Lecture 10

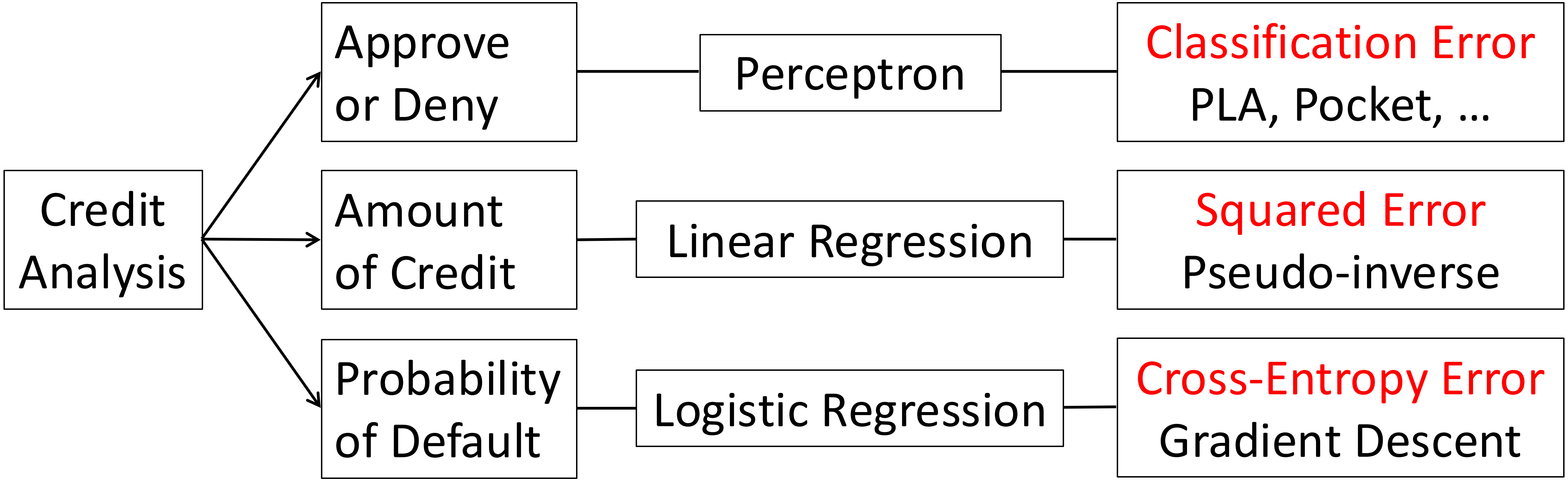
CS436/536: Introduction to Machine Learning

Zhaohan Xi
Binghamton University

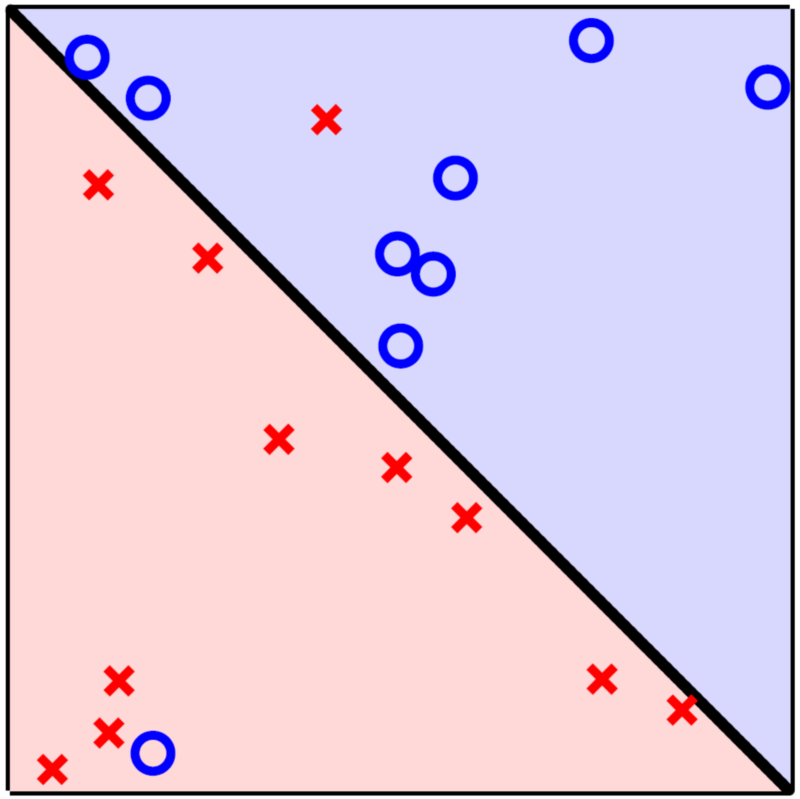
zxi1@binghamton.edu

Recap: Linear Model for Three Learning Problems

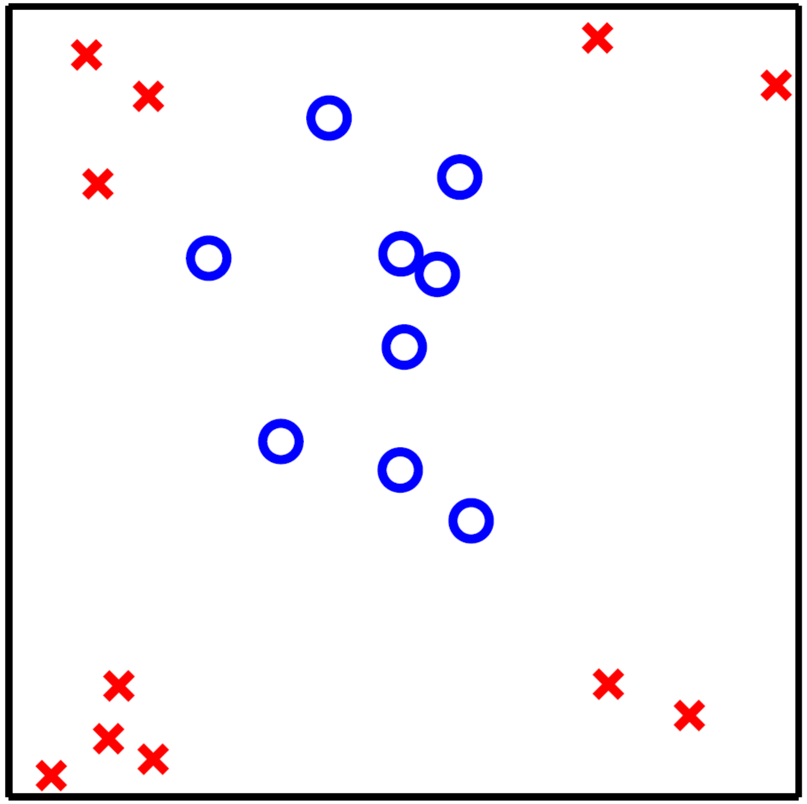
$$h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}, \qquad \text{where } \boldsymbol{w} \in \mathbb{R}^d, \boldsymbol{x} \in 1 \times \mathbb{R}^d$$



But: The Linear Model has Limitations



Linear with Outliers



Essentially Non-linear

Need something non-linear

Please remember these two scenarios for later

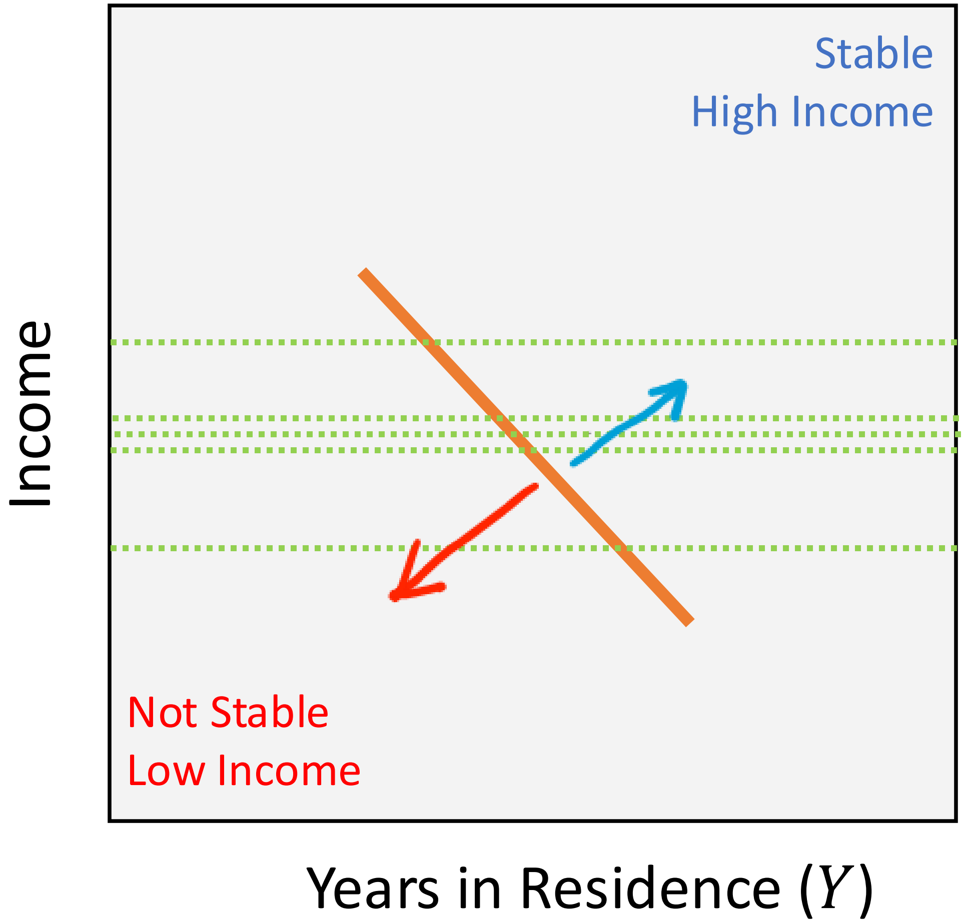
Credit Approval

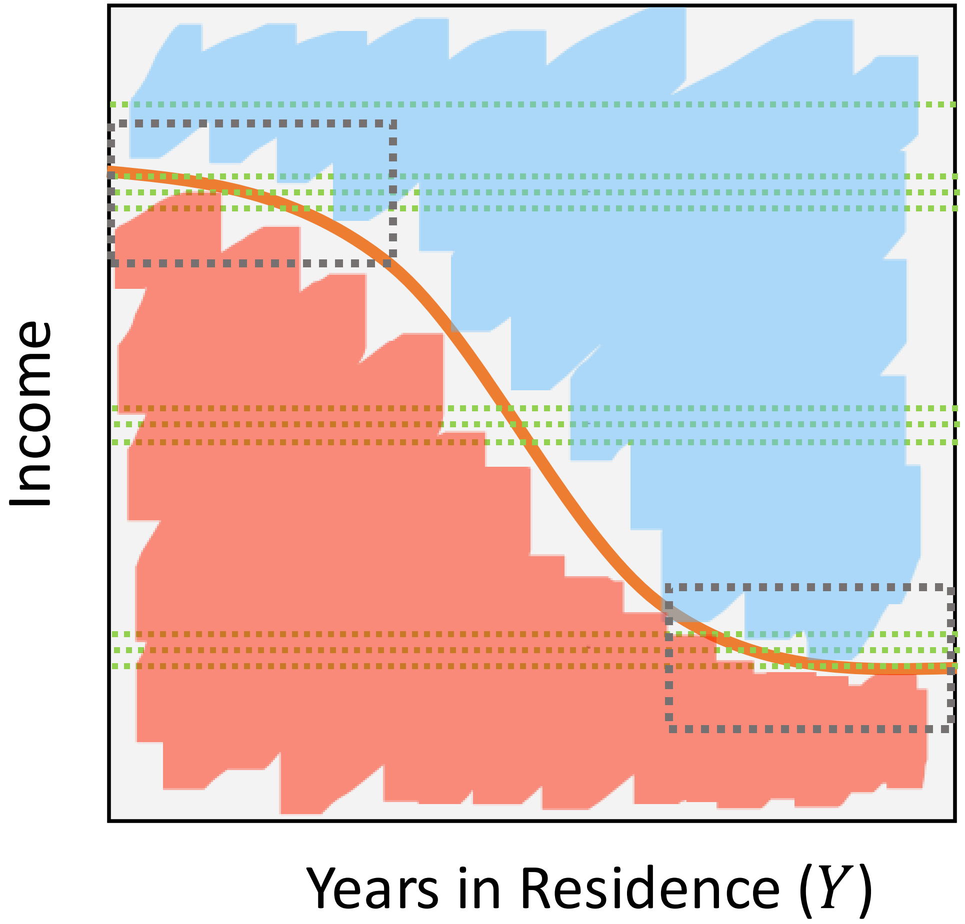
- Using salary, debt, years in residence, etc., approve for credit or not

age	33 years
salary	50,000
debt	27,500
years employed	1
years at residence	2
…	…

Approve for credit?

