Introduction to Machine Learning (CSCI-UA 473): Fall 2021

Lecture 2: Feasibility of Learning

Sumit Chopra
Courant Institute of Mathematical Sciences
Department of Radiology - Grossman School of Medicine
NYU

Slides derived from materials from Benjamin Peherstorfer, Kyunghyun Cho, Andrew Gordon Wilson

Lecture Outline

Recap from last lecture

Feasibility of learning

Why is the feasibility of learning questionable?

The bin experiment

Hoeffding Inequality

Two primary questions in learning

How to answer these questions in practice

Error measures (loss functions)

Noisy targets

Components of a Learning System

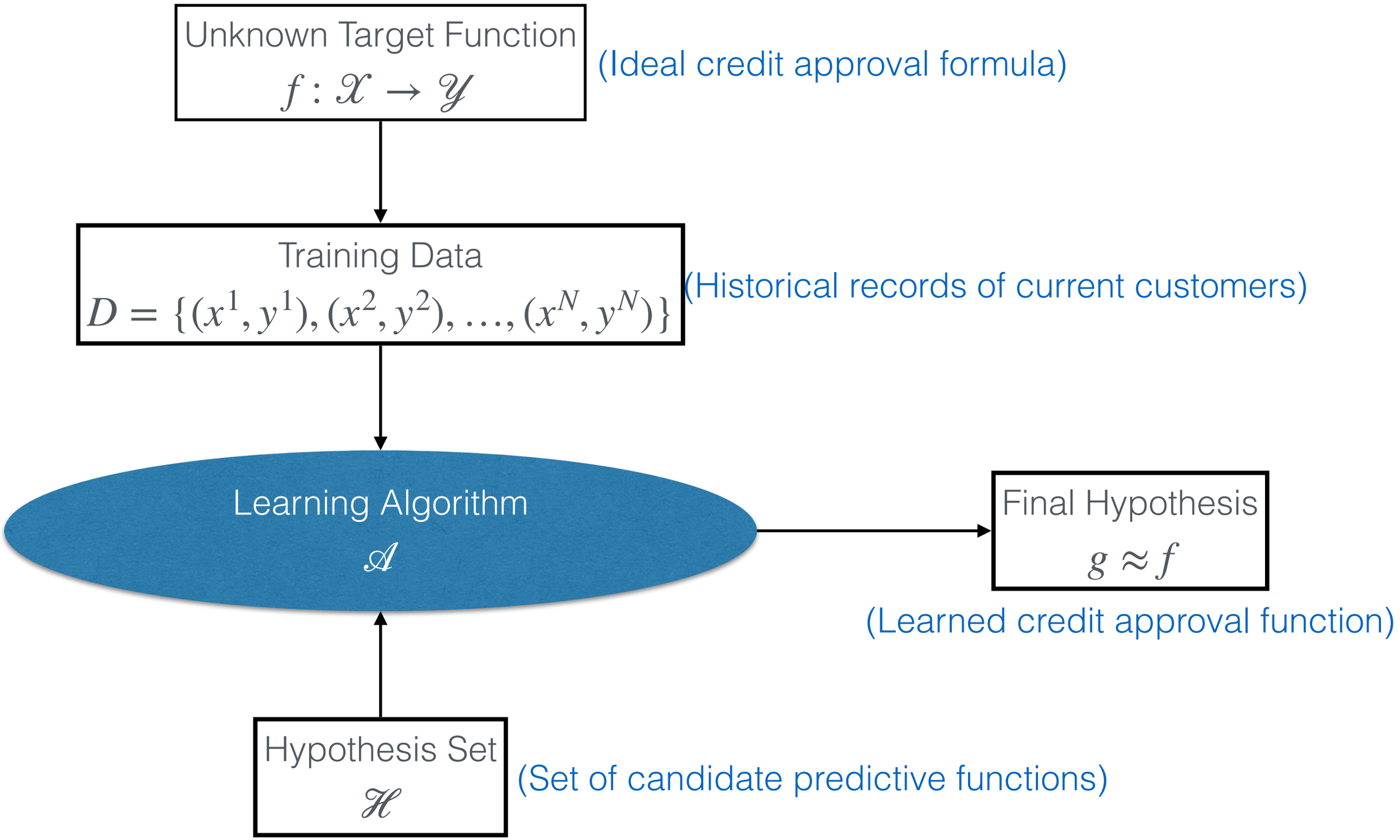


Figure adapted from the book Learning from Data: A Short Course

Components of a Learning System

Example Problem Setting

You are a bank and you receive thousands of credit card applications everyday. For every application you want to answer two questions:

1. Whether to extend a credit card to the applicant?
2. If the answer is “yes” then what should be the credit limit?

You have lots of historical data: your current customers, their detailed information, whether or not you’ve made money from them after offering the credit card, and how much money you’ve made.

Recap

Components of a Learning System

Customer information (age, gender, income etc): x (the Input)

Binary credit decision (you made money or not): y (the output)

Unknown target function: $f : \mathcal{X} \rightarrow \mathcal{Y}$

$\mathcal{X}$: the entire space of inputs

$\mathcal{Y}$: the entire space of outputs (binary "yes/no" in this case)

Customer data (input-output pairs): $D = \{(x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)\}$ **(training data)**

The data is such that: $y^i = f(x^i) : \forall i$

The goal of learning is to use the dataset D to find a function $g : \mathcal{X} \rightarrow \mathcal{Y}$ such that:

g approximates the function f

Learning algorithm chooses g from a set of functions called Hypothesis Set $\mathcal{H}$

For a new customer the learning algorithm will base its decision on the output of g

The decision will only makes so long as the learnt g is faithful to f on the training data

A Simple Learning Model

$$h(x) = sign\left(\left(\sum_{j=1}^{d} w_j x_j\right) + b\right)$$

$x_1, x_2, \ldots x_d$ are the components of the vector x

$h(x) = +1$ implies approve credit; $h(x) = -1$ implies reject credit

$sign(s) = +1$ if $s > 0$; $sign(s) = -1$ if $s < 0$

The weights (parameters) are $w_1, w_2, \ldots, w_d$

The *threshold* is determined by the bias b since according to the equation on previous slide, credit is approved if

$$\sum_{i=1}^{d} w_i x_i > -b$$

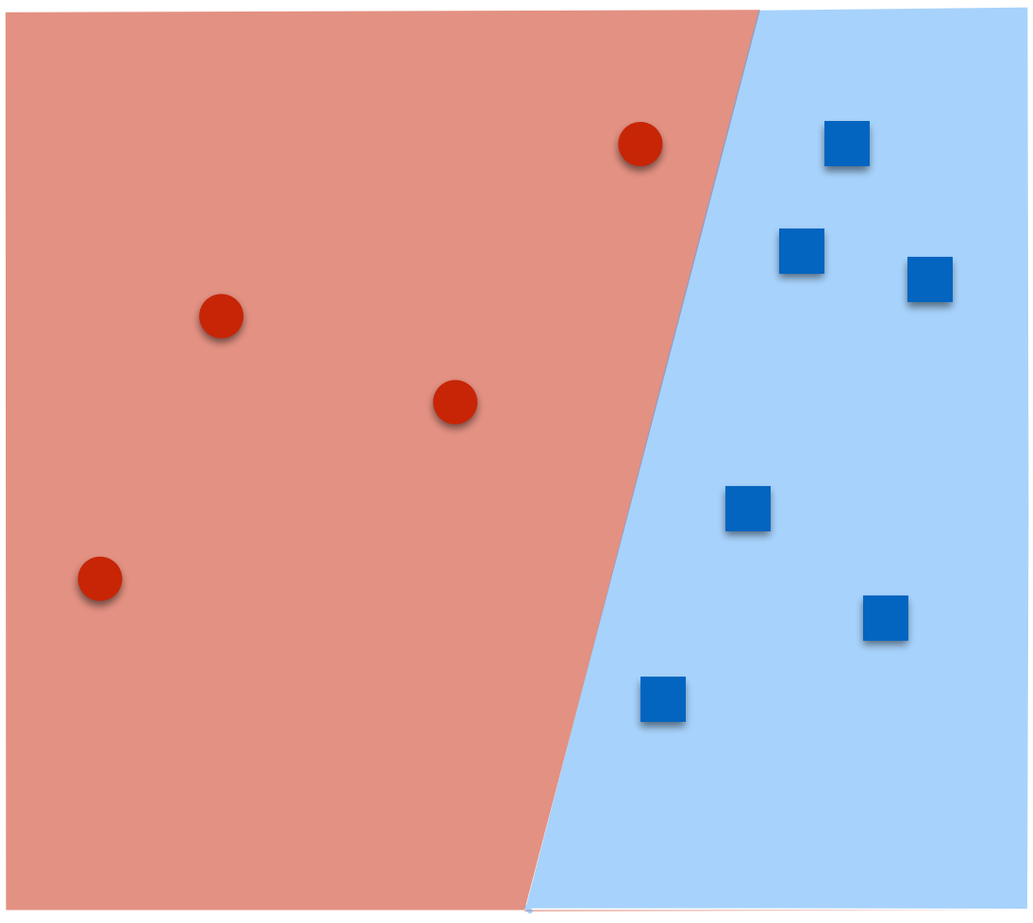
This simple model is called the ***Perceptron***

Recap

Trained Using Perceptron Learning Algorithm (PLA)

$$h(x) = sign\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}\right)$$

$$\mathbf{w} = [b, w_1, w_2, \ldots, w_d] \;\; and \;\; \mathbf{x} = [1, x_1, x_2, \ldots, x_d]$$



Learned Weight Vector

Let $w(t)$ be the weight vector at iteration t for $t = 0$ until no example is misclassified

1. Pick a random sample $(x(t), y(t))$ from the set D which is misclassified
2. Update the weight vector with the following update rule

$$w(t+1) \leftarrow w(t) + y(t)x(t)$$

Note that since the example is misclassified

$$y(t) \neq sign(w(t)^T x(t))$$

One can prove that so long as the data is linearly separable the above algorithm will find a separating hyperplane

Recap Trained Using Perceptron Learning Algorithm (PLA)

$$h(x) = sign\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}\right)$$

$$\mathbf{w} = [b, w_1, w_2, \ldots, w_d] \;\; and \;\; \mathbf{x} = [1, x_1, x_2, \ldots, x_d]$$

While we have fit a model on the given data (a.k.a. training data), what we care about is the model's answers on "new" (unseen) data.

Let $w(t)$ be the weight vector at iteration t

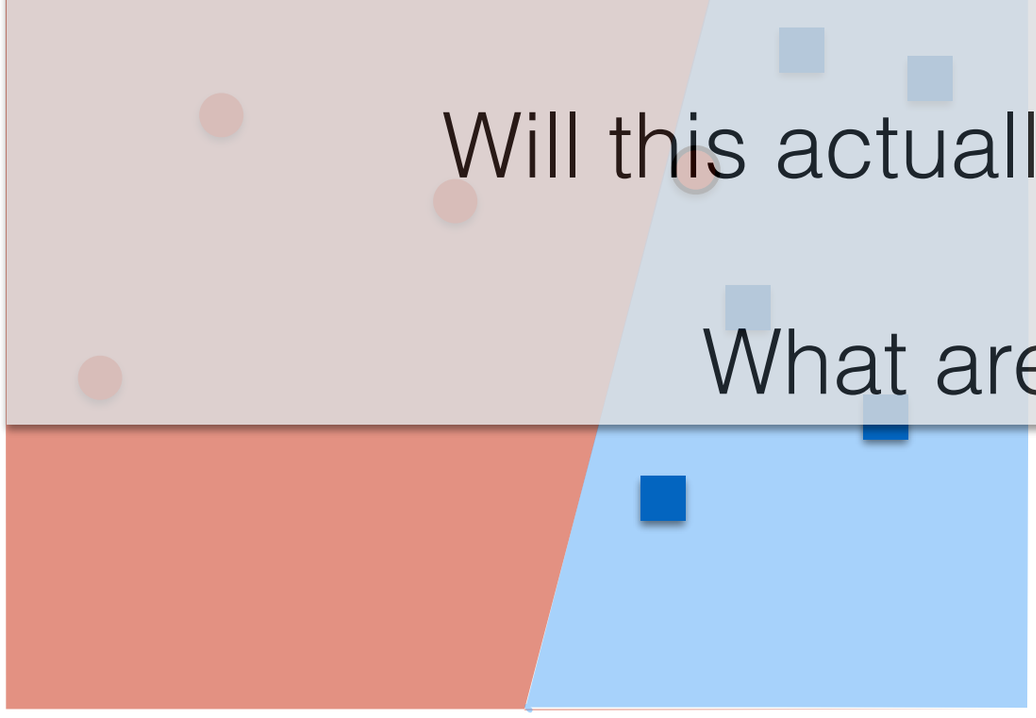
for $t = 0$ until no example is misclassified

1. Pick a random sample $(x(t), y(t))$ from the set D which is misclassified
2. Update the weight vector with the following update rule

$$w(t+1) \leftarrow w(t) + y(t)x(t)$$

Will this actually work on unseen data?

What are the guarantees?



