4 Support Vector Machines

Aim

Support Vector Machines (SVMs) promise to achieve a maximized classification accuracy in difficult cases. We evaluate whether this promise holds true.

Data

In this training, we are using a ROSIS scene covering the city of Pavia, Northern Italy. The ROSIS sensor covers the spectral range from 430 nm to 860 nm in 115 spectral bands with a bandwidth of 4 nm. As part of the pre-processing, all noisy bands have been removed, leaving 102 spectral bands for the analysis. Unfortunately, no wavelength information for the remaining bands is given. Pixel values represent reflectance values [% * 100]. The image has a spatial resolution of 1.3 m on the ground. It is not georeferenced. Ground truth data include the nine land cover classes listed in table 4.1. Both data and ground truth were provided by Paolo Gamba from the Telecommunications and Remote Sensing Laboratory, Pavia, Italy (http://www.ehu.es/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes). We will use these data sets to create a detailed land cover map of Pavia using SVM classification, aiming to achieve the highest accuracy possible.

Loading the data and required packages

Copy the data sets from moodle to your personal directory and set this path as your working directory. In R, you will need the packages 'terra', 'randomForest', and 'e1071'. The latter still has to be installed before it is available for loading. Like before, feel free to deviate from this work flow (or to use a different language) to gain extra points.

```
setwd ("PATH_TO_YOUR_DIRECTORY")
library (terra)
library (randomForest)
install.packages ("e1071") ## this package contains the SVM implementation
library (e1071)
```

Read the image into your workspace and plot it as real and false color composite.

```
rosis <- rast ("rosis_pavia.tif")
plotRGB (rosis, 43, 30, 15, stretch="lin") ## real color
```

Q4.1 (1 pt): Which band combination is suitable for a false color NIR/R/G composite? Remember that ROSIS is a half-range sensor.

Table 4.1: The nine land cover classes in the Pavia data set

Class code	Name	Description
1	Water	Open water bodies
2	Trees	Any woody vegetation
3	Meadows	Grasslands, parks, and any other non-woody vegetation
4	Brick	Roads with a dark red brick pavement
5	Bare soil	Bare soil and gravelbars next to the river
6	Asphalt	Asphalt roads
7	Bitumen	Bitumen roofs
8	Tiles	Red brick roofs
9	Shadow	Weird class; anything we cannot see due to shadows

The ground truth data for calibration and validation of the SVM model are provided in two point shapefiles. Each class is represented with 50 points. Read these shapefiles as well. We can add the calibration points to the image plot, assigning a distinct color to each class.

```
cal <- vect ("rosis_calpoints.shp")
val <- vect ("rosis_valpoints.shp")

cl <- c ("#0000FF", "#007000", "#00FF00", "#AA0000", "#602000", "#000000",
         "#AAAAAA", "#FF0000", "#AA00AA")
points (cal, pch=19, cex=0.5, col=rep (cl, each=50))
```

Q4.2 (1 pt): Do your recognise additional land cover types that are not represented by the calibration points?

Preparing the classification

To build and train the SVM model, we need the reflectance spectra corresponding to the class memberships of the ground truth points. The spectra can be extracted from the image with the extract() function, the class information (numeric class codes as listed in table 4.1) is taken from the attribute table of the shapefiles. In addition, we create a vector with the class names for labeling purposes.

```
calref <- extract (rosis, cal, ID=F)
calclass <- as.factor (cal$class)
class.names <- c ("water", "trees", "meadow", "brick", "soil", "asphalt",
                  "bitumen", "tiles", "shadow")
```

Visualization of the spectra

The SVM algorithm is a very powerful tool for difficult classification problems with limited class separability. This high performance in turn asks for a complex parameterization of the model and affects the user-friendliness of the approach. To illustrate the separability problems associated with the Pavia data, we take a look at the mean calibration spectrum per class and its standard deviation. Both can be calculated from the 50 spectra per class.

```
classmean <- classsd <- matrix (0, 9, 102) ## create two empty matrices for mean
                                           ## and sd
  rownames (classmean) <- rownames (classsd) <- class.names ## label the rows
sel <- matrix (1:450, 9, 50, byrow=T) ## create an auxiliary matrix that
                                ## describes which spectra belong to which class
```

