

Exploring Regularization Techniques for Small Data Sets

Ian Dardik, Heeru Ahuja
CS6140

1 Introduction

In this paper we focus on regularization techniques that improve linear regression modeling on small data sets. The classic Ordinary Least Squares (OLS) method becomes numerically unstable as the number of samples decrease, and has no solution when the number of parameters exceed the sample size. Statistical regularization can be applied to help stabilize the solution to a linear regression problem and make modeling feasible for small data sets. In this paper we will explore and compare several options for building a regularized linear regression model in the context of a data set whose sample size is small compared to the number of parameters.

2 Methods

Statistical regularization helps stabilize our model and reduce the variance, which becomes increasingly important as the number of samples shrink. Unfortunately, regularization introduces bias into our model so we must choose an appropriate balance. The models we consider in this paper control the degree of regularization through one or two tuning parameters that are completely suppressed at zero, and influence our model more as they increase towards infinity. We can estimate the ideal tuning parameter values by using cross-validation to estimate the Mean Squared Error (MSE) for a small universe of tuning parameters, and select the parameters that minimize MSE. Each regularization technique that we consider is formulated as an optimization problem that minimizes an objective function J of the form:

$$\hat{\theta} = \arg \min_{\theta} J(\theta) = \arg \min_{\theta} \sum_{i=1}^N L(\theta) + \sum_{i=1}^P R(\theta, \lambda)$$

Where $L(\theta)$ is the loss function and $R(\theta, \lambda)$ is the regularization rule that depends on the tuning parameter λ , and sometimes an additional tuning parameter α . We often refer to the regularization rule as the *penalty*. We only consider squared loss in this paper, i.e. $L(\theta) = (Y_i - X_i'\theta)^2$, however we consider and discuss several options for choice of the regularization rule. The regularization techniques that we consider are:

- L1 penalty (LASSO)
- L2 penalty (Ridge)
- Elastic Net
- L2 with Custom Soft-Thresholding
- L_2^1 penalty (not included in our Results section)

Many of the regularization techniques we discuss do not have a closed analytical solution, so we will consider two algorithms for numerically arriving at a solution: Batch Gradient Descent and Pathwise Coordinate Descent. The following sections detail how to use each regularization rule with each algorithm.

For the remainder of this paper we assume that all data X and responses Y are standardized.

2.1 Gradient Descent

Gradient Descent algorithms work by moving in the direction that the gradient of θ points, with distance scaled by a learning rate parameter α . The algorithm continuously updates its value for the parameter vector θ until convergence. Gradients point towards local optima, so we clearly need to use convex objective functions to arrive at a globally optimal solution in Gradient Descent. Fortunately, computing the gradient of objective functions listed above is straightforward which makes Gradient Descent fairly simple to implement. Notice that Gradient Descent works by computing the gradient; it does not itself solve an optimization problem per se; since the algorithm is only dependent on the gradient of the objective function, we can parameterize it based on the gradient. Therefore we end up with a core algorithm the form: `GRADIENT_DESCENT($\nabla\theta$)`

2.2 Pathwise Coordinate Descent

While Gradient Descent operates on the entire parameter vector at once, Pathwise Coordinate Descent (PCD) concentrates on each parameter individually. Pathwise Coordinate Descent considers each parameter θ_j as its own optimization problem[1]:

$$\hat{\theta}_j = \arg \min_{\theta_j} J(\theta)$$

In each iteration of the PCD algorithm, we update each θ_j to be the solution to the individual optimization problem.

2.2.1 Parameterizing the PCD Algorithm

Pathwise Coordinate Descent is parameterized by the objective function J that it solves for each θ_j . However, for the universe of objective functions we consider in this paper, we can parameterize the PCD algorithm by just two nonnegative parameters we call A and B . For each objective function we consider below, we will show that the solution to optimizing each θ_j takes the form:

$$\theta_j = S \left(\sum_{i=1}^N X_{ij} (Y_i - \sum_{k \neq j}^p X_{ik} \theta_k), B \right) / A$$

Where we define[1] $S(t, B) = \text{sign}(t)(|t| - B)_+$, and $()_+$ is the soft-thresholding operator. It follows that each algorithm reduces to the following parameterized form $\text{PATHWISE_CD}(A, B)$. In the following sections we will show how each of the objective functions we consider fit into this parameterization.

2.2.2 LASSO

LASSO uses the L1 norm for a penalty:

$$\hat{\theta}_j = \arg \min_{\theta_j} \frac{1}{2} \sum_{i=1}^N \left(Y_i - \sum_{k=1}^p x_{ik} \theta_k \right)^2 + \lambda \sum_{k=1}^p |\theta_k|$$

Where $\lambda \geq 0$ is a tuning parameter. Let J be the objective function we want to optimize above. We can solve for θ_j by finding the stationary points of J :

$$\begin{aligned} \frac{\partial J}{\partial \theta_j} &= - \sum_{i=1}^N X_{ij} (Y_i - X_{ij} \theta_j - \sum_{k \neq j}^p X_{ik} \theta_k) + \lambda \text{sign}(\theta_j) \\ &= - \sum_{i=1}^N X_{ij} (Y_i - \sum_{k \neq j}^p X_{ik} \theta_k) + \sum_{i=1}^N X_{ij}^2 \theta_j + \lambda \text{sign}(\theta_j) \\ &= - \sum_{i=1}^N X_{ij} (Y_i - \sum_{k \neq j}^p X_{ik} \theta_k) + (N-1) \theta_j + \lambda \text{sign}(\theta_j) \end{aligned} \tag{1}$$

The last step follows from the fact that our data is standardized, and thus $\sum_{i=1}^N X_{ij}^2 = N-1$.

Set $\frac{\partial J}{\partial \theta_j} = 0$ and, for convenience, let $Q = \sum_{i=1}^N X_{ij} (Y_i - \sum_{k \neq j}^p X_{ik} \theta_k)$ and we get:

$$(N-1) \theta_j + \lambda \text{sign}(\theta_j) = Q$$

To solve for θ_j notice that when $\theta_j \geq 0$, $\theta_j = \frac{Q-\lambda}{N-1}$ so $Q \geq \lambda$. However when $\theta_j \leq 0$, $\theta_j = \frac{Q+\lambda}{N-1}$ so $Q \leq -\lambda$. Notice the following two properties:

- (1) $|Q| \geq \lambda$
- (2) $\theta_j \geq 0$ iff $Q \geq 0$ and $\theta_j \leq 0$ iff $Q \leq 0$

Therefore, we can write the solution for θ_j as:

$$\theta_j = \frac{\text{sign}(Q)(|Q| - \lambda)_+}{N - 1}$$

Where the soft-thresholding operator guarantees property (1), while multiplying by $\text{sign}(Q)$ makes use of (and guarantees) property (2). Recall that $S(t, B) = \text{sign}(t)(|t| - B)_+$, then our final solution is:

$$\theta_j = S \left(\sum_{i=1}^N X_{ij}(Y_i - \sum_{k \neq j}^p X_{ik}\theta_k), \lambda \right) / (N - 1)$$

Notice this yields a solution nearly identical to Elements of Statistical Learning[1]; the difference is that we divide by $N - 1$ and we believe this is likely a typo in the book.

Notice that the soft-thresholding parameter is λ ; this explains why larger λ values causes LASSO to drive parameter values to zero. It is now clear that LASSO performs subset selection due to the choice of regularization penalty *and* algorithm.

Finally, it is clear that the parameterization for LASSO is `PATHWISE_CD(N - 1, λ)`.

2.2.3 Elastic Net

The Elastic Net uses the tuning parameter $\alpha \in [0, 1]$ to compromise between Ridge and Lasso penalization. Ridge penalizes the L2 norm, so Elastic Net solves the following optimization problem:

$$\hat{\theta}_j = \arg \min_{\theta_j} \frac{1}{2} \sum_{i=1}^N \left(Y_i - \sum_{k=1}^p x_{ik}\theta_k \right)^2 + \lambda \sum_{k=1}^p (\alpha \theta_k^2 + (1 - \alpha)|\theta_k|)$$

This math is very similar to LASSO above so we have left the details to the appendix. Our final solution is:

$$\theta_j = S \left(\sum_{i=1}^N X_{ij}(Y_i - \sum_{k \neq j}^p X_{ik}\theta_k), B \right) / A$$

Where $A = (N - 1) + 2\lambda\alpha$, and $B = \lambda(1 - \alpha)$. This equation is already in our desired form, and hence the parameterization is `PATHWISE_CD(N - 1 + 2 $\lambda\alpha$, $\lambda(1 - \alpha)$)`.

2.2.4 Ridge

Ridge Regularization has a closed form analytical solution so it does not require an algorithm. Nevertheless, we will discuss Ridge in the context of PCD to realize why the L2 penalty does not perform subset selection while the L1 does. We can simply use our solution for the Elastic Net with $\alpha = 1$ for the Ridge solution. This implies that $A = (N - 1) + 2\lambda$ and $B = 0$, yielding:

$$\theta_j = S \left(\sum_{i=1}^N X_{ij}(Y_i - \sum_{k \neq j}^p X_{ik}\theta_k), 0 \right) / (N - 1 + 2\lambda)$$

With parameterization `PATHWISE_CD(N - 1 + 2 λ , 0)`. Notice that the soft-thresholding parameter is zero; this implies that *no* soft-thresholding will be applied for Ridge, regardless of the parameter λ . This explains why Ridge tends to shrink parameter values but doesn't drive them towards zero.

