**INTRODUCTION TO NEURAL NETWORKS**

by Simon Moss

|  |
| --- |
| **Introduction** |

 Often, you want to predict some outcome from a series of predictors. The following table presents some examples

|  |  |
| --- | --- |
| Outcome to predict | Information you could use to predict the outcome |
| Which PhD or Masters by Research candidates are likely to complete their thesis | * grade point average
* number of research papers
* English language ability—called an IELTS score
* personality
 |
| Which PhD or Masters by Research candidates are likely to experience mental health problems | * workload
* household income
* marital status
* grade point average
 |

**Regression analysis**

Traditionally, in the field of statistics, the most common method that researchers utilize to achieve this goal is regression analysis. Specifically, this technique generates an equation that researchers can utilize to predict an outcome. The following box presents an example of an equation that regression analysis might generate

|  |
| --- |
| Likelihood of completion = .03 x GPA + .17 x no. of research paper + .03 x IELTS score |

For example, if someone arrives with a 4.5 GPA, 2 research papers, and a 6.5 IELTS score—a measure of proficiency in English

* Likelihood of completion = .03 x 4.5 + .17 x 2 + .03 x 6.4 = .409
* So, the likelihood this person will complete is about .41 or 41%

**Problems with regression**

 Unfortunately, in many circumstances, these simple equations are not sufficient to explain the association between the predictors and outcome. For example

* perhaps GPA is strongly related to completion, but only when number of papers is below 2 and IELTS is above 6.5 but below 7.5
* perhaps number of papers is strongly related to completion, but only when GPA is between 4 and 5
* hence, the association between the predictors and outcomes might depend on the precise value of other predictors

Even sophisticated variants of regression analysis—including non-linear regression, linear mixed models, multinomial logistic regression, or even generalized estimating equations—disregard these complications. The inclusion of interaction terms, sometimes called moderated regression, is helpful but not usually sufficient to accommodate all the ways in which the values on one predictor may affect the impact of other predictors.

**Role of neural networks**

 Instead, another branch of analysis, called machine learning, is preferable in this instance. Machine learning includes a range of techniques such as decision trees, random forests, and neural networks. This document offers some insight into how to implement neural networks. To begin this discussion, consider the following diagram. Each circle is called a **node**, and each arrow is called a **connection**. Each node is like a neuron—a brain cell—that can be activated at various levels.



 To illustrate how these nodes and connections work, suppose that a PhD candidate arrives with a GPA of 4.3, a track record of 3 papers, and an IELTS score of 6.5. This information is entered into the first set of circles or nodes, called the input layer. For example

* the top node in this layer is activated 4.3 units, representing a GPA of 4.3
* the middle node in this layer is activated 3.0 units, representing 3 papers, and
* the bottom node in this later is activated 6.5 units, representing an IELTS of 6.5

This activation then travels along the arrows or connections to the next set of nodes, called the hidden layer. However, this activation is multiplied by the number alongside each arrow or connection, called a weight. For example, the number .32 appears alongside the arrow that connects the top node in the input layer to the top node of the hidden layer. Therefore, the 4.3 is multiplied by .32 to generate 1.376. Thus

* the top node of the hidden layer receives an activation level of 1.376 from the top node in the input layer
* the top node of the hidden layer also receives an activation level of 3.0 x 1.7 or 5.1 from the middle node in the input layer
* finally, the top node of the hidden layer receives an activation level of 6.5 x -.41 or -2.665 from the bottom node in the input layer

Overall, the top node of the hidden layer thus receives an activation level of 1.376 + 5.1 - 2.665 or 3.811 units. The same procedure could be applied to the other nodes except, for aesthetic purposes, some of the weights are missing. Finally, activation from the hidden layer travels to the output layer, also weighted by the various numbers.

