**Variable importance in neural networks**

If you’re a regular reader of my blog you’ll know that I’ve spent some time dabbling with neural networks. As I explained [here](https://beckmw.wordpress.com/2013/03/04/visualizing-neural-networks-from-the-nnet-package/), I’ve used neural networks in my own research to develop inference into causation. Neural networks fall under two general categories that describe their intended use. Supervised neural networks (e.g., multilayer feed-forward networks) are generally used for prediction, whereas unsupervised networks (e.g., Kohonen self-organizing maps) are used for pattern recognition. This categorization partially describes the role of the analyst during model development. For example, a supervised network is developed from a set of known variables and the end goal of the model is to match the predicted output values with the observed via ‘supervision’ of the training process by the analyst. Development of unsupervised networks are conducted independently from the analyst in the sense that the end product is not known and no direct supervision is required to ensure expected or known results are obtained. Although my research objectives were not concerned specifically with prediction, I’ve focused entirely on supervised networks given the number of tools that have been developed to gain insight into causation. Most of these tools have been described in the primary literature but are not available in R.

My [previous post](https://beckmw.wordpress.com/2013/03/04/visualizing-neural-networks-from-the-nnet-package/) on neural networks described a plotting function that can be used to visually interpret a neural network. Variables in the layers are labelled, in addition to coloring and thickening of weights between the layers. A general goal of statistical modelling is to identify the relative importance of explanatory variables for their relation to one or more response variables. The plotting function is used to portray the neural network in this manner, or more specifically, it plots the neural network as a neural interpretation diagram (NID)[1](https://beckmw.wordpress.com/2013/08/12/variable-importance-in-neural-networks/#ref1). The rationale for use of an NID is to provide insight into variable importance by visually examining the weights between the layers. For example, input (explanatory) variables that have strong positive associations with response variables are expected to have many thick black connections between the layers. This qualitative interpretation can be very challenging for large models, particularly if the sign of the weights switches after passing the hidden layer. I have found the NID to be quite useless for anything but the simplest models.

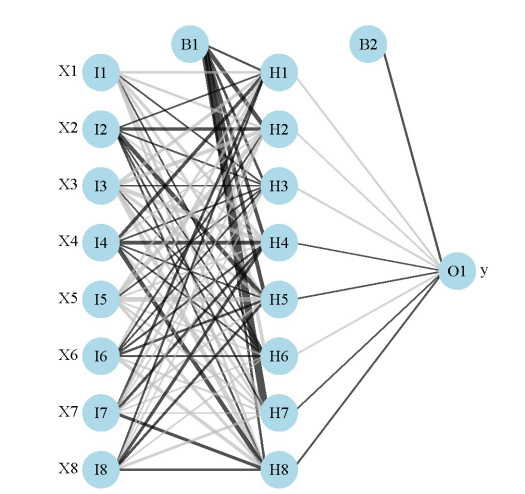


Fig: A neural interpretation diagram for a generic neural network. Weights are color-coded by sign (black +, grey -) and thickness is in proportion to magnitude. The plot function can be obtained [here](https://gist.github.com/fawda123/5086859/raw/17fd6d2adec4dbcf5ce750cbd1f3e0f4be9d8b19/nnet_plot_fun.r).

The weights that connect variables in a neural network are partially analogous to parameter coefficients in a standard regression model and can be used to describe relationships between variables. That is, the weights dictate the relative influence of information that is processed in the network such that input variables that are not relevant in their correlation with a response variable are suppressed by the weights. The opposite effect is seen for weights assigned to explanatory variables that have strong, positive associations with a response variable. An obvious difference between a neural network and a regression model is that the number of weights is excessive in the former case. This characteristic is advantageous in that it makes neural networks very flexible for modeling non-linear functions with multiple interactions, although interpretation of the effects of specific variables is of course challenging.

A method proposed by Garson 1991[2](https://beckmw.wordpress.com/2013/08/12/variable-importance-in-neural-networks/#ref2) (also Goh 1995[3](https://beckmw.wordpress.com/2013/08/12/variable-importance-in-neural-networks/#ref3)) identifies the relative importance of explanatory variables for specific response variables in a supervised neural network by deconstructing the model weights. The basic idea is that the relative importance (or strength of association) of a specific explanatory variable for a specific response variable can be determined by identifying all weighted connections between the nodes of interest. That is, all weights connecting the specific input node that pass through the hidden layer to the specific response variable are identified. This is repeated for all other explanatory variables until the analyst has a list of all weights that are specific to each input variable. The connections are tallied for each input node and scaled relative to all other inputs. A single value is obtained for each explanatory variable that describes the relationship with response variable in the model (see the appendix in Goh 1995 for a more detailed description). The original algorithm presented in Garson 1991 indicated relative importance as the absolute magnitude from zero to one such the direction of the response could not be determined. I modified the approach to preserve the sign, as you’ll see below.

