Machine Learning for Intelligent Systems

Lecture 24: Boosting

Reading: UML 10-10.3

Optional Readings: Schapire's survey and tutorial

Instructors: Nika Haghtalab (this time) and Thorsten Joachims

Fundamental Question

I want a learning algorithm that for any distribution P learns an **excellent** classifier h_{strong} such that $err_P(h_{strong}) \leq 0.01$.

I'm given a learning algorithm A that for any distribution D returns a **not-too-terrible** classifier h_{weak} such that $err_D(h_{weak}) \leq 0.49$.

Can I use this algorithm A to find h_{strong} ,

$err_P(h_{strong}) \leq 0.01$?

Strong versus Weak Learning

Strong Learner

A learning algorithm for PAC learning.

For **every** distribution P and **every** ϵ, a **strong learner** can return a classifier h such that $err_P(h) \leq \epsilon$.

With probability $1 - \delta$

Error of random guessing: For any distribution P, ignore P and

- for each x predict $+1$ or -1 , with probability 50-50.
- What's the error?
- Exactly 0.5

Weak Learner

Better than random guessing.

For **every** distribution P and some $\gamma > 0$, a **weak Learner** returns a classifier h such that $err_P(h) \leq \frac{1}{2} - \gamma$.

With probability $1 - \delta$

Boosting

Is there a **boosting** algorithm that turns a weak learner into a strong learner?

Michael Kearns

Leslie Valiant

Robert Schapire

Yoav Fruend

Yes!

There is boosting algorithm that uses a **weak learner** on an adaptively designed polynomial-size sequence of distributions and **strong learns.**

Weak Learning = Strong Learning

Warmup

Suppose our weak learner knows when it doesn't know!

- $h: x \to \{+1, -1, \text{Not sure}\}$.
- On at most $1 - \epsilon'$ fraction of the data, it can say "Not sure".
- On the fraction of the data that it is sure, it makes ϵ error.
- Leads to a weak learner, if "Not sure" → randomly guess:

$$err_{P(h)} \leq \frac{1}{2}(1 - \epsilon') + \epsilon\epsilon' \leq \frac{1}{2} - \gamma \qquad \text{for } \gamma = \epsilon'\left(\frac{1}{2} - \epsilon\right).$$

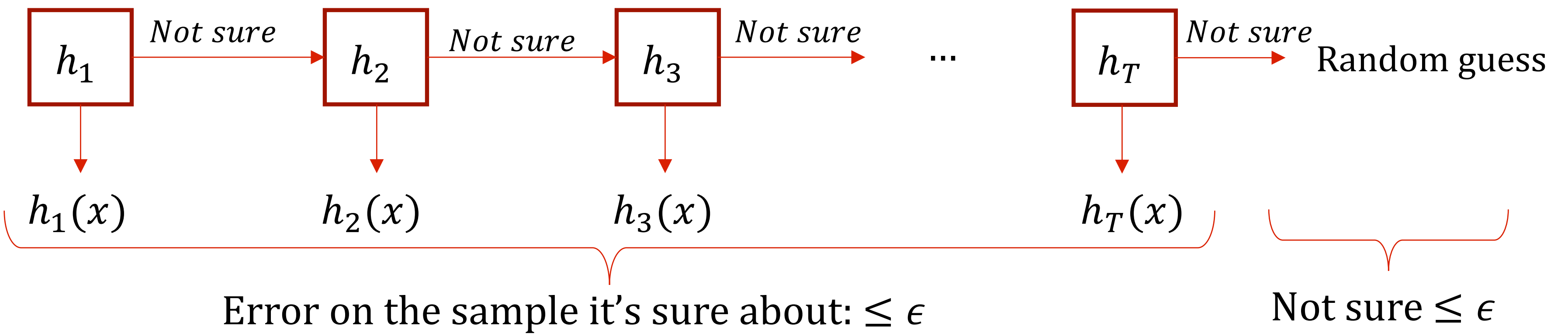
Boosting:

- Start with a weak learner.
- Boost by focusing the distribution on instances the previous learner wasn't sure about.

