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

```
# Clear workspace:
rm(list = ls())

# 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/Research/Repositories/PolSARfromITQualitative/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),
                                "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

```
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         2  11     0
##   Urban       0   1     6
##
```

```
## Overall Statistics
##
##                Accuracy : 0.875
##                  95% CI : (0.6764, 0.9734)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0001386
##
##                   Kappa : 0.7966
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Forest Class: Sea Class: Urban
## Sensitivity                 0.6667     0.9167       1.0000
## Specificity                 1.0000     0.8333       0.9444
## Pos Pred Value              1.0000     0.8462       0.8571
## Neg Pred Value              0.9000     0.9091       1.0000
## Prevalence                  0.2500     0.5000       0.2500
## Detection Rate              0.1667     0.4583       0.2500
## Detection Prevalence        0.1667     0.5417       0.2917
## Balanced Accuracy           0.8333     0.8750       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  0.7952418  0.6635815
##    7  0.7962729  0.6608656
##    9  0.7766905  0.6248621
##   11  0.7861722  0.6356394
##   13  0.7955806  0.6505883
##   15  0.7862088  0.6368851
##   17  0.7727381  0.6130126
##   19  0.7726520  0.6110236
##   21  0.7606722  0.5924932
##   23  0.7435476  0.5669340
##   25  0.7411941  0.5654720
##   27  0.7275311  0.5393842
##   29  0.7240769  0.5297953
##   31  0.7096813  0.5039074
##   33  0.6889304  0.4671878
##   35  0.6805696  0.4498566
##   37  0.6758388  0.4383717
##   39  0.6783663  0.4421392
##   41  0.6746538  0.4333381
##   43  0.6680934  0.4243187
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 7.
```

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

```
## Accuracy:  0.875  Recall:  1  Precision:  0.6666667  F1-Score:  0.8
```

SVM Classifier

Creating svm model using non-linear kernel

```
svmFit <- train(Region ~., data = Entropy.Complexity, method = "svmRadial",
                 trControl=ctrl,
                 preProcess = c("center", "scale"),
                 tuneLength = 20)
pred = predict(svmFit, newdata = x_validation)

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

```
## Confusion Matrix and Statistics
##
##         y_validation
## pred     Forest Sea Urban
##   Forest      3   0     0
##   Sea         3  11     0
##   Urban       0   1     6
##
## Overall Statistics
##
##                Accuracy : 0.8333
##                  95% CI : (0.6262, 0.9526)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0007719
##
##                   Kappa : 0.7241
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Forest Class: Sea Class: Urban
## Sensitivity                 0.5000     0.9167       1.0000
## Specificity                 1.0000     0.7500       0.9444
## Pos Pred Value              1.0000     0.7857       0.8571
## Neg Pred Value              0.8571     0.9000       1.0000
## Prevalence                  0.2500     0.5000       0.2500
## Detection Rate              0.1250     0.4583       0.2500
## Detection Prevalence        0.1250     0.5833       0.2917
## Balanced Accuracy           0.7500     0.8333       0.9722
```

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

```
## Accuracy:  0.8333333  Recall:  1  Precision:  0.5  F1-Score:  0.6666667
```

Random Forest Classifier

Creating Random Forest model and predicting

```
rfFit <- train(Region~., data = Entropy.Complexity, method = "rf",
               trControl = ctrl,
               preProcess = c("center","scale"),
               tuneLength = 20)
```

```
## note: only 1 unique complexity parameters in default grid. Truncating the grid to 1 .
```

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

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

```
## Accuracy:  0.75  Recall:  0.75  Precision:  0.5  F1-Score:  0.6
```