Foundations of Artificial Intelligence

CS472/3

Lecture #32

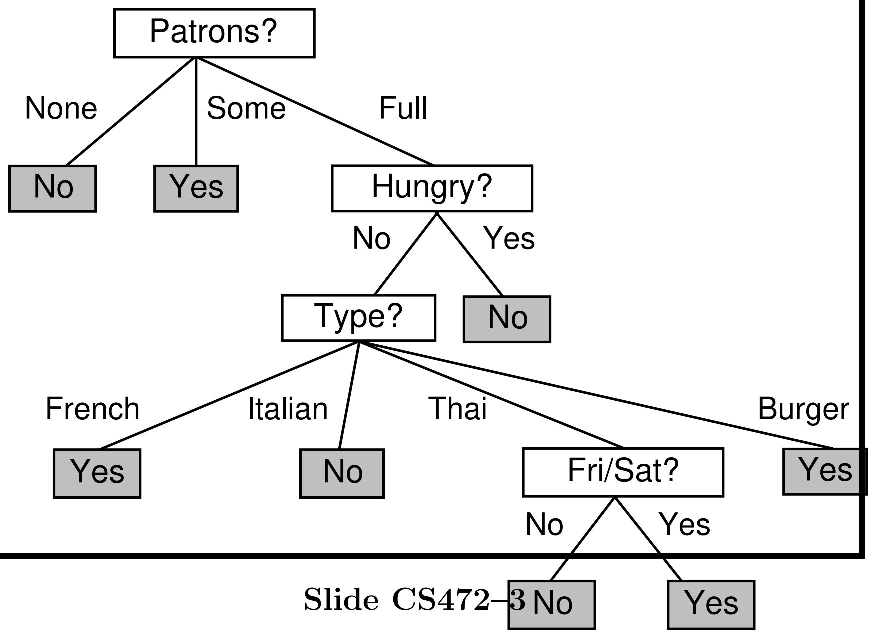
Bart Selman

Slide CS472–1

function DECISION-TREE-LEARNING(*examples, attributes, default*) **returns** a decision tree
  inputs: *examples*, set of examples
    attributes, set of attributes
    default, default value for the goal predicate

  if *examples* is empty **then return** *default*
  else if all *examples* have the same classification **then return** the classification
  else if *attributes* is empty **then return** MAJORITY-VALUE(*examples*)
  else
    best ← CHOOSE-ATTRIBUTE(*attributes*, *examples*)
    tree ← a new decision tree with root test *best*
    for each value v_i of *best* **do**
      $examples_i$ ← {elements of *examples* with $best = v_i$}
      subtree ← DECISION-TREE-LEARNING($examples_i$, $attributes - best$,
        MAJORITY-VALUE(*examples*))
      add a branch to *tree* with label v_i and subtree *subtree*
    end
    return *tree*

Slide CS472–2



Slide CS472–3

Why does learned tree differ from original tree?

Which tree is better?

What if you run out of examples?

Aside: Can trivially find a decision tree for any
set of examples?

So, what makes decision tree learning hard?

Slide CS472–4

Evaluating Performance

1. Collect large set of examples
2. Divide into disjoint sets: **training set** and the **test set**.
3. Learning alg. on **training** set generates hypothesis $h \in H$.
4. Measure percentage **test** set correctly classified by h.
5. Repeat 1 to 4 for different training and test sets / different examples.

