Decision Trees

IT'S A CHRISTMAS TREE WITH A HEAP OF PRESENTS UNDERNEATH!
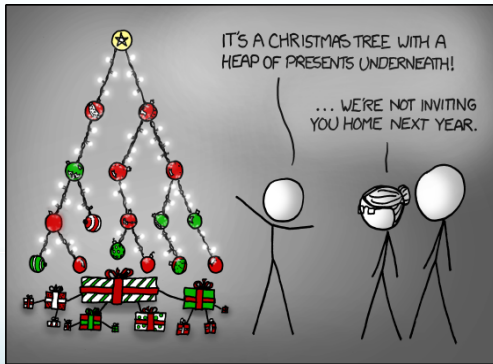
... WE'RE NOT INVITING YOU HOME NEXT YEAR.

Mark Dredze

Machine Learning
CS 600.475

Decision Trees

- Decision trees have a long history in ML
  - First popular algorithms 1979
- Very popular in many real world settings
- Intuitive to understand
- Easy to build

---

History

- EPAM- Elementary Perceiver and Memorizer
  - Feigenbaum 1961
  - Cognitive simulation model of human concept learning
- CLS- Early algorithm for decision tree construction
  - Hunt 1966
- ID3 based on information theory
  - Quinlan 1979
- C4.5 improved over ID3
  - Quinlan 1993
- 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

Decision Graph Example

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:
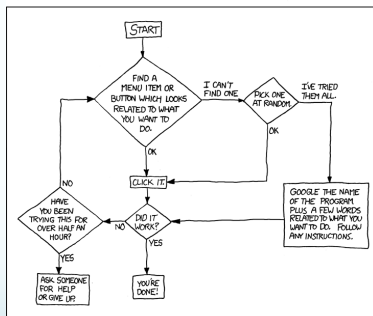
PLEASE PRINT THIS FLOWCHART OUT AND TAPE IT NEAR YOUR SCREEN. CONGRATULATIONS; YOU'RE NOW THE LOCAL COMPUTER EXPERT!

http://www.xkcd.com/627/

Example: Should We Play Tennis?

Play Tennis	Outlook	Temperature	Humidity	Windy
No	Sunny	Hot	High	No
No	Sunny	Hot	High	Yes
Yes	Overcast	Hot	High	No
Yes	Rainy	Mild	High	No
Yes	Rainy	Cold	Normal	No

y x

- If temperature is not hot
  - Play tennis
- If outlook is overcast
  - Play tennis
- Otherwise
  - Don't play tennis

Decision Tree

Play Tennis	Outlook	Temperature	Humidity	Windy
Yes	Sunny	Cold	Normal	No
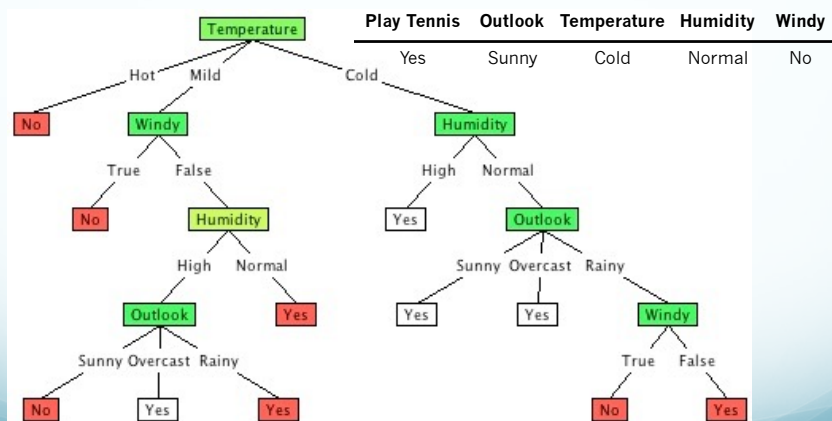
Decision Trees

- A decision tree is formed of
  - Nodes
    - Attribute tests
  - Branches
    - Results of attribute tests
  - Leaves
    - Classifications

Hypothesis Class

- What functions can decision trees model?
  - Non-linear: very powerful hypothesis 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)

Y	X_1	X_2	X_3
1	0	0	0
0	0	0	1
1	0	1	0

- Problem: exponentially large trees!

