10-701: Introduction to Machine Learning Lecture 22 – Ensembles

Pradeep Ravikumar & Geoff Gordon

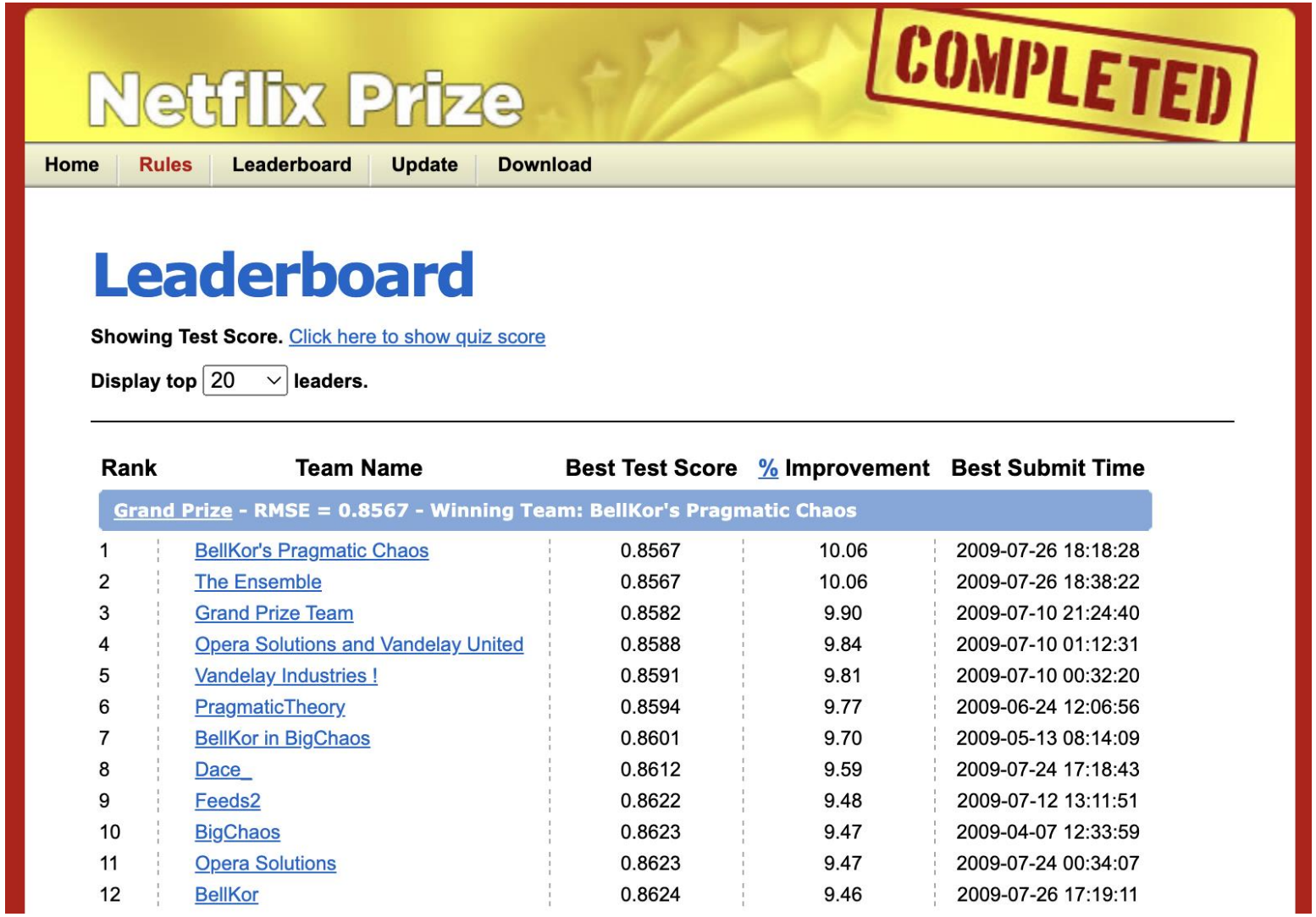
Spring 2026

Front Matter

- Announcements
  - Project check-ins due on 4/15 (extension!) at 11:59 PM

The Netflix Prize

The Netflix Prize



Source: https://web.archive.org/web/20090926213457/http://www.netflixprize.com/leaderboard

Historical summary of the 2009 Netflix Prize finish

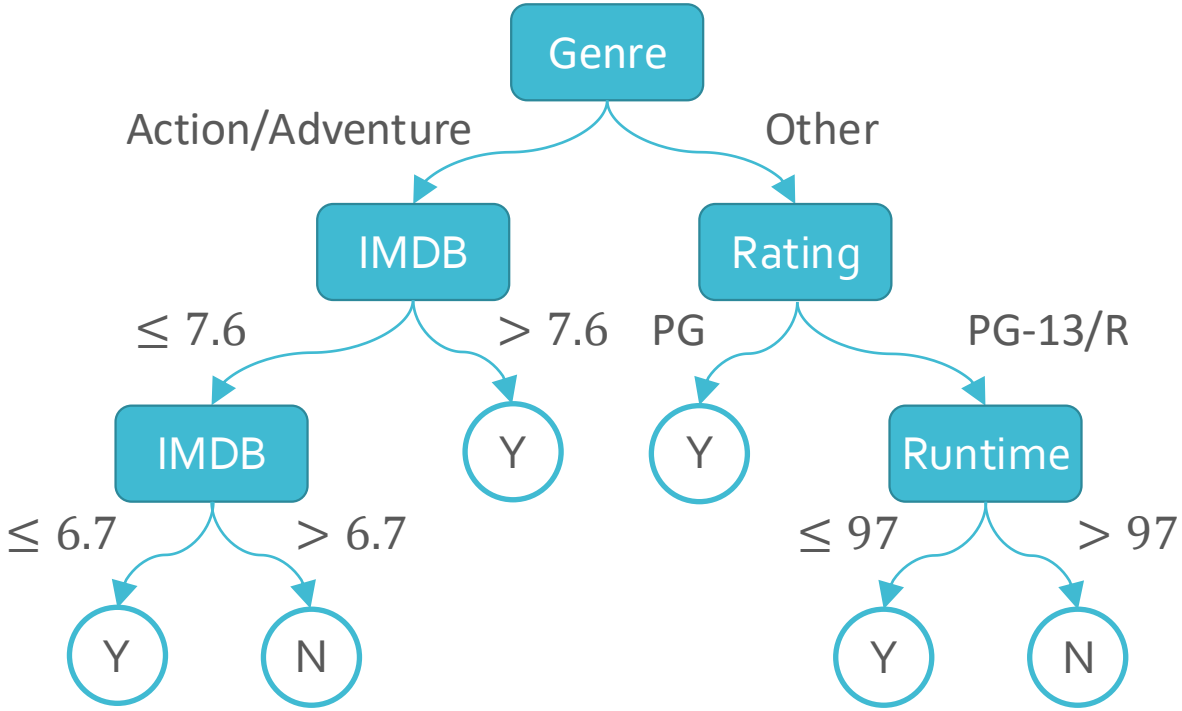
- By the end, the top teams were merging and blending
  - Each team itself was already blending many different recommenders, and not betting on one single model.
- Grand Prize Winner Journey: BellKor (AT&T team) → BellKor in BigChaos (BellKor + BigChaos) → BellKor's Pragmatic Chaos (BellKor in BigChaos + Pragmatic Theory)
- Runner Up: Opera Solutions + Vandelay United
  - matched the winner but submitted later
- Why did they “ensemble”?
  - The topic of today’s lecture

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

Movie Recommendations

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	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