Learned Weight Vector

Note that since the example is misclassified

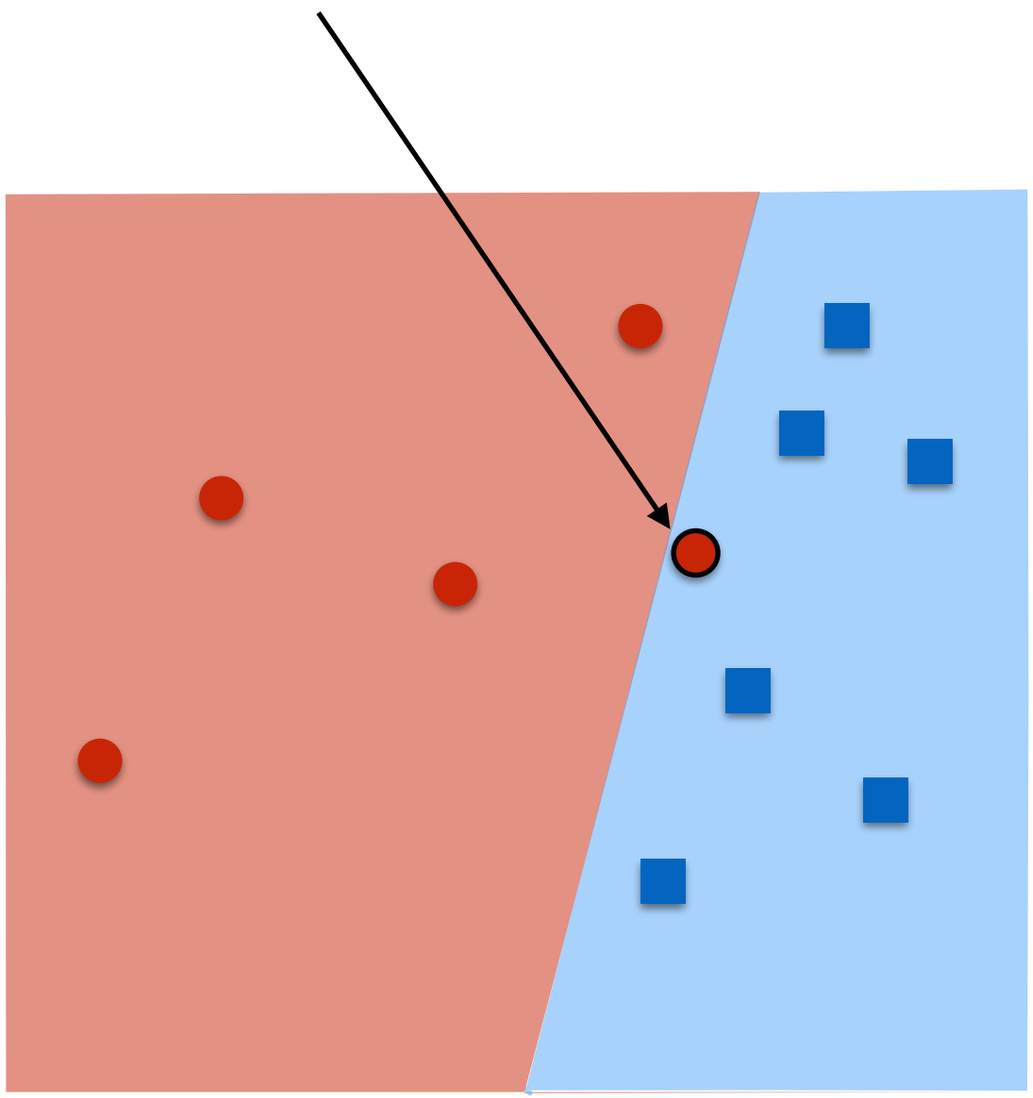
$$y(t) \neq sign(w(t)^T x(t))$$

One can prove that so long as the data is linearly separable the above algorithm will find a separating hyperplane

Recap Trained Using Perceptron Learning Algorithm (PLA)

$$h(x) = sign\left(\mathbf{w}^{\mathbf{T}}\mathbf{x}\right)$$

$$\mathbf{w} = [b, w_1, w_2, \ldots, w_d] \;\; and \;\; \mathbf{x} = [1, x_1, x_2, \ldots, x_d]$$



Let $w(t)$ be the weight vector at iteration t

for $t = 0$ until no example is misclassified

1. Pick a random sample $(x(t), y(t))$ from the set D which is misclassified
2. Update the weight vector with the following update rule

$$w(t+1) \leftarrow w(t) + y(t)x(t)$$

Feasibility of Learning

	x	y	g
1	0 0 0	○	
2	0 0 1	●	
3	0 1 0	●	
4	0 1 1	○	
5	1 0 0	●	
6	1 0 1		
7	1 1 0		
8	1 1 1		

$$D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$$

Consider a boolean target function over a three dimensional boolean space:

$$\mathcal{X} = \{0,1\}^3$$

We are given the value of the unknown function f on the training set D

$$y_i = f(x_i) : i = 1,\ldots,5$$

Feasibility of Learning

	x	y	g
1	0 0 0	○	
2	0 0 1	●	
3	0 1 0	●	
4	0 1 1	○	
5	1 0 0	●	
6	1 0 1		
7	1 1 0		
8	1 1 1		

We want to find a function g which is faithful to f both at the seen AND unseen points

By definition we do not know what the unseen points are and hence we don't know what f would look like on them

$$D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$$

Feasibility of Learning

	x	y	g
1	0 0 0	○	
2	0 0 1	●	
3	0 1 0	●	
4	0 1 1	○	
5	1 0 0	●	
6	1 0 1		
7	1 1 0		
8	1 1 1		

$$D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$$

$$\mathscr{H} = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$$

We want to find a function g which is faithful to f both at the seen AND unseen points

By definition we do not know what the unseen points are and hence we don't know what f would look like on them

Let us define a hypothesis set $\mathscr{H}$ of all the boolean functions over the three boolean variables

There are $2^{2^3} = 256$ distinct functions

Out of these only $2^3 = 8$ functions agree with training set D

Feasibility of Learning

	x	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
						$\mathscr{H}$					
1	0 0 0	○		○	○	○	○	○	○	○	○
2	0 0 1	●		●	●	●	●	●	●	●	●
3	0 1 0	●		●	●	●	●	●	●	●	●
4	0 1 1	○		○	○	○	○	○	○	○	○
5	1 0 0	●		●	●	●	●	●	●	●	●
6	1 0 1										
7	1 1 0										
8	1 1 1										

$$D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$$

$$\mathscr{H} = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$$

Feasibility of Learning

	x	y	g	$\mathscr{H}$: f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	0 0 0	○		○	○	○	○	○	○	○	○
2	0 0 1	●		●	●	●	●	●	●	●	●
3	0 1 0	●		●	●	●	●	●	●	●	●
4	0 1 1	○		○	○	○	○	○	○	○	○
5	1 0 0	●		●	●	●	●	●	●	●	●
6	1 0 1			○	○	○	○	●	●	●	●
7	1 1 0			○	○	●	●	○	○	●	●
8	1 1 1			○	●	○	●	○	●	○	●

$$D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$$

$$\mathscr{H} = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$$

They all are consistent with each other and agree with f on the training set.

But completely disagree with each other on the unseen examples

Feasibility of Learning

				$\mathscr{H}$							
	x	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	0 0 0	○	○	○	○	○	○	○	○	○	○
2	0 0 1	●	●	●	●	●	●	●	●	●	●
3	0 1 0	●	●	●	●	●	●	●	●	●	●
4	0 1 1	○	○	○	○	○	○	○	○	○	○
5	1 0 0	●	●	●	●	●	●	●	●	●	●
6	1 0 1		?	○	○	○	○	●	●	●	●
7	1 1 0		?	○	○	●	●	○	○	●	●
8	1 1 1		?	○	●	○	●	○	●	○	●

$D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$

$\mathscr{H} = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$

Which one to choose as g?

They all are consistent with each other and agree with f on the training set.

But completely disagree with each other on the unseen examples

Feasibility of Learning

	x	y	g	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
1	0 0 0	○	○	○	○	○	○	○	○	○	○
2	0 0 1	●	●	○	○	○	○	●	●	●	●
3	0 1 0	●	●	○	○	●	●	○	○	●	●
4	0 1 1	○	○	○	○	●	●	●	●	○	○
5	1 0 0	●	●	○	●	○	●	○	●	○	●
6	1 0 1	?	○	○	●	○	●	○	●	○	●
7	1 1 0	?	○	●	○	○	●	●	○	○	●
8	1 1 1	?	○	●	○	●	○	●	○	●	○

No matter what the hypothesis space or the function you choose **based on its performance on D**, it makes no difference whatsoever as far as the performance outside of D is concerned.

Yet, all we care about is the performance of the chosen hypothesis outside of D!

$$D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5)\}$$

$$\mathcal{H} = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$$

Which of the functions to choose as g?

They all are consistent with each other and agree with f on the training set.

But completely disagree with each other on the unseen examples

Does this Mean that Learning Doomed?

Does this Mean that Learning Doomed?

The Answer is No!

One needs to make certain assumptions: how the data samples are drawn, what hypothesis set to choose, what the hypothesis is capable of accomplishing, how to find the best hypothesis etc

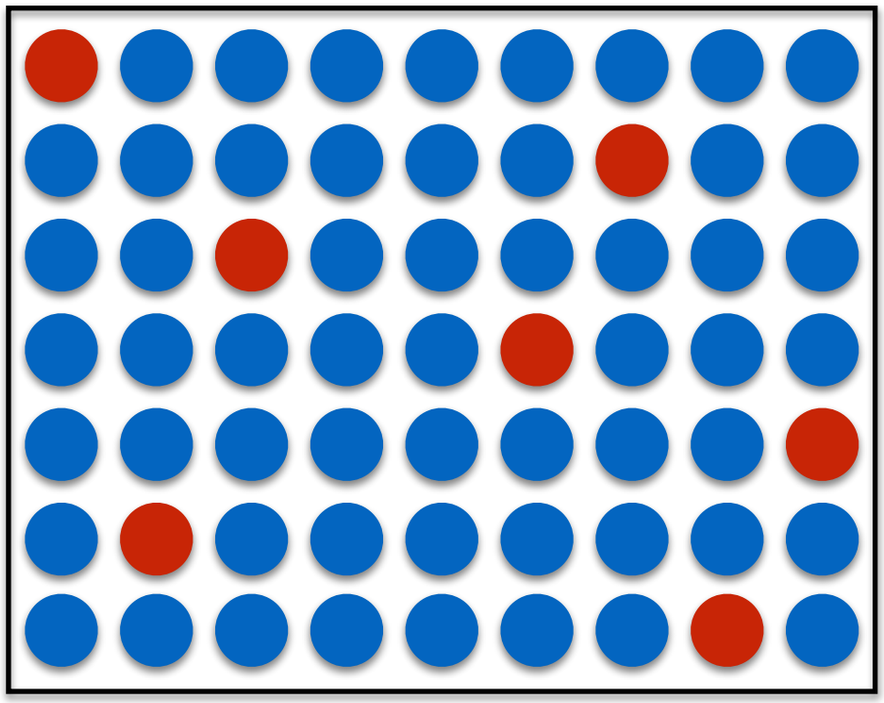
Probability is the savior here

This question is formally discussed under the topic of “Learning Theory”

No matter what we do, we cannot claim that we have found a function g which deterministically replicates the performance of f

We can only give probability estimates: “the function g **approximates** the behavior of function f on unseen examples **with certain probability** so long as the unseen examples are drawn in the same way as the seen examples”

