Lecture 6: Decision Tree, Random Forest, and Boosting

Tuo Zhao

Schools of ISyE and CSE, Georgia Tech
Tree?

It's a Christmas tree with a heap of presents underneath!

...We're not inviting you home next year.
Decision Trees
Decision Trees

- Decision trees have a long history in machine learning
  - The first popular algorithm dates back to 1979

- Very popular in many real world problems

- Intuitive to understand

- Easy to build
History

- EPAM- Elementary Perceiver and Memorizer
  - 1961: Cognitive simulation model of human concept learning
- 1966: CLS-Early algorithm for decision tree construction
- 1979: ID3 based on information theory
- 1993: C4.5 improved over ID3
- Also has history in statistics as CART (Classification and regression tree)
Motivation

- How do people make decisions?
  - Consider a variety of factors
  - Follow a logical path of checks

- Should I eat at this restaurant?
  - If there is no wait
    - Yes
  - If there is short wait and I am hungry
    - Yes
  - Else
    - No
Dear various parents, grandparents, co-workers, and other "not computer people."

We don't magically know how to do everything in every program. When we help you, we're usually just doing this:

1. Start
2. Find a menu item or button which looks related to what you want to do.
3. I can't find one. Pick one at random. I've tried them all.
4. Click it.
5. Have you been trying this for over half an hour?
   - Yes: Ask someone for help or give up.
   - No: Did it work?
     - Yes: You're done!
     - No: Google the name of the program plus a few words related to what you want to do. Follow any instructions.

Please print this flowchart out and tape it near your screen. Congratulations; you're now the local computer expert!
Example: Should We Play Tennis?

<table>
<thead>
<tr>
<th>Play Tennis</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Rainy</td>
<td>Cold</td>
<td>Normal</td>
<td>No</td>
</tr>
</tbody>
</table>

- If temperature is not hot
  - Play

- If outlook is overcast
  - Play tennis

- Otherwise
  - Don’t play tennis
Decision Tree

[Diagram of a decision tree with decision nodes for Temperature, Humidity, Outlook, and Windy, and leaf nodes indicating outcomes like Yes or No for weather prediction.]
Structure

A decision tree consists of

- **Nodes**
  - Tests for variables
- **Branches**
  - Results of tests
- **Leaves**
  - Classification
Function Class

- What functions can decision trees model?
  - Non-linear: very powerful function class
  - A decision tree can encode any Boolean function
  - Proof
    - Given a truth table for a function
    - Construct a path in the tree for each row of the table
    - Given a row as input, follow that path to the desired leaf (output)

- Problem: exponentially large trees!
Smaller Trees

Can we produce smaller decision trees for functions?

- Yes (Possible)
- Key decision factors
- Counter examples
  - Parity function: Return 1 on even inputs, 0 on odd inputs
  - Majority function: Return 1 if more than half of inputs are 1

Bias-Variance Tradeoff

- Bias: Representation Power of Decision Trees
- Variance: require a sample size exponential in depth
Fitting Decision Trees
What Makes a Good Tree?

- Small tree:
  - Occam’s razor: Simpler is better
  - Avoids over-fitting

- A decision tree may be human readable, but not use human logic!

- How do we build small trees that accurately capture data?

- Learning an optimal decision tree is computationally intractable
Greedy Algorithm

- We can get good trees by simple greedy algorithms
- Adjustments are usually to fix greedy selection problems
- Recursive:
  - Select the “best” variable, and generate child nodes: One for each possible value;
  - Partition samples using the possible values, and assign these subsets of samples to the child nodes;
  - Repeat for each child node until all samples associated with a node that are either all positive or all negative.
Variable Selection

- The best variable for partition
  - The most informative variable
  - Select the variable that is most informative about the labels
  - Classification error?

