```
Archive-name: ai-faq/neural-nets/part5
Last-modified: 2002-08-11
URL: ftp://ftp.sas.com/pub/neural/FAQ5.html
Maintainer: saswss@unx.sas.com (Warren S. Sarle)
```


This is part 5 (of 7) of a monthly posting to the Usenet newsgroup comp.ai.neural-nets. See the part 1 of this posting for full information what it is all about.

========== Questions ==========

Subject: Source code on the web?

The following URLs are reputed to have source code for NNs. Use at your own risk.

- C/C++
  http://www.generation5.org/xornet.shtml
  http://www.netwood.net/~edwin/Matrix/
  http://www.netwood.net/~edwin/svmt/
  http://www.geocities.com/Athens/Agora/7256/c-plus-p.html
  http://www.cs.cmu.edu/afs/cs.cmu.edu/user/mitchell/ftp/faces.html
  http://www.cog.brown.edu/~rodrigo/neural_nets_library.html
  http://www.agt.net/public/bmarshal/aiparts/aiparts.htm
  http://www.geocities.com/CapeCanaveral/1624/
  http://www.neuroquest.com/ or http://www.grobe.org/LANE
  http://www.neuro-fuzzy.de/
  http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/systems/cascor/
  http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/systems/qprop/
  http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/systems/rcc/
  etc.

- Java
  http://www.philbrierley.com/code
  http://rfhs8012.fh-regensburg.de/~saj39122/jfroehl/diplom/e-index.html

http://neuron.eng.wayne.edu/software.html
http://www.aist.go.jp/NIBH/~b0616/Lab/Links.html
http://www.aist.go.jp/NIBH/~b0616/Lab/BSOM1/
http://www.neuroinformatik.ruhr-uni-bochum.de/ini/PEOPLE/loos
http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/GNG.html
http://www.isbiel.ch/I/Projects/janet/index.html
http://www.born-again.demon.nl/software.html
http://www.patol.com/java/NN/index.html
http://www-isis.ecs.soton.ac.uk/computing/neural/laboratory/laboratory.html
http://www.neuro-fuzzy.de/
http://sourceforge.net/projects/joone
http://www.geocities.com/aydingurel/neural/
http://www-eco.enst-bretagne.fr/~phan/emergence/complexe/neuron/mlp.html

- FORTRAN
  http://www.philbrierley.com/code
  http://www.cranfield.ac.uk/public/me/fo941992/mlpcode.htm

- Pascal
  http://www.ibrtses.com/delphi/neuralnets.html

If you are using a small computer (PC, Mac, etc.) you may want to have a look at the Central Neural System Electronic Bulletin Board (see question "Other sources of information"). There are lots of small simulator packages. Some of the CNS materials can also be found at http://www.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/areas/neural/cns/0.html

---

Subject: Freeware and shareware packages for NN simulation?

Since the FAQ maintainer works for a software company, he does not recommend or evaluate software in the FAQ. The descriptions below are provided by the developers or distributors of the software.

Note for future submissions: Please restrict product descriptions to a maximum of 60 lines of 72 characters, in either plain-text format or, preferably, HTML format. If you include the standard header (name, company, address, etc.), you need not count the header in the 60 line maximum. Please confine your HTML to features that are supported by primitive browsers, especially NCSA Mosaic 2.0; avoid tables, for example--use <pre> instead. Try to make the descriptions objective, and avoid making implicit or explicit assertions about competing products, such as "Our product is the *only* one that does so-and-so." The FAQ maintainer reserves the right to remove excessive marketing hype and to edit submissions to conform to size requirements; if he is in a good mood, he may also correct your spelling and punctuation.

The following simulators are described below:

1. JavaNNS
2. SNNS
3. PDP++
4. Rochester Connectionist Simulator
5. UCLA-SFINX

6. NeurDS
7. PlaNet (formerly known as SunNet)
8. GENESIS
9. Mactivation
10. Cascade Correlation Simulator
11. Quickprop
12. DartNet
13. Aspirin/MIGRAINES
14. ALN Workbench
15. Uts (Xerion, the sequel)
16. Multi-Module Neural Computing Environment (MUME)
17. LVQ_PAK, SOM_PAK
18. Nevada Backpropagation (NevProp)
19. Fuzzy ARTmap
20. PYGMALION
21. Basis-of-AI-NN Software
22. Matrix Backpropagation
23. BIOSIM
24. FuNeGen
25. NeuDL -- Neural-Network Description Language
26. NeoC Explorer
27. AINET
28. DemoGNG
29. Trajan 2.1 Shareware
30. Neural Networks at your Fingertips
31. NNFit
32. Nenet v1.0
33. Machine Consciousness Toolbox
34. NICO Toolkit (speech recognition)
35. SOM Toolbox for Matlab 5
36. FastICA package for MATLAB
37. NEXUS: Large-scale biological simulations
38. Netlab: Neural network software for Matlab
39. NuTank
40. Lens
41. Joone: Java Object Oriented Neural Engine
42. NV: Neural Viewer
43. EasyNN
44. Multilayer Perceptron - A Java Implementation

See also http://www.emsl.pnl.gov:2080/proj/neuron/neural/systems/shareware.html

1. ## JavaNNS: Java Neural Network Simulator

   http://www-ra.informatik.uni-tuebingen.de/forschung/JavaNNS/welcome_e.html
   JavaNNS is the successor to SNNS. JavaNNS is based on the SNNS computing kernel, but has a newly developed graphical user interface written in Java set on top of it. Hence compatibility with SNNS is achieved while platform-independence is increased.

