Lecture 11: Gradient Boosted Decision Tree

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. Embedding Numerous Features: Kernel Models
2. Combining Predictive Features: Aggregation Models
3. Distilling Implicit Features: Extraction Models

Lecture 10: Random Forest
- bagging of randomized C&RT trees with automatic validation and feature selection

Lecture 11: Gradient Boosted Decision Tree
- Adaptive Boosted Decision Tree
- Optimization View of AdaBoost
- Gradient Boosting
- Summary of Aggregation Models
function **RandomForest**($\mathcal{D}$)
For $t = 1, 2, \ldots, T$

1. request size-$N'$ data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
2. obtain tree $g_t$ by Randomized-DTree($\tilde{\mathcal{D}}_t$)

return $G = \text{Uniform} (\{g_t\})$
function **RandomForest**($\mathcal{D}$)  
For $t = 1, 2, \ldots, T$  
1. request size-$N'$ data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$  
2. obtain tree $g_t$ by Randomized-DTree($\tilde{\mathcal{D}}_t$)  
return $G = \text{Uniform} \{ g_t \}$

function **AdaBoost-DTree**($\mathcal{D}$)  
For $t = 1, 2, \ldots, T$  
1. reweight data by $u^{(t)}$
function **RandomForest**$(\mathcal{D})$

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

1. request size-$N'$ data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
2. obtain tree $g_t$ by Randomized-DTree$(\tilde{\mathcal{D}}_t)$

return $G = \text{Uniform}(\{g_t\})$

function **AdaBoost-DTree**$(\mathcal{D})$

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

1. reweight data by $u^{(t)}$
2. obtain tree $g_t$ by DTree$(\mathcal{D}, u^{(t)})$
From Random Forest to AdaBoost-DTree

function $\text{RandomForest}(\mathcal{D})$

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

1. request size-$N'$ data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
2. obtain tree $g_t$ by $\text{Randomized-DTree}(\tilde{\mathcal{D}}_t)$

return $G = \text{Uniform}(\{g_t\})$

function $\text{AdaBoost-DTree}(\mathcal{D})$

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

1. reweight data by $u^{(t)}$
2. obtain tree $g_t$ by $\text{DTree}(\mathcal{D}, u^{(t)})$
3. calculate ‘vote’ $\alpha_t$ of $g_t$
Gradient Boosted Decision Tree
Adaptive Boosted Decision Tree

From Random Forest to AdaBoost-DTree

function **RandomForest**$(\mathcal{D})$
For $t = 1, 2, \ldots, T$
1. request size-$N'$ data $\tilde{\mathcal{D}}_t$ by bootstrapping with $\mathcal{D}$
2. obtain tree $g_t$ by Randomized-DTree$(\tilde{\mathcal{D}}_t)$

return $G = \text{Uniform}(\{g_t\})$

function **AdaBoost-DTree**$(\mathcal{D})$
For $t = 1, 2, \ldots, T$
1. reweight data by $u(t)$
2. obtain tree $g_t$ by DTree$(\mathcal{D}, u(t))$
3. calculate ‘vote’ $\alpha_t$ of $g_t$

return $G = \text{LinearHypo}(\{(g_t, \alpha_t)\})$

need: weighted DTree$(\mathcal{D}, u(t))$
From Random Forest to AdaBoost-DTree

**function RandomForest(\( \mathcal{D} \))**

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

1. request size-\( N' \) data \( \tilde{\mathcal{D}}_t \) by bootstrapping with \( \mathcal{D} \)

2. obtain tree \( g_t \) by Randomized-DTree(\( \tilde{\mathcal{D}}_t \))

return \( G = \text{Uniform}(\{g_t\}) \)

**function AdaBoost-DTree(\( \mathcal{D} \))**

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

1. reweight data by \( u(t) \)

2. obtain tree \( g_t \) by DTree(\( \mathcal{D}, u(t) \))

3. calculate ‘vote’ \( \alpha_t \) of \( g_t \)

return \( G = \text{LinearHypo}(\{(g_t, \alpha_t)\}) \)

**need:** weighted DTree(\( \mathcal{D}, u(t) \))
Weighted Decision Tree Algorithm

Weighted Algorithm

minimize (regularized) \( E^{u}_{\text{in}}(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(x_n)) \)
Weighted Decision Tree Algorithm

**Weighted Algorithm**

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

if using existing algorithm as **black box** (no modifications),
to get \( E_{in}^u \) approximately optimized......

‘Weighted’ Algorithm in Bagging

weights \( u \) expressed by bootstrap-sampled copies
Weighted Decision Tree Algorithm

**Weighted Algorithm**

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

- If using existing algorithm as **black box** (no modifications), to get \( E_{in}^u \) approximately optimized......

**‘Weighted’ Algorithm in Bagging**

- Weights \( u \) expressed by bootstrap-sampled copies
- Request size-\( N' \) data \( \tilde{D}_t \)
- By bootstrapping with \( D \)
minimize (regularized) $E_{\text{in}}^u(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(x_n))$

if using existing algorithm as black box (no modifications), to get $E_{\text{in}}^u$ approximately optimized......

A General Randomized Base Algorithm
weights $u$ expressed by sampling proportional to $u_n$

‘Weighted’ Algorithm in Bagging
weights $u$ expressed by bootstrap-sampled copies
—request size-$N'$ data $\tilde{D}_t$
by bootstrapping with $D$
Weighted Decision Tree Algorithm

**Weighted Algorithm**

\[
\text{minimize (regularized) } E^u_{in}(h) = \frac{1}{N} \sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(x_n))
\]

if using existing algorithm as **black box** (no modifications),
to get \( E^u_{in} \) approximately optimized......

**‘Weighted’ Algorithm in Bagging**

weights \( u \) expressed by bootstrap-sampled copies
—request size-\( N' \) data \( \tilde{D}_t \)
by bootstrapping with \( D \)

**A General Randomized Base Algorithm**

weights \( u \) expressed by sampling proportional to \( u_n \)
—request size-\( N' \) data \( \tilde{D}_t \)
by sampling \( \propto u \) on \( D \)
Weighted Decision Tree Algorithm

**Weighted Algorithm**

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

if using existing algorithm as **black box** (no modifications),
to get \( E_{\text{in}}^u \) approximately optimized......

**‘Weighted’ Algorithm in Bagging**

weights \( u \) expressed by bootstrap-sampled copies
—request size-\( N' \) data \( \tilde{D}_t \)
by bootstrapping with \( D \)

**A General Randomized Base Algorithm**

weights \( u \) expressed by sampling proportional to \( u_n \)
—request size-\( N' \) data \( \tilde{D}_t \)
by sampling \( \propto u \) on \( D \)

AdaBoost-DTree: often via
AdaBoost + **sampling** \( \propto u^{(t)} \) + DTree(\( \tilde{D}_t \))
without modifying DTree
AdaBoost: votes $\alpha_t = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ with weighted error rate $\epsilon_t$
Weak Decision Tree Algorithm

AdaBoost: votes $\alpha_t = \ln \diamond_t = \ln \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}$ with weighted error rate $\epsilon_t$

if fully grown tree trained on all $x_n$

$\implies E_{in}(g_t) = \text{if all } x_n \text{ different}$
Weak Decision Tree Algorithm

AdaBoost: \( \text{votes } \alpha_t = \ln \diamond_t = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \) with weighted error rate \( \epsilon_t \)

if fully grown tree trained on all \( x_n \)

\[ E_{in}(g_t) = 0 \text{ if all } x_n \text{ different} \]
AdaBoost: votes $\alpha_t = \ln \phi_t = \ln \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}$ with weighted error rate $\epsilon_t$

- if fully grown tree trained on all $x_n$
- $E_{\text{in}}(g_t) = 0$ if all $x_n$ different
- $E_{\text{in}}(g_t) =$
Weak Decision Tree Algorithm

AdaBoost: \( \text{votes } \alpha_t = \ln \delta_t = \ln \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \) with weighted error rate \( \epsilon_t \)

if fully grown tree trained on all \( x_n \)
\[ \implies E_{in}(g_t) = 0 \] if all \( x_n \) different
\[ \implies E_{in}^u(g_t) = 0 \]
Adaboost: \( \text{votes } \alpha_t = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \) with weighted error rate \( \epsilon_t \)

- if fully grown tree trained on all \( x_n \)
  \[ E_{in}(g_t) = 0 \text{ if all } x_n \text{ different} \]
  \[ E_{iu}(g_t) = 0 \]
  \[ \epsilon_t = \]
Weak Decision Tree Algorithm

Adaboost: votes $\alpha_t = \ln \hat{\epsilon}_t = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ with weighted error rate $\epsilon_t$

if fully grown tree trained on all $x_n$

$\implies E_{in}(g_t) = 0$ if all $x_n$ different

$\implies E_{in}^u(g_t) = 0$

$\implies \epsilon_t = 0$
AdaBoost: \textbf{votes} $\alpha_t = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$ with \textbf{weighted error rate} $\epsilon_t$

if fully grown tree trained on all $x_n$

$\implies E_{in}(g_t) = 0$ if all $x_n$ different

$\implies E_{in}^{u}(g_t) = 0$

$\implies \epsilon_t = 0$

$\implies \alpha_t =$
Weak Decision Tree Algorithm

AdaBoost: votes $\alpha_t = \ln \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}$ with weighted error rate $\epsilon_t$

if fully grown tree trained on all $x_n$

$\implies E_{in}(g_t) = 0$ if all $x_n$ different

$\implies E_{in}^u(g_t) = 0$

$\implies \epsilon_t = 0$

$\implies \alpha_t = \infty$ (autocracy!!)
Weak Decision Tree Algorithm

AdaBoost: votes $\alpha_t = \ln \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}$ with weighted error rate $\epsilon_t$

if fully grown tree trained on all $x_n$

$\implies E_{in}(g_t) = 0$ if all $x_n$ different

$\implies E_{in}^u(g_t) = 0$

$\implies \epsilon_t = 0$

$\implies \alpha_t = \infty$ (autocracy!!)

