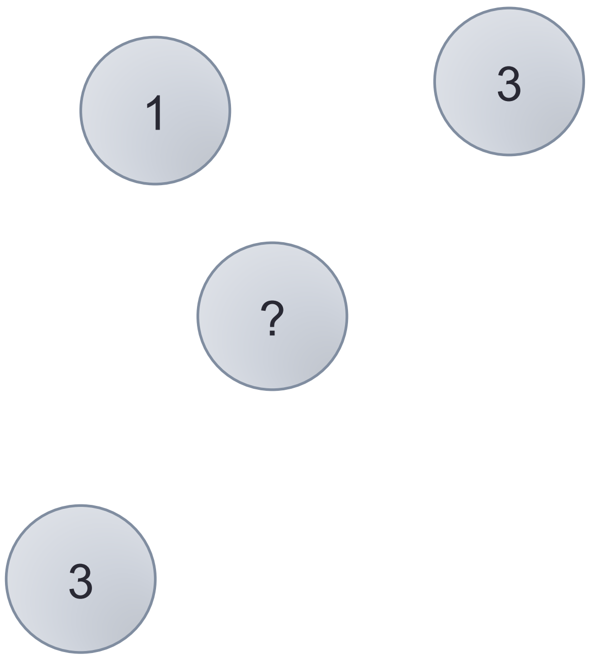
Majority Voting & Ensemble Learning

1. Example

- KNN (k=3: 3 nearest neighbors)



3 votes [1,3,3] for final class decision: class label = 3.

1. Example

- Random Forest

	DT_1	DT_2	……	DT_{100}
X	l_1	l_1	……	l_{100}

100 votes $[l_1, l_2, …, l_{100}]$ for final class decision

- One decision tree (DT_1) is described as follows

C_1	C_2	……	C_{10}
0.15	0.8	……	0.9

Probabilities of being classified to each class

1. Example

- Combination/Fusion of different methods

SVM	KNN	Decision Tree	Linear Regression	Logistic Regression
l_1	l_2	l_3	l_4	l_5

5 votes [l_1, l_2, l_3, l_4, l_5] for final class decision

2. Two-class Voting

- Decision Tree (DT)

	Correct	Incorrect	
p=	0.5	0.5	Bad Tree
p=	0.6	0.4	Good Tree

Probabilities of being classified to correct/incorrect class

2. Two-class Voting

- A variable described as the number of successes in a sequence of independent Bernoulli trials has **Binomial distribution**. Its parameters are n, the number of trials, and p, the probability of success.

- Binomial probability mass function is:

$$P(x) = \boldsymbol{P}\{X = x\} = \binom{n}{x} p^x q^{n-x}, \qquad x = 0, 1, \ldots, n,$$

- Binomial distribution:

$$\boxed{\begin{array}{lcl} n & = & \text{number of trials} \\ p & = & \text{probability of success} \\ P(x) & = & \binom{n}{x} p^x q^{n-x} \\ \mathbf{E}(X) & = & np \\ \mathrm{Var}(X) & = & npq \end{array}}$$

2. Two-class Voting

- Random Forest: Using 3 Decision Trees

DT1	DT2	DT3	Voting	Probability
0.6	0.6	0.6		
C	C	C	3↑	$P\uparrow\uparrow\uparrow = 0.6^3 = 0.216$
C	C	INC	2↑ 1↓	$P\uparrow\uparrow\downarrow = 3 \times 0.6^2 \times 0.4 = 0.432$
C	INC	C		
INC	C	C		
INC	INC	C	1↑ 2↓	$P\uparrow\downarrow\downarrow = 3 \times 0.6 \times 0.4^2 = 0.288$
INC	C	INC		
C	INC	INC		
INC	INC	INC	3↓	$P\downarrow\downarrow\downarrow = 0.4^3 = 0.064$

Accuracy = $\frac{0.216+0.432}{1}$ = 0.648.

Using binomial distribution are for any number of votes.

