

10-701: Introduction to Machine Learning

Lecture 21 – Bagging & Boosting

Hoda Heidari

* Slides adopted from F24 offering of 10701 by Henry Chai.

Bias-Variance Decomposition

- $(x, y) \sim P \rightarrow c^*$ independent
- Suppose $y = f(x) + \varepsilon$, where ε is noise with $E[\varepsilon] = 0, \text{Var}(\varepsilon) = \sigma^2$.
- Learning algorithm takes D , gives us \hat{f}_D .
- The expected squared prediction error at a point x :
$$\begin{aligned} E_{D,\varepsilon}[(y - \hat{f}(x))^2] &= E_{D,\varepsilon} \left[(\underbrace{f(x) + \varepsilon - \hat{f}(x)}_{})^2 \right] \\ &= E[(f(x) - \hat{f}(x))^2] + E[\varepsilon^2] + 2 \underbrace{E[(f(x) - \hat{f}(x))\varepsilon]}_{E(f(x) - \hat{f}(x)) E[\varepsilon]} \\ &= \underbrace{E[(f(x) - \hat{f}(x))^2]}_{\text{Bias}^2} + \sigma^2 \end{aligned}$$

Bias-Variance Decomposition

- $$\begin{aligned} \mathbb{E}_{\mathcal{D}}[(f(x) - \hat{f}(x))^2] &= \mathbb{E}_{\mathcal{D}}\left[\underbrace{(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_D(x)])}_{\text{Bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}[\hat{f}_D(x)] - \hat{f}(x)}_{\text{Variance}}\right]^2 \\ &= \mathbb{E}_{\mathcal{D}}[(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_D(x)])^2] + \mathbb{E}_{\mathcal{D}}[(\hat{f}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_D(x)])^2] \\ &\quad + 2 \mathbb{E}_{\mathcal{D}}\left[\underbrace{(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_D(x)])}_{\text{Bias}} (\hat{f}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_D(x)])\right] \leftarrow 0 \\ &\quad \hookrightarrow 2 \mathbb{E}\left[(f(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_D(x)])\right] \mathbb{E}\left(\hat{f}(x) - \mathbb{E}_{\mathcal{D}}[\hat{f}_D(x)]\right) \\ &= \left(\mathbb{E}_{\mathcal{D}}[\hat{f}(x)] - \mathbb{E}_{\mathcal{D}}[\hat{f}_D(x)]\right)^2 \end{aligned}$$
- $$\mathbb{E}_{\mathcal{D}}[(f(x) - \hat{f}(x))^2] = \underbrace{(f(x) - \mathbb{E}[\hat{f}(x)])^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]}_{\text{Variance}} \text{ or}$$

$$\mathbb{E}[(y - \hat{f}(x))^2] = \text{Bias}^2[\hat{f}(x)] + \text{Var}[\hat{f}(x)] + \sigma^2$$

The Wisdom of Crowds

- In 1906, Francis Galton asked ~800 people at a farmer's fair to guess the weight of a cow, including "experts"
 - Actual weight: 1198 lbs
 - Mean guess: 1197 lbs
 - Mean guess was more accurate than any single guess, even the experts

Ensemble Learning

- **Key idea:** Different models make different errors. By aggregating their predictions, ensembles can reduce variance, bias, or both—leading to better accuracy and robustness than any single model alone.
- **Common ensemble methods**
 - **Bagging (Bootstrap Aggregating):** *Random Forests*
Trains many models independently on different random subsets of the data to reduce variance and overfitting.
 - **Boosting:** *Adaboost*
Trains models sequentially, with each new model focusing on the errors of the previous to reduce bias.
 - **Stacking:** *X*
Combines predictions from multiple models using another model (*a meta-learner*) that learns how best to blend them.

The Netflix Prize

The screenshot shows the official Netflix Prize website. At the top, a yellow banner features the text "Netflix Prize" and a large red "COMPLETED" stamp. Below the banner is a navigation menu with links for "Home", "Rules", "Leaderboard", and "Update". A sidebar on the left lists statistics: "500,000 users", "20,000 movies", and "100 million ratings". The main content area contains a large list bullet-pointing the goal: "Goal: To obtain lower error than Netflix's existing system on 3 million held out ratings". To the right, a white box with a red border displays a "Congratulations!" message. This message explains the prize's purpose ("improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences"), details the winning team ("BellKor's Pragmatic Chaos"), and encourages user interaction ("Leaderboard", "Forum"). It concludes by praising the contributors for improving movie recommendations.

- 500,000 users
- 20,000 movies
- 100 million ratings
- Goal: To obtain lower error than Netflix's existing system on 3 million held out ratings

Congratulations!

The Netflix Prize sought to substantially improve the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences.

On September 21, 2009 we awarded the \$1M Grand Prize to team "BellKor's Pragmatic Chaos". Read about [their algorithm](#), checkout team scores on the [Leaderboard](#), and join the discussions on the [Forum](#).

We applaud all the contributors to this quest, which improves our ability to connect people to the movies they love.

The Netflix Prize

Netflix Prize COMPLETED

Home | Rules | Leaderboard | Update | Download

Leaderboard

Showing Test Score. [Click here to show quiz score](#)

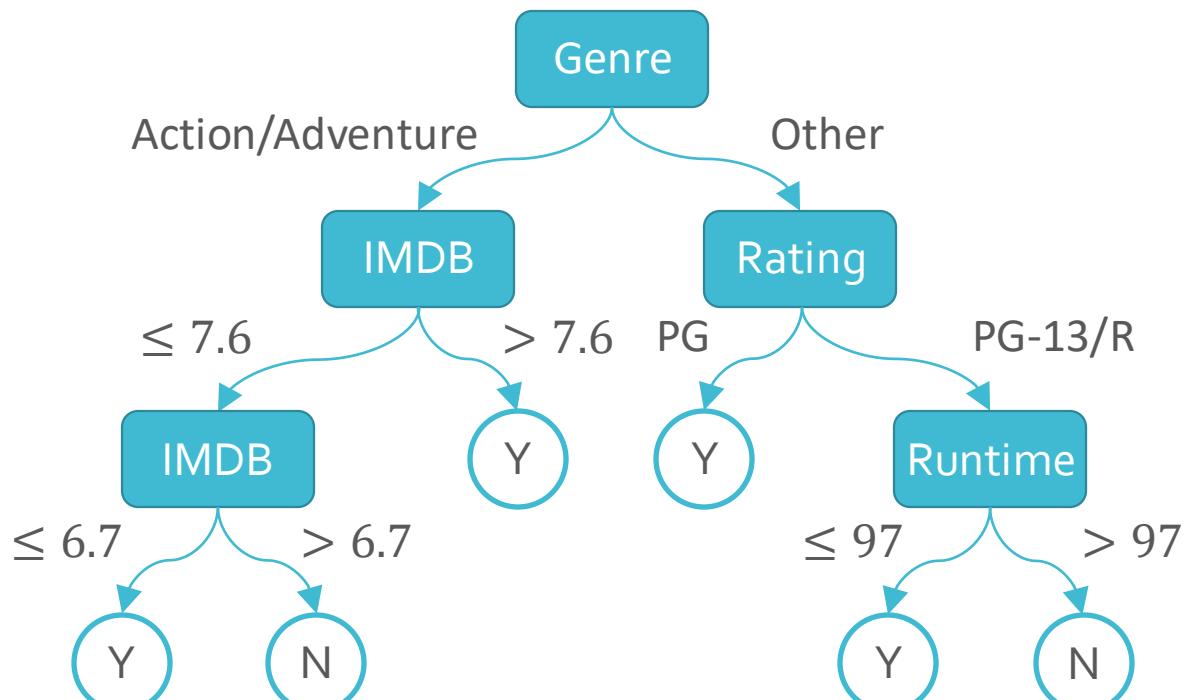
Display top leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	 BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	 PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	 BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	 BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	 BellKor	0.8624	9.46	2009-07-26 17:19:11

