Learning from Observations

Class Text: Chapter 18: Section 1 – 3

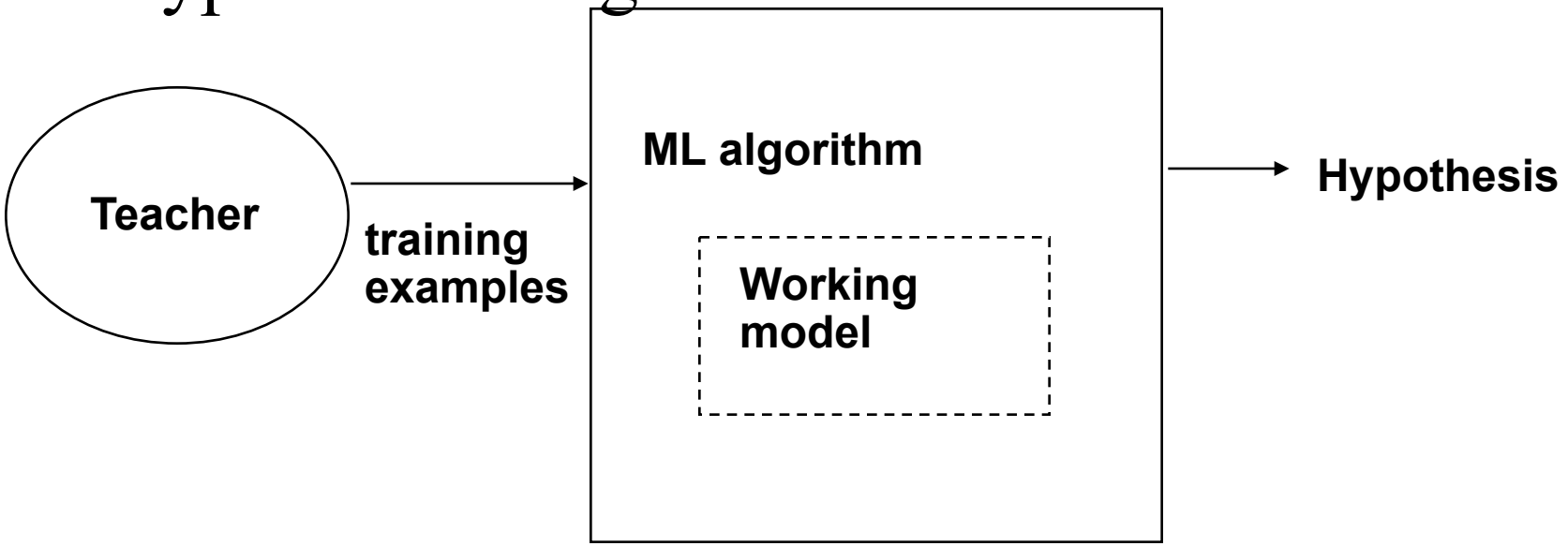
Some material from COSC 4P76 by B.Ombuki

Topics

- Learning agents
- Inductive learning
- Decision tree learning
- A few online examples included

Example of learning

- Typical learning scenario:



What is machine Learning

Machine learning is a subfield of AI concerned with programs that learn from experience

"Learning is constructing or modifying representations of what is being experienced." --Ryszard Michalski

"Learning is making useful changes in our minds." --Marvin Minsky

What does it mean to learn?

- ***Mitchell's definition: A computer program solving task T learns with experience E with respect to performance measure P, if its performance at task T, as measured by P, improves with experience E.***

Learning = Improving with experience at some task
- **Improve over task T,**
- **with respect to performance measure P,**
- **based on experience E.**

Example: Learn to play checkers

T: play checkers (and win)

P: % of games won against opponents

E: play opportunity against itself

examples

- Task: spoken word recognition
  - P:% words correctly classified
  - E: database of spoken words and classification
- Task: predicting share prices
  - P: mean prediction error
  - E: observing previous time courses of share prices
- Task: clustering of animals
  - P: similarity of animals within the clusters
  - E: database of animal species (NO sample clusters)
- Task: playing backgammon
  - P: %wins
  - E: playing practice games against itself

