

# Machine Learning Foundations

## (機器學習基石)



### Lecture 8: Noise and Error

Hsuan-Tien Lin (林軒田)

htlin@csie.ntu.edu.tw

Department of Computer Science  
& Information Engineering

National Taiwan University  
(國立台灣大學資訊工程系)



# Roadmap

- 1 When Can Machines Learn?
- 2 **Why** Can Machines Learn?

## Lecture 7: The VC Dimension

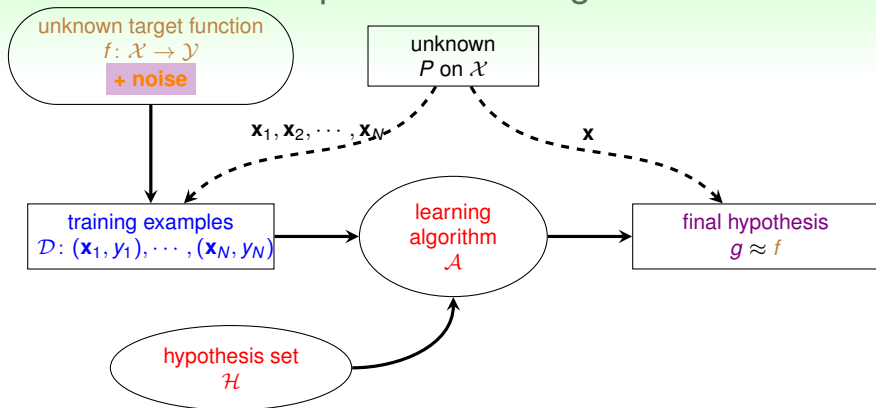
learning happens  
if **finite**  $d_{VC}$ , **large**  $N$ , and **low**  $E_{in}$

## Lecture 8: Noise and Error

- Noise and Probabilistic Target
- Error Measure
- Algorithmic Error Measure
- Weighted Classification

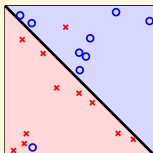
- 3 How Can Machines Learn?
- 4 How Can Machines Learn Better?

# Recap: The Learning Flow



what if there is **noise**?

# Noise



briefly introduced **noise** before **pocket** algorithm

age	23 years
gender	female
annual salary	NTD 1,000,000
year in residence	1 year
year in job	0.5 year
current debt	200,000

credit? {no(-1), yes(+1)}

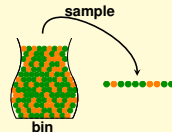
but more!

- **noise in  $y$** : good customer, 'misabeled' as bad?
- **noise in  $y$** : same customers, different labels?
- **noise in  $x$** : inaccurate customer information?

does VC bound work under **noise**?

# Probabilistic Marbles

one key of VC bound: **marbles!**



## 'deterministic' marbles

- marble  $\mathbf{x} \sim P(\mathbf{x})$
- deterministic color  
 $\llbracket f(\mathbf{x}) \neq h(\mathbf{x}) \rrbracket$

## 'probabilistic' (noisy) marbles

- marble  $\mathbf{x} \sim P(\mathbf{x})$
- probabilistic color  
 $\llbracket y \neq h(\mathbf{x}) \rrbracket$  with  $y \sim P(y|\mathbf{x})$

**same nature:** can estimate  $\mathbb{P}[\text{orange}]$  if  $\overset{i.i.d.}{\sim}$

VC holds for  $\underbrace{\mathbf{x} \overset{i.i.d.}{\sim} P(\mathbf{x}), y \overset{i.i.d.}{\sim} P(y|\mathbf{x})}_{(\mathbf{x}, y) \overset{i.i.d.}{\sim} P(\mathbf{x}, y)}$

**y isn't coming from  $f(\mathbf{x})$  but  $P(y|\mathbf{x})$**

$(\mathbf{x}, y) \overset{i.i.d.}{\sim} P(\mathbf{x}, y)$

# Target Distribution $P(y|\mathbf{x})$

characterizes behavior of ‘mini-target’ on one  $\mathbf{x}$

- can be viewed as ‘ideal mini-target’ + noise, e.g.
  - $P(\circ|\mathbf{x}) = 0.7$ ,  $P(\times|\mathbf{x}) = 0.3$
  - ideal mini-target  $f(\mathbf{x}) = \circ$
  - ‘flipping’ noise level =  $0.3 \leftarrow$  noise
- deterministic target  $f$ : **special case of target distribution**
  - $P(y|\mathbf{x}) = 1$  for  $y = f(\mathbf{x})$
  - $P(y|\mathbf{x}) = 0$  for  $y \neq f(\mathbf{x})$