Warmup Analysis

Boost by a decision list:

- Train h_i on P_i. Let $P_{i+1} \leftarrow P_i | \{x: h_i(x) = \text{“}Not\ sure\text{”}\}$.
- Repeat until the total prob. of the “Not sure” region is ϵ.
- Total error at most 2ϵ.
- It only takes $T = \frac{1}{\epsilon'} \ln(\frac{1}{\epsilon})$ rounds: $(1-\epsilon')^T \le \exp(-\epsilon' T) \le \epsilon$.



Added after class: reason for the above. Conditioned on being sure, we are wrong with prob. $\le \epsilon$. So, the total probability is $\le \epsilon$.

Another way to see this is, prob. of error after each round: $\sum_{t=1}^{T} \epsilon \times \epsilon'(1-\epsilon')^{t-1} \le \epsilon$.

$\Pr_x[h_t(x) \text{ is wrong} \mid h_t(x) \text{ is sure}]$ $\Pr_x[h_t(x) \text{ is sure}]$

A Recipe for Boosting

Boosting Recipe

Input: $(x_1, y_1), \ldots, (x_m, y_m)$ and a weak learning algorithm.

Let $P_1(x_i) = \frac{1}{m}$ for all i. i.e., uniform distribution over samples.

For $t = 1, \ldots, T$

- Learn a weak classifier $h_t \in H$ on distribution P_t.
- Construct P_{t+1} that has **higher weight** compared to P on instance where $h_1, \ldots, h_t$ didn't perform well.

Specify these weights

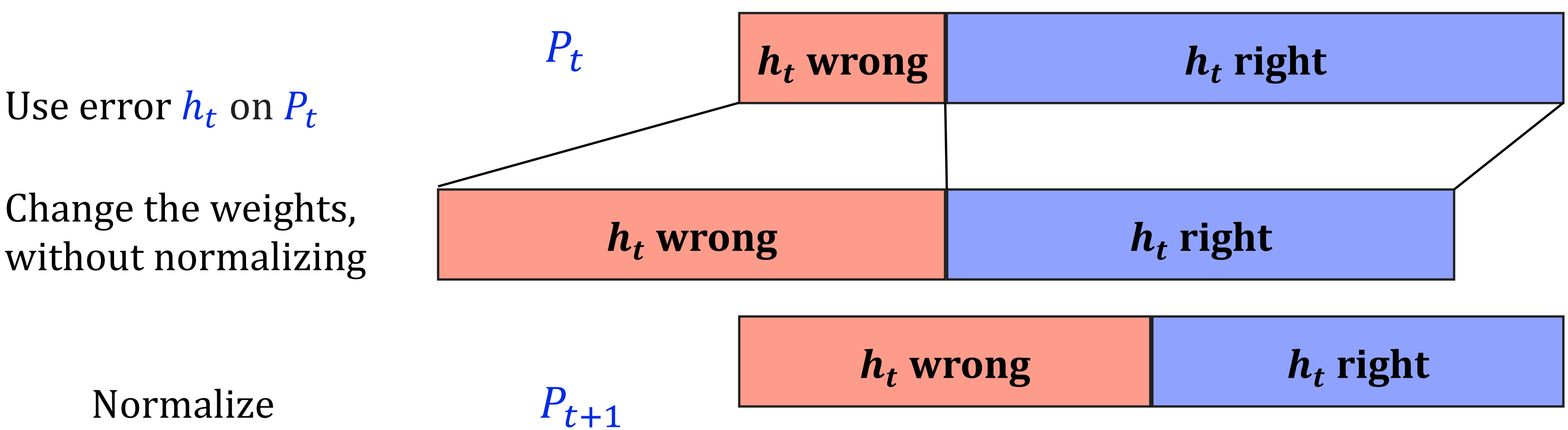
Output the final hypothesis

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

Constructing P_{t+1}

Increase the weight of x_i if h_t made a mistake on it. Decrease the weight if h_t was correct.

