SOLVING NUMBER SERIES
Architectural Properties of Successful Artificial Neural Networks

Marco Ragni and Andreas Klein
Center for Cognitive Science, Friedrichstr. 50, 79098 Freiburg, Germany

Keywords: Number series, Artificial neural networks.

Abstract: Any mathematical pattern can be the generation principle for number series. In contrast to most of the application fields of artificial neural networks (ANN) a successful solution does not only require an approximation of the underlying function but to correctly predict the exact next number. We propose a dynamic learning approach and evaluate our method empirically on number series from the Online Encyclopedia of Integer Sequences. Finally, we investigate research questions about the performance of ANNs, structural properties, and the adequate architecture of the ANN to deal successfully with number series.

1 INTRODUCTION

Number series can represent any mathematical function. Take, for instance

(i) 4, 11, 15, 26, 41, 67, 108, 175, ...

(ii) 5, 6, 7, 8, 10, 11, 14, 15, ...

The first problem (i) represents the Fibonacci series[1]. The second problem (ii) represents two nested series[2] one starting with 5 the other with 6. In number series principally any computable function can be hidden and the set of operators is not restricted. Mathematically, number series are described by a function $g : \mathbb{N} \to \mathbb{R}$ and can contain the periodic sinus function, exponential functions and polynomial functions, and even the digits of π – there is no restriction whatsoever on the underlying function. There is an Online Encyclopedia of Integer Sequences (Sloane, 2003) (OEIS)[3] available online with a database to download and a Journal of Integer Sequences[4]. A small subset of number series problems are used in intelligence tests for determining mathematical pattern-recognition capabilities. Inevitably, identifying patterns requires learning capabilities. As artificial neural networks (ANN) are the silver bullet in Artificial Intelligence for learning approaches – it seems only consequent to test how far we can get with a first approach in untackling the mystery of number series.

Not all number series have a similar difficulty, some problems are easy to solve while others are nearly impossible to deal with. A simple measure might be to identify the underlying function and to count the number of operations. However, the number of operations is typically a symbolic measure (Marr, 1982) and does not necessarily correspond with architectural properties in the neural networks. Solving number series pose an interesting problem as it combines deductive and inductive reasoning. Deductive reasoning is necessary to identify the rule between the given numbers, while inductive reasoning is necessary to predict the next number. So far neural networks have been used for approximations, but rarely for predicting exact number – a necessary condition for solving number series. Related to these problems is the prediction of time series (Martinetz et al., 1993; Connor et al., 1994; Farmer and Sidorowich, 1987), but the numbers to be predicted are integers (and not – as usual – reals), which can pose even more difficulty, the numbers adhere to mathematical laws and are not empirically observed. In this sense, we are much closer to the research fields trying to reproduce "logical properties" (like propositional or boolean reasoning (Franco, 2006)) and not time series analysis. Other related approaches use ANNs for example for teaching multiplication tables by training on empirical findings of pupils performances to identify and provide a guideline for pupils educational materials (Tatuzov, 2006). In this case, the networks used, simulate the pupils behavior and had to learn multiplication skills. In contrast to this approach, we

[1] $a_{n+2} := a_{n+1} + a_n$ with $a_1 := 4$ and $a_2 := 11$
[2] $a_{n+2} := a_{n+1} + (a_{n+1} - a_n) + 1$
[3] http://oeis.org/
[4] http://www.cs.uwaterloo.ca/journals/JIS/

Ragni M. and Klein A..
SOLVING NUMBER SERIES - Architectural Properties of Successful Artificial Neural Networks.
DOI: 10.5220/0003682302240229
In *Proceedings of the International Conference on Neural Computation Theory and Applications* (NCTA-2011), pages 224-229
ISBN: 978-989-8425-84-3

are not interested in modeling error trials.

In the following, we propose a method using artificial neural networks and – to the best of our knowledge – a novel dynamic approach to solve number series problems. The approach is evaluated against over 50.000 number series taken from the OEIS. To analyze the structural properties of successful networks and to identify the best configurations, we will systematically vary the architecture of artificial neural networks.

