Learning Ensembles

290N UCSB, 2015

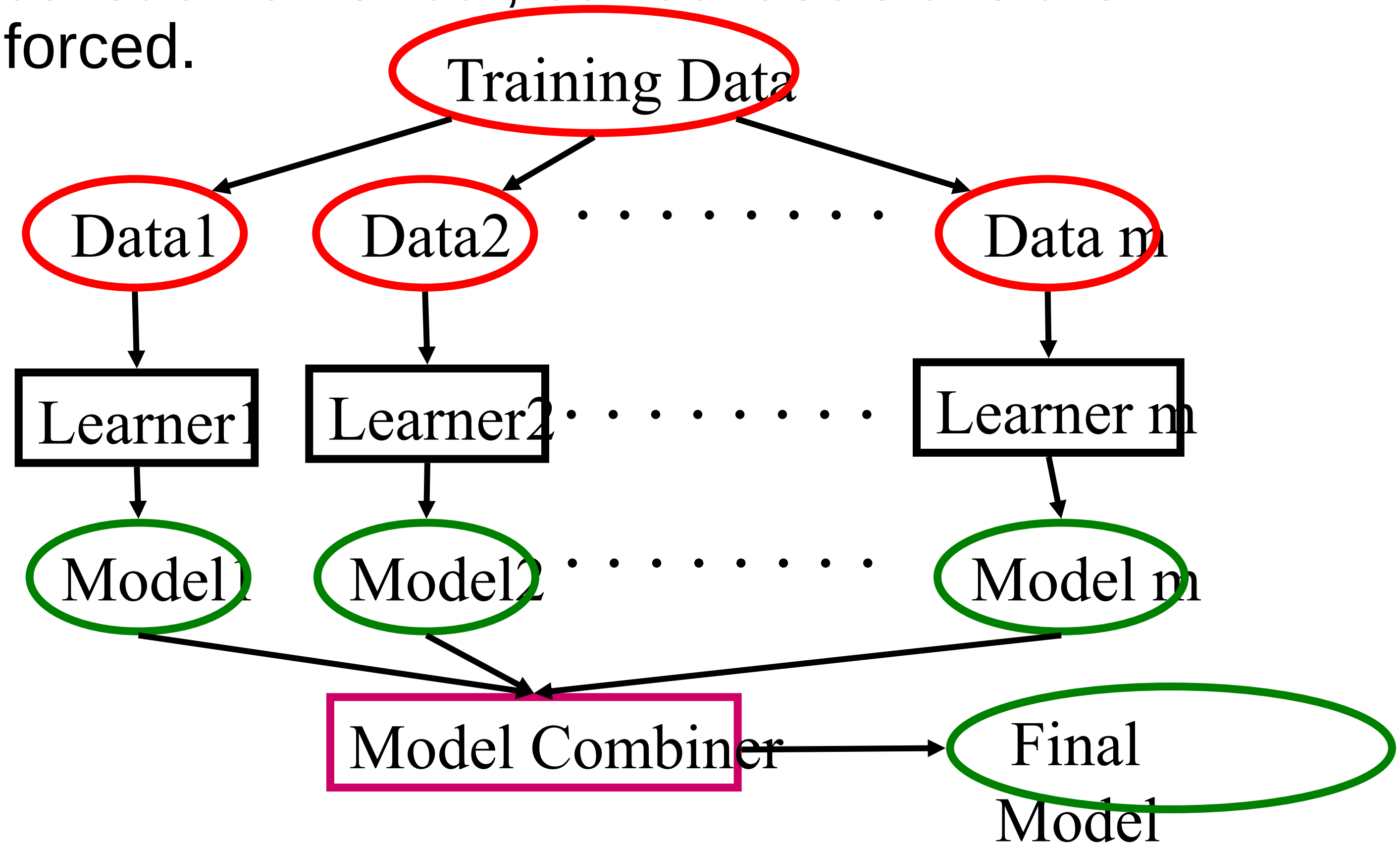
.

