10-301/601: Introduction to Machine Learning Lecture 24 – Ensemble Methods

Matt Gormley & Henry Chai

4/14/25

Front Matter

- Announcements
  - HW8 released 4/8, due 4/16 at 11:59 PM

Recall: AdaBoost

- Intuition: iteratively reweight inputs, giving more weight to inputs that are difficult-to-predict correctly
- Analogy:
  - You all have to take a test (😱) …
  - … but you're going to be taking it one at a time.
  - After you finish, you get to tell the next person the questions you struggled with.
  - Hopefully, they can cover for you because…
  - … if "enough" of you get a question right, you'll all receive full credit for that problem

AdaBoost

- Input: $\mathcal{D}\ \left(y^{(n)} \in \{-1, +1\}\right), T$
- Initialize data point weights: $\omega_0^{(1)}, \dots, \omega_0^{(N)} = \frac{1}{N}$
- For $t = 1, \dots, T$
  1. Train a weak learner, h_t, by minimizing the *weighted* training error
  2. Compute the *weighted* training error of h_t:
     $$\epsilon_t = \sum_{n=1}^{N} \omega_{t-1}^{(n)} \mathbb{1}\left(y^{(n)} \neq h_t\left(\boldsymbol{x}^{(n)}\right)\right)$$
  3. Compute the **importance** of h_t:
     $$\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$
  4. Update the data point weights:
     $$\omega_t^{(n)} = \frac{\omega_{t-1}^{(n)}}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ if } h_t\left(\boldsymbol{x}^{(n)}\right) = y^{(n)} \\ e^{\alpha_t} \quad \text{if } h_t\left(\boldsymbol{x}^{(n)}\right) \neq y^{(n)} \end{cases}$$

$Z_t = \sum_{n=1}^{N} \omega_{t-1}^{(n)} \begin{cases} e^{-\alpha_t} \\ e^{\alpha_t} \dots \end{cases}$

- Output: an aggregated hypothesis

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

Setting α_t

α_t determines the contribution of h_t to the final, aggregated hypothesis:

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

Intuition: we want good weak learners to have high importances

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

Setting α_t

α_t determines the contribution of h_t to the final, aggregated hypothesis:

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

Intuition: we want good weak learners to have high importances

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

Poll Question 1:

How does the importance of a very bad/mostly incorrect weak learner compare to the importance of a very good/mostly correct weak learner?

A. Similar magnitude, same sign **(TOXIC)**
B. Similar magnitude, different sign
C. Different magnitude, same sign
D. Different magnitude, different sign

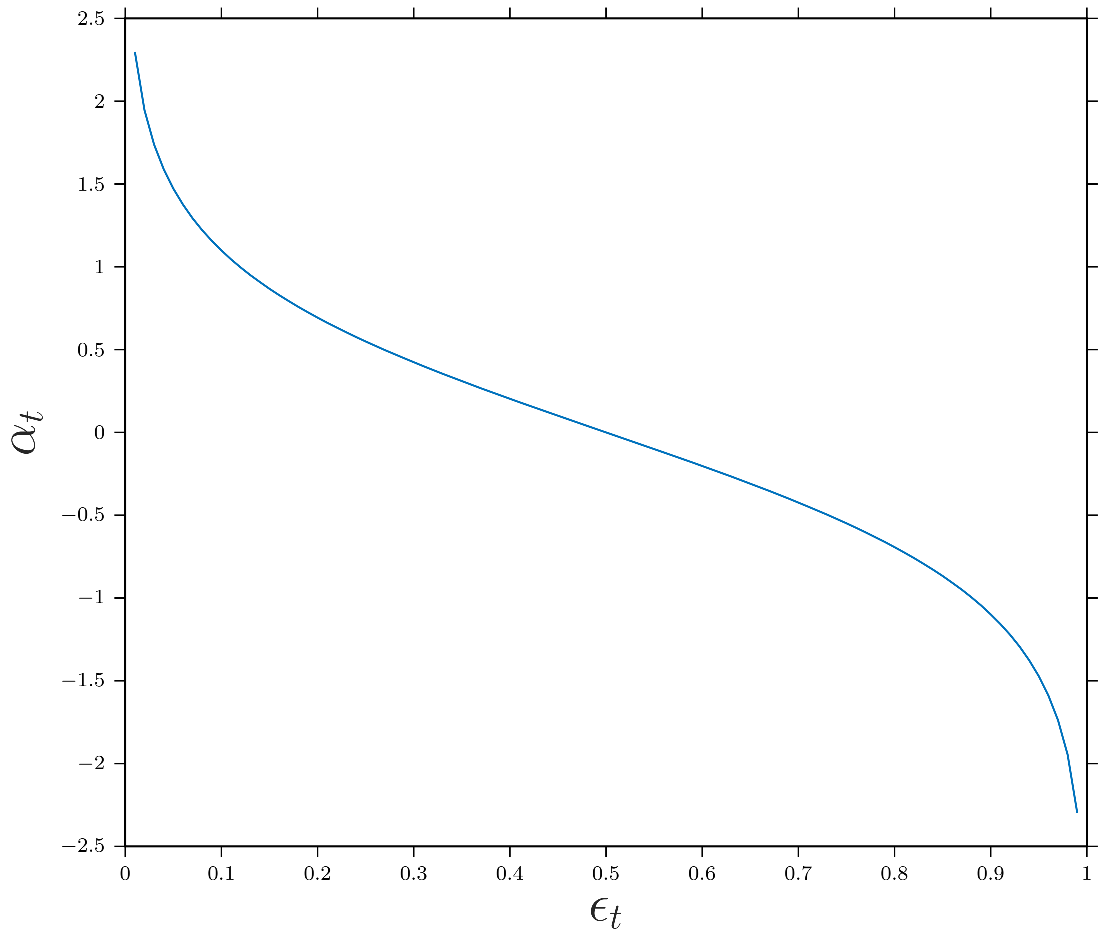
Setting α_t

α_t determines the contribution of h_t to the final, aggregated hypothesis:

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

Intuition: we want good weak learners to have high importances

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$



Updating $\omega^{(n)}$

- Intuition: we want incorrectly classified inputs to receive a higher weight in the next round

$$\omega_t^{(n)} = \frac{\omega_{t-1}^{(n)}}{Z_t} \times \begin{cases} e^{-\alpha_t} \text{ if } h_t\left(\boldsymbol{x}^{(n)}\right) = y^{(n)} \\ e^{\alpha_t} \text{ if } h_t\left(\boldsymbol{x}^{(n)}\right) \neq y^{(n)} \end{cases}$$

