Chapter 2

Coordinate Descent Algorithm for solving the LASSO Problem

2.1 Developed Coordinate Descent Algorithms

We investigate the Coordinate Descent Algorithm's practical application for addressing LASSO problems, based on Tseng's work from 1988 [8]. Incorporating an algorithm proposed by Friedman et al. (2007) [9], we tackle the LASSO problem's optimization with a coordinate descent technique. While the LASSO often performs well in regularisation tasks, its limitations prompt exploration into elastic net models, incorporating multiple penalties. After implementing coordinate descent for both the LASSO and elastic net, we conduct simulations, focus on parameter selection, and compare model performance.

2.1.1 LASSO

In our exploration of the model, we undertake the implementation of the algorithm outlined by Friedman et al. (2007) [9] using the programming language R. The process involves translating their proposed methodology into functional R code to execute the steps of the algorithm in a computationally effective manner. Below, we detail the formal steps of this algorithm.

1. **Initialization:**

   - Set all coefficients β_j to zero for each predictor: $\beta_j = 0$, where j ranges from 1 to p. This step initializes the coefficients before the iterative process begins.

2. **Iteration Process:**

   - **Step (a) - Compute Partial Residuals:** For each j from 1 to p, calculate the partial residuals (r_{ij}) for the i-th observation: The partial residuals are computed by subtracting the contribution of all predictors, except the j-th predictor, from the observed response variable y_i. Mathematically, it is represented as $r_{ij} = y_i - \sum_{k \neq j} x_{ik}\beta_k$, where y_i is the response, x_{ik} represents the i-th observation of predictor k, and β_k are the coefficients excluding the j-th predictor.
   - **Step (b) - Compute Least Squares Coefficient:** For each predictor j, compute the least squares coefficient (β_j^*) based on the partial residuals computed in the previous step: $\beta_j^* = \frac{1}{n}\sum_{i=1}^{n} x_{ij} r_{ij}$, where x_{ij} denotes the i-th observation of the j-th predictor, n is the number of observations, and r_{ij} are the partial residuals for predictor j.
   - **Step (c) - Update β_j by Soft Thresholding:** After obtaining β_j^*, update the coefficient β_j using soft thresholding: $\beta_j = \text{sign}(\beta_j^*)(|\beta_j^*| - \lambda)_+$, where λ is the penalty parameter. The soft thresholding technique involves shrinking coefficients towards zero. It takes the sign of β_j^* and multiplies it by the maximum between $|\beta_j^*| - \lambda$ and zero, resulting in the updated coefficient β_j.

During the process, the coefficients β_j are updated iteratively. This update relies on the calculated residuals and employs the soft thresholding technique, continuing until convergence is reached. This iterative approach gradually minimizes the LASSO objective function, aiming to find the optimal coefficients for the given data while enforcing sparsity in the solution. Throughout our analysis, we maintain default settings for tolerance and the maximum number of iterations in the models, grounded in our observation that adjusting these parameters has minimal impact on the model outcomes. The resulting R-code is outlined in listing 2.1.

```r
# Function to estimate betas using coordinate descent lasso
lasso_coord_desc <- function(X, y, lambda, tolerance = 1e-4,
                             max_iterations = 10000) {
  # Initialize coefficients based on the data structure
  n <- nrow(X)
  p <- ncol(X)
  beta <- rep(0, p)
  r <- y - X %*% beta # compute the initial residual with all betas=zero)

  # Begin iterative coordinate descent
  for(iter in 1:max_iterations) {
    beta_old <- beta # store beta from last iteration for convergence check
    # Loop over each of the p predictors
    for(j in 1:p) {
      # Calculate partial residual excluding current (j-th) predictor
      r_partial <- y - X[, -j] %*% beta[-j]
      # Compute rho (correlation)
      rho <- sum(X[,j] * r_partial)
      # Update beta[j] using soft thresholding as specified
      denominator <- sum(X[,j]^2)
      # implementing the thresholding function as if/else statement
      if(rho < -lambda) {
        beta[j] <- (rho + lambda) / denominator
      } else if(rho > lambda) {
        beta[j] <- (rho - lambda) / denominator
      } else {
        beta[j] <- 0
      }
      # Update residual (for the next iteration) with computed beta
      r <- y - X %*% beta
    }
    # Check for convergence using the specified tolerance
    if(max(abs(beta - beta_old)) < tolerance) {
      break
    }
  }# return the lasso estimated betas
  return(beta) }
```

Listing 2.1: R-code Implementation of the Coordinate Descent LASSO

2.1.2 Elastic Net

Since the LASSO may struggle with constraints like selecting a maximum of n variables when $p > n$, preferring single variables from highly correlated groups, and displaying subpar predictive performance compared to ridge regression in cases where observations outnumber predictors [10], the elastic net regularization method may serve as a solution.

