Boosting Methods

Benk Erika
Kelemen Zsolt

Summary

- Overview
- Boosting – approach, definition, characteristics
- Early Boosting Algorithms
- AdaBoost – introduction, definition, main idea, the algorithm
- AdaBoost – analysis, training error
- Discrete AdaBoost
- AdaBoost – pros and contras
- Boosting Example

Overview

- Introduced in 1990s
- originally designed for classification problems
- extended to regression
- motivation - a procedure that combines the outputs of many “weak” classifiers to produce a powerful “committee”

To add:

- What is a classification problem, (slide)
- What is a weak learner, (slide)
- What is a committee, (slide)
- …… Later ……
- How it is extended to classification…

Boosting Approach

- select small subset of examples
- derive rough rule of thumb
- examine 2nd set of examples
- derive 2nd rule of thumb
- repeat T times
- **questions**:
  - how to choose subsets of examples to examine on each round?
  - how to combine all the rules of thumb into single prediction rule?
- **boosting** = general method of converting rough rules of thumb into highly accurate prediction rule

Ide egy kesobbi slide-ot… …… peldanak

Boosting - definition

- A machine learning algorithm
- Perform supervised learning
- Increments improvement of learned function
- Forces the weak learner to generate new hypotheses that make less mistakes on “harder” parts.

Boosting - characteristics

- iterative
- successive classifiers depends upon its predecessors
- look at errors from previous classifier step to decide how to focus on next iteration over data

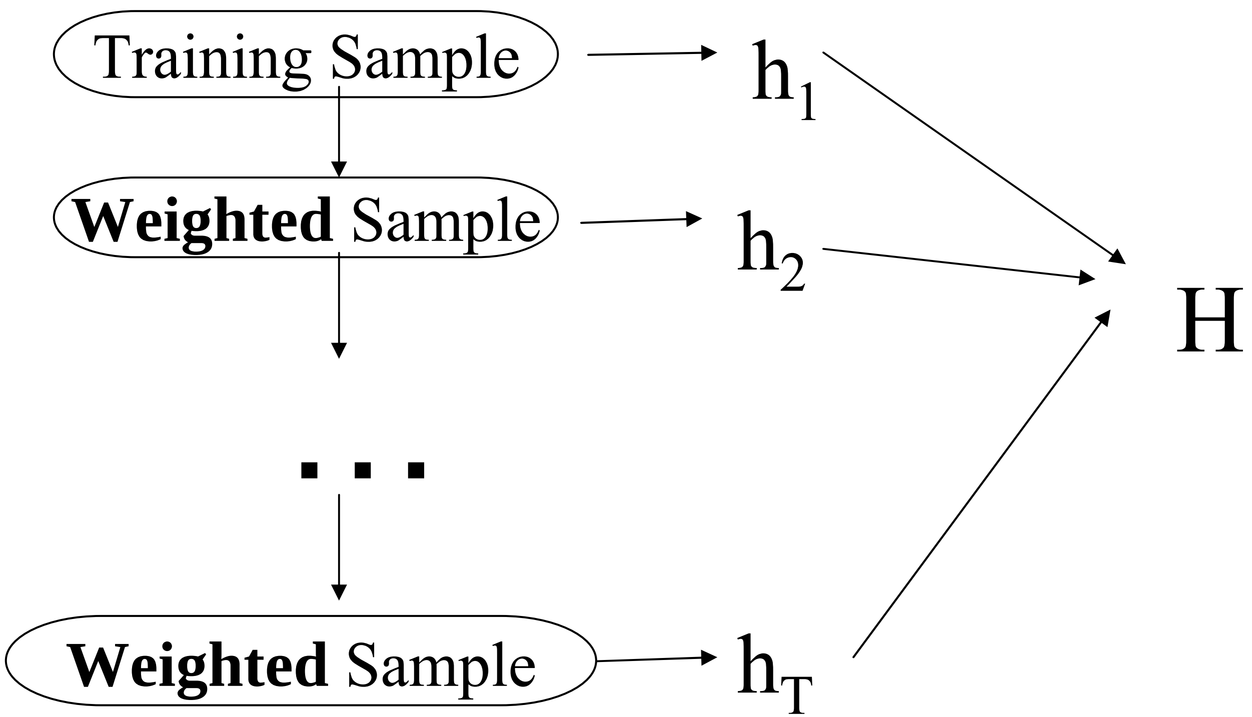
Early Boosting Algorithms

- Schapire (1989):
  - first provable boosting algorithm
  - call weak learner three times on three modified distributions
  - get slight boost in accuracy
  - apply recursively

