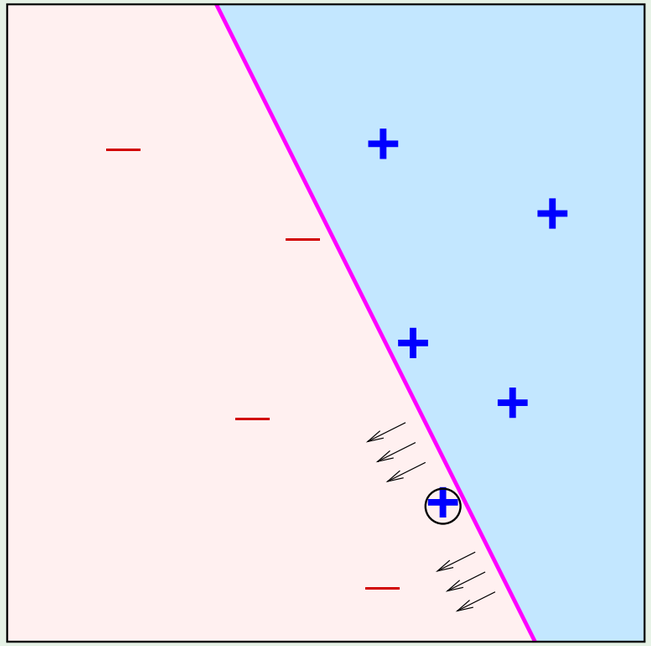
Review of Lecture 1

- **Learning is used when**
  - A pattern exists
  - We cannot pin it down mathematically
  - We have data on it
- **Focus on supervised learning**
  - Unknown target function $y = f(\mathbf{x})$
  - Data set $(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_N, y_N)$
  - Learning algorithm picks $g \approx f$ from a hypothesis set $\mathcal{H}$

Example: Perceptron Learning Algorithm



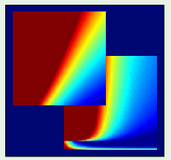
- **Learning an unknown function?**
  - Impossible ☹. The function can assume any value outside the data we have.
  - So what now?

Learning From Data

Yaser S. Abu-Mostafa
California Institute of Technology

Lecture 2: **Is Learning Feasible?**

Sponsored by Caltech's Provost Office, E&AS Division, and IST • Thursday, April 5, 2012



Feasibility of learning - Outline

- Probability to the rescue
- Connection to learning
- Connection to *real* learning
- A dilemma and a solution

A related experiment

- Consider a 'bin' with red and green marbles.

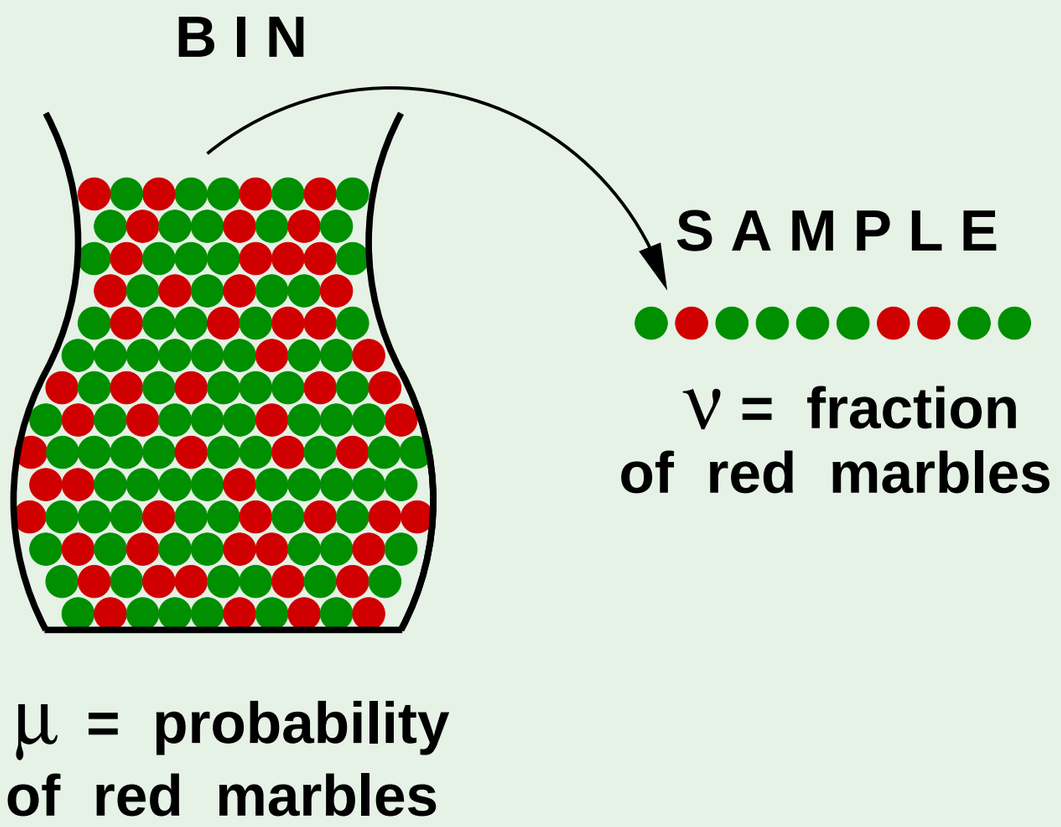
$$\mathbb{P}[\text{ picking a red marble }] = \mu$$

