Confidence and prediction intervals for neural network ensembles

John G. Carney and Pádraig Cunningham
Department of Computer Science
University of Dublin
Trinity College
Ireland
John.Carney@cs.tcd.ie

Umesh Bhagwan
Department of Economics
London School of Economics and Political Science
Houghton Street
London, England
U.Bhagwan@lse.ac.uk

Abstract

In this paper we propose a new technique that uses the bootstrap to estimate confidence and prediction intervals for neural network (regression) ensembles. Our proposed technique can be applied to any ensemble technique that uses the bootstrap to generate the training sets for the ensemble, such as bagging [1] and balancing [5]. Confidence and prediction intervals are estimated that include a significantly improved estimate of underlying model uncertainty (i.e.) the uncertainty of our estimate of the "true" regression. Unlike existing techniques, this estimate of uncertainty will vary according to which ensemble technique is used – if the effect of using a specific ensemble technique is to produce less model uncertainty than using another ensemble technique, then this will be reflected in the confidence and prediction intervals. Preliminary results illustrate how our technique can provide more accurate confidence and prediction intervals (intervals that better reflect the desired level of confidence (e.g.) 90%, 95%, etc.) for neural network ensembles than previous attempts.

1 Introduction

Assume we are given a set of N data pairs $\{(t_n, \mathbf{x}_n)\}_{n=1}^N$, generated according to

$$t(\mathbf{x}) = f(\mathbf{x}) + \epsilon(\mathbf{x}) \tag{1}$$

where $t(\mathbf{x})$ is the observed target value, $f(\mathbf{x})$ is the true regression and $\epsilon(\mathbf{x})$ is noise with zero mean. When we train a neural network on such data, our aim is for the network to approximate $f(\mathbf{x})$. Let us denote this neural network approximation as $\phi(\mathbf{x})$, which can be interpreted as an estimate of the mean of the distribution of target values given an input vector $\mathbf{x}$. For many real-world regression applications, it is highly desirable to have some measure of confidence in this point prediction. As described in [6], there are two "components" of confidence however. The first is concerned with the accuracy of our estimate of the true regression (i.e.) the distribution of the quantity $f(\mathbf{x}) - \phi(\mathbf{x})$. In a regression context, these estimates are usually termed *confidence intervals* (see [6] for example). The second is concerned with estimates of confidence in our prediction of the targets themselves (i.e.) the distribution of the quantity $t(\mathbf{x}) - \phi(\mathbf{x})$. These estimates are usually termed *prediction intervals*. As illustrated below in equation (2), a confidence interval is enclosed in a prediction interval:

$$t(\mathbf{x}) - \phi(\mathbf{x}) = [f(\mathbf{x}) - \phi(\mathbf{x})] + \epsilon(\mathbf{x}) \tag{2}$$

Prediction intervals are of more practical use than confidence intervals for real-world applications. This is because prediction intervals are concerned with the accuracy with which we can predict the targets or observed values themselves, not just the accuracy of our prediction of the true regression.

2 Neural network ensembles

Recently, there has been a lot of interest in neural network ensemble techniques. Here, estimates from a number of neural networks are combined to provide generalization performance superior to that provided by a single network. Ensemble techniques have been studied in many forms. Some of most popular varieties use bootstrapping (see [3] for a good introduction) to generate the training sets for the ensemble, for example, bagging [1] and balancing [5]. Both these techniques attempt to "stabilize" high-variance predictors such as neural networks by generating multiple bootstrap versions of the predictor and then combining the outputs of these individual versions to form "smoother" predictions. They differ in the way the predictions are combined however.

A more concrete description of these techniques goes as follows. Let us assume we have a neural network constructed using a training set T that consists of N training vectors. During training, parameters (the weights) are estimated and the predictor generates an output or prediction when presented with input $\mathbf{x}$:

$$\hat{t}(\mathbf{x}) = \phi(\mathbf{x}; T) \tag{3}$$

Let us generate B bootstrap re-sampled training sets $\{T_b^*\}_{b=1}^B$. Each bootstrap re-sampled training set consists of N input-output training vectors, sampled at random from T with replacement. Using these training sets we now have a set of networks $\{\phi(\mathbf{x}; T_b^*)\}_{b=1}^B$. For bagging, predictions from these bootstrap versions are averaged to form a lower variance, bagged prediction:

$$\hat{t}_{bag}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} \phi(\mathbf{x}; T_b^*) \tag{4}$$

It can be shown (see [1]) that bagging "unstable" or high-variance predictors such as neural networks in this way can substantially improve their generalization performance.

