Course Project

ST443 Machine Learning and Data Mining

Andreas Stöffelbauer

10 December 2020

Contents

1. Real World Data

My team members have completed this part of the project, so I did not include it here.

2. Coordinate Descent Algorithm for Solving the Lasso Problems

This is the part of the project that I have done.

2.1 Introduction

While ordinary least squares (OLS) regression is used for a variety of situations, it suffers from two major shortcomings. It has low **prediction accuracy** (low variance but potentially very large bias) and, if there are many features, its **interpretability** is poor. Instead, we would often like a smaller subset of the predictors that explain the response variable.

The lasso, proposed by Tibshirani (1996), was developed in part to address to these shortcomings. However, the lasso suffers from certain limitations itself, i.e. it can perform badly in some scenarios, especially in the high dimensional case of $p > n$ or if there are correlated predictors. Consequently, the elastic net penalty was developed with the goal of achieving higher prediction accuracy while still maintaining the lasso's advantage as a continuous variable selection method. The goal of this project report is to show that

- the elastic net can dominate the lasso in terms of prediction accuracy, and
- like the lasso, the elastic net performs variable selection.

Both the lasso and the elastic net estimates are defined as the arg min of a penalized residual sum of squares, which is often written in the following form. Note that $\alpha = 1$ corresponds to lasso regression and $\alpha = 0$ corresponds to ridge regression. For intermediate α, we get a mixed penalty (the elastic net penalty).

$$\begin{aligned}\hat{\beta} &= \arg\min_{\beta}\{\frac{1}{2}\|Y - X\beta\|_2^2 + \lambda\left(\alpha\|\beta\|_1 + (1-\alpha)\|\beta\|_2^2\right)\} \\ &= \arg\min_{\beta}\{\frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \sum_{j=0}^{p} x_j^{(i)}\beta_j\right)^2 + \lambda\left(\alpha\sum_{j=0}^{p}|\beta_j| + (1-\alpha)\sum_{j=0}^{p}\beta_j^2\right)\}\end{aligned}$$

2.2 Methodology and Parameter Tuning

The goal is to compare the performance of two statistical learning methods - not on one specific data set, but in general. Therefore, a simulation experiment is used in order to repeat the experiment multiple times, observe the behavior of the methods, and average the results. As we want both methods to perform at their best, an important step in this procedure is the parameter tuning.

In fact, there are several ways to find the optimal regularization parameter λ for the lasso, or λ and α for the elastic net. Two such methods are using an information criterion and cross validation. We first briefly discuss why we do not choose these methods here.

Why we do not use an information criterion. One big drawback of using an information criterion such as C_p or BIC is that it requires us to specify the (effective) degrees of freedom df_λ, which can be difficult for regularized models. (For the lasso, the number of non-zero parameters is an unbiased and consistent estimator. However, this is not true for the elastic net). In fact, using an information criterion is most useful when we do not have a validation and/or test set. Since we do know how the data is generated in this experiment, we simply use separate data sets for parameter tuning (validation data) and model evaluation (test data).

Why we do not use cross validation. Cross validation works by splitting the data into folds (typically 5 or 10) and using each fold as a hold-out validation set. Like cross validation, however, it is also most useful when there is no explicit validation and/or test set and is therefore not necessary in a simulation experiment where we can simply generate multiple data sets from the data generating distribution.

Therefore, we use the setting which is also suggested in the instructions. That is, we simulate 50 data sets $\tau^{(1)}, ..., \tau^{(50)}$ and split each of them into a training, validation, and test set, i.e. $\tau^{(i)} = [\tau^{(i)}_{train}, \tau^{(i)}_{val}, \tau^{(i)}_{test}]$. For example, we may choose sample sizes $n_{train} = n_{val} = 20$ and $n_{test} = 200$.

Procedure. More specifically, we (1) fit the model on the training data, (2) use the validation data to find the optimal hyper-parameter(s), and (3) evaluate the performance of the optimal model on the test data. We apply this procedure to all 50 data sets and to both the lasso and the elastic net in order to obtain an estimate of their performance. For the lasso, we fix $\alpha = 1$ and consequently only optimize over a λ-sequence in step (2) whereas for the elastic net we perform a two-dimensional grid-search over each combination of a λ- and α-sequence.

