Feedforward neural networks as statistical models

Andrew McInerney[1,]

@amcinerney_

andrew.mcinerney@ul.ie

Kevin Burke[1]

[1] *Department of Mathematics & Statistics, University of Limerick*

Introduction

In recent years, neural networks have experienced great success in the prediction of complex problems (LeCun et al., 2015). However, while neural networks exhibit strong predictive performance, they are viewed as "black-box" algorithms, i.e., their predictions are not easily understood and difficult to interpret. However, feedforward neural networks (FNNs) can be viewed through a statistical lens and seen as an alternative statistical model for non-linear regression. Thus, we aim to leverage the inherent intelligibility present in statistical modelling and demonstrate the inferential capabilities of FNNs, highlighting how these models can be used for problems beyond pure prediction". The testing of the irrelevant-input-node hypothesis can inform us whether the effect of a given covariate on the response is significantly different from zero (White, 1989). Combining this with covariate-effect plots and their associated uncertainty, the outputs of FNNs become more akin to classical statistical models.

Hypothesis Testing

Hypothesis tests can be used to determine the statistical significance of individual parameters, or more appropriately for neural networks, they can be used to determine the statistical significance of groups of parameters. As each input node has multiple weights associated with it, we can make use of the multiple-parameter Wald test to test a single hypothesis on each of these parameters, i.e., test the overall relevance of covariate x_j by testing $H_0 : \omega_j = 0_q$, where $\omega_j = (\omega_{j1}, \omega_{j2}, \ldots, \omega_{jq})^T$ is the vector of weights that connects input node j to the hidden layer and 0_q is a zero vector of length q. Using the fact that (asymptotically) $\hat{\omega}_j \sim N(\omega_j, \Sigma_{\hat{\omega}_j})$, where $\Sigma_{\hat{\omega}_j}$ is the relevant $q \times q$ sub-matrix of the variance-covariance matrix, $\hat{\Sigma}$ (which is the inverse of the observed information matrix). Then, we have that

$$(\hat{\omega}_j - \omega_j)^T \Sigma_{\hat{\omega}_j}^{-1} (\hat{\omega}_j - \omega_j) \sim \chi^2_q$$

from which a p-value can be obtained by setting $\omega_j = 0_q$ and comparing this statistic to the χ^2_q distribution. However, the estimation of $\hat{\Sigma}$ can be problematic due to the issue of parameter redundancy in FNNs, which leads to unidentifiability in some of the parameters.

Covariate Effects

When modelling the relationship between a covariate and a response, there are two natural questions to ask. First, is there any relationship? This is covered by the significance testing presented above. Second, if there is a relationship, what is the nature of this relationship? To this end, we consider a graphical approach to understanding the potentially complex covariate effects that are captured by the neural network model. A common approach to assess the relationship between a covariate and the response is using partial dependence plots (Friedman, 2001). The partial dependence" of the response on x_j can be estimated from the data using

$$\overline{\text{NN}}_j(x) = \frac{1}{n} \sum_{i=1}^{n} \text{NN}(x_{(i,1)}, \ldots, x_{(i,j-1)}, x, x_{(i,j+1)}, \ldots, x_{(i,p)}),$$

where $x_{(i,j)}$ is the value of the jth covariate for the ith individual. Equation ??? can be computed for a set of x values, and the pairs of points, $(x, \overline{\text{NN}}_j(x))$, can be used to construct a plot.

However, while the partial dependence plot provides the change in the average predicted response value as x_j varies, it is also useful to consider the in the average predicted response for a d-unit increase in x_j. This then plays the same role as a regression coefficient obtained from classical statistical models. Thus, we define the effect of a d-unit increase in x_j on the response as

$$\hat{\beta}_j(x, d) = \overline{\text{NN}}_j(x + d) - \overline{\text{NN}}_j(x)$$

where d is often set to one or the standard deviation of x_j. Again, the $(x, \hat{\beta}_j(x, d))$ pairs can be used to construct a plot, which we term a Partial Covariate-Effect plot (PCE).

As the weights of the neural network are estimated using maximum likelihood, there are a number of methods available to estimate the associated uncertainty of these functions, e.g., the delta method and bootstrapping.

Results

Here you may have some figures to show off, bellow I have made a scatterplot with the infamous Iris dataset and I can even reference to the figure automatically like this, `Figure \@ref(fig:irisfigure)`, Figure 1.

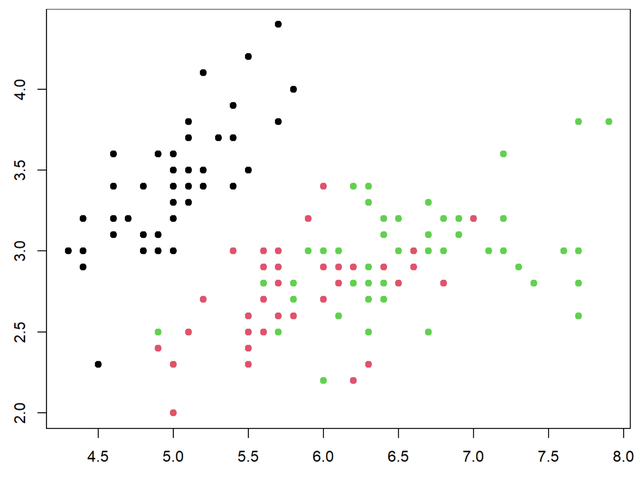

Figure 1: Here is a caption for the figure. This can be added by using the "fig.cap" option in the r code chunk options, see this link from the legend himself, Yihui Xie.

Maybe you want to show off some of that fancy code you spent so much time on to make that figure, well you can do that too! Just use the `echo=TRUE` option in the r code chunk options, Figure 2!

```
#trim whitespace
par(mar=c(2,2,0,0))
#plot boxplots
boxplot(iris$Sepal.Width~iris$Species,
        col = "#008080",
        border = "#0b4545",
        ylab = "Sepal Width (cm)",
        xlab = "Species")
```

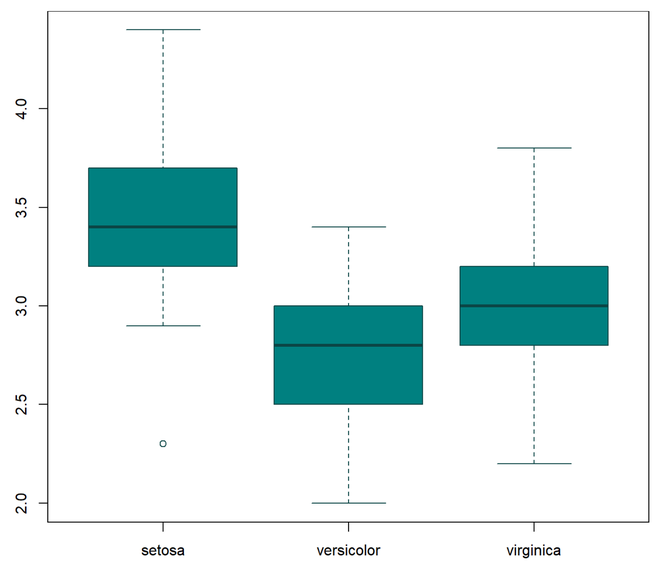

Figure 2: Boxplots, so hot right now!

How about a neat table of data? See, Table 1:

*Table 1: A table made with the **knitr::kable** function.*

Sepal Length	Sepal Width	Petal Length	Petal Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa

References