The Mathematics of Machine Learning 2025

Lecture 7

Tim van Erven

University of Amsterdam

Outline

Refresher and Motivation

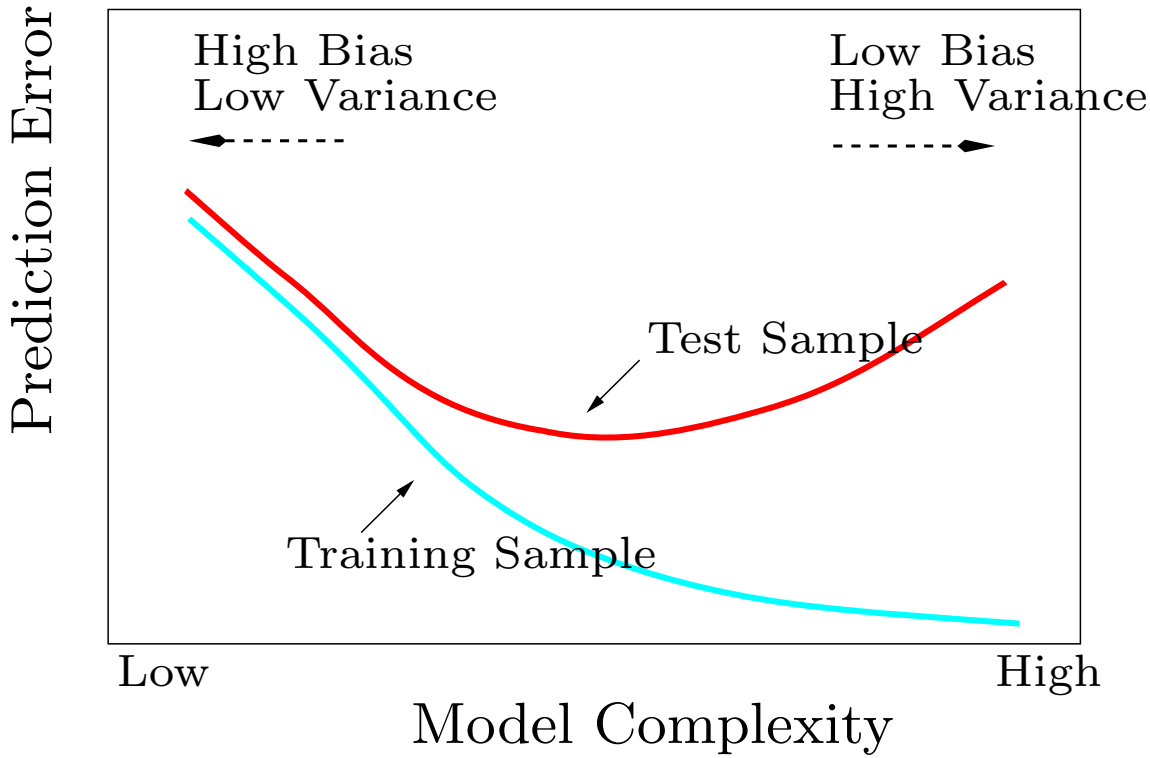
1. Bagging vs Boosting

Boosting

2. Weak Learners
3. AdaBoost
4. Forward Stagewise Additive Modeling
5. AdaBoost as FSAM
6. Theoretical Guarantee
7. Discussion: Exponential Loss