Main types of learning tasks

- Type of experience (feedback)
  - Supervised: correct answers for each example
    - Classification (e.g. spoken words)
    - Regression (e.g. share prices)
  - Unsupervised: correct answers not given
    - (e.g. clustering)
  - Reinforcement: occasional rewards
    - (e.g. backgammon)

State of the Art

- **Speech recognition**
  - Neural Networks and Hidden Markov Models
  - Applies to signal processing
- **Driving an Autonomous Vehicle**
  - 70 mph for 90 miles on public highways
  - Applies to sensor-based control

State of the Art

- **Classification of New Astronomical Structure**
  - Decision tree learning

- **Game playing**
  - Texas Hold'em: PokiPoker
  - Backgammon: TD-Gammon
  - Checkers: Chinook

- **Theoretical results:** characterize the fundamental relationships among the number of training examples, observed the number of hypotheses under considerations and the expected error in the learnt hypotheses

Recall the Learning problem?

Learning = Improving with experience at some task

- **Improve over task T,**
- **with respect to performance measure P,**
- **based on experience E.**

Example: Learn to play checkers

T: play checkers (and win)

P: % of games won against opponents

E: play opportunity against itself

Learning to Play Checkers

T: **play checkers**

P: **Percentage of games won in a world tournament**

- **What experience?**
- **What type of knowledge should be learnt?**
- **How shall it be represented?**
- **What specific algorithm should be applied to learn it?**

Designing a Learning System

- Training Experience
- Target Function
- Function Representation
- Function Approximation
- Final Design

Choosing a training Experience

- Direct or indirect feedback provided to the performance system?
- Teacher or not? Does the learner have control of the sequence of training samples?
- Problem: is the training experience representative of the performance goal?

Choosing the Target Function

- Learn a function:

  ChooseMove: Board -> Move

- Learn an Evaluation function:

  V: Board -> R

- ….

Possible Definition for Target Function V

- if b is is a final board state that is won, then V(b) = 100
- if b is a final board state that is lost, then V(b) = -100
- if b is a final board state that is a draw, then V(b) = 0
- if b is not a final board state, then V(b) = V(b'), where b' is the best final board state that can be achieved from b and playing optimally until the end of the game

This gives correct values, but is not operational

Choosing a Representation for the Target Function

- Table with one entry per board?
- Collection of rules?
- Polynomial function of board features?
- Neural networks?
- …..

Our choice of representation for Learned Function

$V'(b) = w0 + w1x1 + w2x2 + w3x3 + w4x4 + w5x5 + w6x6$

x1: number of black pieces on board b

x2: the number of red pieces on board b

x3: the number of black kings on board b

x4: the number of red kings on board b

x5: the number of black pieces threatened by red

X6: the number of black pieces threatened by black

Obtaining Training examples

- V(b): the true target function
- V’(b): the learned function
- Vtrain(b): the training value

A rule for estimating training values:

Vtrain(b) <- V’(Successor(b))

Choosing Weight Tuning Rule

LMS Weight update rule:

Do repeatedly:

- Select a training example b at random

    1. Compute *error(b):*

        $$error(b) = Vtrain(b) - V'(b)$$

    2. For each board feature xi, update weight wi:

        $$wi \leftarrow wi + c.xi.error(b)$$

c is some small constant , say 0.5, to moderate the learning

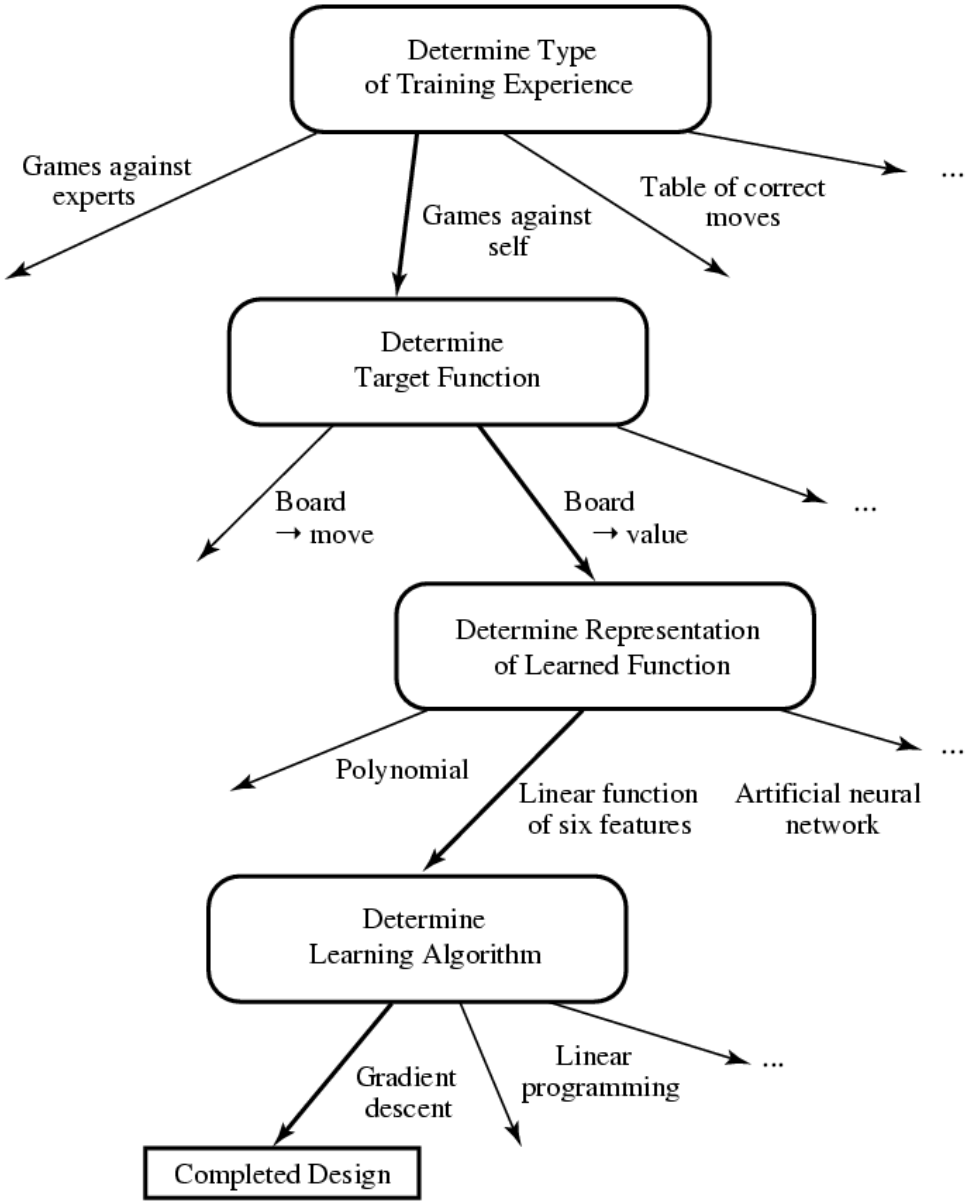
Partial Design of a Checker Learning Program

- Task *T : playing checkers*
- *Performance Measure P :*
  - *percent of games won in the world tournament*
- *Training Experience E : games played against itself*
- *Target Function:*
- *Target Function Representation:*

Would this program have beat the world champion?

Probably not. Why?

