Statistical Ideas for Selecting Network Architectures

B. D. Ripley
Professor of Applied Statistics, University of Oxford
Oxford, UK

Abstract

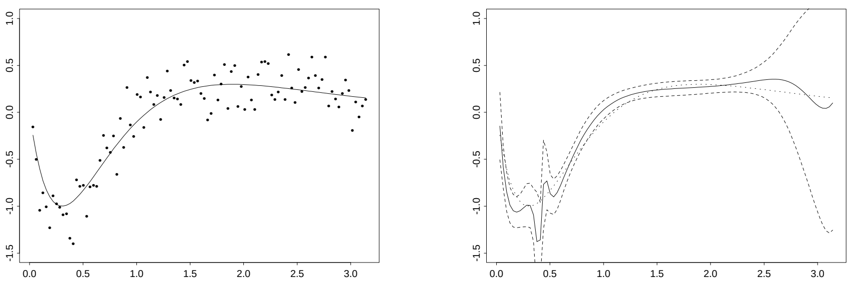
Choosing the architecture of a neural network is one of the most important problems in making neural networks practically useful, but accounts of applications usually sweep these details under the carpet. How many hidden units are needed? Should weight decay be used, and if so how much? What type of output units should be chosen? And so on.

We address these issues within the framework of statistical theory for model choice, which provides a number of workable approximate answers.

This paper is principally concerned with architecture selection issues for feed-forward neural networks (also known as multi-layer perceptrons). Many of the same issues arise in selecting radial basis function networks, recurrent networks and more widely. These problems occur in a much wider context within statistics, and applied statisticians have been selecting and combining models for decades. Two recent discussions are [4, 5]. References [3, 20, 21, 22] discuss neural networks from a statistical perspective.

1 A smoothing example

Figure 1(a) shows a dataset which is a simulation of an example from [29] where it illustrates smoothing splines. Throughout we assume that the true noise variance, $\sigma^2 = (0.2)^2$, is known.



(a) (b)

Figure 1: (a) 100 data points from a curve-fitting problem, with the true curve. (b) Curve fitted by 1–8–1 neural network, with ± 2 standard error bands and true curve (dotted).

Figure 1(b) shows a fitted curve from a neural network with one linear output unit, least-squares fitting and 8 hidden units. The limits shown are ± 2 standard errors, obtained from the theory of non-linear regression (based on local linearization: [25]) since this is a non-linear regression problem. Clearly there is over-fitting, and as Figure 2 shows, there are many local minima of the fitting function. (These *are* local minima; the Hessian was confirmed to be positive definite.) Using weight decay provides one way to combat over-fitting; least squares is replaced by minimizing

$$\sum_{i=1}^{n} \big[y_i - f(x_i; \mathbf{w}) \big]^2 + \lambda \sum_{ij} w_{ij}^2 = S(\mathbf{w}) + \lambda \|\mathbf{w}\|^2, \tag{1}$$

possibly omitting the 'bias' weights or giving them a proportionally lower penalty (as here).

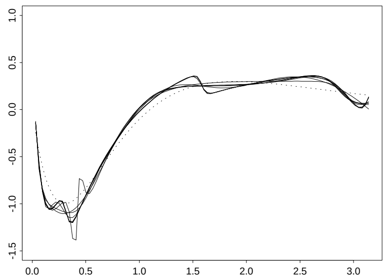
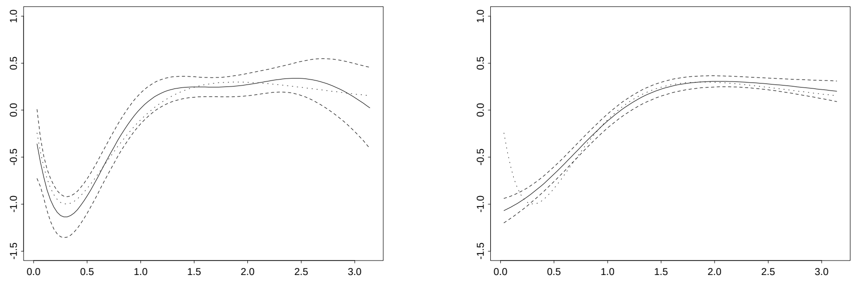


Figure 2: Curves from 1–8–1 neural networks fitted from different starting weights.



(a) (b)

Figure 3: Curve fitted by 1–8–1 neural network, with ± 2 standard error bands and true curve. (a) $\lambda = 10^{-3}$ and (b) $\lambda = 0.1$.

Figure 3 shows the result of using weight decay. Without knowing the truth, we cannot be sure which part is correct. These weight decay figures were not picked from thin air, as a Bayesian argument ([22, 24]) gives some guidance, suggesting $(0.002–0.2)\sigma^2$ for a regression network. When weight decay is used, multiple minima can still occur, although none were found here.

