Ensemble Learning

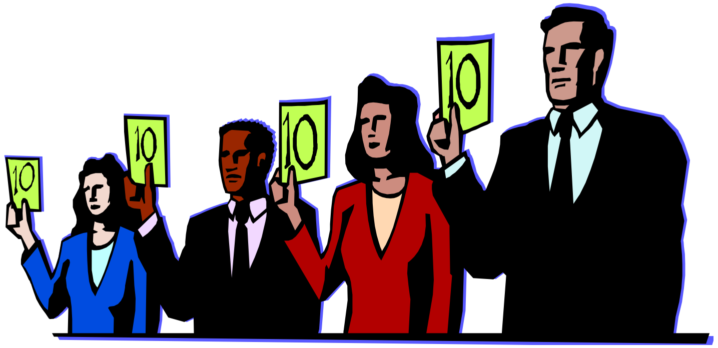
Lecturer: Dr. Bo Yuan

E-mail: yuanb@sz.tsinghua.edu.cn

Real World Scenarios

VS.

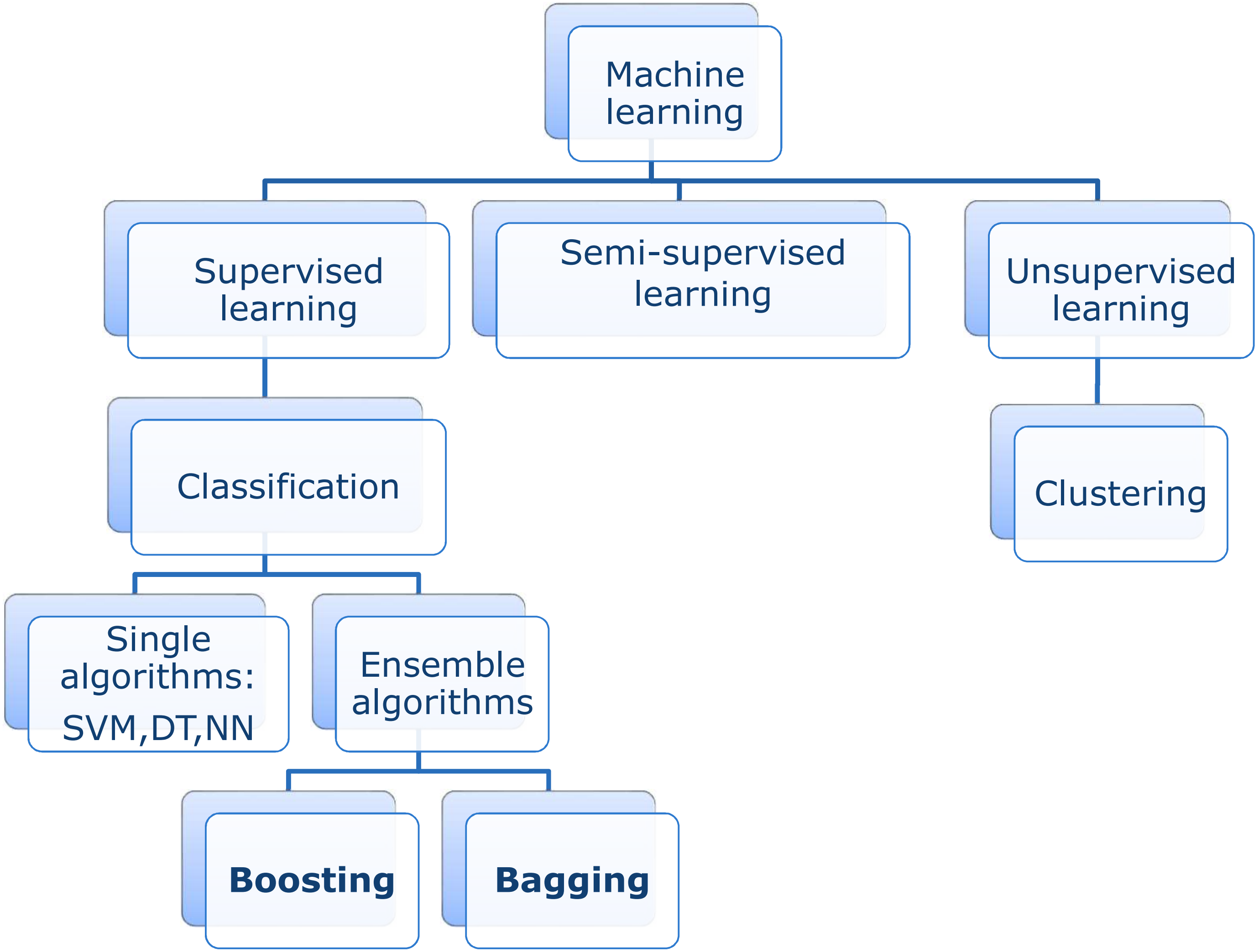
Real World Scenarios



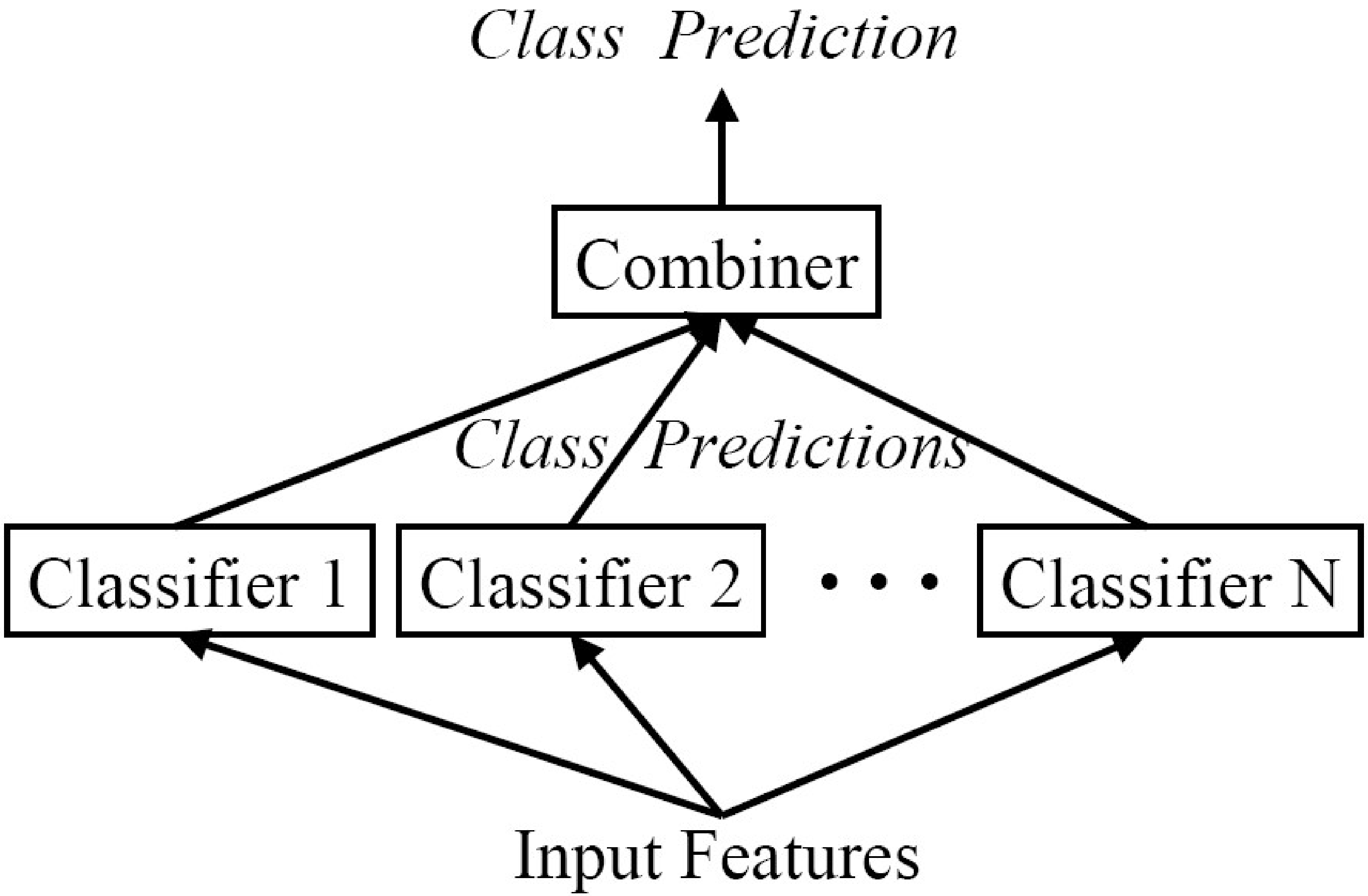
What is ensemble learning?

- Many individual learning algorithms are available:
  - Decision Trees, Neural Networks, Support Vector Machines
- The process by which multiple models are strategically generated and combined in order to better solve a particular Machine Learning problem.
- Motivations
  - To improve the performance of a single model.
  - To reduce the likelihood of an unfortunate selection of a poor model.
- Multiple Classifier Systems
- One idea, many implementations
  - Bagging
  - Boosting

