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

unknown target function \( f: \mathcal{X} \rightarrow \mathcal{Y} \)

training examples \( \mathcal{D}: (x_1, y_1), \ldots, (x_N, y_N) \)

learning algorithm \( \mathcal{A} \)

final hypothesis \( g \approx f \)

what if there is noise?
briefly introduced **noise** before **pocket** algorithm

<table>
<thead>
<tr>
<th>age</th>
<th>23 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>gender</td>
<td>female</td>
</tr>
<tr>
<td>annual salary</td>
<td>NTD 1,000,000</td>
</tr>
<tr>
<td>year in residence</td>
<td>1 year</td>
</tr>
<tr>
<td>year in job</td>
<td>0.5 year</td>
</tr>
<tr>
<td>current debt</td>
<td>200,000</td>
</tr>
</tbody>
</table>

**credit?** \{no(−1), yes(+1)\}

but more!

- **noise in** $y$: good customer, ‘mislabeled’ 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 $x \sim P(x)$
- deterministic color $[f(x) \neq h(x)]$

'Probabilistic' (noisy) marbles
- marble $x \sim P(x)$
- probabilistic color $[y \neq h(x)]$ with $y \sim P(y | x)$

Same nature: can estimate $P[\text{orange}]$ if $i.i.d.$

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

$(x, y) \overset{i.i.d.}{\sim} P(x, y)$
Target Distribution $P(y|x)$

can be viewed as ‘ideal mini-target’ + noise, e.g.

- $P(\circ|x) = 0.7$, $P(\times|x) = 0.3$
- ideal mini-target $f(x) = \circ$
- ‘flipping’ noise level = 0.3

- deterministic target $f$: special case of target distribution
  - $P(y|x) = 1$ for $y = f(x)$
  - $P(y|x) = 0$ for $y \neq f(x)$

**goal of learning:**

predict **ideal mini-target (w.r.t. $P(y|x)$)**
on **often-seen inputs (w.r.t. $P(x)$)**
The New Learning Flow

unknown target distribution $P(y|x)$ containing $f(x) + \text{noise}$

$y_1, y_2, \cdots, y_N$

training examples $D: (x_1, y_1), \cdots, (x_N, y_N)$

learning algorithm $A$

final hypothesis $g \approx f$

hypothesis set $\mathcal{H}$

$\text{VC still works, pocket algorithm explained :-)}$
Let’s revisit PLA/pocket. Which of the following claim is true?

1. In practice, we should try to compute if $D$ is linear separable before deciding to use PLA.
2. If we know that $D$ is not linear separable, then the target function $f$ must not be a linear function.
3. If we know that $D$ 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 $D$ is linear separable, we shall know $w^*$ and then there is no need to use PLA. 2. What about noise? 3. What about ‘sampling luck’? :-)

Hsuan-Tien Lin (NTU CSIE)
Error Measure

- final hypothesis
  \[ g \approx f \]

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

  \[ E_{\text{out}}(g) = \mathbb{E}_{x \sim P} \left[ g(x) \neq f(x) \right] \]

- more generally, error measure \( E(g, f) \)

- naturally considered
  - out-of-sample: averaged over unknown \( x \)
  - pointwise: evaluated on one \( x \)
  - classification: \([\text{prediction} \neq \text{target}]\)

classification error \([\ldots]\): often also called ‘0/1 error’
Pointwise Error Measure

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

$$E_{out}(g) = \mathcal{E}_{x \sim \mathcal{P}} \left[ g(x) \neq f(x) \right]$$

$\text{err}$: called pointwise error measure

**in-sample**

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

**out-of-sample**

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

will mainly consider pointwise $\text{err}$ for simplicity
Two Important Pointwise Error Measures

$$\text{err} \left( \begin{array}{c} g(x), f(x) \\ \tilde{y}, y \end{array} \right)$$

0/1 error
$$\text{err}(\tilde{y}, y) = \left\lceil \tilde{y} \neq y \right\rceil$$
- 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 $\text{err}$ ‘guide’ learning?
**Ideal Mini-Target**

The interplay between noise and error:

\[ P(y|x) \text{ and } \text{err} \text{ define ideal mini-target } f(x) \]

\[
P(y = 1|x) = 0.2, \ P(y = 2|x) = 0.7, \ P(y = 3|x) = 0.1
\]

\[
\text{err}(\tilde{y}, y) = \left[ \tilde{y} \neq y \right]
\]

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

\[
f(x) = \arg\max_{y \in Y} P(y|x)
\]

\[
f(x) = \sum_{y \in Y} y \cdot P(y|x)
\]
Learning Flow with Error Measure

- **unknown target distribution** $P(y|x)$ containing $f(x) +$ noise
- **training examples** $D: (x_1, y_1), \cdots, (x_N, y_N)$
- **learning algorithm** $\mathcal{A}$
- **final hypothesis** $g \approx f$
- **hypothesis set** $\mathcal{H}$
- **error measure** $\text{err}$

Extended VC theory/‘philosophy’

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

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

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

Reference Answer: $\circled{3}$

For the ‘absolute error’, the weighted median provably results in the minimum average $\text{err}$.
Choice of Error Measure

Fingerprint Verification

Two types of error: false accept and false reject

<table>
<thead>
<tr>
<th></th>
<th>+1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f$</td>
<td>+1</td>
<td>no error</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>false accept</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>+1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$g$</td>
<td>+1</td>
<td>false reject</td>
</tr>
<tr>
<td></td>
<td>-1</td>
<td>no error</td>
</tr>
</tbody>
</table>

