Boosting: more than an ensemble method for prediction

Anestis Antoniadis

Historically: Boosting is about multiple predictions

Data: $(X_1, Y_1), \ldots, (X_n, Y_n)$ (i.i.d. or stationary),
predictor variables $X_i \in \mathbb{R}^p$
response variables $Y_i \in \mathbb{R}$ or $Y_i \in \{0, 1, \ldots, J-1\}$

Aim: estimation of function $f(\cdot) : \mathbb{R}^p \to \mathbb{R}$, e.g.
$f(x) = \mathbf{E}[Y|X = x]$ or $f(x) = \mathbb{P}[Y = 1|X = x]$ with $Y \in \{0, 1\}$
or distribution of survival time Y given X depends on some function $f(X)$ only

"historical" view (for classification):

Boosting is a multiple predictions (estimation) & combination method

Base procedure:

$$\text{data} \quad \xrightarrow{\text{algorithm A}} \quad \hat{\theta}(\cdot) \text{ (a function estimate)}$$

e.g.: simple linear regression, tree, MARS, "classical" smoothing, neural nets, ...

Generating multiple predictions:

$$\text{weighted data 1} \quad \xrightarrow{\text{algorithm A}} \quad \hat{\theta}_1(\cdot)$$

$$\text{weighted data 2} \quad \xrightarrow{\text{algorithm A}} \quad \hat{\theta}_2(\cdot)$$

$$\ldots \qquad\qquad \ldots$$

$$\text{weighted data M} \quad \xrightarrow{\text{algorithm A}} \quad \hat{\theta}_M(\cdot)$$

Aggregation: $\hat{f}_A(\cdot) = \sum_{m=1}^{M} a_m \hat{\theta}_m(\cdot)$

data weights? averaging weights a_m?

classification of 2 lymph nodal status in breast cancer using gene expressions from microarray data:

$n = 33$, $p = 7129$ (for CART: gene-preselection, reducing to $p = 50$)

method	test set error	gain over CART
CART	22.5%	–
LogitBoost with trees	16.3%	28%
LogitBoost with bagged trees	12.2%	46%

this kind of boosting: mainly prediction, not much interpretation

Boosting algorithms

AdaBoost proposed for classification by Freund & Schapire (1996)

data weights (rough original idea): large weights to previously heavily misclassified instances (sequential algorithm)

averaging weights a_m: large if in-sample performance in mth round was good

Why should this be good?

Why should this be good?

some common answers 5 years ago ...

because

- it works so well for prediction (which is quite true)
- it concentrates on the “hard cases” (so what?)
- AdaBoost almost never overfits the data no matter how many iterations it is run (not true)

A better explanation

Breiman (1998/99): AdaBoost is functional gradient descent (FGD) procedure

aim: find $f^*(\cdot) = \text{argmin}_{f(\cdot)} \mathbb{E}[\rho(Y, f(X))]$

e.g. for $\rho(y, f) = |y - f|^2 \;\rightsquigarrow\; f^*(x) = \mathbb{E}[Y|X = x]$

FGD solution: consider empirical risk $n^{-1} \sum_{i=1}^{n} \rho(Y_i, f(X_i))$ and

do iterative steepest descent in function space

Generic FGD algorithm

Step 1. $\hat{f}_0 \equiv 0$; set $m = 0$.

Step 2. Increase m by 1. Compute negative gradient $-\frac{\partial}{\partial f}\rho(Y, f)$ and evaluate at $f = \hat{f}_{m-1}(X_i) = U_i$ $(i = 1, \ldots, n)$

Step 3. Fit negative gradient vector $U_1, \ldots, U_n$ by base procedure

$$(X_i, U_i)_{i=1}^n \xrightarrow{\text{algorithm A}} \hat{\theta}_m(\cdot)$$

e.g. $\hat{\theta}_m$ fitted by (weighted) least squares
i.e. $\hat{\theta}_m(\cdot)$ is an approximation of the negative gradient vector

Step 4. Up-date $\hat{f}_m = \hat{f}_{m-1}(\cdot) + \nu s_m \cdot \hat{\theta}_m(\cdot)$
$s_m = \text{argmin}_s n^{-1} \sum_{i=1}^n \rho(Y_i, \hat{f}_{m-1}(X_i) + s \cdot \hat{\theta}_m(X_i))$ and $0 < \nu \leq 1$
i.e. proceed along an estimate of the negative gradient vector

Step 5. Iterate Steps 2-4 until $m = m_{stop}$ for some stopping iteration m_{stop}

Why "functional gradient"?

