

CSE 417T

# Introduction to Machine Learning

Lecture 6

Instructor: Chien-Ju (CJ) Ho

Recap

# Theory of Generalization

- Learning from a **finite** hypothesis set: learn  $g \in \{h_1, \dots, h_M\}$

With prob  $1 - \delta$ ,  $E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{1}{2N} \ln \frac{2M}{\delta}}$

- What if  $M \rightarrow \infty$

- Dichotomies

- Informally, consider a dichotomy as a “data-dependent” hypothesis
- Characterized by both hypothesis set  $H$  and  $N$  data points  $(\vec{x}_1, \dots, \vec{x}_N)$ 
$$H(\vec{x}_1, \dots, \vec{x}_N) = \{(h(\vec{x}_1), \dots, h(\vec{x}_N)) | h \in H\}$$
- The set of possible prediction combinations  $h \in H$  can induce on  $\vec{x}_1, \dots, \vec{x}_N$

- Growth function

- Largest number of dichotomies  $H$  can induce across all possible data sets of size  $N$

$$m_H(N) = \max_{(\vec{x}_1, \dots, \vec{x}_N)} |H(\vec{x}_1, \dots, \vec{x}_N)|$$

- VC Generalization Bound

With prob  $1 - \delta$ ,  $E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_H(2N)}{\delta}}$

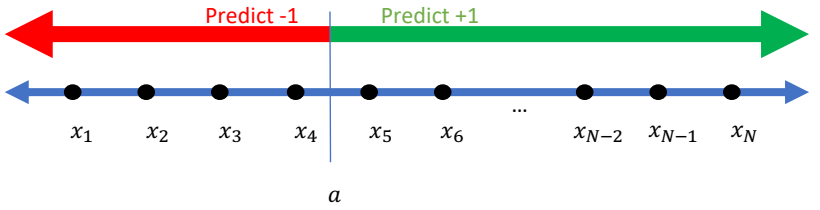
# Bounding Growth Functions

- More 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  $H$ ,  $d_{vc}(H) = k^* - 1$

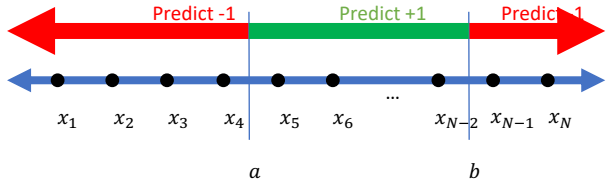
# Examples

	$m_H(N)$					Break Points	VC Dimension
	N=1	N=2	N=3	N=4	N=5		
Positive Rays	2	3	4	5	6	$k = 2, 3, 4, \dots$	1
Positive Intervals	2	4	7	11	16	$k = 3, 4, 5, \dots$	2
Convex Sets	2	4	8	16	32	None	$\infty$
2D Perceptron	2	4	8	14	?	$k = 4, 5, 6, \dots$	3

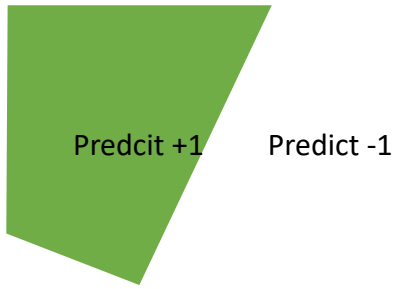
Positive Rays



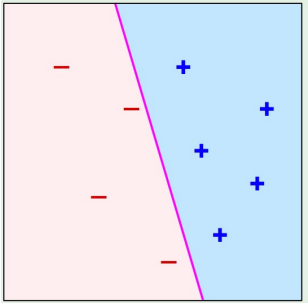
Positive Intervals



Convex Sets



2D Perceptron



# Bounding Growth Functions

- Theorem statement:

- If there is no break point for  $H$ , then  $m_H(N) = 2^N$  for all  $N$ .
- If  $k$  is a break point for  $H$ , i.e., if  $m_H(k) < 2^k$  for some value  $k$ , then

$$m_H(N) \leq \sum_{i=0}^{k-1} \binom{N}{i}$$

- Rephrase the 2<sup>nd</sup> statement of the above theorem

- If  $k$  is a break point for  $H$ , the following statements are true
  - $m_H(N) \leq N^{k-1} + 1$  [Can be proven using induction from above. See LFD Problem 2.5]
  - $m_H(N) = O(N^{k-1})$
  - $m_H(N)$  is polynomial in  $N$

- If  $d_{vc}$  is the VC dimension of  $H$ , then

- $m_H(N) \leq \sum_{i=0}^{d_{vc}} \binom{N}{i}$
- $m_H(N) \leq N^{d_{vc}} + 1$
- $m_H(N) = O(N^{d_{vc}})$

If  $d_{vc}$  is the VC dimension of  $H$ ,  
 $d_{vc} + 1$  is a break point for  $H$

# Vapnik–Chervonenkis (VC) Bound

- VC Generalization Bound

With prob at least  $1 - \delta$

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{4m_H(2N)}{\delta}}$$

- Let  $d_{vc}$  be the VC dimension of  $H$ , we have  $m_H(N) \leq N^{d_{vc}} + 1$ .

With prob at least  $1 - \delta$

$$E_{out}(g) \leq E_{in}(g) + \sqrt{\frac{8}{N} \ln \frac{4((2N)^{d_{vc}} + 1)}{\delta}}$$

- If we treat  $\delta$  as a constant, then we can say, with high probability

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

# Discussion on the VC Bound

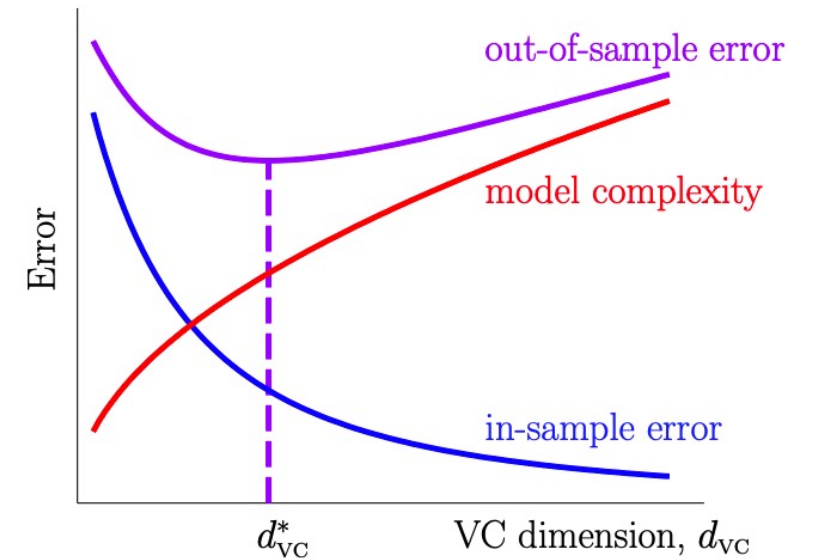
- Think about the high-level tradeoff of choosing  $d_{VC}$  and its dependency on  $N$
- The approximation-generalization trade-off

What we want to minimize

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

How well  $g$  generalizes

How well  $g$  approximates  $f$  in training data





# Today's Lecture

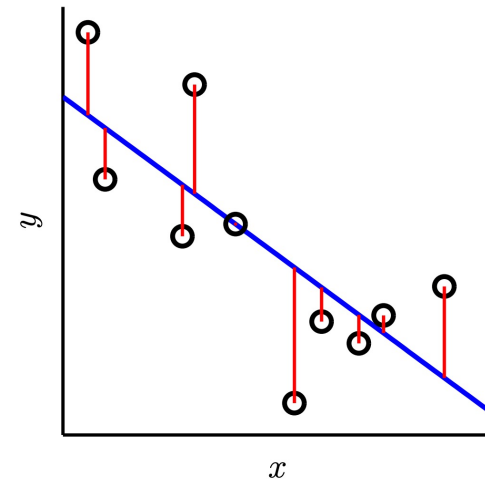
The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.  
Let me know if you spot errors.

# Bias-Variance Decomposition

Another theory of generalization

# Real-Value Target and Squared Error

- So far, we focus on binary target function and binary error
  - Binary target function  $f(\vec{x}) \in \{-1, 1\}$
  - Binary error  $e(h(\vec{x}), f(\vec{x})) = \mathbb{I}[h(\vec{x}) \neq f(\vec{x})]$
- Real-value target functions [“**regression**”] and squared error?
  - Real-value target function  $f(\vec{x}) \in \mathbb{R}$
  - Squared error  $e(h(\vec{x}), f(\vec{x})) = (h(\vec{x}) - f(\vec{x}))^2$



# Real-Value Target and Squared Error

- Real-value target functions [called "regression"] and squared error?
  - Real-value target function  $f(\vec{x}) \in \mathbb{R}$
  - Squared error  $e(h(\vec{x}), f(\vec{x})) = (h(\vec{x}) - f(\vec{x}))^2$
- Errors:
  - In-sample error:  $E_{in}(g) = \frac{1}{N} \sum_{n=1}^N e(h(\vec{x}_n), f(\vec{x}_n)) = \frac{1}{N} \sum_{n=1}^N (h(\vec{x}_n) - f(\vec{x}_n))^2$
  - Out-of-sample error:  $E_{out}(g) = \mathbb{E}_{\vec{x}}[e(h(\vec{x}), f(\vec{x}))] = \mathbb{E}_{\vec{x}}[(g(\vec{x}) - f(\vec{x}))^2]$
- Theory of generalization: What can we say about  $E_{out}(g)$ ?

