MA8701 Advanced methods in statistical inference and learning

Part 3: Ensembles. L14: Boosting

Mette Langaas

~~2/25/23~~

Lectured 27.02.2023

Added after class

Part 3: plan

Wisdom of the crowds

Bagging | Trees | Random forest — L13

Boosting — L14 + video

Stacked ensembles | Hyperparameter tuning — L15 + L16

Evaluating and comparing results from prediction models — L17

Literature

- [ESL] The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition (Springer Series in Statistics, 2009) by Trevor Hastie, Robert Tibshirani, and Jerome Friedman. Ebook. Chapter 10.1-10.6, 10.9-10.10, 10.12, 10.13 (in Part 4).
- Video by Berent Lunde (link on Bb), covering Chapter 10 (in particular 10.10) and the Chen and Guestrin paper.
- Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785. The mathematical notation is not in focus

BOOSTING

slightly better than random guessing

Combine many weak learners (classification/regression) to produce a powerful ensemble (committee)

Differences to bagging/random forest

- no bootstrapping
- sequential training: outcome of learner 1 decides the problem to give to learner 2, etc

AdaBoost.M1

(Freund & Schapire 1997)

$X \in \mathbb{R}^p$

$Y \in \{-1, 1\}$

$G(X) \in \{-1, 1\}$

final classifier

$E_{XY}(I(Y \neq G(X)))$

$\overline{err} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i))$

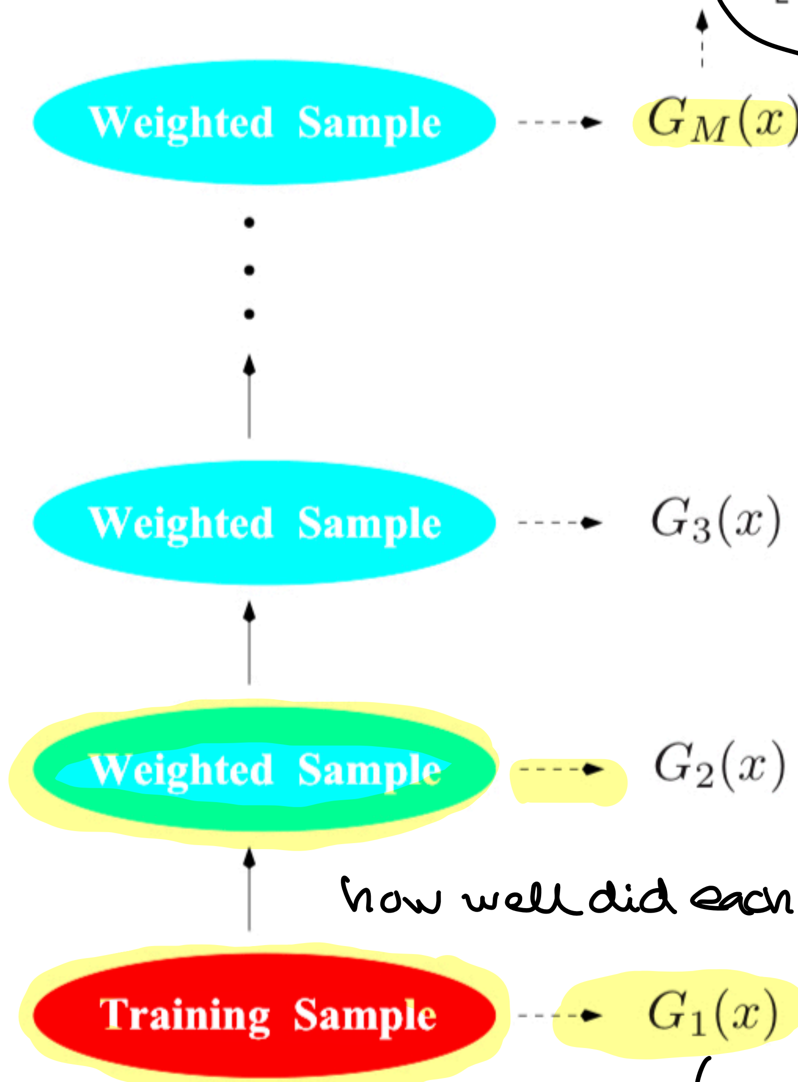
validation set

the final classifier

weighted sum of weak learner

$$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

classify to −1 if sum < 0, else 1



how well did each observation do?

the first weak learner

{−1, 1}

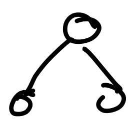
FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Figure 1: Hastie, Tibshirani, and Friedman (2009) Figure 10.1

