

CSE 417T

# Introduction to Machine Learning

Lecture 3

Instructor: Chien-Ju (CJ) Ho

# Logistics

- Course website and Piazza
  - Website: <http://chienjuho.com/courses/cse417t/>
  - Piazza: <http://piazza.com/wustl/fall2022/cse417t>
  - Make sure you follow both regularly
- Office hours
  - Will be announced later this week
  - Will start next week
- Homework 1
  - Will be announced later this week
  - A mixture of math questions and programming questions
    - Programming language: Python (We won't teach you how to program Python)

Recap

**UNKNOWN TARGET FUNCTION**

$$f : \mathcal{X} \mapsto \mathcal{Y}$$

*(ideal credit approval formula)*

$$y_n = f(\mathbf{x}_n)$$

**TRAINING EXAMPLES**

$$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_N, y_N)$$

*(historical records of credit customers)*

Given by the learning problem

**LEARNING  
ALGORITHM**

$\mathcal{A}$

**FINAL  
HYPOTHESIS**

$$g \approx f$$

*(learned credit approval formula)*

Goal of learning

**HYPOTHESIS SET**

$\mathcal{H}$

*(set of candidate formulas)*

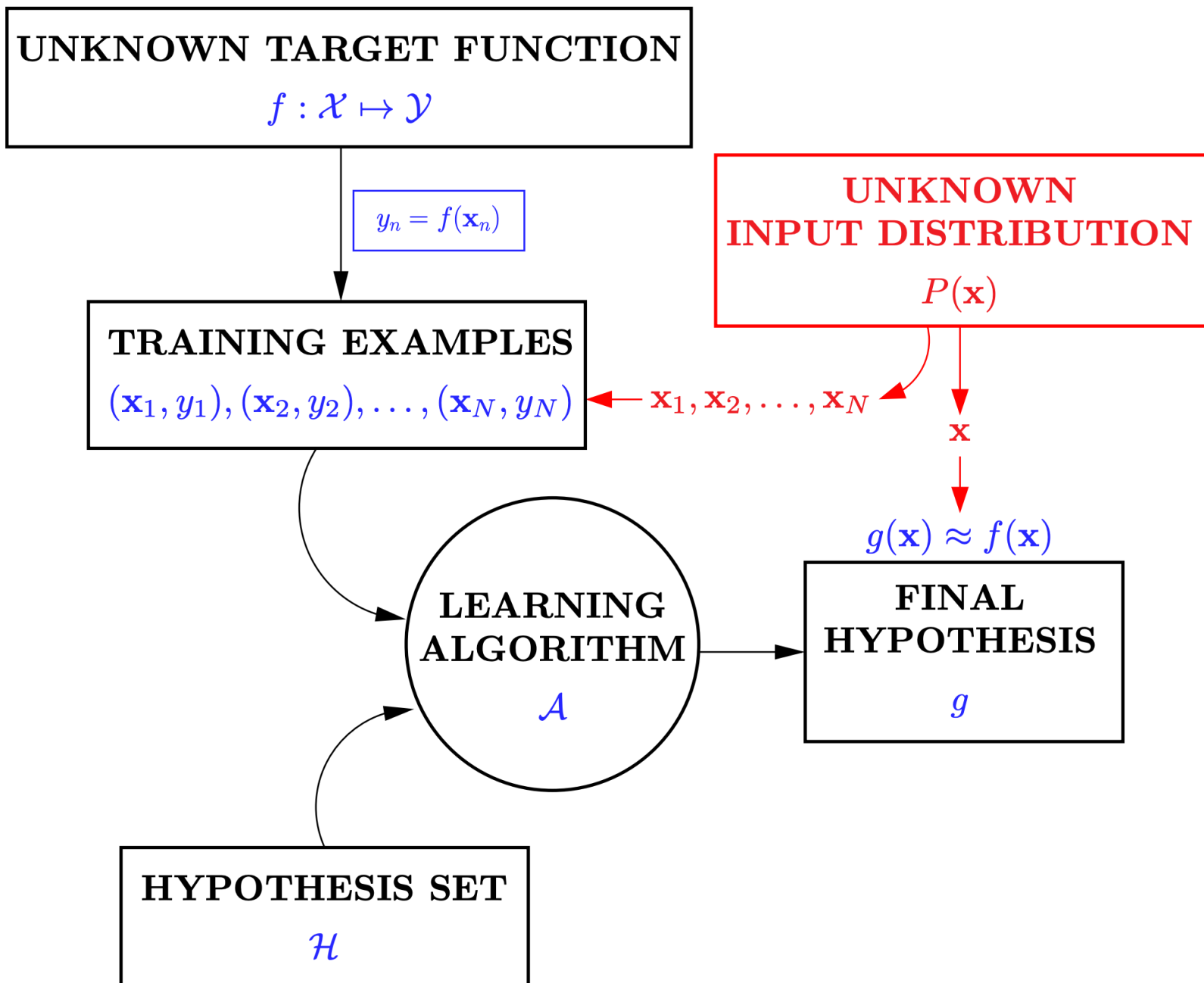
learning model  
(example:  
H: Perceptron  
A: PLA)