A Simple Experiment



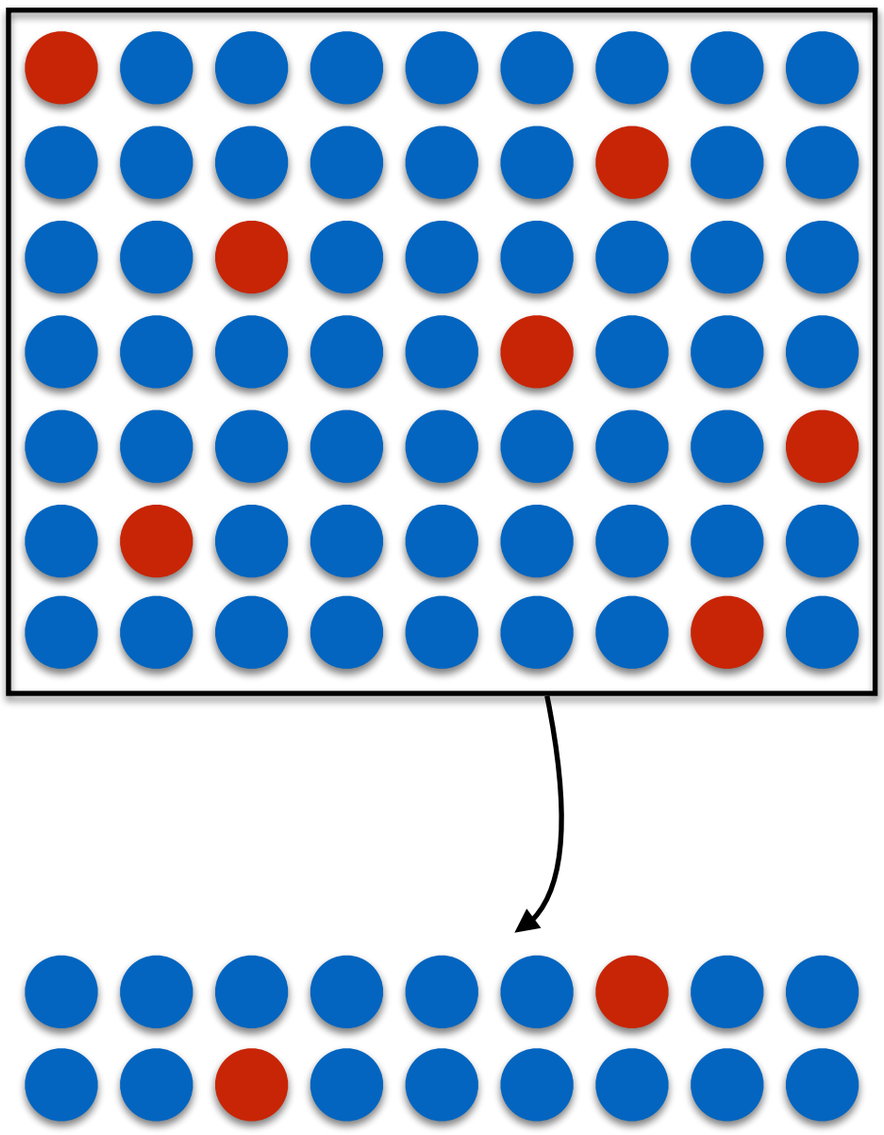
P('picking a red ball') = μ

P('picking a blue ball') = $1 - \mu$

The value of μ is **fixed but unknown**

Figure adapted from the book Learning from Data: A Short Course

A Simple Experiment



P('picking a red ball') = μ

P('picking a blue ball') = $1 - \mu$

The value of μ is **fixed but unknown**

Draw a random ball from the bin, record its color and replace it

Repeat this experiment N times

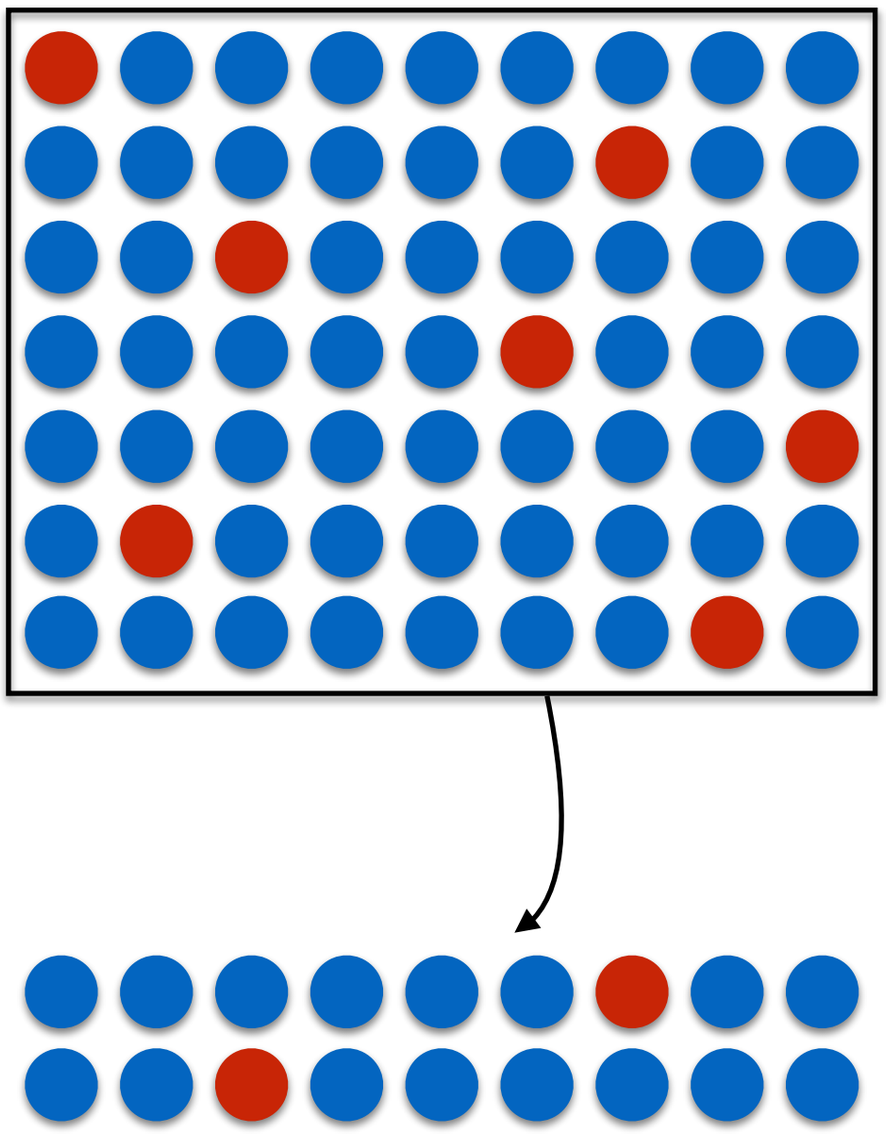
What you get is a **random sample** of the data

Let ν denote the fraction of red balls in sample of size N

We don't know μ however we do know ν

Figure adapted from the book Learning from Data: A Short Course

A Simple Experiment



P('picking a red ball') = μ

P('picking a blue ball') = $1 - \mu$

The value of μ is **fixed but unknown**

Draw a random ball from the bin, record its color and replace it

Repeat this experiment N times

What you get is a **random sample** of the data

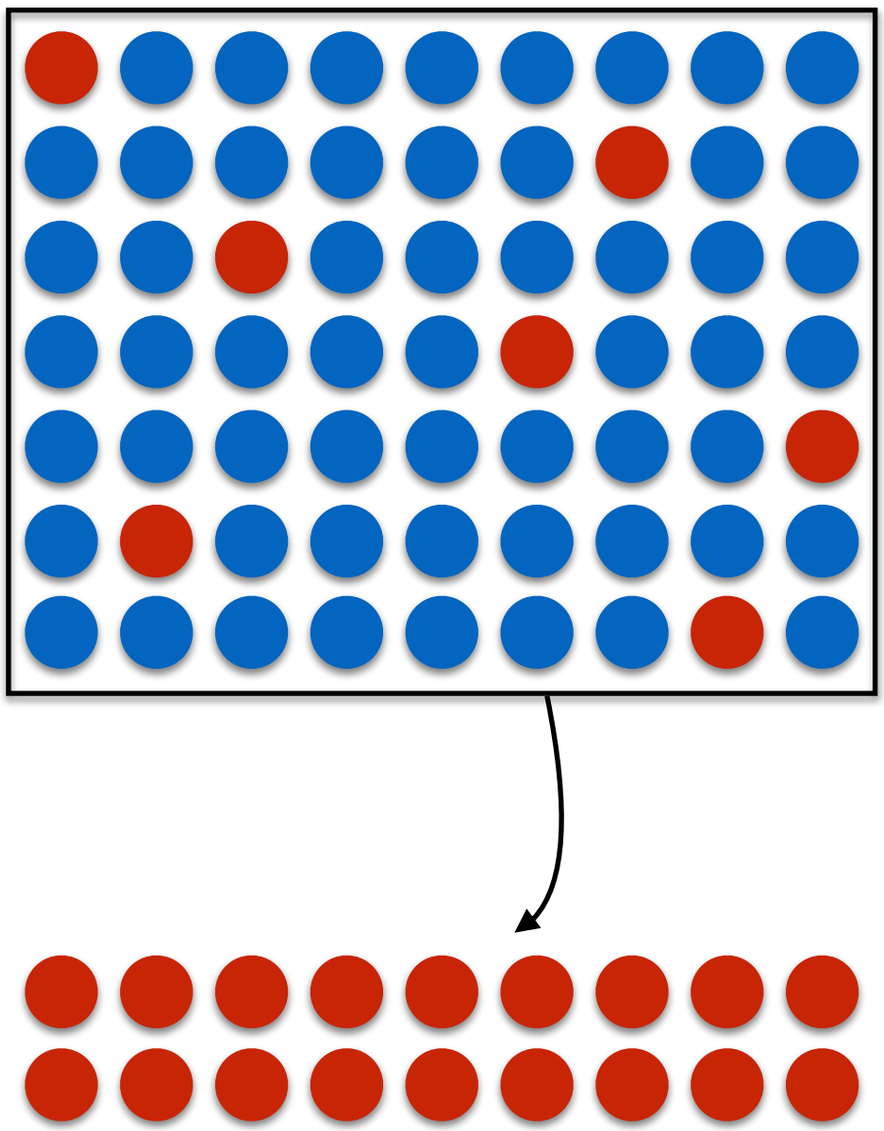
Let ν denote the fraction of red balls in sample of size N

We don't know μ however we do know ν

Can we say anything about μ given the fact that we know ν?

Figure adapted from the book Learning from Data: A Short Course

A Simple Experiment



P('picking a red ball') = μ; P('picking a blue ball') = $1 - \mu$

Can we say anything about μ given the fact that we know ν?

One Scenario

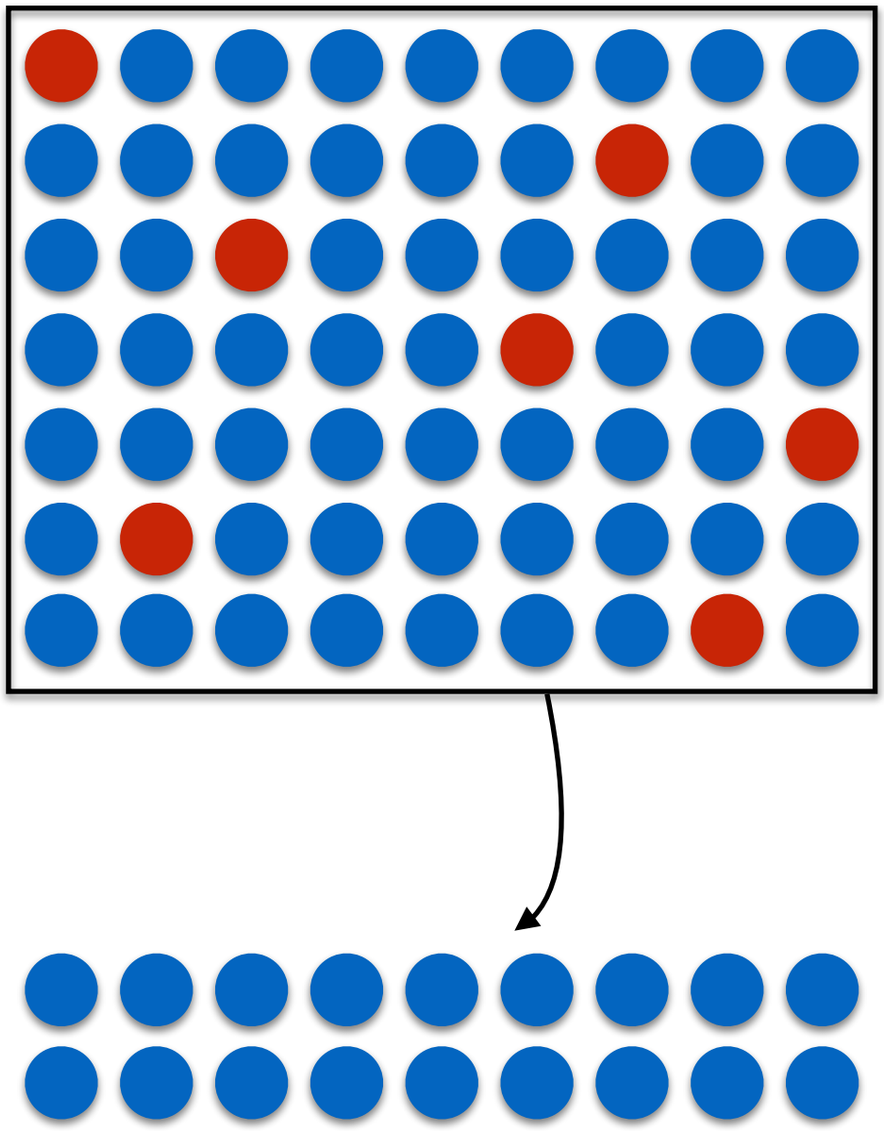
All the balls are red in color

Possible but highly unlikely

Let ν denote the fraction of red balls in sample of size N

Figure adapted from the book Learning from Data: A Short Course

A Simple Experiment



Let ν denote the fraction of red balls in sample of size N

P('picking a red ball') = μ; P('picking a blue ball') = $1 - \mu$

Can we say anything about μ given the fact that we know ν?

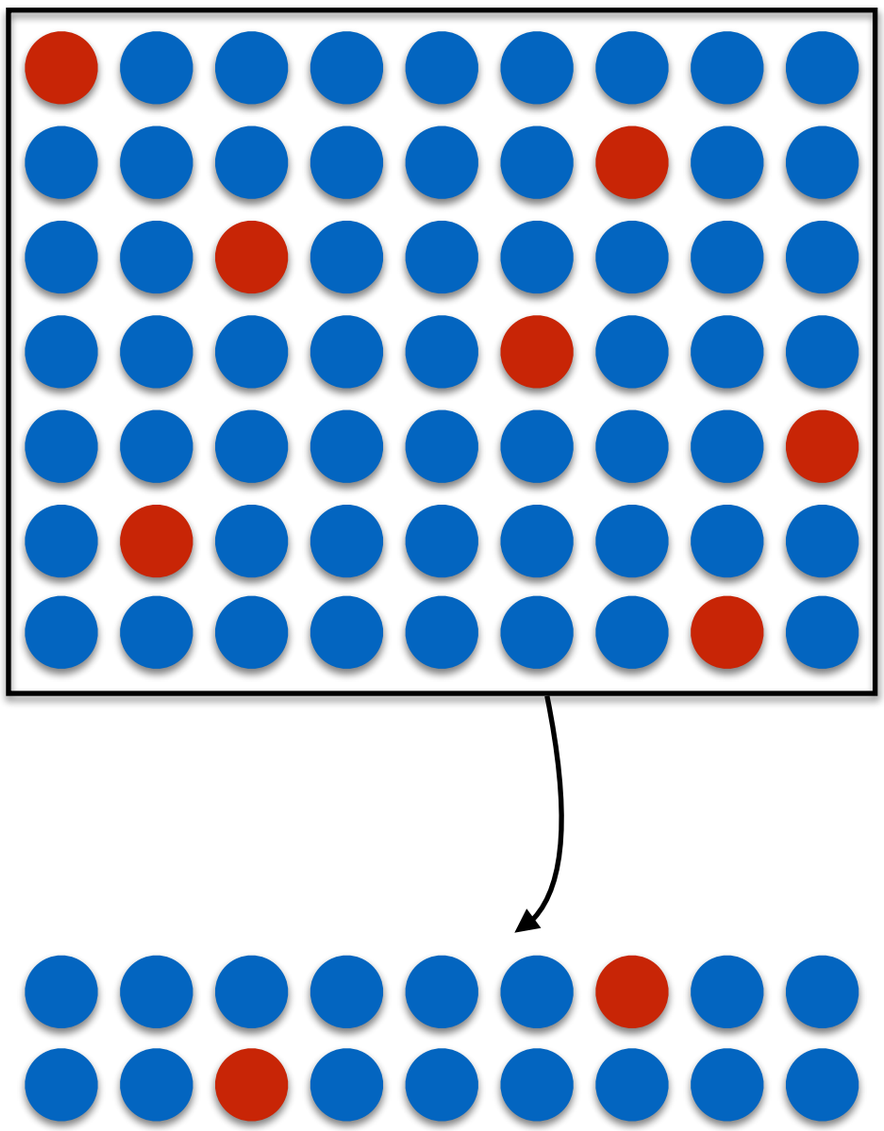
Another Scenario

All balls are blue in color

Also possible but less likely

Figure adapted from the book Learning from Data: A Short Course

A Simple Experiment



P('picking a red ball') = μ; P('picking a blue ball') = $1 - \mu$

Can we say anything about μ given the fact that we know ν?