   In addition to SNNS features, JavaNNS offers the capability of linking HTML browsers to it. This provides for accessing the user manual (available in HTML) or, optionally, a reference coursebook

on neural networks directly from within the program.

JavaNNS is available for Windows NT / Windows 2000, Solaris and RedHat Linux. Additional ports are planed. JavaNNS is freely available and can be downloaded from the URL shown above.

Contact: Igor Fischer, Phone: +49 7071 29-77176, fischer@informatik.uni-tuebingen.de

2. **SNNS 4.2**

SNNS (Stuttgart Neural Network Simulator) is a software simulator for neural networks on Unix workstations developed at the Institute for Parallel and Distributed High Performance Systems (IPVR) at the University of Stuttgart. The goal of the SNNS project is to create an efficient and flexible simulation environment for research on and application of neural nets.

The SNNS simulator consists of two main components:

1. simulator kernel written in C
2. graphical user interface under X11R4 or X11R5

The simulator kernel operates on the internal network data structures of the neural nets and performs all operations of learning and recall. It can also be used without the other parts as a C program embedded in custom applications. It supports arbitrary network topologies and, like RCS, supports the concept of sites. SNNS can be extended by the user with user defined activation functions, output functions, site functions and learning procedures, which are written as simple C programs and linked to the simulator kernel. C code can be generated from a trained network.

Currently the following network architectures and learning procedures are included:

- Backpropagation (BP) for feedforward networks
  - vanilla (online) BP
  - BP with momentum term and flat spot elimination
  - batch BP
  - chunkwise BP
- Counterpropagation
- Quickprop
- Backpercolation 1
- RProp
- Generalized radial basis functions (RBF)
- ART1
- ART2
- ARTMAP
- Cascade Correlation
- Dynamic LVQ
- Backpropagation through time (for recurrent networks)
- Quickprop through time (for recurrent networks)
- Self-organizing maps (Kohonen maps)
- TDNN (time-delay networks) with Backpropagation
- Jordan networks
- Elman networks and extended hierarchical Elman networks
- Associative Memory
- TACOMA

The graphical user interface XGUI (X Graphical User Interface), built on top of the kernel, gives a 2D and a 3D graphical representation of the neural networks and controls the kernel during the simulation run. In addition, the 2D user interface has an integrated network editor which can be used to directly create, manipulate and visualize neural nets in various ways.

SNNSv4.1 has been tested on SUN SparcSt ELC,IPC (SunOS 4.1.2, 4.1.3), SUN SparcSt 2 (SunOS 4.1.2), SUN SparcSt 5, 10, 20 (SunOS 4.1.3, 5.2), DECstation 3100, 5000 (Ultrix V4.2), DEC Alpha AXP 3000 (OSF1 V2.1), IBM-PC 80486, Pentium (Linux), IBM RS 6000/320, 320H, 530H (AIX V3.1, AIX V3.2), HP 9000/720, 730 (HP-UX 8.07), and SGI Indigo 2 (IRIX 4.0.5, 5.3).

The distributed kernel can spread one learning run over a workstation cluster.

SNNS web page: http://www-ra.informatik.uni-tuebingen.de/SNNS
Ftp server: ftp://ftp.informatik.uni-tuebingen.de/pub/SNNS

- SNNSv4.1.Readme
- SNNSv4.1.tar.gz (1.4 MB, Source code)
- SNNSv4.1.Manual.ps.gz (1 MB, Documentation)

Mailing list: http://www-ra.informatik.uni-tuebingen.de/SNNS/about-ml.html

3. **PDP++**

URL: http://www.cnbc.cmu.edu/PDP++/PDP++.html

The PDP++ software is a neural-network simulation system written in C++. It represents the next generation of the PDP software released with the McClelland and Rumelhart "Explorations in Parallel Distributed Processing Handbook", MIT Press, 1987. It is easy enough for novice users, but very powerful and flexible for research use. PDP++ is featured in a new textbook, *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain,* by Randall C. O'Reilly and Yuko Munakata, MIT Press, 2000.

Supported algorithms include:

- Feedforward and recurrent error backpropagation. Recurrent BP includes continuous, real-time models, and Almeida-Pineda.
- Constraint satisfaction algorithms and associated learning algorithms including Boltzmann Machine, Hopfield models, mean-field networks (DBM), Interactive Activation and Competition (IAC), and continuous stochastic networks.
- Self-organizing learning including Competitive Learning, Soft Competitive Learning, simple Hebbian, and Self-organizing Maps ("Kohonen Nets").
- Mixtures-of-experts using backpropagation experts, EM updating, and a SoftMax gating module.
- Leabra algorithm that combines error-driven and Hebbian learning with k-Winners-Take-All inhibitory competition.

The software can be obtained by anonymous ftp from:

- ftp://grey.colorado.edu/pub/oreilly/pdp++ or
- ftp://cnbc.cmu.edu/pub/pdp++/ or
- ftp://unix.hensa.ac.uk/mirrors/pdp++/

4. **Rochester Connectionist Simulator**

A versatile simulator program for arbitrary types of neural nets. Comes with a backprop package and a X11/Sunview interface. Available via anonymous FTP from ftp://ftp.cs.rochester.edu /pub/packages/simulator/simulator_v4.2.tar.Z There's also a patch available from ftp://ftp.cs.rochester.edu/pub/packages/simulator/simulator_v4.2.patch.1

