CSC 4020 Fundamentals of Machine Learning: Boosting

Baoyuan Wu

April 7

[Slide credit: Verma & Thrun]

Boosting

Recall that an *ensemble* is a set of predictors whose individual decisions are combined in some way to classify new examples.

Bagging: Train classifiers independently on random subsets of the training data.

Boosting: Train classifiers sequentially, each time focusing on training data points that were previously misclassified.

Let us start with the concept of **weak learner/classifier** (or base classifiers).

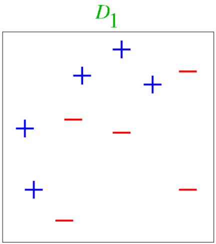
Weak Learner/Classifier

(Informal) Weak learner is a learning algorithm that outputs a hypothesis (e.g., a classifier) that performs slightly better than chance, e.g., it predicts the correct label with probability 0.6.

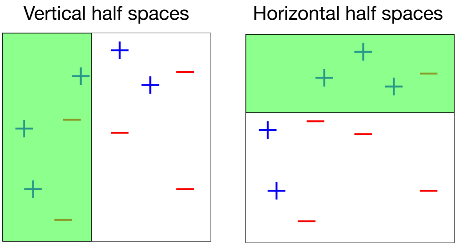
We are interested in weak learners that are *computationally* efficient.

- Decision trees
- Even simpler: **Decision Stump**: A decision tree with only a single split

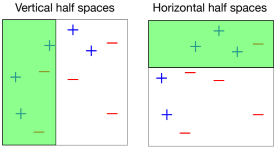
Weak Classifiers



These weak classifiers, which are decision stumps, consist of the set of horizontal and vertical half spaces.



Weak Classifiers



A *single* weak classifier is not capable of making the training error very small. It only perform slightly better than chance, i.e., the error of classifier h according to the given weights $\mathbf{w} = (w_1, \dots, w_N)$ (with $\sum_{i=1}^N w_i = 1$ and $w_i \geq 0$)

$$\text{err} = \sum_{i=1}^{N} w_i \mathbb{I}\{h(\mathbf{x}_i) \neq y_i\}$$

is at most $\frac{1}{2} - \gamma$ for some $\gamma > 0$.

Can we combine a set of weak classifiers in order to make a better ensemble of classifiers?

Boosting: Train classifiers sequentially, each time focusing on training data points that were previously misclassified.

AdaBoost (Adaptive Boosting)

Key steps of AdaBoost:

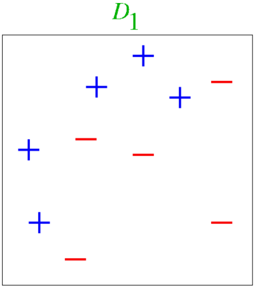
1. At each iteration we re-weight the training samples by **assigning larger weights** to samples (i.e., data points) that were **classified incorrectly**.
2. We train a new weak classifier based on the **re-weighted samples**.
3. We **add this weak classifier to the ensemble** of classifiers. This is our new classifier.
4. Weight each weak classifier in the ensemble with some weights.
5. We repeat the process many times.

The weak learner needs to minimize weighted error.

AdaBoost reduces **bias** by making each classifier focus on previous mistakes.

AdaBoost Example

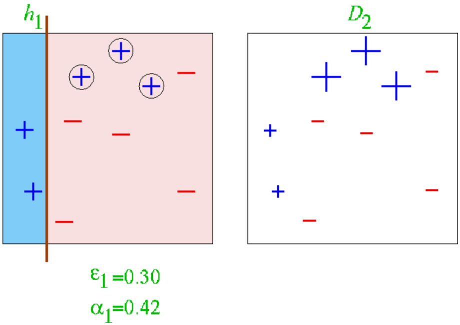
Training data



AdaBoost Example

Round 1

ϵ : *Training error*, α: Weighting of the current tree.

