Lec 12: Regularized Learning

Ailin Zhang

Roadmap for Regularized Learning

- Ridge regression
- Lasso regression
- Coordinate descent
- Spline regression
- Least angle regression
- Stagewise regression for epsilon learning
- Bayesian regression
- Perceptron
- SVM
- Adaboost

Note: We will have midterm after regularized learning! (Est. Nov.7 - 11)

Revisit Overfitting: When Y is a noise vector

True value of β is 0, and $\mathbf{Y} = \epsilon$.

Assume

1. $\epsilon \sim \mathrm{N}(0, I_n)$ Thus $\mathrm{E}[\|\epsilon\|^2] = n$.
2. $\mathbf{X}^\top \mathbf{X} = I_p$

The least squares estimate $\hat{\beta} = \mathbf{X}^\top \epsilon = \delta \sim \mathrm{N}(0, I_p)$. Thus $\mathrm{E}[\|\delta\|^2] = p$.

δ is the projected coordinates of the noise vector ϵ onto the space spanned by $\mathbf{X}$.

The fitted value or the projected vector is $\hat{\epsilon} = \mathbf{X}\hat{\beta} = \mathbf{X}\delta$.

The training error is

$$\mathrm{E}[\|\epsilon - \hat{\epsilon}\|^2] = \mathrm{E}[\|\epsilon\|^2 - \|\hat{\epsilon}\|^2],$$

Revist Overfitting: When Y is a noise vector

According to Pythagarean theorem: $\mathrm{E}[\|\epsilon\|^2] = n$, and $\mathrm{E}[\|\hat{\epsilon}\|^2] = \mathrm{E}[\|\delta\|^2] = p$.

Thus the training error is $n - p$.

Now consider the testing error. $\tilde{\epsilon} \sim \mathrm{N}(0, I_n)$, and $\mathrm{Cov}(\epsilon, \tilde{\epsilon}) = 0$. Then the testing error is

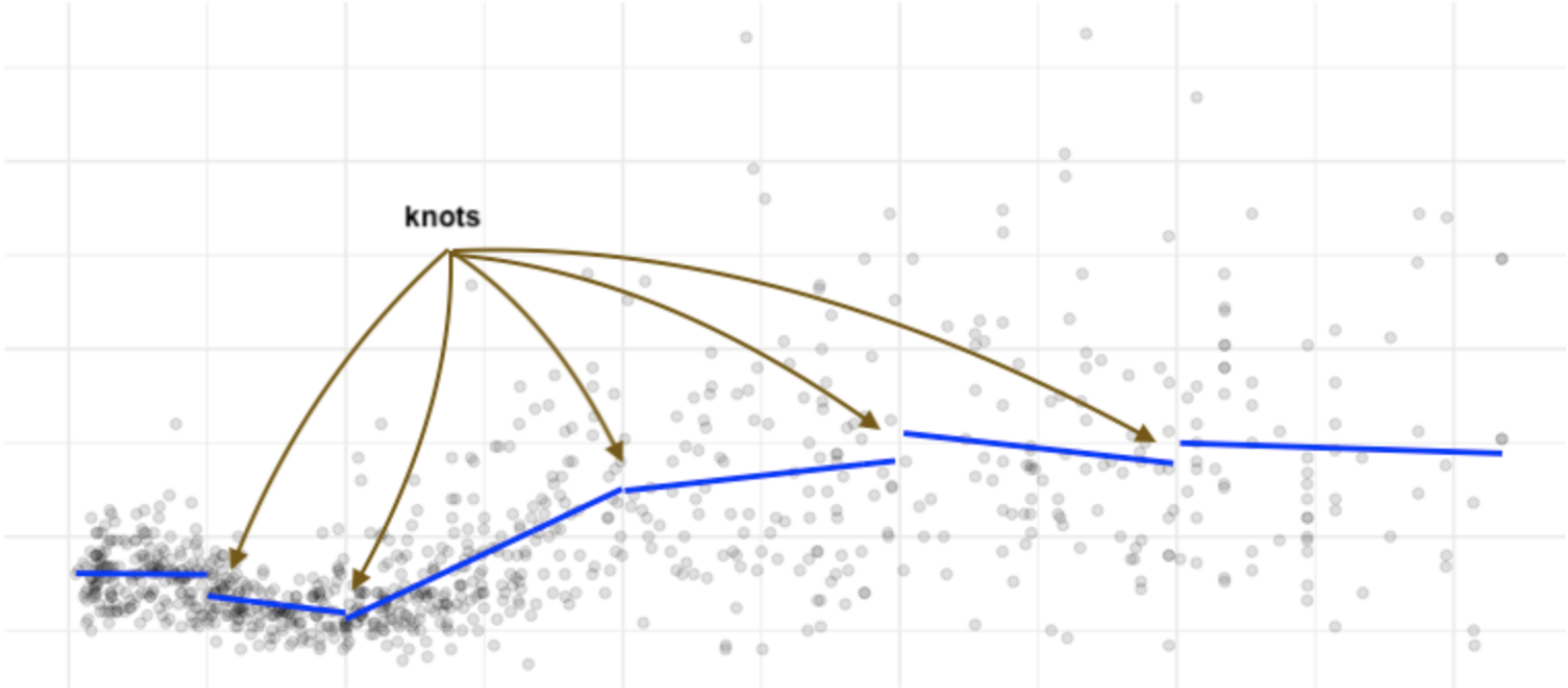
$$\mathrm{E}[\|\tilde{\epsilon} - \hat{\epsilon}\|^2] = \mathrm{E}[\|\tilde{\epsilon}\|^2 + \|\hat{\epsilon}\|^2 - 2\langle\tilde{\epsilon}, \hat{\epsilon}\rangle].$$

