Machine Learning and Data Mining

Decision Trees

Kalev Kask
Decision trees

- Functional form $f(x; \mu)$: nested “if-then-else” statements
  - Discrete features: fully expressive (any function)

- Structure:
  - Internal nodes: check feature, branch on value
  - Leaf nodes: output prediction

```
if X1: # branch on feature at root
  if X2: return +1 # if true, branch on right child feature
  else: return -1 # & return leaf value
else: # left branch:
  if X2: return -1 # branch on left child feature
  else: return +1 # & return leaf value
```

“XOR”

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Parameters?
Tree structure, features, and leaf outputs
Decision trees

- Real-valued features
  - Compare feature value to some threshold
Decision trees

- **Categorical variables**
  - Could have one child per value
  - Binary splits: single values, or by subsets

The discrete variable will not appear again below here...

Could appear again multiple times...

(This ^^^ is easy to implement using a 1-of-K representation…)
Decision trees

- “Complexity” of function depends on the depth
- A depth-1 decision tree is called a decision “stump”
  - Simpler than a linear classifier!
Decision trees

- “Complexity” of function depends on the depth
- More splits provide a finer-grained partitioning

Depth $d = \text{up to } 2^d \text{ regions & predictions}$
Decision trees for regression

- Exactly the same
- Predict real valued numbers at leaf nodes
- Examples on a single scalar feature:
Machine Learning and Data Mining

Learning Decision Trees

Kalev Kask
Learning decision trees

• Break into two parts
  – Should this be a leaf node?
  – If so: what should we predict?
  – If not: how should we further split the data?

• Leaf nodes: best prediction given this data subset
  – Classify: pick majority class;    Regress: predict average value

• Non-leaf nodes: pick a feature and a split
  – Greedy: “score” all possible features and splits
  – Score function measures “purity” of data after split
    • How much easier is our prediction task after we divide the data?

• When to make a leaf node?
  – All training examples the same class (correct), or indistinguishable
  – Fixed depth (fixed complexity decision boundary)
  – Others …

Example algorithms: ID3, C4.5
See e.g. wikipedia, “Classification and regression tree”
Learning decision trees

Algorithm 1 BuildTree(D): Greedy training of a decision tree

Input: A data set $D = (X, Y)$.

Output: A decision tree.

if LeafCondition($D$) then
    $f_n = $ FindBestPrediction($D$)
else
    $j_n, t_n = $ FindBestSplit($D$)
    $D_L = \{(x^{(i)}, y^{(i)}) : x_{j_n}^{(i)} < t_n\}$ and
    $D_R = \{(x^{(i)}, y^{(i)}) : x_{j_n}^{(i)} \geq t_n\}$
    leftChild = BuildTree($D_L$)
    rightChild = BuildTree($D_R$)
end if
Scoring decision tree splits

- Suppose we are considering splitting feature 1
  - How can we score any particular split?
    - “Impurity” – how easy is the prediction problem in the leaves?
- “Greedy” – could choose split with the best accuracy
  - Assume we have to predict a value next
  - MSE (regression)
  - 0/1 loss (classification)
- But: “soft” score can work better
Entropy and information

• “Entropy” is a measure of randomness
  – How hard is it to communicate a result to you?
  – Depends on the probability of the outcomes

• Communicating fair coin tosses
  – Output: H H T H T T T H H H H T …
  – Sequence takes n bits – each outcome totally unpredictable

• Communicating my daily lottery results
  – Output: 0 0 0 0 0 0 …
  – Most likely to take one bit – I lost every day.
  – Small chance I’ll have to send more bits (won & when)

• Takes less work to communicate because it’s less random
  – Use a few bits for the most likely outcome, more for less likely ones
Entropy and information

- Entropy \( H(x) \) \( = \sum p(x) \log \frac{1}{p(x)} \)
  - Log base two, units of entropy are “bits”
  - Two outcomes: \( H = - p \log(p) - (1-p) \log(1-p) \)