- Note that  $g$  is learned by some algorithm on the dataset  $D$ 
  - We'll make the dependency on  $D$  explicit and write it as  $g^{(D)}$  here.
  - [In VC theory, we consider the worst-case  $D$  through the definition of growth function  $m_H(N)$ ]

- $E_{out}(g^{(D)}) = \mathbb{E}_{\vec{x}}[(g^{(D)}(\vec{x}) - f(\vec{x}))^2]$

- $\mathbb{E}_D[E_{out}(g^{(D)})]$

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

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

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

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

- Note that  $\mathbb{E}_D \left[ (g^{(D)}(\vec{x}) - \bar{g}(\vec{x}))(\bar{g}(\vec{x}) - f(\vec{x})) \right] = (\bar{g}(\vec{x}) - f(\vec{x})) \mathbb{E}_D \left[ (g^{(D)}(\vec{x}) - \bar{g}(\vec{x})) \right] = 0$

Define “expected” hypothesis  
 $\bar{g}(\vec{x}) = \mathbb{E}_D[g^{(D)}(\vec{x})]$

$$\bar{g}(\vec{x}) = \mathbb{E}_D[g^{(D)}(\vec{x})]$$

# Finishing Up

- $$\begin{aligned} & \mathbb{E}_D[E_{out}(g^{(D)})] \\ &= \mathbb{E}_{\vec{x}} \left[ \mathbb{E}_D \left[ \left( g^{(D)}(\vec{x}) - \bar{g}(\vec{x}) \right)^2 + \left( \bar{g}(\vec{x}) - f(\vec{x}) \right)^2 \right] \right] \\ &= \mathbb{E}_{\vec{x}} \left[ \mathbb{E}_D \left[ \left( g^{(D)}(\vec{x}) - \bar{g}(\vec{x}) \right)^2 \right] \right] + \mathbb{E}_{\vec{x}} \left[ \left( \bar{g}(\vec{x}) - f(\vec{x}) \right)^2 \right] \\ &= \mathbb{E}_{\vec{x}} [\text{Variance of } g^{(D)}(\vec{x}) + \text{Bias of } \bar{g}(\vec{x})] \\ &= \text{Variance} + \text{Bias} \end{aligned}$$

$X$ : a random variable  
 $\mu$ : the mean of  $X$

Variance of  $X$ :  
 $Var(X) = \mathbb{E}[(X - \mu)^2]$

- Bias-Variance Decomposition

# Discussion

$$\bullet \mathbb{E}_D[E_{out}(g^{(D)})] = \mathbb{E}_{\vec{x}} \left[ \overset{\text{Bias}(\vec{x})}{(\bar{g}(\vec{x}) - f(\vec{x}))^2} \right] + \mathbb{E}_{\vec{x}} \left[ \overset{\text{Var}(\vec{x})}{\mathbb{E}_D \left[ (g^{(D)}(\vec{x}) - \bar{g}(\vec{x}))^2 \right]} \right]$$

- This is a **conceptual** decomposition
  - Both  $\bar{g}$  and  $f$  are unknown
  - We can't really calculate bias and variance in practice
- However, it provides a conceptual guideline in decreasing  $E_{out}$

# Example of Bias-Variance Decomposition

- Fitting a sine function
  - $f(x) = \sin(\pi x)$
  - $x$  is drawn uniformly at random from  $[0,2]$
- Two hypothesis set
  - $H_0: h(x) = b$
  - $H_1: h(x) = ax + b$
- Assume our algorithm finds  $g$  with minimum in-sample error

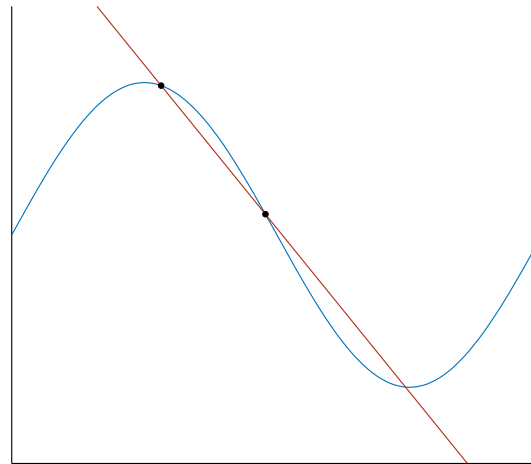
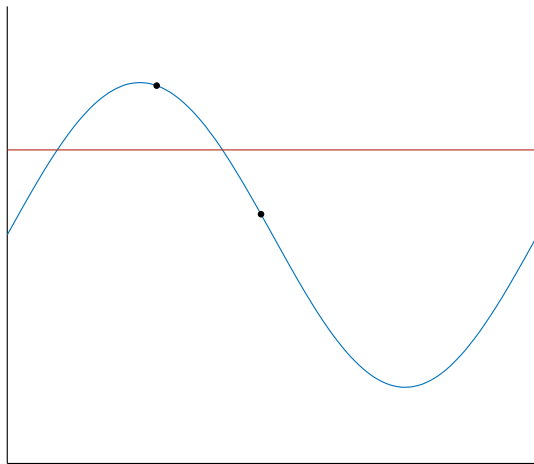


# Example of Bias-Variance Decomposition

$$H_0: h(x) = b$$

$$H_1: h(x) = ax + b$$

$N=2$



$$\mathbb{E}_D[E_{out}(g^{(D)})] = \mathbb{E}_{\vec{x}} \left[ \overset{\text{Bias}(\vec{x})}{(\bar{g}(\vec{x}) - f(\vec{x}))^2} \right] + \mathbb{E}_{\vec{x}} \left[ \mathbb{E}_D \left[ \overset{\text{Var}(\vec{x})}{(g^{(D)}(\vec{x}) - \bar{g}(\vec{x}))^2} \right] \right]$$

## Discussion:

If  $N = 2$ , would you choose  $H_0$  or  $H_1$ ? Why?

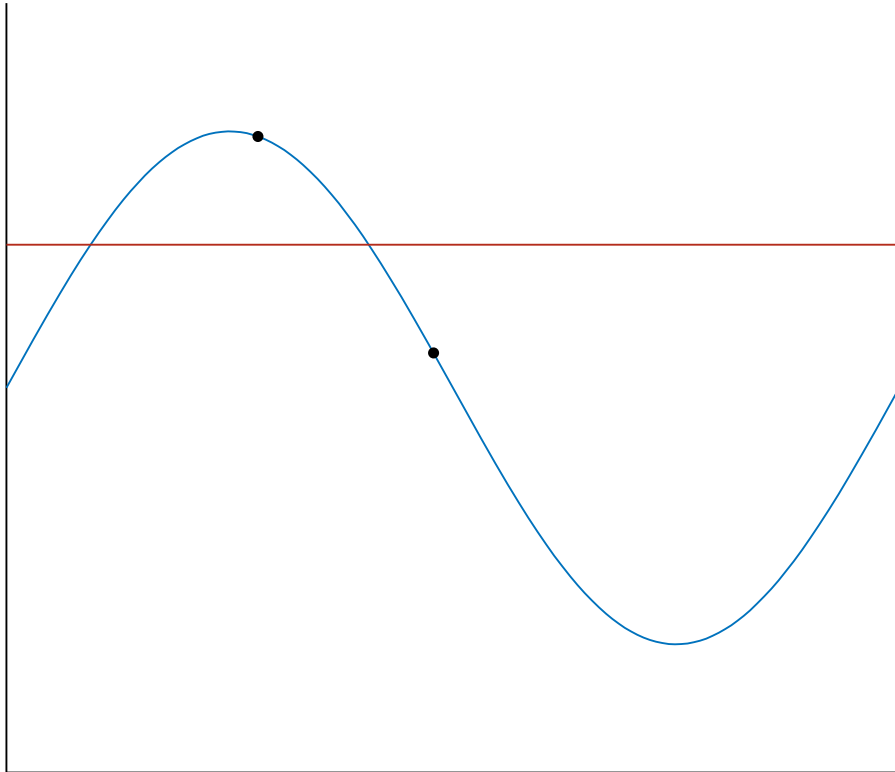
If  $N = 5$ , would you choose  $H_0$  or  $H_1$ ? Why?

What's the change of biases/variances for  $H_0/H_1$  from  $N = 2$  to  $N = 5$ .

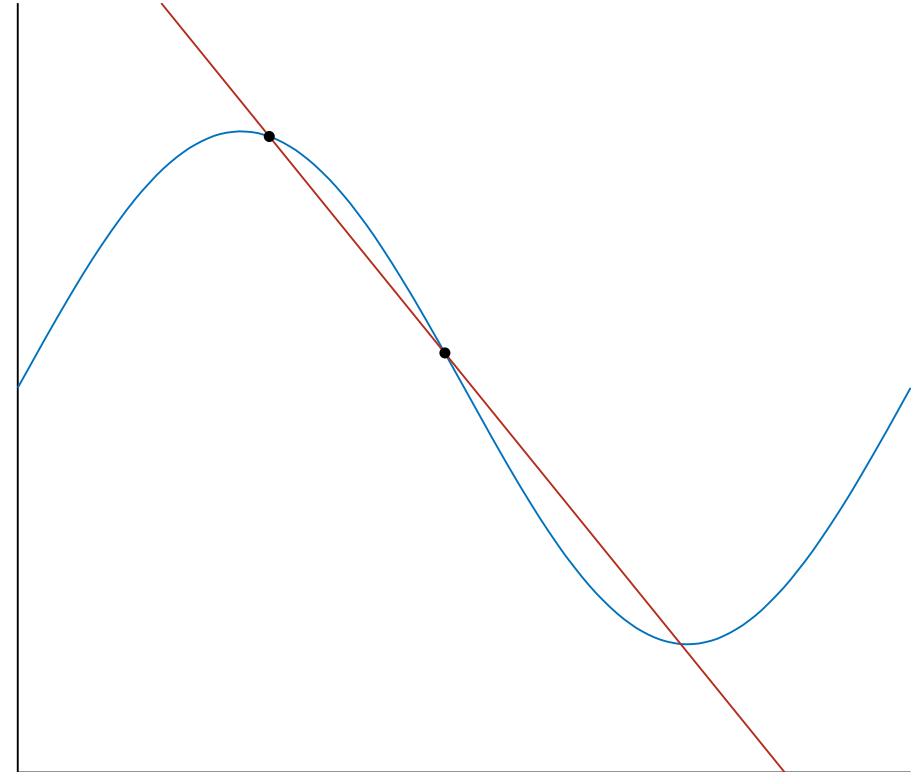
# Example of Bias-Variance Decomposition

