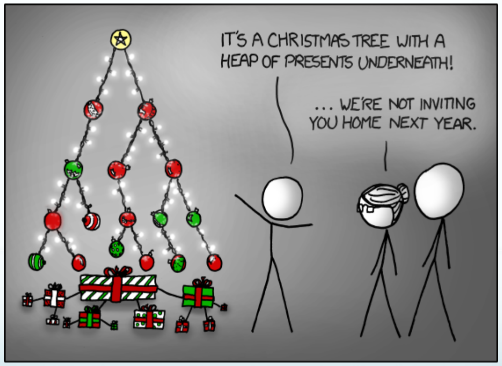
Decision Trees



Mark Dredze

Machine Learning
CS 601.475

Decision Trees

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

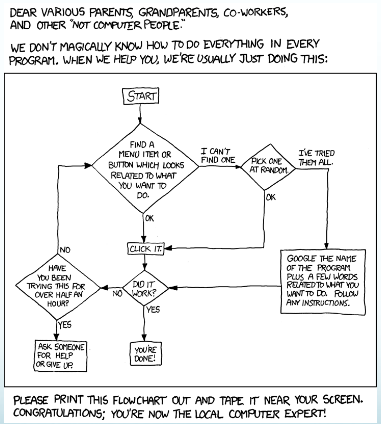
History

- Elementary Perceiver and Memorizer (EPAM)
  - 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?
  - No wait -> Yes
  - Short wait and hungry -> Yes
  - Else -> No

Decision Graph Example



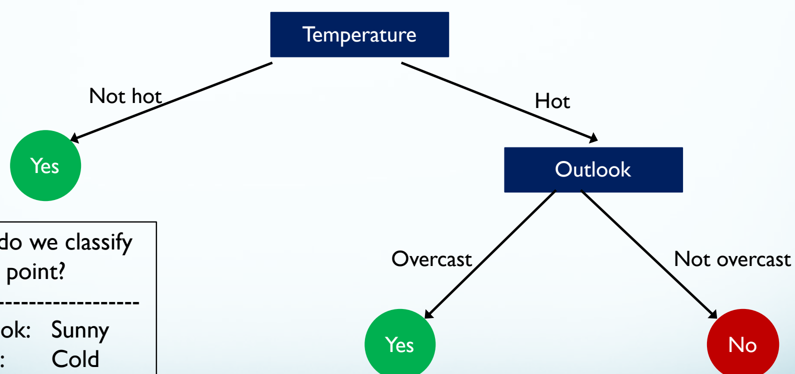
http://www.xkcd.com/627/

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 $\underbrace{\qquad\qquad\qquad\qquad}_{x}$

Should we play tennis?



How do we classify a new point?

Outlook: Sunny
Temp.: Cold
Humidity: Normal
Windy: No

Play: **Yes**

Decision Tree Anatomy

- 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*
    - 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!

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

Smaller Trees

- Can we produce smaller decision trees for functions?
  - Most of the time: Yes
  - 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 more 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?
- Problem: Optimal decision tree learning is NP-complete
  - We can't guarantee that we'll find the optimal tree

- 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 uses greedy algorithms
  - Adjustments 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

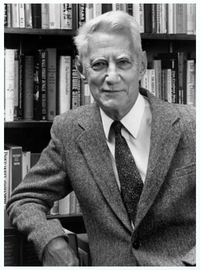
- function BuildDecisionTree(data, labels):
  - if all labels are the same
    - return leaf node for that label
  - else
    - let f be the best feature for splitting
    - left = BuildDecisionTree(data with f=0, labels with f=0)
    - right= BuildDecisionTree(data with f=1, labels with f=1)
    - return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

Selecting Features

- The best feature for splitting
  - The most *informative feature* about the labels
- Relies on *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
- Conditional Entropy
- Information Gain

$$H(X) = -\sum_{x\in X} p(x)\log(p(x))$$

$$H(Y|X) = -\sum_{x\in X} p(x)H(Y|X=x)$$

$$= -\sum_{x\in X} p(x)\sum_{y\in Y} p(y|x)\log p(y|x)$$

$$\mathrm{IG}(Y|X) = H(Y) - H(Y|X)$$

Selecting Features

- Can select the feature which maximizes the information gain
- Measure relative to label distribution
  - X = feature of choice
  - Y = output label
- Equivalent to minimizing the conditional entropy for each leaf