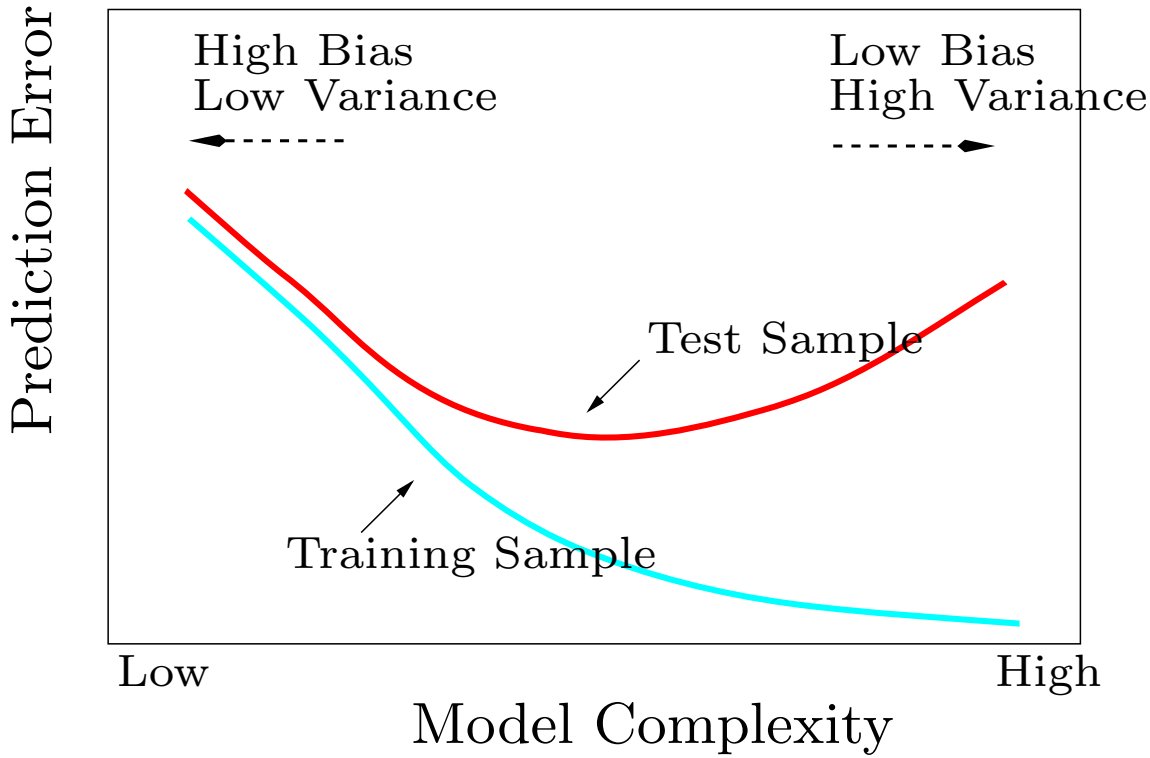
Q&A Session

Bagging vs Boosting



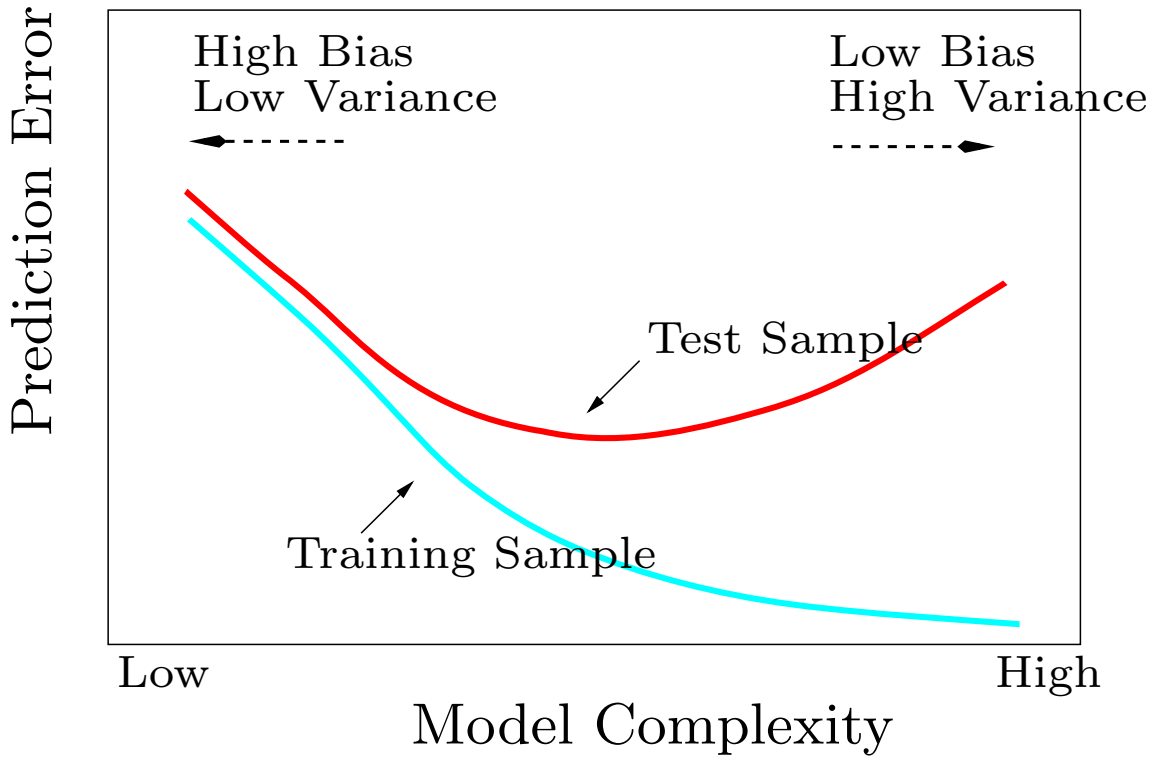
- **Larger model $\mathcal{F}$:** best $f \in \mathcal{F}$ better approximation f_B (small bias), but more difficult to estimate (high variance)
- **Smaller model $\mathcal{F}$:** best $f \in \mathcal{F}$ worse approximation of f_B (large bias), but easier to estimate (small variance)

Bagging vs Boosting



- **Larger model $\mathcal{F}$:** best $f \in \mathcal{F}$ better approximation f_B (small bias), but more difficult to estimate (high variance)
- **Smaller model $\mathcal{F}$:** best $f \in \mathcal{F}$ worse approximation of f_B (large bias), but easier to estimate (small variance)

Bagging vs Boosting



Two ways to make an estimator better:

- **Bagging:** Improve a low bias, high-variance estimator (e.g. unpruned decision tree)
- **Boosting:** Improve a high bias, low-variance estimator

Outline

Refresher and Motivation

1. Bagging vs Boosting

Boosting

2. Weak Learners
3. AdaBoost
4. Forward Stagewise Additive Modeling
5. AdaBoost as FSAM
6. Theoretical Guarantee
7. Discussion: Exponential Loss

