

CSE 417T

Introduction to Machine Learning

Lecture 9

Instructor: Chien-Ju (CJ) Ho

Logistics

- Homework 2 is due on **Feb 24 (Thursday)**
- Return of Homework
 - We plan to return each homework around 2 weeks after the deadline
 - Regrade requests
 - You will have up to 7 days to submit regrade requests
 - the regrade period might be shortened if there are schedule constraints
 - We might check the entire homework for each request, so the grades might go down as well if we find new mistakes
- Exam 1: **Mar 10 (Thursday)**
 - LFD Chap 1 to 5 (The entire hardcopy of the textbook)
 - Covid-permitting
 - Timed exam (75 min) during lecture time in the classroom
 - Closed-book exam with 2 letter-size cheat sheets allowed (4 pages in total)
 - No format limitations (it can be typed, written, or a combination)

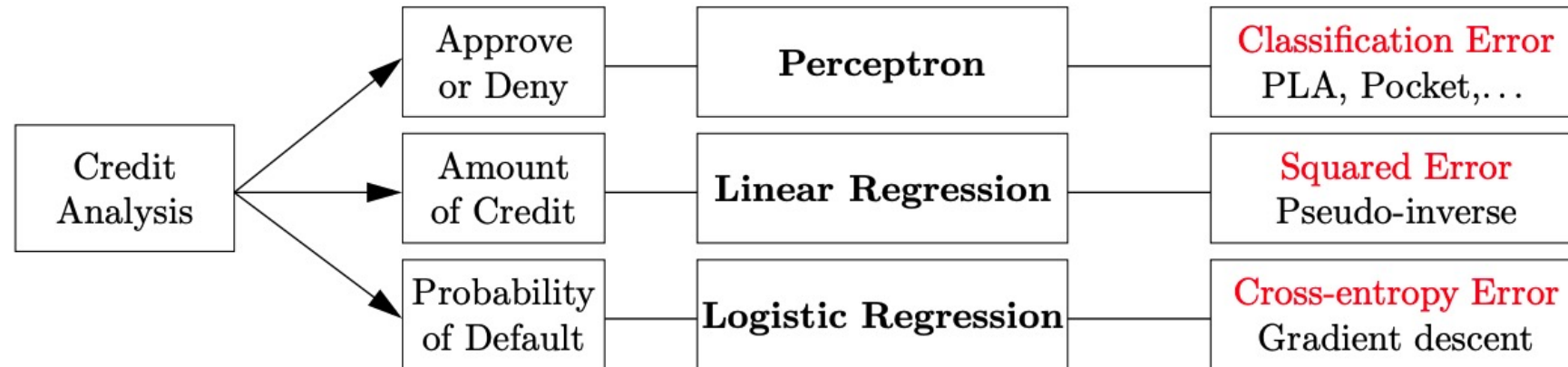
Recap

Linear Models

This is why it's called linear models



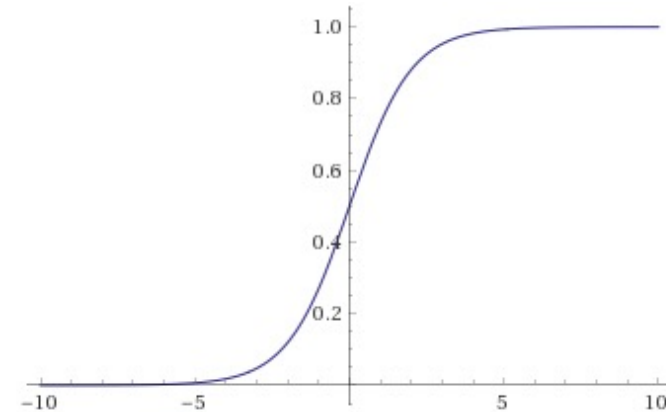
- H contains hypothesis $h(\vec{x})$ as **some function of $\vec{w}^T \vec{x}$**



- Algorithm:
 - Focus on $g = \operatorname{argmin}_{h \in H} E_{in}(h)$
 - **Gradient descent** is one of the common optimization algorithms

Logistic Regression

- Predict a probability
 - Interpreting $h(\vec{x}) \in [0,1]$ as the prob for $y = +1$ given \vec{x}
- Hypothesis set $H = \{h(\vec{x}) = \theta(\vec{w}^T \vec{x})\}$
 - $\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$
- Algorithm
 - Find $g = \operatorname{argmin}_{h \in H} E_{in}(h)$
- Two key questions
 - How to define $E_{in}(h)$?
 - How to perform the optimization (minimizing E_{in})?



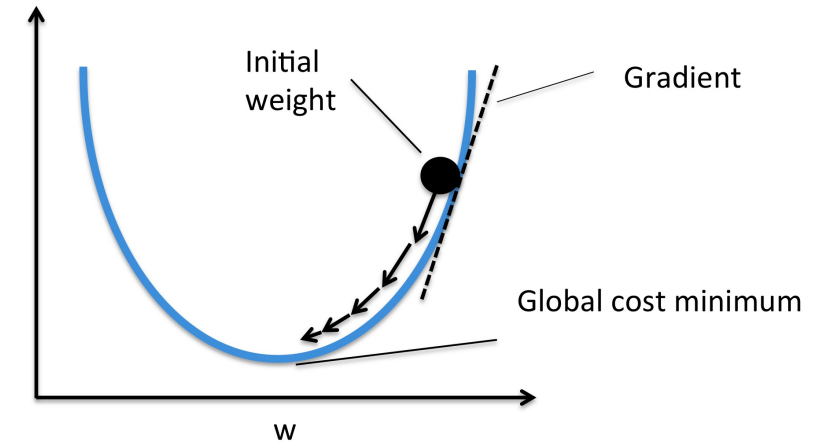
Define $E_{in}(\vec{w})$: Cross-Entropy Error

$$E_{in}(\vec{w}) = \frac{1}{N} \sum_{n=1}^N \ln(1 + e^{-y_n \vec{w}^T \vec{x}_n})$$

- Minimizing cross entropy error is the same as maximizing likelihood
- Likelihood: $\Pr(D|\vec{w})$
 - $\vec{w}^* = \operatorname{argmax}_{\vec{w}} \Pr(D|\vec{w})$ (maximizing likelihood)
 $= \operatorname{argmin}_{\vec{w}} E_{in}(\vec{w})$ (minimizing cross-entropy error)

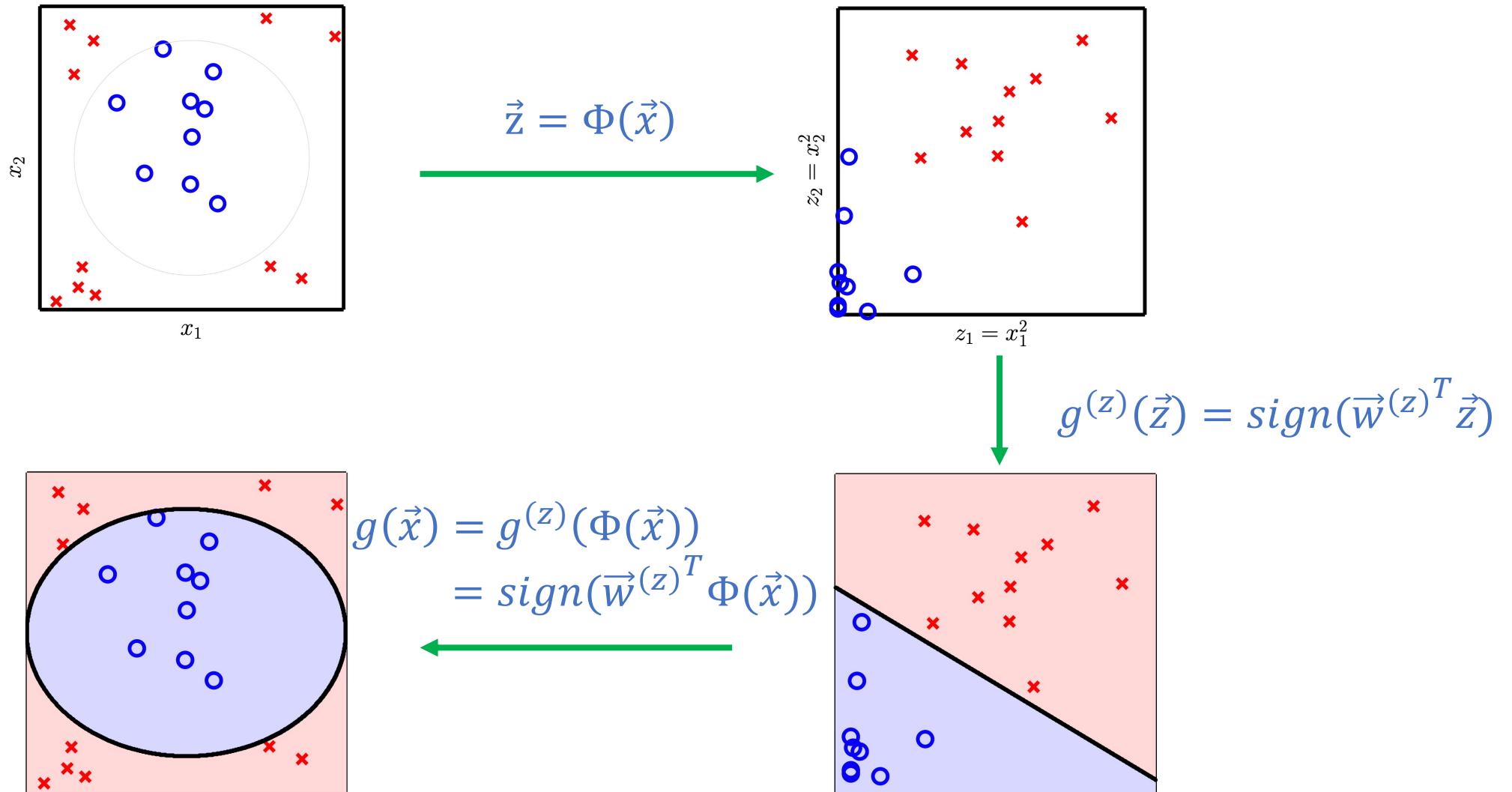
Optimizing $E_{in}(\vec{w})$: Gradient Descent

