Bagging & Boosting

Qinliang Su （苏勤亮）

Sun Yat-sen University

suqliang@mail.sysu.edu.cn

Ensemble Methods Overview

- It is difficult to learn a *strong classifier* that can always classify instances correctly
- But it is easy to learn a lot of *'weak' classifiers*

  A weak classifier may not perform well on the whole dataset, but may perform well on a fraction of samples, *e.g.*, some may be good at recognizing 'cat', while some others may be good at recognizing 'dog'

- If weak classifiers perform well on different fractions of samples, it is possible to derive *a strong classifier by combining these weak classifiers in an appropriate way*
- Two questions
  1) How to produce these weak classifiers?
  2) How to combine the weak classifiers?

Two Types of Combining Methods

1) Unweighted average

   - Majority vote

2) Weighted average

   - Give better classifiers bigger weighting

For example, consider a two-class classification problem {-1, 1}

Two basic classifiers: $\hat{y}_1 = sign(f_1(\boldsymbol{x}))$ $\hat{y}_2 = sign(f_2(\boldsymbol{x}))$

Final classifiers: $\hat{y}_e = sign\left(\alpha_1 f_1(\boldsymbol{x}) + \alpha_2 f_2(\boldsymbol{x})\right)$

Remark: The weak classifiers could be of any kind, *e.g.* decision trees, SVM, neural networks, logistic regression *etc.*

Outline

- Ensemble methods
  - Bagging (majority vote)
  - Boosting (weighted average)

Deriving Weak Classifiers

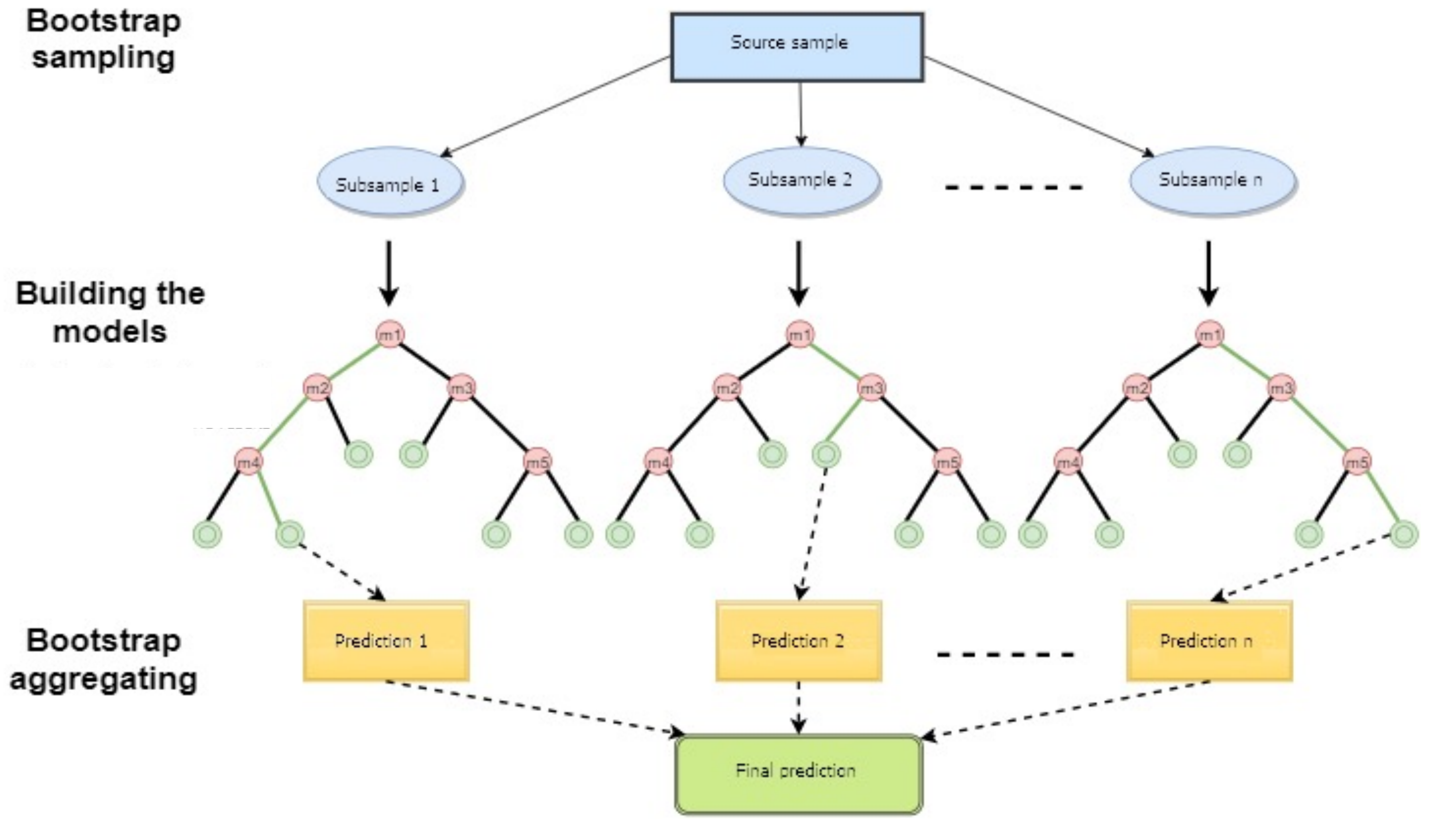
- We don't know how to obtain classifiers that perform well on different fractions of samples

