```
Archive-name: ai-faq/neural-nets/part1
Last-modified: 2002-05-17
URL: ftp://ftp.sas.com/pub/neural/FAQ.html
Maintainer: saswss@unx.sas.com (Warren S. Sarle)
```



```
  ------------------------------------------------------------
    Additions, corrections, or improvements are always welcome.
    Anybody who is willing to contribute any information,
    please email me; if it is relevant, I will incorporate it.

    The monthly posting departs around the 28th of every month.
  ------------------------------------------------------------
```

This is the first of seven parts of a monthly posting to the Usenet newsgroup comp.ai.neural-nets (as well as comp.answers and news.answers, where it should be findable at any time). Its purpose is to provide basic information for individuals who are new to the field of neural networks or who are just beginning to read this group. It will help to avoid lengthy discussion of questions that often arise for beginners.

```
   SO, PLEASE, SEARCH THIS POSTING FIRST IF YOU HAVE A QUESTION
                           and
   DON'T POST ANSWERS TO FAQs: POINT THE ASKER TO THIS POSTING
```

The latest version of the FAQ is available as a hypertext document, readable by any WWW (World Wide Web) browser such as Netscape, under the URL: ftp://ftp.sas.com/pub/neural/FAQ.html.

If you are reading the version of the FAQ posted in comp.ai.neural-nets, be sure to view it with a monospace font such as Courier. If you view it with a proportional font, tables and formulas will be mangled. Some newsreaders or WWW news services garble plain text. If you have trouble viewing plain text, try the HTML version described above.

All seven parts of the FAQ can be downloaded from either of the following URLS:

ftp://ftp.sas.com/pub/neural/FAQ.html.zip
ftp://ftp.sas.com/pub/neural/FAQ.txt.zip

These postings are archived in the periodic posting archive on host rtfm.mit.edu (and on some other hosts as well). Look in the anonymous ftp directory "/pub/usenet/news.answers/ai-faq/neural-nets" under the file names "part1", "part2", ... "part7". If you do not have anonymous ftp access, you can access the archives by mail server as well. Send an E-mail message to mail-server@rtfm.mit.edu with "help" and "index" in the body on separate lines for more information.

For those of you who read this FAQ anywhere other than in Usenet: To read comp.ai.neural-nets (or post articles to it) you need Usenet News access. Try the commands, 'xrn', 'rn', 'nn',

or 'trn' on your Unix machine, 'news' on your VMS machine, or ask a local guru. WWW browsers are often set up for Usenet access, too--try the URL news:comp.ai.neural-nets.

The FAQ posting departs to comp.ai.neural-nets around the 28th of every month. It is also sent to the groups comp.answers and news.answers where it should be available at any time (ask your news manager). The FAQ posting, like any other posting, may a take a few days to find its way over Usenet to your site. Such delays are especially common outside of North America.

All changes to the FAQ from the previous month are shown in another monthly posting having the subject `changes to "comp.ai.neural-nets FAQ" -- monthly posting', which immediately follows the FAQ posting. The `changes' post contains the full text of all changes and can also be found at ftp://ftp.sas.com/pub/neural/changes.txt . There is also a weekly post with the subject "comp.ai.neural-nets FAQ: weekly reminder" that briefly describes any major changes to the FAQ.

This FAQ is not meant to discuss any topic exhaustively. It is neither a tutorial nor a textbook, but should be viewed as a supplement to the many excellent books and online resources described in Part 4: Books, data, etc..

Disclaimer:

> This posting is provided 'as is'. No warranty whatsoever is expressed or implied, in particular, no warranty that the information contained herein is correct or useful in any way, although both are intended.

To find the answer of question "x", search for the string "Subject: x"

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

Part 1: Introduction

Subject: What is this newsgroup for? How shall it be used?

The newsgroup comp.ai.neural-nets is intended as a forum for people who want to use or explore the capabilities of Artificial Neural Networks or Neural-Network-like structures.

Posts should be in plain-text format, not postscript, html, rtf, TEX, MIME, or any word-processor format.

Do not use vcards or other excessively long signatures.

Please do not post homework or take-home exam questions. Please do not post a long source-code listing and ask readers to debug it. And note that chain letters and other get-rich-quick pyramid schemes are illegal in the USA; for example, see http://www.usps.gov/websites/depart/inspect/chainlet.htm

There should be the following types of articles in this newsgroup:

1. **Requests**

   Requests are articles of the form `"I am looking for X"`, where X is something public like a book, an article, a piece of software. The most important about such a request is to be as specific as possible!

   If multiple different answers can be expected, the person making the request should prepare to make a summary of the answers he/she got and announce to do so with a phrase like `"Please reply by email, I'll summarize to the group"` at the end of the posting.

   The Subject line of the posting should then be something like `"Request: X"`

2. **Questions**

   As opposed to requests, questions ask for a larger piece of information or a more or less detailed explanation of something. To avoid lots of redundant traffic it is important that the poster provides with the question all information s/he already has about the subject asked and state the actual question as precise and narrow as possible. The poster should prepare to make a summary of the answers s/he got and announce to do so with a phrase like `"Please reply by email, I'll summarize to the`

group" at the end of the posting.

The Subject line of the posting should be something like "Question: this-and-that" or have the form of a question (i.e., end with a question mark)

Students: please do not ask comp.ai.neural-net readers to do your homework or take-home exams for you.

3. Answers

These are reactions to questions or requests. If an answer is too specific to be of general interest, or if a summary was announced with the question or request, the answer should be e-mailed to the poster, not posted to the newsgroup.

Most news-reader software automatically provides a subject line beginning with "Re:" followed by the subject of the article which is being followed-up. Note that sometimes longer threads of discussion evolve from an answer to a question or request. In this case posters should change the subject line suitably as soon as the topic goes too far away from the one announced in the original subject line. You can still carry along the old subject in parentheses in the form "Re: new subject (was: old subject)"

4. Summaries

In all cases of requests or questions the answers for which can be assumed to be of some general interest, the poster of the request or question shall summarize the answers he/she received. Such a summary should be announced in the original posting of the question or request with a phrase like "Please answer by email, I'll summarize"

In such a case, people who answer to a question should NOT post their answer to the newsgroup but instead mail them to the poster of the question who collects and reviews them. After about 5 to 20 days after the original posting, its poster should make the summary of answers and post it to the newsgroup.

Some care should be invested into a summary:

- simple concatenation of all the answers is not enough: instead, redundancies, irrelevancies, verbosities, and errors should be filtered out (as well as possible)
- the answers should be separated clearly
- the contributors of the individual answers should be identifiable (unless they requested to remain anonymous [yes, that happens])
- the summary should start with the "quintessence" of the answers, as seen by the original poster
- A summary should, when posted, clearly be indicated to be one by giving it a Subject line starting with "SUMMARY:"

Note that a good summary is pure gold for the rest of the newsgroup community, so

summary work will be most appreciated by all of us. Good summaries are more valuable than any moderator ! :-)

5. **Announcements**

   Some articles never need any public reaction. These are called announcements (for instance for a workshop, conference or the availability of some technical report or software system).

   Announcements should be clearly indicated to be such by giving them a subject line of the form `"Announcement: this-and-that"`

6. **Reports**

   Sometimes people spontaneously want to report something to the newsgroup. This might be special experiences with some software, results of own experiments or conceptual work, or especially interesting information from somewhere else.

   Reports should be clearly indicated to be such by giving them a subject line of the form `"Report: this-and-that"`

7. **Discussions**

   An especially valuable possibility of Usenet is of course that of discussing a certain topic with hundreds of potential participants. All traffic in the newsgroup that can not be subsumed under one of the above categories should belong to a discussion.

   If somebody explicitly wants to start a discussion, he/she can do so by giving the posting a subject line of the form `"Discussion: this-and-that"`

   It is quite difficult to keep a discussion from drifting into chaos, but, unfortunately, as many many other newsgroups show there seems to be no secure way to avoid this. On the other hand, comp.ai.neural-nets has not had many problems with this effect in the past, so let's just go and hope...

8. **Jobs Ads**

   Advertisements for jobs requiring expertise in artificial neural networks are appropriate in comp.ai.neural-nets. Job ads should be clearly indicated to be such by giving them a subject line of the form `"Job: this-and-that"`. It is also useful to include the country-state-city abbreviations that are conventional in misc.jobs.offered, such as: `"Job: US-NY-NYC Neural network engineer"`. If an employer has more than one job opening, all such openings should be listed in a single post, not multiple posts. Job ads should not be reposted more than once per month.

Subject: Where is comp.ai.neural-nets archived?

The following archives are available for comp.ai.neural-nets:

- http://groups.google.com, formerly Deja News. Does not work very well yet.
- 94-09-14 through 97-08-16 ftp://ftp.cs.cmu.edu/user/ai/pubs/news/comp.ai.neural-nets

For more information on newsgroup archives, see
http://starbase.neosoft.com/~claird/news.lists/newsgroup_archives.html
or http://www.pitt.edu/~grouprev/Usenet/Archive-List/newsgroup_archives.html

---

Subject: What if my question is not answered in the FAQ?

If your question is not answered in the FAQ, you can try a web search. The following search engines are especially useful:
http://www.google.com/
http://search.yahoo.com/
http://www.altavista.com/
http://citeseer.nj.nec.com/cs

Another excellent web site on NNs is Donald Tveter's Backpropagator's Review at http://www.dontveter.com/bpr/bpr.html or http://gannoo.uce.ac.uk/bpr/bpr.html.

For feedforward NNs, the best reference book is:

> Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press.

If the answer isn't in Bishop, then for more theoretical questions try:

> Ripley, B.D. (1996) *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University Press.

For more practical questions about MLP training, try:

> Masters, T. (1993). *Practical Neural Network Recipes in C++*, San Diego: Academic Press.
>
> Reed, R.D., and Marks, R.J, II (1999), *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, Cambridge, MA: The MIT Press.

There are many more excellent books and web sites listed in the Neural Network FAQ, Part 4: Books, data, etc.

Subject: May I copy this FAQ?

The intent in providing a FAQ is to make the information freely available to whoever needs it. You may copy all or part of the FAQ, but please be sure to include a reference to the URL of the master copy, ftp://ftp.sas.com/pub/neural/FAQ.html, and do not sell copies of the FAQ. If you want to include information from the FAQ in your own web site, it is better to include links to the master copy rather than to copy text from the FAQ to your web pages, because various answers in the FAQ are updated at unpredictable times. To cite the FAQ in an academic-style bibliography, use something along the lines of:

> Sarle, W.S., ed. (1997), *Neural Network FAQ, part 1 of 7: Introduction*, periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: ftp://ftp.sas.com/pub/neural/FAQ.html

---

Subject: What is a neural network (NN)?

> The question 'What is a neural network?' is ill-posed.
> - Pinkus (1999)

First of all, when we are talking about a neural network, we should more properly say "artificial neural network" (ANN), because that is what we mean most of the time in comp.ai.neural-nets. Biological neural networks are much more complicated than the mathematical models we use for ANNs. But it is customary to be lazy and drop the "A" or the "artificial".

