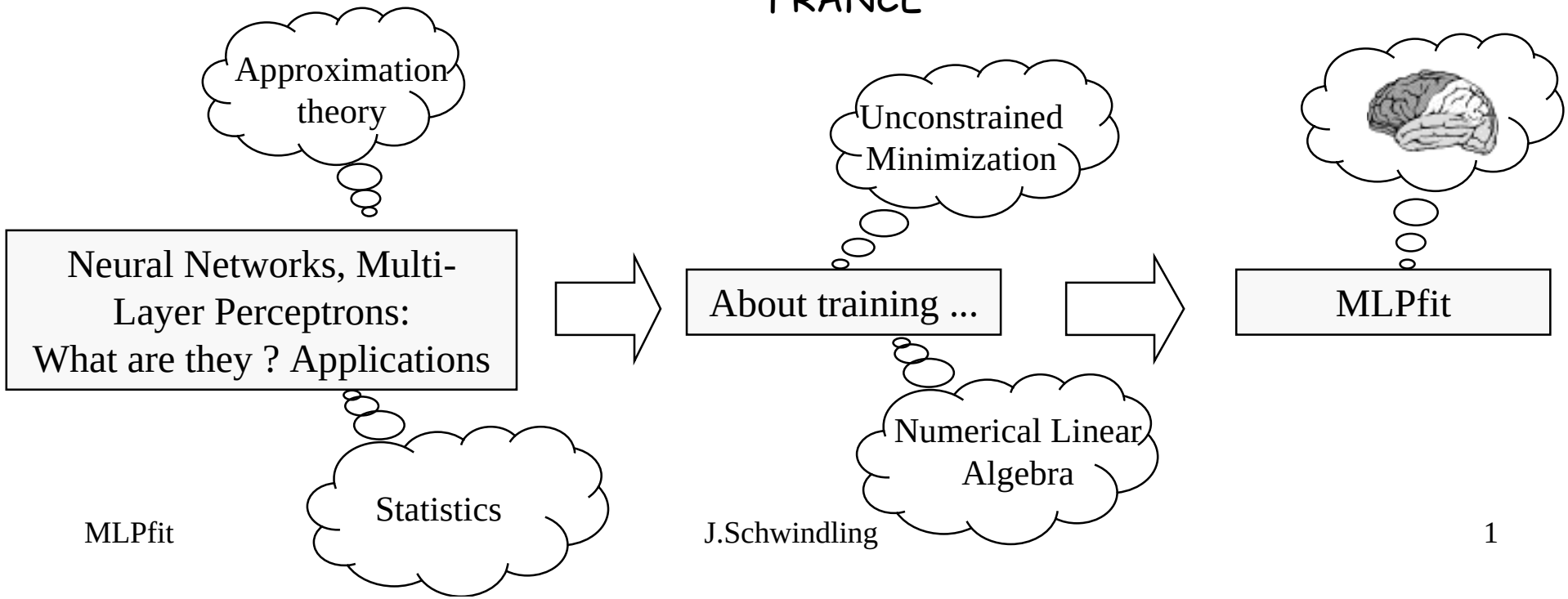
MLPfit: a tool to design and use Multi-Layer Perceptrons

J. Schwindling, B. Mansoulié

CEA / Saclay
FRANCE



(Artificial) Neural Networks

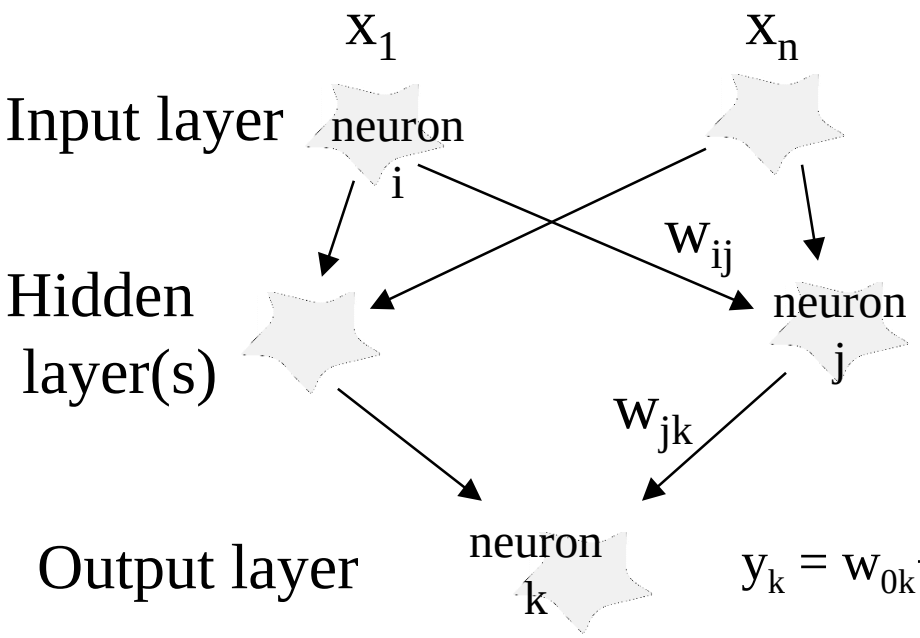
- Appeared in the 40 's
- Now (since Personal Computers) very widely used, in various domains:
  - in medecine (image analysis, help to diagnosis)
  - in meteorology (predictions)
  - in industry (automatic process control, quality checks by image processing, optimization of ressources allocation)

(see for example *IEEE Transactions on Neural Networks*)