Outlines

- Learning Assembles
- Random Forest
- Adaboost

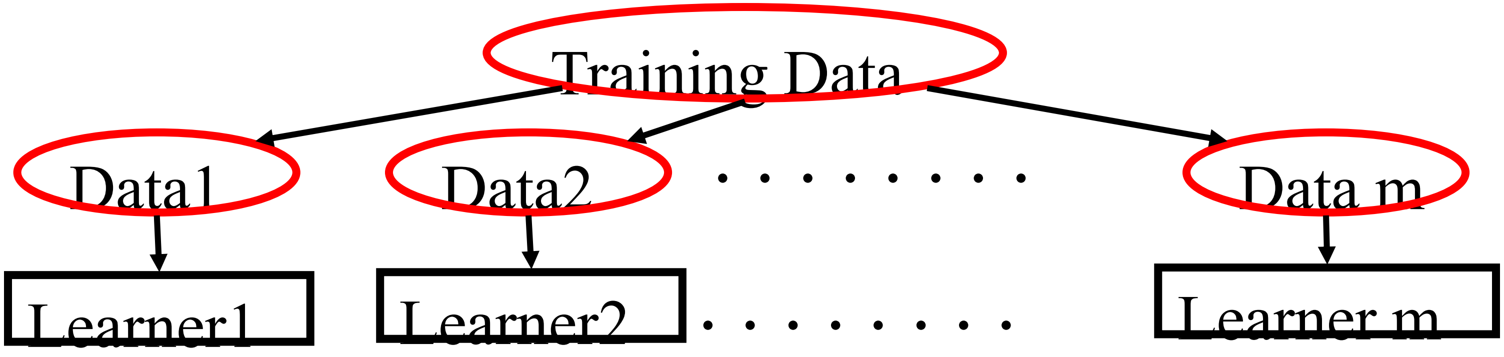
Learning Ensembles

- **Learn multiple classifiers separately**
- **Combine decisions (e.g. using weighted voting)**
- When combing multiple decisions, random errors cancel each other out, correct decisions are reinforced.



Homogenous Ensembles

- **Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.**
  - Data1 ≠ Data2 ≠ … ≠ Data m
  - Learner1 = Learner2 = … = Learner m
- **Methods for changing training data:**
  - Bagging: Resample training data
  - Boosting: Reweight training data
  - DECORATE: Add additional artificial training data



Bagging

- Create ensembles by repeatedly randomly resampling the training data (Brieman, 1996).
- Given a training set of size *n*, create *m* sample sets
  - Each *bootstrap sample set* will on average contain 63.2% of the unique training examples, the rest are replicates.
- Combine the *m* resulting models using majority vote.

- Decreases error by decreasing the variance in the results due to *unstable learners*, algorithms (like decision trees) whose output can change dramatically when the training data is slightly changed.

Random Forests

- **Introduce two sources of randomness: “Bagging” and “Random input vectors”**
  - Each tree is grown using a bootstrap sample of training data
  - At each node, best split is chosen from random sample of *m* variables instead of all variables M.
- m is held constant during the forest growing
- Each tree is grown to the largest extent possible
- Bagging using decision trees is a special case of random forests when m=M