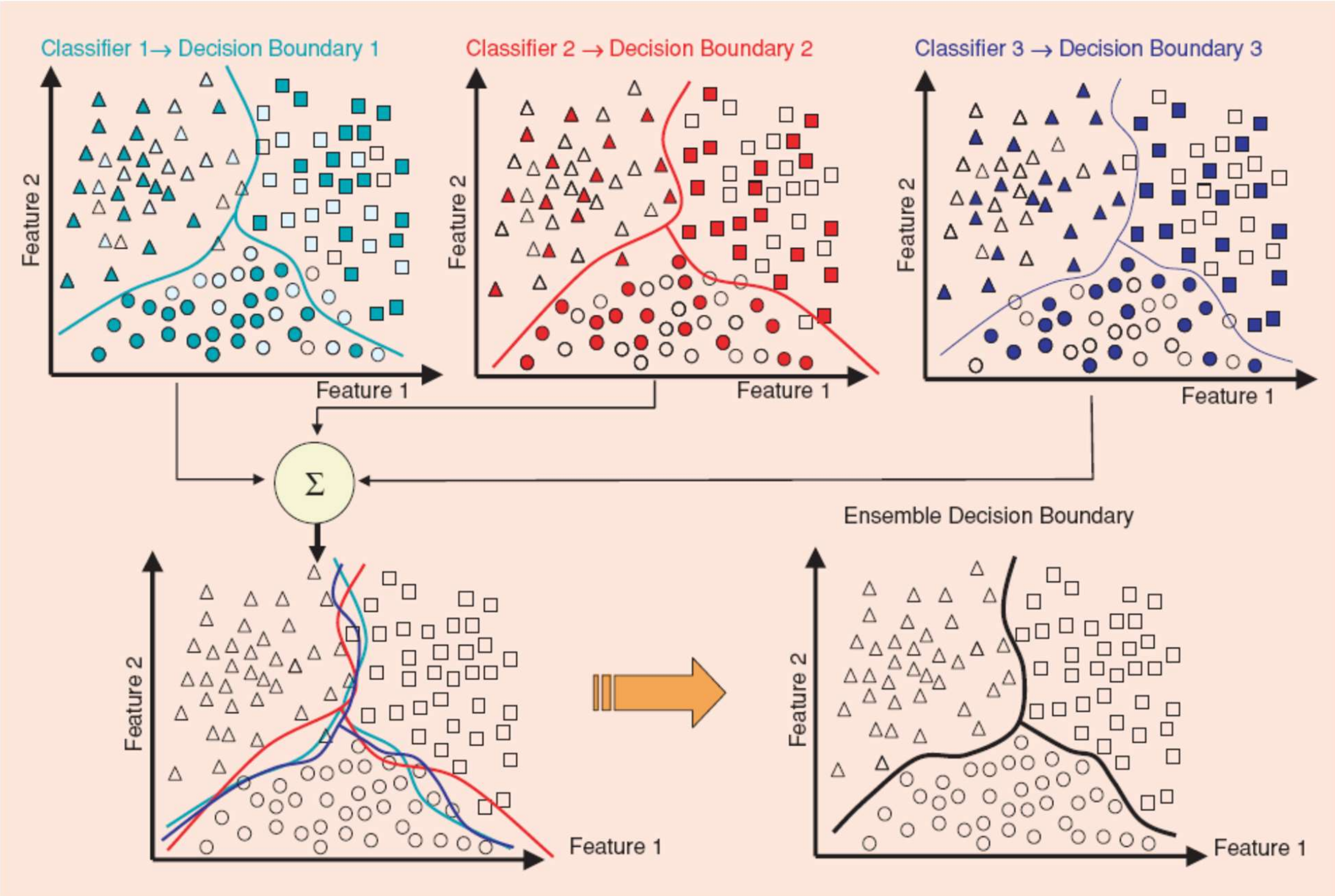
Algorithm Hierarchy



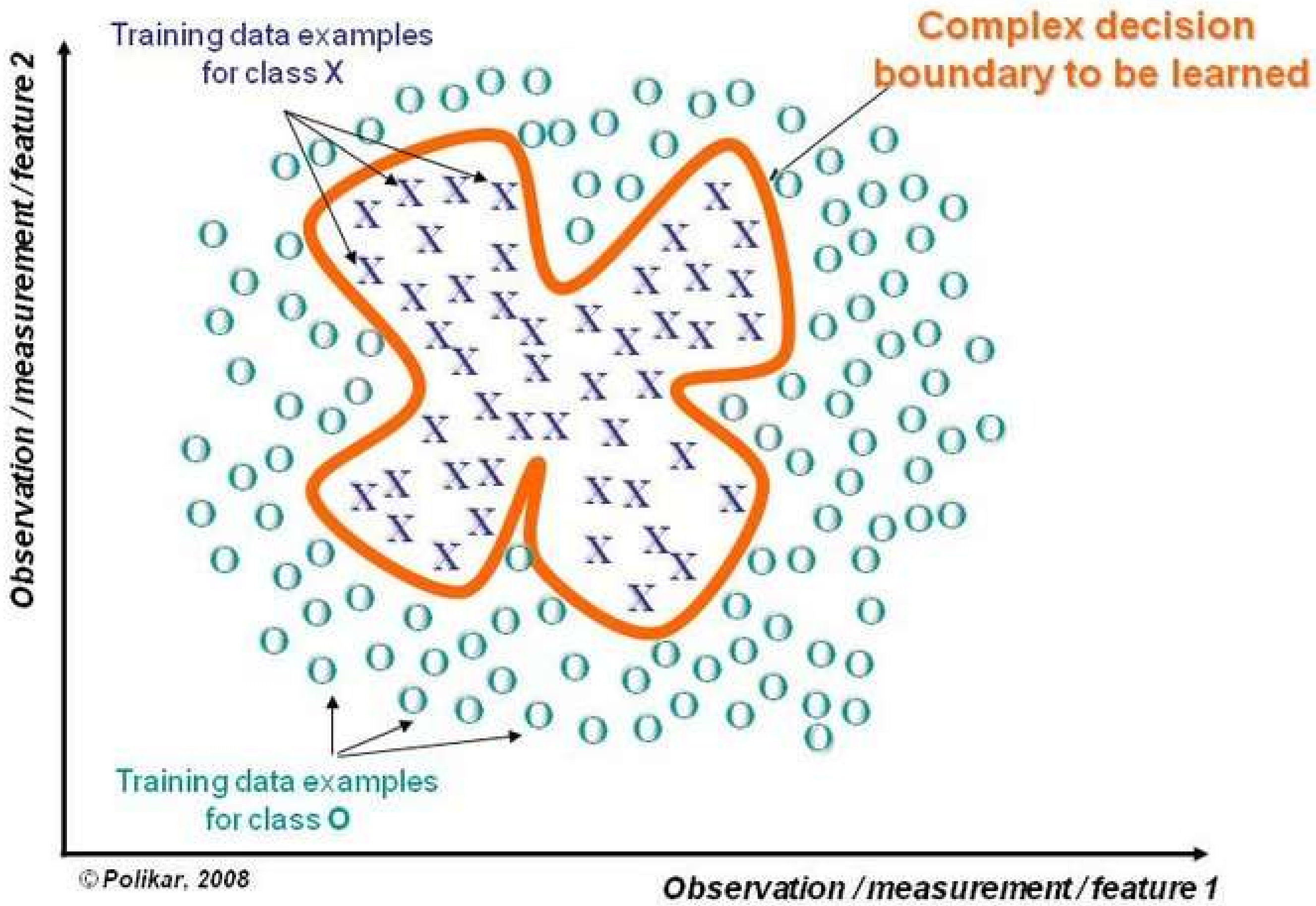
Combination of Classifiers



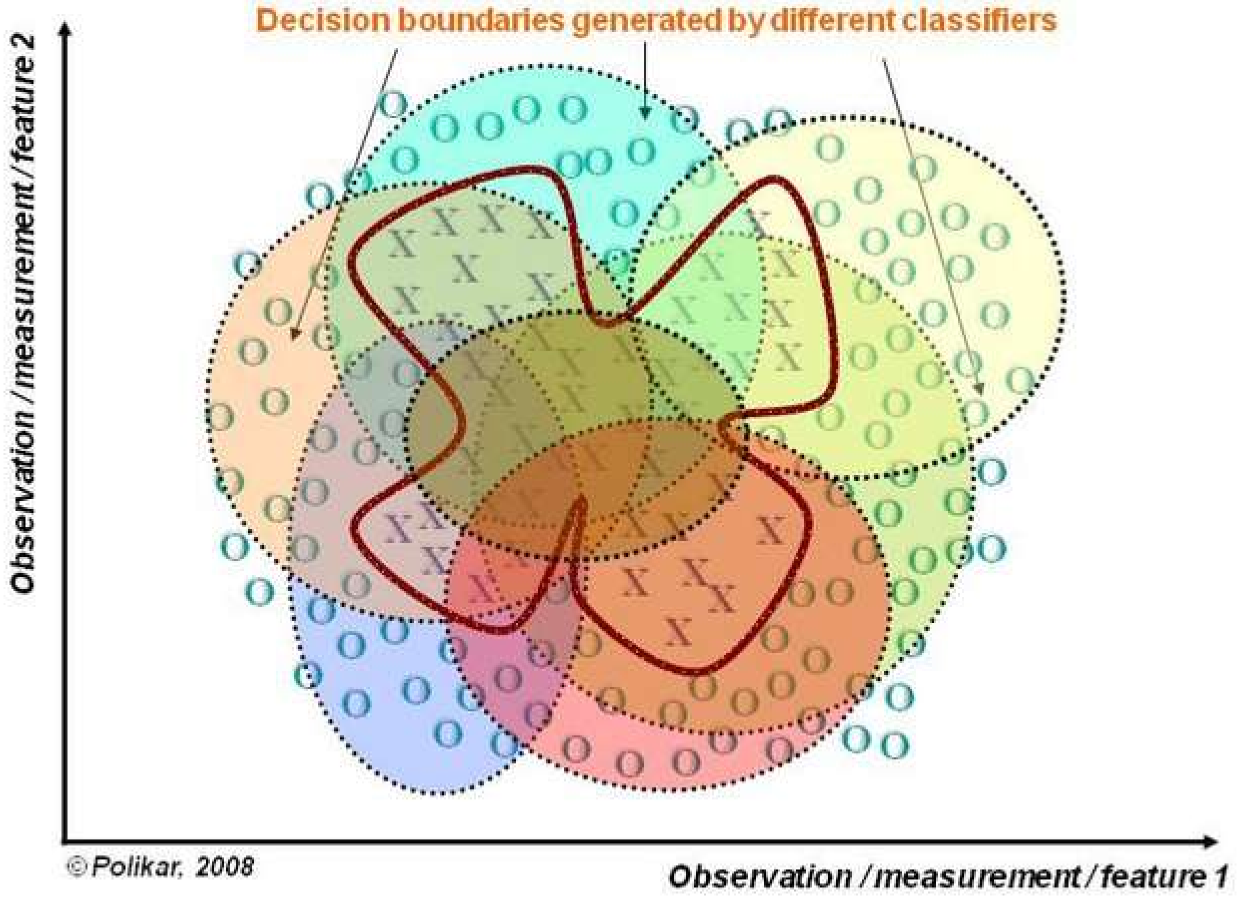
Model Selection



Divide and Conquer

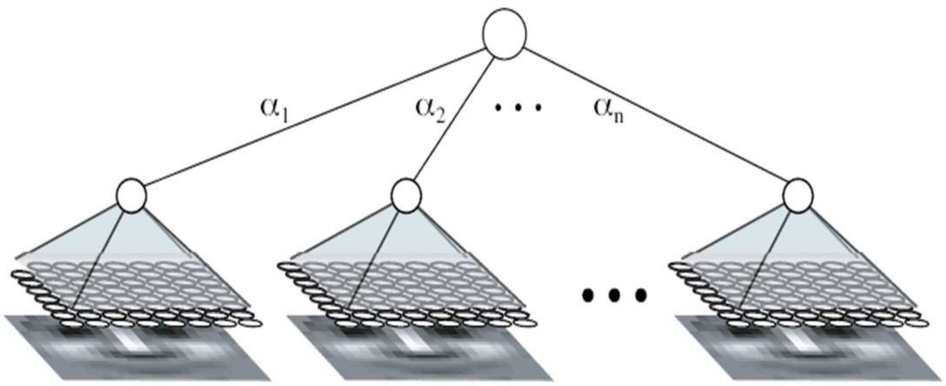


Divide and Conquer



Combiners

- How to combine the outputs of classifiers?
- Averaging
- Voting
  - Majority Voting
    - Random Forest
  - Weighted Majority Voting
    - AdaBoost
- Learning Combiner
  - General Combiner
    - Stacking
  - Piecewise Combiner
    - RegionBoost
- No Free Lunch



Diversity

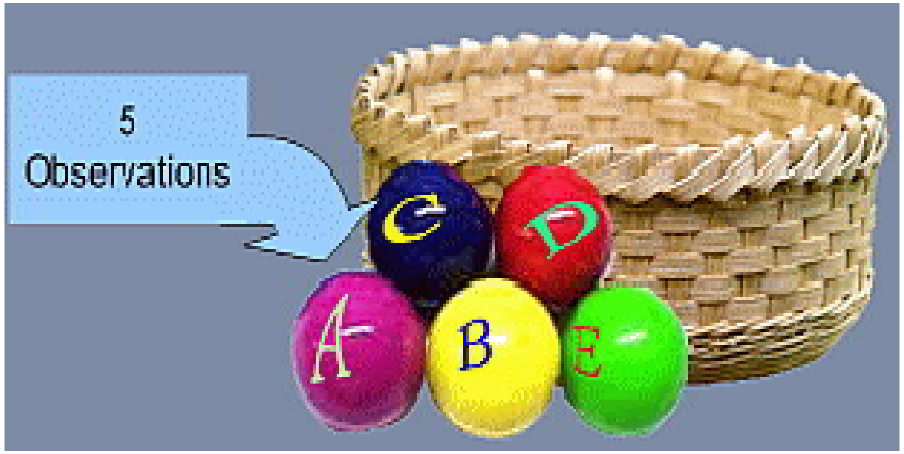
- The key to the success of ensemble learning
  - Need to correct the errors made by other classifiers.
  - Does not work if all models are identical.

