Lecture 8: Adaptive Boosting

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)
Adaptive Boosting

Roadmap

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 7: Blending and Bagging

- blending known diverse hypotheses uniformly, linearly, or even non-linearly; obtaining diverse hypotheses from bootstrapped data

Lecture 8: Adaptive Boosting

- Motivation of Boosting
- Diversity by Re-weighting
- Adaptive Boosting Algorithm
- Adaptive Boosting in Action

3. Distilling Implicit Features: Extraction Models
Apple Recognition Problem

• is this a picture of an apple?
• say, want to teach a class of 6 year olds

Hsuan-Tien Lin (NTU CSIE)
Apple Recognition Problem

- is this a picture of an apple?
- say, want to teach a class of **6 year olds**
- gather photos under CC-BY-2.0 license on Flicker (thanks to the authors below!)

(APAL stands for Apple and Pear Australia Ltd)

Dan Foy
https://flic.kr/p/jNQ55

APAL
https://flic.kr/p/jzP1VB

adrianbartel
https://flic.kr/p/bdy2hZ

ANdrzej cH.
https://flic.kr/p/51DKA8

Stuart Webster
https://flic.kr/p/9C3Ybd

nachans
https://flic.kr/p/9XD7Ag

APAL
https://flic.kr/p/jzRe4u

Jo Jakeman
https://flic.kr/p/7jwtGp

APAL
https://flic.kr/p/jzPYNr

APAL
https://flic.kr/p/jzScif
Apple Recognition Problem

- is this a picture of an apple?
- say, want to teach a class of 6 year olds
- gather photos under CC-BY-2.0 license on Flicker (thanks to the authors below!)

Mr. Roboto.
https://flic.kr/p/i5BN85

Richard North
https://flic.kr/p/bHhPkB

Richard North
https://flic.kr/p/d8tGou

Emilian Vicol
https://flic.kr/p/bpmGXW

Robert McQueen
https://flic.kr/p/pZv1Mf

Crystal
https://flic.kr/p/kaPYp

jfh686
https://flic.kr/p/6vjRFH

skyseeker
https://flic.kr/p/2MynV

Janet Hudson
https://flic.kr/p/7QDBbm

Rennett Stowe
https://flic.kr/p/agmnrk
Our Fruit Class Begins

- Teacher: Please look at the pictures of apples and non-apples below. Based on those pictures, how would you describe an apple? Michael?
Teacher: Please look at the pictures of apples and non-apples below. Based on those pictures, how would you describe an apple? Michael?

Michael: I think apples are **circular**.
Our Fruit Class Begins

- Teacher: Please look at the pictures of apples and non-apples below. Based on those pictures, how would you describe an apple? Michael?
- Michael: I think apples are *circular*.

(Class): Apples are *circular*.
Teacher: Being circular is a good feature for the apples. However, if you only say circular, you could make several mistakes. What else can we say for an apple? Tina?
Our Fruit Class Continues

- Teacher: Being circular is a good feature for the apples. However, if you only say circular, you could make several mistakes. What else can we say for an apple? Tina?
- Tina: It looks like apples are red.
Our Fruit Class Continues

- Teacher: Being circular is a good feature for the apples. However, if you only say circular, you could make several mistakes. What else can we say for an apple? Tina?
- Tina: It looks like apples are red.

(Class): Apples are somewhat circular and somewhat red.
Teacher: Yes. Many apples are red. However, you could still make mistakes based on circular and red. Do you have any other suggestions, Joey?
Our Fruit Class Continues More

- Teacher: Yes. Many apples are red. However, you could still make mistakes based on circular and red. Do you have any other suggestions, Joey?
- Joey: Apples could also be **green**.
Our Fruit Class Continues More

- Teacher: Yes. Many apples are red. However, you could still make mistakes based on circular and red. Do you have any other suggestions, Joey?
- Joey: Apples could also be green.

(Class): Apples are somewhat circular and somewhat red and possibly green.
Our Fruit Class Ends

- Teacher: Yes. It seems that apples might be circular, red, green. But you may confuse them with tomatoes or peaches, right? Any more suggestions, Jessica?
Our Fruit Class Ends

- Teacher: Yes. It seems that apples might be circular, red, green. But you may confuse them with tomatoes or peaches, right? Any more suggestions, Jessica?
- Jessica: Apples have **stems** at the top.
Our Fruit Class Ends

- Teacher: Yes. It seems that apples might be circular, red, green. But you may confuse them with tomatoes or peaches, right? Any more suggestions, Jessica?
- Jessica: Apples have stems at the top.

(Class): Apples are somewhat circular, somewhat red, possibly green, and may have stems at the top.
• students: simple hypotheses $g_t$ (like vertical/horizontal lines)
• students: simple hypotheses $g_t$ (like vertical/horizontal lines)
• (Class): sophisticated hypothesis $G$ (like black curve)
• students: simple hypotheses $g_t$ (like vertical/horizontal lines)
• (Class): sophisticated hypothesis $G$ (like black curve)
• Teacher: a tactic learning algorithm that directs the students to focus on key examples
students: simple hypotheses $g_t$ (like vertical/horizontal lines)
(Class): sophisticated hypothesis $G$ (like black curve)
Teacher: a tactic learning algorithm that directs the students to focus on key examples

next: the ‘math’ of such an algorithm
Fun Time

Which of the following can help recognize an apple?

1. apples are often circular
2. apples are often red or green
3. apples often have stems at the top
4. all of the above
Which of the following can help recognize an apple?

