

# Module 6, part II: Sparse regression and shrinkage

BIOS 526

## Reading

- Sections 3.3 - 3.4 in Elements of Statistical Learning II (ESL II) (Hastie et al)
- Section 6.5 in Introduction to Statistical Learning (ISL) (James et al)

## Concepts

- Ridge regression and L2 norm
- Sparsity
- Lasso and L1 norm
- Selecting the penalty / CV
- Correlated Variables and Elastic Net
- Bonferroni Correction

# Motivation

In modern applications,  $p$  is often very large relative to  $n$ .

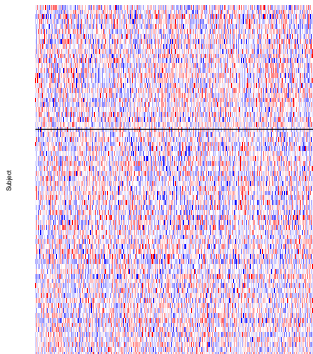
A study is “high dimensional” if  $p$  is large.

There are a number of considerations:

1. Predictive Ability: OLS has low bias but high variance. Can we **sacrifice bias** to achieve **lower variance**, and thus **greater accuracy**?
2. Interpretability: with large  $p$ , our goal is to identify important variables. It would be helpful to have a model that does this, rather than rely upon forward/backward selection procedures and p-values.
3. Moreover, when  $p > n$ , we can't use multiple regression – OLS estimators are not unique.

In  $(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$ ,  $p > n$  then  $(\mathbf{X}'\mathbf{X})$  not invertible

# Motivating dataset: gene expression in leukemia



- Response: binary variable indicating whether the subject has leukemia. 25 subjects with leukemia, 47 controls.  $n = 72$ .
- Covariates: Microarray data from  $p = 3571$  genes. Blue indicates negative values (lower expression), red indicates higher gene expression.

## Review: Bias-Variance Trade-off

$$\begin{aligned} E\{g(\mathbf{x}_i) - \hat{g}(\mathbf{x}_i)\}^2 &= E[\{g(\mathbf{x}_i) - E\hat{g}(\mathbf{x}_i) + E\hat{g}(\mathbf{x}_i) - \hat{g}(\mathbf{x}_i)\}^2] \\ &= [g(\mathbf{x}_i) - E\hat{g}(\mathbf{x}_i)]^2 + E[\hat{g}(\mathbf{x}_i) - E\hat{g}(\mathbf{x}_i)]^2 \\ &= (\text{Bias of } \hat{g}(\mathbf{x}_i))^2 + \text{Variance of } \hat{g}(\mathbf{x}_i) \end{aligned}$$

where  $\hat{g}(\mathbf{x}_i)$  is the estimate of  $E(Y|\mathbf{x}_i) = g(\mathbf{x}_i)$ .

Here,  $g(\mathbf{x}_i) = \mathbf{x}_i' \boldsymbol{\beta}$ .

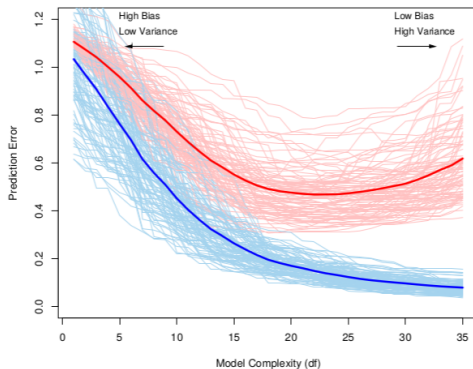
When estimating a large number of parameters, we may overfit the data.

Sacrifice some bias to reduce variance, improve overall accuracy.

To estimate more parameters accurately, you need more data. A rough rule of thumb is 10 observations / parameter.

# Model Complexity

220 7. Model Assessment and Selection



**FIGURE 7.1.** Behavior of test sample and training sample error as the model complexity is varied. The light blue curves show the training error  $\overline{\text{err}}$ , while the light red curves show the conditional test error  $\text{Err}_T$  for 100 training sets of size 50 each, as the model complexity is increased. The solid curves show the expected test error  $\text{Err}$  and the expected training error  $E[\overline{\text{err}}]$ .

