An assignment in neural networks in Åbo Akademi.

Marko Grönroos (Email: magi@utu.fi)
Christian Lehtinen (Email: jcleh@utu.fi)

Department of Computer Science in the University of Turku, 1993

Contents

1. Introduction 2
    1.1. The problem 2
2. Test runs 4
    2.1. Testing methods 4
    2.2. Test runs with our simulator 5
    2.3. Test runs with SNNS 12
3. Summary and conclusions 15

1. Introduction

We wanted to make a network that predicts how a data series will continue in future. The requirements for the data set were that it's x-values should be ascending and discrete (or they could be made such by preprocessing). The y-values should be cyclic or otherwise more sensible than totally random.

1.1. The problem

The set we chose for our study is the *index of cost of living* in Finland. The size of the set is 53 months, from January 1989 to June 1993.

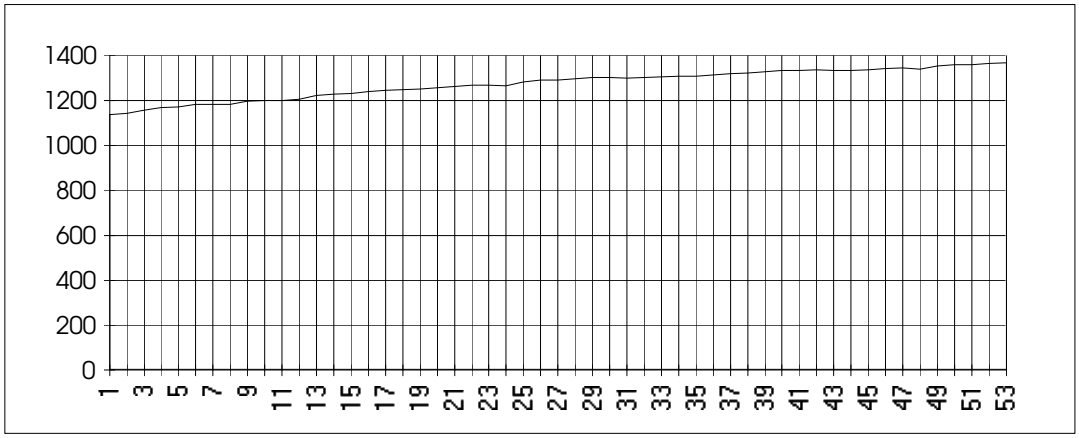

Chart 1.1. The original index curve. The x-axis is the number of month since January 1989 and the y-axis is the index. Index is a relative value, where the year 1952 is 100.

The curve is not suitable input for backpropagation network in this form. BPN is not good in processing absolute values. Therefore we took a differential from the index and got this much more promising curve:

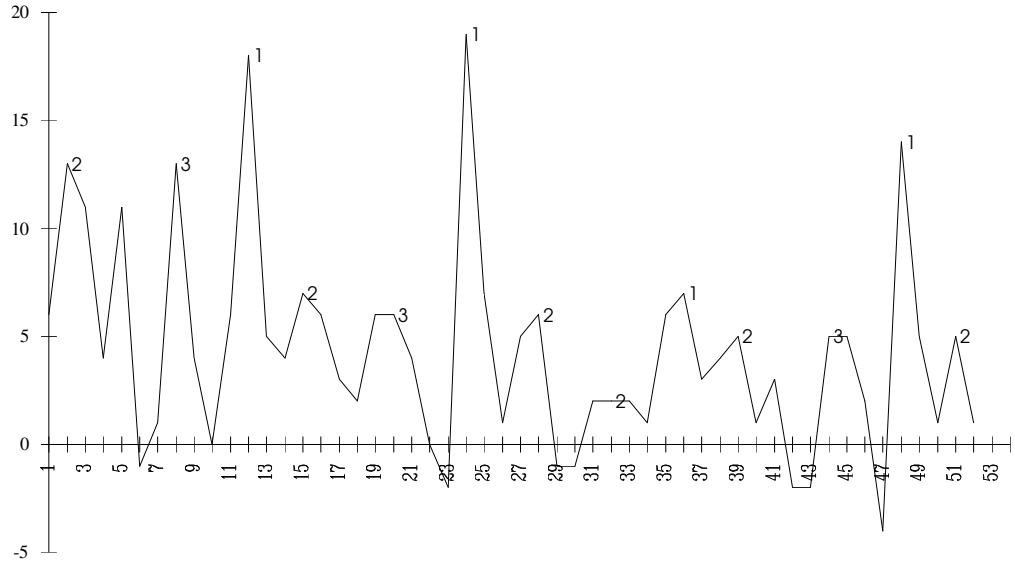

Chart 1.2. The differential of the index. As the differential is taken as a delta between successive months, there is one value less than in the original curve. Eg. the first point in the picture is the change between January and February in 1989.

As one can easily see, there are two visible cycles. First, there is a high peak (1) at every December. Second, there is a four-month cycle (1-2-3), where the December peak (1) is the highest, March-April peak (2) the second highest and July-August peak (3) is the lowest. Occasionally there is an additional peak in May. The network should be able to find these features.

2. Test runs

2.1. Testing methods

We made 12 different test runs with our own simulator and six with SNNS v3.0[1].

Here are the tests made with our simulator. The *linear/sigmoid output layer* refers to the transfer function at the output layer. The transfer functions of all other layers are sigmoid.

1. *Linear output layer, 12x10x1*[2]*, 2000 iterations*. This is the basic test with big input layer and rather big hidden layer.
2. *Linear output layer, 12x10x1, 400 iterations*. This test is made to study the effect of overtraining. In the basic test the network may learn too efficiently and here we try to see how less training affects the results.
3. *Sigmoid output layer, 12x10x1, 2000 iterations*. This tests the effect of preprocessing values of series with sigmoidal function that compresses the curve to the range [0,1].
4. *Linear output layer, 8x5x1, 2000 iterations*. Same as the basic test, but the inputs are from the 7 previous months.
5. *Linear output layer, 8x5x5x1, 2000 iterations*. Same as previous, but with an additional hidden layer.
6. *Linear output layer, 5x5x1, 2000 iterations*. This tests what happens if the input layer is very small and the network therefore has information only from very near in past.

