

Solar variation signals in the Earth's atmosphere
- a study using neural networks.

Johan Arvelius

Swedish Institute of Space Physics
P O Box 812
981 28 Kiruna
Sweden

Master Thesis 20p
Thesis advisor:
Sheila Kirkwood

June 20, 1998

Abstract

The effects of solar activity on the Earth's atmosphere and our weather systems is a very complex but interesting task for research. The purpose with this work is to see whether one can see which aspects of solar activity are best correlated with the local weather in Kiruna using neural networks. This has its interest in choosing future research directions in the area.

After a discussion of which measurable parameters of the solar activity might be relevant, a study is made to see if one can see any correlation between the solar activity and the local weather using time dependent neural networks. The result is, unfortunately, negative.

The rest of the study is spent in study the correlation between solar activity and the ionosphere, where the connection is simpler, deterministic and well understood. By studying this system one can learn much about the influences of solar activity on the Earth's upper atmosphere. Because this is a well understood system I could also learn much about the performance of the neural networks in resolving different changes.

Contents

1	Purpose	3
2	Description of work	3
3	Theory	3
3.1	Solar activity	3
3.1.1	$F_{10.7}$	3
3.1.2	Sunspots	3
3.2	Ionosphere	4
3.2.1	Influences from the sun	5
3.2.2	EM characteristics	5
3.2.3	Working principles of the ionosonde	6
3.2.4	Geomagnetic indices	6
3.3	Troposphere	8
4	Neural networks	8
4.1	Levenberg-Marquardt algorithm	10
4.2	Gradient descent algorithm	11
4.3	Number of neurons	12
4.4	Normalizing data	12
4.5	Validation of the training time	12
5	Training trials	12
5.1	Ionosonde data	12
5.1.1	Used data	13
5.1.2	Number of neurons for best performance	14
5.1.3	Different input parameters	14
5.1.4	Time delay networks	15
5.2	Weather data	15
5.2.1	Used data	16
5.3	Error analysis	16
6	Results	17
6.1	Number of neurons for best performance	17
6.2	Different input parameters	18
6.3	Time delay networks	18
6.4	Weather data	22
7	Conclusions	22
8	Acknowledgement	22
A	Matlab scripts	25
A.1	Comments to scripts	25
A.2	trainnet.m	25
A.3	simunet.m	28
A.4	normal.m	30
A.5	denormal.m	30

1 Purpose

The purpose of this work was to see if the method with neural networks can tell anything about which aspects of solar activity are most closely correlated to the weather and climate change in the Kiruna area. This has it's interest in choosing future research directions to study possible causes of solar-variability influence on climate.

2 Description of work

There are three main steps to be considered in correlating the weather on the Earth with what's happening in the sun. First one has to measure the activity of the sun in some way that is quantifiable and that hopefully measures something that can have an influence on the Earth. Second, one has to study in which way the Earth system is affected by these parameters. Since the direct effect in most cases is in the upper atmosphere the third step is to determine in what way these influence can go further to the ground and affect the troposphere.

Since the purpose of this work is rather to see which parameters affect the direct weather conditions rather than how they do it, I started to look at only the first and last step, i.e to look for correlations between solar activity and weather directly using the neural networks.

The upper atmosphere plasma densities measured with an ionosonde should be used as a control of the behavior of the network. These parameters correlation with solar activity is deterministic and well understood.

3 Theory

This theory section discuss the different interactions devided in their geometrical placement rather than in their nature as discussed earlier. This means that both the direct influenses from the sun and the influenses from the other atmospheric layers on the troposphere are discussed in the same section.

3.1 Solar activity

3.1.1 $F_{10.7}$

There is a spectral component superposed on the thermal background solar radio wave emission that varies with phenomena like sunspots and plagues. This emission is largest in the cm and dm regions, is well correlated with EUV and X-ray radiation flux and appears to have its origin in areas on the solar surface with very high temperature [3].

The normal wavelength to measure this radiation is 10.7 cm and the flux at that wavelength is called $F_{10.7}$. There is a direct correlation between the $F_{10.7}$ and the effective temperature in the emission regions. [3]

$$F_{10.7} = 16.4 \cdot T \text{ W m}^{-2} \text{ Hz}^{-1} \text{ K}^{-1}. \tag{1}$$

Measurements of $F_{10.7}$ are made by the Institute for Telecommunication Sciences and Aeronomy, Boulder, Colorado. They are from an observatory in Ottawa.

3.1.2 Sunspots

The strongest sunspots have been observed by the naked eye for thousands of years and Galileo was the first person to look at them through a telescope in 1616. [8] Sunspot counts are the only

directly observed parameter available for longer series i.e. to look for solar activity affecting the climate changes on Earth on the timescale of several hundred years.

Counting sunspots is like counting islands, two persons never get the same result. In this case the telescope also is a limiting factor. Wolf devised a quantitative definition for a sunspot number to make it more reliable. In its present form it is defined

$$R = k(10g + f), \quad (2)$$

where R is the Wolf sunspot number, g is the number of observed groups of sunspots, f the number of sunspots and k is a calibrating factor for the observatory. [3]

In 1992 Hoyt and Schatten [14] showed from statistics from 100 years of observations that more than 90 % of the variability is attributable to changes in the number of sunspot groups. They also showed that there are a linear relationship between the Wolf sunspot number and the number of reported sunspot groups,

$$R = (12.36 \pm 0.217)g - (3.23 \pm 7.33), \quad (3)$$

that is valid up to 12 sunspot groups. The number of sunspot groups counted by different observatories has a far less noise level and there are no bias effects to all telescope observatories as all sees all sunspot groups. Counting individual sunspots are more subjective for the observer and are more depending on the telescope equipment. [12]

From this they decided to make a new index based on only the sunspot group numbers. In their work [6] they tried to calibrate the reported numbers from different observatories both for new data after Wolf and a recalibration on the older data. This calibration was carried out in the way that for every observer all their data from days they had observed in common with Royal Greenwich Observatory (RGO) were summed up. The ratio of the sums from the two observatories was taken as a linear correction factor. For data older than those of RGO other methods have been used but those are not used here. The group sunspot number was normalized to the Wolf numbers for the period 1874 to 1991 and that resulted in the definition of the Group Sunspot number

$$R_g = 11.93 G_{\text{RGO}}, \quad (4)$$

where G_{RGO} is “the number of sunspot groups which would be counted if RGO were observing” according to [6].

The physical connection that the sunspots are correlated to some measurements on Earth is thought to be due to solar flares appearing in or close to the spots, i.e. it is likely to be the solar flares that have the strongest influences on the Earth’s atmosphere. [7]

The appearance of solar flares is very closely bound to the Wolf sunspot number. If one divides the solar flares into classes according to their strength there is a direct linear connection between the number of eruptions in each class and the Wolf sunspot number [7].

3.2 Ionosphere

The ionosphere is the outermost section of the Earth’s atmosphere in the height region 70 to 1000 km The gas in this region is partly ionized. The daytime ionization is mainly arising from photo ionization from UV- and X-ray radiation from the sun. The photoionization is in balance with different recombination processes and transportation of the plasma to other height regions. The ionization maximum is reached in about 120 km height. Since the incoming solar radiation in a region on Earth varies with both time in the day and the time of year, the ionosphere will show daily and yearly variations. The solar outgoing radiation in the UV- and X-ray regions

also varies (e.g. in 27 days and 11 years cycles) so the ionosphere will be influenced from this also. [2]

The ionosphere is for natural reasons divided into several different layers according to the composition. The first observed layer was the E-layer between 90 and 130 km. The other layers are denoted D and F from alphabetical order. The F-region above 130 km is subdivided into F1 and F2.

3.2.1 Influences from the sun

The balance of ionization is the determining factor for the physical properties of the ionosphere. The processes can be divided into two major parts, the photochemical which describe the production and destruction of ions and the transport which describe the movement of the ionized gas. [11] Especially at high altitudes and during magnetic storms there is some ionization due to energetic particle precipitation rays so not all of the variations in the ionization can be explained from UV and X-ray radiance from the sun.

Chapman made a study in 1931 where he made theoretical calculations on the ionization in the ionosphere under the following simplifications:

- The radiation is monochromatic, its photon flux being $I(h)$.
- The atmosphere consists of a single absorbing gas, its concentration being $n(h)$.
- The atmosphere is plane and horizontally stratified.
- The scale height H is either independent of height or varies linearly with height.

This model leads, after a lot of calculations to the result that the total ion production rate is proportional to the incident energy flux. [11]

The F2 layer is dominated by O^+ ions and the main recombination process in this region is that the O^+ reacts with a neutral molecule in a way that the net effect is that the ion and the molecule together with a free electron forms three neutral atoms. This layer is the most difficult to predict of the ionospheric layers, it has also large vertical movements driven by neutral winds or magnetospheric electric fields [8]. It is in the transition region between the photochemical and transport dominated regimes in the atmosphere.

The F1 and E layers are thought to be quite well understood from Chapmans theoretical model. The ionization in those layers also varies very rapidly and directly with the solar variation.

I jump over the F1 layer and goes to the E-layer from which I have used measurements from in this work.

The E-layer is seen in the daytime free electron density. This layer is dominated by O_2^+ and NO^+ ions, that have been produced by ultraviolet radiation in the 100–150 nm range and X-rays in the 1–10 nm range. [8] These ions recombine readily with free electrons. They have a short lifetime and their concentration so not much affected by transport effects.

Many rocket experiments have showed that the amount of O^+ and N_2^+ ions decreases very rapidly below 150 km and can be neglected in the E-region. [3]

3.2.2 EM characteristics

From electromagnetic theory we can see some of the characteristics of the ionosphere.

Let B denote the flux of the Earth's magnetic field and f a radio frequency entering the ionosphere. We define the plasma and gyro frequencies as follows,

$$(2\pi f_N)^2 = \omega_N^2 = Ne^2/m\epsilon_0, \quad (5)$$

$$2\pi f_H = \omega_H = Be/m \quad (6)$$

and two parameters

$$X = f_N^2/f^2, \quad (7)$$

$$Y = f_H/f. \quad (8)$$

The gyrofrequency is the frequency of the circular motion a charged particle is making in a magnetic field around the fieldline according to the Lorentz force.

In a magnetic field the ionosphere is doubly refracting and the two modes of propagating radiowaves are called “ordinary” and “extraordinary” in the same way as in crystal optics and will further be denoted by an index o or x .

