PACMANN AI

Machine Learning for Beginners

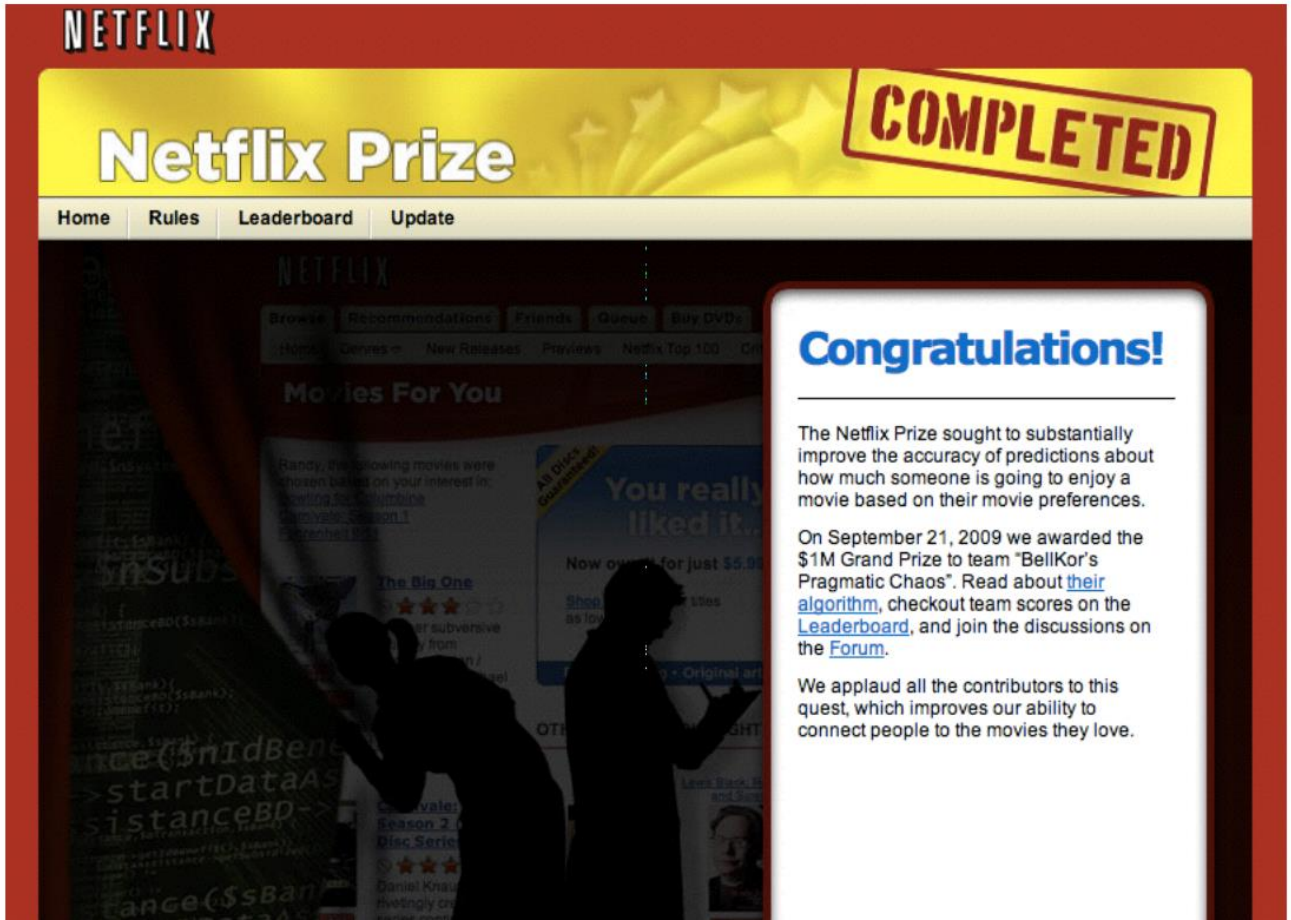
PACMANN AI

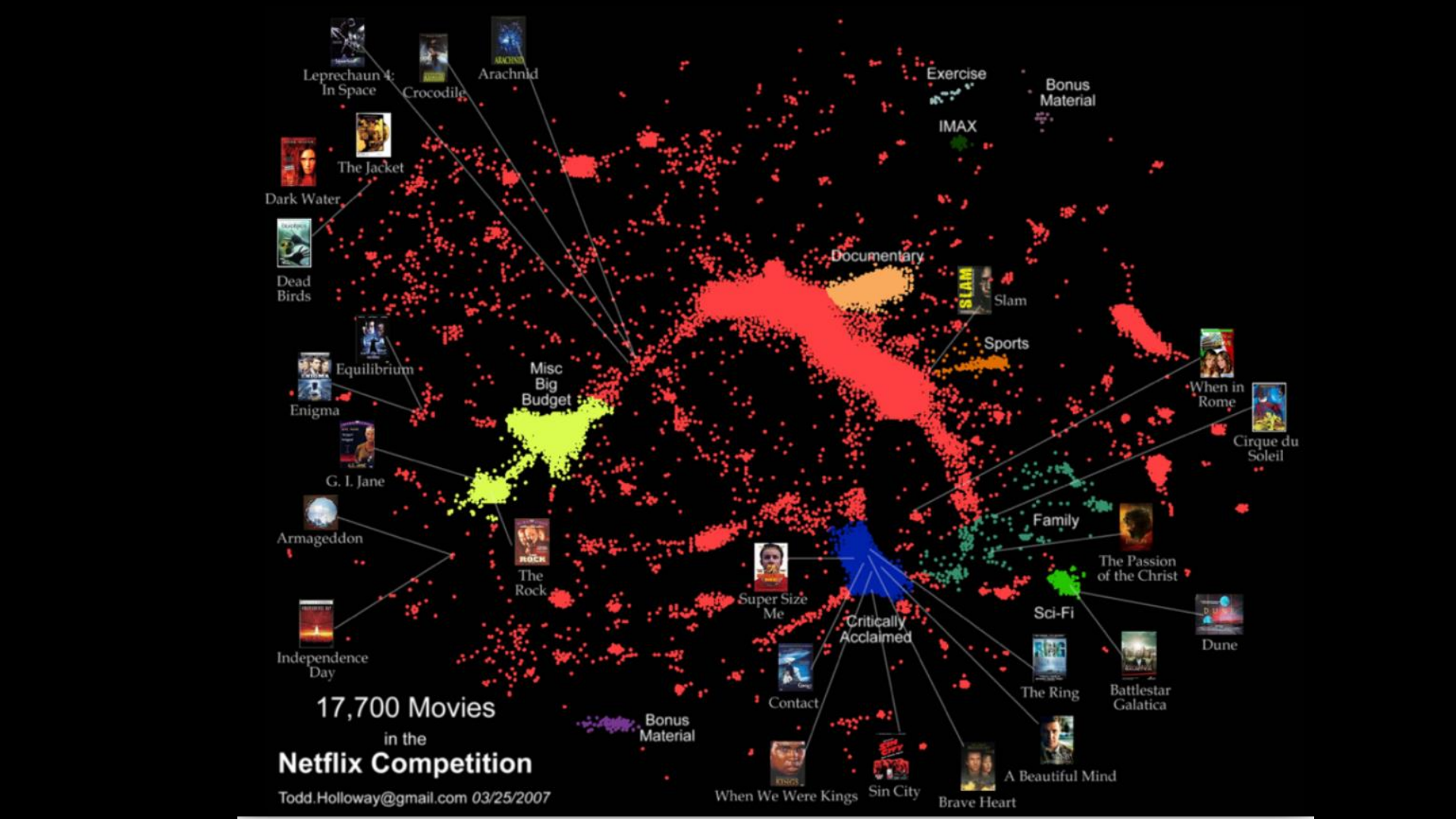
Theoretical Machine Learning

Adityo Sanjaya
Head of Research PACMANN AI
Email: adityosanjaya3.14@gmail.com
+62 85777490099

Slides from: Yasser Mostafa, Learning from Data, California Technology

Netflix Prize




Leprechaun 4: In Space
Crocodile
Arachnid
Exercise
Bonus Material
IMAX
The Jacket
Dark Water
Dead Birds
Documentary
Slam
Sports
Equilibrium
Misc Big Budget
Enigma
When in Rome
Cirque du Soleil
G. I. Jane
Armageddon
Family
The Rock
The Passion of the Christ
Super Size Me
Critically Acclaimed
Sci-Fi
Dune
Independence Day
Contact
The Ring
Battlestar Galatica
17,700 Movies
in the
Netflix Competition
Bonus Material
When We Were Kings
Sin City
Brave Heart
A Beautiful Mind
Todd.Holloway@gmail.com 03/25/2007

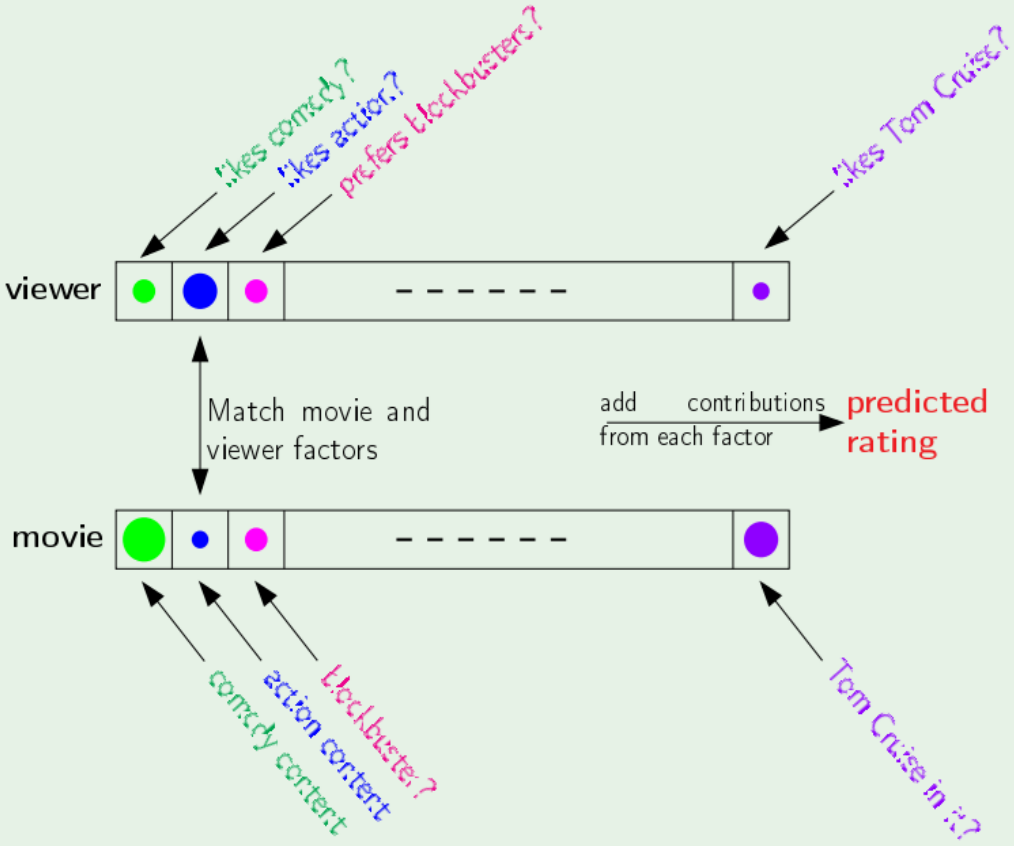
Example: Predicting how a viewer will rate a movie

10% improvement = 1 million dollar prize

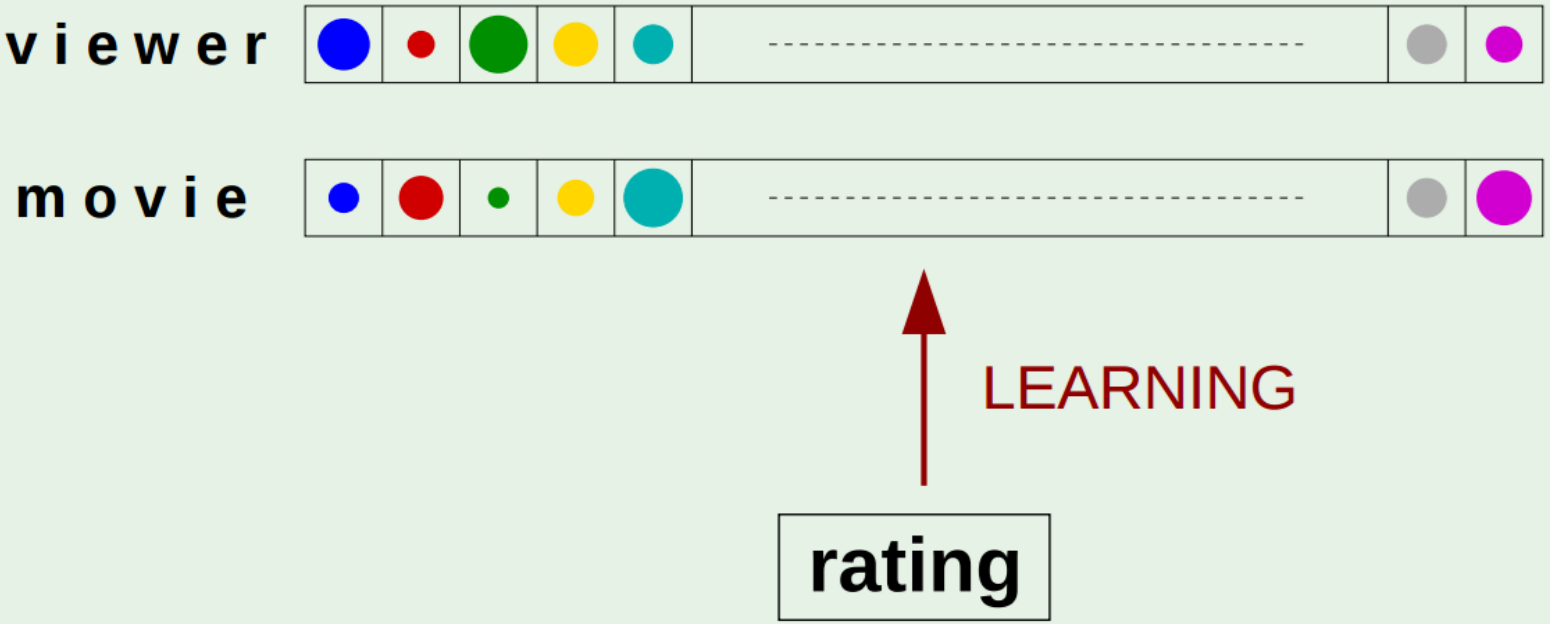
The essence of machine learning:

- A pattern exists.
- We cannot pin it down mathematically.
- We have data on it.

