Spring 2020

Conformal prediction intervals for neural networks using cross validation

Saeed Khaki
Iowa State University

Conformal prediction intervals for neural networks using cross validation

by

Saeed Khaki

A Creative Component submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Statistics

Program of Study Committee:
Dan Nettleton, Major Professor
Lizhi Wang
Peng Liu
Dan Nordman

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this creative component. The Graduate College will ensure this creative component is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2020

DEDICATION

I would like to dedicate this creative component to my family. My wife Zahra who has always supported me with her unconditional love. A special feeling of gratitude to my loving parents, Ali Akbar and Monireh without whose support I would not have been able to complete this work. Last but not least my sister Mina who has never left my side.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this creative component. First and foremost, Dr. Dan Nettleton for his guidance, patience and support throughout this research and the writing of this creative component. His insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee members for their efforts and contributions to this work.

ABSTRACT

Neural networks are among the most powerful nonlinear models used to address supervised learning problems. Similar to most machine learning algorithms, neural networks produce point predictions and do not provide any prediction interval which includes an unobserved response value with a specified probability. In this creative component, we propose the k-fold prediction interval method to construct prediction intervals for neural networks based on k-fold cross validation. Simulation studies and analysis of 10 real datasets are used to compare the finite-sample properties of the prediction intervals produced by the proposed method and the split conformal (SC) method. The results suggest that the proposed method tends to produce narrower prediction intervals compared to the SC method while maintaining the same coverage probability. Our experimental results also reveal that the proposed k-fold prediction interval method produces effective prediction intervals and is especially advantageous relative to competing approaches when the number of training observations is limited.

CHAPTER 1. INTRODUCTION

Neural networks are mathematical functions that map some set of input values to output values (Goodfellow et al., 2016). Neural network models belong to the class of representation learning methods that automatically discover the underlying representations of data. A neural network model is composed of multiple processing layers that each transforms the representation at one level into a more abstract representation starting from the raw input (LeCun et al., 2015). As such, a very complex function can be learned if we combine enough transformations. Such transformations are obtained by stacking nonlinear modules (LeCun et al., 2015; Goodfellow et al., 2016). Neural networks are also known to be universal approximators which means that regardless of the function we want to learn, a large enough neural network can represent such a function (Hornik et al., 1989). However, learning the desired function using neural networks is challenging, and there is no guarantee that we can find the right parameters for neural networks (Goodfellow et al., 2016).

Neural networks are multilayer networks of neurons that are used to solve classification and regression problems. Figure 1.1 shows a neural network with a single hidden layer.

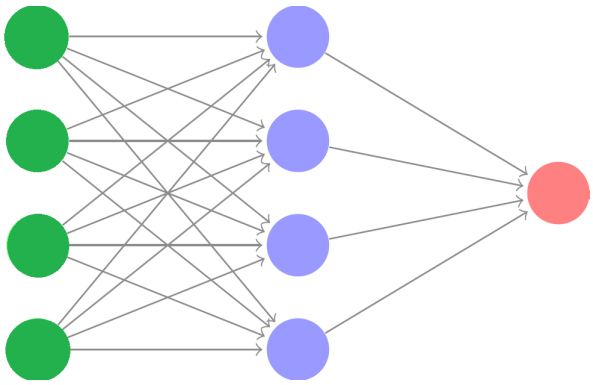

Figure 1.1: A neural network with a single hidden layer.

Starting from the left in Figure 1.1, we have:

- The input layer of the neural network in green.
- The hidden layer of neurons in blue.
- The output layer in red.
- The weights of the network are denoted with arrows.

Neurons are the building blocks of neural networks. A neuron computes the weighted average of its input, and this sum is passed through a nonlinear function, often called an activation function. With an identity activation function, a neuron can be considered as a multiple linear regression. Many activation functions have been proposed in the literature such as sigmoid, tanh, ReLU and Leaky ReLU. We review each briefly below.

sigmoid: this activation function is defined as $y = \frac{1}{1+\exp(-x)}$. The sigmoid activation function is historically popular because the function output has a nice interpretation as probability. This function squashes input numbers to the range $[0, 1]$. The sigmoid activation function has a vanishing gradient problem which is a difficulty found in training neural networks with gradient-based optimization methods using backpropagation. Backpropagation calculates the gradient of the loss function with respect to the neural network's weights. In gradient-based optimization methods, weights of neural networks receive an update proportional to the partial derivative of the loss function with respect to the current weight in each iteration of training. As such, if the partial derivatives are very small (almost zero), it may prevent the neural network from further training. This problem usually happens when an activation function gets saturated, meaning it predominantly outputs values close to the asymptotic ends of the bounded activation function. In the saturated regions of an activation function, the partial derivatives are close to zero, which leads to the vanishing gradient problem. The output of the sigmoid activation function is not zero centered, which also may prevent the neural network from proper training. Having the zero-centered property is very important because it can accelerate the training of neural networks. However, to address this issue, a batch normalization technique has been proposed to normalize the output of the activation functions in neural networks (Ioffe and Szegedy, 2015).

tanh: this activation function is defined as $y = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$. This activation function squashes numbers to the range $[-1, 1]$ and its output is zero-centered. But, it has both saturation and vanishing gradient problems.