MovielID	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

Movie Recommendations

MovielID	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

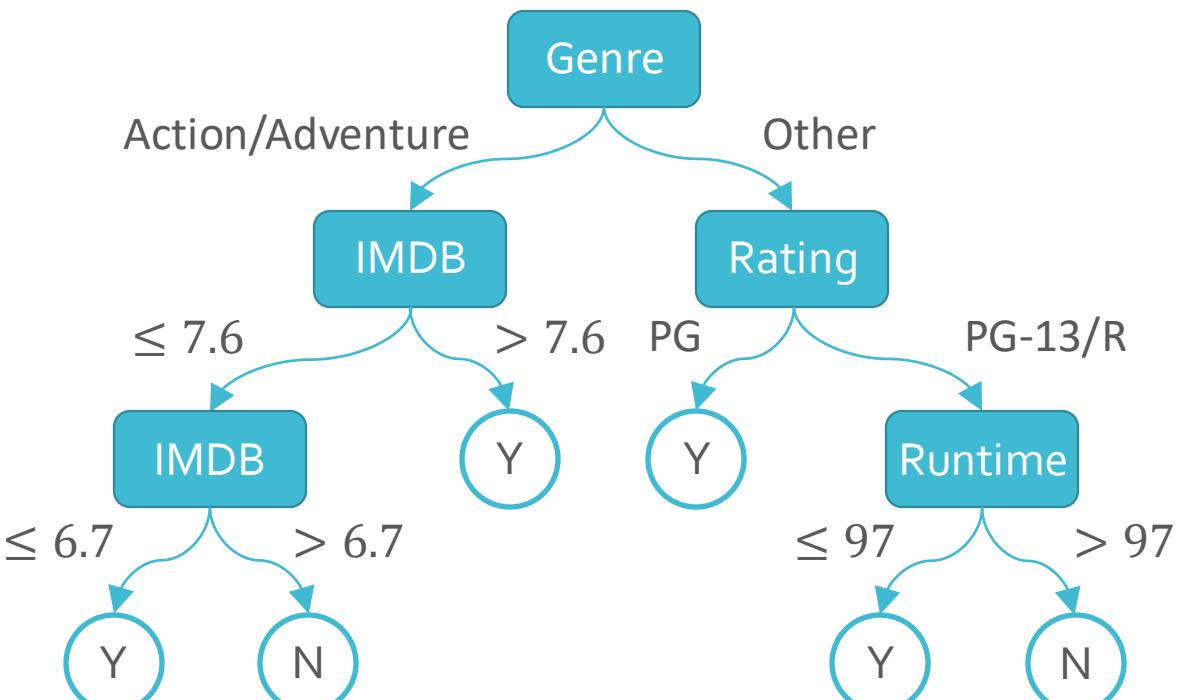


Decision Trees

Recall: Decision Tree Pros & Cons

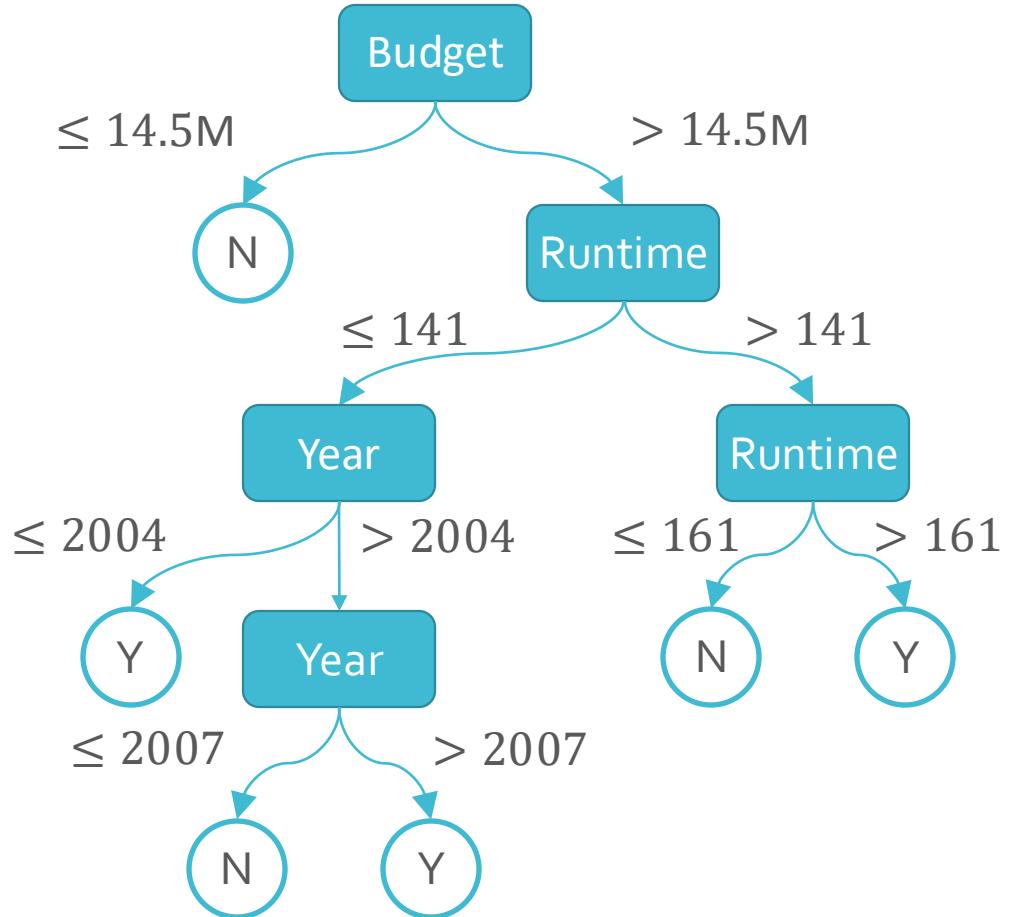
- Pros
 - Interpretable
 - Efficient (computational cost and storage)
 - Can be used for classification and regression tasks
 - Compatible with categorical and real-valued features
- Cons
 - Learned greedily: each split only considers the immediate impact on the splitting criterion
 - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
 - Prone to overfit
 - High variance

