CSE 417T
Introduction to Machine Learning

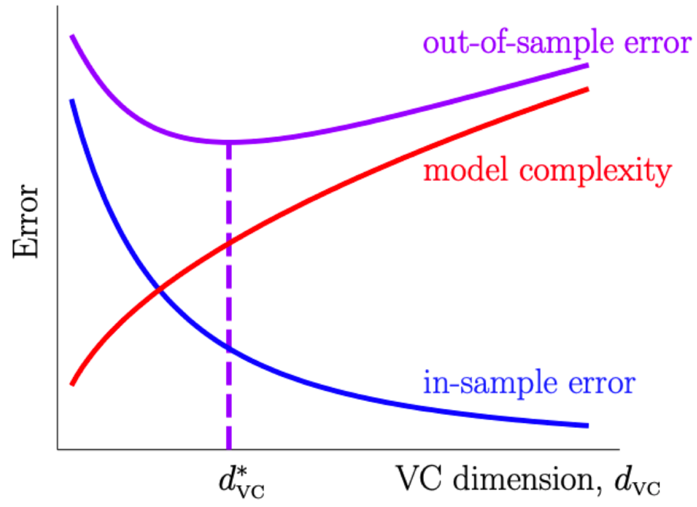
Lecture 7

Instructor: Chien-Ju (CJ) Ho

Recap

VC Generalization Bound

- VC Bound: $E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{vc}\frac{\ln N}{N}}\right)$
- The performance of your learning, i.e., $E_{out}(g)$, depends on
  - How well you fit your data ($E_{in}(g)$)
  - How well your $E_{in}(g)$ generalizes to $E_{out}(g)$



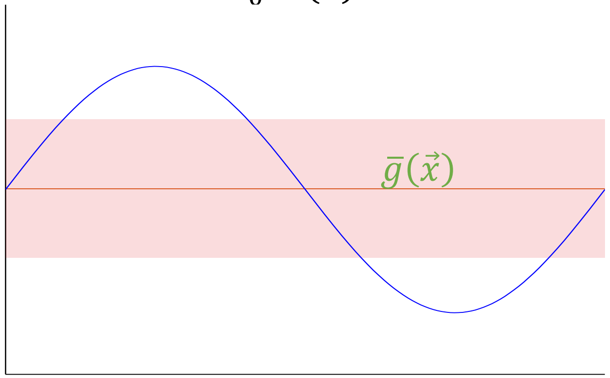
Bias-Variance Decomposition

Bias($\vec{x}$) Var($\vec{x}$)

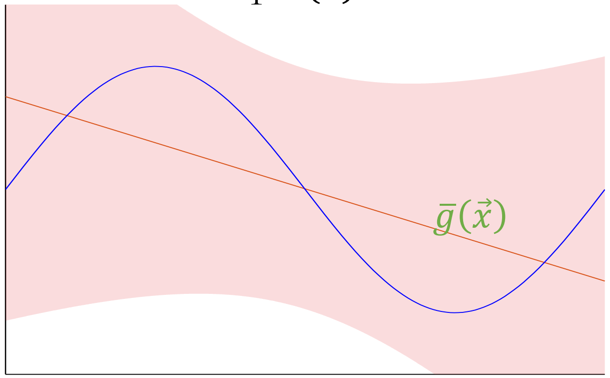
- $\mathbb{E}_D\left[E_{out}\left(g^{(D)}\right)\right] = \mathbb{E}_{\vec{x}}\left[\left(\bar{g}(\vec{x}) - f(\vec{x})\right)^2\right] + \mathbb{E}_{\vec{x}}\left[\mathbb{E}_D\left[\left(g^{(D)}(\vec{x}) - \bar{g}(\vec{x})\right)^2\right]\right]$

- The performance of your learning, i.e., $\mathbb{E}_D\left[E_{out}\left(g^{(D)}\right)\right]$, depends on
  - How well you can fit your data using your hypothesis set (bias)
  - How close to the best fit you can get for a given dataset (variance)

