Class Prediction by Prediction Intervals for Neural Nets

Claus Weihs and Malte Jastrow

Abstract Generally, the unknown coefficients of neural nets are estimated by nonlinear least squares. Therefore, prediction intervals for the true value of the target feature exist. The paper proposes to use such intervals for class prediction and model selection. Only in this way, the uncertainty of class predictions can be indicated.

Claus Weihs
TU Dortmund University, D-44221 Dortmund,
✉ claus.weihs@tu-dortmund.de

Malte Jastrow
TU Dortmund University, D-44221 Dortmund,
✉ jastrow@statistik.tu-dortmund.de

Archives of Data Science, Series A
(Online First)
KIT Scientific Publishing
Vol. 6, No. 1, 2020

DOI: 10.5445/KSP/1000098011/03
ISSN 2363-9881

1 Introduction

The consideration of the uncertainty of class probabilities appears to be not that much in use in methods for class prediction. Only a few classification methods like discriminant analysis and logistic regression (the "statistical methods") are constructed to directly estimate class probabilities. Other methods like support vector machines and neural nets (the "machine learning methods") utilize *softmax* transformations to approximate probabilities (Hastie et al, 2001, p. 351, 384). In the cases where such (approximations of) probabilities are used to decide upon the predicted class, most of the time only its maximum is determined (maximum rule) (Hastie et al, 2001, p. 195) and the relative size of the probabilities is ignored. Moreover, the uncertainty of the probability estimates is not even considered. This way, an important statistical aspect of prediction, namely its uncertainty, is neglected.

This paper develops a statistical view on class prediction by utilizing prediction intervals for the estimated class probabilities in neural nets. Since the unknown coefficients of neural nets are estimated by nonlinear least squares, prediction intervals for the true value of the target feature exist. The paper proposes to use such intervals for class prediction and model selection. Only in this way, the uncertainty of class predictions can be indicated.

In Section 2.1, we will introduce the considered neural nets. Parameter estimation is discussed in Section 2.2 and identifiability problems in Section 2.3. Classification formulations of neural nets are introduced in Section 2.4 and prediction intervals for neural nets in Section 3. Model selection in neural nets for classification is defined in Section 4 and an example is given in Section 5. Section 6 concludes the paper.

2 Neural Nets

In this section we will reconsider the basics of the neural nets we will discuss in this paper.

2.1 Single Layer Nets

In this paper, we will concentrate on so-called **single layer neural nets (SLNN)** in order to show properties more easily. In principle, analogue properties are also valid for multi-layer nets or even deep nets. An SLNN is defined as the model (cp. Figure 1).

$$Y = \alpha_0 + \sum_{i=1}^{d} \alpha_i g(\vec{\beta}_i^T \vec{X} + \beta_{i0}) + \varepsilon =: f(\vec{X}; \vec{\theta}) + \varepsilon, \tag{1}$$

with Y the output, $\vec{X} = (X_1, \ldots, X_K)^T$ the vector of inputs, $\vec{\beta}_i^T = (\beta_{i1}, \ldots, \beta_{iK})$ the vector of weights of the inputs for the i-th node of the hidden layer, β_{i0} the constant input weight for the i-th node of the hidden layer, $\vec{\alpha}^T = (\alpha_1, \ldots, \alpha_d)$ the vector of the weights of the outputs of the nodes in the hidden layer, α_0 the bias (overall constant), and ε a random input with expectation 0 (Hwang and Ding, 1997).

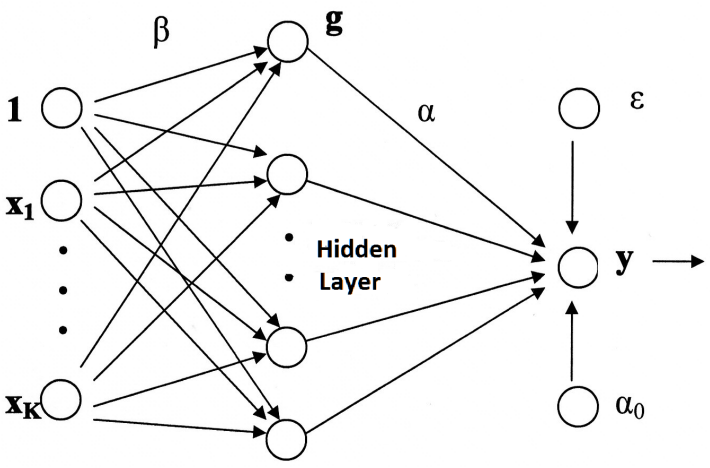

Figure 1: Neural Net with one hidden layer (SLNN)..

