Ensemble methods

Generalization error

- Treat $\tilde{y}(x|S)$ as a random function
  - Depends on training data S
- $\mathcal{L} = E_S[\, f(\tilde{y}(x|S), y)]$
  - Expected generalization error
  - Over the randomness of S
  - One loss function is squared error : $\frac{1}{N}\Sigma(\tilde{y}-y)^2$

Bias/Variance Tradeoff

- Consider one data point (x,y)

  Error: $\xi = \hat{y}(x|S) - y$

  Mean error: $\bar{\xi} = E_S[\xi]$ = **Average error**

$$E_S[(\xi - \bar{\xi})^2] = E_S[\xi^2 - 2\xi\bar{\xi} + \bar{\xi}^2]$$
$$= E_S[\xi^2] - 2E_S[\xi]\bar{\xi} + \bar{\xi}^2$$
$$= E_S[\xi^2] - \bar{\xi}^2$$

Expected Error SSE

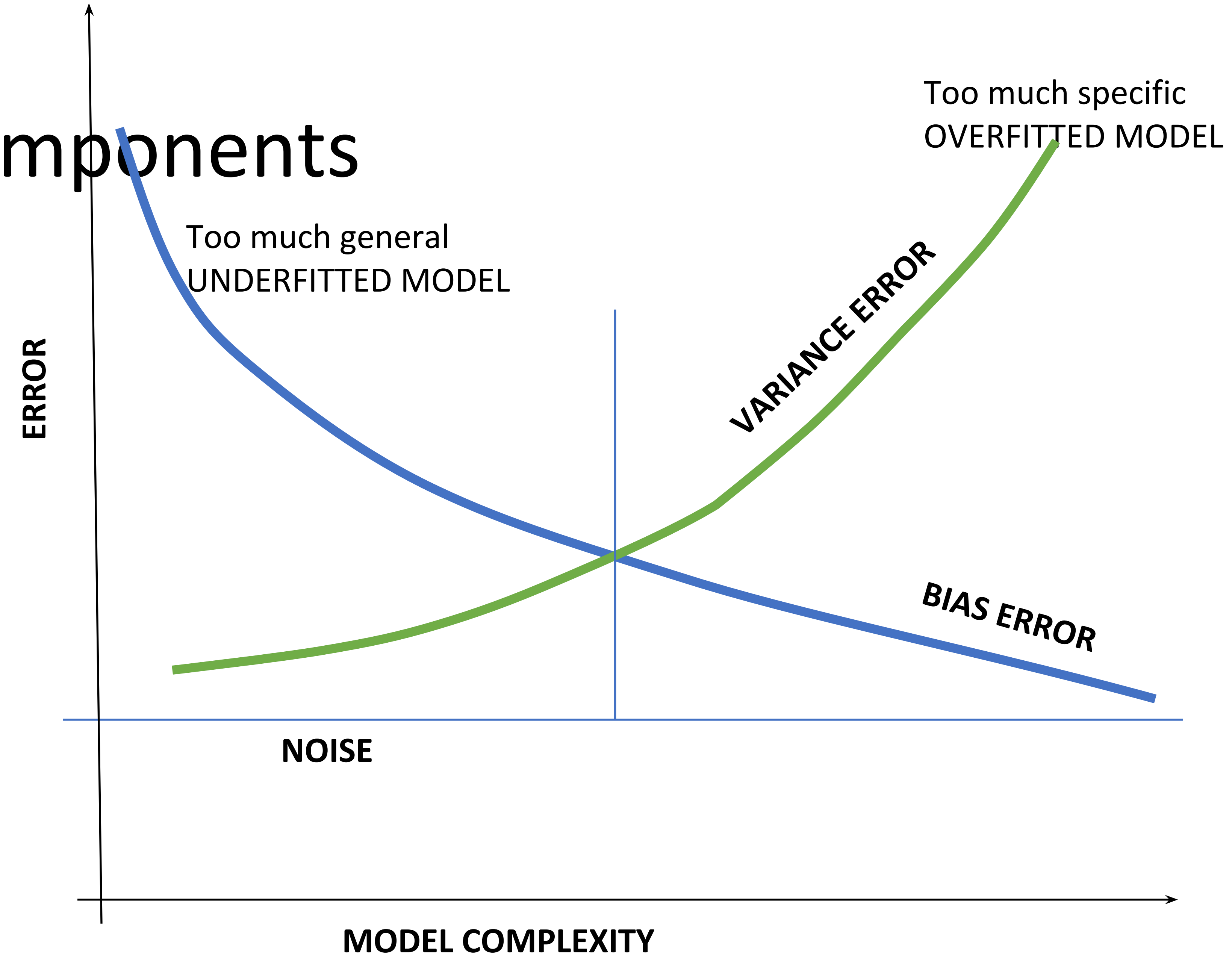
$$E_S[f(\tilde{y}(x|S),y)] = E_S[\xi^2]$$
$$= \underbrace{E_S[(\xi - \bar{\xi})^2]}_{\text{Variance}} + \underbrace{\bar{\xi}^2}_{\text{Bias}}$$

Bias/Variance for all (x,y) is expectation over P(x,y).

