Artificial Neural Networks: Exercise session 2

Stijn Staring (r0620003)

May 23, 2021

1 Exercises of Section 1: Hopfield Network

Create a Hopfield network with attractors T = [1 1;-1 -1; 1 -1]T and the corresponding number of neurons. You can use script rep2 as a basis and modify it to start from some particular points (e.g. of high symmetry) or to generate other numbers of points. Start with various initial vectors and note down the obtained attractors after a sufficient number of iterations. Are the real attractors the same as those used to create the network? If not, why do we get these unwanted attractors? How many iterations does it typically take to reach the attractor? What can you say about the stability of the attractors?

In Figure 1 all the found attractors in the 2D plane can be seen which are more than the three desired ones that were specified to the Hopfield network. It can thus be concluded that despite the "newhop" matlab function tries to design a Hopfield network with the minimum of unspecified target attractors, they often occur. Table **??** shows the found attractor points. When designing the energy function for the system that has a corresponding local minima at the attractor points, it can be that also unwanted attractor points become local minima. When the assumption of no bais term of the model is made, these correspond to the negative of the desired attractors. Also, a mixture of the desired states are possible. The amount of iterations it takes to reach an attractor depends on the start position. All the symmetric starting points of Figure 1 reached an attractor in less then 5 iterations. However, the amount of iterations can dramatically increase when the starting point is very close to an unstable attractor. For example, when the starting point chosen is $[-1; -10^{-20}]$, it takes 308 iterations to reach the stable attractor $[-1; -1]$. When the system is in a stable attractor and a little noise is added the system will move back to the stable attraction point. An unstable attractor on the other hand will move away when this happens. The stability of the different attractor points can be seen in Table 1.
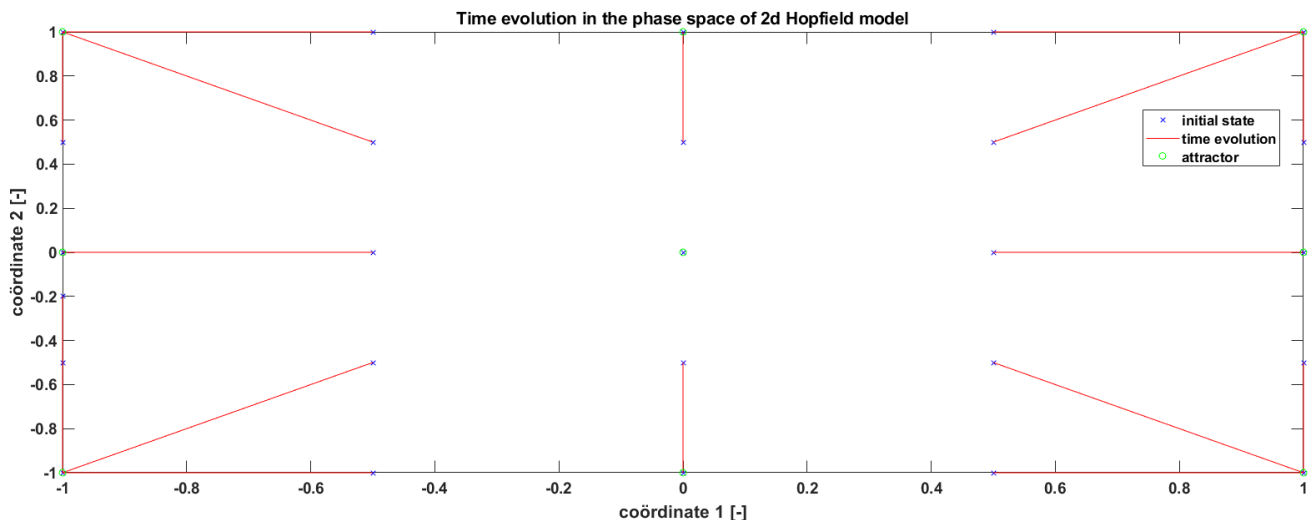

Figure 1: All the different attractor points found in a 2D plane.