5. **UCLA-SFINX**

   The UCLA-SFINX, a "neural" network simulator is now in public domain. UCLA-SFINX (Structure and Function In Neural connec- tions) is an interactive neural network simulation environment designed to provide the investigative tools for studying the behavior of various neural structures. It was designed to easily express and simulate the highly regular patterns often found in large networks, but it is also general enough to model parallel systems of arbitrary interconnectivity. For more information, see http://decus.acornsw.com/vs0121/AISIG/F90/NETS /UCLA_SIM.TXT

6. **NeurDS**

   Neural Design and Simulation System. This is a general purpose tool for building, running and analysing Neural Network Models in an efficient manner. NeurDS will compile and run virtually any Neural Network Model using a consistent user interface that may be either window or "batch" oriented. HP-UX 8.07 source code is available from http://hpux.u-aizu.ac.jp/hppd/hpux/NeuralNets /NeurDS-3.1/ or http://askdonna.ask.uni-karlsruhe.de/hppd/hpux/NeuralNets/NeurDS-3.1/

7. **PlaNet5.7 (formerly known as SunNet)**

   A popular connectionist simulator with versions to run under X Windows, and non-graphics terminals created by Yoshiro Miyata (Chukyo Univ., Japan). 60-page User's Guide in Postscript. Send any questions to miyata@sccs.chukyo-u.ac.jp Available for anonymous ftp from ftp.ira.uka.de as /pub/neuron/PlaNet5.7.tar.gz (800 kb)

8. **GENESIS**

   GENESIS 2.0 (GEneral NEural SImulation System) is a general purpose simulation platform which was developed to support the simulation of neural systems ranging from complex models of single neurons to simulations of large networks made up of more abstract neuronal components. Most current GENESIS applications involve realistic simulations of biological neural systems. Although the software can also model more abstract networks, other simulators are more suitable for backpropagation and similar connectionist modeling. Runs on most Unix platforms. Graphical front end XODUS. Parallel version for networks of workstations, symmetric multiprocessors, and MPPs also available. Further information via WWW at http://www.genesis-sim.org/GENESIS/.

9. **Mactivation**

   A neural network simulator for the Apple Macintosh. Available for ftp from ftp.cs.colorado.edu as /pub/cs/misc/Mactivation-3.3.sea.hqx

10. **Cascade Correlation Simulator**

    A simulator for Scott Fahlman's Cascade Correlation algorithm. Available for ftp from ftp.cs.cmu.edu in directory /afs/cs/project/connect/code/supported as the file cascor-v1.2.shar (223 KB) There is also a version of recurrent cascade correlation in the same directory in file rcc1.c (108

KB).

11. Quickprop

A variation of the back-propagation algorithm developed by Scott Fahlman. A simulator is available in the same directory as the cascade correlation simulator above in file nevprop1.16.shar (137 KB)
(There is also an obsolete simulator called quickprop1.c (21 KB) in the same directory, but it has been superseeded by NevProp. See also the description of NevProp below.)

12. DartNet

DartNet is a Macintosh-based backpropagation simulator, developed at Dartmouth by Jamshed Bharucha and Sean Nolan as a pedagogical tool. It makes use of the Mac's graphical interface, and provides a number of tools for building, editing, training, testing and examining networks. This program is available by anonymous ftp from ftp.dartmouth.edu as /pub/mac/dartnet.sit.hqx (124 KB).

13. Aspirin/MIGRAINES

Aspirin/MIGRAINES 6.0 consists of a code generator that builds neural network simulations by reading a network description (written in a language called "Aspirin") and generates a C simulation. An interface (called "MIGRAINES") is provided to export data from the neural network to visualization tools. The system has been ported to a large number of platforms. The goal of Aspirin is to provide a common extendible front-end language and parser for different network paradigms. The MIGRAINES interface is a terminal based interface that allows you to open Unix pipes to data in the neural network. Users can display the data using either public or commercial graphics/analysis tools. Example filters are included that convert data exported through MIGRAINES to formats readable by Gnuplot 3.0, Matlab, Mathematica, and xgobi.

The software is available from http://www.elegant-software.com/software/aspirin/

14. ALN Workbench (a spreadsheet for Windows)

ALNBench is a free spreadsheet program for MS-Windows (NT, 95) that allows the user to import training and test sets and predict a chosen column of data from the others in the training set. It is an easy-to-use program for research, education and evaluation of ALN technology. Anyone who can use a spreadsheet can quickly understand how to use it. It facilitates interactive access to the power of the Dendronic Learning Engine (DLE), a product in commercial use.

An ALN consists of linear functions with adaptable weights at the leaves of a tree of maximum and minimum operators. The tree grows automatically during training: a linear piece splits if its error is too high. The function computed by an ALN is piecewise linear and continuous. It can learn to approximate any continuous function to arbitrarily high accuracy.

Parameters allow the user to input knowledge about a function to promote good generalization. In particular, bounds on the weights of the linear functions can be directly enforced. Some parameters are chosen automatically in standard mode, and are under user control in expert mode.

The program can be downloaded from http://www.dendronic.com/main.htm

For further information please contact:

```
William W. Armstrong PhD, President
Dendronic Decisions Limited
3624 - 108 Street, NW
Edmonton, Alberta,
Canada T6J 1B4
Email: arms@dendronic.com
URL: http://www.dendronic.com/
Tel. +1 403 421 0800
(Note: The area code 403 changes to 780 after Jan. 25, 1999)
```

15. Uts (Xerion, the sequel)

Uts is a portable artificial neural network simulator written on top of the Tool Control Language (Tcl) and the Tk UI toolkit. As result, the user interface is readily modifiable and it is possible to simultaneously use the graphical user interface and visualization tools and use scripts written in Tcl. Uts itself implements only the connectionist paradigm of linked units in Tcl and the basic elements of the graphical user interface. To make a ready-to-use package, there exist modules which use Uts to do back-propagation (tkbp) and mixed em gaussian optimization (tkmxm). Uts is available in ftp.cs.toronto.edu in directory /pub/xerion.