2.3 Performance Metrics

The most important performance indicator for a statistical learning method arguably is prediction accuracy. Consequently, we use the MSE (specifically, the average of the 50 MSEs) in combination with the corresponding standard error. In addition, we also track the number of non-zero coefficients, how often the two methods choose the correct model (exactly), and how often the two methods choose the correct parameters (among others).

To summarize, we

- calculate $\hat{MSE} = \frac{1}{50}\sum_{i=1}^{50} \hat{MSE}^{(i)}(\tau^{(i)}_{test})$ where each $\hat{MSE}^{(i)}(\tau^{(i)}_{test}) = \frac{1}{50}\sum_i (Y^{(i)} - X\hat{\beta}^{(i)}_{best})^2$
- count the number of non-zero coefficients
- count how often the methods *choose* the right model
- count how often the methods *include* the right parameters among others

Which MSE exactly we are trying to estimate. Since we repeatedly generate training, validation, and test data, we effectively average over the randomness in the data, giving us an estimate of the *expected prediction error* $Err = E\{L(Y, \hat{f}(X)\}$ and not the prediction error conditional on a given data set $Err_\tau = E\{L(Y, \hat{f}(X)|\tau\}$ (also called the *generalization error*), where L is a loss function such as squared error. The latter is typically what is most relevant in practice (as we want to know how well a model trained on a given data set performs), but the former is what we need for comparing two learning algorithms. Interestingly, cross validation also effectively estimates the expected prediction error rather than the generalization error - see ESL Chapter 7.12 (Hastie, T. et al, 2009).

2.4 Coordinate Descent Algorithm

Tseng (1998) shows that component wise coordinate descent can be used to minimize a function

$$g(x) = q(x) + \sum_j p_j(x)$$

where $q(x)$ is continuously differentiable and convex and each $p_j(x)$ is convex. As Friedman et al. (2007) show, this result is applicable to the lasso problem, which aims to minimize the l_1-penalized residual sum of squares. In fact, it can also be used for the elastic net penalty. Essentially, the problem reduces to a component-wise, cyclic minimization problem. The coordinate updates are done using the soft threshold function outlined below (Friedman, J. et al, 2009). Note that in the case of $\alpha = 1$, this reduces to the lasso soft threshold update.

$$\beta_j = \frac{sign(\beta^*)(|\beta^*| - \lambda\alpha)_+}{1 + \lambda(1-\alpha))}$$

Implementation of the Coordinate Descent Algorithm

```
coorDesc = function(X,                # nxp design matrix
                    Y,                # nx1 vector of the response variable
                    lambda,           # regularization parameter
                    alpha = 1,        # alpha = 1 corresponds to LASSO
                    tol = 1e-5,       # tolerance
                    max_iter = 100    # max number of iterations
                    ){

  ######################## coordinate descent algorithm ########################

  n = dim(X)[1]; p = dim(X)[2]
  beta = rep(0, p)        # initialize beta
  Y = Y/sd(Y)             # standardize Y
  delta_beta = tol        # stopping criterion #1
  iter = 0                # stopping criterion #2

  while(delta_beta>=tol & iter<max_iter){
    beta_old = beta
    for(j in 1:p){
      beta_star = X[,j] %*% (Y - X[,-j] %*% beta[-j])/n  # LSE on partial residual
      beta[j] = soft(beta_star, lambda, alpha)           # soft threshold update
    }
    iter = iter + 1
    delta_beta = sum(abs(beta-beta_old))
  }
  return(beta)
}
```

```
soft = function(beta, lambda, alpha){
  #### soft threshold function ####
  return(sign(beta) * max(abs(beta)-alpha*lambda, 0) / (1+lambda*(1-alpha)))
}
```