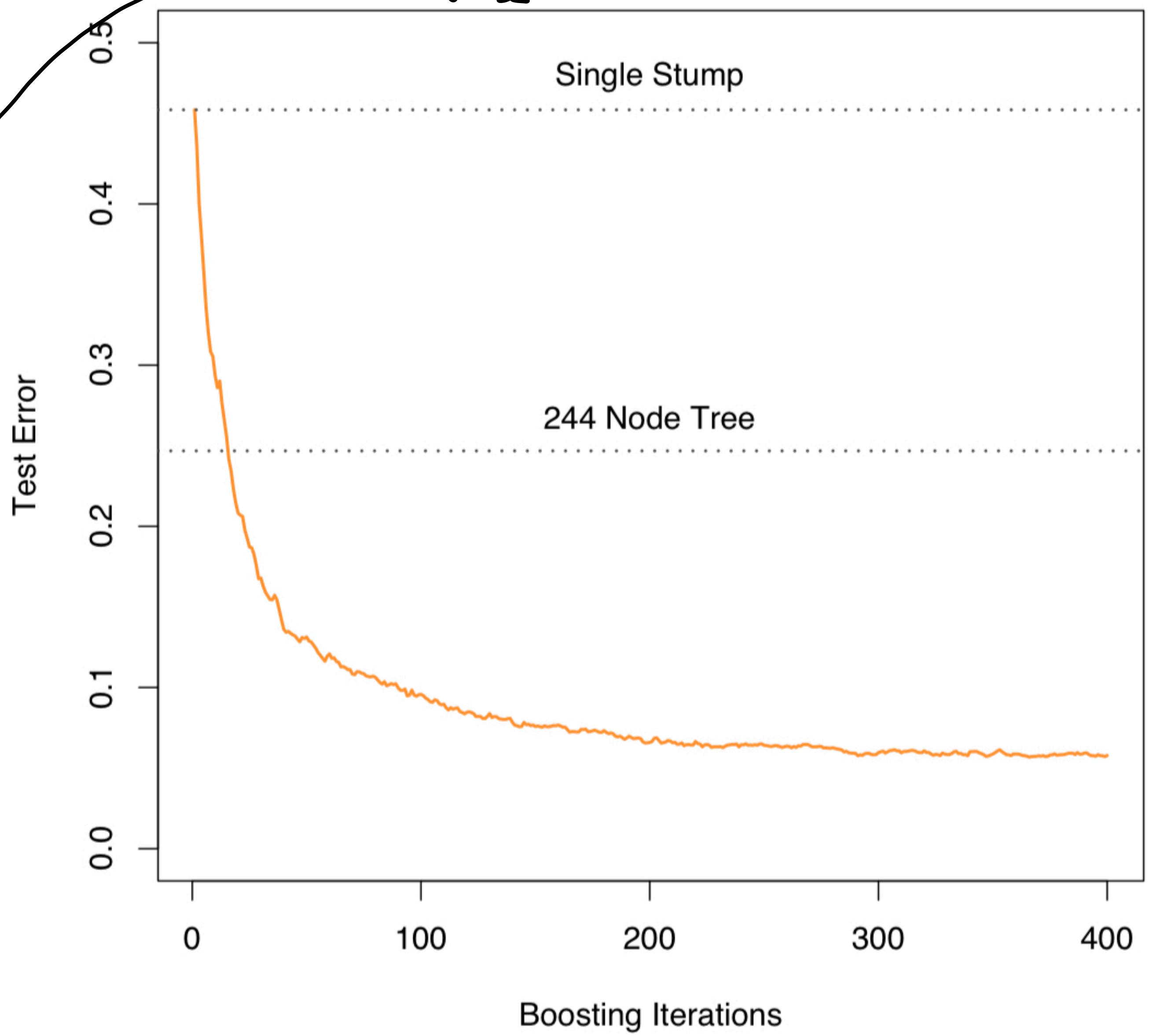
Example 10.2

$N = 1000 + 1000$, $N_{test} = 10000$. Xs (independent) from N(0,1) for $p = 10$ and true class is 1 if $\underbrace{\sum_{j=1}^{10} x_j^2 > 9.34}_{\text{class 1}}$ (median chisq), and $\underbrace{-1 \text{ else}}_{\text{class } -1}$.

weak learner



NB: additive model in x_j^2



Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$. equal weight on all obs.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i. stump Gini index
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}. \quad \in [0,1]$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Figure 3: Hastie, Tibshirani, and Friedman (2009) Algorithm 10.1