Q&A Session

Intro

Start with a **"weak learner"**:
algorithm that learns a **binary classifier**
based on a **small model** $\mathcal{F}_{\mathbf{small}}$ (high bias),
for which we can estimate accurately (low variance)

Intro

Start with a **"weak learner"**:
algorithm that learns a **binary classifier**
based on a **small model** $\mathcal{F}_{\text{small}}$ (high bias),
for which we can estimate accurately (low variance)

Assumptions:

- **No label noise:** $Y = f^*(X)$ for some (unknown) function f^*
- Weak learner does not have to be great: only assume it can fit any (weighted) data set **slightly better than random guessing** (accuracy $\geq 0.5 + \epsilon$ for some $\epsilon > 0$).

Intro

Start with a **"weak learner"**:
algorithm that learns a **binary classifier**
based on a **small model** $\mathcal{F}_{\text{small}}$ (high bias),
for which we can estimate accurately (low variance)

Assumptions:

- **No label noise:** $Y = f^*(X)$ for some (unknown) function f^*
- Weak learner does not have to be great: only assume it can fit any (weighted) data set **slightly better than random guessing** (accuracy $\geq 0.5 + \epsilon$ for some $\epsilon > 0$).

Examples:

- E.g. if weak learner uses ERM over $\mathcal{F}_{\text{small}}$:

$$\min_{G \in \mathcal{F}_{\text{small}}} \sum_{i=1}^{N} w_i L_{0/1}(f^*(X_i), G(X_i)) \leq \frac{1}{2} - \epsilon$$

  for any weights $w_i \geq 0$ such that $\sum_{i=1}^{N} w_i = 1$.
- E.g. $\mathcal{F}_{\text{small}}$ = decision stumps = decision trees with only one binary decision

How can we improve weak learners?

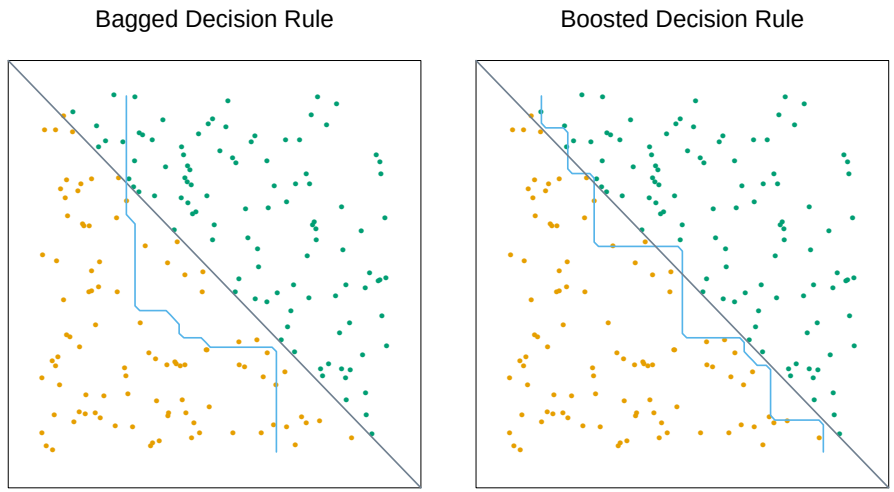


FIGURE 8.12. *Data with two features and two classes, separated by a linear boundary. (Left panel:) Decision boundary estimated from bagging the decision rule from a single split, axis-oriented classifier. (Right panel:) Decision boundary from boosting the decision rule of the same classifier. The test error rates are* 0.166, *and* 0.065, *respectively. Boosting is described in Chapter 10.*

Boosting High-level Approach

$$\begin{pmatrix} Y_1 \\ X_1 \end{pmatrix}, \quad \begin{pmatrix} Y_2 \\ X_2 \end{pmatrix}, \quad \ldots, \quad \begin{pmatrix} Y_N \\ X_N \end{pmatrix} \qquad (Y \in \{-1, +1\})$$

Weights: w_1 w_2 w_N

Boosting High-level Approach

$$\begin{pmatrix} Y_1 \\ X_1 \end{pmatrix}, \quad \begin{pmatrix} Y_2 \\ X_2 \end{pmatrix}, \quad \ldots, \quad \begin{pmatrix} Y_N \\ X_N \end{pmatrix} \qquad (Y \in \{-1, +1\})$$

