CS 60050
Machine Learning

Feasibility of Learning

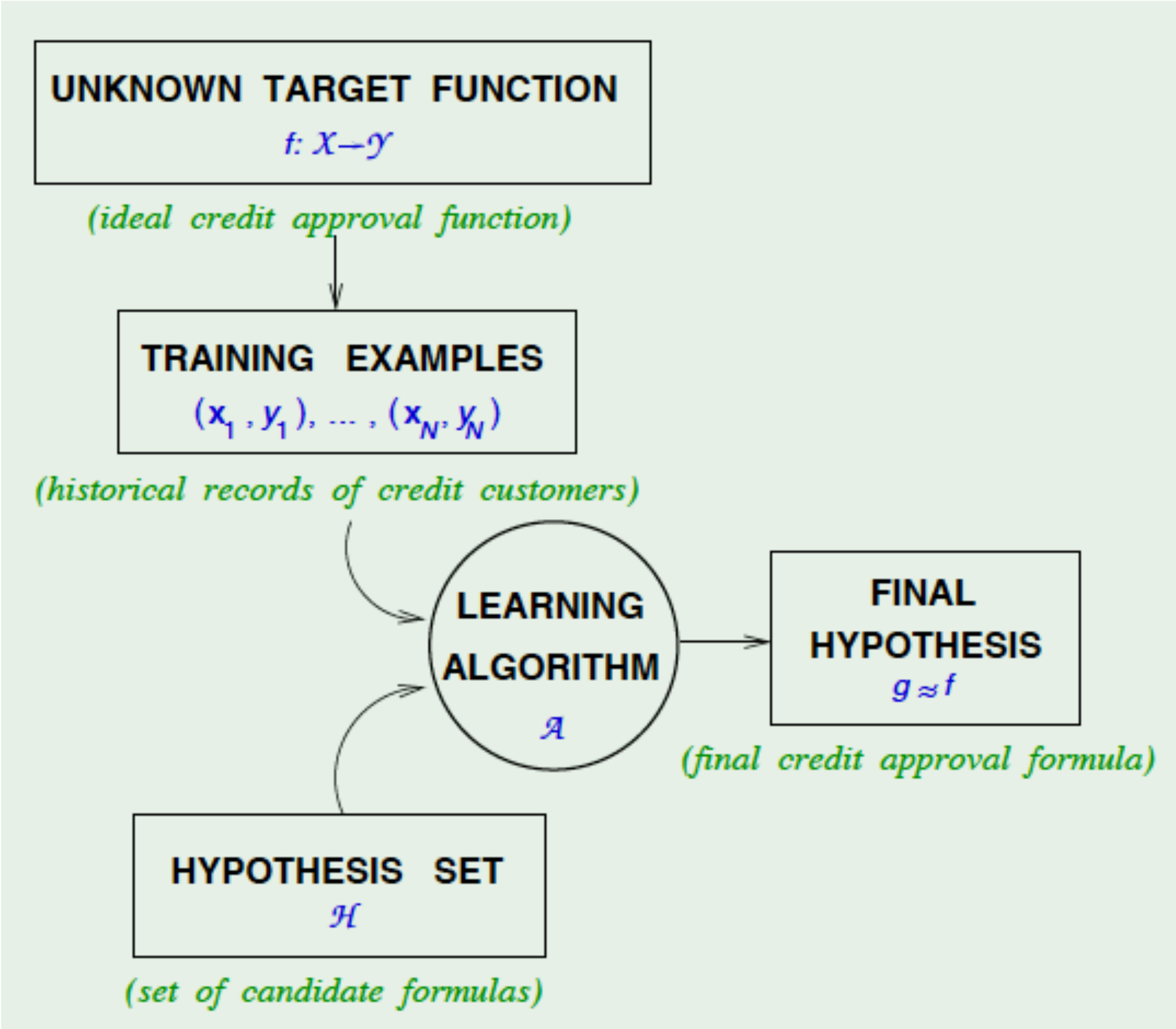
When can learning be used?

- A pattern exists
- The pattern cannot be pinned down mathematically
- There is data about the application

What are we learning?

- An unknown function: target function
- We know the value of the target function for only some inputs (the training set)

- Two components of learning model
  - A hypothesis set
  - A learning algorithm, which picks one particular hypothesis from the hypothesis set
  - Hopefully, the selected hypothesis function matches the target function


UNKNOWN TARGET FUNCTION
$f: \mathcal{X} \to \mathcal{Y}$
(ideal credit approval function)
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)$
(historical records of credit customers)
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g \approx f$
(final credit approval formula)
HYPOTHESIS SET
$\mathcal{H}$
(set of candidate formulas)

Can we actually learn an unknown function?

- Intuitively, no – the function can behave arbitrarily outside of the given training set

- Is learning feasible?
- Can we say something about the target function outside of what we know?

A probabilistic experiment

- Consider a bin with red and green marbles
- P [pick a red marble] = μ
- P [pick a green marble] = $1 - \mu$

- We pick N marbles independently
- Fraction of red marbles in sample = ν

- Does ν (known) say anything about μ (unknown)?
- Possibility vs. Probability

Hoeffding's Inequality

In a big sample (large N), ν is probably close to μ (within ϵ).