The effect of weight decay is to reduce the variability of the fit, at the cost of bias, since the fitted curve will be smoother than the true curve. Model selection is choosing a point on the bias/variance compromise (in the terminology of [6]). Thus it is surprising that the standard errors in Figure 1(b) are tighter than those in Figure 3(a). In fact the former are optimistic: the local linearization breaks down too quickly, and in any case does not take into account the other local minima.

We now have two ways to control the smoothness of the fit, the number of hidden units and λ. The two interact, as for moderate values of λ the number of hidden units has very little effect on the fitted function. (We might do better to use a direct *regularization* penalty, such as the $\lambda \int (f'')^2$ used in spline smoothing, see [2].) How do we choose the degree of smoothness from the data alone?

2 Cross-validation, NIC and p_{eff}

Cross-validation ([27]) is perhaps the most widely used method to select architectural parameters, but the term is often mis-used in the neural networks field. In the language of pattern recognition ([24]), the n available examples may be divided into training, validation and test sets. The training set is used to train the network, that is to choose its parameters (weights). The performance on the validation set is used to select one of a competing family of networks (or a combination of them). Finally, the test set is used to measure the performance of the selected network(s). Note that the validation set must be distinct from the test set for the assessed performance to be valid.

Since examples are normally not abundant (and large validation and test sets are needed to measure performance well), cross-validation divides the available samples up and re-uses them. In its simplest form (V-fold CV), the examples are divided into V roughly comparable sets, and each is used in turn as a validation set with the remainder used as the training set. This allows every example to be used for validation. Leave-one-out cross-validation (LOOCV) takes $V = n$, so predicts each example from the network trained on the remaining examples. We can use cross-validation as a test-set method for one network, but if we wish to both validate and test we need a double layer of cross-validation.

Cross-validation can be very CPU consuming for neural networks (especially LOOCV). The presence of local minima causes difficulties; we can attempt to track local minima by starting at a solution to the full problem ([14, 15]), but removing one outlying point can change dramatically the structure of the local minima.

References [28, 12, 13, 16, 17, 18] take another approach, a generalization of Akaike's AIC criterion. The *deviance* is minus twice the log-likelihood; in a regression problem with known σ^2 this is the least-squares measure S divided by σ^2. (If σ^2 is *not* known, the deviance is $n \log(S/n)$.) Then

$$\text{AIC} = \text{deviance} + 2p$$

where p is the total number of parameters. This is only appropriate for maximum likelihood (here least-squares) fitting; when a regularizer such as weight decay is used, p

must be replaced by $p^* = \mathrm{trace}[KJ^{-1}]$ where

$$J = -\mathsf{E}\frac{\partial^2 g(X_i, \theta_0)}{\partial\theta\,\partial\theta^T} \qquad \text{and} \qquad K = \mathsf{Var}\frac{\partial g(X_i, \theta_0)}{\partial\theta}.$$

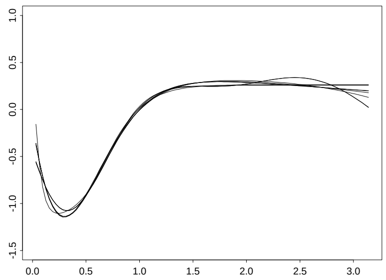
Here the averages are done under the true distribution, θ_0 is the 'least false' parameter (the best possible fit to the true distribution) and g is the log density plus the regularization divided by n. (This follows from standard statistical theory; [7, 24]. Without regularization $p^* = p$.) These quantities can be estimated from the derivatives and the Hessian evaluated at the fit to the training set, replacing θ_0 by $\widehat{\theta}$ and expectations by averages over the training set. Moody's p_{eff} is p^*, and this version of AIC is called NIC. The underlying idea of NIC/AIC is to estimate the deviance for a test set of size n, compensating for the fact that the weights were chosen to fit the training set.

Stone in [28] proved (but did not actually state) that for large training sets LOOCV and NIC are equivalent, and this seems to work well in practice. Thus NIC gives the effect of LOOCV at much less computational effort. However, the theory of NIC relies on a single well-defined minimum to the fitting function, and it can be unreliable when there are several local minima. Leave-one-out cross-validation will also often not leave a single local minimum, and makes order $1/n$ perturbations, which are too small to assess the fit accurately. In consequence, LOOCV has the reputation of being very variable, and 10-fold cross-validation is usually a better guide.

