

# CSE417T – Lecture 16

- Please **mute** yourself and **turn off videos** to save bandwidth.
- If you have questions during the lecture
  - Use slido to type your questions
    - **Join at [slido.com](https://slido.com), #7738**
    - You can ask new questions or upvote others' questions
    - I'll try to answer them in batches
  - You can also un-mute yourself and ask the questions directly.

# Logistics: Homework

- The deadline of HW3 is extended to March 27 (Friday).
- Late days for homework assignments
  - Everyone is getting **2 more late days**
  - You can use up to **3 late days for HW3 and HW4** (if you still have late days left)
  - You can use up to **2 late days for HW5**
- HW4 is a programming-heavy assignment. I'd recommend you to reserve late days for it.

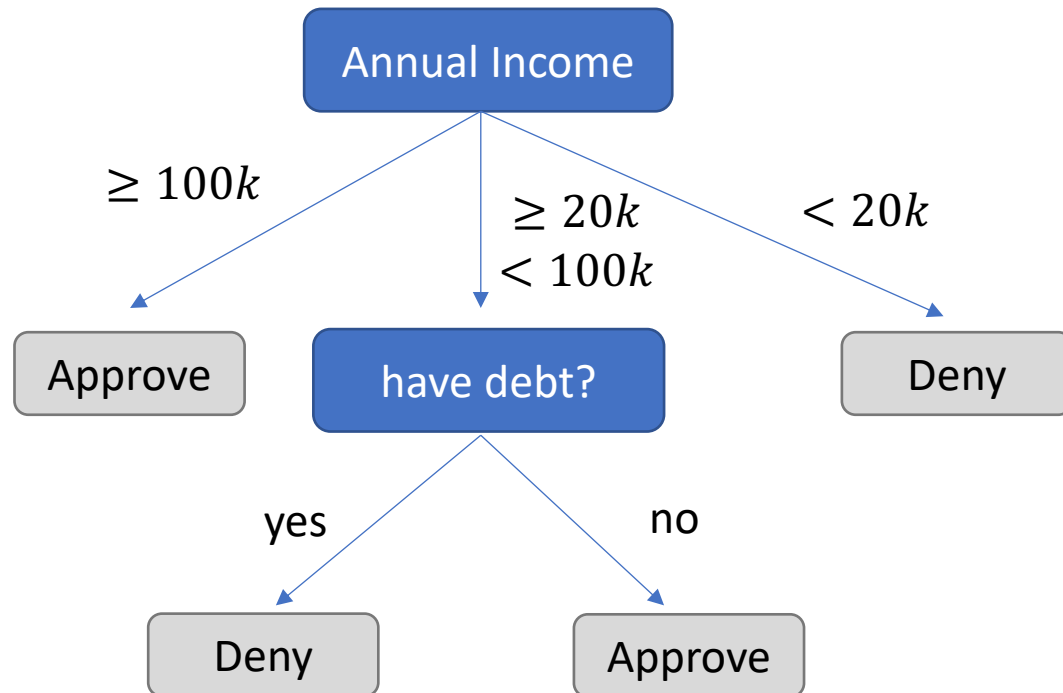
# Questions about Online Instruction?

- Check Piazza posts for updates
- Summary of the current plan
  - Lectures and office hours are online with Zoom
  - Homework: the same as before, but with modified late-day policies
  - Exam 2: more to come in a week or two
- Ask questions on slido.com, #7738
- Consider this week as the public beta
  - There will be compromises, but we'll try to adapt as we go