We start by creating a neural network model (using the [nnet](http://cran.r-project.org/web/packages/nnet/index.html) package) from simulated data before illustrating use of the algorithm. The model is created from eight input variables, one response variable, 10000 observations, and an arbitrary correlation matrix that describes relationships between the explanatory variables. A set of randomly chosen parameters describe the relationship of the response variable with the explanatory variables.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | require(clusterGeneration)  require(nnet)    #define number of variables and observations  set.seed(2)  num.vars<-8  num.obs<-10000    #define correlation matrix for explanatory variables  #define actual parameter values  cov.mat<-genPositiveDefMat(num.vars,covMethod=c("unifcorrmat"))$Sigma  rand.vars<-mvrnorm(num.obs,rep(0,num.vars),Sigma=cov.mat)  parms<-runif(num.vars,-10,10)  y<-rand.vars %\*% matrix(parms) + rnorm(num.obs,sd=20)    #prep data and create neural network  y<-data.frame((y-min(y))/(max(y)-min(y)))  names(y)<-'y'  rand.vars<-data.frame(rand.vars)  mod1<-nnet(rand.vars,y,size=8,linout=T) |

The function for determining relative importance is called gar.fun and can be imported from [my Github account](https://gist.github.com/fawda123) (gist 6206737). The function reverse depends on the plot.nnet function to get the model weights.

|  |  |
| --- | --- |
| 1  2 | #import 'gar.fun' from Github  source\_url('<https://gist.githubusercontent.com/fawda123/6206737/raw/d6f365c283a8cae23fb20892dc223bc5764d50c7/gar_fun.r>') |

The function is very simple to implement and has the following arguments:

|  |  |
| --- | --- |
| out.var | character string indicating name of response variable in the neural network object to be evaluated, only one input is allowed for models with multivariate response |
| mod.in | model object for input created from nnet function |
| bar.plot | logical value indicating if a figure is also created in the output, default T |
| x.names | character string indicating alternative names to be used for explanatory variables in the figure, default is taken from mod.in |
| ... | additional arguments passed to the bar plot function |

The function returns a list with three elements, the most important of which is the last element named rel.imp. This element indicates the relative importance of each input variable for the named response variable as a value from -1 to 1. From these data, we can get an idea of what the neural network is telling us about the specific importance of each explanatory for the response variable. Here’s the function in action:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | #create a pretty color vector for the bar plot  cols<-colorRampPalette(c('lightgreen','lightblue'))(num.vars)    #use the function on the model created above  par(mar=c(3,4,1,1),family='serif')  gar.fun('y',mod1,col=cols,ylab='Rel. importance',ylim=c(-1,1))    #output of the third element looks like this  # $rel.imp  #         X1         X2         X3         X4         X5  #  0.0000000  0.9299522  0.6114887 -0.9699019 -1.0000000  #         X6         X7         X8  # -0.8217887  0.3600374  0.4018899 |

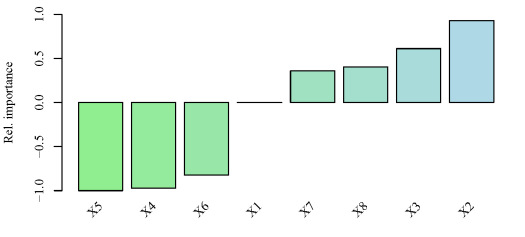
[](https://beckmw.files.wordpress.com/2013/08/relimp.pdf)

Fig: Relative importance of the eight explanatory variables for response variable y using the neural network created above. Relative importance was determined using methods in Garson 1991[2](https://beckmw.wordpress.com/2013/08/12/variable-importance-in-neural-networks/#ref2) and Goh 1995[3](https://beckmw.wordpress.com/2013/08/12/variable-importance-in-neural-networks/#ref3). The function can be obtained [here](https://gist.githubusercontent.com/fawda123/6206737/raw/d6f365c283a8cae23fb20892dc223bc5764d50c7/gar_fun.r).

The output from the function and the bar plot tells us that the variables X5 and X2have the strongest negative and positive relationships, respectively, with the response variable. Similarly, variables that have relative importance close to zero, such as X1, do not have any substantial importance for y. Note that these values indicate relative importance such that a variable with a value of zero will most likely have some marginal effect on the response variable, but its effect is irrelevant in the context of the other explanatory variables.