$$H_0: h(x) = b$$

$N=2$



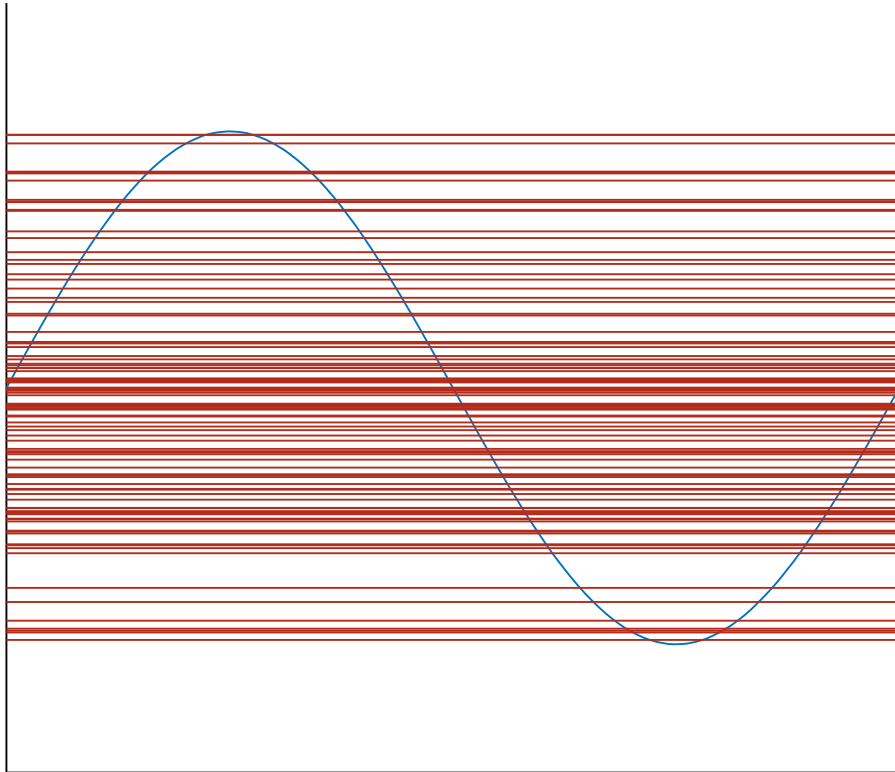
$$H_1: h(x) = ax + b$$



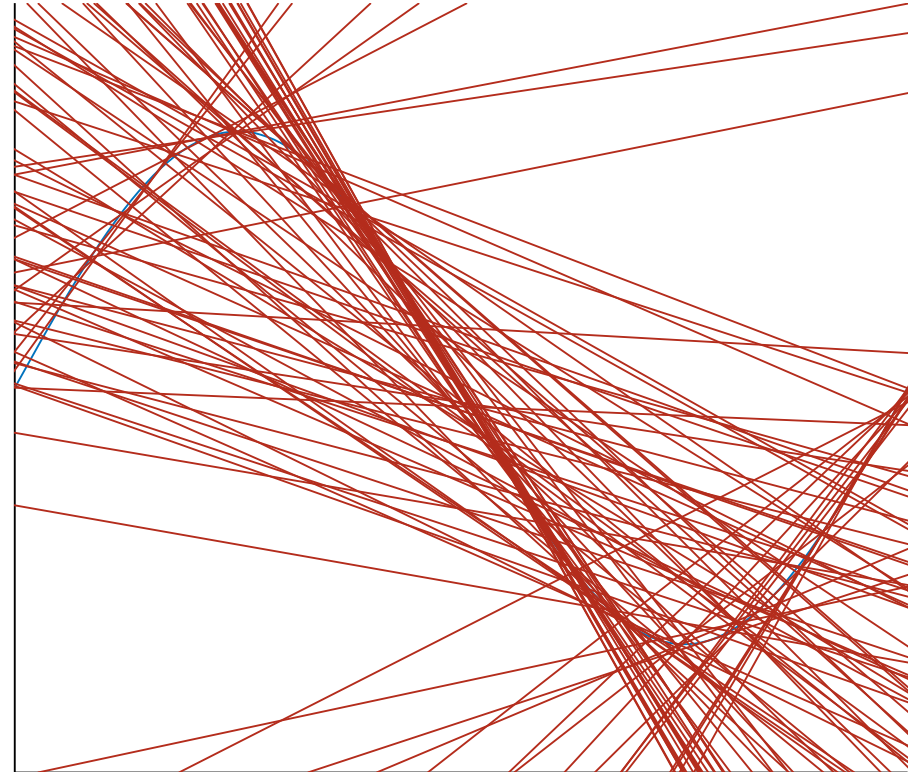
# Example of Bias-Variance Decomposition

$$H_0: h(x) = b$$

$N=2$



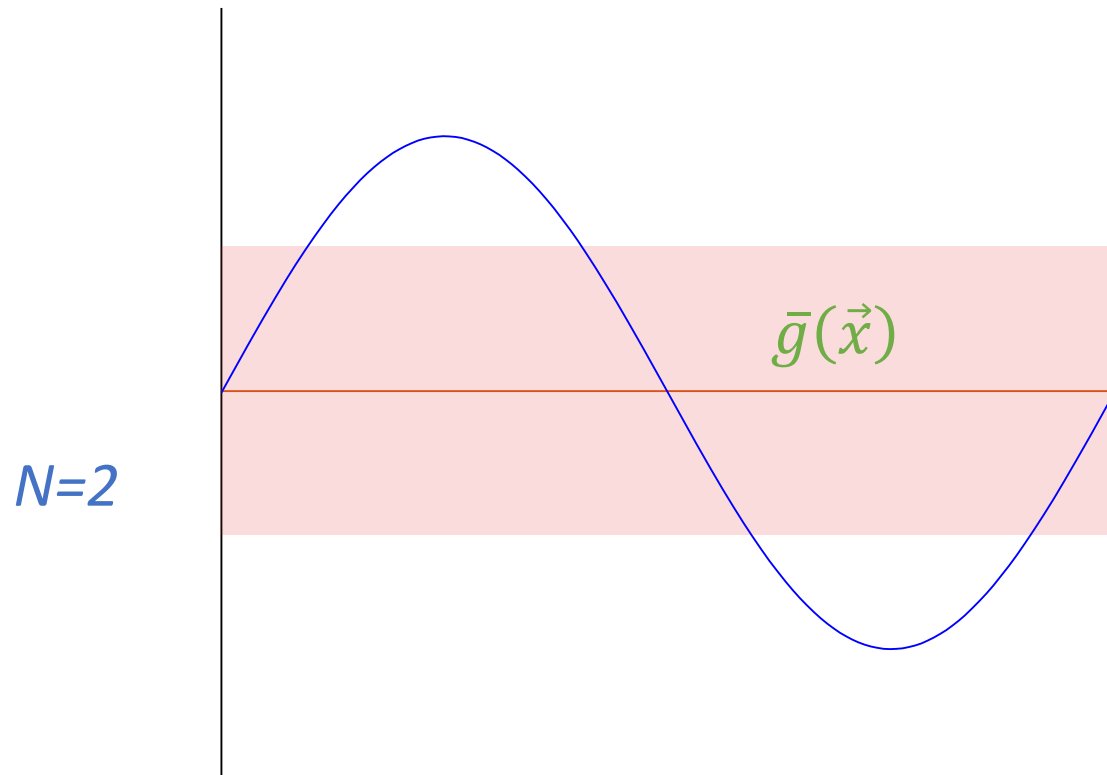
$$H_1: h(x) = ax + b$$



$$\mathbb{E}_D[E_{out}(g^{(D)})] = \mathbb{E}_{\vec{x}} \left[ \underbrace{(\bar{g}(\vec{x}) - f(\vec{x}))^2}_{\text{Bias}(\vec{x})} \right] + \mathbb{E}_{\vec{x}} \left[ \mathbb{E}_D \left[ \underbrace{(g^{(D)}(\vec{x}) - \bar{g}(\vec{x}))^2}_{\text{Var}(\vec{x})} \right] \right]$$

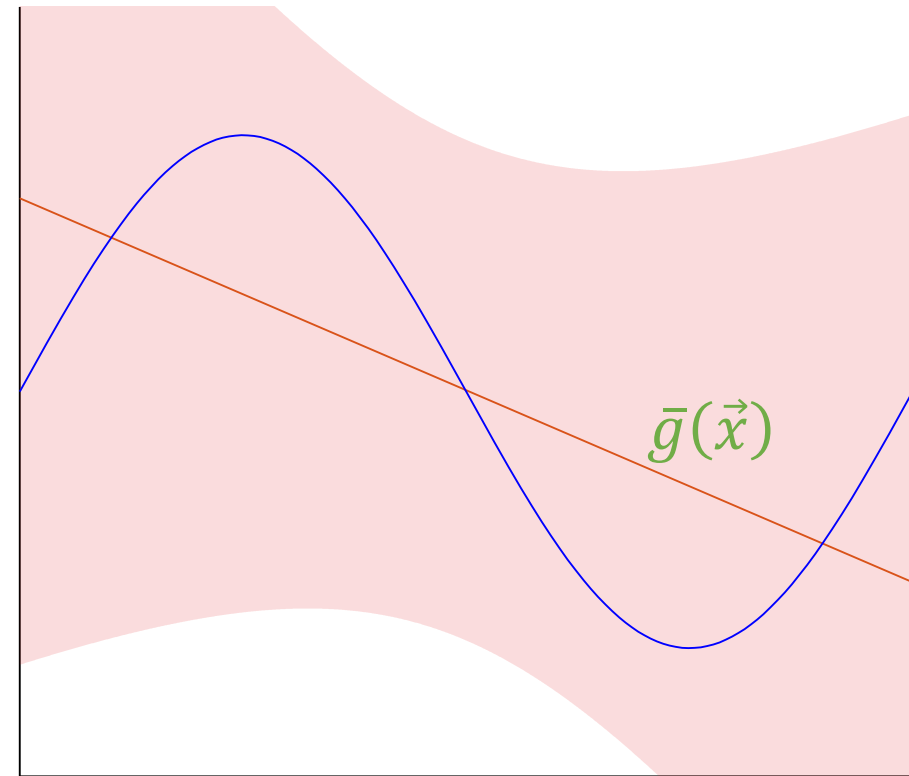
# Example of Bias-Variance Decomposition

$$H_0: h(x) = b$$



Bias of  $\bar{g}(\vec{x}) \approx 0.50$   
 Variance of  $g_D(\vec{x}) \approx 0.25$   
 $\mathbb{E}_D[E_{out}(g_D)] \approx 0.75$

$$H_1: h(x) = ax + b$$

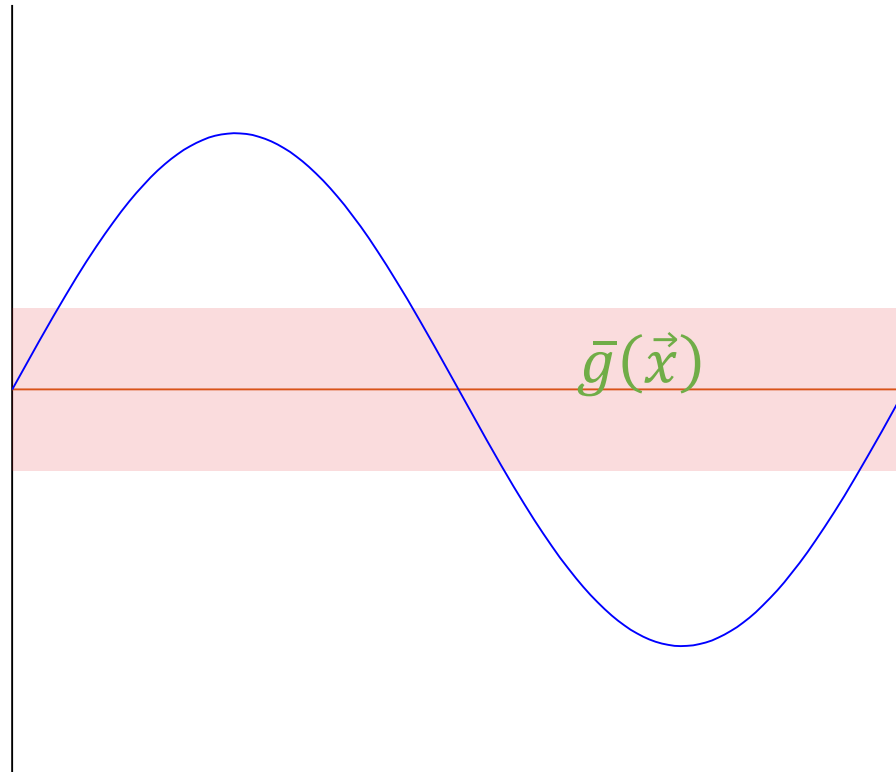


Bias of  $\bar{g}(\vec{x}) \approx 0.21$   
 Variance of  $g_D(\vec{x}) \approx 1.74$   
 $\mathbb{E}_D[E_{out}(g_D)] \approx 1.95$