Recap

# Decision Tree

# Decision Tree Hypothesis



Credit Card Approval Example

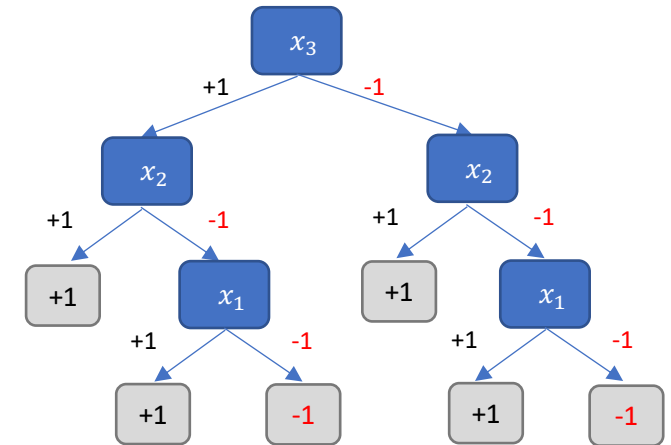
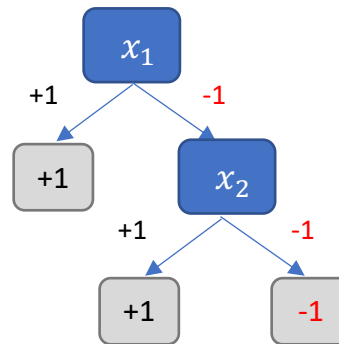
- Pros
  - Easy to interpret (interpretability is getting attention and is important in some domains)
  - Can handle multi-type data (Numerical, categorical. ...)
  - Easy to implement (Bunch of if-else rules)
- Cons
  - Generally speaking, **bad generalization**
  - VC dimension is infinity
  - High variance (small change of data leads to very different hypothesis)
  - Easily overfit
- Why we care?
  - One of the classical model
  - Building block for other models (e.g., random forest)

# Learning Decision Tree from Data

- Given dataset  $D$ , how to learn a decision tree hypothesis?
- Potential approach
  - Find  $g = \operatorname{argmin}_{h \in H} E_{in}(h)$

$x_1$	$x_2$	$x_3$	$y$
+1	+1	+1	+1
+1	+1	-1	+1
+1	-1	+1	+1
+1	-1	-1	+1
-1	+1	+1	+1
-1	+1	-1	+1
-1	-1	+1	-1
-1	-1	-1	-1

- Multiple decision trees with zero  $E_{in}$



# Learning Decision Tree from Data

- Conceptual intuition to deal with overfitting
  - Regularization: **Constrain  $H$**
- Informally, 

$$\begin{array}{ll} \text{minimize} & E_{in}(\vec{w}) \\ \text{subject to} & \text{size}(tree) \leq C \end{array}$$
- This optimization is generally computationally intractable.
- Most decision tree learning algorithms rely on **heuristics** to approximate the goal.



# Greedy-Based Decision Tree Algorithm

- Greedily, recursively, choose the next feature to split
- DecisionTreeLearn( $D$ ): Input a dataset  $D$ , output a decision tree hypothesis
  - Create a root node  $r$
  - If **termination conditions** are met
    - return a single node tree with **leaf prediction** based on  $D$
  - Else: Greedily find a feature  $A$  to split according to **split criteria**
    - For each possible value  $v_i$  of  $A$ 
      - Let  $D_i$  be the dataset containing data with value  $v_i$  for feature  $A$
      - Create a subtree DecisionTreeLearn( $D_i$ ) that being the child of root  $r$
- Most decision tree learning algorithms follow this template, but with different choices of **heuristics**

# ID3: Using Information Gain as Selection Criteria

- Information gain of choosing feature  $A$  to split
  - $Gain(D, A) = H(D) - \sum_i \frac{|D_i|}{|D|} H(D_i)$  [The amount of decrease in entropy]
- ID3: Choose the split that maximize  $Gain(D, A)$

Notations:

$H(D)$ : Entropy of  $D$

$|D|$  is the number of points in  $D$

DecisionTreeLearn( $D$ )

Create a root node  $r$

If **termination conditions** are met

return a single node tree with **leaf prediction** based on

Else: Greedily find a feature  $A$  to split according to **split criteria**

For each possible value  $v_i$  of  $A$

Let  $D_i$  be the dataset containing data with value  $v_i$  for feature  $A$

Create a subtree DecisionTreeLearn( $D_i$ ) that being the child of root  $r$

- ID3 termination conditions
  - If all labels are the same
  - If all features are the same
  - If dataset is empty
- ID3 leaf predictions
  - Most common labels (majority voting)
- ID3 split criteria
  - Information gain

