# Neural networks as universal approximators

## A practical example of space weather modelling

***Fridrich Valach***

*Geomagnetic Observatory GPI SAS, Hurbanovo, Slovakia*

# Aims of the lesson

- Introducing an artificial neural network (NN) as a universal approximator.
- Convincing you that the NN is a useful tool for space weather (SW) modelling.
- Introducing Script_for_NN.m
  - A demonstration of use of NNs for space weather modelling (and an exercise).

# Preview of the lesson

- Space weather and geomagnetic activity (geomagnetic storm)
  - Introducing the Hurbanovo Geomagnetic Observatory GPI SAS
  - Geomagnetic field (GMF) – main field, ring current, Earth's magn. field deformed by solar wind– Earth's magnetosphere
  - GMF elements X, Y, Z or D, H, I
  - CMEs cause geomagnetic storms
  - Geomagnetic storms on magnetograms (H component of GMF)
  - Indexes of the geomagnetic activity (Kp and Dst)
- One single neuron – explanation
- One single neuron – an example of using it in space weather (SW) modelling
- Neural network with a hidden layer (a layer of hidden neurons) – description
- Universal approximation theorem
- Neural network with a hidden layer – examples of using it in space weather modelling
  - Geomagnetic activity forecasts based on solar wind observations in the L1 point
  - Geomagnetic activity forecasts based on the observations of solar energetic events
  - SEPs as an addidional input parameter for geomagnetic activity forecasts
  - Forecasting of SEPs based on the observations of solar energetic events
- Training of neural networks
  - What will be the inputs and outputs of our example SW model – database for the example
  - What does the training of neural networks mean – training patterns
  - How many hidden neurons are needed? – validation patterns
  - A script for NN training in octave – description of Scenario_NN.m
  - Exercise – How to work with Script_for_NN.m
  - Working the example (by the students)
- Summarizing the lesson

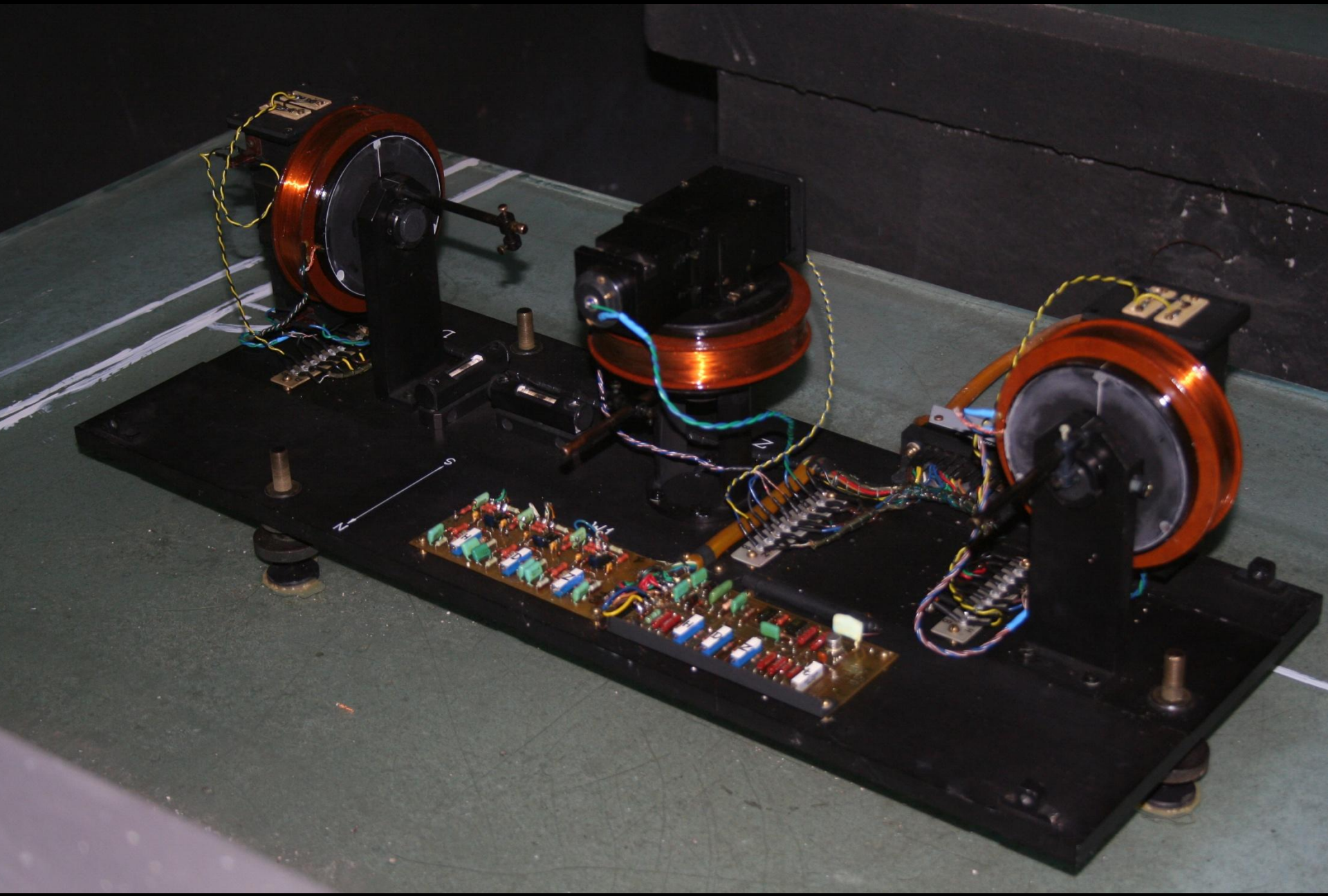
# Space Weather definition

- Space weather is the physical and phenomenological state of natural space environments. The associated discipline aims, through observation, monitoring, analysis and modelling, at understanding and **predicting the state of the** sun, the interplanetary and **planetary environments, and the solar** and non-solar **driven perturbations that affect them**; and also at forecasting and nowcasting the possible impacts on biological and technological systems.
- (*Brussels, November 2007*)

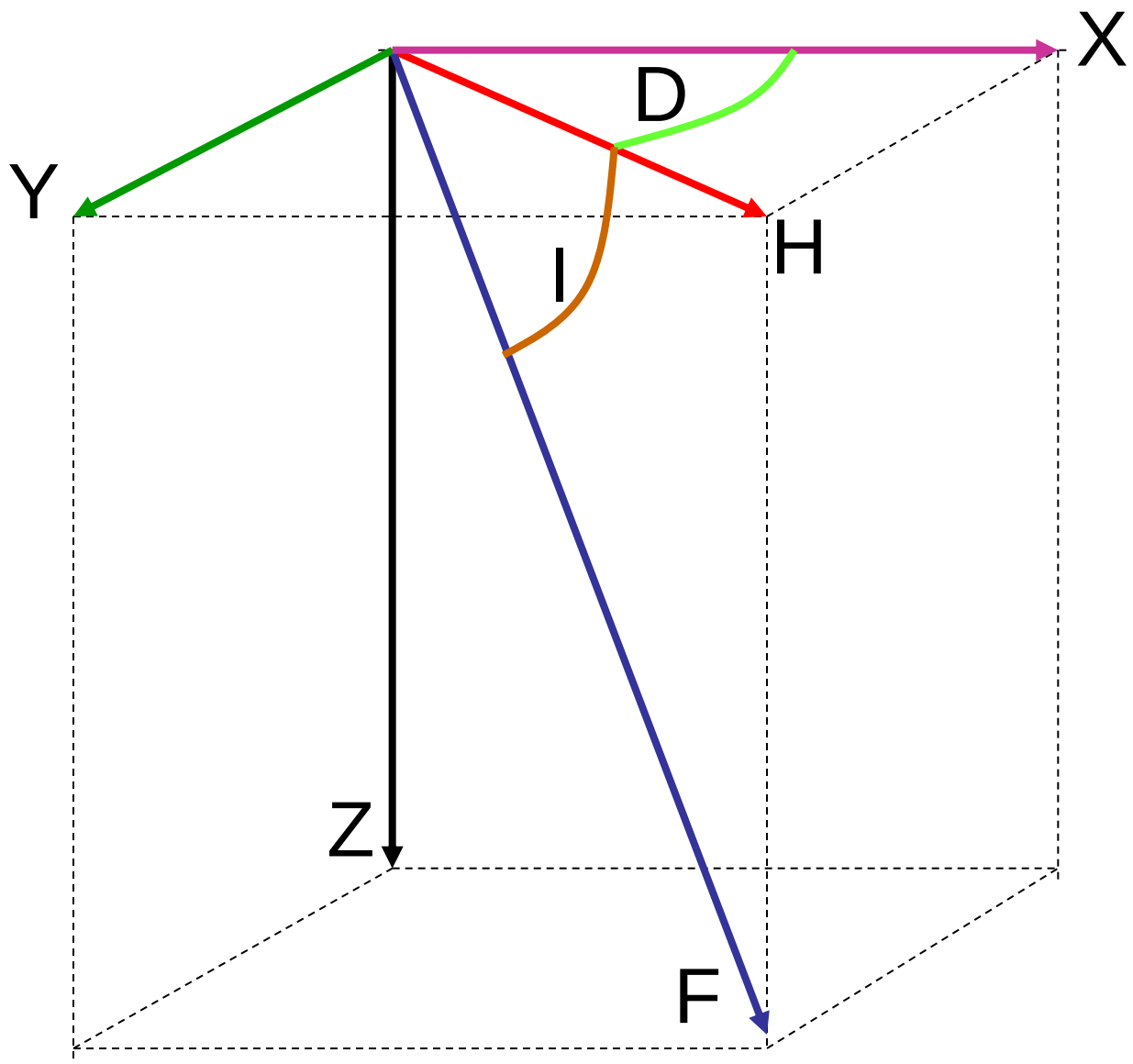
# Hurbanovo Geomagnetic Observatory

# Location of Hurbanovo GO

The founder of the Hurbanovo Geomagnetic Observatory, Dr. Miklós Konkoly Thége


Z
S
N
Z

# Geomagnetic field (GMF) elements




For SW research the H component is important.

# GMF at Hurbanovo as a vector

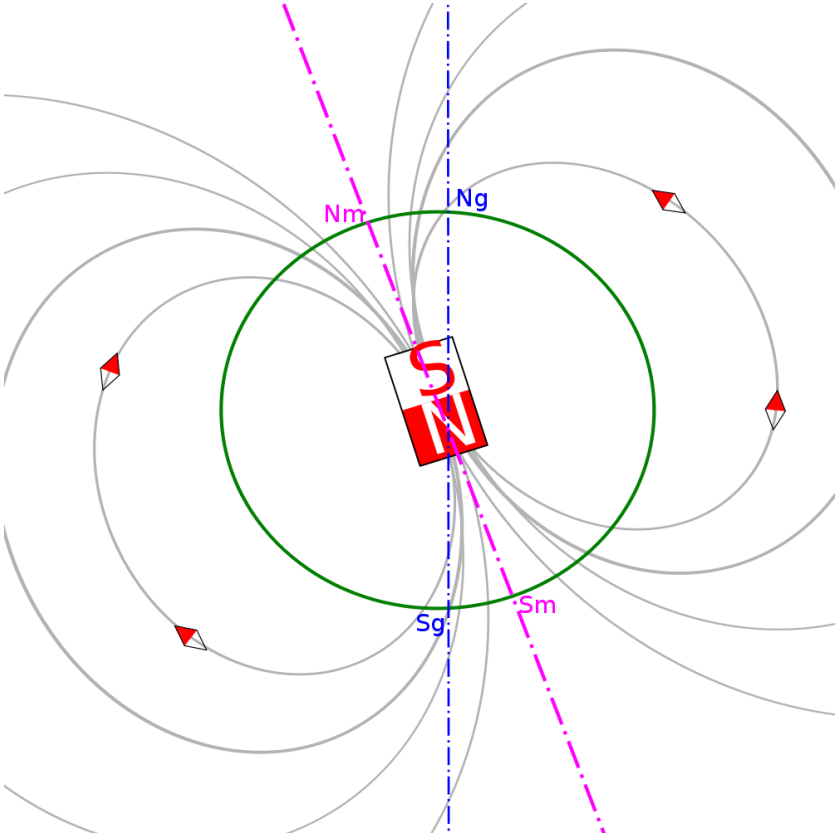
- Present-day values of the GMF at Hurbanovo GO:

- D = 3º 30,67’
- H = 21 030 nT
- I = 64º 14,5’
- T = 48 394 nT
- Z = 43 586 nT

# The GMF on the Earth

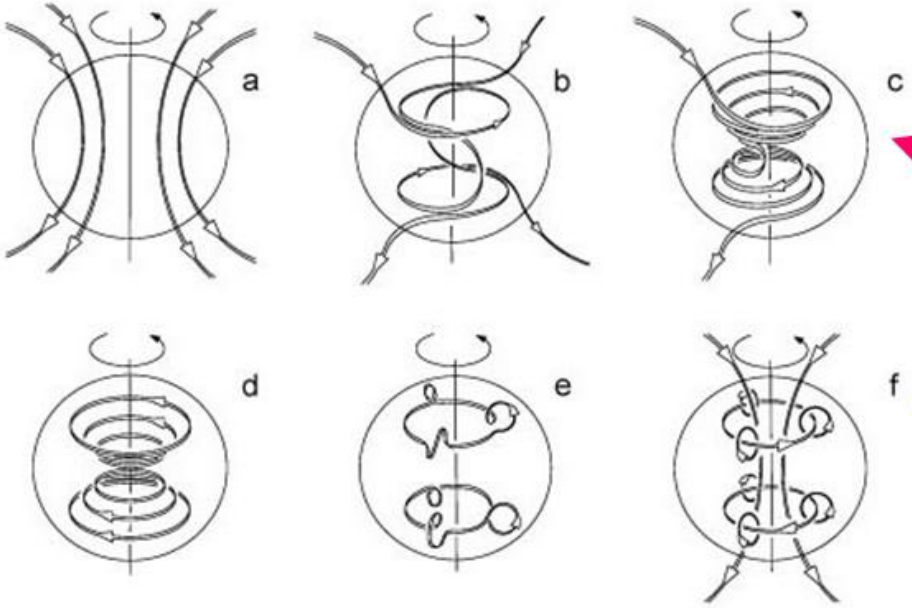
- The GMF on the poles is twice as high as at the equator:
  - approximately 70 000 nT at the poles
  - approximately 35 000 nT at the equator
- The axis of the magnetic dipole and the Earth's spin axis form an angle cca 11°.

**Near the Earth surface the GMF looks like a dipole magnetic field.**

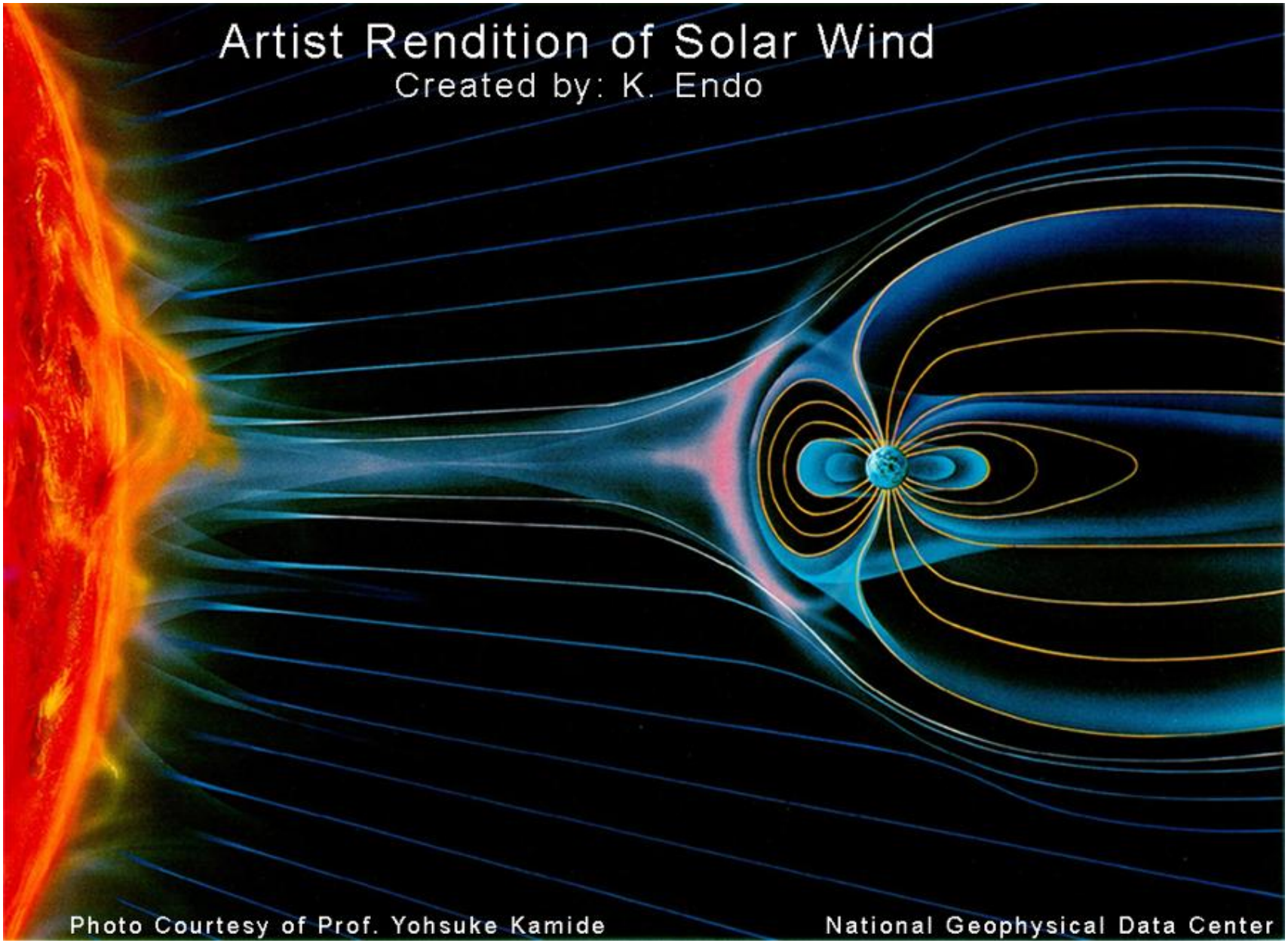



**The main part of the GMF is produced by the geodynamo (MHD dynamo) which works in the liquid Earth's core.**

**This part of the GMF is called the main field.**

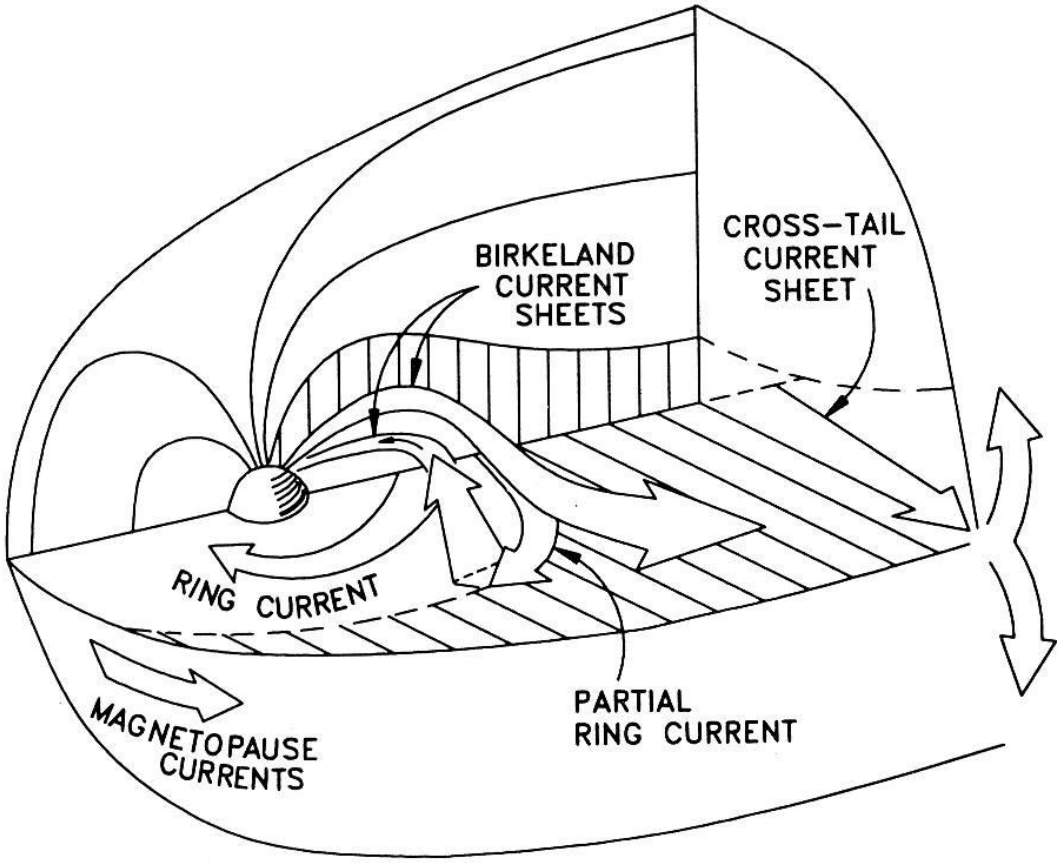


**Artists rendition of solar wind and it's interaction with the earth's magnetic field.**




Reference: http://www.weather.nps.navy.mil/~psguest/EMEO_online/module3/solarwindbig.jpg [cited 2011-08-13]

## Currents in the Earth's magnetosphere




Reference: en.wikipedia.org/wiki/Ring_current [cited: 2011-08-13]

# Earth's ring current

The ring current system consists of a band at a distance of 3 to 5 $R_E$. (In the Earth's equatorial plane.)

It produces a magnetic field which is opposite to the Earth's magnetic field.

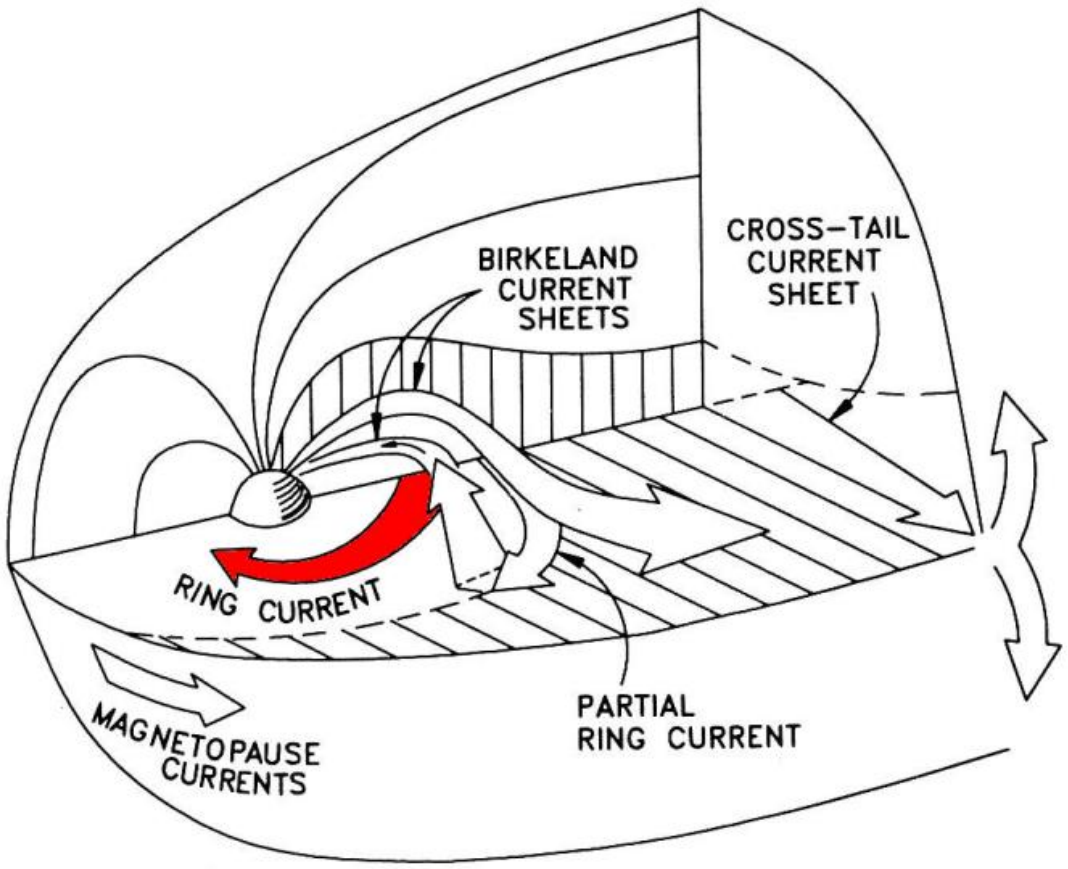
During a **geomagnetic storm**, the number of particles in the ring current increases. As a result there is a decrease of the geomagnetic field.