$$\sum_{n=1}^{N} w_t^{(n)}$$

- If $\epsilon_t < 1/2 \Rightarrow \frac{1-\epsilon_t}{\epsilon_t} > 1$

$\Rightarrow \alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$

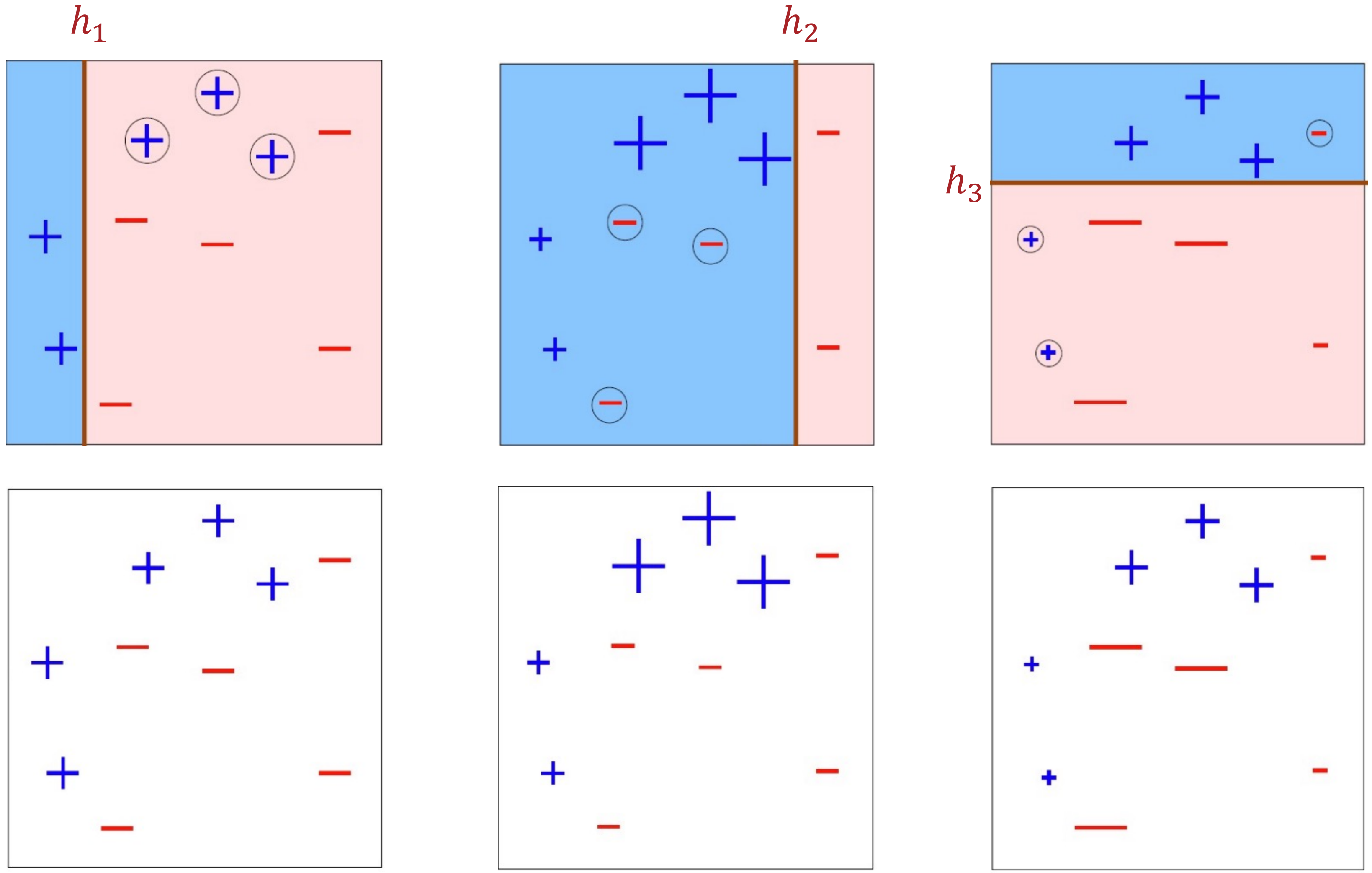
$\Rightarrow e^{-\alpha_t} < 1$ and $e^{\alpha_t} > 1$

AdaBoost: Example

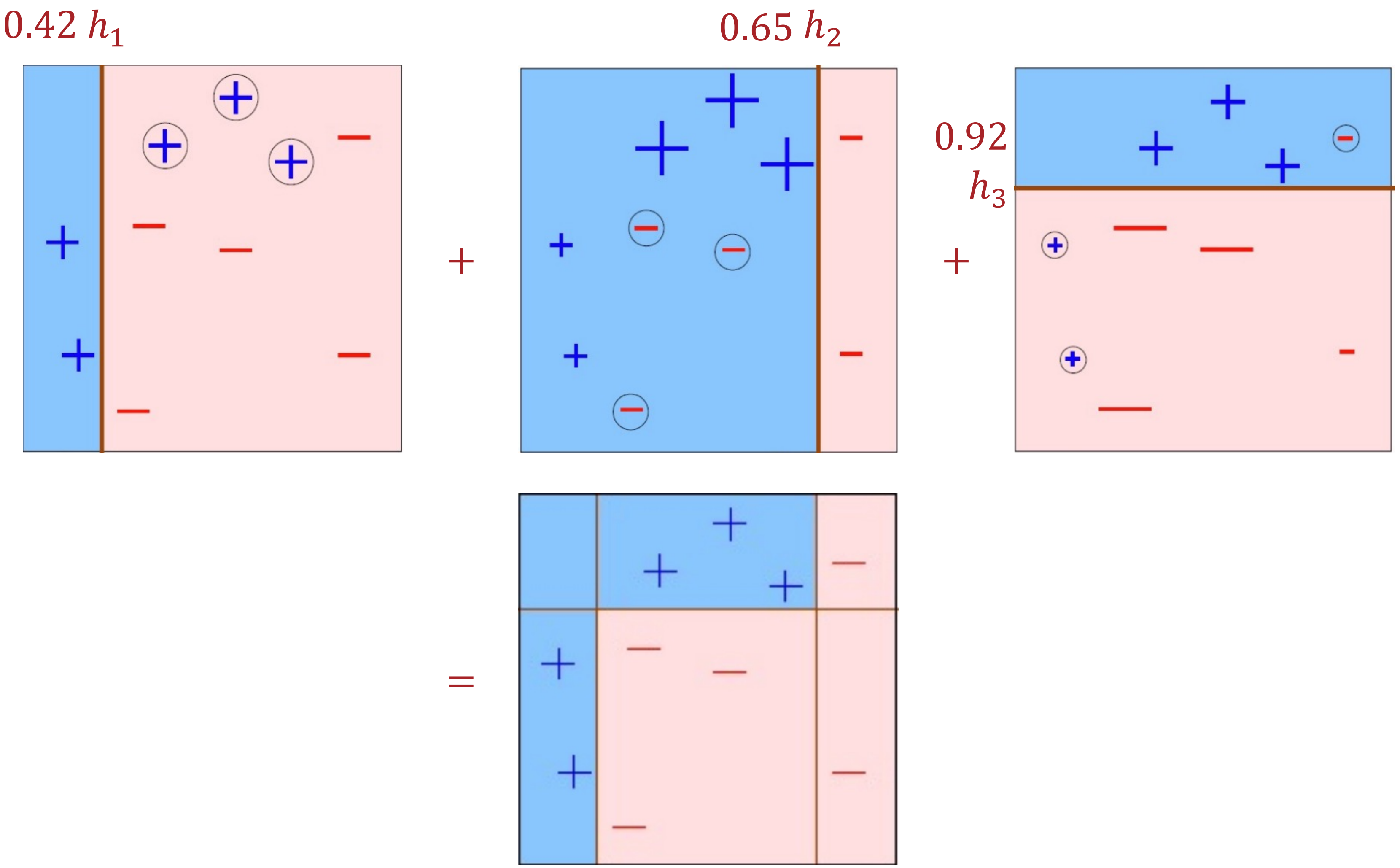
$\epsilon_1 = 0.3$
$\alpha_1 = 0.42$

$\epsilon_2 = 0.21$
$\alpha_2 = 0.65$

$\epsilon_3 = 0.14$
$\alpha_3 = 0.92$



AdaBoost: Example



Why AdaBoost?

