

# Boosting

(cu. 10 of text)

Goal: maintain most advantages of trees while dramatically improving accuracy.

## Predictive learning

training data :  $\{y_i, \underline{z}_i\}_{i=1}^N$

$y_i$  = outcome variable

$\underline{z}_i$  = joint values of predictors

(note: called  $\underline{x}_i$  before)

$$\hat{F}(\underline{z}) = \operatorname{argmin}_{F(\underline{z}) \in \mathcal{F}} \sum_{i=1}^N \ell(y_i, F(\underline{z}_i))$$

function class  $\uparrow$

score criterion

Trees:  $\mathcal{F}$  = piecewise constant functions

$$F(\underline{z}) = T(\underline{z}) = \sum_{m=1}^M c_m I(\underline{z} \in R_m)$$

$\{R_m\}_{m=1}^M$  = disjoint regions of  $\underline{z}$ -space

Basic idea of boosting:

$\mathcal{F}$ (boost) = linear combination of all trees

$$F(\underline{z}) = \sum_{j=1}^J a_j T_j(\underline{z})$$

$\uparrow$  decision tree  
 $\uparrow$  coefficient ("weight")

$J \leq \infty$  (all possible trees - fixed)

Population:  $F^*(\underline{z}) = F(\underline{z}; \{a_j^*\}_{j=1}^J)$

$$\{a_j^*\}_{j=1}^J = \operatorname{argmin}_{\{a_j\}_{j=1}^J} E_{y \times \underline{z}} L(y, \sum_{j=1}^J a_j T_j(\underline{z}))$$

Training data:

$$\hat{F}(\underline{z}) = \operatorname{argmin}_{\{a_j\}_{j=1}^J} \frac{1}{N} \sum_{i=1}^N L(y_i, \sum_{j=1}^J a_j T_j(\underline{z}_i))$$

linear regression where predictors are all possible trees ( $J \leq \infty$ )

Let  $x_j = T_j(z)$  &  $x_{ij} = T_j(z_i)$

$$F(x; \{a_j^*\}_1^J) = \sum_{j=1}^m a_j^* x_j$$

$$\{\hat{a}_j\}_1^m = \underset{\{a_j\}_1^J}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^N L(y_i, \sum_{j=1}^m a_j x_{ij})$$

$$m = J \approx \infty$$

Problem : number of parameters  $m \approx \infty$

>> training sample size

=>  $\infty$  number of solutions,  
almost all bad

What to do?

regularized linear regression

Text : ch 10.12

<http://www-stat.stanford.edu/~jhf/ftp/GPSpub.pdf>

Training data:  $\{y_i, \mathbf{x}_i\}_1^N \sim p(\mathbf{x}, y)$

$$\hat{R}(\mathbf{a}) = \frac{1}{N} \sum_{i=1}^N L \left( y_i, a_0 + \sum_{j=1}^n a_j x_{ij} \right)$$

$$\hat{\mathbf{a}} = \arg \min_{\mathbf{a}} \hat{R}(\mathbf{a})$$

If not  $N \gg n$ , not very good!

$$R(\hat{\mathbf{a}}) \gg R(\mathbf{a}^*) \quad (\text{high variance})$$

## REGULARIZATION (biased learning)

$$\hat{\mathbf{a}}(t) = \arg \min_{\mathbf{a}} \hat{R}(\mathbf{a}) \quad \text{s.t.} \quad P(\mathbf{a}) \leq t$$

$P(\mathbf{a}) \geq 0$  constraining function

$t \geq P(\hat{\mathbf{a}})$ : no constraint  $\Rightarrow$  no bias / max. variance

$t = 0$ : max. constraint  $\Rightarrow$  max. bias / min. variance

$0 < t < P(\hat{\mathbf{a}}) \Rightarrow$  bias–variance trade–off

## EQUIVALENT FORMULATION

$$\hat{\mathbf{a}}(\lambda) = \arg \min_{\mathbf{a}} [\hat{R}(\mathbf{a}) + \lambda \cdot P(\mathbf{a})]$$

Here  $P(\mathbf{a}) =$  “penalty”

$$\infty \leq \lambda \leq 0 \sim 0 \leq t \leq P(\hat{\mathbf{a}})$$

$\hat{\mathbf{a}}(\lambda) \sim$  1-dim. path of solutions  $\in S^{n+1}$

$S^{n+1} =$  parameter space

## MODEL SELECTION ( $\lambda$ )

$$\lambda^* = \arg \min_{0 \leq \lambda \leq \infty} R(\hat{\mathbf{a}}(\lambda))$$