There is no universally accepted definition of an NN. But perhaps most people in the field would agree that an NN is a network of many simple processors ("units"), each possibly having a small amount of local memory. The units are connected by communication channels ("connections") which usually carry numeric (as opposed to symbolic) data, encoded by any of various means. The units operate only on their local data and on the inputs they receive via the connections. The restriction to local operations is often relaxed during training.

Some NNs are models of biological neural networks and some are not, but historically, much of the inspiration for the field of NNs came from the desire to produce artificial systems capable of sophisticated, perhaps "intelligent", computations similar to those that the human brain routinely performs, and thereby possibly to enhance our understanding of the human brain.

Most NNs have some sort of "training" rule whereby the weights of connections are adjusted on the basis of data. In other words, NNs "learn" from examples, as children learn

to distinguish dogs from cats based on examples of dogs and cats. If trained carefully, NNs may exhibit some capability for generalization beyond the training data, that is, to produce approximately correct results for new cases that were not used for training.

NNs normally have great potential for parallelism, since the computations of the components are largely independent of each other. Some people regard massive parallelism and high connectivity to be defining characteristics of NNs, but such requirements rule out various simple models, such as simple linear regression (a minimal feedforward net with only two units plus bias), which are usefully regarded as special cases of NNs.

Here is a sampling of definitions from the books on the FAQ maintainer's shelf. None will please everyone. Perhaps for that reason many NN textbooks do not explicitly define neural networks.

According to the *DARPA Neural Network Study* (1988, AFCEA International Press, p. 60):

> ... a neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.

According to Haykin (1994), p. 2:

> A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:
>
> 1. Knowledge is acquired by the network through a learning process.
> 2. Interneuron connection strengths known as synaptic weights are used to store the knowledge.

According to Nigrin (1993), p. 11:

> A neural network is a circuit composed of a very large number of simple processing elements that are neurally based. Each element operates only on local information. Furthermore each element operates asynchronously; thus there is no overall system clock.

According to Zurada (1992), p. xv:

> Artificial neural systems, or neural networks, are physical cellular systems which can acquire, store, and utilize experiential knowledge.

References:

Pinkus, A. (1999), "Approximation theory of the MLP model in neural networks,"

Acta Numerica, 8, 143-196.

Haykin, S. (1994), *Neural Networks: A Comprehensive Foundation*, NY: Macmillan.

Nigrin, A. (1993), *Neural Networks for Pattern Recognition*, Cambridge, MA: The MIT Press.

Zurada, J.M. (1992), *Introduction To Artificial Neural Systems*, Boston: PWS Publishing Company.

Subject: Where can I find a simple introduction to NNs?

Several excellent introductory books on NNs are listed in part 4 of the FAQ under "Books and articles about Neural Networks?" If you want a book with minimal math, see "The best introductory book for business executives."

Dr. Leslie Smith has a brief on-line introduction to NNs with examples and diagrams at http://www.cs.stir.ac.uk/~lss/NNIntro/InvSlides.html.

If you are a Java enthusiast, see Jochen Fröhlich's diploma at http://rfhs8012.fh-regensburg.de/~saj39122/jfroehl/diplom/e-index.html

For a more detailed introduction, see Donald Tveter's excellent Backpropagator's Review at http://www.dontveter.com/bpr/bpr.html or http://gannoo.uce.ac.uk/bpr/bpr.html, which contains both answers to additional FAQs and an annotated neural net bibliography emphasizing on-line articles.

StatSoft Inc. has an on-line *Electronic Statistics Textbook*, at http://www.statsoft.com/textbook/stathome.html that includes a chapter on neural nets as well as many statistical topics relevant to neural nets.

Subject: Are there any online books about NNs?

Kevin Gurney has on-line a preliminary draft of his book, *An Introduction to Neural Networks*, at http://www.shef.ac.uk/psychology/gurney/notes/index.html The book is now in print and is one of the better general-purpose introductory textbooks on NNs. Here is the table of contents from the on-line version:

1. Computers and Symbols versus Nets and Neurons
2. TLUs and vectors - simple learning rules
3. The delta rule
4. Multilayer nets and backpropagation

5. Associative memories - the Hopfield net
6. Hopfield nets (contd.)
7. Kohonen nets
8. Alternative node types
9. Cubic nodes (contd.) and Reward Penalty training
10. Drawing things together - some perspectives

Another on-line book by Ben Kröse and Patrick van der Smagt, also called *An Introduction to Neural Networks*, can be found at ftp://ftp.wins.uva.nl/pub/computer-systems/aut-sys/reports/neuro-intro/neuro-intro.ps.gz or http://www.robotic.dlr.de/Smagt/books/neuro-intro.ps.gz. or http://www.supelec-rennes.fr/acth/net/neuro-intro.ps.gz
Here is the table of contents:

1. Introduction
2. Fundamantals
3. Perceptron and Adaline
4. Back-Propagation
5. Recurrent Networks
6. Self-Organising Networks
7. Reinforcement Learning
8. Robot Control
9. Vision
10. General Purpose Hardware
11. Dedicated Neuro-Hardware

Subject: What can you do with an NN and what not?

In principle, NNs can compute any computable function, i.e., they can do everything a normal digital computer can do (Valiant, 1988; Siegelmann and Sontag, 1999; Orponen, 2000; Sima and Orponen, 2001), or perhaps even more, under some assumptions of doubtful practicality (see Siegelmann, 1998, but also Hadley, 1999).

Practical applications of NNs most often employ supervised learning. For supervised learning, you must provide training data that includes both the input and the desired result (the target value). After successful training, you can present input data alone to the NN (that is, input data without the desired result), and the NN will compute an output value that approximates the desired result. However, for training to be successful, you may need lots of training data and lots of computer time to do the training. In many applications, such as image and text processing, you will have to do a lot of work to select appropriate input data and to code the data as numeric values.

In practice, NNs are especially useful for classification and function approximation/mapping problems which are tolerant of some imprecision, which have lots of training data available,

but to which hard and fast rules (such as those that might be used in an expert system) cannot easily be applied. Almost any finite-dimensional vector function on a compact set can be approximated to arbitrary precision by feedforward NNs (which are the type most often used in practical applications) if you have enough data and enough computing resources.

To be somewhat more precise, feedforward networks with a single hidden layer and trained by least-squares are statistically consistent estimators of arbitrary square-integrable regression functions under certain practically-satisfiable assumptions regarding sampling, target noise, number of hidden units, size of weights, and form of hidden-unit activation function (White, 1990). Such networks can also be trained as statistically consistent estimators of derivatives of regression functions (White and Gallant, 1992) and quantiles of the conditional noise distribution (White, 1992a). Feedforward networks with a single hidden layer using threshold or sigmoid activation functions are universally consistent estimators of binary classifications (Faragó and Lugosi, 1993; Lugosi and Zeger 1995; Devroye, Györfi, and Lugosi, 1996) under similar assumptions. Note that these results are stronger than the universal approximation theorems that merely show the existence of weights for arbitrarily accurate approximations, without demonstrating that such weights can be obtained by learning.

Unfortunately, the above consistency results depend on one impractical assumption: that the networks are trained by an error (`L_p` error or misclassification rate) minimization technique that comes arbitrarily close to the global minimum. Such minimization is computationally intractable except in small or simple problems (Blum and Rivest, 1989; Judd, 1990). In practice, however, you can usually get good results without doing a full-blown global optimization; e.g., using multiple (say, 10 to 1000) random weight initializations is usually sufficient.

One example of a function that a typical neural net cannot learn is `Y=1/X` on the open interval (0,1). An open interval is not a compact set. With any bounded output activation function, the error will get arbitrarily large as the input approaches zero. Of course, you could make the output activation function a reciprocal function and easily get a perfect fit, but neural networks are most often used in situations where you do not have enough prior knowledge to set the activation function in such a clever way. There are also many other important problems that are so difficult that a neural network will be unable to learn them without memorizing the entire training set, such as:

- Predicting random or pseudo-random numbers.
- Factoring large integers.
- Determing whether a large integer is prime or composite.
- Decrypting anything encrypted by a good algorithm.

And it is important to understand that there are no methods for training NNs that can magically create information that is not contained in the training data.

Feedforward NNs are restricted to finite-dimensional input and output spaces. Recurrent NNs can in theory process arbitrarily long strings of numbers or symbols. But training recurrent NNs has posed much more serious practical difficulties than training feedforward networks. NNs are, at least today, difficult to apply successfully to problems that concern manipulation of symbols and rules, but much research is being done.

There have been attempts to pack recursive structures into finite-dimensional real vectors (Blair, 1997; Pollack, 1990; Chalmers, 1990; Chrisman, 1991; Plate, 1994; Hammerton, 1998). Obviously, finite precision limits how far the recursion can go (Hadley, 1999). The practicality of such methods is open to debate.

As for simulating human consciousness and emotion, that's still in the realm of science fiction. Consciousness is still one of the world's great mysteries. Artificial NNs may be useful for modeling some aspects of or prerequisites for consciousness, such as perception and cognition, but ANNs provide no insight so far into what Chalmers (1996, p. xi) calls the "hard problem":

> Many books and articles on consciousness have appeared in the past few years, and one might think we are making progress. But on a closer look, most of this work leaves the hardest problems about consciousness untouched. Often, such work addresses what might be called the "easy problems" of consciousness: How does the brain process environmental stimulation? How does it integrate information? How do we produce reports on internal states? These are important questions, but to answer them is not to solve the hard problem: Why is all this processing accompanied by an experienced inner life?

For more information on consciousness, see the on-line journal Psychе at http://psyche.cs.monash.edu.au/index.html.

For examples of specific applications of NNs, see What are some applications of NNs?

References:

Blair, A.D. (1997), "Scaling Up RAAMs," Brandeis University Computer Science Technical Report CS-97-192, http://www.demo.cs.brandeis.edu/papers/long.html#sur97

Blum, A., and Rivest, R.L. (1989), "Training a 3-node neural network is NP-complete," in Touretzky, D.S. (ed.), *Advances in Neural Information Processing Systems 1*, San Mateo, CA: Morgan Kaufmann, 494-501.

Chalmers, D.J. (1990), "Syntactic Transformations on Distributed Representations," Connection Science, 2, 53-62, http://ling.ucsc.edu/~chalmers/papers/transformations.ps

Chalmers, D.J. (1996), *The Conscious Mind: In Search of a Fundamental Theory*, NY:

Oxford University Press.

Chrisman, L. (1991), "Learning Recursive Distributed Representations for Holistic Computation", Connection Science, 3, 345-366, ftp://reports.adm.cs.cmu.edu/usr/anon/1991/CMU-CS-91-154.ps

Collier, R. (1994), "An historical overview of natural language processing systems that learn," Artificial Intelligence Review, 8(1), ??-??.

Devroye, L., Györfi, L., and Lugosi, G. (1996), *A Probabilistic Theory of Pattern Recognition*, NY: Springer.