Neural Networks in HEP

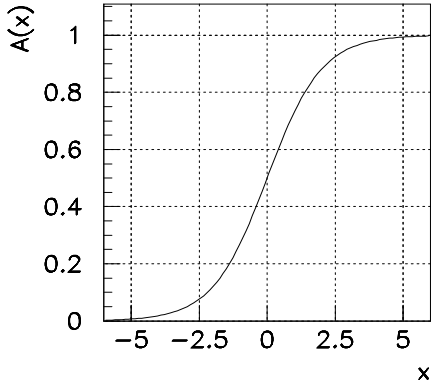
- Used for ~10 years
- Mainly for (offline) classification:
  - particle identification (b quarks)
  - event classification (e.g. WW -> qqqq versus ZZ, qq at LEP)
  - search for new physics (Higgs)
- Track reconstruction
- Trigger
  - in H1
- Function approximation
  - position measurement in ATLAS electromagnetic calorimeter

The most wi(l)dely used: the Multi-Layer Perceptron



$$A(x) = \frac{1}{1+e^{-x}}$$

« sigmoid »



$u_j = A(w_{0j}+\Sigma w_{ij}\, x_i)$

$y_k = w_{0k}+\Sigma w_{jk}\, u_j$

Linear output neuron:

- natural for function approximation
- can be used also for classification
- used in hybrid learning method (see below)

2 theorems

• any continuous function (1 variable or more) on a compact set can be approximated to any accuracy by a linear combination of sigmoïds -> function approximation

[for example: K.Hornik et al. *Multilayer Feedforward Networks are Universal Approximators*, Neural Networks, Vol. 2, pp 359-366 (1989)]

• trained with f(x) = 1 for signal and = 0 for background, the NN function approximates the probability of signal knowing x -> classification (cf Neyman-Pearson test)

[for example: D.W.Ruck et al. *The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function*, IEEE Transactions on Neural Networks, Vol. 1, n^{r} 4, pp 296-298 (1990)]

About learning ...

• loop on the set of exemples p, compute the MLP output y_p, compare with desired answer d_p, update the weights = an epoch

• aim is to minimize $E = \sum_p \omega_p \left(y_p - d_p \right)^2 = \sum_p e_p$ (usually $\omega_p = 1$)

• all the methods use the computation of the first order derivatives $\partial E / \partial w_{ij} = \sum_p \partial e_p / \partial w_{ij}$ computed by «backpropagation of errors»

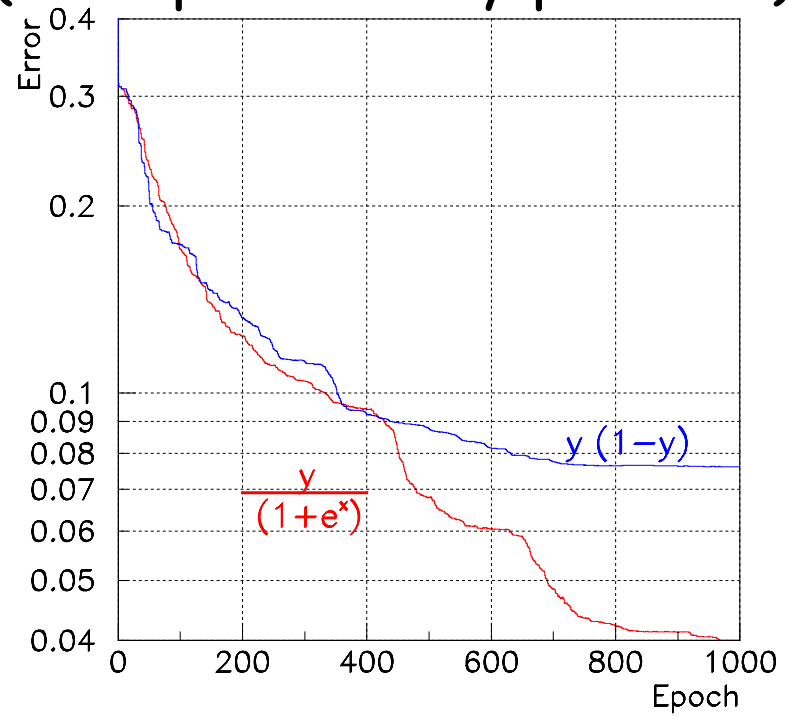
Remark: derivative of sigmoid function

$$y = 1 / (1+e^{-x})$$

- Needed for gradient computation
- Most people use y (1-y)

