Variance Estimation using Subbagging to Quantify Uncertainty in Neural Network Ensembles

Tyler Forrester
Gianforte School of Computing
Montana State University
Bozeman, MT, USA
tylerforrester@msu.montana.edu

Jordan Schupbach
Department of Mathematical Sciences
Montana State University
Bozeman, MT, USA
jordan.schupbach@montana.edu

John Sheppard
Gianforte School of Computing
Montana State University
Bozeman, MT, USA
john.sheppard@montana.edu

***Abstract*—Quantifying uncertainty is critically important to many applications of predictive modeling. In this paper we propose a novel method of estimating uncertainty in neural networks using a U-statistic subbagging method proposed by Mentch and Hooker [1]. This method trains neural networks on subsamples of data and uses the resulting ensemble to estimate variance of point estimates. We demonstrate that an ensemble of neural networks predicting a regression function has the required theoretical properties, and we then perform a coverage probability study of two simulated data sets to show empirically that the coverage probabilities match the theoretical predictions of the subbagging method.**

I. Introduction

One major limitation of neural networks is the reliable and computationally efficient quantification of model uncertainty. This limitation arises from the fact the trained model does not capture probability estimates directly. By considering ensembles of models, however, we can use the behaviors of these ensembles as a basis for estimating confidence in predictions. This paper extends the U-Statistic framework for building confidence intervals in estimator ensembles to neural networks using methods developed by Mentch and Hooker, [1]. We perform a coverage probability analysis of these methods for predictions in linear and nonlinear regression settings.

A. Background

Before presenting our experiments, we begin by introducing some preliminary material. We first give background knowledge on neural networks. We then introduce U-statistics and the related infinite order U-statistics that can be used to estimate model uncertainty. We show that neural networks can be used as the estimator in an U-statistic framework. Finally, we describe others' efforts in quantification of uncertainty in neural networks and neural network ensembles.

1) Feedforward Neural Networks: This paper uses feedforward neural networks [2] as estimators in simple regression. Our neural networks are divided into three layers, an input layer, a hidden layer, and an output layer, (Figure 1). The activation function in the nodes is represented by

$$\sigma(\mathbf{w}^\top \mathbf{x} + b)$$

where σ is the sigmoid function, $\mathbf{w}$ is a vector of weights, $\mathbf{x}$ is vector of inputs, and b is the bias or threshold. In the

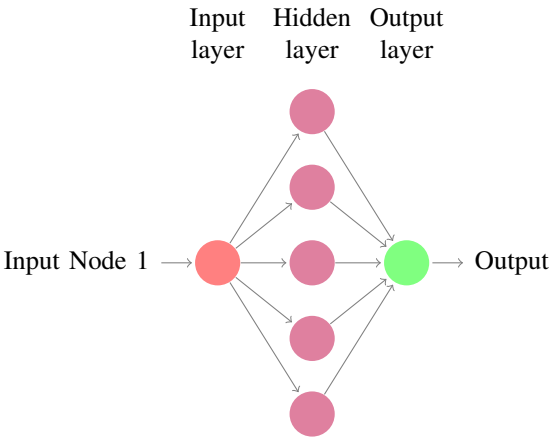

Fig. 1: A feedforward neural network for simple linear regression

training of the neural networks, the loss function is squared error [3] and for the gradient descent algorithm, we use Adam, a stochastic optimization technique that performs step size annealing through the use of the second moment to calculate the gradient [4]. This method has been found to work well empirically [5].

2) Neural Network Ensembles: Neural network ensembles[1] have a long history of being used to improve predictive power. It is standard to train many networks with different hyperparameter combinations, while only using the network that performs the best on the validation set. Researchers have looked for ways to use the discarded neural networks to improve performance, thus creating such an ensemble [6].

Several good surveys on ensembles have been published, most recently by Li [7] with earlier work done by Masoudnia and Ebrahmipour (mixture of experts) [8], Wozniak *et al.* [9] (general), and Mendes-Moreira *et al.* [10] (regression). An introductory textbook into ensemble methods has been written by Rokach [11].

There are many earlier references to ensemble learning in the machine learning literature, such as [12], [13]. Salamon and Hansen were the first to propose using neural network

[1]sometimes called committees

ensembles [14], where their paper proved with some restrictions[2] that if each network performed slightly better than random chance on a prediction, a large enough ensemble would guarantee a correct prediction, giving a solid theoretical foundation for ensemble use. This work has been extended in regression to show that the mean of a neural network ensemble is an estimator that is at least as good as any member's estimate [15], [16].

Using an active learning algorithm that measures the range of outputs from the ensemble members on unlabeled data, Krogh and Vedelspy [17] showed that it is also possible to decompose ensemble error rate into terms representing the differences or disagreement between ensemble members and the individual members' error rates. The optimal ensemble maximizes disagreement while minimizing individual member error.