Model selection criterion:

$$\tilde{R}(\mathbf{a}) = \text{surrogate for } R(\mathbf{a})$$

depends on  $L(y, F)$  &  $P(\mathbf{a})$

$$\hat{\lambda} = \arg \min_{0 \leq \lambda \leq \infty} \tilde{R}(\hat{\mathbf{a}}(\lambda))$$

$$\hat{\mathbf{a}}(\hat{\lambda}) = \text{selected model}$$

Cross-validation: any  $L(y, F)$  &  $P(\mathbf{a})$

## PENALTY SELECTION

$\mathbf{a}^*$  = point in  $S^{n+1}$

Choose penalty that induces paths that

on average come close to  $\mathbf{a}^*$

$$\{y_i, \mathbf{x}_i\}_1^N \sim p(\mathbf{x}, y)$$

Depends on  $\mathbf{a}^*$

Choose  $P(\mathbf{a})$  based on knowledge of  $\mathbf{a}^*$

## SPARSITY

Fraction of non influential variables

$$S(\mathbf{a}) = \#(|a_k| = \text{small})/n$$

Assumption:  $\hat{\mathbf{a}} \simeq \mathbf{a}^* \Rightarrow S(\hat{\mathbf{a}}) \simeq S(\mathbf{a}^*)$

Choose  $P(\mathbf{a})$  s.t.  $S(\hat{\mathbf{a}}(\lambda^*)) \simeq S(\mathbf{a}^*)$

Don't know  $S(\mathbf{a}^*)$  ?

Family of penalties  $P_\gamma(\mathbf{a})$  :  $\gamma$  regulates  $S(\hat{\mathbf{a}})$

bridging sparse  $\rightarrow$  dense solutions

Model selection to jointly estimate  $(\gamma, \lambda)$

## POWER FAMILY

$$P_\gamma(\mathbf{a}) = \sum_{j=1}^n |a_j|^\gamma$$

With  $L(y, F) = (y - F)^2$ :

$\gamma = 2$  : ridge-regression (dense)

$\gamma = 1$  : lasso (moderately sparse)

$\gamma = 0$  : (all) subsets selection (sparsest)

$0 \leq \gamma \leq 2$  bridges subset  $\rightarrow$  ridge

Note:  $\gamma \geq 1 \Rightarrow$  convex,  $\gamma < 1 \Rightarrow$  non convex

## Generalized Elastic Net

$1 \leq \beta \leq 2$  (convex: lasso  $\rightarrow$  ridge):

Elastic Net (Zou & Hastie 2005)

$$P_{\beta}(\mathbf{a}) = \sum_{j=1}^n (\beta - 1) a_j^2 / 2 + (2 - \beta) |a_j|$$

$0 \leq \beta < 1$  (non convex: subset selection  $\rightarrow$  lasso):

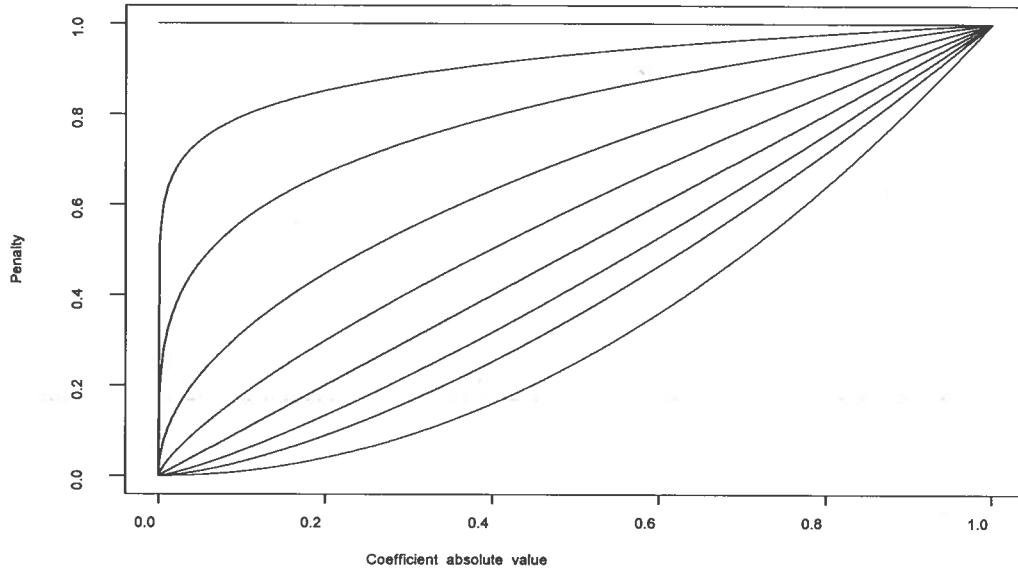
$$P_{\beta}(\mathbf{a}) = \sum_{j=1}^n \log((1 - \beta) |a_j| + \beta)$$

