Sketch of the solutions of STK2100 exam (VÅR 2020)

Riccardo De Bin

June 30, 2020

Problem 1: Golub et al.(1999) data

We start by downloading the data from the provided url:

```
rm(list=ls())
leukemia <- read.csv("http://web.stanford.edu/~hastie/CASI_files/DATA/leukemia_big.csv")
```

Since the data are in "bioinformatics" format, we need to transpose the dataset to have the observations in the rows and the variables in the columns

```
X <- t(leukemia)
```

The response variable can be derived from the columns' names,

```
y <- ifelse(substr(colnames(leukemia), 2, 2) == 'L', 0, 1)
```

Finally, the dataset must be split into a training and a test set,

```
test <- c('ALL.4', 'ALL.8', 'ALL.10', 'ALL.11', 'ALL.13', 'ALL.18', 'AML', 'AML.1',
          'AML.4', 'AML.6', 'AML.8', 'ALL.23', 'ALL.26', 'ALL.29', 'ALL.31', 'ALL.32',
          'ALL.35', 'ALL.39', 'ALL.40', 'ALL.41', 'ALL.42', 'AML.16', 'AML.22', 'AML.24')
test <- sapply(test, function(x, dataset) which(x == rownames(dataset)), dataset = X)

X.train <- X[-test, ]
y.train <- y[-test]
X.test <- X[test, ]
y.test <- y[test]
```

a Penalized logistic regression

a.1

The exercise clearly required to use a deterministic cross-validation procedure, for example LOOCV, to choose among the 10 possible values of λ provided in the text,

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 3.0-2
```

```
lambda.seq <- exp(seq(-7,2))
lambda.ridge <- cv.glmnet(x = X.train, y = y.train, nfolds = 48, alpha = 0,
                          family = 'binomial', lambda = lambda.seq, grouped = FALSE)
lambda.lasso <- cv.glmnet(x = X.train, y = y.train, nfolds = 48, alpha = 1,
                          family = 'binomial', lambda = lambda.seq, grouped = FALSE)
```

The correct λs are

```
lambda.ridge$lambda.min
```

```
## [1] 0.01831564
```

for ridge regresison and

```
lambda.lasso$lambda.min
```

```
## [1] 0.01831564
```

for lasso.

Alternatively, the largest values within 1 standard error distance could have been selected,

```
lambda.ridge$lambda.1se
```

```
## [1] 2.718282
```

```
lambda.lasso$lambda.1se
```

```
## [1] 0.04978707
```

In penalized regression methods like lasso and ridge, the parameter λ is the tuning parameter that controls the model complexity by controlling the amount of penalty to add to the loss function. In particular, it controls the bias-variance trade-off: a larger value of λ leads to a solution closer to the null model (only intercept), so increases the bias and reduces the variance, while smaller values have the opposite effect (less bias, more variance), until, in a low dimensional case, $\lambda = 0$ leads to the same solution of the logistic regression.

Note: tuning λ to minimize the cross-validated missclassification error instead of the cross-validated deviance (i.e., setting `type.measure = 'class'` in `cv.glmnet`) would have lead to slightly different results, obviously accepted.

a.2

The exercise requires the computation of the missclassification error for both models on the training,

```
mod.ridge <- glmnet(x = X.train, y = y.train, alpha = 0, family = 'binomial',
                    lambda = lambda.seq)
mod.lasso <- glmnet(x = X.train, y = y.train, alpha = 1, family = 'binomial',
                    lambda = lambda.seq)

y.tr.ridge.class <- predict(mod.ridge, s = lambda.ridge$lambda.min, newx = X.train,
                            type = 'class')
y.tr.lasso.class <- predict(mod.lasso, s = lambda.lasso$lambda.min, newx = X.train,
                            type = 'class')