Early Boosting Algorithms

- Freund (1990)
  - “optimal” algorithm that “boosts by majority”
- Drucker, Schapire & Simard (1992):
  - first experiments using boosting
  - limited by practical drawbacks
- Freund & Schapire (1995) – **AdaBoost**
  - strong practical advantages over previous boosting algorithms

Boosting



Boosting

- Train a set of weak hypotheses: h_1, …., h_T.
- The combined hypothesis H is a **weighted** majority vote of the T weak hypotheses.
  - ➔ Each hypothesis h_t has a weight α_t.

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

- During the training, focus on the examples that are misclassified.
  - ➔ At round t, example x_i has the weight $D_t(i)$.

Boosting

- Binary classification problem
- Training data:

$$(x_1, y_1), \ldots., (x_m, y_m), \textit{where} \quad x_i \in X, y_i \in Y = \{-1,1\}$$

- $D_t(i)$: the weight of x_i at round t. $D_1(i)=1/m$.
- A learner L that finds a weak hypothesis h_t: X ➔ Y given the training set and D_t
- The error of a weak hypothesis h_t:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} D_t(i)$$

AdaBoost - Introduction

- Linear classifier with all its desirable properties
- Has good generalization properties
- Is a feature selector with a principled strategy (minimisation of upper bound on empirical error)
- Close to sequential decision making

AdaBoost - Definition

- Is an algorithm for constructing a "strong" classifier as linear combination

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

of simple "weak" classifiers $h_t(x)$.

- $h_t(x)$ - "weak" or basis classifier, hypothesis, "feature"
- $H(x) = sign(f(x))$ – "strong" or final classifier/hypothesis

The AdaBoost Algorithm

- Input – a training set: $S = \{(x_1, y_1); \ldots ;(x_m, y_m)\}$
  - $x_i \varepsilon$ X, X instance space
  - $y_i \varepsilon$ Y, Y finite label space
    - in binary case Y = {-1,+1}
- Each round, t=1,…,T, AdaBoost calls a given *weak or base learning algorithm* – accepts as input a sequence of training examples (S) and a set of weights over the training example ($Dt(i)$)

The AdaBoost Algorithm

- The weak learner computes a weak classifier (h_t), : $h_t : X \rightarrow R$
- Once the weak classifier has been received, AdaBoost chooses a parameter ($\alpha_t \varepsilon R$) – intuitively measures the importance that it assigns to h_t.

The main idea of AdaBoost

- to use the weak learner to form a highly accurate prediction rule by calling the weak learner repeatedly on different distributions over the training examples.
- initially, all weights are set equally, but each round the weights of incorrectly classified examples are increased so that those observations that the previously classifier poorly predicts receive greater weight on the next iteration.

The Algorithm

- **Given** $(x_1, y_1),\ldots, (x_m, y_m)$ where $x_i \epsilon X$, $y_i \epsilon \{-1, +1\}$
- **Initialise** weights $D_1(i) = 1/m$
- **Iterate** $t=1,\ldots,T$:
  - Train weak learner using distribution ***Dt***
  - Get weak classifier: $h_t : X \rightarrow R$
  - Choose $\alpha_t \varepsilon R$
  - Update: $D_{t+1}(i) = \dfrac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
    - where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution), and α_t:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) > 0$$

- **Output** – the final classifier

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

AdaBoost - Analysis

- the weights *Dt*(*i*) are updated and normalised on each round. The normalisation factor takes the form

$$Z_t = \sum_{i=1}^{m} D_t(i) \mathrm{e}^{-\alpha_t y_i h_t(x_i)}$$

and it can be verified that *Zt* measures exactly the ratio of the new to the old value of the exponential sum

$$\sum_{i=1}^{m} \exp\left(- y_i \sum_{j=1}^{t} \alpha_j h_j(x_i)\right)$$

on each round, so that $\Pi_t Z_t$ is the final value of this sum. We will see below that this product plays a fundamental role in the analysis of AdaBoost.