2. Two-class Voting

- Random Forest: Using 51 Decision Trees

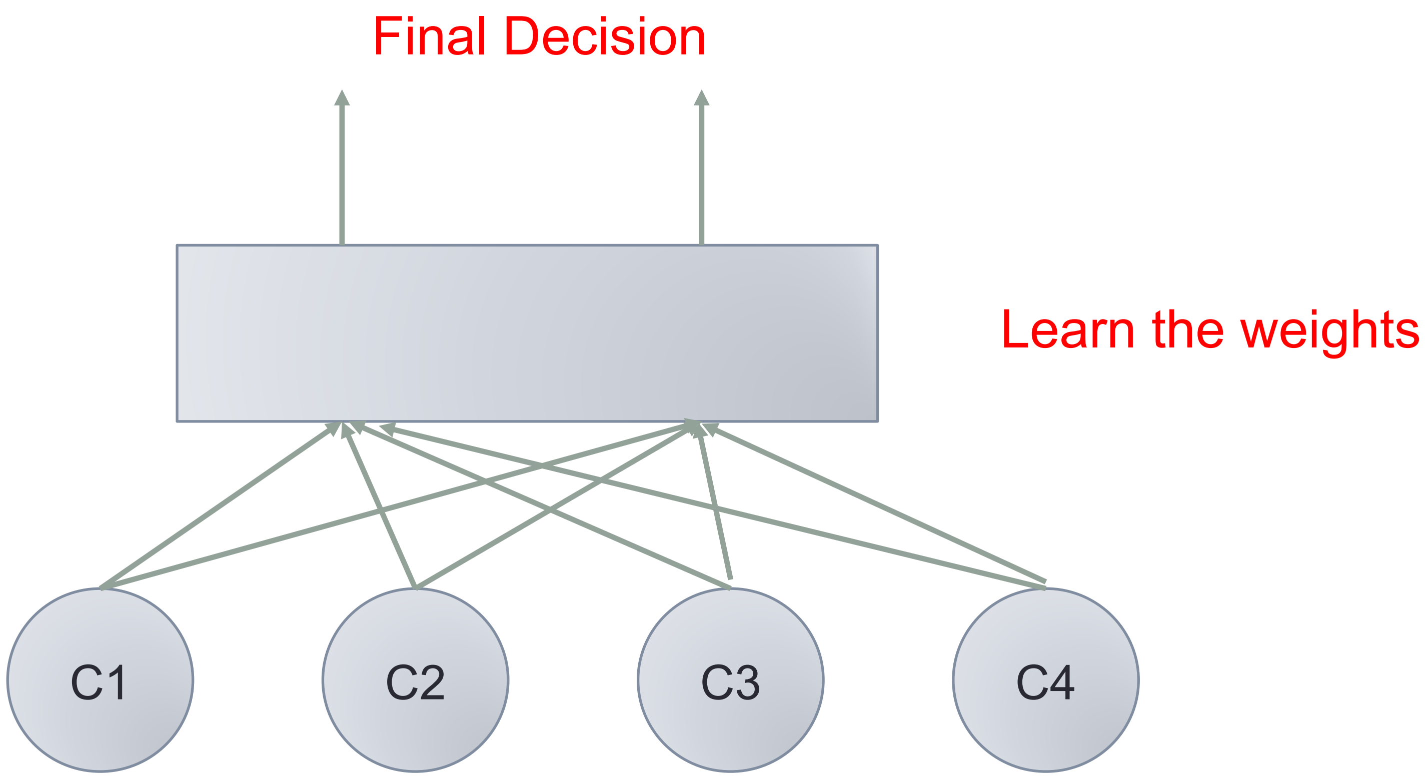
Voting	Probability
51↑	$P(51\uparrow) = 0.6^{51} = 4.8497\text{e-}12$
50↑, 1↓	$P(50\uparrow, 1\downarrow) = \binom{51}{50} \times 0.6^{50} \times 0.4 = 1.6489\text{e-}10$
……	……
26↑, 25↓	$P(26\uparrow, 25\downarrow) = \binom{51}{26} \times 0.6^{26} \times 0.4^{25} = 0.0476$
……	……
51↓	$P(51\downarrow) = 0.4^{51} = 5.0706\text{e-}21$

Accuracy = $\frac{P(51\uparrow)+P(50\uparrow,\ 1\downarrow)+\ldots+P(26\uparrow,\ 25\downarrow)}{1}$ = 0.9265.