need: pruned tree trained on some $x_n$ to be weak
Weak Decision Tree Algorithm

AdaBoost: \textbf{votes} \( \alpha_t = \ln \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \) with \textbf{weighted error rate} \( \epsilon_t \)

If fully grown tree trained on all \( x_n \)
\[ \implies E_{in}(g_t) = 0 \text{ if all } x_n \text{ different} \]
\[ \implies E_{u}(g_t) = 0 \]
\[ \implies \epsilon_t = 0 \]
\[ \implies \alpha_t = \infty \text{ (autocracy)} \]

Need: \textbf{pruned} tree trained on some \( x_n \) to be weak

- some: sampling \( \propto u(t) \)
AdaBoost: \textbf{votes} $\alpha_t = \ln \frac{1 - \epsilon_t}{\epsilon_t}$ with weighted error rate $\epsilon_t$

If fully grown tree trained on all $x_n$

$\implies$ $E_{in}(g_t) = 0$ if all $x_n$ different

$\implies$ $E_{in}^u(g_t) = 0$

$\implies$ $\epsilon_t = 0$

$\implies$ $\alpha_t = \infty$ (autocracy!!)

Need: \textbf{pruned} tree trained on some $x_n$ to be \textbf{weak}

- \textbf{pruned}: usual pruning, or just \textbf{limiting tree height}
- \textbf{some}: sampling $\propto u^{(t)}$
Weak Decision Tree Algorithm

AdaBoost: \( \text{votes } \alpha_t = \ln \diamond_t = \ln \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \) with weighted error rate \( \epsilon_t \)

if fully grown tree trained on all \( x_n \)
\[ \implies E_{\text{in}}(g_t) = 0 \text{ if all } x_n \text{ different} \]
\[ \implies E_{\text{in}}^u(g_t) = 0 \]
\[ \implies \epsilon_t = 0 \]
\[ \implies \alpha_t = \infty \text{ (autocracy!!)} \]

need: pruned tree trained on some \( x_n \) to be weak
- pruned: usual pruning, or just limiting tree height
- some: sampling \( \propto u^{(t)} \)

AdaBoost-DTree: often via AdaBoost + sampling \( \propto u^{(t)} \) + pruned DTree(\( \tilde{D} \))
AdaBoost with Extremely-Pruned Tree

what if DTREE with $\text{height} \leq 1$ (extremely pruned)?

\[ H_{\text{AdaBoost-Stump}} = \text{special case of AdaBoost-DTree} \]

---

Hsuan-Tien Lin (NTU CSIE)
what if DTree with $\text{height} \leq 1$ (extremely pruned)?

DTree (C&RT) with $\text{height} \leq 1$

learn branching criteria

$$b(x) = \text{argmin}_{\text{decision stumps } h(x)} \sum_{c=1}^{2} |D_c \text{ with } h| \cdot \text{impurity}(D_c \text{ with } h)$$
AdaBoost with Extremely-Pruned Tree

what if DTree with $\text{height} \leq 1$ (extremely pruned)?

**DTree (C&RT) with $\text{height} \leq 1$**

Learn branching criteria

$$b(x) = \arg\min_{\text{decision stumps } h(x)} \sum_{c=1}^{2} |D_c \text{ with } h| \cdot \text{impurity}(D_c \text{ with } h)$$

—if impurity = binary classification error,

just a decision stump, remember? :-)

---

Hsuan-Tien Lin (NTU CSIE)  Machine Learning Techniques  5/25
AdaBoost with Extremely-Pruned Tree

what if DT (C&RT) with $\text{height} \leq 1$ (extremely pruned)?

**DT (C&RT) with $\text{height} \leq 1$**

learn branching criteria

$$b(x) = \arg\min_{\text{decision stumps } h(x)} \sum_{c=1}^{2} |\mathcal{D}_c \text{ with } h| \cdot \text{impurity}(\mathcal{D}_c \text{ with } h)$$

—if impurity = binary classification error,  

just a decision stump, remember? :-)

**AdaBoost-Stump**

= special case of AdaBoost-DTree
When running AdaBoost-DTree with sampling and getting a decision tree $g_t$ such that $g_t$ achieves zero error on the sampled data set $\tilde{D}_t$. Which of the following is possible?

1. $\alpha_t < 0$
2. $\alpha_t = 0$
3. $\alpha_t > 0$
4. all of the above
When running AdaBoost-DTree with sampling and getting a decision tree $g_t$ such that $g_t$ achieves zero error on the sampled data set $\tilde{D}_t$. Which of the following is possible?

1. $\alpha_t < 0$
2. $\alpha_t = 0$
3. $\alpha_t > 0$
4. all of the above

Reference Answer: 4

While $g_t$ achieves zero error on $\tilde{D}_t$, $g_t$ may not achieve zero weighted error on $(D, u^{(t)})$ and hence $\epsilon_t$ can be anything, even $\geq \frac{1}{2}$. Then, $\alpha_t$ can be $\leq 0$. 
Example Weights of AdaBoost

\[ u_n^{(t+1)} = \begin{cases} 
    u_n^{(t)} \cdot \diamond_t & \text{if incorrect} \\
    u_n^{(t)} / \diamond_t & \text{if correct}
\end{cases} \]

\[ = u_n^{(t)} \cdot \diamond_t \]
Example Weights of AdaBoost

\[
\begin{align*}
&u_n^{(t+1)} = \begin{cases} 
  u_n^{(t)} \cdot \diamond_t & \text{if incorrect} \\
  u_n^{(t)} / \diamond_t & \text{if correct}
\end{cases} \\
\end{align*}
\]

\[
= u_n^{(t)} \cdot \diamond_t - y_n g_t(x_n)
\]
Example Weights of AdaBoost

\[ u^{(t+1)}_n = \begin{cases} 
  u^{(t)}_n \cdot \diamond_t & \text{if incorrect} \\
  u^{(t)}_n / \diamond_t & \text{if correct}
\end{cases} \]

\[ = u^{(t)}_n \cdot \diamond_t (-y_ng_t(x_n)) = u^{(t)}_n \cdot (-y_n g_t(x_n)) \]
Example Weights of AdaBoost

\[
\begin{align*}
  u_n^{(t+1)} &= \begin{cases}
    u_n^{(t)} \cdot \diamond_t & \text{if incorrect} \\
    u_n^{(t)} / \diamond_t & \text{if correct}
  \end{cases} \\
  &= u_n^{(t)} \cdot \diamond_t - y_n g_t(x_n) = u_n^{(t)} \cdot \exp(-y_n \alpha t g_t(x_n))
\end{align*}
\]
Example Weights of AdaBoost

\[
\begin{align*}
    u_n^{(t+1)} &= \begin{cases} 
    u_n^{(t)} \cdot \diamond_t & \text{if incorrect} \\
    u_n^{(t)}/\diamond_t & \text{if correct} 
    \end{cases} \\
    &= u_n^{(t)} \cdot \diamond_t - y_n g_t(x_n) = u_n^{(t)} \cdot \exp\left(-y_n \alpha_t g_t(x_n)\right)
\end{align*}
\]

\[
\begin{align*}
    u_n^{(T+1)} &= u_n^{(1)} \cdot \prod_{t=1}^{T} \exp(-y_n \alpha_t g_t(x_n)) = \frac{1}{N} \cdot \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(x_n)\right)
\end{align*}
\]
Example Weights of AdaBoost

\[ u_n^{(t+1)} = \begin{cases} 
  u_n^{(t)} \cdot \diamond_t & \text{if incorrect} \\
  u_n^{(t)} / \diamond_t & \text{if correct}
\end{cases} \]

\[ = u_n^{(t)} \cdot \diamond_t - y_n g_t(x_n) = u_n^{(t)} \cdot \exp(-y_n \alpha_t g_t(x_n)) \]

\[ u_n^{(T+1)} = u_n^{(1)} \cdot \prod_{t=1}^{T} \exp(-y_n \alpha_t g_t(x_n)) = \frac{1}{N} \cdot \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(x_n)\right) \]

\[ \text{recall: } G(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(x)\right) \]
Gradient Boosted Decision Tree

Optimization View of AdaBoost

Example Weights of AdaBoost

\[ u_n^{(t+1)} = \begin{cases} 
  u_n^{(t)} \cdot \diamond_t & \text{if incorrect} \\
  u_n^{(t)} / \diamond_t & \text{if correct} 
\end{cases} \]

\[ u_n^{(t+1)} = u_n^{(t)} \cdot \diamond_t - y_n g_t(x_n) = u_n^{(t)} \cdot \exp(-y_n \alpha_t g_t(x_n)) \]

\[ u_n^{(T+1)} = u_n^{(1)} \cdot \prod_{t=1}^{T} \exp(-y_n \alpha_t g_t(x_n)) = \frac{1}{N} \cdot \exp(-y_n \sum_{t=1}^{T} \alpha_t g_t(x_n)) \]