This can be observed directly at the GOs located near the Earth's equator. (On H component of the GMF.)

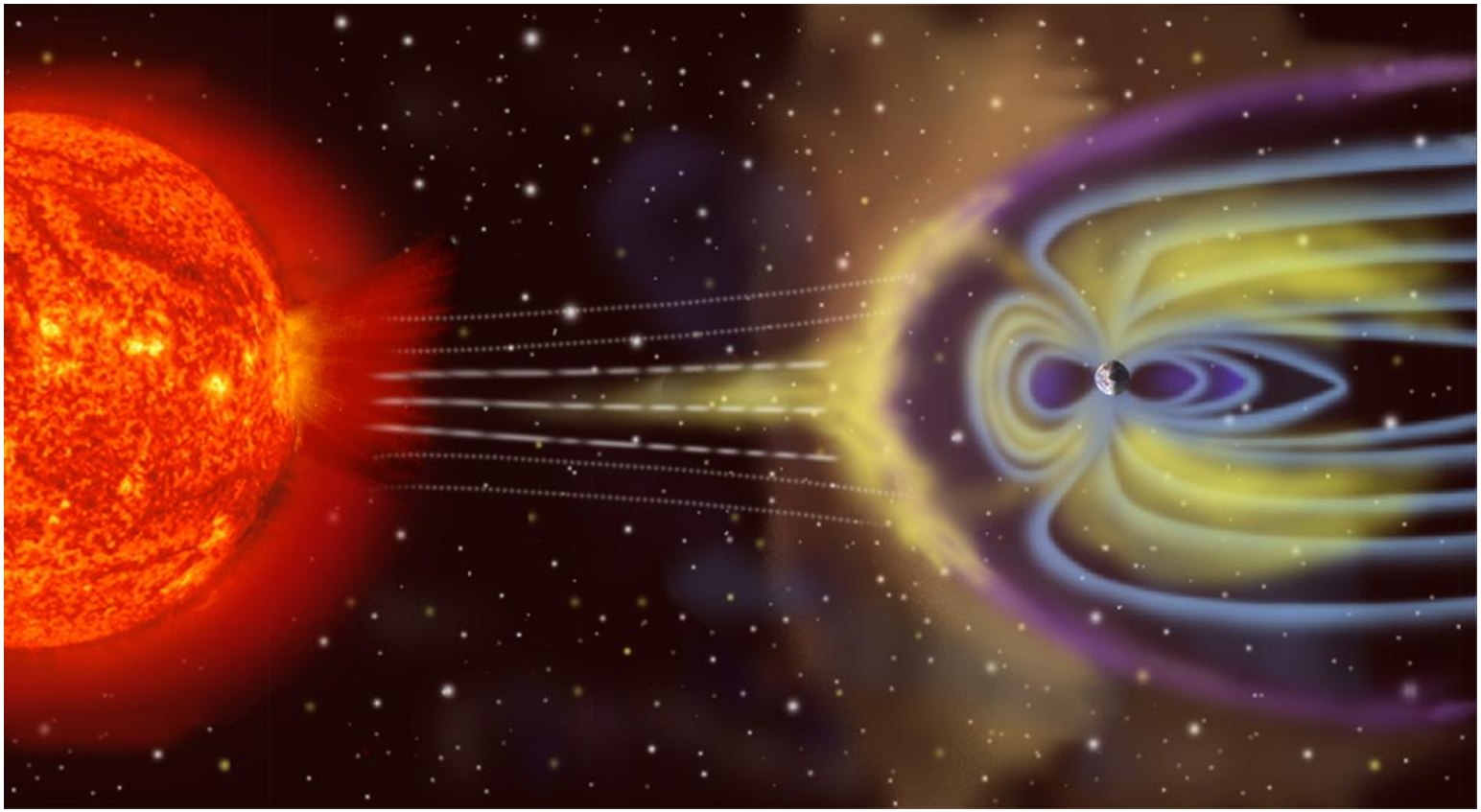
The decrease of the H component during a geomagnetic storm can be interpreted as Dst index:

Dst<-100 nT … intense storm

Dst<-200 nT … super-intense storm



# An approximate idea about the cause of the geomagnetic storm

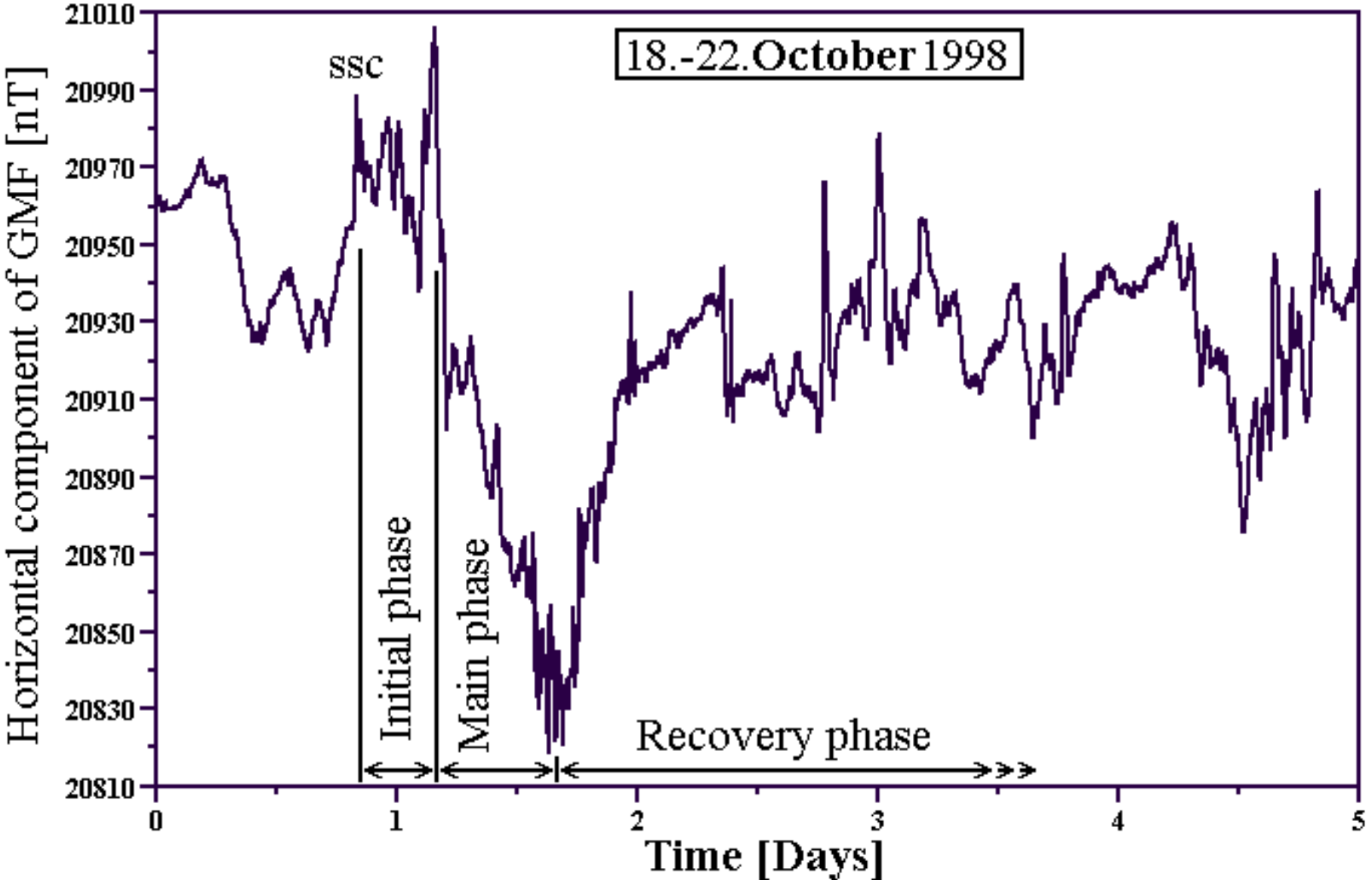


Reference: en.wikipedia.org/wiki/Magnetosphere [cited: 2011-08-13]

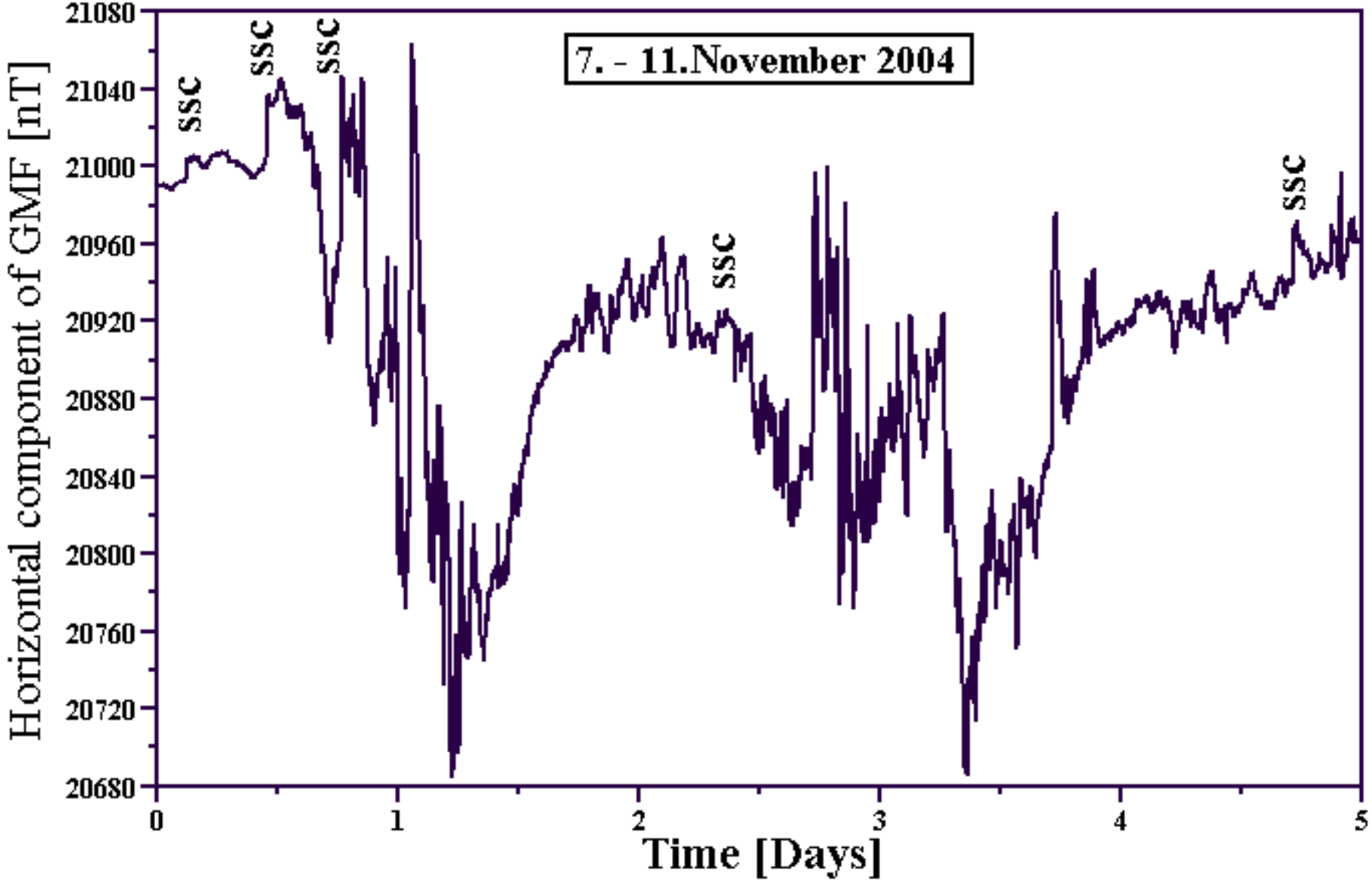
# An approximate idea about the cause of the geomagnetic storm

- An explosive process occurs on the Sun.
- A CME is shot to the Earth.
- 2 – 4 days: A cloud of the ejected solar plasma with a complicated structure of the interplanetary magnetic field (IMF) moving with the plasma reaches the Earth.
- Interaction with the Earth's magnetic field starts – the geomagnetic storm (ssc).

# Geomagnetic storm on a magnetogram, H component



# A more complicated geomagnetic storm



# Indexes of the geomagnetic activity

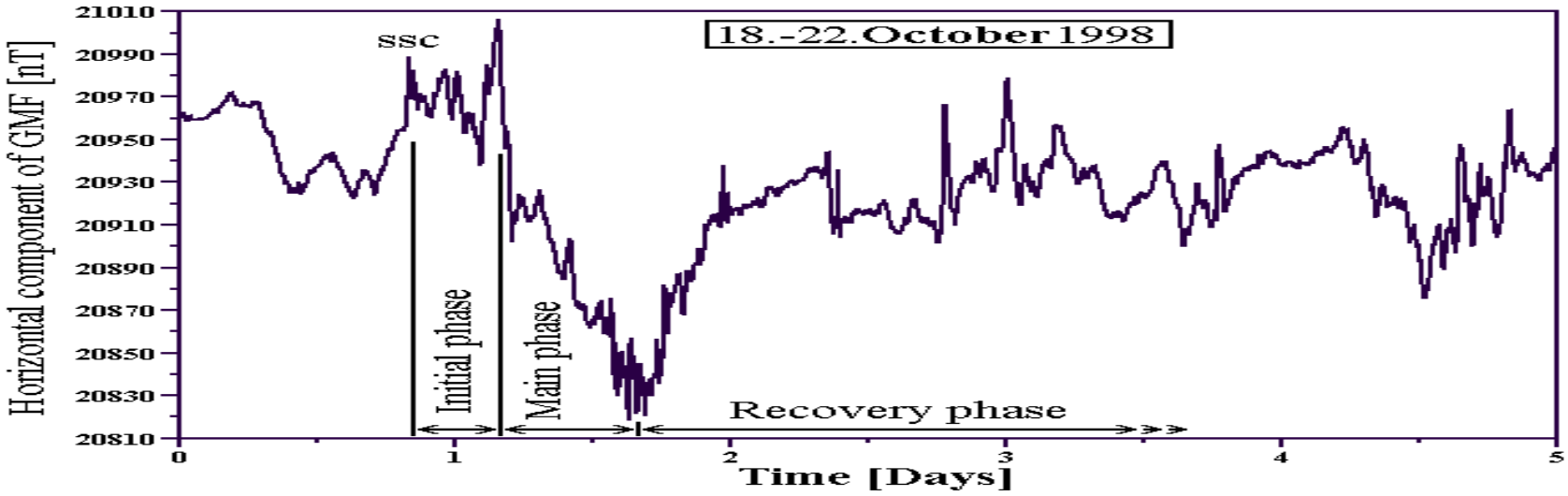
- Index Dst [nT] – Can be interpreted as the decrease of H-component at ground-based equatorial observatories
- Index K (0,1,2,…,9)

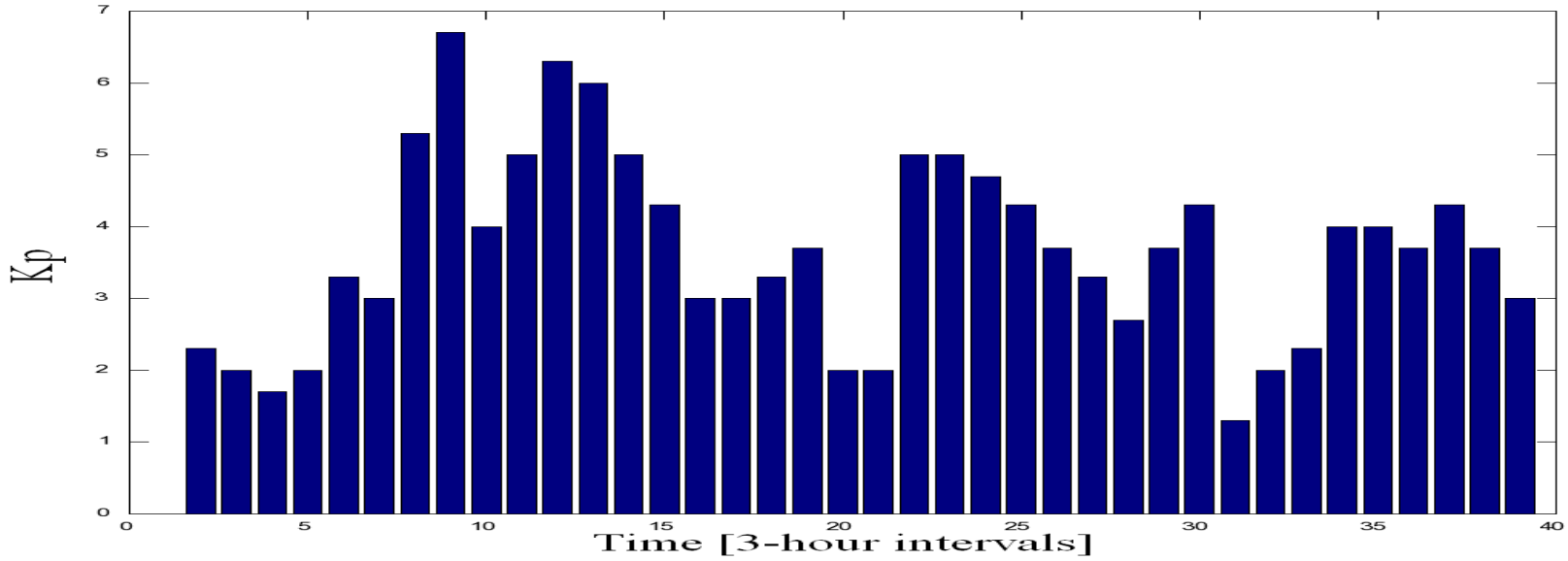
  Kp = 0 … quiet geomagnetic field

  Kp = 9 … extremely disturbed geomag. field
- Index Kp ($0_0$,$0_+$,$1_-$,$1_0$,$1_+$,$2_-$,$2_0$,$2_+$,…,$9_-$,$9_0$)

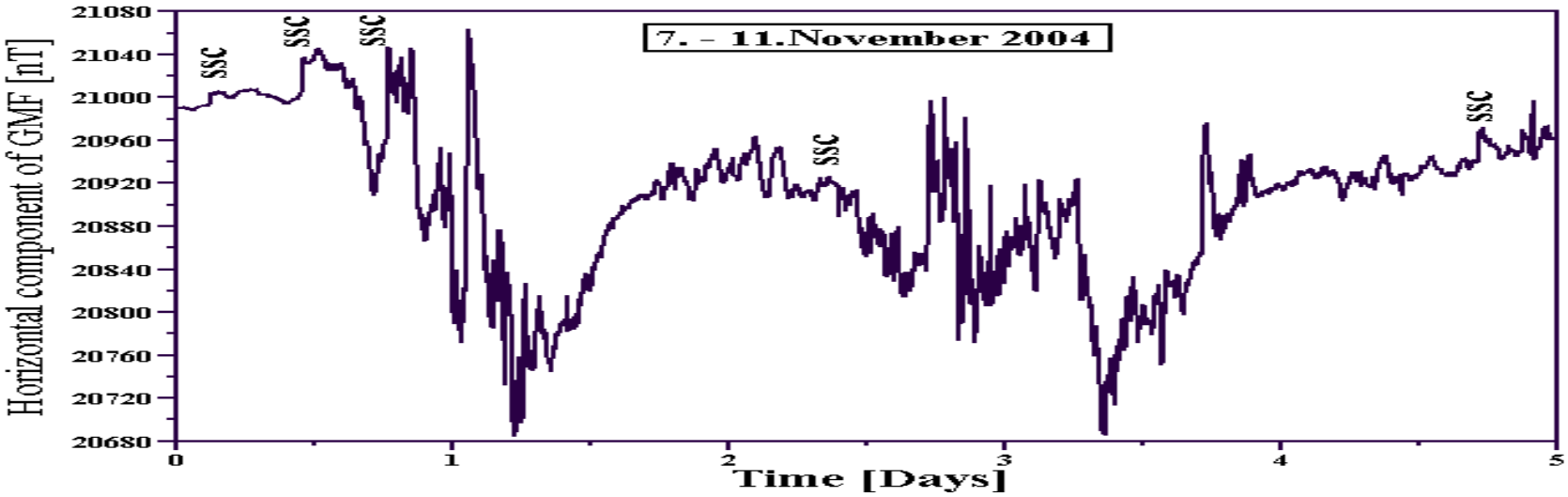
- Other indexes: A, Ap, C, C9, ΣKp, AE, aa, PC, etc.