AdaBoost – Training Error

- Theorem:
  - run Adaboost
  - let $\varepsilon_t=1/2-\gamma_t$
  - then the training error:

$$H_{final} \leq \prod_t 2\sqrt{\varepsilon_t(1-\varepsilon_t)} = \prod_t \sqrt{1-4\gamma_t^2} \leq \exp(-2\sum_t \gamma_t^2)$$

$$\forall t : \gamma_t \geq \gamma > 0 \Rightarrow H_{final} \leq e^{-2\gamma^2 T}$$

Choosing parameters for Discrete AdaBoost

- In Freund and Schapire's original Discrete AdaBoost the algorithm each round selects the weak classifier, h_t, that minimizes the weighted error on the training set

$$\epsilon_t = \sum_i D_t(i)[\![h_t(x_i) \neq y_i]\!] = \sum_i D_t(i) \left(\frac{1 - y_i h_t(x_i)}{2} \right)$$

- Minimizing Z_t, we can rewrite:
- $$\begin{aligned} Z_t &= \sum_i D_t(i) \mathrm{e}^{-\alpha_t y_i h_t(x_i)} \\ &= \sum_i D_t(i) \left(\frac{1 + y_i h_t(x_i)}{2} \mathrm{e}^{-\alpha_t} + \frac{1 - y_i h_t(x_i)}{2} \mathrm{e}^{\alpha_t} \right) \end{aligned}$$

Choosing parameters for Discrete AdaBoost

- analytically we can choose α_t by minimizing the first (ε_t=…) expression:

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

- Plugging this into the second equation (Z_t), we can obtain:

$$Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}.$$

Discrete AdaBoost - Algorithm

- **Given** $(x_1, y_1),\ldots, (x_m, y_m)$ where $x_i \epsilon X$, $y_i \epsilon \{-1, +1\}$
- **Initialise** weights $D_1(i) = 1/m$
- **Iterate** $t=1,\ldots,T$:
  - Find $h_t = \arg\min_{h_j} \epsilon_j$ where $\epsilon_j = \sum_{i=1}^{m} D_t(i)[\![h_t(x_i) \neq y_i]\!]$
  - Set

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

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

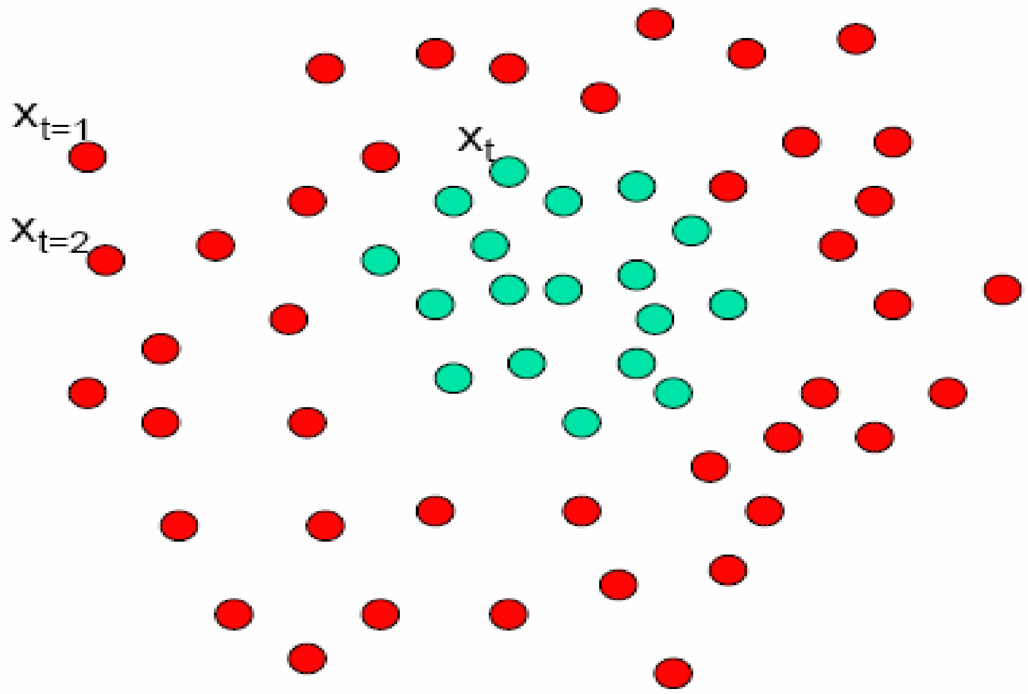
- **Output** – the final classifier

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

AdaBoost – Pros and Contras

- Pros:
  - Very simple to implement
  - Fairly good generalization
  - The prior error need not be known ahead of time
- Contras:
  - Suboptimal solution
  - Can over fit in presence of noise