$$\mathbb{P}[\text{ picking a green marble }] = 1 - \mu$$

- The value of μ is unknown to us.

- We pick N marbles independently.

- The fraction of red marbles in sample $= \nu$



Does ν say anything about μ?

No!

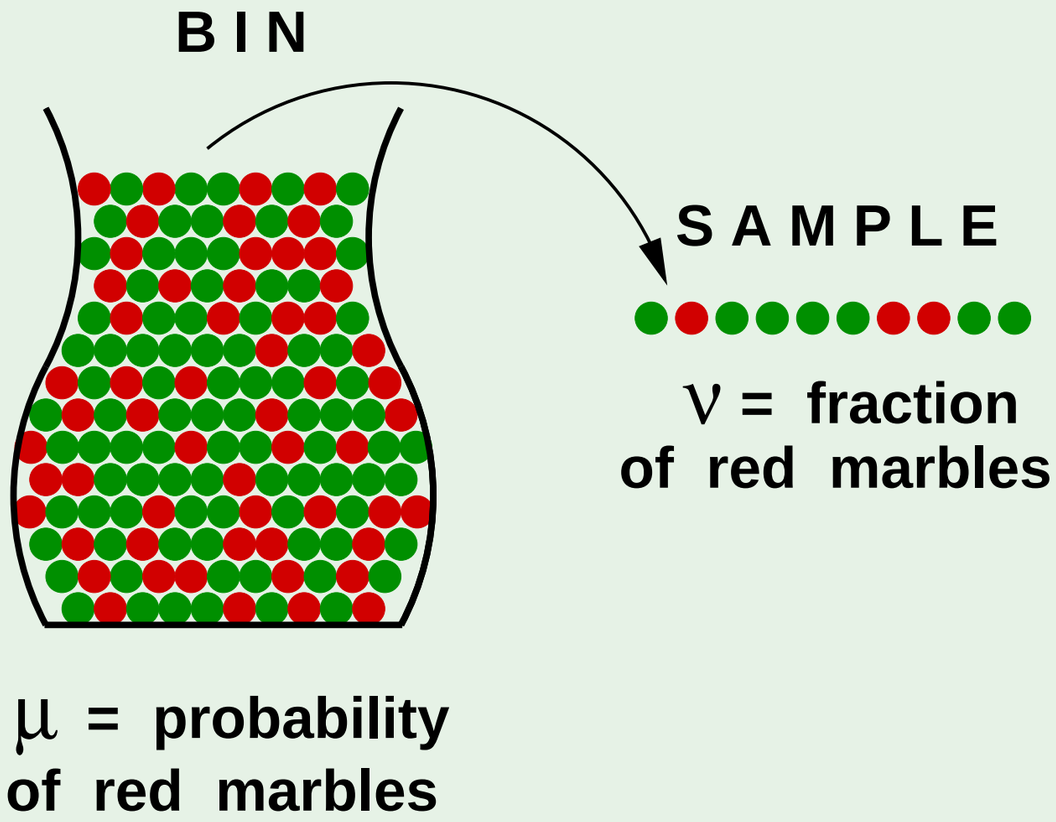
Sample can be mostly green while bin is mostly red.

Yes!

*if the sample is big enough

Sample frequency ν is likely close to bin frequency μ.

possible versus **probable**



What does ν say about μ?