$0 \leq \beta \leq 2$  bridges subset  $\rightarrow$  ridge

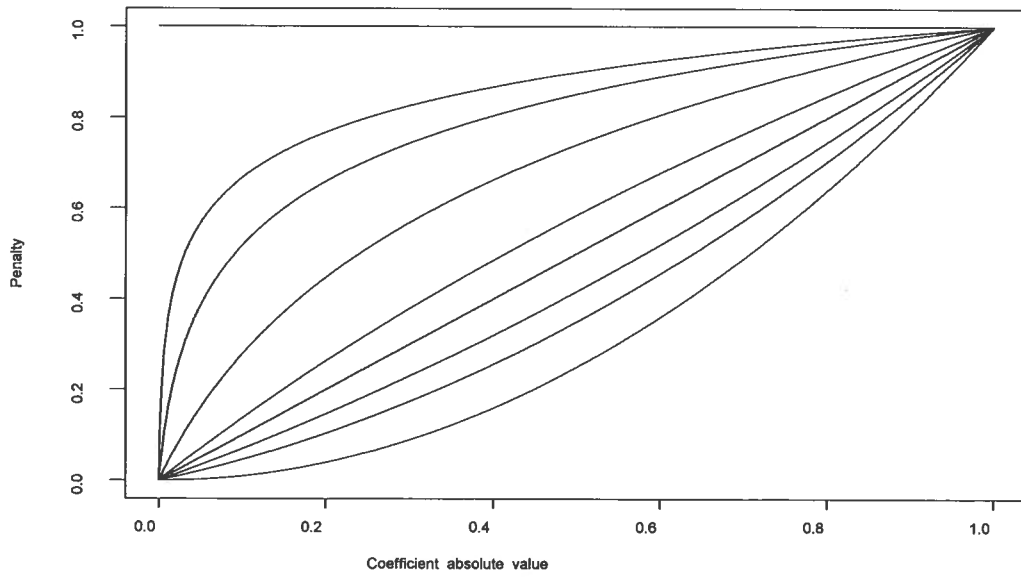
Better statistical & computational properties

Method works for both + many more

**Power family**



**Generalized elastic net**



## BRIDGE REGRESSION

(1) Repeatedly solve:

$$\hat{\mathbf{a}}_{\beta}(\lambda) = \arg \min_{\mathbf{a}} [\hat{R}(\mathbf{a}) + \lambda \cdot P_{\beta}(\mathbf{a})]$$

$$0 \leq \beta \leq 2, \quad 0 \leq \lambda \leq \infty$$

(2)  $(\hat{\beta}, \hat{\lambda}) \leftarrow$  model selection criterion

(3)  $\hat{\mathbf{a}}_{\hat{\beta}}(\hat{\lambda}) =$  solution

Big challenge: fast enough algorithm for (1)

Especially for  $P_{\beta}(\mathbf{a}) =$  non convex

## DIRECT PATH SEEKING

Goal: rapidly produce path  $\simeq$  given  $P(\mathbf{a})$

without repeatedly optimizing

$\nu \geq 0$ : path length;  $\Delta\nu > 0$ : increment

$\mathbf{d}(\nu)$  = direction in parameter space

## Algorithm

Initialize:  $\nu = 0$ ;  $\hat{\mathbf{a}}(0) = 0$

Loop {

$$\hat{\mathbf{a}}(\nu + \Delta\nu) = \hat{\mathbf{a}}(\nu) + \mathbf{d}(\nu) \cdot \Delta\nu$$

$$\nu \leftarrow \nu + \Delta\nu$$

}

Until ( $\hat{R}(\hat{\mathbf{a}}(\nu)) = \min$ )

Methods differ:  $\mathbf{d}(\nu)$  &  $\Delta\nu$

## EXAMPLES

$$L(y, F) = (y - F)^2:$$

PLS  $\simeq$  ridge-regression ( $\beta = 2$ )

LAR  $\simeq$  lasso ( $\beta = 1$ )

Forward stepwise  $\simeq$  all-subsets ( $\beta = 0$ )

## Generalized Path Seeking (GPS)

Fast algorithm for:

(1) any convex  $L(y, F)$  (some non convex)

(2) any  $P(\mathbf{a})$  s.t.  $\frac{\partial P(\mathbf{a})}{\partial |a_j|} \geq 0$

i.e.  $P(\mathbf{a})$  monotone  $\uparrow |a_j|$

## Definitions

$\nu \geq 0$ : path length

$\Delta\nu > 0$ : small increment

$$g_j(\nu) = - \left[ \frac{\partial \hat{R}(\mathbf{a})}{\partial a_j} \right]_{\mathbf{a}=\hat{\mathbf{a}}(\nu)} \quad (\text{loss + data})$$

$$p_j(\nu) = \left[ \frac{\partial P(\mathbf{a})}{\partial |a_j|} \right]_{\mathbf{a}=\hat{\mathbf{a}}(\nu)} \quad (\text{penalty})$$