Alternative formulation in function space:

empirical risk functional: $C(f) = n^{-1} \sum_{i=1}^{n} \rho(Y_i, f(X_i))$

inner product: $\langle f, g \rangle = n^{-1} \sum_{i=1}^{n} f(X_i) g(X_i)$

negative Gateaux derivative:

$$-dC(f)(x) = \frac{\partial}{\partial \alpha} C(f + \alpha 1_x)|_{\alpha=0}, \rightsquigarrow -dC(\hat{f}_{m-1})(X_i) = U_i$$

if $U_1, ..., U_n$ are fitted by least squares:

equivalent to maximize $\langle -dC(f_m), \theta \rangle$ w.r.t. $\theta(\cdot)$ (if$\|\theta\| = 1)$

(over all possible $\theta(\cdot)$'s from the base procedure)

i.e: $\hat{\theta}_m(\cdot)$ is the best approximation (most parallel)

to the negative gradient $-dC(f_m)$

By definition: FGD yields additive combination of base procedure fits

$$\nu \sum_{m=1}^{m_{stop}} s_m \hat{\theta}_m(\cdot)$$

Breiman (1998):

FGD with $\rho(y, f) = \exp((2y - 1) \cdot f)$ for binary classification yields the

AdaBoost algorithm

(great result!)

Remark: FGD can not be represented as some explicit estimation function(al):

$$\hat{f}_m(\cdot) \neq \text{argmin}_{f \in \mathcal{F}} n^{-1} \sum_{i=1}^{n} \rho(Y_i, f(X_i)) \quad \text{for some function class } \mathcal{F}$$

$\rightsquigarrow$ FGD is mathematically more difficult to analyze but

generically applicable (as an algorithm!) in very complex models

L_2Boosting

(see also Friedman, 2001)

loss function $\rho(y, f) = |y - f|^2$

population minimizer: $f^*(x) = \mathbf{E}[Y|X = x]$

FGD with base procedure $\hat{\theta}(\cdot)$: repeated fitting of residuals

$$m = 1: \ (X_i, Y_i)_{i=1}^n \rightsquigarrow \hat{\theta}_1(\cdot),\ \hat{f}_1 = \nu\hat{\theta}_1 \quad \rightsquigarrow \text{resid. } U_i = Y_i - \hat{f}_1(X_i)$$

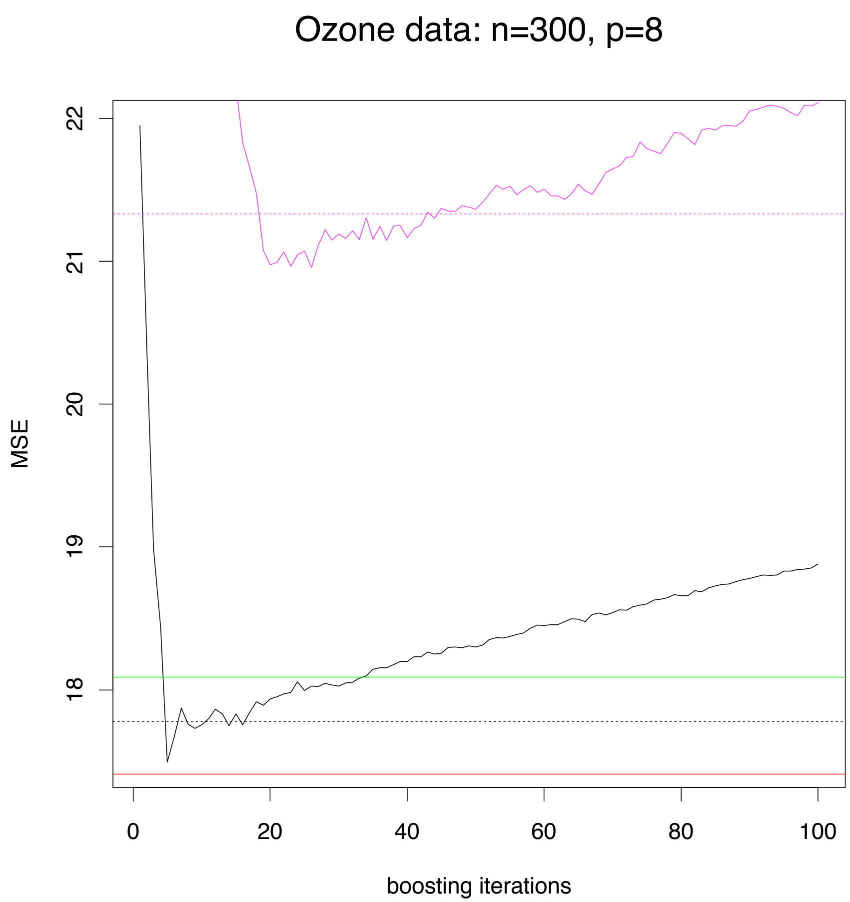
$$m = 2: \ (X_i, U_i)_{i=1}^n \rightsquigarrow \hat{\theta}_2(\cdot),\ \hat{f}_2 = \hat{f}_1 + \nu\hat{\theta}_2 \quad \rightsquigarrow \text{resid. } U_i = Y_i - \hat{f}_2(X_i)$$

