Economics 4803/8803: Machine Learning for Economics

Problem Set 4

Due by: April 20, 5 PM ET

You are allowed to work in groups of up to four students, but you must disclose the members of your groups. Individual submissions are required. The code you submit may be identical to the one of the other group members, but I expect comments and answers to the questions to be your own.

Your submission should consist of:

1. a pdf file with responses to the questions in *(do not attach any code snippets. You will receive a penalty of 0.5 points if you do.)
2. a `R` document with code
3. you should upload these materials separately via the course's Canvas website (please do not email me with your home-work submission).

1 Empirical Problems

1. Analysis I: Run simulations with series, neural nets, and random forests (2 points).
   (a) Install `neuralnet` and `randomForest` packages in `R`.
   (b) Draw five separate x variables with random correlation using the code provided in Section 2.1
   (c) You will estimate two sets of three models. For each, set the seed to 0, sample size to 1,000, and allocate 50% of the data to a test sample.

      Specification 1: $y = x_1 + \frac{x_3 x_2^2}{10} + \frac{x_4 x_1 x_5}{10}$

      Specification 2: $y = \log(|x_1^4/10| + |x_2| + x_3^2) + x_4 x_2 sin(x_5) + u$, where $u \sim N(0,1)$

      For each specification:

      i. Estimate a neural net with 3 hidden layers, each with 64, 32, and 16 neurons respectively.

ii. Estimate a series using the `poly` function. Set the degree to 3.

iii. Estimate a random forest. Use 1000 trees with 4 covariates sampled each time.

(d) Repeat i. - iii., about 50 times and average the test MSE to get a single set of MSE's for each specification. Report these average MSE's in a table (three models across the two specifications). * [1] (keep your answers for the prompts below not more than **4 lines each**)

i. For specification 1, which performs best? Why do you think that is?*

ii. For specification 2, which performs best? Why do you think that is?*

(e) **Challenge problem (optional)**: Deep neural networks often require regularization in order to avoid overfitting. Another way of implementing DNNs in `R` is using the `keras` package.[2] It is faster and allows more control over the neural net architecture, including regularization. Use `keras` to implement the same neural net as before, but with drop-out regularization. Report the test MSE for both specifications. (bonus 1 point)

2. Consider the regression problem of predicting the price of listing from Problem Set 2. Start from the data cleaned for the analysis part and the 22 variables you generated for step 2.(d). Allocate 90% obs for training and 10% for testing to be used only for part (b). (2 points)

(a) Use PCA to compute the Principal Components (PC) of the 22 covariates. Identify the top 4 PCs by the proportion of variance explained (PVE). What is the individual and cumulative PVE of these 4 PCs? *

(b) Now store the top 4 PCs separately, and use these as covariates in a linear regression to predict price of a listing. Use this regression model to predict prices for the test data and compute the test MSE. Show the test MSE in a table along with the test MSE for second order polynomial model and regularized Lasso and Ridge models (without noise) from parts 2(d) and 2(g) respectively.[3]

[1]You can implement parallel computing in `R` using `doParallel` in order to make your code faster. Check out the sample code in Section 2.2.

[2]Refer to Lab 10.9 in ISL for details on how to install and use `keras`.

[3]For PCR, do **not** use the canned function `pcr`, proceed with `prcomp`. You might have to re-estimate the polynomial, ridge, and lasso from PS 2 for 90% training set size. You may use the code from the solution for this. Remember to use the same test data to compute test MSE for all these 4 models.

How does the test MSE from PCR compare with the other models? Why is that? *

3. Now return back to the problem of prediction of whether a host is a superhost from Problem Set 3. Start from the data cleaned for the analysis and allocate 90% obs for training and 10% for testing. (2 points)

   (a) Use the set of 20 predictors you created in part (g) for this problem. Estimate a random forest (use 1000 trees) with 4 predictors chosen at random. How does it perform, in terms of mean classification error, relative to the cross-validated flexible logit model in part (g)? Why do you think this is?*

   (b) Use K-means clustering on the training data to cluster observations into 1000 clusters.[4] Next, summarize the training data by the cluster assignment such that each new observation in the summary data is an average of all the observations within the cluster assignment.[5]

      i. Compare the pairwise correlations amongst `host_is_superhost`, `review_scores_rating`, and `host_experience` in the training and the summary datatset. What do you observe and why? *

      ii. Perform SVM on this new dataset (1000 obs.) with the same tuning parameters as in PS 3, part 2 (f). Report the mean classification error on the test set and compare the performance with your results from SVM in PS 3. Comment on aspects like speed of computation and prediction performance. *

[4]Use `kmeans` command and `host_is_superhost`, `review_scores_rating`, `host_experience` for clustering.

[5]Take mode for the variable `host_is_superhost`

2 Sample code

2.1 Drawing variables for Analysis I

- Use the following code snippet to draw the five x variables to be used in the simulation exercise in analysis I

```
#Draw observable explanatory variables
n <- 1000
x1 = rgamma(n,2,1); x2 = rnorm(n,0,2);
x3 = rweibull(n,2,2); x4 = rlogis(n,2,1);
x5 = rbeta(n,2,1);
x = cbind(x1,x2,x3,x4,x5)

###############################################
#transform into independent random variables
# find the current correlation matrix
c1 <- var(x)
# cholesky decomposition to get independence
chol1 <- solve(chol(c1))
x <- x %*% chol1
###############################################
#generate random correlation matrix
R <- matrix(runif(ncol(x)^2,-1,1), ncol=ncol(x))
RtR <- R %*% t(R)
corr <- cov2cor(RtR)
# check that it is positive definite
sum((eigen(corr)$values>0))==ncol(x)
################################################
#transform according to this correlation matrix
x <- x %*% chol(corr)

datam <- as.data.frame(x)
datam <- datam %>% dplyr::rename(x1 = V1, x2 = V2, x3 = V3, x4 = V4, x5 = V5)
```

2.2 Implementing parallel computing in `R`

- While there are several ways of implementing parallel computing in `R`, we will focus on one such technique using `doParallel` and `foreach`.
- Install the package `doParallel`

```
# number of simulations to run (100 is just an example!)
nsim <- 100
# set parallelization
# detect the number of Cores available in the system
nCores <- parallel::detectCores()

cl <- parallel::makeCluster(nCores);
doParallel::registerDoParallel(cl)

# Note: foreach is different than the traditional for loop
# You need to include the packages in the foreach loop!

results <- foreach(i=1:nsim, .combine=rbind, .packages = c('neuralnet',
'randomForest')) %dopar% {

    # <insert code for simulation exercise here>
    # Note: foreach returns the results as a matrix or list
    # for e.g. the output below will be a list of numbers {(1+5), (2+5),...,(sim+5)}
    example <- i + 5
    c(example)
}

#cleanUp
parallel::stopCluster(cl)
rm(cl)
# report results
print(c(mean(results)))
```