Error Measurement
The learning diagram

**UNKNOWN TARGET FUNCTION**

\[ f: \mathcal{X} \rightarrow \mathcal{Y} \]

**TRAINING EXAMPLES**

\[ (x_1, y_1), \ldots, (x_N, y_N) \]

**LEARNING ALGORITHM**

\[ A \]

**HYPOTHESIS SET**

\[ \mathcal{H} \]

**FINAL HYPOTHESIS**

\[ g: \mathcal{X} \rightarrow \mathcal{Y} \]

**PROBABILITY DISTRIBUTION**

\[ P \text{ on } \mathcal{X} \]

\[ \xrightarrow{x_1, \ldots, x_N} \]
What does “$h \approx f$” mean?

Define an error measure: $E(h, f)$
What does “$h \approx f$” mean?

Define an error measure: $E(h, f)$

Almost always pairwise definition: (define on each $x$)

$$e(h(x), f(x)) \text{ (loss on } x)$$
What does “$h \approx f$” mean?

Define an error measure: $E(h, f)$

Almost always pairwise definition: (define on each $x$)

$$e(h(x), f(x)) \quad \text{(loss on } x)$$

Examples:

Square error: $e(h(x), f(x)) = (h(x) - f(x))^2$

Binary error: $e(h(x), f(x)) = [h(x) \neq f(x)]$
From pointwise to overall

- Overall error $E(h, f) = \text{average of pointwise errors } e(h(x), f(x))$
Overall error $E(h, f) = \text{average of pointwise errors } e(h(x), f(x))$

In-sample error (training error):

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

Out-of-sample error (generalization error):

$$E_{\text{out}}(h) = E_x[e(h(x), f(x))]$$
Learning diagram (with error measurement)

UNKNOWN TARGET FUNCTION
\[ f: X \rightarrow Y \]

TRAINING EXAMPLES
\[ (x_1, y_1), \ldots, (x_N, y_N) \]

LEARNING ALGORITHM
\[ \mathcal{A} \]

HYPOTHESIS SET
\[ \mathcal{H} \]

PROBABILITY DISTRIBUTION
\[ P \text{ on } X \]

\[ x_1, \ldots, x_N \]

FINAL HYPOTHESIS
\[ g: X \rightarrow Y \]

\[ g(x) \approx f(x) \]
Choice of Error Measure

Two types of error: false accept and false reject.
Choice of Error Measure

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>+1</td>
<td>no error</td>
<td>false accept</td>
</tr>
<tr>
<td>-1</td>
<td>false reject</td>
<td>no error</td>
</tr>
</tbody>
</table>
Choice of Error Measure

Application: Supermarket verifies fingerprint for discounts
- False reject is costly: customer gets annoyed
- False accept is minor: just gave away a free discount
Choice of Error Measure

Application: Supermarket verifies fingerprint for discounts
- False reject is costly: customer gets annoyed
- False accept is minor: just gave away a free discount
- Define the error measure:

\[
\begin{array}{c|cc}
& +1 & -1 \\
+1 & no error & 1 \\
h & f & \\
-1 & 10 & no error
\end{array}
\]
Choice of Error Measure

Application: CIA security
- False reject is tolerable
- False accept is disaster
- Define the error measure:

\[
\begin{array}{c|c|c}
\text{h} & +1 & -1 \\
+1 & \text{no error} & \text{1000} \\
-1 & 1 & \text{no error}
\end{array}
\]
Take-home lesson

- The error measure is application/user-dependent
Take-home lesson

- The error measure is application/user-dependent
- Plausible:
  - 0/1: minimum mis-classification
  - regression: square error
- Friendly:
  - Closed-form solution (square loss)
  - Convex objective function (logistic loss)
Weighted Classification

Out-of-sample error:

\[ E_{\text{out}}(h) = E_{x,y}[c(y) \cdot e(h(x), y)] \]

Class-dependent weight: \( c(y) = \begin{cases} 
1 & \text{if } y = +1 \\
1000 & \text{if } y = -1 
\end{cases} \)
Weighted Classification

- Out-of-sample error:

\[ E_{out}(h) = E_{x,y} [c(y) \cdot e(h(x), y)] \]

Class-dependent weight: \( c(y) = \begin{cases} 1 & \text{if } y = +1 \\ 1000 & \text{if } y = -1 \end{cases} \)

- In-sample error:

\[ E_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} c(y_n) \cdot e(h(x_n), y_n) \]
How to solve it?

- In general (any solver):
  Augment the dataset by duplicating each $-1$ example 1000 times.
  Then solve the unweighted version.