# Goal of Learning: Generalization

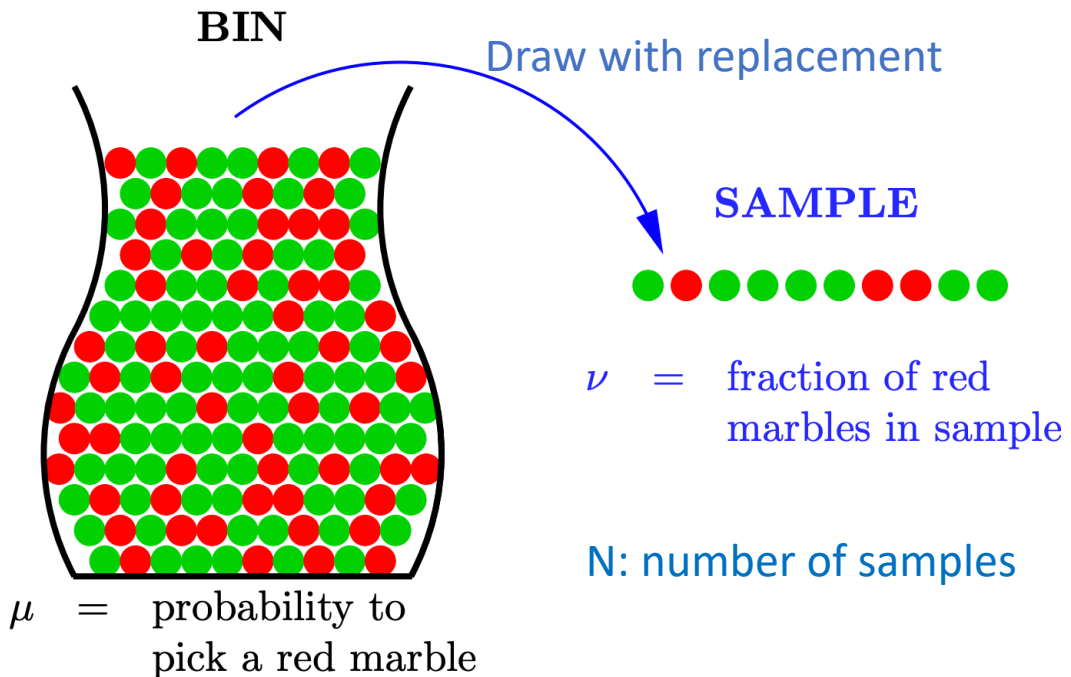
- Given **training data**, find  $g \approx f$  on the **unseen test data**.
- This goal is generally impossible without assumptions.

## Key assumption of ML

**Training** data points and **test** data points are **i.i.d.**  
drawn from the same (unknown) distribution



# A Thought Experiment about Probability



What can we say about  $\mu$  from  $\nu$ ?

Law of large numbers

- When  $N \rightarrow \infty$ ,  $\nu \rightarrow \mu$

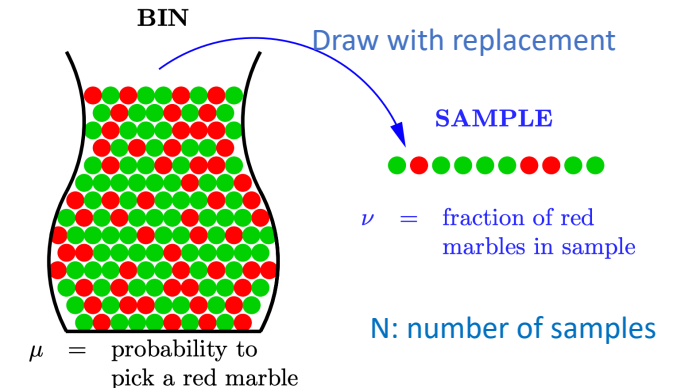
## Hoeffding's Inequality

- $\Pr[|\mu - \nu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$  for any  $\epsilon > 0$

# Connection to Learning

- Let each marble represent a point  $\vec{x}$ , drawn from unknown  $P(\vec{x})$ 
  - Dataset  $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$
  - Recall that  $y_n = f(\vec{x}_n)$  (will discuss noisy target function  $f$  later in the semester)

- “Fix” a hypothesis  $h$ 
  - For each marble  $\vec{x}$ , color it as below
    - If  $h(\vec{x}) = f(\vec{x})$ , color it as green marble [ $h$  is correct on  $\vec{x}$ ]
    - If  $h(\vec{x}) \neq f(\vec{x})$ , color it as red marble [ $h$  is wrong on  $\vec{x}$ ]



- With the above coloring

$$\mu = \Pr_{\vec{x} \sim P(\vec{x})} [h(\vec{x}) \neq f(\vec{x})]$$

$$\stackrel{\text{def}}{=} E_{out}(h) \quad \text{[Out-of-sample error of } h]$$

$$\nu = \frac{1}{N} \sum_{n=1}^N \mathbb{I}[h(\vec{x}_n) \neq f(\vec{x}_n)]$$

$$\stackrel{\text{def}}{=} E_{in}(h) \quad \text{[in-sample error of } h]$$



# Connection to Learning

- $E_{out}(h)$ : What we really want to know but unknown to us
- $E_{in}(h)$ : What we can calculate from dataset
- Fixed a  $h$ , What can we say about  $E_{out}(h)$  from  $E_{in}(h)$ ?

## Hoeffding's Inequality

$$\Pr[|E_{out}(h) - E_{in}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0$$

- This is verification, not learning!

# Verification vs. Learning

- Verification
  - I have a hypothesis  $h$
  - I know  $E_{in}(h)$ , i.e., how well  $h$  performs in my dataset
  - I can infer what  $E_{out}(h)$  (how well  $h$  will perform for unseen data) might be
- Learning
  - Given a dataset  $D$  and hypothesis set  $H$
  - Apply some learning algorithm, that outputs a  $g \in H$
  - Know  $E_{in}(g)$
  - Want to infer  $E_{out}(g)$

# Connection to “Real” Learning

- Given a **finite** hypothesis set  $H = \{h_1, \dots, h_M\}$ 
  - Will discuss the infinite case in the next few lectures.
- Apply some learning algorithm on  $D$ , output a  $g \in H$ 
  - For example, choosing the hypothesis that minimizes in-sample error
    - $g = \operatorname{argmin}_{h \in H} E_{in}(h)$
- Can we apply Hoeffding’s inequality and claim
$$\Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2e^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0$$
- **No!**

# Today's Lecture

The notes are not intended to be comprehensive.  
Let me know if you spot errors.

# An Analogy

- Three fair coins, numbered by 1, 2, 3.
  - Flip each coin 10 times

- Question: (choosing from  $>5$ ,  $=5$ , or  $<5$ )

**Ans:  $=5$**  • For coin 1, what's the expected number of heads among 10 flips?

**Ans:  $=5$**  • Randomly choose a coin, what's the expected number of heads for this coin?

**Ans:  $>5$**  • Look at the realized flips and choose the coin with the largest number of heads. What is the expected number of heads (on the already flipped results) for the coin?

**Ans:  $=5$**  • Without observing the flips, choose the coin anyway you like, what is the expected number of heads of the 10 flips for this coin?

- You will simulate this process (with 1,000 coins) in HW1.