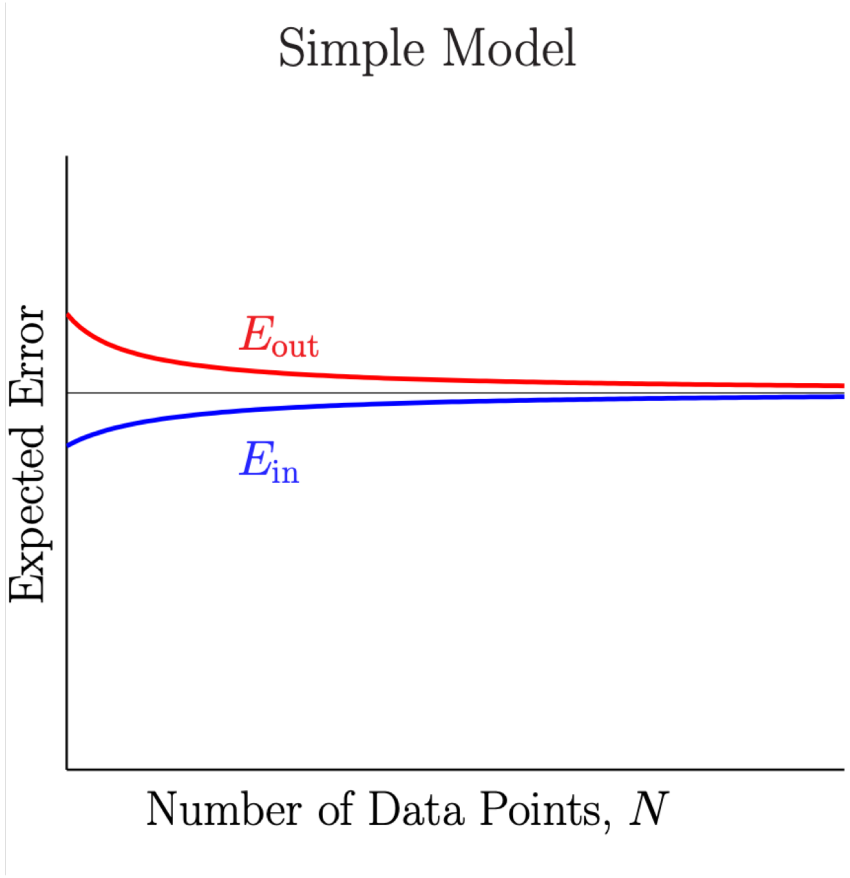
- $H_0: h(x) = b$

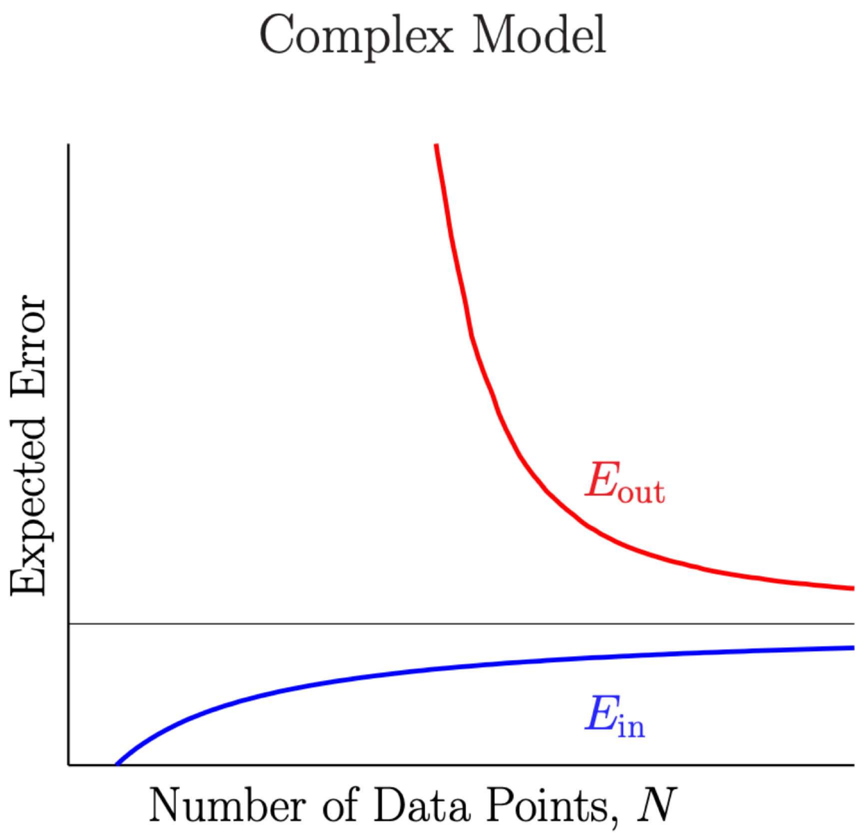


- $H_1: h(x) = ax + b$

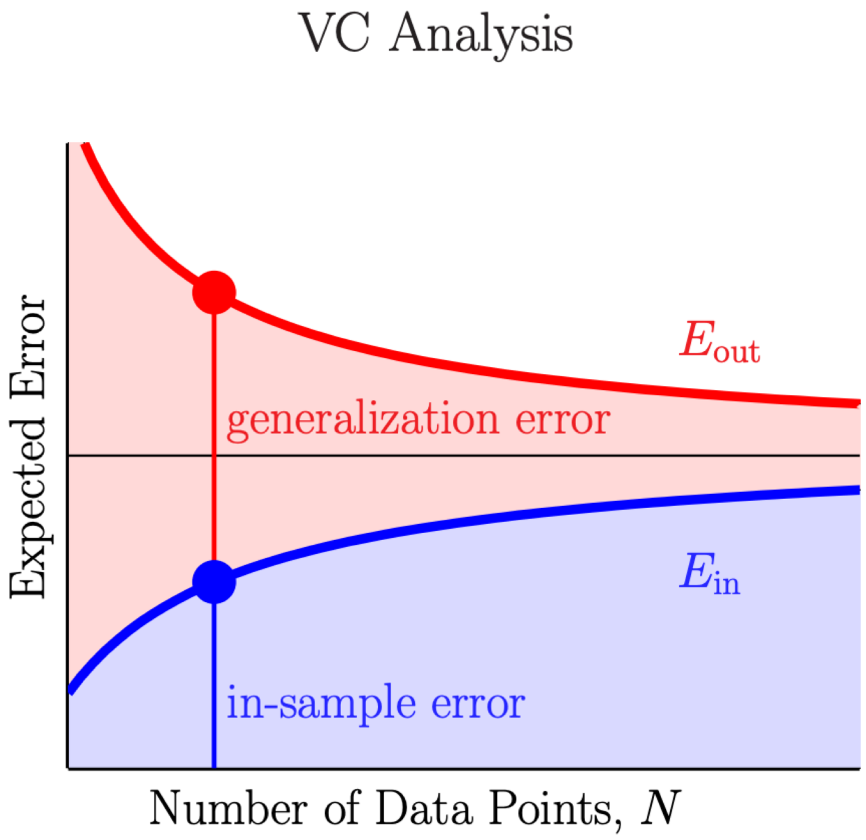


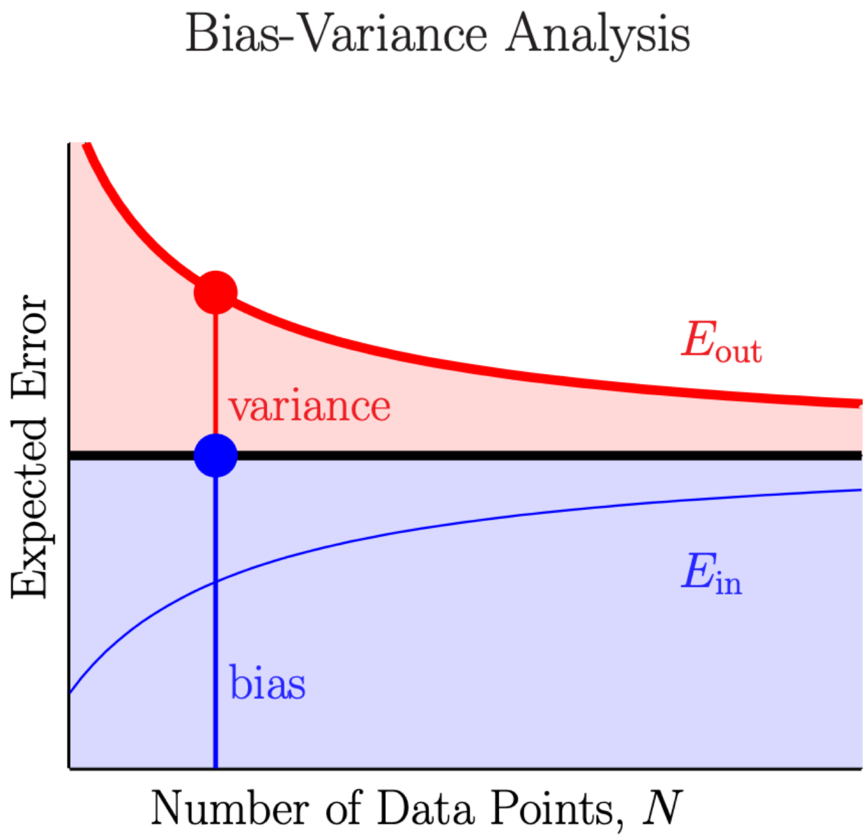
Learning Curves





Learning Curves

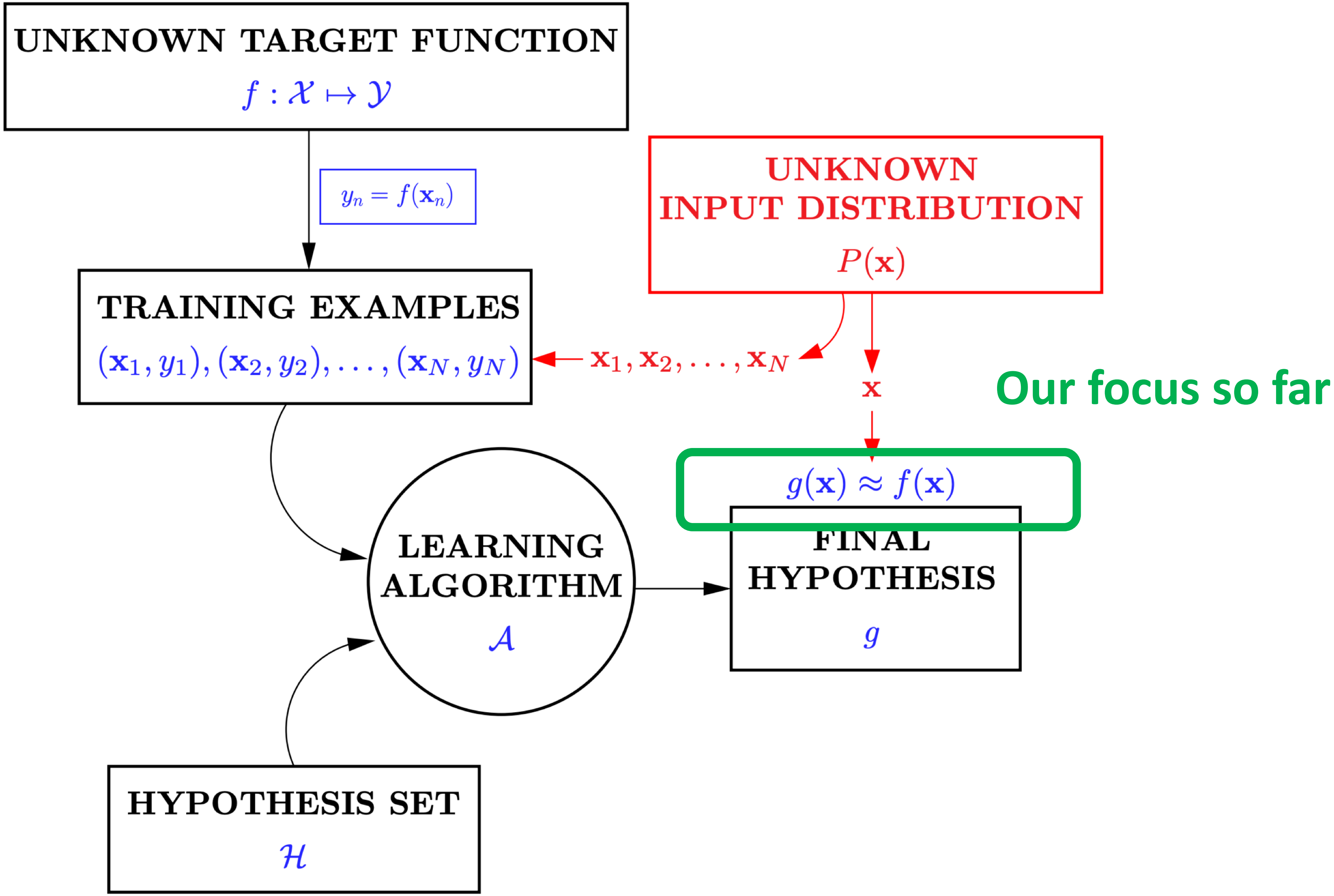