This approach supplements the LASSO's penalty framework by integrating both L1 and L2 penalties, as defined in equation (2.1).

$$\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda_1 \sum_{j=1}^{p}|\beta_j| + \lambda_2 \sum_{j=1}^{p}\beta_j^2. \tag{2.1}$$

To extend our algorithm for the LASSO to encompass both L1 and L2 penalties, we introduce adjustments primarily within the soft thresholding process. Specifically, the integration of the L2 penalty involves modifying the denominator of the beta coefficient update within the thresholding function. This adaptation allows for a combined regularization approach, enhancing the penalty mechanisms by incorporating both L1 and L2 norms in the model's optimization process. The full and detailed R-code implementation is outlined in listing 2.2.

```r
# Function to estimate betas using coordinate descent elastic net
elastic_net <- function(X, y, lambda1, lambda2, tolerance = 1e-4,
                        max_iterations = 10000) {
  # Initialize coefficients based on the data structure
  n <- nrow(X)
  p <- ncol(X)
  beta <- rep(0, p)
  r <- y - X %*% beta # compute the initial residual with all betas=zero)

  # Begin iterative coordinate descent
  for(iter in 1:max_iterations) {
    beta_old <- beta # store beta from last iteration for convergence check
    # Loop over each of the p predictors
    for(j in 1:p) {
      # Calculate partial residual excluding current (j-th) predictor
      r_partial <- y - X[, -j] %*% beta[-j]
       # Compute rho (correlation)
      rho <- sum(X[,j] * r_partial)
      # Update beta[j] using soft thresholding as specified
      # Note the difference in the denominator compared to lasso
      denominator <- sum(X[,j]^2) + lambda2
      if(rho < -lambda1) {
        beta[j] <- (rho + lambda1) / denominator
      } else if(rho > lambda1) {
        beta[j] <- (rho - lambda1) / denominator
      } else {
        beta[j] <- 0
      }
      # Update residual (for the next iteration) with computed beta
      r <- y - X %*% beta
    }
    # Check for convergence using the specified tolerance
    if(max(abs(beta - beta_old)) < tolerance) {
      break
    }
  }# return the elastic net estimated betas
  return(beta)}
```

Listing 2.2: R-code Implementation of the Coordinate Descent Elastic Net

2.2 Regularization Parameter Selection

Selecting the optimal regularization parameters is crucial for the performance of both LASSO and Elastic Net models. This selection process is conducted through cross-validation, where we balance computational efficiency and the precision of the parameters.

For this process, we construct a list of potential values for the regularization parameter λ for LASSO and λ_1 and λ_2 for the elastic net. The initial value in each list is set to zero to include the scenario of no regularization. The subsequent values are determined based on an exponential function, allowing for a gradual increase in step size. This method ensures a more refined search in the lower range of values, where the model is more sensitive to changes in regularization strength, while also exploring sufficient breadth in the higher range to ensure comprehensive coverage of possible values.

In practice, we apply the Elastic Net algorithm across the grid of specified regularization parameters and observe the resulting model performance. The criterion for selection is the minimization of the average Mean Squared Error (MSE) across the validation sets. This ensures that the chosen parameters yield a model that generalizes well to unseen data, avoiding overfitting.