Implementation of the Grid Search Algorithm

```
gridSearch = function(dataset, lambdaSEQ, alphaSEQ = 1){

  ############################# grid search #############################

  # searches over a grid of lambdas and alphas for optimal parameters
  # returns a list with details useful for model evaluation
```

```r
  params = list("lambda" = lambdaSEQ, "alpha" = alphaSEQ)
  grid = expand.grid(params)   # each combination of lambda and alpha
  results = list(lambda=NA, alpha=NA, mse=NA, nonzero=NA,
                 incModel=NA, trueModel=NA, beta=NA)

  # (1) fit models on training data
  betas = map(transpose(grid), ~coorDesc(dataset$X[train,], dataset$Y[train,],
                                         .$lambda, .$alpha))

  # (2) evaluate models on validation data
  msesVal = map_dbl(betas, ~mse(dataset$X[val,], dataset$Y[val,], .))
  bestBeta = betas[[which.min(msesVal)]]   # pick best beta based on lowest val mse

  # useful for discussion of the results
  results$lambda = grid$lambda[which.min(msesVal)]
  results$alpha = grid$alpha[which.min(msesVal)]
  results$nonzero = sum(bestBeta!=0)
  results$incModel = all(bestBeta[beta!=0]!=0)
  results$trueModel = results$incModel & results$nonzero == sum(beta!=0)

  # (3) evaluate model on test data
  results$mse = mse(dataset$X[test,], dataset$Y[test,], bestBeta)
  return(results)
}
```

```r
mse = function(X, Y, beta){
  #### mean squared error ####
  return(mean((Y-(X%*%beta)*sd(Y))^2))
}
```

2.5 Simulation Experiment

We test and compare the two methods in four different scenarios. The scenarios are chosen such that we can uncover some of the lasso's weaknesses and show that the elastic net performs better in these cases.

Before that, we first briefly revisit the issue of the parameter tuning. As mentioned before, we use a grid search over each combination of a λ and α sequence. For the lasso, we choose 40 λ values equally spaced on a log-scale between 10^{-3} and 10. For the elastic net, we use the same λ sequence but additionally choose 20 α values linearly spaced between 0.05 and 1. That means, the grid search will perform 40 fits for the lasso and 800 fits for the elastic net. We found that this grid works for all four scenarios.

Also note that the code used in this and the following chapters has been put into the appendix to make it more readable.

Scenario A. Our first scenario resembles the one suggested in the instructions. We have 8 features with $corr(i, j) = 0.5^{|i-j|}$. The true coefficients are $\beta = (3, 1.5, 0, 0, 2, 0, 0, 0)^T$, i.e. three coefficients are non-zero (features 1, 2 and 5). The standard deviation of the error ϵ is chosen to be 3.

Scenario B. Scenario B is identical to scenario A except that now we choose the coefficients to be $\beta_j = 0.85$ for all $i = 1, ...8$.

Scenario C. Our third scenario has three groups of three highly correlated predictors (within-group $\rho = 0.85$, no correlation between groups) and additionally 11 noise features, i.e. 20 features in total. We also use larger data sets here, i.e. $n_{train} = n_{val} = 100$ and $n_{test} = 200$.

Scenario D. We choose $\beta = (1, ..., 1, 0, ..., 0)$ such that the first 25 coefficients are 1 and the last 5 are zero. In this scenario, $p > n$ since we again use a training data set of just size 20. More specifically, we use $n_{train} = n_{val} = 20$ and $n_{test} = 200$. We do not want correlation between variables to play a role here so we set it to zero.

2.6 Discussion of the Results

Scenarios A, B, and C are meant to show that the lasso's performance suffers under collinearity, i.e. under correlated predictors. The reason is that the lasso tends to choose one variable among correlated variables while driving the other coefficients to zero.

