STK-IN4300
Statistical Learning Methods in Data Science

Riccardo De Bin

debin@math.uio.no

Outline of the lecture

- Ensemble Learning
  - Introduction
  - Boosting and regularization path
  - The “bet on sparsity” principle
- High-Dimensional Problems: $p \gg N$
  - When p is much larger than N
  - Computational short-cuts when $p \gg N$
  - Supervised Principal Component

Ensemble Learning: introduction

With **ensemble learning** we denote methods which:

- apply base learners to the data;
- combine the results of these learner.

Examples:

- bagging;
- random forests;
- boosting,
  - in boosting the learners evolves over time.

Ensemble Learning: boosting and regularization path

Consider the following algorithm,

Algorithm 16.1 *Forward Stagewise Linear Regression.*

```
1. Initialize $\check{\alpha}_k = 0,\ k = 1, \dots, K$. Set $\varepsilon > 0$ to some small constant,
   and $M$ large.

2. For $m = 1$ to $M$:

   (a) $(\beta^*, k^*) = \arg\min_{\beta,k} \sum_{i=1}^{N} \Big(y_i - \sum_{l=1}^{K} \check{\alpha}_l T_l(x_i) - \beta T_k(x_i)\Big)^2$.
   (b) $\check{\alpha}_{k^*} \leftarrow \check{\alpha}_{k^*} + \varepsilon \cdot \operatorname{sign}(\beta^*)$.

3. Output $f_M(x) = \sum_{k=1}^{K} \check{\alpha}_k T_k(x)$.
```

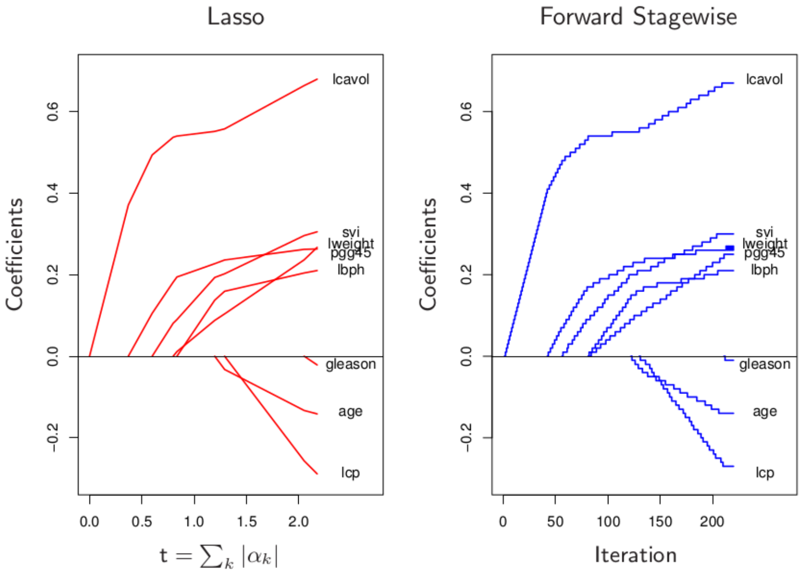
(note its similarities with the boosting algorithm)

Ensemble Learning: boosting and regularization path

In comparison with lasso:

- initialization ($\breve{\alpha}_k = 0, k = 1, \dots, K$) $\longleftrightarrow$ $\lambda = \infty$;
- for small values of M:
  - some $\breve{\alpha}_k$ are not updated $\longleftrightarrow$ coefficients "forced" to be 0;
  - $\breve{\alpha}_k^{[0]} \leqslant \breve{\alpha}_k^{[M]} \leqslant \breve{\alpha}_k^{[\infty]}$ $\longleftrightarrow$ shrinkage;
  - M inversely related to λ;
- for M large enough ($M = \infty$ in boosting) and $K < N$, $\breve{\alpha}_k^{[M]} = \hat{\alpha}_{LS} \longleftrightarrow \lambda = 0$

Ensemble Learning: boosting and regularization path



Ensemble Learning: boosting and regularization path

If

- all the basis learners T_K are mutually uncorrelated,

then

- for $\varepsilon \to 0$ and $M \to \infty$, such that $\varepsilon M \to t$,

Algorithm 16.1 gives the lasso solutions with $t = \sum_k |\alpha_k|$.