The cross-validation for LASSO is analogous but simpler due to its dependence on a single parameter, λ. This simplification results in a more expedited cross-validation process, as fewer iterations are required (see appendix 3.2.2).

```r
# CV function to find optimal elastic net lambda1/lambda2 value given data set
cross_validation_net <- function(x_train,y_train,x_val,y_val) {
  # The number of data sets can be derived from the data list
  sets <- length(y_train)
  # Define the number of steps for the cross-validation
  num_steps <- 100
  # Define the maximum parameter value
  max_param <- 50
  # Initialize an empty vector to store the parameters to check
  parameters <- numeric(num_steps)
  # Generate parameters with increasing step sizes
  for (i in 1:num_steps) {
    parameters[i] <- max_param * (1 - exp(-i^3/num_steps^3))
  }
  parameters[1] <- 0  # The first parameter to check should be 0
  # Assign the range of lambda values for lambda1 and lambda2
  lambda1_values <- parameters
  lambda2_values <- parameters
  # Initialize an empty matrix to store the mean MSE for each combination
  mean_mse_matrix <- matrix(NA, nrow = length(lambda1_values),
                            ncol = length(lambda2_values))
  rownames(mean_mse_matrix) <- lambda1_values # Assign lambda vals as row names
  colnames(mean_mse_matrix) <- lambda2_values# Assign lambda vals as col names
  # Add progress bar to see loop progress as this is computationally expensive
  pb <- progress_bar$new(total = length(lambda1_values),
                         format = "[:bar] :percent :elapsed")
  # Nested loop to iterate over lambda1 and lambda2 values (all combinations)
  for (i in (1:length(lambda1_values))) {
    for (j in 1:length(lambda2_values)) {
      # The current lambda combination to check
      lambda1_cv <- lambda1_values[i]
      lambda2_cv <- lambda2_values[j]
      # Vector the store MSEs for current lambda combination across sets
      mse_cv <- c()
      # Iterate over training sets and perform elastic net given lambdas
      for (set in 1:sets) {
        beta_ests <- elastic_net(x_train[[set]], y_train[[set]],
                                 lambda1_cv, lambda2_cv,
                                 tolerance = 1e-4, max_iterations = 10000)
        # The estimated beta for each set is evaluated on each validation set
        mse_val_step <- c() #Store MSE given train beta for each validation set
        for (val_set in 1:sets) {
          # Compute and store MSE for each validation set
          y_hat <- x_val[[val_set]] %
          *% beta_ests
          temp_mse <- sum((y_val[[val_set]] - y_hat)^2) / nrow(x_val[[val_set]])
          mse_val_step <- c(mse_val_step, temp_mse)
        }#MSE for given lambdas in a SET is the mean across validation MSEs
        mse_cv <- c(mse_cv, mean(mse_val_step))
      }# The mean MSE for given lambdas is the mean MSE across each data set
      mean_mse_matrix[i, j] <- mean(mse_cv) #assign mean mse to matrix
    }
    pb$tick() # update the progress bar
  }
  pb$terminate() # terminate progress bar after the loop


  # Find the combination of lambda1 and lambda2 with the minimum mean MSE
  min_mse <- which(mean_mse_matrix == min(mean_mse_matrix), arr.ind = TRUE)
  optimal_lambda1 <- lambda1_values[min_mse[1]] # store optimal lambda1
  optimal_lambda2 <- lambda2_values[min_mse[2]] # store optimal lambda2
  # Return optimal lambdas and corresponding minimal MSE as a named list
  return(list(optimal_lambda1 = optimal_lambda1,
              optimal_lambda2 = optimal_lambda2,
              min_mse = min_mse)) }
```

Listing 2.3: R-code to perform the Cross Validation for the Elastic Net

2.3 Simulation Settings and Numerical Results

2.3.1 Initial Outline

The data utilized in this investigation adheres to the outline stipulated in the *Group Project Instruction* handout. Utilizing the model described in equation (2.2), we generate 50 training and 50 validation sets, each comprising 20 observations. In addition, we produce one test set encompassing 200 observations. The predictor values for each set are synthesized using the `rmvnorm()` function, which permits the specification of co-linearity between data points x_i and x_j according to corr$(i, j) = 0.5^{|i-j|}$. Moreover, we set $\sigma = 3$, $\boldsymbol{\beta} = (3, 1.5, 0, 0, 2, 0, 0, 0)^T$, and normalize the y values. This methodology is referred to as the *standard procedure* throughout the report. Any investigations involving modifications to these settings are explicitly delineated.