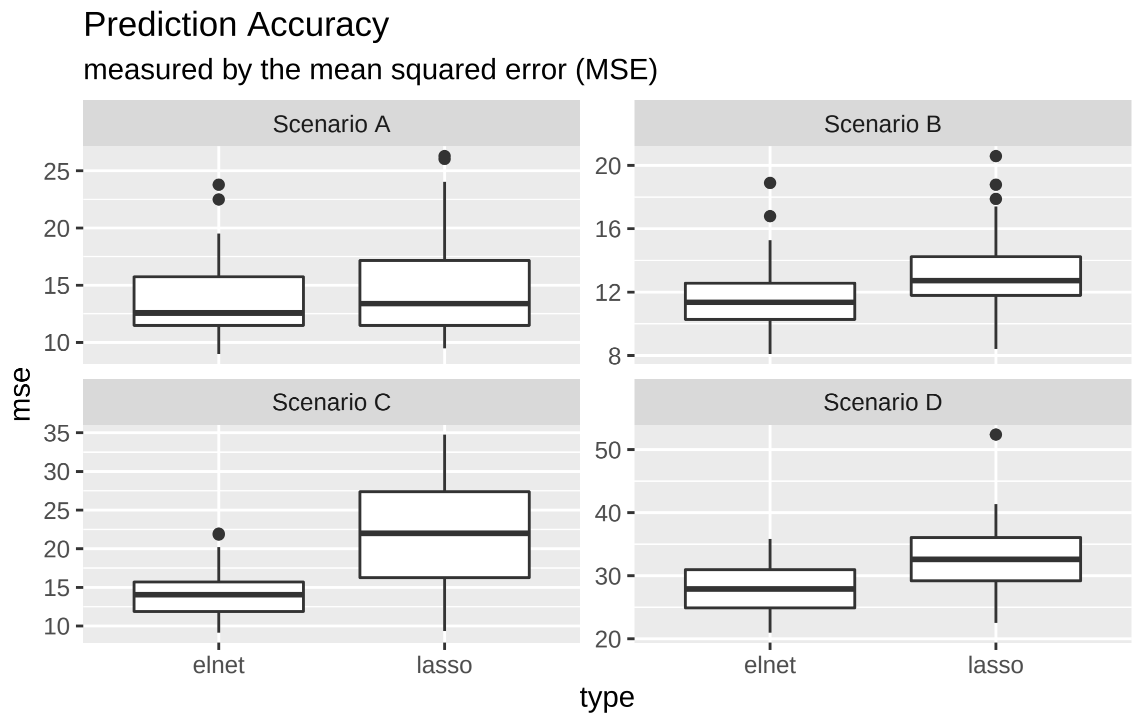

Figure 1

In **Scenario A**, features one and two are correlated. We would therefore expect the lasso to often either choose feature one or feature two but not both, and the elastic net to choose both more frequently. This indeed happens quite often. As can be seen in the table at the end of this section, the elastic net includes the right coefficients (1,2 and 5) 39 times vs only 23 times for the lasso, even though the lasso finds the exactly right model more often (3 vs 0 times). However, the elastic net tends to choose too many variables in this scenario - the median is 6 vs 4 for the lasso (see *Figure 2*). As a result, its prediction accuracy is only slightly (perhaps not significantly) better than the lasso's (see *Figure 1*).

Scenario B is a more extreme version of the same issue. All 8 predictors are nonzero and there is substantial correlation. The lasso tends to set some of the coefficients to zero such that the median of nonzero coefficients is only 5 whereas for the elastic net it is 7, which is closer to the true value of 8 (*Figure 2*). Indeed, the elastic net finds the right model 24 times vs only 8 times for the lasso. Its advantage is therefore much clearer in this scenario. In fact, we can expect that ridge regression would be the method that performs best here because it does not set any coefficients to zero.