# ID3: Using Information Gain as Selection Criteria

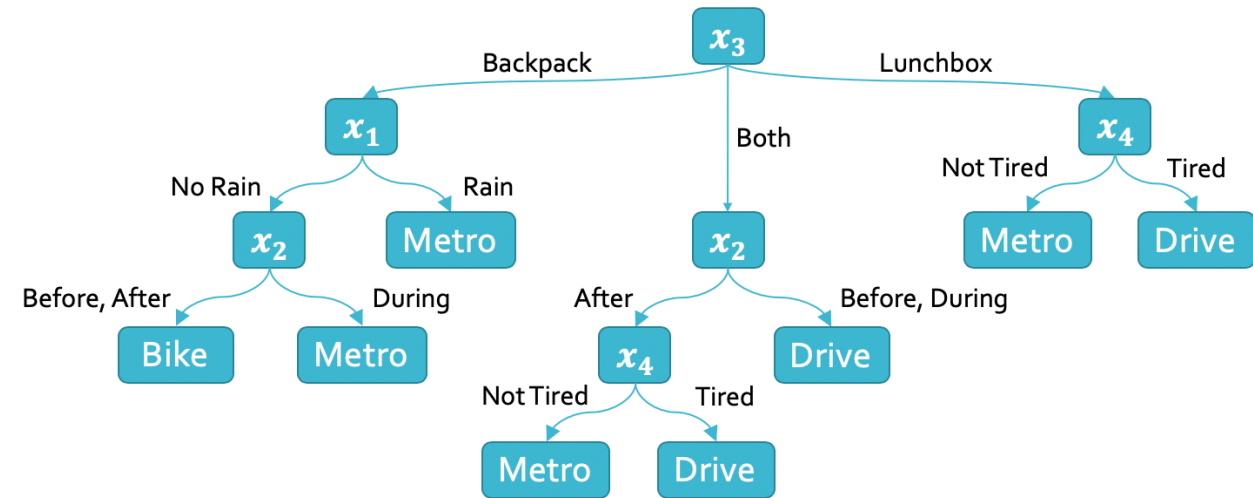
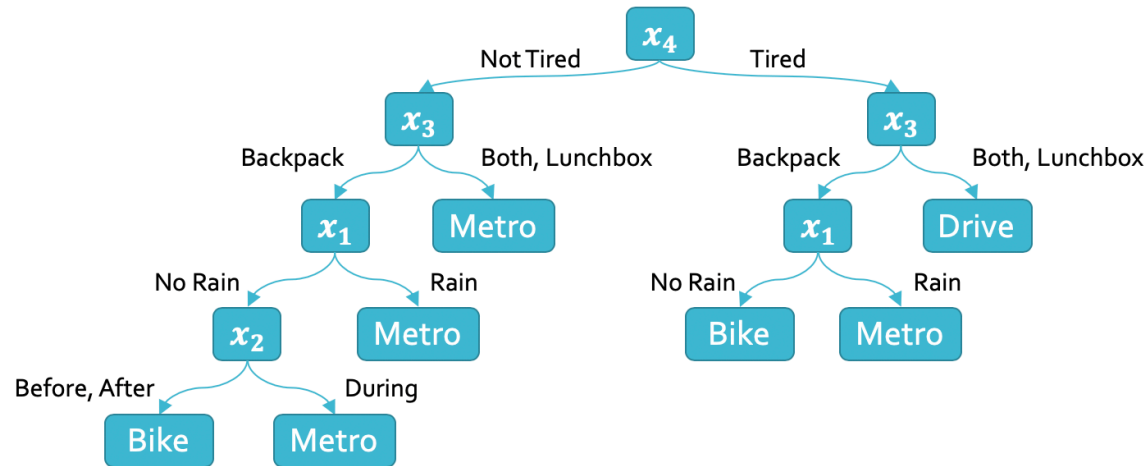
- A greedy algorithm (selects the features with the maximum information gain to split) with no theoretical guarantee.
- Overfitting concerns
  - Apply more regularizations (limiting the tree size, pruning splits with validations)
- “high variance” (in terms of bias-variance decomposition)

# Illustration of "High Variance" of Decision Trees

$x_1$	$x_2$	$x_3$	$x_4$	$y$
Rain	Before	Both	Tired	Drive
Rain	During	Both	Not Tired	Metro
Rain	During	Both	Tired	Drive
Rain	After	Backpack	Not Tired	Metro
Rain	After	Backpack	Tired	Metro
Rain	After	Lunchbox	Tired	Drive
No Rain	Before	Backpack	Tired	Bike
No Rain	Before	Lunchbox	Not Tired	Metro
No Rain	Before	Lunchbox	Tired	Drive
No Rain	During	Backpack	Not Tired	Metro
No Rain	During	Both	Tired	Drive
No Rain	After	Backpack	Not Tired	Bike
No Rain	After	Backpack	Tired	Bike
No Rain	After	Both	Not Tired	Metro
No Rain	After	Both	Tired	Drive
No Rain	After	Lunchbox	Not Tired	Metro