The two most common approaches to training ensembles (in general) are bagging [18] and boosting [19], both of which are designed to maximize disagreement while minimizing individual member error. Bagging is the process of training members of a neural network ensemble on bootstrapped samples and then combining those members using an aggregated average in regression or plurality voting in classification. Bootstrapping is a sampling method where N samples are sampled randomly with replacement from a training set with N observations. Since an observation can be sampled more than once, the expected number of unique observations in a sample is 63% of the observations in the training set. Then the remainder of the training set is used as a validation set. It can be shown that if each single network is unstable that the bagging will create an ensemble classifier with lower variance than an individual network. This method is robust to model misspecification and overfitting [18], [20].

Another resampling technique, boosting, focuses training on the examples on which the algorithm performs the worst. The first boosting method which was only used for classification trained three classifiers. The first classifier was trained using a subsample of the training set and its performance, p_1 on the subsample was recorded. A second classifier was trained on a subsample composed of half new data from the training set and half on samples where p_1 was incorrect. Finally a third classifier was trained on the subset of the training set where the two previous classifiers disagreed. The final prediction was made by majority vote [19].

In Adaboost, a very popular boosting algorithm [21], a randomly chosen subsample of the training set is used to train each epoch. After each epoch, the observations on which the algorithm performed the worst are given greater weight to be chosen for the next epoch. The trained models (the current and previous epoch's models) are combined and tested after each epoch, and the examples are re-weighted [22]. Boosting has been applied to neural networks in contexts including scene classification [23], [24], star classification [25] and more generally [26].

The resampling technique used in this paper is **sub**sample **agg**regat**ing** or subbagging. The technique is best described through U-statistics and the related resampling techniques discussed in the next section.

3) U-Statistics: U-statistics (where the "U" refers to being unbiased) are an important, broad class of minimum variance unbiased estimators. They were considered early on by Kendall [27] and Wilcoxon [28] in the estimation of rank correlation. They were then formalized, being found to be of minimum variance by Halmos [29] and asymptotically normal by Hoeffding [30]. Good introductions to the subject can be found in Lehmann [31], Dasgupta [32], and Serfling [33], and more thorough treatment of the subject can be found in Lee [34]. More relevant to the purpose of this paper is that predictions from ensemble methods can be shown to be U-statistics given some regularity conditions. Here, we give an introduction to U-statistics [1] .

Consider an i.i.d random sample $X_1, \ldots, X_n$ from some population with cdf F. Here, we assume that F is unknown rather than belonging to some parametric class. Next, consider the expectation functional $\theta(F) = E[\phi(X_1, \ldots, X_a)]$, where kernel ϕ is assumed to be permutation symmetric. Then $\phi(X_1, \ldots, X_a)$ is an unbiased estimator for θ with sample of size n, as is $\phi(X_{i_1}, \ldots, X_{i_n})$ for any a-tuple, where $1 \leq i_1 \leq \cdots \leq i_n \leq n$. The minimum variance unbiased estimator of θ is given by

$$U = \frac{1}{\binom{n}{a}} \sum_{i_1} \cdots \sum_{i_a} \phi(X_{i_1}, \ldots, X_{i_a}).$$

Since ϕ is symmetric, so is U. Note that we can view the U-statistics as a generalization of the sample mean, where we average kernel ϕ over all $\binom{n}{a}$ subsamples of size a. Then U converges asymptotically to a normal distribution with variance $\frac{k^2}{n}\zeta_{1,k}$ where

$$\zeta_{1,k} = cov[\phi(X_1, \ldots, X_a); \phi(X_1, X_2', \ldots, X_a')]$$

and $X_2', \ldots, X_a' \sim F$ [30]. That is, asymptotic variance is proportional to the covariance of two subsamples having only one sample in common.

B. Related Work

1) Neural Network Ensembles: Negative correlation learning has been proposed as an ensemble neural network method [35]. In negative correlation learning, an ensemble of neural networks is trained simultaneously with a loss function that contains a penalty for the correlation between the networks. The loss function [36] is

$$\mathcal{L}(F_i) = \frac{1}{N}\left(\sum_{n=1}^{N}(F_i(n) - d(n))^2 + \lambda \sum_{n=1}^{N} p_i(n)\right)$$

$$p_i(n) = (F_i(n) - F(n)) \sum_{i \neq j}^{N} (F_j(n) - F(n))$$

[2]Assuming that each member's prediction was independent of the other members' prediction in the ensemble

where $F_i(n)$ is the output of network i on the nth training pattern, $F(n)$ is the average output of the ensemble on the nth training pattern, $d(n)$ is the target variable, and $0 < \lambda < 1$.

The mean squared error (MSE) of an ensemble can be decomposed into variance, covariance, and bias components. The larger λ is in the loss function, the larger the decrease in the covariance component of the MSE [37]. This type of loss function causes individual network members to decompose tasks into subtasks [36]. This method has recently been shown to be effective in random vector functional link networks [38].

Another approach to training neural network ensembles is to use a genetic feature selection algorithm [39]. The algorithm initializes a population of neural networks by varying the features on which each network is trained. Using genetic operations of mutation and crossover, the algorithm trains new networks for the population, then judging each network based on a fitness score:

$$Fitness_i = Accuracy_i + \lambda \cdot Diversity_i$$

The population of networks is then culled, λ is adjusted, and the culled population becomes the new ensemble. This process is repeated until a stopping criterion is reached [39].