Can also incorporate measurement noise, data entry errorb.

Variance **Bias**

Error components



Bias/Variance Tradeoff

- **Ensemble methods that minimize variance**
  - **Bagging – Bootstrap Aggregation**
  - **Random Forests**

- Ensemble methods that **minimize bias**
  - Boosting - ADABOOST
  - Ensemble Selection – choice of classifier

Different Classifiers with variety

- Different populations (datasets)
- Different features
- Different modelling techniques
- Different initial seeds (choosing sub-populations, transforming data etc)

Bagging= Bootstrap Aggregation

S

Person	Age	Male?	Height > 55″
Alice#	14#	0#	1#
Bob#	10#	1#	1#
Carol#	13#	0#	1#
Dave#	8#	1#	0#
Erin#	11#	0#	0#
Frank#	9#	1#	1#
Gena#	8#	0#	0#

S’

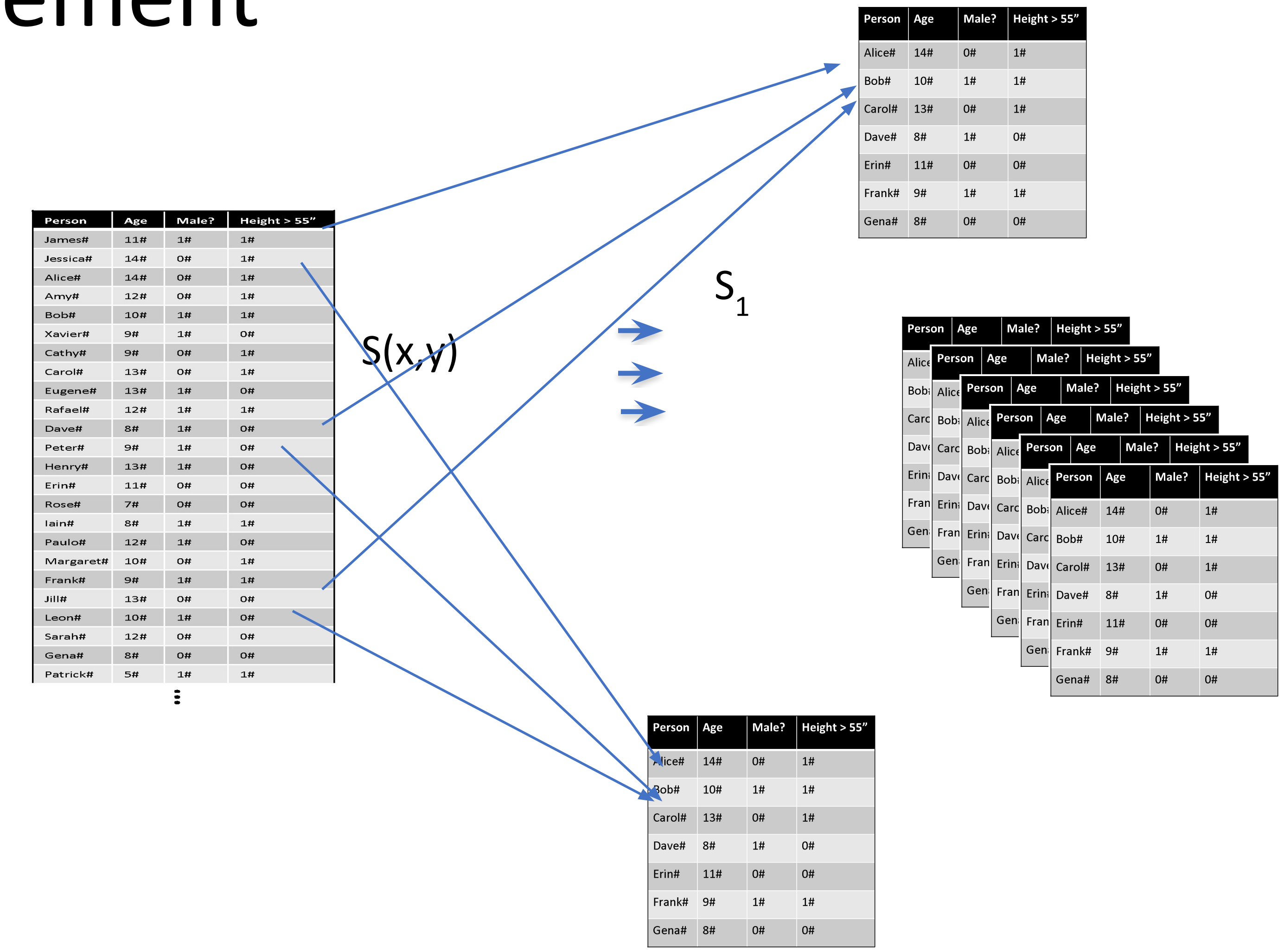
Person	Age	Male?	Height > 55″
Alice#	14#	0#	1#
Bob#	10#	1#	1#
Carol#	13#	0#	1#
Dave#	8#	1#	0#
Erin#	11#	0#	0#
Frank#	9#	1#	1#
Gena#	8#	0#	0#