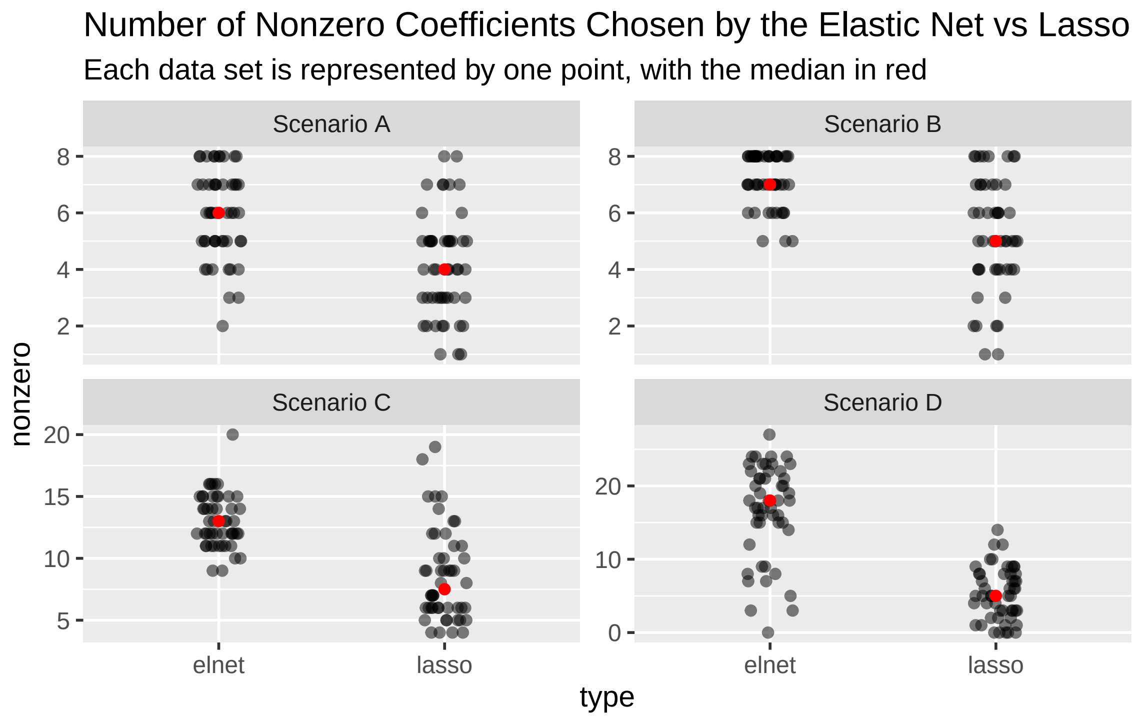

Figure 2

Scenario C is meant to show that the elastic net exhibits the *grouping effect* while the lasso does not. "*Qualitatively speaking, a regression method exhibits the grouping effect if the regression coefficients of a group of highly correlated variables tend to be equal (up to a change of sign if negatively correlated)*" (Zou and Hastie, 2005). The grouping effect can be explained by the fact that the $l2$ penalty drives the coefficients of correlated predictors towards each other while the $l1$ penalty tends to pick only one out of a group of correlated predictors.

In scenario C, there are three groups of correlated predictors. We would therefore expect the lasso to often choose only one from a group and the elastic net to (ideally) choose all three. We can indeed recognize this pattern in Figure 2 above. The lasso chooses fewer than 9 variables quite often (the median is 7.5), sometimes even as few as 4 or 5, while the elastic net tends to choose more (the median is 13, see Figure 2). In fact, it chooses the correct coefficients most of the time (47 times vs only 9 times for the lasso, see table below). As a result, the elastic net again clearly outperforms the lasso in terms of prediction accuracy, as measured by the MSE. Its performance is also much less variable, as we can see in *Figure 1*.

Lastly, **Scenario D** is meant to show the lasso's limitations in the high dimensional setting of $p > n$, i.e. then there are more features than observations. The lasso has a big drawback in that case, namely it chooses at most n variables. This is due to the nature of the convex optimization problem of the lasso. The elastic net, by contrast, can choose more than n variables. As we can see, the median of nonzero coefficients is 18 for the elastic net (close to the true value 20) and only 5 for the lasso (*Figure 2*). However, the lasso chooses fewer coefficients than we would have expect here (see the section on limitations below). Nevertheless, the elastic net again has a lower MSE.

Summary of the Results. We have shown that the elastic net outperforms the lasso in various situations, especially under collinearity and if $p > n$. We have also shown that the elastic net, like the lasso, can produce sparse solutions, but it tends to select more predictors on average because of the additional $l2$ penalty and its grouping effect. We summarize the results again in the following table, which most importantly shows that the elastic net achieved a lower MSE in all four scenarios.

scenario	type	mse	nonzero	trueModel	incModel
Scenario A	elnet	13.59138	6.0	0	39
Scenario A	lasso	14.83118	4.0	3	23
Scenario B	elnet	11.55405	7.0	24	24
Scenario B	lasso	13.19045	5.0	8	8
Scenario C	elnet	14.16486	13.0	1	47
Scenario C	lasso	22.02796	7.5	1	9
Scenario D	elnet	27.81085	18.0	0	0
Scenario D	lasso	32.75806	5.0	0	0

2.7 Limitations and Improvements

Divergence of the coordinate descent. If there is considerable correlation between predictors, the coordinate descent algorithm sometimes diverges, especially for small λ (and in combination with a small training size). This does not, however, influence our results too much as our grid search simply does not choose such a λ. In scenario D, where $p > n$, this issue arises more often, which might explain why the lasso did not quite choose the number of nonzero coefficients we expected, i.e. below but close to 20. Therefore, the true difference in MSE may not be accurately displayed in Figure 1.

Parameter tuning. As mentioned previously, the grid that the methods are optimized on is crucial. Ideally, it should be a large and dense enough grid of values or otherwise sub-optimal (or even bad) parameters are likely. We found that it can be difficult to find an ideal range for both λ and α together so that choosing the grid eventually involved some trial and error. As a result, both methods may be able to achive better prediction accuracy if optimized on an ideal grid.