Another Scenario

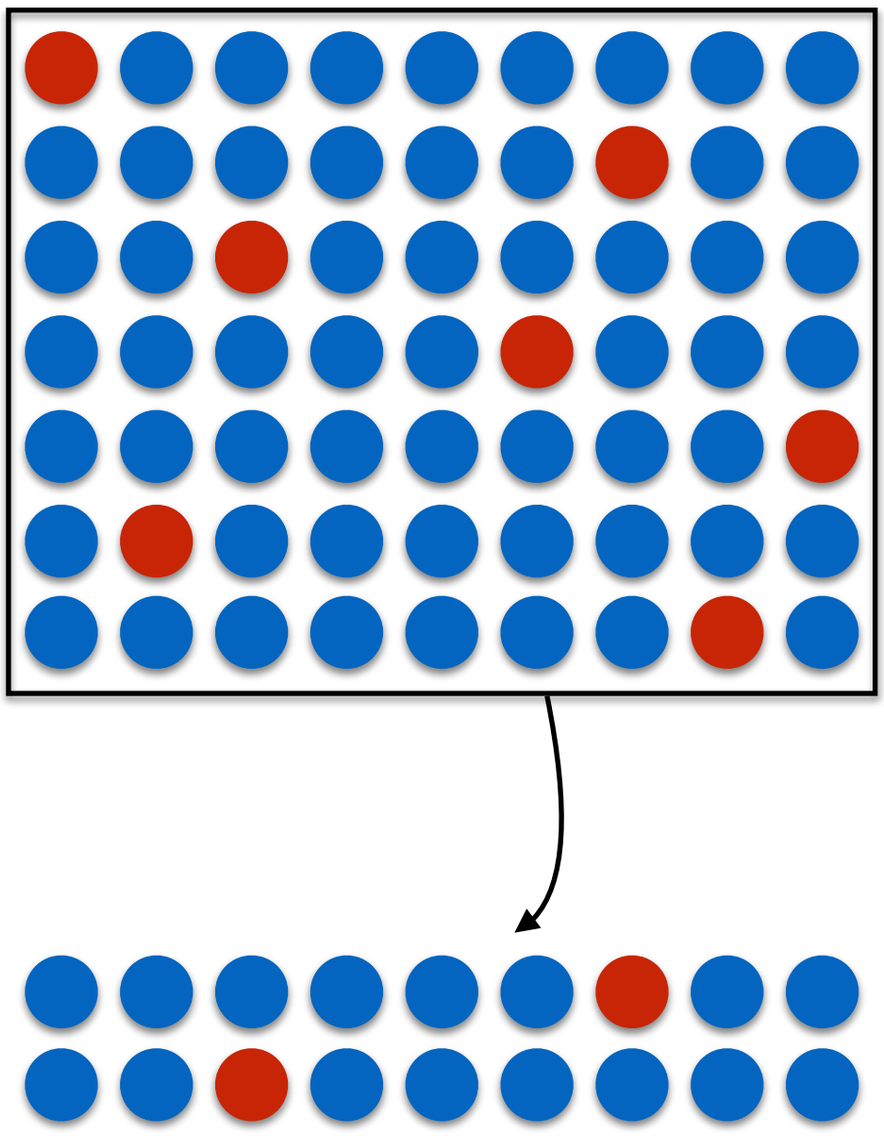
The distribution of the ball colors of the sample is similar to the distribution of the ball colors of the bin

Possible and highly likely

Let ν denote the fraction of red balls in sample of size N

Figure adapted from the book Learning from Data: A Short Course

A Simple Experiment



Let ν denote the fraction of red balls in sample of size N

P('picking a red ball') = μ; P('picking a blue ball') = $1 - \mu$

Can we say anything about μ given the fact that we know ν?

More Formally

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

This is called the Hoeffding Inequality

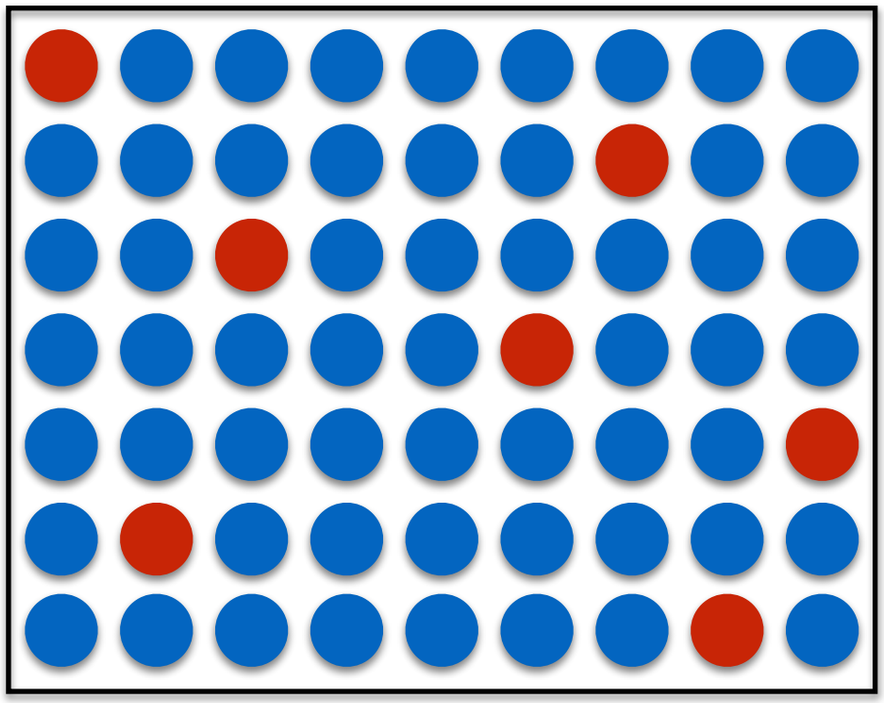
In words, it says that for a large sample N, the probability of μ and ν being close to each other is high

There is a tradeoff between the sample size N and the closeness of approximation ϵ

Note that the bound does not depend on μ

Figure adapted from the book Learning from Data: A Short Course

How is this Connected to Learning?



Unknown quantity in the bin experiment: μ

Unknown quantity in the Learning setting: $f : \mathcal{X} \rightarrow \mathcal{Y}$

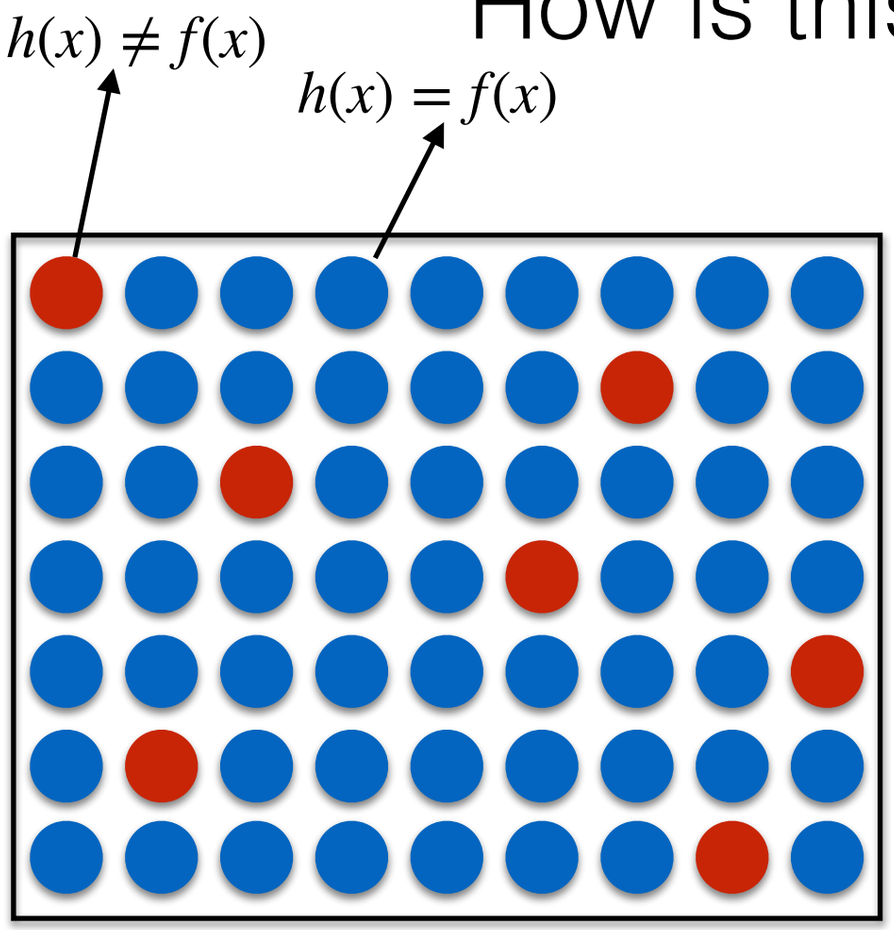
Assume each ball is one data point $x \in \mathcal{X}$

Then the bin is the entire space of inputs $\mathcal{X}$

Consider some pre-selected hypothesis $h \in \mathcal{H}$

Figure adapted from the book Learning from Data: A Short Course

How is this Connected to Learning?



Unknown quantity in the bin experiment: μ

Unknown quantity in the Learning setting: $f : \mathcal{X} \rightarrow \mathcal{Y}$

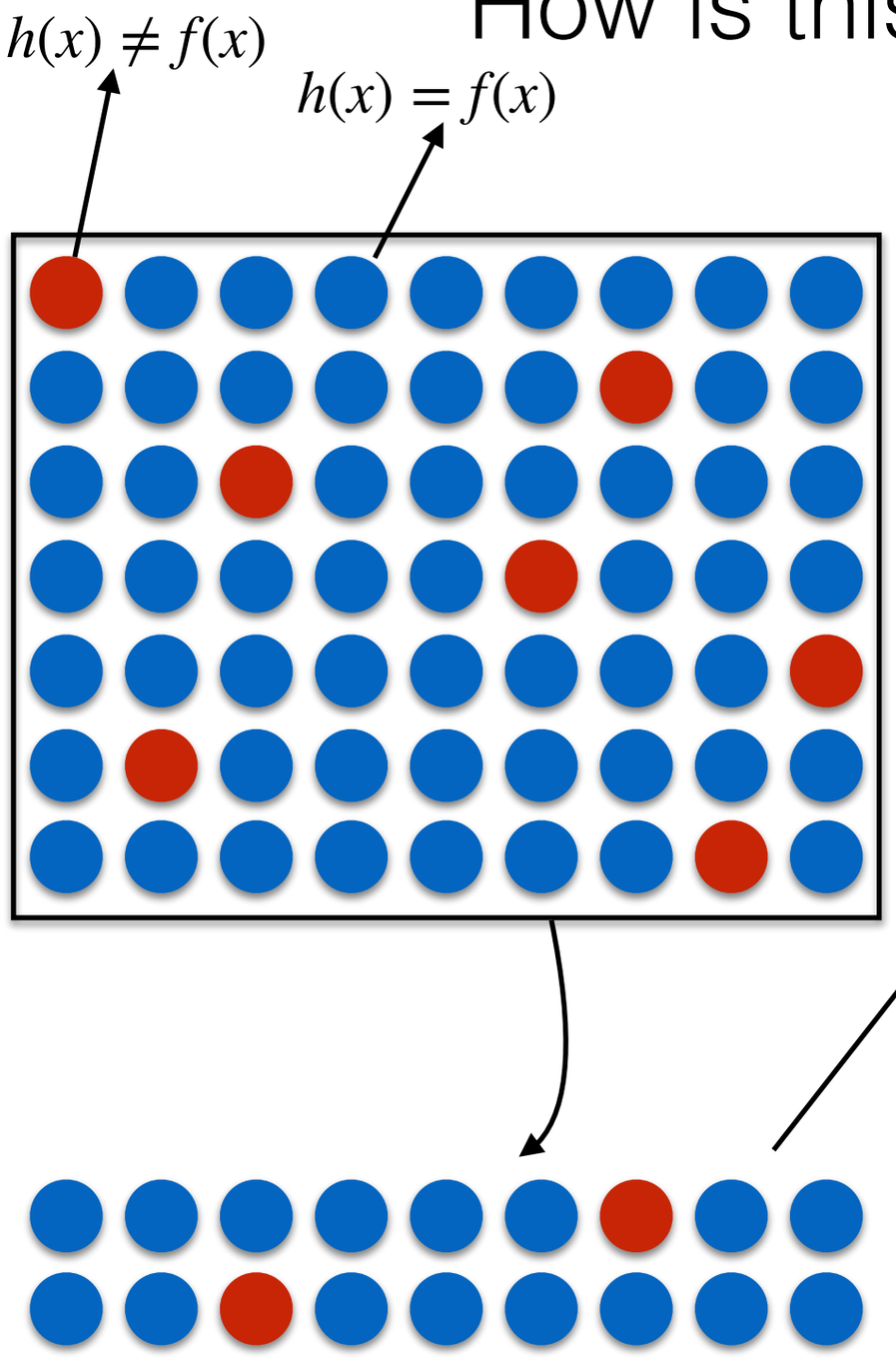
Assume each ball is one data point $x \in \mathcal{X}$

Then the bin is the entire space of inputs $\mathcal{X}$

Consider some pre-selected hypothesis $h \in \mathcal{H}$

Figure adapted from the book Learning from Data: A Short Course

How is this Connected to Learning?