As a trivial complication, the activation that enters each node is usually translated to another level of activation using a formula called the **activation function.** A common example is known as the sigmoid function, presented in the following box

|  |
| --- |
| Activation that leaves the node = 1 / (1 + eactivation that enters the node) |

Eventually, after all the calculations have been implemented, this neural network predicts the likelihood this person will complete a PhD is .65 or 65%

**How can you estimate the weights and activating function?**

In short, neural networks can predict some outcome from a set of predictors or variables. But, to achieve this goal, the researcher needs to ascertain the weights and to choose an activating function. So, how did the researcher decide the weights should be .32, -.27., .17, and so forth. In essence, researchers utilize software that

* begins with random weights
* estimates the outcome for each person or unit
* gradually changes these weights to minimize the difference between the estimated outcomes and the actual outcomes—a difference that is called the cost
* test these weights with another subset of people or units

To illustrate, in the following example, the software predicted the probability the candidate would complete the PhD is .65 or 65%. Now suppose the candidate actually completed the PhD; that is, the probability the candidate complete the PhD is 1.0 or 100%. The difference between these numbers is .35 or 35%. This procedure could then be repeated, but with a specific number of participants, such as 200. The following table presents an extract of this information. Specifically, this table presents

* the difference between the predicted outcome and actual outcome
* the square of this cost—primarily to eliminate the negative numbers
* the sum of these values divided by 2—which, for some reasons, is the measure that researchers use to gauge the accuracy of this neural network.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Person | Predicted outcome | Actual outcome | Difference | Square of difference |
| John | .65 | 1 = completed | -.35 | .12 |
| Karen | .24 | 1 | -.76 | .58 |
| Len | .19 | 1 | -.81 | .66 |
| Marsha | .74 | 0 = did not complete | .75 | .56 |
| Neil | .45 | 0 | .45 | .20 |
| … | … | … | … | … |
| Olivia | .13 | 0 | .13 | .02 |
|  |  |  |  | **Sum = 2.14** |
|  |  |  | **Cost:** | **Sum/2 = 1.07** |

 In this instance, the predicted outcomes diverge considerably from the actual outcomes. For example, the neural network predicted the likelihood that Len would complete his thesis is only .19, but Len did indeed complete his thesis. Consequently

* the software would then adjust these weights using a variety of methods, such as back-propagation, discussed later
* the software would adjust the weights until the cost or difference is as low as possible, roughly speaking
* for example, the software might generate the following weights and costs



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Person | Predicted outcome | Actual outcome | Difference | Square of differences |
| John | .85 | 1 = completed | .15 | 0.02 |
| Karen | .74 | 1 | .26 | 0.07 |
| Len | .99 | 1 | .01 | 0.00 |
| Marsha | .24 | 0 = did not complete | .24 | 0.06 |
| Neil | .15 | 0 | .15 | 0.02 |
| … | … | … | … | … |
| Olivia | .03 | 0 | .03 | .001 |
|  |  |  |  | **Sum = .17** |
|  |  |  | **Cost:** | **Sum/2 = .085** |

 Finally, to assess this neural network, the software would assess another sample of participants—perhaps Bert, Carla, Don, Edith, Fred, George, and so forth. This other sample—sometimes called a test or hold-out sample, can be utilized to gauge whether the weights generated with one subset of people apply to another subset of people, demonstrating generalizability.

|  |
| --- |
| **Conducting neural networks with SPSS** |

**How to apply a neural network**

 Before clarifying the details of neural networks, you should probably observe an example. This section demonstrates how you can apply a neural network in SPSS. First, when you enter the data, each row should represent one person or unit; each column should represent one outcome or one predictor.