2 DYNAMIC LEARNING

For our attempt to solve number series with artificial neural networks (McCulloch and Pitts, 1943; Russell and Norvig, 2003), we use three-layered networks (cf. Figure 1) with error back-propagation and the hyperbolic tangent as the activation function. If x is the length of the largest number in the given part of a number series, we use

$$f_i = \frac{n}{10^x} \tag{1}$$

as input function to project the interval of numbers in the number series to the interval $[-1,1]$. Analog, we project the output of the networks back to the original scale with the function

$$f_o = n * (10^x) \tag{2}$$

with the same x as for input and rounded the result to get an integer value. We trained the networks on all but the last given number, which they had to predict. To achieve this goal we trained the networks rather to extrapolate the given number to extend the number series then to classify or interpolate it. The weights were initially randomly assigned with small positive and negative values between zero and one. A momentum factor of 0.1 was used.

For our analysis, we systematically varied the learning rate, the number of input nodes, and the number of nodes within the hidden layer. Taken together the ANNs can be described by the formula $f(i,h,l)$, with the number of input nodes i, the number of nodes within the hidden layer h, and the learning rate l. These variations should allow for a comparison of the different ANNs, provide us with enough variability to successfully solve number series and to identify successful architectures for ANNs solving these problems.

To generate patterns for training and testing a network, we built patterns as tuples of training values and one target value. The number of training values of a pattern is equivalent to the number of input nodes i of the network used. Starting with the first number, a subsequence of training values was shifted through the number series. As corresponding target value, the next number of the subsequence was used (cf. Table 1).

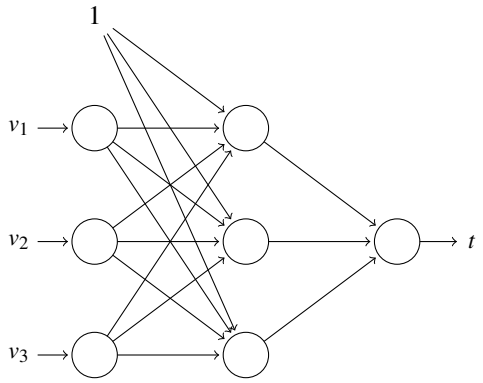


Figure 1: Architecture of an ANN with 3 input nodes and 3 hidden nodes we used to predict the next value of a subsequence of a number series. The ANNs were trained on three successive numbers of a sequence a_1 to a_n (cf. Table 1).

Table 1: Pattern generation example for an ANN using the dynamic approach with three input nodes and a number series of seven given numbers $(n_1,...,n_7)$ with the n_8 to predict. Four training patterns $(p_1,...,p_4)$ are used each with three training values (v_i, v_{i+1}, v_{i+2}) and one target value $t = n_{i+3}$.

Pattern	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8
p_1	v_1	v_2	v_3	t				
p_2		v_2	v_3	v_4	t			
p_3			v_3	v_4	v_5	t		
p_4				v_4	v_5	v_6	t	
p_5					v_5	v_6	v_7	?

Since the last given value of the number series with length n remains as target value and we need at least one training and one test pattern, the maximum length of a subsequence of training values is $n-2$. Hence, for a network configuration with m input nodes exactly $n-m$ patterns were generated. Consequently, the first $(n-m)-1$ patterns were used for training, while the last one remained for testing and thus predicting the last given number of the sequence. Therefore, the last given number of the number series – the main target value – was never used for training the ANNs.

The training of the network was iterated on the patterns for a various number of times. We systematically varied the iterations. After the training phase of the network we tested if the network captured the inherent pattern with the last pattern without a given target value and compared it to the actual number of the number series.

Table 2: Results of an empirical analysis with 17 participants for 20 number series and the performance of 840 network configurations with five variants of training iterations. A step–width of 0.125 for the learning rate was used.