$$\lambda_j(\nu) = g_j(\nu) / p_j(\nu) \quad (\text{combination})$$

# Algorithm

- 1 Initialize:  $\nu = 0; \{\hat{a}_j(0) = 0\}_1^n$
- 2 Loop {
- 3    Compute  $\{\lambda_j(\nu)\}_1^n$
- 4     $S = \{j \mid \lambda_j(\nu) \cdot \hat{a}_j(\nu) < 0\}$
- 5    if ( $S = \text{empty}$ )  $j^* = \arg \max_j |\lambda_j(\nu)|$
- 6    else  $j^* = \arg \max_{j \in S} |\lambda_j(\nu)|$
- 7     $\hat{a}_{j^*}(\nu + \Delta\nu) = \hat{a}_{j^*}(\nu) + \Delta\nu \cdot \text{sign}(\lambda_{j^*}(\nu))$
- 8     $\{\hat{a}_j(\nu + \Delta\nu) = \hat{a}_j(\nu)\}_{j \neq j^*}$
- 9     $\nu \leftarrow \nu + \Delta\nu$
- 10 } Until  $\lambda(\nu) = 0$

## THEOREM

$\hat{\mathbf{a}}(\lambda) = \text{exact path}$

$\hat{\mathbf{a}}(\nu) = \text{GPS path}$

If for all  $\lambda > \lambda_0$

all  $\{\hat{a}_j(\lambda)\}_1^n$  are continuous and monotone

Then for all  $\lambda > \lambda_0$

$\hat{\mathbf{a}}(\nu) = \hat{\mathbf{a}}(\lambda)$ , as  $\Delta\nu \rightarrow 0$

i.e. GPS produces exact path

Otherwise:  $\hat{a}_j(\nu) \simeq \hat{a}_j(\lambda)$

When  $\hat{a}_j(\lambda)$  becomes non monotone:

$\hat{a}_j(\nu)$  tends to slightly delay becoming non monotone

When  $\hat{a}_j(\lambda)$  discontinuous ( $\gamma < 1, \beta < 1/2$ ):

$\hat{a}_j(\nu) =$  continuous (by construction)

$\sim$  interpolates between  $\hat{a}_j(\lambda)$  discontinuities

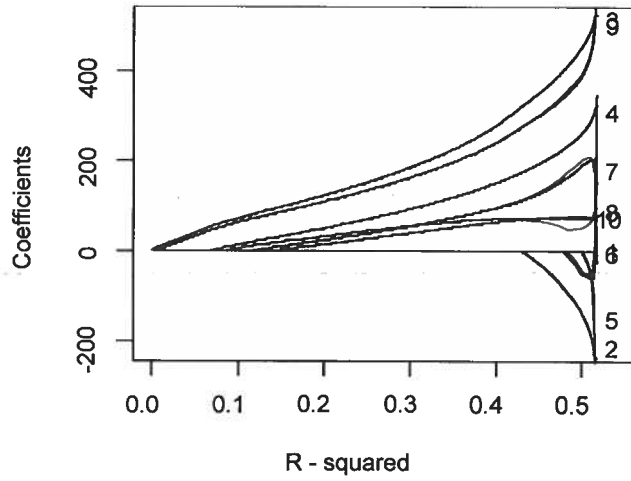
Regression:  $L(y, F) = (y - F)^2$

Diabetes data:  $n = 10$ ,  $N = 442$

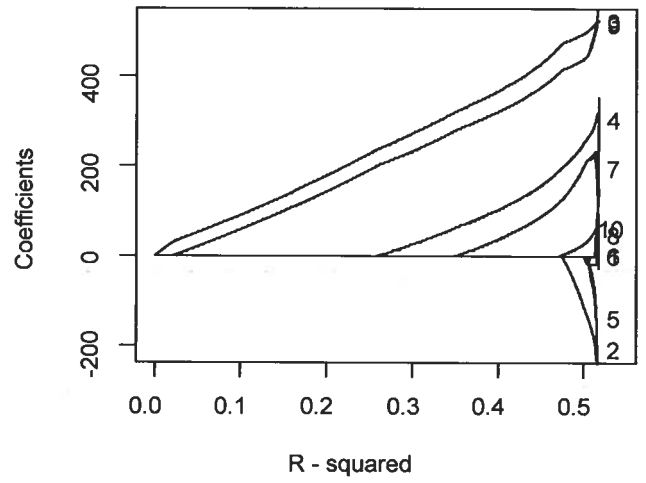
Used in LARS (Efron *et al* 2002)