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

Formally,

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

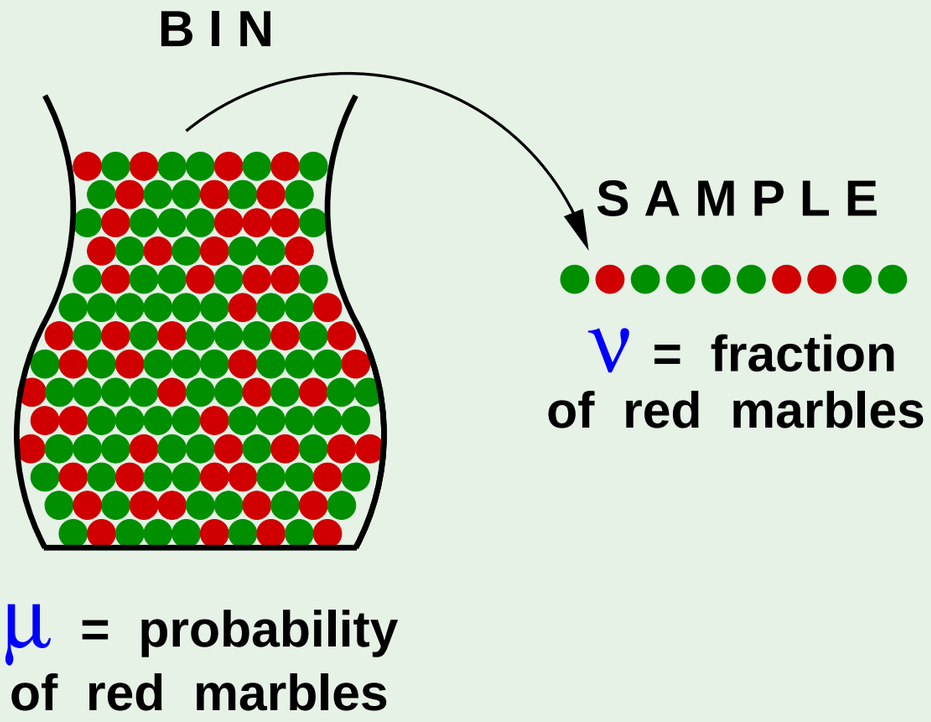
This is called **Hoeffding's Inequality**.

In other words, the statement "$\mu = \nu$" is P.A.C.

(probably approximately correct) - this inequality shows that verification is feasible (i.e. the answer from your sample will be close to the population answer)

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

- Valid for all N and ϵ
- Bound does not depend on μ
- Tradeoff: N, ϵ, and the bound.
- $\nu \approx \mu \implies \mu \approx \nu$ ☺



i.e we infer the bin probability mu from the sample nu (it is easy to do it the other way around: of course nu (a variable dependant on mu due to the many possible samples that could be drawn) may be determined by mu as mu is a constant determined by the state of the bin)

Connection to learning

The greyed-out bin is the space of all possible points in X. The hypothesis adds the colour according to the below rules The sample is the dataset of x-space points we are given.

Bin: The unknown is a number μ

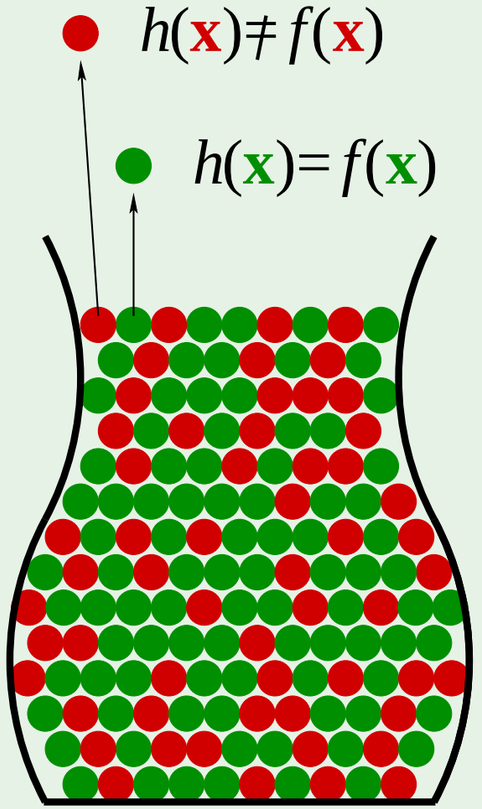
Learning: The unknown is a function $f : \mathcal{X} \rightarrow \mathcal{Y}$

Note: H, the hypothesis set, is determined by the model we are using (perceptron, SVM etc.) while h are essentially all the possible weights/the different boundaries in the plane (referring to the perceptron example)

Each marble ● is a point $\mathbf{x} \in \mathcal{X}$

● : Hypothesis got it right $h(\mathbf{x}) = f(\mathbf{x})$

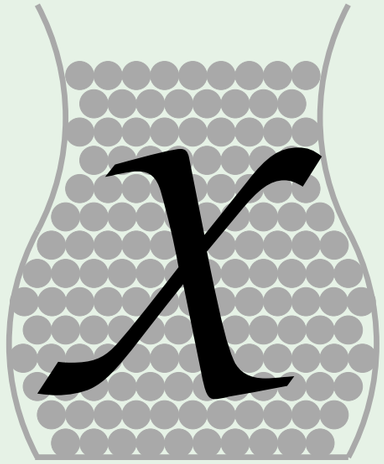
● : Hypothesis got it wrong $h(\mathbf{x}) \neq f(\mathbf{x})$