In general component-wise boosting and lasso do not provide the same solution:

- in practice, often similar in terms of prediction;
- for ε $(\nu) \to 0$ boosting (and forward stagewise in general) tends to the path of the least angle regression algorithm;
- lasso can also be seen as a special case of least angle.

Ensemble Learning: boosting and regularization path

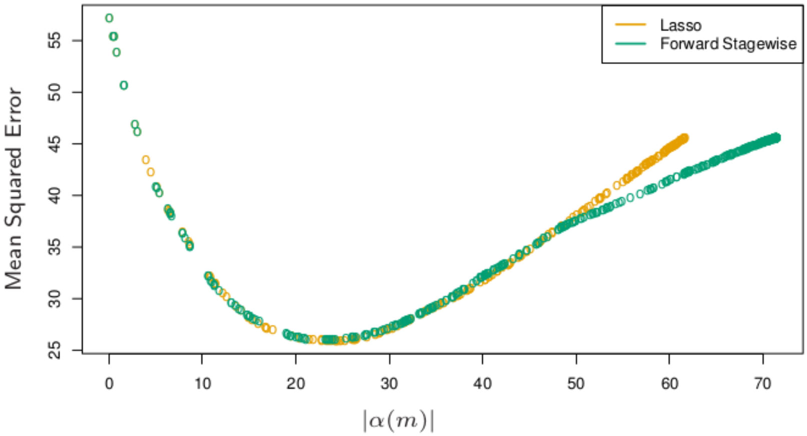
Consider 1000 Gaussian distributed variables:

- strongly correlated ($\rho = 0.95$) in blocks of 20;
- uncorrelated blocks;
- one variable with effect on the outcome for each block;
- effects generated from a standard Gaussian.

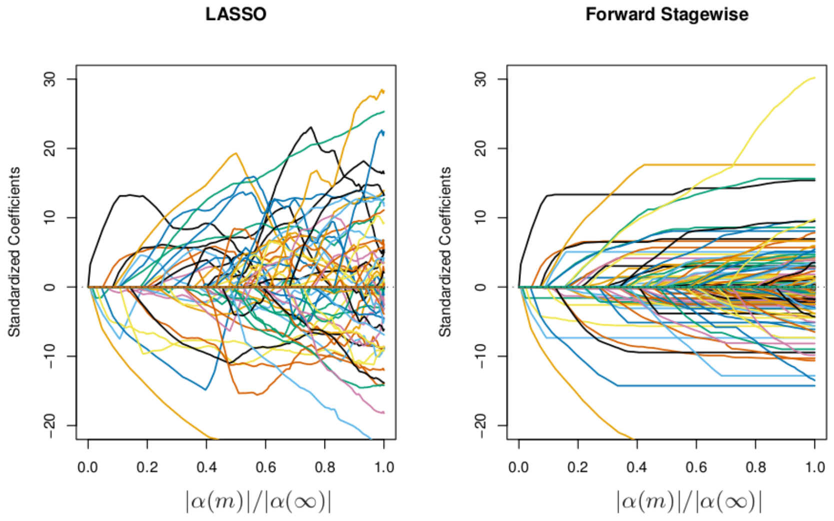
Moreover:

- added Gaussian noise;
- noise-to-signal ratio = 0.72.

Ensemble Learning: boosting and regularization path



Ensemble Learning: boosting and regularization path



Ensemble Learning: the "bet on sparsity" principle

We consider:

- L_1-type of penalty (shrinkage, variable selection);
- L_2-type of penalty (shrinkage, computationally easy);
- boosting's stagewise forward strategy minimizes something close to a L_1 penalized loss function;
- step-by-step minimization.

Can we characterize situations where one is preferable to the other?

Ensemble Learning: the "bet on sparsity" principle

Consider the following framework:

- 50 observations;
- 300 independent Gaussian variables.