red = exact (convex), black = GPS

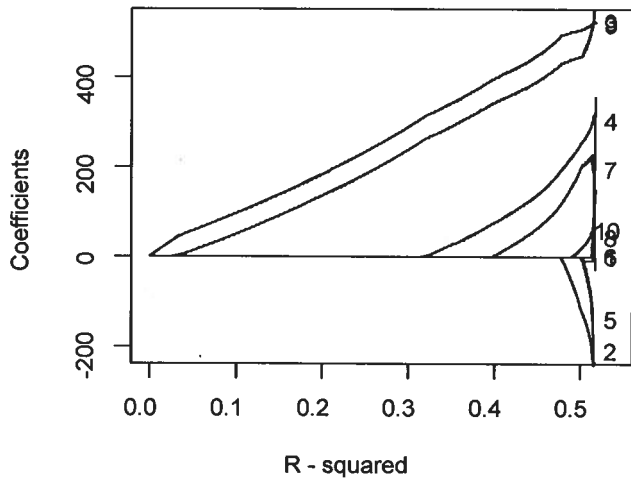
**Elastic Net 1.9**



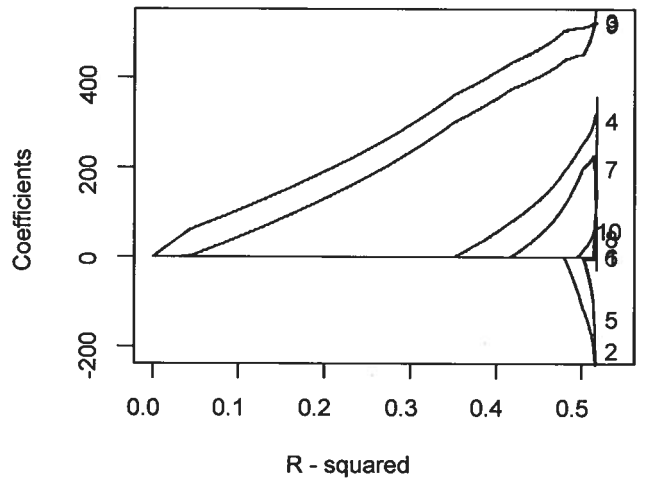
**Elastic Net 1.5**



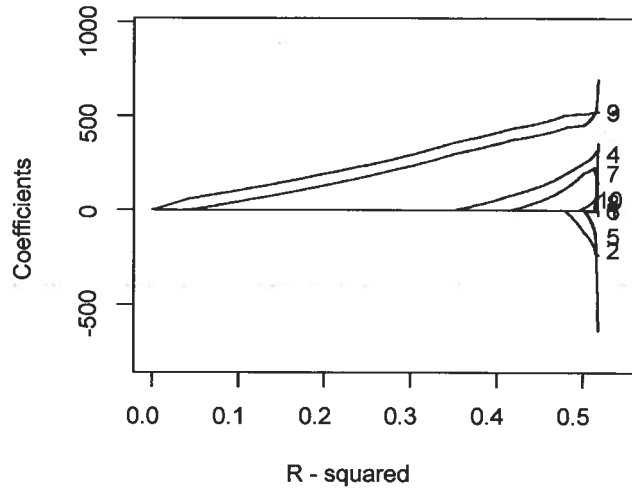
**Elastic Net 1.25**



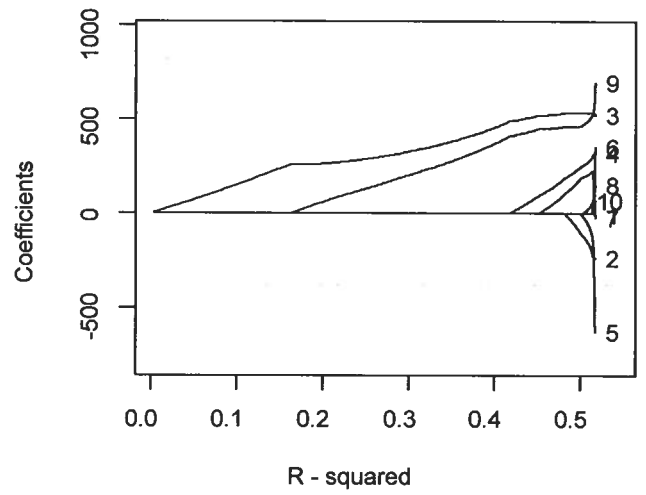
**Lasso**



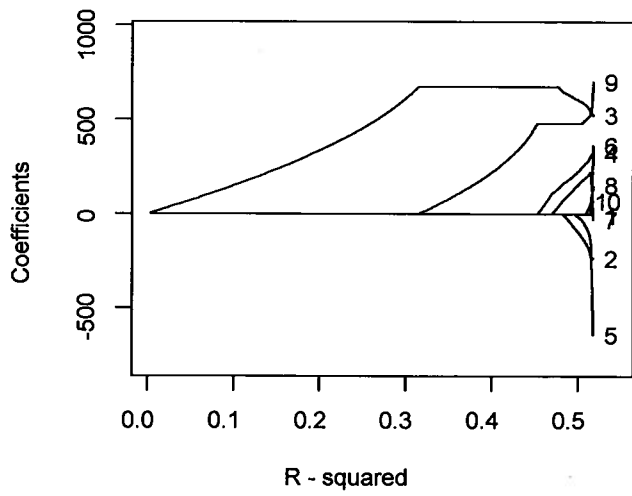
**Lasso**



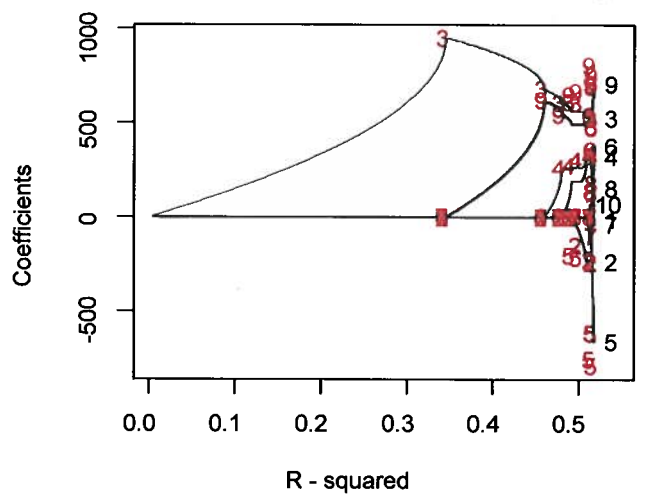
**Elastic Net 0.5**



**Elastic Net 0.25**



**Elastic Net 0.0**



## Logistic Regression / Classification

South African heart transplant data

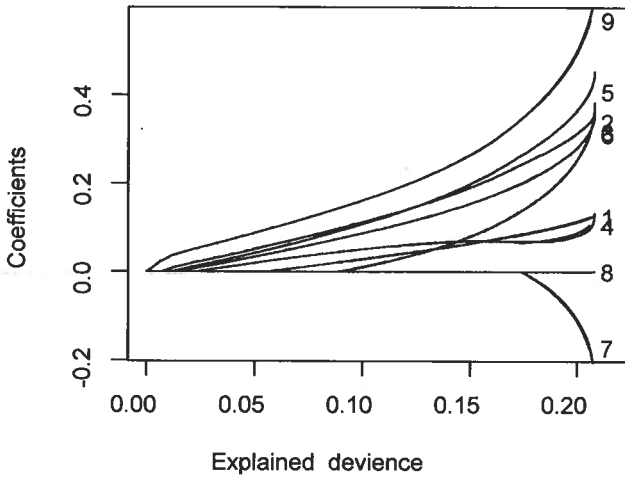
$$y \in \{1, -1\} = \{\text{success, failure}\}$$