In addition, a two dimensional grid search is also computationally very intensive. Consequently, our grids could not be as large and dense as may be ideal. In hindsight, we find that a different tuning strategy would have possibly been more suited. For example, Bayesian parameter tuning is a sequential procedure (simply put). In that case, we would first optimize λ (while setting $\alpha = 1$) and conditional on the best λ, we would select the best α. This procedure may not lead to globally optimal parameters, but it would have probably also allowed us to show the elastic net's superiority over the lasso, which is the goal of the project.

Pathwise coordinate descent. A significant improvement in computational efficiency can be made by fitting the model along a path, making use of warm starts for β instead of initializing it as a zero vector for each coordinate descent. More specifically, we should start with the smallest λ for which all β_j are 0, gradually decrease it and use each β as a warm start for the next descent. That way, we would vastly reduce the number of iterations needed for convergence of the coordinate descents algorithm.

One-standard-error rule. Instead of choosing lambda based on the lowest validation MSE, we could use the one-standard-error rule to choose simpler models and perhaps achieve a higher prediction accuracy - but this is true for both the lasso and the elastic net.

2.8 References

Friedman, J., Hastie, T. and Hofling, H. (2007). Pathwise coordinate optimization, The Annals of Applied Statistics 1: 302-332.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso, Journal of Royal Statistical Society, Series B 58: 267-288.

Tseng, P. (1988). Coordinate ascent for maximizing nondifferentiable concave functions, Technical Report.

Zou, H. and Hastie T. (2005). Regularization and variable selection via the elastic net, Royal Statistical Society, Series B 67: 301–320.

Hastie, T., Tibshirani, R. and Friedman J. (2009). The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Springer, New York, NY.

Friedman, J., Hastie, T. and Tibshirani, R. (2009). Regularization Paths for Generalized Linear Models via Coordinate Descent.

3. Appendix

3.1 Appendix for Part 1

3.2 Appendix for Part 2

The code used for parts 2.3 and 2.4 were hidden in the report to make it more readable. Therefore, we include it in the appendix.

```
# libraries used
library(tidyverse)
library(MASS)
library(kableExtra)
```

```
gen_data = function(n, beta, Sigma, sd=3){
  ### generate a data set ###
  epsilon = rnorm(n, sd = sd)
  X = mvrnorm(n, mu = rep(0, length(beta)), Sigma)
  Y = X%*%beta + epsilon
  return(list("X" = X, "Y" = Y))
}
```

```
# scenario A
n = 240; train = 1:20; val = 21:40; test = 41:240
beta = c(3, 1.5, 0, 0, 2, 0, 0, 0)
Sigma = outer(1:8, 1:8, function(i,j) 0.5^abs(i-j))

# data sets for A
datasets = map(1:50, ~gen_data(n, beta, Sigma))

# sequences for lambda and alpha
lambdaSEQ = c(0, 10^seq(-3, 1, length.out = 39))
alphaSEQ = seq(0.05, 1, length.out = 20)

# model fitting
resLASSOA = map_df(datasets, gridSearch, lambdaSEQ)
resELNETA = map_df(datasets, gridSearch, lambdaSEQ, alphaSEQ)
resLASSOA$type = "lasso"; resELNETA$type = "elnet"
resultsA = rbind(resLASSOA, resELNETA)
resultsA$scenario = 'Scenario A'
```

```r
# scenario B
# data sets for B
beta = rep(0.85, 8)
datasets = map(1:50, ~gen_data(n, beta, Sigma))

# model fitting
resLASSOB = map_df(datasets, gridSearch, lambdaSEQ)
resELNETB = map_df(datasets, gridSearch, lambdaSEQ, alphaSEQ)
resLASSOB$type = "lasso"; resELNETB$type = "elnet"
resultsB = rbind(resLASSOB, resELNETB)
resultsB$scenario = 'Scenario B'
```

```r
# scenario C
n = 400; train = 1:100; val = 101:200; test = 201:400
beta = c(rep(3, 9), rep(0, 11))
rho = matrix(rep(0.85, 9), ncol = 3)
Sigma = rbind(cbind(rho, matrix(rep(0, 17*3), ncol = 17)),
              cbind(matrix(rep(0, 9), ncol = 3), rho, matrix(rep(0, 14*3), ncol = 14)),
              cbind(matrix(rep(0, 3*6), ncol = 6), rho, matrix(rep(0, 11*3), ncol = 11)),
              matrix(rep(0, 20*11), ncol = 20))
diag(Sigma) = rep(1, 20)

# data sets for C
datasets = map(1:50, ~gen_data(n, beta, Sigma, 3))

# model fitting
resLASSOC = map_df(datasets, gridSearch, lambdaSEQ)
resELNETC = map_df(datasets, gridSearch, lambdaSEQ, alphaSEQ)
resLASSOC$type = "lasso"; resELNETC$type = "elnet"
resultsC = rbind(resLASSOC, resELNETC)
resultsC$scenario = 'Scenario C'
```

```r
# scenario D
n = 240; train = 1:20; val = 21:40; test = 41:240
beta = c(rep(1, 25), rep(0, 5))
Sigma = diag(rep(1, 30))

# data sets for D
datasets = map(1:50, ~gen_data(n, beta, Sigma, sd=3))

# sequences for lambda and alpha
lambdaSEQ = c(0, 10^seq(-3, 1, length.out = 39))
alphaSEQ = seq(0.05, 1, length.out = 20)

# model fitting
resLASSOD = map_df(datasets, gridSearch, lambdaSEQ)
resELNETD = map_df(datasets, gridSearch, lambdaSEQ, alphaSEQ)
resLASSOD$type = "lasso"; resELNETD$type = "elnet"
resultsD = rbind(resLASSOD, resELNETD)
resultsD$scenario = 'Scenario D'
```

```r
# MSE plot
results = rbind(resultsA, resultsB, resultsC, resultsD)
```

```
ggplot(results) +
  geom_boxplot(aes(type, mse)) +
  facet_wrap(~scenario, scales = "free_y") +
  labs(title = "Prediction Accuracy",
       subtitle = "measured by the mean squared error (MSE)", caption = "Figure 1")
```

```
# Nonzero coefficients plot
Summary = results %>%
  group_by(scenario, type) %>%
  summarise(mse = mean(mse),
  nonzero = median(nonzero),
  trueModel = sum(trueModel),
  incModel = sum(incModel))

ggplot(results, aes(type, nonzero)) +
  geom_jitter(width = 0.10, height = 0, alpha = 0.5) +
  geom_point(aes(type, nonzero), data = Summary, col='red') +
  facet_wrap(~scenario, scales = "free_y") +
  labs(title = "Number of Nonzero Coefficients chosen by the Elastic Net vs Lasso",
       subtitle = "Each data set is represented by one point, with the median in red",
       caption = "Figure 2")
```

