Ensemble methods

Wang Houfeng
ICL,Peking University

Outline

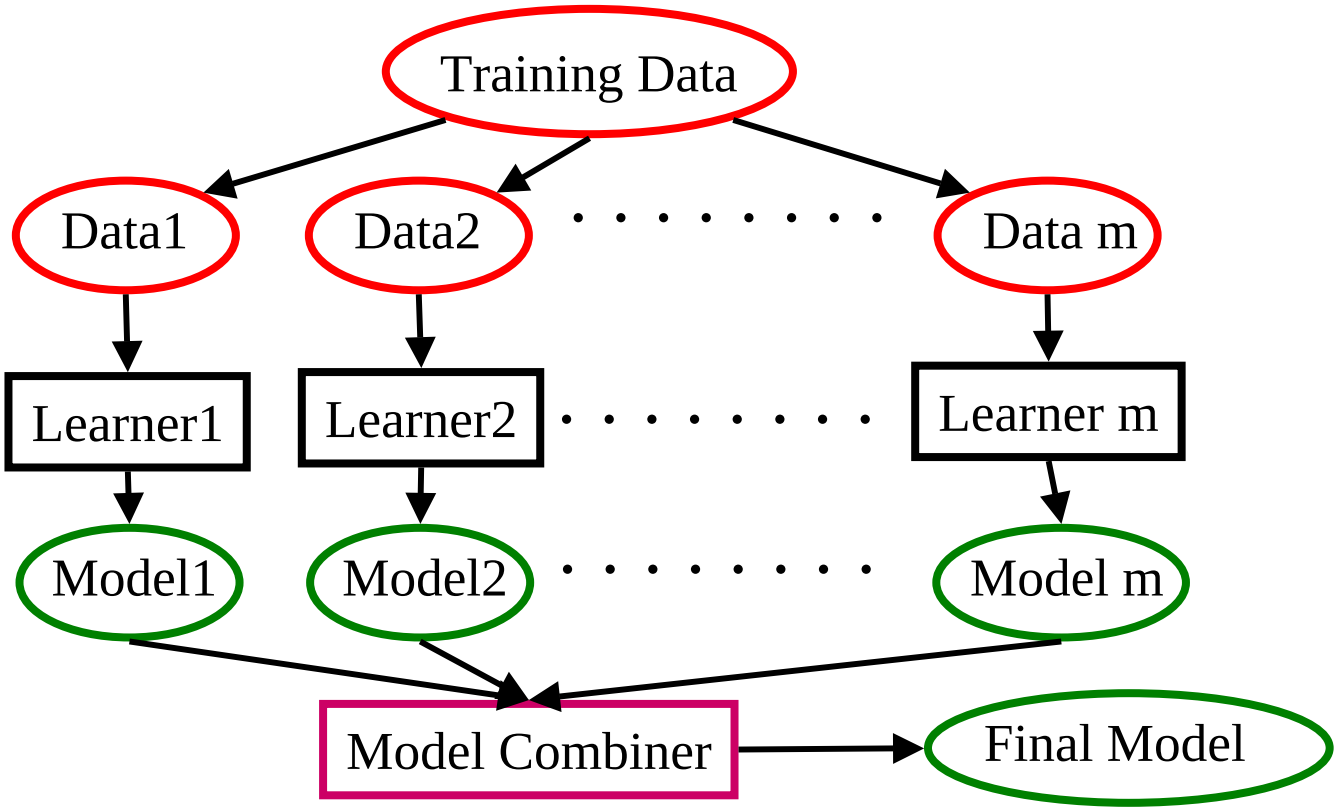
- **Overview**
- Bagging
- Boosting & Adaboost
- Summary

Basics

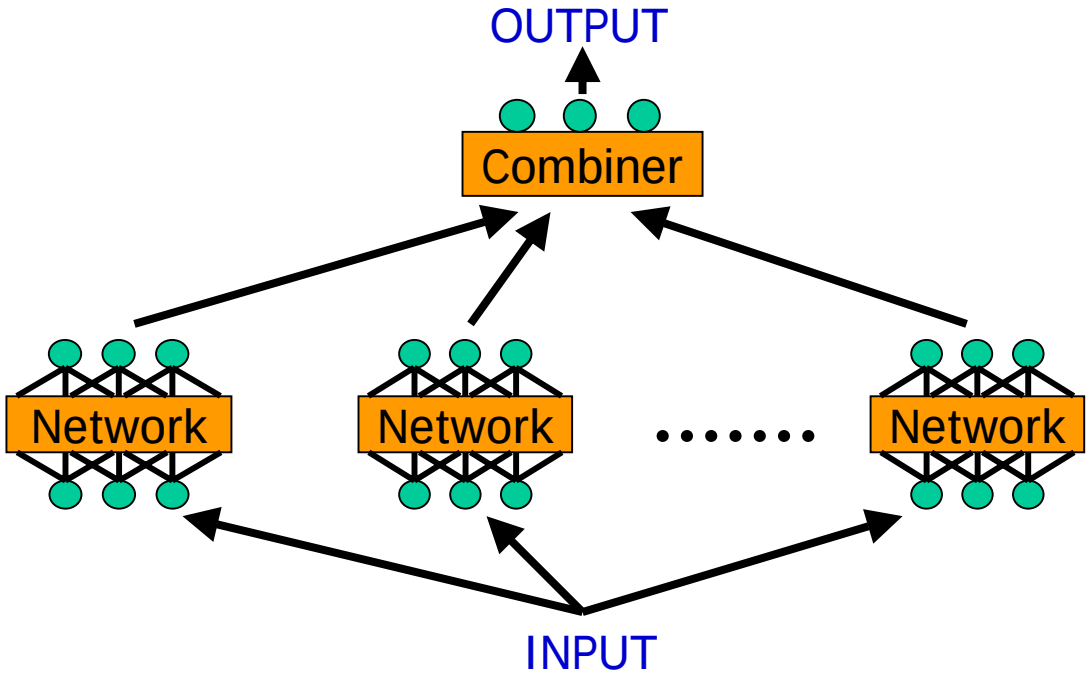
- What is a weak classifier?
  - One not guaranteed to do better than random guessing (1 / *number of classes*)
- What is an instable classifier?
  - Small change to training set causes large change in output hypothesis.
  - True for decision trees, neural networks;
- Goal:
  - combine multiple weak and instable classifiers, get one *at least as accurate as strongest.*

Learning Ensembles

- Learn multiple different learning algorithms.
- Combine decisions of multiple definitions, e.g. using weighted voting.



Ensembles of Neural Networks



- Ensembles often produce accuracy gains of 5-10 percentage points!
- Can combine "classifiers" of various types
  - E.g., **decision trees**, **neural networks**, etc.

