Support Vector Machine and Neural Networks

Blake Pappas

2023-12-17

```
library(dplyr)
library(e1071)
library(caret)
library(neuralnet)
library(FSelectorRcpp)
```

Part I. Support Vector Machine in R

Load the "bank_small.csv" Dataset:

```
bank = read.csv("bank_small.csv", stringsAsFactors = TRUE)

train_rows = createDataPartition(y = bank$y,
                                 p = 0.80, list = FALSE)

bank_train = bank[train_rows, ]
bank_test = bank[-train_rows, ]
```

Train a Linear SVM Model Using All Variables

```
# Train the Model
svm_model_linear = svm(y ~ ., data = bank_train,
                       kernel = "linear")

# Make Predictions
pred_svm_model_linear = predict(svm_model_linear, bank_test)

# Performance Evaluation
# Confusion Matrix
confusionMatrix(pred_svm_model_linear, bank_test$y, mode = "prec_recall", positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
```

```
##        no  873  84
##        yes  10  33
##
##                Accuracy : 0.906
##                  95% CI : (0.8862, 0.9234)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.01167
##
##                   Kappa : 0.3731
##
##  Mcnemar's Test P-Value : 5.098e-14
##
##               Precision : 0.7674
##                  Recall : 0.2821
##                      F1 : 0.4125
##              Prevalence : 0.1170
##          Detection Rate : 0.0330
##    Detection Prevalence : 0.0430
##       Balanced Accuracy : 0.6354
##
##        'Positive' Class : yes
##
```

Train a 2-Degree Polynomial SVM Model Using All Variables

```r
# Train the Model
svm_model_polynomial = svm(y ~ ., data = bank_train,
                kernel = "polynomial", degree = 2)

# Make Predictions
pred_svm_model_polynomial = predict(svm_model_polynomial, bank_test)

# Performance Evaluation
# Confusion Matrix
confusionMatrix(pred_svm_model_polynomial, bank_test$y, mode = "prec_recall", positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  883 117
##        yes   0   0
##
##                Accuracy : 0.883
##                  95% CI : (0.8614, 0.9023)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.5246
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
```

```
##
##               Precision :    NA
##                  Recall : 0.000
##                      F1 :    NA
##              Prevalence : 0.117
##          Detection Rate : 0.000
##    Detection Prevalence : 0.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : yes
##
```

Train a Gaussian (Radial) SVM Model Using All Variables

```
# Train the Model
svm_model_radial = svm(y ~ ., data = bank_train,
                       kernel = "radial")

# Make Predictions
pred_svm_model_radial = predict(svm_model_radial, bank_test)

# Performance Evaluation
# Confusion Matrix
confusionMatrix(pred_svm_model_radial, bank_test$y, mode = "prec_recall", positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  877  91
##        yes   6  26
##
##                Accuracy : 0.903
##                  95% CI : (0.883, 0.9206)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.02527
##
##                   Kappa : 0.3145
##
##  Mcnemar's Test P-Value : < 2e-16
##
##               Precision : 0.8125
##                  Recall : 0.2222
##                      F1 : 0.3490
##              Prevalence : 0.1170
##          Detection Rate : 0.0260
##    Detection Prevalence : 0.0320
##       Balanced Accuracy : 0.6077
##
##        'Positive' Class : yes
##
```

Train a Linear SVM Model Using the Filter Approach (Information Gain)

```
# Information Gain
IG_linear = information_gain(y ~ ., data = bank_train)

# Select Top 10 Attributes
top10 = cut_attrs(IG_linear, k = 10)

bank_top10_train = bank_train %>% select(top10, y)
bank_top10_test = bank_test %>% select(top10, y)

# Train the Model
svm_model_linear_IG = svm(y ~ ., data = bank_top10_train,
                          kernel = "linear")

# Make Predictions
pred_svm_model_linear_IG = predict(svm_model_linear_IG, bank_top10_test)

# Performance Evaluation
# Confusion Matrix
confusionMatrix(pred_svm_model_linear_IG, bank_top10_test$y, mode = "prec_recall", positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  873  84
##        yes  10  33
##
##                Accuracy : 0.906
##                  95% CI : (0.8862, 0.9234)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.01167
##
##                   Kappa : 0.3731
##
##  Mcnemar's Test P-Value : 5.098e-14
##
##               Precision : 0.7674
##                  Recall : 0.2821
##                      F1 : 0.4125
##              Prevalence : 0.1170
##          Detection Rate : 0.0330
##    Detection Prevalence : 0.0430
##       Balanced Accuracy : 0.6354
##
##        'Positive' Class : yes
##
```

