Choosing Network Complexity

B. D. Ripley
University of Oxford, 1 South Parks Road, Oxford OX1 3TG, UK.
E-mail: ripley@stats.ox.ac.uk

Abstract. For both belief networks and neural networks the network topology is normally chosen by the user. For belief networks the topology represents prior knowledge of conditional independence relationships, and for neural networks the topology (for example the number of hidden units) reflects some prior idea of complexity. We consider how it may be possible to choose the network complexity automatically, and also how sub-networks might be combined.

The vast majority of the literature on both Bayesian belief networks and neural networks assumes that the network is God-given, or at the very least is fixed for the purposes of statistical analysis. This is often an unrealistic assumption, and we consider how network might be induced from examples.

1 STATISTICAL NETWORK MODELS

This section merely sets our notation. A useful reference class of problems in considering these methods is classifying a patient as having one of K diseases (y) from the results ($\mathbf{x}$) of a set of tests with discrete outcomes. The data will be examples of test results (possibly incomplete) and the disease. They populate a contingency table of $(\mathbf{x}, y)$, and the methods here are designed to fit a parsimonious summary to that table and/or to classify future examples. Popular classical classes of models are *log-linear* models, and for binary outputs, *logistic* models.

1.1 Graphical models

Consider a graphical model in the sense of Whittaker (1990). The nodes of a graph represent random variables, and the edges represent (possible) dependence, the absence of edges (known) conditional independence. It can be interpreted through its *Markov properties*. For brevity we assume the technical condition that all probability densities are strictly positive. The *local* Markov property is that the conditional distribution of a random variable given all the others depends only on its neighbours on the graph. The whole joint distribution is specified by giving

$$P(X_a \mid X_b, b \neq a) = P(X_a \mid b \neq a, b \text{ is a nhbr of } a)$$

(proved by computing $P(\{X_a\} = \{y_a\})$ from $P(\{X_a\} = \{x_a\})$ by changing one variable at a time, e.g. Geman, 1990, Prop. 2.1). For the *global* Markov property, given sets of nodes A, B, C with no path from A to B not going through C, random variables on A and B must be conditionally independent given those in C. The special case $A = \{a\}$, $B = \{b\}$ and C comprising all nodes except a, b is called the *pairwise* Markov property. Each property implies the other two.

Note that whereas we can read off conditional independence properties from the graphical representation, there can be other conditional independence properties not represented on the graph. If none of the pairwise Markov properties implied by removing an edge hold, we have a minimal graphical representation, which Pearl (1988) calls a *Markov network*.

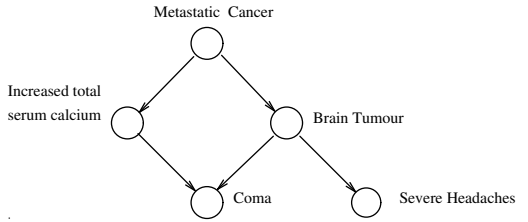

Fig. 1. A hypothetical medical belief network (after Cooper, 1984, and Pearl, 1988).

1.2 Causal probability networks

Graphical models can also be used to represent *causality* by including directed edges, as in figure 1. (There the interest is in the node 'brain tumour' given data on some of the other nodes.) For any node a we can consider the set of parents of a, that is those nodes from which a directed edge points to a. For a graph with only directed edges and which contains no cycle (a list of nodes each of which points to its successor, and the last points to the first) we can write the joint distribution of random variables $\{X_a\}$ by

$$P(\{X_a\}) = \prod_{\text{node } a} P(X_a \mid X_b, b \text{ a parent of } a) \tag{1}$$

Directed acyclic graphs are often called DAGs, and models (1) are sometimes called *recursive models*. The nodes in a DAG can always be ordered so that parents precede children. Then (1) is a restriction of the truism

$$P(\{X_a\}) = \prod_{\text{node } a} P(X_a \mid X_b, b < a) \tag{2}$$

the missing variables in the conditioning sets in (1) reflecting assumptions of conditional independence.

Conditional independence properties of a DAG can be read from a corresponding undirected graph, the *moral graph* (Lauritzen & Spiegelhalter, 1988; Whittaker, 1990, §3.5), which has all the edge directions dropped, and edges added between parents with a common child[1]. This idea allows us to extend the interpretation to acyclic graphs with some directed and some undirected edges, called *chain independence graphs*.

