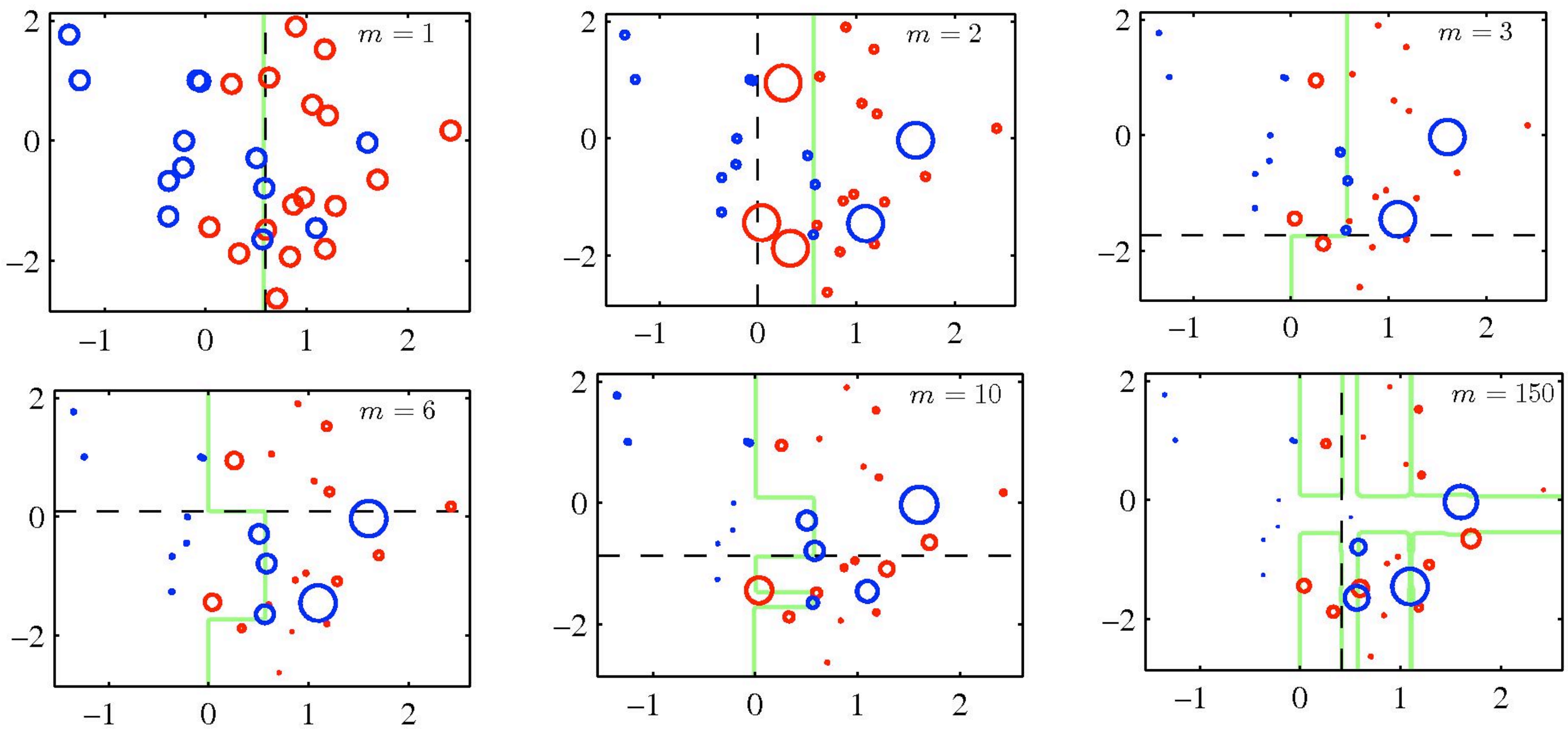
Boosting

- Main idea:
  - train classifiers (e.g. decision trees) in a sequence.
  - a new classifier should focus on those cases which were incorrectly classified in the last round.
  - combine the classifiers by letting them vote on the final prediction (like bagging).
  - each classifier could be (should be) very “weak”, e.g. a decision stump.