Weights: w_1 w_2 w_N

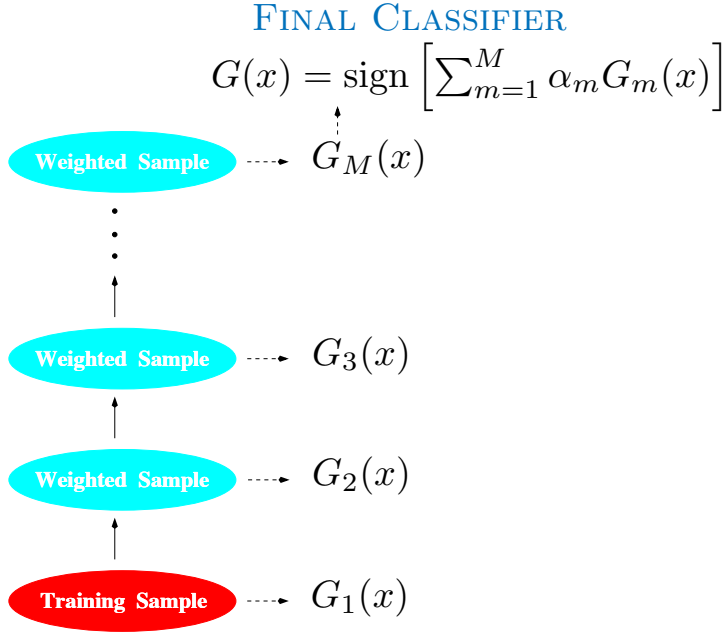


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

Boosting High-level Approach

$$\begin{pmatrix} Y_1 \\ X_1 \end{pmatrix}, \quad \begin{pmatrix} Y_2 \\ X_2 \end{pmatrix}, \quad \ldots, \quad \begin{pmatrix} Y_N \\ X_N \end{pmatrix} \qquad (Y \in \{-1, +1\})$$

Weights: w_1 w_2 w_N

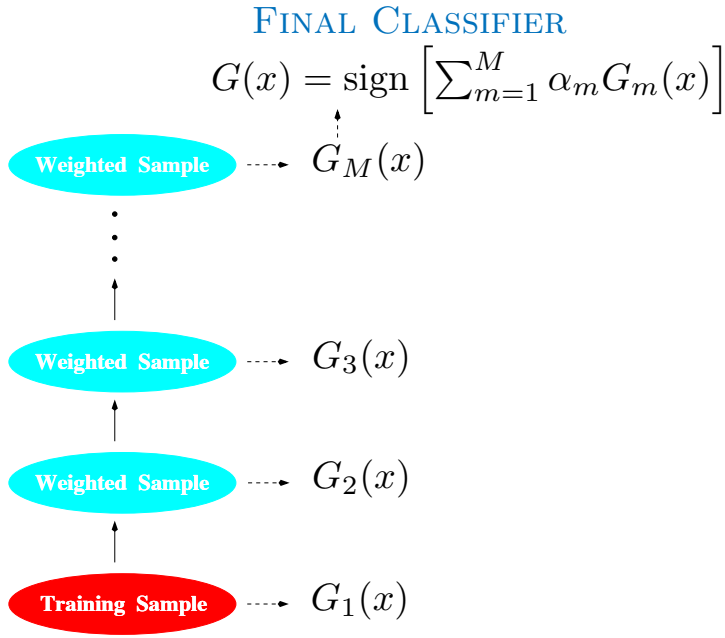


FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

To train G_m, increase weights for examples misclassified by G_{m-1}

- Force learning to concentrate on difficult examples
- But **how** should we update the weights?

Boosting Decision Stumps

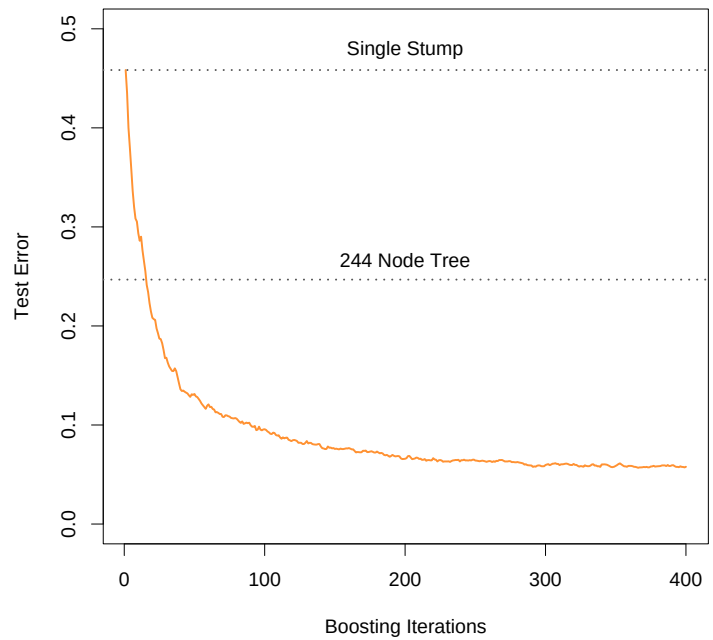


FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a* 244*-node classification tree.*

AdaBoost

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:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (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 $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

AdaBoost

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:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (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 $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

But why update the weights like this?