Faragó, A. and Lugosi, G. (1993), "Strong Universal Consistency of Neural Network Classifiers," IEEE Transactions on Information Theory, 39, 1146-1151.

Hadley, R.F. (1999), "Cognition and the computational power of connectionist networks," http://www.cs.sfu.ca/~hadley/online.html

Hammerton, J.A. (1998), "Holistic Computation: Reconstructing a muddled concept," Connection Science, 10, 3-19, http://www.tardis.ed.ac.uk/~james/CNLP/holcomp.ps.gz

Judd, J.S. (1990), *Neural Network Design and the Complexity of Learning*, Cambridge, MA: The MIT Press.

Lugosi, G., and Zeger, K. (1995), "Nonparametric Estimation via Empirical Risk Minimization," IEEE Transactions on Information Theory, 41, 677-678.

Orponen, P. (2000), "An overview of the computational power of recurrent neural networks," Finnish AI Conference, Helsinki, http://www.math.jyu.fi/~orponen/papers/rnncomp.ps

Plate, T.A. (1994), *Distributed Representations and Nested Compositional Structure*, Ph.D. Thesis, University of Toronto, ftp://ftp.cs.utoronto.ca/pub/tap/

Pollack, J. B. (1990), "Recursive Distributed Representations," Artificial Intelligence 46, 1, 77-105, http://www.demo.cs.brandeis.edu/papers/long.html#raam

Siegelmann, H.T. (1998), *Neural Networks and Analog Computation: Beyond the Turing Limit*, Boston: Birkhauser, ISBN 0-8176-3949-7, http://iew3.technion.ac.il:8080/Home/Users/iehava/book/

Siegelmann, H.T., and Sontag, E.D. (1999), "Turing Computability with Neural Networks," Applied Mathematics Letters, 4, 77-80.

Sima, J., and Orponen, P. (2001), "Computing with continuous-time Liapunov

systems," 33rd ACM STOC, http://www.math.jyu.fi/~orponen/papers/liapcomp.ps

Valiant, L. (1988), "Functionality in Neural Nets," *Learning and Knowledge Acquisition*, Proc. AAAI, 629-634.

White, H. (1990), "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings," Neural Networks, 3, 535-550. Reprinted in White (1992b).

White, H. (1992a), "Nonparametric Estimation of Conditional Quantiles Using Neural Networks," in Page, C. and Le Page, R. (eds.), *Proceedings of the 23rd Sympsium on the Interface: Computing Science and Statistics*, Alexandria, VA: American Statistical Association, pp. 190-199. Reprinted in White (1992b).

White, H. (1992b), *Artificial Neural Networks: Approximation and Learning Theory*, Blackwell.

White, H., and Gallant, A.R. (1992), "On Learning the Derivatives of an Unknown Mapping with Multilayer Feedforward Networks," Neural Networks, 5, 129-138. Reprinted in White (1992b).

Subject: Who is concerned with NNs?

Neural Networks are interesting for quite a lot of very different people:

- Computer scientists want to find out about the properties of non-symbolic information processing with neural nets and about learning systems in general.
- Statisticians use neural nets as flexible, nonlinear regression and classification models.
- Engineers of many kinds exploit the capabilities of neural networks in many areas, such as signal processing and automatic control.
- Cognitive scientists view neural networks as a possible apparatus to describe models of thinking and consciousness (High-level brain function).
- Neuro-physiologists use neural networks to describe and explore medium-level brain function (e.g. memory, sensory system, motorics).
- Physicists use neural networks to model phenomena in statistical mechanics and for a lot of other tasks.
- Biologists use Neural Networks to interpret nucleotide sequences.
- Philosophers and some other people may also be interested in Neural Networks for various reasons.

For world-wide lists of groups doing research on NNs, see the Foundation for Neural Networks's (SNN) page at http://www.mbfys.kun.nl/snn/pointers/groups.html and see Neural Networks Research on the IEEE Neural Network Council's homepage http://www.ieee.org/nnc.

Subject: How many kinds of NNs exist?

There are many many kinds of NNs by now. Nobody knows exactly how many. New ones (or at least variations of old ones) are invented every week. Below is a collection of some of the most well known methods, not claiming to be complete.

The two main kinds of learning algorithms are supervised and unsupervised.

- In supervised learning, the correct results (target values, desired outputs) are known and are given to the NN during training so that the NN can adjust its weights to try match its outputs to the target values. After training, the NN is tested by giving it only input values, not target values, and seeing how close it comes to outputting the correct target values.
- In unsupervised learning, the NN is not provided with the correct results during training. Unsupervised NNs usually perform some kind of data compression, such as dimensionality reduction or clustering. See "What does unsupervised learning learn?"

The distinction between supervised and unsupervised methods is not always clear-cut. An unsupervised method can learn a summary of a probability distribution, then that summarized distribution can be used to make predictions. Furthermore, supervised methods come in two subvarieties: auto-associative and hetero-associative. In auto-associative learning, the target values are the same as the inputs, whereas in hetero-associative learning, the targets are generally different from the inputs. Many unsupervised methods are equivalent to auto-associative supervised methods. For more details, see "What does unsupervised learning learn?"

Two major kinds of network topology are feedforward and feedback.

- In a feedforward NN, the connections between units do not form cycles. Feedforward NNs usually produce a response to an input quickly. Most feedforward NNs can be trained using a wide variety of efficient conventional numerical methods (e.g. see "What are conjugate gradients, Levenberg-Marquardt, etc.?") in addition to algorithms invented by NN reserachers.
- In a feedback or recurrent NN, there are cycles in the connections. In some feedback NNs, each time an input is presented, the NN must iterate for a potentially long time before it produces a response. Feedback NNs are usually more difficult to train than feedforward NNs.

Some kinds of NNs (such as those with winner-take-all units) can be implemented as either feedforward or feedback networks.

NNs also differ in the kinds of data they accept. Two major kinds of data are categorical and quantitative.

- Categorical variables take only a finite (technically, countable) number of possible values, and there are usually several or more cases falling into each category. Categorical variables may have symbolic values (e.g., "male" and "female", or "red", "green" and "blue") that must be encoded into numbers before being given to the network (see "How should categories be encoded?") Both supervised learning with categorical target values and unsupervised learning with categorical outputs are called "classification."
- Quantitative variables are numerical measurements of some attribute, such as length in meters. The measurements must be made in such a way that at least some arithmetic relations among the measurements reflect analogous relations among the attributes of the objects that are measured. For more information on measurement theory, see the Measurement Theory FAQ at ftp://ftp.sas.com/pub/neural/measurement.html. Supervised learning with quantitative target values is called "regression."

Some variables can be treated as either categorical or quantitative, such as number of children or any binary variable. Most regression algorithms can also be used for supervised classification by encoding categorical target values as 0/1 binary variables and using those binary variables as target values for the regression algorithm. The outputs of the network are posterior probabilities when any of the most common training methods are used.

Here are some well-known kinds of NNs:

1. Supervised
    1. Feedforward
        - Linear
            - Hebbian - Hebb (1949), Fausett (1994)
            - Perceptron - Rosenblatt (1958), Minsky and Papert (1969/1988), Fausett (1994)
            - Adaline - Widrow and Hoff (1960), Fausett (1994)
            - Higher Order - Bishop (1995)
            - Functional Link - Pao (1989)
        - MLP: Multilayer perceptron - Bishop (1995), Reed and Marks (1999), Fausett (1994)
            - Backprop - Rumelhart, Hinton, and Williams (1986)
            - Cascade Correlation - Fahlman and Lebiere (1990), Fausett (1994)
            - Quickprop - Fahlman (1989)
            - RPROP - Riedmiller and Braun (1993)
        - RBF networks - Bishop (1995), Moody and Darken (1989), Orr (1996)
            - OLS: Orthogonal Least Squares - Chen, Cowan and Grant (1991)
        - CMAC: Cerebellar Model Articulation Controller - Albus (1975), Brown and Harris (1994)
        - Classification only
            - LVQ: Learning Vector Quantization - Kohonen (1988), Fausett (1994)

        - PNN: Probabilistic Neural Network - Specht (1990), Masters (1993), Hand (1982), Fausett (1994)
    - Regression only
        - GNN: General Regression Neural Network - Specht (1991), Nadaraya (1964), Watson (1964)

2. Feedback - Hertz, Krogh, and Palmer (1991), Medsker and Jain (2000)

    - BAM: Bidirectional Associative Memory - Kosko (1992), Fausett (1994)
    - Boltzman Machine - Ackley et al. (1985), Fausett (1994)
    - Recurrent time series
        - Backpropagation through time - Werbos (1990)
        - Elman - Elman (1990)
        - FIR: Finite Impulse Response - Wan (1990)
        - Jordan - Jordan (1986)
        - Real-time recurrent network - Williams and Zipser (1989)
        - Recurrent backpropagation - Pineda (1989), Fausett (1994)
        - TDNN: Time Delay NN - Lang, Waibel and Hinton (1990)

3. Competitive

    - ARTMAP - Carpenter, Grossberg and Reynolds (1991)
    - Fuzzy ARTMAP - Carpenter, Grossberg, Markuzon, Reynolds and Rosen (1992), Kasuba (1993)
    - Gaussian ARTMAP - Williamson (1995)
    - Counterpropagation - Hecht-Nielsen (1987; 1988; 1990), Fausett (1994)
    - Neocognitron - Fukushima, Miyake, and Ito (1983), Fukushima, (1988), Fausett (1994)

2. Unsupervised - Hertz, Krogh, and Palmer (1991)

    1. Competitive

        - Vector Quantization
            - Grossberg - Grossberg (1976)
            - Kohonen - Kohonen (1984)
            - Conscience - Desieno (1988)
        - Self-Organizing Map
            - Kohonen - Kohonen (1995), Fausett (1994)
            - GTM: - Bishop, Svensén and Williams (1997)
            - Local Linear - Mulier and Cherkassky (1995)
        - Adaptive resonance theory
            - ART 1 - Carpenter and Grossberg (1987a), Moore (1988), Fausett (1994)
            - ART 2 - Carpenter and Grossberg (1987b), Fausett (1994)
            - ART 2-A - Carpenter, Grossberg and Rosen (1991a)

        - ART 3 - Carpenter and Grossberg (1990)
        - Fuzzy ART - Carpenter, Grossberg and Rosen (1991b)
      - DCL: Differential Competitive Learning - Kosko (1992)

  2. Dimension Reduction - Diamantaras and Kung (1996)

      - Hebbian - Hebb (1949), Fausett (1994)
      - Oja - Oja (1989)
      - Sanger - Sanger (1989)
      - Differential Hebbian - Kosko (1992)

  3. Autoassociation

      - Linear autoassociator - Anderson et al. (1977), Fausett (1994)
      - BSB: Brain State in a Box - Anderson et al. (1977), Fausett (1994)
      - Hopfield - Hopfield (1982), Fausett (1994)

3. Nonlearning

    1. Hopfield - Hertz, Krogh, and Palmer (1991)
    2. various networks for optimization - Cichocki and Unbehauen (1993)

References:

Ackley, D.H., Hinton, G.F., and Sejnowski, T.J. (1985), "A learning algorithm for Boltzman machines," Cognitive Science, 9, 147-169.