ReLU: this activation function is defined as $y = \max(0, x)$. It is very computationally efficient and does not have the saturation or the vanishing gradient problems. However, it is not zero-centered, and sometimes the ReLU activation function dies, which means it always outputs the same value (zero) for any input.

Leaky ReLU: this activation function is defined as $y = \max(0.1x, x)$. This activation function is very similar to the ReLU with all above mentioned properties except that it never dies.

The choice of activation functions has a considerable effect on the performance of a neural network. Currently, the most successful and widely used activation function is the ReLU activation function (Ramachandran et al., 2017; Glorot and Bengio, 2010).

The number of layers in a neural networks is a hyperparameter. In general, as the number of layers or neurons is increased, the complexity of the model also increases. Deeper networks are more powerful models, but they are harder to train due to the vanishing gradient problem. Increasing the number of hidden layers might reduce classification or regression errors, but it may also cause the vanishing gradient problem (He et al., 2016). There have been many attempts to solve the vanishing gradient problem such as (1) use activation functions that are less prone to the vanishing gradient problem (e.g. ReLU), (2) use residual blocks to preserve the gradients (He et al., 2016), and (3) use multiple auxiliary loss functions throughout the network (Goodfellow et al., 2016).

The choice of the loss function depends on the type of the problem that we would like to solve. For example, the cross-entropy loss (log loss) is usually used for classification problems. For regression problems, squared loss or absolute loss are usually used. Among gradient-based optimization methods, the following are popular:

- Adam (Kingma and Ba, 2014)
- AdaGrad (Duchi et al., 2011)

- Adadelta (Zeiler, 2012)
- Nesterov Accelerated Gradient (Sutskever et al., 2013)

The weights of the neural networks are usually initialized with small random numbers from a normal distribution or a uniform distribution. More advanced initialization methods have also been proposed such as Xavier initialization (Glorot and Bengio, 2010), which adjusts the standard deviation of the normal distribution based on the number of neurons in the network.

The training process of neural networks is an iterative process including the following steps:

1. Forward propagation: inputs provide initial information and then propagate to the neurons at each layer and finally produce the output.
2. Computing the loss function: the outputs of the network are compared with the response variable values to compute the neural network prediction errors and the value of the loss function for the current weights.
3. Backpropagation: the gradient of the loss function with respect to the neural network's weights is calculated.
4. Updating the weights of the network: the weights of the neural network get updated using gradient-based optimization methods to reduce the prediction error.

Similar to most machine learning methods for prediction, neural networks usually produce point predictions without any information about how far point predictions are from the ground truth response variables. Because point predictions produced by neural networks do not assess the prediction error from the same data used to generate point predictions, neural networks are lacking in inferential capability from a statistical standpoint. In this creative component, we develop prediction intervals based on neural network point predictions that produce a range of values including an unknown continuous univariate response with any specified level of confidence.

Following the setup in (Zhang et al., 2019), $(X, Y) \in \mathbb{R}^p \times \mathbb{R}$ denote the predictor-response pair randomly sampled from some distribution $\mathbb{G}$, where p is the number of predictors and Y

is a continuous univariate response. We develop a prediction interval for the observation (X, Y) denoted as $I_\alpha(X, C_n)$ that will cover the true response variable with the probability $1 - \alpha$, where C_n is a training set including observations $(X_1, Y_1), ..., (X_n, Y_n) \overset{\text{iid}}{\sim} \mathbb{G}$ and (X, Y) is independent of the training set C_n.

Several approaches have been proposed to construct prediction intervals for neural networks. For example, Hwang and Ding (1997) proposed an asymptotic approach to construct prediction intervals for neural networks. They estimated the asymptotic variance of the neural network predictions and used the $1 - \alpha/2$ quantile of a t-distribution to create prediction intervals. De Vleaux et al. (1998) constructed prediction intervals for neural networks based on the asymptotic variance of the estimated parameters of the neural networks. Khosravi et al. (2010) proposed a lower upper bound estimation method to construct two outputs for a neural network model for estimating the prediction interval bounds. Kivaranovic et al. (2019) proposed a distribution-free split conformal prediction interval for neural networks. They designed a prediction interval network which had three outputs to estimate the median and the lower and upper bounds of prediction intervals.

The conformal prediction interval framework is a general method to construct prediction intervals and provides distribution-free predictive inference (Vovk et al., 2005). Many prediction interval methods have been proposed based on conformal inference. For example, Lei et al. (2018) proposed a distribution-free predictive inference for regression leading to split conformal (SC) prediction intervals. However, the SC method may not always have a good performance, especially when the sample size is small because the SC approach uses only half of the data to train the regression function, which may not be sufficient. In this creative component, we propose a k-fold prediction interval method to construct prediction intervals based on k-fold cross validation. This method tends to produce narrower prediction intervals compared to SC intervals while maintaining the same coverage probability. Our experimental results suggested that the proposed k-fold prediction interval method is effective and especially advantageous when the number of training observations is limited.

