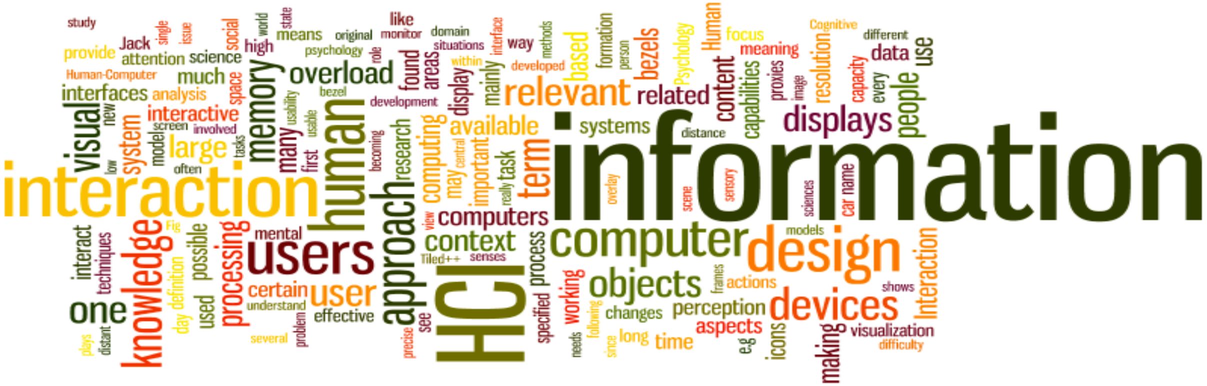
同濟大學
TONGJI UNIVERSITY

机器学习

——第9章 集成学习——



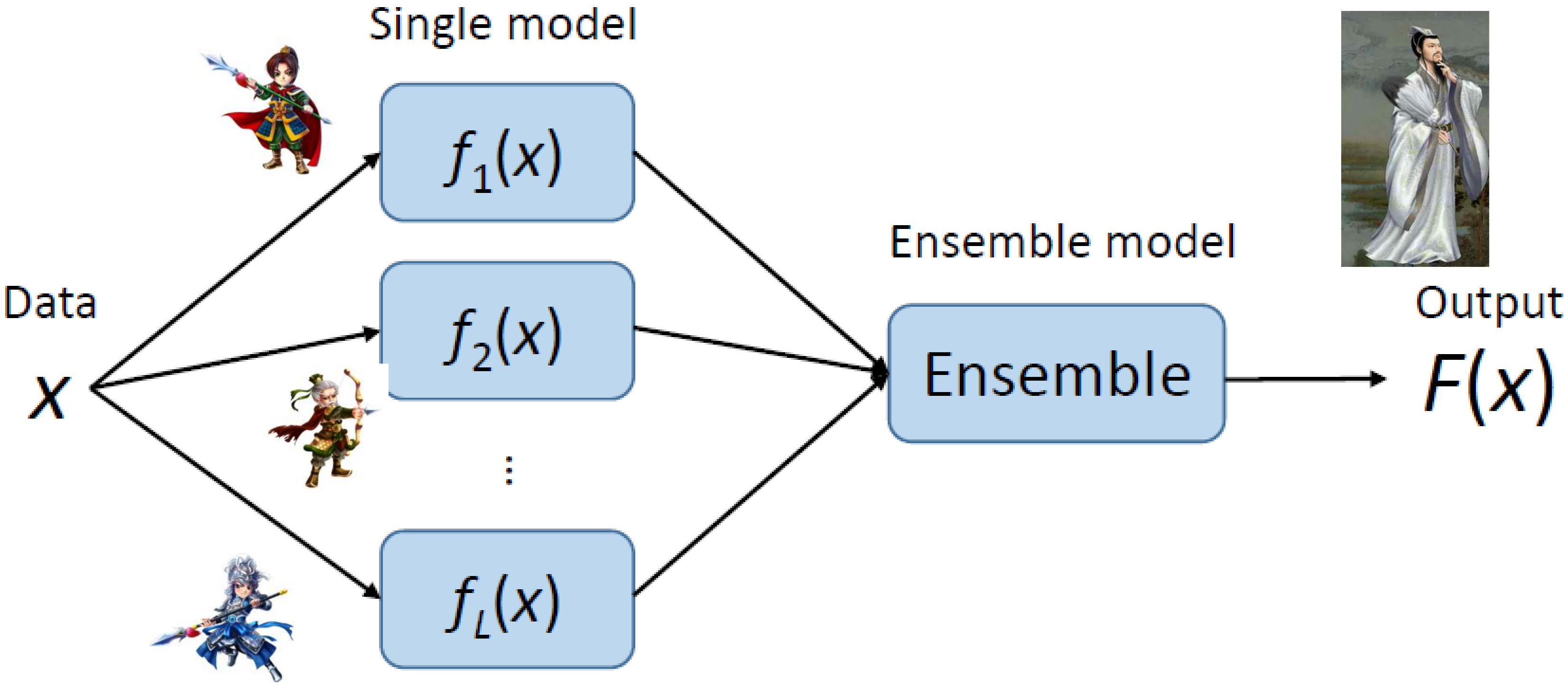
倪张凯

zkni@tongji.edu.cn

https://eezkni.github.io/

什么是集成学习 What is ensemble learning?

It is often found that improved performance can be obtained by *combining multiple models* together in some way, instead of just using a single models in isolation.



什么是集成学习 What is ensemble learning?

- Multiple Classifier Systems/ committee-based learning
- Many individual learning algorithms are available:
    - Decision Trees, Neural Networks, Support Vector Machines…
- The process by which multiple learners are strategically generated and combined in order to better solve a particular Machine Learning problem.
- Individual learner
    - homogeneous： base learner
    - heterogeneous ：component learner

集成学习示例 Example: Ensemble Learning

	Sample1	Sample2	sample3
Model 1	√	√	X
Model 2	X	√	√
Model 3	√	X	√
Ensemble	√	√	√

	Sample1	Sample2	sample3
Model 1	√	√	X
Model 2	√	√	X
Model 3	√	√	X
Ensemble	√	√	X

	Sample1	Sample2	sample3
Model 1	√	X	X
Model 2	X	√	X
Model 3	X	X	√
Ensemble	X	X	X

- construct an ensemble predictor that combines the individual decisions of model1, model2, model3

- Successful ensembles require the member each has low error rates and makes different mistakes

集成的多样性 Diversity for Ensemble

- Different type of learner
  - DT, NN, KNN, SVM, …
- Different Training Processes
  - Different Training Sets: bootstrap sampling in bagging, sequential sampling in boosting…
  - Different Parameters: number of hidden layer neurons and initial connection weights in NN, …
  - Different Feature Sets：random subspace, random forest, …
- Different output representations
  - flipping output, output smearing, ECOC …
- Hybrid

多样性测量 Diversity Measure

For a binary classification task, hi and hj's contingency table

	$h_i=+1$	$h_i=-1$
$h_j=+1$	a	c
$h_j=-1$	b	d

$$a+b+c+d=m$$

- Disagreement Measure([0,1]): $$dis_{ij}=\frac{b+c}{m}$$
- Correlation Coefficient([-1,1]): $$\rho_{ij}=\frac{ad-bc}{\sqrt{(a+b)(a+c)(c+d)(b+d)}}$$
- Q-Statistic($|Q_{ij}|\le|\rho_{ij}|$) : $$Q_{ij}=\frac{ad-bc}{ad+bc}$$
- Kappa-Statistic (usually >=0) $$\kappa=\frac{p_1-p_2}{1-p_2}$$