from S

- **Goal:** reduce variance
- From original population, create subpopulations $\{S_i\}$ by random sampling with replacement
  - Train model using each S’
  - Average for regression / Majority vote for classification

Variance reduces sub-linearly
Bias often increases slightly

Bagging – Random sampling with replacement



Bagging

- Reduces overfitting (variance error reduces)
- Bias error increases slightly, but variance reduces a lot
- Normally uses one type of classifier
- Decision trees are popular
- Easy to parallelize
- A large number of simple DT, not pruned

Random Forests – forest of stubs

- **Goal:** reduce variance
  - Bagging resamples training data
- Random Forest samples data as well as features
  - Sample S’
  - Train DT
    - At each node, sample features (sqrt)
  - Average regression /vote classifications
- Suited only for Decision Trees

Two-way - random division

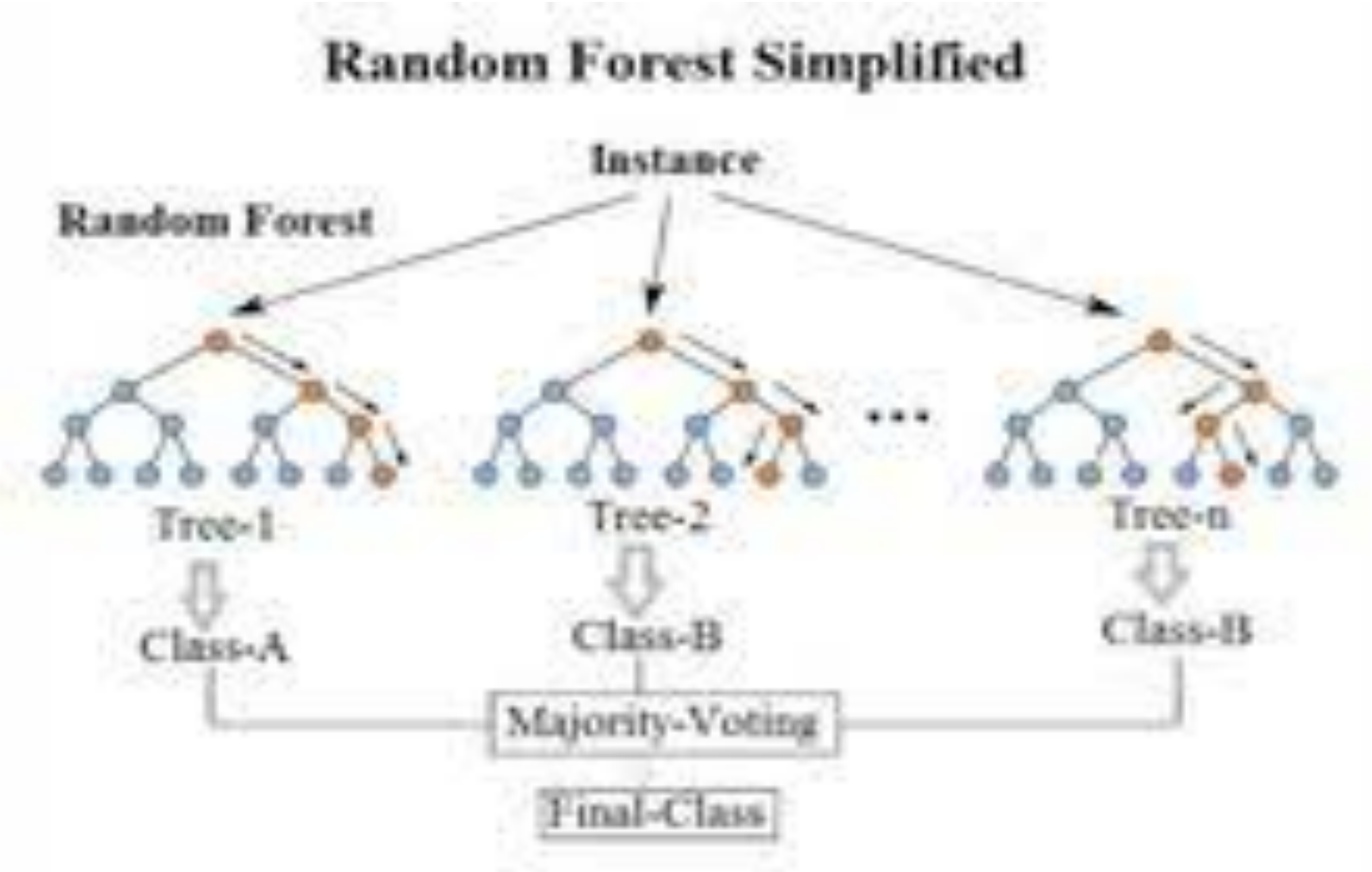
	A1	A2	A3	A4	A5	A6	A7	A8	A9
1									
2									
3									
4									
5									
6									
…									

