Linear Models in Matrix Form

$$y = X\beta + \varepsilon$$

$$y = \begin{pmatrix} y^{(1)} \\ \vdots \\ y^{(n)} \end{pmatrix}, \quad X = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & & & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_p \end{pmatrix} \text{ and } \varepsilon = \begin{pmatrix} \varepsilon^{(1)} \\ \vdots \\ \varepsilon^{(n)} \end{pmatrix}.$$

Estimating β

Suppose we have a vector data y from a linear model

$$y = X\beta + \varepsilon,$$

where $X is a known design matrix.

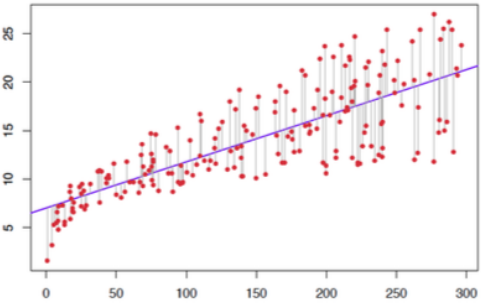
To estimate the parameter vector β we can use a least-square approach: Find $\widehat{\beta} = (\widehat{\beta}_0, \ldots, \widehat{\beta}_p)^\top$ such that

$$\sum_{i=1}^{n} (y^{(i)} - \{\widehat{\beta}_0 + \widehat{\beta}_1 x_{i1} + \widehat{\beta}_2 x_{i2} + \cdots + \widehat{\beta}_p x_{ip}\})^2 \quad \text{is minimal.}$$

We write as:

$$\begin{aligned} \widehat{\beta}^{\text{OLS}} &= \underset{\beta \in \mathfrak{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} (y^{(i)} - \beta^\top x^{(i)})^2 \\ &= \underset{\beta \in \mathfrak{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{least squares}} \end{aligned}$$

Least Squares



Estimating β

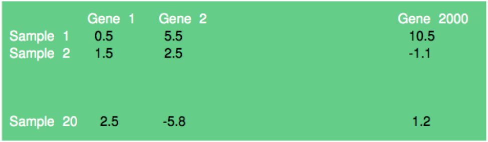
It can be shown that this gives the least squares estimate

$$\widehat{\beta}^{\text{OLS}} = (X^\top X)^{-1} X^\top y,$$

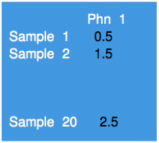
where $(X^\top X)^{-1}$ is the inverse of the matrix $X^\top X$.

The issue with large datasets: p»n

- Simple linear model: $y = X\beta + \varepsilon$



	Gene 1	Gene 2	Gene 2000
Sample 1	0.5	5.5	10.5
Sample 2	1.5	2.5	-1.1
Sample 20	2.5	-5.8	1.2



	Phn 1
Sample 1	0.5
Sample 2	1.5
Sample 20	2.5

$$\widehat{\beta}^{\text{OLS}} = (X^\top X)^{-1} X^\top y,$$

$\hookrightarrow$ $(X^\top X)$ is not invertible if $p > n$ or in presence of strong colinearity.

Alternative: Ridge regression

Ridge regression is like least squares but shrinks the estimated coefficients towards zero.

$$
\begin{aligned}
\widehat{\beta}^{\text{ridge}} &= \operatorname*{argmin}_{\beta \in \Re^p} \sum_{i=1}^{n} (y^{(i)} - \beta^\top x^{(i)})^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \\
&= \operatorname*{argmin}_{\beta \in \Re^p} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}
\end{aligned}
$$

Here $\lambda \geqslant 0$ is a tuning parameter, which controls the strength of the penalty term.

Ridge regression

Theorem
The solution to the ridge regression problem is given by

$$\widehat{\beta}^{\text{ridge}} = (X^T X + \lambda \mathbb{I})^{-1} X^T y$$

Remind that:

$$\widehat{\beta}^{OLS} = (X^T X)^{-1} X^T y$$

Note the similarity to the ordinary least squares solution, but with the addition of a ”ridge” down the diagonal.