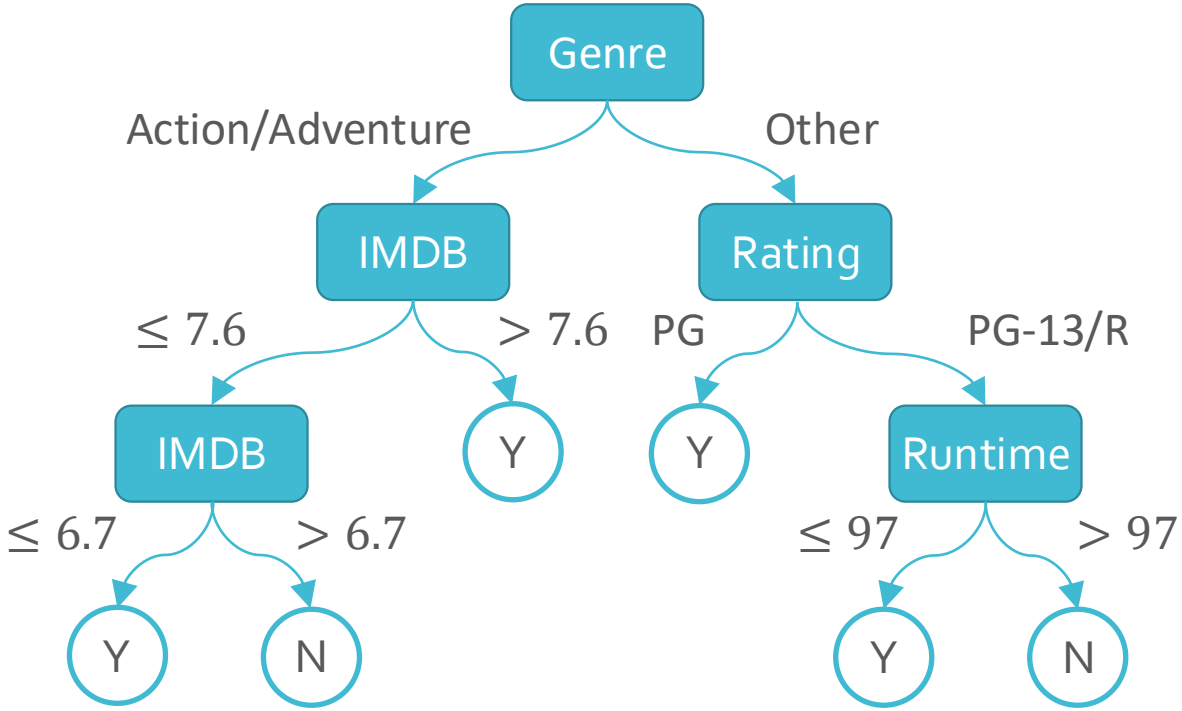
Decision Trees

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

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

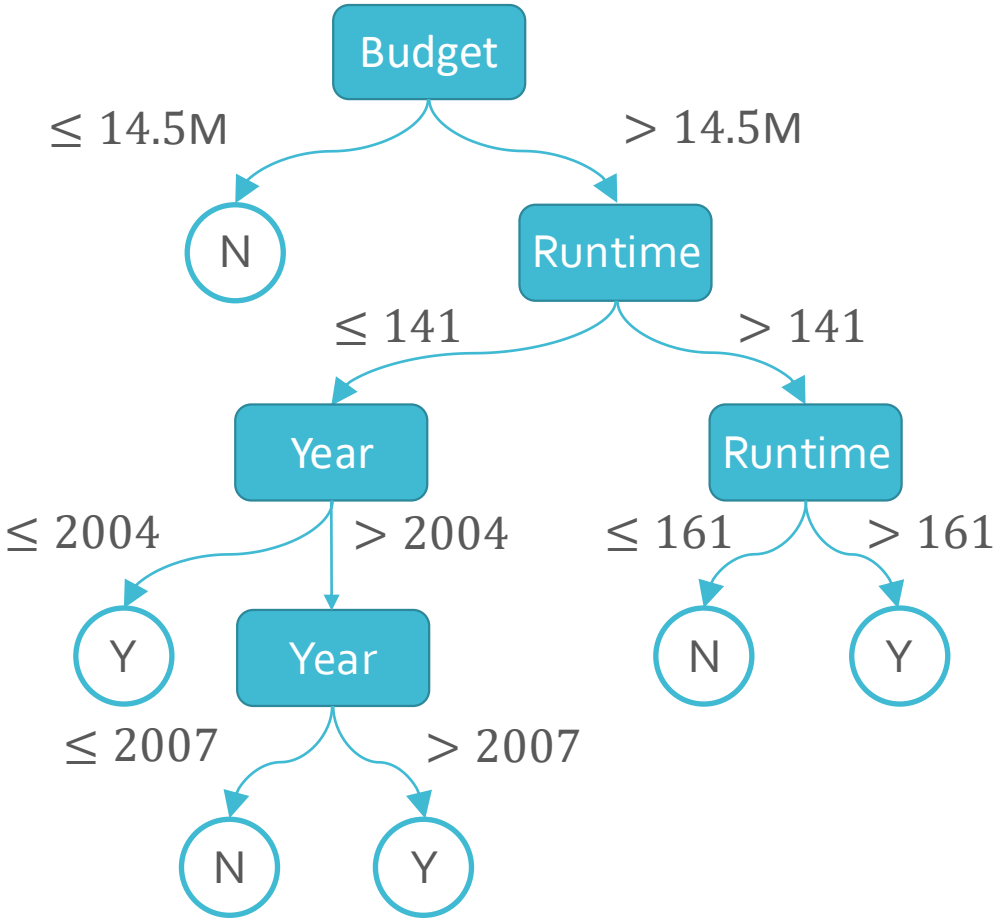
MovieID	Runtime	Genre	Budget	Year	IMDB	Rating	Liked?
1	124	Action	18M	1980	8.7	PG	Y
2	105	Action	30M	1984	7.8	PG	Y
3	103	Comedy	6M	1986	7.8	PG-13	N
4	98	Adventure	16M	1987	8.1	PG	Y
5	128	Comedy	16.4M	1989	8.1	PG	Y
6	120	Comedy	11M	1992	7.6	R	N
7	120	Drama	14.5M	1996	6.7	PG-13	N
8	136	Action	115M	1999	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

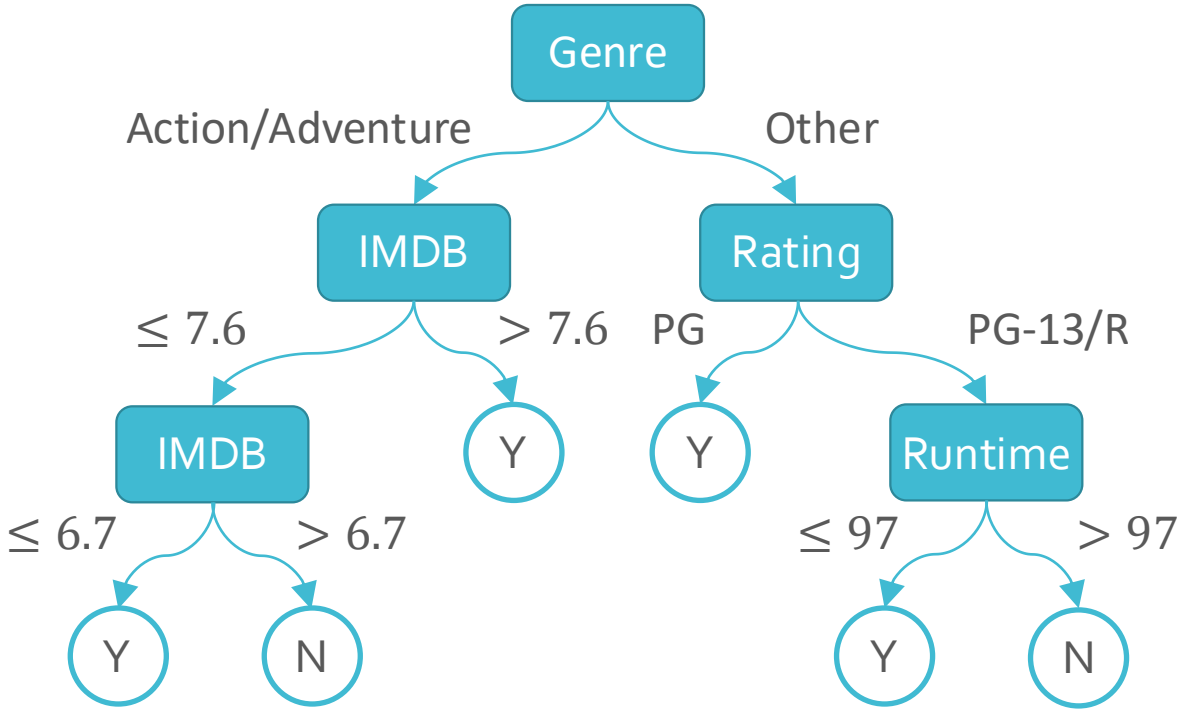
Source: https://www.kaggle.com/datasets/danielgrijalvas/movies

