CUHK RMSC4002 Tutorial 9

Benjamin Chan

November 20, 2018

(Optional) Reference

1. Machine Learning Course by Andrew Ng: https://www.coursera.org/learn/machine-learning
2. Deep Learning Specialization by Andrew Ng: https://www.coursera.org/specializations/deep-learning
3. Deep Learning Book by Ian Goodfellow, Yoshua Bengio and Aaron Courville: http://www.deeplearningbook.org/

Packages

```
library(nnet)                                      # Feed-forward neural networks with one hidden lay
```

Artificial Neural Network

The famous Fisher's iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are iris setosa, versicolor, and virginica.

```
data(iris)                                         # data: load specified data sets
str(iris)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
X <- iris[,1:4]
Y <- (iris[,5] == "setosa")*1 + (iris[,5] == "versicolor")*2 + (iris[,5] == "virginica")*3
```

Linear Output

```
# 4-2-1 Neural Network
iris.nn <- nnet(X, Y, size = 2, linout = T)        # 2 units in hidden layer; linear output
```

```
# weights:  13
initial  value 518.024625
iter  10 value 11.567190
iter  20 value 5.659790
iter  30 value 5.369651
iter  40 value 5.137939
iter  50 value 4.851909
iter  60 value 4.847075
iter  70 value 4.846536
iter  80 value 4.843398
iter  90 value 4.819621
```

```
iter 100 value 4.671845
final  value 4.671845
stopped after 100 iterations
```

```r
summary(iris.nn)                                          # Summary of output
```

```
a 4-2-1 network with 13 weights
options were - linear output units
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1
  1.45  -1.08  -0.46   1.44  -1.77
 b->h2 i1->h2 i2->h2 i3->h2 i4->h2
 -3.02   0.09  -0.53   0.12   2.31
 b->o h1->o h2->o
 0.89  2.52  2.33
```

The result is summarized as:

$$
\begin{aligned}
h_1 &= 1.45 + (-1.08)x_1 + (-0.46)x_2 + (1.44)x_3 + (-1.77)x_4 \\
h_2 &= -3.02 + (0.09)x_1 + (-0.53)x_2 + (0.12)x_3 + (2.31)x_4 \\
h_1' &= \frac{\exp(h_1)}{1 + \exp(h_1)} \\
h_2' &= \frac{\exp(h_2)}{1 + \exp(h_2)} \\
v &= 0.89 + (2.52)h_1' + (2.33)h_2'
\end{aligned}
$$

```r
pred <- round(iris.nn$fit)                                # Round the fitted values
table(iris[,5], levels(iris$Species)[pred])               # Classification table
```

```
             setosa versicolor virginica
  setosa         50          0         0
  versicolor      0         47         3
  virginica       0          1        49
```

Improved Version

To avoid parameter estimates trapped at a local minimum of the error function, we can run several times from different sets of initial parameter values in order to get the optimal weights of ANN (hopefully the true global minimum).

```r
# Try nnet(x,y) k times and output the best trial
# x is the matrix of input variable
# y is the dependent value; y must be factor if linout = F is used

ann <- function(x, y, size, maxit = 100, linout = FALSE, try = 5, ...) {
    ann1 <- nnet(y~., data = x, size = size, maxit = maxit, linout = linout, ...)
    v1 <- ann1$value                                      # First trial

    for (i in 2:try) {
        ann <- nnet(y~., data = x, size = size, maxit = maxit, linout = linout, ...)
        if (ann$value < v1) {
            v1 <- ann$value
            ann1 <- ann
        }
    }
```

```
    return(ann1)
}
```

Logistic Output

The csv file `fin-ratio.csv` contains financial ratios of 680 securities listed in the main board of Hong Kong Stock Exchange in 2002. There are six financial variables, namely, Earning Yield (EY), Cash Flow to Price (CFTP), logarithm of Market Value (ln MV), Dividend Yield (DY), Book to Market Equity (BTME), Debt to Equity Ratio (DTE). Among these companies, there are 32 Blue Chips which are the Hang Seng Index Constituent Stocks. The last column HSI is a binary variable indicating whether the stock is a Blue Chip or not.

```
d <- read.csv("./../Dataset/fin-ratio.csv")

Y <- as.factor(d$HSI)                                    # Output: Y

var <- names(d)[!names(d) %in% "HSI"]                    # Exclude HSI
X <- d[,var]                                             # Input:  X

# results = 'hide', Default: logistic output
fin.nn <- ann(X, Y, size = 2, maxit = 200, try = 10)
```

```
summary(fin.nn)
```

```
a 6-2-1 network with 17 weights
options were - entropy fitting
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1
-152.75    5.80    7.36   16.05    0.30    3.28   -1.17
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2
   6.44   17.17   14.96   -3.01   -0.63    1.70  -14.11
   b->o   h1->o   h2->o
 -42.51   46.27  -18.94
```

```
fin.nn$value                                             # Display the best value
```

```
[1] 5.996485
```

```
Prediction <- round(fin.nn$fit)
Reference <- d$HSI                                       # Ground-truth labels

table(Prediction, Reference)                             # Classification table
```

```
          Reference
Prediction   0   1
         0 647   0
         1   1  32
```

Measure of Performance

Note that:

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total Observation No}} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} = \frac{TP}{TP + FN}$$

$$F_1 = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2}\right)^{-1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

```r
(Accuracy <- sum((Prediction == Reference))/length(Prediction))
```

```
[1] 0.9985294
```

```r
(Precision <- sum(Prediction == 1 & Reference == 1)/sum(Prediction == 1))
```

```
[1] 0.969697
```

```r
(Recall <- sum(Prediction == 1 & Reference == 1)/sum(Reference == 1))
```

```
[1] 1
```

```r
(F1 <- 1/((1/Precision + 1/Recall)/2))
```

```
[1] 0.9846154
```

Remark

Training error rate does not reflect the classification performance accurately. In fact, you can randomly choose some observations as training data and remaining observations as testing data.