# An Analogy

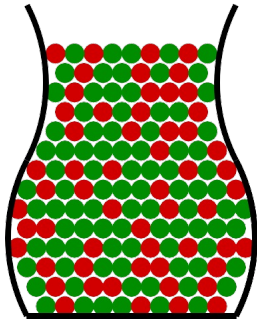
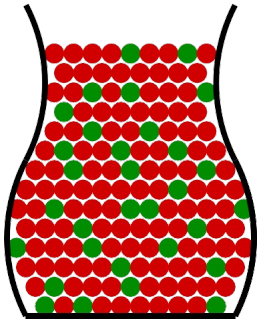
- Connects to learning
  - Coin  $\rightarrow$  Hypothesis
  - Coin flips  $\rightarrow$  Performance of hypothesis in training data  $D$
- Choosing the hypothesis “before” or “after” looking at the data (knowing the realization of the data drawing) makes a big difference!

# An Analogy

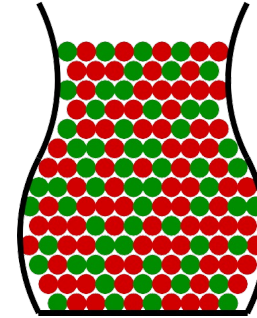
## Hoeffding's Inequality

- $\Pr[|\mu - \nu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$  for any  $\epsilon > 0$

- Some graphical explanations

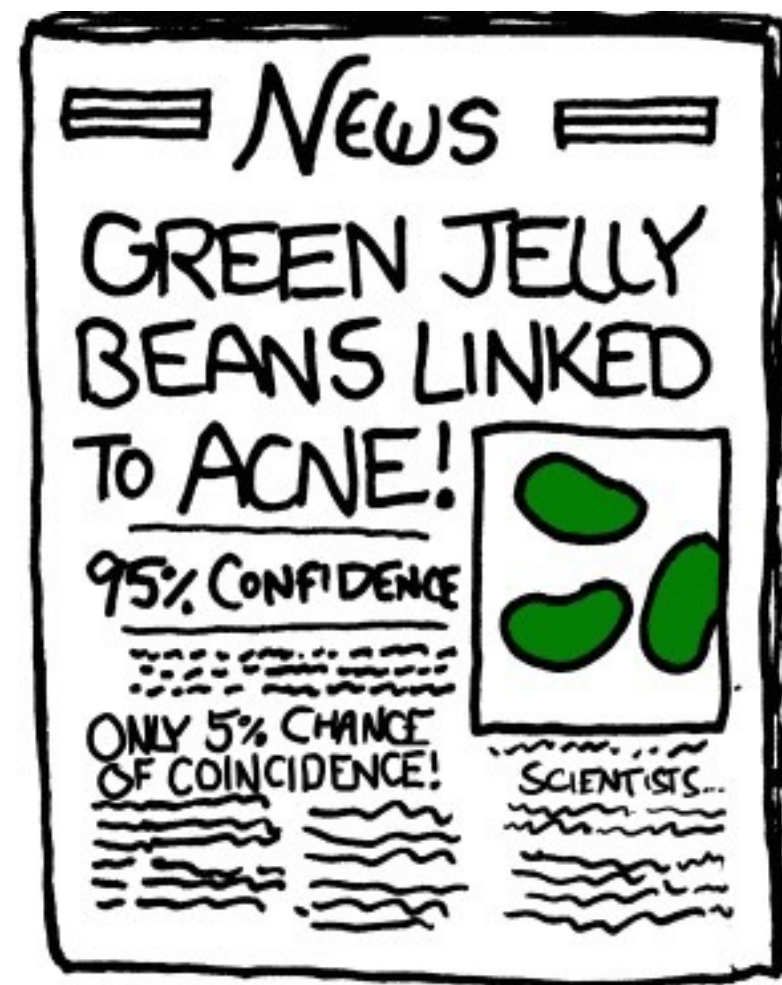
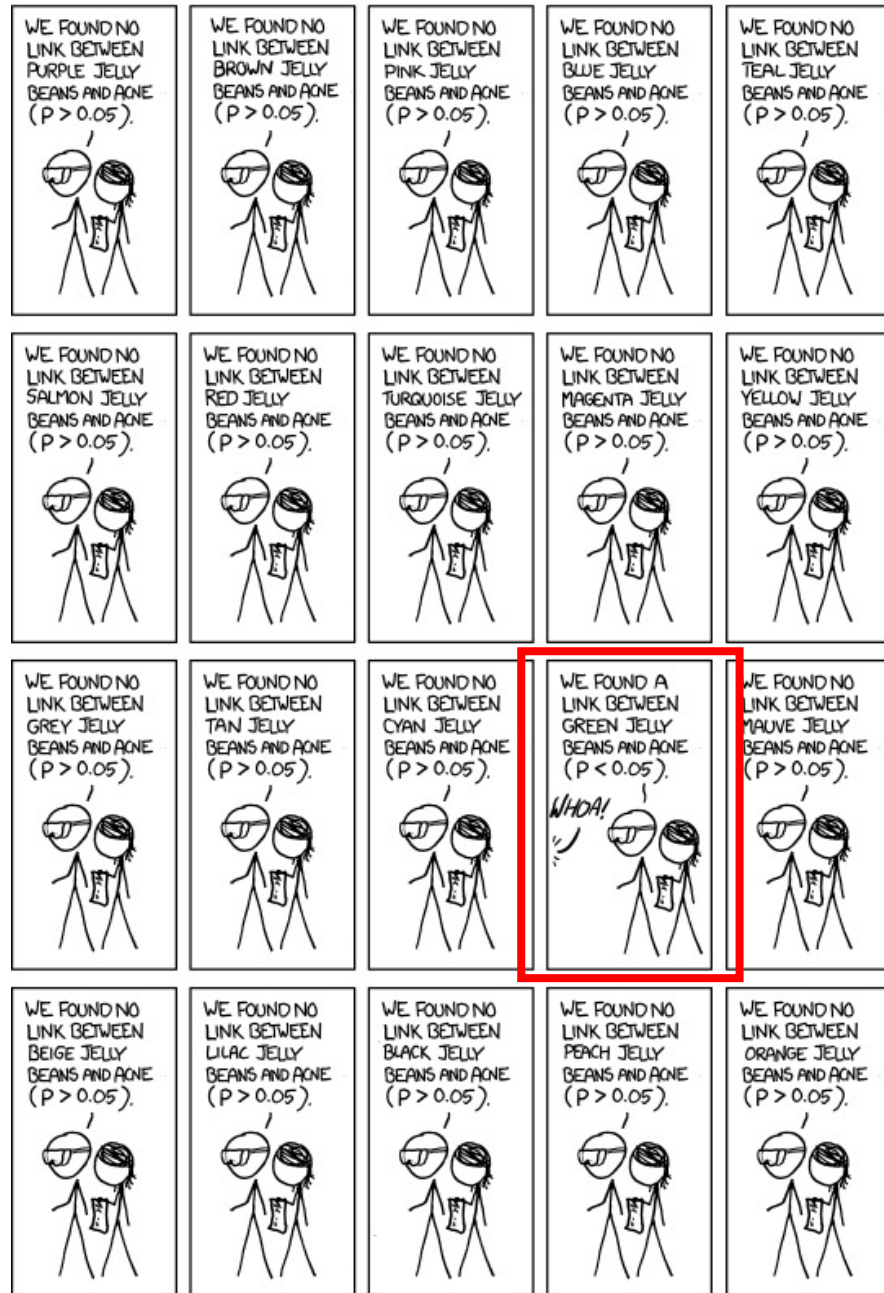