# Review

We discussed penalized regression in Module 5.

Recall the ridge regression penalty from M5.2:

$$\operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^n (y_i - \mathbf{x}_i' \boldsymbol{\beta})^2 + \lambda \|\boldsymbol{\beta}\|_2^2$$

Here,  $\lambda$  is the **penalty**.

This problem can be equivalently formulated as

$$\operatorname{argmin}_{\boldsymbol{\beta}} (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) \quad \text{with constraint } \boldsymbol{\beta}'\boldsymbol{\beta} \leq C$$

where there exists a one-to-one mapping between  $\lambda$  and  $C$ .

## Closed-form solution

We derived the solution, which has a very convenient closed form:

$$\hat{\beta} = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{Y}$$

for some positive number  $\lambda$ .

Note:

- When  $\lambda = 0$ ,  $\hat{\beta}$  becomes the ordinary least squares estimate. So no penalization is present ( $C = \infty$ ).
- When  $\lambda \rightarrow \infty$ ,  $(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}$  becomes small, so  $\hat{\beta} \rightarrow \mathbf{0}$ .
- $(\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})$  is positive definite for  $\lambda > 0$ , so we can invert it even if  $p > n$ !



## Important details

The penalty  $\lambda$  is unfair if the predictors are not on the same scale.

Consequently, we **scale** the columns of  $\mathbf{X}$  to have **unit variance**. Then the penalty for each coefficient is equitable.

We often do not want to penalize the intercept:

$$\operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^n (y_i - \beta_0 - \mathbf{x}'_i \boldsymbol{\beta})^2 + \lambda \|\boldsymbol{\beta}\|_2^2$$

where  $\beta_0$  is not penalized.

Similarly, we can center all columns of  $\mathbf{X} \in \mathbb{R}^{n \times p}$  and center  $\mathbf{y}$ . Then  $\hat{\beta}_0 = 0$ . So one approach is to **center** your data and then penalize all terms.

## Ridge regression versus sparsity

For variable selection, it would be nice to have estimates of  $\beta$  that include **exact zeros**.

If  $\hat{\beta}_j \neq 0$ ,  $\mathbf{x}_j$  is selected.

If  $\hat{\beta}_j = 0$ ,  $\mathbf{x}_j$  is not important.

Although  $\hat{\beta}^{Ridge}$  may have reduced MSE relative to  $\hat{\beta}^{OLS}$ , it does not help us too much with variable selection.

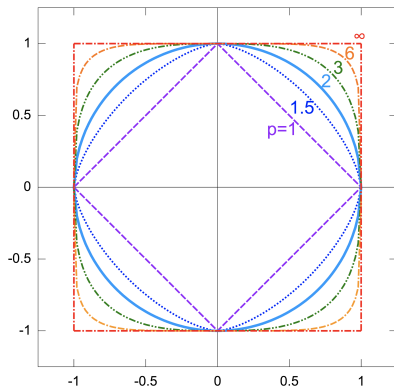
# Norms

We can measure the length of a vector using different metrics.

Ridge regression uses the squared  $L_2$  norm, which is also called the Euclidean norm.

For a vector  $\beta \in \mathbb{R}^d$ , an  $L_p$  norm has the general form

$$\|\beta\|_p = \left( \sum_{j=1}^d |\beta_j|^p \right)^{1/p}$$



**Figure:** Illustration of the unit circles of some  $p$ -norms for  $p=1$ : sum norm,  $p=1.5$ ,  $p=2$ : Euclidean norm,  $p=3$ ,  $p=6$ ,  $p=\infty$ : max norm in two dimensions from [https://en.wikipedia.org/wiki/Lp\\_space#/media/File:Vector-p-Norms-qt11.svg](https://en.wikipedia.org/wiki/Lp_space#/media/File:Vector-p-Norms-qt11.svg).

# Lasso

We will assume all covariates and  $y$  are centered so that we don't need to estimate the intercept.

We will also assume they are scaled.

