Machine Learning Lecture 6

Ensemble and Boosting Algorithms

Qian Ma

Sun Yat-sen University

Spring Semester, 2020

Acknowledgement

- A large part of slides in this lecture are originally from
  - Prof. Andrew Ng (Stanford University)
  - Prof. Weinan Zhang (Shanghai Jiao Tong University)

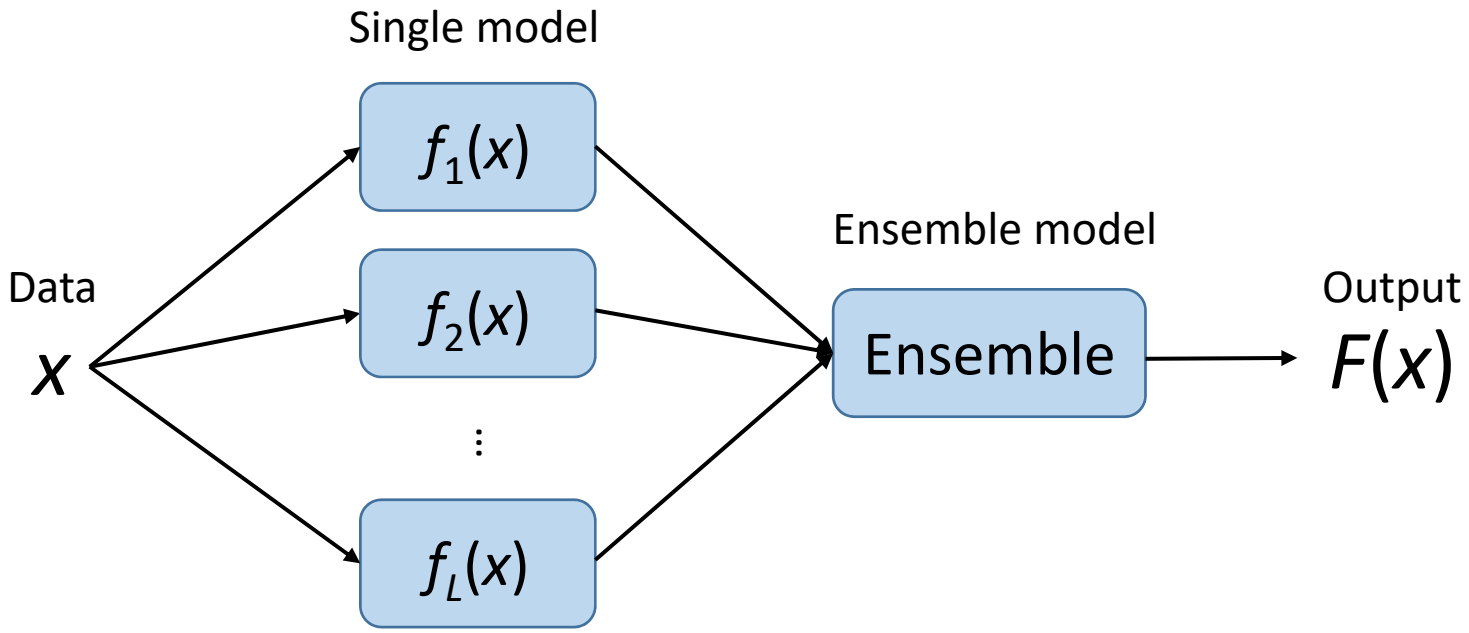
Prof. Andrew Ng
Stanford University

Prof. Weinan Zhang
Shanghai Jiao Tong University

Ensemble Learning

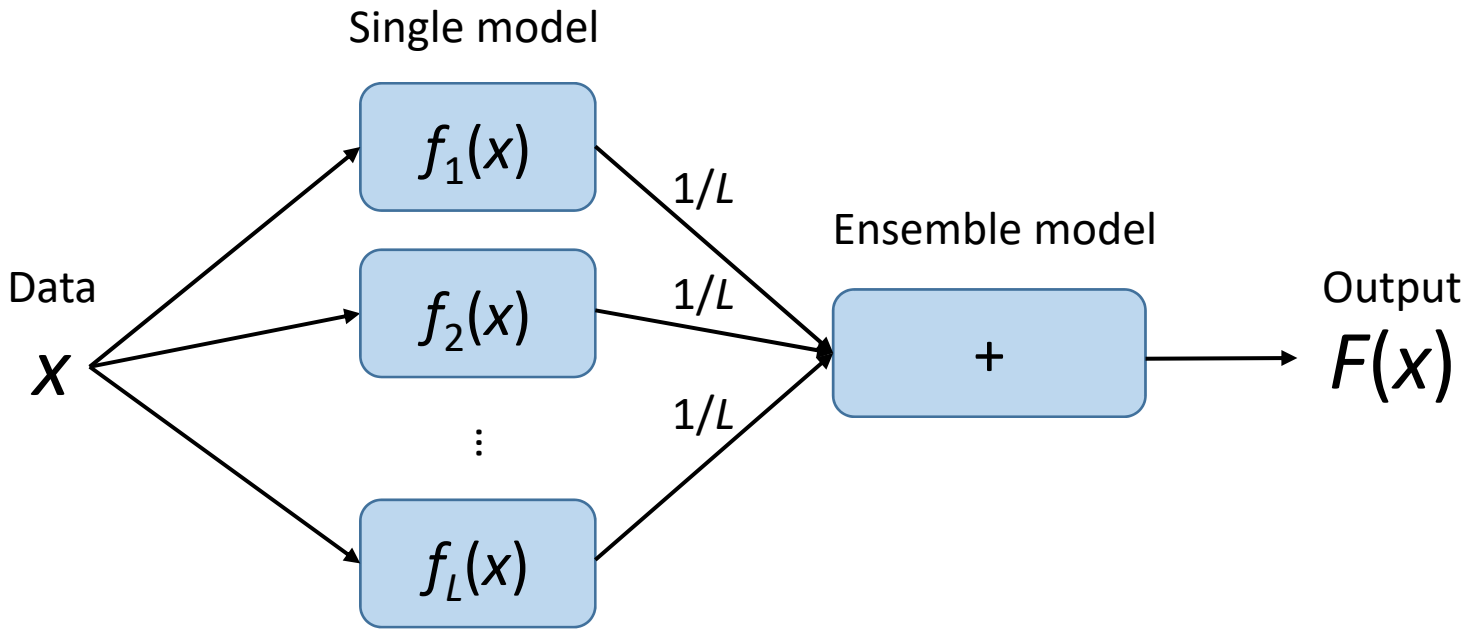
- Consider a set of predictors $f_1, \ldots, f_L$
  - Different predictors have different performance across data
- Idea: construct a predictor $F(x)$ that combines the individual decisions of $f_1, \ldots, f_L$
  - E.g., could have the member predictor vote
  - E.g., could use different members for different region of the data space
  - Works well if the member each has low error rate
- Successful ensembles require diversity
  - Predictors should make different mistakes
  - Encourage to involve different types of predictors

Ensemble Learning



- Although complex, ensemble learning probably offers the most sophisticated output and the best empirical performance!