Formally,

$$\mathbb{P}\left[\,|\nu - \mu| > \epsilon\,\right] \leq 2e^{-2\epsilon^2 N}$$

Sample size N is dampened by ε^2

The statement "$\mu = v$" is P.A.C (probably approximately correct).

Hoeffding’s Inequality

$$\mathbb{P}\left[|\nu - \mu| > \epsilon\right] \leq 2e^{-2\epsilon^2 N}$$

One of the laws of large numbers

Valid for all N and ε
Bound does not depend on μ (desirable)
Tradeoff: N, ε, and the bound

μ is unknown, ν is known

Connection to Learning

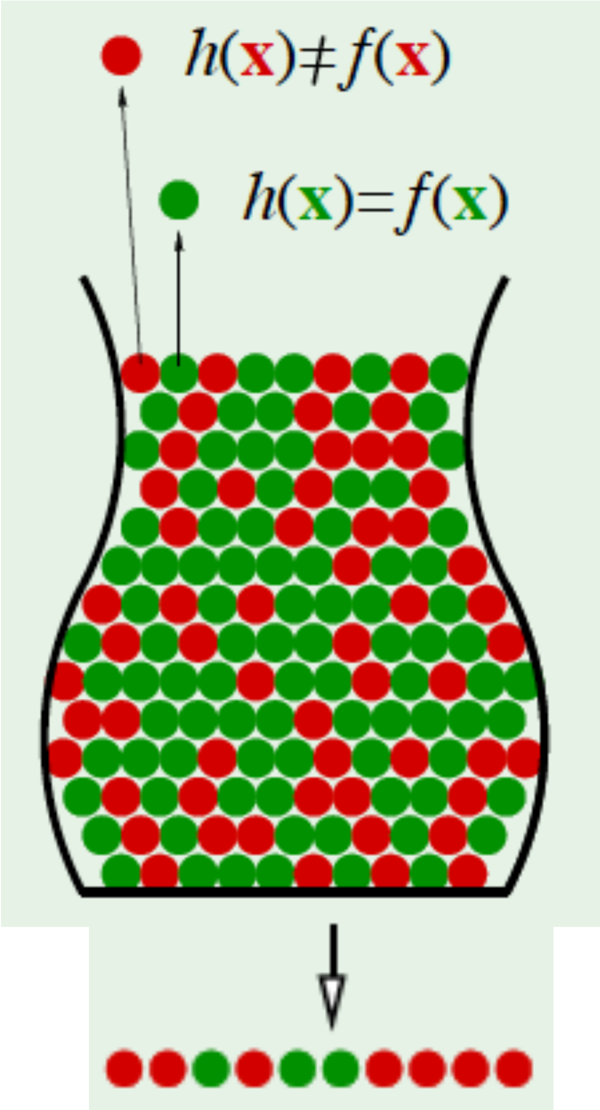
- Bin: the unknown is a number
- Learning: the unknown is the target function f: X → Y

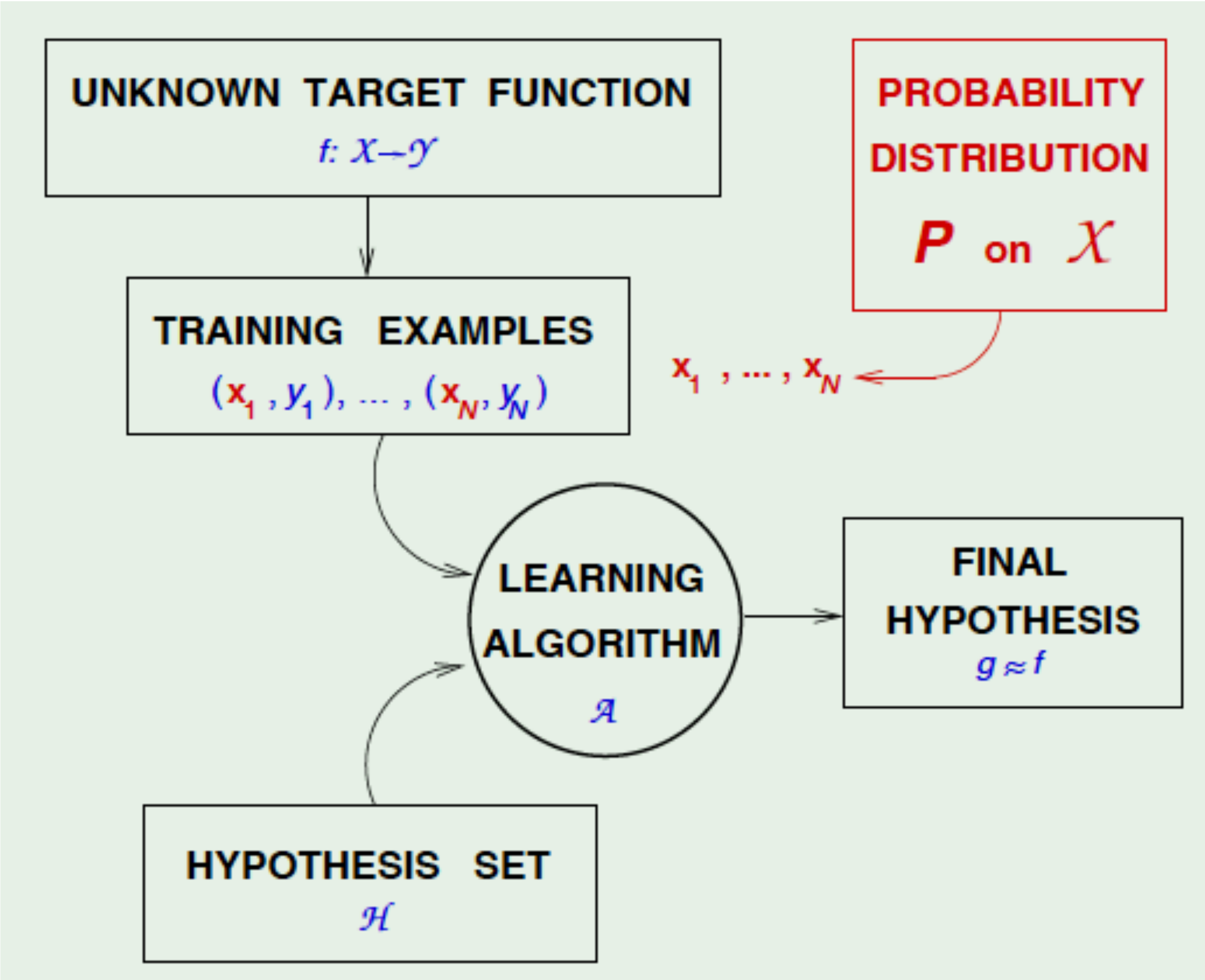
- How to connect the bin analogy to the learning problem?