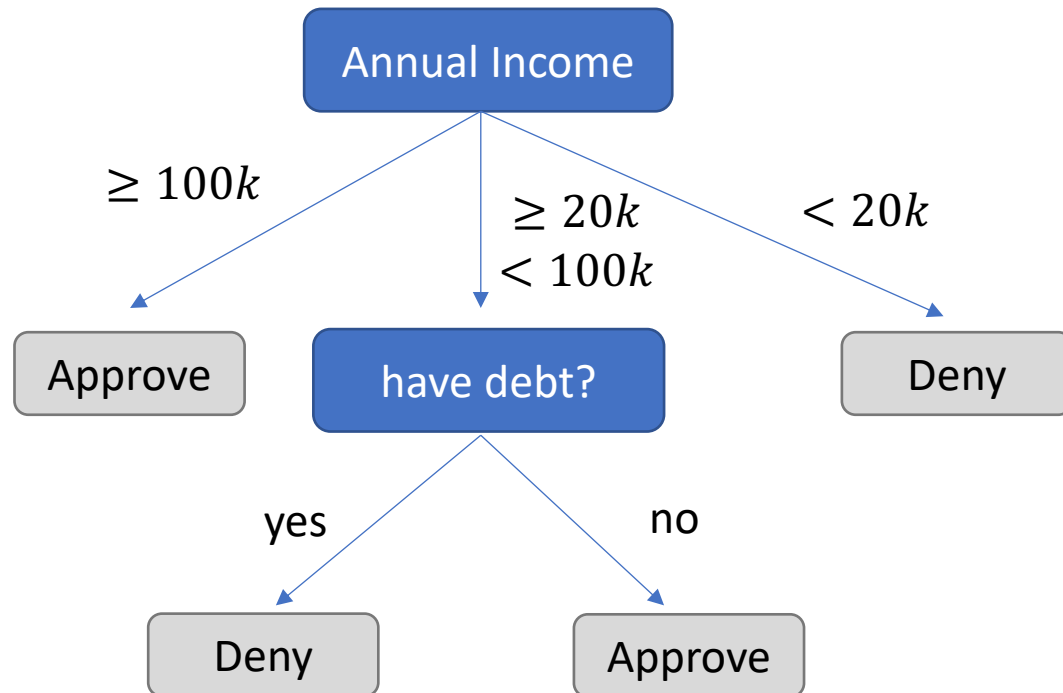
$x_1$	$x_2$	$x_3$	$x_4$	$y$
Rain	Before	Both	Tired	Drive
Rain	During	Both	Not Tired	Drive
Rain	During	Both	Tired	Drive
Rain	After	Backpack	Not Tired	Metro
Rain	After	Backpack	Tired	Metro
Rain	After	Lunchbox	Tired	Drive
No Rain	Before	Backpack	Tired	Bike
No Rain	Before	Lunchbox	Not Tired	Metro
No Rain	Before	Lunchbox	Tired	Drive
No Rain	During	Backpack	Not Tired	Metro
No Rain	During	Both	Tired	Drive
No Rain	After	Backpack	Not Tired	Bike
No Rain	After	Backpack	Tired	Bike
No Rain	After	Both	Not Tired	Metro
No Rain	After	Both	Tired	Drive
No Rain	After	Lunchbox	Not Tired	Metro

# Illustration of "High Variance" of Decision Trees



High variance: A small deviation of data would lead to very different learned hypothesis

# Decision Tree Hypothesis



Credit Card Approval Example

- Pros
  - Easy to interpret (interpretability is getting attention and is important in some domains)
  - Can handle multi-type data (Numerical, categorical. ...)
  - Easy to implement (Bunch of if-else rules)
- Cons
  - Generally speaking, **bad generalization**
  - VC dimension is infinity
  - High variance (small change of data leads to very different hypothesis)
  - Easily overfit
- Why we care?
  - One of the classical model
  - **Building block for other models**

# Lecture Notes Today

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.  
Let me know if you spot errors.