2.2.5 Custom Shrinkage Methods

We have found above that LASSO uses λ exclusively for linear shrinkage and soft-thresholding, while Ridge uses λ exclusively for inverse shrinkage. The Elastic Net implicitly combines both shrinkage techniques using two tuning parameters; here we propose three “custom” ideas for combining the shrinkage types using a single tuning parameter. Our hope is that using a single tuning parameter makes it easier to find the optimal bias-variance tradeoff using cross-validation. These custom ideas, inspired by Nearest Shrunken Centroids, apply soft-thresholding to the Ridge solution in the form: $\text{PATHWISE_CD}(N - 1 + 2\lambda, \text{STP}(\lambda))$, where STP is the soft-thresholding penalty function. The three custom ideas we will explore are:

1. Ridge with Soft-Thresholding: $\text{PATHWISE_CD}(N - 1 + 2\lambda, \lambda)$
2. Ridge with Quadratic Soft-Thresholding: $\text{PATHWISE_CD}(N - 1 + 2\lambda, \lambda^2)$
3. Ridge with Exponential Soft-Thresholding: $\text{PATHWISE_CD}(N - 1 + 2\lambda, e^\lambda - 1)$

2.3 Experimental Design

In order to compare each regularization technique, we generated two data sets on which we trained and tested our suite of algorithms. Our models include 27 individual parameters, where $X_0 \dots X_{25}$ are continuous and X_{26} is a categorical variable representing a month of the year. The first model includes four orthogonal design variables:

$$m_{ortho}(X) = 2X_1 + 3X_7^2 + 4X_8 + 2X_{37} + \epsilon_{ortho}$$

Where $\epsilon_{ortho} \sim \mathcal{N}(0, 4)$. Our second model includes two design variables that are each correlated with two parameters:

$$m_{corr}(X) = 2X_1^3 I_{Dec}(X_{26}) + 3X_{23}X_{24} + \epsilon_{corr}$$

Where $\epsilon_{corr} \sim \mathcal{N}(0, 1)$, and the indicator I_{Dec} is one if the given argument is “Dec” and zero otherwise. For both data sets we generated four sets of samples:

1. “ $n < p$ ”: a training data set where the number of samples is smaller than the number of parameters. We chose 20 samples.
2. “ $n = p$ ”: a training data set where the number of samples is approximately equal to the number of parameters. We chose 40 samples even though we have 38 variables in our design matrix. We use 5-fold cross-validation in which we require the number of samples to be divisible by 5; hence the choice of 40 samples.
3. “ $n > p$ ”: a training data set where the number of samples exceeds the number of parameters. We chose 80 samples; we purposely chose a number of samples that exceeds the number of parameters, but is still relatively small.
4. “Test”: a testing data set with a large number of samples for estimating the Mean Squared Error (MSE) of the fitted model. We chose 10,000 samples.

For each data set, we trained each algorithm using a design matrix that is linear in each of the 26 continuous variables, and included 12 extra indicator variables for the categorical month variable. We chose optimal tuning parameters for each algorithm using 5-fold cross-validation and then computed the test MSE on the “Test” data set. Our results are included in the following section.

3 Results

For each algorithm, we have included the Test MSE, parameter subset selection, and the number of correctly and incorrectly selected parameters. In addition, we have included three columns comparing each algorithm to the sklearn library results, where we subtract our implementation value from the library value. We have the following remarks on our results:

- Our LASSO implementation and sklearn have similar performance based on Test MSE. The margin is small, but our implementation outperforms the library on the orthogonal data set in the $n < p$ and $n > p$ case.
- Our algorithms generally outperform the library in terms of subset selection. In particular, Ridge with Exponential Soft-Thresholding either beats or ties the library in subset selection for every case, and does so by a decent margin in the orthogonal case.
- Both our algorithms and the library performed poorly on the correlated data set.
- LASSO with Batch Gradient Descent performed very poorly on subset selection, which is completely expected by the choice of algorithm.

N<p							
Algorithm	Test MSE	Parameters Selected	# Correct	# Incorrect	MSE Diff	# Correct Diff	# Incorrect Diff
LASSO	7.22	1, 4, 8	2	1	0.93	0	3
Ridge w/ST	7.08	1, 4, 8, 11	2	2	1.07	0	2
Ridge w/Quad ST	8.8	1, 8	2	0	-0.65	0	4
Ridge w/Exp ST	9.64	8	1	0	-1.49	1	4
Elastic Net	9.5	8	1	0	-1.35	1	4
LASSO w/BGD	10.36	All	4	34	-2.21	-2	-30
sklearn.linear_model	8.15	1, 4, 5, 8, 9, 11	2	4	0	0	0

N=p							
Algorithm	Test MSE	Parameters Selected	# Correct	# Incorrect	MSE Diff	# Correct Diff	# Incorrect Diff
LASSO	8.2	8	1	0	-1.27	1	3
Ridge w/ST	7.84	1, 8	2	0	-0.91	0	3
Ridge w/Quad ST	7.85	1, 8	2	0	-0.92	0	3
Ridge w/Exp ST	7.99	1, 8	2	0	-1.06	0	3
Elastic Net	7.48	1, 8	2	0	-0.55	0	3
LASSO w/BGD	7.76	All	4	34	-0.83	-2	-31
sklearn.linear_model	6.93	1, 2, 6, 8, 14	2	3	0	0	0

N>p							
Algorithm	Test MSE	Parameters Selected	# Correct	# Incorrect	MSE Diff	# Correct Diff	# Incorrect Diff
LASSO	7.35	1, 7, 8	3	0	0.2	0	3
Ridge w/ST	7.18	1, 7, 8	3	0	0.37	0	3
Ridge w/Quad ST	7.18	1, 7, 8	3	0	0.37	0	3
Ridge w/Exp ST	7.44	1, 7, 8	3	0	0.11	0	3
Elastic Net	7.78	1, 4, 7, 8	3	1	-0.23	0	2
LASSO w/BGD	7.95	All	4	34	-0.4	-1	-31
sklearn.linear_model	7.55	1, 4, 7, 8, 13, 20	3	3	0	0	0

Figure 1: Orthogonal Dataset Results

N<p							
Algorithm	Test MSE	Parameters Selected	# Correct	# Incorrect	MSE Diff	# Correct Diff	# Incorrect Diff
LASSO	1.42	None	0	0	0	0	0
Ridge w/ST	1.42	None	0	0	0	0	0
Ridge w/Quad ST	1.42	None	0	0	0	0	0
Ridge w/Exp ST	1.42	None	0	0	0	0	0
Elastic Net	1.42	None	0	0	0	0	0
LASSO w/BGD	1.42	All	5	33	0	-5	-33
sklearn.linear_model	1.42	None	0	0	0	0	0

N=p							
Algorithm	Test MSE	Parameters Selected	# Correct	# Incorrect	MSE Diff	# Correct Diff	# Incorrect Diff
LASSO	1.53	14, 18, 21, 25	0	4	-0.11	0	-3
Ridge w/ST	1.5	14, 18, 21	0	3	-0.08	0	-2
Ridge w/Quad ST	1.43	14	0	1	-0.01	0	0
Ridge w/Exp ST	1.44	14	0	1	-0.02	0	0
Elastic Net	1.64	2, 7, 14, 18, 21, 24, 25	2	5	-0.22	-2	-4
LASSO w/BGD	1.42	All	5	33	0	-5	-32
sklearn.linear_model	1.42	14	0	1	0	0	0

N>p							
Algorithm	Test MSE	Parameters Selected	# Correct	# Incorrect	MSE Diff	# Correct Diff	# Incorrect Diff
LASSO	1.41	3	0	1	0.01	0	-1
Ridge w/ST	1.43	3, 14, 24	1	2	-0.01	-1	-2
Ridge w/Quad ST	1.47	3, 4, 8, 14, 24	1	4	-0.05	-1	-4
Ridge w/Exp ST	1.42	None	0	0	0	0	0
Elastic Net	1.42	3, 24	1	1	0	-1	-1
LASSO w/BGD	1.42	All	5	33	0	-5	-33
sklearn.linear_model	1.42	None	0	0	0	0	0

Figure 2: Correlated Dataset Results

4 Discussion

While the main focus of this paper is on comparing regularization techniques, it is worth noting that our data suggests that some form of regularization is important to stabilize linear regression. In section 5.4, the training error graphs in figures 4, 8, 12, 13, 16 and 26 show how the MSE can spike as regularization approaches zero, while the training error graphs in figures 3, 10, 24 and 25 show how the MSE can become highly volatile as regularization approaches zero.

Our results do not show a clear choice of algorithm that minimizes prediction error on our data sets. However, Ridge with Exponential Soft-Thresholding does stand out as the best performing algorithm in terms of subset selection. Unfortunately our experimental data is narrow in scope—using just two data sets, one of which saw poor performances across all algorithms—thus more testing is necessary before drawing any conclusions about Ridge with Exponential Soft-Thresholding. There may be merit in continuing to experiment with this technique: the training error graphs in figures 21, 22, and 23 of section 5.4 suggest that we may not have considered the optimal tuning parameter. The algorithms can take a long time to converge as regularization decreases, making it tough to test all of our desired options of tuning parameters.

More generally, it is not clear that any of the custom shrinkage methods we proposed are worthwhile. However it is worth noting that squashing linear/soft-thresholding and inverse shrinkage into a single tuning parameter has a clear advantage: it is easier to implement. While Elastic Net also combines linear/soft-thresholding and inverse shrinkage into a single regularization technique, we found that it was slightly easier to implement cross-validation for a single tuning parameter and *far* easier to graph the results for a single tuning parameter. We experimented with 3D graphs to no avail; we ended up plotting Elastic Net as a function of α values, choosing the best λ value at each value of α (see section 5.4 figures 27-32).

We have also found that choosing tuning parameters via cross-validation alone is not necessarily easy. Our graphs clearly show that training error is not a convex function of tuning parameters, however our analysis would have benefited greatly from a more robust method of choosing our tuning parameters.

Finally, it is worth noting that implementing the algorithms—particularly PCD—was not trivial. Understanding PCD requires understanding the optimization problem that each regularization technique uses; this was not trivial at first but led to many key insights, including the custom shrinkage methods from section 2.2.5.