Since $\mathrm{E}[\langle\tilde{\epsilon}, \epsilon\rangle] = 0$, the testing error is $n + p$.

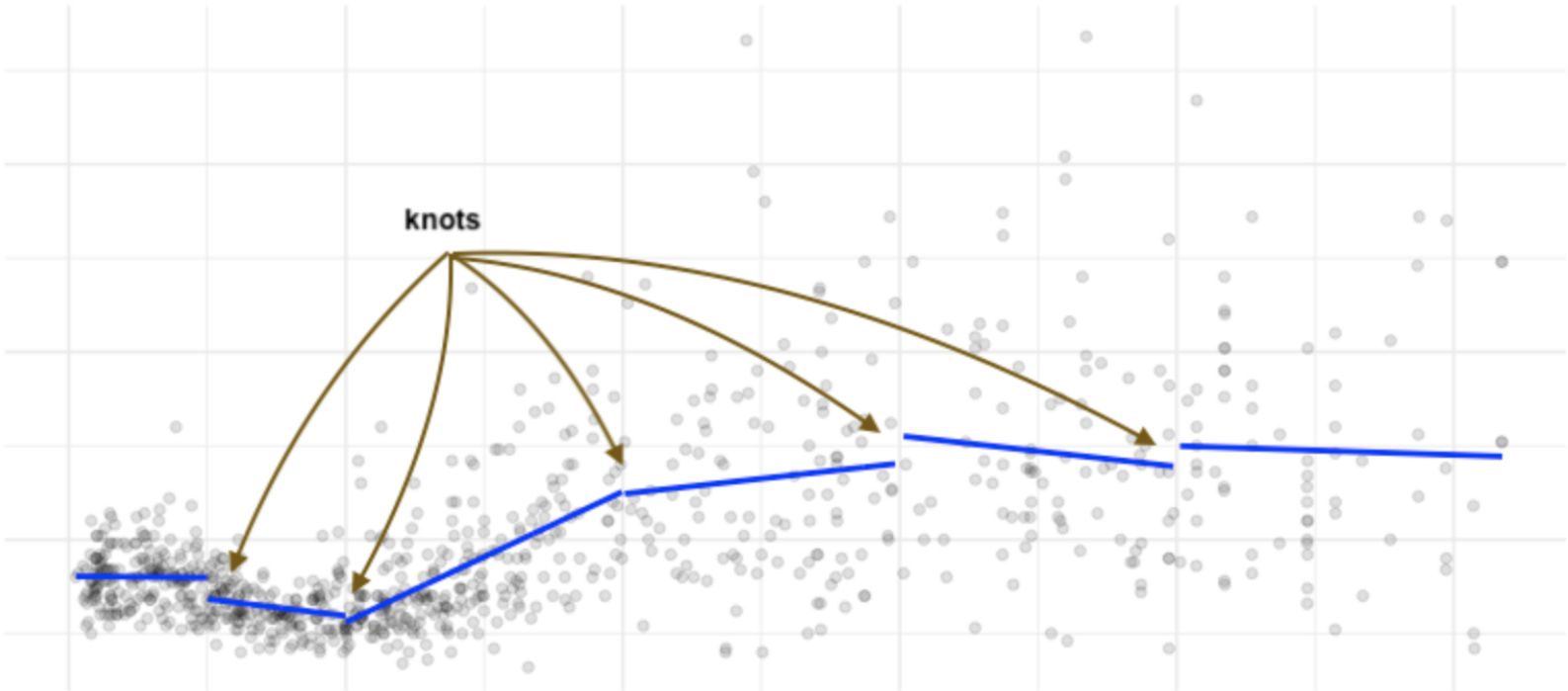
The overfitting is testing error - training error, which is $2p$.