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
- Different methods for changing training data:
  - **Bagging:** Resample training data
  - **Boosting:** Reweight training data

Combining Multiple Models

Three ideas

1. Simple (unweighted) votes(e.g., **majority vote**)
2. Weighted votes
3. Train a combining function

Outline

- Overview
- **Bagging**
- Boosting & Adaboost
- Ensemble

Bagging

- Introduced by Breiman (1996)
- “Bagging” stands for “**b**ootstrap **agg**regat**ing**”.
- Bagging predictors is a method for generating multiple versions of a predictor and using these to get an **aggregated predictor**.
- The aggregation **averages** over the versions when predicting a numerical outcome and does a **majority vote** when predicting a class;
- The multiple versions are formed by making **bootstrap replicates** of the training set and using these as new learning sets.
- The vital element is the **instability of the prediction method**. Bagging can improve unstable estimation or classification.

Bootstrap Sampling

- Given: set $D=\{(x_i, y_i) \mid x_i \text{ in } X, y_i \text{ in } Y\}$ containing n training examples
- Create $S[i]$ m times by drawing n examples at random *with replacement* from D
- $S[i]$ of size n: expected to leave out 0.37 of examples from D

Bootstrap Sampling

- The bootstrap is an estimation method that uses sampling with replacement to form the training set;
- **Training set**: a dataset of *n* examples is sampled with replacement *n* times with replacement to form the training set of *n* examples (possibly with repetitions);
- **Test set**: the examples from the original dataset that don't occur in the training set.

0.632-bootstrap

- A particular example has a probability of (1-1/n) of not being selected for the training set. Thus, an instance will fall in the test set with probability:

$$\lim_{x \to \infty} (1 - \frac{1}{n})^n = \frac{1}{e} = 0.368$$

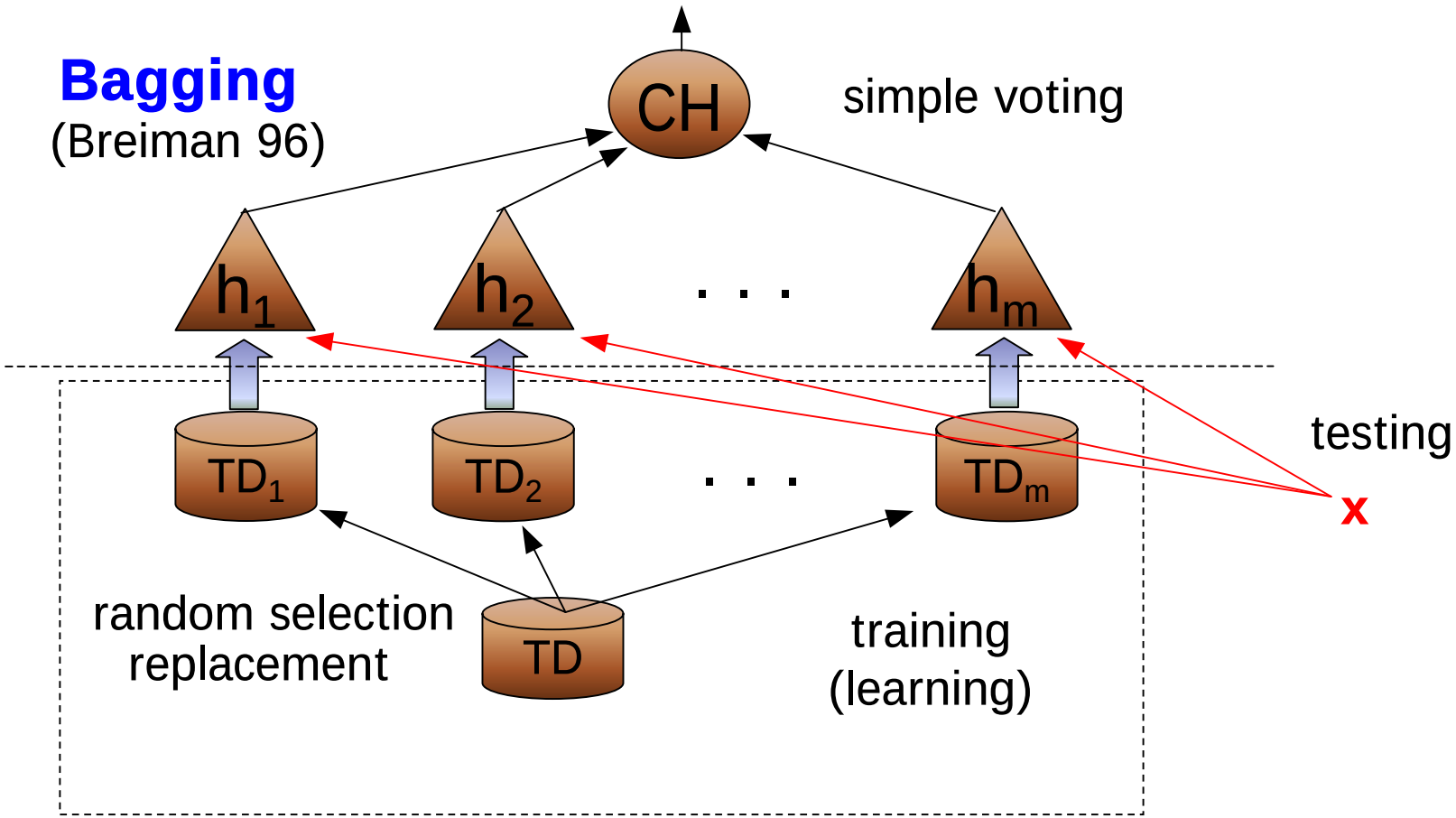
- This means that the training data will contain approximately 63.2% of the examples.

Predictors

- Let D be a training data set $D=\{(x_i, y_i) \mid x_i \text{ in } X, y_i \text{ in } Y\}$
- A predictor h: X ➔ Y is a function that for any given x, it produces y=h(x).
- A learning algorithm produces a predictor h_i for random selection S[*i*].
- Types of predictors:
  - Classifiers: **DTs, DLs**,…
  - Others