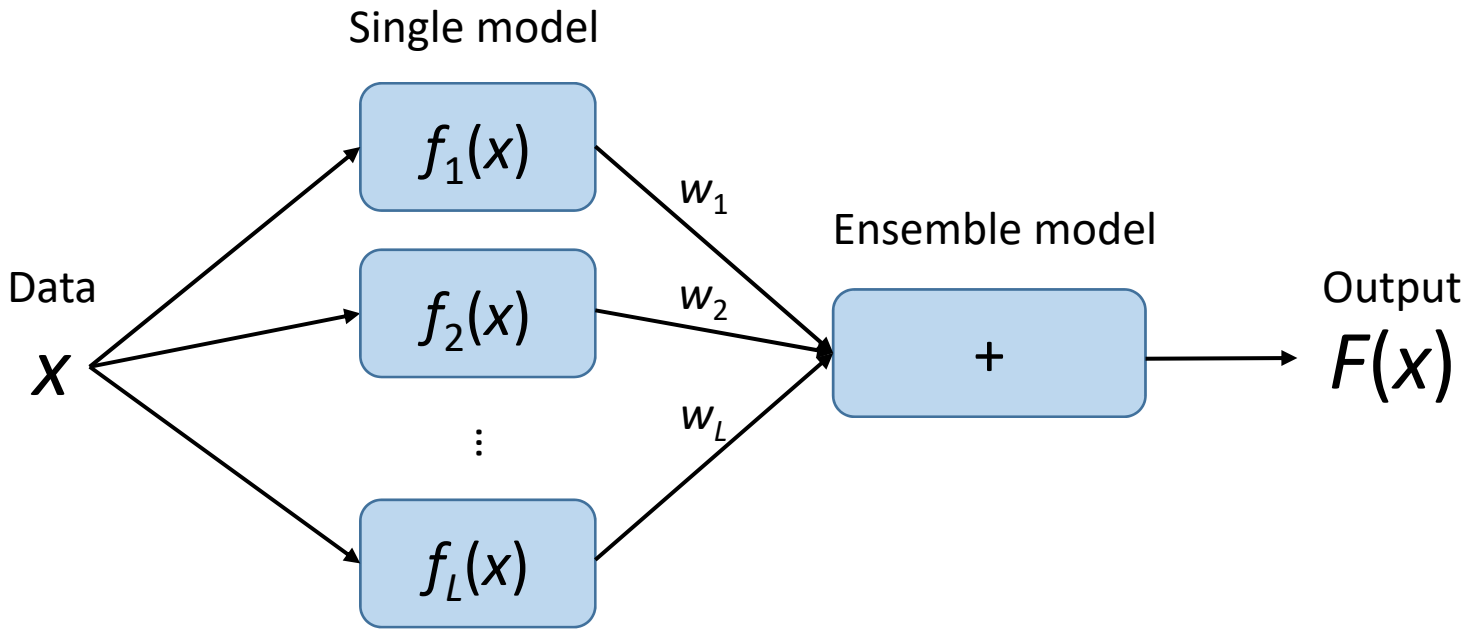
Combining Predictor: Averaging



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

- Averaging for regression; voting for classification

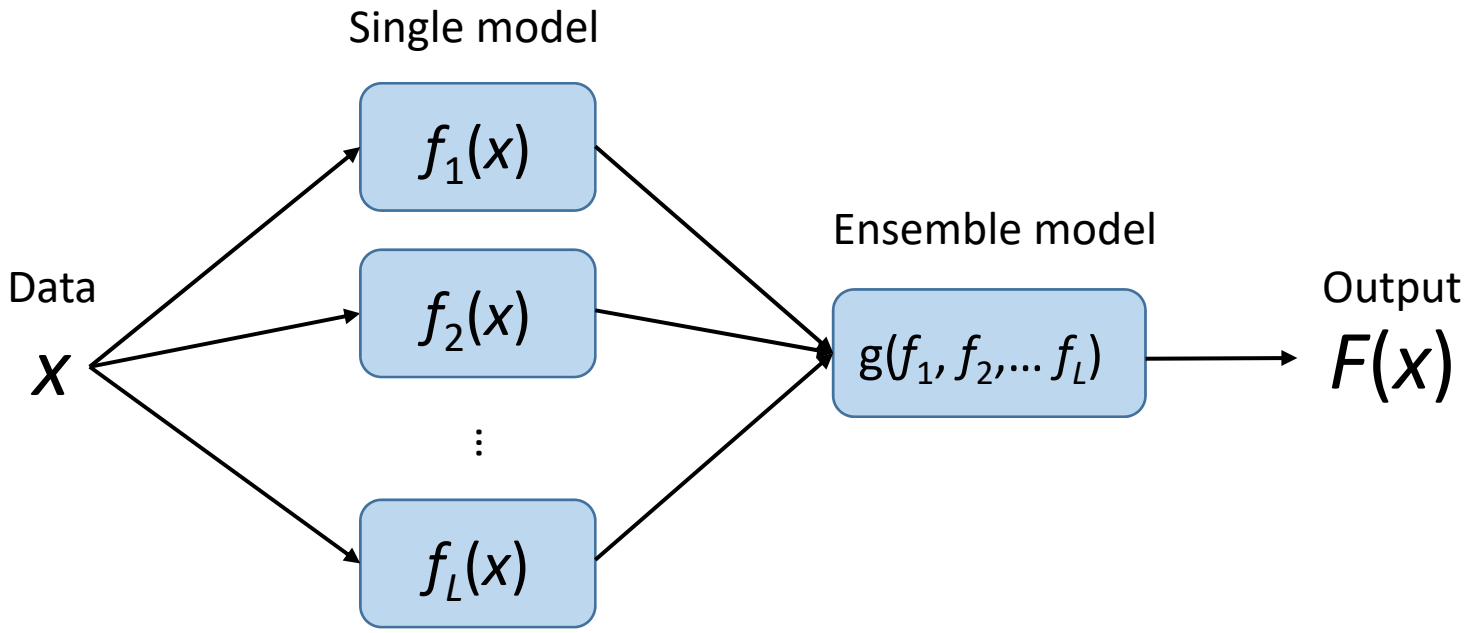
Combining Predictor: Weighted Avg



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

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

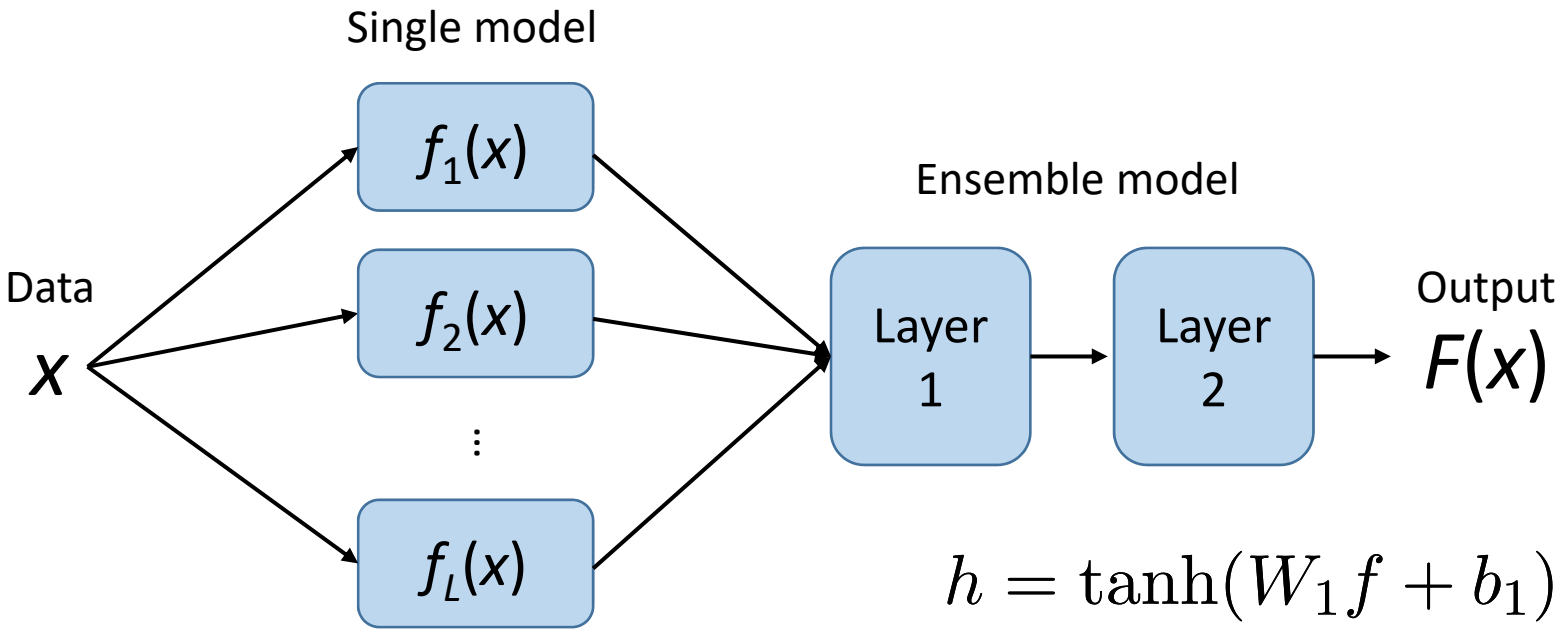
Combining Predictor: Stacking



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

- This is the general formulation of an ensemble

