STK-IN4300
Statistical Learning Methods in Data Science

Riccardo De Bin

debin@math.uio.no

Outline of the lecture

Gradient Boosting: from the last lecture

In the last lecture:
- boosting as implementation of "wisdom of the crowds";
- repeatedly apply a weak learner to modification of the data;
- from AdaBoost to gradient boosting;
- forward stagewise additive modelling;
- importance of shrinkage.

Gradient Boosting: general gradient boosting

The general algorithm for the gradient boosting:
1. initialize the estimate, e.g., $f_0(x) = 0$;
2. for $m = 1, \dots, m_{\text{stop}}$,
   - 2.1 compute the negative gradient vector, $u_m = -\left.\frac{\partial L(y, f(x))}{\partial f(x)}\right|_{f(x)=\hat{f}_{m-1}(x)}$;
   - 2.2 fit the base learner to the negative gradient vector, $h_m(u_m, x)$;
   - 2.3 update the estimate, $f_m(x) = f_{m-1}(x) + \nu h_m(u_m, x)$.
3. final estimate, $\hat{f}_{m_{\text{stop}}}(x) = \sum_{m=1}^{m_{\text{stop}}} \nu h_m(u_m, x)$

Gradient Boosting: L_2 boosting with linear learner

The L_2Boost algorithm (linear learner, L_2 loss):

1. initialize the estimate, e.g., $\hat{\beta}_0 = 0$;
2. for $m = 1, \dots, m_{\text{stop}}$,
   - 2.1 compute the negative gradient vector, $u_m = -\left.\frac{\partial L(y, f(X,\beta))}{\partial f(X,\beta)}\right|_{\beta = \hat{\beta}_{m-1}}$;
   - 2.2 fit the base learner to the negative gradient vector, $b_m(u_m, X) = \nu (X^T X)^{-1} X^T u_m$;
   - 2.3 update the estimate, $\hat{\beta}_m = \hat{\beta}_{m-1} + b_m(u_m, x)$.
3. final estimate, $\hat{f}_{m_{\text{stop}}}(x) = X^T \hat{\beta}_m$.

L_2 boosting with linear learner: boosting operator

Consider the regression model $y_i = f(x_i) + \epsilon_i$, $i = 1, \dots, N$,

- ϵ_i i.i.d. with $E[\epsilon_i] = 0$, $\text{Var}[\epsilon_i] = \sigma^2$.
- linear learner $\mathcal{S} : \mathbb{R}^N \to \mathbb{R}^N$ ($\mathcal{S}y = \hat{y}$);

Note that:

- $\hat{f}_m(x) = \hat{f}_{m-1}(x) + \mathcal{S}u_m$;
- $u_m = y - \hat{f}_{m-1}(x) = u_{m-1} - \mathcal{S}u_{m-1} = (I - \mathcal{S})u_{m-1}$;
- iterating, $u_m = (I - \mathcal{S})^m$, $m = 1, \dots, m_{\text{stop}}$.

Because $\hat{f}_m(x) = \mathcal{S}y$, then $\hat{f}_{m_{\text{stop}}}(x) = \sum_{m=0}^{m_{\text{stop}}} \mathcal{S}(I - \mathcal{S})^m y$, i.e.,

$$\hat{f}_{m_{\text{stop}}}(x) = \underbrace{(I - (I - \mathcal{S})^{m+1})}_{\text{boosting operator } \mathcal{B}_m} y.$$

L_2 boosting with linear learner: properties

Consider a linear learner $\mathcal{S}$ (e.g., least square). Then

Proposition 1 (Bühlmann & Yu, 2003): The eigenvalues of the L_2Boost operator $\mathcal{B}_m$ are

$$\left\{1 - (1 - \lambda_k)^{m_{\text{stop}}+1}, k = 1, \dots, N\right\}.$$

If $\mathcal{S} = \mathcal{S}^T$ (i.e., symmetric), then $\mathcal{B}_m$ can be diagonalized with an orthonormal transformation,

$$\mathcal{B}_m = U D_m U^T, \qquad D_m = \text{diag}(1 - (1 - \lambda_k)^{m_{\text{stop}}+1})$$

where $UU^T = U^T U = I$.

L_2 boosting with linear learner: properties

We can now compute:

- $$\begin{aligned}\text{bias}^2(m, \mathcal{S}; f) &= N^{-1} \sum_{i=1}^{N} (E[\hat{f}_m(x_i)] - f)^2 \\ &= N^{-1} f^T U \text{diag}((1 - \lambda_k)^{2m+2}) U^T f;\end{aligned}$$
- $$\begin{aligned}\text{Var}(m, \mathcal{S}; \sigma^2) &= N^{-1} \sum_{i=1}^{N} (\text{Var}[\hat{f}_m(x_i)]) \\ &= \sigma^2 N^{-1} \sum_{i=1}^{N} (1 - (1 - \lambda_k)^{m+1})^2;\end{aligned}$$

and

- $\text{MSE}(m, \mathcal{S}; f, \sigma^2) = \text{bias}^2(m, \mathcal{S}; f) + \text{Var}(m, \mathcal{S}; \sigma^2)$.