Using binomial distribution are for any # of votes.

3. Ensemble Learning

- Example: Voting Machine



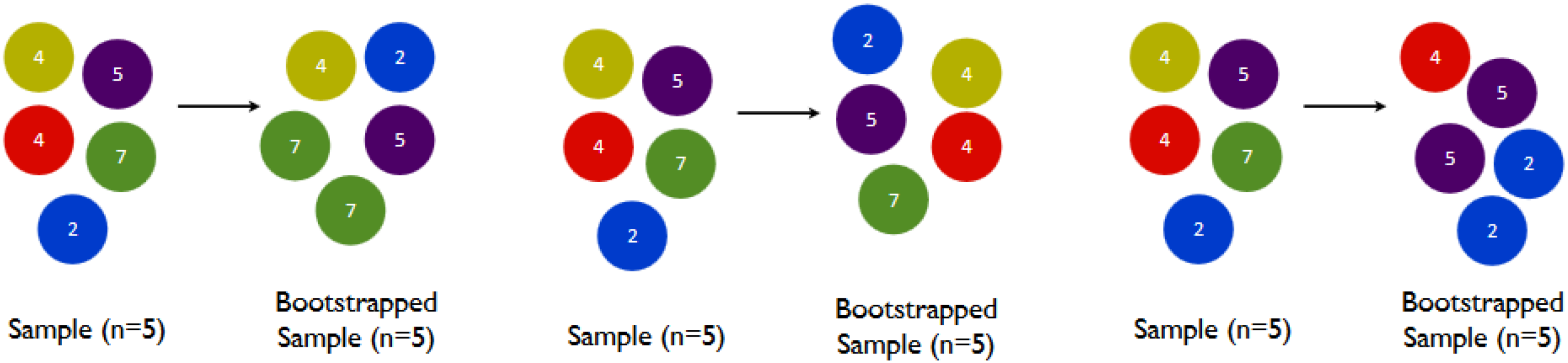
3.1 Bagging

- Bagging = Bootstrap Aggregating

- In the Bootstrap, we replicate our dataset by sampling with replacement:
  - Original Dataset: $Z = (z_1, z_2, \ldots, z_N)$, where $z_i = (x_i, y_i)$.
  - Bootstrap samples:
    - Z^{*1} = sample(x, 100, replace = True)
    - ……
    - Z^{*B} = sample(x, 100, replace = True)

