Lecture 9:
Decision Trees, Bagging &
Random Forests
Announcements

• Homework 2 harder than anticipated
  – Will not happen again
  – Lenient grading

• Homework 3 Released Tonight
  – Due 2 weeks later
  – (Much easier than HW2)

• Homework 1 will be graded soon.
Announcements

• Kaggle Mini-Project Released
  – [https://kaggle.com/join/csee155](https://kaggle.com/join/csee155) (also linked to from course website)
  – Dataset of Financial Reports to SEC
  – Predict if Financial Report was during Great Recession
  – Training set of ~5000, with ~500 features
  – Submit predictions on test set
  – Competition ends in ~3 weeks
  – Report due 2 days after competition ends
  – **Grading**: 80% on cross validation & model selection, 20% on accuracy
  – If you do everything properly, you should get at least 90%
  – **Don’t panic! With proper planning, shouldn’t take that much time.**
  – Recitation tomorrow (short Kaggle & Decision Tree Tutorial)
This Week

• Back to standard classification & regression
  – No more structured models

• Focus to achieve highest possible accuracy
  – Decision Trees
  – Bagging
  – Random Forests
  – Boosting
  – Ensemble Selection
Decision Trees
(Binary) Decision Tree

Don’t overthink this, it is literally what it looks like.

<table>
<thead>
<tr>
<th>Person</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
(Binary) Decision Tree

- Every **internal node** has a binary query function $q(x)$.
- Every **leaf node** has a prediction, e.g., 0 or 1.
**Decision Tree**

**Input:** Alice  
Gender: Female  
Age: 14

**Prediction:** Height > 55”

---

Every **internal node** has a **binary** query function q(x).

Every **leaf node** has a prediction, e.g., 0 or 1.

Prediction starts at **root node**. Recursively calls query function.  
Positive response ➔ Left Child.  
Negative response ➔ Right Child.  
Repeat until Leaf Node.
Queries

• Decision Tree defined by Tree of Queries

• Binary query \( q(x) \) maps features to 0 or 1

• Basic form: \( q(x) = \mathbf{1}[x^d > c] \)
  
  – \( \mathbf{1}[x^3 > 5] \)
  
  – \( \mathbf{1}[x^1 > 0] \)
  
  – \( \mathbf{1}[x^{55} > 1.2] \)

• Axis aligned partitioning of input space
Basic Decision Tree Function Class

• “Piece-wise Static” Function Class
  – All possible partitionings over feature space.
  – Each partition has a static prediction.

• Partitions axis-aligned
  – E.g., No Diagonals

• (Extensions next week)
Decision Trees vs Linear Models

- Decision Trees are NON-LINEAR Models!

Example:

No Linear Model Can Achieve 0 Error

Simple Decision Tree Can Achieve 0 Error
Decision Trees vs Linear Models

• Decision Trees are NON-LINEAR Models!

• Example:

No Linear Model Can Achieve 0 Error

Simple Decision Tree Can Achieve 0 Error
Decision Trees vs Linear Models

• Decision Trees are AXIS-ALIGNED!
  – Cannot easily model diagonal boundaries

• Example:
  Simple Linear SVM can Easily Find Max Margin
  Decision Trees Require Complex Axis-Aligned Partitioning
  Wasted Boundary
More Extreme Example

Decision Tree wastes most of model capacity on useless boundaries.

(Depicting useful boundaries)
Decision Trees vs Linear Models

• Decision Trees are often more accurate!

• Non-linearity is often more important
  – Just use many axis-aligned boundaries to approximate diagonal boundaries
  – (It’s OK to waste model capacity.)

• **Catch:** requires sufficient training data
  – Will become clear later in lecture
Real Decision Trees

Can get much larger!

Image Source: http://www.biomedcentral.com/1471-2105/10/116
Decision Tree Training

Every node = partition/subset of $S$
Every Layer = complete partitioning of $S$
Children = complete partitioning of parent

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Decision Tree Training

Every node = partition/subset of S
Every Layer = complete partitioning of S
Children = complete partitioning of parent

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Thought Experiment

• What if just one node?
  – (i.e., just root node)
  – No queries
  – Single prediction for all data

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Simplest Decision Tree is just single Root Node (Also a Leaf Node)

Corresponds to Entire Training Set

Makes a Single Prediction: Majority class in training set
• What if 2 Levels?
  – (I.e., root node + 2 children)
  – Single query (which one?)
  – 2 predictions
  – How many possible queries?