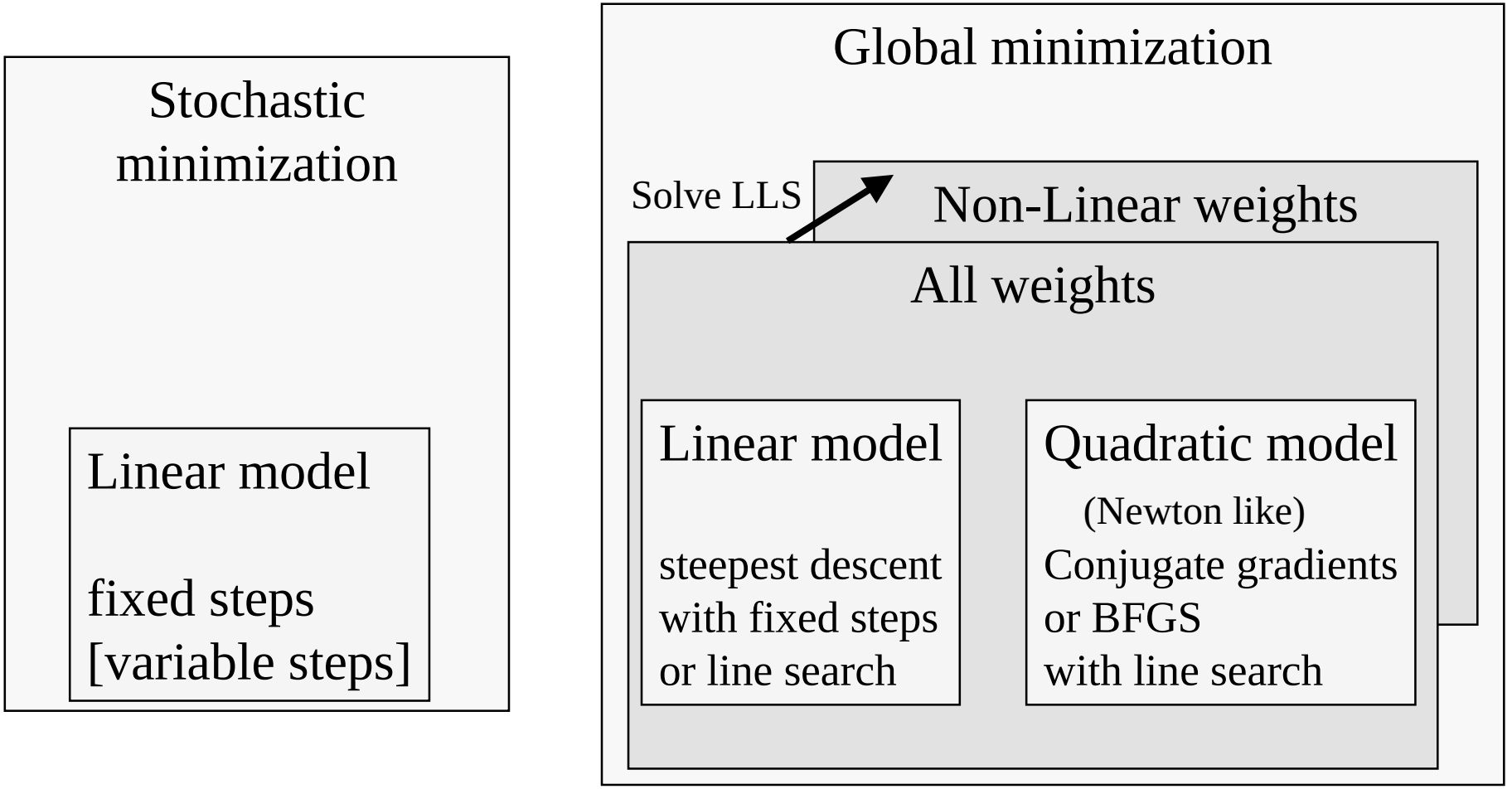
  (no need to recompute e^x)
- However, precision problems when y ~ 1.
- More accurate formula is y / ($1+e^x$)

(exemple on a toy problem)



Learning methods

Remarks: - derivatives known
- other methods exist



The traditional learning method: stochastic minimization

- update the weights after each example according to

$$\Delta w_{ij}^{t} = -\eta \frac{\partial e_p}{\partial w_{ij}} \qquad \left(+ \varepsilon \Delta w_{ij}^{t-1}\right)$$

- «invented» in 1986 for Neural Networks

  Learning representations by back-propagating errors, D.E.Rumelheart et al., Nature vol. 323 (1986), p. 533

  Learning processes in an assymetric threshold network, Y. le Cun, *Disordered Systems and Biological Organization*, Springer Verlag, Les Houches, France (1986), p. 233

- similar to stochastic approximation method

  A Stochastic Approximation Method, H.Robbins et S.Monro, Annals of Math. Stat. 22 (1951), p. 400

Stochastic minimization (continued)

known to converge under certain conditions (η decreasing with time)

Dvoretsky theorem: *On Stochastic Approximation,* A.Dvoretsky, Proc. 3rd Berkeley Sym. on Math. Stat. and Prob., J.Neyman (ed.) (Berkeley: University of California Press, 1956), p. 39

also known to be very slow: «The main effect of random error is to slow down the speed at which a search can be conducted and still be sure of eventually finding the optimum. Stochastic procedures, being very deliberate, should not be used in the absence of experimental error, for deterministic methods are much faster. This point has not been well understood in the past, and stochastic procedures have sometimes been applied to deterministic problems with disappointing results.»

Optimum Seeking Methods, D.J.Wilde, Englewood Cliffs, N.J.Prentice-Hall (1964)

Other methods

- minimize $E = f(w_{ij})$ using «standard» (unconstrained) minimization methods (when 1st order derivatives known)
- for each epoch t:
  - compute gradient $\vec{g} = \partial E / \partial \vec{w}$
  - compute a direction $\vec{s}_t$
  - find a_m which minimizes $E\left(\vec{w}_t + \alpha \vec{s}_t\right)$ (Line Search)
  - set $\vec{w}_{t+1} = \vec{w}_t + \alpha_m \vec{s}_t$
- several ways to choose the direction s_t: conjugate gradients, BFGS
- *Practical Methods of Optimization* R.Fletcher, 2nd edition, Wiley (1987)

The Hybrid method

- *A Hybrid Linear / Nonlinear Training Algorithm for Feedforward Neural Networks* , S. McLoone et al., IEEE Transactions on Neural Networks, vol. 9, n^r 4 (1998), p.669

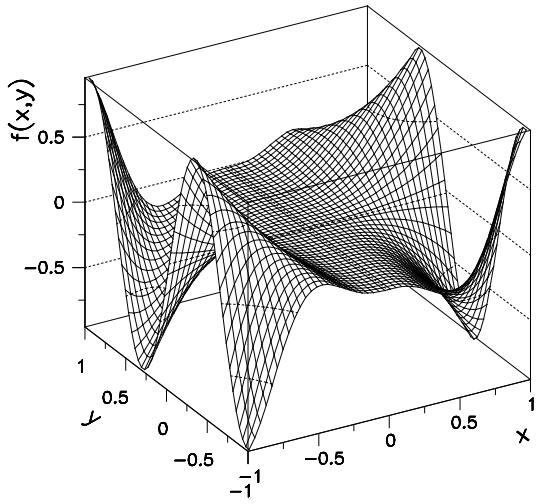
- Idea: for a given set of weights from input layer to hidden layer (w_{NL}: non linear weights), the optimal weights from hidden to output layer (w_L = linear weights) can be obtained by solving a linear system of equations -> w_L^*.