A wave vertically incident on the ionosphere is reflected at a level where $\mu = 0$. For the ordinary wave this happens when $X = 1$ and for the extraordinary when $X = 1 - Y$ if $f > f_H$ and $X = 1 + Y$ if $f < f_H$. [11]

Along the magnetic field the refractive index for the two modes is given by

$$\mu^2 = 1 - \frac{X}{1 \pm Y}. \quad (9)$$

The two modes are (nearly) circularly polarized in opposite directions and if the incoming wave is plane polarized the plane of polarization rotates according to the different refractive indices. The effect is called “Faraday rotation” as in optics. [11]

3.2.3 Working principles of the ionosonde

The ionosonde is a radio transceiver that sends out a plane polarized wave transversely into the sky. The signal will be reflected in the ionosphere and the reflected signal is received by the transceiver at the ionosonde. The time of flight and the orientation of the polarizing plane tells how far away the wave was reflected and what media it has traveled through on the way.

The height calculated by multiplying half the traveltime with the speed of light is called the virtual height h' . By scanning the ionosonde through a frequency interval and plotting this virtual height as a function of frequency one gets a so called ionogram (example in fig 1).

As seen above the ordinary wave is reflected at $f = f_N$, and from the eq 5 we can see that $f_N \propto \sqrt{[e^-]}$. So the ionogram is a plot of the electron density at different virtual heights in the ionosphere. The ordinary and extraordinary waves are seen as one trace each in the ionogram.

From the ionogram in fig 1 one can easily see several critical frequencies, these frequencies represents the electron densities in the mayor layers in the ionosphere. [11] These critical frequencies are called after which branch they belong to and which layer they are reflected against e.g. f_oF2 or f_xE .

3.2.4 Geomagnetic indices

Not only the electron density in the ionosphere is influenced by the solar activity but also the electrical currents in the plasma. Electromagnetic forces in the ionosphere can lead to large electric currents. The drift of ions in the ionosphere can cause ring currents along closed fieldlines. Especially in the van Allen radiation belts at 4–6 Earth radii the currents are strong

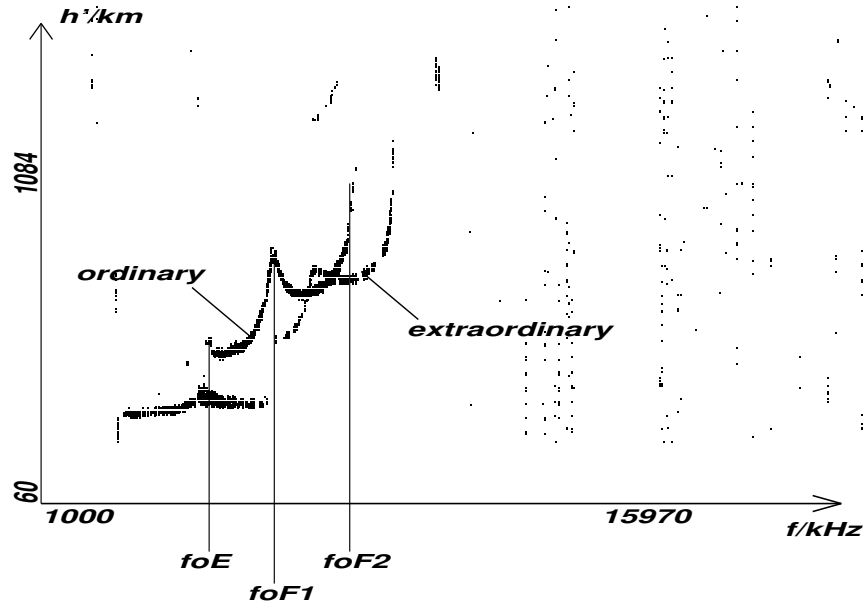


Figure 1: Ionogram from the IRF ionosonde in Uppsala 980518.

because of gradients in the main field. The growth and decay of these currents are referred to as magnetic storms. There are also smaller disturbances when, according to McPherron [10], “a significant amount of energy” is obtained from solar wind-magnetosphere interaction. This is called a geomagnetic substorm. [9]

There are also variations independent of the solar wind that are due to tidal effects from the Sun and the Moon and a daily change according to the Earth spinning around its axis. Changes in sunspot activity also seems to give considerable variations, both from sudden occurrence of solar flares and from the solar cycle. [9]

The first measurements of the Earth’s magnetic field were made in the early 16th century. Systematic measurements were started in the 19th century. [7]

The variations of the magnetic fields seen with a magnetometer are quite complex and not easy to characterize. Starting in 1884 there was an attempt to characterize them in a subjective way — the observer should assign how much activity there was. A more objective treatment was of course needed so J. Bartels et al introduced an index called the K index in 1939. [9]

The K index is derived in the way that one takes the measurements from three-hour intervals in the day, takes the difference between the highest and the lowest measured values in each of the two horizontal components that are measured. This value is then mapped on a scale from 0 to 9 with a normalization function that is different for every station but in principle logarithmic. The value on this ten grade scale is called the K index. As the scale is quasi logarithmic, it is not good to make averages from. To do something about this it is mapped back on a linear scale. This new value is called an a_k index and those are thereafter averaged from 13 specially chosen stations in the world to make up the global a_p index. The eight a_p indices for one day are averaged to a mean day index for the whole planet A_p . [7]

3.3 Troposphere

In the troposphere the direct radiation from the sun is mostly in the wavelengths close to the visible window, plus a much smaller contribution from the radio-wave region. The solar output in the visual region is not effected in the same way as the radio-, UV- and X-ray radiation due to differences in solar activity. The troposphere is in that way not so directly effected by the solar activity as for example the ionosphere.

The big question is whether most of the variation in the weather conditions on the Earth which are correlated to solar activities are due to interactions higher up in the atmosphere that are tranferred down to the troposphere or if there is a more direct influence.

It was hoped to be able to test this by comparing correlations of weather data with direct solar parameters (eg sunspots) and correlations of weather data with parameters which quantify disturbances in the upper atmosphere (eg magnetic indices).

4 Neural networks

Two different network architectures have been used in this work. Timeseries should be best investigated with recurrent networks. Elman is one architecture I've tried that is designed for timeseries. The recurrent networks were however a little bit tricky to get total control of.

Another way to get the network time dependent is to give it several values from the series at the same time, so called time delay networks.

Elman network The elman network is a partly recurrent network. There is a feedback connection in the hidden layer. The output from that layer is taken as an extra input to the corresponding neurons again. The weight of this connection is held constant so it is only the feedforward that is updated. [16] This architecture allows the network to both detect and generate time varying patterns. [4] The hidden layer has a tangent sigmoidal transfer function that normalizes all the values to the range (-1,1) and the output layer has a linear transfer function.

Time delay networks The principle behind a time delay network is to use a network of an ordinary feed forward architecture and present several values from the training series at the same time. The values must be taken in a row and the "window" of the series is changed to higher values with one value in each step. The network gets then a chance to take the whole time period which the window covers.

Training the network The main idea in training the network is to minimize the error function

$$E = \frac{1}{2} \sum_{i=1}^S \sum_{\mu=1}^N (T_{i\mu} - O_{i\mu})^2, \quad (10)$$

where $T_{i\mu}$ and $O_{i\mu}$ is the target and the predicted value for the i th output neuron corresponding to the μ th training input respectively, N is the length of the training series and S is the number of output neurons. [17]

There are some different training strategies to make the network efficient in the training, both for shortening the training time and to make a good result. I have used the two different training algorithms presented below.

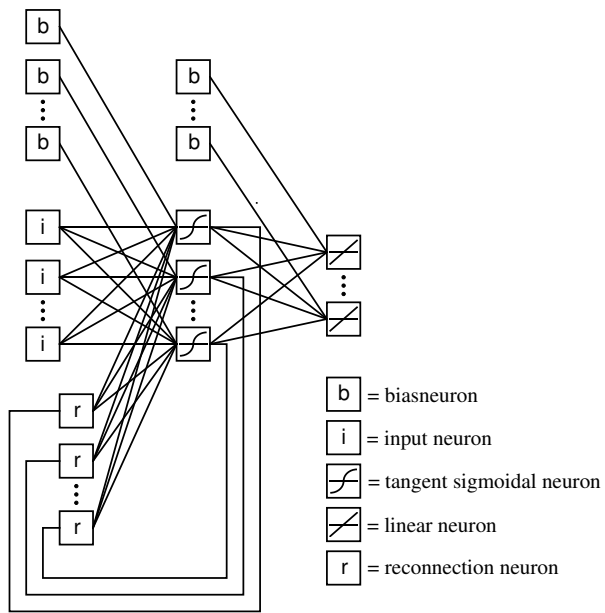


Figure 2: Elman recurrent network architecture.

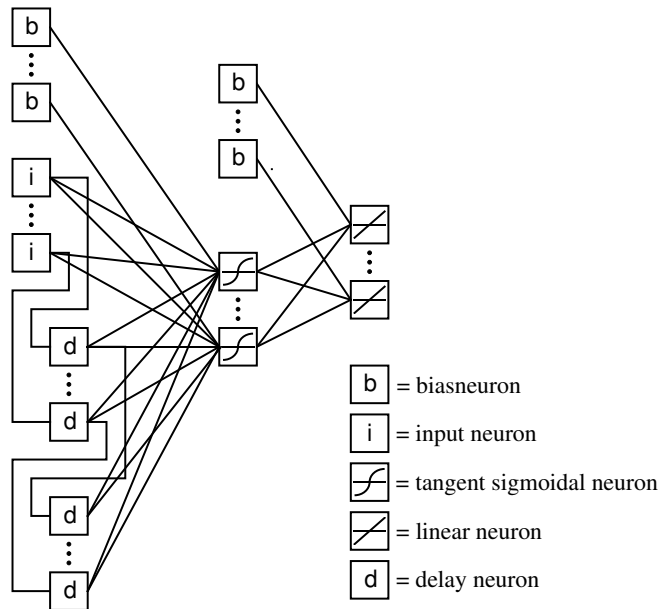


Figure 3: The architecture of the delay network used in this work.

The network works the a way that it tries to predict the values in it's target series. After the predictions are done the network calculates the sum square error of it's prediction. It does a backpropagation to calculate the new weights in the network. This calculation is done to minimize the sum squared error.