- Don't want to cut the weight to 0
  → h_{t+1} could be *arbitrarily bad* on where h_t was good.
  → The majority vote could be bad.
- Change the weights, so that h_t would have head error exactly 0.5



Constructing P_{t+1}

Constructing the next distribution

Let $\epsilon_t = \Pr_{x_i \sim P_t}[h_t(x_i) \neq y_i]$ and let $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$. Let

$$P_{t+1}(x_i) = \frac{P_t(x_i)\exp(-\alpha_t\, y_i\, h_t(x_i))}{Z_t}$$

Where $Z_t = \sum_i P_t(x_i)\exp(-\alpha_t\, y_i\, h_t(x_i))$ is the normalization factor.

$$P_{t+1}(x_i) = \begin{cases} \frac{P_t(x_i)}{Z_t}\exp(-\alpha_t) & \text{if } y_i = h_t(x_i) \\ \frac{P_t(x_i)}{Z_t}\exp(+\alpha_t) & \text{if } y_i \neq h_t(x_i) \end{cases}$$

Weight of P_t on **correct** points

Weight on $h_t(x_i) = y_i$: $\quad \frac{1}{Z_t}(1-\epsilon_t)\ \exp\left(-\frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\right) = \frac{1}{Z_t}(1-\epsilon_t)\left(\frac{\epsilon_t}{1-\epsilon_t}\right)^{1/2} = \frac{\sqrt{\epsilon_t(1-\epsilon_t)}}{Z_t}$

Weight of P_t on **incorrect** points

Weight on $h_t(x_i) \neq y_i$: $\quad \frac{1}{Z_t}\epsilon_t\ \exp\left(\frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\right) = \frac{1}{Z_t}\epsilon_t\left(\frac{1-\epsilon_t}{\epsilon_t}\right)^{1/2} = \frac{\sqrt{\epsilon_t(1-\epsilon_t)}}{Z_t}$

Adaptive Boosting

AdaBoost Algorithm

Input: $(x_1, y_1), \dots, (x_m, y_m)$ and a weak learning algorithm.

Let $P_1(x_i) = \frac{1}{m}$ for all i. i.e., uniform distribution over samples.

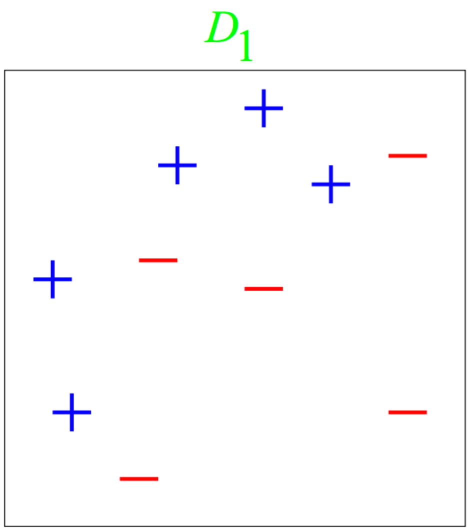
For $t = 1, \dots, T$

- Learn a weak classifier $h_t \in H$ on distribution P_t.
- Let $\epsilon_t = \Pr_{x_i \sim P_t}[h_t(x_i) \neq y_i]$ and let $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$.
- $P_{t+1}(x_i) = \frac{P_t(x_i)\exp(-\alpha_t\, y_i\, h_t(x_i))}{Z_t}$