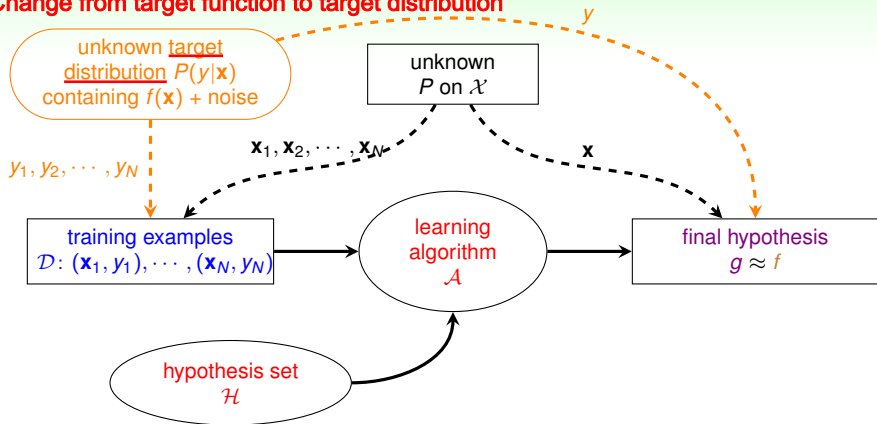
goal of learning:

predict **ideal mini-target (w.r.t.  $P(y|\mathbf{x})$ )**  
on **often-seen inputs (w.r.t.  $P(\mathbf{x})$ )**

**ML: do well on the dataset we often see.**

# The New Learning Flow

## Change from target function to target distribution



VC still works, **pocket algorithm explained :-)**

also

# Fun Time

Let's revisit PLA/pocket. Which of the following claim is true?

- 1 In practice, we should try to compute if  $\mathcal{D}$  is linear separable before deciding to use PLA.
- 2 If we know that  $\mathcal{D}$  is not linear separable, then the target function  $f$  must not be a linear function.
- 3 If we know that  $\mathcal{D}$  is linear separable, then the target function  $f$  must be a linear function.
- 4 None of the above



# Fun Time

Let's revisit PLA/pocket. Which of the following claim is true?

- ① In practice, we should try to compute if  $\mathcal{D}$  is linear separable before deciding to use PLA.
- ② If we know that  $\mathcal{D}$  is not linear separable, then the target function  $f$  must not be a linear function. **linear separable  $f$  + noise  $\rightarrow$  (possible) non linear separable  $D$**
- ③ If we know that  $\mathcal{D}$  is linear separable, then the target function  $f$  must be a linear function.
- ④ None of the above

Reference Answer: ④

① After computing if  $\mathcal{D}$  is linear separable, we shall know  $\mathbf{w}^*$  and then there is no need to use PLA. ② What about noise? ③ What about 'sampling luck'? :-)

# Error Measure

Loss function

final hypothesis  
 $g \approx f$

- how well? previously, considered out-of-sample measure

$$E_{\text{out}}(g) = \mathcal{E}_{\mathbf{x} \sim P} \llbracket g(\mathbf{x}) \neq f(\mathbf{x}) \rrbracket$$

- more generally, **error measure**  $E(g, f)$
- naturally considered
  - **out-of-sample**: averaged over unknown  $\mathbf{x}$
  - **pointwise**: evaluated on one  $\mathbf{x}$
  - **classification**:  $\llbracket \text{prediction} \neq \text{target} \rrbracket$

classification error  $\llbracket \dots \rrbracket$ :  
often also called '**0/1 error**'

# Pointwise Error Measure

can often express  $E(g, f) = \text{averaged err}(g(\mathbf{x}), f(\mathbf{x}))$ , like

$$E_{\text{out}}(g) = \mathcal{E}_{\mathbf{x} \sim P} \underbrace{\llbracket g(\mathbf{x}) \neq f(\mathbf{x}) \rrbracket}_{\text{err}(g(\mathbf{x}), f(\mathbf{x}))}$$