Implementation: OLS and Ridge

```
matX <- matrix(rnorm(20*10),ncol=20,nrow=10)
beta <- c(rep(1,5),rep(0,15))
beta
```

```
 [1] 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
y <- matX%*%beta + rnorm(10,sd=0.1)
mod <- lm(y~matX)
summary(mod)
```

Implementation: OLS and Ridge

```
Call:
lm(formula = y ~ matX - 1)

Residuals:
ALL 10 residuals are 0: no residual degrees of freedom!

Coefficients: (10 not defined because of singularities)
       Estimate Std. Error t value Pr(>|t|)
matX1   0.78954        NaN     NaN      NaN
matX2   0.93865        NaN     NaN      NaN
matX3   1.17603        NaN     NaN      NaN
matX4   0.78489        NaN     NaN      NaN
matX5   1.20890        NaN     NaN      NaN
matX6  -0.03265        NaN     NaN      NaN
matX7   0.19857        NaN     NaN      NaN
matX8  -0.32279        NaN     NaN      NaN
matX9   0.06134        NaN     NaN      NaN
matX10  0.24272        NaN     NaN      NaN
matX11       NA         NA      NA       NA
matX12       NA         NA      NA       NA
matX13       NA         NA      NA       NA
matX14       NA         NA      NA       NA
matX15       NA         NA      NA       NA
matX16       NA         NA      NA       NA
matX17       NA         NA      NA       NA
matX18       NA         NA      NA       NA
matX19       NA         NA      NA       NA
matX20       NA         NA      NA       NA

Residual standard error: NaN on 0 degrees of freedom
Multiple R-squared:      1, Adjusted R-squared:    NaN
F-statistic:   NaN on 10 and 0 DF,  p-value: NA
```

Inverse the design matrix

```r
solve(t(matX)%*%matX)
# Error in solve.default(t(matX) %*% matX) :
#system is computationally singular: reciprocal condition number = 7.65949e-19
```

Ridge model with MASS package

```r
require(MASS)
lm.ridge(y~matX-1,lambda=0.1)
```

```
##       matX1       matX2       matX3       matX4       matX5       matX6 
##  0.81391545  0.53677792  0.76455741  0.89437115  0.54794319  0.08616459 
##       matX7       matX8       matX9      matX10      matX11      matX12 
## -0.55896591 -0.25240448  0.34549913  0.30462063 -0.25491506  0.28705247 
##      matX13      matX14      matX15      matX16      matX17      matX18 
##  0.04316652 -0.27902114 -0.03797998 -0.22352521 -0.10017168  0.16558377 
##      matX19      matX20 
## -0.25023519 -0.05049615
```

Ridge model with glmnet package

```
require(glmnet)
model <- glmnet(matX,y,alpha=0,lambda = 0.1)
as.vector(model$beta)
```

```
##  [1]  0.89315873  0.47423307  0.60033542  0.81764680  0.52854941  0.08287656
##  [7] -0.54911709 -0.25200421  0.35006137  0.30629861 -0.20745540  0.22934065
## [13]  0.14385838 -0.28709342 -0.04065775 -0.21941942 -0.10915131  0.17062777
## [19] -0.20236951 -0.06212413
```

Result is depending of the choice of the tunning parameter:

↪ **Cross-validation** to tune this parameter using the **cv.glmnet()** function

Ridge vs. OLS in the presence of collinearity

The benefits of ridge regression are most striking in the presence of multicollinearity, as illustrated in the following example:

```r
set.seed(10)
x1 <- rnorm(20)
x2 <- rnorm(20,mean=x1,sd=.01)
## cor(x1,x2) ## 0.9999435
y <- 3+x1+x2+rnorm(20)
lm(y~x1+x2)$coef
```

```
## (Intercept)          x1          x2
##    3.300134  -19.042300   21.333330
```

```r
lm.ridge(y~x1+x2,lambda=1)
```

```
##                x1       x2
## 3.142965 1.083529 1.130301
```

Recap: ridge regression

- We learned ridge regression, which minimizes the usual regression criterion plus a penalty term on the squared L_2 norm of the coefficient vector. As such, it shrinks the coeffients towards zero. This introduces some bias, but can greatly reduce the variance, resulting in a better mean-squared error.
- The amount of shrinkage is controlled by λ, the tuning parameter that multiplies the ridge penalty.
- Large λ means more shrinkage, and so we get different coefficient estimates for different values of λ. Choosing an appropriate value of λ is important, and also difficult.

Recap: ridge regression

- We learned ridge regression, which minimizes the usual regression criterion plus a penalty term on the squared L_2 norm of the coefficient vector. As such, it shrinks the coeffients towards zero. This introduces some bias, but can greatly reduce the variance, resulting in a better mean-squared error.
- The amount of shrinkage is controlled by λ, the tuning parameter that multiplies the ridge penalty.
- Large λ means more shrinkage, and so we get different coefficient estimates for different values of λ. Choosing an appropriate value of λ is important, and also difficult.
- Ridge regression performs particularly well when there is a subset of true coefficients that are small or even zero.