Do the same for a three neuron Hopfield network. This time use script rep3.
Table 2 shows an overview of the found attractors in a 3D space. Only the three specified target attractors were found during the experimentation with different start positions. All three are stable points.

Attractor	Point	Stability
Attractor 1	$[1; 1]$	Stable
Attractor 2	$[-1; -1]$	Stable
Attractor 3	$[1; -1]$	Stable
Attractor 4	$[-1; 1]$	Stable
Attractor 5	$[0; 0]$	Unstable
Attractor 6	$[0; 1]$	Unstable
Attractor 7	$[0; -1]$	Unstable
Attractor 8	$[1; 0]$	Unstable
Attractor 9	$[-1; 0]$	Unstable

Table 1: Overview of the different attractor points found in a 2D plane.

Attractor	Point	Stability
Attractor 1	$[1; 1; 1]$	Stable
Attractor 2	$[-1; -1; 1]$	Stable
Attractor 3	$[1; -1; -1]$	Stable

Table 2: Overview of the different attractor points found in a 3D plane.

The function hopdigit creates a Hopfield network which has as attractors the handwritten digits 0,...,9. Then to test the ability of the network to correctly retrieve these patterns some noisy digits are given to the network. Is the Hopfield model always able to reconstruct the noisy digits? If not why? What is the influence of the noise on the number of iterations?

The noise that is added to the digits is gaussian noise. The noise variable determines the standard deviation of the gaussian that is sampled to retrieve the noise that should be added. It can be concluded that when the noise is too much, the Hopfield model is not able to reconstruct the correct digits. The reason is because when the noise is too much, the starting point can become closer to an other digit than the original one and the final prediction will end up in the wrong attractor and thus the wrong digit. Increasing the amount of iterations will not solve this issue. When there is more noise, the system needs more iterations to reach the attractor points.

2 Exercises of Section 2: Long Short-Term Memory Networks

Train a MLP with one hidden layer after standardizing the data set. Investigate the model performance with different lags and number of neurons. Explain clearly how do you tune the parameters and what is the influence on the final prediction. Which combination of parameters gives the best performance on the test set?

In this section the time-serie signal included in the Santa Fe dataset will be forecasted. A training set of 1000 samples and a test set of 100 samples are given. The first step is to standardize the training data by subtracting the mean and dividing it afterwards by the standard deviation of the time signal. This has as effect that the time-serie is shifted to oscillate around zero with a standard deviation of 1. The effect of standardization is in general that it enhances the learning process during training.

First a regression predictor is developed using a feedforward MLP neural network. The neural network consists out of a single hidden layer which uses as activation function "tanh". The amount of input neurons is dependent on the lag value. The lag value determines how many past samples of the time-serie are taken into account to make the forecast. There is only one output neuron that gets inputs from all the hidden neurons and uses the "purelin" activation function. One time-serie sample is predicted at the time and to make a prediction for a longer time horizon, multiple predictions have to be sequentially performed using previous ones as inputs for the next.

A parameter search is conducted to determine the amount of neurons in the hidden layer, the lag value and the regulation parameter. The regulation parameter determines the importance of the $L2$ norm of the weights in the objective function. As update rule the "Gradient descent" method is used. Table 3 gives an overview of the different designs that were tested. 10% of the training set is assigned as the validation set to perform early stopping to avoid overfitting.