- **the linear function is too simple a representation to capture the nuances of the game**
- **but, given a more sophisticated target function, this approach is very successful.**


Determine Type
of Training Experience
Games against
experts
Games against
self
Table of correct
moves
...
Determine
Target Function
Board
→ move
Board
→ value
...
Determine Representation
of Learned Function
Polynomial
Linear function
of six features
Artificial neural
network
...
Determine
Learning Algorithm
Gradient
descent
Linear
programming
...
Completed Design

Some issues in ML

- What algorithms can approximate functions well and in what situations?
- How does the number of training examples influence accuracy?
- How does the complexity of the hypothesis representation impact it?
- What effect does noise in the data have on performance?
- What are the theoretical limits of learn-ability?
- How can prior knowledge help?
- What clues can we use from knowledge of biological learning systems?
- How can systems alter their own representations?

Inductive Learning

Inductive learning: form of learning concepts from examples
- the "teacher" provides examples and the learner is supposed to make conclusions (or generalizations about the examples)
- most researched kind of learning in AI

e.g., from examples: one can learn to diagnose a patient or plant disease, to predict the weather, to improve efficiency in solving symbolic integration problems or to control a dynamic system

Decision Trees

- One of the most popular ML algorithm.
- Form of supervised learning for concept classification.
- Represents a procedure for classifying objects based on their attributes.
- Internal node represents an attribute to test.
- Leaf node: represent a classification.
- To classify an object: start at root node, traverse the branches corresponding to the attribute values of the object.

