CSC 411: Introduction to Machine Learning

Lecture 5: Ensembles II

Mengye Ren and Matthew MacKay

University of Toronto

Boosting

- Recall that an *ensemble* is a set of predictors whose individual decisions are combined in some way to classify new examples.
- (Previous lecture) **Bagging**: Train classifiers independently on random subsets of the training data.
- (This lecture) **Boosting**: Train classifiers sequentially, each time focusing on training data points that were previously misclassified.
- Let's start with the concepts of weighted training sets and weak learner/classifier (or base classifiers).

Weighted Training set

- The misclassification error $\sum_{n=1}^{N} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$ error weights each training example equally

Weighted Training set

- The misclassification error $\sum_{n=1}^{N} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$ error weights each training example equally
- Key idea: can learn a classifier using different costs (aka weights) for examples
  - Classifier "tries harder" on examples with higher cost

Weighted Training set

- The misclassification error $\sum_{n=1}^{N} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$ error weights each training example equally
- Key idea: can learn a classifier using different costs (aka weights) for examples
  - Classifier "tries harder" on examples with higher cost
- Change cost function:

$$\sum_{n=1}^{N} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}] \quad \text{becomes} \quad \sum_{n=1}^{N} w^{(n)} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$$

Weighted Training set

- The misclassification error $\sum_{n=1}^{N} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$ error weights each training example equally
- Key idea: can learn a classifier using different costs (aka weights) for examples
  - Classifier "tries harder" on examples with higher cost
- Change cost function:

$$\sum_{n=1}^{N} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}] \quad \text{becomes} \quad \sum_{n=1}^{N} w^{(n)} \mathbb{I}[h(x^{(n)}) \neq t^{(n)}]$$

- Usually require each $w^{(n)} > 0$ and $\sum_{n=1}^{N} w^{(n)} = 1$

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 $0.5 + \epsilon$ in binary label case.

Weak Learner/Classifier

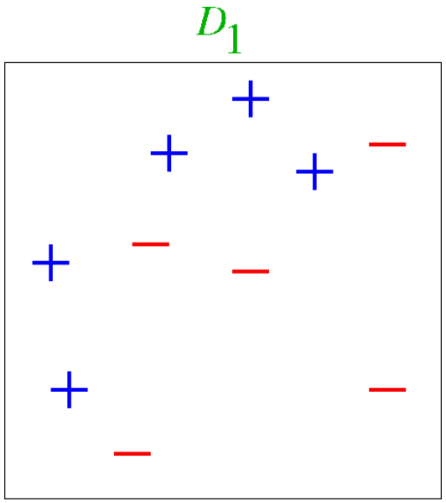
- (Informal) Weak learner is a learning algorithm that outputs a hypothesis (e.g., a classifier) that performs slightly better than chance, e.g., it predicts the correct label with probability $0.5 + \epsilon$ in binary label case.
  - For weighted training set, gets slightly less than 0.5 error

Weak Learner/Classifier

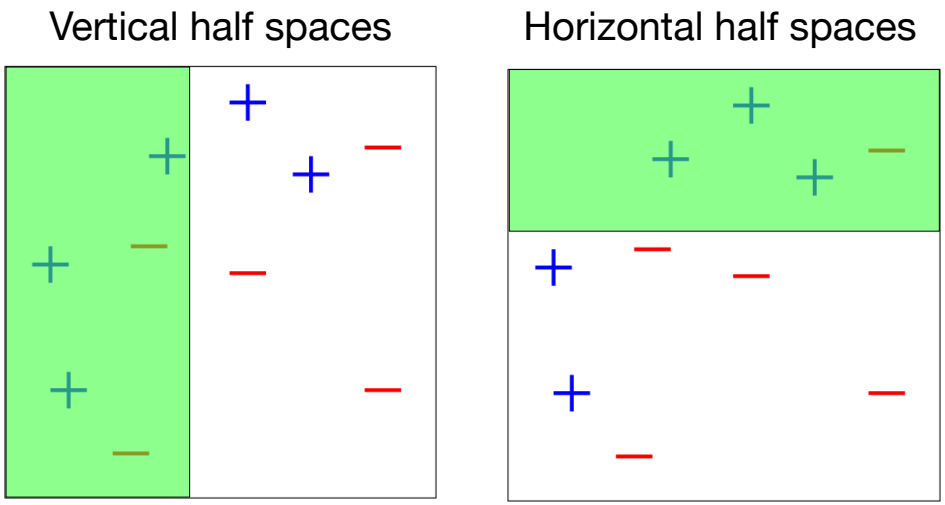
- (Informal) Weak learner is a learning algorithm that outputs a hypothesis (e.g., a classifier) that performs slightly better than chance, e.g., it predicts the correct label with probability $0.5 + \epsilon$ in binary label case.
  - For weighted training set, gets slightly less than 0.5 error
- We are interested in weak learners that are *computationally* efficient.
  - Decision trees
  - Even simpler: Decision Stump: A decision tree with only a single split

[Formal definition of weak learnability has quantifies such as "for any distribution over data" and the requirement that its guarantee holds only probabilistically.]

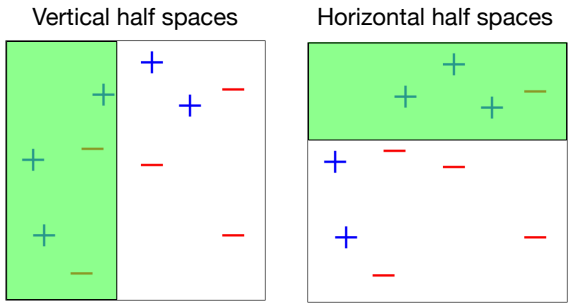
Weak Classifiers



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 $\{w^{(1)}, \ldots, w^{(N)}\}$ (with $\sum_{n=1}^{N} w^{(n)} = 1$ and $w^{(n)} \geq 0$)

$$\text{err} = \sum_{n=1}^{N} w^{(n)} \mathbb{I}[h(\mathbf{x}^{(n)}) \neq t^{(n)}]$$

  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?

AdaBoost (Adaptive Boosting)

- Boosting: Train classifiers sequentially, each time assigning higher weight to training data points that were previously misclassified.

AdaBoost (Adaptive Boosting)

- Boosting: Train classifiers sequentially, each time assigning higher weight to training data points that were previously misclassified.
- Key steps of AdaBoost:

AdaBoost (Adaptive Boosting)

- Boosting: Train classifiers sequentially, each time assigning higher weight to training data points that were previously misclassified.
- 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.

AdaBoost (Adaptive Boosting)

- Boosting: Train classifiers sequentially, each time assigning higher weight to training data points that were previously misclassified.
- 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.

AdaBoost (Adaptive Boosting)

- Boosting: Train classifiers sequentially, each time assigning higher weight to training data points that were previously misclassified.
- 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.

AdaBoost (Adaptive Boosting)

- Boosting: Train classifiers sequentially, each time assigning higher weight to training data points that were previously misclassified.
- 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. We repeat the process many times.