Use the Appropriate Feature



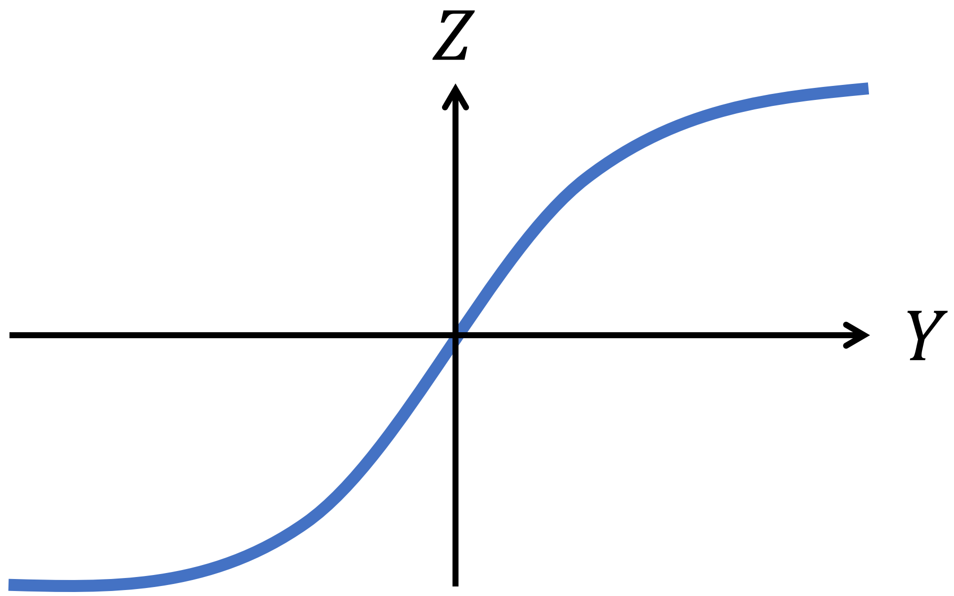


$Y \gg 3$ years, no additional effect beyond $Y = 3$

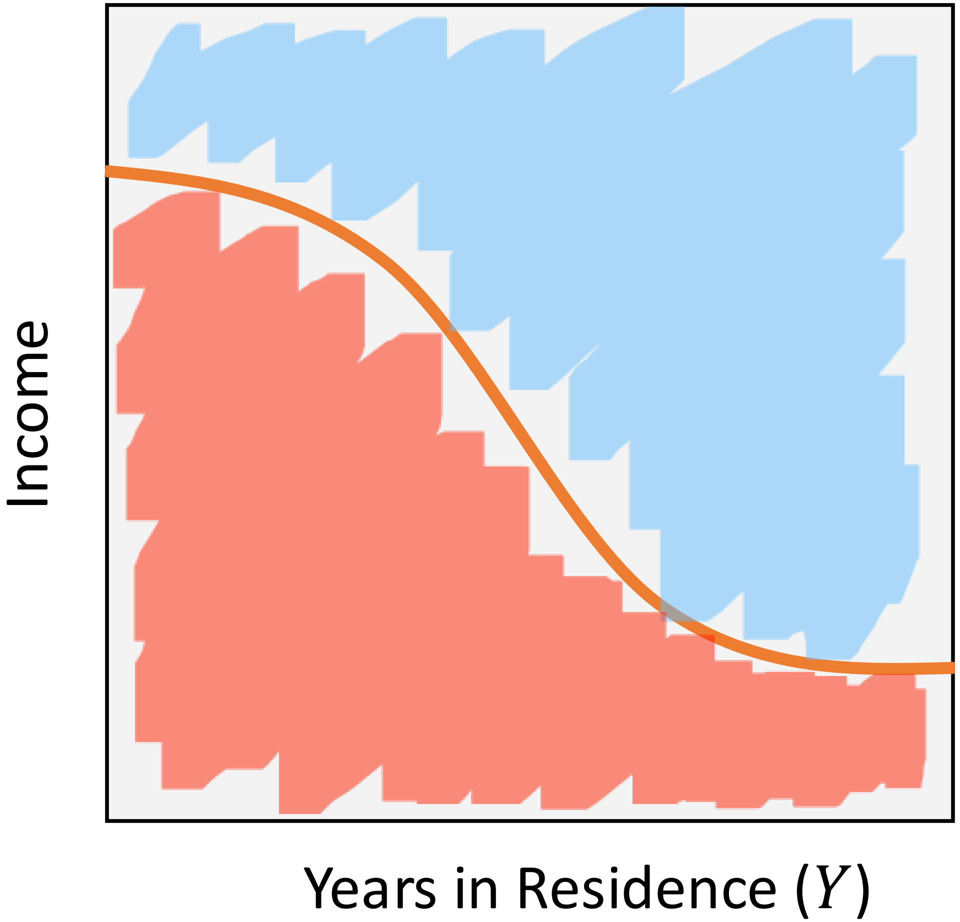
$Y \ll 0.3$ years, no additional effect below $Y = 0.3$

Change the Feature Using a Transform

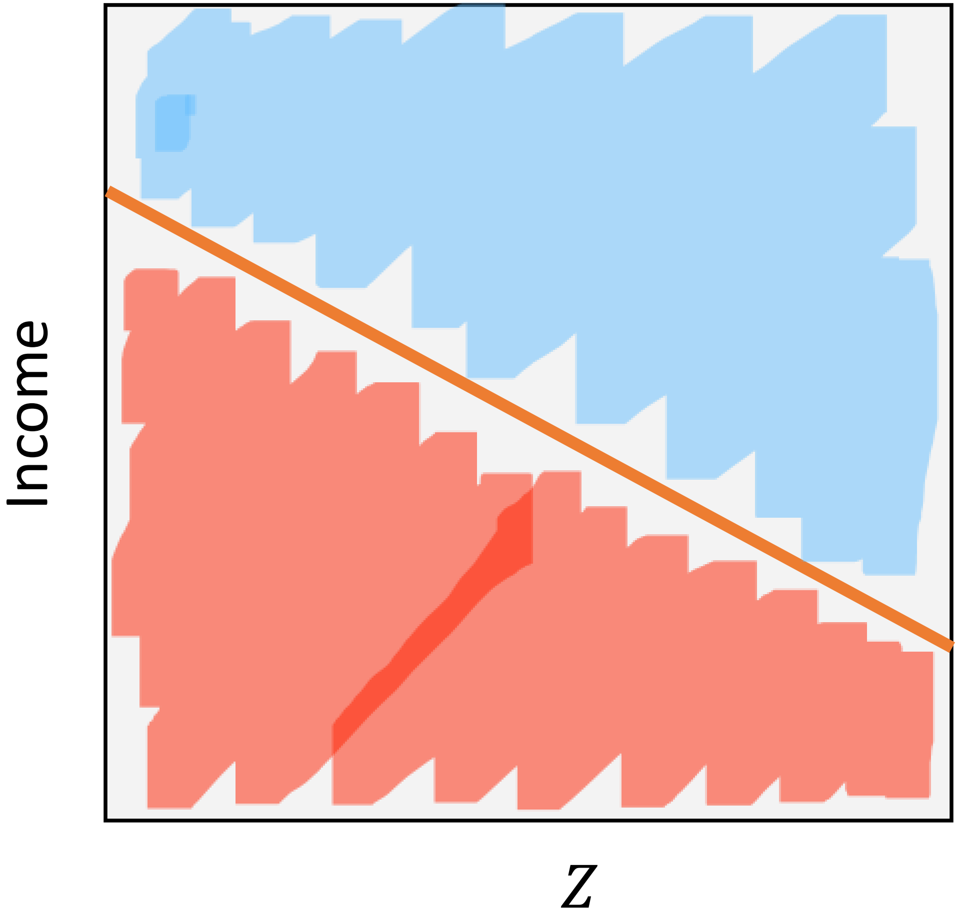
$$Y \xrightarrow{\Phi} Z$$



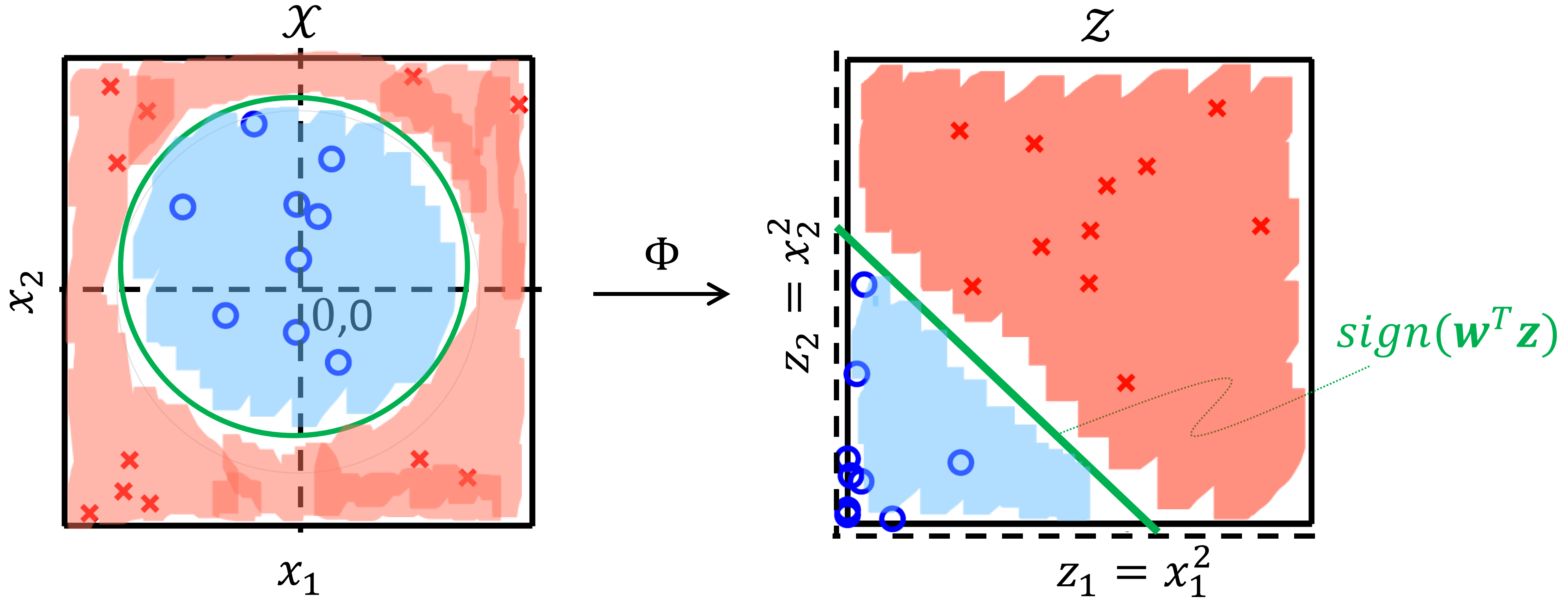
Changing the Feature Using a Transform



$$Y \xrightarrow{\Phi} Z$$



Mechanics of Non-Linear Feature Transforms



$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \xrightarrow{\Phi} z = \begin{bmatrix} 1 \\ z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \Phi(x_1) \\ \Phi(x_2) \end{bmatrix} = \begin{bmatrix} 1 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$

Feature Transforms In General

$\mathcal{X}$-space is $\mathbb{R}^d$

$\mathcal{Z}$-space is $\mathbb{R}^{\tilde{d}}$

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ \cdots \\ x_d \end{bmatrix}$$

$$\boldsymbol{z} = \boldsymbol{\Phi}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ \Phi_1(\boldsymbol{x}) \\ \cdots \\ \Phi_{\tilde{d}}(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ z_1 \\ \cdots \\ z_{\tilde{d}} \end{bmatrix}$$

$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)$

$(\boldsymbol{z}_1, y_1), \ldots, (\boldsymbol{z}_N, y_N)$

$$\tilde{\boldsymbol{w}} = \begin{bmatrix} w_0 \\ w_1 \\ \cdots \\ w_{\tilde{d}} \end{bmatrix}$$

$g(\boldsymbol{x}) = sign(\tilde{\boldsymbol{w}}^T \boldsymbol{\Phi}(\boldsymbol{x}))$

Low E_{in}

What about "Real Learning"? Generalization

- Want $g \approx f$ over all of $\mathcal{X}$,

i.e., $E_{out}(g) \approx 0$

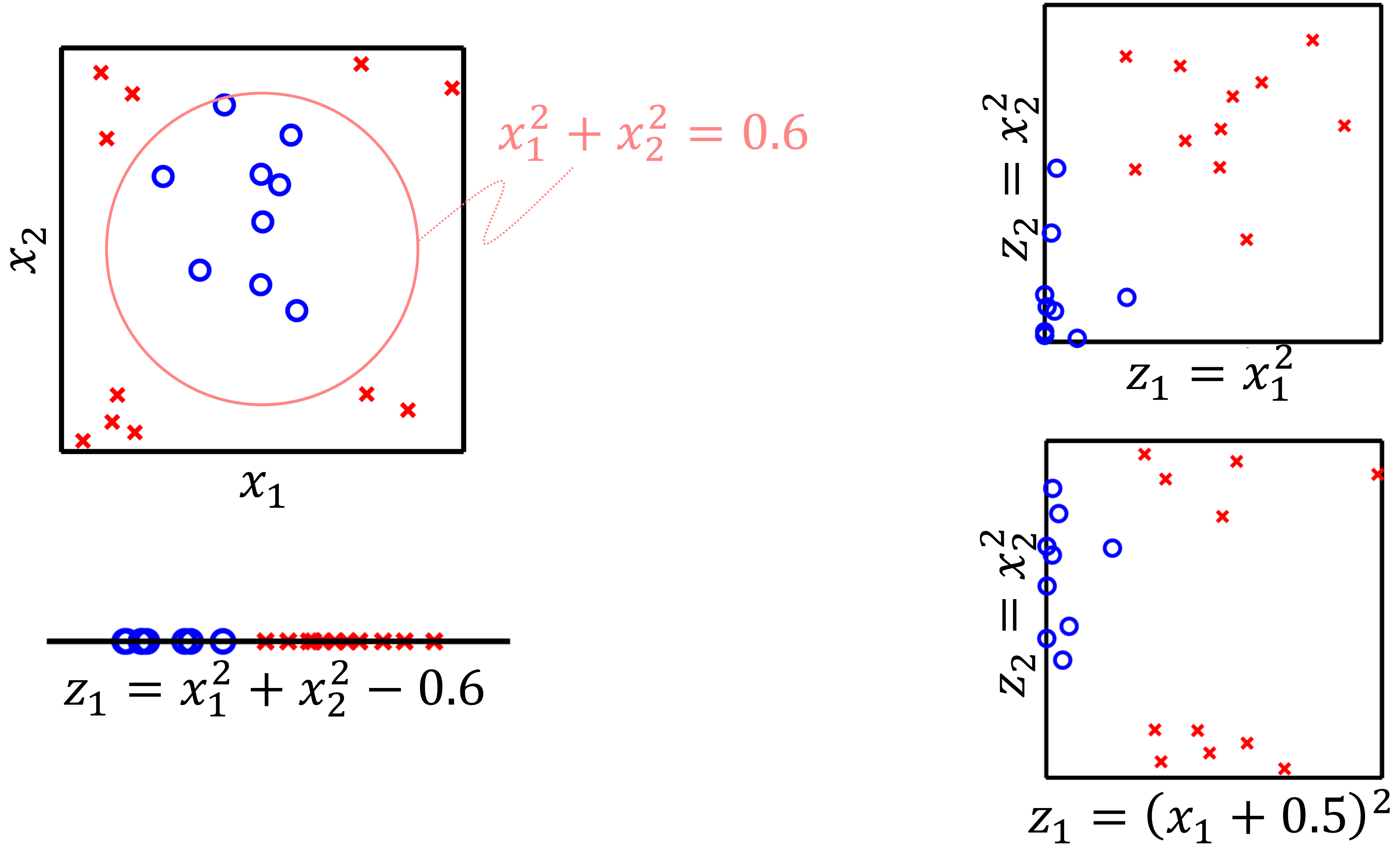
[Approximation] small $E_{in}(g)$

[Generalization] $E_{in}(g) \approx E_{out}(g)$

$$\begin{array}{ccc} d_{VC} & & \tilde{d}_{VC} \\ & \longrightarrow & \\ d+1 & & \tilde{d}+1 \end{array}$$

