

# 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)] ([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(Rregion~., 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	1	1	0
##	Sea	5	11	0

```

## Urban      0    0    6
##
## Overall Statistics
##
##           Accuracy : 0.75
##           95% CI : (0.5329, 0.9023)
##           No Information Rate : 0.5
##           P-Value [Acc > NIR] : 0.01133
##
##           Kappa : 0.5714
##
## McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: Forest Class: Sea Class: Urban
## Sensitivity           0.16667      0.9167      1.00
## Specificity           0.94444      0.5833      1.00
## Pos Pred Value        0.50000      0.6875      1.00
## Neg Pred Value        0.77273      0.8750      1.00
## Prevalence            0.25000      0.5000      0.25
## Detection Rate        0.04167      0.4583      0.25
## Detection Prevalence  0.08333      0.6667      0.25
## Balanced Accuracy      0.55556      0.7500      1.00

```

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 0.7036941 0.5292562
## 7 0.7301300 0.5677146
## 9 0.7333077 0.5728273
## 11 0.7399414 0.5827005
## 13 0.7561245 0.6090629
## 15 0.7492821 0.5981711
## 17 0.7637711 0.6189274
## 19 0.7796227 0.6432683
## 21 0.7861832 0.6532145
## 23 0.7839762 0.6480568
## 25 0.7886099 0.6558995
## 27 0.7909176 0.6594769
## 29 0.7878407 0.6539692
## 31 0.7855879 0.6503894
## 33 0.7883022 0.6541259
## 35 0.7876429 0.6524627

```

```
## 37 0.7923223 0.6583496
## 39 0.7900055 0.6542391
## 41 0.7902198 0.6534399
## 43 0.7908388 0.6542373
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 37.
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
```

## 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  2   0
## Sea         5 10   0
## Urban       0  0   6
##
## Overall Statistics
##
##               Accuracy : 0.7083
##               95% CI : (0.4891, 0.8738)
##      No Information Rate : 0.5
##      P-Value [Acc > NIR] : 0.03196
##
##               Kappa : 0.5088
##
##      McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##               Class: Forest Class: Sea Class: Urban
## Sensitivity           0.16667      0.8333      1.00
## Specificity           0.88889      0.5833      1.00
## Pos Pred Value        0.33333      0.6667      1.00
## Neg Pred Value        0.76190      0.7778      1.00
## Prevalence            0.25000      0.5000      0.25
## Detection Rate        0.04167      0.4167      0.25
## Detection Prevalence  0.12500      0.6250      0.25
```

```
## Balanced Accuracy          0.52778      0.7083      1.00
```

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

```
## Accuracy:  0.7083333  Recall:  0.3333333  Precision:  0.1666667  F1-Score:  0.2222222
```

## 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:  0.5  Precision:  0.5  F1-Score:  0.5
```