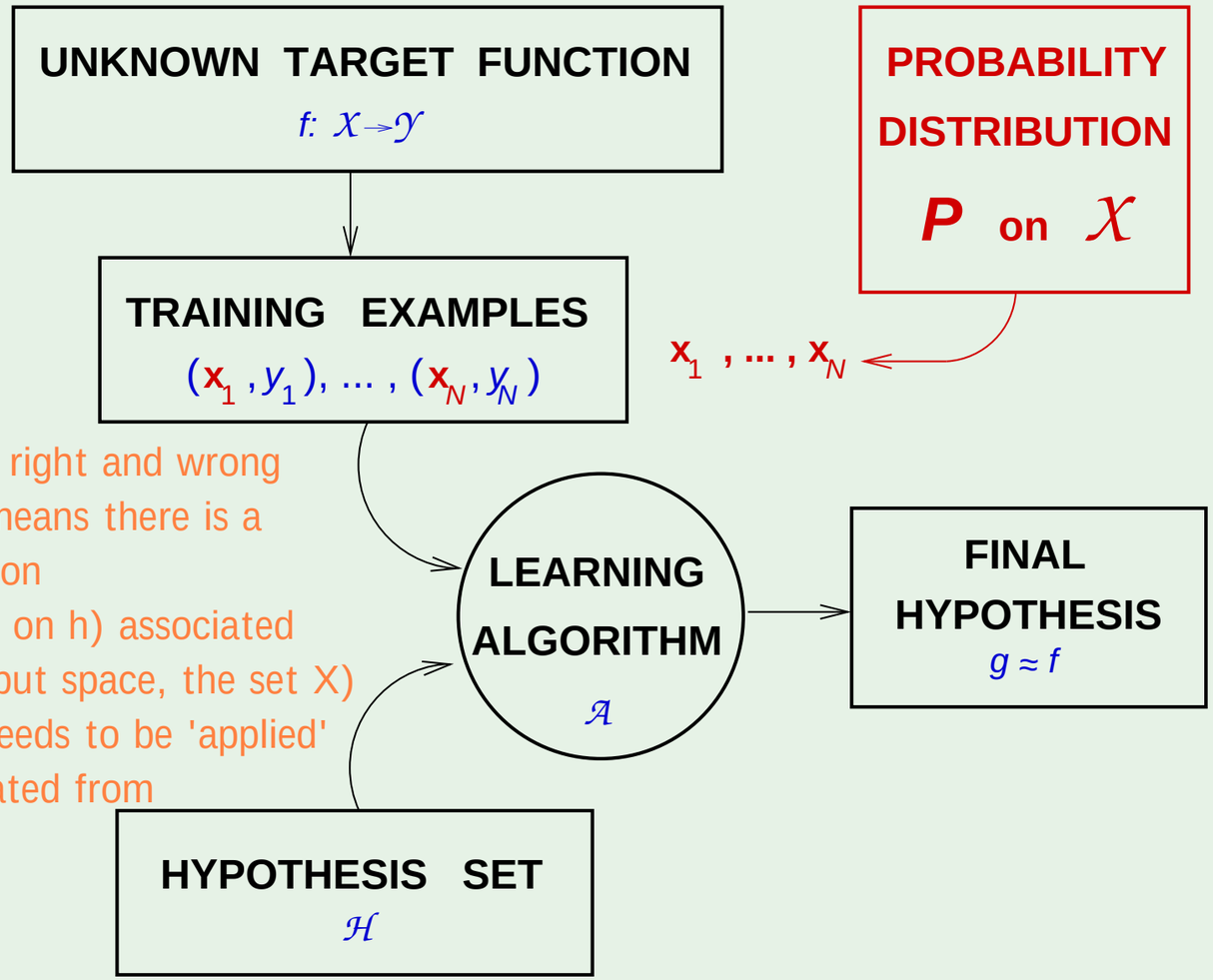


The data we are given is essentially of the form (xi, f(xi)) so we can only compare h(x) to f(x) at the points {xi} (our dataset)

Back to the learning diagram

The bin analogy:





The separation into right and wrong from the last slide means there is a probability distribution (which is dependent on h) associated with the bin (the input space, the set X) so this probability needs to be 'applied' to the points generated from the input space (our sample, {xi})

However, we are not restricting the possible P's over x and we do not need to know this P - due to Hoeffding, we can bound performance independantly of P

Are we done?

Not so fast! h is fixed.