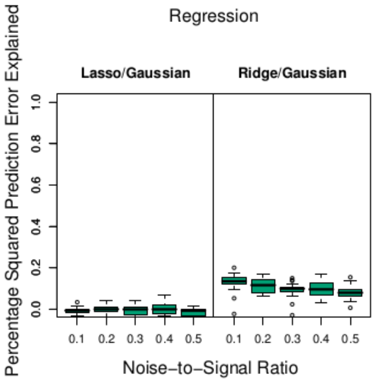
Three scenarios:

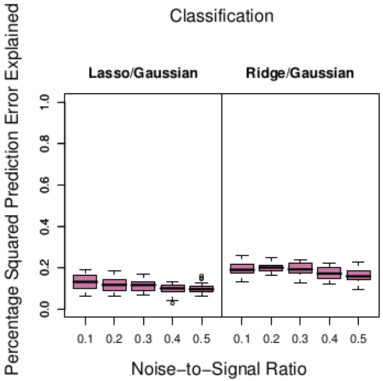
- all 300 variables are relevant;
- only 10 out of 300 variables are relevant;
- 30 out of 300 variables are relevant.

Outcome:

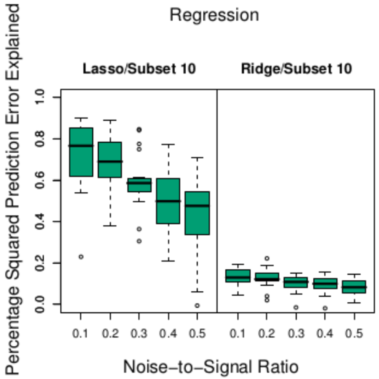
- regression (added standard Gaussian noise);
- classification (from an inverse-logit transformation of the linear predictor)

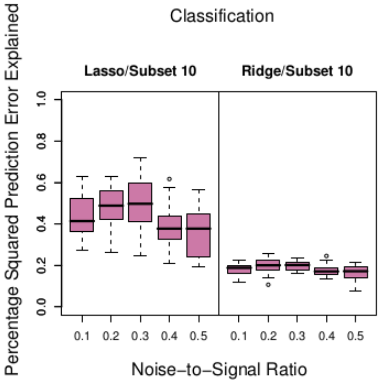
Ensemble Learning: the "bet on sparsity" principle



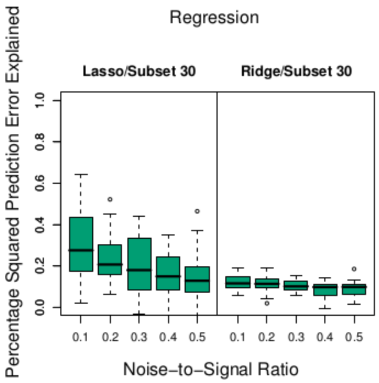


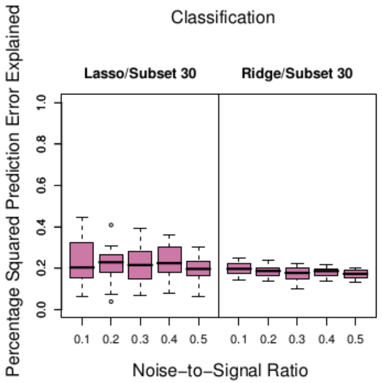
Ensemble Learning: the "bet on sparsity" principle





Ensemble Learning: the “bet on sparsity” principle





Ensemble Learning: the "bet on sparsity" principle

This means that:

- lasso performs better than ridge in sparse contexts;
- ridge gives better results if there are several relevant variables with small effects;
- anyway, in the dense case, the model does not explain a lot,
  - not enough data to estimate correctly several coefficients;

$$\downarrow$$

"bet on sparsity"

$$=$$

"use a procedure that does well in sparse problems, since no procedure does well in dense problems"

Ensemble Learning: the "bet on sparsity" principle

The degree of sparseness depends on:

- the unknown mechanism generating the data;
  - it depends on the number of relevant variables;
- size of the training set;
  - larger sizes allow estimating denser models;
- noise-to-signal ratio;
  - smaller NTR $\rightarrow$ denser models (same as before);
- size of the dictionary;
  - more base learners, potentially sparser models.