$$L(y, F) = \log(1 + e^{-yF})$$

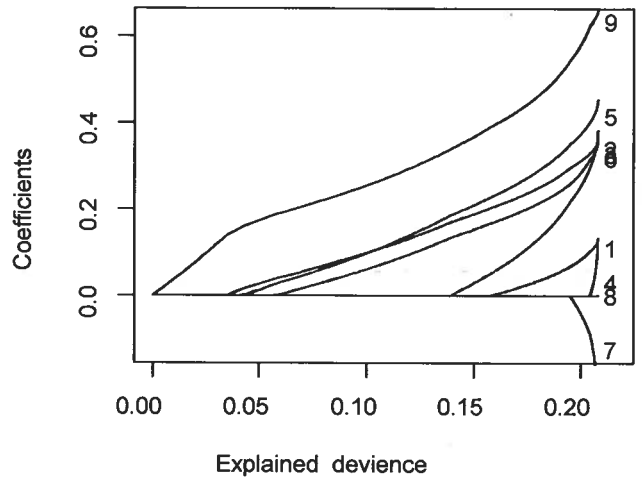
$$n = 9, N = 462$$

red = exact (convex), black = GPS

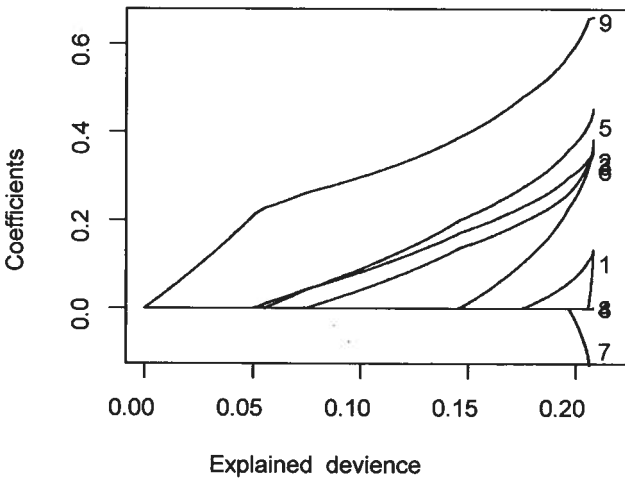
**Elastic Net 1.9**



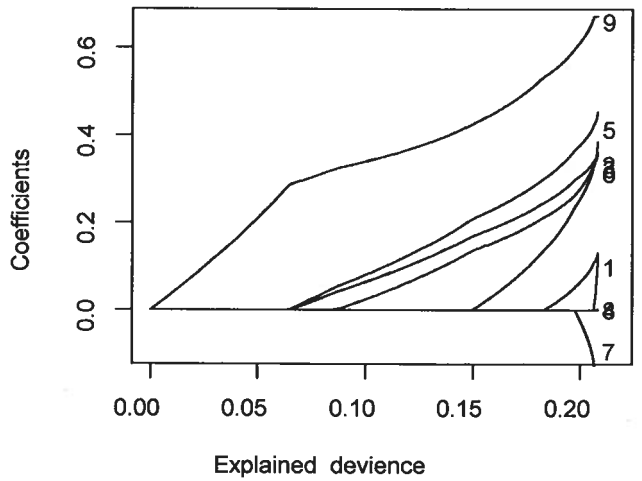
**Elastic Net 1.5**



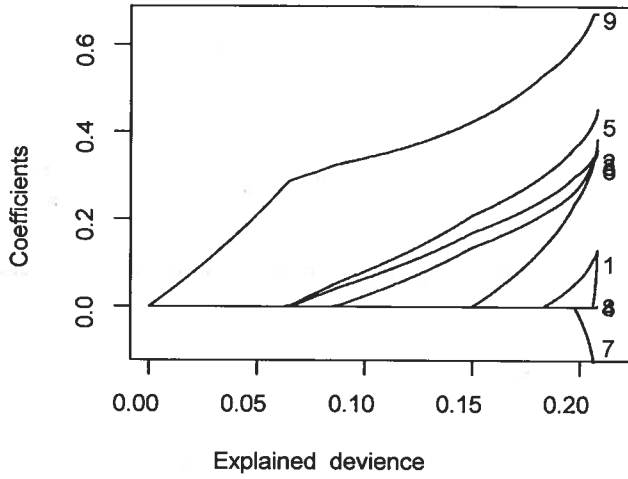
**Elastic Net 1.25**



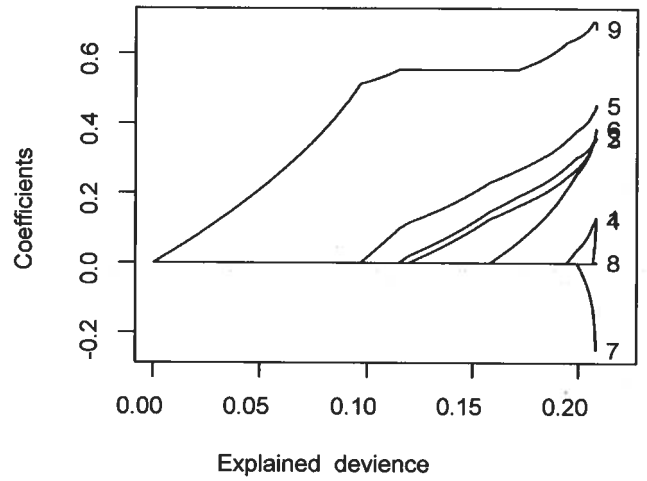
