Bern University of Applied Sciences

OpenGeoHub

OpenGeoHub Summer School

Mastering Machine Learning for Spatial Prediction II

Practical training

Madlene Nussbaum, 20 August 2020

Contents

Preparation

Load needed packages:

```
library(ranger) # for random forest models
library(quantregForest) # for quantile random forest
library(grpreg) # for group lasso
library(geoGAM) # for the Berne test data set
library(ggplot2) # for graphics
library(pdp) # for partial dependence plots
```

Load again the *Berne* data, select the calibration set and remove missing values in covariates.

```
data(berne)
dim(berne)

## [1] 1052  238

# Select soil pH in 0-10 cm as continuous response,
# select calibration data and remove rows with missing pH
```

```
d.ph10 <- berne[ berne$dataset == "calibration" & !is.na(berne$ph.0.10), ]
d.ph10 <- d.ph10[ complete.cases(d.ph10[13:ncol(d.ph10)]), ]
# covariates start at col 13
l.covar <- names(d.ph10[, 13:ncol(d.ph10)])
```

1 Selection of covariates

For tree based ensemble methods covariate importance can be computed. Based on this measure non-relevant covariates can be excluded and possibly model performance can be increased.

Fit random forest model:

```
set.seed(17)
rf.ph <- ranger(x = d.ph10[, l.covar],
                y = d.ph10$ph.0.10,
                importance = "permutation")
```

Create the importance plot:

```
# Create data frame with largest importance first
d.imp <- data.frame(variable = names(rf.ph$variable.importance),
                    importance = rf.ph$variable.importance)
d.imp <- d.imp[order(d.imp$importance, decreasing = T), ]
ggplot(d.imp[1:30, ], aes(x = reorder(variable, importance),
                          y = importance, fill = importance)) +
  geom_bar(stat = "identity", position = "dodge") +
  coord_flip() +
  ylab("Variable Importance") +
  xlab("") +
  theme(legend.position = "none") +
  scale_fill_gradient(low = "grey", high = "darkblue")
```

Then, reduce covariates by recursive backward elimination using permuted covariate importance:

```
# speed up the process by removing 5-10 covariates at a time
s.seq <- sort( c( seq(5, 95, by = 5),
                  seq(100, length(l.covar), by = 10) ),
               decreasing = T)

# collect results in list
qrf.elim <- oob.mse <- list()

# save model and OOB error of current fit
qrf.elim[[1]] <- rf.ph
oob.mse[[1]] <- qrf.elim[[1]]$prediction.error
l.covar.sel <- l.covar

# Iterate through number of retained covariates
```

```
for( ii in 1:length(s.seq) ){
  # Get importance, decreasingly ordered
  t.imp <- qrf.elim[[ii]]$variable.importance[
    order(qrf.elim[[ii]]$variable.importance, decreasing = T) ]

  qrf.elim[[ii+1]] <- ranger(x = d.ph10[, names(t.imp[1:s.seq[ii]])],
                             y = d.ph10$ph.0.10,
                             num.threads = 1, # set the number of CPUs
                             importance = "permutation")
  oob.mse[[ii+1]] <- qrf.elim[[ii+1]]$prediction.error

}

# Prepare a data frame for plot
elim.oob <- data.frame(elim.n = c(length(l.covar), s.seq[1:length(s.seq)]),
                       elim.OOBe = unlist(oob.mse) )
```

Please continue:

- Optimize m_{try} before you start the covariate selection (function `train`, package `caret`). How much does the OOB error decrease? Are both steps (tuning, selection) worth the effort from a point of view of prediction performance?
- Implement the same covariate selection for gradient boosting with trees as baselearners