These six tests are run with two different training sets, resulting in total 12 tests. This means that the training set in the first test is $\{a_1..a_{36}\}$ (three years), and in the second test is $\{a_1..a_{48}\}$ (four years). We didn't use the entire series as the training set becouse we wanted to compare the results of the network to the actual data.

Interpreting the graphs made with our simulator

The thin line is the original data. It ranges from January 1989 to May 1993. The thick line is the prediction made by the network. It starts from the end of the training set and then continues far to the future to show how well the network has learned the main features in the series. How well the network has succeeded in predicting the curve, can be seen where the curves overlap. The first predicted value (the second point in the thick curve) is the most important one.

This prediction is done so that first the last values in the training set are presented to the network. After processing it, we shift all the values in the input layer to left by one step and then move the output value of the network to the rightmost input node. Then we just iterate this as long as we want to.

---

[1]Below the SNNS is used as an abbreviation for Stuttgart Neural Network Simulator, Version 3.0. SNNS is Neural Network Simulator published by Institute for Parallel and Distributed High Performance Systems (IPVR), University of Stuttgart.

[2]12x10x1 means a network with 12 input neurons, 10 hidden neurons and with one output neuron.

It should be noted that the first predicted value (where the thick line begins) actually belongs to the training set[3]. The effect of this can be seen in simulations where the error is large, by making the predicted curve more separate from the actual curve.

Interpreting the graphs made with SNNS

The thin line is the original data, as above, ranging from January 1989 to May 1993. ***The thick line is the RECALL made by the network.*** This means that the training set ranges from (for example) 1..36, and test set ranges from 1..52 (the entire series). So it does not predict the curve iteratively. The output of the network is not used as an input for further estimates as in our own simulator. **NOTICE THIS DIFFERENCE BETWEEN THE GRAPHS!**

2.2. Test runs with our simulator

The alpha was set to 0,2 and eta to 0,1. These were not modified in the tests, since we are studying the feature detection abilities, not their learning speeds.

2.2.1. Linear output layer, 12x10x1, 2000 iterations

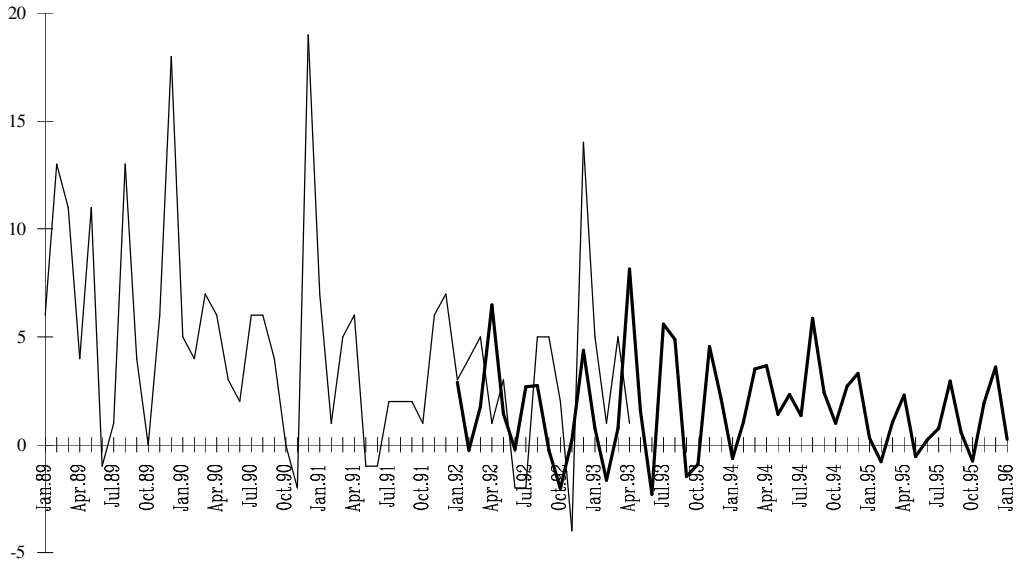

Chart 2.1.a. *Training set range was the first 36 values (Jan 89 to Dec 91). Error was 6E-13.*

The network has learned the 4-month cycle, but very little of the yearly cycle and the first predicted value is totally wrong. The output seems rather weak. It may be that the this training set (1..36) was chosen a bit badly, becouse the curve may be a bit odd at the end of the set, when compared to other similar points in the curve.

---

[3]The reason for this is that the first input pattern is taken from the training set as a normal training pattern and the TrainSet class doesn't permit indexes out of range. This problem could have been eliminated, but as the errors were rather small, we didn't consider it absolutely necessary.