Output the final hypothesis

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

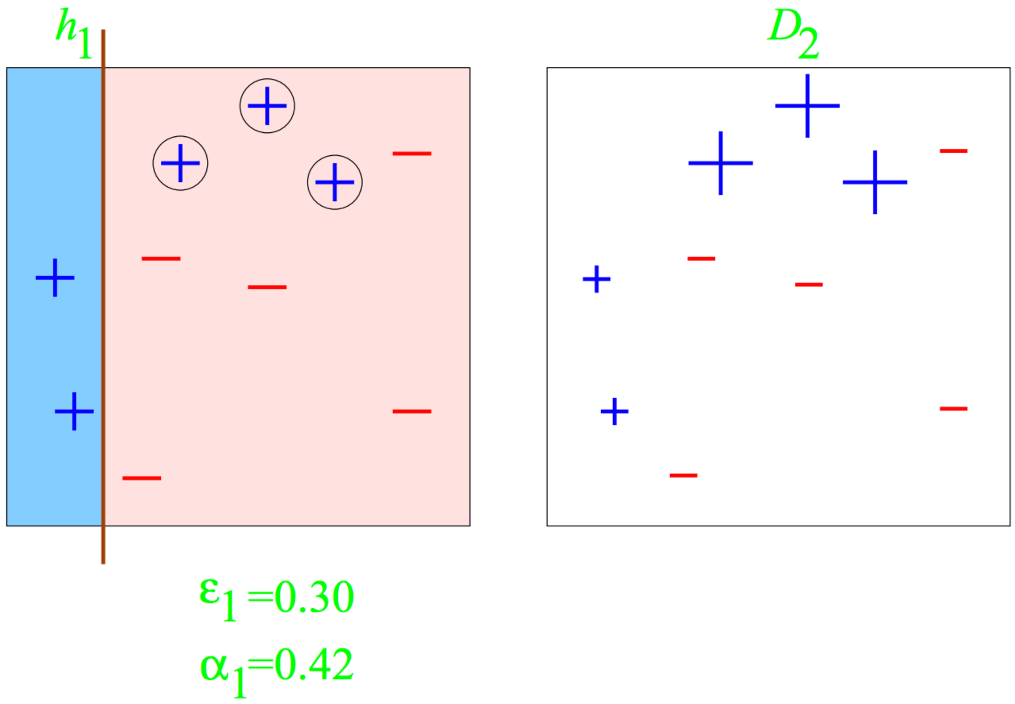
Example



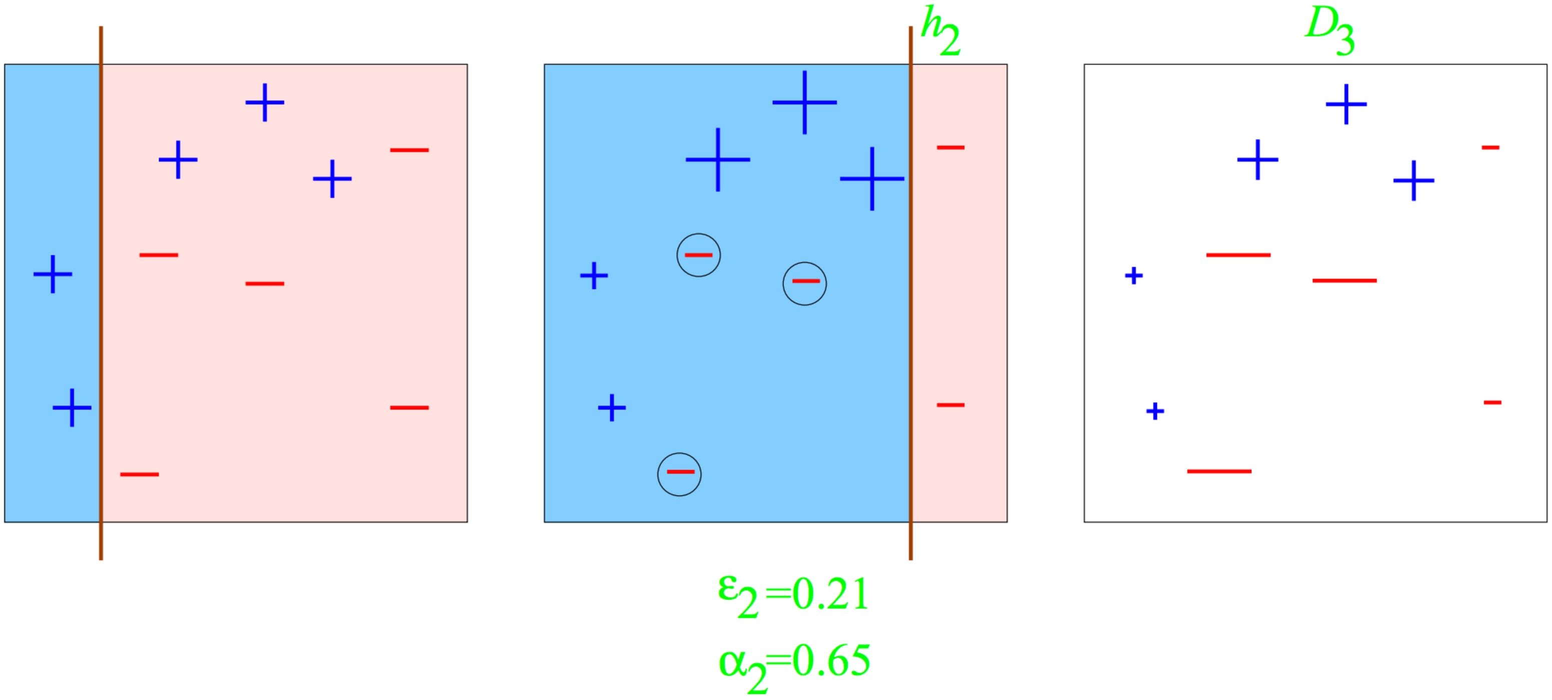
Assume that the weak learner return vertical or horizontal half-spaces (that's the H).

