Basic Neural Network

Ashish Dutt

February 6, 2018

Basics

A neural network is a model characterized by an activation function, which is used by interconnected information processing units to transform input into output. A neural network has always been compared to human nervous system. Information in passed through interconnected units analogous to information passage through neurons in humans. The first layer of the neural network receives the raw input, processes it and passes the processed information to the hidden layers. The hidden layer passes the information to the last layer, which produces the output. The advantage of neural network is that it is adaptive in nature. It learns from the information provided, i.e. trains itself from the data, which has a known outcome and optimizes its weights for a better prediction in situations with unknown outcome.

1. Perceptron Neural Network

A perceptron, viz. single layer neural network, is the most basic form of a neural network. A perceptron receives multidimensional input and processes it using a weighted summation and an activation function. It is trained using a labeled data and learning algorithm that optimize the weights in the summation processor. A major limitation of perceptron model is its inability to deal with non-linearity.

2. Multilayer Neural Network

A multilayered neural network overcomes this limitation and helps solve non-linear problems. The input layer connects with hidden layer, which in turn connects to the output layer. The connections are weighted and weights are optimized using a learning rule.

3. Learning Rules used by Neural Network

The learning rule is used to calculate the error at the output unit. This error is backpropagated to all the units such that the error at each unit is proportional to the contribution of that unit towards total error at the output unit. The errors at each unit are then used to optimize the weight at each connection.

There are many learning rules that are used with neural network:

a) least mean square;
b) gradient descent;
c) newton's rule;
d) conjugate gradient etc.

Fitting a Neural Network in R

I begin by dividing the data into train and test set using random sampling.

Example

```r
# clean the workspace
rm(list=ls())
library(caTools) # for sample.split()
data(iris)
set.seed(2018)
# Split the data into train and test
sample = sample.split(iris$Species, SplitRatio = .75)
iristrain = subset(iris, sample == TRUE)
iristest  = subset(iris, sample == FALSE)

dim(iristrain)
```

```
## [1] 114   5
```

```r
dim(iristest)
```

```
## [1] 36  5
```

```r
nnet_iristrain <- iristrain
```

Next, I binarize the categorical output.

```r
# Binarize the categorical output
nnet_iristrain <- cbind(nnet_iristrain, iristrain$Species == 'setosa')
nnet_iristrain <- cbind(nnet_iristrain, iristrain$Species == 'versicolor')
nnet_iristrain <- cbind(nnet_iristrain, iristrain$Species == 'virginica')

names(nnet_iristrain)[6] <- 'setosa'
names(nnet_iristrain)[7] <- 'versicolor'
names(nnet_iristrain)[8] <- 'virginica'

head(nnet_iristrain)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species setosa
## 1          5.1         3.5          1.4         0.2  setosa   TRUE
## 2          4.9         3.0          1.4         0.2  setosa   TRUE
## 3          4.7         3.2          1.3         0.2  setosa   TRUE
## 4          4.6         3.1          1.5         0.2  setosa   TRUE
## 5          5.0         3.6          1.4         0.2  setosa   TRUE
## 6          5.4         3.9          1.7         0.4  setosa   TRUE
##   versicolor virginica
## 1      FALSE     FALSE
## 2      FALSE     FALSE
## 3      FALSE     FALSE
## 4      FALSE     FALSE
## 5      FALSE     FALSE
## 6      FALSE     FALSE
```

create the neual network

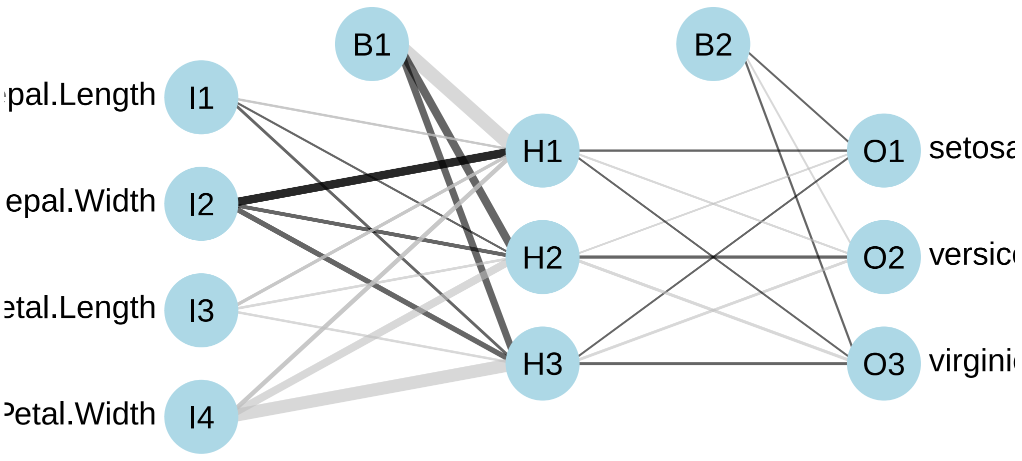
```r
library(neuralnet)
nn <- neuralnet(setosa+versicolor+virginica ~
                  Sepal.Length+Sepal.Width
                +Petal.Length
```

```
                 +Petal.Width,
                 data=nnet_iristrain,
                 hidden=c(3))
```

```
plot(nn)
```

Another plot

```
library(NeuralNetTools)
nn1 <- neuralnet(setosa+versicolor+virginica ~
                   Sepal.Length+Sepal.Width
                 +Petal.Length
                 +Petal.Width,
                 data=nnet_iristrain,
                 hidden=c(3))
plotnet(nn1, alpha=0.6)
```



In the above figure, our model has 3 hidden layers. The black lines show the connections with weights. The weights are calculated using the back propagation algorithm explained earlier. The blue line is the displays the bias term.

Now, we predict the Species using the neural network model.

Make predictions

```
mypredict <- compute(nn, iris[-5])$net.result
# Put multiple binary output to categorical output
```

```r
maxidx <- function(arr) {
  return(which(arr == max(arr)))
}
idx <- apply(mypredict, c(1), maxidx)
prediction <- c('setosa', 'versicolor', 'virginica')[idx]
table(prediction, iris$Species)
```

```
##
## prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         49         1
##   virginica       0          1        49
```