CSE 417T
Introduction to Machine Learning

Lecture 14
Instructor: Chien-Ju (CJ) Ho
Logistics

• Homework 1 Returned
  • Regrade requests till this Saturday
  • Please be concise and polite

• Homework 3: Due **Mar 5 (Sat)**
  • Keep track of your own late-day usages

• Exam 1: **Mar 10 (Thursday)**
  • Topics: LFD Chapters 1 to 5
  • Covid-permitting
    • Timed exam (75 min) during lecture time in the classroom
    • Closed-book exam with 2 letter-size cheat sheets allowed (4 pages in total)
      • No format limitations (it can be typed, written, or a combination)
  • Mar 8 (Tuesday) will be a review lecture
Recap
Decision Tree Hypothesis

Pros
- Easy to interpret (interpretability is getting attention and is important in some domains)
- Can handle multi-type data (Numerical, categorical, ...)
- Easy to implement (Bunch of if-else rules)

Cons
- Generally speaking, bad generalization
- VC dimension is infinity
- High variance (small change of data leads to very different hypothesis)
- Easily overfit

Why we care?
- One of the classical model
- Building block for other models (e.g., random forest)

Credit Card Approval Example

Annual Income

\[ \begin{align*}
\text{≥ 100k} & \quad \text{Approve} \\
\text{≥ 20k} & \quad \text{have debt?} \\
\text{< 20k} & \quad \text{Deny} \\
\text{< 100k} & \quad \text{Approve} \\
\text{yes} & \quad \text{Deny} \\
\text{no} & \quad \text{Approve}
\end{align*} \]
ID3: Using Information Gain as Selection Criteria

• Information gain of choosing feature $A$ to split
  
  $Gain(D, A) = H(D) - \sum_i \frac{|D_i|}{|D|} H(D_i)$ [The amount of decrease in entropy]

• ID3: Choose the split that maximize $Gain(D, A)$

DecisionTreeLearn($D$)
  
  Create a root node $r$
  
  If termination conditions are met
    
    return a single node tree with leaf prediction based on
  
  Else: Greedily find a feature $A$ to split according to split criteria
  
  For each possible value $v_i$ of $A$
    
    Let $D_i$ be the dataset containing data with value $v_i$ for feature $A$
    
    Create a subtree DecisionTreeLearn($D_i$) that being the child of root $r$

• ID3 termination conditions
  
  • If all labels are the same
  
  • If all features are the same
  
  • If dataset is empty

• ID3 leaf predictions
  
  • Most common labels (majority voting)

• ID3 split criteria
  
  • Information gain

Notations:

$H(D)$: Entropy of $D$

$|D|$ is the number of points in $D$
Ensemble Learning

• Goal: Utilize a set of weak learners to obtain a strong learner.

• Format of ensemble learning
  • Construct many diverse weak learners
  • Aggregate the weak learners

Bagging
  • Construct diverse weak learners
    • (Simultaneously) bootstrap datasets
    • Train weak learners on them
  • Aggregate the weak learners
    • Uniform aggregation
Bagging - Bootstrapped Aggregating

- Bootstrap $M$ datasets $\{\tilde{D}^{\{m\}}\}$ (Sample with replacement from $D$)
- Learn a hypothesis from each of them

\[ H \quad \tilde{D}^{\{1\}} \rightarrow g_1 \]
\[ \quad \tilde{D}^{\{2\}} \rightarrow g_2 \]
\[ \quad \ldots \]
\[ \quad \tilde{D}^{\{M\}} \rightarrow g_M \]

- Aggregate the learned hypothesis (assume we are doing classification)

\[ G(\hat{x}) = \bar{g}(\hat{x}) = \text{sign} \left( \frac{1}{M} \sum_{m=1}^{M} g_m(\hat{x}) \right) \]
Out-Of-Bag (OOB) Error

<table>
<thead>
<tr>
<th>$\tilde{x}_n, y_n$</th>
<th>$\tilde{D}^{(1)}$</th>
<th>$\tilde{D}^{(2)}$</th>
<th>$\tilde{D}^{(3)}$</th>
<th>$\tilde{D}^{(4)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\tilde{x}_1, y_1)$</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>$(\tilde{x}_2, y_2)$</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$(\tilde{x}_N, y_N)$</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- $G_n^-$: the aggregation of hypothesis that $\tilde{x}_n$ is OOB of
  - $G_1^- = \text{aggregate}(g_3, g_4, ...)$
  - $G_2^- = \text{aggregate}(g_2, g_3, g_4, ...)$
  - $G_N^- = \text{aggregate}(g_1, ...)$