Smaller Trees

- Can we produce smaller decision trees for functions?
  - Yes (most of the time)
  - 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
- Decision trees are good for some functions but bad for others
- Recall: tradeoff between hypothesis class expressiveness and learnability

Decision Trees
Fitting a function to data

- Fitting: ???
- Function: any boolean function
- Data: Batch: construct a tree using all the data

Building Decision Trees

What Makes a Good Tree?

- Small
  - Ockham's razor
    - Simpler is better
  - Avoids over-fitting
    - We'll discuss this again later

- A decision tree may be human readable, but not use human logic
  - The decision tree you would write for a problem may differ from computer

Small Trees

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

- Optimal decision tree learning is NP-complete

- Constructing Optimal Binary Decision Trees is NP-complete. Laurent Hyafil, RL Rivest. Information Processing Letters, Vol. 5, No. 1. (1976), pp. 15-17.

Greedy Algorithms

- Like many NP-complete problems we can get pretty good solutions

- Most decision tree learning is by greedy algorithms
  - Adjustments are usually to fix greedy selection problems

- Top down decision tree learning
  - Recursive algorithms

ID3

- function BuildDecisionTree(data, labels):
  - if all labels are the same
    - return leaf node for that label
  - else
    - let f be the best feature for splitting
    - left = BuildDecisionTree(data with f=0, labels with f=0)
    - right= BuildDecisionTree(data with f=1, labels with f=1)
    - return Tree(f, left, right)

Does this always terminate?

Base Cases

- 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

ID3

- function BuildDecisionTree(data, labels):
  - if all labels are the same
    - return leaf node for that label
  - else
    - let f be the best feature for splitting
    - left = BuildDecisionTree(data with f=0, labels with f=0)
    - right= BuildDecisionTree(data with f=1, labels with f=1)
    - return Tree(f, left, right)

Selecting Features

- The best feature for splitting
  - The most *informative feature*
  - Select the feature that is most informative about the labels

- Information theory

Information Theory

- The quantification of information

- Founded by Claude Shannon
  - Landmark paper in 1948
  - Noisy channel theorem
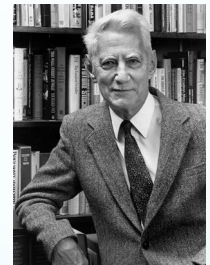
Information Theory

- A brief introduction...

Information Theory

- Entropy

$$H(X) = -\sum_x p(X = x) \log p(X = x)$$

- Conditional Entropy

$$H(Y \mid X) = \sum_x p(X = x) H(Y \mid x = x)$$

- Information Gain

$$IG(Y \mid X) = H(Y) - H(Y \mid X)$$

Selecting Features

- The best feature for splitting
  - The most *informative feature*
  - The feature with the highest *information gain*

Notes for Decision Trees

- Since we compare H(Y|X) across all features, H(Y) is a constant
  - We can omit it for comparisons
- The base of the log doesn't matter as long as it is consistent

Example: Should We Play Tennis?

Play Tennis	Outlook	Temperature	Humidity	Windy
No	Sunny	Hot	High	No
No	Sunny	Hot	High	Yes
Yes	Overcast	Hot	High	No
Yes	Rainy	Mild	High	No
Yes	Rainy	Cold	Normal	No

- $H(\text{Tennis}) = -3/5 \log_2 3/5 - 2/5 \log_2 2/5 = 0.97$

Example: Should We Play Tennis?

