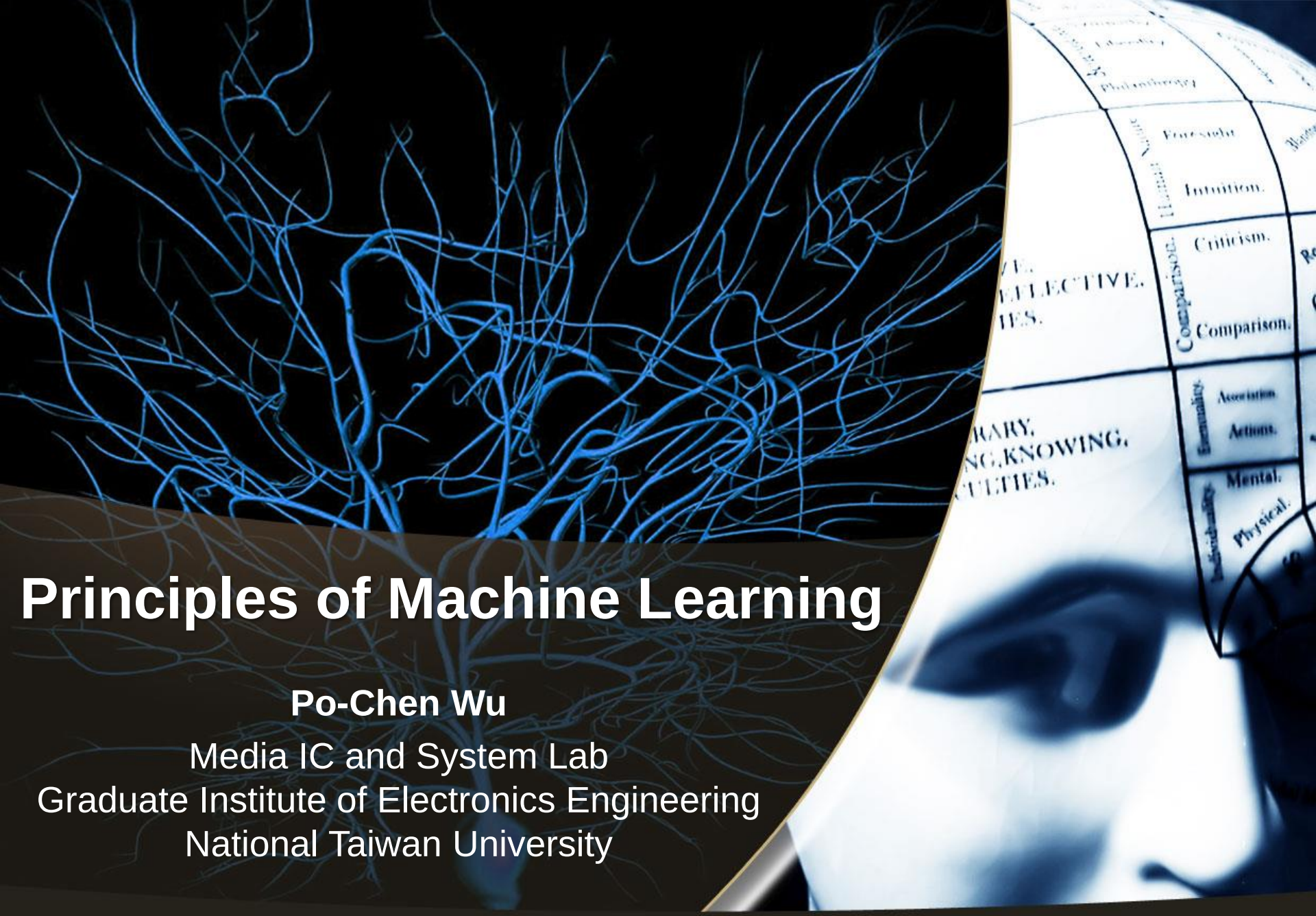


Principles of Machine Learning

Po-Chen Wu

Media IC and System Lab
Graduate Institute of Electronics Engineering
National Taiwan University

Outline

- Introduction to Machine Learning
- Theory of Generalization
- Learning Algorithm
- Hazard of Overfitting
- Blending and Bagging

Outline

- Introduction to Machine Learning
- Theory of Generalization
- Learning Algorithm
- Hazard of Overfitting
- Blending and Bagging

Mars One Project

- A one-way ticket to Mars.
- There is a total of 2,782 applicants.
- The application consists of applicant's
  - General information
  - Motivational letter
  - Résumé
  - Video

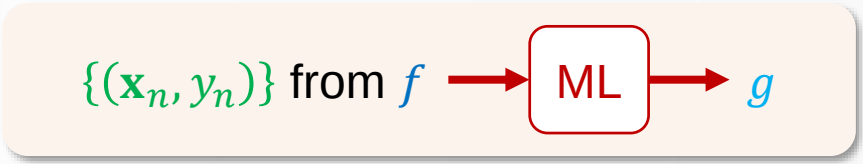
Admission Ticket Approval

Applicant Information	
Age	37 years
Gender	Male
Occupation	Professor
Annual Salary	NTD 2,000,000
Year in Job	11 Years
Current Debt	NTD 110,000

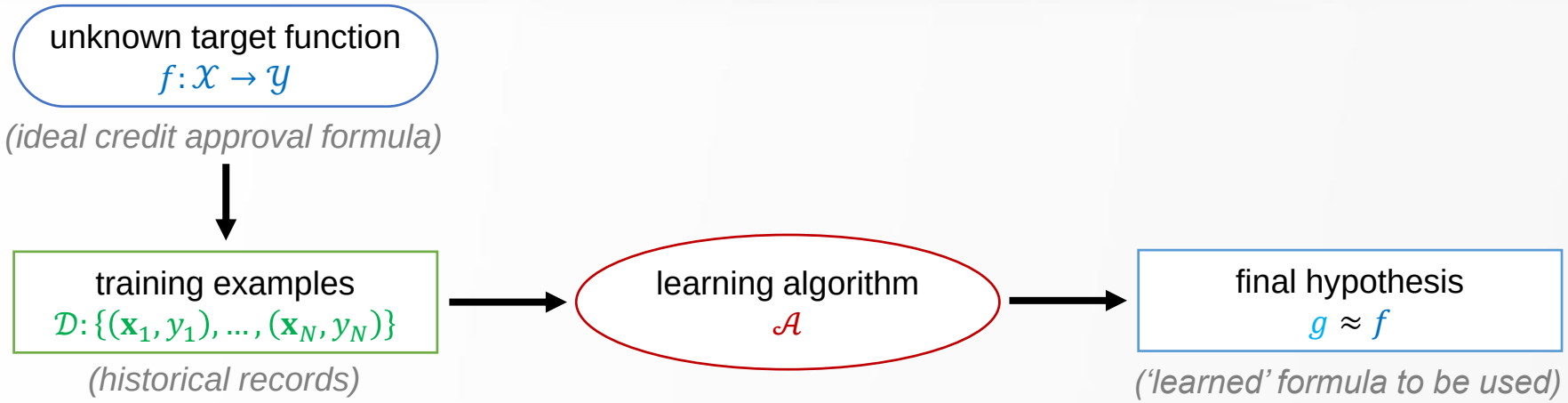
- Unknown target function to be learned:
  - "Should we approve the admission ticket or not?"

Formalize the Learning Problem

- Basic Notations
  - input: $\mathbf{x} \in \mathcal{X}$ (volunteer application)
  - output: $y \in \mathcal{Y}$ (good/bad after approving ticket)
  - unknown target function to be learned
    $f: \mathcal{X} \to \mathcal{Y}$ (ideal ticket approval formula)
  - data $\Leftrightarrow$ training examples:
    $\mathcal{D}: \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ (historical records)
  - hypothesis $\Leftrightarrow$ skill with hopefully good performance:
    $g: \mathcal{X} \to \mathcal{Y}$ ('learned' ticket approval formula)



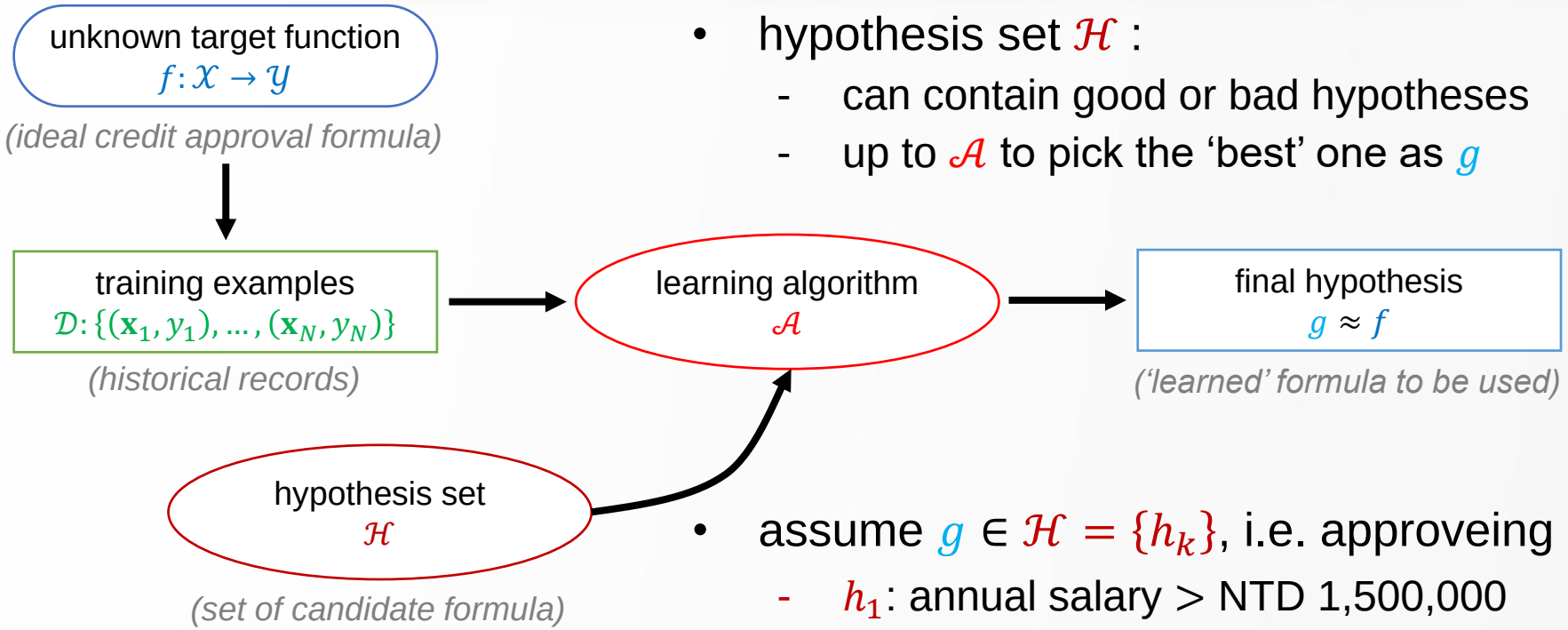
Learning Flow for Ticker Approval



- target f unknown
  (i.e. no programmable definition)
- hypothesis g hopefully $\approx f$ but possibly different from f
  (perfection 'impossible when f unknown'

What does g look like?

The Learning Model

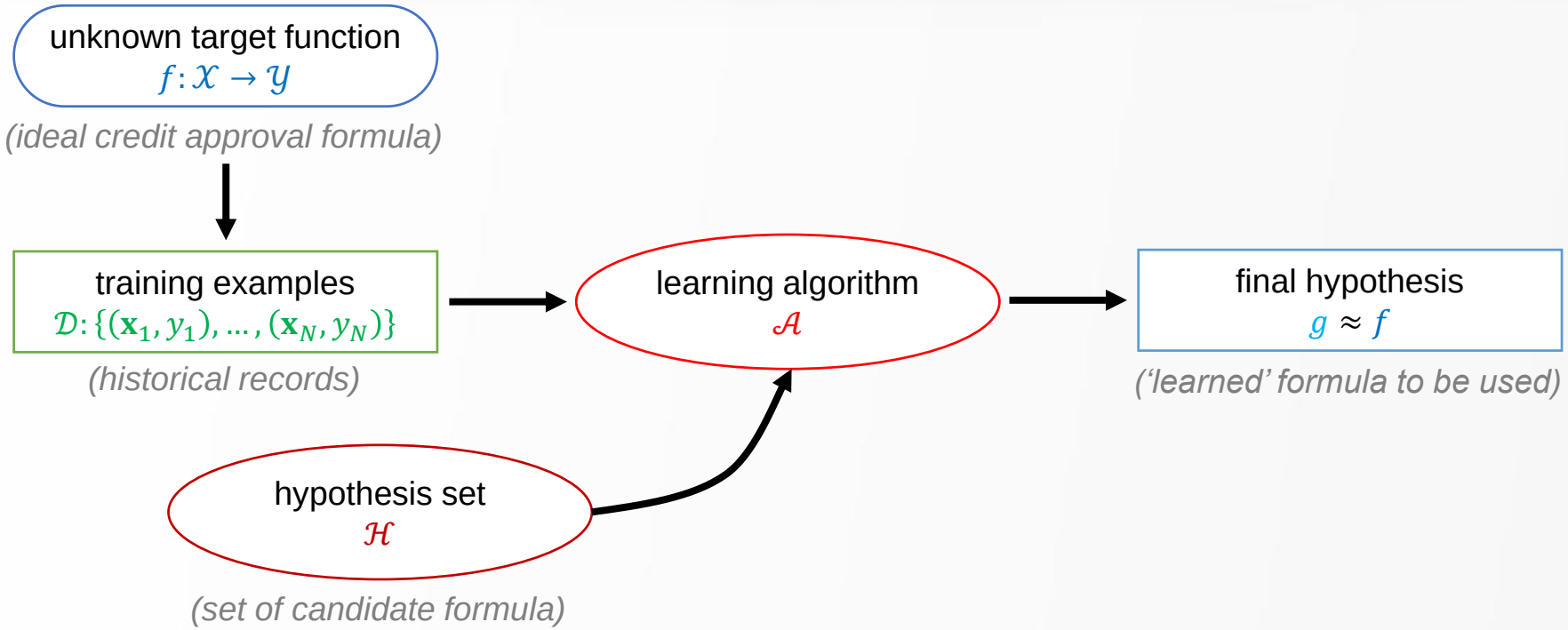


- hypothesis set $\mathcal{H}$:
  - can contain good or bad hypotheses
  - up to $\mathcal{A}$ to pick the 'best' one as g