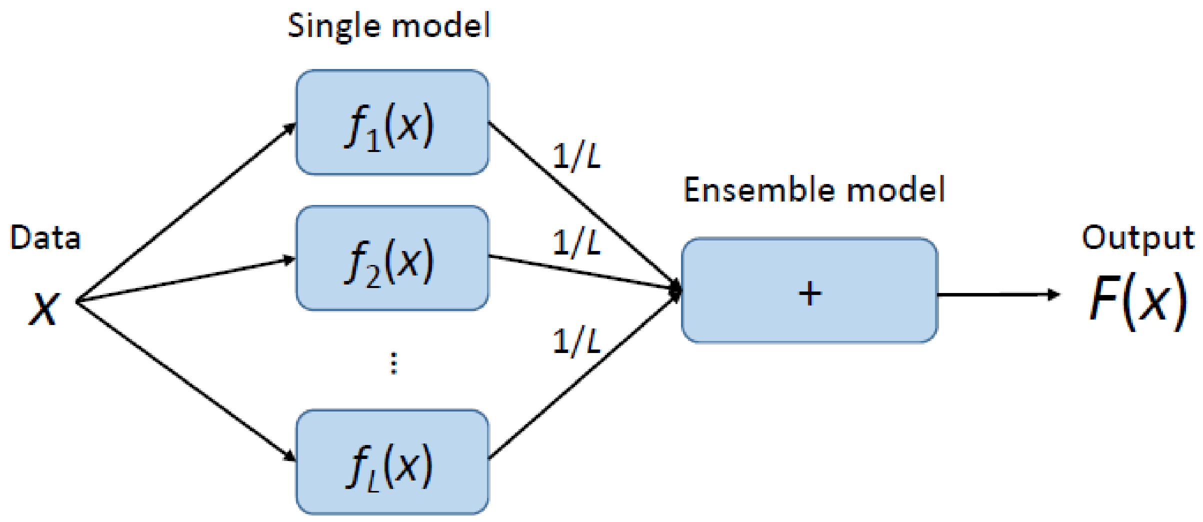
$$p_1 = \frac{a+d}{m},$$

$$p_2 = \frac{(a+b)(a+c)+(c+d)(b+d)}{m^2}$$

…

模型结合的不同方法 Different ways to combine models

Combining Predictor: Averaging

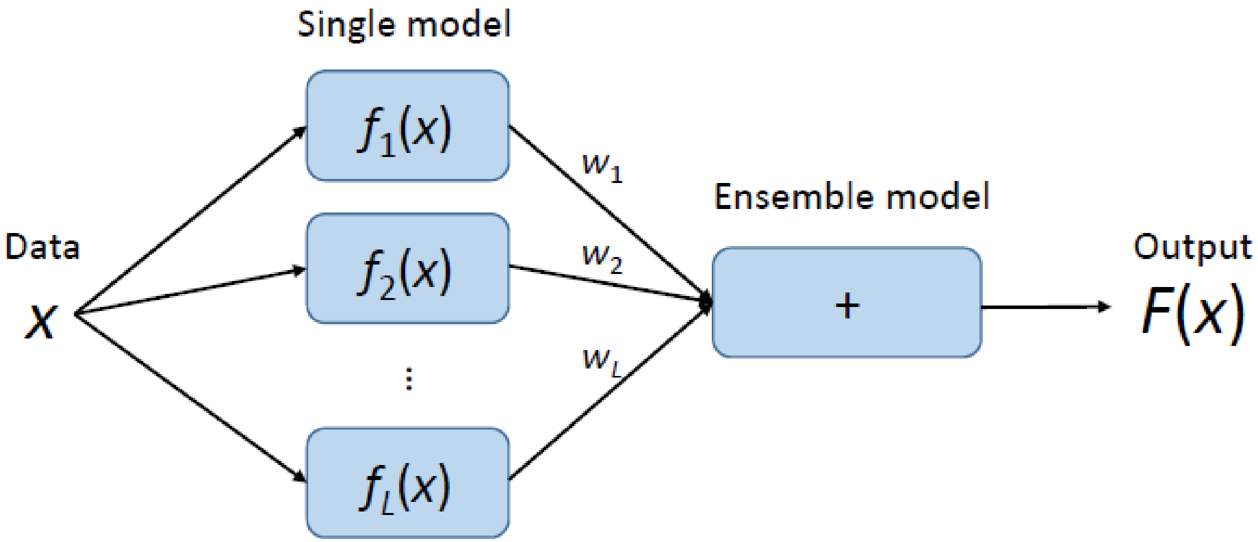


$$F(x) = \frac{1}{L}\sum_{i=1}^{L} f_i(x)$$

- Averaging for regression; voting for classification

模型结合的不同方法 Different ways to combine models

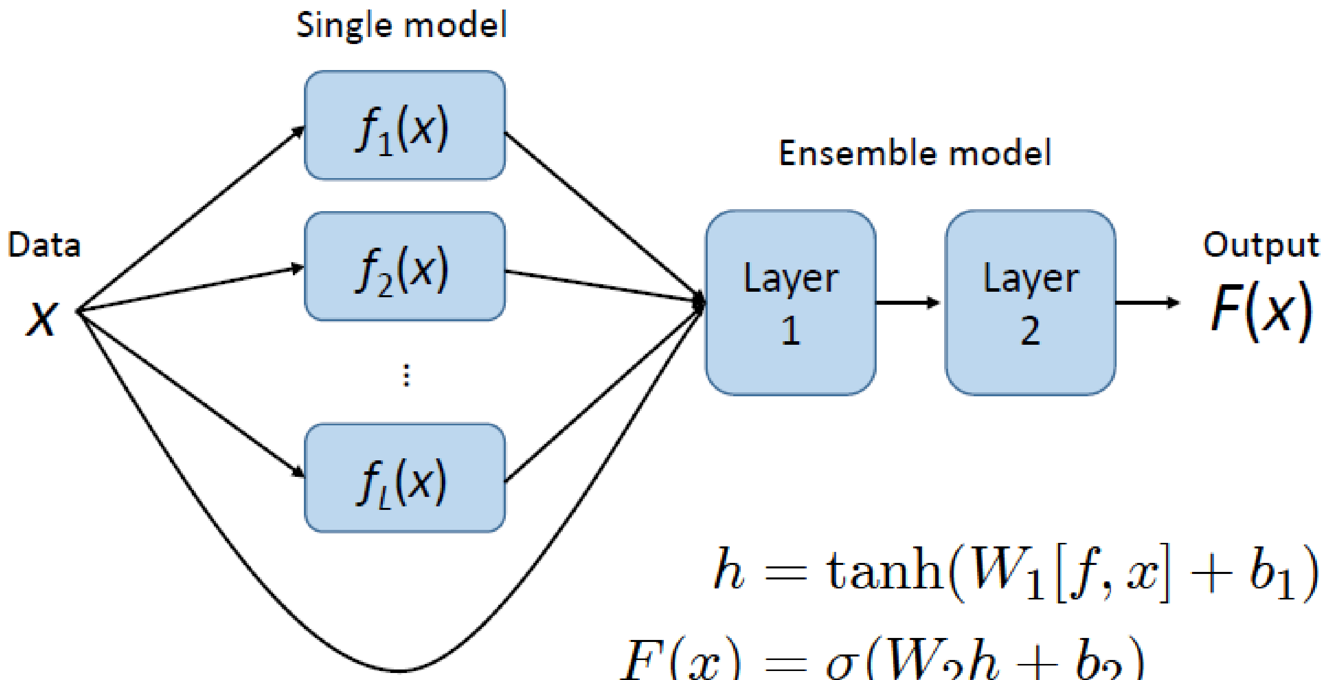
Combining Predictor: Weighted Avg



$$F(x) = \sum_{i=1}^{L} w_i f_i(x)$$

- Just like linear regression or classification
- Note: single model will not be updated when training ensemble model

Combining Predictor: Multi-Layer

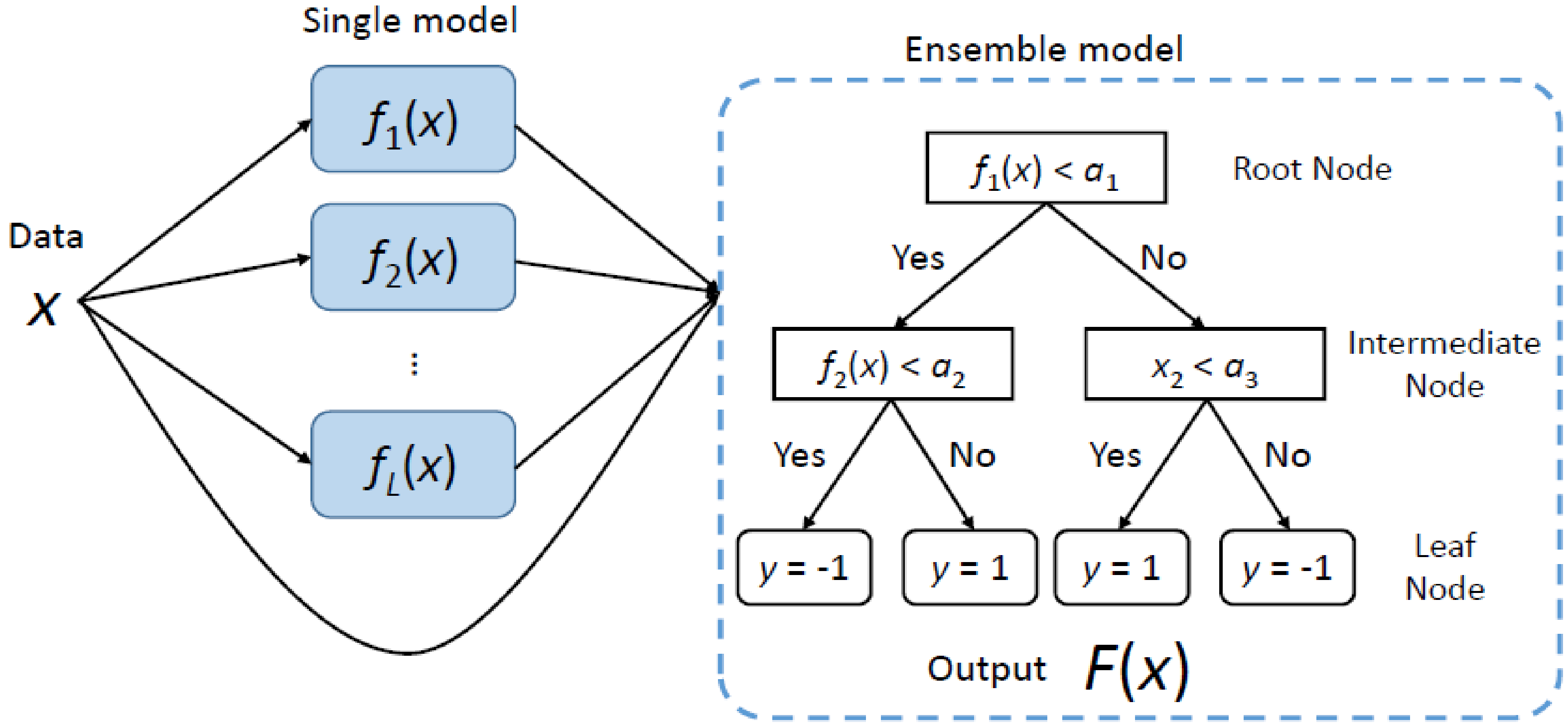


$$h = \tanh(W_1[f, x] + b_1)$$

$$F(x) = \sigma(W_2 h + b_2)$$

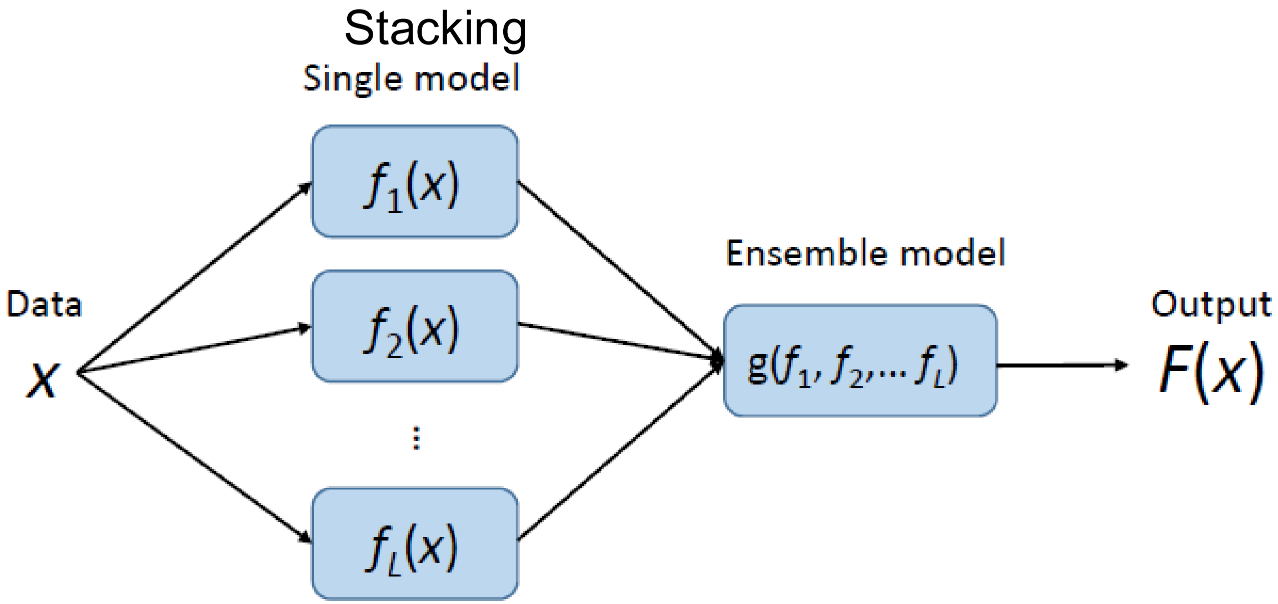
- Use neural networks as the ensemble model
- Incorporate x into the first hidden layer (as gating)