- use BFGS to find minimum of $E(w_{NL}, w_L^*(w_{NL}))$

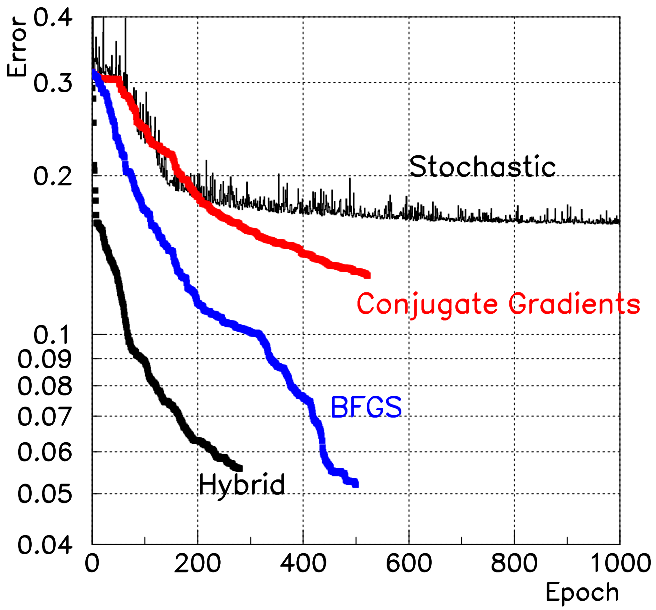
- at some learning steps, linear weights may become too large: add a regularisation term: $E' = E\,(1 + \lambda\, ||w_L||^2 / ||w_{max}||^2)$

Comparison on a toy problem

fit the function $x^2 \sin(5xy)$
by a 2-10-1 MLP



learning curves:
(all curves = 2 minutes CPU)



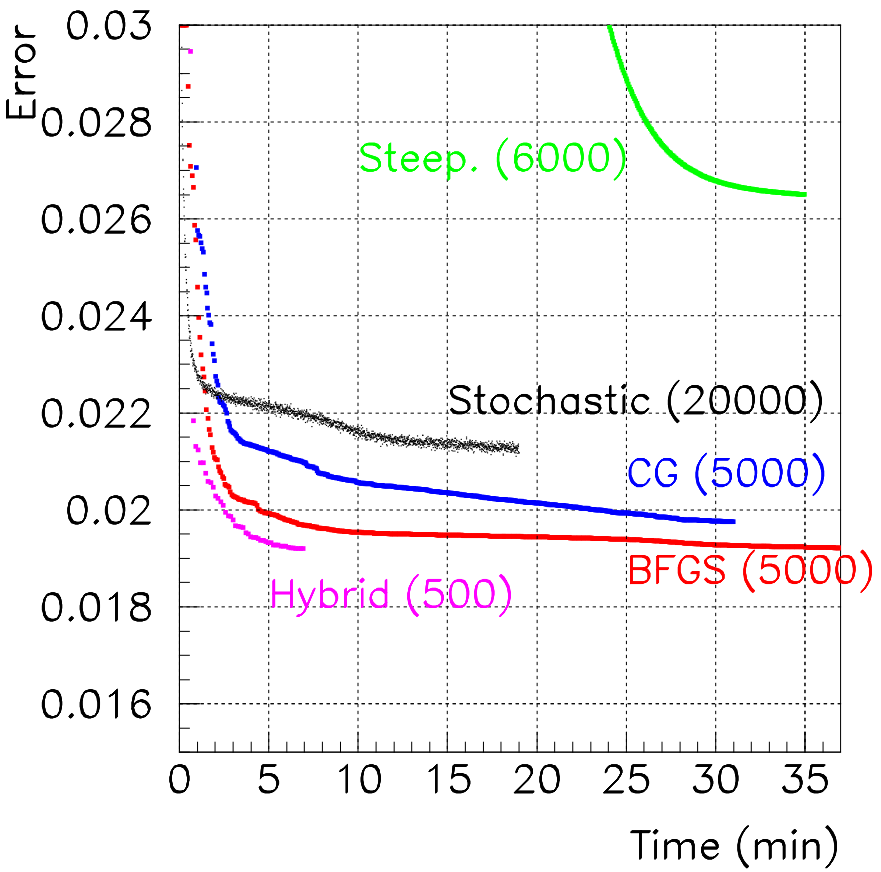
Comparison on a real problem

but small

- Position measurement in ATLAS electromagnetic calo.

- Performance is limited to the calorimeter intrinsic resolution

- However, in a finite time, BFGS or Hybrid methods lead to a resolution ~ 10 % better than stochastic minimisation



The MLPfit package

Based on the mathematical background:

- designed to be used both for function approximation and classification tasks
- implement powerful minimization methods

Software process

Use the POP[(1)] methodology to write a code:

- Simple: less than 3000 lines of (procedural) C in 3 files
- Precise: all computations in double precision, accurate derivative of sigmoid function
- Fast: tests of speed are being conducted (see below)
- Inexpensive: dynamic allocation of memory, examples in single precision (can be turned to double precision by changing one line)
- Easy to use: currently 4 ways to use it (see below)