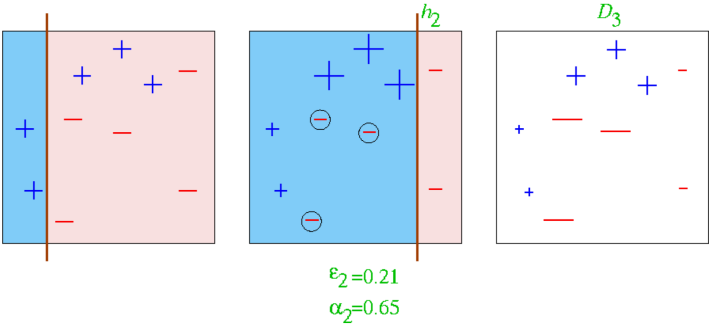


$$\mathbf{w} = \left(\frac{1}{10}, \dots, \frac{1}{10}\right) \Rightarrow \text{Train a classifier (using } \mathbf{w}) \Rightarrow \text{err}_1 = \frac{\sum_{i=1}^{10} w_i \mathbb{I}\{h_1(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} = \frac{3}{10}$$

$$\Rightarrow \alpha_1 = \frac{1}{2} \log \frac{1 - \text{err}_1}{\text{err}_1} = \frac{1}{2} \log\left(\frac{1}{0.3} - 1\right) \approx 0.42 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x})\right)$$

AdaBoost Example

Round 2

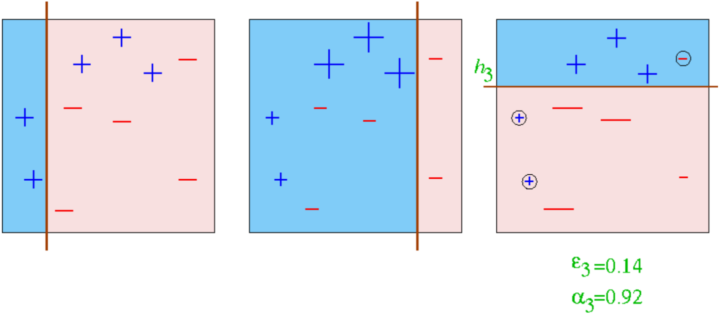


$\mathbf{w}$ = updated weights $\Rightarrow$ Train a classifier (using $\mathbf{w}$) $\Rightarrow \text{err}_2 = \frac{\sum_{i=1}^{10} w_i \mathbb{I}\{h_2(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} = 0.21$

$\Rightarrow \alpha_2 = \frac{1}{2} \log \frac{1 - \text{err}_3}{\text{err}_3} = \frac{1}{2} \log(\frac{1}{0.21} - 1) \approx 0.66 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x}) + \alpha_2 h_2(\mathbf{x})\right)$

AdaBoost Example

Round 3

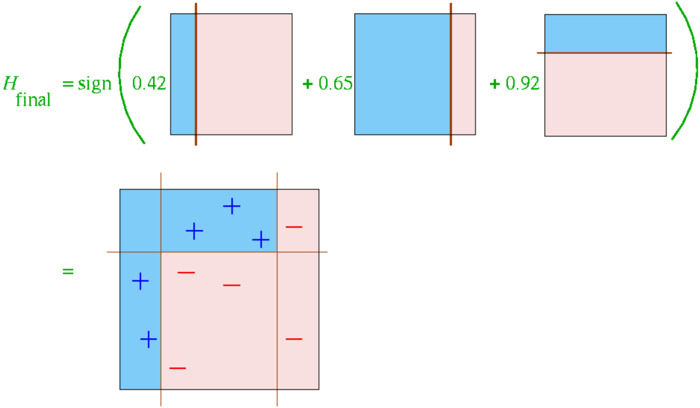


$\mathbf{w}$ = updated weights $\Rightarrow$ Train a classifier (using $\mathbf{w}$) $\Rightarrow$ $\text{err}_3 = \frac{\sum_{i=1}^{10} w_i \mathbb{I}\{h_3(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i} = 0.14$

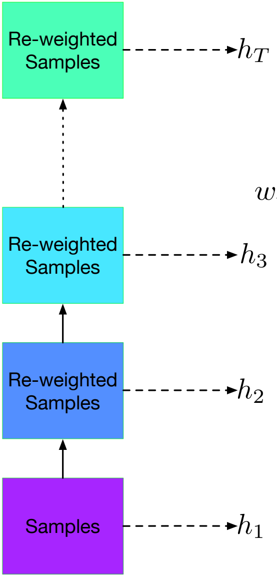
$\Rightarrow \alpha_3 = \frac{1}{2} \log \frac{1-\text{err}_3}{\text{err}_3} = \frac{1}{2} \log(\frac{1}{0.14} - 1) \approx 0.91 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x}) + \alpha_2 h_2(\mathbf{x}) + \alpha_3 h_3(\mathbf{x})\right)$