- OOB Error
  - $E_{OOB}(G) = \frac{1}{N} \sum_{n=1}^{N} \text{error}(G_n^-(\tilde{x}_n), y_n)$

Aggregate:
- Majority voting for classification
- Average for regression

Error:
- Binary error for classification
- Squared error for regression

Whether a point is in a bootstrapped dataset
Out-Of-Bag (OOB) Error

\[ E_{OOB}(G) = \frac{1}{N} \sum_{n=1}^{N} \text{error}(G_{n}^{-}(\hat{x}_n), y_n) \]

• Bagging provided an intrinsic mechanism for us to perform validation
  • We don’t need to split the dataset into training and validation

• Practical issues (you might face this in HW4)
  • What if some \( \hat{x}_n \) appears in all bootstrapped datasets?
    • The probability of this happening is small when the number of bags \( M \) is large
  • Let \( S \) be the set of points that is out of bag for at least one bootstrapped dataset
    • \( E_{OOB}(G) = \frac{1}{|S|} \sum_{(\hat{x}_n,y_n) \in S} \text{error}(G_{n}^{-}(\hat{x}_n), y_n) \)
Random Forest

• Construct many random trees
  • Bootstrap datasets and learn a max-depth tree for each of them
  • Other randomizations (not required in HW4)
    • When choosing split features, choose from a random subset (instead of all features)
    • Randomly project features (similar to non-linear transformation) for each tree

• Aggregate the random trees
  • Classification: Majority vote \( \bar{g}(\tilde{x}) = \text{sign} \left( \frac{1}{M} \sum_{m=1}^{M} g_m(\tilde{x}) \right) \)
  • Regression: Average \( \bar{g}(\tilde{x}) = \frac{1}{M} \sum_{m=1}^{M} g_m(\tilde{x}) \)
Today’s Lecture

The notes are not intended to be comprehensive. They should be accompanied by lectures and/or textbook. Let me know if you spot errors.
Boosting
Ensemble Learning

• Goal: Utilize a set of **weak learners** to obtain a **strong learner**.

• Format of ensemble learning
  • **Construct** many **diverse** weak learners
  • **Aggregate** the weak learners

<table>
<thead>
<tr>
<th>Bagging:</th>
<th>Boosting</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Construct diverse weak learners</td>
<td>• Construct diverse weak learners</td>
</tr>
<tr>
<td>• <em>(Simultaneously)</em> bootstrapping datasets</td>
<td>• <strong>Adaptively</strong> generating datasets</td>
</tr>
<tr>
<td>• Train weak learners on them</td>
<td>• Train weak learners on them</td>
</tr>
<tr>
<td>• Aggregate the weak learners</td>
<td>• Aggregate the weak learners</td>
</tr>
<tr>
<td>• <strong>Uniform</strong> aggregation</td>
<td>• <strong>Weighted</strong> aggregation</td>
</tr>
</tbody>
</table>
Informal Intuitions about Boosting

• Example: Teach a class of kids to identify apples from data
  - Alice: Apples are circular
  - Teacher: Circular is a good feature, but using this feature might make some mistakes
    Let me highlight the mistakes.
    • Make correct images smaller
    • Make incorrect images larger

Example created by Hsuan-Tien Lin
Informal Intuitions about Boosting

• Example: Teach a class of kids to identify apples from data
  
  Alice: Apples are circular
  Bob: Apples are red
Informal Intuitions about Boosting

- Example: Teach a class of kids to identify apples from data

  - Alice: Apples are circular
  - Bob: Apples are red
  - Charlie: Apples could be green
Informal Intuitions about Boosting

• Example: Teach a class of kids to identify apples from data

  Alice: Apples are *circular*

  Bob: Apples are *red*

  Charlie: Apples could be *green*

  David: Apples have *stems* at the top

  Class: Apples are *somewhat circular, somewhat red, possibly green*, and may have stems at the top
Informal Intuitions about Boosting

- Example: Teach a class of kids to identify apples from data

- Alice: Apples are circular
- Bob: Apples are red
- Charlie: Apples could be green
- David: Apples have stems at the top

Class: Apples are somewhat circular, somewhat red, possibly green, and may have stems at the top.