Movie rating - a solution



The learning approach



Components of learning

Metaphor: Credit approval

Applicant information:

age	23 years
gender	male
annual salary	$30,000
years in residence	1 year
years in job	1 year
current debt	$15,000
...	...

Approve credit?

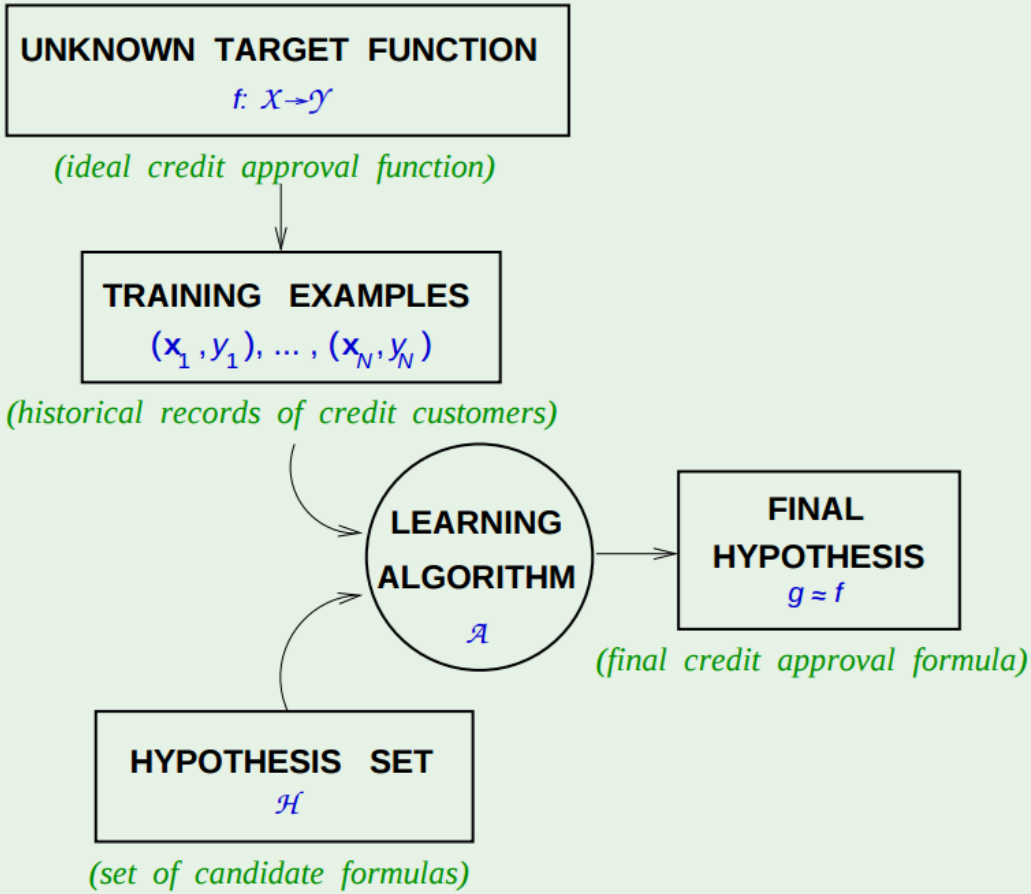
Components of learning

Formalization:

- Input: $\mathbf{x}$ *(customer application)*
- Output: y *(good/bad customer?)*
- Target function: $f : \mathcal{X} \to \mathcal{Y}$ *(ideal credit approval formula)*
- Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$ *(historical records)*

$\downarrow \quad \downarrow \quad \downarrow$

- Hypothesis: $g : \mathcal{X} \to \mathcal{Y}$ *(formula to be used)*


UNKNOWN TARGET FUNCTION
$f: \mathcal{X} \to \mathcal{Y}$
(ideal credit approval function)
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$
(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)

Solution components

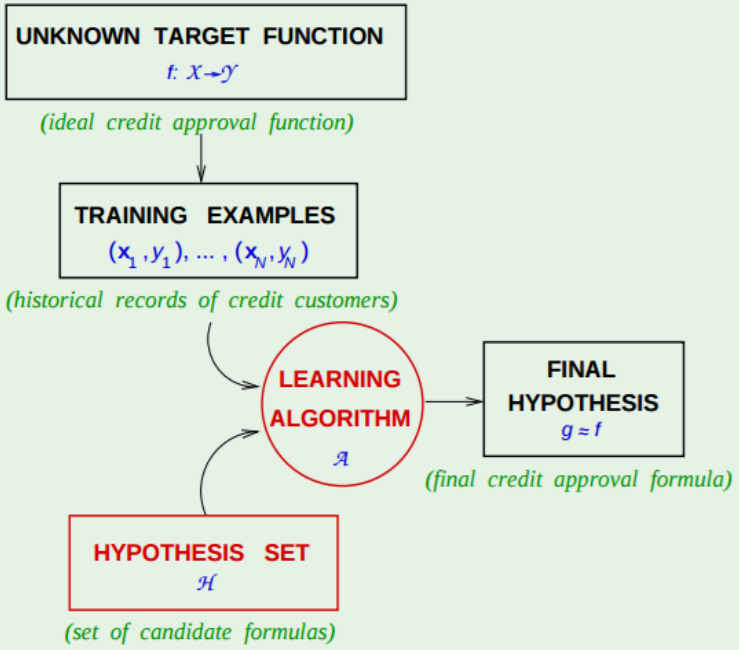
The 2 solution components of the learning problem:

- The Hypothesis Set

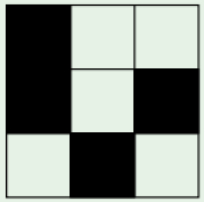
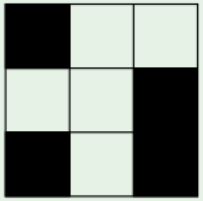
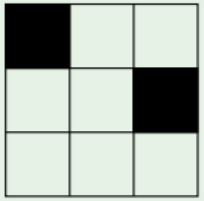
  $\mathcal{H} = \{h\}$ $\quad$ $g \in \mathcal{H}$

- The Learning Algorithm

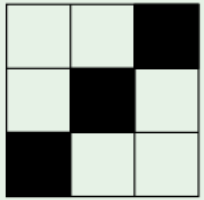
Together, they are referred to as the *learning model*.

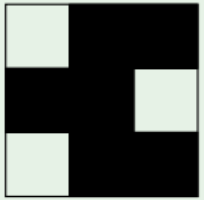


A Learning puzzle

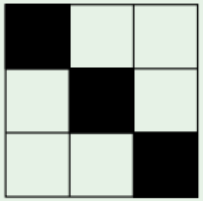




$f = -1$





$f = +1$



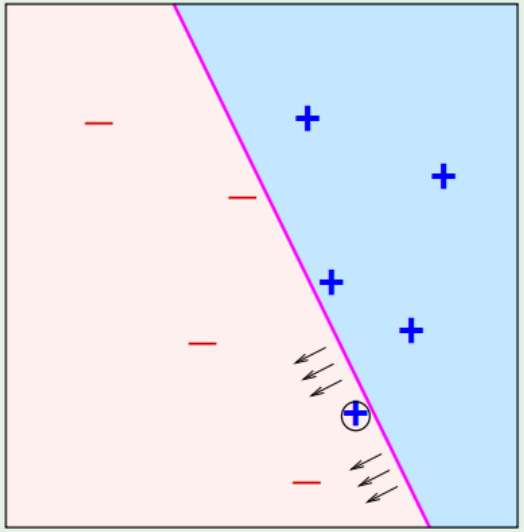
$f = ?$

Is Learning Feasible?

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?

A related experiment

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

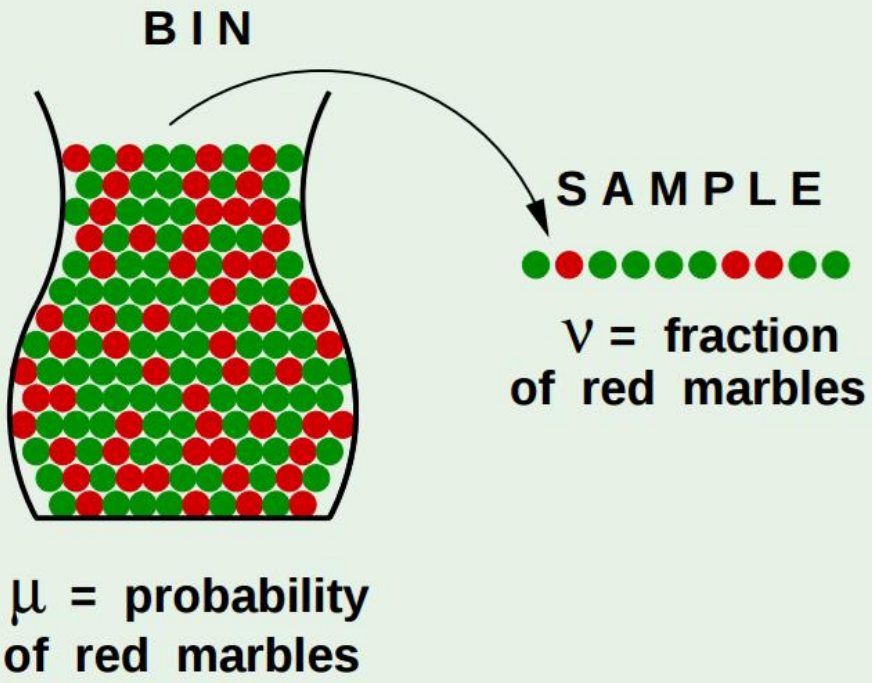
$$\mathbb{P}[\text{ picking a red marble }] = \mu$$

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

- 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?

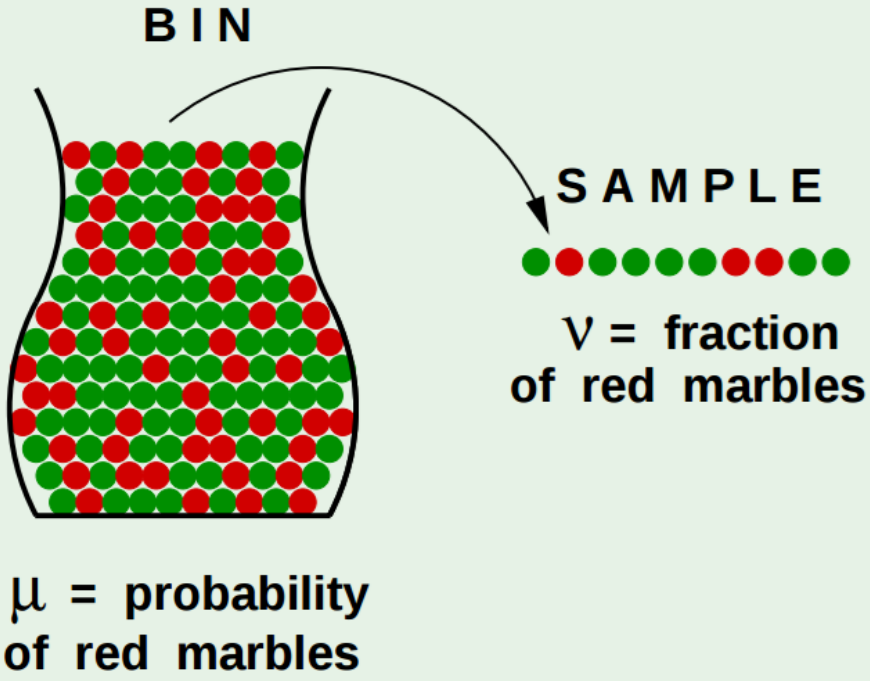
No!