High-Dimensional Problems: when p is much larger than N

The case $p \gg N$ (number of variable much larger than the number of observations):

- very important in current applications,
  - e.g., in a common genetic study, $p \approx 23000$, $N \approx 100$;
- concerns about high variance and overfitting;
- highly regularized approaches are common:
  - lasso;
  - ridge;
  - boosting;
  - elastic-net;
  - ...

High-Dimensional Problems: when p is much larger than N

Consider the following example:

- $N = 100$;
- p variables from standard Gaussians with $\rho = 0.2$,

  (i) $p = 10$ (ii) $p = 100$ (iii) $p = 1000$.
- response from

$$Y = \sum_{j=1}^{p} X_j \beta_j + \sigma\epsilon;$$

- signal to noise ratio Var$[E(Y|X)]/\sigma^2 = 2$;
- true β from a standard Gaussian;

High-Dimensional Problems: when p is much larger than N

As a consequence, averaging over 100 replications:

- p_0, the number of significant β (i.e., $\beta : |\hat{\beta}/\hat{\text{se}}| > 2$)

  (i) $p_0 = 9$ (ii) $p_0 = 33$ (iii) $p_0 = 331$.

Consider 3 values of the penalty λ:

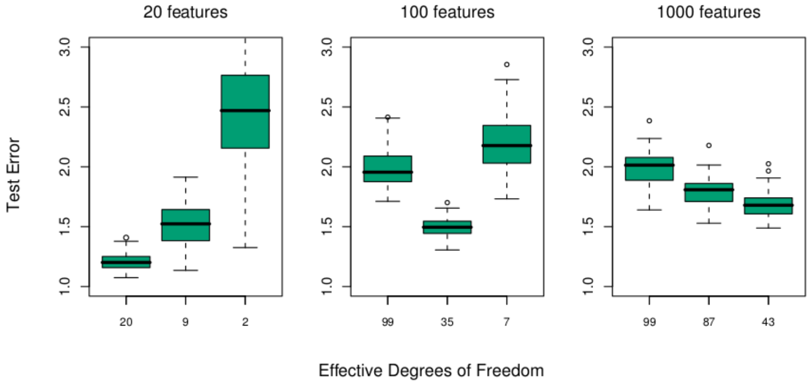
- $\lambda = 0.001$, which corresponds to

  (i) d.o.f. $= 20$ (ii) d.o.f. $= 99$ (iii) d.o.f. $= 99$;
- $\lambda = 100$, which corresponds to

  (i) d.o.f. $= 9$ (ii) d.o.f. $= 35$ (iii) d.o.f. $= 87$;
- $\lambda = 1000$, which corresponds to

  (i) d.o.f. $= 2$ (ii) d.o.f. $= 7$ (iii) d.o.f. $= 43$.

High-Dimensional Problems: when p is much larger than N



High-Dimensional Problems: when p is much larger than N

Remarks:

- with $p = 20$, ridge regression can find the relevant variables;
  - the covariance matrix can be estimated;
- moderate shrinkage works better in the middle case, in which we can find some non-zero effects;
- with $p = 1000$, there is no hope to find the relevant variables, and it is better to shrink down everything;
  - no possibility to estimate the covariance matrix.

High-Dimensional Problems: computational short-cuts when $p \gg N$

Consider the single-value decomposition of X,

$$\begin{aligned} X &= UDV^T \\ &= RV^T \end{aligned}$$

Where

- V is a $p \times N$ matrix with orthonormal columns;
- U is a $N \times N$ orthonormal matrix;
- D is a diagonal matrix with elements $d_1 \geqslant d_2 \geqslant \dots \geqslant d_N \geqslant 0$;
- R is a $N \times N$ with rows r_i.