# What if we increase $N$ to 5?

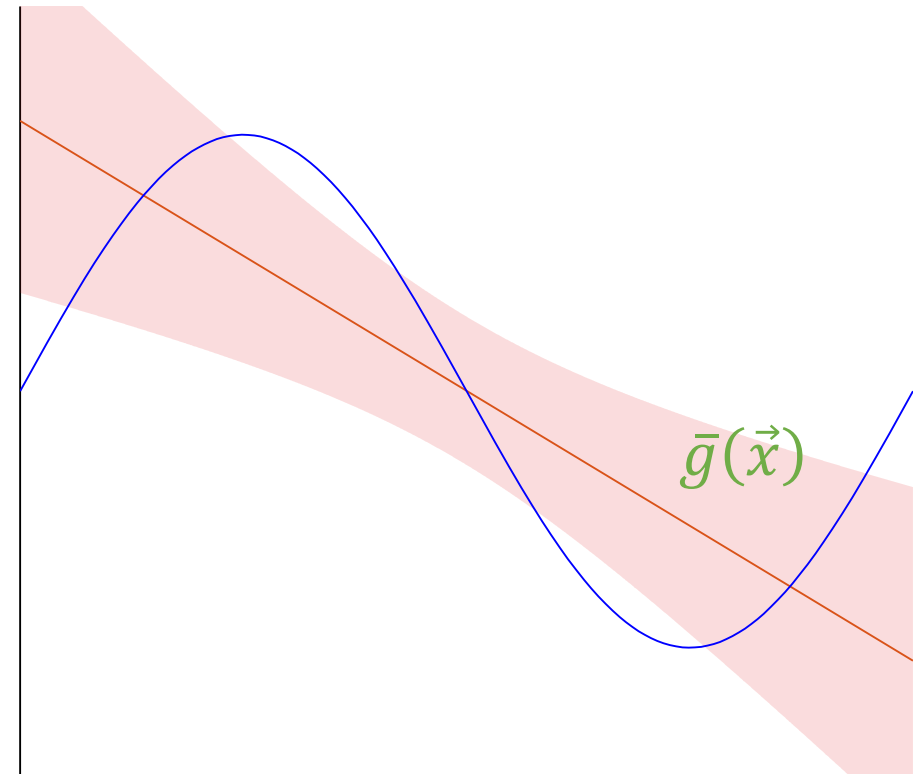
$$\mathbb{E}_{\mathcal{D}}[E_{out}(g^{(\mathcal{D})})] = \mathbb{E}_{\vec{x}} \left[ \underbrace{(\bar{g}(\vec{x}) - f(\vec{x}))^2}_{\text{Bias}(\vec{x})} \right] + \mathbb{E}_{\vec{x}} \left[ \mathbb{E}_{\mathcal{D}} \left[ \underbrace{(g^{(\mathcal{D})}(\vec{x}) - \bar{g}(\vec{x}))^2}_{\text{Var}(\vec{x})} \right] \right]$$

$$H_0: h(x) = b$$



$$\begin{aligned} \text{Bias of } \bar{g}(\vec{x}) &\approx 0.50 \\ \text{Variance of } g_{\mathcal{D}}(\vec{x}) &\approx 0.10 \\ \mathbb{E}_{\mathcal{D}}[E_{out}(g_{\mathcal{D}})] &\approx 0.60 \end{aligned}$$

$$H_1: h(x) = ax + b$$



$$\begin{aligned} \text{Bias of } \bar{g}(\vec{x}) &\approx 0.21 \\ \text{Variance of } g_{\mathcal{D}}(\vec{x}) &\approx 0.21 \\ \mathbb{E}_{\mathcal{D}}[E_{out}(g_{\mathcal{D}})] &\approx 0.42 \end{aligned}$$

# Discussion

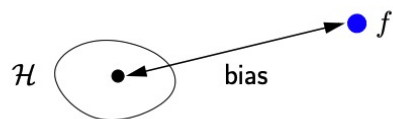
$$\bullet \mathbb{E}_D[E_{out}(g^{(D)})] = \mathbb{E}_{\vec{x}} \left[ \overset{\text{Bias}(\vec{x})}{(\bar{g}(\vec{x}) - f(\vec{x}))^2} \right] + \mathbb{E}_{\vec{x}} \left[ \overset{\text{Var}(\vec{x})}{\mathbb{E}_D \left[ (g^{(D)}(\vec{x}) - \bar{g}(\vec{x}))^2 \right]} \right]$$

- Increasing the number of data points  $N$ 
  - Biases roughly stay the same
  - Variances decrease
  - Expected  $E_{out}$  decreases

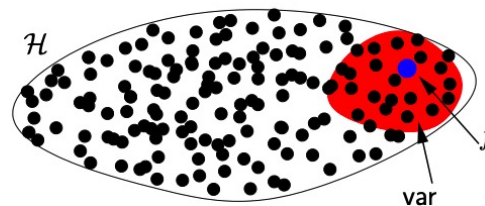
# Discussion

$$\bullet \mathbb{E}_D[E_{out}(g^{(D)})] = \mathbb{E}_{\vec{x}} \left[ \overset{\text{Bias}(\vec{x})}{(\bar{g}(\vec{x}) - f(\vec{x}))^2} \right] + \mathbb{E}_{\vec{x}} \left[ \overset{\text{Var}(\vec{x})}{\mathbb{E}_D \left[ (g^{(D)}(\vec{x}) - \bar{g}(\vec{x}))^2 \right]} \right]$$

- Increasing the complexity of  $H$ 
  - Bias goes down (more likely to approximate  $f$ )
  - Variance goes up (The stability of  $g^{(D)}$  is worse)



Very small model



Very large model

# Discussion

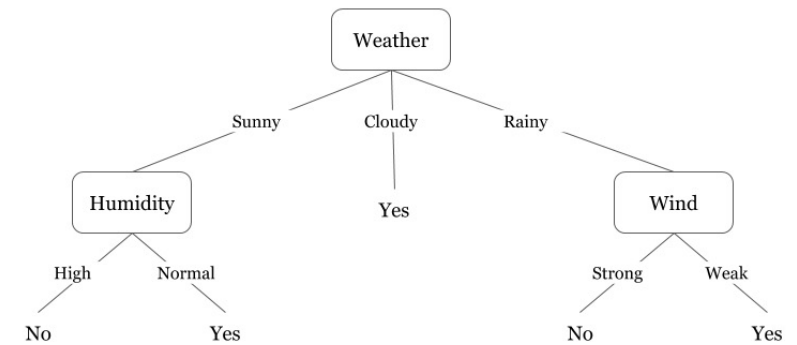
$$\bullet \mathbb{E}_D[E_{out}(g^{(D)})] = \mathbb{E}_{\vec{x}} \left[ \overset{\text{Bias}(\vec{x})}{(\bar{g}(\vec{x}) - f(\vec{x}))^2} \right] + \mathbb{E}_{\vec{x}} \left[ \overset{\text{Var}(\vec{x})}{\mathbb{E}_D \left[ (g^{(D)}(\vec{x}) - \bar{g}(\vec{x}))^2 \right]} \right]$$

- This is a **conceptual** decomposition
  - Both  $\bar{g}$  and  $f$  are unknown
  - We can't really calculate bias and variance for practical problems
- However, it provides a conceptual guidelines in decreasing  $E_{out}$



# Example

- Will talk about this in details in the 2<sup>nd</sup> half of the semester
- Decision tree
  - A low bias but high variance hypothesis set
  - Practical performance is not ideal



- Random forest
  - Trying to reduce the variance while not sacrificing bias
  - Idea: Generate many trees randomly and average them

# Two Theories of Generalization

- VC Generalization Bound

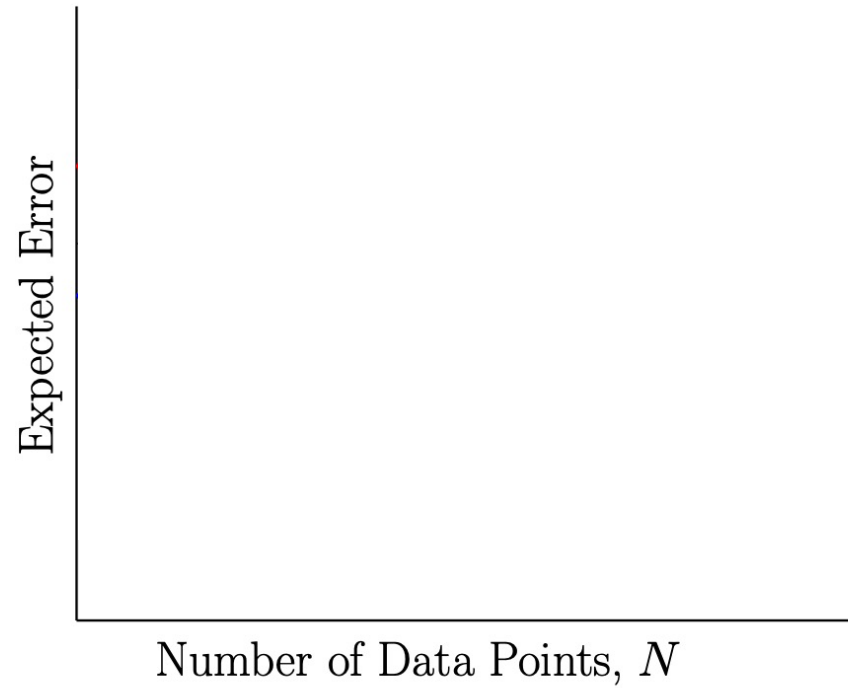
$$E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{vc} \frac{\ln N}{N}}\right)$$