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
#Possible Queries = #Possible Splits
= D*N
(D = #Features)
(N = #training points)

How to choose “best” query?
Impurity

• Define impurity function:

  – E.g., 0/1 Loss: $L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1[\hat{y} \neq y]$

![Diagram showing S, S1, S2 with L(S) = 1, L(S1) = 0, L(S2) = 1

Classification Error of best single prediction

Impurity Reduction = 0

No Benefit From This Split!
Impurity

- Define impurity function:

\[ L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1[\hat{y} \neq y] \]

- E.g., 0/1 Loss: 

\[ L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1[\hat{y} \neq y] \]

Classification Error of best single prediction

Classifica(on Error of best single prediction

No Benefit From This Split!


Impurity

• Define impurity function:

  - E.g., 0/1 Loss: \( L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1[\hat{y} \neq y] \)

  \[ \text{Classification Error of best single prediction} \]

  \[ \text{Choose Split with largest impurity reduction!} \]
Impurity = Loss Function

• **Training Goal:**
  – Find decision tree with low impurity.

• **Impurity Over Leaf Nodes = Training Loss**

\[
L(S) = \sum_{S'} L(S')
\]

\[
L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1[\hat{y} \neq y]
\]

*S’ iterates over leaf nodes*

*Union of S’ = S*

*(Leaf Nodes = partitioning of S)*

Classification Error on S’
Problems with 0/1 Loss

- What split best reduces impurity?

\[ L(S') = \min_{\hat{y} \in \{0,1\}} \sum_{(x,y) \in S'} 1[\hat{y} \neq y] \]

All Partitionings Give Same Impurity Reduction!

\( L(S) = 1 \)    \( L(S_1) = 0 \)    \( L(S_2) = 1 \)
Problems with 0/1 Loss

• 0/1 Loss is discontinuous

• A good partitioning may not improve 0/1 Loss...
  – E.g., leads to an accurate model with subsequent split...

\[ S \]

\[ S(S) = 1 \]

\[ L(S) = 1 \]

\[ S_1 \]

\[ S_2 \]

\[ L(S) = 1 \]

\[ S_1 \]

\[ S_2 \]

\[ S_3 \]

\[ L(S) = 0 \]
Surrogate Impurity Measures

• Want more continuous impurity measure

• First try: Bernoulli Variance:

\[ L(S') = |S'| p_{S'} (1 - p_{S'}) = \frac{\# \text{pos} \times \# \text{neg}}{|S'|} \]

\[ p_{S'} = \text{fraction of } S' \text{ that are positive examples} \]

- Perfect Purity
  - \( P = 1/2, \quad L(S') = |S'| \times 1/4 \)
  - \( P = 1, \quad L(S') = |S'| \times 0 \)
  - \( P = 0, \quad L(S') = |S'| \times 0 \)

- Worst Purity
  - Assuming \( |S'| = 1 \)

- Diagram showing the function \( L(S') \) vs. \( p_{S'} \)
Bernoulli Variance as Impurity

• What split best reduces impurity?

\[ L(S') = |S'| p_{S'}(1 - p_{S'}) = \frac{\# pos \times \# neg}{|S'|} \]

\( p_{S'} \) = fraction of \( S' \) that are positive examples

\[ L(S) = 5/6 \]
\[ L(S_1) = 0 \quad L(S_2) = 1/2 \]

Best!
Interpretation of Bernoulli Variance

• Assume each partition = distribution over $y$
  – $y$ is Bernoulli distributed with expected value $p_\mathcal{S}$.
  – **Goal**: partitioning where each $y$ has low variance

\[
\begin{align*}
L(S) & = 5/6 \\
L(S_1) & = 0 \\
L(S_2) & = 1/2 \\
L(S_1) & = 0 \\
L(S_2) & = 3/4 \\
\text{Best!}
\end{align*}
\]
Other Impurity Measures

• Entropy: \( L(S') = -|S'|\left( p_{S'} \log p_{S'} + (1 - p_{S'}) \log (1 - p_{S'}) \right) \)
  
  – aka: Information Gain:
  
  \( IG(A, B \mid S') = L(S') - L(A) - L(B) \)
  
  – (aka: Entropy Impurity Reduction)
  
  – Most popular.