References

- [1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, The Elements of Statistical Learning 2nd Edition
- [2] Tibshirani's paper

5 Appendix

5.1 Github Repository

5.2 Elastic Net Derivation

5.3 $L_{\frac{1}{2}}$ Norm Derivation

5.4 Cross-Validation Training Results

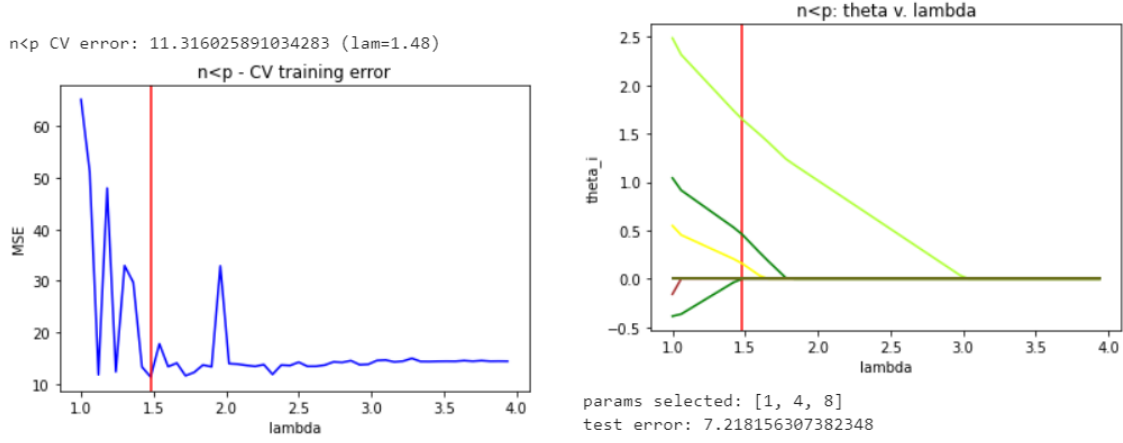


Figure 3: LASSO, $n < p$ orthogonal data set

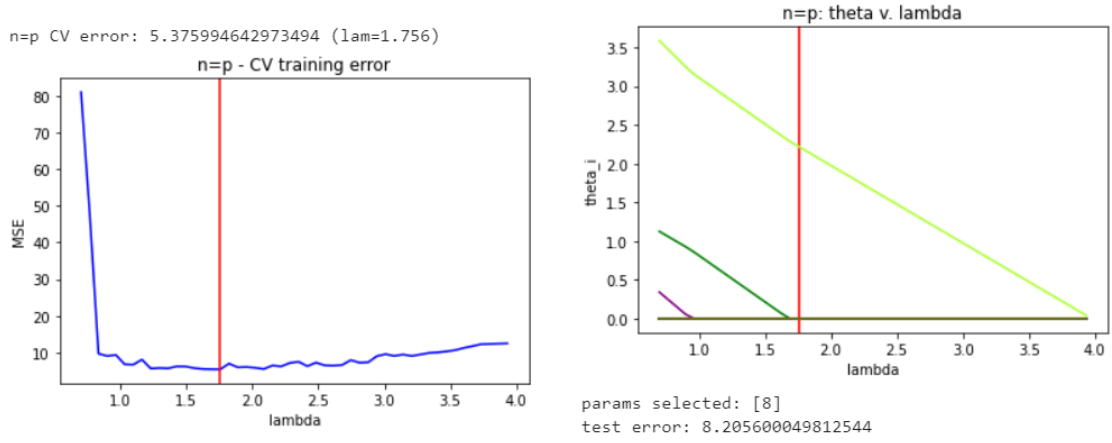


Figure 4: LASSO, $n = p$ orthogonal data set

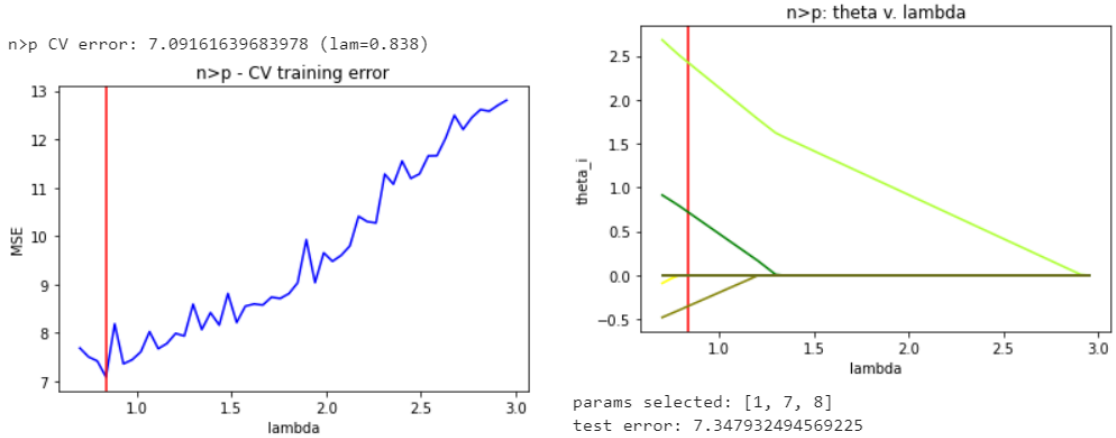


Figure 5: LASSO, $n > p$ orthogonal data set

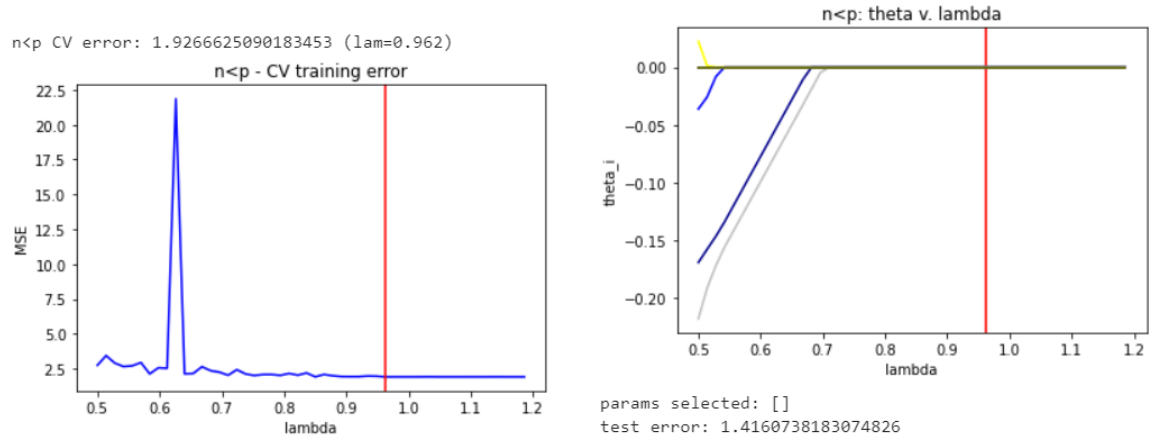


Figure 6: LASSO, $n < p$ correlated data set

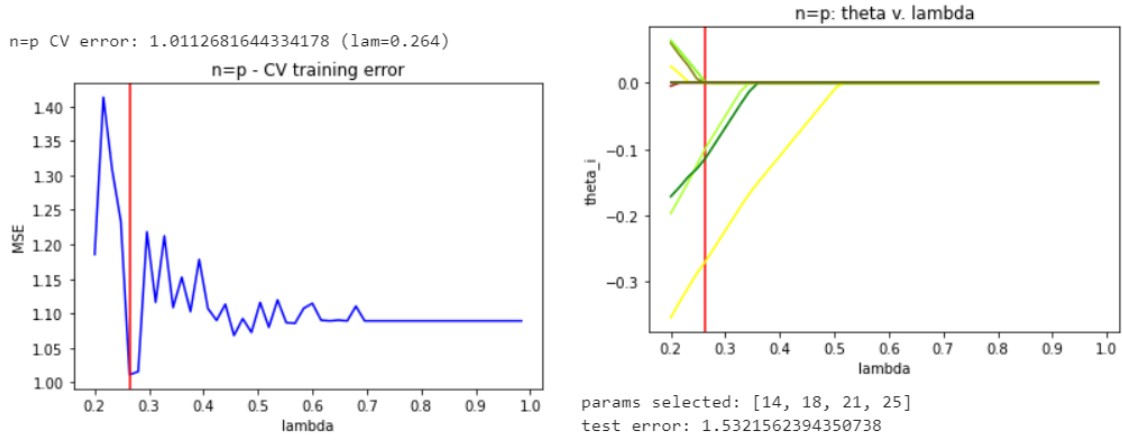


Figure 7: LASSO, $n = p$ correlated data set