- assume $g \in \mathcal{H} = \{h_k\}$, i.e. approveing if
  - h_1: annual salary > NTD 1,500,000
  - h_2: debt > NTD 50,000 (really?)
  - h_3: year in job ≥ 5

learning model = $\mathcal{A}$ and $\mathcal{H}$

Practical Definition of Machine Learning

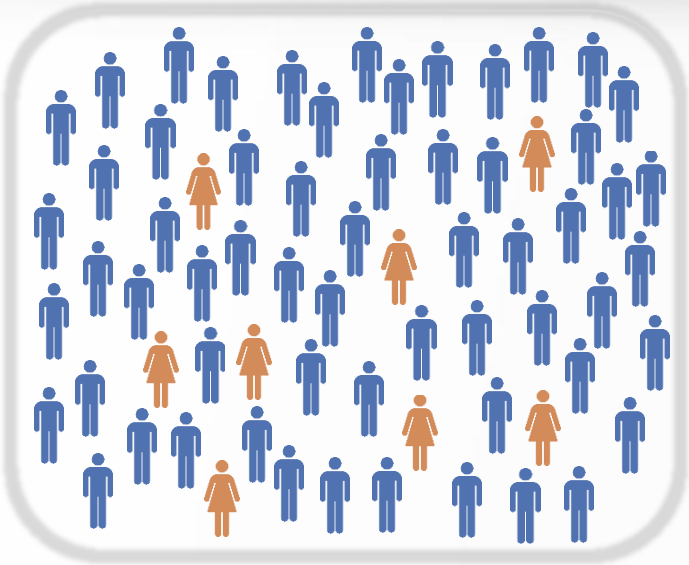


Machine Learning:
use data to compute hypothesis g that approximates target f

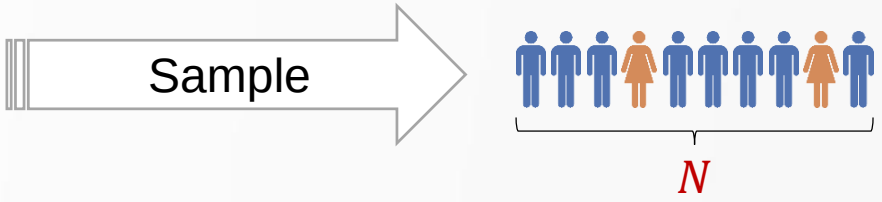
Outline

- Introduction to Machine Learning
- Theory of Generalization
- Learning Algorithm
- Hazard of Overfitting
- Blending and Bagging

Sex Ratio of EE Students



Population



- Population
  - girl proportion $= \mu$
  - boy proportion $= 1 - \mu$

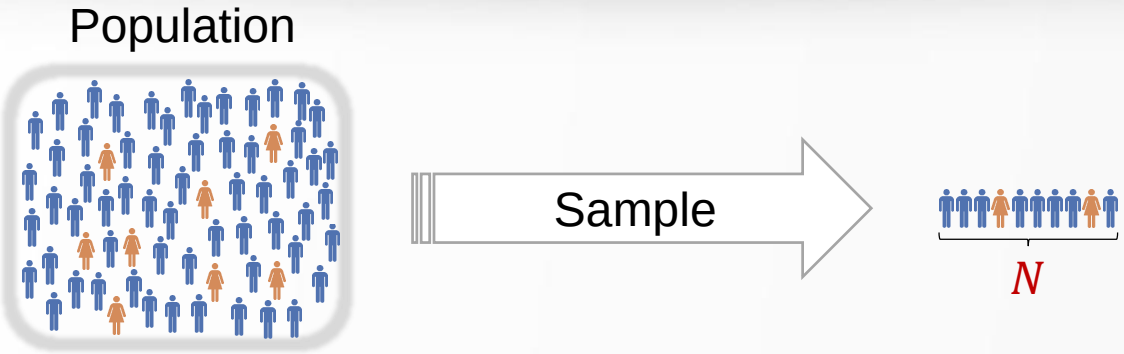
μ: unknown

- Sample
  - girl fraction $= \nu$
  - boy fraction $= 1 - \nu$

ν: known

Does in-sample ν say anything about out-of-sample μ?

Hoeffding's Inequality



μ: girl proportion in population

ν: girl fraction in sample

- In big sample (N large), ν is probably close to μ (within ϵ)

$$\mathbb{P}[|\nu - \mu| > \epsilon] \leq 2\exp(-2\epsilon^2 N)$$ Hoeffding's Inequaltiy

the statement '$\nu = \mu$' is
probably approximately correct (PAC)

Connection to Learning

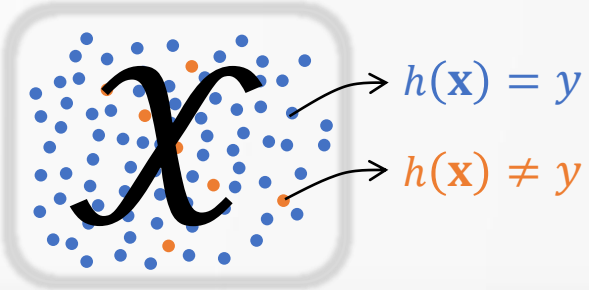
EE

- unknown girl prob. μ
- student  $\in$ EE
- boy 
- girl 
- size-N sample from EE of i.i.d. students.

Learning

- fixed hypothesis $h(\mathbf{x}) \overset{?}{=} f(\mathbf{x})$
- $\mathbf{x} \in \mathcal{X}$
- h is right $\Leftrightarrow h(\mathbf{x}) = f(\mathbf{x})$
- h is wrong $\Leftrightarrow h(\mathbf{x}) \neq f(\mathbf{x})$
- check h on $\mathcal{D}$: $\{(\mathbf{x}_n, y_n)\}$ with i.i.d. $\mathbf{x}_n$

if large N & i.i.d. $\mathbf{x}_n$, can probably infer unknown $[\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$ probability by known $[\![h(\mathbf{x}) \neq y]\!]$ fraction



Error Measure

- Classification error $[\![\dots]\!]$
  - Often also called '0/1 error'
  - $[\![\text{true}]\!] = 1, [\![\text{false}]\!] = 0$

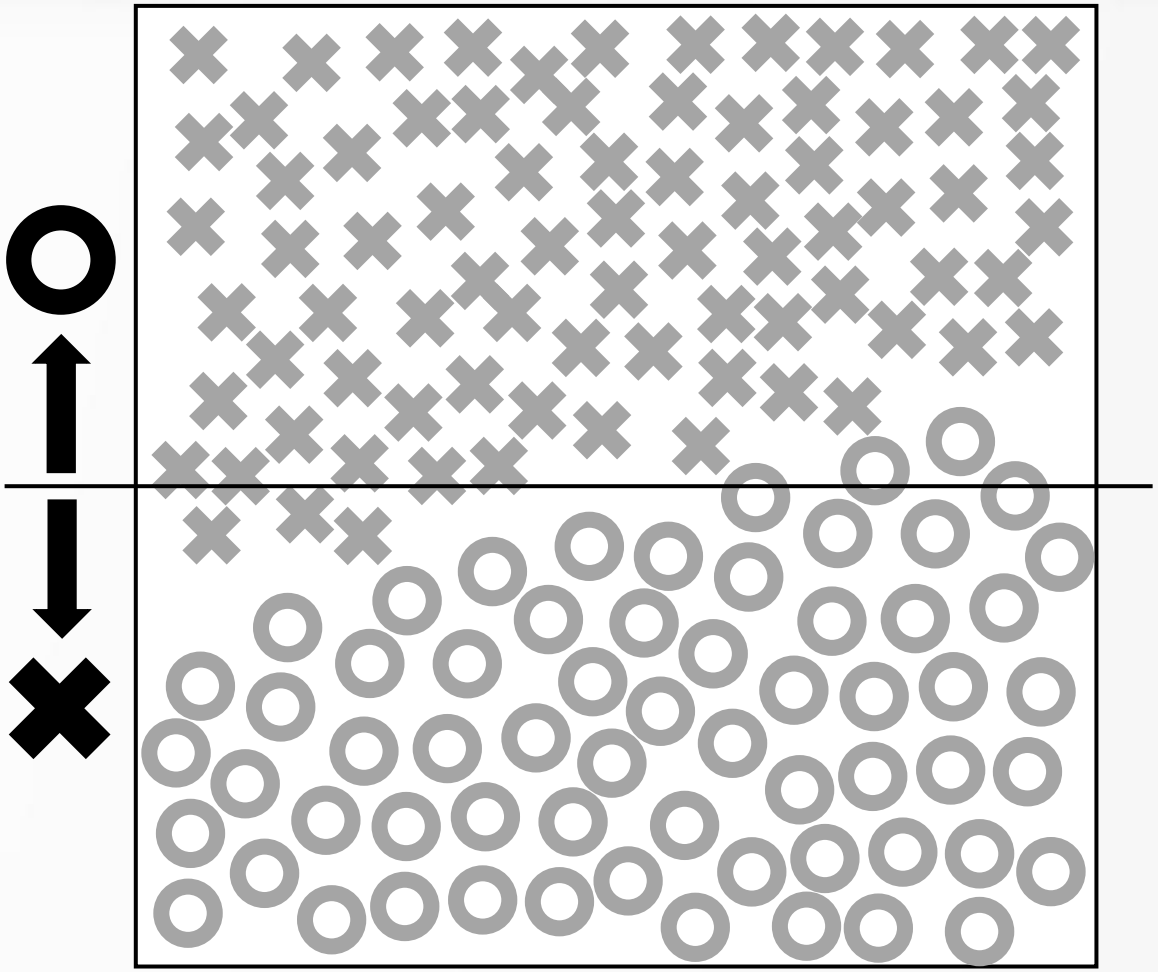
In-sample Error	Out-of-sample Error
$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N} [\![h(\mathbf{x}_n) \neq y_n]\!]$	$E_{\text{out}}(h) = \underset{\mathbf{x}\sim P}{\mathcal{E}} [\![h(\mathbf{x}) \neq f(\mathbf{x})]\!]$

$\mathcal{E}$: expectation value
$\mathbf{x}\sim P$, means the random variable $\mathbf{x}$ has the probability distribution P

Find a Separation Line

h is right $\Leftrightarrow h(\mathbf{x}) = f(\mathbf{x})$
h is wrong $\Leftrightarrow h(\mathbf{x}) \neq f(\mathbf{x})$

Classifier



$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}[\![h(\mathbf{x}_n) \neq y_n]\!]$$

The Formal Guarantee

- For any fixed h, in 'big' data (N large), in-sample error $E_{\text{in}}(h)$ is probably close to out-of-sample error $E_{\text{out}}(h)$ (within ϵ)