1.1 References

De Vleaux, R. D., Schumi, J., Schweinsberg, J., and Ungar, L. H. (1998). Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*, volume 1. MIT Press Cambridge.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hwang, J. G. and Ding, A. A. (1997). Prediction intervals for artificial neural networks. *Journal of the American Statistical Association*, 92(438):748–757.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2010). Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE transactions on neural networks*, 22(3):337–346.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kivaranovic, D., Johnson, K. D., and Leeb, H. (2019). Adaptive, distribution-free prediction intervals for deep neural networks. *arXiv preprint arXiv:1905.10634*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., and Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.

Vovk, V., Gammerman, A., and Shafer, G. (2005). *Algorithmic learning in a random world*. Springer Science & Business Media.

Zeiler, M. D. (2012). Adadelta: An adaptive learning rate method.

Zhang, H., Zimmerman, J., Nettleton, D., and Nordman, D. J. (2019). Random forest prediction intervals. *The American Statistician*, (just-accepted):1–20.

CHAPTER 2. Conformal prediction intervals for neural networks using cross validation

A paper submitted to *ICSA Applied Statistics Symposium*

Saeed Khaki and Dan Nettleton

2.1 Abstract

Neural networks are among the most powerful nonlinear models used to address supervised learning problems. Similar to most machine learning algorithms, neural networks produce point predictions and do not provide any prediction interval which includes an unobserved response value with a specified probability. In this paper, we proposed the k-fold prediction interval method to construct prediction intervals for neural networks based on k-fold cross validation. Simulation studies and analysis of 10 real datasets are used to compare the finite-sample properties of the prediction intervals produced by the proposed method and the split conformal (SC) method. The results suggest that the proposed method tends to produce narrower prediction intervals compared to the SC method while maintaining the same coverage probability. Our experimental results also reveal that the proposed k-fold prediction interval method produces effective prediction intervals and is especially advantageous relative to competing approaches when the number of training observations is limited.

2.2 Introduction

Neural networks are mathematical functions that map some set of input values to output values (Goodfellow et al., 2016). Neural network models belong to the class of representation learning methods that automatically discover the underlying representations of data. A neural network model is composed of multiple processing layers that each transforms the representation at one

level into a more abstract representation starting from the raw input (LeCun et al., 2015). As such, a very complex function can be learned if we combine enough transformations. Such transformations are obtained by stacking nonlinear modules (LeCun et al., 2015; Goodfellow et al., 2016). Neural networks are also known to be universal approximators which means that regardless of the function we want to learn, a large enough neural networks can represent such a function (Hornik et al., 1989). However, learning the desired function using neural networks is challenging and there is no guarantee that we can find the right parameters for the neural networks (Goodfellow et al., 2016).

Similar to most machine learning methods for prediction, neural networks usually produce point predictions without any information about how far point predictions are from the ground truth response variables. Because point predictions produced by neural networks do not assess the prediction error from the same data used to generate point predictions, neural networks are lacking in inferential capability from a statistical standpoint. In this paper, we develop prediction intervals based on neural network point predictions that produce a range of values including an unknown continuous univariate response with any specified level of confidence.

Following the setup in (Zhang et al., 2019), $(X, Y) \in \mathbb{R}^p \times \mathbb{R}$ denote the predictor-response pair randomly sampled from some distribution $\mathbb{G}$, where p is the number of predictors and Y is a continuous univariate response. We develop a prediction interval for the observation (X, Y) denoted as $I_\alpha(X, C_n)$ that will cover the true response variable with the probability $1 - \alpha$, where C_n is a training set including observations $(X_1, Y_1), ..., (X_n, Y_n) \overset{\text{iid}}{\sim} \mathbb{G}$ and (X, Y) is independent of the training set C_n.

Several approaches have been proposed to construct prediction intervals for neural networks. For example, Hwang and Ding (1997) proposed an asymptotic approach to construct prediction intervals for neural networks. They estimated the asymptotic variance of the neural network predictions and used the $1 - \alpha/2$ quantile of a t-distribution to create prediction intervals. De Vleaux et al. (1998) constructed prediction intervals for neural networks based on the asymptotic variance of the estimated parameters of the neural networks. Khosravi et al. (2010) proposed a lower upper bound estimation method to construct two outputs for a neural network model for estimating the

prediction interval bounds. Kivaranovic et al. (2019) proposed a distribution-free split conformal prediction interval for neural networks. They designed a prediction interval network which had three outputs to estimate the median and the lower and upper bounds of prediction intervals.

The conformal prediction interval framework is a general method to construct prediction intervals and provides distribution-free predictive inference (Vovk et al., 2005). Many prediction interval methods have been proposed based on conformal inference. For example, Lei et al. (2018) proposed a distribution-free predictive inference for regression leading to split conformal (SC) prediction intervals. However, the SC method may not always have a good performance, especially when the sample size is small because the SC approach uses only half of the data to train the regression function, which may not be sufficient. In this creative component, we propose a k-fold prediction interval method to construct prediction intervals based on k-fold cross validation. This method tends to produce narrower prediction intervals compared to SC intervals while maintaining the same coverage probability. Our experimental results suggested that the proposed k-fold prediction interval method is effective and especially advantageous when the number of training observations is limited.