We use a for()-loop to run the calculation for every class.

```
for (i in 1:9){ ## for each class
  classmean[i,] <- apply (calref[sel[i,],], 2, mean) / 100 ## calculate the mean
                                        ## spectrum and write it to the 1st matrix
  classsd[i,] <- apply (calref[sel[i,],], 2, sd) / 100 ## and the standard
                                        ## deviation to the 2nd matrix
}
dev.off()
plot (classmean[1,], type="n", xlab="Band #", ylab="Reflectance / %", ylim=c (0,
      60)) ## now create an empty plot
for (i in 1:9){ ## for each class
  polygon (c (1:102, 102:1), c (classmean[i,] - classsd[i,], (classmean[i,] +
           classsd[i,])[102:1]), col=cl[i], density=20+i, border=NA, lwd=0.5)
## add a polygon of the mean +/- SD to illustrate the intra-class variability
```

```
  lines (classmean[i,], col=cl[i], lwd=2) ## add the mean line
}
legend ("topleft",legend=class.names, col=cl, lwd=2, ncol=2) ## create a legend
```

Q4.3 (1 pts): Describe which classes are spectrally distinct. Which classes show some overlap with other classes and are thus likely to be confused in the classification?

Grid search for the SVM parameterization

SVMs tackle the classification problem by transferring it to an artificial feature space with a higher dimensionality than the original space. In our case, the original spectral feature space already has 102 dimensions (i.e., the spectral bands). Accordingly, the SVM space has more than 102 dimensions. In theory, the transformation enables a linear solution to an otherwise non-linear separability problem. Kernel functions are used to make this transformation (and re-transformation) computationally feasible. We have to decide which Kernel function to use and parameterize this function with the kernel parameter γ. Subsequently, the algorithm tries to separate the classes spectrally using this set up. This results in several possible solutions that have to be evaluated with respect to their elegance, parsimony, purity, and accuracy. The trade-off between these characteristics is described as the 'cost' of the solution. Solutions with a high cost are down-weighted in the evaluating comparison. The down-weighting is determined through the cost parameter C. Depending on the form of SVM that is used, additional parameters may be required; here, we stay with γ and C.
The first decision to face is the selection of the Kernel function. Hsu et al. (2010) recommend in their SVM tutorial to use the Radial Basis Kernel function (RBF) that is generally most suitable for remote sensing problems. This function is also the default kernel in the e1071 R-package. They further recommend to use a systematic search (the grid search) to identify the combination of γ and C resulting in the best classification accuracy. The suggested ranges of these parameters are from 2^{-5} to 2^{15} for C and from 2^{-15} to 2^{3} for γ. We follow their recommendations.
The grid search takes some time and effort because all combinations are tested with a full model that is evaluated in a ten-fold cross-validation. A ten-fold cross-validation means that each model is built ten times based on 90% of the sample points. The remaining 10% are predicted and compared to the actual observation. Each sample point is removed exactly once from the model. The errors of the individual model runs are used to assess the total (calibration) error of the model. After the grid search, we will identify the combination of γ and C that results in the most accurate solution and use this combination to parameterize the final model.

Q4.4 (1 pts): We test eleven C values (2^{-5}, 2^{-3}, …, 2^{15}) and ten γ values (2^{-15}, 2^{-13}, …, 2^{3}) in the grid search. Considering the ten-fold cross-validation, how many individual SVM models must your poor computer run in total? Give it a big hug.

For the grid search, we first define a list of the parameter values to be tested.

```
ccoef <- 2^seq (-5, 15, 2)  ## list of C values
gcoef <- 2^seq (-15, 3, 2)  ## list of γ values
```

Then we create an empty matrix to store the model results.

```
gsfit <- matrix (0, 11, 10)
```

Two nested loops are used to pass every possible parameter combination to a SVM. The resulting OAC is written to the gsfit-matrix.

```
for (i in 1:11){  ## for every C value
  for (j in 1:10){  ## and every γ value
    svmrun <- svm (calref, calclass, type="C-classification", cross=10,
                   gamma=gcoef[j], cost=ccoef[i]) ## run a SVM with ten-fold CV
```

```
    cfm <- table (calclass, predict (svmrun)) ## build the confusion matrix
    gsfit[i, j] <- sum (diag (cfm)) / sum (cfm) ## write the OAC to gsfit
  }}
```

The result of the grid search can be visualized for a better understanding. The selected parameter combination is highlighted in this plot.

```
gsres <- rast (gsfit[11:1,], extent = ext(-16, 4, -6, 16))
par (pin=c(2.5,2.75))
plot (gsres, col=gray.colors (255,0,0.9,1), xlab="gamma = 2^", ylab="C = 2^",
     las=2, main=expression (paste ("OACcal")), xlim=c (-16, 4), ylim=c(-6, 16))
```

Q4.5 (2 pts): How do γ and C affect the model accuracy? Which γ and C yields the best model fit?