Random Forests

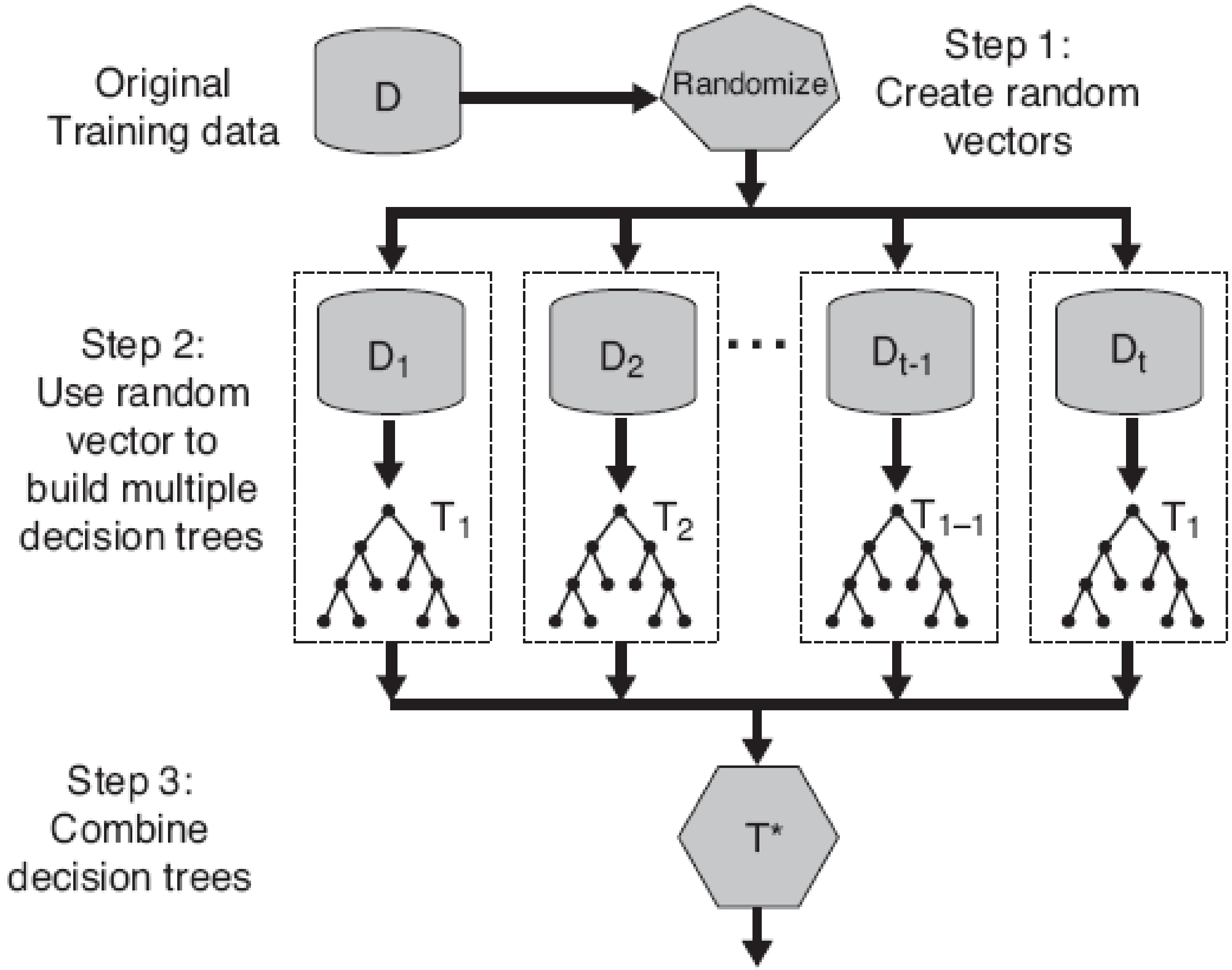


Figure 5.40. Random forests.

Random Forest Algorithm

- Good accuracy without over-fitting
- Fast algorithm (can be faster than growing/pruning a single tree); easily parallelized
- Handle high dimensional data without much problem

Boosting: AdaBoost

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line