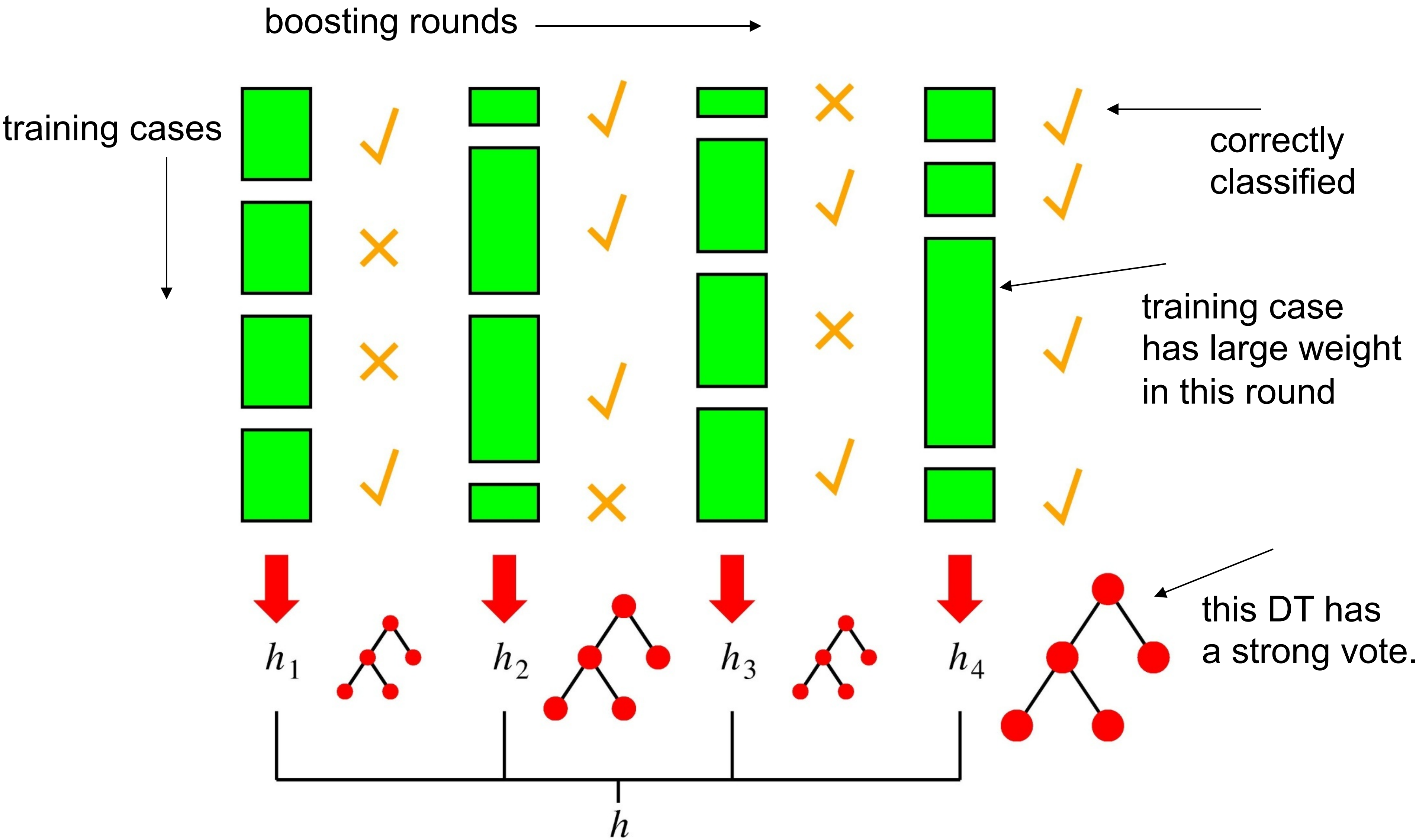
Boosting Intuition

- We adaptively weigh each data case.
- Data cases which are wrongly classified get high weight (the algorithm will focus on them)
- Each boosting round learns a new (simple) classifier on the weighed dataset.
- These classifiers are weighed to combine them into a single powerful classifier.
- Classifiers that obtain low training error rate have high weight.
- We stop by using monitoring a hold out set (cross-validation).