Recap: ridge regression

- We learned ridge regression, which minimizes the usual regression criterion plus a penalty term on the squared L_2 norm of the coefficient vector. As such, it shrinks the coeffients towards zero. This introduces some bias, but can greatly reduce the variance, resulting in a better mean-squared error.
- The amount of shrinkage is controlled by λ, the tuning parameter that multiplies the ridge penalty.
- Large λ means more shrinkage, and so we get different coefficient estimates for different values of λ. Choosing an appropriate value of λ is important, and also difficult.
- Ridge regression performs particularly well when there is a subset of true coefficients that are small or even zero.
- However ridge regression cannot perform variable selection, and even though it performs well in terms of prediction accuracy, it does poorly in terms of offering a clear interpretation

Optimization of ridge regression

- Can rewrite the optimization problem

$$\min_{w_0 \in \Re,\ w \in \Re^p} \sum_{i=1}^{n} \left(y^{(i)} - w_0 - \sum_{j=1}^{p} w_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} w_j^2$$

in the proper objective/constraint form:

$$\min_{w_0 \in \Re,\ w \in \Re^p} \sum_{i=1}^{n} \left(y^{(i)} - w_0 - \sum_{j=1}^{p} w_j x_{ij} \right)^2$$

$$\text{subject to } \sum_{j=1}^{p} w_j^2 \leqslant t$$

- Correspondence $\lambda \Rightarrow t$ can be shown using Lagrance multipliers.

Lasso regression

- The lasso estimate is defined as

$$\begin{aligned}\widehat{\beta}^{\text{lasso}} &= \operatorname*{argmin}_{\beta\in\mathfrak{R}^p} \sum_{i=1}^{n}(y^{(i)} - \beta^\top x^{(i)})^2 + \lambda \sum_{j=1}^{p} |\beta_j| \\ &= \operatorname*{argmin}_{\beta\in\mathfrak{R}^p} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_1}_{\text{Penalty}}\end{aligned}$$

- The objective function to maximize is

$$-\sum_{i=1}^{n}(y^{(i)} - w^T\mathbf{x}^{(i)})^2 - \lambda \sum_{j=1}^{m} |w_j|$$

  where $w \in \mathfrak{R}^m$. (Sorry, we change the notation $m = p$ is the number of predictors)

- It is still concave (i.e. unique maximum), but unfortunately neither closed-form solution nor gradient descent will do the trick.

- The only difference between the lasso problem and ridge regression is that the latter uses a (squared) L_2 penalty $\|\beta\|_2^2$, while the former uses an L_1 penalty $\|\beta\|_1$. But even though these problems look similar, their solutions behave very differently

- Note the name "lasso" is actually an acronym for: Least Absolute Selection and Shrinkage Operator

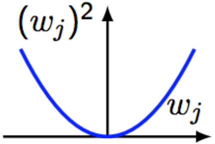
Optimization for Lasso

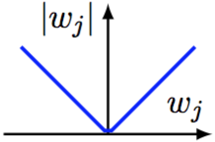
- Similarly, for Lasso:

$$\min_{w_0 \in \Re,\ w \in \Re^m} \sum_{i=1}^{n} \left(y^{(i)} - w_0 - \sum_{j=1}^{m} w_j x_{ij} \right)^2$$

$$\text{subject to } \sum_{j=1}^{m} |w_j| \leqslant t$$

- Compare shape of the penalty as a function of w_j :

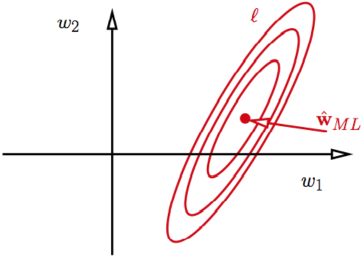




Lasso vs. ridge: geomtery of error surfaces

- An equivalent formulation for L_p regularization: constrained maximization

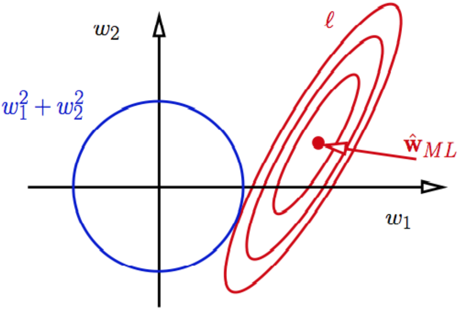
$$\underset{\mathbf{w}:\sum_{j=1}^{m}|w_j|^p \leqslant t}{\operatorname{argmin}} \sum_{i=1}^{n}(y^{(i)} - w^T x^{(i)})^2$$