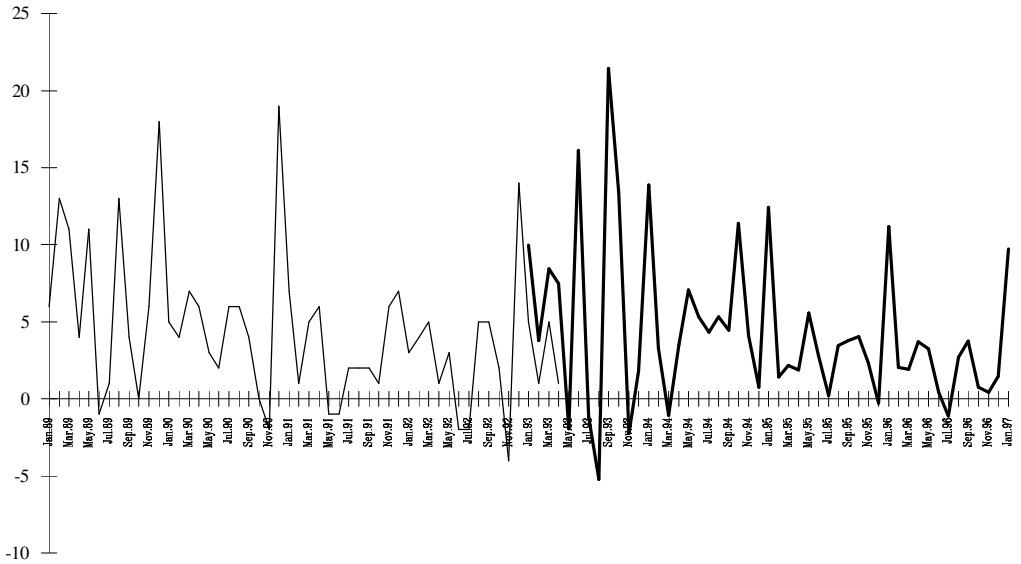

Chart 2.1.b. *Training set range was the first 48 values (Jan 89 to Dec 92).*

Here the network has succeeded in predicting the curve rather well. However, the yearly cycle is still not very clear, but at least it is there.

2.2.2. Linear output layer, 12x10x1, 400 iterations

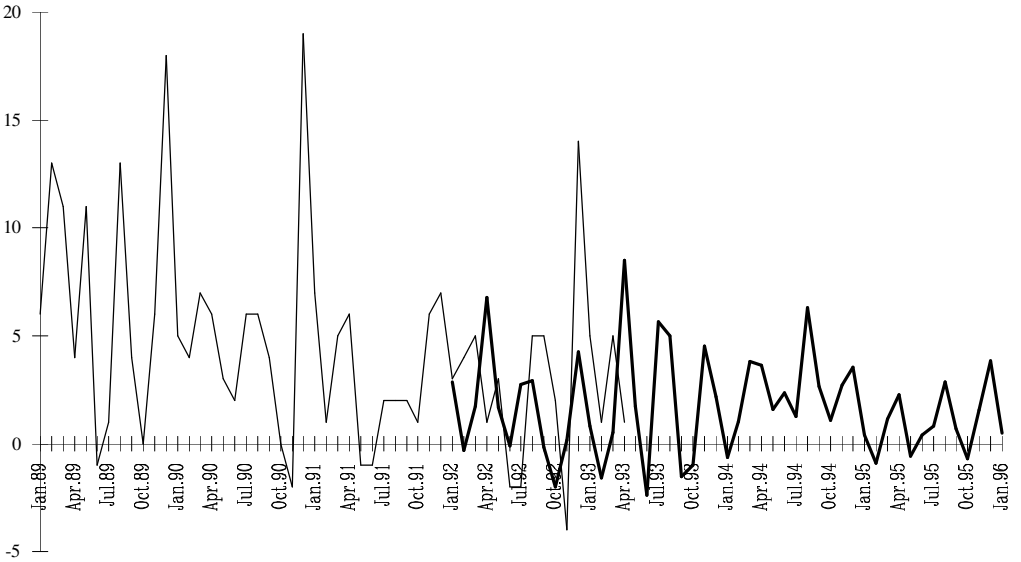

Chart 2.2.a. *Training set range was the first 36 values (Jan 89 to Dec 91). Error was 0,0056.*

This seems to be almost exactly the same as chart *2.1.a*, which might imply that excessive learning time does not affect results notably.

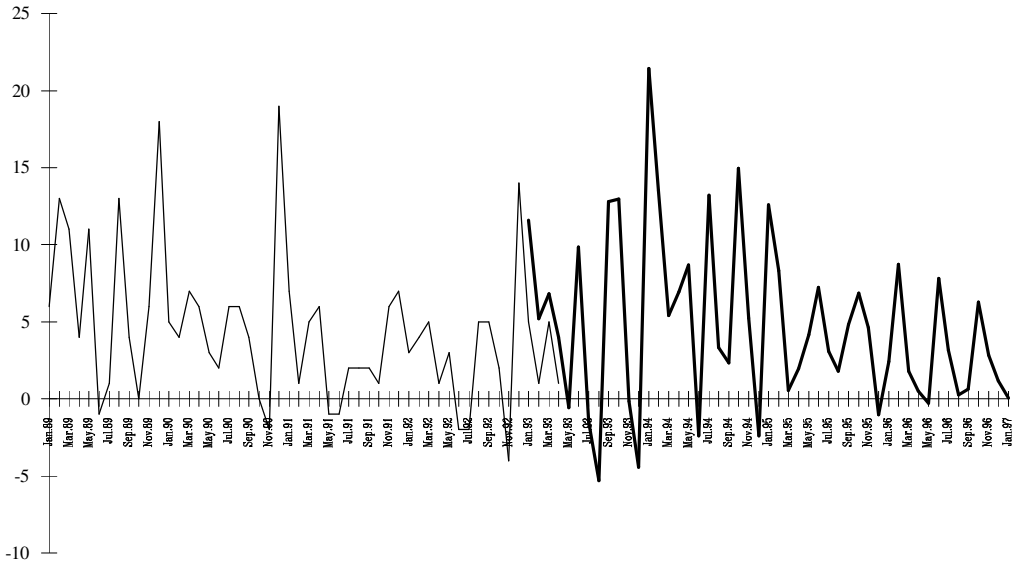

Chart 2.2.b. *Training set range was the first 48 values (Jan 89 to Dec 92).*

This should be compared to the chart *2.1.b.* As it can be seen, there is definite difference with these two charts. This curve predicts the first few months better that above, but then loses the form. Thus the result would seem to be a little worse than before.