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim N(0, \boldsymbol{I_n}) \tag{2.2}$$

Each training set is utilized to estimate a predictor vector $\boldsymbol{\beta}$, which is then employed to predict values of y for each validation set. The Mean Squared Error (MSE) is ascertained for each validation set, and we opt for the one associated with the lowest average MSE.

For the customary scenario where $n > p$, it is empirically observed that the prediction efficacy of LASSO is outperformed by ridge regression [10]. The *standard procedure* engenders a dataset satisfying $n > p$. Table 2.1 presents the results obtained by both algorithms under the *standard procedure*. It is evident that the elastic net behaves akin to a LASSO by assigning $\lambda_2 = 0$, resulting in identical $\boldsymbol{\beta}$ outputs and MSE values, rendering the two algorithms indistinct in terms of their outcomes. Consequently, our results are not in line with the findings of Tibshirani [10], as the LASSO term appears to dominate over the Ridge Regression term within the elastic net.

Table 2.1: Results of LASSO and elastic net on the dataset produced using the *standard procedure*.

Algorithm	MSE	λ_1	λ_2	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
LASSO	0.54	1.33	NA	0.31	0.22	0	0.19	0	0.18	-0.14	-0.17
Elastic Net	0.54	1.33	0	0.31	0.22	0	0.19	0	0.18	-0.14	-0.17

2.3.2 Varying the Number of Predictors (p)

To examine the effect of the number of predictors p on model performance, we start with the *standard procedure* as previously defined. We iterate over an array of $\boldsymbol{\beta}$ vectors, beginning with a base vector $\boldsymbol{\beta} = (3, 1.5, 2, 1)^T$ and subsequently appending it to the $\boldsymbol{\beta}$ of the preceding iteration. This procedure yields values of p ranging from 4 to 36 in increments of 4. Results for the last step are noted in 2.2.

Table 2.2: Results of LASSO and elastic net for $p = 36$.

Algorithm	MSE	λ_1	λ_2
LASSO	0.291	0.006	NA
Elastic Net	0.170	0	4.083

Figure 2.3 illustrates a significant divergence in the performance of the Elastic Net and LASSO algorithms. Notably, as p increases, the MSE for LASSO rises sharply, particularly for $p > n$, the number of observations. This observation corroborates the empirical evidence suggesting that LASSO's variable selection is limited to at most n predictors before reaching saturation. Once this saturation point is reached, LASSO cannot incorporate additional predictors, leading to a deterioration in the accuracy of $\boldsymbol{\beta}$ predictions as p grows relative to n [10]. In contrast, the Elastic Net exhibits a more resilient performance in scenarios where $p > n$, implying the dominance of its Ridge Regression component over the LASSO component in such contexts.

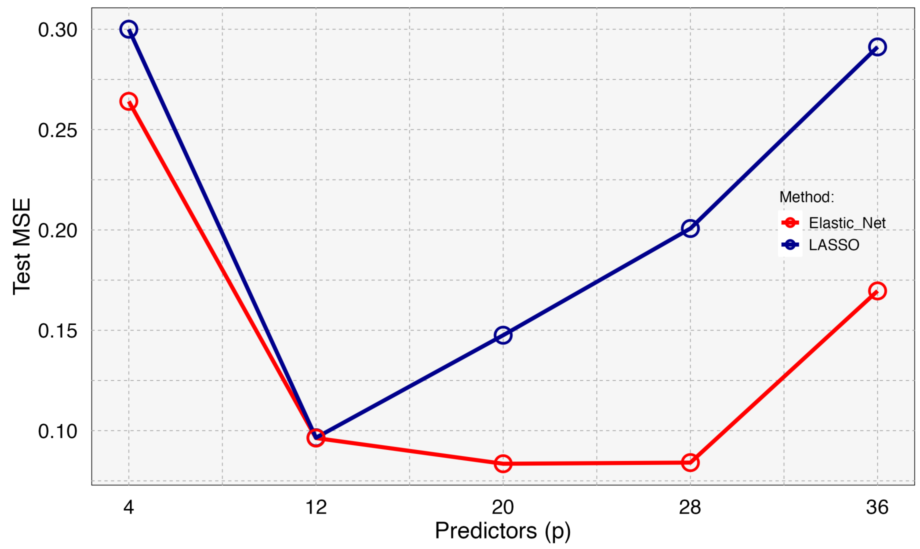

Figure 2.1: MSE plotted according to the *standard procedure* with varying number of predictors

2.3.3 Varying Noise (σ)

To assess the impact of noise amplitude σ on LASSO and elastic net performance, we compute the test MSE for an assortment of σ magnitudes. Figure 2.2 delineates a marked ascending trend in MSE for both methodologies as σ intensifies. The negligible discrepancy in the performance of the algorithms across the examined dataset implies that neither method can be deemed superior in contending with noise.