Let ν denote the fraction of red balls in sample of size N

Unknown quantity in the bin experiment: μ

Unknown quantity in the Learning setting: $f : \mathcal{X} \rightarrow \mathcal{Y}$

Assume each ball is one data point $x \in \mathcal{X}$

Then the bin is the entire space of inputs $\mathcal{X}$

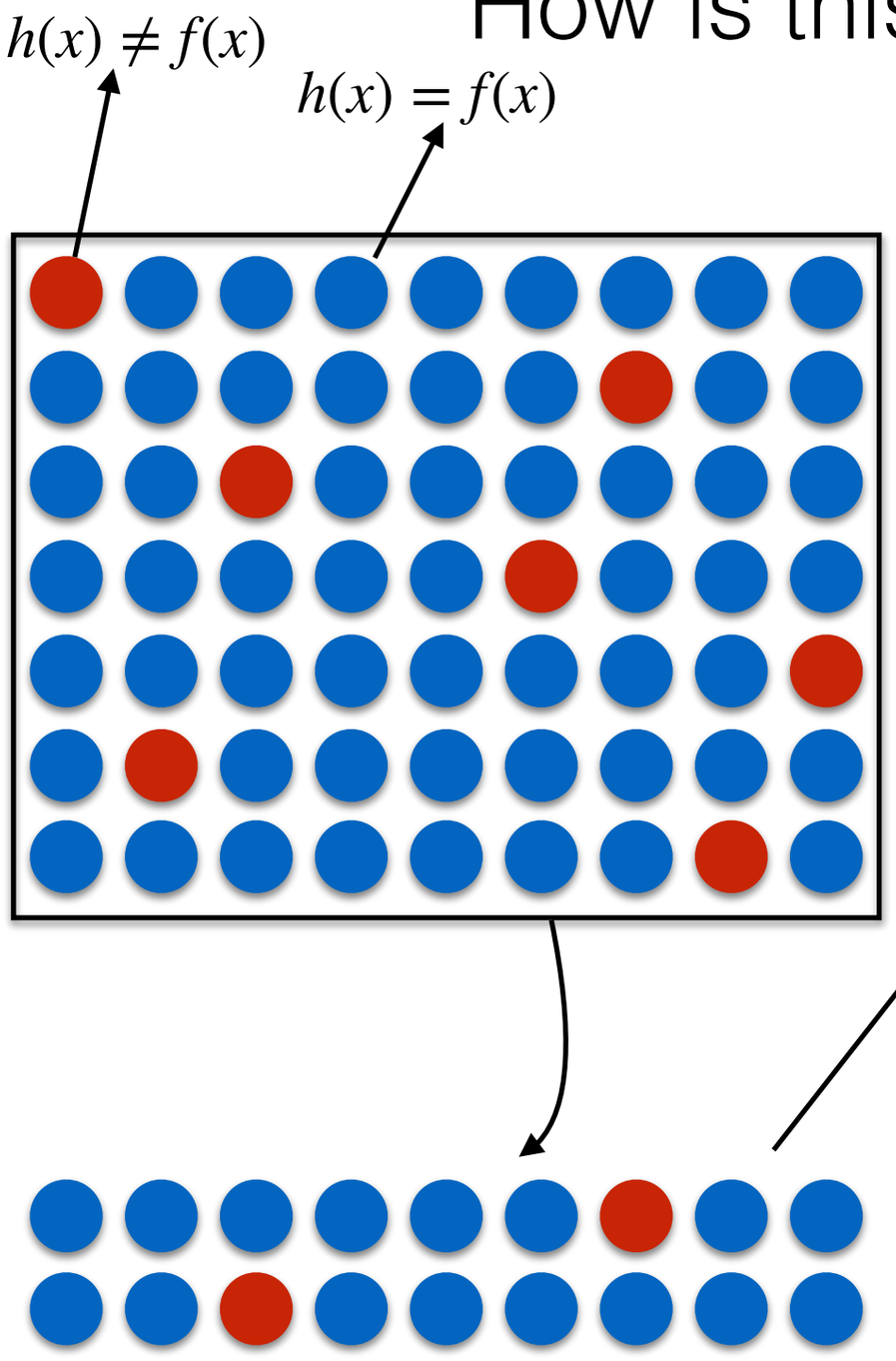
Consider some pre-selected hypothesis $h \in \mathcal{H}$

Then the drawn sample can be thought of as the training set whose data points are drawn independently using some probability distribution P

ν is the fraction of the samples in the training set for which the hypothesis h differs from the actual function f

Figure adapted from the book Learning from Data: A Short Course

How is this Connected to Learning?



Let ν denote the fraction of red balls in sample of size N

Unknown quantity in the bin experiment: μ

Unknown quantity in the Learning setting: $f : \mathcal{X} \rightarrow \mathcal{Y}$

Assume each ball is one data point $x \in \mathcal{X}$

Then the bin is the entire space of inputs $\mathcal{X}$

Consider some pre-selected hypothesis $h \in \mathcal{H}$

Then the drawn sample can be thought of as the training set whose data points are drawn independently using some probability distribution P

ν is the fraction of the samples in the training set for which the hypothesis h differs from the actual function f

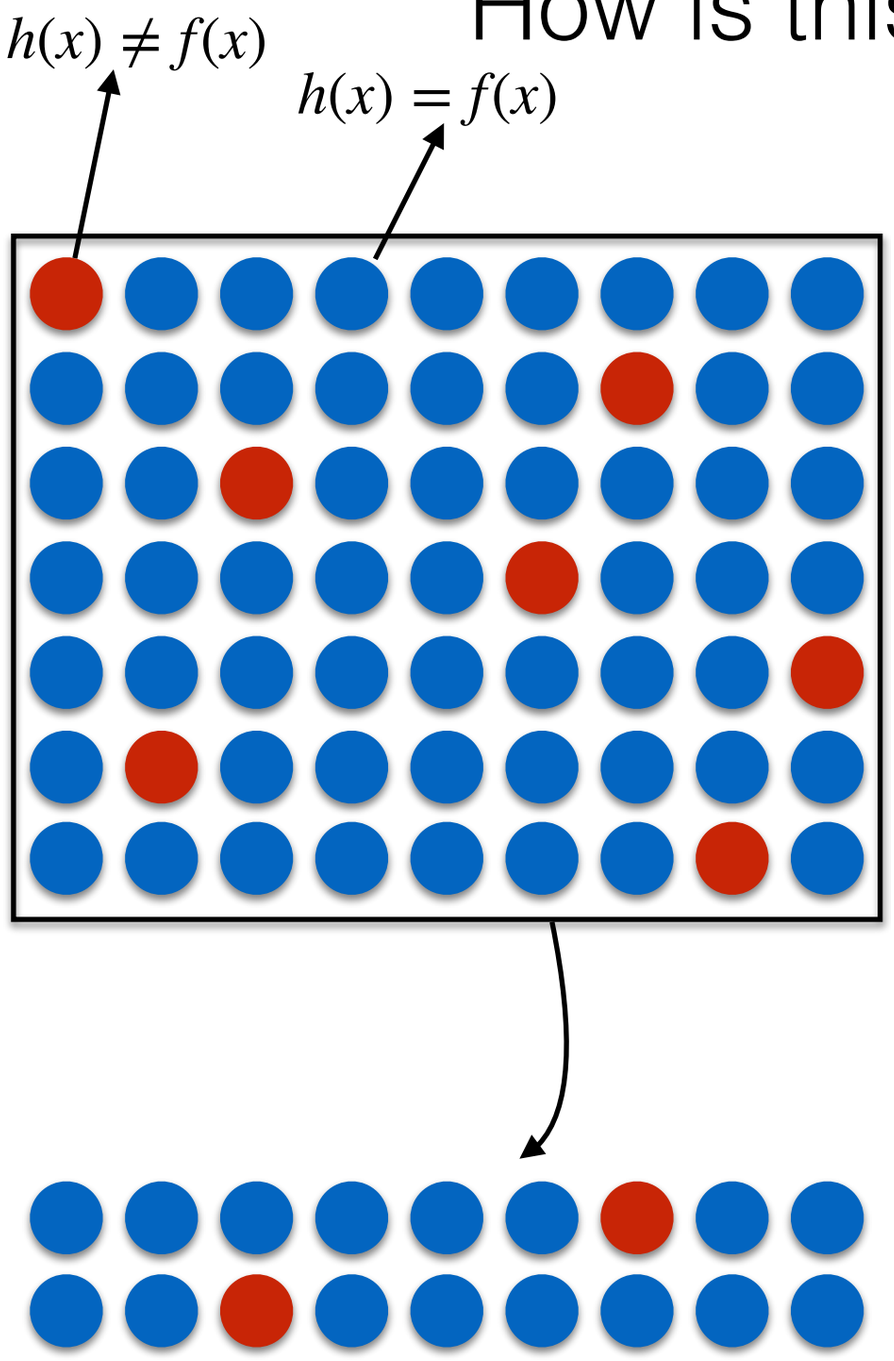
Now we can provide a more descriptive notation:

$$\mu = \mathbb{P}[h(x) \neq f(x)] \equiv E_{out}(h)$$ Out-of-sample error

$$\nu = \frac{1}{N} \sum_{i=1}^{N} [h(x_i) \neq f(x_i)] \equiv E_{in}(h)$$ In-sample error

Figure adapted from the book Learning from Data: A Short Course

How is this Connected to Learning?



Let ν denote the fraction of red balls in sample of size N

$$\mu \equiv E_{out}(h) = \mathbb{P}[h(x) \neq f(x)]$$ Out-of-sample error

$$\nu \equiv E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N}[h(x_i) \neq f(x_i)]$$ In-sample error

Then the Hoeffding Inequality

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

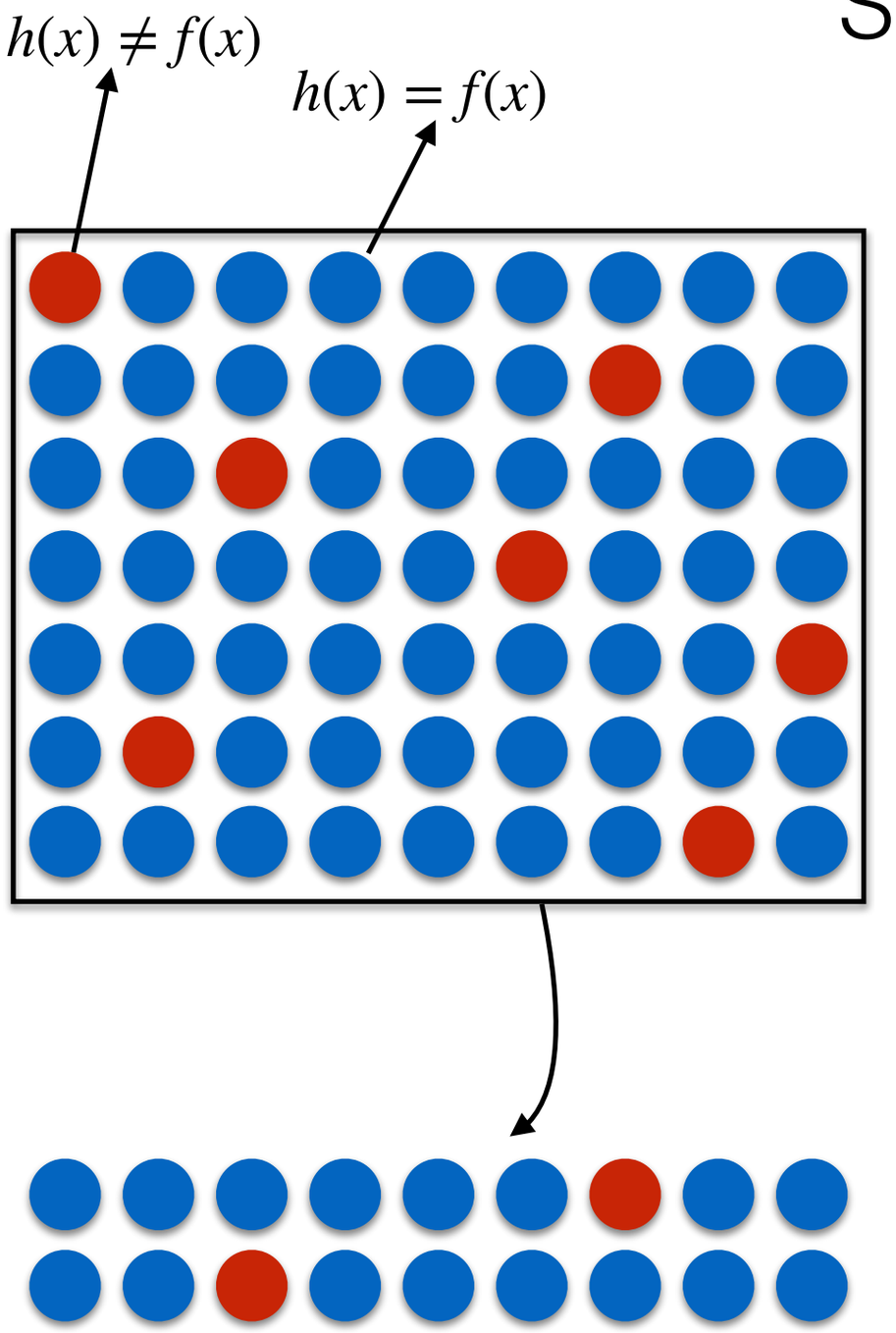
for some fixed h can be written as

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

For a hypothesis h **fixed before looking at the data**, the probability of the in-sample error deviating too much from out-of-sample error is low, so long as the data points in the sample are **drawn independent of each other using some probability distribution** P

Figure adapted from the book Learning from Data: A Short Course

So Are We Done?



