Classification of Regions with glcm

Eduarda Chagas

March 11, 2020

Importing the packages

```r
# 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
```

```r
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
```

```r
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.

```r
n.total = 160
regions = c(rep("Forest",40), rep("Sea",80), rep("Urban", 40))
glcm.descriptors = read.csv(file="../../Data/glcm.csv", header=TRUE, sep=",")
GLCM = data.frame(glcm.descriptors, regions)
```

```
split = 0.85
trainIndex = createDataPartition(GLCM$regions, p = split, list = FALSE)

x = data.frame(GLCM[trainIndex, 1:4])
y = factor(GLCM$regions[trainIndex])

x_validation = data.frame(GLCM[-trainIndex, 1:4])
y_validation = factor(GLCM$regions[-trainIndex])

GLCM = data.frame(GLCM[trainIndex, 1:4], "regions" = GLCM$regions[trainIndex])
```

KNN Classifier

Creating KNN model and predicting

```
set.seed(123)
ctrl = trainControl(method="repeatedcv", number = 10, repeats = 10)
knnFit = train(regions ~., data = GLCM, 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      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
```

```
knnFit
```

```
## k-Nearest Neighbors
##
## 136 samples
##   4 predictor
##   3 classes: 'Forest', 'Sea', 'Urban'
##
## Pre-processing: centered (4), scaled (4)
## 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  0.8496355  0.7511041
##    7  0.8516850  0.7514965
##    9  0.8489322  0.7474340
##   11  0.8518919  0.7526964
##   13  0.8460220  0.7430335
##   15  0.8343993  0.7220611
##   17  0.8197821  0.6944644
##   19  0.8130073  0.6824001
##   21  0.8132985  0.6833919
##   23  0.8054579  0.6688018
##   25  0.7907637  0.6427155
##   27  0.7914304  0.6431130
##   29  0.7943993  0.6477877
##   31  0.7966978  0.6506922
##   33  0.7934451  0.6449398
##   35  0.7874322  0.6339659
##   37  0.7800147  0.6205455
##   39  0.7769615  0.6149037
##   41  0.7775952  0.6163302
##   43  0.7792436  0.6188881
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 11.
```

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

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

SVM Classifier

Creating SVM model and predicting

```
svmFit <- train(regions ~., data = GLCM, 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      6   0     0
##   Sea         0  12     0
##   Urban       0   0     6
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.8575, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 5.96e-08
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: Forest Class: Sea Class: Urban
## Sensitivity                   1.00        1.0         1.00
## Specificity                   1.00        1.0         1.00
## Pos Pred Value                1.00        1.0         1.00
## Neg Pred Value                1.00        1.0         1.00
## Prevalence                    0.25        0.5         0.25
## Detection Rate                0.25        0.5         0.25
## Detection Prevalence          0.25        0.5         0.25
## Balanced Accuracy             1.00        1.0         1.00
```

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

```
## Accuracy:  1  Recall:  1  Precision:  1  F1-Score:  1
```

Random Forest Classifier

Creating Random Forest model and predicting

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

```
## note: only 3 unique complexity parameters in default grid. Truncating the grid to 3 .
```

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

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

```
## Accuracy:  0.9583333  Recall:  1  Precision:  0.8333333  F1-Score:  0.9090909
```