- Different Learning Algorithms
  - DT, SVM, NN, KNN …

- Different Training Processes
  - Different Parameters
  - Different Training Sets
  - Different Feature Sets

- Weak Learners
  - Easy to create different decision boundaries.
  - Stumps …

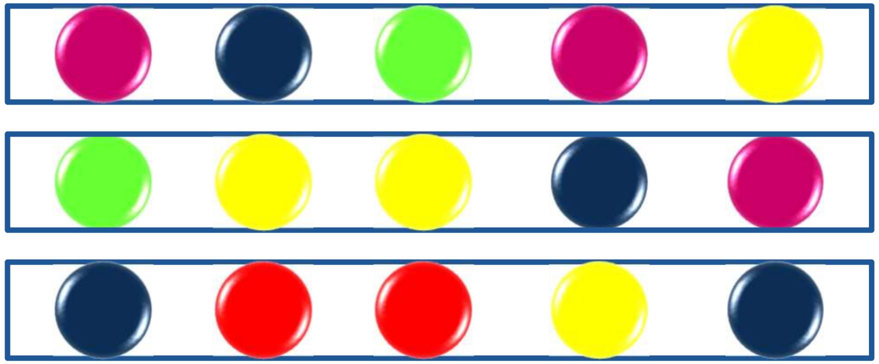
Bootstrap Samples



Sample 1

Sample 2

Sample 3



Bagging (Bootstrap Aggregating)

Algorithm: Bagging

Input:

- Training data S with correct labels $\omega_i \Omega = \{\omega_1,\ldots,\omega_C\}$ representing C classes
- Weak learning algorithm **WeakLearn,**
- Integer T specifying number of iterations.
- Percent (or fraction) F to create bootstrapped training data

Do $t=1, \ldots, T$

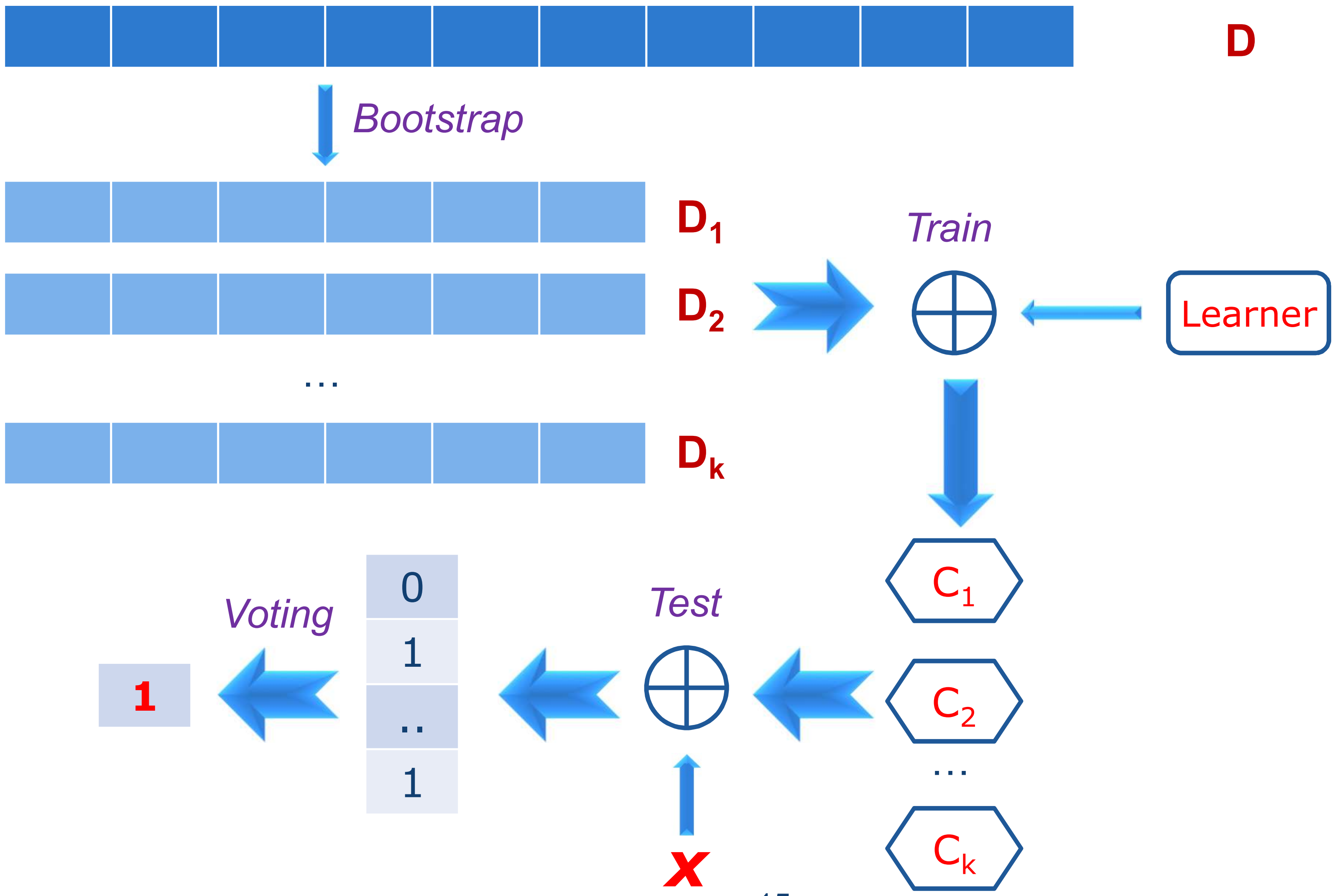
1. Take a bootstrapped replica S_t by randomly drawing F percent of S.
2. Call **WeakLearn** with S_t and receive the hypothesis (classifier) h_t.
3. Add h_t to the ensemble, $\mathcal{E}$.

End

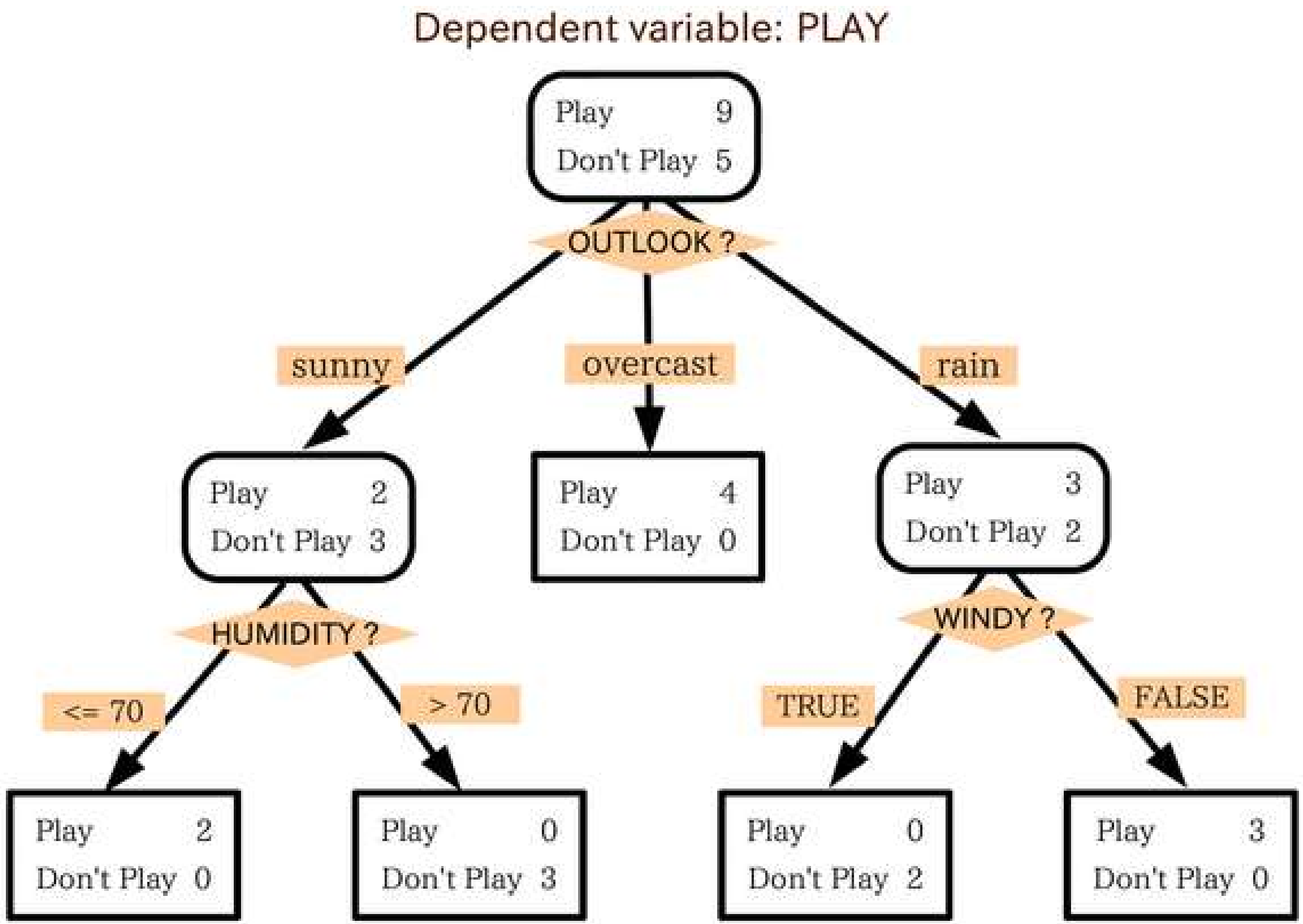
Test: Simple Majority Voting – Given unlabeled instance **x**

1. Evaluate the ensemble $\mathcal{E} = \{h_1,\ldots,h_T\}$ on **x**.
2. Let $\mathbf{v}_{t,j} = \begin{cases} 1, & \text{if } \boldsymbol{h}_t \text{ picks class } \boldsymbol{\omega}_j \\ 0, & \text{otherwise} \end{cases}$ be the vote given to class ω_j by classifier h_t.
3. Obtain total vote received by each class , $V_j = \sum_{t=1}^{T} \mathbf{v}_{t,j}$ $j = 1,\ldots,C$.
4. Choose the class that receives the highest total vote as the final classification.

