Stat 432 Homework 4

Assigned: Sep 16, 2024; Due: 11:59 PM CT, Sep 26, 2024

Instruction

Please remove this section when submitting your homework.

Students are encouraged to work together on homework and/or utilize advanced AI tools. However, **sharing, copying, or providing any part of a homework solution or code to others** is an infraction of the University's rules on Academic Integrity (https://studentcode.illinois.edu/article1/part4/1-401/). Any violation will be punished as severely as possible. Final submissions must be uploaded to Gradescope (https://www.gradescope.com/courses/570816). No email or hard copy will be accepted. For **late submission policy and grading rubrics** (https://teazrq.github.io/stat432/syllabus.html), please refer to the course website.

- You are required to submit the rendered file `HWx_yourNetID.pdf`. For example, `HW01_rqzhu.pdf`. Please note that this must be a `.pdf` file. `.html` format **cannot** be accepted. Make all of your `R` code chunks visible for grading.
- Include your Name and NetID in the report.
- If you use this file or the example homework `.Rmd` file as a template, be sure to **remove this instruction** section.
- Make sure that you **set seed** properly so that the results can be replicated if needed.
- For some questions, there will be restrictions on what packages/functions you can use. Please read the requirements carefully. As long as the question does not specify such restrictions, you can use anything.
- **When using AI tools**, you are encouraged to document your comment on your experience with AI tools especially when it's difficult for them to grasp the idea of the question.
- **On random seed and reproducibility**: Make sure the version of your `R` is $\geq 4.0.0$. This will ensure your random seed generation is the same as everyone else. Please note that updating the `R` version may require you to reinstall all of your packages.

Question 1: Sparsity and Correlation [50 pts]

During our lecture, we considered a simulation model to analyze the variable selection property of Lasso. Now let's further investigate the prediction error of both Lasso and Ridge, and understand the bias-variance trade-off. Consider the linear model defined as:

$$Y = X^{\mathrm{T}}\boldsymbol{\beta} + \epsilon$$

Where $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{100})^T$ with $\beta_1 = \beta_{11} = \beta_{21} = \beta_{31} = 0.4$ and all other β parameters set to zero. The p-dimensional covariate X follows a multivariate Gaussian distribution:

$$\mathbf{X} \sim \square(\mathbf{0}, \Sigma_{p\times p}).$$

In Σ, all diagonal elements are 1, and all off-diagonal elements are ρ.

a. [15 points] A single Simulation Run

   - Generate 200 training and 500 testing samples independently based on the above model.
   - Use $\rho = 0.1$.
   - Fit Lasso using `cv.glmnet()` on the training data with 10-fold cross-validation. Use `lambda.1se` to select the optimal λ.

- Report:
  - Prediction error (MSE) on the test data.
  - Report whether the true model was selected (you may refer to HW3 for this property).

Solution:

```
# Load required packages
library(glmnet)
library(MASS)

# Set random seed for reproducibility
set.seed(432)

# Initialize parameters
p = 100
beta = rep(0, p)
beta[c(1, 11, 21, 31)] = 0.4
rho = 0.1
ntrain = 200
ntest = 500

# Generate covariance matrix Sigma
Sigma = matrix(rho, p, p)
diag(Sigma) = 1

# Generate training and testing data
xtrain = mvrnorm(ntrain, rep(0, p), Sigma)
xtest = mvrnorm(ntest, rep(0, p), Sigma)
ytrain = xtrain %*% beta + rnorm(ntrain)
ytest = xtest %*% beta + rnorm(ntest)

# Fit Lasso regression model using 10-fold CV
lasso.fit = cv.glmnet(xtrain, ytrain, nfolds = 10)

# Make predictions and calculate test error for Lasso
lasso.pred = predict(lasso.fit, xtest, s = "lambda.1se")
lasso.error = mean((ytest - lasso.pred)^2)

# Check if the correct model was selected by Lasso
lasso.beta = as.vector(coef(lasso.fit, s = "lambda.1se"))[-1]
lasso.correct = all((lasso.beta != 0) == (beta != 0))

lasso.error
```

```
## [1] 1.602201
```

```
lasso.correct
```

```
## [1] TRUE
```

Hence, the Lasso model gives us prediction error 1.6022015. The correct model was selected by Lasso: TRUE.

b. [15 points] Higher Correlation and Multiple Simulation Runs

- Write a code to compare the previous simulation with $\rho = 0.1, 0.3, 0.5, 0.7, 0.9$.