The remainder of this paper is organized as follows. Section 2 describes the methodology. Section 3 presents the simulation study. Section 4 explains the data analysis results. Finally, we conclude the paper in section 5.

2.3 k-fold Conformal Prediction Intervals

The conformal prediction interval framework is a general approach for efficiently constructing prediction intervals (Vovk et al., 2005). To decrease the computational cost of the full conformal method, Lei et al. proposed split conformal prediction intervals, which are considerably computationally more efficient than the full conformal method (Lei et al., 2018). The SC prediction interval algorithm includes the following steps:

1. Randomly split $\{1, ..., n\}$ into two equal-sized subsets $L1$ and $L2$.

2. Train a regression function from $\{(X_i, Y_i) : i \in L_1\}$ to estimate the mean function denoted as $\hat{m}_{n/2}(X)$.

3. For $i \in L_2$, compute the prediction error $D_i = Y_i - \hat{Y}_i$, where $\hat{Y}_i = \hat{m}_{n/2}(X_i)$.

4. Construct the prediction interval with coverage probability $1-\alpha$ for Y as $[\hat{Y} - D_{[n/2,\alpha/2]}, \hat{Y} + D_{[n/2,\alpha/2]}]$, where $D_{[n/2,\eta]}$ is the η quantile of the empirical distribution of $D_1, ..., D_{n/2}$.

Although SC method generates reliable prediction intervals, it may not always have good performance, especially when the sample size is small. The SC method uses half of the data to train the regression function which may not always be sufficient. All observations do not get a chance to contribute to the construction of empirical distribution of errors. In this paper, we propose a new method to construct prediction intervals based on k-fold cross validation called k-fold conformal prediction interval. The k-fold conformal prediction interval algorithm includes the following steps:

1. Randomly split $\{1, ..., n\}$ into k equal-sized subsets denoted as L_1, L_2,..., L_k.

2. For each L_j where $j \in \{1, ..., k\}$ do the following:

   (a) Train a regression function from $\{(X_i, Y_i) : i \in \bigcup_{r=1}^{k} L_r,\ r \neq j\}$ to estimate the mean function denoted as $\hat{m}_j(X)$.

   (b) For $i \in L_j$, compute the prediction error $D_i = Y_i - \hat{Y}_i$, where $\hat{Y}_i = \hat{m}_j(X_i)$.

3. Construct the prediction interval with coverage probability $1-\alpha$ for Y as $[\hat{Y} - D_{[n,\alpha/2]}, \hat{Y} + D_{[n,\alpha/2]}]$, where $D_{[n,\eta]}$ is the η quantile of the empirical distribution of $D_1, ..., D_n$.

The proposed k-fold conformal prediction interval requires estimation of k regression functions which results in the empirical distribution of errors based on the all training data observations. Thus, the proposed method's computational cost is on the order of k times that of the SC method. Each training set is larger too, so there could be added expense.

2.4 Simulation Study

To evaluate the finite-sample performance of the proposed approach, we conducted a simulation study to compare our proposed method to the SC method with respect to coverage rate and interval width performance measures. Data are simulated from an additive error model: $Y = m(X) + \epsilon$, where $X = (X_1, ..., X_p)$ with $p = 10$ and $X \sim \mathcal{N}(0, \Sigma_p)$, where Σ_p is an AR(1) covariance matrix with $\rho = 0.6$ and diagonal values equal to 1. We considered three factors, namely the distribution of the error terms, the choice of mean function $m()$, and the number of training observations n. Following (Zhang et al., 2019), we considered the following factorial design for these three factors:

- **Mean functions**:
  1. linear: $m_1(x) = x_1 + x_2$
  2. nonlinear: $m_2(x) = 2\exp(-|x_1| - |x_2|))$
  3. nonlinear with interaction: $m_3(x) = 2\exp(-|x_1| - |x_2|)) + x_1x_2$
- **Distributions of errors**:
  1. homoscedastic: $\epsilon \sim \mathcal{N}(0, 1)$
  2. heavy-tailed: $\epsilon \sim \frac{t_3}{\sqrt{3}}$, where t_3 is a t-distribution with 3 degrees of freedom.
  3. heteroscedastic: $\epsilon \sim \mathcal{N}(0, \frac{1}{2} + \frac{1}{2}\frac{|m(X)|}{E|m(X)|})$
- **Training Sample sizes**: $n = 500, 2500$, and 5000

The full-factorial design has 27 different simulation scenarios. The following hyperparameter were used to train the neural network model. The neural network model has 2 fully connected layers with 15 neurons in each layer. We investigated different activation functions, such as ReLU and tanh, and found that ReLU had the best overall performance. Only results for ReLU are reported here. All weights were initialized with the Xavier method (Glorot and Bengio, 2010). We used stochastic gradient descent (SGD) with a mini-batch size of 32. The Adam optimizer (Kingma

and Ba, 2014) with learning rate of 0.03% was used to minimize the loss function. The model was trained for 20,000 iterations.