Graphical models are principally used by Whittaker (1990) as a way to display conditional independence relationships. They have been exploited powerfully by the BAIES group (see the review paper of Spiegelhalter *et al.*, 1993) who have developed message-passing algorithms for exploiting the network structure to compute, for example marginal probabilities and the effects of observing (and hence conditioning on) a currently unobserved variable. That area is called *belief networks, Bayesian networks* or *causal probability nets*. It arose simultaneously in an AI context through the work of Judea Pearl, culminating in Pearl (1986, 1988, 1993).

The simplest form of belief network is a tree, and it is easiest to understand how message-passing might work for a tree, since each node has at most one parent and the structure is a series of linked chains to which we apply hierarchical Bayes inference. Indeed, Lauritzen & Spiegelhalter's algorithms depend on grouping nodes into super-nodes whose graph is a tree. Pearl extends his algorithms to *polytrees*, directed graphs in which at most one path exists between any two nodes (which *inter alia* excludes cycles). The motivation for polytrees is that they allow the leaves (terminal nodes with causes but no effects) to be connected to several logically independent causal trees.

1.3 Where do the probabilities come from?

To use either Markov or belief networks we must specify the conditional distributions of each variable given its neighbours or its parents, respectively. In many applications these probabilities are elicited from experts (and that such probabilities can be incorporated is a considerable strength of these networks). In the traditional statistical approach they are specified as parameters in the model and estimated by maximum likelihood.

There has been recent work on combining both points of view and updating conditional probability tables from data. Suppose that θ_a denotes the parameters in the conditional distribution of X_a. Then Bayesian inference can be performed on the joint distribution of all variables and parameters, *provided* we can give a prior over all the θ's. *If* we assume that the prior gives independence between nodes, then (1) becomes

$$P(\{X_a\}, \{\theta_a\}) = \prod_{\text{node } a} P(X_a \mid X_b, b \text{ a parent of } a) P(\theta_a) \qquad (3)$$

and Spiegelhalter & Lauritzen (1990) pointed that this can be represented by an enlarged DAG in which each θ_a is another node with a single edge pointing to a. Their computational technology can then be applied to updating this distribution.

Let us consider only discrete random variables with a finite number of states (which is a very common restriction in applications). Then specifying the conditional distribution amounts to specifying a multinomial distribution for each of a finite set of states of the neighbours/parents. The most natural parametrization is a Dirichlet distribution for the probability distribution specifying the multinomial, as this is the conjugate prior (Berger, 1985, pp. 287, 561). The calculations are then reasonably straightforward *provided* there are no missing values in the data; some suggestions for that case are given by Spiegelhalter *et al.* (1993).

2 CHOOSING THE TOPOLOGY OF A BELIEF NETWORK

In expert-systems studies (such as those surveyed by Pearl, 1988, and Spiegelhalter *et al.* 1993) the network topology, that is the DAG for a belief network, is also elicited from experts by specifying the perceived conditional independence properties. Is there any hope to induce the network from examples? There have been a number of approaches.

From the traditional statistical viewpoint a graphical model represents a restriction of a full dependence, for example (and most commonly) a log-linear model for a contingency table. The model-building strategy is to move around the space of possible models, and eventually to select a small number of simplified models which are complex enough to explain the patterns in the examples, yet have comprehensible interpretations. (It is this last point which often leads to a consideration of only subsets of graphical models.) The strategy can involve either a series of hypothesis tests or searching to minimize a criterion which minimizes a penalized fit criterion such as Akaike's AIC. (Lauritzen *et al.*, 1992, give a case study of both.) This approach usually involves a great deal of user control, although is beginning to be partially automated. Since interpretability is paramount, a domain-expert will always be needed to monitor the process.

Another tradition from the social sciences for inferring a causal network from data in a regression setting is represented by Glymour *et al.* (1987) and Spirtes *et al.* (1993).

Chow & Liu (1968) (see also Pearl, 1988, chapter 8) extend the contingency-table approach by seeking the belief *tree* that best approximates (in Kullback-Leibler divergence) a general multi-way discrete distribution. They show that:

- Given the tree topology, the conditional distribution of each node given its parent in the best approximation is that in original distribution, and

[1] All conditional independence properties are preserved (Lauritzen *et al.*, 1990) but not all need be represented on the graph. Pearl's equivalent notion of d-separation extracts conditional independence properties directly from the DAG.