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

$$\mathbb{P}[|\nu - \mu| > \epsilon] \leq 2\exp(-2\epsilon^2 N)$$

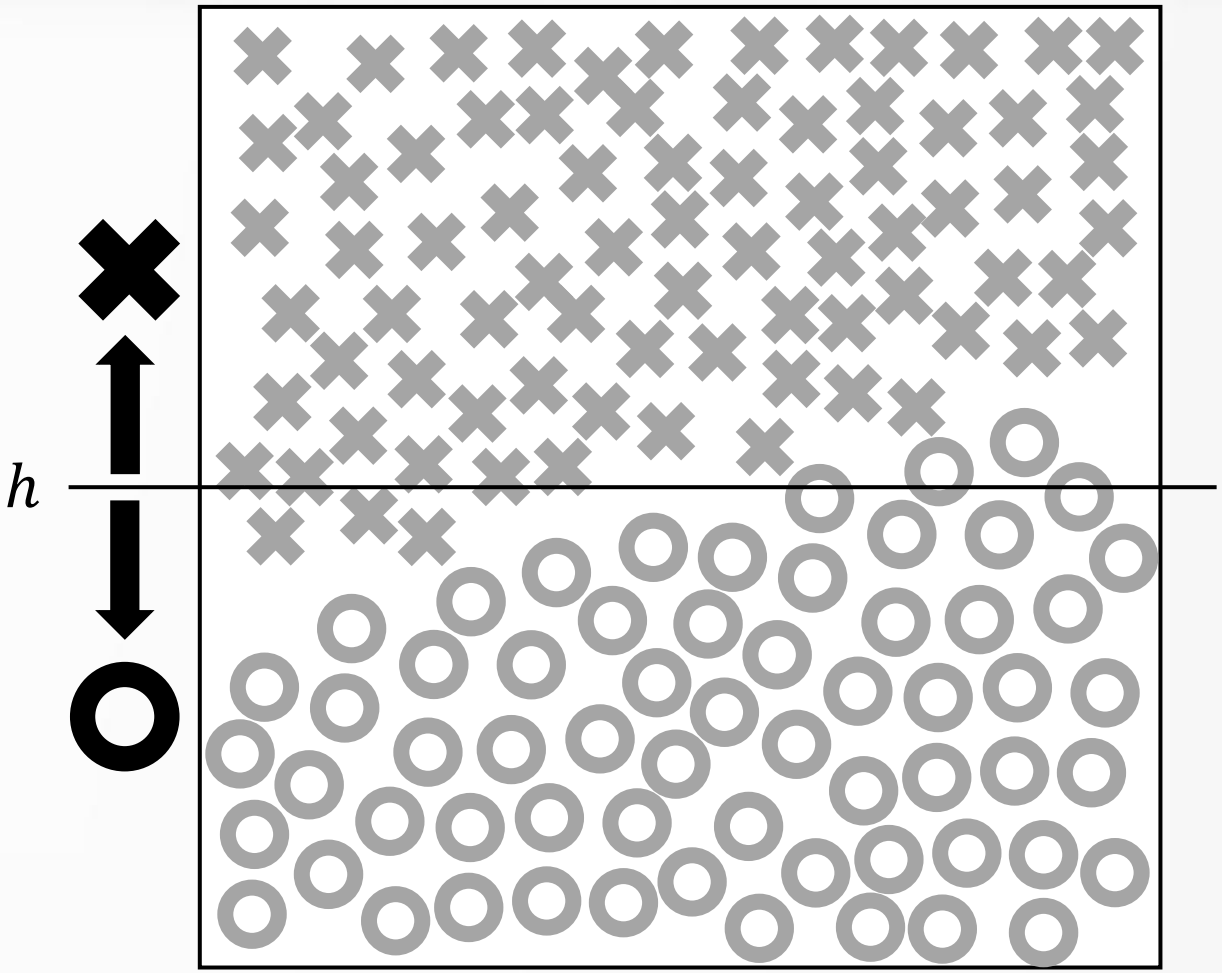
the statement '$E_{\text{in}}(h) = E_{\text{out}}(h)$' is probably approximately correct (PAC)

the statement '$\nu = \mu$' is probably approximately correct (PAC)

If '$E_{\text{in}}(h) \approx E_{\text{out}}(h)$' and '$E_{\text{in}}(h)$ small' $\Longrightarrow E_{\text{out}}(h)$ small $\Longrightarrow h \approx f$ with respect to P

Find a Separation Line

h is right $\Leftrightarrow h(\mathbf{x}) = f(\mathbf{x})$
h is wrong $\Leftrightarrow h(\mathbf{x}) \neq f(\mathbf{x})$



$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}[\![h(\mathbf{x}_n) \neq y_n]\!]$$

From Fixed h to Set $\mathcal{H}$

- Vapnik-Chervonenkis (VC) bound:

$$\mathbb{P}[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon] \leq 4\underbrace{m_{\mathcal{H}}(2N)}\exp\left(-\frac{1}{8}\epsilon^2 N\right)$$

Model Complexity $O\left(N^{d_{VC}}\right)$

d_{VC}: VC dimension of $\mathcal{H}$

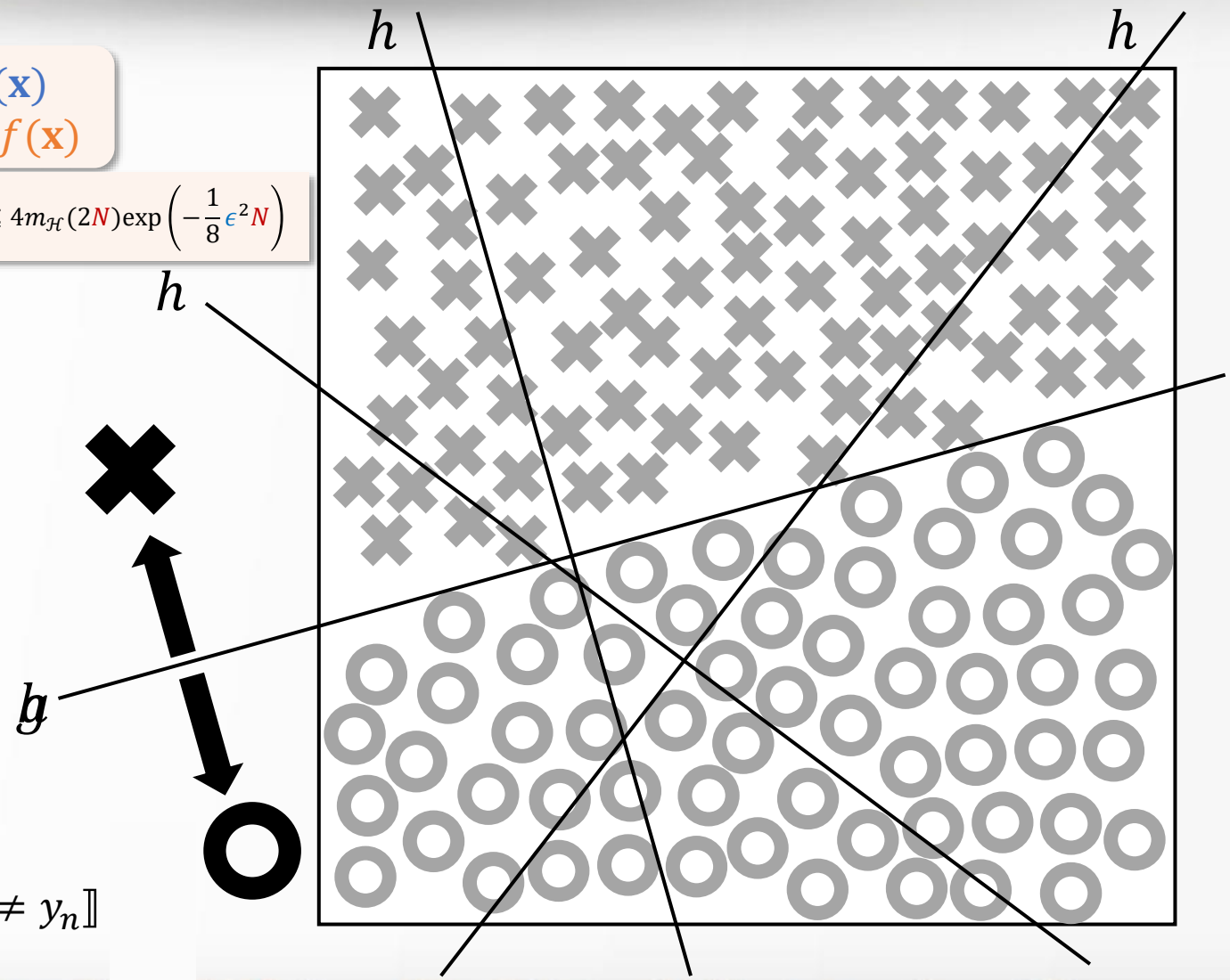
$$\mathbb{P}[|E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon] \leq 2\exp(-2\epsilon^2 N)$$

Theory of Generalization: $E_{\text{in}} \approx E_{\text{out}}$ if d_{VC} is small and N is large enough

Find a Separation Line

h is right $\Leftrightarrow h(\mathbf{x}) = f(\mathbf{x})$
h is wrong $\Leftrightarrow h(\mathbf{x}) \neq f(\mathbf{x})$

$$\mathbb{P}[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon] \leq 4m_{\mathcal{H}}(2N)\exp\left(-\frac{1}{8}\epsilon^2 N\right)$$



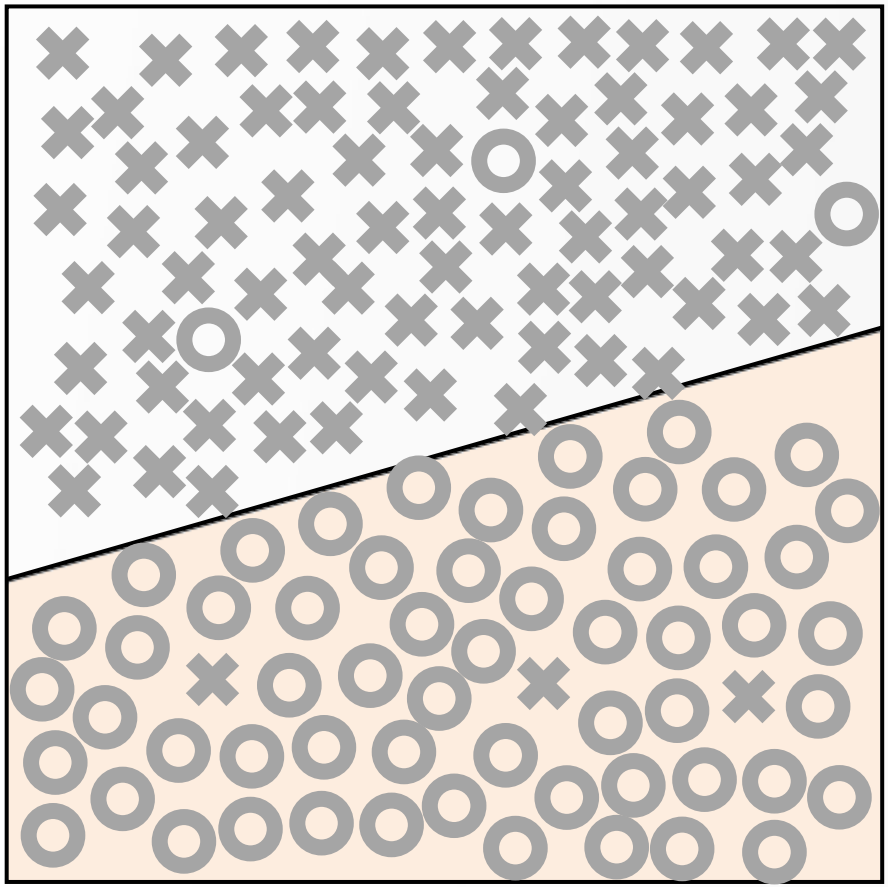
Linear Seperable!

$$E_{\text{in}}(h) = \frac{1}{N}\sum_{n=1}^{N}[\![h(\mathbf{x}_n) \neq y_n]\!]$$

Noise & Model Complexity

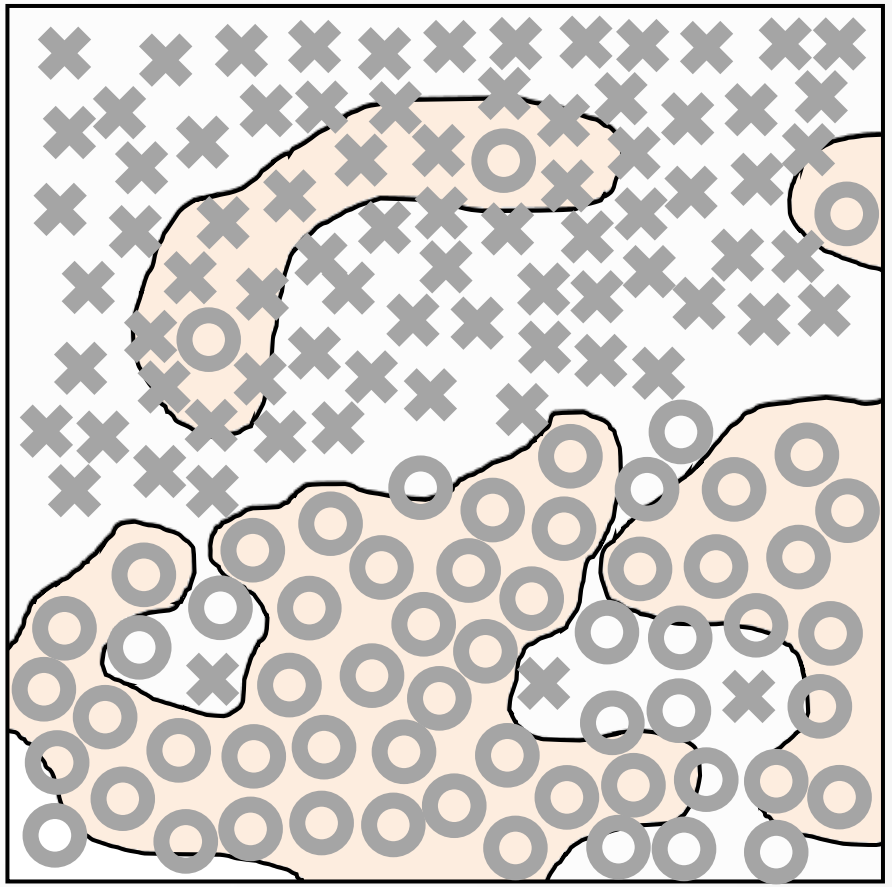
$$\mathbb{P}[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon] \leq 4 m_{\mathcal{H}}(2N) \exp\left(-\frac{1}{8}\epsilon^2 N\right)$$