Combining Predictor: Tree Models



- Use decision trees as the ensemble model
- Splitting according to the value of f's and x

Combining Predictor: Stacking

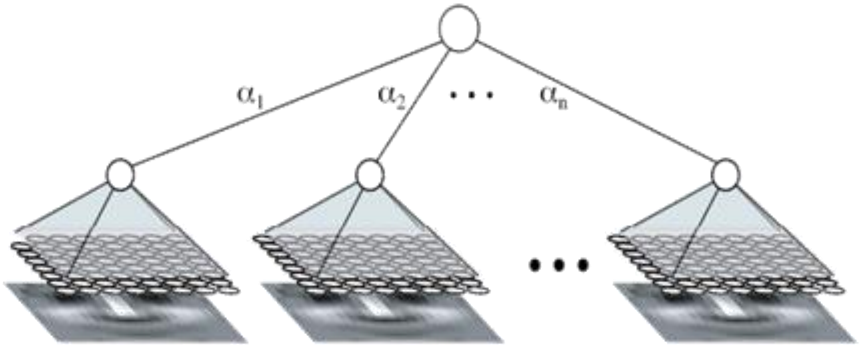


$$F(x) = g(f_1(x), f_2(x), \dots, f_L(x))$$

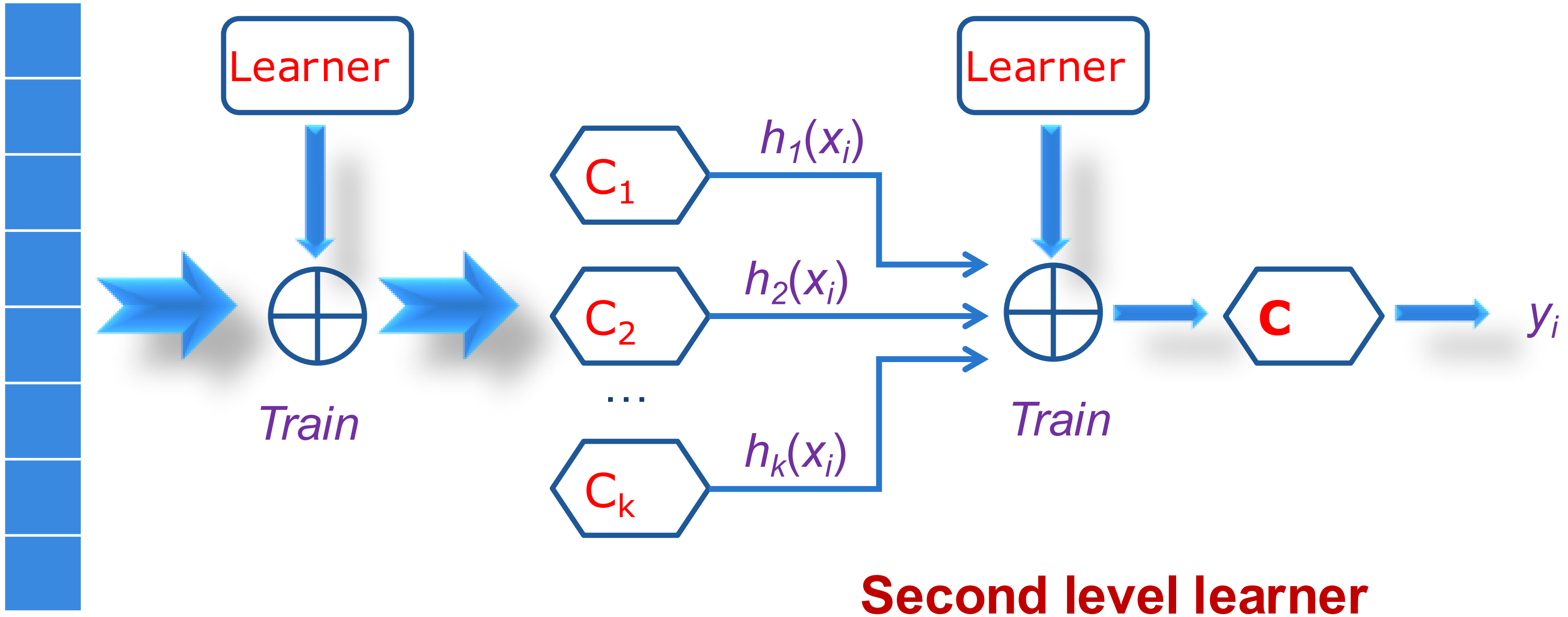
- This is the general formulation of an ensemble

模型结合的不同方法 Different ways to combine models

- Averaging
  - simple averaging
  - weighted averaging
- Voting
  - Majority Voting
    - Random Forest
  - plurality voting
  - Weighted Majority Voting
    - AdaBoost
- Learning Combiner
  - General Combiner
    - Stacking
  - Bayes Model averaging
  - Piecewise Combiner
    - RegionBoost



模型结合的学习法 Stacking



Second level learner

Meta Classifier

$$\{(h_1 x^{(i)}, h_2 x^{(i)}, \dots, h_k x^{(i)}, y^{(i)})\}$$

D **First level learner**

(Base learner)

$$\{(x^{(1)}, y^{(1)}) \dots (x^{(n)}, y^{(n)})\}$$

模型结合的不同方法 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: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \cdots, h_T(\boldsymbol{x}))$

集成方法 Ensemble Methods

classify according to the generation mode of individual learners(base learners, usually weak learners)

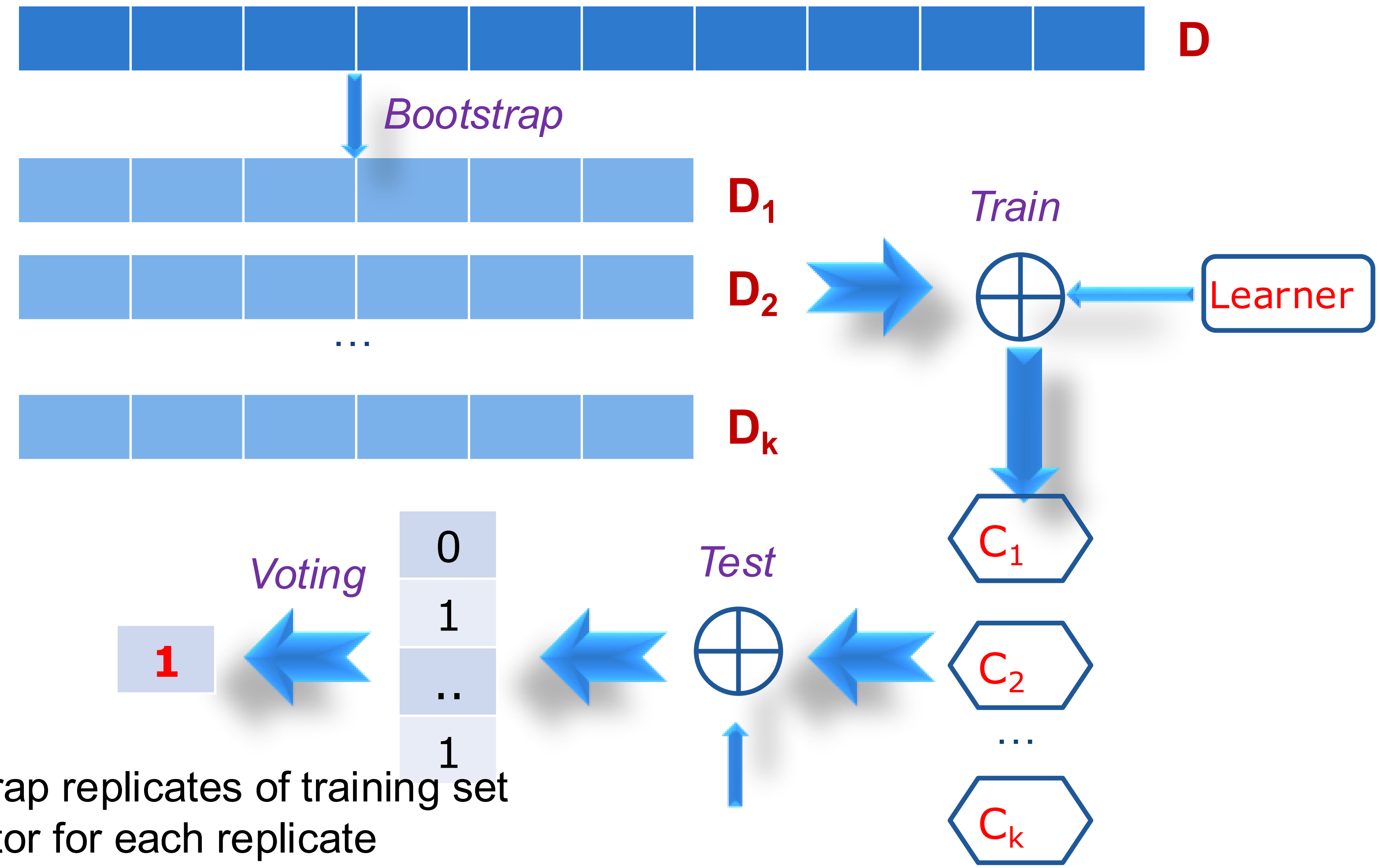
Parallel Methods

Bagging — Random Forest — for each base learner: randomly select both sample units and features