2.2.3. Sigmoid output layer, 12x10x1, 2000 iterations

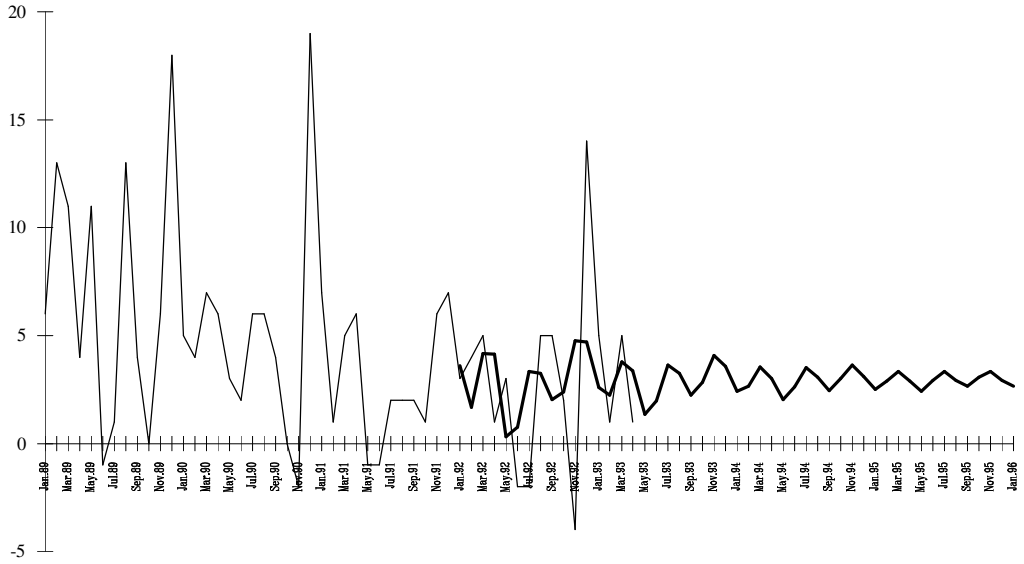

Chart 2.3.a. *Training set range was the first 36 values (Jan 89 to Dec 91). The error was 0,15.*

This is the best prediction with training set of 36 months, but although the network has learned the 4-month period, it doesn't know almost anything about the yearly cycle. It may be that the

good results in the beginning were just good luck, becouse the curve there doesn't differ much from the curve later.

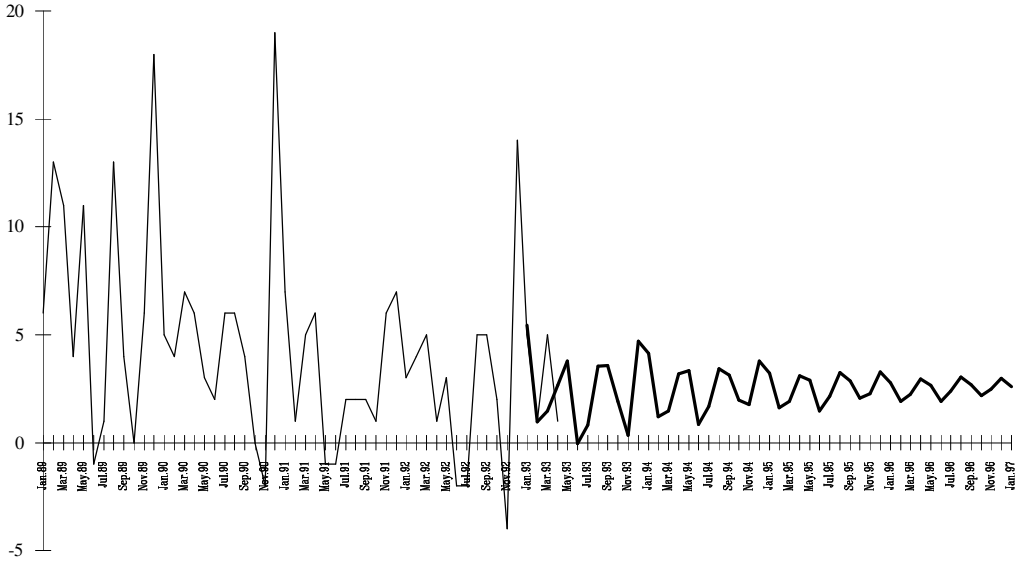

Chart 2.3.b. *Training set range was the first 48 values (Jan 89 to Dec 92).*

Here the network has found the 4-month cycle, and been able to predict the first month quite accurately (perfectly), but it loses the yearly cycle very fast. It would seem that although sigmoid scaling might be the most suitable for backpropagation networks generally, it is not that with this kind of data and processing method (referring to the feedback iteration).