Train a 2-Degree Polynomial SVM Model Using the Filter Approach (Information Gain)

```
# Information Gain
IG_polynomial = information_gain(y ~ ., data = bank_train)

# Select Top 7 Attributes
top7 = cut_attrs(IG_polynomial, k = 7)

bank_top7_train = bank_train %>% select(top7, y)
bank_top7_test = bank_test %>% select(top7, y)

# Train the Model
svm_model_polynomial_IG = svm(y ~ ., data = bank_top7_train,
                          kernel = "polynomial")

# Make Predictions
pred_svm_model_polynomial_IG = predict(svm_model_polynomial_IG, bank_top7_test)

# Performance Evaluation
# Confusion Matrix
confusionMatrix(pred_svm_model_polynomial_IG, bank_top7_test$y, mode = "prec_recall", positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  883 117
##        yes   0   0
##
##                Accuracy : 0.883
##                  95% CI : (0.8614, 0.9023)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.5246
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : <2e-16
##
##               Precision :    NA
##                  Recall : 0.000
##                      F1 :    NA
##              Prevalence : 0.117
##          Detection Rate : 0.000
##    Detection Prevalence : 0.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : yes
##
```

Train a Gaussian (Radial) SVM Model Using the Filter Approach (Information Gain)

```r
# Information Gain
IG_radial = information_gain(y ~ ., data = bank_train)

# Select Top 5 Attributes
top5 = cut_attrs(IG_radial, k = 5)

bank_top5_train = bank_train %>% select(top5, y)
bank_top5_test = bank_test %>% select(top5, y)

# Train the Model
svm_model_radial_IG = svm(y ~ ., data = bank_top5_train,
                          kernel = "radial")

# Make Predictions
pred_svm_model_radial_IG = predict(svm_model_radial_IG, bank_top5_test)

# Performance Evaluation
# Confusion Matrix
confusionMatrix(pred_svm_model_radial_IG, bank_top5_test$y, mode = "prec_recall", positive = "yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  no yes
##        no  876  91
##        yes   7  26
##
##                Accuracy : 0.902
##                  95% CI : (0.8819, 0.9197)
##     No Information Rate : 0.883
##     P-Value [Acc > NIR] : 0.03202
##
##                   Kappa : 0.3112
##
##  Mcnemar's Test P-Value : < 2e-16
##
##               Precision : 0.7879
##                  Recall : 0.2222
##                      F1 : 0.3467
##              Prevalence : 0.1170
##          Detection Rate : 0.0260
##    Detection Prevalence : 0.0330
##       Balanced Accuracy : 0.6071
##
##        'Positive' Class : yes
##
```

Part II. Neural Networks in R

Load the "wine.csv" Dataset:

```
wine = read.csv("wine.csv")

train_rows = createDataPartition(y = wine$Type,
                                 p = 0.80, list = FALSE)
```

```
# Data Pre-Processing
normalize = function(x) {
  return((x - min(x)) / (max(x) - min(x)))
  }

wine = wine %>%
  mutate(A = ifelse(Type == "A", 1, 0),
         B = ifelse(Type == "B", 1, 0),
         C = ifelse(Type == "C", 1, 0)) %>%
  mutate_at(2:14, normalize)

wine_train = wine[train_rows, ]
wine_test = wine[-train_rows, ]
```

Train a Neural Network Classifier: 1 Hidden Layer with 2 Nodes and a Learning Rate of 0.1

```
# Train the Neural Network
nn_model = neuralnet(A + B + C ~ Alcohol + Malic_Acid + Ash + Ash_Alcalinity + Magnesium + Total_Phenols
                     data = wine_train,
                     act.fct = "logistic",
                     linear.output = FALSE,
                     hidden = 2,
                     algorithm = "backprop",
                     learningrate = 0.1,
                     rep = 1)

# Make Predictions and Evaluate Performance
pred = neuralnet::compute(nn_model, wine_test[, 2:14])$net.result

outcomes = c("A", "B", "C")
pred_label = outcomes[max.col(pred)]
confusionMatrix(factor(pred_label), factor(wine_test$Type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C
##          A 11  0  0
```

```
##          B  0 14  1
##          C  0  0  8
##
## Overall Statistics
##
##                Accuracy : 0.9706
##                  95% CI : (0.8467, 0.9993)
##     No Information Rate : 0.4118
##     P-Value [Acc > NIR] : 3.92e-12
##
##                   Kappa : 0.9548
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            1.0000   1.0000   0.8889
## Specificity            1.0000   0.9500   1.0000
## Pos Pred Value         1.0000   0.9333   1.0000
## Neg Pred Value         1.0000   1.0000   0.9615
## Prevalence             0.3235   0.4118   0.2647
## Detection Rate         0.3235   0.4118   0.2353
## Detection Prevalence   0.3235   0.4412   0.2353
## Balanced Accuracy      1.0000   0.9750   0.9444
```