AdaBoost Example

Final classifier



AdaBoost Algorithm



$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

$$w_i \leftarrow w_i \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x^{(i)}}) \neq \mathbf{t^{(i)}}\}\right)$$

$$\alpha_t = \frac{1}{2}\log\left(\frac{1 - \text{err}_t}{\text{err}_t}\right)$$

$$\text{err}_t = \frac{\sum_{i=1}^{N} w_i \mathbb{I}\{h_t(\mathbf{x^{(i)}} \neq \mathbf{t^{(i)}}\}}{\sum_{i=1}^{N} w_i}$$

AdaBoost Algorithm

Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(i)}, t^{(i)}\}_{i=1}^N$, weak classifier WeakLearn (a classification procedure that return a classifier from base hypothesis space $\mathcal{H}$ with $h : \mathbf{x} \to \{-1, +1\}$ for $h \in \mathcal{H}$), number of iterations T

Output: Classifier $H(x)$

Initialize sample weights: $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$

For $t = 1, \ldots, T$

- Fit a classifier to data using weighted samples ($h_t \leftarrow WeakLearn(\mathcal{D}_N, \mathbf{w})$), e.g.,

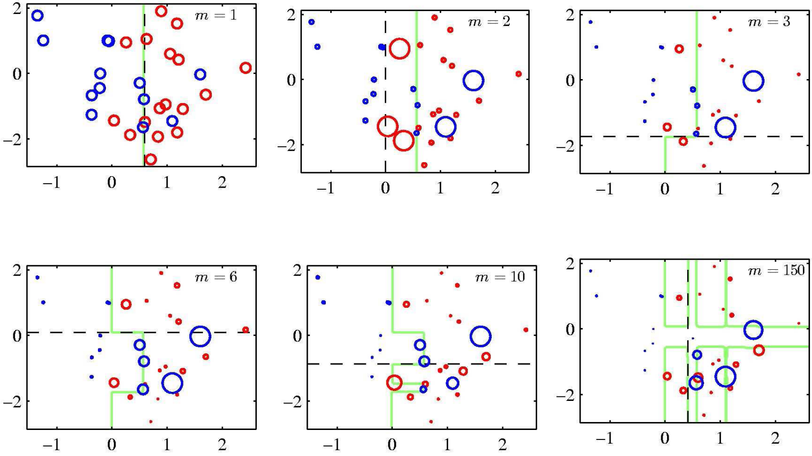
$$h_t \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{N} w_i \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t^{(i)}\}$$

- Compute weighted error $\text{err}_t = \frac{\sum_{i=1}^N w_i \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^N w_i}$
- Compute classifier coefficient $\alpha_t = \frac{1}{2} \log \frac{1 - \text{err}_t}{\text{err}_t}$
- Update data weights

$$w_i \leftarrow w_i \exp\left(-\alpha_t t^{(i)} h_t(\mathbf{x}^{(i)})\right) \left[\equiv w_i \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(i)}) \neq t^{(i)}\}\right)\right]$$

Return $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$

AdaBoost Example



Each figure shows the number m of base learners trained so far, the decision of the most recent learner (dashed black), and the boundary of the ensemble (green)

AdaBoost Minimizes the Training Error

Theorem

Assume that at each iteration of AdaBoost the WeakLearn returns a hypothesis with error $\mathrm{err}_t \leq \frac{1}{2} - \gamma$ for all $t = 1, \ldots, T$ with $\gamma > 0$. The training error of the output hypothesis $H(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$ is at most

$$L_N(H) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\{H(\mathbf{x}^{(i)}) \neq t^{(i)})\} \leq \exp\left(-2\gamma^2 T\right).$$

This is under the simplifying assumption that each weak learner is γ-better than a random predictor.

Analyzing the convergence of AdaBoost is generally difficult.

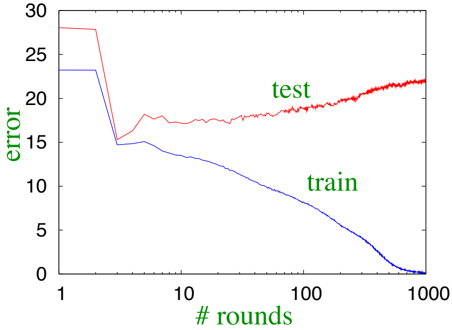
Generalization Error of AdaBoost

AdaBoost's training error (loss) converges to zero. What about the test error of H?

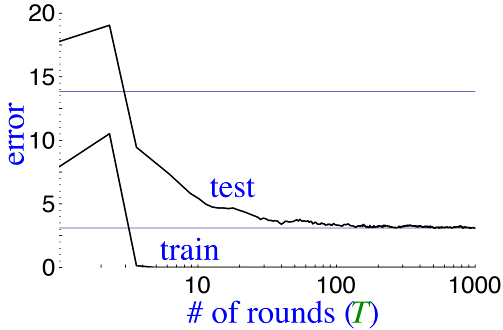
As we add more weak classifiers, the overall classifier H becomes more "complex".