—err: called **pointwise error measure**

in-sample

$$E_{\text{in}}(g) = \frac{1}{N} \sum_{n=1}^N \text{err}(g(\mathbf{x}_n), f(\mathbf{x}_n))$$

out-of-sample

$$E_{\text{out}}(g) = \mathcal{E}_{\mathbf{x} \sim P} \text{err}(g(\mathbf{x}), f(\mathbf{x}))$$

will mainly consider pointwise **err** for simplicity

# Two Important Pointwise Error Measures

$$\text{err} \left( \underbrace{g(\mathbf{x})}_{\tilde{y}}, \underbrace{f(\mathbf{x})}_y \right)$$

prediction                      ground truth

## 0/1 error

$$\text{err}(\tilde{y}, y) = \mathbb{I}[\tilde{y} \neq y]$$

- correct or incorrect?
- often for **classification**

## squared error

$$\text{err}(\tilde{y}, y) = (\tilde{y} - y)^2$$

- how far is  $\tilde{y}$  from  $y$ ?
- often for **regression**

how does err **'guide' learning**?

# Ideal Mini-Target

interplay between **noise** and **error**:

$P(y|\mathbf{x})$  and **err** define ideal mini-target  $f(\mathbf{x})$

$$P(y = 1|\mathbf{x}) = 0.2, P(y = 2|\mathbf{x}) = 0.7, P(y = 3|\mathbf{x}) = 0.1$$

$$\text{err}(\tilde{y}, y) = \llbracket \tilde{y} \neq y \rrbracket$$

$$\tilde{y} = \begin{cases} 1 & \text{avg. err } 0.8 \\ 2 & \text{avg. err } 0.3(*) \\ 3 & \text{avg. err } 0.9 \\ 1.9 & \text{avg. err } 1.0 \end{cases}$$

Please guess carefully  
(really? :-))

We only have "1", "2", and "3" in ground truth.

$$f(\mathbf{x}) = \underset{y \in \mathcal{Y}}{\text{argmax}} P(y|\mathbf{x})$$

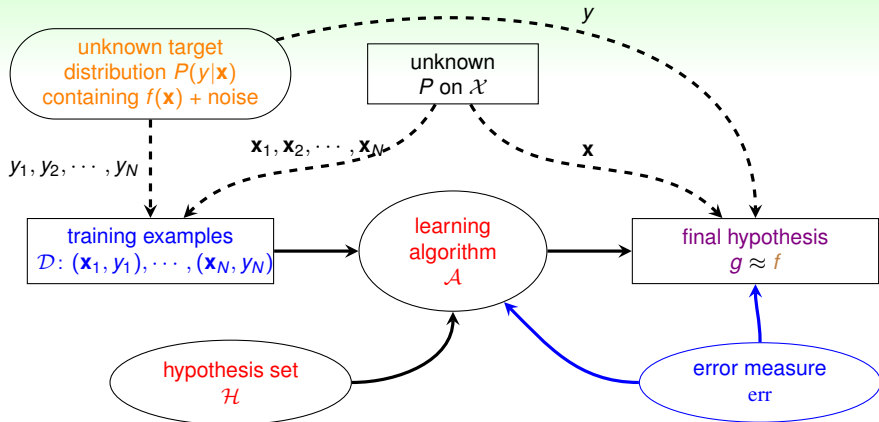
$$\text{err}(\tilde{y}, y) = (\tilde{y} - y)^2$$

$$0.2 \times (1-1)^2 + 0.7 \times (2-1)^2 + 0.1 \times (3-1)^2$$

$$\begin{cases} 1 & \text{avg. err } 1.1 \uparrow \\ 2 & \text{avg. err } 0.3 \\ 3 & \text{avg. err } 1.5 \\ 1.9 & \text{avg. err } 0.29(*) \end{cases}$$

$$f(\mathbf{x}) = \sum_{y \in \mathcal{Y}} y \cdot P(y|\mathbf{x})$$

# Learning Flow with Error Measure



extended VC theory/‘philosophy’  
**works for most  $\mathcal{H}$  and  $\text{err}$**

# Fun Time

Consider the following  $P(y|\mathbf{x})$  and  $\text{err}(\tilde{y}, y) = |\tilde{y} - y|$ . Which of the following is the ideal mini-target  $f(\mathbf{x})$ ?

$$P(y = 1|\mathbf{x}) = 0.10, P(y = 2|\mathbf{x}) = 0.35, \\ P(y = 3|\mathbf{x}) = 0.15, P(y = 4|\mathbf{x}) = 0.40.$$