learning and an application to boosting. *Journal of Computer and System Sciences,*

55(1):119–139, August 1997.

- Simple with theoretical foundation

Adaboost - Adaptive Boosting

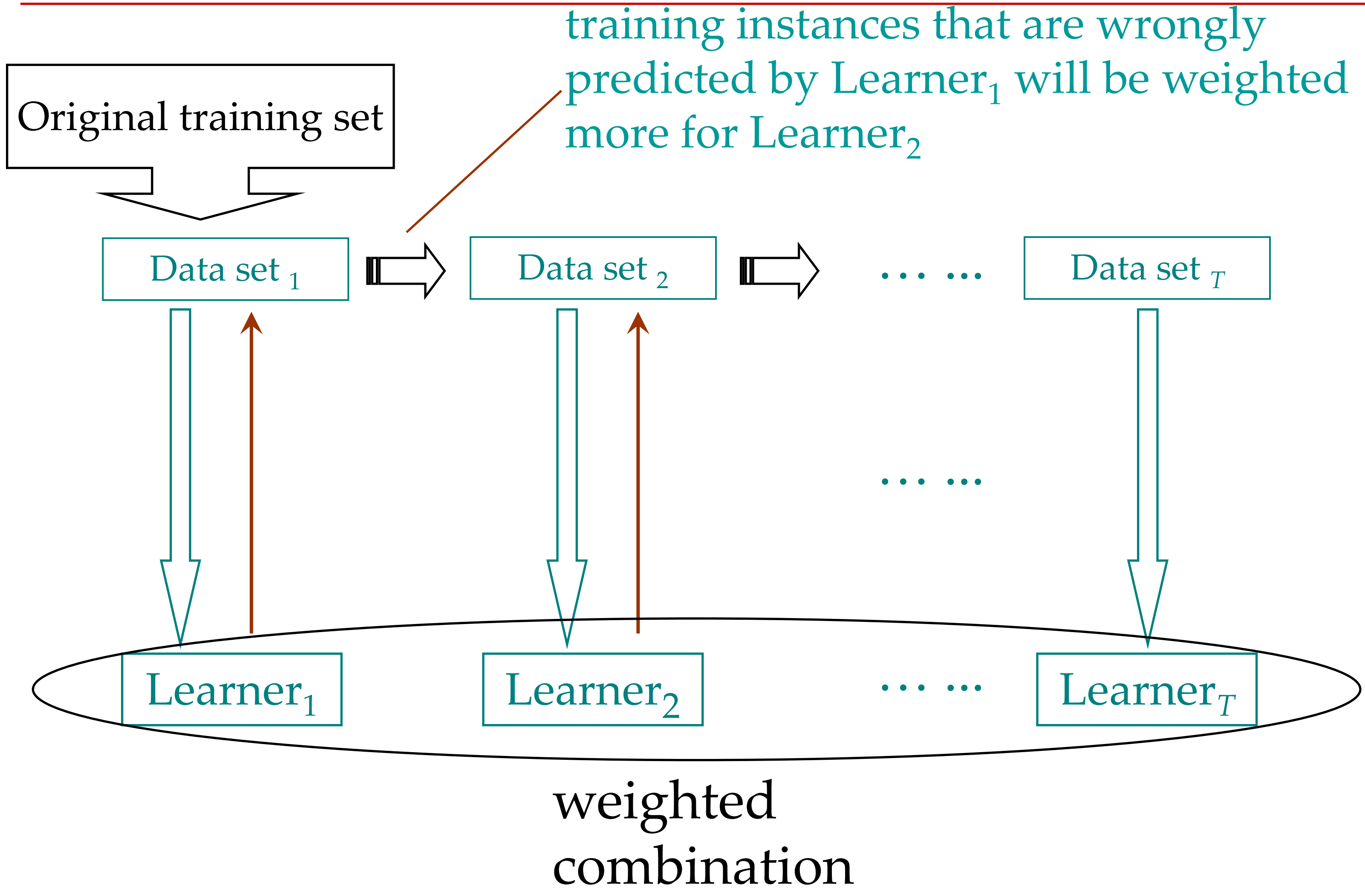
- **Use training set re-weighting**
  - Each training sample uses a weight to determine the probability of being selected for a training set.