Select transform Φ with small $\tilde{d}$

Possibly Many ‘Good’ Nonlinear Transforms



Polynomial Feature Transform

E.g. Polynomial Transform for 2-D $\mathcal{X}$-space

Degree 1: $(1, x_1, x_2) \xrightarrow{\mathbf{\Phi}_1} (1, x_1, x_2) \qquad \Rightarrow \tilde{d}_{VC} = 3$

Degree 2: $\xrightarrow{\mathbf{\Phi}_2} (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2) \qquad \Rightarrow \tilde{d}_{VC} = 6$

Degree 3: $\xrightarrow{\mathbf{\Phi}_3} (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2, x_1^3, x_1^2 x_2, x_1 x_2^2, x_2^3) \quad \Rightarrow \tilde{d}_{VC} = 10$

Degree 4: $\xrightarrow{\mathbf{\Phi}_4} (1, x_1, x_2, x_1^2, \ldots, x_2^2, x_1^3, \ldots, x_2^3, x_1^4, \ldots, x_2^4) \quad \Rightarrow \tilde{d}_{VC} = 15$

In general:

For a Q-th order polynomial feature transform, with original x of d dimensions,

$$\tilde{d} = \binom{Q+d}{Q} - 1 \Rightarrow \tilde{d}_{VC} = \binom{Q+d}{Q}$$

Pick Feature Transform **BEFORE** Seeing Data

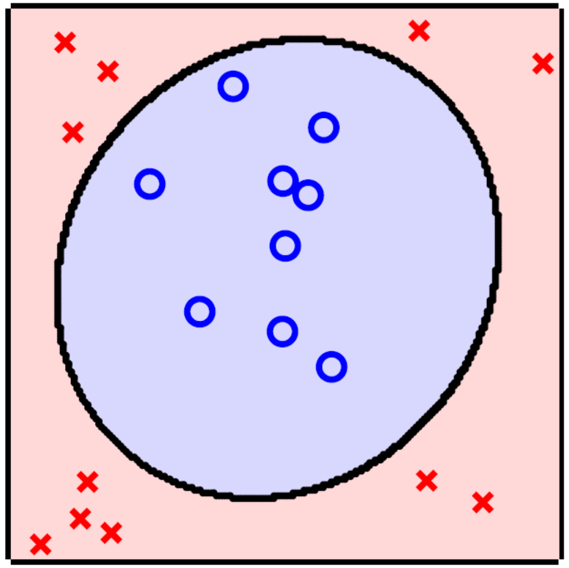
Construct Features Carefully, Before Seeing the Data

- Do NOT Pick $\mathbf{\Phi}$ after looking at the data
- Pick $\mathbf{\Phi}$ BEFORE seeing the data

If you suspect linear model is insufficient to approximate the target,

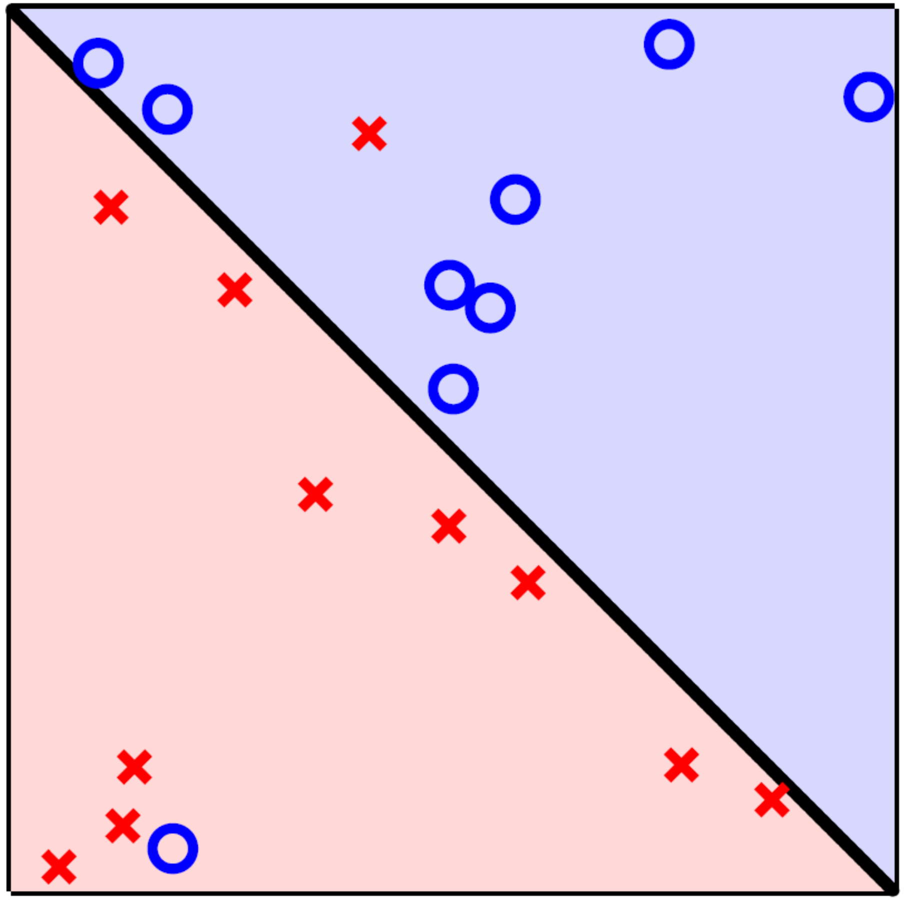
Pick one of the standard feature transforms

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \quad \xrightarrow{\mathbf{\Phi}_2} \quad \mathbf{\Phi}_2(\boldsymbol{x}) = \begin{bmatrix} 1 \\ \Phi_1(\boldsymbol{x}) \\ \Phi_2(\boldsymbol{x}) \\ \Phi_3(\boldsymbol{x}) \\ \Phi_4(\boldsymbol{x}) \\ \Phi_5(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_1 x_2 \\ x_2^2 \end{bmatrix}$$

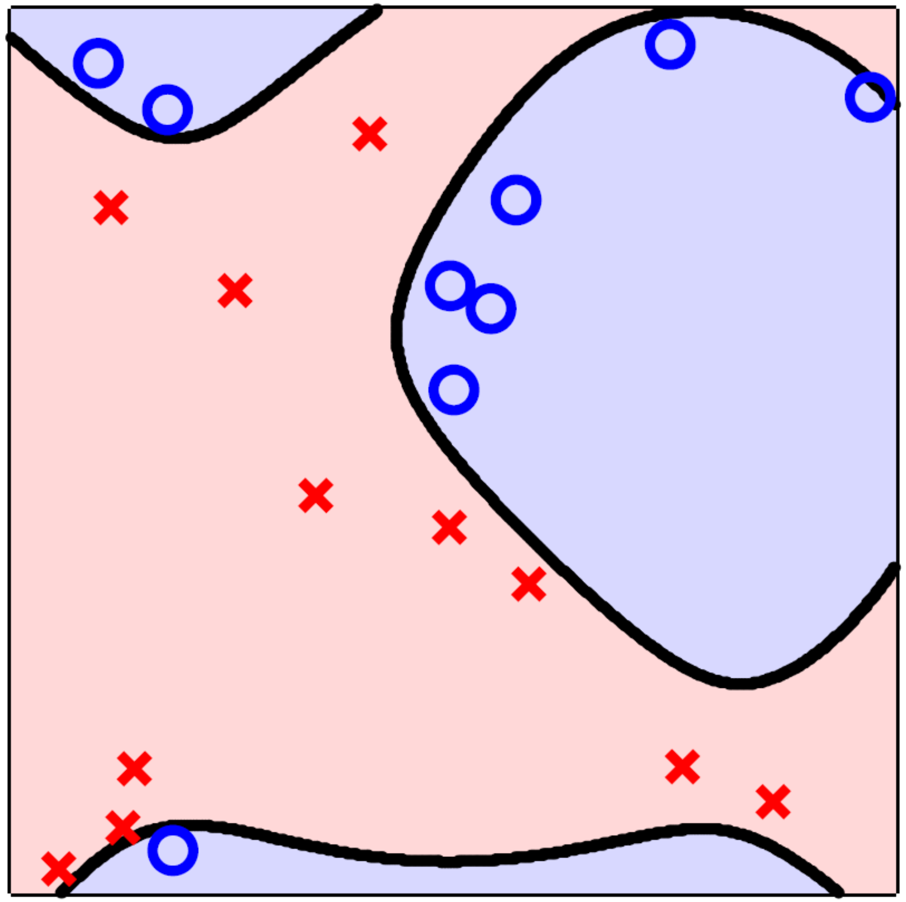


Use Feature Transforms Responsibly

- Approximation – Generalization Tradeoff

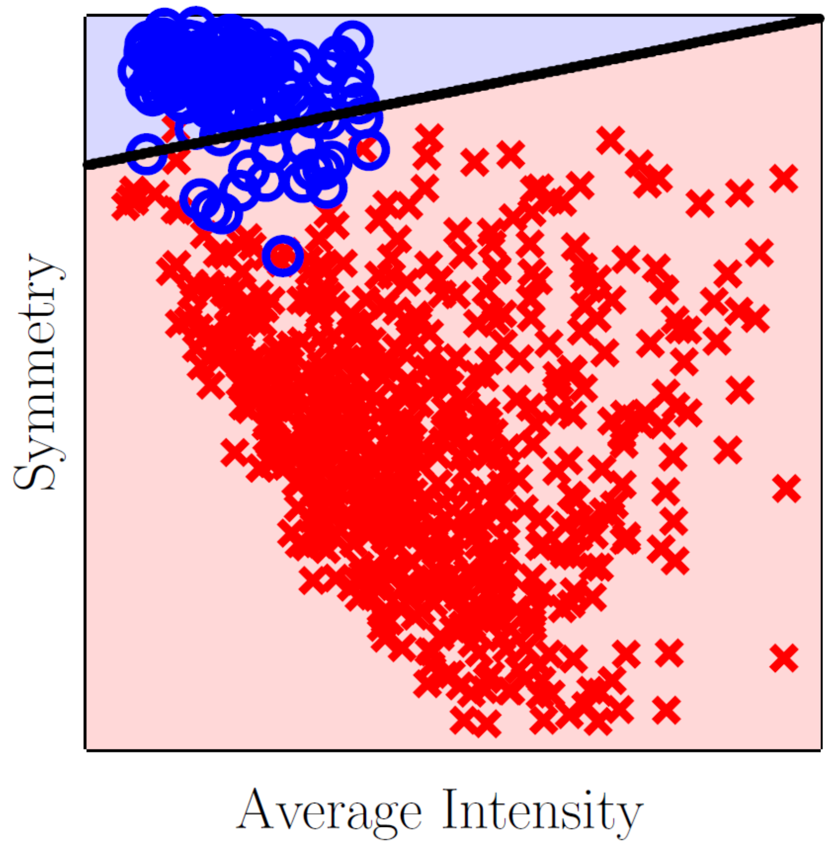


$Q = 1$



$Q = 4$

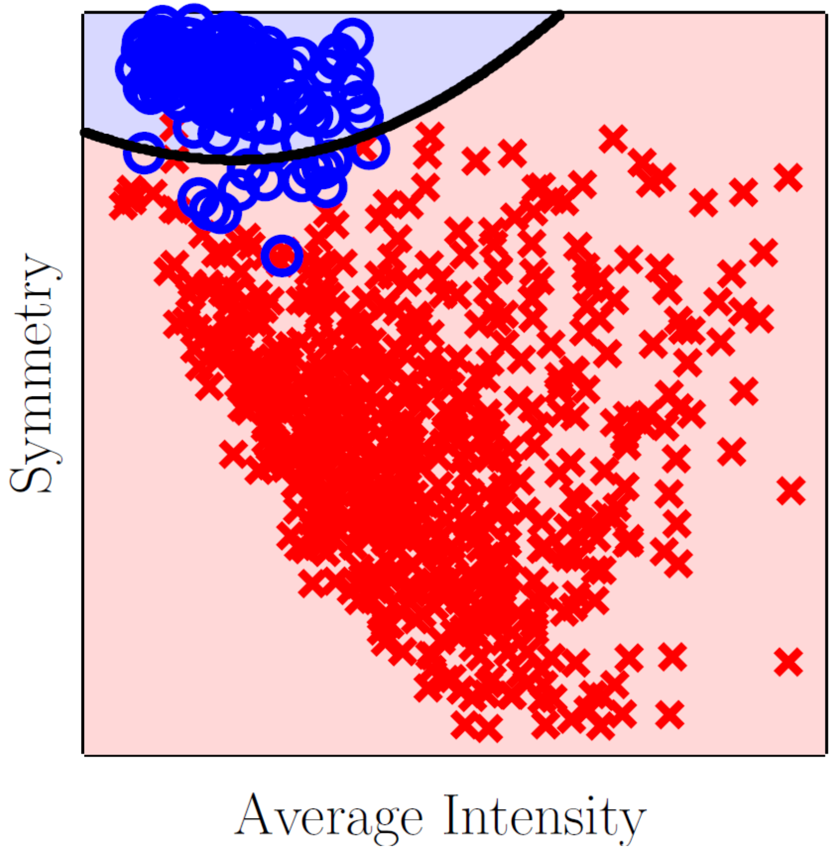
Digits Data – “1” vs. “All”



