Classification with SEMMS in the large P setting

Haim Bar

March 18, 2021

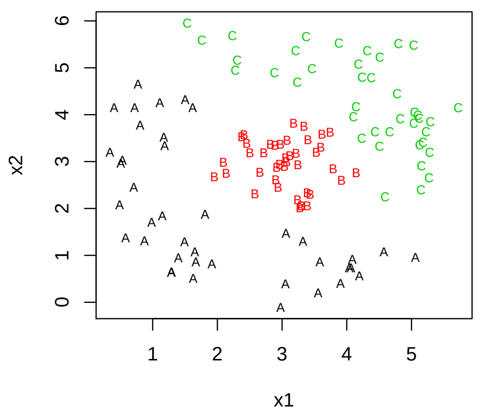
Why pre-screening variables is important in large P classification
Classification into categories is quite challenging when the number of predictors is large. Here, we will use the random forest to perform classification, but the results apply to other classifiers. The main point is, that when the number of predictors is large, it is recommended to perform pre-screening using variable selection, and reduce the number of potential predictors before running the classification step. We will show it through a simple example. For more details, see [1].

Simulated data
We simulate data in the two-dimensional space, such that there are three classes, but they are not linearly separable. Still, two predictors are sufficient determine the class. To make the point of this report, we add 1,998 predictors which have nothing to do with the classification. We generate 40 data points from each class, and we later use half for training and half for testing.

```r
set.seed(3344)
N0 <- 120 # total sample size
S <- matrix(0,2000,2000)
diag(S) <- 0.2 # Used to add i.i.d. Gaussian noise

# Class B is centered around (3,3). Points from classes A and C are centered
# around points drawn from the following sets:
classA <- rbind(c(1,2),c(1,3),c(1,1),c(2,1),c(3,1),c(4,1),c(1,4))
classC <- rbind(c(2,5),c(3,5),c(4,5),c(5,5),c(5,4),c(5,3),c(5,2))
means <- rbind(classA[sample(1:7,40,replace = TRUE),],
               matrix(3,40,2),
               classC[sample(1:7,40,replace = TRUE),])
x <- mvrnorm(N0, rep(0,2000), S)   # create the predictor data
x[,1:2] <- x[,1:2] + means
class0 <- c(rep("A",40), rep("B",40), rep("C",40))
plot(x[,1:2],col=as.factor(class0), pch=class0, cex=0.7, xlab="x1", ylab="x2")
```

The complete data set is displayed in the following figure:



Classification with randomForest - using only the true predictors
First, we classify the data using only the true predictors, x_1 and x_2.

```r
# take a sub-sample with 50% from each class, and use for training the random forest
sset <- sort(c(sample(1:40,20), sample(41:80,20), sample(81:120,20)))
x0 <- x       # the full data set, with 2000 predictors
x <- x[,1:2]  # first, fit RF using only the true predictors
# create the training and testing data sets, using the true predictors only.
df <- data.frame(class0,x)
dftr <- df[sset,]
dfts <- df[-sset,]
plot(dftr[,2:3],col=as.factor(class0[sset]), pch=class0[sset],cex=0.7,xlab="x1",ylab="x2")
## training:
set.seed(99889)
fit.forest <- randomForest(class0 ~ ., data=dftr)
sum(diag(fit.forest$confusion[,-4]))/sum(fit.forest$confusion[,-4])

## testing:
forest.pred <- predict(fit.forest, dfts)
forest.perf <- table(dfts$class, forest.pred, dnn=c("Actual", "Predicted"))
cat("Accuracy:",sum(diag(forest.perf))/sum(forest.perf),"\n")
print(forest.perf)
```

We see in the table below that the classification is very good, with only 3 errors (5% error rate) when using the test data.

```
       Predicted
Actual  A  B  C
     A 20  0  0
     B  0 18  2
     C  1  0 19
```

Classification with randomForest - using all the predictors

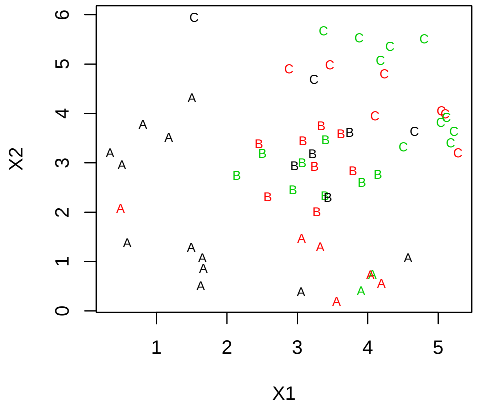
Next, we use randomForest to classify the data with all 2,000 predictors. The error rate increases to 50% !

```r
# Next, use all 2000 predictors
df2 <- data.frame(class0, x0)
dftr <- df2[sset,]
dfts <- df2[-sset,]
## training:
set.seed(99889)
fit.forest <- randomForest(class0 ~., data=dftr)
sum(diag(fit.forest$confusion[,-4]))/sum(fit.forest$confusion[,-4])

## testing:
forest.pred <- predict(fit.forest, dfts)
forest.perf <- table(dfts$class, forest.pred, dnn=c("Actual", "Predicted"))
cat("RF, 2000 predictors:",sum(diag(forest.perf))/sum(forest.perf),"\n")
print(forest.perf)
```

```
      Predicted
Actual  A  B  C
     A 12  6  2
     B  4  8  8
     C  3  7 10
```

In the following plot the true classification is shown by the character (A,B,C), and the classification from the random forest algorithm is represented by the color (black=A, red=B, green=C). The classification is very bad, indeed.



Pre-screening - variable selection with SEMMS

We perform variable selection using SEMMS [2] for three binary outcome models: A vs. not A, B vs. not B, and C vs. not C. We collect all the predictors found to be relevant in any of these models.

```
# perform pre-screening, using SEMMS
slct <- c()
for (ch in LETTERS[1:3]) {
  Y <- ifelse(dftr[,1] == ch, 1, 0)
  Z <- as.matrix(dftr[,-1])
  forSEMMS <- data.frame(Y,Z)
  save(forSEMMS, file="tmp/forSEMMS.RData")
  dataYXZ <- readInputFile("tmp/forSEMMS.RData", ycol=1, Zcols=2:2001)
  fittedSEMMS <- fitSEMMS(dataYXZ, mincor=0.8, nn=5, minchange= 1,
  ↪  distribution="B",verbose=T,rnd=F)
  slct <- c(slct, fittedSEMMS$gam.out$nn)
}
cat("selected ", ncol(x1), "variables:",unique(sort(slct)),"\n")
```

Eight variables are selected: 1 and 2 (the true predictors), and six false positives (284, 561, 878, 1052, 1407, and 1685). For the model B vs. not B, only false positive are found. In this simulation scenario, this is not very surprising since the separation between the two groups is far from linear, and the binary classification in SEMMS relies on logistic regression. Also, the size of the training data set is rather small. Still, overall the classification after pre-screening the predictors is excellent (95%) and the only three errors are indeed when the true class is B.

```
       Predicted
Actual  A  B  C
     A 20  0  0
     B  2 17  1
     C  0  0 20
```

References

[1] H. Bar, J. Booth, M. T. Wells, and K. Liu. Facilitating high-dimensional transparent classification via empirical Bayes variable selection. *Applied Stochastic Models in Business and Industry*, 34(6):949–961, 2018.

[2] Bar, H. Y., Booth, J. G., and Wells, M. T. (2020). A Scalable Empirical Bayes Approach to Variable Selection in Generalized Linear Models. *Journal of Computational and Graphical Statistics*, **0**(0), 1–12.