...









What Can We Do?

# Connection to “Real” Learning

- Given a **finite** hypothesis set  $H = \{h_1, \dots, h_M\}$
- Apply some learning algorithm on  $D$ , output a  $g \in H$ 
  - For example, choosing the hypothesis that minimizes in-sample error
    - $g = \operatorname{argmin}_{h \in H} E_{in}(h)$
- Question: What can we say about  $E_{out}(g)$  from  $E_{in}(g)$ ?

# Derivations

- Define “bad event of  $h$ ”  $B(h)$  as  $|E_{out}(h) - E_{in}(h)| > \epsilon$ 
  - Informally, you can interpret “bad event of  $h$ ” as the event that we draw a “unrepresentative dataset  $D$ ” that makes the in-sample errors of  $h$  to be far away from out-of-sample error of  $h$

For each fixed  $h \in H$ , we have  $\Pr[B(h)] \leq 2e^{-2\epsilon^2 N}$

- Recall  $g$  is selected from  $H$  (it could be any  $h \in H$ )
- What can we say about  $\Pr[B(g)]$ ?

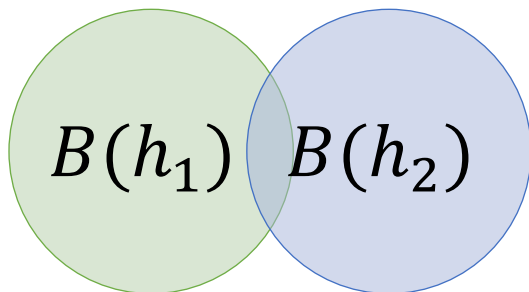
# Derivations

- Define “bad event of  $h$ ”  $B(h)$  as  $|E_{out}(h) - E_{in}(h)| > \epsilon$ 
  - Informally, you can interpret “bad event of  $h$ ” as the event that we draw a “unrepresentative dataset  $D$ ” that makes the in-sample errors of  $h$  to be far away from out-of-sample error of  $h$

For each fixed  $h \in H$ , we have  $\Pr[B(h)] \leq 2e^{-2\epsilon^2 N}$

- Recall  $g$  is selected from  $H$  (it could be any  $h \in H$ )
- What can we say about  $\Pr[B(g)]$ ?

If  $g$  is selected from  $\{h_1, h_2\}$



$$B(g) \subseteq B(h_1) \cup B(h_2)$$

$$\begin{aligned}\Pr[B(g)] &\leq \Pr[B(h_1) \text{ or } B(h_2)] \\ &\leq \Pr[B(h_1)] + \Pr[B(h_2)]\end{aligned}$$

(Union Bound)

# Derivations

- Define “bad event of  $h$ ”  $B(h)$  as  $|E_{out}(h) - E_{in}(h)| > \epsilon$ 
  - Informally, you can interpret “bad event of  $h$ ” as the event that we draw a “unrepresentative dataset  $D$ ” that makes the in-sample errors of  $h$  to be far away from out-of-sample error of  $h$

For each fixed  $h \in H$ , we have  $\Pr[B(h)] \leq 2e^{-2\epsilon^2 N}$

- Recall  $g$  is selected from  $H$  (it could be any  $h \in H$ )
- What can we say about  $\Pr[B(g)]$ ?

$$\begin{aligned}\Pr[B(g)] &\leq \Pr[B(h_1) \text{ or } B(h_2) \text{ or } \dots \text{ or } B(h_M)] \\ &\leq \Pr[B(h_1)] + \Pr[B(h_2)] + \dots + \Pr[B(h_M)] \\ &\leq M 2e^{-2\epsilon^2 N}\end{aligned}$$

# Connection to “Real” Learning

- Given a **finite** hypothesis set  $H = \{h_1, \dots, h_M\}$
- Apply some learning algorithm on  $D$ , output a  $g \in H$
- Question: What can we say about  $E_{out}(g)$  from  $E_{in}(g)$ ?

$$\Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2\mathbf{M}e^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0$$

- $\mathbf{M}$  can be considered as a proxy of the “complexity” of the hypothesis set
  - Will talk about what happens when  $\mathbf{M} \rightarrow \infty$  in the next few lectures

Interpreting  $\Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$



Interpreting  $\Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$

- Playing around with the math
  - Define  $\delta = \Pr[|E_{out}(g) - E_{in}(g)| > \epsilon]$
  - We have  $\delta \leq 2Me^{-2\epsilon^2 N} \Rightarrow \epsilon \leq \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$