High-Dimensional Problems: computational short-cuts when $p \gg N$

Theorem (Hastie et al., 2009, page 660):
Let $f^*(r_i) = \theta_0 + r_i^T\theta$ and consider the optimization problems:

$$(\hat{\beta}_0, \hat{\beta}) = \text{argmin}_{\beta_0, \beta \in \mathbb{R}^p} \sum_{i=1}^N L(y_i, \beta_0 + x_i\beta) + \lambda\beta^T\beta;$$

$$(\hat{\theta}_0, \hat{\theta}) = \text{argmin}_{\theta_0, \theta \in \mathbb{R}^N} \sum_{i=1}^N L(y_i, \theta_0 + r_i\theta) + \lambda\theta^T\theta.$$

Then $\beta_0 = \theta_0$ and $\hat{\beta} = V\hat{\theta}$.

High-Dimensional Problems: computational short-cuts when $p \gg N$

Note:

- we can replace the p-dim vectors x_i with the N-dim vectors r_i;
- same penalization, but with way fewer predictors;
- the N-dimensional solution $\hat{\theta}$ is transformed back to the p-dimensional $\hat{\beta}$ by simple matrix multiplication;
- it only works for linear models;
- it only works for quadratic penalties.

- a $O(p^3)$ problem is reduced to a $O(p^2N)$ problem;
  - relevant for $p > N$.

High-Dimensional Problems: computational short-cuts when $p \gg N$

Example (ridge regression):
Consider the estimate of a ridge regression,

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

Replacing X with RV^T, we obtain

$$\hat{\beta} = V(R^T R + \lambda I)^{-1} R^T y,$$

i.e.,

$$\hat{\beta} = V\hat{\theta},$$

where

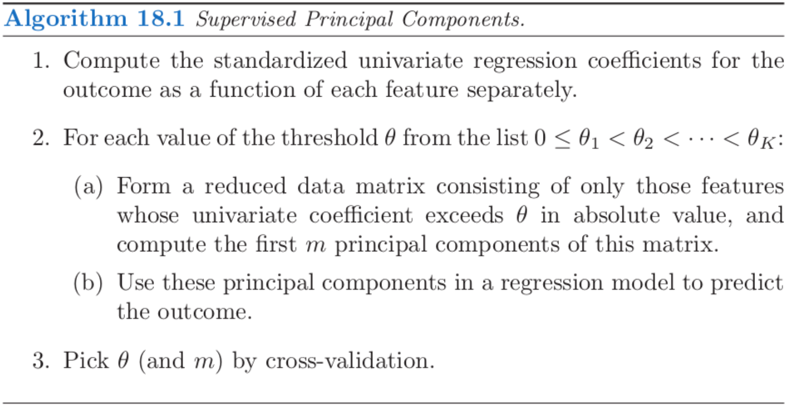
$$\hat{\theta} = (R^T R + \lambda I)^{-1} R^T y.$$

High-Dimensional Problems: computational short-cuts when $p \gg N$

Note:

- it cannot be applied to lasso;
- the short-cut is particularly relevant for finding the best λ via cross-validation;
- it can be shown that one need to construct R only once,
  - use the same R for each of the CV-folds.

High-Dimensional Problems: supervised principal component



Algorithm 18.1 *Supervised Principal Components.*

```
1. Compute the standardized univariate regression coefficients for the
   outcome as a function of each feature separately.

2. For each value of the threshold $\theta$ from the list $0 \leq \theta_1 < \theta_2 < \cdots < \theta_K$:

   (a) Form a reduced data matrix consisting of only those features
       whose univariate coefficient exceeds $\theta$ in absolute value, and
       compute the first $m$ principal components of this matrix.

   (b) Use these principal components in a regression model to predict
       the outcome.

3. Pick $\theta$ (and $m$) by cross-validation.
```

High-Dimensional Problems: supervised principal component

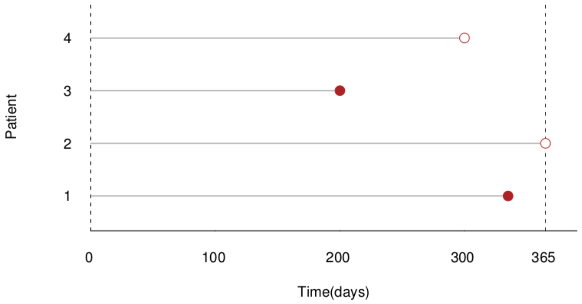
Note:

- in the first step, the variables univariately most associated with the outcome are selected:
  - the measure depends on the nature of the outcome;
  - include all associations larger than a threshold θ;
  - may include highly correlated variables.
- in step 2, perform PCR on the reduced variable space:
  - use the first m components;
  - assure shrinkage.
- both θ and m must be computed by cross-validation:
  - 2-dimension tuning parameter.

