CSE4088 Introduction to Machine Learning

Is learning feasible?

Some of the slides are adopted from lecture notes of Yaser Abu-Mostafa

Outline – Feasibility of Learning

- A learning puzzle
- A related probability problem
- Connection to real learning
- A dilemma and solution

A Learning Puzzle

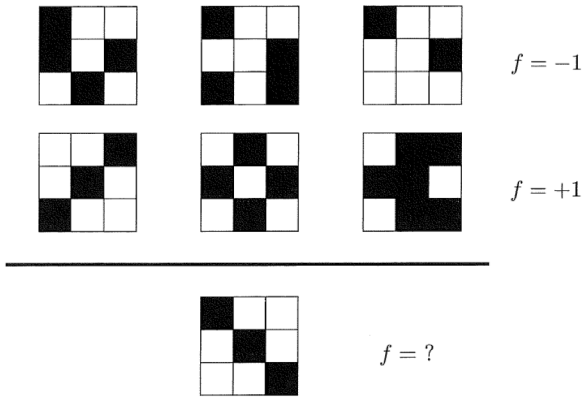


Figure 1.7: A visual learning problem. The first two rows show the training examples (each input $\mathbf{x}$ is a 9-bit vector represented visually as a 3×3 black-and-white array). The inputs in the first row have $f(\mathbf{x}) = -1$, and the inputs in the second row have $f(\mathbf{x}) = +1$. Your task is to learn from this data set what f is, then apply f to the test input at the bottom. Do you get -1 or $+1$?

A Learning Puzzle

The chances are the answers were not unanimous, and for good reason. There is simply more than one function that fits the 6 training examples, and some of these functions have a value of -1 on the test point and others have a value of $+1$. For instance, if the true f is $+1$ when the pattern is symmetric, the value for the test point would be $+1$. If the true f is $+1$ when the top left square of the pattern is white, the value for the test point would be -1. Both functions agree with all the examples in the data set, so there isn't enough information to tell us which would be the correct answer.

A Learning Puzzle

- The target function f is the object of learning. Target function is unknown.
- Question: How could a limited data set reveal enough information to pin down the entire target function?
- Puzzle example:
  - If we get the training data *D,* e.g. the first two rows of the puzzle example, we know the value of *f* on all the points in *D*.
  - This does not mean that we have learned *f,* since it does not guarantee that we know anything about *f* outside of *D*.
  - We know what we have already seen, but that's not learning. That's memorizing.

Feasibility of Learning

- Does the dataset *D* tell us anything outside of *D* that we didn't know before?
  - If the answer is yes, then we have learned *something*.
  - If the answer if no, we can conclude that learning is not feasible.

Example

Boolean target function over a three-dimensional input space $\mathcal{X} = \{0,1\}^3$. We are given a data set $\mathcal{D}$ of five examples represented in the table below. We denote the binary output by $\circ/\bullet$ for visual clarity,

$\mathbf{x}_n$	y_n
0 0 0	$\circ$
0 0 1	$\bullet$
0 1 0	$\bullet$
0 1 1	$\circ$
1 0 0	$\bullet$

$y_n = f(\mathbf{x}_n)$ for $n = 1, 2, 3, 4, 5$.

Example

- We can enumerate the entire input space since there are only 2^3=8 distinct input vectors.
- We can enumerate the set of all possible target functions since there are 2^8=256 distinct Boolean functions on 3 Boolean inputs.