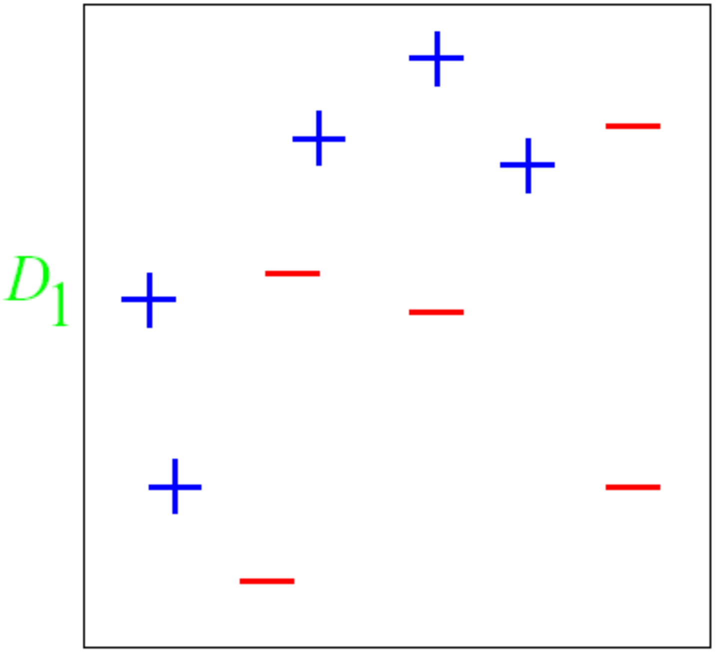
Example



Boosting in a Picture



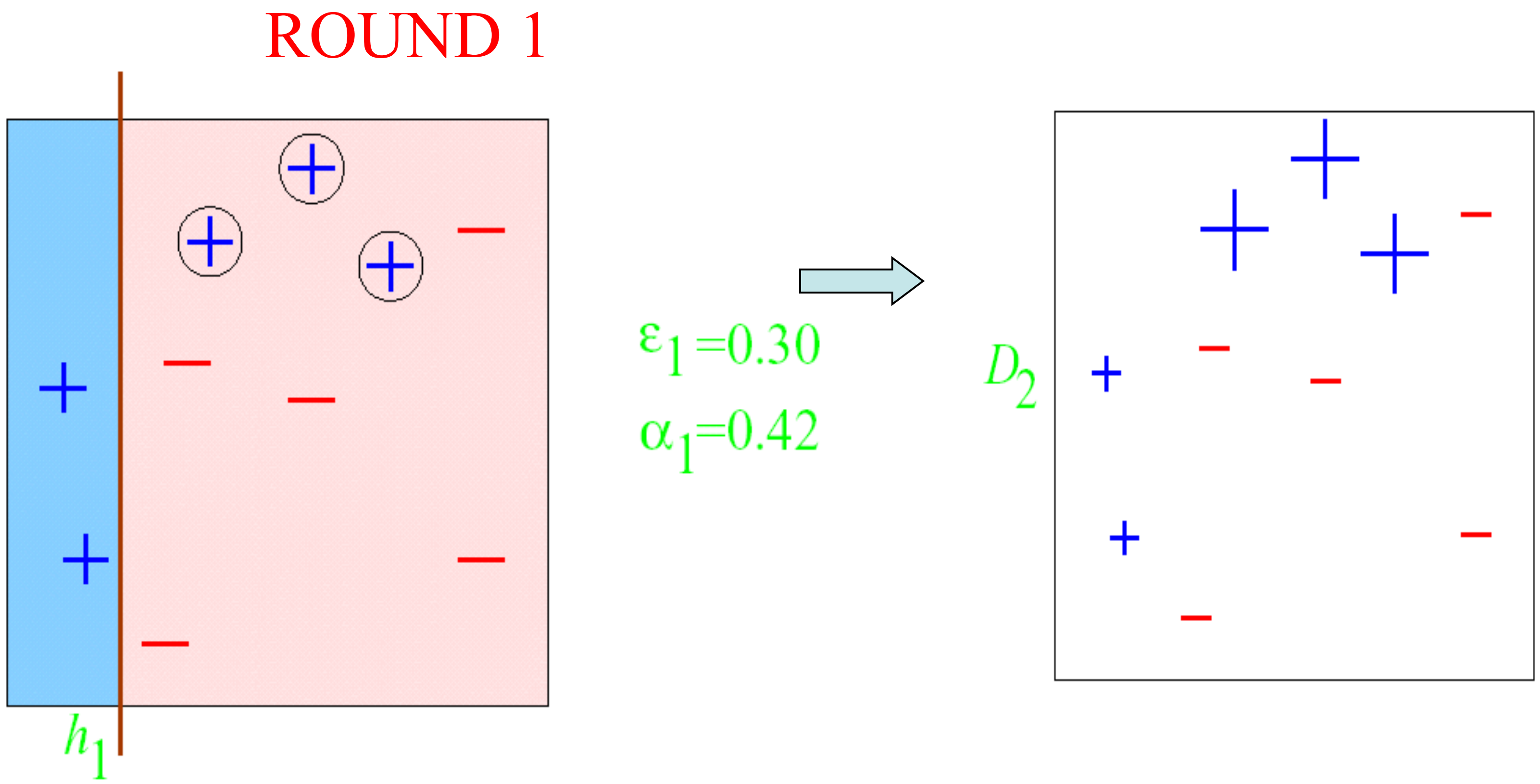