AdaBoost (Adaptive Boosting)

- Boosting: Train classifiers sequentially, each time assigning higher weight to training data points that were previously misclassified.
- 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. 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 Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$, 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

AdaBoost Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$, 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 $H(x)$

AdaBoost Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$, 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: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$

AdaBoost Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$, 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: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$
- For $t = 1, \ldots, T$

AdaBoost Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$, 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: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$
- For $t = 1, \ldots, T$
  - Fit a classifier to data using weighted samples ($h_t \leftarrow \mathit{WeakLearn}(\mathcal{D}_N, \mathbf{w})$), e.g.,

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

AdaBoost Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$, 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: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$
- For $t = 1, \ldots, T$
  - Fit a classifier to data using weighted samples ($h_t \leftarrow \mathit{WeakLearn}(\mathcal{D}_N, \mathbf{w})$), e.g.,

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

  - Compute weighted error $\text{err}_t = \frac{\sum_{n=1}^N w^{(n)} \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}}{\sum_{n=1}^N w^{(n)}}$

AdaBoost Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$, 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: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$
- For $t = 1, \ldots, T$
  - Fit a classifier to data using weighted samples ($h_t \leftarrow \mathit{WeakLearn}(\mathcal{D}_N, \mathbf{w})$), e.g.,

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

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

AdaBoost Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$, 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: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$
- For $t = 1, \ldots, T$
  - Fit a classifier to data using weighted samples ($h_t \leftarrow \mathit{WeakLearn}(\mathcal{D}_N, \mathbf{w})$), e.g.,

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

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

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

AdaBoost Algorithm

- Input: Data $\mathcal{D}_N = \{\mathbf{x}^{(n)}, t^{(n)}\}_{n=1}^N$, 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: $w^{(n)} = \frac{1}{N}$ for $n = 1, \ldots, N$
- For $t = 1, \ldots, T$
  - Fit a classifier to data using weighted samples ($h_t \leftarrow \mathit{WeakLearn}(\mathcal{D}_N, \mathbf{w})$), e.g.,

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

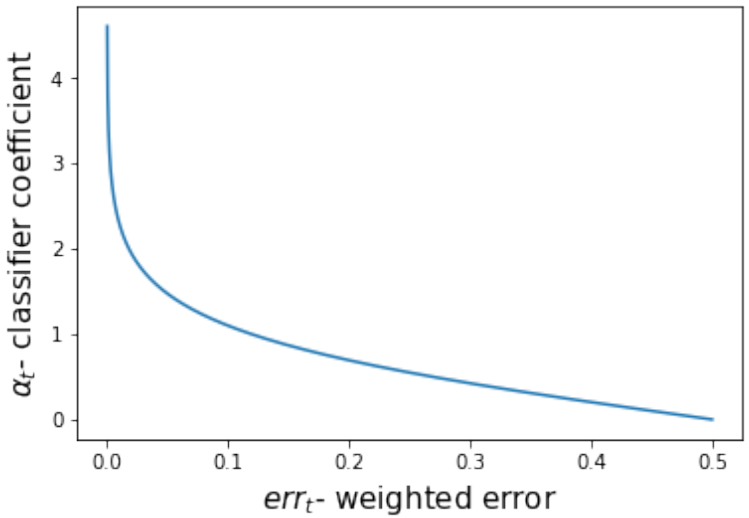
  - Compute weighted error $\text{err}_t = \frac{\sum_{n=1}^N w^{(n)} \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}}{\sum_{n=1}^N w^{(n)}}$
  - Compute classifier coefficient $\alpha_t = \frac{1}{2} \log \frac{1-\text{err}_t}{\text{err}_t}$
  - Update data weights

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

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

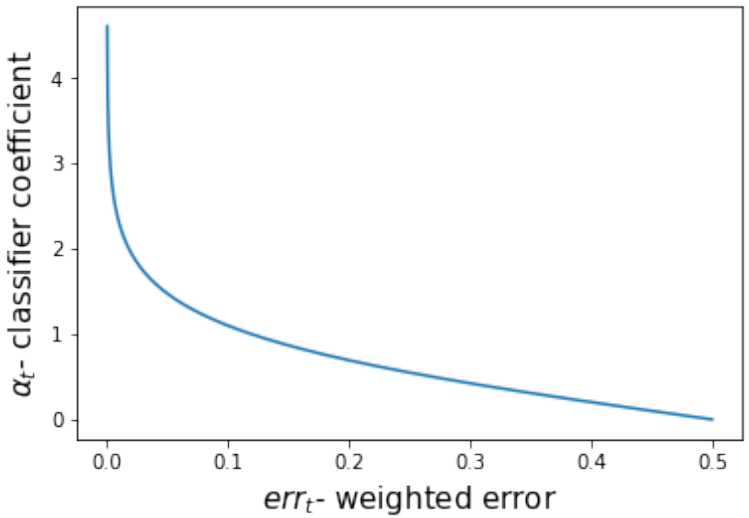
Weighting Intuition

- Recall: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$ where $\alpha_t = \frac{1}{2} \log \frac{1-\text{err}_t}{\text{err}_t}$



Weighting Intuition

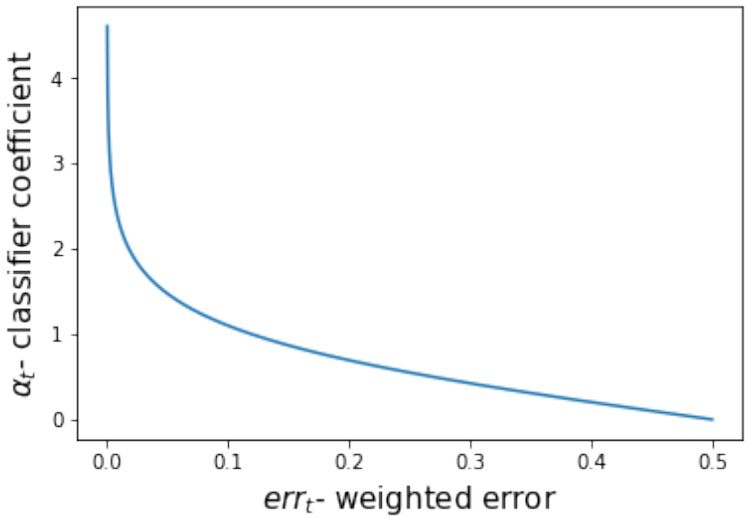
- Recall: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$ where $\alpha_t = \frac{1}{2} \log \frac{1-\text{err}_t}{\text{err}_t}$



- Weak classifiers which get lower weighted error get more weight in the final classifier

Weighting Intuition

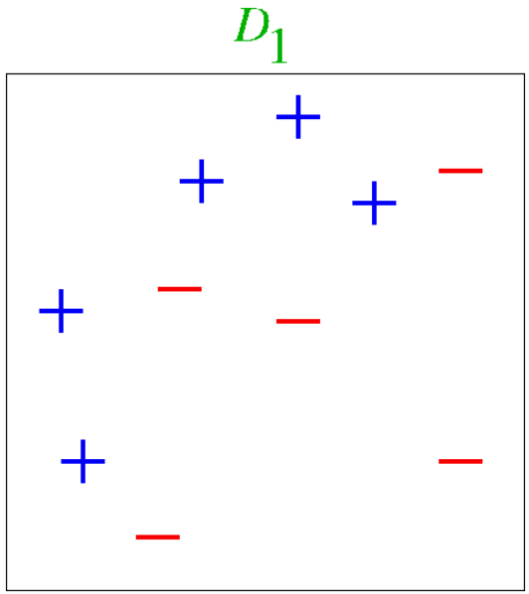
- Recall: $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$ where $\alpha_t = \frac{1}{2} \log \frac{1-\text{err}_t}{\text{err}_t}$