Low Complexity



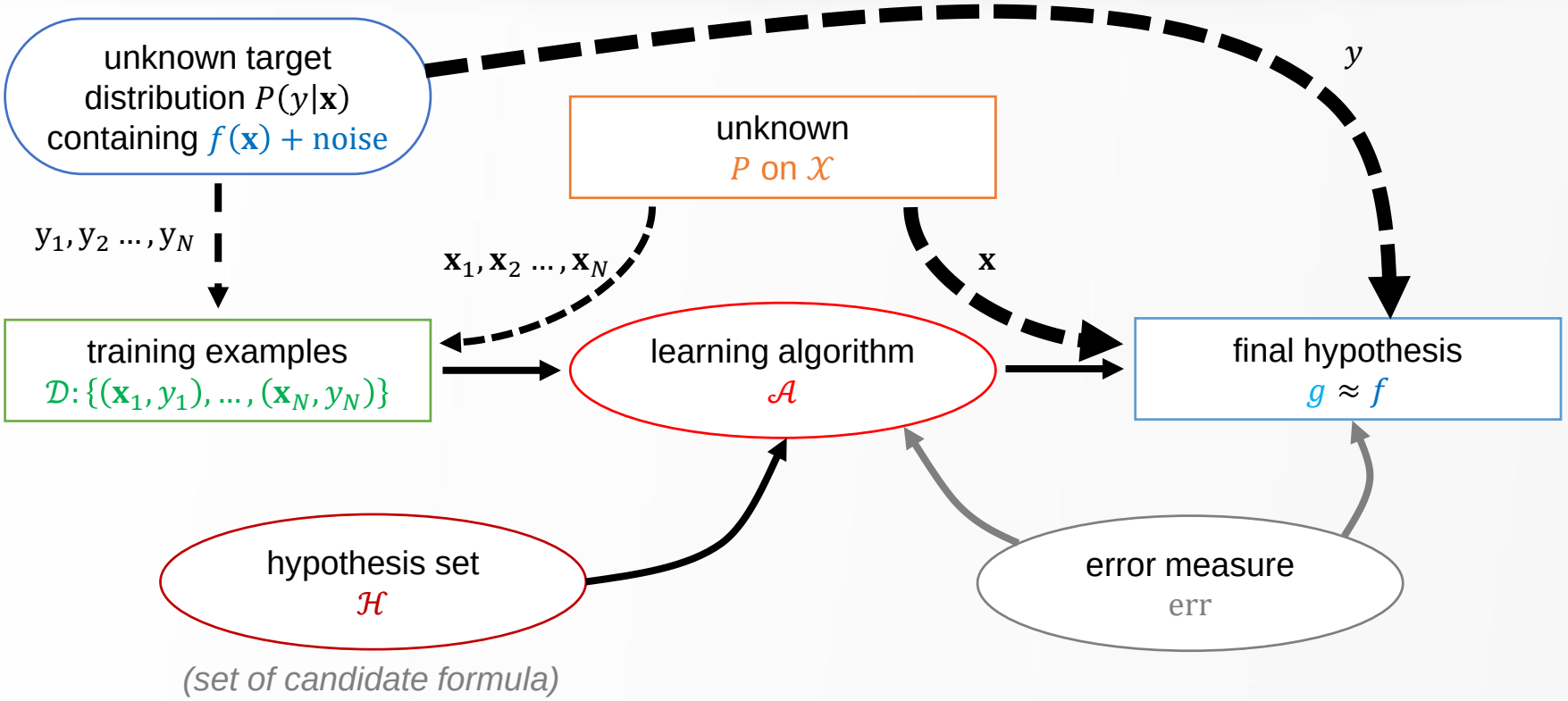
$E_{\text{in}}(h) = \frac{6}{72}$ $E_{\text{out}}(h) \fallingdotseq \frac{6}{140}$

High Complexity



$E_{\text{in}}(h) = \frac{0}{72}$ $E_{\text{out}}(h) \fallingdotseq \frac{27}{140}$

Statistical Learning Flow



If 1) finds one $h^* \in \mathcal{H}$ with $E_{\text{in}}(h^*) \approx 0$
2) PAC guarantee for $E_{\text{out}}(h^*) \approx 0$

$\Longrightarrow h^* = g \approx f$

Outline

- Introduction to Machine Learning
- Theory of Generalization
- Learning Algorithm
- Hazard of Overfitting
- Blending and Bagging

A Simple Hypothesis Set : Perceptron

- For $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_d)$, 'features of customer', compute a weighted 'score' and

  Approve ticket if $\sum_{i=1}^{d} w_i \mathrm{x}_1 > \text{threshold}$

  Deny ticket if $\sum_{i=1}^{d} w_i \mathrm{x}_1 < \text{threshold}$

- $\mathcal{Y}: \{+1(\text{good}), -1(\text{bad})\}$, 0 ignored
  - linear formula $h \in \mathcal{H}$ are

$$h(\mathbf{x}) = \text{sign}\left(\left(\sum_{i=1}^{d} w_i \mathrm{x}_1\right) - \text{threshold}\right)$$

called 'perceptron' hypothesis historically

Vector Form of Perception Hypothesis

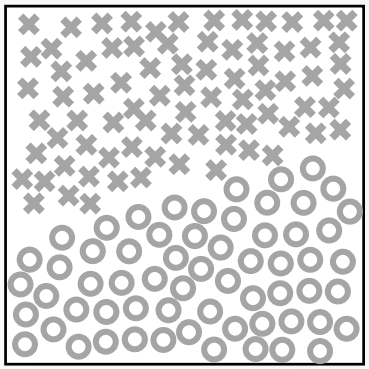
$$
\begin{aligned}
h(\mathbf{x}) &= \text{sign}\left(\left(\sum_{i=1}^{d} w_i \mathrm{x}_i\right) - \text{threshold}\right) \\
&= \text{sign}\left(\left(\sum_{i=1}^{d} w_i \mathrm{x}_i\right) + (-\text{threshold}) \cdot (+1)\right) \\
&= \text{sign}\left(\left(\sum_{i=0}^{d} w_i \mathrm{x}_i\right)\right) \\
&= \text{sign}(\mathbf{w}^T \mathbf{x})
\end{aligned}
$$

- Each 'tall' $\mathbf{w}$ represents hypothesis h & is multiplied with 'tall' $\mathbf{x}$, and we will use tall versions to simplify notation.

perceptron $\Leftrightarrow$ linear (binary) classifiers

Select g from $\mathcal{H}$

- $\mathcal{H}$ = all possible perceptrons, g = ?
  - want: $g \approx f$ (hard when f unknown)
  - almost necessary: $g \approx f$ on $\mathcal{D}$
    - ideally $g(\mathbf{x}_n) = f(\mathbf{x}_n) = y_n$
  - difficult: $\mathcal{H}$ is of infinite size
  - idea: start from some g_0, and 'correct' its mistakes on $\mathcal{D}$
  - will represent g_0 by its weight vector $\mathbf{w}_0$



Perceptron Learning Algorithm

- Start from some $\mathbf{w}_0$ (say $\mathbf{0}$), and 'correct' its mistakes on $\mathcal{D}$

for $t = 0, 1, \ldots$

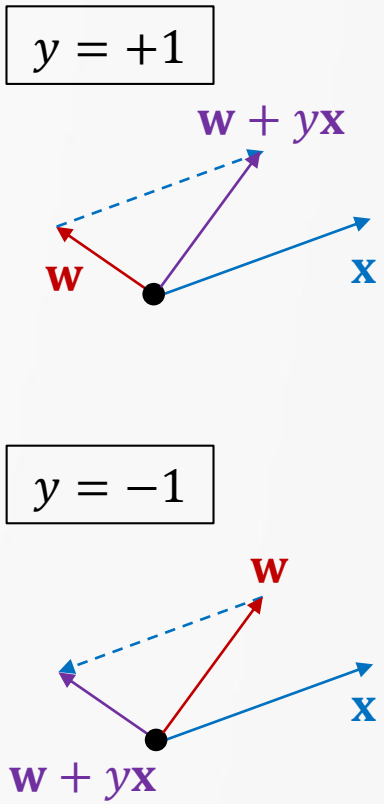
① find a mistake of $\mathbf{w}_t$ called $(\mathbf{x}_{n(t)}, y_{n(t)})$

$$\text{sign}\left(\mathbf{w}_t^T \mathbf{x}_{n(t)}\right) \neq y_{n(t)}$$

② (try to) correct the mistake by

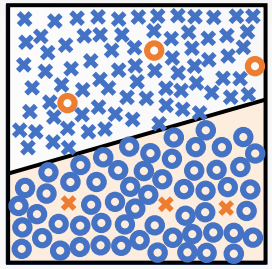
$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}$$

…until no more mistakes
return last $\mathbf{w}$ (called $\mathbf{w}_{\text{PLA}}$) as g



Line with Noise Tolerance

- Assume 'little' noise: $y_n = f(\mathbf{x}_n)$ usually
- If so, $g \approx f$ on $\mathcal{D}$ $\Leftrightarrow$ $y_n = g(\mathbf{x}_n)$ usually
- How about



$$\mathbf{w}_g \leftarrow \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} [\![y_n \neq \operatorname{sign}(\mathbf{w}^T \mathbf{x}_n)]\!]$$

  - NP-hard to solve, unfortunately

Can we modify PLA to get an 'approximately good' g?

Pocket Algorithm

- Modify PLA algorithm (black lines) by keeping best weights in pocket.

initialize pocket weights $\hat{\mathbf{w}}$

for $t = 0, 1, \ldots$

① find a (random) mistake of $\mathbf{w}_t$ called $(\mathbf{x}_{n(t)}, y_{n(t)})$

② (try to) correct the mistake by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\mathbf{x}_{n(t)}$$

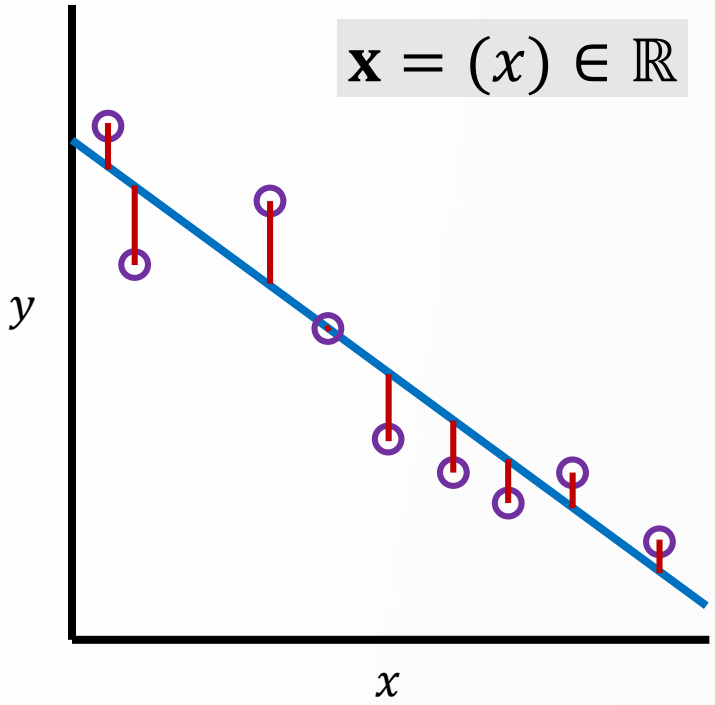
③ if $\mathbf{w}_{t+1}$ makes fewer mistakes than $\hat{\mathbf{w}}$, replace $\hat{\mathbf{w}}$ by $\mathbf{w}_{t+1}$

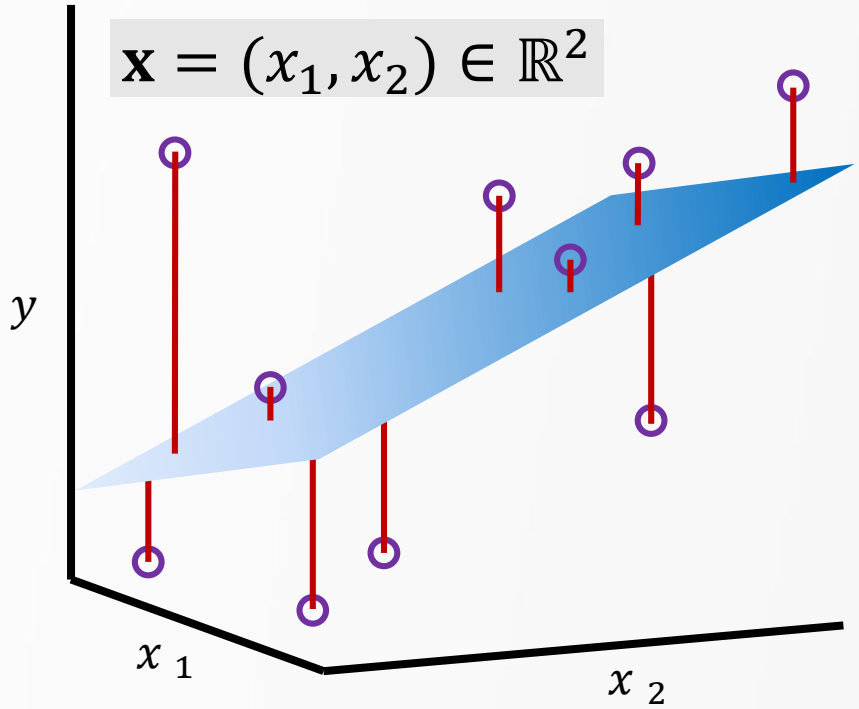
…until enough iterations

return $\hat{\mathbf{w}}$ (called $\mathbf{w}_{\text{POCKET}}$) as g

Regression: $\mathcal{Y} = \mathbb{R}$

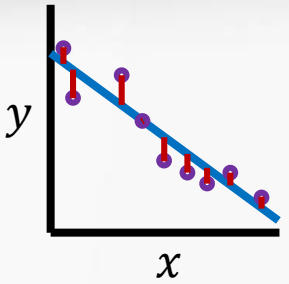
- Linear regression
  - find lines/hyperplanes with small residuals





Error Measure

- Popular/historical error measure:
  - square error $\text{err}(\hat{y}, y) = (\hat{y} - y)^2$



In-sample Error	Out-of-sample Error
$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}\left(\underbrace{h(\mathbf{x}_n)}_{\mathbf{w}^T\mathbf{x}_n} - y_n\right)^2$	$E_{\text{out}}(\mathbf{w}) = \underset{(\mathbf{x},y)\sim P}{\mathcal{E}}\left(\mathbf{w}^T\mathbf{x}_n - y_n\right)^2$

How to minimize $E_{\text{in}}(\mathbf{w})$?