MovielID	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

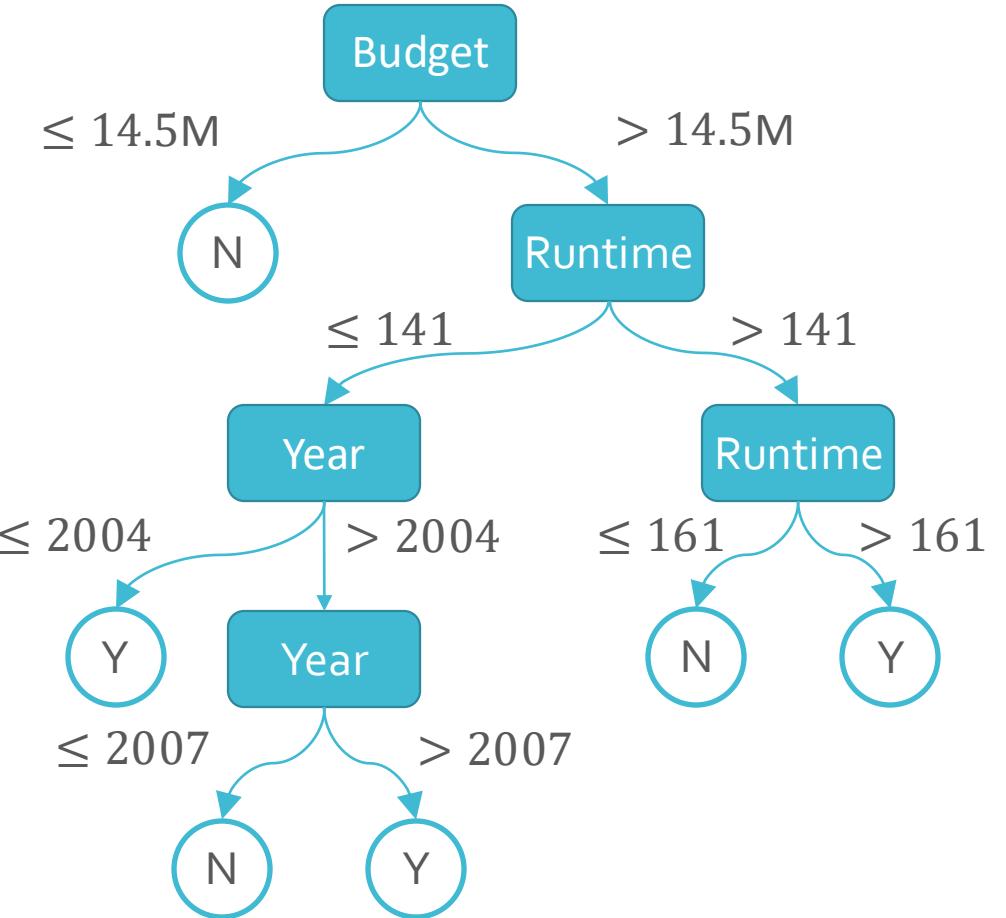
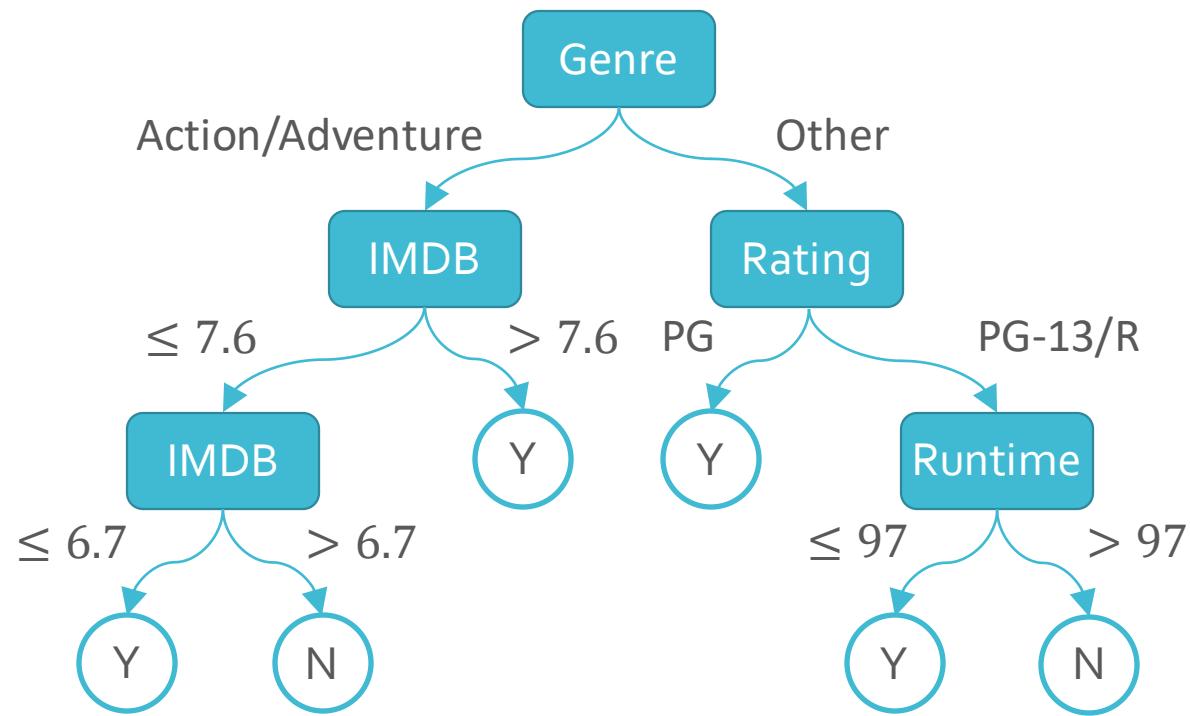


Decision Trees

MovielID	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



Decision Trees

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
 - • High variance
 - **Can be addressed via ensembles → random forests**

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

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, $\{t_1, t_2, \dots, t_B\}$, to arrive at a single prediction?

- Regression - average the predictions:

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

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

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

Random Forests

- Combines the prediction of many diverse decision trees to reduce their variability
- If B independent random variables $x^{(1)}, x^{(2)}, \dots, x^{(B)}$ 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

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

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***

MovielID	...
1	...
2	...
3	...
:	:
19	...
20	...

Training data

MovielID	...
1	...
1	...
1	...
:	:
14	...
19	...

Bootstrapped
Sample 1

MovielID	...
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

MovielID	...
1	...
2	...
3	...
:	:
19	...
20	...

Training data

MovielID	...
1	...
1	...
1	...
:	:
14	...
19	...

Bootstrapped
Sample 1

MovielID	...
4	...
4	...
5	...
:	:
16	...
16	...

