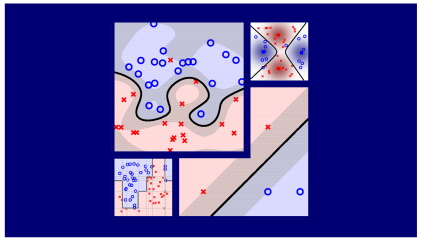
# Machine Learning Techniques
# (機器學習技法)



## 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

**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

3. Distilling Implicit Features: Extraction Models

# 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 $\mathbf{u}^{(t)}$
2. obtain tree $g_t$ by DTree($\mathcal{D}, \mathbf{u}^{(t)}$)
3. calculate 'vote' $\alpha_t$ of $g_t$

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

need: weighted DTree($\mathcal{D}, \mathbf{u}^{(t)}$)

# Weighted Decision Tree Algorithm

## Weighted Algorithm

minimize (regularized) $E_{\text{in}}^{\mathbf{u}}(h) = \frac{1}{N}\sum_{n=1}^{N} u_n \cdot \text{err}(y_n, h(\mathbf{x}_n))$

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

### 'Weighted' Algorithm in Bagging

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

### A General Randomized Base Algorithm

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

AdaBoost-DTree: often via
AdaBoost + **sampling** $\propto \mathbf{u}^{(t)}$ + DTree($\tilde{\mathcal{D}}_t$)
without modifying DTree

# Weak Decision Tree Algorithm

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

if fully grown tree trained on all $\mathbf{x}_n$
$\Longrightarrow E_{\text{in}}(g_t) = 0$ if all $\mathbf{x}_n$ different
$\Longrightarrow E^{\mathbf{u}}_{\text{in}}(g_t) = 0$
$\Longrightarrow \epsilon_t = 0$
$\Longrightarrow \alpha_t = \infty$ (**autocracy**!!)

need: **pruned** tree trained on **some** $\mathbf{x}_n$ to be **weak**

- **pruned**: usual pruning, or just **limiting tree height**
- **some**: **sampling** $\propto \mathbf{u}^{(t)}$

AdaBoost-DTree: often via AdaBoost + **sampling** $\propto \mathbf{u}^{(t)}$ + **pruned** DTree($\tilde{\mathcal{D}}$)

# AdaBoost with Extremely-Pruned Tree

what if DTree with **height** $\leq 1$ (extremely pruned)?

## DTree (C&RT) with **height** $\leq 1$

learn branching criteria

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

—if impurity = binary classification error,

**just a decision stump, remember? :-)**

AdaBoost-Stump
= **special case** of AdaBoost-DTree

# Fun Time

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

# Fun Time

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

# Example Weights of AdaBoost

$$u_n^{(t+1)} = \begin{cases} u_n^{(t)} \cdot \blacklozenge_t & \textbf{if incorrect} \\ u_n^{(t)} / \blacklozenge_t & \textbf{if correct} \end{cases}$$

$$= u_n^{(t)} \cdot \blacklozenge_t^{-y_n g_t(\mathbf{x}_n)} = u_n^{(t)} \cdot \exp\left(-y_n \alpha_t g_t(\mathbf{x}_n)\right)$$

$$u_n^{(T+1)} = u_n^{(1)} \cdot \prod_{t=1}^{T} \exp\left(-y_n \alpha_t g_t(\mathbf{x}_n)\right) = \frac{1}{N} \cdot \exp\left(-y_n \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x}_n)\right)$$

- recall: $G(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})\right)$
- $\sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})$ : **voting score** of $\{g_t\}$ on $\mathbf{x}$

AdaBoost: $u_n^{(T+1)} \propto \exp\left(-y_n(\text{ voting score on } \mathbf{x}_n)\right)$

# Voting Score and Margin

linear blending = LinModel + hypotheses as transform + ~~constraints~~

$$G(\mathbf{x}_n) = \text{sign}\left(\overbrace{\sum_{t=1}^{T} \underbrace{\alpha_t}_{w_i} \underbrace{g_t(\mathbf{x}_n)}_{\phi_i(\mathbf{x}_n)}}^{\text{voting score}}\right)$$

and hard-margin SVM **margin** = $\frac{y_n \cdot (\mathbf{w}^T \phi(\mathbf{x}_n) + b)}{\|\mathbf{w}\|}$, **remember? :-)**

$y_n$(voting score) = signed & unnormalized **margin**

want $y_n$(voting score) **positive & large**

$\Leftrightarrow$ exp($-y_n$(voting score)) **small**

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

claim: AdaBoost **decreases** $\sum_{n=1}^{N} u_n^{(t)}$

# AdaBoost Error Function

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

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

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

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

$\widehat{\text{err}}_{\text{ADA}}$: **algorithmic error measure** by **convex upper bound** of $\text{err}_{0/1}$