Matrix Form of $E_{\text{in}}(\mathbf{w})$

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

$$= \frac{1}{N}\left\| \begin{bmatrix} \mathbf{x}_1{}^T\mathbf{w} - y_1 \\ \mathbf{x}_2{}^T\mathbf{w} - y_2 \\ \cdots \\ \mathbf{x}_N{}^T\mathbf{w} - y_N \end{bmatrix} \right\|^2$$

$$= \frac{1}{N}\left\| \begin{bmatrix} - - \mathbf{x}_1{}^T - - \\ - - \mathbf{x}_2{}^T - - \\ \cdots \\ - - \mathbf{x}_N{}^T - - \end{bmatrix} \mathbf{w} - \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix} \right\|^2$$

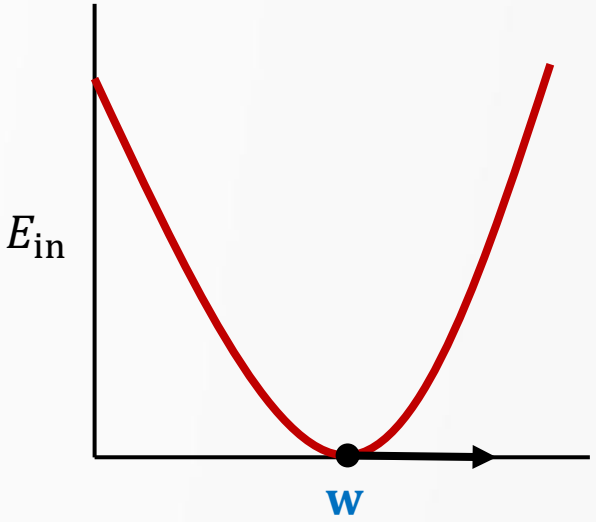
$$= \frac{1}{N}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

$\mathbf{X}: N \times (d+1)$
$\mathbf{w}: (\mathrm{d}+1) \times 1$
$\mathbf{y}: N \times 1$

How to Minimize $E_{\text{in}}(\mathbf{w})$?

- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, **convex**
- Necessary condition of 'best' $\mathbf{w}$

$$\nabla E_{\text{in}}(\mathbf{w}) \equiv \begin{bmatrix} \frac{\partial E_{\text{in}}}{\partial w_0}(\mathbf{w}) \\ \frac{\partial E_{\text{in}}}{\partial w_1}(\mathbf{w}) \\ \cdots \\ \frac{\partial E_{\text{in}}}{\partial w_d}(\mathbf{w}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix}$$



– not possible to 'roll down'

task: find $\mathbf{w}_{\text{LIN}}$ such that $\nabla E_{\text{in}}(\mathbf{w}_{\text{LIN}}) = 0$

Recap: Matrix Calculus

- Several useful vector derivative formulas
  - $\mathbf{A} : n \times n$
  - $\mathbf{x} : n \times 1$
  - $\mathbf{y} : n \times 1$

Denominator-layout notation

$\mathbf{y}$	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\mathbf{Ax}$	$\mathbf{A}^T$
$\mathbf{x}^T\mathbf{A}$	$\mathbf{A}$
$\mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^T\mathbf{Ax}$	$\mathbf{Ax} + \mathbf{A}^T\mathbf{x}$

The Gradient $E_{\text{in}}(\mathbf{w})$

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

- $\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}(2\mathbf{X}^T\mathbf{X}\mathbf{w} - 2\mathbf{X}^T\mathbf{y})$
- Task: find $\mathbf{w}_{\text{LIN}}$ such that $\frac{2}{N}(\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}) = \mathbf{0}$

invertible $\mathbf{X}^T\mathbf{X}$

- unique solution

$$\mathbf{w}_{\text{LIN}} = \underbrace{(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T}_{\text{pseudo-inverse } \mathbf{X}^\dagger}\mathbf{y}$$

- often the case because $N \gg d + 1$

singular $\mathbf{X}^T\mathbf{X}$

- many optimal solutions
- one of the solutions

$$\mathbf{w}_{\text{LIN}} = \mathbf{X}^\dagger\mathbf{y}$$

$$(\mathbf{X} \equiv \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \longrightarrow \mathbf{X}^\dagger \equiv \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^T)$$

Linear Regression Algorithm

① from $\mathcal{D}$, construct input matrix $\mathbf{X}$ and output vector $\mathbf{y}$ by

$$\mathbf{X} = \underbrace{\begin{bmatrix} -- \mathbf{x}_1{}^T -- \\ -- \mathbf{x}_2{}^T -- \\ \cdots \\ -- \mathbf{x}_N{}^T -- \end{bmatrix}}_{N \times (d+1)}, \qquad \mathbf{y} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix}}_{N \times 1}$$

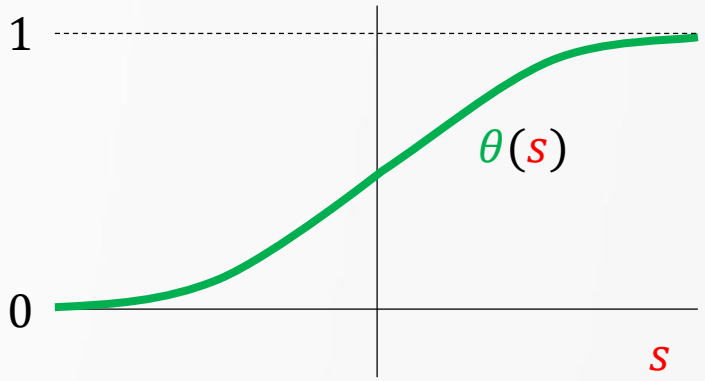
② calculate pseudo-inverse $\underbrace{\mathbf{X}^\dagger}_{(d+1) \times N}$

③ return $\underbrace{\mathbf{w}_{\text{LIN}}}_{(d+1) \times 1} = \mathbf{X}^\dagger \mathbf{y}$

Logistic Hypothesis

- For $\mathbf{x} = (x_0, x_1, x_2, \ldots, x_d)$, 'features of customer', compute a weighted 'score':

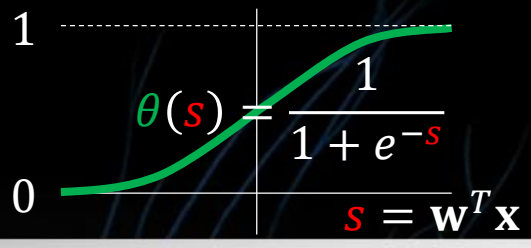
$$s = \sum_{i=0}^{d} w_i x_i = \mathbf{w}^T \mathbf{x}$$



- Convert the score to estimate probability by logistic function $\theta(s)$

logistic hypothesis: $h(\mathbf{x}) = \theta(\mathbf{w}^T \mathbf{x})$

Logistic Function



$$\theta(-\infty) = 0$$
$$\theta(0) = \frac{1}{2} \quad \Rightarrow \quad \theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$
$$\theta(\infty) = 1$$

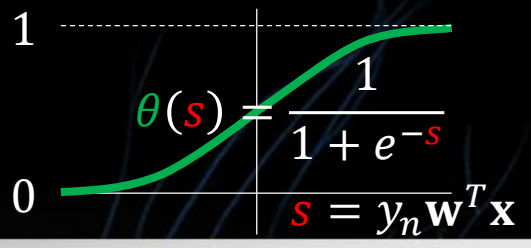
smooth, monotonic, sigmoid function of s

- **Logistic regression**: use

$$h(\mathbf{x}) = \theta(s) = \theta(\mathbf{w}^T\mathbf{x}) = \frac{1}{1+\exp(-\mathbf{w}^T\mathbf{x})}$$

to approximate target function $f(\mathbf{x}) = P(y|\mathbf{x})$

Cross-Entropy Error



$$g = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{n=1}^{N} \theta(y_n \mathbf{w}^T \mathbf{x_n}) \equiv \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{n=1}^{N} -\ln\theta(y_n \mathbf{w}^T \mathbf{x_n})$$

- $E_{\text{in}}(\mathbf{w})$ for logistic regression:

$$\frac{1}{N}\sum_{n=1}^{N} \ln\left(1+\exp(-y_n \mathbf{w}^T \mathbf{x_n})\right) = \frac{1}{N}\sum_{n=1}^{N} \text{err}(\mathbf{w}, \mathbf{x_n}, y_n)$$

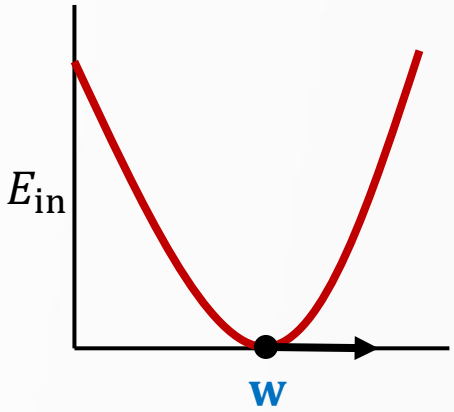
$$\text{err}(\mathbf{w}, \mathbf{x_n}, y_n) = \ln\left(1+\exp(-y_n \mathbf{w}^T \mathbf{x_n})\right)$$

cross-entropy error

Minimizing $E_{\text{in}}(\mathbf{w})$

$$\underset{\mathbf{w}}{\operatorname{argmin}}\, E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1+\exp\left(-y_n \mathbf{w}^T \mathbf{x}_n\right)\right)$$

- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, **convex**



- How to minimize? Locate valley $\Longrightarrow$ $\nabla E_{\text{in}}(\mathbf{w}) = 0$

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

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

$$\underset{\mathbf{w}}{\operatorname{argmin}}\, E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \ln\left(1+\exp(-y_n \mathbf{w}^T \mathbf{x_n})\right)$$

$$\Longrightarrow \nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \theta(-y_n \mathbf{w}^T \mathbf{x_n})(-y_n \mathbf{x_n}) = 0$$

- Scaled θ-weighted sum of $-y_n \mathbf{x_n}$
  - all $\theta(\cdot) = 0$?
    - only if $y_n \mathbf{w}^T \mathbf{x_n} \gg 0$ (linear separable $\mathcal{D}$)
  - weighted sum $= 0$?
    - non-linear

No closed-form solution!