1. If you want to use weak learners …
2. … and want your final hypothesis to be a weighted combination of weak learners, …
3. … then Adaboost greedily minimizes the exponential loss:

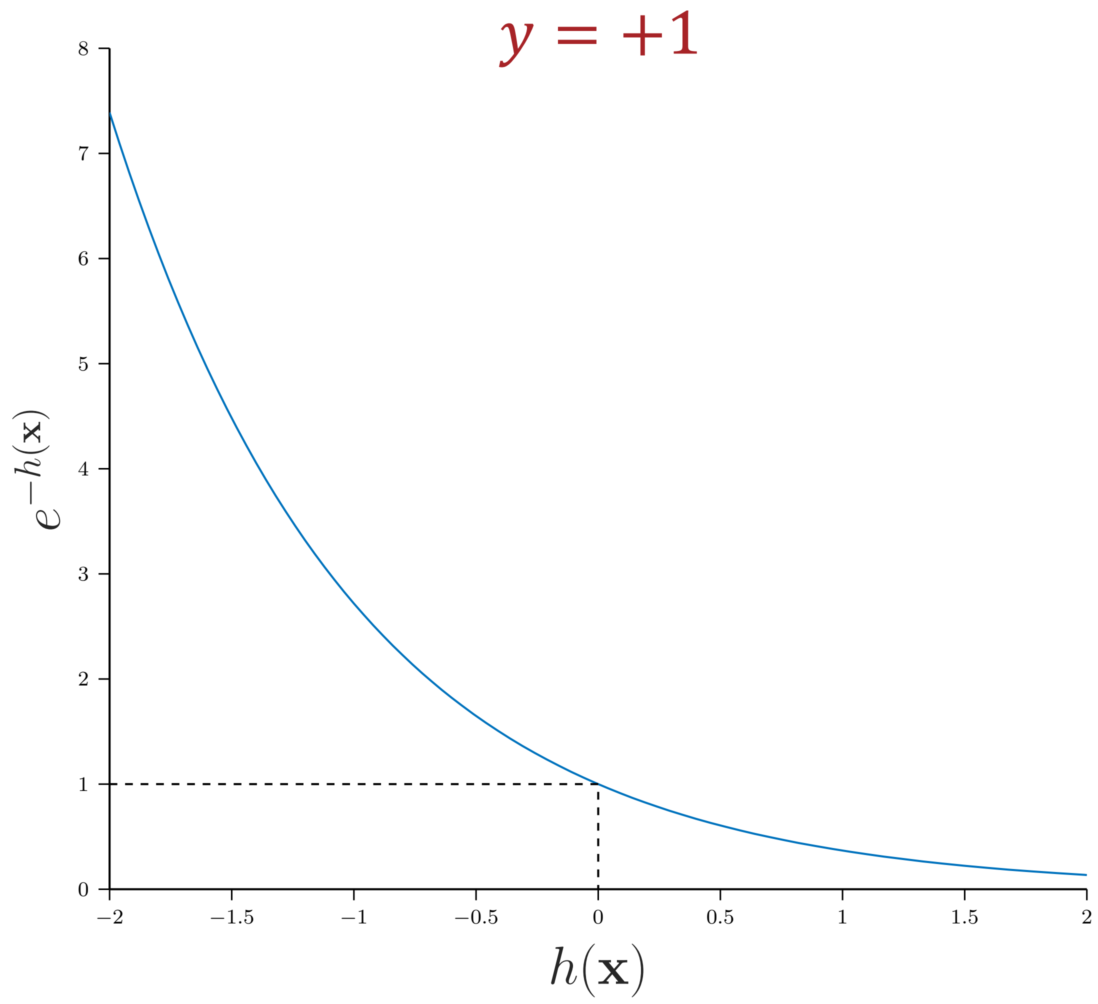
$$e(h(\boldsymbol{x}), y) = e^{(-yh(\boldsymbol{x}))}$$

1. Because they're low variance / computational constraints
2. Because weak learners are not great on their own
3. Because the exponential loss upper bounds binary error

Exponential Loss

$$e(h(\boldsymbol{x}), y) = e^{(-yh(\boldsymbol{x}))}$$

The more $h(\boldsymbol{x})$ "agrees with" y, the smaller the loss and the more $h(\boldsymbol{x})$ "disagrees with" y, the greater the loss



True Error (Freund & Schapire, 1995)

- For AdaBoost, with high probability:

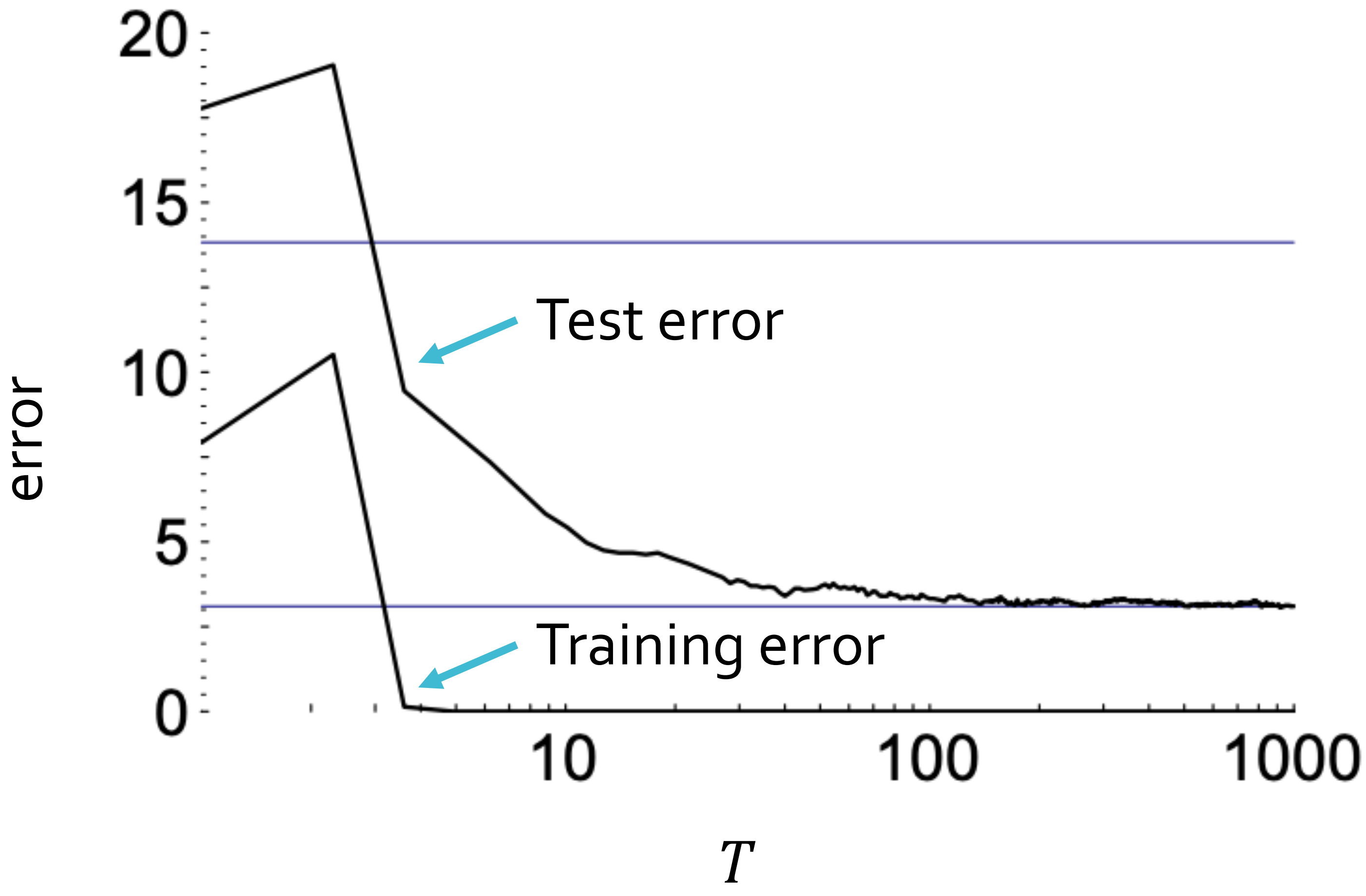
$$\text{True Error} \leq \text{Training Error} + \tilde{O}\left(\sqrt{\frac{d_{vc}(\mathcal{H})T}{N}}\right)$$