1. apples are often circular
2. apples are often red or green
3. apples often have stems at the top
4. all of the above

Reference Answer: 4

Congratulations! You have passed first grade. :-(
Bootstrapping as Re-weighting Process

\[\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}\]

bootstrap \[\Rightarrow\] \[\mathcal{D}_t = \{(x_1, y_1), (x_1, y_1), (x_2, y_2), (x_4, y_4)\}\]
Bootstrapping as Re-weighting Process

\[ D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\} \]

bootstrap \( \Rightarrow \) \[ \tilde{D}_t = \{(x_1, y_1), (x_1, y_1), (x_2, y_2), (x_4, y_4)\} \]

\[ E_{in}^{0/1}(h) = \frac{1}{4} \sum_{(x,y) \in \tilde{D}_t} \mathbb{1}[y \neq h(x)] \]

\[
\begin{align*}
(x_1, y_1), & 
(x_1, y_1) \\
(x_2, y_2), & \\
(x_4, y_4), & 
\end{align*}
\]
**Bootstrapping as Re-weighting Process**

Let \( D = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\} \)

**Bootstrap**

\[
\text{bootstrap} \implies \tilde{D}_t = \{(x_1, y_1), (x_1, y_1), (x_2, y_2), (x_4, y_4)\}
\]

**Weighted \( E_{\text{in}} \) on \( D \)**

- \((x_1, y_1), u_1 = 2\)
- \((x_2, y_2), u_2 = 1\)
- \((x_3, y_3), u_3 = 0\)
- \((x_4, y_4), u_4 = 1\)

**\( E_{\text{in}} \) on \( \tilde{D}_t \)**

\[
E_{\text{in}}^{0/1}(h) = \frac{1}{4} \sum_{(x, y) \in \tilde{D}_t} \mathbb{1}[y \neq h(x)]
\]

- \((x_1, y_1), (x_1, y_1)\)
- \((x_2, y_2)\)
- \((x_4, y_4)\)
Bootstrapping as Re-weighting Process

\[ \mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\} \]

\[
\text{bootstrap} \quad \rightarrow \quad \tilde{\mathcal{D}}_t = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}
\]

**weighted** \( E_{\text{in}} \text{ on } \mathcal{D} \)

\[
E_{\text{in}}^u(h) = \frac{1}{4} \sum_{n=1}^{4} u_{n}^{(t)} \cdot [y_n \neq h(x_n)]
\]

- \((x_1, y_1), u_1 = 2\)
- \((x_2, y_2), u_2 = 1\)
- \((x_3, y_3), u_3 = 0\)
- \((x_4, y_4), u_4 = 1\)

**\( E_{\text{in}} \text{ on } \tilde{\mathcal{D}}_t \)**

\[
E_{\text{in}}^{0/1}(h) = \frac{1}{4} \sum_{(x, y) \in \tilde{\mathcal{D}}_t} [y \neq h(x)]
\]

- \((x_1, y_1), (x_1, y_1)\)
- \((x_2, y_2)\)
- \((x_4, y_4)\)
Bootstrapping as Re-weighting Process

\[
\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}
\]

\[
\text{bootstrap} \quad \Rightarrow \quad \tilde{\mathcal{D}}_t = \{(x_1, y_1), (x_1, y_1), (x_2, y_2), (x_4, y_4)\}
\]

**weighted** \(E_{in} \) on \( \mathcal{D} \)

\[
E_{in}^u(h) = \frac{1}{4} \sum_{n=1}^{4} u_n^{(t)} \cdot [y_n \neq h(x_n)]
\]

\((x_1, y_1), u_1 = 2\)

\((x_2, y_2), u_2 = 1\)

\((x_3, y_3), u_3 = 0\)

\((x_4, y_4), u_4 = 1\)

**\(E_{in} \) on \( \tilde{\mathcal{D}}_t \)**

\[
E_{in}^{0/1}(h) = \frac{1}{4} \sum_{(x,y) \in \tilde{\mathcal{D}}_t} [y \neq h(x)]
\]

\((x_1, y_1), (x_1, y_1)\)

\((x_2, y_2)\)

\((x_4, y_4)\)

each diverse \(g_t\) in bagging:

by minimizing **bootstrap-weighted** error
Weighted Base Algorithm

minimize (regularized)

\[ E_{in}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(x_n)) \]

SVM

\[ E_{in} \propto C \sum_{n=1}^{N} u_n \hat{\text{err}}_{SVM} \]

by dual QP

\[ \Leftrightarrow \]

0 \leq \alpha_n \leq C u_n \log \text{logistic regression}

\[ E_{in} \propto N \sum_{n=1}^{N} u_n \text{err}_{CE} \]

by SGD

\[ \Leftrightarrow \]

sample \((x_n, y_n)\) with probability proportional to \(u_n\)
Weighted Base Algorithm

minimize (regularized)

\[ E_{\text{in}}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(x_n)) \]

SVM

\[ E_{\text{in}}^u \propto C \sum_{n=1}^{N} u_n \hat{\text{err}}_{\text{SVM}} \text{ by dual QP} \]
Weighted Base Algorithm

minimize (regularized)

\[ E_{\text{in}}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(x_n)) \]

**SVM**

\[ E_{\text{in}}^u \propto C \sum_{n=1}^{N} u_n \widehat{\text{err}}_{\text{SVM}} \text{ by dual QP} \]

\[ \leftrightarrow \text{adjusted upper bound} \]

\[ 0 \leq \alpha_n \leq C u_n \]
Weighted Base Algorithm

minimize (regularized)

\[ E_{\text{in}}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(x_n)) \]

**SVM**

\[ E_{\text{in}}^u \propto C \sum_{n=1}^{N} u_n \hat{\text{err}}_{\text{SVM}} \text{ by dual QP} \]

\[ 0 \leq \alpha_n \leq C u_n \]

**logistic regression**

\[ E_{\text{in}}^u \propto \sum_{n=1}^{N} u_n \text{err}_{\text{CE}} \text{ by SGD} \]
Weighted Base Algorithm

minimize (regularized)

\[ E_{in}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(x_n)) \]

SVM

\[ E_{in}^u \propto C \sum_{n=1}^{N} u_n \hat{\text{err}}_{SVM} \text{ by dual QP} \]

\[ \iff \text{adjusted upper bound} \]

\[ 0 \leq \alpha_n \leq C u_n \]

logistic regression

\[ E_{in}^u \propto \sum_{n=1}^{N} u_n \text{err}_{CE} \text{ by SGD} \]

\[ \iff \text{sample } (x_n, y_n) \text{ with probability proportional to } u_n \]
Weighted Base Algorithm

minimize (regularized)

\[ E_{\text{in}}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(x_n)) \]

**SVM**

\[ E_{\text{in}}^u \propto C \sum_{n=1}^{N} u_n \hat{\text{err}}_{\text{SVM}} \text{ by dual QP} \]

\[ \iff \text{adjusted upper bound} \]

\[ 0 \leq \alpha_n \leq C u_n \]