Appendix

We want to solve the following optimization problem.

$$\begin{aligned}\hat{\beta} &= \arg\min_{\beta}\{\frac{1}{2}\|Y - X\beta\|_2^2 + \lambda\left(\alpha\|\beta\|_1 + (1-\alpha)\|\beta\|_2^2\right)\}\\ &= \arg\min_{\beta}\{\frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \sum_{j=0}^{p} x_j^{(i)}\beta_j\right)^2 + \lambda\left(\alpha\sum_{j=0}^{p}|\beta_j| + (1-\alpha)\sum_{j=0}^{p}\beta_j^2\right)\}\end{aligned}$$

In order to highlight the component-wise nature of the optimization problem, the above can be rewritten as

$$\arg\min_{\beta}\frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \sum_{k\neq j} x_k^{(i)}\beta_k - x_j^{(i)}\beta_j\right)^2 + \lambda\left(\sum_{k\neq j}\alpha|\beta_k| + (1-\alpha)\beta_k^2\right) + \lambda\left(\alpha|\beta_j| + (1-\alpha)\beta_j^2\right)$$

In the next step, we derive the gradient with respect to β.

$$\begin{aligned}\frac{d}{d\beta_j}RSS_{OLS}(\beta) &= \frac{d}{d\beta_j}\frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \sum_{k\neq j}^{p} x_k^{(i)}\beta_k - x_j^{(i)}\beta_j\right)^2\\ &= -\sum_{i=1}^{n}\left(y^{(i)} - \sum_{k\neq j}^{p} x_k^{(i)}\beta_k - x_j^{(i)}\beta_j\right)x_j^{(i)}\\ &= -\sum_{i=1}^{n}\left(y^{(i)} - \sum_{k\neq j}^{p} x_k^{(i)}\beta_k\right) + \beta_j\sum_{i=1}^{n}(x_j^{(i)})^2\end{aligned}$$