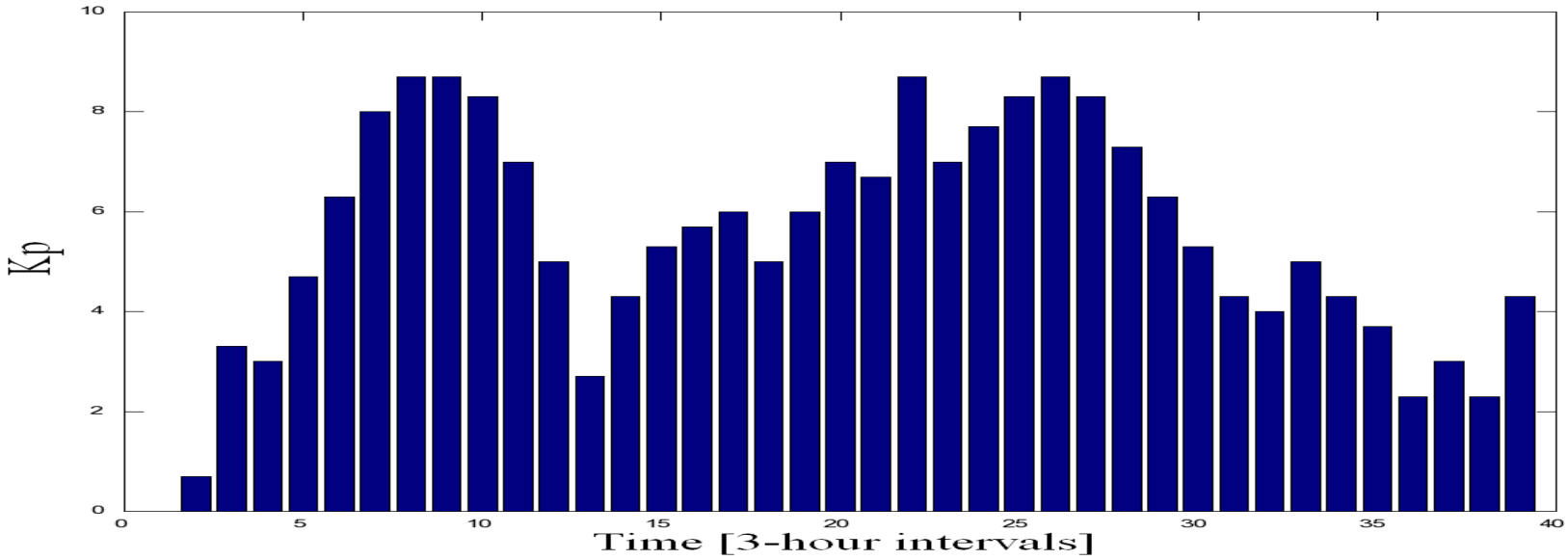
# Kp during a geomagnetic storm



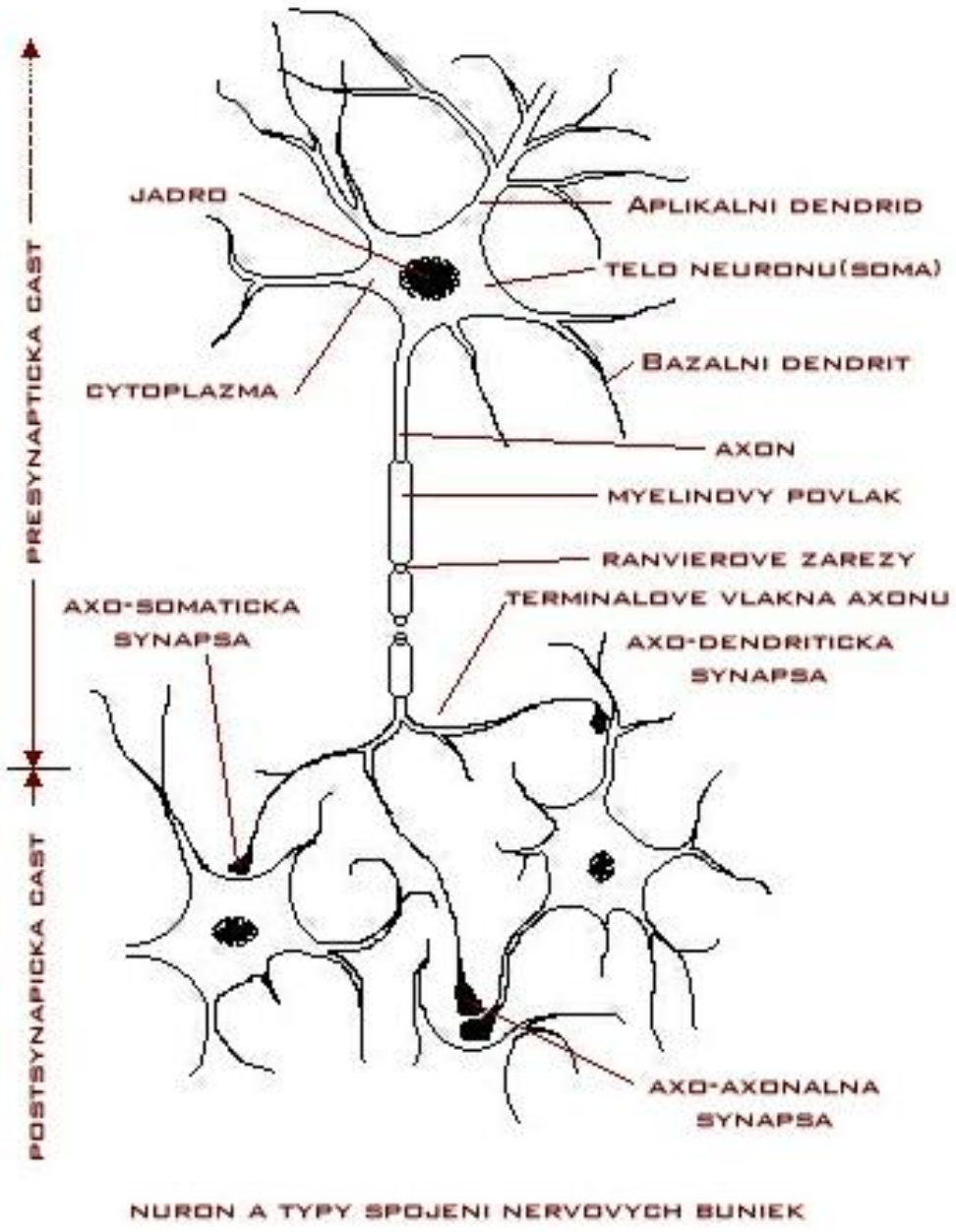
# Kp during a complicated geomagnetic storm



# Artificial neural network

(A tool fol space weather modelling)

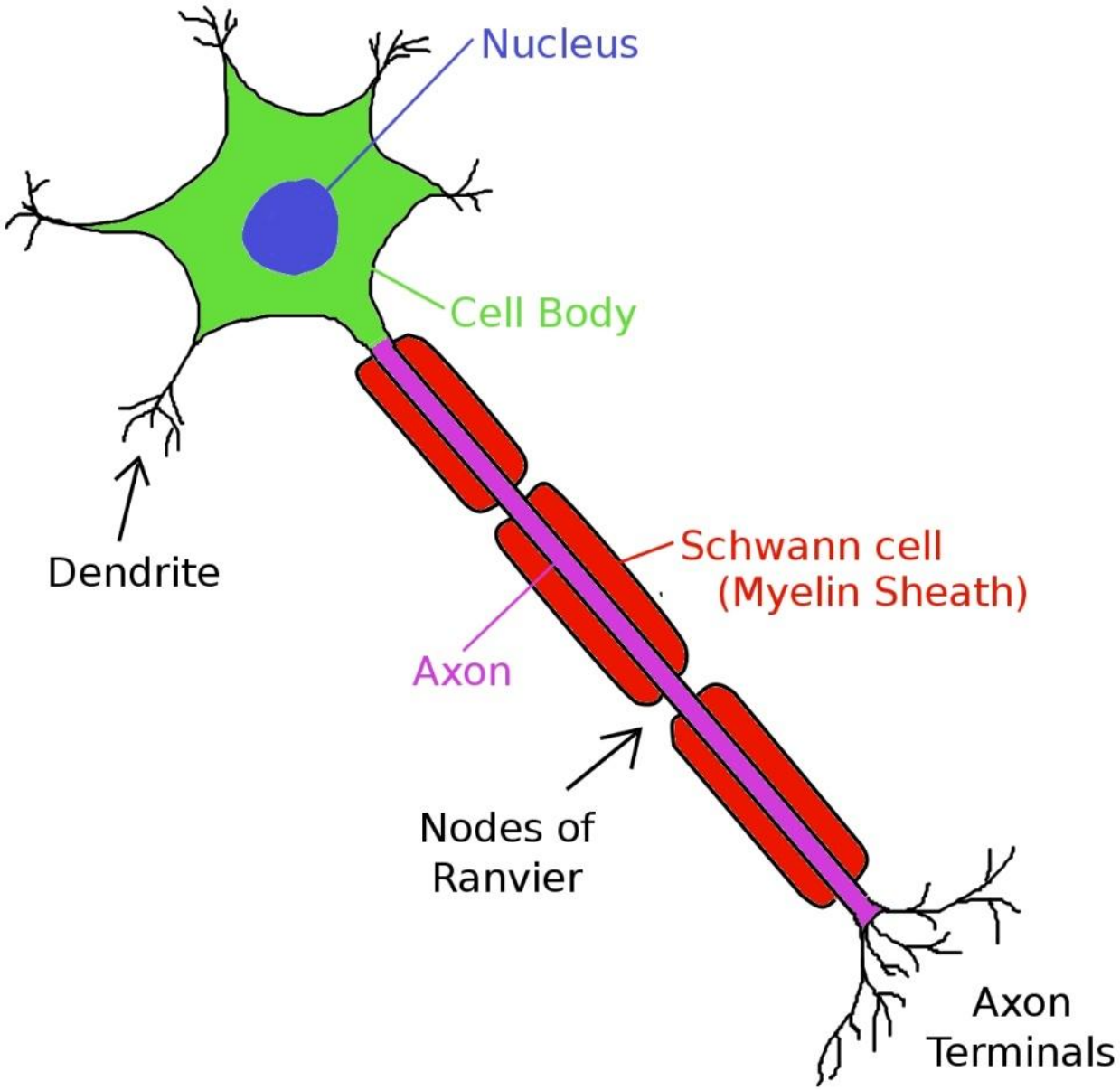



NURON A TYPY SPOJENI NERVOVYCH BUNIEK

A neural network is an interconnected assembly of simple processing units, whose functionality is loosely based on the animal neuron.

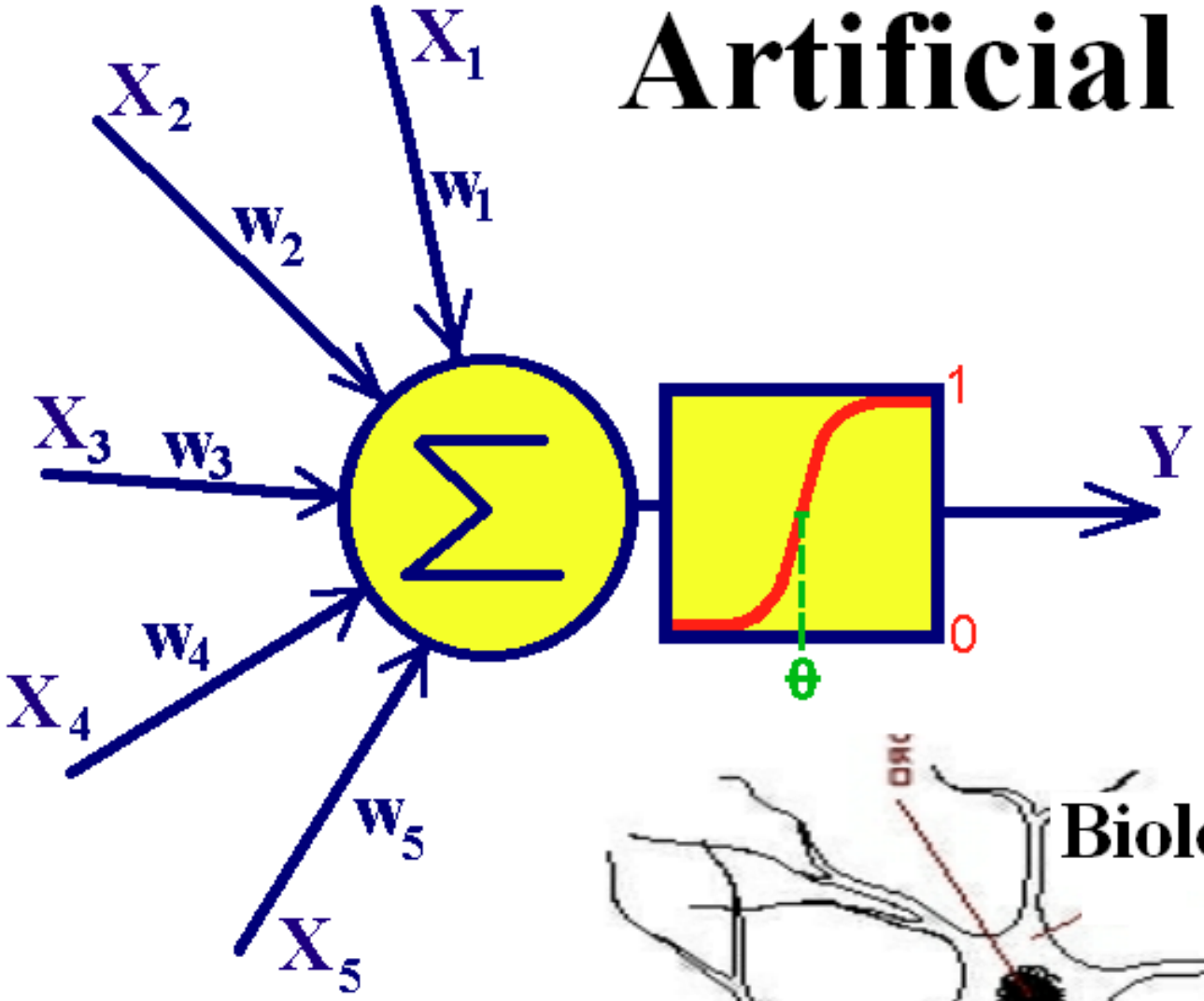
Biological neurons

# Diagram of neuron

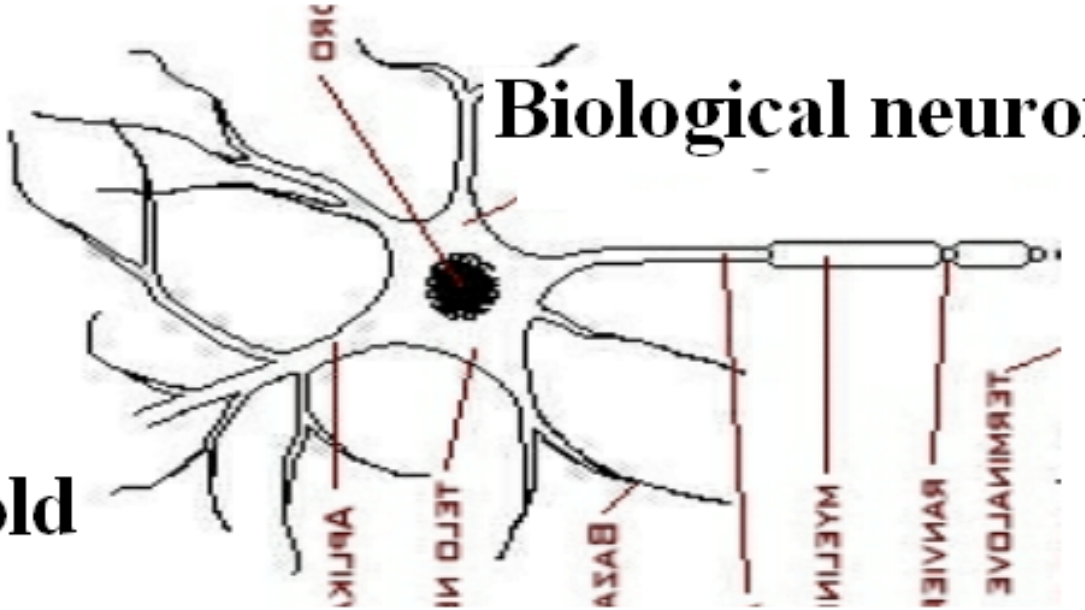



Reference:
http://en.wikipedia.org/wiki/File:Neuron1.jpg
[cited: 2011-08-13]

# Artificial neuron

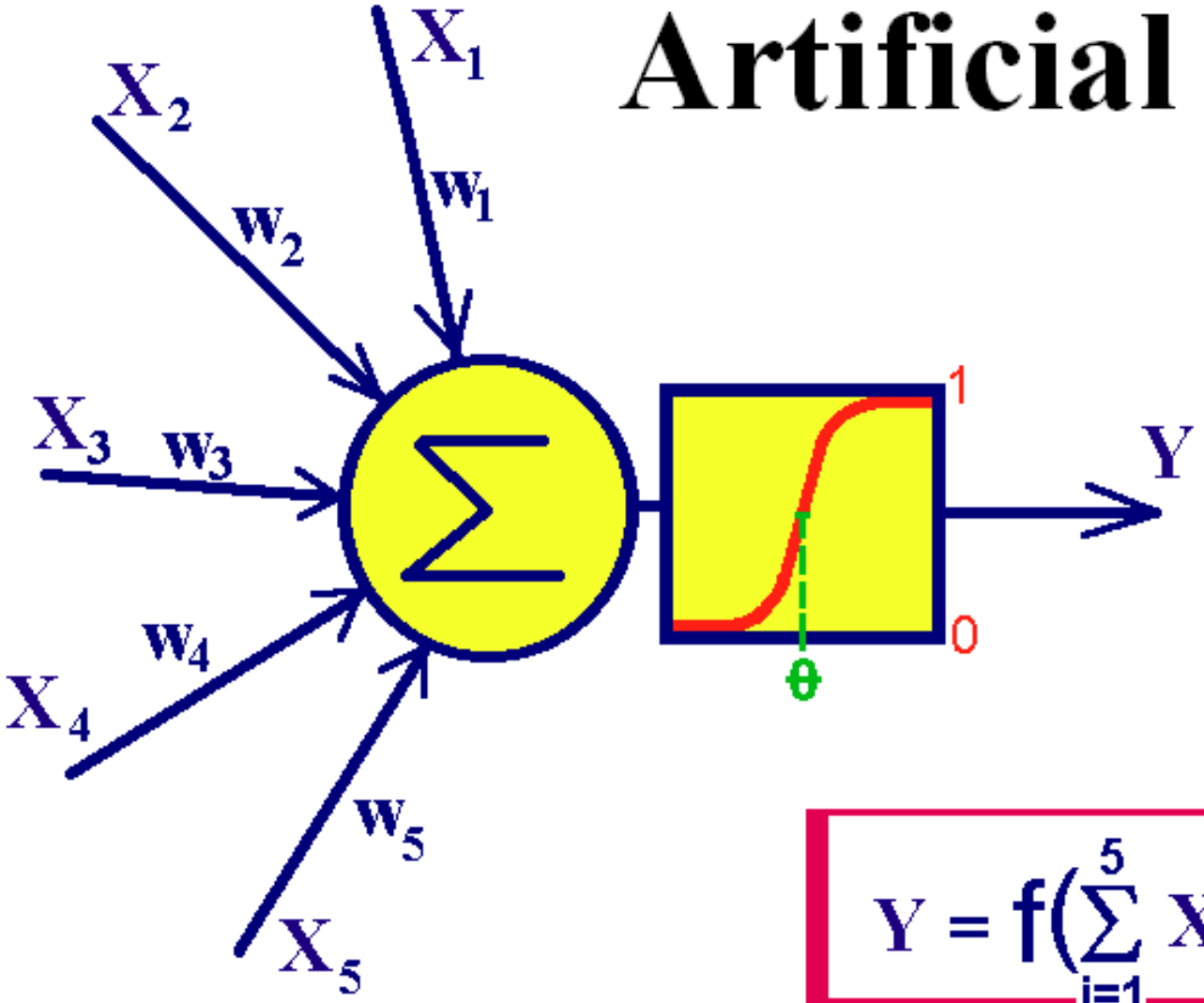



## Biological neuron



$X_1 \dots X_5$ - inputs
$w_1 \dots w_5$ - weights
$\theta$ - sensitivity threshold
Y - output

# Artificial neuron

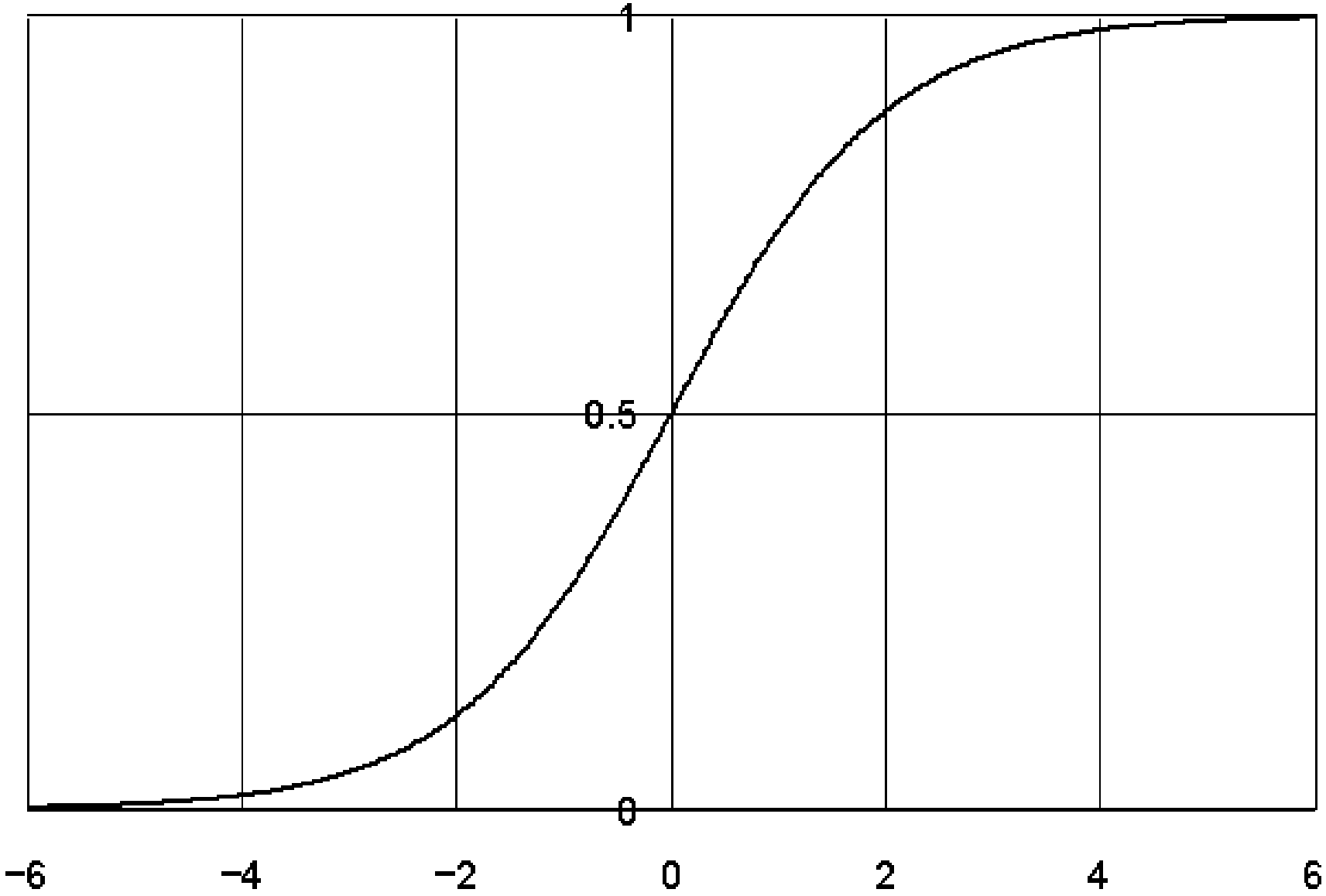



$$Y = f\left(\sum_{i=1}^{5} X_i w_i - \theta\right)$$

$X_1 \dots X_5$ - inputs
$w_1 \dots w_5$ - weights
$\theta$ - sensitivity threshold
$Y$ - output

$$f(z) = \frac{1}{1 + \exp(-z)}$$

# Activation function (Logistic sigmoid function)



# A logistic regression model for predicting the occurrence of intense geomagnetic storms

Author: Nandita Srivastava

In: Annales Geophysicae, 23, 2969-2974, 2005

# Coded values of dependent and independent values of the logistic regression model (Nandita Srivastava, 2005)

| | | Encoding | | |
|---|---|---|---|---|
| Names of variables | Type of variable | Measured Parameter | Value | Code |
| $D_{st}$ index | Binary and dependent | $D_{st}$ | $-200$ to $-100$ nT | 0 |
| | | | $<-200$ nT | 1 |
| Halos | Binary and independent | Full halos | 360° angular span | 1 |
| | | Partial | $>140°$ angular span | 0 |
| | | None | | $-1$ |
| Location | Binary and independent | Location-bin | Within $\pm 40°$ latitude $\pm 40°$ longitude | 1 |
| | | | Outside $\pm 40°$ latitude $\pm 40°$ longitude | 0 |
| Association with other activity | Binary and independent | Flare-bin | Flares | 1 |
| | | | EPs | 0 |
| Initial Speeds | Numeric and independent | $V_i$ | Value in km $s^{-1}$ | – |
| Southward IMF | Numeric and independent | $B_z$ | Value in nT | – |
| Total IMF | Numeric and independent | $B_T$ | Value in nT | – |
| Ram pressure | Numeric and independent | $P_R$ | Value in dynes $cm^{-2}$ | – |