**logistic regression**

\[ E_{\text{in}}^u \propto \sum_{n=1}^{N} u_n \text{err}_{\text{CE}} \text{ by SGD} \]

\[ \iff \text{sample } (x_n, y_n) \text{ with probability proportional to } u_n \]

example-weighted learning:
extension of class-weighted learning in Lecture 8 of ML Foundations
Re-weighting for More Diverse Hypothesis

‘improving’ bagging for binary classification:
how to re-weight for more diverse hypotheses?
Re-weighting for More Diverse Hypothesis

‘improving’ bagging for binary classification:
how to re-weight for more diverse hypotheses?

$$g_t \leftarrow \arg\min_{h \in H} \left( \sum_{n=1}^{N} u_n^{(t)} [y_n \neq h(x_n)] \right)$$

$$g_{t+1} \leftarrow \arg\min_{h \in H} \left( \sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq h(x_n)] \right)$$
‘improving’ bagging for binary classification: how to re-weight for more diverse hypotheses?

\[
g_t \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t)} \left[ y_n \neq h(x_n) \right] \right)
\]

\[
g_{t+1} \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t+1)} \left[ y_n \neq h(x_n) \right] \right)
\]

if \( g_t \) ‘not good’ for \( u^{(t+1)} \)
Re-weighting for More Diverse Hypothesis

‘improving’ bagging for binary classification: how to re-weight for more diverse hypotheses?

\[
\begin{align*}
g_t & \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t)} [y_n \neq h(x_n)] \right) \\
g_{t+1} & \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq h(x_n)] \right)
\end{align*}
\]

if \(g_t\) ‘not good’ for \(u^{(t+1)}\) \(\implies\) \(g_t\)-like hypotheses not returned as \(g_{t+1}\)
Re-weighting for More Diverse Hypothesis

‘improving’ bagging for binary classification: how to re-weight for **more diverse hypotheses**?

\[
g_t \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t)} [y_n \neq h(x_n)] \right)
\]

\[
g_{t+1} \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq h(x_n)] \right)
\]

if \( g_t \) ‘not good’ for \( u^{(t+1)} \) \( \implies \) \( g_t \)-like hypotheses not returned as \( g_{t+1} \)

\( \implies \) \( g_{t+1} \) diverse from \( g_t \)
Re-weighting for More Diverse Hypothesis

‘improving’ bagging for binary classification:
how to re-weight for **more diverse hypotheses**?

\[
g_t \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t)} [y_n \neq h(x_n)] \right)
\]

\[
g_{t+1} \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq h(x_n)] \right)
\]

If \( g_t \) ‘not good’ for \( u^{(t+1)} \) \( \Rightarrow g_t \)-like hypotheses not returned as \( g_{t+1} \)

\( \Rightarrow g_{t+1} \) diverse from \( g_t \)

**idea:** construct \( u^{(t+1)} \) to make \( g_t \) random-like

\[
\frac{\sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t+1)}} =
\]
Re-weighting for More Diverse Hypothesis

‘improving’ bagging for binary classification: how to re-weight for **more diverse hypotheses**?

\[
g_t \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t)} \left[ y_n \neq h(x_n) \right] \right)
\]

\[
g_{t+1} \leftarrow \arg\min_{h \in \mathcal{H}} \left( \sum_{n=1}^{N} u_n^{(t+1)} \left[ y_n \neq h(x_n) \right] \right)
\]

if \( g_t \) ‘not good’ for \( u^{(t+1)} \) \implies \( g_t \)-like hypotheses not returned as \( g_{t+1} \)

\implies \( g_{t+1} \) diverse from \( g_t \)

idea: **construct** \( u^{(t+1)} \) to make \( g_t \) random-like

\[
\frac{\sum_{n=1}^{N} u_n^{(t+1)} \left[ y_n \neq g_t(x_n) \right]}{\sum_{n=1}^{N} u_n^{(t+1)}} = \frac{1}{2}
\]
‘Optimal’ Re-weighting

want: \[
\sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)] \]
\[
\sum_{n=1}^{N} u_n^{(t+1)}
\]

= \frac{1}{2}, \text{ where}

\[
\sum_{n=1}^{N} u_n^{(t+1)}
\]
'Optimal' Re-weighting

\[
\text{want: } \frac{\sum_{n=1}^{N} u_n^{(t+1)} \left[y_n \neq g_t(x_n)\right]}{\sum_{n=1}^{N} u_n^{(t+1)}} = \frac{\Box^{t+1}}{\Box^{t+1} + \bullet^{t+1}} = \frac{1}{2}, \text{ where }
\]

\[
\Box^{t+1} = \sum_{n=1}^{N} u_n^{(t+1)}, \quad \bullet^{t+1} = \sum_{n=1}^{N} u_n^{(t+1)}
\]
'Optimal' Re-weighting

want: \[
\sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)] = \frac{1}{2}, \text{ where}
\]

\[t+1 = \sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)], \quad \sum_{n=1}^{N} u_n^{(t+1)} \]
‘Optimal’ Re-weighting

\[
\begin{align*}
\text{want: } & \frac{\sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t+1)}} = \frac{\Box_{t+1}}{\Box_{t+1} + \bigcirc_{t+1}} = \frac{1}{2}, \text{ where} \\
\Box_{t+1} &= \sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)], \\
\bigcirc_{t+1} &= \sum_{n=1}^{N} u_n^{(t+1)} [y_n = g_t(x_n)]
\end{align*}
\]
'Optimal' Re-weighting

want: \[
\sum_{n=1}^{N} u_n^{(t+1)} \left[ y_n \neq g_t(x_n) \right] = \frac{\Box_{t+1}}{\Box_{t+1} + \bullet_{t+1}} = \frac{1}{2}, \text{ where }
\]

\[
\Box_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} \left[ y_n \neq g_t(x_n) \right], \quad \bullet_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} \left[ y_n = g_t(x_n) \right]
\]

need: \( \text{(total } u_n^{(t+1)} \text{ of incorrect)} = \frac{\Box_{t+1}}{\bullet_{t+1}} = \text{(total } u_n^{(t+1)} \text{ of correct)} \)
‘Optimal’ Re-weighting

want: \( \frac{\sum_{n=1}^{N} u_n^{(t+1)} \left[ y_n \neq g_t(x_n) \right]}{\sum_{n=1}^{N} u_n^{(t+1)}} = \frac{\square_{t+1}}{\square_{t+1} + \bullet_{t+1}} = \frac{1}{2} \), where

\[ \square_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} \left[ y_n \neq g_t(x_n) \right], \quad \bullet_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} \left[ y_n = g_t(x_n) \right] \]