- **AdaBoost is an algorithm for constructing a "strong" classifier as linear combination of "simple" "weak" classifier**

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

- **Final classification based on weighted sum of weak classifiers**

AdaBoost: An Easy Flow



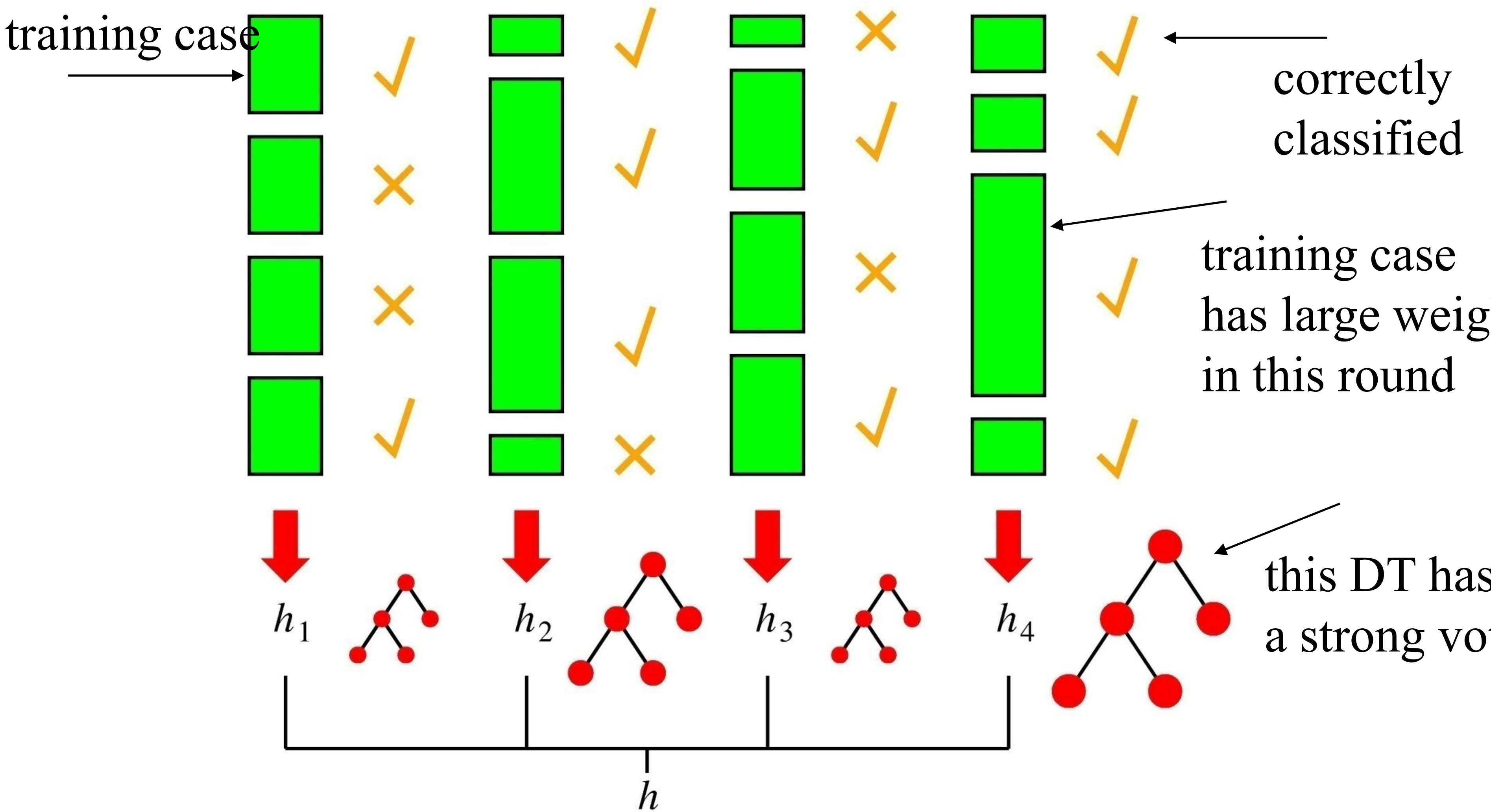
Adaboost Terminology

- $h_t(x)$… "weak" or basis classifier
- $H(x) = sign(f(x))$ … "strong" or final classifier