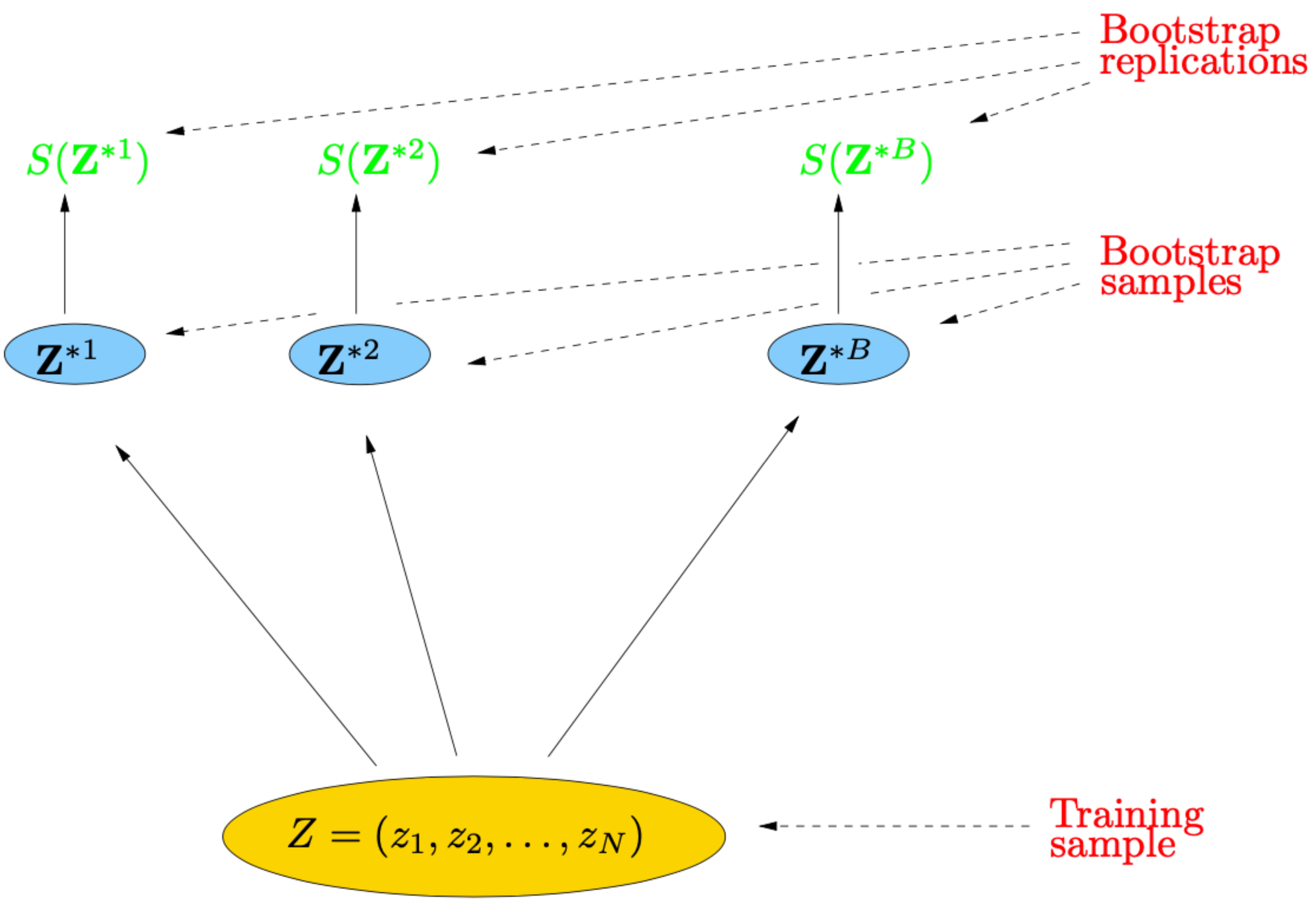
3.1 Bagging

- Bootstrap Samples (with replacement):



3.1 Bagging

- Bootstrap Process:



3.1 Bagging

- S(Z) is any quantity computed from the data Z.
  - For example, its variance:

$$\widehat{\mathrm{Var}}[S(\mathbf{Z})] = \frac{1}{B-1}\sum_{b=1}^{B}(S(\mathbf{Z}^{*b}) - \bar{S}^{*})^2$$

  where $\bar{S}^{*} = \sum_b S(\mathbf{Z}^{*b})/B$

- Apply the bootstrap to estimate prediction error.
  - If f^*b(x_i) is the predicted value at x_i, from the model fitted to the b-th bootstrap dataset, our estimate is:

$$\widehat{\mathrm{Err}}_{\mathrm{boot}} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L(y_i, \hat{f}^{*b}(x_i))$$