Bootstrapped
Sample 2

Split-feature Randomization

- Issue: decision trees trained on bootstrapped samples still tend to 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

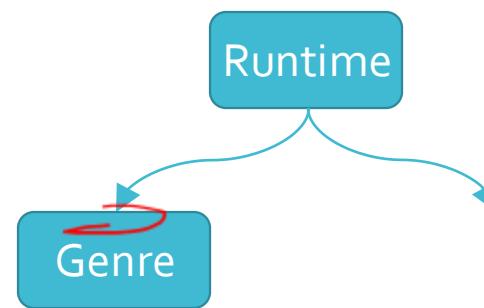
- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset



Split-feature Randomization

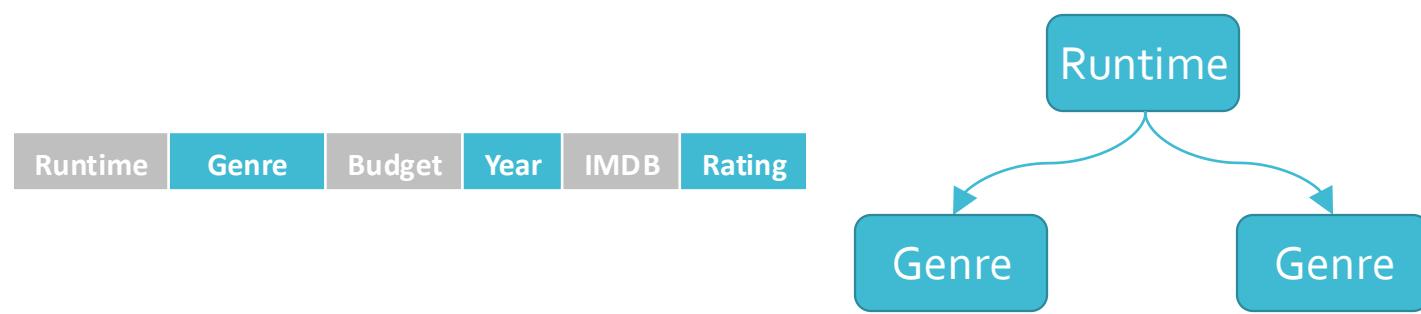
- Issue: decision trees trained on bootstrapped samples still behave similarly
- Idea: in addition to sampling the data points (i.e., the rows), also sample the features (i.e., the columns)
- Each time a split is being considered, limit the possible features to a randomly sampled subset

Runtime	Genre	Budget	Year	IMDB	Rating
---------	-------	--------	------	------	--------



Split-feature Randomization

- 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



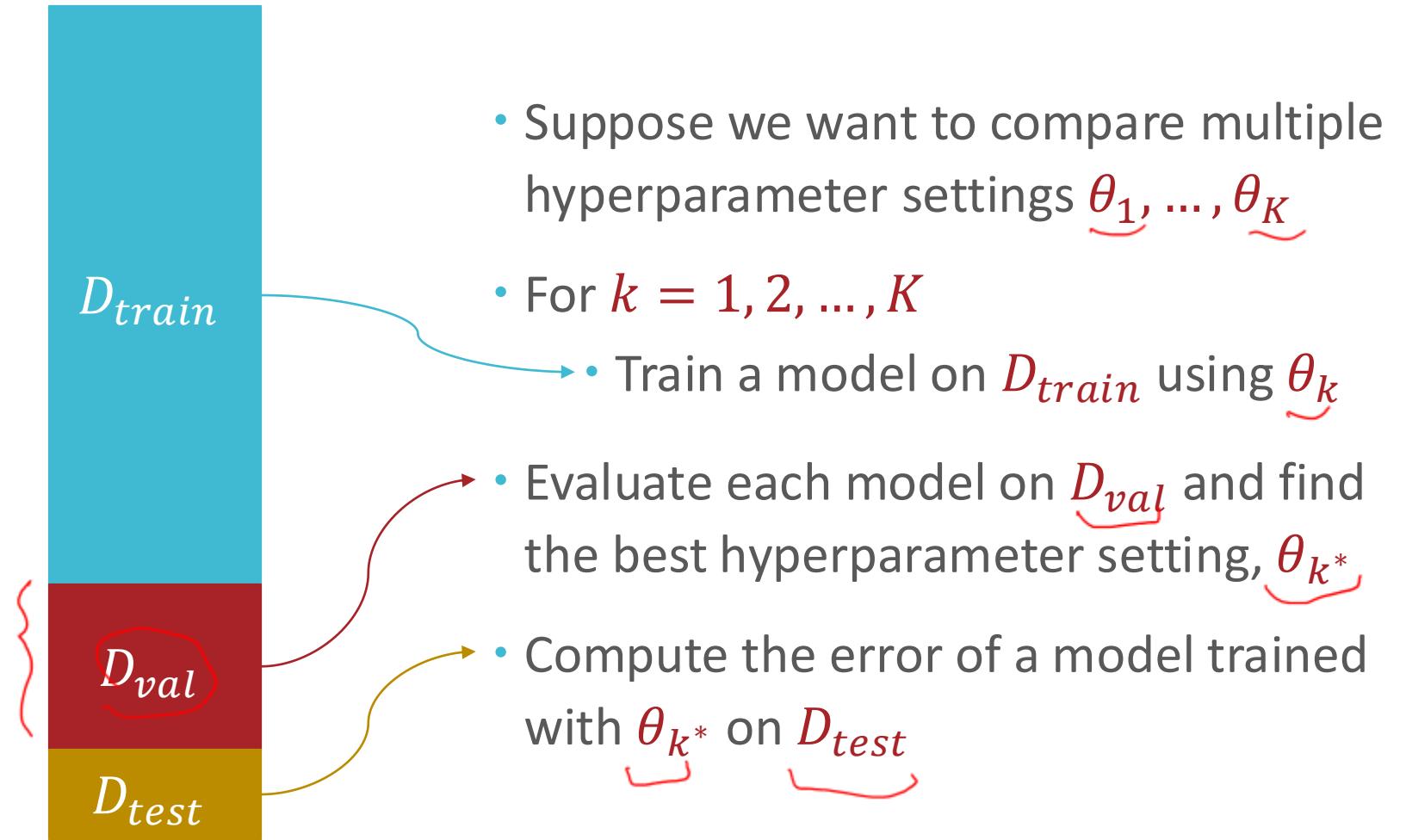
Random Forests

- Input: $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N, B, \rho$
- 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: $\mathcal{D} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^N, B, \rho$
- 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

Recall: Validation Sets



Out-of-bag Error

$$B \left(1 - \frac{1}{e}\right)$$

- For each training point, $\underline{x}^{(n)}$, there are some decision trees which $x^{(n)}$ was not used to train (roughly B/e trees or 37%)
 - Let these be $\underline{t}^{(-n)} = \{\underline{t}_1^{(-n)}, \underline{t}_2^{(-n)}, \dots, \underline{t}_{N-n}^{(-n)}\}$
 - Compute an aggregated prediction for each $\underline{x}^{(n)}$ using the trees in $t^{(-n)}, \bar{t}^{(-n)}(\underline{x}^{(n)})$
 - Compute the out-of-bag (OOB) error, e.g., for regression