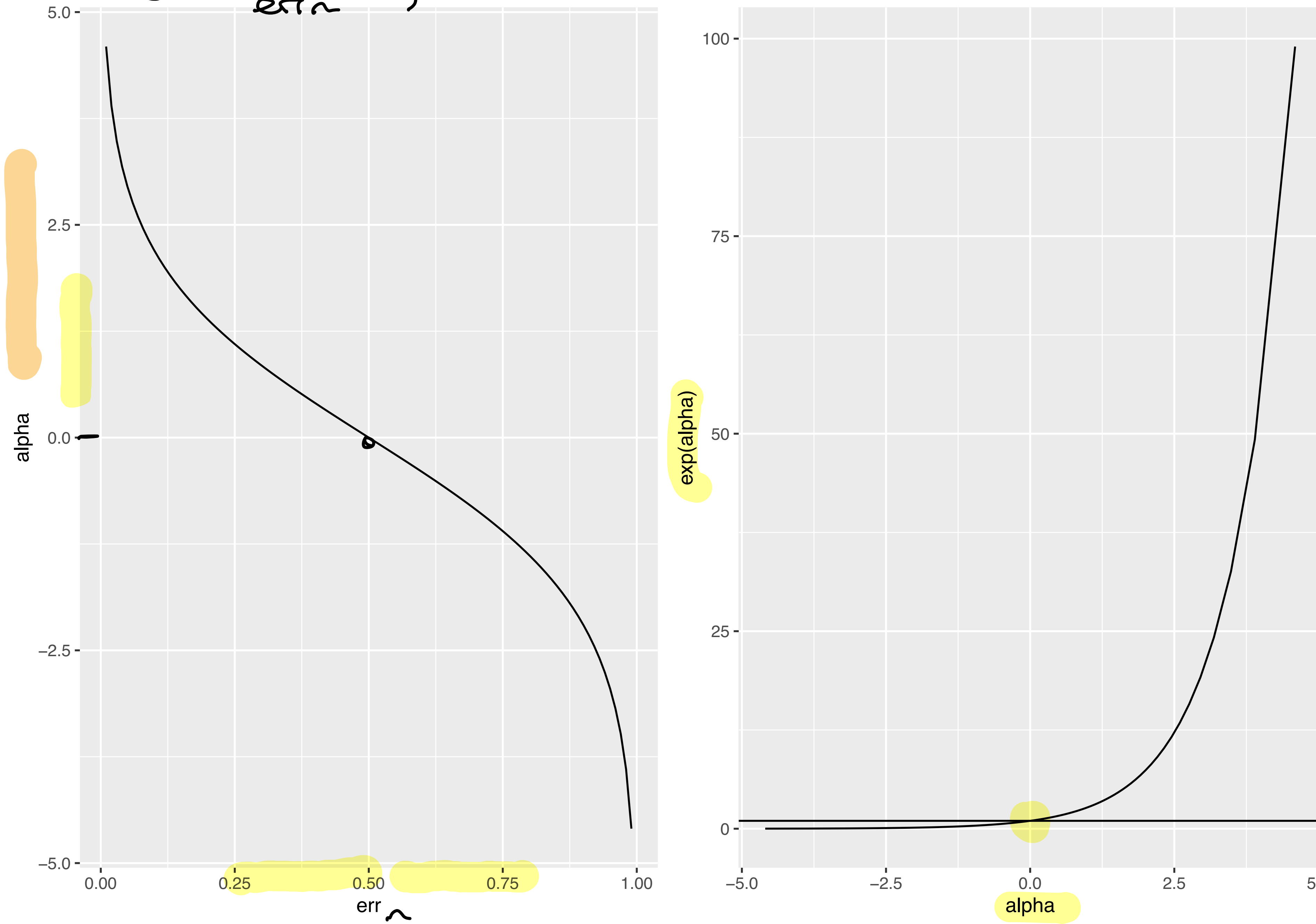
2b) err_m = weighted miscl. rate

2c) $err_m = \frac{1}{2} \Rightarrow \alpha_m = 0$, $err_m > \frac{1}{2}$ — new weight dist. pn

2d) change of sample weights :

$$y_i = G_m(x_i) \quad w_i \cdot \exp(0)$$
$$\neq \quad w_i \cdot \exp(\alpha_m)$$

$$\log\left(\frac{1-err_m}{err_m}\right)$$



Understanding AdaBoost.M1

AIM: Why does this work and how to develop further.

Additive models + Loss function $\rightarrow$ Minimize loss in a stagewise manner

$G(x)$ is an additive model

$G_m(x)$ elementary basis function and Adaboost fit an

additive set of these

$$f(x) = \sum_{m=1}^{M} \overbrace{\beta_m}^{\alpha_m} \cdot \overbrace{b(x; \gamma_m)}^{G_m(x)}$$

could be (in the linear prediction GLM → GAM

expansion coeff

single trees with paramel (j, s)

variable split

Additive models are known outside boosting and is fit

by minimizing some loss L over the trainingset

$$\min_{\beta_m, \gamma_m} \sum_{i=1}^{N} L\left(y_i, \sum_{m=1}^{B} \beta_m b(x_i; \gamma_m)\right)$$

Algorithm 10.2 *Forward Stagewise Additive Modeling.*

1. Initialize $f_0(x) = 0$. ← empty model
2. For $m = 1$ to M:
   (a) Compute

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \overbrace{\beta b(x_i; \gamma)}^{\text{new}}).$$

known
do not modify

   (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Figure 4: Hastie, Tibshirani, and Friedman (2009) Algorithm 10.2

stagewise ⇔ stepwise

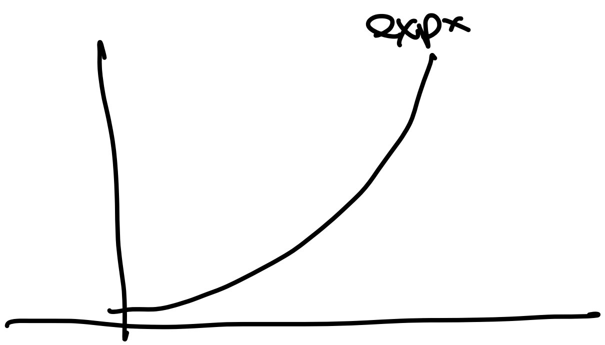
no reestimation ↑ reestimate previous step

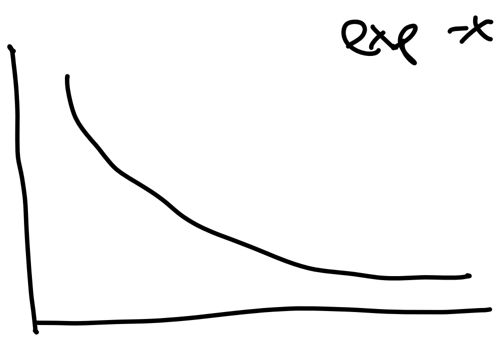
Forward Stagewise Additive Modelling with Exponential loss = AdaBoost.M1

$$L(y, f(x)) = \exp(-y\, f(x))$$

$e^{f(x)}$ if $y = -1$

$e^{-f(x)}$ $y = 1$





LONG straightforward derivation → scroll through

Link on webpage for this AdaboostExploss

THE steps of the derivation

1) $(\beta_m, b_m) = \arg\min \sum_{i=1}^{N} w_i^{(m-1)} \cdot \exp(-y_i \beta b(x_i))$

$$w_i^{(m-1)} \uparrow \exp(-y_i \cdot \hat{f}^{(m-1)}(x_i))$$

2) $b_m : \arg\min_{b} \left\{ \sum_{i=1}^{N} w_i^{(m-1)} I(y_i \neq b(x_i)) \right\}$

minimize the weighted miscl. rate

3) $\hat{\beta}_m = \frac{1}{2} \ln\left(\frac{1-err_m}{err_m} \right)$

$\hat{\alpha}_m = 2\beta_m$

4) $w_i^{(m)} = w_i^{(m-1)} \cdot \exp(\hat{\alpha}_m I(y_i \neq b_m(x_i))) \cdot \exp\left(-\frac{\hat{\alpha}_m}{2}\right)$

scaling

Group discussion:

- We showed
  look at this derivation of the equivalence of the AdaBoost.M1 and the forward stagewise modelling with exponential loss.
- For the steps 2a-2d in Algorithm 10.1 what is your new insight into what is done at each step?

Algorithm 10.1 *AdaBoost.M1.*

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:

$\hat{b}_m = \underset{b}{\operatorname{argmin}} \left(\sum w_i^{(m-1)} I(y_i \neq b(x_i)) \right)$

   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

   (b) Compute

$$\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$

intermediate

   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

solution to $2 \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m-1)} \exp(-y_i \beta b(x_i))$

   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.

ease notation

⇓

stagewise

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.

