CSE 417T
Introduction to Machine Learning

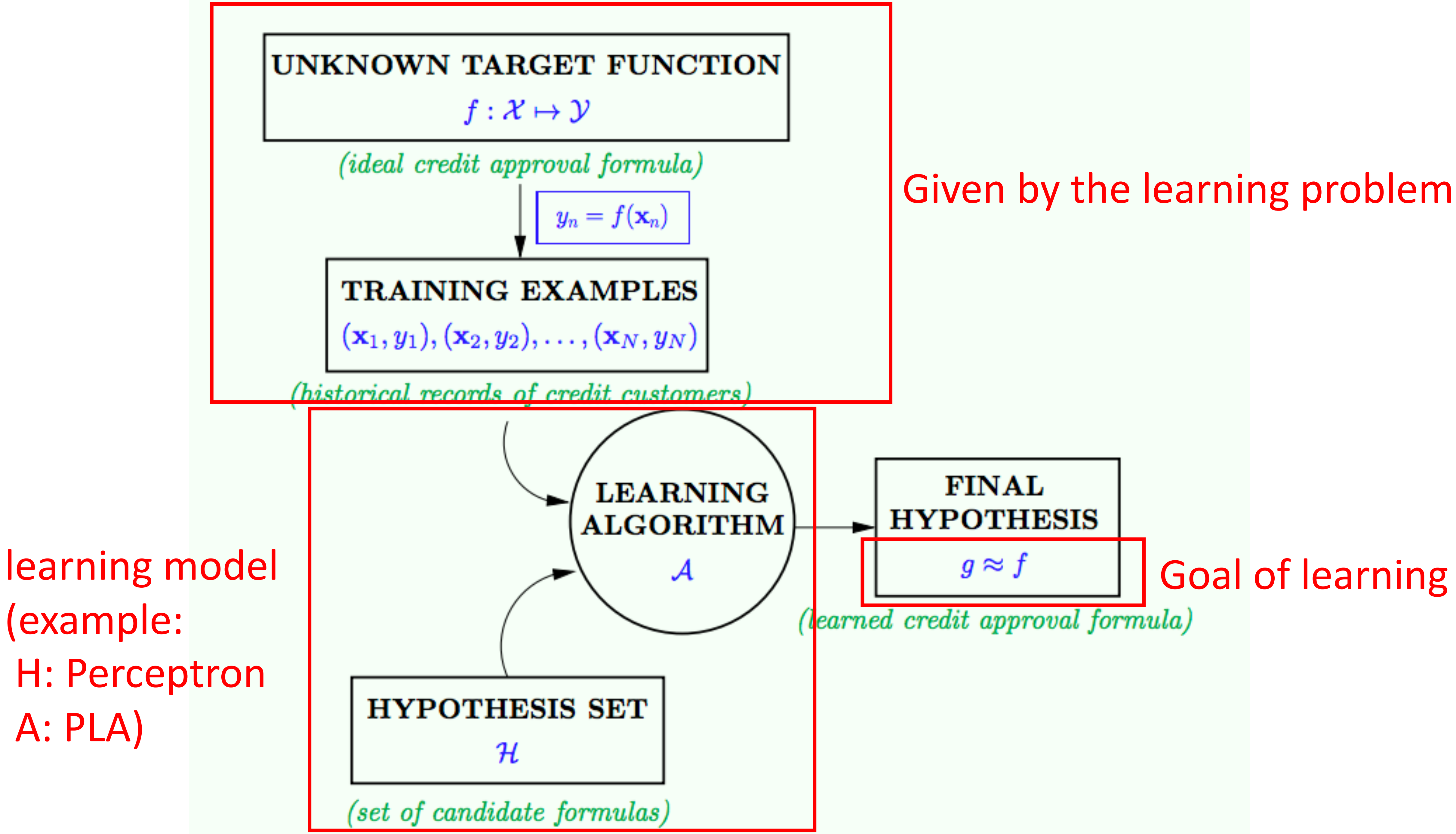
Lecture 3

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
$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
(historical records of credit customers)
Given by the learning problem
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g \approx f$
Goal of learning
(learned credit approval formula)
HYPOTHESIS SET
$\mathcal{H}$
(set of candidate formulas)
learning model
(example:
H: Perceptron
A: PLA)