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

- \( \sum_{t=1}^{T} \alpha_t g_t(x) \): voting score of \( \{ g_t \} \) on \( x \)
Example Weights of AdaBoost

\[
\begin{aligned}
    u_{n}^{(t+1)} &= \begin{cases} 
        u_{n}^{(t)} \cdot \diamondsuit_t & \text{if incorrect} \\
        u_{n}^{(t)} / \diamondsuit_t & \text{if correct}
    \end{cases} \\
    &= u_{n}^{(t)} \cdot \diamondsuit_t - y_{n}g_{t}(x_{n}) = u_{n}^{(t)} \cdot \exp \left( - y_{n} \alpha_{t} g_{t}(x_{n}) \right)
\end{aligned}
\]

\[
\begin{aligned}
    u_{n}^{(T+1)} = u_{n}^{(1)} \cdot \prod_{t=1}^{T} \exp \left( - y_{n} \alpha_{t} g_{t}(x_{n}) \right) = \frac{1}{N} \cdot \exp \left( - y_{n} \sum_{t=1}^{T} \alpha_{t} g_{t}(x_{n}) \right)
\end{aligned}
\]

- recall: \( G(x) = \text{sign} \left( \sum_{t=1}^{T} \alpha_{t} g_{t}(x) \right) \)

- \( \sum_{t=1}^{T} \alpha_{t} g_{t}(x) \) : voting score of \( \{g_{t}\} \) on \( x \)

AdaBoost: \( u_{n}^{(T+1)} \propto \exp \left( - y_{n} \text{ voting score on } x_{n} \right) \)
Gradient Boosted Decision Tree

Optimization View of AdaBoost

Voting Score and Margin

linear blending = \textbf{LinModel} + hypotheses as transform

\[ G(x_n) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x_n) \right) \]

\textbf{constraints}
Voting Score and Margin

linear blending = \textbf{LinModel} + hypotheses as transform + constraints

\[ G(x_n) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x_n) \right) \]

and hard-margin SVM margin = \[ y_n \cdot (w^T \phi(x_n) + b), \text{ remember? :-)} \]
Voting Score and Margin

linear blending = \textbf{LinModel} + hypotheses as transform + constraints

\begin{equation}
G(x_n) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x_n) \right)
\end{equation}

and hard-margin SVM \textbf{margin} = \frac{y_n \cdot (w^T \phi(x_n) + b)}{\|w\|}, \text{ remember? :-)}

\begin{equation}
y_n(\text{voting score}) = \text{signed \\ & unnormalized margin}
\end{equation}
Gradient Boosted Decision Tree

Optimization View of AdaBoost

Voting Score and Margin

linear blending = \text{LinModel} + hypotheses as transform + \text{constraints}

\[ G(x_n) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x_n) \right) \]

and hard-margin SVM margin = \[
\frac{y_n \cdot (w^T \phi(x_n) + b)}{||w||}, \text{ remember? :-)}
\]

\[ y_n(\text{voting score}) = \text{signed & unnormalized margin} \]

want \[ y_n(\text{voting score}) \text{ positive & large} \]
Voting Score and Margin

**Linear Blending:**

\[ \text{linear blending} = \text{LinModel} + \text{hypotheses as transform} + \text{constraints} \]

\[
G(x_n) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x_n) \right)
\]

**Voting Score:**

\[
G(x_n) = \text{sign} \left( \sum_{t=1}^{T} \alpha_t g_t(x_n) \right)
\]

**Hard-Margin SVM Margin:**

\[
\text{margin} = \frac{y_n \cdot (w^T \phi(x_n) + b)}{\|w\|}, \text{ remember? :-)}
\]

\[
y_n(\text{voting score}) = \text{signed & unnormalized margin}
\]

**Claim:** AdaBoost decreases

\[
\sum_{n=1}^{N} u(t) = \text{small}
\]

- want \( y_n(\text{voting score}) \) **positive & large**
- \( \Leftrightarrow \) \( \exp(-y_n(\text{voting score})) \) **small**
Voting Score and Margin

linear blending = \textbf{LinModel} + hypotheses as transform + constraints

\[
G(x_n) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \frac{g_t(x_n)}{w_i} \phi_i(x_n)\right)
\]

and hard-margin SVM \textbf{margin} = \[ y_n \cdot \left(\frac{w^T \phi(x_n) + b}{\|w\|}\right) \text{, remember? :-)} \]

\[ y_n(\text{voting score}) = \text{signed & unnormalized margin} \]

\(\Leftrightarrow\) \[ \exp(-y_n(\text{voting score})) \text{ small} \]
\(\Leftrightarrow\) \[ u_n^{(T+1)} \text{ small} \]
Voting Score and Margin

linear blending = $\text{LinModel} + \text{hypotheses as transform} + \text{constraints}$

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

and hard-margin SVM margin = $y_n \cdot (w^T \phi(x_n) + b)$, remember? :-)

$y_n(\text{voting score}) = \text{signed & unnormalized margin}$

want $y_n(\text{voting score})$ positive & large

$\iff \exp(-y_n(\text{voting score}))$ small

$\iff u_n^{(T+1)}$ small

claim: AdaBoost decreases $\sum_{n=1}^{N} u_n^{(t)}$
claim: AdaBoost decreases \( \sum_{n=1}^{N} u_n^{(t)} \) and thus somewhat minimizes

\[
\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \sum_{t=1}^{T} \alpha_t g_t(x_n) \right)
\]
claim: AdaBoost decreases \( \sum_{n=1}^{N} u_n^{(t)} \) and thus somewhat \textbf{minimizes}

\[
\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \sum_{t=1}^{T} \alpha_t g_t(x_n) \right)
\]

linear score \( s = \sum_{t=1}^{T} \alpha_t g_t(x_n) \)
claim: AdaBoost decreases $\sum_{n=1}^{N} u^{(t)}_n$ and thus somewhat **minimizes**

$$\sum_{n=1}^{N} u^{(T+1)}_n = \frac{1}{N} \sum_{n=1}^{N} \exp \left(-y_n \sum_{t=1}^{T} \alpha_t g_t(x_n) \right)$$

linear score $s = \sum_{t=1}^{T} \alpha_t g_t(x_n)$

- $\text{err}_{0/1}(s, y) = [ys \leq 0]$
AdaBoost Error Function

**Claim:** AdaBoost decreases \( \sum_{n=1}^{N} u_n^{(T)} \) and thus somewhat **minimizes**

\[
\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \sum_{t=1}^{T} \alpha_t g_t(x_n) \right)
\]

**Linear Score:**
\[
s = \sum_{t=1}^{T} \alpha_t g_t(x_n)
\]

- \( \text{err}_{0/1}(s, y) = [ys \leq 0] \)
- \( \hat{\text{err}}_{\text{ADA}}(s, y) = \exp(-ys) \): upper bound of \( \text{err}_{0/1} \)
  —called **exponential error measure**
Gradient Boosted Decision Tree

Optimization View of AdaBoost

**AdaBoost Error Function**

Claim: AdaBoost decreases \( \sum_{n=1}^{N} u_n^{(t)} \) and thus somewhat minimizes

\[
\sum_{n=1}^{N} u_n^{(T+1)} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \sum_{t=1}^{T} \alpha_t g_t(x_n) \right)
\]

Linear score \( s = \sum_{t=1}^{T} \alpha_t g_t(x_n) \)

- \( \text{err}_{0/1}(s, y) = \mathbb{I}[ys \leq 0] \)
- \( \hat{\text{err}}_{\text{ADA}}(s, y) = \exp(-ys) \): upper bound of \( \text{err}_{0/1} \)
  —called exponential error measure

\( \hat{\text{err}}_{\text{ADA}} \): algorithmic error measure
by convex upper bound of \( \text{err}_{0/1} \)
Gradient Descent on AdaBoost Error Function

recall: gradient descent (remember? :-)), at iteration $t$

$$\min_{\|v\|=1} E_{\text{in}}(w_t + \eta v) \approx E_{\text{in}}(w_t) + \eta v^T \nabla E_{\text{in}}(w_t)$$

- Known
- Given positive
- Known
Gradient Descent on AdaBoost Error Function

recall: gradient descent \((\text{remember? :-)})\), at iteration \(t\)

\[
\min_{\|v\|=1} E_{\text{in}}(w_t + \eta v) \approx E_{\text{in}}(w_t) + \eta v^T \nabla E_{\text{in}}(w_t)
\]

\(\text{known}\) \(\text{given positive}\) \(\text{known}\)

at iteration \(t\), to find \(g_t\), solve

\[
\min_h \hat{E}_{\text{ADA}} = \frac{1}{N} \sum_{n=1}^{N} \exp\left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)
\]
Gradient Boosting Decision Tree
Optimization View of AdaBoost

Gradient Descent on AdaBoost Error Function

recall: gradient descent (remember? :-)), at iteration $t$

$$
\min_{\|v\|=1} E_{\text{in}}(w_t + \eta v) \approx E_{\text{in}}(w_t) + \eta v^T \nabla E_{\text{in}}(w_t)
$$

at iteration $t$, to find $g_t$, solve

$$
\min_h \widehat{E}_{\text{ADA}} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n) + \eta h(x_n) \right) \right) \\
= \sum_{n=1}^{N} \exp \left( -y_n \eta h(x_n) \right)
$$
Gradient Descent on AdaBoost Error Function