$\beta_m \; b_m(x)$

Figure 3: Hastie, Tibshirani, and Friedman (2009) Algorithm 10.1

What is great with exponential loss?

1) computational: easy weight scheme

2) $L(y, f(x)) = \exp(-y f(x))$ is not a neg loglikelihood, but

$$\hat{f}(x) = \underset{f(x)}{\arg\min}\, E_{Y|X}\left(e^{-Yf(x)}\right) \overset{*}{=} \underbrace{\frac{1}{2} \ln \frac{P(Y=1|x)}{P(Y=-1|x)}}_{\text{log odds}}$$

"population minimizer"

Exercise! ESL Ex 10.2 to derive this (link handouts)

It can be shown that the population minimizer of negative binomial deviance can be written as

$$E\left(\ln\left(1+e^{-2Yf(x)}\right)\right)$$ both have the minimizer → *

But for finite data sets: give different results, although both functions of $y \cdot f(x)$

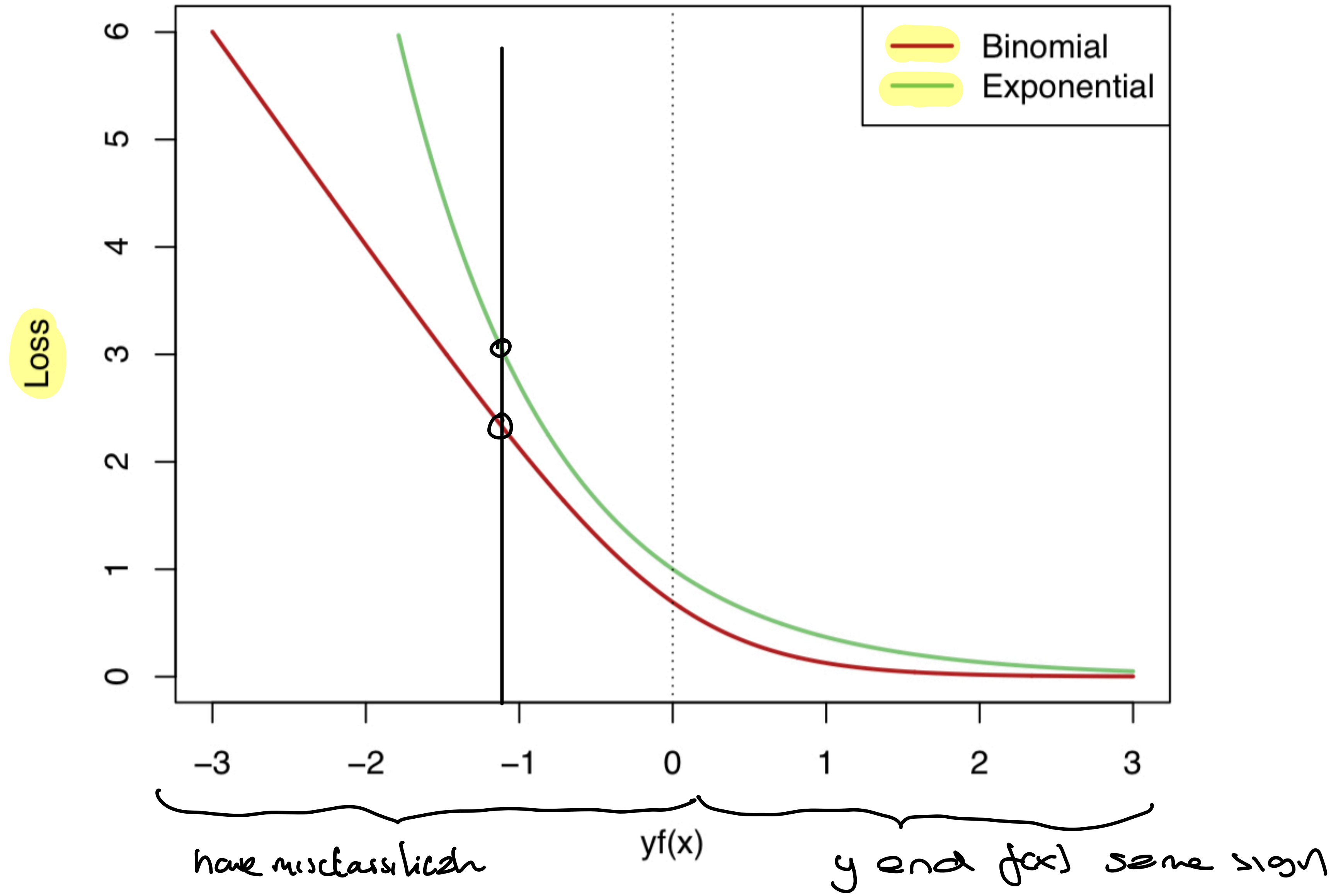


Figure 17.12 Exponential loss used in Adaboost, versus the binomial loss used in the usual logistic regression. Both estimate the logit function. The exponential left tail, which punishes misclassifications, is much more severe than the asymptotically linear tail of the binomial.