- Instead, we attempt to obtain classifiers that produce independent prediction errors, that is, encouraging their predictions *to be uncorrelated*. For example,

1) Creating subsets of the training dataset by bootstrapping
   - Randomly draw N' samples from the N-sample training dataset *with replacement*
   - Repeat the above procedure K times, generating subsets $S_1, S_2, \cdots, S_K$

2) Training a decision tree on each of the subset S_k

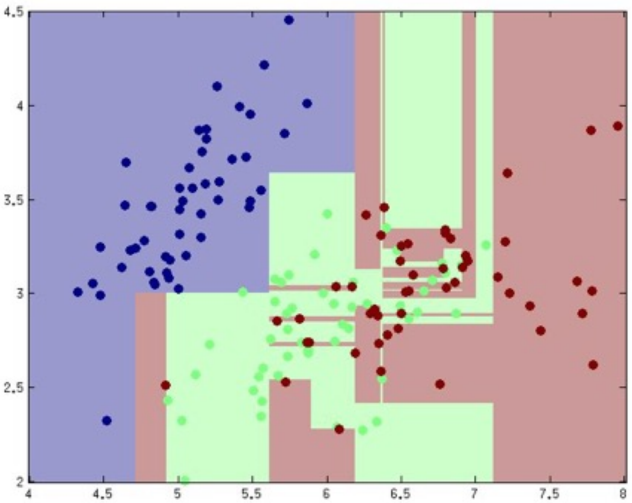
3) Combine K decision trees into one by majority voting



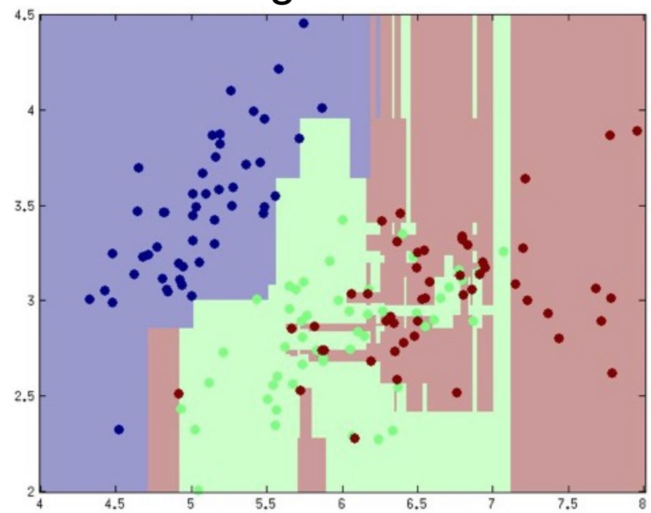
At testing, pass test data through all K classifiers, using the majority voting result as the final prediction