Recall: gradient descent (remember? :-)), at iteration $t$

$$\min_{\|v\|=1} E_{\text{in}}(w_t + \eta v) \approx E_{\text{in}}(w_t) + \eta v^T \nabla E_{\text{in}}(w_t)$$

At iteration $t$, to find $g_t$, solve

$$\min_h \; \hat{E}_{\text{ADA}} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n) + \eta h(x_n) \right) \right)$$

$$= \sum_{n=1}^{N} u_n^{(t)} \exp \left( -y_n \eta h(x_n) \right)$$
Gradient Descent on AdaBoost Error Function

recall: gradient descent (remember? :-)), at iteration $t$

\[
\min_{\|v\|=1} E_{\text{in}}(w_t + \eta v) \approx E_{\text{in}}(w_t) + \eta v^T \nabla E_{\text{in}}(w_t)
\]

at iteration $t$, to find $g_t$, solve

\[
\min_h \hat{E}_{\text{ADA}} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)
\]

\[
= \sum_{n=1}^{N} u_n^{(t)} \exp \left( -y_n \eta h(x_n) \right)
\]

taylor

\[
\approx \sum_{n=1}^{N} u_n^{(t)} ( )
\]
Gradient Descent on AdaBoost Error Function

recall: gradient descent (remember? :-)), at iteration $t$

$$\min_{\|v\|=1} E_{\text{in}}(w_t + \eta v) \approx E_{\text{in}}(w_t) + \eta v^T \nabla E_{\text{in}}(w_t)$$

known

given positive

known

at iteration $t$, to find $g_t$, solve

$$\min_h \hat{E}_{\text{ADA}} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)$$

$$= \sum_{n=1}^{N} u^{(t)}_n \exp (-y_n \eta h(x_n))$$

taylor

$$\approx \sum_{n=1}^{N} u^{(t)}_n (1 - y_n \eta h(x_n))$$
Gradient Descent on AdaBoost Error Function

Recall: gradient descent (remember? :-)), at iteration $t$

$$\min_{\|v\|=1} E_{in}(w_t + \eta v) \approx E_{in}(w_t) + \eta v^T \nabla E_{in}(w_t)$$

At iteration $t$, to find $g_t$, solve

$$\min_h \tilde{E}_{ADA} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)$$

$$= \sum_{n=1}^{N} u^{(t)}_n \exp \left( -y_n \eta h(x_n) \right)$$

Taylor

$$\approx \sum_{n=1}^{N} u^{(t)}_n \left( 1 - y_n \eta h(x_n) \right) = \sum_{n=1}^{N} u^{(t)}_n - \eta \sum_{n=1}^{N} u^{(t)}_n$$
Gradient Descent on AdaBoost Error Function

recall: gradient descent (remember? :-)), at iteration $t$

$$
\min_{\|v\|=1} E_{in}(w_t + \eta v) \approx E_{in}(w_t) + \eta v^T \nabla E_{in}(w_t)
$$

at iteration $t$, to find $g_t$, solve

$$
\min_h \hat{E}_{ADA} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)
$$

$$
= \sum_{n=1}^{N} u_n^{(t)} \exp \left( -y_n \eta h(x_n) \right)
$$

taylor

$$
\approx \sum_{n=1}^{N} u_n^{(t)} \left( 1 - y_n \eta h(x_n) \right) = \sum_{n=1}^{N} u_n^{(t)} - \eta \sum_{n=1}^{N} u_n^{(t)} y_n h(x_n)
$$
Gradient Boosted Decision Tree

Optimization View of AdaBoost

Gradient Descent on AdaBoost Error Function

recall: gradient descent (remember? :-)), at iteration \( t \)

\[
\min_{\|v\|=1} E_{\text{in}}(w_t + \eta v) \approx E_{\text{in}}(w_t) + \underbrace{\eta v^T \nabla E_{\text{in}}(w_t)}_{\text{known}}
\]

at iteration \( t \), to find \( g_t \), solve

\[
\min_{h} \hat{E}_{\text{ADA}} = \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n) + \eta h(x_n) \right) \right)
\]

\[
= \sum_{n=1}^{N} u_{n}^{(t)} \exp \left( -y_n \eta h(x_n) \right)
\]

taylor

\[
\approx \sum_{n=1}^{N} u_{n}^{(t)} \left( 1 - y_n \eta h(x_n) \right) = \sum_{n=1}^{N} u_{n}^{(t)} - \eta \sum_{n=1}^{N} u_{n}^{(t)} y_n h(x_n)
\]

good \( h \): minimize \( \sum_{n=1}^{N} u_{n}^{(t)} (-y_n h(x_n)) \)
Learning Hypothesis as Optimization

finding good $h$ (function direction) ⇔ minimize $\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n))$
Learning Hypothesis as Optimization

finding good $h$ (function direction) $\iff$ minimize $\sum_{n=1}^{N} u_n^{(t)} (-y_nh(x_n))$

for binary classification, where $y_n$ and $h(x_n)$ both $\in \{-1, +1\}$:
Learning Hypothesis as Optimization

finding good $h$ (function direction) $\iff$ minimize $\sum_{n=1}^{N} u^{(t)}_n (-y_n h(x_n))$

for binary classification, where $y_n$ and $h(x_n)$ both $\in \{-1, +1\}$:

$$\sum_{n=1}^{N} u^{(t)}_n (-y_n h(x_n)) = \sum_{n=1}^{N} u^{(t)}_n \begin{cases} \text{if } y_n = h(x_n) \\ \text{if } y_n \neq h(x_n) \end{cases}$$
Learning Hypothesis as Optimization

finding good $h$ (function direction) $\iff$ minimize $\sum_{n=1}^{N} u_{n}^{(t)} (-y_{n} h(x_{n}))$

for binary classification, where $y_{n}$ and $h(x_{n})$ both $\in \{-1, +1\}$:

$$\sum_{n=1}^{N} u_{n}^{(t)} (-y_{n} h(x_{n})) = \sum_{n=1}^{N} u_{n}^{(t)} \left\{ \begin{array}{ll} -1 & \text{if } y_{n} = h(x_{n}) \\ +1 & \text{if } y_{n} \neq h(x_{n}) \end{array} \right.$$
Learning Hypothesis as Optimization

finding good \( h \) (function direction) ⇔ minimize \( \sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n)) \)

for binary classification, where \( y_n \) and \( h(x_n) \) both \( \in \{ -1, +1 \} \):

\[
\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n)) = \sum_{n=1}^{N} u_n^{(t)} \left\{ \begin{array}{ll}
-1 & \text{if } y_n = h(x_n) \\
+1 & \text{if } y_n \neq h(x_n)
\end{array} \right.
\]

\[
= -\sum_{n=1}^{N} u_n^{(t)} + \sum_{n=1}^{N} u_n^{(t)} \left\{ \begin{array}{ll}
& \text{if } y_n = h(x_n) \\
& \text{if } y_n \neq h(x_n)
\end{array} \right.
\]
Learning Hypothesis as Optimization

Finding good $h$ (function direction) $\iff$ minimize $\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n))$

For binary classification, where $y_n$ and $h(x_n)$ both $\in \{-1, +1\}$:

$$\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n)) = \sum_{n=1}^{N} u_n^{(t)} \begin{cases} -1 & \text{if } y_n = h(x_n) \\ +1 & \text{if } y_n \neq h(x_n) \end{cases} = -\sum_{n=1}^{N} u_n^{(t)} + \sum_{n=1}^{N} u_n^{(t)} \begin{cases} 0 & \text{if } y_n = h(x_n) \\ 2 & \text{if } y_n \neq h(x_n) \end{cases}$$
Learning Hypothesis as Optimization

finding good $h$ (function direction) $\iff$ minimize $\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n))$

for binary classification, where $y_n$ and $h(x_n)$ both $\in \{-1, +1\}$:

$$\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n)) = \sum_{n=1}^{N} u_n^{(t)} \begin{cases} -1 & \text{if } y_n = h(x_n) \\ +1 & \text{if } y_n \neq h(x_n) \end{cases}$$

$$= -\sum_{n=1}^{N} u_n^{(t)} + \sum_{n=1}^{N} u_n^{(t)} \begin{cases} 0 & \text{if } y_n = h(x_n) \\ 2 & \text{if } y_n \neq h(x_n) \end{cases}$$

$$= -\sum_{n=1}^{N} u_n^{(t)} + 2 \cdot N$$
Learning Hypothesis as Optimization

finding good $h$ (function direction) $\Leftrightarrow$ minimize $\sum_{n=1}^{N} u_{n}^{(t)} (-y_{n} h(x_{n}))$

for binary classification, where $y_{n}$ and $h(x_{n})$ both $\in \{-1, +1\}$:

$$\sum_{n=1}^{N} u_{n}^{(t)} (-y_{n} h(x_{n})) = \sum_{n=1}^{N} u_{n}^{(t)} \begin{cases} -1 & \text{if } y_{n} = h(x_{n}) \\ +1 & \text{if } y_{n} \neq h(x_{n}) \end{cases}$$

$$= - \sum_{n=1}^{N} u_{n}^{(t)} + \sum_{n=1}^{N} u_{n}^{(t)} \begin{cases} 0 & \text{if } y_{n} = h(x_{n}) \\ 2 & \text{if } y_{n} \neq h(x_{n}) \end{cases}$$

$$= - \sum_{n=1}^{N} u_{n}^{(t)} + 2 E_{\text{in}}^{u(t)}(h) \cdot N$$
Finding good $h$ (function direction) $\Leftrightarrow$ minimize $\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n))$