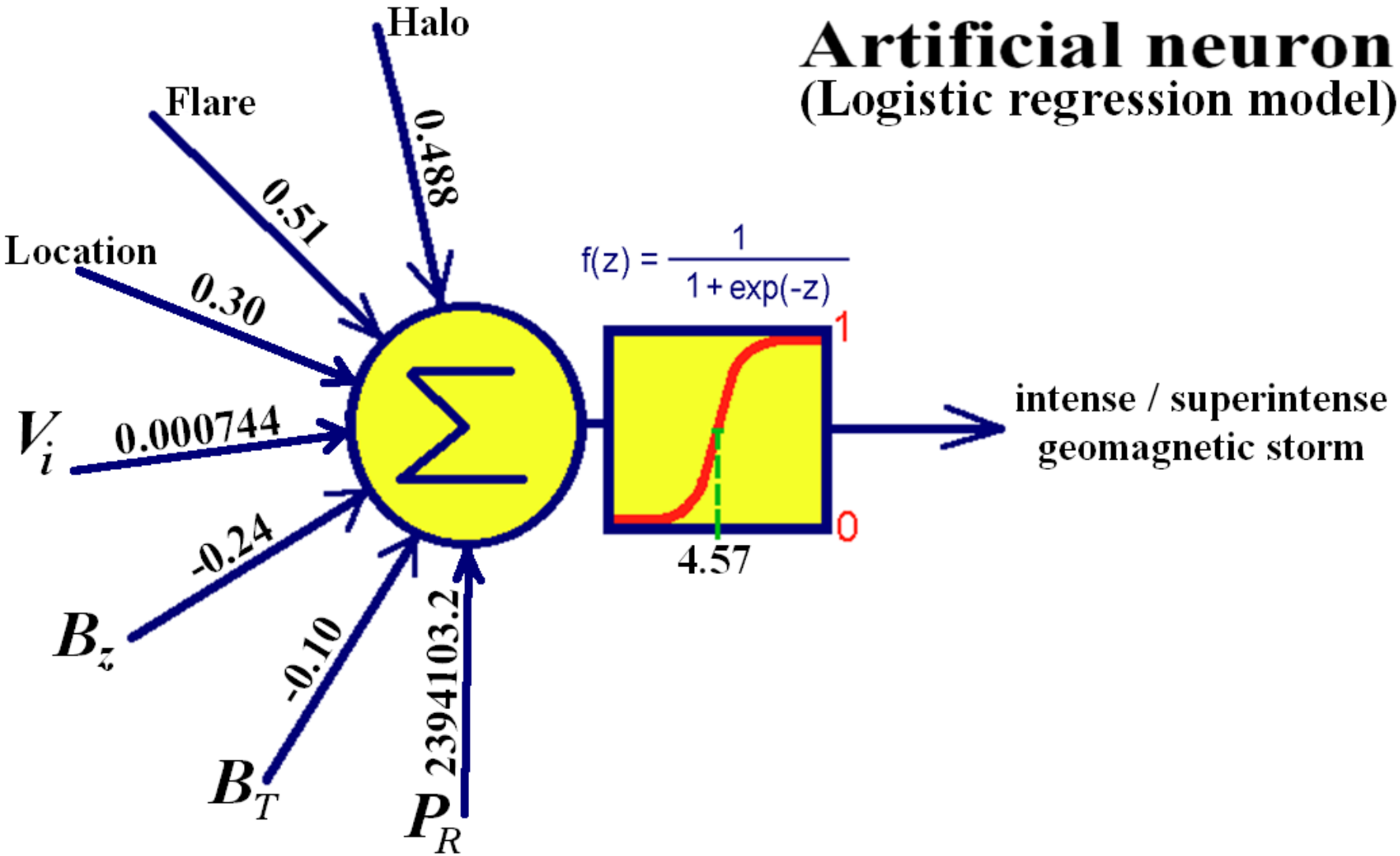
# Logistic regression model for classification Dst index to be [-200 nT, -100 nT] (intense storm) or to be <-200 nT (superintense storm)

$$P = \frac{1}{(1 + exp(-Z))},$$

where

$$\begin{aligned} Z = (&-4.57 + 0.488 \times \text{Halo}-\text{bin} + 0.51 \times \text{Flare}-\text{bin} \\ &+0.30 \times \text{Location}-\text{bin} + 7.44 \times \text{E}-04 \times \text{V}_\text{i} \\ &-0.24 \times B_z - 0.10 \times B_T + 2394103.2 \times P_R). \end{aligned}$$

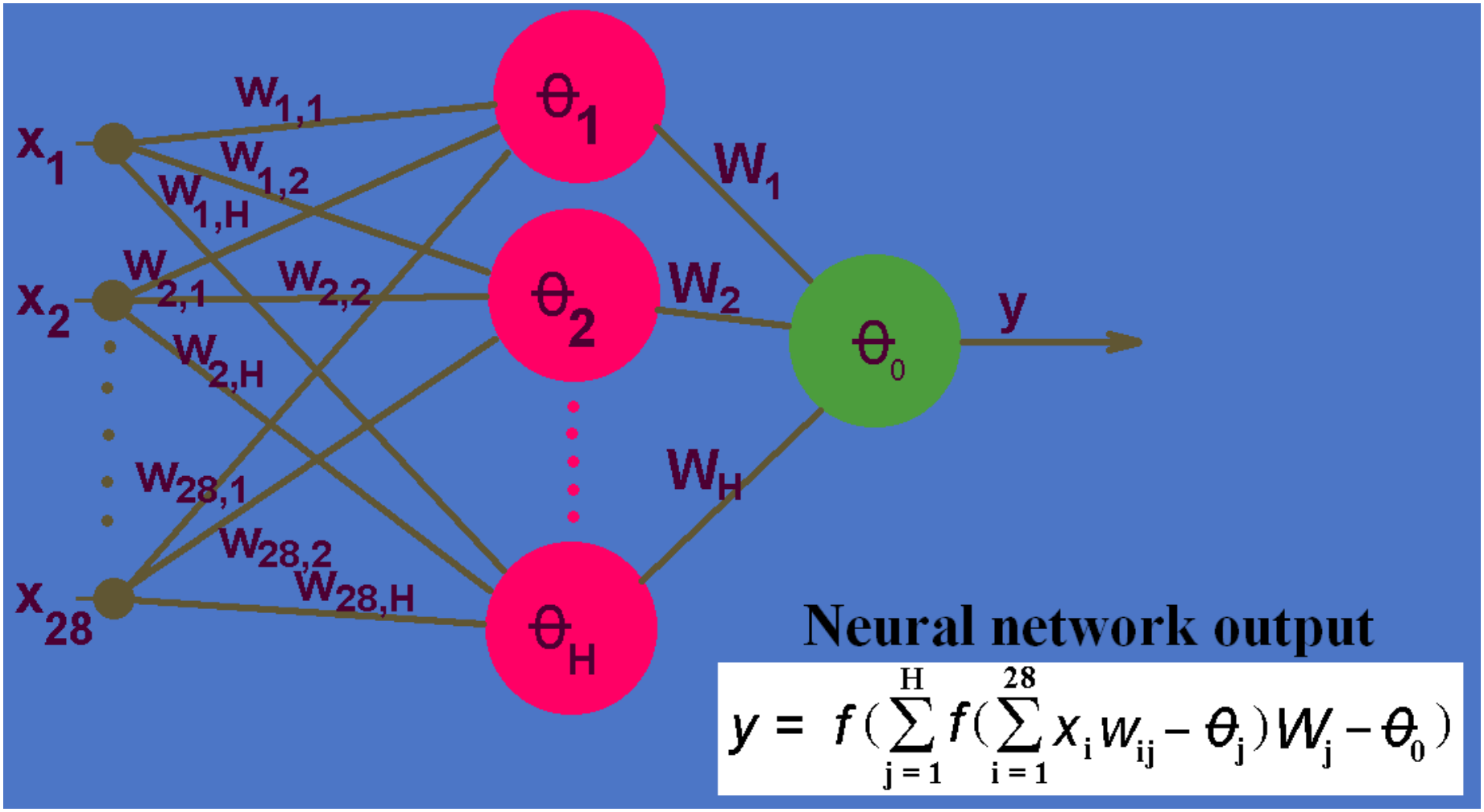
# Logistic regression model for classification Dst index to be [-200 nT, -100 nT] (intense storm) or to be <-200 nT (superintense storm)



# Validation of the logistic regression model

| Data-sets | | Observed | Predicted | % Correct prediction |
|---|---|---|---|---|
| Training | Super-Intense | 16 | 10 | 62.5% |
| | Intense | 30 | 29 | 97% |
| | Total | 46 | 39 | 85% |
| Validation | Super-intense | 4 | 2 | 50% |
| | Intense | 5 | 5 | 100% |
| | Total | 9 | 7 | 77.7% |

# Neural network with a layer of "hidden" neurons



Neural netwok can be used as a universal approximation tool

# UNIVERSAL APPROXIMATION THEOREM

# Assumptions:

- $F$ is a continuous function which is transforming n-dimensional space to the open interval (0, 1)

  $F$: $R^n \rightarrow (0, 1)$

  Where $y = F(\boldsymbol{x}) = F(x_1, x_2, \ldots, x_n)$.
- Training set $A_{train}$ contains $r$ points from n-dimensional space $R^n$

  $A_{train} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_r\}$
- $f$: $R \rightarrow (0, 1)$ is a continuous and monotonically-increasing function which satisfies the asymptotic conditions $f(-\infty) = 0$ and $f(\infty) = 1$.

# Statement:

For arbitrary $\varepsilon$ , there is a function

$$G(\vec{x}) = f(\Sigma_{i=1}^{q} \alpha_i f(\theta_i + \vec{w}_i.\vec{x}))$$

satisfying

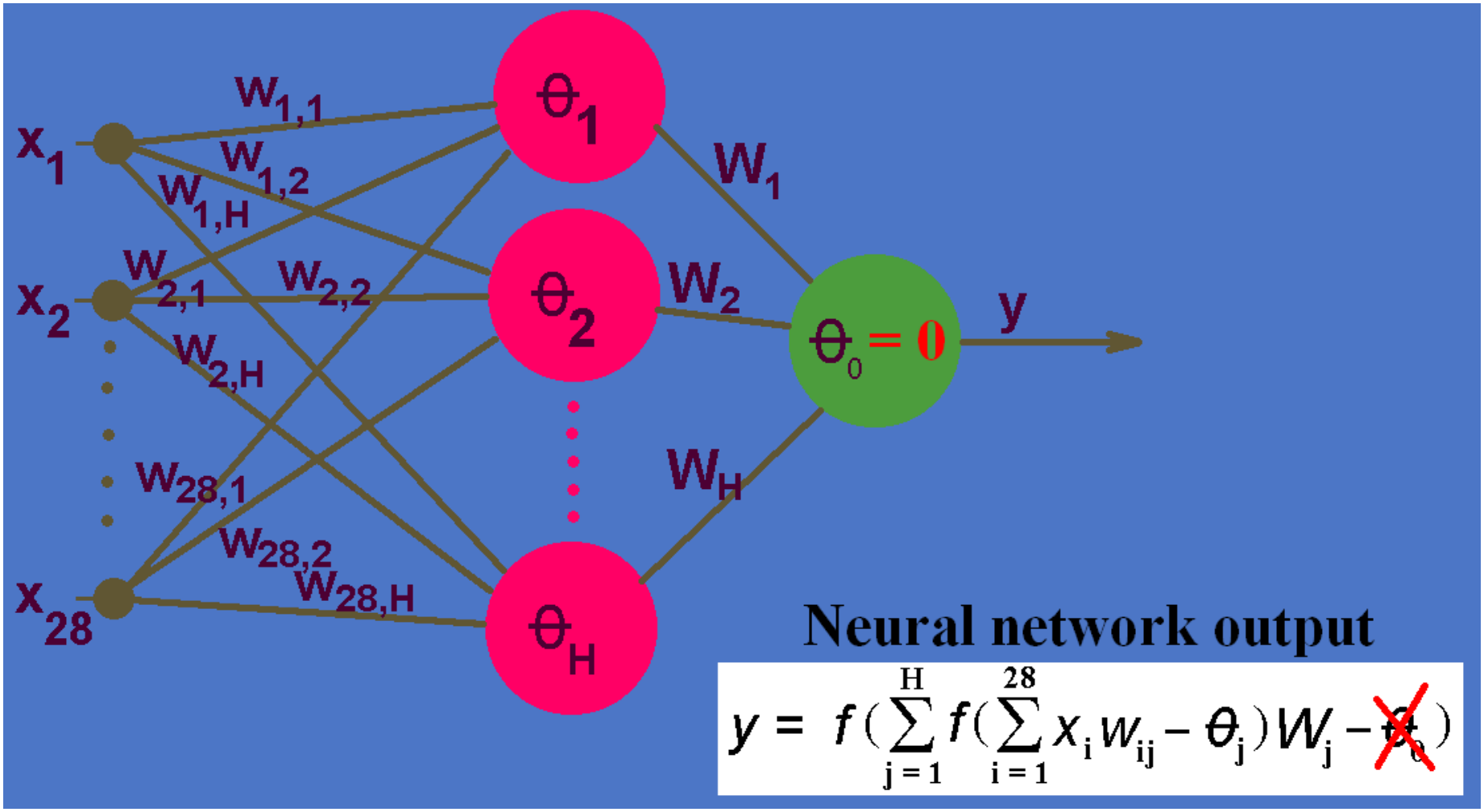
$$\Sigma_{k=1}^{r} |F(\vec{x_k}) - G(\vec{x_k})| < \varepsilon \ .$$

Here $\alpha_i$, $\theta_i$ are real coefficients and $\vec{w}_i$ are the vectors containing $n$ real components.

# From the theorem follows that:

- Having a sufficient number of the hidden neurons, the neural network is able to approximate any function which is defined by the set of training patterns.

# Neural network in accordance with the universal approximation theorem

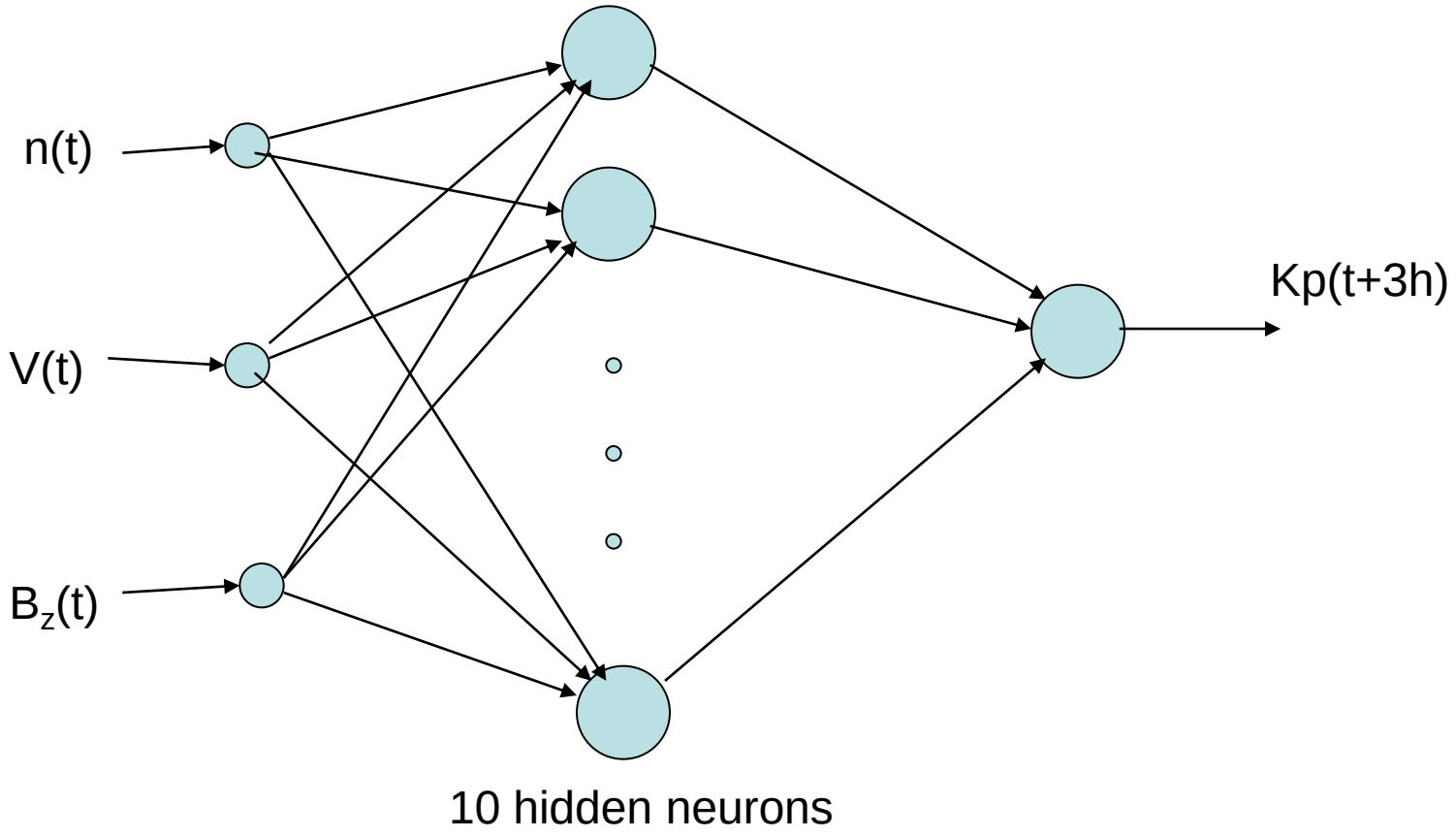


# Real time Kp predictions from solar wind data using artificial neural networks

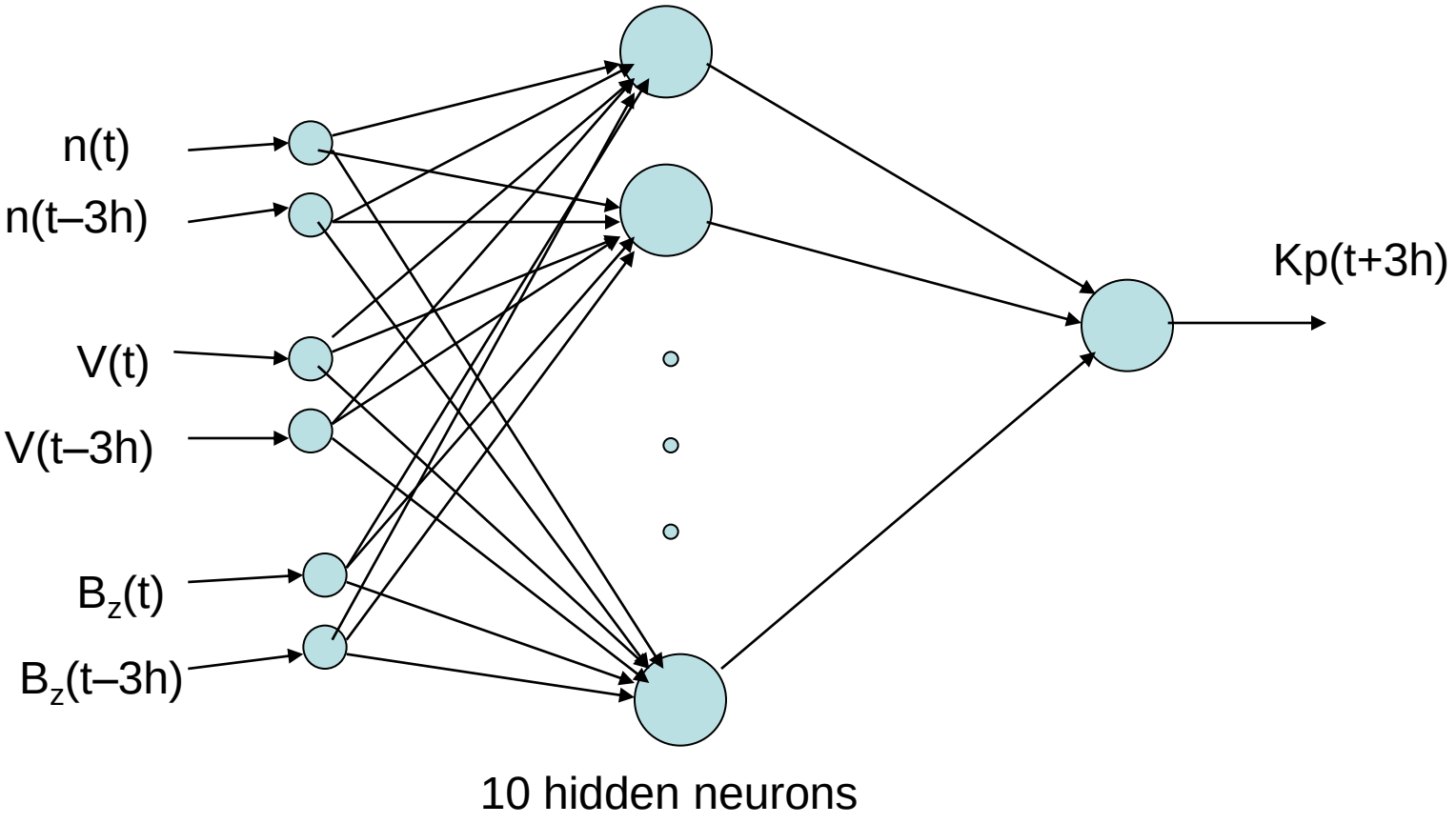
Authors: Fredrik Boberg, Peter Wintoft, and Henrik Lundstedt

Phys. Chem. Earth (C), Vol. 25, No. 4, pp. 275-280, 2000

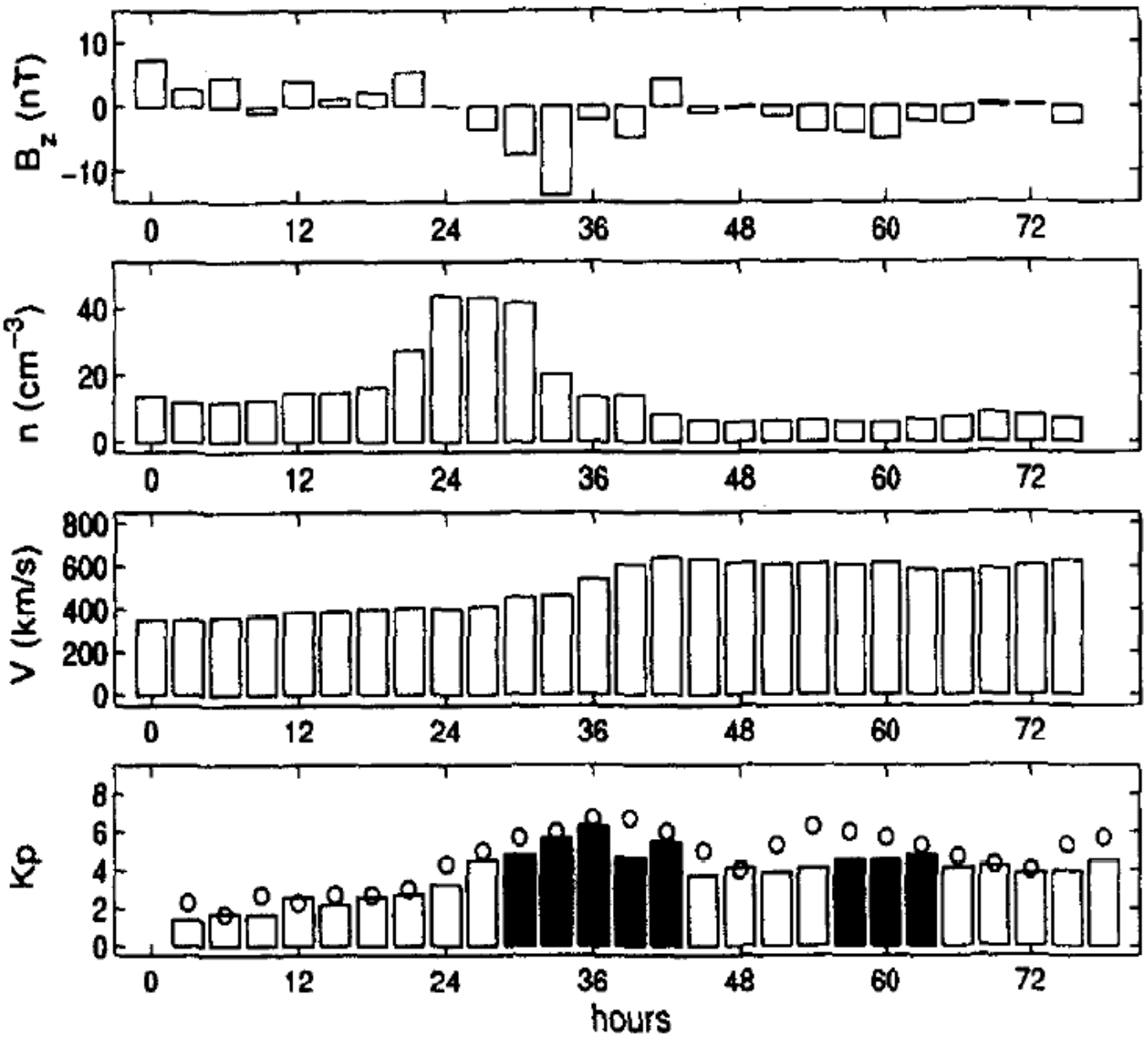
An expert network specialized in making one-step-ahead predictions of Kp index during geomagnetically quiet periods.