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

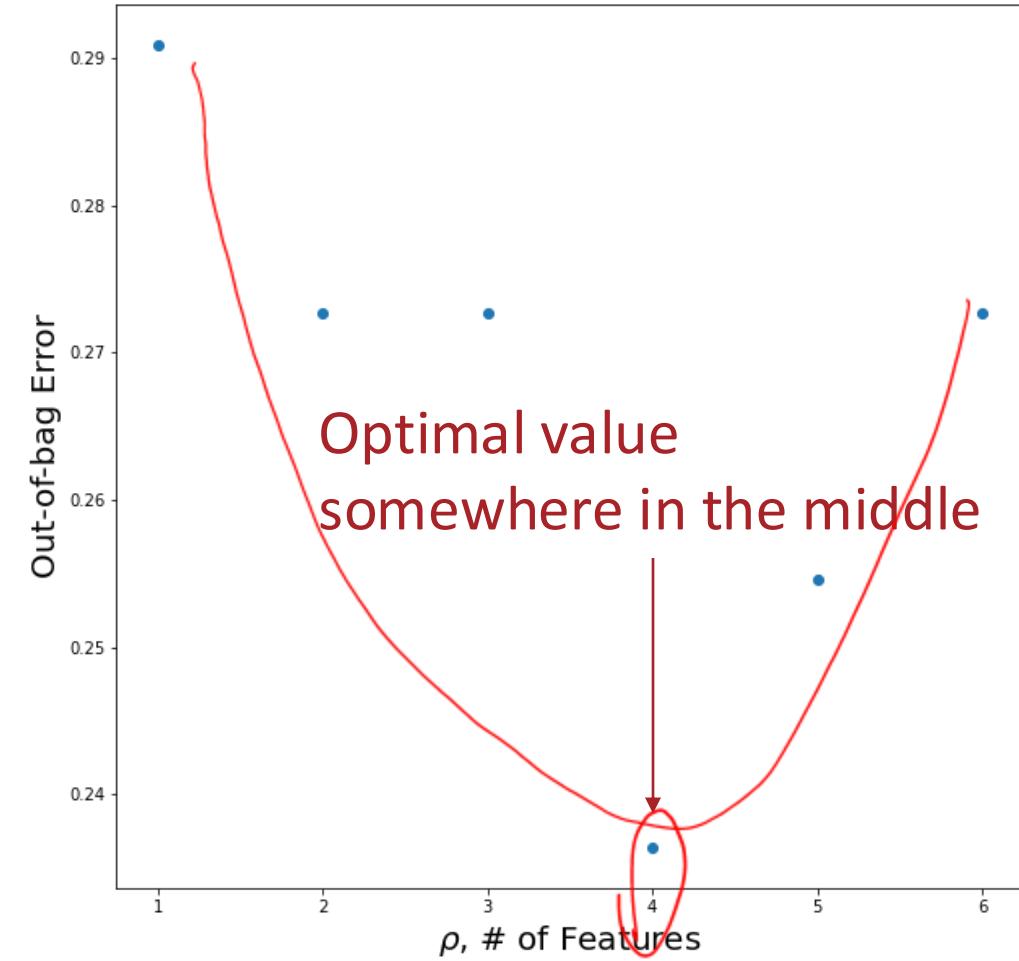
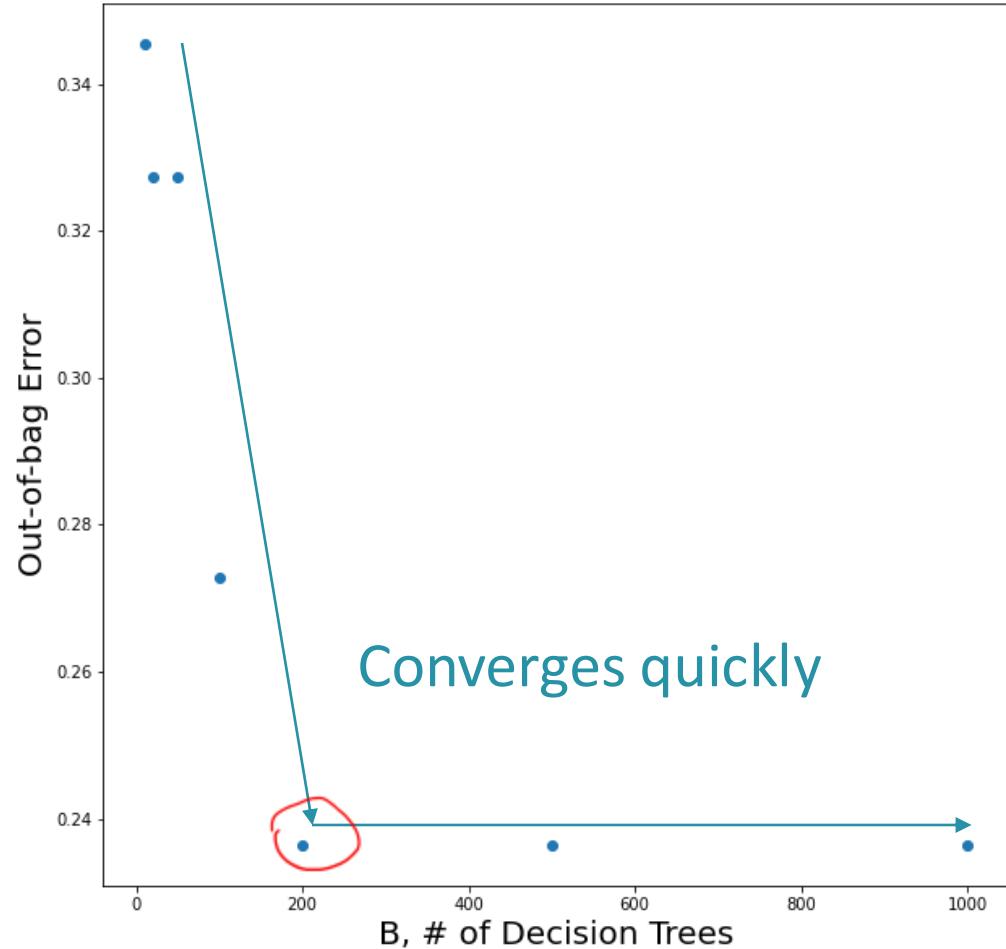
Out-of-bag Error

- For each training point, $\mathbf{x}^{(n)}$, there are some decision trees which $\mathbf{x}^{(n)}$ was not used to train (roughly B/e trees or 37%)
 - Let these be $\mathbf{t}^{(-n)} = \{t_1^{(-n)}, t_2^{(-n)}, \dots, t_{N-n}^{(-n)}\}$
 - Compute an aggregated prediction for each $\mathbf{x}^{(n)}$ using the trees in $\mathbf{t}^{(-n)}, \bar{\mathbf{t}}^{(-n)}(\mathbf{x}^{(n)})$
 - Compute the out-of-bag (OOB) error, e.g., for classification
$$E_{OOB} = \frac{1}{N} \sum_{n=1}^N \mathbb{1}(\bar{\mathbf{t}}^{(-n)}(\mathbf{x}^{(n)}) \neq y^{(n)})$$
 - 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}