16. Multi-Module Neural Computing Environment (MUME)

MUME is a simulation environment for multi-modules neural computing. It provides an object oriented facility for the simulation and training of multiple nets with various architectures and learning algorithms. MUME includes a library of network architectures including feedforward, simple recurrent, and continuously running recurrent neural networks. Each architecture is supported by a variety of learning algorithms. MUME can be used for large scale neural network simulations as it provides support for learning in multi-net environments. It also provide pre- and post-processing facilities. For more information, see http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/systems/mume/0.html

17. LVQ_PAK, SOM_PAK

These are packages for Learning Vector Quantization and Self-Organizing Maps, respectively. They have been built by the LVQ/SOM Programming Team of the Helsinki University of Technology, Laboratory of Computer and Information Science, Rakentajanaukio 2 C, SF-02150 Espoo, FINLAND There are versions for Unix and MS-DOS available from http://nucleus.hut.fi/nnrc/nnrc-programs.html

18. Nevada Backpropagation (NevProp)

NevProp, version 3, is a relatively easy-to-use, feedforward backpropagation multilayer perceptron simulator-that is, statistically speaking, a multivariate nonlinear regression program. NevProp3 is distributed for free under the terms of the GNU Public License and can be downloaded from http://brain.cs.unr.edu/publications/NevProp.zip and http://brain.cs.unr.edu/publications/NevPropManual.pdf

The program is distributed as C source code that should compile and run on most platforms. In addition, precompiled executables are available for Macintosh and DOS platforms. Limited support is available from Phil Goodman (goodman@unr.edu), University of Nevada Center for Biomedical Research.

MAJOR FEATURES OF NevProp3 OPERATION (* indicates feature new in version 3)

1. Character-based interface common to the UNIX, DOS, and Macintosh platforms.
2. Command-line argument format to efficiently initiate NevProp3. For Generalized Nonlinear Modeling (GNLM) mode, beginners may opt to use an interactive interface.
3. Option to pre-standardize the training data (z-score or forced range*).
4. Option to pre-impute missing elements in training data (case-wise deletion, or imputation with mean, median, random selection, or k-nearest neighbor).*
5. Primary error (criterion) measures include mean square error, hyperbolic tangent error, and log likelihood (cross-entropy), as penalized an unpenalized values.
6. Secondary measures include ROC-curve area (c-index), thresholded classification, R-squared and Nagelkerke R-squared. Also reported at intervals are the weight configuration, and the sum of square weights.
7. Allows simultaneous use of logistic (for dichotomous outputs) and linear output activation functions (automatically detected to assign activation and error function).*
8. 1-of-N (Softmax)* and M-of-N options for binary classification.
9. Optimization options: flexible learning rate (fixed global adaptive, weight-specific, quickprop), split learn rate (inversely proportional to number of incoming connections), stochastic (case-wise updating), sigmoidprime offset (to prevent locking at logistic tails).
10. Regularization options: fixed weight decay, optional decay on bias weights, Bayesian hyperpenalty* (partial and full Automatic Relevance Determination-also used to select important predictors), automated early stopping (full dataset stopping based on multiple subset cross-validations) by error criterion.
11. Validation options: upload held-out validation test set; select subset of outputs for joint summary statistics;* select automated bootstrapped modeling to correct optimistically biased summary statistics (with standard deviations) without use of hold-out.
12. Saving predictions: for training data and uploaded validation test set, save file with identifiers, true targets, predictions, and (if bootstrapped models selected) lower and upper 95% confidence limits* for each prediction.
13. Inference options: determination of the mean predictor effects and level effects (for multilevel predictor variables); confidence limits within main model or across bootstrapped models.*
14. ANN-kNN (k-nearest neighbor) emulation mode options: impute missing data elements and save to new data file; classify test data (with or without missing elements) using ANN-kNN model trained on data with or without missing elements (complete ANN-based expectation maximization).*
15. AGE (ANN-Gated Ensemble) options: adaptively weight predictions (any scale of scores) obtained from multiple (human or computational) "experts"; validate on new prediction sets; optional internal prior-probability expert.*

19. Fuzzy ARTmap

This is just a small example program. Available for anonymous ftp from park.bu.edu [128.176.121.56] ftp://cns-ftp.bu.edu/pub/fuzzy-artmap.tar.Z (44 kB).

20. PYGMALION

This is a prototype that stems from an ESPRIT project. It implements back-propagation, self organising map, and Hopfield nets. Avaliable for ftp from ftp.funet.fi [128.214.248.6] as /pub/sci/neural/sims/pygmalion.tar.Z (1534 kb). (Original site is imag.imag.fr: archive/pygmalion/pygmalion.tar.Z).

21. Basis-of-AI-NN Software

Non-GUI DOS and UNIX source code, DOS binaries and examples are available in the following different program sets and the backprop package has a Windows 3.x binary and a Unix/Tcl/Tk version:

```
[backprop, quickprop, delta-bar-delta, recurrent networks],
[simple clustering, k-nearest neighbor, LVQ1, DSM],
[Hopfield, Boltzman, interactive activation network],
[interactive activation network],
[feedforward counterpropagation],
[ART I],
[a simple BAM] and
[the linear pattern classifier]
```

For details see: http://www.dontveter.com/nnsoft/nnsoft.html

An improved professional version of backprop is also available; see Part 6 of the FAQ.

