A comparison of some error estimates for neural network models

Robert Tibshirani

Department of Preventive Medicine and Biostatistics

and

Department of Statistics

University of Toronto

October 6, 1995

Summary

We discuss a number of methods for estimating the standard error of predicted values from a multi-layer perceptron. These methods include the delta method based on the Hessian, bootstrap estimators, and the "sandwich" estimator. The methods are described and compared in a number of examples. We find that the bootstrap methods perform best, partly because they capture variability due to the choice of starting weights.

1. Introduction

We consider a multi-layer perceptron with one hidden layer and a linear output layer. See Lippman (1989), and Hinton (1989), and Hertz *et. al.* (1991) for details and references. A perceptron is a nonlinear model for predicting a response y based on p measurements of predictors (or input patterns or features) $x_1, x_2, \cdots x_p$. For convenience we assume that $x_1 \equiv 1$. The model with H hidden units has the form

$$
\begin{aligned}
y &= \phi_0(w_0 + \sum_{h=1}^{H} w_h z_h) + \epsilon \\
z_h &= \phi(\sum_{j=1}^{p} \beta_{jh} x_j)
\end{aligned} \tag{1}
$$

where the errors ϵ have mean zero, variance σ^2 and are independent across training cases. Since Y is a continuous response variable, we take the output function ϕ_0 to be the identity. The standard choice for the hidden layer output function ϕ is the sigmoid

$$
\phi(x) = \frac{1}{1 + \exp(-x)} \tag{2}
$$

Our training sample has n observations $(\mathbf{x}_1, y_1), \cdots (\mathbf{x}_n, y_n)$. Denote the ensemble of parameters (weights) by $\theta = (w_0, w_1, \ldots w_H, \beta_{11} \ldots \beta_{pH})$ and let $y(\mathbf{x}_i; \theta)$ be the predicted value for input $\mathbf{x}_i$ and parameter θ. The total number of parameters is $p \cdot H + 1 + H$.

Estimation of θ is usually carried out by minimization of $\sum [y_i - y(\mathbf{x}_i; \theta)]^2$, with either early stopping or some form of regularization to prevent overfitting. Commonly used optimization techniques include back-propagation (gradient descent), conjugate gradients, and quasi-Newton (variable metric) methods. Since the dimension of θ is usually quite large, search techniques requiring computation of the Hessian are usually impractical.

In this paper we focus on the problem of estimation of the standard error of the predicted values $y(\hat{\theta}; \mathbf{x}_i)$. A reference for these techniques is Efron and Tibshirani (1993), especially chapter 21. One approach is through likelihood theory. If we assume that the errors in model (1) are distributed as $N(0, \sigma^2)$, then the log-likelihood is

$$
\ell(\theta) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} [y_i - y(\mathbf{x}; \theta)]^2 - \frac{1}{2} \log \sigma^2 \tag{3}
$$

We eliminate σ^2 by replacing it by $\hat{\sigma}^2 = \sum_{i=1}^{n} [y_i - y(\mathbf{x}; \theta)]^2 / n$ in $\ell(\theta)$. The first and second derivatives have the form

$$
\begin{aligned}
\frac{\partial \ell}{\partial \theta_k} &= \frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} [y_i - y(\mathbf{x}_i; \theta)] \frac{\partial y(\mathbf{x}_i; \theta)}{\partial \theta_k} \\
\frac{\partial^2 \ell}{\partial \theta_k \partial \theta_l} &= -\frac{1}{\hat{\sigma}^2} \sum_{i=1}^{n} \Big[\frac{\partial y(\mathbf{x}_i; \theta)}{\partial \theta_k} \frac{\partial y(\mathbf{x}_i; \theta)}{\partial \theta_l} - (y_i - y(\mathbf{x}_i; \theta)) \frac{\partial^2 y(\mathbf{x}_i; \theta)}{\partial \theta_k \partial \theta_l} \Big]
\end{aligned} \tag{4}
$$

The exact form of these derivatives is simple to derive for a neural network, and we don't give them here. Because of the structure of the network, the only non-zero second derivative terms are those of the form $\partial^2 y/\partial\beta_{jh}\partial\beta_{kh}$ and $\partial^2 y/\partial w_h\partial\beta_{kh}$ and there are a total of $H\cdot p^2 + H\cdot p$ such terms. Buntine and Weigend (1994) describe efficient methods for computing the Hessian.