- Gradient descent algorithm
 - Initialize $\vec{w}(0)$
 - For $t = 0, \dots$
 - $\vec{w}(t+1) \leftarrow \vec{w}(t) - \eta \nabla_{\vec{w}} E_{in}(\vec{w}(t))$
 - Terminate if the stop conditions are met
 - Return the final weights
- Stochastic gradient decent
 - Replace the update step:
 - Randomly choose n from $\{1, \dots, N\}$
 - $\vec{w}(t+1) \leftarrow \vec{w}(t) - \eta \nabla_{\vec{w}} e_n(\vec{w}(t))$

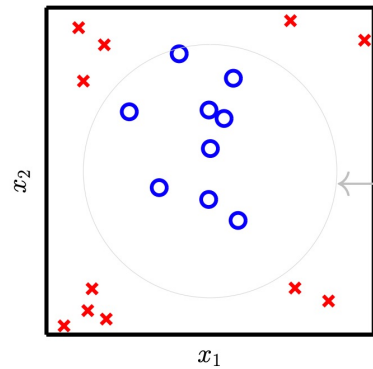


Works for functions where gradient exists everywhere

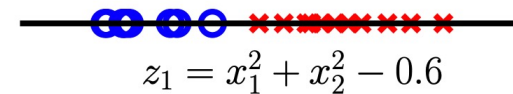
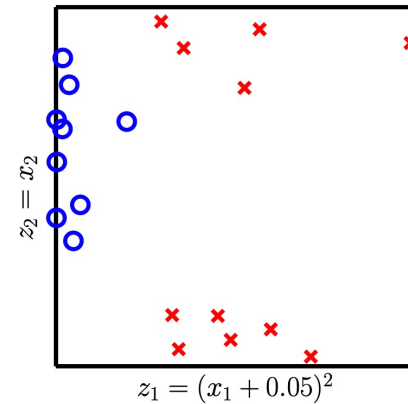
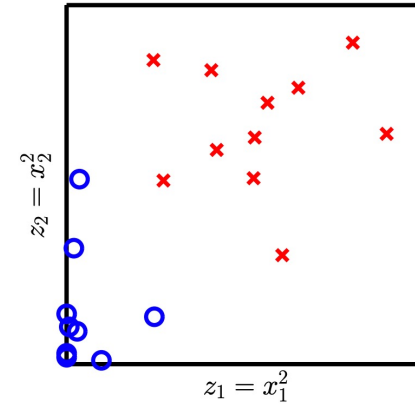
Nonlinear Transformation



How to Choose Feature Transform Φ



$$x_1^2 + x_2^2 = 0.6$$



Something Seems Wrong!

Must choose Φ
BEFORE looking at the data

Otherwise, you are doing “data snooping”

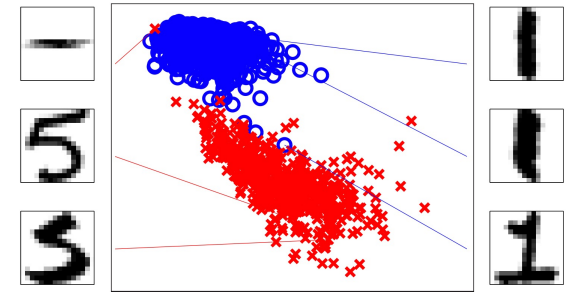
The hypothesis set H is as large as anything your brain can think of

Today's Lecture

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.
Let me know if you spot errors.

Choose Φ Before Seeing Data

- Rely on domain knowledge (feature engineering)
 - Handwriting digit recognition example
- Use common sets of feature transformation
 - Polynomial transformation
 - 2nd order Polynomial transformation
 - $\vec{x} = (1, x_1, x_2)$
 - $\Phi_2(\vec{x}) = (1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$
 - Pros: more powerful (contains circle, ellipse, hyperbola, etc)
 - Cons: 2-d \Rightarrow 5-d
 - More computation/storage
 - Worse generalization error



The VC dimension of d-dim perceptron is $d+1$

Q-th Order Polynomial Transform

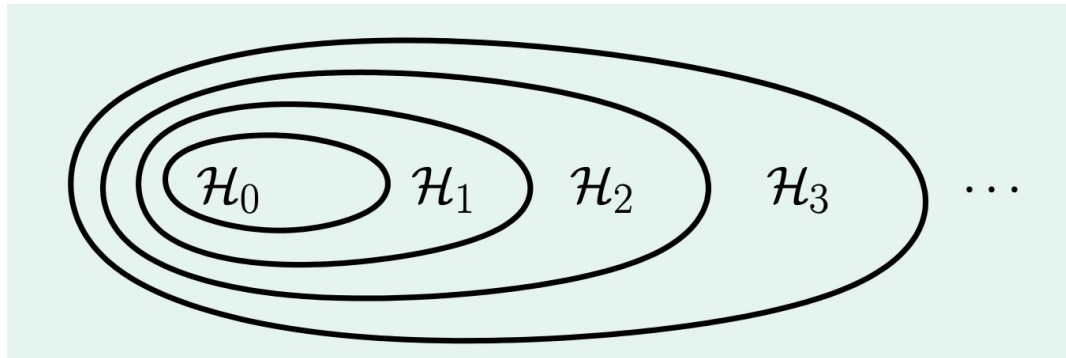
- $\vec{x} = (1, x_1, \dots, x_d)$
- From 1-st order to Q-th order polynomial transform:
 - $\Phi_1(\vec{x}) = \vec{x}$
 - $\Phi_2(\vec{x}) = (\Phi_1(\vec{x}), x_1^2, x_1x_2, x_1x_3, \dots, x_d^2)$
 - ...
 - $\Phi_Q(\vec{x}) = (\Phi_{Q-1}(\vec{x}), x_1^Q, x_1^{Q-1}x_2, \dots, x_d^Q)$
- Number of elements in $\Phi_Q(\vec{x})$

Q-th Order Polynomial Transform

- $\vec{x} = (1, x_1, \dots, x_d)$
- From 1-st order to Q-th order polynomial transform:
 - $\Phi_1(\vec{x}) = \vec{x}$
 - $\Phi_2(\vec{x}) = (\Phi_1(\vec{x}), x_1^2, x_1x_2, x_1x_3, \dots, x_d^2)$
 - ...
 - $\Phi_Q(\vec{x}) = (\Phi_{Q-1}(\vec{x}), x_1^Q, x_1^{Q-1}x_2, \dots, x_d^Q)$
- Number of elements in $\Phi_Q(\vec{x})$
 - $\binom{Q+d}{Q}$

Structural Hypothesis Sets

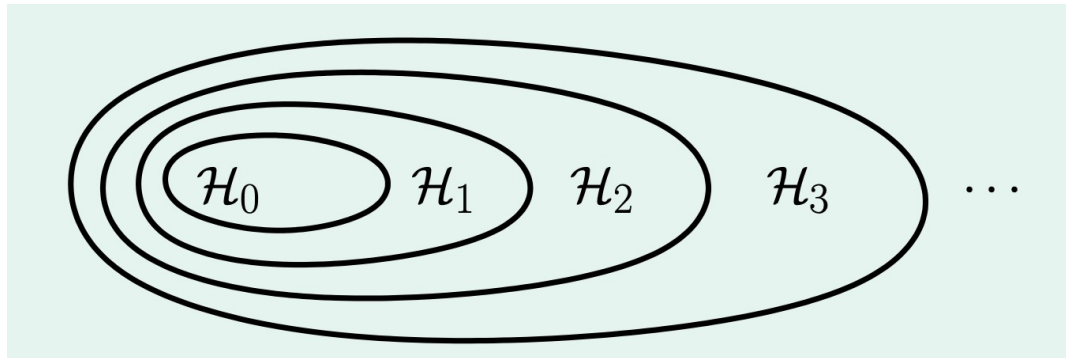
- Let H_Q be the linear model for the $\Phi_Q(\vec{x})$ space



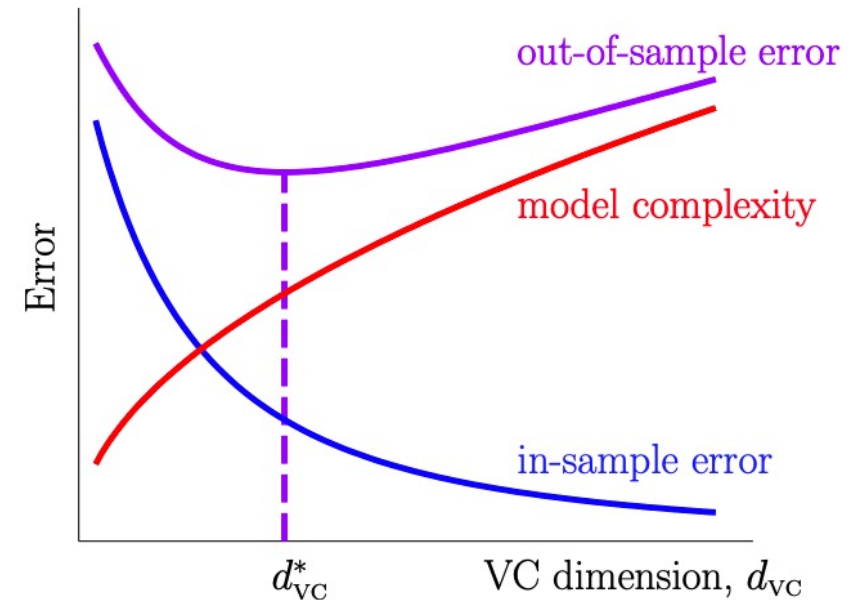
- Let $g_Q = \operatorname{argmin}_{h \in H_Q} E_{in}(h)$
 - $H_0 \quad H_1 \quad H_2 \dots$
 - $d_{vc}(H_0) \quad d_{vc}(H_1) \quad d_{vc}(H_2) \dots$
 - $E_{in}(g_0) \quad E_{in}(g_1) \quad E_{in}(g_2) \dots$