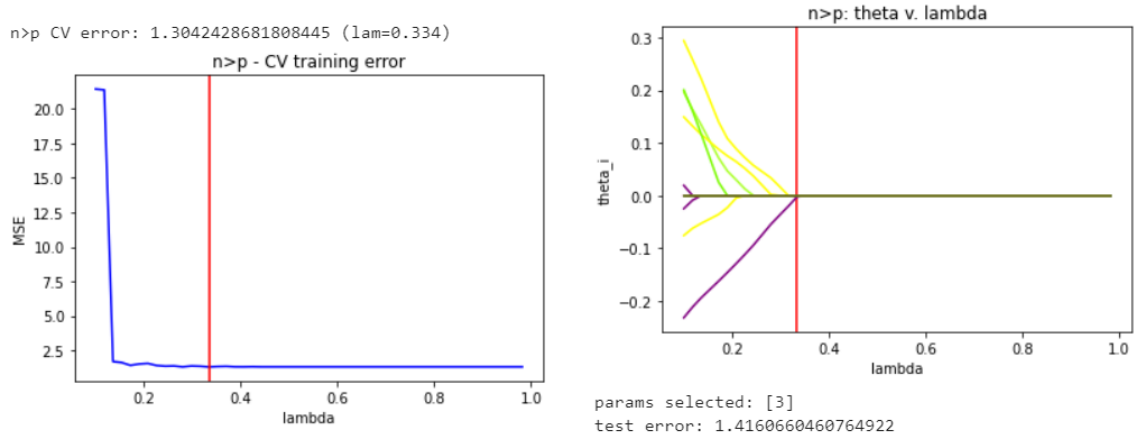


Figure 8: LASSO, $n > p$ correlated data set

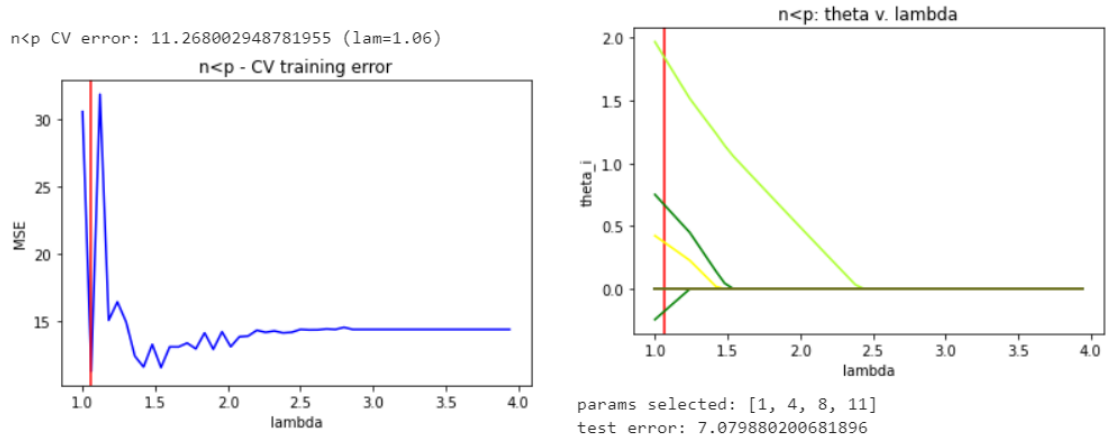


Figure 9: Ridge with Soft-Thresholding, $n < p$ orthogonal data set

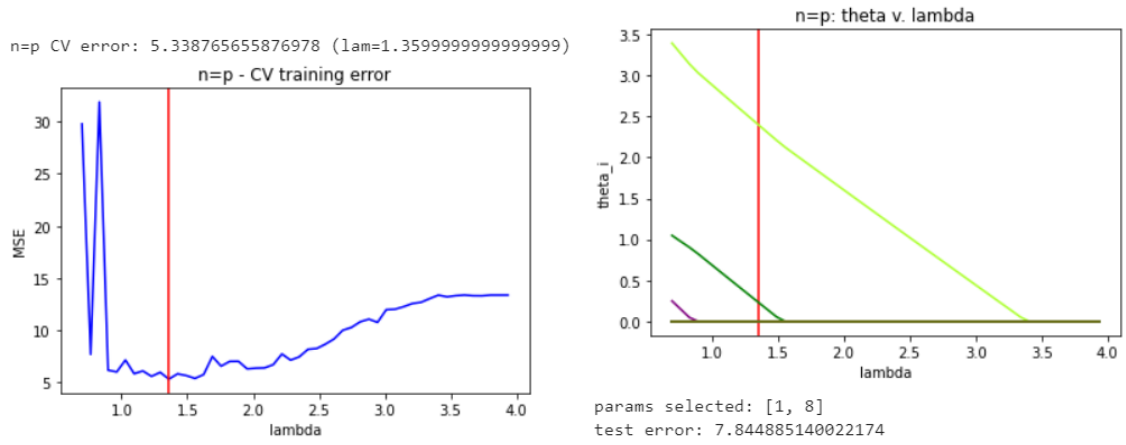


Figure 10: Ridge with Soft-Thresholding, $n = p$ orthogonal data set

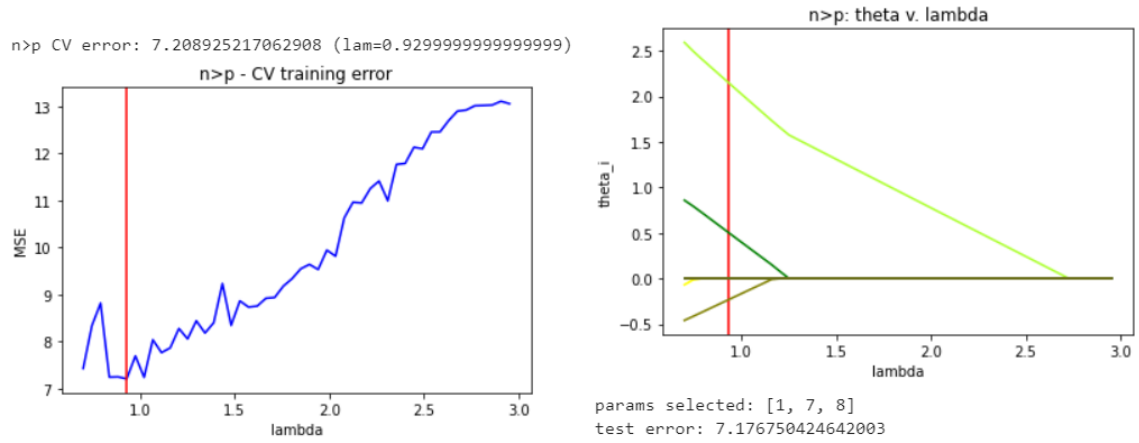


Figure 11: Ridge with Soft-Thresholding, $n > p$ orthogonal data set

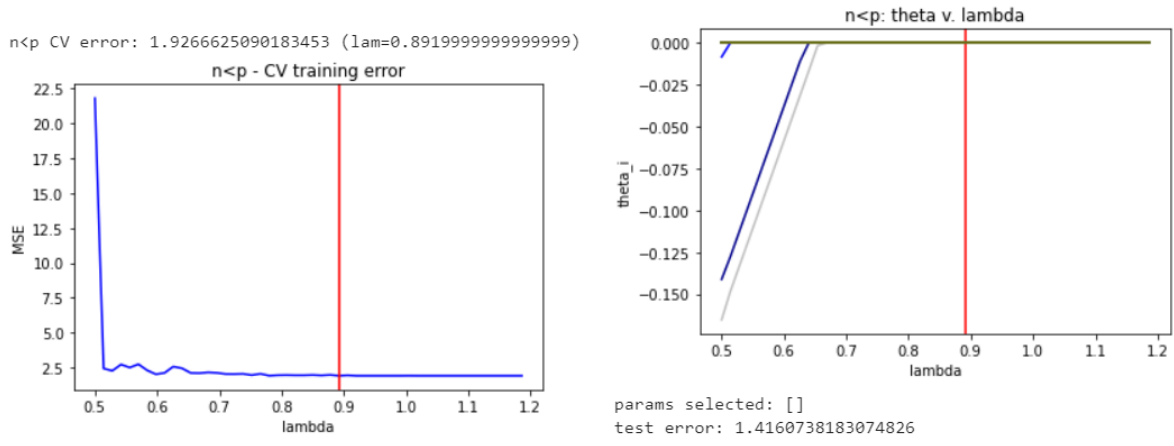


Figure 12: Ridge with Soft-Thresholding, $n < p$ correlated data set

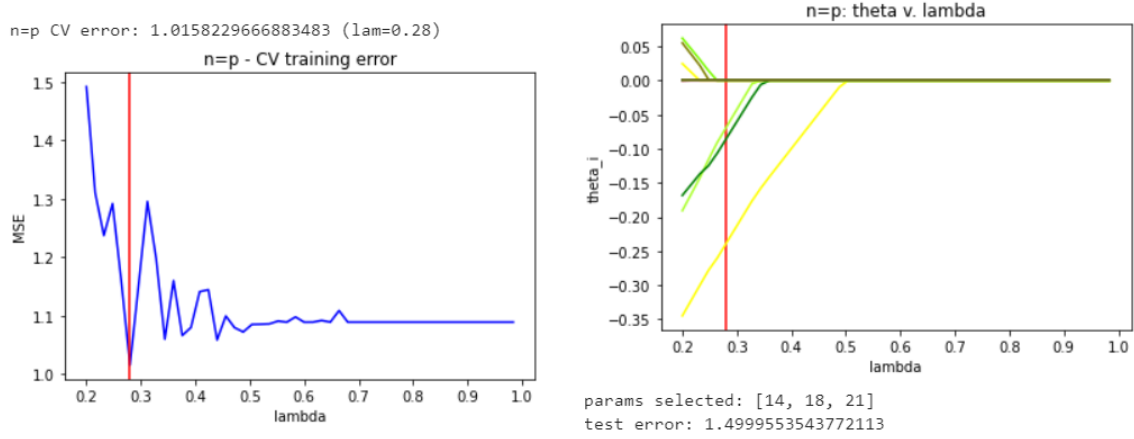


Figure 13: Ridge with Soft-Thresholding, $n = p$ correlated data set

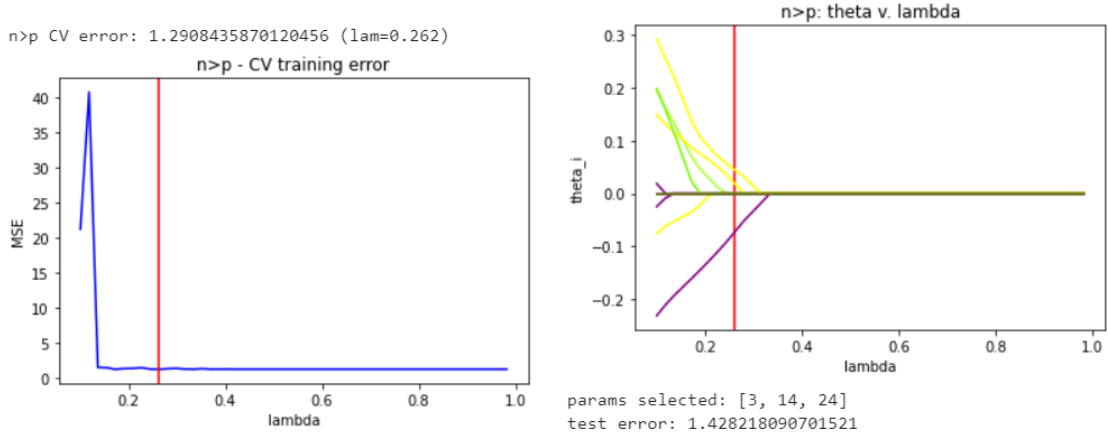