Let $\hat{I}$ equal $-\partial^2\ell/\partial\theta_k\partial\theta_l$ evaluated at $\theta=\hat{\theta}$, (the negative Hessian or "observed information" matrix), and $\hat{\mathbf{g}}_i = \partial y(\mathbf{x}_i;\theta)/\partial\theta$ evaluated at $\hat{\theta}$. Then using a Taylor series approximation we obtain

$$\widehat{\mathrm{se}}(y(\mathbf{x}_i;\hat{\theta})) \approx \left[\hat{\mathbf{g}}_i^T\cdot\hat{I}^{-1}\cdot\hat{\mathbf{g}}_i\right]^{1/2} \tag{5}$$

This is often called the *delta method* estimate of standard error (see Efron and Tibshirani 1993, chapter 21).

For computational simplicity, we can leave out the terms in (4) involving second derivatives. These are often small because the multipliers $y_i - y(\mathbf{x}_i;\theta)$ tend to be small. We will denote the resulting approximate information matrix by $\tilde{I}$.

With weight decay induced by a penalty term $\lambda\sum\theta_j^2$, it might be preferable to use the Hessian of the regularized log-likelihood $\ell(\theta) - \lambda\sum\theta_j^2$. This simply replaces $\hat{I}$ by $\hat{I}+2\lambda$ in formula (5), and will tend to reduce the delta method standard error estimates. This is the approach taken in MacKay (1992).

2. The Sandwich estimator

Like the information-based approach, the sandwich estimator has a closed form. Unlike the information however, its derivation does not rely on model correctness and hence it can potentially perform well under model-misspecification.

Let $\mathbf{s}_i = (s_{i1}, s_{i2}, \ldots)$ be the gradient vector of ℓ for the ith observation:

$$s_{ik} = \frac{1}{\hat{\sigma}^2}[y_i - y(\mathbf{x}_i;\theta)]\frac{\partial y(\mathbf{x}_i;\theta)}{\partial\theta_k} \tag{6}$$

Then the sandwich estimator of variance of $\hat{\theta}$ is defined by

$$\hat{V}_{\mathrm{sand}} = \hat{I}^{-1}\Big[\frac{1}{n}\sum_1^n \mathbf{s}_i\mathbf{s}_i^T\Big]\hat{I}^{-1} \tag{7}$$

To estimate the standard error of $y(\mathbf{x}_i;\hat{\theta})$, we substitute $\hat{V}_{\mathrm{sand}}$ for $\hat{I}^{-1}$ in equation (5):

$$\widehat{\mathrm{se}}_{\mathrm{sand}}(y(\mathbf{x}_i;\hat{\theta})) = \left[\mathbf{g}^T\cdot\hat{V}_{\mathrm{sand}}\cdot\mathbf{g}\right]^{1/2} \tag{8}$$

Note that $\frac{1}{n}\sum_1^n \mathbf{s}_i\mathbf{s}_i^T$ estimates $E\frac{\partial \ell}{\partial \theta}\frac{\partial \ell}{\partial \theta}^T$.

The idea behind the sandwich estimator is the following. If the model is specified correctly,

$$E\frac{\partial \ell}{\partial \theta}\frac{\partial \ell}{\partial \theta}^T = -E\frac{\partial^2 \ell}{\partial \theta \partial \theta^T} = I \tag{9}$$

Therefore $\hat{V}_{\text{sand}} \approx \hat{I}^{-1}E(I)\hat{I}^{-1} \approx \hat{I}^{-1}$ if the model is correct. Suppose however that the expected value of Y is modelled correctly but the errors have different variances. Then the sandwich estimator still provides a consistent estimate of variance, but (9) does not hold and hence the inverse information is not consistent. Details may be found in Kent (1982) and Efron and Tibshirani (1993, chapter 21).

3. Bootstrap methods

A different approach to error estimation is based on the bootstrap. It works by creating many pseudo-replicates ("bootstrap samples") of the training set and then re-estimating θ on each bootstrap sample.