• Gini Index:
  
  \( L(S') = |S'|\left( 1 - p_{S'}^2 - (1 - p_{S'})^2 \right) \)

Define: 0*\( \log(0) = 0 \)

See also: http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
(Terminology is slightly different.)
Other Impurity Measures

- **Entropy:**  
  \[ L(S') = -|S'| \left( p_{S'} \log p_{S'} + (1 - p_{S'}) \log (1 - p_{S'}) \right) \]

  – aka: **Information Gain:**
  \[ IG(A, B \mid S') = L(S') - L(A) - L(B) \]
  – (aka: Entropy Impurity Reduction)
  – Most popular.

Most Good Impurity Measures
Look Qualitatively The Same!

See also: http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
(Terminology is slightly different.)
Top-Down Training

- Define impurity measure $L(S')$
  - E.g., $L(S') = \text{Bernoulli Variance}$

**Loop:** Choose split with greatest impurity reduction (over all leaf nodes).

**Repeat:** until stopping condition.

Step 1:
$L(S) = 12/7$

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height $&gt; 55”$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

See TreeGrowing (Fig 9.2) in http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
Top-Down Training

- Define impurity measure $L(S')$
  - E.g., $L(S') = \text{Bernoulli Variance}$

**Loop:** Choose split with greatest impurity reduction (over all leaf nodes).

**Repeat:** until stopping condition.

Step 1:
$L(S) = 12/7$

Step 2:
$L(S) = 5/3$

See TreeGrowing (Fig 9.2) in http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
Top-Down Training

- Define impurity measure $L(S')$
  - E.g., $L(S') = \text{Bernoulli Variance}$

**Loop:** Choose split with greatest impurity reduction (over all leaf nodes).

**Repeat:** until stopping condition.

**Step 1:**
$L(S) = \frac{12}{7}$

**Step 2:**
$L(S) = \frac{5}{3}$

**Step 3:** Loop over all leaves, find best split.

$$L(S') = \frac{2}{3}$$
$$L(S') = 1$$

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

See TreeGrowing (Fig 9.2) in http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
Top-Down Training

• Define impurity measure $L(S')$
  – E.g., $L(S') = \text{Bernoulli Variance}$

Loop: Choose split with greatest impurity reduction (over all leaf nodes).
Repeat: until stopping condition.

Step 1:
$L(S) = 12/7$

Step 2:
$L(S) = 5/3$

Step 3:
$L(S) = 1$

See TreeGrowing (Fig 9.2) in http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
Top-Down Training

- Define impurity measure $L(S')$
  - E.g., $L(S') = \text{Bernoulli Variance}$

**Loop:** Choose split with greatest impurity reduction (over all leaf nodes).

**Repeat:** until stopping condition.

### Step 1:
$L(S) = \frac{12}{7}$

### Step 2:
$L(S) = \frac{5}{3}$

### Step 3:
$L(S) = \frac{2}{3}$

---

**Table:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

See TreeGrowing (Fig 9.2) in [http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf](http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf)
Top-Down Training

- Define impurity measure \( L(S') \)
  - E.g., \( L(S') = \text{Bernoulli Variance} \)

Loop: Choose split with greatest impurity reduction (over all leaf nodes).
Repeat: until stopping condition.

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Step 1: \( L(S) = 12/7 \)  
Step 2: \( L(S) = 5/3 \)  
Step 3: \( L(S) = 2/3 \)  
Step 4: \( L(S) = 0 \)

See TreeGrowing (Fig 9.2) in http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
Properties of Top-Down Training

• Every intermediate step is a decision tree
  – You can stop any time and have a model

• Greedy algorithm
  – Doesn’t backtrack
  – Cannot reconsider different higher-level splits.
When to Stop?

• In kept going, can learn tree with zero training error.
  – But such tree is probably overfitting to training set.
• How to stop training tree earlier?
  – I.e., how to regularize?

Which one has better test error?
Stopping Conditions (Regularizers)

- **Minimum Size**: do not split if resulting children are smaller than a minimum size.
  - Most common stopping condition.

- **Maximum Depth**: do not split if the resulting children are beyond some maximum depth of tree.

- **Maximum #Nodes**: do not split if tree already has maximum number of allowable nodes.

- **Minimum Reduction in Impurity**: do not split if resulting children do not reduce impurity by at least $\delta\%$.