In [5] bagging is extended to balancing. Here, quadratic programming techniques are used to estimate weights for a weighted average of networks. The motivation here is that some networks will perform better than others and should therefore have greater influence on the final combined prediction.

There are many other ensemble techniques for neural networks that do not use the bootstrap to generate different predictors. An overview can be found in [8]. In this paper, we focus only on bootstrap ensemble techniques.

3 Predicting uncertainty

3.1 Confidence intervals

As mentioned in section 1 above, confidence intervals are enclosed in prediction intervals and are concerned with the accuracy of our estimate of the true regression (i.e.) the distribution of the quantity $f(\mathbf{x}) - \phi(\mathbf{x})$ or $P(f(\mathbf{x})|\phi(\mathbf{x}))$. When we use an ensemble technique, we are interested in the distribution of the quantity $f(\mathbf{x}) - \phi_{com}(\mathbf{x})$ or $P(f(\mathbf{x})|\phi_{com}(\mathbf{x}))$, where $\phi_{com}(\mathbf{x})$ denotes some combination of the network outputs in the ensemble. In this section we show how confidence intervals can be computed for the estimate $\phi_{com}(\mathbf{x})$ if we assume a normal distribution for $P(f(\mathbf{x})|\phi_{com}(\mathbf{x}))$.

To generate the confidence (and prediction) intervals we also need to assume our neural networks provide an unbiased estimate of the true regression $f(\mathbf{x})$. In other words (for confidence intervals) we need to assume the distribution $P(f(\mathbf{x})|\phi_{com}(x))$ is centered on the estimate $\phi_{com}(\mathbf{x})$. This assumption, of course, does not hold in practice – as with any other estimator, neural networks can and usually are biased (i.e.) residual errors are not caused by variance alone. However, it is generally accepted that the variance component of residual error in neural network learning dominates the bias component (see [9] for more on this).

To form our confidence intervals, we need to estimate the variance of the distribution $P(f(\mathbf{x})|\phi_{com}(\mathbf{x})$. However, we have no direct access to this distribution and we do not know what the true regression $f(\mathbf{x})$ is. Using the outputs of the bootstrap re-sampled networks in the ensemble, we can however approximate it.

The bootstrap outputs provide us with an empirical estimate of the distribution $P(\phi_{com}(\mathbf{x})|f(\mathbf{x}))$ which is the "inverse" of the distribution $P(f(\mathbf{x})|\phi_{com}(\mathbf{x}))$. This empirical estimate of $P(\phi_{com}(\mathbf{x})|f(\mathbf{x}))$ is $P(\phi(\mathbf{x})|\phi_{com}(\mathbf{x}))$. Here, $\phi_{com}(\mathbf{x})$ replaces the true regression to which we have no access. The variance of this distribution can be found by calculating the variance across the bootstrap outputs. Note that by assuming $P(f(\mathbf{x})|\phi_{com}(\mathbf{x}))$ is Gaussian we also assume its inverse $P(\phi_{com}(\mathbf{x})|f(\mathbf{x}))$ is Gaussian and so any estimates of variance for $P(\phi_{com}(\mathbf{x})|f(\mathbf{x}))$ can be used as estimates of variance for $P(f(\mathbf{x})|\phi_{com}(\mathbf{x}))$. This gives us

$$\sigma^2(\mathbf{x}) = \frac{1}{B-1}\sum_{b=1}^{B}(\phi_b(\mathbf{x}) - \phi_{avg}(\mathbf{x}))^2 \tag{5}$$

where

$$\phi_{avg}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B}\phi_b(\mathbf{x}) \tag{6}$$

In [6], this variance measure is used to construct standard Gaussian confidence intervals for bagged ensemble predictions. However, this variance estimate will be biased upwards for most predictions. This is because it more accurately reflects the variance of the distribution $P(f(\mathbf{x})|\phi(\mathbf{x}))$, not $P(f(\mathbf{x})|\phi_{com}(\mathbf{x}))$. In other words, it really only provides a variance measure suitable for computing a confidence interval for a single network prediction $\phi(\mathbf{x})$. Most ensemble techniques, especially those studied here, have the effect of reducing the variance of a predictor. This reduced variance should be reflected in the confidence interval. We will now propose a way to do this.