There are two different ways of bootstrapping in regression settings. One can consider each training case as a sampling unit, and sample with replacement from the training set cases to create a bootstrap sample. This is often called the "bootstrap pairs" approach. On the other hand, one can consider the predictors as fixed, treat the model residuals $y_i - \hat{y}_i$ as the sampling units, and create a bootstrap sample by adding residuals to the model fit $\hat{y}_i$. This is called the "bootstrap residual" approach. The details are given below:

Bootstrap pairs sampling algorithm

1. Generate B samples, each one of size n drawn with replacement from the n training observations $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots (\mathbf{x}_n, y_n)\}$. Denote the bth sample by $\{(\mathbf{x}_1^{*b}, y_1^{*b}), (\mathbf{x}_2^{*b}, y_2^{*b}), \ldots (\mathbf{x}_n^{*b}, y_n^{*b})\}$.

2. For each bootstrap sample $b = 1, \ldots B$, minimize $\sum_{i=1}^{n} [y_i^{*b} - y(\mathbf{x}_i^*; \theta)]^2$ giving $\hat{\theta}^{*b}$.

3. Estimate the standard error of the ith predicted value by

$$\left\{ \frac{1}{B-1} \sum_{1}^{B} [y(\mathbf{x}_i; \hat{\theta}^{*b}) - y(\mathbf{x}_i; \cdot)]^2 \right\}^{1/2} \tag{10}$$

where $y(\mathbf{x}_i; \cdot) = \sum_{b=1}^{B} y(\mathbf{x}_i; \hat{\theta}^{*b})/B$.

Bootstrap residual sampling algorithm

1. Estimate $\hat{\theta}$ from the training sample and let $r_i = y_i - y(\mathbf{x}_i; \hat{\theta})$, $i = 1, 2, \ldots n$.

2. Generate B samples, each one of size n drawn with replacement from $r_1, r_2, \ldots r_n$. Denote the bth sample by $r_1^{*b}, r_2^{*b} \ldots r_n^{*b}$ and let $y_i^{*b} = y(\mathbf{x}_i; \hat{\theta}) + r_i^{*b}$.

3. For each bootstrap sample $b = 1, \ldots B$, minimize $\sum_{i=1}^{n} [y_i^{*b} - y(\mathbf{x}_i; \theta)]^2$ giving $\hat{\theta}^{*b}$.

4. Estimate the standard error of the ith predicted value by

$$\left\{ \frac{1}{B-1} \sum_{1}^{B} [y(\mathbf{x}_i; \hat{\theta}^{*b}) - y(\mathbf{x}_i; \cdot)]^2 \right\}^{1/2} \tag{11}$$

where $y(\mathbf{x}_i; \cdot) = \sum_{b=1}^{B} y(\mathbf{x}_i; \hat{\theta}^{*b})/B$.

Note that each method requires refitting of the model (retraining the network) B times. Typically B is in the range $20 \leq B \leq 200$.

In simple linear least squares regression, it can be shown that both the information-based estimate (5) and the bootstrap residual sampling estimate (as $B \to \infty$) both agree with the standard least squares formula $[\mathbf{x}_i^T (X^T X)^{-1} \mathbf{x}_i \hat{\sigma}^2]^{1/2}$, X denoting the design matrix having rows $\mathbf{x}_i$.

How do the two bootstrap approaches compare? The bootstrap residual procedure is model-

based, and relies on the fact that the errors $y_i - \hat{y}_i$ are representative of the true model errors. If the model is either misspecified or overfit, the bootstrap pairs approach is more robust. On the other hand, the bootstrap pairs approach results in a different set of predictor values in each bootstrap sample, and in some settings, this may be inappropriate. In some situations the set of predictor values is chosen by design, and we wish to condition on those values in our inference procedure. Such situations are fairly common in statistics (design of experiments) but probably less common in applications of neural networks.