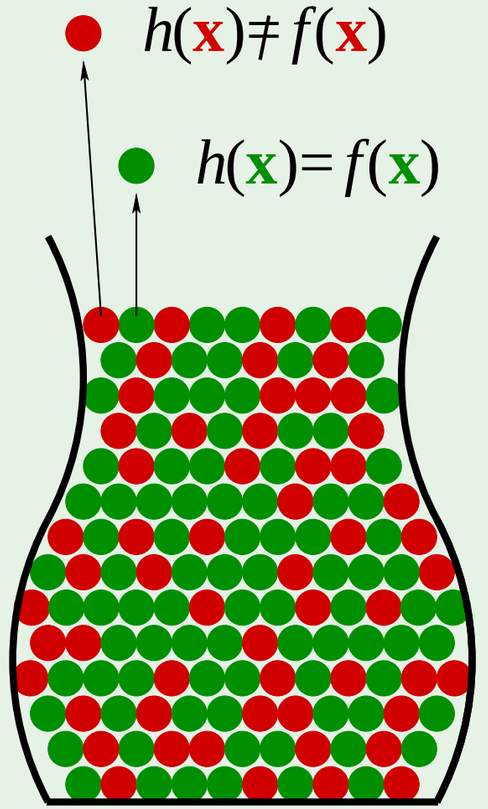
For this h, ν generalizes to μ.

This is equivalent to a...

'**verification**' of h, not **learning**

No guarantee ν will be small.

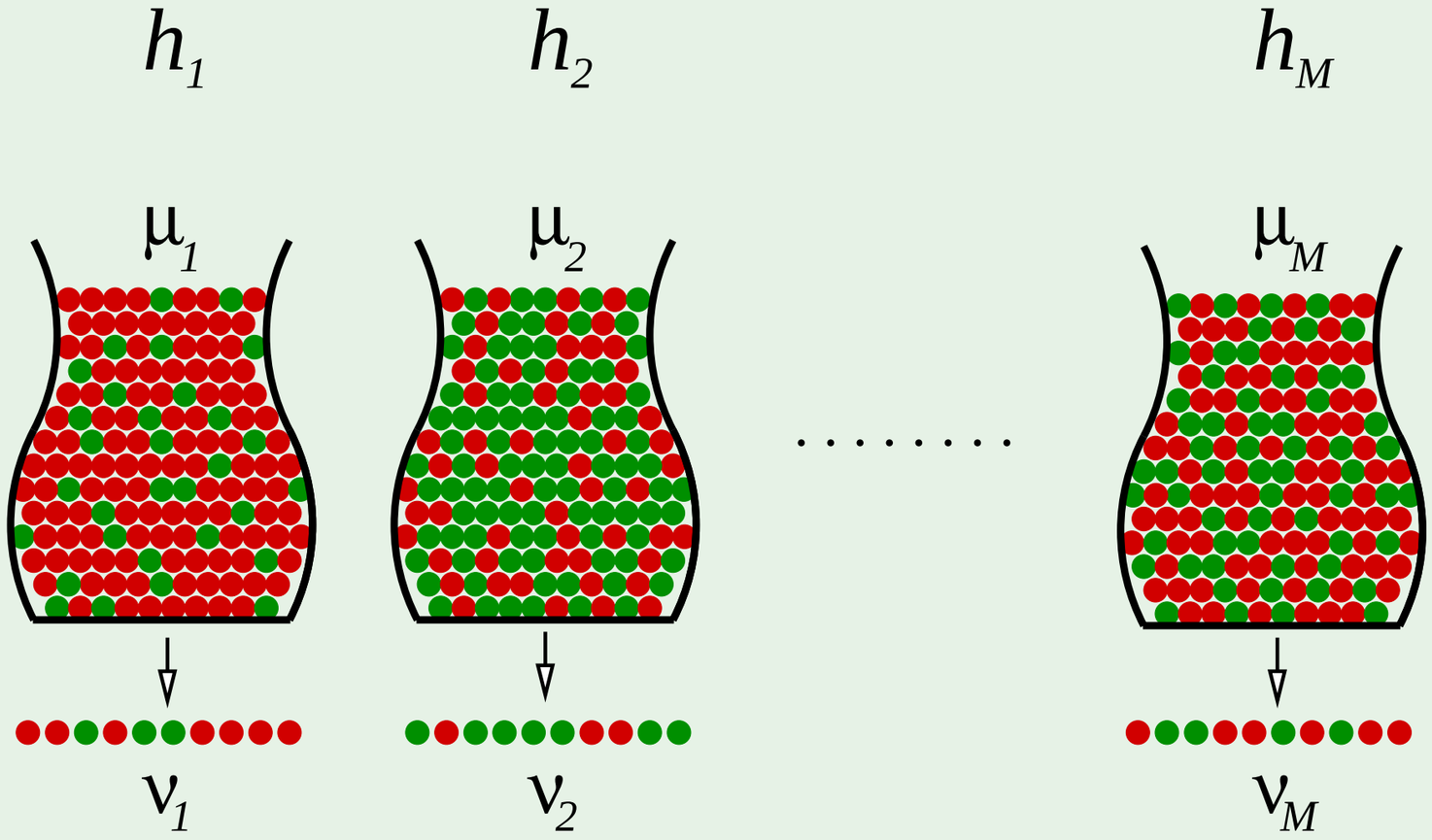
We need to **choose** from multiple h's.