Lasso vs. ridge: geomtery of error surfaces

- An equivalent formulation for L_p regularization: constrained maximization

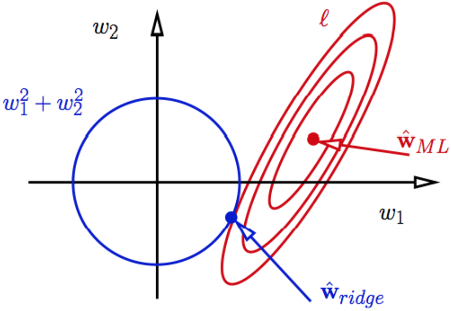
$$\underset{\mathbf{w}:\sum_{j=1}^{m}|w_j|^p\leqslant t}{\operatorname{argmin}} \sum_{i=1}^{n}(y^{(i)} - w^T x^{(i)})^2$$



Lasso vs. ridge: geomtery of error surfaces

- An equivalent formulation for L_p regularization: constrained maximization

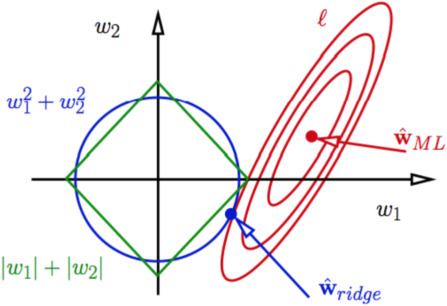
$$\underset{\mathbf{w}:\sum_{j=1}^{m}|w_j|^p \leqslant t}{\operatorname{argmin}} \sum_{i=1}^{n}(y^{(i)} - w^T x^{(i)})^2$$



Lasso vs. ridge: geomtery of error surfaces

- An equivalent formulation for L_p regularization: constrained maximization

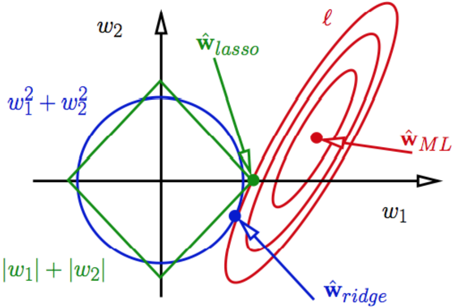
$$\underset{\mathbf{w}:\sum_{j=1}^{m}|w_j|^p\leqslant t}{\operatorname{argmin}} \sum_{i=1}^{n}(y^{(i)} - w^T x^{(i)})^2$$



Lasso vs. ridge: geomtery of error surfaces

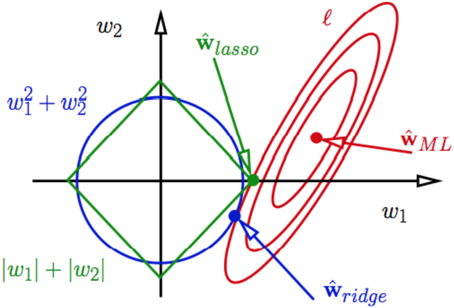
- An equivalent formulation for L_p regularization: constrained maximization

$$\underset{\mathbf{w}:\sum_{j=1}^{m}|w_j|^p \leqslant t}{\operatorname{argmin}} \sum_{i=1}^{n}(y^{(i)} - w^T x^{(i)})^2$$



Lasso vs. ridge: geomtery of error surfaces

$$\underset{\mathbf{w}:\sum_{j=1}^{m}|w_j|^p \leqslant t}{\operatorname{argmin}} \sum_{i=1}^{n} (y^{(i)} - w^T x^{(i)})^2$$



- With sufficiently large λ, lasso leads to sparsity.
- Must explicitly solve the above optimization problem- e.g., using Lagrange multipliers.

Advantage in interpretation: Prostate cancer study

- An an example, consider the data from a 1989 study examining the relationship prostate-specific antigen (PSA) and a number of clinical measures in a sample of 97 men who were about to receive a radical prostatectomy.
- PSA is typically elevated in patients with prostate cancer, and serves a biomarker for the early detection of the cancer
- The explanatory variables:
  - `lcavol`: Log cancer volume
  - `lweight`: Log prostate weight
  - `age`
  - `lbph`: Log benign prostatic hyperplasia
  - `svi`: Seminal vesicle invasion
  - `lcp`: Log capsular penetration
  - `gleason`: Gleason score
  - `pgg45`: % Gleason score 4or 5.