- Bias-Variance Tradeoff

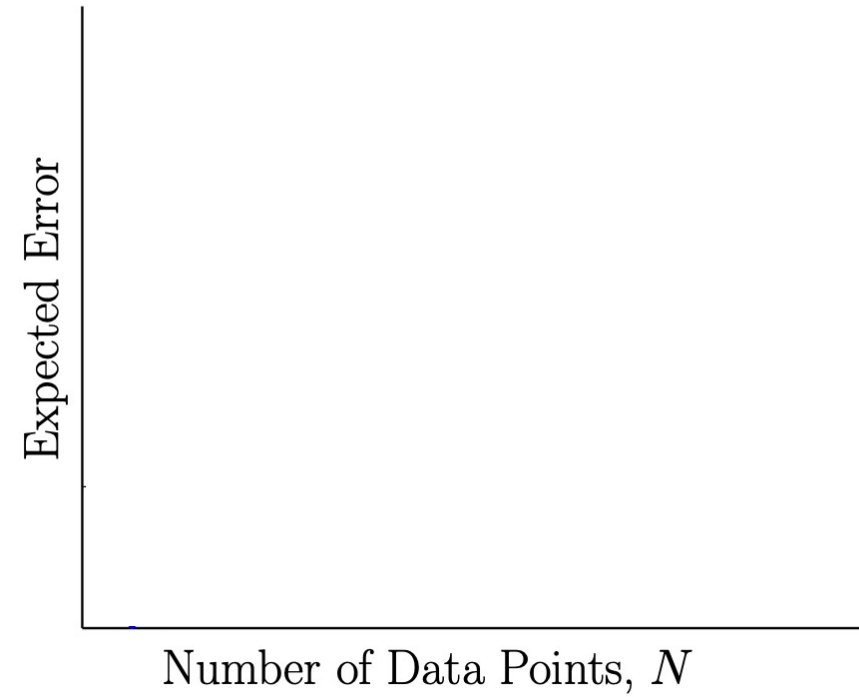
$$\mathbb{E}_D[E_{out}(g^{(D)})] = \mathbb{E}_{\vec{x}} \left[ (\bar{g}(\vec{x}) - f(\vec{x}))^2 \right] + \mathbb{E}_{\vec{x}} \left[ \mathbb{E}_D \left[ (g^{(D)}(\vec{x}) - \bar{g}(\vec{x}))^2 \right] \right]$$