- Weak Classifier: < 50% error over any distribution

- Strong Classifier: thresholded linear combination of weak classifier outputs

And in a Picture



AdaBoost.M1

Each training sample has a weight, which determines the probability of being selected for training the component classifier

- **Given training set $X=\{(x_1,y_1), …$**
- **$y_i \in \{-1,+1\}$ correct label …**
- **Initialize distribution $D_1(i)=1/m$; (weight of training cases)**
- **for $t = 1,\ldots,T$:**
  - Find a weak classifier ("rule of th…

    $$h_t : X \rightarrow \{-1,+1\}$$

    with small error ε_t on D_t:
  - Update distribution D_t on $\{1,\ldots,m\}$ … α_t … ε_t …

    $$\ldots(i)\exp(-\alpha_t y_i h_t(x_i)) \over Z_t$$

$y_i * h_t(x_i) > 0$, if correct
$y_i * h_t(x_i) < 0$, if wrong

… case … last- … esis

Weighted voting for the final decision-making

- **ou…**

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

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

Reweighting

Effect on the training set

Reweighting formula:

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

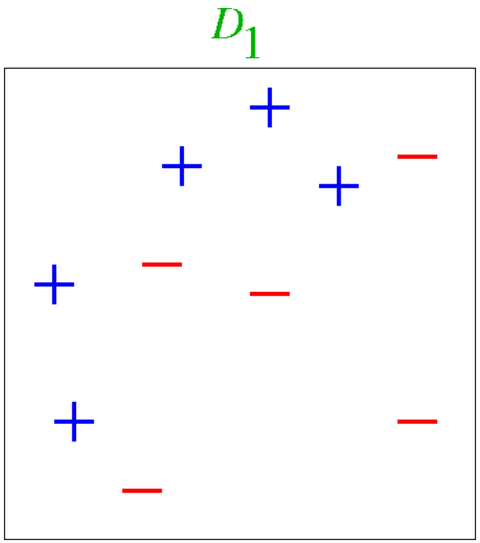
y * h(x) = 1

$$exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$$

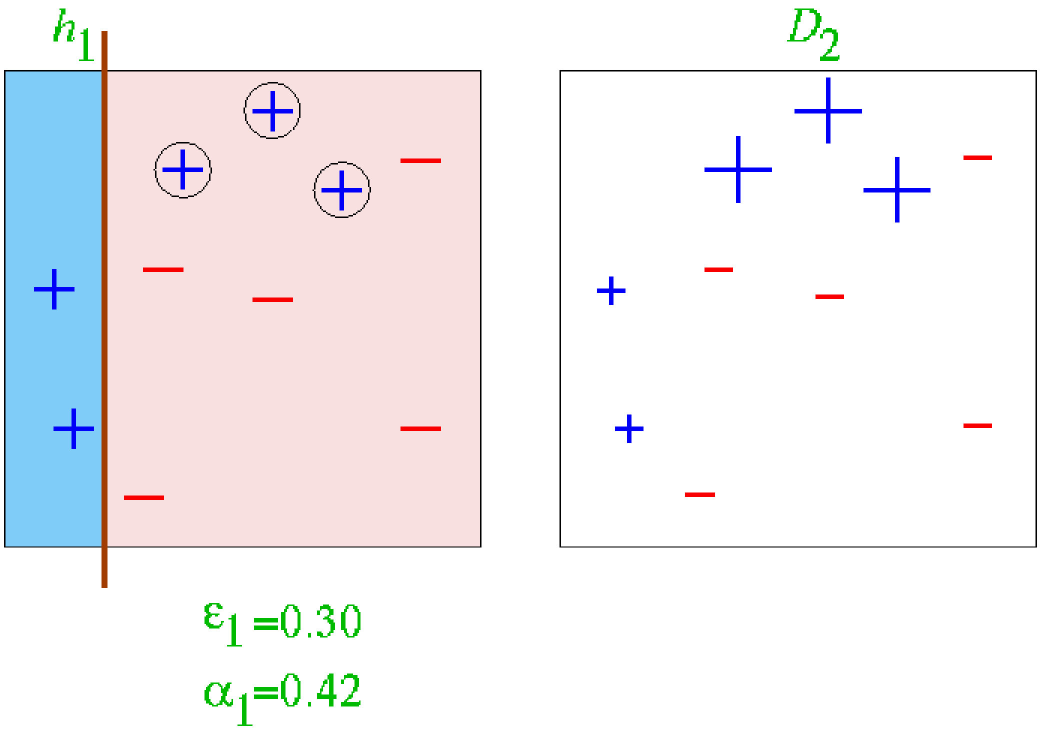
y * h(x) = -1

$\Rightarrow$ Increase (decrease) weight of wrongly (correctly) classified examples