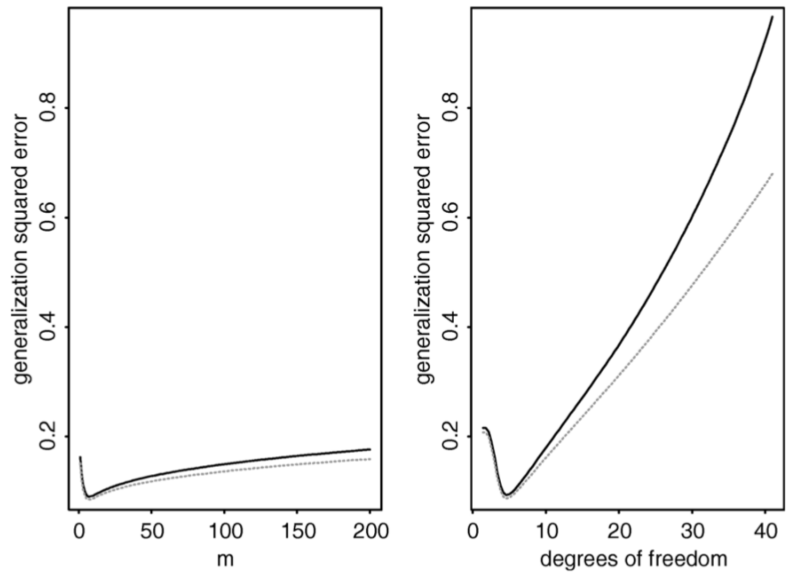
L_2 boosting with linear learner: properties

Assuming $0 < \lambda_k \leqslant 1$, $k = 1, \dots, N$, note that:

- bias$^2(m, \mathcal{S}; f)$ decays exponentially fast for m increasing;
- Var$(m, \mathcal{S}; \sigma^2)$ increases exponentially slow for m increasing;
- $\lim_{m \to \infty}$ MSE$(m, \mathcal{S}; f, \sigma^2) = \sigma^2$;
- if $\exists k : \lambda_k < 1$ (i.e., $\mathcal{S} \neq I$), then $\exists m :$ MSE$(m, \mathcal{S}; f, \sigma^2) < \sigma^2$;
- if $\forall k : \lambda_k < 1, \frac{\mu_k}{\sigma^2} > \frac{1}{(1-\lambda_k)^2} - 1$, then $MSE_{\mathcal{B}_m} < MSE_{\mathcal{S}}$, where $\mu = U^T f$ (μ represents f in the coordinate system of the eigenvectors of $\mathcal{S}$).

(for the proof, see Bühlmann & Yu, 2003, Theorem 1)

L_2 boosting with linear learner: properties



L_2 boosting with linear learner: properties

About $\frac{\mu_k}{\sigma^2} > \frac{1}{(1-\lambda_k)^2} - 1$:

- a large left side means that f is relatively complex compared with the noise level σ^2;
- a small right side means that λ_k is small, i.e. the learner shrinks strongly in the direction of the k-th eigenvector;
- therefore, to have boosting bringing improvements:
  - there must be a large signal to noise ratio;
  - the value of λ_k must be sufficiently small;

$\downarrow$

use a weak learner!!!

L_2 boosting with linear learner: properties

There is a further intersting theorem in Bühlmann & Yu (2003),

Theorem 2: Under the assumption seen till here and $0 < \lambda_k \leqslant 1$, $k = 1, \dots, N$, and assuming that $E[|\epsilon_1|^p] < \infty$ for $p \in \mathbb{N}$,

$$N^{-1} \sum_{i=1}^{N} E[(\hat{f}_m(x_i) - f(x_i))^p] = E[\epsilon_1^p] + O(e^{-Cm}), \quad m \to \infty$$

where $C > 0$ does not depend on m (but on N and p).

This theorem can be used to argue that boosting for classification is resistant to overfitting (for $m \to \infty$, exponentially small overfitting).