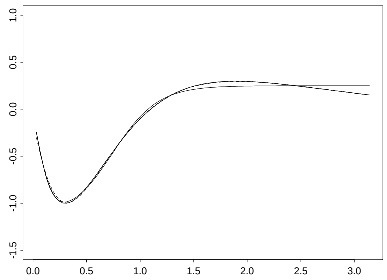
Size	λ	SSq	SSq$_{CV}$	p^*	NIC	Bayes
2	0	3.47	4.29	7	4.03	—
2	10^{-3}	3.57	4.78	5	3.98	3.30
3	0	4.12	4.81	10	4.92	—
3	10^{-3}	3.40	4.54	6	3.93	3.30
3	10^{-2}	3.61	4.91	6	4.11	3.72
3	10^{-1}	5.40	6.20	6	5.94	6.08
5	0	3.14	4.57	16	4.42	—
5	10^{-3}	3.21	4.51	17	4.59	3.38
5	10^{-2}	3.61	4.91	6	4.11	4.11
5	10^{-1}	5.38	6.18	6	5.92	5.98
8	0	2.89	5.35	25	4.89	—
8	10^{-3}	3.21	4.54	18	4.67	4.74
8	10^{-2}	3.58	4.94	7	4.16	4.63
8	10^{-1}	5.36	6.14	7	5.90	5.83

The table shows various quantities for a range of fits to the data of Figure 1(a). 'SSq' is the fitted sum of squares, and 'SSq$_{CV}$' is the 10-fold CV prediction of the fit to another dataset. (Clearly for 3 hidden units and $\lambda = 0$ the optimizer got stuck in a local minimum.) The table shows that NIC and SSq$_{CV}$ are usually similar, and that

p^* can be much less than p and is effectively controlled by λ. Figure 4 shows that there are only small differences between the models with small NIC (and small SSq_{CV}).



(a)



(b)

Figure 4: (a) The fitted curves for all the well-supported models. (b) Curves fitting these models to the true curve (with no added noise). Only the two hidden-unit model shows bias.

3 Bayesian model selection

In the Bayesian framework ([5]) it has been established for many years (at least since the 1948 edition of [8]) that the way to choose models is via the *Bayes factor*, (ratios of) the probability density of the data given the model. We have a prior density $p(\theta)$ over the parameters (weights) which must be integrable. Weight decay corresponds to an assumption of independent normal distributions for the weights (as is well-known in the context of ridge regression).

Finding the Bayes factor implies an integration over weight space which is normally impossible, but good approximations are known ([4, 5, 24]). In fitting we will maximize $E(\theta) = \log \ell(\theta; \mathrm{data}) + \log p(\theta)$. One approximation is

$$\log P(\mathcal{T} \mid \mathrm{model}) \approx E(\widehat{\theta}) + \frac{p}{2} \log 2\pi - \frac{1}{2} \log |H| \tag{2}$$

where H is the Hessian of E. This is easily computed from the fit.

For a regression problem, $\log \ell(\widehat{\theta}; \mathrm{data})$ is essentially the sum of squares divided by $2\sigma^2$, so (2) was converted into a corrected sum of squares, shown in the table. The agreement with the other methods is good, although no results are available without weight decay.

In this approach there is no difficulty with multiple local minima, which are combined by integration over the weight space (although (2) will need to be summed about each local minimum).

Markov chain Monte Carlo methods ([1]) have been used to explore the model space and so compute Bayes factors, for example in [26, 11].

4 Model averaging

Bayesians have long known that the logical endpoint of their theory is to average the predictions of their models, using the Bayes factors as weights, and this has become computationally feasible recently ([26, 10, 4]). In particular, rather than choosing a hyperparameter such as λ, we should integrate over a proper distribution. This is feasible with a coarse grid for λ. It also insists that we average over the predictions from local minima, computing the weights from (2).

Model averaging has been discussed in non-Bayesian contexts, for example Wolpert's [30] *stacked generalization*. It seems to have been missed that this was part of Stone's first paper on cross-validation ([27], pp. 126–7). His suggestion was to take a convex combination of the predictions of several models, using cross-validation to choose the weights of the combination. Averaging (often unweighted) has often been proposed for neural networks (e.g. [9, 19]).

There is little argument that model averaging is a good idea for predictions (and neural networks have almost no explanatory purpose). The main argument against is computational; for many pattern recognition problems it is difficult to get one network to predict fast enough, let alone many. But if it is computationally feasible, it has been a long established practice in applied statistics to average over a few models. The Bayesian view averages over all models, even those which do not fit well, and this has been applied ([26]), but Madigan & Raftery in [10] advocate an "Occam's window" to exclude all but models with good fits, and to exclude models which subsume others that are included. Thus only the simplest models which fit well are included in their averaging process.

References

[1] Besag, J., Green, P., Higdon, D. and Mengersen, K. (1995) Bayesian computation and stochastic systems. *Statistical Science* 1995.

[2] Bishop, C. Improving the generalization properties of radial basis function neural networks. *Neural Computation* 1991; 3:579–588.