We need to search H to find a candidate hypothesis which works well on the data:
we analyze the application of each candidate to the data sample and pick the one with least error/most favorable.

Multiple bins

Generalizing the bin model to more than one hypothesis:

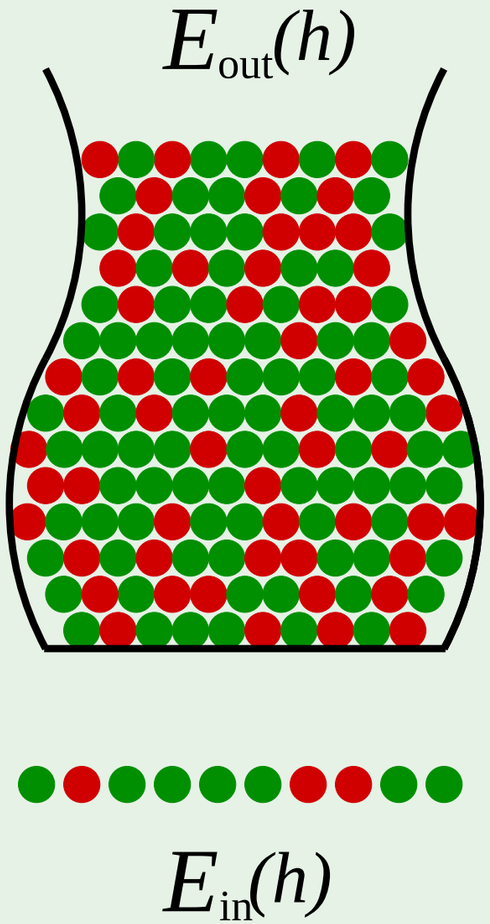


Notation for learning

Both μ and ν depend on which hypothesis h

ν is '**in sample**' denoted by $E_{\text{in}}(h)$ = in-sample or 'dataset' performance

μ is '**out of sample**' denoted by $E_{\text{out}}(h)$ = out-of-sample or 'general' performance on the input space