# Learning Curves

Simple Model

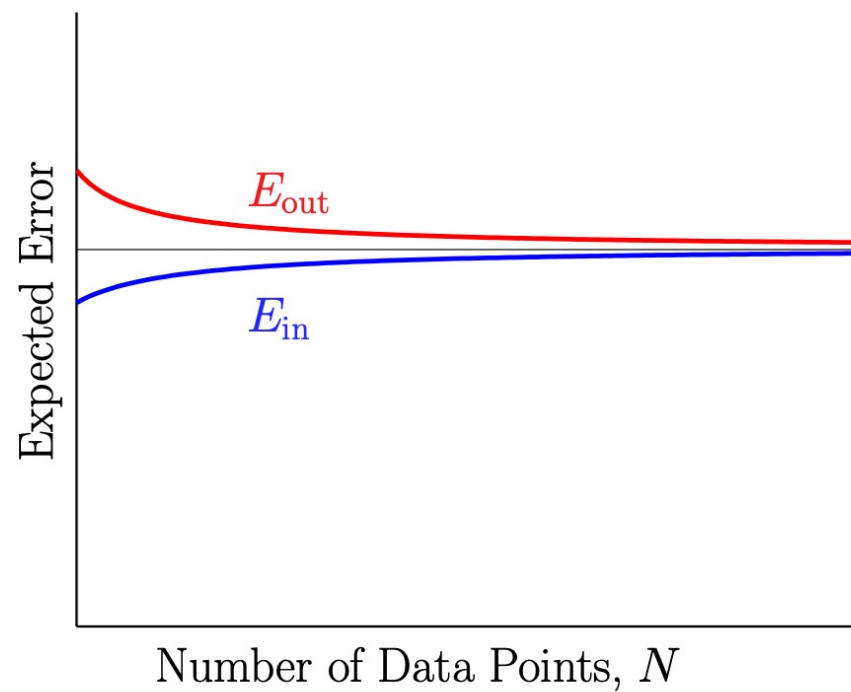


Complex Model

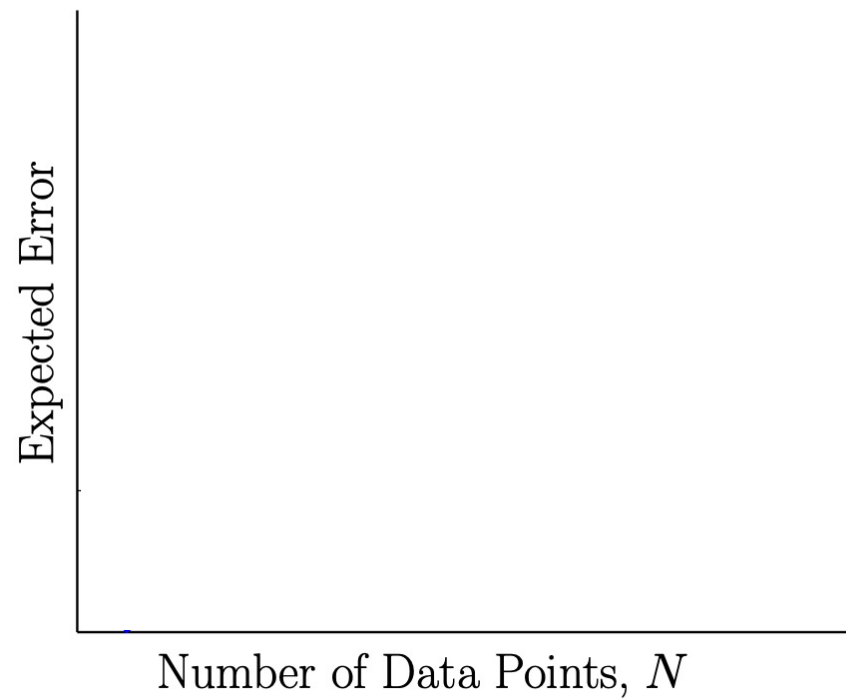


# Learning Curves

Simple Model

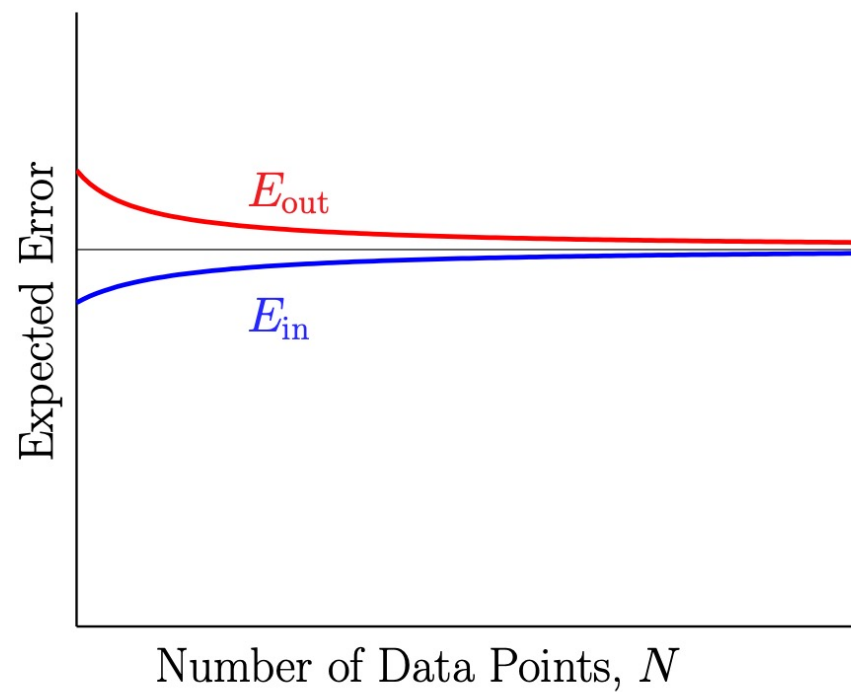


Complex Model

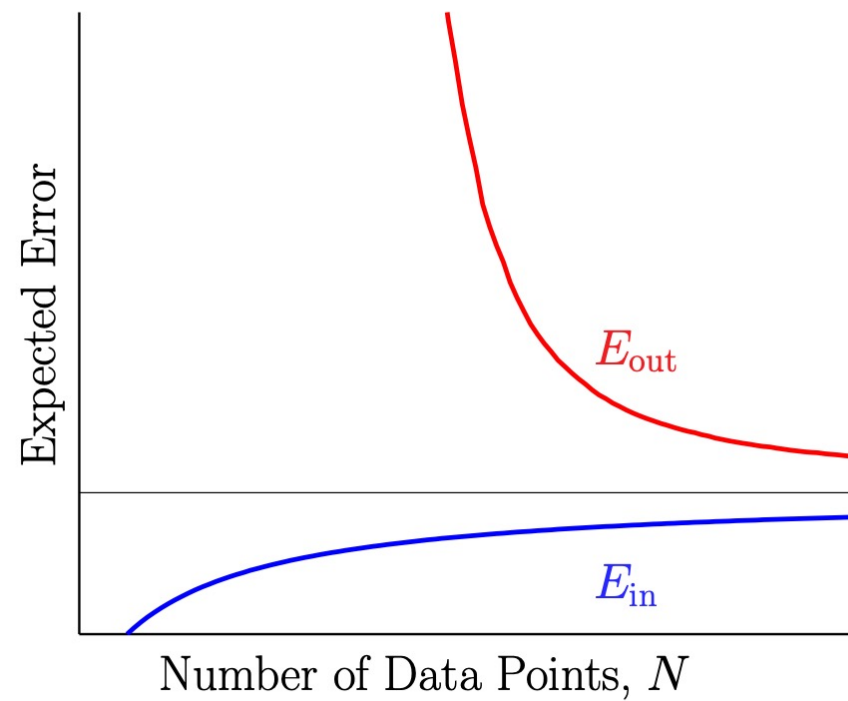


# Learning Curves

Simple Model

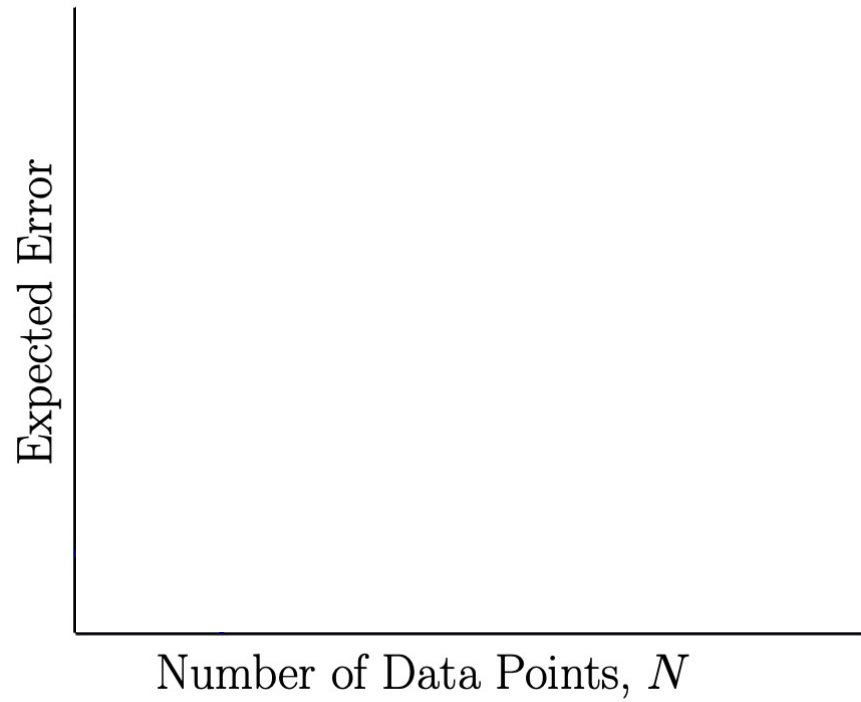


Complex Model

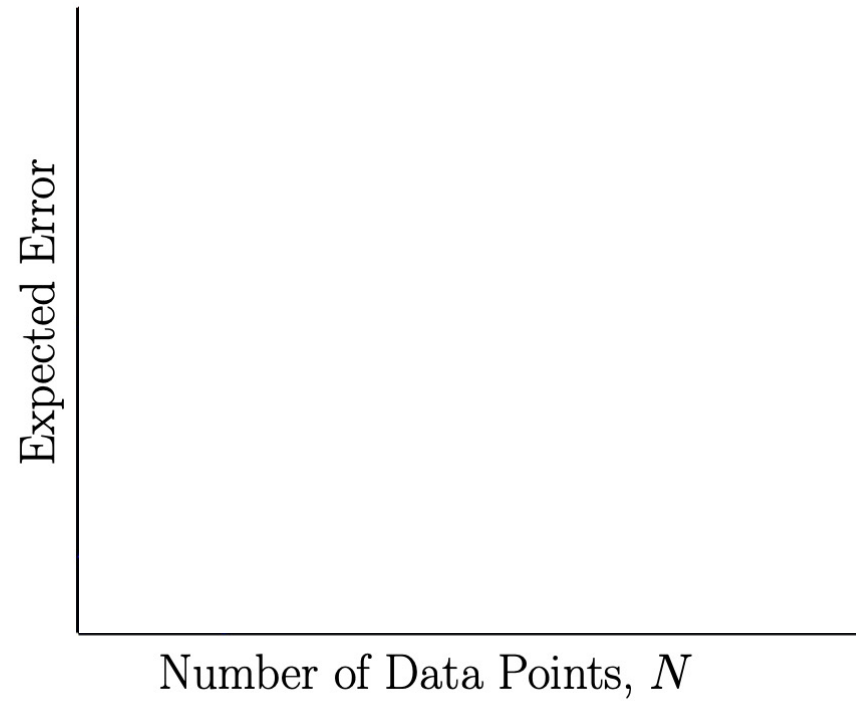


# Learning Curves

VC Analysis

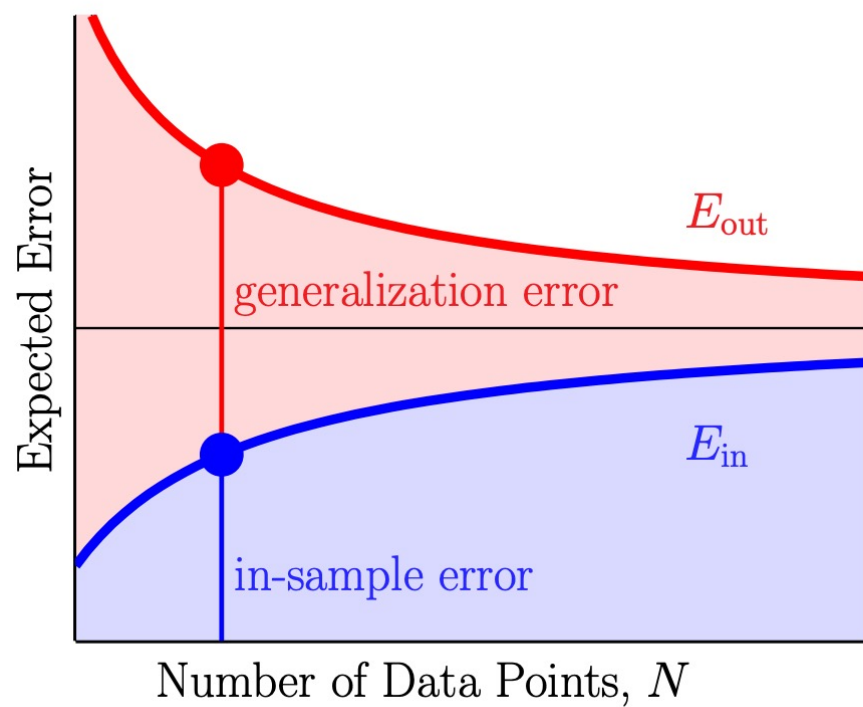


Bias-Variance Analysis

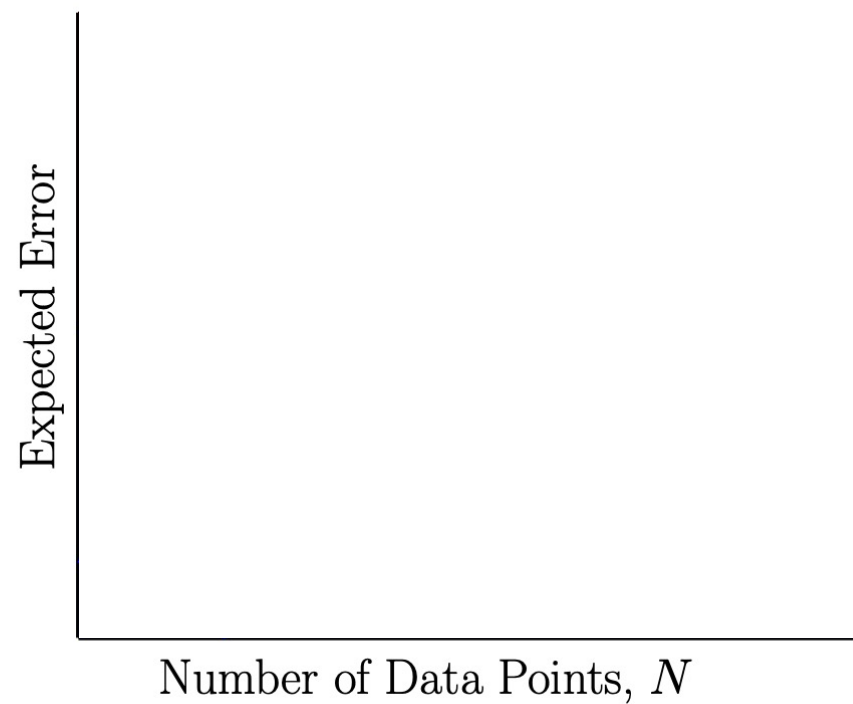


# Learning Curves

VC Analysis

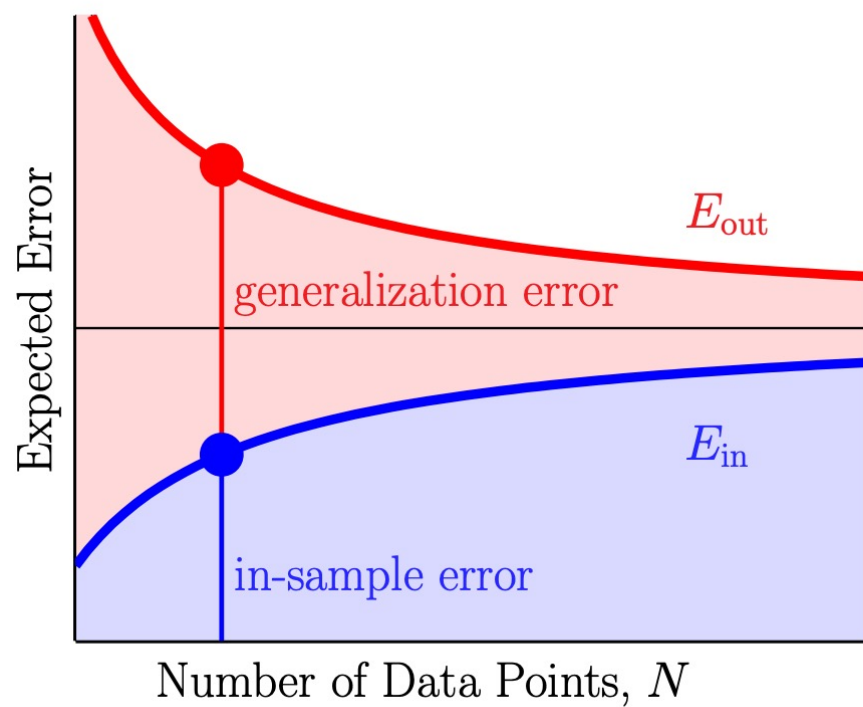


Bias-Variance Analysis

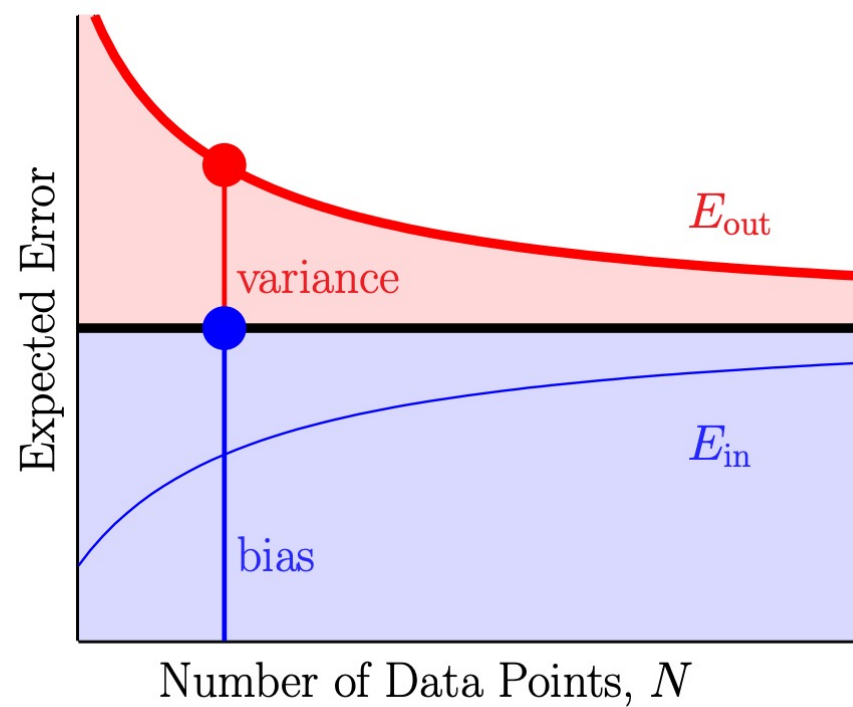


# Learning Curves

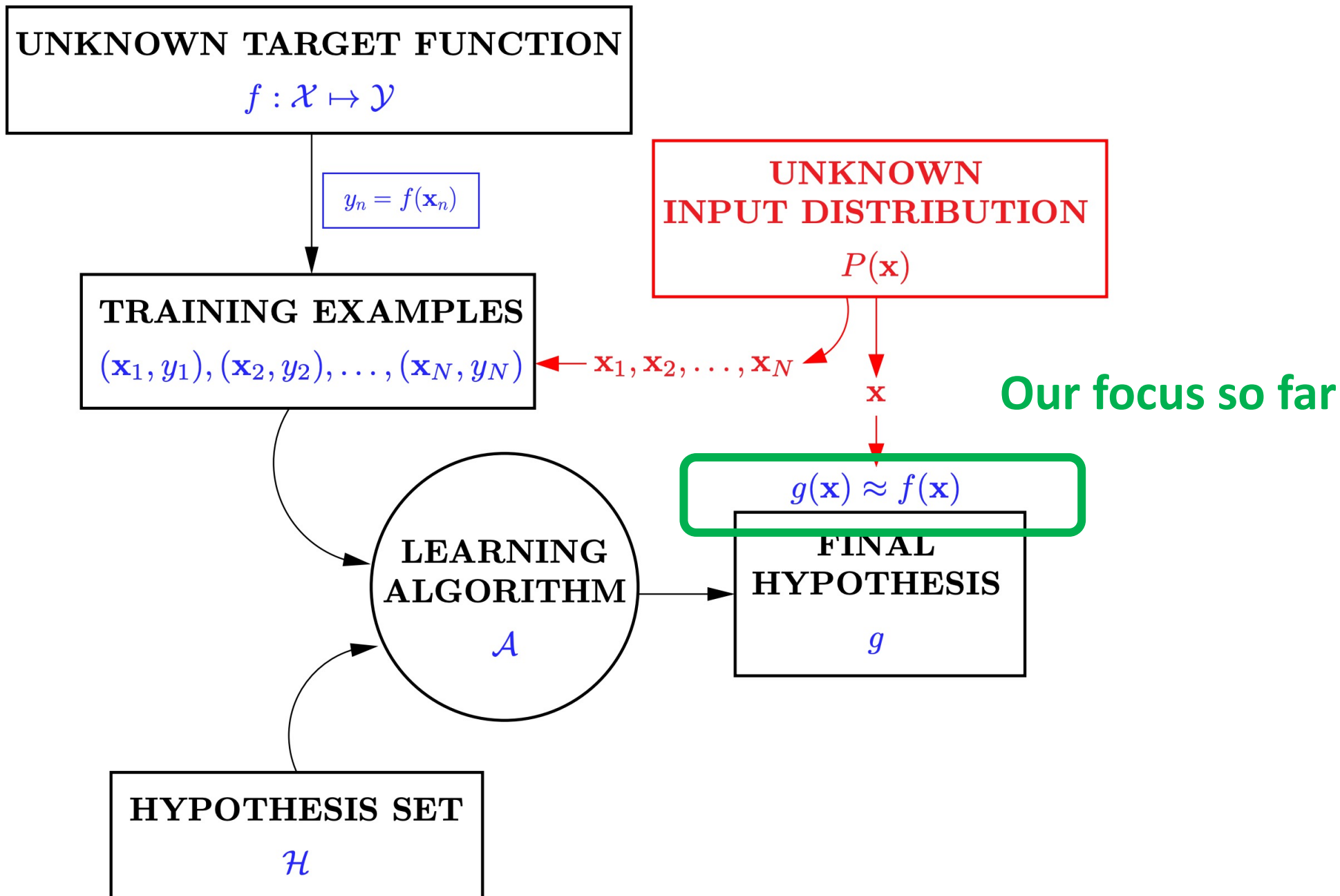
VC Analysis

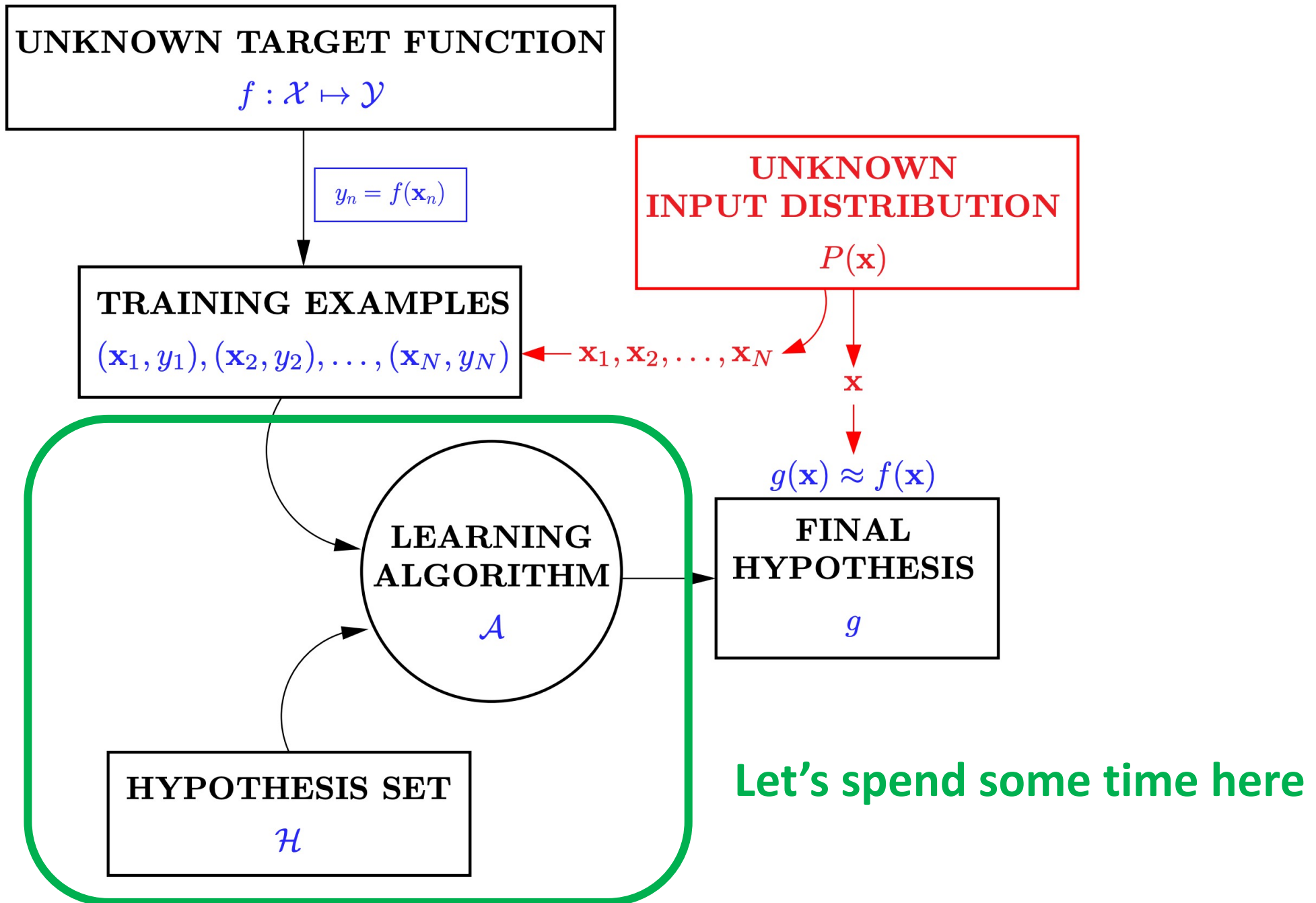


Bias-Variance Analysis









# Linear Models

# Linear Models

This is why it's called linear models

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

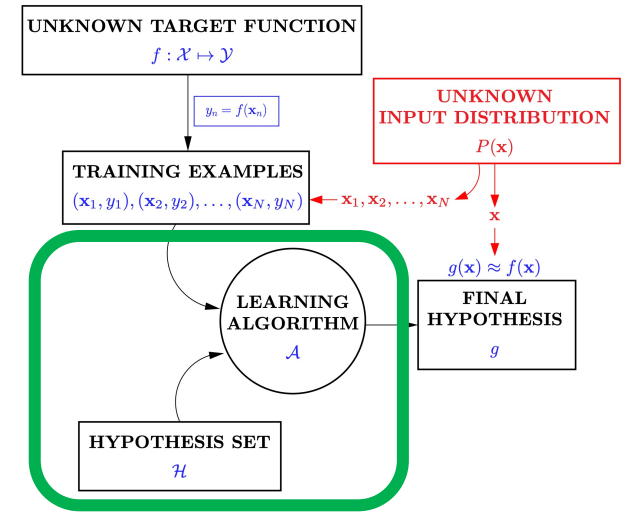
	Domain	Model	Credit Card Example
Linear Classification	$y \in \{-1, +1\}$	$H = \{h(\vec{x}) = \text{sign}(\vec{w}^T \vec{x})\}$	Approve or not
Linear Regression	$y \in \mathbb{R}$	$H = \{h(\vec{x}) = \vec{w}^T \vec{x}\}$	Credit line
Logistic Regression	$y \in [0,1]$	$H = \{h(\vec{x}) = \theta(\vec{w}^T \vec{x})\}$	Prob. of default

$$\theta(s) = \frac{e^s}{1 + e^s}$$