**Lasso**



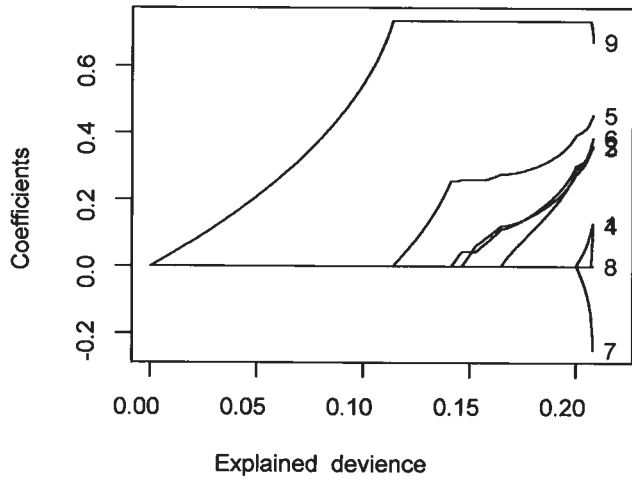
**Lasso**



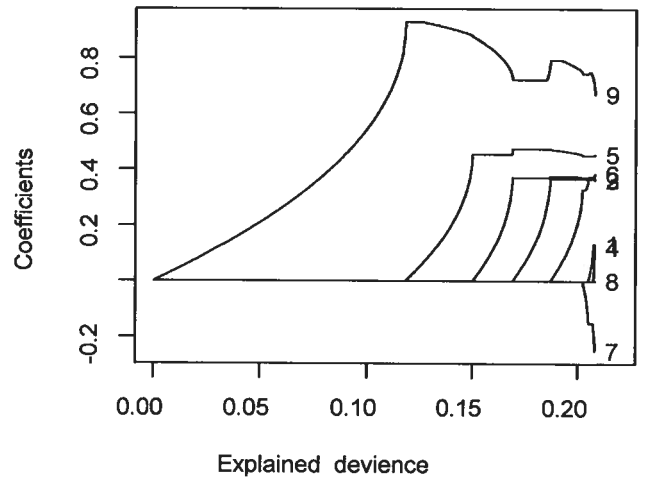
**Elastic Net 0.5**



**Elastic Net 0.25**



**Elastic Net 0.0**



## Regression: under-determined example

$$n = 10000, \quad N = 200$$

$$\mathbf{x}_i \sim N(\mathbf{0}, \mathbf{C}); \quad C_{jj} = 1, \quad C_{jk} = 0.4$$