Questions to: Don Tveter, don@dontveter.com

22. Matrix Backpropagation

MBP (Matrix Back Propagation) is a very efficient implementation of the back-propagation algorithm for current-generation workstations. The algorithm includes a per-epoch adaptive technique for gradient descent. All the computations are done through matrix multiplications and make use of highly optimized C code. The goal is to reach almost peak-performances on RISCs with superscalar capabilities and fast caches. On some machines (and with large networks) a 30-40x speed-up can be measured with respect to conventional implementations. The software is available by anonymous ftp from ftp.esng.dibe.unige.it as /neural/MBP/MBPv1.1.tar.Z (Unix version), or /neural/MBP/MBPv11.zip (PC version)., For more information, contact Davide Anguita (anguita@dibe.unige.it).

23. BIOSIM

BIOSIM is a biologically oriented neural network simulator. Public domain, runs on Unix (less powerful PC-version is available, too), easy to install, bilingual (german and english), has a GUI (Graphical User Interface), designed for research and teaching, provides online help facilities, offers controlling interfaces, batch version is available, a DEMO is provided.

REQUIREMENTS (Unix version): X11 Rel. 3 and above, Motif Rel 1.0 and above, 12 MB of physical memory, recommended are 24 MB and more, 20 MB disc space. REQUIREMENTS (PC version): PC-compatible with MS Windows 3.0 and above, 4 MB of physical memory, recommended are 8 MB and more, 1 MB disc space.

Four neuron models are implemented in BIOSIM: a simple model only switching ion channels on and off, the original Hodgkin-Huxley model, the SWIM model (a modified HH model) and the Golowasch-Buchholz model. Dendrites consist of a chain of segments without bifurcation. A neural network can be created by using the interactive network editor which is part of BIOSIM. Parameters can be changed via context sensitive menus and the results of the simulation can be visualized in observation windows for neurons and synapses. Stochastic processes such as noise can be included. In addition, biologically orientied learning and forgetting processes are modeled, e.g. sensitization, habituation, conditioning, hebbian learning and competitive learning. Three synaptic

types are predefined (an excitatatory synapse type, an inhibitory synapse type and an electrical synapse). Additional synaptic types can be created interactively as desired.

Available for ftp from ftp.uni-kl.de in directory /pub/bio/neurobio: Get /pub/bio/neurobio/biosim.readme (2 kb) and /pub/bio/neurobio/biosim.tar.Z (2.6 MB) for the Unix version or /pub/bio/neurobio/biosimpc.readme (2 kb) and /pub/bio/neurobio/biosimpc.zip (150 kb) for the PC version.

Contact:
Stefan Bergdoll
Department of Software Engineering (ZXA/US)
BASF Inc.
D-67056 Ludwigshafen; Germany
bergdoll@zxa.basf-ag.de phone 0621-60-21372 fax 0621-60-43735

24. FuNeGen 1.0

FuNeGen is a MLP based software program to generate fuzzy rule based classifiers. For more information, see http://www-2.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/areas/fuzzy/systems/funegen/

25. NeuDL -- Neural-Network Description Language

NeuDL is a description language for the design, training, and operation of neural networks. It is currently limited to the backpropagation neural-network model; however, it offers a great deal of flexibility. For example, the user can explicitly specify the connections between nodes and can create or destroy connections dynamically as training progresses. NeuDL is an interpreted language resembling C or C++. It also has instructions dealing with training/testing set manipulation as well as neural network operation. A NeuDL program can be run in interpreted mode or it can be automatically translated into C++ which can be compiled and then executed. The NeuDL interpreter is written in C++ and can be easly extended with new instructions. For more information, see http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/neural/systems/neudl/0.html

26. NeoC Explorer (Pattern Maker included)

The NeoC software is an implementation of Fukushima's Neocognitron neural network. Its purpose is to test the model and to facilitate interactivity for the experiments. Some substantial features: GUI, explorer and tester operation modes, recognition statistics, performance analysis, elements displaying, easy net construction. PLUS, a pattern maker utility for testing ANN: GUI, text file output, transformations. For more information, see http://www.simtel.net/pub/pd/39893.html

27. AINET

AINET is a probabilistic neural network application which runs on Windows 95/NT. It was designed specifically to facilitate the modeling task in all neural network problems. It is lightning fast and can be used in conjunction with many different programming languages. It does not require iterative learning, has no limits in variables (input and output neurons), no limits in sample size. It is not sensitive toward noise in the data. The database can be changed dynamically. It provides a way to estimate the rate of error in your prediction. It has a graphical spreadsheet-like user interface. The AINET manual (more than 100 pages) is divided into: "User's Guide", "Basics About Modeling with the AINET", "Examples", "The AINET DLL library" and "Appendix" where the

theoretical background is revealed. You can get a full working copy from: http://www.ainet-sp.si/

28. DemoGNG

This simulator is written in Java and should therefore run without compilation on all platforms where a Java interpreter (or a browser with Java support) is available. It implements the following algorithms and neural network models:

- Hard Competitive Learning (standard algorithm)
- Neural Gas (Martinetz and Schulten 1991)
- Competitive Hebbian Learning (Martinetz and Schulten 1991, Martinetz 1993)
- Neural Gas with Competitive Hebbian Learning (Martinetz and Schulten 1991)
- Growing Neural Gas (Fritzke 1995)

DemoGNG is distributed under the GNU General Public License. It allows to experiment with the different methods using various probability distributions. All model parameters can be set interactively on the graphical user interface. A teach modus is provided to observe the models in "slow-motion" if so desired. It is currently **not** possible to experiment with user-provided data, so the simulator is useful basically for demonstration and teaching purposes and as a sample implementation of the above algorithms.