Play Tennis	Outlook	Temperature	Humidity	Windy
No	Sunny	Hot	High	No
No	Sunny	Hot	High	Yes
Yes	Overcast	Hot	High	No
Yes	Rainy	Mild	High	No
Yes	Rainy	Cold	Normal	No

- $H(\text{Tennis}|\text{Outlook=Sunny}) = -2/2 \log_2 2/2 - 0/2 \log_2 0/2 = 0$
- $H(\text{Tennis}|\text{Outlook=Overcast}) = -0/1 \log_2 0/1 - 1/1 \log_2 1/1 = 0$
- $H(\text{Tennis}|\text{Outlook=Rainy}) = -0/2 \log_2 0/2 - 2/2 \log_2 2/2 = 0$
- $H(\text{Tennis}|\text{Outlook}) = 2/5 * 0 + 1/5 * 0 + 2/5 * 0 = 0$

Example: Should We Play Tennis?

Play Tennis	Outlook	Temperature	Humidity	Windy
No	Sunny	Hot	High	No
No	Sunny	Hot	High	Yes
Yes	Overcast	Hot	High	No
Yes	Rainy	Mild	High	No
Yes	Rainy	Cold	Normal	No

- $IG(\text{Tennis}|\text{Outlook}) = 0.97 - 0 = 0.97$
- If we knew the Outlook we'd be able to perfectly predict Tennis!

- Outlook is a great feature to pick for our decision tree

ID3

- function BuildDecisionTree(data, labels):
  - if basecase
    - return appropriate leaf node
  - Else
    - $f = \arg\max IG(\text{label}|f)$
    - left = BuildDecisionTree(data with f=0, labels with f=0)
    - right= BuildDecisionTree(data with f=1, labels with f=1)
    - return Tree(f, left, right)

Base Cases

- 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 max IG = 0?

IG=0 As a Base Case

- Consider the following

Y	X_1	X_2
0	0	0
1	0	1
1	1	0
0	1	1

- Both features give 0 IG

- Once we divide the data, perfect classification!

Training vs. Test Accuracy

- On the tree shown for tennis

- 100% training accuracy

- 30% testing accuracy

- Why?

Over-fitting

- X_5 perfectly predicts Y

- Let's randomly flip Y with probability ¼

- X_5 will be the first split

- But tree will keep going
  - Duplicate training data into train/test
  - Train accuracy will be 100%
  - Test accuracy will be 62.5% (5/8)
    - 1/16 examples are doubly corrupted
    - 9/16 are uncorrupted
    - 6/16 will be bad
  - Single node test accuracy: 75%

Y	X_1	X_2	X_3	X_4	X_5
0	0	0	0	0	0
1	0	0	0	0	1
0	0	0	0	1	0
1	0	0	0	1	1
0	0	0	1	0	0
1	0	0	1	0	1
0	0	0	1	1	0
1	0	0	1	1	1

··· 32 examples total···

Bias/Variance Tradeoff

- Complete trees have no bias
  - But can over-fit badly
  - Lots of variance
- 0 depth trees (return most likely label) have no variance
  - Totally biased towards majority label
- A good tree balances between these two
  - How do we learn balanced trees?

Pruning: New Base Cases

- Stop when too few examples
- Stop when max depth reached
  - You will use this for HW
- Stop when my classification error is not much more than average of my children
- X^2 pruning- stop when remainder is no more likely than chance

Parameters

- All of these are parameters
- How do you select parameter values?
  - Train data?
  - Test data?
  - Development data!

Extensions

- Non-binary attributes
  - Categorical
  - Continuous (real valued)
    - In homework
    - Handle by thresholding- find the best single threshold to split the range
    - Regression trees
- Missing attributes
- Alternatives to information gain: Gini index, miss-classification rate
- Non-greedy algorithms?

Decision Trees
Fitting a function to data

- Fitting: greedy algorithm to find a good tree
  - Extra heuristics to help with over-fitting
  - Optimal decision tree learning: NP-complete
- Function: any boolean function
- Data: Batch: construct a tree using all the data