The point is that you may be able to explain noises by overfitting the data in training, you will not be able to do so in testing.

Spline Regression - Graphical Understanding



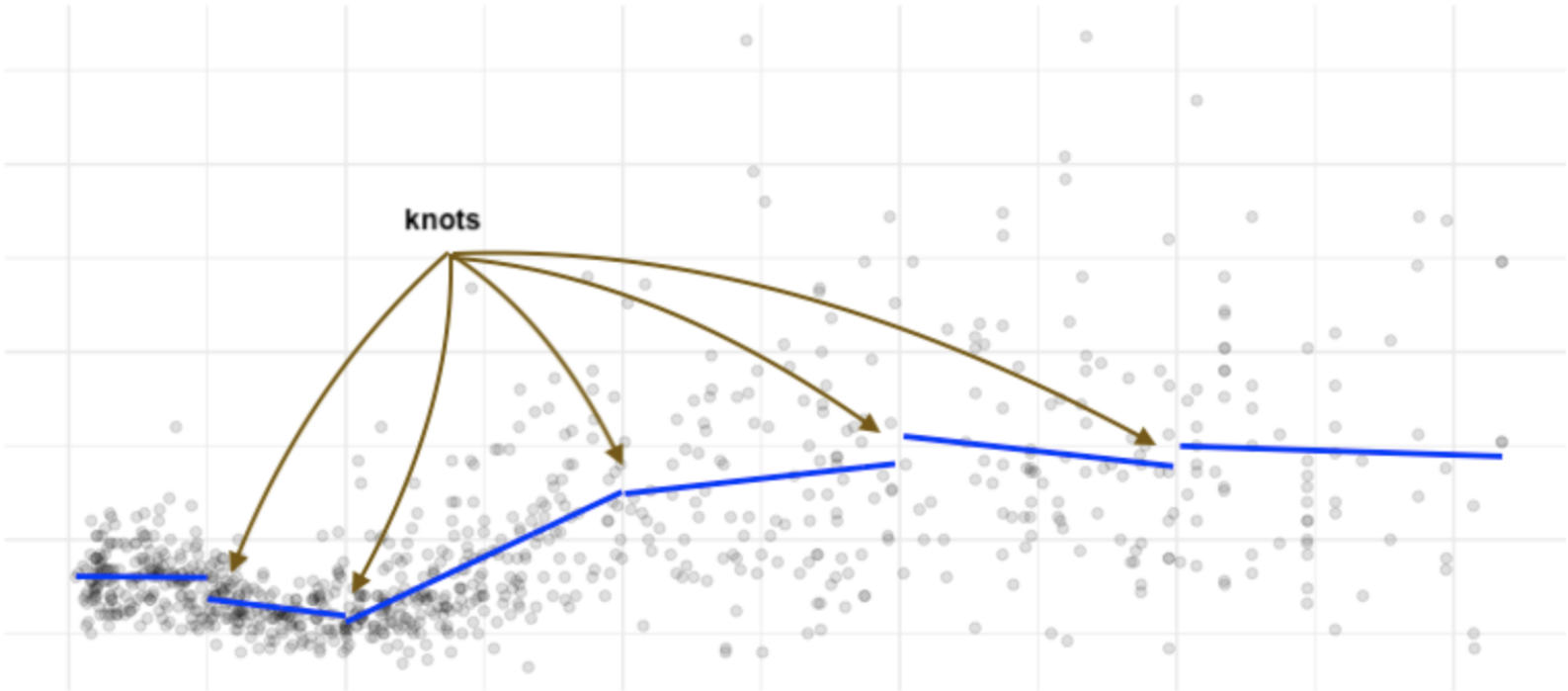
Spline Regression - Graphical Understanding



Piecewise linear

Question: How to avoid the jumps?