Bagging



A Decision Tree



Tree vs. Forest

Random Forests

- Developed by Prof. Leo Breiman
  - Inventor of CART
  - www.stat.berkeley.edu/users/breiman/
  - Breiman, L.: Random Forests. *Machine Learning* 45(1), 5–32, 2001
- Bootstrap Aggregation (Bagging)
  - Resample with Replacement
  - Use around two third of the original data.

$$\boxed{1 - \lim_{n\to\infty}\left(1 - \frac{1}{n}\right)^n}$$

- A Collection of CART-like Trees
  - Binary Partition
  - No Pruning
  - Inherent Randomness
- Majority Voting

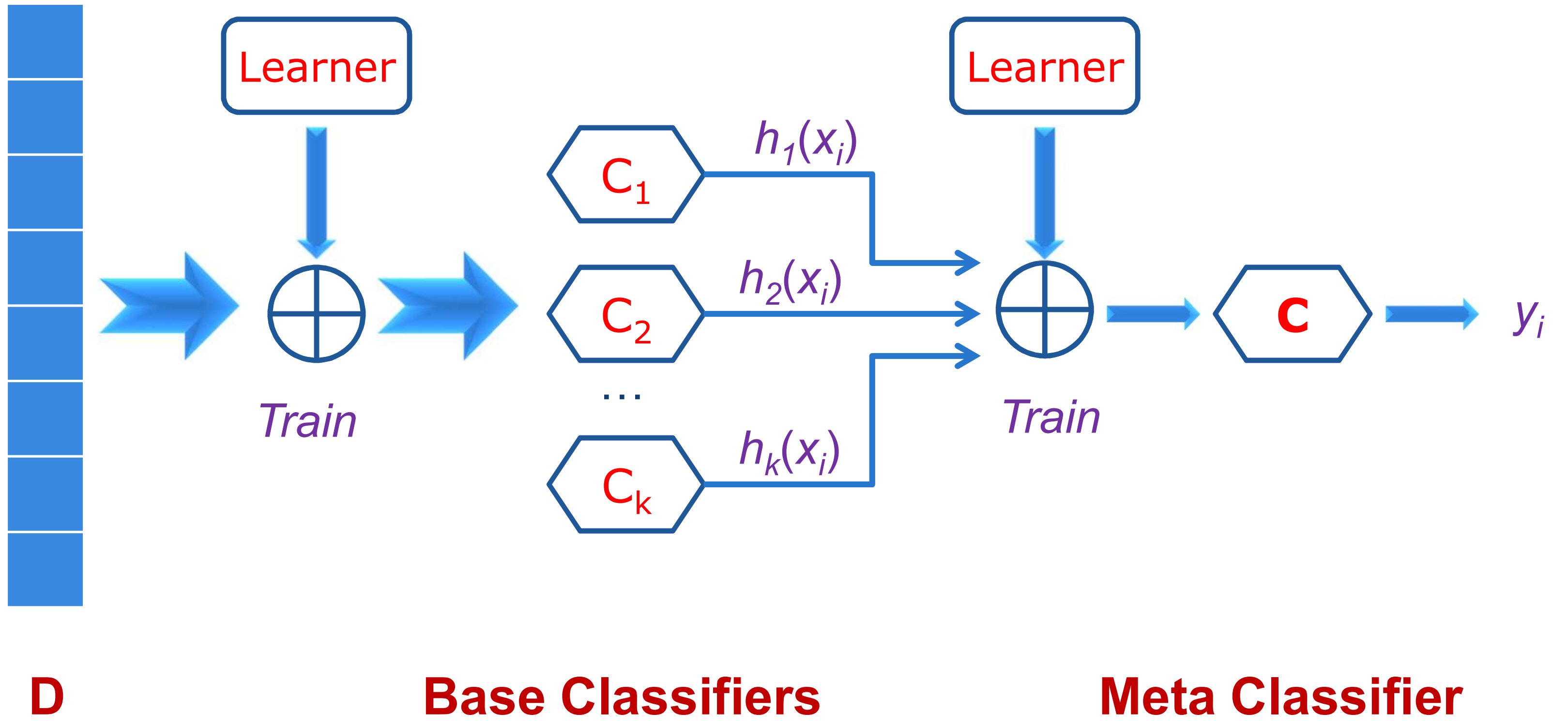
RF Main Features

- Generates substantially different trees:
  - Use random bootstrap samples of the training data.
  - Use random subsets of variables for each node.
- Number of Variables
  - Square Root (K)
  - K: total number of available variables
  - Can dramatically speed up the tree building process.
- Number of Trees
  - 500 or more
- Self-Testing
  - Around one third of the original data are left out.
  - Out of Bag (OOB)
  - Similar to Cross-Validation

RF Advantages

- All data can be used in the training process.
  - No need to leave some data for testing.
  - No need to do conventional cross-validation.
  - Data in OOB are used to evaluate the current tree.
- Performance of the entire RF
  - Each data point is tested over a subset of trees.
  - Depends on whether it is in the OOB.
- High levels of predictive accuracy
  - Only a few parameters to experiment with.
  - Suitable for both classification and regression.
- Resistant to overtraining (overfitting).
- No need for prior feature selection.

Stacking



$\{(x_1, y_1) \ldots (x_n, y_n)\}$

$\{(h_1(x_i), h_2(x_i), \ldots, h_k(x_i), y_i)\}$

Stacking

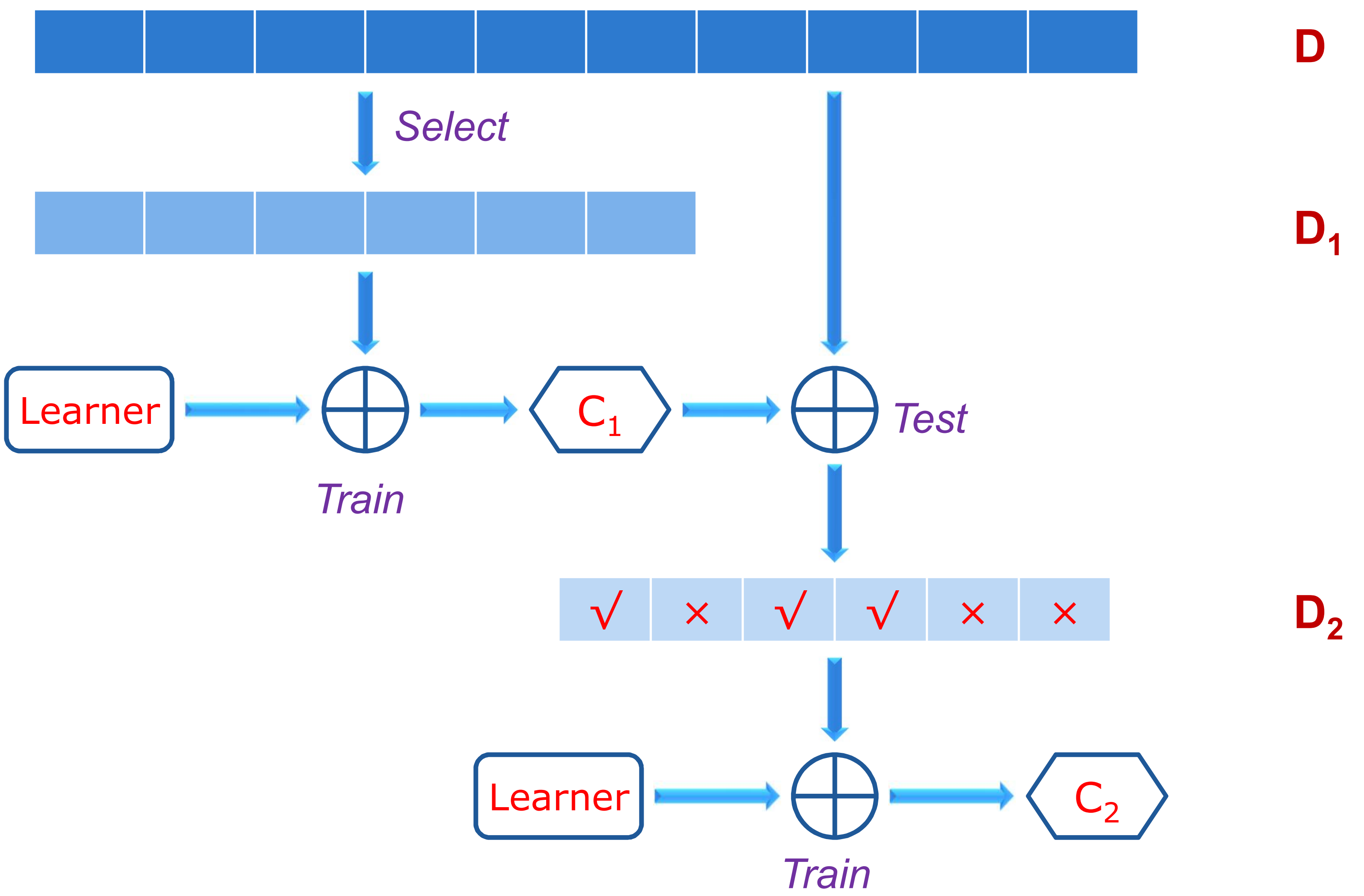
Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
First-level learning algorithms $\mathcal{L}_1, \cdots, \mathcal{L}_T$;
Second-level learning algorithm $\mathcal{L}$.

Process:

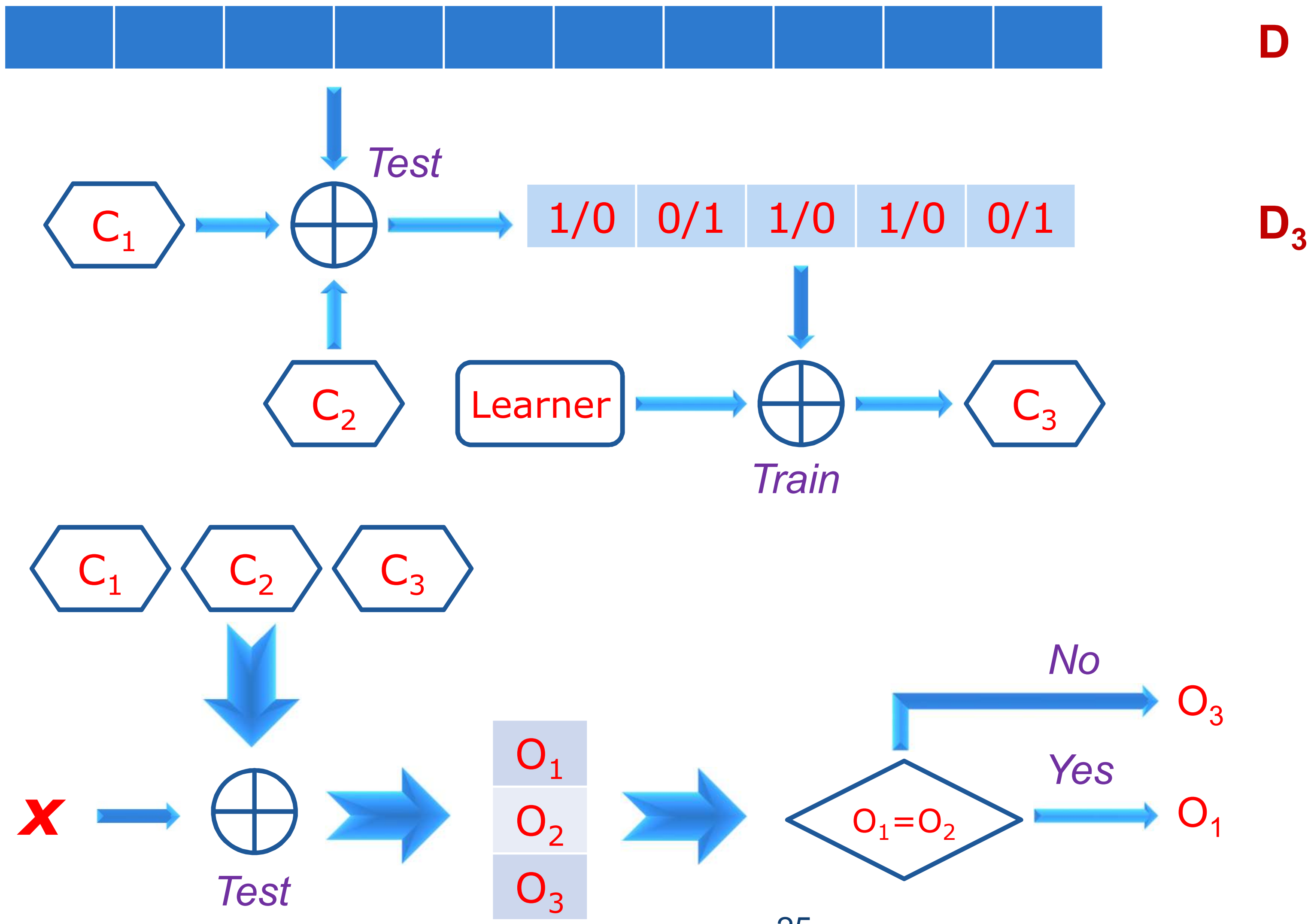
for $t = 1, \cdots, T$:
  $h_t = \mathcal{L}_t(\mathcal{D})$ % Train a first-level individual learner h_t by applying the first-level
end;   % learning algorithm $\mathcal{L}_t$ to the original data set $\mathcal{D}$
$\mathcal{D}' = \emptyset$;   % Generate a new data set
for $i = 1, \cdots, m$:
  for $t = 1, \cdots, T$:
    $z_{it} = h_t(\boldsymbol{x}_i)$ % Use h_t to classify the training example $\boldsymbol{x}_i$
  end;
  $\mathcal{D}' = \mathcal{D}' \cup \{((z_{i1}, z_{i2}, \cdots, z_{iT}), y_i)\}$
end;
$h' = \mathcal{L}(\mathcal{D}')$.   % Train the second-level learner h' by applying the second-level
  % learning algorithm $\mathcal{L}$ to the new data set $\mathcal{D}'$

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \cdots, h_T(\boldsymbol{x}))$

Boosting



Boosting


D
Test
C1
1/0
0/1
1/0
1/0
0/1
D3
C2
Learner
Train
C3
C1
C2
C3
No
O3
x
Test
O1
O2
O3
O1=O2
Yes
O1

Boosting

Input: Instance distribution $\mathcal{D}$;
Base learning algorithm $\mathcal{L}$;
Number of learning rounds T.