The so-called **activation function** $g(x)$ makes the model nonlinear in that it is generally chosen as a continuous symmetric sigmoid approximation to a jump

function ('firing' at a prefixed activation potential):

$$g(x) \to 0 \quad \text{for} \quad x \to -\infty, \tag{2}$$
$$g(x) \to 1 \quad \text{for} \quad x \to \infty, \text{and} \tag{3}$$
$$g(x) + g(-x) = 1. \tag{4}$$

We will concentrate on the *logistic activation function* (also called *softmax* transformation, see Section 2.4) (Figure 2):

$$g(x) = \frac{1}{1 + e^{-x}}. \tag{5}$$

Other possibilities are *symmetric distribution functions*.

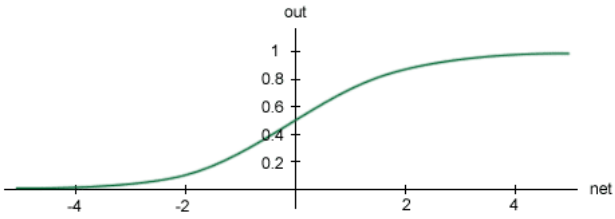

Figure 2: Logistic Activation Function..

2.2 Estimation

Altogether, there are the unknown model coefficients $\vec{\theta} = (\alpha_0, \ldots, \alpha_d, \beta_{10}, \ldots, \beta_{d0}, \vec{\beta}_1^T, \ldots, \vec{\beta}_d^T)^T$. These coefficients $\vec{\theta}$ are estimated via the *nonlinear ordinary least squares method*, minimizing

$$\frac{1}{2} \sum_{j=1}^{n} (f(\vec{x}_j; \vec{\theta}) - y_j)^2 =: \sum_{j=1}^{n} e(\vec{x}_j, \vec{\theta}) \tag{6}$$

with the so-called error function $e(\vec{x}, \vec{\theta})$. The standard estimation method is *backpropagation*, a gradient descent method of the form:

$$\theta_l^m = \theta_l^{m-1} - t^m \frac{\partial e(\vec{x}_j, \vec{\theta}^{m-1})}{\partial \theta_l}, \tag{7}$$

where $\theta_l \in \vec{\theta} = (\alpha_0, \ldots, \alpha_d, \beta_{10}, \ldots, \beta_{d0}, \vec{\beta}_1^T, \ldots, \vec{\beta}_d^T)^T$, m is the iteration index, and $t^m > 0$ the learning rate (step length) in iteration m. Weights are adapted backwards from output to input, motivating the term **backpropagation**. To mimic gradient steepest descent, observations have to be presented repeatedly to the net.

The error function is highly non-linear and non-convex (because of the activation function). Thus, every descent method can only find a *local minimum*. Different starting values may help approaching the global minimum.

For large datasets, backpropagation is more efficient than more advanced (second order) methods like Levenberg-Marquardt, because the inversion of (an approximation to) the Hessian is avoided (cp. Weihs et al (2014, pp. 164, 186, 204)).

2.3 Non-Identifiability

The main problem with neural nets is that model (1) is never identifiable, i.e. that always multiple coefficient vectors lead to the same value of the *model function* $f(\vec{X}; \vec{\theta})$. Hwang and Ding (1997) derived the following results.

Theorem 1 (Non-Identifiability of Neural Nets) Neural Nets with the **logistic activation function** (Equation 5) are never identifiable, since two kinds of transformations of the model coefficients $\vec{\theta}$ leave the model function invariant:

(i) Permutations of the neurons or rather the coefficients $\vec{\mu}_i = (\alpha_i, \beta_{i0}, \vec{\beta}_i^T)^T$.

(ii) Transformations: $(\alpha_0, \vec{\mu}_1, \ldots, \vec{\mu}_i, \ldots, \vec{\mu}_d) \rightarrow (\alpha_0 + \alpha_i, \vec{\mu}_1, \ldots, -\vec{\mu}_i, \ldots, \vec{\mu}_d)$.