Example: Bagged Decision Trees

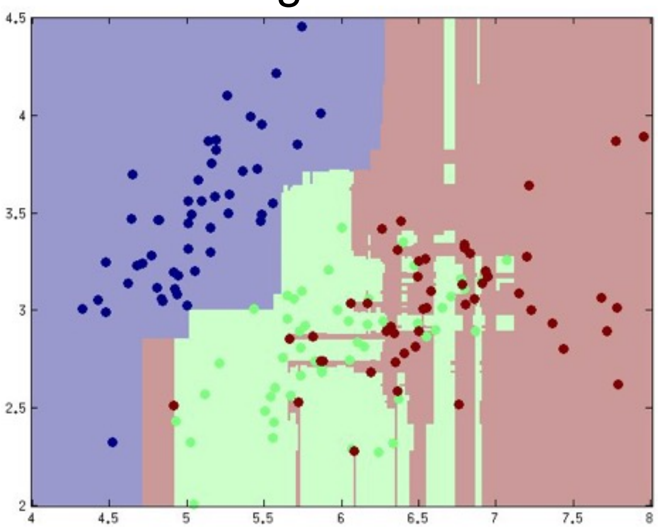
1 tree



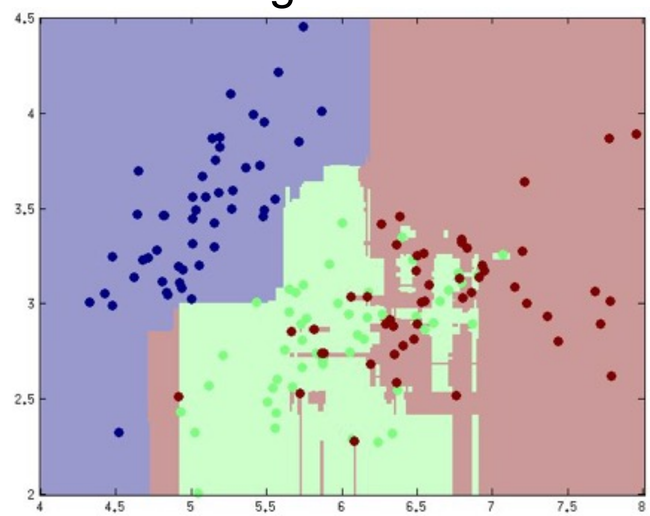
Average of 5 trees



Average of 25 trees



Average of 100 trees



Random Forest

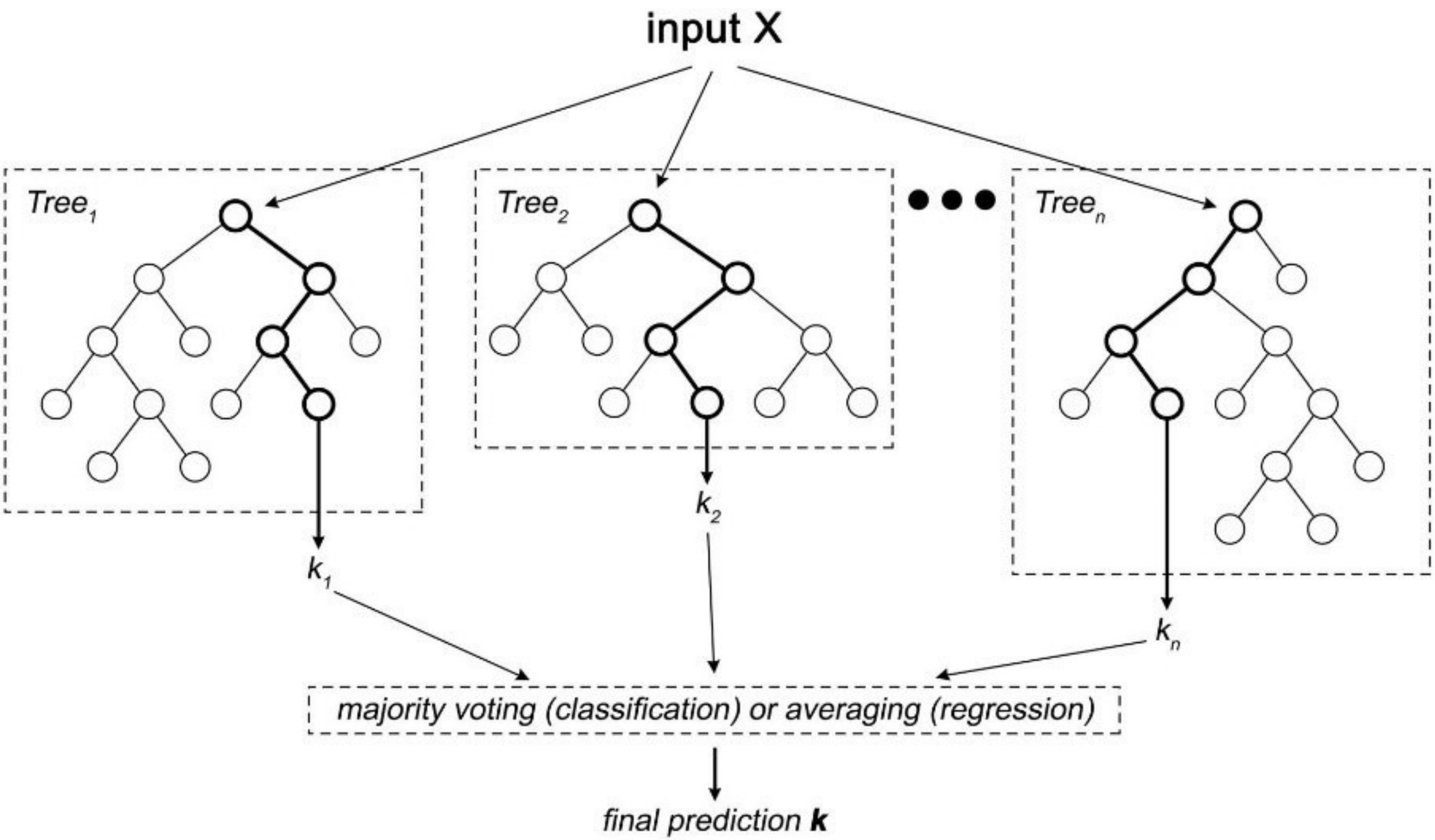
- What happens if the prediction errors in the classifiers derived above are *still strongly correlated* ?