- The best tree is any minimal spanning tree with distance between a and b given by the mutual information of (X_a, X_b),

$$I(X_a, X_b) = E \log \frac{P(X_a, X_b)}{P(X_a)P(X_b)}$$

Since there are efficient algorithms for finding minimum spanning trees this allows a best approximating belief tree to be constructed. (Note that the choice of root of the tree is arbitrary.) Pearl (1988) describes his extension of this algorithm to recovering polytrees (without the approximation property); this is extended by Geiger *et al.* (1990).

Full-blooded Bayesians would put a prior on the topology of belief networks, and then average the results over belief networks weighted by their posterior probabilities. This principle has been pursued by Cooper & Herskovits (1992), in tandem with learning the conditional probability tables from data. There is an enormous number of different networks; for example Cooper & Herskovits (1992, p. 319) quote approximately 4.2×10^{18} for ten nodes. Cooper & Herskovits make some assumptions on the prior on topologies (such as a uniform distribution) to simplify computation. All such assumptions are unrealistic, as considering that set of nodes implies a belief in a sparsely connected network. (However, the prior may be swamped by the data and so be practically irrelevant.) My own prior might often be approximated by one on the number of edges.

The full Bayesian approach is normally computationally impossible as we cannot average over all topologies. Full averaging can be replaced by averaging over the few most plausible topologies, maybe even just the most plausible. This can be considered as the approach of traditional model selection and of Chow & Liu, with slightly different measures of plausibility. (Cooper & Herskovits suggest how to calculate the most plausible topology.) It is possible to use simulation methods, but they will also only average over a small subset of the topologies, so the method will need to be carefully constructed to give useful results.

Other approaches are due to Pearl & Verma (1991), Verma & Pearl (1991) and, for Markov networks, Fung & Crawford (1990).

2.1 Hidden variables

A Markov or belief network can have one or more nodes representing unobserved latent variables. This device is widely used in medical applications, for example to represent the true (rather than reported) test result. They will cause observations to have missing values, and so complicate the learning of conditional probability tables. Unsurprisingly, the EM algorithm and variations have been used (Spiegelhalter *et al.*, 1993, section 7).

Hidden variables can also be allowed within topologies inferred from data, in which case their interpretation is not specified in advance. Pearl (1988, section 8.3) considers hidden nodes in trees, and other methods have been developed.

3 NEURAL NETWORK MODELS

Neural networks have been very widely used for classification in the last five years (Ripley, 1993, 1994a,b; Ripley & Hjort, 1994; Weiss & Kulikowski, 1991; Hertz *et al.*, 1991) and to create expert systems (Gallant, 1993). Their most widely used form is as a model for the posterior probabilities of the class Y given the input features $\mathbf{X} = \mathbf{x}$ of the form

$$\log P(Y = k \mid \mathbf{X} = \mathbf{x}) = \alpha_k + \sum_j w_{jk}\, \phi\Big(\alpha_j + \sum_i w_{ij}\, x_i\Big) \qquad (4)$$

where ϕ is the logistic function $\phi(x) = e^x/(1 + e^x)$, $\{x_i\}$ are the input variables, and $\{x_h\}$ are the variables at nodes in a single *hidden layer*. This is data-flow network, with the hidden variables often fancifully described as representing a degree of abstraction of the problem. Think of this as a non-linear version of a log-linear model. (In fact variations of (4), notably a separate logistic model for each class *versus* the rest, are more commonly used.)

The right-hand side of (4) parametrizes a very flexible class of functions; they can uniformly approximate continuous functions, approximate L_2 functions in mean-square and so on. The price is that we need to increase the number H of hidden units, hence the number of parameters (called *weights*) and the size of the sum over j in (4).

If we allow the function ϕ in (4) to depend on the unit and be an arbitrary smooth function, we have *projection pursuit regression* (Friedman & Stuetzle, 1981) for which earlier approximation results are known (Diaconis & Shahshahani, 1984)

Since the right-hand side of (4) is so flexible, we have to control it to avoid over-fitting the examples to hand. The major ways to do so are to limit the number of hidden units (similar to fixing the degree of a polynomial approximation) and to *regularize*, effectively to put a prior on the space of fitted functions expressing a belief in simplicity (for example, the weight-decay prior gives independent Gaussian beliefs in the weights, and so penalizes high slopes). Explicitly, (4) is fitted by minimizing the deviance plus a penalty such as

$$E = \sum_{\text{examples}} -2 \log p(y \mid \mathbf{x}) + \frac{1}{\sigma_w^2} \sum_{\text{weights}} w_{ij}^2 \qquad (5)$$

and extensions which allow different variances for different classes of weights.