2) Neural Networks and the Statistical Community: Geman *et al.* [40] showed consistency in neural networks by increasing the number of hidden layers asymptotically and demonstrating that, as the number of nodes increase, the amount of prediction bias decreases and prediction variance increases, the bias-variance trade off. The authors verified the result empirically on the MNIST dataset as well as generated data. They used both early stopping and cross validation as a form of variance reduction and framed overfitting as a form of variance enlargement.

An uncertainty measure assuming a Gaussian distribution for neural networks has been proposed by Tibshirani [20]. Assuming that the errors of a neural network are Gaussian with homoskedacity, the author measures the uncertainty of parameter estimates with a maximum likelihood estimator that is calculated using the Hessian. Since calculating the Hessian in large networks is impractical, this methodology has limitations.

The sandwich estimator has been proposed to estimate parameter variance in neural networks [20]. This method produces an asymptotically consistent covariance matrix without distributional assumptions or even an assurance that the correct model generated the parameter as long as that parameter is consistent. It is also robust to heteroskedacity [41], [42]. These relaxed conditions make it appealing for neural networks, though some researchers have raised concerns about its accuracy in practice[42], [43].

Bootstrapping has also been proposed to quantify uncertainty. This approach to quantification uses the bootstrap resampling process to generate error estimates. This has been shown to produce larger confidence intervals than models with stricter model assumptions [44].

Another uncertainty quantification method that has been applied to neural networks is the use of Taylor series expansion on a least squares estimator to approximate confidence intervals [45]. More recently, a method called MC (Monte Carlo) dropout has been used to quantify uncertainty in neural networks [46]. The method averages outputs over ensembles formed from subsamples of nodes of one neural network using dropout to select the active nodes in the network, turning a single network into an ensemble of networks. Another recent method uses samples from the training set, augments those samples with synthetic adversarial observations, and then assumes the average of the ensembles trained on the resulting sample follows the Gaussian distribution. [47].

Variance in neural networks can be divided into two categories, the accumulation of small random noise from unknown features in the data and the neural network's optimum approximation. It has been proved that the variance in neural networks can be reduced by training multiple neural networks while varying the initial conditions of the network, fixing both training set and architecture, and then averaging their results [48]. As mentioned above, Gaussian confidence intervals have been used to construct neural network confidence intervals [20]. However, these intervals typically overstate the true variability of neural network ensembles because the ensemble methods have a dampening effect on variance [49]. Bagging has also been used to estimate both confidence intervals and prediction intervals in neural network ensembles. Some research supports that these methods produce better coverage probabilities than the previously proposed Gaussian confidence intervals [49].

TABLE I: Notation Used

Term	Definition
n	Size of random sample
k_n	Size of subsample
m_n	Ensemble size
$\tilde{\mathbf{z}}^{(i)}$	Fixed sample
n_{MC}	Ensemble size trained with $\tilde{\mathbf{z}}^{(i)}$
$n_{\tilde{z}}$	Number of fixed samples

II. Neural Network Ensemble-based U-Statistics

In the following, we describe the theoretical motivation for our work. We follow this with an explanation of the algorithms implemented in our experiments. For this discussion, the main pieces of notation are explained in Table I.

A. Theoretical Motivation

Consider a random sample $(\boldsymbol{X}, \boldsymbol{Y}) \overset{\text{iid}}{\sim} F$ of size n. Suppose we build a neural network, N, from a subbagged sample of size a taken from our dataset. Suppose further that we do this for all $\binom{n}{a}$ subsamples. We can then take the average of the predictions for some $\boldsymbol{x}^*$ from these neural networks as the estimate of our predicted value. Let us write this sum as

$$b(\boldsymbol{x}^*) = \frac{1}{\binom{n}{a}} \sum_{(i)} N_{\boldsymbol{x}^*}((X_{i_1}, Y_{i_1}), \ldots, (X_{i_a}, Y_{i_a})).$$

If N is unbiased for $\theta = E[b(\boldsymbol{x}^*)]$, we have a procedure that results in a U-statistic for these predicted values. However, it is often computationally infeasible to build neural networks for all $\binom{n}{a}$ subsamples of the data. It has been shown that taking $m \leq \binom{n}{a}$ subsamples of size a results in

$$b_m(\boldsymbol{x}^*) = \frac{1}{m} \sum_{(i)} N_{\boldsymbol{x}^*}((X_{i_1}, Y_{i_1}), \ldots, (X_{i_a}, Y_{i_a})),$$

which is an incomplete U-statistic. This has been shown to be asymptotically normal and unbiased by [50], assuming the variance of the estimator converges to zero at a rate faster than $\sqrt{n}$. Neural networks have been shown to be strongly consistent with a convergence rate of $\frac{k \log n}{n} \to 0$, where k is the number of hidden nodes [51]. It may also make sense for us to scale m with n. Specifically, we could consider taking subsamples of size $m_n = \binom{n}{k_n}$ giving us

$$b_{m_n,k_n}(\boldsymbol{x}^*) = \frac{1}{m_n} \sum_{(i)} N_{\boldsymbol{x}^*}((X_{i_1}, Y_{i_1}), \ldots, (X_{i_a}, Y_{i_a})),$$

which is an infinite order U-statistic or a resampled statistic when $m_n \neq \binom{n}{k_n}$.