Sequential Methods

Boosting
- **Adaboost** — re-weighting sample units in each iteration
- **Gradient Boosting** — residual-decreasing way

装袋法 Bagging



- Create bootstrap replicates of training set
- Train a predictor for each replicate
- Validate the predictor using out-of-bootstrap data
- Average output of all predictors

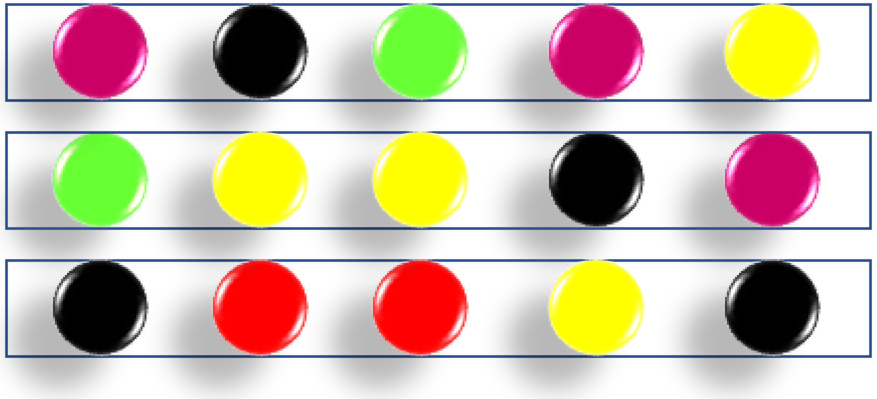
Breiman 1996a

自助采样 Bootstrap Samples

Sample 1

Sample 2

Sample 3



- Bootstrap replication
  - Given *n* training samples *Z*, construct a new training set *Z** by sampling *n* instances with replacement
  - Excludes about 37% of the training instances

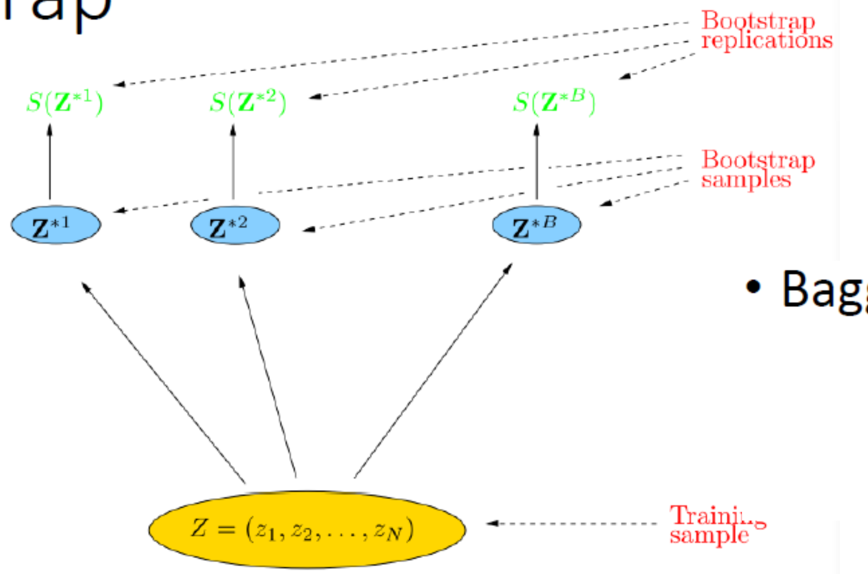
$$P\{\text{observation } i \in \text{bootstrap samples}\} = 1 - \left(1 - \frac{1}{N}\right)^N$$
$$\simeq 1 - e^{-1} = 0.632$$

Validate the predictor using out-of-bootstrap data

装袋法（自举汇聚法）Bagging (Bootstrap Aggregating)

- Bootstrap replication
  - Given n training samples $Z = \{(x_1,y_1), (x_2,y_2),...,(x_n,y_n)\}$, construct a new training set Z^* by sampling n instances with replacement
  - Construct B bootstrap samples Z^{*b} , $b = 1,2,...,B$
  - Train a set of predictors $\hat{f}^{*1}(x), \hat{f}^{*2}(x), \ldots, \hat{f}^{*B}(x)$

Bootstrap



- Bagging average the predictions

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$$

- Basic idea

装袋法之随机森林 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.
- A Collection of CART-like Trees
  - Binary Partition
  - No Pruning
  - Inherent Randomness
- Majority Voting

随机森林 Random forests

- Breiman, Leo. "Random forests." *Machine learning* 45.1 (2001): 532.
- Random forest is a substantial modification of bagging that builds a large collection of de-correlated trees, and then average them.

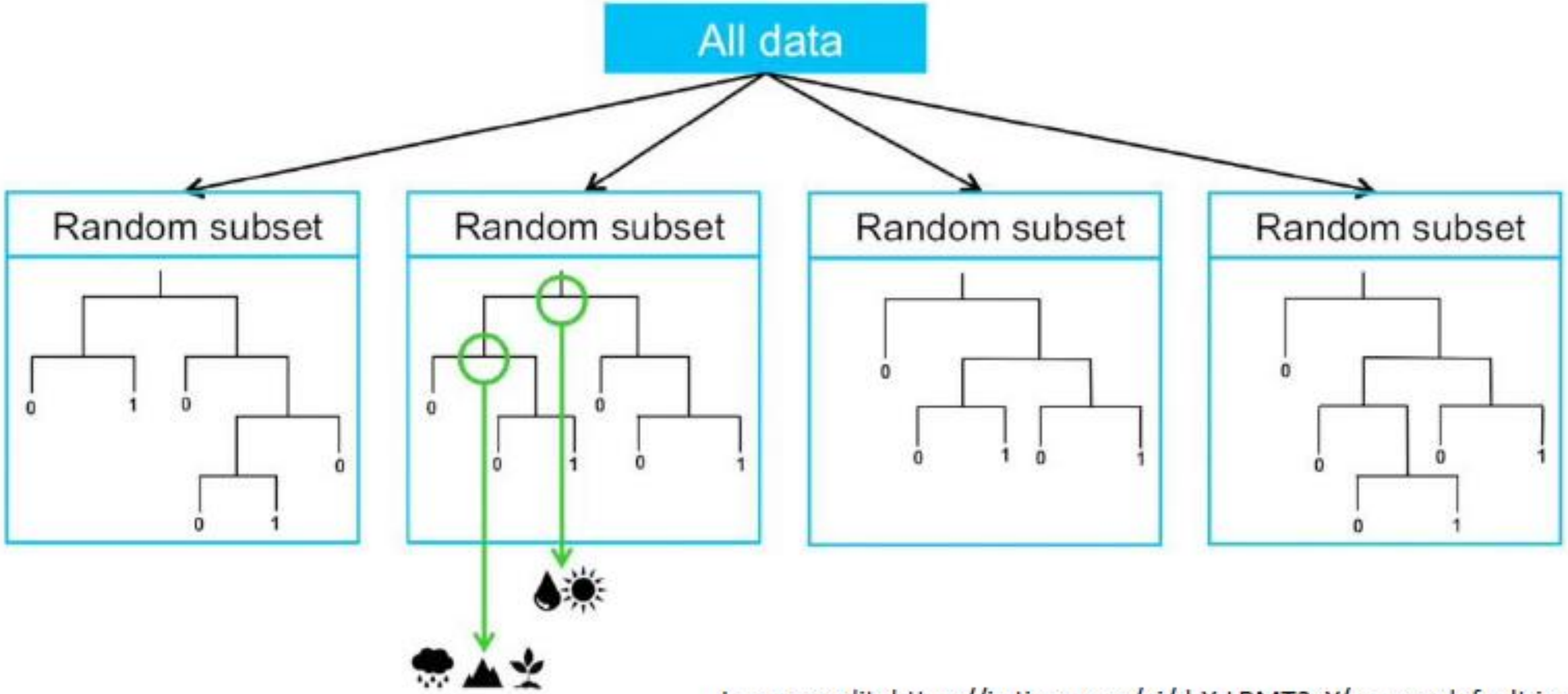
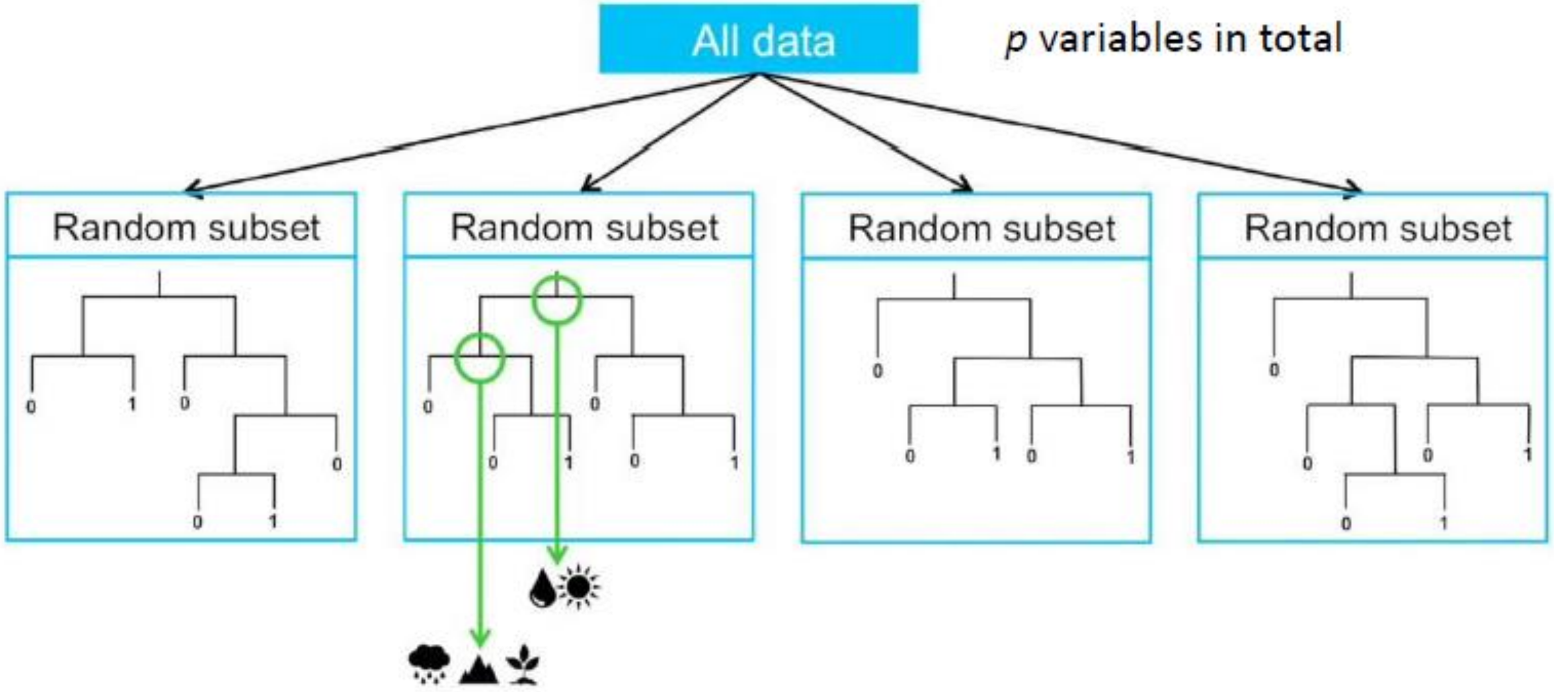