Example: Problem of learning *f*

- Since *f* is unknown except inside *D,* any function that agrees with *D* could be *f*.
- The table below shows such functions $f_1, f_2 \ldots f_8$. We can not exclude any of them to be the true *f. g* is chosen to match *f* on these examples.

x	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
0 0 0	○	○	○	○	○	○	○	○	○	○
0 0 1	●	●	●	●	●	●	●	●	●	●
0 1 0	●	●	●	●	●	●	●	●	●	●
0 1 1	○	○	○	○	○	○	○	○	○	○
1 0 0	●	●	●	●	●	●	●	●	●	●
1 0 1		?	○	○	○	○	●	●	●	●
1 1 0		?	○	○	●	●	○	○	●	●
1 1 1		?	○	●	○	●	○	●	○	●

Dilemma

- The whole purpose of learning *f* is to be able to predict the value of *f* on points that we haven't seen before.
- Whether the hypothesis chosen agrees with *D* or not makes no difference as far as the performance outside of *D* is concerned.
- Yet the performance outside *D* is all that matters in learning!
- The dilemma is not restricted to Boolean functions, but extends to the general learning problem.
- As long as *f* is an unknown function, knowing *D* can not exclude any pattern of values for *f* outside of *D.* Therefore predictions of *g* outside of *D* are meaningless.
- However, learning is still possible!!

Probability to the Rescue

- We will show that we can indeed infer something outside *D* using only *D* but in a probabilistic way.
- What we infer may not be much compared to learning a full target function, but it will establish the principle that we can reach outside *D.*
- Once we establish that, we will take it to the general learning problem and pin down what we can and cannot learn.

A related experiment

- Case of picking a sample from a bin containing infinitely many red and green marbles.

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

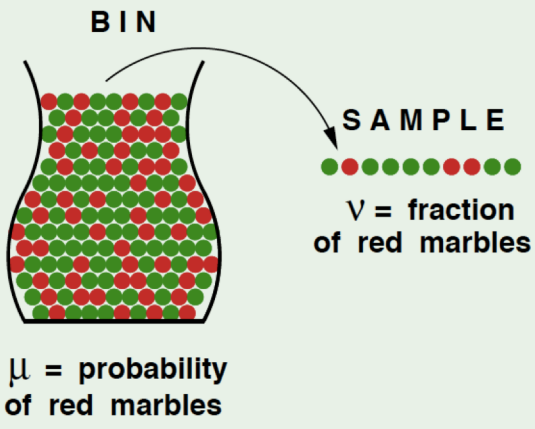
$$\mathbb{P}[\text{ picking a red marble }] = \mu$$

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

- 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?

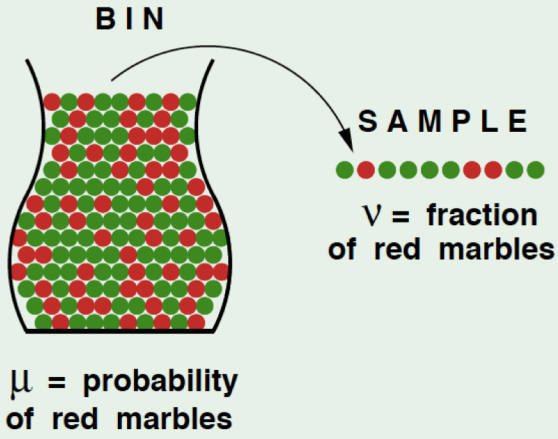
No!

Sample can be mostly green while bin is mostly red.

Yes!

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.

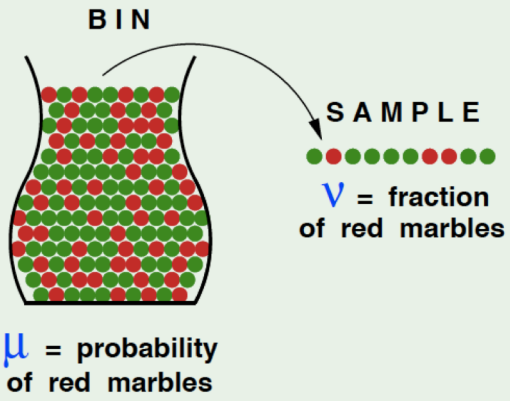
The above equation says that as the sample size *N* grows, it becomes exponentially unlikely that ν will deviate from μ by more than our 'tolerance' ϵ.

