STATS601 Final Project

Chandler Nielsen

2025-03-30

In this document, I run some experiments to explore the ideas in the paper *On the cross-validation bias due to unsupervised preprocessing* by Amit Moscovich and Saharon Rosset. I focus on the paper's main example, wherein feature selection for high-dimensional linear regression is performed prior to the computation of cross-validated errors. This is performed using the simulated dataset described in the paper. I begin each section by attempting to mimic the results that were produced in the paper, followed by some of my own experiments to explore the ideas beyond the scope of the paper.

Loading Libraries

```
library(caret)
library(tidyverse)
```

Simulated Dataset Example

Reproducing Results - Functions

The following code generates data according the the *sampling distribution* part in Section 4 of the paper under review.

```
set.seed(82803)
generate_sample <- function(n = 200,
                            p = 100,
                            M = 5,
                            C = 5,
                            distribution = c("gaussian", "t"),
                            df = 4,
                            eta = 1) {

  distribution <- match.arg(distribution)

  # Generating design X from chosen distribution
  if (distribution == "gaussian") {

    X <- matrix(rnorm(n * p), nrow = n, ncol = p)

  } else {
```

```
    # t-distribution with 'df' degrees of freedom
    X <- matrix(rt(n * p, df = df), nrow = n, ncol = p)

  }

  # Scale the appropriate columns by C
  if (M > 0) {
    X[, 1:M] <- C * X[, 1:M]
  }

  # Generating betas from normal distribution
  beta_true <- rnorm(p, mean = 0, sd = 1)

  # Generating the error terms
  sigma2 <- eta * ((p - M) + (C^2) * M) # Noise variance
  eps <- rnorm(n, mean = 0, sd = sqrt(sigma2)) # The error

  # Generating the output y
  y <- as.numeric(X %*% beta_true + eps)

  # Compressing output
  return_list <- list(X = X, y = y,
                      betaTrue = beta_true,
                      sigma2 = sigma2)

  return(return_list)

}
```

Data Generation Now, we perform the *preprocessing* step described in the same section. Notice that this step is being performed **on the entire dataset**.

```
var_select <- function(X, K) {

  # Computing variances
  col_vars <- apply(X, 2, var)

  # Identify the top K columns
  highestVar <- order(col_vars,
                      decreasing = TRUE)[1:K]

  # Selecting these columns
  Xsel <- X[, highestVar, drop=FALSE]

  return_list <- list(XhighVar = Xsel,
                      bestIds = highestVar)

  return(return_list)
}
```

Preprocessing Step Finally, we create a function to perform the *predictor* step. This involves performing least squares regression without an intercept, both in the case where cross validation is performed on a dataset for which preprocessing was performed on the entire dataset and for which the preprocessing was only performed on the training dataset. We have the following:

```
improper_cv <- function(data_list, K = 10, numfolds = 5) {

  outputValues <- data_list$y
  covariates <- data_list$X

  ### Preprocessing on entire dataset
  selected_variables <- var_select(covariates, K)

  ### Creating Folds
  folds <- createFolds(outputValues,
                       k = numfolds,
                       list = TRUE,
                       returnTrain = TRUE)

  ### Performing Cross Validation
  mse_full <- numeric(length = numfolds)

  for (i in seq_along(folds)) {

    training_ID <- folds[[i]]
    trainingData <- as.data.frame(selected_variables$XhighVar[training_ID, ])
    testingData <- as.data.frame(selected_variables$XhighVar[-training_ID, ])

    outputValues_train <- outputValues[training_ID]
    outputValues_test <- outputValues[-training_ID]

    finalTraining <- cbind(trainingData, outputValues_train)
    finalTesting <- cbind(testingData, outputValues_test)

    ### Producing Model on training data
    myModel <- lm(outputValues_train ~ 0 + .,
                  data = finalTraining)

    ### Predicting on test subset
    my_predictions <- predict(myModel,
                              newData = finalTesting)

    ### Computing MSE for this Fold
    mse_full[i] <- mean((finalTesting$outputValues_test - my_predictions)^2)

  }

  ### Mean MSE Across Folds
  MSE_Mean_Full <- mean(mse_full)

  return(MSE_Mean_Full)
}
```