- Weak classifiers which get lower weighted error get more weight in the final classifier
- Also: $w^{(n)} \leftarrow w^{(n)} \exp\left(2\alpha_t \mathbb{I}\{h_t(\mathbf{x}^{(n)}) \neq t^{(n)}\}\right)$
  - If $err_t \approx 0$, α_t high so misclassified examples more attention
  - If $err_t \approx 0.5$, α_t low so misclassified examples are not emphasized

AdaBoost Example

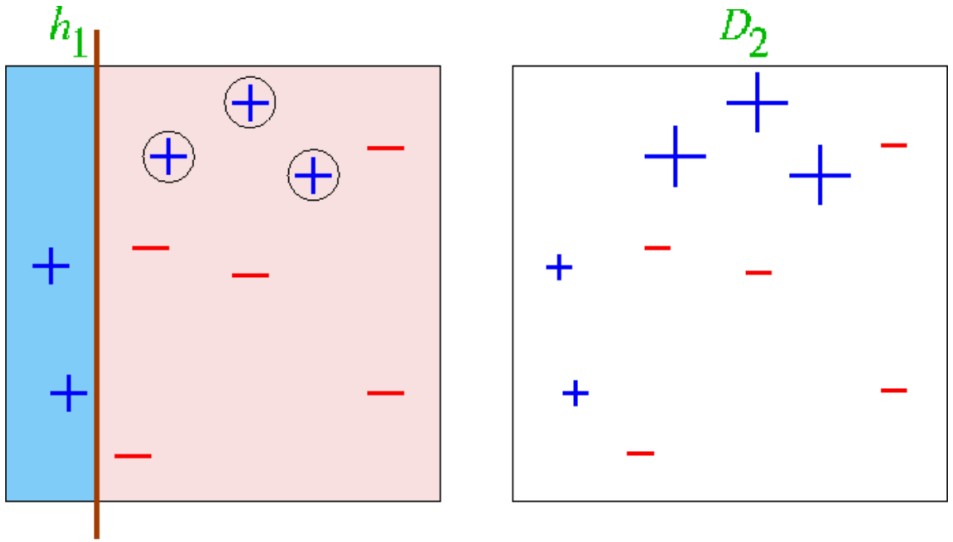
- Training data



[Slide credit: Verma & Thrun]

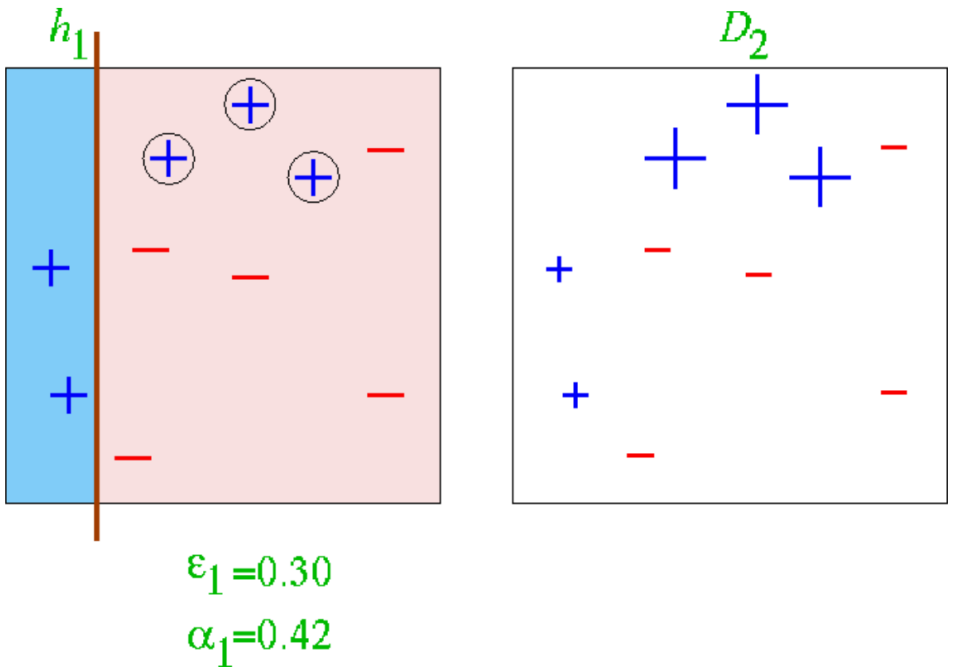
AdaBoost Example

- Round 1



AdaBoost Example

- Round 1



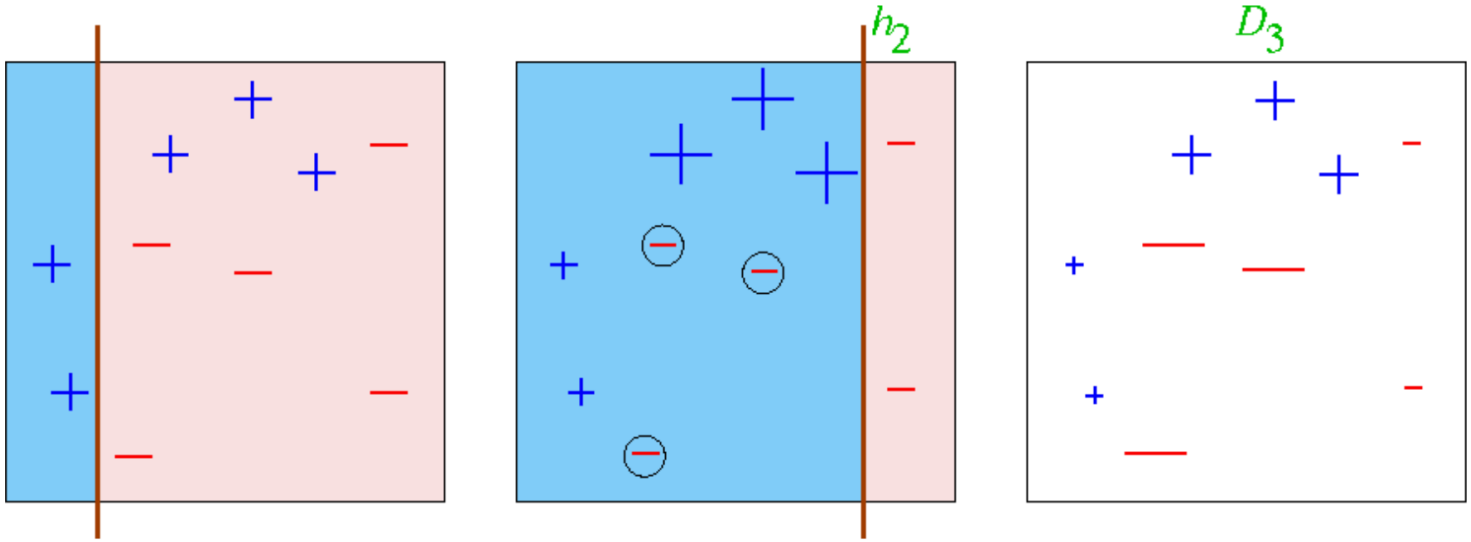
$$\mathbf{w} = \left(\frac{1}{10}, \ldots, \frac{1}{10}\right) \Rightarrow \text{Train a classifier (using } \mathbf{w}) \Rightarrow \text{err}_1 = \frac{\sum_{n=1}^{10} w^{(n)} \mathbb{I}[h_1(\mathbf{x}^{(n)}) \neq t^{(n)}]}{\sum_{n=1}^{10} w^{(n)}} = \frac{3}{10}$$