Structural Hypothesis Sets

- Let H_Q be the linear model for the $\Phi_Q(\vec{x})$ space



- Let $g_Q = \operatorname{argmin}_{h \in H_Q} E_{in}(h)$
 - $H_0 \subset H_1 \subset H_2 \dots$
 - $d_{vc}(H_0) \leq d_{vc}(H_1) \leq d_{vc}(H_2) \dots$
 - $E_{in}(g_0) \geq E_{in}(g_1) \geq E_{in}(g_2) \dots$



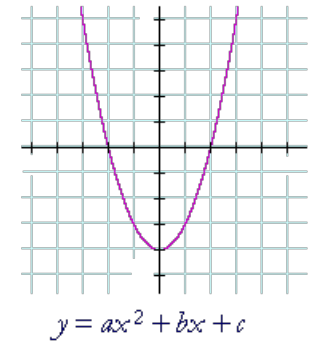
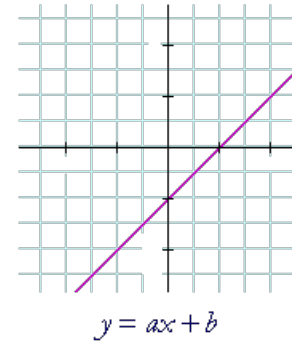
Overfitting

[Adapted from the slides by Malik Magdon-Ismail]

Setup of the Discussion

- Regression with polynomial transform

- Input: 1-dimensional x
- $\Phi_Q(x) = (1, x, x^2, x^3, \dots, x^Q)$
- $H_Q = \{h(x) = w_0 + w_1x + w_2x^2 + \dots + w_Qx^Q\}$

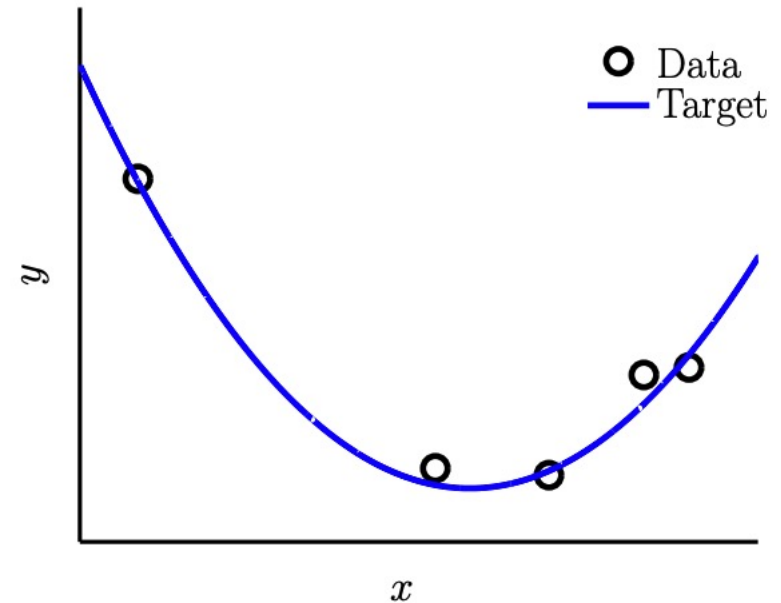


- Q th-order polynomial fit

- Solve linear regression on the $\Phi_Q(\vec{x})$ space using H_Q
- Looking to minimize E_{in} : $g_Q = \operatorname{argmin}_{h \in H_Q} E_{in}(h)$

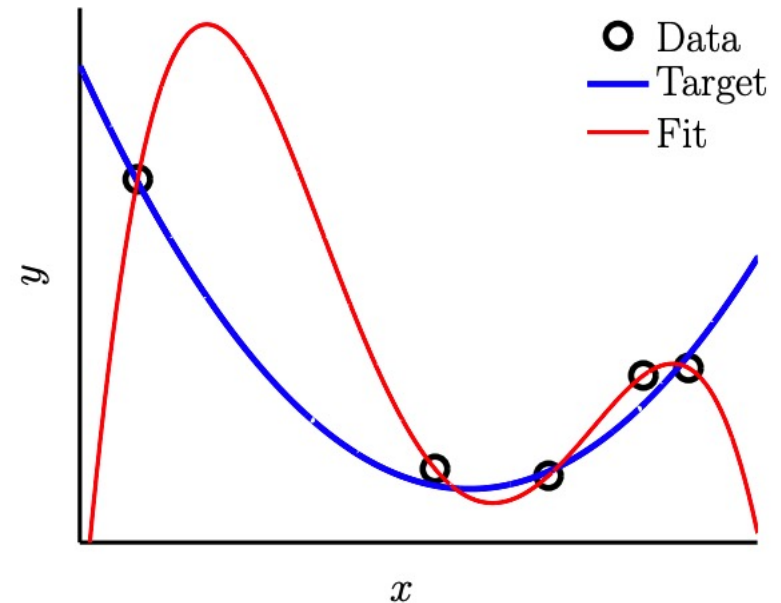
A Simple Example

- Target f : 4th order function
- # data points: $N = 5$
- Small noise:
 - $y = f(x) + \epsilon$ with small ϵ
- 4th order polynomial fit
 - $h(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$
 - Find $g_4 = \operatorname{argmin}_h E_{in}(h)$



A Simple Example

- Target f : 4th order function
- # data points: $N = 5$
- Small noise:
 - $y = f(x) + \epsilon$ with small ϵ
- 4th order polynomial fit
 - $h(x) = w_0 + w_1x + w_2x^2 + w_3x^3 + w_4x^4$
 - Find $g_4 = \operatorname{argmin}_h E_{in}(h)$



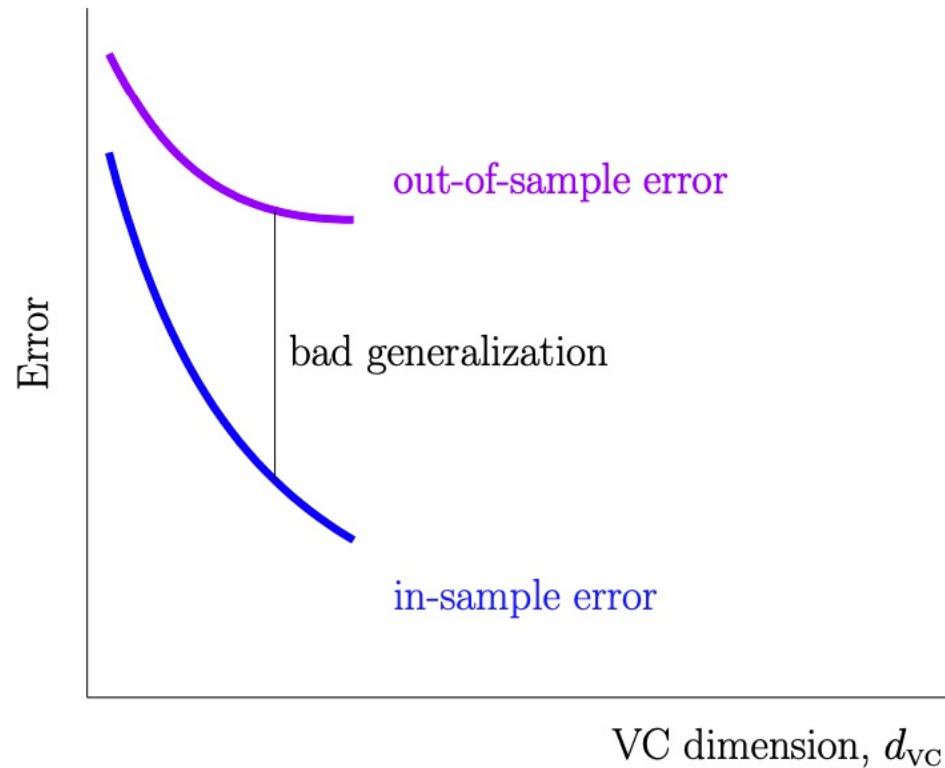
Classical overfitting: $E_{in} = 0$, but lead to a large E_{out}