2.2.4. Linear output layer, 8x5x1, 2000 iterations

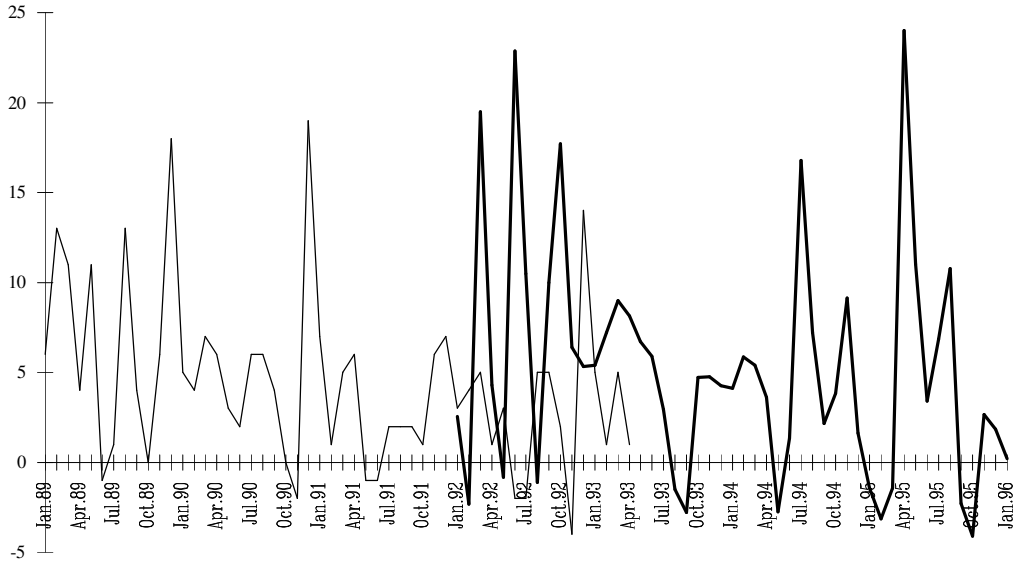

Chart 2.4.a. *The training set range was the first 36 values (Jan 89 to Dec 91). The error was 0,0007.*

As before, this 36-month set has not succeeded in predicting anything. Output values seem to be a bit too high.

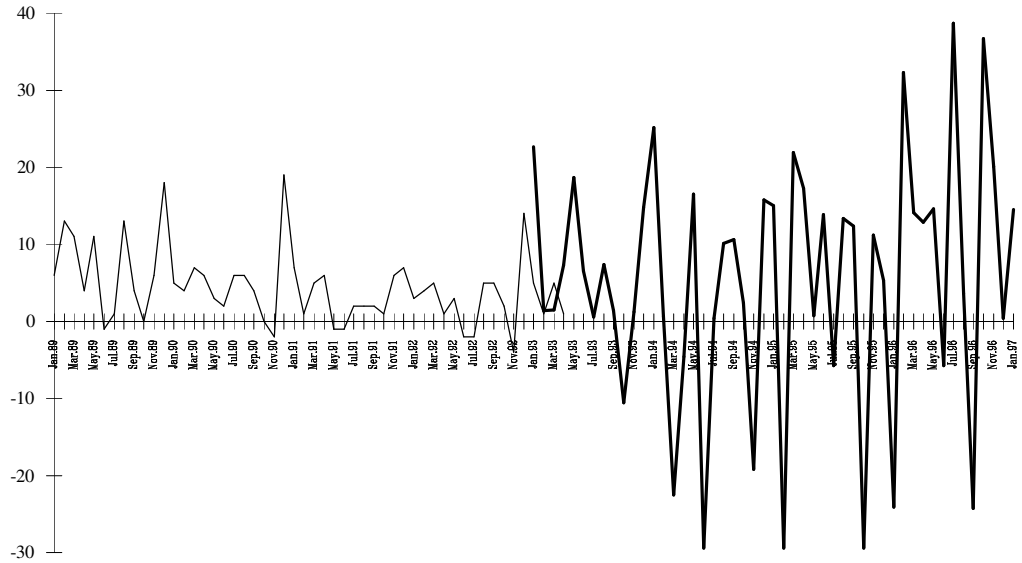

Chart 2.4.b. *The training set range was the first 48 values (Jan 89 to Dec 92). The error was 0,15.*

Here the training has not been very successfull. The network seems to have been able to predict a little, but it does not look very assuring. Now have the output values jumped to totally bizarre values.

2.2.5. Linear output layer, 8x5x5x1, 2000 iterations

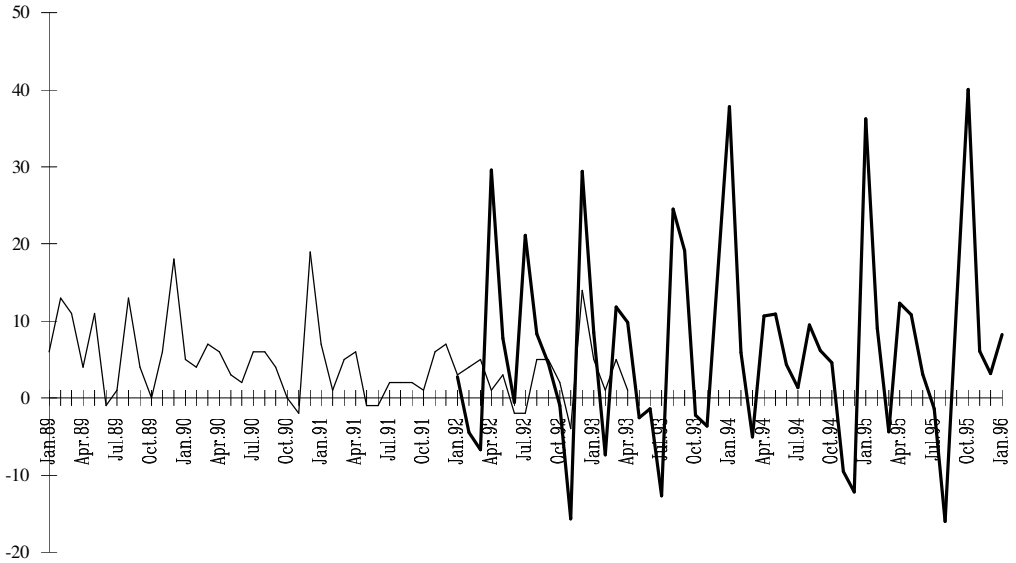

Chart 2.5.a. *The training set range was the first 36 values (Jan 89 to Dec 91) and the error was 0,03.*

Still no prediction and the values are still much too high. This should be compared to chart 2.4a.