Linear model

$E_{in} \approx 2.13\%$

$E_{out} \approx 2.38\%$



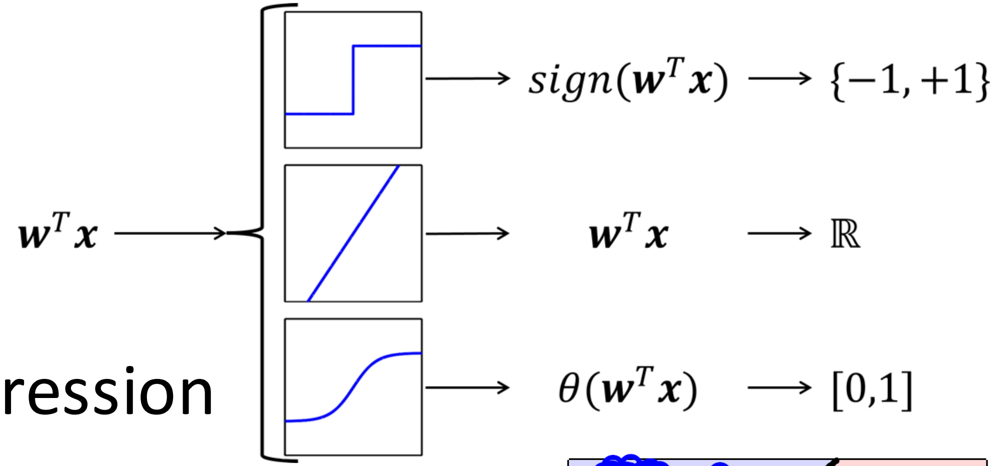
3rd Order Polynomial model

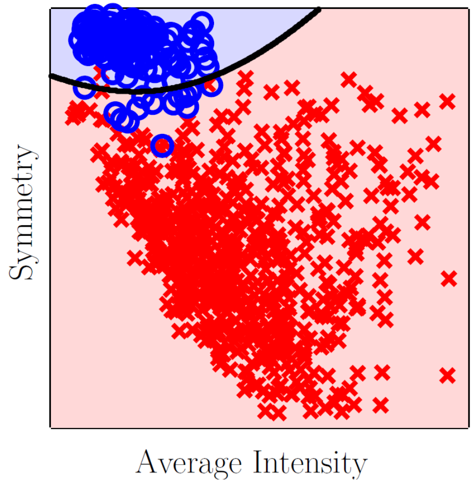
$E_{in} \approx 1.75\%$

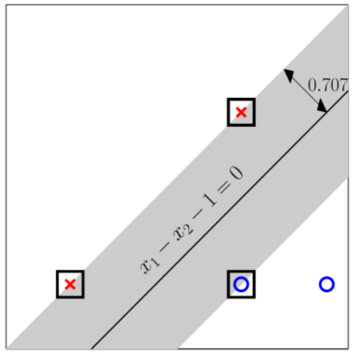
$E_{out} \approx 1.87\%$

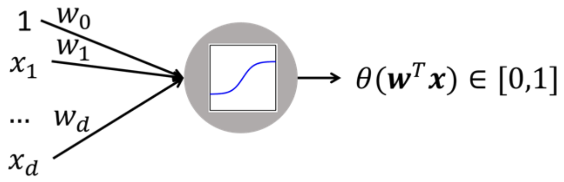
Use the Linear Model!

- Try it first – simple, robust, works
- Works for different kinds of problems:
  - Classification, Regression, Logistic Regression
- Works for nonlinear targets:
  - All the machinery and algorithms
    - Can tolerate errors
    - Easily extended to use nonlinear transforms
- Pick feature transform before seeing the data
- Linear Model is Fundamental!
  - Building block for more complex models

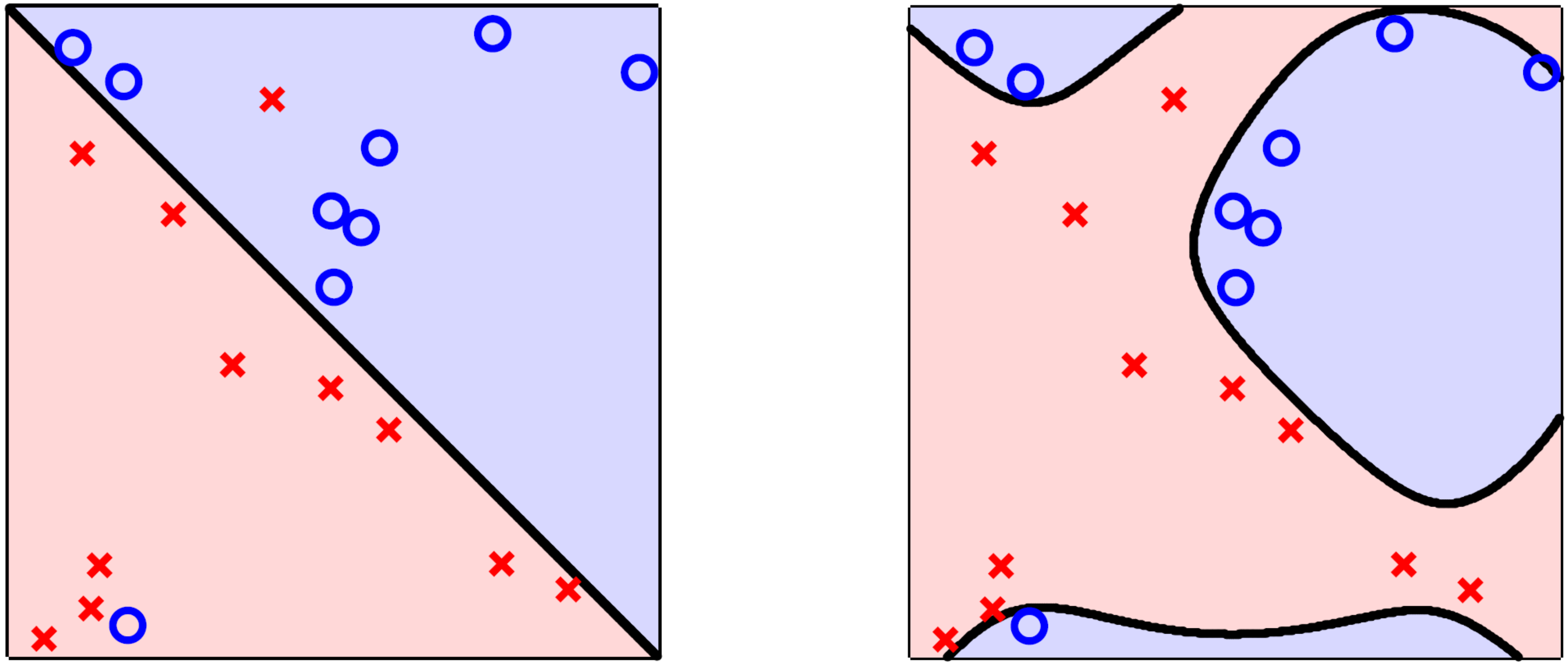








Overfitting

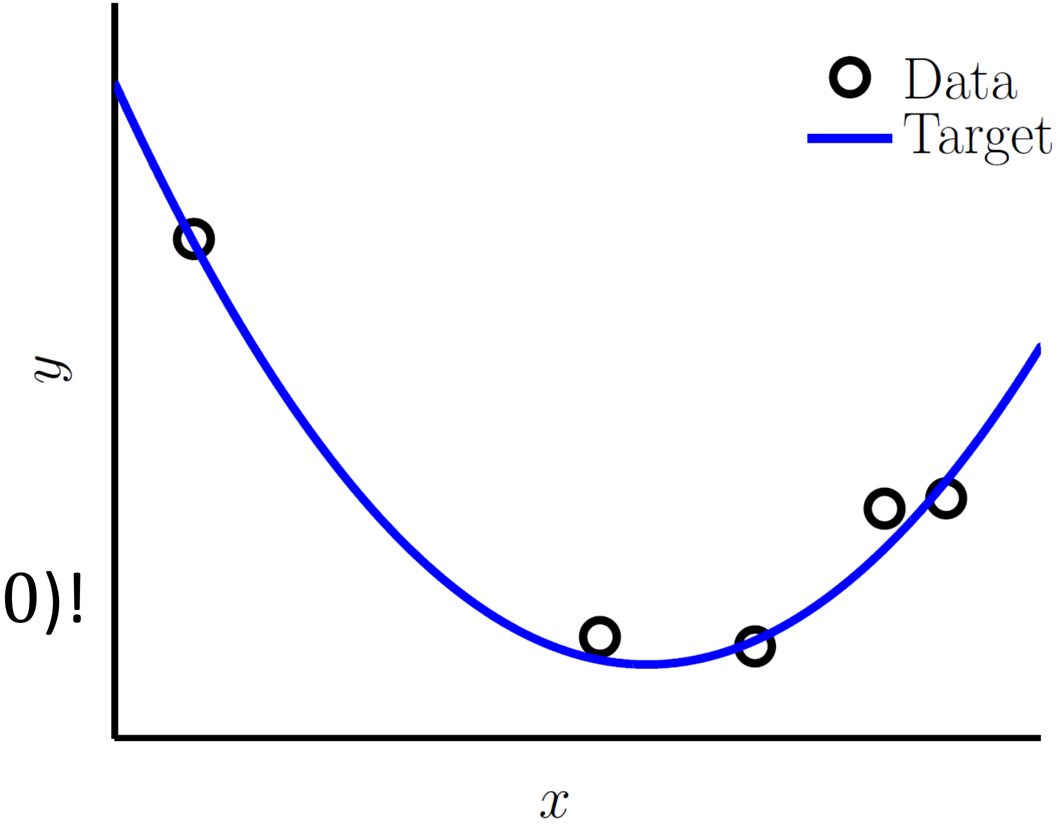


Another Simple Learning Problem

- Target f is *quadratic*
- 5 data points $x_1, x_2, \dots, x_5$
- A little noise (small measurement error)

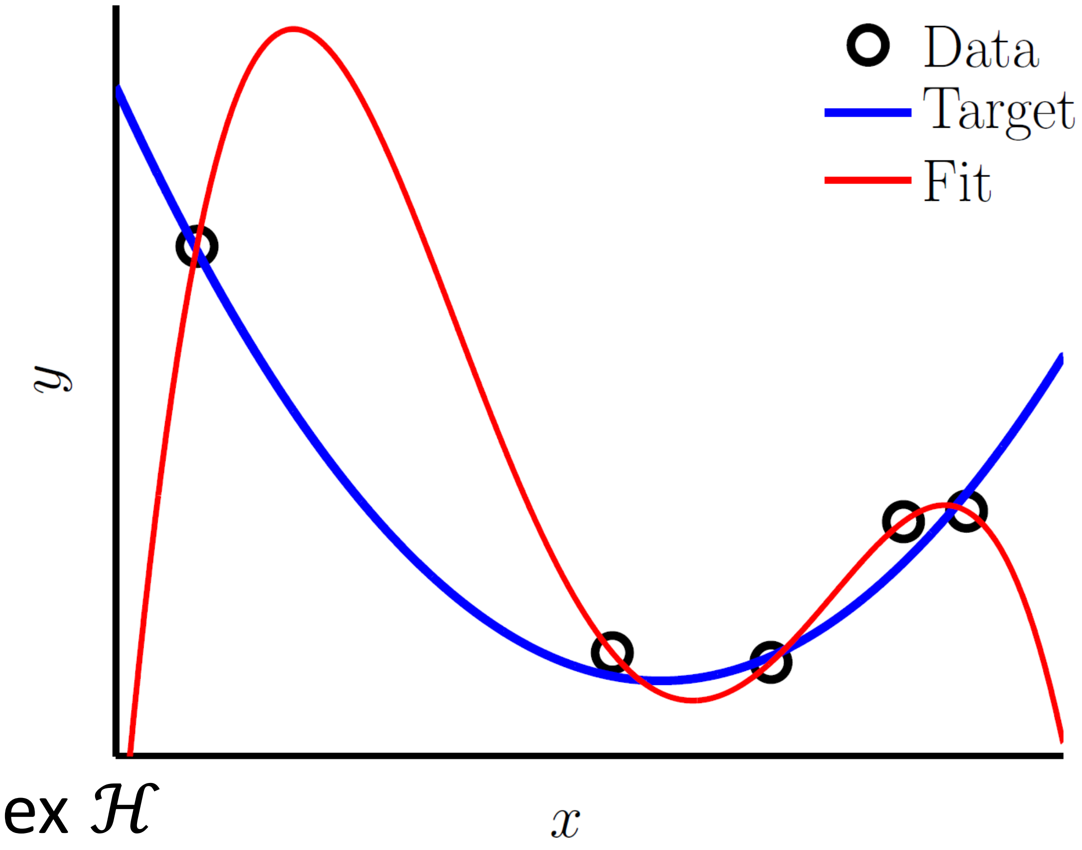
- 5 data points?