Key steps of this process:
- Learn a simple hypothesis for each dataset
- Iteratively update the dataset to focus on what we got wrong (i.e., create diversity)
- Aggregate the learned simple hypothesis
Outline of a Boosting Algorithm

• Initialize $D_1$ (usually the same as the initial dataset $D$)

• For $t = 1$ to $T$
  • Learn $g_t$ from $D_t$
  • Reweight the distribution and obtain $D_{t+1}$ based on $g_t$ and $D_t$

• Output weighted-aggregate$(g_1, \ldots, g_T)$
  • Classification: $G(\vec{x}) = \bar{g}(\vec{x}) = \text{sign} \left( \frac{1}{T} \sum_{t=1}^{M} \alpha_t g_t(\vec{x}) \right)$

Questions

- How to learn $g_t$ from $D_t$
- How to reweight the distribution and obtain $D_{t+1}$
- How to perform weighted aggregation
Discussion on Re-weighted $D_t$ (What does re-weighting mean?)

- Original Dataset $D = \{(\tilde{x}_1, y_1), \ldots, (\tilde{x}_N, y_N)\}$
- Notation of $D_t$
  - $D_t(n)$ is the weight/probability of data point $(\tilde{x}_n, y_n)$ in $D_t$
  - $\sum_{n=1}^N D_t(n) = 1$

- What is $E_{in}(h)$ on $D_t$? (Expressed as $E_{in}^{(D_t)}(h)$)
  - Re-sample dataset (noisier)
    - Re-sample the dataset from $D$ according to distribution $D_t$
    - Calculate $E_{in}$ on the re-sampled dataset as usual

- Calculate weighted error
  - $E_{in}^{(D_t)}(h) = \sum_{n=1}^N D_t(n) \text{error}(h(\tilde{x}_n), y_n)$

When $D_t(n) = 1/N$. This reduces to standard definition of $E_{in}$. 
AdaBoost – Adaptive Boosting

How to learn $g_t$ from $D_t$
How to reweight the distribution and obtain $D_{t+1}$
How to perform weighted aggregation

[AdaBoost focuses on classification problem]
Boosting Background

• A theoretical question asked by Kearns and Valiant
  • Whether a “weak” learning algorithm which performs just slightly better than random guessing in the PAC model can be “boosted” into an arbitrarily accurate “strong” learning algorithm

• AdaBoost
  • The first adaptive boosting algorithm that
    • has nice theoretical guarantees
    • successfully deployed in real-world applications
What Does AdaBoost Do?

Outline of a Boosting Algorithm

Initialize $D_1$ (usually the same as the initial dataset $D$)
For $t = 1$ to $T$
  Learn $g_t$ from $D_t$
  Reweight the distribution and obtain $D_{t+1}$ based on $g_t$ and $D_t$
Output weighted-aggregate($g_1, \ldots, g_T$)
Classification: $G(\vec{x}) = \bar{g}(\vec{x}) = \text{sign} \left( \frac{1}{T} \sum_{t=1}^{M} \alpha_t g_t(\vec{x}) \right)$

• Will discuss the following for AdaBoost
  1. How to learn $g_t$ from $D_t$
  2. How to reweight the distribution and obtain $D_{t+1}$
  3. How to perform weighted aggregation
1. Learn a Weak Learner $g_t$ from $D_t$

• AdaBoost uses \textit{simple} weak learners
  • low variance, high bias
  • Decision stump (one-level decision tree) is one good option

- How to learn $g_t$ from $D_t$
  • Find the decision stump that
    • Minimizes $E_{in}^{(D_t)}$
    • Maximize (weighted) information gain (you can call decision tree library directly)
2. How to Reweight $D_{t+1}$

- We want to make $g_{t+1}$ (learned from $D_{t+1}$) to be diverse from $g_t$
  - Increase the weights of points that $g_t$ makes wrong predictions
  - Decrease the weights of points that $g_t$ makes correct predictions

- Define a parameter $\gamma > 1$
  - If $g_t$ makes wrong predictions on $\vec{x}_n$
    - $D_{t+1}(n) = \frac{1}{Z_t} D_t(n) \cdot \gamma$ (increase the weight)
  - If $g_t$ makes correct predictions on $\vec{x}_n$
    - $D_{t+1}(n) = \frac{1}{Z_t} D_t(n) / \gamma$ (decrease the weight)