Toy Example

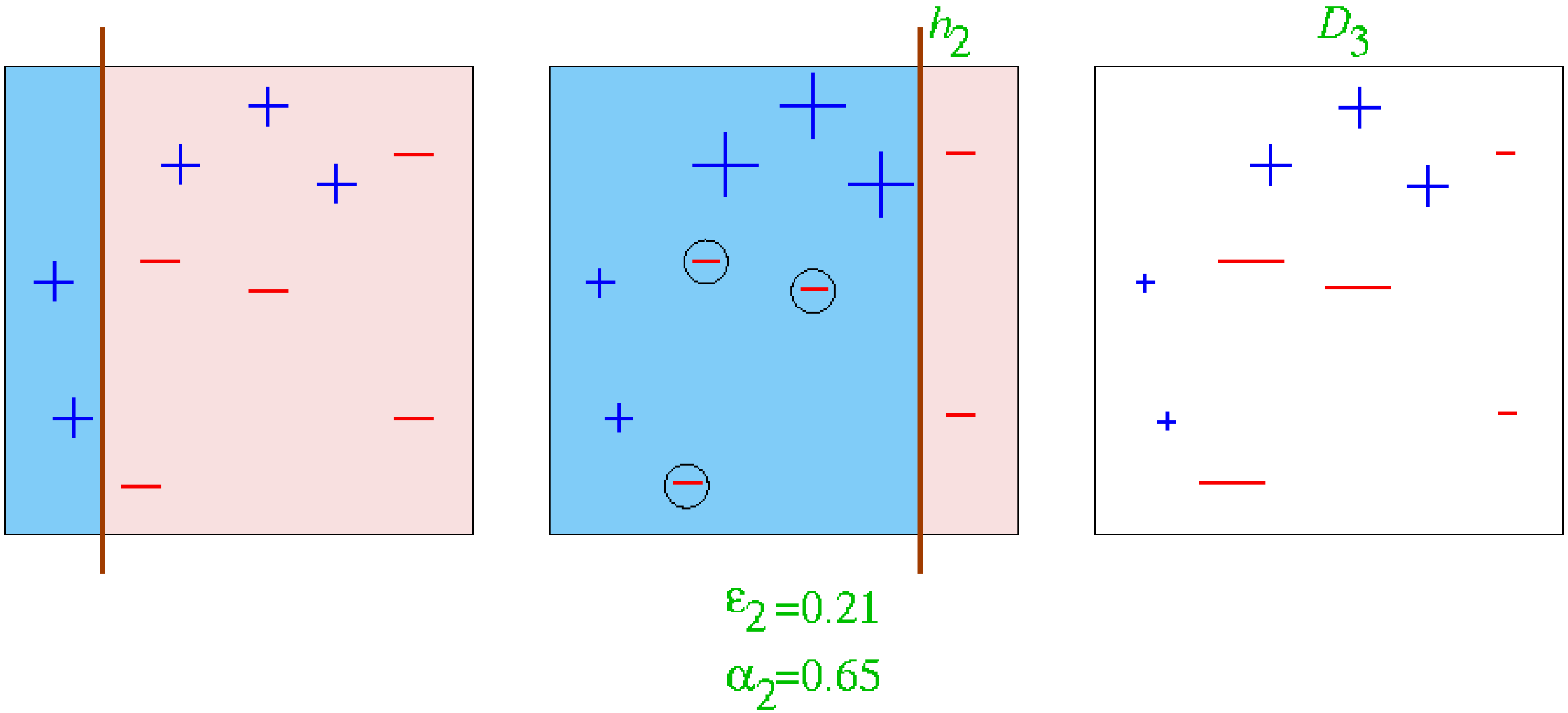


Round 1



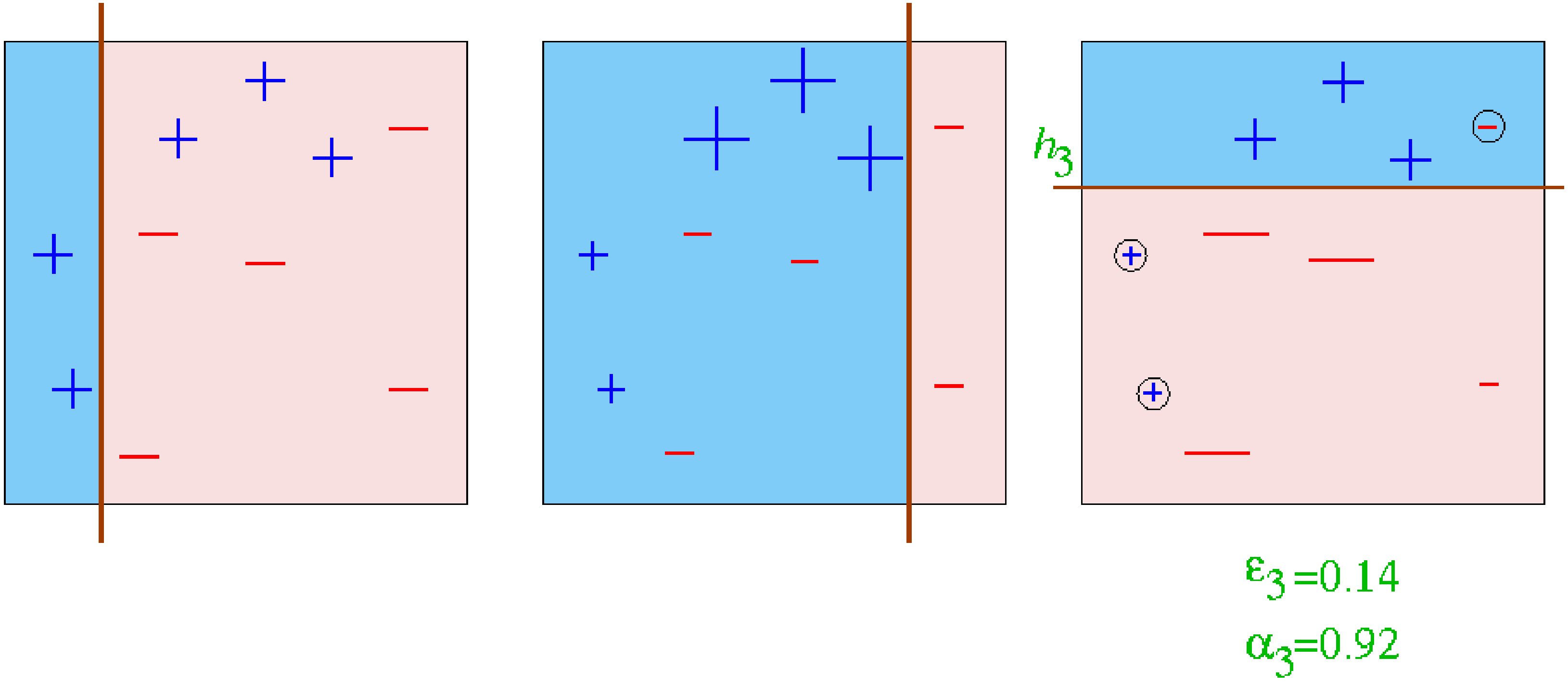
Weak classifier: if $h_1 < 0.2 \rightarrow 1$ else -1