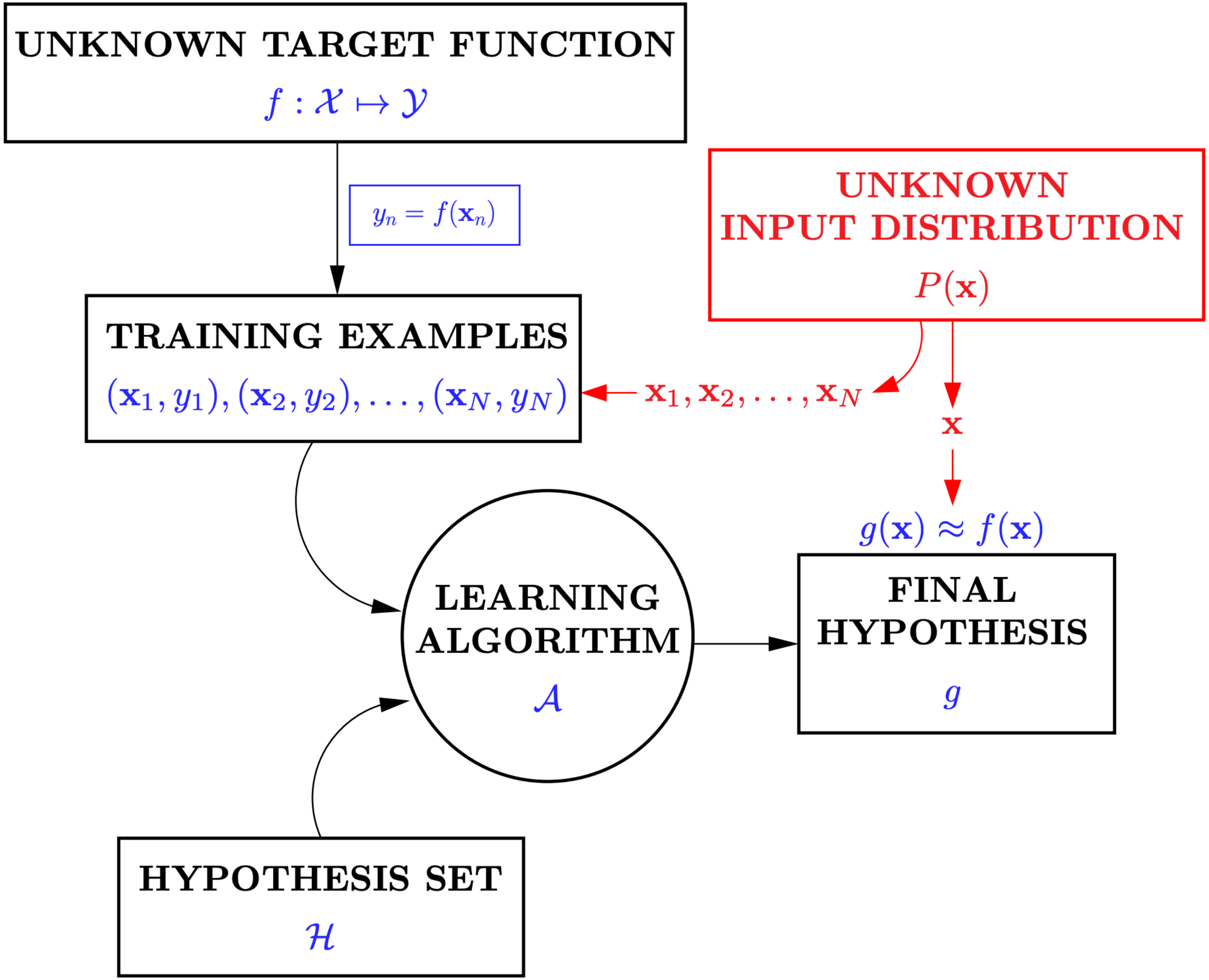
Goal of Learning: Generalization

- Given **training data**, find $g \approx f$ on the **unseen test data**.