- Goal:
  - Choose $\gamma$ such that $E_{in}^{(D_{t+1})}(g_t) = 0.5$
  - Since $g_{t+1}$ minimizes $E_{in}^{(D_{t+1})} => g_t$ and $g_{t+1}$ are diverse
Choose $\gamma$ such that $E_{in}^{(D_{t+1})}(g_t) = 0.5$

Math derivations in the next few slides
Define \( \epsilon_t = E_{in}^{(D_t)}(g_t) = \sum_{n=1}^{N} D_t(n) [g_t(\tilde{x}_n) \neq y_n] \)

- Weighted in-sample error of \( g_t \) on \( D_t \)
- \( \epsilon_t < 0.5 \) (requirement of weak learners)

Goal: Want to make \( E_{in}^{(D_{t+1})}(g_t) = 0.5 \)

We consider the case weak learners are better than random guessing: \( \epsilon_t < 0.5 \)

\[
E_{in}^{(D_t)}(h) = \sum_{n=1}^{N} D_t(n) \text{error}(h(\tilde{x}_n), y_n)
\]

- If \( g_t \) makes wrong predictions on \( \tilde{x}_n \)
  - \( D_{t+1}(n) = \frac{1}{z_t} D_t(n) \cdot \gamma \) (increase the weight)
- If \( g_t \) makes correct predictions on \( \tilde{x}_n \)
  - \( D_{t+1}(n) = \frac{1}{z_t} D_t(n) / \gamma \) (decrease the weight)
• Define \( \epsilon_t = E_{in}^{(D_t)}(g_t) = \sum_{n=1}^{N} D_t(n)[g_t(\tilde{x}_n) \neq y_n] \)

• Weighted in-sample error of \( g_t \) on \( D_t \)

• \( \epsilon_t < 0.5 \) (requirement of weak learners)

• Goal: Want to make \( E_{in}^{(D_{t+1})}(g_t) = 0.5 \)

\[
E_{in}^{(D_{t+1})}(g_t) = \sum_{n=1}^{N} D_{t+1}(n)[g_t(\tilde{x}_n) \neq y_n] \\
= \sum_{n=1}^{N} \frac{1}{Z_t} D_t(n)\gamma [g_t(\tilde{x}_n) \neq y_n] \\
= \frac{\gamma}{Z_t} \sum_{n=1}^{N} D_t(n) [g_t(\tilde{x}_n) \neq y_n] = \frac{\gamma}{Z_t} \epsilon_t
\]

• Remember \( Z_t \) is the normalization constant

\[
Z_t = \sum_{n=1}^{N} D_t(n)\gamma [g_t(\tilde{x}_n) \neq y_n] + \sum_{n=1}^{N} D_t(n) \frac{1}{\gamma} [g_t(\tilde{x}_n) = y_n] \\
= \gamma \epsilon_t + \frac{1}{\gamma} (1 - \epsilon_t)
\]

We consider the case weak learners are better than random guessing: \( \epsilon_t < 0.5 \)

\[
E_{in}^{(D_t)}(h) = \sum_{n=1}^{N} D_t(n) \text{error}(h(\tilde{x}_n), y_n)
\]

• If \( g_t \) makes wrong predictions on \( \tilde{x}_n \)
  * \( D_{t+1}(n) = \frac{1}{Z_t} D_t(n) \cdot \gamma \) (increase the weight)

• If \( g_t \) makes correct predictions on \( \tilde{x}_n \)
  * \( D_{t+1}(n) = \frac{1}{Z_t} D_t(n) / \gamma \) (decrease the weight)
• Want to make $E_{in}^{(D_{t+1})}(g_t) = 0.5$
  • $E_{in}^{(D_{t+1})}(g_t) = \frac{\gamma}{Z_t} \epsilon_t$
  • $Z_t = \gamma \epsilon_t + \frac{1}{\gamma} (1 - \epsilon_t)$
• Want to make $E_{in}^{(D_{t+1})}(g_t) = 0.5$

• $E_{in}^{(D_{t+1})}(g_t) = \frac{\gamma}{Z_t} \varepsilon_t$

• $Z_t = \gamma \varepsilon_t + \frac{1}{\gamma} (1 - \varepsilon_t)$

$\frac{\gamma \varepsilon_t}{\gamma \varepsilon_t + (1 - \varepsilon_t) / \gamma} = 0.5 \Rightarrow \frac{1 - \varepsilon_t}{\gamma} = \gamma \varepsilon_t \Rightarrow \gamma = \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}}$