Connection to Learning

- Each marble is a point x ε X
- Color a marble x green if h(x)=f(x)
- Color a marble x red if h(x) ≠ f(x)
- Sample analogous to training set
- Bin analogous to actual population

- How is the sample generated from the bin?



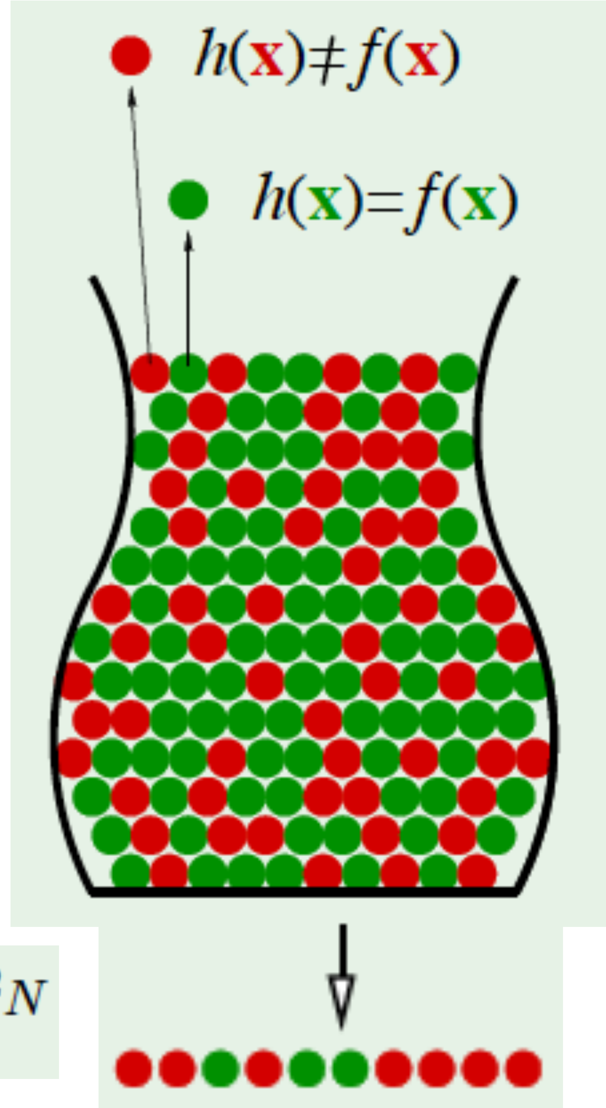

UNKNOWN TARGET FUNCTION
$f: \mathcal{X} \to \mathcal{Y}$
PROBABILITY DISTRIBUTION
P on $\mathcal{X}$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
$\mathbf{x}_1, \ldots, \mathbf{x}_N$
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
$g \approx f$
HYPOTHESIS SET
$\mathcal{H}$

Connection to Learning

- ν: fraction of red marbles in sample = in-sample error $E_{in}(h)$
- μ: fraction of red marbles in population = out-of-sample error $E_{out}(h)$