3.1 Constructing neural network topologies

Most of the techniques of statistical model building have been applied here too. Analogues of AIC have been sought, such as Moody's (1992) *effective number of parameters* $p^\star$ and Amari's NIC (Amari, 1993; Murata *et al.*, 1991 and references therein) which are essentially the same. They come from considering the large-sample statistical theory of the problem of estimating the parameters (the weights

in a neural network) *assuming the model to be false*. Then

$$p^* = \text{trace}(GQ^{-1})$$

where Q is the expected Hessian matrix of the log-likelihood and G is the expected value of the outer product of the score functions (the first derivatives of the log-likelihood), both evaluated at the best-fitting set of parameters (in Kullback-Leibler divergence). It is well-known that if the model were true, $G = Q$ and so $p^* = p$, the actual number of parameters, but in general p^* can be *very* much less.

In general, G and Q have to be estimated, and this is usually done by replacing the expected values by averages over the training set, and the best-fitting set of weights by the fitted set. This is somewhat dangerous, as the estimates can be too low due to over-fitting. (Perhaps cross-validation should be used – see section 4.) Note too that these calculations are for maximum-likelihood estimation only, and can not take account of regularization penalties.

The standard algorithm (known as SMART) for projection pursuit regression uses forward and then backward selection; another forward selection algorithm for neural networks is *Cascade Correlation* (Fahlman & Lebiere, 1990). Hwang *et al.* (1993) argue that the SMART algorithm is superior.

The issue of the 'correct' Bayesian approach to neural networks has raised arguments of religious fervour – Buntine & Weigend (1991), MacKay (1992a, b, c) and Wolpert (1993). The hierarchical Bayesian approach is to integrate out the uncertainty in the weights in (4). The parameter σ_w^2 in (5) is a hyperparameter, and controversy revolves around whether this should be fixed, given a hyperprior, or estimated (ML-II empirical Bayes; Berger, 1985, p. 99). The integration over weights can be done by Monte Carlo (as in Neal, 1993) or by various approximations such as a local quadratic about the maximizer of E. However, there is another layer, that of choosing network complexity, where most protagonists choose a model but the full Bayesian approach to classification would be to average over models. As in all the other Bayesian approaches reviewed here, computational considerations demand approximations in finding the posterior probabilities, and establishing the adequacy of those approximations is difficult (and rarely attempted).

4 COMBINING NETWORKS

One way to produce complex networks is to combine simpler ones, and we will consider this in the context of neural networks. Wolpert (1992) had a central insight which he termed *stacked generalization*. Suppose we have a set of possible classifiers, possibly several neural networks, a k-nearest neighbour rule and so on. Wolpert's idea is to construct and train another classifier with their outputs as its inputs — that is the second-level classifier arbitrates between the first-level classifiers.

How could we ever train the second-level classifier? This is done by taking a training set, and noting the true output for each of the sets of predicted inputs for the first-level classfiers. We then have to learn this relationship, for example by taking a convex combination of the output probabilities from level one, or by using a second-level neural network. Note that a convex combination is exactly what the full-blooded Bayesian paradigm demands, the weights being the posterior probabilities of the models underlying the level-one classifiers (which must also be fully Bayesian). However, these weights often demand computationally intractable integrations, so we can view stacked generalization as a (possibly crude but robust) way to learn the weights.

Using a separate test set to train the level-two classifer may be wasteful of data, so it is commoner to use cross-validation to generate the training data. That is, each case in the training set is removed in turn, all the level-one classifiers (re-)trained on the reduced data set and made to predict the removed case. (It may be better both computationally and statistically to remove groups of cases; for example 10-fold cross-validation removes 10% of the data at a time.) LeBlanc & Tibshirani (1993) consider 'bootstrap' methods rather than cross-validation.

The level-two classifier needs to be very 'smooth', and heavy regularization and/or a very simple network structure is used. Using a classification tree at level two essentially partitions the input space amongst the classifiers.