Using a large number of bootstrap networks for the ensemble (we use 200, but fewer would suffice) we divide the ensemble into M smaller ensembles (we use 8 groups of 25 networks each). This gives us a set of M $\phi_{com}(\mathbf{x})$ values

$$\zeta = \{\phi^i_{com}(\mathbf{x})\}_{i=1}^{M} \tag{7}$$

from which we approximate a more accurate variance measure for the distribution $P(\phi_{com}(\mathbf{x})|f(\mathbf{x}))$. However, we don't simply compute a variance measure across (in our case) the 8 ensemble outputs - this variance measure itself would be highly variable and unreliable. Instead we undergo one more iteration of bootstrapping. We form P (we use $P = 1000$) bootstrap re-sampled sets of ζ

$$\Upsilon = \{\zeta^*_j\}_{j=1}^{P} \tag{8}$$

where

$$\zeta^*_j = \{\phi^{j_1}_{com}{}^*(\mathbf{x}), ..., \phi^{j_M}_{com}{}^*(\mathbf{x})\} \tag{9}$$

each containing M $\phi_{com}(\mathbf{x})$ values sampled with replacement from ζ. We calculate a variance measure for each of these sets and then calculate an average of these to provide a smoother, lower variance estimate of the variance of the distribution $P(\phi_{com}(\mathbf{x})|f(\mathbf{x}))$:

$$\sigma^2_S(\mathbf{x}) = \frac{1}{P}\sum_{j=1}^{P}\sigma^{2*}_j(\mathbf{x}) \tag{10}$$

where

$$\sigma^{2*}_j(\mathbf{x}) = \frac{1}{M}\sum_{k=1}^{M}(\phi^{j_k}_{com}{}^*(\mathbf{x}) - \phi^j_{avg}(\mathbf{x}))^2 \tag{11}$$

Note that here, to estimate the true regression, we use the combination of values across all the B networks in the ensemble and denote this as $\phi_C(\mathbf{x})$. In other words, we approximate the distribution $P(\phi_{com}(\mathbf{x})|f(\mathbf{x}))$ from $P(\phi_{com}(\mathbf{x})|\phi_C(\mathbf{x}))$. This second iteration of bootstrapping is not computationally intensive (there are no networks to train) and is easy to implement.

Now that we have a good variance estimate for the distribution $P(f(\mathbf{x})|\phi_{com}(\mathbf{x}))$, which we assume to be Gaussian, we can calculate a confidence interval in the usual fashion:

$$\phi_C(\mathbf{x}) - z^{(1-\alpha)}\sigma_S(\mathbf{x}) \le f(\mathbf{x}) \le \phi_C(\mathbf{x}) + z^{(1-\alpha)}\sigma_S(\mathbf{x}) \tag{12}$$

The factor depends on the desired level of confidence ((e.g.) 90%, 95% etc.) and can be taken from a standard Gaussian distribution table.

3.2 Prediction intervals

In this section we describe how the improved confidence intervals above should be extended to provide improved prediction intervals for neural network regressions. Prediction intervals deal with the distribution of the quantity $t(\mathbf{x}) - \phi_{com}(\mathbf{x})$ or

$P(t(\mathbf{x})|\phi_{com}(\mathbf{x}))$ (i.e.) they estimate the accuracy with whch we can predict the targets themselves.

To estimate prediction intervals, we must incorporate the noise variance factor of a regression. This is the variance of the noise component $\epsilon(\mathbf{x})$ given in equations (1) and (2). As in [6] we will denote this noise variance as $\chi^2(\mathbf{x})$. Incorporating this factor, the variance of the complete regression can be given as

$$s^2(\mathbf{x}) = \left\langle (t(\mathbf{x}) - \phi_C(\mathbf{x}))^2 \right\rangle = \left\langle (f(\mathbf{x}) - \phi_C(\mathbf{x}))^2 \right\rangle + \left\langle \epsilon^2(\mathbf{x}) \right\rangle = \sigma_S^2(\mathbf{x}) + \chi^2(\mathbf{x}) \quad (13)$$

where $\langle . \rangle$ denotes expectation. Our first task then is to develop a way to build models that provide an estimate for $\chi^2(\mathbf{x})$ when presented with an input vector $\mathbf{x}$. Once we have done this we can calculate an estimate for $s^2(\mathbf{x})$ and build a prediction interval.