Frees developed necessary and sufficient conditions for asymptotic normality when m_n grows faster than n [52]. Mentch and Hooker developed conditions for individual means $E[b(\boldsymbol{x}^*)]$ for all growth rates of m_n with respect to n [1]. We omit the details here, but so long as the regression function is bounded and has exponential tails, the variance of the kernel function ϕ is bounded, $\lim \frac{n}{m_n} = \alpha$, $\lim \frac{k_n}{\sqrt{n}} = 0$, and $\lim \sigma_{1,k_n} \neq 0$, then the infinite order U-statistic will be asymptotically normal with the following distributions [1]:

$$\text{if } \alpha = 0, \text{ then } \frac{\sqrt{n}(U_{n,k_n,m_n} - \theta_{k_n})}{\sqrt{k_n^2 \zeta_{1,k_n}}} \xrightarrow{d} \mathcal{N}(0,1)$$

$$\text{if } 0 < \alpha < \infty, \text{ then } \frac{\sqrt{m_n}(U_{n,k_n,m_n} - \theta_{k_n})}{\sqrt{\frac{k_n^2}{\alpha}\zeta_{1,k_n} + \zeta_{k_n,k_n}}} \xrightarrow{d} \mathcal{N}(0,1)$$

$$\text{if } \alpha = \infty, \text{ then } \frac{\sqrt{m_n}(U_{n,k_n,m_n} - \theta_{k_n})}{\sqrt{\zeta_{k_n,k_n}}} \xrightarrow{d} \mathcal{N}(0,1)$$

We will choose k_n approximately on the order of $\sqrt{n}$ [1]. This choice of k_n replaces the requirement of exponential tails on the error distribution with the requirement that $nP(|\epsilon| > \sqrt{n}) \to 0$. It also assures that the $\lim \frac{k_n}{\sqrt{n}} = 0$. Finally by choosing a small k_n, the time complexity is similar to a bootstrap method, while generating large ensembles. It should be noted that it is not a requirement to choose k_n on the order of $\sqrt{n}$.

The subbagging method assumes that the estimator is built using the same procedure. The distributional results above do not rely on the method of building the neural network outside the weak regularity conditions. However, the subbagging procedure does require that each neural network be built using the same method. This would preclude using dropout as each estimator would be built using randomly selected samples of nodes on each training interval. We would then need a different justification for the estimator, similar to what Mentch and Hooker did for random forests [1].

Algorithm 1 NN Subbagging

1: Select size of subsamples k_n and number of subsamples m_n
2: **for** i in 1 to m_n **do**
3: Take subsample of size k_n from training set
4: Train NN using subsample
5: NN estimates $\mathbf{x}^*$
6: Store NN estimation
7: Average m_n predictions estimate $b_{n,k_n,m_n}(\boldsymbol{x}^*)$

Algorithm 2 NN ζ_{1,k_n} Estimation

1: **for** i in 1 to $n_{\tilde{z}}$ **do**
2: Select initial fixed point $\tilde{\mathbf{z}}^{(i)}$
3: **for** i in 1 to n_{MC} **do**
4: Select subsample $S_{\tilde{\mathbf{z}}^{(i)},j}$ of size k_n
5: Train NN using subsample $S_{\tilde{\mathbf{z}}^{(i)},j}$
6: Store NN prediction at $\mathbf{x}^*$
7: Record average of the n_{MC} predictions
8: Compute variance of the $n_{\tilde{z}}$ averages

B. Algorithms

Mentch and Hooker propose two routes to estimating variance using infinite order U-statistics. The first method is described as an external method because the estimation of ζ_{1,k_n} and ζ_{k_n,k_n} are done in a separate step outside of the point estimation. The second method is called internal as ζ_{1,k_n} and ζ_{k_n,k_n} are both estimated in the same procedure as the point estimate. The external algorithms are described here.

The neural network subbagging method, Algorithm 1, trains a neural network on a subsample of the random sample, suggested to be on the order of $\sqrt{n}$. Once the network is trained, the point estimate is saved. Another neural network is then trained on a new subsample whose point estimates are saved, and this process is repeated until a collection of k_n point estimates is obtained. The average of the point estimates is the subbagging method's estimate.

The estimation of ζ_{1,k_n} in Algorithm 2 is done by keeping one sample constant between subsamples. The size of the subsample is again suggested to be on the order of $\sqrt{n}$. The neural network is trained on this subsample, and its prediction is stored. For n_{MC} trials, a new sample is selected with the same fixed sample point $\tilde{\mathbf{z}}^{(i)}$ in each sample, then the average of the n_{MC} predictions is recorded. Another $\tilde{\mathbf{z}}^{(i)}$ is selected and the process repeated $n_{\tilde{z}}$ times, then the variance of the collection of ensemble predictions is computed.

To estimate ζ_{k_n,k_n} in Algorithm 3, we repeat the previously mentioned subbagging procedure for $n_{\tilde{z}}$ rather than m_n times, and instead of calculating the mean of the ensemble point estimate, we calculate the variance of the point estimates. We can combine the three algorithms into the internal variance estimation (Algorithm 4) [1].