Fitting the **noise** instead of the target

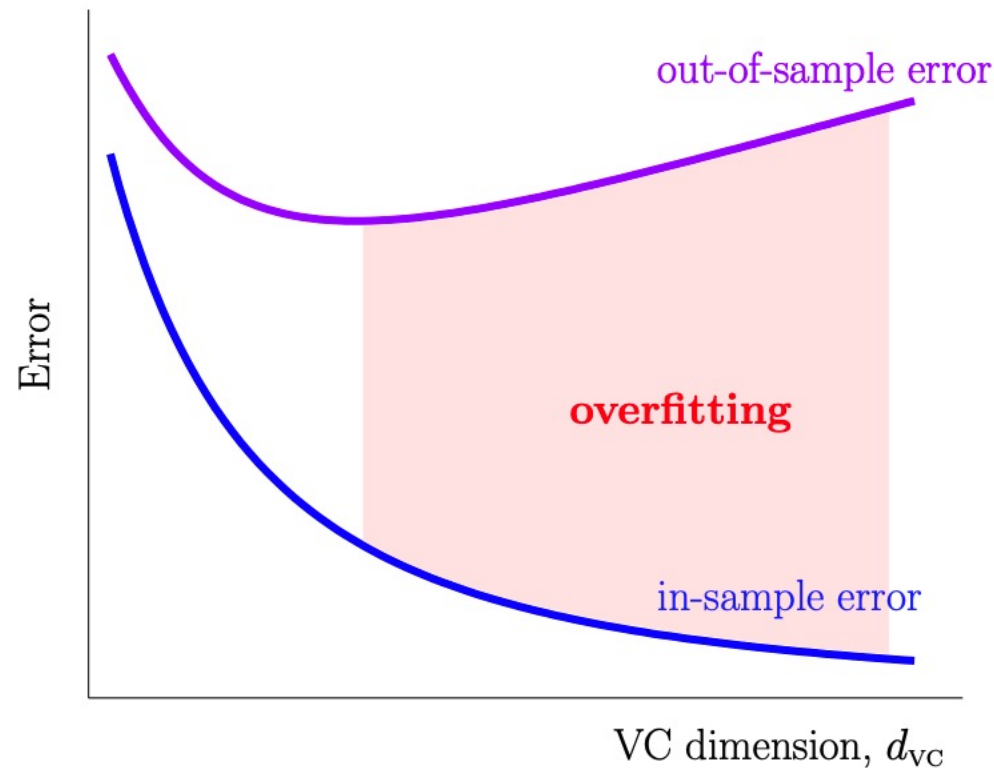
What is Overfitting?

Fitting the data **more** than is **warranted**

Overfitting is Not Just Bad Generalization



Overfitting is Not Just Bad Generalization



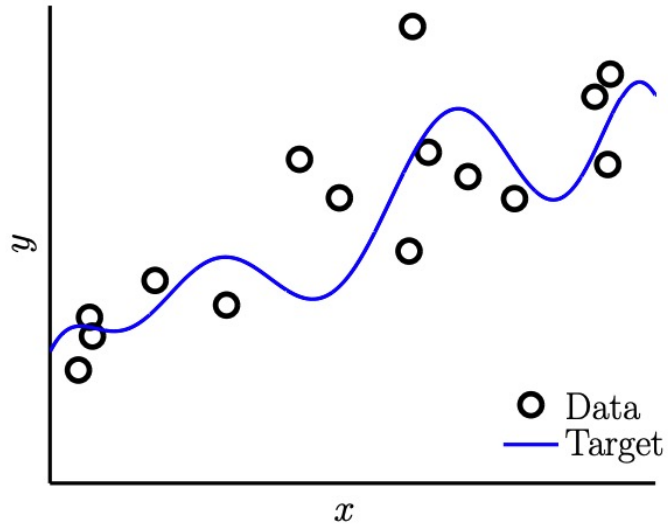
Overfitting

Going for **lower and lower** E_{in} results in **higher and higher** E_{out}

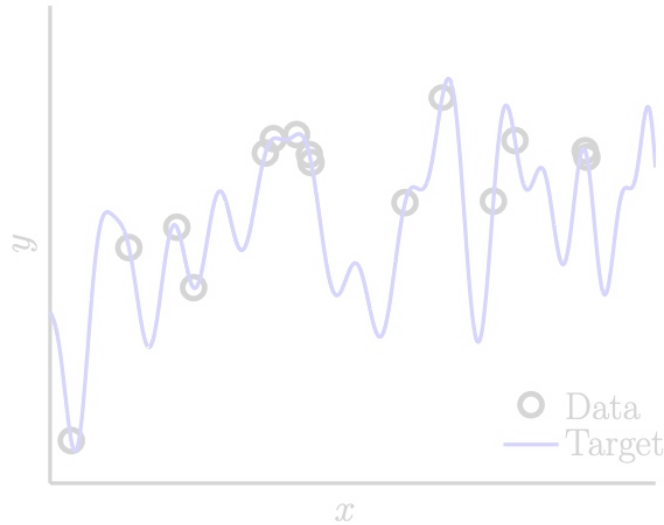
Case Study:

2^{nd} vs 10^{th} Order Polynomial Fit

N=15



10th order f with noise.



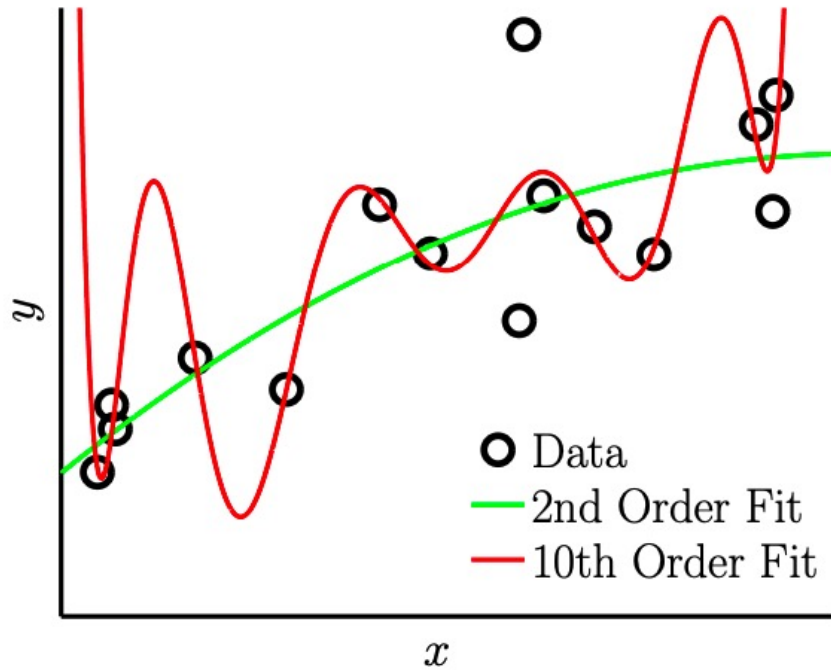
50th order f with no noise.

H_2 : 2nd order polynomial fit

H_{10} : 10th order polynomial fit

Which model would you choose for the left problem and why?

Target Function: 10th Order f with Noise



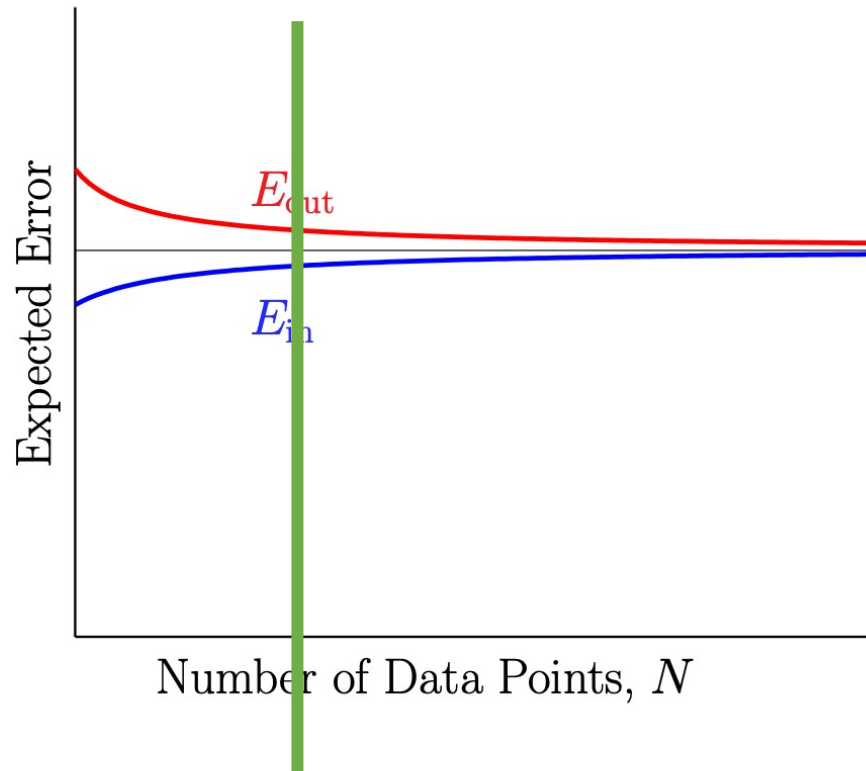
simple noisy target

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

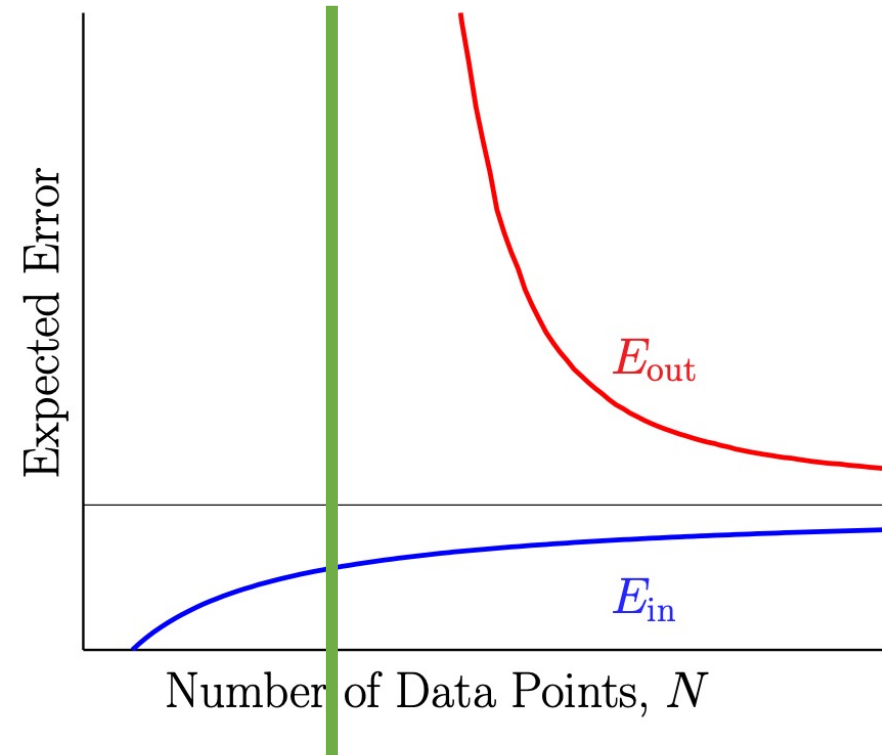
- Irony of two learners **Red** and **Green**
- Both know the target is 10th order
- **Red** chooses H_{10}
- **Green** chooses H_2
- **Green** outperforms **Red**

Why is H_2 Better than H_{10} ?