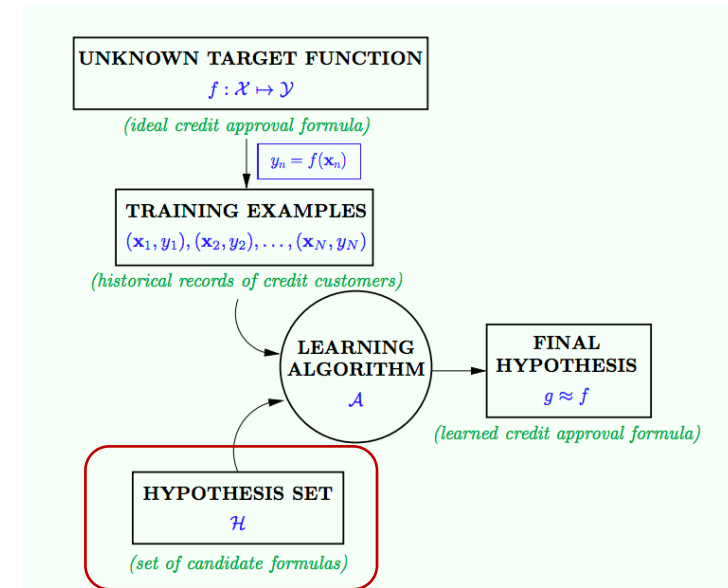
Interpreting  $\Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$

- Playing around with the math
  - Define  $\delta = \Pr[|E_{out}(g) - E_{in}(g)| > \epsilon]$
  - We have  $\delta \leq 2Me^{-2\epsilon^2 N} \Rightarrow \epsilon \leq \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$
- This means, with probability  $1 - \delta$ 
  - $E_{out}(g) \leq E_{in}(g) + \epsilon \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$

# More Discussion

- With probability  $1 - \delta$

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$$



Consider  $M$  as a proxy measure on the “complexity” of  $H$

- Our ultimate goal is to have a small  $E_{out}(g)$ 
  - There is a tradeoff of choosing  $M$  (what “learning model” to use)
    - Increase  $M$  -> Smaller  $E_{in}(g)$  (more hypothesis to “fit” the training data)
    - Increase  $M$  -> Larger  $\epsilon$
  - It also depends on  $N$ , the number of data points you have
    - A small number of data points => use simple models (e.g., linear models)
    - Complex models (e.g., deep learning) work when you have a lot of data

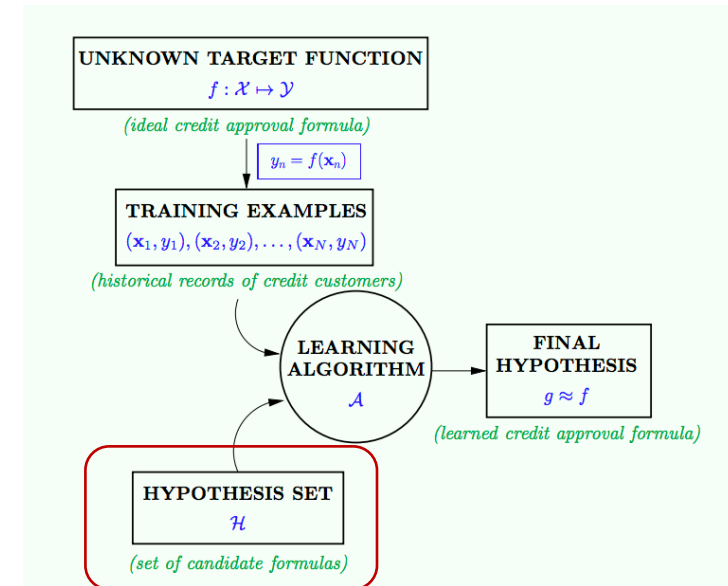
# More Discussion

- With probability  $1 - \delta$

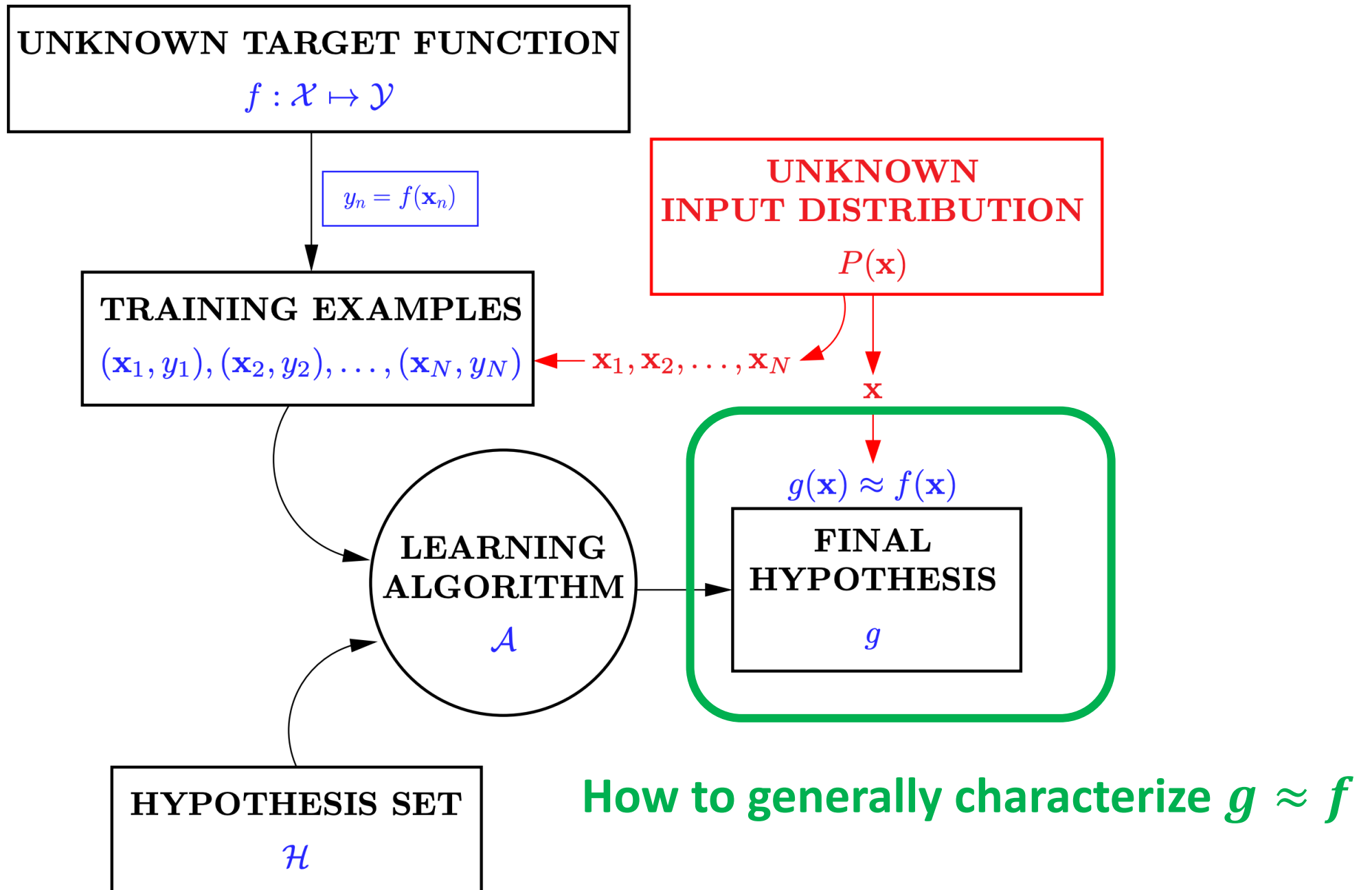
$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$$