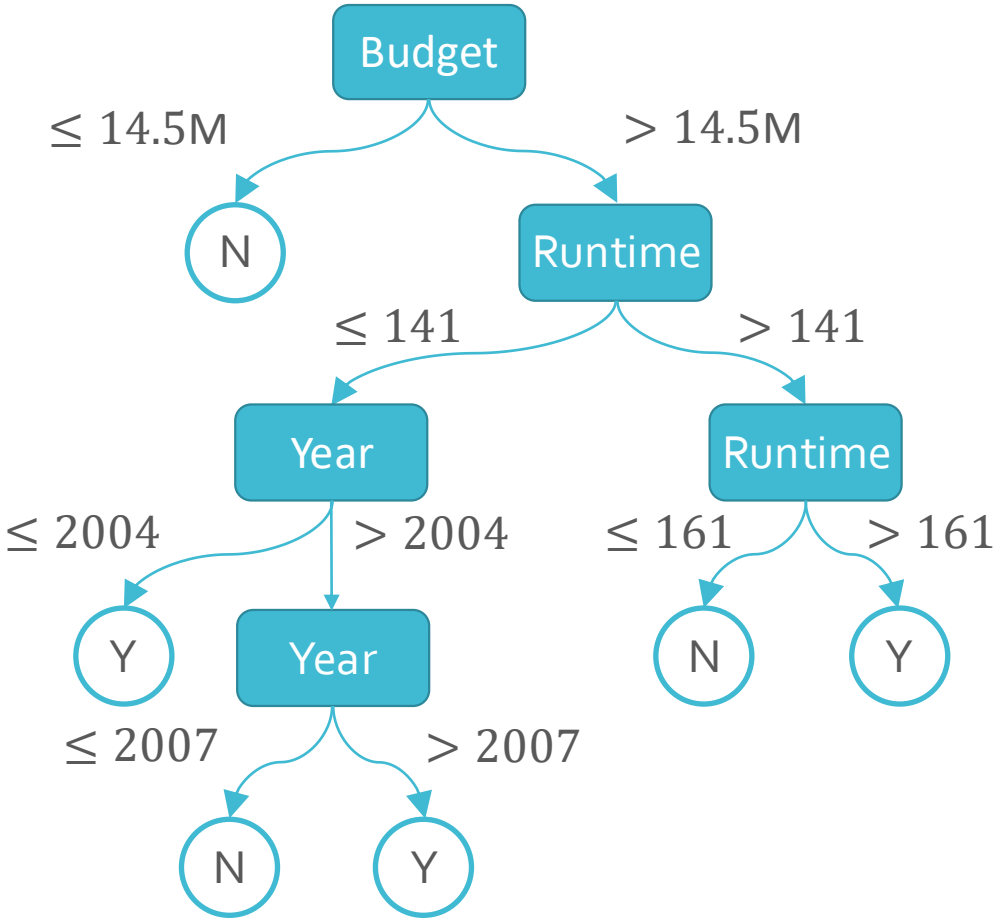
MovieID	Runtime	Genre	Budget	Year	IMDB	Rating	Liked?
1	124	Action	18M	1980	8.7	PG	Y
2	105	Action	30M	1984	7.8	PG	Y
3	103	Comedy	6M	1986	7.8	PG-13	N
4	98	Adventure	16M	1987	8.1	PG	Y
5	128	Comedy	16.4M	1989	8.1	PG	Y
6	120	Comedy	11M	1992	7.6	R	N
7	120	Drama	14.5M	1996	6.7	PG-13	N
8	136	Action	115M	1999	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



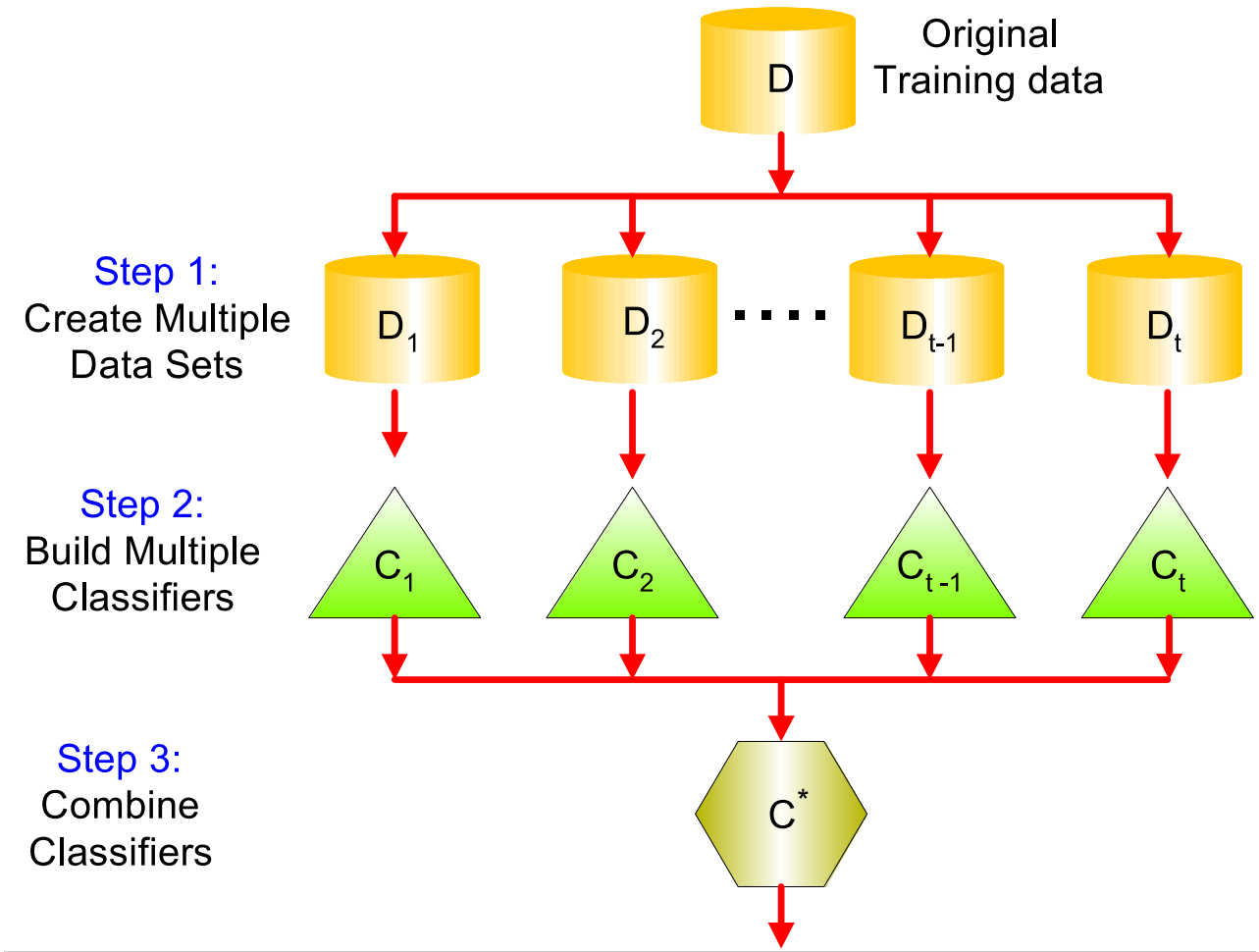


Decision Trees

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

Ensembles


D
Original
Training data
Step 1:
Create Multiple
Data Sets
D1
D2
Dt-1
Dt
Step 2:
Build Multiple
Classifiers
C1
C2
Ct -1
Ct
Step 3:
Combine
Classifiers
C*

Power of Ensembles

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

Power of Ensembles

- Suppose there are 25 base classifiers
  - Each classifier has error rate, $\varepsilon = 0.35$
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

Bagging: a simpler ensemble approach

- Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample

Final Prediction: Majority vote over ensemble of classifiers

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

Source: https://galton.org/essays/1900-1911/galton-1907-vox-populi.pdf

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

  = **b**ootstrap **agg**regat**ing** + split-feature randomization

Aggregating

- How can we combine multiple decision trees, $\{t_1, t_2, \ldots, t_B\}$, 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 $\{-1, +1\}$:

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

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

- 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

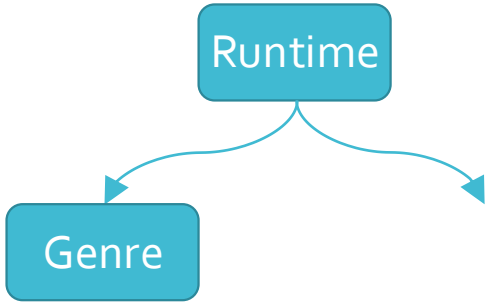
Runtime	Genre	Budget	Year	IMDB	Rating

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

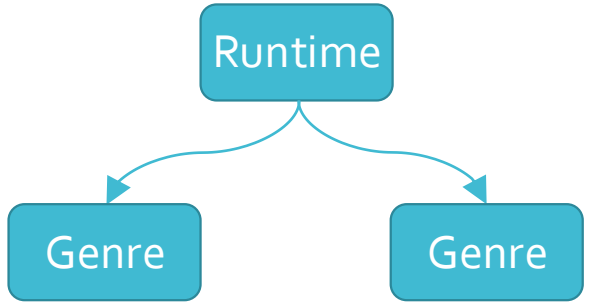
Split-feature Randomization

- 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



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

How can we set B and ρ?