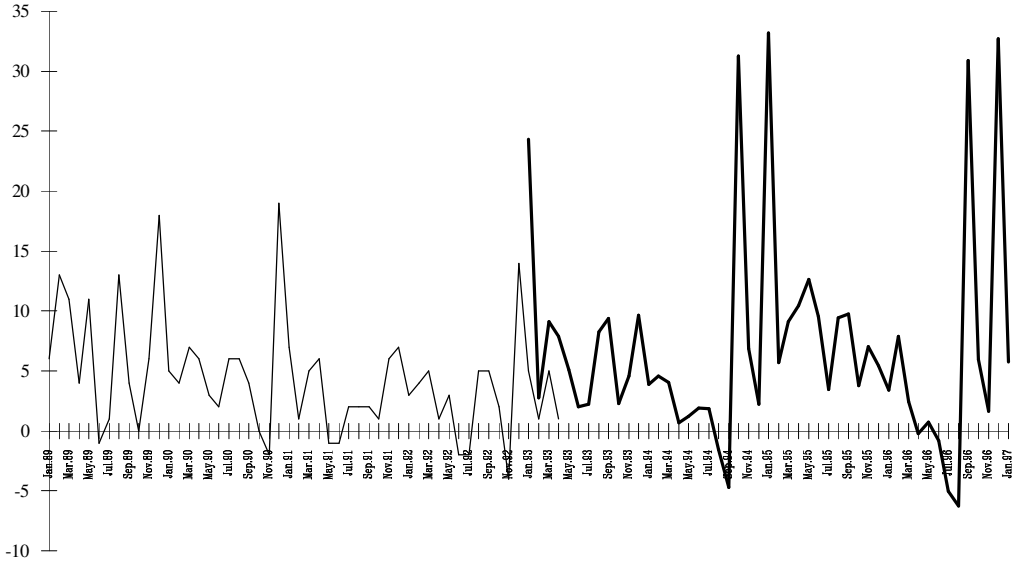

Chart 2.5.b. *The training set range was the first 48 values (Jan 89 to Dec 92). The error was 0,06.*

Here the network has been able to predict something, but the error of the first point is quite large and the curve seems to be a bit unstable; the yearly cycle has become something strange.

2.2.6. Linear output layer, 5x5x1, 2000 iterations

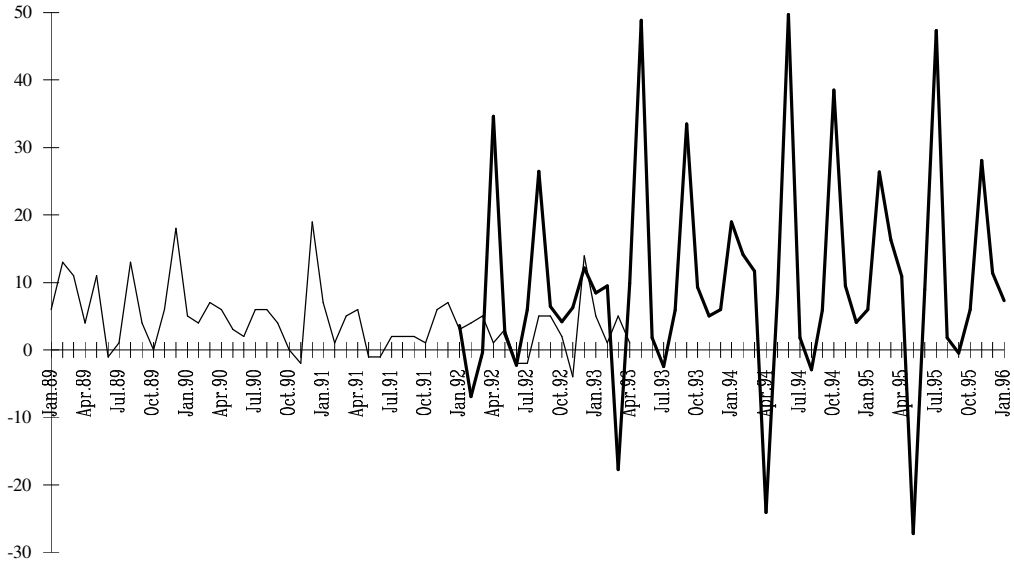

Chart 2.6.a. *The training set range was the first 36 values (Jan 89 to Dec 91). The error was 0,23.*

No prediction. The yearly curve seems to be strangely stable and even.

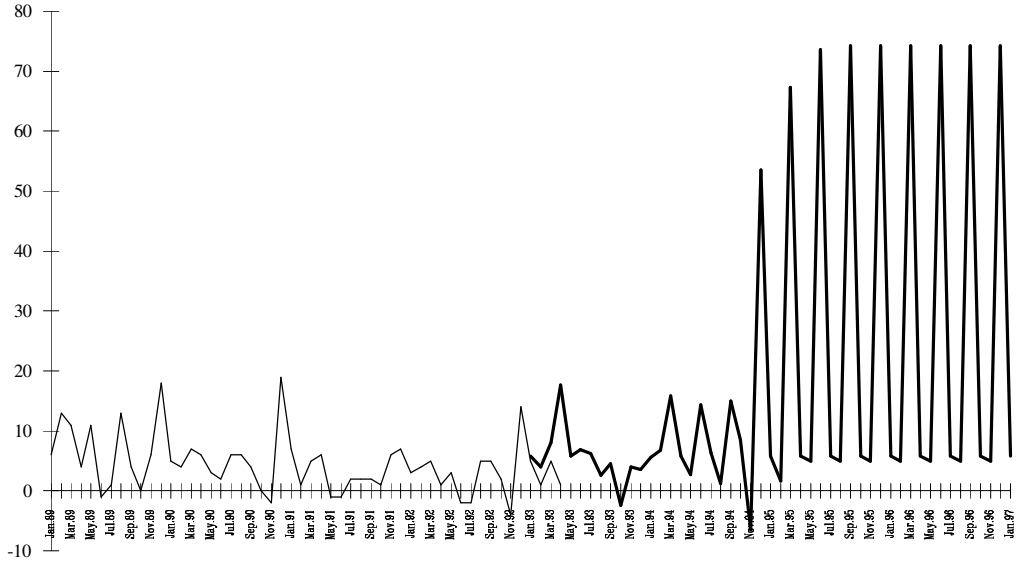

Chart 2.6.b. *The training set range was the first 48 values (Jan 89 to Dec 92) and the error 1,8.*

Here the error is very high, but amazingly doesn't affect the first point much. The network has predicted the first months somewhat correctly, but then becomes unstable before completely freaking out. It seems that this small network is not adequate for learning patterns of this complexicity, as with simpler data (smaller training set) this worked much better.