Sample can be mostly green while bin is mostly red.

Yes!

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,

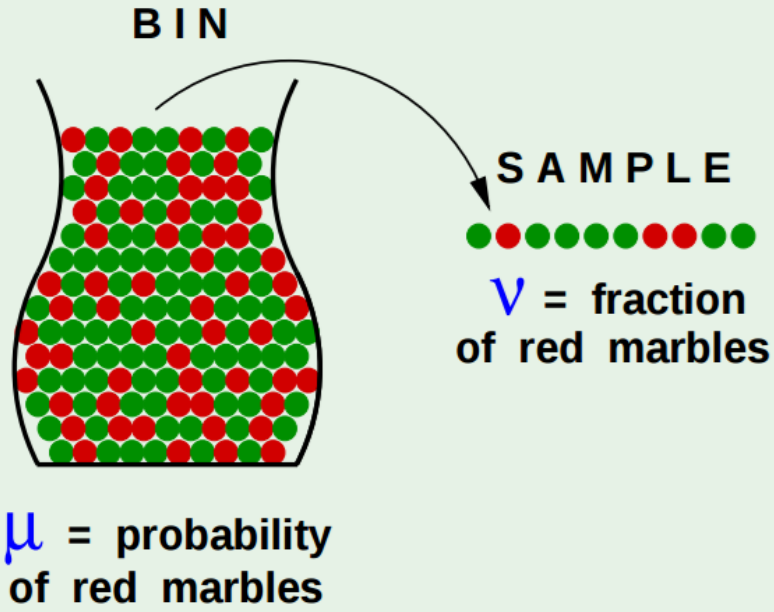
$$\mathbb{P}\left[\left|\nu - \mu\right| > \epsilon\right] \leq 2e^{-2\epsilon^2 N}$$

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

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

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

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



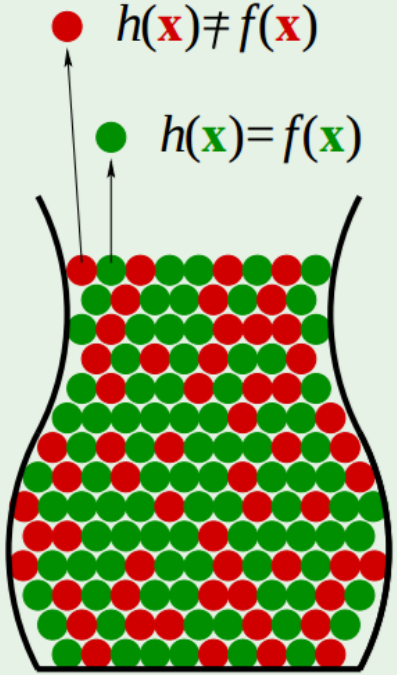
Connection to learning

Bin: The unknown is a number μ

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

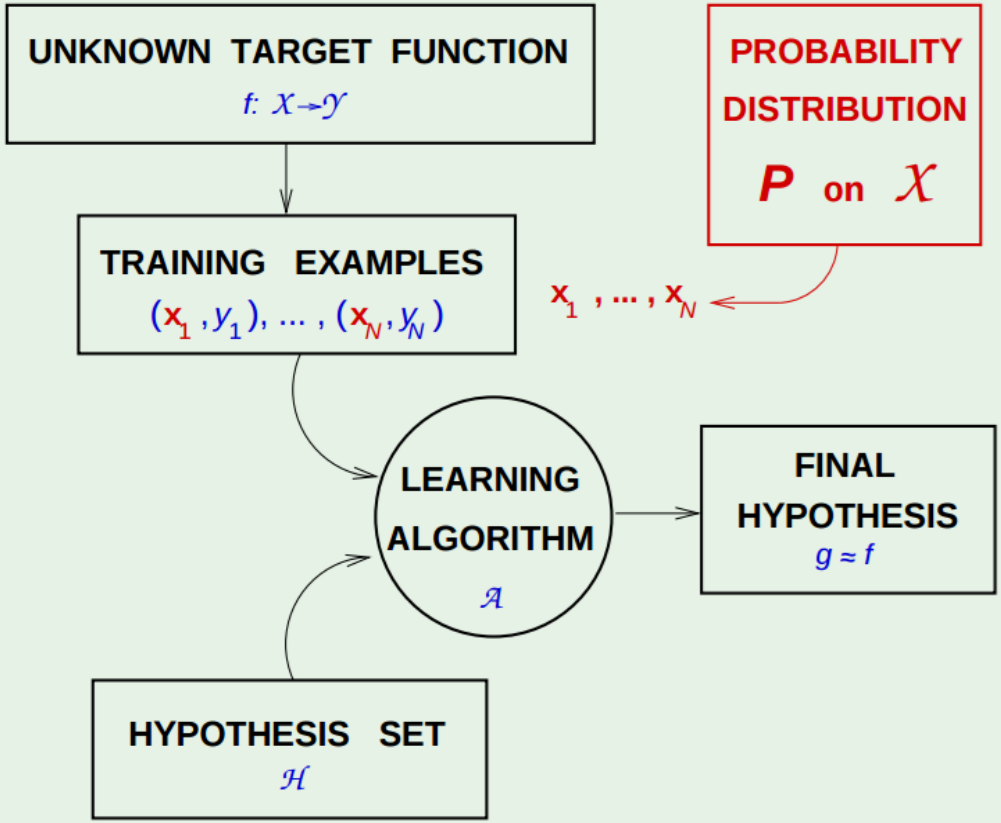
Each marble ● is a point $\mathbf{x} \in \mathcal{X}$

- ● : 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?

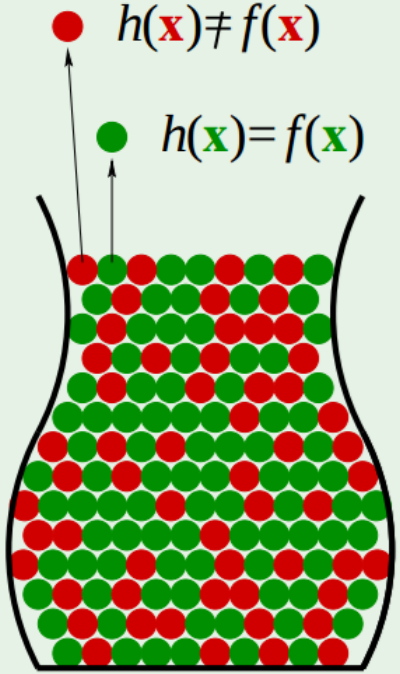
Not so fast! h is fixed.

For this h, ν generalizes to μ.

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

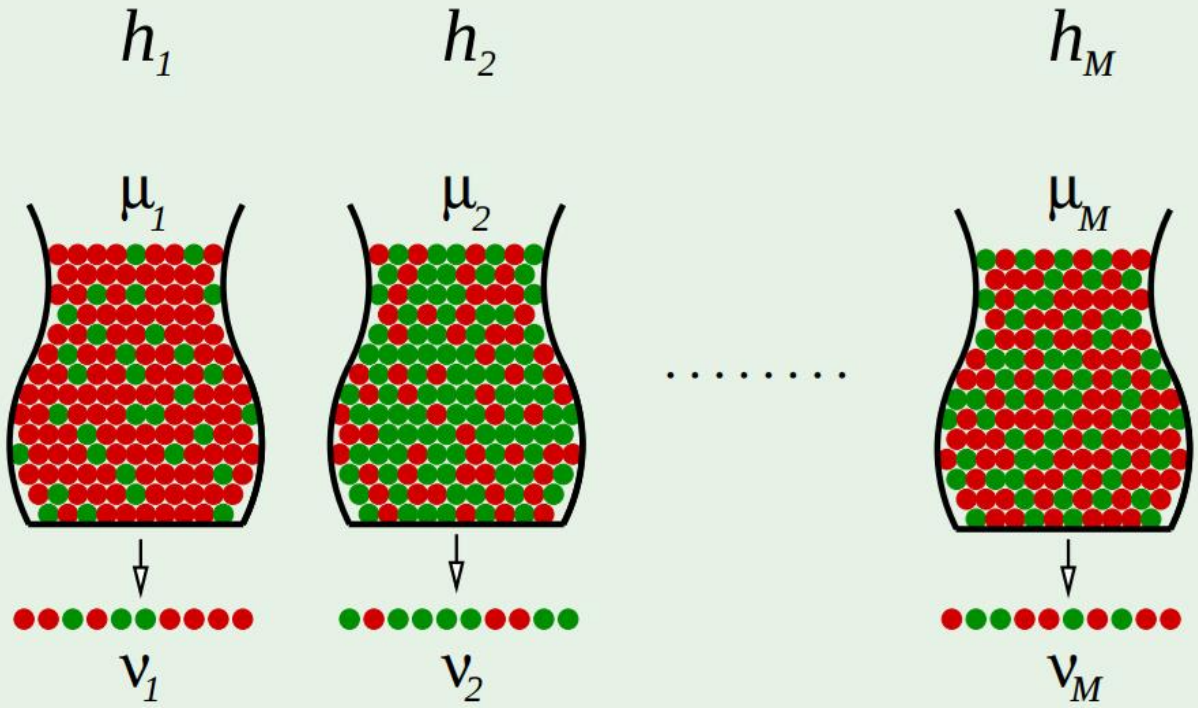
No guarantee ν will be small.

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



Multiple bins

Generalizing the bin model to more than one hypothesis:



Notation for learning

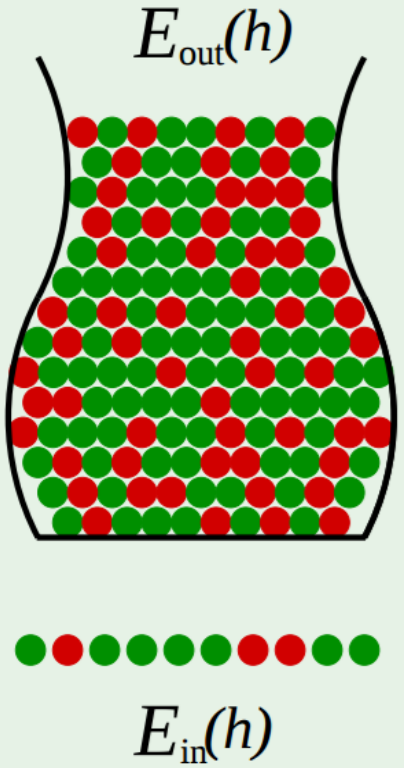
Both μ and ν depend on which hypothesis h

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

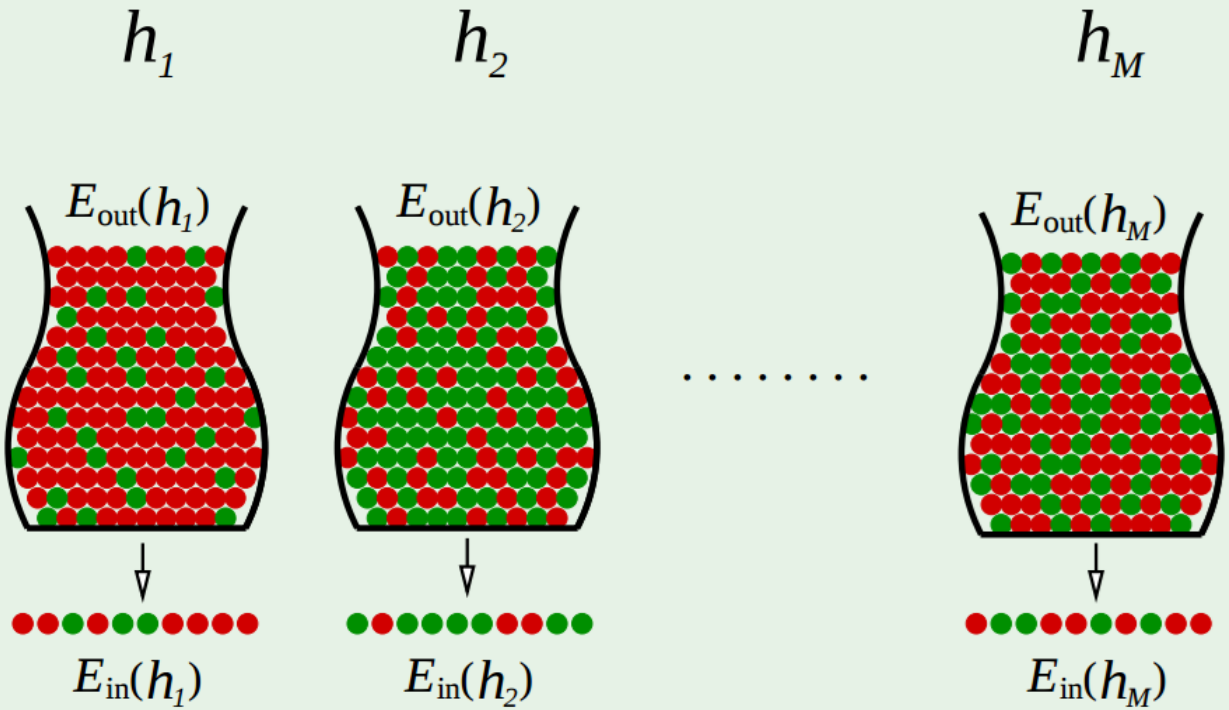
μ is '**out of sample**' denoted by $E_{\text{out}}(h)$

The Hoeffding inequality becomes:

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



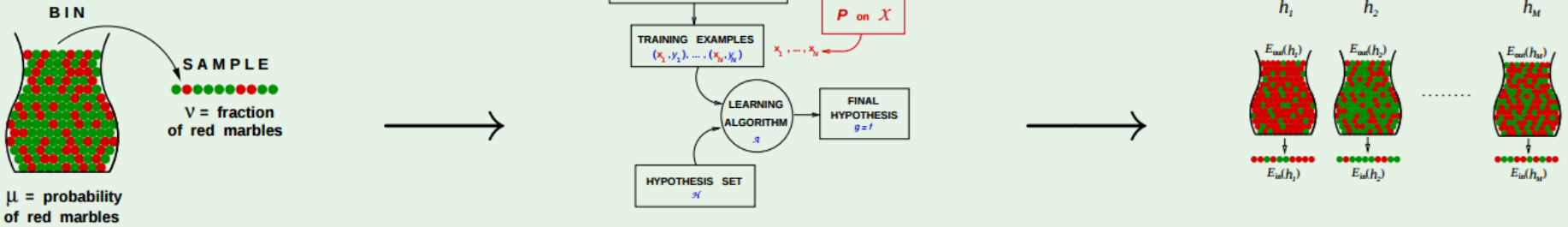
Notation with multiple bins



Are we done already? ☺

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

What?