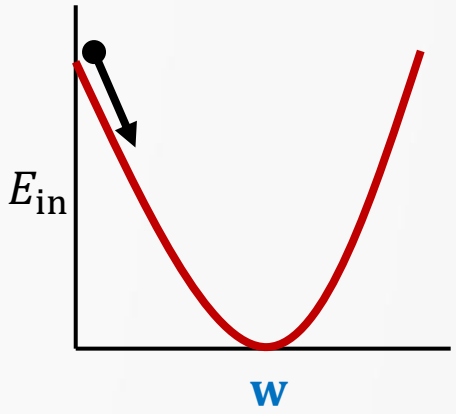
Iterative Optimization

for $t = 0,1, \ldots$

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \mathbf{v}$$

when stop, return last $\mathbf{w}$ as g

- Smooth $E_{\text{in}}(\mathbf{w})$ for logistic regression: choose $\mathbf{v}$ to get the ball roll 'downhill'?
  - direction $\mathbf{v}$ (assumed of unit length)
  - step size η (assumed positive)
- A greedy approach for some given $\eta > 0$:



$$\min_{\|\mathbf{v}\|=1} E_{\text{in}} \left(\underbrace{\mathbf{w}_t + \eta \mathbf{v}}_{\mathbf{w}_{t+1}} \right)$$

Gradient Descent

$$\min_{\|\mathbf{v}\|=1} E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v})$$

- If η is small, then by Taylor expansion:

$$E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v}) \approx \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \mathbf{v}^T \underbrace{\nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

- Optimal $\mathbf{v}$: opposite direction of $\nabla E_{\text{in}}(\mathbf{w}_t)$

$$\mathbf{v} = -\frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$$

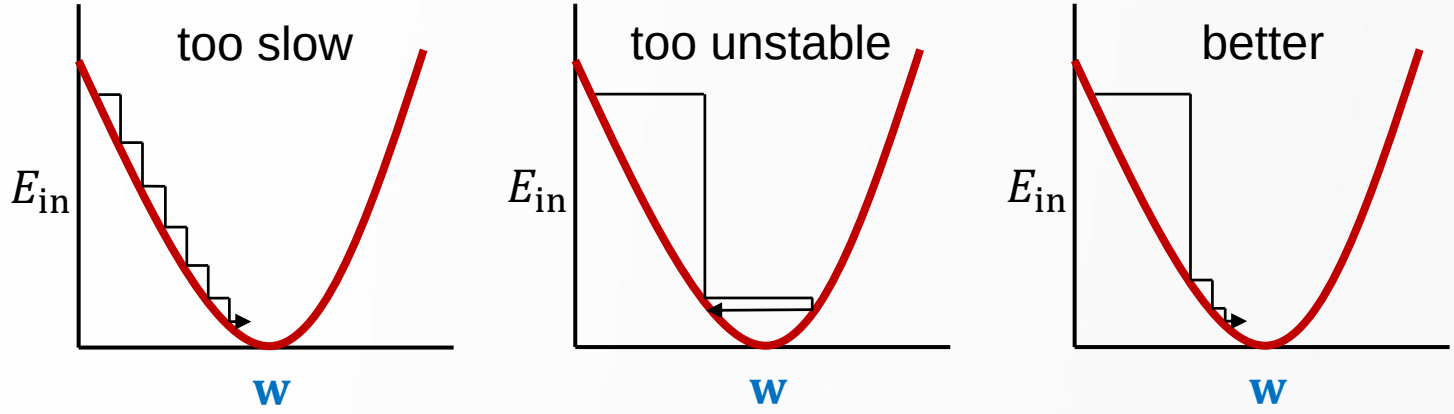
- Gradient descent: for small η

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|}$$

Gradient descent: a simple & popular optimization tool

Choice of η

- η should better be monotonic of $\|\nabla E_{\text{in}}(\mathbf{w}_t)\|$



- If red $\eta \propto \|\nabla E_{\text{in}}(\mathbf{w}_t)\|$ by ratio purple η

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \frac{\nabla E_{\text{in}}(\mathbf{w}_t)}{\|\nabla E_{\text{in}}(\mathbf{w}_t)\|} \implies \mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

the fixed learning rate

Logistic Regression Algorithm

initialize $\mathbf{w}_0$

for $t = 0,1, \ldots$

① compute

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N} \theta(-y_n \mathbf{w}^T \mathbf{x}_n)(-y_n \mathbf{x}_n)$$

② update by

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta \nabla E_{\text{in}}(\mathbf{w}_t)$$

…until $\nabla E_{\text{in}}(\mathbf{w}_t) = 0$ or enough iterations

return last $\mathbf{w}_t$ as g

Stochastic Gradient Descent (SGD)

SGD logistic regression:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta\theta(-y_n\mathbf{w}^T\mathbf{x}_n)(y_n\mathbf{x}_n)$$

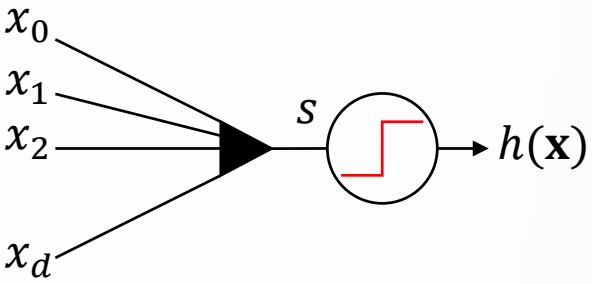
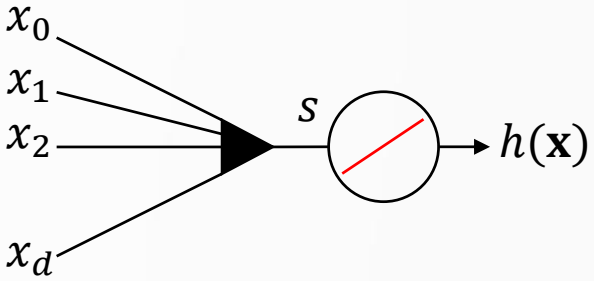
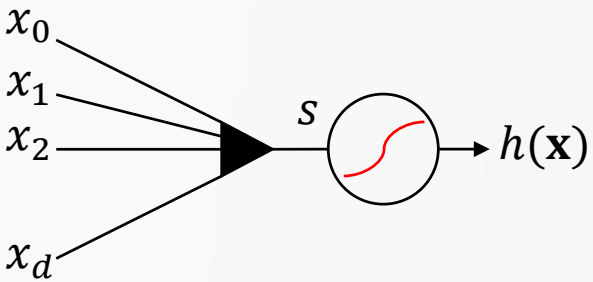
over uniform choice of n

- Idea: replace true gradient by stochastic gradient
- After enough steps,

  average true gradient $\approx$ average stochastic gradient
- pros: simple & cheaper computation
  - useful for big data or online learning
- cons: less stable in nature

Three Linear Models

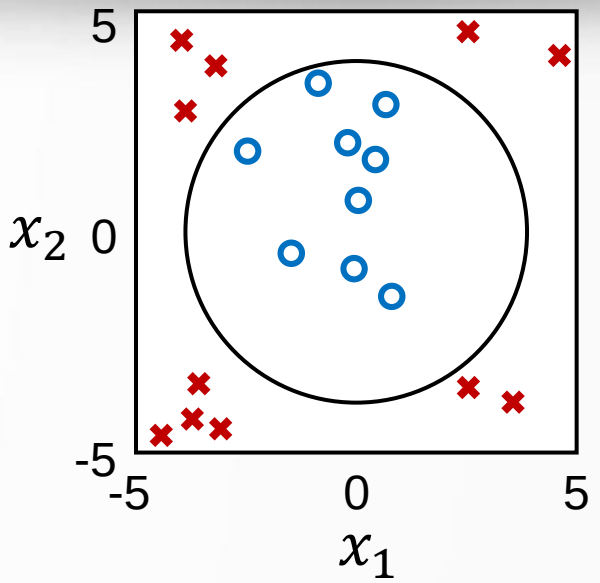
- Linear weighted sum: $s = \mathbf{w}^T\mathbf{x}$

linear classification	linear regression	logistic regression
$h(\mathbf{x}) = \text{sign}(s)$	$h(\mathbf{x}) = s$	$h(\mathbf{x}) = \theta(s)$
		
error = 0/1	error = squared	error = cross-entropy
discrete $E_{\text{in}}(\mathbf{w})$: NP-hard to solve	quadratic convex $E_{\text{in}}(\mathbf{w})$: closed-form solution	smooth convex $E_{\text{in}}(\mathbf{w})$: gradient descent

Outline

- Introduction to Machine Learning
- Theory of Generalization
- Learning Algorithm
- Hazard of Overfitting
- Blending and Bagging

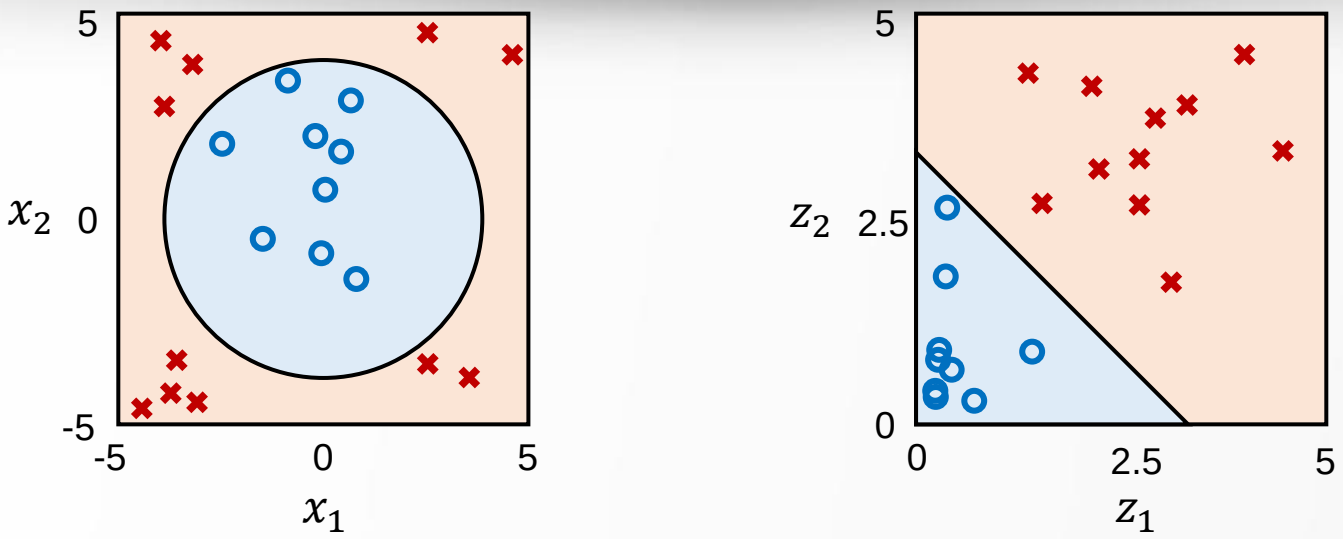
Circular Separable



- $\mathcal{D}$ is not linear separable but circular separable by a circle of radius $\underline{3}$ centered at origin:

$$h(\mathbf{x}) = \text{sign}(-x_1^2 - x_2^2 + 9)$$
$$= \text{sign}(\underbrace{9}_{\tilde{w}_0} \cdot \underbrace{1}_{z_0} + \underbrace{(-1)}_{\tilde{w}_1} \cdot \underbrace{x_1^2}_{z_1} + \underbrace{(-1)}_{\tilde{w}_2} \underbrace{x_2^2}_{z_2})$$
$$= \text{sign}(\tilde{w}^T \mathbf{z})$$

Circular Separable and Linear Separable



- $\{(\mathbf{x}_n, y_n)\}$ circular separable $\Longrightarrow$ $\{(\mathbf{z}_n, y_n)\}$ linear separable
- **Nonlinear Feature Transform**: $\mathbf{x} \in \mathcal{X} \xrightarrow{\mathbf{\Phi}} \mathbf{z} \in \mathcal{Z}$

$$(z_0, z_1, z_2) = \mathbf{z} = \mathbf{\Phi}(\mathbf{x}) = (1, x_1^2, x_2^2)$$

$$h(\mathbf{x}) = \tilde{h}(\mathbf{z}) = \text{sign}\left(\tilde{w}^T \mathbf{\Phi}(\mathbf{x})\right) = \text{sign}(\tilde{w}_0 + \tilde{w}_1 x_1^2 + \tilde{w}_2 x_2^2)$$

Linear Hypothesis in $\mathcal{Z}$-Space

- General quadratic hypothesis set: A 'bigger' $\mathcal{Z}$-Space

$$\boldsymbol{\Phi}_2(\mathbf{x}) = (1, x_1, x_2, x_1^2, x_1x_2, x_2^2)$$

  - perceptron in $\mathcal{Z}$-Space $\Leftrightarrow$ quadratic hypotheses in $\mathcal{X}$-Space

- Q-th order polynomial transform:

$$\boldsymbol{\Phi}_Q(\mathbf{x}) = \left(1, x_1, x_2, \dots, x_d, x_1^2, x_1x_2, \dots, x_d^2, \dots, x_1^Q, x_1^{Q-1}x_2, \dots, x_d^Q\right)$$

  - $\underbrace{1}_{\widetilde{w}_0} + \underbrace{\tilde{d}}_{\text{others}}$ dimensions $= O(Q^d)$

  - number of free parameters $\widetilde{w}_i = 1 + \tilde{d} \approx d_{\text{VC}}(\mathcal{H}_{\boldsymbol{\Phi}_Q})$

$$\mathcal{H}_{\boldsymbol{\Phi}_Q} = \left\{h(\mathbf{x}) : h(\mathbf{x}) = \tilde{h}\left(\boldsymbol{\Phi}_Q(\mathbf{x})\right) \text{ for some linear } \tilde{h} \text{ on } \mathcal{Z}\right\}$$