- need: (total \( u_n^{(t+1)} \) of incorrect) \( = \) (total \( u_n^{(t+1)} \) of correct)

- one possibility by re-scaling (multiplying) weights, if

<table>
<thead>
<tr>
<th>(total ( u_n^{(t)} ) of incorrect) = 1126</th>
<th>(total ( u_n^{(t)} ) of correct) = 6211</th>
</tr>
</thead>
<tbody>
<tr>
<td>incorrect: ( u_n^{(t+1)} \leftarrow u_n^{(t)} )</td>
<td>correct: ( u_n^{(t+1)} \leftarrow u_n^{(t)} )</td>
</tr>
</tbody>
</table>
‘Optimal’ Re-weighting

want: \( \frac{\sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t+1)}} = \frac{\blacksquare_{t+1}}{\blacksquare_{t+1} + \bullet_{t+1}} = \frac{1}{2} \), where

\( \blacksquare_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)] \), \( \bullet_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} [y_n = g_t(x_n)] \)

- need: (total \( u_n^{(t+1)} \) of incorrect) = (total \( u_n^{(t+1)} \) of correct)
  - \( \blacksquare_{t+1} \)
  - \( \bullet_{t+1} \)
- one possibility by re-scaling (multiplying) weights, if
  - (total \( u_n^{(t)} \) of incorrect) = 1126 ; (total \( u_n^{(t)} \) of correct) = 6211 ;
  - incorrect: \( u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot 6211 \)
  - correct: \( u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot 1126 \)
‘Optimal’ Re-weighting

want: $$\sum_{n=1}^{N} u_{n}^{(t+1)} [y_{n} \neq g_{t}(x_{n})] \over \sum_{n=1}^{N} u_{n}^{(t+1)} = \frac{t+1}{t+1 + \bullet t+1} = \frac{1}{2}$$, where

$$\square t+1 = \sum_{n=1}^{N} u_{n}^{(t+1)} [y_{n} \neq g_{t}(x_{n})]$$, $$\bullet t+1 = \sum_{n=1}^{N} u_{n}^{(t+1)} [y_{n} = g_{t}(x_{n})]$$

- need: (total $$u_{n}^{(t+1)}$$ of incorrect) = (total $$u_{n}^{(t+1)}$$ of correct)

- one possibility by re-scaling (multiplying) weights, if

<table>
<thead>
<tr>
<th>(total $$u_{n}^{(t)}$$ of incorrect)</th>
<th>= 1126</th>
<th>(total $$u_{n}^{(t)}$$ of correct)</th>
<th>= 6211</th>
</tr>
</thead>
<tbody>
<tr>
<td>(weighted incorrect rate)</td>
<td>= $\frac{1126}{7337}$</td>
<td>(weighted correct rate)</td>
<td>= $\frac{6211}{7337}$</td>
</tr>
<tr>
<td>incorrect: $$u_{n}^{(t+1)} \leftarrow u_{n}^{(t)} \cdot 6211$$</td>
<td>corrected: $$u_{n}^{(t+1)} \leftarrow u_{n}^{(t)} \cdot 1126$$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
‘Optimal’ Re-weighting

\[
\frac{\sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t+1)}} = \frac{\bullet_{t+1}}{\bullet_{t+1} + \bullet_{t+1}} = \frac{1}{2}, \text{ where}
\]

\[
\bullet_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} [y_n \neq g_t(x_n)], \quad \bullet_{t+1} = \sum_{n=1}^{N} u_n^{(t+1)} [y_n = g_t(x_n)].
\]

- need: \(\text{(total } u_n^{(t+1)} \text{ of incorrect)} = \text{(total } u_n^{(t+1)} \text{ of correct)}\)
- one possibility by \text{re-scaling (multiplying) weights}, if

| \(\text{total } u_n^{(t)} \text{ of incorrect)} | = 1126 ; | \(\text{total } u_n^{(t)} \text{ of correct)} | = 6211 ;
| \(\text{weighted incorrect rate)} | = \frac{1126}{7337} ; | \(\text{weighted correct rate)} | = \frac{6211}{7337} |

incorrect: \(u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot 6211 \)
correct: \(u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot 1126 \)

‘optimal’ re-weighting under weighted incorrect rate \(\epsilon_t\):
\[\text{multiply incorrect } \propto (1 - \epsilon_t); \text{ multiply correct } \propto \epsilon_t\]
For four examples with $u_n^{(1)} = \frac{1}{4}$ for all examples. If $g_1$ predicts the first example wrongly but all the other three examples correctly. After the ‘optimal’ re-weighting, what is $u_1^{(2)}/u_2^{(2)}$?

1. 4
2. 3
3. 1/3
4. 1/4
For four examples with $u_n^{(1)} = \frac{1}{4}$ for all examples. If $g_1$ predicts the first example wrongly but all the other three examples correctly. After the ‘optimal’ re-weighting, what is $u_1^{(2)}/u_2^{(2)}$?