2.4.1 Evaluation of Coverage Rates and Interval Widths

The nominal coverage level was set at 0.9 for the construction of all prediction intervals in the simulation study. 50 datasets were simulated for each simulation scenario. We also generated 500 test samples independently from the joint distribution of (X, Y) for each simulation scenario. We defined the coverage rate as the percentage of response values contained in their corresponding prediction intervals for the test data. We estimated the coverage rate by mean of coverage rates obtained from 50 simulated datasets for each simulation scenario. To evaluate the effect of k on the performance of prediction intervals, we considered three different k values, namely 2, 5 and 10. Figure 2.1 compares the coverage rate estimates of the SC method, 2-fold prediction interval method, 5-fold prediction interval method, and 10-fold prediction interval method. The white circle in each boxplot is the average of the 50 coverage rate estimates for each simulation scenario. The estimates of the coverage rates for k5 and k10 are closer to 0.9 (the nominal level) especially for the larger sample sizes compared to the SC and k2 methods. The SC and k2 methods tend to over-cover the response values based on coverage rates especially when the sample is small because these methods use only half of the training data to train the mean function which may be insufficient when n is small. Thus, the prediction errors would be larger resulting in a wider prediction intervals and over-coverage. The SC method uses only half of the training data to find the prediction errors for the other half of the training data which makes it similar to the k2 method except that k2 method finds prediction errors for all training data. As such, SC and k2 methods have very similar boxplots for the coverage rates. As the sample size increases, the coverage estimates of all methods become more concentrated around the nominal level due to having adequate information to estimate the mean functions using neural networks. The results suggest that all competing methods showed stable

behavior in terms of the coverage rate estimates across all factors including the mean functions and the choice of error distributions.

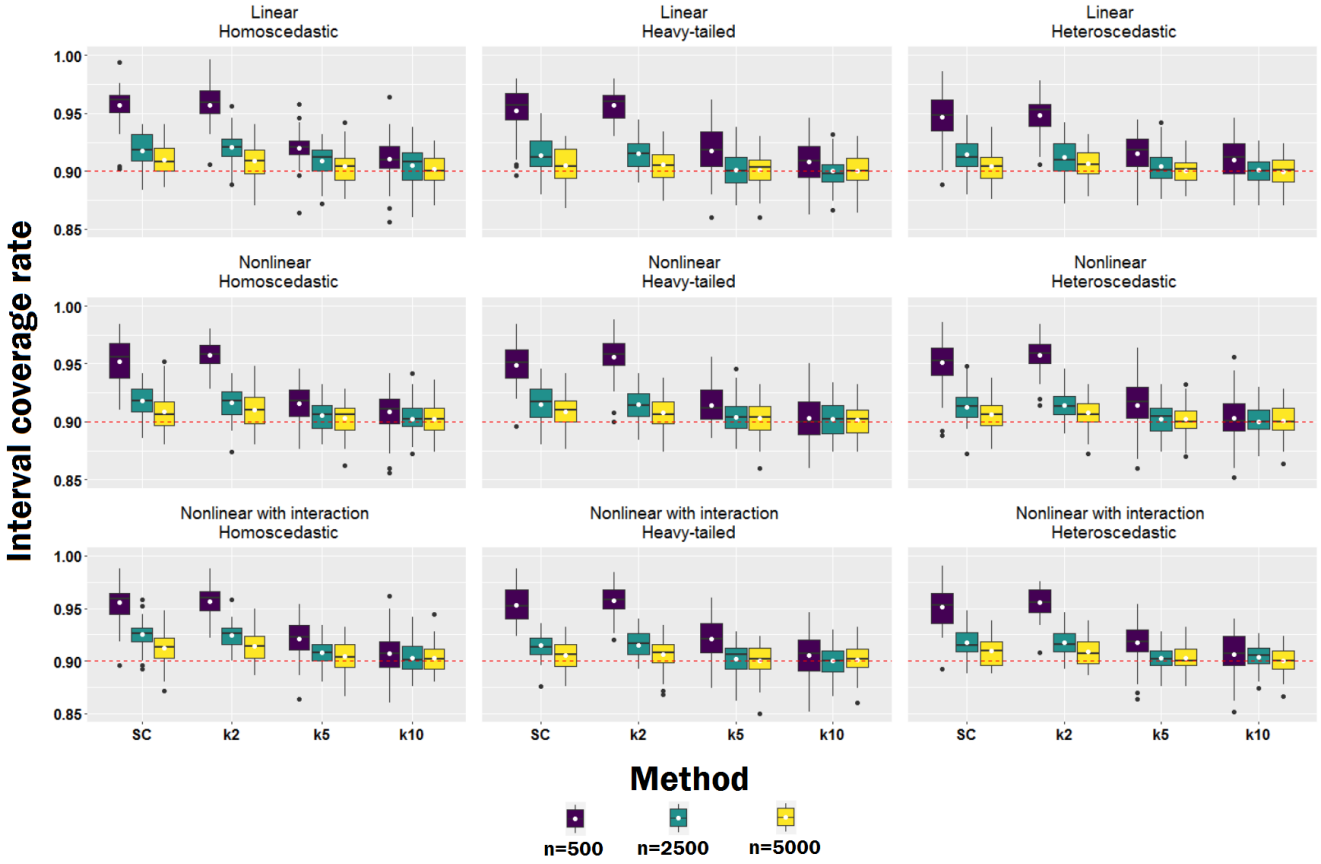

Figure 2.1: Boxplots of the coverage rate estimates of the split conformal method (SC), 2-fold prediction interval method (k2), 5-fold prediction interval method (k5), and 10-fold prediction interval method (k10). The white circle in each boxplot is the average of the 50 coverage rate estimates for each simulation scenarios. The dashed red lines show the nominal coverage level which is set to be 0.9 in our study.