  ⇒ Combining them by majority vote may not help too much on the final performance

- *Remedy:* Introducing extra randomness into the learning process of decision trees

As building the decision trees, only use a subset of randomly selected attributes

- Random forest illustration



Outline

- Ensemble methods
  - Bagging (majority vote)
  - Boosting (weighted average)

Overview of Boosting Methods

- *Weak classifiers derivation*

  Repeat the following steps several times

  1) Identifying the examples that are incorrectly classified
  2) Re-training the classifier by *giving more weighting to the misclassified examples*

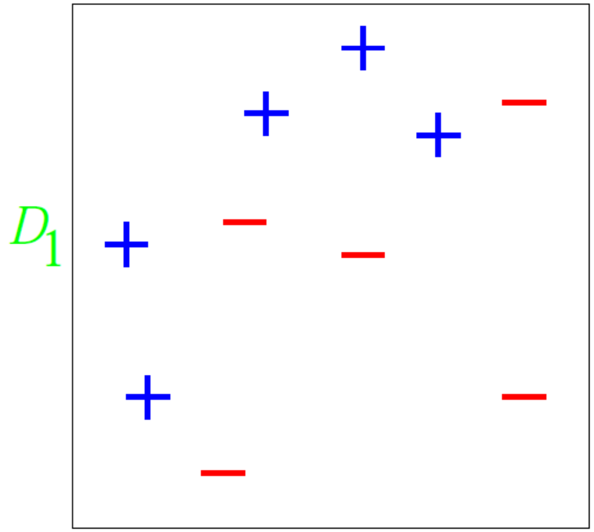
- *Combining*

  Combing the prediction results of each classifier by *weighted average*

How to *weight the examples and prediction* results is the key

Adaboost

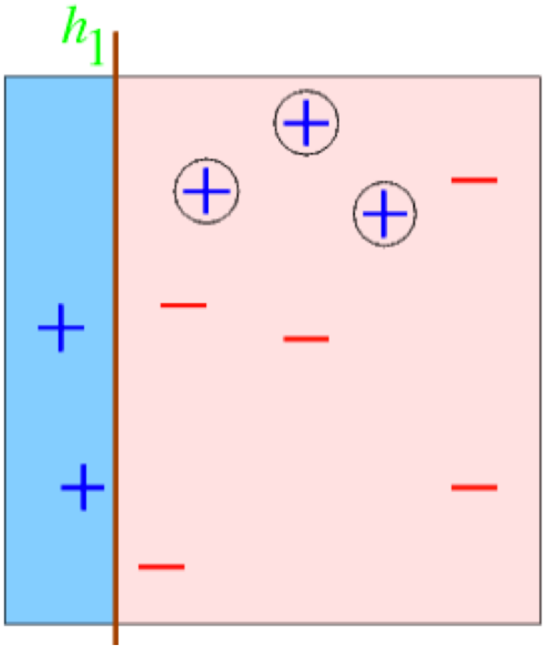
- Consider a two-class classification problem with 10 training examples