Let's consider using a different norm, the  $L_1$  norm.

$$\hat{\beta}^{Lasso} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n (y_i - \mathbf{x}_i' \beta)^2 + \lambda \|\beta\|_1$$

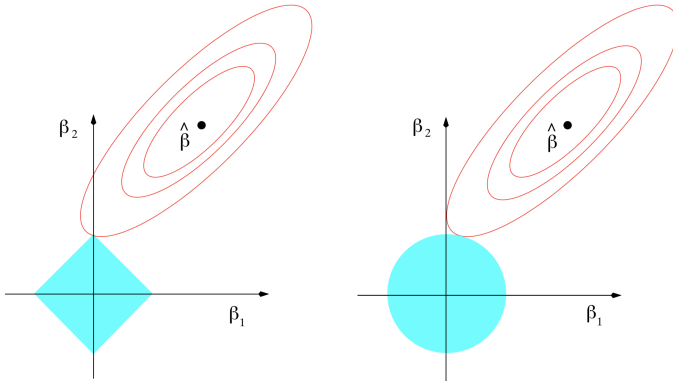
where  $\|\beta\|_1 = \sum_{j=1}^p |\beta_j|$ .

Statisticians call this the **LASSO**: Least absolute shrinkage and selection operator.

This penalty is chosen because it induces [sparsity](#).

[Sparsity](#) is when we have many zeros and a relatively small number of non-zeros.

## $L_1$ and $L_2$ constraints



**FIGURE 3.11.** Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions  $|\beta_1| + |\beta_2| \leq t$  and  $\beta_1^2 + \beta_2^2 \leq t^2$ , respectively, while the red ellipses are the contours of the least squares error function.

Figure: From ESLII p. 71

# Lasso

Why does the Lasso induce sparsity?

For  $\|\beta\|_1 \leq C$ , the  $L_1$  constraint creates sharp points at the boundary corresponding to  $(\beta_1, \beta_2) = (C, 0)$ ,  $(0, C)$ ,  $(-C, 0)$ , and  $(0, -C)$ .

For  $\beta = (\beta_1, \beta_2)$ , consider where the contours of  $(\mathbf{Y} - \mathbf{X}\beta)'(\mathbf{Y} - \mathbf{X}\beta)$  intersect the constraint .

For sufficiently small  $C$ , i.e., large  $\lambda$ , the constraint leads to the selection of either  $\beta_2$  or  $\beta_1$ .

In contrast, the smooth circle from the ridge regression,  $\beta_1^2 + \beta_2^2 = C$ , is tangent to the contours of  $(\mathbf{Y} - \mathbf{X}\beta)'(\mathbf{Y} - \mathbf{X}\beta)$  at  $\beta_1 \neq 0$  and  $\beta_2 \neq 0$ .

# Obtaining Lasso estimates

Unfortunately, there is no closed form solution for the Lasso.

Fortunately, it is convex problem.

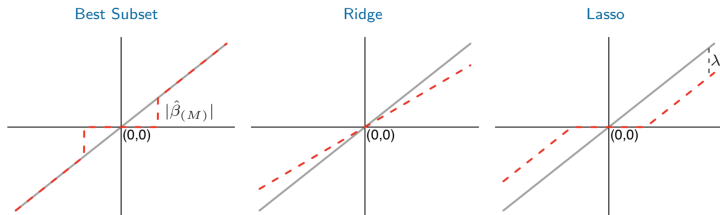
In the R package `glmnet`, there are efficient ways to calculate the solutions for a set of  $\lambda_k$ .

In the **special case** of orthogonal and standardized covariates, we have

$$\hat{\beta}_j^{Lasso} = \text{sign}(\hat{\beta}_j^{OLS})(|\hat{\beta}_j^{OLS} - \lambda|)_+.$$

# Perspectives on Lasso and Ridge: Orthonormal covariates

Estimator	Formula
Best subset (size $M$ )	$\hat{\beta}_j \cdot I( \hat{\beta}_j  \geq  \hat{\beta}_{(M)} )$
Ridge	$\hat{\beta}_j / (1 + \lambda)$
Lasso	$\text{sign}(\hat{\beta}_j)( \hat{\beta}_j  - \lambda)_+$



**Figure:** Table 3.4 from ESL II. Closed forms for Best Subset, Ridge, and Lasso for the **special case** of orthonormal covariates.