Learning curve for H_2

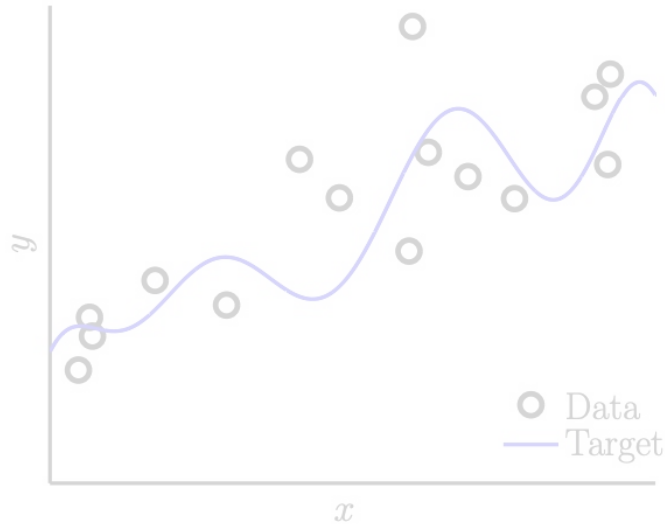


Learning curve for H_{10}

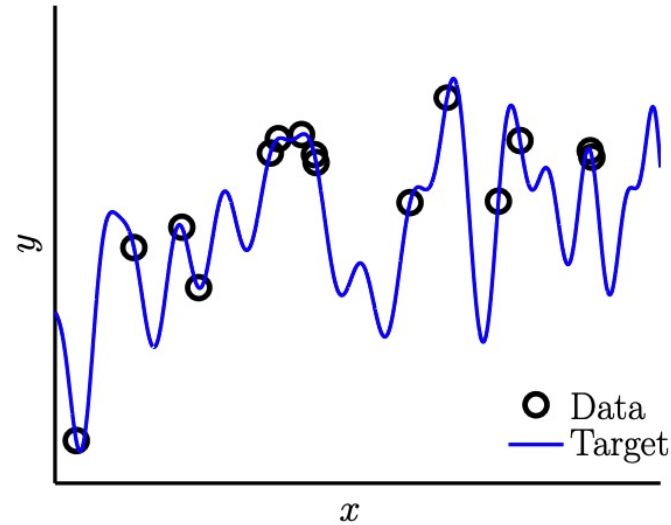


When N is small, $E_{out}(g_{10}) \geq E_{out}(g_2)$

N=15



10th order f with noise.



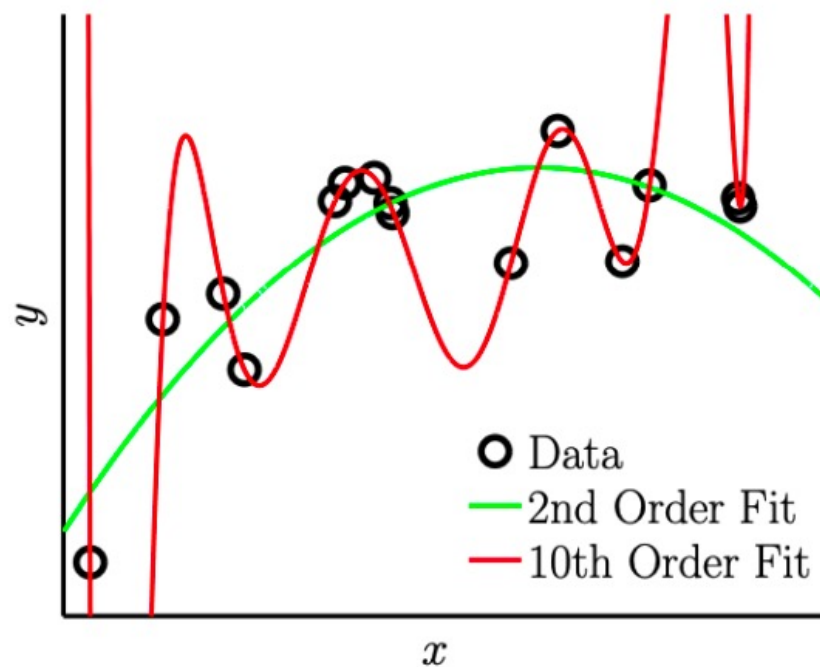
50th order f with no noise.

H_2 : 2nd order polynomial fit

H_{10} : 10th order polynomial fit

Which model do you choose for the right problem and why?

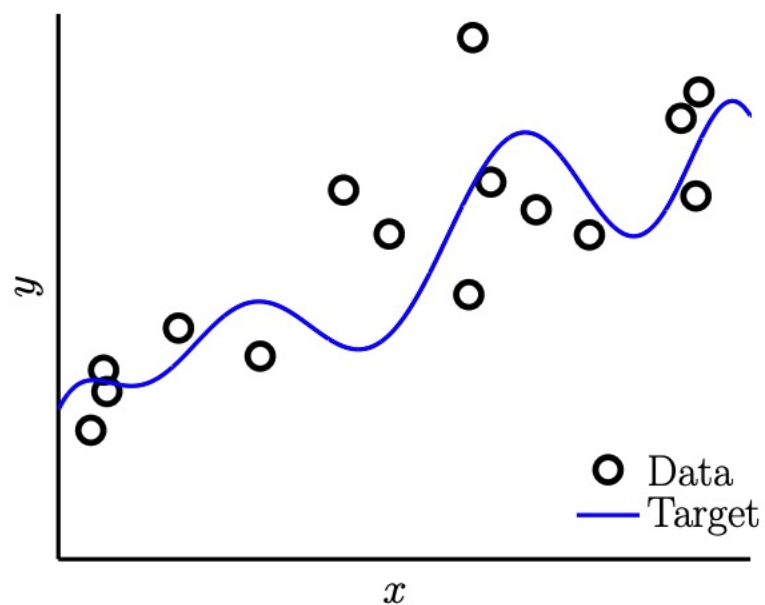
Simpler H is better even for complex target with **no noise**



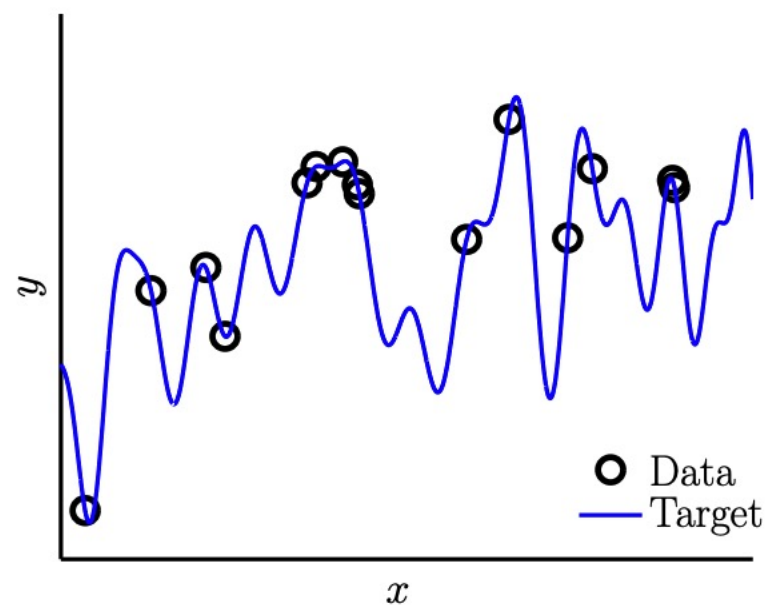
complex noiseless target

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

Is There Really “No Noise”?