- For simplicity, only consider the weak classifiers whose decision boundaries are parallel to the axes, that is,

$$\hat{y} = sign(x_1 + b) \text{ or } \hat{y} = sign(x_2 + b)$$

- First iteration

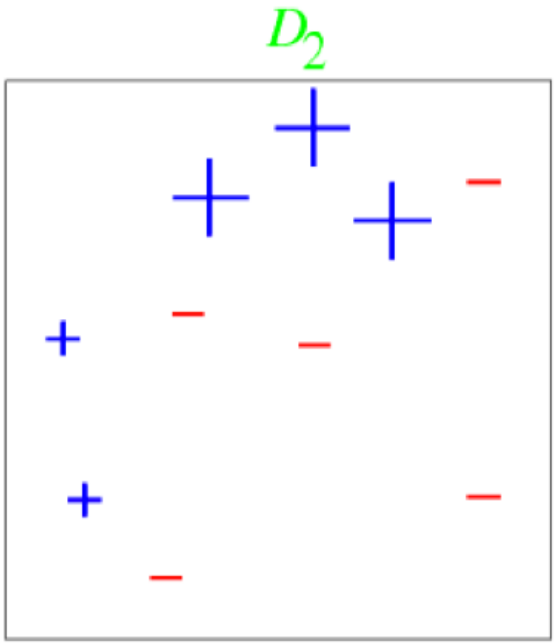


- Error rate of the first classifier h_1: $\epsilon_1 = 0.3$

- Weighting of the classifier h_1: $\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = 0.42$

$$\frac{1-\epsilon_1}{\epsilon_1} = \frac{\text{correct rate}}{\text{error rate}}$$

$\Rightarrow$ The weighting is *positively proportional* to performance of the classifier