(1) POP = Performance Oriented Programming

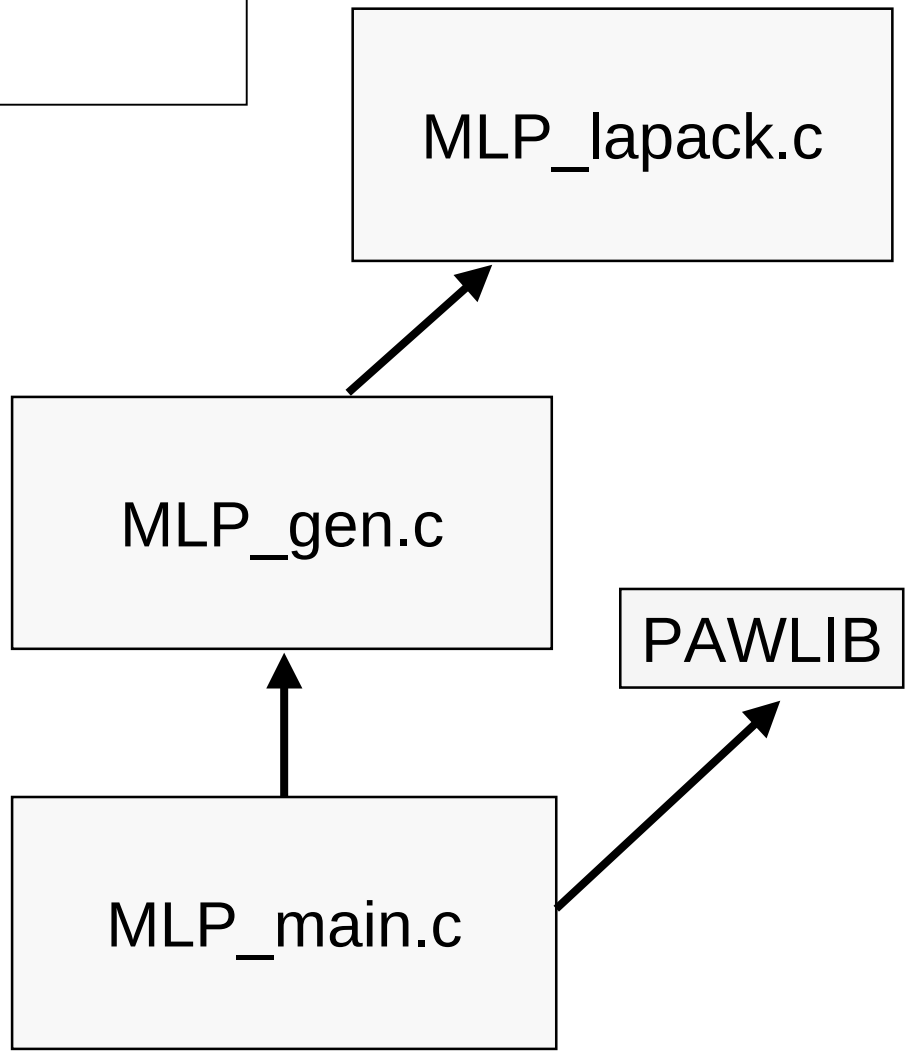
Performances

	JETNET 3.4	SNNS 4.1	MLPfit 1.32
• Possibilities			
Learning Methods	Stochastic + … Conjugate Grad.	Stochastic + …	Stochastic, CG, BFGS, Hybrid
Default interface	your code	Graphical User 's Interface	See below
• Accuracy			
Derivatives	inaccurate	inaccurate	accurate
Precision	single	single	double
• Speed / Memory	(on a classification task using a 14-20-20-1 network, 60000 learning examples, 15000 test examples, stochastic meth.)		
Time/epoch	8.3 s	17.8 s	8.8 s
Memory	8.7 Mb	11.6 Mb	10.1 Mb

Using MLPfit as a standalone package

- Reads ASCII files containing
  - description of network, learning, running conditions
  - examples (learn and test)
  - [initial weights]
- Writes ASCII files containing
  - output weights
  - MLP (fortran or C) function
- Writes PAW file containing learning curves

(Tested on HP-UX and alphavax / OSF)



What is MLP_lapack.c ?

- Need good routine to solve Linear Least Square problem in hybrid learning method
- first tried tls from CERNLIB:
  - + free, source available
  - = fortran
  - - single precision
- then used f04jaf from NAGLIB:
  - + double precision
  - = fortran
  - - not free
- now use dgels from LAPACK:
  - + double precision, C, free, source available
  - - 35% slower than f04jaf