4. Examples

In the following examples we compare a number of different estimates of the standard error of predicted values. The methods are:

1. Delta: the delta method (5)
2. Delta$_1$: the approximate delta method, using the approximate information matrix $\tilde{I}$ that ignores second derivatives.
3. Delta$_2$: the delta method, adding the term 2λ (from the regularization penalty) to the diagonal of the Hessian
4. Sand: the sandwich estimator (8)
5. Sand$_1$: the approximate sandwich estimator that uses $\tilde{I}$ in place of $\hat{I}$ in (7) and (8)
6. Bootp: bootstrapping pairs
7. Bootr: bootstrapping residuals

We used Brian Ripley's "nnet1" S-language function for the fitting, which uses the BFGS variable metric optimizer, with weight decay parameter set at 0.01. The optimizer is based on the Pascal code given in Nash (1979). Only $B = 20$ bootstrap replications were used. This is a lower limit on the number required in most bootstrap applications, but a perhaps a reasonable number when fitting a complicated model like a neural network. In the simulation studies (Examples 2-5), we carried out 25 simulations of each experiment.

Table 1: Results for example 1- air pollution data. Standard error estimates at five randomly chosen feature vectors.

	Point				
Method	1	2	3	4	5
Delta	0.15	0.13	0.24	0.38	0.20
Delta_1	0.13	0.12	0.16	0.12	0.17
Delta_2	0.13	0.12	0.17	0.24	0.17
Sand	0.14	0.12	0.32	0.29	0.25
Sand_1	0.10	0.11	0.21	0.12	0.20
Bootp	0.28	0.26	0.56	0.23	0.25
Bootr	0.19	0.24	0.24	0.23	0.24

4·1. Example 1: air pollution data

In this first example we illustrate the preceding techniques on 111 observations on air pollution, taken from Chambers and Hastie (1991). The goal is to predict ozone concentration from radiation, temperature and wind speed. We fit a multi-layer perceptron with one hidden layer of 3 hidden units, and a linear output layer. The various estimates of standard error, at five randomly chosen feature vectors, are shown in Table 1. Notice that the larger standard errors are given by the bootstrap methods in four of the five cases. As we will see in the simulations below, this is partly because the bootstrap captures the variability due the choice of random starting weights. In this example, repeated training of the neural network with different starting weights resulted in an average standard error of .07 for the predicted values.

One potential source of bias in the delta method estimate is our use of the maximum likelihood estimate for σ^2, namely $\hat{\sigma}^2 = \sum_{i=1}^{n}[y_i - y(\mathbf{x};\theta)]^2/n$. We could instead use an unbiassed estimate of the form $\hat{\sigma}^2 = \sum_{i=1}^{n}[y_i - y(\mathbf{x};\theta)]^2/(n-k)$, where k is an estimate of the number of effective parameters used by the network. However in this example, an upper bound for k is $4 \cdot 3 + 3 + 1 = 16$, and hence $\hat{\sigma}$ increases only by a factor of $(111/95)^{1/2} = 1.07$.

There is more information from the bootstrap process besides the estimated standard errors.

Figure 1 shows boxplots of the predicted values at each of the 5 feature vectors. Each boxplot contains values from 50 bootstrap simulations. Notice for example point 3 in the bottom plot. Its predicted values are skewed upward, and so we are less sure about the upper range of the prediction than the lower range.

4·2. Example 2: Fixed-X sampling

In this example we define x_1, x_2, x_3, x_4 to be multivariate Gaussian with mean zero, variance 1 and pairwise correlation 0.5. This predictor set was generated once and then fixed for all 25 of the simulations. We generated y as

$$y = 3\phi(2x_1 + 4x_2 + 3x_3 + 3x_4) + 3\phi(2x_1 + 4x_2 - 3x_3 - 3x_4) + \epsilon \tag{12}$$

where ϵ is Gaussian with mean zero and standard deviation 0.7. This gave a signal-to-noise ratio of roughly 1.2. There were 100 observations in each training set. Note that this function could be modelled exactly by a sigmoid net with two hidden nodes and a linear output node.

The results are shown in Table 2. In the left half of the table, a perceptron with one hidden layer of 2 hidden units, and a linear output was fit. In the right half, the perceptron had only one hidden unit in the hidden layer.

Let $\hat{s}_{ik}$ be the estimated standard deviation of $y(\mathbf{x}_i; \hat{\theta})$, for the kth simulated sample. Then we define $\text{se}_k \equiv \text{median}_i(\hat{s}_{ik})$, the median over the training cases of the estimated standard deviation of $\hat{y}_i$, for the kth simulated sample. Let s_i be the actual standard deviation of $y(\mathbf{x}_i; \hat{\theta})$. The actual value of the median standard deviation $\text{med}(s_i)$ is 0.86, as estimated over the 25 simulations. To measure the absolute error of the estimate over each of the training cases, we define $e_k = \text{median}_i|s_i - \hat{s}_{ik}|$.