Gradient Boosting: boosting in high-dimensions

The boosting algorithm is working in high-dimension frameworks:

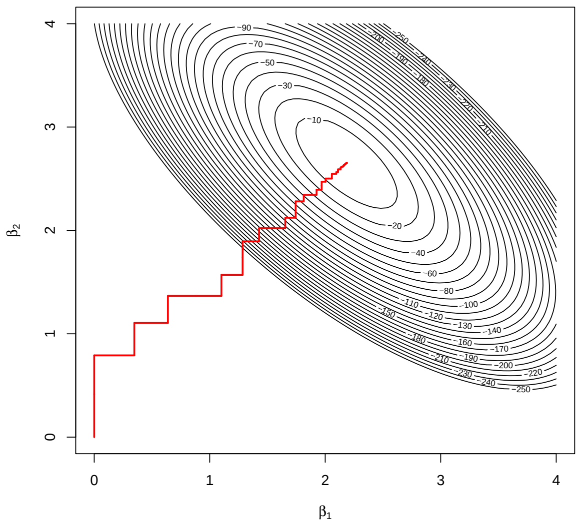
- forward stagewise additive modelling;
- at each step, only one dimension (component) of X is updated at each iteration;
- in a parametric setting, only one $\hat{\beta}_j$ is updated;
- an additional step in which it is decided which component to update must be computed at each iteration.

Gradient Boosting: component-wise L_2Boost with linear learner

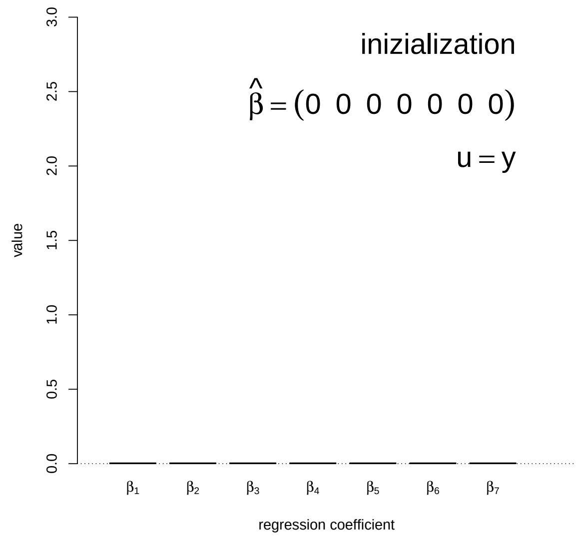
Component-wise L_2Boost algorithm:

1. initialize the estimate, e.g., $\hat{\beta} = (0, \dots, 0)$;
2. for $m = 1, \dots, m_{\text{stop}}$,
   - compute the negative gradient vector, $u = -\frac{\partial L(y, f(x,\beta))}{\partial f(x,\beta)}\Big|_{\beta=\hat{\beta}}$ for the j-th component only;
   - fit the base learner to the negative gradient vector, $\hat{h}(u, x_j)$;
   - select the best update j^* (usually that minimizes the loss);
   - include the shrinkage factor, $\hat{b}_j = \nu \hat{h}(u, x_j)$;
   - update the estimate, $\hat{\beta}_{j*} = \hat{\beta}_{j*} + \hat{b}_{j*}$.
3. final estimate, $\hat{f}_{m_{\text{stop}}}(x) = X^T \hat{\beta}^{[m_{\text{stop}}]}$ (for linear regression).

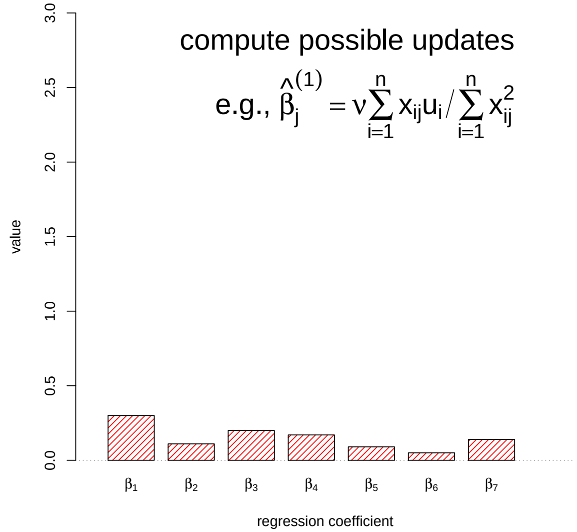
Boosting: minimization of the loss function