Observations

- The only quantity that is random is ν, which depends on the random sample. By contrast μ is not random.
- The utility of the previous inequality is to infer the value of μ using the value of ν, although it is μ that affects ν, not vice versa.
- However, since the effect is that ν tends to be close to μ, we infer that μ 'tends' to be close to ν.
- The bound does not depend on μ.
- Only the size N of the sample affects the bound, not the size of the bin.

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

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



Observations

- If we choose ϵ to be very small in order to make ν a good approximation of μ, we need a larger sample size N to make the RHS of inequality small.
- Knowing that we are within $\mp\epsilon$ of μ most of the time is a significant improvement over not knowing anything at all.
- The fact that the sample was randomly selected from the bin is the reason we are able to make any kind of statement about μ being close to ν.
- If the sample was not randomly selected but picked in a particular way, we would lose the benefit of the probabilistic analysis and would again be in the dark outside the sample.

Connection to learning

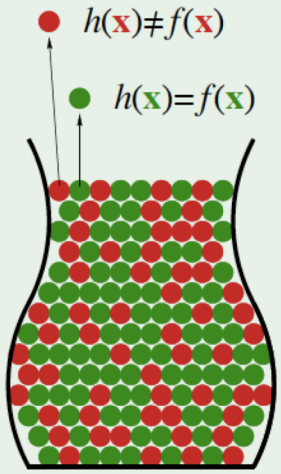
Bin: The unknown is a number μ

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

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

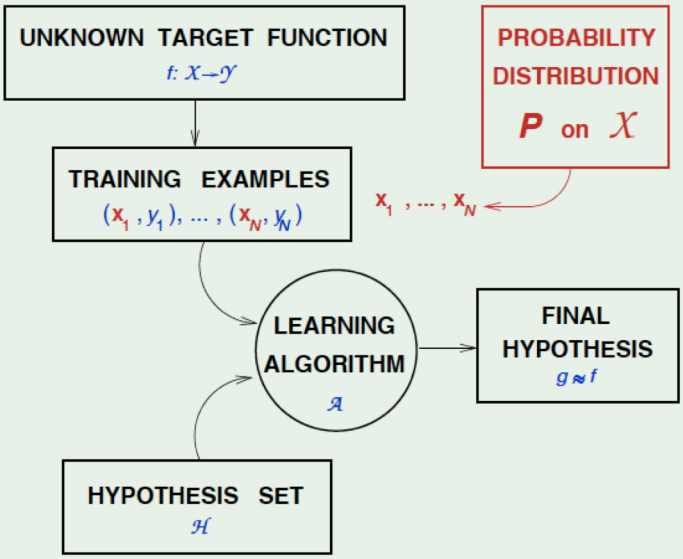
● : Hypothesis got it right $h(\mathbf{x})=f(\mathbf{x})$

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



Back to the learning diagram

The bin analogy:



Are we done?

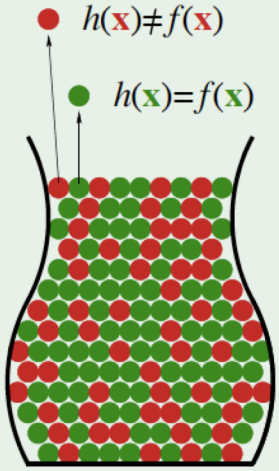
Not so fast! h is fixed.

For this h, ν generalizes to μ.