Combining Predictor: Multi-Layer

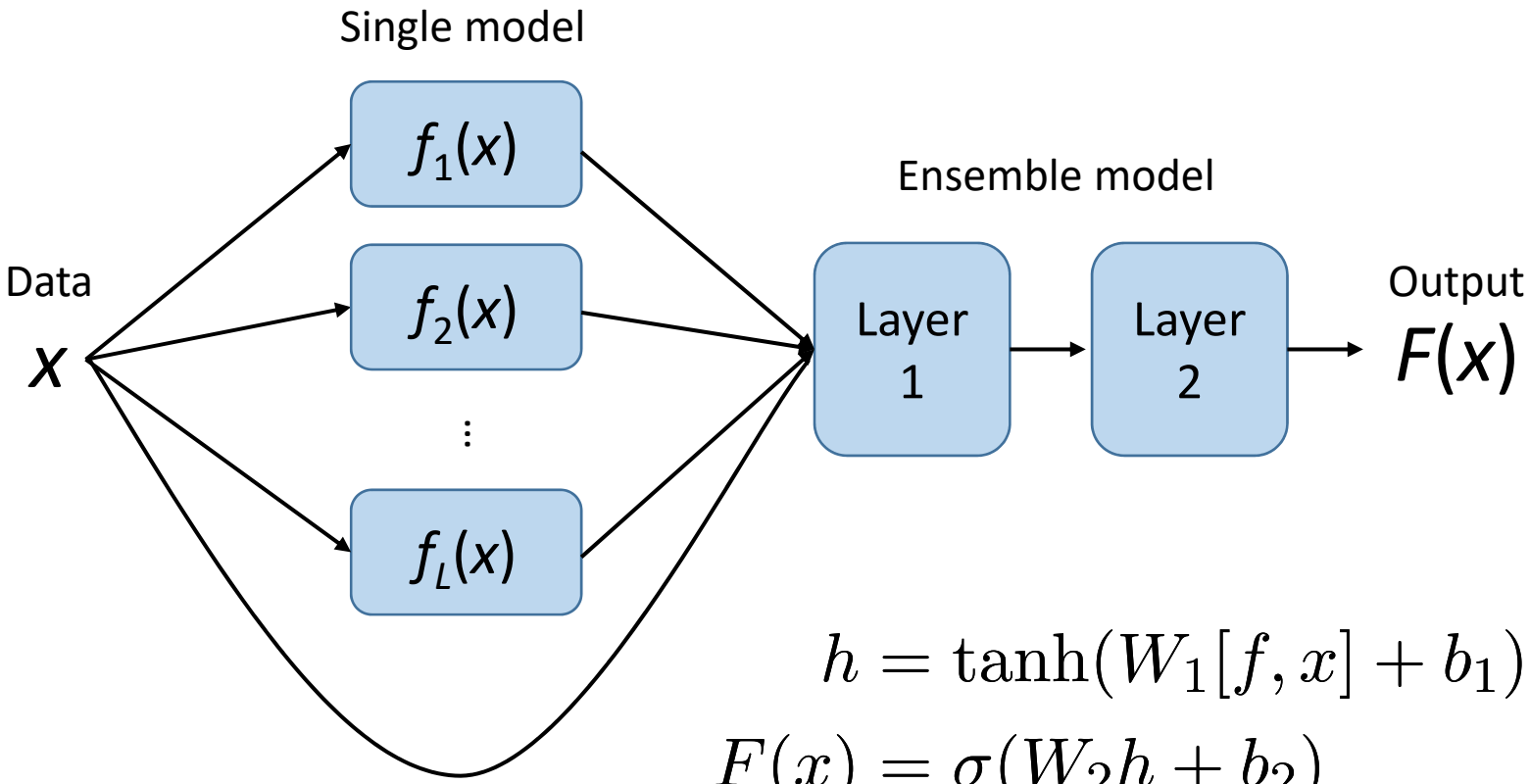


$$h = \tanh(W_1 f + b_1)$$

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

- Use neural networks as the ensemble model

Combining Predictor: Multi-Layer

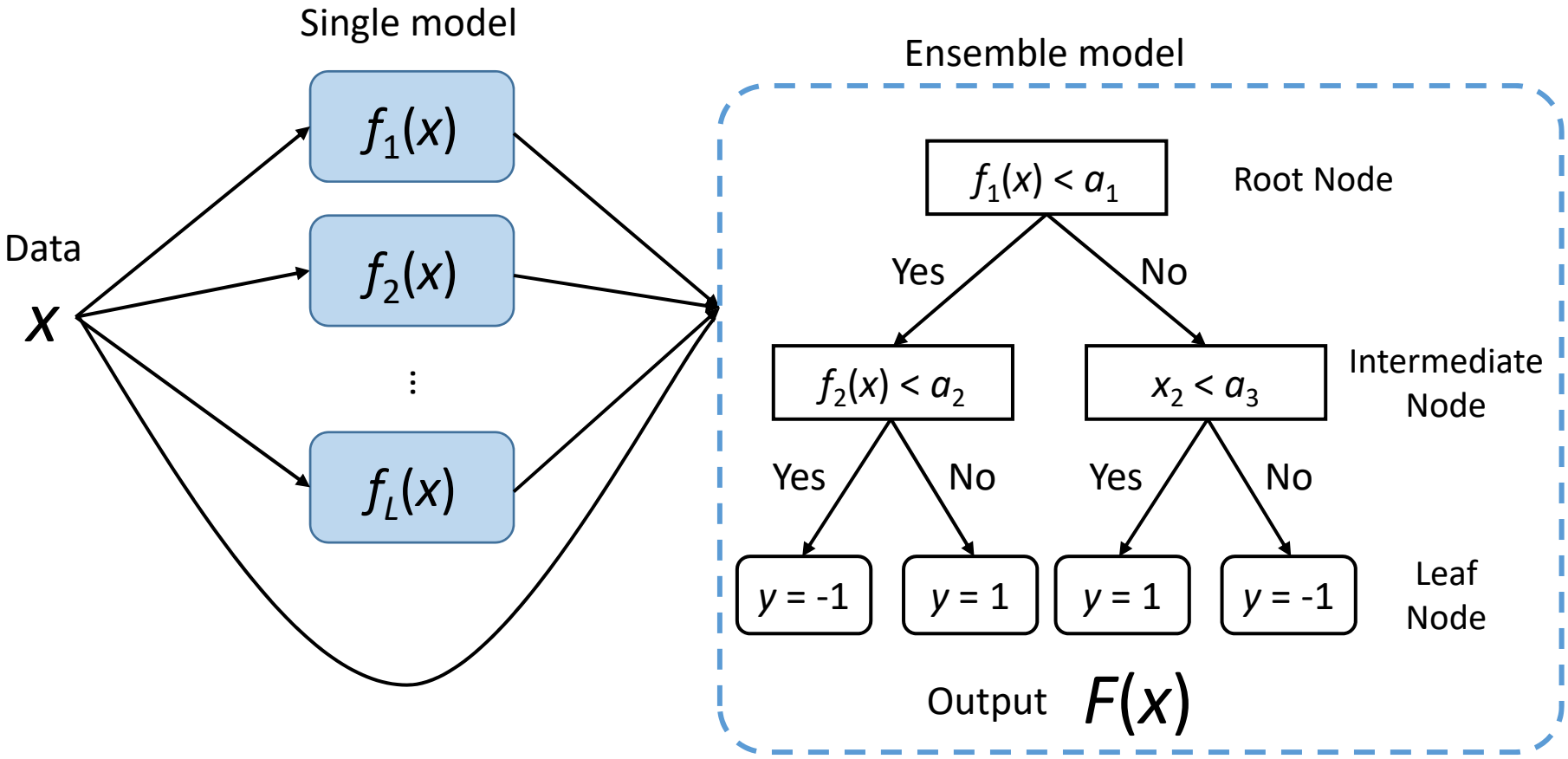


$$h = \tanh(W_1[f, x] + b_1)$$

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

- 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*

Diversity for Ensemble Input

- Successful ensembles require diversity
  - Predictors may make different mistakes
  - Encourage to
    - involve different types of predictors