Boosting - Example



Each data point has

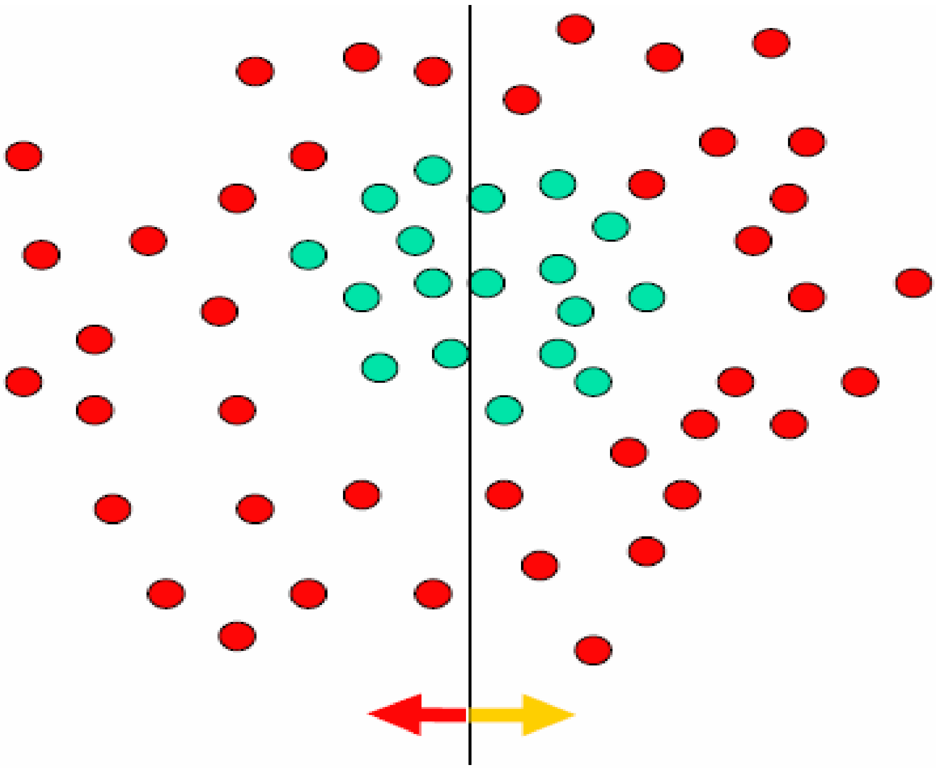
a class label:

$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{green}) \end{cases}$$

and a weight:

$w_t = 1$

Boosting - Example



Each data point has a class label:

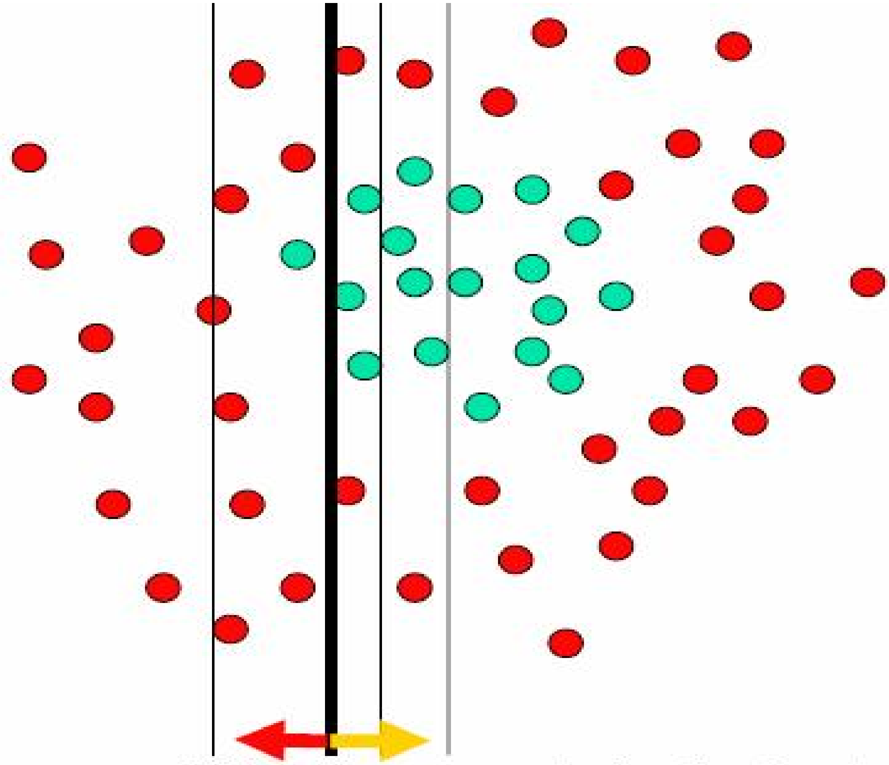
$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{green}) \end{cases}$$

