is built into a so called activation-function within the artificial neurons (we will use the tangens hyperbolicus function).

2.2 Structure of ANN's

Artificial neural networks consist of several neurons. These neurons, also called Units, receive information from neurons of the previous layer and pass them on to others. Units are affiliated with each other by edges. The intensity of the connection between two neurons is represented by a weight value and this weight value stores the knowledge of the neural network. We can say that the bigger the absolute value of the weight is, the bigger is its influence from one unit to another. This also means that the influence for an error is bigger. The learning procedure of a neural network is defined by its change of weight values between the neurons. How this change of weights happens depends on the used learning rule. We will use backpropagation with momentum as learning rule and shortly explain it. Before the learning process begins the weight values are randomized.

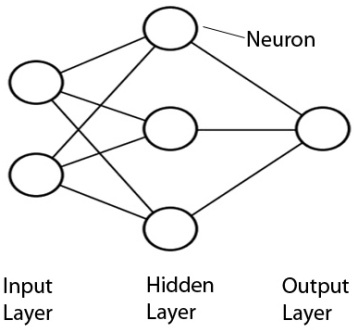

Figure 1: Feed-Forward-Network

Figure 1 shows how the network functions. The Input Units receive the signals like stimuli or patterns. Every Input processes the signals by a separate weight value. The weight values get passed to the hidden layers which contain an internal representation of the environment. The signals pass through the layers and the Output Units pass the output signals to the environment. In the following we will shortly introduce our possibilites to manipulate the learning abilities the neural networks.

2.3 Variables of neural networks

2.3.1 Layers

Artificial neural neural networks, as we have said before, are structured by layers of neurons. We always need an input and one output layer. In between those two are the so called hidden layers. There may be an arbitrary number of hidden layers with each of them of arbitrary size.

2.3.2 Learning Rate

At each training step the network computes the direction in which each bias and link value can be changed to calculate a more correct output. The rate of improvement at that solution state is also known. A learning rate is user-designated in order

to determine how much the link weights and node biases can be modified based on the change direction and change rate. The higher the learning rate (max. of 1.0) the faster the network is trained.

2.3.3 Momentum

To help avoid settling into a local minimum, a momentum rate allows the network to potentially skip through local minima. A history of change rate and direction are maintained and used, in part, to push the solution past local minima. A momentum rate set at the maximum of 1.0 may result in training which is highly unstable and thus may not achieve even a local minima, or the network may take an inordinate amount of training time. If set at a low of 0.0, momentum is not considered and the network is more likely to settle into a local minimum. A process of "simulated annealing" is performed if the momentum rate starts high and is slowly shifted to 0 over a training session. Like other statistical and mathematical solutions, back propagation networks can be over- parameterized. This leads to the ability of the statistics to find parameters which can accurately compute the desired output at the expense of the systems ability to interpolate and compute appropriate output for different inputs.

2.3.4 Learning Iterations

This describes how often all training patterns are repeated to learn from these examples.

2.4 Backpropagation

Backpropagation is a strategy for teaching the network by comparing the result of a computed input with the expected output. The difference is the current error of the network. Since we only know the expected output of the last layer, namely the ouput layer, and do not know about the errors in the hidden layers, we need a strategy to minimize these hidden errors.
The Backpropagation is an iterative process. It searches a configuration of weights in the net, which results in the minimum of an error function on all training patterns. The Backpropagation-Algorithm can be divided in three phases. First there is the forward pass which propagates the computed signal for each input pattern through the network until it reaches the output layer. Then an error signal is determined by comparing the existing output value with the desired output value. Lastly there is the backward pass which calculates the error signal backwards through the net by changing the weight values successively. This process is executed until the error signal is below a specified value or another exit condition (e.g. number of learning iterations) is reached.
A very complete description of artificial neural networks is made by Kriesel (2007), (Kuan, 2006) also gives a short overview. A compact explanation is made by Russell and Norvig (2010).

