Exercise no 10

Corentin Lacroix 1812554

MTH 6312: Méthodes Statistiques D'Apprentissage

November 20, 2015

Exercise 1 Plot these two functions.

Solution 1 Plots are in annex.

R-code :

```
> plot(seq(0.01, 0.99, 0.01), sapply(seq(0.01, 0.99, 0.01), g), xlab="x",
 ylab="y", type="lines", col="blue")
> lines(seq(0.01, 0.99, 0.01), sapply(seq(0.01, 0.99, 0.01), f), col="orange")
> legend(0.9, 0.65, c("g", "f"), lty=c(1,1), lwd=c(2.5,2.5),
 col=c("blue", "orange"))
```

Exercise 2 Argue how these two functions are related to the concept of information and and the concept of entropy.

Solution 2 The function f is clearly related to the entropy defined in probability theory by Shannon for a random variable X. For a discrete random variable taking values $x_1, \cdots, x_n$, the entropy is :

$$\mathrm{H}(X) = \mathrm{E}(-\log(X)) = -\sum_i P(x_i)\log(x_i)$$

For a binary discrete random variable, if x denotes one of the 2 probability associated to this random variable, the entropy of the variable is exactly the g function. In probability, the more the entropy of a random variable is "high", the more its behaviour is "random" and unpredictible. At the contrary a deterministic system has a entropy of zero. Hence, here for ou 2 values discrete random variable, the entropy gets to its max value when probabilities of observing values are the same ($\frac{1}{2}$).

For a discrete random variable with observations probabilites being p_i, the quantity $\sum_{k' \neq k} p'_k p_k$ (deviance) gives the same kind of indication about the random variable than the entropy (predictability of the variable/system). Its maximum value is reached for all discrete probabilites being the same. This quantity is $2 \ \times \ f$ for a binary variable.

For growing trees, when we want to find a splitting variable and a split point, these two functions applied to the 2 new obtained nodes give us indication about the amount of information gained by the process of spliting a node.

Exercise 3 Write the Tylor expansion of $f(x)$ up to a quadratic term.

Solution 3 I don't really get here we have to do here. First, around which points must we find a Taylor expansion ?

For $x = \frac{1}{2}$, the second derivative of the function f is null, so we get $f(x) = f(\frac{1}{2}) + (x - \frac{1}{2}) + o_{\frac{1}{2}}((x - \frac{1}{2})^2)$.

Hence, f has the same curve than g around $x = \frac{1}{2}$ because second derivative of g is also 0 for this point. And both functions can be approximated by straight lines and not quadratic functions..

Moreover, as $\dfrac{df}{dx} = log(\frac{1-x}{x})$, g doesn't have Taylor expansion around 0 and 1 (unbound limits for its first derivative around these points). Thus, it doesn't allow us to say that f and g have same curvatures around these points..

Exercise 4 Find $\lim\limits_{x \to 0} x \log(x)$. How does this limit help to define $f(x)$ over $x \in [0, 1]$.

Solution 4 This limit is clearly 0 (we can prove it by the Hospital Rule), which makes an extension of f possible in 0 and 1. Limits of f in 0 and 1 are both 0 knowing this limit.

Exercise 5 Look at Figure 3.10 in page 70 of the ESL. Write a code that finds the appropriate penalization constant for the lasso over the prostate cancer data using : - one leave out cross validation

Solution 5 Here is a function making one-leave-out validation for Lasso regresion (using glmnet() R function) :

```
oneLeaveOut <- function(X, Y, lambdaSeq) {
    errVector <- rep(0, length(lambdaSeq))
    for(k in 1:nrow(X)){
        Xtemp <- X[-k,]
        Ytemp <- Y[-k,]
        fit <-glmnet(as.matrix(Xtemp), Ytemp, alpha = 1,
                     lambda=lambdaSeq, standardize=TRUE)
        err <- (as.numeric(predict(fit, as.matrix(X[k,])))
                           - rep(as.numeric(Y[k,]), length(lambdaSeq)))
        errVector <- errVector + err*err
    }
    return(errVector/nrow(X))
}
```

And a function giving the best regularization parameter found by Leave-One-Out :

```
oneLeaveOutBestLambda <- function(X, Y , lambdaSeq) {
    lambdaSeq[which.min(oneLeaveOut(X, Y, lambdaSeq))]
}
```

Here, as we have passed a lambda sequence to the glmnet() function, predict() function used with our Lasso fit outputs a vector of size $|lambdaSequence|$ of predicted responses corresponding to the $|lambdaSequence|$ fitted models.

There must be something wrong in the implementation (I have passed many hours on that function trying to understand where a problem might come from) because the resulting plot of MSE Lasso models vs. lambda the regularization parameter doesn't show any significant cross-validated MSE reduction by penalizing the Loss function (in annex).