To evaluate the prediction interval widths, we averaged the 500 test cases' interval widths for each simulated dataset. To better compare the proposed k-fold prediction interval method with the SC method, we computed the ratio of the SC interval width to the k-fold prediction interval width. Figure 2.2 shows the $\log_2$ ratios of the interval widths. As shown in Figure 2.2, interval width decreases as the sample size increases due to availability of enough training data to estimate the mean functions well. The SC and k2 produce intervals of approximately the same width as

indicated by log ratios close to zero. However, the SC method tends to have a slightly smaller interval width, especially when the sample size is small. Results demonstrate that k5 and k10 prediction intervals are smaller than intervals constructed by the SC method. The k5 and k10 prediction interval methods have a comparable performance in terms of prediction interval widths, which indicates that increasing k in the k-fold prediction interval method does not always improve the performance. This indicates an opportunity to choose k to obtain narrow prediction intervals while maintaining low computational costs. The log ratios show that the k5 and k10 methods have the biggest advantages over SC intervals, in terms of width, when the training sample size is small.

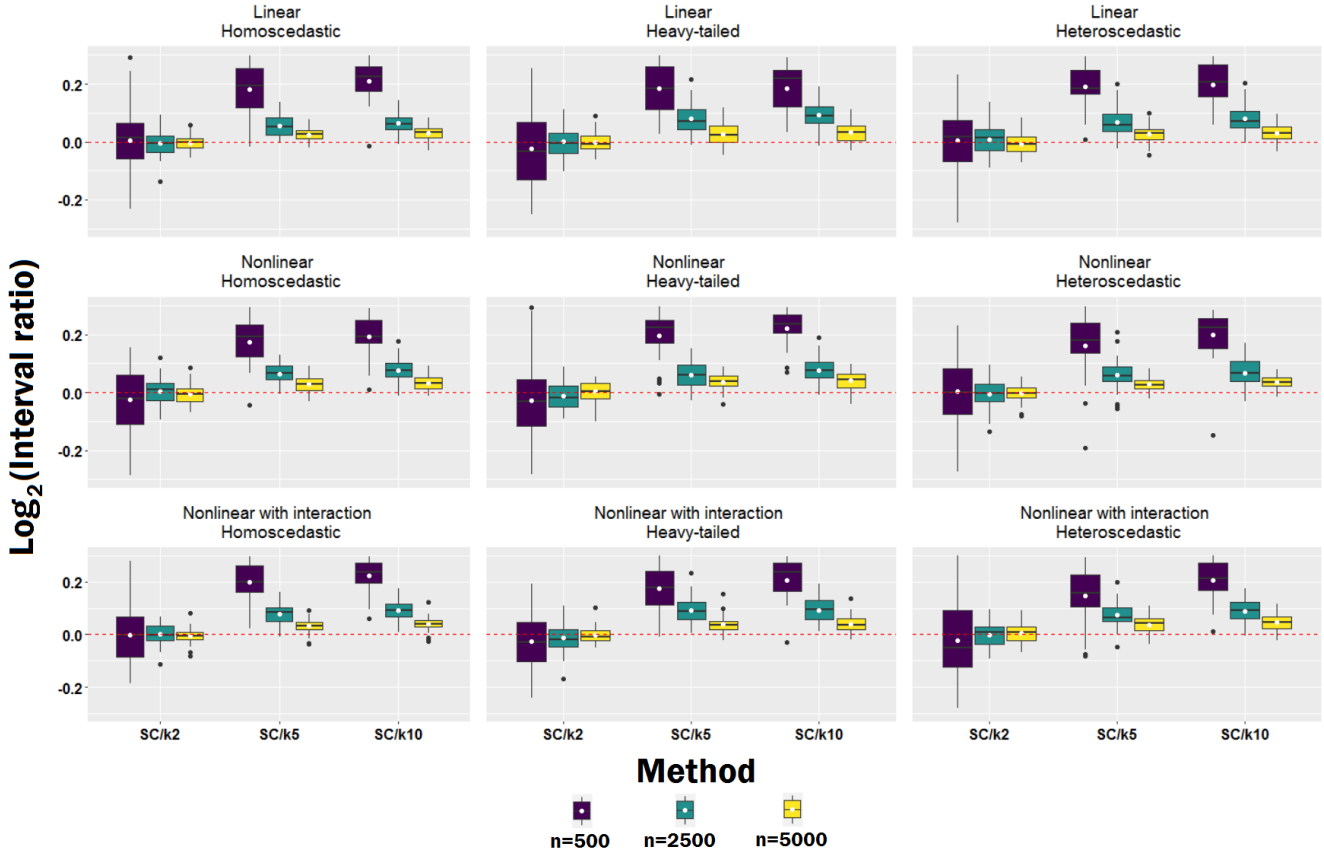

Figure 2.2: Boxplots of the $\log_2$ ratios of split conformal (SC) interval widths to 2-fold prediction interval (k2) widths, 5-fold prediction interval (k5) widths, and 10-fold prediction interval (k10) widths. The white circle in each boxplot is the average of the 50 $\log_2$ interval width ratios for each simulation scenario.