4.1 Levenberg-Marquardt algorithm

If we express the errorfunction 10 in the way

$$E = \frac{1}{2} \sum_n (\varepsilon_n)^2 = \frac{1}{2} \|\boldsymbol{\varepsilon}\|^2, \quad (11)$$

where ε_n is the error for the n th input and the elements of $\boldsymbol{\varepsilon}$. If we are at a point \mathbf{w}_{old} and moves to a place \mathbf{w}_{new} we can by defining a matrix \mathbf{Z} of the derivatives

$$\mathbf{Z}_{ni} \equiv \frac{\partial \varepsilon_n}{\partial w_i}, \quad (12)$$

expand the error function in a first order taylor series

$$E = \frac{1}{2} \|\boldsymbol{\varepsilon}(\mathbf{w}_{\text{old}}) + \mathbf{Z}(\mathbf{w}_{\text{new}} - \mathbf{w}_{\text{old}})\|^2. \quad (13)$$

To minimize this function with respect to the new weights \mathbf{w}_{new} calculation of the Hessian matrix \mathbf{H} is a main step. The idea with the LM algorithm is that this matrix that is quite tricky to calculate can be approximated to first order by $\mathbf{Z}^T \mathbf{Z}$. It is not obvious that this is a good approximation. Levenberg and Marquardt made an algorithm that ensured that the step sizes were small enough to make the first order expansion good enough by modifying the error function to

$$\tilde{E} = \frac{1}{2} \|\boldsymbol{\varepsilon}(\mathbf{w}_{\text{old}}) + \mathbf{Z}(\mathbf{w}_{\text{new}} - \mathbf{w}_{\text{old}})\|^2 + \lambda \|\mathbf{w}_{\text{new}} - \mathbf{w}_{\text{old}}\|^2. \quad (14)$$

Minimizing \tilde{E} with respect to \mathbf{w}_{new} gives

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - (\mathbf{Z}^T \mathbf{Z} + \lambda \mathbf{I})^{-1} \mathbf{Z}^T \boldsymbol{\varepsilon}(\mathbf{w}_{\text{old}}), \quad (15)$$

where \mathbf{I} is the unit matrix. The parameter λ is upgraded in the same way as the learning rate η for the gradient descent discussed below in eq 19. This ensures it is getting bigger as long as training succeeds in making the error smaller and should give a stable training.

This algorithm is much faster than the enhanced gradient descent algorithm but there is one problem with it. The Jacobian \mathbf{Z} is for long test series quite big and takes a lot of computer memory. There is a limitation in your computer from how large series you can handle with this algorithm. [1]

To manage this problem one can make another approximation of the Hessian matrix by dividing the Jacobian into submatrixes \mathbf{Z}_n ,

$$\mathbf{H} \approx \mathbf{Z}^T \mathbf{Z} = \sum_n \mathbf{Z}_n^T \mathbf{Z}_n. \quad (16)$$

The cost for the lower memory requirement is that there is a significant computational overhead computing the Jacobian submatrixes and it is not so fast any longer. [5]

4.2 Gradient descent algorithm

Another way to update the weights in order to minimize the error function is by calculating the gradient of the sum square error function in the space of the weights. The network thereafter takes a move in the direction of the gradient. The main thing in training the network is to come to the global minimum in the error function.

There are always a risk that one gets stuck in a local minimum. To avoid this the network needs to take large enough steps to avoid small minima in the function but not so large that the whole training is divergent and never finds the minimum.

The easiest way to train the network is to set a constant to multiply with the gradient to get the step. That constant is called learning rate and normally signed η . The new weight matrix \mathbf{w}_{new} is calculated by adding a change matrix

$$\delta \mathbf{w}(\tau) = -\eta \nabla E|_{\mathbf{w}(\tau)}, \quad (17)$$

where E is the sumsquared error function we want to minimize[1].

This is called the gradient descent. It is not an efficient method though the optimal steplength depends a lot on where in the parameter space the actual value is. To make the training more efficient one can use different parameter optimization methods. Its main advantage in comparison with the Levenberg-Marquardt algorithm is its lower memory requirement.

To make the training more efficient and to make sure that it will converge to the global minimum there are several techniques to change the step length during the training.

Momentum One way to speed up the training is to put in a momentum term. This is done by just adding a term that points in the same direction as the last step.

$$\delta \mathbf{w}(\tau) = -\eta \nabla E|_{\mathbf{w}(\tau)} + \mu \delta \mathbf{w}(\tau - 1) \quad (18)$$

The new parameter μ is called the momentum parameter and must be chosen. Where the slope is smooth the momentum parameter will contribute to the gradient term and increase the speed of the training. In regions where the gradient is oscillating the momentum will have less effect because the momentum gets large when the steps are large. It will always help the stepfunction to avoid minima that are small in comparison to the overall curvature. Still the training rate must be chosen to be quite small to avoid divergent training, and the momentum constant must not be so large that it takes over. To get the speed significantly larger you must be able to change these parameters during the training.

Enhanced gradient descent The main idea with enhanced gradient descent is to make the learning rate larger as long as it does not overshoot the minimum (to get closer to it fast) and, when the minimum is overshoot to quickly decrease the learning rate so it converges. This is done by upgrading the learning rate after every step. If the error function has decreased the learning rate is slightly increased. If on the other hand the error function has increased the step is not made and the learning rate is decreased until the error function increases again.[1] The learning rate parameter is updated by

$$\eta_{\text{new}} = \begin{cases} \rho \eta_{\text{old}} & \text{if } \Delta E < 0 \\ \sigma \eta_{\text{old}} & \text{if } \Delta E > 0. \end{cases} \quad (19)$$

There are many variations on this theme by changing the learning rate linearly and so on but the enhanced gradient descent I've used works this way. [4]

4.3 Number of neurons

The maybe most important parameter in the training is the number of neurons in the hidden layer. The more neurons you have the more freedom you give the network. With more neurons the network has larger freedom to correlate the data, but this also means that it can correlate any data even if they are not really correlated with each other. The network's opportunity to get a good fit of the training set of data is not unique, in the same way that a polynomial function can be fitted to a number of points less than its order in many ways. You must test out so you get neurons enough to solve your problem but not much more than that.

According to Wu [15] a rule of thumb is that the number of neurons should be chosen so the number of weights in the networks is one tenth of the length of the training series. Swingler [13] on the other hand says that the number of hidden neurons should never be more than twice the number of input neurons.

4.4 Normalizing data

The first neuron in the nets has in my cases a transfer function of arctangent form. This normalizes all the data to the range (-1,1). If the data has absolute values that are much larger than unity the transfer function maps them to values close to -1 or 1 with small differences. Much better data to work with for the network is if they are normalized earlier. One could expect that this should be best if the values were in a symmetric range round zero but in fact my best results are with the inputs in the range (0,0.8).

The performance was also improved if the target values were normalized the same way.

4.5 Validation of the training time

A neural network always minimizes the total sum squared error in the series it is trained on. If one trains the network on a problem that is solvable with this network architecture it always approaches zero if it can run for a long time. It isn't necessarily true that the best network is the one that has the lowest error in training. If the network has more neurons and thereby more freedom than necessary it can be overtrained, as discussed in section 4.3.

From the beginning the network weights are held quite small and close to each other. During training the weights get more different from each other as they are updated. From the beginning they act to take the output series closer to the target. When the output series is closer to the target in general the weights change to get a perfect fit to the training series but this can be done in many ways and the network can even get a worse performance on data outside the training series because of unrealistic weights.

In order to avoid that the networks get overtrained I've used a method with validation stop in the training. This works in the way that after every training epoch the network was simulated with another part of the series than that it was trained on. The error from this simulation is compared with the error of the former simulations. When there have been several training epochs in a row with no decrease of this validation error the training is stopped. [5]

5 Training trials

5.1 Ionosonde data

In this simulation I've taken the group sunspot number defined in eq 4 and $F_{10.7}$ and tried to correlate them to the f_oE and f_oF2 parameters in the ionosonde data.

This data is known from earlier to be rather well correlated so the main purpose with this was to check that the network really predicts the ionosonde data in a good way to see that the methods work out as thought.

There are also thoughts that the variations in the magnetic fields represented by the geomagnetic A_p index could be correlated to the variations in the plasma density in the ionosphere. Therefore I have made all training trials with and without the A_p index to see if it improves the performances.

5.1.1 Used data

The ionosonde data is from the ionosonde at IRF in Kiruna. They are from the time January 1958 to December 1995 with several longer and shorter periods of missing data. The ionosonde makes one readout every hour and I have used the average from the values in the three hours 10, 11 and 12 am. These seemed to be quite near each other and together covered most of the days with available data at all. For the missing data the latest value before has been used to fill the gap to avoid many small breaks since this should damage the time dependence. The target series with these two parameters is shown in fig 4.

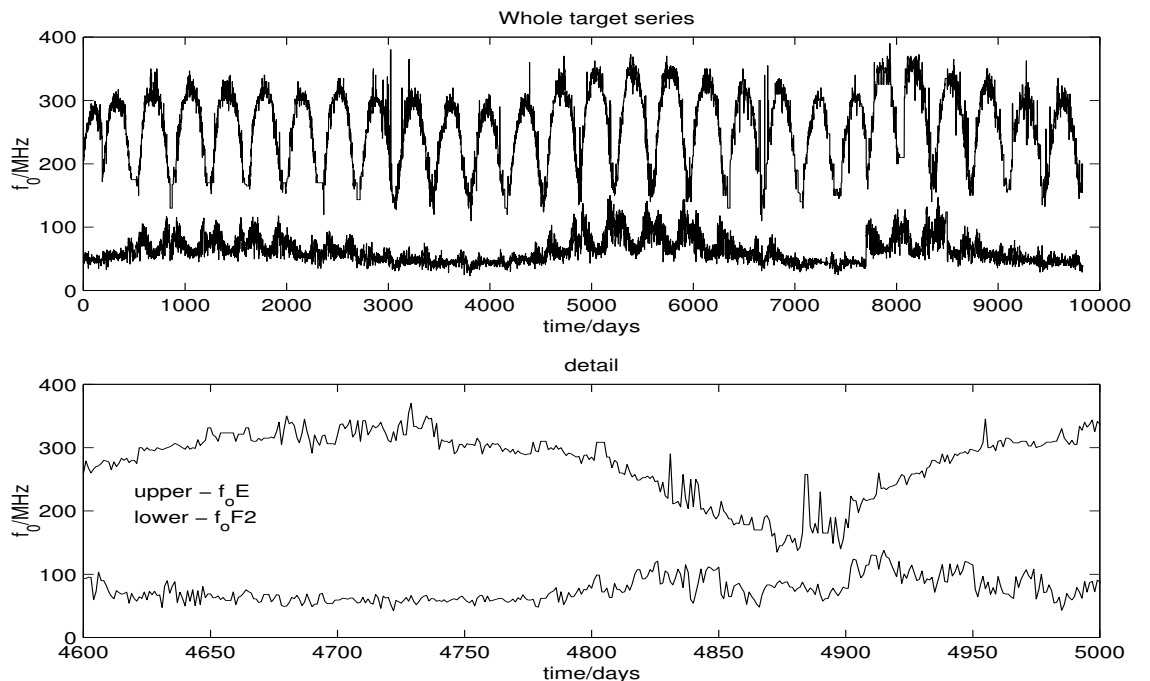


Figure 4: Target for the simulation. f_oE and f_oF2 parameters from the ionosond at IRF Kiruna average from three hours in the middle of the day.