and a weight:

$w_t = 1$

h => p(error) = 0.5 it is at chance

Boosting - Example



This one seems to be the best

Each data point has

a class label:

$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{green}) \end{cases}$$

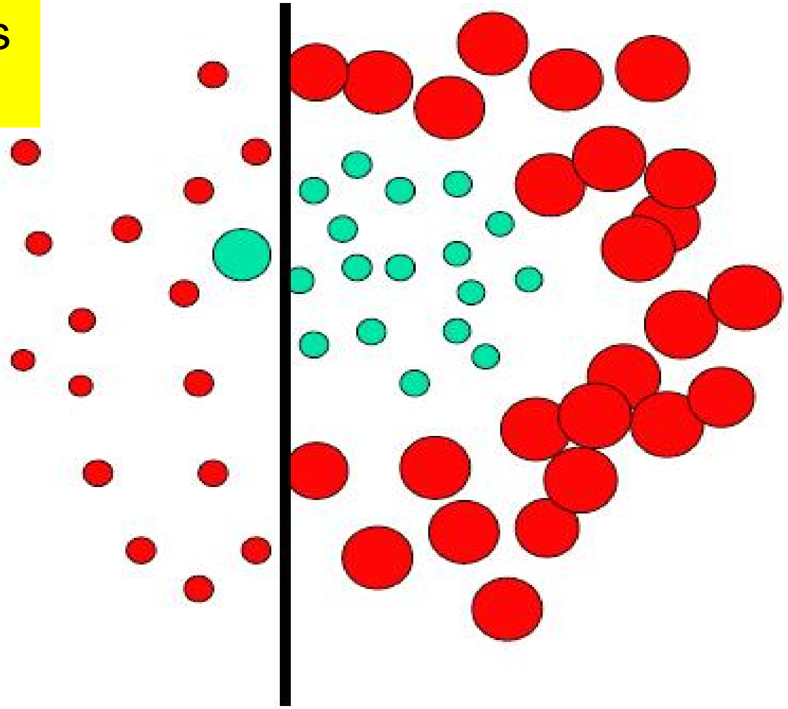
and a weight:

$w_t = 1$

This is a ‘**weak classifier**’: It performs slightly better than chance.

Boosting - Example

Ezt kellene korabban is mutatni …… peldanak



Each data point has a class label:

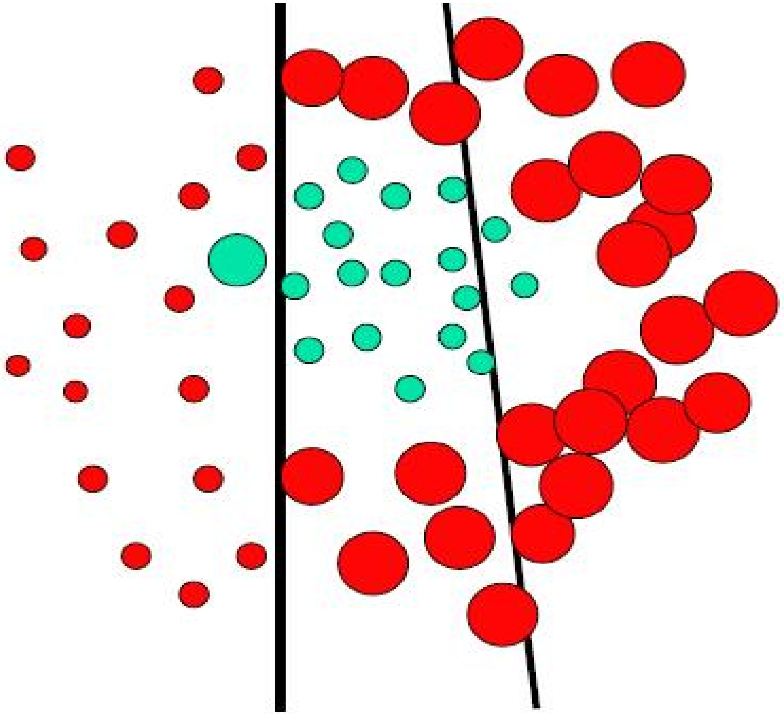
$$y_t = \begin{cases} +1 & (\bullet) \\ -1 & (\bullet) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