where $d_{vc}(\mathcal{H})$ is the VC-dimension of the weak learners and T is the number of weak learners.

- Empirical results indicate that increasing T does not lead to overfitting as this bound would suggest!

Source: http://rob.schapire.net/papers/FreundSc95.pdf

Test Error (Schapire, 1989)



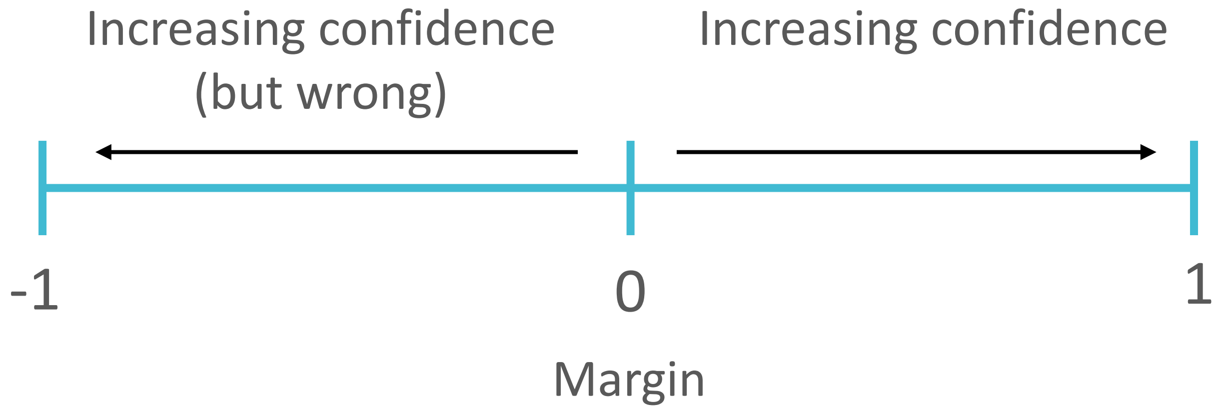
Source: http://rob.schapire.net/papers/msri.pdf

Margins

- The *margin* of training point $(\boldsymbol{x}^{(i)}, y^{(i)})$ is defined as:

$$m(\boldsymbol{x}^{(i)}, y^{(i)}) = \frac{y^{(i)} \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}^{(i)})}{\sum_{t=1}^{T} \alpha_t}$$

- The margin can be interpreted as how confident g_T is in its prediction: the bigger the margin, the more confident.



True Error (Schapire, Freund et al., 1998)

- For AdaBoost, with high probability:

$$\text{True Error} \leq \frac{1}{N}\sum_{i=1}^{N} [\![m(\boldsymbol{x}^{(i)}, y^{(i)}) \leq \epsilon]\!] + \tilde{O}\left(\sqrt{\frac{d_{vc}(\mathcal{H})}{N\epsilon^2}}\right)$$

where $d_{vc}(\mathcal{H})$ is the VC-dimension of the weak learners and $\epsilon > 0$ is a tolerance parameter.

- Even after AdaBoost has driven the training error to 0, it continues to target the “training margin”

Source: http://rob.schapire.net/papers/SchapireFrBaLe98.pdf

Learning Objectives: Boosting

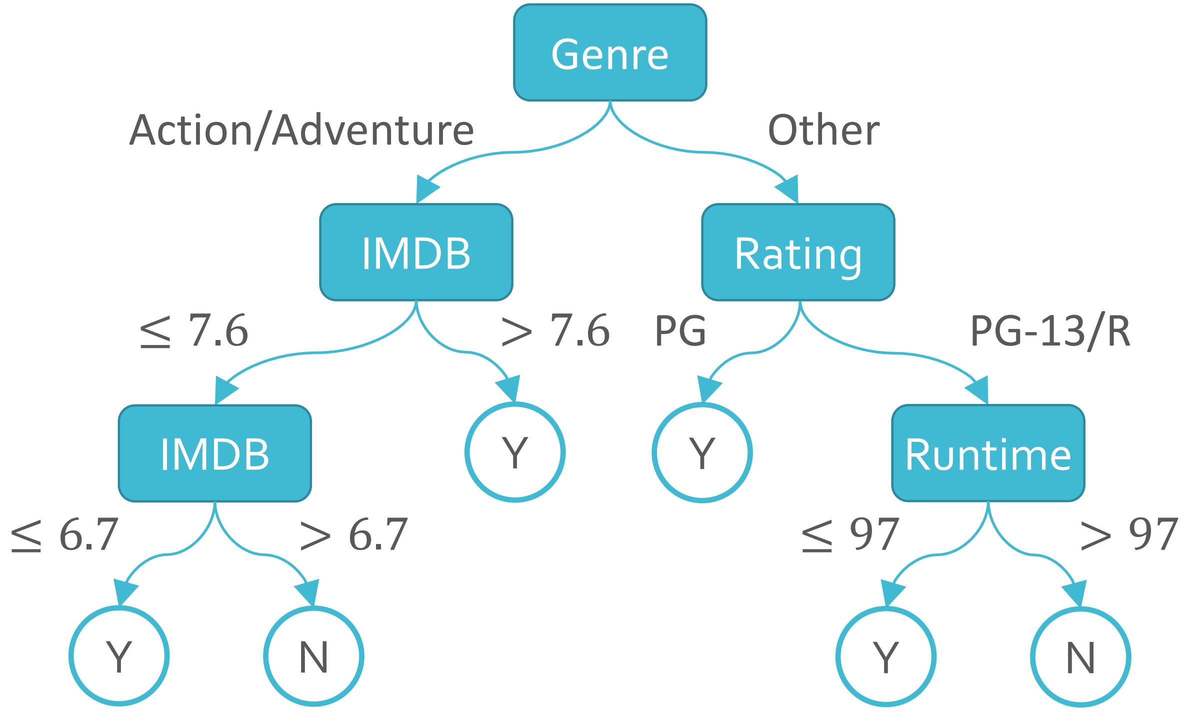
You should be able to…

1. Explain how a weighted majority vote over linear classifiers can lead to a non-linear decision boundary
2. Implement AdaBoost
3. Describe a surprisingly common empirical result regarding Adaboost train/test curves

Decision Trees: Pros & Cons

- Pros
  - Interpretable
  - Efficient (computational cost and storage)
  - Can be used for classification and regression tasks
  - Compatible with categorical and real-valued features
- Cons
  - Learned greedily: each split only considers the immediate impact on the splitting criterion
    - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
  - Prone to overfit
  - Limited expressivity (especially short trees, i.e., stumps)
    - Can be addressed via boosting
  - Highly variable
    - Can be addressed via bagging → random forests