	A2	A6	A9
1			
5			
…			

	A1	A8	A3
6			
2			
…			

	A5	A4	A7
3			
4			
…			

Random Forest



Boosting

Boosting works very differently.

1. No bootstrap sampling – all data samples used each time sequentially
2. Next Tree derives from previous Tree

ADABOOST

- Short for Adaptive Boosting
- Each time entire data is used to train the model (unlike Bagging /RF)
- Training happens repeatedly in a sequence (unlike Bagging/RF and akin to cross validation)
- Results in a long tree (unlike Bagging/RF)

ADABOOST

- Difference is that each data point has a different weight after each training-testing cycle
- They start with equal weights
- After training-testing, Misclassified examples are given more weight to increase their learning
- This ensures that currently misclassified data points have greater probability of getting correctly classified in future

ADABOOST

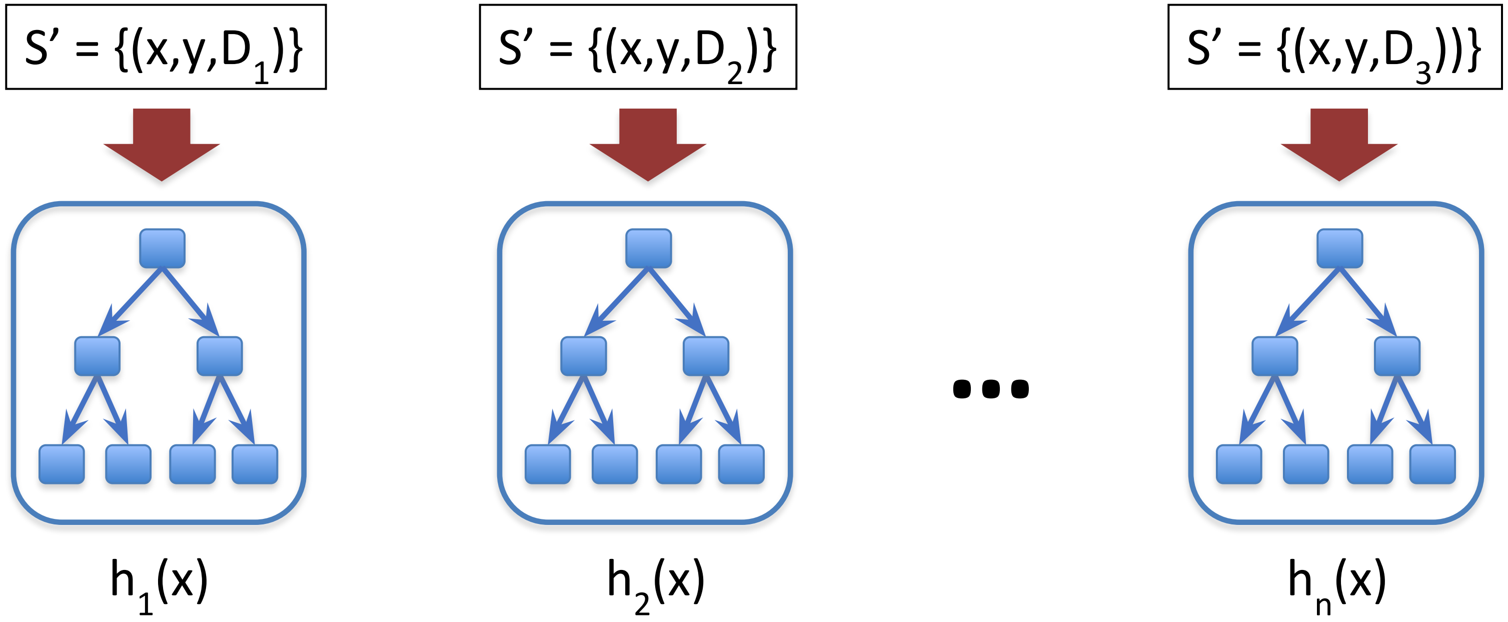
- Another difference is that the output from each training cycle is assigned a weight
- Greater the accuracy, more is the weight of output
- Final decision, for a given data point, is a weighted combination of the predicted output obtained at each stage