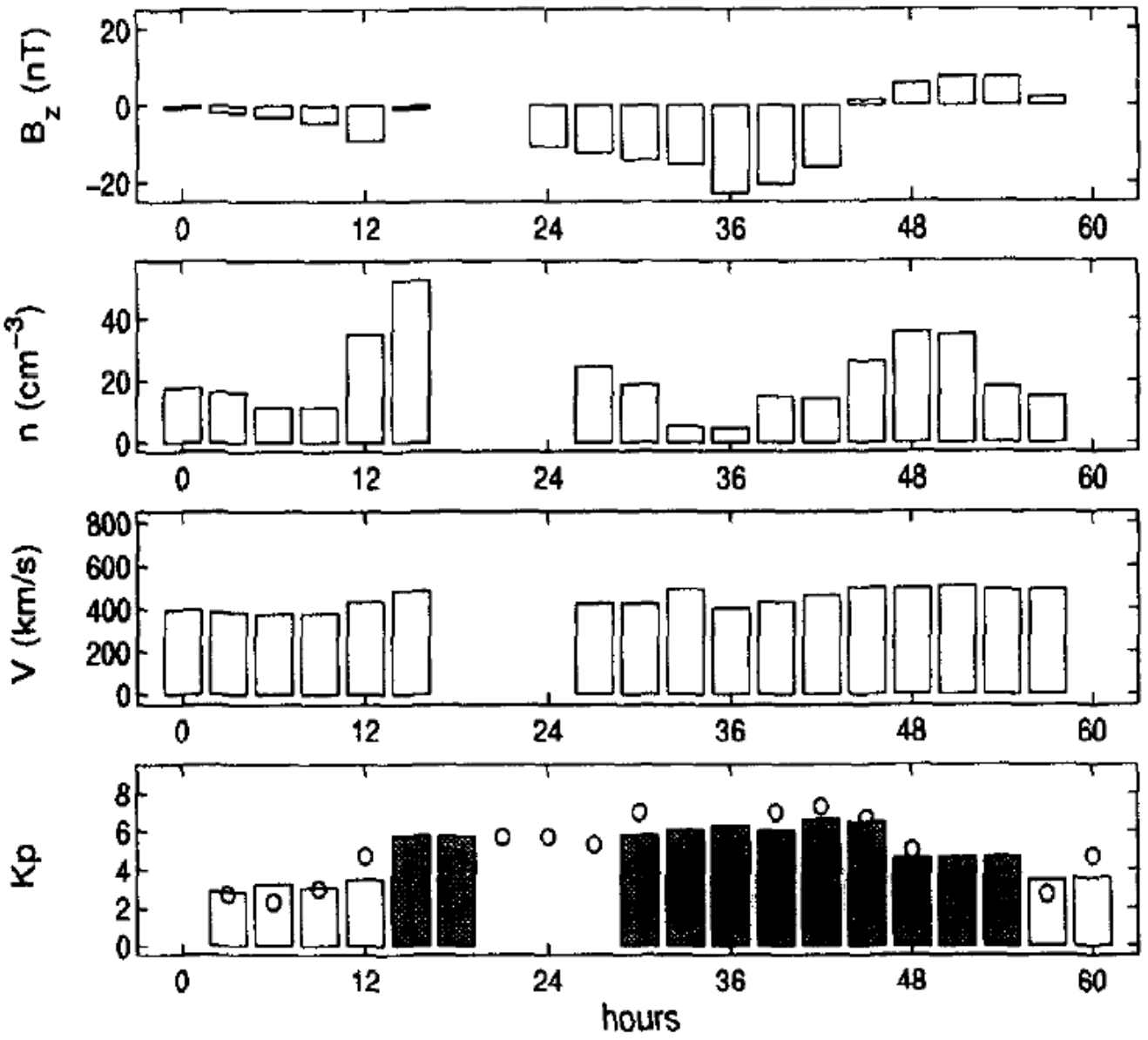
An expert network specialized in making one-step-ahead predictions of Kp index during geomagnetic storm periods.



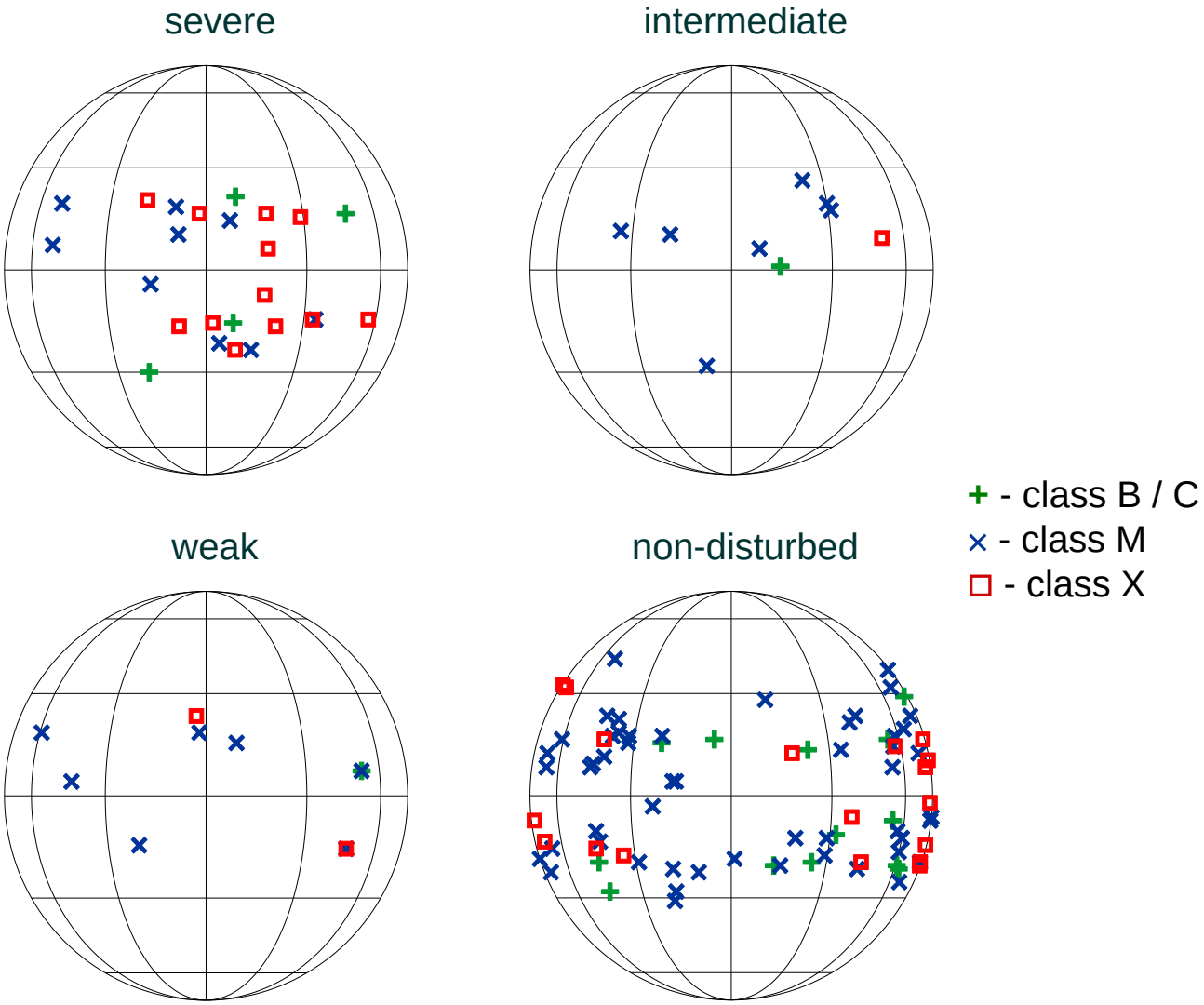
# Fig.1: Time series of solar-wind parameters, predicted Kp's as well as observed Kp's



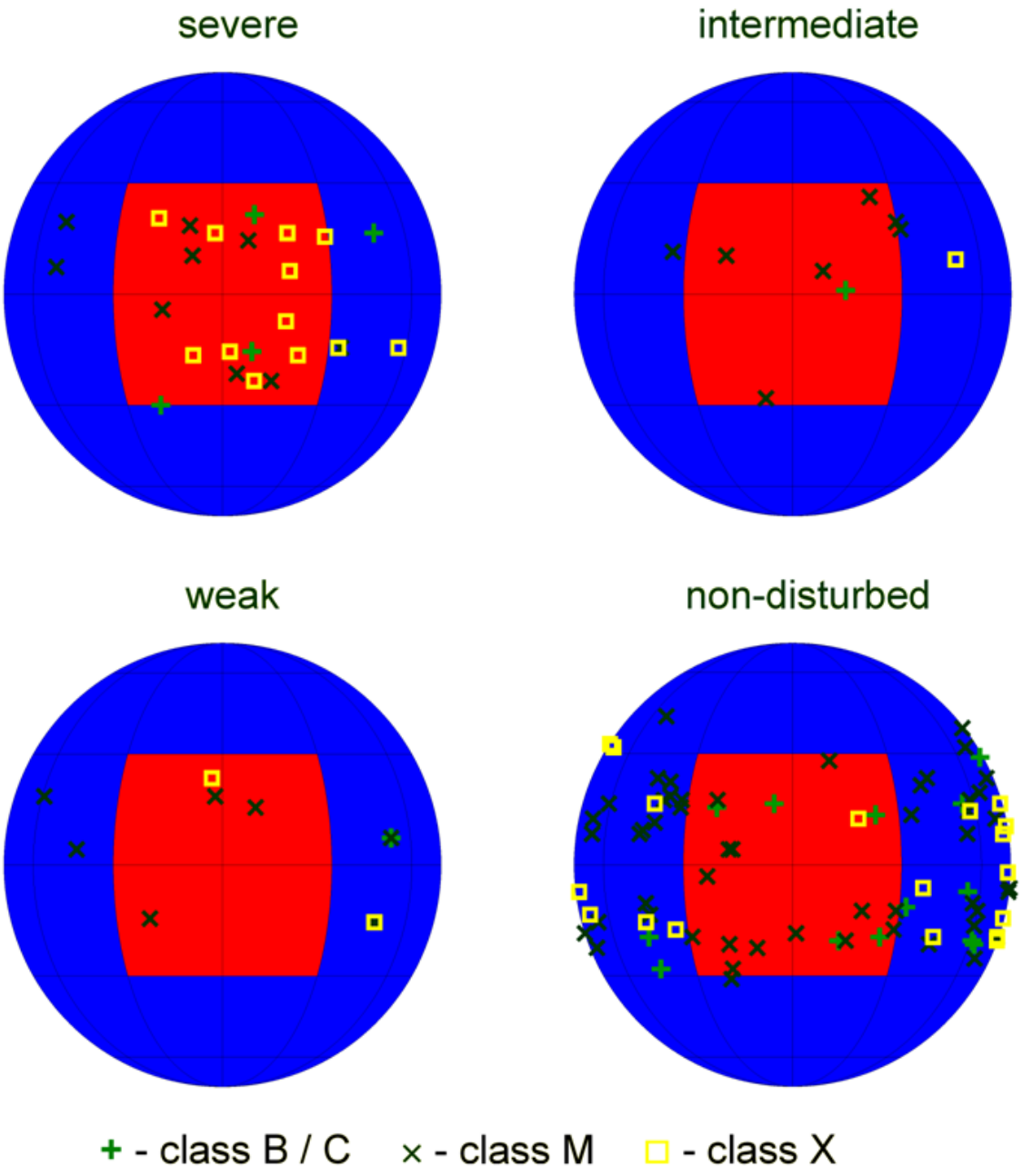
# Fig.2: Time series of solar-wind parameters, predicted Kp's as well as observed Kp's



# Geoeffectiveness of some solar events modelled using artificial neural network




Scatter graph of X-ray events associated with sweep-frequency radio bursts (RSP) of type II (shock wave) and type IV (plasmoid) observed on the solar disc in the period 1996−2004, classified according to the level of their geoeffective-ness.




Geoeffective X-ray flares were those which were accompanied with RSP II/IV and they occurred close to the centre of the solar disk.

# Reference

**SOLAR ENERGETIC EVENTS IN THE YEARS 1996-2004. THE ANALYSIS OF THEIR GEOEFFECTIVENESS**

J. BOCHNÍČEK[1], P. HEJDA[1], F. VALACH[2]

1 Institute of Geophysics AS CR., 141 31 Prague 4, Czech Republic (jboch@ig.cas.cz)

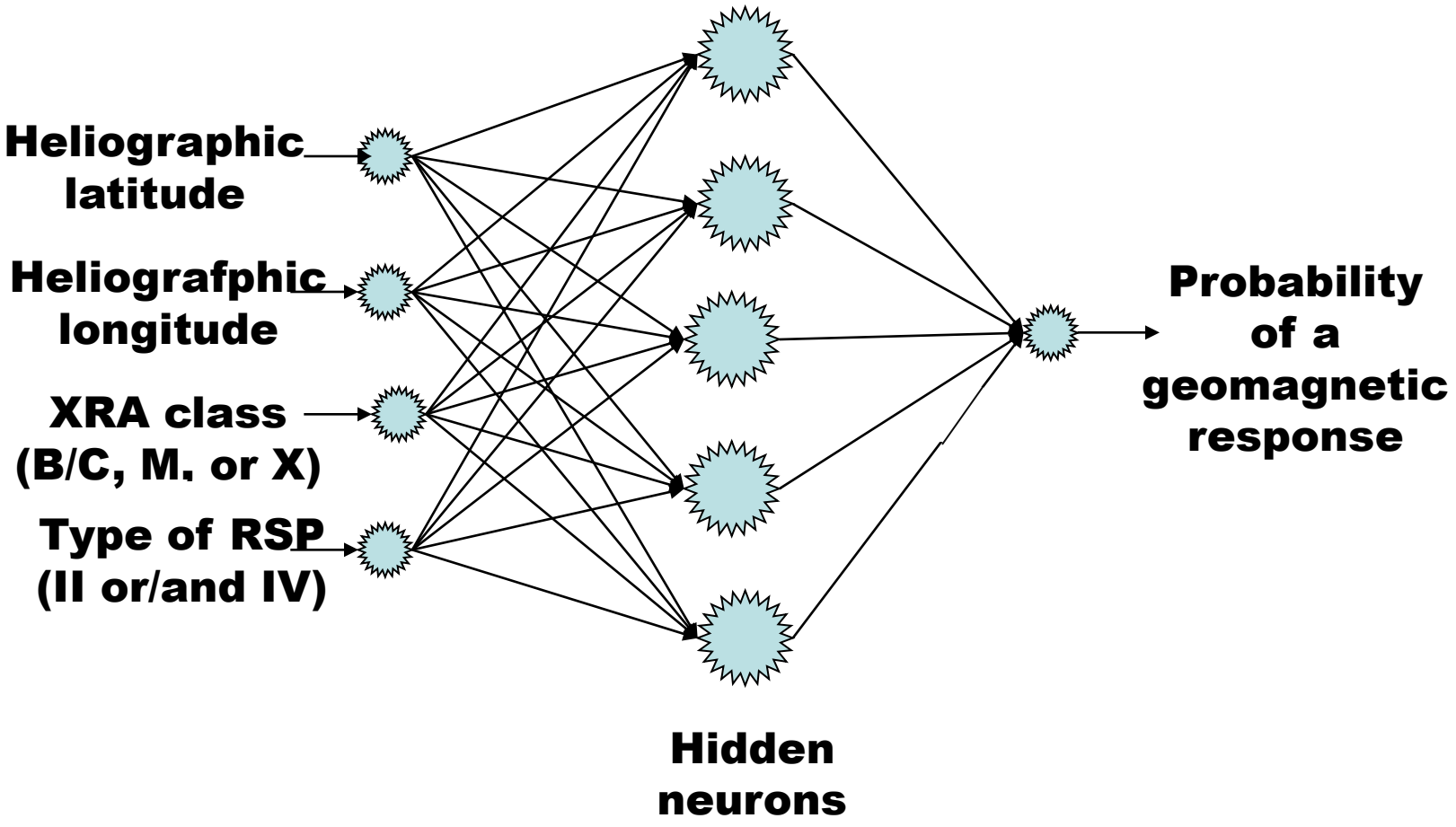
2 Geomagnetic Observatory, Geophysical Institute SAS, 947 01 Hurbanovo, Slovak Republic (fridrich@geomag.sk)

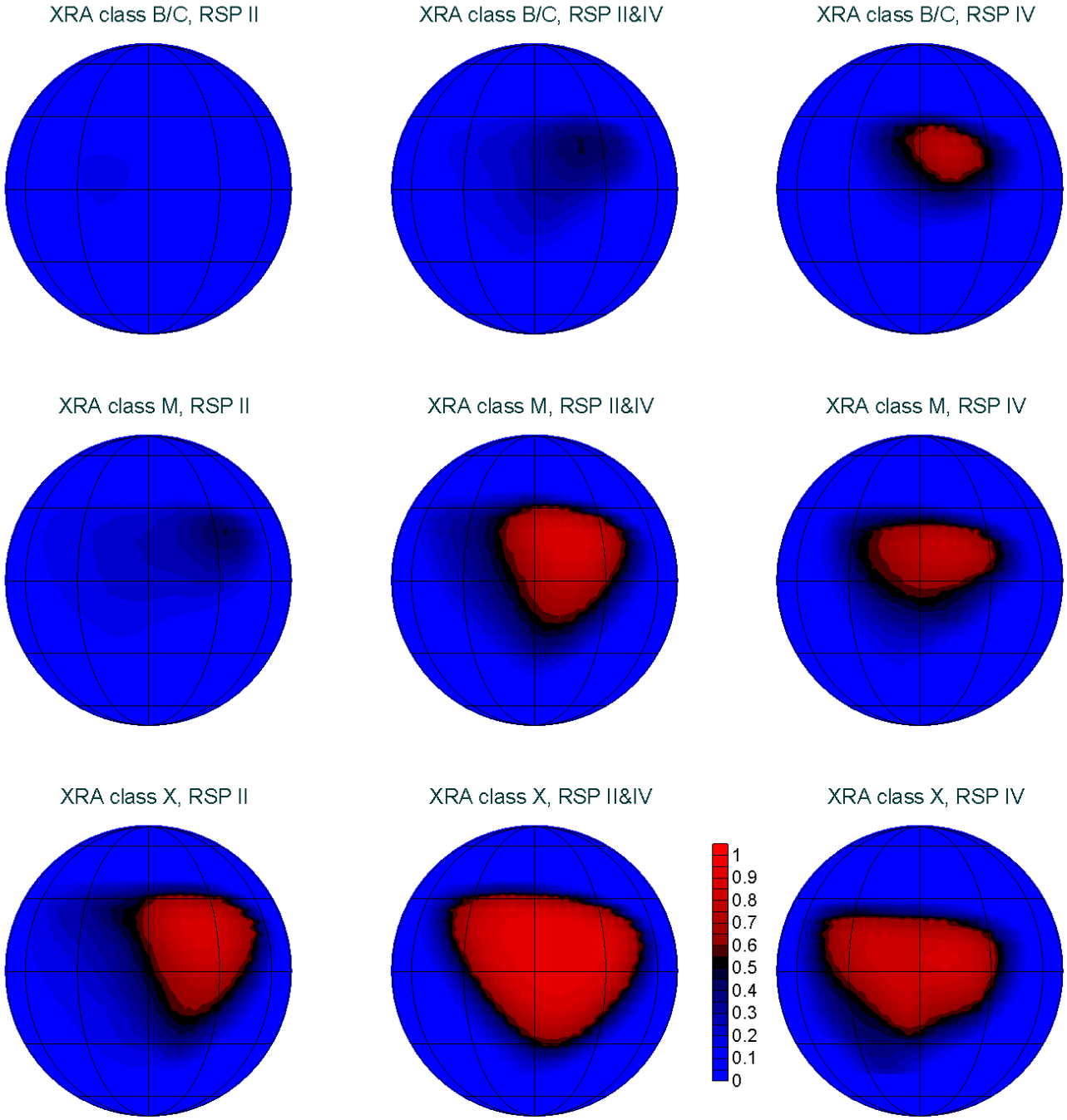
*Received: January 9, 2007; Revised: April 5, 2007; Accepted: May 11, 2007*

Stud. Geophys. Geod., 51 (2007), 439–447

© 2007 StudiaGeo s.r.o., Prague

# Geoeffectiveness of some solar events modelled using artificial neural network






Regions on the solar disc where the geoeffective XRA events occur.