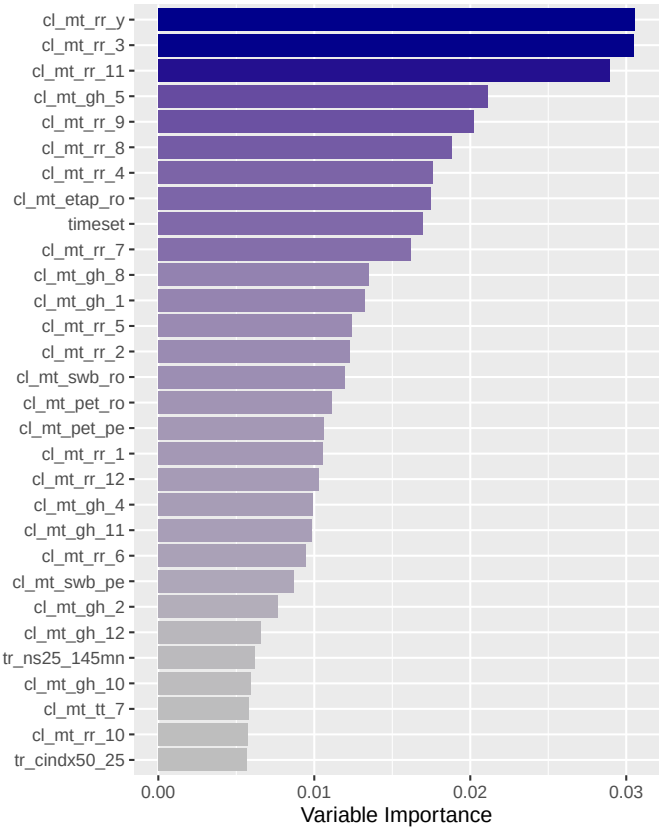

Figure 2: Covariate importance of 30 most important covariates for topsoil pH (before selection).

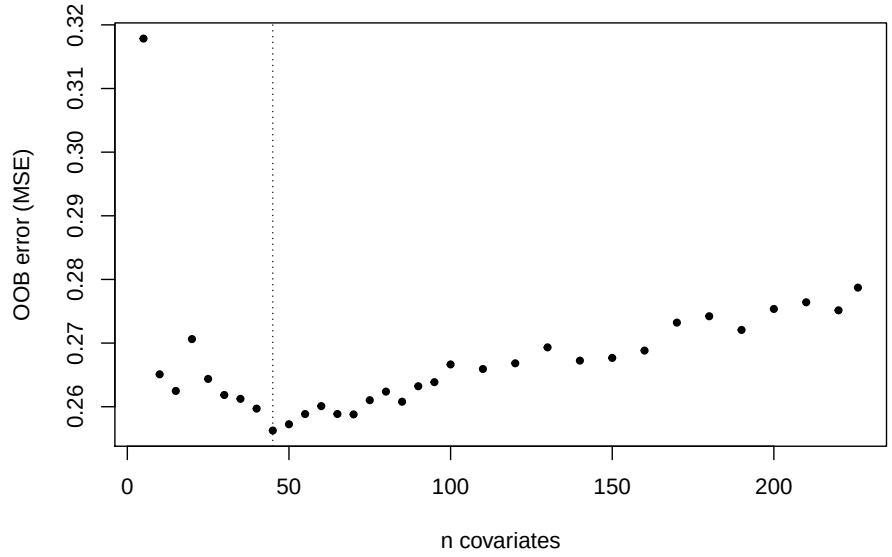

Figure 3: Path of out-of-bag mean squared error as covariates are removed. Minimum is found at 55 covariates.

(package `gbm` or `caret`). Do you find the same covariates in the final set? Why do you expect differences?

