Neural Networks

Blake Pappas

2023-12-17

Neural Networks in R

We need to load the following packages:

```
library(caret)
library(dplyr)
library(neuralnet)
```

In this exercise, we use the "breast_cancer.csv" file.

The goal is to predict the type of cancer based on characteristics of cells.

The "Class" variable contains cancer type. 2 means benign and 4 means malignant.

Split the data into 60% training and 40% testing:

```
cancer = read.csv("breast_cancer.csv")

train_rows = createDataPartition(y = cancer$Class,
                                 p = 0.60, list = FALSE)

cancer = cancer %>%
  mutate(benign = ifelse(Class == 2, 1, 0),
         malignant = ifelse(Class == 4, 1, 0))

cancer_train = cancer[train_rows, ]
cancer_test = cancer[-train_rows, ]
```

Use a "for" loop to build a series of neural network models, with increasing numbers of hidden layers (1 layer to 5 layers).

Evaluate the performance of each model, and report the best model (i.e., the best number of hidden layers and the best accuracy).

Keep the number of neurons at each layer constant (2 neurons each layer), the learning rate constant (set learningrate to be 0.01), and set the training epochs to be 3.

```
results = data.frame()

best_accuracy = 0

best_layer = 0

for(i in 1:5) {

  hidden_layers = rep(2, i)

  # Train the Neural Network
  NN_model = neuralnet(benign + malignant ~ Clump.Thickness + Uniformity.of.Cell.Size + Uniformity.of.C
                       data = cancer_train,
                       act.fct = "logistic",
                       linear.output = FALSE,
                       hidden = hidden_layers,
                       algorithm = "backprop",
                       learningrate = 0.01,
                       rep = 3)

  # Make Predictions
  pred = neuralnet::compute(NN_model, cancer_test[, 1:9])$net.result

  # Convert to Class Labels
  outcomes = c("2", "4")
  pred_label = outcomes[max.col(pred)]

  # Evaluate Performance
  accuracy = confusionMatrix(factor(pred_label), factor(cancer_test$Class),
                             positive = "4")$overall[1]

  cm = confusionMatrix(factor(pred_label), factor(cancer_test$Class),
                       positive = "4")

  print(cm)

  # Store the Results in the Data Frame
  run = data.frame(Layers = i, Accuracy = accuracy)
```

```
  # Append the Each Consecutive Run's Results to the Data Frame
  results = rbind(results, run)

  if(accuracy > best_accuracy) {

    best_run = NN_model

    best_layer = i

    best_accuracy = accuracy

  }
}
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   2   4
##          2 171   0
##          4   5  97
##
##                Accuracy : 0.9817
##                  95% CI : (0.9578, 0.994)
##     No Information Rate : 0.6447
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9605
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 1.0000
##             Specificity : 0.9716
##          Pos Pred Value : 0.9510
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3553
##          Detection Rate : 0.3553
##    Detection Prevalence : 0.3736
##       Balanced Accuracy : 0.9858
##
##        'Positive' Class : 4
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   2   4
##          2 171   0
##          4   5  97
##
##                Accuracy : 0.9817
##                  95% CI : (0.9578, 0.994)
##     No Information Rate : 0.6447
##     P-Value [Acc > NIR] : < 2e-16
##
```

```
##                  Kappa : 0.9605
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 1.0000
##             Specificity : 0.9716
##          Pos Pred Value : 0.9510
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3553
##          Detection Rate : 0.3553
##    Detection Prevalence : 0.3736
##       Balanced Accuracy : 0.9858
##
##        'Positive' Class : 4
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   2   4
##          2 170   0
##          4   6  97
##
##                Accuracy : 0.978
##                  95% CI : (0.9528, 0.9919)
##     No Information Rate : 0.6447
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.9527
##
##  Mcnemar's Test P-Value : 0.04123
##
##             Sensitivity : 1.0000
##             Specificity : 0.9659
##          Pos Pred Value : 0.9417
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3553
##          Detection Rate : 0.3553
##    Detection Prevalence : 0.3773
##       Balanced Accuracy : 0.9830
##
##        'Positive' Class : 4
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   2   4
##          2 171   0
##          4   5  97
##
##                Accuracy : 0.9817
##                  95% CI : (0.9578, 0.994)
##     No Information Rate : 0.6447
##     P-Value [Acc > NIR] : < 2e-16
##
```

```
##                  Kappa : 0.9605
##
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 1.0000
##             Specificity : 0.9716
##          Pos Pred Value : 0.9510
##          Neg Pred Value : 1.0000
##              Prevalence : 0.3553
##          Detection Rate : 0.3553
##    Detection Prevalence : 0.3736
##       Balanced Accuracy : 0.9858
##
##        'Positive' Class : 4
##
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   2   4
##          2 176  97
##          4   0   0
##
##                Accuracy : 0.6447
##                  95% CI : (0.5848, 0.7014)
##     No Information Rate : 0.6447
##     P-Value [Acc > NIR] : 0.5276
##
##                  Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.0000
##             Specificity : 1.0000
##          Pos Pred Value :    NaN
##          Neg Pred Value : 0.6447
##              Prevalence : 0.3553
##          Detection Rate : 0.0000
##    Detection Prevalence : 0.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 4
##
```

```
print(results)
```

```
##           Layers  Accuracy
## Accuracy       1 0.9816850
## Accuracy1      2 0.9816850
## Accuracy2      3 0.9780220
## Accuracy3      4 0.9816850
## Accuracy4      5 0.6446886
```