# Ensemble Learning



# Ensemble Learning

- Assume we are given a set of learned hypothesis
  - $g_1, g_2, \dots, g_M$
- What can we do?
  - Select the best one: use validation for model selection
  - What if all of them are not good enough?
- Can we **aggregate** them?

# Aggregation

- Given a set of **weak learners**  $g_1, \dots, g_M$ , how to output a **stronger learner** that has better performance.

Mathematically, majority voting and average is similar with +1/-1 labels

- Uniform aggregation

- Regression (average):  $\bar{g}(\vec{x}) = \frac{1}{M} \sum_{m=1}^M g_m(\vec{x})$
- Classification (majority vote):  $\bar{g}(\vec{x}) = \text{sign} \left( \frac{1}{M} \sum_{m=1}^M g_m(\vec{x}) \right)$

- Weighted aggregation

- Regression (average):  $\bar{g}(\vec{x}) = \frac{1}{M} \sum_{m=1}^M \alpha_m g_m(\vec{x})$
- Classification (majority vote):  $\bar{g}(\vec{x}) = \text{sign} \left( \frac{1}{M} \sum_{m=1}^M \alpha_m g_m(\vec{x}) \right)$

- Stacking (won't talk about this in this course)

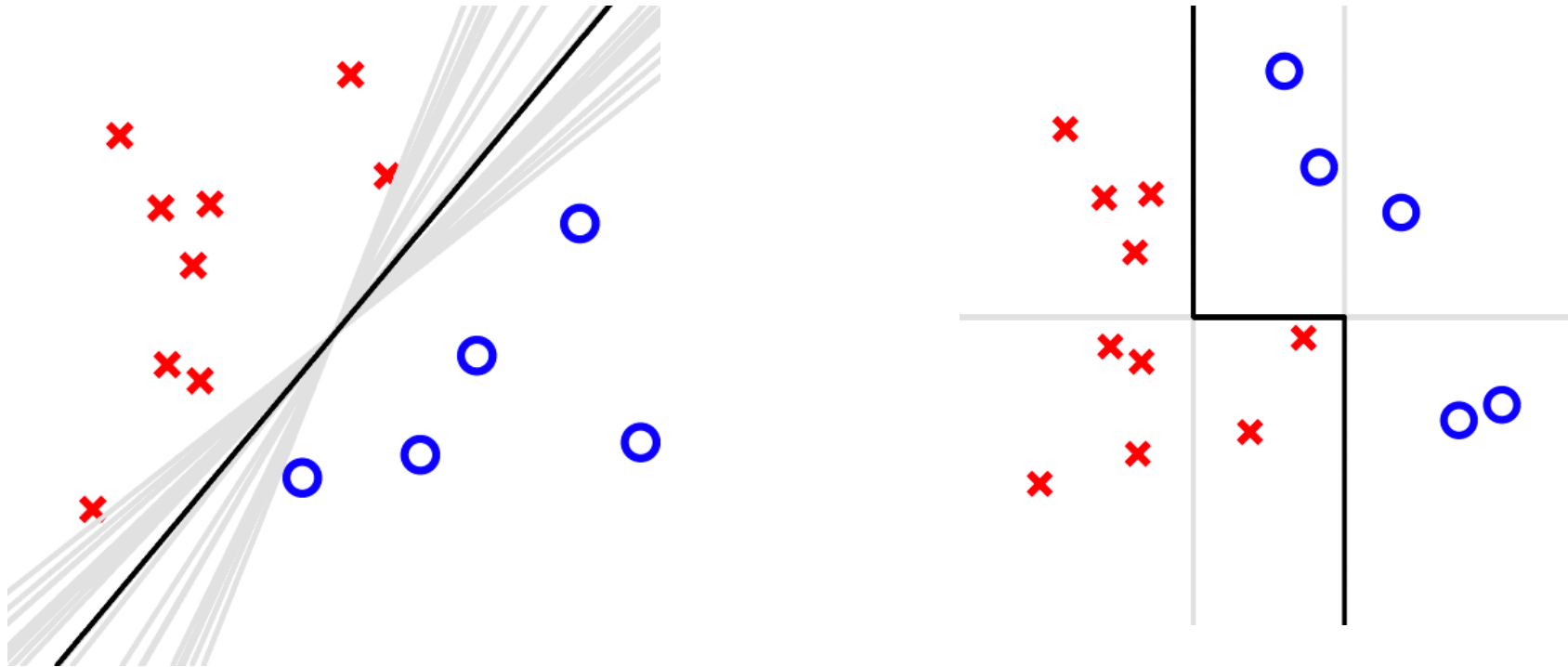
- Take the prediction of  $g_1$  to  $g_m$  as input, train another model on top of that