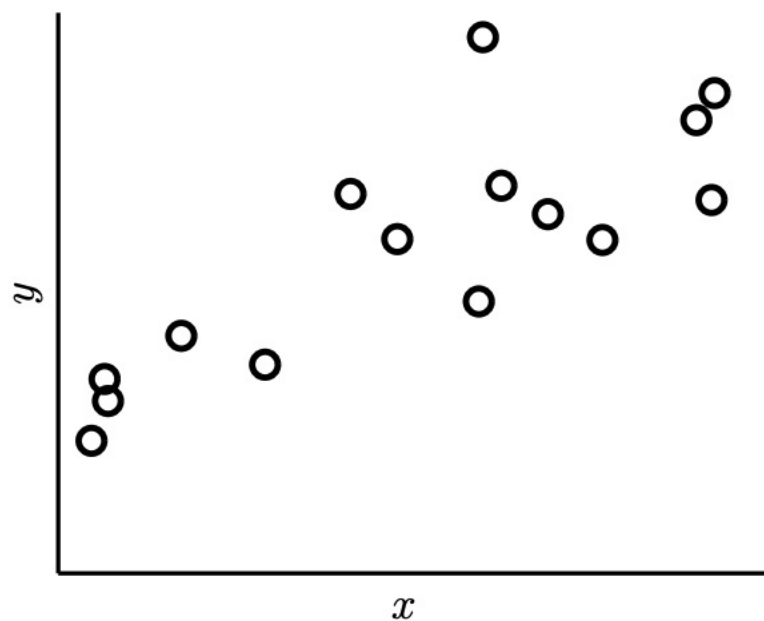


Simple f with noise.

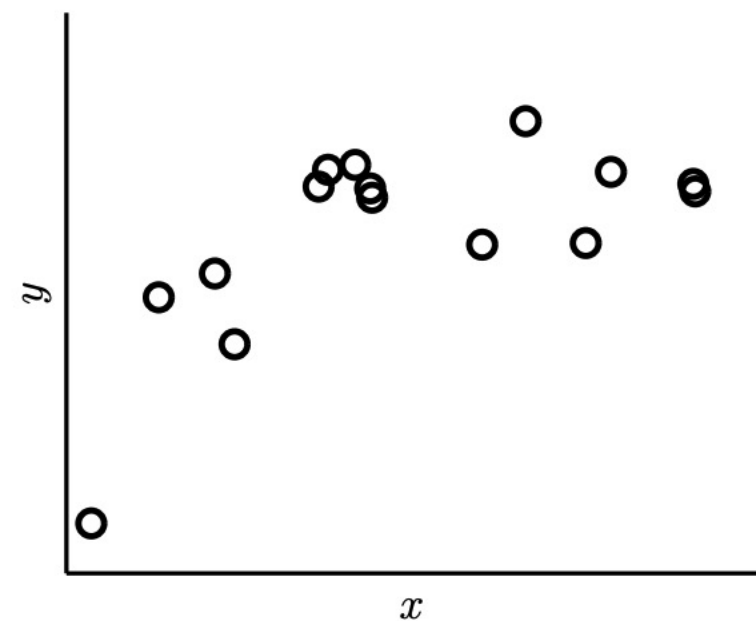


Complex f with no noise.

Is There Really “No Noise”?



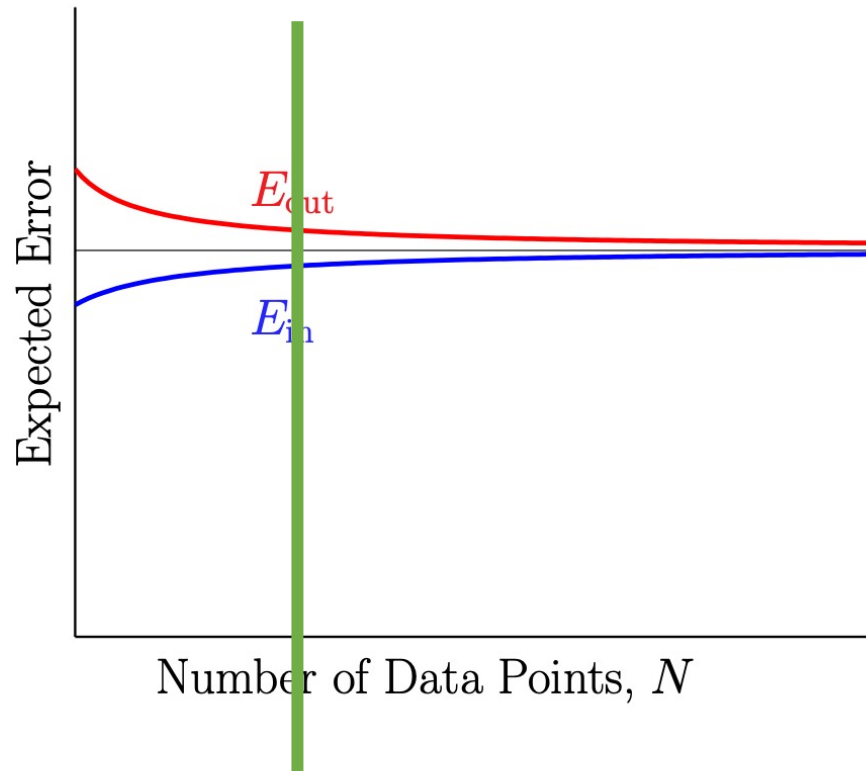
Simple f with noise.



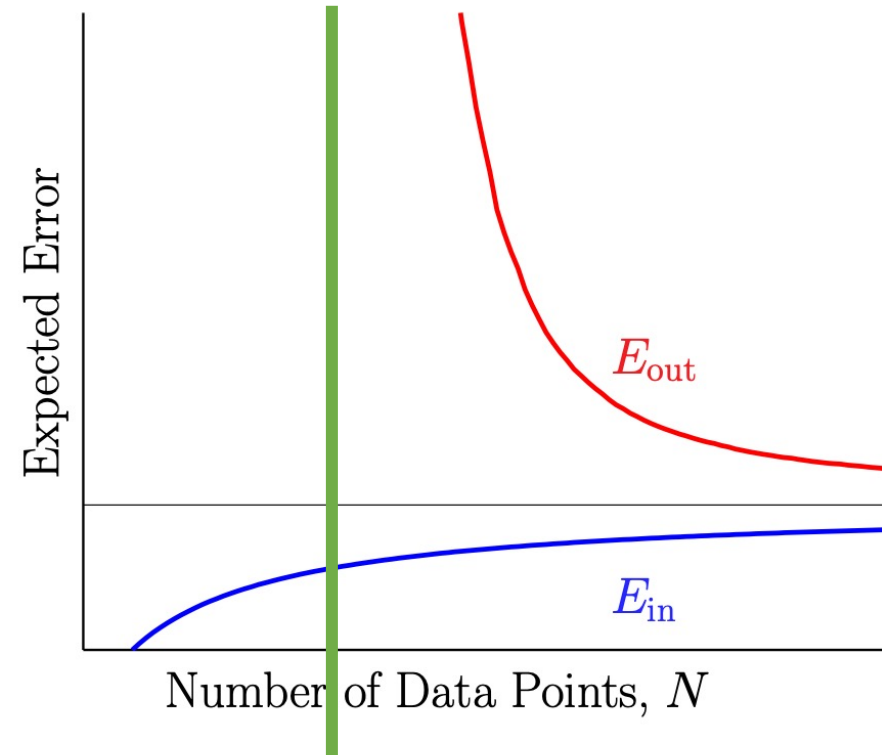
Complex f with no noise.

Why is H_2 Better than H_{10} ?

Learning curve for H_2



Learning curve for H_{10}



When N is small, $E_{out}(g_{10}) \geq E_{out}(g_2)$

A Detailed Experiment

Study the **level of noise** and **target complexity**, and **# data points N**

$$y = f(x) + \epsilon(x) = \sum_{q=0}^{Q_f} \alpha_q x^q + \epsilon(x)$$

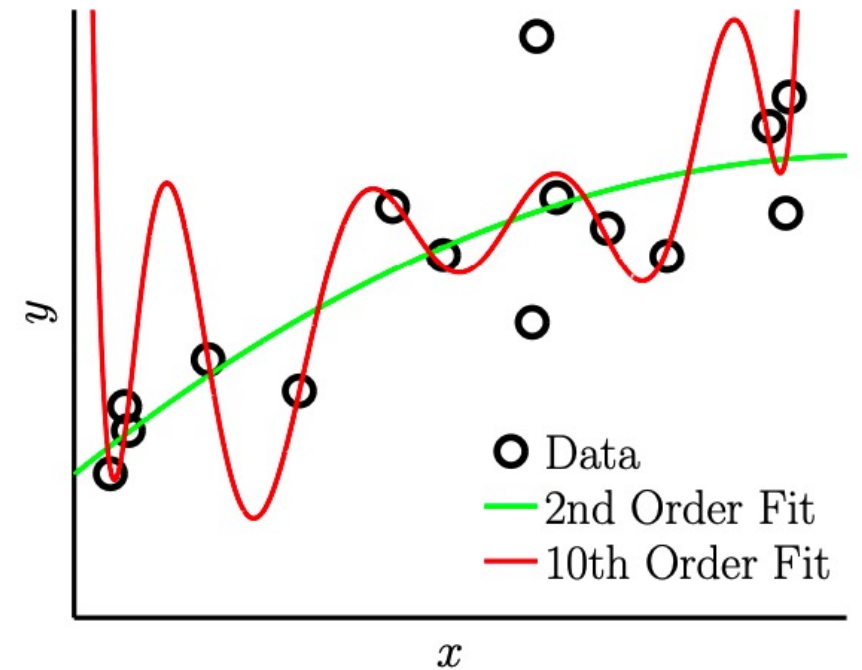
Noise level: variance σ^2 of $\epsilon(x)$