Notes for Decision Trees

- We compare H(Y|X) across different choices for feature X
  - 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(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(Tennis|Outlook=Sunny) = $-2/2 \log_2 2/2 – 0/2 \log_2 0/2 = 0$
- H(Tennis|Outlook=Overcast) = $-0/1 \log_2 0/1 – 1/1 \log_2 1/1 = 0$
- H(Tennis|Outlook=Rainy) = $-0/2 \log_2 0/2 – 2/2 \log_2 2/2 = 0$
- H(Tennis|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(Tennis|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

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

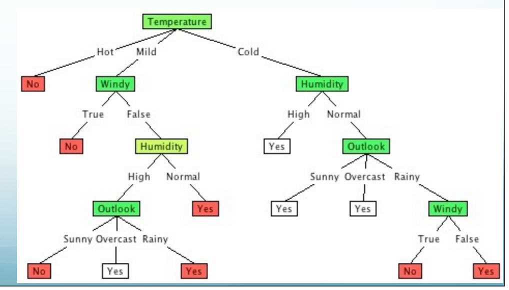
- Consider the following

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

- Consider a similar tree built for tennis data:
  - Non-binary branches
- 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 in the branch
- Stop when max depth reached
- Stop when my classification error is not much more than the average of my children
  - Requires first computing then removing
- 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!

Decision Trees

Fitting a function to data

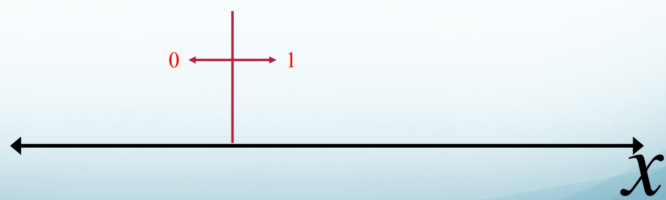
- Fitting: greedy algorithm to find a good tree
  - Extra heuristics to help with over-fitting
  - Optimal decision tree learning: NP-complete
- Function: any boolean function
- Data: Batch: construct a tree using all the data

Extensions

- Non-binary attributes
  - Categorical
  - Continuous (real valued)
    - Handle by thresholding on splitting the range of values
    - Regression trees
- Missing attributes
- Alternatives to information gain
  - Gini index
  - Miss-classification rate
- Non-greedy algorithms?

Continuous Parameters

- How do we handle continuous inputs?
  - We make them categorical!
  - But how?
- Thresholding!



Alternatives to Information Gain

- Misclassification rate
  - Label by most probable class in training data at each leaf
  - Error rate is fraction of test cases with wrong predicted label
  - Simple to evaluate
- Gini Index
  - Expected error rate based on distribution of classes at each leaf

$$\pi_c = \frac{1}{|D|} \sum_{i \in D} I(y_i = c) \qquad G = \sum_{c=1}^{C} \pi_c(1 - \pi_c) = \sum_c \pi_c - \sum_c \pi_c^2 = 1 - \sum_c \pi_c^2$$

Regression Trees

- We can perform regression with trees as well
- At each leaf, predict either:
  - 1) Mean of response variable for data points at leaf
  - 2) Fit a linear model for the data points at the leaf
- #1 is faster. but #2 is more precise
- Cost function is typical least-squares-error
- Constructs piecewise linear function

Pros and Cons of Decision Trees

- Pros
  - Easy to interpret
  - Easily handle mixed continuous and discrete data types
  - Insensitive to monotonic transformations of inputs
  - Perform variable selection automatically
  - Scalable
  - Can handle missing inputs

Pros and Cons of Decision Trees

- Cons
  - Not as accurate as other models, partly due to greedy training
- Unstable
  - Small changes in training data can have large effects on tree structure
  - Errors at the top propagate down due to hierarchical nature