# Is Aggregation a Good Idea?

- Some illustrative examples

# Is Aggregation a Good Idea?

- Some illustrative examples



# Is Aggregation a Good Idea?

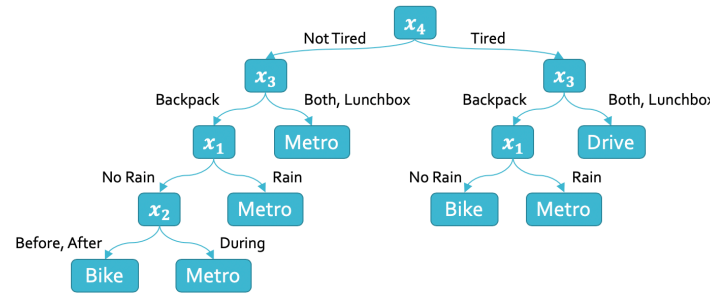
- Maybe
  - If the hypothesis is **diverse**, and “on average” they seem good
  - (If you take humans as weak learners, this is almost democracy)
- Question:
  - How do we **find** a set of hypothesis that are **diverse** and “in average” good
  - How do we **aggregate** the set of hypothesis
- Ensemble learning
  - Bagging – Random Forest (This lecture)
  - Boosting – AdaBoost (Next lecture)

# Diverse Weak Learners

- One common way to construct weak learners is via **decision trees**

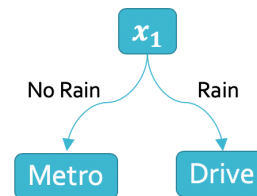
- Fully grown decision trees

- High variance
- Low bias



- Decision stump (One-depth decision trees, split on only one attribute)

- Low variance
- High bias



- We will discuss how to construct diverse weak learners next

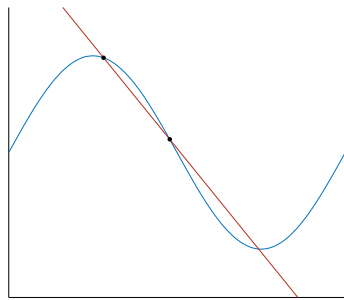
- Hint: Randomization

# Bagging

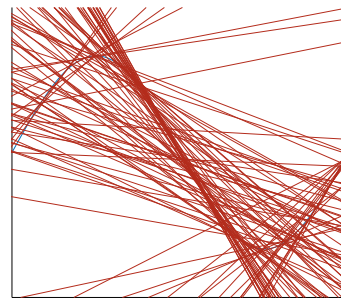
Bootstrapped Aggregating

# Review: Bias-Variance Decomposition

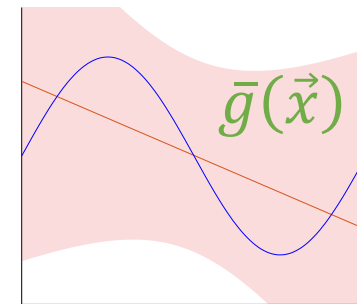
- $f$ : sine function,  $H: h(x) = ax + b$ ,  $N=2$



For each dataset,  
learn a hypothesis.



Draw many datasets,  
learn many hypothesis



Take average.

- Observations

- The variance of each learned hypothesis is high
- The variance of “average” of them ( $\bar{g}(\vec{x})$ ) is lower

- Can we apply similar intuitions?

- Generate a lot of **high-variance but low bias** weak learners
- Aggregate them using uniform aggregation

We only have one  
dataset in practice!



