Classification of Regions with Bandt-Pompe

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/EntropyComplexityBPD3T1.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)
```

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)
#knnFit$results

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

```
## Confusion Matrix and Statistics
##
##          y_validation
## pred      Forest Sea Urban
##   Forest       4   2     1
##   Sea          2  10     0
```

```
##   Urban       0   0     5
##
## Overall Statistics
##
##                Accuracy : 0.7917
##                  95% CI : (0.5785, 0.9287)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.003305
##
##                   Kappa : 0.6667
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Forest Class: Sea Class: Urban
## Sensitivity                 0.6667     0.8333       0.8333
## Specificity                 0.8333     0.8333       1.0000
## Pos Pred Value              0.5714     0.8333       1.0000
## Neg Pred Value              0.8824     0.8333       0.9474
## Prevalence                  0.2500     0.5000       0.2500
## Detection Rate              0.1667     0.4167       0.2083
## Detection Prevalence        0.2917     0.5000       0.2083
## Balanced Accuracy           0.7500     0.8333       0.9167
```

```
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  0.6968059  0.5097741
##    7  0.7077234  0.5312806
##    9  0.7268498  0.5633856
##   11  0.7304121  0.5662909
##   13  0.7424176  0.5882426
##   15  0.7333828  0.5718841
##   17  0.7163425  0.5422465
##   19  0.7284707  0.5603357
##   21  0.7373315  0.5727652
##   23  0.7488462  0.5905829
##   25  0.7565641  0.6029099
##   27  0.7629432  0.6119402
##   29  0.7612070  0.6075207
##   31  0.7656410  0.6138986
##   33  0.7633901  0.6104460
##   35  0.7583095  0.5996170
```

```
##   37  0.7571026  0.5957892
##   39  0.7600220  0.6002503
##   41  0.7585311  0.5957122
##   43  0.7564908  0.5916450
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 31.
```

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

```
## Accuracy:  0.7916667  Recall:  0.5714286  Precision:  0.6666667  F1-Score:  0.6153846
```

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       1   0     1
##   Sea          5  12     0
##   Urban        0   0     5
##
## Overall Statistics
##
##                Accuracy : 0.75
##                  95% CI : (0.5329, 0.9023)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.01133
##
##                   Kappa : 0.5636
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Forest Class: Sea Class: Urban
## Sensitivity                0.16667     1.0000       0.8333
## Specificity                0.94444     0.5833       1.0000
## Pos Pred Value             0.50000     0.7059       1.0000
## Neg Pred Value             0.77273     1.0000       0.9474
## Prevalence                 0.25000     0.5000       0.2500
## Detection Rate             0.04167     0.5000       0.2083
## Detection Prevalence       0.08333     0.7083       0.2083
```

```
## Balanced Accuracy          0.55556     0.7917       0.9167
```

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

```
## Accuracy:  0.75  Recall:  0.5  Precision:  0.1666667  F1-Score:  0.25
```

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:  0.6666667  Recall:  0.375  Precision:  0.5  F1-Score:  0.4285714
```