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

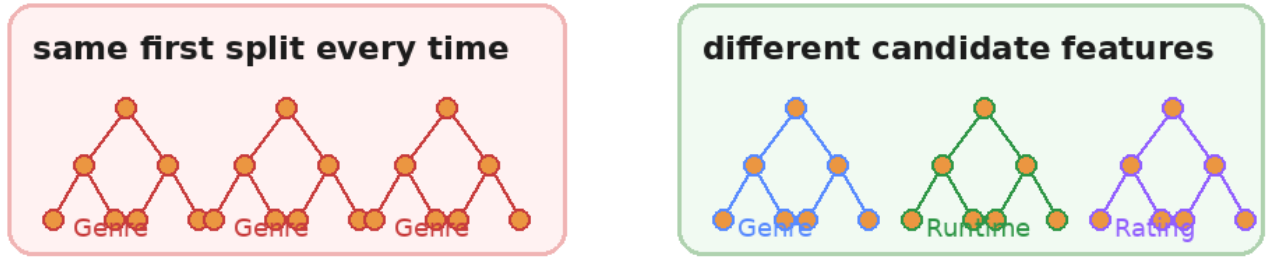
Myth Busters

MythBusters #1

“Random feature sampling is just a speed trick.”

Myth: choose only ρ features at each split mainly to make training cheaper.

Reality: the lecture’s point is diversity. If every tree keeps choosing the same strong feature first, the forest behaves too similarly and averaging helps less.



Random rows + random split features = less correlation among trees.

Recall: Validation Sets

- Suppose we want to compare multiple hyperparameter settings $\theta_1, \ldots, \theta_K$
- For $k = 1, 2, \ldots, K$
  - 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 $t^{(-n)} = \left\{t_1^{(-n)}, t_2^{(-n)}, \ldots, t_{N-n}^{(-n)}\right\}$
- 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 $t^{(-n)} = \left\{t_1^{(-n)}, t_2^{(-n)}, \ldots, t_{N-n}^{(-n)}\right\}$
- 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!

Quick Game

Quick game: who is out-of-bag?

For point ID = 3, which trees can be used to form its OOB prediction?

Original data	Tree 1 sample	Tree 2 sample	Tree 3 sample
1	1	2	1
2	1	3	2
3	2	3	2
4	4	4	4
5	4	6	5
6	5	6	5
7	7	7	7
8	8	8	8

Vote

A. Tree 2 only

B. Trees 1 and 3 only

C. All three trees

D. None of the trees

Quick Game

Quick game: who is out-of-bag?

For point ID = 3, which trees can be used to form its OOB prediction?

Original data	Tree 1 sample	Tree 2 sample	Tree 3 sample
1	1	2	1
2	1	3	2
3	2	3	2
4	4	4	4
5	4	6	5
6	5	6	5
7	7	7	7
8	8	8	8

Vote

A. Tree 2 only

B. Trees 1 and 3 only

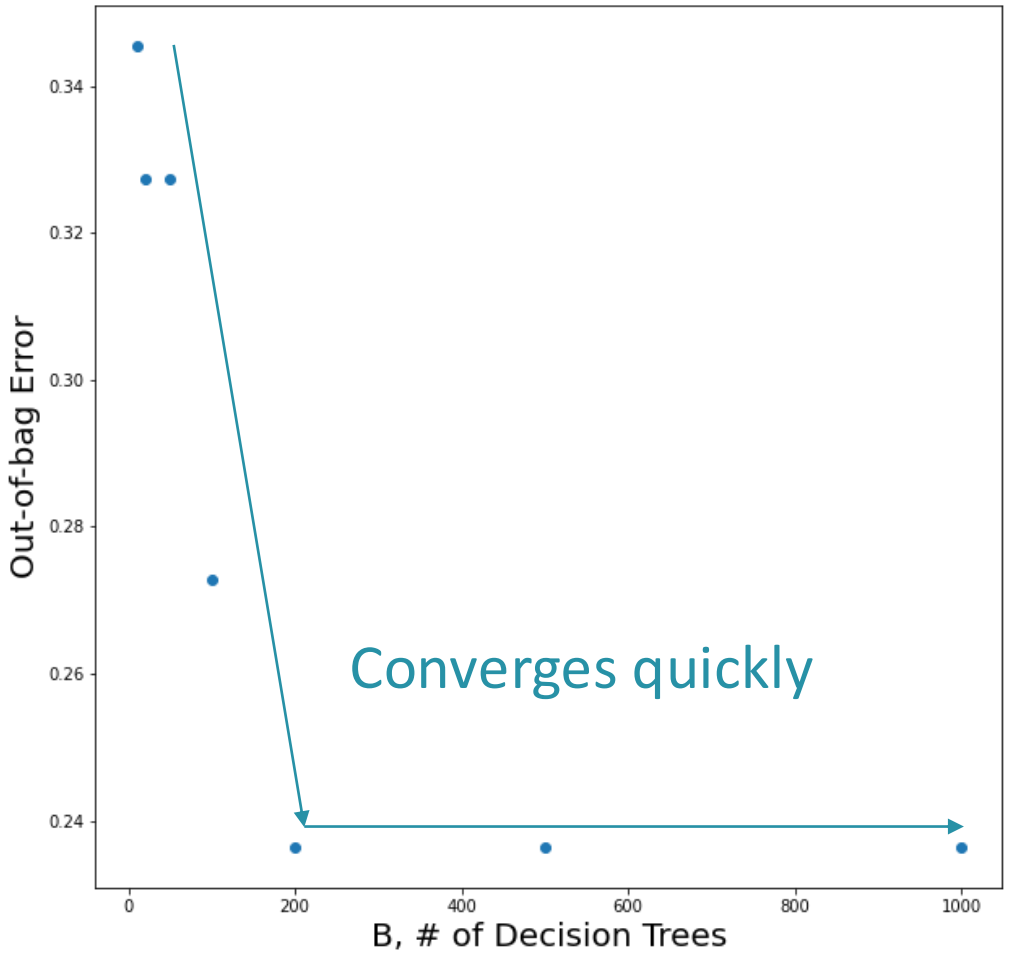
C. All three trees

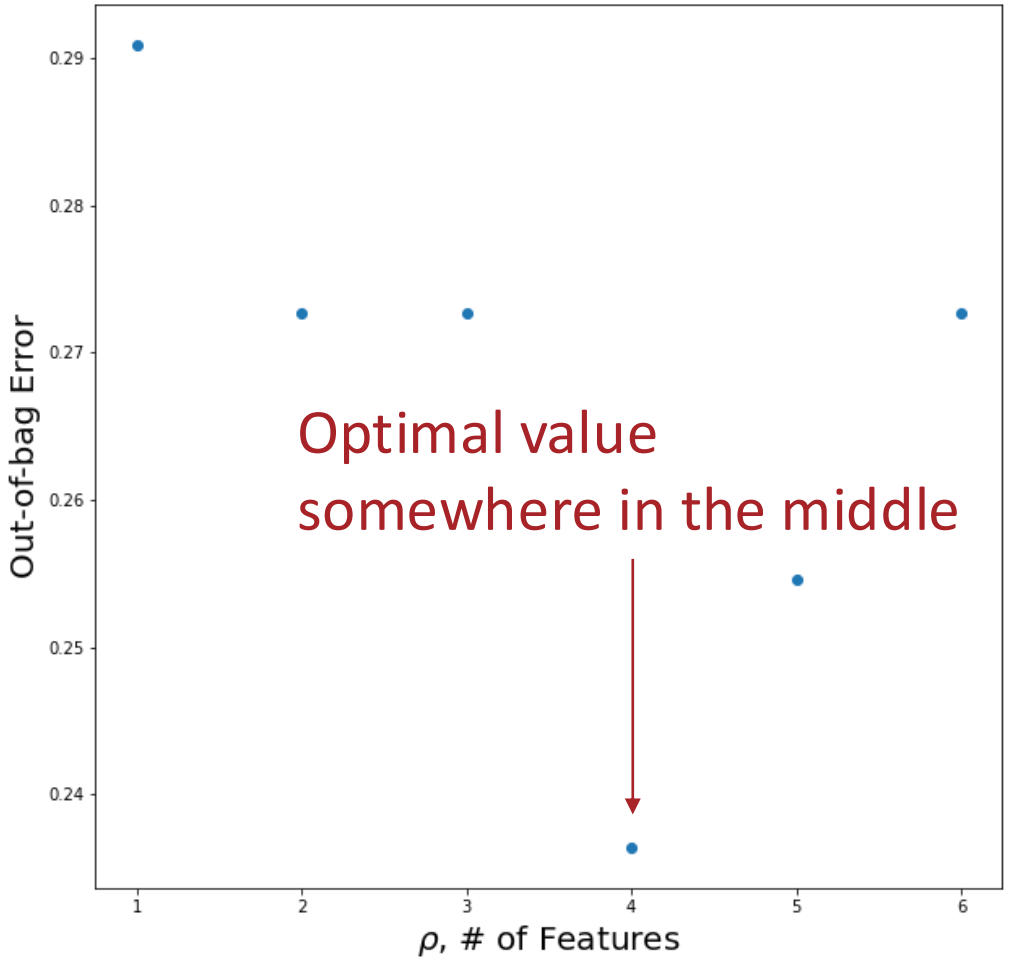
D. None of the trees

Reveal: B. OOB prediction for point 3 uses only trees whose bootstrap sample omitted point 3.