# Bootstrapping

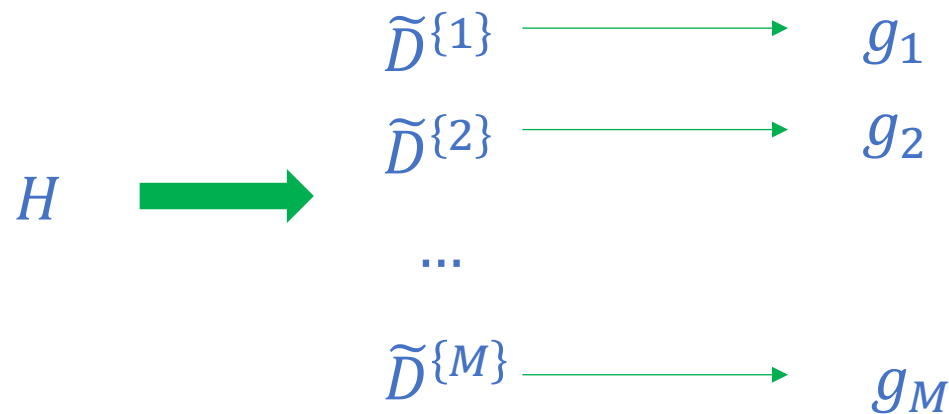
- Intuition:
  - Use the dataset  $D$  we have to approximate the data distribution
  - Sample (with replacement) from  $D$  to create bootstrapped datasets
- Bootstrapping:
  - Let  $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$  be the dataset we have
  - Repeatedly uniformly sample  $N$  points from  $D$  with replacement
    - The number of sampled points doesn't have to be  $N$ , but it's a reasonable choice.
  - Obtain many bootstrapped datasets
    - $\tilde{D}^{(1)} = \{(x_1, y_1), (x_1, y_1), (x_4, y_4), \dots\}$
    - ...
    - $\tilde{D}^{(M)}$

# Bagging - Bootstrapped Aggregating

- Bootstrap  $M$  datasets  $\{\tilde{D}^{\{m\}}\}$  and learn a hypothesis from each of them
- Aggregate the learned hypothesis (assume we are doing classification)

# Bootstrapped Aggregating - Bagging

- Bootstrap  $M$  datasets  $\{\tilde{D}^{\{m\}}\}$  and learn a hypothesis from each of them



- Aggregate the learned hypothesis (assume we are doing classification)

$$G(\vec{x}) = \bar{g}(\vec{x}) = \text{sign} \left( \frac{1}{M} \sum_{m=1}^M g_m(\vec{x}) \right)$$

# Why/When Bagging Might Be Helpful?

- What we know from statistics
  - Consider  $M$  independent random variables  $x_1, x_2, \dots, x_M$  with variance  $\sigma^2$
  - The variance of  $\frac{1}{M} \sum_{m=1}^M x_m$  is  $\frac{\sigma^2}{M}$
- If you have “weak learners” that have **high variance** but **low bias**
  - Bagging helps **reduce the variance** and **maintain low bias**
  - From bias-variance decomposition, this implies a strong learner

# Break and Question Time

Exercise:

Given a dataset  $D$  with  $N$  points. Consider we bootstrap a dataset  $\tilde{D}^{(m)}$  by sampling  $N$  points with replacement from  $D$ , what's the probability that a given point  $(\vec{x}_n, y_n)$  is not in  $\tilde{D}^{(m)}$  ?

Out-Of-Bag (OOB) Error

# Probability for a Point to be Out of Bag

- Consider we bootstrap a dataset  $\tilde{D}^{(m)}$  by sampling  $N$  points from  $D$ , what's the probability that a given point  $(\vec{x}_n, y_n)$  is not in  $\tilde{D}^{(m)}$ .

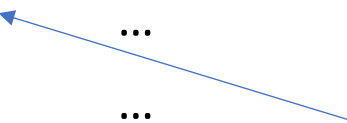
$$\begin{aligned} & \left(1 - \frac{1}{N}\right)^N \\ &= \left(\frac{1}{1 + \frac{N}{N-1}}\right)^N \\ &\approx \frac{1}{e} \approx 0.36 \text{ when } N \rightarrow \infty \end{aligned}$$

When  $N$  is large, for each bootstrapped dataset  $\tilde{D}^{(m)}$ , a significant proportion of points in  $D$  is not included.

- A point is not in  $\tilde{D}^{(m)}$  is not involved in training  $g_m$ 
  - Can we utilize it to **validate** the performance of  $g_m$ ?
  - Yes, but we care about the overall performance, not just the performance of  $g_m$ ...