- ①  $2.5 = \text{average within } \mathcal{Y} = \{1, 2, 3, 4\}$
- ②  $2.85 = \text{weighted mean from } P(y|\mathbf{x})$
- ③  $3 = \text{weighted median from } P(y|\mathbf{x})$
- ④  $4 = \text{argmax } P(y|\mathbf{x})$

# Fun Time

Consider the following  $P(y|\mathbf{x})$  and  $\text{err}(\tilde{y}, y) = |\tilde{y} - y|$ . Which of the following is the ideal mini-target  $f(\mathbf{x})$ ?

$$P(y = 1|\mathbf{x}) = 0.10, P(y = 2|\mathbf{x}) = 0.35, \\ P(y = 3|\mathbf{x}) = 0.15, P(y = 4|\mathbf{x}) = 0.40.$$

- ①  $2.5 = \text{average within } \mathcal{Y} = \{1, 2, 3, 4\} = 1$
- ②  $2.85 = \text{weighted mean from } P(y|\mathbf{x}) = 0.965$
- ③  $3 = \text{weighted median from } P(y|\mathbf{x}) = 0.95$
- ④  $4 = \text{argmax } P(y|\mathbf{x}) = 1.15$

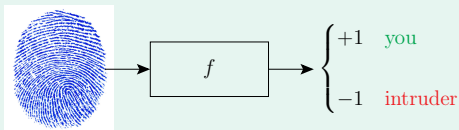
Reference Answer: ③

For the 'absolute error', the weighted median provably results in the minimum average err.



# Choice of Error Measure

## Fingerprint Verification



two types of error: **false accept** and **false reject**

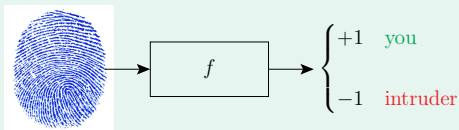
		$g$ prediction	
		+1	-1
$f$ ground truth	+1	no error	false reject ← False negative
	-1	false accept	no error

False positive (arrow pointing to false accept)

0/1 error penalizes both types **equally**

# Fingerprint Verification for Supermarket

## Fingerprint Verification



two types of error: **false accept** and **false reject**

		$g$	
		+1	-1
$f$	+1	no error	<b>false reject</b>
	-1	<b>false accept</b>	no error

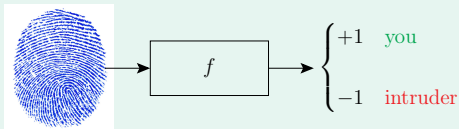
**Cost matrix**

		$g$	
		+1	-1
$f$	+1	0	<b>10</b>
	-1	<b>1</b>	0

- supermarket: fingerprint for discount
- false reject: very unhappy customer, lose future business**
- false accept: give away a minor discount, intruder left fingerprint :-)**

# Fingerprint Verification for CIA

## Fingerprint Verification



two types of error: false accept and false reject

		$g$	
		+1	-1
$f$	+1	no error	false reject
	-1	false accept	no error

		$g$	
		+1	-1
$f$	+1	0	1
	-1	1000	0

- CIA: fingerprint for entrance
- false accept: **very serious consequences!**
- false reject: unhappy employee, but so what? :-)

# Take-home Message for Now

$\text{err}$  is **application/user-dependent**

Although the market and CIA examples are both binary classifications, they use completely different loss functions.

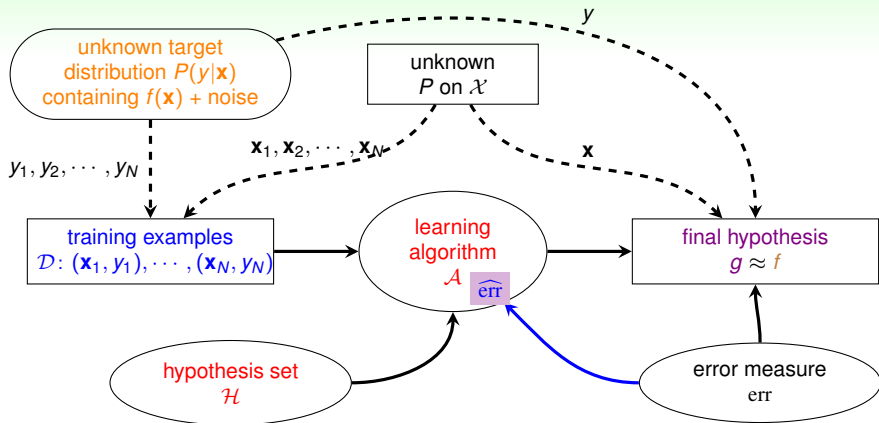
## Algorithmic Error Measures $\widehat{\text{err}}$