ID	Number Series	Correct responses	Incorrect responses	No response	No. of solving configurations with iterations:				
					0.5k	1k	5k	10k	15k
E001	12,15,8,11,4,7,0,3	15		2	306	385	475	530	554
E002	148,84,52,36,28,24,22,21	12	2	3	555	637	670	689	683
E003	2,12,21,29,36,42,47,51	14	1	2	405	440	502	539	551
E004	2,3,5,9,17,33,65,129	13	1	3	3	7	61	192	218
E005	2,5,8,11,14,17,20,23	9	3	5	581	618	659	667	675
E006	2,5,9,19,37,75,149,299	6	4	7	0	0	0	0	0
E007	25,22,19,16,13,10,7,4	16		1	562	615	648	667	670
E008	28,33,31,36,34,39,37,42	17			121	183	315	332	342
E009	3,6,12,24,48,96,192,384	13	1	3	0	0	0	0	0
E010	3,7,15,31,63,127,255,511	12	3	2	0	0	0	0	0
E011	4,11,15,26,41,67,108,175	8	1	8	6	14	32	22	18
E012	5,6,7,8,10,11,14,15	10	1	6	83	91	65	114	202
E013	54,48,42,36,30,24,18,12	16	1		274	299	338	376	401
E014	6,8,5,7,4,6,3,5	16		1	134	169	198	219	213
E015	6,9,18,21,42,45,90,93	14	1	2	48	24	94	101	103
E016	7,10,9,12,11,14,13,16	14		3	111	202	380	404	409
E017	8,10,14,18,26,34,50,66	13	1	3	57	46	30	29	24
E018	8,12,10,16,12,20,14,24	17			37	75	41	51	71
E019	8,12,16,20,24,28,32,36	15		2	507	546	594	613	634
E020	9,20,6,17,3,14,0,11	16		1	255	305	397	406	411

2.1 Human Performance

In a previous experiment (Ragni and Klein, 2011) with humans we tested 20 number series to evaluate reasoning difficulty and to benchmark the results of our ANN. All number series can be found in Table 2. We only briefly report the results. Each of the 17 participants in this paper and pencil experiment received each number series in a randomized order and had to fill in the last number of the series. The problems differed in the underlying construction principle and varied from simple additions, multiplications to combinations of those operations. Also nested number series like $5,6,7,8,10,11,14,15\ldots$ (cp. (ii) from the introduction) were used.

For our analysis with ANNs we varied the learning rate of the configuration ($f(i,h,l)$) between 0.125 and 0.875 with a step-width of 0.125 and later on with a step-width of 0.1 ranging from 0.1 to 0.9 ($0.125 \leq l \leq 0.875$ (or $0.1 \leq l \leq 0.9$)). The number of nodes within the hidden layer was iterated from one to twenty ($1 \leq h \leq 20$). The number of input nodes was varied between one and six nodes ($1 \leq i \leq 6$). For all configurations only one output node was used. The number of training itations was varied in five steps starting with as low as 500 iterations on each pattern before testing. Then raised the number in four steps over 1.000, 5.000 and 10.000, up to 15.000 iterations.

Table 3: Results of solving configurations with a step–width of 0.1 for the learning rate out of 1080 configurations.

ID	No. of solving configurations with iterations:				
	0.5k	1k	5k	10k	15k
E001	410	496	626	669	715
E002	698	800	863	871	887
E003	516	575	663	708	733
E004	8	11	83	246	283
E005	723	800	843	854	882
E006	2	4	0	0	0
E007	711	756	828	845	861
E008	165	238	409	425	437
E009	0	0	0	0	0
E010	0	0	0	0	0
E011	6	16	45	32	29
E012	103	111	88	151	257
E013	334	354	437	477	506
E014	166	202	274	277	285
E015	61	48	127	127	132
E016	155	259	485	527	523
E017	78	64	41	25	25
E018	38	108	59	71	87
E019	646	667	743	775	797
E020	304	400	517	521	528

There are three number series which were not solved by any ANN configuration tested with a

0.125 step-with configuration. All others could be solved (cf. Table 2). With a smaller step–width for the learning rate (0.1) the number series $2,5,9,19,37,75,149\ldots$ was solved too, but the two other number series remain unsolved (cf. Table 3).