These parameters are chosen because their different characteristics. f_oE is directly related to the EUV-flux while the f_oF2 is influenced by transport mechanism as discussed in section 3.2.1

I have tried several different input datasets to the network. In all tests I have fed the network with a sinus curve with 365.25 days period to take into account the variation according to the time of the year.

The solar spot number are the group sunspot number defined in eq 4 according to [6]. The data was taken from the National Geophysical Data Center's FTP-archive on the address

ftp://ftp.ngdc.noaa.gov/STP/SOLAR_DATA/SUNSPOT_NUMBERS/GROUP_SUNSPOT_NUMBERS/illdata
The sinus curve and solar spot number input data is shown in fig 5.

The $F_{10.7}$ is taken from World Data Center C1 for Solar-Terrestrial Physics at the url: <http://www.wdc.rl.ac.uk/cgi-bin/wdcc1/secure/wdccdata> the dataseries are complete and with only one measurement per day. The A_p index is taken from the same source. These two input parameters are shown in figure 6.

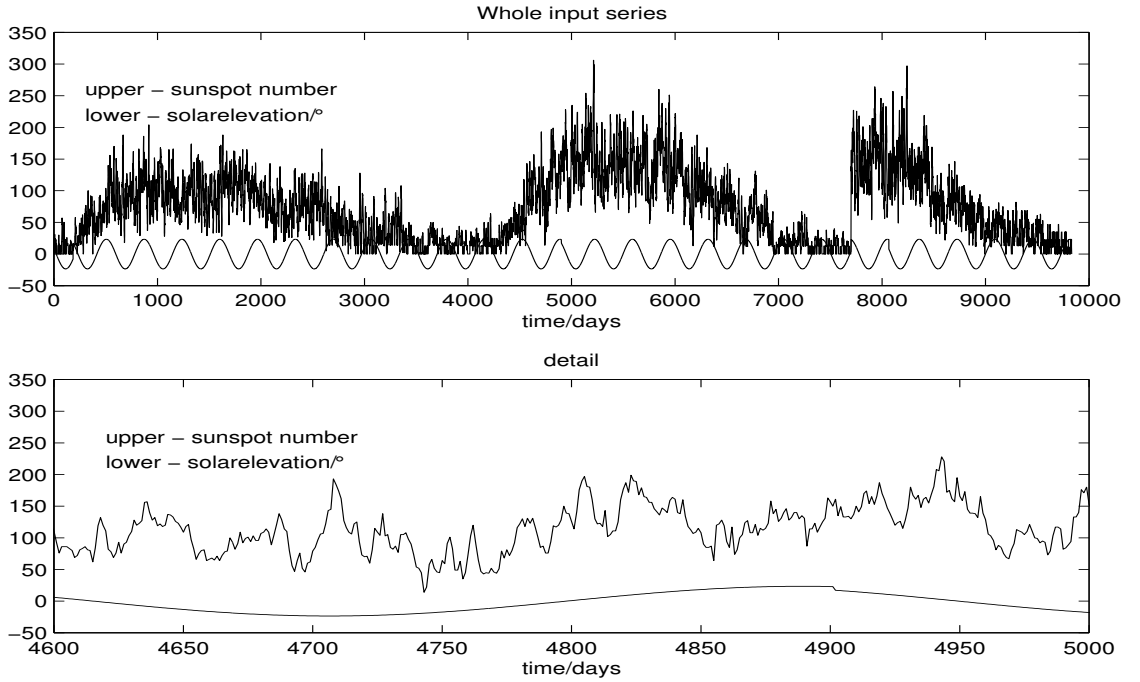


Figure 5: Solar elevation and group sunspot numbers input data series. The series is covering the time from jan -58 to dec -97 with several missing parts as seen from the discontinuities in the solar elevation.

In all of the simulations in this part the Levenberg-Marquardt training algorithm has been used because of its fast training.

5.1.2 Number of neurons for best performance

In order to find out how many hidden neurons are needed to make the prediction well I made a series of trainings for 1 to 30 neurons and studied how the statistical parameters ρ defined by eq (20) and ARV defined by eq (21) depend on the number of neurons. All trainings performed with a validation stop to protect the networks from being overtrained and to give the same conditions for all of the trainings.

In this test the input parameters was the solarelevation and the $F_{10.7}$.

5.1.3 Different input parameters

From the statistical test results in section 6.1 I decided by looking in the figures 8 and 9 that about five neurons seems to be able to solve the problem. As it does not seem to be too sensitive to how many neurons you use I decided to use ten independently from how many input neurons I use. That should be enough even to solve the problem with all four input parameters in the same

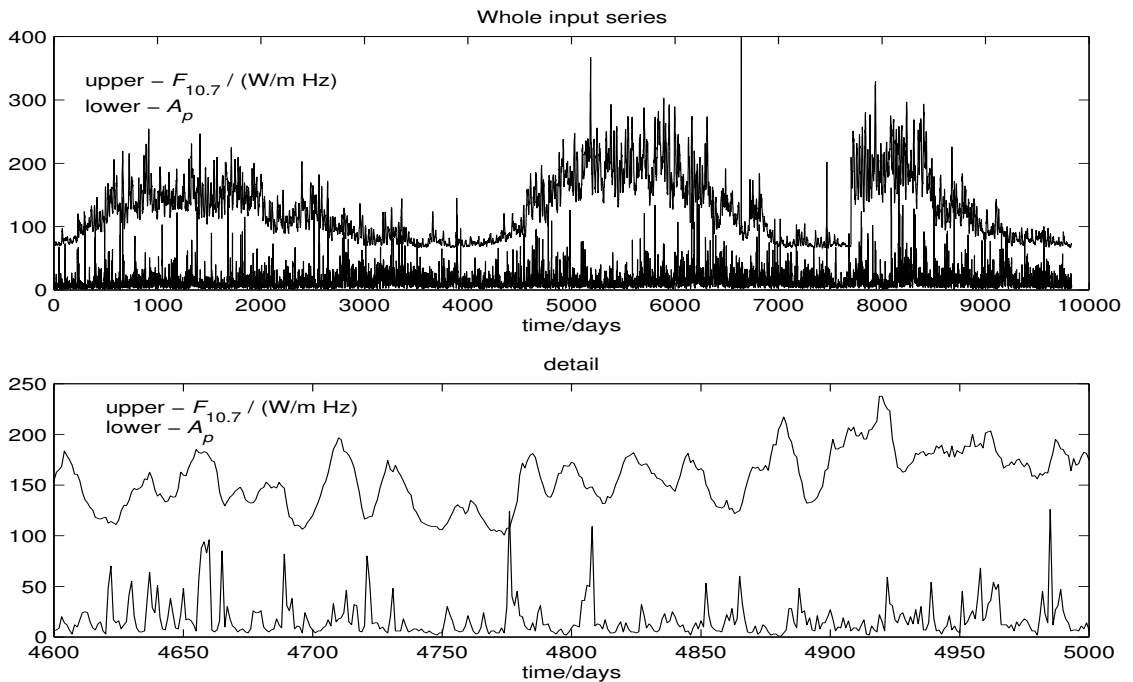


Figure 6: $F_{10.7}$ and A_p input data series. The series is covering the same time interval as the sunspot number in fig 5 and the ionosond data in fig 4.

time. Exactly as in the test with different numbers of neurons I plotted the best performance network from in this case four tries. The training was stopped with validation stopping.

5.1.4 Time delay networks

One test has been made to see whether time delay networks make a better performance. The same number of neurons as for the elman networks has been tested and all of them have been tested for different number of delay steps. The same inputs as in the test for the most efficient number of neurons for Elman networks (sec 5.1.2) has been used i e solar elevation and $F_{10.7}$. The results from this test is presented in figures 12 and 13

As the training times especially with many neurons, both delay- and hidden layer- is rather long this has only been made once for each configuration. As in the trials with the elman networks validation stopping and Levenberg-Marquardt training algorithm has been used in the trainings. In the treaining with many neurons the reduced memory algorithm is used. The 25 and 30 neurons test with 20 delay steps could not be done on the computer I worked on by deviding the Jacobian in two submatrixes and I saw no meaning in making a very slow try with another algorithm.

5.2 Weather data

I have done one trial on weather data in order to see if there are any correlations that can be seen with the network between the solar activity and the weather data.

This test was made with the enhanced gradient descent algorithm with momentum as this was the fastest available algorithm that could be used on the computer I was working. An alternative to this might have been to divide the training part several parts to be able to use

the Levenberg-Marquardt algorithm.

The test was first done with just the raw data day by day and after that I made a try with weekly running mean for each day to get a better signal to noise ratio.

5.2.1 Used data

The purpose with the work was to see the local effects in the Kiruna area in tropospheric parameters from the solar activity. The availability data from most of the stations in the neighborhood is only since the 80's and cover just about one solar cycle. There are one series that is significantly longer, that is the series of temperature and air humidity from Abisko scientific station that covers the whole time since first january 1913.

In order to use as old data as there are available from Abisko the problem is to find data about the solar activity for such a long time. The only data available is the solar sunspot number.

The inputs, the solar elevation and sunspot number, are the same as in the earlier as shown in fig 5. The target series of temperature and air humidity are shown in fig 7.