Albus, J.S (1975), "New Approach to Manipulator Control: The Cerebellar Model Articulation Controller (CMAC)," Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control, September 1975, 220-27.

Anderson, J.A., and Rosenfeld, E., eds. (1988), *Neurocomputing: Foundatons of Research*, Cambridge, MA: The MIT Press.

Anderson, J.A., Silverstein, J.W., Ritz, S.A., and Jones, R.S. (1977) "Distinctive features, categorical perception, and probability learning: Some applications of a neural model," Psychological Rveiew, 84, 413-451. Reprinted in Anderson and Rosenfeld (1988).

Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press.

Bishop, C.M., Svensén, M., and Williams, C.K.I (1997), "GTM: A principled alternative to the self-organizing map," in Mozer, M.C., Jordan, M.I., and Petsche, T., (eds.) *Advances in Neural Information Processing Systems 9*, Cambrideg, MA: The MIT Press, pp. 354-360. Also see http://www.ncrg.aston.ac.uk/GTM/

Brown, M., and Harris, C. (1994), *Neurofuzzy Adaptive Modelling and Control*, NY: Prentice Hall.

Carpenter, G.A., Grossberg, S. (1987a), "A massively parallel architecture for a self-organizing neural pattern recognition machine," Computer Vision, Graphics, and Image Processing, 37, 54-115.

Carpenter, G.A., Grossberg, S. (1987b), "ART 2: Self-organization of stable category recognition codes for analog input patterns," Applied Optics, 26, 4919-4930.

Carpenter, G.A., Grossberg, S. (1990), "ART 3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. Neural Networks, 3, 129-152.

Carpenter, G.A., Grossberg, S., Markuzon, N., Reynolds, J.H., and Rosen, D.B. (1992), "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," IEEE Transactions on Neural Networks, 3, 698-713

Carpenter, G.A., Grossberg, S., Reynolds, J.H. (1991), "ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network," Neural Networks, 4, 565-588.

Carpenter, G.A., Grossberg, S., Rosen, D.B. (1991a), "ART 2-A: An adaptive resonance algorithm for rapid category learning and recognition," Neural Networks, 4, 493-504.

Carpenter, G.A., Grossberg, S., Rosen, D.B. (1991b), "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," Neural Networks, 4, 759-771.

Chen, S., Cowan, C.F.N., and Grant, P.M. (1991), "Orthogonal least squares learning for radial basis function networks," IEEE Transactions on Neural Networks, 2, 302-309.

Cichocki, A. and Unbehauen, R. (1993). *Neural Networks for Optimization and Signal Processing.* NY: John Wiley & Sons, ISBN 0-471-93010-5.

Desieno, D. (1988), "Adding a conscience to competitive learning," Proc. Int. Conf. on Neural Networks, I, 117-124, IEEE Press.

Diamantaras, K.I., and Kung, S.Y. (1996) *Principal Component Neural Networks: Theory and Applications*, NY: Wiley.

Elman, J.L. (1990), "Finding structure in time," Cognitive Science, 14, 179-211.

Fahlman, S.E. (1989), "Faster-Learning Variations on Back-Propagation: An Empirical Study", in Touretzky, D., Hinton, G, and Sejnowski, T., eds., *Proceedings of the 1988 Connectionist Models Summer School*, Morgan Kaufmann, 38-51.

Fahlman, S.E., and Lebiere, C. (1990), "The Cascade-Correlation Learning Architecture", in Touretzky, D. S. (ed.), *Advances in Neural Information Processing Systems 2*,, Los Altos, CA: Morgan Kaufmann Publishers, pp. 524-532.

Fausett, L. (1994), *Fundamentals of Neural Networks*, Englewood Cliffs, NJ: Prentice Hall.

Fukushima, K., Miyake, S., and Ito, T. (1983), "Neocognitron: A neural network model for a mechanism of visual pattern recognition," IEEE Transactions on Systems, Man, and Cybernetics, 13, 826-834.

Fukushima, K. (1988), "Neocognitron: A hierarchical neural network capable of visual pattern recognition," Neural Networks, 1, 119-130.

Grossberg, S. (1976), "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors," Biological Cybernetics, 23, 121-134

Hand, D.J. (1982) *Kernel Discriminant Analysis*, Research Studies Press.

Hebb, D.O. (1949), *The Organization of Behavior*, NY: John Wiley & Sons.

Hecht-Nielsen, R. (1987), "Counterpropagation networks," Applied Optics, 26, 4979-4984.

Hecht-Nielsen, R. (1988), "Applications of counterpropagation networks," Neural Networks, 1, 131-139.

Hecht-Nielsen, R. (1990), *Neurocomputing*, Reading, MA: Addison-Wesley.

Hertz, J., Krogh, A., and Palmer, R. (1991). *Introduction to the Theory of Neural Computation.* Addison-Wesley: Redwood City, California.

Hopfield, J.J. (1982), "Neural networks and physical systems with emergent collective computational abilities," Proceedings of the National Academy of Sciences, 79, 2554-2558. Reprinted in Anderson and Rosenfeld (1988).

Jordan, M. I. (1986), "Attractor dynamics and parallelism in a connectionist sequential machine," In Proceedings of the Eighth Annual conference of the Cognitive Science Society, pages 531-546. Lawrence Erlbaum.

Kasuba, T. (1993), "Simplified Fuzzy ARTMAP," AI Expert, 8, 18-25.

Kohonen, T. (1984), *Self-Organization and Associative Memory*, Berlin: Springer.

Kohonen, T. (1988), "Learning Vector Quantization," Neural Networks, 1 (suppl 1), 303.

Kohonen, T. (1995/1997), *Self-Organizing Maps*, Berlin: Springer-Verlag. First edition was 1995, second edition 1997. See http://www.cis.hut.fi/nnrc/new_book.html for information on the second edition.

Kosko, B.(1992), *Neural Networks and Fuzzy Systems*, Englewood Cliffs, N.J.: Prentice-Hall.

Lang, K. J., Waibel, A. H., and Hinton, G. (1990), "A time-delay neural network architecture for isolated word recognition," Neural Networks, 3, 23-44.

Masters, T. (1993). *Practical Neural Network Recipes in C++*, San Diego: Academic Press.

Masters, T. (1995) *Advanced Algorithms for Neural Networks: A C++ Sourcebook*, NY: John Wiley and Sons, ISBN 0-471-10588-0

Medsker, L.R., and Jain, L.C., eds. (2000), *Recurrent Neural Networks: Design and Applications*, Boca Raton, FL: CRC Press, ISBN 0-8493-7181-3.

Minsky, M.L., and Papert, S.A. (1969/1988), *Perceptrons*, Cambridge, MA: The MIT Press (first edition, 1969; expanded edition, 1988).

Moody, J. and Darken, C.J. (1989), "Fast learning in networks of locally-tuned processing units," Neural Computation, 1, 281-294.

Moore, B. (1988), "ART 1 and Pattern Clustering," in Touretzky, D., Hinton, G. and Sejnowski, T., eds., *Proceedings of the 1988 Connectionist Models Summer School*, 174-185, San Mateo, CA: Morgan Kaufmann.

Mulier, F. and Cherkassky, V. (1995), "Self-Organization as an Iterative Kernel Smoothing Process," Neural Computation, 7, 1165-1177.

Nadaraya, E.A. (1964) "On estimating regression", Theory Probab. Applic. 10, 186-90.

Oja, E. (1989), "Neural networks, principal components, and subspaces," International Journal of Neural Systems, 1, 61-68.

Orr, M.J.L. (1996), "Introduction to radial basis function networks," http://www.anc.ed.ac.uk/~mjo/papers/intro.ps or http://www.anc.ed.ac.uk/~mjo/papers/intro.ps.gz

Pao, Y. H. (1989), *Adaptive Pattern Recognition and Neural Networks*, Reading, MA: Addison-Wesley Publishing Company, ISBN 0-201-12584-6.

Pineda, F.J. (1989), "Recurrent back-propagation and the dynamical approach to neural computation," Neural Computation, 1, 161-172.

Reed, R.D., and Marks, R.J, II (1999), *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks*, Cambridge, MA: The MIT Press, ISBN 0-262-18190-8.

Riedmiller, M. and Braun, H. (1993), "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm", *Proceedings of the IEEE International Conference on Neural Networks* 1993, San Francisco: IEEE.

Rosenblatt, F. (1958), "The perceptron: A probabilistic model for information storage and organization in the brain., Psychological Review, 65, 386-408.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986), "Learning internal representations by error propagation", in Rumelhart, D.E. and McClelland, J. L., eds. (1986), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Volume 1, 318-362, Cambridge, MA: The MIT Press.

Sanger, T.D. (1989), "Optimal unsupervised learning in a single-layer linear feedforward neural network," Neural Networks, 2, 459-473.

Specht, D.F. (1990) "Probabilistic neural networks," Neural Networks, 3, 110-118.

Specht, D.F. (1991) "A Generalized Regression Neural Network", IEEE Transactions on Neural Networks, 2, Nov. 1991, 568-576.

Wan, E.A. (1990), "Temporal backpropagation: An efficient algorithm for finite impulse response neural networks," in *Proceedings of the 1990 Connectionist Models Summer School*, Touretzky, D.S., Elman, J.L., Sejnowski, T.J., and Hinton, G.E., eds., San Mateo, CA: Morgan Kaufmann, pp. 131-140.

Watson, G.S. (1964) "Smooth regression analysis", Sankhy{\=a}, Series A, 26, 359-72.

Werbos, P.J. (1990), "Backpropagtion through time: What it is and how to do it," Proceedings of the IEEE, 78, 1550-1560.

Widrow, B., and Hoff, M.E., Jr., (1960), "Adaptive switching circuits," IRE WESCON Convention Record. part 4, pp. 96-104. Reprinted in Anderson and Rosenfeld (1988).

Williams, R.J., and Zipser, D., (1989), "A learning algorithm for continually running fully recurrent neurla networks," Neural Computation, 1, 270-280.

Williamson, J.R. (1995), "Gaussian ARTMAP: A neural network for fast incremental learning of noisy multidimensional maps," Technical Report CAS/CNS-95-003, Boston University, Center of Adaptive Systems and Department of Cognitive and Neural Systems.

Subject: How many kinds of Kohonen networks exist? (And what is k-means?)

