ABE6933 SML HW4

Directions

Please submit **one PDF** file including all your reports (answer + code + figures + comments; must be easily readable and have file size under a few megabytes) and **one R code script**. The R script is supplementary material to ensure that your code runs correctly. If you are using RMarkdown, please also include your `.Rmd` file.

Place these two (or three) files in a folder, make a zip or rar archive, and submit the archive electronically via Dropbox file request at `tinyurl.com/nbliznyuk-submit-files` (on the landing page, enter your name so that we know it is you and email so that you get a confirmation).

For the full list of rules/policies/expectations, please visit "hw.rules.pdf" document.

Deadline: 15-Oct-2020, 11:59 PM EST.

Practice/Optional Problems (do not submit)

1. Complete tutorial the R tutorial in ISLR section 4.6. You may find the Youtube videos by Trevor Hastie helpful; for links, see file `!_youtube_lab_links.txt` in the subfolder `"[2].code/islr_labs/"`
2. ISLR ch.4: problem 1

Required Problems (for submission)

ISLR ch.4: problems 5, 9, 11 (except g - KNN)

Problem 11 clarification: use the following code to define the training and test sets

```
# n = ?; # define n as number of observations/rows in the data frame
propTrain = 0.7;                # proportion of obs for training
nt = floor(propTrain * n);      # number of obs for training
set.seed(0); permID = sample(n); # see documentation and default args to the sample() fun
train = permID[1:nt];           # ids/row numbers of the training observations
test  = permID[(nt+1):n];       # ids/row numbers of the test observations
```

Typed Problem 1. maximum likelihood estimation in logistic regression.

Consider the Challenger O-rings dataset (see the "hw3/challenger" folder); case study details are in the Powerpoint file.

(a) Implement "by hand" the negative log-likelihood (as a function of parameter vector $\beta = [\beta_0, \beta_1]'$) for this dataset. You are allowed to use the R function `dbinom` (set log=TRUE), but you not required to use it. Important: if you implement it directly without `dbinom`, make sure that you simplify as much as possible, i.e., implementing the likelihood and then applying the logarithm without simplifications (i.e., converting products into sums) is particularly dangerous. Test your implementation to ensure that it does not return NA's, infinities and negative infinities.

(b) Optimize your objective function in 1(a) using a gradient-based algorithm using R function `optim` function (e.g., with BFGS option) and compare your estimates with the ones produced by R function `glm` fit. Important: if you start very far from the solution, you may experience convergence issues even if your objecive function has been implemented correctly. Run numerical optimization for multiple starting points, and use `b=c(15,0)` as one of your starting points.

(c) (optional) estimate the approximate variance-covariance matrix of the MLE betahat as the inverse of the "observed Fisher information" (which is the curvature/Hessian with respect to β - estimated by finite differences - of the negative log-likelihood evaluated at the MLE $\widehat{\beta}$.) Compare with the standard errors reported by the summary of the model fitted by the R function `glm`.

Specifically, to do (the optional) 1(c):

(i) write a function that uses finite differences to approximately compute the matrix of second derivatives (known as the Hessian) at a given point. (Alternatively, use function `fdHess` from the R library `nlme`.)

(ii) obtain the approximate Hessian of the negative log-likelihood at the MLE solution (your $\widehat{\beta}$ here); call it H;

(iii) compute the inverse of H - call it S; S is our estimate of the covariance matrix of $\widehat{\beta}$

(iv) examine the square root of the diagonal of S (estimated standard errors of $\widehat{\beta}$) and compare it with the standard errors reported by (summary of) the GLM fit.

Typed Problem 2. classifiers with linear and nonlinear decision boundaries.

Consider the simulated dataset used in hw1, in "SML.NN.data.csv." Here, combine "train" and "valid" subsets to train the models below, then report misclassification rate on the "test" subset and discuss your results. Models for consideration:

L1: logistic regression with an intercept and additive (main) effects of X1 and X2

L2: logistic regression with an intercept, additive (main) effects of X1 and X2, as well as squares of X1 and X2 and the X1*X2 interaction

D1: linear discriminant analysis

D2: quadratic discriminant analysis

(Optional): try to visualize the decision boundaries, particularly for L2 and D2. To do so, you can setup a grid of equally spaced points (e.g., use $m \approx 50$ points per dimension and $m^2 \approx 50^2$ points total; if necessary, increase m to about 100; use small markers/dots as in the lectures) and color those with respect to the corresponding predicted classes.

Note: in this problem, we pool "train" and "valid" sets and use those for training because the classifiers do not have extra tuning parameters that require additional calibration (and all model parameters are estimated statistically). In later chapters, we'll see that this is generally not the case (e.g., we have already seen an example of a NN method where tuning parameter (neighborhood size/radius) needs to be calibrated).