3.1 Bagging

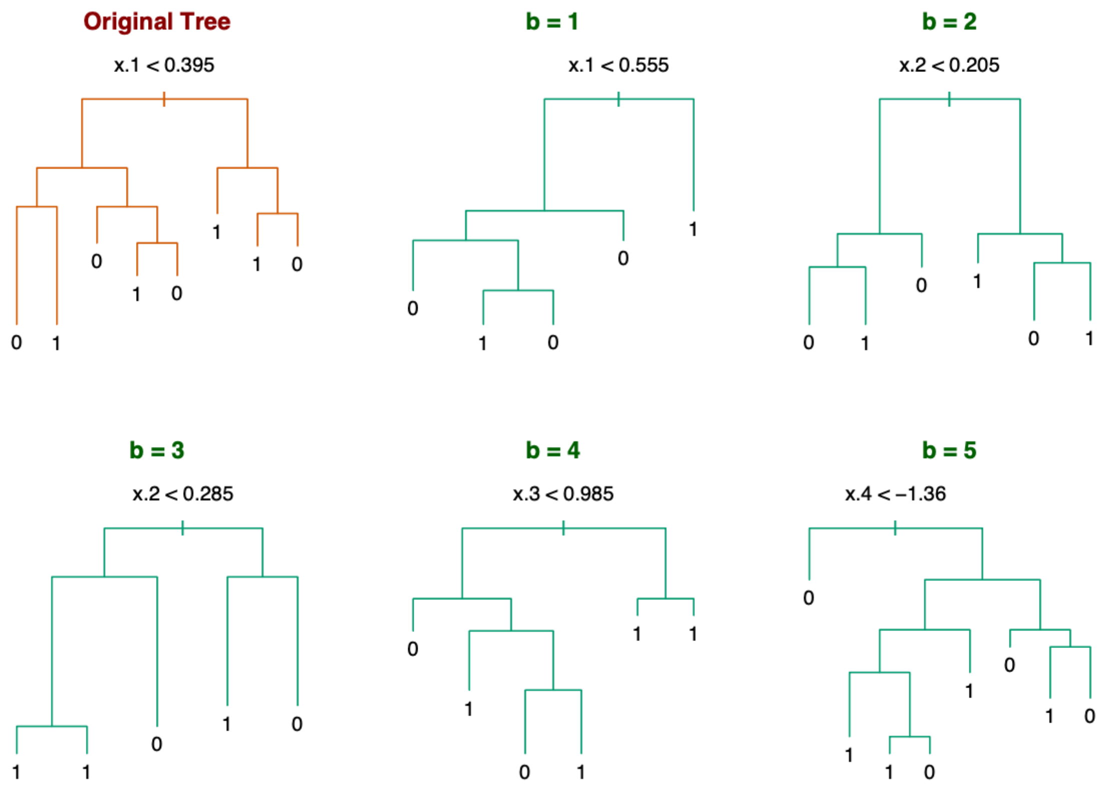
- For each bootstrap sample Z^{*b}, b = 1,2,...,B, we fit our model, giving prediction $\hat{f}^{*b}(x)$.
- The bagging estimate is defined by

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

- Example: bagging the “linear regression”.
  - Let $\hat{y}^{L,b}$ be the prediction of the decision tree applied to the b-th bootstrap sample.
  - Bagging prediction: $\hat{y}^{\text{boot}} = \frac{1}{B}\sum_{b=1}^{B}\hat{y}^{L,b}.$
- When a regression method or a classifier has a tendency to overfit, Bagging reduces the variance of the prediction.