Most SVM implementations offer some functionality to tune the model without the need to perform a manual grid search. In the e1071 package, this is provided through the tune() function.

```
tunemodel <- tune (svm, train.x=calref, train.y=calclass,
                   type="C-classification", kernel="radial")
tune.gamma <- tunemodel$best.model$gamma
tune.cost <- tunemodel$best.model$cost
       abline (v=log2 (tune.gamma), col="#FF0000", lty=2) ## add to the plots
       abline (h=log2 (tune.cost), col="#FF0000", lty=2)
```

In my experience, the result of the tuning yields not the best fit and is hence still outperformed by the manual grid search. This may obviously differ between models an SVM-implementations. Judge yourself.

Q4.6 (2 pts): How do the tuned γ and C differ from the outcome of your own grid search? Which overall accuracy do they yield?

Building the final SVM model

Use the γ and C that you consider the superior combination for a final SVM model. Build this model using your selected parameter combination. We use this final model and apply it on the image data for a pixel-wise prediction of the land cover classes. Subsequently, the prediction is validated against the actual observation in the validation data set.

```
svmmodel <- svm (calref, calclass, type="C-classification", cross=10,...)
       ## replace '...' with your γ and C values

map <- predict (rosis, svmmodel)
clgrad <- colorRampPalette ( c ("#0000FF", "#007000", "#00FF00", "#AA0000",
                          "#602000", "#000000", "#AAAAAA", "#FF0000", "#AA00AA"))
plot (map, col=clgrad(9), legend=F, axes=F)
  add_legend ("topleft",legend=class.names, pch=22, pt.bg=cl, cex=0.7)

cfm.val <- table (val$class, t (extract (map, val, ID=F)))
  rownames (cfm.val) <- colnames (cfm.val) <- class.names
cfm.val
oac <- round (sum (diag (cfm.val)) / sum (cfm.val), 3)
oac
```

```
users.acc <- round (diag (cfm.val) / apply (cfm.val, 1, sum), 2)
users.acc
producers.acc <- round (diag (cfm.val) / apply (cfm.val, 2, sum), 2)
producers.acc
```

Q4.7 (1 pts): Which classes show the lowest accuracies? Does this meet your expectations based on Q4.3?

Comparison to randomForest

We repeat the classification using a random forest model for a better assessment of the SVM performance. Because all data preparation is already done, this is no big deal.

Q4.8 (3 pts): Build a random forest classification model for the Pavia data and perform an accuracy assessment as for the SVM model above. Provide the code for this operation.

Q4.9 (2 pts): Compare the SVM results to the RF results. How do you judge the performance of the approaches?

Q4.10 (3 pts): Read the paper by De Boissieu et al. (2018). The authors claim in paragraph 3.2.3 that 'SVM can work with small training datasets (...) with classes having less than 50 samples.' Does this promise hold true? Take a random subsample of the Pavia data with approximately 30 calibration points per class and repeat the SVM modeling. How does the smaller sample size affect the model performance? Does the Hughes Phenomenonen (i.e., the curse of dimensionality) apply or are there signs of over-fitting?

Q4.11 (3pts):
Come up with one (or more) additional questions of your own that address(es) the paper and its methodology.

Copy the answers to all ten problems and your own question(s) into a single document. You can again get a maximum of 20 points for all answers being correct and earn two extra points for for providing an individual solution. Grading will follow the scheme provided in training #2. Have fun!.

References

Hsu CW, Chang CC, Lin CJ (2003). A practical guide to Support Vector Classification. https://www.cs.sfu.ca/people/Faculty/teaching/726/spring11/svmguide.pdf

De Boissieu F, Sevin B, Cudahy T, Mangeas M, Chevrel S, Ong C, Rodger A, Maurizot P, Laukamp C, Lau I, Touraivane T, Cluzel D, Despinoy M (2018). Regolith-geology mapping with support vector machine: A case study over weathered Ni-bearing peridotites, New Caledonia. International Journal of Applied Earth Observation and Geoinformation 64, 377-385.