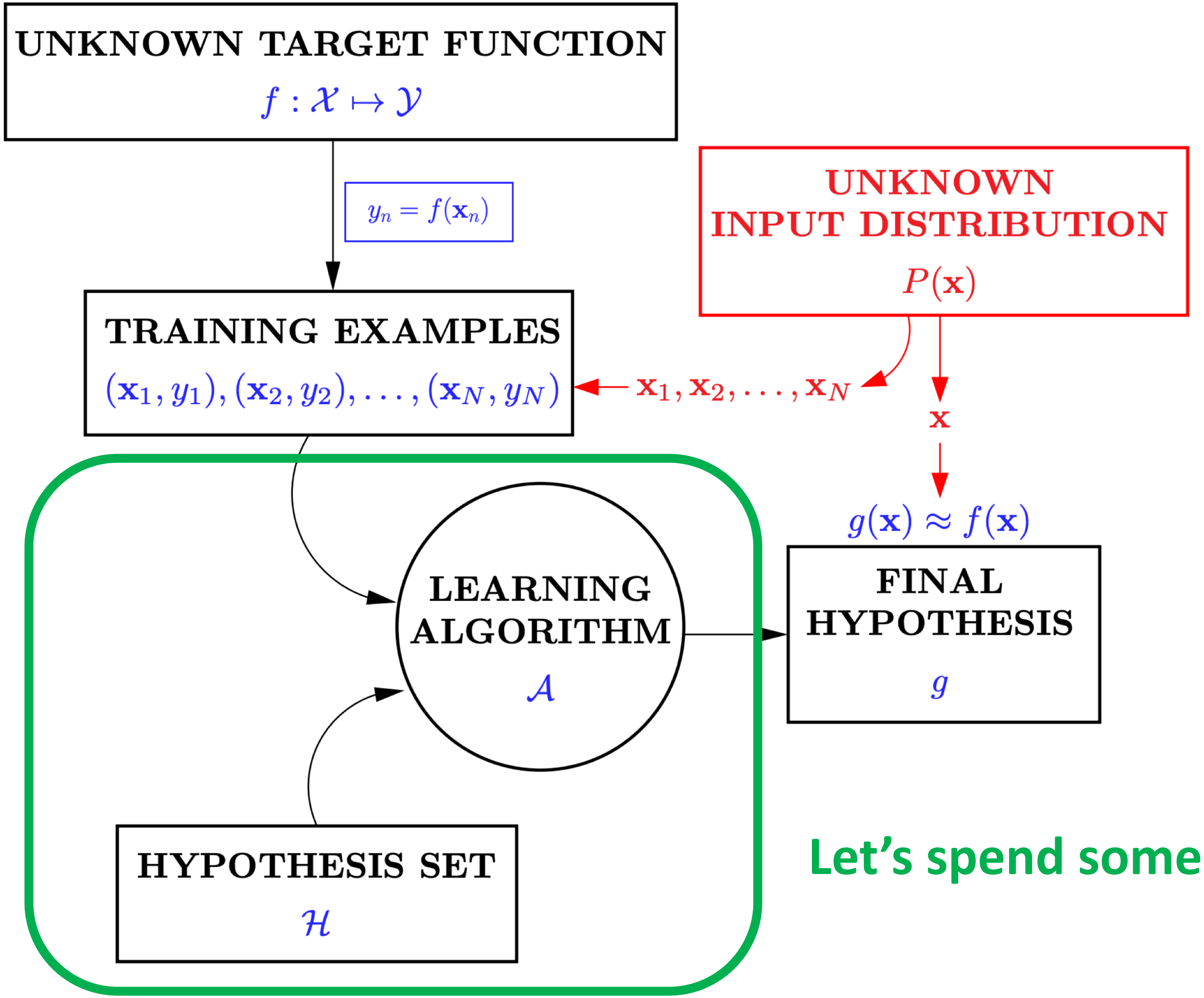


Brief Lecture Notes Today

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook.

Let me know if you spot errors.


UNKNOWN TARGET FUNCTION
$f : \mathcal{X} \mapsto \mathcal{Y}$
$y_n = f(\mathbf{x}_n)$
UNKNOWN INPUT DISTRIBUTION
$P(\mathbf{x})$
TRAINING EXAMPLES
$(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$
$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$
$\mathbf{x}$
Our focus so far
$g(\mathbf{x}) \approx f(\mathbf{x})$
LEARNING ALGORITHM
$\mathcal{A}$
FINAL HYPOTHESIS
g
HYPOTHESIS SET
$\mathcal{H}$



Let’s spend some time here

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

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

- Linear models:
  - Simple models => Good generalization error

- Reminder:
  - We will interchangeably use h and $\vec{w}$ to represent a hypothesis in linear models

Algorithms?

