Decision Trees

- Introduction
- Decision Tree Definition
- Using Decision Trees
- Building Decision Trees
  - Recursive Partitioning
  - Information Gain Ratio
- Overfitting
- Pruning
- When are Decision Trees Useful?

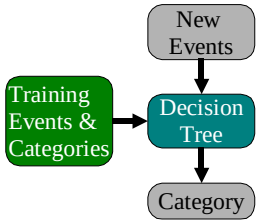
Introduction

Goal: Categorization

- Given an event, predict is category. Examples:
  - Who won a given ball game?
  - How should we file a given email?
  - What word sense was intended for a given occurrence of a word?
- Event = list of features. Examples:
  - Ball game: Which players were on offense?
  - Email: Who sent the email?
  - Disambiguation: What was the preceding word?

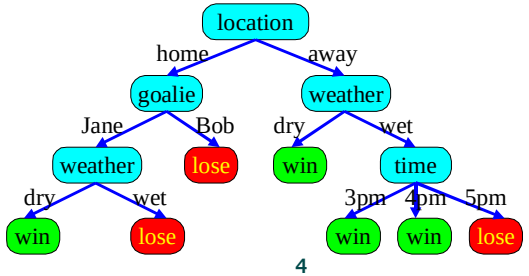
Introduction (2)

- Use a decision tree to predict categories for new events.
- Use training data to build the decision tree.



Decision Trees

- A decision tree is a tree where:
  - Each interior node is labeled with a feature
  - Each arc out of an interior node is labeled with a feature value for the node's feature
  - Each leaf is labeled with a category



Word Sense Disambiguation

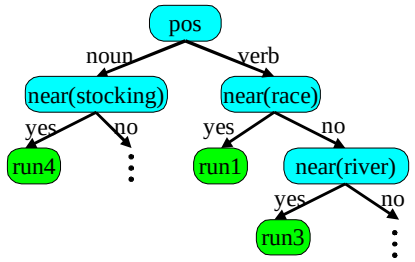
- **Word sense disambiguation (WSD)**
  - **Given an occurance of a word, decide which sense, or meaning, was intended.**
  - **Example: "run"**
    - **run1: move swiftly** (I ran to the store.)
    - **run2: operate** (I run a store.)
    - **run3: flow** (Water runs from the spring.)
    - **run4: length of torn stitches** (Her stockings had a run.)
    - **etc.**

Word Sense Disambiguation (2)

- **Categories**
  - **Use word sense labels (run1, run2, etc.) to name the possible categories.**
- **Features**
  - **Features describe the *context* of the word we want to disambiguate.**
  - **Possible features include:**
    - `near(w)`**: is the given word near an occurrence of word w?**
    - `pos`**: the word's part of speech**
    - `left(w)`**: is the word immediately preceded by the word w?**
    - **etc.**

Word Sense Disambiguation (3)

- **Example decision tree:**



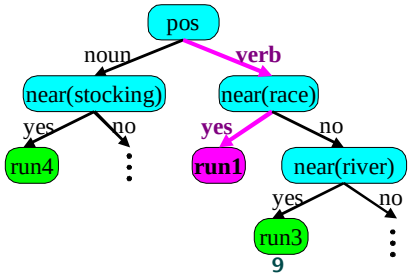
(Note: Decision trees for WSD tend to be quite large)

WSD: Sample Training Data

Features				Word Sense
POS	*near(race)*	*near(river)*	*near(stockings)*	
Noun	No	No	No	run4
Verb	No	No	No	run1
Verb	No	Yes	No	run3
Noun	Yes	Yes	Yes	run4
Verb	No	No	Yes	run1
Verb	Yes	Yes	No	run2
Verb	No	Yes	Yes	run3

Using a Decision Tree

- Given an event (=list of feature values):
  - Start at the root.
  - At each interior node, follow the outgoing arc for the feature value that matches our event.
  - When we reach a leaf node, return its category.



"I saw John run a race by the river."