- true: just  $\text{err}$  ← The real, unknown loss function
- plausible:
  - 0/1: minimum 'flipping noise'—NP-hard to optimize, remember? :-)
  - squared: minimum Gaussian noise
- friendly: easy to optimize for  $\mathcal{A}$ 
  - closed-form solution
  - convex objective function

The loss function we use

$\widehat{\text{err}}$ : more in next lectures

# Learning Flow with Algorithmic Error Measure



err: application goal;  
 $\widehat{\text{err}}$ : a key part of many  $\mathcal{A}$

## Fun Time

Consider err below for CIA. What is  $E_{\text{in}}(g)$  when using this err?

		$g$	
		+1	-1
$f$	+1	0	1
	-1	1000	0

①  $\frac{1}{N} \sum_{n=1}^N \mathbb{I}[y_n \neq g(\mathbf{x}_n)]$   
 ②  $\frac{1}{N} \left( \sum_{y_n=+1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] + 1000 \sum_{y_n=-1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] \right)$   
 ③  $\frac{1}{N} \left( \sum_{y_n=+1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] - 1000 \sum_{y_n=-1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] \right)$   
 ④  $\frac{1}{N} \left( 1000 \sum_{y_n=+1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] + \sum_{y_n=-1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] \right)$

## Fun Time

Consider err below for CIA. What is  $E_{\text{in}}(g)$  when using this err?

		$g$	
		+1	-1
$f$	+1	0	1
	-1	1000	0

①  $\frac{1}{N} \sum_{n=1}^N \mathbb{I}[y_n \neq g(\mathbf{x}_n)]$   
 ②  $\frac{1}{N} \left( \sum_{y_n=+1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] + 1000 \sum_{y_n=-1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] \right)$   
 ③  $\frac{1}{N} \left( \sum_{y_n=+1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] - 1000 \sum_{y_n=-1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] \right)$   
 ④  $\frac{1}{N} \left( 1000 \sum_{y_n=+1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] + \sum_{y_n=-1} \mathbb{I}[y_n \neq g(\mathbf{x}_n)] \right)$

Reference Answer: ②

When  $y_n = -1$ , the false positive made on such  $(\mathbf{x}_n, y_n)$  is penalized 1000 times more!

# Weighted Classification

## CIA Cost (Error, Loss, ...) Matrix

		$h(\mathbf{x})$	
		+1	-1
$y$	+1	0	1
	-1	1000	0

## out-of-sample

$$E_{\text{out}}(h) = \mathbb{E}_{(\mathbf{x}, y) \sim P} \left\{ \begin{array}{ll} 1 & \text{if } y = +1 \\ 1000 & \text{if } y = -1 \end{array} \right\} \cdot \mathbb{I}[y \neq h(\mathbf{x})]$$

## in-sample

$$E_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^N \left\{ \begin{array}{ll} 1 & \text{if } y_n = +1 \\ 1000 & \text{if } y_n = -1 \end{array} \right\} \cdot \mathbb{I}[y_n \neq h(\mathbf{x}_n)]$$

weighted classification:

**different 'weight' for different  $(\mathbf{x}, y)$**



# Minimizing $E_{\text{in}}$ for Weighted Classification

$$E_{\text{in}}^w(h) = \frac{1}{N} \sum_{n=1}^N \left\{ \begin{array}{ll} 1 & \text{if } y_n = +1 \\ 1000 & \text{if } y_n = -1 \end{array} \right\} \cdot \mathbb{I}[y_n \neq h(\mathbf{x}_n)]$$

## Naïve Thoughts

- PLA: **doesn't matter if linear separable. :-)**
- pocket: modify **pocket-replacement rule**  
—if  $\mathbf{w}_{t+1}$  reaches smaller  $E_{\text{in}}^w$  than  $\hat{\mathbf{w}}$ , replace  $\hat{\mathbf{w}}$  by  $\mathbf{w}_{t+1}$

Instead of just looking at the amount of misclassification,  
now we use weighted classification to evaluate which  $\mathbf{w}$  is better.

pocket: some guarantee on  $E_{\text{in}}^{0/1}$ ;  
modified pocket: similar guarantee on  $E_{\text{in}}^w$ ?