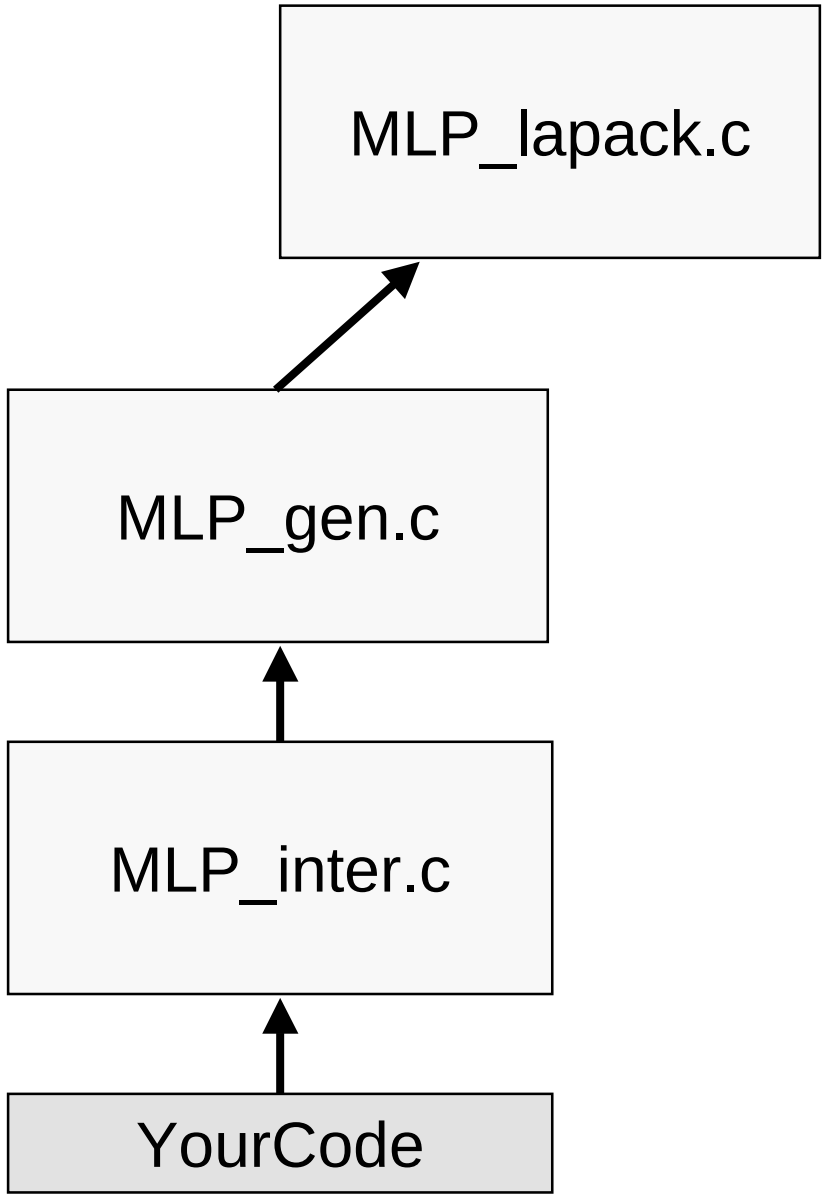
MLP_lapack.c

(~ 9000 lines)

Calling MLPfit from your own code

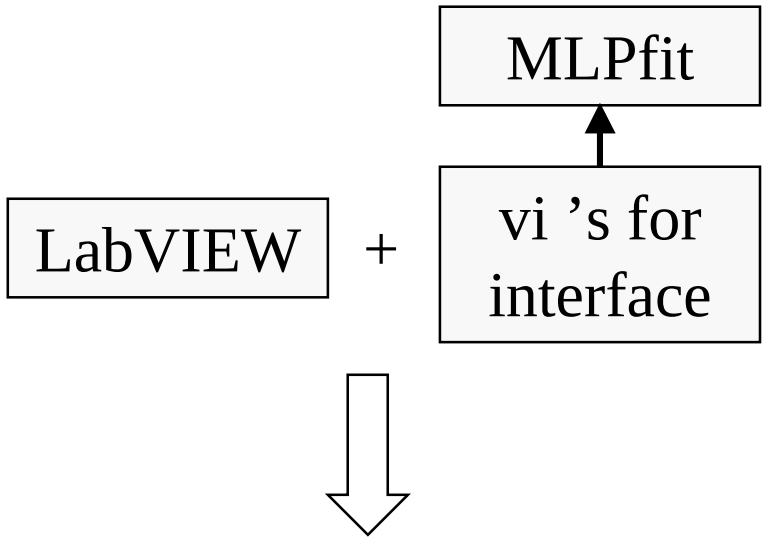
- Can be called from fortran or C(++) code
- Routines to:
  - define network
  - choose learning method and parameters
  - set the examples
  - train, test, use the network

(Tested on HP-UX and alphavax / OSF)



Calling MLPfit from LabVIEW

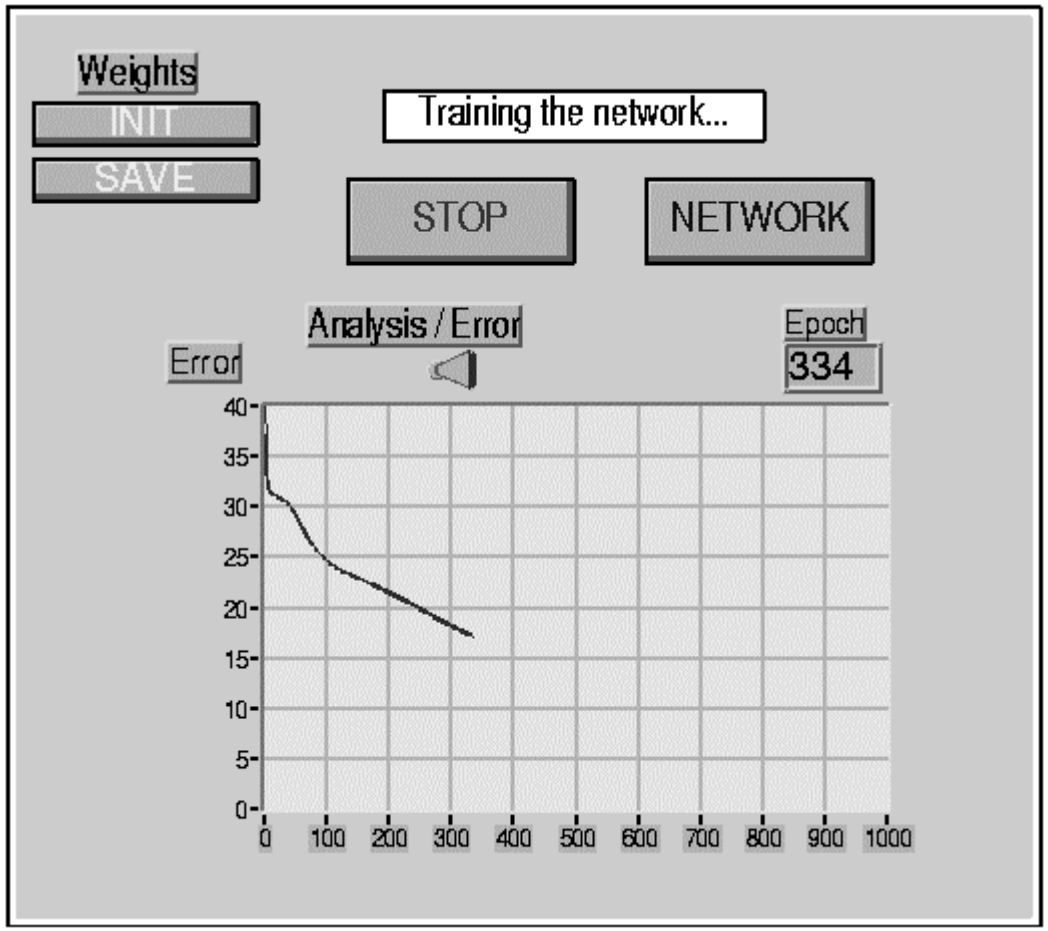
(tested on PC / Windows 95 + LabVIEW 4.1)



your optimized:

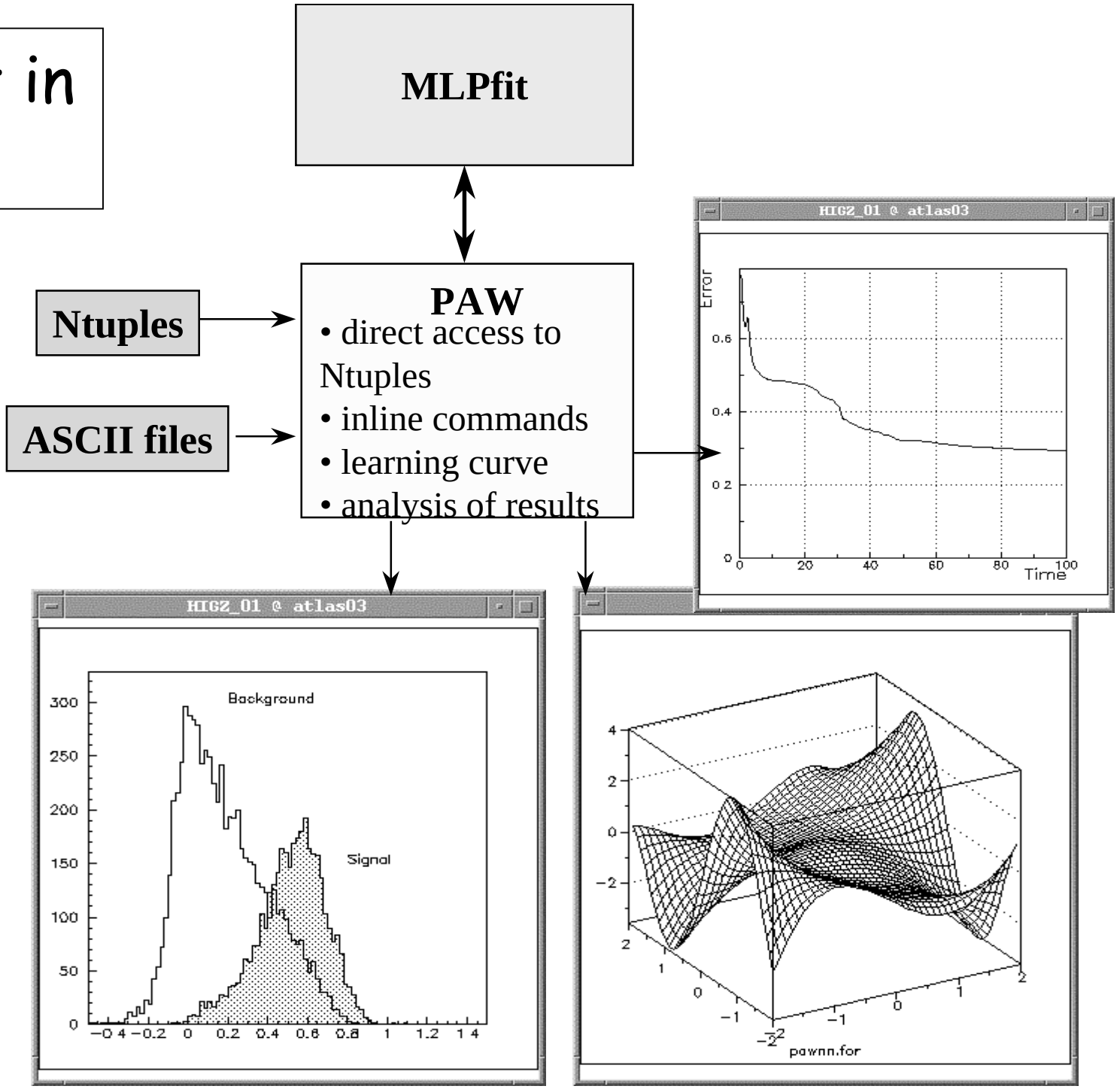
- MLP application
- Graphical User 's Interface

- automatic process control
- image analysis



Using MLPfit in PAW

(tested on HP-UX)



(an interface to ROOT is also under study)

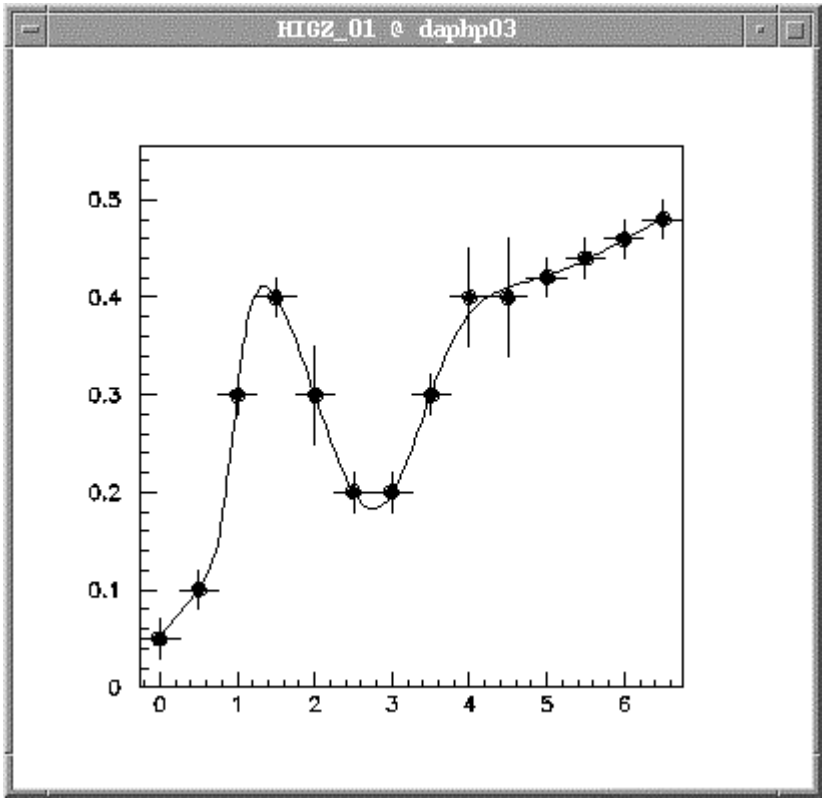
MLPfit in PAW: modified vec/fit command

- vec/fit can be used to fit the parameters of a known function on a set of points
- However, function not always known (and not always easy to guess)…
- Can then use