For binary classification, where $y_n$ and $h(x_n)$ both $\in \{-1, +1\}$:

$$\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n)) = \sum_{n=1}^{N} u_n^{(t)} \begin{cases} -1 & \text{if } y_n = h(x_n) \\ +1 & \text{if } y_n \neq h(x_n) \end{cases}$$

$$= - \sum_{n=1}^{N} u_n^{(t)} + \sum_{n=1}^{N} u_n^{(t)} \begin{cases} 0 & \text{if } y_n = h(x_n) \\ 2 & \text{if } y_n \neq h(x_n) \end{cases}$$

$$= - \sum_{n=1}^{N} u_n^{(t)} + 2E_{in}^{u(t)}(h) \cdot N$$

—Who minimizes $E_{in}^{u(t)}(h)$?
finding good $h$ (function direction) $\iff$ minimize $\sum_{n=1}^{N} u^{(t)}_n (-y_n h(x_n))$

for binary classification, where $y_n$ and $h(x_n)$ both $\in \{-1, +1\}$:

$$\sum_{n=1}^{N} u^{(t)}_n (-y_n h(x_n)) = \sum_{n=1}^{N} u^{(t)}_n \left\{ \begin{array}{ll} -1 & \text{if } y_n = h(x_n) \\ +1 & \text{if } y_n \neq h(x_n) \end{array} \right.$$  

$$= -\sum_{n=1}^{N} u^{(t)}_n + \sum_{n=1}^{N} u^{(t)}_n \left\{ \begin{array}{ll} 0 & \text{if } y_n = h(x_n) \\ 2 & \text{if } y_n \neq h(x_n) \end{array} \right.$$  

$$= -\sum_{n=1}^{N} u^{(t)}_n + 2E_{in}^{u^{(t)}}(h) \cdot N$$

—who minimizes $E_{in}^{u^{(t)}}(h)$? $A$ in AdaBoost! :-(
Learning Hypothesis as Optimization

finding good $h$ (function direction) $\iff$ minimize $\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n))$

for binary classification, where $y_n$ and $h(x_n)$ both $\in \{-1, +1\}$:

$$
\sum_{n=1}^{N} u_n^{(t)} (-y_n h(x_n)) = \sum_{n=1}^{N} u_n^{(t)} \begin{cases} 
-1 & \text{if } y_n = h(x_n) \\
+1 & \text{if } y_n \neq h(x_n)
\end{cases}
$$

$$
= - \sum_{n=1}^{N} u_n^{(t)} + \sum_{n=1}^{N} u_n^{(t)} \begin{cases} 
0 & \text{if } y_n = h(x_n) \\
2 & \text{if } y_n \neq h(x_n)
\end{cases}
$$

$$
= - \sum_{n=1}^{N} u_n^{(t)} + 2E_{\text{in}}^{u(t)}(h) \cdot N
$$

—who minimizes $E_{\text{in}}^{u(t)}(h)$? $A$ in AdaBoost! :-)

$A$: good $g_t = h$ for ‘gradient descent’
Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately

$$\min_h \hat{E}_{ADA} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta h(x_n))$$
Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately

$$\min_h \hat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta h(x_n))$$

after finding $g_t$, how about

$$\min_\eta \hat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta g_t(x_n))$$
Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately minimizing

$$\hat{E}_{ADA} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta h(x_n))$$

after finding $g_t$, how about minimizing

$$\min_{\eta} \hat{E}_{ADA} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta g_t(x_n))$$

- optimal $\eta_t$ somewhat ‘greedily faster’ than fixed (small) $\eta$
Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately

$$\min_h \hat{E}_{ADA} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta h(x_n))$$

after finding $g_t$, how about

$$\min_\eta \hat{E}_{ADA} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta g_t(x_n))$$

- optimal $\eta_t$ somewhat ‘greedily faster’ than fixed (small) $\eta$
  —called steepest descent for optimization
Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately minimizing $\min_h \hat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp(-y_n \eta h(x_n))$

after finding $g_t$, how about $\min_\eta \hat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp(-y_n \eta g_t(x_n))$

- optimal $\eta_t$ somewhat ‘greedily faster’ than fixed (small) $\eta$
  —called steepest descent for optimization
- two cases inside summation:
  - $y_n = g_t(x_n)$: (correct)
  - $y_n \neq g_t(x_n)$: (incorrect)
Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately minimizing

$$
\hat{E}_{\text{ADA}} = \min_{h} \sum_{n=1}^{N} u_n(t) \exp \left( -y_n \eta h(x_n) \right)
$$

after finding $g_t$, how about

$$
\min_{\eta} \hat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n(t) \exp \left( -y_n \eta g_t(x_n) \right)
$$

- optimal $\eta_t$ somewhat ‘greedily faster’ than fixed (small) $\eta$
  —called **steepest** descent for optimization
- two cases inside summation:
  - $y_n = g_t(x_n)$: $u_n(t) \exp (-\eta)$ (correct)
  - $y_n \neq g_t(x_n)$: $u_n(t) \exp (+\eta)$ (incorrect)
AdaBoost finds $g_t$ by approximately minimizing $\hat{E}_{ADA} = \sum_{h=1}^{N} u_n^{(t)} \exp (-y_n \eta h(x_n))$.

After finding $g_t$, how about minimizing $\min_{\eta} \hat{E}_{ADA} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta g_t(x_n))$?

- Optimal $\eta_t$ somewhat 'greedily faster' than fixed (small) $\eta$—called **steepest** descent for optimization.
- Two cases inside summation:
  - $y_n = g_t(x_n): u_n^{(t)} \exp (-\eta)$ (correct)
  - $y_n \neq g_t(x_n): u_n^{(t)} \exp (+\eta)$ (incorrect)

$$\hat{E}_{ADA} = \left( \sum_{n=1}^{N} u_n^{(t)} \right) \cdot \left( \exp (-\eta) + \exp (+\eta) \right)$$
Gradient Boosted Decision Tree

Optimization View of AdaBoost

Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately

$$\min_h \hat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta h(x_n))$$

after finding $g_t$, how about

$$\min_\eta \hat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta g_t(x_n))$$

- optimal $\eta_t$ somewhat 'greedily faster' than fixed (small) $\eta$
  —called steepest descent for optimization
- two cases inside summation:
  - $y_n = g_t(x_n) : u_n^{(t)} \exp (-\eta)$ (correct)
  - $y_n \neq g_t(x_n) : u_n^{(t)} \exp (+\eta)$ (incorrect)
- $$\hat{E}_{\text{ADA}} = \left( \sum_{n=1}^{N} u_n^{(t)} \right) \cdot \left( (1 - \epsilon_t) \exp (-\eta) + \epsilon_t \exp (+\eta) \right)$$
Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately

$$
\hat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp \left( -y_n \eta h(x_n) \right)
$$

after finding $g_t$, how about

$$
\min_{\eta} \hat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp \left( -y_n \eta g_t(x_n) \right)
$$

- optimal $\eta_t$ somewhat ‘greedily faster’ than fixed (small) $\eta$
  —called steepest descent for optimization
- two cases inside summation:
  - $y_n = g_t(x_n)$: $u_n^{(t)} \exp (-\eta)$ (correct)
  - $y_n \neq g_t(x_n)$: $u_n^{(t)} \exp (+\eta)$ (incorrect)
- $\hat{E}_{\text{ADA}} = \left( \sum_{n=1}^{N} u_n^{(t)} \right) \cdot \left( (1 - \epsilon_t) \exp (-\eta) + \epsilon_t \exp (+\eta) \right)$

by solving $\frac{\partial \hat{E}_{\text{ADA}}}{\partial \eta} = 0$, steepest $\eta_t = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = $
Deciding Blending Weight as Optimization

**Gradient Boosted Decision Tree**

**Optimization View of AdaBoost**

---

AdaBoost finds $g_t$ by approximately

$$\widehat{E}_{\text{ADA}} = \min_{h} \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta h(x_n))$$

after finding $g_t$, how about

$$\min_{\eta} \widehat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp (-y_n \eta g_t(x_n))$$

---

- optimal $\eta_t$ somewhat ‘**greedily faster**’ than fixed (small) $\eta$ —called **steepest** descent for optimization

- two cases inside summation:
  - $y_n = g_t(x_n) : u_n^{(t)} \exp (-\eta)$ (correct)
  - $y_n \neq g_t(x_n) : u_n^{(t)} \exp (+\eta)$ (incorrect)

- $\widehat{E}_{\text{ADA}} = \left( \sum_{n=1}^{N} u_n^{(t)} \right) \cdot \left( (1 - \epsilon_t) \exp (-\eta) + \epsilon_t \exp (+\eta) \right)$

---

by solving $\frac{\partial \widehat{E}_{\text{ADA}}}{\partial \eta} = 0$, **steepest** $\eta_t = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \alpha_t$, *remember?* :-)}
Gradient Boosted Decision Tree