 Then, choose the *Analyze* menu and the *Neural Networks* option. For this example, choose *Multilayer perceptron* as well, although SPSS can be used to apply another class of neural networks called radial basis functions. When the following screen appears,

* use the arrows to specify the outcome—in this instance, completion—into the box called *Dependent Variables*
* specify categorical predictors, such as gender, in the box called *Factors*
* specify the numerical predictors, such as GPA, papers, and IELTS, in the box called *Covariates*



 The other tabs are usually optional. However

* press *Partitions* if you want to change the percentage of people in the test or hold-out sample;
* press *Architecture* if you want to include more than one hidden layer—or more nodes in the hidden layer; this tab can also be used to adjust the activation function and other features of the network
* press *Output* to instruct SPSS to present the weights as well, called synaptic weights; strangely, SPSS does not generate this important information by default
* furthermore, in this tab, perhaps also choose *Independent variable important analysis*
* when ready, press *OK* to execute the analysis

**How to interpret a neural network**

 SPSS generates several tables and figures. Here is the key figure.



In this figure, SPSS has not attached numbers or weight to the lines that connect the nodes. Instead, the colour and thickness of these lines represents the weights

* blue indicates the weight exceeds 0; grey indicates the weight is less than 0
* thick lines indicate the weight diverges considerably from 0

Otherwise, this figure is similar to the neural network that appeared earlier, except

* the output layer comprises two nodes, representing completion and non-completion
* the boxes called bias is designed to correct systematic errors in the predictions

Another table is called parameter estimates. This table primarily specifies the weights and biases more precisely



The next table is useful whenever the output is categorical. In the testing sample—the information that was not used to generate the weights—19 people who were predicted to complete the thesis did compete; in contrast, 13 people who were predicted to complete the thesis did not compete. The difference between these two numbers is not especially impressive, indicating the neural network did not predict the outcome especially well.



 The final graph can be informative too. This graph indicates that, overall, GPA affects the outcome more than IELTS or number of papers. So, if you wanted to expedite the procedure and collect information about one or two characteristics only, you would confine your data collection to GPA. Likewise, if candidates wanted to know how to increase the likelihood they will complete, they would also focus on their GPA.



|  |
| --- |
| **Rationale** |

**How does SPSS estimate the weights**?

 If you learn more about neural networks, you might be able to enhance the accuracy of outcomes. For example, one vital question revolves around how statistical software estimate the weights. Does the software continue to adjust the weights haphazardly until the cost—or difference between the predicted outcome and actual outcome—is minimized? Or does the software utilize a more systematic method? In practice, software programs use systematic methods. One of the most common methods is called back propagation. In essence

* initially, the software utilizes random weights and biases
* next, the software examines one of the weights more closely—a weight that connects the outcome layer to the previous hidden layer.
* the software constructs a graph that relates this weight to the cost—the overall difference between the predicted outcome and actual outcome



* the software then attempts to choose a weight that minimizes the cost—right at the bottom of this graph
* to construct this graph and identify the bottom, the software could attempt every possible weight
* but instead, to save time, the software can utilize the cost as well as some differential equations; these differential equations help the software identify the point at which the graph is flat
* the software then calculates the cost after this weight has been updated—and repeats this procedure—until the cost is as low as possible.

Finally, the software applied this procedure to the other weights and biases. The software begins with the arrows that connect the output layer with the previous hidden layer. The software ends with the arrows that connect the input layer with the first hidden layer—and is thus called back propagation.

|  |
| --- |
| **Some other details** |

**Radial basis functions**

 The multilayer perceptron—the neural network we have discussed so far—is the most common. But, some software also applies an alternative, called radial basis functions. More information about radial basis functions will appear soon. In essence

* radial basis functions are more robust to random error
* but these radial basis functions are not quite as versatile; a multilayer perceptron is more diverse and, for example, can entail many hidden layers to represent more complex relationships between the predictors and outcomes

**Common terms**

 Neural networks are one example of machine learning or deep learning. In particular

* Machine learning is an informal term to describe circumstances in which a machine gradually learns more accurate formulas over time
* Deep learning is a technical term for machine learning but primarily used when the network entails at least one hidden layer
* In addition to neural networks, a popular example of deep learning is support vector machines.