Advantage in interpretation: Prostate cancer study

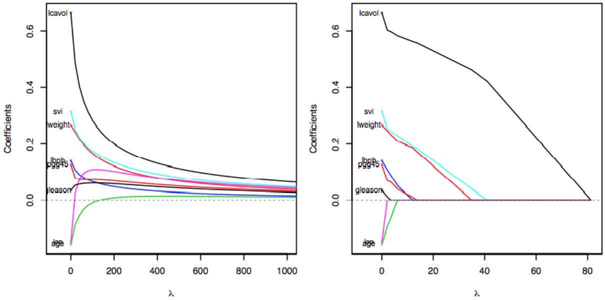
- An an example, consider the data from a 1989 study examining the relationship prostate-specific antigen (PSA) and a number of clinical measures in a sample of 97 men who were about to receive a radical prostatectomy.
- PSA is typically elevated in patients with prostate cancer, and serves a biomarker for the early detection of the cancer
- The explanatory variables:
  - `lcavol`: Log cancer volume
  - `lweight`: Log prostate weight
  - `age`
  - `lbph`: Log benign prostatic hyperplasia
  - `svi`: Seminal vesicle invasion
  - `lcp`: Log capsular penetration
  - `gleason`: Gleason score
  - `pgg45`: % Gleason score 4or 5.

We are interested in identifying a small number of predictors, say 2 or 3, that drive PSA

Advantage in interpretation

On top the fact that the lasso is competitive with ridge regression in terms of this prediction error, it has a big advantage with respect to interpretation. This is exactly because it sets coefficients exactly to zero, i.e., it performs variable selection in the linear model.

Prostate cancer data example:



Choice of λ: degrees of freedom

- For linear regression, $\widehat{y} = X\widehat{\beta}^{\text{OLS}}$, we have $df(\widehat{y}) = p$
- For ridge regression, $\widehat{y} = X\widehat{\beta}^{\text{ridge}}$, where

$$\widehat{\beta}^{\text{ridge}} = \underset{\beta \in \Re^p}{\text{argmin}} ||y - X\beta||_2^2 + \lambda||\beta||_2^2$$

we have $df(\widehat{y}) = trace(X(X^TX + \lambda\mathbf{I})^{-\mathbf{1}}\mathbf{X}^{\mathbf{T}})$

- For the lasso $\widehat{y} = X\widehat{\beta}^{\text{lasso}}$, where

$$\widehat{\beta}^{\text{lasso}} = \underset{\beta \in \Re^p}{\text{argmin}} ||y - X\beta||_2^2 + \lambda||\beta||_1$$

we have $df(\widehat{y})$ =E[number of nonzero coefficients in $\widehat{\beta}^{\text{lasso}}$]

Recap: the lasso before choising λ

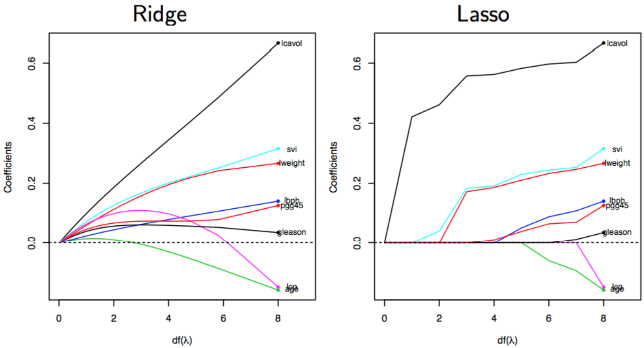
- we learned a variable selection method in the linear model setting: the lasso. The lasso uses a penalty like ridge regression, except the penalty is the L_1 norm of the coefficient vector, which causes the estimates of some coefficients to be exactly zero. This is in constrast to ridge regression which never sets coefficients to zero

- The tuning parameter λ controls the strength of the L_1 penalty. The lasso estimates are generally biased, but have good mean squared error (comparable to ridge regression). On top of this, the fact that it sets coefficients to zero can be a big advantage for the sake of interpretation

- We defined the concept of degrees of freedom, which measures the effective number of parameters used by a estimator. This allows us to compare estimators with different tuning parameters

Comparing ridge and lasso for the prostate cancer data

One usage of degrees of freedom is to put two different estimates on equal footing