2.3. Test runs with SNNS

SNNS is capable of simulating several neural network models. Here we have used it to simulate traditional back propagation in feed forward network, without feedbacks or time delays. Every layer in the net is connected to its adjacent layers through weight matrix. In addition all neurons have a threshold. As we could not get linear output functions to work properly with this version of SNNS, we have used only sigmoid output function. Thus some results are not directly comparable to the results we got with our own back propagation simulator.

In our first experiment with SNNS we scaled the input data with sigmoid function. Result of output neuron was scaled back to original range with inverse of sigmoid (that is, *-ln(1/x-1)*). In chart 2.7 we have the differential curve of the index after teaching to the net the indexes from January 1989 to December 1992. The net has produced the last five months without knowing the real index values. In chart 2.8 there is the same setting, except that the teaching range is smaller. In this experiment we taught indexes only from January 1989 to December 1991. The net has produced the last 17 months by itself. It can be seen that the net tries to underestimate high peaks.

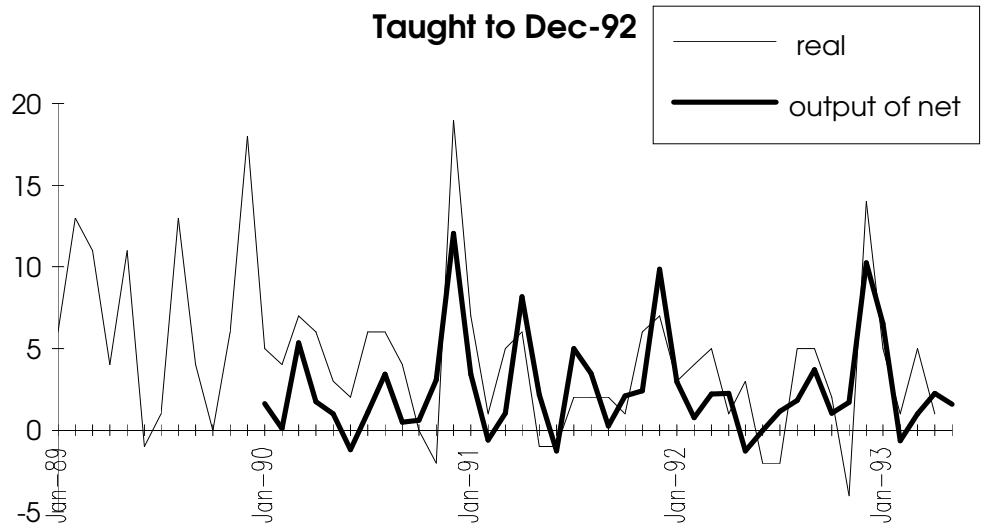

Chart 2.7 *The output of the network with 12 input units, 10 hidden units and one output unit. The network studied 36 patterns (Jan-89 to Dec-92), each 2000 times.*

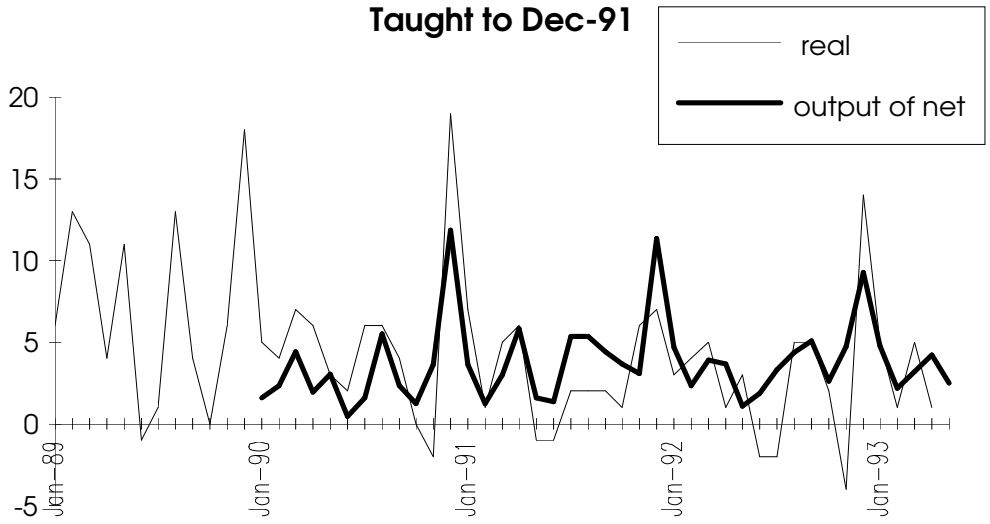

Chart 2.8 *The output of the network with 12 input units, 10 hidden units and one output unit. The network studied 24 patterns (Jan-89 to Dec-91), each 2000 times.*

In chart 2.9 there is an index curve produced by a net with two hidden layers. Also the input range has been changed from one year to eight months. Input and output values are scaled with a sigmoid and it's inverse function. The net was taught four years of index changes. The result is clearly weaker than in the experiment with one hidden layer and larger input layer (chart 2.8). The curve has flattened and some peaks have moved one month either to right or left.

