Introducing Penalized Regression*

Simon Wood

*Reading: chapters 3 and 4 of *Elements of Statistical Learning* Hastie, Tibshirani and Friedman `https://web.stanford.edu/ hastie/ElemStatLearn/`

Linear model

- Consider a linear model for data vector $\mathbf{y}$:

$$\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \mathbf{I}\sigma^2),$$

  where model matrix $\mathbf{X}$ is $n \times p$ while $\boldsymbol{\beta}$ and σ^2 are parameters.
- If $n \geq p$ and $\mathbf{X}$ full column rank then

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 = (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}.$$

- So if $\boldsymbol{\mu} \equiv \mathbb{E}(\mathbf{y})$, the *predicted values* are

$$\hat{\boldsymbol{\mu}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y} = \mathbf{A}\mathbf{y}$$

  where $\mathbf{A} \equiv \mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}$. Note $\mathbf{A}\mathbf{A} = \mathbf{A}$ (idempotent).

Predicted value variance as p grows

- Since $\hat{\boldsymbol{\mu}} = \mathbf{A}\mathbf{y}$ and $\boldsymbol{\Sigma}_y = \mathbf{I}\sigma^2$ the covariance matrix of $\hat{\boldsymbol{\mu}}$ is

$$\boldsymbol{\Sigma}_{\hat{\mu}} = \mathbf{A}\mathbf{I}\mathbf{A}^\mathsf{T}\sigma^2 = \mathbf{A}\mathbf{A}\sigma^2 = \mathbf{A}\sigma^2.$$

- Also $\mathrm{tr}(\mathbf{A}) = \mathrm{tr}\{(\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{X}\} = p$ as $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$.
- So the average variance of $\hat{\mu}_i$ is $\sigma^2 p/n$.
- As the number of parameters, p, increases the average predicted value variance increases until it equals $\mathrm{var}(y_i) = \sigma^2$ when $p = n$.
- If we go further so that $p > n$ then $\mathbf{X}^\mathsf{T}\mathbf{X}$ is no-longer full rank, has no inverse and so $\hat{\boldsymbol{\beta}}$ does not exist.
- But in many applications p is comparable to n or even larger e.g.
  - genetic data where far more genes may be screened than patients.
  - models in which each predictor has a complex relationship with the response, y, requiring many parameters per predictor.

$p > n$ and Tikhonov regularization

- A simple approach to rank deficiency of $\mathbf{X}^\mathsf{T}\mathbf{X}$ when $p > n$ is regularization, replacing $\mathbf{X}^\mathsf{T}\mathbf{X}$ by $\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I}$ for some $\lambda > 0$.
- Since $\mathbf{X}^\mathsf{T}\mathbf{X}$ is rank n but clearly $\boldsymbol{\beta}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\beta} \geq 0$ for any $\boldsymbol{\beta}$, $\mathbf{X}^\mathsf{T}\mathbf{X}$ is positive semi-definite with at least $p - n$ zero eigenvalues.
- Hence eigenvalues$(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I}) \geq \lambda$ implying that $\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I}$ is strictly positive definite and hence invertible.
- So we have the regularized estimator

$$\hat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}.$$

- Notice how this is also

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2$$

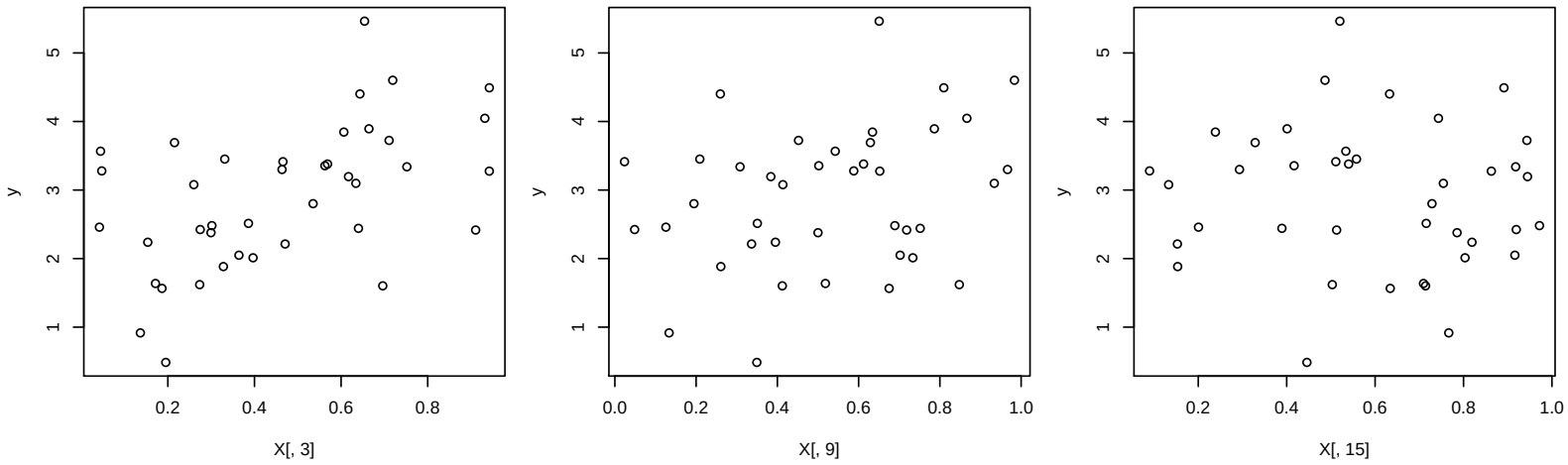
— $\lambda\|\boldsymbol{\beta}\|_2^2$ is the 'L_2 penalty' or 'ridge penalty'.

Simulated toy example

- $n = 40, p = 50, X_{ij} \underset{\text{i.i.d.}}{\sim} U(0,1), \beta_{1:10} \sim U(0,1), \beta_{11:50} = 0.$

```
n <- 40;p <- 50
X <- matrix(runif(n*p),n,p)   ## predictors
mu <- X[,1:10] %*% runif(10) ## mean response
y <- mu + rnorm(n)            ## response
```

- The relationships in the resulting data are not very clear...

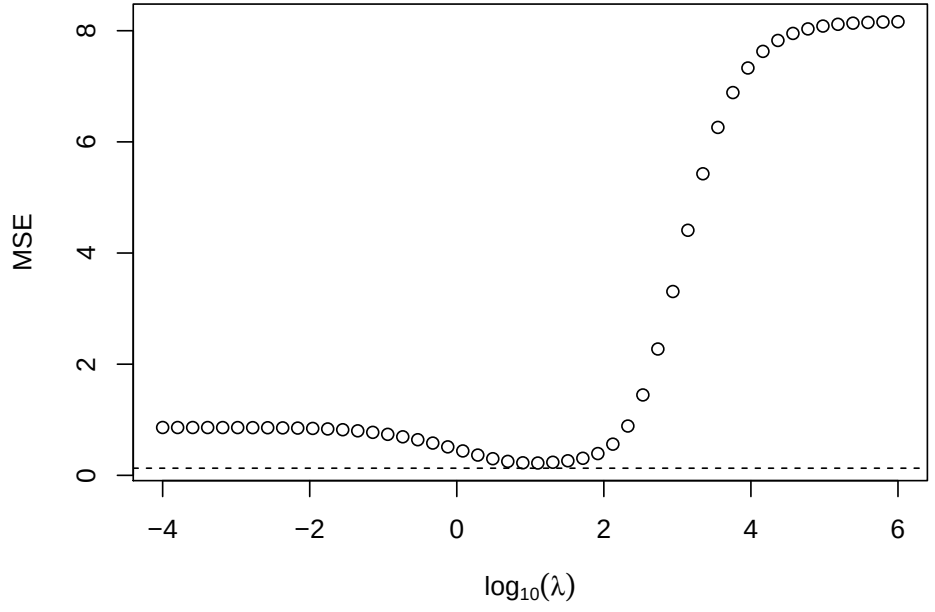


Does penalization achieve anything?