Best lambda found with 1-leave-out process is either 0 or 0.01 depending on the lambda sequence given in input.

Do you think that the problem comes from the implementation or is due to the fact that 1-leave-out MSE estimation is here not so good ?

Exercise 6 5-Fold Cross validation, 10-Fold Cross validation

Solution 6 Here I used a common function for these 2 cross validation processes, taking as input 2 dataframes (1 for the predictors, 1 for the response) and an integer for the number of folds in the cross validation process.

```
cv <- function(X,Y, lambdaSeq, k) {
    errVector <- rep(0, length(lambdaSeq))

    kFolds <- cvFolds(nrow(X), k)

    X["subset"] = sapply(kFolds$subsets, function(x) {x %% k + 1})

    for(i in 1:k) {

        Xtr <- X[X$subset != i,-c(9)]

        Xte <- X[X$subset == i,-c(9)]

        Ytr <- Y[X$subset != i,]

        Yte <- Y[X$subset == i,]

        lassoFit <- glmnet(as.matrix(Xtr), Ytr, alpha=1, lambda=lambdaSeq)

        Xpred <- predict(lassoFit, as.matrix(Xte))

        Xerr = sweep(Xpred, 1, as.matrix(Yte), '-')

        errVector <- errVector +  colSums(Xerr * Xerr)

    }

    return(errVector/nrow(X))

}
```

The function is quite close to the previous one (but seems to work normally). Folds are generated by cvFolds() function from cvTools package. predict() function used with our Lasso fit with multiple lambdas outputs a $|fold| \times |lambdaSequence|$ matrix containing predicted responses for all observations out of the training set (1 row by observation) for all fitted models (1 by column). We subtract to all columns of this matrix the real response Y, make a matrix with these squared coefficients (by the element-wise matrix multiplication operator *) and add it to the previous errors vector.

I have also implemented a function that averages K-fold cross validated MSE (over N cross validations) for all Lasso models and returns the best chosen lambda :

```
crossValidation <- function(X, Y, lambdaSeq, k, N) {
    errors = rep(0, length(lambdaSeq))
    for(i in 1:N){
        errors <- errors + cv(X,Y, lambdaSeq, k)
    }
    errors <- errors/N
    return(lambdaSeq[which.min(errors)])
}
```

For both K in $\{5, 10\}$, best regularization parameter is 0.03 which is in average the best λ value returned by the function cv.glmnet() (function that surprinsigly doesn't average cross-validated MSE over a large number of K-fold cross validations).

I also realized that this function returns (by default) optimized β^{lasso} corresponding to the biggest lambda such that error is within 1 standard deviation from lowest error. This explains why I didn't get the same parameters nor the same number of parameters than in the ESL book. I get also different result because I have used the whole set for the cross-validation.

For $\lambda = 0.03$, we get a model with 7 predictors (instead of 8 for a full model and 5 in the book because they don't take he best model but a "good" model with error 1 std unit close to the minimal error).

Exercise 7 - Generalized cross-validation

Solution 7 I was not able to produce this function. ESL book gives the following formula for Generalized cross-validation :

$$GCV(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N}\frac{y_i - \hat{f}(x_i)}{1 - \frac{trace(\boldsymbol{S})}{N}}$$

Where $\boldsymbol{S}$ is the Hat matrix obtained by fitting Lasso regression. But is seems much more complicated to get an analytical expression of $\boldsymbol{S}$ than for Ridge Regression. I have found that, under the assumption about the data that $\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{I}$ (which means that the predictors are uncorrelated ??) we can find an exact analytical expression of β^{lasso} in function of $\hat{\beta}^{OLS}$.

Must we take this analytical expression even if data is not orthonormal at all (like in this case) ? What about the data standardization of the algorithm ? How does it affect the analytical result for $\hat{\beta}$ and $\boldsymbol{S}$? Is there simple way to compute/estimate $\boldsymbol{S}$?

Exercise 8 Grow and prune a classification tree on the spam data of the ESL.

Solution 8 For this exercise, I have used the R package tree, that seems to be the more used and documented R package for manipulating trees.

I used the following piece of code for growing and pruning a tree (with the same data) with the ESL spam data :

```
treeFit <- tree(spam ~., data=spam, split="deviance")

prunedTreeFit <- prune.tree(treeFit, k = seq(0.0, 100.0, 0.1))

prunedTreeFit <- prune.tree(treeFit, k = 120)
```

Used with a ”k-sequence”, prunedTreeFit is a sequence of pruned trees. I also tried the cv.tree function that estimates the true deviance of the tree by K-folds cross-validation :

```
cvPrunedTree <- cv.tree(treeFit, FUN=prune.tree, K=5)
```