DemoGNG can be accessed most easily at http://www.neuroinformatik.ruhr-uni-bochum.de/ in the file /ini/VDM/research/gsn/DemoGNG/GNG.html where it is embedded as Java applet into a Web page and is downloaded for immediate execution when you visit this page. An accompanying paper entitled "Some competitive learning methods" describes the implemented models in detail and is available in html at the same server in the directory ini/VDM/research/gsn/JavaPaper/.

It is also possible to download the complete source code and a Postscript version of the paper via anonymous ftp from ftp.neuroinformatik.ruhr-uni-bochum [134.147.176.16] in directory /pub/software/NN/DemoGNG/. The software is in the file DemoGNG-1.00.tar.gz (193 KB) and the paper in the file sclm.ps.gz (89 KB). There is also a README file (9 KB). Please send any comments and questions to demogng@neuroinformatik.ruhr-uni-bochum.de which will reach Hartmut Loos who has written DemoGNG as well as Bernd Fritzke, the author of the accompanying paper.

29. Trajan 2.1 Shareware

Trajan 2.1 Shareware is a Windows-based Neural Network simulation package. It includes support for the two most popular forms of Neural Network: Multilayer Perceptrons with Back Propagation and Kohonen networks.

Trajan 2.1 Shareware concentrates on ease-of-use and feedback. It includes Graphs, Bar Charts and Data Sheets presenting a range of Statistical feedback in a simple, intuitive form. It also features extensive on-line Help.

The Registered version of the package can support very large networks (up to 128 layers with up to 8,192 units each, subject to memory limitations in the machine), and allows simple Cut and Paste transfer of data to/from other Windows-packages, such as spreadsheet programs. The Unregistered version features limited network size and no Clipboard Cut-and-Paste.

There is also a Professional version of Trajan 2.1, which supports a wider range of network models, training algorithms and other features.

See Trajan Software's Home Page at http://www.trajan-software.demon.co.uk for further details,

and a free copy of the Shareware version.

Alternatively, email andrew@trajan-software.demon.co.uk for more details.

30. Neural Networks at your Fingertips

"Neural Networks at your Fingertips" is a package of ready-to-reuse neural network simulation source code which was prepared for educational purposes by Karsten Kutza. The package consists of eight programs, each of which implements a particular network architecture together with an embedded example application from a typical application domain.
Supported network architectures are

- Adaline,
- Backpropagation,
- Hopfield Model,
- Bidirectional Associative Memory,
- Boltzmann Machine,
- Counterpropagation,
- Self-Organizing Map, and
- Adaptive Resonance Theory.

The applications demonstrate use of the networks in various domains such as pattern recognition, time-series forecasting, associative memory, optimization, vision, and control and include e.g. a sunspot prediction, the traveling salesman problem, and a pole balancer.
The programs are coded in portable, self-contained ANSI C and can be obtained from the web pages at http://www.geocities.com/CapeCanaveral/1624.

31. NNFit

NNFit (Neural Network data Fitting) is a user-friendly software that allows the development of empirical correlations between input and output data. Multilayered neural models have been implemented using a quasi-newton method as learning algorithm. Early stopping method is available and various tables and figures are provided to evaluate fitting performances of the neural models. The software is available for most of the Unix platforms with X-Windows (IBM-AIX, HP-UX, SUN, SGI, DEC, Linux). Informations, manual and executable codes (english and french versions) are available at http://www.gch.ulaval.ca/~nnfit
Contact: Bernard P.A. Grandjean, department of chemical engineering,
Laval University; Sainte-Foy (Quibec) Canada G1K 7P4;
grandjean@gch.ulaval.ca

32. Nenet v1.0

Nenet v1.0 is a 32-bit Windows 95 and Windows NT 4.0 application designed to facilitate the use of a Self-Organizing Map (SOM) algorithm.

The major motivation for Nenet was to create a user-friendly SOM algorithm tool with good visualization capabilities and with a GUI allowing efficient control of the SOM parameters. The use scenarios have stemmed from the user's point of view and a considerable amount of work has been placed on the ease of use and versatile visualization methods.

With Nenet, all the basic steps in map control can be performed. In addition, Nenet also includes some more exotic and involved features especially in the area of visualization.

Features in Nenet version 1.0:

- Implements the standard Kohonen SOM algorithm
- Supports 2 common data preprocessing methods
- 5 different visualization methods with rectangular or hexagonal topology
- Capability to animate both train and test sequences in all visualization methods
- Labelling
  - Both neurons and parameter levels can be labelled
  - Provides also autolabelling
- Neuron values can be inspected easily
- Arbitrary selection of parameter levels can be visualized with Umatrix simultaneously
- Multiple views can be opened on the same map data
- Maps can be printed
- Extensive help system provides fast and accurate online help
- SOM_PAK compatible file formats
- Easy to install and uninstall
- Conforms to the common Windows 95 application style - all functionality in one application

Nenet web site is at: http://www.mbnet.fi/~phodju/nenet/Nenet/General.html The web site contains further information on Nenet and also the downloadable Nenet files (3 disks totalling about 3 Megs)

If you have any questions whatsoever, please contact: Nenet-Team@hut.fi or phassine@cc.hut.fi

33. Machine Consciousness Toolbox

See listing for Machine Consciousness Toolbox in part 6 of the FAQ.