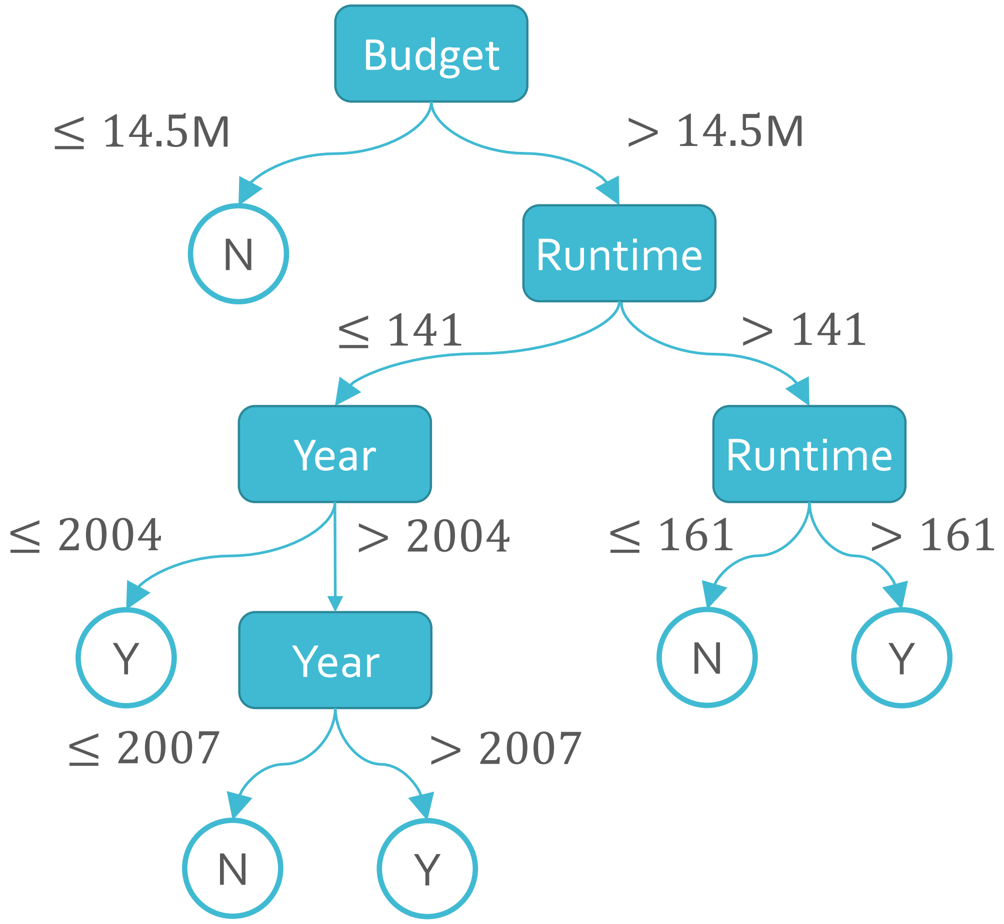
MovieID	Runtime	Genre	Budget	Year	IMDB	Rating	Liked?
1	124	Action	18M	1980	8.7	PG	Y
2	105	Action	30M	1984	7.8	PG	Y
3	103	Comedy	6M	1986	7.8	PG-13	N
4	98	Adventure	16M	1987	8.1	PG	Y
5	128	Comedy	16.4M	1989	8.1	PG	Y
6	120	Comedy	11M	1992	7.6	R	N
7	120	Drama	14.5M	1996	6.7	PG-13	N
8	136	Action	115M	1999	6.5	PG	Y
9	90	Action	90M	2001	6.6	PG-13	Y
10	161	Adventure	100M	2002	7.4	PG	N
11	201	Action	94M	2003	8.9	PG-13	Y
12	94	Comedy	26M	2004	7.2	PG-13	Y
13	157	Biography	100M	2007	7.8	R	N
14	128	Action	110M	2007	7.1	PG-13	N
15	107	Drama	39M	2009	7.1	PG-13	N
16	158	Drama	61M	2012	7.6	PG-13	N
17	169	Adventure	165M	2014	8.6	PG-13	Y
18	100	Biography	9M	2016	6.7	R	N
19	130	Action	180M	2017	7.9	PG-13	Y
20	141	Action	275M	2019	6.5	PG-13	Y



Decision Trees

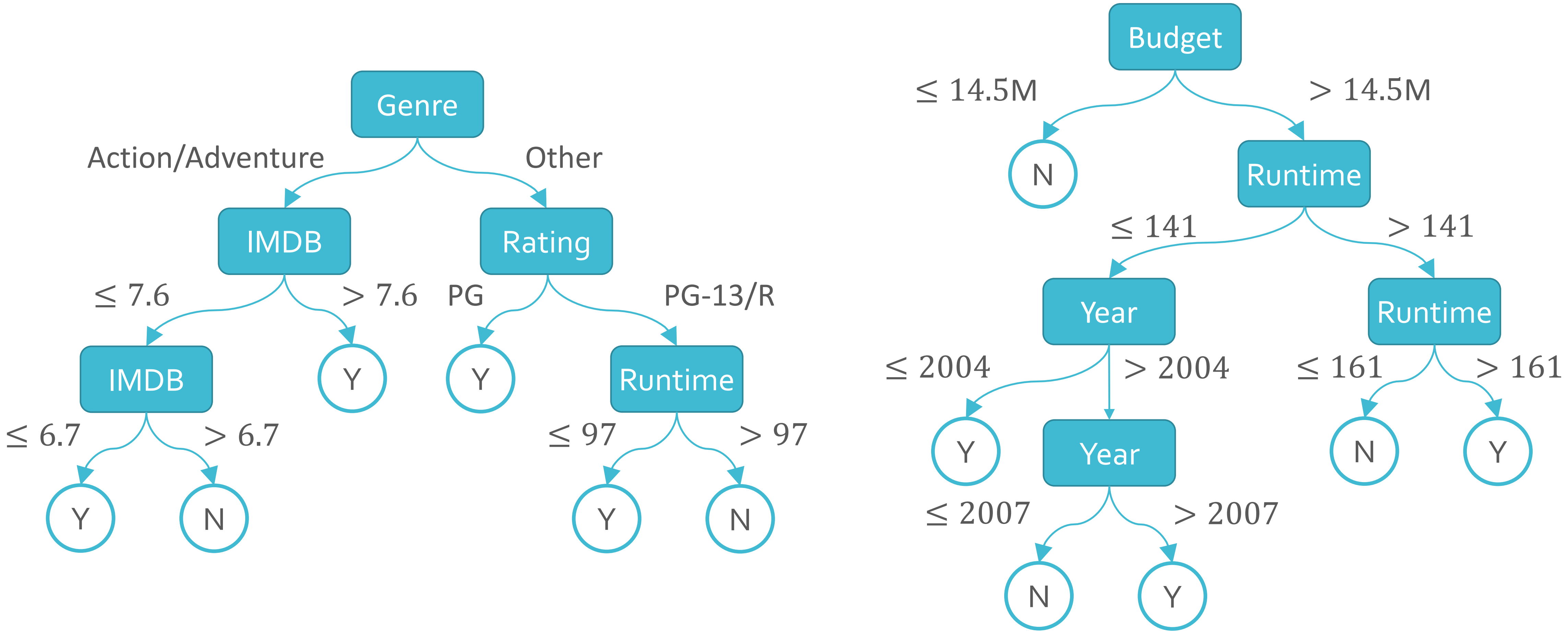
Source: https://www.kaggle.com/datasets/danielgrijalvas/movies

