Caret bigdata demo

Gertjan Verhoeven

5 juni 2018

Summary

With big data sets and relatively simple models, we can have (much) more data than we actually need to get stable estimates. The goal of this notebook is to demonstrate how one would determine, using the `caret` package, what the required minimum sample size is to get stable estimates. Using such a curve, we can determine the optimal trade-off between computation time and estimate accuracy.

We mimick the process of grabbing a new random sample from a big data set by repeatedly generating simulated data from the same Data generating process. We pick a sequence of sample sizes, starting out small (say a 100 datapoints) and each time roughly doubling the amount.

For each sample size, we grab 30 samples from our big dataset (i.e. we simulate 30 datasets). We treat each dataset as if it was our only dataset for the analysis task at hand. For a prediction problem, we would typically select a statistical learning algorithm, and tune it using cross-validation. For this step we use the `caret` package. We get an smoothed average test error by doing 5-fold cross validation, repeated 6 times (so repeately partioning the data in 5 folds).

We now have for each sample size a set of 30 smoothed test errors. By examining the variance of these test errors, we get an impression of the variability / uncertainty in the test error estimate. To choose an optimal sample size, for a particalar tolerance level (i.e. we accept 1% variation in the test error estimate) we can simply check at which sample size the variation in the test errors drops below 1% tolerance.

Function to generate simulated clean data

```
rdunif <- function(n,k) sample(1:k, n, replace = T)

# simulate noise dataset with signal on x2 as function of relevance
generateData <- function(nsize, relevance, interaction = 0){
  y <- rbinom(n = nsize, size = 1, prob = 0.5)
  x1 <- rnorm(n = nsize, mean = 0, sd = 1)
  x3 <- rdunif(n = nsize, k = 4)
  x4 <- rdunif(n = nsize, k = 10)
  x5 <- rdunif(n = nsize, k = 20)
  x2 <- rep(-1, nsize)
  x2[y == 1 & x1 < 0] <- rbinom(n = sum(y == 1 & x1 < 0), size = 1, prob = 0.5 - relevance - interactio
  x2[y == 1 & x1 >= 0] <- rbinom(n = sum(y == 1 & x1 >= 0), size = 1, prob = 0.5 - relevance + interacti
  x2[y == 0 & x1 < 0] <- rbinom(n = sum(y == 0 & x1 < 0), size = 1, prob = 0.5 + relevance - interactio
  x2[y == 0 & x1 >= 0] <- rbinom(n = sum(y == 0 & x1 >= 0), size = 1, prob = 0.5 + relevance + interacti

  my_df <- data.frame(y = as.factor(y), x1, x2, x3, x4, x5)
  my_df
}
```