To estimate $\chi^2(\mathbf{x})$, we use a technique first put forward by [7]. Here, a parametric (Gaussian) noise variance model is assumed (i.e.) $\epsilon(\mathbf{x}) \sim N(0, \chi^2(\mathbf{x}))$. A model is then found (we use a separate, individual neural network) for $\chi^2(\mathbf{x})$ to fit the residuals

$$R^2 = \{((t(\mathbf{x}_n) - \phi_C(\mathbf{x}_n))^2 - \sigma_S^2(\mathbf{x}_n))\}_{n=1}^N \quad (14)$$

using minus the logliklihood as the error measure:

$$C = -\sum_{n=1}^{N} log \left(\frac{1}{\sqrt{2\pi\chi^2(\mathbf{x}_n)}} exp\left(-\frac{r^2(\mathbf{x}_n)}{2\chi^2(\mathbf{x}_n)} \right) \right) \quad (15)$$

In other words, we train a neural network using equation (15) as the cost function and a training set constructed with input components the same as in our original training set $\{(t_n, \mathbf{x})\}_{n=1}^N$ and targets estimated from the set of residuals in equation (14). Since estimates of variance should always be positive, an exponential activation function for the output unit of this network is used to impose this bound.

To avoid over-fitting and underestimating the noise variance, validation or "out-of-bag" residuals should be used as the outputs in this training set - see [6] for more on this. This technique also assumes that the residual errors are caused by variance alone (i.e.) the neural network ensemble is not biased.

Now that we have a way to estimate the noise deviation $\chi(\mathbf{x})$, we can compute a prediction interval by combining it with our estimate of model deviation $\sigma_S(\mathbf{x})$:

$$\phi_C(\mathbf{x}) - z^{(1-\alpha)} s(\mathbf{x}) \leq f(\mathbf{x}) \leq \phi_C(\mathbf{x}) + z^{(1+\alpha)} s(\mathbf{x}) \quad (16)$$

Again, the factor $z^{(1-\alpha)}$ depends on the desired level of confidence and can be taken from a standard Gaussian distribution table.

4 Experiments

In this section we describe experiments we performed to evaluate our technique using a simple sythetic data-set. Inputs in this data-set are uniformly drawn at random from the interval $[0, 2]$. Here, the input vector contains only a single value. The target values are generated according to

$$t(\mathbf{x}) = 4.26(e^{-\mathbf{x}} - 4e^{-2\mathbf{x}} + 3e^{-3\mathbf{x}}) + \epsilon(\mathbf{x}) \quad (17)$$

where $\epsilon(\mathbf{x}) \sim N(0, \chi^2(\mathbf{x}))$ and

$$\left\langle \epsilon^2(\mathbf{x}) \right\rangle = -0.1(e^{-18\mathbf{x}} - e^{-4\mathbf{x}}) \quad (18)$$

We generated 1000 data pairs for training and 1000 for testing. The test set is illustrated in figure 1.

For our experiments, 200 networks were trained for the (bagged) ensemble. We ran the test set through our ensemble following the steps outlined in sections 3.1 and 3.2 to produce 90% confidence and prediction intervals for each test point.

The width of the confidence intervals for our technique are very close to the ideal width for a 90% confidence interval, giving non-coverage across the 1000 test points of 112, which compares very well to the ideal, expected non-coverage of 100. We repeated the experiment using the simpler technique described in [6]. Here the confidence intervals are consistently too wide for an ensemble prediction and give no non-coverage.

The resulting prediction intervals are also illustrated in figure 1. The estimate for noise variance is not perfect and so the results for the prediction intervals are degraded somewhat. Here, our technique provides total non-coverage of 143 and the simpler technique described in [6] of 6. Again, our technique provides superior performance.

5 Conclusion

In this paper we proposed a new technique for estimating confidence and prediction intervals for neural network bootstrap ensembles. The intervals incorporate a significantly improved estimate of the underlying model uncertainty (i.e.) the uncertainty of our estimate of the true regression. Unlike existing techniques, this estimate of uncertainty will vary according to which ensemble technique is used. The only drawback of the technique when compared to previous similar techniques such as that described in [6] is that more computational power is required - we need to train more networks for the ensemble for the intervals to be meaningful. However, we proposed another bootstrap technique that minimizes this extra computational requirement.

The preliminary results described in section 4 above are promising. However, we did use a very simple synthetic regression problem to evaluate our technique. Our motivation here was to maintain some statistical transparency and flexibility (e.g.) we could control the magnitude of the noise variance at each data point, separate the confidence intervals from the prediction intervals and estimate how biased our ensembles were by comparing their prediction to the true regression. The next step is to test our technique using real-world noisy training and test data. Here we would not expect our technique to perform as well as for the synthetic data - for most real-world tasks (e.g.) financial time-series data, the noise is likely to be only approximately Gaussian and the networks more biased given the difficult learning task that this is.

Also, it would be interesting to compare the performance of our parametric prediction interval technique to other techniques used to estimate prediction uncertainity such as the non-parametric density estimation technique described in [4]. This (and other similar techniques) can be used to predict the entire conditional probability distribution of a prediction without assuming any parameterized form for the noise variance. However, model uncertainty is rarely accounted for in any rigorous fashion. Also, very large quantities of data are required, something that is very difficult to fulfill in real-world domains.