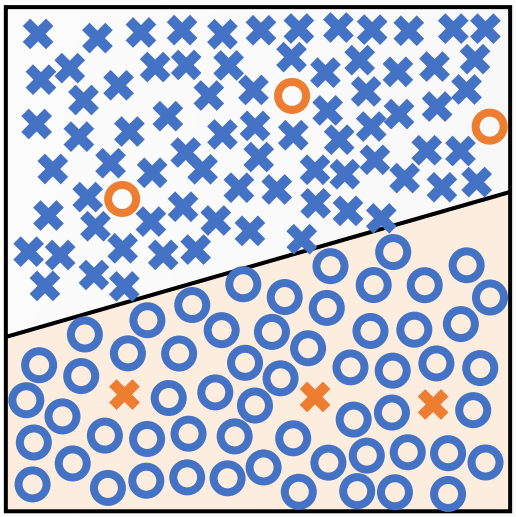
Q large $\Longrightarrow$ large d_{VC}

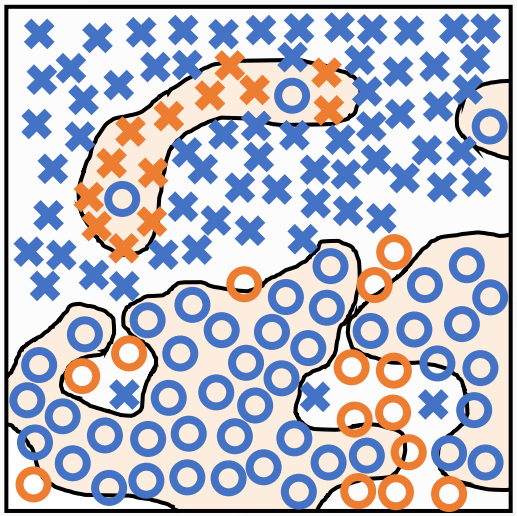
Hazard of Overfitting

- Vapnik-Chervonenkis (VC) bound (remember?):

$$\mathbb{P}[\exists h \in \mathcal{H} \text{ s.t. } |E_{\text{in}}(h) - E_{\text{out}}(h)| > \epsilon] \leq 4\underbrace{m_{\mathcal{H}}(2N)}_{\text{Model Complexity } O(N^{d_{VC}})}\exp\left(-\frac{1}{8}\epsilon^2 N\right)$$

small d_{VC}



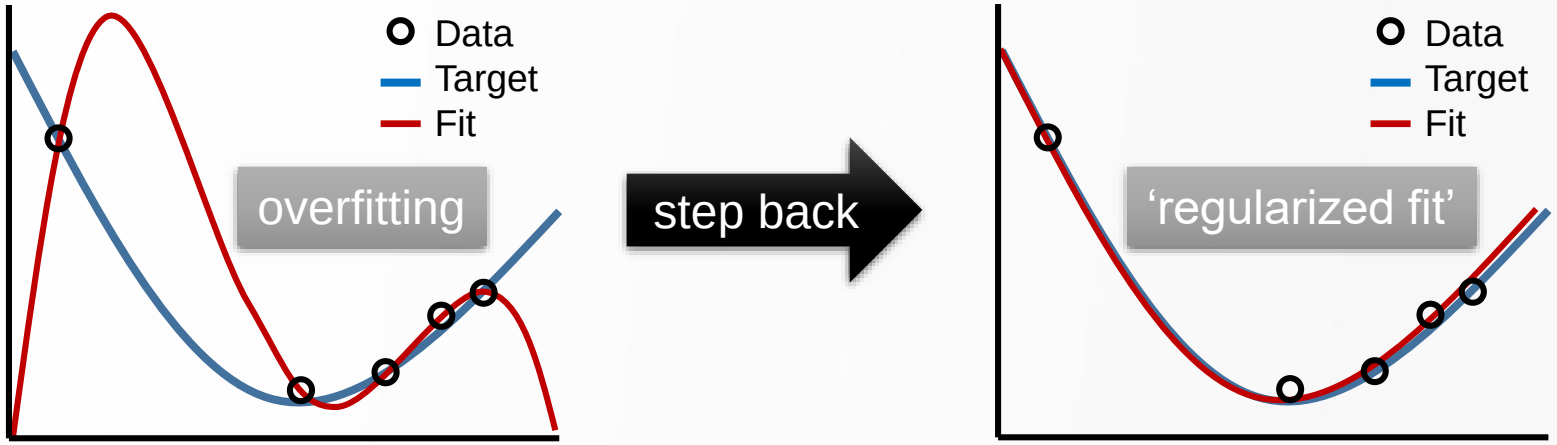


large d_{VC}

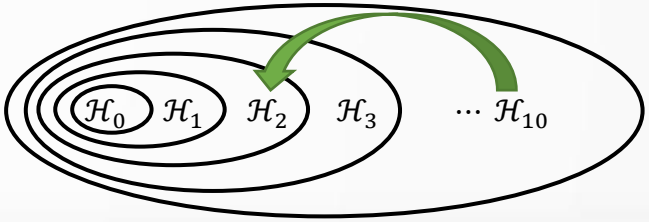
Regularization: The Magic

- Q-th order polynomial transform for $x \in \mathbb{R}$ + linear regression:

$$\boldsymbol{\Phi}_Q(\mathbf{x}) = (1, x, x^2, \dots, x^Q)$$



- Idea: 'step back' from $\mathcal{H}_{10}$ to $\mathcal{H}_2$



Stepping Back as Constraint

- hypothesis $\mathbf{w}$ in $\mathcal{H}_{10}$: $w_0 + w_1 x + w_2 x^2 + w_3 x^3, \ldots, + w_{10} x^{10}$
- hypothesis $\mathbf{w}$ in $\mathcal{H}_2$: $w_0 + w_1 x + w_2 x^2$

$\Longrightarrow$ that is, $\boxed{\mathcal{H}_2} = \boxed{\mathcal{H}_{10} \text{ \& constraint that } w_3 = w_4 = \cdots = w_{10} = 0}$

$\mathcal{H}_{10} \equiv \{\mathbf{w} \in \mathbb{R}^{10+1}\}$	$\mathcal{H}_2 \equiv \{\mathbf{w} \in \mathbb{R}^{10+1} \text{ while } w_3 = w_4 = \cdots = w_{10} = 0\}$
regression with $\mathcal{H}_{10}$: $\min_{\mathbf{w} \in \mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w})$	regression with $\mathcal{H}_2$: $\min_{\mathbf{w} \in \mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w})$ s.t. $w_3 = w_4 = \cdots = w_{10} = 0$

step back = constrained optimization of E_{in}

In fact, we can just use $\mathbf{w} \in \mathbb{R}^{2+1}$ in this case.

Regression with Looser Constraint

- regression with $\mathcal{H}_2 \equiv \{\mathbf{w} \in \mathbb{R}^{10+1}, \text{ while } w_3 = \cdots = w_{10} = 0\}$:

$$\min_{\mathbf{w}\in\mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w}) \text{ s.t. } w_3 = \cdots = w_{10} = 0$$

- regression with $\mathcal{H}_2' \equiv \{\mathbf{w} \in \mathbb{R}^{10+1}, \text{ while } \geq 8 \text{ of } w_q = 0\}$:

$$\min_{\mathbf{w}\in\mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w}) \text{ s.t. } \sum_{q=0}^{10} [\![w_q \neq 0]\!] \leq 3$$

  - more flexible than $\mathcal{H}_2$: $\mathcal{H}_2 \subset \mathcal{H}_2'$
  - less risky than $\mathcal{H}_{10}$: $\mathcal{H}_2' \subset \mathcal{H}_{10}$

Bad news for sparse hypothesis set $\mathcal{H}_2'$:
NP-hard to solve.

Regression with Softer Constraint

- regression with $\mathcal{H}_2' \equiv \{\mathbf{w} \in \mathbb{R}^{10+1},\ \text{while}\ \geq 8\ \text{of}\ w_q = 0\}$:

$$\min_{\mathbf{w}\in\mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w})\ \text{s.t.} \sum_{q=0}^{10} [\![w_q \neq 0]\!] \leq 3$$

- regression with $\mathcal{H}(C) \equiv \{\mathbf{w} \in \mathbb{R}^{10+1},\ \text{while}\ \|\mathbf{w}\|^2 \leq C\}$:

$$\min_{\mathbf{w}\in\mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w})\ \text{s.t.} \sum_{q=0}^{10} w_q^2 \leq C$$

  - $\mathcal{H}(C)$: overlaps but not exactly the same as $\mathcal{H}_2'$
  - soft and smooth strudcture over $C \geq 0$ $\quad \mathcal{H}(0) \subset \mathcal{H}(1) \subset \mathcal{H}(2) \subset \cdots \subset \mathcal{H}(\infty) = \mathcal{H}_{10}$

regularized hypothesis $\mathbf{w}_{\text{REG}}$:
optimal solution from regularized hypothesis set $\mathcal{H}(C)$

The Lagrange Multiplier

- $\min_{\mathbf{w}\in\mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w}) = \frac{1}{\text{N}} \underbrace{\sum_{\text{n}=1}^{\text{N}} (\mathbf{w}^T \mathbf{z}_{\text{n}} - \text{y}_{\text{n}})^2}_{(\mathbf{Z}\mathbf{w}-\mathbf{y})^T(\mathbf{Z}\mathbf{w}-\mathbf{y})}$ s.t. $\underbrace{\sum_{\text{q}=0}^{\text{Q}} \text{w}_{\text{q}}^2}_{\mathbf{w}^T\mathbf{w}} \leq C$

$$\Longrightarrow \min_{\mathbf{w}\in\mathbb{R}^{10+1}} E_{\text{in}}(\mathbf{w}) = \frac{1}{\text{N}} (\mathbf{Z}\mathbf{w} - \mathbf{y})^T (\mathbf{Z}\mathbf{w} - \mathbf{y}) \text{ s.t. } \mathbf{w}^T\mathbf{w} \leq C$$

- Assume $\mathbf{w}^T\mathbf{w} = C$, find Lagrange multiplier $\lambda > 0$ and $\mathbf{w}_{\text{REG}}$ such that

$$\frac{\partial}{\partial \mathbf{w}} (E_{\text{in}}(\mathbf{w}) + \underbrace{\lambda}_{\lambda/N} \mathbf{w}^T\mathbf{w}) = 0$$

$$\Longrightarrow \nabla E_{\text{in}}(\mathbf{w}_{\text{REG}}) + 2\underbrace{\lambda}_{\lambda/N} \mathbf{w}_{\text{REG}} = 0$$

$$\Longrightarrow \frac{2}{N} (\mathbf{Z}^T\mathbf{Z}\mathbf{w}_{\text{REG}} - \mathbf{Z}^T\mathbf{y}) + 2\underbrace{\lambda}_{\lambda/N} \mathbf{w}_{\text{REG}} = 0$$

Ridge Regression

$$\frac{\partial}{\partial \mathbf{w}}\left(E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}\right) = 0$$

- Augmented error $E_{\text{aug}}(\mathbf{w})$: $E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\underbrace{\mathbf{w}^T\mathbf{w}}_{\text{regularizer}}$
- Regularization with E_{aug} instead of constrained E_{in}:

$$\mathbf{w}_{\text{REG}} \leftarrow \underset{\mathbf{w}}{\text{argmin}}\, E_{\text{aug}}(\mathbf{w}) \text{ for given } \lambda > 0$$

optimal solution:

$$\frac{\partial}{\partial \mathbf{w}_{\text{REG}}} E_{\text{aug}}(\mathbf{w}_{\text{REG}}) = 0 \implies \mathbf{w}_{\text{REG}} \leftarrow (\mathbf{Z}^T\mathbf{Z} + \lambda \mathrm{I})^{-1}\mathbf{Z}^T\mathbf{y}$$

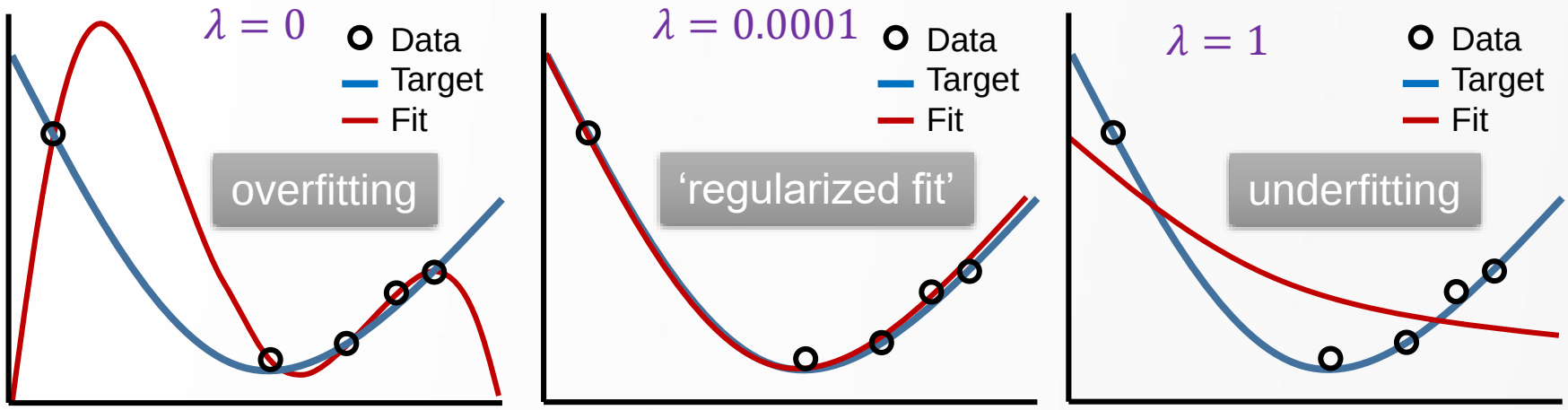
- called ridge regression in Statistics

minimizing unconstrained E_{aug} effectively minimizes some C-constrained E_{in}

The Results

$$\frac{\partial}{\partial \mathbf{w}}\left(E_{\text{in}}(\mathbf{w}) + \frac{\lambda}{N}\mathbf{w}^T\mathbf{w}\right) = 0$$