- Examples:
  - \( H(x) = \frac{1}{4} \log 4 + \frac{1}{4} \log 4 + \frac{1}{4} \log 4 + \frac{1}{4} \log 4 = \log 4 = 2 \text{ bits} \)
  - \( H(x) = 0.75 \log 4/3 + 0.25 \log 4 \quad \frac{1}{4} \cdot 0.8133 \text{ bits} \)
  - \( H(x) = \log 1 = 0 \text{ bits} \)

Max entropy for 4 outcomes

Min entropy
Entropy and information

- Information gain
  - How much is entropy reduced by measurement?
- Information: expected information gain

\[
\text{Information} = \frac{13}{18} \times (.99 - .77) + \frac{5}{18} \times (.99 - 0)
\]

Equivalent: \( \sum p(s,c) \log \left[ \frac{p(s,c)}{p(s) \ p(c)} \right] \)
\[
= \frac{10}{18} \log \left[ \frac{10/18}{13/18 \ (10/18)} \right] + \frac{3}{18} \log \left[ \frac{3/18}{(13/18)(8/18)} \right] + \ldots
\]
Entropy and information

- Information gain
  - How much is entropy reduced by measurement?
- Information: expected information gain

Information = \frac{17}{18} \times (0.99 - 0.97) + \frac{1}{18} \times (0.99 - 0) 

Less information reduction – a less desirable split of the data
Gini index & impurity

- An alternative to information gain
  - Measures variance in the allocation (instead of entropy)
- $H_{\text{gini}} = \sum_c p(c) (1-p(c))$ vs. $H_{\text{ent}} = -\sum_c p(c) \log p(c)$

Gini Index = $\frac{13}{18} \times (0.494 - 0.355) + \frac{5}{18} \times (0.494 - 0)$
Entropy vs Gini impurity

- The two are nearly the same…
  - Pick whichever one you like
For regression

- Most common is to measure variance reduction
  - Equivalent to “information gain” in a Gaussian model…

\[ \text{Var reduction} = \frac{4}{10} \times (.25 - .1) + \frac{6}{10} \times (.25 - .2) \]
Scoring decision tree splits

Algorithm 1 FindBestSplit(D)

**Input:** A data set $D = (X, Y)$ of size $m$; impurity function $H(\cdot)$.

**Output:** A split $j^*, t^*$ minimizing impurity $H$

Initialize $H^* = 0$

for each feature $j$ do

Sort $\{x_j^{(i)}\}$ in order of increasing value

for each $i$ such that $x^{(i)} < x^{(i+1)}$ do

Compute $p_c^L = \frac{1}{i} \sum_{k \leq i} 1[y^{(k)} = c]$

and $p_c^R = \frac{1}{k-i} \sum_{k > i} 1[y^{(k)} = c]$

Set $H' = \frac{i}{m} H(p^L) + \frac{m-i}{m} H(p^R)$

if $H' < H^*$ then

Set $j^* = j$, $t^* = (x^{(i)} - x^{(i+1)})/2$, $H^* = H'$

end if

end for

end for

Return $j^*, t^*$
Building a decision tree

Algorithm 1 BuildTree($D$): Greedy training of a decision tree

Input: A data set $D = (X, Y)$.

Output: A decision tree.

if LeafCondition($D$) then
  $f_n = \text{FindBestPrediction}(D)$
else
  $j_n, t_n = \text{FindBestSplit}(D)$
  $D_L = \{(x^{(i)}, y^{(i)}) : x^{(i)}_{j_n} < t_n\}$ and
  $D_R = \{(x^{(i)}, y^{(i)}) : x^{(i)}_{j_n} \geq t_n\}$
  leftChild = BuildTree($D_L$)
  rightChild = BuildTree($D_R$)
end if

Stopping conditions:
* # of data < K
* Depth > D
* All data indistinguishable (discrete features)
* Prediction sufficiently accurate