Forward Stagewise Additive Modeling

Additive Models:

$$f(x) = \sum_{m=1}^{M} \beta_m \, b_{\gamma_m}(x)$$

weights

basis functions with parameters γ_m

Forward Stagewise Additive Modeling

Additive Models:

$$f(x) = \sum_{m=1}^{M} \beta_m \, b_{\gamma_m}(x)$$

weights

basis functions with parameters γ_m

Fitting by ERM often computationally difficult:

$$\min_{(\beta_m, \gamma_m)_{m=1,\dots,M}} \quad \sum_{i=1}^{N} L\Big(Y_i, \sum_{m=1}^{M} \beta_m b_{\gamma_m}(X_i)\Big)$$

Forward Stagewise Additive Modeling

Additive Models:

$$f(x) = \sum_{m=1}^{M} \beta_m \, b_{\gamma_m}(x)$$

weights

basis functions with parameters γ_m

Fitting by ERM often computationally difficult:

$$\min_{(\beta_m, \gamma_m)_{m=1,\dots,M}} \quad \sum_{i=1}^{N} L\Big(Y_i, \sum_{m=1}^{M} \beta_m b_{\gamma_m}(X_i)\Big)$$

Forward Stagewise Additive Modeling (FSAM):

For $m = 1, \dots, M$:

- $f_m(x) = f_{m-1}(x) + \beta_m b_{\gamma_m}(x)$
- where we find parameters β_m, γ_m by ERM:

$$\min_{\beta_m, \gamma_m} \quad \sum_{i=1}^{N} L\Big(Y_i, f_{m-1}(X_i) + \beta_m b_{\gamma_m}(X_i)\Big)$$

AdaBoost as FSAM I

Basis functions: G_m
Exponential loss: $L\big(Y, f(X)\big) = e^{-Yf(X)}$

$$\begin{aligned}\beta_m, G_m &= \arg\min_{\beta, G} \sum_{i=1}^{N} e^{-Y_i\left(f_{m-1}(X_i)+\beta G(X_i)\right)} \\ &= \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} e^{-\beta Y_i G(X_i)},\end{aligned}$$

where $w_i^{(m)} = \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}}$ for normalization $Z^{(m)}$ s.t. $\sum_{i=1}^{N} w_i^{(m)} = 1$.

AdaBoost as FSAM I

Basis functions: G_m
Exponential loss: $L\big(Y, f(X)\big) = e^{-Yf(X)}$

$$\beta_m, G_m = \underset{\beta, G}{\arg\min} \sum_{i=1}^{N} w_i^{(m)} e^{-\beta Y_i G(X_i)}$$

$$\boxed{w_i^{(m)} = \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}}}$$

AdaBoost as FSAM I

Basis functions: G_m
Exponential loss: $L(Y, f(X)) = e^{-Yf(X)}$

$$\beta_m, G_m = \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} e^{-\beta Y_i G(X_i)}$$

$$\boxed{w_i^{(m)} = \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}}}$$

Solve first for G given any fixed β:

$$\min_G \sum_{i:G(X_i)=Y_i} w_i^{(m)} e^{-\beta} + \sum_{i:G(X_i)\neq Y_i} w_i^{(m)} e^{\beta}$$

$$= \min_G \; (e^{\beta} - e^{-\beta}) \sum_{i=1}^{N} w_i^{(m)} L_{0/1}(Y_i, G(X_i)) + e^{-\beta} \underbrace{\sum_{i=1}^{N} w_i^{(m)}}_{1}$$

AdaBoost as FSAM I

Basis functions: G_m
Exponential loss: $L(Y, f(X)) = e^{-Yf(X)}$

$$\beta_m, G_m = \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} e^{-\beta Y_i G(X_i)}$$

$$\boxed{w_i^{(m)} = \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}}}$$

Solve first for G given any fixed β:

$$\min_G \sum_{i:G(X_i)=Y_i} w_i^{(m)} e^{-\beta} + \sum_{i:G(X_i)\neq Y_i} w_i^{(m)} e^{\beta}$$

$$= \min_G \; (e^{\beta} - e^{-\beta}) \sum_{i=1}^{N} w_i^{(m)} L_{0/1}(Y_i, G(X_i)) + e^{-\beta} \underbrace{\sum_{i=1}^{N} w_i^{(m)}}_{1}$$

Weighted ERM (view 2d in AdaBoost as approximately solving this):

$$\min_G \sum_{i=1}^{N} w_i^{(m)} L_{0/1}(Y_i, G(X_i)) \qquad \leftarrow \text{err}_m$$

AdaBoost as FSAM I

Basis functions: G_m
Exponential loss: $L(Y, f(X)) = e^{-Yf(X)}$

$$\beta_m, G_m = \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} e^{-\beta Y_i G(X_i)}$$

$$\boxed{w_i^{(m)} = \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}}}$$

Solve first for G given any fixed β:

$$\min_G \sum_{i:G(X_i)=Y_i} w_i^{(m)} e^{-\beta} + \sum_{i:G(X_i)\neq Y_i} w_i^{(m)} e^{\beta}$$

$$= \min_G \; (e^{\beta} - e^{-\beta}) \sum_{i=1}^{N} w_i^{(m)} L_{0/1}(Y_i, G(X_i)) + e^{-\beta} \underbrace{\sum_{i=1}^{N} w_i^{(m)}}_{1}$$

Weighted ERM (view 2d in AdaBoost as approximately solving this):

$$\min_G \sum_{i=1}^{N} w_i^{(m)} L_{0/1}(Y_i, G(X_i)) \qquad \leftarrow \text{err}_m \leq \frac{1}{2} - \epsilon$$