4th order polynomial can fit it exactly ($E_{in} = 0$)!



An Illustration of Overfitting

- Target f is *quadratic*
- 5 data points $x_1, x_2, \ldots, x_5$
- A little noise (small measurement error)



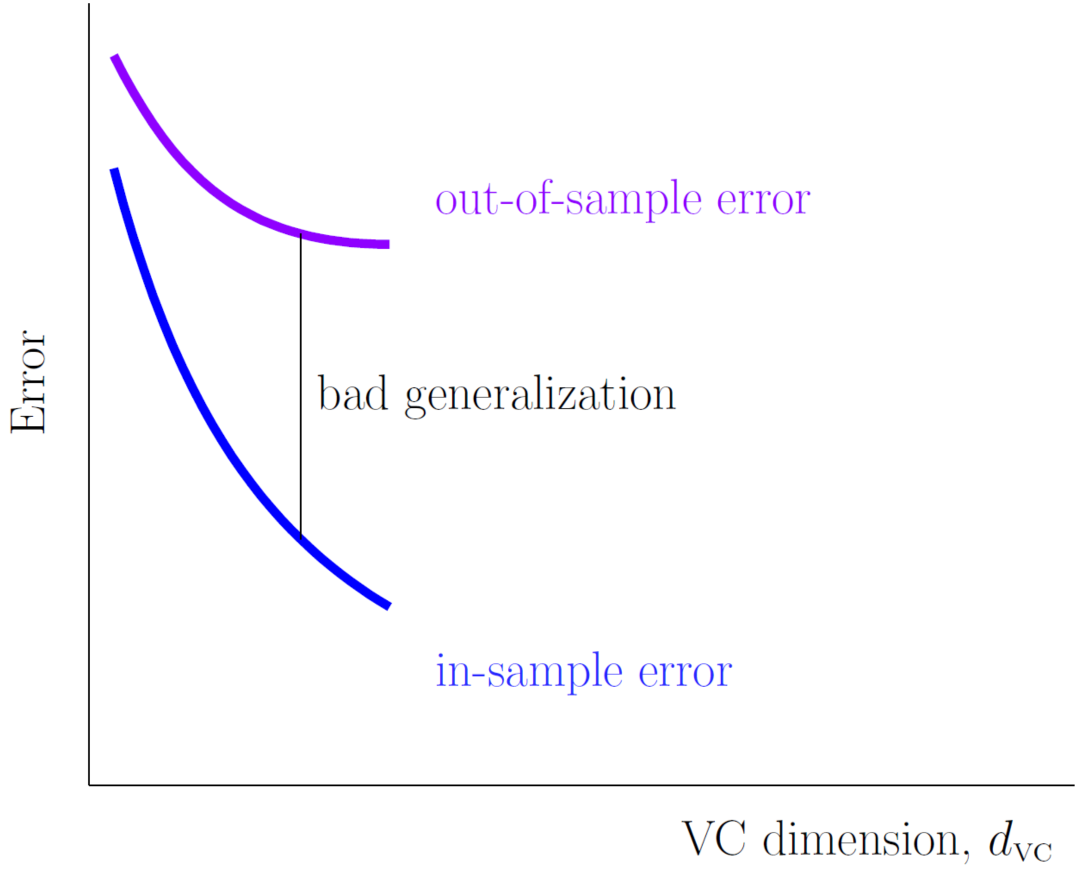
- Classic Overfitting

Learning a simple target with excessively complex $\mathcal{H}$

Small $E_{in} \approx 0$, but High $E_{out} \gg 0$

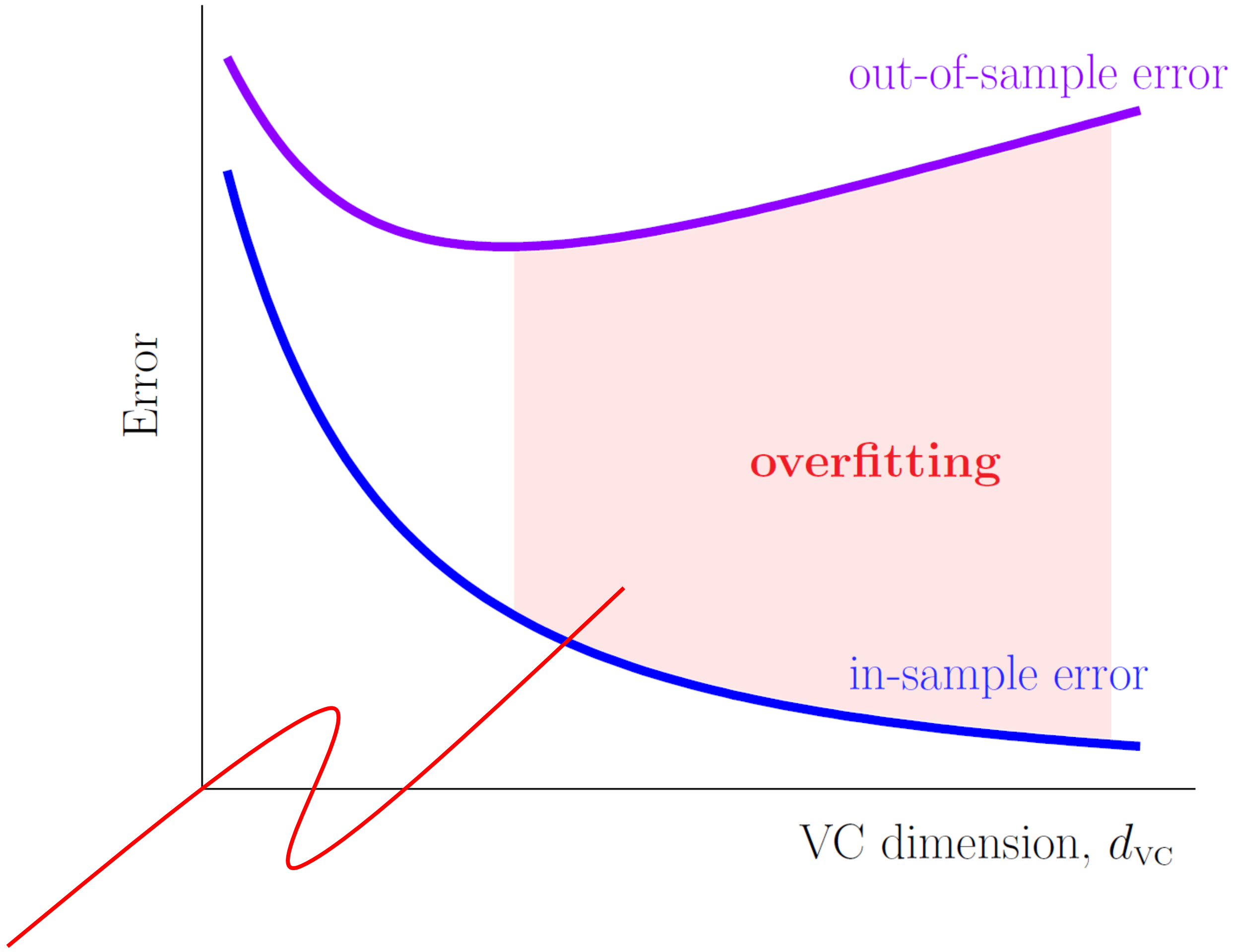
Because we listened to the **noise**. (We fit the data more than was warranted)

Overfitting is Not Just Bad Generalization



- VC analysis covers bad generalization
- Fitting the data, lowering E_{in}, is fine so long as E_{out} is also lowered

Overfitting: When Lower E_{in} results in *Higher* E_{out}



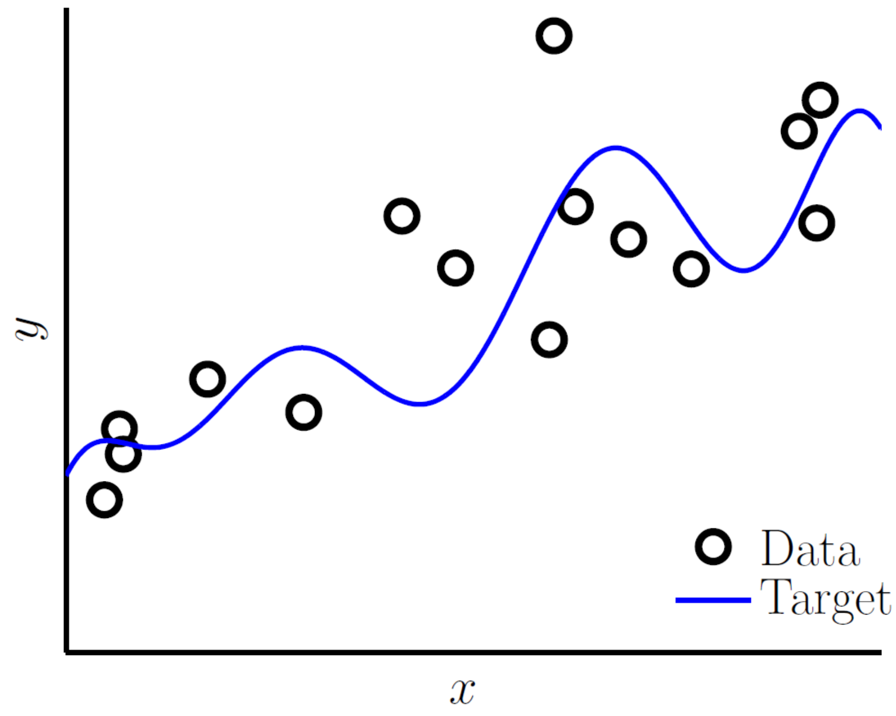
Lowering E_{in} is no longer a good guide to lower E_{out}

The Curious Case of the Simple Model

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

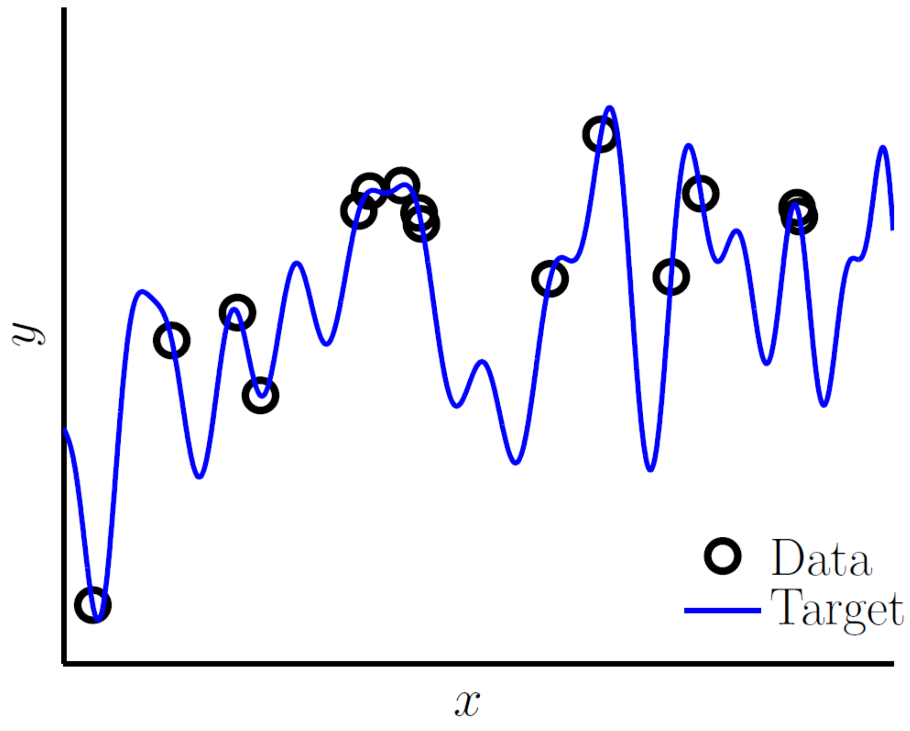
Two Learning Problems

Simple Noisy Target



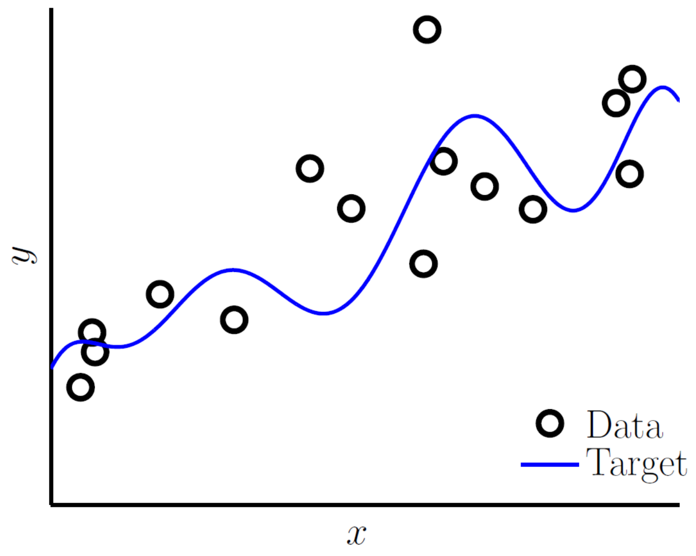
10th order f with noise.