However, for the logistic activation function, transformations (i) and (ii) are the only transformations to leave the model function invariant (maximal identifiability) if the model is not reducible:

Definition 1 (Reducibility) A coefficient vector $\vec{\theta} = (\alpha_0, \ldots, \alpha_d, \beta_{10}, \ldots, \beta_{d0}, \vec{\beta}_1^T, \ldots, \vec{\beta}_d^T)^T$ is called *reducible*, if one the following three conditions are met

(a) $\alpha_i = 0$ for any $i = 1, \ldots, d$,

(b) $\vec{\beta}_i = \mathbf{0}$ for any $i = 1, \ldots, d$, where $\mathbf{0}$ is the null vector of length K, or

(c) $(\beta_{i0}, \vec{\beta}_i^T) = \pm(\beta_{j0}, \vec{\beta}_j^T)$ for any $i \neq j$.

Decisive for maximal identifiability is that all neurons in the hidden layer independently contribute to the model function, i.e. the following condition 1 is fulfilled:

Condition 1 The class of functions $\{g(bz + b_0), b > 0\} \cup \{g \equiv 1\}$ is linear independent, i.e. for each $d \in \mathbb{N}$ and scalars $a_0, a_i, b_{i0} \in \mathbb{R}$ and $b_i > 0$, $i = 1, \ldots, d$, with $(b_i, b_{i0}) \neq (b_j, b_{j0})$ for each $i \neq j$:

$$a_0 + \sum_{i=1}^{d} a_i g(b_i z + b_{i0}) = 0 \ \ \forall z \in \mathbb{R} \quad \Rightarrow \quad a_0 = a_1 = \cdots = a_d = 0. \tag{8}$$

Non-identifiability obviously leads to non-interpretability. Despite this non-interpretability of coefficients, however, neural nets can be well used for interval prediction as will be seen in Section 3.

2.4 Neural Nets in Classification

For classification of **2 classes**, you may use $f(\vec{X}; \vec{\theta})$ itself as decision function. As a classification rule you may use: Beyond a threshold τ, predict class 1, else class 0, where τ is optimally adapted to the special classification problem.

For **more than 2 classes**, each class is assigned an individual output. The corresponding values of the model function $f_c(\vec{X}; \vec{\theta}_c)$, $c = 0, \ldots, G-1$, are then typically transformed to represent / approximate the probability of class c discriminated from all other classes by the *softmax* transformation

$$s_c(\vec{X}; \vec{\theta}_c) = \frac{1}{1 + \exp(-f(\vec{X}; \vec{\theta}_c))}. \tag{9}$$

This transformation corresponds to another application of the activation function (Fritsch et al, 2019)! As a classification rule you may use: Predict class $\text{argmax}_c \ s_c(\vec{X}; \vec{\theta}_c)$.

3 Prediction Intervals for Neural Nets

Up to now, we ignored the stochastic structure of model (1). Now, we will construct prediction intervals for the model values of neural nets. For classification, this is equivalent to the construction of prediction intervals for class probabilities. For the construction of prediction intervals, we need the invertibility of the covariance matrix of the model coefficients. This way, (an approximation to) the Hessian is re-appearing even for backpropagation. For ease of notation, we restrict ourselves to the 2-class case with no additional transformation of the single output.

Theorem 2 (Prediction interval for nonlinear models with additive error (cp. Bianchi and Calzolari (1980); Hwang and Ding (1997)).) The $\alpha\%-$**prediction interval** of a response Y for values $x_0 := (x_{01} \dots x_{0K})^T$ of the inputs $X_1, \dots, X_K$ for the nonlinear model $Y = f(X_1, \dots, X_K; \beta_1, \dots, \beta_L) + \varepsilon$ is given by:

$$\begin{bmatrix} \hat{Y}(x_0) - t_{n-L;(1+\alpha)/2}\hat{\sigma}\sqrt{1 + \hat{J}_0^T (n\hat{B})^{-1}\hat{J}_0}, \\ \hat{Y}(x_0) + t_{n-L;(1+\alpha)/2}\hat{\sigma}\sqrt{1 + \hat{J}_0^T (n\hat{B})^{-1}\hat{J}_0}, \end{bmatrix}^T \tag{10}$$

if $\hat{B}$ is invertible, the gradient vector $J_0 := \frac{\partial f}{\partial \vec{\beta}}(x_0; \vec{\beta}; 0)^T$ of f is consistently estimated by $\hat{J}_0 := \frac{\partial f}{\partial \vec{\beta}}(x_0; \hat{\vec{\beta}}; 0)^T$, the asymptotic covariance matrix of the nonlinear least-squares estimator is given by $\sigma^2 B^{-1}$ with

