Machine Learning and Data Mining

Lecture 8.1: Random Forests and Ensemble Methods

CS 434

CS499: Introduction to Information Retrieval

Oregon State University
College of Engineering

Instructor: Huazheng Wang

Assistant Professor of Computer Science

huazheng.wang@oregonstate.edu

Course page:

https://huazhengwang.github.io/IR2024Winter.html

Tuesday & Thursday 4:00-5:50pm at Weniger Hall 287

Pre-requisites

- Python. We will use Python/Jupyter notebook for programming assignments.
- Basic knowledge of probability and statistics.
- Basic understanding of machine learning / deep learning.

Course Description

- Foundations of **modern search engine**, fundamental and advanced techniques of text-based information systems
- Covered topics: web crawling, indexing, query understanding, language models, learning to rank, interactive search, evaluation and user models, question answering, conversational system, and neural retrieval models.
- **In short, you will learn the past, present and future of search engine and build a search engine yourself!**

Gradings

- Homework – (3*15%) 45%
- Midterm exam – 30%
- Final project – 25%
- Total – 100%

RECAP

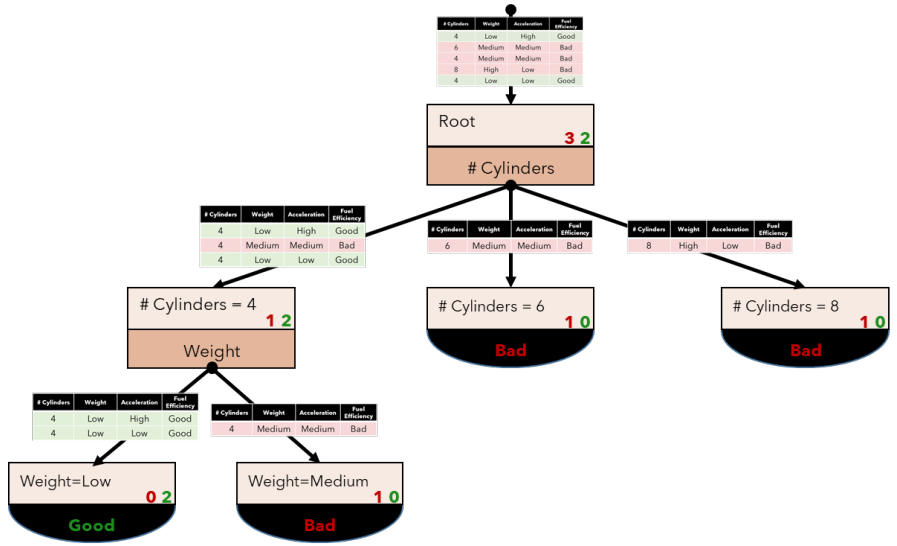
From Last Lecture

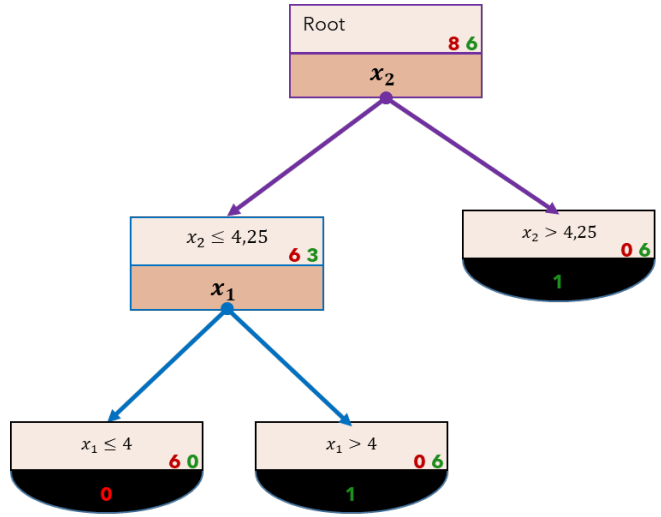
New Topic: Decision Trees

Decision Trees

A decision tree is a tree structured model where:

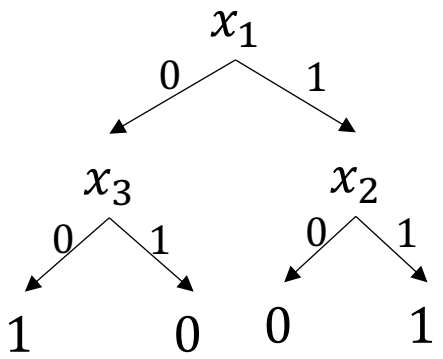
- **Internal nodes** perform tests against an individual input feature value
- Each branch from an internal node represents a potential value or range of values of the tested attribute.
- Each **leaf node** predicts an output
  - Either class or continuous value





As the number of nodes (or depth) of the tree increases, the model can represent more complex functions (aka the hypothesis space grows):

- **Depth 1** ("decision stump") can represent any Boolean function of one feature.
- **Depth 2** Any Boolean function of two features and some Boolean functions of three features (e.g., *(x_1 and x_2) or (not x_1 and not x_3)*)
- Etc..



Greedy Decision Tree Algorithm

Back to our recursive learning algorithm:

BasicGreedyAlgorithm(dataset S):

If S all have same label or all same features:

return Leaf(majorityLabel(S))

Based on S, choose "best" test t to split the data:

Evaluate the information gain of possible splits and pick the largest

Split S into subset $S_1, \ldots, S_k$ for each unique outcome of t applied to S

For i in {1,…,k}:

Create child node c_i = BasicGreedyAlgorithm(S_i)

Entropy of a Random Variable Y: More uncertainty, more entropy!

Discrete Distributions:

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

Continuous Distributions:

$$H(Y) = -\int_{y \in Y} P(Y = y) \log_2 P(Y = y)\, dy$$

Conditional Entropy of a Random Variable Y Given X:

Entropy of Y when only considering records were $X = x_j$

$$H(Y|X) = -\sum_{j}^{v} P(X = x_j) \sum_{i=1}^{k} P\left(Y = y_i | X = x_j\right) \log_2 P(Y = y_i | X = x_j)$$

Weighted by probability of $X = x_j$

Continuous distribution version replaces sums with integrals as is typical.

How to choose tests / splits / attributes?

Information Gain: $\mathrm{IG}(X) = H(Y) - H(Y|X)$

Entropy of class label before splitting on an attribute minus the conditional entropy after the split.

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
6	Medium	Medium	Bad
4	Medium	Medium	Bad
8	High	Low	Bad
4	Low	Low	Good

$H(Y) \approx 0.971$

Split on # Cylinders

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Medium	Medium	Bad
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad

# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$H(Y|Cylinders) \approx 0.551$

Split on Weight

# Cylinders	Weight	Acceleration	Fuel Efficiency
4	Low	High	Good
4	Low	Low	Good

# Cylinders	Weight	Acceleration	Fuel Efficiency
6	Medium	Medium	Bad
4	Medium	Medium	Bad

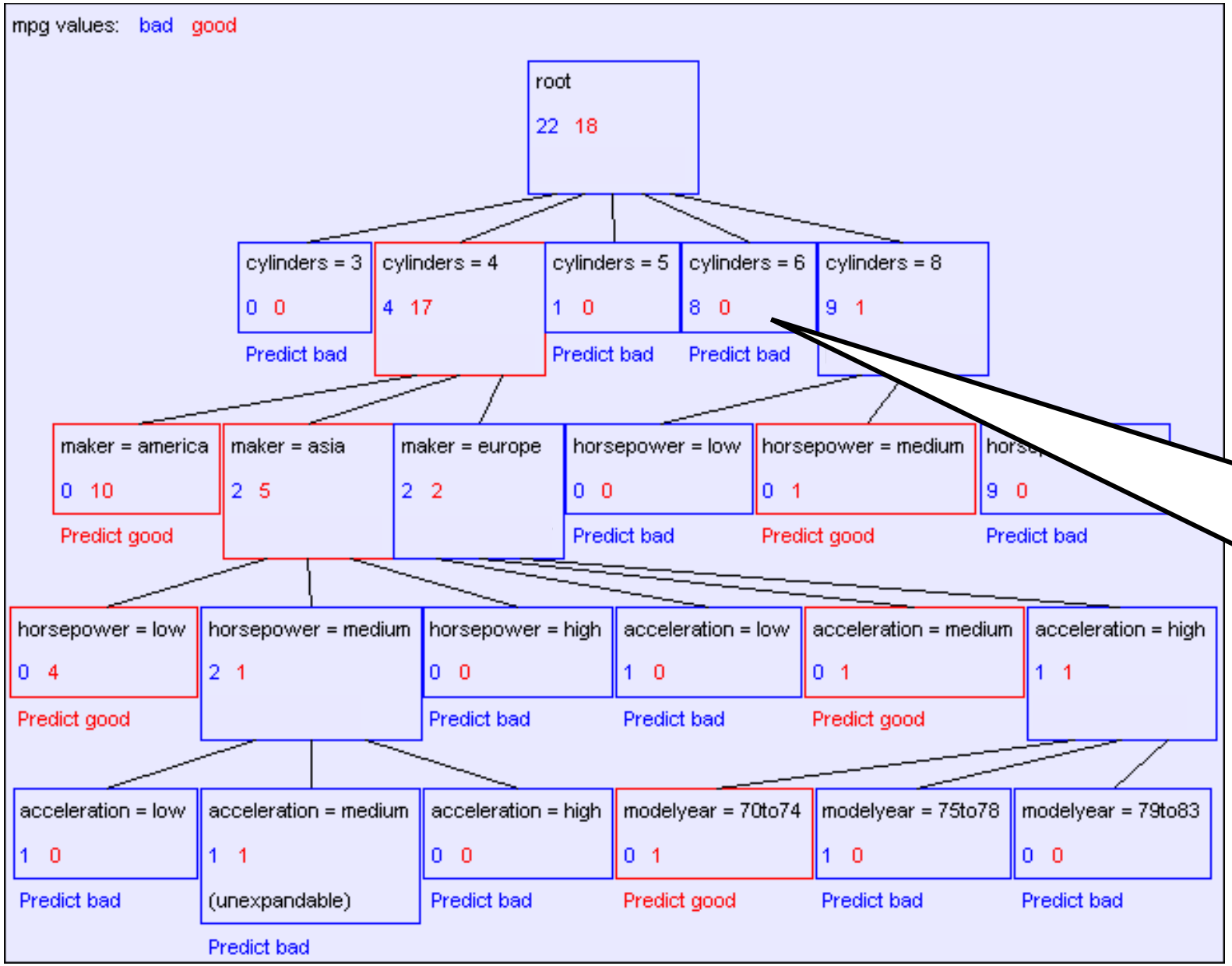
# Cylinders	Weight	Acceleration	Fuel Efficiency
8	High	Low	Bad

$H(Y|Weight) \approx 0$

$$\mathrm{IG}(\#\mathrm{Cylinders}) = H(Y) - H(Y|\#\mathrm{Cylinders}) = 0.971 - 0.551 = 0.42$$
$$\mathrm{IG}(\mathrm{Weight}) = H(Y) - H(Y|\mathrm{Weight}) = 0.971 - 0 = 0.971$$

We reduce uncertainty *more* by choosing to split on the Weight attribute.

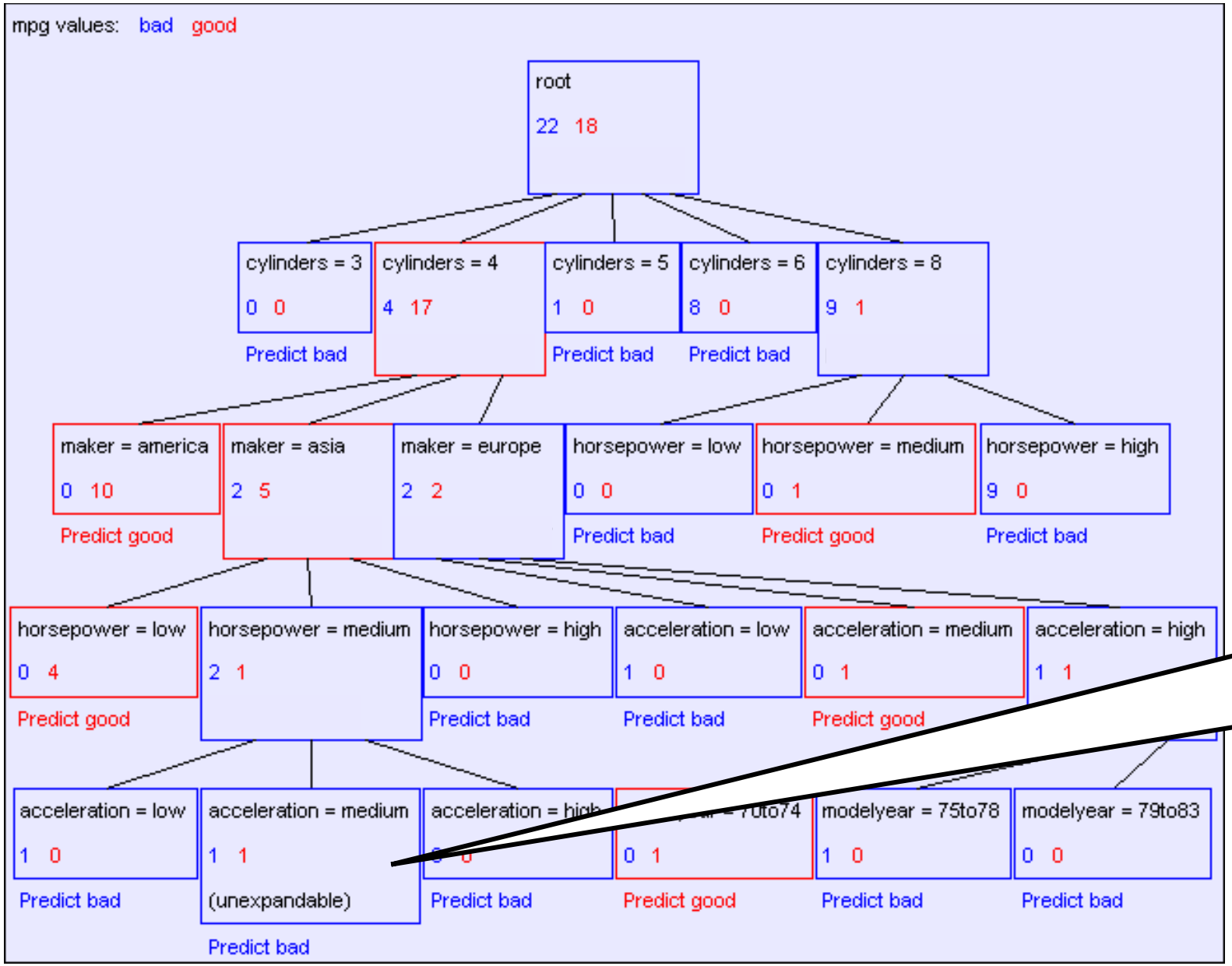
When do we stop? **Base Case 1**



Don't split a node if all matching records have the same output value

Example from Carlos Geustrin

When do we stop? **Base Case 2**



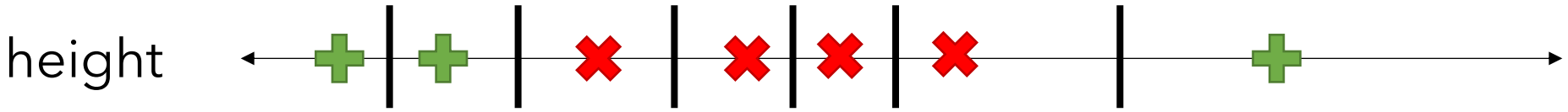
Don't split if none of the attributes can divide the examples

Example from Carlos Geustrin

Dealing with Continuous Features

Want to make splits of the form $X_i \geq t$ for some threshold t:

- Sort the values of X_i in the dataset, consider thresholds that occur in between consecutive datapoints.
- Compute information gain for each and choose the max.