• The rule for reweighting

  • If $g_t(\tilde{x}_n) \neq y_n$, then $D_{t+1}(n) = \frac{1}{Z_t} D_t(n) \left( \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \right)$

  • If $g_t(\tilde{x}_n) = y_n$, then $D_{t+1}(n) = \frac{1}{Z_t} D_t(n) \left( \sqrt{\frac{1 - \varepsilon_t}{\varepsilon_t}} \right)^{-1}$

Both $g_t(\tilde{x}_n)$ and $y_n$ are either +1 or -1

If $g_t(\tilde{x}_n) \neq y_n$, $g_t(\tilde{x}_n)y_n = -1$

If $g_t(\tilde{x}_n) = y_n$, $g_t(\tilde{x}_n)y_n = 1$
Want to make $E_{in}^{(D_{t+1})}(g_t) = 0.5$

- $E_{in}^{(D_{t+1})}(g_t) = \frac{\gamma}{Z_t} \epsilon_t$
- $Z_t = \gamma \epsilon_t + \frac{1}{\gamma} (1 - \epsilon_t)$

\[
\frac{\gamma \epsilon_t}{\gamma \epsilon_t + (1 - \epsilon_t)/\gamma} = 0.5 \implies \frac{1 - \epsilon_t}{\gamma} = \gamma \epsilon_t \implies \gamma = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}
\]

The rule for reweighting

- If $g_t(\hat{x}_n) \neq y_n$, then
  
  $D_{t+1}(n) = \frac{1}{Z_t} D_t(n) \left( \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \right) = \frac{1}{Z_t} D_t(n) \left( \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \right)^{-g_t(\hat{x}_n)y_n}$

- If $g_t(\hat{x}_n) = y_n$, then
  
  $D_{t+1}(n) = \frac{1}{Z_t} D_t(n) \left( \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \right)^{-1} = \frac{1}{Z_t} D_t(n) \left( \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \right)^{-g_t(\hat{x}_n)y_n}$

Reweight rule: $D_{t+1}(n) = \frac{1}{Z_t} D_t(n) \left( \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}} \right)^{-g_t(\hat{x}_n)y_n}$
2. How to Reweight $D_{t+1}$: Summary

• Reweight rule:
  
  \[ D_{t+1}(n) = \frac{1}{z_t} D_t(n) \left( \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \right)^{-g_t(x_n)\gamma_n} \]

• A bit more manipulations (the reason will be clear later)
  
  • Define $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$

  • $e^{\alpha_t} = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

• Final reweight rule:
  
  \[ D_{t+1}(n) = \frac{1}{z_t} D_t(n) e^{-\alpha_t g_t(x_n)\gamma_n} \]
3. How to Aggregate Weak Learners

• Intuition:
  - We want to put more weights on better weak learners
  - \( \epsilon_t = E_{in}^{(D_t)}(g_t) \) is a proxy on how well \( g_t \) performs (smaller \( \epsilon_t \) => better \( g_t \))

  - Recall that \( \alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right) \)
    - Better \( g_t \), smaller \( \epsilon_t \), higher \( \alpha_t \)
    - When \( \epsilon_t = 0.5 \), \( \alpha_t = 0 \) (random guessing leads to 0 weights)
    - When \( \epsilon_t = 0 \), \( \alpha_t = \infty \) (if a feature perfectly classifies the data, use it as our final hypothesis)

• Aggregation rule
  - \( G(\hat{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t g_t(\hat{x})) \)
AdaBoost Algorithm

- Given $D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_N, y_N)\}$
- Initialize $D_1(n) = 1/N$ for all $n = 1, \ldots, N$
- For $t = 1, \ldots, T$
  - Learn $g_t$ from $D_t$ (using decision stumps)
  - Calculate $\epsilon_t = E_{\text{in}}^{(D_t)}(g_t)$
  - Set $\alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right)$
  - Update $D_{t+1}(n) = \frac{1}{Z_t} D_t(n) e^{-\alpha_t y_n g_t(\vec{x}_n)}$
- Output $G(\vec{x}) = \text{sign}(\sum_{t=1}^T \alpha_t g_t(\vec{x}))$
Theoretical Properties of AdaBoost