Optimization View of AdaBoost

Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately

$$\min_h \hat{E}_{ADA} = \sum_{n=1}^{N} u^{(t)}_n \exp (-y_n \eta h(x_n))$$

after finding $g_t$, how about

$$\min_\eta \hat{E}_{ADA} = \sum_{n=1}^{N} u^{(t)}_n \exp (-y_n \eta g_t(x_n))$$

- optimal $\eta_t$ somewhat ‘greedily faster’ than fixed (small) $\eta$
  —called steepest descent for optimization
- two cases inside summation:
  - $y_n = g_t(x_n): u^{(t)}_n \exp (-\eta)$
    (correct)
  - $y_n \neq g_t(x_n): u^{(t)}_n \exp (+\eta)$
    (incorrect)
- $\hat{E}_{ADA} = \left( \sum_{n=1}^{N} u^{(t)}_n \right) \cdot \left( (1 - \epsilon_t) \exp (-\eta) + \epsilon_t \exp (+\eta) \right)$

by solving $\frac{\partial \hat{E}_{ADA}}{\partial \eta} = 0$, steepest $\eta_t = \ln \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} = \alpha_t$, remember? :-)

—AdaBoost: steepest descent with approximate functional gradient
With $\hat{E}_{\text{ADA}} = \left( \sum_{n=1}^{N} u_n^{(t)} \right) \cdot \left( (1 - \epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \right)$, which of the following is $\frac{\partial \hat{E}_{\text{ADA}}}{\partial \eta}$ that can be used for solving the optimal $\eta_t$?

1. $\left( \sum_{n=1}^{N} u_n^{(t)} \right) \cdot \left( (1 - \epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \right)$
2. $\left( \sum_{n=1}^{N} u_n^{(t)} \right) \cdot \left( (1 - \epsilon_t) \exp(-\eta) - \epsilon_t \exp(+\eta) \right)$
3. $\left( \sum_{n=1}^{N} u_n^{(t)} \right) \cdot \left( -(1 - \epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \right)$
4. $\left( \sum_{n=1}^{N} u_n^{(t)} \right) \cdot \left( -(1 - \epsilon_t) \exp(-\eta) - \epsilon_t \exp(+\eta) \right)$

Reference Answer: 3

Differentiate $\exp(-\eta)$ and $\exp(+\eta)$ with respect to $\eta$ and you should easily get the result.
With $\hat{E}_{\text{ADA}} = \left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left( (1 - \epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \right)$, which of the following is $\frac{\partial \hat{E}_{\text{ADA}}}{\partial \eta}$ that can be used for solving the optimal $\eta_t$?

1. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left( + (1 - \epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \right)$
2. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left( + (1 - \epsilon_t) \exp(-\eta) - \epsilon_t \exp(+\eta) \right)$
3. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left( - (1 - \epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \right)$
4. $\left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \left( - (1 - \epsilon_t) \exp(-\eta) - \epsilon_t \exp(+\eta) \right)$

Reference Answer: 3

Differentiate $\exp(-\eta)$ and $\exp(+\eta)$ with respect to $\eta$ and you should easily get the result.
Gradient Boosting for Arbitrary Error Function

**AdaBoost**

\[
\min_{\eta} \min_{h} \min \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)
\]

with binary-output hypothesis \( h \)
Gradient Boosting for Arbitrary Error Function

**AdaBoost**

\[
\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)
\]

with binary-output hypothesis \( h \)

**GradientBoost**

Gradient Boosted Decision Tree
Gradient Boosting for Arbitrary Error Function

### AdaBoost

$$\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)$$

with binary-output hypothesis $h$

### GradientBoost

$$\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err} \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n), y_n \right)$$
Gradient Boosting for Arbitrary Error Function

**AdaBoost**

\[
\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)
\]

with binary-output hypothesis \( h \)

**GradientBoost**

\[
\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err} \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n), y_n \right)
\]

with any hypothesis \( h \) (usually real-output hypothesis)
Gradient Boosting for Arbitrary Error Function

**AdaBoost**

\[
\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \exp \left( -y_n \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n) \right) \right)
\]

with binary-output hypothesis \( h \)

**GradientBoost**

\[
\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err} \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n), y_n \right)
\]

with any hypothesis \( h \) (usually real-output hypothesis)

GradientBoost: allows extension to different \text{err} for regression/soft classification/etc.
GradientBoost for Regression

\[
\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err}\left( \sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n) + \eta h(x_n), y_n \right) + \text{err}(s_n, y_n)
\]

with \( \text{err}(s, y) = (s - y)^2 \)
Gradient Boosting for Regression

\[
\min_{\eta} \min_{h} \frac{1}{N} \sum_{n=1}^{N} \text{err}\left(\sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n), y_n\right) \]

with \( \text{err}(s, y) = (s - y)^2 \)

\[
\min \ldots \approx \min_{h} \frac{1}{N} \sum_{n=1}^{N} \text{err}(s_n, y_n) + \frac{1}{N} \sum_{n=1}^{N} \eta h(x_n) \]

(\text{constant})
GradientBoost for Regression

\[
\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err}\left(\sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n), y_n\right) + \eta h(x_n)
\]

with \(\text{err}(s, y) = (s - y)^2\)

\[
\min_{h} \approx \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err}(s_n, y_n) + \frac{1}{N} \sum_{n=1}^{N} \eta h(x_n) \frac{\partial \text{err}(s, y_n)}{\partial s} \bigg|_{s=s_n}
\]
GradientBoost for Regression

\[ \min_{\eta} \min_{h} \frac{1}{N} \sum_{n=1}^{N} \text{err}\left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n), y_n \right) \]

with \( \text{err}(s, y) = (s - y)^2 \)

\[ \approx \min_{h} \frac{1}{N} \sum_{n=1}^{N} \text{err}(s_n, y_n) + \frac{1}{N} \sum_{n=1}^{N} \eta h(x_n) \]

\[ \frac{\partial \text{err}(s, y_n)}{\partial s} \bigg|_{s=s_n} \]

\[ = \min_{h} \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} h(x_n) \cdot \text{constant} \]
Gradient Boosting for Regression

\[
\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err}\left(\sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n), y_n\right) 
\]

with \(\text{err}(s, y) = (s - y)^2\)

\[
\approx \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err}\left(s_n, y_n\right) + \frac{1}{N} \sum_{n=1}^{N} \eta h(x_n) \left. \frac{\partial \text{err}(s, y_n)}{\partial s} \right|_{s=s_n}
\]

\[
= \min_h \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} h(x_n) \cdot 2(s_n - y_n)
\]
Gradient Boosting

Gradient Boost for Regression

\[
\min_{\eta} \min_h \left( \frac{1}{N} \sum_{n=1}^{N} \text{err} \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n), y_n \right) \right) \quad \text{with} \quad \text{err}(s, y) = (s - y)^2
\]

\[
\min_h \left( \frac{1}{N} \sum_{n=1}^{N} \text{err}(s_n, y_n) + \frac{1}{N} \sum_{n=1}^{N} \eta h(x_n) \right) \quad \text{constant} \quad \frac{\partial \text{err}(s, y_n)}{\partial s} \bigg|_{s=s_n}
\]

\[
= \min_h \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} h(x_n) \cdot 2(s_n - y_n)
\]

naïve solution \( h(x_n) = - (s_n - y_n) \)

if no constraint on \( h \)
Gradient Boosting for Regression

\[
\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err}\left(\sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta h(x_n), y_n\right) \]

with \(\text{err}(s, y) = (s - y)^2\)

\[
\text{taylor} \approx \min \frac{1}{N} \sum_{n=1}^{N} \text{err}\left(\text{const}, y_n\right) + \frac{1}{N} \sum_{n=1}^{N} \eta h(x_n) \cdot \left. \frac{\partial \text{err}(s, y_n)}{\partial s} \right|_{s=s_n}
\]

\[
= \min \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} h(x_n) \cdot 2(s_n - y_n)
\]

naïve solution \(h(x_n) = -\infty \cdot (s_n - y_n)\)

if no constraint on \(h\)
Learning Hypothesis as Optimization

\[
\min_h \text{ constants} + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]
Learning Hypothesis as Optimization

\[
\min_h \text{ constants} + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]

- **magnitude** of \( h \) does not matter:
Learning Hypothesis as Optimization

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]

- magnitude of \( h \) does not matter: because \( \eta \) will be optimized next
Learning Hypothesis as Optimization

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]

- magnitude of \( h \) does not matter: because \( \eta \) will be optimized next
- penalize large magnitude to avoid naïve solution
Learning Hypothesis as Optimization

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]

- magnitude of \( h \) does not matter: because \( \eta \) will be optimized next
- penalize large magnitude to avoid naïve solution

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} (2h(x_n)(s_n - y_n) + (h(x_n))^2)
\]
Gradient Boosted Decision Tree

Gradient Boosting

Learning Hypothesis as Optimization

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]

- **magnitude** of \( h \) does not matter: because \( \eta \) will be optimized next
- **penalize large magnitude** to avoid naïve solution

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} \left( 2h(x_n)(s_n - y_n) + (h(x_n))^2 \right)
\]

\[
= \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} \left( \right)
\]
Gradient Boosting

Learning Hypothesis as Optimization

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]

- **magnitude** of \( h \) does not matter: because \( \eta \) will be optimized next
- **penalize large magnitude** to avoid naïve solution