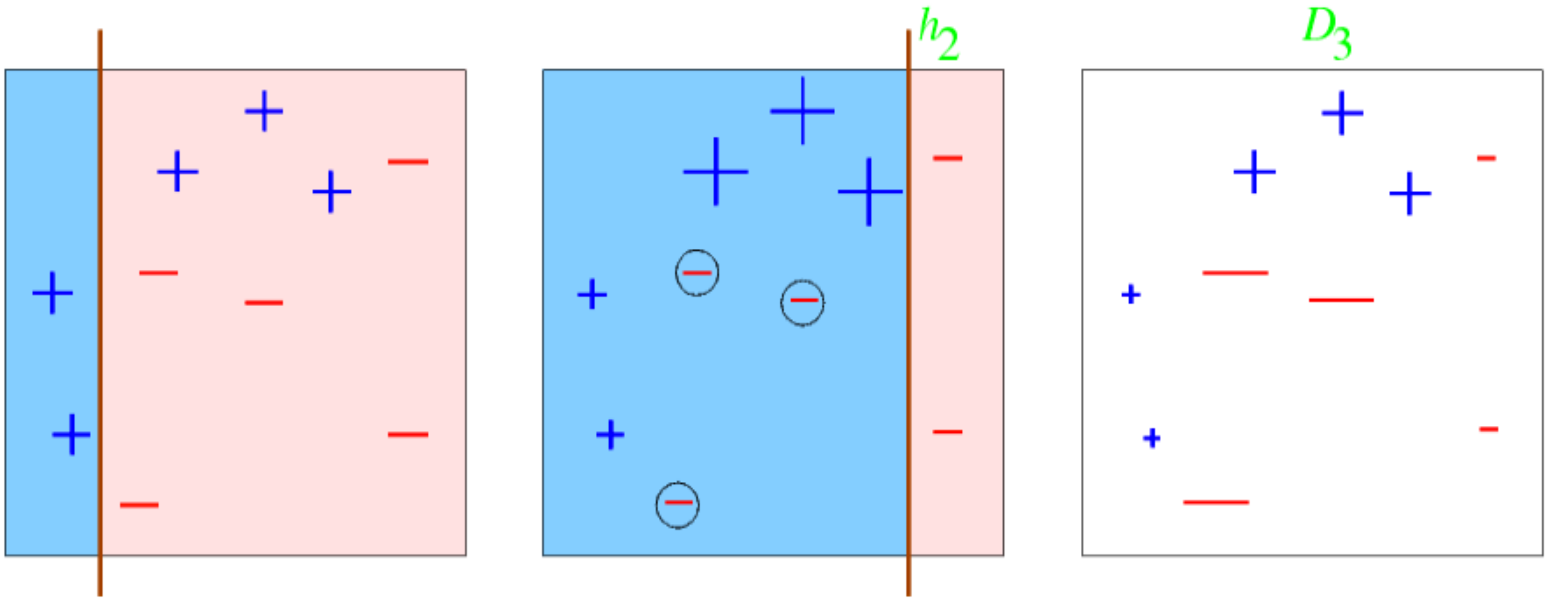


- *Misclassified* examples' weights are amplified by e^{α_1}

$$e^{\alpha_1} = \sqrt{\frac{1-\epsilon_1}{\epsilon_1}} = \sqrt{\frac{\text{correct rate}}{\text{error rate}}}$$

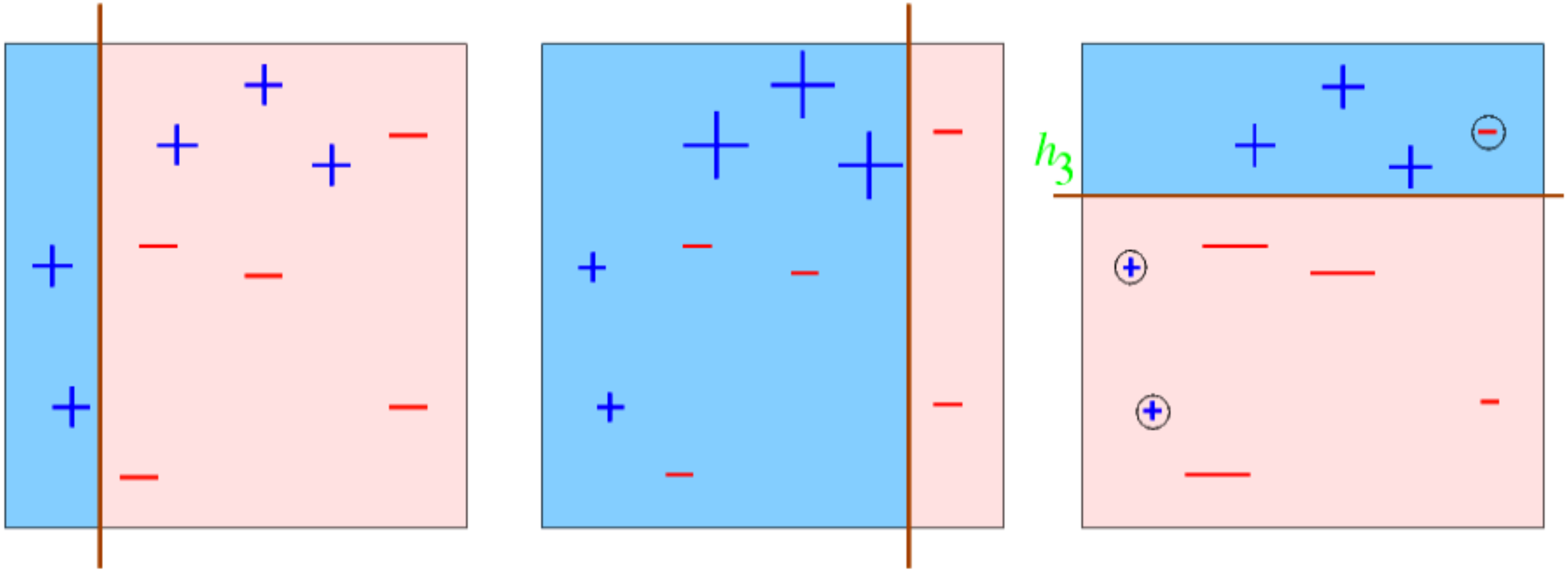
- Correctly classified examples' weights are dampened by $e^{-\alpha_1}$