Teuvo Kohonen is one of the most famous and prolific researchers in neurocomputing, and he has invented a variety of networks. But many people refer to "Kohonen networks" without specifying which *kind* of Kohonen network, and this lack of precision can lead to confusion. The phrase "Kohonen network" most often refers to one of the following three types of networks:

- VQ: Vector Quantization--competitive networks that can be viewed as unsupervised density estimators or autoassociators (Kohonen, 1995/1997; Hecht-Nielsen 1990), closely related to k-means cluster analysis (MacQueen, 1967; Anderberg, 1973). Each competitive unit corresponds to a cluster, the center of which is called a "codebook vector". Kohonen's learning law is an on-line algorithm that finds the codebook vector closest to each training case and moves the "winning" codebook vector closer to the training case. The codebook vector is moved a certain proportion of the distance between it and the training case, the proportion being specified by the learning rate, that is:

  ```
  new_codebook = old_codebook * (1-learning_rate)
               + data * learning_rate
  ```

  Numerous similar algorithms have been developed in the neural net and machine learning literature; see Hecht-Nielsen (1990) for a brief historical overview, and Kosko (1992) for a more technical overview of competitive learning.

  MacQueen's on-line k-means algorithm is essentially the same as Kohonen's learning law except that the learning rate is the reciprocal of the number of cases that have been assigned to the winnning cluster. Suppose that when processing a given training case, `N` cases have been previously assigned to the winning codebook vector. Then the codebook vector is updated as:

  ```
  new_codebook = old_codebook * N/(N+1)
               + data * 1/(N+1)
  ```

  This reduction of the learning rate makes each codebook vector the mean of all cases assigned to its cluster and guarantees convergence of the algorithm to an optimum

value of the error function (the sum of squared Euclidean distances between cases and codebook vectors) as the number of training cases goes to infinity. Kohonen's learning law with a fixed learning rate does not converge. As is well known from stochastic approximation theory, convergence requires the sum of the infinite sequence of learning rates to be infinite, while the sum of squared learning rates must be finite (Kohonen, 1995, p. 34). These requirements are satisfied by MacQueen's k-means algorithm.

Kohonen VQ is often used for off-line learning, in which case the training data are stored and Kohonen's learning law is applied to each case in turn, cycling over the data set many times (incremental training). Convergence to a local optimum can be obtained as the training time goes to infinity if the learning rate is reduced in a suitable manner as described above. However, there are off-line k-means algorithms, both batch and incremental, that converge in a finite number of iterations (Anderberg, 1973; Hartigan, 1975; Hartigan and Wong, 1979). The batch algorithms such as Forgy's (1965; Anderberg, 1973) have the advantage for large data sets, since the incremental methods require you either to store the cluster membership of each case or to do two nearest-cluster computations as each case is processed. Forgy's algorithm is a simple alternating least-squares algorithm consisting of the following steps:

1. Initialize the codebook vectors.
2. Repeat the following two steps until convergence:
   A. Read the data, assigning each case to the nearest (using Euclidean distance) codebook vector.
   B. Replace each codebook vector with the mean of the cases that were assigned to it.

Fastest training is usually obtained if MacQueen's on-line algorithm is used for the first pass and off-line k-means algorithms are applied on subsequent passes (Bottou and Bengio, 1995). However, these training methods do not necessarily converge to a global optimum of the error function. The chance of finding a global optimum can be improved by using rational initialization (SAS Institute, 1989, pp. 824-825), multiple random initializations, or various time-consuming training methods intended for global optimization (Ismail and Kamel, 1989; Zeger, Vaisy, and Gersho, 1992).

VQ has been a popular topic in the signal processing literature, which has been largely separate from the literature on Kohonen networks and from the cluster analysis literature in statistics and taxonomy. In signal processing, on-line methods such as Kohonen's and MacQueen's are called "adaptive vector quantization" (AVQ), while off-line k-means methods go by the names of "Lloyd" or "Lloyd I" (Lloyd, 1982) and "LBG" (Linde, Buzo, and Gray, 1980). There is a recent textbook on VQ by Gersho and Gray (1992) that summarizes these algorithms as information compression methods.

Kohonen's work emphasized VQ as density estimation and hence the desirability of

equiprobable clusters (Kohonen 1984; Hecht-Nielsen 1990). However, Kohonen's learning law does not produce equiprobable clusters--that is, the proportions of training cases assigned to each cluster are not usually equal. If there are `I` inputs and the number of clusters is large, the density of the codebook vectors approximates the `I/(I+2)` power of the density of the training data (Kohonen, 1995, p. 35; Ripley, 1996, p. 202; Zador, 1982), so the clusters are approximately equiprobable only if the data density is uniform or the number of inputs is large. The most popular method for obtaining equiprobability is Desieno's (1988) algorithm which adds a "conscience" value to each distance prior to the competition. The conscience value for each cluster is adjusted during training so that clusters that win more often have larger conscience values and are thus handicapped to even out the probabilities of winning in later iterations.

Kohonen's learning law is an approximation to the k-means model, which is an approximation to normal mixture estimation by maximum likelihood assuming that the mixture components (clusters) all have spherical covariance matrices and equal sampling probabilities. Hence if the population contains clusters that are not equiprobable, k-means will tend to produce sample clusters that are more nearly equiprobable than the population clusters. Corrections for this bias can be obtained by maximizing the likelihood without the assumption of equal sampling probabilities Symons (1981). Such corrections are similar to conscience but have the opposite effect.

In cluster analysis, the purpose is not to compress information but to recover the true cluster memberships. K-means differs from mixture models in that, for k-means, the cluster membership for each case is considered a separate parameter to be estimated, while mixture models estimate a posterior probability for each case based on the means, covariances, and sampling probabilities of each cluster. Balakrishnan, Cooper, Jacob, and Lewis (1994) found that k-means algorithms recovered cluster membership more accurately than Kohonen VQ.

- SOM: Self-Organizing Map--competitive networks that provide a "topological" mapping from the input space to the clusters (Kohonen, 1995). The SOM was inspired by the way in which various human sensory impressions are neurologically mapped into the brain such that spatial or other relations among stimuli correspond to spatial relations among the neurons. In a SOM, the neurons (clusters) are organized into a grid--usually two-dimensional, but sometimes one-dimensional or (rarely) three- or more-dimensional. The grid exists in a space that is separate from the input space; any number of inputs may be used as long as the number of inputs is greater than the dimensionality of the grid space. A SOM tries to find clusters such that any two clusters that are close to each other in the grid space have codebook vectors close to each other in the input space. But the converse does not hold: codebook vectors that are close to each other in the input space do not necessarily correspond to clusters that are close to each other in the grid. Another way to look at this is that a SOM tries to embed the grid in the input space such every training case is close to some codebook

vector, but the grid is bent or stretched as little as possible. Yet another way to look at it is that a SOM is a (discretely) smooth mapping between regions in the input space and points in the grid space. The best way to undestand this is to look at the pictures in Kohonen (1995) or various other NN textbooks.

The Kohonen algorithm for SOMs is very similar to the Kohonen algorithm for AVQ. Let the codebook vectors be indexed by a subscript `j`, and let the index of the codebook vector nearest to the current training case be `n`. The Kohonen SOM algorithm requires a kernel function `K(j,n)`, where `K(j,j)=1` and `K(j,n)` is usually a non-increasing function of the distance (in any metric) between codebook vectors `j` and `n` in the grid space (not the input space). Usually, `K(j,n)` is zero for codebook vectors that are far apart in the grid space. As each training case is processed, all the codebook vectors are updated as:

```
new_codebook  = old_codebook  * [1 - K(j,n) * learning_rate]
           j               j

              + data * K(j,n) * learning_rate
```

The kernel function does not necessarily remain constant during training. The neighborhood of a given codebook vector is the set of codebook vectors for which `K(j,n)>0`. To avoid poor results (akin to local minima), it is usually advisable to start with a large neighborhood, and let the neighborhood gradually shrink during training. If `K(j,n)=0` for `j` not equal to `n`, then the SOM update formula reduces to the formula for Kohonen vector quantization. In other words, if the neighborhood size (for example, the radius of the support of the kernel function `K(j,n)`) is zero, the SOM algorithm degenerates into simple VQ. Hence it is important not to let the neighborhood size shrink all the way to zero during training. Indeed, the choice of the final neighborhood size is the most important tuning parameter for SOM training, as we will see shortly.

A SOM works by smoothing the codebook vectors in a manner similar to kernel estimation methods, but the smoothing is done in neighborhoods in the grid space rather than in the input space (Mulier and Cherkassky 1995). This is easier to see in a batch algorithm for SOMs, which is similar to Forgy's algorithm for batch k-means, but incorporates an extra smoothing process:

1. Initialize the codebook vectors.
2. Repeat the following two steps until convergence or boredom:
   A. Read the data, assigning each case to the nearest (using Euclidean distance) codebook vector. While you are doing this, keep track of the mean and the number of cases for each cluster.
   B. Do a nonparametric regression using `K(j,n)` as a kernel function, with the grid points as inputs, the cluster means as target values, and the number of cases in each cluster as an case weight. Replace each codebook vector with the output of the nonparametric regression function evaluated at its grid point.

If the nonparametric regression method is Nadaraya-Watson kernel regression (see What is GRNN?), the batch SOM algorithm produces essentially the same results as Kohonen's algorithm, barring local minima. The main difference is that the batch algorithm often converges. Mulier and Cherkassky (1995) note that other nonparametric regression methods can be used to provide superior SOM algorithms. In particular, local-linear smoothing eliminates the notorious "border effect", whereby the codebook vectors near the border of the grid are compressed in the input space. The border effect is especially problematic when you try to use a high degree of smoothing in a Kohonen SOM, since all the codebook vectors will collapse into the center of the input space. The SOM border effect has the same mathematical cause as the "boundary effect" in kernel regression, which causes the estimated regression function to flatten out near the edges of the regression input space. There are various cures for the edge effect in nonparametric regression, of which local-linear smoothing is the simplest (Wand and Jones, 1995). Hence, local-linear smoothing also cures the border effect in SOMs. Furthermore, local-linear smoothing is much more general and reliable than the heuristic weighting rule proposed by Kohonen (1995, p. 129).

Since nonparametric regression is used in the batch SOM algorithm, various properties of nonparametric regression extend to SOMs. In particular, it is well known that the shape of the kernel function is not a crucial matter in nonparametric regression, hence it is not crucial in SOMs. On the other hand, the amount of smoothing used for nonparametric regression is crucial, hence the choice of the final neighborhood size in a SOM is crucial. Unfortunately, I am not aware of any systematic studies of methods for choosing the final neighborhood size.

The batch SOM algorithm is very similar to the principal curve and surface algorithm proposed by Hastie and Stuetzle (1989), as pointed out by Ritter, Martinetz, and Schulten (1992) and Mulier and Cherkassky (1995). A principal curve is a nonlinear generalization of a principal component. Given the probability distribution of a population, a principal curve is defined by the following self-consistency condition:

1. If you choose any point on a principal curve,
2. then find the set of all the points in the input space that are closer to the chosen point than any other point on the curve,
3. and compute the expected value (mean) of that set of points with respect to the probability distribution, then
4. you end up with the same point on the curve that you chose originally.

See http://www.iro.umontreal.ca/~kegl/research/pcurves/ for more information about principal curves and surfaces.

In a multivariate normal distribution, the principal curves are the same as the principal components. A principal surface is the obvious generalization from a curve to a surface. In a multivariate normal distribution, the principal surfaces are the subspaces spanned by any two principal components.