Process:

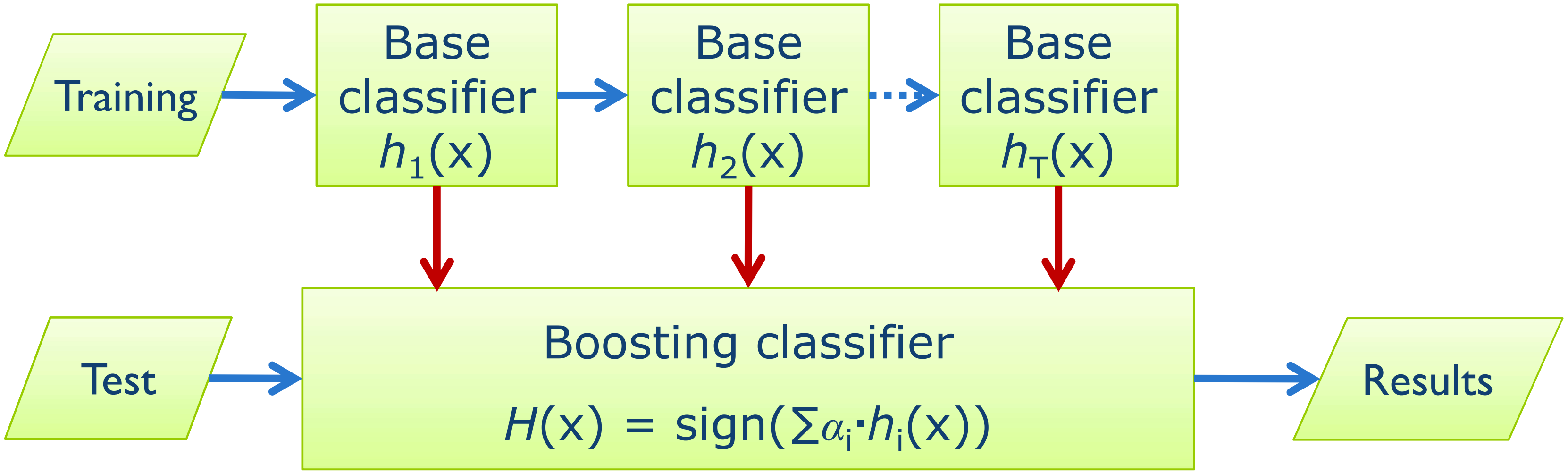
1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \cdots, T$:
3. $h_t = \mathcal{L}(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $\epsilon_t = \Pr_{\boldsymbol{x} \sim D_t, y} \boldsymbol{I}[h_t(\boldsymbol{x}) \neq y]$; % Measure the error of h_t
5. $\mathcal{D}_{t+1} = Adjust\_Distribution(\mathcal{D}_t, \epsilon_t)$
6. **end**

Output: $H(\boldsymbol{x}) = Combine\_Outputs(\{h_t(\boldsymbol{x})\})$

Boosting

- Bagging aims at reducing variance, not bias.
- In Boosting, classifiers are generated sequentially.
- Focuses on most informative data points.
- Training samples are weighted.
- Outputs are combined via weighted voting.
- Can create arbitrarily strong classifiers.
- The base learners can be arbitrarily weak.
- As long as they are better than random guess!

Boosting



Training
Base classifier $h_1(x)$
Base classifier $h_2(x)$
Base classifier $h_T(x)$
Test
Boosting classifier
$H(x) = \text{sign}(\sum \alpha_i \cdot h_i(x))$
Results

AdaBoost

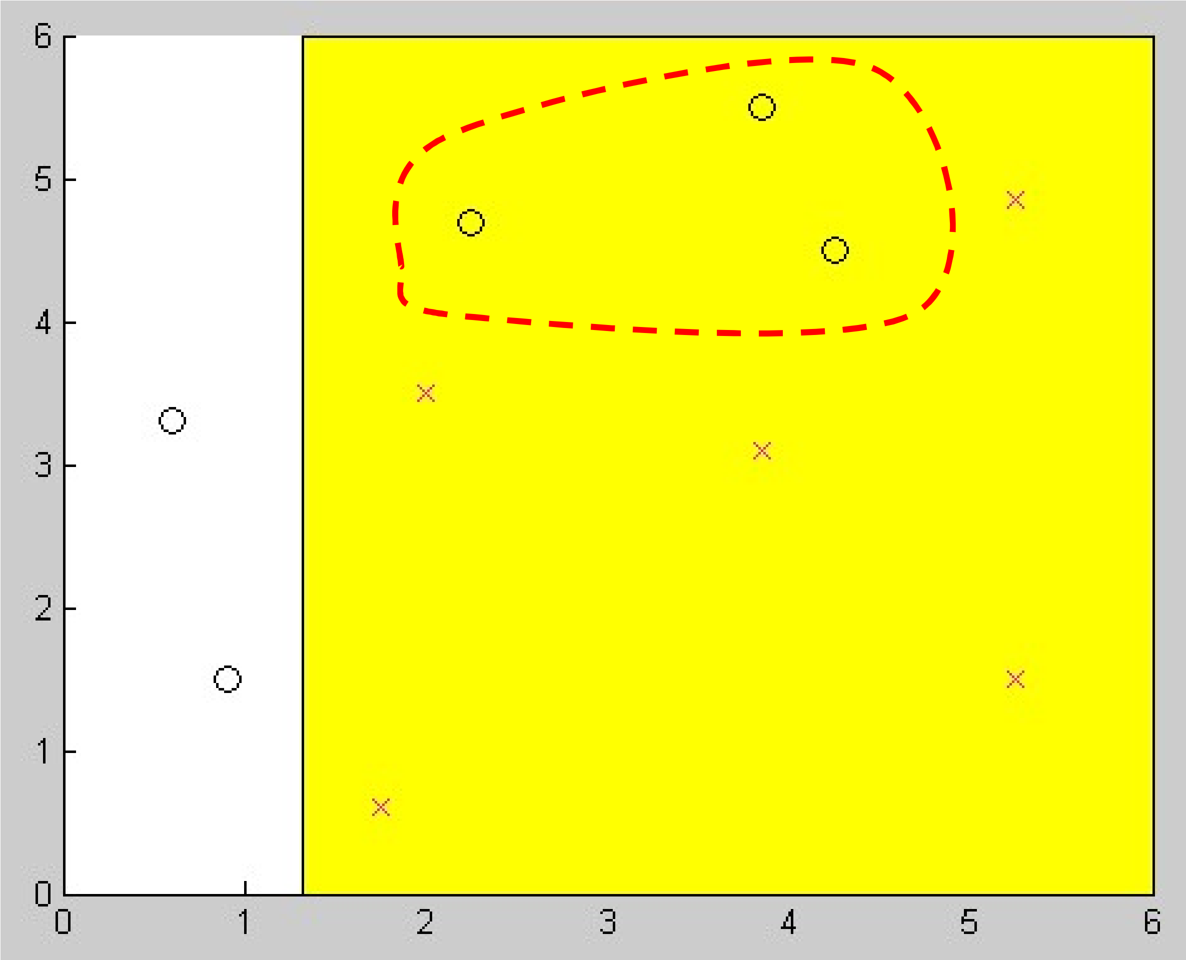
Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:

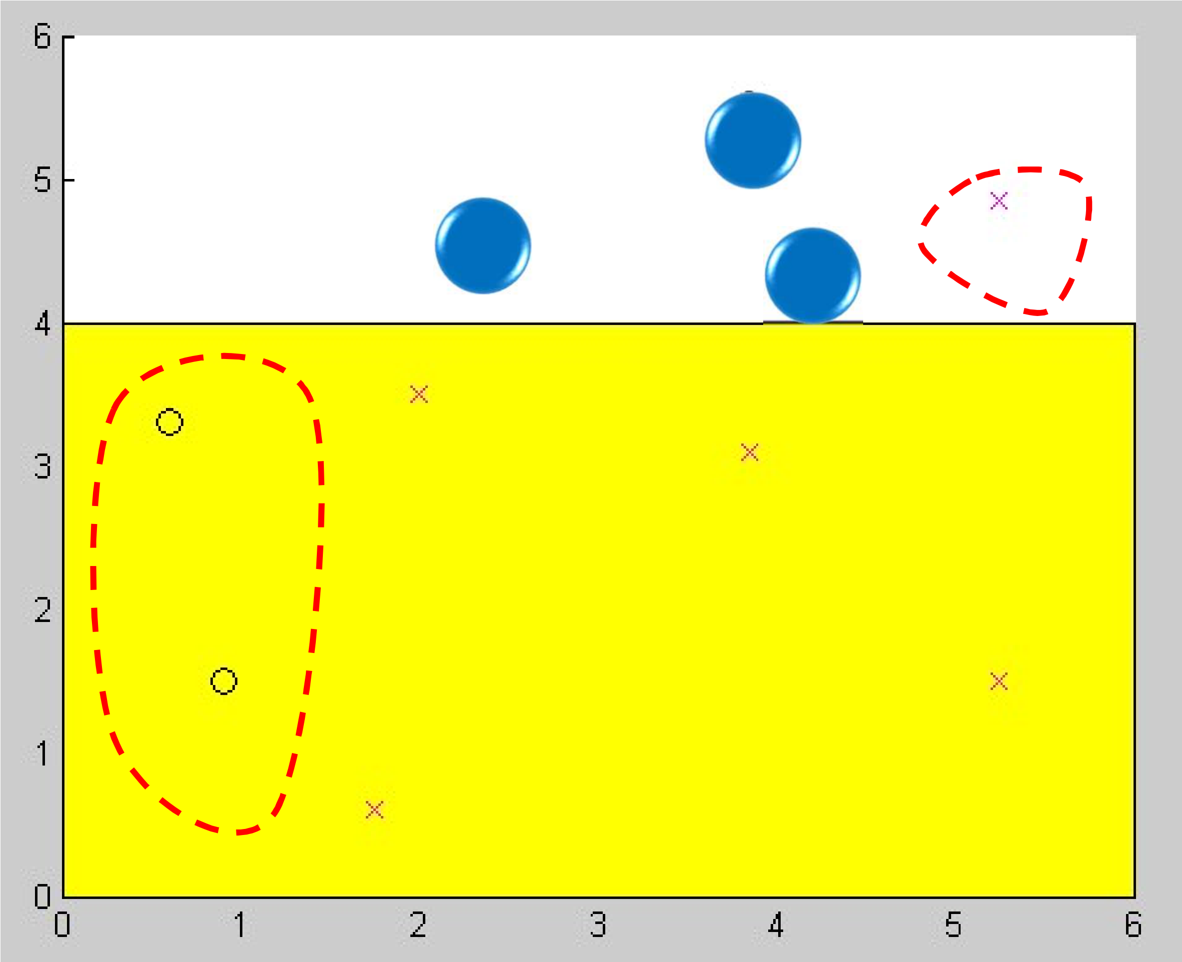
1. $\mathcal{D}_1(i) = 1/m$. % Initialize the weight distribution
2. **for** $t = 1, \cdots, T$:
3. $h_t = \mathfrak{L}(D, \mathcal{D}_t)$; % Train a learner h_t from D using distribution $\mathcal{D}_t$
4. $\epsilon_t = \Pr_{\boldsymbol{x} \sim \mathcal{D}_t, y} \boldsymbol{I}[h_t(\boldsymbol{x}) \neq y]$; % Measure the error of h_t
5. **if** $\epsilon_t > 0.5$ **then break**
6. $\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$; % Determine the weight of h_t
7. $\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$
$= \frac{\mathcal{D}_t(i) \exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$ % Update the distribution, where
% Z_t is a normalization factor which
% enables $\mathcal{D}_{t+1}$ to be a distribution
8. **end**

