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)))
```

1.1

d)
i. The neural network model performs the best for Specification 1, achieving the lowest MSE of 0.0816894. This suggests that the neural network's capacity for capturing nonlinear relationships and complex interactions between predictors is particularly effective for the mathematical structure defined in Specification 1, where interactions and non-linear terms are explicitly modeled.

ii. For Specification 2, the random forest model yields the best performance, with the lowest MSE of 1.455309. This is likely due to the random forest's ability to handle high-dimensional data and complex relationships without overfitting, especially beneficial in Specification 2, where the response variable is influenced by complex transformations and interactions among predictors.

e)
Specification 1 had a test MSE of 0.327792435884476 and Specification 2 had a test MSE of 2.14555215835571

1.2

a)

Individual PVE of Top 4 PCs:

- PC1: 34.239% of the variance is explained by the first principal component
- PC2: 20.68% of the variance is explained by the second principal component
- PC3: 15.054% of the variance is explained by the third principal component
- PC4: 9.144% of the variance is explained by the fourth principal component

Cumulative PVE of Top 4 PCs:

- After the first PC, 34.239% of the total variance in the data is captured
- After the second PC, the cumulative variance explained rises to 54.919%
- After the third PC, it increases to 69.973%
- After the fourth PC, it totals 79.117%

- The top four PCs together explain approximately 79.117% of the variance in the 22 covariates. This indicates a strong representation of the original data's variability, suggesting that these four components capture most of the essential information contained in the full set of covariates.
- Using these four PCs for further analysis or modeling could significantly reduce the dimensionality of the data while still retaining a majority of the information, which is beneficial in avoiding overfitting and improving model interpretability and efficiency.
- The steep drop in the percentage of variance explained from PC1 to PC4 suggests that the most significant patterns in the data are highly concentrated in the first few components. This could indicate that a few underlying factors (possibly related to stronger signals in the data) are influencing the variation in the dataset significantly more than others.

b)

Analysis of MSE Results:

- PCR MSE: 24192.75
- Polynomial MSE: 24597.43
- Ridge MSE: 13252.53
- Lasso MSE: 13213.06

Dimensionality Reduction in PCR:

- PCR involves reducing the predictor variables to a few principal components that explain most of the variance. This process can lead to loss of information if relevant predictive information is contained in components that are discarded. If the top principal components do not capture key predictors of the target variable, the MSE can be higher as compared to more comprehensive models.

Model Complexity and Regularization:

- Polynomial Regression may suffer from overfitting especially with higher-degree terms and multiple predictor interactions, hence the high MSE comparable to PCR.

- Ridge and Lasso Regressions introduce regularization that penalizes the magnitude of coefficients which can effectively reduce overfitting and improve model performance on test data. This is why both have significantly lower MSEs compared to PCR and Polynomial models. Ridge handles collinearity better by shrinking coefficients, while Lasso does feature selection by setting coefficients of less important features to zero.

Suitability of Linear Models:

- PCR and Polynomial models both rely on linear combinations of features or their transformations. If the true relationship between the features and the target variable is highly non-linear or involves complex interactions not captured by the top principal components or specific polynomial terms, these models may not perform as well.

Optimization of Regularization in Ridge and Lasso:

- The better performance of Ridge and Lasso may also be due to the optimization of their hyperparameters (like lambda) through cross-validation, which is evident from the usage of cv.glmnet. This process helps in finding a sweet spot for regularization strength, balancing bias and variance effectively.

1.3

a)
The Random Forest model yielded an MCE of 0.03556 compared to 0.1804586 from the lasso logit model.

Random Forest Performance

- Random Forest is generally more robust against noise and outliers than logistic regression. It does not assume a linear relationship or distribution of the data, which can be beneficial if the relationship between features and the target is complex or nonlinear.
- Random Forest can capture non-linear interactions between variables more effectively without the need for explicit feature engineering (like interaction terms or polynomial features), which is crucial in this scenario as the predictors include squared terms and interactions.
- Random Forest inherently performs feature selection by selecting the most informative features through the splitting criteria in the tree construction process. This might have helped in better capturing the predictive signals in the data with the random subset of features (mtry = 4) used at each split.

Lasso Logit Model Limitations

- The lasso logit model assumes a linear relationship between the log-odds of the outcome and each predictor, which might not adequately capture the complexities and non-linear relationships in the data.
- While lasso helps in feature selection by shrinking coefficients of less important predictors to zero, it may also oversimplify the model when important interactions or non-linear relationships exist.
- Logistic regression can be sensitive to the scale of the features and the underlying distribution of the data. The performance can degrade if the feature scaling and normalization are not appropriately handled.

b)
i.

- The correlation between host_is_superhost and review_scores_rating increased from 0.236 in the training data to 0.307 in the summary data. Similarly, the correlations involving host_experience with other variables also show slight increases.
- Generally, the summary data shows higher correlation coefficients than the raw training data. This can be indicative of the averaging process highlighting broader trends that are less apparent in the more granular training data.

Reasoning:

- When data is summarized by cluster, each cluster's mean value tends to represent a central tendency that reduces individual variability. This averaging can enhance the underlying patterns between variables, making correlations clearer and potentially stronger.

- The process of clustering and summarizing reduces noise in the data. Individual outliers or variations that might obscure relationships in the full data set are less influential when data is aggregated into cluster means. This typically leads to a more pronounced expression of the main trends, reflected in higher correlation coefficients.
- The clusters created by K-means are likely grouping together observations that are similar not just in terms of one variable, but across multiple variables. This co-grouping can sharpen the relationships between variables, as each cluster mean will reflect the average characteristics of similarly behaving groups.

ii.

MCE Values:

- Unclustered Data (SVM): MCE = 0.0867
- Clustered Data (SVM_clustered): MCE = 0.16

Prediction Performance:

- The SVM on the unclustered data has a significantly lower MCE compared to the SVM on the clustered data. This suggests that the SVM model trained on the original, more detailed dataset was able to capture the nuances and variability in the data better than the model trained on the clustered summary data.
- The reduction in performance on the clustered data could be due to the loss of information as a result of averaging the variables within clusters. Such data aggregation may obscure individual variations that are critical for accurately predicting whether a host is a superhost.

Speed of Computation:

- Using K-means to cluster the data into 1000 groups and then summarizing it likely reduced the computational load for the SVM model due to the reduced number of observations (from potentially thousands to just 1000 clusters).
- SVM models, particularly those with radial kernels, are computationally intensive, especially with large datasets. Reducing the dataset size through clustering can significantly decrease the model training time.
- However, the trade-off for faster computation here is the reduced prediction accuracy, as evidenced by the higher MCE.