Considering both discrete and continuous features

Suppose we have both continuous and discrete features:

- **Continuous:** Cylinder Volume, Vehicle Weight
- **Discrete:** Manufacturer, Diesel/Regular

Don't really need to do anything special for this case.

- At each step consider all possible splits including:
  - Splits on discrete attributes
  - Threshold splits in-between datapoints for continuous attributes

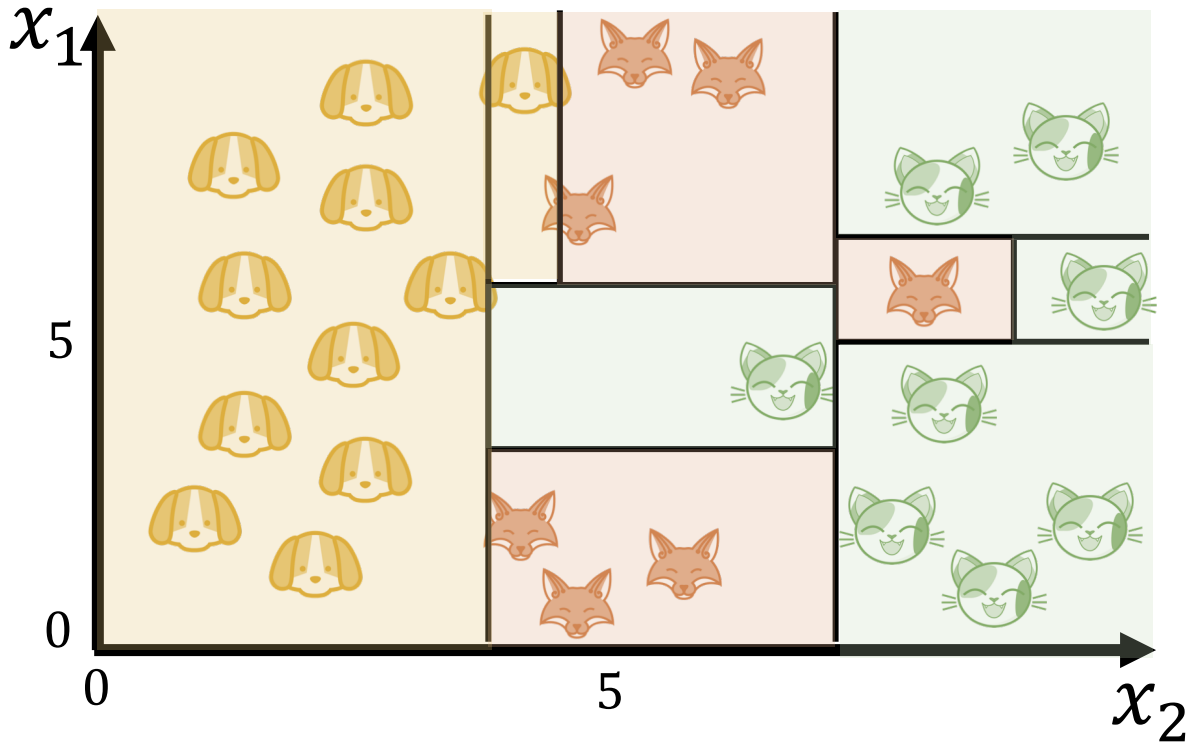
Today's Learning Objectives

Be able to answer:

- Wrap up Decision Trees.
- What is the bias-variance tradeoff?
- What is an ensemble?
  - How do ensembles reduce error?
- What is bagging?
  - What are random forests?
- What is boosting?

Problem of Overfitting

As decision trees get bigger (deeper), they can represent very complex functions.



Really large trees can achieve minimal error on training sets, but may not generalize to test data – overfitting.

Problem of Overfitting

Really large trees can achieve minimal error on **training sets**, but may not generalize to **test data** – overfitting.

Avoiding Overfitting

Option 1: Add more hyperpameters to control tree size:

- Limit depth / limit number of nodes / only split if at least K datapoints present

Option 2: Early stopping based on validation performance

- Monitor the validation accuracy and stop splitting a branch when performance saturates.
- Can be tricky for the same reasons that our proposed Base Case 3 can be.

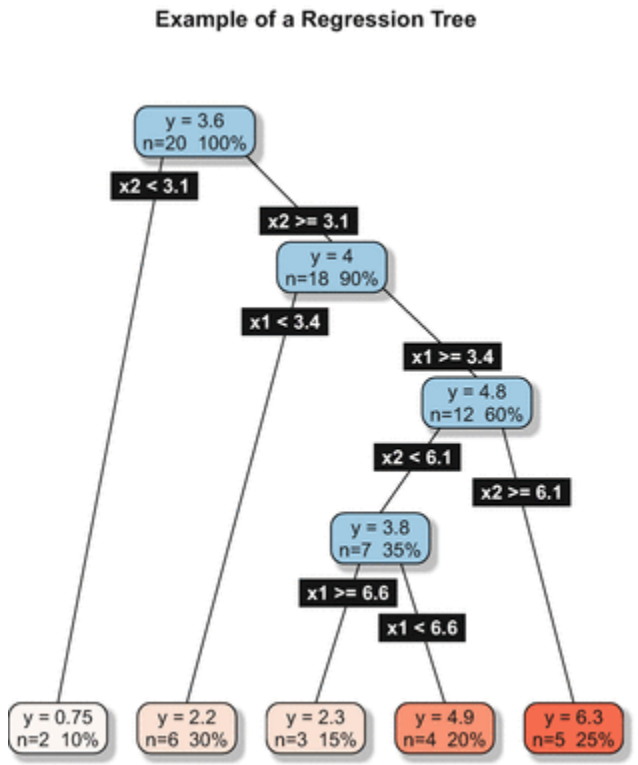
Option 3: Post Pruning

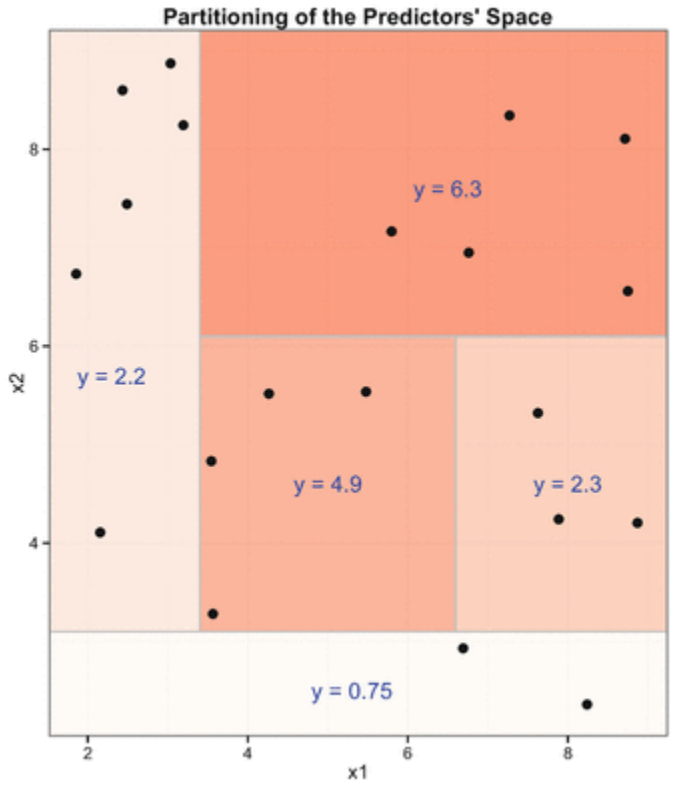
- Grow full tree on the training set, then consider the impact of removing each node on validation performance.
- Greedily prune the node that most improves validation set performance.

Miscellaneous: Regression Trees

We've talked mostly about classification trees. Can also do regression:

- Leaf nodes store the average output value of datapoints in them rather than the majority class.
- Rather the conditional entropy, consider something like change in squared error when choosing splits.





Miscellaneous: Fitting models at leaves

So far, we've proposed two things to do at the leaves of the tree:

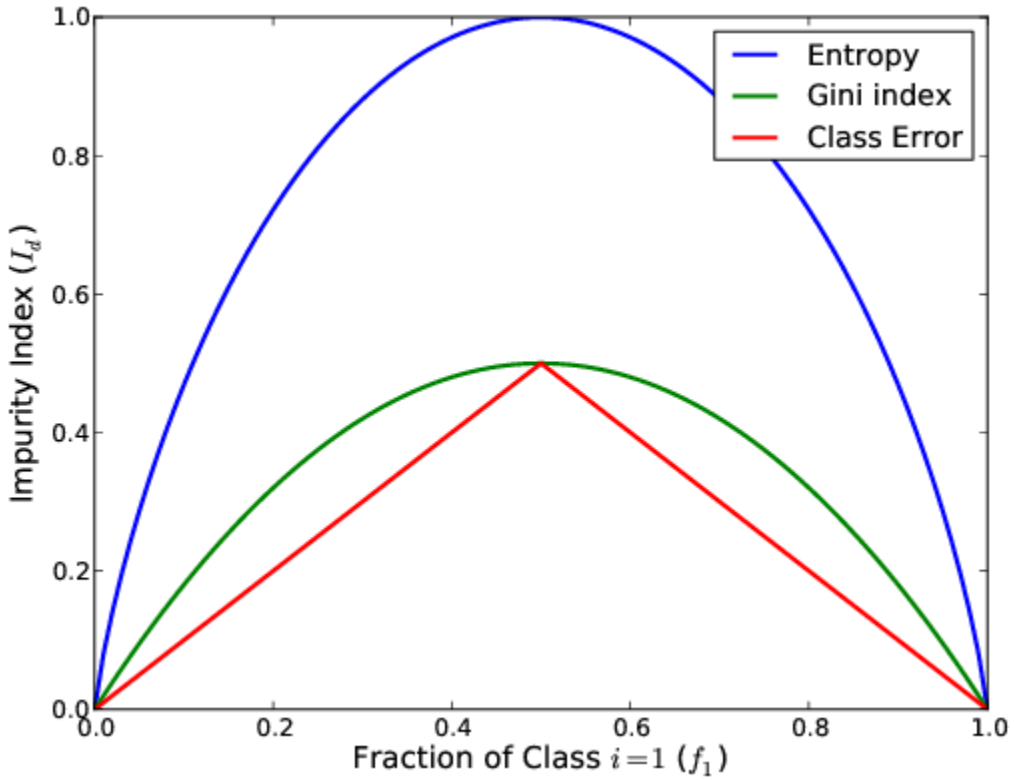
- Store the label of the majority of the datapoints in the leaf (classification)
- Store the average of the datapoints in the leaf (regression)

Could do something fancier:

- For each leaf, fit a model on the data points in that leaf. For example:
  - Logistic regression for classification
  - Linear regression for regression

Miscellaneous: Alternatives to Information Gain

We used Information Gain to determine splits. Plenty of alternatives.



Summary of Decision Trees

Decision trees are very flexible classifiers

- Can model arbitrarily complex decision boundaries
- Complexity of model can be controlled by tree depth
- Handles both continuous and discrete features
- Can be used for both classification and regression

Learning decision trees

- Greedy top-down selection of splits
- Not guaranteed to find an optimal tree (smallest with minimal error)

Decision trees can easily overfit

- Can avoid this with early stopping or post pruning

In Class Exercise: Learning a Decision Tree

Married	Education	Credit Score	Approve Loan
No	Highschool	680	No
Yes	Highschool	720	Yes
Yes	College	720	Yes
No	Grad School	800	Yes
No	Grad School	540	No
Yes	Grad School	680	No
No	College	680	Yes

$$\mathrm{IG}(X) = H(Y) - H(Y|X)$$

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X) = -\sum_{j}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i|X = x_j) \log_2 P(Y = y_i|X = x_j)$$

1) List Possible Splits:

- Married, Education, Credit Score > t (t = 610, 700, 760)

2) Compute Information Gain for Each Possible Split:

- **Married**: (4 3) → No (2 2) Yes (2 1)

$$H(Y|X) = -\frac{4}{7}\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) - \frac{3}{7}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) =$$

- **Education**: (4 3) → Highschool (1 1) College (2 0) Grad School (1 2)

$$H(Y|X) = -\frac{2}{7}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) - \frac{2}{7}\left(\frac{2}{2}\log_2\frac{2}{2}\right) - \frac{3}{7}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) =$$

- **Credit Score > 610**: (4 3) → ≤610 (0 1) >610 (4 2)

$$H(Y|X) = -\frac{1}{7}\left(\frac{1}{1}\log_2\frac{1}{1}\right) - \frac{6}{7}\left(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}\right) =$$

- **Credit Score > 700**: (4 3) → ≤700 (1 3) >700 (3 0)

$$H(Y|X) = -\frac{4}{7}\left(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right) - \frac{3}{7}\left(\frac{3}{3}\log_2\frac{3}{3}\right) =$$

- **Credit Score > 760**: (4 3) → ≤760 (3 3) >760 (1 0)

$$H(Y|X) = -\frac{6}{7}\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) - \frac{1}{7}\left(\frac{1}{1}\log_2\frac{1}{1}\right) =$$

In Class Exercise: Learning a Decision Tree

Married	Education	Credit Score	Approve Loan
No	Highschool	680	No
Yes	Highschool	720	Yes
Yes	College	720	Yes
No	Grad School	800	Yes
No	Grad School	540	No
Yes	Grad School	680	No
No	College	680	Yes

$$\mathrm{IG}(X) = H(Y) - H(Y|X)$$

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X) = -\sum_{j}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i|X = x_j) \log_2 P(Y = y_i|X = x_j)$$

1) List Possible Splits:

- Married, Education, Credit Score > t (t = 610, 700, 760)

2) Compute Information Gain for Each Possible Split:

- **Married**: (4 3) → No (2 2) Yes (2 1)

$$H(Y|X) = -\frac{4}{7}\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4}\right) - \frac{3}{7}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) \approx 0.964$$

- **Education**: (4 3) → Highschool (1 1) College (2 0) Grad School (1 2)

$$H(Y|X) = -\frac{2}{7}\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) - \frac{2}{7}\left(\frac{2}{2}\log_2\frac{2}{2}\right) - \frac{3}{7}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) \approx 0.512$$

- **Credit Score > 610**: (4 3) → ≤610 (0 1) >610 (4 2)

$$H(Y|X) = -\frac{1}{7}\left(\frac{1}{1}\log_2\frac{1}{1}\right) - \frac{6}{7}\left(\frac{4}{6}\log_2\frac{4}{6} + \frac{2}{6}\log_2\frac{2}{6}\right) \approx 0.787$$

- **Credit Score > 700**: (4 3) → ≤700 (1 3) >700 (3 0)

$$H(Y|X) = -\frac{4}{7}\left(\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right) - \frac{3}{7}\left(\frac{3}{3}\log_2\frac{3}{3}\right) \approx 0.463$$

- **Credit Score > 760**: (4 3) → ≤760 (3 3) >760 (1 0)

$$H(Y|X) = -\frac{6}{7}\left(\frac{3}{6}\log_2\frac{3}{6} + \frac{3}{6}\log_2\frac{3}{6}\right) - \frac{1}{7}\left(\frac{1}{1}\log_2\frac{1}{1}\right) \approx 0.857$$

In Class Exercise: Learning a Decision Tree

Married	Education	Credit Score	Approve Loan
No	Highschool	680	No
No	Grad School	540	No
Yes	Grad School	680	No
No	College	680	Yes

Married	Education	Credit Score	Approve Loan
Yes	Highschool	720	Yes
Yes	College	720	Yes
No	Grad School	800	Yes

$$\mathrm{IG}(X) = H(Y) - H(Y|X)$$

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X) = -\sum_{j}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i|X = x_j) \log_2 P(Y = y_i|X = x_j)$$

1) List Possible Splits:

- Married, Education, Credit Score > t (t = 610, ~~700~~, ~~760~~)

2) Compute Information Gain for Each Possible Split:

- **Married**: (1 3) → No (1 2) Yes (0 1)