<u>Published in:</u>

F. Valach, P. Hejda, J. Bochnníček,

GEOEFFECTIVENESS OF XRA EVENTS ASSOCIATED WITH RSP II AND/OR RSP IV ESTIMATED USING THE ARTIFICIAL NEURAL NETWORK,

*Stud. Geophys. Geod.*, 51 (2007), 551-562.

Forecasts of “after-the-fact” geomagnetic responses.
All classes of XRA accompanied by
RSP II and/or IV, observed in 1996-2004, are considered.

| Number of observed geomagnetic responses | Number of predicted geomagnetic responses | Number of false alerts |
|---|---|---|
| 93 | 37 (40 %) | 14 |

# “After-the-fact” forecasts of geomagnetic responses for events from the years 1996-2004, classified by RSP type.

| | Number of observed geomag. responses | Number of predicted geomag. responses | Number of false alerts |
|---|---|---|---|
| RSP II | 40 | 6 (15%) | 2 |
| RSP II & IV | 41 | 23 (56%) | 7 |
| RSP IV | 12 | 8 (67%) | 5 |

# “After-the-fact” forecasts of geomagnetic activity divided by XRA class and RSP type.

| Type of RSP | XRA class | Number of observed responses | Number of predicted responses | Number of false alerts |
|---|---|---|---|---|
| II | B/C | 10 | 0 (0 %) | 0 |
| II | M | 21 | 0 (0 %) | 0 |
| II | X | 9 | 6 (67 %) | 2 |
| II&IV | B/C | 4 | 0 (0 %) | 0 |
| II&IV | M | 22 | 10 (45 %) | 5 |
| II&IV | X | 15 | 13 (87 %) | 2 |
| IV | B/C | 2 | 1 (50 %) | 1 |
| IV | M | 7 | 4 (57 %) | 3 |
| IV | X | 3 | 3 (100 %) | 1 |

# Reference

**GEOEFFECTIVENESS OF XRA EVENTS ASSOCIATED WITH RSP II AND/OR RSP IV ESTIMATED USING THE ARTIFICIAL NEURAL NETWORK**

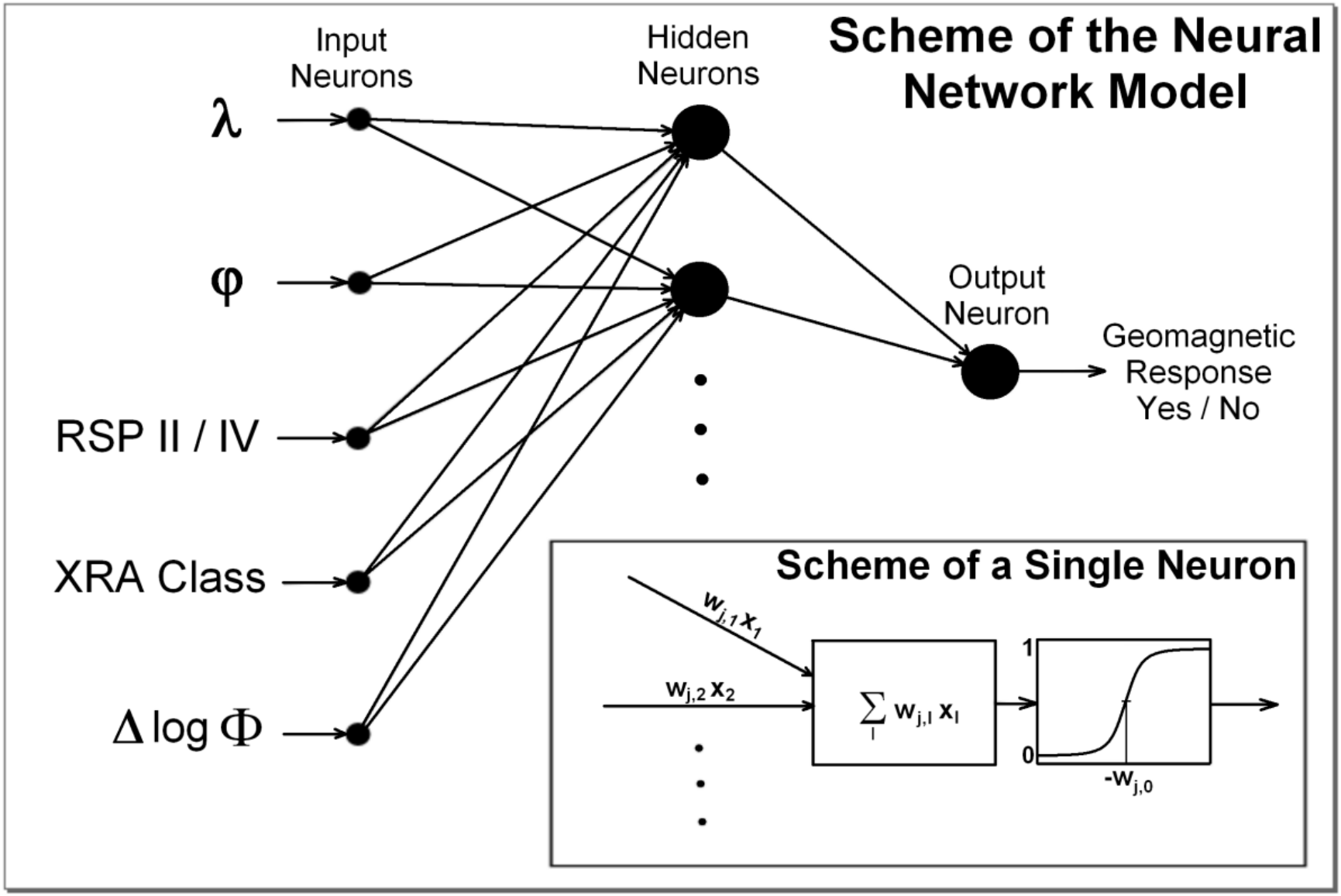
F. VALACH[1], P. HEJDA[2], J. BOCHNÍČEK[2]

1 Geomagnetic Observatory, Geophysical Institute SAS, 947 01 Hurbanovo, Slovak Republic (fridrich@geomag.sk)
2 Institute of Geophysics AS CR, Boční II/1401, 141 31 Prague 4, Czech Republic (ph@ig.cas.cz, jboch@ig.cas.cz)

*Received: January 9, 2007; Revised: July 11, 2007; Accepted: August 9, 2007*

Stud. Geophys. Geod., 51 (2007), 551–562
© 2007 StudiaGeo s.r.o., Prague

We added information about SEPs (HEPF > 10 MeV) to the geomagnetic activity forecasting scheme (beside information on solar flares). This improved the forecasts.



# Reference

SPACE WEATHER, VOL. 7, S04004, doi:10.1029/2008SW000421, 2009

Click Here for Full Article

## Solar energetic particle flux enhancement as a predictor of geomagnetic activity in a neural network-based model

F. Valach,[1] M. Revallo,[2] J. Bochníček,[3] and P. Hejda[3]

Received 26 June 2008; revised 31 October 2008; accepted 22 December 2008; published 16 April 2009.

# Forecasts of SEP events

Using Dynamic Networks

# Input parameters (Day-by-day data):

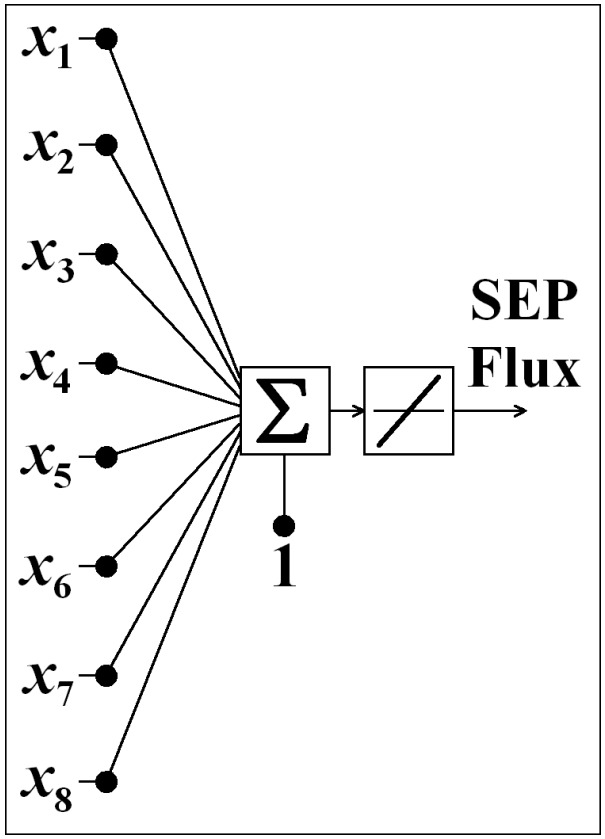
- X-ray flares:
  - The **class** of the most significant **XRA** (only those originated near the centre of the solar disk ± 40º)
  - **RSP** type II and/or type IV
- Full or partial halo **CMEs**:
  - **Linear speed** of CME (of the most significant CME)
  - **Angular width** of CME (of the most significant CME)
  - **Position angle** of CME (of the most significant CME)

# Variables in the neural network model (Lin. filter / Layer recurrent network):

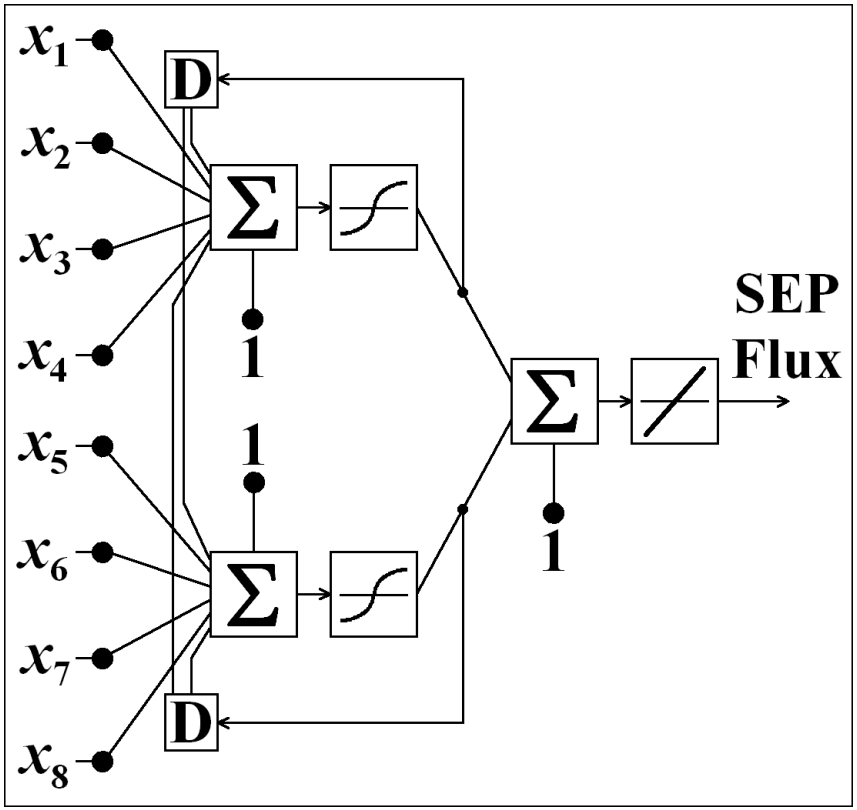
| Input variables | | Output variable |
|---|---|---|
| **Information about full or partial halo CMEs** | **Information about X-ray flares (XRAs)** | |
| **1. Position angle of the most important CME (the one with the greater width was considered to be more important).** | **5. The class of the most important X-ray flare (XRA Class) observed close to the centre of the solar disk (±40°).** | **Fluxes of protons with the energies exceeding 10 MeV measured in the libration point L1.** |
| **2. The greatest width of the CME which was observed this particular day.** | **6. The information about type II radio burst (RPS II) accompanied the X-ray flare.** | |
| **3. The linear speed of the most important CME** | **7. The same for type IV radio burst (RSP IV).** | |
| **4. The number of CMEs which were observed during the given day.** | **8. The number of X-ray flares which were observed close to the centre of the solar disk.** | |

# Neural networks used for SEP modelling

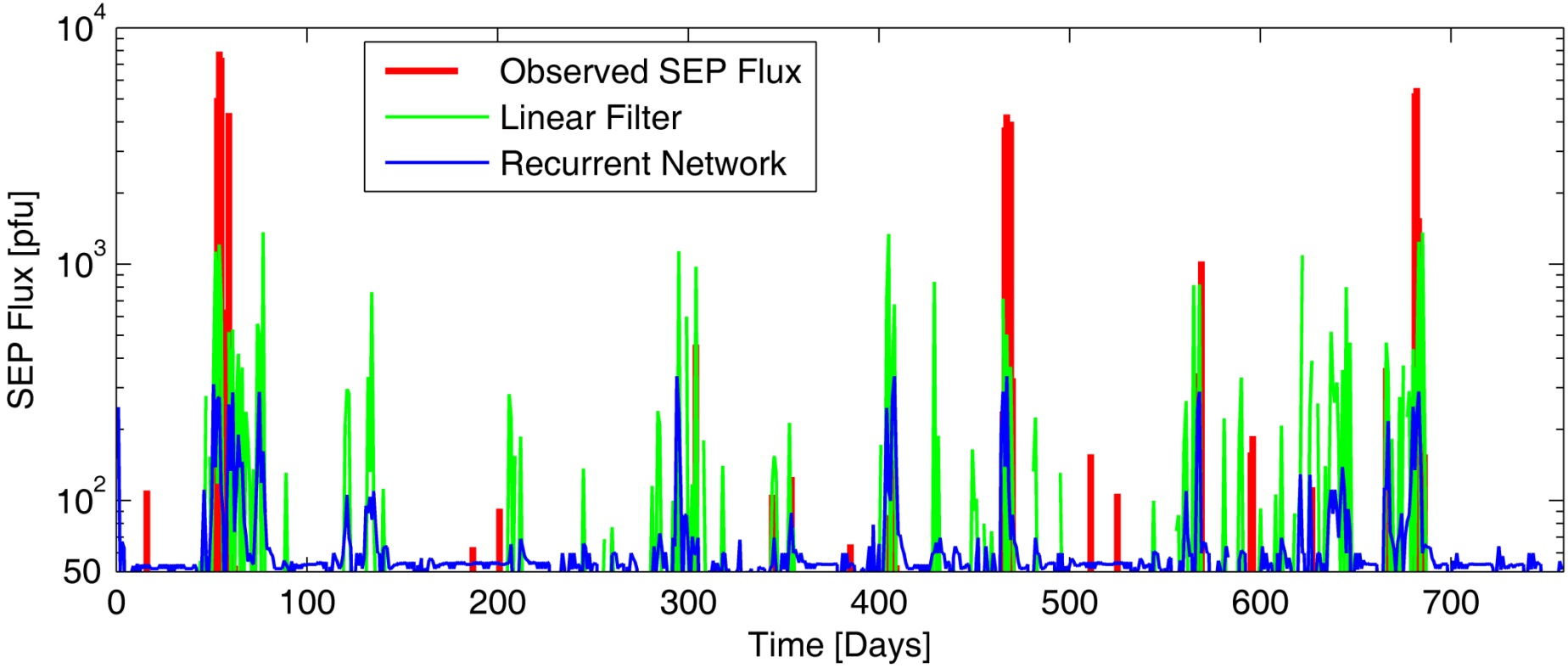
**Linear filter**

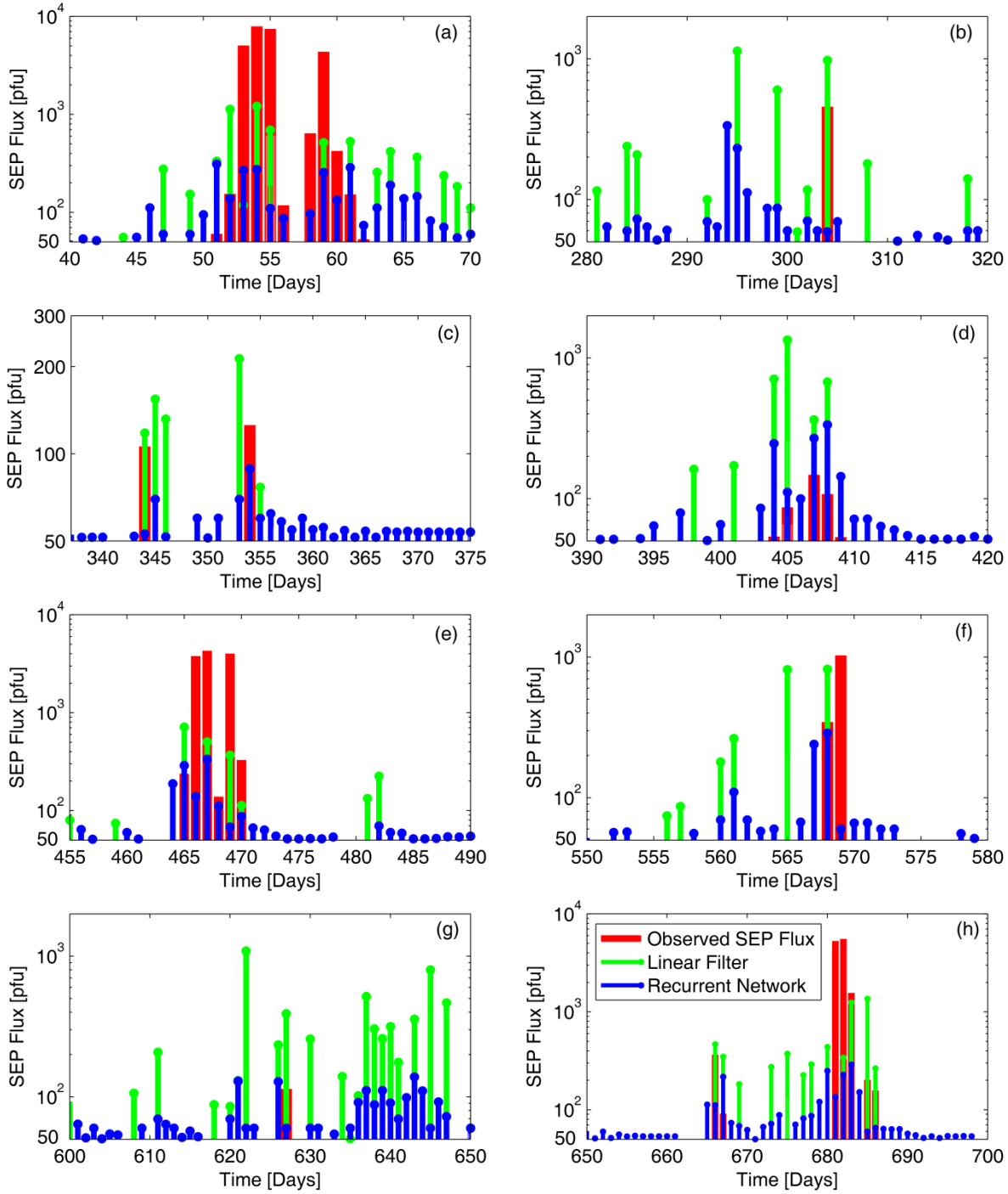



**Layer recurrent network**



# Observed and forecast fluxes of SEP during test period 13/08/2003 – 26/11/2005






Observed and forecast fluxes of SEP

---

Some details for the most interesting parts of the time series of the **test**

# Reference

**Acta Astronautica**
**Article in Press,** Corrected Proof - Note to users

doi:10.1016/j.actaastro.2011.06.003 | How to Cite or Link Using DOI
Permissions & Reprints

## Predictions of SEP events by means of a linear filter and layer-recurrent neural network

**Fridrich Valach**[a], **Miloš Revallo**[b], **Pavel Hejda**[c], and **Josef Bochníček**[c]

[a] Geomagnetic Observatory, Geophysical Institute, Slovak Academy of Sciences, Komárňanská 108, 947 01 Hurbanovo, Slovakia

[b] Geophysical Institute, Slovak Academy of Sciences, Bratislava, Slovakia

[c] Institute of Geophysics, Academy of Sciences of the Czech Republic, Prague, Czech Republic

Received 3 January 2011; revised 23 May 2011; accepted 1 June 2011. Available online 22 June 2011.

**Abstract**

Solar energetic particle (SEP) modelling has gained great interest in the community, specifically in connection with the safety of crews and the protection of technological systems of spacecraft situated outside the shielding of Earth's magnetosphere. Two models for the prediction of SEP events are presented in this paper. The models are based on a linear filter and on a special type of dynamic artificial neural network known as the layer-recurrent neural network. In this work they use as input the following parameters: the X-ray flare class for flares originating close to the centre of the solar disk; observed type II or IV radio bursts; and of the position angle, width, and linear speed of observed full or partial halo CMEs. The models are designed to provide forecasts of proton fluxes with energies exceeding 10 MeV at the L1 libration point.

**Highlights**

► The models based on a linear filter and a layer-recurrent neural network. ► Information on X-ray flares, radio bursts, and CMEs were used as input parameters. ► Forecasts of proton fluxes with energies exceeding 10 MeV at the L1 libration point. ► Practical recipes for alert announcements for critical proton fluxes were proposed.

**Keywords:** Coronal mass ejection; X-ray flare; Solar energetic particles; Artificial neural network

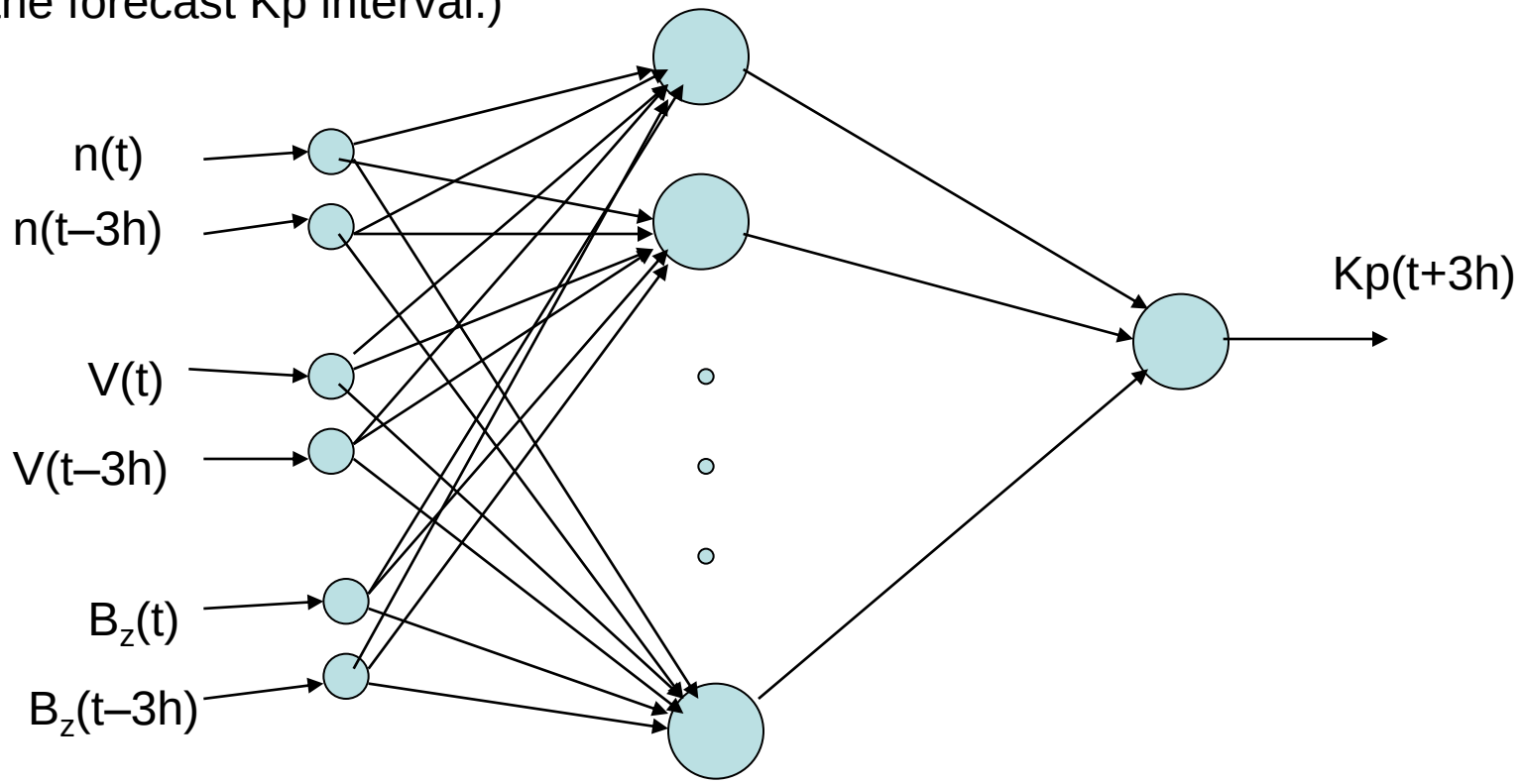
# Exercise

The task is:

Train the neural network to predict Kp index during a geomagnetically disturbed period from solar wind data.