Figure 5: Efron and Hastie (2016) Figure 17.10: Importance of learning

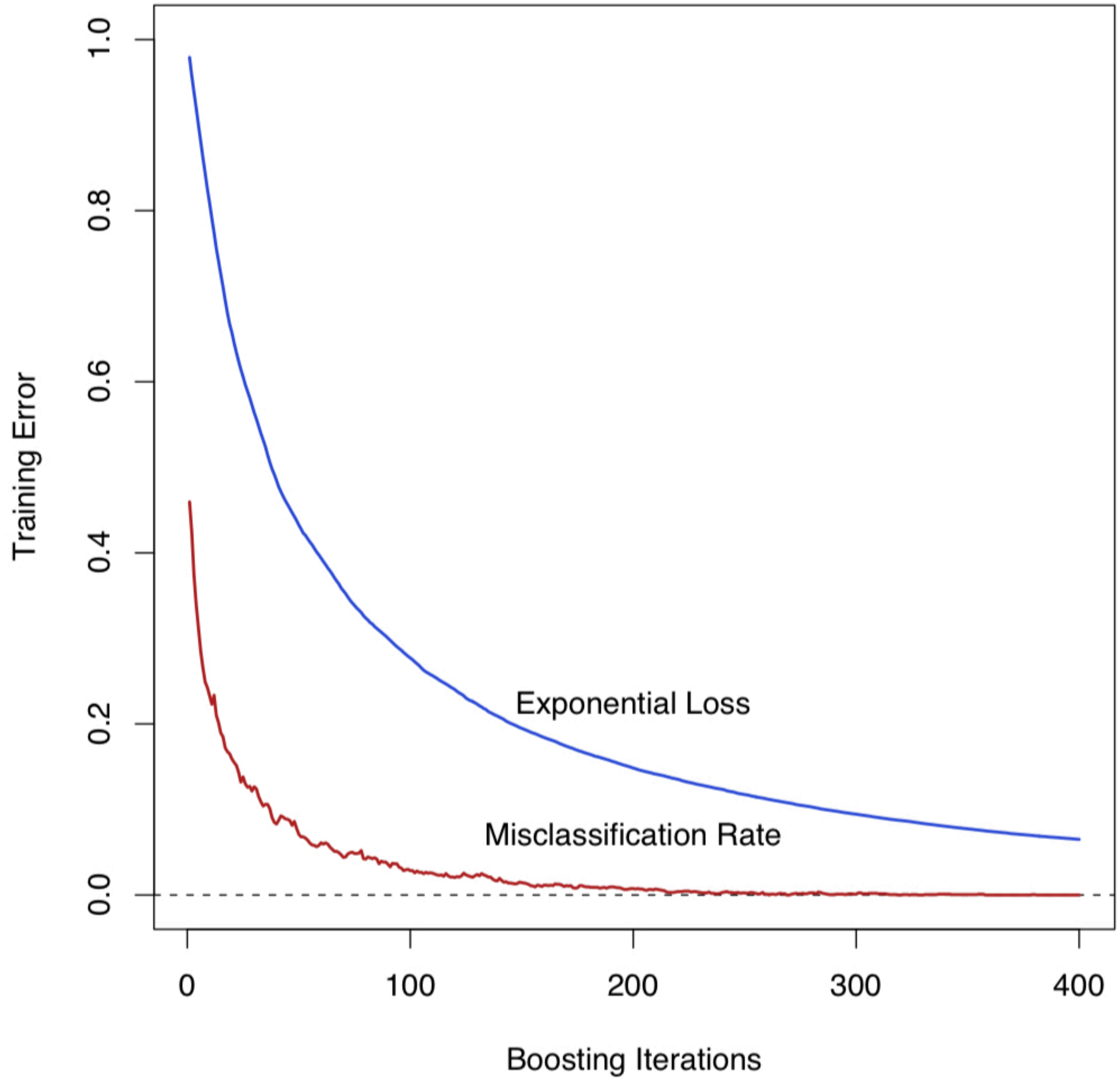


FIGURE 10.3. *Simulated data, boosting with stumps: misclassification error rate on the training set, and average exponential loss:* $(1/N)\sum_{i=1}^{N}\exp(-y_i f(x_i))$. *After about 250 iterations, the misclassification error is zero, while the exponential loss continues to decrease.*

Based on the insight gained from AdaBoost Friedman in 2000 presented "gradient boosting" – we will look at trees as the base learners.

Estimation of some function f can be done by minimizing a differentiable loss (and also convex) by applying "functional gradient descent". The function f is an additive expansion of base learners, and each step (new learner) goes in the direction of the negative gradient of the loss function at the current function estimate

Gradient boosting

Regression

WHAT if $L(y,f) = \frac{1}{2}(y-f)^2$?

Algorithm 10.3 *Gradient Tree Boosting Algorithm.*

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^N L(y_i, \gamma)$. $= \text{ave}(y)$

2. For $m = 1$ to M:

   (a) For $i = 1, 2, \ldots, N$ compute $\quad -\frac{\partial L}{\partial f} \frac{1}{2}(y-f)^2 = \overbrace{(y-f)}^{\text{residual}}$

   prev. val

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

negative gradient

   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.

   region

   (c) For $j = 1, 2, \ldots, J_m$ compute

$$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

value in leafs

   (d) Update $f_m(x) = f_{m-1}(x) + \underbrace{\sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})}_{\text{tree}}$.

3. Output $\hat{f}(x) = f_M(x)$.

ν

Figure 8: Hastie, Tibshirani, and Friedman (2009) Algorithm 10.3

But, there is no weighing of the observations! Now instead we work with the value of the negative gradient at the last step of the algo - for each obs r_{im}

obs step

For squared loss - we fit the new base learner to the sum of the residuals.

pseudo

This method is called first order GTB - and second order involves the Hessian (done xgboost) — watch Berent video.