1. 4
2. 3
3. 1/3
4. 1/4

Reference Answer: 2

By ‘optimal’ re-weighting, $u_1$ is scaled proportional to $\frac{3}{4}$ and every other $u_n$ is scaled proportional to $\frac{1}{4}$. So example 1 is now three times more important than any other example.
‘optimal’ re-weighting: let $\epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t)}},$

multiply incorrect $\propto (1 - \epsilon_t)$; multiply correct $\propto \epsilon_t$
Scaling Factor

‘optimal’ re-weighting: let \( \epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)}[y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t)}} \),

multiply incorrect \( \propto (1 - \epsilon_t) \); multiply correct \( \propto \epsilon_t \)

define scaling factor \( \diamond t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \)

incorrect \( \leftarrow \) incorrect \( . \) \( \diamond t \)
correct \( \leftarrow \) correct \( / \) \( \diamond t \)
Scaling Factor

'optimal' re-weighting: let \( \epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t)}} \),

multiply incorrect \( \propto (1 - \epsilon_t) \); multiply correct \( \propto \epsilon_t \)

define scaling factor \( \diamondsuit_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \)

incorrect \( \leftarrow \) incorrect \( \cdot \) \( \diamondsuit_t \)
correct \( \leftarrow \) correct \( \div \) \( \diamondsuit_t \)

- equivalent to optimal re-weighting
Scaling Factor

`optimal` re-weighting: let $\epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t)}}$, multiply incorrect $\propto (1 - \epsilon_t)$; multiply correct $\propto \epsilon_t$

define scaling factor $\star_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

- incorrect $\leftarrow$ incorrect $\cdot \star_t$
- correct $\leftarrow$ correct $/ \star_t$

- equivalent to optimal re-weighting
- $\star_t \geq 1$ iff $\epsilon_t \leq \frac{1}{2}$
  —physical meaning:
Scaling Factor

'optimal' re-weighting: let \( \epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} \mathbb{1}_{y_n \neq g_t(x_n)}}{\sum_{n=1}^{N} u_n^{(t)}} \),

multiply incorrect \( \propto (1 - \epsilon_t) \); multiply correct \( \propto \epsilon_t \)

define scaling factor \( \diamondsuit_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \)

\[
\begin{align*}
\text{incorrect} & \leftarrow \text{incorrect} \cdot \diamondsuit_t \\
\text{correct} & \leftarrow \text{correct} / \diamondsuit_t
\end{align*}
\]

- equivalent to optimal re-weighting
- \( \diamondsuit_t \geq 1 \) iff \( \epsilon_t \leq \frac{1}{2} \)
  —physical meaning: scale up incorrect; scale down correct
Scaling Factor

‘optimal’ re-weighting: let $\epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t)}}$,

multiply incorrect $\propto (1 - \epsilon_t)$; multiply correct $\propto \epsilon_t$

define scaling factor $\diamondsuit_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}$

- incorrect $\leftarrow$ incorrect $\cdot \diamondsuit_t$
- correct $\leftarrow$ correct $/ \diamondsuit_t$

- equivalent to optimal re-weighting

$\diamondsuit_t \geq 1$ iff $\epsilon_t \leq \frac{1}{2}$
—physical meaning: scale up incorrect; scale down correct
—like what Teacher does
Scaling Factor

‘optimal’ re-weighting: let $\epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} [y_n \neq g_t(x_n)]}{\sum_{n=1}^{N} u_n^{(t)}}$,

multiply incorrect $\propto (1 - \epsilon_t)$; multiply correct $\propto \epsilon_t$

define scaling factor $\downarrow_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

incorrect $\leftarrow$ incorrect $\cdot \downarrow_t$

correct $\leftarrow$ correct $/ \downarrow_t$

- equivalent to optimal re-weighting
- $\downarrow_t \geq 1$ iff $\epsilon_t \leq \frac{1}{2}$
  —physical meaning: scale up incorrect; scale down correct
  —like what Teacher does

scaling-up incorrect examples leads to diverse hypotheses
**Adaptive Boosting**

**Adaptive Boosting Algorithm**

A Preliminary Algorithm

$\mathbf{u}^{(1)} = ?$

for $t = 1, 2, \ldots, T$

1. obtain $g_t$ by $\mathcal{A}(\mathcal{D}, \mathbf{u}^{(t)})$,
   where $\mathcal{A}$ tries to minimize $\mathbf{u}^{(t)}$-weighted 0/1 error
A Preliminary Algorithm

\[ u^{(1)} = ? \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(\mathcal{D}, u^{(t)}) \), where \( A \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamond_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \), where \( \epsilon_t \) = weighted error (incorrect) rate of \( g_t \)
A Preliminary Algorithm

\( u^{(1)} =? \)

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \),
   where \( A \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \),
   where \( \epsilon_t = \) weighted error (incorrect) rate of \( g_t \)

return \( G(x) =? \)
A Preliminary Algorithm

\( u^{(1)} = ? \)

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \),
   where \( A \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \),
   where \( \epsilon_t = \) weighted error (incorrect) rate of \( g_t \)

return \( G(x) = ? \)

- want \( g_1 \) ‘best’ for \( E_{in} : u_n^{(1)} = \frac{1}{N} \)
A Preliminary Algorithm

\[ u^{(1)} = ? \]
for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \),
   where \( A \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by
   \[ \diamondsuit_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} , \]
   where \( \epsilon_t = \) weighted error (incorrect) rate of \( g_t \)

return \( G(x) = ? \)

- want \( g_1 \) ‘best’ for \( E_{in} : u^{(1)}_n = \frac{1}{N} \)
- \( G(x) : \)
Adaptive Boosting

A Preliminary Algorithm

\[ u^{(1)} = ? \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \),
   where \( A \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamondsuit_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \),
   where \( \epsilon_t = \) weighted error (incorrect) rate of \( g_t \)

return \( G(x) = ? \)

- want \( g_1 \) ‘best’ for \( E_{in} \): \( u_{in}^{(1)} = \frac{1}{N} \)
- \( G(x) \):
  - uniform?
Adaptive Boosting

A Preliminary Algorithm

\[ u^{(1)} = ? \]
for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \),
   where \( A \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \),
   where \( \epsilon_t \) = weighted error (incorrect) rate of \( g_t \)

return \( G(x) = ? \)

- want \( g_1 \) ‘best’ for \( E_{\text{in}} \): \( u_n^{(1)} = \frac{1}{N} \)
- \( G(x) \):
  - uniform? but \( g_2 \) very bad for \( E_{\text{in}} \) (why? :-))
Adaptive Boosting

Adaptive Boosting Algorithm

A Preliminary Algorithm

\[ u^{(1)} =? \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \),
   where \( A \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by
   \[ \diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \]
   where \( \epsilon_t = \) weighted error (incorrect) rate of \( g_t \)

return \( G(x) =? \)