In the left half of the table the two bootstrap methods are clearly superior to the other methods. The "Random weight se" of 0.39 is the standard error due solely to the choice of starting weights, estimated by fixing the data and retraining with different initial weights. This component of variance is missed by the first four methods. The right half of the table, all of the estimates have average values of e_k. Surprisingly, the bootstrap residual method is closest on the average to the actual se, closer than the bootstrap pairs approach. This may be because the bootstrap pairs method varies the X values and hence inflates the variance compared to the fixed-X sampling variance.

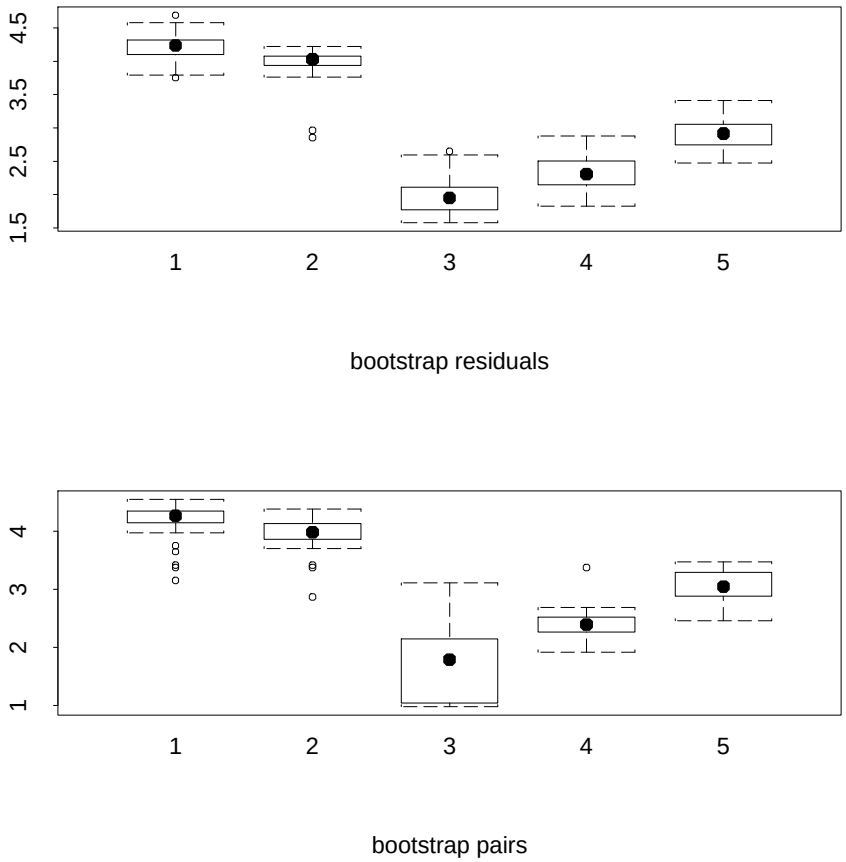

Figure 1: Boxplots of bootstrap replications for each of five randomly chosen feature vectors, from example 1. The bold dot in each box indicates the median, while the lower and upper edges are the 25% and 75% percentiles. The broken lines are the hinges, beyond which points are considered to be outliers.

Table 2: Results for example 2- Fixed-X sampling. See text for details.

	Correct model (2 hidden units)		Incorrect model (1 hidden unit)	
Method	Mean (sd) of se_k	Mean (sd) of e_k	Mean (sd) of se_k	Mean (sd) of e_k
Delta	0.39(.08)	0.39(.06)	0.41(.09)	0.15(.07)
$Delta_1$	0.35(.07)	0.42(.07)	0.43(.09)	0.15(.08)
$Delta_2$	0.36(.09)	0.41 (.05)	0.41(.09)	0.15(.08)
Sand	0.40(.06)	0.41(.04)	0.41(.08)	0.17(.05)
$Sand_1$	0.39(.09)	0.42(.08)	0.47(.14)	0.18(.06)
Bootp	0.93(.09)	0.17(.05)	0.72(.14)	0.22(.09)
Bootr	0.74(.07)	0.15(.04)	0.58(.06)	0.16(.04)
Actual se	0.86(—)	—	0.56(—)	—
Random weight se	0.39(—)	—	0.38(—)	—