Algorithm 3 NN ζ_{k_n,k_n} Estimation

for i in 1 to $n_{\tilde{z}}$ **do**
    Select subsample, S_{k_n}, of size k_n
    Train NN using subsample S_{k_n}
    Store NN prediction at $\mathbf{x}^*$
Compute variance of the $n_{\tilde{z}}$ predictions

Algorithm 4 Internal Variance Estimation ζ_{1,k_n} and ζ_{k_n,k_n}

for i in 1 to $n_{\tilde{z}}$ **do**
    Select initial fixed point $\tilde{z}^{(i)}$
    for j in 1 to n_{mc} **do**
        Select subsample $S_{i,j}$ of size $(k_n - 1)$
        Append $\tilde{z}^{(i)}$ to $S_{i,j}$
        Train NN
        NN estimates $N_{\boldsymbol{x}^*}((X_{k_1}, Y_{k_1}), \ldots, (X_{k_n}, Y_{k_n}))$
        Store NN estimation
    Record average prediction n_{mc} NNs
Compute variance for $n_{\tilde{z}}$ averages to estimate ζ_{1,k_n}
Compute variance of all predictions to estimate ζ_{k_n,k_n}
Compute the mean of all predictions to estimate θ_{k_n}

Rather than estimating θ_k outside the variance estimation, it becomes the average of the fixed point ensemble used to estimate ζ_{1,k_n}. This also addresses the primary bottleneck of the external method.

The subbagging method has been found to be most effective when n_z is an order of magnitude smaller than the n_{MC}. Since it is computationally efficient to keep n_z small and thereby keep n_{MC} from exploding, the estimate of ζ_{k_n,k_n} in the external method tends to be unstable. By estimating ζ_{k_n,k_n} through the variance of all ensemble networks generated for the ζ_{1,k_n} calculation, the estimates become more stable since the ensemble size is several orders of magnitude larger than n_z.

III. Experiments

Our experiments assess empirically through a coverage probability study of two simulated regression functions, the confidence interval estimates using resampled statistics [1] of neural network point estimates. The two functions are the simple linear regression (SLR) model given in Equation 1 and the Weibull model as a simple nonlinear regression (SNLR) model given in Equation 2. Examples of samples drawn uniformly from their supports are shown in Figure 3. The SNLR model has been used as a model for pasture regrowth [53].

$$f(x) = 2x_1 + \epsilon; \;\; \mathcal{X} = [0, 20] \;\; \epsilon \sim N(0, 10) \tag{1}$$

$$f(x, \theta) = 70 - 61 \exp(-\exp(-10 + 2.4 \log\ x)) + \epsilon; \\ \mathcal{X} = [0, 80]; \epsilon \sim N(0, 10) \tag{2}$$

TABLE II: Experimental Parameters

	SLR	SNLR 1	SNLR 2
Coverage Study Size	500	150	120
n	1000	1000	1000
k_n	31	31	31
m_n	51	51	153
n_{MC}	17	17	51
$n_{\tilde{z}}$	3	3	3
Learning Rate	.01	.01	.01
Hidden Layers	1	1	1
Number of Nodes	10	12	12
Activation	Sigmoid	Sigmoid	Sigmoid
Epochs	1500	3000	3000

To ascertain the reliability of the variance estimations generated from the subbagging method empirically, we generated a large number of confidence intervals to estimate the number of true parameters that fell within the α confidence interval.

We built a feed forward network with one layer, sigmoid activation functions, squared error loss, and using the stochastic gradient descent method, Adam, [4] from the framework, Keras [54] with a Tensorflow backend [55]. A GPU trial was performed, but the experiments were found to be faster on the CPU. Experiments were performed on 8 3.4 GHz cores. Hyperparameters are in Table II.

IV. Results

A. Internal Variance Estimation Coverage Probabilities

We generated 500 variance and mean estimates of $\mathbf{x}^* = \mathbf{10}$ of SLR (Equation 1) using a small feed forward network with one hidden layer of 10 nodes. On the suggestion of Mentch and Hooker, we chose n_{MC} to be substantially larger than $n_{\tilde{z}}$. We found that the estimated two-standard deviation confidence interval covered the true mean $96 \pm 0.34\%$ of the time, and the estimated three-standard deviation confidence interval covered the true mean $99.6 \pm 0.036\%$ of the time. If these confidence intervals were generated from a standard normal distribution, we would expect the confidence intervals to cover 95.5% and 99.7% respectively.

For SNLR (Equation 2), we ran two coverage studies, both predicting $\mathbf{x}^* = \mathbf{45}$. The first, SNLR 1, was trained for more epochs and had the same size m_n as the SLR coverage study. In this study we produced 150 confidence intervals, which covered the true mean 90.88 $\pm$ 1.3% and 98.66 $\pm 0.2\%$ respectively. We believe that the failure of the confidence intervals to adhere to the standard normal distribution is due to the bias presented in the number of n_{MC}, so we increased the number in the second coverage study (SNLR 2). In this coverage study with 120 confidence intervals, the true mean was covered 97.5 $\pm 0.44\%$ and 100% respectively. A histogram of the SLR and SNLR2 point estimates is given in Figure 2.