Spline Regression - Graphical Understanding



Piecewise linear

Question: How to avoid the jumps?

$(x-k)_+ = \max(0, x-k)$

Spline Regression

Consider the piecewise linear spline model, where the training examples are (x_i, y_i), for $i = 1, ..., n$, and x_i is one-dimensional.

The model assumes

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j \max(0, x_i - k_j) + \epsilon_i,$$

where k_j is the j-th knot of the spline, and β_j is the change of slope at knot k_j. We can estimate β by minimizing

$$\sum_{i=1}^{n} \left[y_i - \left(\beta_0 + \sum_{j=1}^{p} \beta_j \max\left(0, x_i - k_j\right) \right) \right]^2 + \lambda \sum_{j=1}^{p} \beta_j^2,$$

Basis Function

Spline regression models are in fact special cases of a basis function approach.

The idea is to have at hand a family of basis functions or transformations that can be applied to a variable X: function $b_1(X), b_2(X), ..., b_K(X)$. Instead of fitting a linear model in X, we fit the model

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \beta_3 b_3(x_i) + ... + \beta_K b_K(x_i) + \epsilon_i$$

All of the tools for linear models are available in this setting.

How to choose knots?

The regression spline is most flexible in regions that contain a lot of knots. Hence, one option is to place more knots in places where we feel the function might vary most rapidly, and to place fewer knots where it seems more stable.

How to choose knots?

The regression spline is most flexible in regions that contain a lot of knots. Hence, one option is to place more knots in places where we feel the function might vary most rapidly, and to place fewer knots where it seems more stable.

Cross-validation

we remove a portion of the data (say 10 %), fit a spline with a certain number of knots to the remaining data, and then use the spline to make predictions for the held-out portion. We repeat this process multiple times until each observation has been left out once, and then compute the overall cross-validated RSS. This procedure can be repeated for different numbers of knots K. Then the value of K giving the smallest RSS is chosen.

R code

```
n = 20
p = 500
sigma = .1
lambda = 1.
x = runif(n)
x = sort(x)
Y = x^2 + rnorm(n)*sigma
X = matrix(x, nrow=n)
for (k in (1:(p-1))/p)
  X = cbind(X, (x>k)*(x-k))
beta = myRidge(X, Y, lambda)
Yhat = cbind(rep(1, n), X)%*%beta
plot(x, Y, ylim = c(-.2, 1.2), col = "red")
par(new = TRUE)
plot(x, Yhat, ylim = c(-.2, 1.2), type = 'l', col = "green")
```

Spline Generalization

The above spline model can be extended in the following two directions:

(1) Cubic spline. We replace the linear piece to cubic piece to make it continuously differentiable at the knots.

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j \max(0, x_i - k_j)^3 + \epsilon_i,$$

Spline Generalization

The above spline model can be extended in the following two directions:

(1) Cubic spline. We replace the linear piece to cubic piece to make it continuously differentiable at the knots.