Improper CV Function Now we write the proper CV function. Because the validation is an unbiased estimator of the generalization error assuming that cross validation is performed appropriately, we will use this function to produce the plot for the generalization error.

```
proper_cv <- function(data_list, K = 10, numfolds = 5) {

  outputValues <- data_list$y
  covariates <- data_list$X

  ### Creating Folds
  folds <- createFolds(outputValues,
                       k = numfolds,
                       list = TRUE,
                       returnTrain = TRUE)

  ### Performing Cross Validation Correctly
  mse_correct <- numeric(length = numfolds)

  for (i in seq_along(folds)) {

    ### Splitting Data into Training/Testing Sets for this fold
    training_ID <- folds[[i]]
    train_covariates <- covariates[training_ID, ]
    train_output <- outputValues[training_ID]
    test_covariates <- covariates[-training_ID, ]
    test_output <- outputValues[-training_ID]

    ### Selecting high variance variables (only training data!)
    selected_variables <- var_select(train_covariates, K)

    ### Selecting variables only on the basis of training data
    train_covariates_final <- train_covariates[,
                                               selected_variables$bestIds]

    test_covariates_final <- test_covariates[,
                                             selected_variables$bestIds]

    train_covariates_final <- as.data.frame(train_covariates_final)
    test_covariates_final <- as.data.frame(test_covariates_final)

    ### Creating final data frames
    final_train <- cbind(train_covariates_final, train_output)
    final_test <- cbind(test_covariates_final, test_output)

    ### Creating model
    myModel <- lm(train_output ~ .,
                  data = final_train)

    ### Making predictions
    pred <- predict(myModel,
                    newdata = final_test)
```

```
    ### Computing MSE
    mse_correct[i] <- mean((final_test$test_output - pred)^2)

  }

  ### Mean MSE Across Folds
  MSE_Mean_Correct <- mean(mse_correct)

  return(MSE_Mean_Correct)

}
```

Proper CV Function

Reproducing Results - 1000 Runs

In the paper by Moscovich and Rosset, the above procedure is performed for over 100,000 runs. As a result, the authors omit error bars in their paper, since the uncertainty is negligible. Due to time and computational constraints, we perform the above procedure for 1000 runs instead. Like the authors, we omit the error bars, but note that for each n there is really a distribution of the MSE. Since the "proper cross validation" method takes into account preprocessing correctly, it is an unbiased estimator for the generalization error or risk.

```
num_runs <- 1000
sample_size <- seq(from = 200, to = 600, by = 50)

mse_incorrect <- rep(0, length(sample_size))
mse_correct <- rep(0, length(sample_size))

for (i in sample_size) {

  mse_vec_incorrect <- rep(0, num_runs)
  mse_vec_correct <- rep(0, num_runs)

  for (j in 1:num_runs) {

    ### Generate Dataset
    sample <- generate_sample(n = i,
                              p = 100,
                              M = 5,
                              C = 5,
                              distribution = "t",
                              df = 4,
                              eta = 1)


    ### Computing MSE for Incorrect Pipeline
    mse_vec_incorrect[j] <- improper_cv(sample,
                                        K = 10,
                                        numfolds = 5)

    ### Computing MSE for Correct Pipeline
    mse_vec_correct[j] <- proper_cv(sample,
```

```
                                    K = 10,
                                    numfolds = 5)


  }

  mse_incorrect[i] <- mean(mse_vec_incorrect)
  mse_correct[i] <- mean(mse_vec_correct)

}
```

Reproducing Results - Plots

$p = 100$, $M = C = 5$, $K = 10$, $x \sim t(4)$