And in animation



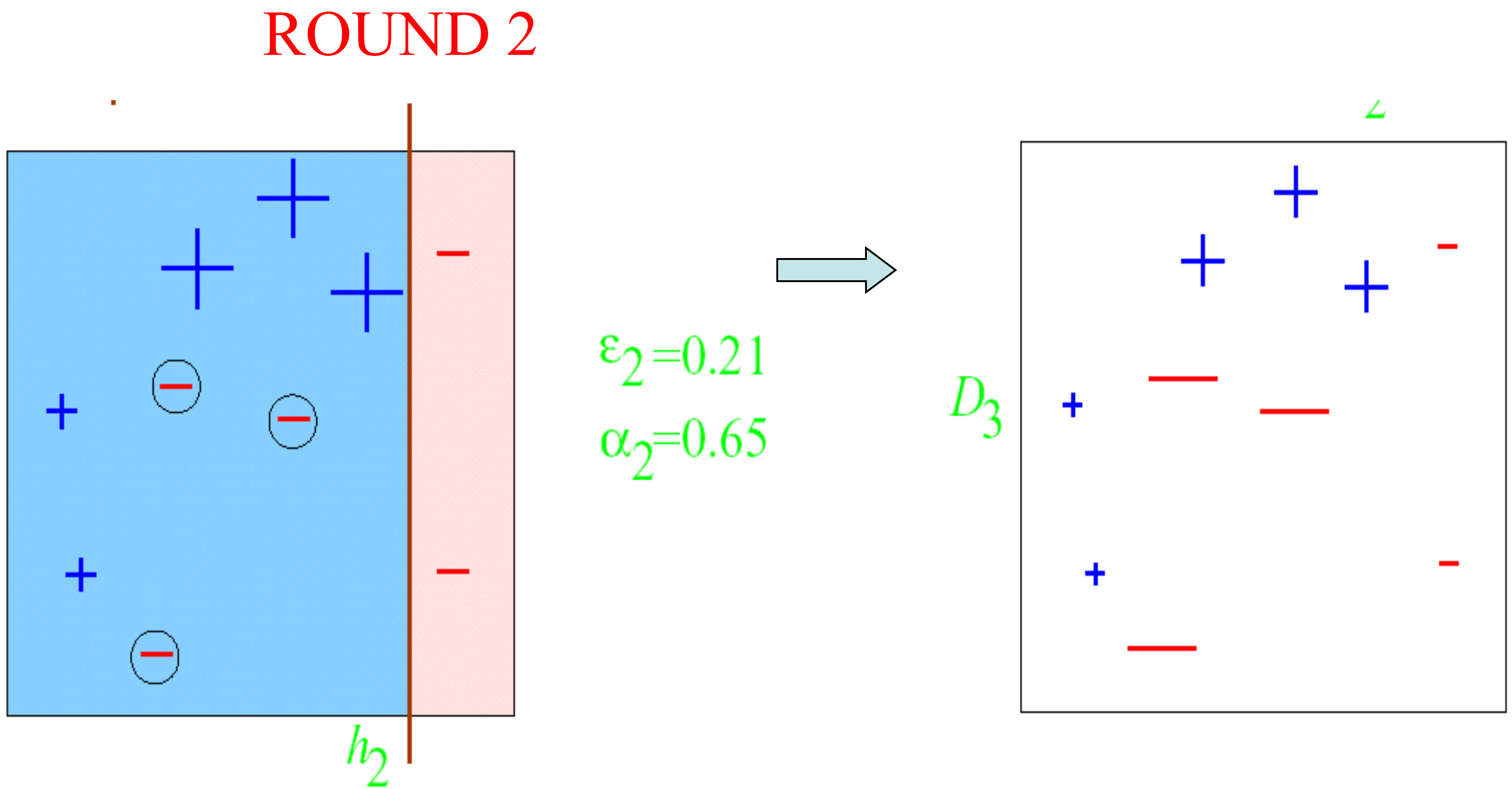
Original Training set : Equal Weights to all training samples