- want \( g_1 \) ‘best’ for \( E_{in} : u_n^{(1)} = \frac{1}{N} \)
- \( G(x) : \)
  - uniform? but \( g_2 \) very bad for \( E_{in} \) (why? :-))
  - linear, non-linear? as you wish
A Preliminary Algorithm

\[ u^{(1)} = \gamma \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \),
   where \( A \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by
   \[ \epsilon_t = \sqrt{1 - \frac{1 - \epsilon_t}{\epsilon_t}} \]
   where \( \epsilon_t \) = weighted error (incorrect) rate of \( g_t \)

return \( G(x) = \gamma \)

- want \( g_1 \) ‘best’ for \( E_{in} \): \( u^{(1)}_n = \frac{1}{N} \)
- \( G(x) \):
  - uniform? but \( g_2 \) very bad for \( E_{in} \) (why? :-))
  - linear, non-linear? as you wish

next: a special algorithm to aggregate linearly on the fly with theoretical guarantee
Adaptive Boosting

Adaptive Boosting Algorithm

Linear Aggregation on the Fly

\[ \mathbf{u}^{(1)} = \left[ \frac{1}{N}, \frac{1}{N}, \cdots, \frac{1}{N} \right] \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, \mathbf{u}^{(t)}) \), where ...

2. update \( \mathbf{u}^{(t)} \) to \( \mathbf{u}^{(t+1)} \) by \( \diamondsuit_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \), where ...

Hsuan-Tien Lin (NTU CSIE)  Machine Learning Techniques
Linear Aggregation on the Fly

\[ u^{(1)} = \left[ \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \right] \]

for \( t = 1, 2, \ldots, T \)

1. Obtain \( g_t \) by \( A(D, u^{(t)}) \), where ...
2. Update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \), where ...
3. Compute \( \alpha_t \)

return \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x) \right) \)
Adaptive Boosting
Adaptive Boosting Algorithm

Linear Aggregation on the Fly

\[ u^{(1)} = \left[ \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \right] \]
for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \), where ...

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamond_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \), where ...

3. compute \( \alpha_t \)

return \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x) \right) \)

- wish: large \( \alpha_t \) for ‘good’ \( g_t \)
Adaptive Boosting

Linear Aggregation on the Fly

\[ u^{(1)} = [\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N}] \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \), where ...
2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamond_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \), where ...
3. compute \( \alpha_t \)

return \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x) \right) \)

- wish: large \( \alpha_t \) for ‘good’ \( g_t \) \( \Leftarrow \) \( \alpha_t = \text{monotonic}(\diamond_t) \)
Adaptive Boosting Algorithm

Linear Aggregation on the Fly

\[ u^{(1)} = [\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N}] \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \), where ...
2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamond_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \), where ...
3. compute \( \alpha_t = \ln(\diamond_t) \)

return \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x) \right) \)

- wish: large \( \alpha_t \) for ‘good’ \( g_t \leftarrow \alpha_t = \text{monotonic}(\diamond_t) \)
- will take \( \alpha_t = \ln(\diamond_t) \)
Adaptive Boosting = weak base learning algorithm

Adaptive Boosting = optimal re-weighting factor

Adaptive Boosting = 'magic' linear aggregation

Adaptive Boosting Algorithm

Linear Aggregation on the Fly

\[ u^{(1)} = \left[ \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \right] \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \), where ...

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamondsuit_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \), where ...

3. compute \( \alpha_t = \ln(\diamondsuit_t) \)

return \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x) \right) \)

- wish: large \( \alpha_t \) for 'good' \( g_t \leftarrow \alpha_t = \text{monotonic}(\diamondsuit_t) \)
- will take \( \alpha_t = \ln(\diamondsuit_t) \)
  - \( \epsilon_t = \frac{1}{2} \implies \diamondsuit_t = 1 \implies \alpha_t = 0 \) (bad \( g_t \) zero weight)
Adaptive Boosting Algorithm

**Linear Aggregation on the Fly**

\[
\mathbf{u}^{(1)} = \left[ \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \right]
\]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, \mathbf{u}^{(t)}) \), where ...

2. update \( \mathbf{u}^{(t)} \) to \( \mathbf{u}^{(t+1)} \) by \( \diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \), where ...

3. compute \( \alpha_t = \ln(\diamond_t) \)

return \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x) \right) \)

- wish: large \( \alpha_t \) for ‘good’ \( g_t \) \( \Leftarrow \) \( \alpha_t = \) monotonic(\( \diamond_t \))
- will take \( \alpha_t = \ln(\diamond_t) \)
  - \( \epsilon_t = \frac{1}{2} \Rightarrow \diamond_t = 1 \Rightarrow \alpha_t = 0 \) (bad \( g_t \) zero weight)
  - \( \epsilon_t = 0 \Rightarrow \diamond_t = \infty \Rightarrow \alpha_t = \infty \) (super \( g_t \) superior weight)
Adaptive Boosting Algorithm

Linear Aggregation on the Fly

\[ u^{(1)} = \left[ \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \right] \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(D, u^{(t)}) \), where ...

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by \( \diamondsuit_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \), where ...

3. compute \( \alpha_t = \ln(\diamondsuit_t) \)

return \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x) \right) \)

- wish: large \( \alpha_t \) for 'good' \( g_t \) \( \Leftarrow \) \( \alpha_t = \text{monotonic}(\diamondsuit_t) \)
- will take \( \alpha_t = \ln(\diamondsuit_t) \)
  - \( \epsilon_t = \frac{1}{2} \Rightarrow \diamondsuit_t = 1 \Rightarrow \alpha_t = 0 \) (bad \( g_t \) zero weight)
  - \( \epsilon_t = 0 \Rightarrow \diamondsuit_t = \infty \Rightarrow \alpha_t = \infty \) (super \( g_t \) superior weight)

Adaptive Boosting = weak base learning algorithm \( A \) (Student)

+ optimal re-weighting factor \( \diamondsuit_t \) (Teacher)