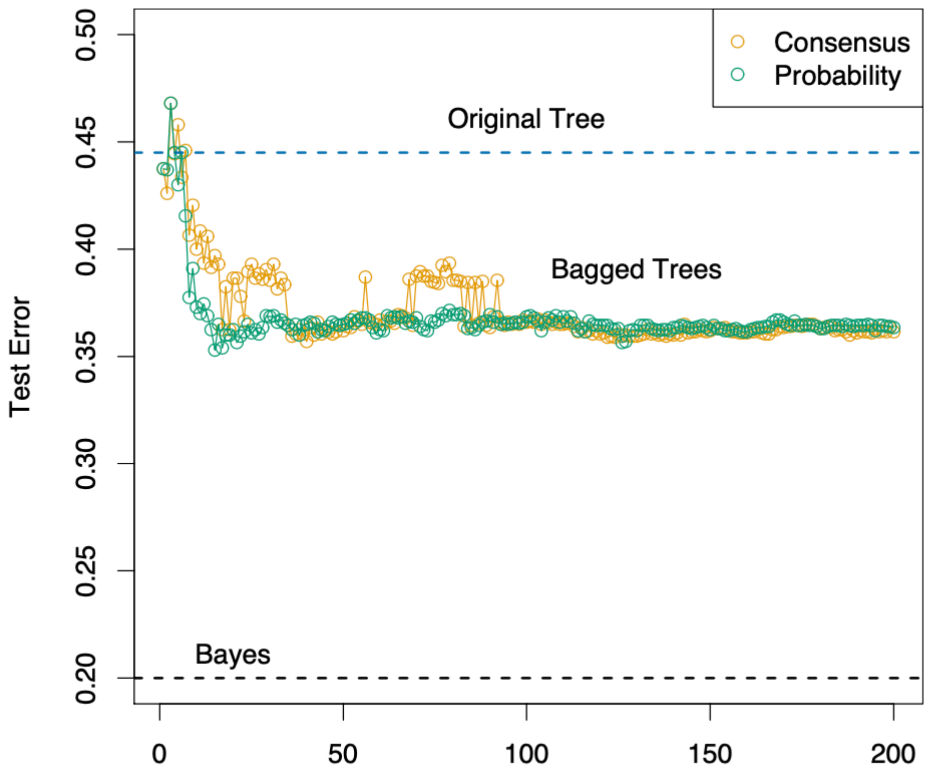
3.1 Bagging

- Example: Tree with simulated data.
  - We generated a sample of size N = 30, with two classes and p = 5 features.



3.1 Bagging

- Example: Tree with simulated data.



The orange points correspond to the consensus vote, while the green points average the probabilities.

3.2 Boosting

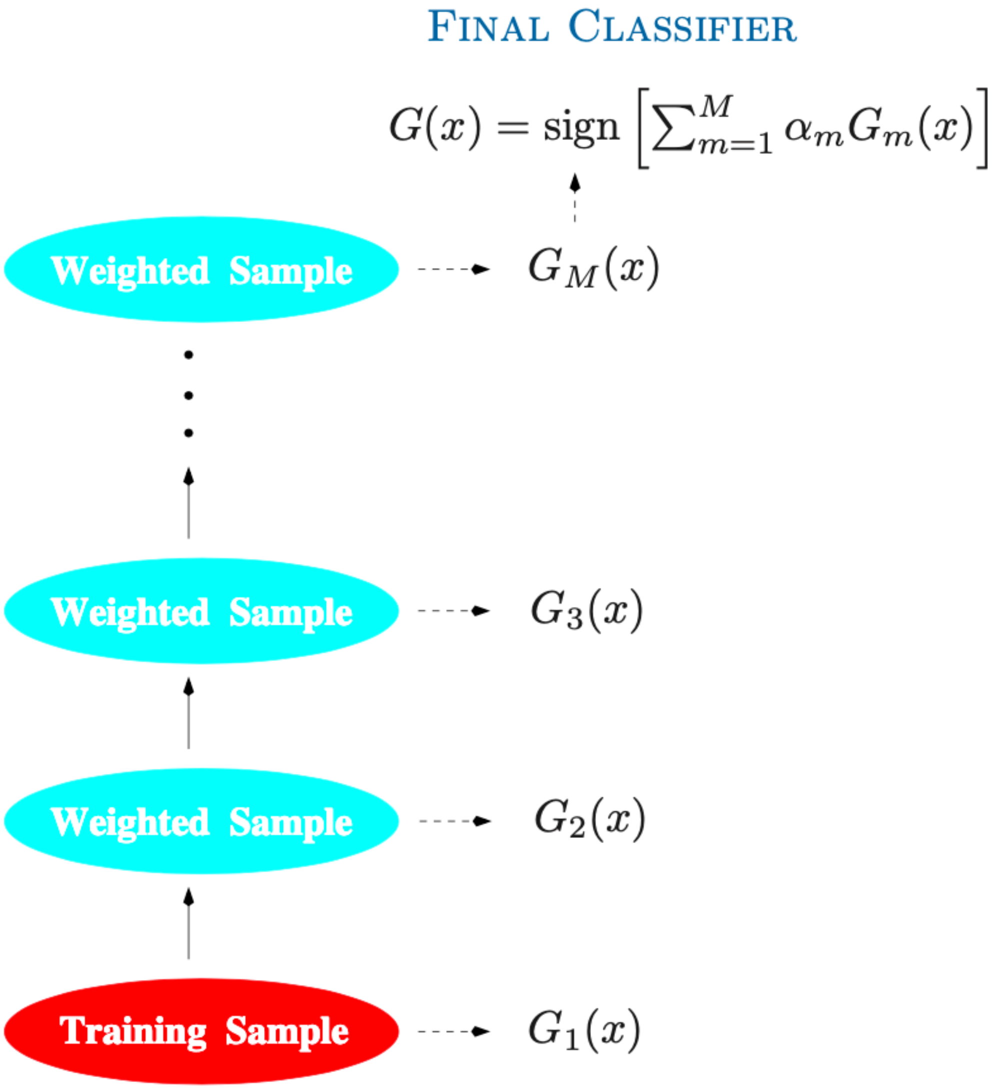
- The motivation for boosting was a procedure that combines the outputs of many “weak” classifiers to produce a powerful “committee.”