2.5 Data Analysis

To evaluate the performance of our proposed prediction interval method on real-world datasets, we selected 10 datasets from UC Irvine Machine Learning Repository website which are summarized in the Table 2.1.

No.	Name of dataset	Number of predictors	Number of observations
1	Power Plant	4	9,568
2	Facebook Metrics	18	500
3	Parkinsons Telemonitoring	21	5,875
4	Bodyfat	13	252
5	Residential Building	106	372
6	Real Estate Valuation	5	414
7	Wine Quality	11	4898
8	Aquatic Toxicity	8	546
9	Fish Toxicity	6	908
10	Energy Efficiency	8	768

Table 2.1: The summary of real datasets.

To obtain the data analysis results in this section, we used the following hyperparameters for neural networks. The neural networks model has 2 fully connected layers with 10 neurons in each layer. As in the simulation described in the section 3, ReLU activation functions were used and all weights were initialized with Xavier method (Glorot and Bengio, 2010). We used stochastic gradient descent (SGD) with a mini-batch size of 16. The Adam optimizer (Kingma and Ba, 2014) with learning rate of 0.03% was used to minimize the loss function. The model was trained for 25,000 iterations.

To estimate the coverage rates and interval widths, we used 5-fold cross validation which was repeated 20 times for each dataset. Since the simulation study suggested that the SC method tends to perform better compared to the 2-fold prediction interval method (k2), we did not use the k2 method in this section. We employed the 5-fold prediction interval method (k5) rather than the 10-fold prediction interval method (k10) to decrease the computational cost. Figure 2.3 compares the coverage rate estimates of the SC method and the k5 method. The white circle in each boxplot is the average of the 20 coverage estimates that resulted from the cross validation procedure.

As shown in Figure 2.3, the k5 prediction interval has higher average coverage than the SC method except for the Fish dataset. The results also indicate under coverage for some of the datasets. To evaluate the prediction interval widths, we averaged the 20 interval width estimates for each dataset. To better compare the proposed 5-fold prediction interval method with the SC method, we computed the ratio of the SC interval width to the width of the 5-fold prediction interval method. Figure 2.4 shows the $\log_2$ ratios of the interval widths. The results indicate that the k5 method tends to have a smaller interval width than the SC method especially for the datasets with the smaller sample sizes since the $\log_2$ ratios of the average SC interval widths to the average k5 widths are highest when the sample sizes are small. The results also reveal that the k5 method and SC method have a comparable performance for datasets with sufficiently large sample size. The SC method had a smaller interval width compared to the k5 method for the Power Plant dataset, but this was due to under coverage.