Bagging

$h^*(x) = \text{maxcnt}_i\ h_i(x)$



Bagging Frame

Let the original training data be D

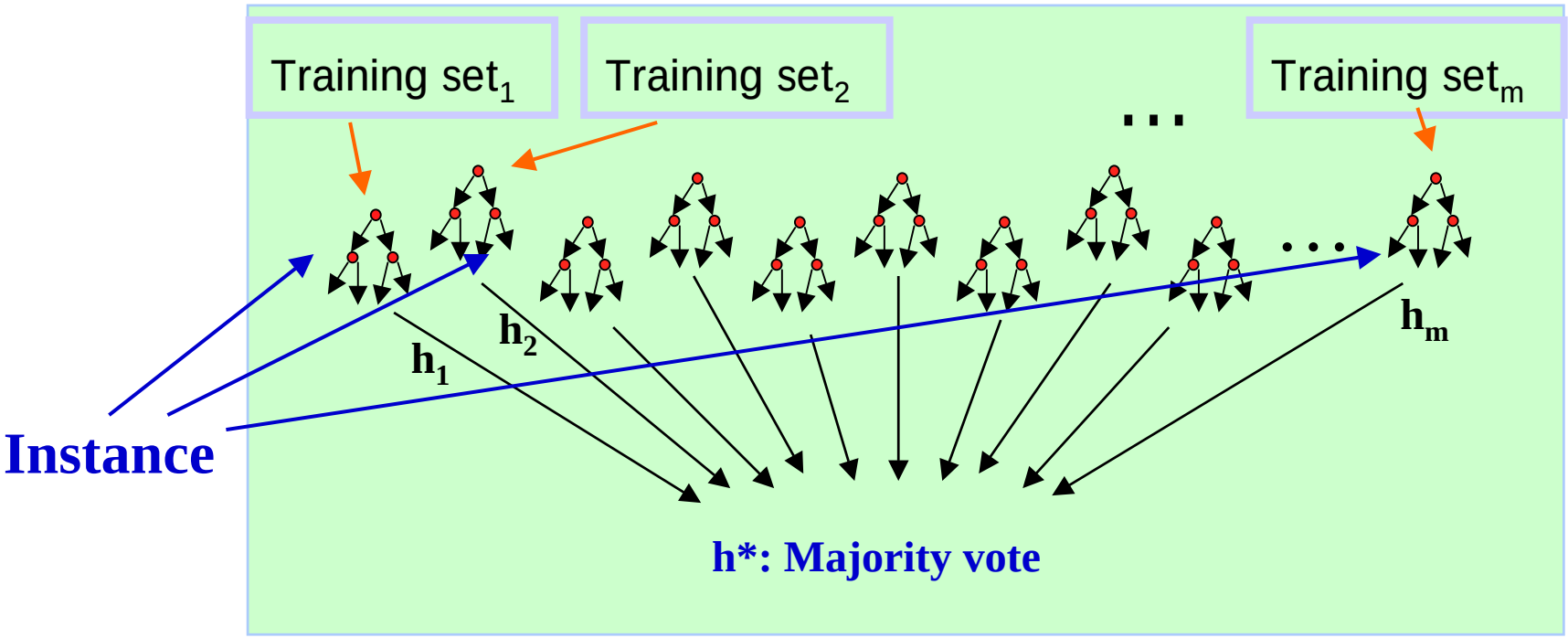
- Repeat m times:
  - Select a sample S[i]from D with replacement randomly.
  - Train a predictor using S[i] to obtain h_i.
- Combine m predictors (h_1 … h_m) by
  - Voting (for classification problem)
  - Averaging (for estimation problem)
  - …

When to use Bagging

- "The vital element is the **instability** of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy." (Breiman 1996)
- **Instability**: the classification results will change saliently as the input data change slightly. E.g, decision tree, neural net, etc.
- Experimentally, bagging can help substantially for unstable predictor, can **somewhat degrade results for stable predictor**.

Example: Bagged Decision Trees

- Draw m bootstrap samples of data
- Train decision trees on each sample → m trees
- majority votes on testing instance



Outline

- Overview
- Bagging
- ➢ **Boosting & Adaboost**
- Summary

Overview of boosting

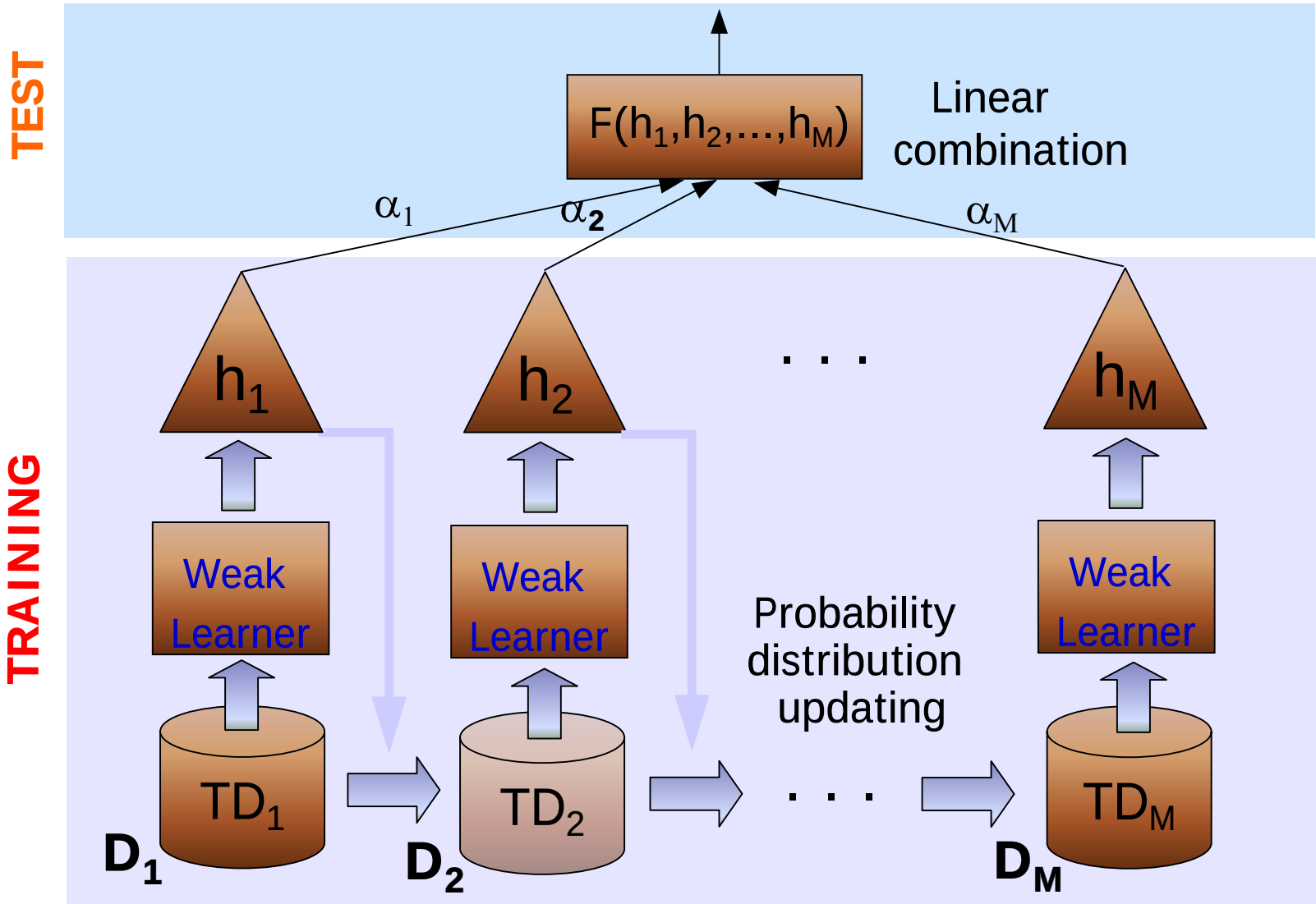
- Introduced by Schapire and Freund in 1990s.
- “Boosting”: convert a weak learning algorithm into a strong one.
- Main idea: Combine many **weak** classifiers(predictor) to produce a powerful committee.
- Algorithms:
  - **AdaBoost**: adaptive boosting
  - Gentle AdaBoost
  - …

Early Boosting (Schapire 1989)