- Linear models:
  - Simple models => Good generalization error
- Reminder:
  - We will **interchangeably use**  $h$  and  $\vec{w}$  to represent a hypothesis in linear models

# Learning Algorithm?

- Goal of the algorithm: Find  $g \in H$  that minimizes  $E_{out}(g)$   
(We don't know  $E_{out}$ )
- Common algorithms:
  - $g = \operatorname{argmin}_{h \in H} E_{in}(h)$ 
    - Works well when the model is simple (generalization error is small)
    - Will focus on this in the discussion of linear models
  - $g = \operatorname{argmin}_{h \in H} \{E_{in}(h) + \Omega(h)\}$ 
    - $\Omega(h)$ : penalty for complex  $h$
    - Will discuss this when we get to LFD Section 4



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

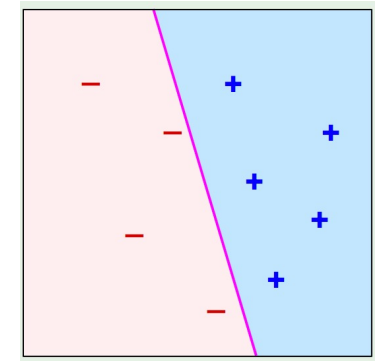
- **Optimization** is a key component in machine learning

# Linear Classification

# Linear Classification (Perceptron)

- Formulation

- Hypothesis set  $H = \{h(\vec{x}) = \text{sign}(\vec{w}^T \vec{x})\}$
- Error measure: binary error  $e(h(\vec{x}), y) = \mathbb{I}[h(\vec{x}) \neq y]$



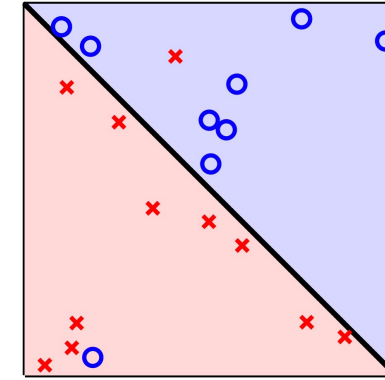
- Property

- Simple model (Fact: the VC dimension of d-dim perceptron is d+1)
- Good generalization error

- When data is linearly separable

- Run PLA
  - => find  $g$  with  $E_{in}(g) = 0$
  - =>  $E_{out}(g)$  is close to  $E_{in}(g) = 0$