Terminology

- S : Initial population with equal weights for all data points
- S': Next populations with changed weights
- x: Feature values of a row (entity)
- y: Actual output /label of the row
- $\widetilde{y} = h(x_i)$ Predicted output /label for the row
- D_i: weight of a data point at i^{th} iteration of adaboost

Boosting (AdaBoost)

$$h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \ldots + \alpha_n h_n(x) = \text{real value}$$



D – weight distribution on data points
α – weight of linear combination

Stop when validation performance reaches plateau

Given Data points:

$\{(x_1,y_1)(x_2,y_2)\ldots.(x_m,y_m)\}$, $x \in \chi$, $y \in \{-1, +1\}$

Initial Distribution of data point weights is uniform

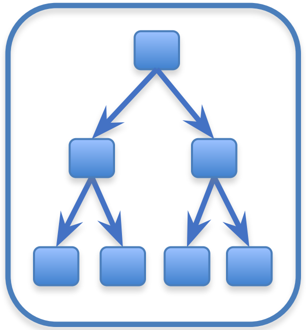
$D(i) = 1/m \quad i \in 1\ldots.m$

X1	X2	X3	y	weight
1.2	A	3	+1	1/m
1.4	A	6	-1	1/m
4.6	C	5	+1	1/m
5.8	D	9	+1	1/m

Now train DT with initial distribution

Get hypothesis h_1

Get predicted outputs on training and validation data



Calculate probability of error for each data point:

$\varepsilon_1(x_i, y_i, 1/m) = Pr_{i \in U}(\, h_1(x_i) \neq y_i)$

Calculate log odds – coefficient of model α_1

$$\boxed{\propto_1 = \frac{1}{2} \ln \frac{1 - \varepsilon_1}{\varepsilon_1}}$$

Now update Distribution:

$\tilde{y}$

$$D_2(i) = \frac{D_1(i) \times e^{-\alpha_1 h(x_i\ ).y_i}}{Z_t}$$

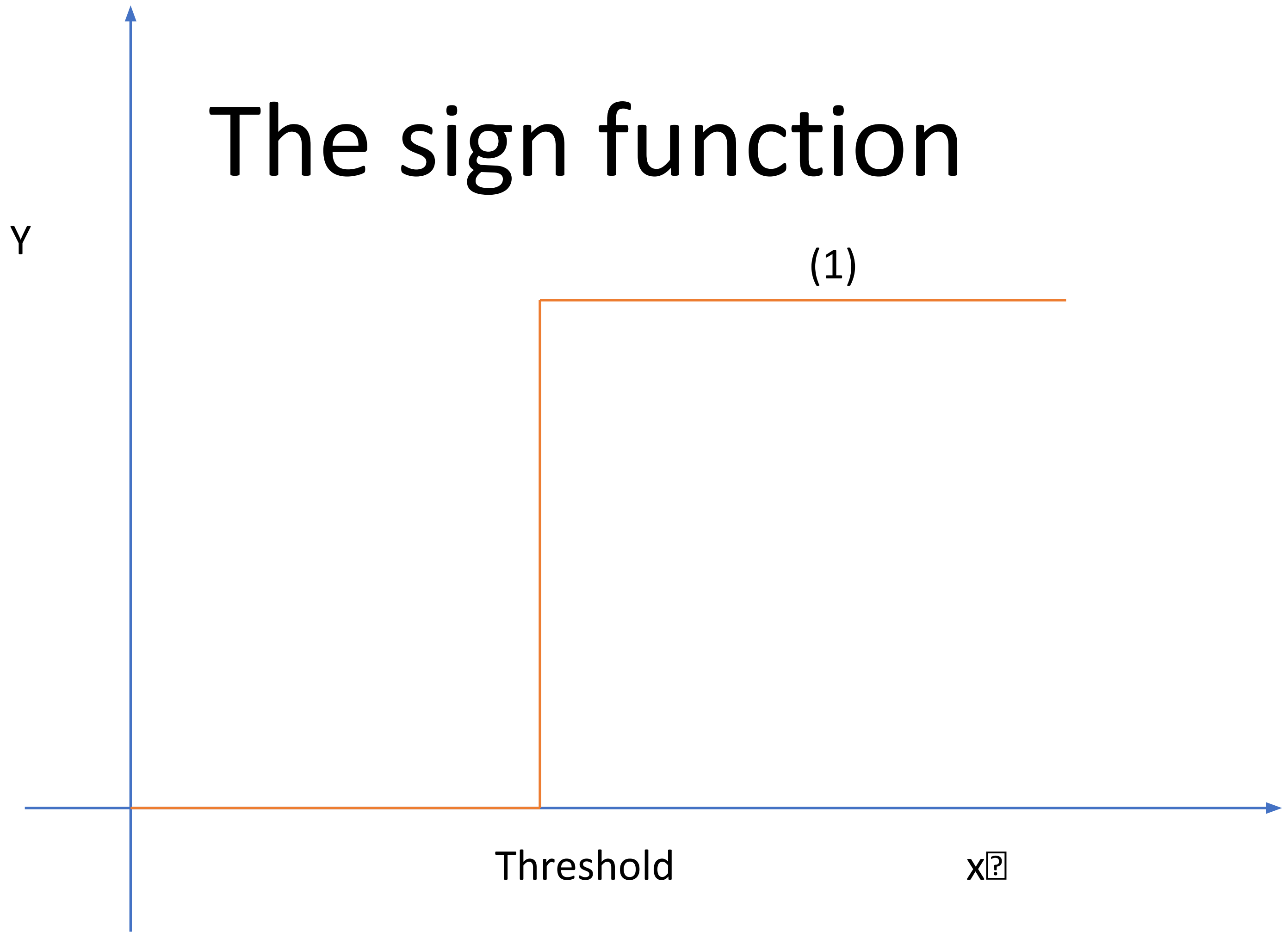
Here Z is a normalization factor to make distribution

- After all Trees are generated sequentially, find overall prediction

$$H(x) = Sign\left(\sum_{t=1}^{n} \propto_t h_t(x)\right)$$

Prediction error decays exponentially

The sign function



Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.

Initial Distribution of Data

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.

Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$. Initial Distribution of Data

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t. Train model
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$. Initial Distribution of Data
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$. Train model
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

Error of model

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.

Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.

Initial Distribution of Data

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

Train model

$$\varepsilon_t = \Pr_{i \sim D_t}\left[h_t(x_i) \neq y_i\right].$$

Error of model

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$.

Coefficient of model – log odds

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathscr{X}$, $y_i \in \{-1, +1\}$.

Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$. Initial Distribution of Data

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathscr{X} \to \{-1, +1\}$. Train model
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

Error of model

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$. Coefficient of model
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Update Distribution

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$. Initial Distribution of Data
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$. Train model
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

Error of model

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$. Coefficient of model
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Update Distribution

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Final average for x

Theorem: training error drops exponentially fast

Method	Minimize Bias?	Minimize Variance?	Other Comments
Bagging	No. Slightly increases bias. Overall Mean square error is low due to larger reduction in variance	Yes. Bootstrap aggregation (resampling training data)	1. Can be parallelized 2. Used for combination of DTs and other models such as SVM/Log Reg. 3. No need for pruning
Random Forests	-do-	Yes. Bootstrap aggregation + bootstrapping features	1. Mainly for decision trees. 2. Can be parallelized 3. No need for pruning
Gradient Boosting (AdaBoost)	Yes. Optimizes training performance by focusing on misclassified examples.	Can reduce variance, if each DT is pruned after training	1. Cannot be parallelized – sequential 2. Change weight distribution of data each time 3. Determines which model to add at run-time.