AdaBoost as FSAM I

Basis functions: G_m
Exponential loss: $L(Y, f(X)) = e^{-Yf(X)}$

$$\beta_m, G_m = \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} e^{-\beta Y_i G(X_i)}$$

$$\boxed{w_i^{(m)} = \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}}}$$

Solve first for G given any fixed β:

$$\begin{aligned} &\min_G \sum_{i:G(X_i)=Y_i} w_i^{(m)} e^{-\beta} + \sum_{i:G(X_i)\neq Y_i} w_i^{(m)} e^{\beta} \\ = \; &(e^{\beta} - e^{-\beta}) \, \text{err}_m + e^{-\beta} \end{aligned}$$

Weighted ERM (view 2d in AdaBoost as approximately solving this):

$$\min_G \sum_{i=1}^{N} w_i^{(m)} L_{0/1}(Y_i, G(X_i)) \qquad \leftarrow \text{err}_m \leq \frac{1}{2} - \epsilon$$

AdaBoost as FSAM I

Basis functions: G_m
Exponential loss: $L(Y, f(X)) = e^{-Yf(X)}$

$$\beta_m, G_m = \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} e^{-\beta Y_i G(X_i)}$$

$$\boxed{w_i^{(m)} = \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}}}$$

Solve first for G given any fixed β:

$$\begin{aligned} &\min_G \sum_{i:G(X_i)=Y_i} w_i^{(m)} e^{-\beta} + \sum_{i:G(X_i)\neq Y_i} w_i^{(m)} e^{\beta} \\ = \;& (e^{\beta} - e^{-\beta}) \, \mathrm{err}_m + e^{-\beta} \end{aligned}$$

Solve for β:

$$0 = \frac{\mathrm{d}}{\mathrm{d}\beta}(e^{\beta} - e^{-\beta}) \, \mathrm{err}_m + e^{-\beta} = (e^{\beta} + e^{-\beta}) \, \mathrm{err}_m - e^{-\beta}$$

$$(e^{2\beta} + 1) \, \mathrm{err}_m = 1 \quad \rightarrow \quad \beta = \tfrac{1}{2} \log \frac{1 - \mathrm{err}_m}{\mathrm{err}_m} = \tfrac{1}{2} \alpha_m$$

AdaBoost as FSAM II

Updating the weights:

$$w_i^{(m)} = \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}}$$

$$\propto \frac{e^{-Y_i f_{m-2}(X_i)}}{Z^{(m-1)}} e^{-Y_i \beta_{m-1} G_{m-1}(X_i)}$$

$$\boxed{Z^{(m)} = \sum_{i=1}^{N} e^{-Y_i f_{m-1}(X_i)}}$$

AdaBoost as FSAM II

Updating the weights:

$$
\begin{aligned}
w_i^{(m)} &= \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}} \\
&\propto \frac{e^{-Y_i f_{m-2}(X_i)}}{Z^{(m-1)}} e^{-Y_i \beta_{m-1} G_{m-1}(X_i)} \\
&= w_i^{(m-1)} e^{-Y_i \beta_{m-1} G_{m-1}(X_i)} \\
&= w_i^{(m-1)} e^{\beta_{m-1}\left(2\,\mathbb{1}[G_{m-1}(X_i) \neq Y_i] - 1\right)}
\end{aligned}
$$

$$
\boxed{Z^{(m)} = \sum_{i=1}^{N} e^{-Y_i f_{m-1}(X_i)}}
$$

AdaBoost as FSAM II

Updating the weights:

$$
\begin{aligned}
w_i^{(m)} &= \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}} \\
&\propto \frac{e^{-Y_i f_{m-2}(X_i)}}{Z^{(m-1)}} e^{-Y_i \beta_{m-1} G_{m-1}(X_i)} \\
&= w_i^{(m-1)} e^{-Y_i \beta_{m-1} G_{m-1}(X_i)} \\
&= w_i^{(m-1)} e^{\beta_{m-1}\left(2\,\mathbb{1}[G_{m-1}(X_i) \neq Y_i] - 1\right)} \\
&\propto w_i^{(m-1)} e^{\alpha_{m-1}\mathbb{1}[G_{m-1}(X_i) \neq Y_i]} \qquad \leftarrow \text{Weight update 2d in AdaBoost!}
\end{aligned}
$$

$$
\boxed{Z^{(m)} = \sum_{i=1}^{N} e^{-Y_i f_{m-1}(X_i)}}
$$

AdaBoost as FSAM II

Updating the weights:

$$w_i^{(m)} = \frac{e^{-Y_i f_{m-1}(X_i)}}{Z^{(m)}}$$

$$\boxed{Z^{(m)} = \sum_{i=1}^{N} e^{-Y_i f_{m-1}(X_i)}}$$

$$\propto \frac{e^{-Y_i f_{m-2}(X_i)}}{Z^{(m-1)}} e^{-Y_i \beta_{m-1} G_{m-1}(X_i)}$$

$$= w_i^{(m-1)} e^{-Y_i \beta_{m-1} G_{m-1}(X_i)}$$

$$= w_i^{(m-1)} e^{\beta_{m-1}\left(2\mathbb{1}[G_{m-1}(X_i) \neq Y_i] - 1\right)}$$

$$\propto w_i^{(m-1)} e^{\alpha_{m-1}\mathbb{1}[G_{m-1}(X_i) \neq Y_i]} \quad \leftarrow \text{Weight update 2d in AdaBoost!}$$