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

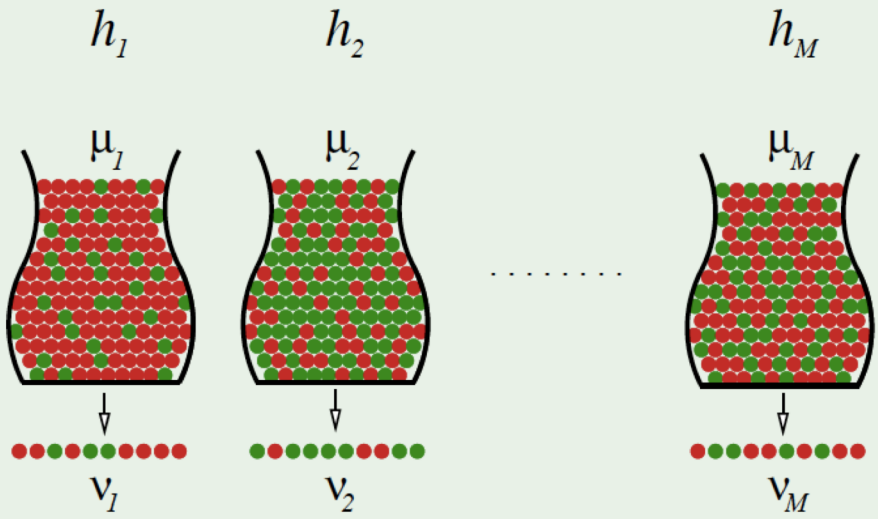
No guarantee ν will be small.

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



Multiple bins

Generalizing the bin model to more than one hypothesis:



Notation

- in-sample error

$$E_{\text{in}}(h) = (\text{fraction of } \mathcal{D} \text{ where } f \text{ and } h \text{ disagree})$$
$$= \frac{1}{N}\sum_{n=1}^{N} [\![h(\mathbf{x}_n) \neq f(\mathbf{x}_n)]\!],$$

$$[\![\text{statement}]\!] = 1 \text{ if the statement is true,}$$

- Out of sample error

$$E_{\text{out}}(h) = \mathbb{P}[h(\mathbf{x}) \neq f(\mathbf{x})]$$

Notation for learning

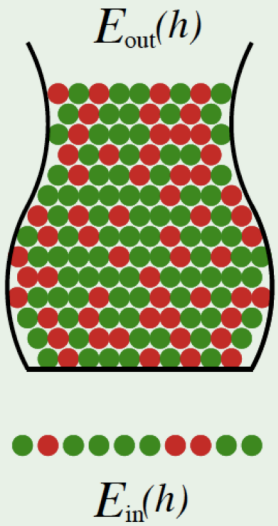
Both μ and ν depend on which hypothesis h

ν is 'in sample' denoted by $E_{\text{in}}(h)$

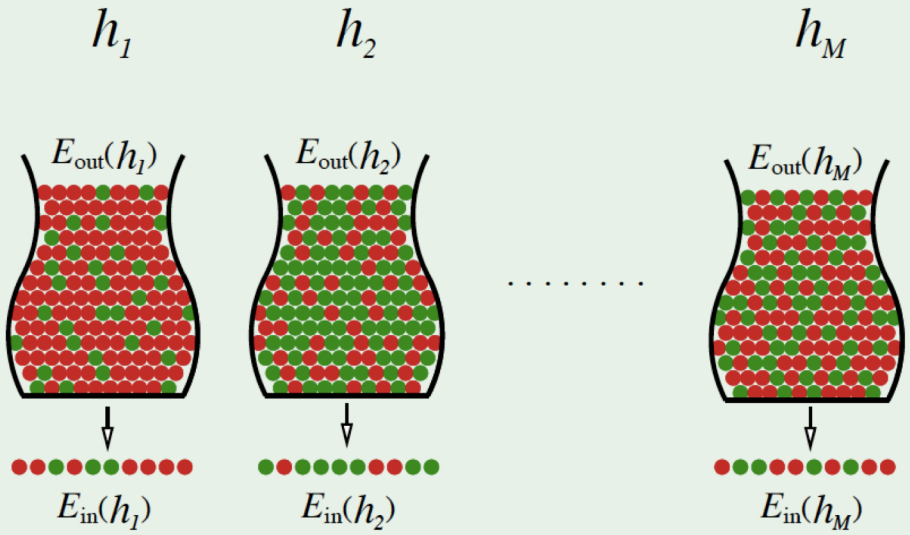
μ is 'out of sample' denoted by $E_{\text{out}}(h)$