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}



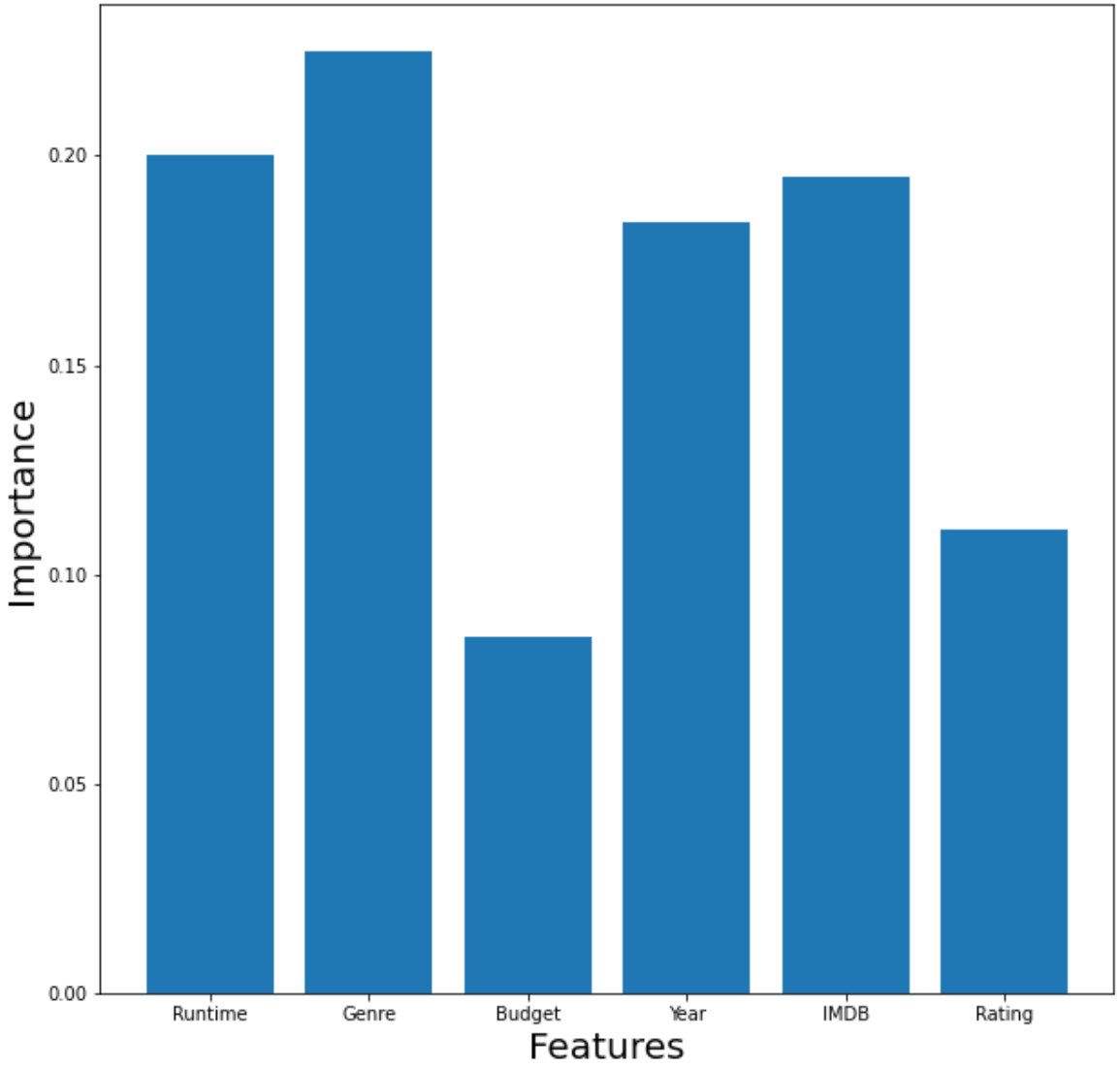


Setting Hyperparameters

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 Gini impurity (weighted by the number of data points in the split) to its importance

Feature Importance



Quick Game

Two truths and a lie (random forests)

Vote for the lie: A, B, or C.

A. Bagging changes the rows by resampling the training data with replacement.

B. Random forests add feature subsampling at each split to increase diversity among trees.

C. Out-of-bag error needs a separate validation set, because every tree trains on every point eventually.

Quick Game

Two truths and a lie (random forests)

Vote for the lie: A, B, or C.

A. Bagging changes the rows by resampling the training data with replacement.

B. Random forests add feature subsampling at each split to increase diversity among trees.

C. Out-of-bag error needs a separate validation set, because every tree trains on every point eventually.

Reveal: C is the lie. OOB uses the trees that omitted a point, so it gives a built-in validation signal.

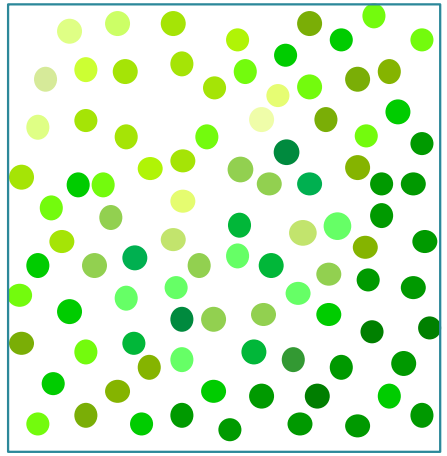
Local Ensembles

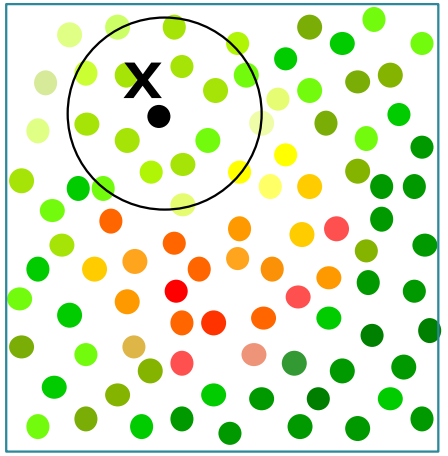
- Even a single decision tree can be thought of as a “local ensemble” i.e. an ensemble of very “local” classifiers
- This requires formalizing notion of “local” via “smoothing kernels”

Local Ensemble/ Regression

- What is the temperature in the room?

at location x?





$$\hat{T} = \frac{1}{n} \sum_{i=1}^{n} Y_i$$

Average

$$\hat{T}(x) = \frac{\sum_{i=1}^{n} Y_i 1_{||X_i - x|| \leq h}}{\sum_{i=1}^{n} 1_{||X_i - x|| \leq h}}$$