A similar approach is taken by Jacobs *et al.* (1991). They combine simple (log-linear) classifiers by using a convex combination of their outputs selected by another log-linear classifer (which they term a *gating* network. This is the same set-up as a special case of stacked generalization, *but* they train all the classifiers simultaneously, using the EM algorithm. That is, the level-one classifiers are not required to work well over the whole input space, but can be learn specialist functions. Jordan and Jacobs (1993) extend this methodology to further layers (as indeed stacking can be repeated in stacked generalization).

Acknowledgements

I am grateful to Wray Buntine, Greg Cooper and David Spiegelhalter for providing insight and references. The preparation of this paper was done whilst visiting the *Isaac Newton Institute for the Mathematical Sciences* in Cambridge, England.

REFERENCES

Amari, S.-I. (1993) Statistical neurodynamics of various types of associative nets. In *Networks and Chaos – Statistical and Probabilistic Aspects* (eds O. E. Barndorff-Nielsen, J. L. Jensen & W. S. Kendall), Chapman & Hall, London, pp. 1–39.

Berger, J. O. (1985) *Statistical Decision Theory and Bayesian Analysis,* Springer, New York.

Buntine, W. L. and Weigend, A. S. (1991) Bayesian back-propagation. *Complex Systems* **5**, 603–643.

Chow, C. K. and Liu, C. N. (1968) Approximating discrete probability distributions with dependence trees. *IEEE Trans. Inform. Theory* **14**, 462–467.

Cooper, G. F. (1984) *NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge,* Ph.D. thesis, Dept. of Computer Science, Stanford University.

Cooper, G. F. and Herskovits, E. (1992) A Bayesian method for the induction of probabilistic networks from data. *Machine Learning* **9**, 309–347.

Diaconis, P. and Shahshahani, M (1984) On non-linear functions of linear combinations. *SIAM J. Sci. Statist. Comput.* **5**, 175–191.

Fahlman, S. E. and Lebiere, C. (1990) The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2 (Denver, 1989)* (ed. D. S. Touretzky), Morgan Kaufmann, San Mateo, CA, pp. 524–532.

Friedman, J. H. and Stuetzle, W. (1981) Projection pursuit regression. *J. Amer. Statist. Soc.* **76**, 817–823.

Fung, R. and Crawford, S. L. (1990) A system for induction of probabilistic models. In *Proc. Eighth National Conf. Artificial Intelligence, Boston,* The MIT Press, Cambridge, MA, pp. 762–769.

Gallant, S. L. (1993) *Neural Network Learning and Expert Systems,* The MIT Press, Cambridge, MA.

Geiger, D., Paz, A. and Pearl, J. (1990) Learning dependence trees from causal information. In *Proc. Eighth National Conf. Artificial Intelligence, Boston*, The MIT Press, Cambridge, MA, pp. 770–776.

Geman, D. (1990) Random fields and inverse problems in imaging. In *École d'Été de Probabilités de Saint-Flour XVIII – 1988* (ed. P. L. Hennequin) *Lecture Notes in Math.* **1427**, 113–193.

Glymour, C., Scheines, R., Spirtes, P. and Kelley, K. (1987) *Discovering Causal Structure: Artificial Intelligence, Philosophy of Science, and Statistical Modeling,* Academic Press, New York.

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

Hwang, J.-N., You, S.-.s, Lay, S.-R. and Jou, I-C. (1993) What's wrong with a cascaded correlation learning network: a projection pursuit learning perspective. Technical Report, Dept. of Electrical Engineering, University of Washington.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J. and Hinton, G. E. (1991) Adaptive mixtures of local experts. *Neural Computation* **3**, 79–87.

Jordan, M. I. and Jacobs, R. A. (1993) Hierarchical mixtures of experts and the EM algorithm. Submitted to *Neural Computation*.

Lauritzen, S. L. and Spiegelhalter, D. J. (1988) Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *J. Roy. Statist. Soc. B* **50**, 157–224.

Lauritzen, S. L., Dawid, A. P., Larsen, B. N. and Leimer, H.-G. (1990) Independence properties of directed Markov fields. *Networks* **20**, 491–505.

Lauritzen, S.-L., Thiesson, B. and Spiegelhalter, D. G. (1992) Diagnostic systems created by model selection methods – a case study. Institute for Electronic Systems, University of Aalborg technical report R 92-2018.