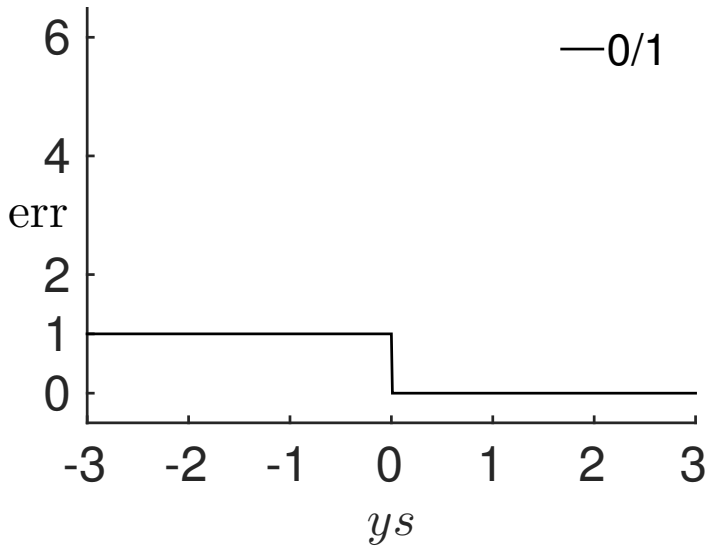
# AdaBoost Error Function

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

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

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

- $\text{err}_{0/1}(s, y) = [\![ys \le 0]\!]$
- $\widehat{\text{err}}_{\text{ADA}}(s, y) = \exp(-ys)$: upper bound of $\text{err}_{0/1}$ —called **exponential error measure**




$\widehat{\text{err}}_{\text{ADA}}$: **algorithmic error measure** by **convex upper bound** of $\text{err}_{0/1}$

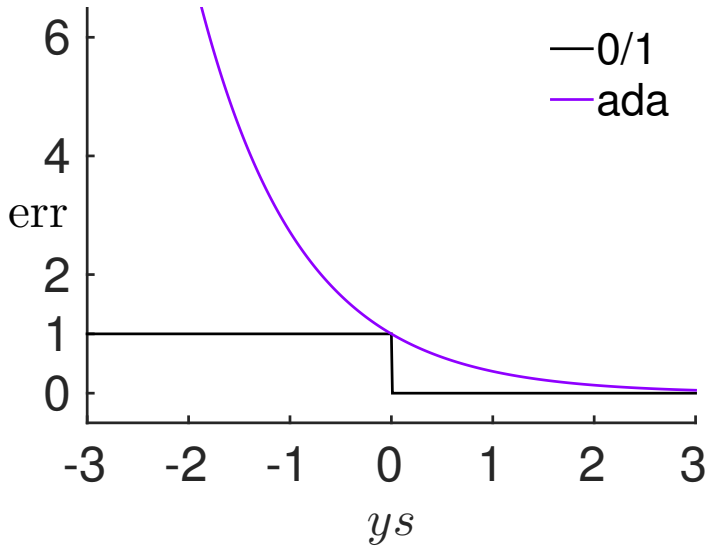
# AdaBoost Error Function

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

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

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

- $\text{err}_{0/1}(s, y) = [\![ys \le 0]\!]$
- $\widehat{\text{err}}_{\text{ADA}}(s, y) = \exp(-ys)$: upper bound of $\text{err}_{0/1}$ —called **exponential error measure**




$\widehat{\text{err}}_{\text{ADA}}$: **algorithmic error measure** by **convex upper bound** of $\text{err}_{0/1}$

# Gradient Descent on AdaBoost Error Function

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

$$\min_{\|\mathbf{v}\|=1} \quad E_{\text{in}}(\mathbf{w}_t + \eta\mathbf{v}) \approx \underbrace{E_{\text{in}}(\mathbf{w}_t)}_{\text{known}} + \underbrace{\eta}_{\text{given positive}} \underbrace{\mathbf{v}^T \nabla E_{\text{in}}(\mathbf{w}_t)}_{\text{known}}$$

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

$$\min_{h} \quad \widehat{E}_{\text{ADA}} = \frac{1}{N}\sum_{n=1}^{N} \exp\left(-y_n\left(\sum_{\tau=1}^{t-1}\alpha_\tau g_\tau(\mathbf{x}_n) + \eta h(\mathbf{x}_n)\right)\right)$$