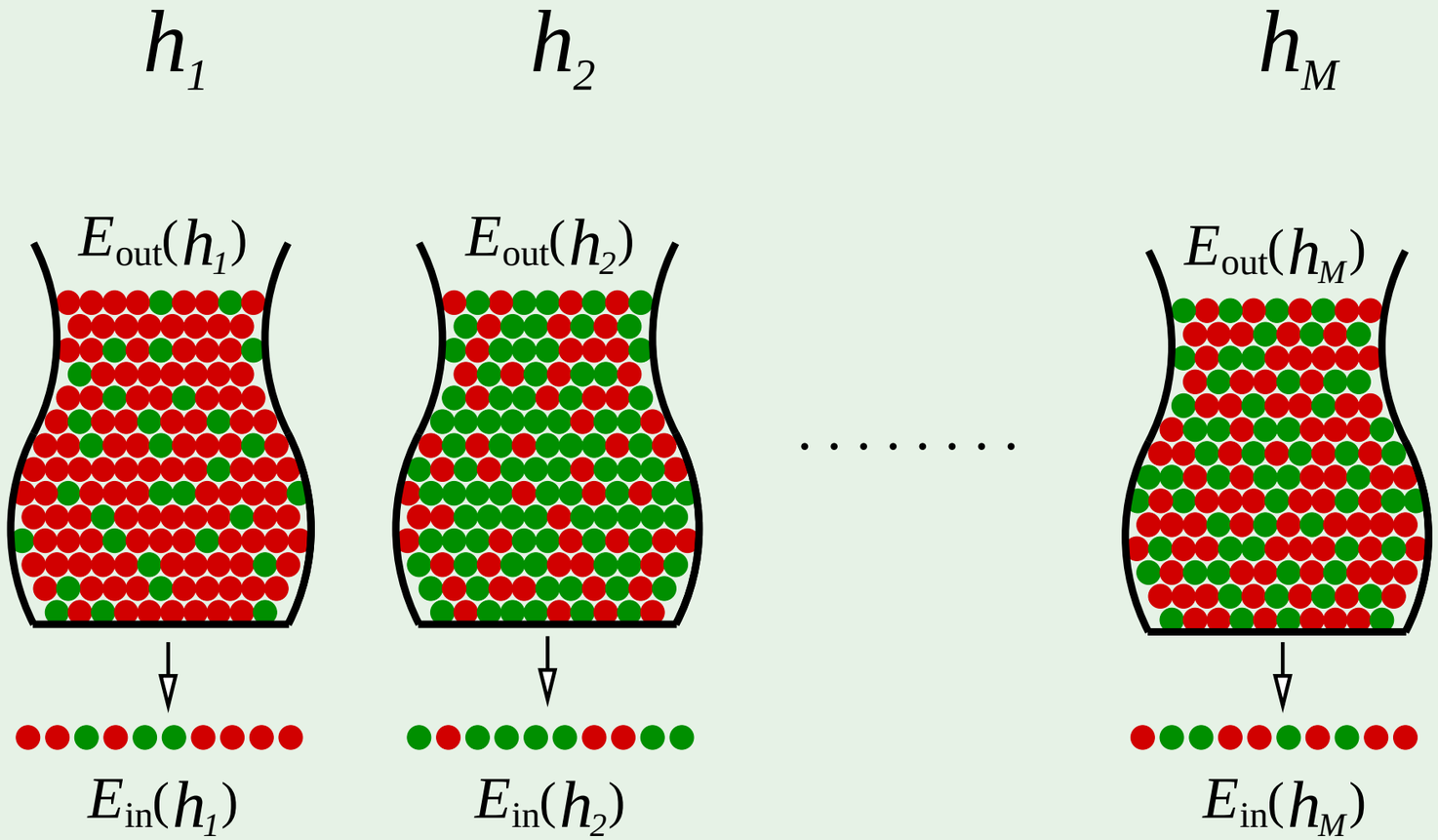


The Hoeffding inequality becomes:

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

We want a very small prob. of the difference between in- and out-of- sample performance being greater than our tolerance.

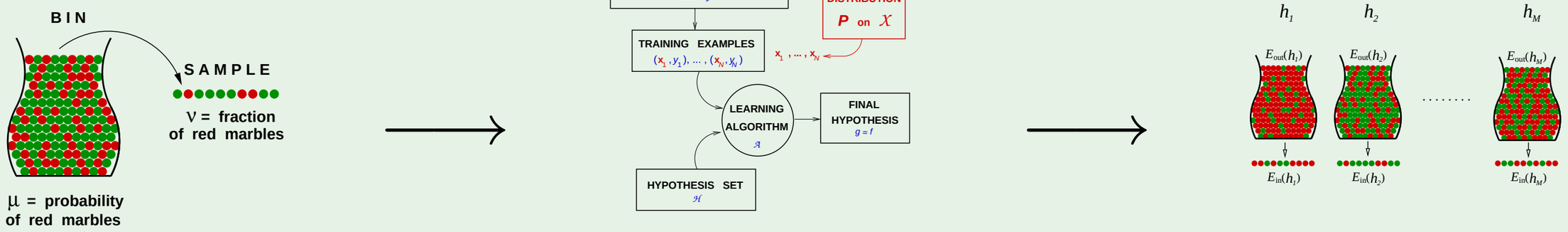
Notation with multiple bins



Are we done already? ☺

Not so fast!! Hoeffding doesn't apply to multiple bins.

What?



Coin analogy

Question: If you toss a fair coin 10 times, what is the probability that you will get 10 heads?