Comparisons between the different configurations show a negligible difference between the number of iterations (cf. Table 2 and 3). It seems that 1.000 iterations might be already a good approximation for solving number series.

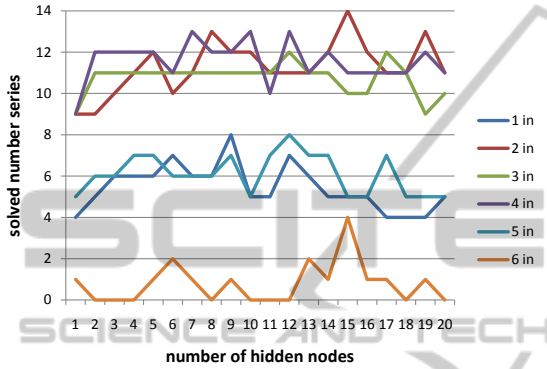


Figure 2: Results for ANN configurations with $lr = 0.125$, $1 \leq i \leq 6$, and 10.000 training iterations applied to the 20 problems depicted in Table 2.

The input nodes divide the number of solutions in three classes (cf. Figure 2): We get the worst results for ANNs with six input nodes. One and five input nodes span the next level, and finally two to four input-nodes solve the problems considerably well, with a peak for an ANN with two input nodes and about 15 hidden nodes. In average, if we increase the learning rate the number of solved tasks decreases with an increasing number of hidden nodes. Networks with only one input node showed a slightly different behavior. For these networks, the number of solved tasks increases with the learning rate and decreases with learning rates above 0.625. Taken together this first analysis already shows that simple number series can be adequately solved. An interesting result is that the higher the number of input nodes is – the worser are the results. We will see that this does not hold for arbitrary number series from the OEIS.

2.2 The OEIS Number Series

To assess the power of our dynamic approach we chose to use the number series from the OEIS database. The OEIS database contains a total of 187.440 number series. We selected those number series which consist of at least 20 numbers (to be able to apply our dynamic training method) with values smaller or equal to ± 1.000. This constraint is satisfied by exactly 57.524 number series, which were chosen to be used to benchmark our ANN approach with dynamic learning.

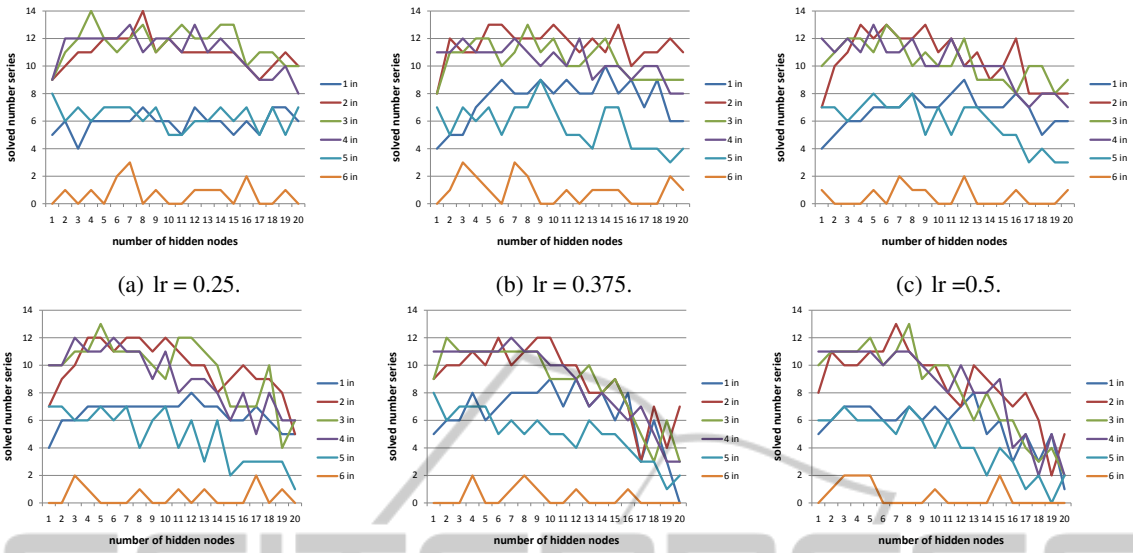
Due to the large number of problems, we varied the learning rate ($0.125 \leq l \leq 0.875$), the number of nodes within the hidden layer ($1 \leq h \leq 7$) of the configuration ($f(i,h,l)$). We used four, eight and twelve input nodes and only one output node. For all configurations, the number of training itations was 1.000.