# Out-Of-Bag (OOB) Error

	$\tilde{D}^{(1)}$	$\tilde{D}^{(2)}$	$\tilde{D}^{(3)}$	$\tilde{D}^{(4)}$	...
$(\vec{x}_1, y_1)$	Yes	Yes	No	No	...
$(\vec{x}_2, y_2)$	Yes	No	No	No	...
...	...	...	...	...	...
$(\vec{x}_N, y_N)$	No	Yes	Yes	Yes	...



Whether a point is in a bootstrapped dataset

- Recall that we learn  $g_1, \dots, g_M$  using  $\tilde{D}^{(1)}, \dots, \tilde{D}^{(M)}$
- Which set of hypothesis can  $(\vec{x}_1, y_1)$  be used for **validation**?



# Out-Of-Bag (OOB) Error

	$\tilde{D}^{(1)}$	$\tilde{D}^{(2)}$	$\tilde{D}^{(3)}$	$\tilde{D}^{(4)}$	...
$(\vec{x}_1, y_1)$	Yes	Yes	No	No	...
$(\vec{x}_2, y_2)$	Yes	No	No	No	...
...	...	...	...	...	...
$(\vec{x}_N, y_N)$	No	Yes	Yes	Yes	...

Whether a point is in a bootstrapped dataset

- $G_n^-$ : the aggregation of hypothesis that  $\vec{x}_n$  is OOB of

- $G_1^- = \text{aggregate}(g_3, g_4, \dots)$
- $G_2^- = \text{aggregate}(g_2, g_3, g_4, \dots)$
- $G_N^- = \text{aggregate}(g_1, \dots)$

Aggregate:  
Majority voting for classification  
Average for regression

- OOB Error

- $E_{OOB}(G) = \frac{1}{N} \sum_{n=1}^N \text{error}(G_n^-(\vec{x}_n), y_n)$

Error:  
Binary error for classification  
Squared error for regression

# Out-Of-Bag (OOB) Error

$$E_{OOB}(G) = \frac{1}{N} \sum_{n=1}^N \text{error}(G_n^-(\vec{x}_n), y_n)$$

- Bagging provided an **intrinsic** mechanism for us to perform validation.
  - We don't need to split the dataset into training and validation
- Practical issues (you might face this in HW4)
  - What if some  $\vec{x}_n$  appears in all bootstrapped datasets?
    - The probability of this happening is small when the number of bags  $M$  is large
  - Let  $S$  be the set of points that is out of bag for at least one bootstrapped dataset
    - $E_{OOB}(G) = \frac{1}{|S|} \sum_{(\vec{x}_n, y_n) \in S} \text{error}(G_n^-(\vec{x}_n), y_n)$

Random Forest

# What We Have Learned

## Bagging:

A method to generate and aggregate many **high-variance** weak learners into a stronger one.



## Decision tree:

Various nice properties  
Bad generalization  
- Due to **high variance**



## Random Forest:

1. Construct many random trees
2. Aggregate the random trees

# Random Forest

- Construct many random trees
  - Bootstrapping datasets and learn a **max-depth tree** for each of them
  - Other randomizations (not required in HW4)
    - When choosing split features, choose from a random subset (instead of all features)
    - Randomly project features (similar to non-linear transformation) for each tree
- Aggregate the random trees
  - Classification: Majority vote  $\bar{g}(\vec{x}) = \text{sign} \left( \frac{1}{M} \sum_{m=1}^M g_m(\vec{x}) \right)$
  - Regression: Average  $\bar{g}(\vec{x}) = \frac{1}{M} \sum_{m=1}^M g_m(\vec{x})$

# Questions?

- Note that in HW4, you will be asked to implement Bagging Decision Tree and calculate the OOB errors.
- Make sure you know the definitions/algorithm well.

# Brief Discussion on Feature Importance

- Not all features are equally important
  - Some features could be **redundant** -- (birth year, age)
  - Some features might be **irrelevant** -- feature: name, label: prob of heart attack
- How do we know which features are more important?
  - Linear models:
    - The **size of the weight** is a proxy for feature importance
    - Applying L1 regularization is one way to reduce the number of features.
  - Decision tree:
    - The feature **closer to the root** is probably more important
  - Random forest:
    - **Average “information gain”** of all trees is a proxy for feature importance
- See LFD e-Chap 9.2 for more discussion on feature selection