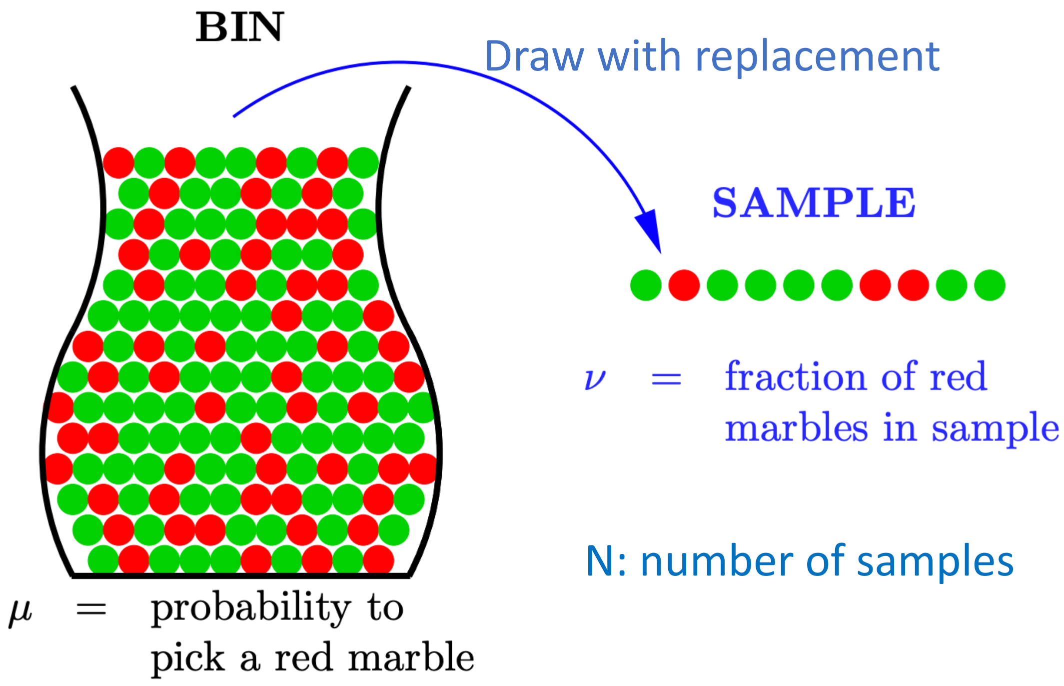
- This goal is generally impossible without assumptions.

Key assumption of ML

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


UNKNOWN TARGET FUNCTION
$f: \mathcal{X} \mapsto \mathcal{Y}$
$y_n = f(\mathbf{x}_n)$
UNKNOWN
INPUT DISTRIBUTION
$P(\mathbf{x})$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$
$\mathbf{x}$
$g(\mathbf{x}) \approx f(\mathbf{x})$
LEARNING
ALGORITHM
$\mathcal{A}$
FINAL
HYPOTHESIS
g
HYPOTHESIS SET
$\mathcal{H}$

A Thought Experiment about Probability



What can we say about μ from ν?

Law of large numbers

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

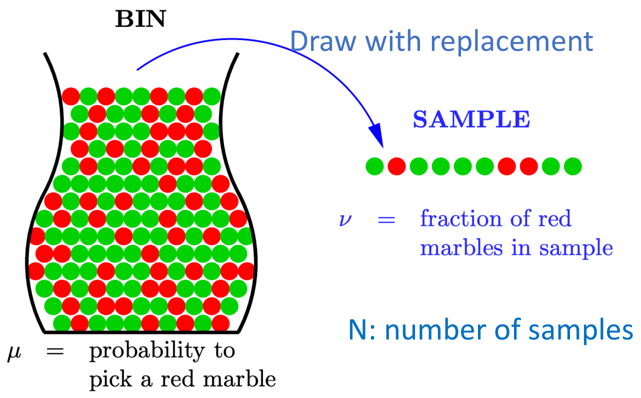
Hoeffding's Inequality

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

$$\mu = \Pr_{\vec{x} \sim P(\vec{x})}[h(\vec{x}) \neq f(\vec{x})] \stackrel{\text{def}}{=} E_{out}(h)$$ **[Out-of-sample error of h]**