LeBlanc, M. and Tibshirani, R. (1993) Combining estimates in regression and classification. Technical report, Department of Statistics, University of Toronto.

MacKay, D. J. C. (1992a) Bayesian model comparison and backprop nets. In *Advances in Neural Information Processing Systems 4* (eds J. E. Moody, S. J. Hanson and R. P. Lippmann), Morgan Kaufmann, San Mateo, CA, pp. 839–846.

MacKay, D. J. C. (1992b) A practical Bayesian framework for backprop networks. *Neural Computation* **4**, 448–472.

MacKay, D. J. C. (1992c) The evidence framework applied to classification networks. *Neural Computation* **4**, 720–736.

Moody, J. E. (1992) The *effective* number of parameters: an analysis of generalization and regularization in non-linear learning systems. In *Advances in Neural Information Processing Systems 4* (eds J. E. Moody, S. J. Hanson and R. P. Lippmann), Morgan Kaufmann, San Mateo, CA, pp. 847–854.

Murata, N., Yoshizawa, S. and Amari, S. (1991) A criterion for determining the number of parameters in an artificial neural networks model. In *Artificial Neural Networks* (eds T. Kohonen, K. Mäkisara, O. Simula and J. Kangas), North Holland, Amsterdam, pp. 9–14.

Neal, R. (1993) Bayesian learning via stochastic dynamics. In *Advances in Neural Information Processing Systems 5* (eds C. L. Giles, S. J. Hanson, and J. D. Cowan), Morgan Kaufmann, San Mateo, CA, pp. 475–482.

Pearl, J. (1986) Fusion, propagation, and structuring in belief networks. *Artificial Intelligence* **29**, 241–288.

Pearl, J. (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference,* Morgan Kaufmann, San Mateo, CA.

Pearl, J. (1993) Belief networks revisited. *Artificial Intelligence* **59**, 49–56.

Pearl, J. & Verma, T. S. (1991) A theory of inferred causality In *Proc. Second. Int. Conf. on the Principles of Knowledge Representation and Reasoning, Boston* (eds J. Allen, R. Fikes and E. Sandewall), Morgan Kaufmann, San Mateo, CA, pp. 441–451.

Ripley, B. D. (1993) Statistical aspects of neural networks. In *Networks and Chaos – Statistical and Probabilistic Aspects* (eds O. E. Barndorff-Nielsen, J. L. Jensen and W. S. Kendall), Chapman & Hall, London, pp. 40–123.

Ripley, B. D. (1994a) Neural networks and related methods for classification. *J. Roy. Statist. Soc. B* **56**, 409–456.

Ripley, B. D. (1994b) Flexible non-linear approaches to classification. In *From Statistics to Neural Networks* (eds V. Cherkassky, J. H. Friedman and H. Wechsler), Springer.

Ripley, B. D. and Hjort, N. L. (1994) *Pattern Recognition and Neural Networks,* Cambridge University Press, Cambridge.

Spiegelhalter, D. J. and Lauritzen, S. L. (1990) Sequential updating of conditional probabilities on directed graphical structures. *Networks* **20**, 579–605.

Spiegelhalter, D. J., Dawid, A. P., Lauritzen, S. L. and Cowell, R. G. (1993) Bayesian analysis in expert systems. *Statistical Science* **8**, 219–283.

Spirtes, P., Glymour, C. and Scheines, R. (1993) *Causality, Prediction and Search, Lecture Notes in Statistics* **87**, Springer-Verlag, New York.

Verma, T. and Pearl, J. (1991) Equivalence and synthesis of causal models. *Uncertainty in Artificial Intelligence 6*, (ed. P. P. Bonissone), North Holland, Amsterdam, pp. 255-268.

Weiss, S. M. and Kulikowski, C. A. (1991) *Computer Systems that Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems,* Morgan Kaufmann, San Mateo, CA.

Whittaker, J. (1990) *Graphical Models in Applied Multivariate Statistics,* Wiley, Chichester.

Wolpert, D. H. (1992) Stacked generalization. *Neural Networks* **5**, 241–259.

Wolpert, D. H. (1993) On the use of evidence in neural networks. In *Advances in Neural Information Processing Systems 5* (eds C. L. Giles, S. J. Hanson, and J. D. Cowan), Morgan Kaufmann, San Mateo, CA.