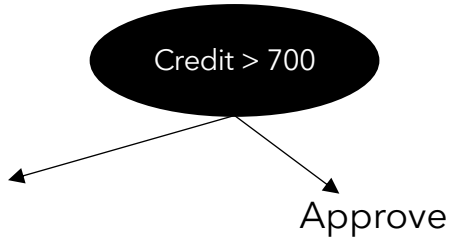
$$H(Y|X) = -\frac{3}{4}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) - \frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right)$$

- **Education**: (1 3) → Highschool (0 1) College (1 0) Grad School (0 2)

$$H(Y|X) = -\frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right) - \frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right) - \frac{2}{4}\left(\frac{2}{2}\log_2\frac{2}{2}\right)$$

- **Credit Score > 610**: (1 3) → ≤610 (0 1) >610 (1 2)

$$H(Y|X) = -\frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right) - \frac{3}{4}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right)$$



In Class Exercise: Learning a Decision Tree

Married	Education	Credit Score	Approve Loan
No	Highschool	680	No
No	Grad School	540	No
Yes	Grad School	680	No
No	College	680	Yes

Married	Education	Credit Score	Approve Loan
Yes	Highschool	720	Yes
Yes	College	720	Yes
No	Grad School	800	Yes

$$\mathrm{IG}(X) = H(Y) - H(Y|X)$$

$$H(Y) = -\sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

$$H(Y|X) = -\sum_{j}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i|X = x_j) \log_2 P(Y = y_i|X = x_j)$$

1) List Possible Splits:

- Married, Education, Credit Score > t (t = 610, ~~700~~, ~~760~~)

2) Compute Information Gain for Each Possible Split:

- **Married**: (1 3) → No (1 2) Yes (0 1)

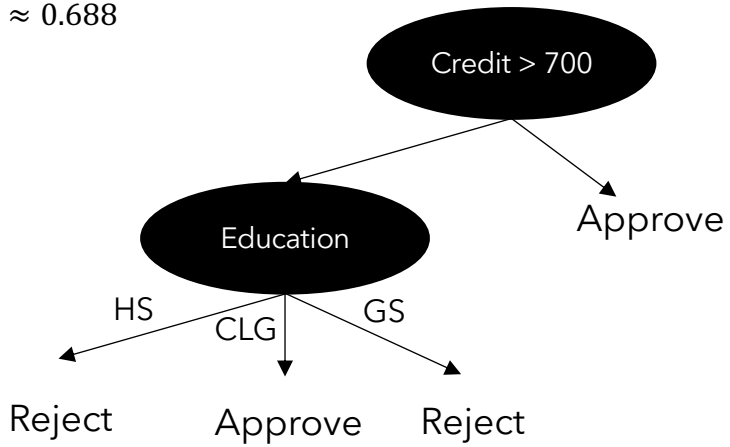
$$H(Y|X) = -\frac{3}{4}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) - \frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right) \approx 0.688$$

- **Education**: (1 3) → Highschool (0 1) College (1 0) Grad School (0 2)

$$H(Y|X) = -\frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right) - \frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right) - \frac{2}{4}\left(\frac{2}{2}\log_2\frac{2}{2}\right) \approx 0$$

- **Credit Score > 610**: (1 3) → ≤610 (0 1) >610 (1 2)

$$H(Y|X) = -\frac{1}{4}\left(\frac{1}{1}\log_2\frac{1}{1}\right) - \frac{3}{4}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{2}{3}\log_2\frac{2}{3}\right) \approx 0.688$$



Today's Learning Objectives

Be able to answer:

- ~~Wrap up Decision Trees.~~

- What is the bias-variance tradeoff?

- What is an ensemble?
  - How do ensembles reduce error?

- What is bagging?
  - What are random forests?

- What is boosting?

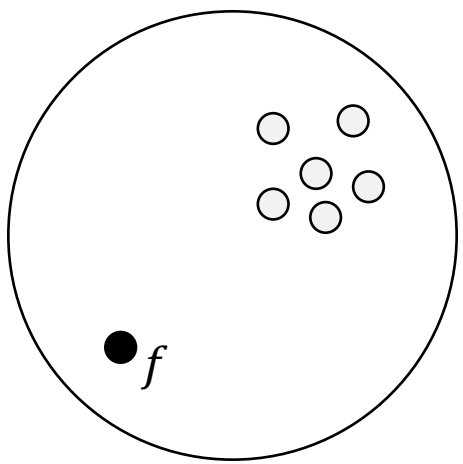
Bias-Variance Tradeoff

Bias: Error due to assumptions in the model not matching the problem (aka modelling error)

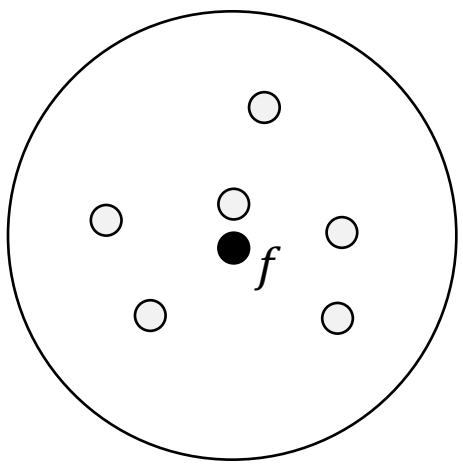
- More formally, average error between model and the true function over all possible datasets.

Variance: Error due to sensitivity to changes in the dataset (aka estimation + optimization)

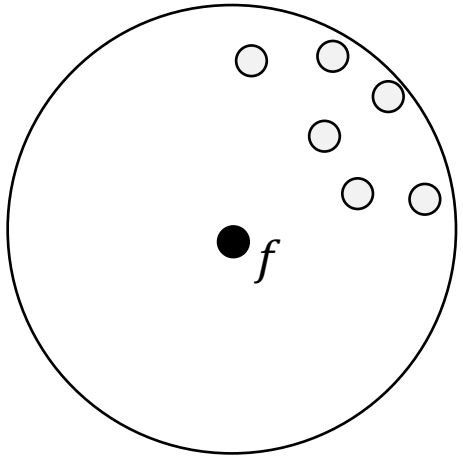
- More formally, variance of error between model and the true function over all possible datasets.



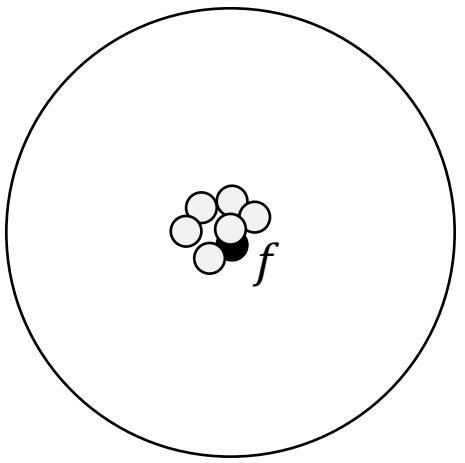
High Bias
Low Variance



Low Bias
High Variance

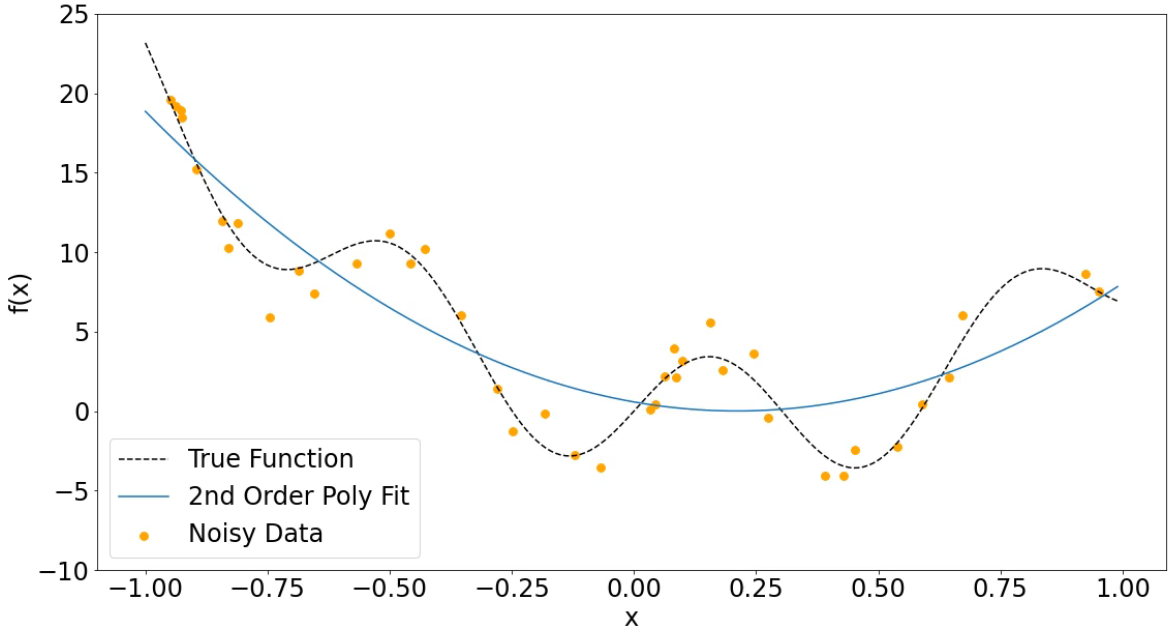


High Bias
High Variance



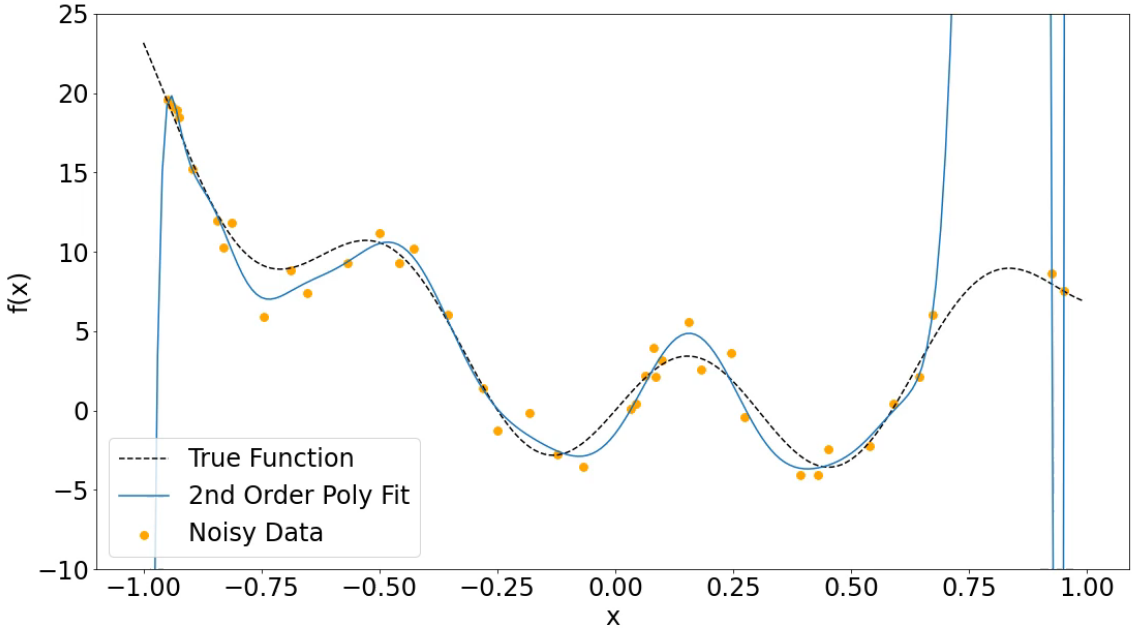
Low Bias
Low Variance

Bias-Variance Tradeoff



2nd Order Polynomial Fit:
Low variance / high bias

20th Order Polynomial Fit:
High variance / low bias



Bias-Variance Decomposition of Error

Assume we are doing regression and trying to learn a model $\hat{f}(\mathrm{x})$ that minimize squared error on some dataset D. Further, assume our exampled in the dataset are from some true function $f(x)$ but have been corrupted by Gaussian noise such that $y_i = f(x_i) + \epsilon_i$ where $\epsilon_i \sim N(0, \sigma^2)$. What is our expected error?

$$E_D\left[\left(f(x) + \epsilon - \hat{f}(x;D)\right)^2\right] = E_D\left[f(x) + \epsilon - \hat{f}(x;D)\right]^2 + Var_D\left[f(x) + \epsilon - \hat{f}(x;D)\right]$$

$$= \left(E_D[f(x)] + E_D[\epsilon] - E_D\left[\hat{f}(x;D)\right]\right)^2 + Var_D\left[f(x) + \epsilon - \hat{f}(x;D)\right]$$

$$= \left(f(x) - E_D\left[\hat{f}(x;D)\right]\right)^2 + Var_D\left[f(x) + \epsilon - \hat{f}(x;D)\right]$$

$$= \left(f(x) - E_D\left[-\hat{f}(x;D)\right]\right)^2 + Var_D\left[f(x) - \hat{f}(x;D)\right] + Var_D[\epsilon]$$

$$= \left(f(x) - E_D\left[-\hat{f}(x;D)\right]\right)^2 + Var_D\left[f(x) - \hat{f}(x;D)\right] + \sigma^2$$

$$= Bias^2 + Variance + Bayes\ Error$$

$Note: E[X^2] = E[X]^2 + Var[X]$

What We've Learned in Context of the Bias-Variance Tradeoff

Low Variance / High Bias Learners: (aka "weak" learners)

- Naïve Bayes, Logistic Regression, Decision Stumps (or shallow decision trees)
  - Good: Low variance – don't usually overfit
  - Bad: High bias – can't represent complex functions

High Variance / Low Bias Learners: (aka "strong" learners)

- Kernel methods, Neural Networks, Large Decision Trees, kNN
  - Good: Low bias – have the potential to learn complex functions
  - Bad: High variance – easy to overfit to training dataset

Can we get the best of both?

- In general, no… no free lunch.
- But often yes!

Today's Learning Objectives

Be able to answer:

- ~~Wrap up Decision Trees.~~
- ~~What is the bias-variance tradeoff?~~
- What is an ensemble?
  - How do ensembles reduce error?
- What is bagging?
  - What are random forests?
- What is boosting?

Core Intuition of Ensemble Methods: A combination of multiple classifiers may perform better than a single classifier.

Instead of learning a single model, learn many models. Final output of the "ensemble" of models is a combination of each model's output.