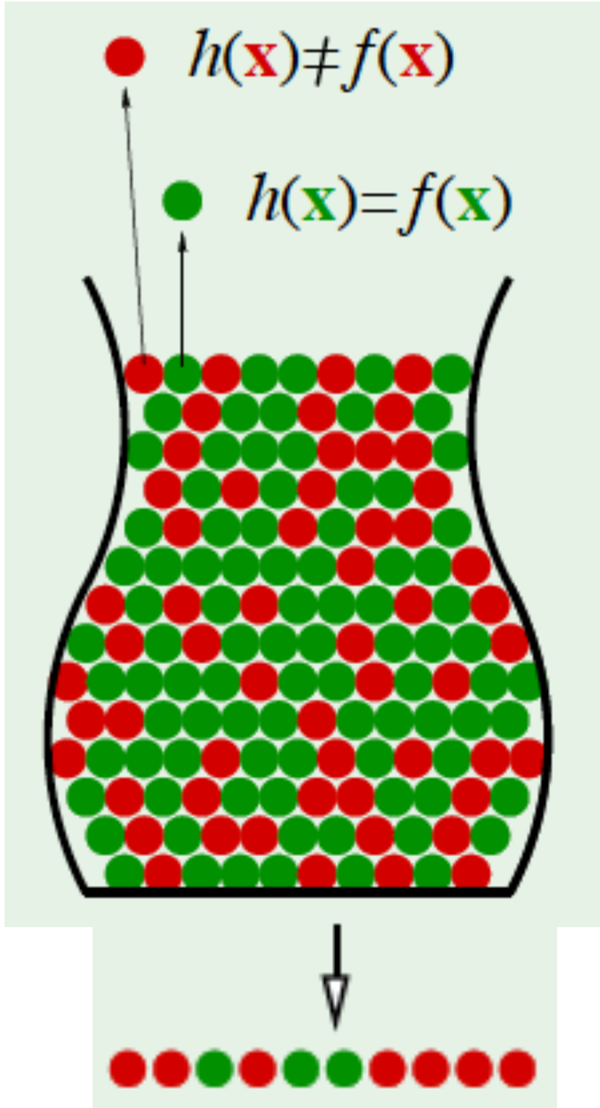
- Hoeffding's inequality:

$$\mathbb{P}\left[\, |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon \,\right] \le 2e^{-2\epsilon^2 N}$$

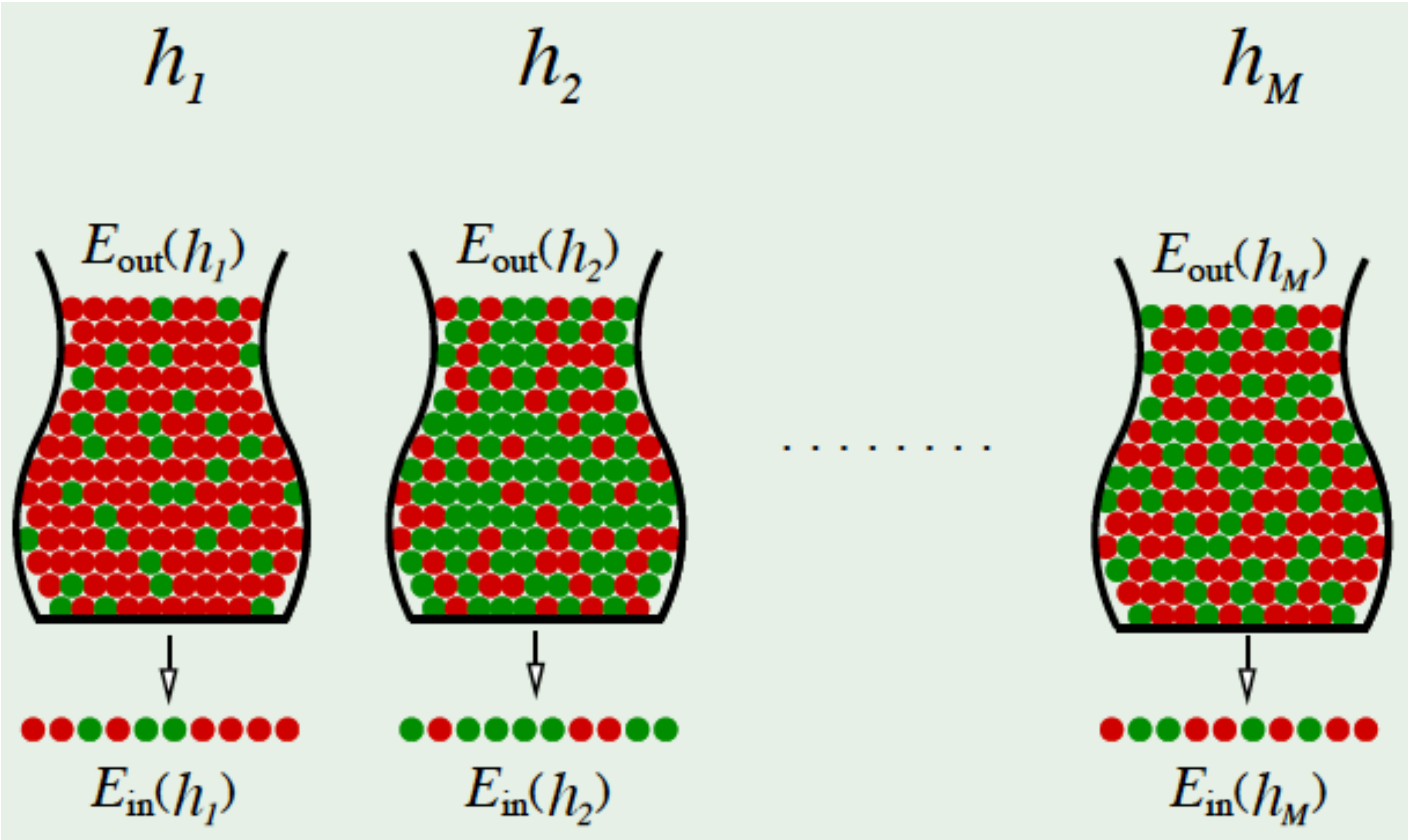


A problem with our formulation

- Both $E_{in}(h)$ and $E_{out}(h)$ is decided by the hypothesis h
- No guarantee that $E_{in}(h)$ will be small
- We need to find a h for which v (hence μ) is small



Multiple bins = multiple hypotheses



Another problem with our formulation

- Hoeffding’s inequality does <u>not</u> apply to multiple bins

- If an experiment is tried many times, probability of an event in <u>some</u> trial can be much greater than the probability of that event in a particular trial

Example

- Toss a fair coin 10 times. What is the probability of getting 10 heads?
- Toss 1000 fair coins 10 times each. What is the probability of getting 10 heads with <u>some</u> coin?