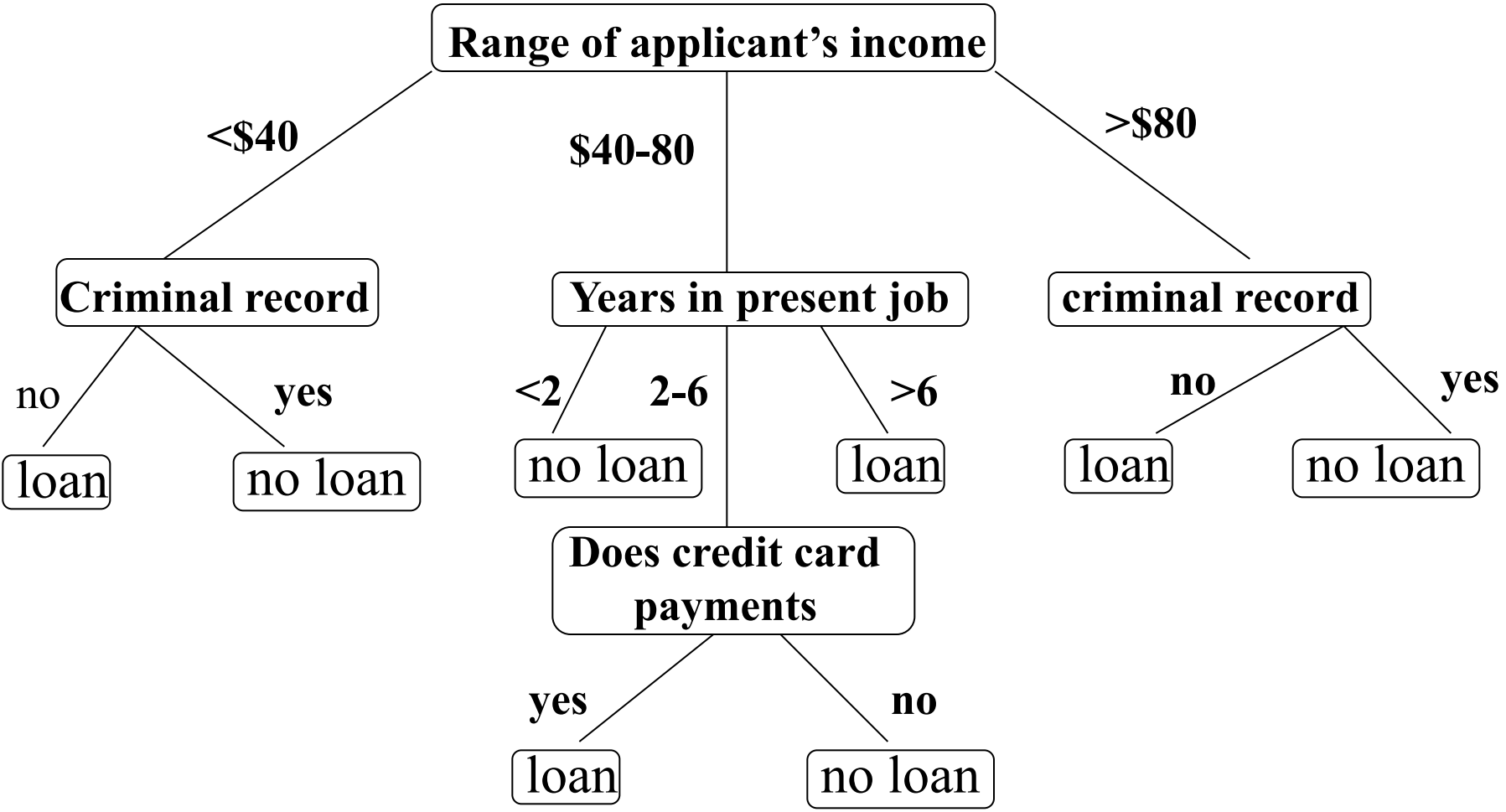
Learning by Decision Trees

Objective: build a decision tree for classifying examples as positive or negative instances of a concept

A decision tree takes as input an instance (a list of features, or situation described by a set of properties) and outputs a YES/NO decision.

Thus decision trees represent Boolean functions, however functions with a larger range of classifications can be represented

Example: Decision Tree



Learning Decision trees

-Bias is towards simple decision trees

- The non-leaf nodes are labeled with attributes (feature)

- each leaf-node is labeled with a classification (of + or -)

- each arc out of a node is labeled with one of the possible value of the attributes at the node where the arc is directed from

Learning decision trees

Suppose you are given a database of classified examples.

Randomly partition database into training & test sets.

Use training examples to construct a decision tree that classifies them correctly.