Consider  $M$  as a proxy measure on the “complexity” of  $H$

- Learning curves



# Revisit the Learning Problem



Goal:  $g \approx f$

- A general approach:
  - Define an error function  $E(h, f)$  that quantify how far away  $h$  is to  $f$
  - choose  $g = \operatorname{argmin}_{h \in \mathcal{H}} E(h, f)$
- $E$  is usually defined in terms of a **pointwise** error function  $e(h(\vec{x}), f(\vec{x}))$ 
  - Binary error (classification):  $e(h(\vec{x}), f(\vec{x})) = \mathbb{I}[h(\vec{x}_n) \neq f(\vec{x}_n)]$
  - Squared error (regression):  $e(h(\vec{x}), f(\vec{x})) = (f(\vec{x}) - h(\vec{x}))^2$

$$E_{in}(h) = \frac{1}{N} \sum_{n=1}^N e(h(\vec{x}_n), f(\vec{x}_n))$$
$$E_{out}(h) = \mathbb{E}_{\vec{x}}[e(h(\vec{x}), f(\vec{x}))]$$

The discussion on the Hoeffding's inequality applies for general (bounded) error functions.

# How to choose the error function?

- Consideration 1: Properties of domain applications
- Example: Fingerprint recognition
  - Input: fingerprints
  - Outputs: whether the person is authorized

		$f(\vec{x})$	
		+1	-1
$h(\vec{x})$	+1	No error	False positive
	-1	False negative	No error

		$f(\vec{x})$	
		+1	-1
$h(\vec{x})$	+1	0	Small
	-1	Large	0

		$f(\vec{x})$	
		+1	-1
$h(\vec{x})$	+1	0	Large
	-1	Small	0



# How to choose the error function?

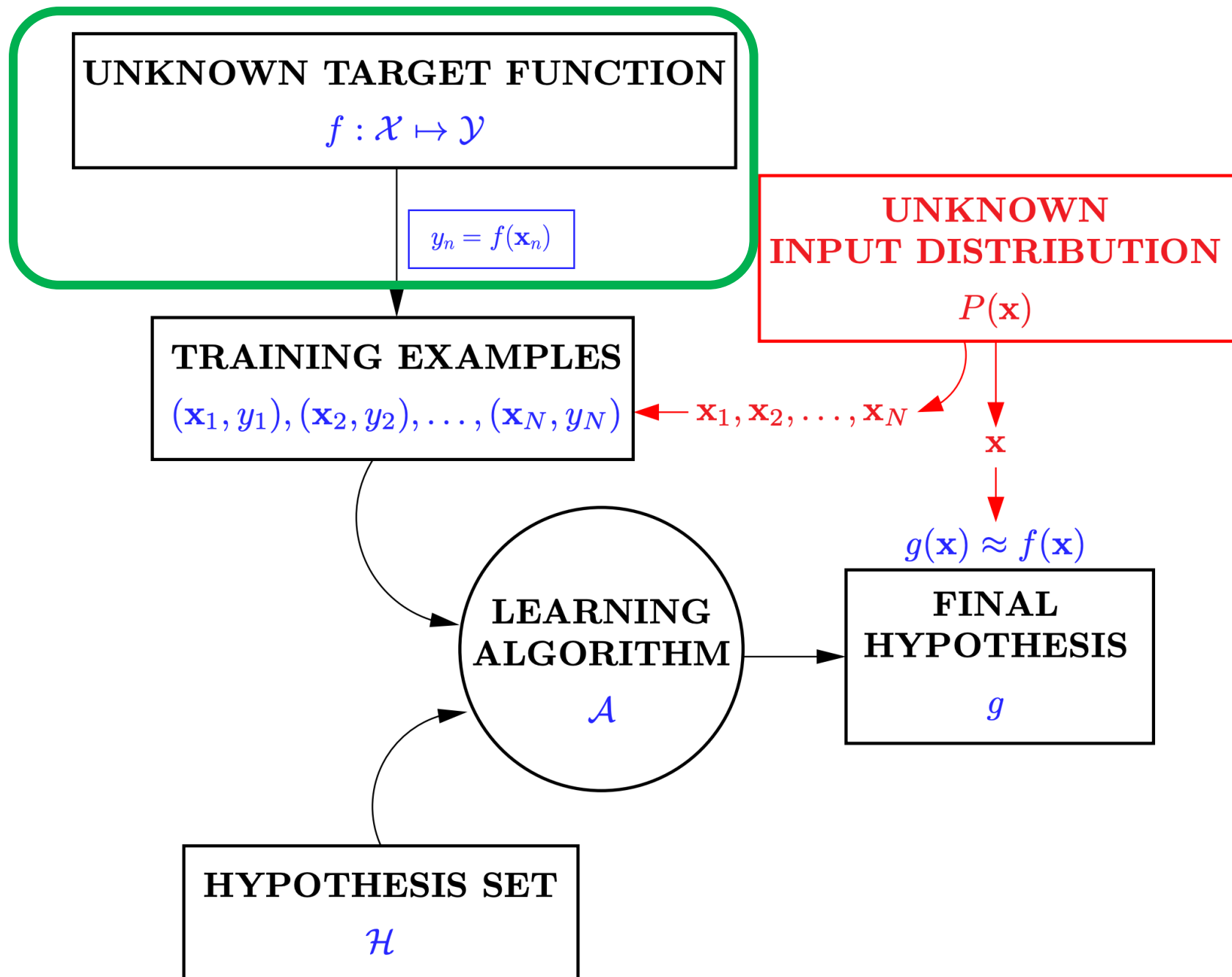
- Consideration 1: Properties of application problems
- Consideration 2: Computation
  - ML Algorithm is essentially doing **optimization** (finding  $g$  with smallest error)

$$g = \operatorname{argmin}_{h \in \mathcal{H}} E(h, f)$$

- Choosing the error that is “easier” to optimize
  - e.g., if the error function is convex, continuous, differentiable, we usually have efficient algorithms