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

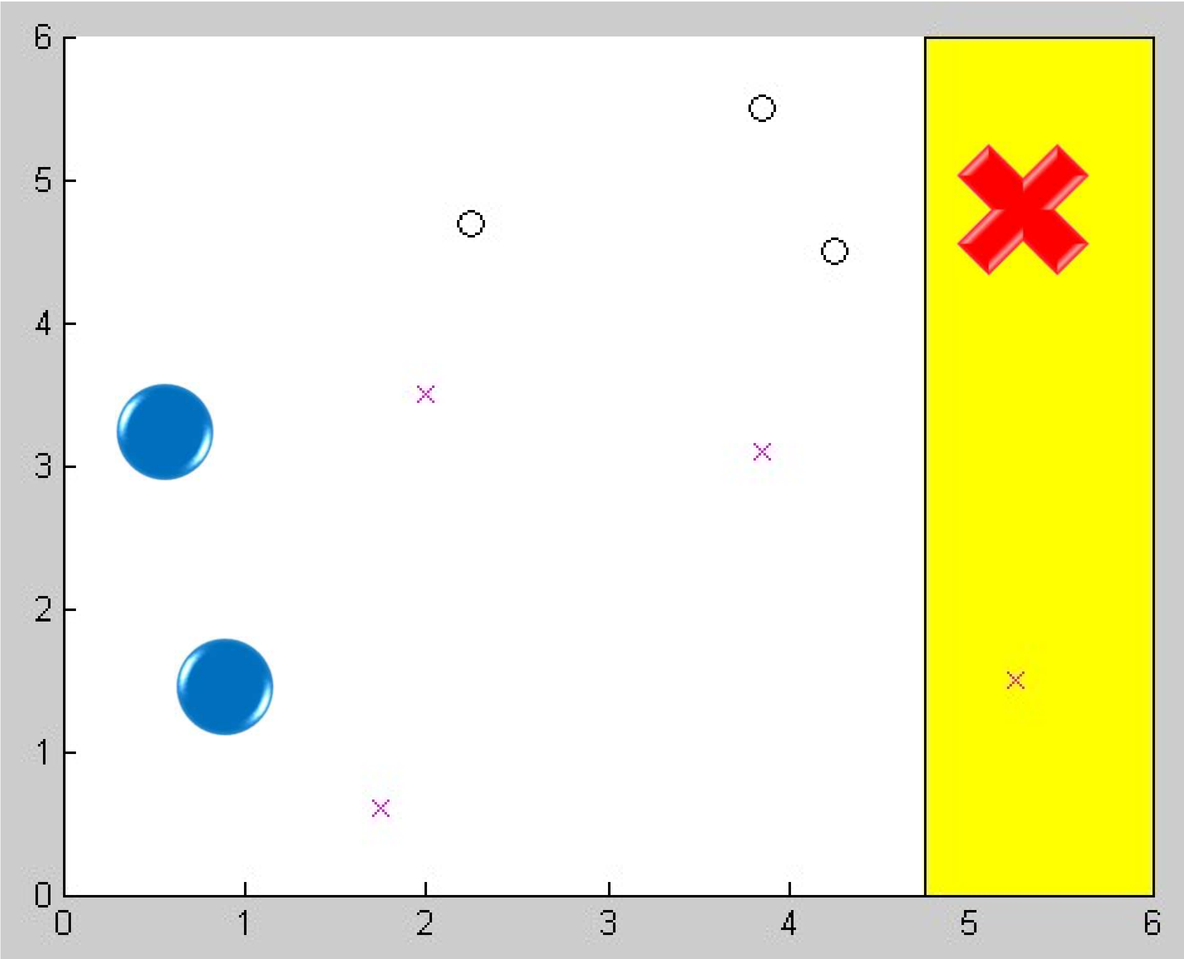
Demo: Classifier 1



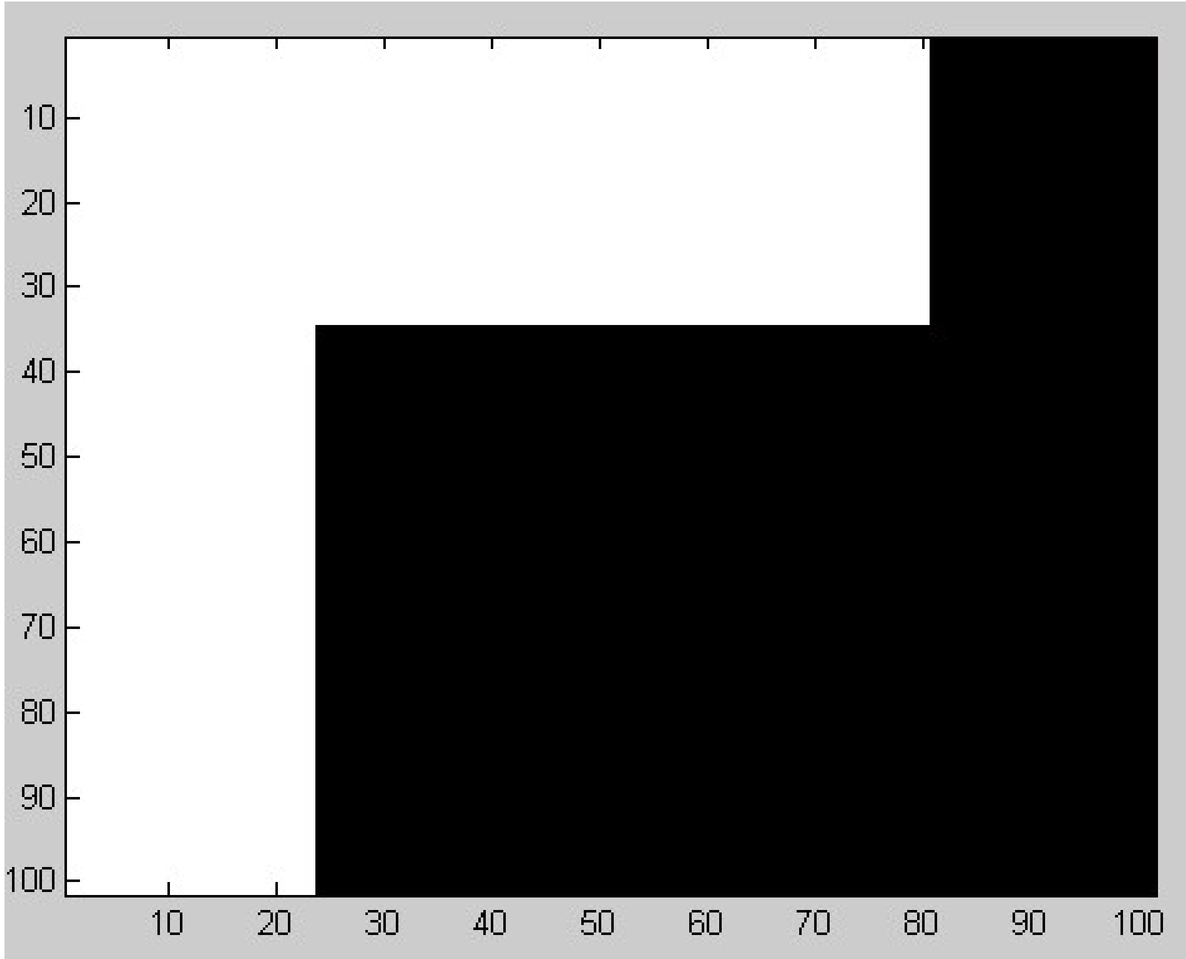
Demo: Classifier 2



Demo: Classifier 3



Demo: Combined Classifier



The Choice of α

Theorem 1: *Error is minimized by minimizing* Z_t

Proof:

$$D_{T+1}(i) = \frac{1}{m} \cdot \frac{e^{-y_i \alpha_1 h_1(x_i)}}{Z_1} \cdot \ldots \cdot \frac{e^{-y_i \alpha_T h_T(x_i)}}{Z_T}$$
$$= \frac{e^{\sum_t -y_i \alpha_t h_t(x_i)}}{m \prod_t Z_t} = \frac{e^{-y_i \sum_t \alpha_t h_t(x_i)}}{m \prod_t Z_t}$$
$$= \frac{e^{-y_i f(x_i)}}{m \prod_t Z_t}$$

$$f(x_i) = \sum_t \alpha_t h_t(x_i)$$

$$H(x_i) \neq y_i \Rightarrow y_i f(x_i) \leq 0 \Rightarrow e^{-y_i f(x_i)} \geq 1$$

$$[\![H(x_i) \neq y_i]\!] \leq e^{-y_i f(x_i)}$$

$$\frac{1}{m}\sum_i [\![H(x_i) \neq y_i]\!] \leq \frac{1}{m}\sum_i e^{-y_i f(x_i)}$$

Model Error

The Choice of α

$$D_{T+1}(i) = \frac{e^{-y_i\ f(x_i)}}{m \prod_t Z_t}$$

Combining these results,

$$\frac{1}{m}\sum_i [\![H(x_i) \neq y_i]\!] \leq \frac{1}{m}\sum_i e^{-y_i f(x_i)}$$

$$= \sum_i \left(\prod_t Z_t\right) D_{T+1}(i)$$

$$= \prod_t Z_t \quad \text{(since } D_{T+1} \text{ sums to 1).}$$

Thus, we can see that minimizing Z_t will minimize this error bound.

$$\min_{\alpha} Z_t \Rightarrow \min \prod_t Z_t$$

The Choice of α

$$y, h(x) \in \{-1, +1\} \qquad Z = \sum_i D_i e^{-\alpha y_i h(x_i)}$$

$$e^{-\alpha y_i h(x_i)} = e^{-\alpha} P(y_i = h(x_i)) + e^{\alpha} P(y_i \neq h(x_i))$$

$$\frac{\partial Z}{\partial \alpha} = -e^{-\alpha} \sum_i D_i P\big(y_i = h(x_i)\big) + e^{\alpha} \sum_i D_i P\big(y_i \neq h(x_i)\big) = 0$$

$$\alpha = \frac{1}{2} \ln \frac{\sum_i D_i \big(1 - P\big(y_i \neq h(x_i)\big)\big)}{\sum_i D_i P\big(y_i \neq h(x_i)\big)} = \boxed{\frac{1}{2} \ln \frac{1-\varepsilon}{\varepsilon}}$$

