Classification of Regions with Transition Graphs

Eduarda Chagas

Mar 18, 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
```

```
setwd("/home/eduarda/Desktop/Repositories/SAR/SAR-WATG-master/Code/Classification")
```

###Importing the dataset

For this analysis, three SAR images with different regions were used, they are:

- Sierra del Lacandon National Park, Guatemala (purchased April 10, 2015), available at [https://uavsar.jpl.nasa.gov/cgi-bin/product.pl?jobName=Lacand_30202_15043_ 006_150410_L090_CX_01 # data] (https://uavsar.jpl.nasa.gov/cgi-bin/product.pl?jobName=Lacand_30202_15043_006_150410_L090_CX_01 # data);
- Oceanic regions of Cape Canaveral (acquired on September 22, 2016);
- Urban area of the city of Munich, Germany (acquired on June 5, 2015).

A total of 160 samples were considered during the investigation, with 40 forest regions in Guatemala, 80 ocean regions in Cape Canaveral and 40 urban regions in the city of Munich.

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

Entropy.Complexity = data.frame("Entropy" = numeric(n.total),
```

```r
                                 "Complexity" = numeric(n.total),
                                 "Region" = character(n.total),
                                 stringsAsFactors=FALSE)

Entropy.Complexity.csv = read.csv(file="../../Data/EntropyComplexityTGD3T1.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$Entropy[-trainIndex], "Complexity" = Entropy.Co
y_validation = factor(Entropy.Complexity$Region[-trainIndex])

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

##KNN Classifier

###Creating KNN model and predicting

```r
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      4   0     0
##   Sea         1  12     0
##   Urban       1   0     6
##
## Overall Statistics
##
##                Accuracy : 0.9167
##                  95% CI : (0.73, 0.9897)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 1.794e-05
##
```

```
##                   Kappa : 0.8644
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Forest Class: Sea Class: Urban
## Sensitivity                 0.6667     1.0000       1.0000
## Specificity                 1.0000     0.9167       0.9444
## Pos Pred Value              1.0000     0.9231       0.8571
## Neg Pred Value              0.9000     1.0000       1.0000
## Prevalence                  0.2500     0.5000       0.2500
## Detection Rate              0.1667     0.5000       0.2500
## Detection Prevalence        0.1667     0.5417       0.2917
## Balanced Accuracy           0.8333     0.9583       0.9722
```

```
knnFit
```

```
## k-Nearest Neighbors
##
## 136 samples
##   2 predictor
##   3 classes: 'Forest', 'Sea', 'Urban'
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 10 times)
## Summary of sample sizes: 122, 122, 122, 122, 123, 123, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    5  0.8850220  0.8116195
##    7  0.8678132  0.7832829
##    9  0.8656081  0.7783373
##   11  0.8581429  0.7648078
##   13  0.8486538  0.7478204
##   15  0.8486538  0.7478204
##   17  0.8449176  0.7406766
##   19  0.8412912  0.7343622
##   21  0.8354048  0.7231553
##   23  0.8265238  0.7078966
##   25  0.8234542  0.7023128
##   27  0.8293333  0.7109247
##   29  0.8260147  0.7061685
##   31  0.8296960  0.7130329
##   33  0.8377015  0.7254681
##   35  0.8243901  0.7027158
##   37  0.8224524  0.6998713
##   39  0.8151850  0.6870227
##   41  0.8116062  0.6809599
##   43  0.8002930  0.6613918
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 5.
```

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

```
## Accuracy:  0.9166667  Recall:  1  Precision:  0.6666667  F1-Score:  0.8
```