Round 2



Weak classifier: if $h_2 < 0.8 \rightarrow$ 1 else -1

Round 3

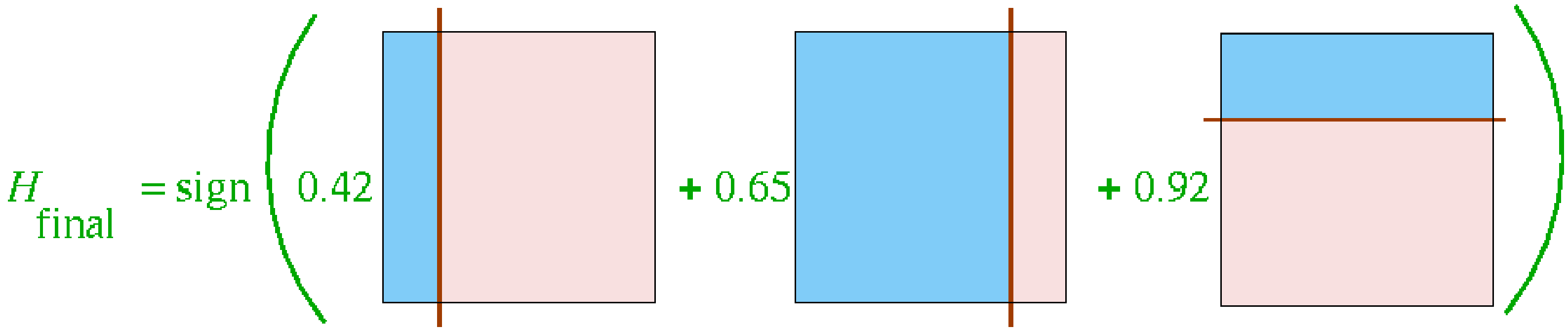


Weak classifier: if $h_3 > 0.7 \rightarrow$ 1 else -1

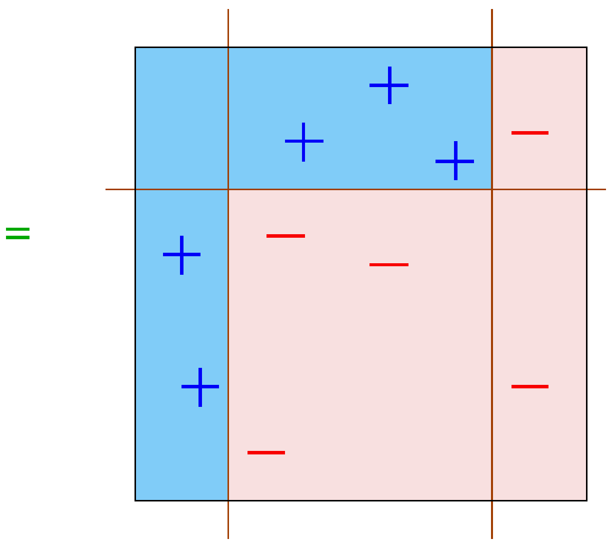
Final Combination

if $h_1 < 0.2 \rightarrow$ 1 else -1

if $h_2 < 0.8 \rightarrow$ 1 else -1



if $h_3 > 0.7 \rightarrow$ 1 else -1



Pros and cons of AdaBoost

Advantages

- Very simple to implement
- Does feature selection resulting in relatively simple classifier
- Fairly good generalization

Disadvantages

- Suboptimal solution
- Sensitive to noisy data and outliers

References

- **Duda, Hart, ect – *Pattern Classification***
- **Freund – "An adaptive version of the boost by majority algorithm"**
- **Freund – "Experiments with a new boosting algorithm"**
- **Freund, Schapire – "A decision-theoretic generalization of on-line learning and an application to boosting"**
- **Friedman, Hastie, etc – "Additive Logistic Regression: A Statistical View of Boosting"**
- **Jin, Liu, etc (CMU) – "A New Boosting Algorithm Using Input-Dependent Regularizer"**
- **Li, Zhang, etc – "Floatboost Learning for Classification"**
- **Opitz, Maclin – "Popular Ensemble Methods: An Empirical Study"**
- **Ratsch, Warmuth – "Efficient Margin Maximization with Boosting"**
- **Schapire, Freund, etc – "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods"**
- **Schapire, Singer – "Improved Boosting Algorithms Using Confidence-Weighted Predictions"**
- **Schapire – "The Boosting Approach to Machine Learning: An overview"**
- **Zhang, Li, etc – "Multi-view Face Detection with Floatboost"**

AdaBoost: Training Error Analysis

- Suppose $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$ $\qquad H(x) = \mathrm{sign}(f(x))$

Equivalent

if $H(x_i) \neq y_i$ then $y_i f(x_i) \leq 0$ implying that $\exp(-y_i f(x_i)) \geq 1$. Thus,

$$[\![H(x_i) \neq y_i]\!] \leq \exp(-y_i f(x_i)).$$

- Therefore, training error is:

$$\frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right| \leq \frac{1}{m}\sum$$

- As:

$$D_{T+1}(i) = \frac{\exp\left(-\sum_t \alpha_t y_i \cdots\right.}{m \prod_t Z_t}$$

Considering

{i: $H(x_i) \neq y_i$} is a vector which i-th element is [$H(x_i) \neq y_i$].
|{i: $H(xi) \neq yi$}| is the sum of all the element in the vector

Finally: $\sum_i D_{T+1}(i) = 1$, $\frac{1}{m}\sum_i \exp(-y_i f(x_i)) = \prod_t Z_t$

$$\frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right| \leq \prod_{t=1}^{T} Z_t$$

AdaBoost: How to choose α_t

- According to $\frac{1}{m}\left|\{i : H(x_i) \neq y_i\}\right| \leq \prod_{t=1}^{T} z_t$

This equation is obvious if we treat u_i as a binary-valued variable.

Minimize the error bound could be done

- Let $u_i = y_i h_t($

$$Z = \sum_i D(i)e^{-}$$

By settir
consider
easily ge

Actually AdaBoost can just minimize the training error.

$$\begin{cases} 1 = \sum_{h=y} D_t(i) + \sum_{h \neq y} D_t(i) \\ r = \sum_{h=y} D_t(i) - \sum_{h \neq y} D_t(i) \end{cases}, \text{ let } \varepsilon = \sum_{h \neq y} D_t(i), \text{ we have } \varepsilon = \frac{1-r}{2}$$

- The right term is minimized

$$\alpha = \frac{1}{2}\ln\left(\frac{1+r}{1-r}\right) \qquad \sum_i D(i)u_i$$

Then the training error of H is at most $\prod_{t=1}^{T} \sqrt{1 - r_t^2}$.