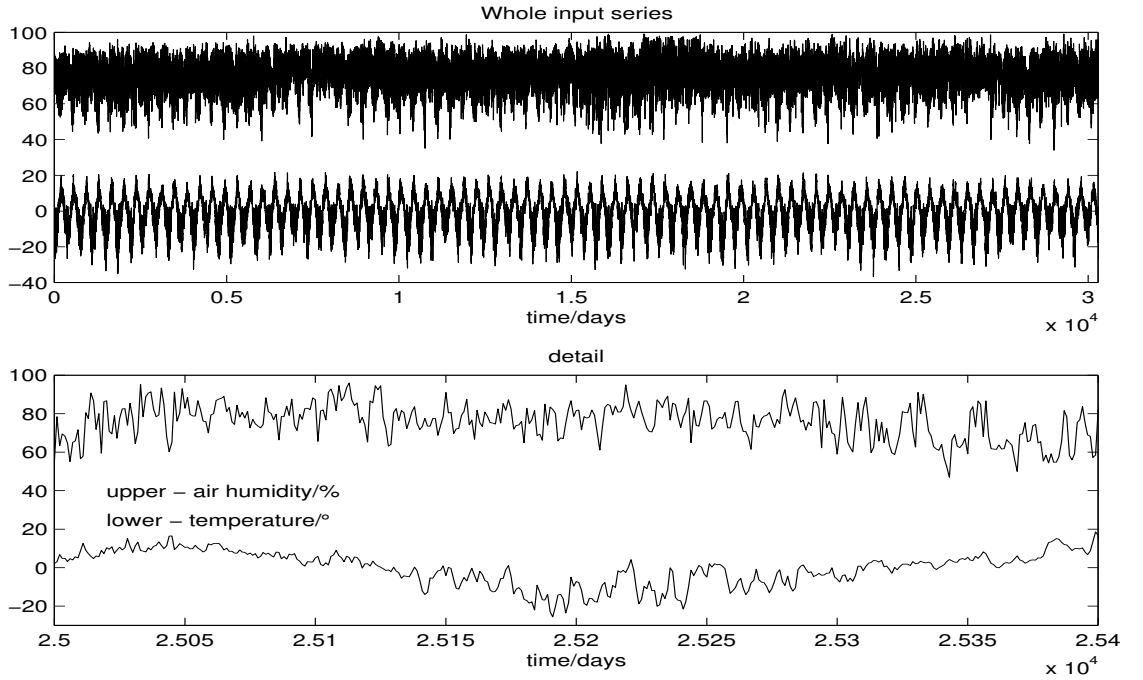


Figure 7: Target series for the simulation. Temperature and air humidity measurements from Abisko scientific station from 1913 to 1997

5.3 Error analysis

It is important to see whether the predictions of the neural network are statistically significant. One can then look at the linear correlation coefficient ρ

$$\rho = \frac{\sum_{\mu=1}^N (T_{\mu} - \langle T \rangle)(O_{\mu} - \langle O \rangle)}{\sqrt{\sum_{\mu=1}^N (T_{\mu} - \langle T \rangle)^2} \sqrt{\sum_{\mu=1}^N (O_{\mu} - \langle O \rangle)^2}}, \quad (20)$$

where O_{μ} is the predicted values and T_{μ} is the true values [15].

Another parameter that is a good quality of the predicted values is the average relative variance (ARV), i.e. the mean squared error normalized by the variance of the data [15],

$$\text{ARV} = \frac{\sum_{\mu=1}^N (T_{\mu} - O_{\mu})^2}{\sum_{\mu=1}^N (T_{\mu} - \langle T \rangle)^2}. \quad (21)$$

6 Results

The results are mostly discussed in terms of the statistical parameters ARV and ρ defined earlier. In all cases these are from the testpart of the series that have not been used for training or validation during the training.

6.1 Number of neurons for best performance

The results from the test with different number of neurons are given in figures 8 and 9. Both parameters are the best result over 6 tests for each configuration. The reason to plot the best result instead of some kind of average is that it is more interesting to know what is the best performance you can get from such a net than see what happens when they are stuck in a local minimum somewhere. To get a net that works out well it is better to make a few tries to avoid those ‘stuck’ networks. To give a more concrete picture of how the nets are doing their work figure 10 is a plot of the outputs from two different nets that have different success in their training. The first with 15 hidden neurons have for the two outputs $\rho_{f_oE} = 0.95$ and $\rho_{f_oF2} = 0.79$. The second with 9 hidden neurons have $\rho_{f_oE} = 0.64$ and $\rho_{f_oF2} = 0.39$.

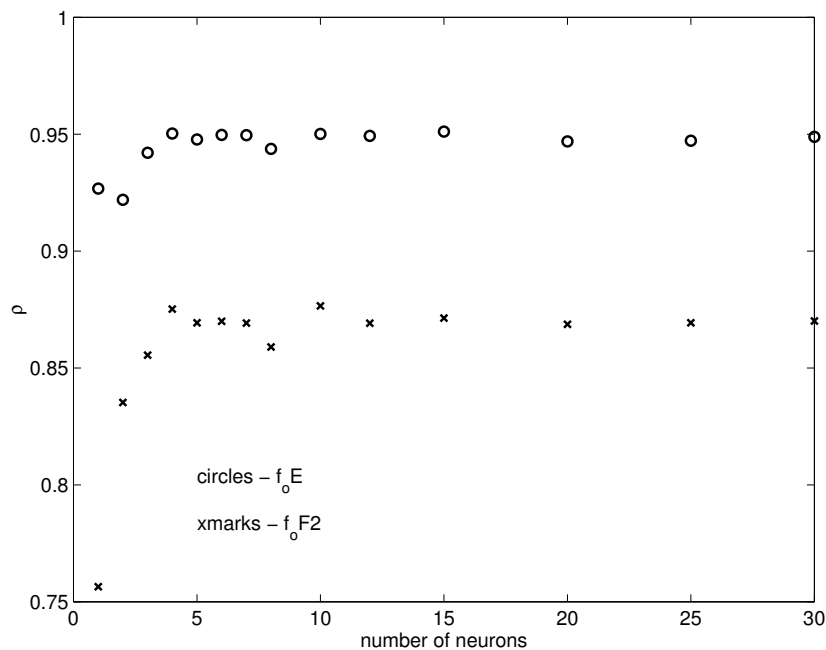


Figure 8: Correlation coefficient between output from net and real value taken on a test series with different number of hidden neurons. The values are the best among six equal tests for each configuration. f_oF2 and f_oE ionosonde parameters simulated from given solar activity.

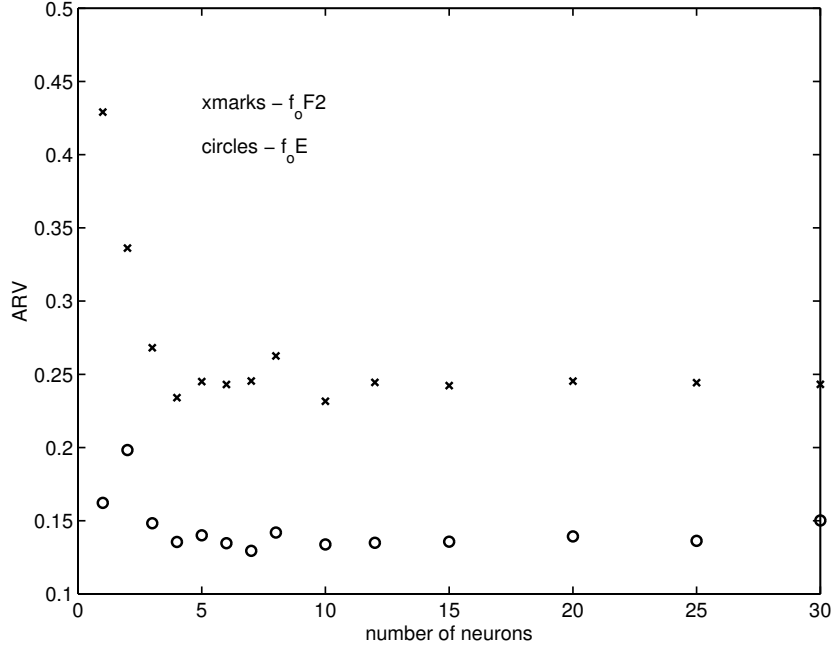


Figure 9: Average relative variance taken on a test series with different number of hidden neurons. The values are the best among six equal tests for each configuration. f_oF2 and f_oE ionosonde parameters simulated from given solar activity.

6.2 Different input parameters

The result from the trials with different input parameters is shown in fig 11 in the form of plotting their correlation coefficients.

Remarkable is to see that the f_oE parameter is so self consistent periodically that the network can predict it with a correlation coefficient of 0.9 even with just the sun elevation as input parameter.

The geomagnetic A_p index seem to add much information correlated to the f_oF2 parameter but almost nothing that can help to predict the f_oE . In the case with just solar elevation and $F_{10.7}$ flux inputs it even decreases the performance. One must take into account the fact that there are no absolute best solution for the network problem, so the performance is a bit of gambling of where in the error function landscape one ends up.

In the comparison between the trainings done with both $F_{10.7}$ and sunspot numbers one can see that the performance is decreased with the sunspot number. Theoretically the performance shall just be better the more information you put in to the system, all redundant inputs shall be ignored by that their weights approaches zero. There are two explanations for this as I can think of. Either is the signal to noise ratio decreased in the sunspot numbers and they do not contain any information that the $F_{10.7}$ does not and the training is stopped before all redundant information is ignored by the network. The other explanation I have is that the networks freedom is too small for so many inputs so that ten hidden neurons is not enough.

6.3 Time delay networks

The statistical parameters ρ and ARV for this test is shown in figures 12, 13 and 14.