MovieID	Runtime	Genre	Budget	Year	IMDB	Rating	Liked?
1	124	Action	18M	1980	8.7	PG	Y
2	105	Action	30M	1984	7.8	PG	Y
3	103	Comedy	6M	1986	7.8	PG-13	N
4	98	Adventure	16M	1987	8.1	PG	Y
5	128	Comedy	16.4M	1989	8.1	PG	Y
6	120	Comedy	11M	1992	7.6	R	N
7	120	Drama	14.5M	1996	6.7	PG-13	N
8	136	Action	115M	1999	6.5	PG	Y
9	90	Action	90M	2001	6.6	PG-13	Y
10	161	Adventure	100M	2002	7.4	PG	N
11	201	Action	94M	2003	8.9	PG-13	Y
12	94	Comedy	26M	2004	7.2	PG-13	Y
13	157	Biography	100M	2007	7.8	R	N
14	128	Action	110M	2007	7.1	PG-13	N
15	107	Drama	39M	2009	7.1	PG-13	N
16	158	Drama	61M	2012	7.6	PG-13	Y
17	169	Adventure	165M	2014	8.6	PG-13	Y
18	100	Biography	9M	2016	6.7	R	N
19	130	Action	180M	2017	7.9	PG-13	Y
20	141	Action	275M	2019	6.5	PG-13	Y



Decision Trees

Source: https://www.kaggle.com/datasets/danielgrijalvas/movies



Decision Trees

Random Forests

- Combines the prediction of many diverse decision trees to reduce their variability
- If B independent random variables $x^{(1)}, x^{(2)}, \dots, x^{(B)}$ all have variance σ^2, then the variance of $\frac{1}{B}\sum_{b=1}^{B} x^{(b)}$ is $\frac{\sigma^2}{B}$
- Random forests = bagging + split-feature randomization

  = <u>b</u>ootstrap <u>agg</u>regat<u>ing</u> + split-feature randomization

Random Forests

- Combines the prediction of many diverse decision trees to reduce their variability
- If B independent random variables $x^{(1)}, x^{(2)}, \dots, x^{(B)}$ all have variance σ^2, then the variance of $\frac{1}{B}\sum_{b=1}^{B} x^{(b)}$ is $\frac{\sigma^2}{B}$
- Random forests = bagging + split-feature randomization

  = bootstrap aggregating + split-feature randomization

Aggregating

- How can we combine multiple decision trees, $\boxed{\{t_1, t_2, \dots, t_B\}}$, to arrive at a single prediction?
- Regression - average the predictions:

$$\bar{t}(\boldsymbol{x}) = \frac{1}{B}\sum_{b=1}^{B} t_b(\boldsymbol{x})$$

- Classification - plurality (or majority) vote; for binary labels encoded as $\{-1, +1\}$:

$$\bar{t}(\boldsymbol{x}) = \operatorname{sign}\left(\frac{1}{B}\sum_{b=1}^{B} t_b(\boldsymbol{x})\right)$$

Random Forests

- Combines the prediction of many diverse decision trees to reduce their variability
- If B independent random variables $x^{(1)}, x^{(2)}, \ldots, x^{(B)}$ all have variance σ^2, then the variance of $\frac{1}{B}\sum_{b=1}^{B} x^{(b)}$ is $\frac{\sigma^2}{B}$
- Random forests = bagging + split-feature randomization

  = bootstrap aggregating + split-feature randomization

Bootstrapping

- Insight: one way of generating different decision trees is by changing the training data set
- Issue: often, we only have one fixed set of training data
- Idea: resample the data multiple times ***with replacement***

MovieID	…
1	…
2	…
3	…
⋮	⋮
19	…
20	…

Training data

MovieID	…
1	…
1	…
1	…
⋮	⋮
14	…
19	…

Bootstrapped Sample 1

MovieID	…
4	…
4	…
5	…
⋮	⋮
16	…
16	…

Bootstrapped Sample 2

…

…

Bootstrapping

- Idea: resample the data multiple times ***with replacement***
  - Each bootstrapped sample has the same number of data points as the original data set
  - Duplicated points cause different decision trees to focus on different parts of the input space

MovieID	…
1	…
2	…
3	…
⋮	⋮
19	…
20	…

Training data

MovieID	…
1	…
1	…
1	…
⋮	⋮
14	…
19	…

Bootstrapped Sample 1

MovieID	…
4	…
4	…
5	…
⋮	⋮
16	…
16	…

Bootstrapped Sample 2

…

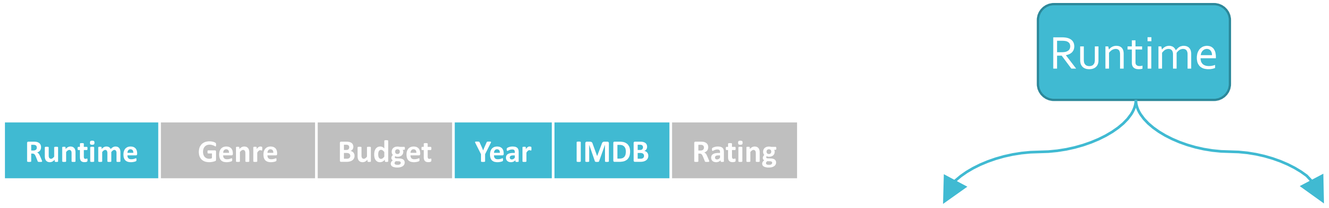
Split-feature Randomization (a.k.a. Feature Bagging or Random Subspace Methods)

- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset

Runtime	Genre	Budget	Year	IMDB	Rating

Split-feature Randomization (a.k.a. Feature Bagging or Random Subspace Methods)

- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset

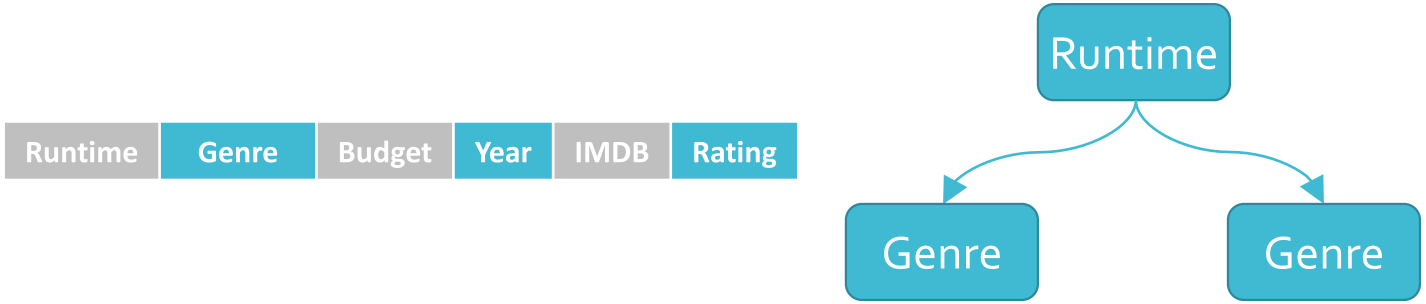


Split-feature Randomization (a.k.a. Feature Bagging or Random Subspace Methods)

- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset

Split-feature Randomization (a.k.a. Feature Bagging or Random Subspace Methods)

- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset



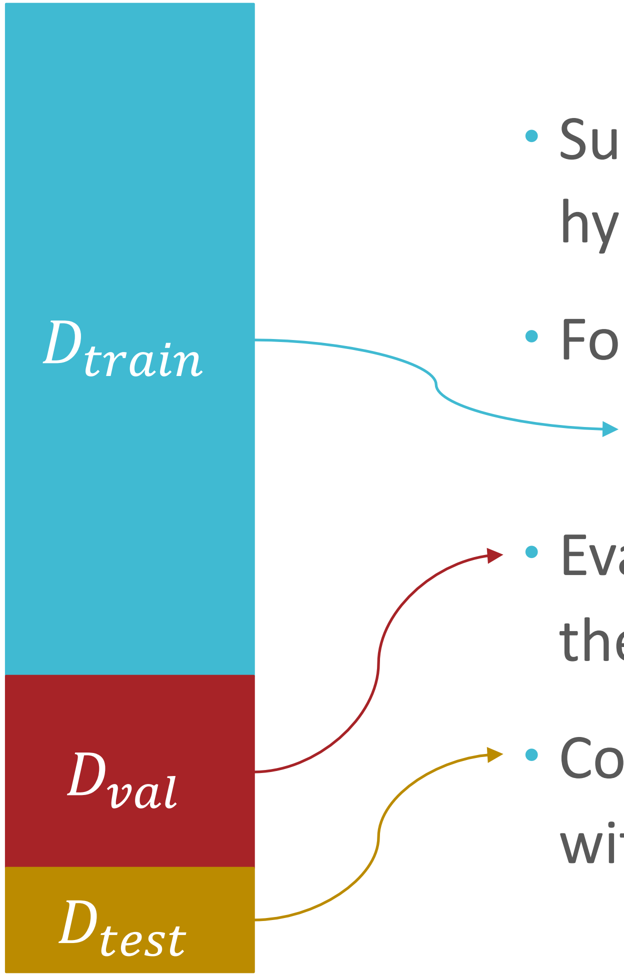
Random Forests

- Input: $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}, B, \rho$
- For $b = 1, 2, \dots, B$
  - Create a dataset, $\mathcal{D}_b$, by sampling N points from the original training data $\mathcal{D}$ ***with replacement***
  - Learn a decision tree, t_b, using $\mathcal{D}_b$ and the ID3 algorithm ***with split-feature randomization***, sampling ρ features for each split
- Output: $\bar{t} = f(t_1, \dots, t_B)$, the aggregated hypothesis

How can we set B and ρ?

- Input: $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$, B, ρ
- For $b = 1, 2, \ldots, B$
  - Create a dataset, $\mathcal{D}_b$, by sampling N points from the original training data $\mathcal{D}$ ***with replacement***
  - Learn a decision tree, t_b, using $\mathcal{D}_b$ and the ID3 algorithm ***with split-feature randomization***, sampling ρ features for each split
- Output: $\bar{t} = f(t_1, \ldots, t_B)$, the aggregated hypothesis

Recall: Validation Sets



- Suppose we want to compare multiple hyperparameter settings $\theta_1, \dots, \theta_K$
- For $k = 1, 2, \dots, K$
  - Train a model on D_{train} using θ_k
- Evaluate each model on D_{val} and find the best hyperparameter setting, θ_{k^*}
- Compute the error of a model trained with θ_{k^*} on D_{test}

Out-of-bag Error

- For each training point, $\boldsymbol{x}^{(n)}$, there are some decision trees which $\boldsymbol{x}^{(n)}$ was not used to train (roughly B/e trees or 37%)
  - Let these be $t^{(-n)} = \left\{t_1^{(-n)}, t_2^{(-n)}, \dots, t_{N_{-n}}^{(-n)}\right\}$
- Compute an aggregated prediction for each $\boldsymbol{x}^{(n)}$ using the trees in $t^{(-n)}$, $\bar{t}^{(-n)}\left(\boldsymbol{x}^{(n)}\right)$
- Compute the out-of-bag (OOB) error, e.g., for regression

$$E_{OOB} = \frac{1}{N}\sum_{n=1}^{N}\left(\bar{t}^{(-n)}\left(\boldsymbol{x}^{(n)}\right) - y^{(n)}\right)^2$$

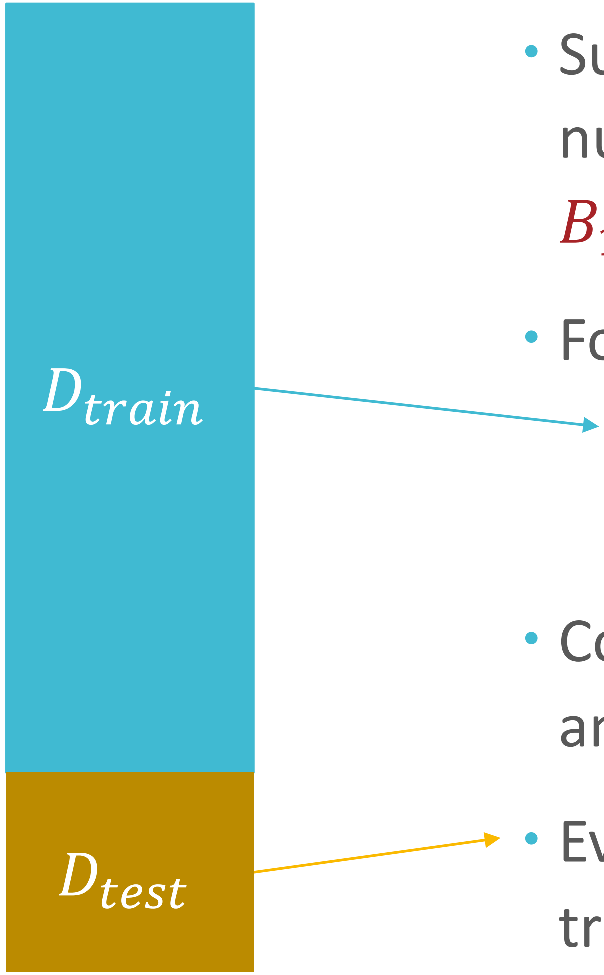
Out-of-bag Error

- For each training point, $\boldsymbol{x}^{(n)}$, there are some decision trees which $\boldsymbol{x}^{(n)}$ was not used to train (roughly B/e trees or 37%)
  - Let these be $t^{(-n)} = \left\{t_1^{(-n)}, t_2^{(-n)}, \dots, t_{N_{-n}}^{(-n)}\right\}$
- Compute an aggregated prediction for each $\boldsymbol{x}^{(n)}$ using the trees in $t^{(-n)}$, $\bar{t}^{(-n)}\left(\boldsymbol{x}^{(n)}\right)$
- Compute the out-of-bag (OOB) error, e.g., for classification

$$E_{OOB} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\left(\bar{t}^{(-n)}\left(\boldsymbol{x}^{(n)}\right) \neq y^{(n)}\right)$$

- E_{OOB} can be used for hyperparameter optimization!