- Randomly select $n_1 < n$ samples from D without replacement to obtain D_1
  - Train weak learner C_1
- Select $n_2 < n$ samples from D with half of the samples misclassified by C_1 to obtain D_2
  - Train weak learner C_2
- Select all samples from D that C_1 and C_2 disagree on
  - Train weak learner C_3
- Final classifier is vote of weak learners

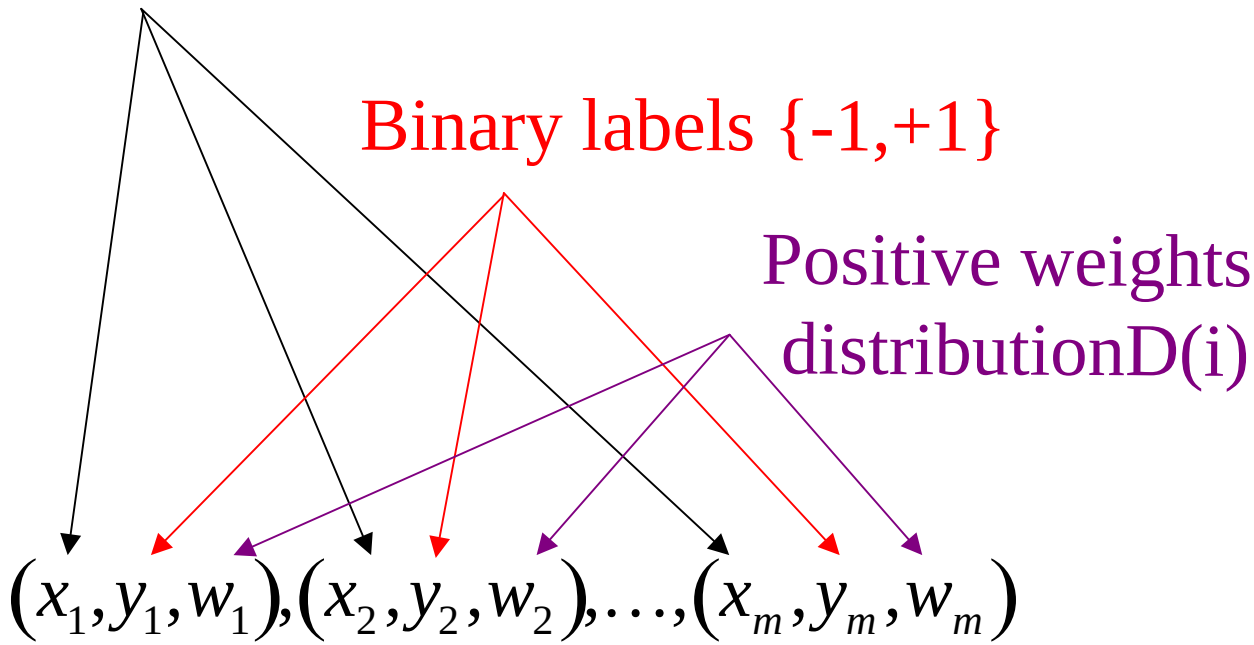
General Idea of Boosting

- Weak Classifiers (error rate slightly better than random, or ,the error rate < 50%).
- **Sequentially** apply weak classifiers to modified versions of data.
- Predictions of these classifiers are combined to produce a powerful classifier.
- Note the difference between it and “Bagging”.

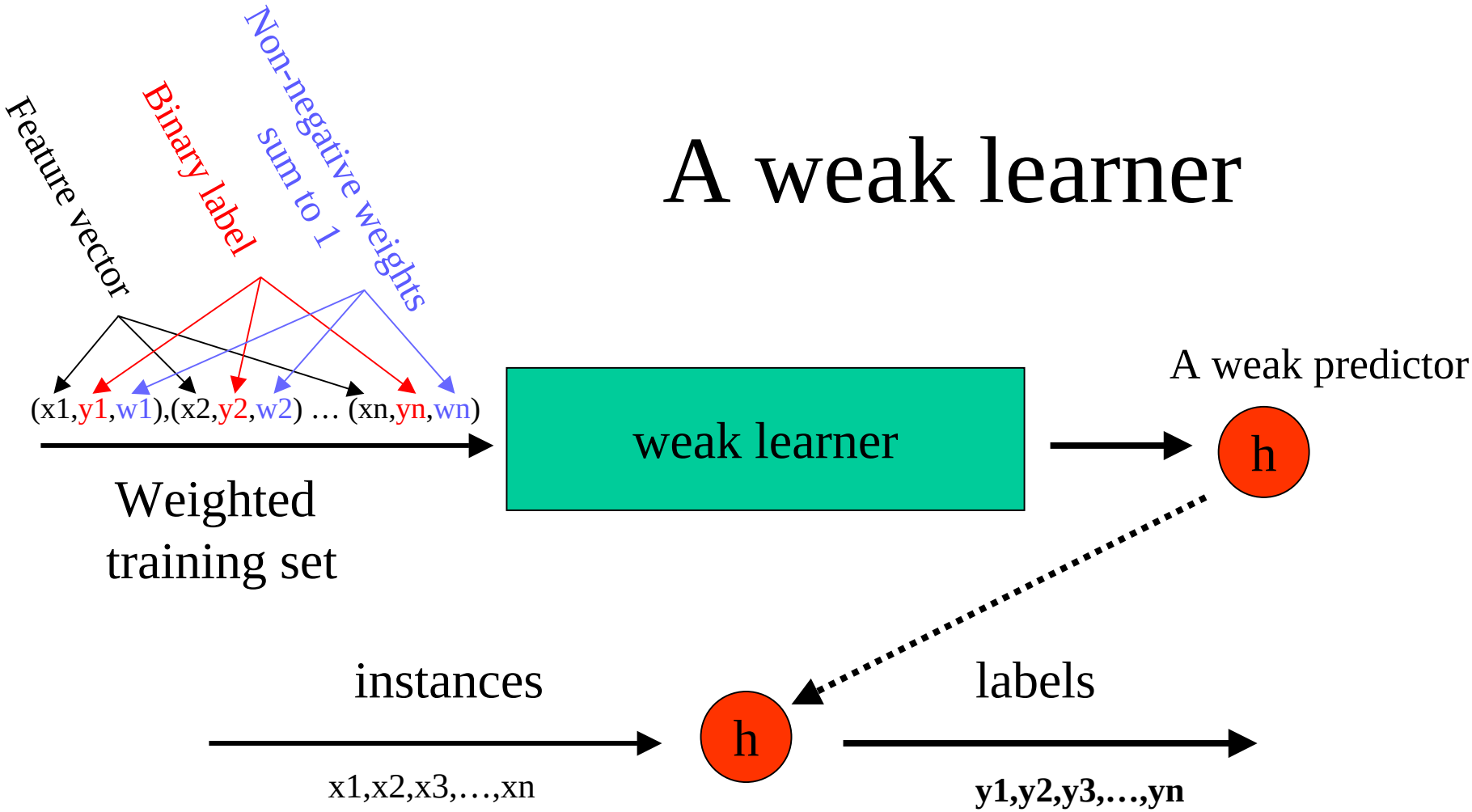

TEST
TRAINING
F(h1,h2,...,hM)
Linear combination
α1
α2
αM
h1
h2
. . .
hM
Weak Learner
Weak Learner
Weak Learner
Probability distribution updating
TD1
TD2
. . .
TDM
D1
D2
DM