Each data point has a class label:

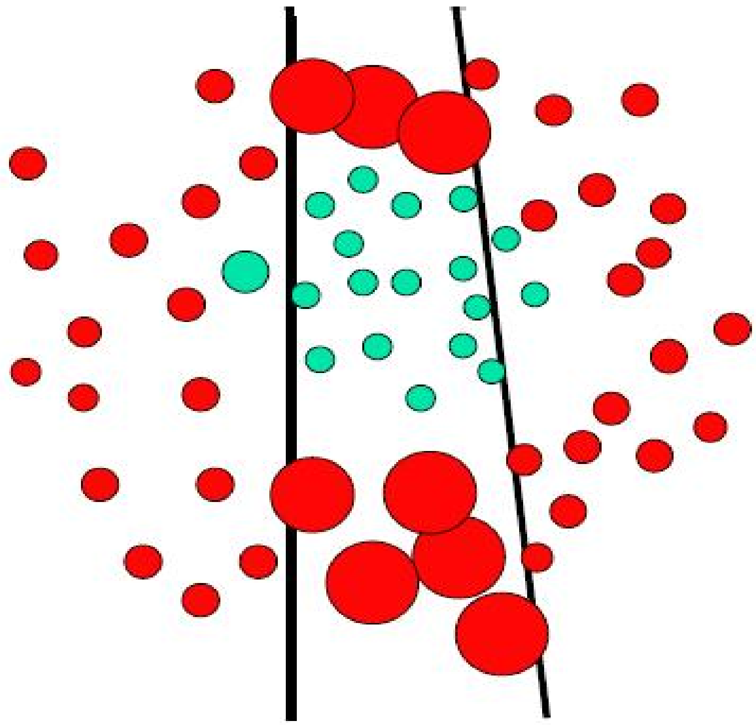
$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



Each data point has

a class label:

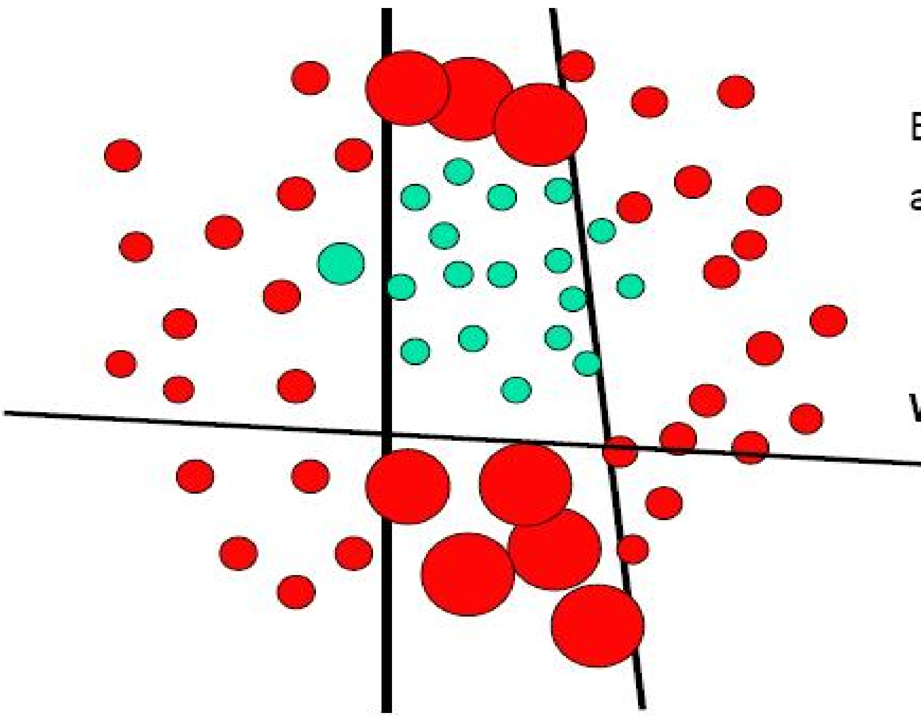
$$y_t = \begin{cases} +1 \; () \\ -1 \; () \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