Coin analogy

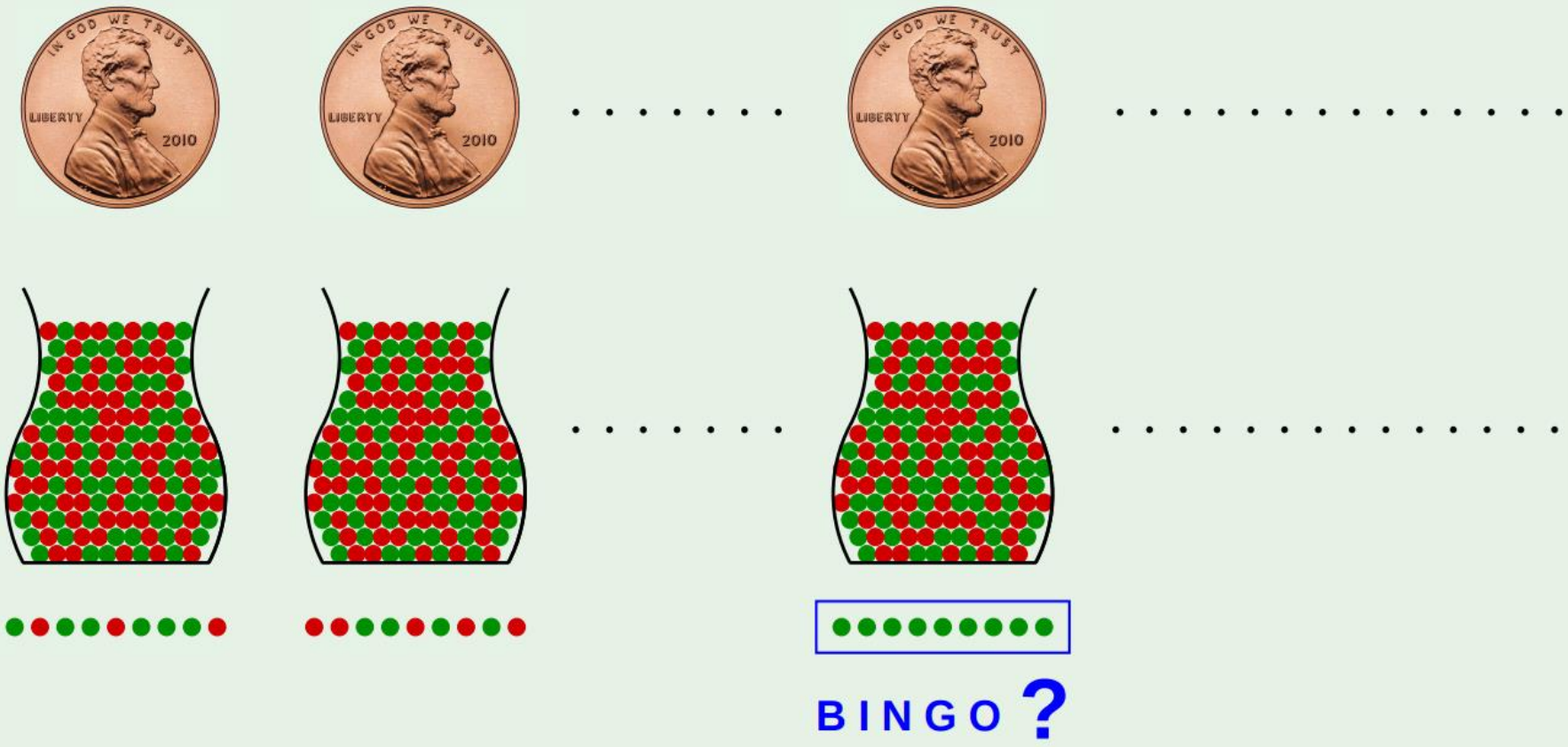
Question: If you toss a fair coin 10 times, what is the probability that you will get 10 heads?

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: $\approx 63\%$

From coins to learning



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

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

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

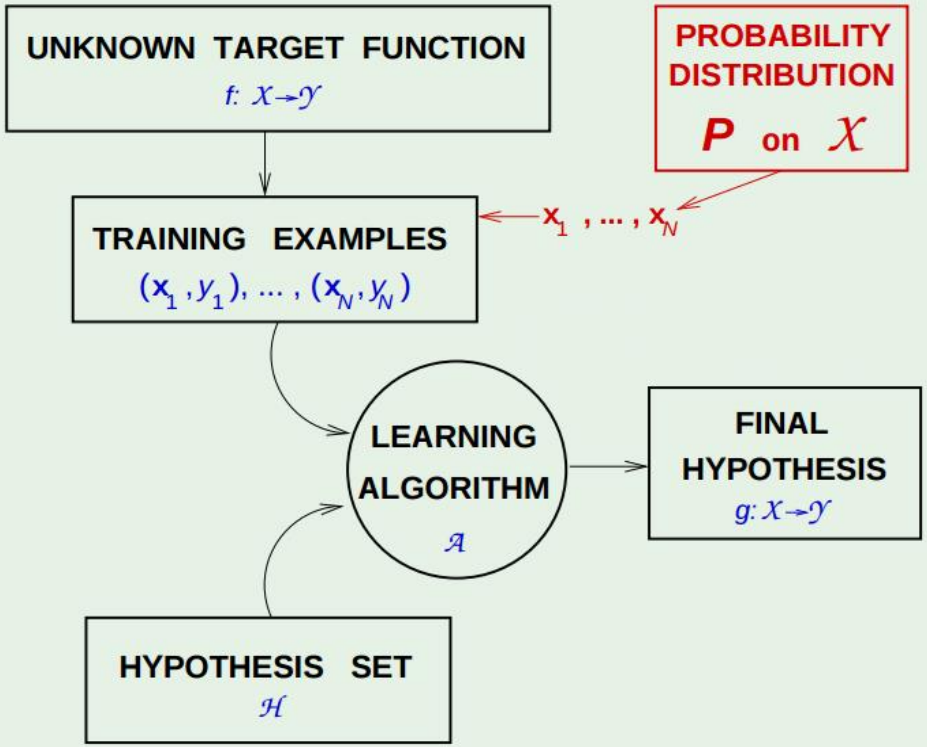
Training Error vs Test Error

Regression and Classification

- Given some input $\boldsymbol{x} = \{x^{(1)}, x^{(2)}, \dots, x^{(p)}\}$

Regression	Classification
Output: • $f(\mathbf{x})$: $\mathbb{R}^p \rightarrow \mathbb{R}$	Output: • Binary Classification: $\Omega = \{-1,1\}$ • Multi-class Classification: $\Omega = \{1, 2, 3, \dots, C\}$ • $f(\mathbf{x})$: $\mathbb{R}^p \rightarrow \Omega$

The learning diagram - where we left it



Error measures

What does "$h \approx f$" mean?

Error measure: $E(h, f)$

Almost always pointwise definition: $\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right)$

Examples:

Squared error: $\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right) = \left(h(\mathbf{x}) - f(\mathbf{x})\right)^2$

Binary error: $\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right) = [\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$

From pointwise to overall

Overall error $E(h, f)$ = average of pointwise errors $\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right)$.

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:

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

The 2 questions of learning

$E_{out}(g) \approx 0$ is achieved through:

$$\underbrace{E_{out}(g) \approx E_{in}(g)}_{\text{Lecture 2}} \qquad \text{and} \qquad \underbrace{E_{in}(g) \approx 0}_{\text{Lecture 3}}$$

Learning is thus split into 2 questions:

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

What we want to have is a low Test Error.

Overfitting

Error measures

What does "$h \approx f$" mean?

Error measure: $E(h, f)$

Almost always pointwise definition: $\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right)$

Examples:

Squared error: $\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right) = \left(h(\mathbf{x}) - f(\mathbf{x})\right)^2$

Binary error: $\mathrm{e}\left(h(\mathbf{x}), f(\mathbf{x})\right) = [\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$

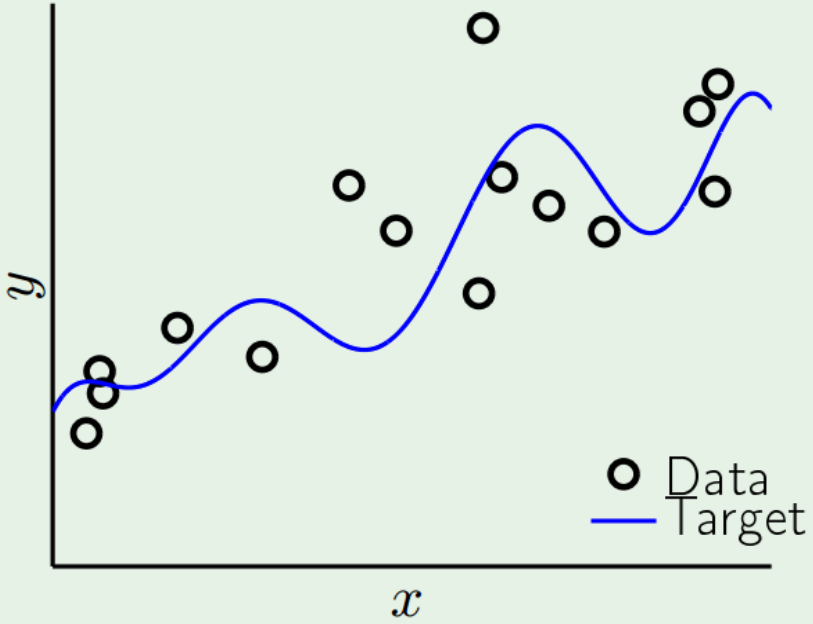
The culprit

Overfitting: "fitting the data more than is warranted"