2 Model interpretation

2.1 Partial resiudal plots

To demonstrate the principle we create a partial residual plot for an ordinary least squares fit. Then, we add the same plot for the lasso fitted on the topsoil pH data above:

```
# create a linear model (example, with covariates from lasso)
ols <- lm( ph.0.10 ~ timeset + ge_geo500h3id + cl_mt_gh_4 +
             tr_se_curvplan2m_std_25c, data = d.ph10 )
par(mfrow = c(1,2)) # two plots on same figure
# residual plot for covariate cl_mt_gh_4
termplot(ols, partial.resid = T, terms = "cl_mt_gh_4",
         ylim = c(-2,2),
         main = "Ordinary Least Squares")
abline(h=0, lty = 2)

## Create partial residual plot for lasso
# there is no direct function available, but we can easily
# construct the plot with
# y-axis: residuals + effect of term (XBi), scaled
# x-axis: values covariate
# regression line: model fit of axis y~x

## First setup and fit the model
l.factors <- names(d.ph10[l.covar])[
```

```
  t.f <- unlist( lapply(d.ph10[l.covar], is.factor) ) ]
l.numeric <-  names(t.f[ !t.f ])
# create a vector that labels the groups with the same number
g.groups <- c( 1:length(l.numeric),
               unlist(
                 sapply(1:length(l.factors), function(n){
                   rep(n+length(l.numeric), nlevels(d.ph10[, l.factors[n]])-1)
                 })
               )
)
# grpreg needs model matrix as input
XX <- model.matrix( ~., d.ph10[, c(l.numeric, l.factors), F])[,-1]
# cross validation (CV) to find lambda
ph.cvfit <- cv.grpreg(X = XX, y = d.ph10$ph.0.10,
                      group = g.groups,
                      penalty = "grLasso",
                      returnY = T) # access CV results
# choose optimal lambda: CV minimum error + 1 SE (see glmnet)
l.se <- ph.cvfit$cvse[ ph.cvfit$min ] + ph.cvfit$cve[ ph.cvfit$min ]
idx.se <- min( which( ph.cvfit$cve < l.se ) ) - 1

# get the non-zero coefficients:
t.coef <- ph.cvfit$fit$beta[, idx.se ]

# get the index of the covariate
idx <- which( names(t.coef) == "cl_mt_gh_4" )

# residuals of lasso model chosen above
residuals <- d.ph10$ph.0.10 - ph.cvfit$Y[,idx.se]
# prediction for this covariate XBi
Xbeta <- ph.cvfit$fit$beta[idx, idx.se] * d.ph10$cl_mt_gh_4
# calculate partial residuals and center with mean
part.resid <- scale(residuals + Xbeta, scale = F)[,1]

# plot with similar settings
plot(d.ph10$cl_mt_gh_4,
     part.resid, pch = 1, col = "grey",
     ylim = c(-2,2),
     ylab = "partial residuals [%]", xlab = "cl_mt_gh_4",
     main = "Lasso")
abline(lm(part.resid ~ d.ph10$cl_mt_gh_4), col = "red")
abline(h=0, lty = 2)
```

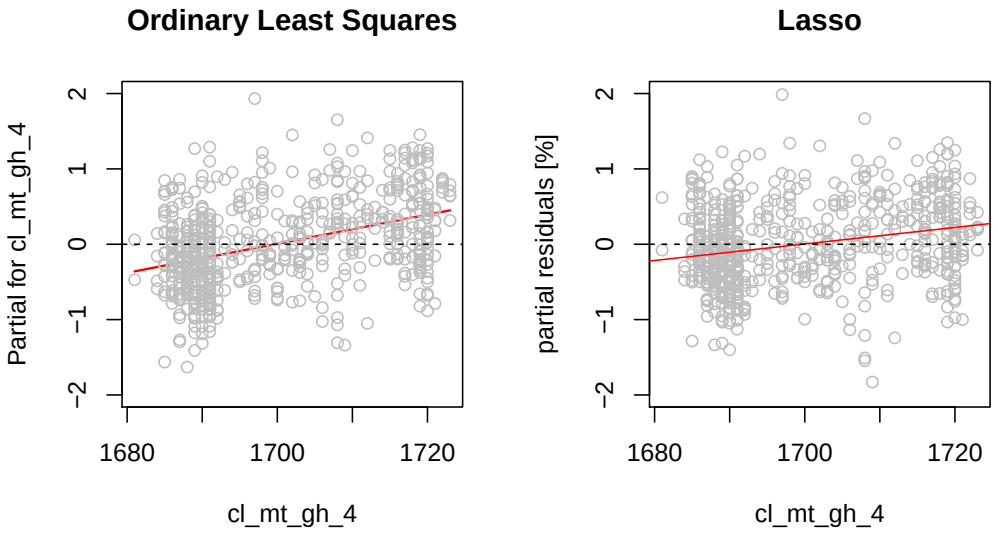

Figure 4: Partial residual plots for a climate covariate in the ordinary least squares fit and the lasso.

2.2 Partial dependence plots

Interpretation of the most important covariates of a random forest model can be done by partial dependence plots. But keep in mind that the remaining covariates after model selection might be still multi-collinear, hence covariates might be exchangeable.