2.3.4 Varying the number of data points (n)

Analyzing the impact of the number of data points on model accuracy, we observe from Figure 2.3 that the Test Mean Squared Error (MSE) for both LASSO and Elastic Net initially decreases with an increasing number of observations. This decrease in MSE underscores the improvement in prediction accuracy as more data becomes available for model training.

As the number of data points continues to grow, the performance of the two algorithms becomes identical. This occurs as the elastic net model sets $\lambda_2 = 0$ and becomes a LASSO algorithm (see Table 2.3). The MSEs of both algorithms stabilize and then experience a modest upswing, indicating potential overfitting or lack of model robustness with larger datasets.

Table 2.3: Results of LASSO and elastic net for $n = 100$. $\beta_i = 0.00$ are non-zero for higher precision, however, only 2 decimal points are shown here.

Algorithm	MSE	λ_1	λ_2	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8
LASSO	0.28	2.66	NA	0.49	0.27	0.01	0.00	0.30	0.01	0.00	0.00
Elastic Net	2.66	1.33	0	0.49	0.27	0.01	0.00	0.30	0.01	0.00	0.00

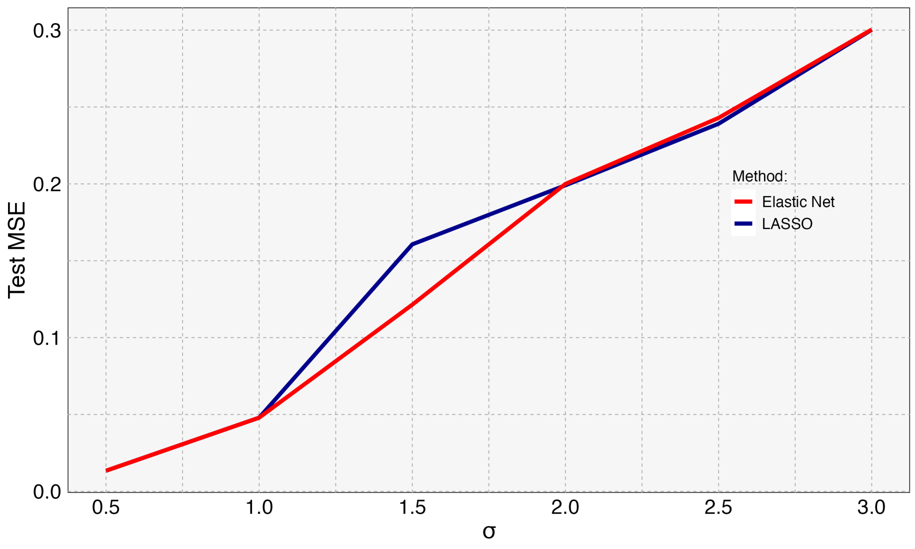

Figure 2.2: MSE plotted according to the *standard procedure* with varying noise amplitudes