Train a Neural Network Classifier: 3 Hidden Layers with 2 Nodes Each and a Learning Rate of 0.1

```
# Train the Neural Network
nn_model = neuralnet(A + B + C ~ Alcohol + Malic_Acid + Ash + Ash_Alcalinity + Magnesium + Total_Phenols
                     data = wine_train,
                     act.fct = "logistic",
                     linear.output = FALSE,
                     hidden = rep(2, 3),
                     algorithm = "backprop",
                     learningrate = 0.1,
                     rep = 1)

# Make Predictions and Evaluate Performance
pred = neuralnet::compute(nn_model, wine_test[, 2:14])$net.result

outcomes = c("A", "B", "C")
pred_label = outcomes[max.col(pred)]
confusionMatrix(factor(pred_label), factor(wine_test$Type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C
##          A 11  0  0
```

```
##          B  0 14  0
##          C  0  0  9
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.8972, 1)
##     No Information Rate : 0.4118
##     P-Value [Acc > NIR] : 7.908e-14
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            1.0000   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000
## Prevalence             0.3235   0.4118   0.2647
## Detection Rate         0.3235   0.4118   0.2647
## Detection Prevalence   0.3235   0.4118   0.2647
## Balanced Accuracy      1.0000   1.0000   1.0000
```

Train a Neural Network Classifier: 5 Hidden Layers with 2, 3, 5, 4, and 2 Nodes and a Learning Rate of 0.1

```
# Train the Neural Network
nn_model = neuralnet(A + B + C ~ Alcohol + Malic_Acid + Ash + Ash_Alcalinity + Magnesium + Total_Phenols
                     data = wine_train,
                     act.fct = "logistic",
                     linear.output = FALSE,
                     hidden = c(2, 3, 5, 4, 2),
                     algorithm = "backprop",
                     learningrate = 0.1,
                     rep = 1)

# Make Predictions and Evaluate Performance
pred = neuralnet::compute(nn_model, wine_test[, 2:14])$net.result

outcomes = c("A", "B", "C")
pred_label = outcomes[max.col(pred)]
confusionMatrix(factor(pred_label), factor(wine_test$Type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C
##          A  0  0  0
```

```
##          B 11 14  9
##          C  0  0  0
##
## Overall Statistics
##
##                Accuracy : 0.4118
##                  95% CI : (0.2465, 0.593)
##     No Information Rate : 0.4118
##     P-Value [Acc > NIR] : 0.565
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            0.0000   1.0000   0.0000
## Specificity            1.0000   0.0000   1.0000
## Pos Pred Value            NaN   0.4118      NaN
## Neg Pred Value         0.6765      NaN   0.7353
## Prevalence             0.3235   0.4118   0.2647
## Detection Rate         0.0000   0.4118   0.0000
## Detection Prevalence   0.0000   1.0000   0.0000
## Balanced Accuracy      0.5000   0.5000   0.5000
```

Train a Neural Network Classifier: 3 Hidden Layers with 2 Nodes Each and a Learning Rate of 0.01

```
# Train the Neural Network
nn_model = neuralnet(A + B + C ~ Alcohol + Malic_Acid + Ash + Ash_Alcalinity + Magnesium + Total_Phenols
                     data = wine_train,
                     act.fct = "logistic",
                     linear.output = FALSE,
                     hidden = rep(2, 3),
                     algorithm = "backprop",
                     learningrate = 0.01,
                     rep = 1)

# Make Predictions and Evaluate Performance
pred = neuralnet::compute(nn_model, wine_test[, 2:14])$net.result

outcomes = c("A", "B", "C")
pred_label = outcomes[max.col(pred)]
confusionMatrix(factor(pred_label), factor(wine_test$Type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  A  B  C
##          A 11  0  0
```

```
##          B  0 14  1
##          C  0  0  8
##
## Overall Statistics
##
##                Accuracy : 0.9706
##                  95% CI : (0.8467, 0.9993)
##     No Information Rate : 0.4118
##     P-Value [Acc > NIR] : 3.92e-12
##
##                   Kappa : 0.9548
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            1.0000   1.0000   0.8889
## Specificity            1.0000   0.9500   1.0000
## Pos Pred Value         1.0000   0.9333   1.0000
## Neg Pred Value         1.0000   1.0000   0.9615
## Prevalence             0.3235   0.4118   0.2647
## Detection Rate         0.3235   0.4118   0.2353
## Detection Prevalence   0.3235   0.4412   0.2353
## Balanced Accuracy      1.0000   0.9750   0.9444
```