c(mean(y.tr.lasso.class != y.train), mean(y.tr.ridge.class != y.train))
```

```
## [1] 0 0
```

and the test set,

```
y.hat.ridge.class <- predict(mod.ridge, s = lambda.ridge$lambda.min, newx = X.test,
                             type = 'class')
y.hat.lasso.class <- predict(mod.lasso, s = lambda.lasso$lambda.min, newx = X.test,
                             type = 'class')

c(mean(y.hat.lasso.class != y.test), mean(y.hat.ridge.class != y.test))
```

```
## [1] 0.00000000 0.04166667
```

The training error is in general smaller than the test error because the model is fitted on those specific data, so in the case the model does not only capture the systematic part of the process, but also part of the randomness, we have an eccess of optimism due to overfitting.

a.3

We have chosen leave-one-out cross-validation (LOOCV) because it is the fastest procedure among the deterministic cross-validation procedures (alternatively, a complete K-fold cross validation could have been used, but it would have implied to consider all the $\binom{48}{K}$ possible splits). LOOCV is deterministic because all observations are in separate folds and there is only one way to assign them (basically, $\binom{48}{K} = 1$).

Advantages of LOOCV with respect to other cross-validation procedures include no randomness and a lower bias; disadvantages include heavier computations (a part for special cases) and an higher variance.

b Model assessment

b1

As the lift plot and the missclassification error focus on different aspects of the model, it is not strange that the former gives the same result for lasso and ridge, while the latter does not. In particular, the lift plot sorts the observations based on the estimated probability to belong to one or the other class: in this specific example, observation `AML.8` is incorrectly classified by the ridge model as `ALL`,

```
predict(mod.ridge, s = lambda.ridge$lambda.min, newx = X.test,
        type = 'class')[rownames(X.test)=='AML.8']
```

```
## [1] "0"
```

but if we look at the estimated probability,

```
round(predict(mod.ridge, s = lambda.ridge$lambda.min, newx = X.test, type = 'response'),
      2)[rownames(X.test)=='AML.8']
```

```
## [1] 0.49
```

it is very close to the threshold 0.50. Basically, in the missclassification error this observation makes the error be 1/24, while it does not harm the lift curve as its probability to be `ALL` is higher than any of the `AML` observations.

b2

That point says that if we choose (with the goal of selecting `ALL` observations) the 25% of the observations to which the models assign the highest probability to be `ALL`, our selection is 3 time better than if we choose them completely at random.

c Pre-selection

c1

The exercise asks to perform a two-sample t-test on the whole dataset for each variable and to select those with the smallest p-values, i.e., those with the highest difference in mean. The result is

```
pval <- apply(X, 2, function(x, class) t.test(x[class == 0], x[class == 1])$p.value,
              class = y)
selected.all <- order(pval)[1:9]
sort(selected.all)
```

```
## [1]  758 1745 1882 2121 3252 4377 4847 6041 6854
```

Once these variables have been identified, a logistic model can be fitted,

```
mod.logit.all <- glm(y.train ~ ., data = as.data.frame(X.train[, selected.all]),
                     family = binomial)
```

and the misclassification error computed,

```
y.hat.logit.all <- predict(mod.logit.all, newdata = as.data.frame(X.test[, selected.all]),
                           type = 'response')
mean(round(y.hat.logit.all) != y.test)
```

```
## [1] 0
```

The pre-selection step is necessary to fit the logistic model, as the number of variables must be smaller than the number of observations.

Note: due to complete separation, here one may have experienced some issues with the model convergence. Nevertheless, the key point was the correct implementation of the procedure, rather than the results themselves (that were not asked to be commented).

c2

The procedure is wrong because we are using the test data in the training of the model, so the error cannot be compared to those obtained before for lasso and ridge regression, in which the models are correctly trained only using the traing set.

c3

Here to correct the procedure means to perform the pre-selection step on the training data only, obtaining

```
pval <- apply(X.train, 2, function(x, class) t.test(x[class == 0], x[class == 1])$p.value,
              class = y.train)
selected.sep <- order(pval)[1:9]
sort(selected.sep)
```