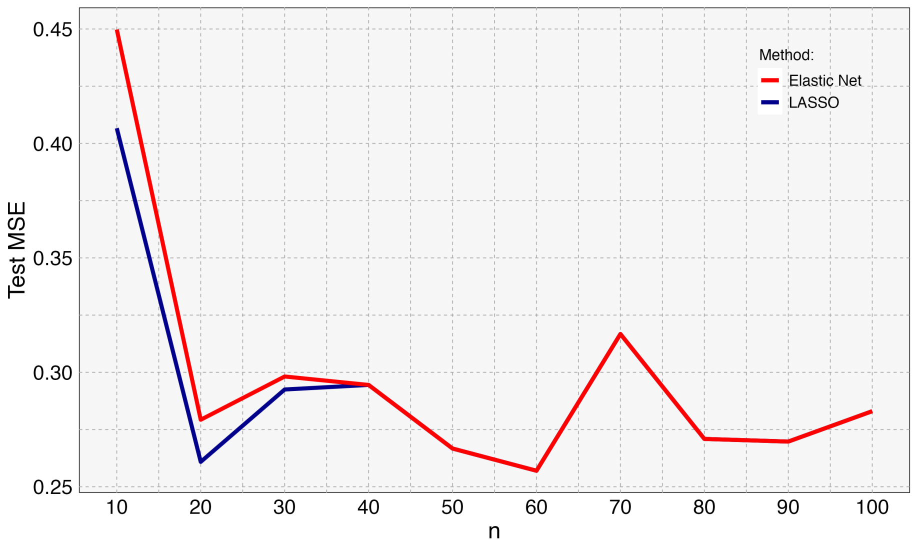

Figure 2.3: MSE plotted according to the *standard procedure* with varying number of data points.

2.3.5 Varying the correlation (ρ)

Our analysis investigates how the LASSO and Elastic Net methods handle variable selection amidst varying predictor correlation levels. A prevalent hypothesis suggests that LASSO often favours selecting a single variable from highly correlated groups, ignoring others [10]. This idea finds subtle support in Table 2.4: As correlation increases, LASSO tends to zero out more coefficients than the Elastic Net. Surprisingly, this does not impact overall model performance, as indicated in Figure 2.4, where the Elastic Net does not outperform LASSO based on the MSE metric.

Table 2.4 provides a detailed account of the number of times each β element has been estimated to be zero across the 50 β vectors, which were obtained by using the training set for different values of correlation in the generated x data. These counts further substantiate the observation that LASSO frequently imposes zero estimates compared to Elastic Net, particularly as the correlation among predictors increases. However, at extreme correlations ($\rho = 0.9$), Elastic Net stops zeroing out β elements, which may indicate an overshrinking bias in highly correlated contexts.

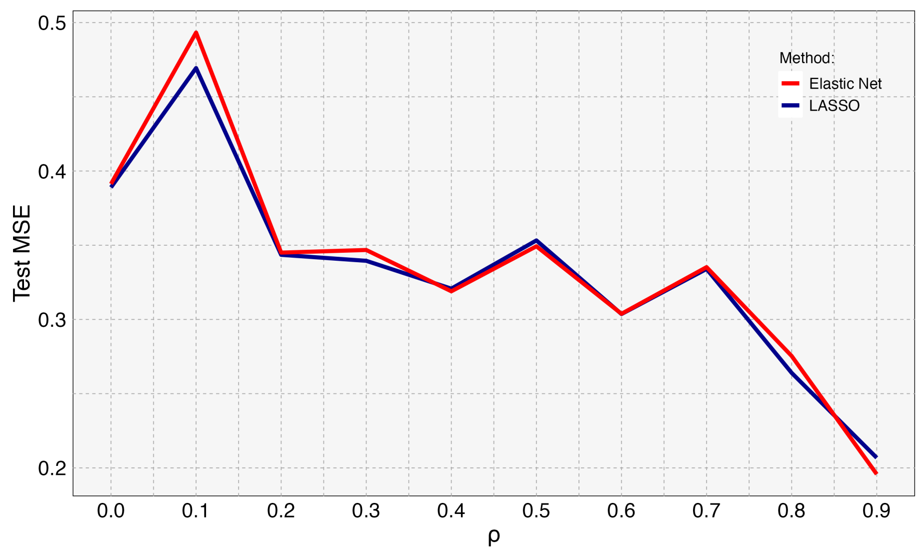

Figure 2.4: MSE plotted according to the *standard procedure* with varying correlation between generated x values

Table 2.4: Number of times each β element has been estimated to be 0 in the set of β vectors estimated by using the training set for different values of correlation between the generated x data.