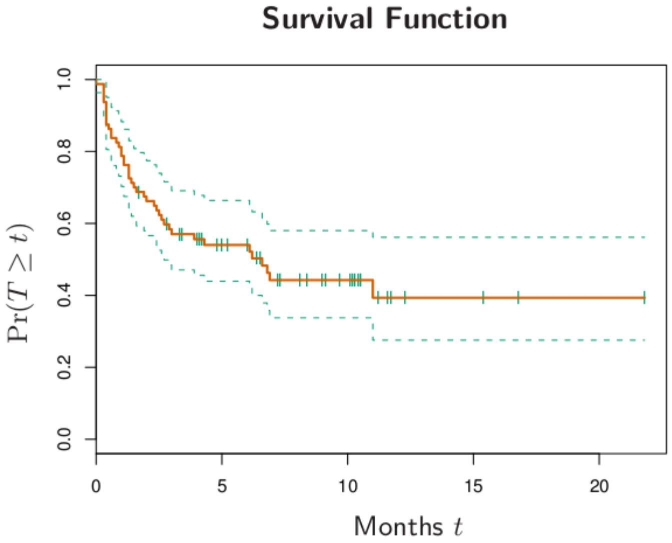
High-Dimensional Problems: supervised principal component

Example (survival analysis with microarray data):

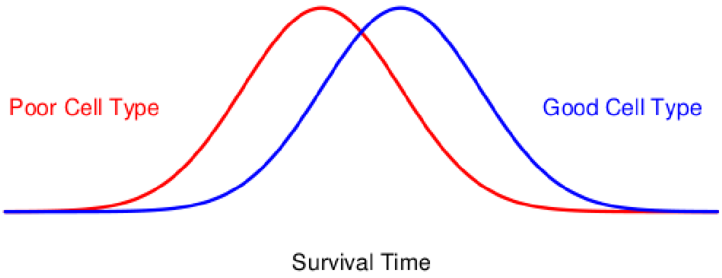
- data from Rosenwald et al. (2002);
- input: 7399 gene expressions of 240 patients;
- response: survival time (potentially right censored);
- divided in a training (160 patients) and (80) test set.



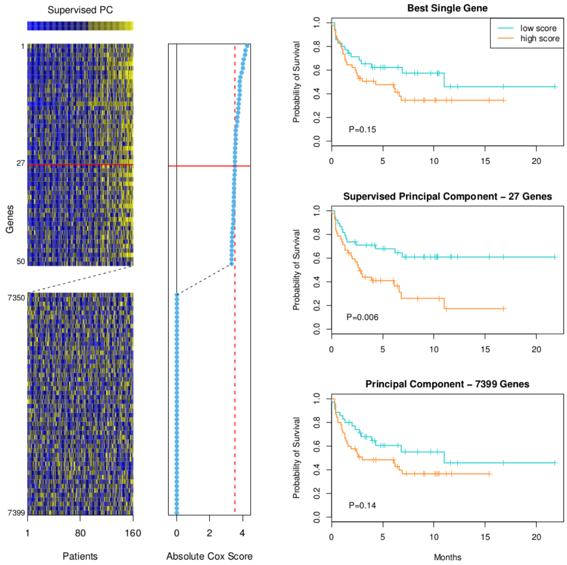
High-Dimensional Problems: supervised principal component