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

$$\overset{\text{taylor}}{\approx} \sum_{n=1}^{N} u_n^{(t)} \left(1 - y_n \eta h(\mathbf{x}_n)\right) = \sum_{n=1}^{N} u_n^{(t)} - \eta \sum_{n=1}^{N} u_n^{(t)} y_n h(\mathbf{x}_n)$$

good $h$: minimize $\sum_{n=1}^{N} u_n^{(t)} \left(-y_n h(\mathbf{x}_n)\right)$

# Learning Hypothesis as Optimization

finding good $h$ (function direction) $\Leftrightarrow$ minimize $\sum_{n=1}^{N} u_n^{(t)} (-y_n h(\mathbf{x}_n))$

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

$$\begin{aligned}
\sum_{n=1}^{N} u_n^{(t)} (-y_n h(\mathbf{x}_n)) &= \sum_{n=1}^{N} u_n^{(t)} \begin{cases} -1 & \text{if } y_n = h(\mathbf{x}_n) \\ +1 & \text{if } y_n \neq h(\mathbf{x}_n) \end{cases} \\
&= -\sum_{n=1}^{N} u_n^{(t)} + \sum_{n=1}^{N} u_n^{(t)} \begin{cases} 0 & \text{if } y_n = h(\mathbf{x}_n) \\ 2 & \text{if } y_n \neq h(\mathbf{x}_n) \end{cases} \\
&= -\sum_{n=1}^{N} u_n^{(t)} + 2 E_{\text{in}}^{\mathbf{u}^{(t)}}(h) \cdot N
\end{aligned}$$

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

$\mathcal{A}$: **good** $g_t = h$ for 'gradient descent'

# Deciding Blending Weight as Optimization

AdaBoost finds $g_t$ by approximately $\min_{h} \widehat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp\left(-y_n \eta h(\mathbf{x}_n)\right)$

after finding $g_t$, how about $\min_{\eta} \widehat{E}_{\text{ADA}} = \sum_{n=1}^{N} u_n^{(t)} \exp\left(-y_n \eta g_t(\mathbf{x}_n)\right)$

- optimal $\eta_t$ somewhat **'greedily faster'** than fixed (small) $\eta$
  —called **steepest** descent for optimization
- two cases inside summation:
  - $y_n = g_t(\mathbf{x}_n)$ : $u_n^{(t)} \exp(-\eta)$ (correct)
  - $y_n \neq g_t(\mathbf{x}_n)$ : $u_n^{(t)} \exp(+\eta)$ (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? :-)**
—AdaBoost: steepest descent with approximate functional gradient

# Fun Time

With $\widehat{E}_{\text{ADA}} = \left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \Big( (1-\epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \Big)$, which of the following is $\frac{\partial \widehat{E}_{\text{ADA}}}{\partial \eta}$ that can be used for solving the optimal $\eta_t$?

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

# Fun Time

With $\widehat{E}_{\text{ADA}} = \left(\sum_{n=1}^{N} u_n^{(t)}\right) \cdot \Big( (1-\epsilon_t) \exp(-\eta) + \epsilon_t \exp(+\eta) \Big)$, which of the following is $\frac{\partial \widehat{E}_{\text{ADA}}}{\partial \eta}$ that can be used for solving the optimal $\eta_t$?

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

**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

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

with binary-output hypothesis $h$

## GradientBoost

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

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

GradientBoost: allows **extension to different** err for regression/soft classification/etc.

# GradientBoost for Regression

$$\min_{\eta} \min_{h} \frac{1}{N}\sum_{n=1}^{N} \text{err}\Big(\underbrace{\sum_{\tau=1}^{t-1}\alpha_\tau g_\tau(\mathbf{x}_n)}_{s_n} + \eta h(\mathbf{x}_n), y_n\Big) \quad \text{with } \text{err}(s,y) = (s-y)^2$$

$$\min_{h} \ldots \overset{\text{taylor}}{\approx} \min_{h} \frac{1}{N}\sum_{n=1}^{N} \underbrace{\text{err}(s_n, y_n)}_{\text{constant}} + \frac{1}{N}\sum_{n=1}^{N} \eta h(\mathbf{x}_n) \left.\frac{\partial \text{err}(s, y_n)}{\partial s}\right|_{s=s_n}$$

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

naïve solution $h(\mathbf{x}_n) = -\infty \cdot (s_n - y_n)$
if no constraint on $h$

# Learning Hypothesis as Optimization

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

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

$$\min_{h} \quad \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} \left(2h(\mathbf{x}_n)(s_n - y_n) + (h(\mathbf{x}_n))^2\right)$$

$$= \quad \text{constants} + \frac{\eta}{N} \sum_{n=1}^{N} \left(\text{constant} + \left(h(\mathbf{x}_n) - (y_n - s_n)\right)^2\right)$$