A one-dimensional local-linear batch SOM can be used to estimate a principal curve

by letting the number of codebook vectors approach infinity while choosing a kernel function of appropriate smoothness. A two-dimensional local-linear batch SOM can be used to estimate a principal surface by letting the number of both rows and columns in the grid approach infinity. This connection between SOMs and principal curves and surfaces shows that the choice of the number of codebook vectors is not crucial, provided the number is fairly large.

If the final neighborhood size in a local-linear batch SOM is large, the SOM approximates a subspace spanned by principal components--usually the first principal component if the SOM is one-dimensional, the first two principal components if the SOM is two-dimensional, and so on. This result does not depend on the data having a multivariate normal distribution.

Principal component analysis is a method of data compression, not a statistical model. However, there is a related method called "common factor analysis" that is often confused with principal component analysis but is indeed a statistical model. Common factor analysis posits that the relations among the observed variables can be explained by a smaller number of unobserved, "latent" variables. Tibshirani (1992) has proposed a latent-variable variant of principal curves, and latent-variable modifications of SOMs have been introduced by Utsugi (1996, 1997) and Bishop, Svensén, and Williams (1997).

The choice of the number of codebook vectors is usually not critical as long as the number is fairly large. But results can be sensitive to the shape of the grid, e.g., square or an elongated rectangle. And the dimensionality of the grid is a crucial choice. It is difficult to guess the appropriate shape and dimensionality before analyzing the data. Determining the shape and dimensionality by trial and error can be quite tedious. Hence, a variety of methods have been tried for growing SOMs and related kinds of NNs during training. For more information on growing SOMs, see Bernd Fritzke's home page at http://pikas.inf.tu-dresden.de/~fritzke/

Using a 1-by-2 SOM is pointless. There is no "topological structure" in a 1-by-2 grid. A 1-by-2 SOM is essentially the same as VQ with two clusters, except that the SOM clusters will be closer together than the VQ clusters if the final neighborhood size for the SOM is large.

In a Kohonen SOM, as in VQ, it is necessary to reduce the learning rate during training to obtain convergence. Greg Heath has commented in this regard:

> I favor separate learning rates for each winning SOM node (or k-means cluster) in the form `1/(N_0i + N_i + 1)`, where `N_i` is the count of vectors that have caused node `i` to be a winner and `N_0i` is an initializing count that indicates the confidence in the initial weight vector assignment. The winning node expression is based on stochastic estimation convergence constraints and pseudo-Bayesian estimation of mean vectors. Kohonen

derived a heuristic recursion relation for the "optimal" rate. To my surprise, when I solved the recursion relation I obtained the same above expression that I've been using for years.

In addition, I have had success using the similar form `(1/n)/(N_0j + N_j + (1/n))` for the `n` nodes in the shrinking updating-neighborhood. Before the final "winners-only" stage when neighbors are no longer updated, the number of updating neighbors eventually shrinks to `n = 6` or `8` for hexagonal or rectangular neighborhoods, respectively.

Kohonen's neighbor-update formula is more precise replacing my constant fraction `(1/n)` with a node-pair specific `h_ij (h_ij < 1)`. However, as long as the initial neighborhood is sufficiently large, the shrinking rate is sufficiently slow, and the final winner-only stage is sufficiently long, the results should be relatively insensitive to exact form of `h_ij`.

Another advantage of batch SOMs is that there is no learning rate, so these complications evaporate.

Kohonen (1995, p. VII) says that SOMs are not intended for pattern recognition but for clustering, visualization, and abstraction. Kohonen has used a "supervised SOM" (1995, pp. 160-161) that is similar to counterpropagation (Hecht-Nielsen 1990), but he seems to prefer LVQ (see below) for supervised classification. Many people continue to use SOMs for classification tasks, sometimes with surprisingly (I am tempted to say "inexplicably") good results (Cho, 1997).

- LVQ: Learning Vector Quantization--competitive networks for supervised classification (Kohonen, 1988, 1995; Ripley, 1996). Each codebook vector is assigned to one of the target classes. Each class may have one or more codebook vectors. A case is classified by finding the nearest codebook vector and assigning the case to the class corresponding to the codebook vector. Hence LVQ is a kind of nearest-neighbor rule.

  Ordinary VQ methods, such as Kohonen's VQ or k-means, can easily be used for supervised classification. Simply count the number of training cases from each class assigned to each cluster, and divide by the total number of cases in the cluster to get the posterior probability. For a given case, output the class with the greatest posterior probability--i.e. the class that forms a majority in the nearest cluster. Such methods can provide universally consistent classifiers (Devroye et al., 1996) even when the codebook vectors are obtained by unsupervised algorithms. LVQ tries to improve on this approach by adapting the codebook vectors in a supervised way. There are several variants of LVQ--called LVQ1, OLVQ1, LVQ2, and LVQ3--based on heuristics. However, a smoothed version of LVQ can be trained as a feedforward network using a NRBFEQ architecture (see "How do MLPs compare with RBFs?") and optimizing any of the usual error functions; as the width of the RBFs goes to zero, the NRBFEQ network approaches an optimized LVQ network.

There are several other kinds of Kohonen networks described in Kohonen (1995), including:

- DEC--Dynamically Expanding Context
- LSM--Learning Subspace Method
- ASSOM--Adaptive Subspace SOM
- FASSOM--Feedback-controlled Adaptive Subspace SOM
- Supervised SOM
- LVQ-SOM

More information on the error functions (or absence thereof) used by Kohonen VQ and SOM is provided under "What does unsupervised learning learn?"

For more on-line information on Kohonen networks and other varieties of SOMs, see:

- The web page of The Neural Networks Research Centre, Helsinki University of Technology, at http://www.cis.hut.fi/research/
- The SOM of articles from comp.ai.neural-nets at http://websom.hut.fi/websom/comp.ai.neural-nets-new/html/root.html
- Akio Utsugi's web page on Bayesian SOMs at the National Institute of Bioscience and Human-Technology, Agency of Industrial Science and Technology, M.I.T.I., 1-1, Higashi, Tsukuba, Ibaraki, 305 Japan, at http://www.aist.go.jp/NIBH/~b0616/Lab/index-e.html
- The GTM (generative topographic mapping) home page at the Neural Computing Research Group, Aston University, Birmingham, UK, at http://www.ncrg.aston.ac.uk/GTM/
- Nenet SOM software at http://www.mbnet.fi/~phodju/nenet/nenet.html
- Bernd Fritzke's home page at http://pikas.inf.tu-dresden.de/~fritzke/ has information on growing SOMs and other related types of NNs

References:

Anderberg, M.R. (1973), *Cluster Analysis for Applications*, New York: Academic Press, Inc.

Balakrishnan, P.V., Cooper, M.C., Jacob, V.S., and Lewis, P.A. (1994) "A study of the classification capabilities of neural networks using unsupervised learning: A comparison with k-means clustering", Psychometrika, 59, 509-525.

Bishop, C.M., Svensén, M., and Williams, C.K.I (1997), "GTM: A principled alternative to the self-organizing map," in Mozer, M.C., Jordan, M.I., and Petsche, T., (eds.) *Advances in Neural Information Processing Systems 9*, Cambridge, MA: The MIT Press, pp. 354-360. Also see http://www.ncrg.aston.ac.uk/GTM/

Bottou, L., and Bengio, Y. (1995), "Convergence properties of the K-Means algorithms," in Tesauro, G., Touretzky, D., and Leen, T., (eds.) *Advances in Neural Information Processing Systems 7*, Cambridge, MA: The MIT Press, pp. 585-592.

Cho, S.-B. (1997), "Self-organizing map with dynamical node-splitting: Application to handwritten digit recognition," Neural Computation, 9, 1345-1355.

Desieno, D. (1988), "Adding a conscience to competitive learning," Proc. Int. Conf. on Neural Networks, I, 117-124, IEEE Press.

Devroye, L., Györfi, L., and Lugosi, G. (1996), *A Probabilistic Theory of Pattern Recognition*, NY: Springer,

Forgy, E.W. (1965), "Cluster analysis of multivariate data: Efficiency versus interpretability," Biometric Society Meetings, Riverside, CA. Abstract in Biomatrics, 21, 768.

Gersho, A. and Gray, R.M. (1992), Vector Quantization and Signal Compression, Boston: Kluwer Academic Publishers.

Hartigan, J.A. (1975), *Clustering Algorithms*, NY: Wiley.

Hartigan, J.A., and Wong, M.A. (1979), "Algorithm AS136: A k-means clustering algorithm," Applied Statistics, 28-100-108.

Hastie, T., and Stuetzle, W. (1989), "Principal curves," Journal of the American Statistical Association, 84, 502-516.

Hecht-Nielsen, R. (1990), *Neurocomputing*, Reading, MA: Addison-Wesley.

Ismail, M.A., and Kamel, M.S. (1989), "Multidimensional data clustering utilizing hybrid search strategies," Pattern Recognition, 22, 75-89.

Kohonen, T (1984), *Self-Organization and Associative Memory*, Berlin: Springer-Verlag.

Kohonen, T (1988), "Learning Vector Quantization," Neural Networks, 1 (suppl 1), 303.

Kohonen, T. (1995/1997), *Self-Organizing Maps*, Berlin: Springer-Verlag. First edition was 1995, second edition 1997. See http://www.cis.hut.fi/nnrc/new_book.html for information on the second edition.

Kosko, B.(1992), *Neural Networks and Fuzzy Systems*, Englewood Cliffs, N.J.: Prentice-Hall.

Linde, Y., Buzo, A., and Gray, R. (1980), "An algorithm for vector quantizer design," IEEE Transactions on Communications, 28, 84-95.

Lloyd, S. (1982), "Least squares quantization in PCM," IEEE Transactions on

Information Theory, 28, 129-137.

MacQueen, J.B. (1967), "Some Methods for Classification and Analysis of Multivariate Observations,"Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, 281-297.

Max, J. (1960), "Quantizing for minimum distortion," IEEE Transactions on Information Theory, 6, 7-12.

Mulier, F. and Cherkassky, V. (1995), "Self-Organization as an iterative kernel smoothing process," Neural Computation, 7, 1165-1177.

Ripley, B.D. (1996), *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University Press.

Ritter, H., Martinetz, T., and Schulten, K. (1992), *Neural Computation and Self-Organizing Maps: An Introduction*, Reading, MA: Addison-Wesley.

SAS Institute (1989), *SAS/STAT User's Guide*, Version 6, 4th edition, Cary, NC: SAS Institute.

Symons, M.J. (1981), "Clustering Criteria and Multivariate Normal Mixtures," Biometrics, 37, 35-43.

Tibshirani, R. (1992), "Principal curves revisited," Statistics and Computing, 2, 183-190.

