HDS Exercise set 4.

YOUR NAME (STUDENT NUMBER)

Return by **10.15** o'clock on **26.11.2019** to the Moodle area of the course. You can use 'HDS_ex4.Rmd' file as a template for your answers. Return the final file in pdf format with name "HDS4_yourname.pdf". Some exercises take some time to compile (Problem 4) and you can include `eval = F` to the initialization of a code block (like {r, eval = F}) to avoid the code being executed while you work on the other parts. But remember to turn it to `eval = T` before making the final version!

Problem 1 (a-d) & Problem 2 (e-g)

Let's use `Boston` data set from `MASS` package. Outcome is once again `medv` and all other variables are used as predictors.

Split the data into training and testing part by having 350 indexes from file "HDS_ex3.3_tr.txt" as training samples and the rest as test samples.

(a) Use `glmnet` package to fit ridge regression model of `medv` on all other variables in the training set. (Help on using glmnet are at https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html and in lecture notes.) Start by calling `cv.glmnet()` and plot its output to get an idea of effect of λ values on prediction accuracy. What does the plot tell about the need to penalize ordinary linear model in this task? Print out `lambda.min` and `lambda.1se`. Why would you sometimes might want to use `lambda.1se` for predictions even though the minimum of MSE is seen at `lambda.min`?

(b) Plot the ridge regression paths of the coefficients as function of `log lambda` by calling plot on the `glmnet` object within the `cv.glmnet` object that you got in part (a). Label the variables. Which variables have the largest effects here? Check their names and check what ordinary `lm()` says about them when you regress `medv` on everything else jointly. Do they have smallest P-values?

(c) Repeat the analysis of part (b) but now standardize all predictors (not outcome) before giving them to `glmnet`. Plot again the paths of coefficients. Which ones now have the largest effects? How do these compare to smallest P-values in the standard linear model analysis done above?

(d) Use the ridge regression model with `lambda.min` value to predict the `medv` in test data and compute MSE in test data. Remember to give the test data variables in the same scale that was used in training the model (either original scale in both or standardized scale in both but not mixed up). How large a proportion is this MSE out of the total variance in `medv` in test data?

(e) Fit a LASSO model by first calling `cv.glmnet` and then using the default parameter `lambda.1se` in prediction. How does LASSO's MSE in test data compare to that of ridge regression computed in (d)?

(f) Fit the standard linear model in training data and test its predictive ability in test data using MSE. How does it compare to ridge regression (d) and LASSO (e)? Which of the three methods should we use for this data set?

(g) Print out a table with 14 rows and four columns that show the estimated coefficients (including intercept) for linear model (col=2), ridge regression at `lambda.min` (col=3) and LASSO at `lambda.1se` (col=4). The 1st column gives the name of the variable. How many variables are set to 0 by LASSO? Are the coefficients from ridge regression and LASSO always shrunk towards 0 compared to the unpenalized linear model estimates? If not, why not?

Problem 3.

Read in data "HDS_ex4_n300p240.txt". It has 243 columns: Two outcome variables `y1` and `y2`, training data set indicator `train` (1=training data sample, 0=test data sample) and 240 predictors `X.i` where i=1,...,240.

Use `glmnet` package to fit ridge regression and LASSO in training data for both outcome variables separately. Use cross-validation (from `glmnet` package) to choose tuning parameter λ in each analysis. (Make sure that the grid of λ values cover the whole region of interest and if not, adjust the `lambda` parameter manually in the function call.) Show cross-validation plots for each analysis and show coefficient plots for each analysis. Once you have chosen the λs, do prediction in test data set and compute test data MSE. Which method is preferred for which outcome variable?

Problem 4.

Continue with "HDS_ex4_n300p240.txt" from problem 3.

(a) Compute univariate P-values using `lm()` for each outcome-predictor pair (2 x 240 = 480 P-values altogether) in the training data. (You can use `lm()` within `apply()` as in HDS1.) Make a QQ-plot of -log10(P-values) of these univariate associations comparing the distributions of P-values between the two outcome variables. How could you had guessed from this plot whether RIDGE or LASSO was preferred for each outcome variable?

(b) Use stepwise forward regression with BIC as the criterion to choose separate models for `y1` and `y2` in training data. Compute training and test MSEs of those models. How do the test MSEs from BIC compare to the smaller of MSEs of RIDGE and LASSO for each outcome variable that you computed in problem 3?

Problem 5.

Let's implement K-fold cross-validation to choose λ parameter in ridge regression.

We work with `Boston` data set from `MASS` package and with `medv` as outcome and all other variables are used as potential predictors.

(a) Split the data into training and testing part by having indexes HDS_ex3.3_tr.txt as training samples and the rest as test samples. Write a function `CV(K, lambda)` that takes in parameters K and λ and returns an estimate of MSE using K-fold cross-validation in training data. `CV()` should use `glmnet(x, y, alpha=0, lambda = lambda)` to fit the ridge regression model and use `predict.glmnet()` to get the predictions to compare with the validation set for each fold in the cross-validation process. CV should also divide the training data into K-folds randomly so that sequential calls to `CV()` will return slightly different MSE estimates. (You can assume that `K` and `lambda` are scalars (length=1), that the data are automatically available to your function in the current environment, and you can hard code the outcome variable name within function. So no need to make `CV()` any more general than that it does the job for this particular problem.)

(b) Run your cross-validation function for $K = 10$ and sequentially for 5 values $\lambda = 0.01, 1, 10, 50, 100$. For each value of λ call the CV function $R = 10$ times and collect the MSE results of all R replicates. Show 5 boxplots of MSE values in one figure having λ on x-axis and print out the mean MSEs for each λ. What do the result tell about the need for penalized regression in this data set? (We already saw the same thing in Problem 1&2.)

(c) Run `cv.glmnet(...,alpha=0, lambda=c(0.01, 1, 10, 50, 100))` on the same training data and extract only the fields `lambda` and `cvm` from the output. Compare to your values in (b).