```
## [1] 1745 1882 2121 3252 4847 4973 6041 6623 6854
```

The logistic model is now fitted using vairables selected "fairly",

```
mod.logit.sep <- glm(y.train ~ ., data = as.data.frame(X.train[, selected.sep]),
                     family = binomial)
y.hat.logit.sep <- predict(mod.logit.sep, newdata = as.data.frame(X.test[, selected.sep]),
                           type = 'response')
```

and the missclassification error for this procedure,

```
mean(round(y.hat.logit.sep) != y.test)
```

```
## [1] 0.08333333
```

can be compared to those of lasso and ridge regression found at point **a.2**.

d Non-hierarchical clustering

d1

Implementing K-means with the required initial centroids,

```
missclassication.kmeans <- rep(NA, 10)
centers <- list(rbind(X[1,], X[11,]),
                rbind(X[1,], X[12,]),
                rbind(X[1,], X[21,]),
```

```
                rbind(X[1,], X[23,]),
                rbind(X[11,], X[12,]),
                rbind(X[11,], X[21,]),
                rbind(X[11,], X[23,]),
                rbind(X[12,], X[21,]),
                rbind(X[12,], X[23,]),
                rbind(X[21,], X[23,]))

for (i in 1:10)
{
  cluster.kmeans <- kmeans(X, 2, centers = centers[[i]])
  table.tmp <- table(y, cluster.kmeans$cluster)
  missclassication.kmeans[i] <- min(sum(diag(table.tmp))/sum(table.tmp),
                                    1 - sum(diag(table.tmp))/sum(table.tmp))
}
```

leads to the following results,