    - vary the training sets
    - vary the feature sets

Cause of the Mistake	**Diversification Strategy**
Pattern was difficult	Try different models
Overfitting	Vary the training sets
Some features are noisy	Vary the set of input features

[Based on slide by Leon Bottou]

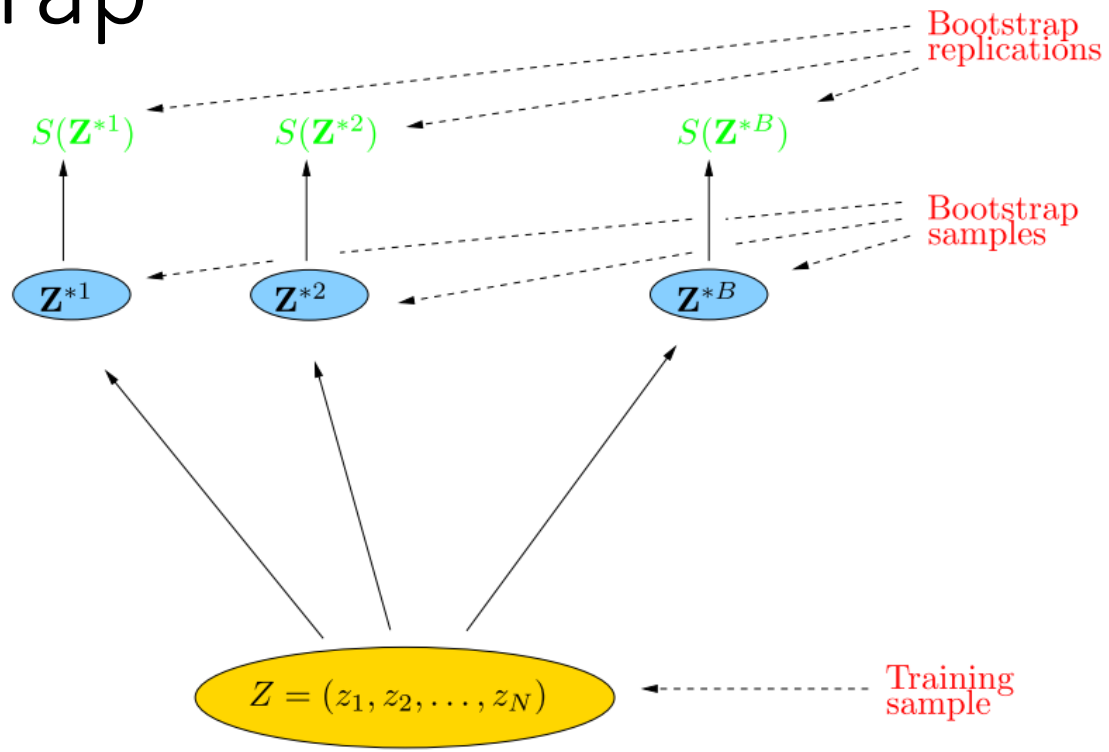
Manipulating the Training Data

- 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

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

- Bagging (Bootstrap Aggregating)
  - 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