- Goal of the algorithm:
  - Find $g \in H$ such that $g \approx f$
  - Define error measures to quantify $g \approx f$
  - Find $g \in H$ that minimizes $E_{out}(g)$

- Recall on the error measure
  - Often focus on point-wise error $e\left(h(\vec{x})\right), f(\vec{x}))$
    - Binary error for classification
    - Squared error for regression
  - $E_{in}(h) = \frac{1}{N}\sum_{n=1}^{N} e(h(\vec{x}_n), f(\vec{x}_n))$
  - $E_{out}(h) = \mathbb{E}_{\vec{x}}[e(h(\vec{x}), f(\vec{x})]$

Algorithms?

- Goal of the algorithm: Find $g \in H$ that minimizes $E_{out}(g)$
- VC Bound: $E_{out}(g) \leq E_{in}(g) + O\left(\sqrt{d_{vc}\frac{\ln N}{N}}\right)$
- Common algorithms:
  - $g = argmin_{h \in H} E_{in}(h)$
    - Works well when the model is simple (generalization error is small)
    - Will focus on this in the discussion of linear models
  - $g = argmin_{h \in H}\{E_{in}(h) + \Omega(h)\}$
    - $\Omega(h)$: penalty for complex h
    - Will discuss this when we get to LFD Section 4
- Optimization is a key component in machine learning

Linear Classification

Linear Classification

- Formulation
  - Hypothesis set $H = \{h(\vec{x}) = sign(\vec{w}^T\vec{x})\}$
  - Error measure: binary error $e(h(\vec{x}), y) = \mathbb{I}[h(\vec{x}) \neq y]$
- Property
  - Simple model (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 in general
  - Reason: binary error is discrete and hard to optimize

- Alternative approaches
  - Changing the problem formulation (will discuss in later lectures)
    - Example: Support vector machines in 2^{nd} half of the semester
  - Engineering the features to make data closer to be separable
    - the handwriting digit recognition example
  - Pocket algorithm
    - Run PLA for T rounds
    - Keep track of the best weights $\vec{w}^*$ ($\vec{w}(t)$ that minimizes E_{in})

Linear Regression

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 $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$
  - $E_{in}(\vec{w}) = \frac{1}{N} \sum_{n=1}^{N} (\vec{x}_n - y_n)^2$

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

Matrix Representation

- $D = \{(\vec{x}_1, y_1), \dots, (\vec{x}_N, y_N)\}$
- $X = \begin{bmatrix} \vec{x}_1^T \\ \vdots \\ \vec{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{1,0} & x_{1,1} & \cdots & x_{1,d} \\ x_{2,0} & x_{2,1} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{2,0} & x_{N,1} & \cdots & x_{N,d} \end{bmatrix}$ and $\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

$x_{n,i}$: the i-th element of vector $\vec{x}_n$

Rewriting the In-Sample Error In Matrix Form

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

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

$$= \frac{1}{N}\sum_{n=1}^{N}\left(\vec{x_n}^T\vec{w} - y_n\right)^2$$

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

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

$$\longrightarrow$$

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

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

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

- Answer: Solve for $\nabla_{\vec{w}} E_{in}(\vec{w}) = 0$

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

  - $\nabla_{\vec{w}} E_{in}(\vec{w}_{lin}) = 0$ ==> $X^T X \vec{w}_{lin} = 2X^T \vec{y}$

- $X^T X \vec{w}_{lin} = 2X^T \vec{y}$

- 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):
  - $\vec{w}_{lin} = X^{\dagger} \vec{y}$

Discussion

- Special case of zero–dimensional space

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

$$\vec{w}_{lin} = (X^T X)^{-1} X^T \vec{\mathrm{y}} = \begin{bmatrix} \frac{1}{N} \dots \frac{1}{N} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \frac{1}{N} \sum_{n=1}^{N} y_n$$

Squared error => 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: $g(\vec{x}) = \mathrm{sign}\left(\vec{w}_{lin}^T \vec{x}\right)$

  - Alternatively, use $\vec{w}_{lin}$ as the initialization for Pocket Algorithm