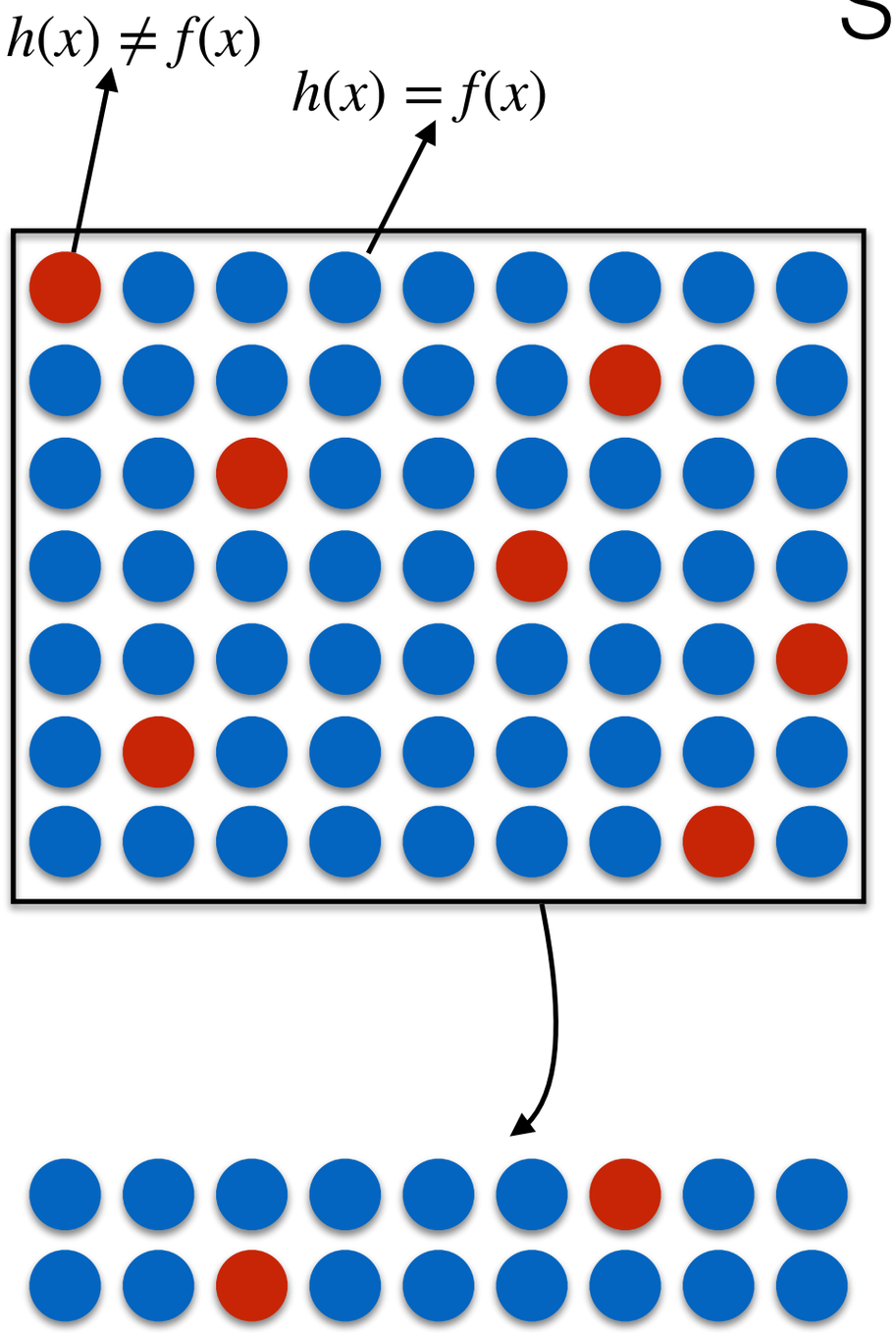
$$\mu \equiv E_{out}(h) = \mathbb{P}[h(x) \neq f(x)]$$ Out-of-sample error

$$\nu \equiv E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N}[h(x_i) \neq f(x_i)]$$ In-sample error

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

Figure adapted from the book Learning from Data: A Short Course

So Are We Done?



$$\mu \equiv E_{out}(h) = \mathbb{P}[h(x) \neq f(x)]$$ Out-of-sample error

$$\nu \equiv E_{in}(h) = \frac{1}{N}\sum_{i=1}^{N}[h(x_i) \neq f(x_i)]$$ In-sample error

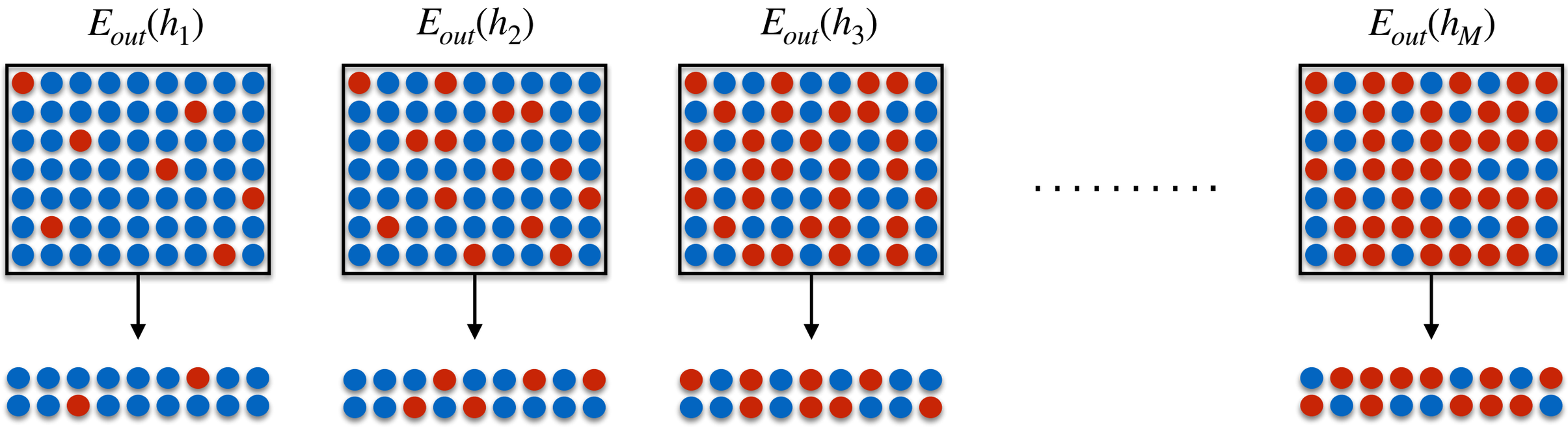
$$\mathbb{P}[\,|E_{in}(h) - E_{out}(h)| > \epsilon] \leq 2e^{-2\epsilon^2 N}$$

Not really!

1. This is only for a single fixed hypothesis h
2. What we have is a hypothesis space $\mathscr{H}$
3. With a single hypothesis its more like "verification" than "learning"
4. Furthermore the h needs to be selected before seeing the data
5. We need to come up with guarantees on $g \in \mathscr{H}$ which by definition is obtained after seeing the data

Figure adapted from the book Learning from Data: A Short Course

Generalizing Bins to Multiple Hypothesis

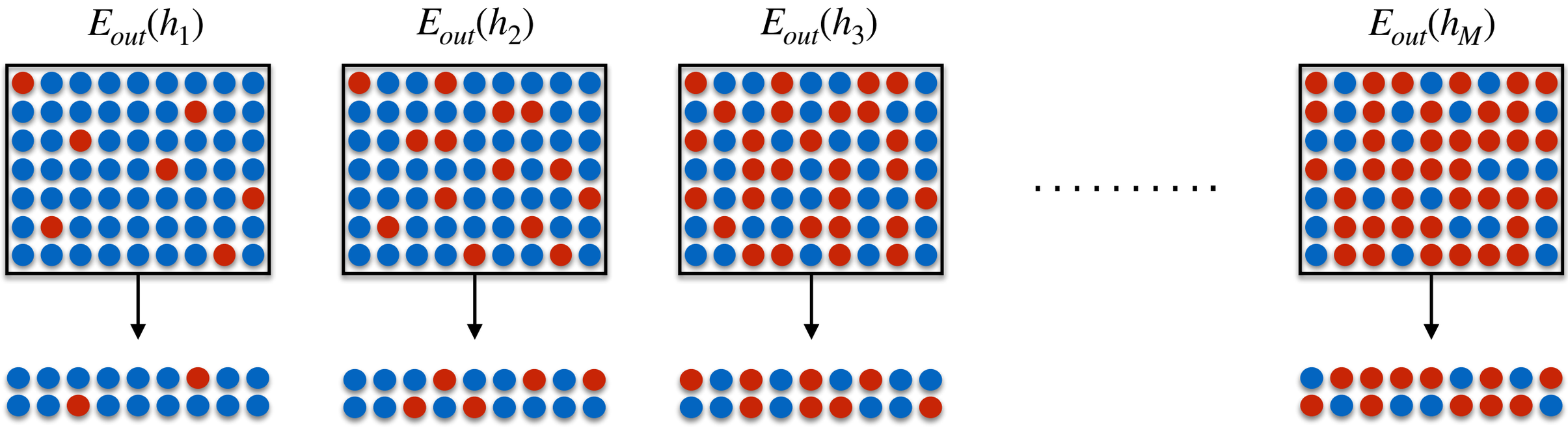


We are interested in estimates for a function $g \in \{h_1, h_2, \ldots, h_M\} = \mathscr{H}$

$$\begin{aligned}\mathbb{P}[\,|E_{in}(g) - E_{out}(g)| > \epsilon] \leq \mathbb{P}[\,&|E_{in}(h_1) - E_{out}(h_1)| > \epsilon \\ &or\ |E_{in}(h_2) - E_{out}(h_2)| > \epsilon \\ &or\ |E_{in}(h_3) - E_{out}(h_3)| > \epsilon \\ &\cdots \\ &or\ |E_{in}(h_M - E_{out}(h_M)| > \epsilon] \\ &\leq \sum_{m=1}^{M} \mathbb{P}[\,|E_{in}(h_m) - E_{out}(h_m)| > \epsilon]\end{aligned}$$

Figure adapted from the book Learning from Data: A Short Course

Generalizing Bins to Multiple Hypothesis



We are interested in estimates for a function $g \in \{h_1, h_2, \ldots, h_M\} = \mathscr{H}$

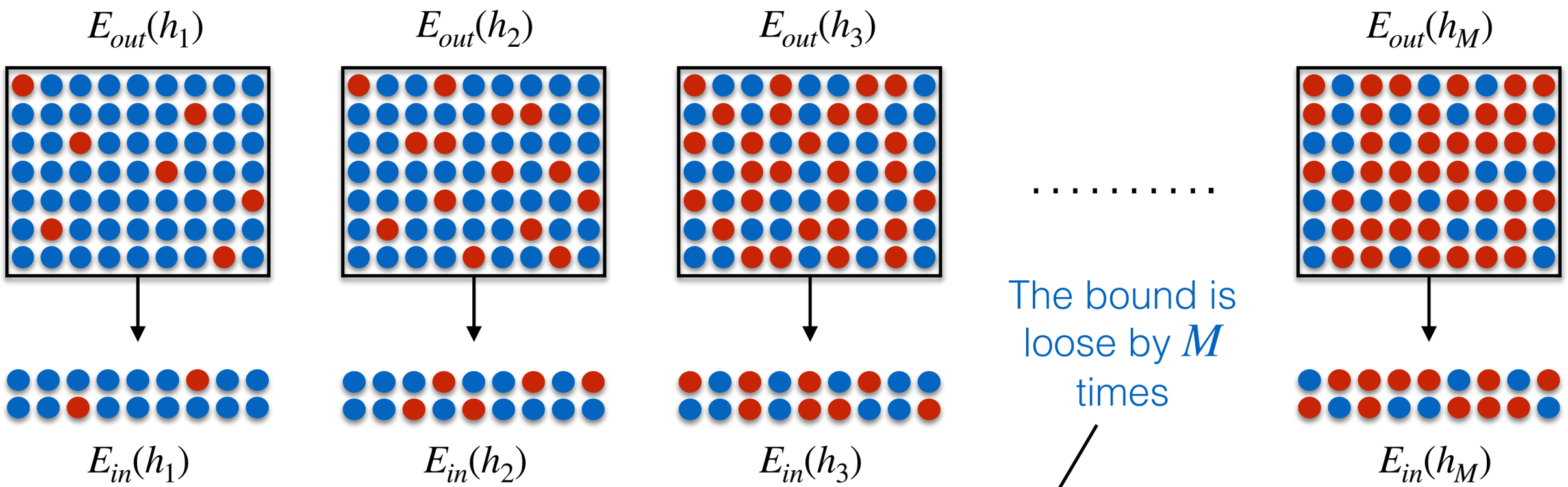
$$\mathbb{P}[\,|E_{in}(g) - E_{out}(g)| > \epsilon] \leq \sum_{m=1}^{M} \mathbb{P}[\,|E_{in}(h_m) - E_{out}(h_m)| > \epsilon]$$

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

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

Figure adapted from the book Learning from Data: A Short Course

Generalizing Bins to Multiple Hypothesis



We are interested in estimates for a function $g \in \{h_1, h_2, \ldots, h_M\} = \mathcal{H}$

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

M can be seen as the complexity of the hypothesis space

Bound is valid only when the hypothesis space is finite

For infinite hypothesis space (often the case) complexity is measured using VC dimension

Its a generalization of this idea

Figure adapted from the book Learning from Data: A Short Course

To Summarize

The goal of learning is to approximate the behavior of the unknown function f

We are usually provided with a finite set of examples which respect the behavior of f (Training Set)

The assumption is that each example is independently drawn from drawn from the entire sample set with some probability distribution P

We identify a hypothesis set $\mathcal{H}$ from which our goal is to pick a function g

g should be such that it approximates the function f on **unseen data**

Under certain assumptions (unseen samples are drawn with the same probability distribution P as the seen examples) we can bound the difference between in-sample and out-of-sample errors

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

This inequality implies that if we can bring the in-sample error towards zero we can probabilistically guarantee that out-of-sample error will not be far either

This essentially means that the hypothesis g will probabilistically approximate function f

Two Primary Question About Learning

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

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

Hoeffding Inequality only addresses the first question

The second question is addressed by actually choosing a hypothesis set, running the learning algorithm and seeing how low can we make $E_{in}(g)$

Complexity of $\mathscr{H}$

There is a tradeoff in deciding how complex should the hypothesis space (M) be

More complex hypothesis enables us to choose a complex g which increases the chances of making $E_{in}(g) \approx 0$

At the same time a complex hypothesis space makes the above bound loose making the gap between E_{in} and E_{out} large

Two Primary Question About Learning

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

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

Hoeffding Inequality only addresses the first question

The second question is addressed by actually choosing a hypothesis set, running the learning algorithm and seeing how low can we make $E_{in}(g)$

Complexity of $\mathcal{H}$

There is a tradeoff in deciding how complex should the hypothesis space (M) be

More complex hypothesis enables us to choose a complex g which increases the chances of making $E_{in}(g) \approx 0$