Figure 14: Ridge with Soft-Thresholding, $n > p$ correlated data set

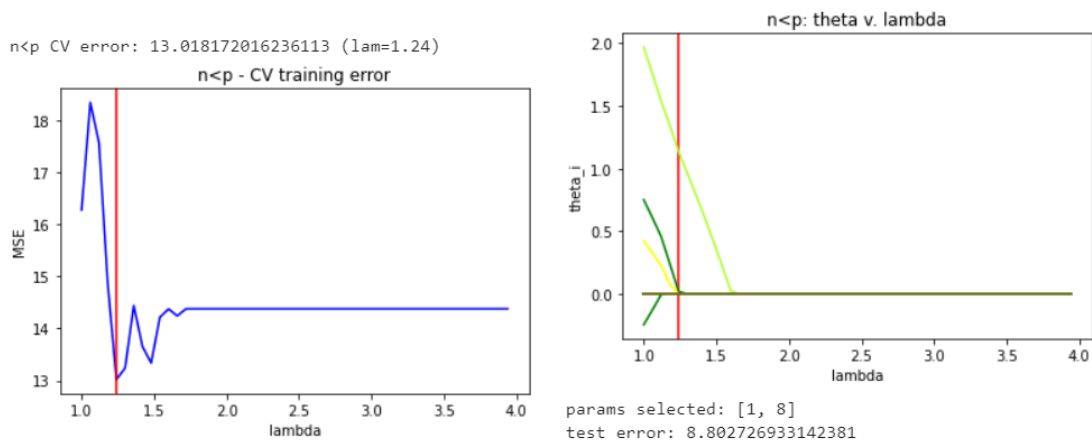


Figure 15: Ridge with Quadratic Soft-Thresholding, $n < p$ orthogonal data set

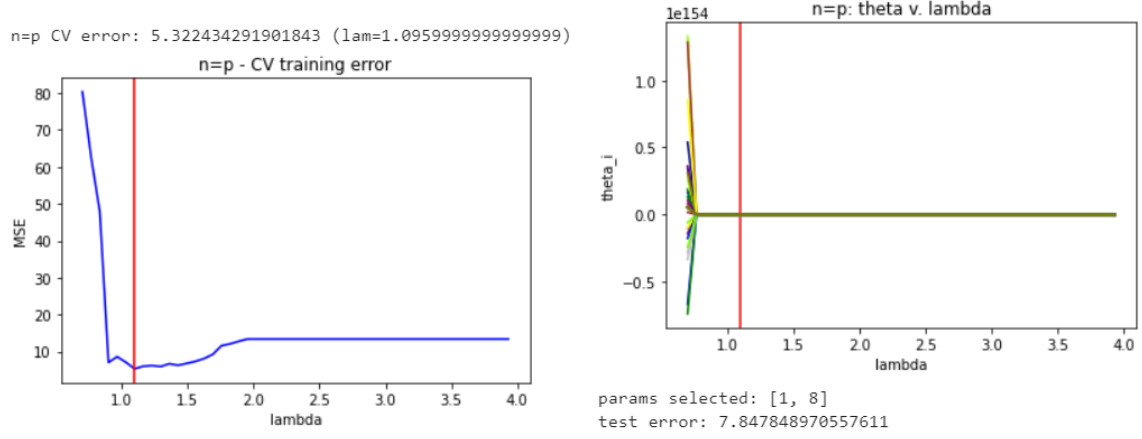


Figure 16: Ridge with Quadratic Soft-Thresholding, $n = p$ orthogonal data set

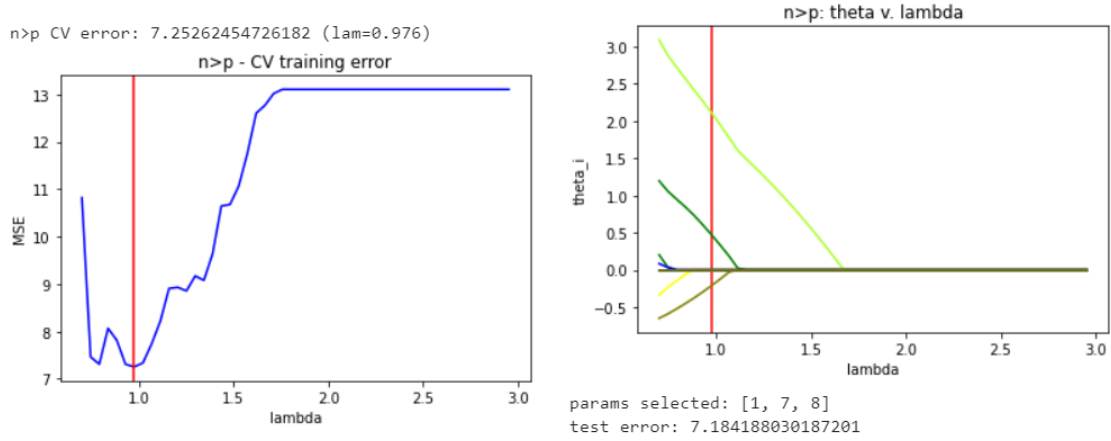


Figure 17: Ridge with Quadratic Soft-Thresholding, $n > p$ orthogonal data set

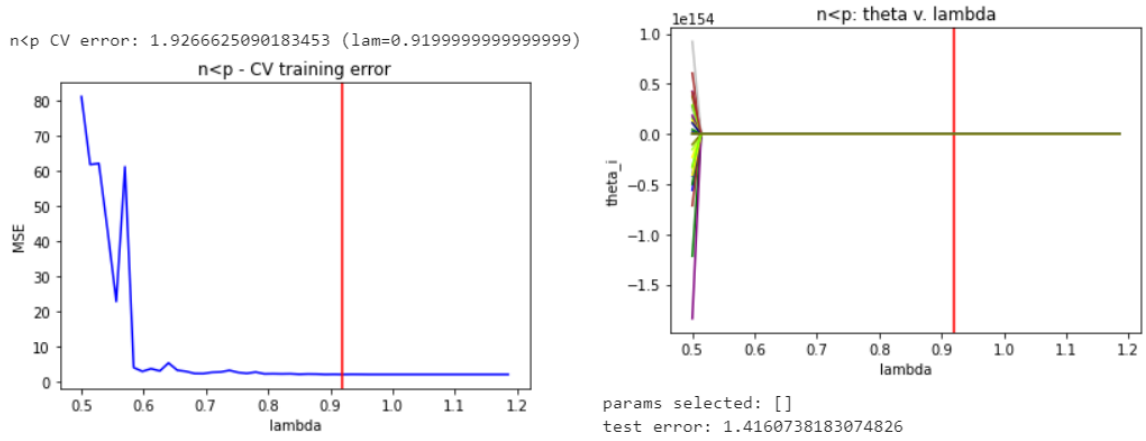


Figure 18: Ridge with Quadratic Soft-Thresholding, $n < p$ correlated data set

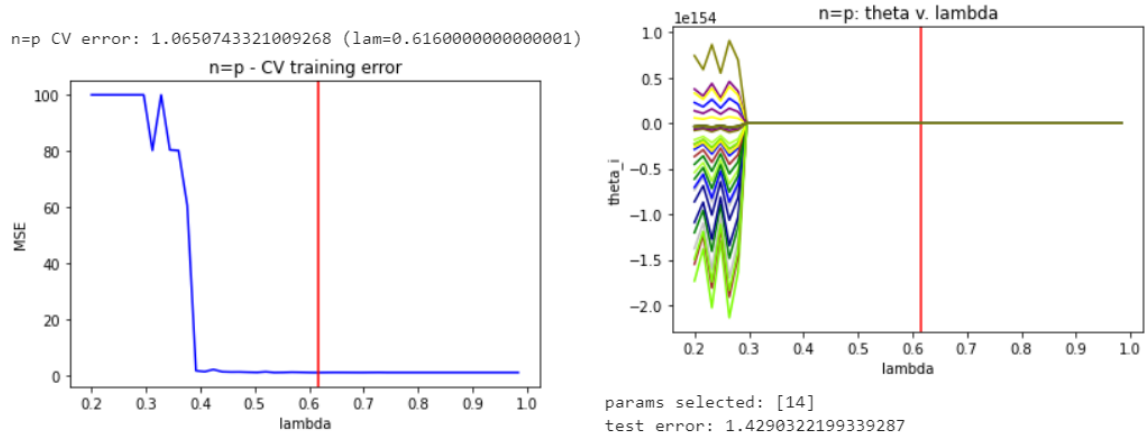


Figure 19: Ridge with Quadratic Soft-Thresholding, $n = p$ correlated data set

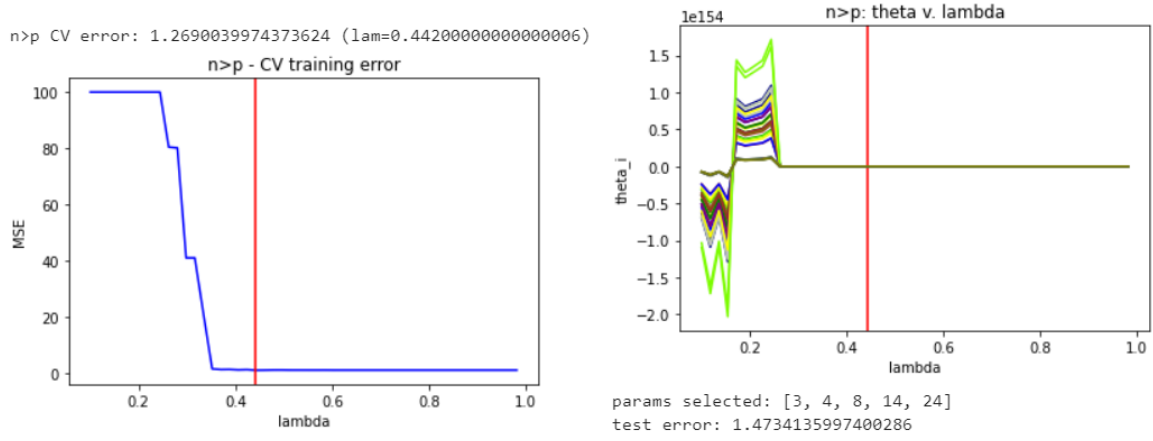