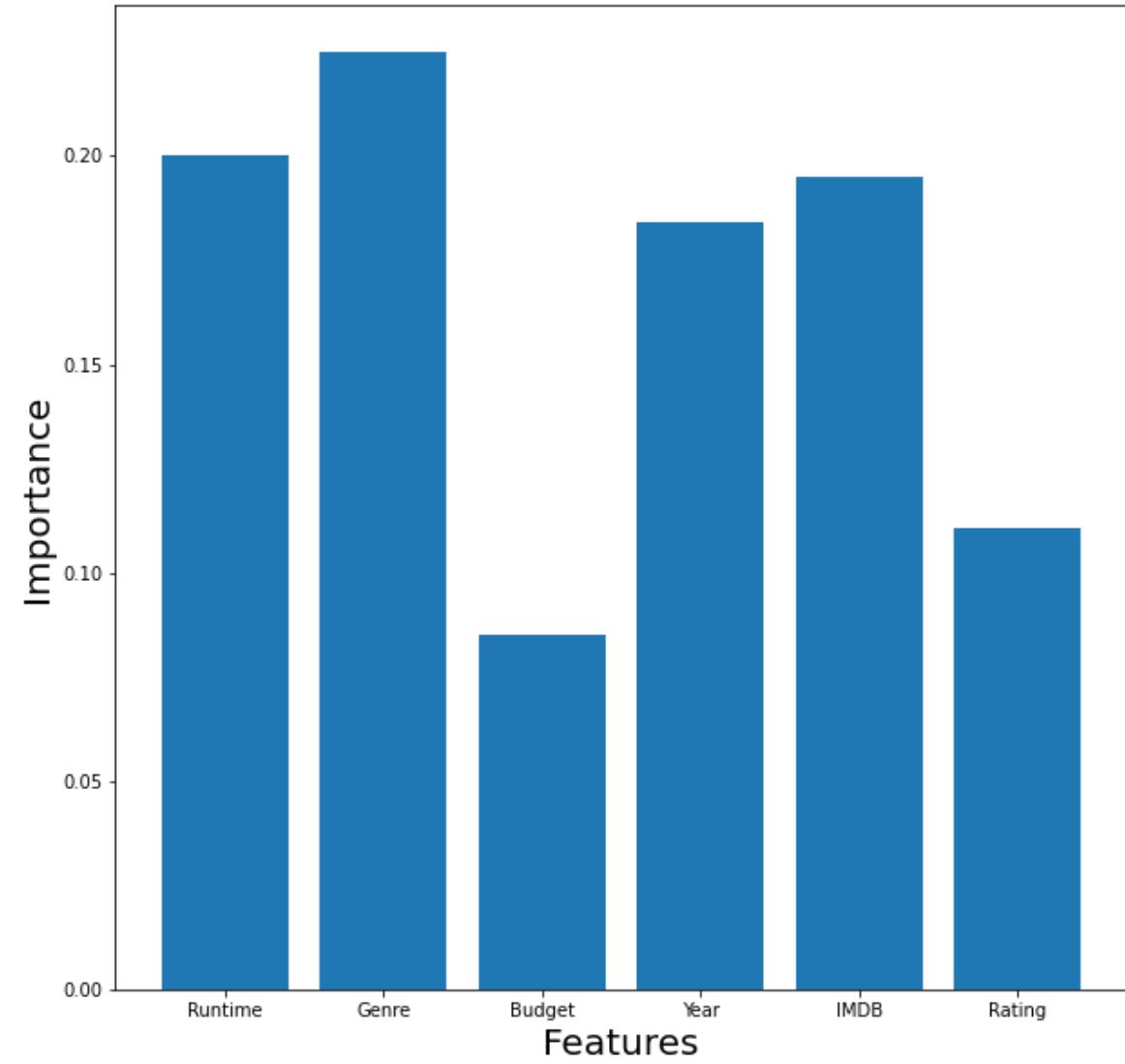


Setting Hyperparameters

Feature Importance

- Some of the interpretability of decision trees gets lost when switching to random forests
- Random forests allow for the computation of “feature importance”, a way of ranking features based on how useful they are at predicting the target
- Initialize each feature’s importance to zero
- Each time a feature is chosen to be split on, add the reduction in IG (weighted by the number of data points in the split) to its importance

Feature Importance



Key Takeaways

- Ensemble methods employ a “wisdom of crowds” philosophy
 - Can reduce the variance of high variance methods
- Random forests = bagging + split-feature randomization
 - Aggregate multiple decision trees together
 - Bootstrapping and split-feature randomization increase diversity in the decision trees
 - Use out-of-bag errors for hyperparameter optimization

Decision Trees: Pros & Cons

- Pros
 - ...
- 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
 - High variance
 - Can be addressed via **bagging** → random forests
 - Limited expressivity/high bias (especially short trees)
 - Can be addressed via **boosting**

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

- Input: \mathcal{D} ($y^{(n)} \in \{-1, +1\}$), 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 :

$$\epsilon_t = \sum_{n=1}^N \omega_{t-1}^{(n)} \mathbb{1}(y^{(n)} \neq h_t(\mathbf{x}^{(n)}))$$

- 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:

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

- Output: an aggregated hypothesis

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

Setting α_t

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

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

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:

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

Intuition: we want good weak learners to have high importances

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

In-class Poll:

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

- Same sign and similar magnitude
- Same sign, different magnitude
- Different sign, similar magnitude
- Different sign, different magnitude