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

[Slide credit: Verma & Thrun]

AdaBoost Example

- Round 2



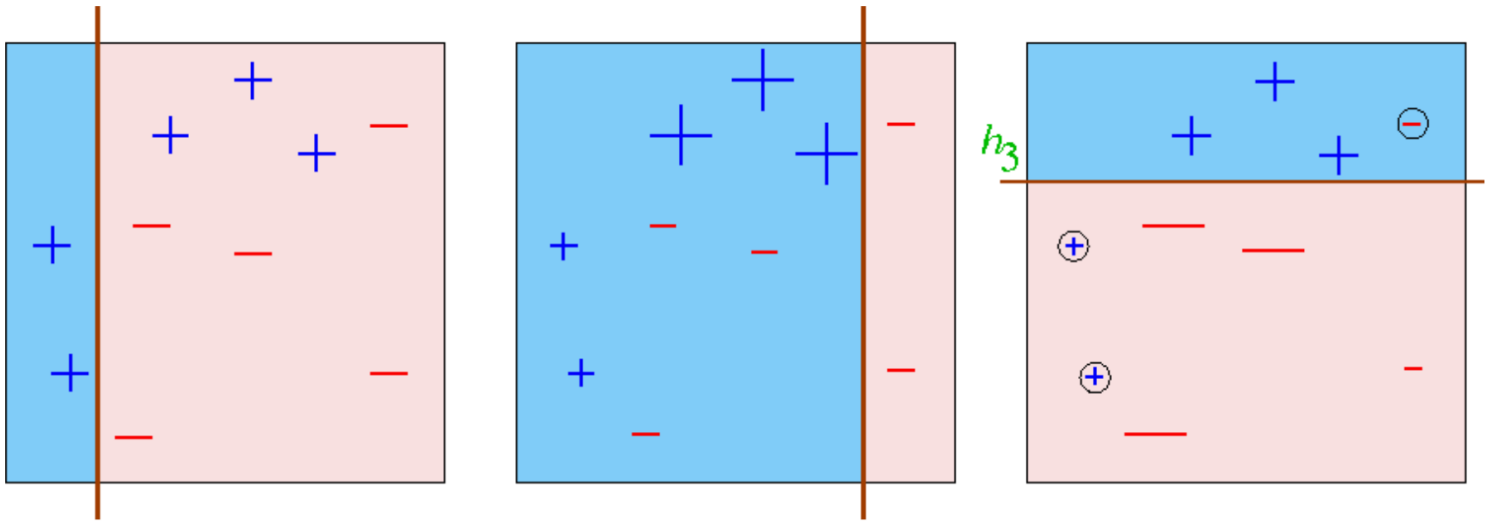
$\mathbf{w} \leftarrow$ new weights $\Rightarrow$ Train a classifier (using $\mathbf{w}$) $\Rightarrow \text{err}_2 = \frac{\sum_{n=1}^{10} w^{(n)} \mathbb{I}\{h_2(\mathbf{x}^{(n)}) \neq t^{(n)}\}}{\sum_{n=1}^{10} w^{(n)}} = 0.21$

$$\Rightarrow \alpha_2 = \frac{1}{2} \log \frac{1 - \text{err}_2}{\text{err}_2} = \frac{1}{2} \log(\frac{1}{0.21} - 1) \approx 0.66 \Rightarrow H(\mathbf{x}) = \text{sign}\left(\alpha_1 h_1(\mathbf{x}) + \alpha_2 h_2(\mathbf{x})\right)$$

[Slide credit: Verma & Thrun]

AdaBoost Example

- Round 3



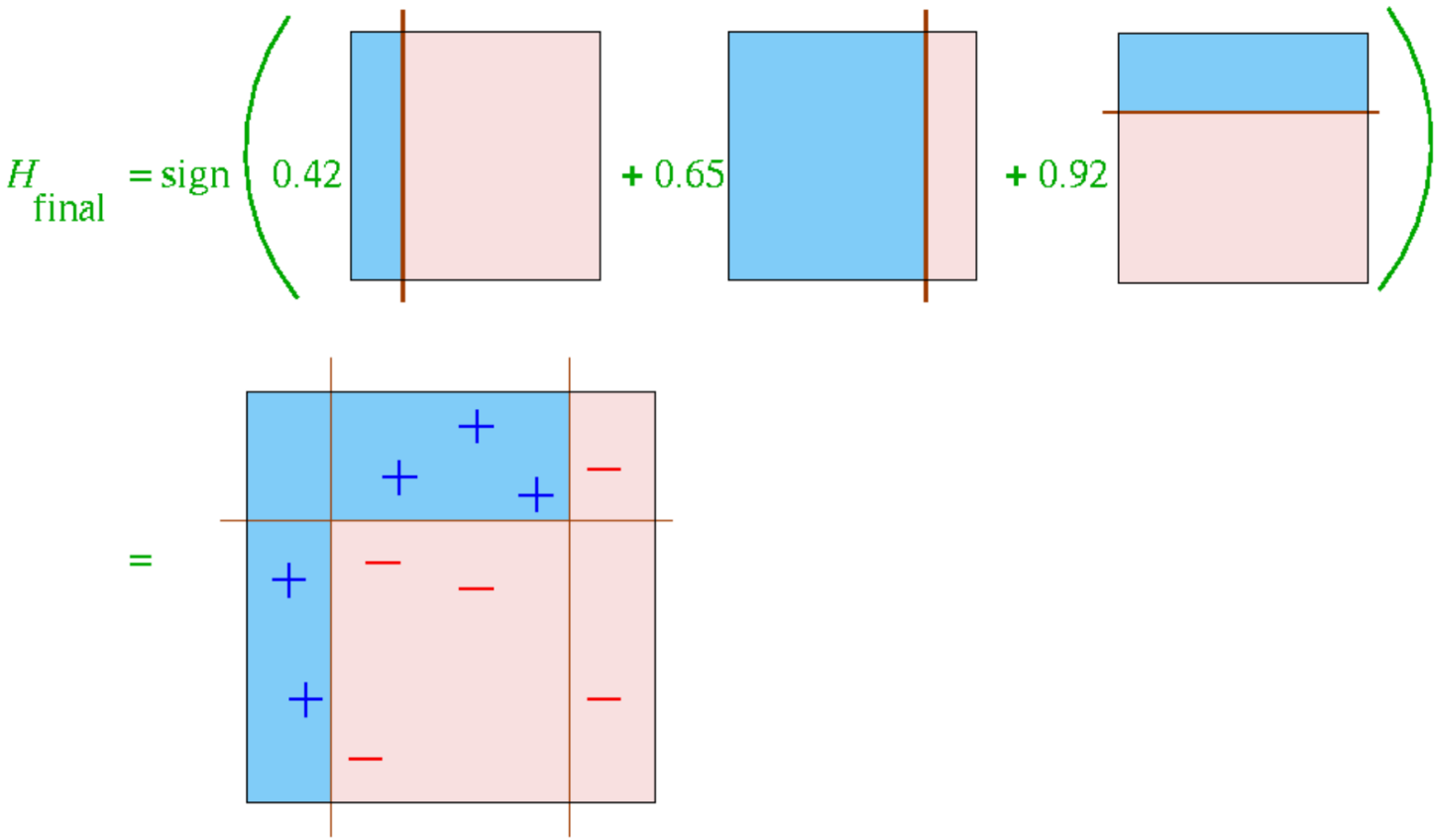
$$\mathbf{w} \leftarrow \text{new weights} \Rightarrow \text{Train a classifier (using } \mathbf{w}) \Rightarrow \text{err}_3 = \frac{\sum_{n=1}^{10} w^{(n)} \mathbb{I}\{h_3(\mathbf{x}^{(n)}) \neq t^{(n)}\}}{\sum_{i=1}^{10} w^{(n)}} = 0.14$$