Does the way we update  $\mathbf{w}$  guarantee that  $E_{\text{in}}^w$  gets smaller and smaller?

# Systematic Route: Connect $E_{\text{in}}^w$ and $E_{\text{in}}^{0/1}$

## original problem

		$h(\mathbf{x})$	
		+1	-1
$y$	+1	0	1
	-1	1000	0

$\mathcal{D}$ :

$(\mathbf{x}_1, +1)$   
 $(\mathbf{x}_2, -1)$   
 $(\mathbf{x}_3, -1)$   
 $\dots$   
 $(\mathbf{x}_{N-1}, +1)$   
 $(\mathbf{x}_N, +1)$

## equivalent problem

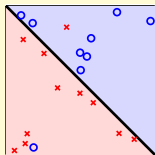
		$h(\mathbf{x})$	
		+1	-1
$y$	+1	0	1
	-1	1	0

$(\mathbf{x}_1, +1)$   
 $(\mathbf{x}_2, -1), (\mathbf{x}_2, -1), \dots, (\mathbf{x}_2, -1)$   
 $(\mathbf{x}_3, -1), (\mathbf{x}_3, -1), \dots, (\mathbf{x}_3, -1)$   
 $\dots$   
 $(\mathbf{x}_{N-1}, +1)$   
 $(\mathbf{x}_N, +1)$

after **copying  $-1$  examples 1000 times**,

$E_{\text{in}}^w$  for LHS  $\equiv E_{\text{in}}^{0/1}$  for RHS!

# Weighted Pocket Algorithm



		$h(\mathbf{x})$	
		+1	-1
$y$	+1	0	1
	-1	1000	0

using 'virtual copying', **weighted pocket algorithm** include:

- weighted PLA:  
randomly check **-1 example** mistakes with **1000** times more probability  
The probability of -1 being checked changes from being the same as +1 to 1000 times greater than +1
- weighted pocket replacement:  
if  $\mathbf{w}_{t+1}$  reaches smaller  $E_{in}^w$  than  $\hat{\mathbf{w}}$ , replace  $\hat{\mathbf{w}}$  by  $\mathbf{w}_{t+1}$

systematic route (called 'reduction'):  
**can be applied to many other algorithms!**

# Fun Time

Consider the CIA cost matrix. If there are 10 examples with  $y_n = -1$  (intruder) and 999,990 examples with  $y_n = +1$  (you). What would  $E_{\text{in}}^w(h)$  be for a constant  $h(\mathbf{x})$  that always returns  $+1$ ?

		$h(\mathbf{x})$	
		+1	-1
$y$	+1	0	1
	-1	1000	0

- ① 0.001
- ② 0.01
- ③ 0.1
- ④ 1

# Fun Time

Consider the CIA cost matrix. If there are 10 examples with  $y_n = -1$  (intruder) and 999,990 examples with  $y_n = +1$  (you). What would  $E_{\text{in}}^w(h)$  be for a constant  $h(\mathbf{x})$  that always returns  $+1$ ?

		$h(\mathbf{x})$	
		+1	-1
$y$	+1	0	1
	-1	1000	0

- ① 0.001
- ② 0.01
- ③ 0.1
- ④ 1

Reference Answer: ②

While the quiz is a simple evaluation, it is not uncommon that the data is very **unbalanced** for such an application. Properly 'setting' the weights can be used to avoid the lazy constant prediction.

$$\frac{10 \times 1,000 + 999,990 \times 0}{1,000,000}$$

# Summary

- 1 When Can Machines Learn?
- 2 **Why** Can Machines Learn?

## Lecture 7: The VC Dimension

## Lecture 8: Noise and Error

- Noise and Probabilistic Target  
**can replace  $f(x)$  by  $P(y|x)$**
- Error Measure  
**affect 'ideal' target**
- Algorithmic Error Measure  
**user-dependent  $\implies$  plausible or friendly**
- Weighted Classification  
**easily done by virtual 'example copying'**

- **next: more algorithms, please? :-)**

- 3 How Can Machines Learn?
- 4 How Can Machines Learn Better?