Bounds with multiple bins

- Let g be the hypothesis with minimum in-sample error

$$
\begin{aligned}
\mathbb{P}[\ |E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\ ] \ \leq\ & \mathbb{P}[\ \ \ \ \ |E_{\text{in}}(h_1) - E_{\text{out}}(h_1)| > \epsilon \\
& \quad\ \textbf{or } |E_{\text{in}}(h_2) - E_{\text{out}}(h_2)| > \epsilon \\
& \quad\ \cdots \\
& \quad\ \textbf{or } |E_{\text{in}}(h_M) - E_{\text{out}}(h_M)| > \epsilon\ ] \\
\leq\ & \sum_{m=1}^{M} \mathbb{P}\left[|E_{\text{in}}(h_m) - E_{\text{out}}(h_m)| > \epsilon\right]
\end{aligned}
$$

Bounds with multiple bins

- Let g be the hypothesis with minimum in-sample error

$$\begin{aligned}\mathbb{P}[\ |E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\ ] \ &\leq \ \sum_{m=1}^{M} \mathbb{P}\left[|E_{\text{in}}(h_m) - E_{\text{out}}(h_m)| > \epsilon\right] \\ &\leq \ \sum_{m=1}^{M} 2e^{-2\epsilon^2 N}\end{aligned}$$

The final bound

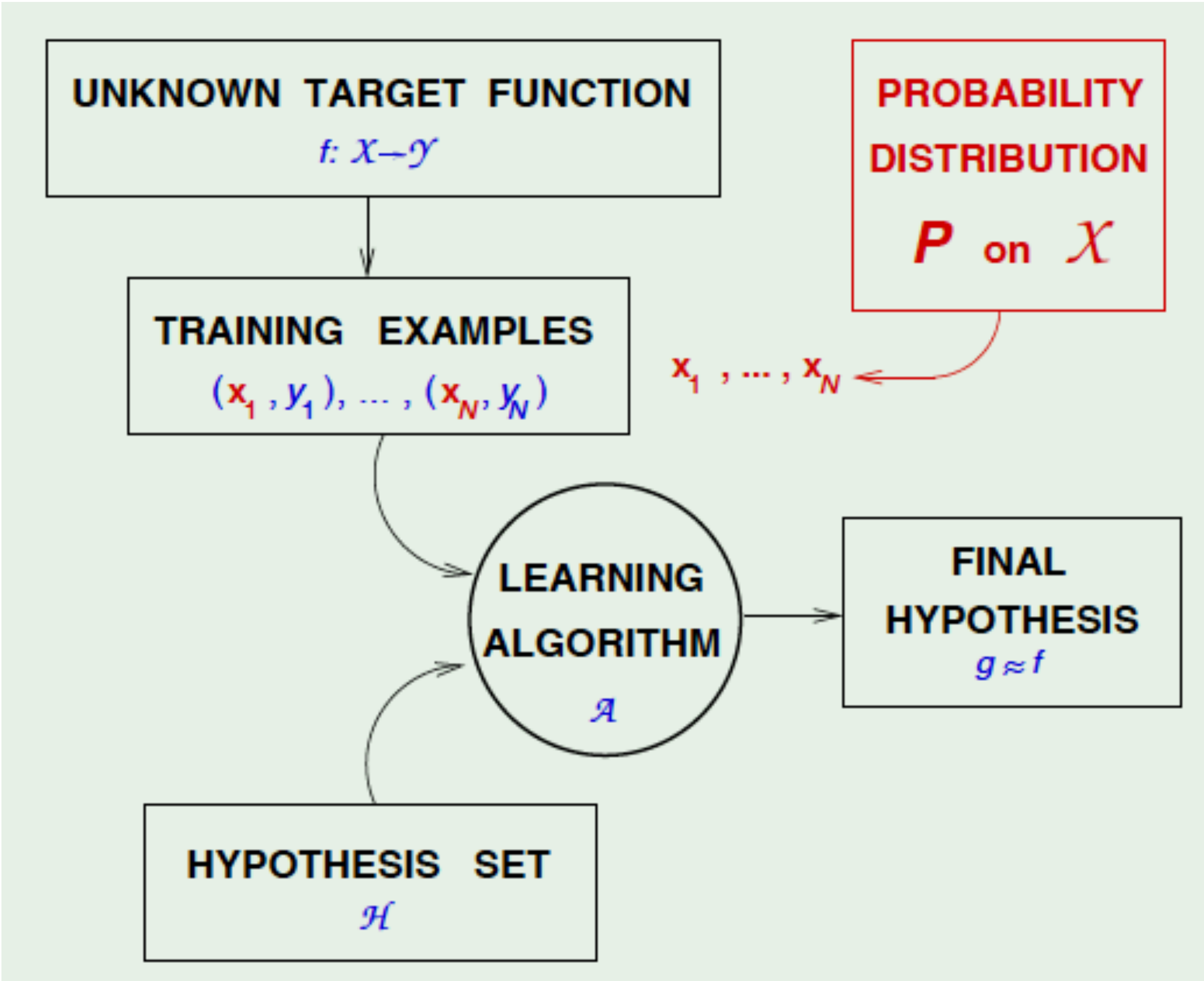
- Let g be the hypothesis with minimum in-sample error