- Compute mean square error, MSE $= \sum_{i=1}^{n}(\mu_i - \hat{\mu}_i)^2/n$, as a function of λ where $\hat{\boldsymbol{\mu}} = \mathbf{X}\hat{\boldsymbol{\beta}}_\lambda$.

```
mse <- lam <- 10^seq(-4,6,length=50)
for (i in 1:length(lam)) {
  b.hat <- solve(crossprod(X)+lam[i]*diag(p),t(X)%*%y)
  mu.hat <- X %*% b.hat
  mse[i] <- mean((mu-mu.hat)^2)
}
```

- Here is a plot. Dashed line is MSE for regression on `X[,1:10]`



Choosing λ in practice

- So, with the right λ, Tikhonov regularization/ridge regression performs reasonably well in MSE terms.
- But we do not know μ for real data so can not directly optimise MSE in practice.
- We have the noisy observations $y_i = \mu_i + \epsilon_i$ but $\sum_{i=1}^{n}(y_i - \hat{\mu}_i)^2$ is minimised by $\lambda \to 0$.
- This overfitting is avoided if we measure the ability of the model to predict data *to which it was not fitted*.
- One such approach is leave-one-out cross validation. Let $\hat{\mu}_i^{[-i]}$ be the prediction of y_i from a model fitted to all the data except y_i.
- Find the λ minimizing

$$\text{OCV} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\mu}_i^{[-i]})^2$$

Leave one out cross validation

- In the penalized case $\mathbf{A} = \mathbf{X}(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}$. It turns out that

$$\text{OCV} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\mu}_i^{[-i]})^2 = \frac{1}{n}\sum_{i=1}^{n}\frac{(y_i - \hat{\mu}_i)^2}{(1 - A_{ii})^2}$$

- So OCV is relatively cheap to compute - it only needs 1 fit, not n, for each λ. For example...

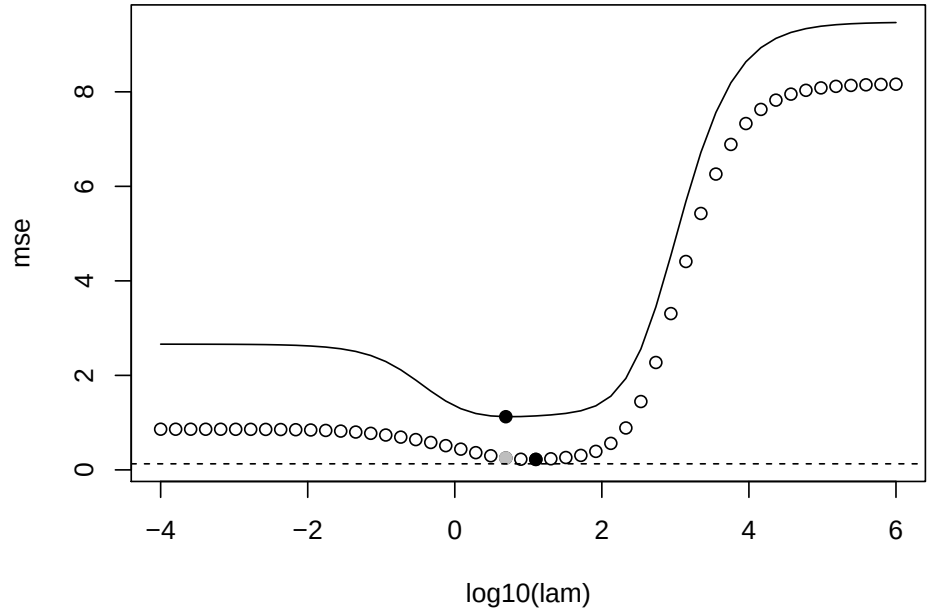
```
for (i in 1:length(lam)) {
  A <- X%*% solve(crossprod(X)+lam[i]*diag(p),t(X))
  mu.hat <- A %*% y
  ocv[i] <- mean((y-mu.hat)^2/(1-diag(A))^2)
}
```

- Note that there are more efficient ways of structuring the OCV computation than explicitly forming $\mathbf{A}$[†].

[†] `diagA <- colSums(t(X)*solve(crossprod(X)+lam[i]*diag(p),t(X))` would be a start.

OCV example

- Here is a plot of the OCV score against $\log \lambda$ as a black curve. The MSE and *oracle* MSE from the true model fit are shown too.



- The black dots show the OCV and MSE minima, and the grey dot the MSE of the OCV optimal λ.
- The noise in y_i relative to μ_i means that the OCV and MSE optima are not identical, but they are close and the OCV not far from optimal in MSE terms.

Other validation/cross validation methods

- OCV is only one possibility...
  1. Generalized cross-validation (GCV) replaces the A_{ii} with their average in OCV for yet greater computational efficiency and some extra invariance.
  2. $k-$fold cross-validation divides the data into k blocks and bases cross validation on the ability to predict each block when it is omitted from the fit. $k = 5$ or 10 are typical.
  3. Leave-out-several cross validation leaves out and predicts a few observations at a time. For small numbers left out, efficient computation from a single fit is possible, similarly to OCV.
  4. Sometimes data are simply split into a fit set and a validation set. The validation data are predicted, but never fitted.
- Note that for non-Gaussian likelihoods we would usually substitute the likelihood or deviance for sums of squares.

Taking stock

- We have seen empirically that adding a simple L_2 penalty to a regression problem can help to improve model predictive performance and even allow model fitting at all.
- Prediction error criteria such as OCV give a practical way to optimize the strength of penalization to use.
- What price has been paid for improved predictive performance?
- $\mathbb{E}(\hat{\boldsymbol{\beta}}_\lambda) = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbb{E}(\mathbf{y}) = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\beta} \neq \boldsymbol{\beta}$.
- So $\hat{\boldsymbol{\beta}}_\lambda$ is biased. The penalty has led to *shrinkage*.
- The bias is $\mathbf{b} = \mathbb{E}(\hat{\boldsymbol{\beta}}_\lambda) - \boldsymbol{\beta} = -\lambda(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\boldsymbol{\beta}$ which increases with penalization.
- Good MSE performance has been obtained by trading variance for bias.

A Bayesian perspective

- A Bayesian view of regularization/penalized regression helps.
- Consider a linear model with simple Gaussian prior

$$\mathbf{y} \sim N(\mathbf{X}\boldsymbol{\beta}, \mathbf{I}\sigma^2) \qquad \boldsymbol{\beta} \sim N(\mathbf{0}, \mathbf{I}\sigma^2/\lambda)$$

- Since $\pi(\boldsymbol{\beta}|\mathbf{y}) \propto \pi(\mathbf{y}|\boldsymbol{\beta})\pi(\boldsymbol{\beta})$ it follows immediately that

$$\log \pi(\boldsymbol{\beta}|\mathbf{y}) = -\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2/(2\sigma^2) - \boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta}\lambda/(2\sigma^2) + k$$

  where k is a constant not dependent on $\boldsymbol{\beta}$.
- Discarding k and multiplying by $-2\sigma^2$ we have the L_2 penalized regression problem, so $\hat{\boldsymbol{\beta}}_\lambda$ is clearly the *posterior mode* for $\boldsymbol{\beta}$.

The distribution of $\boldsymbol{\beta}|\mathbf{y}$

- If we are happy to treat the ridge penalty as induced by a Gaussian prior on $\boldsymbol{\beta}$ then we can also consider the posterior.
- To find the posterior of $\boldsymbol{\beta}$ we must 'complete the square'...

$$\begin{aligned}
\log \pi(\boldsymbol{\beta}|\mathbf{y}) &= -\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2/(2\sigma^2) - \boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta}\lambda/(2\sigma^2) + k \\
&= -(\mathbf{y}^\mathsf{T}\mathbf{y} + \boldsymbol{\beta}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\beta} + \lambda\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\beta} - 2\mathbf{y}^\mathsf{T}\mathbf{X}\boldsymbol{\beta})/(2\sigma^2) + k \\
&= -\frac{1}{2\sigma^2}(\boldsymbol{\beta} - (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y})^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})(\boldsymbol{\beta} - (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}) + k' \\
&= -\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_\lambda)^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})(\boldsymbol{\beta} - \hat{\boldsymbol{\beta}}_\lambda) + k'
\end{aligned}$$

where $\hat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$, as before.