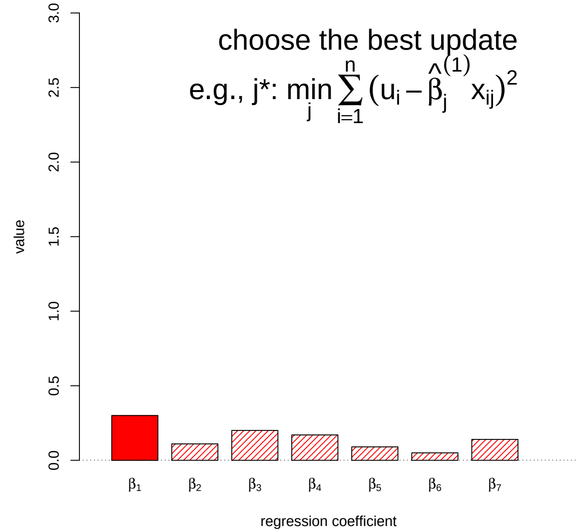
Boosting: parameter estimation



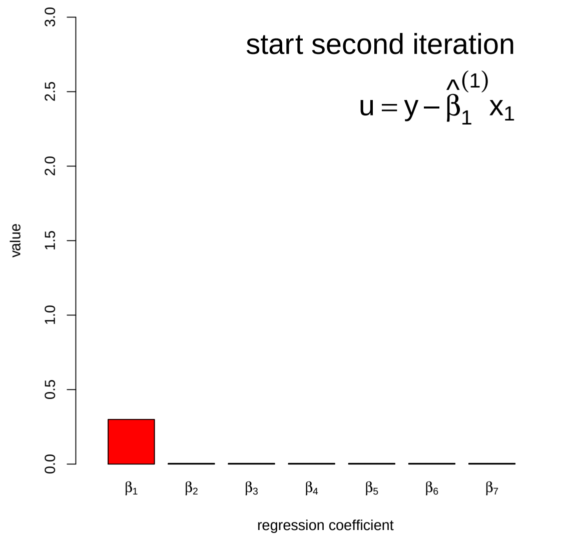
Boosting: parameter estimation



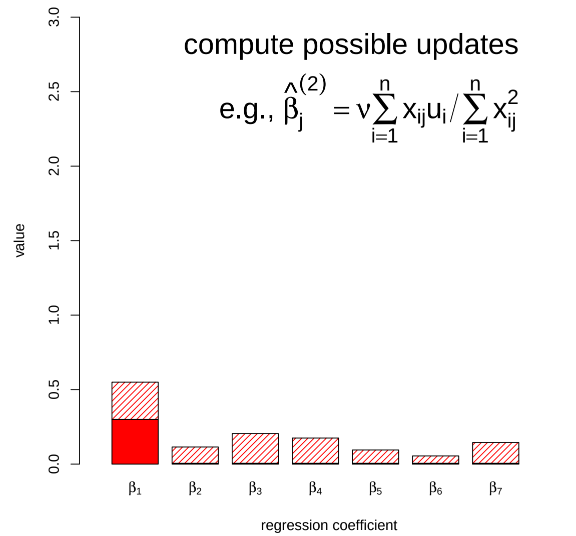
Boosting: parameter estimation



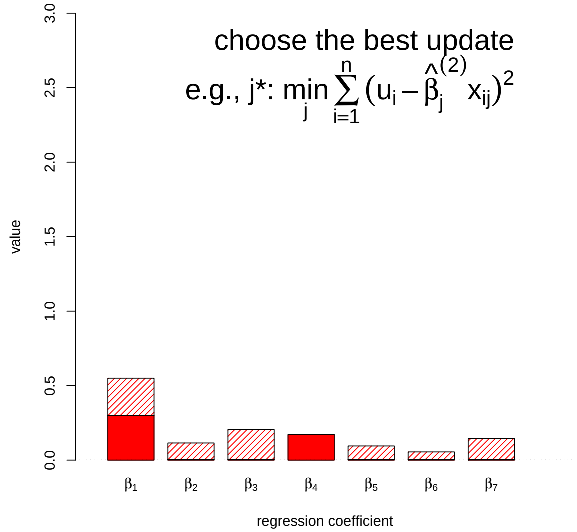
Boosting: parameter estimation



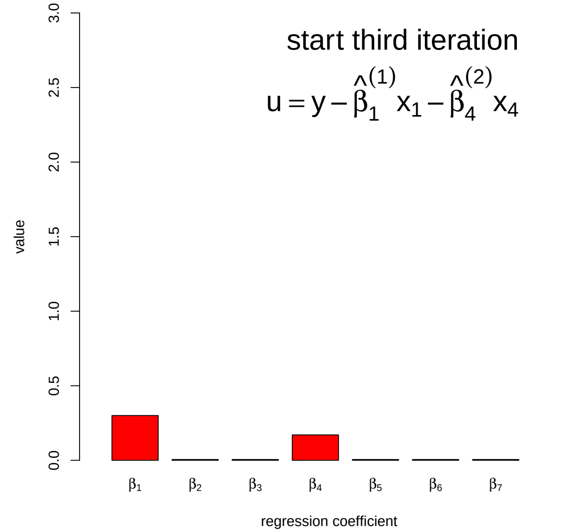
Boosting: parameter estimation



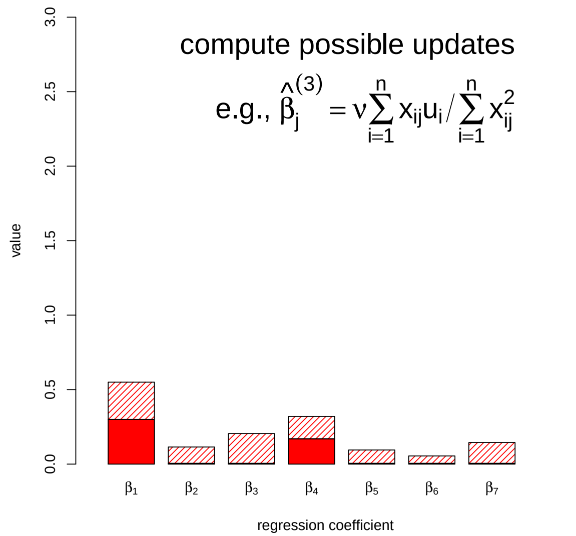
Boosting: parameter estimation



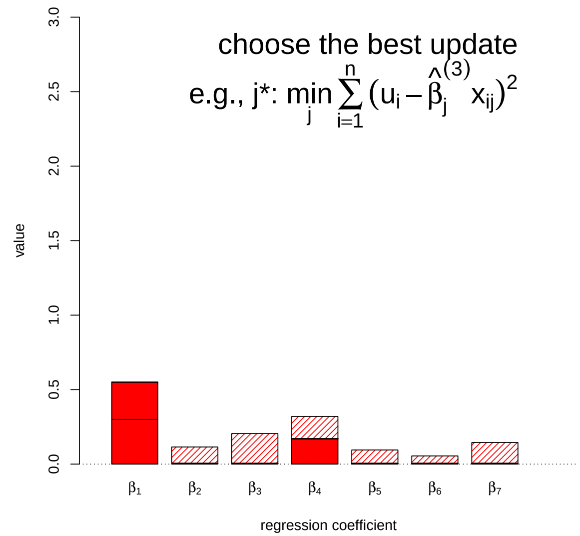
Boosting: parameter estimation



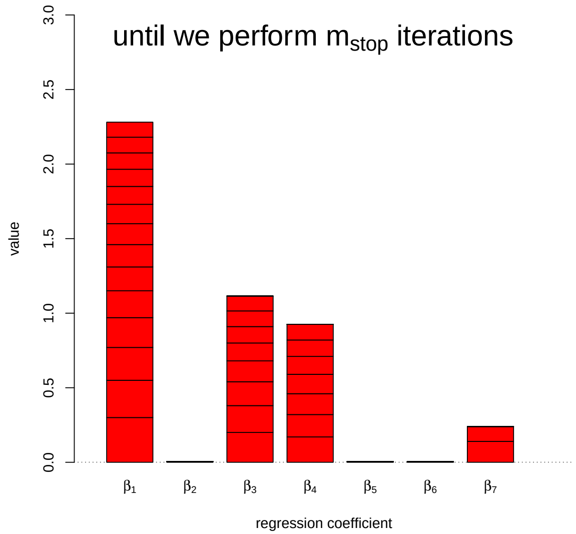
Boosting: parameter estimation



Boosting: parameter estimation



Boosting: parameter estimation



Boosting: tuning parameters

- The update step is regulated by the shrinkage parameter ν;
- as long as its magnitude is reasonable, the choice of the penalty parameter does not influence the procedure;
- the choice of the number of iterations m_{stop} is highly relevant;
- m_{stop} (complexity parameter) influences variable selection properties and model sparsity;
- m_{stop} controls the amount of shrinkage;
  - m_{stop} too small results in a model which is not able to describe the data variability;
  - an excessively large m_{stop} causes overfitting and causes the selection of irrelevant variables.
- there is no standard approach → repeated cross-validation (Seibold et al., 2016).

Likelihood-based Boosting: introduction

A different version of boosting is the so-called **likelihood-based boosting** (Tutz & Binder, 2006):

- based on the concept of GAM as well;
- loss function as a negative log-likelihood;
- uses standard statistical tools (Fisher scoring, basically a Newton-Raphson algorithm) to minimize the loss function;
- likelihood-based boosting and gradient boosting are equal only in Gaussian regression (De Bin, 2016).

Likelihood-based Boosting: algorithm

The simplest implementation of the likelihood-based boosting is BoostR, based on the ridge estimator:

Algorithm: BoostR

Step 1: Initialization. $\hat{\beta}_{(0)} = (X^{\mathrm{T}}X + \lambda I_p)^{-1}X^{\mathrm{T}}y$, $\hat{\mu}_{(0)} = X\hat{\beta}_{(0)}$.