- Example:

\[ P(Y = 1 | X_1 = 1) = 0.75 \quad \text{v.s.} \quad P(Y = 1 | X_2 = 1) = 0.55 \]

Which to choose?
Information Theory

- The quantification of information
- Founded by Claude Shannon
- Simple concepts:
  - Entropy: \( H(X) = -\sum_x \mathbb{P}(X = x) \log \mathbb{P}(X = x) \)
  - Conditional Entropy: \( H(Y|X) = \sum_x \mathbb{P}(X = x) H(Y|X = x) \)
  - Information Gain: \( IG(Y|X) = H(Y) - H(Y|X) \)
- Select the variable with the highest information gain
Example: Should We Play Tennis?

<table>
<thead>
<tr>
<th>Play Tennis</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Rainy</td>
<td>Cold</td>
<td>Normal</td>
<td>No</td>
</tr>
</tbody>
</table>

\[
H(\text{Tennis}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.97
\]

\[
H(\text{Tennis}|\text{Out.} = \text{Sunny}) = -\frac{2}{2} \log_2 \frac{2}{2} - \frac{0}{2} \log_2 \frac{0}{2} = 0
\]

\[
H(\text{Tennis}|\text{Out.} = \text{Overcast}) = -\frac{0}{1} \log_2 \frac{0}{1} - \frac{1}{1} \log_2 \frac{1}{1} = 0
\]

\[
H(\text{Tennis}|\text{Out.} = \text{Rainy}) = -\frac{0}{2} \log_2 \frac{0}{2} - \frac{2}{2} \log_2 \frac{2}{2} = 0
\]

\[
H(\text{Tennis}|\text{Out.}) = \frac{2}{5} \times 0 + \frac{1}{5} \times 0 + \frac{2}{5} \times 0 = 0
\]
**Example: Should We Play Tennis?**

<table>
<thead>
<tr>
<th>Play Tennis</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Rainy</td>
<td>Cold</td>
<td>Normal</td>
<td>No</td>
</tr>
</tbody>
</table>

- \( IG(\text{Tennis}|Out.) = 0.97 - 0 = 0.97 \)
- If we knew the Outlook we’d be able to predict Tennis!
- Outlook is the variable to pick for our decision tree
When to Stop Training?

- All data have same label
  - Return that label
- No examples
  - Return majority label of all data
- No further splits possible
  - Return majority label of passed data
- If $\text{max IG} = 0$?
### Example: No Information Gain?

<table>
<thead>
<tr>
<th>Y</th>
<th>X₁</th>
<th>X₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Both features give 0 IG
- Once we divide the data, perfect classification!
- We need a little exploration sometimes (Unstable Equilibrium)
Decision Tree with Continuous Features

- Decision Stump
  - Threshold for binary classification:
    \[ \text{Check } X_j \geq \delta_j \text{ or } X_j < \delta_j. \]

- More than two classes:
  - Threshold for three classes:
    \[ \text{Check } X_j \geq \delta_j, \gamma_j \geq X_j < \delta_j \text{ or } \gamma_j < X_j. \]

- Decompose one node to two modes:
  - First node: Check \( X_j \geq \delta_j \) or \( X_j < \delta_j \),
  - Second node: If \( X_j < \delta_j \), check \( \gamma_j \geq X_j \) or \( \gamma_j < X_j \).
Decision Tree for Spam Classification

- ch$ < 0.0555
- ch$ > 0.0555
- hp < 0.405
- hp > 0.405
- CAPAVE < 2.7505
- CAPAVE > 2.7505
- 1999 < 0.58
- 1999 > 0.58
- receive < 0.125
- receive > 0.125
- edu < 0.045
- edu > 0.045
- business < 0.145
- business > 0.145
- our < 1.2
- our > 1.2
**Decision Tree for Spam Classification**

- **Sensitivity**: proportion of true spam identified
- **Specificity**: proportion of true email identified.
Decision Tree v.s. SVM

SVM outperforms Decision Tree.