Each data point has

a class label:

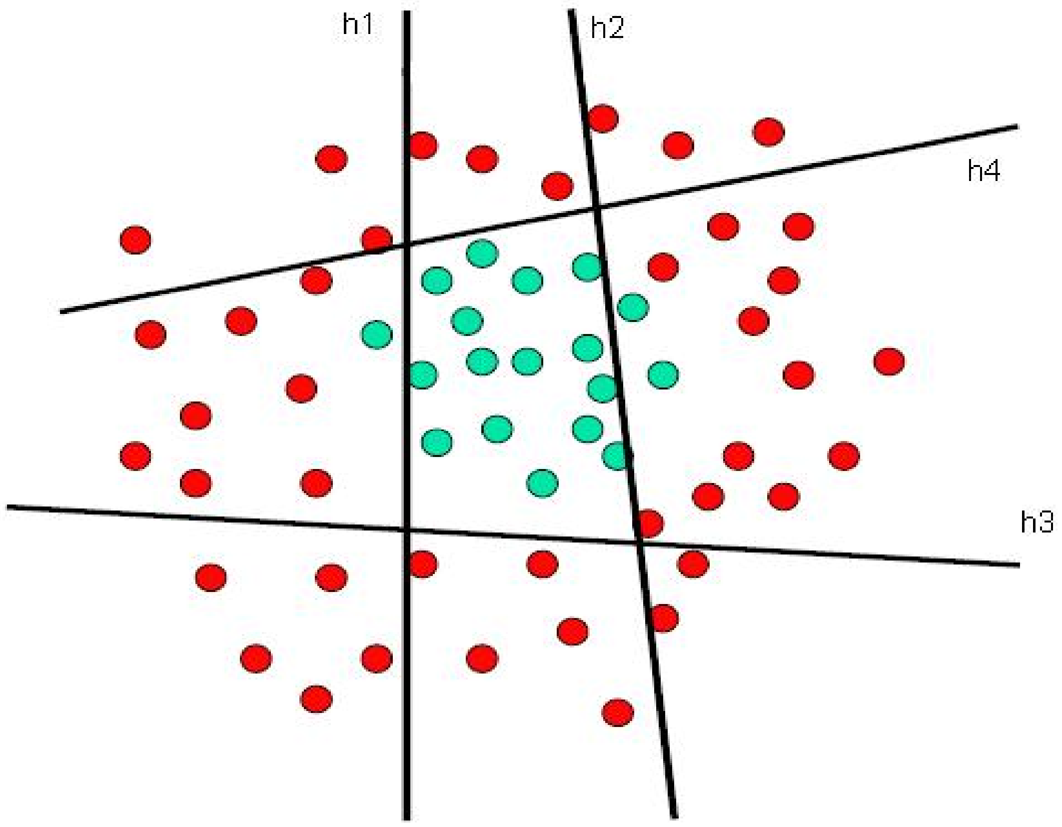
$$y_t = \begin{cases} +1 & (\ ) \\ -1 & (\ ) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Bibliography

- Friedman, Hastie & Tibshirani: The Elements of Statistical Learning (Ch. 10), 2001
- Y. Freund: Boosting a weak learning algorithm by majority. In *Proceedings of the Workshop on Computational Learning Theory*, 1990.
- Y. Freund and R.E. Schapire: A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory*, 1995.

Bibliography

- J. Friedman, T. Hastie, and R. Tibshirani: Additive logistic regression: a statistical view of boosting. *Technical Report, Dept. of Statistics, Stanford University*, 1998.
- Thomas G. Dietterich: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 139–158, 2000.