Ensemble methods work well when each individual member is:

- Accurate (Better than chance)
- Diverse (Uncorrelated errors on new examples) ← **Why?**

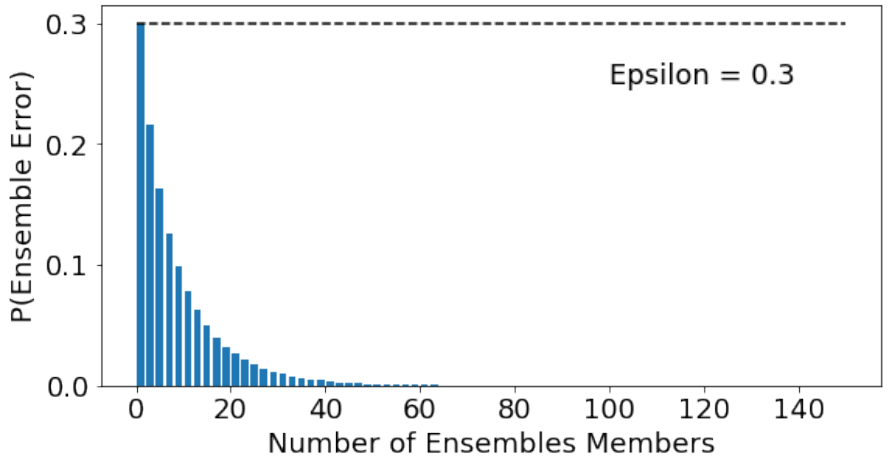
Consider an ensemble of M models that takes the majority vote of its members as the final output. Assume each as an uncorrelated error rate ϵ

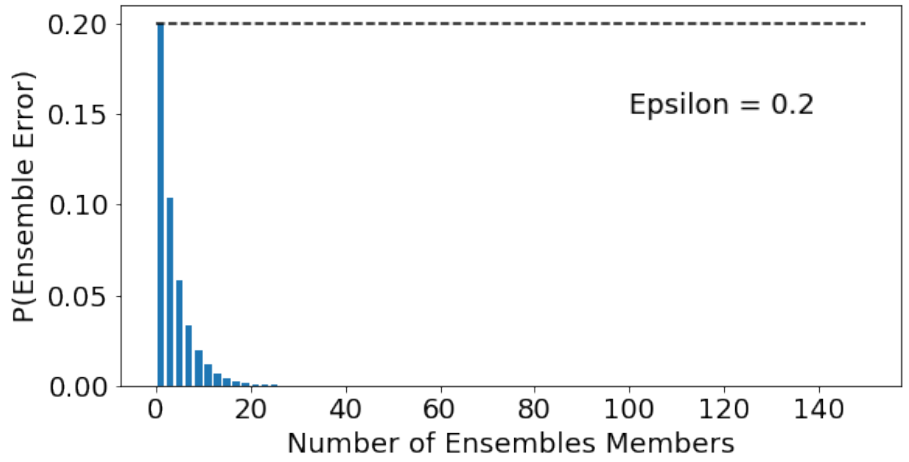
Probability that a majority (> M/2) models simultaneously make an error:

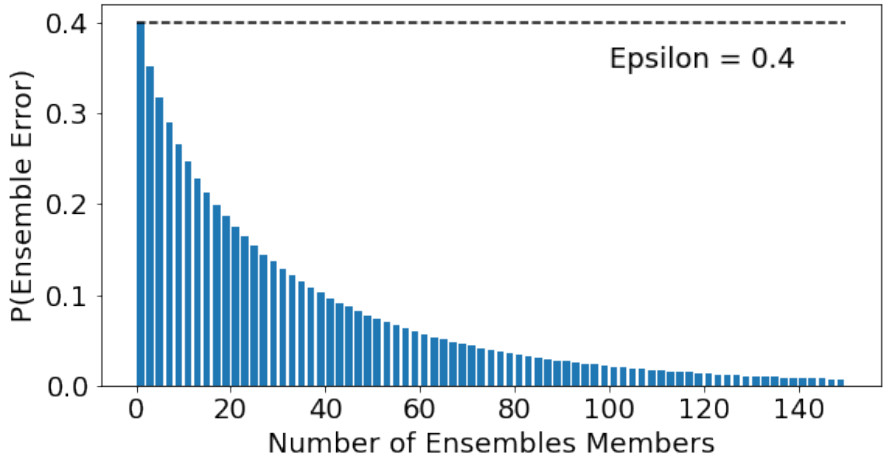
$$P\left(\#errors \geq \frac{M}{2}\right) = \sum_{h=M/2}^{M} \binom{M}{h} \epsilon^h (1-\epsilon)^{M-h}$$

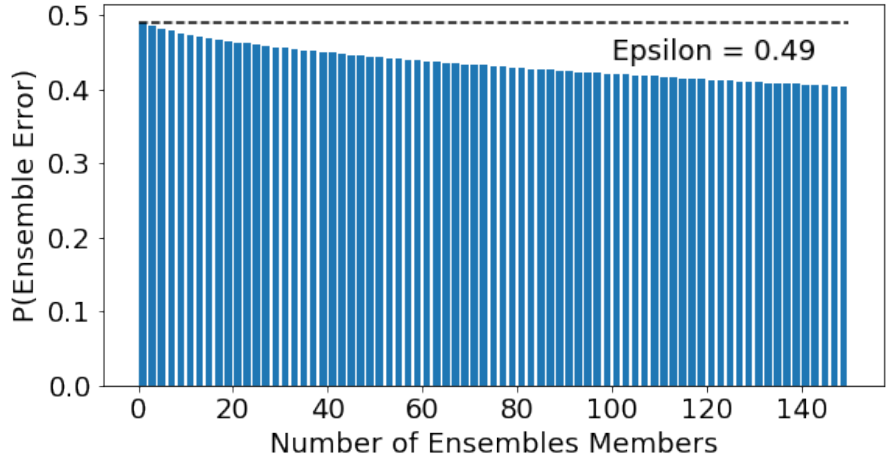
Assuming uncorrelated error rates of ϵ,

$$P\left(\#errors \geq \frac{M}{2}\right) = \sum_{h=M/2}^{M} \binom{M}{h} \epsilon^h (1-\epsilon)^{M-h}$$

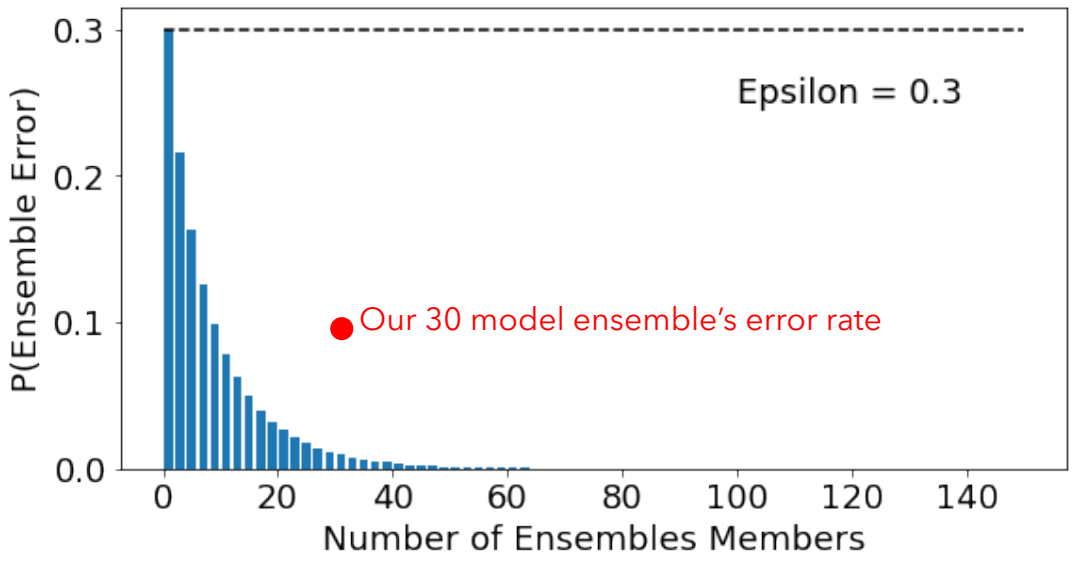








Question Break!



Suppose you have an ensemble of 30 models, each with a validation error of 30%. We observe our ensemble has an error rate of 10% -- way higher than our derivations suggest.

Why is it higher than predicted?

A Our model's errors are uncorrelated

B Our model's errors are correlated

C There must be a bug.

D You sit on a throne of math lies.

Ensembles

If ensembles reduce error so well, why doesn't everyone use them? **They do.**

Almost all challenges are won with ensemble methods.

Netflix Prize Challenge -- $1,000,000 for a 10% gain over Netflix's model for predicting user ratings.

"

Table 5: Best results of single approaches and their combinations

Method/Combination	RMSE
MF	0.9190
NB	0.9313
CL	0.9606
NB + CL	0.9275
MF + CL	0.9137
MF + NB	0.9089
MF + NB + CL	0.9089

-- Gravity (3)

"

Leaderboard

Display top 40 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	--	--	--
Grand Prize - RMSE <= 0.8563				
--	No Progress Prize candidates yet	--	--	--
Progress Prize - RMSE <= 0.8625				
1	When Gravity and Dinosaurs Unite	0.8686	8.70	2008-02-12 12:03:24
2	BellKor	0.8693	8.63	2008-02-10 02:42:07
3	Gravity	0.8708	8.47	2008-02-06 14:12:44
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell				
4	KorBell	0.8712	8.43	2007-10-01 23:25:23
5	Dan Tillberg	0.8727	8.27	2008-02-18 03:48:03
6	basho	0.8729	8.25	2007-11-24 14:27:00
7	Just a guy in a garage	0.8740	8.14	2008-02-06 12:16:40
8	Dinosaur Planet	0.8753	8.00	2007-10-04 04:56:45
9	BigChaos	0.8759	7.94	2008-02-15 23:24:47
10	Reel Ingenuity	0.8774	7.78	2008-02-14 19:28:30
11	acmehill	0.8777	7.75	2008-02-16 16:33:18
12	Three Blind Mice	0.8778	7.74	2008-02-16 20:47:39
13	ML@UToronto A	0.8787	7.64	2007-09-30 20:41:54
14	Arek Paterek	0.8789	7.62	2007-09-30 11:35:42
15	HowLowCanHeGo2	0.8794	7.57	2008-02-15 00:52:14
16	NIPS Reject	0.8808	7.42	2007-09-13 21:02:32
17	One Million Monkeys	0.8808	7.42	2008-02-15 15:21:47
18	Ces	0.8811	7.39	2008-02-14 07:26:49
19	ATTEAM	0.8822	7.27	2008-02-13 05:08:14
20	Efratko	0.8827	7.22	2008-02-13 21:22:49
21	Ensemble Experts	0.8841	7.07	2007-10-01 04:37:18
22	SecondaryResults	0.8842	7.06	2008-02-13 15:33:20
23	mathematical capital	0.8844	7.04	2008-02-06 13:59:43
24	Newman!	0.8848	7.00	2008-02-08 21:07:26
25	The Thought Gang	0.8849	6.99	2007-10-01 21:31:46
26	HowGoodCanHeBe	0.8856	6.92	2008-02-16 23:52:03
27	HAT	0.8857	6.91	2008-01-03 20:49:32
28	strudeltamale	0.8859	6.88	2007-09-25 16:50:45
29	NIPS Submission	0.8861	6.86	2007-06-08 23:27:03
30	Geoff Dean	0.8863	6.84	2007-11-18 09:05:30
31	fools	0.8866	6.81	2008-02-06 08:44:31

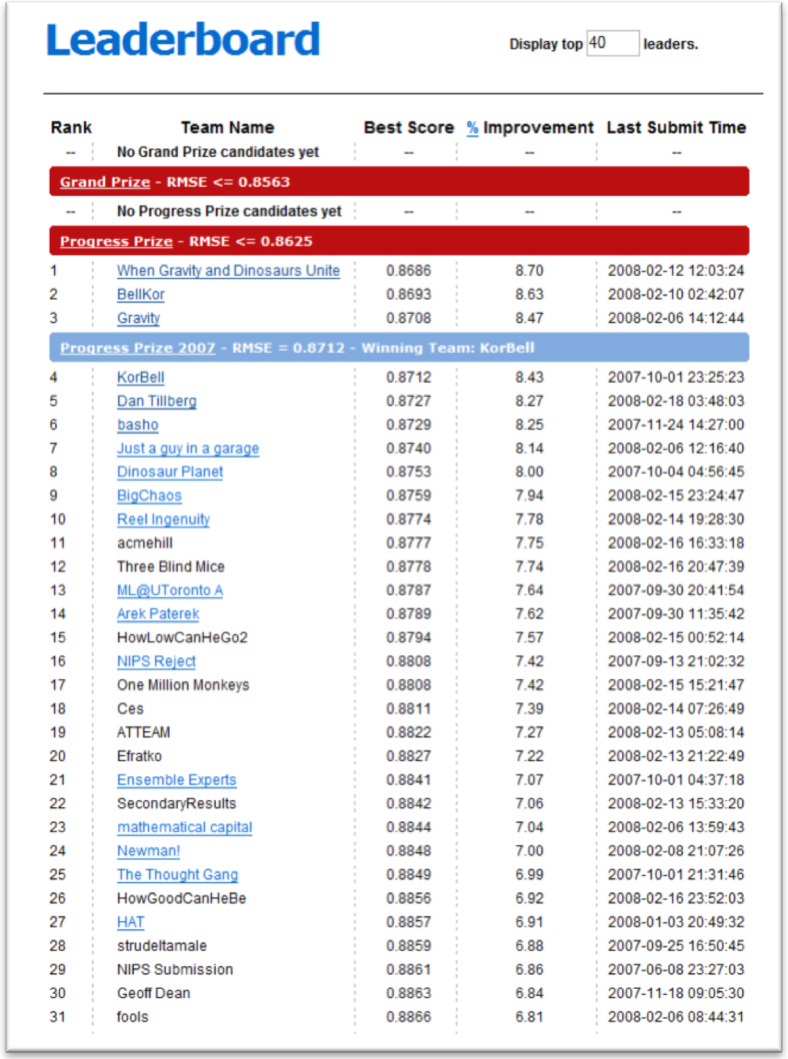
Ensembles

If ensembles reduce error so well, why doesn't everyone use them? **They do.**

Almost all challenges are won with ensemble methods.

Netflix Prize Challenge -- $1,000,000 for a 10% gain over Netflix's model for predicting user ratings.

> "Predictive accuracy is substantially improved when blending multiple predictors. Our experience is that most efforts should be concentrated in deriving substantially different approaches, rather than refining a single technique. […] Our final solution consists of blending 107 individual results. " **– BellKor (2)**



Leaderboard

Display top 40 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	--	--	--
Grand Prize - RMSE <= 0.8563				
--	No Progress Prize candidates yet	--	--	--
Progress Prize - RMSE <= 0.8625				
1	When Gravity and Dinosaurs Unite	0.8686	8.70	2008-02-12 12:03:24
2	BellKor	0.8693	8.63	2008-02-10 02:42:07
3	Gravity	0.8708	8.47	2008-02-06 14:12:44
Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell				
4	KorBell	0.8712	8.43	2007-10-01 23:25:23
5	Dan Tillberg	0.8727	8.27	2008-02-18 03:48:03
6	basho	0.8729	8.25	2007-11-24 14:27:00
7	Just a guy in a garage	0.8740	8.14	2008-02-06 12:16:40
8	Dinosaur Planet	0.8753	8.00	2007-10-04 04:56:45
9	BigChaos	0.8759	7.94	2008-02-15 23:24:47
10	Reel Ingenuity	0.8774	7.78	2008-02-14 19:28:30
11	acmehill	0.8777	7.75	2008-02-16 16:33:18
12	Three Blind Mice	0.8778	7.74	2008-02-16 20:47:39
13	ML@UToronto A	0.8787	7.64	2007-09-30 20:41:54
14	Arek Paterek	0.8789	7.62	2007-09-30 11:35:42
15	HowLowCanHeGo2	0.8794	7.57	2008-02-15 00:52:14
16	NIPS Reject	0.8808	7.42	2007-09-13 21:02:32
17	One Million Monkeys	0.8808	7.42	2008-02-15 15:21:47
18	Ces	0.8811	7.39	2008-02-14 07:26:49
19	ATTEAM	0.8822	7.27	2008-02-13 05:08:14
20	Efratko	0.8827	7.22	2008-02-13 21:22:49
21	Ensemble Experts	0.8841	7.07	2007-10-01 04:37:18
22	SecondaryResults	0.8842	7.06	2008-02-13 15:33:20
23	mathematical capital	0.8844	7.04	2008-02-06 13:59:43
24	Newman!	0.8848	7.00	2008-02-08 21:07:26
25	The Thought Gang	0.8849	6.99	2007-10-01 21:31:46
26	HowGoodCanHeBe	0.8856	6.92	2008-02-16 23:52:03
27	HAT	0.8857	6.91	2008-01-03 20:49:32
28	strudeltamale	0.8859	6.88	2007-09-25 16:50:45
29	NIPS Submission	0.8861	6.86	2007-06-08 23:27:03
30	Geoff Dean	0.8863	6.84	2007-11-18 09:05:30
31	fools	0.8866	6.81	2008-02-06 08:44:31