- Second iteration



- Error rate of the second classifier h_2: $\epsilon_2 = 0.21$
- Weighting of the classifier h_2: $\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_2}{\epsilon_2}\right) = 0.65$
- Misclassified examples' weights are amplified by $e^{\alpha_2} = \sqrt{\frac{1-\epsilon_2}{\epsilon_2}}$
- Correctly classified examples' weights are dampened by $e^{-\alpha_2} = \sqrt{\frac{\epsilon_2}{1-\epsilon_2}}$

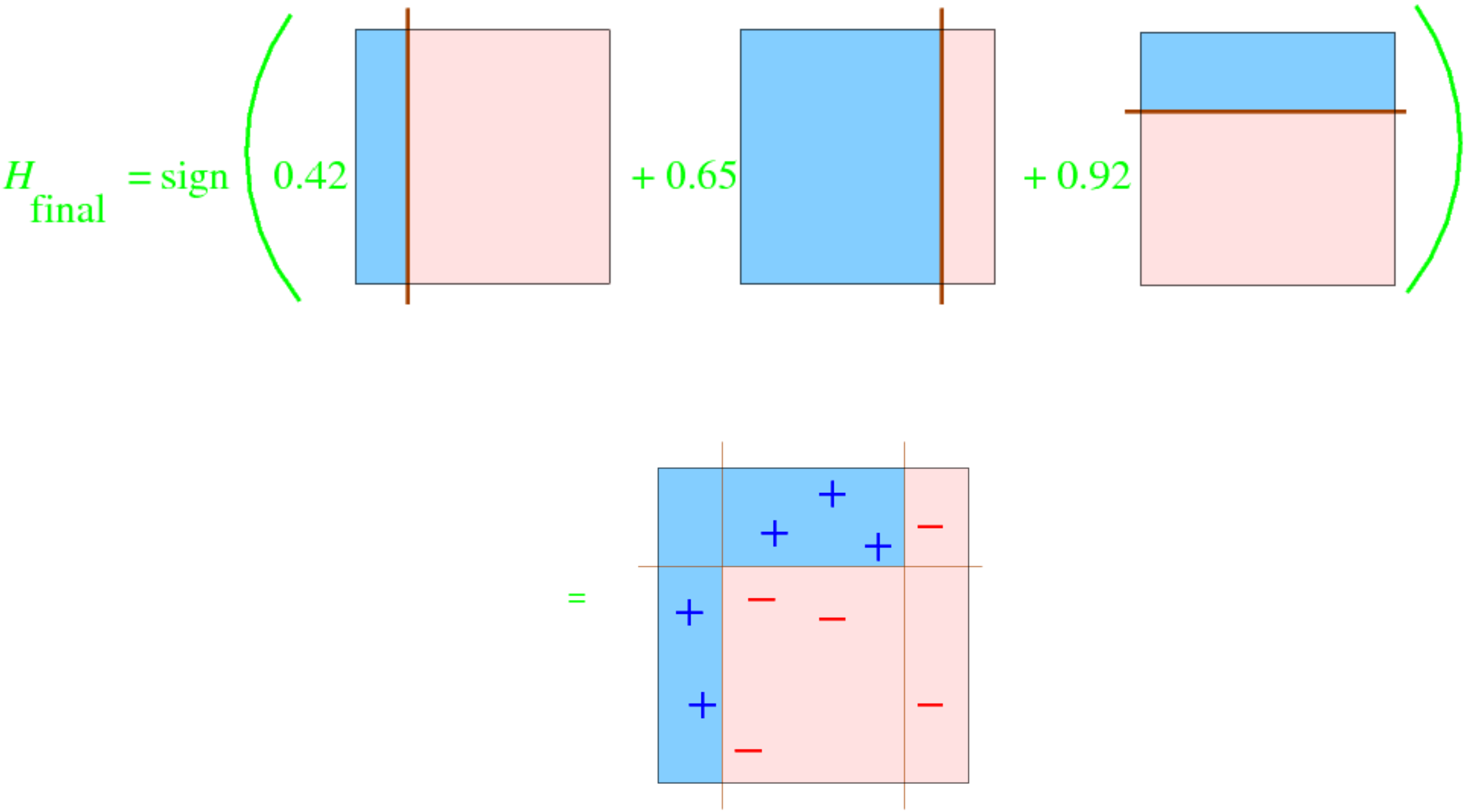
- Third iteration



- ➢ Error rate of the second classifier h_3: $\epsilon_3 = 0.14$
- ➢ Weighting of the classifier h_3: $\alpha_3 = \frac{1}{2}\ln\left(\frac{1-\epsilon_3}{\epsilon_3}\right) = 0.92$
- ➢ Stop the iteration

- Final classifier

Combining the three classifiers with a linear combination



- Adaboost algorithm

1) Initialize the weight of examples as $\omega_0^{(n)} = \frac{1}{N}$ for $n = 1, \cdots, N$

2) For the k-th iteration, train a classifier $h_k(\boldsymbol{x})$ with the training examples weighted by $w_{k-1}^{(n)}$

3) Evaluate the weighted classification error

$$\epsilon_k = \frac{\sum_{n=1}^{N} \omega_{k-1}^{(n)} I\left(y_i \neq h_k(\boldsymbol{x}_n)\right)}{\sum_{n=1}^{N} \omega_{k-1}^{(n)}}$$