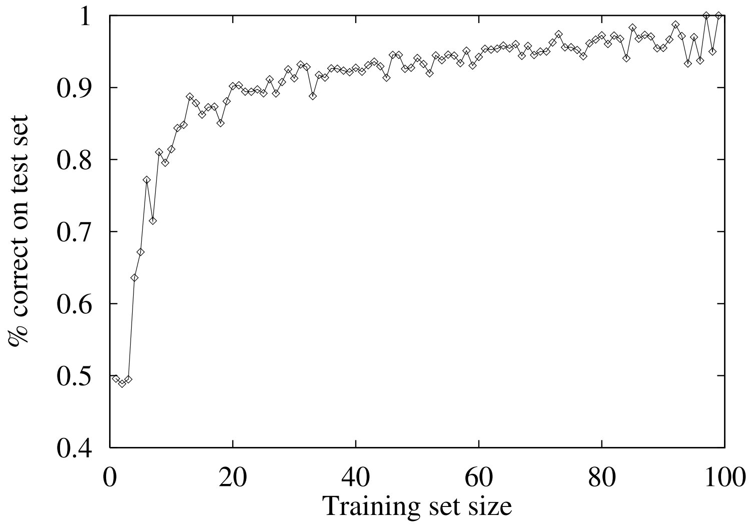
Slide CS472–5

- a form of **cross-validation**
- measures **generalization** to unseen data
- generally considers error on the training set and error on the test set.
- **overfitting** means fits training set "too well" starts to degrade performance on test set. uses e.g. **irrelevant attributes**.

Slide CS472–6

Learning Curve

Slide CS472–7



Slide CS472–8

figure for overfitting

Slide CS472–9

NOTE: often a clear optimal point:
large a enough to capture regularities but
not too large for overfitting.

Slide CS472–10

Overfitting

Consider error in hypothesis h over:

- training data: $error_{train}(h)$
- entire distr. D of data: $error_D(h)$

Hypothesis $h \in H$ **overfits** training data if
there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

but

$$error_D(h) > error_D(h')$$

Dealing with overfitting:

decision tree pruning — prevent splitting on
attributes that are not clearly relevant.
Can test relevance with statistical techiques or
using cross-validation ("watch generalization error")
(See p. 542 / 543, R&N.)
Grow full tree, then post-prune.
Rule post-pruning – later.

How to pick best tree?

- Measure performance over training data.
- Measure performance over separate validation data set.
- MDL (minimal description length): minimize
  $$size(tree) + size(misclassifications(tree))$$

Slide CS472–13

Extensions Decision Tree Learning

- Missing data (unknown values for attributes)
- Multivalued attributes (e.g. $RestaurantName$, high info gain but bad generalization).
- Continuous-values attributes.

Slide CS472–14

Converting Trees to Rules

Examples:

IF $(Patrons = Full) \wedge (WaitEstimate = 0 - 10)$
THEN $(WillWait = Yes)$

IF $(Patrons = Full) \wedge (WaitEstimate > 60)$
THEN $(WillWait = No)$

Fight Overfitting Using Rule Post-Pruning

1. Grow decision tree. Fit as much data as possible. Allow overfitting.
2. Convert tree to equivalent set of rules. One rule for each path from root to leaf.
3. Prune (generalize) each rule independently of others. i.e. delete preconditions that improve its accuracy.
4. Sort final rules into desired sequence for use depending on accuracy.
5. Use ordered sequence for classification.

This is the strategy of the most successful commercial decision tree learning method (C4.5 — Quinlan 1993). Widely used in data mining.

What is advantage of rule representation over the decision tree?

Decision trees are a restricted from of general logical statements.

We can also describe our target function directly in first-order sentences.

Example:
$\forall WillWait(r) \Leftrightarrow Patrons(r, Some)$
$\vee(Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, French))$
$\vee(Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, Thai) \wedge Fri/Sat(r))$
$\vee(Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, Burger))$

This is our **hyposthesis** H_r. In general, we search from among a space of hypotheses:
$H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_n$.

Slide CS472–19

Here's an **example** in logical form:
$Alternate(X1) \wedge \neg Bar(X_1) \wedge \neg Fri/Sat(X_1) \wedge$
$Hungry(X_1) \wedge \ldots \wedge WillWait(X_1)$

We can test if this example is **consistent** with our hyposthesis. If not, we may have to **generalize** or **specialize** our hypothesis:
Current-best-hypothesis search.
See example: p. 548 R&N.

Slide CS472–20