# Ridge Regression and Bayes Estimates

In the Bayesian framework,  $f(\boldsymbol{\beta}|\mathbf{y}) \propto f(\mathbf{y}|\boldsymbol{\beta})f(\boldsymbol{\beta})$ .

Let's assume a Gaussian likelihood with variance  $\sigma^2$  and independent mean-zero Gaussian priors on  $\boldsymbol{\beta}$  with common variance  $\tau^2$ :

$$f(\boldsymbol{\beta}|\mathbf{y}) \propto (2\pi\sigma^2)^{-n/2} \exp \left\{ - \sum_{i=1}^n (y_i - \mathbf{x}_i' \boldsymbol{\beta})^2 / (2\sigma^2) \right\} \\ \times (2\pi\tau^2)^{-p/2} \exp \left\{ - \sum_{j=1}^p \beta_j^2 / (2\tau^2) \right\}$$

# Bayesian Perspective on Ridge Regression

Assumes  $\sigma^2$  and  $\tau^2$  are given.

On the log scale, and dropping constants, the maximum a posteriori estimate (i.e., mode) is obtained as

$$\begin{aligned}\hat{\boldsymbol{\beta}}^{MAP} &= \underset{\boldsymbol{\beta}}{\operatorname{argmax}} \quad -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mathbf{x}_i' \boldsymbol{\beta})^2 - \frac{1}{2\tau^2} \sum_{j=1}^p \beta_j^2 \\ &= \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \quad \sum_{i=1}^n (y_i - \mathbf{x}_i' \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^p \beta_j^2.\end{aligned}$$

for  $\lambda = \sigma^2 / \tau^2$ .

Thus,  $\hat{\boldsymbol{\beta}}^{Ridge}$  can be derived from the Bayesian perspective in which the prior distribution is **Gaussian** and the **prior precision** determines the degree of shrinkage.

# Bayesian Perspective on the Lasso

The Laplace distribution (also called double exponential) is a shrinkage prior.

For mean zero,  
 $f(\beta_j) = \frac{1}{2\tau} \exp(-|\beta_j|/\tau)$ , which has  
variance  $2\tau^2$ .

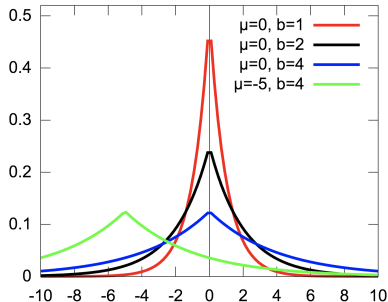


Figure: [https://en.wikipedia.org/wiki/Laplace\\_distribution](https://en.wikipedia.org/wiki/Laplace_distribution)

# Bayesian Perspective on the Lasso

Then we have

$$\begin{aligned}\hat{\boldsymbol{\beta}}^{MAP} &= \operatorname{argmax}_{\boldsymbol{\beta}} -\frac{1}{2\sigma^2} \sum_{i=1}^n (y_i - \mathbf{x}'_i \boldsymbol{\beta})^2 - \frac{1}{2\tau} \sum_{j=1}^p |\beta_j| \\ &= \operatorname{argmin}_{\boldsymbol{\beta}} \sum_{i=1}^n (y_i - \mathbf{x}'_i \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^p |\beta_j|.\end{aligned}$$

for  $\lambda = \sigma^2/\tau$ .

Thus, we again see a stronger prior (smaller  $\tau$ ) leads to greater shrinkage. For sufficiently large  $\tau$ , we get variable selection.

## Selecting $\lambda$ : Cross-validation

Recall from M5.2:

- In  $K$ -Fold CV, the data are divided into  $K$  subsets

$$MSE_k = \frac{1}{n/K} \sum_{i \in S_k} (y_i - \hat{y}_i^{-S_k})^2$$

$$CV = \widehat{MSE} = \frac{1}{K} \sum_{k=1}^K MSE_k$$

- Each partition has a training dataset with  $(K - 1) * n/K$  observations and test dataset with  $n/K$  observations.
- Then the model is fit  $K$  times.
- LOOCV is a special case where the number of folds is  $n$ .