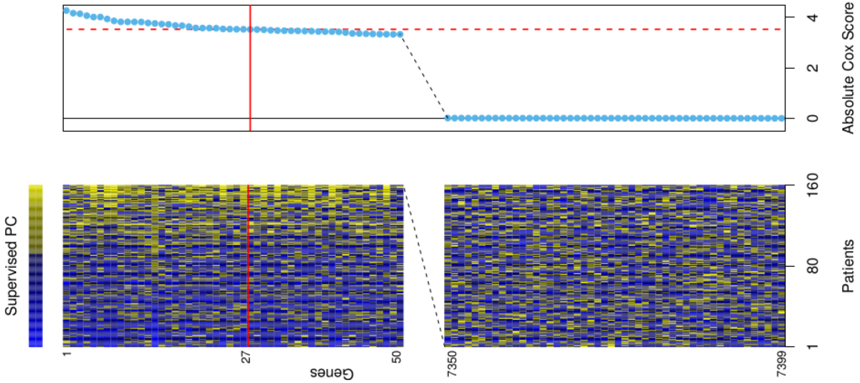
High-Dimensional Problems: supervised principal component



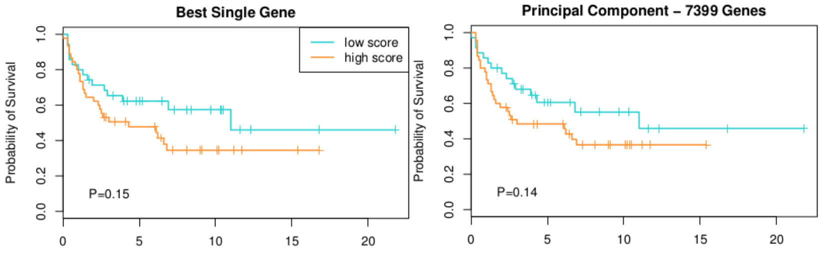
High-Dimensional Problems: supervised principal component

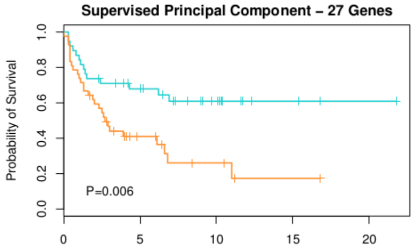


High-Dimensional Problems: supervised principal component



High-Dimensional Problems: supervised principal component





High-Dimensional Problems: supervised-PC and latent-variable modelling

Consider a **latent-variable model**,

$$Y = \beta_0 + \beta_1 U + \epsilon$$

where

$$X_j = \alpha_{0j} + \alpha 1 j U + \epsilon_j \qquad \text{if } j \in \mathcal{P}$$

- ϵ and ϵ_j have mean 0 and they are independent of the other variables in their respective models;
- X_j are independent of U if $j \notin \mathcal{P}$.

High-Dimensional Problems: supervised-PC and latent-variable modelling

Supervised-PC can be seen as a method to fit this kind of model:

- step 1: identify the $j \in \mathcal{P}$;
  - on average β_1 is not zero only if $\alpha_{1j} \neq 0$;
- step 2a: estimate α_{0j} and α_{1j}
  - natural if $\epsilon_j \sim N(0; \sigma^2)$;
- step 2b: estimate β_0 and β_1 (fit the model).

High-Dimensional Problems: supervised-PC and partial least square

Supervised-PC versus partial least square:

- both aim at considering both large variation and correlation with the outcome;
- supervised-PC removes those which are not relevant;
- partial least square downgrades them

$$\downarrow$$

"thresholded PLS":

- apply PLS on only the variables selected by supervised-PC.