B. External Variance Estimation Coverage Probabilities

We performed two smaller coverage studies with 150 and 50 trials respectively on SLR using the external variance methods. The first study kept the parameters from the internal algorithm

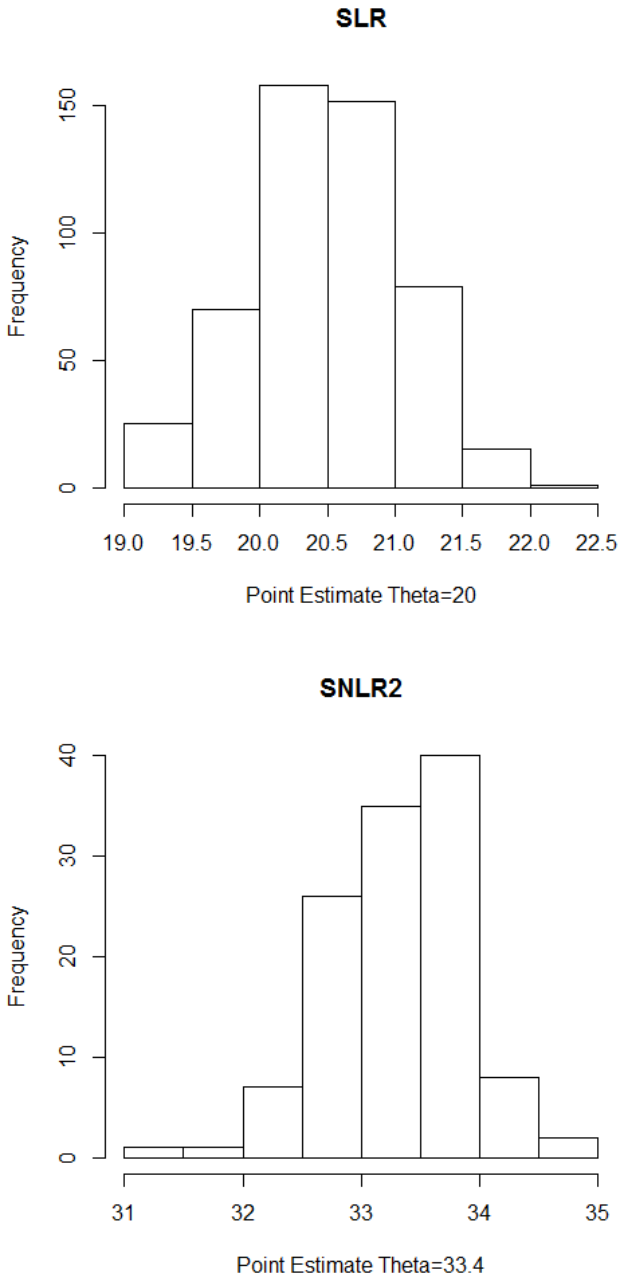

Fig. 2: Coverage Study Results

except $n_M C = 30$. Since ζ_{k_n,k_n} was only estimated with $n_z = 3$, the variance calculation was unstable with ζ_{k_n,k_n} displaying high variance, resulting in unsatisfactory coverage probabilities. The second study with 50 trials was done using $n_M C = 30$ and $n_z = 30$, and the coverage probabilities were $0.96 \pm 1.08\%$ and $0.98 \pm 0.55\%$.

V. Discussion

From the internal coverage study with a m_n of 5% and 15% of the random sample, the point estimates displayed a small bias $\approx 2.5\%$, but the wider confidence intervals still covered the true mean at rates consistent with a standard Gaussian distribution. In preparation for the coverage probability study, we built large neural network ensembles of approximately the same size as the random sample, and we found our point estimates to be very accurate. This leads us to believe that small bias was introduced by m_n rather than the neural network estimator.

Empirically, the external method provided the same coverage probability but does not appear to have any offsetting advantages to the increased computationally costs of use. We suggest that the practitioner use the internal method as the theoretical guarantees are the same, but the computational cost is smaller.

This neural network ensemble method may be useful in situations with large numbers of observations. It may be impractical to train a network using the full training set, but it may be feasible to train many networks on smaller numbers of observations and then use the resulting ensemble prediction with the ancillary effect of generating accurate variance estimation.

On a practical note, we observed that the confidence intervals on the edges of the sample space were consistently wider than for point estimates in the interior. With more samples on one side of the region than the other, this is to be expected, but we suggest that only internal point estimates be examined with this method.

VI. Future Work

Many exciting research questions remain open. We believe the most pressing is to study the application of this method using real-world data sets with known variance properties. These studies will strengthen the argument for use by practitioners.

Another useful research opportunity is to extend the theory beyond regression. Most neural networks predict multidimensional targets, so the limitation to regression restricts the method's use. U-statistics of covariance matrices are currently an active research topic [56].

Another direction is the comparison of variance estimation with Monte Carlo dropout and adversarial resampling in neural network regression settings. Especially Monte Carlo dropout has been applied widely in practice (see, for example, [57]), and the subbagging method proposed here provides a viable alternative to quantifying uncertainty in regression.