References

[1] L. Breiman. Bagging predictors. *Machine Learning*, 24:123-140, 1996.

[2] J.G. Carney and P. Cunningham. The NeuralBAG algorithm: Optimizing generalization performance in bagged neural networks. In *Proceedings of the 7th European Symposium on Artificial Neural Networks*, Bruges, Belgium, 1999. D-Facto.

[3] B. Efron and R. Tibshirani. *An Introduction to the Bootstrap.* Chapman Hall, London, 1993.

[4] A. M. Fraser and A. Dimitriadis. Forecasting probability densities using hidden markov models with mixed states. In *Time Series Prediction: Forecasting the Future and Understanding the Past*, A. Weigend and N. Gershenfeld, eds., Addison-Wesley, pp. 175-193.

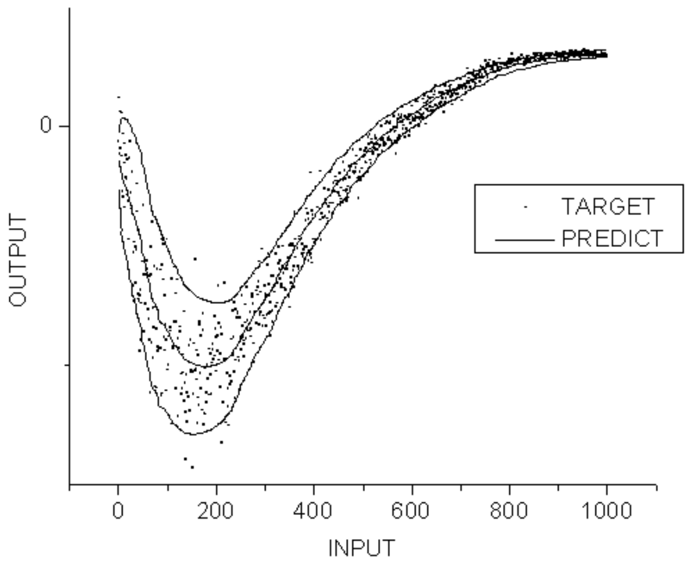
[5] T. Heskes. Balancing between bagging and bumping. In M. Mozer, M. Jordan and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 466-472, Cambridge, 1997. MIT Press.

[6] T. Heskes. Practical confidence and prediction intervals. In M. Mozer, M. Jordan and T. Peskes, editors, *Advances in Neural Information Processing Systems 9*, pages 466-472, Cambridge, 1997, MIT Press.

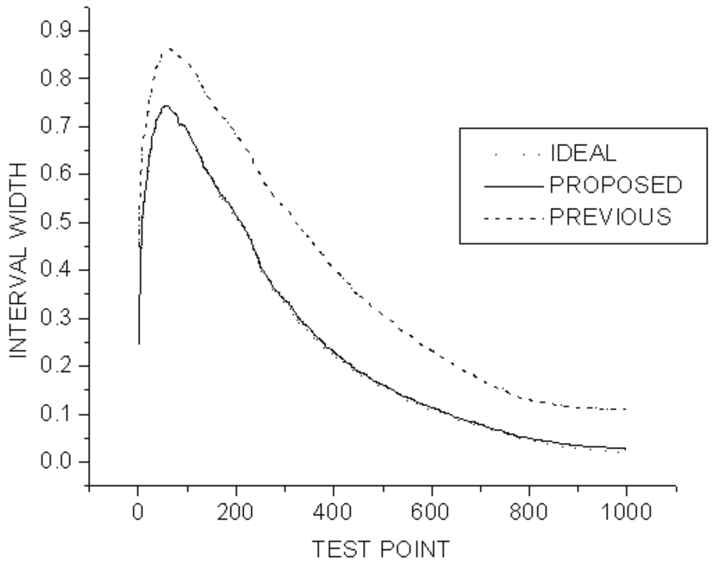
[7] D. Nix and A. Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of the IJCNN '94*, pages 55-60. IEEE.

[8] A.J C. Sharkey. *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems.* Springer-Verlag, 1999.

[9] G. Stuart, E. Bienenstock and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4,1:1-58, 1992.



(a)



(b)

Figure 1: (a) 1000 test point targets, neural network prediction of the true regression and bounds for a 90% prediction interval estimated using our proposed technique. (b) Width of prediction intervals across the 1000 test points for our technique, the simpler technique described in [6] and the ideal, target width for a 90% prediction interval.