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

[Slide credit: Verma & Thrun]

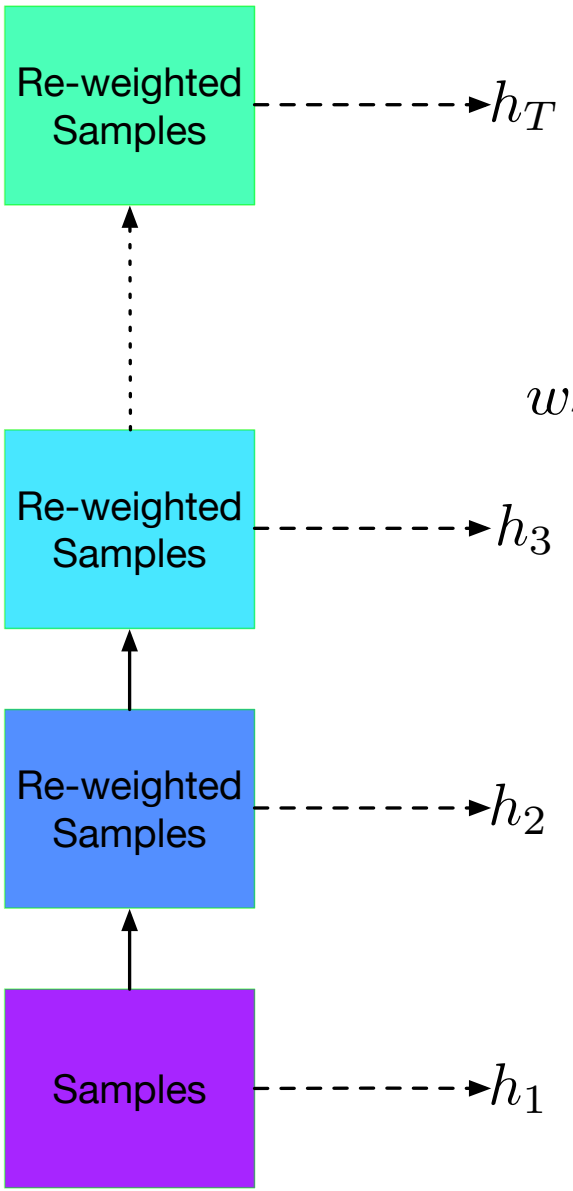
AdaBoost Example

- Final classifier



[Slide credit: Verma & Thrun]

AdaBoost Algorithm



h_T

h_3

h_2

h_1

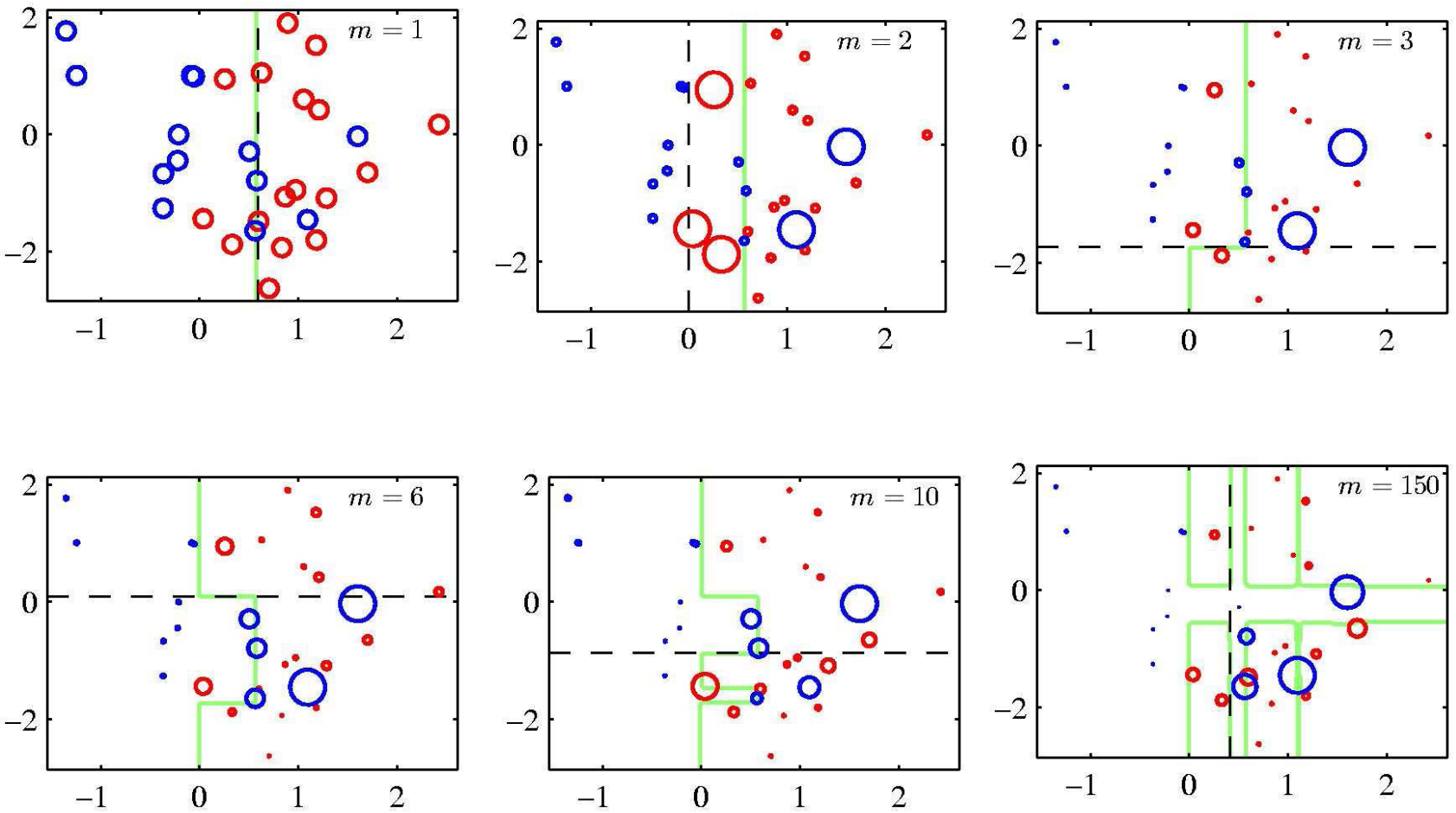
$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