$$\nu = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}[h(\vec{x}_n) \neq f(\vec{x}_n)] \stackrel{\text{def}}{=} E_{in}(h)$$ **[in-sample error of h]**

Connection to Learning

- $\boldsymbol{E_{out}(h)}$: What we really want to know but unknown to us
- $\boldsymbol{E_{in}(h)}$: What we can calculate from dataset

- Fixed a h, What can we say about $E_{out}(h)$ from $E_{in}(h)$?

<u>Hoeffding's Inequality</u>

$\Pr[|E_{out}(h) - E_{in}(h)| > \epsilon] \le 2e^{-2\epsilon^2 N}$ 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 $g \in \mathrm{H}$
  - 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
    - $g = argmin_{h \in H} E_{in}(h)$

- 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 -> Hypothesis
  - Coin flips -> Performance of hypothesis in training data D

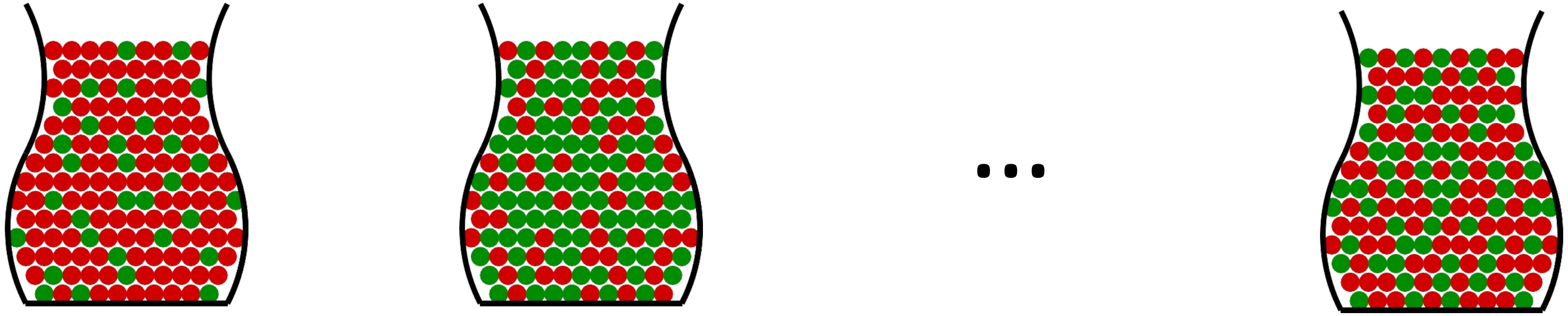
- Choosing the hypothesis "before" or "after" looking at the data (knowing the realization of the data drawing) makes a big difference!