- Hence

$$\boldsymbol{\beta}|\mathbf{y} \sim N(\hat{\boldsymbol{\beta}}, (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda\mathbf{I})^{-1}\sigma^2).$$

Bayes and bias-variance

- The cov. matrix of $\mathbf{y}$ is $\mathbf{I}\sigma^2$ and $\hat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\intercal\mathbf{y}$ so

$$\boldsymbol{\Sigma}_{\hat{\boldsymbol{\beta}}} = (\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\intercal\mathbf{X}(\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1}\sigma^2.$$

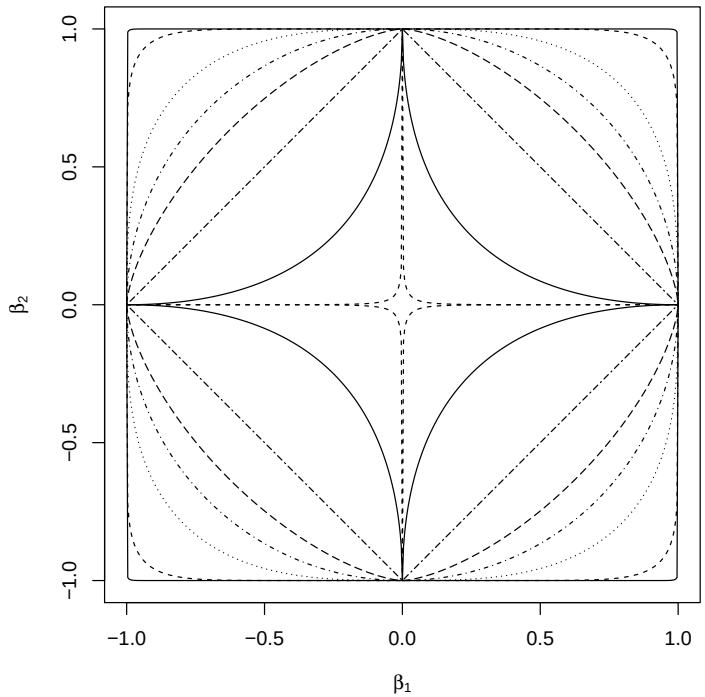
- Recall the bias $\mathbf{b} = -\lambda(\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1}\boldsymbol{\beta}$. According to the prior

$$\begin{aligned}\mathbb{E}(\mathbf{b}\mathbf{b}^\intercal) &= \lambda^2(\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbb{E}(\boldsymbol{\beta}\boldsymbol{\beta}^\intercal)(\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1} \\ &= \lambda\sigma^2(\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1}(\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1}\end{aligned}$$

- Hence $\boldsymbol{\Sigma}_{\hat{\boldsymbol{\beta}}} + \mathbb{E}(\mathbf{b}\mathbf{b}^\intercal) = (\mathbf{X}^\intercal\mathbf{X} + \lambda\mathbf{I})^{-1}\sigma^2$.
- So the Bayesian posterior covariance matrix for $\boldsymbol{\beta}$ can be viewed as the sum of the covariance of $\hat{\boldsymbol{\beta}}$ and expected squared bias of $\hat{\boldsymbol{\beta}}$.
- The bias component increases and the variance component decreases with λ.

Other penalties

- So far we considered only the L_2 penalty $\|\boldsymbol{\beta}\|_2^2 = \sum_{i=1}^{p} \beta_i^2$.
- We could consider other L_q penalties $\mathcal{P}_q(\boldsymbol{\beta}) = \sum_{i=1}^{p} |\beta_i|^q$. ‡
- Here is a plot of the unit contour for 2-d penalties, $\mathcal{P}_q$, with $q = 100, 10, 3, 2, 1.5, 1, 0.5, 0.2$ (working inwards).



- The limiting 'L_0' penalty simply counts the non-zero β_is.

‡These penalties can define vector norms for $q \geq 1$, but not for $q < 1$.

Basic penalty geometry

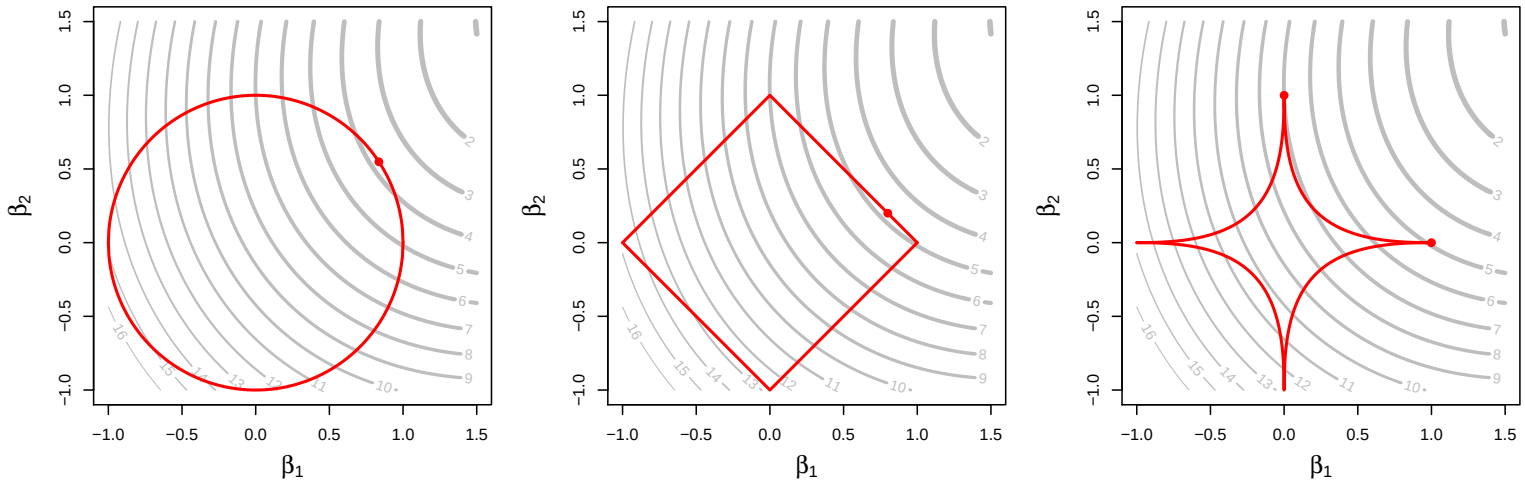
- Consider the penalized fitting problem

$$\hat{\boldsymbol{\beta}}_\lambda = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, D(\boldsymbol{\beta}) + \lambda \mathcal{P}_q(\boldsymbol{\beta})$$

  where $D(\boldsymbol{\beta})$ is some fitting objective (e.g. residual sum of squares or negative log likelihood).
- Obviously $\hat{\boldsymbol{\beta}}_\lambda$ lies on some contour of the penalty, $\mathcal{P}_q(\boldsymbol{\beta}) = c$.
- Equally obviously $\hat{\boldsymbol{\beta}}_\lambda$ must be the minimiser of $D(\boldsymbol{\beta})$ along the contour. i.e. $\hat{\boldsymbol{\beta}}_\lambda = \operatorname{argmin}_{\boldsymbol{\beta}} D(\boldsymbol{\beta})$ s.t. $\mathcal{P}_q(\boldsymbol{\beta}) = c$.
- Otherwise we could reduce $D(\boldsymbol{\beta})$ while leaving $\mathcal{P}_q(\boldsymbol{\beta})$ unchanged and $\hat{\boldsymbol{\beta}}_\lambda$ could not be the optimum.

Penalties, zero coefficients, uniqueness and convexity

- Here are some penalized solutions (red dots) for $q = 2$, 1 and 0.45. $D(\boldsymbol{\beta})$ contours in grey, solution contour for $\mathcal{P}_q(\boldsymbol{\beta})$ in red.



- As q is reduced the $\mathcal{P}$ contour develops corners at $q = 1$ and then ceases to be convex, leading to the possibility of developing multiple local minima for $q < 1$.
- Once the $\mathcal{P}$ contours have corners $\hat{\beta}_i = 0$ becomes possible.
- Unique solution if contours of D and $\mathcal{P}$ are convex and unique.

Sparsity

- The L_2 penalty shrinks the $\hat{\beta}_i$ towards zero, and will typically shrink most those coefficients for which the data provide little evidence that $\beta_i \neq 0$.
- But the penalty does not shrink any $\hat{\beta}_i$ exactly to zero.
- Often we would like to penalize coefficients to exactly zero, so that the corresponding effect is dropped from the model.
- For large p we would like *sparse* solutions with many $\hat{\beta}_i = 0$.
- So the 'L_0' penalty penalizing the number of non-zero $\hat{\beta}_i$ might seem ideal, but this leads to a very difficult optimization problem.
- Indeed multiple optima plague optimization for all penalties with $q < 1$, as seen on the previous slide.
- But $q = 1$ is special and unique: convex contours, with the corners that make $\hat{\beta}_i = 0$ (sparsity) possible.