```
missclassication.kmeans
```

```
##  [1] 0.33333333 0.29166667 0.01388889 0.27777778 0.34722222 0.01388889
##  [7] 0.27777778 0.01388889 0.01388889 0.01388889
```

Note that the correct solution must contain only values smaller than 0.5: the clustering procedure only separates the observations in two groups, so the labels have nothing to do with the AML/ALL ones.

The results are so different because K-means is very sensitive to the starting point (initial values of the centroids): there is no guarantee that the method finds the global minimum, and, in contrast, as clearly shown here, often stops at a local minimum.

d2

The problem is that we usually do not know the correct number of cluster in advance, and K-means is not able to find it automatically. Without a previous knowledge of the data structure, is very hard to initialize K correctly.

In this specific case, nevertheless, we know that there are two diseases in the collected data, so we can set $K = 2$.

Cross-validation is not a reasonable option in unsupervised learning problems, as we do not know the true outcome, so we cannot have a measure of the error.

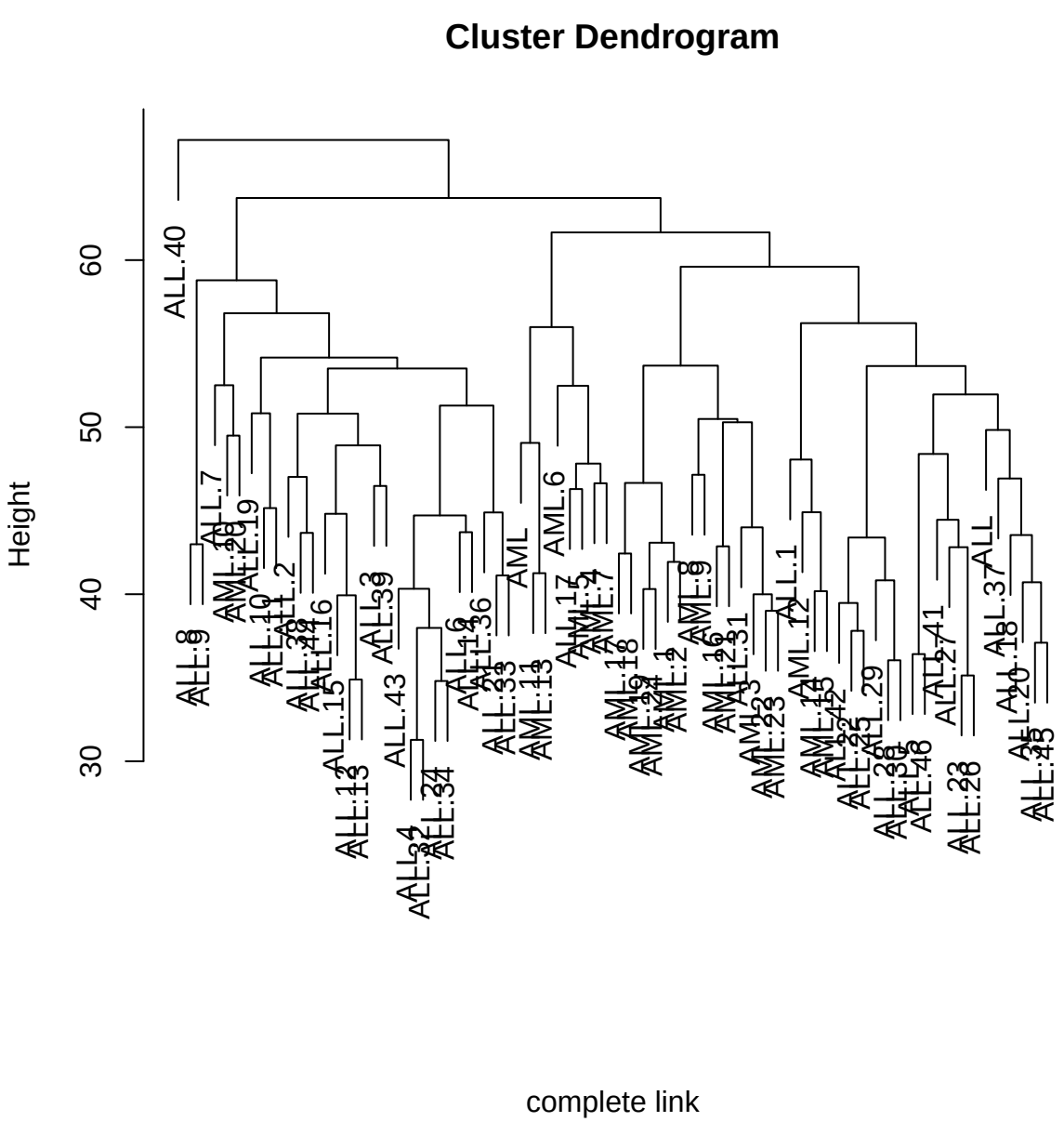
A possible way to solve the problem of selecting the right K is to implement the so called "elbow method": K-means is run with an increasing number of clusters, which leads to a decreasing deviance within the groups. Initially the increase in the number of clusters leads to a substantial decrease in the within deviance, but at a certain point this decrease will start to be small, not justifing an increase of the number of cluster. That point corresponds to the selected K. A possible alternative is to use the GAP statistic.

e Hierarchical clustering

e1

Using an agglomerative strategy and a complete link, a hierarchical clustering method provides this dendrogram,