[3] Cheng, B, and Titterington, D. M. Neural networks: a review from a statistical perspective (with discussion). *Statistical Science* 1994; 9:2–54.

[4] Draper, D. Assessment and propagation of model uncertainty (with discussion). *Journal of the Royal Statistical Society series B* 1995; 57:45–97.

[5] Gelman, A., Carlin, J. B., Stern, H. S. and Rubin, D. B. *Bayesian Data Analysis.* Chapman & Hall, New York, 1995.

[6] Geman, S., Bienenstock, E. and Doursat, R. Neural networks and the bias/variance dilemma. *Neural Computation* 1992; 4:1–58.

[7] Huber, P. J. The behavior of maximum likelihood estimates under nonstandard conditions. In: Le Cam, L. M. and Neyman, J. (eds) *Proceedings of the Fifth Berkeley*

Symposium on Mathematical Statistics and Probability. University of California Press, Berkeley, 1967, 1:221–233

[8] Jeffreys, H. *Theory of Probability.* Third edition. Clarendon Press, Oxford, 1961.

[9] Lincoln, W. P. and Skrzypek, J. Synergy of clustering multiple backpropagation networks. In: Touretzky, D. S. (ed) *Advances in Neural Information Processing Systems 2*. Morgan Kaufmann, San Mateo, CA, 1990, pp. 650–657.

[10] Madigan, D. and Raftery, A. E. Model selection and accounting for model uncertainty in graphical models using Occam's window. *Journal of the American Statistical Association* 1994; 89:1535–1546.

[11] Madigan, D. and York, J. Bayesian graphical models for discrete data. Technical report 239, Department of Statistics, University of Washington, 1993.

[12] Moody, J. E. Note on generalization, regularization and architecture selection in nonlinear learning systems. In *First IEEE-SP Workshop on Neural Networks in Signal Processing*. IEEE Computer Society Press, 1991, pp. 1–10.

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

[14] Moody, J. and Utans, J. Principled architecture selection for neural networks: Application to corporate bond rating prediction. In: Moody, J. E., Hanson, S. J. and Lippmann, R. P. (eds) *Advances in Neural Information Processing Systems 4*. Morgan Kaufmann, San Mateo, CA, 1992, pp. 683–690.

[15] Moody, J. and Utans, J. Architecture selection strategies for neural networks: Application to corporate bond rating prediction. In: Refenes, A.-P. (ed) *Neural Networks in the Capital Markets*. Wiley, Chichester, 1995, pp. 277–300.

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

[17] Murata, N., Yoshizawa, S. and Amari, S. Learning curves, model selection and complexity of neural networks. In: Hanson, S. J., Cowan, J. D. and Giles, C. L. (eds) *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, San Mateo, CA, 1993, pp. 607–614.

[18] Murata, N., Yoshizawa, S. and Amari, S. Network information criterion determining the number of hidden units for artificial neural network models. *IEEE Transactions on Neural Networks* 1994; 5:865–872.

[19] Perrone, M. P. and Cooper, L. N. When networks disagree: Ensemble methods for hybrid neural networks. In: Mammone, R. J. (ed) *Artificial Neural Networks for Speech and Vision*. Chapman & Hall, London, 1993, pp. 126–142.

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

[21] Ripley, B. D. Neural networks and related methods for classification (with discussion). *Journal of the Royal Statistical Society series B* 1994; 56:409–456

[22] Ripley, B. D. Neural networks and flexible regression and discrimination. In: Mardia, K. V. (ed) *Statistics and Images*. Carfax, Abingdon, 1994, pp. 39-57 (Advances in Applied Statistics 2).

[23] Ripley, B. D. Flexible non-linear approaches to classification. In: Cherkassky, V., Friedman, J. H. and Wechsler, H. (eds) *From Statistics to Neural Networks. Theory and Pattern Recognition Applications*. Springer, Berlin, 1994, pp. 105–126.

[24] Ripley, B. D. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, 1995.

[25] Seber, G. A. F. and Wild, C. J. *Nonlinear Regression.* Wiley, New York, 1989.

[26] Stewart, L. Hierarchical Bayesian analysis using Monte Carlo integration: computing posterior distributions when there are many possible models. *The Statistician* 1987; 36:211–219.

[27] Stone, M. Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society series B* 1974; 36:111–147.

[28] Stone, M. An asymptotic equivalence of choice of model by cross-validation and Akaike's criterion. *Journal of the Royal Statistical Society series B* 1977; 39:44–47.

[29] Wahba, G. and Wold, S. A completely automatic French curve. *Communications in Statistics* 1975; 4:1–17.

[30] Wolpert, D. H. Stacked generalization. *Neural Networks* 1992; 5:241–259.