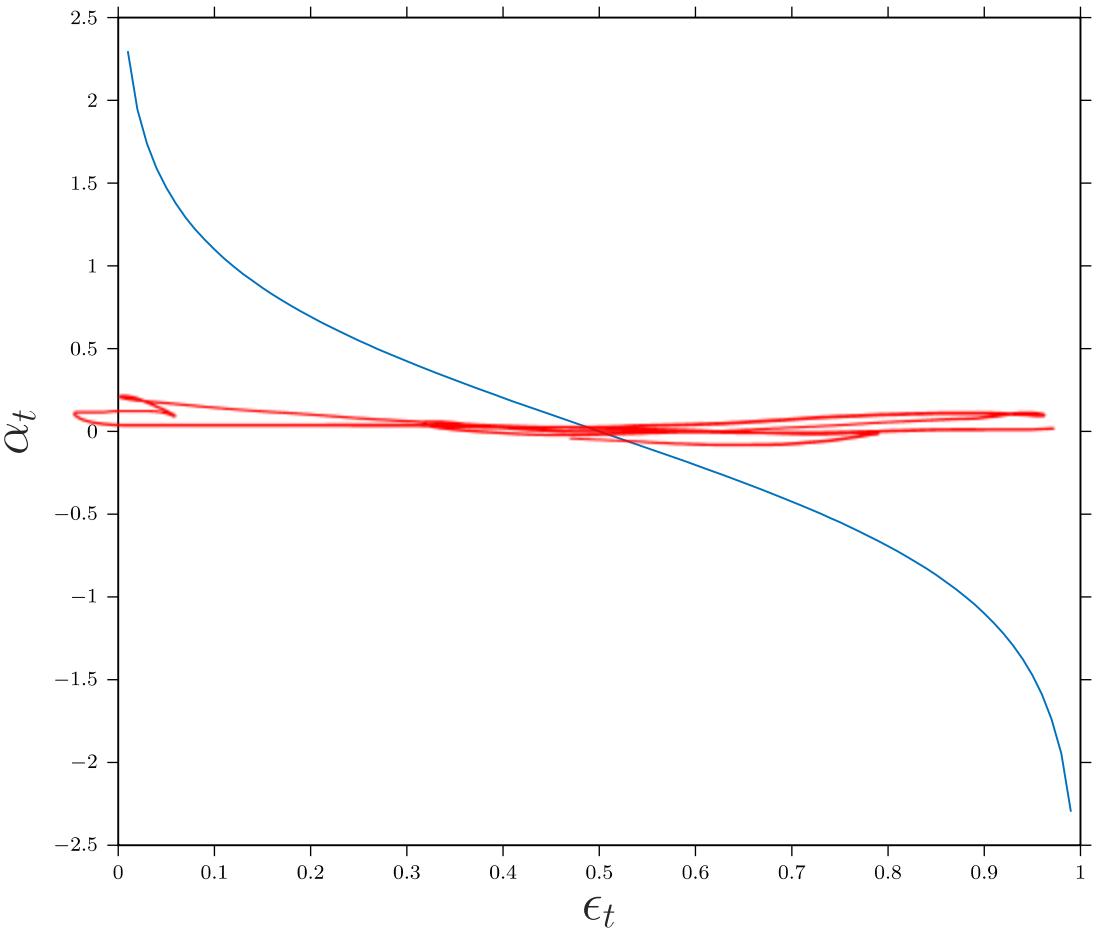
Setting α_t

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

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

Intuition: we want good weak learners to have high importances

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



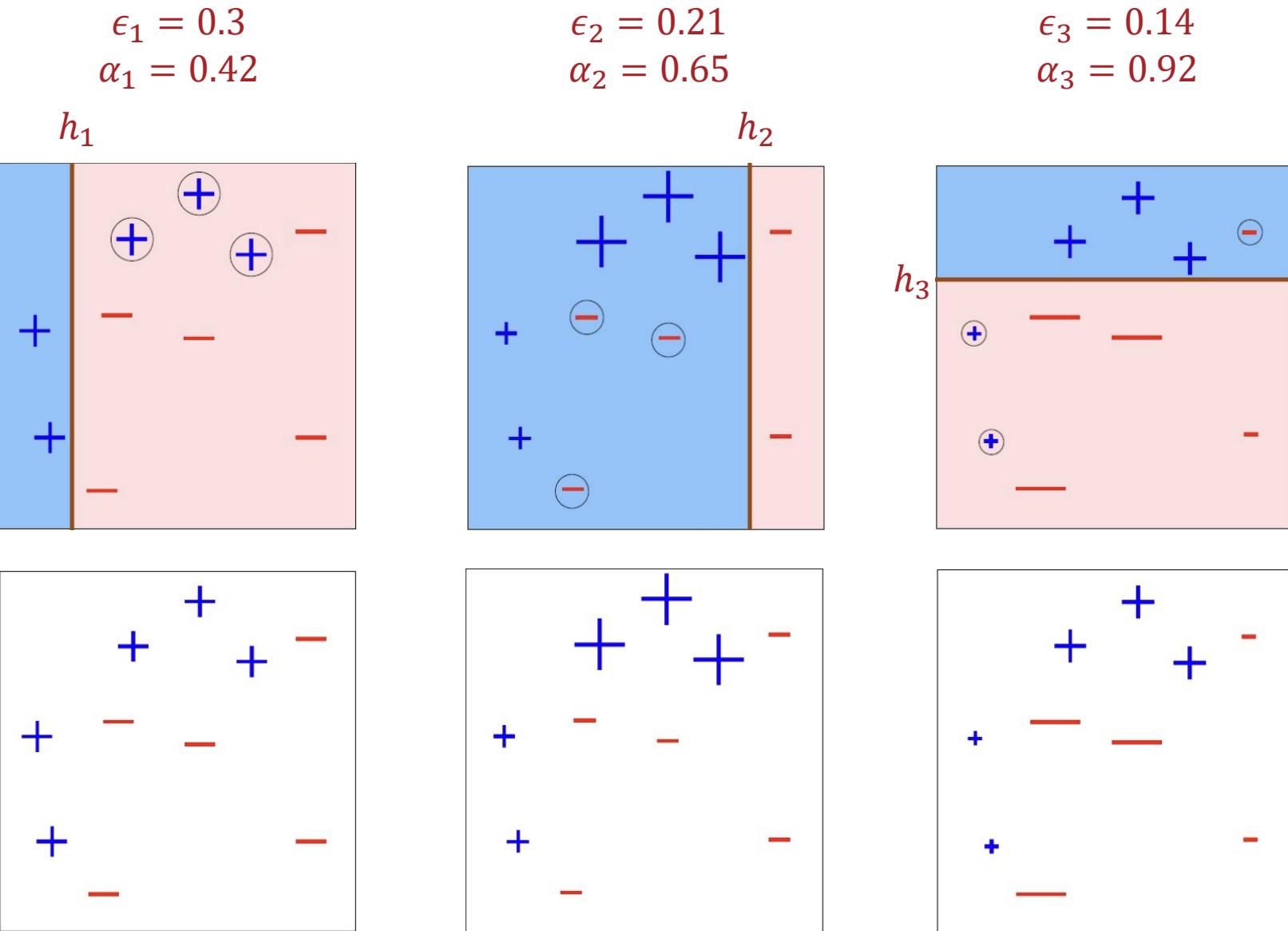
Updating $\omega^{(n)}$

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

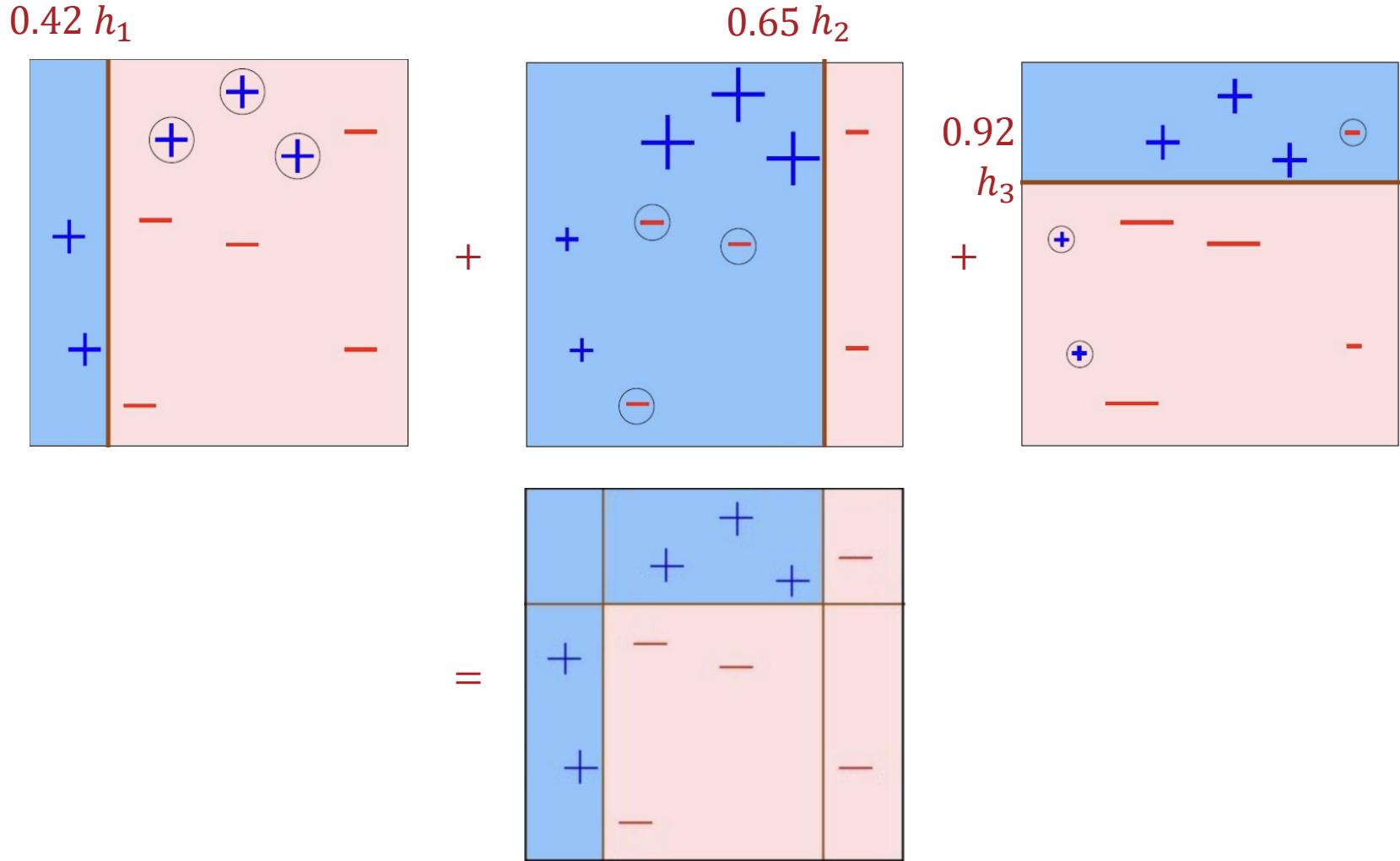
$$\omega_t^{(n)} = \frac{\omega_{t-1}^{(n)}}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(\mathbf{x}^{(n)}) = y^{(n)} \\ e^{\alpha_t} & \text{if } h_t(\mathbf{x}^{(n)}) \neq y^{(n)} \end{cases} = \frac{\omega_{t-1}^{(n)} e^{-\alpha_t y^{(n)} h_t(\mathbf{x}^{(n)})}}{Z_t}$$