We would also like to extend the subbagging procedure to neural networks that are built using dropout [58]. Dropout randomly selects nodes to be turned off during training, providing a regularization and improving accuracy. This, however, means that each ensemble member is now trained using a different estimator. This extension should be possible as Mentch and Hooker show that the random forests can be used as estimators in a modified framework.

VII. Acknowledgments*

We would like to thank members of the Numerical Intelligent Systems Laboratory at Montana State University for many useful discussions in the completion of this project. We would

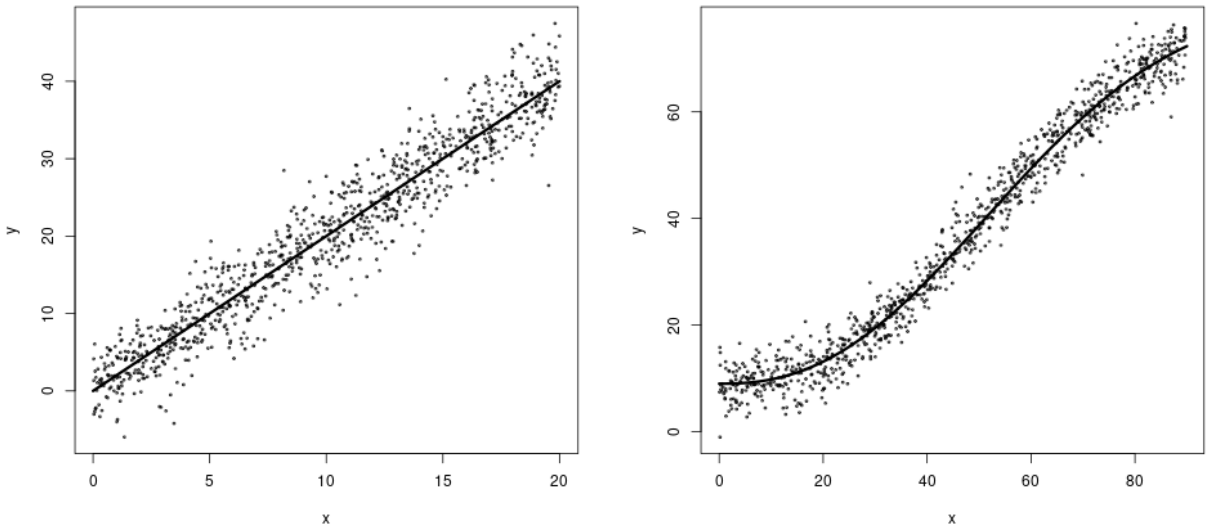

Fig. 3: Examples of samples of size $n = 1000$ drawn from the linear model (left) and nonlinear model (right)

also like to thanks members of the 2018 CSCI 547 Machine Learning class following presentation of an earlier version of this work.

REFERENCES

[1] L. Mentch and G. Hooker, "Quantifying uncertainty in random forests via confidence intervals and hypothesis tests," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 841–881, 2016.

[2] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[3] D. F. Specht, "A general regression neural network," *IEEE transactions on neural networks*, vol. 2, no. 6, pp. 568–576, 1991.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[5] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[6] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[7] H. Li, X. Wang, and S. Ding, "Research and development of neural network ensembles: A survey," *Artificial Intelligence Review*, vol. 49, no. 4, pp. 455–479, 2018.

[8] S. Masoudnia and R. Ebrahimpour, "Mixture of experts: A literature survey," *Artificial Intelligence Review*, vol. 42, no. 2, pp. 275–293, 2014.

[9] M. Woźniak, M. Graña, and E. Corchado, "A survey of multiple classifier systems as hybrid systems," *Information Fusion*, vol. 16, pp. 3–17, 2014.

[10] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, "Ensemble approaches for regression: A survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, p. 10, 2012.

[11] L. Rokach, *Pattern classification using ensemble methods*. World Scientific, 2010, vol. 75.

[12] N. J. Nilsson, "Learning machines.," 1965.

[13] L. Kanal, "Patterns in pattern recognition: 1968-1974," *IEEE Transactions on information theory*, vol. 20, no. 6, pp. 697–722, 1974.

[14] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[15] M. P. Perrone and L. N. Cooper, "When networks disagree: Ensemble methods for hybrid neural networks," BROWN UNIV PROVIDENCE RI INST FOR BRAIN and NEURAL SYSTEMS, Tech. Rep., 1992.

[16] M. P. Perrone, "General averaging results for convex optimization," in *Proceedings of the 1993 Connectionist Models Summer School*, 1994, pp. 364–371.

[17] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in neural information processing systems*, 1995, pp. 231–238.

[18] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[19] R. E. Schapire, "The strength of weak learnability," *Machine learning*, vol. 5, no. 2, pp. 197–227, 1990.

[20] R. Tibshirani, "A comparison of some error estimates for neural network models," *Neural Computation*, vol. 8, no. 1, pp. 152–163, 1996.