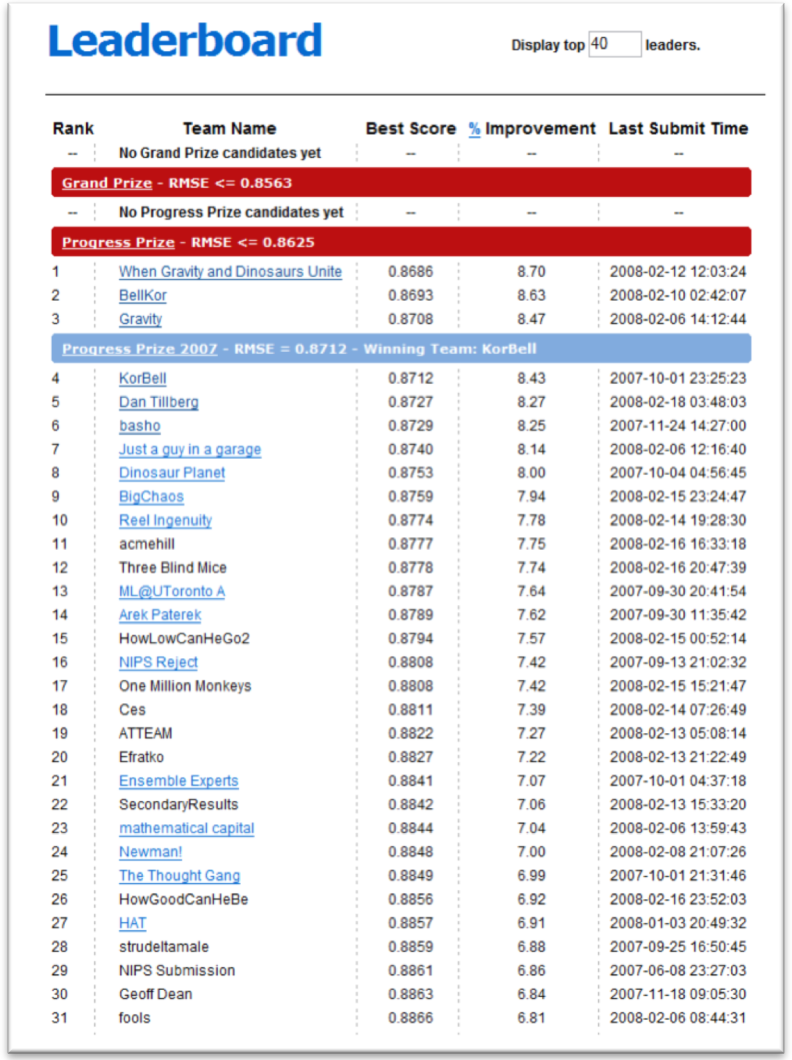
Ensembles

If ensembles reduce error so well, why doesn't everyone use them? **They do.**

Almost all challenges are won with ensemble methods.

Netflix Prize Challenge -- $1,000,000 for a 10% gain over Netflix's model for predicting user ratings.

> "Our common team blends the result of team Gravity and team Dinosaur Planet. -- **When Gravity and Dinosaurs Unite (1)**"



Leaderboard

Display top 40 leaders.

Rank	Team Name	Best Score	% Improvement	Last Submit Time
--	No Grand Prize candidates yet	--	--	--
	Grand Prize - RMSE <= 0.8563			
--	No Progress Prize candidates yet	--	--	--
	Progress Prize - RMSE <= 0.8625			
1	When Gravity and Dinosaurs Unite	0.8686	8.70	2008-02-12 12:03:24
2	BellKor	0.8693	8.63	2008-02-10 02:42:07
3	Gravity	0.8708	8.47	2008-02-06 14:12:44
	Progress Prize 2007 - RMSE = 0.8712 - Winning Team: KorBell			
4	KorBell	0.8712	8.43	2007-10-01 23:25:23
5	Dan Tillberg	0.8727	8.27	2008-02-18 03:48:03
6	basho	0.8729	8.25	2007-11-24 14:27:00
7	Just a guy in a garage	0.8740	8.14	2008-02-06 12:16:40
8	Dinosaur Planet	0.8753	8.00	2007-10-04 04:56:45
9	BigChaos	0.8759	7.94	2008-02-15 23:24:47
10	Reel Ingenuity	0.8774	7.78	2008-02-14 19:28:30
11	acmehill	0.8777	7.75	2008-02-16 16:33:18
12	Three Blind Mice	0.8778	7.74	2008-02-16 20:47:39
13	ML@UToronto A	0.8787	7.64	2007-09-30 20:41:54
14	Arek Paterek	0.8789	7.62	2007-09-30 11:35:42
15	HowLowCanHeGo2	0.8794	7.57	2008-02-15 00:52:14
16	NIPS Reject	0.8808	7.42	2007-09-13 21:02:32
17	One Million Monkeys	0.8808	7.42	2008-02-15 15:21:47
18	Ces	0.8811	7.39	2008-02-14 07:26:49
19	ATTEAM	0.8822	7.27	2008-02-13 05:08:14
20	Efratko	0.8827	7.22	2008-02-13 21:22:49
21	Ensemble Experts	0.8841	7.07	2007-10-01 04:37:18
22	SecondaryResults	0.8842	7.06	2008-02-13 15:33:20
23	mathematical capital	0.8844	7.04	2008-02-06 13:59:43
24	Newman!	0.8848	7.00	2008-02-08 21:07:26
25	The Thought Gang	0.8849	6.99	2007-10-01 21:31:46
26	HowGoodCanHeBe	0.8856	6.92	2008-02-16 23:52:03
27	HAT	0.8857	6.91	2008-01-03 20:49:32
28	strudeltamale	0.8859	6.88	2007-09-25 16:50:45
29	NIPS Submission	0.8861	6.86	2007-06-08 23:27:03
30	Geoff Dean	0.8863	6.84	2007-11-18 09:05:30
31	fools	0.8866	6.81	2008-02-06 08:44:31

Today's Learning Objectives

Be able to answer:

- ~~Wrap up Decision Trees.~~
- ~~What is the bias-variance tradeoff?~~
- ~~What is an ensemble?~~
  - ~~How do ensembles reduce error?~~
- What is bagging?
  - What are random forests?
- What is boosting?
  - How does L2 boosting work?
  - How does Adaboost work?

Ensemble Methods

Idea: Instead of learning a single model, learn many models. Final output of the "ensemble" of models is a weighted combination of each model's output.

Bagging

Boosting

Ensemble Methods

Idea: Instead of learning a single model, learn many models. Final output of the "ensemble" of models is a weighted combination of each model's output.

Bagging: Try to reduce variance in strong learners

- Uncorrelated errors → expected error goes down
- On average, do better than single classifier!

Boosting: Try to reduce bias in weak learners

- Each model good at different parts of the input space
- On average, do better than single classifier!

Adapted from Dhruv Batra. The cats are from me.

Ensemble Methods

Idea: Instead of learning a single model, learn many models. Final output of the "ensemble" of models is a weighted combination of each model's output.

Bagging: Try to reduce variance in strong learners

- Uncorrelated errors → expected error goes down
- On average, do better than single classifier!

Boosting: Try to reduce bias in weak learners

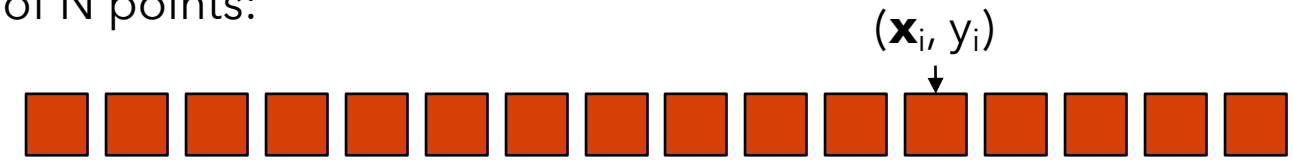
- Each model good at different parts of the input space
- On average, do better than single classifier!

Adapted from Dhruv Batra. The cats are from me.

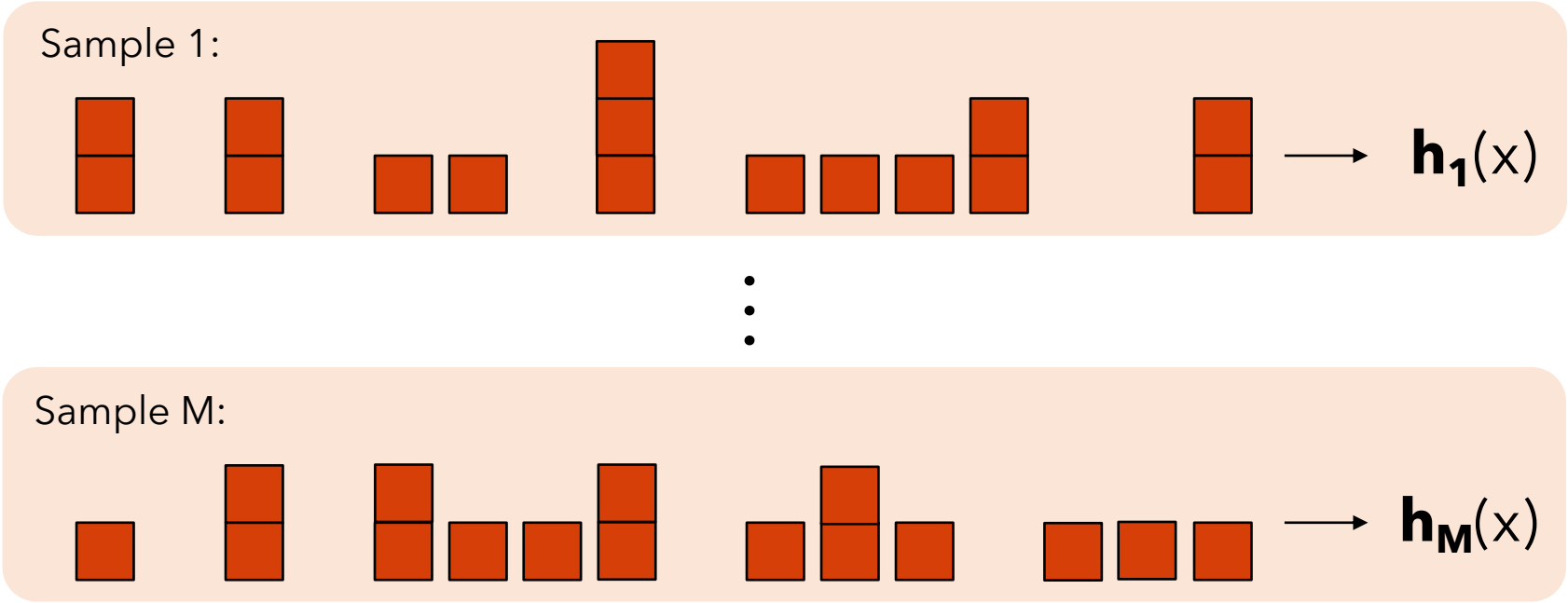
Bagging (**B**ootstrap **Agg**regat**ing**)

Bagging: Average multiple models trained from resamples of the data.

Given a dataset of N points:



Sample N training points **with replacement** and train a model, repeat M times:



Bagging (**B**ootstrap **Agg**regat**ing**)

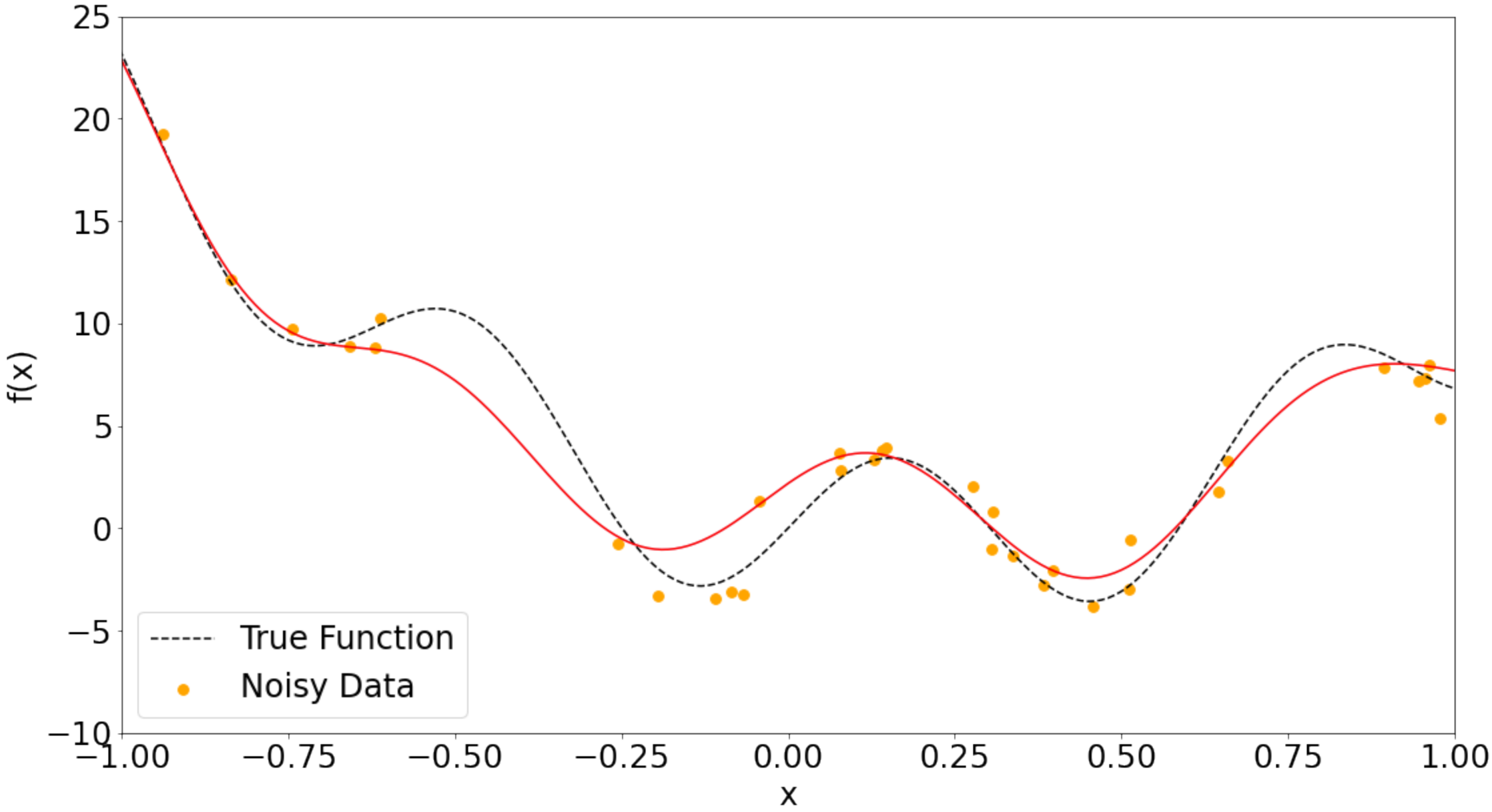
Bagging: Average multiple models trained from resamples of the data.