"Local" Average

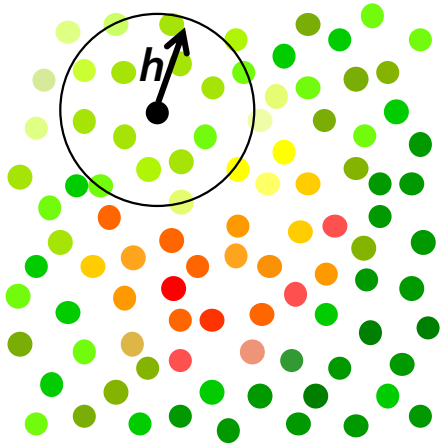
Local Average Regression

$$\Rightarrow \widehat{f}_n(X) = \widehat{\beta} = \sum_{i=1}^{n} w_i Y_i$$

$$= \frac{1}{n_X^h} \sum_{i=1}^{n} Y_i \; 1_{|X - X_i| \le h}$$

#pts in h ball around X

Sum of Ys in h ball around X



Nadaraya-Watson Kernel Regression

$$\Rightarrow \hat{f}_n(X) = \hat{\beta} = \sum_{i=1}^{n} w_i Y_i$$

with box-car kernel

$$= \frac{1}{n_X^h} \sum_{i=1}^{n} Y_i \, 1_{|X - X_i| \le h}$$

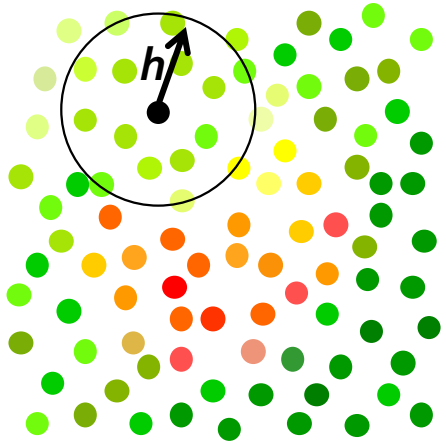
#pts in h ball around X

Sum of Ys in h ball around X

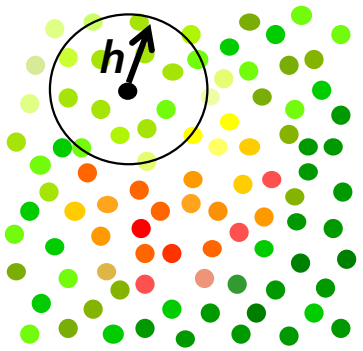
$$w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

boxcar kernel :

$$K\left(\frac{X - X_i}{h}\right) = 1_{|X - X_i| \le h}$$



Local Kernel Regression



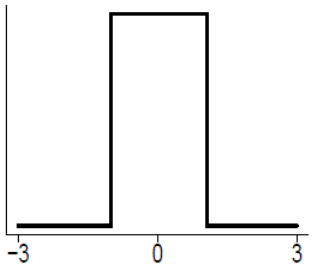
- Nonparametric estimator akin to kNN
- Nadaraya-Watson Kernel Estimator

$$\hat{f}_n(X) = \sum_{i=1}^{n} w_i Y_i \quad \text{where} \quad w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

- Weight each training point based on distance to test point
- Boxcar kernel yields

  local average

$$\textbf{boxcar kernel}: \quad K(x) = \frac{1}{2} I(x),$$



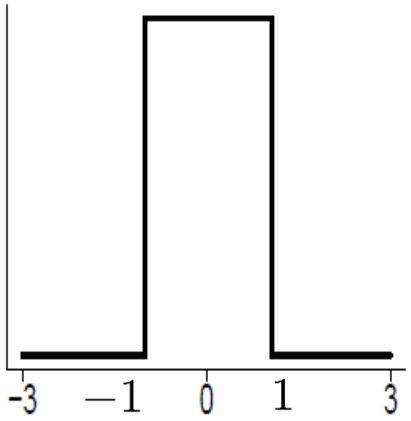
Smoothing Kernels

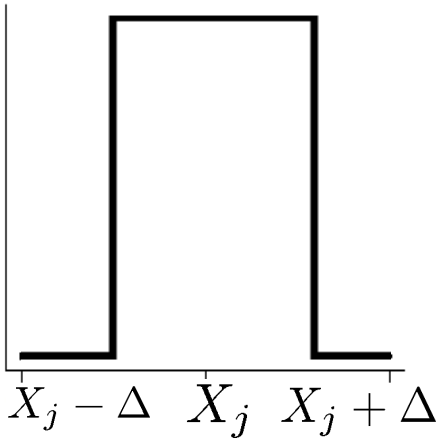
$$K(x) \geq 0,$$
$$\int K(x)dx = 1$$

$$K\left(\frac{X_j - x}{\Delta}\right)$$

boxcar kernel :

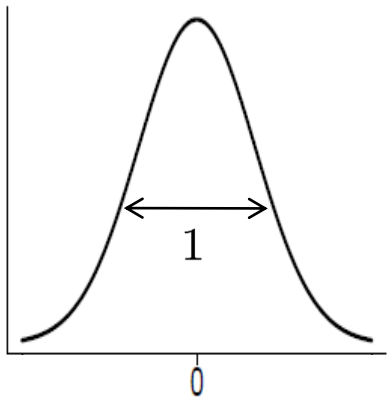
$$K(x) = \frac{1}{2}I(x),$$

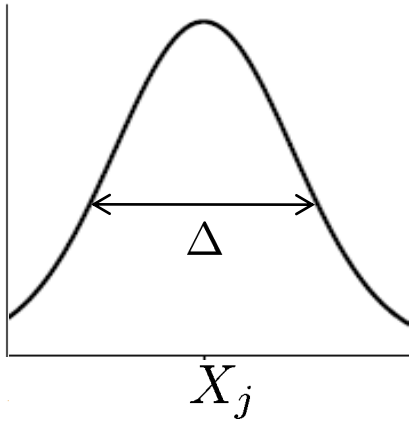




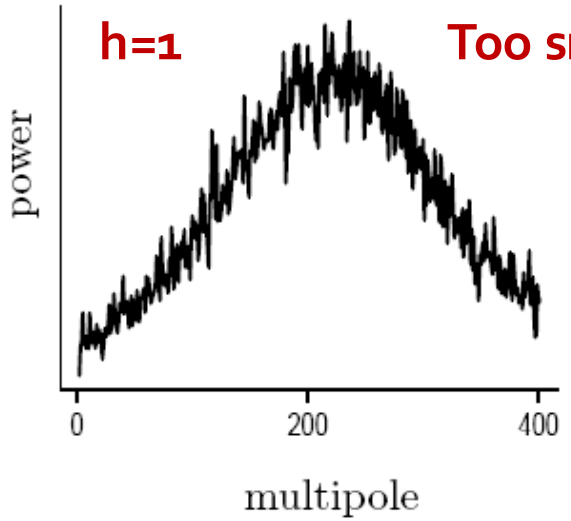
Gaussian kernel :

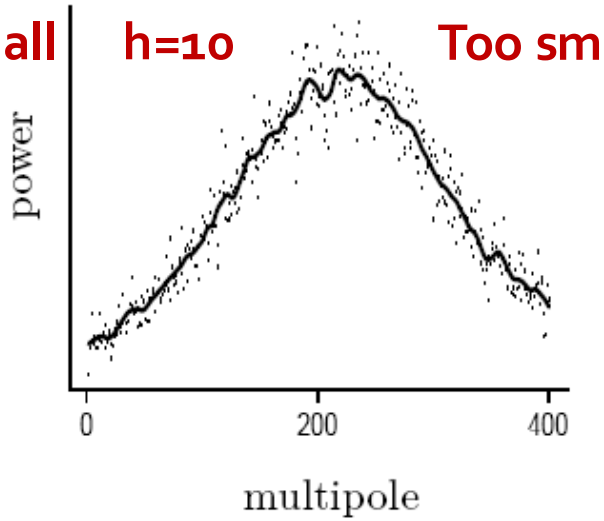
$$K(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$$

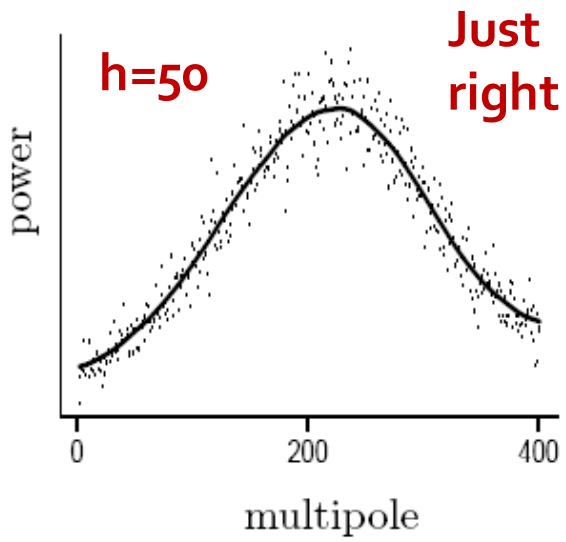




Choice of kernel bandwidth h







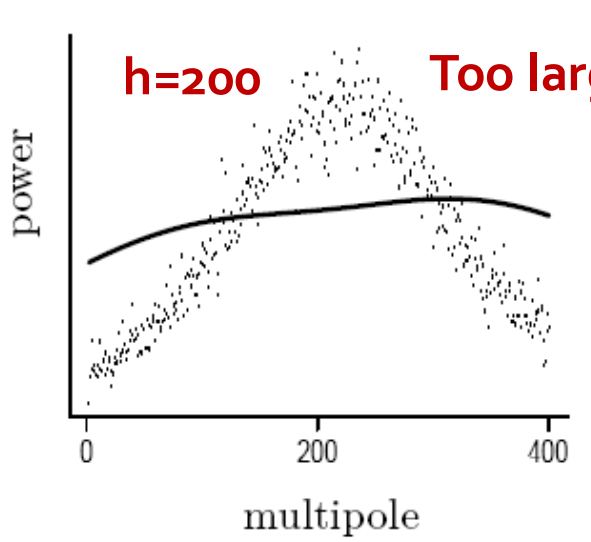


Image Source: Larry's book – All of Nonparametric Statistics

Choice of kernel is not that important

Kernel Regression as Weighted Least Squares

$$\min_f \frac{1}{n}\sum_{i=1}^{n} w_i(f(X_i) - Y_i)^2 \qquad \frac{1}{n}\sum_{i=1}^{n} w_i = 1$$