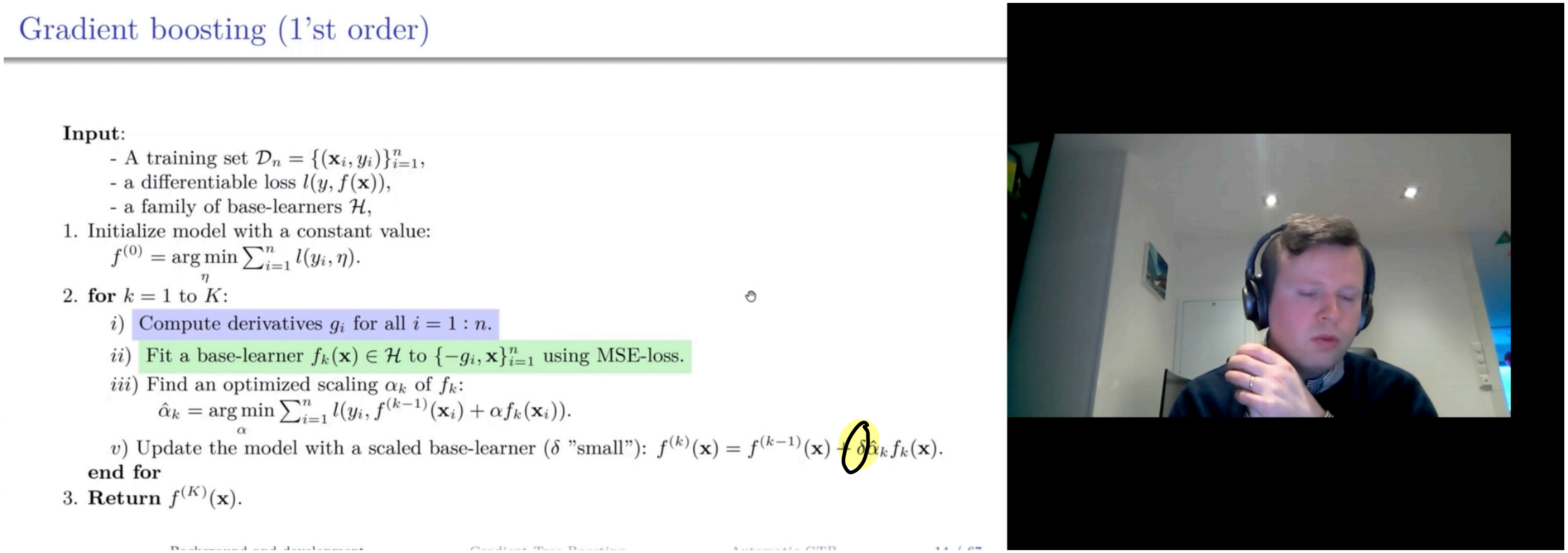
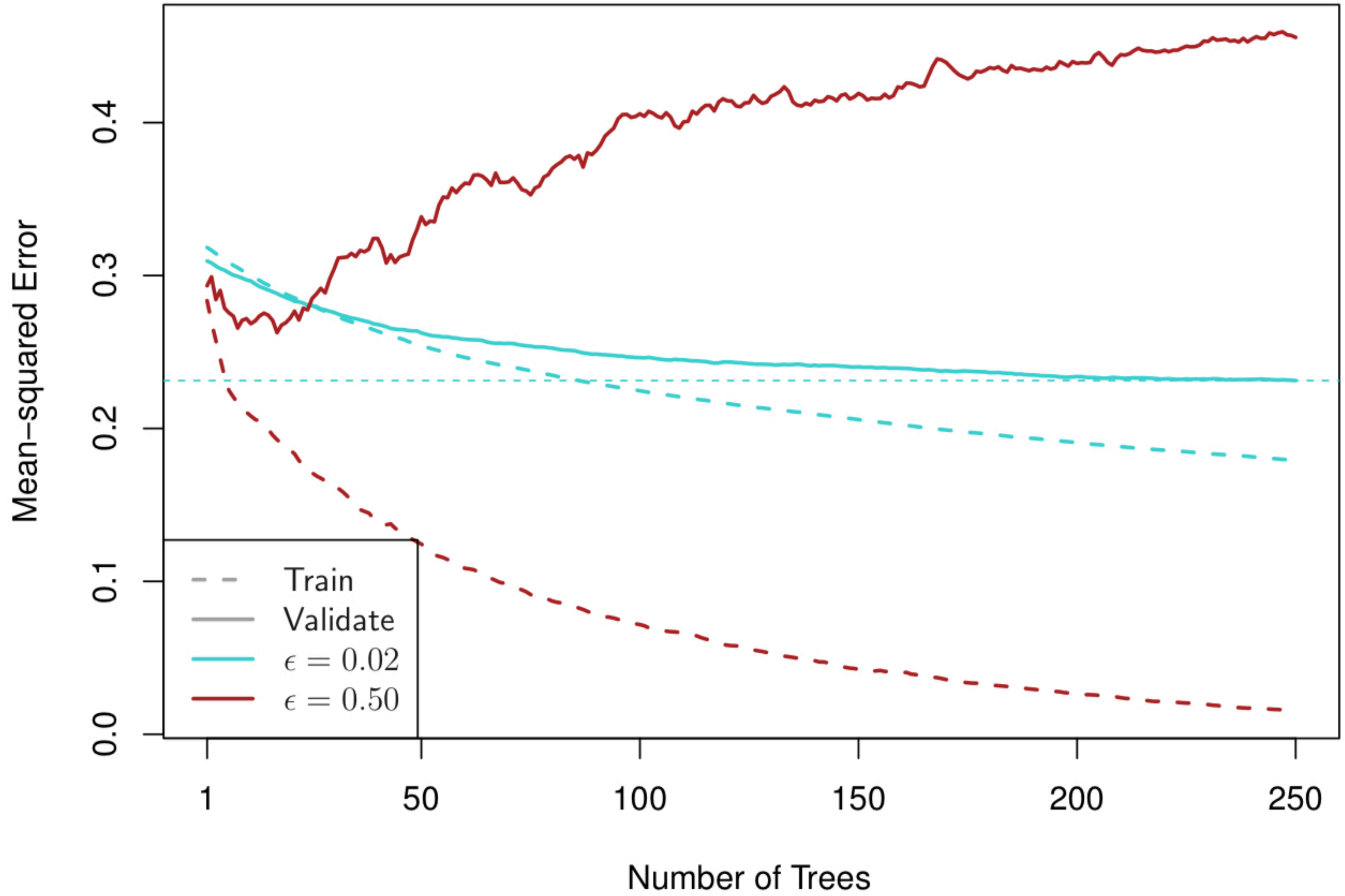


Figure 9: Guest lecture by Berent (at 25 minutes in the video)
into

Comments from Berent: essential to add an extra learning rate δ between 0 and 1 and $\delta = 0.05$ not uncommon. In Hastie, Tibshirani, and Friedman (2009) 10.12.1 Equation (10.41).