• See Freund & Schapire's Tutorial for more discussion

• The training error of AdaBoost converges fast
  • Let $\gamma_t = \frac{1}{2} - \epsilon_t$ (how good each weak learner is better than random guessing)
  • $E_{in} \leq e^{-2 \sum_{t=1}^{T} \gamma_t^2}$

• Generalization error
  • VC analysis gives us $E_{out} \leq E_{in} + \tilde{O} \left( \sqrt{\frac{T d_{vc}}{m}} \right)$
  • It seems as $T$ goes large, overfitting could happen
  • Empirically, AdaBoost is relatively robust to overfitting
  • There are some more delicate analysis using the idea of margins to explain why
AdaBoost in Action
AdaBoost in Action

• A toy example (by Yoav Freund Rob Schapire)
• Weak learner: decision stump (one-level decision tree)
Round 1

\[ h_1 \]

\[ \epsilon_1 = 0.30 \]
\[ \alpha_1 = 0.42 \]

\[ D_2 \]
Round 2

\[ \varepsilon_2 = 0.21 \]
\[ \alpha_2 = 0.65 \]

\[ D_3 \]
Round 3

\[ h_3 \]

\[ \varepsilon_3 = 0.14 \]
\[ \alpha_3 = 0.92 \]
\[ H_{\text{final}} = \text{sign} \left( \begin{array}{c} 0.42 \\ + 0.65 \\ + 0.92 \\ - \end{array} \right) \]
Practical Success of AdaBoost
Viola-Jones Face Detection (2001)

• First real-time object detection framework
Weak Learners (Haar wavelet features)
Weak Learners (Haar wavelet features)

- Each hypothesis is very weak.
- There are many possible features.
  - For a 24x24 detection region, more than 160,000 features

- AdaBoost!
  - Training is slow
  - Testing is fast
  - (inherent feature selection)
Brief Discussion on Gradient Boosting

Gradient boosting is **safe to skip** for Exam 2
Look at the AdaBoost Algorithm Again

- The format is similar to gradient descent!
  - If we consider the space of the weak learners (i.e., $g_t(\tilde{x})$) as the space of “weights”
  - This observation leads to a general class of boosting algorithms: gradient boosting
- XGBoost is one implementation of gradient boosting that is popular in practice
- See CASI 17.4 and the reference in CASI P.350 for more discussion

\[ \text{Given } D = \{(\tilde{x}_1, y_1), \ldots, (\tilde{x}_N, y_N)\} \]

\[ \text{Initialize } D_1(n) = \frac{1}{N} \text{ for all } n = 1, \ldots, N \]

\[ \text{For } t = 1, \ldots, T \]

\[ \text{Learn } g_t \text{ from } D_t \text{ (using decision stumps)} \]

\[ \text{Calculate } \epsilon_t = E_{in}(g_t) \]

\[ \text{Set } \alpha_t = \frac{1}{2} \ln \left( \frac{1-\epsilon_t}{\epsilon_t} \right) \]

\[ \text{Update } D_{t+1}(n) = \frac{1}{2t} D_t(n) e^{-\alpha ty_ng_t(\tilde{x}_n)} \]

\[ \text{Output } G(\tilde{x}) = \text{sign}(\sum_{t=1}^{T} \alpha_t g_t(\tilde{x})) \]

\[ \text{Initialize } G(\tilde{x}) = 0 \]

\[ \text{For } t = 1, \ldots, T \]

\[ G(\tilde{x}) \leftarrow G(\tilde{x}) + \alpha_t g_T(\tilde{x}) \]

\[ \text{Output } \text{sign}(G(\tilde{x})) \]
Gradient Boosting

- AdaBoost is a special case of Gradient Boosting
  - minimizing the exponential loss \( e_{\exp}(h(x), y) = e^{-yh(x)} \)
  - using decision stump as the weak learners

\[ f(x) = +1 \]

- \( e_{\exp} \) is a surrogate loss function of the binary classification error we care about
  - Minimizing an alternative error (loss function) is a common trick in ML, especially when the target loss function is hard to optimize.
  - There are some theoretical discussions on when doing this makes sense (“calibration”: whether minimizing the surrogate is consistent with minimizing the original loss).

Initialize \( G(\bar{x}) = 0 \)
For \( t = 1, ..., T \)
  \[ G(\bar{x}) \leftarrow G(\bar{x}) + \alpha_t g_T(\bar{x}) \]
Output \( \text{sign}(G(\bar{x})) \)

[Safe to Skip]