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

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

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

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

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

AdaBoost Minimizes the Training Error

Theorem

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

Generalization Error of AdaBoost

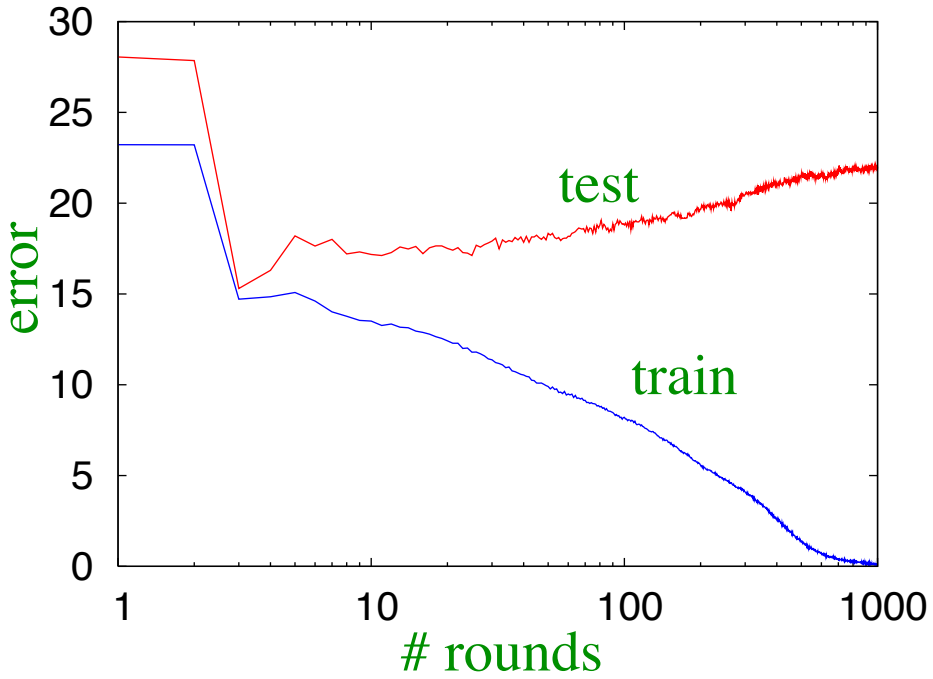
- AdaBoost's training error (loss) converges to zero. What about the test error of H?
- As we add more weak classifiers, the overall classifier H becomes more "complex".

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.

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.

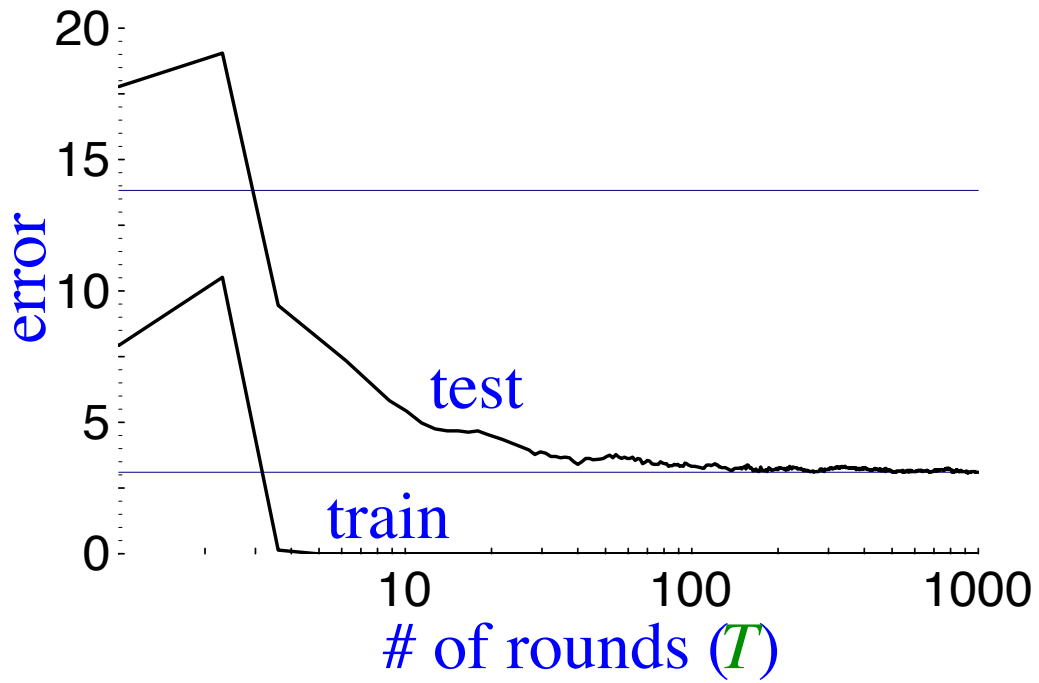


Generalization Error of AdaBoost

- But often it does not!

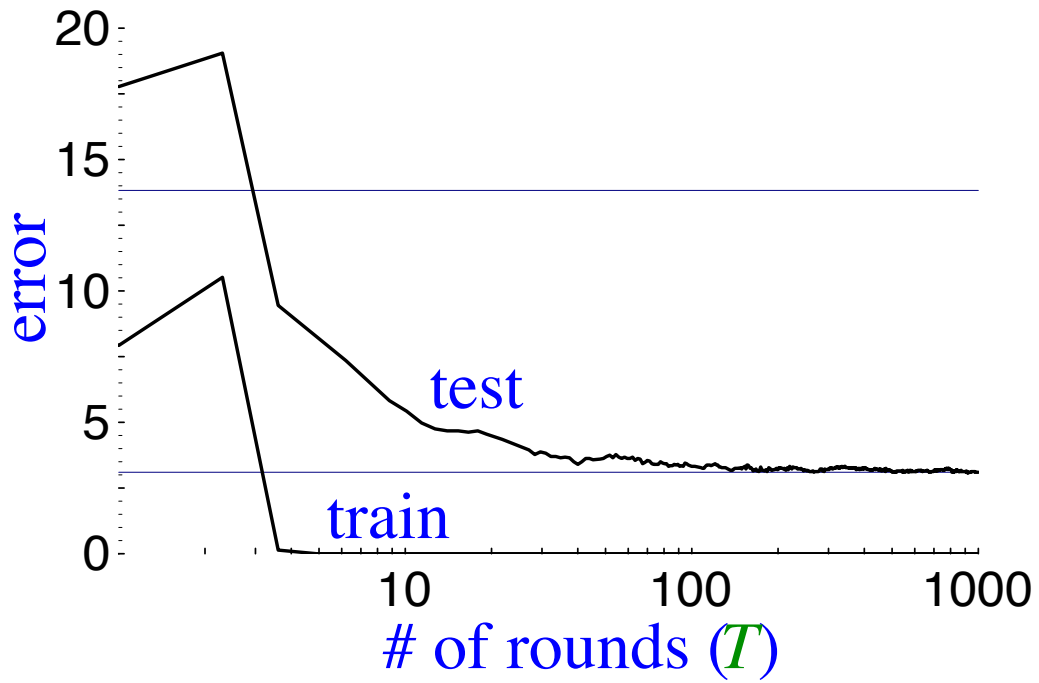
Generalization Error of AdaBoost

- But often it does not!
- Sometimes the test error decreases even after the training error is zero!