We expect more complex classifiers overfit.

If one runs AdaBoost long enough, it can in fact overfit.



Generalization Error of AdaBoost



How does that happen?

[Slide credit: Robert Shapire's Slides, `http://www.cs.princeton.edu/courses/archive/spring12/cos598A/schedule.html`]

Alternative derivation of AdaBoost: Additive Models

Consider a hypothesis class $\mathcal{H}$ with each $h_i : \mathbf{x} \mapsto \{-1, +1\}$ within $\mathcal{H}$, i.e., $h_i \in \mathcal{H}$. These are the "weak learners", and in this context they're also called **bases**.

An **additive model** with m terms is given by

$$H_m(x) = \sum_{i=1}^{m} \alpha_i h_i(\mathbf{x}),$$

where $(\alpha_1, \cdots, \alpha_m) \in \mathbb{R}^m$.

Observe that we're taking a linear combination of base classifiers, just like in boosting.

We'll now interpret AdaBoost as a way of fitting an additive model.

Stagewise Training of Additive Models

A greedy approach to fitting additive models, known as **stagewise training**:

1. Initialize $H_0(x) = 0$
2. For $m = 1$ to T:
   - Compute the m-th hypothesis and its coefficient

$$(h_m, \alpha_m) \leftarrow \underset{h \in \mathcal{H}, \alpha}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\left(H_{m-1}(\mathbf{x}^{(i)}) + \alpha h(\mathbf{x}^{(i)}), t^{(i)}\right)$$

   - Add it to the additive model

$$H_m = H_{m-1} + \alpha_m h_m$$

Additive Models with Exponential Loss

Consider the exponential loss

$$\mathcal{L}_{\mathrm{E}}(y, t) = \exp(-ty).$$

We want to see how the stagewise training of additive models can be done.

$$\begin{aligned}
(h_m, \alpha_m) \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}, \alpha} \sum_{i=1}^{N} \exp\left(-\left[H_{m-1}(\mathbf{x}^{(i)}) + \alpha h(\mathbf{x}^{(i)})\right] t^{(i)}\right) \\
= \sum_{i=1}^{N} \exp\left(-H_{m-1}(\mathbf{x}^{(i)}) t^{(i)} - \alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) \\
= \sum_{i=1}^{N} \exp\left(-H_{m-1}(\mathbf{x}^{(i)}) t^{(i)}\right) \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) \\
= \sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right).
\end{aligned}$$

Here we defined $w_i^{(m)} \triangleq \exp\left(-H_{m-1}(\mathbf{x}^{(i)}) t^{(i)}\right)$.

Additive Models with Exponential Loss

We want to solve the following minimization problem:

$$(h_m, \alpha_m) \leftarrow \underset{h \in \mathcal{H}, \alpha}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right).$$

If $h(\mathbf{x}^{(i)}) = t^{(i)}$, we have $\exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = \exp(-\alpha)$.

If $h(\mathbf{x}^{(i)}) \neq t^{(i)}$, we have $\exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = \exp(+\alpha)$.

(recall that we are in the binary classification case with $\{-1, +1\}$ output values).
We can divide the summation to two parts:

$$\begin{aligned}
\sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) =& e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) = t_i\} + e^{\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} \\
=& (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} + \\
& e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \left[\mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} + \mathbb{I}\{h(\mathbf{x}^{(i)}) = t_i\}\right]
\end{aligned}$$

Additive Models with Exponential Loss

$$\sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)} \neq t_i\} +$$

$$e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \left[\mathbb{I}\{h(\mathbf{x}^{(i)} \neq t_i\} + \mathbb{I}\{h(\mathbf{x}^{(i)}) = t_i\}\right]$$

$$= (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\} + e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)}.$$

Let us first optimize h:
The second term on the RHS does not depend on h. So we get

$$h_m \leftarrow \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{N} w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) \equiv \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\}.$$

This means that h_m is the minimizer of the weighted 0/1-loss.