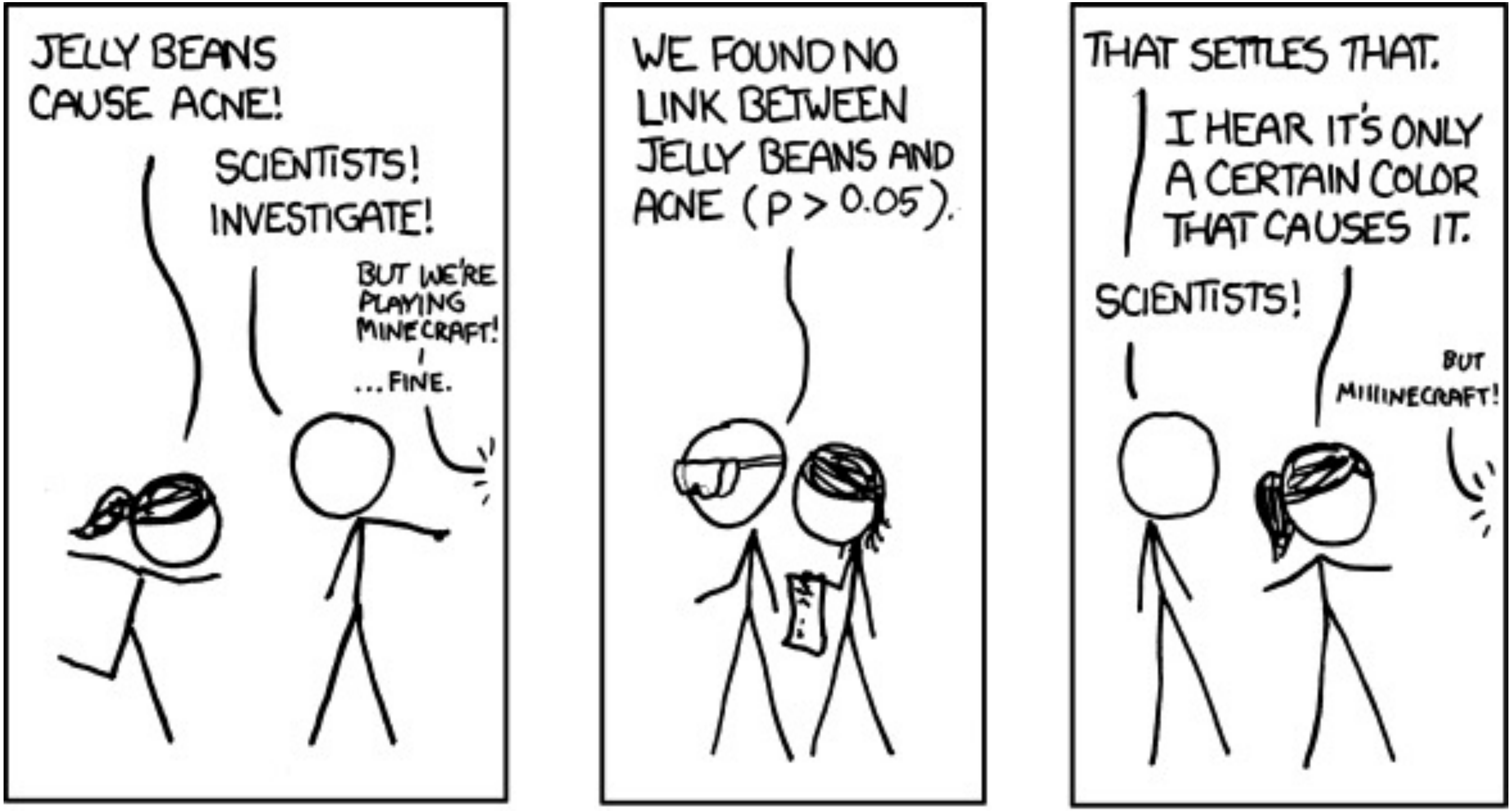
An Analogy

<u>Hoeffding's Inequality</u>

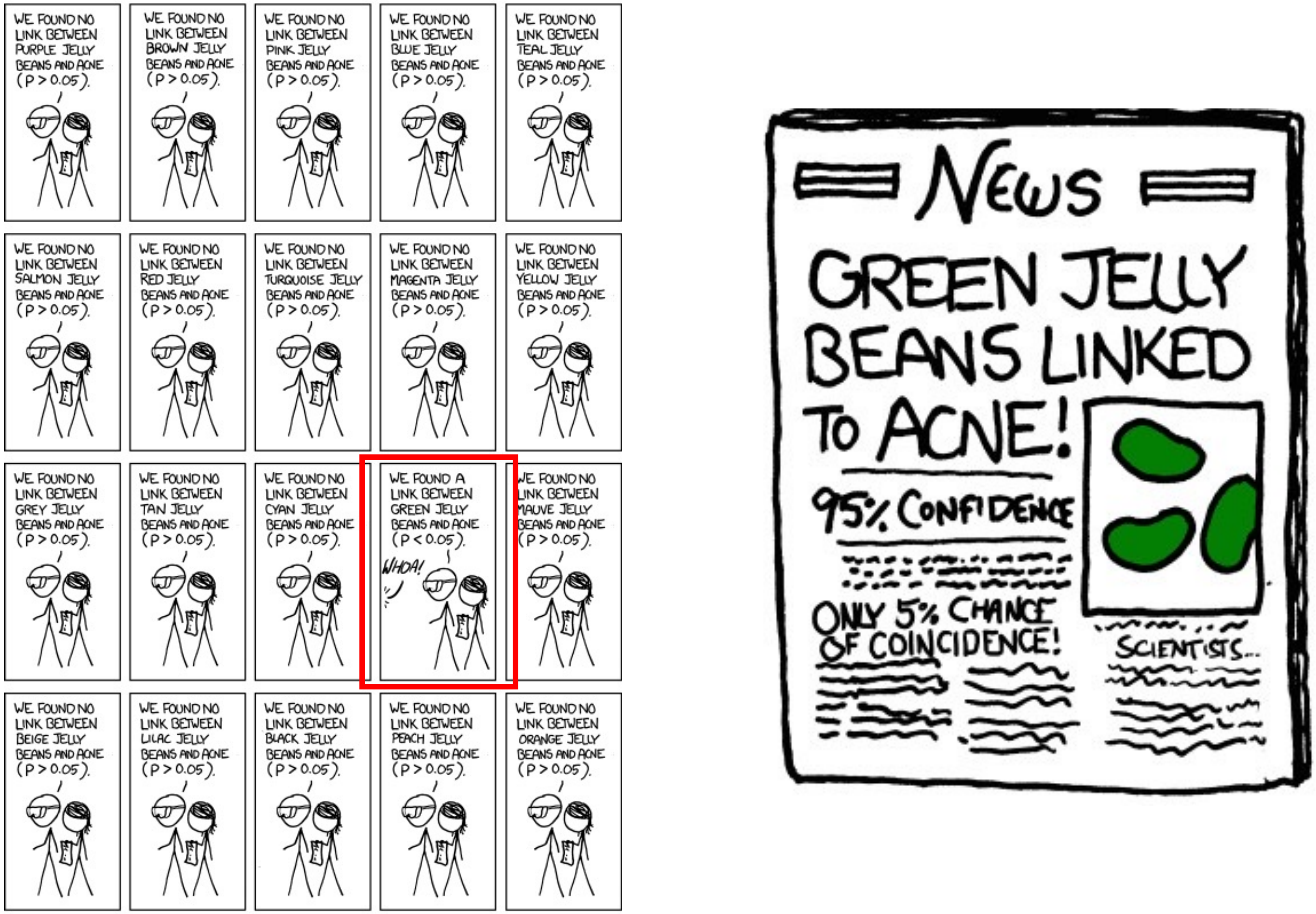
- $\mathbf{Pr}[|\mu - \nu| > \epsilon] \leq 2e^{-2\epsilon^2 N}$ for any $\epsilon > 0$

- Some graphical explanations





From xkcd, by Randall Munroe: http://xkcd.com/882



From xkcd, by Randall Munroe: http://xkcd.com/882

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
    - $g = argmin_{h \in H} E_{in}(h)$

- 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 $\Pr[B(g)]$?

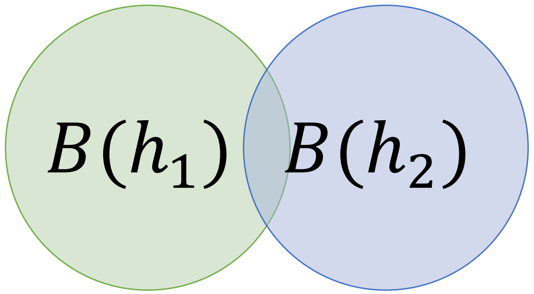
Derivations

- 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 $\{h_1, h_2\}$



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

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

(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 $\Pr[B(g)]$?