Generalization Error of AdaBoost

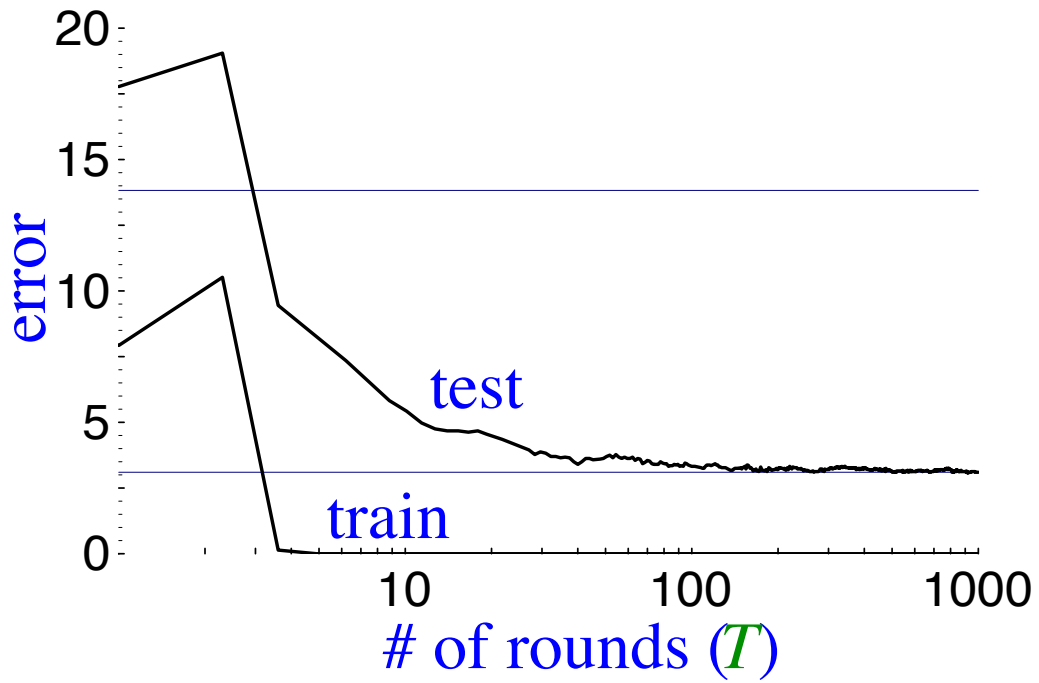
- But often it does not!
- Sometimes the test error decreases even after the training error is zero!



- How does that happen?

Generalization Error of AdaBoost

- But often it does not!
- Sometimes the test error decreases even after the training error is zero!



- How does that happen?
- We will provide an alternative viewpoint on AdaBoost later in the course.

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

AdaBoost for Face Detection

- Famous application of boosting: detecting faces in images
- Viola and Jones created a very fast face detector that can be scanned across a large image to find the faces.
- A few twists on standard algorithm

AdaBoost for Face Detection

- Famous application of boosting: detecting faces in images
- Viola and Jones created a very fast face detector that can be scanned across a large image to find the faces.
- A few twists on standard algorithm

AdaBoost for Face Detection

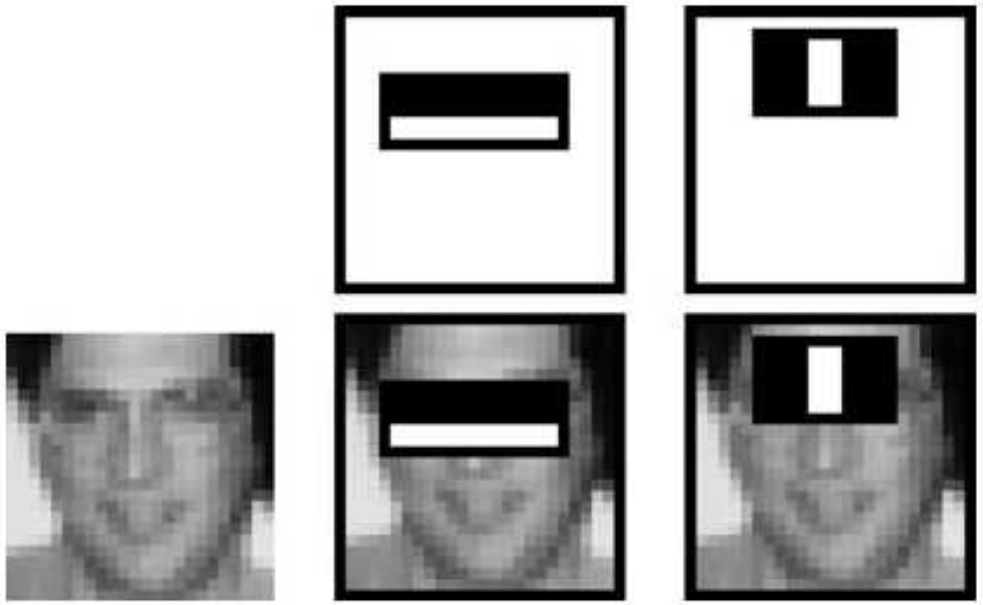
- Famous application of boosting: detecting faces in images
- Viola and Jones created a very fast face detector that can be scanned across a large image to find the faces.
- A few twists on standard algorithm

AdaBoost for Face Detection

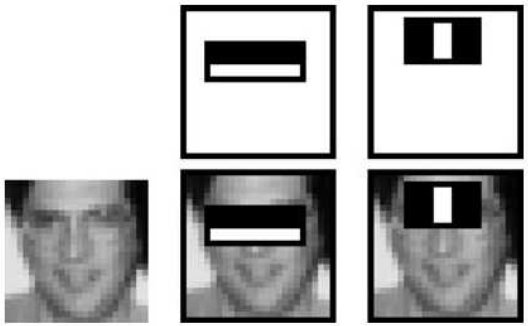
- Famous application of boosting: detecting faces in images
- Viola and Jones created a very fast face detector that can be scanned across a large image to find the faces.
- A few twists on standard algorithm
  - Change loss function for weak learners: false positives less costly than misses

AdaBoost for Face Detection

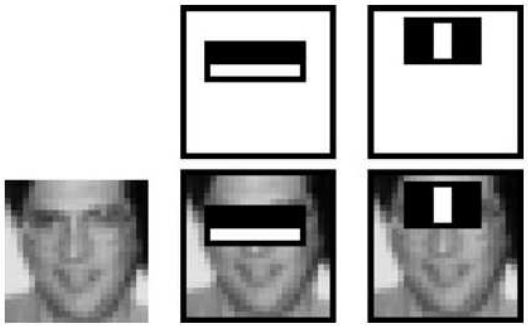
- Famous application of boosting: detecting faces in images
- Viola and Jones created a very fast face detector that can be scanned across a large image to find the faces.
- A few twists on standard algorithm
  - Change loss function for weak learners: false positives less costly than misses
  - Smart way to do inference in real-time (in 2001 hardware)