Computing with the L_1 penalty

- Optimization of non-differentiable objective functions such as

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \mathcal{P}_q(\boldsymbol{\beta}) \text{ where } q \leq 1$$

  is difficult, but again $q = 1$ (known as the 'Lasso') is special.
- For $q = 1$, $\hat{\boldsymbol{\beta}}_\lambda$ can be obtained *simultaneously for all relevant* λ at the $\min\{O(np^2), O(n^3)\}$ cost of a single least squares fit.
- Note that a tempting approach is to use the form

$$\hat{\boldsymbol{\beta}}_\lambda = \text{argmin}_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 \text{ s.t.} \sum_{i=1}^{p} |\beta_i| \leq c$$

  a *quadratic programming problem* with constraint $\mathbf{C}\boldsymbol{\beta} \leq \mathbf{1}c$. $\mathbf{C}$ is the $2^p \times p$ matrix whose rows are all combinations of 1 and -1.
- But 2^p can be large + it costs more than regression, for a single c.

Piecewise linearity of the Lasso path

- Write the Lasso objective as $L = \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2/2 + \lambda \sum_{i=1}^{p} |\beta_i|$
- Consider a λ_0 at which $\hat{\beta}_i = 0$ for all i except for $i \in A$, the *active set*. We are interested in *decreasing* λ from λ_0.
- To optimize L, differentiate w.r.t. the active set variables, $\boldsymbol{\beta}_A$

$$\mathbf{X}_A^\intercal(\mathbf{X}_A\hat{\boldsymbol{\beta}}_A - \mathbf{y}) + \lambda \text{sign}(\hat{\boldsymbol{\beta}}_A) = \mathbf{0}$$

- Re-arranging and defining vectors $\mathbf{a}$ and $\mathbf{b}$ we have

$$\hat{\boldsymbol{\beta}}_A = (\mathbf{X}_A^\intercal\mathbf{X}_A)^{-1}\mathbf{X}_A^\intercal\mathbf{y} - \lambda(\mathbf{X}_A^\intercal\mathbf{X}_A)^{-1}\text{sign}(\hat{\boldsymbol{\beta}}_A) = \mathbf{b} - \lambda\mathbf{a}$$

- i.e. until A changes $\hat{\boldsymbol{\beta}}_A$ is linear in λ.[§]

[§]Notice how for $\text{sign}(\boldsymbol{\beta}_A)$ to change some $\hat{\beta}_i$ must pass through zero, but at zero it has left the active set.

Lasso path active set additions

- For a variable, β_k, to enter the active set it must decrease L.
- If $\mathbf{X}_k$ is column k of $\mathbf{X}$, the gradient of L w.r.t. β_k becomes

$$\mathbf{X}_k^\mathsf{T}\mathbf{X}\boldsymbol{\beta} - \mathbf{X}_k^\mathsf{T}\mathbf{y} + \lambda \text{sign}(\beta_k)$$

  – must be negative if $\beta_k > 0$ and positive if $\beta_k < 0$ to decrease L.
- Hence if $|\mathbf{X}_k^\mathsf{T}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})| \leq \lambda$ then β_k stays inactive at zero[¶].
- So as λ decreases, A changes as soon as one of this system is true

$$\mathbf{X}_A^\mathsf{T}(\mathbf{y} - \mathbf{X}_A\hat{\boldsymbol{\beta}}_A) = \pm\lambda\mathbf{1}$$

- Substituting $\hat{\boldsymbol{\beta}}_A = \mathbf{b} - \lambda\mathbf{a}$, and defining $\mathbf{c}$ and $\mathbf{d}$ this becomes

$$\mathbf{X}_A^\mathsf{T}(\mathbf{y} - \mathbf{X}_A\mathbf{b}) + \lambda(\mathbf{X}_A^\mathsf{T}\mathbf{X}_A\mathbf{a} \pm 1) = \mathbf{0} \text{ or } \mathbf{c} + \lambda(\mathbf{d} \pm 1) = \mathbf{0}$$

- i.e. next addition is at $\lambda^+ = \max_{<\lambda_0}\{-c_i/(d_i \pm 1)\}$.

[¶]but when a β_k *is* added, its sign is that of $\mathbf{X}_k^\mathsf{T}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})$.

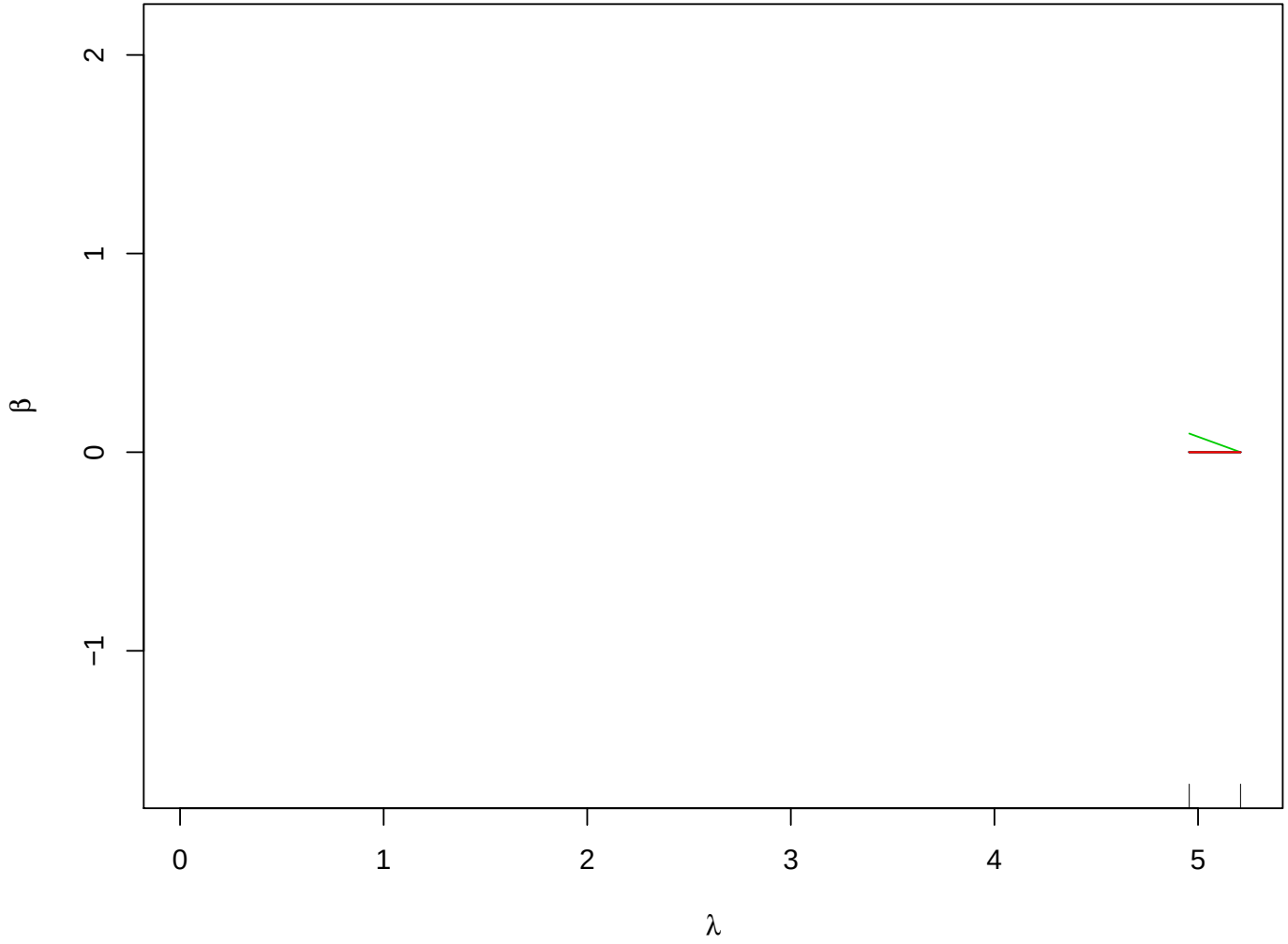
Lasso path deletions and algorithm

- Alternatively as λ is decreased the next change could be an active $\hat{\beta}_k$ hitting zero. Since $\hat{\boldsymbol{\beta}}_A = \mathbf{b} - \lambda \mathbf{a}$ this occurs the first time that one of the system $\mathbf{b} - \lambda \mathbf{a} = \mathbf{0}$ is true.
- i.e. the next deletion (if any) occurs at $\lambda^- = \max_{<\lambda_0}(b_i/a_i)$[‖].
- So as λ decreases the next A change depends on which of λ^+ or λ^- is larger.
- Defining $\mathbf{s} = \text{sign}(\boldsymbol{\beta})$, the algorithm is:
  1. Find highest λ at which first variable is active, initializing A and $\mathbf{s}$, then repeat...
  2. Find next λ (downwards) at which A changes, store it and corresponding $\hat{\boldsymbol{\beta}}$ and update A and $\mathbf{s}$.
- $(\mathbf{X}_A^\mathsf{T}\mathbf{X}_A)^{-1}$ looks costly, but by cheaply updating the Cholesky factor of $\mathbf{X}_A^\mathsf{T}\mathbf{X}_A$ on each A change, the cost is only $O(p_A^2)$ per step.