Example from Schapire NeurIPS's 03 Tutorial

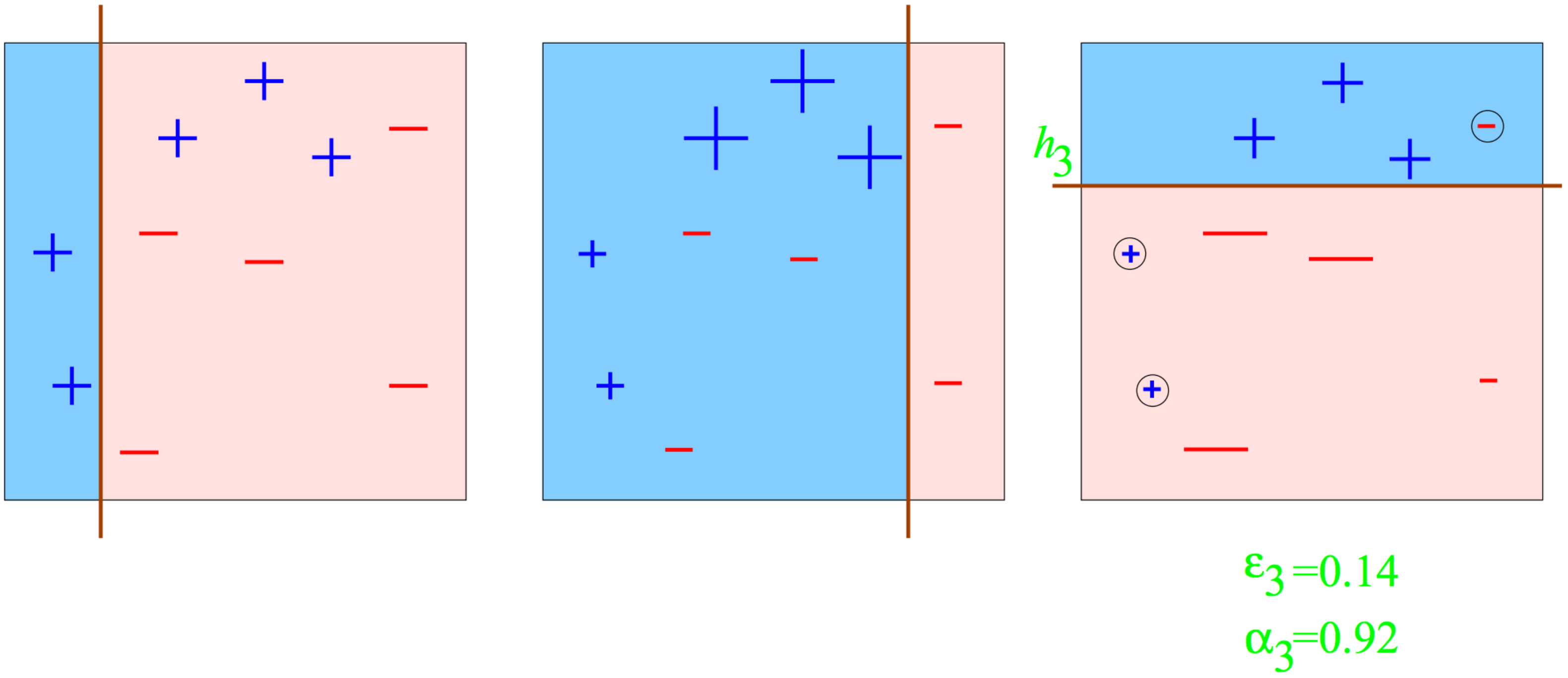
Round 1



Round 2

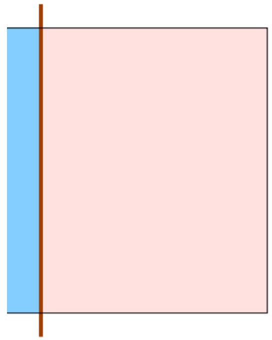


Round 3

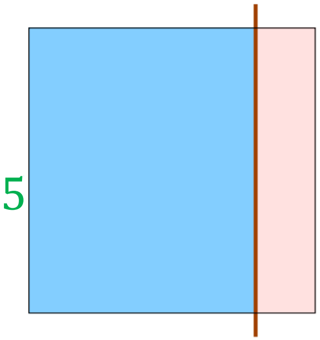


The combined classifier

$h_{final} = sign\Big($ 0.42



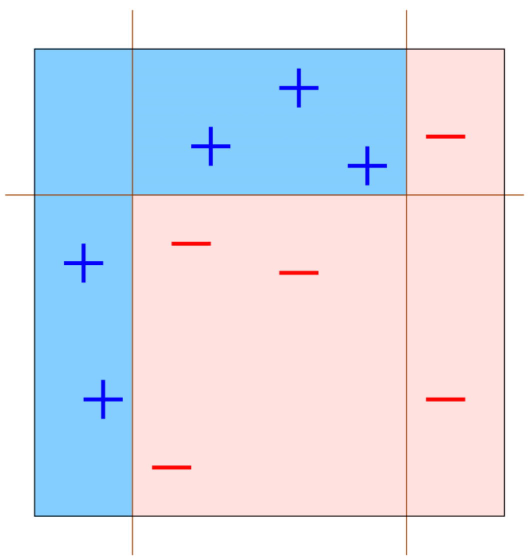
$+0.65$



$+0.92$

$\Big)$

$=$



Bounding the Sample Error

Theorem: AdaBoost's training error

Let $\gamma_t = \frac{1}{2} - \epsilon_t$. For any T, $h_{final}(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$ has training error

$$err_S(h_{final}) \leq \exp\left(-2\sum_{t=1}^{T} \gamma_t^2\right)$$

So, for weak learners where $\gamma_t > \gamma$, and $\mathrm{T} = O\left(\frac{1}{\gamma^2}\ln(\frac{1}{\epsilon})\right)$ we have $err_S(h_{final}) \leq \epsilon$.

Ada(ptive)**Boost:**

- Adaptive: We don't need to know γ or T before we start.
- Can adapt to γ_t.
- Automatically better when $\gamma_t \gg \gamma$.
- Practical algorithm.

Generalization Error

We gave a guarantee that the sample error is at most $err_S(H) \leq \epsilon$.
What about generalization?