- If $\epsilon_t < \frac{1}{2}$, then $\frac{1-\epsilon_t}{\epsilon_t} > 1$
- If $\frac{1-\epsilon_t}{\epsilon_t} > 1$, then $\alpha_t = \frac{1}{2} \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) > 0$
- If $\alpha_t > 0$, then $e^{-\alpha_t} < 1$ and $e^{\alpha_t} > 1$

AdaBoost: Example



AdaBoost: Example



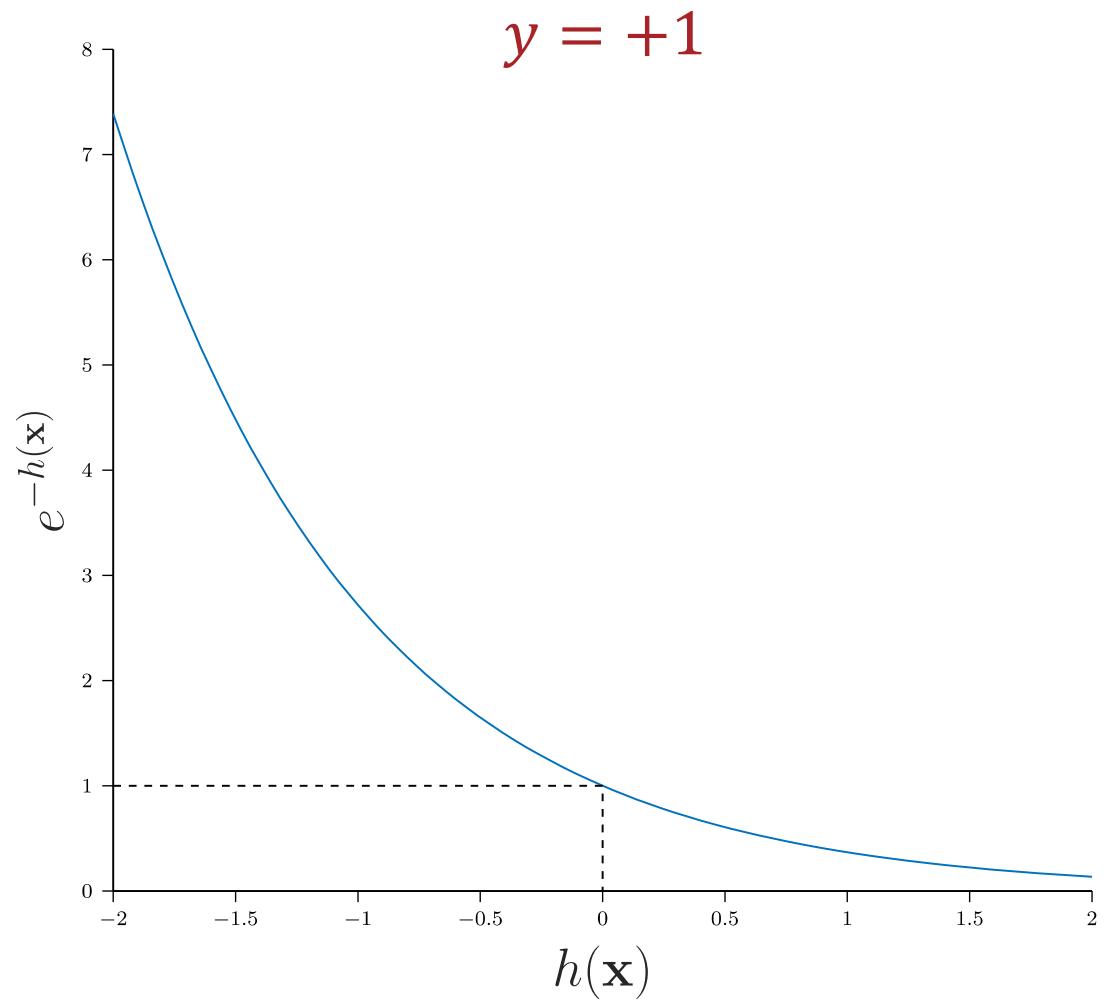
Why AdaBoost?

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:
$$e(h(\mathbf{x}), y) = e^{(-y h(\mathbf{x}))}$$
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

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

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



Exponential Loss

- Claim:

$$\frac{1}{N} \sum_{n=1}^N e^{(-y^{(n)} h(\mathbf{x}^{(n)}))} \geq \frac{1}{N} \sum_{n=1}^N \mathbb{1}(\text{sign}(h(\mathbf{x}^{(n)})) \neq y^{(n)})$$

- Consequence:

$$\frac{1}{N} \sum_{n=1}^N e^{(-y^{(n)} h(\mathbf{x}^{(n)}))} \rightarrow 0$$

$$\Rightarrow \frac{1}{N} \sum_{n=1}^N \mathbb{1}(\text{sign}(h(\mathbf{x}^{(n)})) \neq y^{(n)}) \rightarrow 0$$

Key Takeaways

- Boosting targets simple models, i.e., weak learners
- Greedily minimizes the exponential loss, an upper bound of the classification error
- Theoretical (and empirical) results show resilience to overfitting by targeting training margin