Target complexity: Q_f

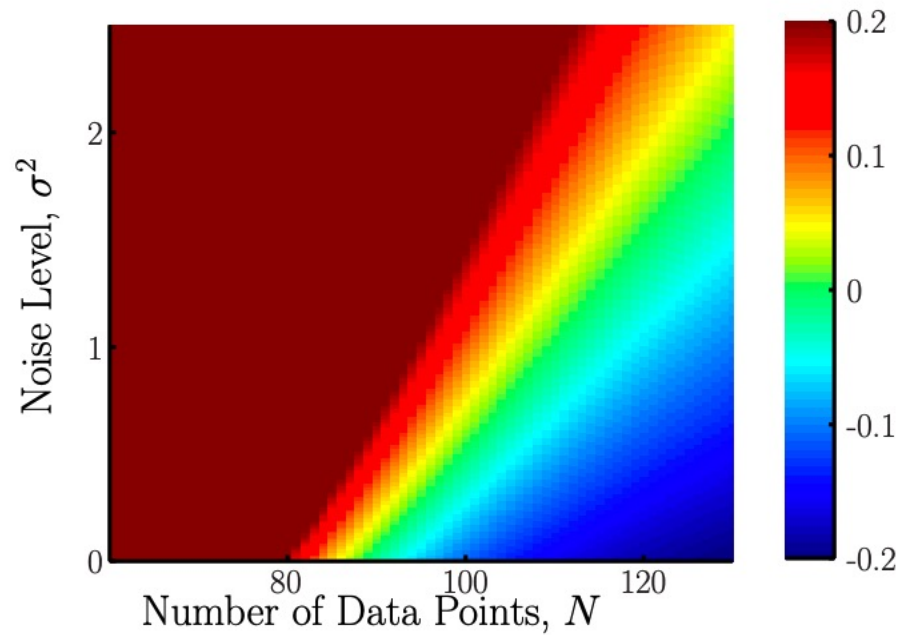
Data set size: N

The Overfit Measure

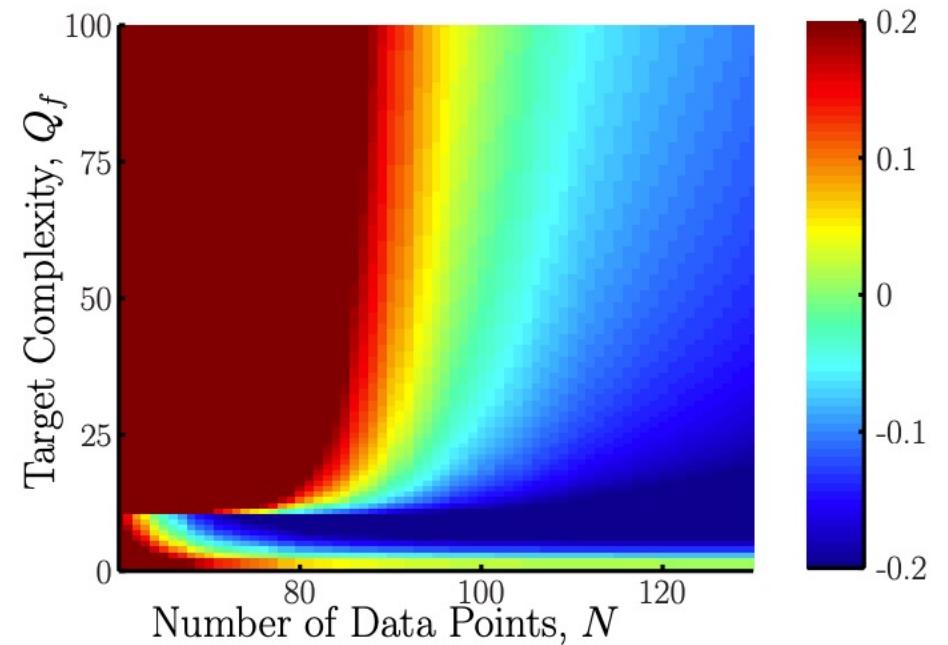
- Fit the data set using H_2 and H_{10}
 - Let g_2 and g_{10} be the learned hypothesis
- Overfit measure
 - $E_{out}(g_{10}) - E_{out}(g_2)$
 - This value is large is overfitting happens



Overfit Measure: $E_{out}(g_{10}) - E_{out}(g_2)$



Stochastic noise



deterministic noise

Number of data points \uparrow	Overfitting \downarrow
Noise \uparrow	Overfitting \uparrow
Target complexity \uparrow	Overfitting \uparrow

Noise:

The part of y we cannot model

Stochastic Noise

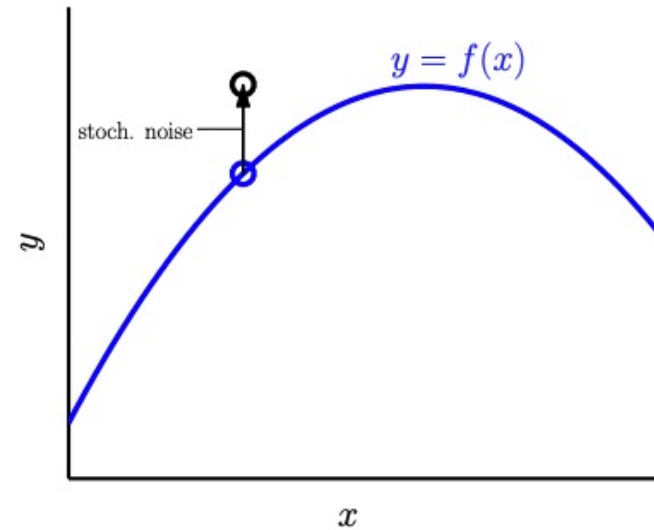
We would like to learn from ○:

$$y_n = f(x_n)$$

Unfortunately, we only observe ●:

$$y_n = f(x_n) + \text{'stochastic noise'}$$

↑
no one can model this



Stochastic Noise: fluctuations/measurement errors we cannot model.

Stochastic Noise

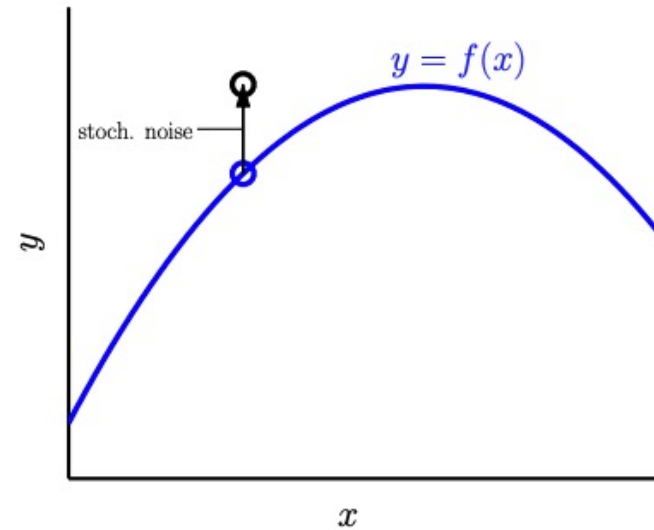
We would like to learn from \bigcirc :

$$y_n = f(x_n)$$

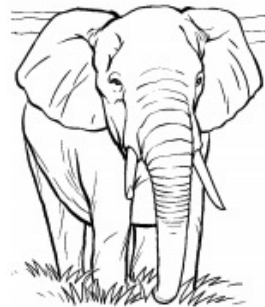
Unfortunately, we only observe \bigcirc :

$$y_n = f(x_n) + \text{'stochastic noise'}$$

↑
no one can model this



Stochastic Noise: fluctuations/measurement errors we cannot model.



Deterministic Noise

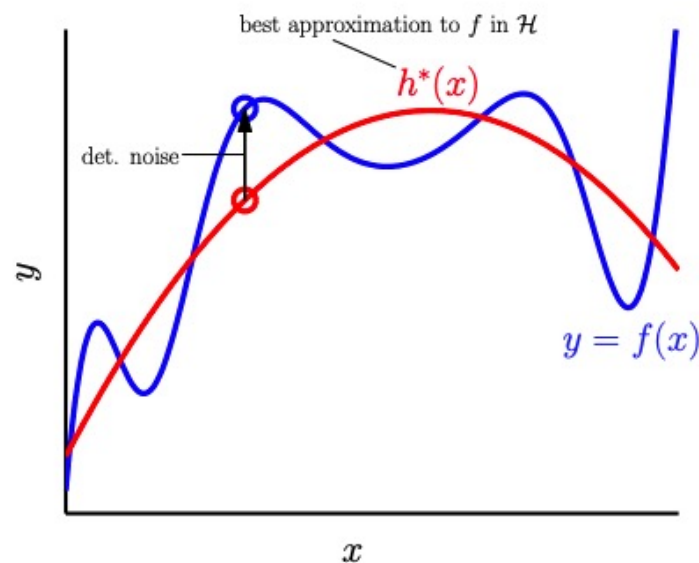
We would like to learn from \circ :

$$y_n = h^*(x_n)$$

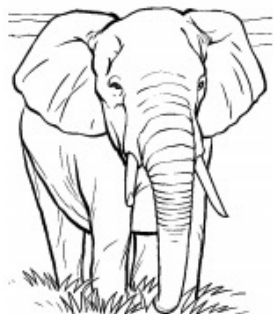
Unfortunately, we only observe \circ :

$$\begin{aligned} y_n &= f(x_n) \\ &= h^*(x_n) + \text{'deterministic noise'} \end{aligned}$$

↑
 \mathcal{H} cannot model this



Deterministic Noise: the part of f we cannot model.



