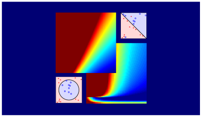
Machine Learning Foundations
(機器學習基石)



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

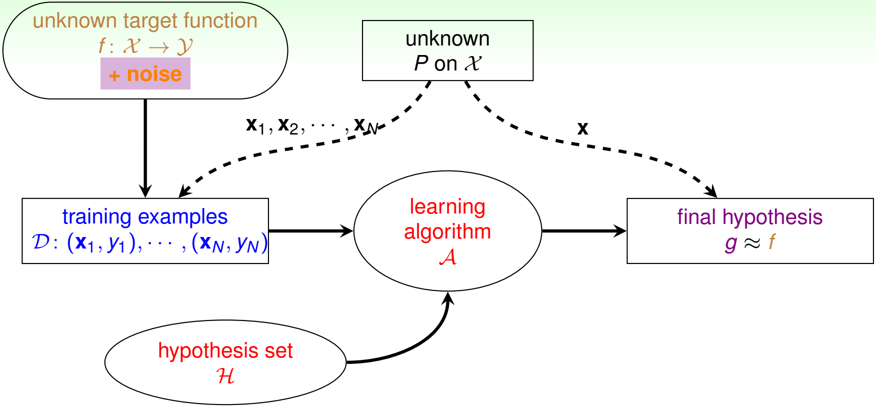
learning happens
if **finite** d_{VC}, **large** N, and **low** E_{in}

Lecture 8: Noise and Error

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

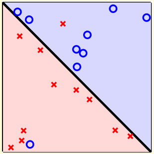
3. How Can Machines Learn?
4. How Can Machines Learn Better?

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? $\{\text{no}(-1), \text{yes}(+1)\}$

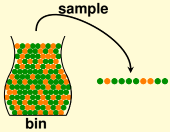
but more!

- **noise in y**: good customer, 'mislabeled' as bad?
- **noise in y**: same customers, different labels?
- **noise in $\mathbf{x}$**: 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 $[\![f(\mathbf{x}) \neq h(\mathbf{x})]\!]$

'probabilistic' (noisy) marbles

- marble $\mathbf{x} \sim P(\mathbf{x})$
- probabilistic color $[\![y \neq h(\mathbf{x})]\!]$ with $y \sim P(y|\mathbf{x})$

same nature: can estimate $\mathbb{P}$[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(x) but P(y|x)

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 $y \neq f(\mathbf{x})$

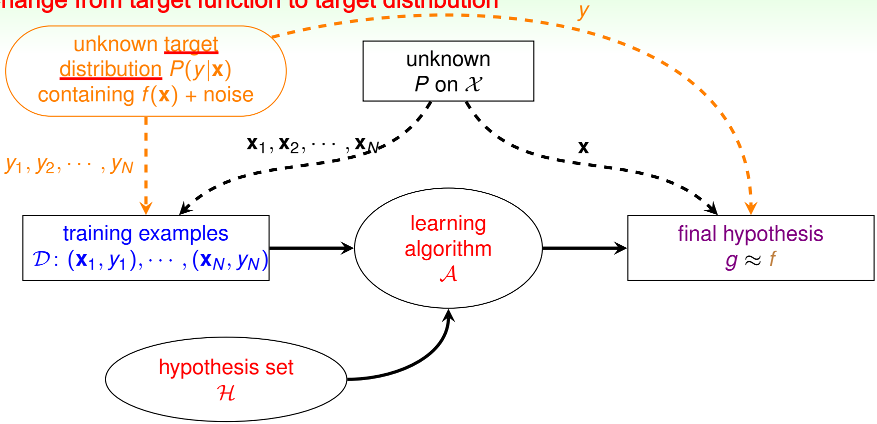
goal of learning:

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 → (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

$$E_{\text{out}}(g) = \underset{\mathbf{x}\sim P}{\mathcal{E}} [\![g(\mathbf{x}) \neq f(\mathbf{x})]\!]$$

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

classification error $[\![\ldots]\!]$:
often also called **'0/1 error'**

Pointwise Error Measure

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

$$E_{\text{out}}(g) = \underset{\mathbf{x}\sim P}{\mathcal{E}} \underbrace{[\![g(\mathbf{x}) \neq f(\mathbf{x})]\!]}_{\mathrm{err}(g(\mathbf{x}), f(\mathbf{x}))}$$

—err: called **pointwise error measure**

in-sample

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

out-of-sample

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

will mainly consider pointwise err for simplicity

Two Important Pointwise Error Measures

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

prediction ground truth

0/1 error

$$\mathrm{err}(\tilde{y}, y) = [\![\tilde{y} \neq y]\!]$$

- correct or incorrect?
- often for classification

squared error

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

- 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 $f(\mathbf{x})$

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

$$\mathrm{err}(\tilde{y}, y) = [\![\tilde{y} \neq y]\!]$$

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

Please guess carefully

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

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

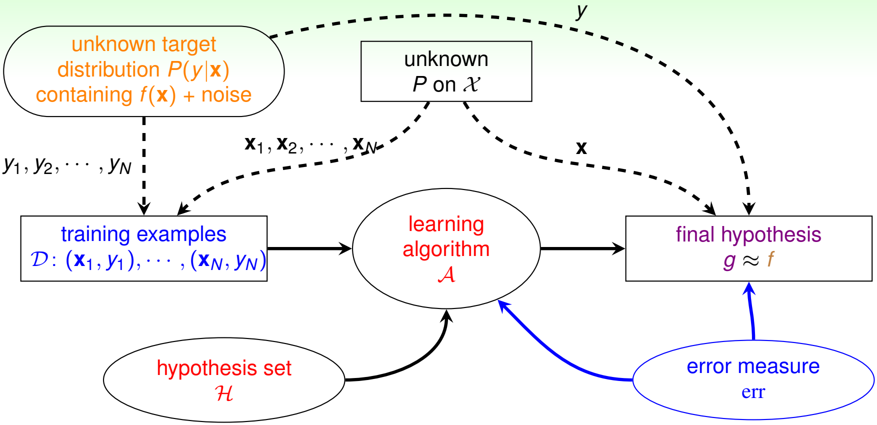
$$\mathrm{err}(\tilde{y}, y) = (\tilde{y} - y)^2$$