The Hoeffding inequality becomes:

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



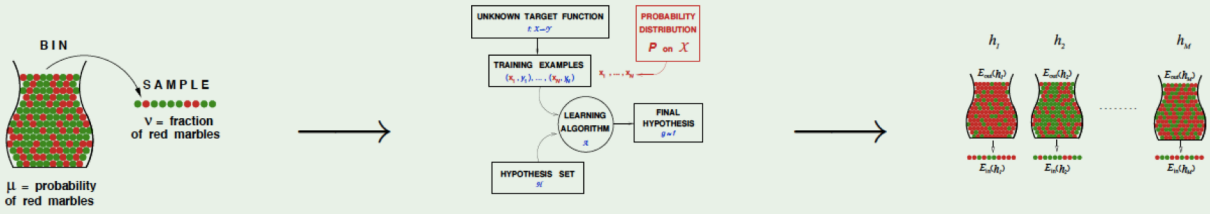
Notation with multiple bins



Are we done already? ☺

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

What?



Multiple Bins

- The statement we would like to make is not

$$\text{“}\mathbb{P}[|E_{\text{in}}(h_m) - E_{\text{out}}(h_m)| > \epsilon] \text{ is small”}$$

(for any particular, fixed $h_m \in \mathcal{H}$), but rather

$$\text{“}\mathbb{P}[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon] \text{ is small” for the final hypothesis } g.$$

The hypothesis g is *not fixed* ahead of time before generating the data, because which hypothesis is selected to be g depends on the data. So, we cannot just plug in g for h in the Hoeffding inequality. The next exercise considers a simple coin experiment that further illustrates the difference between a fixed h and the final hypothesis g selected by the learning algorithm.

Coin analogy

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

Answer: $\approx 0.1\%$

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

Answer: $\approx 63\%$

Solution

The probability that you get no tails when you flip a fair coin 10 times is $\left(\frac{1}{2}\right)^{10}$. The probability that you get at least one tail is therefore $1 - \left(\frac{1}{2}\right)^{10}$. The probability that each of the 1000 coins comes up tails at least once is then

$$\left(1 - \left(\frac{1}{2}\right)^{10}\right)^{1000} \approx 0.37642 .$$

We want the probability of the complementary event, which is therefore about 0.62357.

Another Solution

If the probability that ``some coin will get 10 heads'' is the probability that exactly 1 of the 1000 coins shows up heads on all 10 tosses, then we're looking at

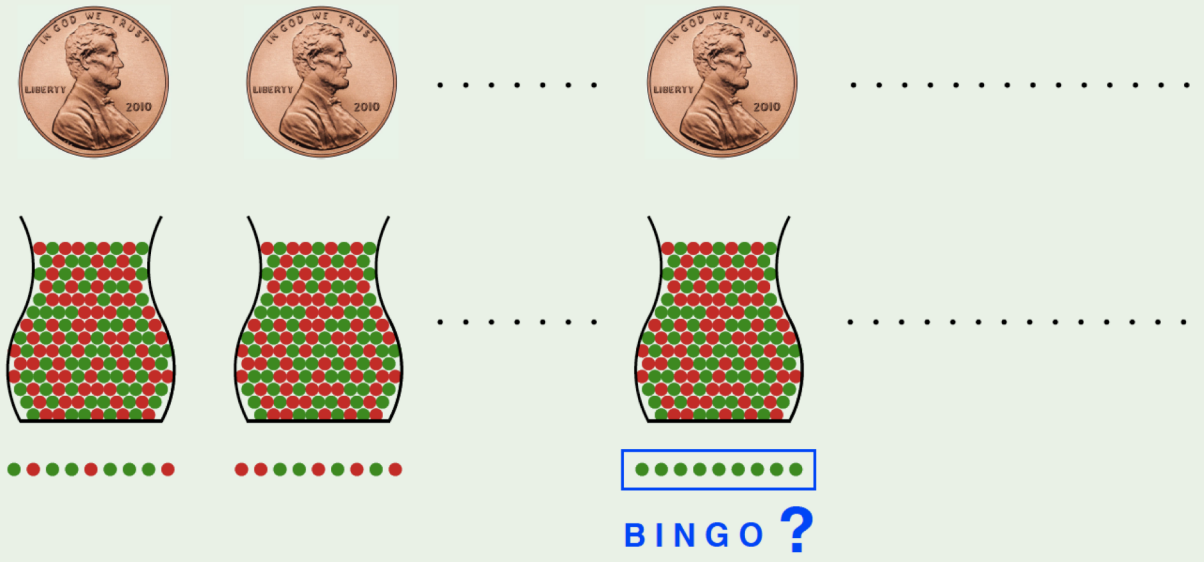
$$\mathrm{P}\{Y = 1\} = \binom{1000}{1}\left(\frac{1}{2^{10}}\right)^{1}\left(1 - \frac{1}{2^{10}}\right)^{999} .$$