3 Experiment

3.1 Problem categorization

As a whole we have 20 different number series, respectively five by each group. We classified these sequences in five different categories, based on their difficulty. The first category is defined by the fact that there is the same solution calculation between every number and this solution calculation is simple, e.g. +1. The second category is like the first, only there are more challenging calculations between each

number, e.g. $/2 + 1$. The third includes values which change according to a specific pattern, e.g. $+2^n$. The fourth category spans number series in which the calculation changes between at least two patterns but remain simple, e.g. $+2, *2$. The last category are the most difficult number series, the calculation changes between at least two patterns with more than one operator, e.g. $n^2 + 1, n^2 - 1$.
Table 1 shows the 20 chosen number series and the results of our experiment. Our small experiment was carried out by six test persons, which varied in age (19-48) and profession. The test setup was to present the participants with the 20 number series which they had to solve without time restriction. They had to write down their solution, the solution strategy and their thoughts during solving the series. They also could choose the order of solving the number series by themselves.

3.2 Results

All number series could be solved though not every testperson was able to solve every number series. With the help of the notices we concluded that number series with subtraction are considered difficult, but are solved without problems, while number sequences with addition are considered the easiest. Series with numbers of a wide range, e.g. $99, 18, 36$, are seen as difficult. Also multiplication (if not by two) is harder than subtraction and division and n-th powers are even harder. Even though most participants did not now the Fibonacci-Series they had no problem solving this sequence. We could also see that our test persons do not give wrong answers. Instead of guessing, they would rather say they couldn't find a solution than giving a wrong number.
An interesting observation was that other solutions could be found with a good explanation for the deviant pattern. So some of these short extracts of the number series are not distinct.

4 Testing the Artificial Neural Network

4.1 Software used for designing ANNs

We used two different systems to model our ANNs. Our first approach was using Membrain (Jetter, n.d.) due to the fact that it has a clearly arranged user interface and thusly allows amateurs a simple access. Also changes on our ANNs could be directly perceived. But soon we were faced with Membrains constraints regarding extensive tests of ANNs. Changes to settings which affect several neurons (e.g. changing normalization, different structure of network etc) were rather time consuming and not efficient.
Therefore we also used the java neural network framework Encog (HeatonResearch, n.d.) for constructing and testing our different ANNs. With this tool creating and testing new networks with different settings is much easier and faster, although there is no graphic illustration of the networks setup and changes due to learning process also as the fact that you need knowledge in java programming to be able to implement the neural network.

4.2 Preliminary Theoretical Considerations

This section introduces the values we chose for the variables from chapter 2.3

Table 1: The number series, their solution and empirical results (s = solved, u = unsolved, w = wrongly solved)

Number Series	Solution Strategy	Category	s	u	w
$2, 5, 8, 11 \ldots \rightarrow 20$	$+3$	1	5	1	0
$148, 84, 52, 36, 28 \ldots \rightarrow 24$	$/2, +10$	2	6	0	0
$3, 7, 15, 31, 63 \ldots \rightarrow 127$	$(*2 + 1)$	2	6	0	0
$2, 5, 9, 19, 37 \ldots \rightarrow 75$	$(*2 + 1), (*2 - 1)$	2	5	1	0
$16, 22, 34, 58, 106 \ldots \rightarrow 202$	$+(3 * n),\ n = 1, 2, 3, \ldots$	2	5	1	0
$2, 3, 5, 6, 7, 8, 10 \ldots \rightarrow 11$	n^2 *are missing*, $n = 1, 2, 3, \ldots$	3	5	1	0
$2, 3, 5, 9, 17 \ldots \rightarrow 33$	$+2^n, n = 1, 2, 3, \ldots$	3	5	1	0
$0, 1, 4, 13, 40 \ldots \rightarrow 121$	$+3^n,\ n = 1, 2, 3, \ldots$	3	5	1	0
$4, 5, 7, 11, 19 \ldots \rightarrow 35$	$+2^n,\ n = 1, 2, 3, \ldots$	3	6	0	0
$1, 1, 2, 3, 5 \ldots \rightarrow 8$	*Fibonacci*	3	5	1	0
$2, 5, 10, 17, 26 \ldots \rightarrow 37$	$+3, +5, +7, \ldots$	3	5	1	0
$9, 20, 6, 17, 3 \ldots \rightarrow 14$	-3 *over two numbers*	4	5	1	0
$1, 3, 6, 8, 16 \ldots \rightarrow 18$	$+2, *2$	4	5	1	0
$33, 30, 15, 45, 42, 21, 63 \ldots \rightarrow 60$	$-3, /2, *3$	4	5	1	0
$25, 27, 30, 15, 5, 7, 10 \ldots \rightarrow 5$	$+2, +3, /2, /3$	4	5	1	0
$11, 8, 24, 27, 9, 6, 18 \ldots \rightarrow 21$	$-3, *3, +3, /3$	4	5	1	0
$-2, 5, -4, 3, -6 \ldots \rightarrow 1$	$+7, -9$	4	5	1	0
$99, 18, 36, 9, 18 \ldots \rightarrow 9$	$-3^n, *2,\ n = 4, 3, 2, 1$	5	6	0	0
$10, 45, 15, 38, 20 \ldots \rightarrow 31$	$+5, -7$ *over two numbers*	5	5	1	0
$2, 3, 10, 15, 26 \ldots \rightarrow 35$	$n^2 + 1, n^2 - 1,\ n =$	5	4	1	1