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

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

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

Learning Flow with Error Measure



extended VC theory/'philosophy'
works for most $\mathcal{H}$ and $\mathbf{err}$

Fun Time

Consider the following $P(y|\mathbf{x})$ and $\mathrm{err}(\tilde{y}, y) = |\tilde{y} - y|$. 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. $2.5 =$ average within $\mathcal{Y} = \{1, 2, 3, 4\}$
2. $2.85 =$ weighted mean from $P(y|\mathbf{x})$
3. $3 =$ weighted median from $P(y|\mathbf{x})$
4. $4 = \mathrm{argmax}\, P(y|\mathbf{x})$

Fun Time

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

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

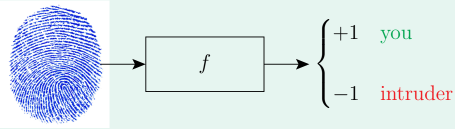
1. $2.5 =$ average within $\mathcal{Y} = \{1, 2, 3, 4\}$ = 1
2. $2.85 =$ weighted mean from $P(y|\mathbf{x})$ = 0.965
3. $3 =$ weighted median from $P(y|\mathbf{x})$ = 0.95
4. $4 = \mathrm{argmax}\, P(y|\mathbf{x})$ = 1.15

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 False positive	no error	

0/1 error penalizes both types equally

Fingerprint Verification for Supermarket

Fingerprint Verification

$$\rightarrow \boxed{\quad f \quad} \rightarrow \begin{cases} +1 & \text{you} \\ -1 & \text{intruder} \end{cases}$$

two types of error: false accept and false reject

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

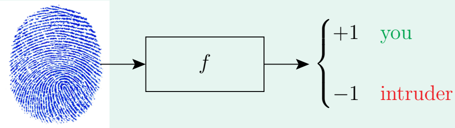
Cost matrix

		g: +1	g: -1
f	+1	0	10
	-1	1	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	g: -1
f	+1	no error	false reject
	-1	false accept	no error

		g: +1	g: -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

err is **application/user-dependent**

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

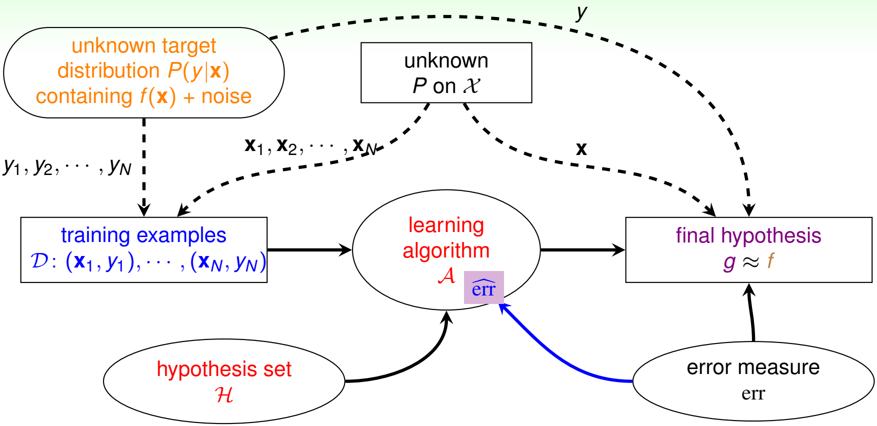
Algorithmic Error Measures $\widehat{\mathrm{err}}$

- true: just 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

$\widehat{\mathrm{err}}$: 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	g: -1
f	+1	0	1
	-1	1000	0

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

Fun Time

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

		g: +1	g: -1
f	+1	0	1
	-1	1000	0

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

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

out-of-sample

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

in-sample

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

weighted classification:
different 'weight' for different $(\mathbf{x}, y)$

Minimizing E_{in} for Weighted Classification

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

Naïve Thoughts

- PLA: **doesn't matter if linear separable. :-)**
- pocket: modify **pocket-replacement rule**
  —if $\mathbf{w}_{t+1}$ reaches smaller E_{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 w is better.

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

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

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

original problem

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

$\mathcal{D}$:

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

equivalent problem

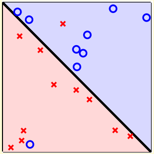
		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -1
y	+1	0	1
	-1	1	0

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

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

E_{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 $E_{\text{in}}^{\text{w}}(h)$ be for a constant $h(\mathbf{x})$ that always returns $+1$?

		$h(\mathbf{x})$ +1	$h(\mathbf{x})$ -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 $E_{\text{in}}^{\text{w}}(h)$ 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 $f(\mathbf{x})$ by $P(y|\mathbf{x})$
- Error Measure
  affect 'ideal' target
- Algorithmic Error Measure
  user-dependent $\Longrightarrow$ 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?