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

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 $E_{in}(g)$?

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

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

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

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

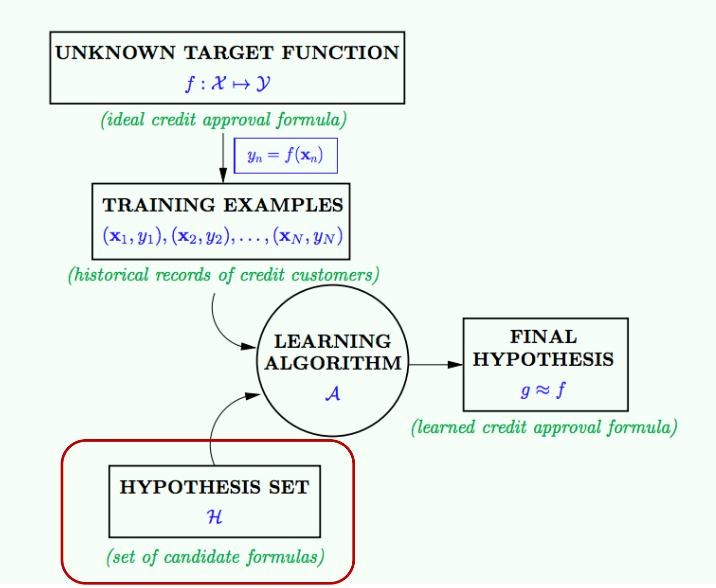
- Playing around with the math
  - Define $\delta = \Pr[|E_{out}(g) - E_{in}(g)| > \epsilon]$
  - We have $\delta \le 2Me^{-2\epsilon^2 N}$ => $\epsilon \le \sqrt{\frac{1}{2N}\ln\frac{2M}{\delta}}$

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 $\delta \leq 2Me^{-2\epsilon^2 N}$ => $\epsilon \leq \sqrt{\frac{1}{2N}\ln\frac{2M}{\delta}}$

- 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



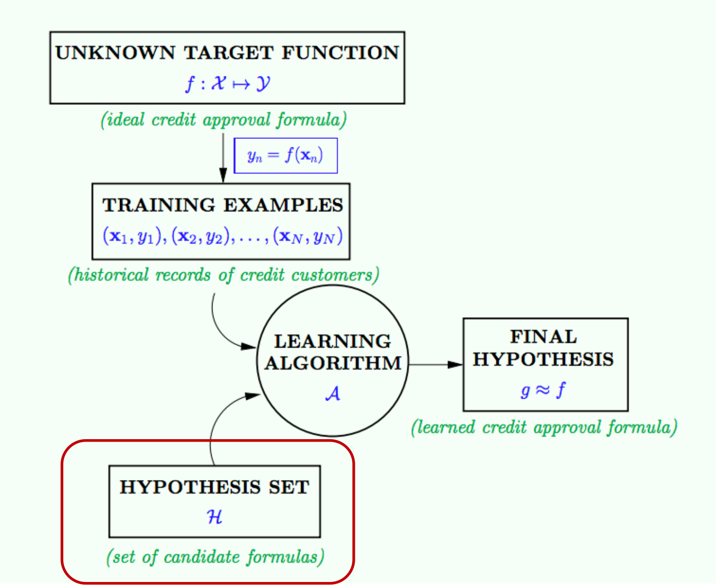
- With probability $1 - \delta$

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

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 ϵ
  - 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