Weighted Least Squares

Weigh each training point based on distance to test point

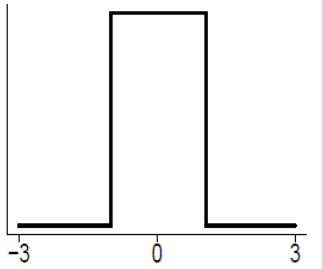
$$w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$

K – Kernel

h – Bandwidth of kernel

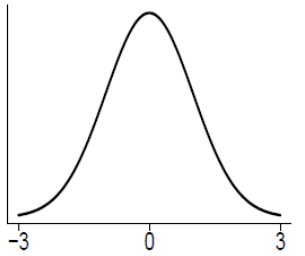
boxcar kernel :

$$K(x) = \frac{1}{2}I(x),$$



Gaussian kernel :

$$K(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$$



Kernel Regression as Weighted Least Squares

set $f(X_i) = \beta$ (a constant)

$$\min_{\beta} \sum_{i=1}^{n} w_i(\beta - Y_i)^2$$

β: constant

$$w_i(X) = \frac{K\left(\frac{X - X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X - X_i}{h}\right)}$$

$$\frac{\partial J(\beta)}{\partial \beta} = 2\sum_{i=1}^{n} w_i(\beta - Y_i) = 0$$

Notice that $\sum_{i=1}^{n} w_i = 1$

$$\Rightarrow \widehat{f}_n(X) = \widehat{\beta} = \sum_{i=1}^{n} w_i Y_i$$

Local Linear/ Polynomial Regression

$$\min_f \sum_{i=1}^{n} w_i(f(X_i) - Y_i)^2 \qquad w_i(X) = \frac{K\left(\frac{X-X_i}{h}\right)}{\sum_{i=1}^{n} K\left(\frac{X-X_i}{h}\right)}$$

Weighted Least Squares

Local Polynomial regression corresponds to locally polynomial estimator obtained from (locally) weighted least squares

$$f(X_i) = \beta_0 + \beta_1(X_i - X) + \frac{\beta_2}{2!}(X_i - X)^2 + \cdots + \frac{\beta_p}{p!}(X_i - X)^p$$

i.e. set

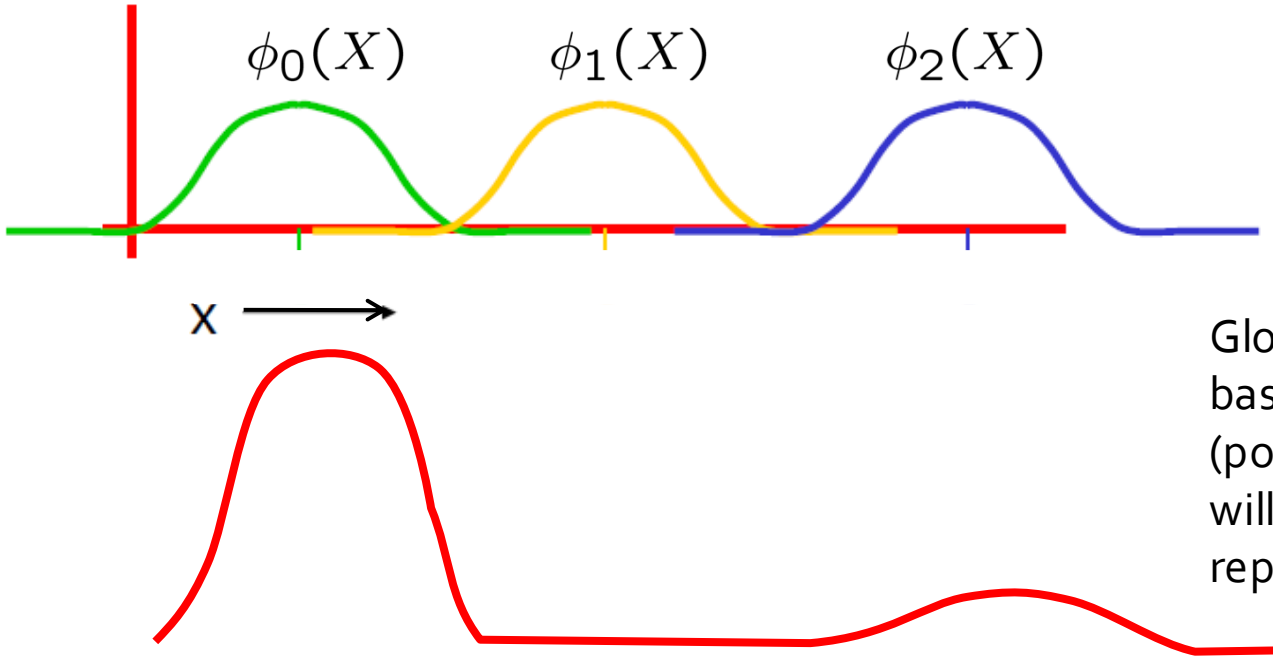
(local polynomial of degree p around X)

Local Regression

$$f(X) = \sum_{j=0}^{m} \beta_j \phi_j(X)$$

Basis coefficients

Nonlinear features/basis functions



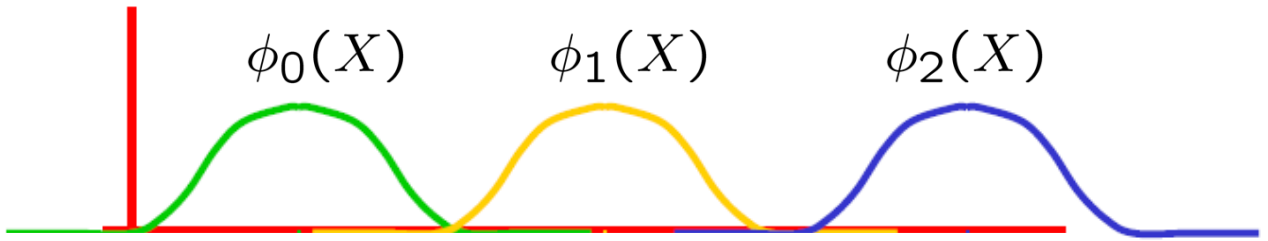
Globally supported basis functions (polynomial, fourier) will not yield a good representation