The time delay networks was just tested once in each combination of number of delay and

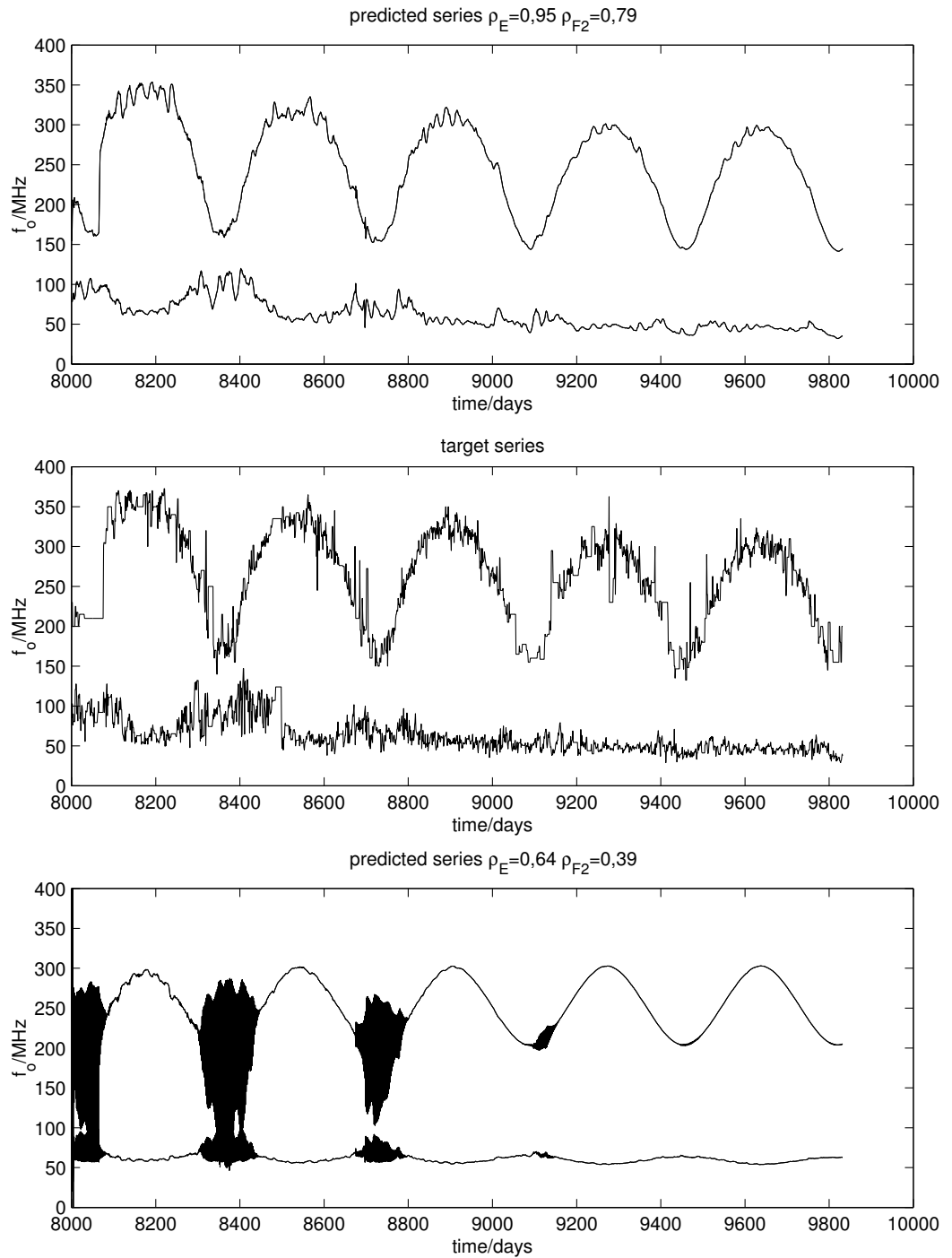


Figure 10: Simulation with two network that made different success in their training. The inputs where solar elevation and $F_{10.7}$ and the target that is shown in the middle for comparison is the f_oE and f_oF2 ionosonde parameters. results from testpart of the series.

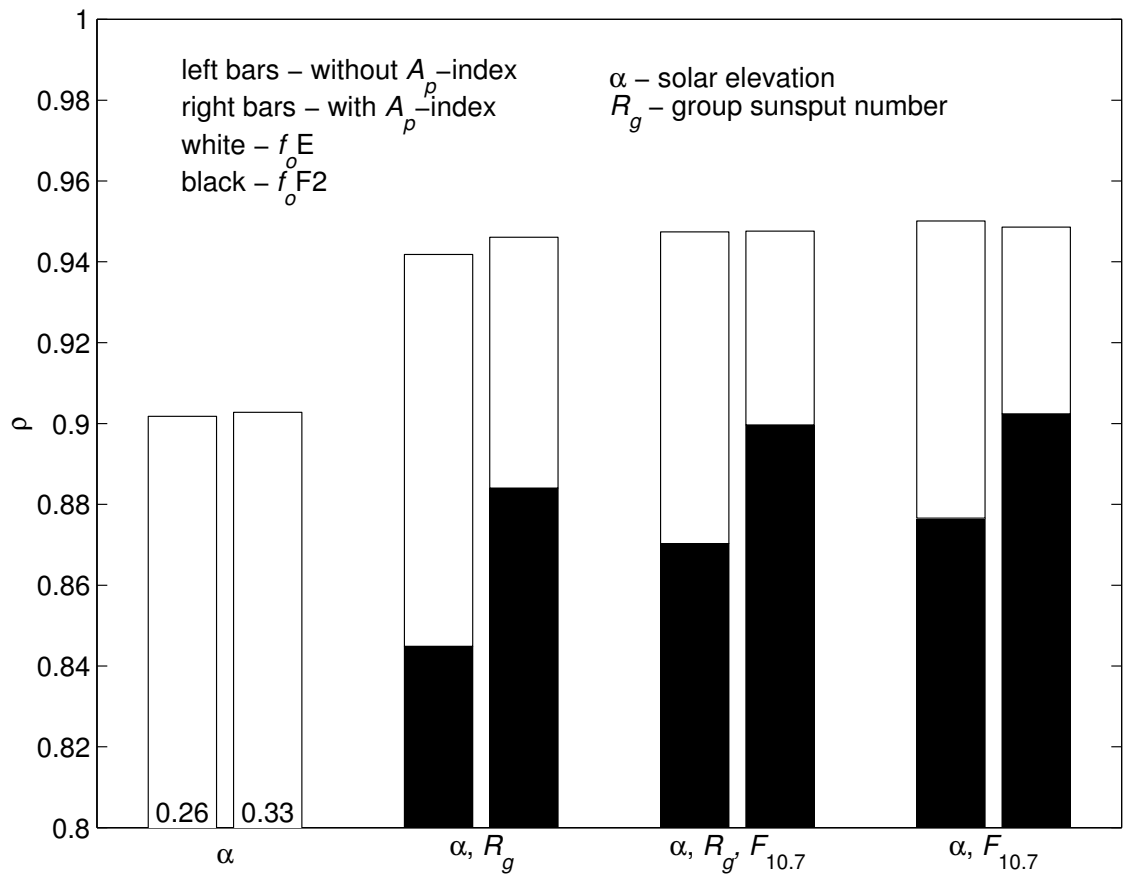


Figure 11: Correlation coefficient between output from net and real value from tests with different inputs to the networks. All done with elman networks with ten hidden neurons. f_oF2 and f_oE ionosonde parameters simulated from given solar activity.

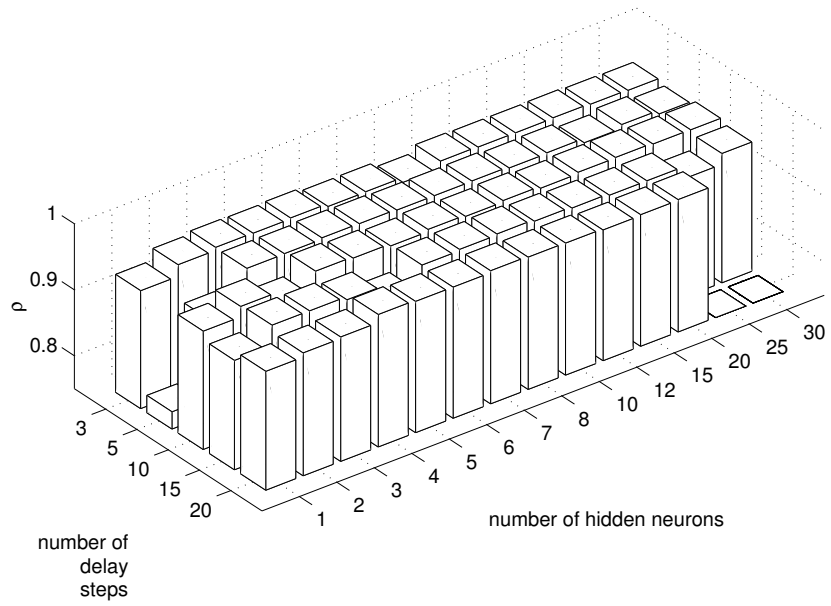


Figure 12: Correlation coefficient between output from net and real value of f_oE taken on a test series with different number of hidden and delay neurons with a time delay network architecture. f_oF2 and f_oE ionosonde parameters simulated from given solar activity. Simulations for 20 delay steps and 25 resp 30 hidden neurons are not made.

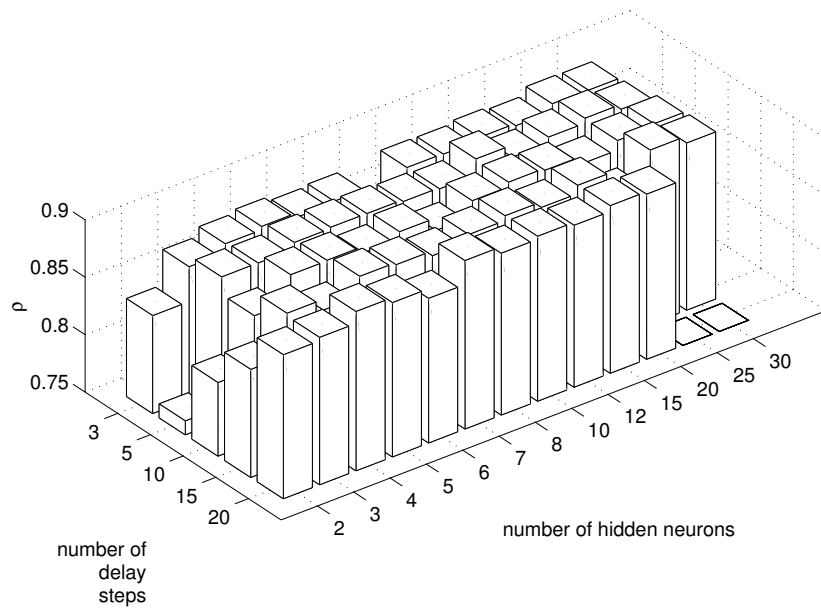


Figure 13: Correlation coefficient for f_oF2 in the same test as in fig 12

hidden neurons. So in comparison between the ρ and ARV values between the earlier result in figure 8 and the result from this test in figures 12 and 13 one should take into account that the former are the best out of four tests.

6.4 Weather data

The test with the weather data shows that there is no good success in correlating these to the solar variations. The correlation coefficients and average relative variance for the different trials are as shown in table 1. The predicted series for the two trials are shown in figure 15.