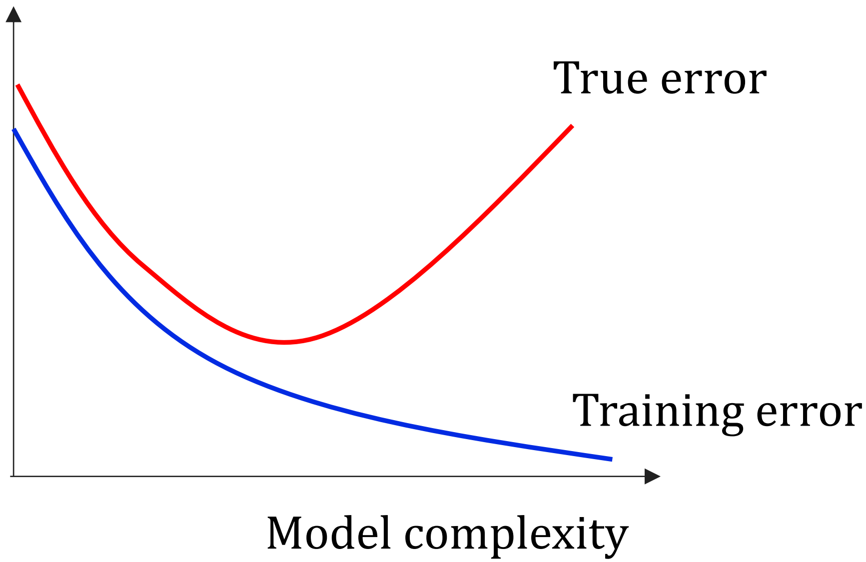
- h_{final} is a combination of T hypothesis $h_1, \dots, h_T \in H$.
- $h_{final} \notin H$ possibly, but it's still structured.
- Recall from Homework 3
  - → Combination of T hypothesis from H has a bounded Growth function.
  - → **Roughly speaking:** This means h_{final} comes from a class of with VC dimension $\tilde{O}(T\ \mathrm{VCDim}(H))$.

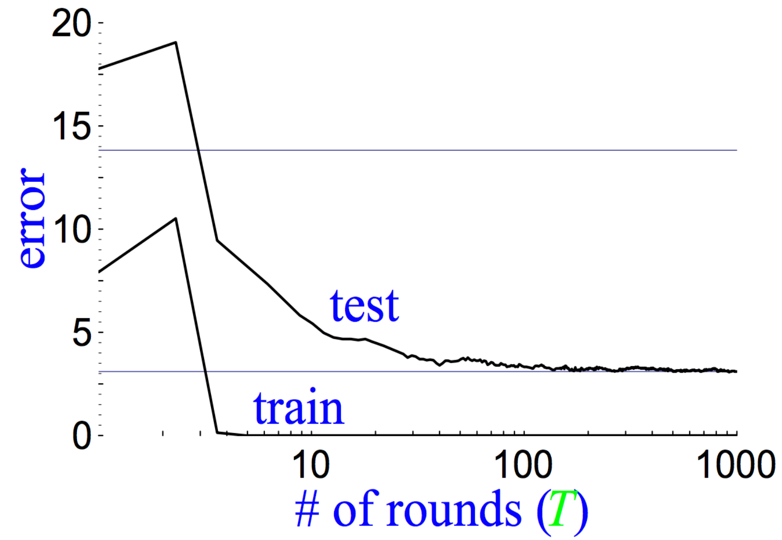
Theorem: AdaBoost's true error

When S has $\tilde{\Omega}\left(\frac{VCDim(H)}{\gamma^2 \epsilon}\right)$ many samples, then $err_P(h_{final}) \leq \epsilon$.

Better Generalization Guarantee

Last slide: VC dimension $\tilde{O}(T\ \mathrm{VCDim}(H))$
→ Keep T small. As T increases there is a chance of overfitting.





Our first guess!