# How to choose the error function?

- Consideration 1: Properties of application problems
- Consideration 2: Computation
- Specifying the error function is part of setting up the learning problem
  - It impacts what you eventually learn

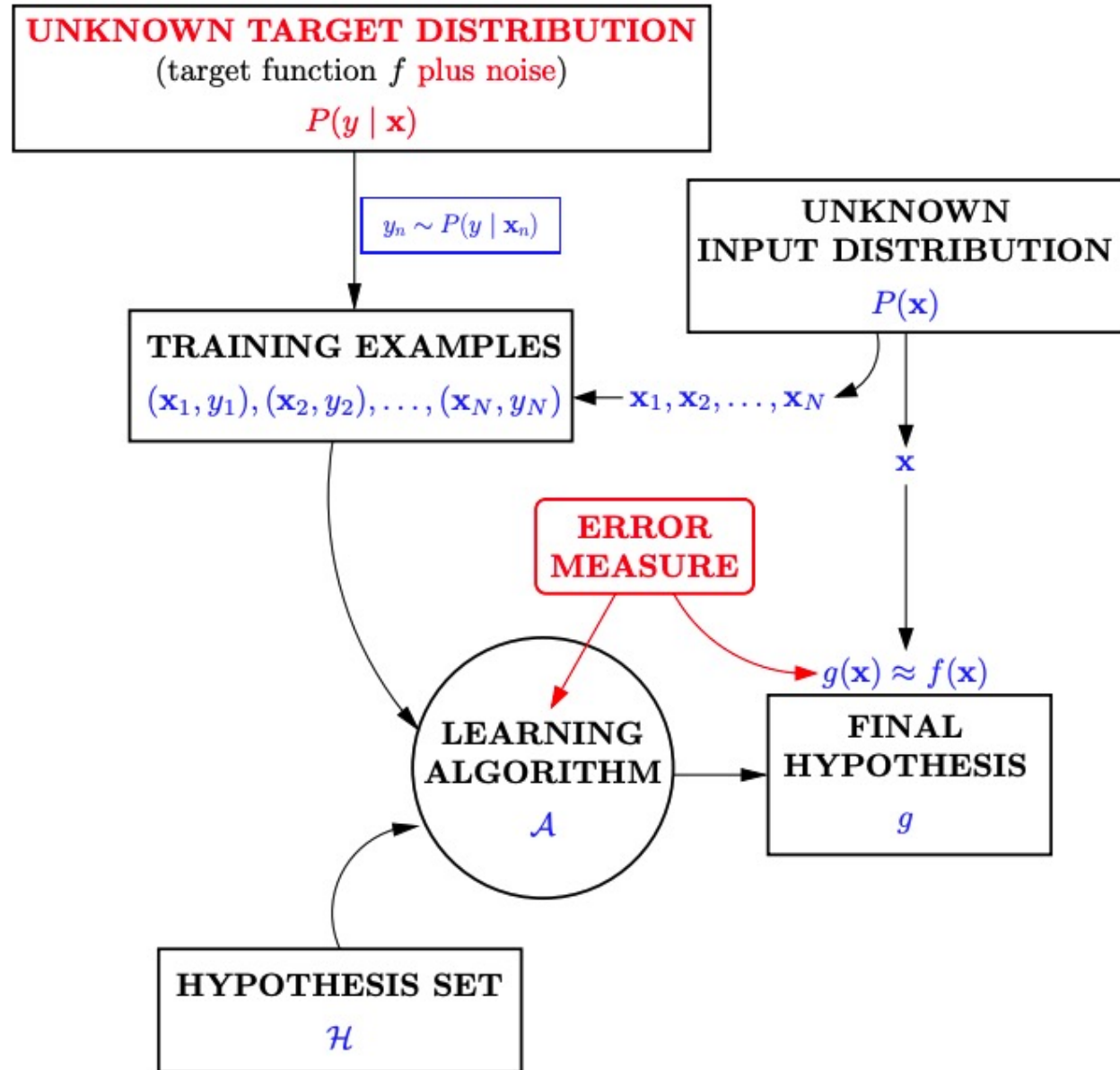


# Noisy Target

- What if there doesn't exist  $f$  such that  $y = f(\vec{x})$ ?
  - $f$  is stochastic instead of deterministic
- Common approach
  - Instead of a target function, define a target **distribution**
  - Instead of  $y = f(\vec{x})$ ,  $y$  is drawn from a conditional distribution  $P(y|\vec{x})$
  - $y = f(\vec{x}) + \epsilon$  where  $\epsilon$  is zero-mean noise

The discussion on the Hoeffding's inequality applies for noisy targets.

# General Setup of (Supervised) Learning



# Theory of Generalization

# Revisit the “Multi-Hypothesis” Bound

- Given a **finite** hypothesis set  $H = \{h_1, \dots, h_M\}$
- Apply some learning algorithm on  $D$ , output a  $g \in H$
- What can we say about  $E_{out}(g)$  from  $E_{in}(g)$ ?

$$\Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N} \quad \text{for any } \epsilon > 0$$

What if  $M$  is infinite?