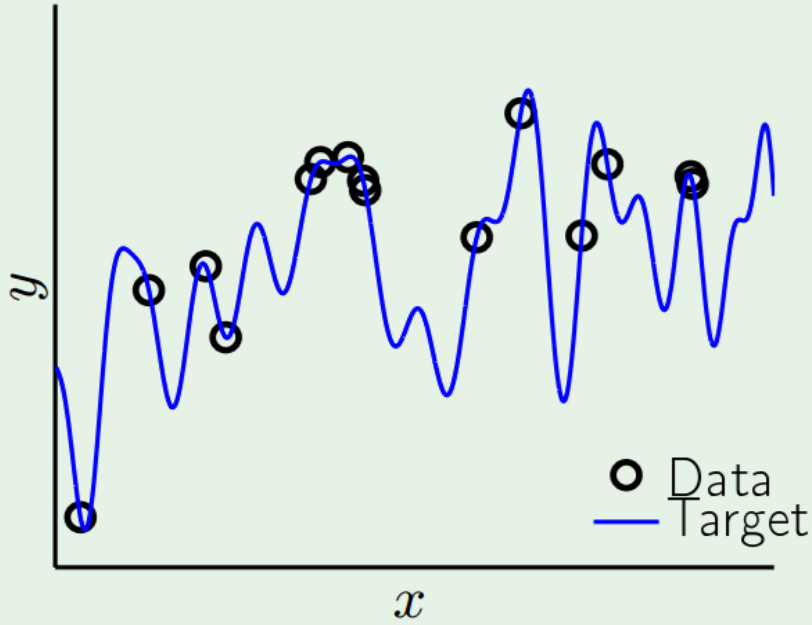
Culprit: fitting the noise - harmful

Case study

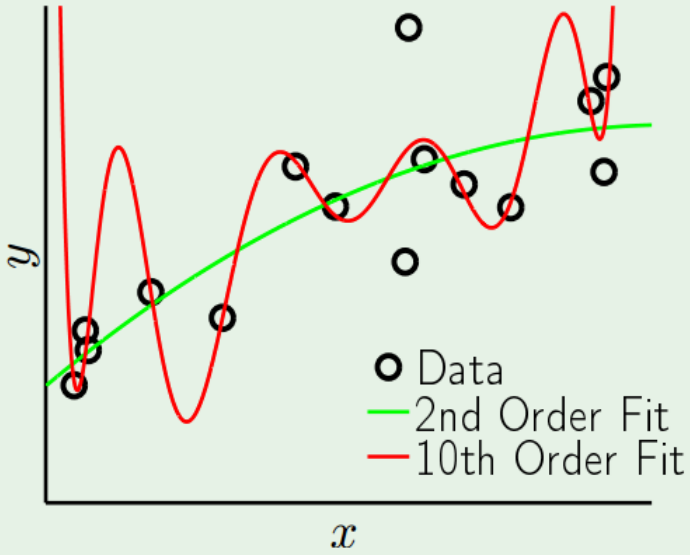
10th-order target + noise



50th-order target

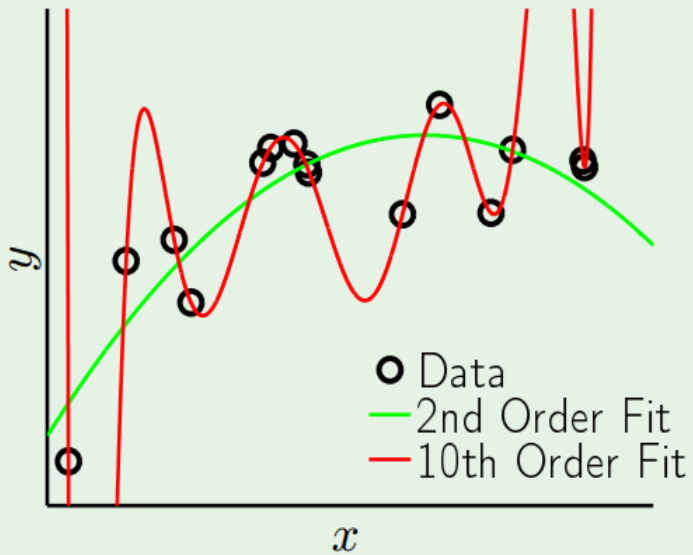


Two fits for each target



Noisy low-order target

	2nd Order	10th Order
E_{in}	0.050	0.034
E_{out}	0.127	9.00



Noiseless high-order target

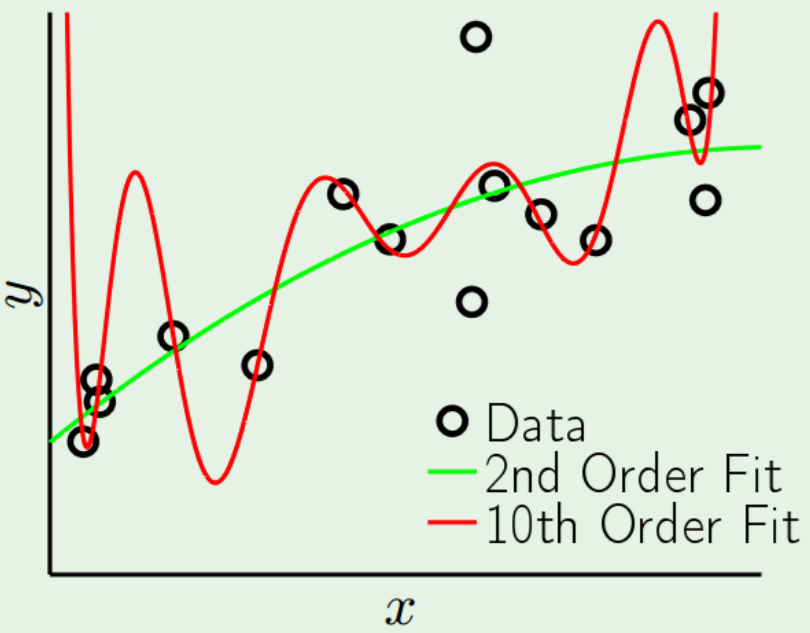
	2nd Order	10th Order
E_{in}	0.029	10^{-5}
E_{out}	0.120	7680

An irony of two learners

Two learners O and R

They know the target is 10th order

O chooses $\mathcal{H}_{10}$ R chooses $\mathcal{H}_2$



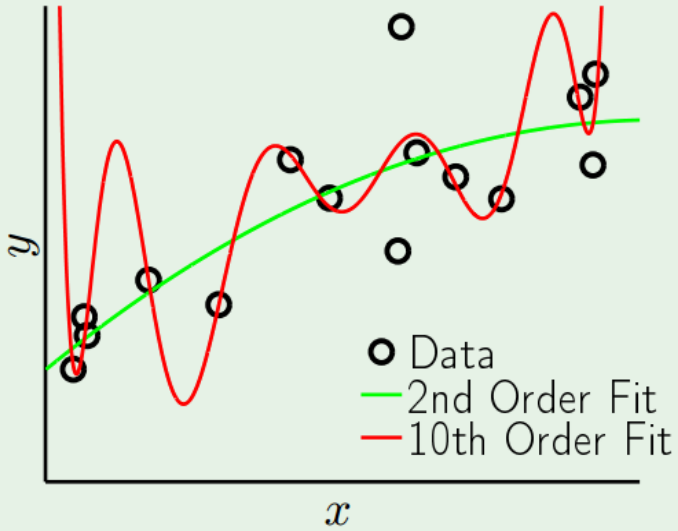
Learning a 10th-order target

The overfit measure

We fit the data set $(x_1, y_1), \cdots, (x_N, y_N)$ using our two models:

$\mathcal{H}_2$: 2nd-order polynomials

$\mathcal{H}_{10}$: 10th-order polynomials



Compare out-of-sample errors of

$g_2 \in \mathcal{H}_2$ and $g_{10} \in \mathcal{H}_{10}$

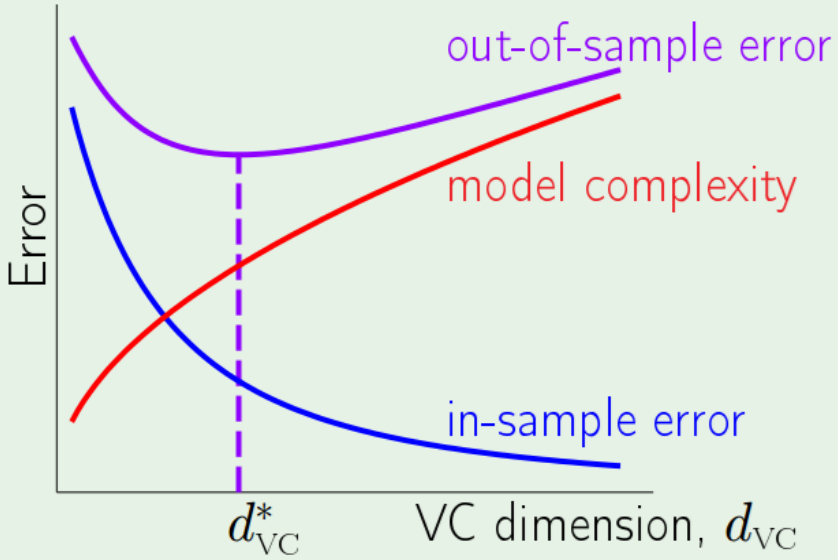
overfit measure: $E_{\text{out}}(g_{10}) - E_{\text{out}}(g_2)$

What the theory will achieve

Characterizing the feasibility of learning for infinite M

Characterizing the tradeoff:

Model complexity	$\uparrow$	E_{in}	$\downarrow$
Model complexity	$\uparrow$	$E_{\text{out}} - E_{\text{in}}$	$\uparrow$



How to have a good test error?

Bias-Variance Tradeoff

Start with E_{out}

$$E_{\text{out}}(g^{(\mathcal{D})}) = \mathbb{E}_{\mathbf{x}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$$

$$\begin{aligned}\mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(g^{(\mathcal{D})})\right] &= \mathbb{E}_{\mathcal{D}}\left[\mathbb{E}_{\mathbf{x}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]\right] \\ &= \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]\right]\end{aligned}$$

Now, let us focus on:

$$\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$$

The average hypothesis

To evaluate $\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$

we define the 'average' hypothesis $\bar{g}(\mathbf{x})$:

$$\bar{g}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}\left[g^{(\mathcal{D})}(\mathbf{x})\right]$$

Imagine **many** data sets $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_K$

$$\bar{g}(\mathbf{x}) \approx \frac{1}{K}\sum_{k=1}^{K} g^{(\mathcal{D}_k)}(\mathbf{x})$$

Using $\bar{g}(\mathbf{x})$

$$\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right] = \mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x}) + \bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]$$

$$= \mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2 + \left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2\right.$$

$$\left. + \; 2\,\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)\,\left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)\right]$$

$$= \mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right] + \left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2$$

Bias and variance

$$\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right] = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right]}_{\text{var}(\mathbf{x})} + \underbrace{\left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2}_{\text{bias}(\mathbf{x})}$$

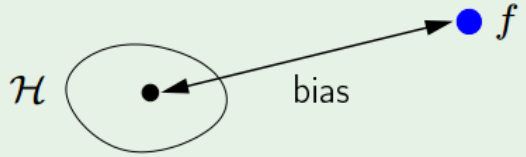
Therefore, $\mathbb{E}_{\mathcal{D}}\left[E_{\text{out}}(g^{(\mathcal{D})})\right] = \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - f(\mathbf{x})\right)^2\right]\right]$

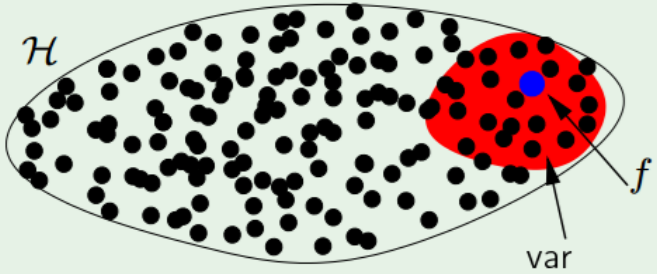
$$= \mathbb{E}_{\mathbf{x}}[\text{bias}(\mathbf{x}) + \text{var}(\mathbf{x})]$$