Taken from "**A Tutorial on Boosting**" **by** Yoav Freund and Rob Schapire

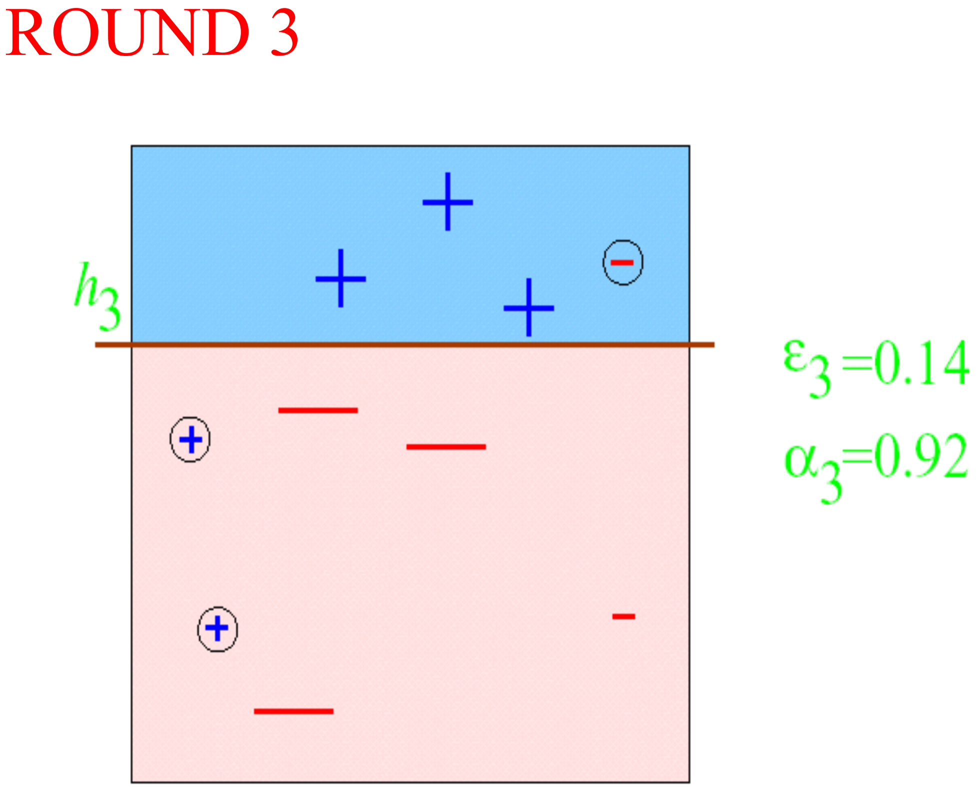
AdaBoost(Example)