	α only				α and R_g			
	daily		weekly		daily		weekly	
	ρ	ARV	ρ	ARV	ρ	ARV	ρ	ARV
Temperature	0.60	0.83	0.85	0.28	0.47	0.88	0.77	0.41
Air humidity	0.29	1.15	0.44	0.84	0.16	1.12	0.43	0.82

Table 1: Statistical parameters from trial to predict air moisture and temperature in Abisko from solar elevation (α) and group sunspot numbers (R_g).

7 Conclusions

As a physical model the interaction between solar activity and weather parameters seems to be a much too big system so study at the same time and that may be the reason why there was no success in correlating these data.

Since the nets was far more difficult to handle and did not seem particularly efficient in extracting the signal from such noisy data as weather parameters most of the time spent in this work has been spent in just studying this problem and the response from different network architectures and sizes.

8 Acknowledgement

I want to thank all people at Swedish Institute of Space Physics for their great support during this work. Especially I want to thank Sheila Kirkwood and Carl-Erik Magnusson, for their creative criticism while reading my manuscripts and help with many problems, Björn Eriksson for many good advises and ideas and Hans Nilsson, Peter Wintoft, Runar Jørgensen and Christer Jurn for help on special topics.

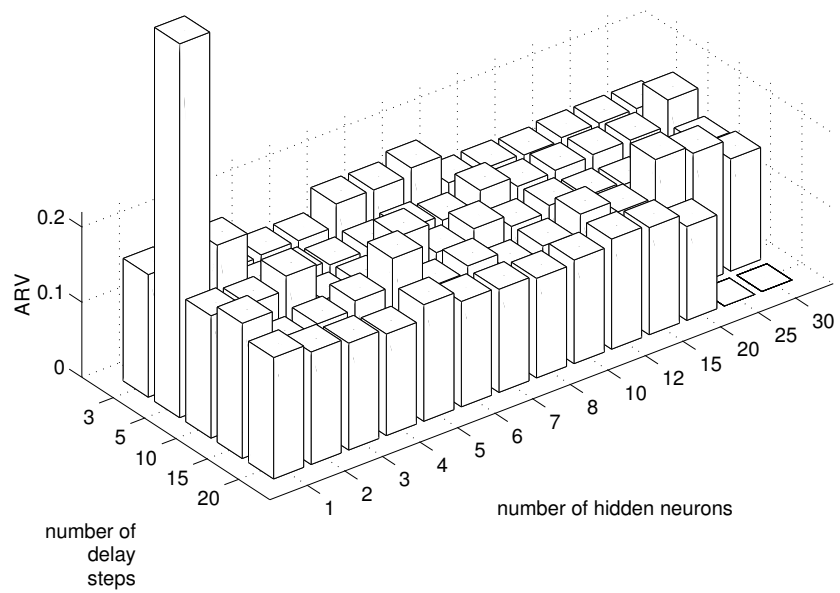


Figure 14: Average relative variance for $f_o E$ in the same test as in fig 12

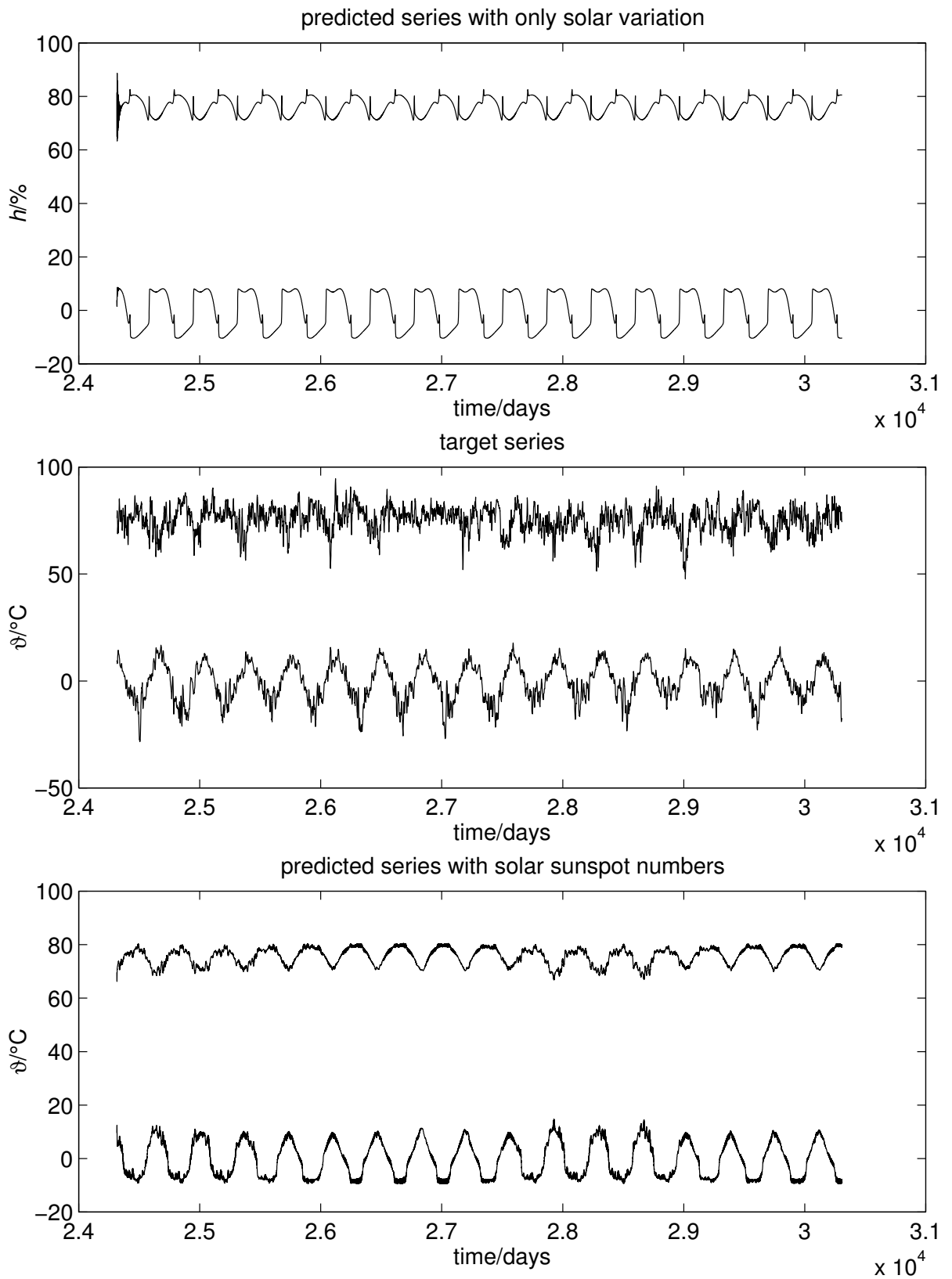


Figure 15: The predicted series for temperature and air humidity in Abisko form solar elevation and group sunspot numbers. The target in the middle plot for comparison.

A Matlab scripts

The scripts written during this work are only for practical reasons to use the code in Matlab in an easier way. No new programming code or algorithms have been written.

For typographical reasons long rows has been broken with a \ in the end in this printout. These do *not* belong to the code!

A.1 Comments to scripts

The files are matlab functions and they include help ?? which can be got by the help command in matlab if the file is in the matlab search path.

trainnet.m The main purpose with the file was to simplify the handling of the timedependent networks used (recurrent and delay types). These must have the data in cell array form to train the network as a timeseries. This script allows input as norml matrixes that for me seems more easy to handle. It uses several functions included in matlab 5.2 neural network toolbox and the files normal.m and denormal.m described below. As it in my case is more convenient to handle the whole dataseries available in a single series I wrote the file to take the whole series at one time and then split it into three parts for training, validation and testing. It uses the validation stop included in the neural network toolbox.

The file takes as inputs the training- and target series, network architecture (the ones used in this work), number of hidden neurons, length of delay series, splitting points for series in training validation and test parts, training algorithm and learning function. The last two are directly forwarded to the initiation command and the functions to choose from are the ones according to the Matlab Neural Networks manual [5, page 13-132 resp 13-136].

The outputs from the file are the network object and a vector of the linear regression coefficients ρ according to eq 20.

simunet.m This script simulates for the neural net trained with trainnet.m, if one wants to go back to see the simulated output from an earlier trained network.

normal.m The script normalizes matrices linearly so every row is in the range (0,0.8). The purpose was to normalize the input and training data. **N.B.** if one later would like to use the same transfer function (for example to scale the output data from the network to be compared with the target series), one must not delete or overwrite the input series.

denormal.m Rescale a matrix to another range using the inverse transfer function as in normal.m.

A.2 trainnet.m

```
function [net,rho,ARV]=trainnet(P,T,A,S1,D1,v,BTF)
%
%       [net,rho]=trainnet(P,T,A,S1,D1,v,BTF)
%       Initializes and trains an elman or delay two layer network.
%       Training stopped with validation stop.
%
%       inputs
```

```

%      P - input matrix of (column-) input vectors
%      T - target matrix of (column-) target vectors
%      A - network architecture. String; 'elman' or 'delay'
%      S1 - number of hidden neurons
%      D1 - length of delay
%      v - breakpoints for validation and test series parts of data.
%          1x2 vector with the two input numbers after which the
%          training series are splitted. First part:validation
%          series, second part:training series and third part:
%          test series.
%      BTF-backprop training function
%
% returns
%      net - the network object
%      rho - the regression coefficients for the regression of the
%            output vs the target. 3xn matrix for n target series,
%            each column includes the values for the testseries,
%            the training series and the validation series.
%      ARV - average mean variance. 3xn matrix arranged as rho.
%
% file output
%      rho'S1' - rho saved in ascii format
%      ARV'S1' - ARV          "
%
% version 1.1, Johan Arvelius, IRF, 980522

[b,t] = normal(P,T);
to = T;

[hP,lP] = size(P);
[hT,lT] = size(T);
i = [zeros(hP,1),0.8*ones(hP,1)];
switch lower(A)
    case 'elman'
        net = newelm(i,[S1,hT],{'tansig','purelin'},BTF);
    case 'delay'
        net = newfftd(i,[0:1:D1],[S1,hT],{'tansig','purelin'},BTF);
    otherwise
        error('Unknown network architecture.')
end
P = con2seq(b(:,v(1)+1:v(2)));
T = con2seq(t(:,v(1)+1:v(2)));
Pc = (b(:,v(1)+1:v(2)));
t2 = (to(:,v(1)+1:v(2)));

net.trainParam.epochs = 3000;
VV.P = con2seq(b(:,1:v(1)));
VV.T = con2seq(t(:,1:v(1)));