See also, Section 5 in: http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
Pseudocode for Training

**Algorithm 1** TREE(): Initialize Decision (Sub-)Tree Data Structure

1: input: $S$ // data partition
2: input: $L$ // loss function
3: Initialize data structure $T$:
4: $T$.data $\leftarrow S$ // pointer to training data partition
5: $T$.q $\leftarrow$ NULL // decision query
6: $T$.left $\leftarrow$ NULL // subtree for positive query response
7: $T$.right $\leftarrow$ NULL // subtree for negative query response
8: $T$.l $\leftarrow L(S)$ // impurity/loss on training data partition
9: return: $T$

**Algorithm 3** TRAIN(): Top-Down Decision Tree Training

1: input: $S$, $Q$, $N_{\text{min}}$, $L$ // root node
2: $T \leftarrow$ TREE($S$)
3: repeat
4: $Q \leftarrow \emptyset$
5: for every leaf node $\tau$ in $T$ do
6: for every $q \in Q$ do
7: $S_1 \leftarrow \{(\hat{x}, \hat{y}) \in \tau$.data $\mid q(\hat{x}) = 1\}$
8: $S_2 \leftarrow \{(\hat{x}, \hat{y}) \in \tau$.data $\mid q(\hat{x}) = 0\}$
9: if $|S_1| \geq N_{\min} \land |S_2| \geq N_{\min}$ then
10: $\tau_1 \leftarrow$ TREE($S_1$, $L$)
11: $\tau_2 \leftarrow$ TREE($S_2$, $L$)
12: $Q \leftarrow Q \cup \{(\tau, \tau_1, \tau_2)\}$
13: end if
14: end for
15: end for
16: if $|Q| > 0$ then
17: $(\tau, q, \tau_1, \tau_2) \leftarrow \arg \min_{(\tau', q', \tau_1', \tau_2')} (\tau'.l - (\tau_1'.l + \tau_2'.l))$
18: $\tau.q \leftarrow q$
19: $\tau$.left $\leftarrow \tau_1$
20: $\tau$.right $\leftarrow \tau_2$
21: end if
22: until $|Q| = 0$
23: return: $T$

Stopping condition is minimum leaf node size: $N_{\text{min}}$

See TreeGrowing (Fig 9.2) in http://www.ise.bgu.ac.il/faculty/liorr/hbchap9.pdf
Classification vs Regression

<table>
<thead>
<tr>
<th>Classification</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labels are {0,1}</td>
<td>Labels are Real Valued</td>
</tr>
<tr>
<td>Predict Majority Class in Leaf Node</td>
<td>Predict Mean of Labels in Leaf Node</td>
</tr>
<tr>
<td>Piecewise Constant Function Class</td>
<td>Piecewise Constant Function Class</td>
</tr>
<tr>
<td>Goal: minimize 0/1 Loss</td>
<td>Goal: minimize squared loss</td>
</tr>
<tr>
<td>Impurity based on fraction of positives vs negatives</td>
<td>Impurity = Squared Loss</td>
</tr>
</tbody>
</table>
Recap: Decision Tree Training

• Train Top-Down
  – Iteratively split existing leaf node into 2 leaf nodes

• Minimize Impurity (= Training Loss)
  – E.g., Entropy

• Until Stopping Condition (= Regularization)
  – E.g., Minimum Node Size

• Finding optimal tree is intractable
  – E.g., tree satisfying minimal leaf sizes with lowest impurity.
Recap: Decision Trees

• Piecewise Constant Model Class
  – Non-linear!
  – Axis-aligned partitions of feature space

• Train to minimize impurity of training data in leaf partitions
  – Top-Down Greedy Training

• Often more accurate than linear models
  – If enough training data
Bagging
(Bootstrap Aggregation)
Outline

• Recap: Bias/Variance Tradeoff

• Bagging
  – Method for minimizing variance
  – Not specific to Decision Trees

• Random Forests
  – Extension of Bagging
  – Specific to Decision Trees
Outline

• Recap: Bias/Variance Tradeoff

• Bagging
  – Method for minimizing variance
  – Not specific to Decision Trees

• Random Forests
  – Extension of Bagging
  – Specific to Decision Trees
Test Error

• “True” distribution: \( P(x,y) \)
  – Unknown to us

• Train: \( h_S(x) = y \)
  – Using training data: \( S = \{(x_i, y_i)\}_{i=1}^N \)
  – Sampled from \( P(x,y) \)

• Test Error:
  \[
  L_P(h_S) = E_{(x,y) \sim P(x,y)} \left[ L(y, h_S(x)) \right]
  \]