A weighted training set

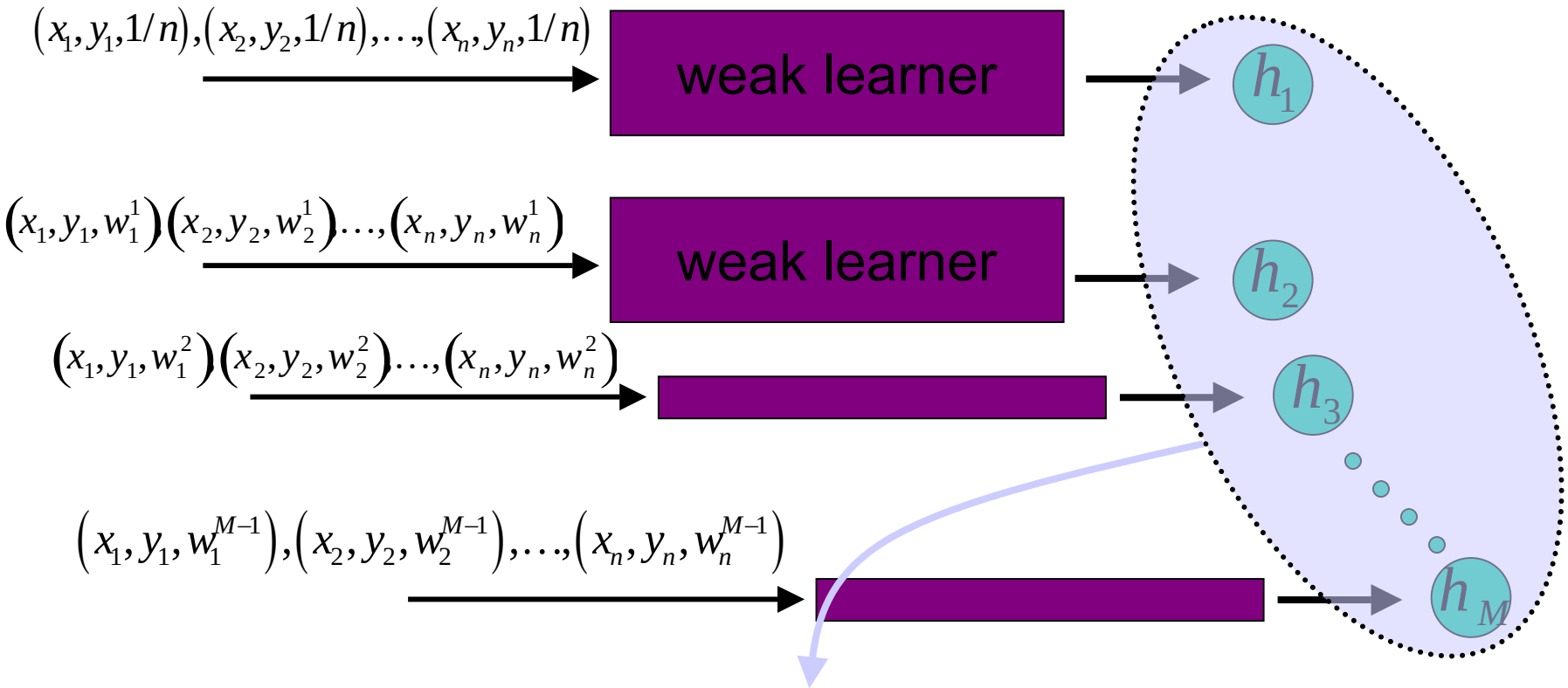
Feature vectors



A weak learner



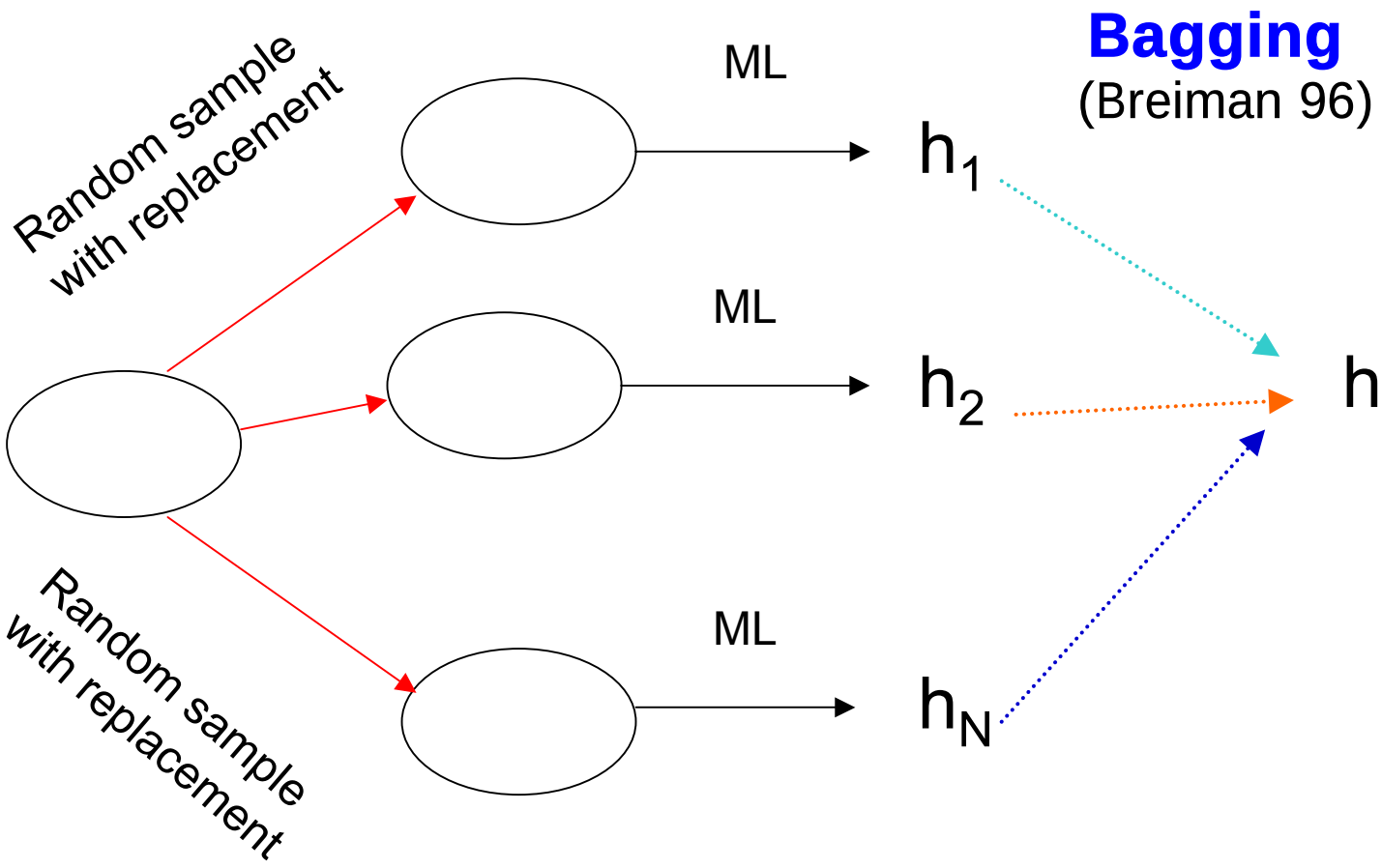
The weak requirement: $\frac{\sum_{i:y_i \neq \mathbf{y}_i} w_i}{\sum_{i=1}^{n} w_i} < \frac{1}{2} - \gamma$