- solution of penalized approximate functional gradient: squared-error regression on $\{(\mathbf{x}_n, \underbrace{y_n - s_n}_{\text{residual}})\}$

GradientBoost for regression:
find $g_t = h$ by regression with **residuals**

# Deciding Blending Weight as Optimization

after finding $g_t = h$,

$$\min_{\eta} \cancel{\min_{h}} \frac{1}{N} \sum_{n=1}^{N} \text{err}\Big( \underbrace{\sum_{\tau=1}^{t-1} \alpha_\tau g_\tau(\mathbf{x}_n)}_{s_n} + \eta g_t(\mathbf{x}_n), y_n \Big) \text{ with } \text{err}(s, y) = (s - y)^2$$

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

—one-variable linear regression on $\{(g_t\text{-transformed input}, \textbf{residual})\}$

GradientBoost for regression: $\alpha_t =$ optimal $\eta$
by $g_t$-transformed linear regression

# 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 $\mathcal{A}(\{(\mathbf{x}_n, y_n - s_n)\})$ where $\mathcal{A}$ is a (squared-error) regression algorithm
   —**how about sampled and pruned C&RT?**
2. compute $\alpha_t = \text{OneVarLinearRegression}(\{(g_t(\mathbf{x}_n), y_n - s_n)\})$
3. update $s_n \leftarrow s_n + \alpha_t g_t(\mathbf{x}_n)$

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

GBDT: 'regression sibling' of AdaBoost-DTree
—**popular in practice**

# Fun Time

Which of the following is the optimal $\eta$ for

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

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

# Fun Time

Which of the following is the optimal $\eta$ for

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

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

## Reference Answer: 2

Derived within Lecture 9 of ML Foundations, **remember? :-)**

# Map of Blending Models

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

| uniform | non-uniform | conditional |
| --- | --- | --- |
| simple voting/averaging of $g_t$ | linear model on $g_t$-transformed inputs | 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 **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**

**Random Forest**

randomized bagging
+ 'strong' DTree

**AdaBoost-DTree**

AdaBoost
+ 'weak' DTree

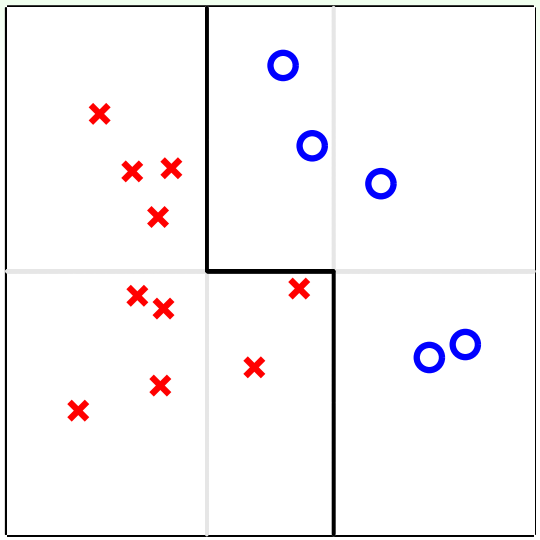
**GradientBoost**

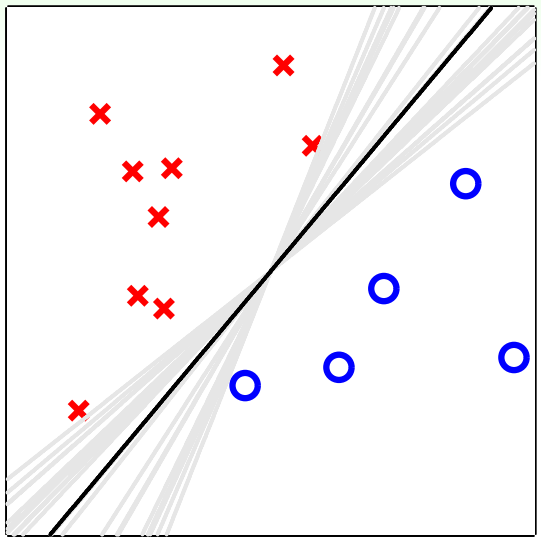
**GBDT**

GradientBoost
+ 'weak' DTree

**all three** frequently used in practice

# Specialty of Aggregation Models





## cure underfitting

- $G(\mathbf{x})$ 'strong'
- aggregation

$\Longrightarrow$ **feature transform**

## cure overfitting

- $G(\mathbf{x})$ 'moderate'
- aggregation

$\Longrightarrow$ **regularization**

proper aggregation (a.k.a. 'ensemble')

$\Longrightarrow$ **better performance**

# Fun Time

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

# Fun Time

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**