$\parallel$

called ν

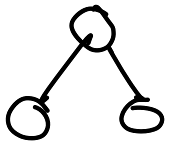


CASi book Efron & Hastie

Figure 17.10 Boosted $d = 3$ models with different shrinkage parameters, fit to a subset of the ALS data. The solid curves are validation errors, the dashed curves training errors, with red for $\epsilon = 0.5$ and blue for $\epsilon = 0.02$. With $\epsilon = 0.5$, the training error drops rapidly with the number of trees, but the validation error starts to increase rapidly after an initial decrease. With $\epsilon = 0.02$ (25 times smaller), the training error drops more slowly. The validation error also drops more slowly, but reaches a lower minimum (the horizontal dotted line) than the $\epsilon = 0.5$ case. In this case, the slower learning has paid off.

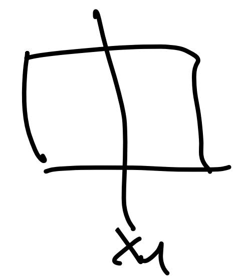
Tree depth

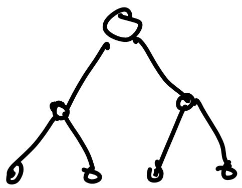
is a measure of the complexity of the trees (used in the GTB)



depth 1 ← only split on one variable

$J = 2$ → main effect of x_1





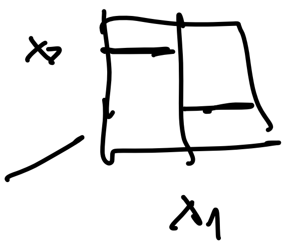
depth 2

$J = 3$

4 nodes

can model $x_1 + x_2 + x_1 \cdot x_2$

effect of x_2 depends on value of x_1 called interaction

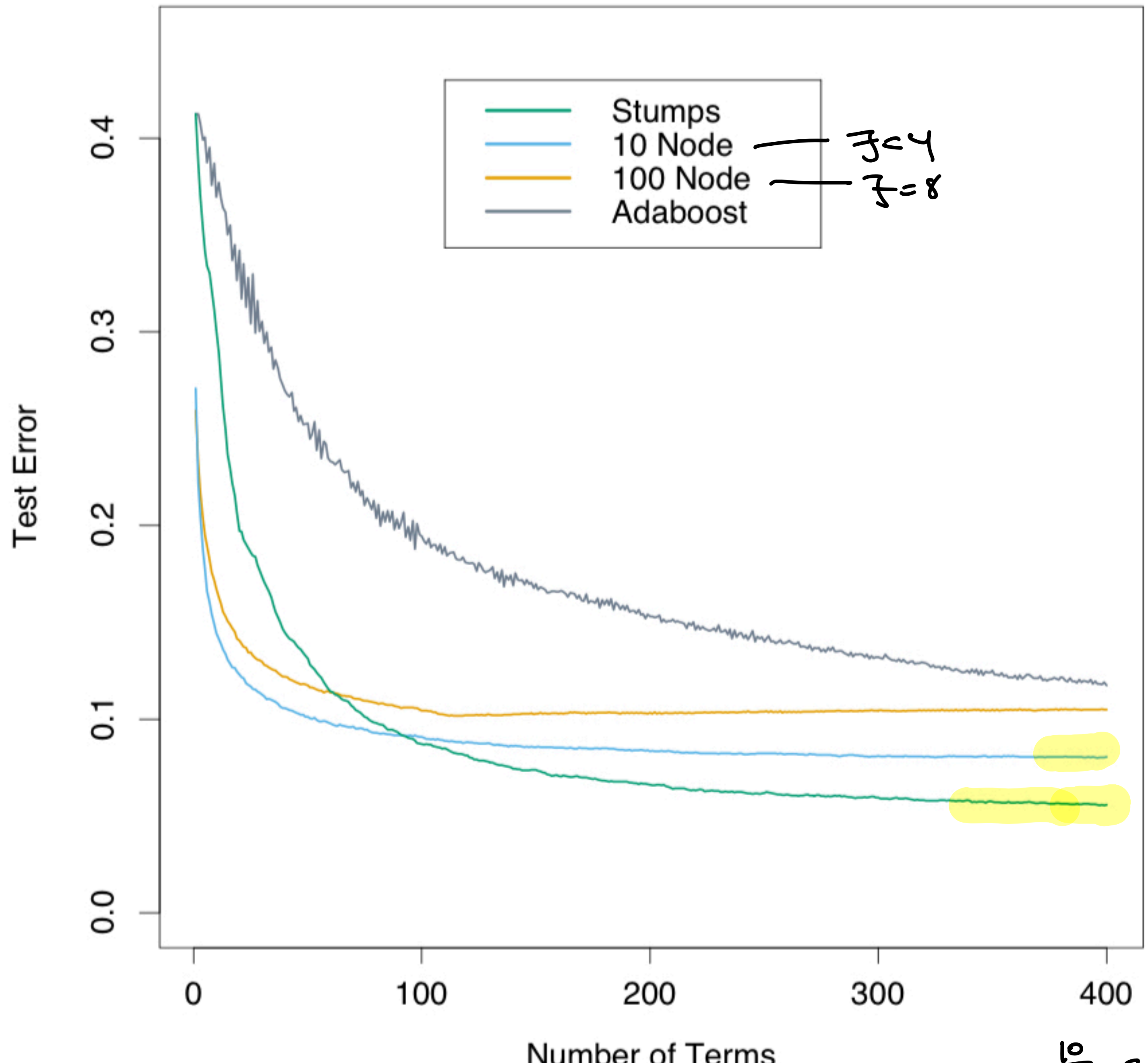


Common strategy: all trees same size J

Since low order models (remember TMA4267 DOE project) models seem to dominate $4 \le J \le 8$ often and even smaller

$J = 4$ (8 nodes) $J = 5$ (16 nodes) $J = 6$ (32 nodes) $J = 7$ (64 nodes)

$J = 8$ (128 nodes)



J=4
J=8

J=8
J=4
J=2

$\sum^{10} x_j^2$ - problem

FIGURE 10.9. *Boosting with different sized trees, applied to the example (10.2) used in Figure 10.2. Since the generative model is additive, stumps perform the best. The boosting algorithm used the binomial deviance loss in Algorithm 10.3;*

Regularization

(10.12)

- The number of weak learners, M, is chosen by monitoring prediction risk on a validation sample (same as early stopping in Deep nets - stop training when error validation set increases).