Additive Models with Exponential Loss

Now that we obtained h_m, we want to find α: Define the weighted classification error:

$$\text{err}_m = \frac{\sum_{i=1}^N w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^N w_i^{(m)}}$$

With this definition and $\min_{h \in \mathcal{H}} \sum_{i=1}^N w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) = \sum_{i=1}^N w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t_i\}$, we have

$$\min_{\alpha} \min_{h \in \mathcal{H}} \sum_{i=1}^N w_i^{(m)} \exp\left(-\alpha h(\mathbf{x}^{(i)}) t^{(i)}\right) =$$

$$\min_{\alpha} \left\{ (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^N w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t_i\} + e^{-\alpha} \sum_{i=1}^N w_i^{(m)} \right\}$$

$$= \min_{\alpha} \left\{ (e^{\alpha} - e^{-\alpha}) \text{err}_m \left(\sum_{i=1}^N w_i^{(m)} \right) + e^{-\alpha} \left(\sum_{i=1}^N w_i^{(m)} \right) \right\}$$

Take derivative w.r.t. α and set it to zero. We get that

$$e^{2\alpha} = \frac{1 - \text{err}_m}{\text{err}_m} \Rightarrow \alpha = \frac{1}{2} \log \left(\frac{1 - \text{err}_m}{\text{err}_m} \right).$$

Additive Models with Exponential Loss

The updated weights for the next iteration is

$$
\begin{aligned}
w_i^{(m+1)} &= \exp\left(-H_m(\mathbf{x}^{(i)})t^{(i)}\right) \\
&= \exp\left(-\left[H_{m-1}(\mathbf{x}^{(i)}) + \alpha_m h_m(\mathbf{x}^{(i)})\right] t^{(i)}\right) \\
&= \exp\left(-H_{m-1}(\mathbf{x}^{(i)})t^{(i)}\right) \exp\left(-\alpha_m h_m(\mathbf{x}^{(i)})t^{(i)}\right) \\
&= w_i^{(m)} \exp\left(-\alpha_m h_m(\mathbf{x}^{(i)})t^{(i)}\right) \\
&= w_i^{(m)} \exp\left(-\alpha_m \left(2\mathbb{I}\{h_m(\mathbf{x}^{(i)}) = t^{(i)}\} - 1\right)\right) \\
&= \exp(\alpha_m) w_i^{(m)} \exp\left(-2\alpha_m \mathbb{I}\{h_m(\mathbf{x}^{(i)}) = t^{(i)}\}\right).
\end{aligned}
$$

The term $\exp(\alpha_m)$ multiplies the weight corresponding to all samples, so it does not affect the minimization of h_{m+1} or α_{m+1}.

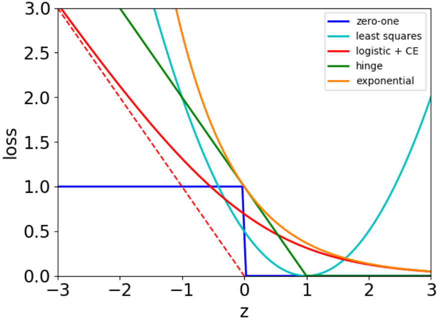
Additive Models with Exponential Loss

To summarize, we obtain the additive model $H_m(x) = \sum_{i=1}^{m} \alpha_i h_i(\mathbf{x})$ with

$$
\begin{aligned}
h_m &\leftarrow \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h(\mathbf{x}^{(i)}) \neq t_i\}, \\
\alpha &= \frac{1}{2} \log \left(\frac{1 - \mathrm{err}_m}{\mathrm{err}_m} \right), \qquad \text{where } \mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} \mathbb{I}\{h_m(\mathbf{x}^{(i)}) \neq t^{(i)}\}}{\sum_{i=1}^{N} w_i^{(m)}}, \\
w_i^{(m+1)} &= w_i^{(m)} \exp\left(-\alpha_m h_m(\mathbf{x}^{(i)}) t^{(i)} \right).
\end{aligned}
$$

We derived the AdaBoost algorithm!

Revisiting Loss Functions for Classification



This interpretation allows boosting to be generalized to lots of other loss functions.

Summary

Boosting reduces bias by generating an ensemble of weak classifiers.

Each classifier is trained to reduce errors of previous ensemble.

It is quite resilient to overfitting, though it can overfit.

We will later provide a loss minimization viewpoint to AdaBoost. It allows us to derive other boosting algorithms for regression, ranking, etc.

Ensembles Recap

Ensembles combine classifiers to improve performance

Boosting

- Reduces bias
- Increases variance (large ensemble can cause overfitting)
- Sequential
- High dependency between ensemble elements

Bagging

- Reduces variance (large ensemble can't cause overfitting)
- Bias is not changed (much)
- Parallel
- Want to minimize correlation between ensemble elements.

Supervised Learning Recap