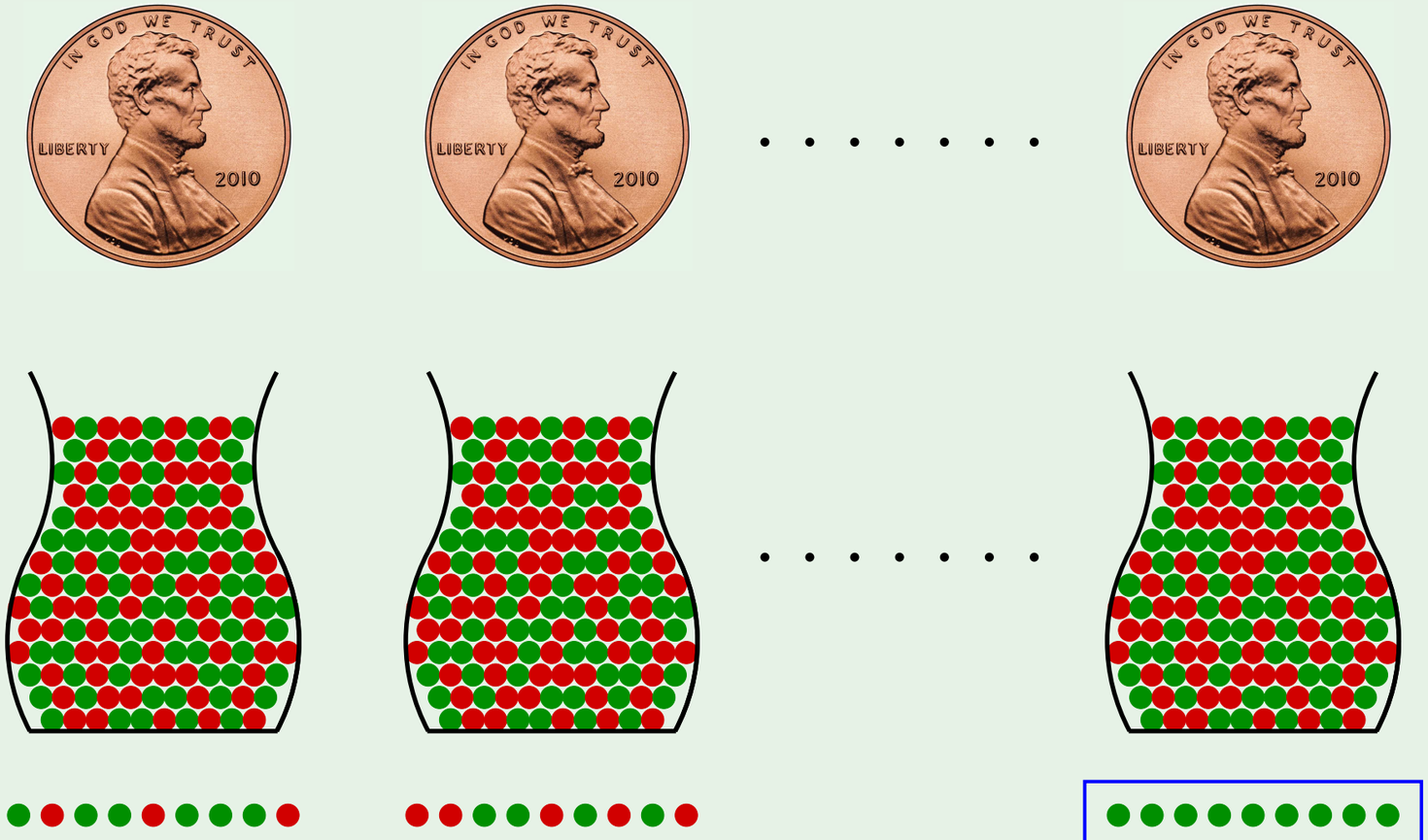
Answer: $\approx 0.1\%$

Just because one person flipping a coin gets 10 heads in a row, it does not mean that we can apply Hoeffding Ineq. and say that the input space only gives heads (i.e. it is a biased coin). - The ten heads give no indication of the real probability. => For multiple bins, we can not use the standard expression of Hoeffding and assume one sample is representative of the entire input space (see next slide).

Question: If you toss 1000 fair coins 10 times each, what is the probability that some coin will get 10 heads?

Answer: $\approx 63\%$

From coins to learning



Even if something has a small probability of happening, if you try hard enough with enough trials it will happen, but you cannot just apply Hoeffding to this improbable event (ignoring the rest) and say it is representative of the probability.

Actual learning requires the testing of multiple hypotheses (multiple bins, since different h means different marbles are red/green) and the application of the new Hoeffding below (as opposed to the vanilla version, only applicable to single hypotheses (which is not learning as we know it - defined as the exploration of multiple hypotheses based on their performance in sample, picking the one with best in sample performance and hoping this generalizes out of sample))

A simple solution - extend Hoeffding to multiple bins (many hypotheses)

g = the hypothesis chosen by the learning process, it is no longer a fixed hypothesis (it is 'loaded') -
it corresponds to/relies on the many hypotheses we have trialled (hence Hoeffding does not apply in its initial form)

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

P(chosen hyp. is bad) <= P(h1 is bad or h2 is bad or ... or hM is bad)

The final verdict

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

$$\leq \ \sum_{m=1}^{M} 2e^{-2\epsilon^2 N}$$

In a genuine learning scenario, where we try find g (in our example, the bin with a sample with as much green as possible - or test each hypothesis and find the one with the best performance on our sample dataset), the resultant Hoeffding expression ends up depending on the number of candidate hypotheses M

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

So, with a more sophisticated model (M very large), the 'looser' the in sample performance will track the out of sample - it is possible to memorize in-sample and not generalize well out of sample because there are so many parameters to work with.