Utsugi, A. (1996), "Topology selection for self-organizing maps," Network: Computation in Neural Systems, 7, 727-740, available on-line at http://www.aist.go.jp/NIBH/~b0616/Lab/index-e.html

Utsugi, A. (1997), "Hyperparameter selection for self-organizing maps," Neural Computation, 9, 623-635, available on-line at http://www.aist.go.jp/NIBH/~b0616/Lab/index-e.html

Wand, M.P., and Jones, M.C. (1995), *Kernel Smoothing*, London: Chapman & Hall.

Zador, P.L. (1982), "Asymptotic quantization error of continuous signals and the quantization dimension," IEEE Transactions on Information Theory, 28, 139-149.

Zeger, K., Vaisey, J., and Gersho, A. (1992), "Globally optimal vector quantizer design by stochastic relaxation," IEEE Transactions on Signal Procesing, 40, 310-322.

Subject: How are layers counted?

How to count layers is a matter of considerable dispute.

- Some people count layers of *units*. But of these people, some count the input layer and some don't.
- Some people count layers of *weights*. But I have no idea how they count skip-layer connections.

To avoid ambiguity, you should speak of a 2-hidden-layer network, not a 4-layer network (as some would call it) or 3-layer network (as others would call it). And if the connections follow any pattern other than fully connecting each layer to the next and to no others, you should carefully specify the connections.

Subject: What are cases and variables?

A vector of values presented at one time to all the input units of a neural network is called a "case", "example", "pattern, "sample", etc. The term "case" will be used in this FAQ because it is widely recognized, unambiguous, and requires less typing than the other terms. A case may include not only input values, but also target values and possibly other information.

A vector of values presented at different times to a single input unit is often called an "input variable" or "feature". To a statistician, it is a "predictor", "regressor", "covariate", "independent variable", "explanatory variable", etc. A vector of target values associated with a given output unit of the network during training will be called a "target variable" in this FAQ. To a statistician, it is usually a "response" or "dependent variable".

A "data set" is a matrix containing one or (usually) more cases. In this FAQ, it will be assumed that cases are rows of the matrix, while variables are columns.

Note that the often-used term "input vector" is ambiguous; it can mean either an input case or an input variable.

Subject: What are the population, sample, training set, design set, validation set, and test set?

It is rarely useful to have a NN simply memorize a set of data, since memorization can be done much more efficiently by numerous algorithms for table look-up. Typically, you want the NN to be able to perform accurately on new data, that is, to generalize.

There seems to be no term in the NN literature for the set of all cases that you want to be

able to generalize to. Statisticians call this set the "population". Tsypkin (1971) called it the "grand truth distribution," but this term has never caught on.

Neither is there a consistent term in the NN literature for the set of cases that are available for training and evaluating an NN. Statisticians call this set the "sample". The sample is usually a subset of the population.

(Neurobiologists mean something entirely different by "population," apparently some collection of neurons, but I have never found out the exact meaning. I am going to continue to use "population" in the statistical sense until NN researchers reach a consensus on some other terms for "population" and "sample"; I suspect this will never happen.)

In NN methodology, the sample is often subdivided into "training", "validation", and "test" sets. The distinctions among these subsets are crucial, but the terms "validation" and "test" sets are often confused. Bishop (1995), an indispensable reference on neural networks, provides the following explanation (p. 372):

> Since our goal is to find the network having the best performance on new data, the simplest approach to the comparison of different networks is to evaluate the error function using data which is independent of that used for training. Various networks are trained by minimization of an appropriate error function defined with respect to a *training* data set. The performance of the networks is then compared by evaluating the error function using an independent *validation* set, and the network having the smallest error with respect to the validation set is selected. This approach is called the *hold out method.* Since this procedure can itself lead to some overfitting to the validation set, the performance of the selected network should be confirmed by measuring its performance on a third independent set of data called a *test* set.

And there is no book in the NN literature more authoritative than Ripley (1996), from which the following definitions are taken (p.354):

Training set:
: A set of examples used for learning, that is to fit the parameters [i.e., weights] of the classifier.

Validation set:
: A set of examples used to tune the parameters [i.e., architecture, not weights] of a classifier, for example to choose the number of hidden units in a neural network.

Test set:
: A set of examples used only to assess the performance [generalization] of a fully-specified classifier.

The literature on machine learning often reverses the meaning of "validation" and "test" sets. This is the most blatant example of the terminological confusion that pervades artificial intelligence research.

The crucial point is that a test set, by the standard definition in the NN literature, is *never* used to choose among two or more networks, so that the error on the test set provides an unbiased estimate of the generalization error (assuming that the test set is representative of the population, etc.). Any data set that is used to choose the best of two or more networks is, by definition, a validation set, and the error of the chosen network on the validation set is optimistically biased.

There is a problem with the usual distinction between training and validation sets. Some training approaches, such as early stopping, require a validation set, so in a sense, the validation set is used for training. Other approaches, such as maximum likelihood, do not inherently require a validation set. So the "training" set for maximum likelihood might encompass both the "training" and "validation" sets for early stopping. Greg Heath has suggested the term "design" set be used for cases that are used solely to adjust the weights in a network, while "training" set be used to encompass both design and validation sets. There is considerable merit to this suggestion, but it has not yet been widely adopted.

But things can get more complicated. Suppose you want to train nets with 5 ,10, and 20 hidden units using maximum likelihood, and you want to train nets with 20 and 50 hidden units using early stopping. You also want to use a validation set to choose the best of these various networks. Should you use the same validation set for early stopping that you use for the final network choice, or should you use two separate validation sets? That is, you could divide the sample into 3 subsets, say A, B, C and proceed as follows:

- Do maximum likelihood using A.
- Do early stopping with A to adjust the weights and B to decide when to stop (this makes B a validation set).
- Choose among all 3 nets trained by maximum likelihood and the 2 nets trained by early stopping based on the error computed on B (the validation set).
- Estimate the generalization error of the chosen network using C (the test set).

Or you could divide the sample into 4 subsets, say A, B, C, and D and proceed as follows:

- Do maximum likelihood using A and B combined.
- Do early stopping with A to adjust the weights and B to decide when to stop (this makes B a validation set with respect to early stopping).
- Choose among all 3 nets trained by maximum likelihood and the 2 nets trained by early stopping based on the error computed on C (this makes C a second validation set).
- Estimate the generalization error of the chosen network using D (the test set).

Or, with the same 4 subsets, you could take a third approach:

- Do maximum likelihood using A.
- Choose among the 3 nets trained by maximum likelihood based on the error computed on B (the first validation set)
- Do early stopping with A to adjust the weights and B (the first validation set) to decide

when to stop.
- Choose among the best net trained by maximum likelihood and the 2 nets trained by early stopping based on the error computed on C (the second validation set).
- Estimate the generalization error of the chosen network using D (the test set).

You could argue that the first approach is biased towards choosing a net trained by early stopping. Early stopping involves a choice among a potentially large number of networks, and therefore provides more opportunity for overfitting the validation set than does the choice among only 3 networks trained by maximum likelihood. Hence if you make the final choice of networks using the same validation set (B) that was used for early stopping, you give an unfair advantage to early stopping. If you are writing an article to compare various training methods, this bias could be a serious flaw. But if you are using NNs for some practical application, this bias might not matter at all, since you obtain an honest estimate of generalization error using C.

You could also argue that the second and third approaches are too wasteful in their use of data. This objection could be important if your sample contains 100 cases, but will probably be of little concern if your sample contains 100,000,000 cases. For small samples, there are other methods that make more efficient use of data; see "What are cross-validation and bootstrapping?"

References:

Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press.

Ripley, B.D. (1996) *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University Press.

Tsypkin, Y. (1971), *Adaptation and Learning in Automatic Systems*, NY: Academic Press.

Subject: How are NNs related to statistical methods?

There is considerable overlap between the fields of neural networks and statistics. Statistics is concerned with data analysis. In neural network terminology, statistical inference means learning to generalize from noisy data. Some neural networks are not concerned with data analysis (e.g., those intended to model biological systems) and therefore have little to do with statistics. Some neural networks do not learn (e.g., Hopfield nets) and therefore have little to do with statistics. Some neural networks can learn successfully only from noise-free data (e.g., ART or the perceptron rule) and therefore would not be considered statistical methods. But most neural networks that can learn to generalize effectively from noisy data are similar or identical to statistical methods. For example:

- Feedforward nets with no hidden layer (including functional-link neural nets and higher-order neural nets) are basically generalized linear models.
- Feedforward nets with one hidden layer are closely related to projection pursuit regression.
- Probabilistic neural nets are identical to kernel discriminant analysis.
- Kohonen nets for adaptive vector quantization are very similar to k-means cluster analysis.
- Kohonen self-organizing maps are discrete approximations to principal curves and surfaces.
- Hebbian learning is closely related to principal component analysis.

Some neural network areas that appear to have no close relatives in the existing statistical literature are:

- Reinforcement learning (although this is treated in the operations research literature on Markov decision processes).
- Stopped training (the purpose and effect of stopped training are similar to shrinkage estimation, but the method is quite different).

Feedforward nets are a subset of the class of nonlinear regression and discrimination models. Statisticians have studied the properties of this general class but had not considered the specific case of feedforward neural nets before such networks were popularized in the neural network field. Still, many results from the statistical theory of nonlinear models apply directly to feedforward nets, and the methods that are commonly used for fitting nonlinear models, such as various Levenberg-Marquardt and conjugate gradient algorithms, can be used to train feedforward nets. The application of statistical theory to neural networks is explored in detail by Bishop (1995) and Ripley (1996). Several summary articles have also been published relating statistical models to neural networks, including Cheng and Titterington (1994), Kuan and White (1994), Ripley (1993, 1994), Sarle (1994), and several articles in Cherkassky, Friedman, and Wechsler (1994). Among the many statistical concepts important to neural nets is the bias/variance trade-off in nonparametric estimation, discussed by Geman, Bienenstock, and Doursat, R. (1992). Some more advanced results of statistical theory applied to neural networks are given by White (1989a, 1989b, 1990, 1992a) and White and Gallant (1992), reprinted in White (1992b).

While neural nets are often defined in terms of their algorithms or implementations, statistical methods are usually defined in terms of their results. The arithmetic mean, for example, can be computed by a (very simple) backprop net, by applying the usual formula SUM(x_i)/n, or by various other methods. What you get is still an arithmetic mean regardless of how you compute it. So a statistician would consider standard backprop, Quickprop, and Levenberg-Marquardt as different algorithms for implementing the same statistical model such as a feedforward net. On the other hand, different training criteria, such as least squares and cross entropy, are viewed by statisticians as fundamentally different estimation methods with different statistical properties.

It is sometimes claimed that neural networks, unlike statistical models, require no distributional assumptions. In fact, neural networks involve exactly the same sort of distributional assumptions as statistical models (Bishop, 1995), but statisticians study the consequences and importance of these assumptions while many neural networkers ignore them. For example, least-squares training methods are widely used by statisticians and neural networkers. Statisticians realize that least-squares training involves implicit distributional assumptions in that least-squares estimates have certain optimality properties for noise that is normally distributed with equal variance for all training cases and that is independent between different cases. These optimality properties are consequences of the fact that least-squares estimation is maximum likelihood under those conditions. Similarly, cross-entropy is maximum likelihood for noise with a Bernoulli distribution. If you study the distributional assumptions, then you can recognize and deal with violations of the assumptions. For example, if you have normally distributed noise but some training cases have greater noise variance than others, then you may be able to use weighted least squares instead of ordinary least squares to obtain more efficient estimates.