$$= \text{bias} + \text{var}$$

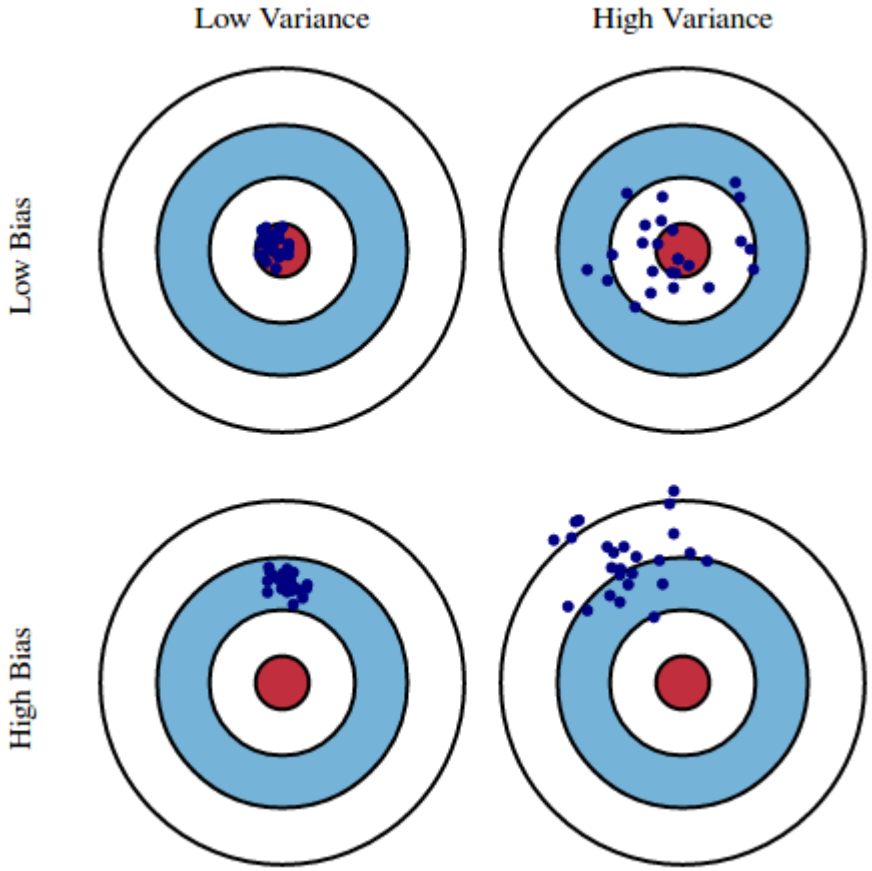
The tradeoff

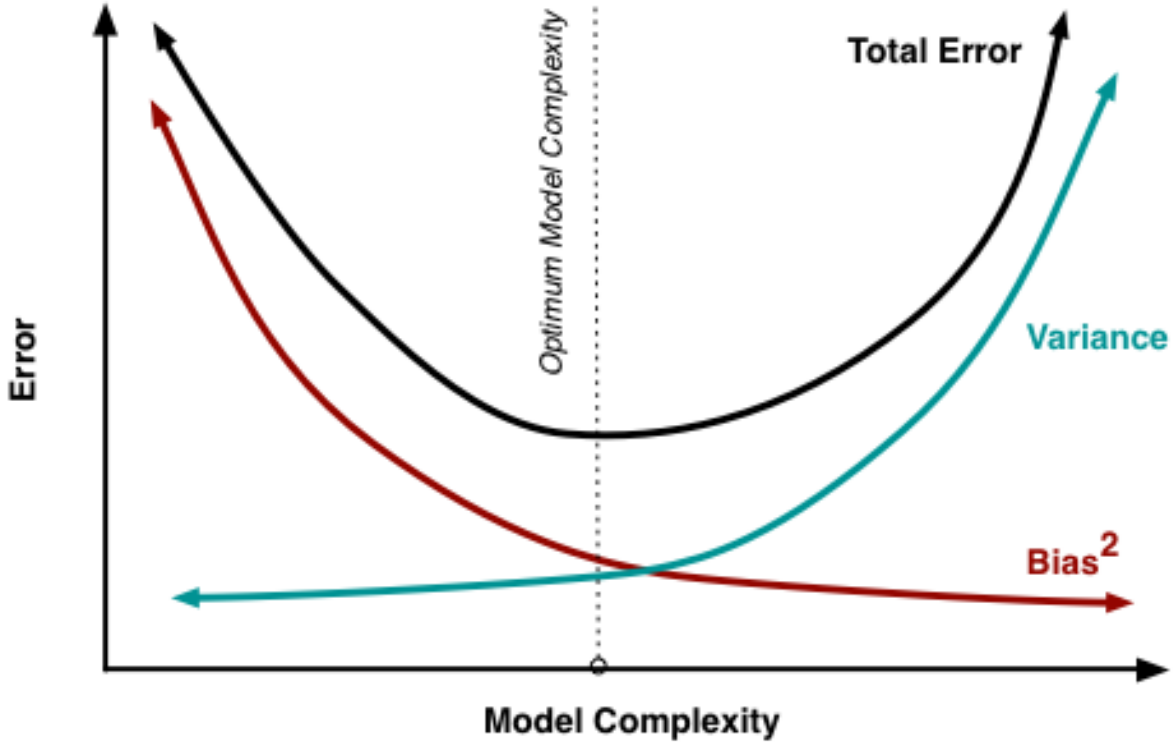
$$\text{bias} = \mathbb{E}_{\mathbf{x}}\left[\left(\bar{g}(\mathbf{x}) - f(\mathbf{x})\right)^2\right] \qquad \text{var} = \mathbb{E}_{\mathbf{x}}\left[\mathbb{E}_{\mathcal{D}}\left[\left(g^{(\mathcal{D})}(\mathbf{x}) - \bar{g}(\mathbf{x})\right)^2\right]\right]$$





$\downarrow$ $\qquad$ $\mathcal{H}\uparrow$ $\qquad$ $\uparrow$


Low Variance
High Variance
Low Bias
High Bias


Error
Optimum Model Complexity
Total Error
Variance
Bias^2
Model Complexity

Example: sine target

f

$f : [-1, 1] \to \mathbb{R} \qquad f(x) = \sin(\pi x)$

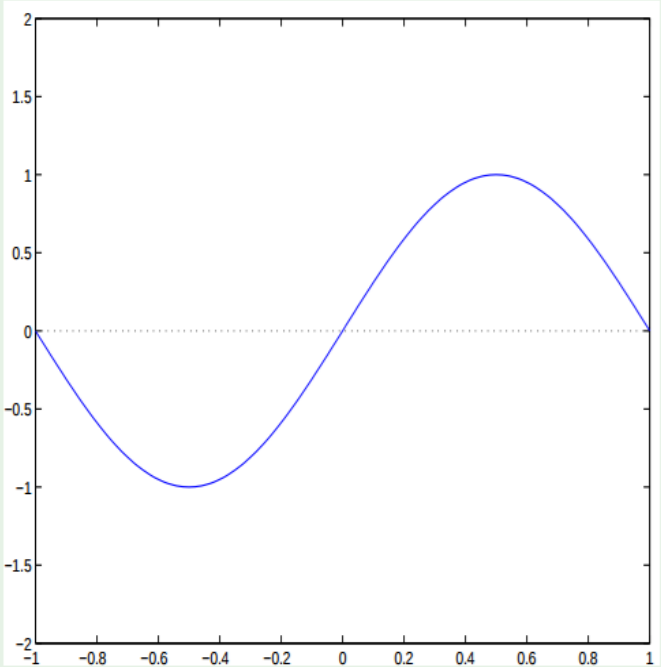
Only two training examples! $N = 2$

Two models used for learning:

$\mathcal{H}_0$: $h(x) = b$

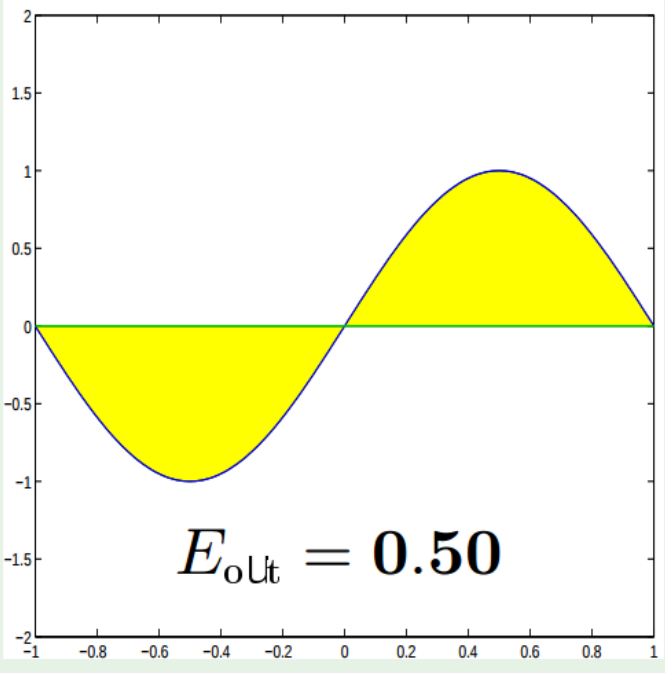
$\mathcal{H}_1$: $h(x) = ax + b$

Which is better, $\mathcal{H}_0$ or $\mathcal{H}_1$?