$$y_i = \sum_{j=1}^n a_j^* x_{ij} + \varepsilon_i$$

$$\varepsilon_i \sim N(0, \sigma^2); \quad \sigma \sim 3/1 \text{ signal/noise}$$

$$|a_j^*| = [31 - j]_+, \quad \text{sign}(a_{j+1}^*) = -\text{sign}(a_j^*)$$

## Penalty Selection ( $\beta$ )

Regression: under-determined example

$$n = 10000, \quad N = 200$$

50 data sets  $\sim p(\mathbf{x}, y)$

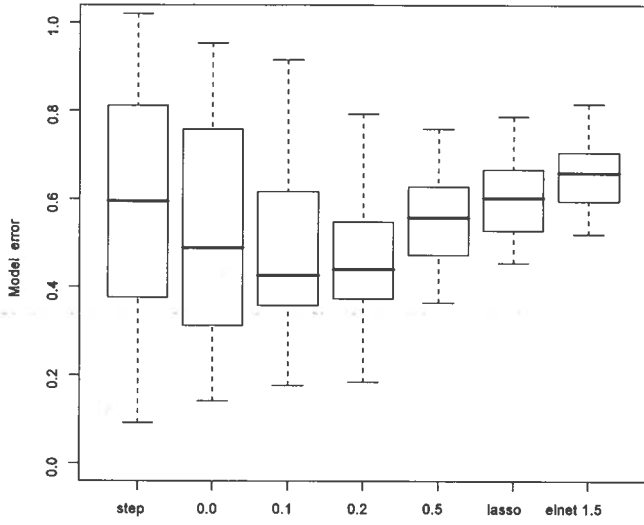
Distribution of closest “distance” to truth  $\mathbf{a}^*$  (risk)

Methods:

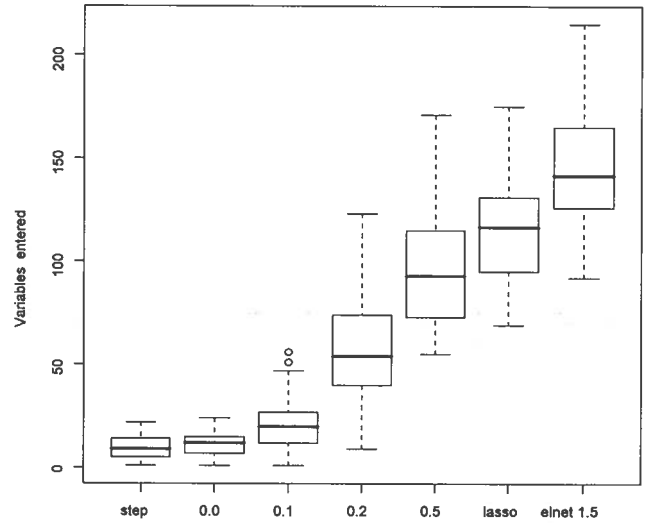
GPS:  $\beta \in \{0.0, 0.1, 0.2, 0.5\}$

forward stepwise, lasso, elastic net (1.5)

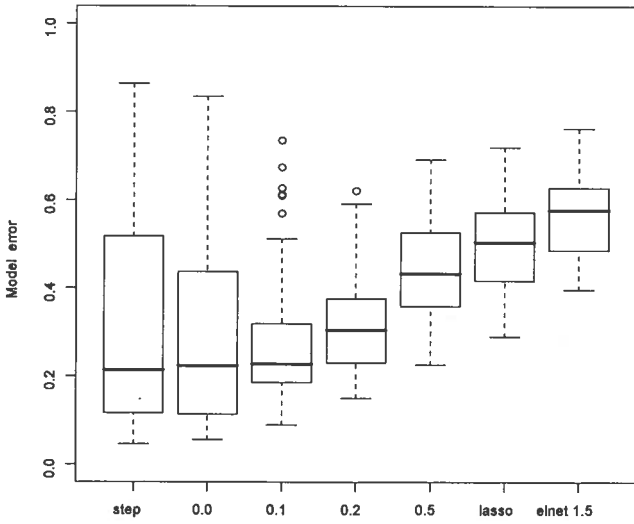
Corr = 0.4



Corr = 0.4



Corr = 0.0



Corr = 0.4, a > 0