Learning Rate:	0,01
Momentum:	0.1
Iterations:	10000

We tried different layer settings. We had at least one and maximum two hidden layers.All our tested network configuration have four input nodes and, as we expect one single output value, one output neuron.

4.3 Test arrangement

4.3.1 Input specification

In contrast to Ragni and Kleins work, that tested number series which have at least 20 numbers with values ± 1000, we only provide five input patterns and their solutions as teaching input respectively output to the network. This is very little data for a neural network to actually learn something, but it is comparable to the patterns we gave our human test subjects. Also some of the series would not allow to be continued that far.

4.3.2 Output specification

We can not expect to get integer values as an output from a neural network since neural networks just build approximations to functions. Therefore we interpret the rounded output as the answer given by the network. That means the actual error will be between 0 and ± 0.5. Applying the conditions from the previous experiment we accept a number series as solved if the network can compute the next value that would follow in the series. As an input we will give the according pattern, that has not been used for training. This is the same condition as in Ragni and Kleins work.
A learning example may look like this:

16; 22; 34; 58; 106
22; 34; 58; 106; 202
34; 58; 106; 202; 394
58; 106; 202; 394; 778
106; 202; 394; 778; 1526

Where each line is one teaching pattern. The last value is the expected output.

4.4 Results

In small number series the ANN mostly finds the correct next number, although many times it fits when we round down or up. As higher and complex the number series gets as more problems the ANN has to find the right solution. Sometimes the ANN just gets stuck on one number for all input patterns. It is hard to find a configuration that is able to solve a number series and for each number series a different configuration needs to be set up.
Table 2 shows the 20 chosen number series and the results of our network trying to solve them. We tried to solve the series with four different configurations in terms of Learning Rate, Momentum and layout. In summary, we could not solve twelve of the sequences, only eight could be solved by our various networks, each with the configuration which produced the lowest error.

5 Conclusion

With reference to our results we conclude that artificial neural networks are not yet comparable to the complexity of the human brain. At least not in the field of solving number series and not in a way we expected at first. Easy sequences like addition and subtraction can be solved correctly if there are only small gaps between. This means our categorization was quite wrong, at least as a rating of difficulty for the neural networks. Looking at the results we can see

Table 2: Tested sequences and configuration with lowest error (M = Momentum, L = Learning Rate, H = Hidden Layers, E = Error)