## K-fold Cross-Validation

$$MSE_k = \frac{1}{n/K} \sum_{i \in \mathcal{S}_k} (y_i - \hat{y}_i^{-\mathcal{S}_k})^2$$

$$CV = \widehat{MSE} = \frac{1}{K} \sum_{k=1}^K MSE_k$$

1	2	3	4	5
Train	Train	Validation	Train	Train

Figure: R Tibshirani Lecture notes, <http://www.stat.cmu.edu/~ryantibs/datamining/lectures/18-val1.pdf>.

## Prediction error to choose $\lambda$

Note: the goal of CV is to improve prediction.

The resulting choice of  $\lambda$  may not be optimal for addressing our goal, which is to select the most important covariates.

CV tends to over-select the number of parameters.

There are heuristics such as the one standard error rule:

- Find  $\lambda_{min}$  using 10-fold CV.
- Estimate the variance of the CV using each fold's  $MSE_k$  as a sample:

$$\frac{1}{K} \sum_{k=1}^K (MSE_k - \widehat{MSE_k})^2$$

- Increase  $\lambda$  to result in an MSE that is +1 SE of CV for  $\lambda_{min}$ .

## Correlated covariates

It turns out that variable selection via the Lasso is **sensitive to correlation** between your covariates.

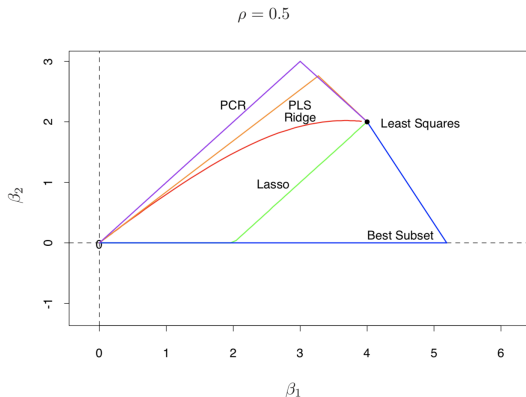
For two correlated covariates, Lasso picks one but not the other, e.g.,  $\hat{\beta}_1$  is set to zero while  $\hat{\beta}_2$  is not.

For small samples, which variable is chosen can be arbitrary.

Ridge regression is less sensitive to correlation between covariates, and shrinks covariates together.



# Variable Selection Methods for Correlated Covariates



**Figure:** Figure 3.18 from ESL II. Correlation between  $x_1$  and  $x_2$  is 0.5. The true regression coefficients are  $(4, 2)$ . Note that Ridge and Lasso are estimated over a continuous range of  $\lambda$ , whereas other methods correspond to the discrete steps selecting 0, 1, or 2 variables.

# Elastic Net

The **elastic net** offers a middle ground and uses a combination of the two. For  $\alpha \in (0, 1)$ ,

$$\hat{\beta}^{ElNet} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n (y_i - \mathbf{x}'_i \beta)^2 + \lambda \sum_{j=1}^p (\alpha |\beta_j| + (1 - \alpha) \beta_j^2).$$

Elastic net tends to shrink some coefficients to zero, like the lasso, but shrinks correlated covariates together, like ridge.

In the R package `glmnet`, the penalty is  $\lambda \sum_{j=1}^p \{\alpha |\beta_j| + (1 - \alpha) \beta_j^2 / 2\}$ .

According to the vignette, when variables are correlated in groups,  $\alpha = 0.5$  tends to result in variables in the group either all selected or dropped.

Note in `glmnet`,  $\alpha = 0 = \text{ridge}$ ,  $\alpha = 1 = \text{Lasso}$ .

# GLM Elastic Net

We can also use penalized regression in the generalized linear model framework. Let  $\ell$  be the log density of  $y_i$ .

$$\hat{\boldsymbol{\beta}}^{Elnet} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \quad - \sum_{i=1}^n \ell(y_i; \mathbf{x}_i' \boldsymbol{\beta}) + \lambda \sum_{j=1}^p (\alpha |\beta_j| + (1 - \alpha) \beta_j^2) .$$

Note that rather than penalizing the squared loss, here we penalize the negative log-likelihood.

With the penalty, the objective function is maximized at smaller values of  $\beta_j$  relative to the MLE.