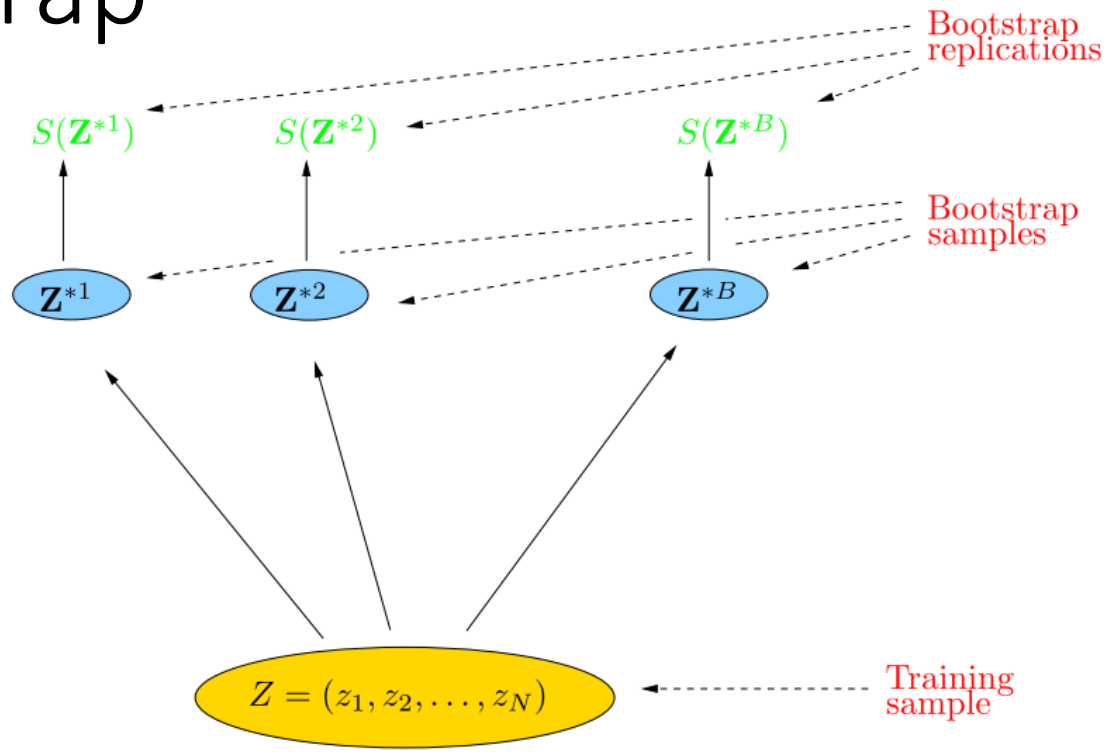
Bootstrap



- Basic idea
  - Randomly draw datasets with replacement from the training data
  - Each replicate with the same size as the training set
  - Evaluate any statistics *S*() over the replicates
    - For example, variance

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

Bootstrap



- Basic idea
  - Randomly draw datasets with replacement from the training data
  - Each replicate with the same size as the training set
  - Evaluate any statistics *S*() over the replicates
    - For example, model error

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

Bootstrap for Model Evaluation

- If we directly evaluate the model using the whole training data

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

- As the probability of a data instance in the bootstrap samples is

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

- If validate on training data, it is much likely to overfit
  - For example in a binary classification problem where *y* is indeed independent with *x*
    - Correct error rate: 0.5
    - Above bootstrap error rate: 0.632*0 + (1-0.632)*0.5=0.184

Leave-One-Out Bootstrap

- Build a bootstrap replicate with one instance *i* out, then evaluate the model using instance *i*

$$\hat{\mathrm{Err}}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}^{*b}(x_i))$$

  - C^{-i} is the set of indices of the bootstrap samples *b* that do not contain the instance *i*
  - For some instance *i*, the set C^{-i} could be null set, just ignore such cases

- We shall come back to the model evaluation and select in later lectures.

Bagging: Bootstrap Aggregating

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

- Bagging average the predictions

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

Random Forest

- 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.

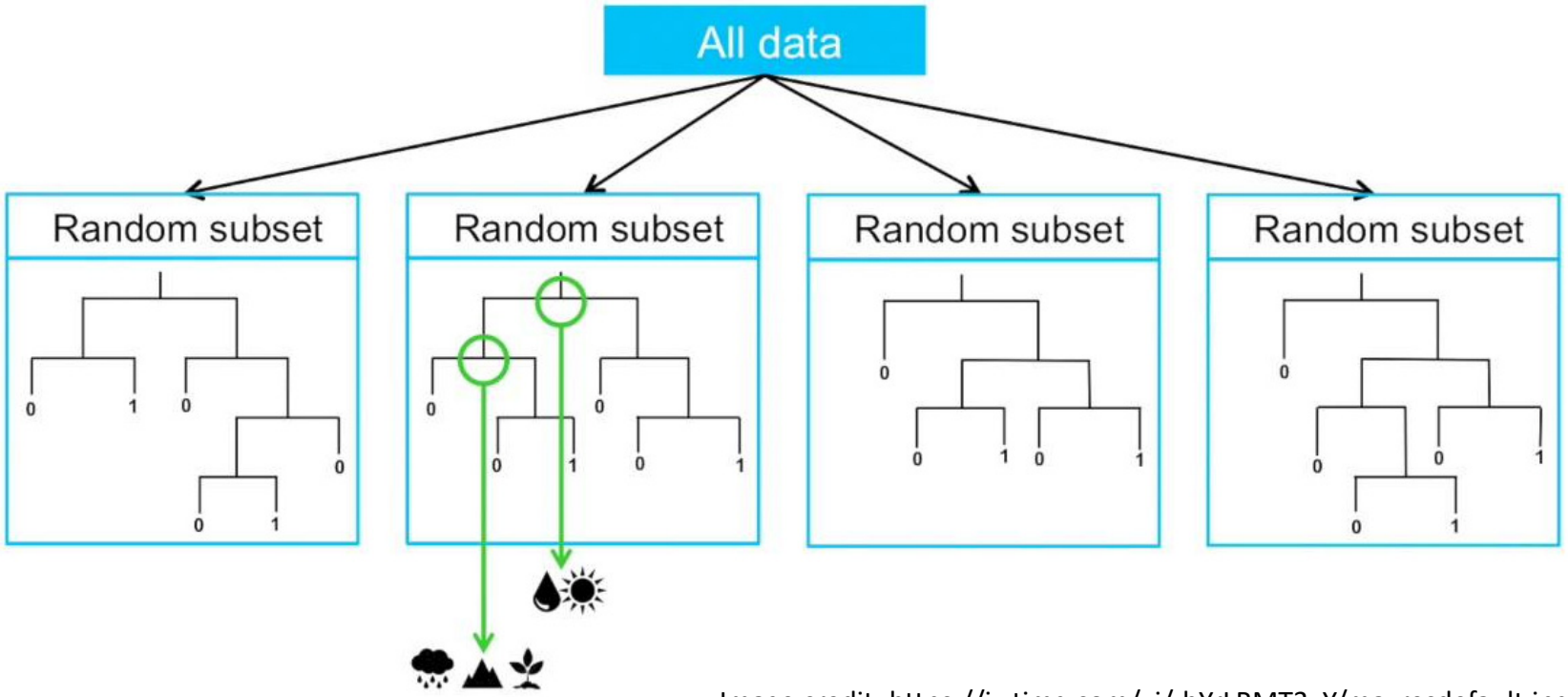
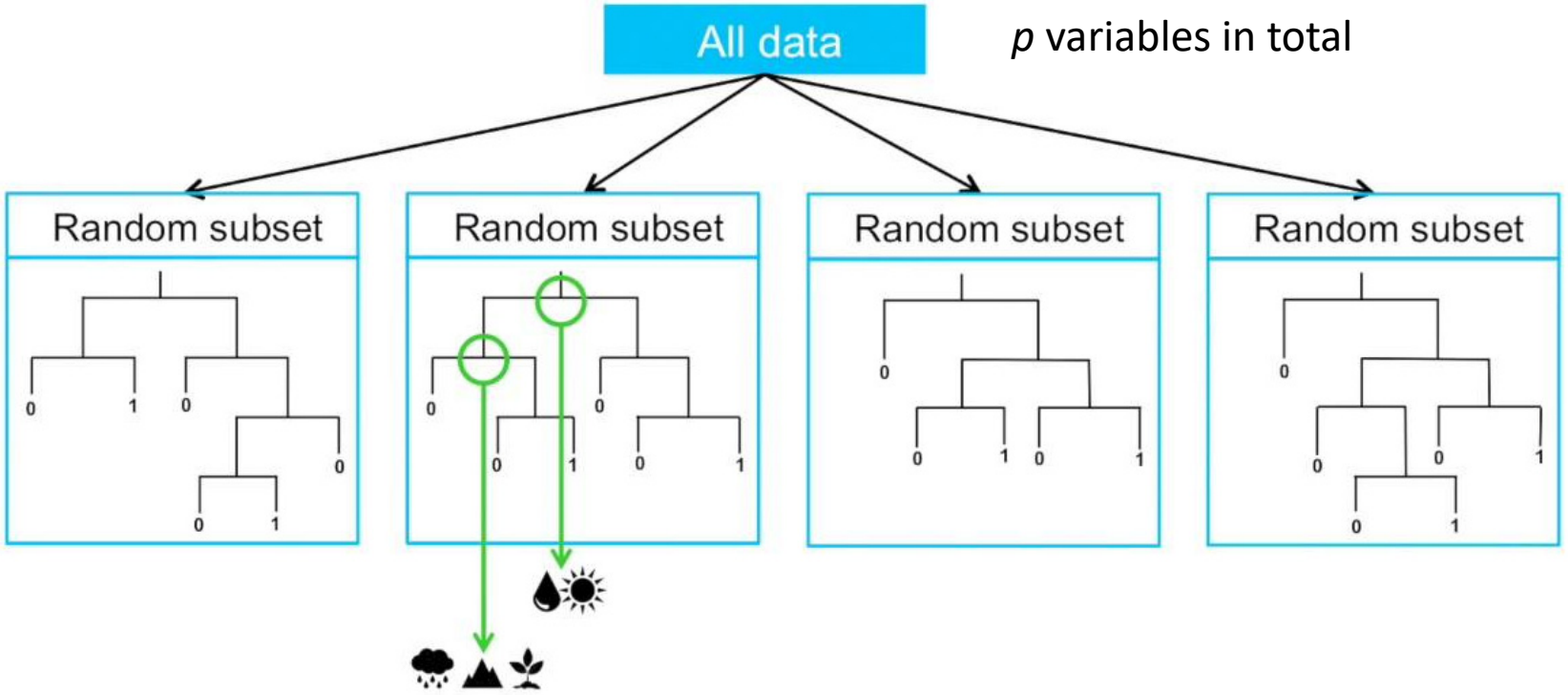