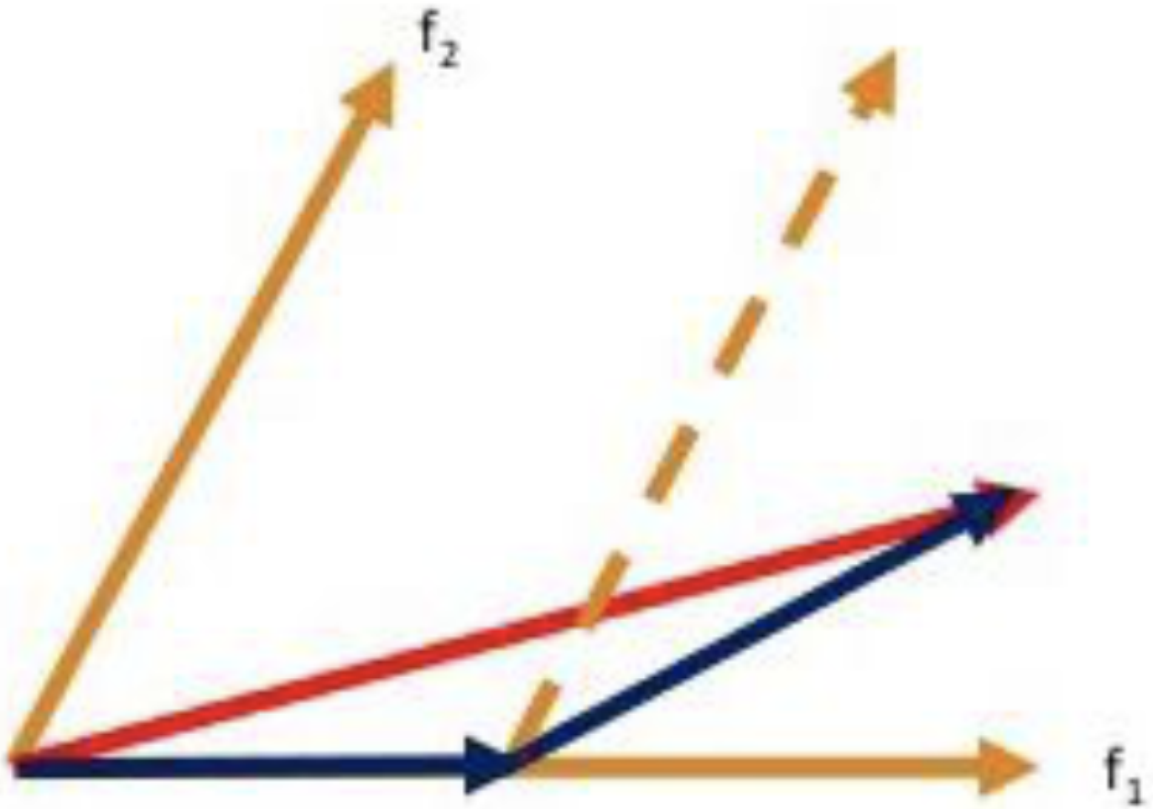
$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j \max(0, x_i - k_j)^3 + \epsilon_i,$$

(2) Rectified neural network. The piecewise linear form of the linear spline also underlies modern multi-layer neural networks, where $\max(0, r)$ is called rectified linear unit (ReLU).

Least Angle Regression (LAR)

- LAR is a type of forward step-wise regression.
  - Forward step-wise regression builds a model sequentially, adding one variable at a time. At each step, it identifies the best variable to include in the active set, and then updates the least squares fit to include all the active variables.
- LAR is connected to Lasso regression.
- LAR was defined in the Efron et al., 2004. It is a relatively newer algorithm and is viewed as a democratic version of the forward step-wise regression.

Least Angle Regress (LAR)