Regression: At test time, simply run each model and take the average of their outputs:

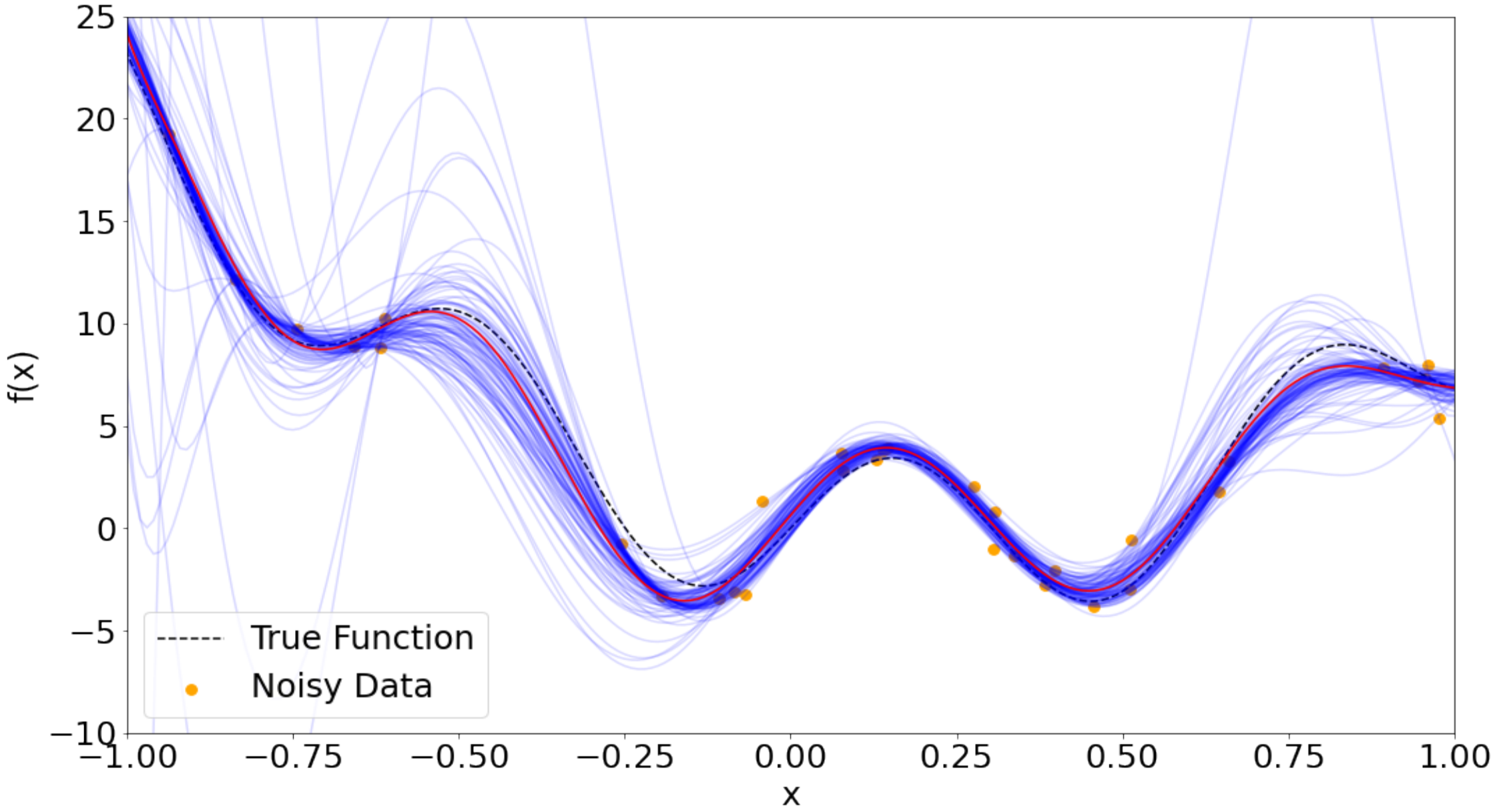
$$y_{pred} = \frac{1}{M} \sum_{m=1}^{M} h_m(x)$$

Classification: Can take the majority vote instead (if averaging classifier output isn't meaningful)

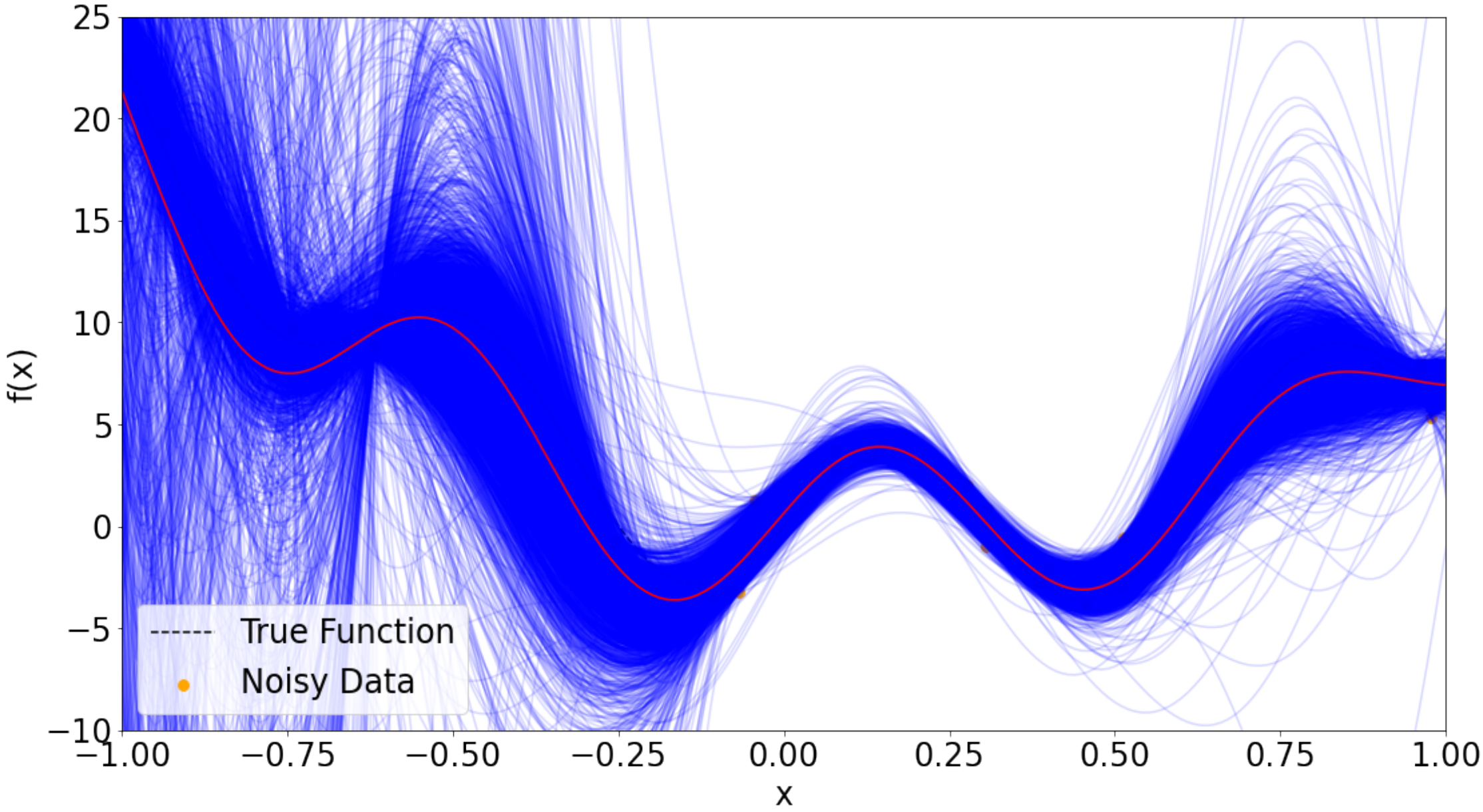
Bagging Example: Single Model



Bagging Example: 100 Models

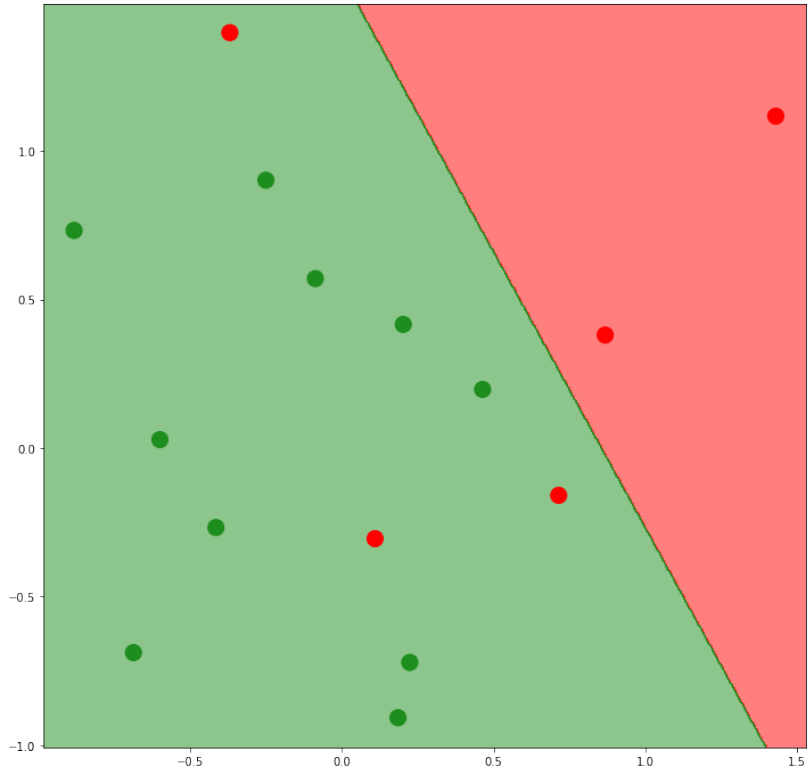


Bagging Example: 10,000 Models

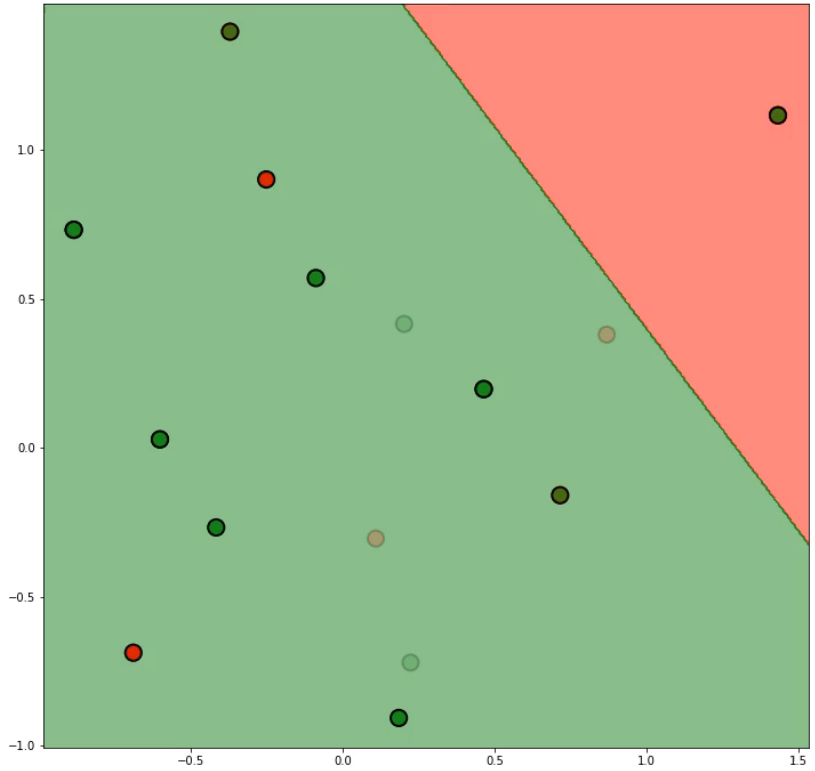


Bagging Example: Logistic Regression

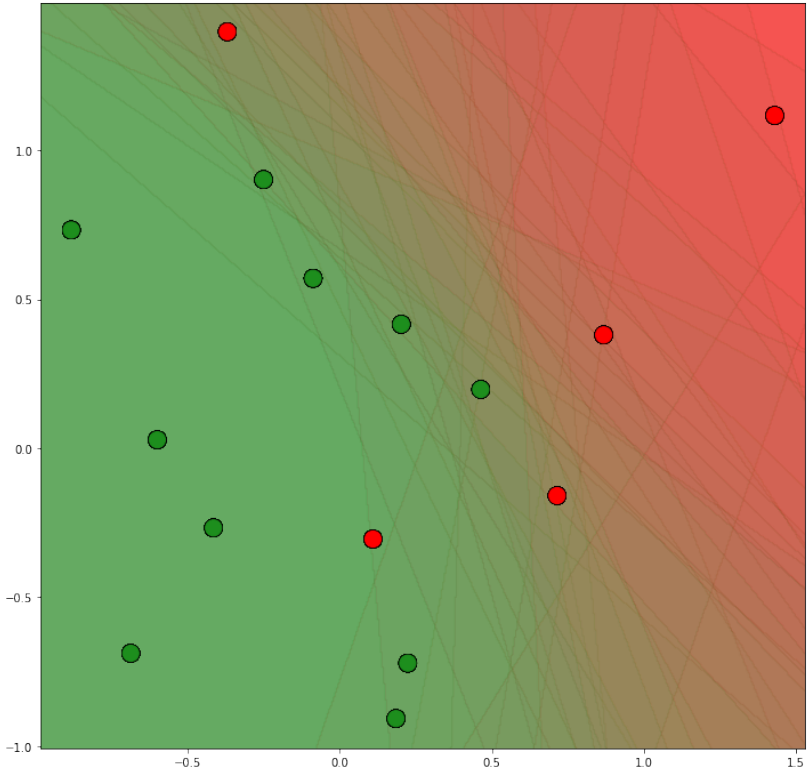
Single Model



Bootstrap Models



Ensemble



Bagging

Bagging is most useful for strong learners like neural networks and decision trees that have high variance but low bias.

- **Works best when errors in each ensemble member are not correlated.**
- Each ensemble member can be trained **in parallel**.

Use bagging when…

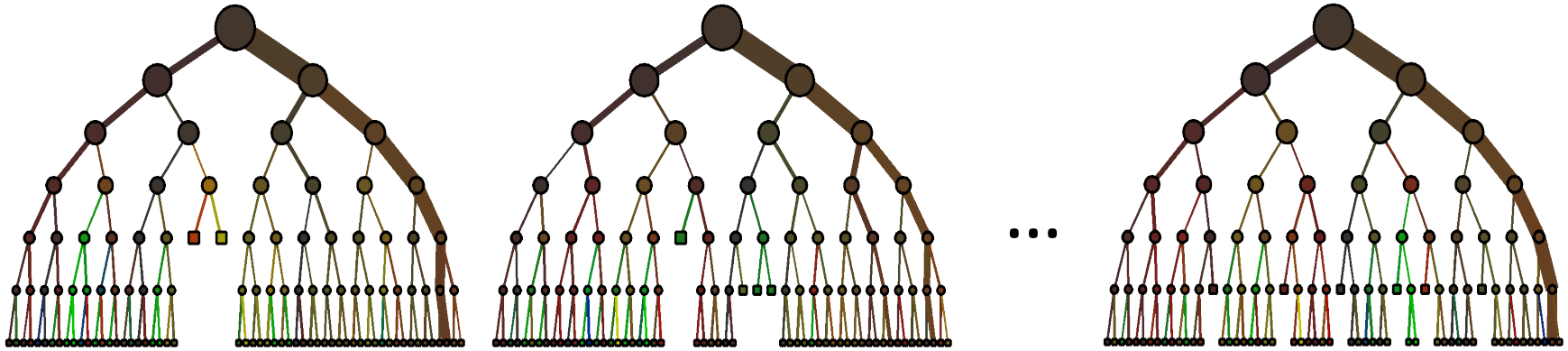
… you have overfit strong learners

… you have a somewhat reasonably sized dataset

… you want an extra bit of performance from your models

Let's consider applying bagging to decision trees.

Train a M trees on different resamplings of the data and call it a **forest**.

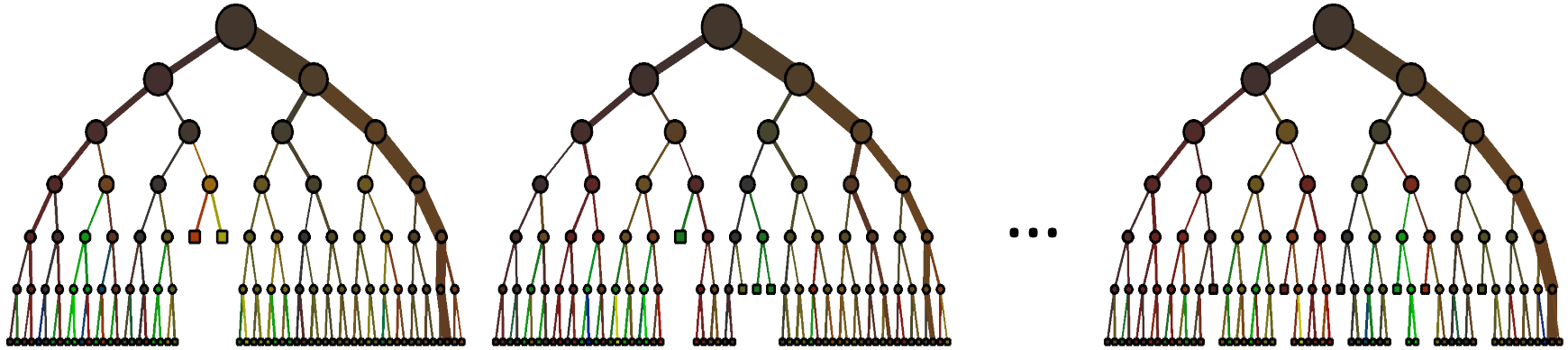


Make prediction by majority vote over the trees.