4·3. Example 3: Random-X sampling

The setup in this example is the same as in the last one, except that a new set of predictor values was generated for each simulation. The predictions were done at a fixed set of predictor values, however, to allow pooling across simulations. The bootstrap methods perform the best again: surprisingly, the bootstrap pairs method only does best in the "incorrect model" case. This is surprising because its resampling of the predictors matches the actual simulation sampling used in the example.

4·4. Example 4: Overfitting

In this example, the setup is the same as in the left hand side of Table 3, except that the neural net was trained with 7 hidden units and no weight decay. Thus the model has 5 more units than is necessary, and with no weight decay, should overfit the training data. The results of the simulation experiment are shown in Table 4. We had difficulty in computing the inverse information matrix due to near singularities in the models, and hence report only the bootstrap results. As expected, the bootstrap residual method underestimates the true standard error because the overfitting has

Table 3: Results for example 3: Random-X sampling. See text for details.

	Correct model (2 hidden units)		Incorrect model (1 hidden unit)	
Method	Mean (sd) of se_k	Mean (sd) of e_k	Mean (sd) of se_k	Mean (sd) of e_k
Delta	0.38(.08)	0.45(.05)	0.38(.07)	0.36(.09)
Delta_1	0.35(.11)	0.48(.09)	0.34(.14)	0.38(.15)
Delta_2	0.34(.08)	0.47(.09)	0.33(.14)	0.38(.15)
Sand	0.39(.08)	0.47(.06)	0.39(.08)	0.34(.07)
Sand_1	0.46(.17)	0.45(.08)	0.49(.24)	0.40(.13)
Bootp	1.05(.11)	0.26(.06)	0.73(.12)	0.21(.06)
Bootr	0.81(.09)	0.22(.05)	0.53(.14)	0.24(.04)
Actual se	0.87(—)	—	0.76(—)	—
Random weight se	0.39(—)	—	0.38(—)	—

Table 4: Results for example 4- overfitting. See text for details.

Method	Mean (sd) of se_k	Mean (sd) of e_k
Bootp	2.20(.13)	0.61(.09)
Bootr	1.23(.06)	0.62(.07)
Actual se	1.84(—)	—
Random weight se	0.52(—)	—

Table 5: Results for example 5- averaging over runs. See text for details.

Method	Mean (sd) of se_k	Mean (sd) of e_k
Delta	0.37(.10)	0.24(.07)
$Delta_1$	0.30(.10)	0.30(.07)
$Delta_2$	0.35(.09)	0.25(.06)
Sand	0.38(.06)	0.38(.05)
$Sand_1$	0.52(.29)	0.52(.08)
Bootp	0.68(.06)	0.16(.03)
Bootr	0.57(.02)	0.13(.01)
Actual se	0.61(—)	—
Random weight se	0.25(—)	—

biased the residuals towards zero. In the extreme case, if we were to completely saturate the model, the residuals would all be zero and the resulting standard error estimate would also be zero. The bootstrap pairs method seems to capture the variation better, but suffers from excess variability across simulations.

4·5. Example 5: Averaging over runs

The setup here is the same as in the left hand side of Table 2, except that the training is done by averaging the predicted values over three runs with different random starting weights. The results are shown in Table 5. The bootstrap methods still perform the best, but by a lesser amount than before. The reason is that the variation due to the choice of random starting weights has been reduced by the averaging. Presumably, if we were to average over a larger number of runs, this variation would be further reduced.

5. Discussion

In the simulation experiments of this paper, we found that:

- The bootstrap methods provided the most accurate estimates of the standard errors of predicted values
- The non-simulation methods (delta method, sandwich estimator) missed the substantial variability due to the random choice of starting values

Of course the results found here may not generalize to all other applications of neural networks. For example, the non-simulation approaches may work better with fitting methods that are less sensitive to the choice of starting weights. Larger training sets, and the use of gradient descent methods will probably lead to fewer local minima and hence less dependence on the random starting weights than seen here. In addition, in very large problems the bootstrap approaches may require too much computing time to be useful. Note that the bootstrap methods illustrated here don't suffer from matrix inversion problems in overfit networks, and don't require the existence of derivatives.