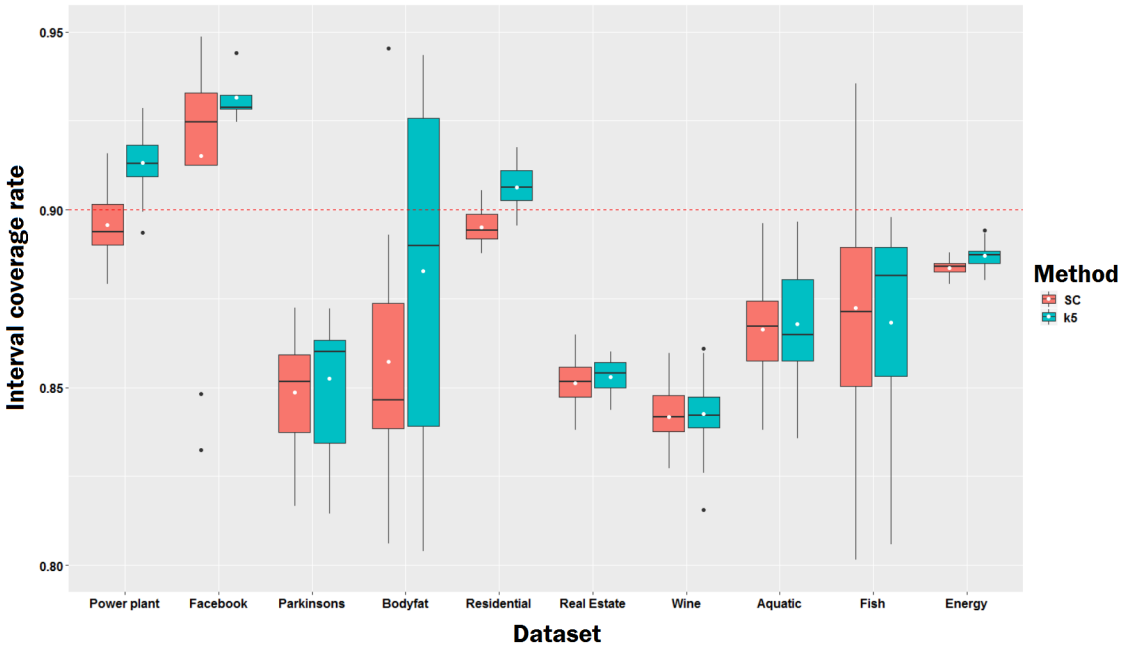

Figure 2.3: Boxplots of the coverage rate estimates of split conformal method (SC) and 5-fold prediction interval method (k5). The white circle in each boxplot is the average of the 20 coverage rate estimates resulting from the cross validation procedure. The dashed red lines show the nominal coverage level which is set to be 0.9 in our study.

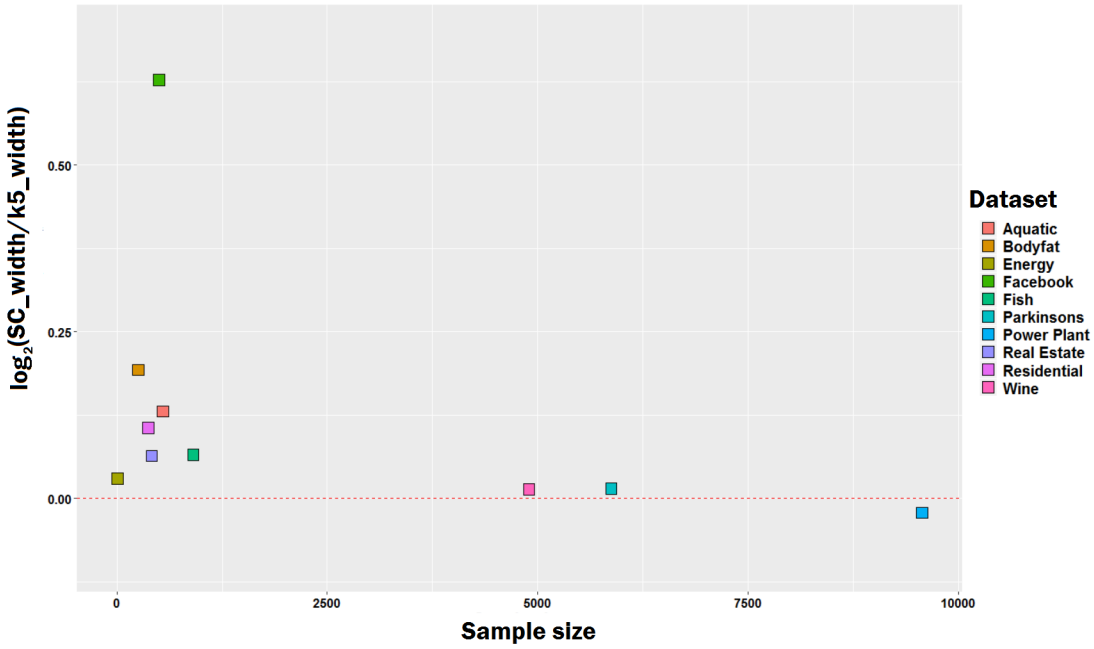

Figure 2.4: Scatter plot of the $\log_2$ ratios of the average split conformal (SC) interval widths to the average 5-fold prediction interval (k5) widths.

2.6 Conclusion

In this paper, we presented a conformal prediction interval method for neural networks using cross validation. The proposed method uses k-fold cross validation on the training data to estimate the empirical error distribution of the errors. Then, the quantiles of the empirical error distribution are used to construct prediction intervals for test data. Our experimental results indicate that the k-fold prediction interval method can efficiently construct prediction intervals for neural networks that compare favorably with intervals produced by the split conformal method. Our results suggest that the proposed method is well-suited for datasets with a small number of observations. We also found that the performance of the proposed model is somewhat sensitive to the choice of k. As such, it is important to tune k in the proposed method to get narrow prediction intervals while maintaining low computational cost.

2.7 References

De Vleaux, R. D., Schumi, J., Schweinsberg, J., and Ungar, L. H. (1998). Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*, volume 1. MIT Press Cambridge.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hwang, J. G. and Ding, A. A. (1997). Prediction intervals for artificial neural networks. *Journal of the American Statistical Association*, 92(438):748–757.

Khosravi, A., Nahavandi, S., Creighton, D., and Atiya, A. F. (2010). Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE transactions on neural networks*, 22(3):337–346.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kivaranovic, D., Johnson, K. D., and Leeb, H. (2019). Adaptive, distribution-free prediction intervals for deep neural networks. *arXiv preprint arXiv:1905.10634*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., and Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111.

Vovk, V., Gammerman, A., and Shafer, G. (2005). *Algorithmic learning in a random world*. Springer Science & Business Media.

Zhang, H., Zimmerman, J., Nettleton, D., and Nordman, D. J. (2019). Random forest prediction intervals. *The American Statistician*, (just-accepted):1–20.