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

- M: number of different hypotheses, from which g is chosen

Till now

- Learning is feasible, in a probabilistic sense



Next

- Two components that connect the learning problem to a practical application
  - Error measures
  - Noisy targets

Error measures

- How closely does a hypothesis h resemble the target function f?
- Error measure E(h, f)
  - Almost always a point-wise definition in terms of point x: e(h(x), f(x))
  - E.g., squared error $e = (h(x) – f(x))^2$
  - E.g., binary error e = 1 if h(x) = f(x), 0 otherwise
- How to go from point-wise to global?

Error measures

In-sample error:

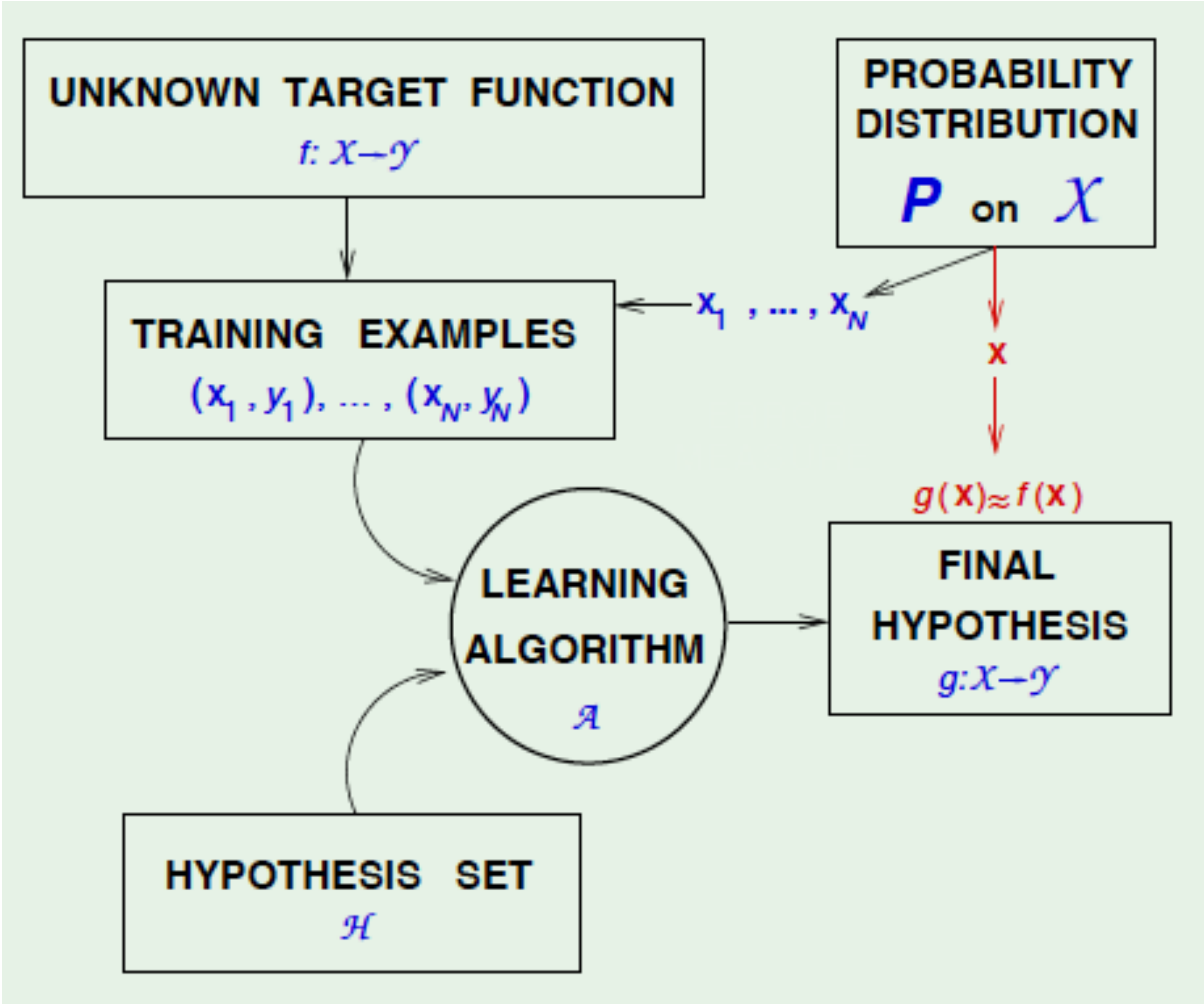
$$E_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^{N} \mathrm{e}\left(h(\mathbf{x}_n), f(\mathbf{x}_n)\right)$$