[‖] In finite precision computational practice it is necessary to exclude entries that were added to A at the previous step here

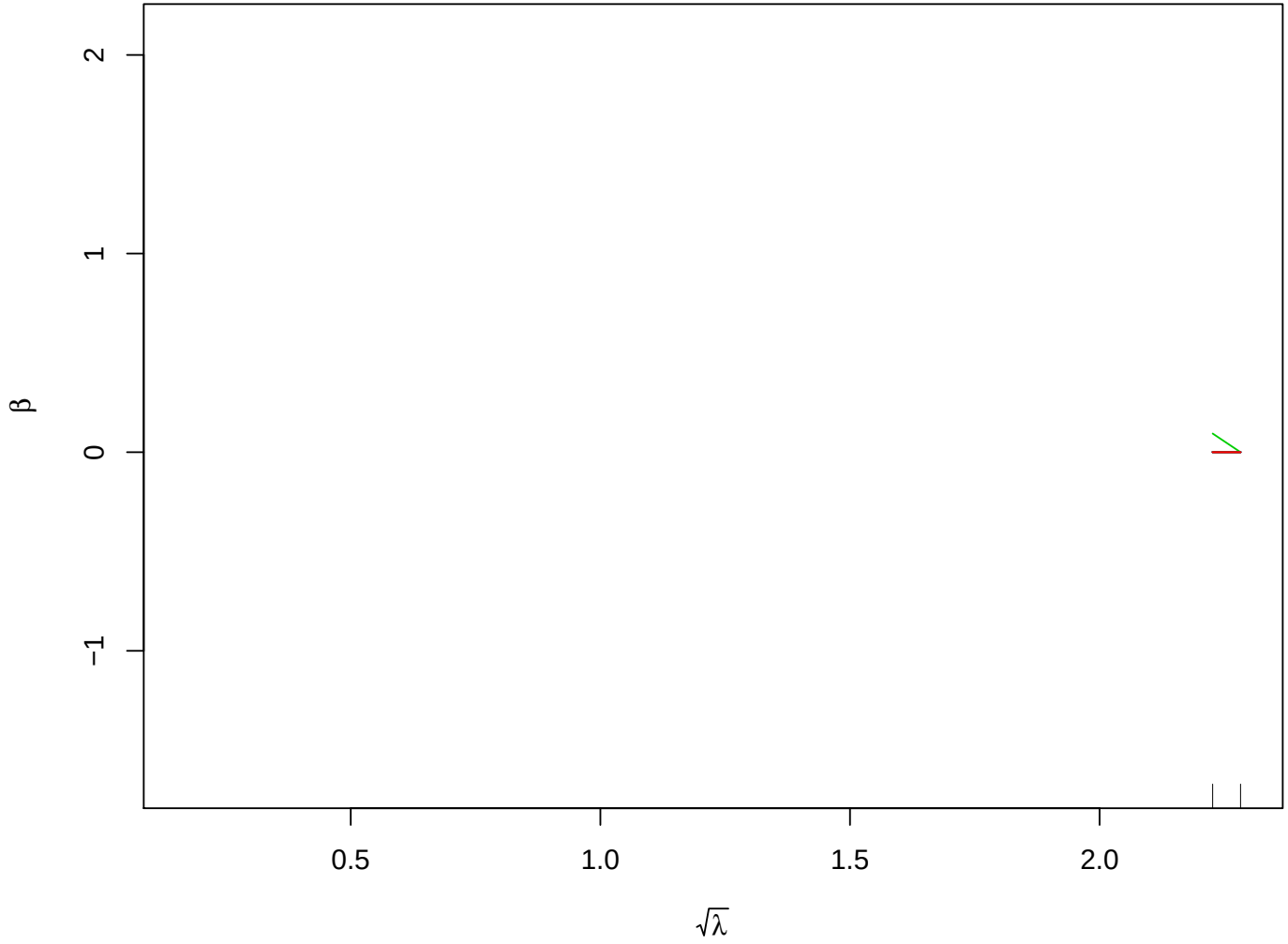
The Lasso path

- It's easy to code the algorithm in R - here it is applied to the earlier simulated example. . .



Plotting the Lasso path

- The path is only piecewise linear in λ, but is often plotted as piecewise linear against transformations of λ for convenience. . .

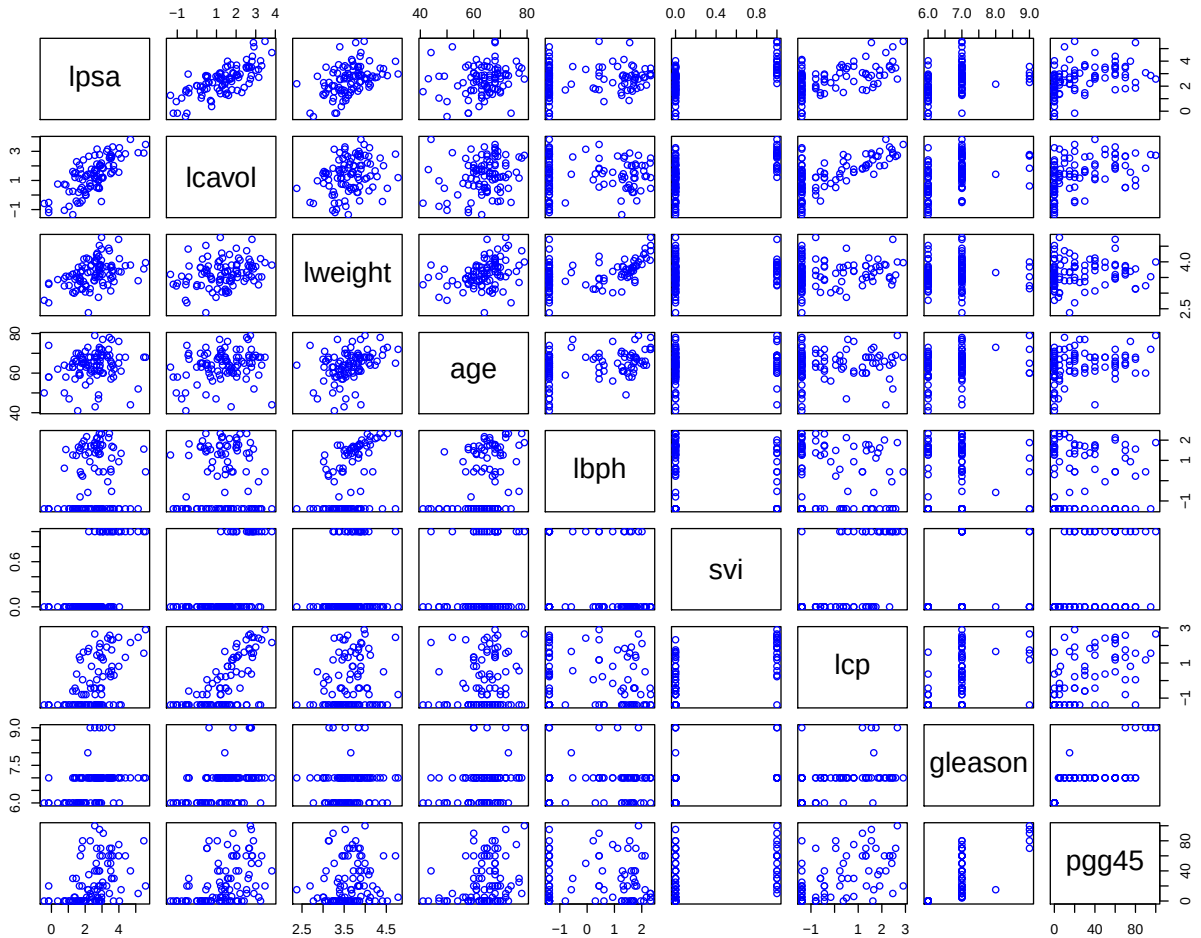


Lasso practicalities

- Since real covariates (columns of $\mathbf{X}$) may not be on similar scales, it is usual to standardize them to have zero mean and constant (e.g. unit) variance.
- Whether this *really* makes the coefficients comparable in a way that justifies adding their magnitudes in the penalty is unclear!
- Any model intercept is certainly different in kind (e.g. its dummy covariate has 0 variance), so rather than include an intercept, it is usual to simply subtract its mean from y.
- To select λ we can divide the data into fit and test sets and optimize prediction error on the test set.
- Note that when $p \gg n$ the $O(n^3)$ computational cost of Lasso is much cheaper than the $O(np^2)$ cost of ridge regression.