## Cross-validation and loss functions

For Gaussian data, we used the PMSE in cross-validation:

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2.$$

In GLMs, we use the negative log likelihood, which is often multiplied by two and called the **deviance**:

$$D = -2 \sum_{i=1}^n \ell(y_i; \mathbf{x}'_i \boldsymbol{\beta})$$

.

Note that for iid Gaussian data with variance 1, the deviance equals

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2.$$

For CV, we can use the deviance in place of PMSE. Other options are available.

# The R package glmnet

The R package `glmnet` contains a function, also called `glmnet`

Some of the options are listed below:

```
glmnet(x, y, family = c("gaussian", "binomial", "poisson", "multinomial",  
"cox", "mgaussian"), alpha = 1, nlambda = 100,  
lambda = NULL, standardize = TRUE, intercept = TRUE,
```

A key parameter here is `alpha`: the elastic net mixing parameter with  $0 \leq \alpha \leq 1$ . The penalty is defined as

$$\frac{1 - \alpha}{2} \|\beta\|_2^2 + \alpha \|\beta\|_1$$

`alpha=1` is the lasso penalty, and `alpha=0` the ridge penalty.

Standardizes data by default.

No “data” argument.

# Leukemia Dataset: Ridge Regression

```
> x = Leukemia$x
> y = Leukemia$y
> model.ridge = glmnet(x,y,alpha=0,family = 'binomial')
> print(model.ridge)
```

```
Call: glmnet(x = x, y = y, family = "binomial", alpha = 0)
```

	Df	%Dev	Lambda
1	3571	0.00	409.30
2	3571	18.35	390.70
3	3571	19.01	372.90
4	3571	19.69	356.00
5	3571	20.38	339.80
6	3571	21.09	324.40
7	3571	21.82	309.60
8	3571	22.56	295.60
9	3571	23.32	282.10
10	3571	24.09	269.30

# Leukemia Dataset: Ridge Regression

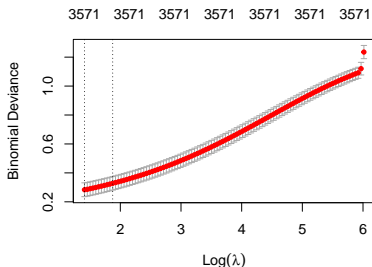
```
> dim(coef(model.ridge))
[1] 3572 100
> coef(model.ridge)[1:10,1:5]
10 x 5 sparse Matrix of class "dgCMatrix"

      s0      s1      s2      s3      s4
(Intercept) -6.312718e-01 -6.449201e-01 -6.459674e-01 -6.470840e-01 -6.482734e-01
V1          -4.780168e-38 -1.203411e-04 -1.259193e-04 -1.317424e-04 -1.378195e-04
V2           3.994960e-38  8.905586e-05  9.273885e-05  9.654906e-05  1.004888e-04
V3           9.777714e-38  2.086957e-04  2.169559e-04  2.254734e-04  2.342506e-04
V4           3.692679e-38  7.792437e-05  8.089428e-05  8.394169e-05  8.706528e-05
V5          -9.494342e-39 -1.533100e-05 -1.579680e-05 -1.626987e-05 -1.675012e-05
V6          -3.815336e-38 -1.003254e-04 -1.050186e-04 -1.099108e-04 -1.150079e-04
V7          -7.641146e-38 -1.800051e-04 -1.878006e-04 -1.958899e-04 -2.042796e-04
V8          -8.192134e-38 -1.976823e-04 -2.064352e-04 -2.155335e-04 -2.249867e-04
V9           2.647430e-38  4.418512e-05  4.547821e-05  4.677791e-05  4.808161e-05
```

# Leukemia Dataset: Ridge Regression

Fit model using 10-fold CV to select lambda:

```
> set.seed(123)
> cv.model.ridge = cv.glmnet(x,y,alpha=0,family='binomial')
> plot(cv.model.ridge)
```



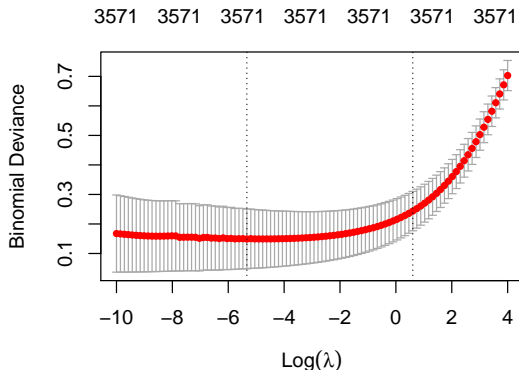
The default values of lambda did not work.