Image credit: https://i.ytimg.com/vi/-bYrLRMT3vY/maxresdefault.jpg

随机森林中树的去相关 Tree De-correlation in Random Forest



- Before each tree node split, select $m \leq p$ variables at random as candidates of splitting
  - Typically values $m = \sqrt{p}$ or even low as 1

随机森林算法 Random Forest Algorithm

- For b = 1 to B:
  a) Draw a bootstrap sample Z^* of size n from training data
  b) Grow a random-forest tree T_b to the bootstrap data, by recursively repeating the following steps for each leaf node of the tree, until the minimum node size is reached
     I. Select m variables at random from the p variables
     II. Pick the best variable & split-point among the m
     III. Split the node into two child nodes
- Output the ensemble of trees $\{T_b\}_{b=1...B}$
- To make a prediction at a new point x

Regression: prediction average $$\hat{f}_{\mathrm{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$$

Classification: majority voting $$\hat{C}_{\mathrm{rf}}^B(x) = \text{majority vote } \{\hat{C}_b(x)\}_1^B$$

Algorithm 15.1 of Hastie et al. The elements of statistical learning.

随机森林优点 Random Forest 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(out of bag) are used to evaluate the current tree.
- 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.

集成方法 Ensemble Methods

classify according to the generation mode of individual learners(base learners, usually weak learners)

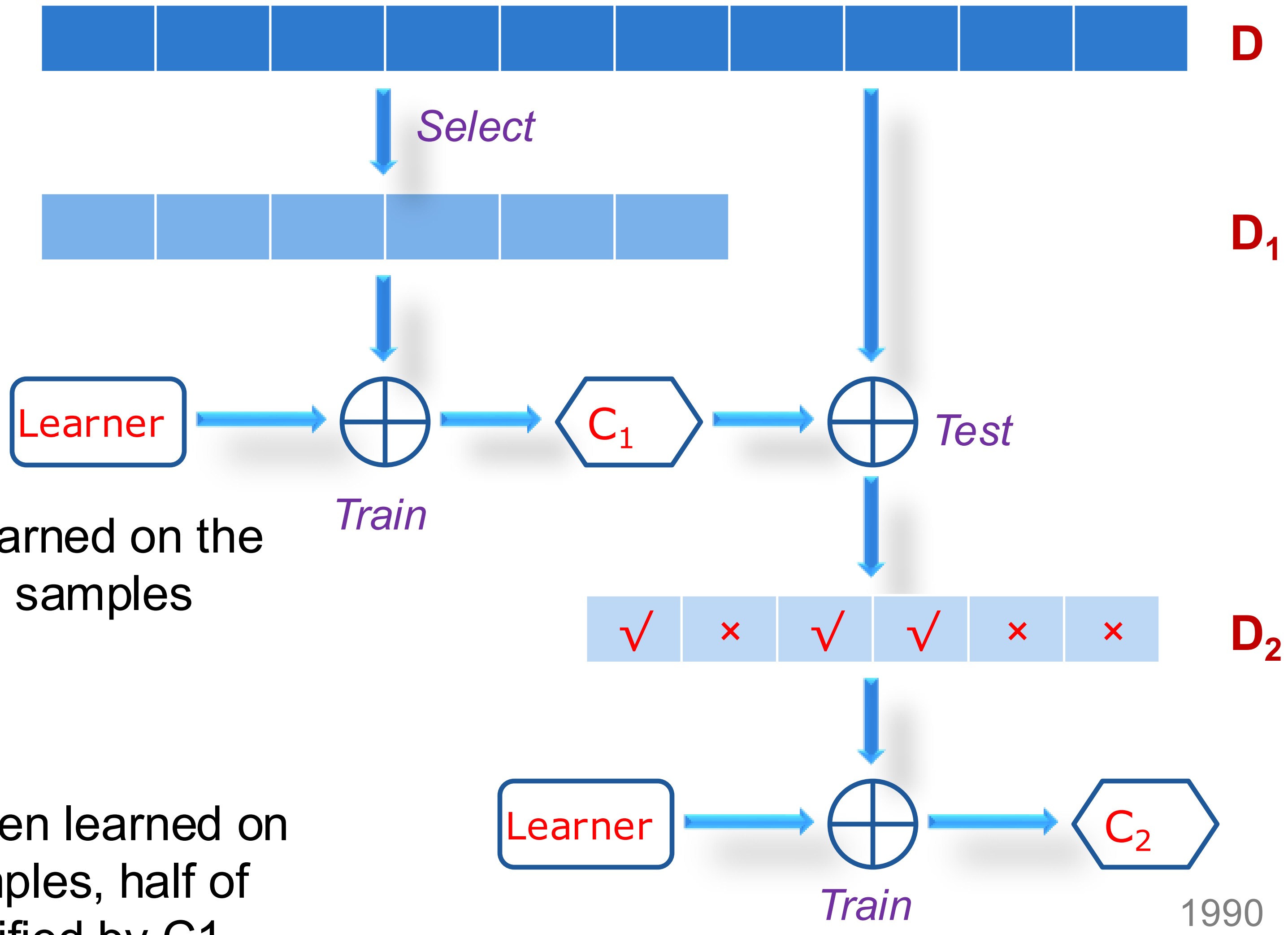
Parallel Methods

- Bagging — Random Forest — for each base learner: randomly select both sample units and features

Sequential Methods

- Boosting
  - Adaboost — re-weighting sample units in each iteration
  - Gradient Boosting — residual-decreasing way