Deterministic Noise

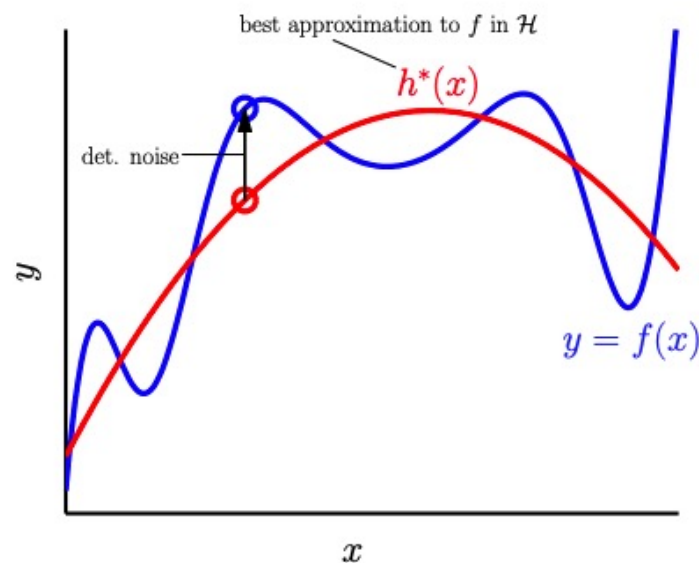
We would like to learn from \circ :

$$y_n = h^*(x_n)$$

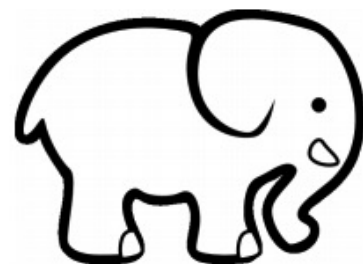
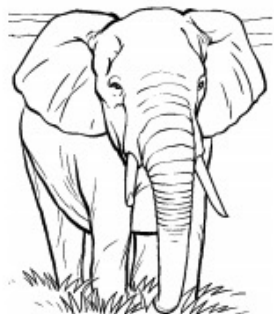
Unfortunately, we only observe \circ :

$$\begin{aligned} y_n &= f(x_n) \\ &= h^*(x_n) + \text{'deterministic noise'} \end{aligned}$$

↑
 \mathcal{H} cannot model this

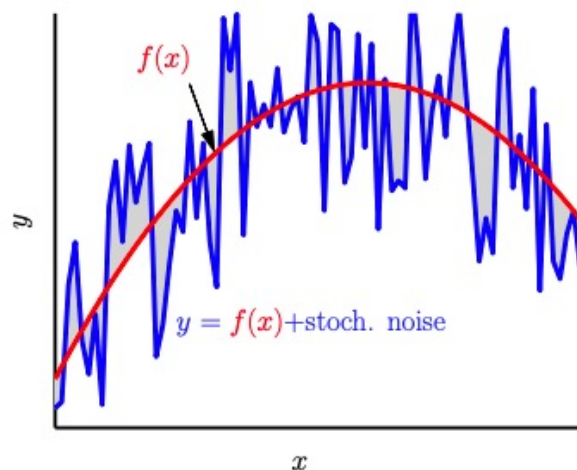


Deterministic Noise: the part of f we cannot model.



Both sources of noises hurt learning

Stochastic Noise



source: random measurement errors

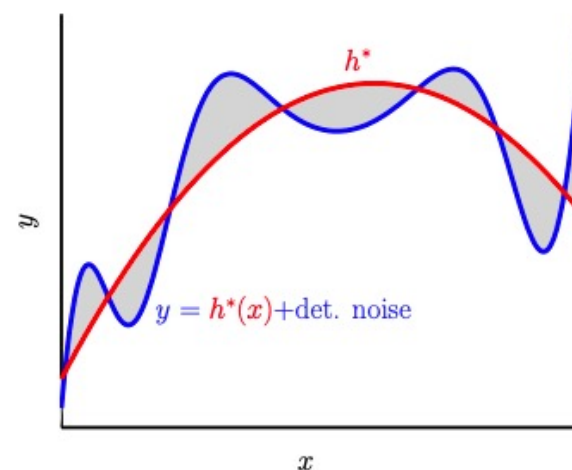
re-measure y_n

stochastic noise changes.

change \mathcal{H}

stochastic noise the same.

Deterministic Noise



source: learner's \mathcal{H} cannot model f

re-measure y_n

deterministic noise the same.

change \mathcal{H}

deterministic noise changes.

We have single \mathcal{D} and fixed \mathcal{H} so we cannot distinguish

Noise and Bias-Variance Decomposition

$$y = f(\vec{x}) + \epsilon$$

$$\mathbb{E}[E_{out}(\vec{x})] = \sigma^2 + \text{bias} + \text{variance}$$



Stochastic Noise



Deterministic noise

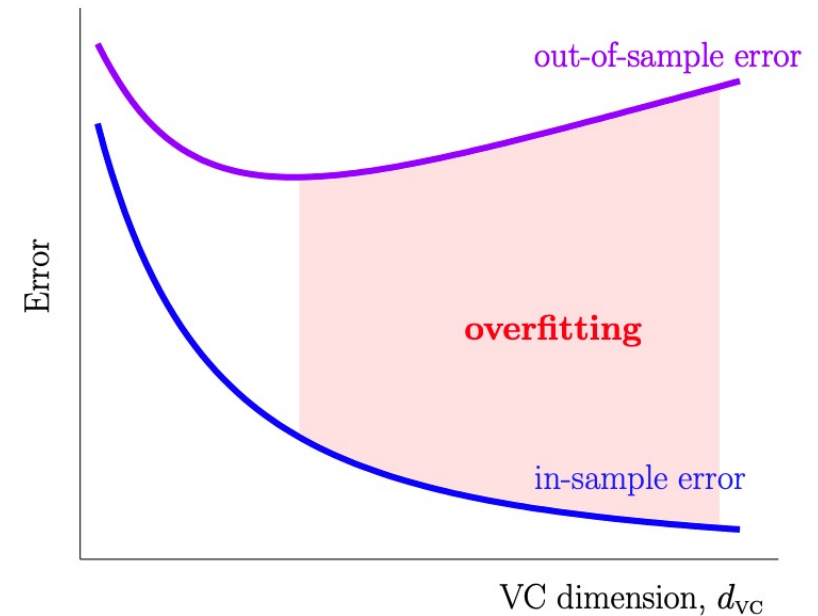
How to Fight Overfitting

- VC Bound

$$E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{vc} \frac{\ln N}{N}}\right)$$

- Fighting overfitting

- Regularization
- Validation
- (The focus of the next two lectures)



VC Dimension of d -dim Perceptron

Recall the Definitions

- Shatter

- H **shatters** $(\vec{x}_1, \dots, \vec{x}_N)$ if $|H(\vec{x}_1, \dots, \vec{x}_N)| = 2^N$
- H can induce all label combinations for $(\vec{x}_1, \dots, \vec{x}_N)$

- Break point

- k is a **break point** for H if no data set of size k can be shattered by H
- k is a break point for $H \leftrightarrow m_H(k) < 2^k$

- VC Dimension: $d_{vc}(H)$ or d_{vc}

- The VC dimension of H is the largest N such that $m_H(N) = 2^N$
- Equivalently, if k^* is the smallest break point for H , $d_{vc}(H) = k^* - 1$

VC Dimension of d-dimension Perceptron

- Claim:
 - The VC Dimension of d-dim perceptron is $d + 1$
- How to prove it?
 1. Show that the VC dimension of d-dim perceptron $\geq d + 1$
 2. Show that the VC dimension of d-dim perceptron $\leq d + 1$

- To prove $d_{vc}(H) \geq d + 1$, what do we need to prove?
 - A. There is a set of $d + 1$ points that can be shattered by H
 - B. There is a set of $d + 1$ points that cannot be shattered by H
 - C. Every set of $d + 1$ points can be shattered by H
 - D. Every set of $d + 1$ points cannot be shattered by H

- To prove $d_{vc}(H) \geq d + 1$, what do we need to prove?
 - A. There is a set of $d + 1$ points that can be shattered by H
 - B. There is a set of $d + 1$ points that cannot be shattered by H
 - C. Every set of $d + 1$ points can be shattered by H
 - D. Every set of $d + 1$ points cannot be shattered by H

- To prove $d_{vc}(H) \geq d + 1$, what do we need to prove?
 - A. There is a set of $d + 1$ points that can be shattered by H
 - B. There is a set of $d + 1$ points that cannot be shattered by H
 - C. Every set of $d + 1$ points can be shattered by H
 - D. Every set of $d + 1$ points cannot be shattered by H
- To prove $d_{vc}(H) \leq d + 1$, what do we need to prove?
 - A. There is a set of $d + 1$ points that can be shattered by H
 - B. There is a set of $d + 2$ points that cannot be shattered by H
 - C. Every set of $d + 2$ points can be shattered by H
 - D. Every set of $d + 1$ points cannot be shattered by H
 - E. Every set of $d + 2$ points cannot be shattered by H

- To prove $d_{vc}(H) \geq d + 1$, what do we need to prove?
 - A. There is a set of $d + 1$ points that can be shattered by H
 - B. There is a set of $d + 1$ points that cannot be shattered by H
 - C. Every set of $d + 1$ points can be shattered by H
 - D. Every set of $d + 1$ points cannot be shattered by H
- To prove $d_{vc}(H) \leq d + 1$, what do we need to prove?
 - A. There is a set of $d + 1$ points that can be shattered by H
 - B. There is a set of $d + 2$ points that cannot be shattered by H
 - C. Every set of $d + 2$ points can be shattered by H
 - D. Every set of $d + 1$ points cannot be shattered by H
 - E. Every set of $d + 2$ points cannot be shattered by H

- To prove $d_{vc}(H) \geq d + 1$, what do we need to prove?
There is a set of $d + 1$ points that can be shattered by H
- To prove $d_{vc}(H) \leq d + 1$, what do we need to prove?
Every set of $d + 2$ points cannot be shattered by H