$$B := \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \frac{\partial f}{\partial \vec{\beta}}(x_i; \vec{\beta}; 0)^T \frac{\partial f}{\partial \vec{\beta}}(x_i; \vec{\beta}; 0), \tag{11}$$

if the limit exists and is invertible, and B is consistently estimated by

$$\hat{B} := \frac{1}{n} \sum_{i=1}^{n} \frac{\partial f}{\partial \vec{\beta}}(x_i; \hat{\vec{\beta}}; 0)^T \frac{\partial f}{\partial \vec{\beta}}(x_i; \hat{\vec{\beta}}; 0), \tag{12}$$

where $t_{n-L;(1+\alpha)/2}$ is the $(1+\alpha)/2$-quantile of the t-distribution with $n - L$ degrees of freedom and $\hat{\sigma}^2 := \frac{1}{n-L} \sum_{i=1}^{n} \hat{\varepsilon}_i^2$.

Nonlinear least squares estimators are asymptotically normal under weak conditions with asymptotic covariance matrix $\sigma^2 B^{-1}$, consistently estimated by

$\sigma^2 \hat{B}^{-1}$. Under condition 2 for neural nets, defined below, B is invertible. The logistic activation function meets condition 2. Thus, $\hat{B}$ is asymptotically invertible for the logistic activation function.

Condition 2 (Extended identifiability condition (Hwang and Ding, 1997):)
Let g be differentiable and g' the first derivative. The class of functions

$$\{g(bz+b_0), b>0\} \cup \{g'(bz+b_0), b>0\} \cup \{x\, g'(bz+b_0), b>0\} \cup \{g \equiv 1\} \quad (13)$$

is linearly independent, i.e. for each $d \in \mathbb{N}$ and scalars $a_0, a_i, e_i, f_i, b_{i0} \in \mathbb{R}$ and $b_i > 0$, $i = 1, \ldots, d$, with $(b_i, b_{i0}) \neq (b_j, b_{j0})$ for every $i \neq j$:

$$a_0 + \sum_{i=1}^{d} [a_i g(b_i z+b_{i0}) + e_i g'(b_i z+b_{i0}) + f_i x g'(b_i z+b_{i0})] = 0 \quad \forall z \in \mathbb{R} \quad (14)$$

$$\Rightarrow a_0 = a_1 = \cdots a_d = e_1 = \cdots = e_d = f_1 = \cdots = f_d = 0. \quad (15)$$

Taking all this together, asymptotically, prediction intervals of model values exist for the logistic activation function. And even though coefficients of neural nets are not interpretable, neural nets appear to be theoretically unproblematic for the uncertainty estimation of the predicted class probabilities for the logistic activation function.

Unfortunately, typical activation functions as the logistic function can lead to (numerically nearly) singular Hessians (Saarinen et al, 1993). At least observations far away from a decision border can have very large prediction intervals or even non-existing ones with numerically exact singularities. In order to avoid this, we use the Moore-Penrose inverse instead of the exact inverse.

4 Model Selection

In this paper, model selection means the choice of the size of the hidden layer. We distinguish three different model selection rules:

1. **Maximum Rule** based on point predictions: For each input vector, predict the class with the highest estimated probability: $\text{argmax}_c\ s_c(\vec{X}, \vec{\theta}_c)$ (cp. Section 2.4).

To determine the number of neurons in the hidden layer, maximize the predictive power of the net. As predictive power choose (1 − error rate), i.e. minimize the error rate. Use re-substitution or resampling to determine the error rate.

2. **Prediction Interval Rule 1** based on prediction intervals: For each input vector, compute probabilities and their prediction intervals for each class. For each input vector, predict the class whose upper prediction interval limit is greater or equal 1 (= 100%). If there is no such class or more than one class with this property, predict 'class uncertain' (distinguish the different cases, see Section 5!).
   To determine the number of neurons in the hidden layer, use the same procedure as for the maximum rule. Additionally, as a 2^{nd} criterion, minimize the mean length of class prediction intervals for observed inputs.

3. **Prediction Interval Rule 2** based on prediction intervals: The only difference to Prediction Interval Rule 1 is that for each input vector that class is predicted whose lower prediction interval limit is greater than the posteriors (predicted probabilities) of the other classes. If there is no such class with this property, predict 'class uncertain'.