It was found that the best results were obtained when using a regulation parameter of 0.4, 9 hidden neurons and a lag value of 16. The MSE value is calculated as the average of 5 neural networks simulations that were

Design Variable	Domain
Regulation parameter	$0:0.2:0.4$
Amount of hidden layers	1
Amount of neurons in hidden layer	$1:2:9$
Learning rule	Gradient descent
Training time per training session	$5s$
Max amount of sequential increases of the validation error in early stopping	15
Goal of training error	10^{-3}
Amount of iterations to calculate average MSE	5

Table 3: The different design variables and their corresponding domain.

trained with the same settings according to Table 3, but with different randomly initialised weights. The error is the difference between the target values and the predicted values on the validation set that were retrieved by sequentially predicting the next value while using previous predictions as new inputs. The selected design of the neural network is now used to learn for 5 minutes on the training data. This network is afterwards used to do predictions on the 100 samples of the test set which can be seen in Figure 2. The figure shows that in the beginning the first three peaks can be reasonably approximated but the error increases rapidly when the predictions are further in time due to a cumulative error effect. However this is not the only reason for the bad behaviour of the forecast after sample 50. A test is performed where a forecast starts from sample 50 what is just at the moment that the test set shows a reset of the magnitude of the oscillation. It can be seen that the neural network gives still unsatisfactory results because it can not identify the "reset" but predicts big peaks instead. It can be concluded that MLP regression neural network is only reasonably able to capture the periodic peak behaviour of the Santa Fe time-serie. The MSE value of the prediction on the test set in Figure 2 is 2.54×10^3.
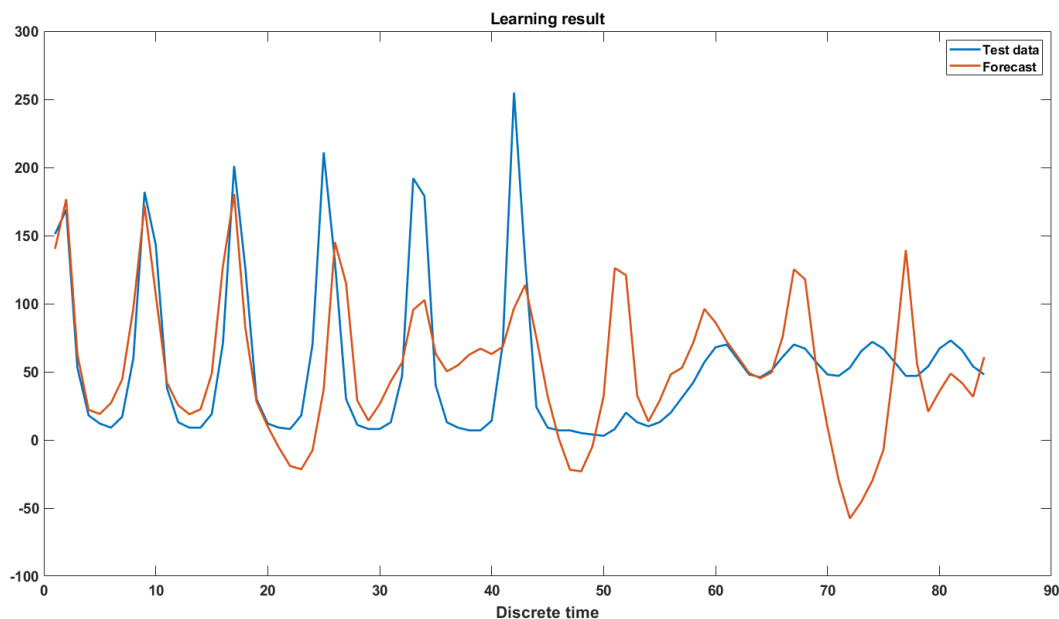

Figure 2: The forecasting result on the test set with the selected Neural Network that is trained for 5 minutes.

Implementation of LSTM
Next, a LSTM (long short term memory) neural network will be assessed to model the Santa Fe data set instead of the MLP regression neural network that was used in the previous section. A LSTM has as advantage over a standard RNN that it can better handle the vanishing gradient problem and LSTM is not suffering as much of a short term memory. Therefore, it can longer take important aspects of the presented time-serie into account when working on a new prediction. LSTM makes use of three gates to learn which data of the sequence is important and should be remembered. In connection of these gates is a cell state which serves as a memory state and a connection to transport relative information throughout the different time steps of the time-serie. The forget gate decides what information should be kept or removed from the new input and hidden state. The input gate updates the cell state and the output gate decides what the next hidden state will be.
There are now four different weight matrices and thus more parameters that have to be learned. One form the forget gate, two from the input gate and one from the output gate. It can be noted that in order to train all these weights, the network is more data hungry than a simple RNN.