AUC: Area Under Curve.
Bagging and Random Forest
Toy Example

- Nonlinear separable data.
- Optimal decision boundary: $X_1^2 + X_2^2 = 1$. 
**Toy Example: Decision Tree**

- **Sample size**: 200
- **7 branching nodes; 6 layers.**
- **Classification error**: 7.3% when $d = 2$; > 30% when $d = 10$. 
Model Averaging

Decision trees can be simple, but often produce noisy (bushy) or weak (stunted) classifiers.

- **Bagging** (Breiman, 1996): Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.

- **Boosting** (Freund & Shapire, 1996): Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.


In general Boosting > Random Forests > Bagging > Single Tree.
Bagging/Bootstrap Aggregation

Motivation: Average a given procedure over many samples to reduce the variance.

- Given a classifier $C(S, x)$, based on our training data $S$, producing a predicted class label at input point $x$.

- To bag $C$, we draw bootstrap samples $S_1, ..., S_B$ each of size N with replacement from the training data. Then

$$\hat{C}_{Bag} = \text{Majority Vote}\{C(S_b, x)\}_{b=1}^B.$$ 

- Bagging can dramatically reduce the variance of unstable procedures (like trees), leading to improved prediction.

- All simple structures in a tree are lost.
Toy Example: Bagging Trees

- 2 branching nodes; 2 layers.
- 5 dependent trees.
Toy Example: Bagging Trees

- A smoother decision boundary.
- Classification error: 3.2% (Single deeper tree 7.3%).
Random Forest

Bagging features and samples simultaneously:

- At each tree split, a random sample of $m$ features is drawn, and only those $m$ features are considered for splitting. Typically $m = \sqrt{d}$ or $\log_2 d$, where $d$ is the number of features.

- For each tree grown on a bootstrap sample, the error rate for observations left out of the bootstrap sample is monitored. This is called the “out-of-bag” error rate.

- random forests tries to improve on bagging by “de-correlating” the trees. Each tree has the same expectation.
Random Forest for Spam Classification

- RF outperforms SVM.
- 500 Trees.
Random Forest: Variable Importance Scores

- The bootstrap roughly covers 1/3 samples for each time.
- Do permutations over variables to check how important they are.
Boosting
Boosting

- Average many trees, each grown to re-weighted versions of the training data (Iterative).
- Final Classifier is weighted average of classifiers:

\[ C(x) = \text{sign} \left[ \sum_{M} \alpha_m C_m(x) \right] \]
Adaboost v.s. Bagging

- Each classifier for bagging is a tree with a single node (decision stump).
- 2000 samples from Spheres in $\mathbb{R}^{10}$.

Boosting Trevor Hastie, Stanford University 24

Number of Terms
Test Error
0 100 200 300 400
0.0 0.1 0.2 0.3 0.4

Bagging
AdaBoost

Boosting vs Bagging
• 2000 points from Nested Spheres in $\mathbb{R}^{10}$
• Bayes error rate is 0%
• Trees are grown best first without pruning.
• Leftmost term is a single tree.
Adaboost

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.

2. For $m = 1$ to $M$ repeat steps (a)–(d):
   (a) Fit a classifier $C_m(x)$ to the training data using weights $w_i$.
   (b) Compute weighted error of newest tree
   \[
   \text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq C_m(x_i))}{\sum_{i=1}^{N} w_i}.
   \]
   (c) Compute $\alpha_m = \log[(1 - \text{err}_m)/\text{err}_m]$.
   (d) Update weights for $i = 1, \ldots, N$:
   \[
   w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq C_m(x_i))]
   \]
   and renormalize to $w_i$ to sum to 1.

3. Output $C(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m C_m(x)\right]$.

- $w_i$’s are the weights of the samples.
- $\text{err}_m$ is the weighted training error.
Adaboost

- The ensemble of classifiers is much more efficient than the simple combination of classifiers.
Overfitting of Adaboost

- More iterations continue to improve test error in many examples.
- The Adaboost is often observed to be robust to overfitting.
Overfitting of Adaboost

- Optimal Bayes classification error: $25\%$.
- The test error does increase, but quite slowly.