- Table:

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

- Dataset $D$:

- Original problem:

<table>
<thead>
<tr>
<th>$x_i$, $y$</th>
<th>$h(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x₁, +1)</td>
<td>+1</td>
</tr>
<tr>
<td>(x₂, −1)</td>
<td>-1</td>
</tr>
<tr>
<td>(x₃, −1)</td>
<td>-1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>(xₙ₋₁, +1)</td>
<td>+1</td>
</tr>
<tr>
<td>(xₙ, +1)</td>
<td>+1</td>
</tr>
</tbody>
</table>

- Equivalent problem:

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

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

<table>
<thead>
<tr>
<th>$x_i$, $y$</th>
<th>$h(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x₁, +1)</td>
<td>+1</td>
</tr>
<tr>
<td>(x₂, −1)</td>
<td>-1</td>
</tr>
<tr>
<td>(x₃, −1)</td>
<td>-1</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>(xₙ₋₁, +1)</td>
<td>+1</td>
</tr>
<tr>
<td>(xₙ, +1)</td>
<td>+1</td>
</tr>
</tbody>
</table>
How to solve it?

- Require much more memory
- For **most algorithms**, we can incorporate the weight in the algorithm without duplicating data

In Stochastic Gradient Descent (SGD):

**Approach I:**
Update the model according to the weight:

\[ w ← w - η_t · c(y_n) \nabla e(w^T x_n, y_n) \]

**Approach II:**
Change the sample rate
Increase the probability of choosing \(-1\) examples by 1000 times
How to solve it?

- Require much more memory
- For **most algorithms**, we can incorporate the weight in the algorithm without duplicating data
- In Stochastic Gradient Descent (SGD):
  - Approach I:
    Update the model according to the weight:
    \[ w \leftarrow w - \eta^t \cdot c(y_n) \nabla e(w^T x_n, y_n) \]
  - Approach II:
    Change the sample rate
    Increase the probability of choosing \(-1\) examples by 1000 times
Noisy Targets

- The “target function” is not always a 1-1 function
- Consider the credit card approval:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>age</td>
<td>23</td>
</tr>
<tr>
<td>Annual salary</td>
<td>30,000</td>
</tr>
<tr>
<td>Year in residence</td>
<td>1</td>
</tr>
<tr>
<td>Year in job</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Two **identical** customers $\rightarrow$ two **different** behaviors
 Instead of \( y = f(x) \), we use target distribution

\[
P(y | x)
\]
Instead of $y = f(x)$, we use target distribution

$$P(y \mid x)$$

$(x, y)$ is now generated by the joint distribution

$$P(x)P(y \mid x)$$
Instead of \( y = f(x) \), we use target distribution

\[ P(y \mid x) \]

\((x, y)\) is now generated by the joint distribution

\[ P(x)P(y \mid x) \]

Noisy target \( \approx \) deterministic target \( f(x) = E(y \mid x) \) plus noise \( y - f(x) \)
Learning diagram with noisy target

- **Unknown Target Distribution**: $P(y \mid x)$
  - Target function $f: X \rightarrow Y$ plus noise

- **Training Examples**: $(x_1, y_1), \ldots, (x_N, y_N)$

- **Learning Algorithm**: $\mathcal{A}$

- **Error Measure**: $e()$

- **Probability Distribution**: $P$ on $X$

- **Final Hypothesis**: $g: X \rightarrow Y$

- **Hypothesis Set**: $\mathcal{H}$
Distinction between $P(y \mid x)$ and $P(x)$

- The target distribution $P(y \mid x)$ is what we are trying to learn.
- The input distribution $P(x)$ quantifies relative importance of $x$.
Preamble to the theory
What we know so far?

- It is likely that

\[ E_{out}(g) \approx E_{in}(g) \]
What we know so far?

- It is likely that
  \[ E_{out}(g) \approx E_{in}(g) \]

- Not yet learning
What we know so far?

- It is likely that
  \[ E_{\text{out}}(g) \approx E_{\text{in}}(g) \]

- Not yet learning

- For machine learning, we need \( g \approx f \), which means
  \[ E_{\text{out}}(g) \approx 0 \]
The 2 questions of learning

- $E_{out} \approx 0$ is achieved through:

  $$E_{out}(g) \approx E_{in}(g) \quad \text{and} \quad E_{in}(g) \approx 0$$
The 2 questions of learning

- \( E_{\text{out}} \approx 0 \) is achieved through:
  \[
  E_{\text{out}}(g) \approx E_{\text{in}}(g) \quad \text{and} \quad E_{\text{in}}(g) \approx 0
  \]

- Learning is thus split into 2 questions:
  - Can we make sure that \( E_{\text{out}}(g) \) is close enough to \( E_{\text{in}}(g) \)?
    Hoeffding’s inequality
  - Can we make \( E_{\text{in}}(g) \) small?
    Optimization
What the theory will achieve

- Currently we only know
  \[ P[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2M e^{-2\epsilon^2 N} \]

- Show \( E_{in}(g) \approx E_{out}(g) \) for infinite \( M \) (number of hypothesis)
What the theory will achieve

- Currently we only know

\[ P[|E_{in}(g) - E_{out}(g)| > \epsilon] \leq 2Me^{-2\epsilon^2N} \]

- Show \( E_{in}(g) \approx E_{out}(g) \) for infinite \( M \) (number of hypothesis)
Conclusions

- Next class: LFD 2.1, 2.2, 2.3 (VC-dimension)

Questions?