Step 2: Iteration. For $m = 1, 2, \ldots$ apply ridge regression to the model for residuals

$$y - \hat{\mu}_{(m-1)} = X\beta^R + \varepsilon,$$

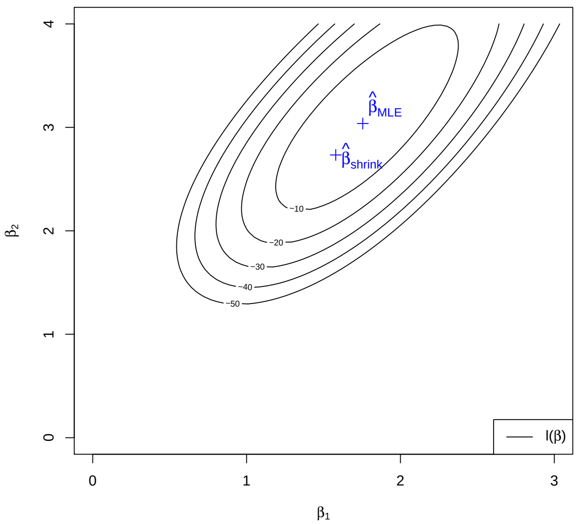
yielding solutions $\hat{\beta}^R_{(m)} = (X^{\mathrm{T}}X + \lambda I_p)^{-1}X^{\mathrm{T}}(y - \hat{\mu}_{(m-1)})$, $\hat{\mu}^R_{(m)} = X\hat{\beta}^R_{(m)}$.

The new estimate is obtained by $\hat{\mu}_{(m)} = \hat{\mu}_{(m-1)} + \hat{\mu}^R_{(m)}$.

see also Tutz & Binder (2007).

In the rest of the lecture we will give the general idea and see its implementation as a special case of gradient boosting.

Likelihood-based Boosting: introduction



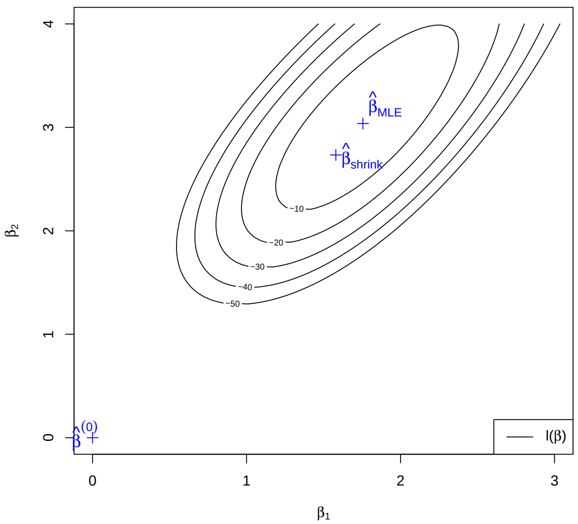
Following the statistical interpretation of boosting:

maximize the log-likelihood $\ell(\beta)$ (equivalently, $-\ell(\beta)$ is the loss function to minimize);

prediction → shrinkage aim at $\hat{\beta}_{shrink}$, not $\hat{\beta}_{MLE}$;

best solution is "between" 0 and $\hat{\beta}_{MLE}$.

Likelihood-based Boosting: introduction



starting point...
maximize a log-likelihood...

$\Downarrow$

Newton-Raphson method (or Fisher scoring).

Basic idea:
- apply Newton-Raphson;
- stop early enough to end in $\hat{\beta}_{shrink}$ and not in $\hat{\beta}_{MLE}$.

Likelihood-based Boosting: Newton-Raphson

General Newton-Raphson step:

$$\hat{\beta}^{[m]} = \hat{\beta}^{[m-1]} + \left(-\ell_{\beta\beta}(\beta)|_{\beta=\hat{\beta}^{[m-1]}}\right)^{-1} \ell_{\beta}(\beta)|_{\beta=\hat{\beta}^{[m-1]}},$$

where:
- $\ell_{\beta}(\beta) = \frac{\partial \ell(\beta)}{\partial \beta}$;
- $\ell_{\beta\beta}(\beta) = \frac{\partial^2 \ell(\beta)}{\partial \beta^T \partial \beta}$.

For convenience, let us rewrite the general step as

$$\underbrace{\hat{\beta}^{[m]} - \hat{\beta}^{[m-1]}}_{\text{improvement at step } m} = 0 + \left(-\ell_{\beta\beta}(\beta|\hat{\beta}^{[m-1]})\Big|_{\beta=0}\right)^{-1} \ell_{\beta}(\beta|\hat{\beta}^{[m-1]})\Big|_{\beta=0}.$$

Likelihood-based Boosting: Newton-Raphson

Control the Newton-Raphson algorithm:
- we need to force the estimates to be between 0 and $\hat{\beta}_{MLE}$;
- we need to be able to stop at $\hat{\beta}_{shrink}$.