You are asked to train an expert network specialized in making one-step-ahead predictions of Kp index during geomagnetic storm periods.

(Input parameters: The solar-wind parameters of the two 3-hour intervals prior to the forecast Kp interval.)




Unknown number of hidden neurons.

Note: This example is inspired by the paper of Boberg, Wintoft, and Lundstedt (2000).

# Data at your disposal:

- Quantities:
  - 3-hour mean values of solar wind parameters:
    - $B_z$ component of the IMF
    - Density of protons
    - Velocity of solar wind
  - Kp index
- Geomagnetic storms:

  May 1997, May 1998, June 1998, August 1998, September 1998, October 1998, November 2004

# Patterns for the neural network

- Neural networks learn from training patterns.
- A pattern has two parts:
  - Inputs that enter the NN at the same time.
  - Output which is expected to be obtained.
- Our database – files containing 7 columns:
    - Inputs: Bz(t-1), n(t-1), V(t-1), Bz(t), n(t), V(t)
    - Desired output (target): Kp(t+1)
  - Every row represents one training pattern.

Lister - [c:\Octave\Pracovny_adresar\NNs_UnivApprox\tra.dat]

Súbor Editácia Možnosti Pomocník 25 %

| | | | | | | |
|---|---|---|---|---|---|---|
| 3.22 | 17.40 | 361.80 | 0.25 | 12.20 | 382.07 | 1.7 |
| 0.25 | 12.20 | 382.07 | 2.03 | 7.55 | 399.89 | 1 |
| 2.03 | 7.55 | 399.89 | 1.73 | 7.92 | 409.55 | 2 |
| 1.73 | 7.92 | 409.55 | 3.78 | 8.23 | 433.98 | 1.3 |
| 3.78 | 8.23 | 433.98 | 2.51 | 8.24 | 450.40 | 1.7 |
| 2.51 | 8.24 | 450.40 | 4.61 | 5.57 | 471.31 | 4.3 |
| 4.61 | 5.57 | 471.31 | 6.86 | 14.31 | 559.69 | 3.3 |
| 6.86 | 14.31 | 559.69 | 9.48 | 11.72 | 571.40 | 5 |
| 9.48 | 11.72 | 571.40 | -0.83 | 8.93 | 636.97 | 4.7 |
| -0.83 | 8.93 | 636.97 | -2.28 | 5.52 | 634.12 | 6 |
| -2.28 | 5.52 | 634.12 | -9.13 | 4.48 | 645.22 | 6.7 |
| -9.13 | 4.48 | 645.22 | -10.36 | 9.13 | 606.18 | 6 |
| -10.36 | 9.13 | 606.18 | -6.81 | 6.54 | 591.71 | 5 |
| -6.81 | 6.54 | 591.71 | -4.17 | 10.45 | 575.05 | 6 |
| -4.17 | 10.45 | 575.05 | -3.33 | 23.17 | 544.30 | 6 |
| -3.33 | 23.17 | 544.30 | -3.83 | 28.50 | 517.46 | 6 |
| -3.83 | 28.50 | 517.46 | -3.50 | 30.84 | 496.72 | 5.3 |
| -3.50 | 30.84 | 496.72 | -3.76 | 29.41 | 474.59 | 4.3 |
| -3.76 | 29.41 | 474.59 | -2.88 | 25.79 | 457.89 | 5 |
| -2.88 | 25.79 | 457.89 | -0.99 | 18.02 | 442.33 | 4.7 |
| -0.99 | 18.02 | 442.33 | -2.91 | 6.78 | 447.53 | 6 |
| -2.91 | 6.78 | 447.53 | -7.00 | 7.69 | 483.55 | 6 |
| -7.00 | 7.69 | 483.55 | -7.21 | 5.15 | 477.40 | 6 |
| -7.21 | 5.15 | 477.40 | -8.61 | 9.78 | 538.15 | 8.7 |
| -8.61 | 9.78 | 538.15 | -20.55 | 14.42 | 847.45 | 8.3 |
| -20.55 | 14.42 | 847.45 | 0.79 | 18.56 | 776.79 | 5.7 |
| 0.79 | 18.56 | 776.79 | 0.25 | 17.18 | 732.04 | 6 |
| 0.25 | 17.18 | 732.04 | 3.01 | 6.02 | 679.40 | 3.7 |
| 3.01 | 6.02 | 679.40 | 0.09 | 3.28 | 654.23 | 2.3 |
| 0.09 | 3.28 | 654.23 | 0.64 | 2.43 | 598.42 | 3 |
| 0.64 | 2.43 | 598.42 | 3.45 | 1.87 | 653.40 | 3.3 |
| 3.45 | 1.87 | 653.40 | -1.82 | 2.27 | 632.65 | 6.7 |

File **tra.dat** contains training patterns.

tra.dat consists of the storms:

**May 1998**
**June 1998**
**Sep. 1998**
**Nov. 2004**

# The algorithm for the training the neural network

- Backpropagation (BP) algorithm based on the generalized delta rule improved with a momentum term.
- Learning parameters that has to be set up:
  - Learning rate – It affects the speed of learning.
  - Momentum term – It prevents sticking in a local minimum of a function… (↓) .
- The aim is to reach a global minimum of a function describing the error between the actual and desired NN outputs.

# BP algorithm written in Octave

## The name of the file is Learn_NN.m

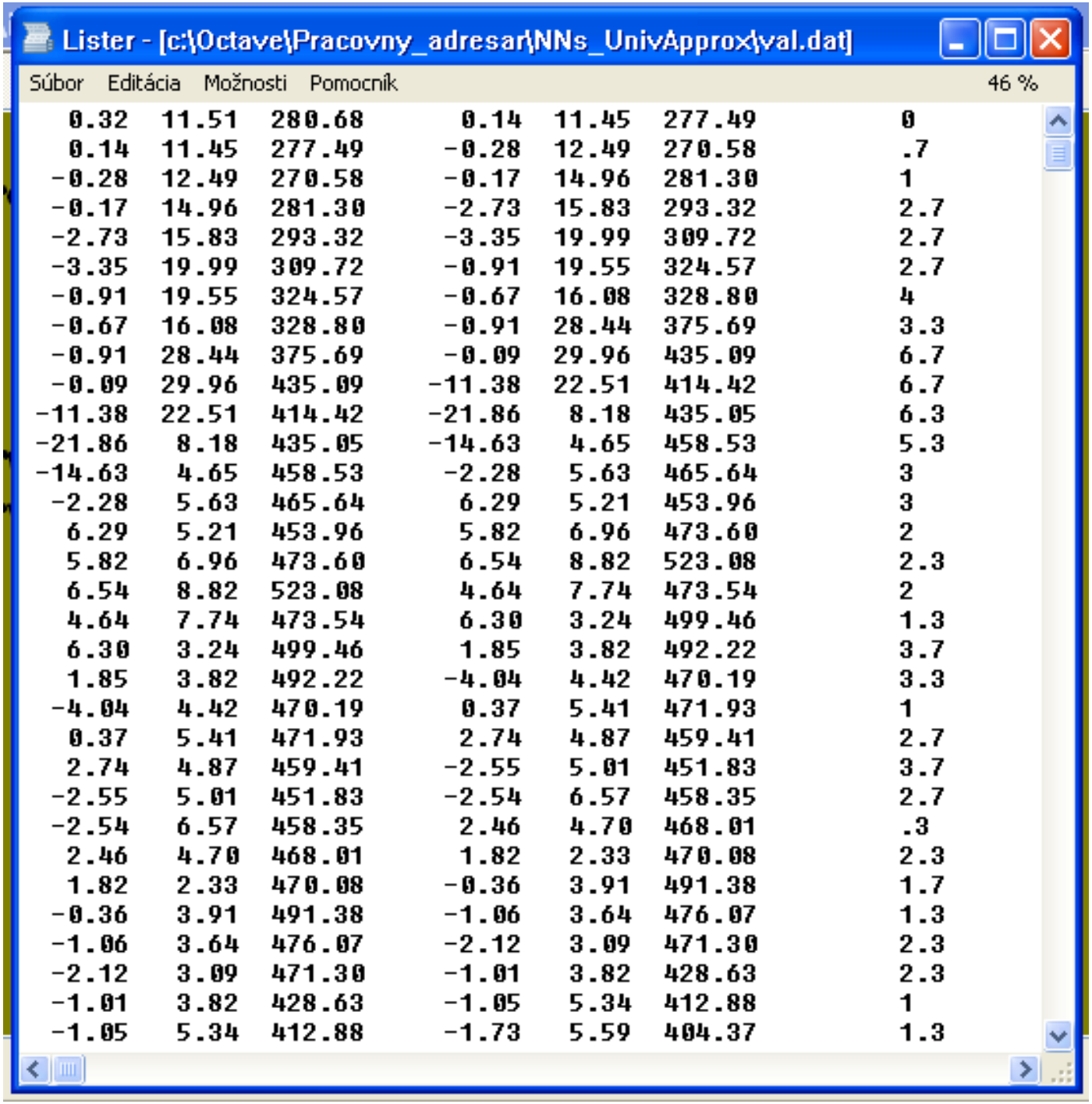
```
function[w,dw,ww,dww,th,dth]=Learn_NN(matrix_of_inputs,vector_of_targets,w,dw,ww,
dww,th,dth,alpha,lambda)
pv=size(matrix_of_inputs)(1);
H=length(th);
for p=1:pv;
        x=matrix_of_inputs(p,:);
        t=vector_of_targets(p);
        a=x*w-th;
        a=1./(1+exp(-a));
        aa=a*ww';
        aa=1/(1+exp(-aa));
        dd=aa*(1-aa)*(t-aa);
        dww=alpha*dd*a+lambda*dww;
        ww=ww+dww;
        d=a*(1-a)'*dd*ww;
        dw=alpha*(x'*d)+lambda*dw;
        w=w+dw;
        dth=alpha*d*(-1)+lambda*dth;
        th=th+dth;
endfor
endfunction
```

# Why Octave?

- Octave is a free software (freeware).
- Multiplatform (Windows, Linux, etc.)
- The syntax of the Octave language is similar to the Matlab’s syntax, which is widely used.
- Octave enables data visualization.
- Octave works with matrices.

# How many hidden neurons do you need?

- At the beginning you will not know how many hidden neurons (H) will be needed.
- You must try to train several different NNs, with different H, and then you will choose which of them yields the best results. (**Using the training patterns**.)
- In order to find which NN is the best, you need to evaluate a test, for which you will use patterns which were not used for training. (**Validation patterns**.)
- The structures of both the training and validation patterns have to be the same, *i.e.*:
    - Inputs: Bz(t-1), n(t-1), V(t-1), Bz(t), n(t), V(t)
    - Desired output (target): Kp(t+1)
- The name of the file with validation patterns is "**val.dat**".




File **val.dat** contains validation patterns.

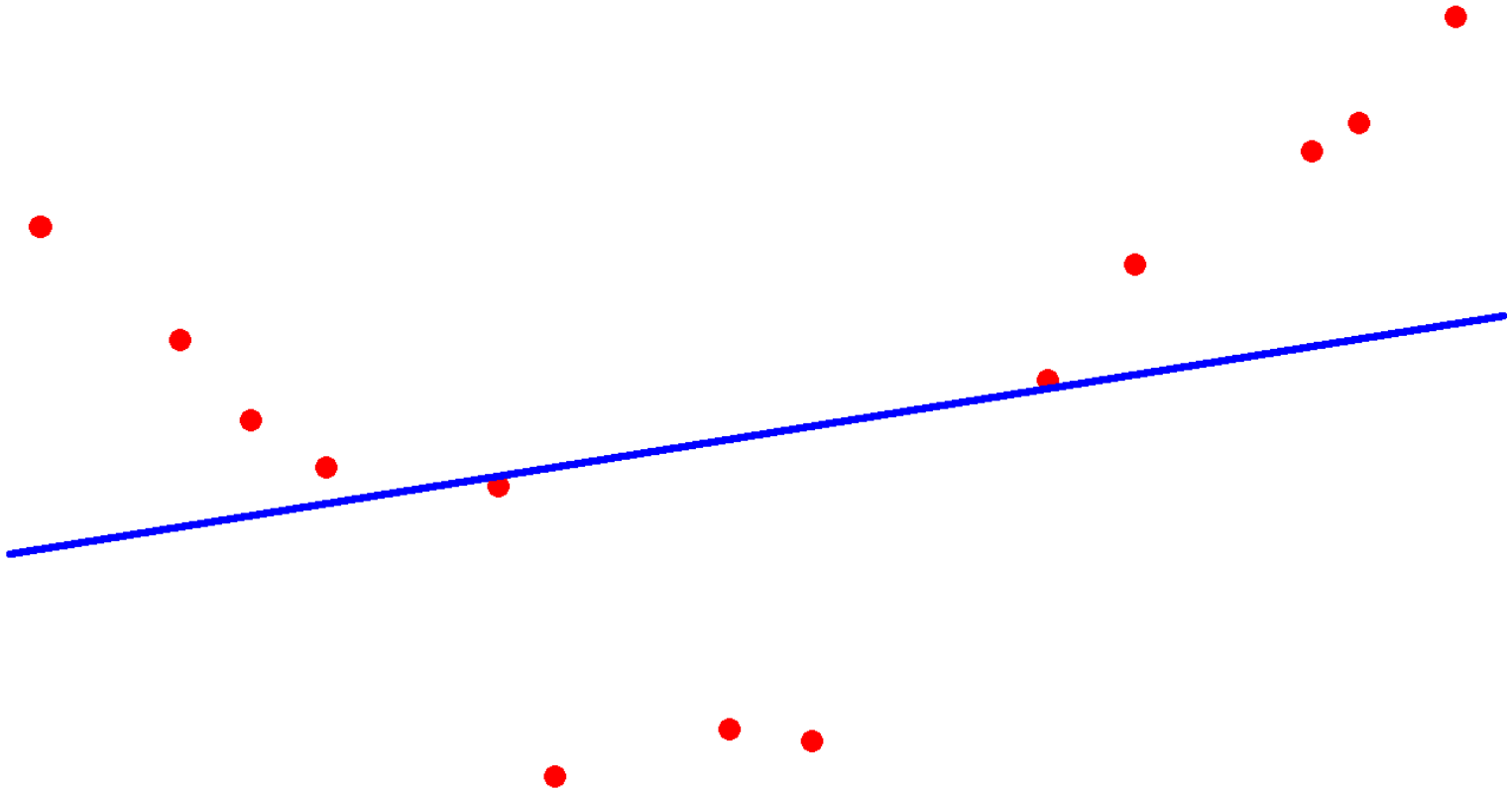
val.dat consists of the storms:

**May 1997**
**October 1998**

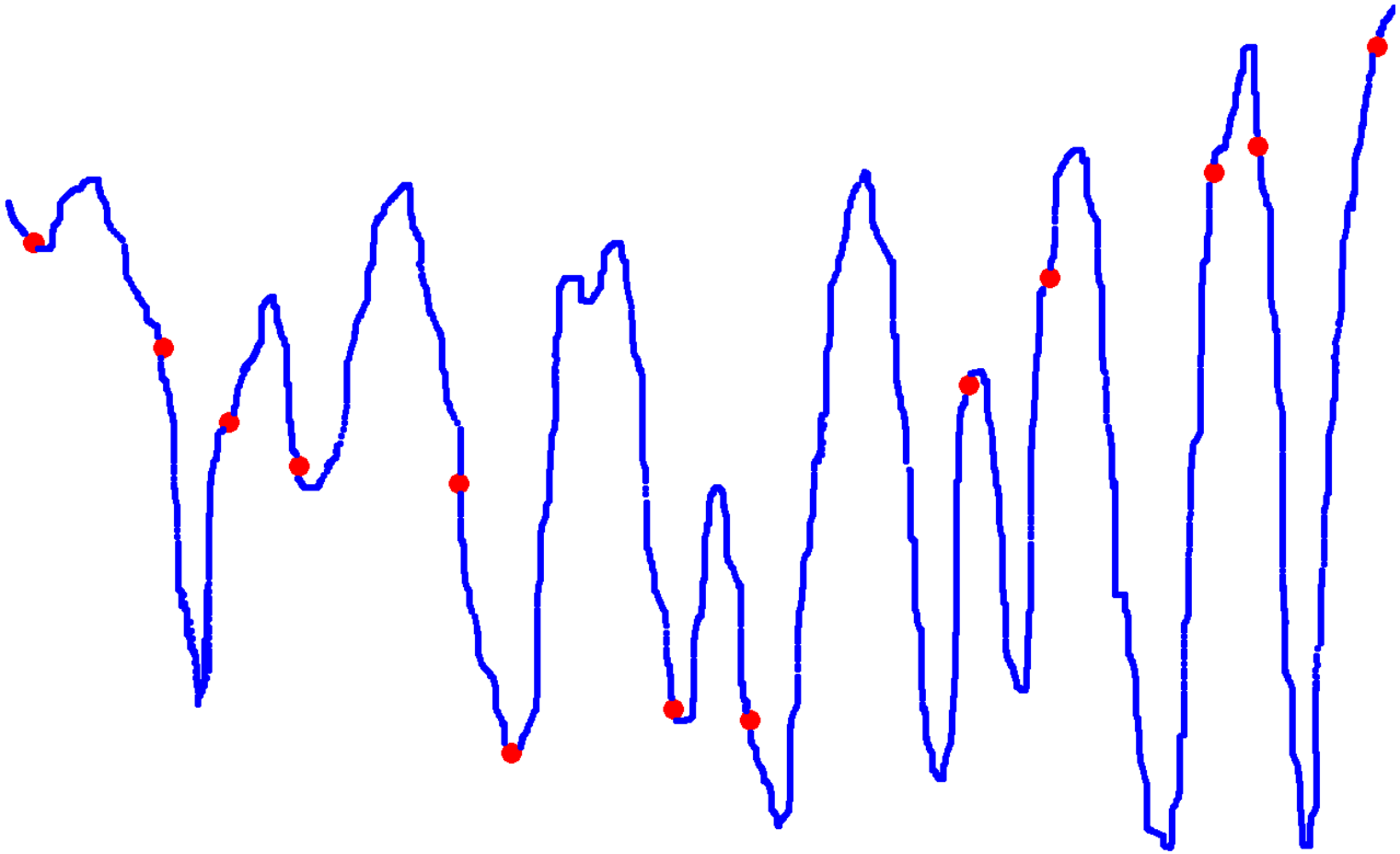
# Why is it so important to have a reasonable number of hidden neurons?

- **Few hidden neurons:** The model is too simple and inadaptable. It is not enough for describing complicated relations.
- **Too many hidden neurons:** The model reproduces the training patterns very literally. However, the NN cannot generalize from training patterns!
  (**over-sizing**)

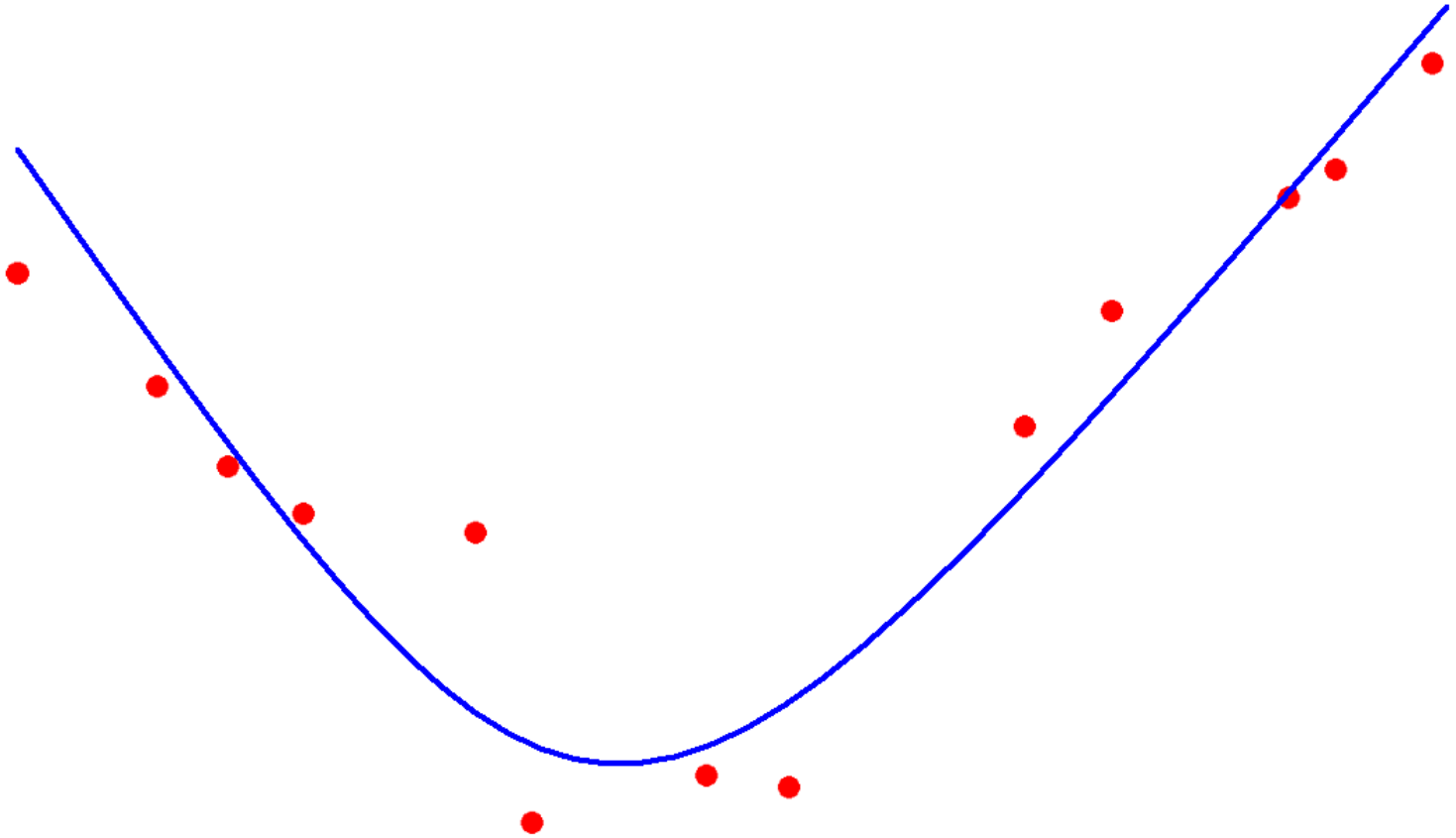
# An analogy with curve fitting:
# Too simple model (= few hidden neurons)