Creating a Decision Tree

- Use training events & categories to create a new decision tree.
- What properties do we want the decision tree to have?
- It should be consistent with the training data.
  - *Many* decision trees are consistent
- It should be simple.
  - Occam's Razor
    - Simple models tend to generalize better
  - In simpler decision trees, each node is based on more data (on average). ➡ More reliable

Finding a Consistent Decision Tree

- Too many decision trees to try them all.
- Build a decision tree top-down, using "recursive partitioning":

`dtlearn`(training events & categories)

if all of the events have the same category:
- create a leaf with that category.

else:
- pick a feature, and create a node for that feature.
- for each feature value:
  - use `dtlearn` to build a subtree for the events with that feature value.

} "Split" on a feature

Finding a Good Decision Tree

- What features should we split on?
  - We want a *small* decision tree.
  - Pick the feature that gives us the most information about what the category should be.
- 20 Questions
  - I am thinking of a number from 1 to 1,000.
  - You can ask yes/no questions.
  - What is the first question you would ask?
    - On average, some questions give you more information than others:
      - *Is it 752?*
      - *Is it prime?*
      - *Is it between 1 and 500?*
    - Pick a question that provides the most information.

Entropy

- Entropy provides a measure of how much we know about the category.
  - On average, how many more yes/no questions do we need to ask to determine the category?
  - The more we know, the lower the entropy.
- The entropy of a set of events *E* is *H(E)*:

$$H(E) = -\sum_{c \in C} P(c) \log_2 P(c)$$

- Where *P(c)* is the probability that an event in *E* has category *c*.

Information Gain

- How much information does a feature give us about what the category should be?
  - *H(E)* = entropy of event set *E*.
  - *H(E | f)* = expected entropy of event set *E*, once we know the value of feature *f*.
  - *G(E,f) = H(E)−H(E | f)* = the amount of *new* information provided by feature *f*.
- Split on the feature that maximizes information gain.

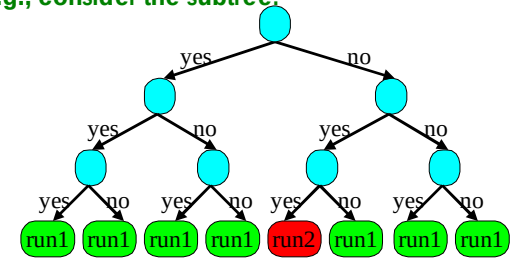
Information Gain Ratio

- But: information gain is biased towards features that have many values.
  - maximum information gain for a feature depends on the number feature values:
    - max information gain for a binary feature is 2.
    - max information gain for a feature w/ 1024 values is 10.
- Use information gain ratio to remove this bias:

$$GR(E,f) = \frac{G(E,f)}{-\sum_{v \in V} P(v) \log_2 P(v)}$$

Overfitting

- Decision tree may encode idiosyncrasies of the training data.
  - E.g., consider the subtree:



  - How much should we trust the "run2" leaf, if it's based on just one training sample?

Overfitting (2)

- Fully consistant decision trees tend to over−generalize.
  - Especially if the decision tree is big.
  - Or if there is not much training data.
- Trade−off full consistancy for compactness:
  - Larger decision trees can be more consistant.
  - Smaller decision trees generalize better.

Pruning

- We can reduce overfitting by reducing the size of the decision tree:
  1. Create the complete decision tree.
  2. Discard any unreliable leafs.
  3. Repeat (2) until the decision tree is compact enough.
- Which leaves are unreliable
  - Leafs with low counts?
- When should we stop
  - No more unreliable leaves?

Held−Out Data

- How can we tell if we're overfitting?
  - Use a subset of the training data to train the model (the "training set").
  - Use the rest of the training data to test for overfitting (the "held−out set").
- Pruning with held−out data:
  1. Create the complete decision tree, from the training set.
  2. Test the performance on the held−out set.
  3. For each leaf:
     1. Test the performance of the decision tree on the held−out set *without that leaf*.
     2. If the performance improves, discard the leaf.
  4. Repeat (3) until no more leafs are discarded.

When are Decision Trees Useful?

- Disadvantages of decision trees:
  - Problems with sparse data & overfitting
    - Pruning helps these problems, but doesn't solve them.
  - Don't (directly) combine information about different features
- Advantages of decision trees:
  - Fast
  - Easy to interpret
    - Helps us understand what's important in new domains.
    - We can explain "why" the decision tree makes decisions.

Reminders

- Assignment 11 due today.
- Reading quiz monday
  - Poole et al Ch. 10–11
- Final exam: Wednesday, April 17th