- Perform 100 simulation runs as in part a) and record the prediction error and the status of the variable selection for Lasso.
- Report the average prediction error and the proportion of runs where the correct model was selected for each value of ρ.
- Discuss the reasons behind any observed changes.

Solution:

```
nsim = 100
rho_all = c(0.1, 0.3, 0.5, 0.7, 0.9)
error = rep(0, 5)
correct = rep(FALSE, 5)

for (i in 1:length(rho_all)) {
  rho = rho_all[i]
  Sigma = matrix(rho, p, p)
  diag(Sigma) = 1

  error_all = rep(0, nsim)
  correct_all = rep(FALSE, nsim)

  for (j in 1:nsim) {
    xtrain = mvrnorm(ntrain, rep(0, p), Sigma)
    xtest = mvrnorm(ntest, rep(0, p), Sigma)
    ytrain = xtrain %*% beta + rnorm(ntrain)
    ytest = xtest %*% beta + rnorm(ntest)

    lasso.fit = cv.glmnet(xtrain, ytrain, nfolds = 10)
    lasso.pred = predict(lasso.fit, xtest, s = "lambda.1se")
    error_all[j] = mean((ytest - lasso.pred)^2)

    lasso.beta = as.vector(coef(lasso.fit, s = "lambda.1se"))[-1]
    correct_all[j] = all((lasso.beta != 0) == (beta != 0))
  }

  # record
  error[i] = mean(error_all)
  correct[i] = mean(correct_all)
}

error
```

```
## [1] 1.190825 1.196359 1.172139 1.170749 1.157582
```

```
correct
```

```
## [1] 0.25 0.06 0.05 0.00 0.00
```

As ρ increases, the proportion of correct estimations decreases. This is mainly because Lasso can be unstable when facing correlated variables, and essentially, using one or its correlated ones may lead to similar loss. Hence it may falsely select a variable instead of the original one. However, this does not affect the prediction error significantly, as the average prediction error remains relatively stable across different values of ρ. This is mainly because correlated variables still have similar predictive power.

c. [10 points] Ridge Regression

- Repeat task b) with the ridge regression. You do not need to record the variable selection status since ridge always select all variables.
- Report the average prediction error, do you see any difference between ridge and Lasso? Any performance differences within ridge regression as ρ changes?
- Discuss the reasons behind any observed changes.

Solution:

```
nsim = 100
rho_all = c(0.1, 0.3, 0.5, 0.7, 0.9)
error = rep(0, 5)

for (i in 1:length(rho_all)) {
  rho = rho_all[i]
  Sigma = matrix(rho, p, p)
  diag(Sigma) = 1

  error_all = rep(0, nsim)

  for (j in 1:nsim) {
    xtrain = mvrnorm(ntrain, rep(0, p), Sigma)
    xtest = mvrnorm(ntest, rep(0, p), Sigma)
    ytrain = xtrain %*% beta + rnorm(ntrain)
    ytest = xtest %*% beta + rnorm(ntest)

    ridge.fit = cv.glmnet(xtrain, ytrain, nfolds = 10, alpha = 0)
    ridge.pred = predict(ridge.fit, xtest, s = "lambda.1se")
    error_all[j] = mean((ytest - ridge.pred)^2)
  }

  # record
  error[i] = mean(error_all)
}

error
```

```
## [1] 1.448439 1.396814 1.341430 1.271292 1.168431
```

The average prediction error for the ridge regression is decreasing as the correlation increases. This is mainly because ridge regression utilize the correlation between variables to improve the prediction accuracy. The ridge regression can be interpreted as performing PCA on the design matrix and then project the response onto the principal components, and then shrink the coefficients of these projections. When the correlation increases, the principal components can capture more variance of the response. And the true model can be pretty much captured by the first principal direction as the correlation increases, hence the prediction accuracy can be improved.