Complex Noiseless Target

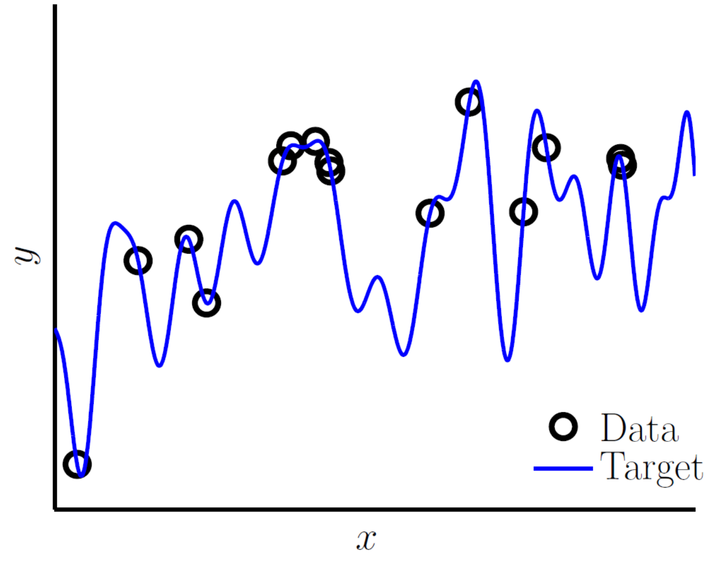


50th order f with no noise.

Two Contenders: 2nd Order vs 10th Order Polynomial



10th order f with noise.



50th order f with no noise.

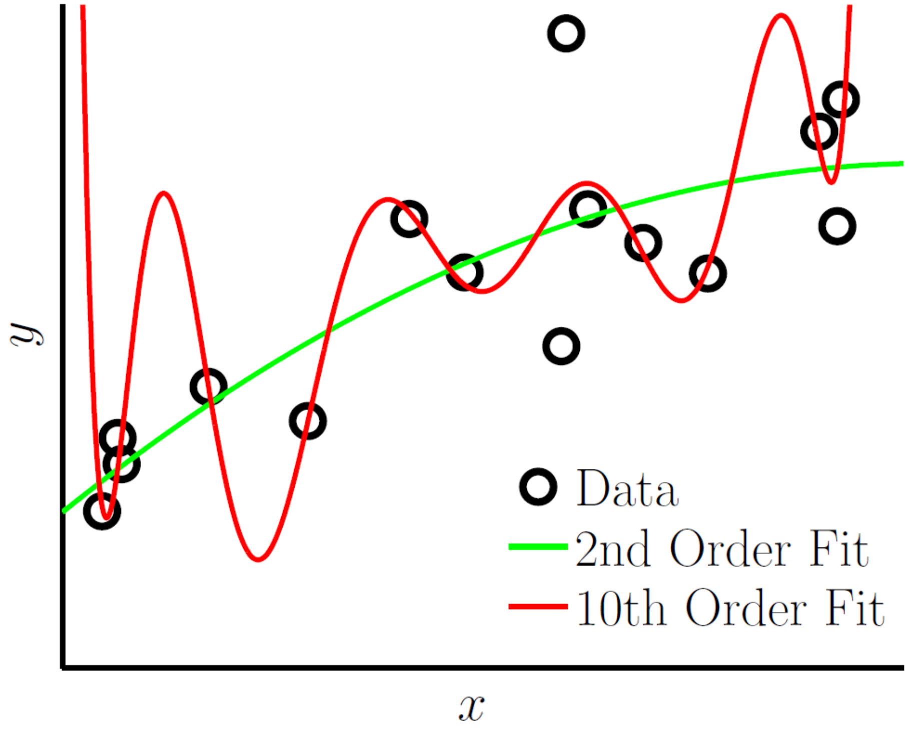
$\mathcal{H}_2 : h(\boldsymbol{x}) = w^T \Phi_2(\boldsymbol{x})$

Learn linear model with 2nd order polynomial transform

$\mathcal{H}_{10} : h(\boldsymbol{x}) = w^T \Phi_{10}(\boldsymbol{x})$

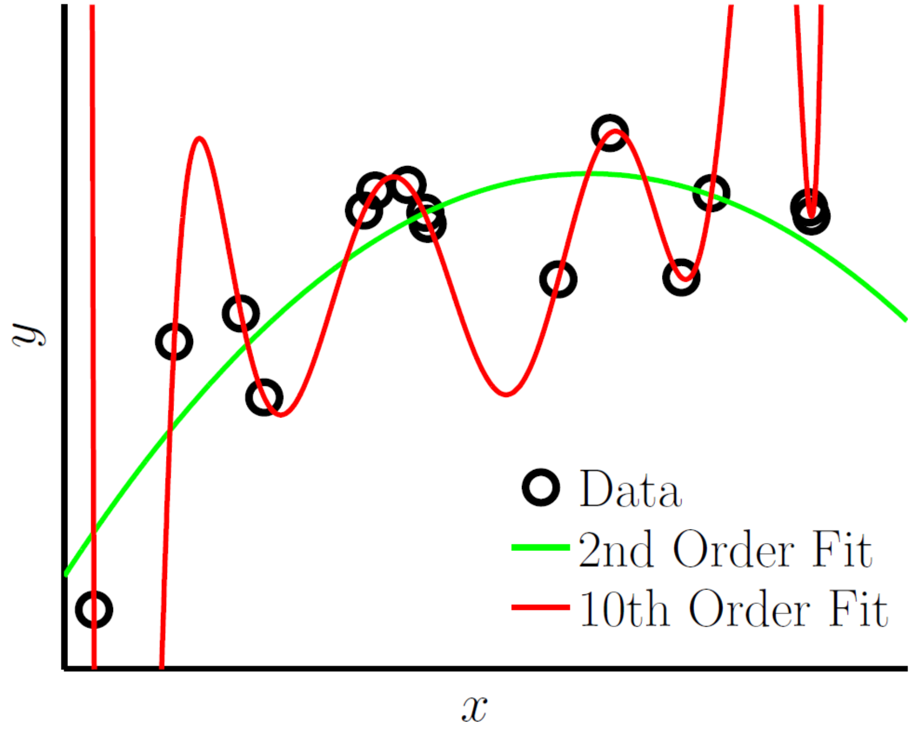
Learn linear model with 10th order polynomial transform

2nd Order vs 10th Order Polynomial



simple noisy target

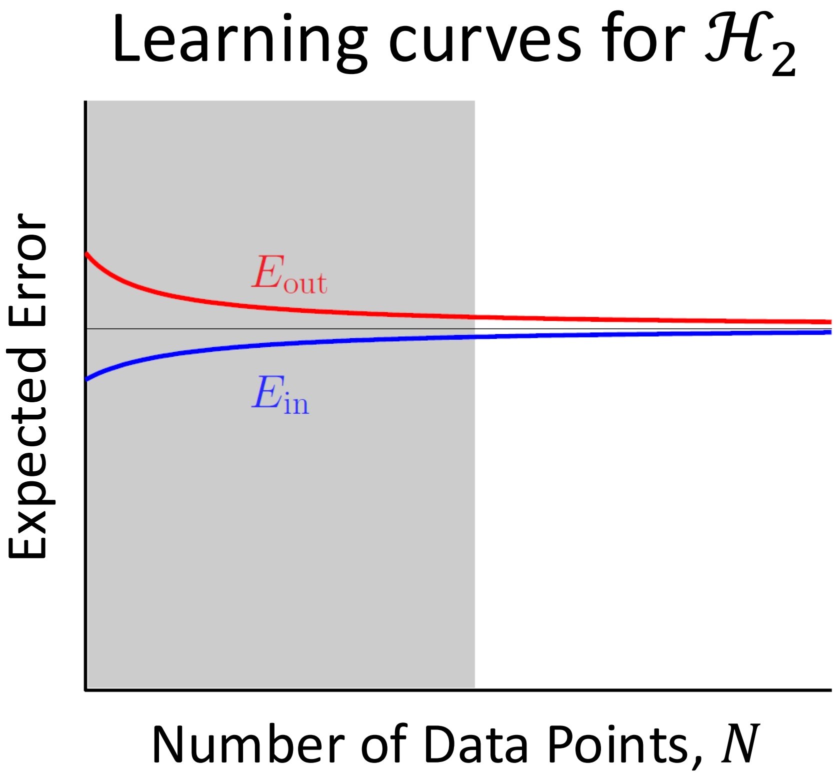
	2nd order	10th order
E_{in}	0.050	0.034
E_{out}	**0.127**	**9.00**

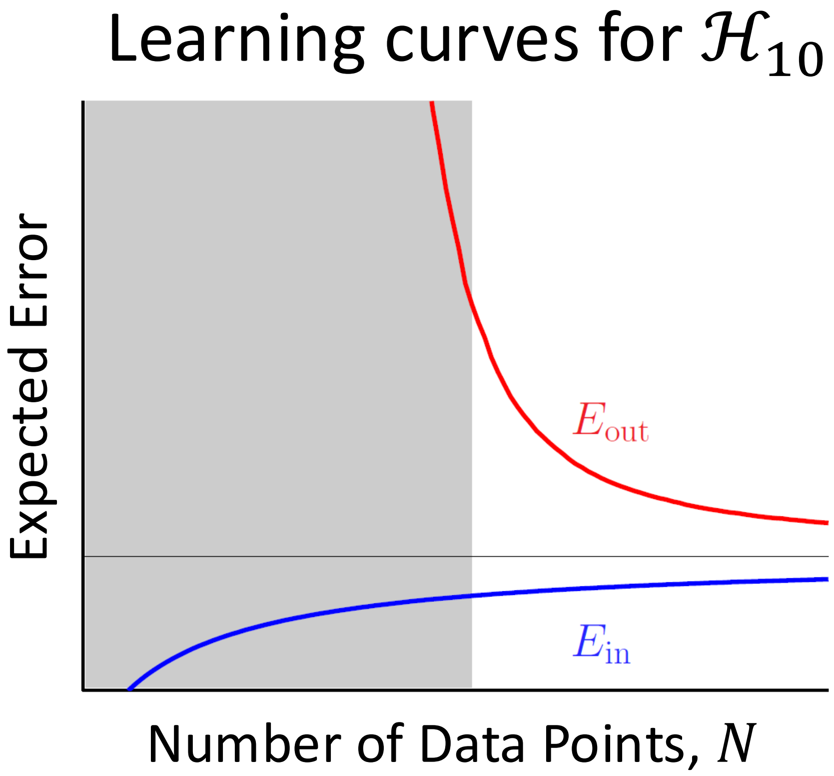


complex noiseless target Small N

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

When is $\mathcal{H}_2$ better than $\mathcal{H}_{10}$?





$\mathcal{H}$ should match quantity (N) and quality of data (noise), not the target (f)

Overfitting: When $E_{out}(\mathcal{H}_{10}) > E_{out}(\mathcal{H}_2)$

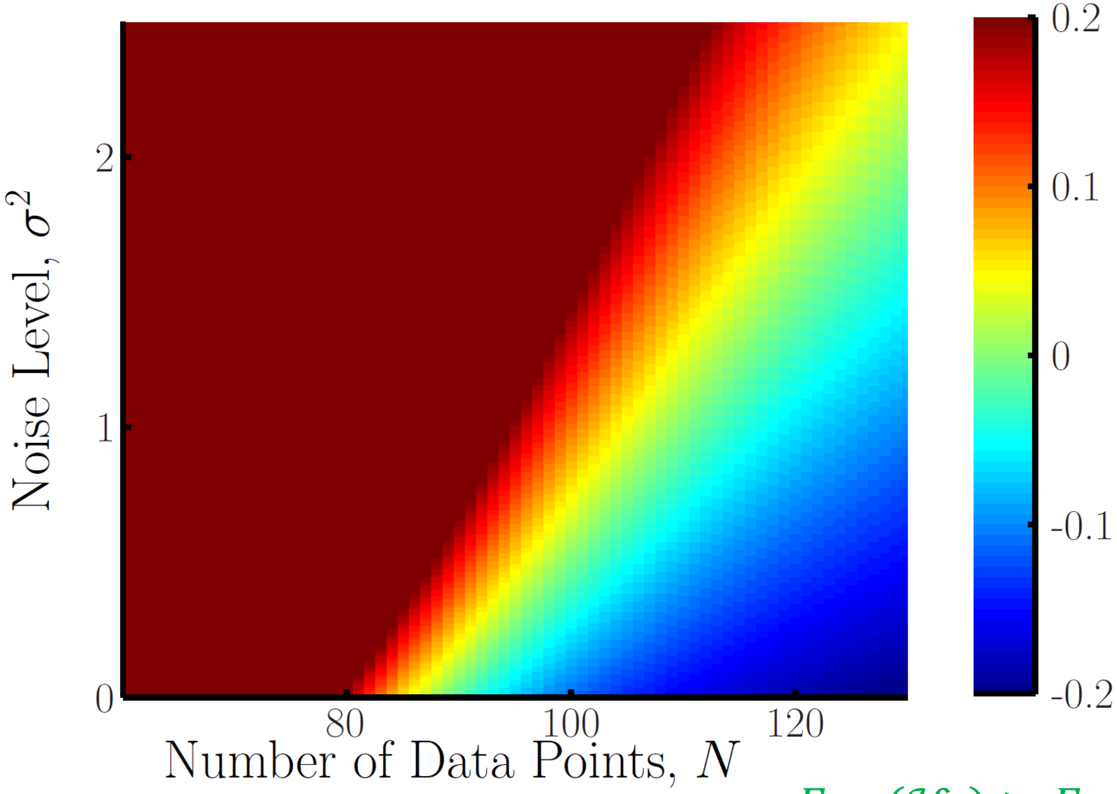
Overfit Measure: $E_{out}(\mathcal{H}_{10}) - E_{out}(\mathcal{H}_2)$