Hundreds, perhaps thousands of people have run comparisons of neural nets with "traditional statistics" (whatever that means). Most such studies involve one or two data sets, and are of little use to anyone else unless they happen to be analyzing the same kind of data. But there is an impressive comparative study of supervised classification by Michie, Spiegelhalter, and Taylor (1994), which not only compares many classification methods on many data sets, but also provides unusually extensive analyses of the results. Another useful study on supervised classification by Lim, Loh, and Shih (1999) is available on-line. There is an excellent comparison of unsupervised Kohonen networks and k-means clustering by Balakrishnan, Cooper, Jacob, and Lewis (1994).

There are many methods in the statistical literature that can be used for flexible nonlinear modeling. These methods include:

- Polynomial regression (Eubank, 1999)
- Fourier series regression (Eubank, 1999; Haerdle, 1990)
- Wavelet smoothing (Donoho and Johnstone, 1995; Donoho, Johnstone, Kerkyacharian, and Picard, 1995)
- K-nearest neighbor regression and discriminant analysis (Haerdle, 1990; Hand, 1981, 1997; Ripley, 1996)
- Kernel regression and discriminant analysis (Eubank, 1999; Haerdle, 1990; Hand, 1981, 1982, 1997; Ripley, 1996)
- Local polynomial smoothing (Eubank, 1999; Wand and Jones, 1995; Fan and Gijbels, 1995)
- LOESS (Cleveland and Gross, 1991)
- Smoothing splines (such as thin-plate splines) (Eubank, 1999; Wahba, 1990; Green and Silverman, 1994; Haerdle, 1990)
- B-splines (Eubank, 1999)
- Tree-based models (CART, AID, etc.) (Haerdle, 1990; Lim, Loh, and Shih, 1997; Hand, 1997; Ripley, 1996)

- Multivariate adaptive regression splines (MARS) (Friedman, 1991)
- Projection pursuit (Friedman and Stuetzle, 1981; Haerdle, 1990; Ripley, 1996)
- Various Bayesian methods (Dey, 1998)
- GMDH (Farlow, 1984)

Why use neural nets rather than any of the above methods? There are many answers to that question depending on what kind of neural net you're interested in. The most popular variety of neural net, the MLP, tends to be useful in the same situations as projection pursuit regression, i.e.:

- the number of inputs is fairly large,
- many of the inputs are relevant, but
- most of the predictive information lies in a low-dimensional subspace.

The main advantage of MLPs over projection pursuit regression is that computing predicted values from MLPs is simpler and faster. Also, MLPs are better at learning moderately pathological functions than are many other methods with stronger smoothness assumptions (see ftp://ftp.sas.com/pub/neural/dojo/dojo.html) as long as the number of pathological features (such as discontinuities) in the function is not too large. For more discussion of the theoretical benefits of various types of neural nets, see How do MLPs compare with RBFs?

Communication between statisticians and neural net researchers is often hindered by the different terminology used in the two fields. There is a comparison of neural net and statistical jargon in ftp://ftp.sas.com/pub/neural/jargon

For free statistical software, see the StatLib repository at http://lib.stat.cmu.edu/ at Carnegie Mellon University.

There are zillions of introductory textbooks on statistics. One of the better ones is Moore and McCabe (1989). At an intermediate level, the books on linear regression by Weisberg (1985) and Myers (1986), on logistic regression by Hosmer and Lemeshow (1989), and on discriminant analysis by Hand (1981) can be recommended. At a more advanced level, the book on generalized linear models by McCullagh and Nelder (1989) is an essential reference, and the book on nonlinear regression by Gallant (1987) has much material relevant to neural nets.

Several introductory statistics texts are available on the web:

- David Lane, *HyperStat*, at http://www.ruf.rice.edu/~lane/hyperstat/contents.html
- Jan de Leeuw (ed.), *Statistics: The Study of Stability in Variation* , at http://www.stat.ucla.edu/textbook/
- StatSoft, Inc., *Electronic Statistics Textbook*, at http://www.statsoft.com/textbook/stathome.html
- David Stockburger, *Introductory Statistics: Concepts, Models, and Applications*, at http://www.psychstat.smsu.edu/sbk00.htm
- University of Newcastle (Australia) Statistics Department, *SurfStat Australia*,

http://surfstat.newcastle.edu.au/surfstat/

A more advanced book covering many topics that are also relevant to NNs is:

- Frank Harrell, *REGRESSION MODELING STRATEGIES With Applications to Linear Models, Logistic Regression, and Survival Analysis*, at http://hesweb1.med.virginia.edu/biostat/rms/

References:

Balakrishnan, P.V., Cooper, M.C., Jacob, V.S., and Lewis, P.A. (1994) "A study of the classification capabilities of neural networks using unsupervised learning: A comparison with k-means clustering", Psychometrika, 59, 509-525.

Bishop, C.M. (1995), *Neural Networks for Pattern Recognition*, Oxford: Oxford University Press.

Cheng, B. and Titterington, D.M. (1994), "Neural Networks: A Review from a Statistical Perspective", Statistical Science, 9, 2-54.

Cherkassky, V., Friedman, J.H., and Wechsler, H., eds. (1994), *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, Berlin: Springer-Verlag.

Cleveland and Gross (1991), "Computational Methods for Local Regression," Statistics and Computing 1, 47-62.

Dey, D., ed. (1998) *Practical Nonparametric and Semiparametric Bayesian Statistics*, Springer Verlag.

Donoho, D.L., and Johnstone, I.M. (1995), "Adapting to unknown smoothness via wavelet shrinkage," J. of the American Statistical Association, 90, 1200-1224.

Donoho, D.L., Johnstone, I.M., Kerkyacharian, G., and Picard, D. (1995), "Wavelet shrinkage: asymptopia (with discussion)?" J. of the Royal Statistical Society, Series B, 57, 301-369.

Eubank, R.L. (1999), *Nonparametric Regression and Spline Smoothing*, 2nd ed., Marcel Dekker, ISBN 0-8247-9337-4.

Fan, J., and Gijbels, I. (1995), "Data-driven bandwidth selection in local polynomial: variable bandwidth and spatial adaptation," J. of the Royal Statistical Society, Series B, 57, 371-394.

Farlow, S.J. (1984), *Self-organizing Methods in Modeling: GMDH Type Algorithms*, NY: Marcel Dekker. (GMDH)

Friedman, J.H. (1991), "Multivariate adaptive regression splines", Annals of Statistics, 19, 1-141. (MARS)

Friedman, J.H. and Stuetzle, W. (1981) "Projection pursuit regression," J. of the American Statistical Association, 76, 817-823.

Gallant, A.R. (1987) *Nonlinear Statistical Models*, NY: Wiley.

Geman, S., Bienenstock, E. and Doursat, R. (1992), "Neural Networks and the Bias/Variance Dilemma", Neural Computation, 4, 1-58.

Green, P.J., and Silverman, B.W. (1994), *Nonparametric Regression and Generalized Linear Models: A Roughness Penalty Approach*, London: Chapman & Hall.

Haerdle, W. (1990), *Applied Nonparametric Regression*, Cambridge Univ. Press.

Hand, D.J. (1981) *Discrimination and Classification*, NY: Wiley.

Hand, D.J. (1982) *Kernel Discriminant Analysis*, Research Studies Press.

Hand, D.J. (1997) *Construction and Assessment of Classification Rules*, NY: Wiley.

Hill, T., Marquez, L., O'Connor, M., and Remus, W. (1994), "Artificial neural network models for forecasting and decision making," International J. of Forecasting, 10, 5-15.

Kuan, C.-M. and White, H. (1994), "Artificial Neural Networks: An Econometric Perspective", Econometric Reviews, 13, 1-91.

Kushner, H. & Clark, D. (1978), *Stochastic Approximation Methods for Constrained and Unconstrained Systems*, Springer-Verlag.

Lim, T.-S., Loh, W.-Y. and Shih, Y.-S. (1999?), "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," Machine Learning, forthcoming, preprint available at http://www.recursive-partitioning.com/mach1317.pdf, and appendix containing complete tables of error rates, ranks, and training times at http://www.recursive-partitioning.com/appendix.pdf

McCullagh, P. and Nelder, J.A. (1989) *Generalized Linear Models*, 2nd ed., London: Chapman & Hall.

Michie, D., Spiegelhalter, D.J. and Taylor, C.C., eds. (1994), *Machine Learning, Neural and Statistical Classification*, NY: Ellis Horwood; this book is out of print but available online at http://www.amsta.leeds.ac.uk/~charles/statlog/

Moore, D.S., and McCabe, G.P. (1989), *Introduction to the Practice of Statistics*, NY:

W.H. Freeman.

Myers, R.H. (1986), *Classical and Modern Regression with Applications*, Boston: Duxbury Press.

Ripley, B.D. (1993), "Statistical Aspects of Neural Networks", in O.E. Barndorff-Nielsen, J.L. Jensen and W.S. Kendall, eds., *Networks and Chaos: Statistical and Probabilistic Aspects*, Chapman & Hall. ISBN 0 412 46530 2.

Ripley, B.D. (1994), "Neural Networks and Related Methods for Classification," Journal of the Royal Statistical Society, Series B, 56, 409-456.

Ripley, B.D. (1996) *Pattern Recognition and Neural Networks*, Cambridge: Cambridge University Press.

Sarle, W.S. (1994), "Neural Networks and Statistical Models," *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, Cary, NC: SAS Institute, pp 1538-1550. (ftp://ftp.sas.com/pub/neural/neural1.ps)

Wahba, G. (1990), *Spline Models for Observational Data*, SIAM.

Wand, M.P., and Jones, M.C. (1995), *Kernel Smoothing*, London: Chapman & Hall.

Weisberg, S. (1985), *Applied Linear Regression*, NY: Wiley

White, H. (1989a), "Learning in Artificial Neural Networks: A Statistical Perspective," Neural Computation, 1, 425-464.

White, H. (1989b), "Some Asymptotic Results for Learning in Single Hidden Layer Feedforward Network Models", J. of the American Statistical Assoc., 84, 1008-1013.

White, H. (1990), "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings," Neural Networks, 3, 535-550.

White, H. (1992a), "Nonparametric Estimation of Conditional Quantiles Using Neural Networks," in Page, C. and Le Page, R. (eds.), Computing Science and Statistics.

White, H., and Gallant, A.R. (1992), "On Learning the Derivatives of an Unknown Mapping with Multilayer Feedforward Networks," Neural Networks, 5, 129-138.

White, H. (1992b), *Artificial Neural Networks: Approximation and Learning Theory*, Blackwell.

Next part is part 2 (of 7).