34. NICO Toolkit (speech recognition)

```
      Name: NICO Artificial Neural Network Toolkit
    Author: Nikko Strom
   Address: Speech, Music and Hearing, KTH, S-100 44, Stockholm, Sweden
     Email: nikko@speech.kth.se
       URL: http://www.speech.kth.se/NICO/index.html
 Platforms: UNIX, ANSI C; Source code tested on: HPUX, SUN Solaris, Linux
     Price: Free
```

The NICO Toolkit is an artificial neural network toolkit designed and optimized for automatic speech recognition applications. Networks with both recurrent connections and time-delay windows are easily constructed. The network topology is very flexible -- any number of layers is allowed and layers can be arbitrarily connected. Sparse connectivity between layers can be specified. Tools for extracting input-features from the speech signal are included as well as tools for computing target values from several standard phonetic label-file formats.

Algorithms:

- Back-propagation through time,
- Speech feature extraction (Mel cepstrum coefficients, filter-bank)

35. SOM Toolbox for Matlab 5

SOM Toolbox, a shareware Matlab 5 toolbox for data analysis with self-organizing maps is available at the URL http://www.cis.hut.fi/projects/somtoolbox/. If you are interested in practical data analysis and/or self-organizing maps and have Matlab 5 in your computer, be sure to check this

out!

Highlights of the SOM Toolbox include the following:

- Tools for all the stages of data analysis: besides the basic SOM training and visualization tools, the package includes also tools for data preprocessing and model validation and interpretation.
- Graphical user interface (GUI): the GUI first guides the user through the initialization and training procedures, and then offers a variety of different methods to visualize the data on the trained map.
- Modular programming style: the Toolbox code utilizes Matlab structures, and the functions are constructed in a modular manner, which makes it convenient to tailor the code for each user's specific needs.
- Advanced graphics: building on the Matlab's strong graphics capabilities, attractive figures can be easily produced.
- Compatibility with SOM_PAK: import/export functions for SOM_PAK codebook and data files are included in the package.
- Component weights and names: the input vector components may be given different weights according to their relative importance, and the components can be given names to make the figures easier to read.
- Batch or sequential training: in data analysis applications, the speed of training may be considerably improved by using the batch version.
- Map dimension: maps may be N-dimensional (but visualization is not supported when N > 2).

36. FastICA package for MATLAB

The FastICA algorithm for independent component analysis.

Independent component analysis, or ICA, is neural network or signal processing technique that represents a multidimensional random vector as a linear combination of nongaussian random variables ('independent components') that are as independent as possible. ICA is a nongaussian version of factor analysis, and somewhat similar to principal component analysis. ICA has many applications in data analysis, source separation, and feature extraction.

The FastICA algorithm is a computationally optimized method for performing the estimation of ICA. It uses a fixed-point iteration scheme that has been found in independent experiments to be 10-100 times faster than conventional gradient descent methods for ICA. Another advantage of the FastICA algorithm is that it can be used to estimate the independent components one-by-one, as in projection pursuit, which is very practical in exploratory data analysis.

The FastICA package for MATLAB (versions 5 or 4) is freeware package with a graphical user interface that implements the fixed-point algorithm for ICA. The package is available on the Web at http://www.cis.hut.fi/projects/ica/fastica/.
Email contact: Aapo Hyvarinen <Aapo.Hyvarinen@hut.fi>

37. NEXUS: Large-scale biological simulations

Large-scale biological neural network simulation engine. Includes automated network construction tool that allows extremely complex networks to be generated according to user-supplied architectural specifications.

The network engine is an attempt at creating a biological neural network simulator. It consists of a C++ class, called "network". A network object houses a set of objects of another C++ class, called "neuron". The neuron class is a detailed functional simulation of a neuron (i.e. the actual chemical processes that lead to a biological neuron's behavior are not modeled explicitly, but the behavior itself is). The simulation of the neuron is handled entirely by the neuron class. The network class coordinates the functioning of the neurons that make up the neural network, as well as providing addressing services that allow the neurons to interact. It is also responsible for facilitating the interface of the neural network it houses onto any existing software into which the neural network is to be integrated.

Since a simulated neural network consisting of a large number of heavily interconnected neurons is extremely difficult to generate manually, NEXUS was developed. To create a network with NEXUS, one need only describe the network in general terms, in terms of groups of sets of specifically arranged neurons, and how the groups interface onto each other and onto themselves. This information constitutes a network architecture descriptor. A network architecture descriptor is read by NEXUS, and NEXUS uses the information to generate a network, building all the neurons and connecting them together appropriately. This system is analogous to nature's brain construction system. For example, human brains, in general, are very similar. The basic design is stored in human DNA. Since it is certainly not possible to record information about each neuron and its connections, DNA must instead contain (in some form) what is essentially a set of guidelines, a set of rules about how the brain is to be laid out. These guidelines are used to build the brain, just like NEXUS uses the guidelines set out in the network architecture descriptor to build the simulated neural network.

NEXUS and the network engine have deliberately been engineered to be highly efficient and very compact. Even so, large, complex networks require tremendous amounts of memory and processing power.