• Overfitting: Test Error >> Training Error
<table>
<thead>
<tr>
<th>Person</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>James</td>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Jessica</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Amy</td>
<td>12</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Xavier</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Cathy</td>
<td>9</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Eugene</td>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Rafael</td>
<td>12</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Peter</td>
<td>9</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Henry</td>
<td>13</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Rose</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Iain</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Paulo</td>
<td>12</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Margaret</td>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Jill</td>
<td>13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Leon</td>
<td>10</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Sarah</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Gena</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Patrick</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Person</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Test Error:
\[ \mathcal{L}(h) = E_{(x,y)\sim P(x,y)}[ L(h(x),y) ] \]
Bias-Variance Decomposition

\[ E_S[L_P(h_S)] = E_S[E_{(x,y)\sim P(x,y)}[L(y, h_S(x))]] \]

- For squared error:

\[ E_S[L_P(h_S)] = E_{(x,y)\sim P(x,y)}[E_S[(h_S(x) - H(x))^2] + (H(x) - y)^2] \]

\[ H(x) = E_S[h_S(x)] \]

“Average prediction on x”

Variance Term

Bias Term
Example $P(x,y)$
$h_S(x)$ Linear
$h_S(x)$ Quadratic
$h_S(x)$ Cubic
Bias-Variance Trade-off

The diagrams illustrate the bias-variance trade-off in machine learning models. The graphs on the left show the relationship between bias and variance for different model complexities. The graphs on the right highlight the concept of overfitting and underfitting, where a model with high variance underfits the data, and a model with high bias overfits the data. The diagrams also demonstrate the optimal model complexity balancing both bias and variance to achieve the best performance.
Overfitting vs Underfitting

- High variance implies **overfitting**
  - Model class unstable
  - Variance increases with model complexity
  - Variance reduces with more training data.

- High bias implies **underfitting**
  - Even with no variance, model class has high error
  - Bias decreases with model complexity
  - Independent of training data size
Decision Trees are Low Bias, High Variance Models
Unless you Regularize a lot...
...but then often worse than Linear Models

Highly Non-Linear, Can Easily Overfit

Different Training Samples Can Lead to Very Different Trees
Bagging

- **Goal:** reduce variance

- **Ideal setting:** many training sets $S'$
  - Train model using each $S'$
  - Average predictions

\[
E_S[(h_S(x) - y)^2] = E_S[(Z - \bar{z})^2] + \bar{z}^2
\]

Expected Error
On single (x,y)

Variance
Bias

\[
Z = h_S(x) - y
\]

\[
\bar{z} = E_S[Z]
\]

“Bagging Predictors” [Leo Breiman, 1994]

http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf
Bagging

- **Goal**: reduce variance

- **In practice**: resample $S'$ with replacement
  - Train model using each $S'$
  - Average predictions

\[
E_S[(h_S(x) - y)^2] = E_S[(Z - \bar{Z})^2] + \bar{Z}^2
\]

Expected Error
On single (x,y)

\[Z = h_S(x) - y\]
\[\bar{Z} = E_S[Z]\]

Variance

Bias

Variance reduces sub-linearly
(Because $S'$ are correlated)
Bias often increases slightly

“Bagging Predictors” [Leo Breiman, 1994]
http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf
Recap: Bagging for DTs

- **Given:** Training Set $S$
- **Bagging:** Generate Many Bootstrap Samples $S'$
  - Sampled with replacement from $S$
    - $|S'| = |S|$
  - Train Minimally Regularized DT on $S'$
    - High Variance, Low Bias
- **Final Predictor:** Average of all DTs
  - Averaging reduces variance
“An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants”
Why Bagging Works

• Define Ideal Aggregation Predictor $h_A(x)$:
  – Each $S'$ drawn from true distribution $P$
    $$h_A(x) = E_{S \sim P(x,y)}[h_S(x)]$$

• We will first compare the error of $h_A(x)$ vs $h_S(x)$

• Then show how to adapt comparison to Bagging

“Bagging Predictors” [Leo Breiman, 1994]
http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf
Analysis of Ideal Aggregate Predictor (Squared Loss)

\[ h_A(x) = E_{S \sim P(x,y)}[h_S(x)] \]