$$\ldots \qquad \ldots$$

$\hat{f}_{m_{stop}}(\cdot) = \nu \sum_{m=1}^{m_{stop}} \hat{\theta}_m(\cdot)$ (stagewise greedy fitting of residuals)

Tukey (1977): twicing for $m_{stop} = 2$ and $\nu = 1$

Any gain over classical methods? (for additive modeling)



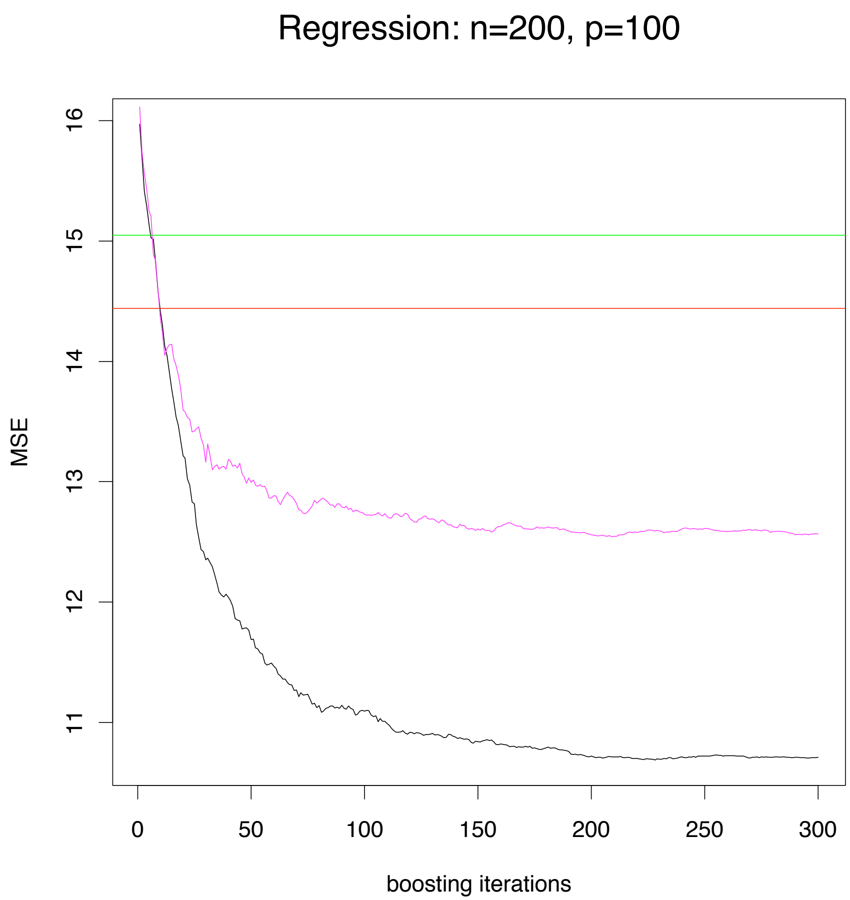
$n = 300,\ p = 8$

- magenta: L_2Boosting with stumps
  (horiz. line = cross-validated stopping)
- black: L_2Boosting with componentwise smoothing spline
  (horiz. line = cross-validated stopping)
  i.e: smoothing spline fitting against the selected predictor which reduces RSS most
- green: MARS restricted to additive modeling
- red: additive model using backfitting

L_2Boosting with stumps or comp. smoothing splines also yields additive model:

$$\sum_{m=0}^{m_stop} \hat{\theta}_m(x^{(\hat{S}_m)}) = \hat{g}_1(x^{(1)}) + \ldots + \hat{g}_p(x^{(p)})$$