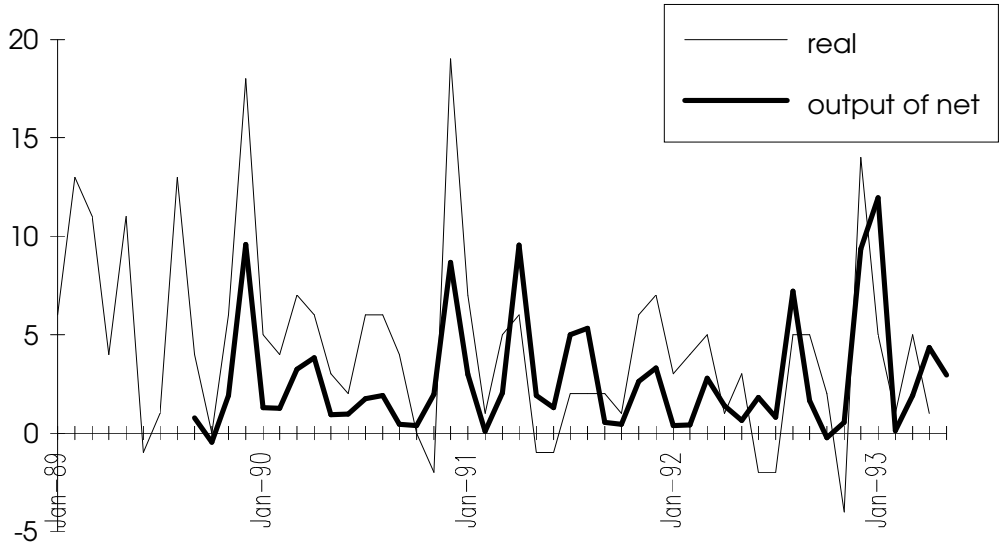

Chart 2.9 *The output of a network with 8 input units, 5 units in the first hidden layer, 5 units in the second hidden layer and one output unit. Network studied 40 patterns (Jan-89 to Dec-92), each 10,000 times.*

In the next experiment we tried to demonstrate the influence of the number of iterations. In chart 2.10 there is the result curve of a net that has studied all input patterns for 20,000 times and another curve in which the net has studied the patterns for 2,000 times. Increasing the number of

iterations made the result curve too sensitive to minor disturbances. Input values are here scaled linearly. We gave to the highest peak a value little less than 1 and to the deepest bottom a value little over 0. The output was finally scaled to the original range. This linear scaling caused some overestimates which did not occur in other experiments.

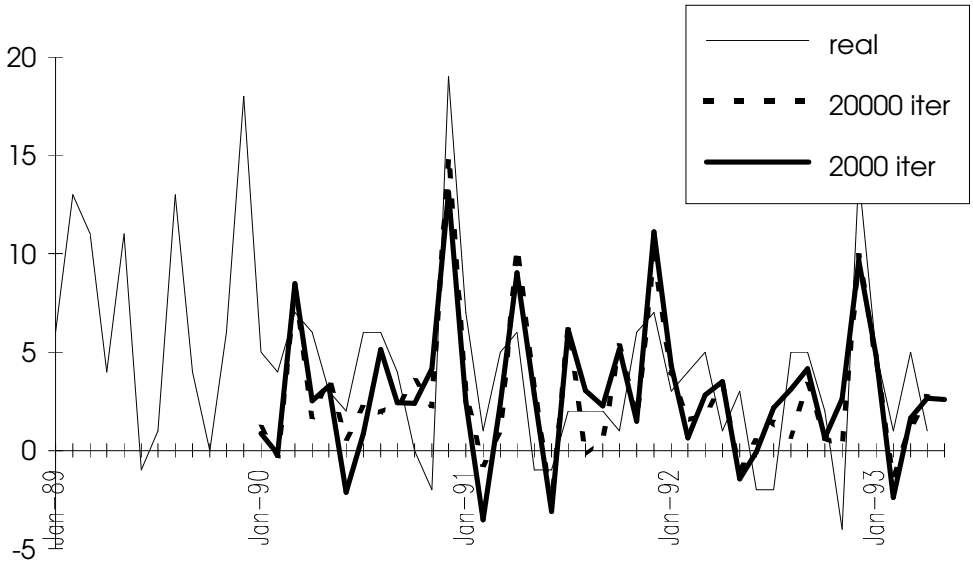

Chart 2.10 *The output of the network with 12 input units, 10 hidden units and one output unit. Network studied 36 patterns (Jan-89 to Dec-92), first for 2,000 times and later for 20,000 times total.*

3. Summary and conclusions

The tests showed that it is possible to extrapolate periodic curves easily. The best network for the job was the biggest. The reason for this is propably that the networks with small number of input units cannot have the entire cycle as input at one time. So as with this data, the network would have to, for example, predict the December peak from values of the last few months, while a network with more inputs could predict it by seeing that last December peak is now 12 months away. The input layer size also affected the exact length of the yearly cycle, for the same reason.

Smaller networks produced a little more simple curves than bigger ones. Also, only three years of training data was apparently too little to learn the yearly cycle. All networks learned the four-month cycle.

Linear output layer was better than sigmoidal. Maybe not so much for learning, but for making the curve look more normal.

We were disappointed in the limitations and bugs of SNNS. The simulator was inappropriate for this kind of future prediction. However, our own BP-simulator surprised us with good prediction capabilities. We did change our program in order get it more suitable for this specific assignment. With SNNS we did not have the same possibility. Thus the results of these two simulators are hardly comparable. SNNS did not predict future; it rather worked as an ordinary neural network. It produced output value for given input vector. Our own BP-simulator used output value as a part of new input vector, which was given to the network in order get estimation for next month. To get the final approval for the results, we just need to wait until new index tables are published and then compare those to our results here.