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} \left( 2h(x_n)(s_n - y_n) + (h(x_n))^2 \right)
\]

\[
= \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} \left( \text{constant} + (h(x_n) - (y_n - s_n))^2 \right)
\]
Learning Hypothesis as Optimization

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]

- magnitude of \( h \) does not matter: because \( \eta \) will be optimized next
- penalize large magnitude to avoid naïve solution

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} \left( 2h(x_n)(s_n - y_n) + (h(x_n))^2 \right)
\]

\[
= \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} \left( \text{constant} + (h(x_n) - (y_n - s_n))^2 \right)
\]

- solution of penalized approximate functional gradient:
  squared-error regression on \( \{(x_n, y_n - s_n)\} \)
Gradient Boosted Decision Tree

Gradient Boosting

Learning Hypothesis as Optimization

\[
\min_h \text{ constants} + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]

- **magnitude** of \( h \) does not matter: because \( \eta \) will be optimized next
- **penalize large magnitude** to avoid naïve solution

\[
\min_h \text{ constants} + \frac{\eta}{N} \sum_{n=1}^{N} (2h(x_n)(s_n - y_n) + (h(x_n))^2)
\]

\[
= \text{ constants} + \frac{\eta}{N} \sum_{n=1}^{N} \left( \text{constant} + (h(x_n) - (y_n - s_n))^2 \right)
\]

- **solution of penalized approximate functional gradient**: squared-error regression on \( \{(x_n, y_n - s_n)\} \)

Hsuan-Tien Lin (NTU CSIE) 16/25
Learning Hypothesis as Optimization

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} 2h(x_n)(s_n - y_n)
\]

- **magnitude** of \( h \) does not matter: because \( \eta \) will be optimized next
- **penalize large magnitude** to avoid naïve solution

\[
\min_h \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} (2h(x_n)(s_n - y_n) + (h(x_n))^2)
\]

\[
= \text{ constants } + \frac{\eta}{N} \sum_{n=1}^{N} \left( \text{constant } + (h(x_n) - (y_n - s_n))^2 \right)
\]

- **solution of penalized approximate functional gradient**: squared-error regression on \( \{(x_n, y_n - s_n)\} \)
Deciding Blending Weight as Optimization

after finding $g_t = h$, 

$$\min_{\eta} \min_{n} \frac{1}{N} \sum_{n=1}^{N} \text{err} \left( \sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n) + \eta g_t(x_n), y_n \right)$$

with $\text{err}(s, y) = (s - y)^2$
Deciding Blending Weight as Optimization

after finding \( g_t = h \),

\[
\min_{\eta} \min_{xn} \frac{1}{N} \sum_{n=1}^{N} \text{err}\left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta g_t(x_n), y_n \right)
\]

\[
\text{with } \text{err}(s, y) = (s - y)^2
\]

\[
\min_{\eta} \frac{1}{N} \sum_{n=1}^{N} (s_n + \eta g_t(x_n) - y_n)^2
\]
Deciding Blending Weight as Optimization

after finding $g_t = h$, 

$$\min_{\eta} \min_n \frac{1}{N} \sum_{n=1}^{N} \text{err}\left(\sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta g_t(x_n), y_n\right)$$

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

\[ \min_{\eta} \frac{1}{N} \sum_{n=1}^{N} (s_n + \eta g_t(x_n) - y_n)^2 = \frac{1}{N} \sum_{n=1}^{N} (s_n - \eta g_t(x_n))^2 \]
Deciding Blending Weight as Optimization

after finding $g_t = h$,

$$\min_{\eta} \min_h \frac{1}{N} \sum_{n=1}^{N} \text{err} \left( \sum_{\tau=1}^{t-1} \alpha_{\tau} g_{\tau}(x_n) + \eta g_t(x_n), y_n \right)$$

with \( \text{err}(s, y) = (s - y)^2 \)

and

$$\min_{\eta} \frac{1}{N} \sum_{n=1}^{N} (s_n + \eta g_t(x_n) - y_n)^2 = \frac{1}{N} \sum_{n=1}^{N} ((y_n - s_n) - \eta g_t(x_n))^2$$
Deciding Blending Weight as Optimization

after finding $g_t = h$,

$$
\min_{\eta} \min_{\mathbf{n}} \frac{1}{N} \sum_{n=1}^{N} \text{err}\left( \sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(\mathbf{x}_n) + \eta g_t(\mathbf{x}_n), y_n \right)
$$

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

---

$$
\min_{\eta} \frac{1}{N} \sum_{n=1}^{N} (s_n + \eta g_t(\mathbf{x}_n) - y_n)^2 = \frac{1}{N} \sum_{n=1}^{N} (y_n - s_n - \eta g_t(\mathbf{x}_n))^2
$$

—one-variable linear regression on $\{(g_t\text{-transformed input, residual})\}$
Deciding Blending Weight as Optimization

after finding $g_t = h$,

$$
\min_\eta \min_n \frac{1}{N} \sum_{n=1}^{N} \text{err}\left( \sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(x_n) + \eta g_t(x_n), y_n \right) \text{ with } \text{err}(s, y) = (s - y)^2
$$

---

---

GradientBoost for regression: $\alpha_t = \text{optimal } \eta$ by $g_t$-transformed linear regression
Putting Everything Together

Gradient Boosted Decision Tree (GBDT)

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

return $G(x) = \sum_{t=1}^{T} \alpha_t g_t(x)$
Putting Everything Together

Gradient Boosted Decision Tree (GBDT)

$s_1 = s_2 = \ldots = s_N = 0$

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

1. obtain $g_t$ by $\mathcal{A}(\{(x_n, y_n - s_n)\})$ where $\mathcal{A}$ is a (squared-error) regression algorithm

return $G(x) = \sum_{t=1}^{T} \alpha_t g_t(x)$
Gradient Boosted Decision Tree (GBDT)

\[ s_1 = s_2 = \ldots = s_N = 0 \]

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

1. obtain \( g_t \) by \( A(\{(x_n, y_n - s_n)\}) \) where \( A \) is a (squared-error) regression algorithm
   —how about sampled and pruned C&RT?

return \( G(x) = \sum_{t=1}^{T} \alpha_t g_t(x) \)
Gradient Boosted Decision Tree (GBDT)

\[ s_1 = s_2 = \ldots = s_N = 0 \]
for \( t = 1, 2, \ldots, T \)

1. obtain \( g_t \) by \( A(\{(x_n, y_n - s_n)\}) \) where \( A \) is a (squared-error) regression algorithm
   —how about sampled and pruned C&RT?

2. compute \( \alpha_t = \text{OneVarLinearRegression}(\{(g_t(x_n), y_n - s_n)\}) \)

return \( G(x) = \sum_{t=1}^{T} \alpha_t g_t(x) \)
Gradient Boosted Decision Tree (GBDT)

\[ s_1 = s_2 = \ldots = s_N = 0 \]

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

1. obtain \( g_t \) by \( A(\{(x_n, y_n - s_n)\}) \) where \( A \) is a (squared-error) regression algorithm
   —how about sampled and pruned C\&RT?

2. compute \( \alpha_t = \text{OneVarLinearRegression}(\{(g_t(x_n), y_n - s_n)\}) \)

3. update \( s_n \leftarrow s_n + \alpha_t g_t(x_n) \)

return \( G(x) = \sum_{t=1}^{T} \alpha_t g_t(x) \)
Putting Everything Together

Gradient Boosted Decision Tree (GBDT)

\[ s_1 = s_2 = \ldots = s_N = 0 \]

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

1. obtain \( g_t \) by \( A(\{(x_n, y_n - s_n)\}) \) where \( A \) is a (squared-error) regression algorithm

2. how about sampled and pruned C&RT?

3. compute \( \alpha_t = \text{OneVarLinearRegression}(\{(g_t(x_n), y_n - s_n)\}) \)

4. update \( s_n \leftarrow s_n + \alpha_t g_t(x_n) \)

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

GBDT: ‘regression sibling’ of AdaBoost-DTree — popular in practice
Which of the following is the optimal $\eta$ for

$$
\min_{\eta} \frac{1}{N} \sum_{n=1}^{N} \left( (y_n - s_n) - \eta g_t(x_n) \right)^2
$$

1. $(\sum_{n=1}^{N} g_t(x_n)(y_n - s_n)) \cdot \left( \sum_{n=1}^{N} g_t^2(x_n) \right)$
2. $(\sum_{n=1}^{N} g_t(x_n)(y_n - s_n)) \div \left( \sum_{n=1}^{N} g_t^2(x_n) \right)$
3. $(\sum_{n=1}^{N} g_t(x_n)(y_n - s_n)) + \left( \sum_{n=1}^{N} g_t^2(x_n) \right)$
4. $(\sum_{n=1}^{N} g_t(x_n)(y_n - s_n)) - \left( \sum_{n=1}^{N} g_t^2(x_n) \right)$

Reference Answer: 2
Derived within Lecture 9 of ML Foundations, remember? :-)
Hsuan-Tien Lin (NTU CSIE)
Which of the following is the optimal $\eta$ for

$$\min_{\eta} \frac{1}{N} \sum_{n=1}^{N} ((y_n - s_n) - \eta g_t(x_n))^2$$

1. $$(\sum_{n=1}^{N} g_t(x_n)(y_n - s_n)) \cdot (\sum_{n=1}^{N} g_t^2(x_n))$$
2. $$(\sum_{n=1}^{N} g_t(x_n)(y_n - s_n)) / (\sum_{n=1}^{N} g_t^2(x_n))$$
3. $$(\sum_{n=1}^{N} g_t(x_n)(y_n - s_n)) + (\sum_{n=1}^{N} g_t^2(x_n))$$
4. $$(\sum_{n=1}^{N} g_t(x_n)(y_n - s_n)) - (\sum_{n=1}^{N} g_t^2(x_n))$$