```
d0 <- dist(X)
h1 <- hclust(d0, method = "complete")
plot(h1, xlab = '', sub = 'complete link')
```



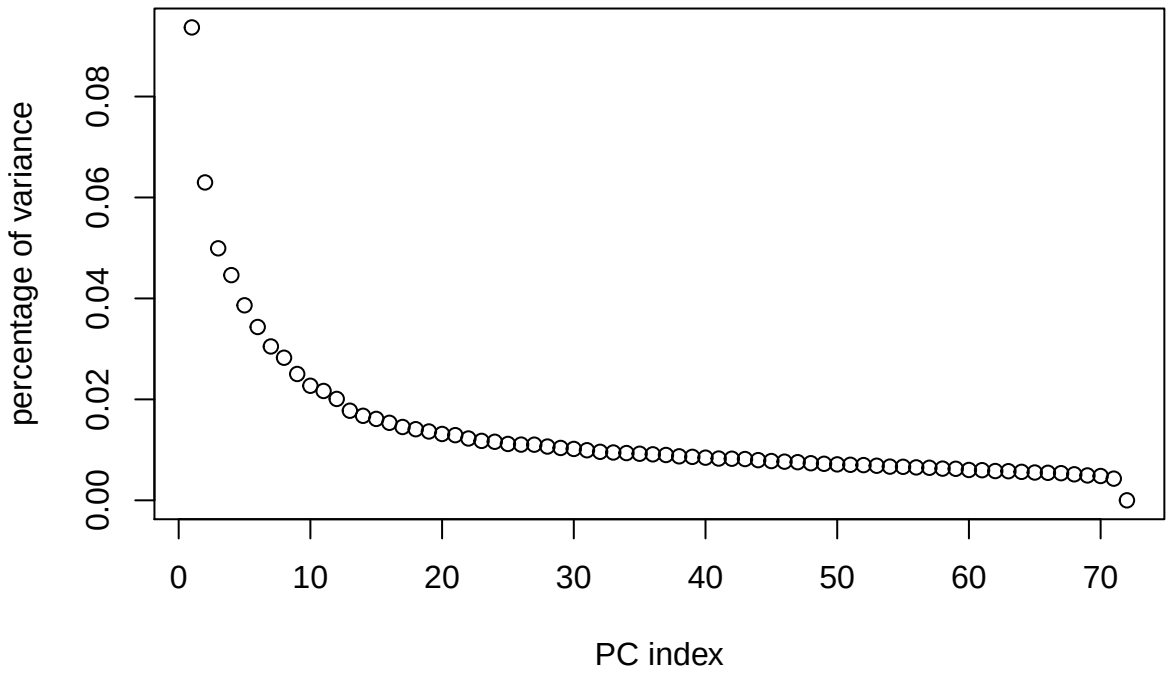
Things to notice here:

- there is a group formed by only one observation, which seems an outlier. An automatic division with the "correct" $K = 2$ would provide a very bad result;
- even not considering this specifc point, the observations are not well separated, and one must select many clusters to have a good separation between AML and ALL observations;
- if we did not know the true K, it would be hard to select it: the decrease in deviance is relatively small for each increase of the number of clusters;
- it seems that we are experiencing the so called "curse of dimensionality".

e2

Here only a plot was required, with the percentage of original variance "contained" in each principal components,

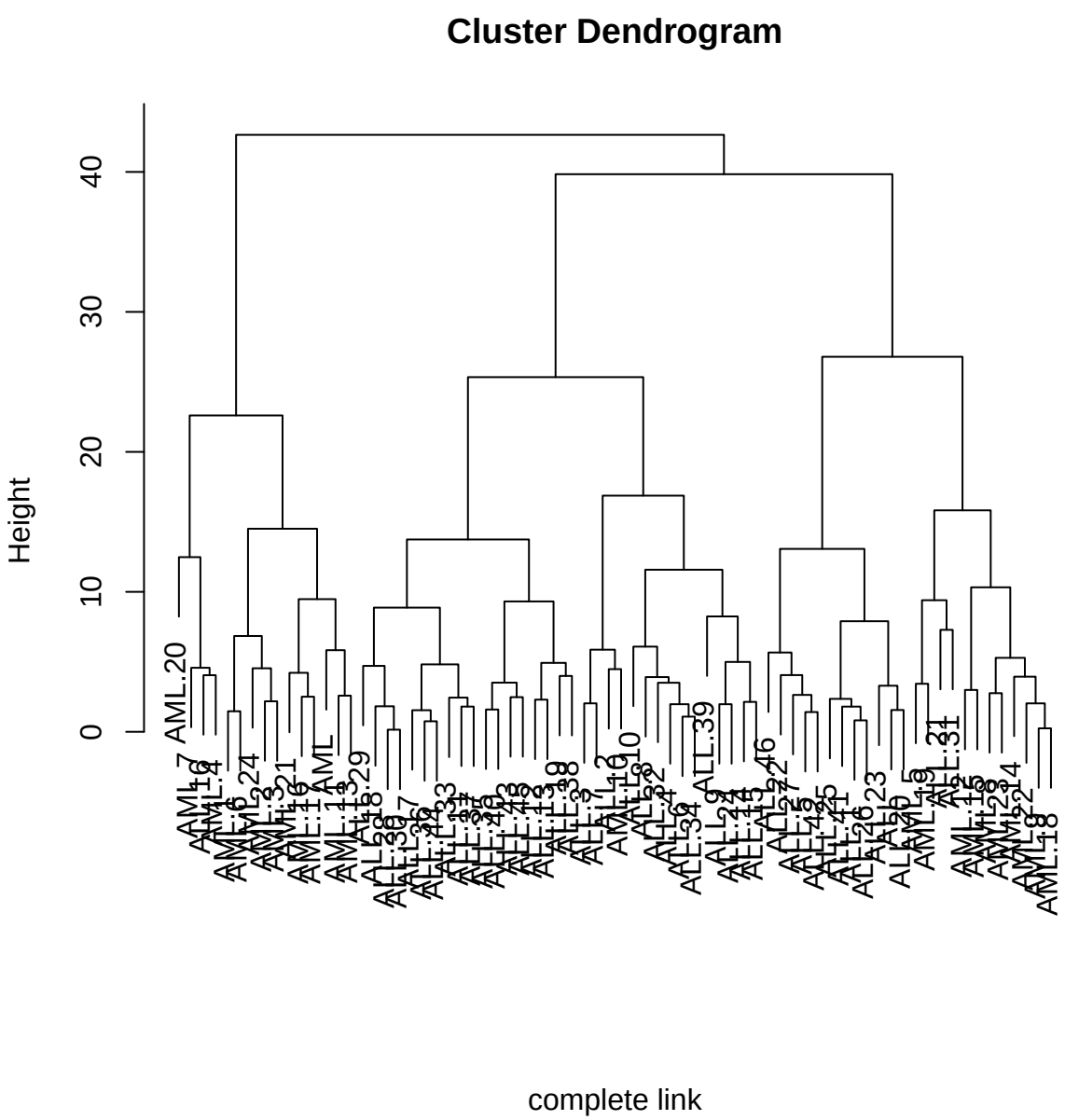
```
principal.comp <- prcomp(X)
plot(principal.comp$sdev^2/sum(principal.comp$sdev^2), xlab = 'PC index',
      ylab = 'percentage of variance')
```



e3

Finally, applying the hierarchical clustering algorithm to the first two principal components only, leads to the following dendrogram,

```
d2 <- dist(principal.comp$x[, 1:2])
h2 <- hclust(d2, method = "complete")
plot(h2, xlab = '', sub = 'complete link')
```



Things to notice here:

- using the "correct" $K = 2$, we would have a relatively good result (the left cluster including almost only AML observations, the right one almost all the ALL);
- already with $K = 4$ the split is extremely good;
- using an heuristic procedure, we would choose $K = 3$, very close to the true value (2) and to a value (the aforementioned 4) that provides a very good separation. With $K = 3$ we would have a group clearly AML (the left one), one clearly ALL (the central one) and one with mixed type of observations (the right one), that could be in turn easily split in two separate subclusters;
- the improvement with respect to the dendrogam of point $e.1$ is substantial.

Note: both in points **e.1** and **e.3** the privided interpretation is one out of many reasonable ones, as long as the discussion was reasonable and supported by evidence, the answer was valid.