If the probability that ``some coin will get 10 heads'' is the probability that at least 1 of the 1000 coins shows up heads on all 10 tosses, then we're looking at

$$\mathrm{P}\{Y \geq 1\} = 1 - \mathrm{P}\{Y = 0\} = 1 - \binom{1000}{0}\left(\frac{1}{2^{10}}\right)^{0}\left(1 - \frac{1}{2^{10}}\right)^{1000} = 1 - \left(1 - \frac{1}{2^{10}}\right)^{1000} ,$$

which is approximately 0.624.

From coins to learning



The way to get around this is to try to bound $\mathbb{P}[|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon]$ in a way that does not depend on which g the learning algorithm picks. There is a simple but crude way of doing that. Since g has to be one of the h_m's regardless of the algorithm and the sample, it is always true that

$$\text{“}|E_{\text{in}}(g) - E_{\text{out}}(g)| > \epsilon\text{”} \implies \text{“} \begin{array}{l} |E_{\text{in}}(h_1) - E_{\text{out}}(h_1)| > \epsilon \\ \textbf{or } |E_{\text{in}}(h_2) - E_{\text{out}}(h_2)| > \epsilon \\ \cdots \\ \textbf{or } |E_{\text{in}}(h_M) - E_{\text{out}}(h_M)| > \epsilon \text{”}. \end{array}$$

A simple solution

- Apply two basic rules in probability

$$\text{if } \mathcal{B}_1 \implies \mathcal{B}_2, \text{ then } \mathbb{P}[\mathcal{B}_1] \leq \mathbb{P}[\mathcal{B}_2],$$

and, if $\mathcal{B}_1, \mathcal{B}_2, \cdots, \mathcal{B}_M$ are any events, then

$$\mathbb{P}[\mathcal{B}_1 \textbf{ or } \mathcal{B}_2 \textbf{ or } \cdots \textbf{ or } \mathcal{B}_M] \leq \mathbb{P}[\mathcal{B}_1] + \mathbb{P}[\mathcal{B}_2] + \cdots + \mathbb{P}[\mathcal{B}_M].$$

The second rule is known as the *union bound*. Putting the two rules together, we get

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

A simple solution

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

The final verdict

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

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

Feasibility of Learning

- Question: Does *D* tell us anyhting outside of *D?*
  - If we insist on a deterministic answer, which means that *D* tells us something certain about f outside of *D*, then the answer is NO.
  - If we accept a probabilistic answer, which means that *D* tells us something likely about *f* outside of *D*, then the answer is YES.
    - The only assumption we make is that the examples in *D* are generated independently.

Feasibility of Learning

The feasibility of learning is thus split into two questions:

1. Can we make sure that $E_{\text{out}}(g)$ is close enough to $E_{\text{in}}(g)$?
2. Can we make $E_{\text{in}}(g)$ small enough?

The Hoeffding Inequality (1.6) addresses the first question only. The second question is answered after we run the learning algorithm on the actual data and see how small we can get E_{in} to be.