Question 2: Shrinkage Methods and Testing Error

In this question, we will predict the number of applications received using the variables in the College dataset that can be found in `ISLR2` package. The output variable will be the number of applications (Apps) and the other variables are predictors. If you use Python, consider migrating the data to an excel file and read it in Python.

a. [5 pts] Use the code below to divide the data set into a training set (600 observations) and a test set (177 observations). Fit a linear model (with all the input variables) using least squares on the training set using `lm()`, and report the test

error (i.e., testing MSE).

```
library(ISLR2)
```

```
##
## Attaching package: 'ISLR2'
```

```
## The following object is masked from 'package:MASS':
##
##     Boston
```

```
data(College)

# generate the indices for the testing data
set.seed(7)
test_idx = sample(nrow(College), 177)
train = College[-test_idx,]
test = College[test_idx,]
```

Solution:

```
model_linear <- lm(Apps ~ ., data = train)
summary(model_linear)
```

```
##
## Call:
## lm(formula = Apps ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3644.2  -439.4   -23.2   331.8  6997.3
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -421.53187  479.06765  -0.880 0.379276
## PrivateYes  -481.17081  165.98099  -2.899 0.003885 **
## Accept         1.63866    0.04557  35.959  < 2e-16 ***
## Enroll        -1.01396    0.23640  -4.289 2.10e-05 ***
## Top10perc     58.07527    6.79021   8.553  < 2e-16 ***
## Top25perc    -19.63197    5.35400  -3.667 0.000268 ***
## F.Undergrad    0.06800    0.04301   1.581 0.114440
## P.Undergrad    0.01053    0.04782   0.220 0.825724
## Outstate      -0.08698    0.02271  -3.830 0.000142 ***
## Room.Board     0.14536    0.05889   2.468 0.013859 *
## Books          0.11725    0.26326   0.445 0.656209
## Personal       0.01811    0.07241   0.250 0.802545
## PhD           -7.90837    5.21170  -1.517 0.129702
## Terminal      -5.47538    5.85494  -0.935 0.350087
## S.F.Ratio     18.01171   15.42608   1.168 0.243441
## perc.alumni    0.31887    4.77088   0.067 0.946734
## Expend         0.07428    0.01409   5.271 1.92e-07 ***
## Grad.Rate     11.10438    3.45399   3.215 0.001377 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1067 on 582 degrees of freedom
## Multiple R-squared:  0.9264, Adjusted R-squared:  0.9242
## F-statistic: 430.8 on 17 and 582 DF,  p-value: < 2.2e-16
```

```
ols_pred <- predict(model_linear, test)
ols_mse <- mean((ols_pred - test$Apps)^2)
ols_mse
```

```
## [1] 961142.3
```

The testing MSE for the linear model is 9.6114227^{5}.

b. [10 pts] Compare Lasso and Ridge regression on this problem. Train the model using cross-validation on the training set. Report the test error for both Lasso and Ridge regression. Use `lambda.min` and `lambda.1se` to select the optimal λ for both methods.

Solution:

```
train_data = data.matrix(train)
test_data = data.matrix(test)
```

Fitting the models

```
  ridge.fit <- cv.glmnet(y = train_data[, 2], x = train_data[, -2],
                         alpha = 0, standardize = TRUE)

  lasso.fit <- cv.glmnet(y = train_data[, 2], x = train_data[, -2],
                         alpha = 1, standardize = TRUE)
```

Summarize the models with prediction errors

```
  ridge_mse1 <- mean((predict(ridge.fit, s = "lambda.min", newx = test_data[, -2]) - test_data[,
2])^2)
  ridge_mse2 <- mean((predict(ridge.fit, s = "lambda.1se", newx = test_data[, -2]) - test_data[,
2])^2)

  lasso_mse1 <- mean((predict(lasso.fit, s = "lambda.min", newx = test_data[, -2]) - test_data[,
2])^2)
  lasso_mse2 <- mean((predict(lasso.fit, s = "lambda.1se", newx = test_data[, -2]) - test_data[,
2])^2)

   cbind(c("Ridge (lambda.min)", "Ridge (lambda.1se)", "Lasso (lambda.min)", "Lasso (lambda.1s
e)"),
         round(c(ridge_mse1, ridge_mse2, lasso_mse1, lasso_mse2), 2))
```

```
##      [,1]                 [,2]
## [1,] "Ridge (lambda.min)" "875548.88"
## [2,] "Ridge (lambda.1se)" "1399187.51"
## [3,] "Lasso (lambda.min)" "908340.33"
## [4,] "Lasso (lambda.1se)" "1076205.51"
```