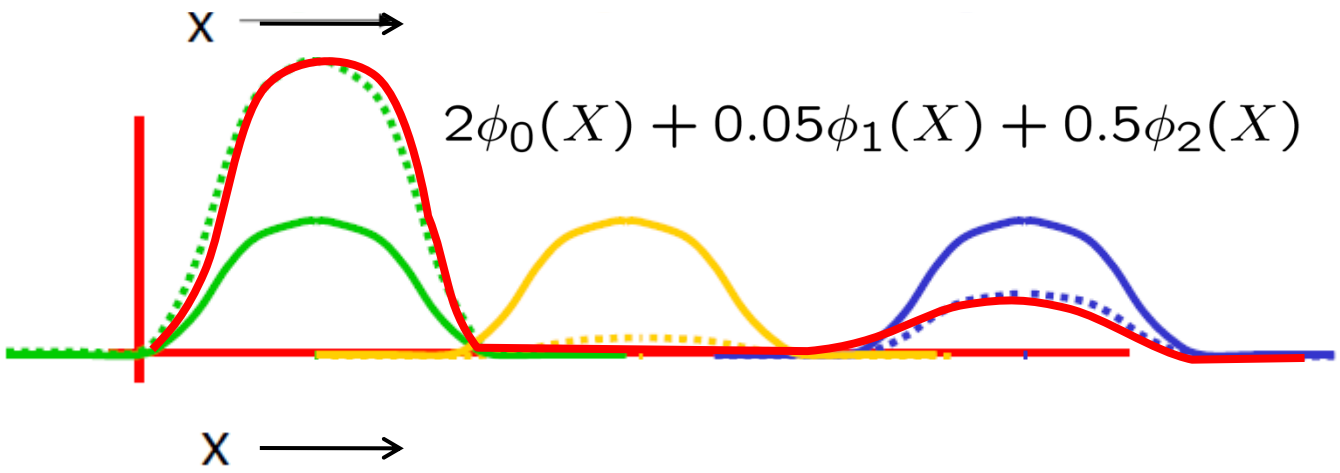
Local Regression

$$f(X) = \sum_{j=0}^{m} \beta_j \phi_j(X)$$

Basis coefficients

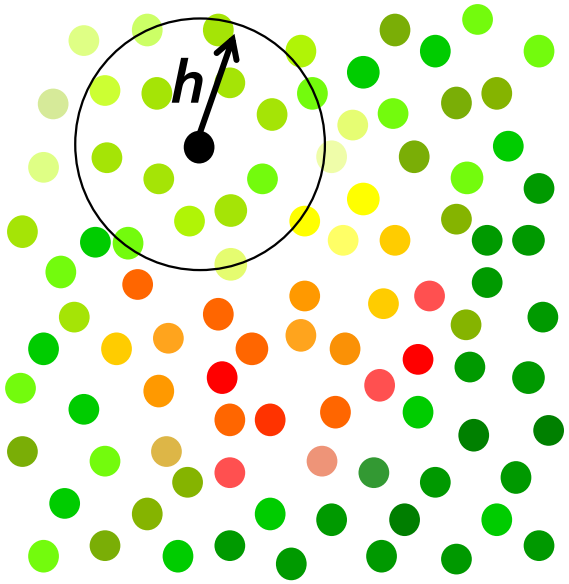
Nonlinear features/basis functions





Globally supported basis functions (polynomial, fourier) will not yield a good representation

Choice of Bandwidth



Should depend on n, # training data
(determines variance)

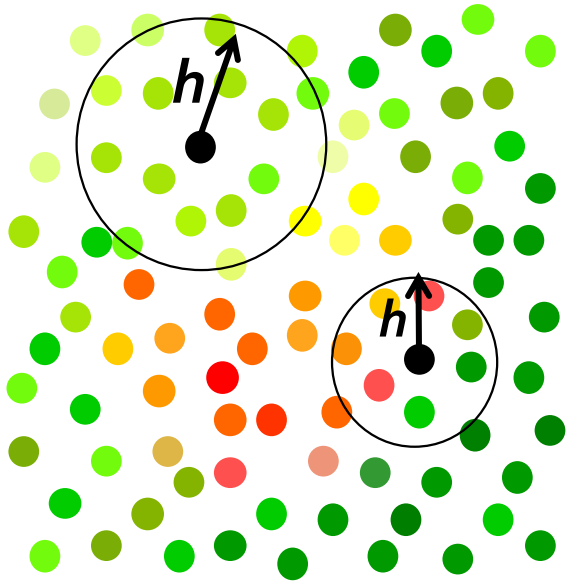
Should depend on smoothness of function
(determines bias)

Large Bandwidth – average more data points, reduce noise (Lower variance)

Small Bandwidth – less smoothing, more accurate fit (Lower bias)

Bias – Variance tradeoff

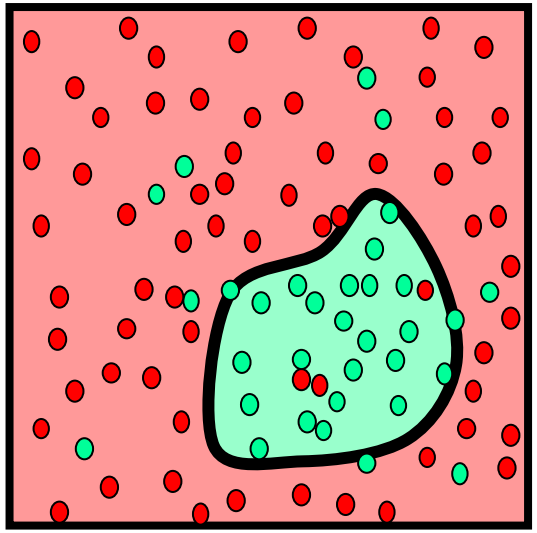
Spatially adaptive regression

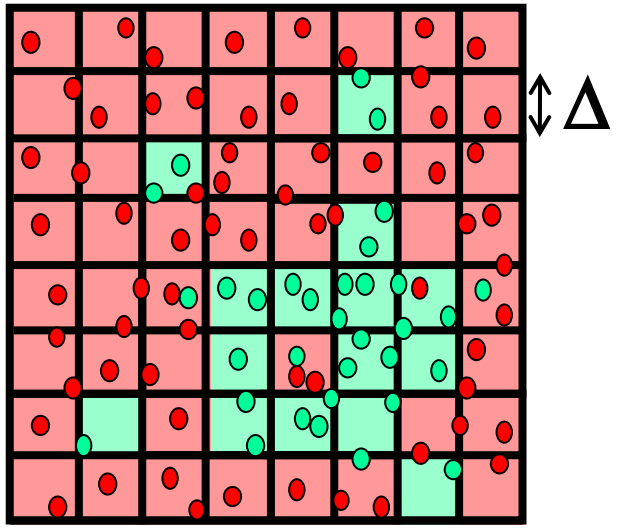


If function smoothness varies spatially, we want to allow bandwidth h to depend on X

Local polynomials, splines, wavelets, regression trees …

Histogram/ Partition Classifier





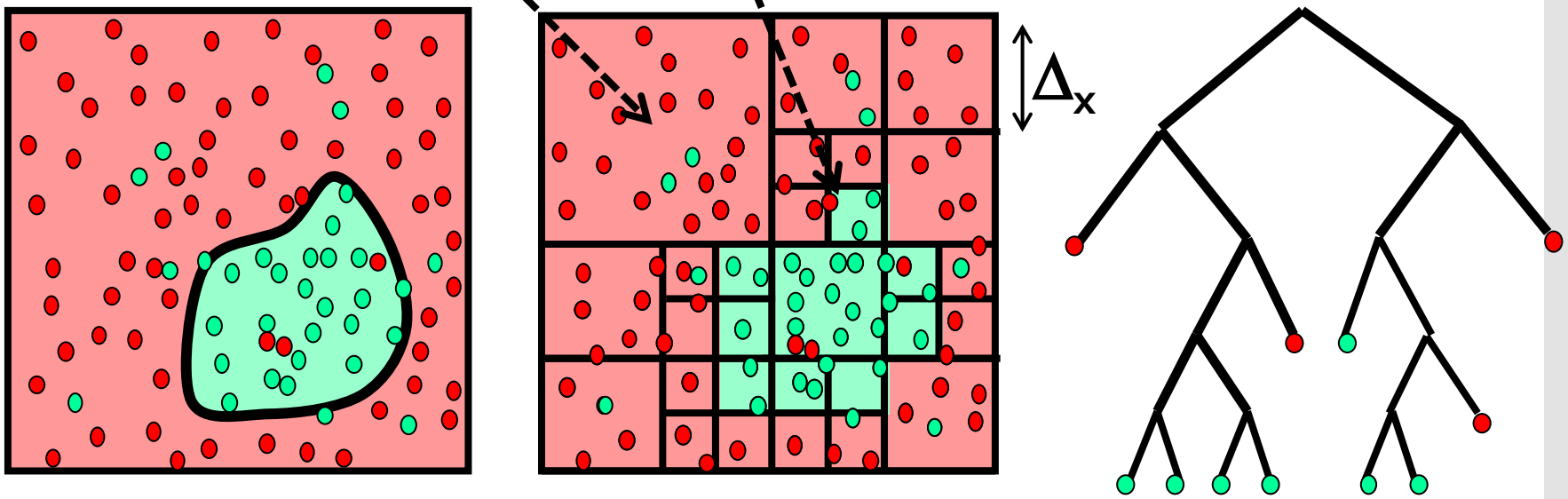
Histogram Classifier

Partition input space into equal sized bins

In each bin, predict average of training responses that fell in that bin

Adaptive Partitions

Let neighborhood size adapt to data – small neighborhoods near decision boundary (small bias), large neighborhoods elsewhere (small variance)



Majority vote at each leaf

Decision Tree Classifier

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

Final Poll

Poll:

Final poll (pick one)

Which statement best matches the lecture’s main message about random forests?

A. Random forests = pruning + boosting.

B. Split-feature randomization is mostly a computational shortcut and has little effect on diversity.

C. OOB error predicts each training point using only trees that did not train on it, so it can help tune hyperparameters without a separate validation set.

D. Averaging identical trees is enough; bootstrapping is optional.

E. Feature importance is computed by ranking features according to validation accuracy only.