AdaBoost for Face Recognition

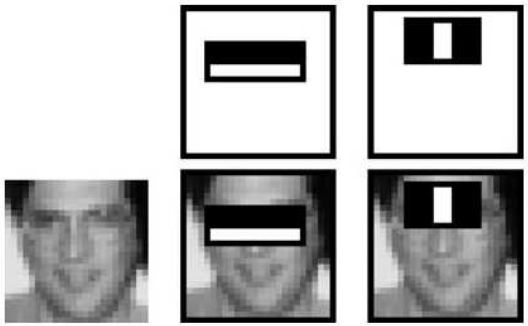


AdaBoost for Face Recognition



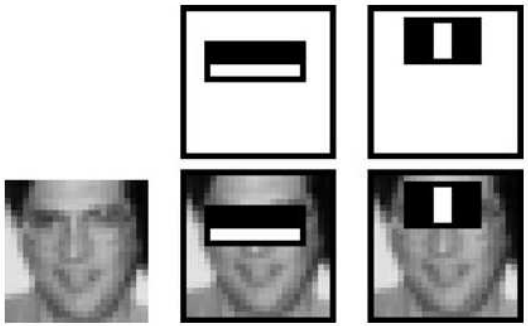
- The base classifier/weak learner just compares the total intensity in two rectangular pieces of the image and classifies based on comparison of this difference to some threshold.

AdaBoost for Face Recognition



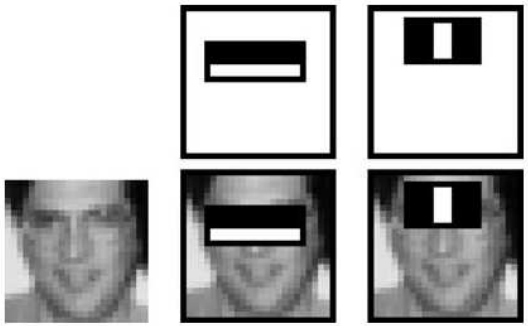
- The base classifier/weak learner just compares the total intensity in two rectangular pieces of the image and classifies based on comparison of this difference to some threshold.
  - There is a neat trick for computing the total intensity in a rectangle in a few operations.

AdaBoost for Face Recognition



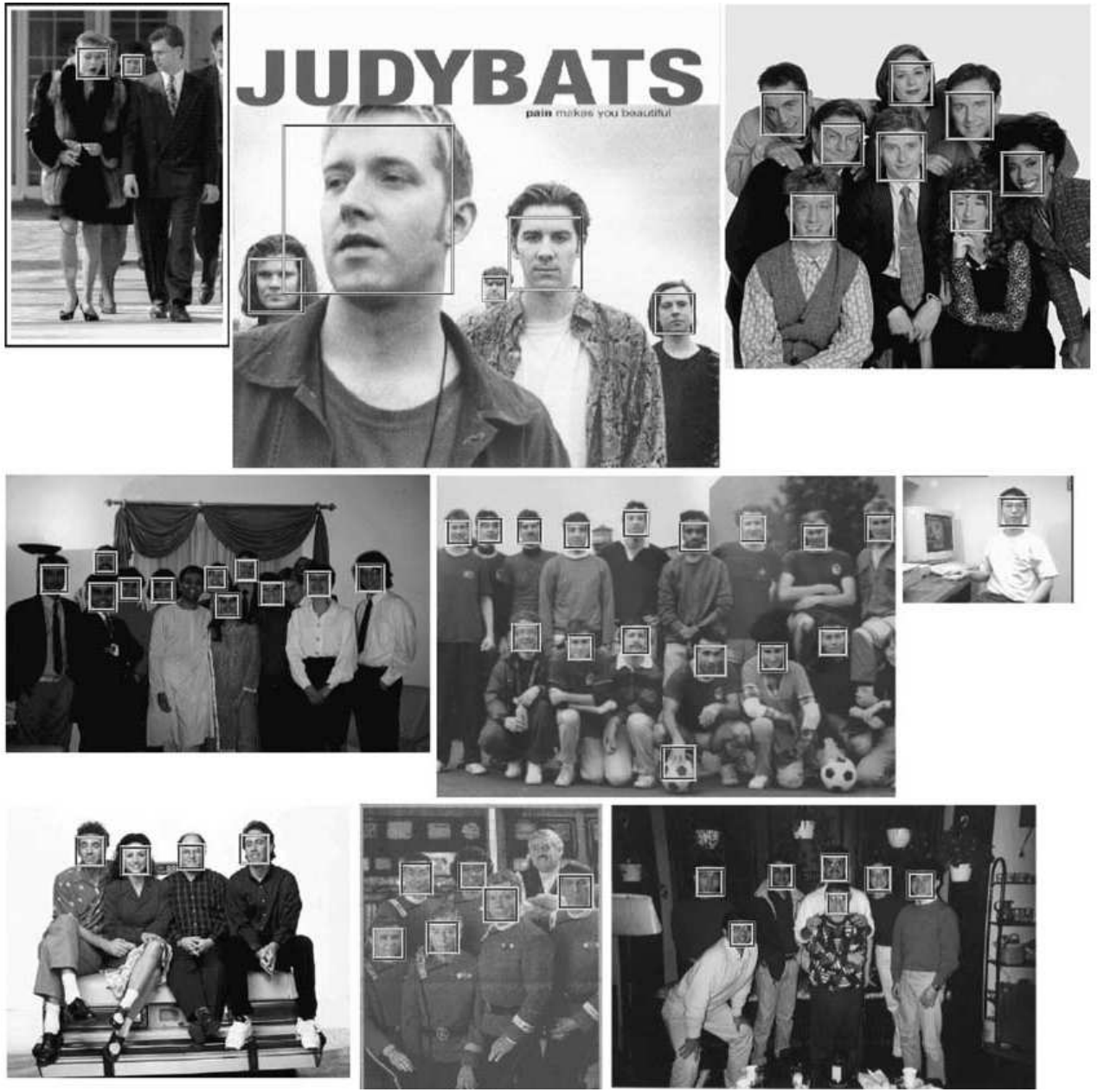
- The base classifier/weak learner just compares the total intensity in two rectangular pieces of the image and classifies based on comparison of this difference to some threshold.
  - There is a neat trick for computing the total intensity in a rectangle in a few operations.
    - So it is easy to evaluate a huge number of base classifiers and they are very fast at runtime.

AdaBoost for Face Recognition



- The base classifier/weak learner just compares the total intensity in two rectangular pieces of the image and classifies based on comparison of this difference to some threshold.
  - There is a neat trick for computing the total intensity in a rectangle in a few operations.
    - So it is easy to evaluate a huge number of base classifiers and they are very fast at runtime.
  - The algorithm adds classifiers greedily based on their quality on the weighted training cases
    - Each classifier uses just one feature

AdaBoost Face Detection Results



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.
- Next Lecture: Linear Regression