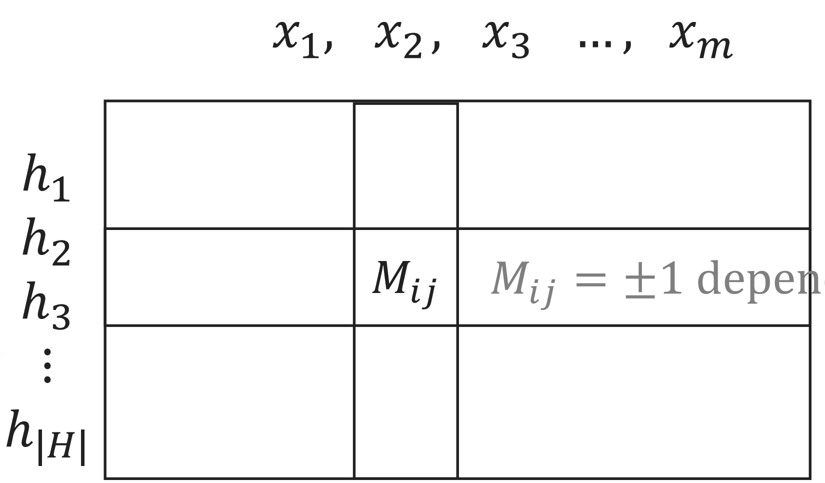
Actual run of AdaBoost.

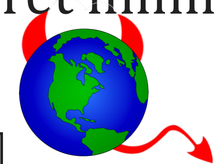
Cool theory for proving why AdaBoost doesn't overfit.

Boosting & Regret Minimization

Schapire and Fruend also gave online learning algorithms (last lecture).

Connection between boosting and regret minimization





Robert Schapire

Yoav Fruend

For every distribution P over the columns, there is a row with expected payoff $\geq \frac{1}{2} + \gamma$.

➔ **Boosting:** Distribution Q over $h_1, h_2, \ldots$ that is $\geq \frac{1}{2} + \gamma$ for every x_i.

➔ Regret minimization against an adversary who is best responding results in the sequence $h_1, h_2, \ldots$

Optional Material

Ensemble Methods

Meta learning algorithms that call multiple algorithms to improve learning performance.

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

Boosting: Take one sample set S, learn h_t for different weight on these samples. Take α_t-weighted majority vote.

→ Improve training error of the weak classifiers h_t's.

Bagging

Even if the training error is already good (bias) , can we decrease the variance?

$\alpha_t = 1$

h_t trained on subsamples

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

Bagging (Bootstrap Aggregating)

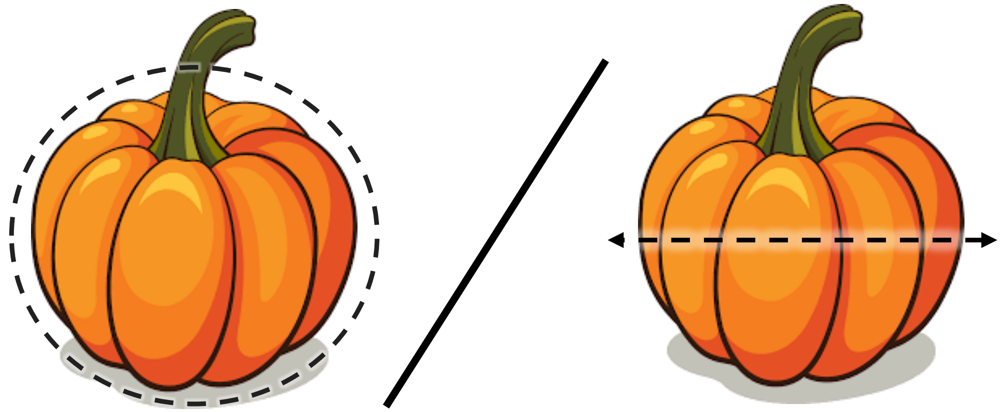
Input: $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and any learning algorithm.

For $t = 1, \ldots, T$

- $S_t =$ sample with replacement from S.
- $h_t =$ train on the sample set S_t.

Return $\text{sign}(\sum_{t=1}^{T} h_t(x))$

Enjoy the



Happy Thanksgiving!