High-Dimensional Problems: supervised-PC and partial least square

Thresholded PLS can be seen as a noisy version of supervised-PC,

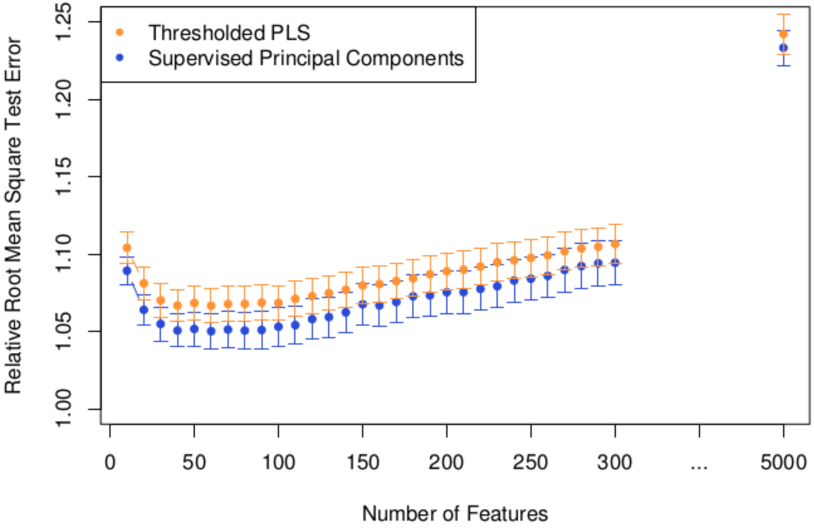
- first PLS variate,
$$z = \sum_{j \in \mathcal{P}} \langle y, x_j \rangle x_j;$$
- supervised principal component direction,
$$u = \frac{1}{d^2} \sum_{j \in \mathcal{P}} \langle y, x_j \rangle x_j.$$

Set $p_1 = |\mathcal{P}|$. It can be shown (Bair & Tibshirani, 2004) that, for $N, p_1, p \rightarrow \infty$, s.t. $p_1/N \rightarrow 0$

$$z = u + O_p(1)$$
$$\hat{u} = u + O_p(\sqrt{p_1/N})$$

where u is the true latent variable.

High-Dimensional Problems: supervised-PC and partial least square



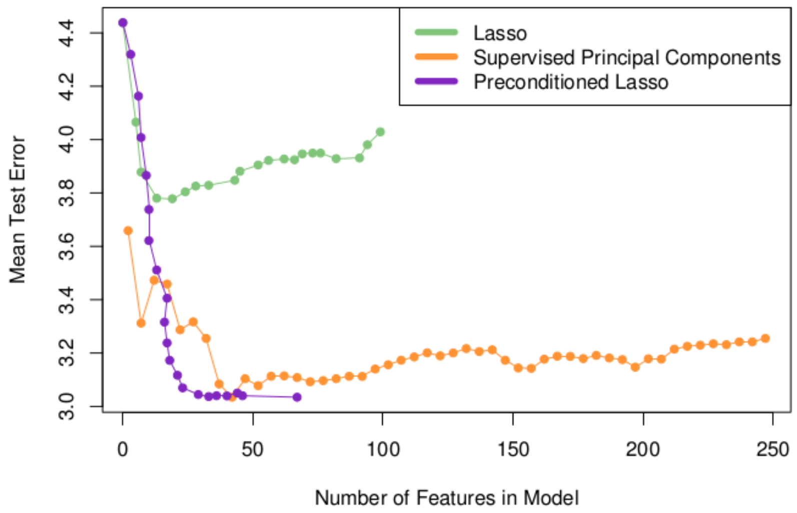
High-Dimensional Problems: pre-conditioning

Supervised-PC can also be used to improve lasso performance, through pre-conditioning

- compute $\hat{y}_i$ the supervised-PC prediction for each observation;
- apply lasso using $\hat{y}_i$ instead of y_i as outcome;
  - using all variables, not only those selected by supervised-PC.

The idea is to remove the noise, so lasso is not affected by the large number of noisy variables.

High-Dimensional Problems: pre-conditioning



References I

Bair, E. & Tibshirani, R. (2004). Semi-supervised methods to predict patient survival from gene expression data. *PLoS Biology* **2**, e108.

Hastie, T., Tibshirani, R. & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction (2nd Edition)*. Springer, New York.

Rosenwald, A., Wright, G., Chan, W. C., Connors, J. M., Campo, E., Fisher, R. I., Gascoyne, R. D., Muller-Hermelink, H. K., Smeland, E. B., Giltnane, J. M. et al. (2002). The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma. *New England Journal of Medicine* **346**, 1937–1947.