The best ANN, in the sense of solving the most number series, with four input nodes (and two hidden nodes and a learning factor of .75) was able to solve 12.764 number series problems and all 49 tested ANN configurations with four input nodes together were able to solve 26.951 of the number series. The best ANN, in the sense of a minimal deviation over all tasks, was achieved by a network with four nodes in the input layer and two in the hidden layer and a learning rate of .375. The deviation of the prediction and the actual value, summed over all tasks, was 859.144. The deviation of the network, solving the most tasks was 868.506.

The best ANN with 8 input nodes (and two hidden nodes and a learning rate of .75) solved 13.591 number series tasks and all 49 settings together were able to solve 31.914 of the total of 57.524 tasks. The minimal deviating network was the analog as in the case of four nodes in the input layer. Its overall deviation was 904.134. The deviation of the network, solving the most tasks with eight nodes in the input layer, was 926.262.

Considering the configurations with 12 input nodes, the most number series (14.021) were solved by a configuration with two nodes in the hidden layer and a learning rate of .75. Over all number series this configuration deviated by 992.875. The minimal deviating configuration in this case was the one with two nodes in the hidden layer and a learning rate of .25. Its deviation was 962.510. All configurations (with $i = 12$) together, were able to solve 33.979 number series correctly.

The configurations solving the most number series are summed up in Table 4, showing that with an increasing number of input nodes the number of solved problem increases, too. This is also true for the number of solved number series over all tested configurations. In contrast, as shown in Table 5, with an increase of the input nodes the deviation of each task summed over all tasks rises. Combining these results, this means that with an increasing number of input nodes more number series could be solved, but the prediction of the unsolved problems deviate more from the correct solution.



(a) lr = 0.25. (b) lr = 0.375. (c) lr =0.5.

(d) lr = 0.625. (e) lr = 0.75. (f) lr = 0.875.

Figure 3: Results for ANN configurations (lr= learning rate, in = input nodes) with 10.000 training iterations applied to the 20 problems depicted in Table 2.

To capture the performance of this approach, we additionally analyzed deviations of ± 5 and ± 10 from the correct solution. The configurations solving the most problems and also the number of solved problems over all configurations rise drastically, which is also shown in Table 4.

Table 4: Configurations with 4, 8 and 12 input nodes (in), solving the most number series of the OEIS database. Results are shown for exact solutions, ± 5, and ± 10 around the exact solution.

Input	Exact	± 5	± 10
4	12.764 with h=2, lr=.75	39.003 with h=2, lr=.75	44.848 with h=5, lr=.875
8	13.591 with h=2, lr=.75	39.065 with h=2, lr=.625	45.064 with h=4, lr=.875
12	14.021 with h=2, lr=.75	39.052 with h=2, lr=.375	45.086 with h=2, lr=.75

Table 5: Configurations with the smallest deviation summed over all number series of the OEIS database.

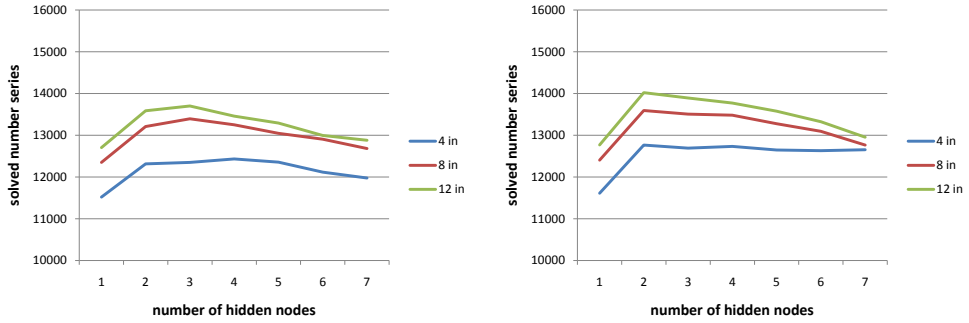
No. input nodes	deviation and configuration
4	859.144 with $h = 2$, lr=.375
8	904.134 with $h = 2$, lr=.375
12	962.510 with $h = 2$, lr=.25