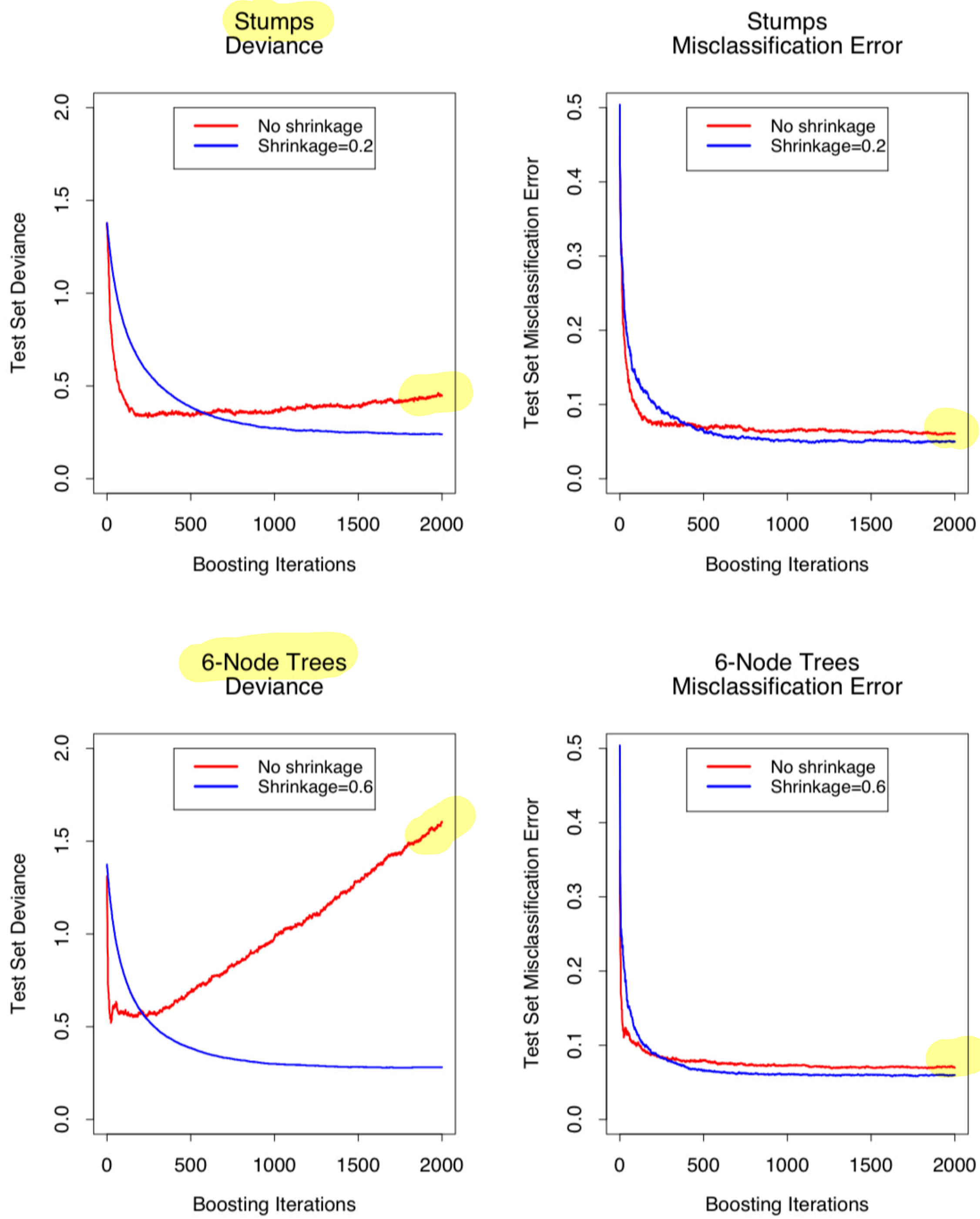
  $f_m(x) = f_{m-1}(x) + \nu \cdot \text{tree}$

- Learning rate - low rate generally recommended, but may lead to M then being large. (2d in Algo 10.3 add ν.)

  Shrinkage on next page

- Decorrelated functions: subsampling of both observations (rows) and variables (columns). Same motivation as for random forest. When subsampl observations this is also called stochastic gradient boosting.

- L1 and L2 regularization term can be added (more in 16.2)



shrinkage ν
aka learning rate
is important

FIGURE 10.11. *Test error curves for simulated example (10.2) of Figure 10.9, using gradient boosting (MART). The models were trained using binomial deviance, either stumps or six terminal-node trees, and with or without shrinkage. The left panels report test deviance, while the right panels show misclassification error. The beneficial effect of shrinkage can be seen in all cases, especially for deviance in the left panels.*

Video by Berent — part 1

Replaces lecture
03.07.2023

01:40 Berent starts - with motivation
11:45: Boosting timeline
16:27: Boosting principle
18:42: AdaBoost
22:45: From AdaBoost to gradient boosting
31:26: Relationship to L1 regularization
34:39: Techniques for improvement
End of first part

Video by Berent — part 2

38:10 Gradient Tree Boosting 1st
39:15: Why does trees work
43:49: 2nd order GTB
52:17: Algorithm for 2nd order GTB
55:07: Loss vs complexity trade-off in GTB
56:05: XGBoost
1:02 XGBoost regularization
1:03 Hyperparameter tuning
1:10: Other GTB implmentations (LightGBM, CatBoost, NGBoost)
End of part two

+ article Chen & Guestrin

Video by Berent — part 3

1:20 Answer questions
1:22 Automatic GTB (not on the reading list - the phd-topic of Berent)
1:49 Full lecture recap