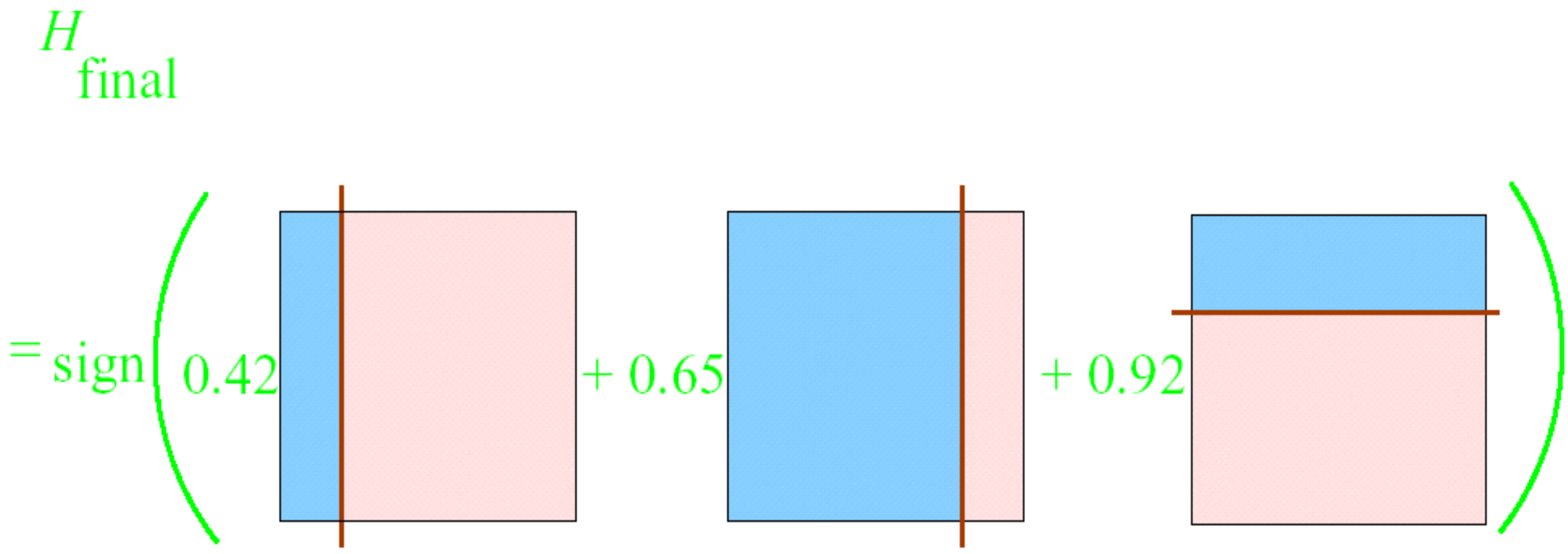
AdaBoost(Example)



AdaBoost(Example)



AdaBoost(Example)



AdaBoost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X,\ y_i \in Y = \{-1, +1\}$

Initialise $D_1(i) = \frac{1}{m}$.

For $t = 1, \ldots, T$:

- Find the classifier $h_t : X \to \{-1, +1\}$ that minimizes the error with respect to the distribution D_t:

$$h_t = \arg\min_{h_j \in \mathcal{H}} \epsilon_j = \sum_{i=1}^{m} D_t(i)[y(i) \neq h_j(x_i)]$$

- Prerequisite: $\varepsilon_t < 0.5$, otherwise stop.

- Choose $\alpha_t \in \mathbf{R}$, typically $\alpha_t = \frac{1}{2}\ln\frac{1-\epsilon_t}{\epsilon_t}$ where ε_t is the weighted error rate of classifier h_t.
- Update:

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

where Z_t is a normalisation factor (chosen so that D_{t+1} will be a distribution).
Output the final classifier:

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

The equation to update the distribution D_t is constructed so that:

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

Thus, after selecting an optimal classifier h_t for the distribution D_t, the examples x_i that the classifier h_t identified correctly are weighted less and those that it identified incorrectly are weighted more.

Therefore, when the algorithm is testing the classifiers on the distribution D_{t+1}, it will select a classifier that better identifies those examples that the previous classifer missed.