Classification rule for the learned function:

$$\operatorname{sign}\left(f_M(X)\right) = \operatorname{sign}\left(\sum_{m=1}^{M} \beta_m G_m(X)\right)$$

$$= \operatorname{sign}\left(\sum_{m=1}^{M} \alpha_m G_m(X)\right) = G(X) \quad \leftarrow \text{Line 3 in AdaBoost}$$

Theoretical Guarantee

Assumptions:

- **No label noise:** $Y = f^*(X)$ for some (unknown) function f^*
- Weak learner does not have to be great: only assume it can fit any (weighted) data set **slightly better than random guessing** (accuracy $\geq 0.5 + \epsilon$ for some $\epsilon > 0$).

Theorem (Freund and Schapire [1997])

Under these assumptions, the classifier $\hat{f}(X) = \text{sign}\left(f_M(X)\right)$ *produced by AdaBoost after* M *iterations can fit the training data as well as we like by taking* M *large enough:*

$$\frac{1}{N}\sum_{i=1}^{N} L_{0/1}(Y_i, \hat{f}(X_i)) \leq (1 - 4\epsilon^2)^{M/2}$$

Proved in Machine Learning Theory master course

Theoretical Guarantee

Assumptions:

- **No label noise:** $Y = f^*(X)$ for some (unknown) function f^*
- Weak learner does not have to be great: only assume it can fit any (weighted) data set **slightly better than random guessing** (accuracy $\geq 0.5 + \epsilon$ for some $\epsilon > 0$).

Theorem (Freund and Schapire [1997])

Under these assumptions, the classifier $\hat{f}(X) = \text{sign}\left(f_M(X)\right)$ *produced by AdaBoost after* M *iterations can fit the training data as well as we like by taking* M *large enough:*

$$\frac{1}{N}\sum_{i=1}^{N} L_{0/1}(Y_i, \hat{f}(X_i)) \leq (1 - 4\epsilon^2)^{M/2}$$

- Theory: If $\mathcal{F}_{\text{small}}$ not too complex, then majority vote among M elements from $\mathcal{F}_{\text{small}}$ does not overfit as long as M is not too large
- Practice: error on test set often does not get worse even if we take M to be very large

Illustration

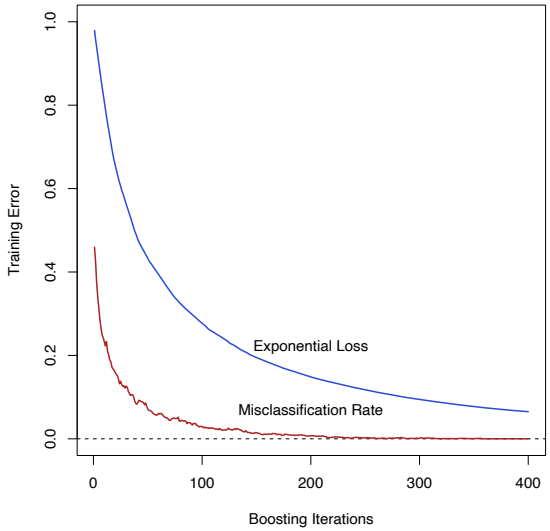


FIGURE 10.3. *Simulated data, boosting with stumps: misclassification error rate on the training set, and average exponential loss:* $(1/N)\sum_{i=1}^{N}\exp(-y_i f(x_i))$. *After about 250 iterations, the misclassification error is zero, while the exponential loss continues to decrease.*