It is important to note that an interval formed by taking say plus and minus 1.96 times a standard error estimate from this paper, would be an approximate confidence interval for the mean of a predicted value. This differs from a *prediction interval*, which is an interval for a future realization of the process. A prediction interval is typically wider than a confidence interval, because it must account for the variance of the future realization. Such an interval can be produced by increasing the width of the confidence interval by an appropriate function of the noise variance $\hat{\sigma}^2$.

We have considered only regression problems here, but the methods generalize easily to classification problems. With k classes, one usually specifies k output units, each with a sigmoidal output function ϕ_0 and minimize either squared error or the multinomial log-likelihood (cross-entropy). The only non-trivial change occurs for the bootstrap residual method. There are no natural residuals for classification problems, and instead we proceed as follows. Suppose for simplicity that we have two classes 0, and 1, and let $\hat{p}(\mathbf{x}_i)$ be the estimated probability that y equals one for feature vector $\mathbf{x}_i$. We fix each $\mathbf{x}_i$ and generate Bernoulli random variables y_i^{*b} according to $\text{Prob}(y_i^{*b} = 1) = \hat{p}(\mathbf{x}_i)$, for $i = 1, \ldots n$ and $b = 1, \ldots B$. Then we proceed as in steps 3 and 4 of the bootstrap residual sampling algorithm, using either squared error or cross-entropy in step 3. An application of this procedure is described in Baxt and White (1994).

A Bayesian approach to error estimation in neural networks may be found in Buntine and Weigend (1991), and MacKay (1992). Nix and Weigend (1994) propose a method for estimating the variance of the target, allowing it to vary as a function of the input features. LeBaron and Weigend (Snowbird 1994) propose a method similar to the bootstrap pairs approach, that uses a test set to generate the predicted values. Leonard, Kramer and Ungar (1992) describe an alternative approach to confidence interval estimation that can be applied to radial basis networks.

6. Acknowledgements

The author thanks Richard Lippmann, Andreas Weigend and two referees for their valuable comments, and acknowledges the Natural Sciences and Engineering Research Council of Canada for its support.

References

Baxt, W. & White, H. (1994), Bootstrapping confidence intervals for clinical input variable effects in a network trained to identify the presence of acute myocardial infarction, Technical report, Univ. of Cal., San Diego.

Buntine, W. & Weigend, A. (1994), 'Computing second derivatives in feed forward neural networks: a review (to appear)', *IEEE trans. Neur. Networks*.

Chambers, J. & Hastie, T. (1991), *Statistical Models in S*, Wadsworth/Brooks Cole, Pacific Grove.

Efron, B. & Tibshirani, R. (1993), *An Introduction to the Bootstrap*, Chapman and Hall.

Hertz, J., Krogh, A. & Palmer, R. (1991), *Introduction to the theory of neural computation*, Addison Wesley, Redwood City.

Hinton, G. (1989), 'Connectionist learning procedures', *Artificial intelligence* **40**, 185–234.

Kent, T. (1982), 'Robust properties of likelihood ratio tests', *Biometrika* **69**, 19–27.

LeBaron, A. & Weigend, A. (1994), Evaluating neural network predictors by bootstrapping, *in* 'Proceedings of the International Conference on Neural Information Processing (ICONIP'94)', Seoul, Korea.

Leonard, J., Kramer, M. & Ungar, L. (1992), 'A neural network architecture that computes its own reliability', *Computers and Chemical Engineering* **16**, 819–835.

Lippman, R. (1989), 'Pattern classification using neural networks', *IEEE communications magazine* **11**, 47–64.

MacKay, D. (1992), 'A practical bayesian framework for backpropagation neural networks', *Neural Computation* **4**, 448–472.

Nash, J. (1979), *Compact Numerical Methods for Computers*, Halsted.

Nix, D. & Weigend, A. (1994), Estimating the mean and variance of a target probability distribution, *in* 'Proceedings of the IJCNN', Orlando.