Issue: Our greedy decision tree learning algorithm will likely use the same attributes early on, despite a resampling. **Why is this an issue?**

Questions Break!

How might we add further diversity to our forest?



Random Forests:

- Train an ensemble of decision trees with bagging.

- During tree learning, at each split consider only a random subset of attributes / thresholds when choosing what to split on.
  - Often $\sqrt{d}$ features are considered at each split for d-dimensional inputs

- Output is majority vote from the forest.

Random forests are incredibly widely used in practical applications involving medium-sized dataset. An excellent "first attempt" for supervised problems with relatively low dimensionality.

Today's Learning Objectives

Be able to answer:

- ~~Wrap up Decision Trees.~~
- ~~What is the bias-variance tradeoff?~~
- ~~What is an ensemble?~~
  - ~~How do ensembles reduce error?~~
- ~~What is bagging?~~
  - ~~What are random forests?~~
- What is boosting?
  - How does L2 boosting work?
  - How does Adaboost work?

Ensemble Methods

Idea: Instead of learning a single model, learn many models. Final output of the "ensemble" of models is a weighted combination of each model's output.

Bagging: Try to reduce variance in strong learners

- Uncorrelated errors → expected error goes down
- On average, do better than single classifier!

Boosting: Try to reduce bias in weak learners

- Each model good at different parts of the input space
- On average, do better than single classifier!

Adapted from Dhruv Batra. The cats are from me.

Boosting

Idea: Train classifiers sequentially. After each classifier, focus more on points the previous models have large errors on.

- First train the base classifier on all the training data with equal importance weights on each datapoint.
- Then re-weight the training data to emphasize the hard datapoints and train a second model.
  - How do we re-weight the data?
- Repeat this and keep training new models on re-weighted data
- Finally, use a weighted ensemble of all the models for the test data.
  - How do we weight the models in the committee?

Example: L2 Boosting (Regression)

Consider a regression setting where we are minimizing the squared error.

We are going to learn a sequence of models $h_1, h_2, \ldots, h_M$ and will combine them with weights $\alpha_1, \alpha_2, \ldots, \alpha_M$ such that our final model can be expressed as:

$$\hat{y} = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_M h_M(x)$$

We would like to find models and weights that minimize error here:

$$\underset{\substack{h_1,\ldots,h_M \\ \alpha_1,\ldots,\alpha_M}}{\operatorname{argmin}} \sum_i (y_i - \alpha_1 h_1(x_i) - \alpha_2 h_2(x_i) - \cdots - \alpha_M h_M(x_i))^2$$

Seems hard. Let's take a greedy approach and train one by one.

Example: L2 Boosting (Regression)

Let's assume we've trained the first model $h_1(x)$ and it produces errors:

$$e_i^{(1)} = y_i - h_1(x_i)$$

How can we train the next model to improve over this?

$$h_2 = \underset{h}{argmin} \sum_i \left(e_i^{(1)} - h(x_i)\right)^2$$

Second model simply tries to predict the error. Why does this help?

$$y_i = h_1(x_i) + e_i^{(1)} \Rightarrow y_i = h_1(x_i) + h_2(x_i)$$

Example: L2 Boosting (Regression)

More generally, let the error after the m^{th} model be:

$$e_i^{(m)} = y_i - \sum_{j=1}^{m} h_j(x_i)$$

Train the next classifier against this error:

$$h_{m+1} = \underset{h}{argmin} \sum_i \left(e_i^{(m)} - h(x_i)\right)^2$$

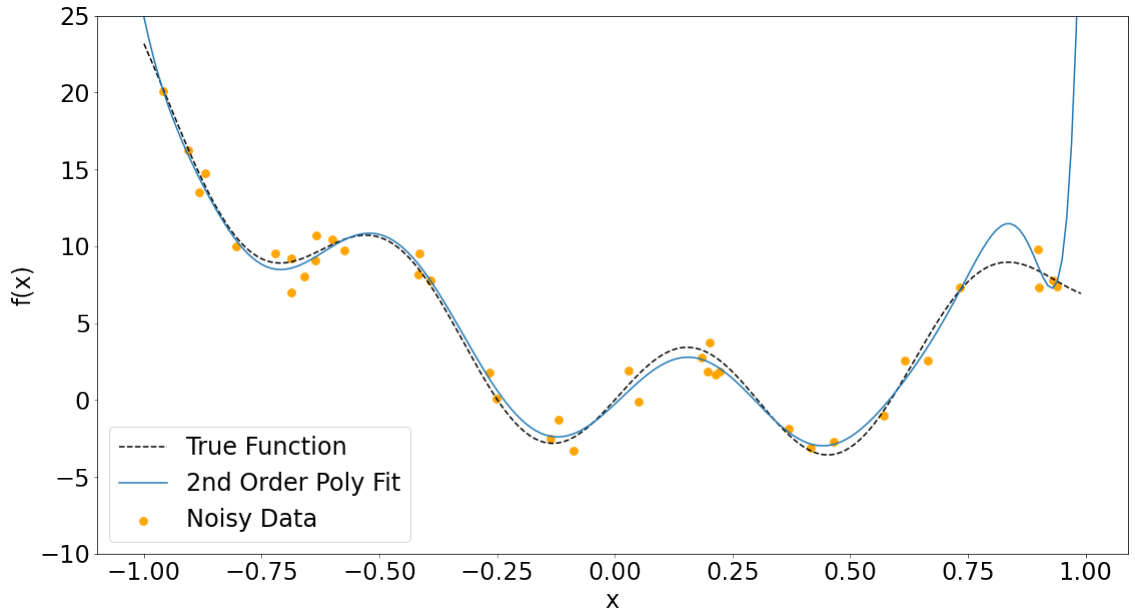
Easy to implement, just replacing y_i with $e_i^{(m)}$ in your favorite regression algorithm (e.g. regression trees).

Example: L2 Boosting (Regression)

Let's see this in practice.

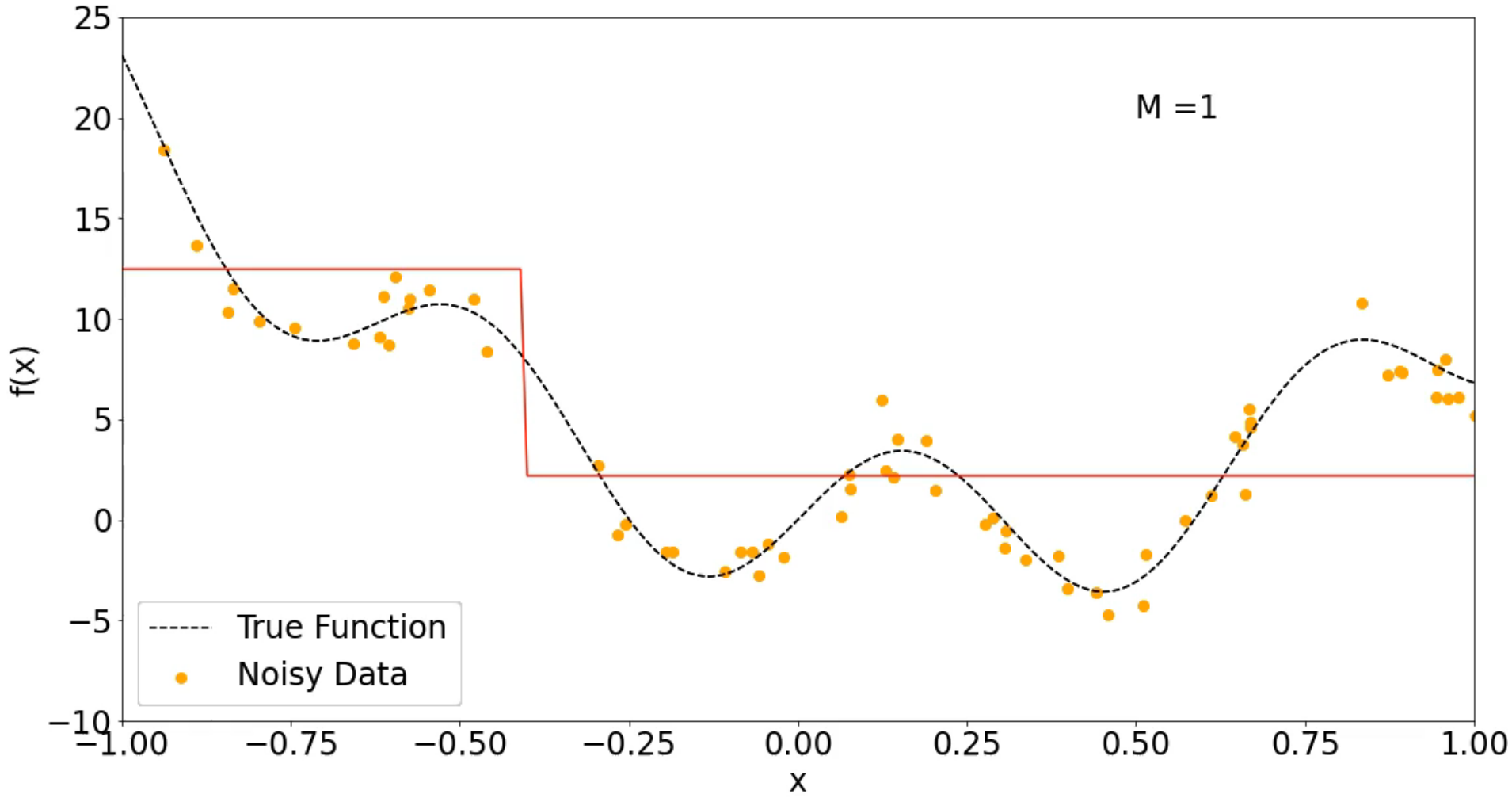
Model: Decision tree for regression, max depth 2.

Data: Same squiggly thing I keep showing.



Example: L2 Boosting (Regression)

M is number of depth-2 regression trees in the boosted ensemble.



Example: AdaBoost (Classification)

Consider a classification setting. AdaBoost operates as follows:

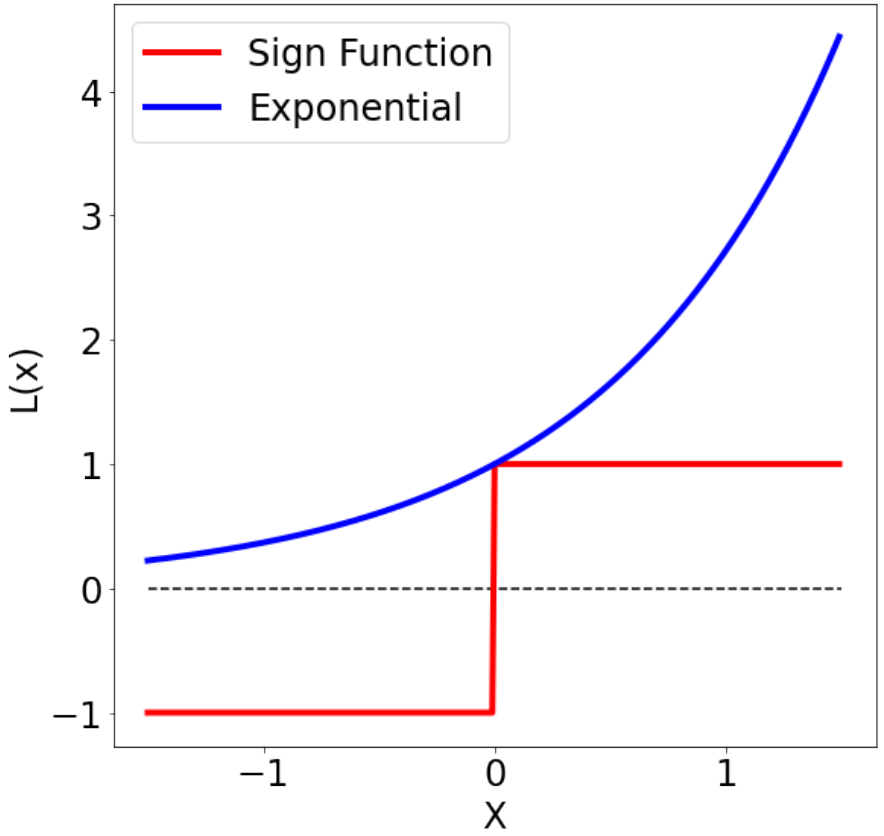
1. Initialize an importance weight w_i for each datapoint uniformly.
2. Train a classifier h_m to minimize weighted error on these points.
3. Add the classifier to the ensemble with a weight α_m according to it's performance
4. Update the weights based on the ensemble's error.
5. Repeat from 2.

Example: AdaBoost (Classification)

Consider a classification setting where we are minimizing errors (e.g., perceptron) where the class labels are {-1, 1}. We will do this by minimizing the exponential loss.

$$\underset{\alpha_1 \ldots, \alpha_M, h_1, \ldots, h_M}{argmin} \sum_i e^{-y_i(\alpha_1 h_1(x_i) + \cdots + \alpha_M h_M(x_i))}$$

Exponential loss forms an upper bound on the sign function. Datapoint is an error if $-y_i(\alpha_1 h_1(x_i) + \cdots + \alpha_M h_M(x_i)) > \mathbf{0}$, so we are minimizing errors by minimizing this loss.