Image credit: https://i.ytimg.com/vi/-bYrLRMT3vY/maxresdefault.jpg

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

Completely random tree

Random Forest Algorithm

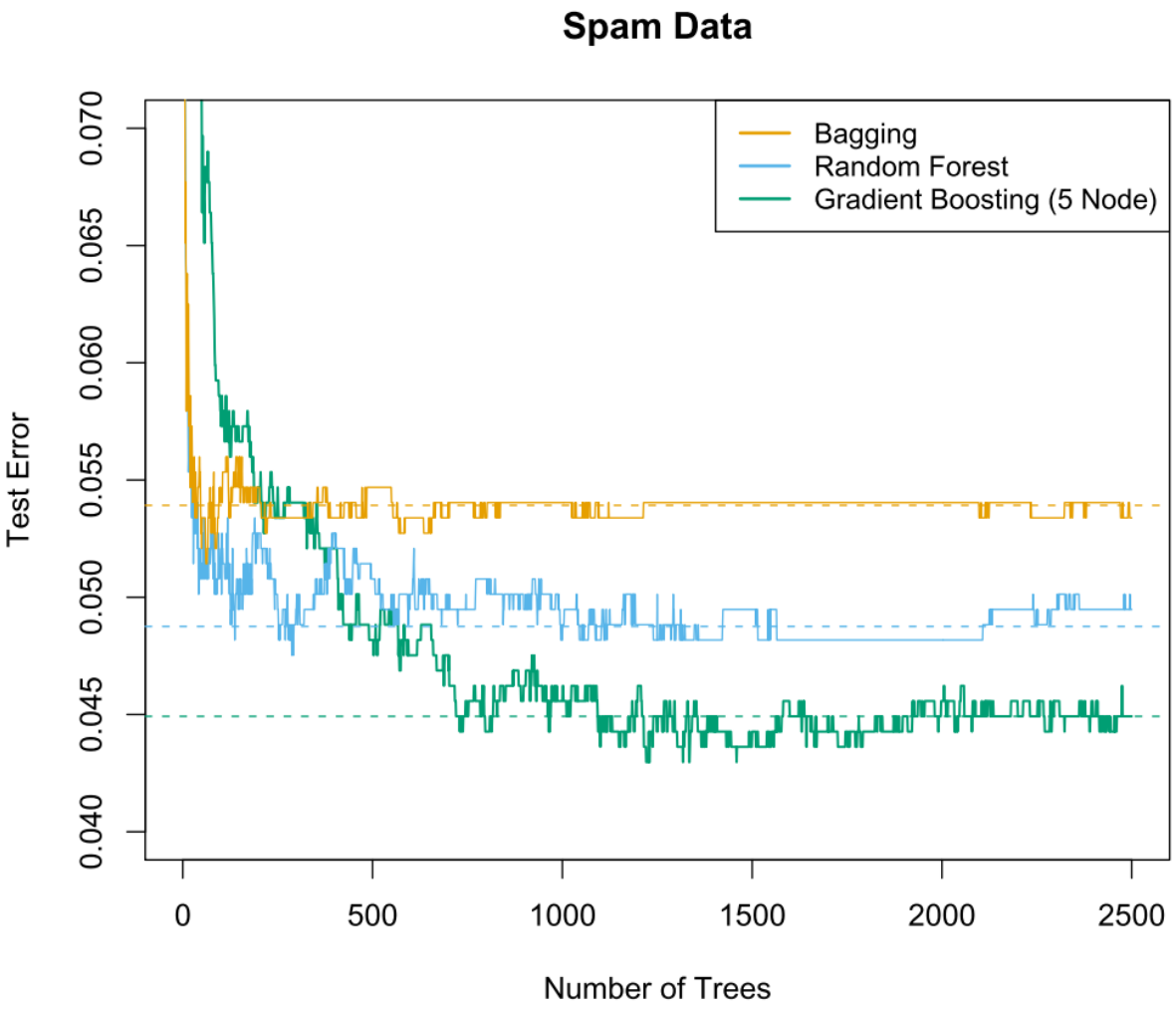
- For b = 1 to B:
  a) Draw a bootstrap sample Z^* of size n from training data
  b) Grow a random-forest tree T_b to the bootstrap data, by recursively repeating the following steps for each leaf node of the tree, until the minimum node size is reached
     I. Select m variables at random from the p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

Performance Comparison



1536 test data instances

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

Bagging vs. Random Forest vs. Boosting

- Bagging (bootstrap aggregating) simply treats each predictor trained on a bootstrap set with the same weight

- Random forest tries to de-correlate the bootstrap-trained predictors (decision trees) by sampling features

- Boosting strategically learns and combines the next predictor based on previous predictors

Additive Models

- General form of an additive model

$$F(x) = \sum_{m=1}^{M} f_m(x)$$

- For regression problem

$$f_m(x) = \beta_m b(x; \gamma_m) \qquad F_M(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m)$$

  - Least-square learning of a predictor with others fixed

$$\{\beta_m, \gamma_m\} \leftarrow \arg\min_{\beta,\gamma} \mathbb{E}\Big[y - \sum_{k \neq m} \beta_k b(x; \gamma_k) - \beta b(x; \gamma)\Big]^2$$

  - Stepwise least-square learning of a predictor with previous ones fixed

$$\{\beta_m, \gamma_m\} \leftarrow \arg\min_{\beta,\gamma} \mathbb{E}\Big[y - F_{m-1}(x) - \beta b(x; \gamma)\Big]^2$$