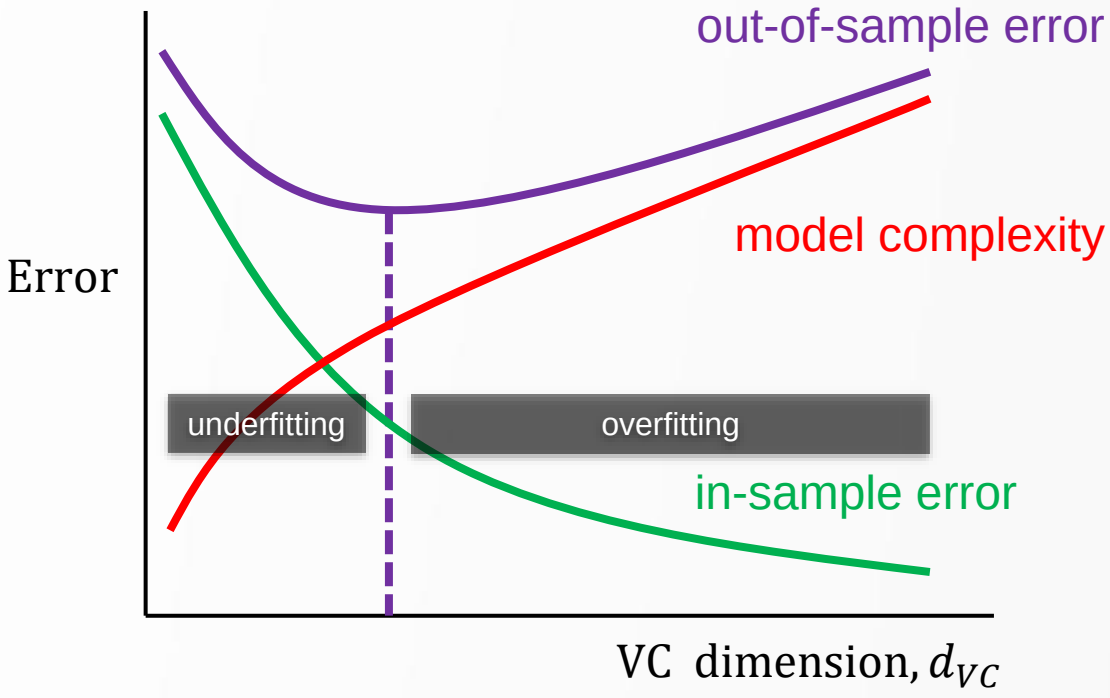
- Seeing is believing



  - philosophy: a little **regularization** goes a long way!
  - call '$\frac{\lambda}{N}\mathbf{w}^T\mathbf{w}$' weight-decay regularization:
    - larger λ $\Leftrightarrow$ prefer shorter $\mathbf{w}$ $\Leftrightarrow$ effectively smaller C

The VC Message

- The best d_{VC}^{*} is in the middle.

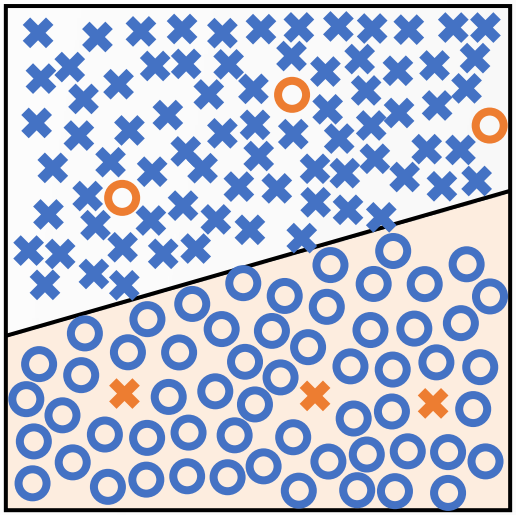


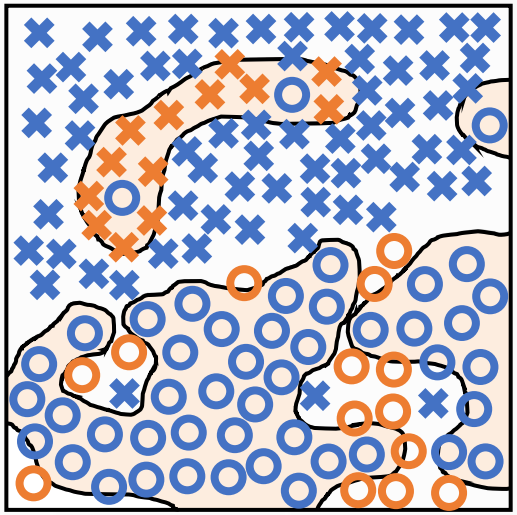
Model Selection Problem

- Which one is better?

$E_{\text{in}}(h) = \frac{6}{72}$

$E_{\text{out}}(h) = \frac{6}{140}$





$E_{\text{in}}(h) = \frac{0}{72}$

$E_{\text{out}}(h) = \frac{27}{140}$

selecting by E_{val}

Validation Set $\mathcal{D}_{\text{val}}$

$$E_{\text{in}}(h) \quad \uparrow \quad \underbrace{\mathcal{D}}_{\text{size } N} \quad \downarrow \quad g_m = \mathcal{A}_m(\mathcal{D})$$

$$\Longrightarrow$$

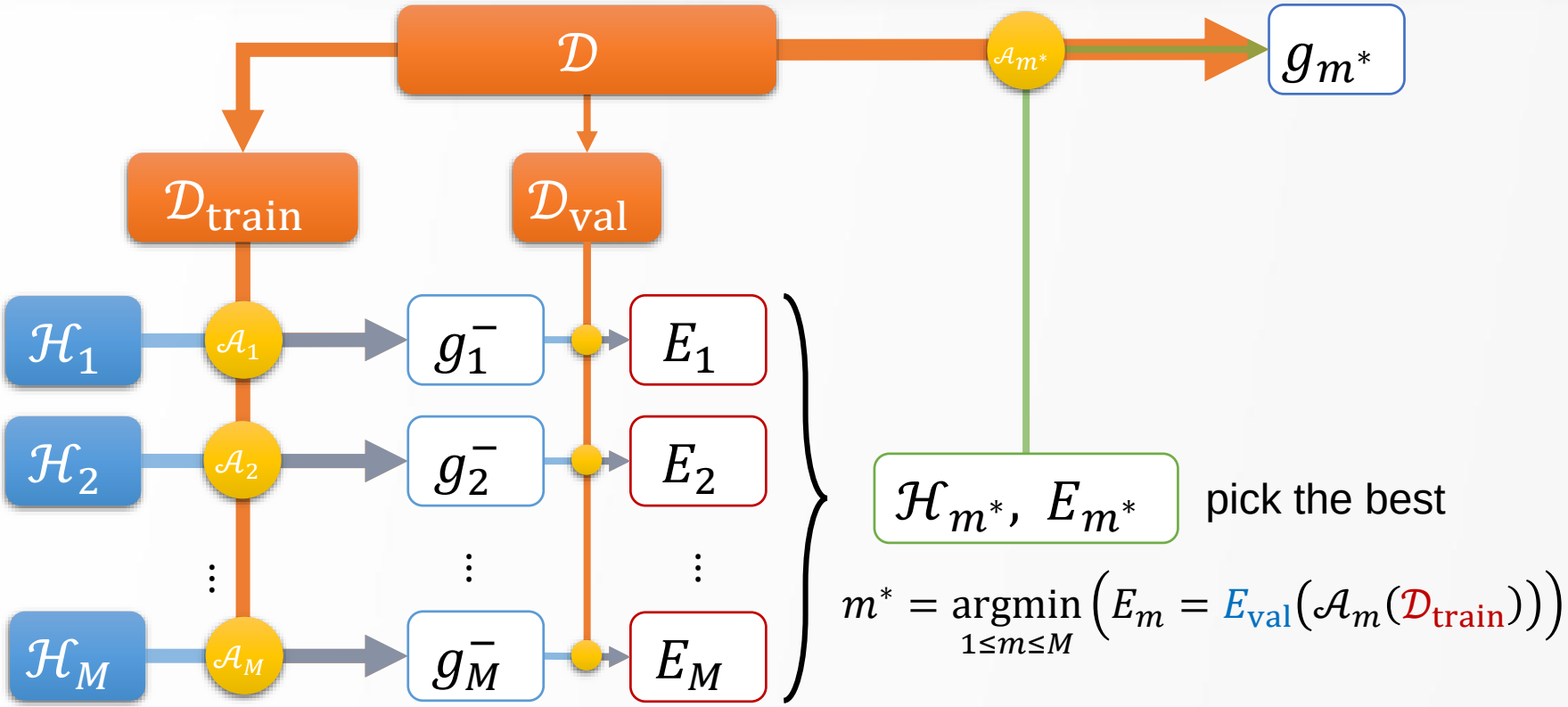
$$\underbrace{\mathcal{D}_{\text{train}}}_{\text{size } K} \quad \downarrow \quad g_m^- = \mathcal{A}_m(\mathcal{D}_{\text{train}})$$

$$\cup \quad \underbrace{\mathcal{D}_{\text{val}}}_{\text{size } N-K} \quad \uparrow \quad E_{\text{val}}(h)$$

m: model index

- $\mathcal{D}_{\text{val}} \subset \mathcal{D}$: called validation set
  - 'on-hand' simulation of test set
- Make sure $\mathcal{D}_{\text{val}}$ is 'clean'
  - feed only $\mathcal{D}_{\text{train}}$ to $\mathcal{A}_m$ for model selection

Model Selection by Best E_{val}



How to select $\mathcal{D}_{\text{val}}$?

V-fold Cross Validation

- V-fold cross-validation: random-partition of $\mathcal{D}$ to V equal parts
  - take $V-1$ for training and 1 for validation orderly

$$E_{\text{CV}}(\mathcal{H},\mathcal{A}) = \frac{1}{V}\sum_{v=1}^{V} E_{\text{VAL}}^{(v)}(g_v^-)$$

  - selection by E_{CV}:

$$m^* = \underset{1\le m\le M}{\text{argmin}}\left(E_m = E_{\text{CV}}(\mathcal{H}_m,\mathcal{A}_m)\right)$$

practical rule of thumb: $V = 10$

Outline

- Introduction to Machine Learning
- Theory of Generalization
- Learning Algorithm
- Hazard of Overfitting
- Blending and Bagging

Blending (Aggregation)

- Blending (or aggregation) : mix or combine hypotheses for better performance
  - Uniform blending (voting) for classification:

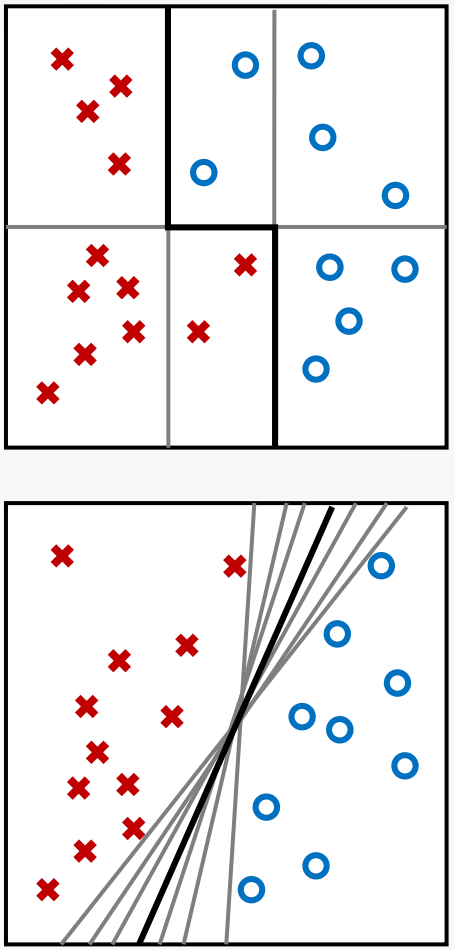
$$G(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} 1 \cdot g_m(\mathbf{x})\right)$$

  - Uniform blending regression:

$$G(\mathbf{x}) = \frac{1}{T}\sum_{m=1}^{M} 1 \cdot g_m(\mathbf{x})$$

  - Linear blending:

$$G(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_t \cdot g_m(\mathbf{x})\right) \text{ with } \alpha_m \geq 0$$



Bagging (Bootstrap Aggregation)

- Bootstrapping
  - Bootstrap sample $\widetilde{\mathcal{D}}_m$: re-sample N examples from $\mathcal{D}$ uniformly with replacement – can also use arbitrary N' instead of original N
- **B**ootstrap **Ag**gregation (**BAG**ging)
  - Consider a iterative process that for $m = 1,2,\dots,M$
    ① Request size-N' data $\widetilde{\mathcal{D}}_m$ from bootstrapping
    ② Obtain g_t by $\mathcal{A}(\widetilde{\mathcal{D}}_m)$, and $G = \text{Uniform}(\{g_t\})$

Bootstrap aggregation (Bagging):
a simple meta algorithm on top of base algorithm $\mathcal{A}$

Reference

- Machine learning slides by Prof. Hsuan-Tien Lin
  http://www.csie.ntu.edu.tw/~htlin/course/ml14fall/