vec/fit x,y,ey NNi ! 500

Number of hidden neurons

Number of epochs



MLPfit in PAW: interface to Ntuples

- Define signal and background examples directly from your Ntuples:
  - 1000 signal events from file 1

    nn/set/lpat //lun1/2000 v1%sqrt(v2)%v3+v4 1. 1. 1000 1 v5>1.5

    ntuple 3 variables answer weight. #examples first cut

  - add 1000 background events from file 2

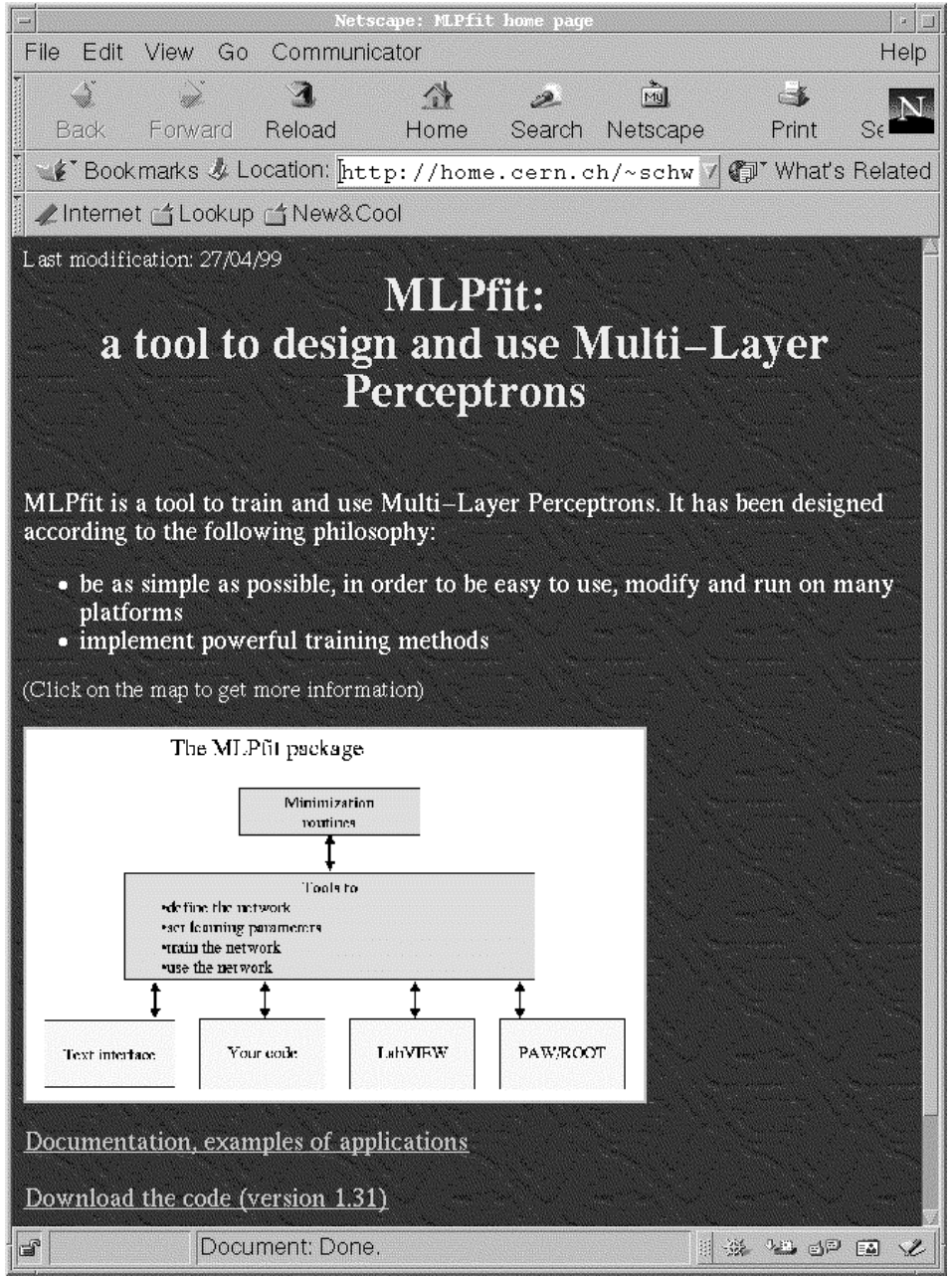
    nn/set/lpat //lun2/2000 v1%sqrt(v2)%v3+v4 0. 1. 1000 1 v5>1.5 +

- Then train network within PAW ... Test it in the same session

Documentation …

Everything (doc, code, examples, …) can be found at

http://home.cern.ch/~schwind/MLPfit.html



Conclusions

- Neural Networks are very widely used
- Multi-Layer Perceptrons are based on mathematical grounds (function approximation, classification, learning)
- MLPfit implements powerful learning methods in a simple, precise, fast way
- Several (simple) interfaces to use it

We are willing to ...

- Support the code:
  - help to use it, give advice on MLP, correct bugs
- Improve the code:
  - try a few more learning methods
  - implement additional features (computation of second derivatives, …) if useful
  - provide other interfaces on request
- Test the code:
  - compare MLPfit with other NN packages