Additive Regression Models

- Least-square learning of a predictor with others fixed

$$\{\beta_m, \gamma_m\} \leftarrow \arg\min_{\beta,\gamma} \mathbb{E}\Big[y - \sum_{k\neq m} \beta_k b(x;\gamma_k) - \beta b(x;\gamma)\Big]^2$$

  - Essentially, the additive learning is equivalent with modifying the original data as

$$y_m \leftarrow y - \sum_{k\neq m} f_k(x)$$

- Stepwise least-square learning of a predictor with previous ones fixed

$$\{\beta_m, \gamma_m\} \leftarrow \arg\min_{\beta,\gamma} \mathbb{E}\Big[y - F_{m-1}(x) - \beta b(x;\gamma)\Big]^2$$

  - Essentially, the additive learning is equivalent with modifying the original data as

$$y_m \leftarrow y - F_{m-1}(x) = y_{m-1} - f_{m-1}(x)$$

Additive Classification Models

- For binary classification $y = \{1, -1\}$

$$F(x) = \sum_{m=1}^{M} f_m(x)$$

$$P(y = 1|x) = \frac{\exp(F(x))}{1 + \exp(F(x))}$$

$$P(y = -1|x) = \frac{1}{1 + \exp(F(x))}$$

- The monotone logit transformation

$$\log \frac{P(y = 1|x)}{1 - P(y = 1|x)} = F(x)$$

AdaBoost

- For binary classification, consider minimizing the criterion

$$J(F) = \mathbb{E}[e^{-yF(x)}]$$

[proposed by Schapire and Singer 1998 as an upper bound on misclassification error]

- It is (almost) equivalent with logistic cross entropy loss
  - for y=+1 and -1 label

$$\begin{aligned}
\mathcal{L}(y,x) &= -\frac{1+y}{2}\log\frac{e^{F(x)}}{1+e^{F(x)}} - \frac{1-y}{2}\log\frac{1}{1+e^{F(x)}} \\
&= -\frac{1+y}{2}\Big(F(x) - \log(1+e^{F(x)})\Big) + \frac{1-y}{2}\log(1+e^{F(x)}) \\
&= -\frac{1+y}{2}F(x) + \log(1+e^{F(x)}) \\
&= \log\frac{1+e^{F(x)}}{e^{\frac{1+y}{2}F(x)}} = \begin{cases} \log(1+e^{F(x)}) & \text{if } y=-1 \\ \log(1+e^{-F(x)}) & \text{if } y=+1 \end{cases} = \log(1+e^{-yF(x)})
\end{aligned}$$

AdaBoost: an Exponential Criterion

- For binary classification, consider minimizing the criterion

$$J(F) = \mathbb{E}[e^{-yF(x)}]$$

- Solution

$$\mathbb{E}[e^{-yF(x)}] = \int \mathbb{E}[e^{-yF(x)}|x]p(x)dx$$

$$\mathbb{E}[e^{-yF(x)}|x] = P(y=1|x)e^{-F(x)} + P(y=-1|x)e^{F(x)}$$

$$\frac{\partial \mathbb{E}[e^{-yF(x)}|x]}{\partial F(x)} = -P(y=1|x)e^{-F(x)} + P(y=-1|x)e^{F(x)}$$

$$\frac{\partial \mathbb{E}[e^{-yF(x)}|x]}{\partial F(x)} = 0 \quad \Rightarrow \quad F(x) = \frac{1}{2}\log\frac{P(y=1|x)}{P(y=-1|x)}$$

Discrete AdaBoost $f(x) = \pm 1$

- Criterion $J(F) = \mathbb{E}[e^{-yF(x)}]$
- Current estimate $F(x)$
- Seek an improved estimate $F(x) + cf(x)$
- Taylor series

$$f(a+x) = f(a) + \frac{f'(a)}{1!}x + \frac{f''(a)}{2!}x^2 + \frac{f'''(a)}{3!}x^3 + \cdots$$

- With second-order Taylor series

$$\begin{aligned} J(F + cf) &= \mathbb{E}[e^{-y(F(x)+cf(x))}] \\ &\simeq \mathbb{E}[e^{-yF(x)}(1 - ycf(x) + c^2y^2f(x)^2/2)] \\ &= \mathbb{E}[e^{-yF(x)}(1 - ycf(x) + c^2/2)] \end{aligned}$$

Note that $y^2 = 1 \quad f(x)^2 = 1$

Discrete AdaBoost $f(x) = \pm 1$

- Criterion $J(F) = \mathbb{E}[e^{-yF(x)}]$

$$J(F + cf) \simeq \mathbb{E}[e^{-yF(x)}(1 - ycf(x) + c^2/2)]$$

- Solve *f* with fixed *c*

$$\begin{aligned} f = \arg\min_f J(F + cf) &= \arg\min_f \mathbb{E}[e^{-yF(x)}(1 - ycf(x) + c^2/2)] \\ &= \arg\min_f \mathbb{E}_w[1 - ycf(x) + c^2/2|x] \\ &= \arg\max_f \mathbb{E}_w[yf(x)|x] \quad (\text{for } c > 0) \end{aligned}$$

where the weighted conditional expectation

$$\mathbb{E}_w[yf(x)|x] = \frac{\mathbb{E}[e^{-yF(x)}yf(x)]}{\mathbb{E}[e^{-yF(x)}]}$$

The weight is the normalized error factor $e^{-yF(x)}$ on each data instance

Discrete AdaBoost $f(x) = \pm 1$

- Solve *f* with fixed *c*

$$f = \arg\min_f J(F + cf) = \arg\max_f \mathbb{E}_w[yf(x)|x] \quad (\text{for } c > 0)$$

$$\mathbb{E}_w[yf(x)|x] = \frac{\mathbb{E}[e^{-yF(x)} yf(x)]}{\mathbb{E}[e^{-yF(x)}]}$$

Weighted expectation

  - i.e., train an *f*() with each training data instance weighted proportional to its previous error factor $e^{-yF(x)}$

- Solution

$$f(x) = \begin{cases} 1, & \text{if } \mathbb{E}_w(y|x) = P_w(y = 1|x) - P_w(y = -1|x) > 0 \\ -1, & \text{otherwise} \end{cases}$$

Discrete AdaBoost $f(x) = \pm 1$

- Criterion $J(F) = \mathbb{E}[e^{-yF(x)}]$
- Solve *c* with fixed *f*

$$c = \arg\min_c J(F + cf) = \arg\min_c \mathbb{E}_w[e^{-cyf(x)}]$$

$$\begin{aligned}
\frac{\partial \mathbb{E}_w[e^{-cyf(x)}]}{\partial c} &= \mathbb{E}_w[-e^{-cyf(x)} yf(x)] \\
&= \mathbb{E}_w[P(y \neq f(x)) \cdot e^c + (1 - P(y \neq f(x))) \cdot (-e^{-c})] \\
&= \text{err} \cdot e^c + (1 - \text{err}) \cdot (-e^{-c}) = 0 \\
&\Rightarrow c = \frac{1}{2} \log \frac{1 - \text{err}}{\text{err}}
\end{aligned}$$