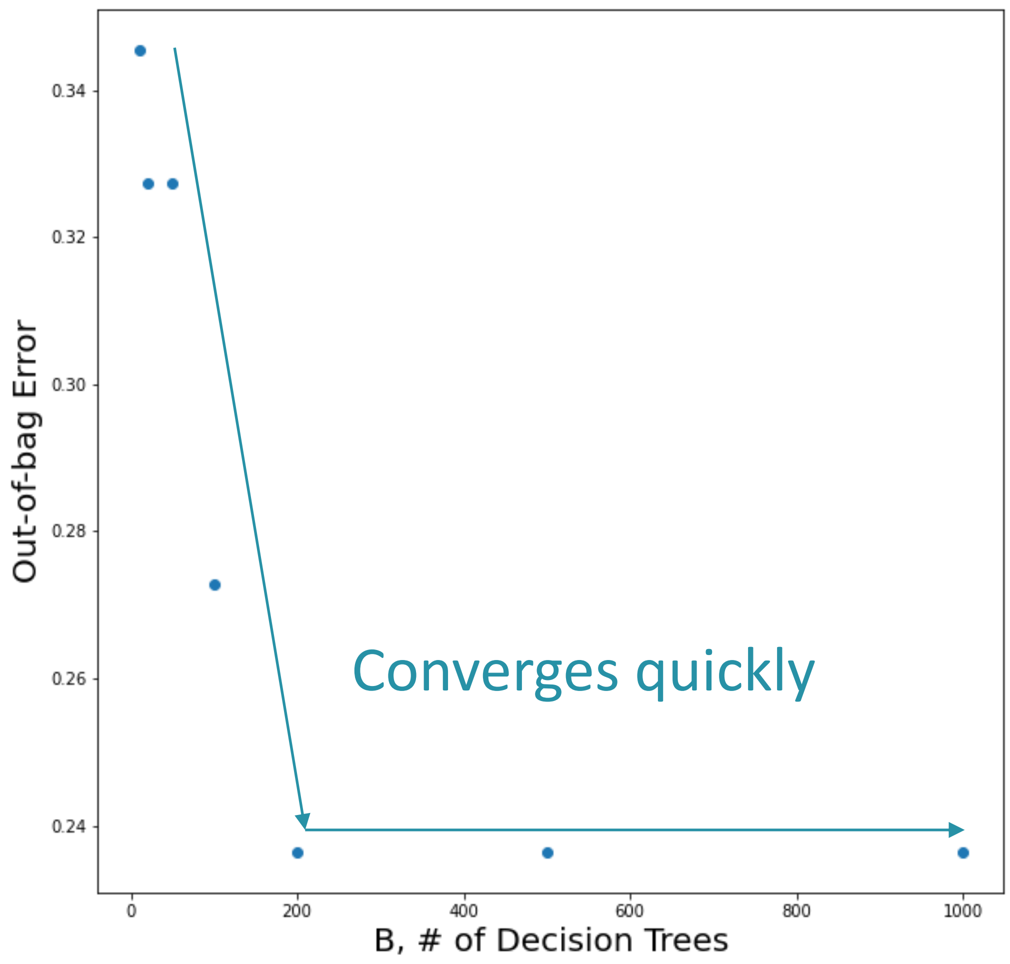
Out-of-bag Error

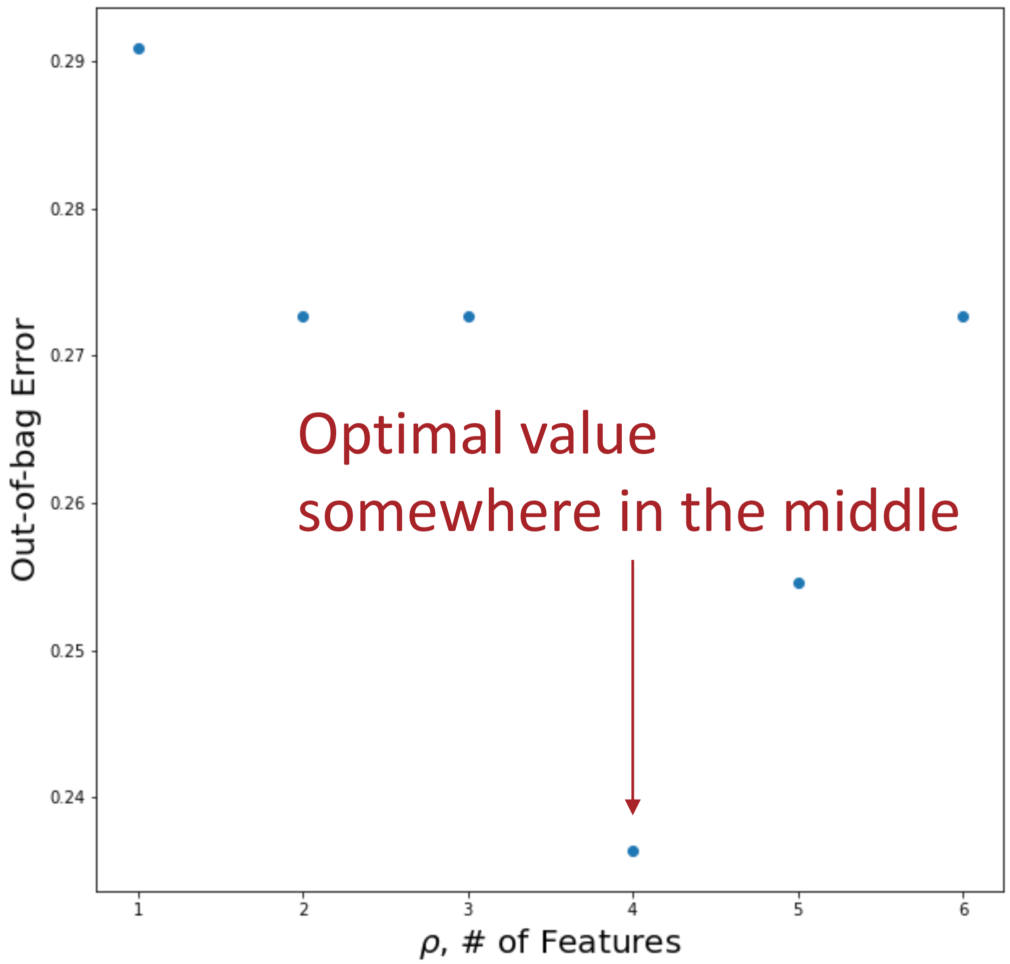


- Suppose we want to compare different numbers of trees in our random forest $B_1, \dots, B_K$
- For $k = 1, 2, \dots, K$
  - Train a random forest on D_{train} with B_k trees
- Compute E_{OOB} for each random forest and find the best number of trees, B_{k^*}
- Evaluate the random forest with B_{k^*} trees on D_{test}

Out-of-bag Error

- Suppose we want to compare different numbers of trees in our random forest $B_1, \ldots, B_K$
- For $k = 1, 2, \ldots, K$
  - Train a random forest on D_{train} with B_k trees
- Compute E_{OOB} for each random forest and find the best number of trees, B_{k^*}
- Evaluate the random forest with B_{k^*} trees on D_{test}





Setting Hyperparameters

Learning Objectives: Bagging

You should be able to…

1. Distinguish between (sample) bagging, split-feature randomization, and random forests.
2. Implement (sample) bagging for an arbitrary base classifier/regressor.
3. Implement the split-feature randomization for an arbitrary base classifier/ regressor.
4. Implement random forests.
5. Contrast out-of-bag error with cross-validation error.
6. Differentiate boosting from bagging.
7. Compare and contrast weighted and unweighted majority votes of a collection of classifiers.
8. Discuss the relationship between sample size and variance of the base classifier/regressor in the context of bagging