The network engine:

- flexible and elegant design; highly customizable simulation parameters; extremely efficient
- throughout, nonlinear magnitude decay modeling
- dendritic tree complexity and network connection density limited only by the computer hardware
- simulation of dendritic logic gate behaviors via a sophisticated excitation thresholding and conduction model
- detailed simulation of backprop, allowing realistic simulation of associated memory formation processes
- simulation of all known postsynaptic memory formation mechanisms (STP, STD, LTP, LTD)
- dynamic presynaptic output pattern modeling, including excitation magnitude dependent output pattern selection
- simulation of all known presynaptic activity-based output modifiers (PPF, PTP, depression)

NEXUS:

- allows networks to be designed concisely and as precisely as is necessary
- makes massively complex large-scale neural network design and construction possible
- allows existing networks to be augmented without disturbing existing network structure
- UNIX and Win32 compatible

URL: http://www.sfu.ca/~loryan/neural.html
Email: Lawrence O. Ryan <loryan@sfu.ca>

38. Netlab: Neural network software for Matlab

http://www.ncrg.aston.ac.uk/netlab/index.html

The Netlab simulation software is designed to provide the central tools necessary for the simulation of theoretically well founded neural network algorithms for use in teaching, research and applications development. It consists of a library of Matlab functions and scripts based on the approach and techniques described in Neural Networks for Pattern Recognition by Christopher M. Bishop, (Oxford University Press, 1995). The functions come with on-line help, and further explanation is available via HTML files.

The Netlab library includes software implementations of a wide range of data analysis techniques. Netlab works with Matlab version 5.0 and higher. It is not compatible with earlier versions of Matlab.

39. NuTank

NuTank stands for NeuralTank. It is educational and entertainment software. In this program one is given the shell of a 2 dimentional robotic tank. The tank has various I/O devices like wheels, whiskers, optical sensors, smell, fuel level, sound and such. These I/O sensors are connected to Neurons. The player/designer uses more Neurons to interconnect the I/O devices. One can have any level of complexity desired (memory limited) and do subsumptive designs. More complex design take slightly more fuel, so life is not free. All movement costs fuel too. One can also tag neuron connections as "adaptable" that adapt their weights in acordance with the target neuron. This allows neurons to learn. The Neuron editor can handle 3 dimention arrays of neurons as single entities with very flexible interconect patterns.

One can then design a scenario with walls, rocks, lights, fat (fuel) sources (that can be smelled) and many other such things. Robot tanks are then introduced into the Scenario and allowed interact or battle it out. The last one alive wins, or maybe one just watches the motion of the robots for fun. While the scenario is running it can be stopped, edited, zoom'd, and can track on any robot.

The entire program is mouse and graphicly based. It uses DOS and VGA and is written in TurboC++. There will also be the ability to download designs to another computer and source code will be available for the core neural simulator. This will allow one to design neural systems and download them to real robots. The design tools can handle three dimentional networks so will work with video camera inputs and such.

NuTank source code is free from http://www.xmission.com/~rkeene/NuTankSrc.ZIP
Contact: Richard Keene; Keene Educational Software
Email: rkeene@xmission.com or r.keene@center7.com

40. Lens

http://www.cs.cmu.edu/~dr/Lens

Lens (the light, efficient network simulator) is a fast, flexible, and customizable neural network package written primarily in C. It currently handles standard backpropagation networks, simple recurrent (including Jordan and Elman) and fully recurrent nets, deterministic Boltzmann machines, self-organizing maps, and interactive-activation models.

Lens runs under Windows as well as a variety of Unix platforms. It includes a graphical interface

and an embedded script language (Tcl). The key to the speed of Lens is its use of tight inner-loops that minimize memory references when traversing links. Frequently accessed values are stored in contiguous memory to achieve good cache performance. It is also able to do batch-level parallel training on multiple processors.

Because it is recognized that no simulator will satisfy sophisticated users out of the box, Lens was designed to facilitate code modification. Users can create and register such things as new network or group types, new weight update algorithms, or new shell commands without altering the main body of code. Therefore, modifications can be easily transferred to new releases.

Lens is available free-of-charge to those conducting research at academic or non-profit institutions. Other users should contact Douglas Rohde for licensing information at dr+lens@cs.cmu.edu.

41. Joone: Java Object Oriented Neural Engine

http://sourceforge.net/projects/joone

Joone is a neural net engine written in Java. It's a modular, scalable, multitasking and extensible engine. It can be extended by writing new modules to implement new algorithms or new architectures starting from simple base components. It's an Open Source project and everybody can contribute to its development.

Contact: Paolo Marrone, paolo@marrone.org

42. NV: Neural Viewer

http://www.btinternet.com/~cfinnie/

A free software application for modelling and visualizing complex recurrent neural networks in 3D.

43. EasyNN

URL: http://www.easynn.com/

EasyNN is a neural network system for Microsoft Windows. It can generate multi layer neural networks from text files or grids with minimal user intervention. The networks can then be trained, validated and queried. Network diagrams, graphs, input/output data and all the network details can be displayed and printed. Nodes can be added or deleted while the network is learning. The graph, grid, network and detail displays are updated dynamically so you can see how the neural networks work. EasyNN runs on Windows 95, 98, ME, NT 4.0, 2000 or XP.

44. Multilayer Perceptron - A Java Implementation

Download java from: http://www.geocities.com/aydingurel/neural/

What can you exactly do with it? You can:

- Build nets with any number of layers and units. Layers are connected to each other consecutively, each unit in a layer is connected to all of the units on the next layer (and vice versa) if there is one,
- Set units with linear and sigmoid activation functions and set them separately for each layer,
- Set parameters for sigmoid functions and set them separately for each layer,

- Use momentum, set different momentum parameters for each layer,
- Initialize the net using your own set of weights,
- Train the net using backpropagation and with any training rate.

Contact: Aydin Gurel, aydin.gurel@lycos.com

For some of these simulators there are user mailing lists. Get the packages and look into their documentation for further info.

Next part is part 6 (of 7). Previous part is part 4.