## Leukemia Dataset: Ridge Regression

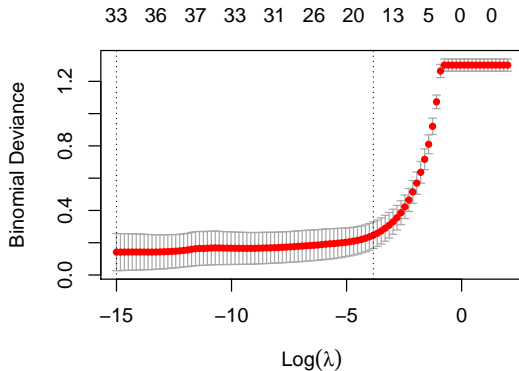
Refit using smaller values of lambda to get a minimum:

```
> lambda = exp(seq(4,-10,length=100))  
> cv.model.ridge = cv.glmnet(x,y,alpha=0,lambda = lambda,family='binomial')  
> cv.model.ridge$lambda.min  
[1] 0.00482795  
> cv.model.ridge$lambda.1se  
[1] 1.833195
```



# Leukemia Dataset: Lasso

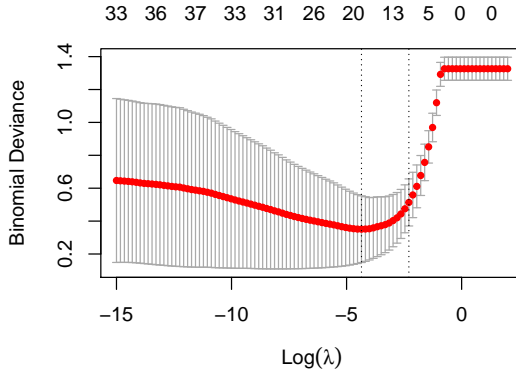
```
> set.seed(123)
> lambda = exp(seq(2,-15,length=100))
> cv.model.lasso = cv.glmnet(x,y,alpha=1,lambda=lambda,family='binomial')
> plot(cv.model.lasso)
```



## Leukemia Dataset: Lasso

Unfortunately, the CV estimate can be highly variable.

This is running the same code but using a different random partition of the data into folds:

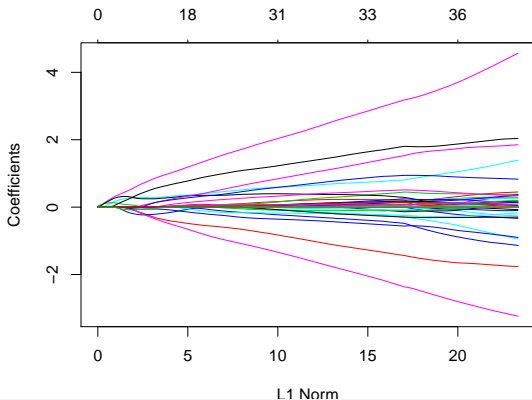


## Coefficient Paths

We can also look at coefficient paths.

These plot the coefficients across the grid of  $\lambda$ .

Variables that enter the model first are more important (when using standardization).



# Leukemia Dataset: Elastic Net

For elastic net, a typical default is  $\alpha = 0.5$ .