Simulated data: non-additive regression function, $n = 200, p = 100$



- magenta: L_2Boosting with stumps

- black: L_2Boosting with componentwise

- green: MARS restricted to additive modeling

- red: additive model using backfi tting and fwd. var. selection similar for classification

Boosting for binary classification

binary lymph node classification using gene expressions: data
$(X_i, Y_i),\ X_i \in \mathbb{R}^{7129},\ Y_i \in \{-1, 1\}$

Various loss functions

$\rho(y, f) = \log_2(1 + \exp(-yf))$: negative binomial log-likelihood
$f^*(x) = \log(\frac{p(x)}{1-p(x)})$

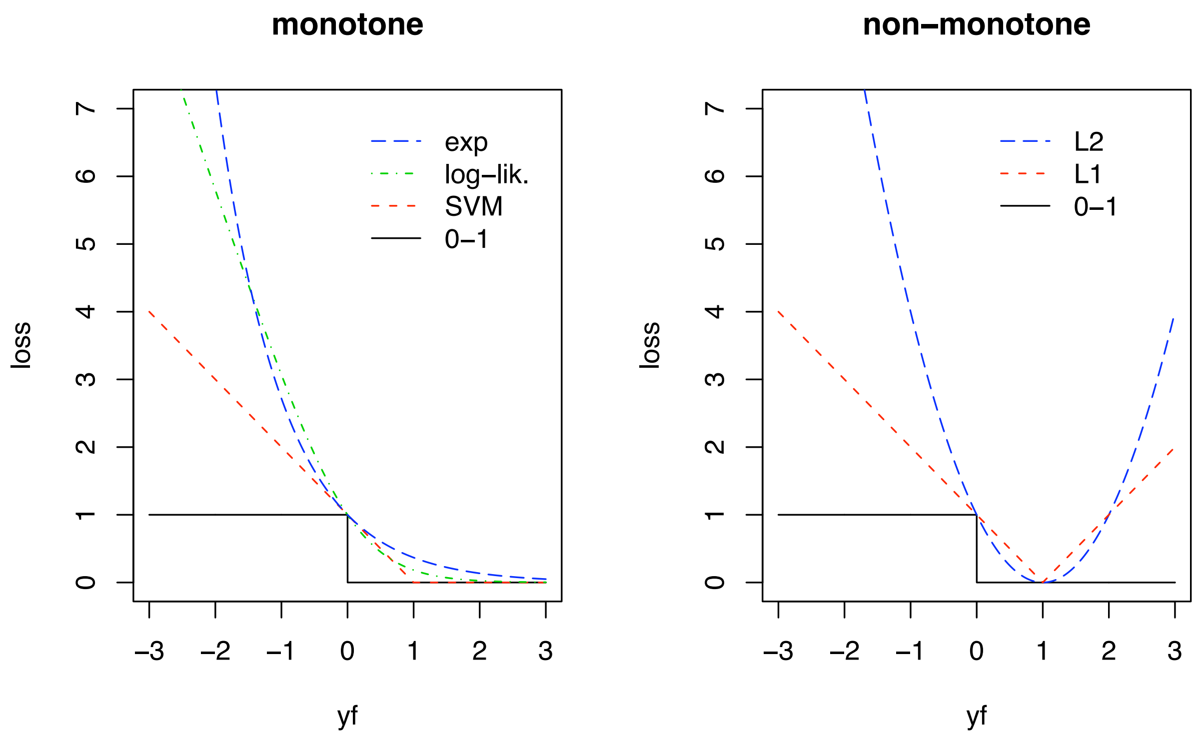
$\rho(y, f) = |y - f|^2 = 1 - 2yf + (yf)^2$: squared error
$f^*(x) = \mathbf{E}[Y|X = x] = 2p(x) - 1$

$\rho(y, f) = \exp(-yf)$: exponential loss in AdaBoost
$f^*(x) = \frac{1}{2}\log(\frac{p(x)}{1-p(x)})$

$\rho(y, f) = \mathbb{I}_{[yf<0]}$: misclassification loss
$f^*(x) = \mathbb{I}_{[p(x) \geq 1/2]}$

all these loss functions: $\rho(y, f) = \rho(yf)$:

function of the margin value yf



minimization of the non-convex misclassification loss: computationally infeasible

other loss functions: convex surrogate loss functions, dominating misclass. error

Conclusions

statistical view of boosting:

a regularization method for estimation and variable selection

mainly useful for high-dimensional data problems

- boosting is very generic
- boosting is computationally attractive: complexity $O(p)$ for $p \gg n$
- simple statistical inference is possible, but more needs to be done