Analysis of amplitude information techniques in classification of regions

Eduarda Chagas

May 7, 2020

In this script, we will evaluate the performance of the WATG technique for region classification in PolSAR textures.

Importing the packages

```
# Load some packages:
if(!require(caret)) install.packages("caret")
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
if(!require(MLmetrics)) install.packages("MLmetrics")
```

```
## Loading required package: MLmetrics
```

```
##
## Attaching package: 'MLmetrics'
```

```
## The following objects are masked from 'package:caret':
##
##     MAE, RMSE
```

```
## The following object is masked from 'package:base':
##
##     Recall
```

FGPE

```
n.total = 160
regions = c(rep("Forest",40), rep("Sea",80), rep("Urban", 40))

Entropy.Complexity = data.frame("Entropy" = numeric(n.total),
                                "Complexity" = numeric(n.total),
                                "Region" = character(n.total),
                                stringsAsFactors=FALSE)

Entropy.Complexity.csv = read.csv(file="../../Data/EntropyComplexityFGPET1A1.csv",
                                  header=TRUE, sep=",")
Entropy.Complexity$Entropy = Entropy.Complexity.csv[,1]
Entropy.Complexity$Complexity = Entropy.Complexity.csv[,2]
Entropy.Complexity$Region = regions
```

```
split = 0.85
trainIndex = createDataPartition(Entropy.Complexity$Region, p = split, list = FALSE)

x = data.frame(Entropy.Complexity$Entropy[trainIndex], Entropy.Complexity$Complexity[trainIndex])
y = factor(Entropy.Complexity$Region[trainIndex])

x_validation = data.frame("Entropy" = Entropy.Complexity$Entropy[-trainIndex], "Complexity" = Entropy.C
y_validation = factor(Entropy.Complexity$Region[-trainIndex])

Entropy.Complexity = data.frame("Entropy" = Entropy.Complexity$Entropy[trainIndex],
                                "Complexity" = Entropy.Complexity$Complexity[trainIndex],
                                "Region" = Entropy.Complexity$Region[trainIndex],
                                stringsAsFactors=FALSE)
```

Creating KNN model and predicting

```
set.seed(123)
ctrl = trainControl(method="repeatedcv", number = 10, repeats = 10)
knnFit = train(Region~., data = Entropy.Complexity, method = "knn",
               trControl = ctrl,
               preProcess = c("center","scale"),
               tuneLength = 20)

pred = predict(knnFit, newdata = x_validation)

xtab = table(pred, y_validation)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##          y_validation
## pred      Forest Sea Urban
##   Forest       2   0     0
##   Sea          4  12     0
##   Urban        0   0     6
##
## Overall Statistics
##
##                Accuracy : 0.8333
##                  95% CI : (0.6262, 0.9526)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0007719
##
##                   Kappa : 0.7143
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Forest Class: Sea Class: Urban
## Sensitivity                0.33333     1.0000         1.00
## Specificity                1.00000     0.6667         1.00
## Pos Pred Value             1.00000     0.7500         1.00
```

```
## Neg Pred Value              0.81818     1.0000         1.00
## Prevalence                  0.25000     0.5000         0.25
## Detection Rate              0.08333     0.5000         0.25
## Detection Prevalence        0.08333     0.6667         0.25
## Balanced Accuracy           0.66667     0.8333         1.00
```

```
cat("Accuracy: ", Accuracy(pred, y_validation), " Recall: ", Recall(pred, y_validation), " Precision: "
```

```
## Accuracy:  0.8333333  Recall:  1  Precision:  0.3333333  F1-Score:  0.5
```

AAPE

```
n.total = 160
regions = c(rep("Forest",40), rep("Sea",80), rep("Urban", 40))

Entropy.Complexity = data.frame("Entropy" = numeric(n.total),
                                "Complexity" = numeric(n.total),
                                "Region" = character(n.total),
                                stringsAsFactors=FALSE)

Entropy.Complexity.csv = read.csv(file="../../Data/EntropyComplexityAAPED3T1A1.csv",
                                  header=TRUE, sep=",")
Entropy.Complexity$Entropy = Entropy.Complexity.csv[,1]
Entropy.Complexity$Complexity = Entropy.Complexity.csv[,2]
Entropy.Complexity$Region = regions


split = 0.85
trainIndex = createDataPartition(Entropy.Complexity$Region, p = split, list = FALSE)

x = data.frame(Entropy.Complexity$Entropy[trainIndex], Entropy.Complexity$Complexity[trainIndex])
y = factor(Entropy.Complexity$Region[trainIndex])

x_validation = data.frame("Entropy" = Entropy.Complexity$Entropy[-trainIndex], "Complexity" = Entropy.C
y_validation = factor(Entropy.Complexity$Region[-trainIndex])

Entropy.Complexity = data.frame("Entropy" = Entropy.Complexity$Entropy[trainIndex],
                                "Complexity" = Entropy.Complexity$Complexity[trainIndex],
                                "Region" = Entropy.Complexity$Region[trainIndex],
                                stringsAsFactors=FALSE)
```

Creating KNN model and predicting

```
set.seed(123)
ctrl = trainControl(method="repeatedcv", number = 10, repeats = 10)
knnFit = train(Region~., data = Entropy.Complexity, method = "knn",
               trControl = ctrl,
               preProcess = c("center","scale"),
               tuneLength = 20)

pred = predict(knnFit, newdata = x_validation)
```

```r
xtab = table(pred, y_validation)
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##         y_validation
## pred     Forest Sea Urban
##   Forest      3   0     0
##   Sea         3  12     0
##   Urban       0   0     6
##
## Overall Statistics
##
##                Accuracy : 0.875
##                  95% CI : (0.6764, 0.9734)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0001386
##
##                   Kappa : 0.7895
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Forest Class: Sea Class: Urban
## Sensitivity                 0.5000      1.000         1.00
## Specificity                 1.0000      0.750         1.00
## Pos Pred Value              1.0000      0.800         1.00
## Neg Pred Value              0.8571      1.000         1.00
## Prevalence                  0.2500      0.500         0.25
## Detection Rate              0.1250      0.500         0.25
## Detection Prevalence        0.1250      0.625         0.25
## Balanced Accuracy           0.7500      0.875         1.00
```

```r
cat("Accuracy: ", Accuracy(pred, y_validation), " Recall: ", Recall(pred, y_validation), " Precision: "
```

```
## Accuracy:  0.875  Recall:  1  Precision:  0.5  F1-Score:  0.6666667
```