Again, solving means, they were able to correctly predict the next number of the series, even though they had never trained it. Figure 4 depicts the results of the ANNs based on the OEIS problems.

3 CONCLUSIONS

The problem of selecting an adequate neural network architecture for a given problem has come recently more and more to the research focus (Gómez et al., 2009). Similar questions have been lately investigated for Boolean functions (Franco, 2006). We followed this research line in dealing with number series. To identify the adequate structures, we systematically varied input nodes, hidden nodes, and the learning rate to compare the different artificial neural networks structures. Using a dynamic learning approach, we are able to predict 90% correctly (18 of 20) of simple number series (which typically appear in intelligence tests). If we use the above specified subset of the OEIS database as a benchmark we are still able to solve about 59% of the number series correctly (about 33.979 of 57.524 number series, cp. Table 6). Relaxing the goal to compute the correct number and allowing deviations of ± 10 allows to capture 50.139 number series.

A first conclusion we can draw is that the structure of the artificial neural networks can determine the success of solving a number sequence – there is a systematic pattern between learning rate, input nodes and the number of nodes within the hidden layer – showing that 2-4 input nodes and about 5-6 hidden nodes provide the best framework for solving typical intelligence test like number series. If the number series can be mathematically much more complex number



(a) lr = 0.25. (b) lr = 0.75.

Figure 4: Results for ANN configurations with 1.000 training iterations applied to a subset of the OEIS database.

series, then eight input nodes seem much better than four input nodes while the number of hidden nodes remain stable.

Table 6: Number of solved number series of the OEIS database over all 49 tested configurations. Results are shown for exact solutions, ±5, and ±10 around the exact solution.

No. input nodes	exact	±5	±10
4	26.951	44.689	48.176
8	31.914	46.349	49.580
12	33.434	47.076	49.931

The right artificial neural networks seem – as a method – powerful enough to expand into one of the still privileged realms of human reasoning – to identify patterns and solve number series successfully. The structure of the used ANNs can provide useful insights.

Future work will systematically investigate to what extend other types of artificial neural networks and approaches show a better performance than back propagating ones used in this approach.

REFERENCES

Connor, J. T., Martin, R. D., and Atlas, L. E. (1994). Recurrent neural networks and robust time series prediction. *IEEE Transaction on Neural Networks*, 51(2):240–254.

Farmer, J. D. and Sidorowich, J. J. (1987). Predicting chaotic time series. *Phys. Rev. Lett.*, 59(8):845–848.

Franco, L. (2006). Generalization ability of boolean functions implemented in feedforward neural networks. *Neurocomputing*, 70:351–361.

Gómez, I., Franco, L., and Jérez, J. M. (2009). Neural network architecture selection: Can function complexity help ? *Neural Processing Letters*, 30(2):71–87.

Marr, D. (1982). *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. Freeman, New York.

Martinetz, T. M., Berkovich, S. G., and Schulten, K. J. (1993). "Neuralgas" network for vector quantization and its application to time-series prediction. *IEEE Transaction on Neural Networks*, 4:558–569.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Ragni, M. and Klein, A. (2011). Predicting numbers: An AI approach to solving number series. In Edelkamp, S. and Bach, J., editors, *KI-2011*.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition.

Sloane, N. J. A. (2003). The on-line encyclopedia of integer sequences. *Notices of the American Mathematical Society*, 50(8):912–915.

Tatuzov, A. L. (2006). Neural network models for teaching multiplication table in primary school. In *IJCNN '06. International Joint Conference on Neural Networks, 2006*, pages 5212 – 5217, Vancouver, BC, Canada.