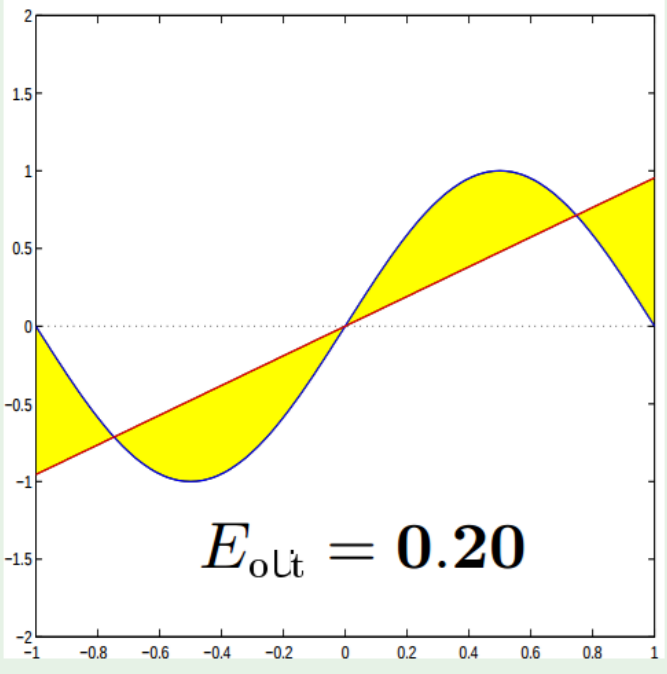


Approximation - $\mathcal{H}_0$ versus $\mathcal{H}_1$

$\mathcal{H}_0$

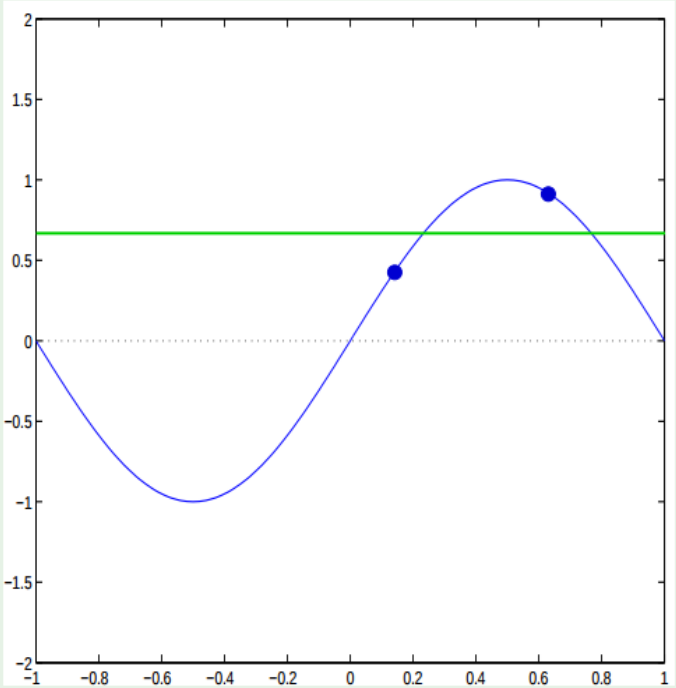


$\mathcal{H}_1$

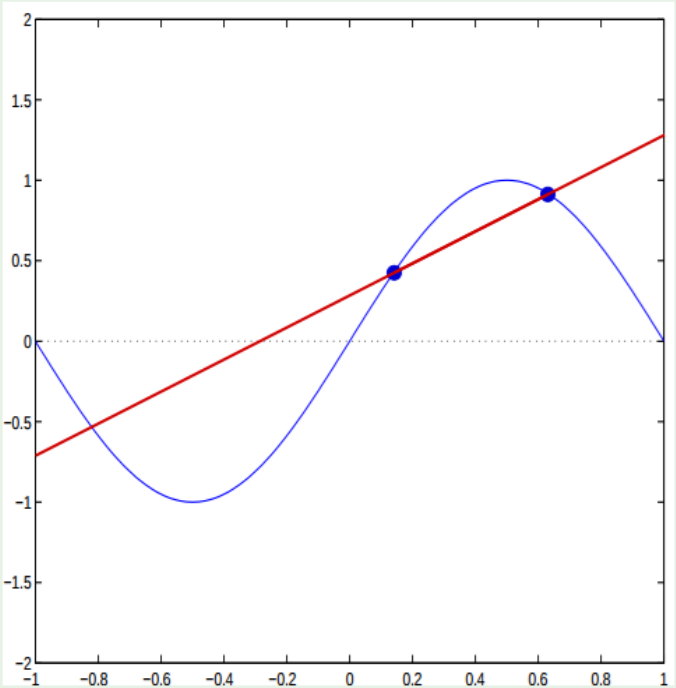


Learning - $\mathcal{H}_0$ versus $\mathcal{H}_1$

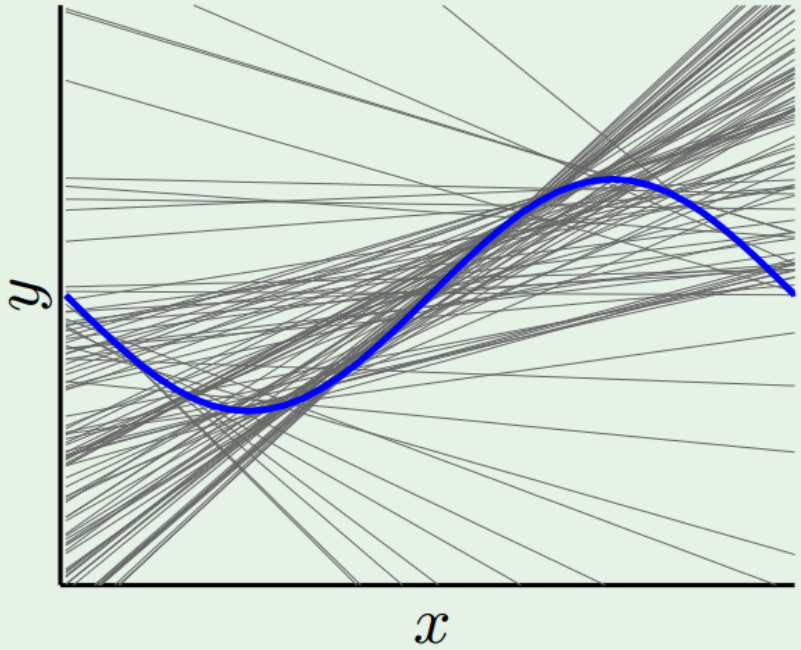
$\mathcal{H}_0$

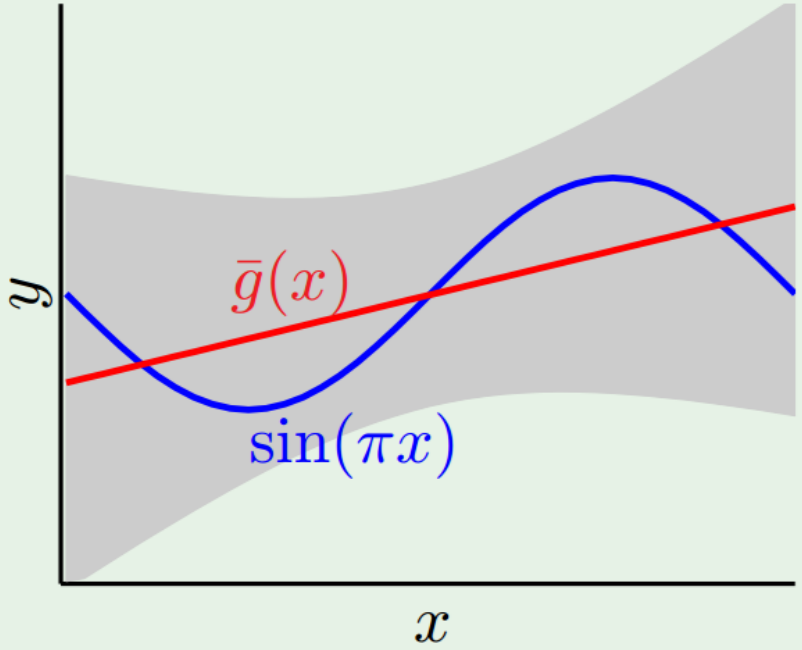


$\mathcal{H}_1$

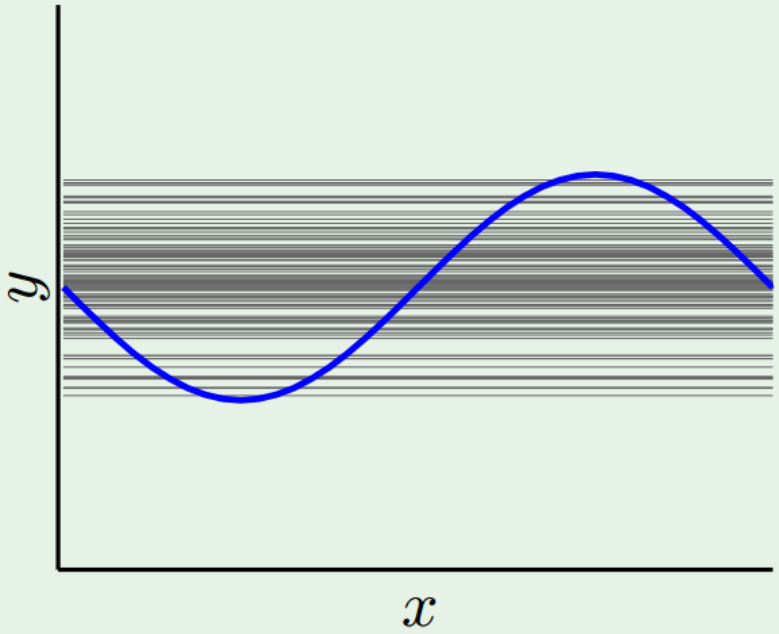


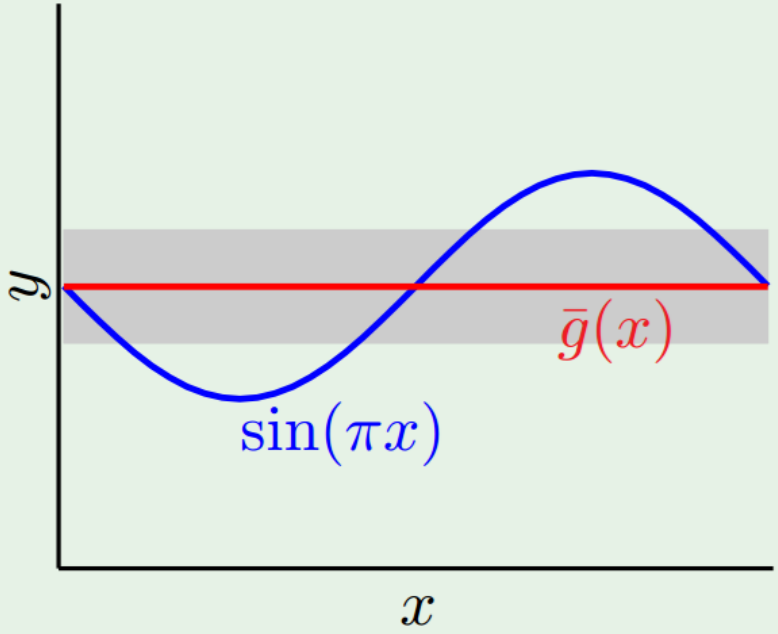
Bias and variance - $\mathcal{H}_1$



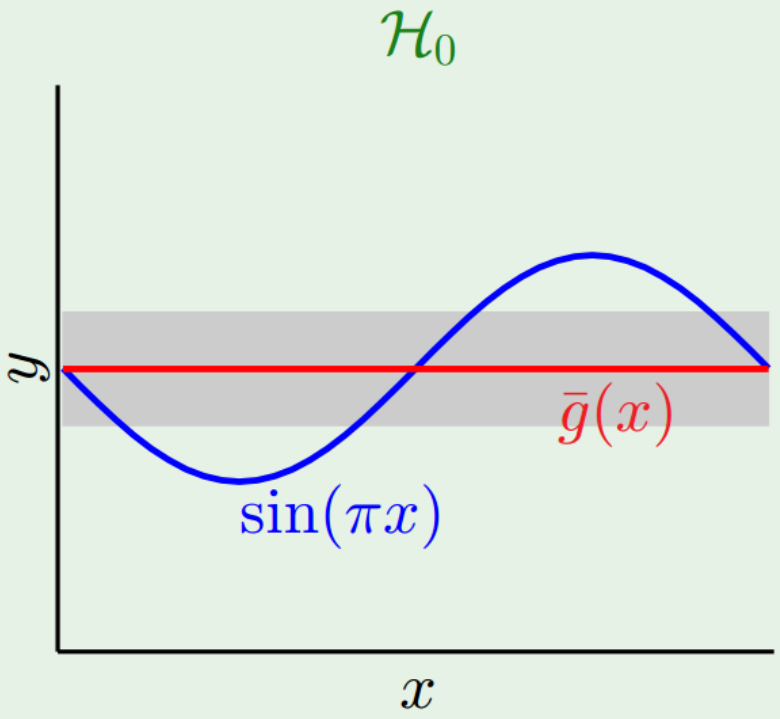


Bias and variance - $\mathcal{H}_0$

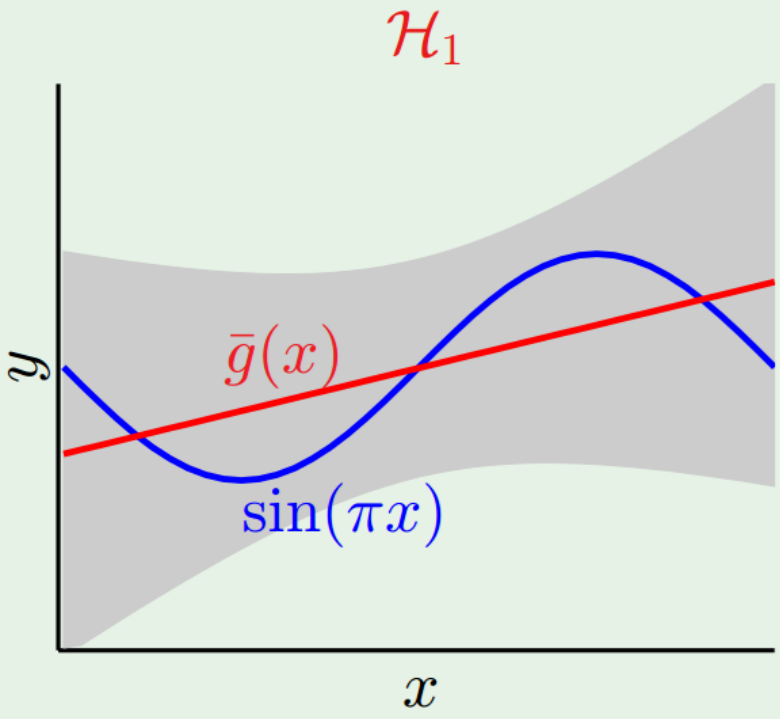




and the winner is ...



bias = **0.50** var = **0.25**



bias = **0.21** var = **1.69**

Expected E_{out} and E_{in}

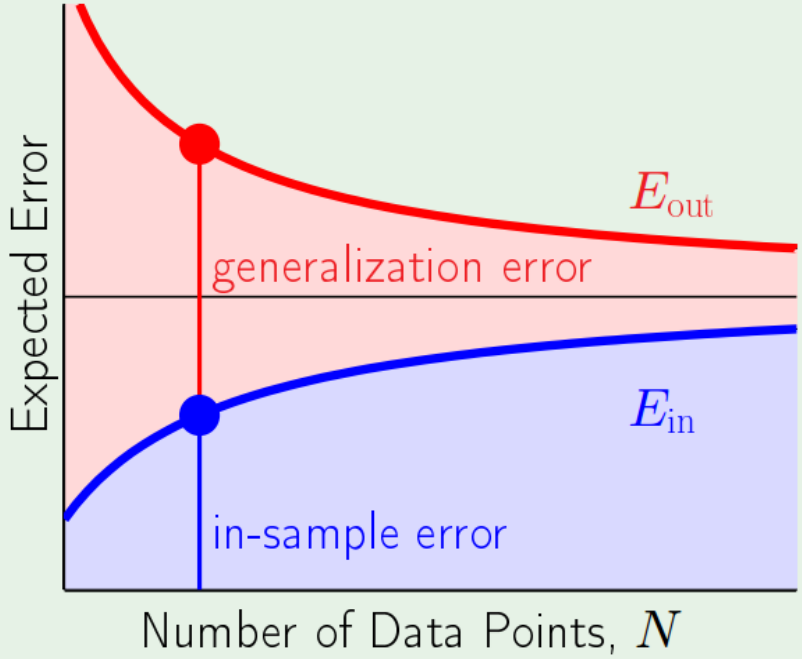
Data set $\mathcal{D}$ of size N

Expected out-of-sample error $\mathbb{E}_{\mathcal{D}}[E_{\text{out}}(g^{(\mathcal{D})})]$

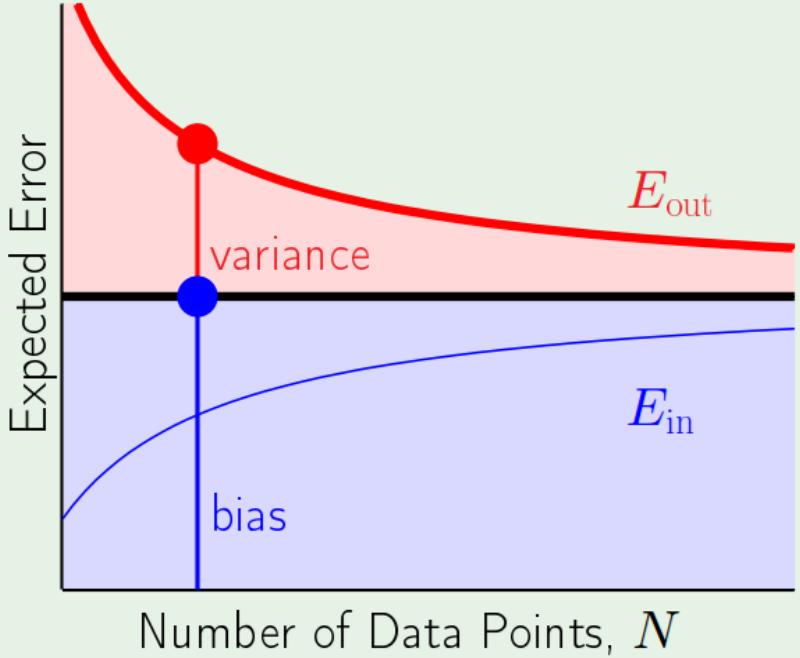
Expected in-sample error $\mathbb{E}_{\mathcal{D}}[E_{\text{in}}(g^{(\mathcal{D})})]$

How do they vary with N?