提升法 Boosting

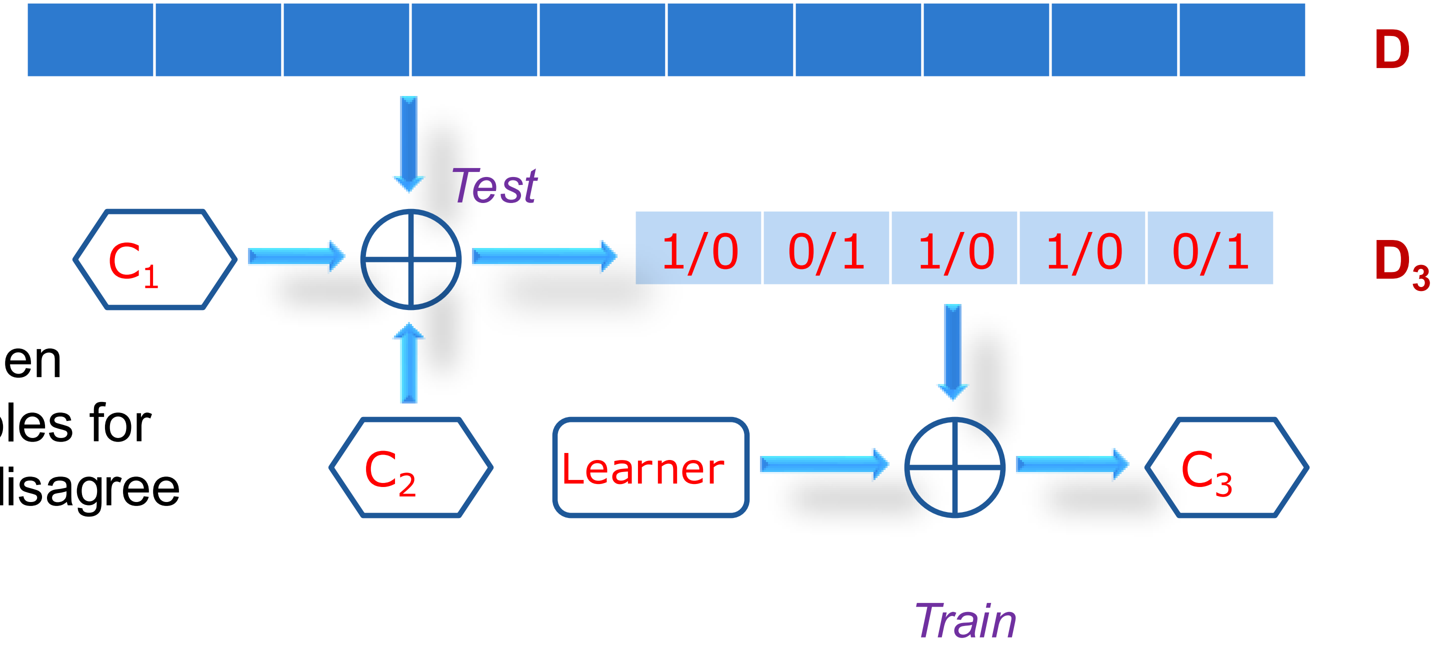


- Classifier C1 is learned on the original data with N samples

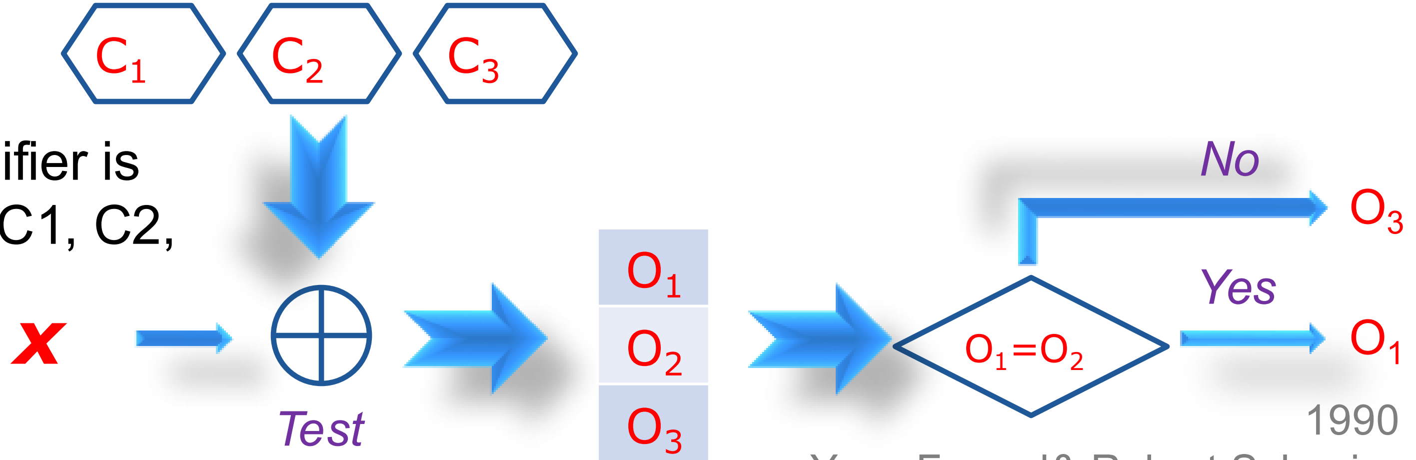
- Classifier C2 is then learned on a new set of N samples, half of which are misclassified by C1

1990
Yoav Freund& Robert Schapire

提升法 Boosting



- Classifier C3 is then learned on N samples for which C1 and C2 disagree



- The boosted classifier is CB = Majority Vote(C1, C2, C3)

1990
Yoav Freund& Robert Schapire

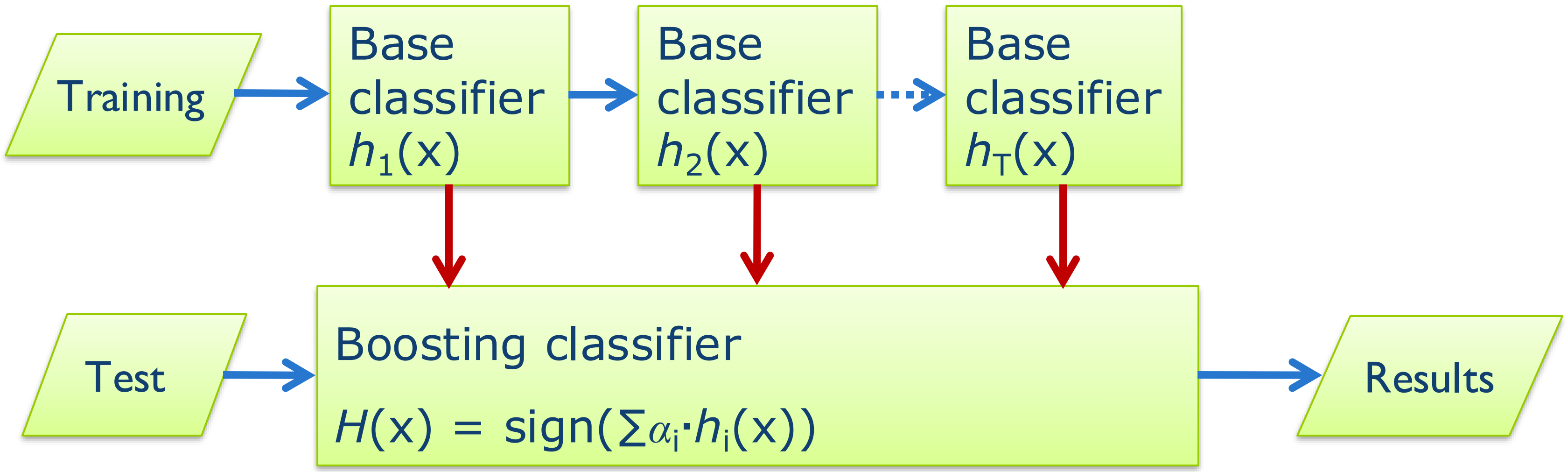
提升法 Boosting

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

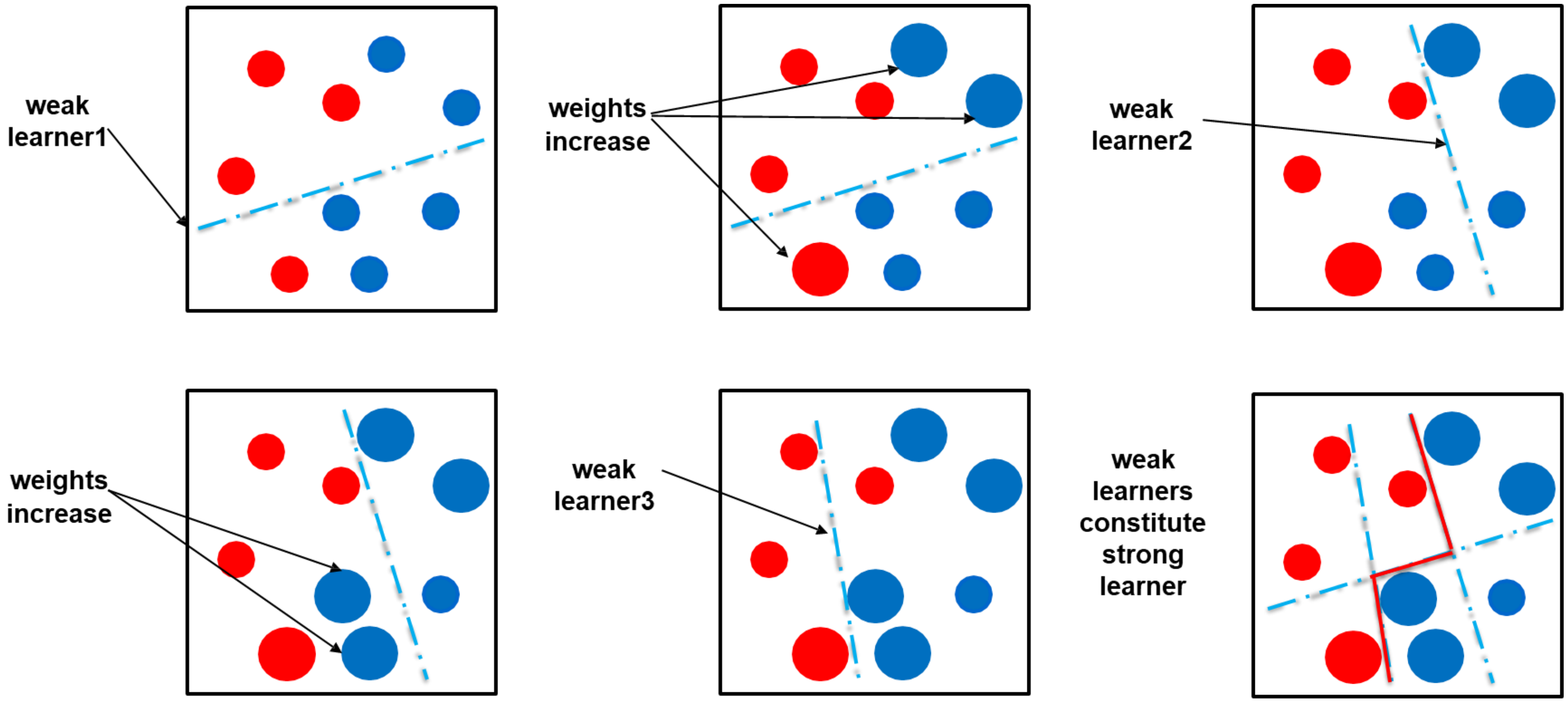
Process:

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



提升法 Boosting



提升法 Boosting

- 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!

自适应增强 AdaBoost （Adaptive Boosting）

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. $$\begin{aligned} \mathcal{D}_{t+1}(i) &= \frac{\mathcal{D}_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases} \\ &= \frac{\mathcal{D}_t(i)\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t} \end{aligned}$$
   % Update the distribution, where
   % Z_t is a normalization factor which
   % enables $\mathcal{D}_{t+1}$ to be a distribution
8. **end**

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

1997

Freund and Schapire

自适应增强 AdaBoost （Adaptive Boosting）

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. $\boxed{\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right);}$ % Determine the weight of h_t
7. $$\boxed{\mathcal{D}_{t+1}(i) = \frac{\mathcal{D}_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases}}$$
   $$= \frac{\mathcal{D}_t(i)\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$$
   % Update the distribution, where
   % Z_t is a normalization factor which
   % enables $\mathcal{D}_{t+1}$ to be a distribution
8. **end**

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

1997

Freund and Schapire

梯度提升树 Gradient Boosting Decision Tree (GBDT)

Boosting tree:

$$f_M(x) = \sum_{m=1}^{M} T(x, \theta_m)$$

$T(x, \theta_m)$: Regression Decision Tree （ DT）

- It consists of three concepts:
  - Regression Decision Tree （ DT）
  - Gradient Boosting （GB）
  - Shrinkage
- The CART is applied in GBDT as base learner.