Train a Neural Network Classifier: 3 Hidden Layers with 2 Nodes Each and a Learning Rate of 0.3

```
# Train the Neural Network
nn_model = neuralnet(A + B + C ~ Alcohol + Malic_Acid + Ash + Ash_Alcalinity + Magnesium
                     + Total_Phenols + Flavanoids + Nonflavanoid_Phenols + Proanthocyanins
                     + Color_Intensity + Hue + OD280_OD315 + Proline,
                     data = wine_train,
                     act.fct = "logistic",
                     linear.output = FALSE,
                     hidden = 2,
                     algorithm = "backprop",
                     learningrate = 0.1,
                     rep = 1)

# Make Predictions and Evaluate Performance
pred = neuralnet::compute(nn_model, wine_test[, 2:14])$net.result

outcomes = c("A", "B", "C")
pred_label = outcomes[max.col(pred)]
confusionMatrix(factor(pred_label), factor(wine_test$Type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  A  B  C
##          A 11  0  0
##          B  0 14  1
##          C  0  0  8
##
## Overall Statistics
##
##                Accuracy : 0.9706
##                  95% CI : (0.8467, 0.9993)
##     No Information Rate : 0.4118
##     P-Value [Acc > NIR] : 3.92e-12
##
##                   Kappa : 0.9548
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            1.0000   1.0000   0.8889
## Specificity            1.0000   0.9500   1.0000
## Pos Pred Value         1.0000   0.9333   1.0000
## Neg Pred Value         1.0000   1.0000   0.9615
## Prevalence             0.3235   0.4118   0.2647
## Detection Rate         0.3235   0.4118   0.2353
## Detection Prevalence   0.3235   0.4412   0.2353
## Balanced Accuracy      1.0000   0.9750   0.9444
```

Train a Neural Network Classifier: 1 Hidden Layer with 2 Nodes and a Learning Rate of 0.01

```
# Train the Neural Network
nn_model = neuralnet(A + B + C ~ Alcohol + Malic_Acid + Ash + Ash_Alcalinity + Magnesium + Total_Phenols
                     data = wine_train,
                     act.fct = "logistic",
                     linear.output = FALSE,
                     hidden = 2,
                     algorithm = "backprop",
                     learningrate = 0.01,
                     rep = 1)

# Make Predictions and Evaluate Performance
pred = neuralnet::compute(nn_model, wine_test[, 2:14])$net.result

outcomes = c("A", "B", "C")
pred_label = outcomes[max.col(pred)]
confusionMatrix(factor(pred_label), factor(wine_test$Type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  A  B  C
##          A 11  0  0
##          B  0 14  1
##          C  0  0  8
##
## Overall Statistics
##
##                Accuracy : 0.9706
##                  95% CI : (0.8467, 0.9993)
##     No Information Rate : 0.4118
##     P-Value [Acc > NIR] : 3.92e-12
##
##                   Kappa : 0.9548
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            1.0000   1.0000   0.8889
## Specificity            1.0000   0.9500   1.0000
## Pos Pred Value         1.0000   0.9333   1.0000
## Neg Pred Value         1.0000   1.0000   0.9615
## Prevalence             0.3235   0.4118   0.2647
## Detection Rate         0.3235   0.4118   0.2353
## Detection Prevalence   0.3235   0.4412   0.2353
## Balanced Accuracy      1.0000   0.9750   0.9444
```

Train a Neural Network Classifier: 1 Hidden Layer with 2 Nodes and a Learning Rate of 0.5

```
# Train the Neural Network
nn_model = neuralnet(A + B + C ~ Alcohol + Malic_Acid + Ash + Ash_Alcalinity + Magnesium + Total_Phenols
                     data = wine_train,
                     act.fct = "logistic",
                     linear.output = FALSE,
                     hidden = 2,
                     algorithm = "backprop",
                     learningrate = 0.5,
                     rep = 1)

# Make Predictions and Evaluate Performance
pred = neuralnet::compute(nn_model, wine_test[, 2:14])$net.result

outcomes = c("A", "B", "C")
pred_label = outcomes[max.col(pred)]
confusionMatrix(factor(pred_label), factor(wine_test$Type))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  A  B  C
##          A 11  1  0
##          B  0 13  1
##          C  0  0  8
##
## Overall Statistics
##
##                Accuracy : 0.9412
##                  95% CI : (0.8032, 0.9928)
##     No Information Rate : 0.4118
##     P-Value [Acc > NIR] : 9.446e-11
##
##                   Kappa : 0.9101
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C
## Sensitivity            1.0000   0.9286   0.8889
## Specificity            0.9565   0.9500   1.0000
## Pos Pred Value         0.9167   0.9286   1.0000
## Neg Pred Value         1.0000   0.9500   0.9615
## Prevalence             0.3235   0.4118   0.2647
## Detection Rate         0.3235   0.3824   0.2353
## Detection Prevalence   0.3529   0.4118   0.2353
## Balanced Accuracy      0.9783   0.9393   0.9444
```