Design Variable	Value
Max epochs	250
Lag value	16
Hidden neurons	200
Gradient threshold	1
Initial Learning rate	0.005
Learning rate drop period	125
Learning rate drop factor	0.2

Table 4: The different design variables and their set value for the LSTM neural network.

The parameters of the LSTM are picked on sight without a parameter tuning as was done for the MLP regression neural network in Figure 2. The parameters of the LSTM are summarized in Table 4. Figure 3 shows the result of the trained LSTM neural network on the test set. Previous predictions are used as inputs for predictions further in time. It can be seen that the network is able to predict the different periodic peaks, to capture the rising magnitude of these peaks and the reset behaviour of the magnitude. However, Figure 3 also shows that the prediction of the amount of peaks is not completely correct. Also, the effect of the time horizon which causes the error to increase further in time, can be seen. The capturing of the reset in magnitude of the peaks proofs that the LSTM can better remember important information on a long term basis than the MLP regression neural network. The MSE value that is found on the test set using the LSTM neural network is 1.67×10^3 which is in comparison with the MSE of 2.54×10^3 on the MLP regression neural network a much better result even though a parameter search was performed the latter.
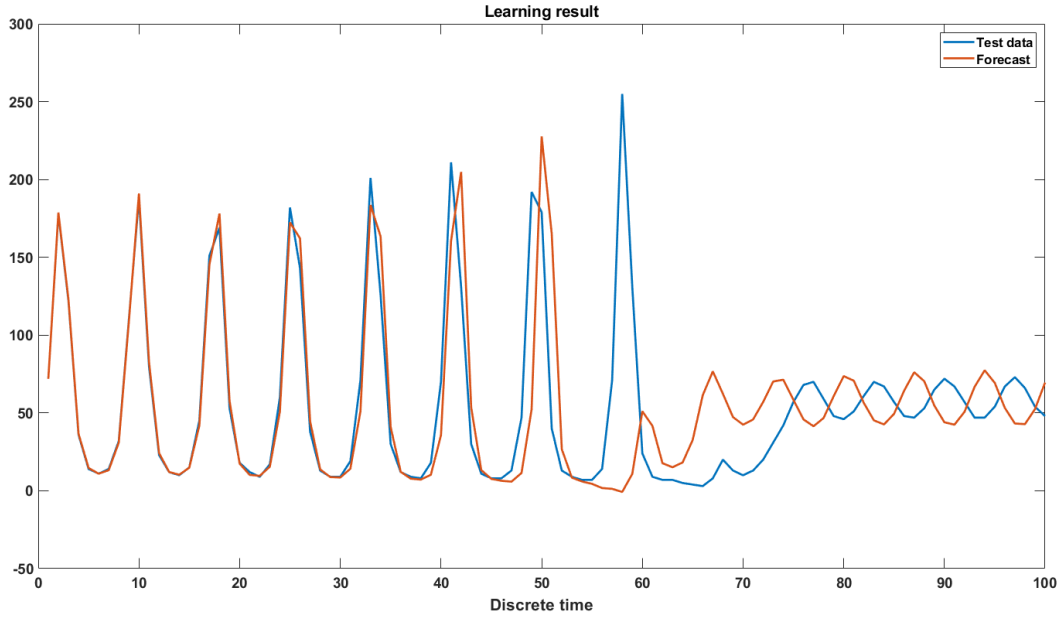

Figure 3: The forecasting result on the test set with the LSTM with parameters as indicated in Table 4 and that is trained for 250 epochs.