* Information gain threshold?
  Often not a good idea!
  No single split improves, but, two splits do.
  Better: build full tree, then prune
Example

Restaurant data:

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_2$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
</tr>
<tr>
<td>$X_3$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_4$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
</tr>
<tr>
<td>$X_5$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>$X_6$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_7$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>$X_8$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_9$</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>$X_{12}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
</tr>
</tbody>
</table>

Split on:

Root entropy: $0.5 \times \log(2) + 0.5 \times \log(2) = 1$ bit

Leaf entropies: $2/12 \times 1 + 2/12 \times 1 + \ldots = 1$ bit

No reduction!
### Example

**Restaurant data:**

<table>
<thead>
<tr>
<th>Example</th>
<th>Alt</th>
<th>Bar</th>
<th>Fri</th>
<th>Hun</th>
<th>Pat</th>
<th>Price</th>
<th>Rain</th>
<th>Res</th>
<th>Type</th>
<th>Est</th>
<th>Wait</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_2$</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>30–60</td>
<td>F</td>
</tr>
<tr>
<td>$X_3$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>Some</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_4$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>10–30</td>
<td>T</td>
</tr>
<tr>
<td>$X_5$</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>French</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>$X_6$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Italian</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_7$</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>$X_8$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>Some</td>
<td>$$</td>
<td>T</td>
<td>T</td>
<td>Thai</td>
<td>0–10</td>
<td>T</td>
</tr>
<tr>
<td>$X_9$</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>Full</td>
<td>$</td>
<td>T</td>
<td>F</td>
<td>Burger</td>
<td>&gt;60</td>
<td>F</td>
</tr>
<tr>
<td>$X_{10}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$$$</td>
<td>F</td>
<td>T</td>
<td>Italian</td>
<td>10–30</td>
<td>F</td>
</tr>
<tr>
<td>$X_{11}$</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>None</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Thai</td>
<td>0–10</td>
<td>F</td>
</tr>
<tr>
<td>$X_{12}$</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>Full</td>
<td>$</td>
<td>F</td>
<td>F</td>
<td>Burger</td>
<td>30–60</td>
<td>T</td>
</tr>
</tbody>
</table>

**Split on:**

Root entropy: 0.5 * \(\log(2)\) + 0.5 * \(\log(2)\) = 1 bit

Leaf entropies: \(2/12 \times 0 + 4/12 \times 0 + 6/12 \times 0.9\)

Lower entropy after split!
Controlling complexity

- Maximum depth cutoff
Controlling complexity

- Minimum # parent data

- Alternate (similar): min # of data per leaf
Computational complexity

- “FindBestSplit”: on $M'$ data
  - Try each feature: $N$ features
  - Sort data: $O(M' \log M')$
  - Try each split: update $p$, find $H(p)$: $O(M \times C)$
  - Total: $O(N M' \log M')$

- “BuildTree”:
  - Root has $M$ data points: $O(N M \log M)$
  - Next level has $M$ *total* data points:
    $$O(N M_L \log M_L) + O(N M_R \log M_R) < O(N M \log M)$$
  - ...
Decision trees in python

- Many implementations
- Class implementation:
  - real-valued features (can use 1-of-k for discrete)
  - Uses entropy (easy to extend)

```python
T = dt.treeClassify()
T.train(X, Y, maxDepth=2)
print T

if x[0] < 5.602476:
    if x[1] < 3.009747:
        Predict 1.0  # green
    else:
        Predict 0.0  # blue
else:
    if x[0] < 6.186588:
        Predict 1.0  # green
    else:
        Predict 2.0  # red

ml.plotClassify2D(T, X, Y)
```
Summary

• Decision trees
  – Flexible functional form
  – At each level, pick a variable and split condition
  – At leaves, predict a value

• Learning decision trees
  – Score all splits & pick best
    • Classification: Information gain, Gini index
    • Regression: Expected variance reduction
  – Stopping criteria

• Complexity depends on depth
  – Decision stumps: very simple classifiers