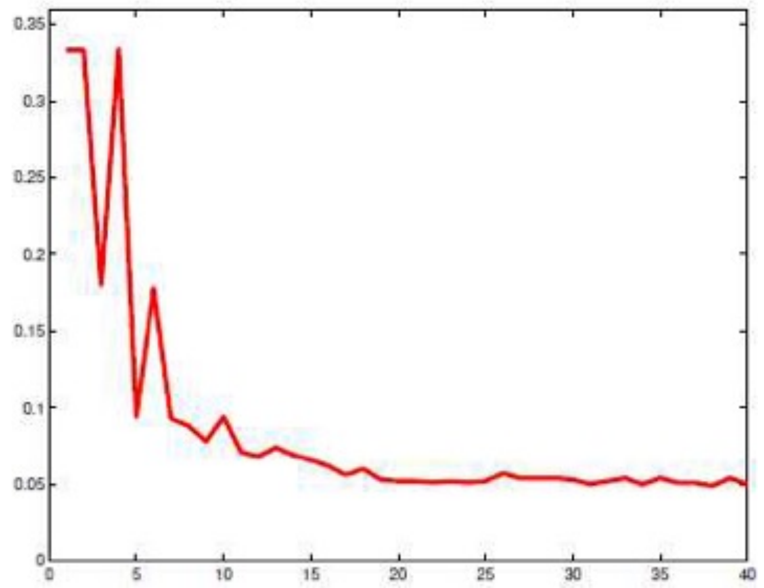
4) Determine the vote stake of the k-th classifier

$$\alpha_k = \frac{1}{2} \ln\left(\frac{1-\epsilon_k}{\epsilon_k}\right)$$

5) Update the weights of examples as

$$\omega_k^{(n)} = \omega_{k-1}^{(n)} \exp\{-y_i h_k(\boldsymbol{x}_i) \alpha_k\}$$

- A typical error rate curve as a function of the number of weak classifiers



- Typical weak classifiers
  - Decision trees
  - Logistic regressions
  - Neural networks

Theories behind the Adaboost

- Define the following *exponential loss*

$$\mathcal{L} = \sum_{n=1}^{N} \exp\left\{-y^{(n)} h_{combine}\left(\boldsymbol{x}^{(n)}\right)\right\}$$

where $\boldsymbol{x}^{(n)}$ is the input, $y^{(n)} \in \{-1, 1\}$ is the label; and $h_{combine}(\cdot)$ is the combined classifier

$$h_{combine}(\boldsymbol{x}) = \alpha_1 h_1(\boldsymbol{x}) + \cdots + \alpha_K h_K(\boldsymbol{x})$$

with $h_k(\boldsymbol{x})$ representing the k-th component classifier, *e.g.*, $h_k(\boldsymbol{x}) = sign(\boldsymbol{w}_k^T \boldsymbol{x} + b_k)$

- It can be proved that the Adaboost algorithm is *equivalent to minimize the exponential loss* in a sequential fashion