Expected Loss of \( h_S \) on single \((x,y)\):

\[
E_S[L(y, h_S(x))] = E_S[(y - h_S(x))^2]
\]

\[
= E_S[y^2] - 2E_S[yh_S(x)] + E_S[h_S(x)^2]
\]

\[
\geq y^2 - 2yE_S[h_S(x)] + E_S[h_S(x)^2]
\]

\[
= y^2 - 2yh_A(x) + h_A(x)^2
\]

\[
= (y - h_A(x))^2
\]

\[
= L(y, h_A(x))
\]

“Bagging Predictors” [Leo Breiman, 1994]

http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf
Key Insight

• Ideal Aggregate Predictor Improves if:

\[ E_S \left[ h_S(x)^2 \right] > E_S \left[ h_S(x) \right]^2 = h_A(x)^2 \]

Large improvement if \( h_S(x) \) is “unstable” (high variance)
\( h_A(x) \) is guaranteed to be at least as good as \( h_S(x) \).

• Bagging Predictor Improves if:

\[ E_S \left[ h_S(x)^2 \right] > E_S \left[ E_{S'} \left[ h_{S'}(x)^2 \right] \right] = E_S \left[ h_B(x)^2 \right] \]

Improves if \( h_B(x) \) is much more stable than \( h_S(x) \)
\( h_B(x) \) can sometimes be more unstable than \( h_S(x) \)
Bias of \( h_B(x) \) can be worse than \( h_S(x) \).

“Bagging Predictors” [Leo Breiman, 1994]
http://statistics.berkeley.edu/sites/default/files/tech-reports/421.pdf
Random Forests
Random Forests

• **Goal:** reduce variance
  – Bagging can only do so much
  – Resampling training data asymptotes

• **Random Forests:** sample data & features!
  – Sample $S'$
  – Train DT
    • At each node, sample features
  – Average predictions

http://oz.berkeley.edu/~breiman/random-forests.pdf
Top-Down Random Forest Training

**Loop:** Sample $T$ random splits at each Leaf. Choose split with greatest impurity reduction.

**Repeat:** until stopping condition.

Step 1:

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>


http://oz.berkeley.edu/~breiman/random-forests.pdf
Top-Down Random Forest Training

**Loop:** Sample $T$ random splits at each Leaf. Choose split with greatest impurity reduction.

**Repeat:** until stopping condition.

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Step 1:

Step 2:

Randomly decide only look at age, Not gender.

http://oz.berkeley.edu/~breiman/random-forests.pdf
Top-Down Random Forest Training

**Loop:** Sample \( T \) random splits at each Leaf. Choose split with greatest impurity reduction.  
**Repeat:** until stopping condition.

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Step 1:**

```
Try Age>9?
```

**Step 2:**

```
Male?
```

**Step 3:**

```
1 1
```

Randomly decide only look at gender.

[http://oz.berkeley.edu/~breiman/random-forests.pdf](http://oz.berkeley.edu/~breiman/random-forests.pdf)
Top-Down Random Forest Training

Loop: Sample T random splits at each Leaf. Choose split with greatest impurity reduction.
Repeat: until stopping condition.

Step 1:

Age>9?

Try

1

Age>8?

1 0

Step 2:

Step 3:

Randomly decide only look at age.

<table>
<thead>
<tr>
<th>Name</th>
<th>Age</th>
<th>Male?</th>
<th>Height &gt; 55”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>14</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bob</td>
<td>10</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Carol</td>
<td>13</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dave</td>
<td>8</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Erin</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Frank</td>
<td>9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Gena</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

http://oz.berkeley.edu/~breiman/random-forests.pdf
Recap: Random Forests

• Extension of Bagging to sampling Features

• Generate Bootstrap $S'$ from $S$
  – Train DT Top-Down on $S'$
  – Each node, sample subset of features for splitting
    • Can also sample a subset of splits as well

• Average Predictions of all DTs

http://oz.berkeley.edu/~breiman/random-forests.pdf
Average performance over many datasets
Random Forests perform the best

“An Empirical Evaluation of Supervised Learning in High Dimensions”
Caruana, Karampatziakis & Yessenalina, ICML 2008
Next Lecture

• **Boosting**
  – Method for reducing bias

• **Ensemble Selection**
  – Very general method for combining classifiers
  – Multiple-time winner of ML competitions

• **Recitation Tomorrow:**
  – Brief Tutorial of Kaggle
  – Brief Tutorial of Decision Tree Scikit-Learn Package
    • Although many decision tree packages available online.