Out-of-sample error:

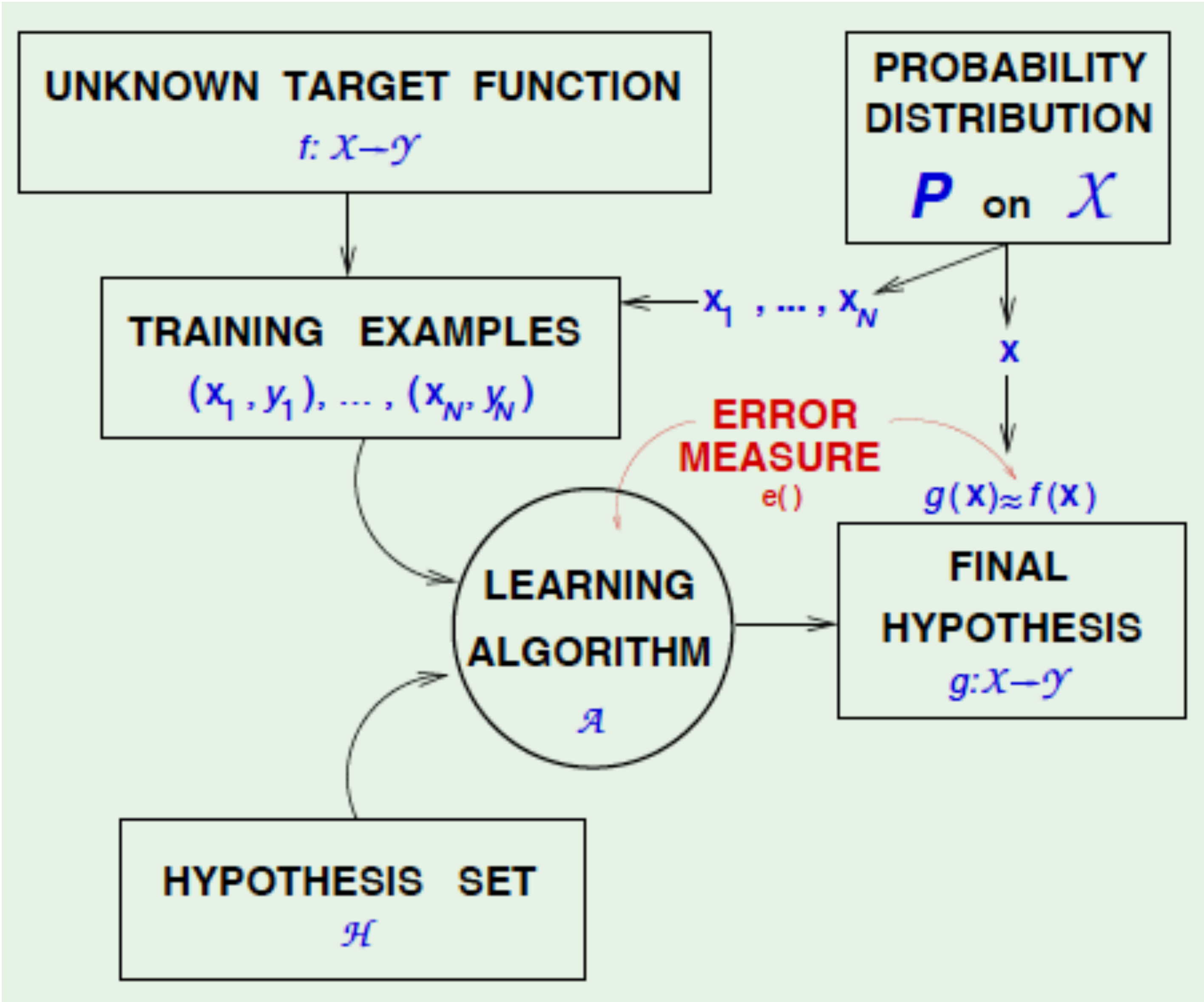
$$E_{\text{out}}(h) = \mathbb{E}_{\mathbf{x}}\left[\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right)\right]$$

- Test data will be selected through a probability distribution
- Hence, out-of-sample error is an expectation

Learning diagram with point-wise error



Learning diagram with error function



How to choose the error measure

- Two types of errors:
  - False positive/accept: hypothesis +1, target -1
  - False negative/reject: hypothesis -1, target +1
- How do we penalize each type?

		f	
		$+1$	-1
h	$+1$	no error	*false accept*
	-1	*false reject*	no error

Example: Fingerprint verification

- Input fingerprint, classify as known identity or intruder
- Application 1: Supermarket verifies customers for giving a discount
- Application 2: For entering into RAW, GoI