+ 'magic' linear aggregation \( \alpha_t \) (Class)
Adaptive Boosting (AdaBoost) Algorithm

\[ u^{(1)} = \left[ \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \right] \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( \mathcal{A}(D, u^{(t)}) \),
   where \( \mathcal{A} \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by
   \[ \left[ y_n \neq g_t(x_n) \right] \text{ (incorrect examples)}: \quad u_n^{(t+1)} \leftarrow u_n^{(t)} \cdot \diamondsuit_t \]
   \[ \left[ y_n = g_t(x_n) \right] \text{ (correct examples)}: \quad u_n^{(t+1)} \leftarrow u_n^{(t)} / \diamondsuit_t \]
   where \( \diamondsuit_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \) and
   \[ \epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} \left[ y_n \neq g_t(x_n) \right]}{\sum_{n=1}^{N} u_n^{(t)}} \]

3. compute \( \alpha_t = \ln(\diamondsuit_t) \)

return \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x) \right) \)
Adaptive Boosting (AdaBoost) Algorithm

\[ u^{(1)} = \left[ \frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N} \right] \]

for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(\mathcal{D}, u^{(t)}) \), where \( A \) tries to minimize \( u^{(t)} \)-weighted 0/1 error

2. update \( u^{(t)} \) to \( u^{(t+1)} \) by

\[
\begin{align*}
\llbracket y_n \neq g_t(x_n) \rrbracket \text{ (incorrect examples):} & \quad u^{(t+1)}_n \leftarrow u^{(t)}_n \cdot \diamond^t \\
\llbracket y_n = g_t(x_n) \rrbracket \text{ (correct examples):} & \quad u^{(t+1)}_n \leftarrow u^{(t)}_n / \diamond^t 
\end{align*}
\]

where \( \diamond^t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \) and \( \epsilon_t = \frac{\sum_{n=1}^{N} u^{(t)}_n \llbracket y_n \neq g_t(x_n) \rrbracket}{\sum_{n=1}^{N} u^{(t)}_n} \)

3. compute \( \alpha_t = \ln(\diamond^t) \)

return \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x) \right) \)

AdaBoost: provable boosting property
Theoretical Guarantee of AdaBoost

- From VC bound

\[ E_{\text{out}}(G) \leq E_{\text{in}}(G) + O\left( \sqrt{d_{\text{VC}}(H) \cdot T \log T} \cdot \frac{\log N}{N} \right) \]

- First term can be small:
  \( E_{\text{in}}(G) = 0 \) after \( T = O(\log N) \) iterations if \( \epsilon_t \leq \epsilon < \frac{1}{2} \) always

- Second term can be small:
  Overall \( d_{\text{VC}} \) grows "slowly" with \( T \) boosting view of AdaBoost:
  If \( A \) is weak but always slightly better than random \( (\epsilon_t \leq \epsilon < \frac{1}{2}) \), then (AdaBoost + \( A \)) can be strong \( (E_{\text{in}} = 0 \) and \( E_{\text{out}} \) small)
Theoretical Guarantee of AdaBoost

- From VC bound

\[ E_{\text{out}}(G) \leq E_{\text{in}}(G) + O\left( O(d_{\text{VC}}(\mathcal{H}) \cdot T \log T) \cdot \frac{\log N}{N} \right) \]

- first term can be small:

\[ E_{\text{in}}(G) = 0 \text{ after } T = O(\log N) \text{ iterations if } \epsilon_t \leq \epsilon < \frac{1}{2} \text{ always} \]
Theoretical Guarantee of AdaBoost

- From VC bound
  \[
  E_{\text{out}}(G) \leq E_{\text{in}}(G) + O\left(\sqrt{\frac{O(d_{\text{VC}}(\mathcal{H}) \cdot T \log T)}{N} \cdot \frac{\log N}{N}}\right)
  \]

  - first term can be small:
    \[E_{\text{in}}(G) = 0\] after \(T = O(\log N)\) iterations if \(\epsilon_t \leq \epsilon < \frac{1}{2}\) always

  - second term can be small:
    overall \(d_{\text{VC}}\) grows “slowly” with \(T\)
Theoretical Guarantee of AdaBoost

- From VC bound

\[
E_{\text{out}}(G) \leq E_{\text{in}}(G) + O\left(\frac{O(d_{\text{VC}}(\mathcal{H}) \cdot T \log T) \cdot \log N}{d_{\text{VC}} \text{ of all possible } G} \right)
\]

- first term can be small:

\[E_{\text{in}}(G) = 0 \text{ after } T = O(\log N) \text{ iterations if } \epsilon_t \leq \epsilon < \frac{1}{2} \text{ always}\]

- second term can be small:

overall \(d_{\text{VC}}\) grows “slowly” with \(T\)

boosting view of AdaBoost:

if \(A\) is weak but always slightly better than random \((\epsilon_t \leq \epsilon < \frac{1}{2})\),
then (AdaBoost+\(A\)) can be strong \((E_{\text{in}} = 0 \text{ and } E_{\text{out}} \text{ small})\)
According to $\alpha_t = \ln(\diamond_t)$, and $\diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$, when would $\alpha_t > 0$?

1. $\epsilon_t < \frac{1}{2}$
2. $\epsilon_t > \frac{1}{2}$
3. $\epsilon_t \neq 1$
4. $\epsilon_t \neq 0$
According to $\alpha_t = \ln(\diamond_t)$, and $\diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$, when would $\alpha_t > 0$?