Fitting lasso models in R

- The glmnet package can fit a wide variety of models (linear models, generalized linear models, multinomial models, proportional hazards models) with lasso penalties

- The syntax is fairly straightforward, though it differs from lm in that it requires you to form your own design matrix:

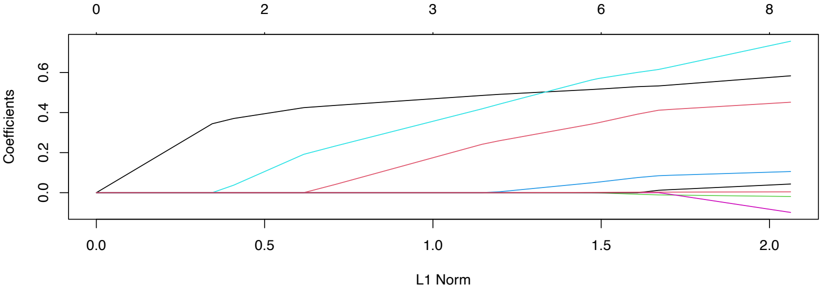
```
fit <- glmnet(X,y)
```

- The package also allows you to conveniently carry out cross-validation:

```
cvfit <- cv.glmnet(X,y)
plot(cvfit)
```

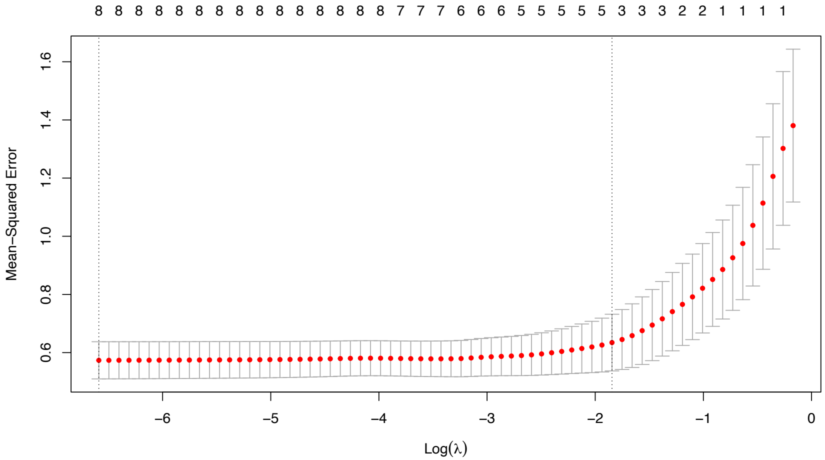
Lasso model on Prostate Cancer data

```
prostate <- read.table(file="prostate.txt"
                         ,header=TRUE)
X <- as.matrix(prostate[,1:8])
y <- prostate[,9]
model.lasso <- glmnet(X,y)
plot(model.lasso)
```



Lasso model on Prostate Cancer data

```r
cv.lasso <- cv.glmnet(X,y)
plot(cv.lasso)
```



Selected variable

```r
final.lasso <- glmnet(X,y,lambda=cv.lasso$lambda.1se)
final.lasso$beta
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                s0
## lcavol  0.4855482
## lweight 0.2415506
## age     .
## lbph    .
## svi     0.4186507
## lcp     .
## gleason .
## pgg45   .
```

Ressources

- Chapter 6.2 Shrinkage Methods (page 237 to 250) of An Introduction to Statistical Learning available here https://www.statlearning.com/
- Chapter 3.4 Shrinkage Methods of The Elements of Statistical Learning https://hastie.su.domains/ElemStatLearn/
- Practice using R:
  - Regularized regression: http://uc-r.github.io/regularized_regression and https://bradleyboehmke.github.io/HOML/regularized-regression.html

Take Home Message

- Ridge regression
- Lasso Model
- Tuning parameter
- Variable selection
- Colinearity
- Regularization

Some References on Lasso

- Friedman J, Hastie T, Tibshirani R. 2010. *Regularization paths for generalized linear models via coordinate descent*. J Stat Soft 33:1-22.

- Simon N, Friedman J, Hastie T, Tibshirani. 2013 *A Sparse-Group Lasso*. Journal of Computational and Graphical Statistics Vol. 22, Iss. 2.

- Tibshirani R (1996). *Regression Shrinkage and Selection via the Lasso*. Journal of the Royal Statistical Society B, 58(1), 267-288.

- Zou H, Hastie T (2005). *Regularization and variable selection via the Elastic Net*. Journal of the Royal Statistical Society B, 67, 301-20.