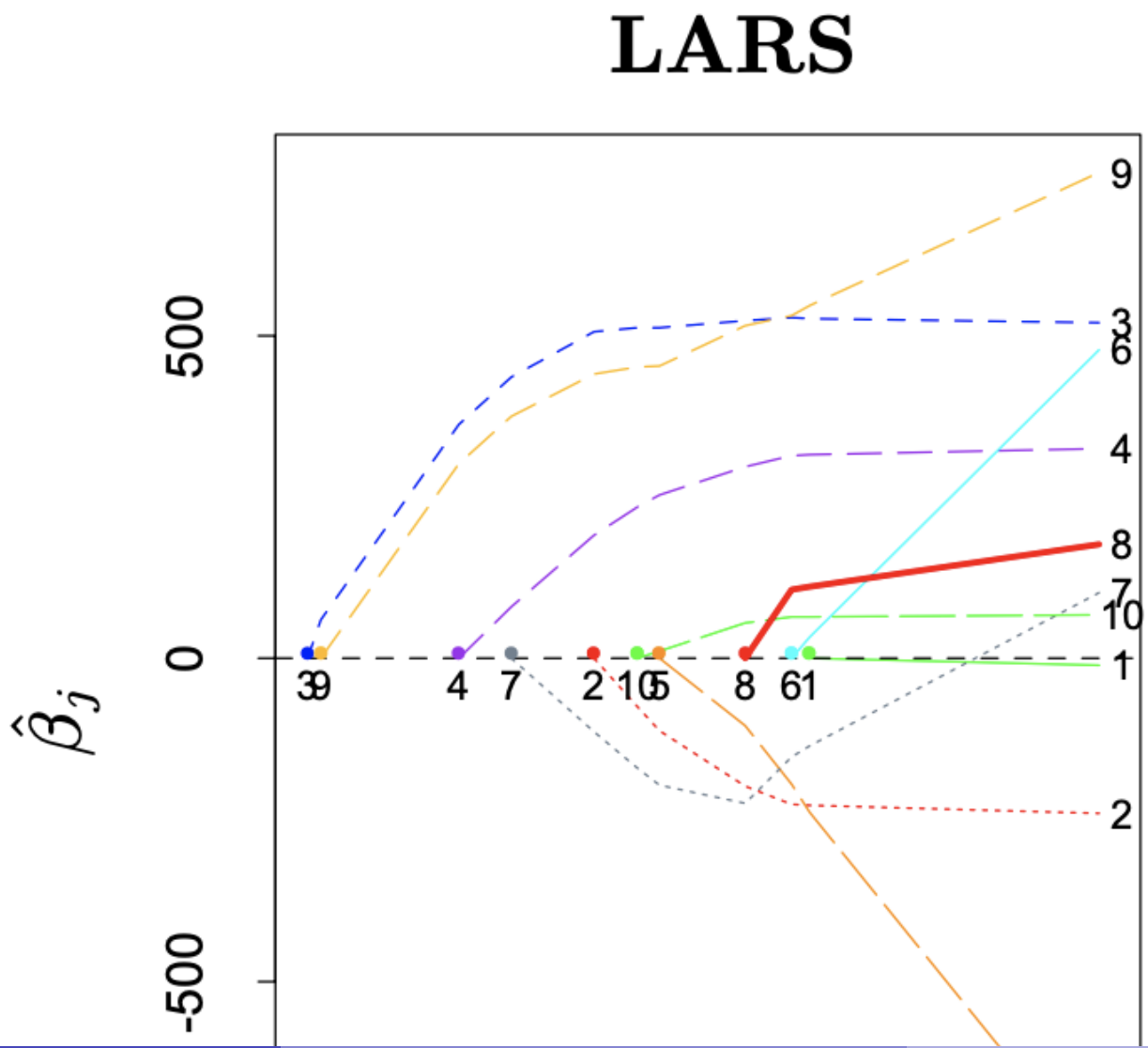


Least Angle Regression (LAR)

The basic steps of the Least-angle regression algorithm are:

1. Start with all coefficients β equal to zero.
2. Find the predictor x_j most correlated with y
3. Increase the coefficient β_j in the direction of the sign of its correlation with y. Take residuals $R = y - \hat{y}$ along the way. Stop when some other predictor x_k has as much correlation with r as x_j has.
4. Increase (β_j, β_k) in their joint least squares direction, until some other predictor x_m has as much correlation with the residual$r$.
5. Increase $(\beta_j, \beta_k, \beta_m)$ in their joint least squares direction, until some other predictor x_n has as much correlation with the residual r.
6. Continue until: all predictors are in the model

LAR Solution Path



LAR vs Lasso

If β is the Lasso solution, at any given λ, let $\mathbf{R} = \mathbf{Y} - \sum_{j=1}^{p} \mathbf{X}_j \beta_j$, then $\hat{\beta}_j = \beta_j + \langle \mathbf{R}, \mathbf{X}_j \rangle / \|\mathbf{X}_j\|_{\ell_2}^2$. then

$$\langle \mathbf{R}, \mathbf{X}_j \rangle = \begin{cases} \lambda, & \text{if } \beta_j > 0, \\ -\lambda, & \text{if } \beta_j < 0, \\ s\lambda & \text{if } \beta_j = 0. \end{cases}$$

where $|s| < 1$.

Thus in the above process, for all of those selected $\mathbf{X}_j$, the algorithm maintains that $\langle \mathbf{R}, \mathbf{X}_j \rangle$ to be λ or $-\lambda$, for all selected $\mathbf{X}_j$. If we interpret $|\langle \mathbf{R}, \mathbf{X}_j \rangle|$ in terms of the angle between $\mathbf{R}$ and $\mathbf{X}_j$, then we may call the above process as least angle regression (LAR). In fact, the solution path is piecewise linear, and the LARS computes the linear pieces analytically instead of gradually reducing λ as in coordinate descent.

Lasso can be thought of as restricted versions of LAR