5 Example

As a demonstration example, we use $n = 52$ observations of $p = 2$ inputs, motivated by height x and weight y of a person. We distinguish three classes: $y < 70$, $70 \leq y \leq 80$, $y > 80$. Note that the presumed class borders are horizontal, x is not involved in class building (cp. Figure 3). We applied the R routine (R Core Team, 2018) 'neuralnet' (Fritsch et al, 2019) utilizing standard backpropagation.

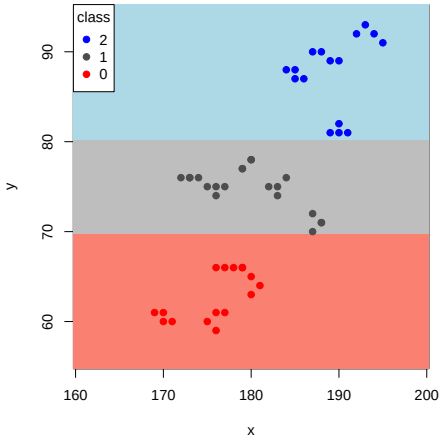

Figure 3: Example data with class partition.

We applied the Maximum Rule and the Prediction Interval Rules to decide between 1, 2, 3 neurons in the hidden layer. To approach the globally best neural net models via nonlinear ordinary least squares, models are fit with 10 different starting values. The model with the lowest training error according to Interval Rule 1 is chosen. The same procedure is used to compute leave-one-out error rates. In some rare cases, for some leave-one-out folds all 10 starting values do not lead to convergence and therefore produce no model. In those cases, another 10 starting values are taken into account to find a model.

We predicted the unknown class on the grid $160 \leq x \leq 200$, $55 \leq y \leq 95$. The predicted class is indicated as background color. Note that for Interval Rule 1 we distinguish three (!) different in-between 'uncertain' cases ("classes 2 and 1" possible, "classes 1 and 0" possible, "none of the classes" identified). 'Uncertain cases' count as error.

First note that the found class borders never come near to horizontal. For 1 hidden neuron, the Maximum Rule assignment is always best, whereas for the Interval Rules assignment is often uncertain though borders are similar, but more ragged (Figure 4). For 2 hidden neurons, borders of the Maximum Rule and the Interval Rules are again similar, but the lower border has changed direction, and assignment of the Interval Rules is much less often uncertain (Figure 5). For 3 hidden neurons, Interval Rule 2 identifies problematic areas best, but Interval Rule 1 additionally identifies regions, where two classes are possible (Figure 6).

Training error is always 0 for the Maximum Rule, for the Interval Rules it is acceptable from 2 hidden neurons on (Table 1). Leave-one-out error rates are only somewhat greater than training errors. The mean prediction interval length is very small for 2 and 3 neurons. Therefore, the Maximum Rule might even get along with 1 hidden neuron, whereas the Interval Rules might choose 2 hidden neurons. Concerning the question which Prediction Interval Rule to be used, we can say that only Interval Rule 1 is able to identify regions where two classes are possible, whereas Interval Rule 2 gives lower error rates and identifies uncertain regions best.

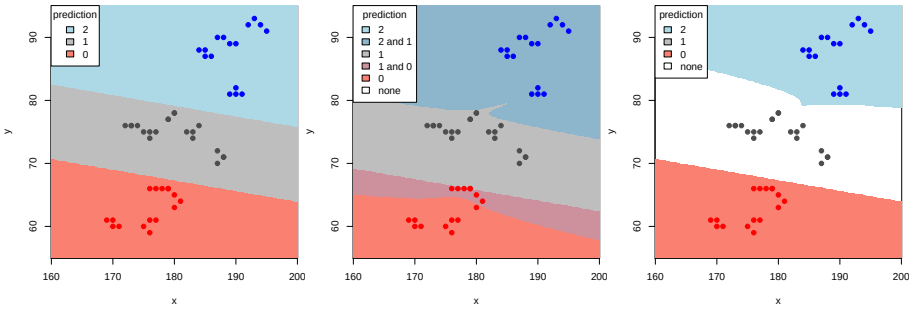

Figure 4: 1 hidden neuron: Maximum Rule (left) vs. Prediction Interval Rules 1 (middle) and 2 (right).