VC versus bias-variance

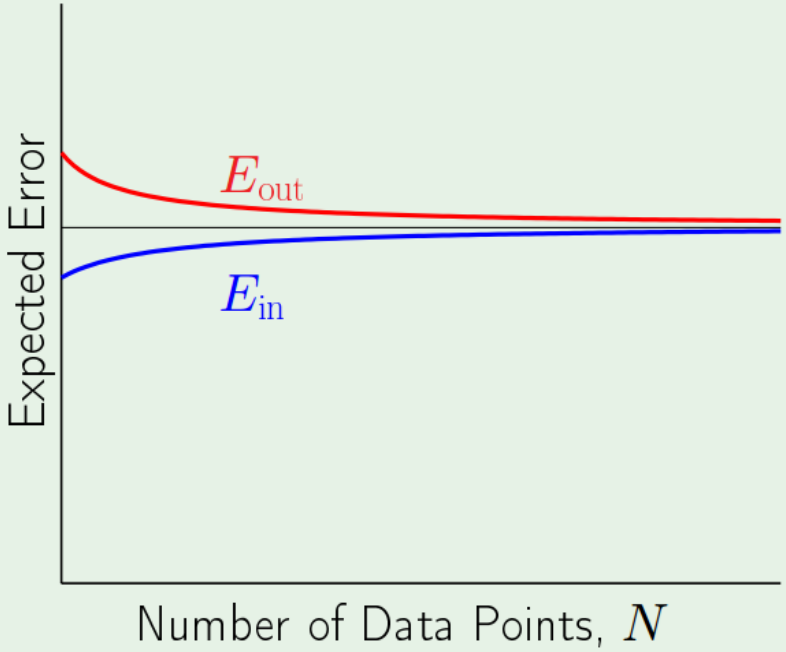


VC analysis

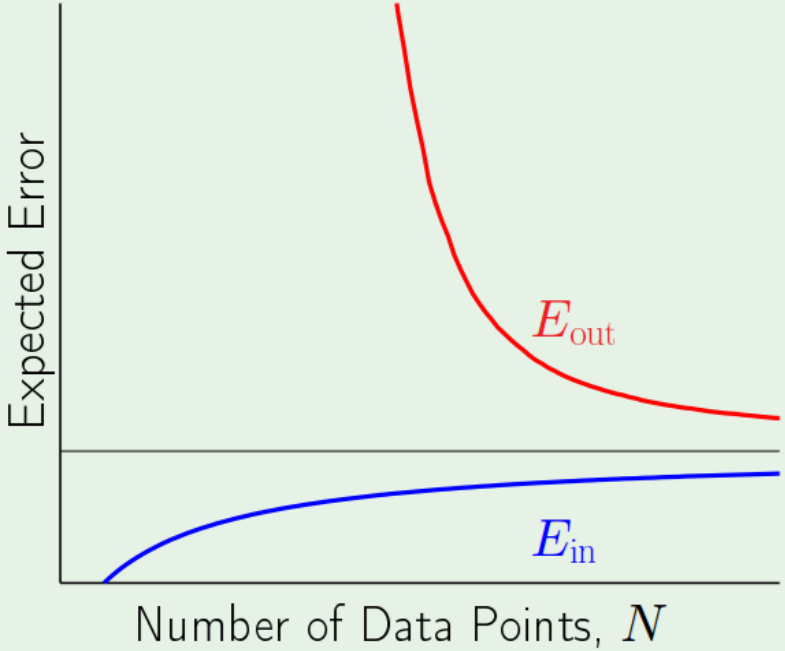


bias-variance

The curves



Simple Model



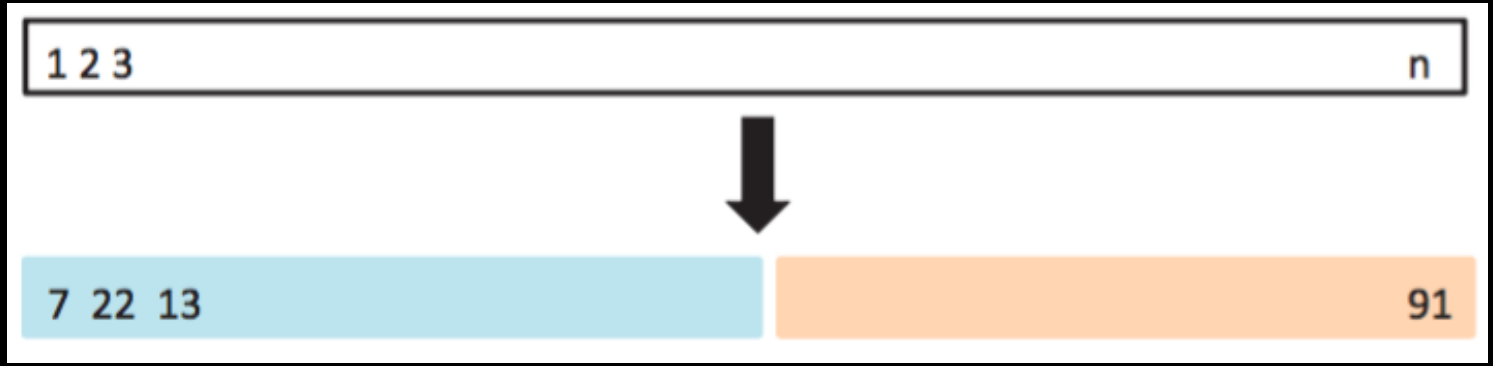
Complex Model

Validation

Validation Set Approach

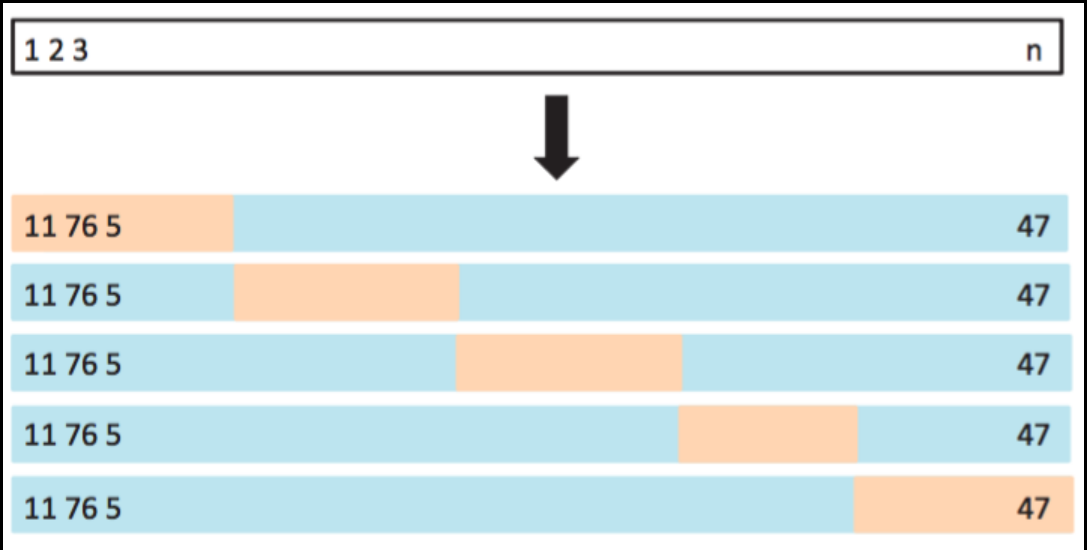
It involves randomly dividing the available set of observations into two parts.

1. a training set.
2. a validation set or hold-out set.



k-Fold Cross Validation

- This approach involves randomly dividing the set of observations into k groups, or folds, of approximately equal size.
- The first fold is treated as a validation set, and the method is fit on the remaining $k-1$ folds.
- The k-fold CV estimate is computed by averaging these values



$$CV_k = \frac{1}{k}\sum_{i=1}^{k} MSE_i.$$

Type of Machine Learning

Basic premise of learning

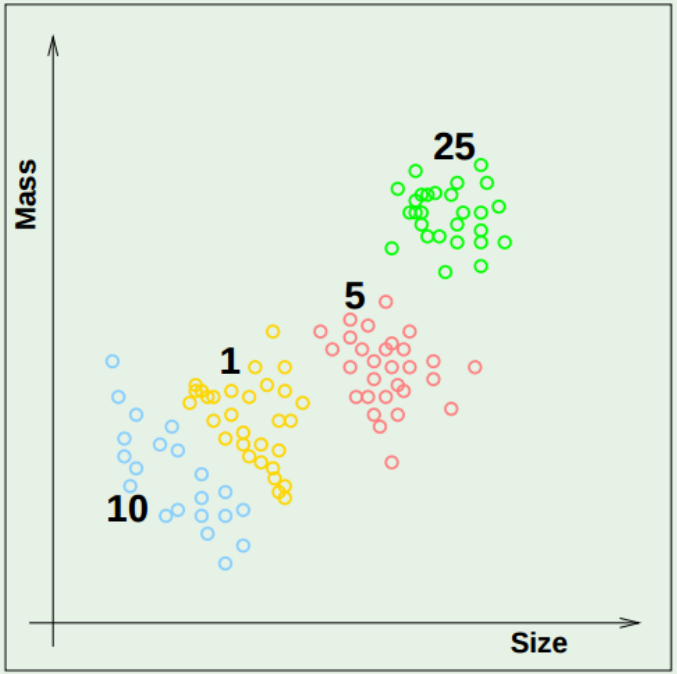
"using a set of observations to uncover an underlying process"

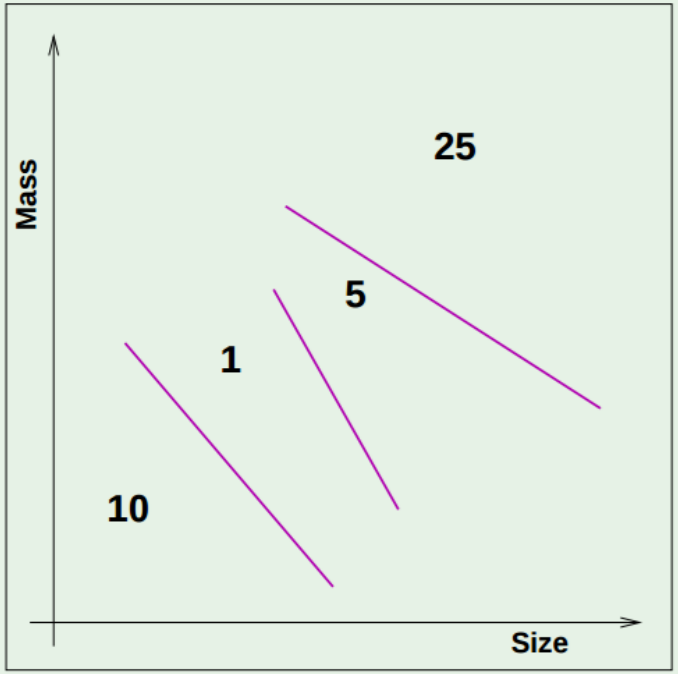
broad premise $\Longrightarrow$ many variations

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

Supervised learning

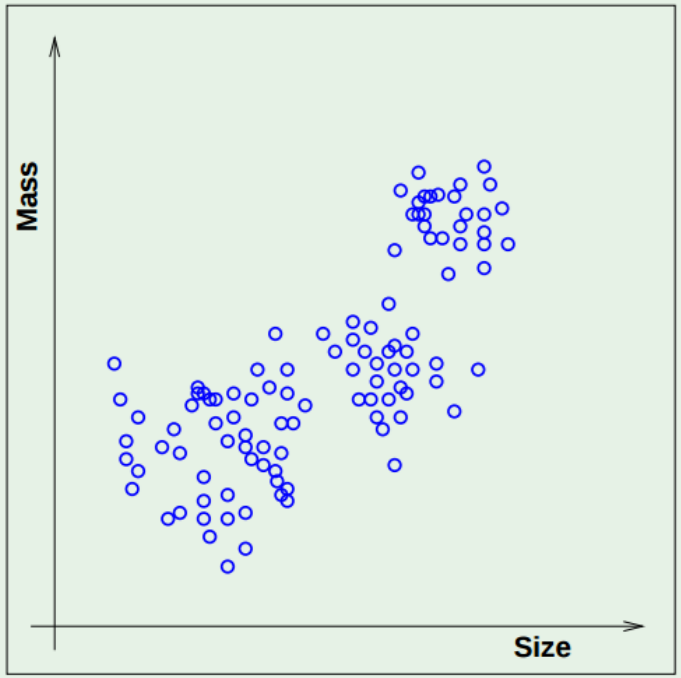
Example from vending machines – **coin recognition**





Unsupervised learning

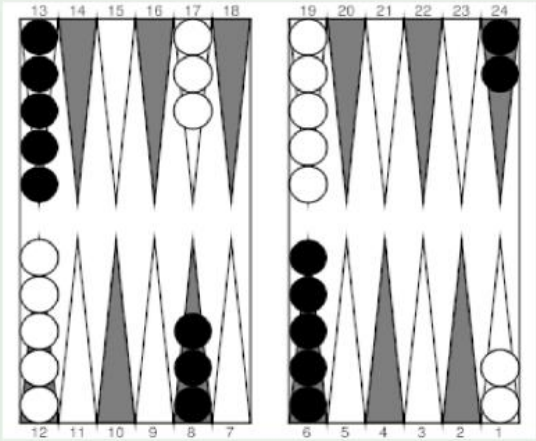
Instead of (input,correct output), we get (input, ?)



Reinforcement learning

Instead of (input, correct output),

we get (input, *some* output, grade for this output)



The world champion was a neural network!

Thank you, question?