Figure 20: Ridge with Quadratic Soft-Thresholding, $n > p$ correlated data set

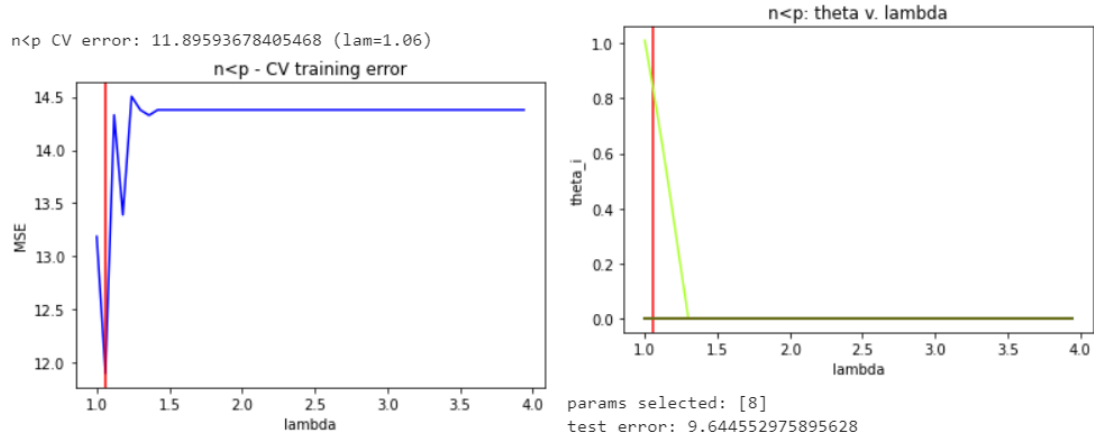


Figure 21: Ridge with Exponential Soft-Thresholding, $n < p$ orthogonal data set

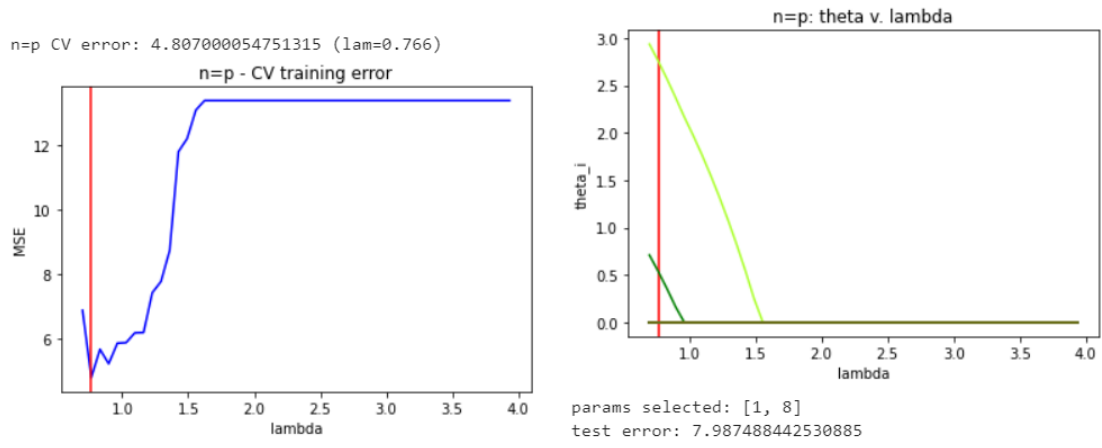


Figure 22: Ridge with Exponential Soft-Thresholding, $n = p$ orthogonal data set

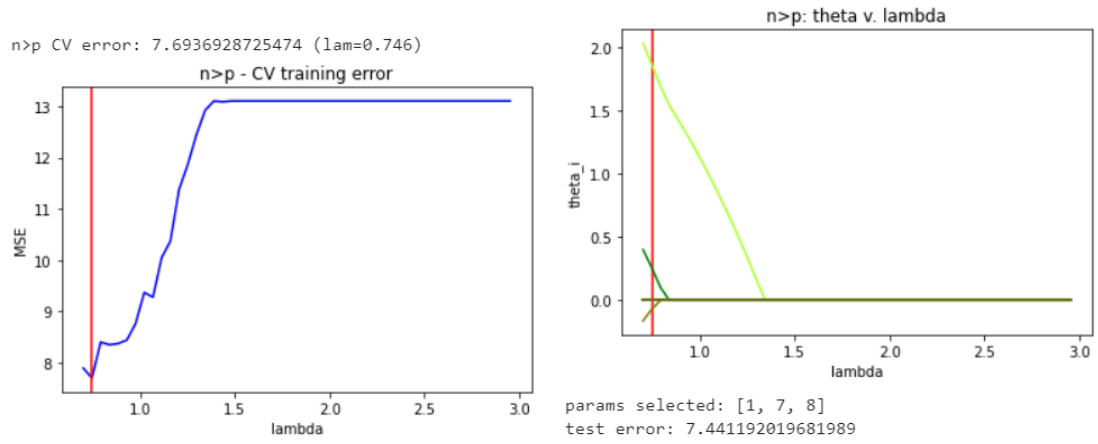


Figure 23: Ridge with Exponential Soft-Thresholding, $n > p$ orthogonal data set

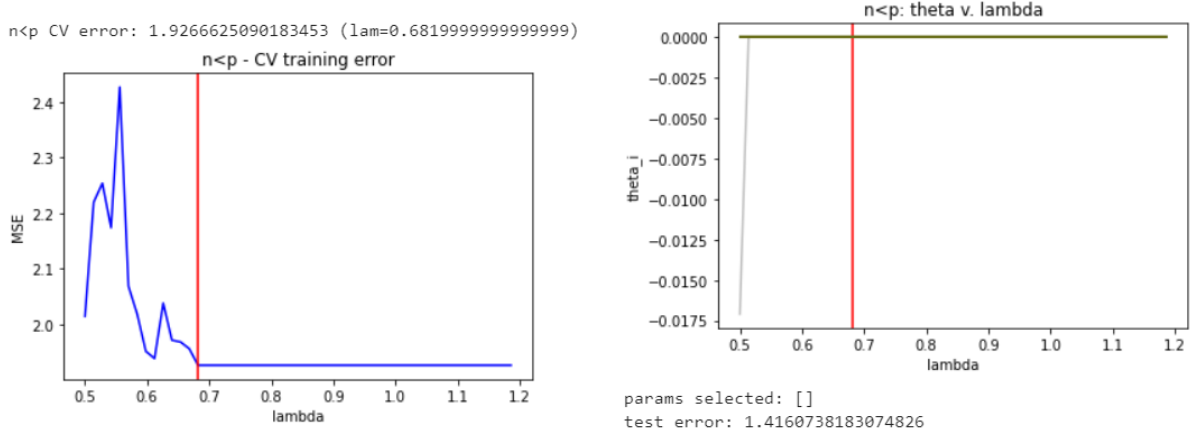


Figure 24: Ridge with Exponential Soft-Thresholding, $n < p$ correlated data set

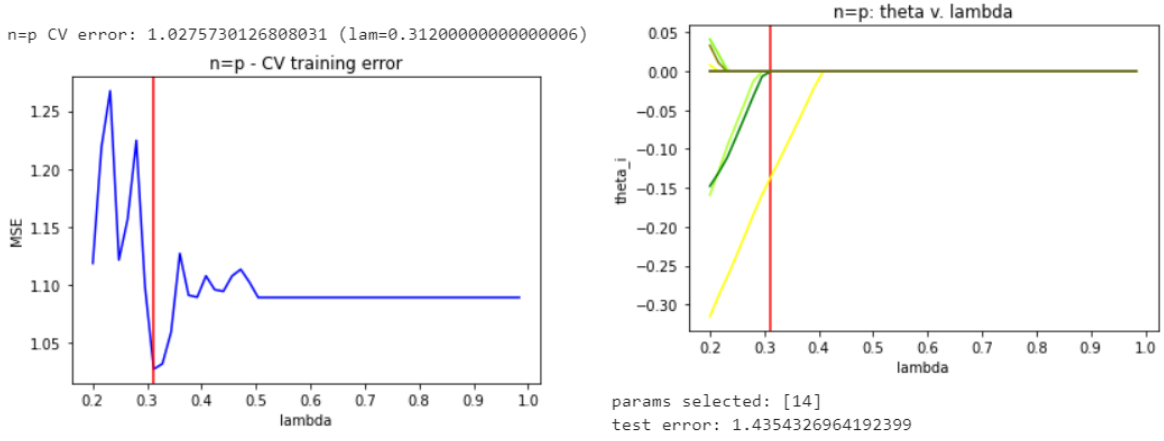


Figure 25: Ridge with Exponential Soft-Thresholding, $n = p$ correlated data set

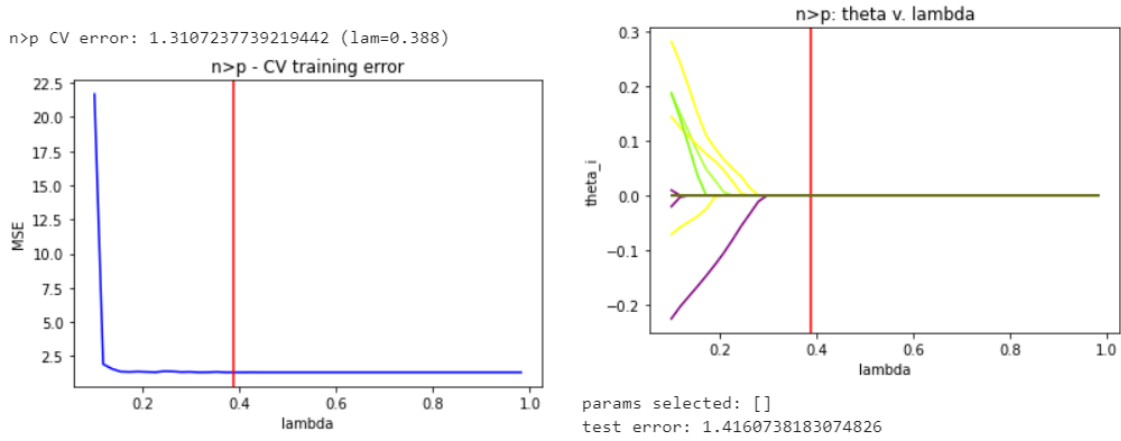