Example: AdaBoost (Classification)

Let's assume we've trained the first model $h_1(x)$. How can we train the next model to improve over this?

$$\alpha_2, h_2 = \underset{\alpha,h}{argmin} \sum_i e^{-y_i(\alpha_1 h_1(x_i) + \alpha h(x_i))}$$

$$= \underset{\alpha,h}{argmin} \sum_i e^{-y_i \alpha_1 h_1(x_i)} * e^{-y_i \alpha h(x_i)}$$

Let $w_i^{(k)} = e^{-y_i(\alpha_1 h_1(x_i) + \ldots + a_k h_k(x_i))}$

$$\alpha_m, h_m = \underset{\alpha,h}{argmin} \sum_i w_i^{(m-1)} e^{-y_i \alpha h(x_i)}$$

Example: AdaBoost (Classification)

Adaboot Algorithm (Training):

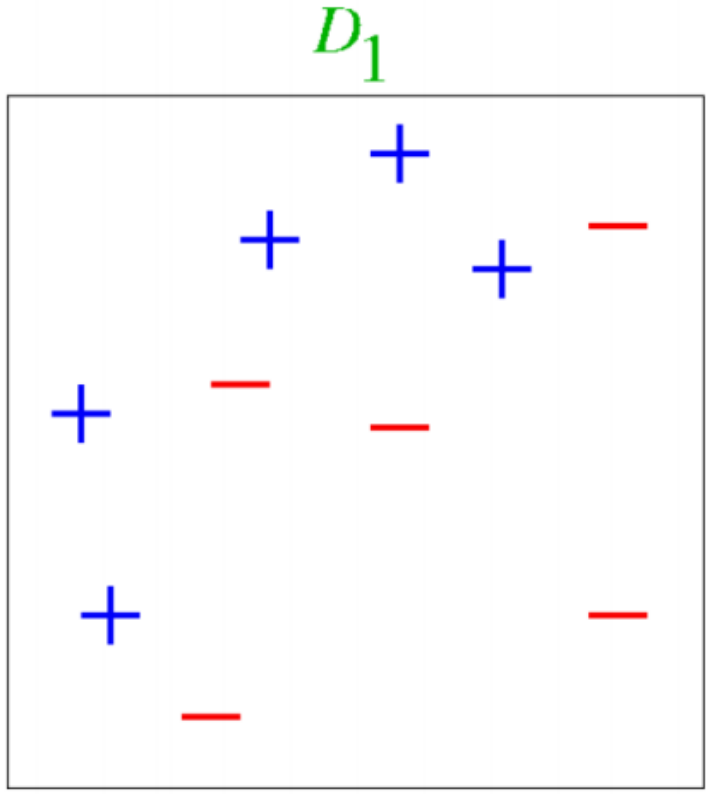
1. Initialize all importance weights a $w_i^{(0)} = 1$
2. For m in range(M):
    1. Train classifier h_m to minimize weighted exponential loss $\sum_i w_i^{(m-1)} e^{-y_i h_m(x_i)}$
    2. Compute the weighted error rate of the classifier as: $\epsilon_m = \frac{\sum_i w_i^{(m-1)} I[y_i h_m(x_i) > 0]}{\sum_j w_j^{(m-1)}}$
    3. Compute classifier quality as $\alpha_m = \frac{1}{2} \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$
    4. Update weights $w_i^{(m)} = w_i^{(m-1)} * e^{-y_i \alpha_m h_m(x_i)}$

Example: AdaBoost (Classification)

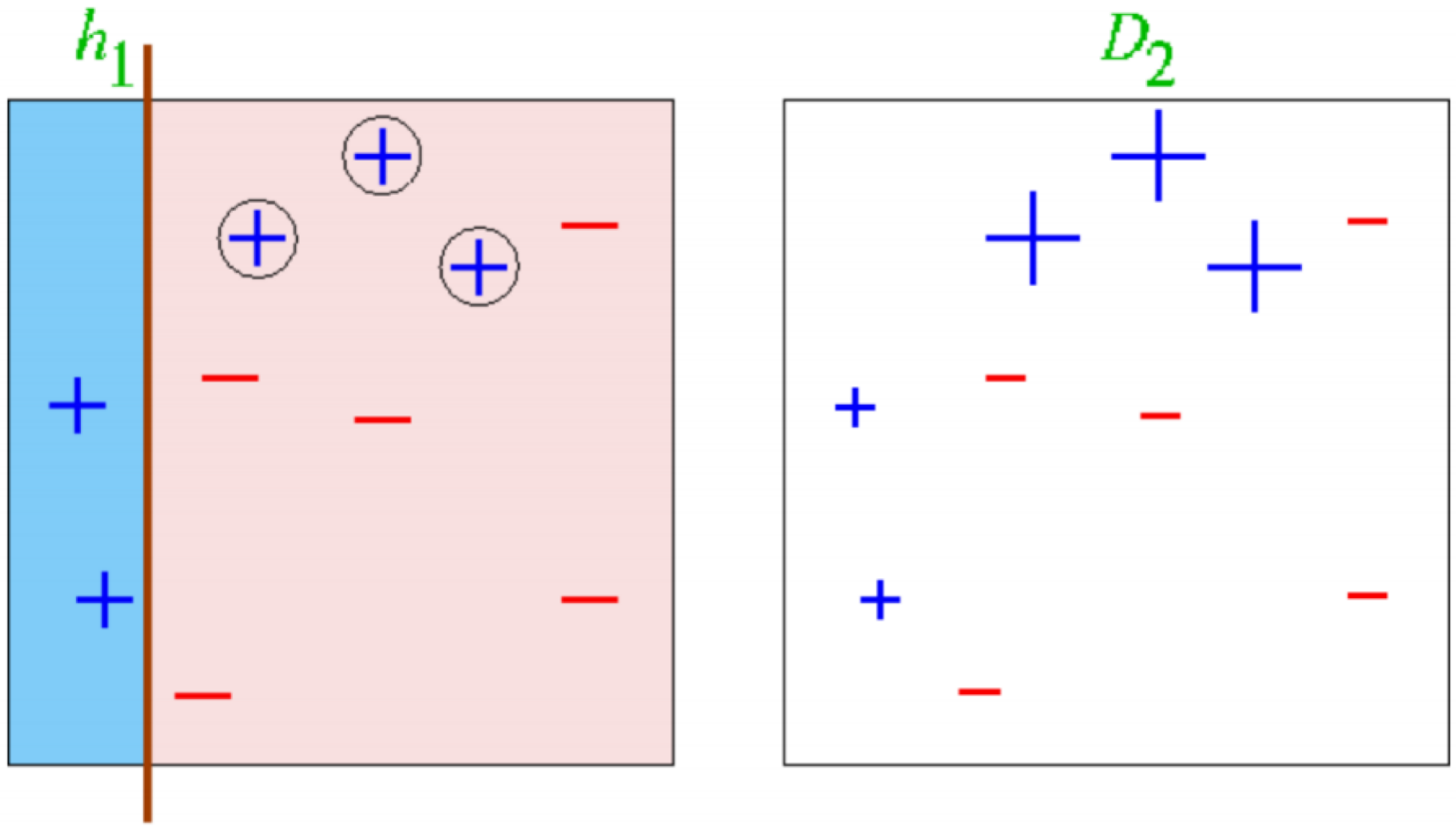
Adaboot Algorithm (Test):

Compute prediction as $sign\left(\alpha_1 h_1(x) + \ \ldots \ + \alpha_M h_M(x)\right)$

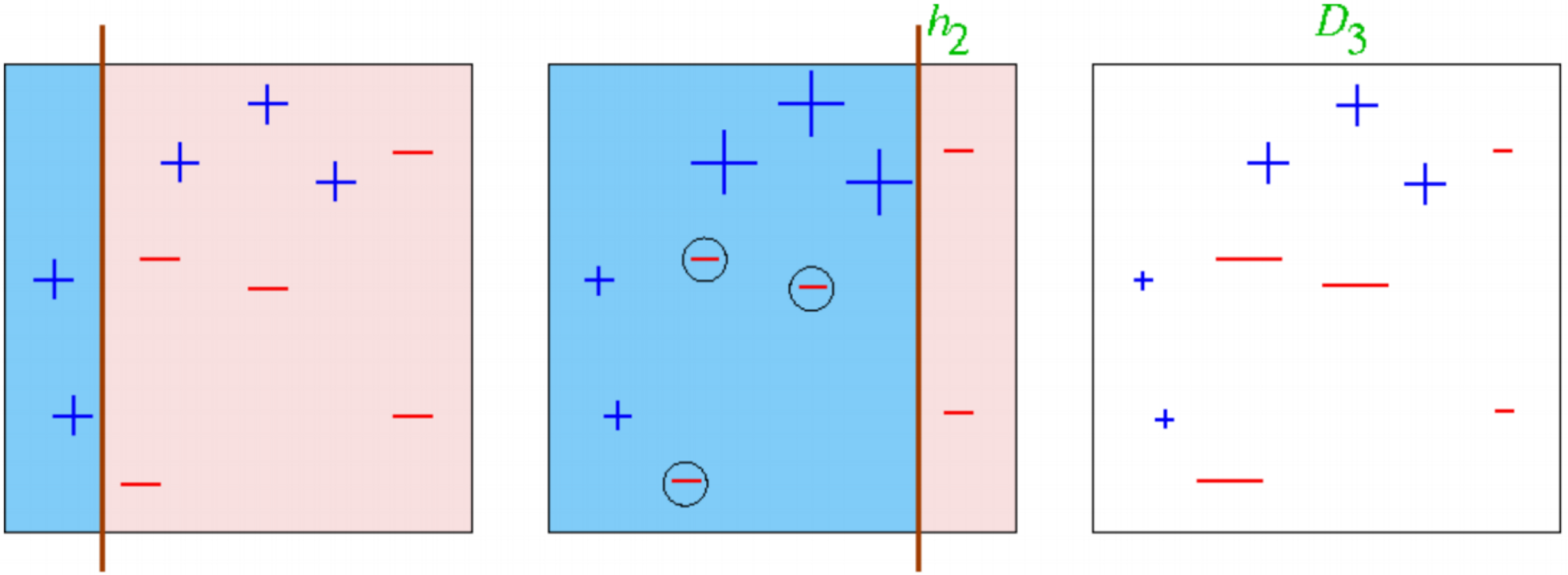
Example: AdaBoost (Classification)



Example: AdaBoost (Classification)



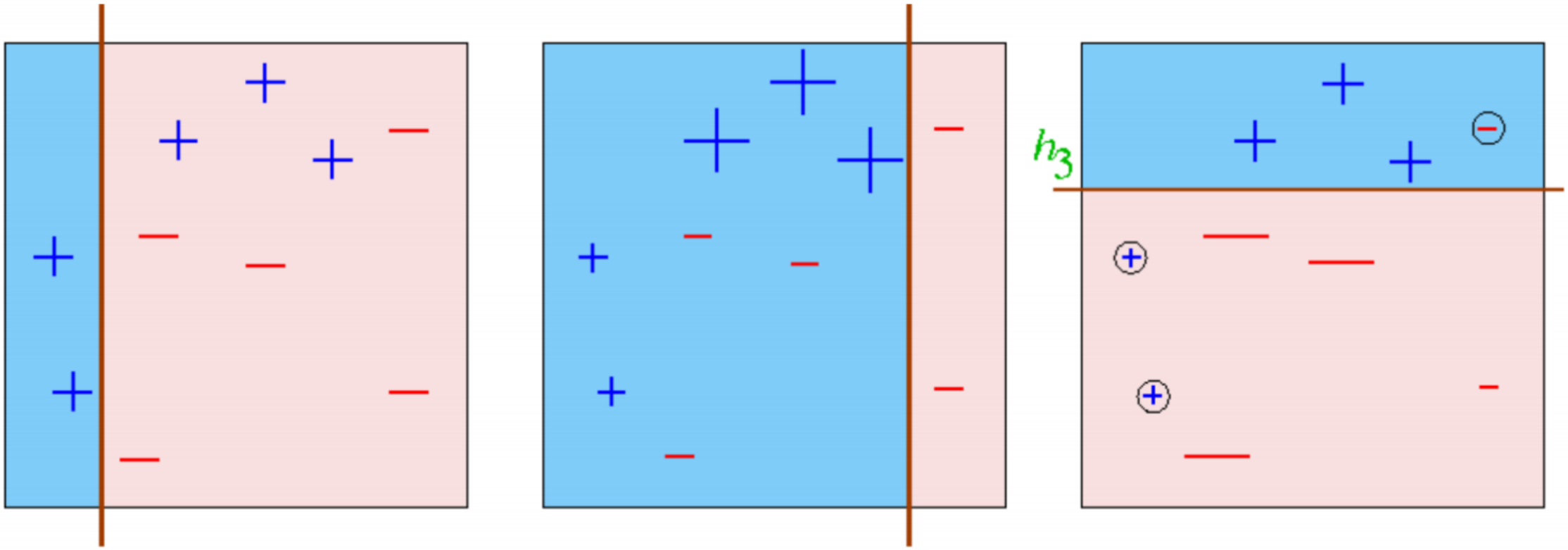
Example: AdaBoost (Classification)



$\varepsilon_2 = 0.21$

$\alpha_2 = 0.65$

Example: AdaBoost (Classification)

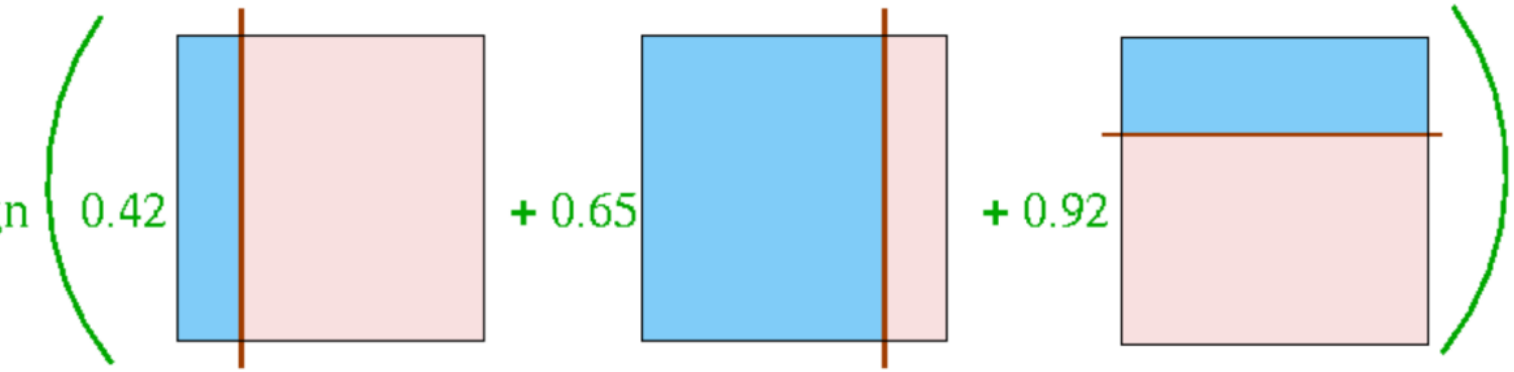


$\varepsilon_3 = 0.14$

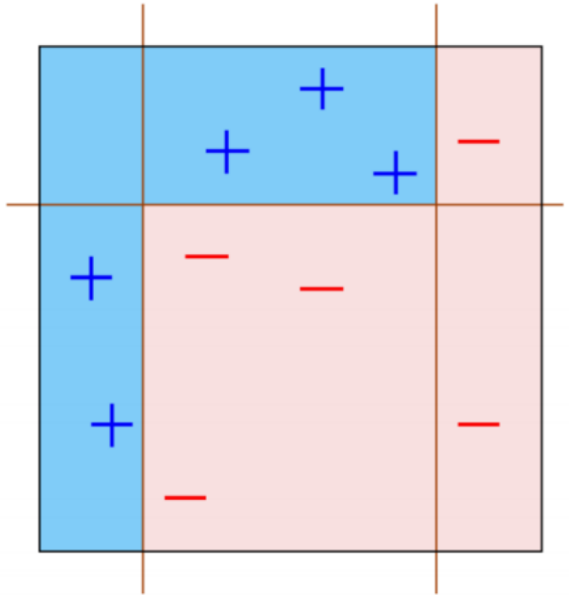
$\alpha_3 = 0.92$

Example: AdaBoost (Classification)

$$H_{\text{final}} = \text{sign}\left(0.42 \; + 0.65 \; + 0.92 \right)$$



$=$



Lots of Different Boosting Algorithms

Loss Name	Loss Formula	Boosting Name
Regression: Squared Loss	$(y - f(x))^2$	L2Boosting
Regression: Absolute Loss	$\|y - f(x)\|$	Gradient Boosting
Classification: Exponential Loss	$e^{-yf(x)}$	AdaBoost
Classification: Log/Logistic Loss	$\log\left(1 + e^{-yf(x)}\right)$	LogitBoost

Generally, will assume you can train a model on a weighted dataset.

Summary of Ensemble Methods

Ensembles are sets of models trained for the same problem.

- Ensemble output is often the average (regression) or majority (classification) class output.
- Ensembles almost always lead to improvements over single models and are widely used in practice.

Bagging is a way to make an ensemble that reduces variance of strong models.

- Resample dataset with replacement to learn each model in the ensemble.
- Models can be trained in parallel.
- One example we talked about is **Random Forests.**

Boosting is a way to make an ensemble that reduces bias of weak models.

- Sequentially learn classifiers that focus more on datapoints where the current models have errors.

Today's Learning Objectives

Be able to answer:

- ~~Wrap up Decision Trees.~~
- ~~What is the bias-variance tradeoff?~~
- ~~What is an ensemble?~~
  - ~~How do ensembles reduce error?~~
- ~~What is bagging?~~
  - ~~What are random forests?~~
- ~~What is boosting?~~
  - ~~How does L2 boosting work?~~
  - ~~How does Adaboost work?~~

Next time

Next Time: We'll leave supervised learning behind and move on to topics in unsupervised learning.