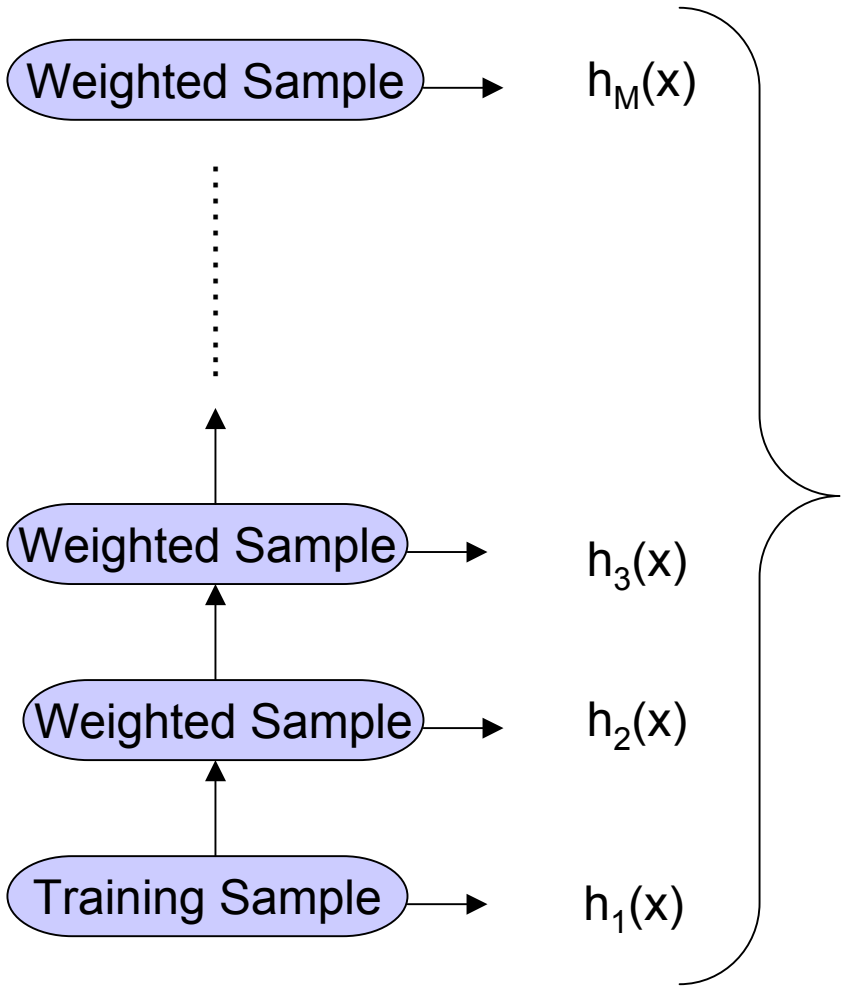
$$H_M(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_M h_M(x)$$

Final rule: $h(x) = \text{sign}\left(H_M(x)\right)$

Comparison


Bagging
(Breiman 96)
Random sample with replacement
ML
ML
ML
Random sample with replacement
h_1
h_2
h_N
h

Schematic of boosting



$$h(x) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right).$$

Intuition

- Train a set of weak hypotheses: h_1, …., h_M.
- The combined hypothesis h is a **weighted** majority vote of the M weak hypotheses.

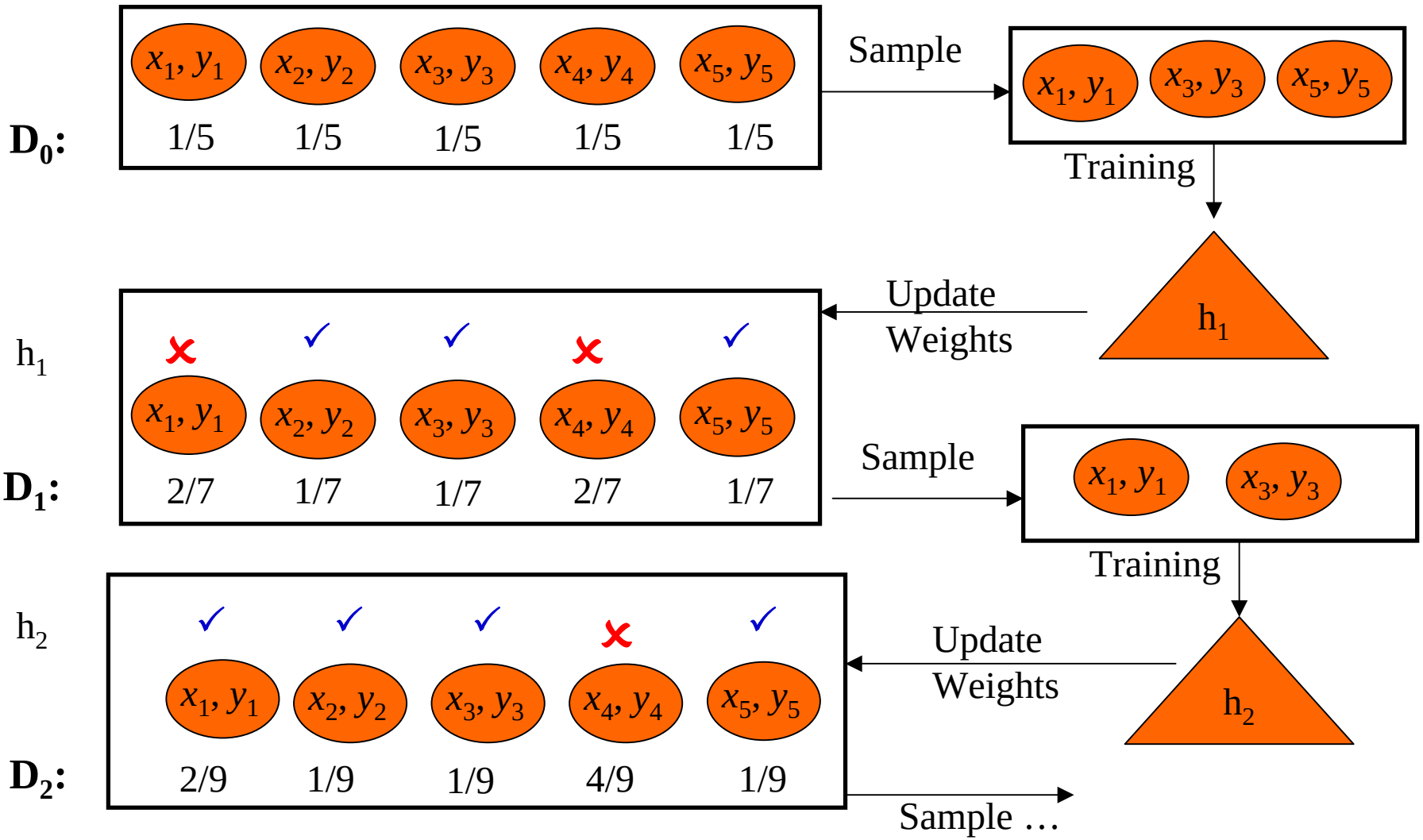
  ➔Each **hypothesis h_m** has a weight α_m.

$$h(x) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right).$$

- During the training, focus on the examples that are misclassified.