$$\text{err} = \mathbb{E}_w[1_{[y \neq f(x)]}]$$

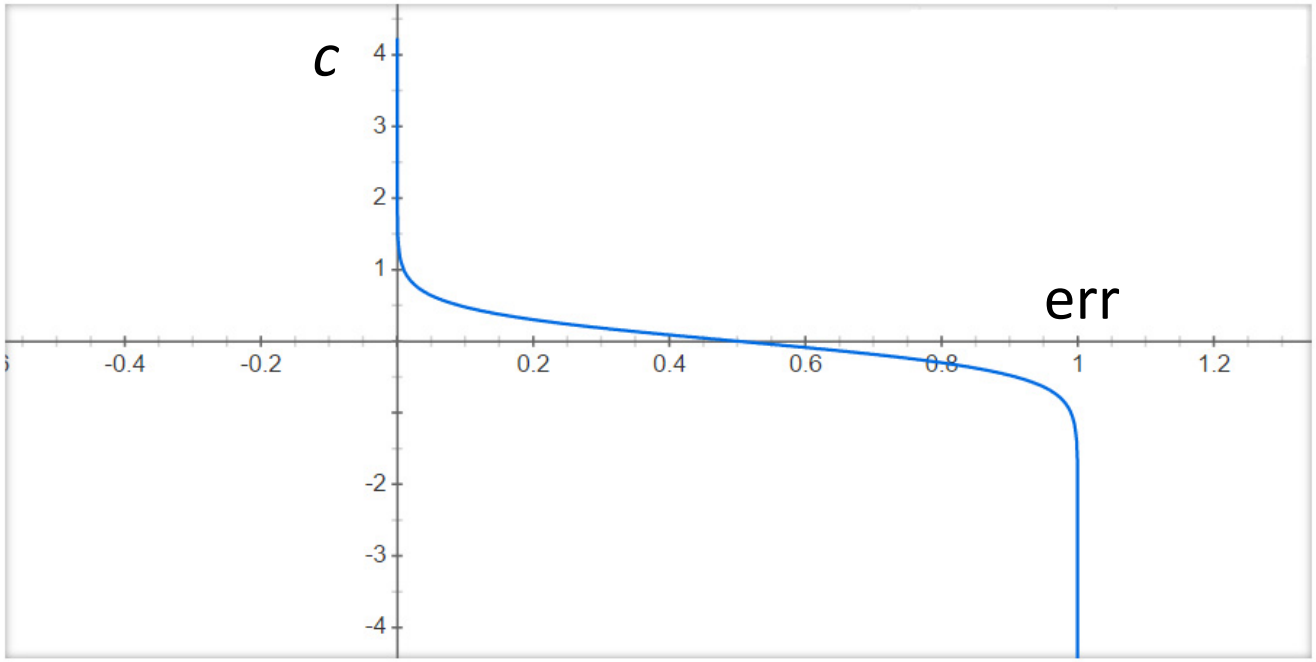
The overall error rate of the weighted instances

Discrete AdaBoost $f(x) = \pm 1$

- Criterion $J(F) = \mathbb{E}[e^{-yF(x)}]$
- Solve *c* with fixed *f*

$$c = \frac{1}{2} \log \frac{1 - \text{err}}{\text{err}}$$

$$\text{err} = \mathbb{E}_w[1_{[y \neq f(x)]}]$$



Discrete AdaBoost Algorithm

Discrete AdaBoost [Freund and Schapire (1996b)]

1. Start with weights $w_i = 1/N, i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$.

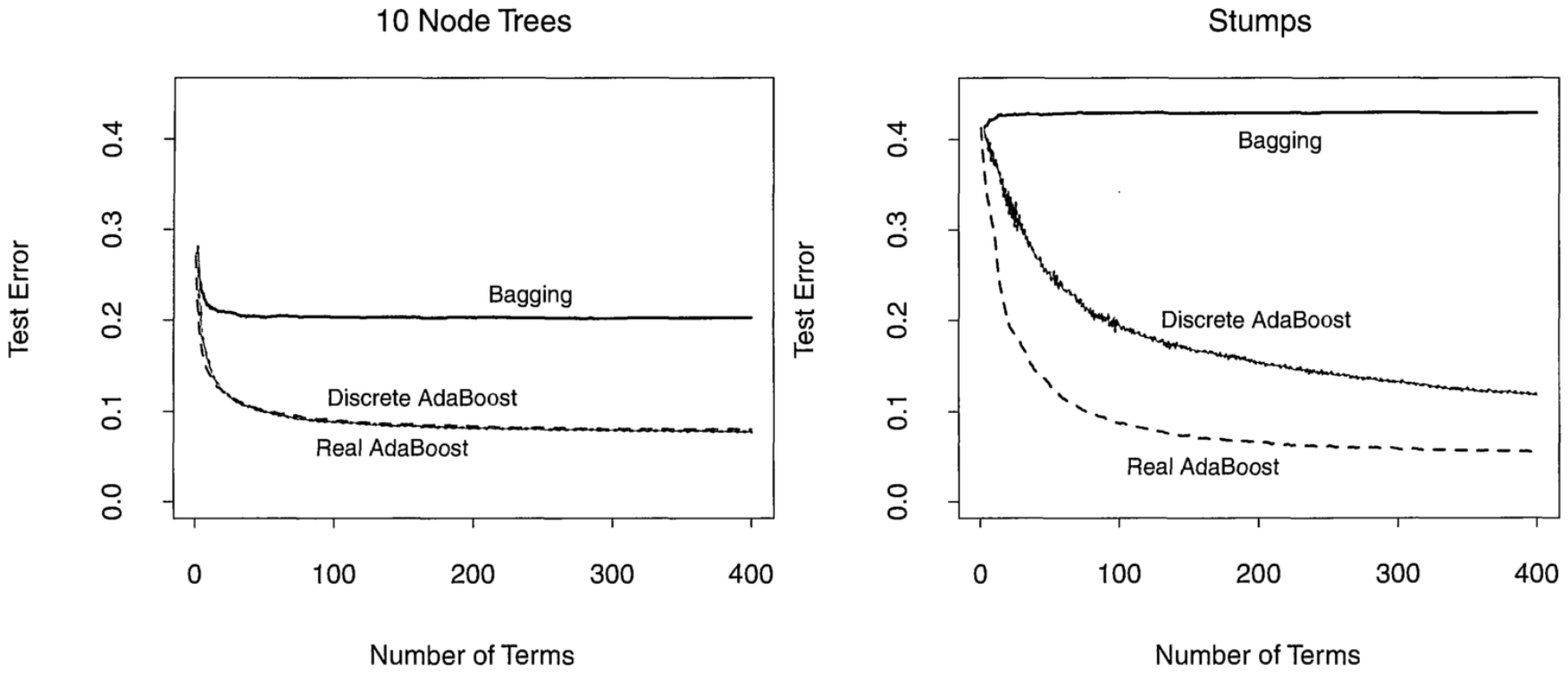
Real AdaBoost Algorithm

Real AdaBoost

1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier to obtain a class probability estimate $p_m(x) = \hat{P}_w(y = 1|x) \in [0, 1]$, using weights w_i on the training data.
   (b) Set $f_m(x) \leftarrow \frac{1}{2} \log p_m(x)/(1 - p_m(x)) \in R$.
   (c) Set $w_i \leftarrow w_i \exp[-y_i f_m(x_i)]$, $i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} f_m(x)]$.

- Real AdaBoost uses class probability estimates $p_m(x)$ to construct real-valued contributions $f_m(x)$.

Bagging vs. Boosting



- Stump: a single-split tree with only two terminal nodes.