A practical Lasso example

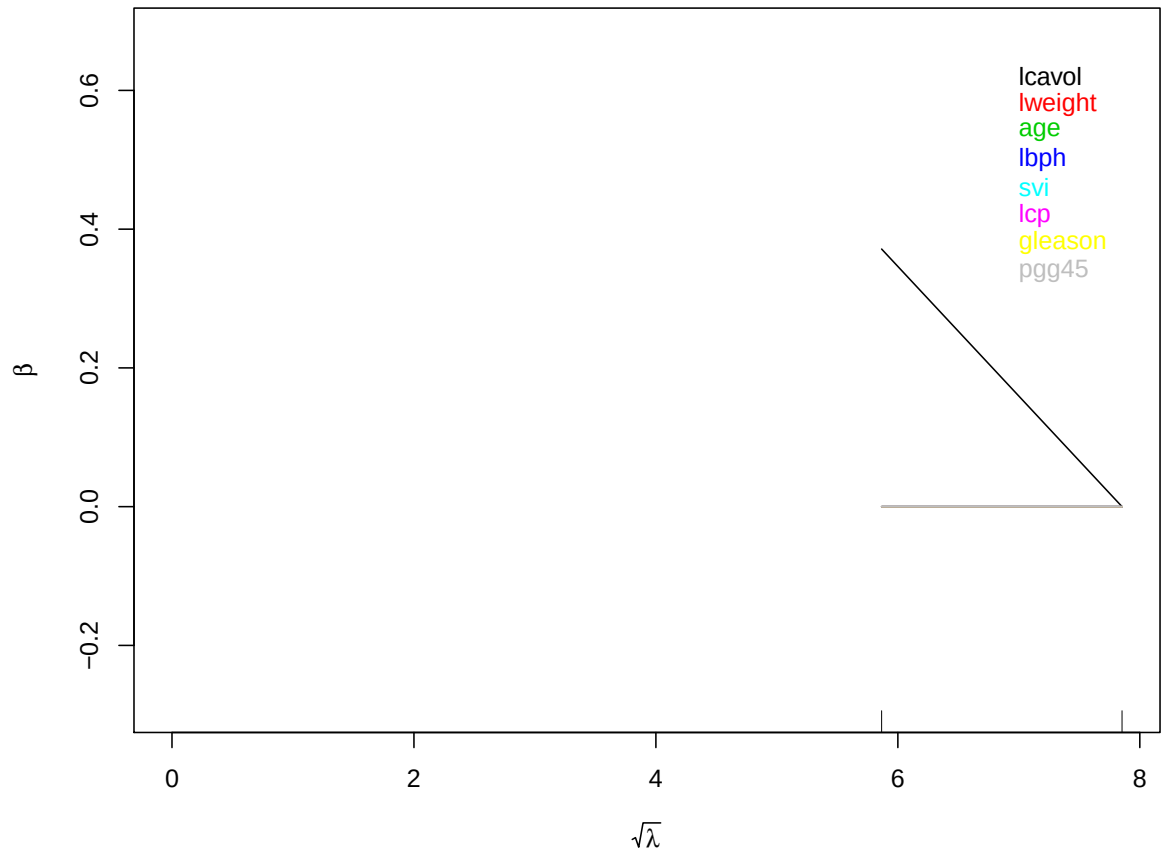
- Hastie et al.** provide an example of predicting log Prostate Specific Antigen from 8 predictors, based on 97 patients' data.



**Hastie, Tibshirani and Friedman (2009) *Elements of Statistical Learning*
https://web.stanford.edu/~hastie/ElemStatLearn/

Prostate Lasso path

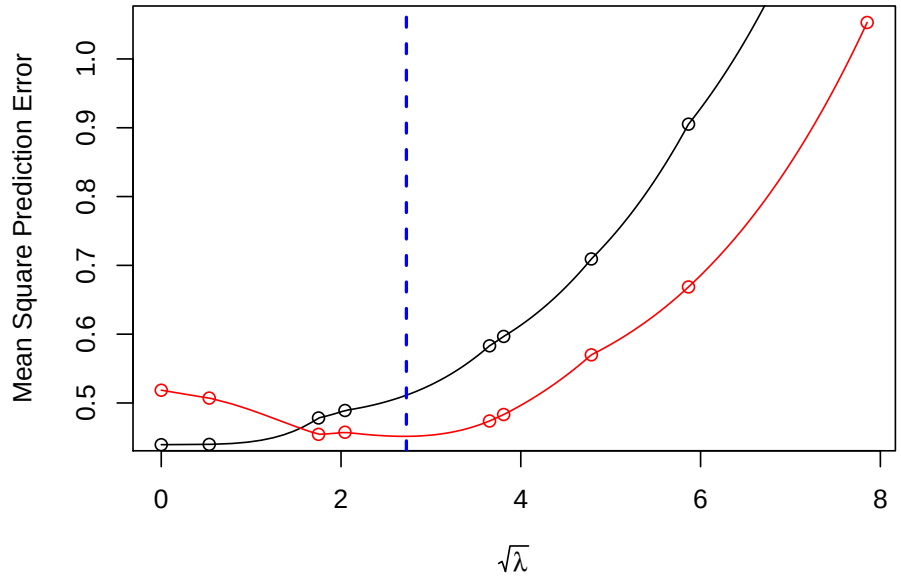
- Having standardized the variables[††], let's follow Hastie et al. and use the first 67 patients for fit data. The Lasso path is here...



[††] See `?scale` in R.

Prostate Lasso path - choosing λ

- λ can be chosen to minimise the mean squared error in predicting `lpsa` using the test set.



. . . red is for test, black for fit, blue is at minimum test error.

- Note that the predicted values are piecewise linear in λ, making computation cheap and easy.

Lasso with other loss functions

- The preceding Lasso path algorithm is elegant and exact, but does not generalize beyond least squares/quadratic loss.
- For example suppose that we want to model binary data using the negative log likelihood as the loss.
- If we don't need the exact Lasso path for all λ, but are happy to simply evaluate at a finite number of λ values then a very efficient and simple *coordinate descent* algorithm is effective.
- Coordinate descent updates $\hat{\boldsymbol{\beta}}$ iteratively, updating each $\hat{\beta}_i$ in turn while holding the the rest of $\hat{\boldsymbol{\beta}}$ constant, to improve the fitting objective (or possibly leave if unchanged if $\hat{\beta}_i = 0$).
- Since each update is one dimensional they are very cheap, which offsets the fact that many steps may be needed.