0/1 error penalizes both types equally
### Fingerprint Verification for Supermarket

#### Algorithmic Error Measure

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

<table>
<thead>
<tr>
<th>$f$</th>
<th>$g$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>+1</td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
</tr>
</tbody>
</table>

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

Fingerprint Verification:

- Supermarket: fingerprint for discount
Fingerprint Verification for CIA

Noise and Error
Algorithmic Error Measure

Fingerprint Verification

two types of error: false accept and false reject

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

\[
\begin{array}{c|cc|c|cc}
  & +1 & -1 & & +1 & -1 \\
  f & +1 & \text{no error} & \text{false reject} & 0 & 1 \\
  & -1 & \text{false accept} & \text{no error} & 1000 & 0 \\
\end{array}
\]
Take-home Message for Now

\[ \textit{err} \text{ is application/user-dependent} \]

**Algorithmic Error Measures \( \hat{\text{err}} \)**

- **true:** just \( \text{err} \)
- **plausible:**
  - 0/1: minimum ‘flipping noise’—NP-hard to optimize, remember? :-)
  - squared: minimum Gaussian noise
- **friendly:** easy to optimize for \( A \)
  - closed-form solution
  - convex objective function

\( \hat{\text{err}} \): more in next lectures
Learning Flow with Algorithmic Error Measure

unknown target distribution \( P(y|x) \) containing \( f(x) + \text{noise} \)

\( y_1, y_2, \cdots, y_N \)

\( x_1, x_2, \cdots, x_N \)

training examples \( D: (x_1, y_1), \cdots, (x_N, y_N) \)

unknown \( P \) on \( \mathcal{X} \)

learning algorithm \( \mathcal{A} \)

final hypothesis \( g \approx f \)

hypothesis set \( \mathcal{H} \)

error measure \( \hat{err} \)

err: application goal;
\( \hat{err} \): a key part of many \( \mathcal{A} \)
Consider $err$ below for CIA. What is $E_{in}(g)$ when using this $err$?

<table>
<thead>
<tr>
<th>$g$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

1. $\frac{1}{N} \sum_{n=1}^{N} [y_n \neq g(x_n)]$

2. $\frac{1}{N} \left( \sum_{y_n=-1} [y_n \neq g(x_n)] + 1000 \sum_{y_n=+1} [y_n \neq g(x_n)] \right)$

3. $\frac{1}{N} \left( \sum_{y_n=-1} [y_n \neq g(x_n)] - 1000 \sum_{y_n=+1} [y_n \neq g(x_n)] \right)$

4. $\frac{1}{N} \left( 1000 \sum_{y_n=-1} [y_n \neq g(x_n)] + \sum_{y_n=+1} [y_n \neq g(x_n)] \right)$

Reference Answer: 2

When $y_n = -1$, the false positive made on such $(x_n, y_n)$ is penalized 1000 times more!
### Weighted Classification

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

<table>
<thead>
<tr>
<th></th>
<th>$h(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>+1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y$</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>1000</td>
<td>0</td>
</tr>
</tbody>
</table>

### Out-of-sample

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

### In-sample

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

**weighted classification:**

**different ‘weight’ for different** $(x, y)$
Minimizing $E_{in}$ for Weighted Classification

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

Naïve Thoughts

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

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

#### Original Problem

<table>
<thead>
<tr>
<th>$y$</th>
<th>$h(x)$</th>
<th>$+1$</th>
<th>$-1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+1$</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$-1$</td>
<td>0</td>
<td>1000</td>
<td>0</td>
</tr>
</tbody>
</table>

- $(x_1, +1)$
- $(x_2, -1)$
- $(x_3, -1)$
- $(x_{N-1}, +1)$
- $(x_N, +1)$

**$D$:**

- $(x_1, +1)$
- $(x_2, -1)$, $(x_2, -1)$, ..., $(x_2, -1)$
- $(x_3, -1)$, $(x_3, -1)$, ..., $(x_3, -1)$
- $(x_{N-1}, +1)$
- $(x_N, +1)$

#### Equivalent Problem

<table>
<thead>
<tr>
<th>$y$</th>
<th>$h(x)$</th>
<th>$+1$</th>
<th>$-1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$+1$</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>$-1$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

- $(x_1, +1)$
- $(x_2, -1)$, $(x_2, -1)$, ..., $(x_2, -1)$
- $(x_3, -1)$, $(x_3, -1)$, ..., $(x_3, -1)$
- $(x_{N-1}, +1)$
- $(x_N, +1)$

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

$E_{\text{in}}^w$ for LHS $\equiv E_{\text{in}}^{0/1}$ for RHS!
using ‘virtual copying’, **weighted pocket algorithm** include:

- **weighted PLA:** randomly check \(-1\) example mistakes with 1000 times more probability
- **weighted pocket replacement:** if \(w_{t+1}\) reaches smaller \(E_{in}^w\) than \(\hat{w}\), replace \(\hat{w}\) by \(w_{t+1}\)

**systematic route (called ‘reduction’):** can be applied to many other algorithms!
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 $E^w_{in}(h)$ be for a constant $h(x)$ that always returns $+1$?

<table>
<thead>
<tr>
<th>$y$</th>
<th>$h(x)$</th>
<th>$E^w_{in}(h)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1</td>
<td>0</td>
<td>0.001</td>
</tr>
<tr>
<td>-1</td>
<td>1000</td>
<td>0.01</td>
</tr>
<tr>
<td>+1</td>
<td>1</td>
<td>0.1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

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.
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?