# An analogy with curve fitting:
# Over-fitting (= over-sizing the hidden layer)



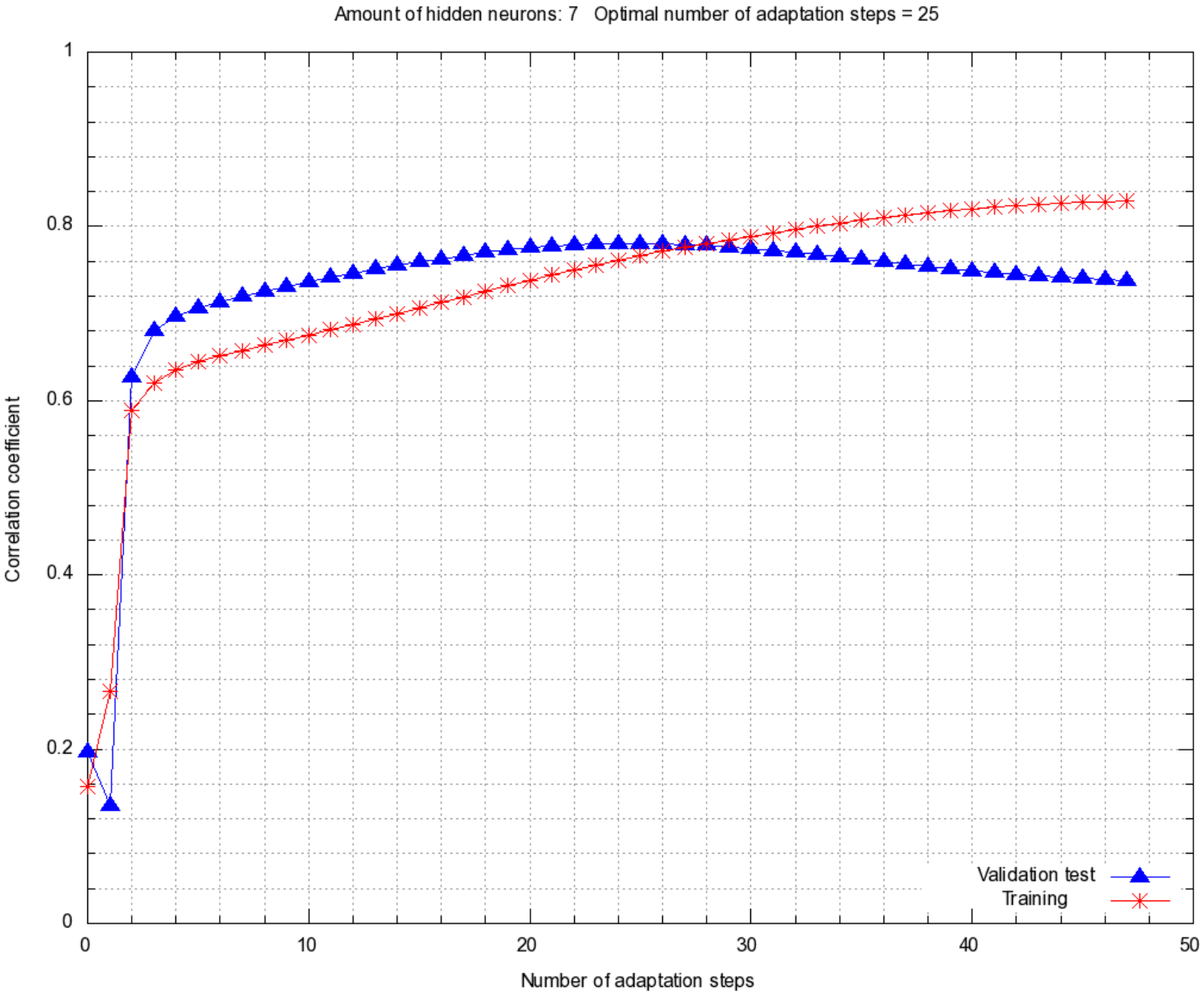
# An analogy with curve fitting:
# Well fitted (= reasonable number of hidden neurons)



# How many adaptation steps (iterations) has to be done?

- During the training, the training patterns are presented to the network repeatedly. Every time the weights and sensitivity thresholds are slightly improved (adapted).
  **When the training has to be stoped?**
- **Few adaptation steps**: The NN is learned poorly. There are great differences between actual and desired outputs of the NN.
- **Too many adaptation steps**: The model reproduces the training patterns very literally. However, the NN cannot generalize from training patterns!
  (**over-learning**)
- Again, the problem can be solved performing a test with validation patterns.

# Training and validation tests' results as a function of the number of adaptation steps



The core of a program for training neural networks

# Scenario_NN.m for Octave

# Walking through Scenario_NN.m

Lister - [c:\Octave\Pracovny_adresar\NNs_UnivApprox\Scenario_NN.m]

Súbor Editácia Možnosti Pomocník 16 %

```
# A scenario for training the neural network.

% Author:
% Fridrich Valach, Geophysical Institute SAS, Geomagnetic Observatory Hurbanovo,
Slovakia
% E-mail: fridrich@geomag.sk

% 2011 ISWI-Europe Summer School in Space Science
% August 21-27, 2011, High Tatras, Slovakia


# Loading training and validation patterns:
load tra.dat
load val.dat

% Detecting the number of input neurons:
n_of_inputs=size(tra)(2)-1;


matrix_inputs_tra=tra(:,1:n_of_inputs);
vector_targets_tra=tra(:,n_of_inputs+1);
matrix_inputs_val=val(:,1:n_of_inputs);
vector_targets_val=val(:,n_of_inputs+1);

% Input data normalization (Not necessary, but recommended.):

for n=1:n_of_inputs
        matrix_inputs_tra(:,n)=(tra(:,n).-mean(tra(:,n)))./std(tra(:,n));
endfor
for n=1:n_of_inputs
        matrix_inputs_val(:,n)=(val(:,n).-mean(tra(:,n)))./std(tra(:,n));
endfor
```

*Loading training and validation patterns.*

# Walking through Scenario_NN.m

Lister - [c:\Octave\Pracovny_adresar\NNs_UnivApprox\Scenario_NN.m]

Súbor Editácia Možnosti Pomocník 33 %

```
vector_targets_tra=tra(:,n_of_inputs+1);
matrix_inputs_val=val(:,1:n_of_inputs);
vector_targets_val=val(:,n_of_inputs+1);

% Input data normalization (Not necessary, but recommended.):

for n=1:n_of_inputs
        matrix_inputs_tra(:,n)=(tra(:,n).-mean(tra(:,n)))./std(tra(:,n));
endfor
for n=1:n_of_inputs
        matrix_inputs_val(:,n)=(val(:,n).-mean(tra(:,n)))./std(tra(:,n));
endfor


% Targets (or NN outputs) have to be between 0 and 1:
vector_targets_tra=vector_targets_tra/9;
vector_targets_val=vector_targets_val/9;


# Setting the training parameters:
'-------------------------------------------------------------'
% Setting the value of the learning rate:
'Input a value of the learning rate:'
alpha=input('(Recommended is value around .1 .)  ... ');
'-------------------------------------------------------------'
% Setting the value of the momentum term:
'Input a value of the momentum term:'
lambda=input('(Recommendation between .5 - .7 .)  ... ');
'-------------------------------------------------------------'
% Setting the number of hidden neurons:
H=input('Input the number of neurons in the hidden layer: ');
'-------------------------------------------------------------'
% Setting the number of adaptation steps (iterations) in one training cycle:
```

*Input data normalization.*

*Output data scaling to interval (0,1).*

# Walking through Scenario_NN.m

Lister - [c:\Octave\Pracovny_adresar\NNs_UnivApprox\Scenario_NN.m]

Súbor Editácia Možnosti Pomocník 46 %

```
vector_targets_tra=vector_targets_tra/9;
vector_targets_val=vector_targets_val/9;

# Setting the training parameters:
'-------------------------------------------------------------'
% Setting the value of the learning rate:
'Input a value of the learning rate:'
alpha=input('(Recommended is value around .1 .)  ... ');
'-------------------------------------------------------------'
% Setting the value of the momentum term:
'Input a value of the momentum term:'
lambda=input('(Recommendation between .5 - .7 .)  ... ');
'-------------------------------------------------------------'
% Setting the number of hidden neurons:
H=input('Input the number of neurons in the hidden layer: ');
'-------------------------------------------------------------'
% Setting the number of adaptation steps (iterations) in one training cycle:
iterations_in_a_cycle=1;
'-------------------------------------------------------------'
% Setting the maximum number of training cycles:
max_no_of_cycles=input("Input the maximal number of adaptation steps: ");
'-------------------------------------------------------------'
"How many more adaptation steps will be done after reaching"
max_for_how_many_more=input("      the best (up to now) results for validation?
.......... ");
'-------------------------------------------------------------'


% Random initiating the weights and sensitivity thresholds:
[w,dw,ww,dww,th,dth]=Zero_shots(matrix_inputs_tra,H);

[NNout]=Output_of_NN(w,ww,th,matrix_inputs_tra);
```

*Setting the training parameters*

# Walking through Scenario_NN.m

Lister - [c:\Octave\Pracovny_adresar\NNs_UnivApprox\Scenario_NN.m]

Súbor Editácia Možnosti Pomocník 58 %

```
max_no_of_cycles=input("Input the maximal number of adaptation steps: ");
'-------------------------------------------------------------'
"How many more adaptation steps will be done after reaching"
max_for_how_many_more=input("      the best (up to now) results for validation?
.......... ");
'-------------------------------------------------------------'


% Random initiating the weights and sensitivity thresholds:
[w,dw,ww,dww,th,dth]=Zero_shots(matrix_inputs_tra,H);

[NNout]=Output_of_NN(w,ww,th,matrix_inputs_tra);
cc_tra=corrcoef(NNout,vector_targets_tra');
[NNout]=Output_of_NN(w,ww,th,matrix_inputs_val);
cc_val=corrcoef(NNout,vector_targets_val');
# Correlation coefficients are used in order to evaluate the validation.

cc_val_max=cc_val;
wBest=w; wwBest=ww; thBest=th;
counter=0; counterBest=0;

counter_how_many_more=0;

info_about_cc=[0 cc_tra cc_val];

# We will draw a picture during the training.
# The picture will show correlation coefficients - comparing targets (desired
outputs) with
# the current neural-network outputs for both training and validation databases.
newplot

% First values in the picture are for the randomly initiated weights and
sensitivity thresholds:
```

*The weights and sensitivity thresholds are set to small random numbers.*

# Walking through Scenario_NN.m

Lister - [c:\Octave\Pracovny_adresar\NNs_UnivApprox\Scenario_NN.m]

Súbor Editácia Možnosti Pomocník 91 %

```
for cc=1:max_no_of_cycles

    for c=1:iterations_in_a_cycle
        % Training. That means an adaptation of the weights as well as the sensitivity
thresholds:
        [w,dw,ww,dww,th,dth]=Learn_NN(matrix_inputs_tra,vector_targets_tra,w,dw,ww,dww,t
h,dth,alpha,lambda);
    endfor

    counter=counter+iterations_in_a_cycle;
    % Calculating the neural network outputs for the training patterns:
    [NNout]=Output_of_NN(w,ww,th,matrix_inputs_tra);
    % Comparing the NN outputs with the desired outputs (targets):
    cc_tra=corrcoef(NNout,vector_targets_tra');
    % Calculating the neural network outputs for the validation patterns:
    [NNout]=Output_of_NN(w,ww,th,matrix_inputs_val);
    % Comparing the NN outputs with the desired outputs (targets):
    cc_val=corrcoef(NNout,vector_targets_val');
    % Recording the information about correlation coefficients for training and
validation:
    info_about_cc=[info_about_cc;counter cc_tra cc_val];

    # Finding the best weights and sensitivity thresholds, for which the
validation error is minimal:
    if (cc_val>cc_val_max)
        cc_val_max=cc_val;
        wBest=w; wwBest=ww; thBest=th;
        counterBest=counter;
        counter_how_many_more=0;
      else
        counter_how_many_more=counter_how_many_more+1;
        if (counter_how_many_more==max_for_how_many_more+1)
```

*Repeatedly adapting weights and sensitivity thresholds.*

*Correlation between desired and actual outputs of the NN is calculated for both training and validation patterns.*

*The best weights and sensitivity thresholds are stored during the training process.*

# Walking through Scenario_NN.m

Lister - [c:\Octave\Pracovny_adresar\NNs_UnivApprox\Scenario_NN.m]

Súbor Editácia Možnosti Pomocník 97 %

```
    [NNout]=Output_of_NN(w,ww,th,matrix_inputs_tra);
    % Comparing the NN outputs with the desired outputs (targets):
    cc_tra=corrcoef(NNout,vector_targets_tra');
    % Calculating the neural network outputs for the validation patterns:
    [NNout]=Output_of_NN(w,ww,th,matrix_inputs_val);
    % Comparing the NN outputs with the desired outputs (targets):
    cc_val=corrcoef(NNout,vector_targets_val');
    % Recording the information about correlation coefficients for training and
validation:
    info_about_cc=[info_about_cc;counter cc_tra cc_val];

    # Finding the best weights and sensitivity thresholds, for which the
validation error is minimal:
    if (cc_val>cc_val_max)
        cc_val_max=cc_val;
        wBest=w; wwBest=ww; thBest=th;
        counterBest=counter;
        counter_how_many_more=0;
    else
        counter_how_many_more=counter_how_many_more+1;
        if (counter_how_many_more==max_for_how_many_more+1)
                'The training of the neural network has been finished.'
                break;
        endif
    endif

# Updating the picture:
plot(info_about_cc(:,1),info_about_cc(:,3)','-^b');
plot(info_about_cc(:,1),info_about_cc(:,2)','-*r');
pause(1)

endfor
```

*If the results for the validation test are not improving, the training is terminated.*

# Some complementing scripts:

- Comparing the forecast Kp's with the observed Kp's. (*You can compare the forecasts with the observed values visually or using CC, RMSE, mean absolute error, or median of absolute errors*.)
- Saving the figures produced during the training.
- Saving the parameters of the neural network.
- Performing a final test.
- ----------
- There is a script named Script_for_NN.m, which brings together all the scripts you need to use.
- ----------

# The exercise is:

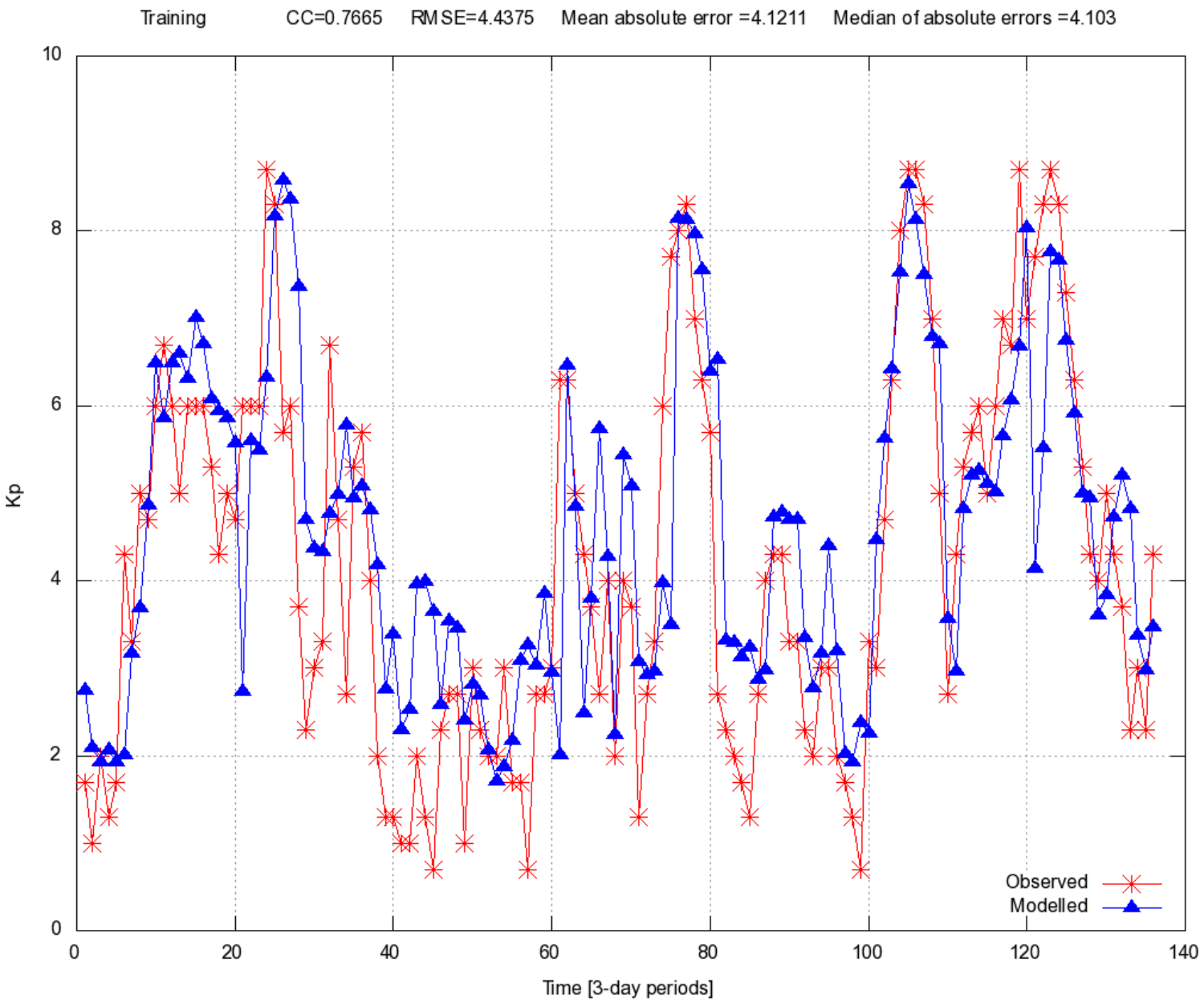
- Train a neural network.
- Forecast Kp indexes for the geomagnetic storms of August 1998, October 1998, and November 2004.

- (Use Script_for_NN.m for this purpose.)

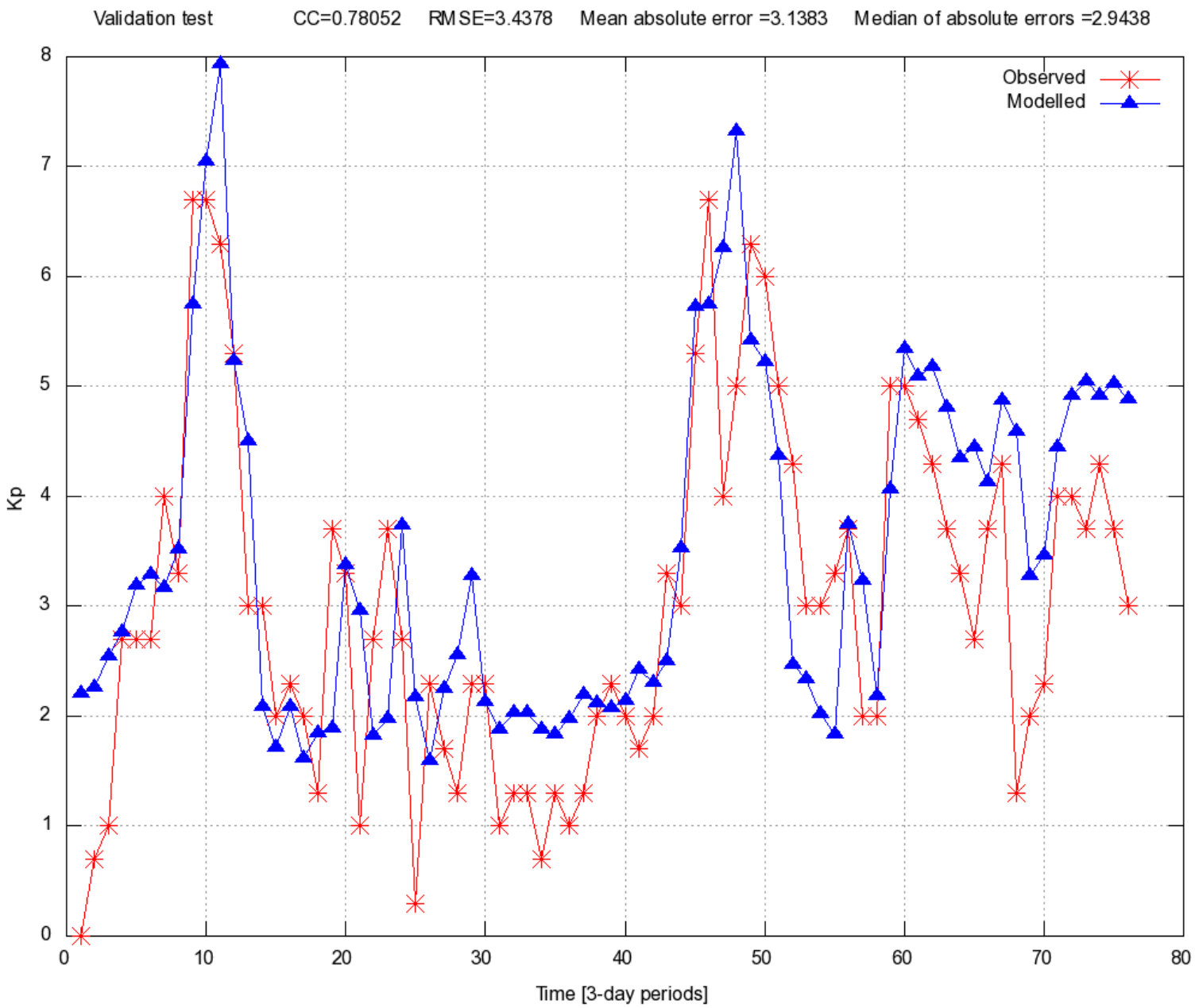
# Now, it is the time to run Script_for_NN.m

- How to do it:
  - Start up octave.
  - Change the working directory writing, e.g., cd ‘c:\\Octave\\HighTatras’
  - Write **Script_for_NN** to the octave’s command line and press **Enter**.
  - Follow the instructions which will appear running the program.
- If some problem occurs, please, ask me or my assistant for help.

# Satisfying results of the test for the training patterns



# Satisfying results of the test for the validation patterns



# Please send me your results.

- When you finish the work with Script_for_NN, you will have three files named "Results_for_Aug98.dat", "Results_for_Oct98.dat" and "Results_for_Nov04.dat", respectively, created in your working directory.
- Please send me these files to my e-mail address fridrich@geomag.sk.
- Write me also how many hidden neurons did you find to be optimal.
- I will summarize the results. – You will find the summary on the billboard in the entrance-hall tomorrow morning.

# Conclusions

- A simple neural network with the layer of hidden neurons was introduced.
- I convinced you (I hope so) that the neural network is a usefull tool for space weather modelling.
- Script_for_NN.m was introduced and you tried to train a neural network by yourselves using it.

Thanks for your attention