  ➔ At round m, **example x_i** has the weight $D_m(i)$.

AdaBoost Example



Basic Setting

- Binary classification problem
- Training data:

$$(x_1, y_1), ..., (x_n, y_n), where \;\; x_i \in X, y_i \in Y = \{-1, 1\}$$

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

$$\varepsilon_m = Pr_{i \sim D_m}(h_m(x_i) \neq y_i) = \sum_{i=1}^{n} D_m(i).(y_i \neq h_m(x_i))$$

The basic AdaBoost algorithm

For m=1, …, M

- Train weak learner using training data
- Get h_m : X ➔ Y={-1,1} with error

$$\varepsilon_m = \sum_{i:h_m(x_i)\neq y_i} D_m(i)$$

- Choose $\alpha_m = \frac{1}{2}\ln\frac{1-\varepsilon_m}{\varepsilon_m}$ (>0)

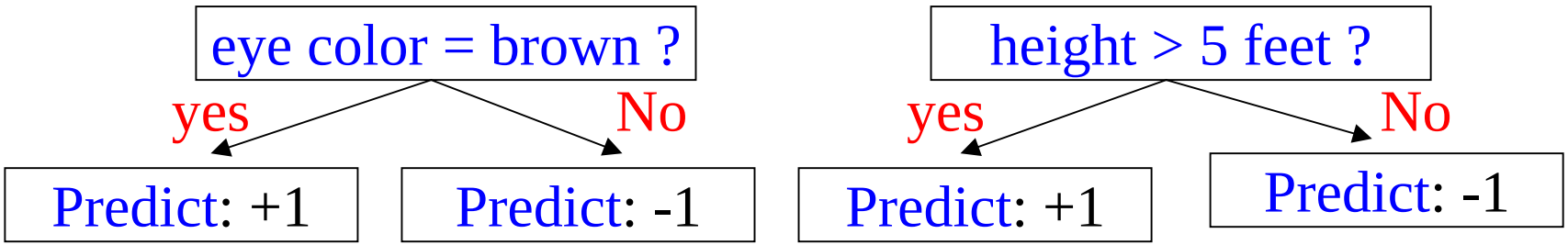
- Update

$$D_{m+1}(i) = \frac{D_m(i)}{Z_m} * \begin{cases} e^{-\alpha_m} & if\ h_m(x_i) = y_i \\ e^{\alpha_m} & if\ h_m(x_i) \neq y_i \end{cases}$$

$$= \frac{D_m(i)e^{-\alpha_m y_i h_m(x_i)}}{Z_m}, \text{ where, } Z_m \text{ is a normalization factor}$$

Choice of Weak Learner

- Neural Network
- Decision Tree(DT): C4.5/ID3
  - recursive partition of input space into a set of nested regions;
- Decision stumps: a simply one-level DT
  - A classifier formed by splitting the input space an axis-parallel fashion once(very simple rules of thumb that test on single attributes)



Analyzing the training error

- Theorm:
  - run AdaBoost

Let $\varepsilon_m = \frac{1}{2} - \gamma_m$

So,

$$\varepsilon_{Train}(h) \le \prod_{m=1}^{M} [2\sqrt{\varepsilon_m(1-\varepsilon_m)}] = \prod_{m=1}^{M} \sqrt{1-4\gamma_m^2} \le e^{-2\sum_m \gamma_m^2}$$

If $\forall \mathrm{m}, \gamma_{\mathrm{m}} \ge \gamma > 0$, then $\varepsilon_{Train}(h) \le e^{-2\gamma^2 M}$

$$h(x) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(x)\right).$$

More discussion

- The main objective is to minimize

$$\varepsilon_{Train} = \frac{1}{n}\sum_{i=1}^{n}(y_i \neq h(x_i))$$

Choosing a_m

- Select a_m to greedily minimize $Z_m(a)$ in each step
- $Z_m(a)$ is convex function with one extreme
- $h_m(x) \in \{-1,1\}$, then optimal

$$\alpha_m = \frac{1}{2}\log(\frac{1+r_m}{1-r_m})$$

$$where, \quad r_m = \sum_{i=1}^{n} D_m(i) h_m(x_i) y_i$$

AdaBoost: Pros&Cons

- Easy to implement and few parameters to set
- Time and space grow linearly with number of examples. Ability to manage very large learning problems
- Does not constrain explicitly the complexity of the learner
- Naturally combines feature selection with learning
- Has been succesfully applied to many practical problems
- Can perform poorly when there is insufficient training data relative to the complexity of the base classifiers, the training errors of the base classifiers become too large

AdaBoost Extensions: Multiclass

- $y \in Y = \{ 1, 2, \ldots k \}$;
- **Approach:**
- Straightforward extension: **AdaBoost.M1** (Freund & Schapire 97)
- **AdaBoost.MH** (Schapire & Singer 99)

  For each example *x* and each possible label *y*, create a binary problem of the form: "For example *x*, is the correct label *y* or is it one of the other labels?"

AdaBoost.M1

$$D_{m+1}(i) = \frac{D_m(i)}{Z_m} * \begin{cases} e^{-\alpha_m} & if\ h_m(x_i) = y_i \\ e^{\alpha_m} & if\ h_m(x_i) \neq y_i \end{cases}$$

$$= \frac{D_m(i) e^{-\alpha_m y_i h_m(x_i)}}{Z_m}$$, where, Z_m is a normalization factor

$$h(x) = \underset{y \in Y}{\text{Argmax}} \sum_m \alpha_m (h_m(x) = y)$$

AdaBoost.MH

- Formally, h_m: X × Y→{-1,1}

$$D_{m+1}(i,y) = \frac{D_m(i,y)}{Z_m} * e^{-\alpha_m v_i(y) h_m(x_i,y)}$$

$$where, v_i(y) = \begin{cases} 1 & if \ y_i = y \\ -1 & if \ y_i \neq y \end{cases}$$