Figure 26: Ridge with Exponential Soft-Thresholding, $n > p$ correlated data set

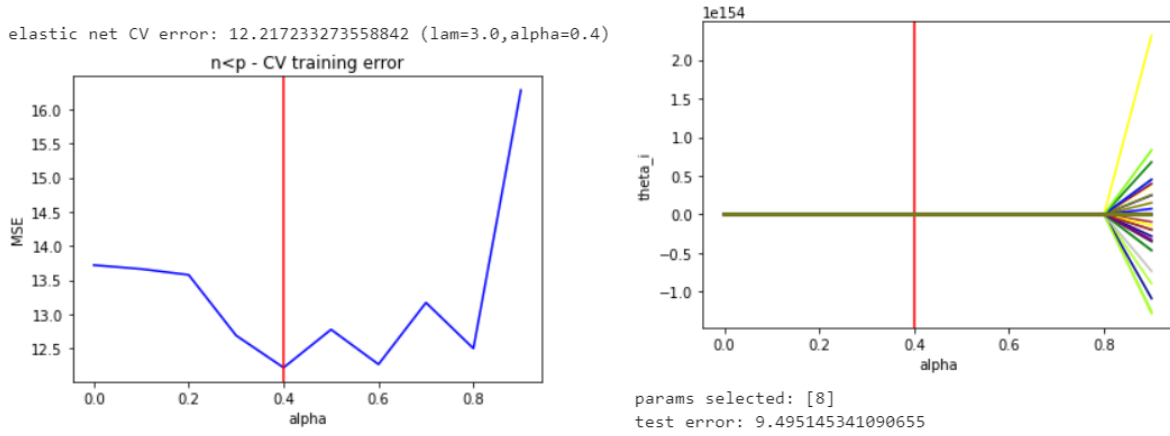


Figure 27: Elastic Net, $n < p$ orthogonal data set

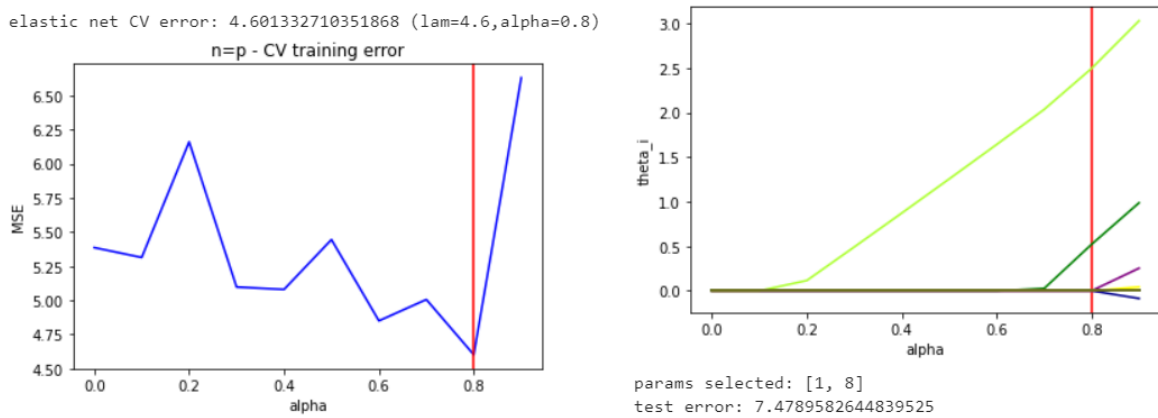


Figure 28: Elastic Net, $n = p$ orthogonal data set

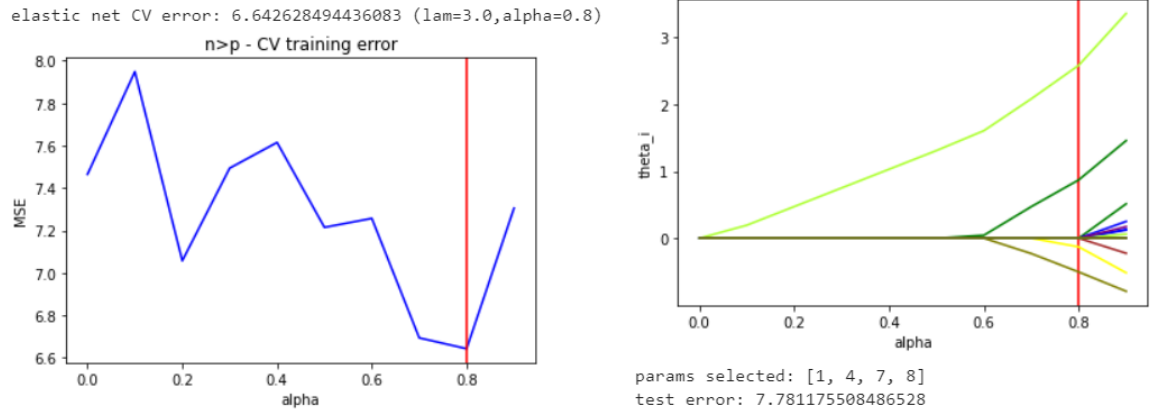


Figure 29: Elastic Net, $n > p$ orthogonal data set

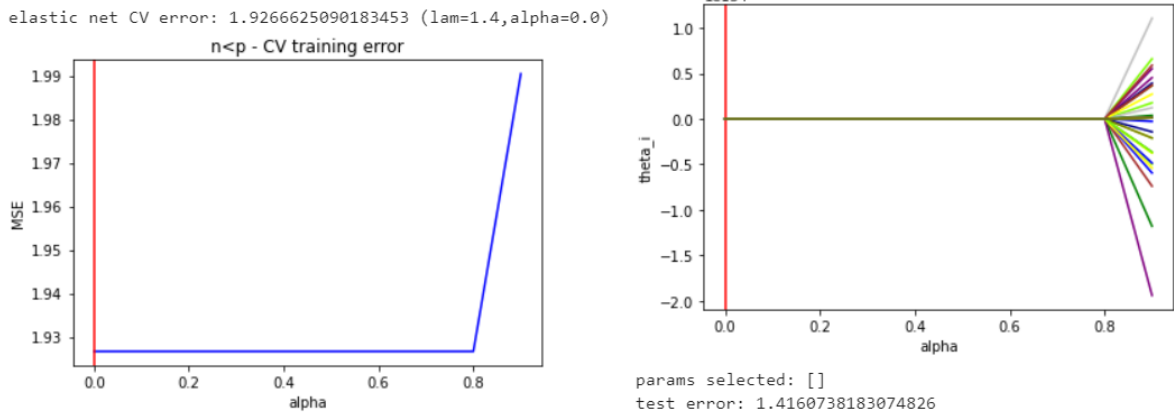


Figure 30: Elastic Net, $n < p$ correlated data set

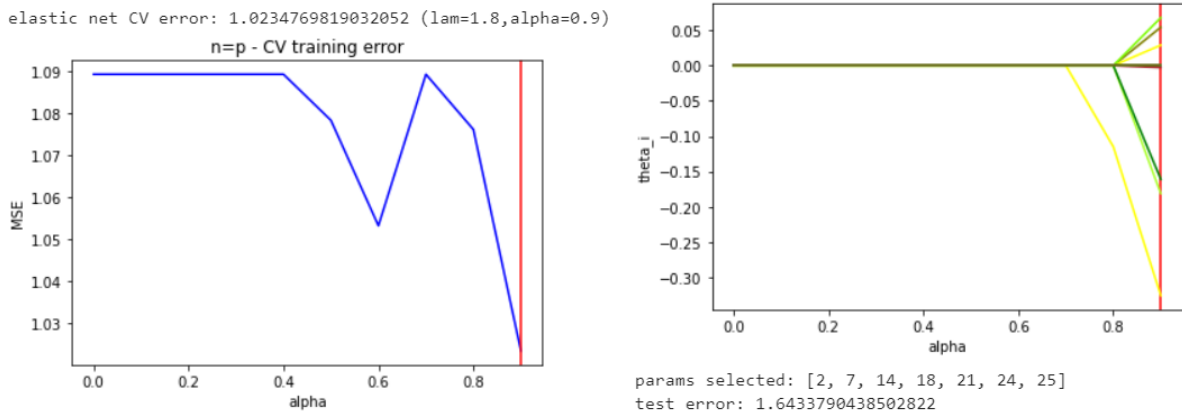


Figure 31: Elastic Net, $n = p$ correlated data set

elastic net CV error: 1.3027029115677597 (lam=1.0,alpha=0.7)

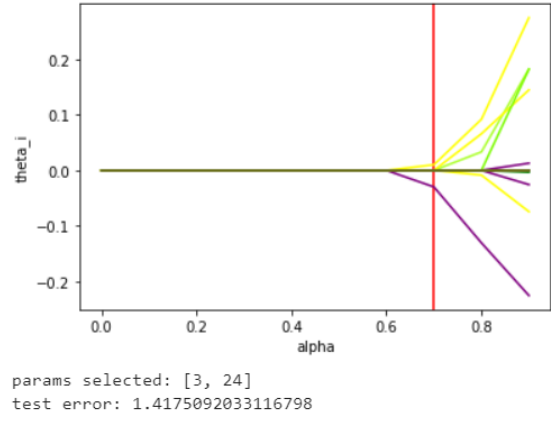
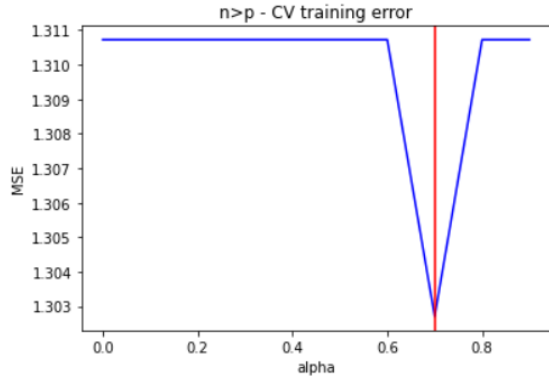