1. $\epsilon_t < \frac{1}{2}$
2. $\epsilon_t > \frac{1}{2}$
3. $\epsilon_t \neq 1$
4. $\epsilon_t \neq 0$

Reference Answer: 1

The math part should be easy for you, and it is interesting to think about the physical meaning: $\alpha_t > 0 \ (g_t \text{ is useful for } G)$ if and only if the weighted error rate of $g_t$ is better than random!
want: a ‘weak’ base learning algorithm $A$ that minimizes $E_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot [y_n \neq h(x_n)]$ a little bit
Decision Stump

want: a ‘weak’ base learning algorithm \( \mathcal{A} \) that minimizes \( E_{\text{in}}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot [y_n \neq h(x_n)] \) a little bit

a popular choice: decision stump

- in ML Foundations Homework 2, remember? :-)

\[ h_{s,i,\theta}(x) = s \cdot \text{sign}(x_i - \theta) \]
**Decision Stump**

**Adaptive Boosting in Action**

Want: a ‘**weak**’ base learning algorithm $\mathcal{A}$ that minimizes $E_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot [y_n \neq h(x_n)]$

A little bit +

A popular choice: decision stump

- **In ML Foundations Homework 2, remember? :-)**

$$h_{s,i,\theta}(x) = s \cdot \text{sign}(x_i - \theta)$$

- Positive and negative rays on some feature: three parameters (feature $i$, threshold $\theta$, direction $s$)
Decision Stump

want: a ‘weak’ base learning algorithm $\mathcal{A}$ that minimizes $E_{\text{in}}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot [y_n \neq h(x_n)]$ a little bit

a popular choice: decision stump

- in ML Foundations Homework 2, remember? :-)

$$h_{s,i,\theta}(x) = s \cdot \text{sign}(x_i - \theta)$$

- positive and negative rays on some feature: three parameters (feature $i$, threshold $\theta$, direction $s$)

- physical meaning: vertical/horizontal lines in 2D
Adaptive Boosting

Decision Stump

want: a ‘weak’ base learning algorithm $\mathcal{A}$ that minimizes $E_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot [y_n \neq h(x_n)]$ a little bit

a popular choice: decision stump

- in ML Foundations Homework 2, remember? :-) 

$$h_{s,i,\theta}(x) = s \cdot \text{sign}(x_i - \theta)$$

- positive and negative rays on some feature: three parameters (feature $i$, threshold $\theta$, direction $s$)
- physical meaning: vertical/horizontal lines in 2D
- efficient to optimize: $O(d \cdot N \log N)$ time
Decision Stump

want: a ‘weak’ base learning algorithm \( \mathcal{A} \)

that minimizes

\[
E_{\text{in}}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \left[ y_n \neq h(x_n) \right]
\]

a little bit

a popular choice: decision stump

- in ML Foundations Homework 2, remember? :-) 

\[
h_{s,i,\theta}(x) = s \cdot \text{sign}(x_i - \theta)
\]

- positive and negative rays on some feature: three parameters
  (feature \( i \), threshold \( \theta \), direction \( s \))

- physical meaning: vertical/horizontal lines in 2D

- efficient to optimize: \( O(d \cdot N \log N) \) time

\[\text{decision stump model:}
\]

allows efficient minimization of \( E_{\text{in}}^u \)

but perhaps too weak to work by itself
A Simple Data Set

initially
Adaptive Boosting in Action

A Simple Data Set

t = 1
A Simple Data Set

t = 2
A Simple Data Set

$t = 3$
A Simple Data Set

t = 4

Hsuan-Tien Lin (NTU CSIE)

Machine Learning Techniques
A Simple Data Set
A Simple Data Set

‘Teacher’-like algorithm works!
A Complicated Data Set

initially
A Complicated Data Set

AdaBoost-Stump: non-linear yet efficient
A Complicated Data Set

Adaptive Boosting

AdaBoost-Stump: non-linear yet efficient
A Complicated Data Set
A Complicated Data Set
A Complicated Data Set

AdaBoost-Stump: non-linear yet efficient
A Complicated Data Set
A Complicated Data Set

t = 70
A Complicated Data Set

AdaBoost-Stump: non-linear yet efficient

Hsuan-Tien Lin (NTU CSIE)
A Complicated Data Set

t = 90
A Complicated Data Set

AdaBoost-Stump: non-linear yet efficient
The World’s First ‘Real-Time’ Face Detection Program

original picture by F.U.S.I.A. assistant and derivative work by Sylenius via Wikimedia Commons
The World’s First ‘Real-Time’ Face Detection Program

- **AdaBoost-Stump** as core model: linear aggregation of key patches selected out of 162,336 possibilities in 24x24 images — feature selection achieved through AdaBoost-Stump
AdaBoost-Stump in Application

**The World’s First ‘Real-Time’ Face Detection Program**

- **AdaBoost-Stump** as core model: linear aggregation of key patches selected out of 162,336 possibilities in 24x24 images — **feature selection** achieved through AdaBoost-Stump
- modified linear aggregation $G$ to rule out non-face earlier — **efficiency** achieved through modified linear aggregation
Adaptive Boosting

AdaBoost-Stump in Application

original picture by F.U.S.I.A. assistant and derivative work by Sylenius via Wikimedia Commons

The World’s First ‘Real-Time’ Face Detection Program

- **AdaBoost-Stump** as core model: linear aggregation of key patches selected out of 162,336 possibilities in 24x24 images —feature selection achieved through **AdaBoost-Stump**

- modified linear aggregation \( G \) to rule out non-face earlier —efficiency achieved through **modified linear aggregation**

**AdaBoost-Stump:**
  efficient feature selection and aggregation
For a data set of size 9876 that contains $\mathbf{x}_n \in \mathbb{R}^{5566}$, after running AdaBoost-Stump for 1126 iterations, what is the number of distinct features within $\mathbf{x}$ that are effectively used by $G$?

1. $0 \leq \text{number} \leq 1126$
2. $1126 < \text{number} \leq 5566$
3. $5566 < \text{number} \leq 9876$
4. $9876 < \text{number}$
For a data set of size 9876 that contains $x_n \in \mathbb{R}^{5566}$, after running AdaBoost-Stump for 1126 iterations, what is the number of distinct features within $x$ that are effectively used by $G$?

1. $0 \leq \text{number} \leq 1126$
2. $1126 < \text{number} \leq 5566$
3. $5566 < \text{number} \leq 9876$
4. $9876 < \text{number}$

Reference Answer: ①

Each decision stump takes only one feature. So 1126 decision stumps need at most 1126 distinct features.
Summary

1. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models

Lecture 8: Adaptive Boosting

- Motivation of Boosting
  aggregate weak hypotheses for strength
- Diversity by Re-weighting
  scale up incorrect, scale down correct
- Adaptive Boosting Algorithm
  two heads are better than one, theoretically
- Adaptive Boosting in Action
  AdaBoost-Stump useful and efficient

- next: learning conditional aggregation instead of linear one

3. Distilling Implicit Features: Extraction Models