$$h(x) = \underset{y \in Y}{\text{Argmax}} \sum_m \alpha_m h_m(x,y)$$

AdaBoost.MH applied to WSD

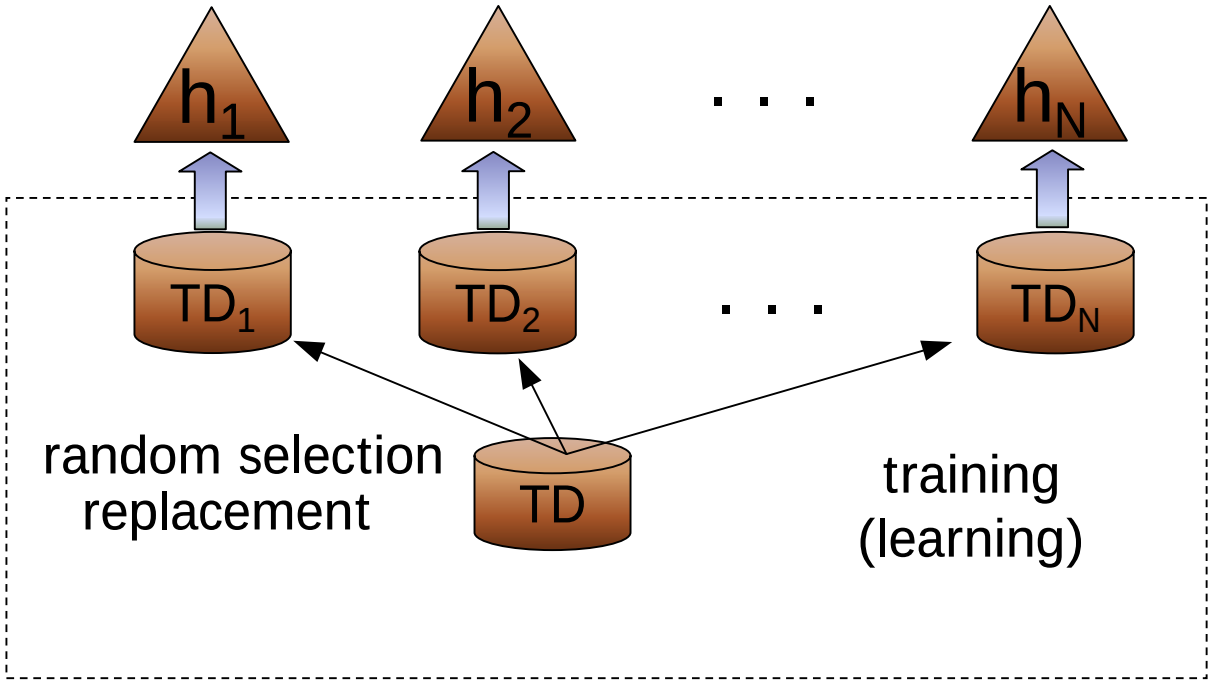
- 中文信息学报，2006(3).
- Decision Tree used as weak learner

Outline

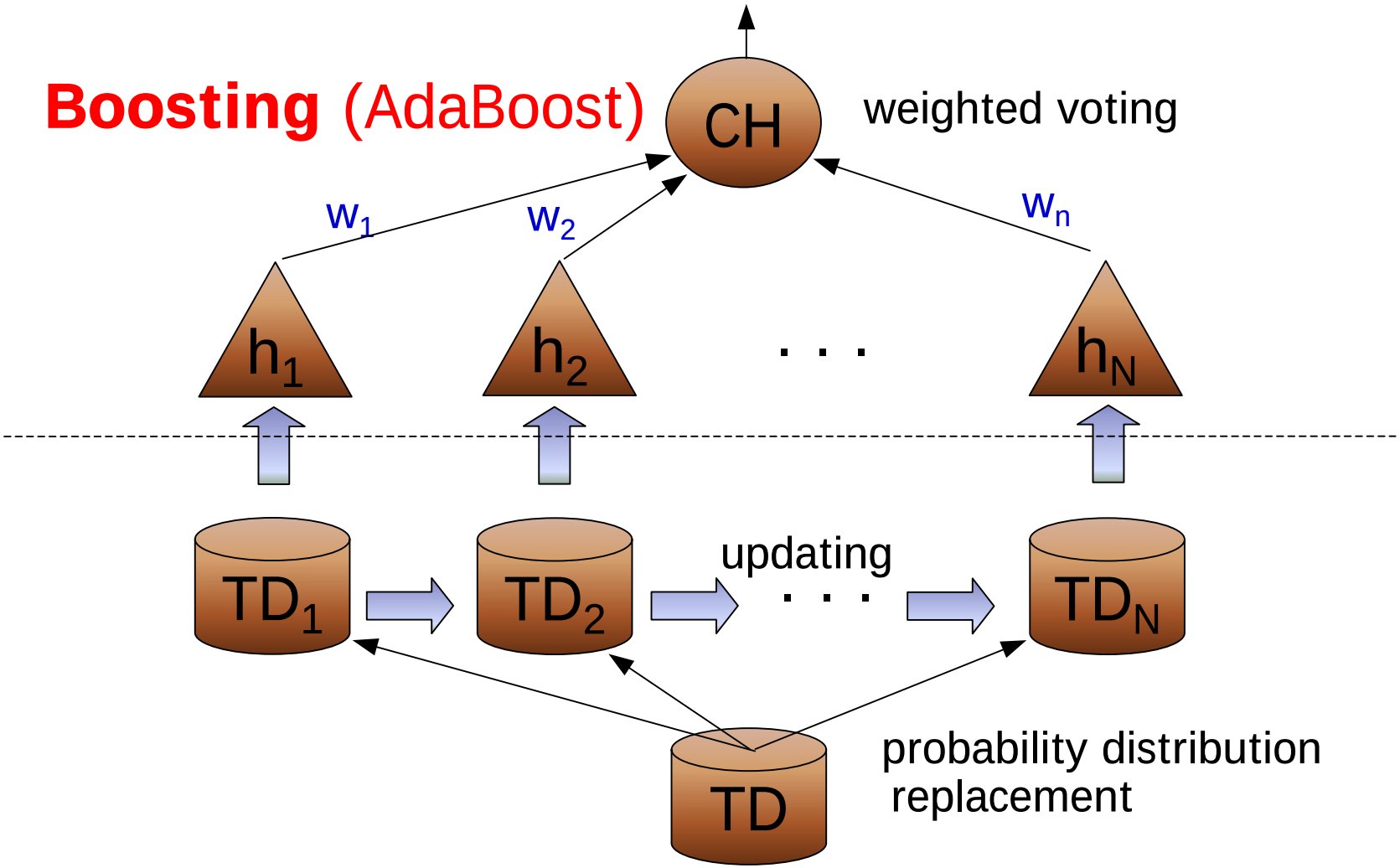
- Overview
- Bagging
- Boosting & Adaboost
- **Summary**

Re-sampling

Bagging



Re-sampling


Boosting (AdaBoost)
CH
weighted voting
w1
w2
wn
h1
h2
. . .
hN
TD1
TD2
updating
. . .
TDN
TD
probability distribution
replacement

Ensembles: Summary

- **Pros**
  - Very active area of research
  - Many different techniques can be applied
  - Impressive results are obtained

- **Cons**
  - Some learning methods are not well suited for the combination approach (e.g. Exemplar-based learning)
  - Some methods can degrade results in the presence of noise due to the tendency to overfit training data
  - Computational overhead