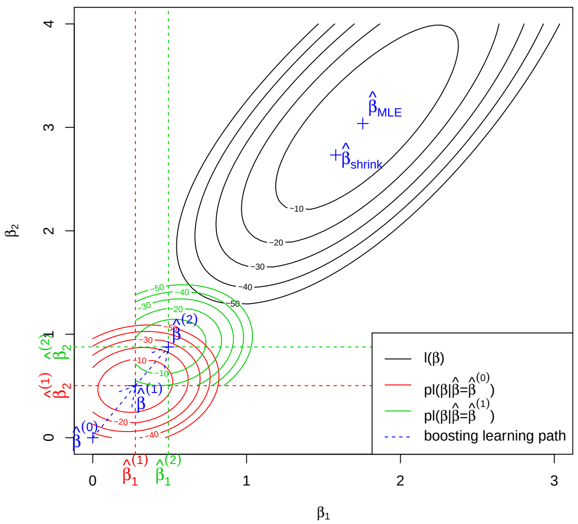
$\Rightarrow$ we need smaller "controlled" improvements.

Solution: penalize the log-likelihood!
- $p\ell(\beta) \leftarrow \ell(\beta) - \frac{1}{2}\lambda||\beta||_2^2$;
- $p\ell_{\beta}(\beta) \leftarrow \ell_{\beta}(\beta) - \lambda||\beta||_1$;
- $p\ell_{\beta\beta}(\beta) \leftarrow \ell_{\beta\beta}(\beta) - \lambda$;

Now the general step is:

$$\underbrace{\hat{\beta}^{[m]} - \hat{\beta}^{[m-1]}}_{\text{improvement at step } m} = \left(-\ell_{\beta\beta}(\beta|\hat{\beta}^{[m-1]})\Big|_{\beta=0} + \lambda\right)^{-1} \ell_{\beta}(\beta|\hat{\beta}^{[m-1]})\Big|_{\beta=0}$$

Likelihood-based Boosting: visualization



As long as λ is ‘big enough’, the boosting learning path is going to hit $\hat{\beta}_{shrink}$.

We must stop at that point: the number of boosting iterations (m_{stop}) is crucial!

Likelihood-based Boosting: likelihood-based vs gradient

In the likelihood-based boosting we:

- repeatedly implement the first step of Newton-Raphson;
- update at each step estimates and likelihood.

Small improvements:

- parabolic approximation;
- fit the negative gradient on the data by a base-learner (e.g., least-square estimator)

$$\hat{\beta}^{[m]} - \hat{\beta}^{[m-1]} = \left(X^T X + \lambda\right)^{-1} X^T \underbrace{\left.\frac{\partial \ell(\eta(\beta, X))}{\partial \eta(\beta, X)}\right|_{\hat{\beta}^{[m-1]}}}_{\text{negative gradient}}$$

Likelihood-based Boosting: likelihood-based vs gradient

Substituting

$$\nu = \left(X^T X + \lambda\right)^{-1} X^T X$$

one obtains the expression of the L_2Boost for (generalized) linear models seen before,

$$\hat{\beta}^{[m]} - \hat{\beta}^{[m-1]} = \nu \left(X^T X\right)^{-1} X^T \left.\frac{\partial \ell(\eta(\beta, X))}{\partial \eta(\beta, X)}\right|_{\hat{\beta}^{[m-1]}}$$

- gradient boosting is a much more general algorithm;
- likelihood-based boosting and gradient boosting are equal in Gaussian regression because the log-likelihood is a parabola;
- both have a componentwise version.

Likelihood-based Boosting: likelihood-based vs gradient

Alternatively (more correctly) we can see the likelihood-based boosting as a special case of the gradient boosting (De Bin, 2016):

1. initialize $\hat{\beta} = (0, \dots, 0)$;
2. for $m = 1, \dots, m_{\text{stop}}$
    - compute the negative gradient vector, $u = \left.\frac{\partial \ell(f(x,\beta))}{\partial f(x,\beta)}\right|_{\beta=\hat{\beta}}$
    - compute the update,
    $$\hat{b}^{LB} = \left(\left.\frac{\partial f(x,\beta)}{\partial \beta}\right|_{\beta=0}^{\top} u\right) \Big/ \left(-\left.\frac{\partial \frac{\partial f(x,\beta)}{\partial \beta}}{\partial \beta}\right|_{\beta=0}^{\top} u + \lambda\right);$$
    - update the estimate, $\hat{\beta}^{[m]} = \hat{\beta}^{[m-1]} + \hat{b}^{LB}$.
3. compute the final prediction, e.g., for lin. regr. $\hat{y} = X^T \hat{\beta}^{[m_{\text{stop}}]}$

Tree-based boosting: introduction

The base (weak) learner in a boosting algorithm can be a tree:

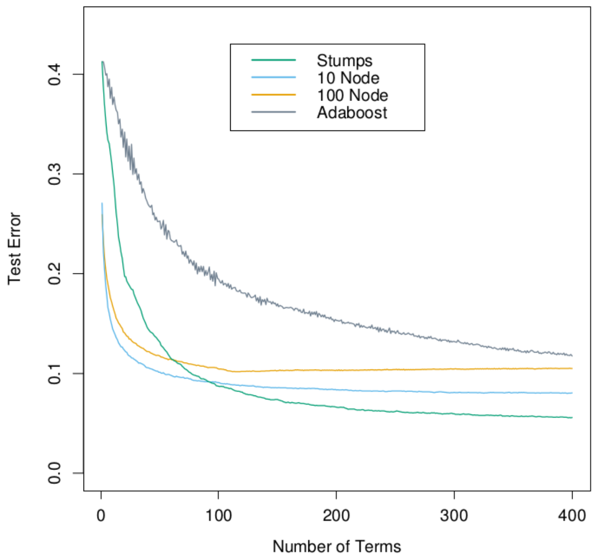
- largely used in practice;
- very powerful and fast algorithm;
- R package `XGBoost`;
- we lose part of the statistical interpretation.