Lasso coordinate descent

- Let $D(\boldsymbol{\beta})$ be a general Loss (preferably with convex contours), e.g. a negative log likelihood for binary data. β_0 is an intercept.
- The Lasso objective becomes $D(\boldsymbol{\beta}) + \lambda \sum_{i=1}^{p} |\beta_i|$.
- As before, for β_i to move from zero we require

$$\left.\frac{\partial D}{\partial \beta_i}\right|_{\beta_i=0} + \lambda \text{sign}(\beta_i)$$

  to be positive for $\beta_i < 0$ and negative for $\beta_i > 0$.
- Therefore, for $i > 0$, if $|\, \partial D/\partial \beta_i|_0 \,| \leq \lambda$ then $\hat{\beta}_i = 0$ and otherwise $\text{sign}(\hat{\beta}_i) = -\text{sign}(\partial D/\partial \beta_i|_0)$
- If $|\, \partial D/\partial \beta_i|_0 \,| > \lambda$ we can update $\hat{\beta}_i$ by taking the Newton step

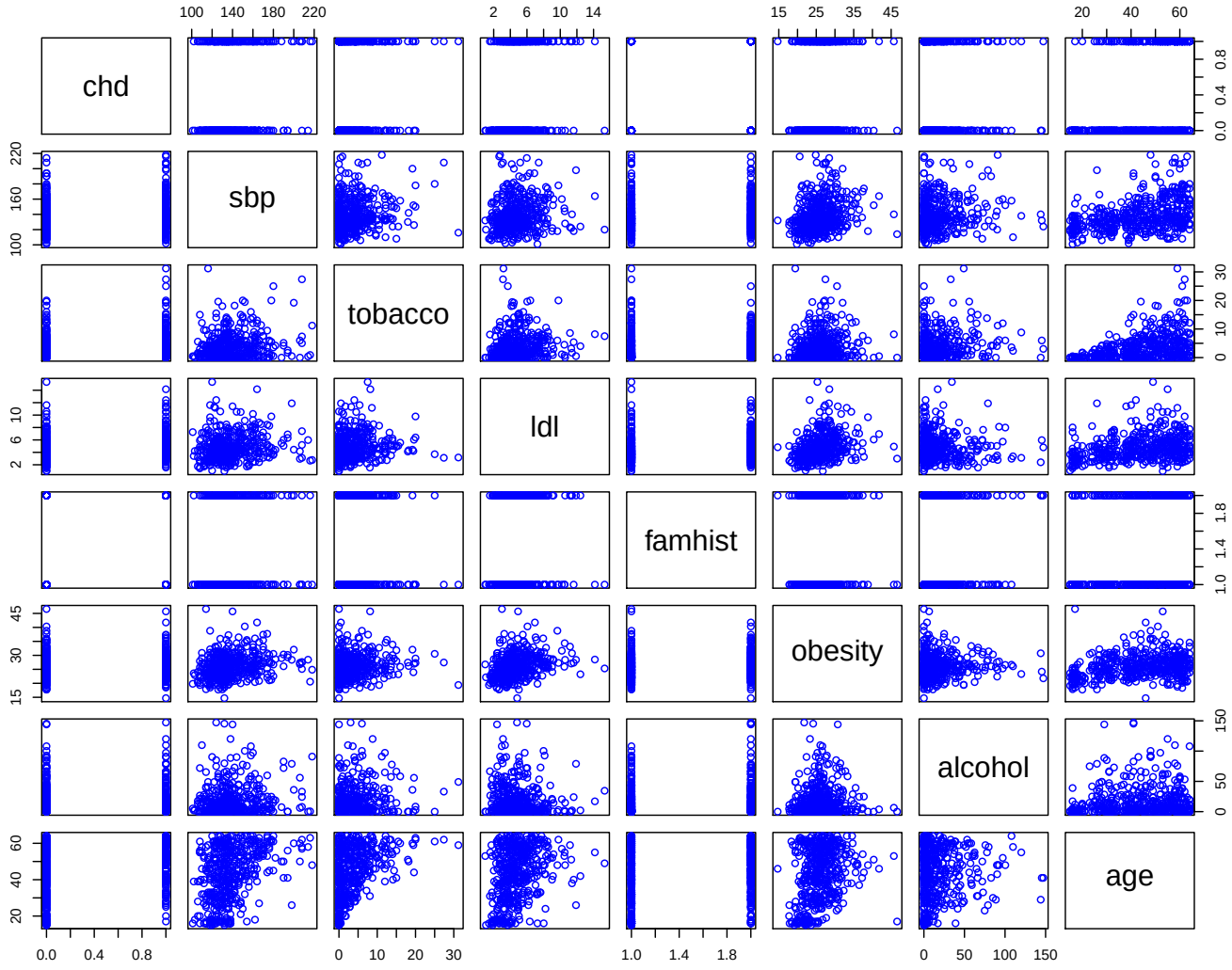
$$\Delta\beta_i = -\left(\frac{\partial^2 D}{\partial \beta_i^2}\right)^{-1}\left(\frac{\partial D}{\partial \beta_i} + \lambda \text{sign}(\beta_i)\right)$$

Lasso coordinate descent practicalities

- β_0 is never constrained to zero, or penalized (computationally it is as if we set $\text{sign}(\beta_0) = 0$).
- All preceding derivatives are evaluated at the current $\hat{\boldsymbol{\beta}}$, except where stated that β_i is set to zero.
- To guarantee convergence we must check that the step from $\hat{\beta}_i$ to $\hat{\beta}_i + \Delta\beta_i$ decreased the penalized loss, halving $\Delta\beta_i$ until it does.
- Let $\hat{\boldsymbol{\eta}} = \mathbf{X}\hat{\boldsymbol{\beta}}$. Note that each β_i update requires an $O(n)$ update $\hat{\boldsymbol{\eta}} \rightarrow \hat{\boldsymbol{\eta}} + \mathbf{X}[, i]\Delta\beta_i$, rather than an $O(np)$ formation of $\mathbf{X}\hat{\boldsymbol{\beta}}$.
- Hence each cycle through the coefficient updates has $O(np)$ cost, and it typically takes a few cycles to converge. Contrast this to standard logistic regression, which takes several $O(np^2)$ steps.

Binary Lasso example

- Here are data from another Hastie et al example, on predicting coronary heart disease in a South African case control study…



Binary Lasso example

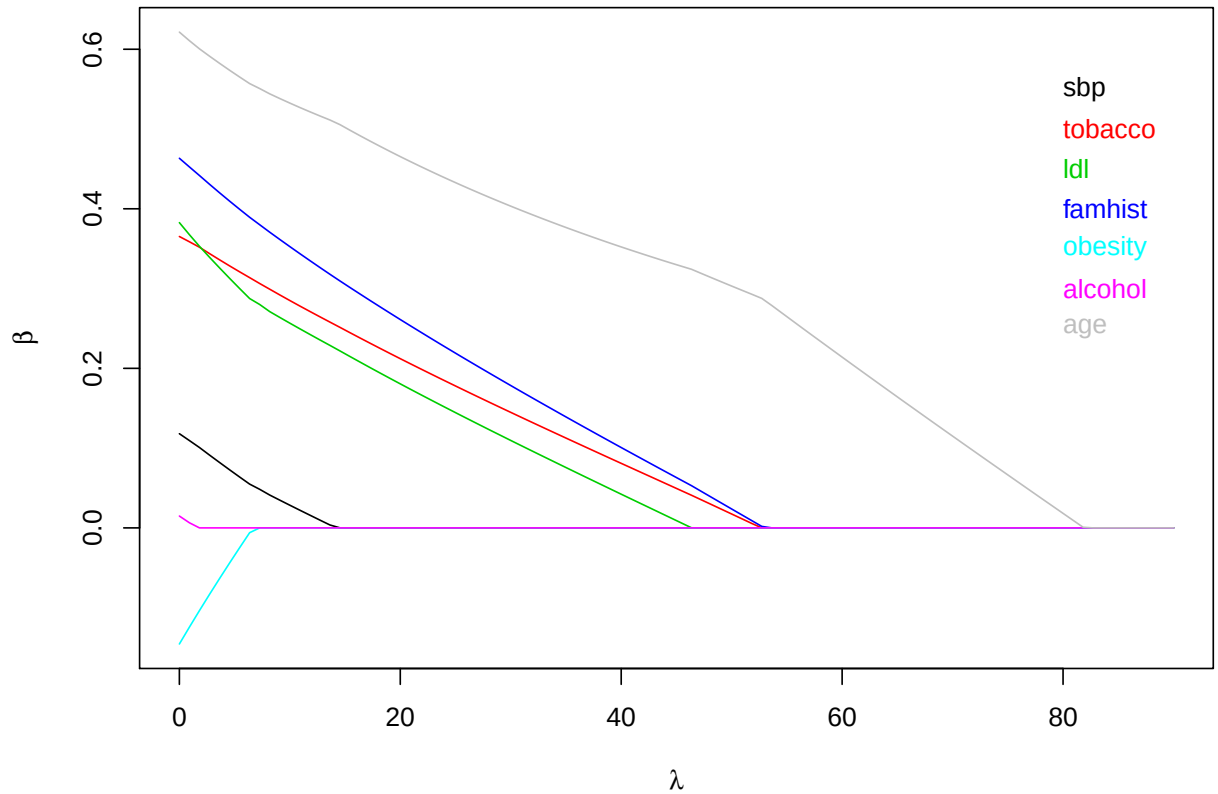
- Again the predictors are centred and standardized to unit variance (`sdl` is systolic blood pressure and `ldl` cholesterol measurement)
- It makes no sense to center a binary response, `chd`, and anyway the intercept, β_0, removes any need to.
- The negative log likelihood (to within an additive constant) is

$$D = \sum_i \log(1 + e_i^{\eta}) - y_i \eta_i$$

  where $\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}$ ($\mathbf{X}$ augmented with a unit 'intercept' column).
- The coordinate descent algorithm is easily programmed in R following the preceding recipe.