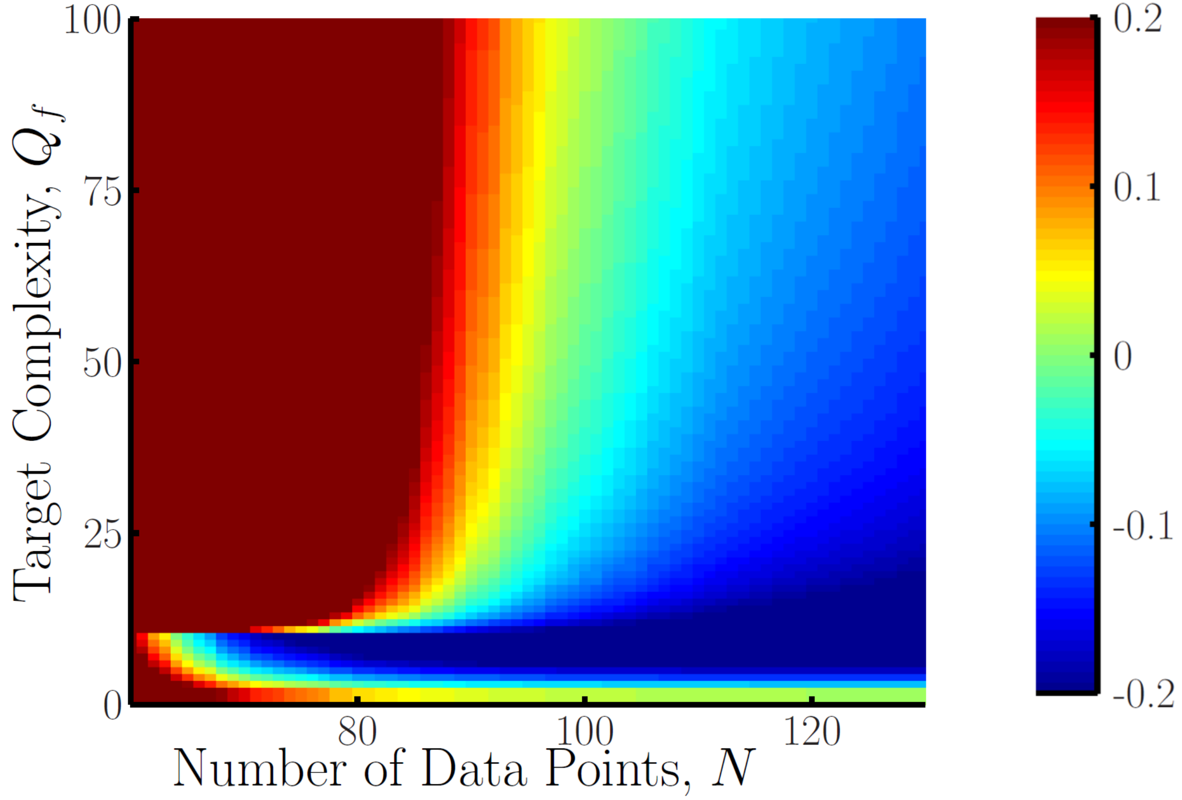
Learning Scenario Parameters:

- Noise level σ^2 (variance)
- Target complexity Q_f
- Number of Data Points N

Overfit Measure: $E_{out}(\mathcal{H}_{10}) - E_{out}(\mathcal{H}_2)$

$E_{out}(\mathcal{H}_{10}) > E_{out}(\mathcal{H}_2)$





$E_{out}(\mathcal{H}_2) > E_{out}(\mathcal{H}_{10})$

Noise level σ^2 (variance)	↑	↑	Overfitting
Target complexity Q_f	↑	↑	Overfitting
Number of Data Points N	↑	↓	Overfitting

Noise: The Part of y we cannot model

- Towards a unified view of overfitting

- Noise:
  - Stochastic noise: measurement error
  - Deterministic noise (?)

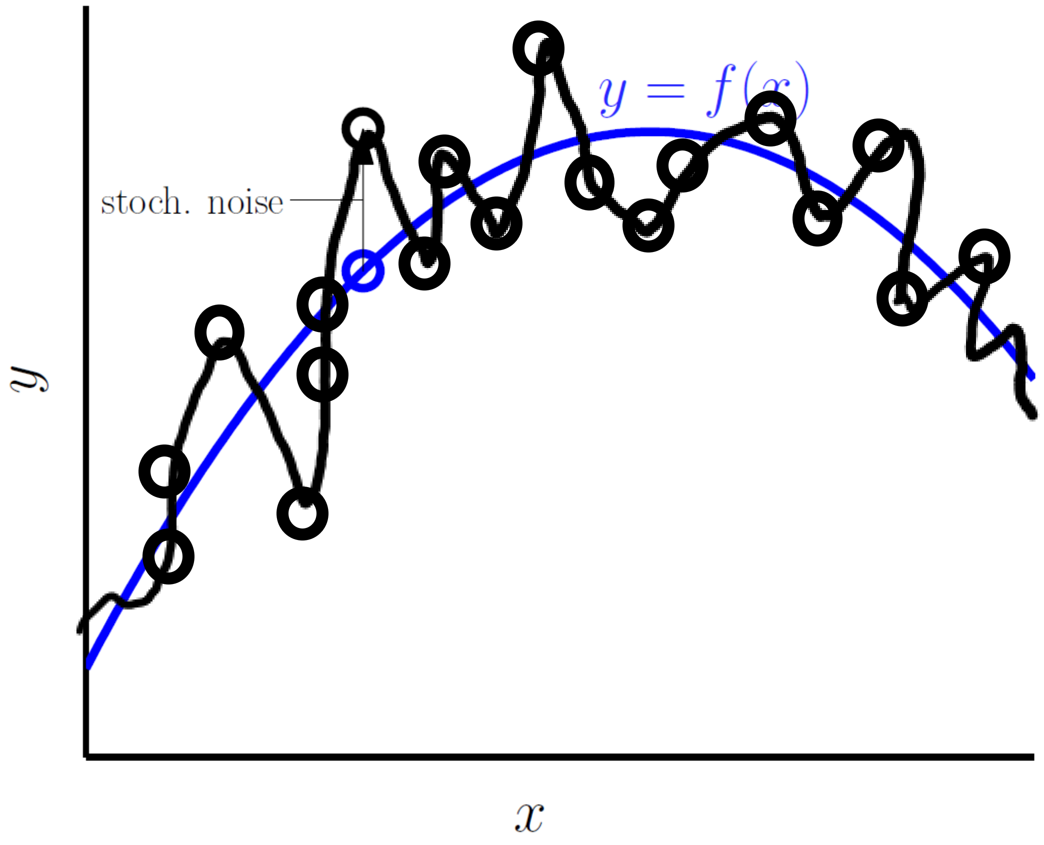
Stochastic Noise: A Part of y we cannot model

Want to learn from **o**: $y = f(x)$

Unfortunately we only observe **o**:

$$y = f(x) + \text{stochastic noise}$$

We have no way to model stochastic noise



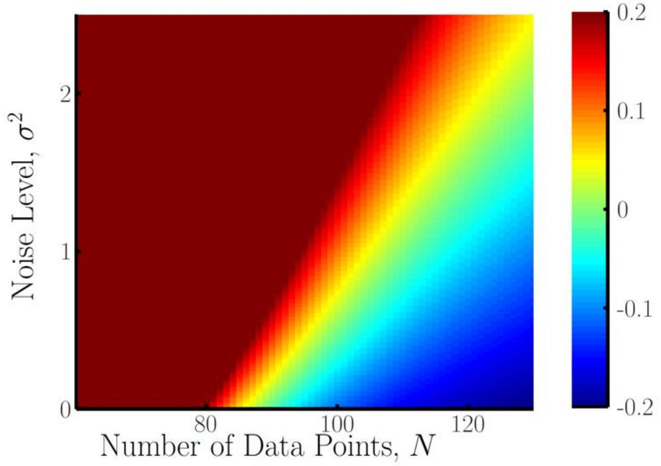
Stochastic Noise: A Part of y we cannot model

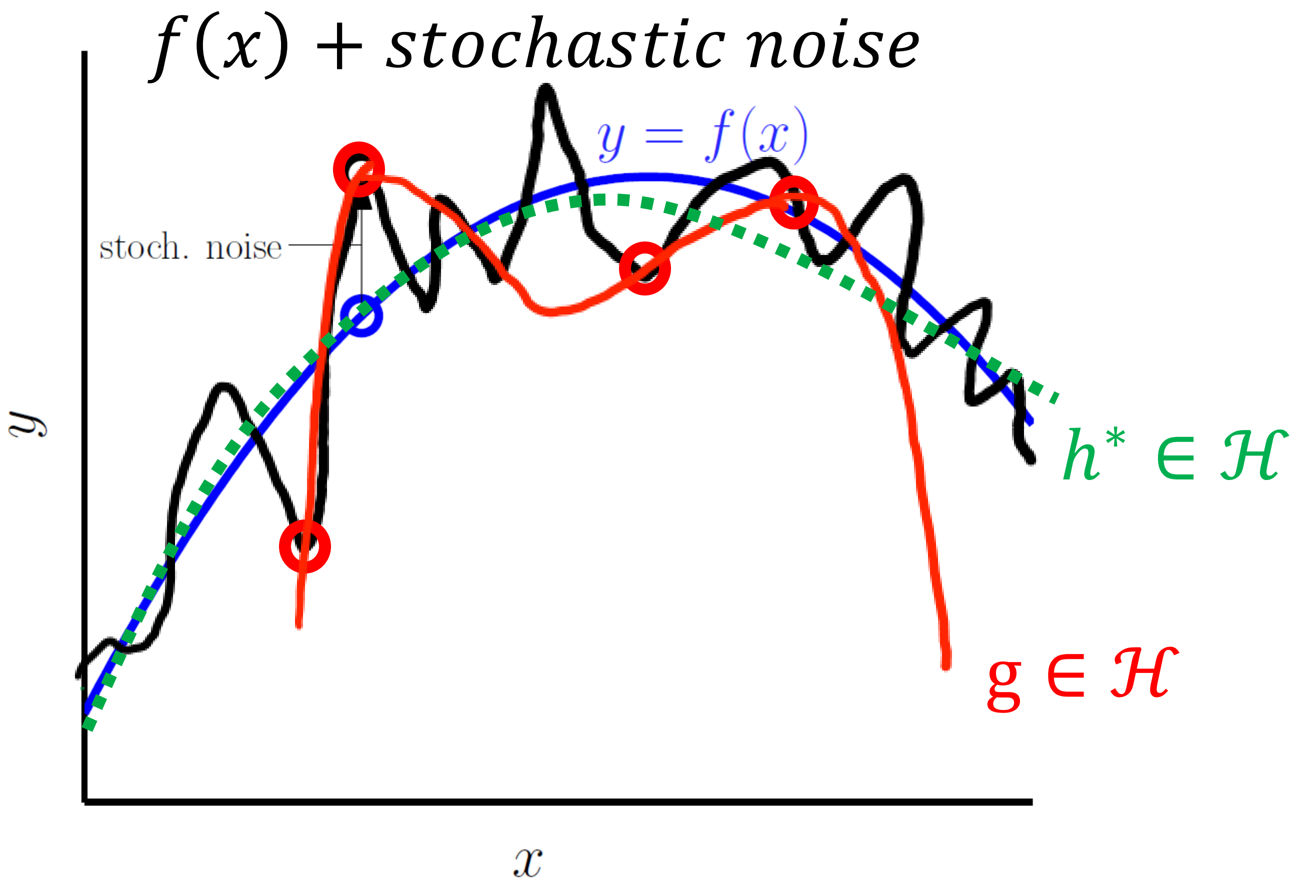
Want to learn from **o**: $y = f(x)$

Unfortunately we only observe **o**:

$y = f(x)$ + stochastic noise

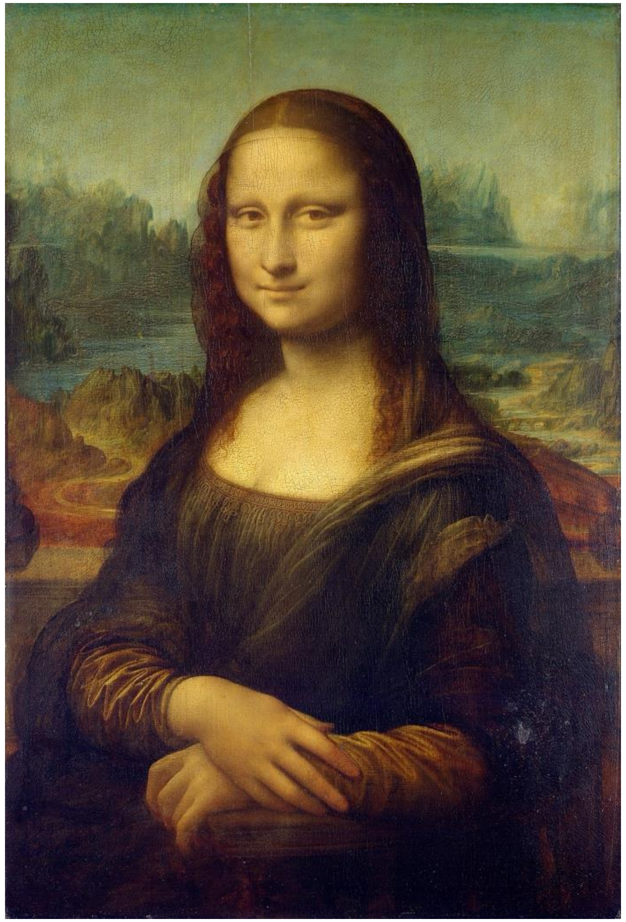
We have no way to model stochastic noise