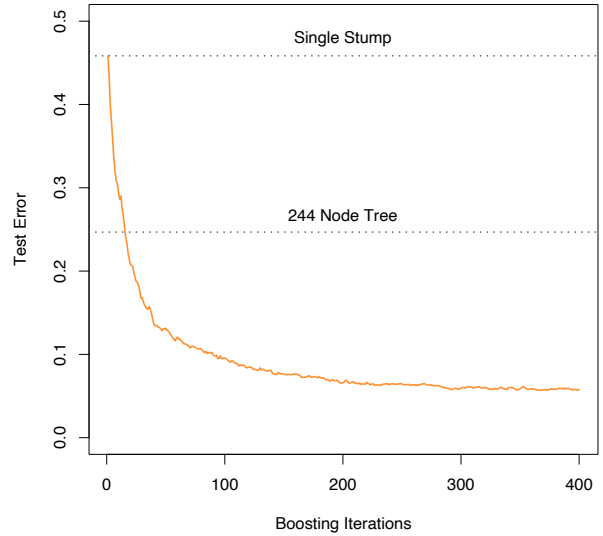
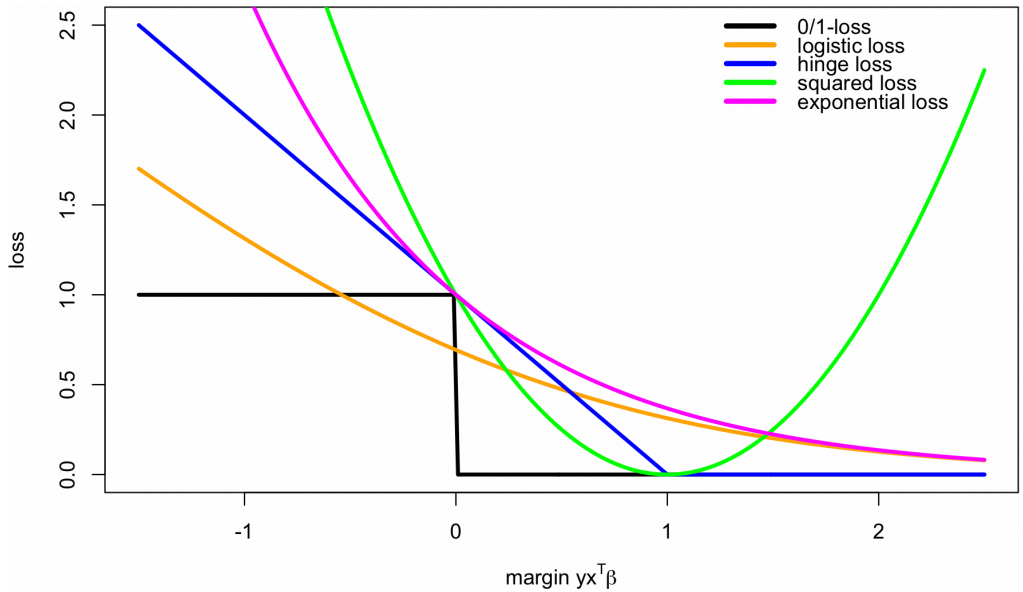


FIGURE 10.2. *Simulated data (10.2): test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree.*

As number of boosting iterations M grows:

- Training error/misclassification rate goes to 0 (left)
- Test set error keeps decreasing and then stabilizes (right), but does not go up, so does not overfit even for large M
- Exponential loss also keeps decreasing (left)

Discussion: Exponential Loss

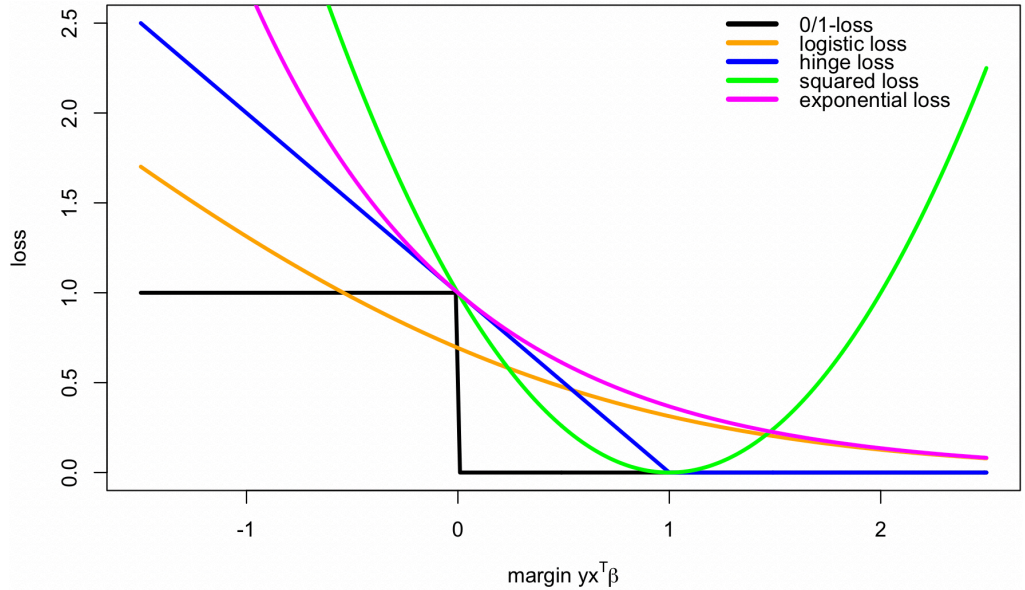


Estimate in population:

$$f_{\mathsf{B}} = \arg\min_f \mathbb{E}\left[e^{-Yf(X)} \mid X\right] = \tfrac{1}{2}\log\frac{P^*(Y=+1 \mid X)}{P^*(Y=-1 \mid X)}$$

So exponential loss tries to estimate log odds of classes.

Discussion: Exponential Loss



Behavior under label noise:

- Suppose assumption of no label noise is violated
- Exponential loss: strongly punishes large negative margin
- $\rightarrow$ AdaBoost gives large weights to data points that are misclassified with large margin
- $\rightarrow$ Not robust in settings where this is unavoidable: large noise or mislabeled data
- Then might be better to use alternative loss like binomial deviance $\approx$ logistic loss, although might be computationally more challenging

Outline

Refresher and Motivation

1. Bagging vs Boosting

Boosting

2. Weak Learners
3. AdaBoost
4. Forward Stagewise Additive Modeling
5. AdaBoost as FSAM
6. Theoretical Guarantee
7. Discussion: Exponential Loss

Q&A Session

References

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1997.