Correlation	LASSO	LASSO Total	Elastic Net	Elastic Net Total
0.0	(0 2 4 1 1 5 2 4)	19	(0 2 4 1 1 5 2 4)	19
0.1	(0 0 7 4 0 6 3 4)	24	(0 0 6 2 0 4 2 2)	16
0.2	(0 0 4 5 0 5 6 6)	26	(0 0 3 4 0 5 5 5)	22
0.3	(1 1 7 7 0 6 5 7)	34	(0 0 4 4 0 3 2 5)	18
0.4	(0 1 7 5 0 6 8 6)	33	(0 1 7 5 0 6 8 6)	33
0.5	(0 0 8 5 0 6 3 6)	28	(0 0 8 5 0 6 4 7)	30
0.6	(1 0 5 6 1 4 6 7)	30	(0 0 1 1 0 1 1 3)	7
0.7	(0 4 8 5 2 6 3 5)	33	(0 0 2 0 0 1 0 1)	4
0.8	(0 5 6 6 2 7 9 7)	42	(0 4 6 5 2 7 9 6)	39
0.9	(2 1 7 6 4 6 8 6)	40	(0 0 0 0 0 0 0 0)	0

2.4 Summary

In this report, we delved into the Coordinate Descent Algorithm as an effective tool for solving the LASSO problem, building upon work by Tseng et al, Friedman et al, and Tibshirani [8][9][10]. Through the practical application and implementation in R, we have not only tackled the optimization challenges of the LASSO but also explored the enhancements afforded by the Elastic Net regularization.

Our computational experiments have yielded several insights. We observed that while LASSO is proficient with a smaller number of predictors, its performance is hindered as the predictor count increases, especially when the number of predictors outstrips the number of observations. This was evidenced by the increase in Mean Squared Error (MSE), which aligns with the theoretical constraints on LASSO's variable selection capabilities.

In contrast, the Elastic Net has demonstrated notable robustness across various numbers of predictors, showcasing its superiority in high-dimensional settings. This robustness is attributed to the integration of the L1 and L2 penalties, which manage the increase in variance effectively, a scenario where LASSO alone falls short.

The crucial role of regularization parameter selection via cross-validation has been highlighted in our study. The selection of λ values critically impacts the models' balance between complexity and generalization error. The Elastic Net, in particular, offers a more adaptive regularization approach than LASSO, allowing for a more delicate balance between bias and variance.

To summarize, the Elastic Net's combination of L1 and L2 penalties provides a strategic edge over the LASSO, particularly in high-dimensional spaces prone to multicollinearity. Our empirical findings lend support to the theoretical advantages of the Elastic Net for complex predictive modelling, underscoring the significance of regularization techniques in the landscape of statistical analysis.

Bibliography

[1] Sloan Digital Sky Survey DR17. Stellar classification dataset - sdss17, January 2022. URL `https://www.kaggle.com/fedesoriano/stellar-classification-dataset-sdss17`. Accessed: December 23, 2023. 2

[2] The National Aeronautics and Space Administration (NASA). Reference systems, 2023. URL `https://science.nasa.gov/learn/basics-of-space-flight/chapter2-2/`. Accessed: December 23, 2023. 2

[3] J.B. Kaler. *Stars and their Spectra: An Introduction to the Spectral Sequence*. Cambridge University Press, 1989. 2

[4] The European Space Agency (ESA). What is 'red shift'?, 2023. URL `https://www.esa.int/Science_Exploration/Space_Science/What_is_red_shift`. Accessed: December 23, 2023. 2

[5] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer, 2013. 2, 3, 4, 5

[6] Zhang Wen-hui Li Chao and Lin Ji-ming. Study of star/galaxy classification based on the xgboost algorithm. *Chinese Astronomy and Astrophysics*, 43(4):539 – 548, 2019. ISSN 0275-1062. URL `https://www.sciencedirect.com/science/article/abs/pii/S0275106219300815`. 5

[7] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. page 785–794, 2016. 5

[8] Paul Tseng and Decision Systems. Coordinate ascent for maximizing nondifferentiable concave functions. 1988. URL `https://api.semanticscholar.org/CorpusID:6625221`. 6, 13

[9] Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302 – 332, 2007. doi: 10.1214/07-AOAS131. URL `https://doi.org/10.1214/07-AOAS131`. 6, 13

[10] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 00359246. URL `http://www.jstor.org/stable/2346178`. 7, 10, 11, 12, 13