Partial dependence plots are a general method and can be also applied to other supervised machine learning methods.

```
# select the model with minimum OOB error
rf.selected <- qrf.elim[[ which.min(elim.oob$elim.OOBe)]]

t.imp <- rf.selected$variable.importance[
  order(rf.selected$variable.importance, decreasing = T)]

# 6 most important covariates
t.6 <- names( t.imp[ 1:6 ] )

# Create partial dependence plots for the 6 most important covariates
# list with 6 plots
l.plots <- lapply(t.6,
                  function(n.var){
                    plotPartial(partial(rf.selected,
                                        pred.var = n.var,
                                        train = d.ph10))
                    }
                  )
# Create layout for the resulting trellis plots (Package lattice)
grid.arrange(l.plots[[1]], l.plots[[2]], l.plots[[3]],
             l.plots[[4]], l.plots[[5]], l.plots[[6]], ncol = 2)
```

Please continue:

- Create partial dependence plots for the boosted trees model (`?plot.gbm`, `plot(.., i.var = ..)`). Do you find the same relationships?
- What do you conclude from the plots? Are the plotted covariates good predictors?

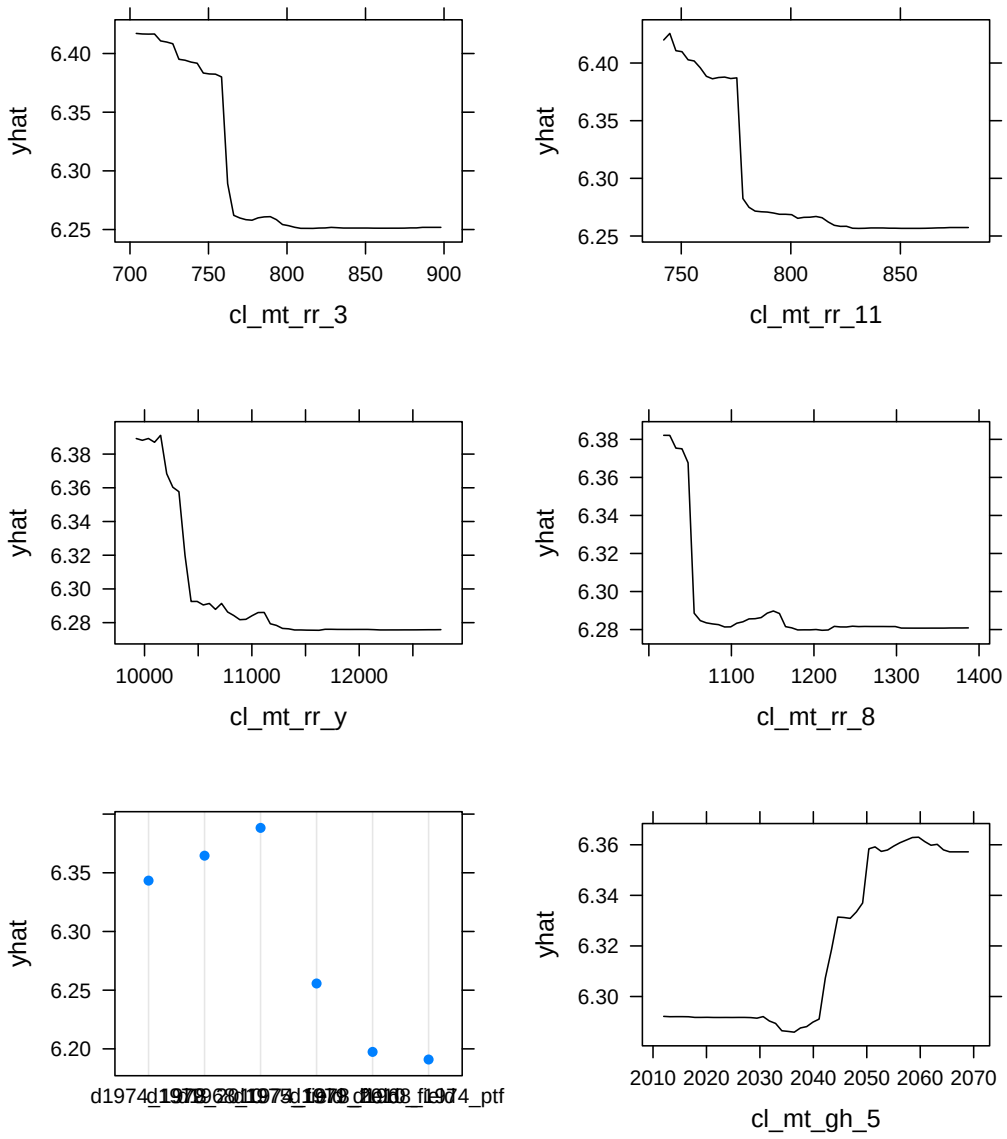

Figure 5: Partial dependence plots for the 4 most important covariates.