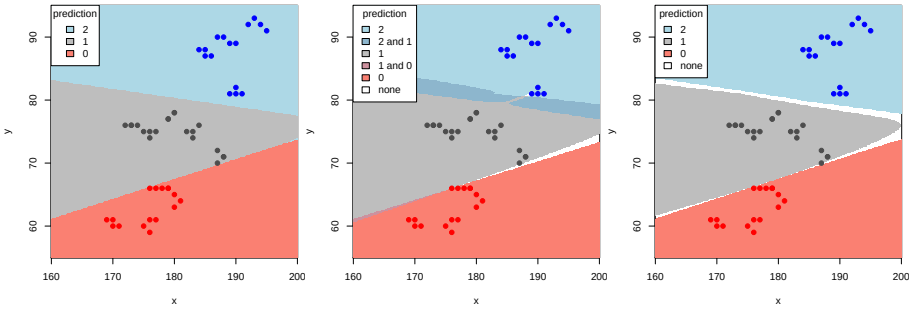

Figure 5: 2 hidden neurons: Results of Maximum Rule and Prediction Interval Rules 1 and 2.

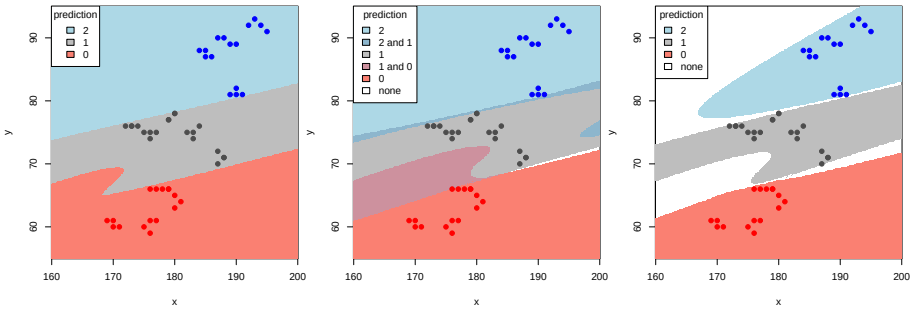

Figure 6: 3 hidden neurons: Results of Maximum Rule and Prediction Interval Rules 1 and 2.

Table 1: Model Selection: Error Rates (in Percent) and Mean Prediction Interval Lengths.

Hidden	Training error			Leave-one-out error			Mean prediction
Neurons	Max.	Interval 1	Interval 2	Max.	Interval 1	Interval 2	Interval length
1	**0**	44.2	38.5	**5.8**	46.2	44.2	0.662
2	**0**	5.8	**0**	**0**	9.6	3.8	0.002
3	**0**	5.8	**0**	5.8	9.6	5.8	0.006

6 Discussion

In order to construct a reliable prediction rule, we propose to utilize the stochastic model's properties to create prediction intervals for assessing prediction quality. We propose to allow for 'class uncertain' as a possible prediction outcome. Only this way, the uncertainty of class predictions can be indicated. For prediction quality, one should rely on a variety of quality measures, e.g. on training error vs. leave-one-out error. The discussion which prediction rule should be used might have to be continued.

Acknowledgements We thank an unknown reviewer for proposing Prediction Interval Rule 2.

References

Bianchi C, Calzolari G (1980) The One-Period Forecast Errors in Nonlinear Econometric Models. International Economic Review 21(1):201–208. DOI: 10.2307/2526248.

Fritsch S, Guenther F, Wright M (2019) neuralnet: Training of Neural Networks. R package version 1.44.2. URL: `https://CRAN.R-project.org/package=neuralnet`.

Hastie T, Tibshirani R, Friedman J (2001) The Elements of Statistical Learning. Springer, New York. DOI: 10.1007/978-0-387-84858-7.

Hwang J, Ding A (1997) Prediction intervals for artificial neural networks. Journal of the American Statistical Association 92(438):748–757. DOI: 10.1080/01621459.1997.10474027.

R Core Team (2018) R: A language and environment for statistical computing. URL: `https://www.R-project.org/`.

Saarinen S, Bramley R, Cybenko G (1993) Ill-conditioning in Neural Network Training Problems. SIAM Journal on Scientific Computing 14(3):693–714. DOI: 10.1137/0914044.

Weihs C, Mersmann O, Ligges U (2014) Foundations of statistical algorithms: with references to R packages. Chapman & Hall. DOI: 10.1111/anzs.12233.