Figure 32: Elastic Net, $n > p$ correlated data set

n < p CV error: 12.22141402920315 (lam=1.16)

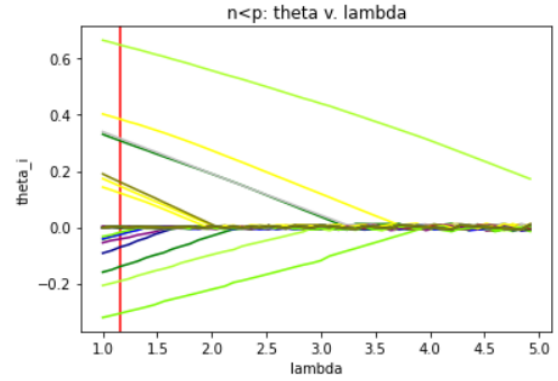


Figure 33: LASSO with Batch Gradient Descent, $n < p$ orthogonal data set

n = p CV error: 6.443097458137994 (lam=1.4)

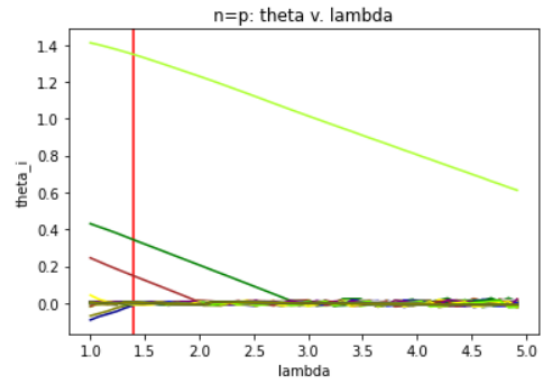
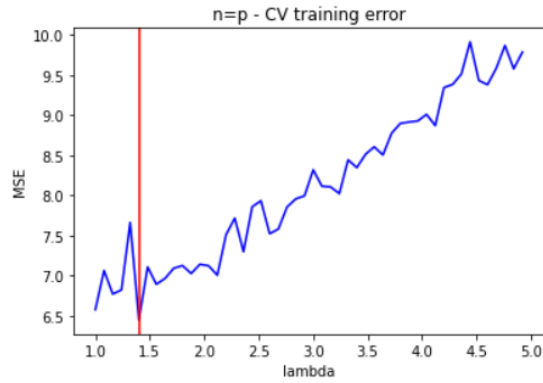


Figure 34: LASSO with Batch Gradient Descent, $n = p$ orthogonal data set

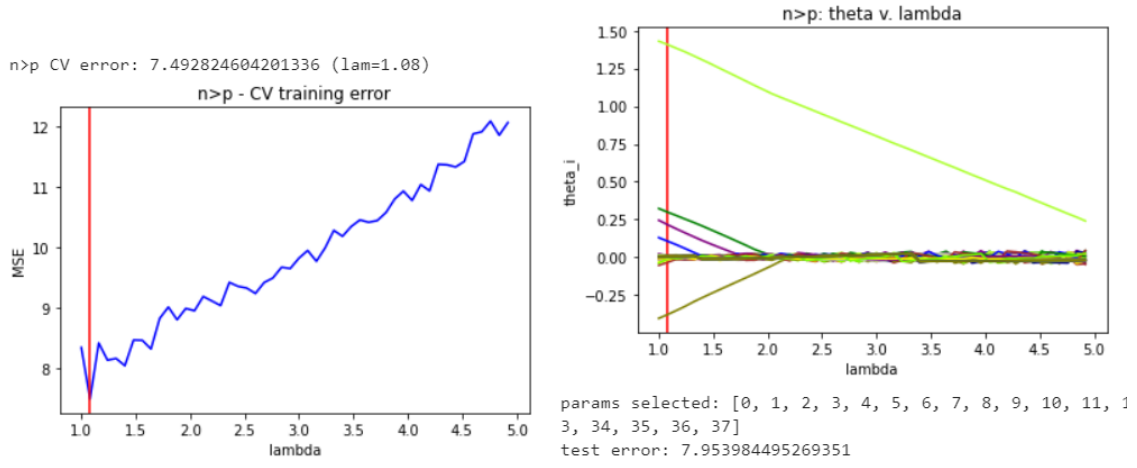


Figure 35: LASSO with Batch Gradient Descent, $n > p$ orthogonal data set

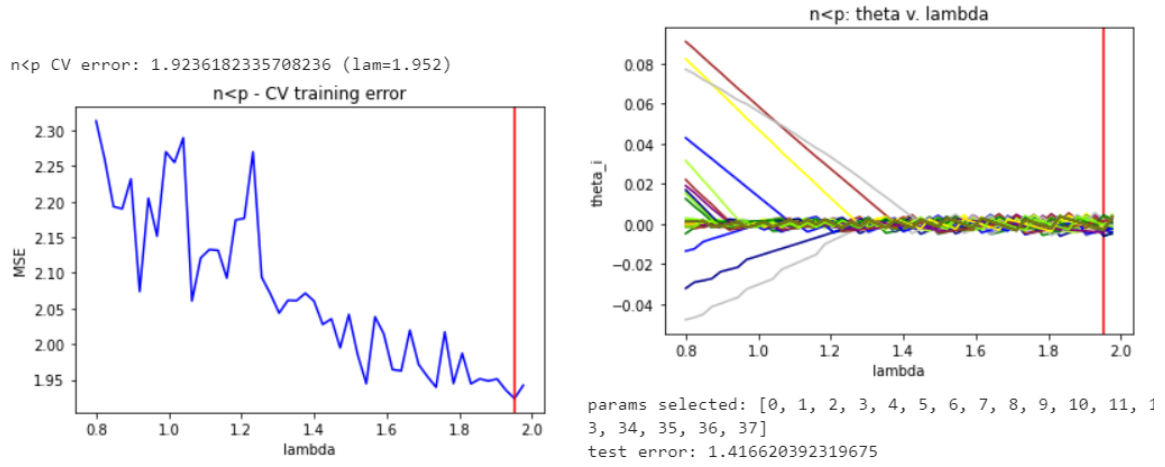


Figure 36: LASSO with Batch Gradient Descent, $n < p$ correlated data set

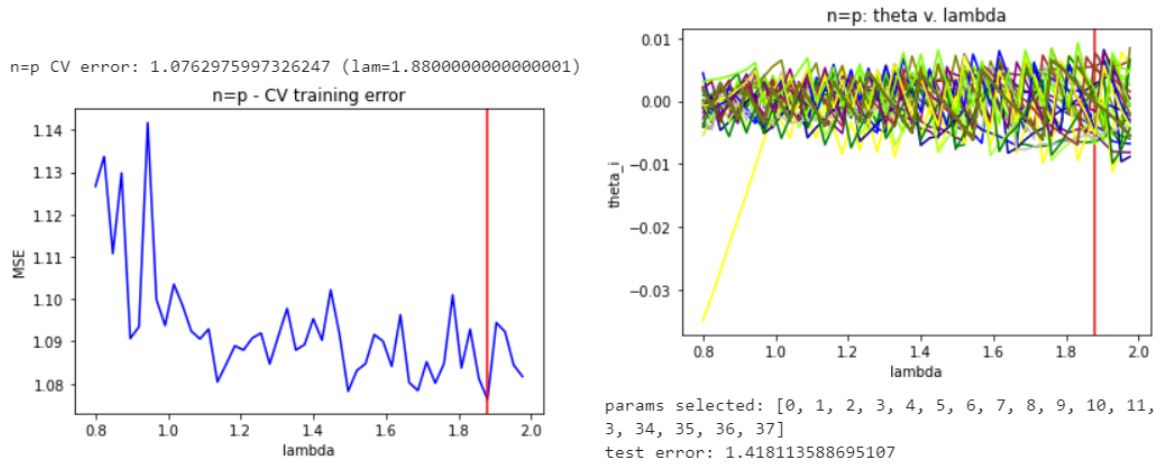


Figure 37: LASSO with Batch Gradient Descent, $n = p$ correlated data set

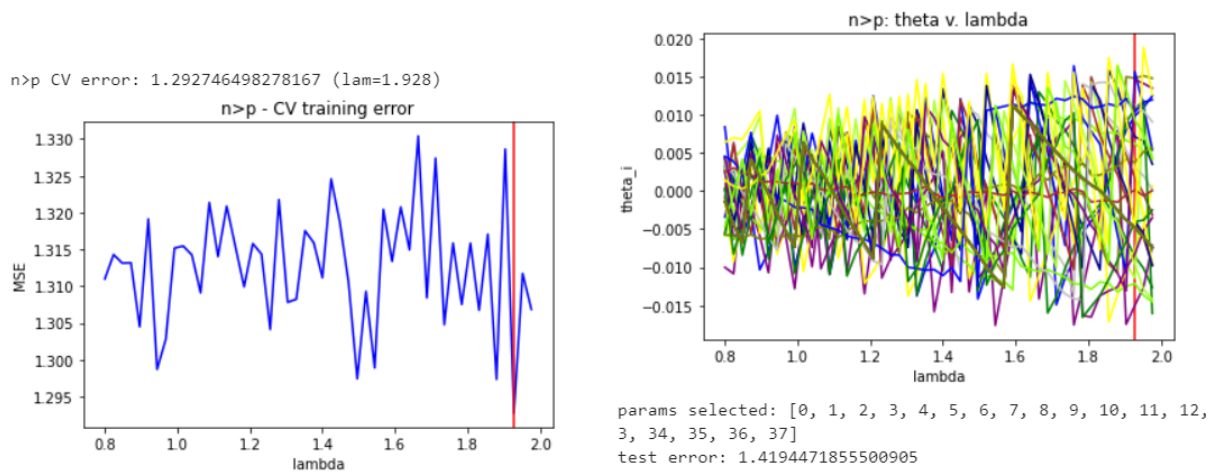


Figure 38: LASSO with Batch Gradient Descent, $n > p$ correlated data set