- Boosting learns slowly:
  - We first use the samples that are easiest to predict, then slowly down weigh these cases, moving on to harder samples.

3.2 Boosting

- Example: AdaBoost.

The data modifications at each boosting step consist of applying weights $w_1, w_2, \ldots, w_N$ to each of the training observations (x_i, y_i), $i = 1, 2, \ldots, N$.



Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.

3.2 Boosting

- Example: Algorithm - AdaBoost.M1

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \mathrm{err}_m)/\mathrm{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.
3. Output $G(x) = \mathrm{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

3.2 Boosting

- Boosted Trees
  - A tree can be formally expressed as

$$T(x;\Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$$

  - The parameters are found by minimizing the empirical risk

$$\tilde{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} \tilde{L}(y_i, T(x_i, \Theta))$$

3.2 Boosting

- Boosted Trees
  - The boosted tree model is a sum of such trees

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$$

  - Where at each step in the forward stagewise procedure one must solve

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

3.2 Boosting

- Forward stagewise boosting: greedy strategy.
  - At each step the solution tree is the one that maximally reduces the loss, given the current model f_{m-1} and its fits $f_{m-1}(x_i)$.
  - Thus, the tree predictions $T(x_i; \Theta_m)$ are analogous to the components of the negative gradient:

$$g_{im} = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x_i)=f_{m-1}(x_i)}$$

- Gradient boosting:
  - Induce a tree $T(x; \Theta_m)$ at the m-th iteration whose predictions tm are as close **as possible to the negative gradient**.
  - Using **squared error** to measure closeness, this leads us to:

$$\tilde{\Theta}_m = \arg\min_{\Theta} \sum_{i=1}^{N} (-g_{im} - T(x_i; \Theta))^2$$

3.2 Boosting

- Gradient Tree Boosting Algorithm

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.
2. For $m = 1$ to M:

   (a) For $i = 1, 2, \ldots, N$ compute

   Compute Negative Gradient with respect to 1,...,m-1 trees

   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

   Fit m-th Tree

   (b) Fit a regression tree to the targets r_{im} giving terminal regions $R_{jm},\ j = 1, 2, \ldots, J_m$.

   (c) For $j = 1, 2, \ldots, J_m$ compute

   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   Update Boosted Tree

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.

3. Output $\hat{f}(x) = f_M(x)$.