Noise level σ^2 (variance) ↑↑ Overfitting

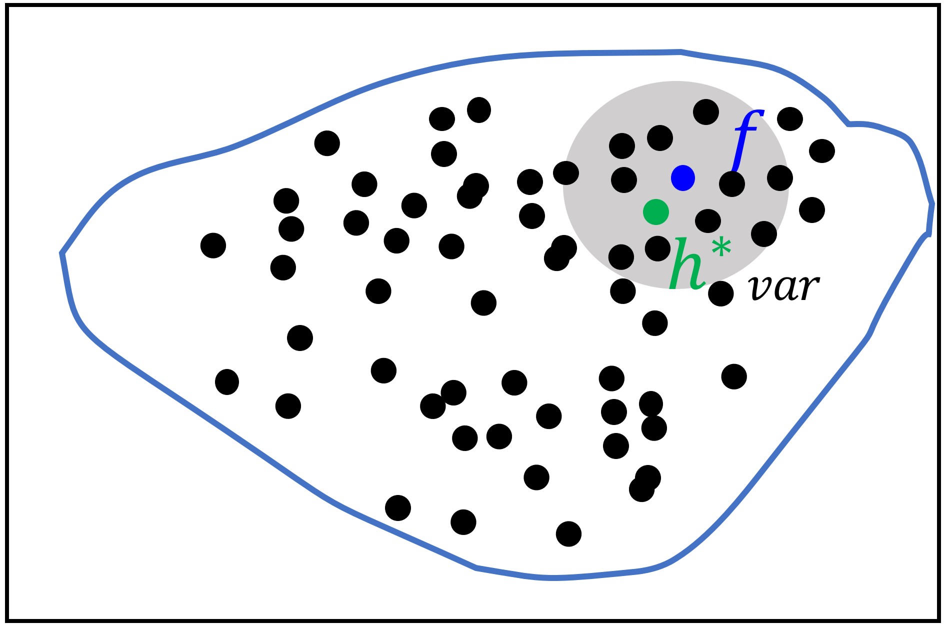
Stochastic and Deterministic Noise

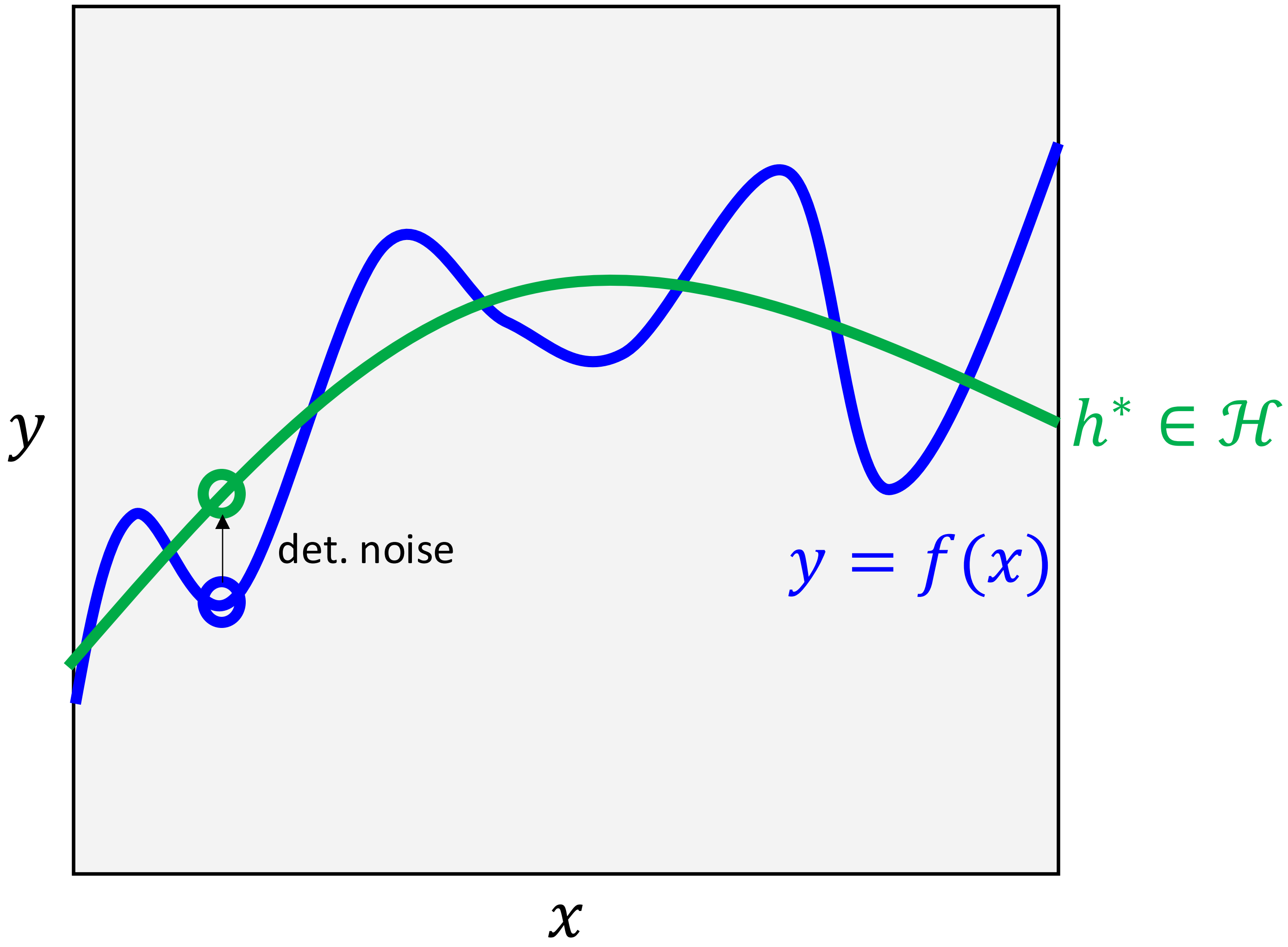


Stochastic noise:
measurement errors we cannot model

Deterministic noise:
the other part of y we cannot model
the best we can possibly do with $\mathcal{H}$

Deterministic Noise



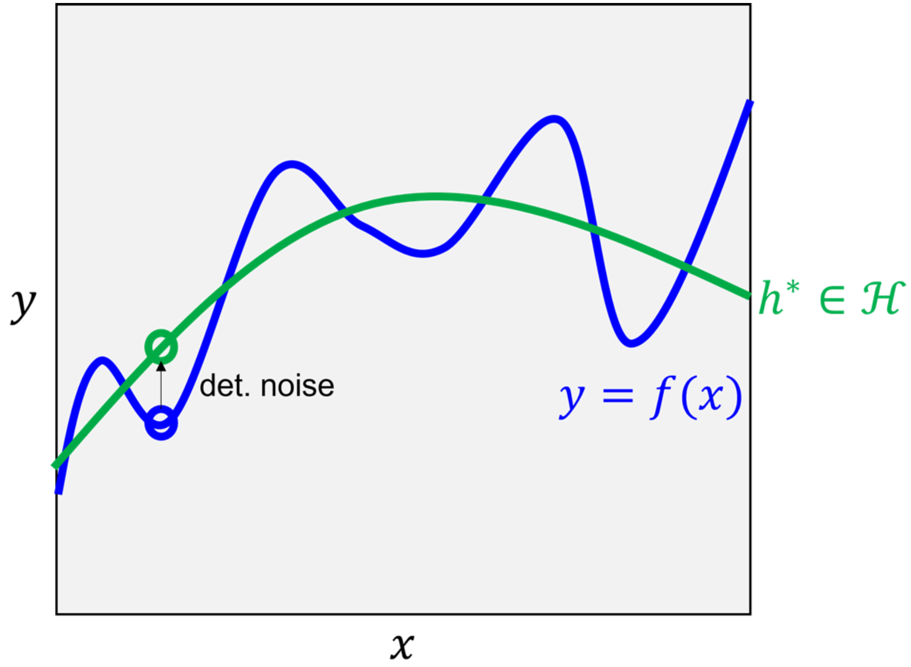


Deterministic Noise: Model Error

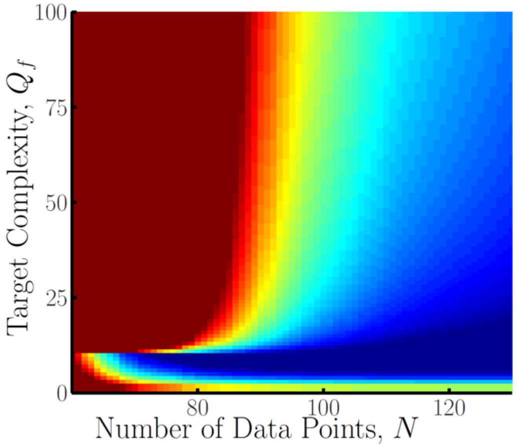
Want: Pick h^*
the best approximation of f in $\mathcal{H}$

Want: To learn from **o**

Unfortunately, we observe **o**
$y_n = f(x_n)$
$y_n = h^*(x_n)$ + deterministic noise



Target complexity $Q_f \uparrow\uparrow$ Overfitting



Stochastic and Deterministic Noise Hurt Learning

Stochastic Noise

random measurement errors

- Measuring y_n again

stochastic noise changes

- Independent of $\mathcal{H}$

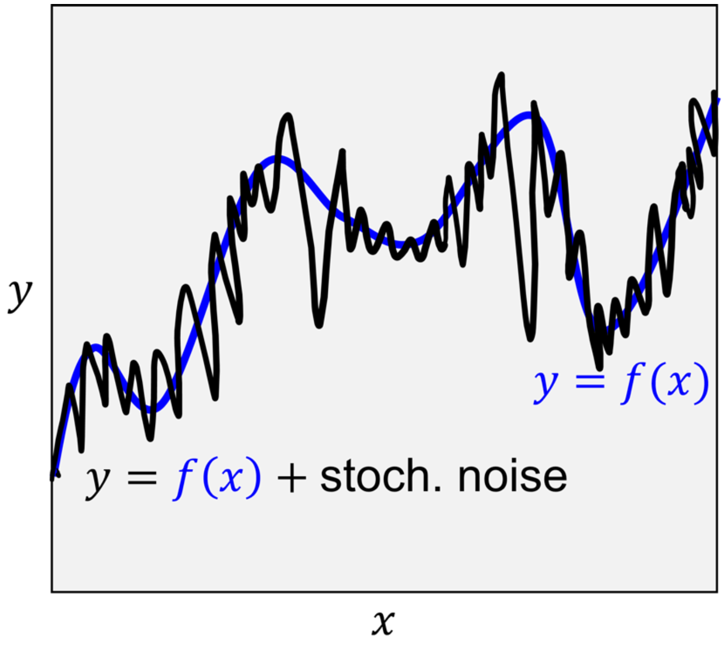
Deterministic Noise

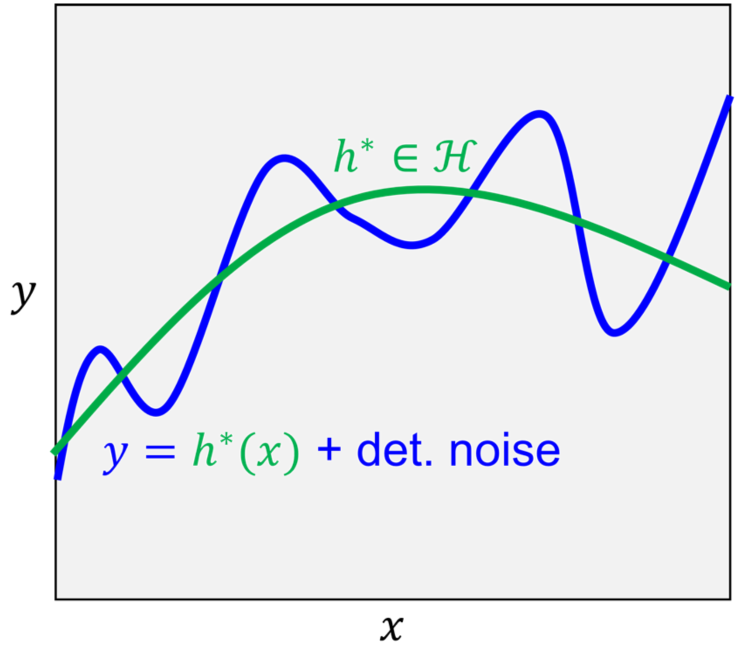
$\mathcal{H}$ cannot model f

- Measuring y_n again

deterministic noise constant

- Depends on $\mathcal{H}$





Bias Variance Decomposition with Stochastic Noise

$$\mathbb{E}[E_{out}(\boldsymbol{x})] = bias(\boldsymbol{x}) \quad + \quad var(\boldsymbol{x})$$

$$bias(\boldsymbol{x}) = \left(\bar{g}(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2$$

$$var(\boldsymbol{x}) = var(g(\boldsymbol{x}))$$

$$E_{out}(\boldsymbol{x}) = \left(g(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2$$

With noise:

$$\begin{aligned} E_{out}(\boldsymbol{x}) \quad &= (g(\boldsymbol{x}) - y)^2, \text{ where } y = f(\boldsymbol{x}) + \epsilon \\ &= (g(\boldsymbol{x}) - f(\boldsymbol{x}) - \epsilon)^2 \\ &= \left(g(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 - 2\epsilon\left(g(\boldsymbol{x}) - f(\boldsymbol{x})\right) + \epsilon^2 \end{aligned}$$

Bias Variance Decomposition with Stochastic Noise

$$E_{out}(\boldsymbol{x}) \quad = \quad \left(g(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 - 2\epsilon\left(g(\boldsymbol{x}) - f(\boldsymbol{x})\right) + \epsilon^2$$

$$\mathbb{E}_{\boldsymbol{x}}[E_{out}] \quad = \mathbb{E}_{\boldsymbol{x}}\left[\left(g(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2 - 2\epsilon\left(g(\boldsymbol{x}) - f(\boldsymbol{x})\right) + \epsilon^2\right]$$

$\mathbb{E}[\epsilon] = 0 \Rightarrow$

$$\mathbb{E}_{\boldsymbol{x}}[E_{out}] \quad = \mathbb{E}_{\boldsymbol{x}}\left[\left(g(\boldsymbol{x}) - f(\boldsymbol{x})\right)^2\right] + \mathbb{E}_{\boldsymbol{x}}[\epsilon^2]$$

$$\mathbb{E}_{\boldsymbol{x}}[E_{out}] \quad = \mathbb{E}_{\boldsymbol{x}}[bias(\boldsymbol{x}) \; + var(\boldsymbol{x}) \;\; + \epsilon^2]$$

$$E_{out} \quad = \quad bias \quad + \quad \sigma^2 \quad + \quad var$$

depends on N

det. noise   stoch. noise   indirect impact of noise