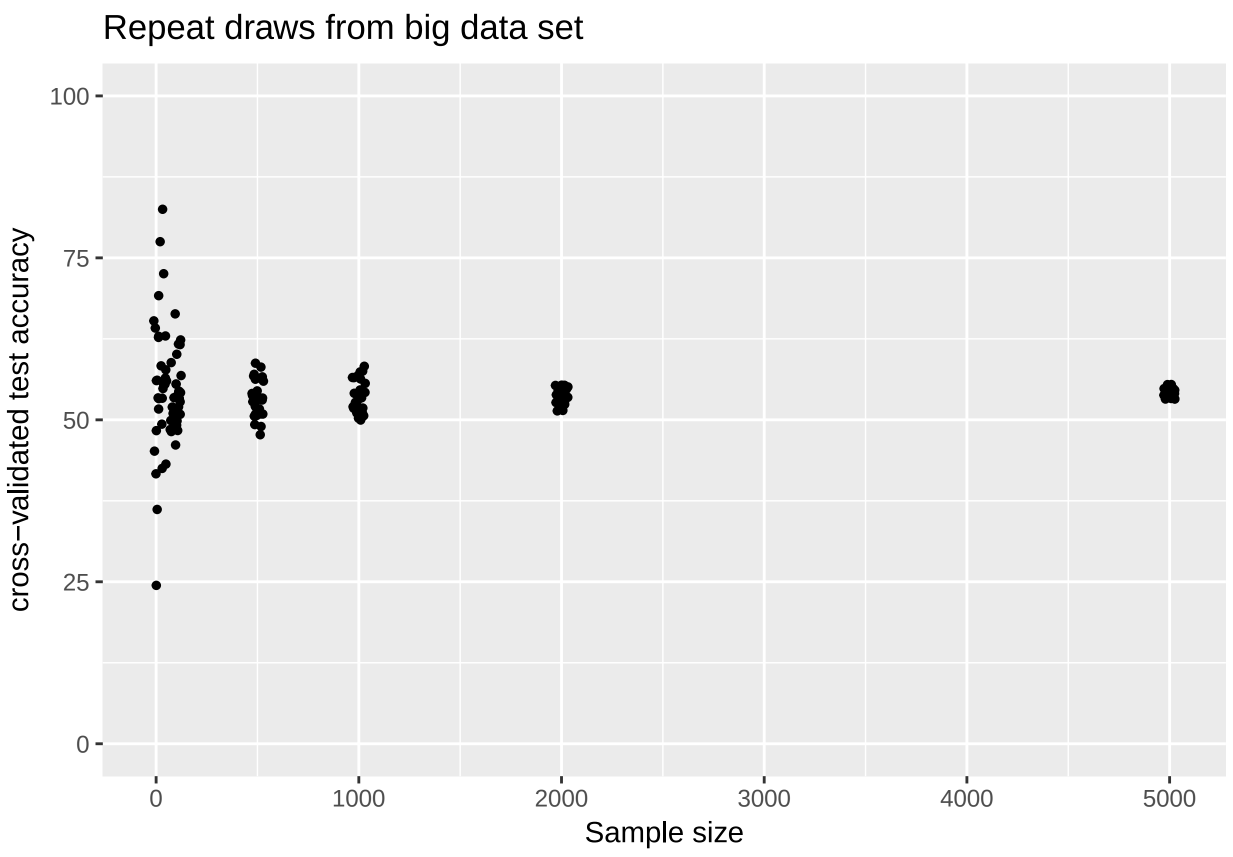
Set train control

```r
train.control <- trainControl(method = "repeatedcv",
                              number = 5, repeats = 6,
                              savePredictions = F)

fullrun <- 1

set.seed(123)

# tuning pars mtry, splitrule, min.node.size (hoger wordt niet beter)
rf.grid <- expand.grid(mtry = 3,
                       splitrule = "gini",
                       min.node.size = 1)
```

Run simulation

```r
fullrun <- 0

sample_size_vec <- c(20, 100, 500, 1000, 2000, 5000)
repeats <- 30

myres <- data.frame()

set.seed(1)
if(fullrun){
  z <- 1
  for(i in 1:length(sample_size_vec)){
      for(j in 1:repeats){
        # grab a new sample from our unlimited big data set
        my_df <- generateData(sample_size_vec[i], 0.1)

        caret_fit <- train(y ~ .,
                              data = my_df,
                              method = "ranger",
                              trControl = train.control,
                              tuneGrid = rf.grid)
        myres[z,1]<- sample_size_vec[i]
        myres[z,2] <- j
        myres[z,3] <- mean(caret_fit$resample$Accuracy)
        z <- z + 1
      }
  }
  colnames(myres) <- c("Sample_size", "repeat", "Mean_accuracy")
  myres <- data.table(myres)
myres <- myres[, Mean_accuracy := Mean_accuracy * 100]
  saveRDS(myres, file = "output/myres.rds")
} else { myres <- readRDS("output/myres.rds") }
```

```r
res <- myres[, .(Min = min(Mean_accuracy),
                 Mean = mean(Mean_accuracy),
                  Max = max(Mean_accuracy)), .(Sample_size)]
res$order <- 1:nrow(res)


gp <- ggplot(myres, aes(x = Sample_size , y = Mean_accuracy)) +
#geom_errorbar(aes(ymin = Min, ymax = Max)) +
geom_point(size = 1, col = "black", position = "jitter") + ylim(0, 100) + ylab("cross-validated test acc
ggtitle("Repeat draws from big data set") #+ geom_vline(xintercept = sample_size_vec, color = "blue")

gp
```


Repeat draws from big data set
cross−validated test accuracy
Sample size