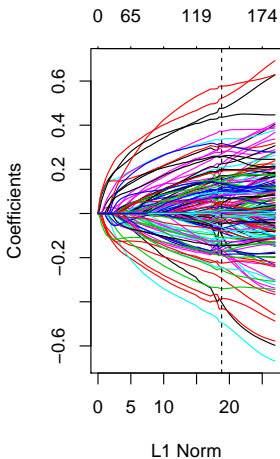
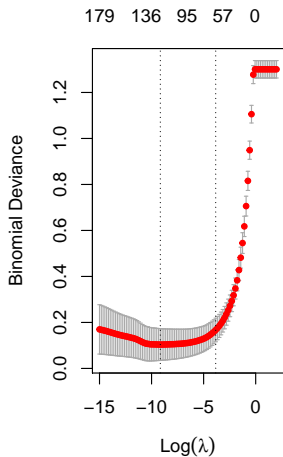
You could try to perform a 2D-CV varying  $\lambda$  and  $\alpha$ , but this is computationally costly and may not work very well due to high CV variability.

Elastic net with  $\alpha = 0.5$  is a good default for high-dimensional regression. Tends to be more reliable than lasso, and unlike ridge, achieves variable selection.

To select fewer variables, increase  $\alpha$ .

```
> set.seed(123)
> cv.model.elnet = cv.glmnet(x,y,alpha=0.5,lambda=lambda,family='binomial')
> elnet.estimates = coef(cv.model.elnet,s=cv.model.elnet$lambda.min)
> model.elnet = glmnet(x,y,alpha=0.5,lambda=lambda,family='binomial')
> plot(model.elnet)
> abline(v=sum(abs(elnet.estimates)),lty=2)
```

# Leukemia Dataset: Elastic Net



## What about inference?

Inference in the Lasso and penalized regression is an active area of research.

Recent distributional results are available when conditioning on the selection event: Lee, J. D., Sun, D. L., Sun, Y., & Taylor, J. E. (2016). Exact post-selection inference, with application to the lasso. *The Annals of Statistics*, 44(3), 907-927.

Since we are also choosing the tuning parameter via cross-validation, “conditioning on the selection event” is not incorporating all sources of uncertainty.

Another promising approach is “knockoffs,” where you create “knockoffs” of  $\mathbf{X}$  unrelated to  $\mathbf{Y}$  and estimate the distribution of test statistics from these knockoffs: Barber, R. F., Candès, E. J., & Samworth, R. J. (2020). Robust inference with knockoffs. *Annals of Statistics*, 48(3), 1409-1431.

## Other issues with inference

For elastic net and lasso using `glmnet`, we have replaced inference with variable selection and CV.

A common approach to large  $p$  (high dimensional) studies is to conduct thousands of univariate tests.

E.g.,  $t$  tests for comparing cancer and control in gene expression.

Multiple comparisons is a problem: when conducting thousands of tests, we need to consider the probability that we falsely reject null hypotheses.



## Note on Bonferroni Correction

When we conduct many hypothesis tests, we increase the chance of false detections.

Let  $H_1, \dots, H_m$  be a family of null hypothesis tests, and  $p_1, \dots, p_m$  their associated p-values.

Here we use  $m$  instead of  $p$  predictors due to notational conflict.

The **family-wise error rate** is the probability of at least one false positive.

Let  $m_0 < m$  be the number of true null hypotheses.

We control the FWER if

$$P \left\{ \bigcup_{j=1}^{m_0} (H_j \text{ is rejected} \mid H_j \text{ is true}) \right\} < \alpha.$$

## Bonferroni correction, cont.

The Bonferroni correction is popular in part due to its simplicity: let  $\alpha^* = \alpha/m$ , where  $\alpha$  is typically 0.05.

$$\begin{aligned} P \left\{ \bigcup_{j=1}^{m_0} (p_j \leq 0.05/m) \right\} &\leq \sum_{j=1}^{m_0} P(p_j \leq 0.05/m) \\ &= m_0 * 0.05/m \\ &\leq m * 0.05/m \\ &= 0.05. \end{aligned}$$

So when we control the individual tests at  $\alpha^*$ , the FWER is less than or equal to  $\alpha$ .

## Bonferroni Correction, cont.

The Bonferroni Correction is a very conservative approach to correcting for multiple comparisons in variable selection.

It is often *too* conservative, and we will not find much (or even any) signal for very large  $p$ .

Other methods can be more powerful: controlling the false discovery rate (FDR), which is the proportion of falsely rejected null hypotheses, i.e., among the set of rejected null hypotheses, what proportion were in fact Type 1 errors.

There are also more powerful approaches to FWER, particularly under some types of dependence.