Reference Answer: 2

Derived within Lecture 9 of ML Foundations, remember? :-(
Map of Blending Models

blending: aggregate after getting diverse $g_t$
Map of Blending Models

blending: aggregate after getting diverse $g_t$

uniform

simple
voting/averaging of $g_t$
Map of Blending Models

blending: aggregate after getting diverse $g_t$

- uniform
  - simple voting/averaging of $g_t$
- non-uniform
  - linear model on $g_t$-transformed inputs
Map of Blending Models

blending: aggregate after getting diverse $g_t$

- **uniform**
  - simple
  - voting/averaging of $g_t$

- **non-uniform**
  - linear model on $g_t$-transformed inputs

- **conditional**
  - nonlinear model on $g_t$-transformed inputs
blending: aggregate after getting diverse $g_t$

**uniform**
- simple
- voting/averaging of $g_t$

**non-uniform**
- linear model on $g_t$-transformed inputs

**conditional**
- nonlinear model on $g_t$-transformed inputs

uniform for ‘stability’;
Map of Blending Models

blending: aggregate after getting **diverse** $g_t$

- **uniform**
  - simple
  - voting/averaging of $g_t$

- **non-uniform**
  - linear model on $g_t$-transformed inputs

- **conditional**
  - nonlinear model on $g_t$-transformed inputs

uniform for ‘stability’;
non-uniform/conditional **carefully** for ‘complexity’
Map of Aggregation-Learning Models

learning: aggregate as well as getting \textit{diverse} $g_t$
Map of Aggregation-Learning Models

learning: aggregate as well as getting diverse $g_t$

Bagging

diverse $g_t$ by bootstrapping;
Map of Aggregation-Learning Models

learning: aggregate as well as getting diverse $g_t$

Bagging

- diverse $g_t$ by bootstrapping;
- uniform vote by nothing :-)

Gradient Boosted Decision Tree

Summary of Aggregation Models

Hsuan-Tien Lin (NTU CSIE)
Map of Aggregation-Learning Models

learning: aggregate as well as getting diverse \( g_t \)

**Bagging**
- diverse \( g_t \) by bootstrapping;
- uniform vote by nothing :-)

**AdaBoost**
- diverse \( g_t \) by reweighting;
Map of Aggregation-Learning Models

learning: aggregate **as well as getting diverse** $g_t$

**Bagging**

diverse $g_t$ by bootstrapping;
uniform vote by nothing :-)

**AdaBoost**

diverse $g_t$
by reweighting;
linear vote by steepest search
Map of Aggregation-Learning Models

learning: aggregate as well as getting diverse $g_t$

Bagging
- diverse $g_t$ by bootstrapping; uniform vote by nothing :-)

AdaBoost
- diverse $g_t$ by reweighting; linear vote by steepest search

Decision Tree
- conditional vote by branching
Map of Aggregation-Learning Models

learning: aggregate as well as getting diverse $g_t$

Bagging
- diverse $g_t$ by bootstrapping;
- uniform vote by nothing :-)

AdaBoost
- diverse $g_t$
- by reweighting;
- linear vote by steepest search

Decision Tree
- diverse $g_t$
- by data splitting;
- conditional vote by branching
Map of Aggregation-Learning Models

learning: aggregate **as well as getting** diverse $g_t$

<table>
<thead>
<tr>
<th>Bagging</th>
<th>AdaBoost</th>
<th>Decision Tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>diverse $g_t$ by bootstrapping; uniform vote by nothing :-</td>
<td>diverse $g_t$ by reweighting; linear vote by steepest search</td>
<td>diverse $g_t$ by data splitting; conditional vote by branching</td>
</tr>
</tbody>
</table>

GradientBoost

diverse $g_t$
by residual fitting;
Map of Aggregation-Learning Models

learning: aggregate as well as getting diverse $g_t$

- **Bagging**
  - diverse $g_t$ by bootstrapping;
  - uniform vote by nothing :-)

- **AdaBoost**
  - diverse $g_t$ by reweighting;
  - linear vote by steepest search

- **Decision Tree**
  - diverse $g_t$ by data splitting;
  - conditional vote by branching

- **GradientBoost**
  - diverse $g_t$ by residual fitting;
  - linear vote by steepest search
Map of Aggregation-Learning Models

learning: aggregate as well as getting diverse $g_t$

Bagging
- diverse $g_t$ by bootstrapping;
  uniform vote by nothing :-)

AdaBoost
- diverse $g_t$ by reweighting;
  linear vote by steepest search

Decision Tree
- diverse $g_t$ by data splitting;
  conditional vote by branching

GradientBoost
- diverse $g_t$ by residual fitting;
  linear vote by steepest search

boosting-like algorithms most popular
Map of Aggregation of Aggregation Models

- Bagging
- AdaBoost
- Decision Tree

- GradientBoost

Gradient Boosted Decision Tree
Map of Aggregation of Aggregation Models

- Bagging
- AdaBoost
- Decision Tree
- Random Forest
  - randomized bagging
  - + ‘strong’ DTree
- GradientBoost

Summary of Aggregation Models

Gradient Boosted Decision Tree
Map of Aggregation of Aggregation Models

- **Bagging**
- **AdaBoost**
- **Decision Tree**
  - Random Forest
    - randomized bagging
    - + ‘strong’ DTree
  - AdaBoost-DTree
    - AdaBoost
    - + ‘weak’ DTree
  - GradientBoost
Map of Aggregation of Aggregation Models

- **Bagging**
  - Random Forest
    - randomized bagging
    - + ‘strong’ DTree

- **AdaBoost**
  - AdaBoost-DTree
    - AdaBoost
      - + ‘weak’ DTree

- **Decision Tree**

- **GradientBoost**

- **GBDT**
  - GradientBoost
    - + ‘weak’ DTree
Gradient Boosted Decision Tree

Summary of Aggregation Models

Map of Aggregation of Aggregation Models

- **Bagging**
  - Random Forest
    - randomized bagging + ‘strong’ DTree

- **AdaBoost**
  - AdaBoost-DTree
    - AdaBoost + ‘weak’ DTree

- **Decision Tree**
  - GradientBoost
  - GBDT
    - GradientBoost + ‘weak’ DTree

**all three** frequently used in practice
Gradient Boosted Decision Tree

Summary of Aggregation Models

Specialty of Aggregation Models

- **cure underfitting**
  - $G(x) \rightarrow \text{'strong'}$
  - aggregation $= \Rightarrow$ feature transform

- **cure overfitting**
  - $G(x) \rightarrow \text{'moderate'}$
  - aggregation $= \Rightarrow$ regularization

Proper aggregation (a.k.a. 'ensemble') $\Rightarrow$ better performance
Specialty of Aggregation Models

- **cure underfitting**
  - $G(x)$ ‘strong’
Specialty of Aggregation Models

- **cure underfitting**
  - $G(x)$ ‘strong’
  - aggregation
  $\implies$ **feature transform**

- **cure overfitting**
  - $G(x)$ ‘moderate’
  - aggregation $\implies$ regularization
  $\implies$ **better performance**
cure underfitting

- \( G(x) \) ‘strong’
- aggregation

\[ \rightarrow \text{feature transform} \]
Specialty of Aggregation Models

- **cure underfitting**
  - $G(x)$ ‘strong’
  - aggregation $\Rightarrow$ feature transform

- **cure overfitting**
  - $G(x)$ ‘moderate’
Specialty of Aggregation Models

**cure underfitting**
- $G(x)$ ‘strong’
- aggregation
  $\Rightarrow$ **feature transform**

**cure overfitting**
- $G(x)$ ‘moderate’
- aggregation
  $\Rightarrow$ **regularization**
Gradient Boosted Decision Tree

Summary of Aggregation Models

Specialty of Aggregation Models

cure underfitting

• $G(x)$ ‘strong’
• aggregation

⇒ feature transform


cure overfitting

• $G(x)$ ‘moderate’
• aggregation

⇒ regularization

proper aggregation (a.k.a. ‘ensemble’)

⇒ better performance
Which of the following aggregation model learns diverse $g_t$ by reweighting and calculates linear vote by steepest search?

1. AdaBoost
2. Random Forest
3. Decision Tree
4. Linear Blending
Which of the following aggregation model learns diverse $g_t$ by reweighting and calculates linear vote by steepest search?

1. AdaBoost
2. Random Forest
3. Decision Tree
4. Linear Blending

Reference Answer: 1

Congratulations on being an expert in aggregation models! :-}
Summary

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

Lecture 11: Gradient Boosted Decision Tree
- Adaptive Boosted Decision Tree
  sampling and pruning for ‘weak’ trees
- Optimization View of AdaBoost
  functional gradient descent on exponential error
- Gradient Boosting
  iterative steepest residual fitting
- Summary of Aggregation Models
  some cure underfitting; some cure overfitting

3. Distilling Implicit Features: Extraction Models
- next: extract features other than hypotheses