Error Bounds

$$r = \sum_i D_i y_i h(x_i) \longrightarrow \varepsilon = \frac{1-r}{2} \longrightarrow \alpha = \frac{1}{2}\ln\frac{1+r}{1-r}$$

$$Z = \sum_i D_i e^{-\alpha y_i h(x_i)} = \sum_i D_i e^{-\left(\frac{1}{2}\ln\frac{1+r}{1-r}\right) y_i h(x_i)} = \sum_i D_i \left(\sqrt{\frac{1+r}{1-r}}\right)^{-y_i h(x_i)}$$

$$= \sum_i D_i \left(\sqrt{\frac{1+r}{1-r}} P\big(y_i \neq h(x_i)\big) + \sqrt{\frac{1-r}{1+r}} P\big(y_i = h(x_i)\big)\right)$$

$$= \sqrt{\frac{1+r}{1-r}}\varepsilon + \sqrt{\frac{1-r}{1+r}}(1-\varepsilon) = \frac{1}{1-r}\sqrt{1-r^2}\frac{1-r}{2} + \frac{1}{1+r}\sqrt{1-r^2}\frac{1+r}{2}$$

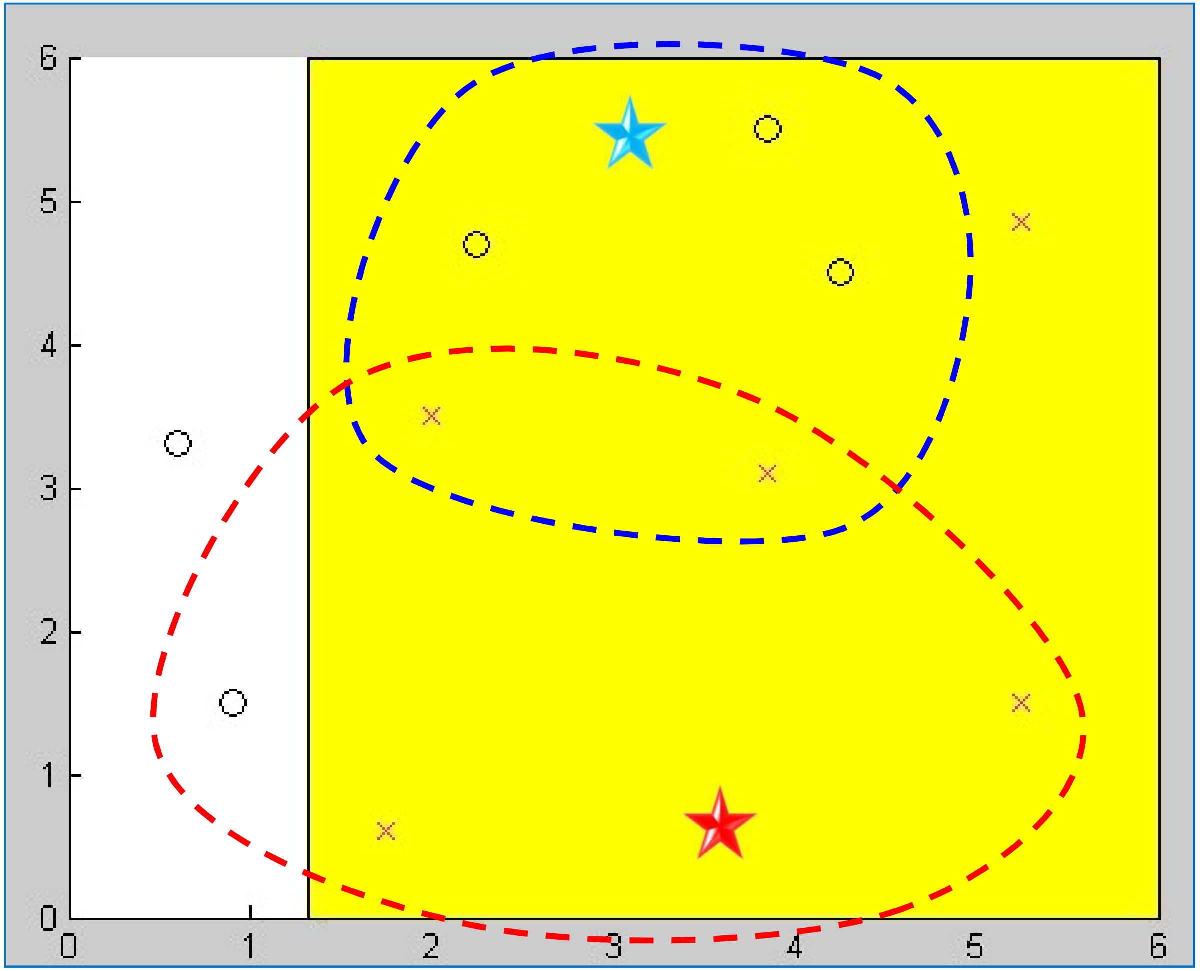
$$= \sqrt{1-r^2}$$

$$\frac{1}{m}[\![H(x_i) \neq y_i]\!] \leq \prod_t Z_t = \prod_t \sqrt{1-{r_t}^2}$$

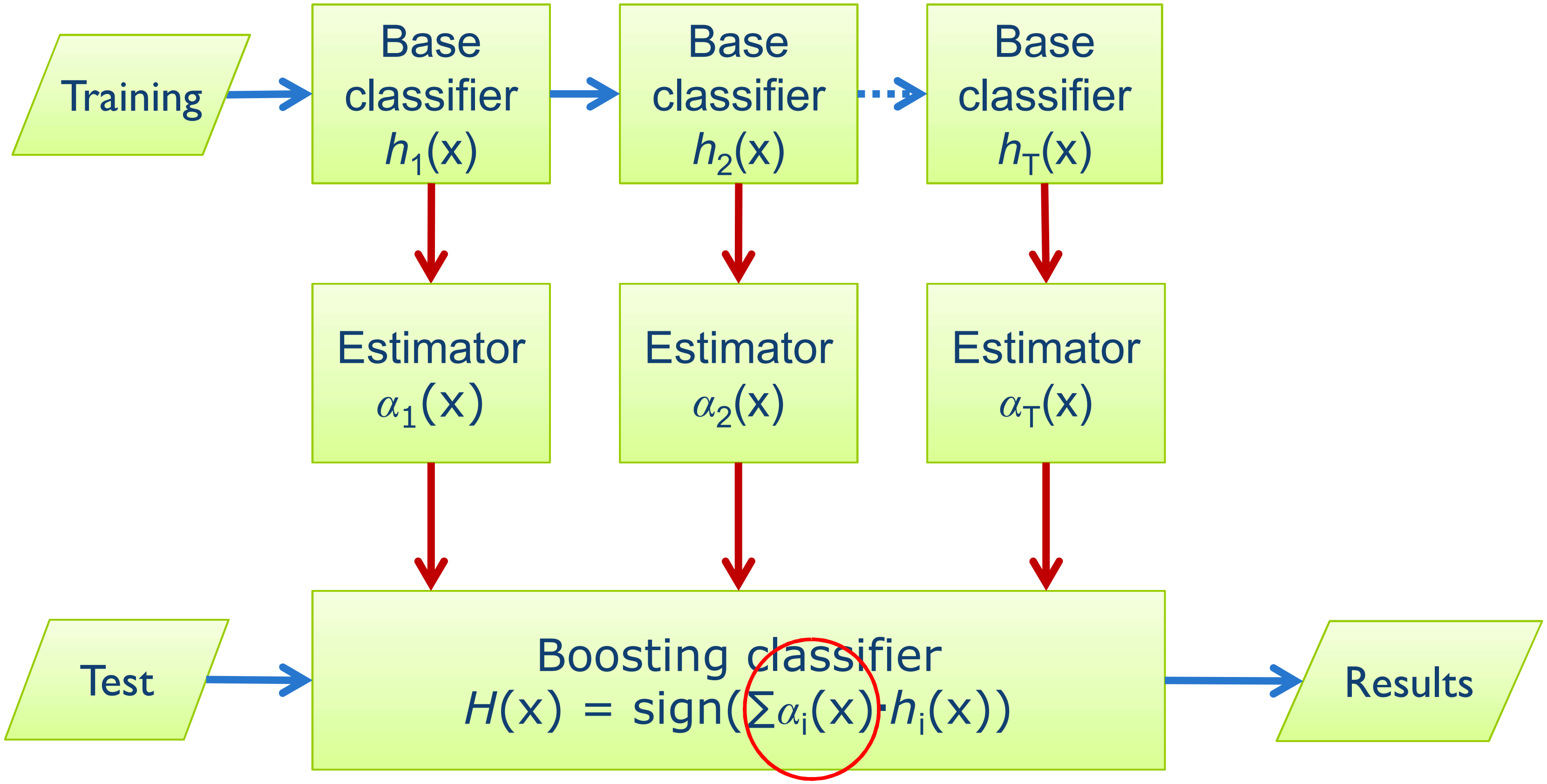
Summary of AdaBoost

- Advantages
  - Simple and easy to implement
  - Almost no parameters to tune
  - Proven upper bounds on training set
  - Immune to overfitting
- Disadvantages
  - Suboptimal α values
  - Steepest descent
  - Sensitive to noise
- Future Work
  - Theory
  - Comprehensibility
  - New Framework