3 Prediction uncertainty with quantile regression forest

When reporting predictions it is important to give prediction uncertainty along with them. For any method not yielding uncertainty estimates form the method itself (e.g. kriging variances) a non-parametric or a model-based bootstrap approach can be used. Quantile regression forest computes quantiles of the predictions directly from the bootstrap (bootstrap aggregation, bagging) that is done within random forest.

```
# Fit quantile regression forest
ph.quantRF <- ranger(x = d.ph10[, l.covar[1:30]],
                     y = d.ph10$ph.0.10,
                     quantreg = T)

# select validation data
d.ph10.val <- berne[berne$dataset == "validation" & !is.na(berne$ph.0.10), ]
d.ph10.val <- d.ph10.val[complete.cases(d.ph10.val[l.covar]), ]

# compute predictions (mean) for each validation site
ph.pred <- predict(ph.quantRF, data = d.ph10.val, what = mean)
```

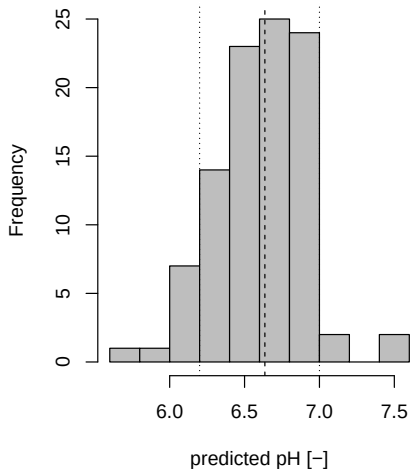

Figure 6: Histogram of predictive distribution for one single prediction point (dotted lines: 90 % prediction interval, dashed line: mean prediction).

Plot for evaluation of prediction intervals (as shown in presentation):

```
# Coverage probabilities plot
# create sequence of nominal probabilities
ss <- seq(0,1,0.01)
# compute coverage for sequence
t.prop.inside <- sapply(ss, function(ii){
  boot.quantile <-  t( apply(ph.pred.distribution$predictions, 1, quantile,
                             probs = c(0,ii) ) )[,2]
  return( sum(boot.quantile <= d.ph10.val$ph.0.10)/nrow(d.ph10.val) )
})
```

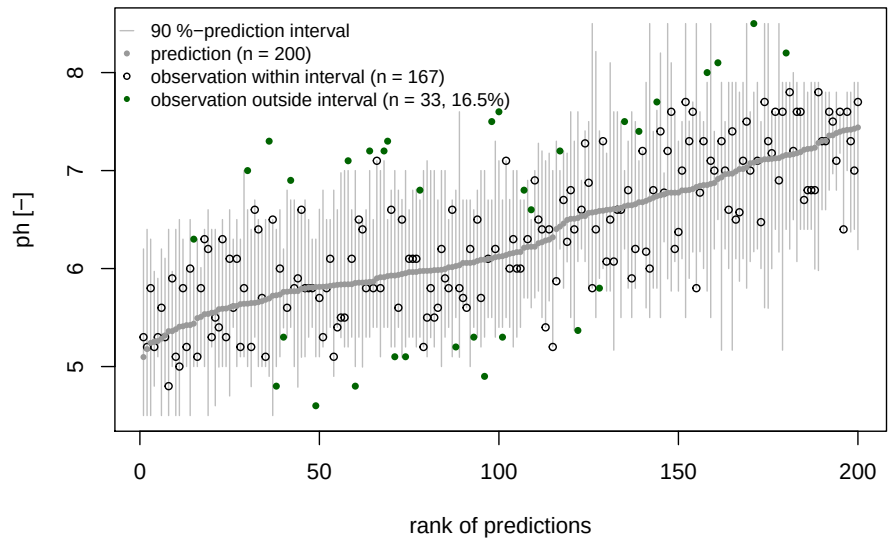

Figure 7: Coverage of 90 %-prediction intervals computed by model-based boostrap.

```
plot(x = ss, y = t.prop.inside[length(ss):1],
     type = "l", asp = 1,
     ylab = "coverage probabilities",
     xlab="nominal probabilities" )
# add 1:1-line
abline(0,1, lty = 2, col = "grey60")
# add lines of the two-sided 90 %-prediction interval
abline(v = c(0.05, 0.95), lty = "dotted", col = "grey20")
```