Tree-based boosting: algorithm

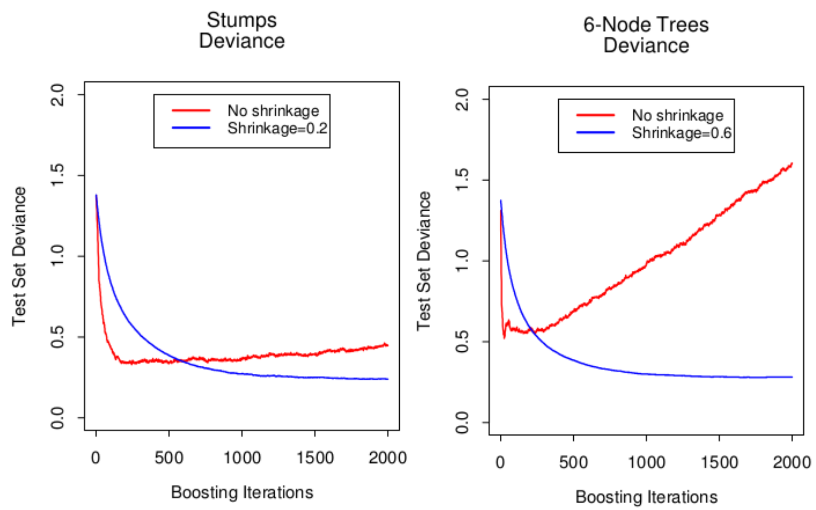
Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(y_i, \gamma)$.
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \dots, N$ compute
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$
   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \dots, J_m$.
   (c) For $j = 1, 2, \dots, J_m$ compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.
3. Output $\hat{f}(x) = f_M(x)$.

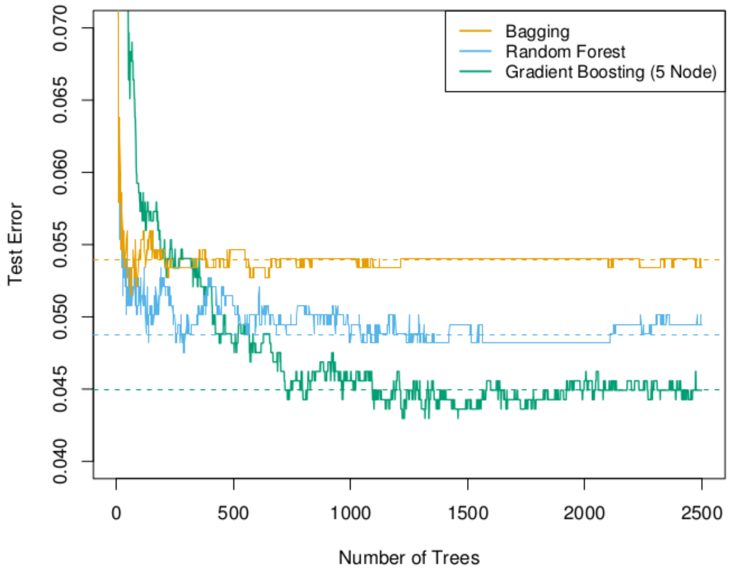
Tree-based boosting: importance of "weakness"



Tree-based boosting: importance of "shrinkage"



Tree-based boosting: comparison



References I

Bühlmann, P. & Yu, B. (2003). Boosting with the L_2 loss: regression and classification. *Journal of the American Statistical Association* **98**, 324–339.

De Bin, R. (2016). Boosting in Cox regression: a comparison between the likelihood-based and the model-based approaches with focus on the R-packages CoxBoost and mboost. *Computational Statistics* **31**, 513–531.

Seibold, H., Bernau, C., Boulesteix, A.-L. & De Bin, R. (2016). On the choice and influence of the number of boosting steps. Tech. Rep. 188, University of Munich.

Tutz, G. & Binder, H. (2006). Generalized additive modeling with implicit variable selection by likelihood-based boosting. *Biometrics* **62**, 961–971.

Tutz, G. & Binder, H. (2007). Boosting ridge regression. *Computational Statistics & Data Analysis* **51**, 6044–6059.