Fixed Weighting Scheme



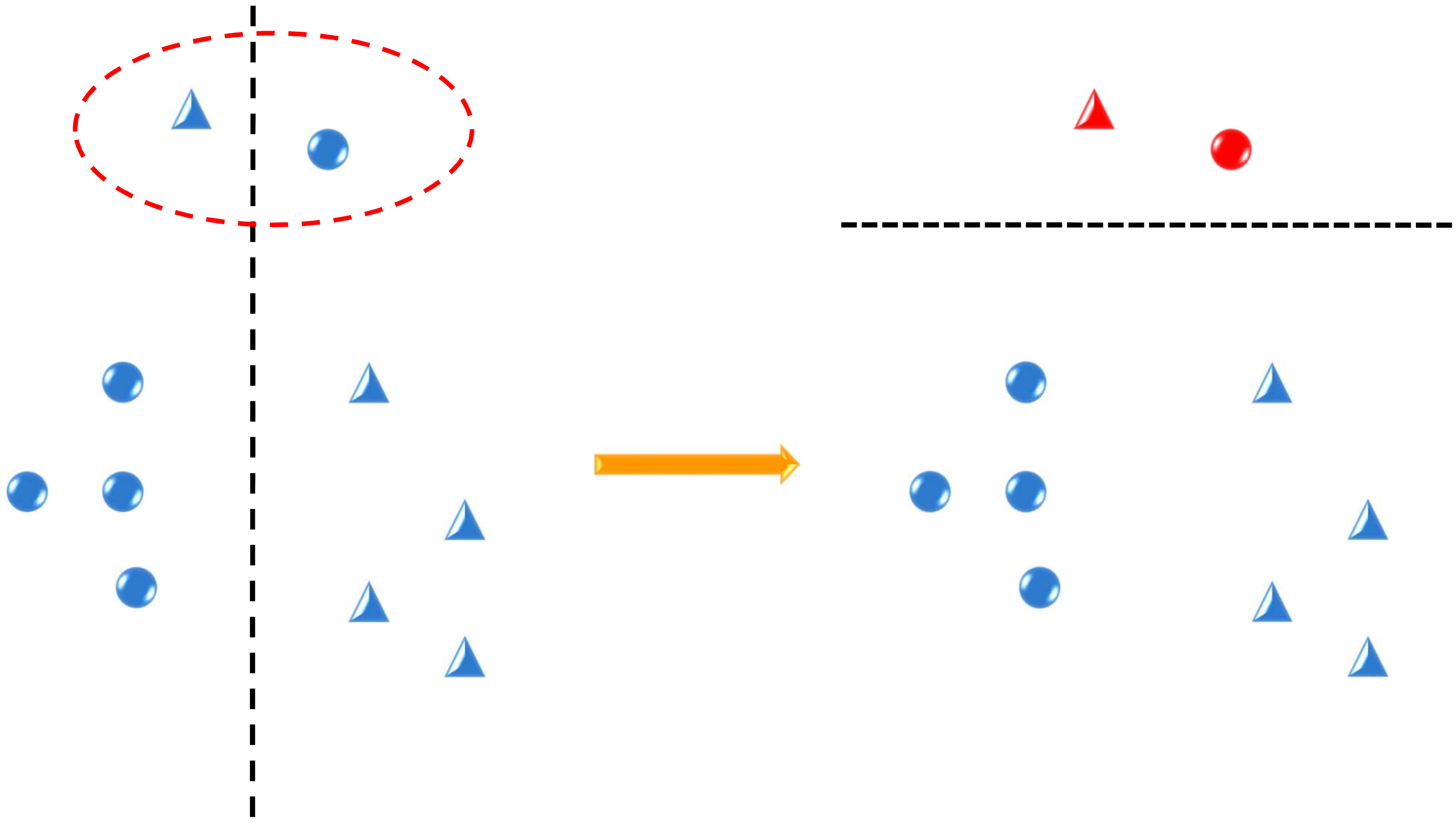
Dynamic Weighting Scheme



RegionBoost

- AdaBoost assigns fixed weights to models.
- However, different models emphasize different regions.
- The weights of models should be input-dependent.
- Given an input, only invoke appropriate models.
- Train a competency predictor for each model.
- Estimate whether the model is likely to make a right decision.
- Use this information as the weight.
- Maclin, R.: Boosting classifiers regionally. AAAI, 700-705, 1998.

RegionBoost

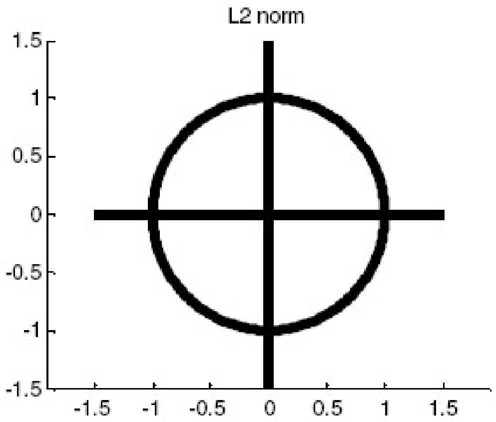


Base Classifier

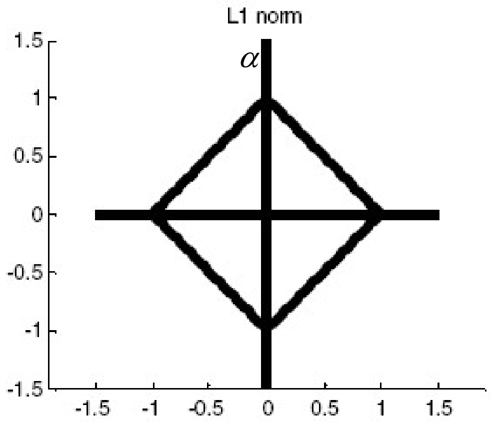
Competency Predicator

RegionBoost with KNN

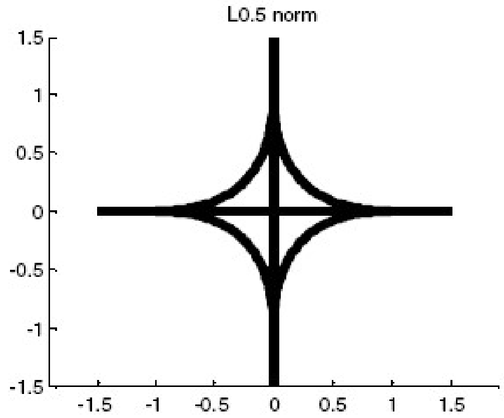
- To calculate $\alpha_j(x_i)$:
  - Find the K nearest neighbors of x_i in the training set.
  - Calculate the percentage of points correctly classified by h_j.



L_2 norm

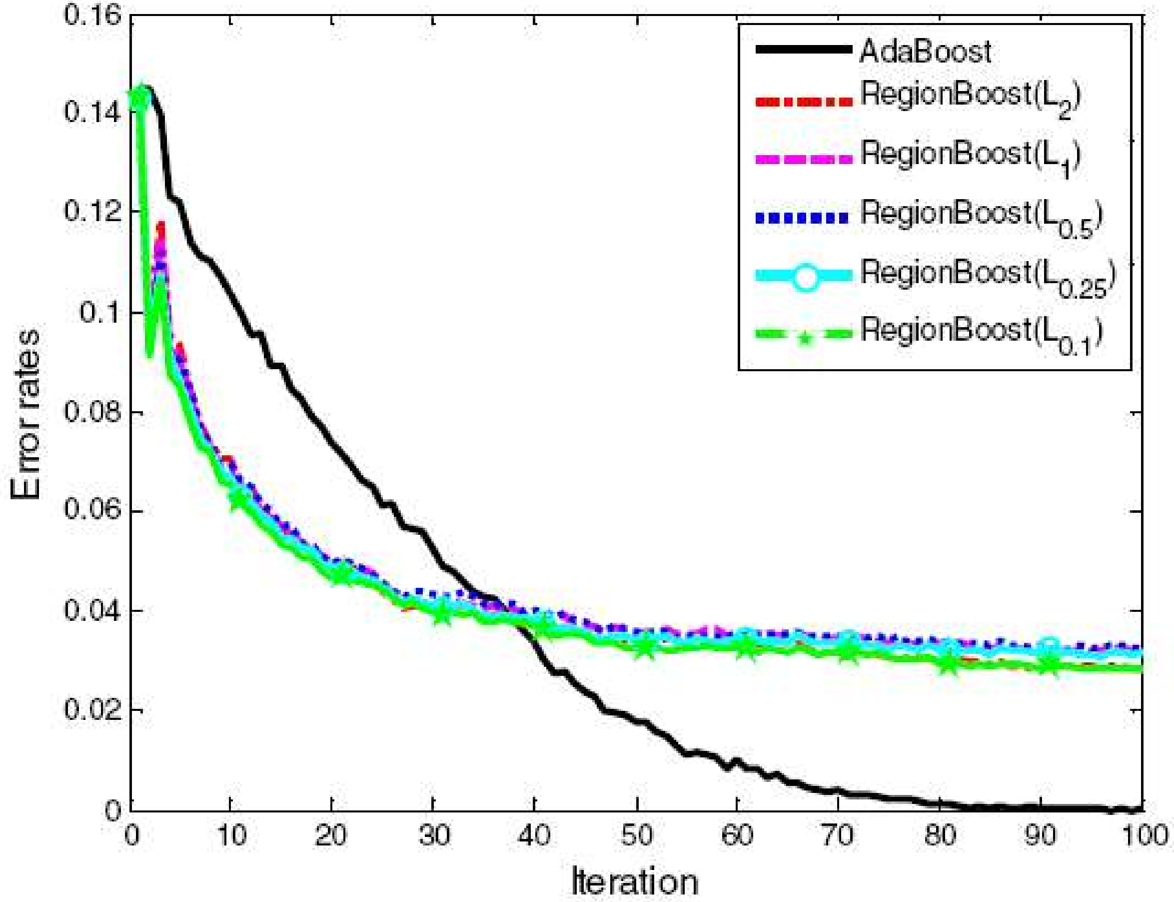


L_1 norm



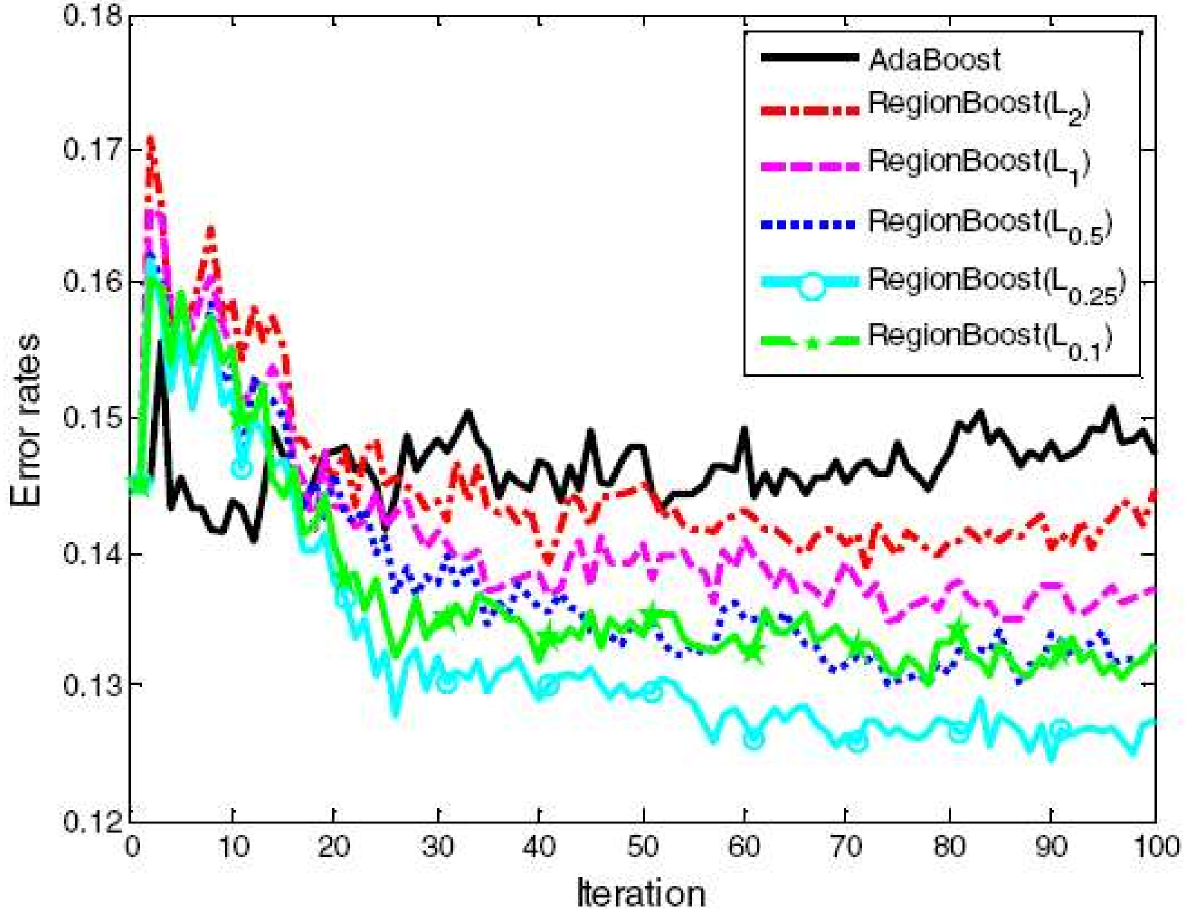
$L_{0.5}$ norm

RegionBoost Results



(a) Training error rates on Credit_Aus

RegionBoost Results



(b) Test error rates on Credit_Aus

Review

- What is ensemble learning?
- What can ensemble learning help us?
- Two major types of ensemble learning:
  - Parallel (Bagging)
  - Sequential (Boosting)
- Different ways to combine models:
  - Average
  - Majority Voting
  - Weighted Majority Voting
- Some representative algorithms:
  - Random Forests
  - AdaBoost
  - RegionBoost