[21] Y. Freund, R. E. Schapire, *et al.*, "Experiments with a new boosting algorithm," in *Icml*, Citeseer, vol. 96, 1996, pp. 148–156.

[22] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[23] F. Zhang, B. Du, and L. Zhang, "Scene classification via a gradient boosting random convolutional network framework," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1793–1802, 2016.

[24] L. Liu, Y. Hua, Q. Zhao, H. Huang, and A. C. Bovik, "Blind image quality assessment by relative gradient statistics and adaboosting neural network," *Signal Processing: Image Communication*, vol. 40, pp. 1–15, 2016.

[25] N. S. Philip, Y. Wadadekar, A. Kembhavi, and K. B. Joseph, "A difference boosting neural network for automated star-galaxy classification," *Astronomy & Astrophysics*, vol. 385, no. 3, pp. 1119–1126, 2002.

[26] H. Schwenk and Y. Bengio, "Boosting neural networks," *Neural computation*, vol. 12, no. 8, pp. 1869–1887, 2000.

[27] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[28] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[29] P. R. Halmos *et al.*, "The theory of unbiased estimation," *The Annals of Mathematical Statistics*, vol. 17, no. 1, pp. 34–43, 1946.

[30] W. Hoeffding, "A class of statistics with asymptotically normal distribution," *The annals of mathematical statistics*, pp. 293–325, 1948.

[31] E. L. Lehmann, *Elements of large-sample theory*. Springer Science & Business Media, 2004.

[32] A. DasGupta, *Asymptotic Theory of Statistics and Probability*. Springer, 2008.

[33] R. J. Serfling, *Approximation theorems of mathematical statistics*. John Wiley & Sons, 2009, vol. 162.

[34] A. Lee, "U-statistics," *Theory and Practice," Marcel Dekker, New York*, 1990.

[35] Y. Liu and X. Yao, "Negatively correlated neural networks can produce best ensembles," *Australian journal of intelligent information processing systems*, vol. 4, no. 3/4, pp. 176–185, 1997.

[36] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.

[37] G. Brown and J. Wyatt, "Negative correlation learning and the ambiguity family of ensemble methods," in *International Workshop on Multiple Classifier Systems*, Springer, 2003, pp. 266–275.

[38] M. Alhamdoosh and D. Wang, "Fast decorrelated neural network ensembles with random weights," *Information Sciences*, vol. 264, pp. 104–117, 2014.

[39] D. W. Opitz, "Feature selection for ensembles," *AAAI/IAAI*, vol. 379, p. 384, 1999.

[40] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[41] A. Zeileis, "Object-oriented computation of sandwich estimators," *Journal of Statistical Software, Articles*, vol. 16, no. 9, pp. 1–16, 2006, ISSN: 1548-7660. DOI: 10.18637/jss.v016.i09. [Online]. Available: https://www.jstatsoft.org/v016/i09.

[42] G. Kauermann and R. J. Carroll, "A note on the efficiency of sandwich covariance matrix estimation," *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1387–1396, 2001.

[43] C.-F. J. Wu, "Jackknife, bootstrap and other resampling methods in regression analysis," *the Annals of Statistics*, pp. 1261–1295, 1986.

[44] B. Efron and T. Hastie, *Computer age statistical inference*. Cambridge University Press, 2016, vol. 5.

[45] I. Rivals and L. Personnaz, "Construction of confidence intervals for neural networks based on least squares estimation," *Neural Networks*, vol. 13, no. 4-5, pp. 463–484, 2000.

[46] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[47] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, 2017, pp. 6402–6413.

[48] U. Naftaly, N. Intrator, and D. Horn, "Optimal ensemble averaging of neural networks," *Network: Computation in Neural Systems*, vol. 8, no. 3, pp. 283–296, 1997.

[49] J. G. Carney, P. Cunningham, and U. Bhagwan, "Confidence and prediction intervals for neural network ensembles," in *International Joint Conference on Neural Networks, 1999. IJCNN'99*, IEEE, vol. 2, 1999, pp. 1215–1218.

[50] S. Janson, "The asymptotic distributions of incomplete u-statistics," *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, vol. 66, no. 4, pp. 495–505, 1984.

[51] A. Faragó and G. Lugosi, "Strong universal consistency of neural network classifiers," *IEEE Transactions on Information Theory*, vol. 39, no. 4, pp. 1146–1151, 1993.

[52] E. W. Frees, "Infinite order u-statistics," *Scandinavian Journal of Statistics*, pp. 29–45, 1989.

[53] S. Huet, A. Bouvier, M.-A. Poursat, and E. Jolivet, *Statistical tools for nonlinear regression: a practical guide with S-PLUS and R examples*. Springer Science & Business Media, 2006.

[54] F. Chollet *et al.*, *Keras*, https://keras.io, 2015.

[55] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, vol. 16, 2016, pp. 265–283.

[56] S. Minsker and X. Wei, "Robust modifications of u-statistics and applications to covariance estimation problems," *arXiv preprint arXiv:1801.05565*, 2018.

[57] J. Senecal and J. Sheppard, "Efficient convolutional neural networks for multi-spectral image classification," in *submitted IEEE International Joint Conference on Neural Networks*, 2018.

[58] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.