```

```

TV.P = con2seq(b(:,v(2)+1:1P));
TV.T = con2seq(t(:,v(2)+1:1P));
VVPc = (b(:,1:v(1)));
t3 = (to(:,1:v(1)));
TVPc = (b(:,v(2)+1:1P));
t1 = (to(:,v(2)+1:1P));

net = train(net,P,T,[],[],VV,TV);

switch lower(A)
case 'elman'
    sparp = ['print -dmfile trainelm' int2str(S1) 'plot'];
    sparn = ['save elm_net' int2str(S1) ' net'];
case 'delay'
    sparp = ['print -dmfile traindel' int2str(S1) '_' int2str(D1) 'plot'];
    sparn = ['save del_net' int2str(S1) '_' int2str(D1) ' net'];
end

eval(sparp)
eval(sparn)
TV.P;
os1 = sim(net,TV.P);
os2 = sim(net,P);
os3 = sim(net,VV.P);
o1 = seq2con(os1);
o1 = cell2struct(o1,'as');
o1 = o1.as;
o1 = denormal(o1,to);
o2 = seq2con(os2);
o2 = cell2struct(o2,'as');
o2 = o2.as;
o2 = denormal(o2,to);
o3 = seq2con(os3);
o3 = cell2struct(o3,'as');
o3 = o3.as;
o3 = denormal(o3,to);

for n = 1:hT
    [mt(1,n),bt(1,n),rt(1,n)] = postreg(o1(n,:),t1(n,:));
    [m(1,n),b(1,n),r(1,n)] = postreg(o2(n,:),t2(n,:));
    [mv(1,n),bv(1,n),rv(1,n)] = postreg(o3(n,:),t3(n,:));
end

rho = [rt;r;rv];

spara = ['save -ascii rho' int2str(S1) ' rho'];
eval(spara)
for n = 1:hT

```

```

ARV1(1,n) = ((t1(n,:)-o1(n,:))*(t1(n,:)'-o1(n,:)'))/((t1(n,:)-mean(t1(n,:)))*\
(t1(n,:)'-mean(t1(n,:))));
ARV2(1,n) = ((t2(n,:)-o2(n,:))*(t2(n,:)'-o2(n,:)'))/((t2(n,:)-mean(t2(n,:)))*\
(t2(n,:)'-mean(t2(n,:))));
ARV3(1,n) = ((t3(n,:)-o3(n,:))*(t3(n,:)'-o3(n,:)'))/((t3(n,:)-mean(t3(n,:)))*\
(t3(n,:)'-mean(t3(n,:))));
end

ARV = [ARV1;ARV2;ARV3];

spara = ['save -ascii ARV' int2str(S1) ' ARV'];
eval(spara)

```

A.3 simunet.m

```

function [rho,ARV,o1]=simunet(net,P,T,v,S1)
%
%       [rho,ARV]=simunet(net,P,T,v,S1)
%       Simulates and gives the statistical parameters for a network
%       simulation
%
%       inputs
%       P - input matrix of (column-) input vectors
%       T - target matrix of (column-) target vectors
%       net- network object to be used in simulation
%       v - breakpoints for validation and test series parts of data.
%           1x2 vector with the two input numbers after which the
%           training series are splitted. First part:validation
%           series, second part:training series and third part:
%           test series.
%
%       returns
%       rho - the regression coefficients for the regression of the
%           output vs the target. 3xn matrix for n target series,
%           each column includes the values for the testseries,
%           the training series and the validation series.
%       ARV - average mean variance. 3xn matrix arranged as rho.
%
%       file output
%       rho'S1' - rho saved in ascii format
%       ARV'S1' - ARV          "
%
% version 1.0, Johan Arvelius, IRF, 980602

[b,t] = normal(P,T);
to = T;

```

```

[hP,lP] = size(P);
[hT,lT] = size(T);
P = con2seq(b(:,v(1)+1:v(2)));
T = con2seq(t(:,v(1)+1:v(2)));
Pc = (b(:,v(1)+1:v(2)));
t2 = (t(:,v(1)+1:v(2)));

VV.P = con2seq(b(:,1:v(1)));
VV.T = con2seq(t(:,1:v(1)));
TV.P = con2seq(b(:,v(2)+1:lP));
TV.T = con2seq(t(:,v(2)+1:lP));
VVPc = (b(:,1:v(1)));
t3 = (t(:,1:v(1)));
TVPc = (b(:,v(2)+1:lP));
t1 = (t(:,v(2)+1:lP));
os1 = sim(net,TV.P);
os2 = sim(net,P);
os3 = sim(net,VV.P);
o1 = seq2con(os1);
o1 = cell2struct(o1,'as');
o1 = o1.as;
o1 = denormal(o1,to);
o2 = seq2con(os2);
o2 = cell2struct(o2,'as');
o2 = o2.as;
o2 = denormal(o2,to);
o3 = seq2con(os3);
o3 = cell2struct(o3,'as');
o3 = o3.as;
o3 = denormal(o3,to);

for n = 1:hT
    [mt(1,n),bt(1,n),rt(1,n)] = postreg(o1(n,:),t1(n,:));
    [m(1,n),b(1,n),r(1,n)] = postreg(o2(n,:),t2(n,:));
    [mv(1,n),bv(1,n),rv(1,n)] = postreg(o3(n,:),t3(n,:));
end

rho = [rt;r;rv]

spara = ['save -ascii rho' int2str(S1) ' rho'];
eval(spara)
for n = 1:hT
    ARV1(1,n) = ((t1(n,:)-o1(n,:))*(t1(n,:)'-o1(n,:)'))/((t1(n,:)-mean(t1(n,:)))*\
        (t1(n,:)'-mean(t1(n,:))));
    ARV2(1,n) = ((t2(n,:)-o2(n,:))*(t2(n,:)'-o2(n,:)'))/((t2(n,:)-mean(t2(n,:)))*\
        (t2(n,:)'-mean(t2(n,:))));
    ARV3(1,n) = ((t3(n,:)-o3(n,:))*(t3(n,:)'-o3(n,:)'))/((t3(n,:)-mean(t3(n,:)))*\
        (t3(n,:)'-mean(t3(n,:))));

```

```

end

ARV = [ARV1;ARV2;ARV3]

spara = ['save -ascii ARV' int2str(S1) ' ARV'];
eval(spara)

```

A.4 normal.m

```

function [P,T] = normal(P,T)
%
% [P,T] = normal(P,T)
% Create input and output matrixes normalized to the range [0 0,8].
% N.B. do not overwrite the original series if you want to use the inverse
% of this function with denormal.m
%
% Inputs
% P = input matrix of (column-) input vectors
% T = target matrix of (column-) target vectors
% Outputs
% P = new input matrix of (column-) input vectors
% T = new target matrix of (column-) target vectors
%
% version 1.0, Johan Arvelius, IRF 980501

[a,b] = size(P);
for n = 1:a
    P(n,:) = 0.8*(P(n,:)-min(P(n,:)'))/(max(P(n,:)')-min(P(n,:)'));
end

[c,d] = size(T);
for n = 1:c
    T(n,:) = 0.8*(T(n,:)-min(T(n,:)'))/(max(T(n,:)')-min(T(n,:)'));
end

```

A.5 denormal.m

```

function O = denormal(O,T)
%
% O = denormal(O,T)
% Recreate the output from the network scaled to the same scale as
% the target before normalization with 'normal'.
%
% Inputs
% O = Output matrix from network
% T = Target matrix before normalization

```

```
%      Outputs
%      O = Rescaled output matrix
%
% version 1.0 Johan Arvelius, IRF 980519

[a,b] = size(T);
for n = 1:a
    O(n,:) = 1.25*O(n,)*(max(T(n,:)')-min(T(n,:)')) + min(T(n,:)'));
end
```

References

- [1] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University press, Oxford, 1995.
- [2] Rolf Boström. *Nationalencyklopedin*, volume 10, page 171. Bokförlaget bra böcker, 1993.
- [3] Asgeir Brekke. *Physics of the upper polar atmosphere*. Wiley, 1997.
- [4] Howard Demuth and Mark Beale. *Neural Network Toolbox Users Guide*. The Mathworks inc, Natick, version 2 edition, 1997.
- [5] Howard Demuth and Mark Beale. *Neural Network Toolbox Users Guide*. The Mathworks inc, Natick, version 3.0 edition, 1998.
- [6] Douglas V. Hoyt, Kenneth H Schatten, and Elisabeth Nesmes-Ribes. The one hundredth year of Rudolf Wolf's death: Do we have the correct reconstruction of solar activity? *Geophysical research letters*, 21(18):2067–2070, 1994.
- [7] Bengt Hultqvist. *Introduktion till Geokosmofysiken*. Natur och Kultur, Stockholm, 1967.
- [8] Margret G. Kivelson and Christopher T. Russell, editors. *Introduction to space physics*. Cambridge university press, 1995.
- [9] Ronald T. Merrill, Michael W. McElhinny, and Phillip L. McFadden. *The Magnetic Field of the Earth*. Number 63 in International Geophysics series. Academic press, 1996.
- [10] Robert L. Pherron. Magnetospheric substorms. *Reviews of geophysics and space physics*, 17(4):657–681, 1979.
- [11] Henry Risbeth and Owen K. Garriott. *Introduction to Ionospheric Physics*. Academic press, Inc, 1969.
- [12] Bradley E. Schaefer. Visibility of sunspots. *The Astrophysical Journal*, 411(2):909–919, 1993.
- [13] Kevin Swingler. *Applying Neural Networks A Practical Guide*. Academic press, London, 1996.
- [14] Douglas V. Hoyt and Kenneth H. Schatten. New information on solar activity, 1779–1818, from sir William Herschel's unpublished notebooks. *The Astrophysical Journal*, 384(1):361–384, 1992.
- [15] Jian-Gou Wu. *Dynamic Neural Network Studies of Solar Wind-Magnetosphere Coupling*. PhD thesis, University of Lund, 1997.
- [16] Jian-Gou Wu and Henrik Lundstedt. Prediction of geomagnetic storm from solar wind data using elman recurrent neural networks. *Geophysical research letters*, 23(4):319–322, February 1996.
- [17] Jian-Gou Wu and Henrik Lundstedt. A study of solar wind-magnetosphere coupling using neural networks. In W. Burke and T.-D. Guyenne, editors, *ESA SP-392 Proceedings Environment Modelling for Space-based Applications*, pages 277–284, Noordwijk, 1996. ESTEC.