For this particular run, Ridge regression with `lambda.min` gives the lowest prediction error.

c. [15 pts] The `glmnet` package implemented a new feature called `relaxed` fits and the associated tuning parameter `gamma` . You can find some brief explaination of this feature at the documentation of this package. See

```
* [CRAN Documentation](https://cran.r-project.org/web/packages/glmnet/glmnet.pdf)
* [glmnet Vignette](https://glmnet.stanford.edu/articles/glmnet.html)

Read these documentations regarding the `gamma` parameter, and summarize the idea of this feature
in terms of the loss function being used. You need to write it specifically in terms of the data
vectors $\mathbf y$ and matrix $\mathbf X$ and define any notations you need. Only consider the L
asso penalty for this question.

After this, implement this feature and utilize the cross-validation to find the optimal $\lambda$
and $\gamma$ for the College dataset. Report the test error for the optimal model.
```

Solution:

The new feature `relaxed` fits in the `glmnet` package considers a mixture of the model fit: one as the original Lasso fit and the other as the fit of the OLS estimator without penalty. This means that first each fixed λ and γ values, the model fit reported is

$$\gamma \frac{1}{2n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j| + (1-\gamma) \frac{1}{2n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^T \widehat{\boldsymbol{\beta}}_{\text{sub}})^2$$

where $\widehat{\beta}_{\text{sub}}$ is the OLS estimator fitted on the subset of variables selected (non-zero coefficients) by the Lasso. To implement this feature, we need to set `relax = TRUE` in the `cv.glmnet()` function.

```
  relaxed_fit <- cv.glmnet(y = train_data[, 2], x = train_data[, -2], relax = TRUE, gamma = seq
(0, 1, 0.2))
  relaxed_mse <- mean((predict(relaxed_fit, s = "lambda.min", newx = test_data[, -2]) - test_data
[, 2])^2)
  relaxed_mse
```

```
## [1] 948102.7
```

Question 3: Penalized Logistic Regression

In HW3, we used `golub` dataset from the `multtest` package. This dataset contains 3051 genes from 38 tumor mRNA samples from the leukemia microarray study Golub et al. (1999). The outcome `golub.cl` is an indicator for two leukemia types: Acute Lymphoblastic Leukemia (ALL) or Acute Myeloid Leukemia (AML). In genetic analysis, many gene expressions are highly correlated. Hence we could consider the Elastic net model for both sparsity and correlation.

```
if (!require("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install("multtest")
```

- [15 pts] Fit logistic regression to this dataset. Use a grid of α values in $[0, 1]$ and report the best α and λ values using cross-validation.

Solution:

```
# Load required packages
library(glmnet)
library(multtest)
```

```
## Loading required package: BiocGenerics
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, table,
##     tapply, union, unique, unsplit, which.max, which.min
```

```
## Loading required package: Biobase
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
    # Load Golub leukemia data
    data(golub)

    # Extract expression data (transposed for glmnet) and class labels
    X = t(as.matrix(golub))
    y = as.factor(golub.cl)

    # Initialize variables to keep track of results
    best_alpha = NULL
    best_lambda = NULL
    min_cv_error = Inf

    # List of alpha values to try
    alpha_values = seq(0, 1, by=0.1)

    # Loop over alpha values
    for (alpha in alpha_values) {

      # Fit elastic net model with current alpha value using cv.glmnet()
      cvfit = cv.glmnet(X, y, family="binomial", alpha=alpha, folds = 19)

      # Retrieve the mean cross-validated error for the optimal lambda
      cv_error = min(cvfit$cvm)

      # Update best_alpha and best_lambda if this model is better than previous best
      if (cv_error < min_cv_error) {
        best_alpha = alpha
        best_lambda = cvfit$lambda.min
        min_cv_error = cv_error
        best_model = cvfit
      }
    }
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
    # Output the best alpha and lambda
    cat("Best alpha:", best_alpha, "\n")
```

```
## Best alpha: 0.2
```

```
    cat("Best lambda:", best_lambda, "\n")
```

```
## Best lambda: 0.01957254
```