Example: Fingerprint verification

- Input fingerprint, classify as known identity or intruder

- Application 1: Supermarket verifies customers for giving a discount

		f	
		$+1$	-1
h	$+1$	0	1
	-1	10	0

- Application 2: For entering into RAW, GoI

Example: Fingerprint verification

- Input fingerprint, classify as known identity or intruder

- Application 1: Supermarket verifies customers for giving a discount

		f: $+1$	f: -1
h	$+1$	0	1
	-1	10	0

- Application 2: For entering into RAW, GoI

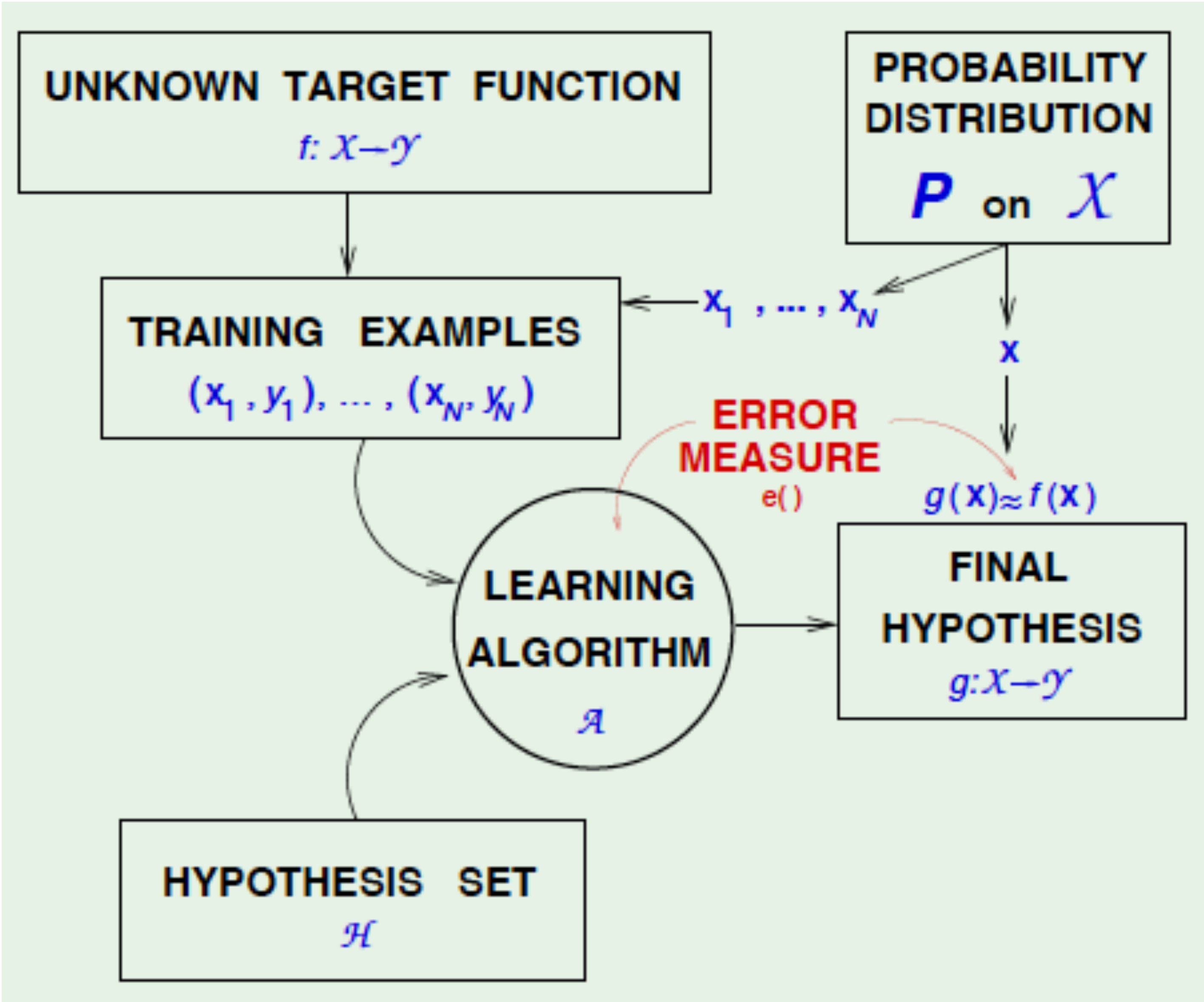
		f: $+1$	f: -1
h	$+1$	0	1000
	-1	1	0

Error measure

- Ideally, e(h(x), f(x)) should be defined by the end-user or domain expert

- Alternatives:
  - Something plausible
  - Measures which make the learning efficient / give closed-form solutions, e.g., squared error

Learning diagram with error function



Noisy targets

- The ‘target function’ is not always a function
- Two identical inputs can lead to two different behaviors / decisions
  - A particular user may rate a particular movie differently at different times, based on mood
  - Given two identical applications for a job / for credit, one may be selected but not the other

Noisy targets

- Instead of deterministic target function $y = f(x)$, consider target distribution: $y \sim P(y \mid x)$

- Deterministic target is a special case of noisy target: $P(y \mid x)$ is zero except for $y = f(x)$

- Noisy target = deterministic target plus noise

$$f(x) = E(y \mid x) \qquad\qquad y - f(x)$$

P(y | x) and P(x)

- P(y | x) is the target distribution that we are trying to learn
- P(x) is the input distribution that quantifies relative importance of x in the training sample (and hopefully also in test set)

P(y | x) and P(x)

- $P(y \mid x)$ is the target distribution
- $P(x)$ is the input distribution

- Training examples (x_1,y_1), (x_2,y_2), …, (x_N,y_N) generated by the joint distribution $P(x)P(y \mid x)$
- We assume each training example (x, y) to be generated independently

- Out-of-sample error is now $E_{x,y}[\, e(h(x), y) \,]$

Final learning diagram