At the same time a complex hypothesis space makes the above bound loose making the gap between E_{in} and E_{out} large

---

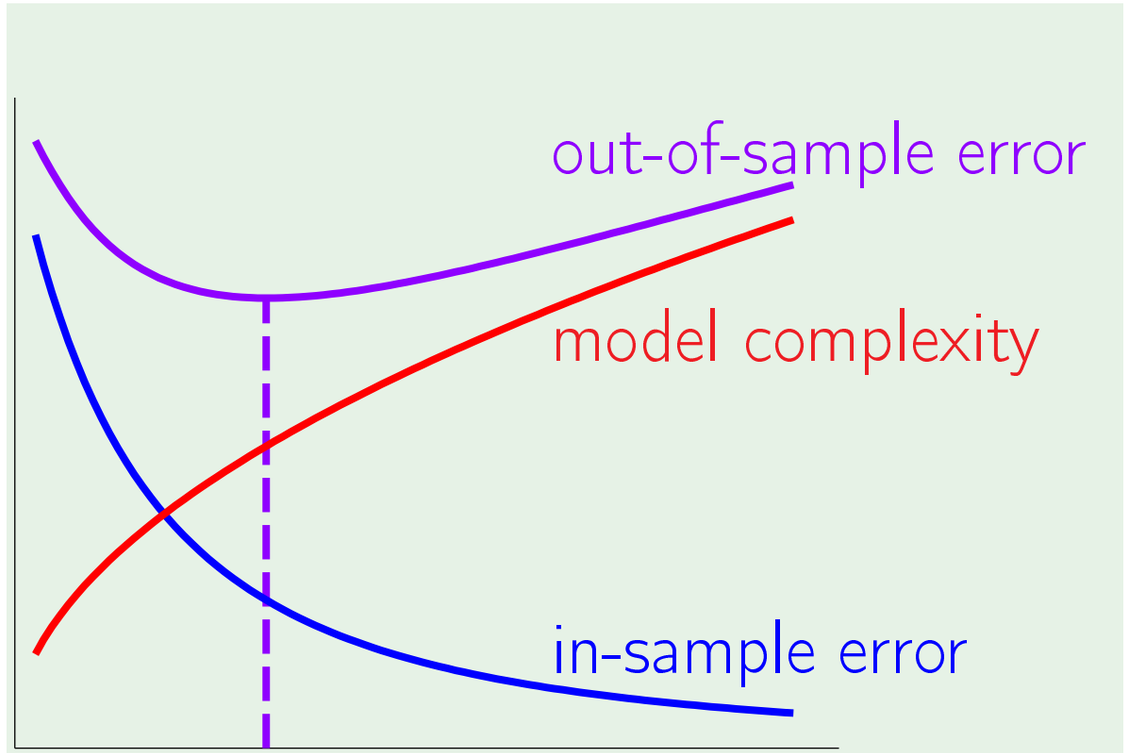
Complexity of f

Complexity of f impacts the second question above

A more complex f is harder to learn and hence will require a more complex g which in-turn implies a more complex hypothesis set $\mathcal{H}$

Two Primary Question About Learning

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



Complexity of $\mathscr{H}$

There is a tradeoff in deciding how complex should the hypothesis space (M) be

More complex hypothesis enables us to choose a complex g which increases the chances of making $E_{in}(g) \approx 0$

At the same time a complex hypothesis space makes the above bound loose making the gap between E_{in} and E_{out} large

Complexity of f

Complexity of f impacts the second question above

A more complex f is harder to learn and hence will require a more complex g which in-turn implies a more complex hypothesis set $\mathscr{H}$

Figure adapted from the book Learning from Data: A Short Course

Measuring Complexity of $\mathscr{H}$

VC Dimension

We will not cover in this course

Strongly encourage you to read about it on your own

Suggested tutorial

Ask questions if you do not understand anything

Bias-Variance

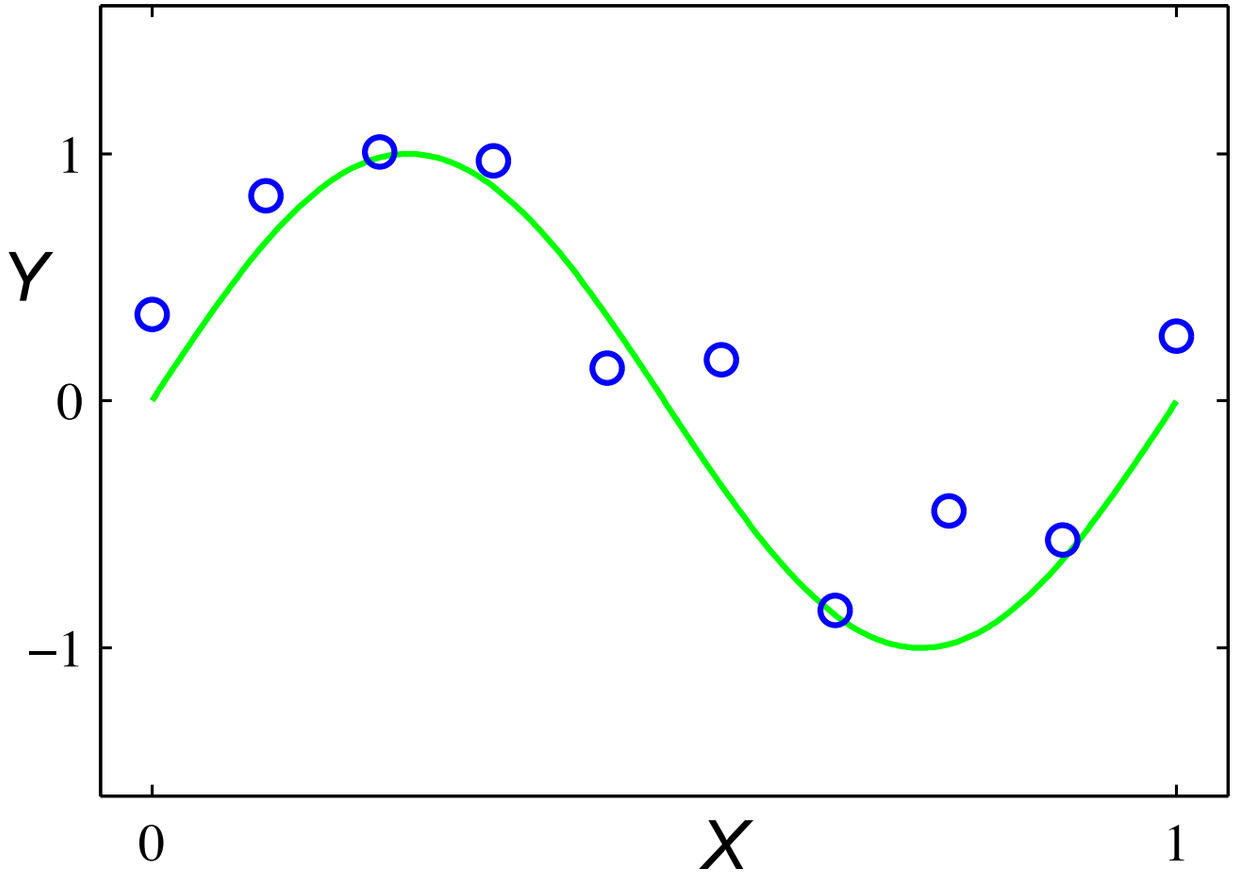
We will cover in the next lecture

Another Example Problem

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

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

Complexity of $\mathscr{H}$ and Complexity of f



Training data: $D = \{(x_1, y_1), \ldots, (x_{10}, y_{10})\}$ obtained from a Sin function with **some noise**

Of course we do not know that the data is drawn from a Sin function

Both X and Y are real

Task is to find g that best approximate this underlying unknown function

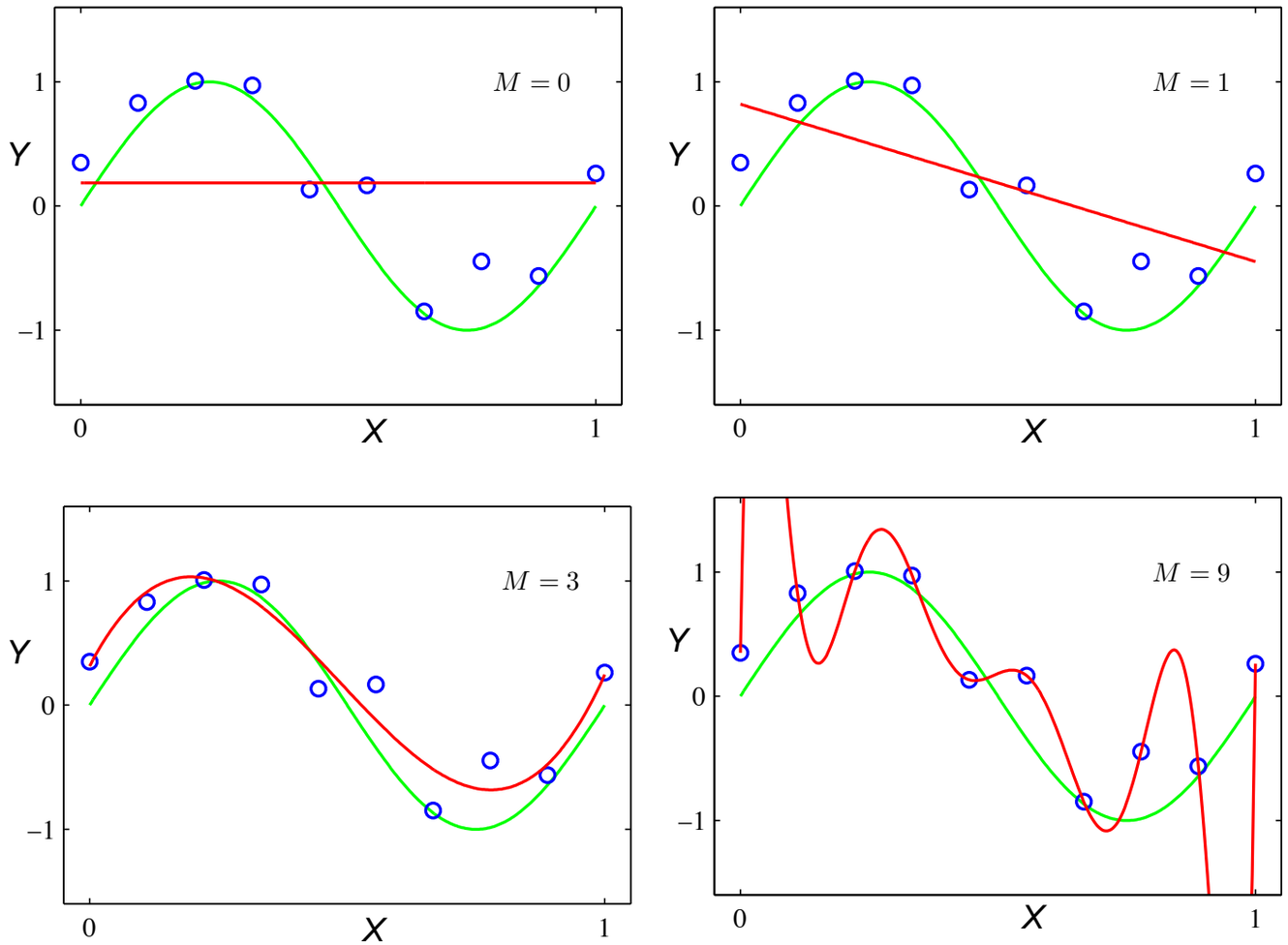
Figure adapted from the book Pattern Recognition and Machine Learning

Another Example Problem

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

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

Complexity of $\mathscr{H}$ and Complexity of f



Let the hypothesis space $\mathscr{H}$ be the space of all the polynomials of degree M, for some fixed M

So complexity of $\mathscr{H}$ is defined by M

What's the best hypothesis?

Figure adapted from the book Pattern Recognition and Machine Learning

How to Accomplish This In Practice?