Binary Lasso heart disease results

- Here is the Lasso path, evaluated at 100 evenly space λ values.



λ choice and uncertainty

- Statistical properties of the Lasso are less straightforward than L_2 penalized ridge regression, in part because of the very sparsity inducing properties that make L_1 penalization attractive.
- A simple approach to investigating estimator uncertainty is to non-parametrically bootstrap. That is to simulate the process of repeatedly sampling replicate data sets and re-estimating the model, by *re-sampling* the data in the original dataset.
- Specifically we re-sample n rows of data from the original (n row) dataset, with replacement.
- Model estimation then proceeds for each replicate, with the resulting variability in estimates approximating the sampling variability in $\hat{\boldsymbol{\beta}}$.
- We can select λ to minimise the average loss when predicting the data not included in each bootstrap sample, from the model fitted to that sample.

Practical Lasso bootstrap

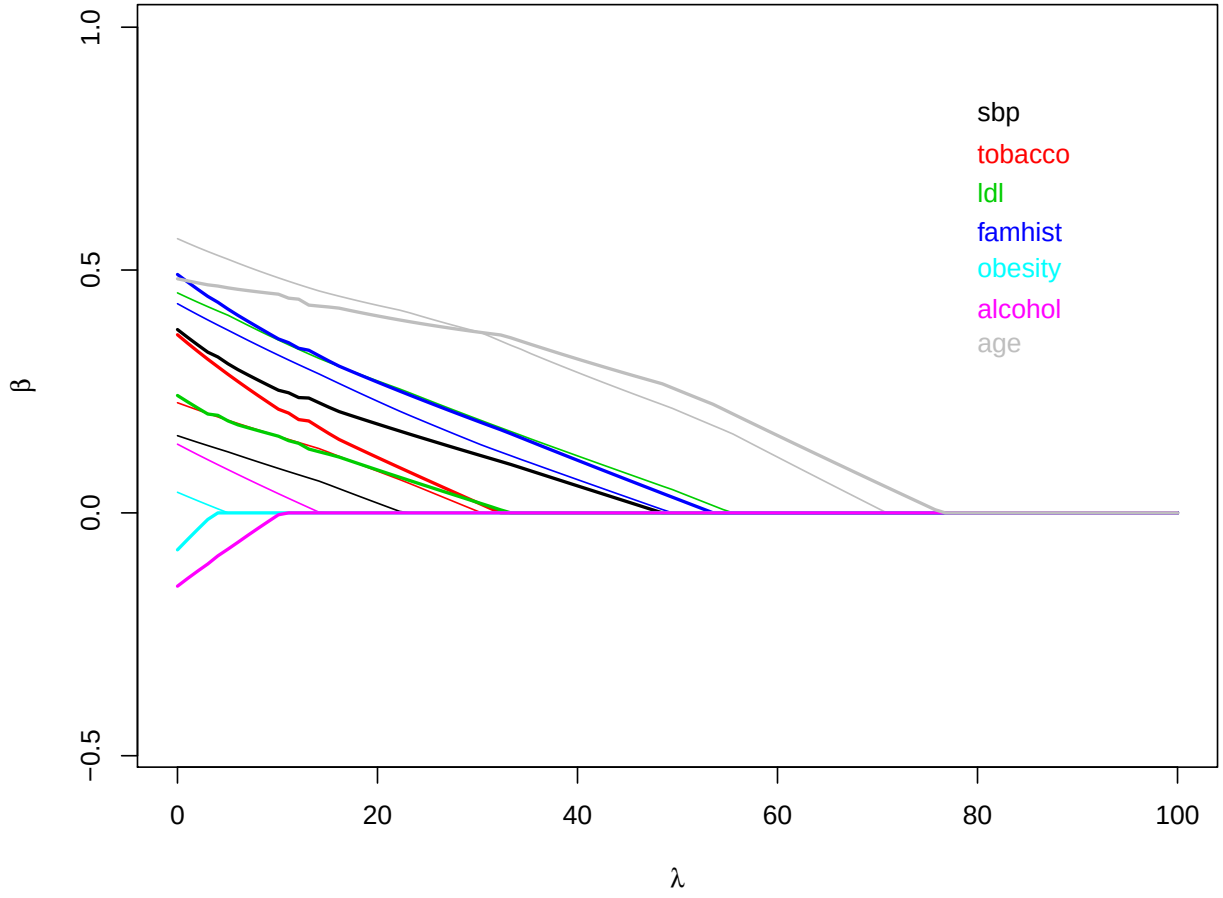
- If `lagd` is a function for binary Lasso fitting, and `lob` evaluates the corresponding loss, the following R loop performs the bootstrapping, given appropriate arrays, `lam`, `bs` and `lcv`

```
for (b in 1:nb) { ## BS loop
  ii <- sample(1:n,n,replace=TRUE) ## BS indices
  Xb <- X[ii,];yb <- y[ii]   ## BS fit data
  Xv <- X[-ii,];yv <- y[-ii] ## BS test data
  for (i in 1:m) { ## loop over lambda values
    fit <- lagd(yb,Xb,lam[i]) ## Lasso fit
    bs[b,,i] <- fit$beta ## store beta
    lcv[i] <- lcv[i] + lob(fit$beta, beta,yv,Xv)/nb
  }
}
```

- It's now easy to plot the cross validated loss, `lcv`, against the λ values in `lam`, and to use the replicated $\hat{\boldsymbol{\beta}}$ vectors in `bs` to examine estimator uncertainty.

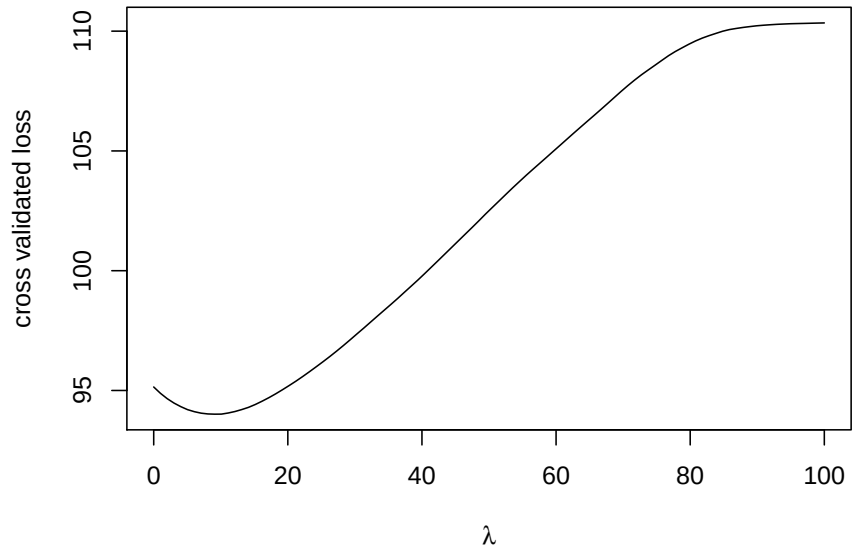
Lasso bootstrap path uncertainty

- The path and parameter estimate variability is quite high for the heart disease data. . .



λ choice etc.

- Here is the plot of bootstrap cross validated loss against λ, suggesting $\hat{\lambda} \simeq 9$.



- With the corresponding standard deviations and medians for $\hat{\boldsymbol{\beta}}$

```
> diag(vcov(glm(y~X,family=binomial())))^.5 ## unpenalized logistic regression
(Intercept)    Xsbp  Xtobacco      Xldl  Xfamhist  Xobesity  Xalcohol     Xage
  0.12013   0.11545   0.12041   0.11890   0.11094   0.12264   0.10906   0.1486
> apply(bs[,,10],2,sd)
  0.12953   0.07955   0.12594   0.12984   0.09182   0.06110   0.02294   0.1212
> apply(bs[,,10],2,median)
 -0.80316   0.08956   0.26862   0.27685   0.36663   0.00000   0.00000   0.5441
> coef(glm(y~X,family=binomial())) ## unpenalized logistic regression
 -0.8453    0.1181    0.3653    0.3827    0.4634   -0.1456    0.0148    0.6215
```