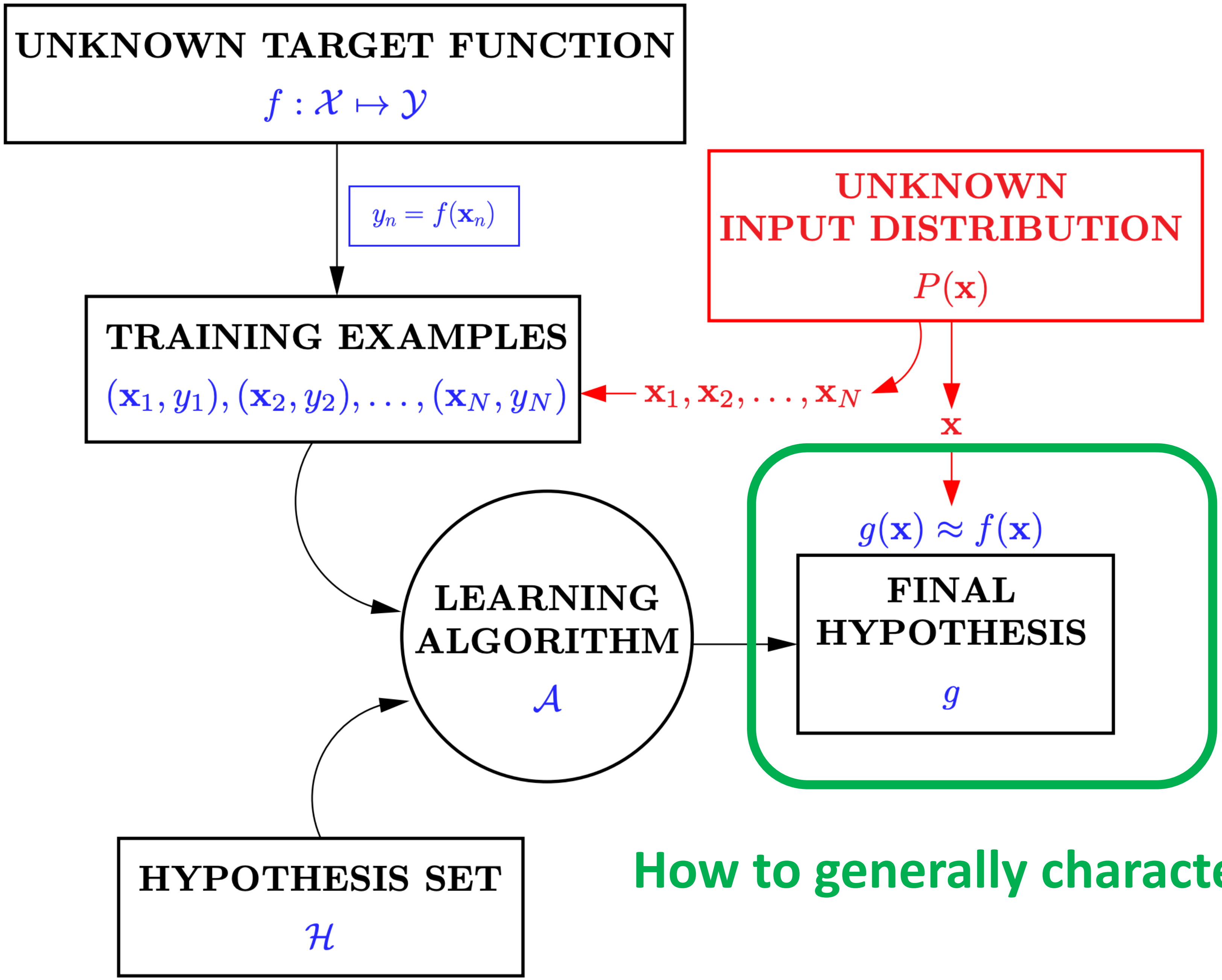
- With probability $1 - \delta$

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

Consider M as a proxy measure on the "complexity" of H

- Learning curves

Revisit the Learning Problem



How to generally characterize $g \approx f$

Goal: $g \approx f$

- A general approach:
  - Define an error function $E(h, f)$ that quantify how far away h is to f
  - choose $g = \underset{h \in \mathcal{H}}{\operatorname{argmin}} E(h, f)$
- 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): $e(h(\vec{x}), f(\vec{x})) = \big(f(\vec{x}) - h(\vec{x})\big)^2$

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

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

Supermarket

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

FBI

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