Number Series	Category	Configuration
$2, 5, 8, 11 \ldots \rightarrow 20$	1	*Not solved*
$148, 84, 52, 36, 28 \ldots \rightarrow 24$	2	*Not solved*
$3, 7, 15, 31, 63 \ldots \rightarrow 127$	2	*Not solved*
$2, 5, 9, 19, 37 \ldots \rightarrow 75$	2	*Not solved*
$16, 22, 34, 58, 106 \ldots \rightarrow 202$	2	*Not solved*
$2, 3, 5, 6, 7, 8, 10 \ldots \rightarrow 11$	3	$M = 0,01; L = 0,1; H = 4; E = -0,04$
$2, 3, 5, 9, 17 \ldots \rightarrow 33$	3	*Not solved*
$0, 1, 4, 13, 40 \ldots \rightarrow 121$	3	*Not solved*
$4, 5, 7, 11, 19 \ldots \rightarrow 35$	3	*Not solved*
$1, 1, 2, 3, 5 \ldots \rightarrow 8$	3	*Not solved*
$2, 5, 10, 17, 26 \ldots \rightarrow 37$	3	*Not solved*
$9, 20, 6, 17, 3 \ldots \rightarrow 14$	4	$M = 0,05; L = 0,1; H = 12,6; E = 0,90$
$1, 3, 6, 8, 16 \ldots \rightarrow 18$	4	$M = 0,01; L = 0,5; H = 4,8; E = -0,39$
$33, 30, 15, 45, 42, 21, 63 \ldots \rightarrow 60$	4	$M = 0,1; L = 0,1; H = 12,6; E = 0,12$
$25, 27, 30, 15, 5, 7, 10 \ldots \rightarrow 5$	4	$M = 0,01; L = 0,5; H = 8,4; E = -0,06$
$11, 8, 24, 27, 9, 6, 18 \ldots \rightarrow 21$	4	$M = 0,01; L = 0,1; H = 4,8; E = -0,01$
$-2, 5, -4, 3, -6 \ldots \rightarrow 1$	4	$M = 0,1; L = 0,1; H = 4,8; E = -0,01$
$99, 18, 36, 9, 18 \ldots \rightarrow 9$	5	$M = 0,05; L = 0,1; H = 6; E = -0,01$
$10, 45, 15, 38, 20 \ldots \rightarrow 31$	5	*Not solved*
$2, 3, 10, 15, 26 \ldots \rightarrow 35$	5	*Not solved*

that all number series of category four can be solved. When looking at the number series we can see that the gaps betwenn values are rather small. When there are gaps of about more than two digits between one number and its successor or a multiplication has to be applied to our neural networks were mostly incapable to solve the sequence rightly. Especially n-th degrees cannot be solved by our tested configurations. Our conclusion is that ANNs are not powerful enough to solve difficult sequences and are in no case able to reflect the human brains processes during solving number series at all. But we can still interpret our networks as a mental model for solving number series. During the experiment with our test persons we saw that they were not able to get the number series right right away. They had to try different strategies for each new number series. We find a resemblance to this behavior in the fact that we need differently structured and with different learning options set up networks to find a merely good solution. But in this finding there is also one big difference. Humans would not give wrong answers. They rather give none. Opposing that artificial neural networks, by design, always give an answer.

5.1 Problems occurred

Neural networks are designed to be trained with a big training set. Our number series were probably way too short. The networks are able to learn the series by heart, but they are not able to continue them. The problem of overtraining is explained more specifically in Oklahoma and Abraham (2005) Especially alternating patterns cannot be solved. This is most probably due to the fact that each situation that may occur in the series is trained only once or maybe twice. We are aware of these problems, but we wanted to compare the machine to a humans ability to solve number series with (nearly) same conditions. One interesting problem that occurred was that the normalization used to translate the input values to values between -1 and 1 has a big impact on the networks output. We assume that this is the fault of insufficient precision even in the double values of Java.

References

HeatonResearch. (n.d.). *Encog.* Available from `http://www.heatonresearch.com/encog`

Hebb, D. O. (1949). *The organization of behavior.* Wiley and Sons, New York.

Jetter, T. (n.d.). *Membrain.* Available from `http://www.membrain-nn.de`

Kriesel, D. (2007). *A brief introduction to neural networks, zeta version.* (available at http://www.dkriesel.com)

Kuan, C.-M. (2006). *Artificial neural networks.*

Oklahoma, A. A., & Abraham, A. (2005). *Artificial neural networks.*

Russell, S., & Norvig, P. (2010). Artificial intelligence: A modern approach. In (2nd edition ed., p. 727-737). Prentice-Hall, Englewood Cliffs, NJ.