# Non-Separable Data

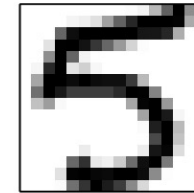
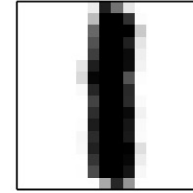


- Generally a hard problem
  - Minimizing  $E_{in}$  is a NP-hard problem
  - Reason: binary error is discrete and hard to optimize
- Alternative approaches
  - Pocket algorithm
    - Run PLA for a finite pre-determined  $T$  rounds
    - Keep track of the best weights  $\vec{w}^*$  ( $\vec{w}(t)$  that minimizes  $E_{in}$ )
  - Engineering the features to make data closer to be separable
    - Feature engineering (requiring domain knowledge, e.g., see LFD Example 3.1 and next slide)
  - Non-linear transformation (will discuss this in later lectures)
  - Changing the problem formulation (will discuss this in later lectures)
    - Example: Support vector machines in 2<sup>nd</sup> half of the semester



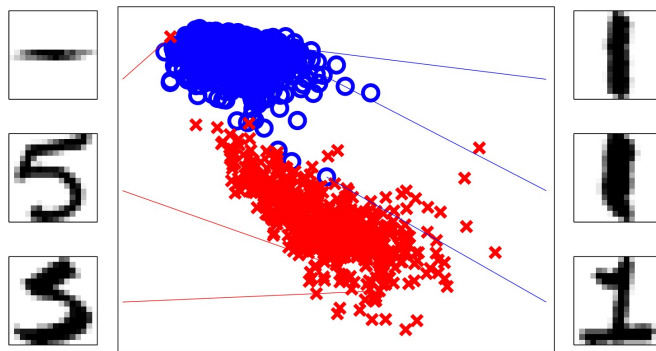
# Example on Feature Engineering

- Task: Classify handwritten digits of 1 and 5



- Linearly separable?

- What are the features  $\vec{x}$ ?
  - Each pixel as a feature (deep learning approach. requires data)
  - $\vec{x} = (\text{intensity}, \text{symmetry})$



Feature engineering is a practical issue in applied ML but not the focus of this course (requires domain knowledge).

# Linear Regression

	Domain	Model
Linear Classification	$y \in \{-1, +1\}$	$H = \{h(\vec{x}) = \text{sign}(\vec{w}^T \vec{x})\}$
Linear Regression	$y \in \mathbb{R}$	$H = \{h(\vec{x}) = \vec{w}^T \vec{x}\}$
Logistic Regression	$y \in [0,1]$	$H = \{h(\vec{x}) = \theta(\vec{w}^T \vec{x})\}$

# Linear Regression

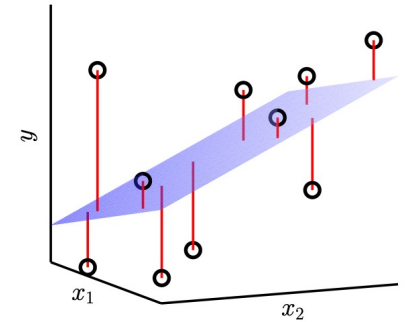
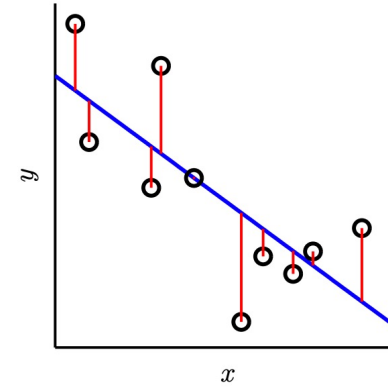
- Formulation

- Hypothesis set  $H = \{h(\vec{x}) = \vec{w}^T \vec{x}\}$
- Squared error  $e(h(\vec{x}), y) = (h(\vec{x}) - y)^2$

- Given dataset  $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$

- $E_{in}(\vec{w}) = \frac{1}{N} \sum_{n=1}^N (\vec{w}^T \vec{x}_n - y_n)^2$

- Goal: find  $\vec{w}_{lin} = \operatorname{argmin}_{\vec{w}} E_{in}(\vec{w})$



# Matrix Representation

- $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$

- $X = \begin{bmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{1,1} & \cdots & x_{1,d} \\ x_{2,0} & x_{2,1} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,0} & x_{N,1} & \cdots & x_{N,d} \end{bmatrix} \longrightarrow \boxed{x_{n,i}: \text{the } i\text{-th element of vector } \vec{x}_n}$

- $\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$

# Rewriting the In-Sample Error In Matrix Form

$$E_{in}(\vec{w}) = \frac{1}{N} \sum_{n=1}^N (\vec{w}^T \vec{x}_n - y_n)^2$$

$$X = \begin{bmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_N^T \end{bmatrix}; \quad \vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}$$

$$X\vec{w} = \begin{bmatrix} \vec{x}_1^T \vec{w} \\ \vdots \\ \vec{x}_N^T \vec{w} \end{bmatrix}$$

$$X\vec{w} - \vec{y} = \begin{bmatrix} \vec{x}_1^T \vec{w} - y_1 \\ \vdots \\ \vec{x}_N^T \vec{w} - y_N \end{bmatrix}$$

$$= \frac{1}{N} \sum_{n=1}^N (\vec{x}_n^T \vec{w} - y_n)^2$$

$$\|\vec{z}\| = \sqrt{\vec{z}^T \vec{z}} = \sqrt{\sum_{i=1}^d z_i^2}$$
$$\|\vec{z}\|^2 = \vec{z}^T \vec{z} = \sum_{i=1}^d z_i^2$$

$$= \frac{1}{N} \|X\vec{w} - \vec{y}\|^2$$

$$= \frac{1}{N} (X\vec{w} - \vec{y})^T (X\vec{w} - \vec{y})$$

$$\longrightarrow E_{in}(\vec{w}) = \frac{1}{N} \left( (X\vec{w})^T - \vec{y}^T \right) (X\vec{w} - \vec{y})$$
$$= \frac{1}{N} (\vec{w}^T X^T X \vec{w} - 2\vec{w}^T X^T \vec{y} + \vec{y}^T \vec{y})$$

How to find  $\vec{w}_{lin} = \operatorname{argmin}_{\vec{w}} E_{in}(\vec{w})$ ?

- Given  $E_{in}(\vec{w}) = \frac{1}{N} (\vec{w}^T X^T X \vec{w} - 2\vec{w}^T X^T \vec{y} + \vec{y}^T \vec{y})$
- Solve for  $\nabla_{\vec{w}} E_{in}(\vec{w}) = 0$ 
  - Think about what you'll do for one-dimensional case

- Derivations

- $E_{in}(\vec{w}) = \frac{1}{N} (\vec{w}^T X^T X \vec{w} - 2\vec{w}^T X^T \vec{y} + \vec{y}^T \vec{y})$

- $\nabla_{\vec{w}} E_{in}(\vec{w}) = \frac{1}{N} (2X^T X \vec{w} - 2X^T \vec{y})$

- $\nabla_{\vec{w}} E_{in}(\vec{w}_{lin}) = 0 \implies X^T X \vec{w}_{lin} = X^T \vec{y}$

$$\nabla f(\vec{w}) = \nabla_{\vec{w}} f(\vec{w}) = \begin{bmatrix} \frac{\partial}{\partial w_0} f(\vec{w}) \\ \frac{\partial}{\partial w_1} f(\vec{w}) \\ \vdots \\ \frac{\partial}{\partial w_d} f(\vec{w}) \end{bmatrix}$$

- $X^T X \vec{w}_{lin} = 2X^T \vec{y}$
  - Two cases:
    - If  $X^T X$  is **invertible** (When  $N \gg d$ , most of the time, it is invertible)
      - $\vec{w}_{lin} = (X^T X)^{-1} X^T \vec{y}$
    - If  $X^T X$  is not invertible
      - Requires special handling (See LFD Problem 3.15 for an example)
  - In practice
    - Define  $X^\dagger$  as the pseudo-inverse of  $X$ 
      - When  $X^T X$  is invertible,  $X^\dagger = (X^T X)^{-1} X^T$
      - When  $X^T X$  is not invertible, “handle” it appropriately (usually done in the library for you)
- Linear regression algorithm (a single step algorithm):
    - $\vec{w}_{lin} = X^\dagger \vec{y}$

# Linear Regression “Algorithm”

- Input:  $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

1. Construct  $X$  and  $\vec{y}$

2. Compute the pseudo-inverse of  $X$ :  $X^\dagger$   
( $X^\dagger = (X^T X)^{-1} X^T$  when  $(X^T X)$  is invertible)

3. Compute  $\vec{w}_{lin} = X^\dagger \vec{y}$

- Output:  $\vec{w}_{lin}$



# Practice

## Linear Regression “Algorithm”

- Input:  $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

1. Construct  $X$  and  $\vec{y}$

2. Compute the pseudo-inverse of  $X$ :  $X^\dagger$   
( $X^\dagger = (X^T X)^{-1} X^T$  when  $(X^T X)$  is invertible)

3. Compute  $\vec{w}_{lin} = X^\dagger \vec{y}$

- Output:  $\vec{w}_{lin}$

- What happens in 0-dimensional model
  - $\vec{x} = (x_0)$
  - Given  $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$
  - What's  $\vec{w}_{lin}$

# Discussion

## Linear Regression “Algorithm”

- Input:  $D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \dots, (\vec{x}_N, y_N)\}$

1. Construct  $X$  and  $\vec{y}$

2. Compute the pseudo-inverse of  $X$ :  $X^\dagger$   
( $X^\dagger = (X^T X)^{-1} X^T$  when  $(X^T X)$  is invertible)

3. Compute  $\vec{w}_{lin} = X^\dagger \vec{y}$

- Output:  $\vec{w}_{lin}$

- Special case of **zero-dimensional** space

$$X = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \Rightarrow X^T X = N \Rightarrow (X^T X)^{-1} = 1/N$$

$$\vec{w}_{lin} = (X^T X)^{-1} X^T \vec{y}$$

$$= \begin{bmatrix} \frac{1}{N} & \dots & \frac{1}{N} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \frac{1}{N} \sum_{n=1}^N y_n$$

Squared error  $\Rightarrow$  mean

# Discussion

- Linear regression generalizes very well
  - Under mild conditions (See LFD Exercise 3.4 for an example)

$$E_{out}(g) = E_{in}(g) + O\left(\frac{d}{N}\right)$$

- Use regression for classification
  - Note that  $\{-1, +1\} \subset \mathbb{R}$
  - Use linear regression to find  $\vec{w}_{lin} = (X^T X)^{-1} X^T \vec{y}$  for data with  $y \in \{-1, +1\}$
  - Use  $\vec{w}_{lin}$  for classification:  $g(\vec{x}) = \text{sign}(\vec{w}_{lin}^T \vec{x})$
  - Alternatively, use  $\vec{w}_{lin}$  as the initialization for Pocket Algorithm