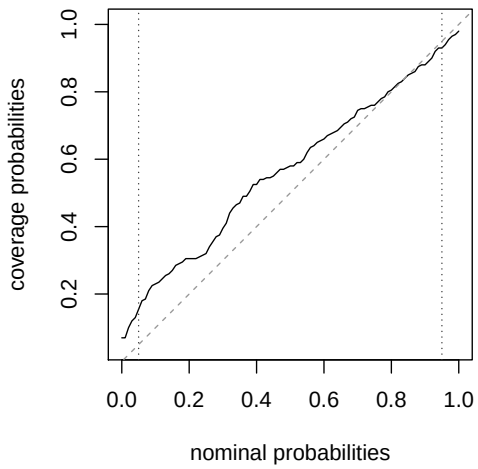

Figure 8: Coverage probabilities of one-sided prediction intervals computed for the validation data set of topsoil pH of the Berne study area.

Please continue:

- Are you satisfied with the prediction intervals?
- Create maps of the prediction intervals for the `berne.grid` data. Is there much spatial structure in the uncertainty?

R session information

This document was generated with:

```
toLatex(sessionInfo(), locale = FALSE)
```

- R version 3.6.3 (2020-02-29), `x86_64-pc-linux-gnu`
- Running under: `Progress Linux 5+ (engywuck-backports)`
- Matrix products: default
- BLAS: `/usr/lib/x86_64-linux-gnu/openblas/libblas.so.3`
- LAPACK: `/usr/lib/x86_64-linux-gnu/libopenblasp-r0.3.5.so`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: caret 6.0-86, gbm 2.1.8, geoGAM 0.1-2, ggplot2 3.3.2, glmnet 4.0-2, grpreg 3.3.0, kernlab 0.9-29, knitr 1.29, lattice 0.20-41, Matrix 1.2-18, mboost 2.9-3, pdp 0.7.0, quantregForest 1.3-7, randomForest 4.6-14, ranger 0.12.1, raster 3.3-13, RColorBrewer 1.1-2, sp 1.4-2, stabs 0.6-3
- Loaded via a namespace (and not attached): class 7.3-17, codetools 0.2-16, colorspace 1.4-1, compiler 3.6.3, crayon 1.3.4, data.table 1.13.0, digest 0.6.25, dplyr 1.0.2, ellipsis 0.3.1, evaluate 0.14, farver 2.0.3, foreach 1.5.0, Formula 1.2-3, generics 0.0.2, glue 1.4.1, gower 0.2.2, grid 3.6.3, gridExtra 2.3, gtable 0.3.0, highr 0.8, inum 1.0-1, ipred 0.9-9, iterators 1.0.12, labeling 0.3, lava 1.6.7, libcoin 1.0-6, lifecycle 0.2.0, lubridate 1.7.9, magrittr 1.5, MASS 7.3-52, mgcv 1.8-31, ModelMetrics 1.2.2.2, munsell 0.5.0, mvtnorm 1.1-1, nlme 3.1-148, nnet 7.3-14, nnls 1.4, partykit 1.2-9, pillar 1.4.6, pkgconfig 2.0.3, plyr 1.8.6, pROC 1.16.2, prodlim 2019.11.13, purrr 0.3.4, quadprog 1.5-8, R6 2.4.1, Rcpp 1.0.5, recipes 0.1.13, reshape2 1.4.4, rlang 0.4.7, rpart 4.1-15, scales 1.1.1, shape 1.4.4, splines 3.6.3, stats4 3.6.3, stringi 1.4.6, stringr 1.4.0, survival 3.2-3, tibble 3.0.3, tidyselect 1.1.0, timeDate 3043.102, tools 3.6.3, vctrs 0.3.2, withr 2.2.0, xfun 0.16