Many aliases: GBT（Gradient Boosting Tree）,GTB（Gradient Tree Boosting）, GBRT（Gradient Boosting Regression Tree）,MART(Multiple Additive Regression Tree) (GradientTree Boosting: GradientBoostingClassifier, GradientBoostingRegressor in Sklearn)

梯度提升树 Gradient Boosting Decision Tree (GBDT)

Boosting tree:

$$f_M(x) = \sum_{m=1}^{M} T(x, \theta_m)$$

Forward stagewise additive modeling algorithm:

- Initialize the boosting tree:

$$f_0(x) = 0$$

- Iterative calculation of the m th boosting tree:

$$f_m(x) = f_{m-1}(x) + T(x, \theta_m), m = 1,2, \dots . M$$

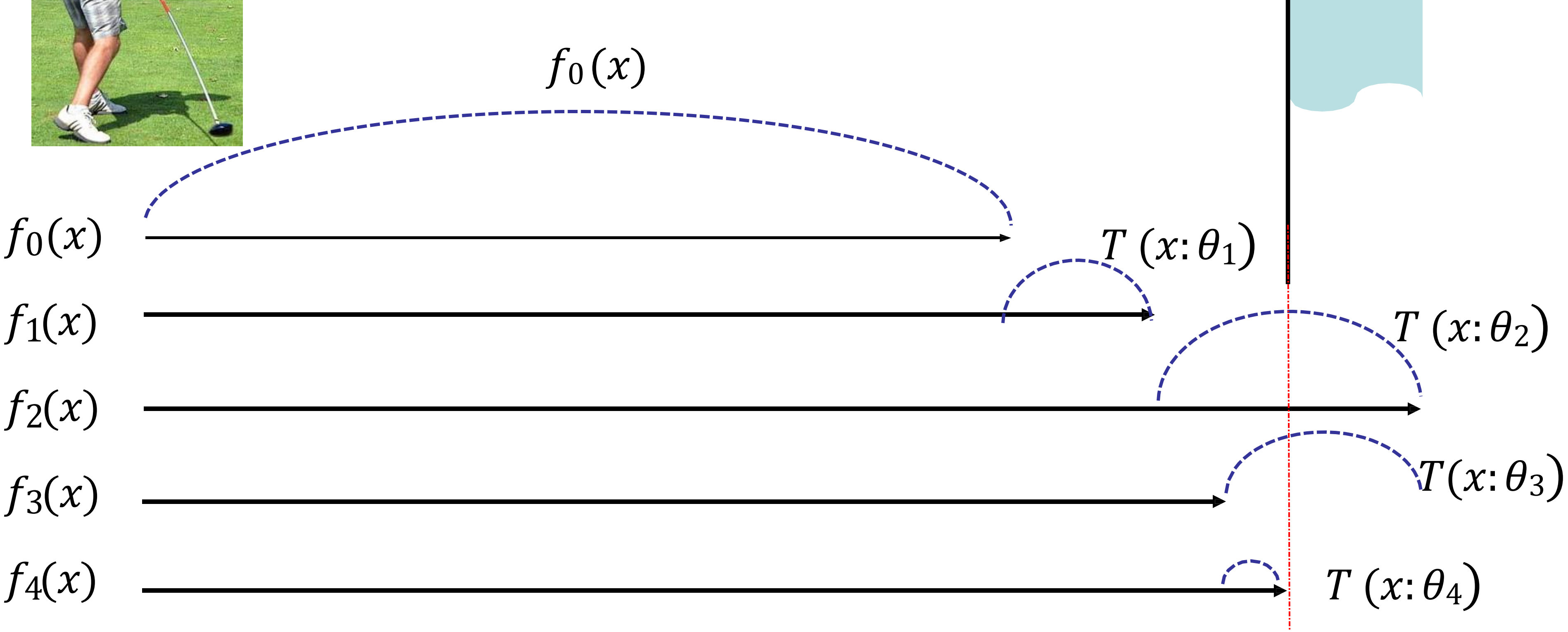
$$\hat{\theta}_m = arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i, \theta_m))$$

Grow the m decision tree to minimize the loss function

梯度提升树 Gradient Boosting Decision Tree (GBDT)

Forward stagewise additive modeling algorithm:

$$f_m(x) = f_{m-1}(x) + T(x, \theta_m), m = 1,2, \dots . M$$



梯度提升树 Gradient Boosting Decision Tree (GBDT)

Forward stagewise additive modeling algorithm:

$$L(y, f(x)) = L(y, f_m(x)) \qquad f_m(x) = f_{m-1}(x) + T(x; \theta_m)$$

$$= L\big(y, f_{m-1}(x) + T(x; \theta_m)\big)$$

$$= (y - f_m(x))^2$$

$$= [\underline{y - f_{m-1}(x)} - T(x; \theta_m)]^2$$

$$= [r - T(x; \theta_m)]^2$$

$$T(x, \theta_m) \xrightarrow{\text{fit}} r \approx -\left[\frac{\partial L(y, f(x))}{\partial f(x)}\right]_{f(x) = f_{m-1}(x)}$$

梯度提升树 Gradient Boosting Decision Tree (GBDT)

Forward stagewise additive modeling algorithm:

Input: Data set $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$;

Loss Function $L\big(y, f(x)\big)$;

Process:

1. $f_0(x) = arg\min\limits_{c} \sum_{i=1}^{N} L(y_i, c)$ % Initialization
2. for $m = 1,2, \dots, M$:
3. for $i = 1,2, \dots, M$:
4. $r_{mi} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x) = f_{m-1}(x)}$
5. $r_{mi} \xrightarrow{\text{fit}} T(x: \theta_m)$ % Fit r_{mi} to generate Regression Tree
6. $\sigma_m = arg\min\limits_{c} \sum_{i=1} L(y_i, f_{m-1}(x_i) + \sigma T(x: \theta_m))$ % Calculate step
7. $f_m(x) = f_{m-1}(x) + \sigma_m T(x: \theta_m)$ % Update
8. end

Output: $f_M(x)$

CART树的损失函数 Loss function in CART

$$a_*, v_* = \underset{a \in A}{argmin} \left[\min_{c^l} \sum\nolimits_{x^i \in D^l} \left(y^i - c^l\right)^2 + \min_{c^r} \sum\nolimits_{x^i \in D^r} \left(y^i - c^r\right)^2 \right]$$

$$c_l = \frac{1}{N^l} \sum_{x^i \in D^l} y^i, \qquad c_r = \frac{1}{N^r} \sum_{x^i \in D^r} y^i$$

D^l and D^r are the subsets of D splitted by a =v.

The CART be applied in GBDT as base learner.

均方误差和CART算法 MSE and CART

x	1	2	3	4	5	6	7	8	9	10
y	5.56	5.7	5.91	6.4	6.8	7.05	8.9	8.7	9	9.05

x	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
c^l									
c^r									
MSE									

GBDT Example

$$\min_{s}[\min_{c_1}\sum(y_i-c_1)^2+\min_{c_2}\sum(y_i-c_2)^2]$$

$$R_1=\{x|x\le s\}\qquad R_2=\{x|x\ge s\}$$

⬇

$$c_1=\frac{1}{N_1}\sum_{x_i\epsilon R_1}y_i\qquad c_2=\frac{1}{N_1}\sum_{x_i\epsilon R_2}y_i$$

$$m(s)=\min_{c_1}\sum_{x_i\epsilon R_1}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\epsilon R_2}(y_i-c_2)^2$$

x	1	2	3	4	5	6	7	8	9	10
y	5.56	5.7	5.91	6.4	6.8	7.05	8.9	8.7	9	9.05

GBDT VS CART

Better Predictive Performance: GBDT tends to make more accurate predictions due to its ensemble learning approach.

Capturing Complex Patterns: GBDT is good at capturing complex relationships and non-linear patterns in data.

Gradient Optimization: GBDT uses gradient boosting, enabling faster convergence during training.

mitigate the risk of overfitting :multiple weak learners are sequentially trained to correct the residuals of the previous tree, tends to improve the model's generalization performance.

In short, GBDT is often more effective in predictive tasks, especially when dealing with complex data patterns.

分布式梯度增强库 XGBoost

- XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

- The most effective and efficient toolkit for GBDT

- https://xgboost.readthedocs.io/en/stable/tutorials/index.html

XGBoost

$$obj^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \omega(f_i) = \sum_{i}^{n} \underline{l(y_i, \hat{y}_i^{(t-1)}} + \underline{\underline{f_t(x_i)}}) + \omega(f_t) + constant$$

take the *Taylor expansion of the loss function up to the second order*:

where the g_i and h_i are defined as

$$g_i = \partial_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$

$$h_i = \partial^2_{\hat{y}_i^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$

$$obj^{(t)} \approx \sum_{i=1}^{n} \left[\underline{l(y_i, \hat{y}_i^{(t-1)})} + g_i \underline{\underline{f_t(x_i)}} + \frac{1}{2} h_i f_t^2(x_i) \right] + \omega(f_t) + constant$$

$$f(x + \Delta x) \approx \underline{f(x)} + f'(x)\underline{\underline{\Delta x}} + \frac{1}{2!} f''(x)\Delta x^2$$

Tips: n-order Taylor formula:

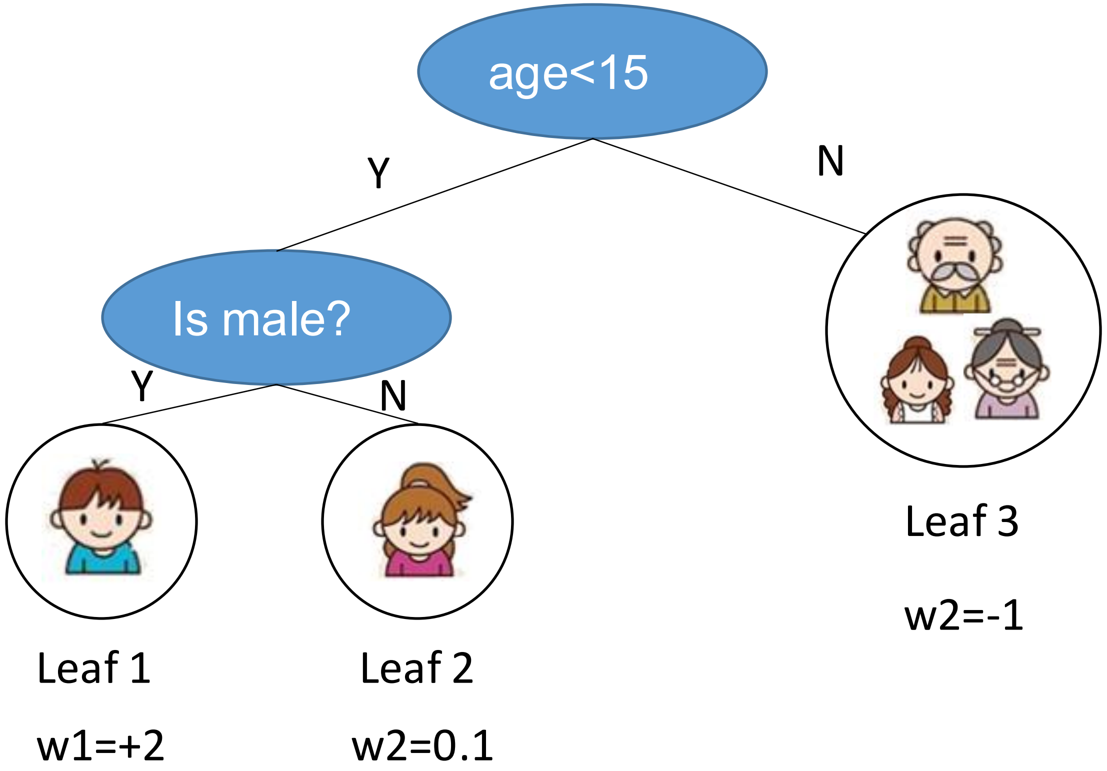
$$f(x) = f(0) + f'(0)x + \frac{1}{2!} f''(0)x^2 + \cdots + \frac{f^{(n)}(0)}{n!} x^n + R_n(x) \ldots\ldots$$

$R_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1}$ is defined as the n-order Taylor remainder of $f(x)$ at point x_0

XGBoost

$$obj^{(t)} = \sum_{i}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \omega(f_t) + constant$$

$$\omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$



$$\omega = \gamma 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

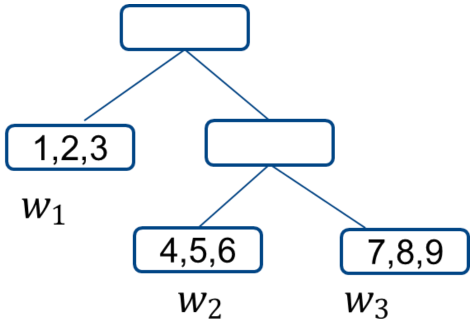
XGBoost

$$obj^{(t)} \approx \sum_{i=1}^{n}\left[\boxed{l(y_i, \hat{y}_i^{(t-1)})} + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \omega(f_t) \boxed{+\ constant}$$

$$obj^{(t)} \approx \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \omega(f_t)$$

$$f_t(x) = w_{q(x)}, w \in R^T, q: R^d \to \{1,2,\dots,T\}$$

$$\omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$



$$obj^{(t)} \approx \boxed{\sum_{i=1}^{n}}\left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \boxed{\sum_{j=1}^{T}}\left[\left(\sum_{i \in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i\right) w_j^2\right] + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 + \gamma T$$

$$= \sum_{j=1}^{T}\left[\left(\sum_{i \in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i \in I_j} h_i + \lambda\right) w_j^2\right] + \gamma T$$

XGBoost

$$obj^{(t)} \approx \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2\right] + \gamma T$$

By defining $G_j = \sum_{i\in I_j} g_i$ and $H_j = \sum_{i\in I_j} h_i$:

$$obj^{(t)} = \sum_{j=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j + \lambda)w_j^2\right] + \gamma T$$

$$\frac{\partial obj^{(t)}}{\partial w_j} == 0$$

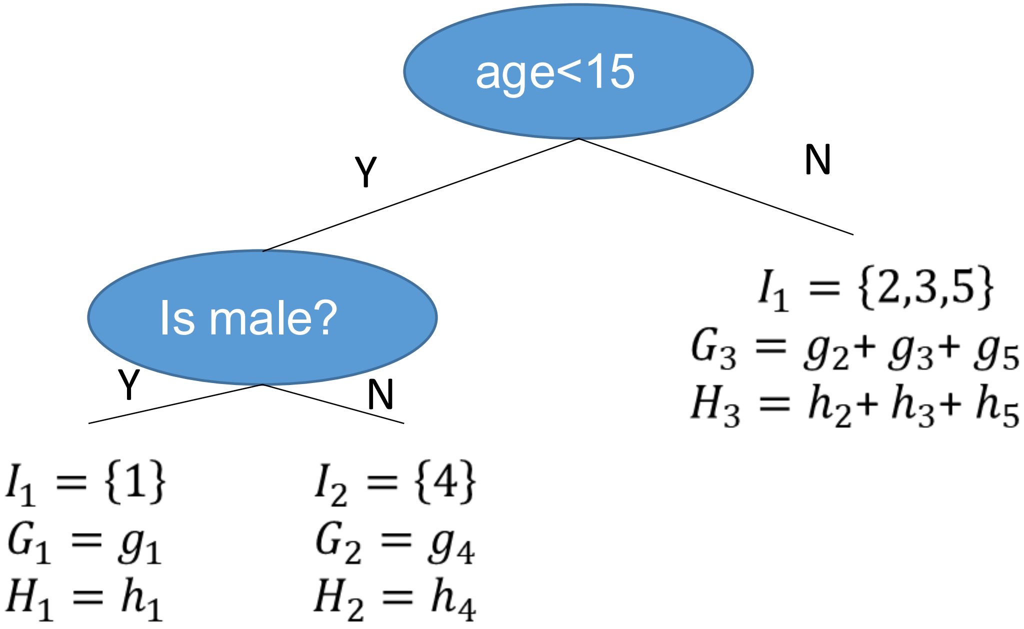
$$w_j^* = -\frac{G_j}{H_j + \lambda}$$

$$obj^* = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T$$

$$\begin{aligned} obj^* &= \sum_{j=1}^{T}\left[G_j\left(-\frac{G_j}{H_j+\lambda}\right) + \frac{1}{2}(H_j+\lambda)\left(-\frac{G_j}{H_j+\lambda}\right)^2\right] + \gamma T \\ &= \sum_{j=1}^{T}\left[-\frac{G_j^2}{H_j+\lambda} + \frac{G_j^2}{2(H_j+\lambda)}\right] + \gamma T \\ &= -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T \end{aligned}$$

XGBoost

Instance index	gradient statistics
1	g1,h1
2	g2,h2
3	g3,h3
4	g4,h4
5	g5,h5

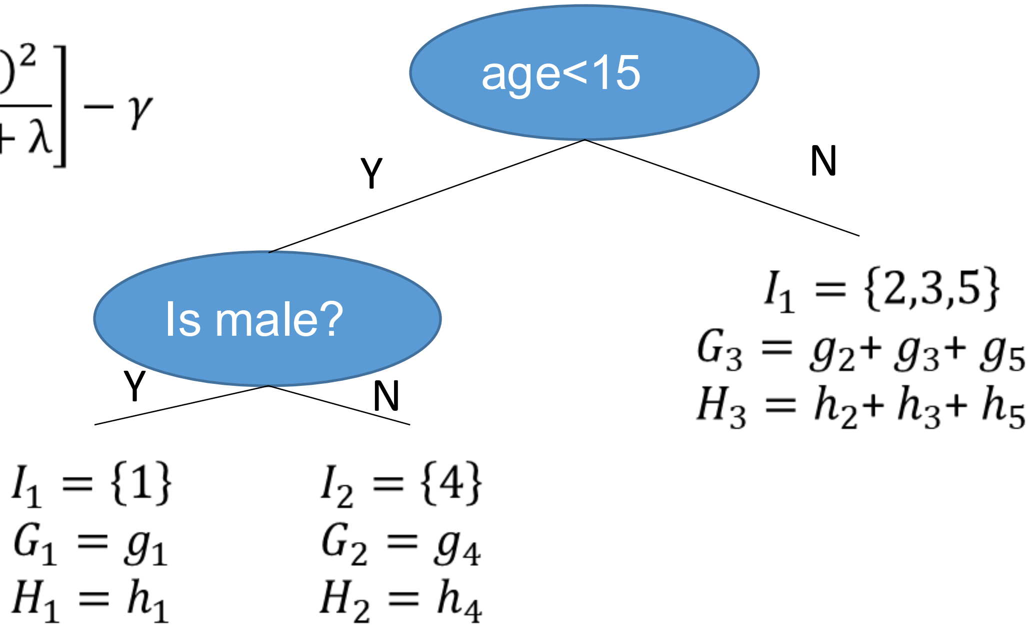


$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

The smaller the score is, the better the structure is

XGBoost

$$gain(\emptyset) = gain(brfore) - gain(after)$$
$$= \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$



Using a greedy approach, select the split with the maximum gain.

$$Obj = -\sum_j \frac{G_j^2}{H_j+\lambda} + 3\gamma$$

The smaller the score is, the better the structure is

轻量级梯度提升机器 LightGBM

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel, distributed, and GPU learning.
- Capable of handling large-scale data.

- https://lightgbm.readthedocs.io/en/latest/

轻量级梯度提升机器 LightGBM

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages:

- **Optimization in Speed and Memory Usage**

    Gradient based one-side sampling(GOSS)

    Exclusive feature bunding(EFB)

    histogram

LightGBM = XGboost+GOSS+EFB+ histogram

- **Optimization in Accuracy**

    Leaf-wise (Best-first) Tree Growth

- **Optimization in Network Communication**
- **Optimization in Distributed Learning**