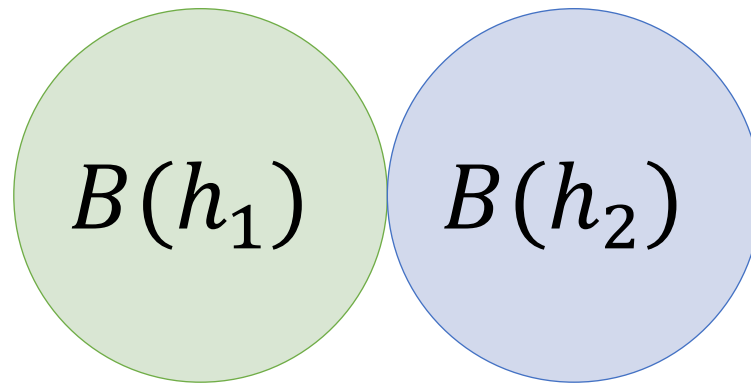
$Pr[|E_{out}(g) - E_{in}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2 N}$  don't seem to carry any meanings



# Key Intuitions in the Multi-Hypothesis Analysis

- Define "bad event of  $h$ "  $B(h)$  as  $|E_{out}(h) - E_{in}(h)| > \epsilon$
- If  $g$  is selected from  $\{h_1, h_2\}$ 
  - $B(g) \subseteq B(h_1) \cup B(h_2)$
  - $\Pr[B(g)] \leq \Pr[B(h_1) \text{ or } B(h_2)]$

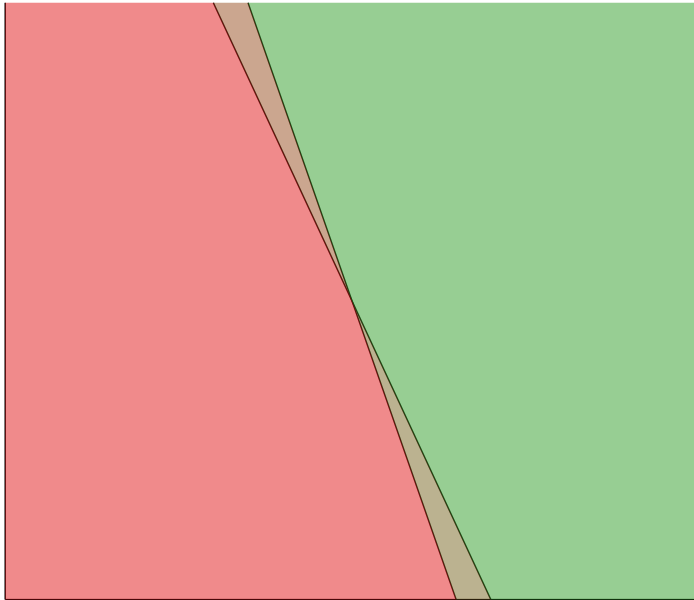
$$\leq \Pr[B(h_1)] + \Pr[B(h_2)] \quad (\text{Union Bound})$$



- Union bound considers the **worst case: Bad events don't overlap**

# Do Bad Events Overlap?

- Oftentimes, they overlap a lot!



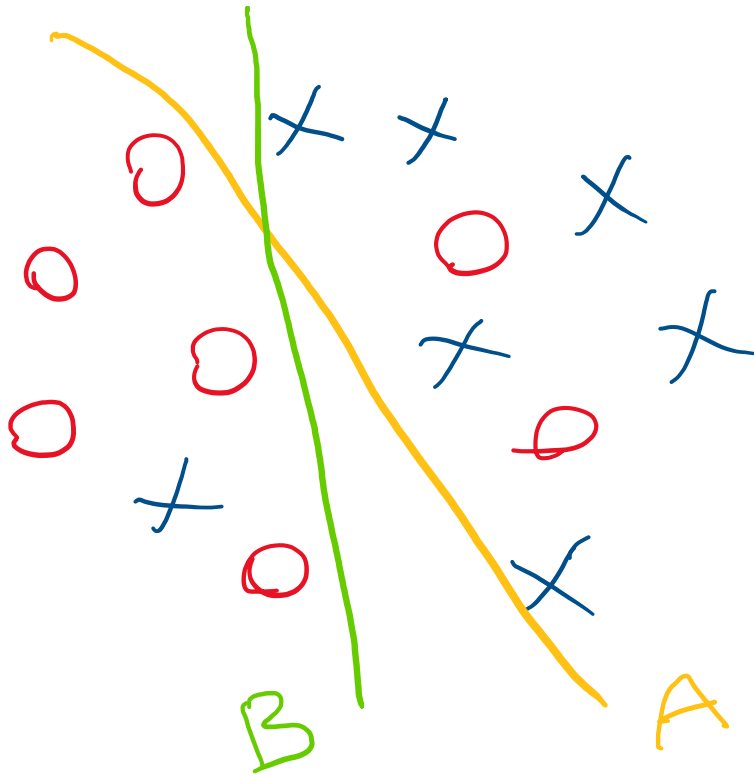
The two linear separators on the left make the same predictions for most points.

If it's a bad event for one, it's likely to be a bad event for the other.

$$\text{"bad event of } h\text{" } B(h): |E_{out}(h) - E_{in}(h)| > \epsilon$$

Recall: Informally, you can interpret “bad event of  $h$ ” as the event that we draw a “unrepresentative dataset  $D$ ” that makes the in-sample errors of  $h$  to be far away from out-of-sample error of  $h$

# What Can We Do?




Any difference between **A** and **B**?

For this dataset, probably not.

They make the same predictions for every data point in this dataset.

# What Can We Do?

- Let's define “data-dependent” hypothesis, call it **dichotomy**.

 di·chot·o·my  
*/dī'kädəmə/*  
*noun*  
a division or contrast between two things that are or are represented as being opposed or entirely different.  
"a rigid **dichotomy** between science and mysticism"

- A hypothesis  $h: X \rightarrow \{-1, +1\}$
- A dichotomy for a set of data points  $(\vec{x}_1, \dots, \vec{x}_N)$ :
  - Assign either **+1** or **-1** for each of the data points  
(divide the data points into two groups)
- Why dichotomies?
  - It helps us count “effective number of hypothesis” (to replace  $M$ )

# More Formal Definitions

- Dichotomies

- Informally, consider a dichotomy as “data-dependent” hypothesis
- Characterized by both hypothesis set  $H$  and  $N$  data points  $(\vec{x}_1, \dots, \vec{x}_N)$

$$H(\vec{x}_1, \dots, \vec{x}_N) = \{h(\vec{x}_1), \dots, h(\vec{x}_N) | h \in H\}$$

- The set of possible prediction combinations  $h \in H$  can induce on  $\vec{x}_1, \dots, \vec{x}_N$

- Growth function

- Largest number of dichotomies  $H$  can induce across all possible data sets of size  $N$

$$m_H(N) = \max_{(\vec{x}_1, \dots, \vec{x}_N)} |H(\vec{x}_1, \dots, \vec{x}_N)|$$