Split your available dataset into three components

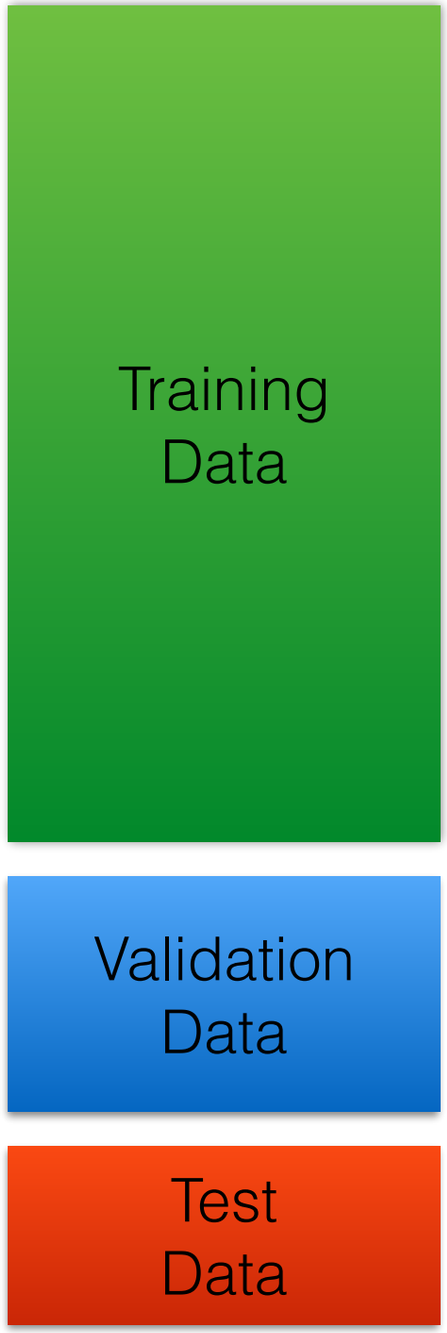
Training data: the data you use to tune the parameters of the model (search for the hypothesis)

Validation data: part of the data you use to test the goodness of the running hypothesis. Performance on the validation set is a surrogate of the performance on unseen examples — so long as the validation set is drawn from the same distribution as the test set

Test data: once you are happy with your chosen hypothesis, test data is used to measure the final performance of the model. **There is no going back after this!** Otherwise you will be cheating and all bets are off whether your model will generalize to unseen data

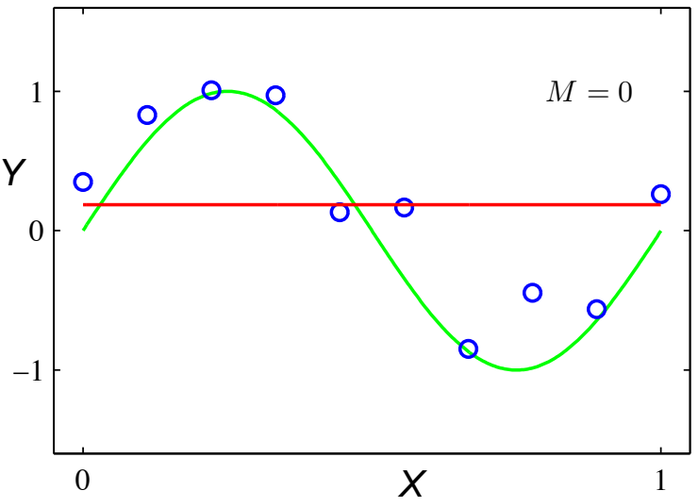
The big assumption here is that the dataset you start with is representative of the real world

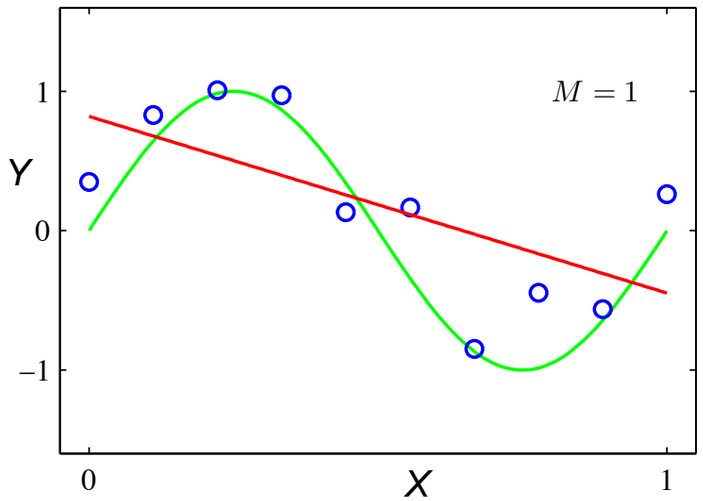
Throughout learning you track the performance on the validation set!

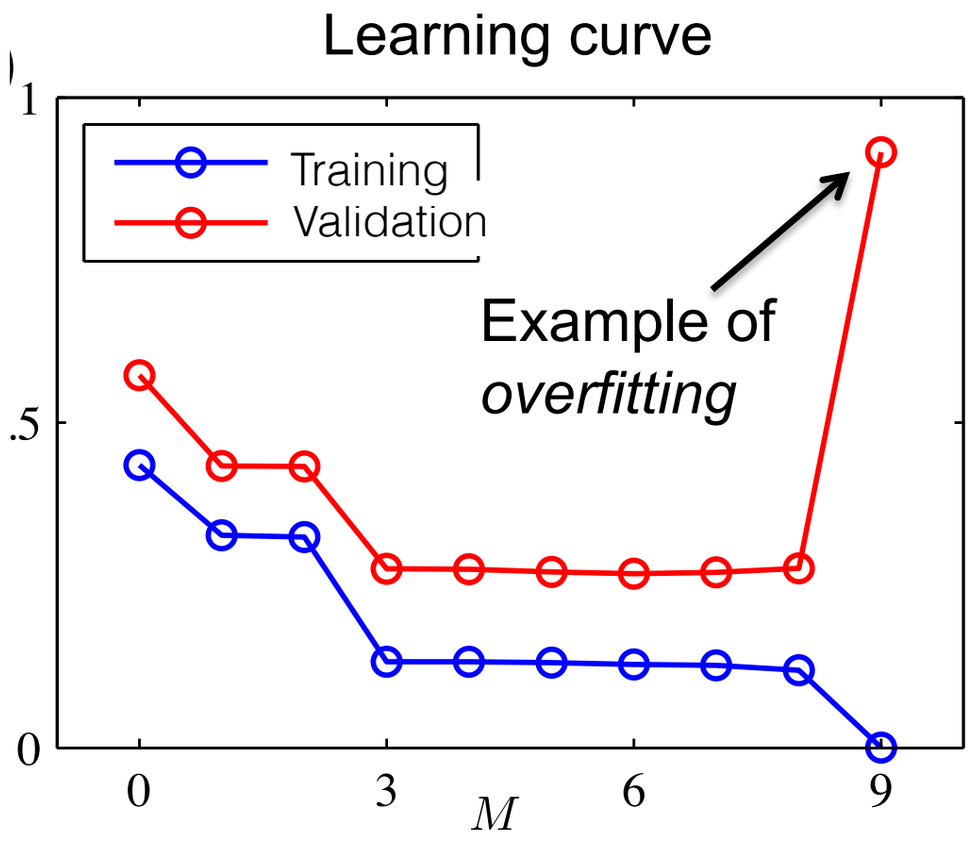


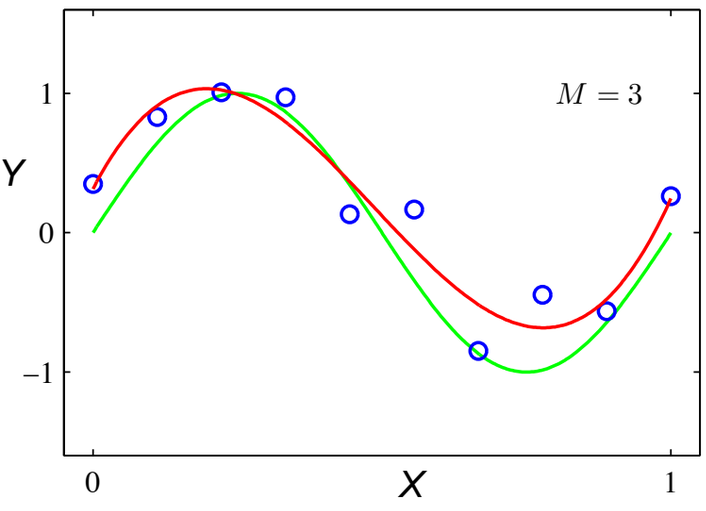
How to Accomplish This In Practice?

This is also a practical way to avoid **Overfitting**









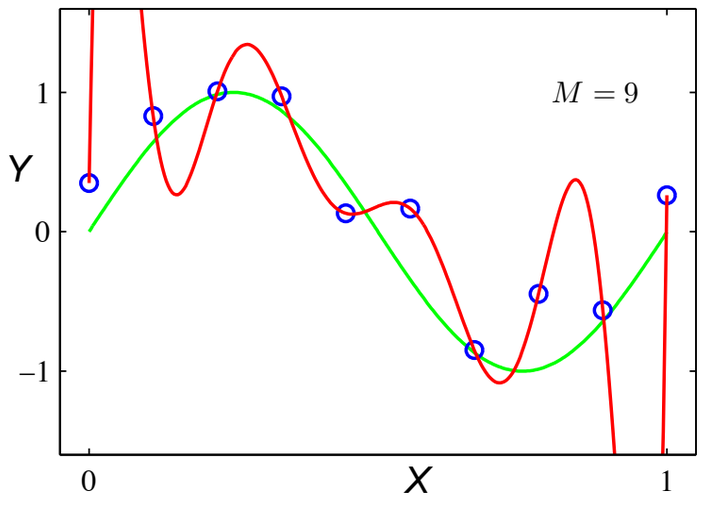


Figure adapted from the book Pattern Recognition and Machine Learning

Error Measure (a.k.a Loss Function)

The hypothesis h only approximates the unknown function f

Its goodness is measured by defining an error measure $E(h, f)$

Even though the sample space $\mathcal{X}$ could be continuous in almost every practical scenario we define a point-wise error measure

$$E(h, f) = \sum_{i=1}^{N} e(h(x_i), f(x_i))^2$$

There are many error measures other than the classification error

Binary classification error: $e(h, f) = [h(x) \neq f(x)]$

Squared error: $e(h, f) = (h(x) - f(x))^2$

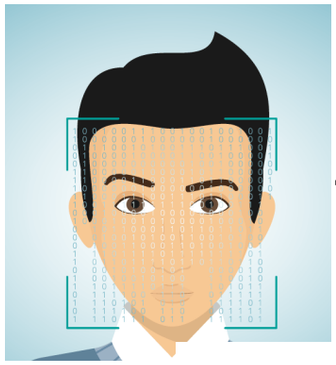
Cross-entropy loss: $-\left[y \log p + (1 - y) \log(1 - p)\right]$

Error Measure (a.k.a Loss Function)

The error measure should be defined by the user and is problem dependent

Its the error measure that decides which hypothesis gets selected at the end

Different error measures on the same task can lead to different hypothesis



ML Model

+1 if it is you

-1 if it is not you

		f	
		1	-1
h	1	No error	False accept
	-1	False reject	No error

Error Function for Super Market

		f	
		Accept	Reject
h	Accept	0	1
	Reject	10	0

Error Function for Bank

		f	
		Accept	Reject
h	Accept	0	1000
	Reject	10	0

Figure adapted from the book Learning from Data: A Short Course

Noise

So far we've assumed the existence of a deterministic function $y = f(x)$

In reality, the mapping $x \rightarrow y$ is rarely deterministic

Consider the bank's problem of issuing credit cards to customers

Two customers having the same exact input features might exhibit different behavior of paying back the loans

This can be modeled formally as follows:

Instead of using $y = f(x)$, we use $P(y \mid x)$

A data point (x, y) is now generated by a joint distribution $P(x, y) = P(x)P(y \mid x)$

One can think of noisy targets as a deterministic target $y = \mathbb{E}(y \mid x)$ and some added noise given by $y - f(x)$

$P(x)$ provides us with the relative importance of the input samples — Input Distribution

$P(y \mid x)$ is what we are really interested in finding out — Target Distribution

Components of a Learning System

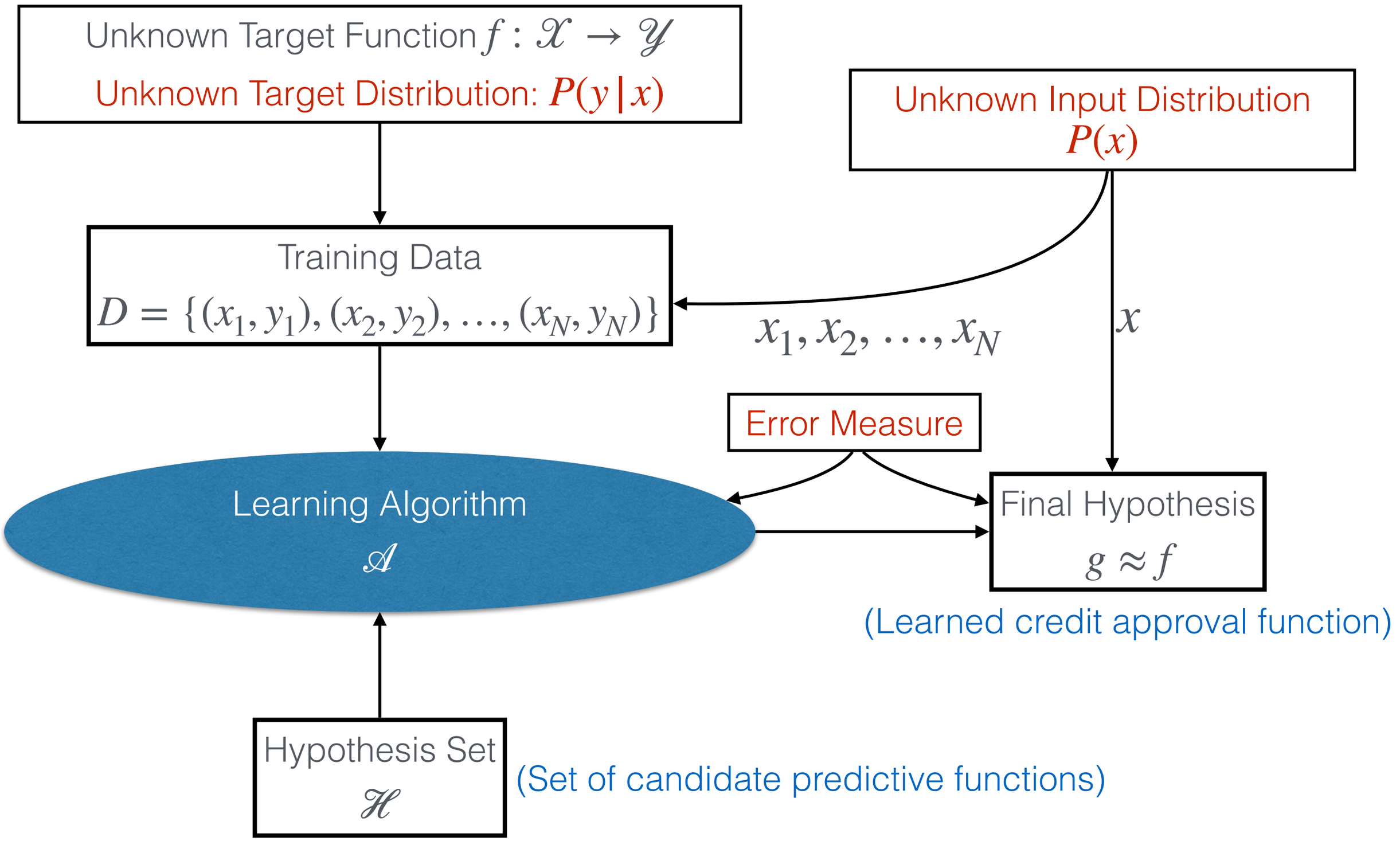


Figure adapted from the book Learning from Data: A Short Course