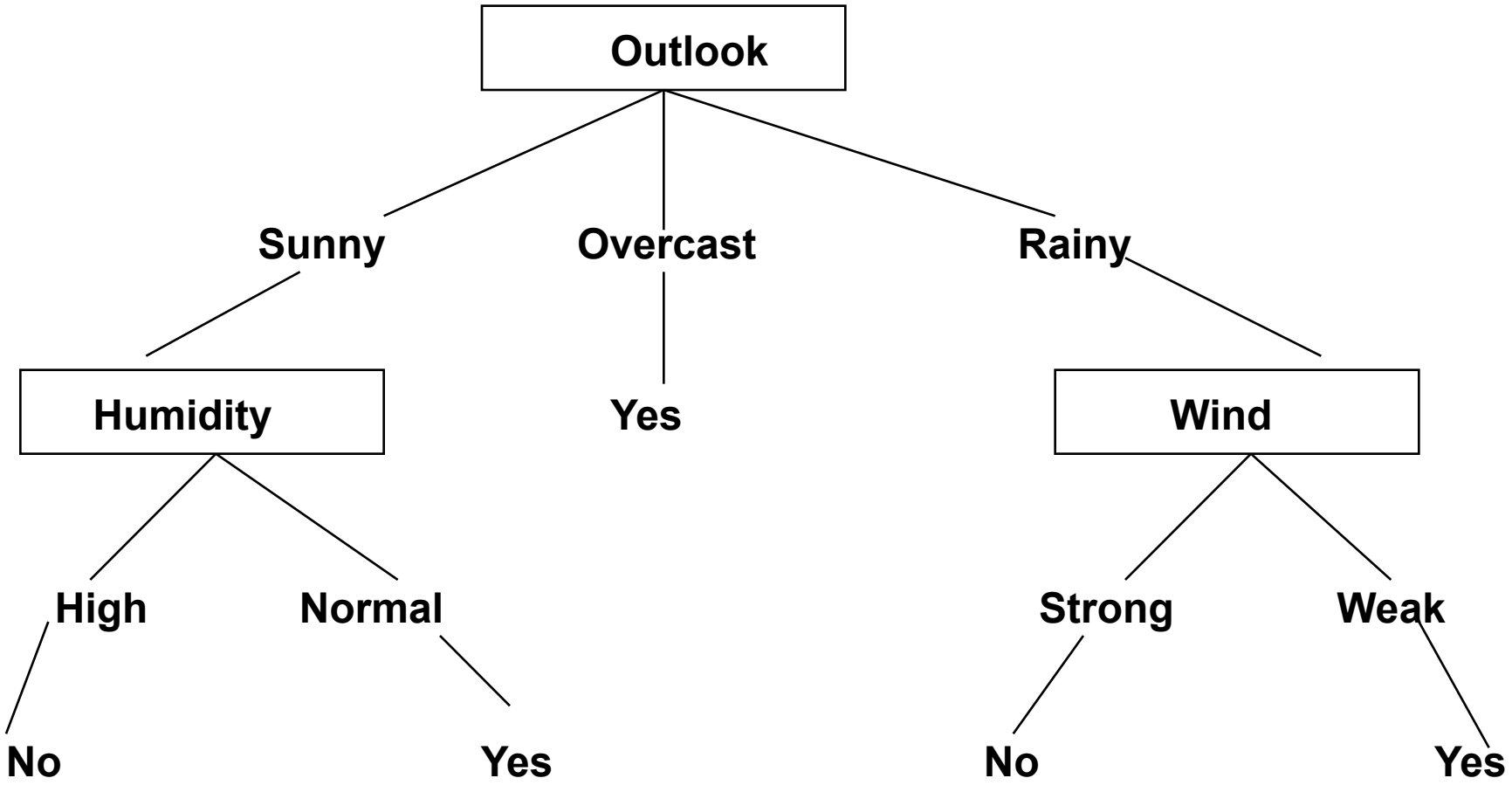
Measure performance on testing examples

Example:“model of restaurant domain R & N pp. 653-659”

Example : A concept learning Task: *PlayTennis*

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes

Decision Tree for *PlayTennis*



Converting tree to rules (contd.)

IF (Outlook = Sunny) ^ (Humidity = High)
THEN *PlayTennis* = No

IF (Outlook = Sunny) ^ (Humidity = Normal)
THEN *PlayTennis* = Yes

IF (Outlook = Overcast)
THEN *PlayTennis* = Yes….

Unknown Attribute Values

- Assign the most common value for the attribute among the training examples that reached the same node in the decision tree.

- Push the example down the decision tree in fractions, probabilistically. The fractions are based on the proportion of examples at the node that have each attribute values.

When to consider decision tree learning

- Instances represented by attribute-value pairs
- Target function has a discrete number of output values
- Disjunctive hypothesis or descriptions maybe required
- Possibly noisy training data (i.e., data may contain errors)

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

Designing a Decision Tree Learning Algorithm

Goal: Construct a Decision tree that agrees (is consistent) with the training set

Trivial Solution: construct a decision tree that has one path to a leaf for every example

problem with trivial solution?

Non-trivial solution: find a concise decision tree that agrees with the training data.

Learning Decision trees

Which tree should be generated from a given data? A decision tree can represent any discrete function of inputs.

We need a bias. E.g., Preference Bias: in which the set of possible hypotheses is ordered: e.g., prefer polynomials of smaller degree, prefer the smallest tree. Fewest nodes? Least depth?

At the extreme, preference bias yields Ockham's Razor.

Learning decision trees

Ockham's Razor

"Therefore the smallest decision tree that is consistent with the samples is the one that is most likely to identify unknown objects correctly."

How does one build a decision tree? Is it practical to search for the smallest decision tree? Finding the smallest decision tree is an NP-Hard problem, so instead of absolute smallest tree construct one that is pretty small. Algorithm called ID3 or C5.0 used , originally developed by Quinlan(1987). ID3 conducts a greedy search through the search space of possible decision trees.

ID3

- ID3 is a well-known decision tree algorithm that uses a top-down greedy search through the hypothesis space
- ID3 was designed to handle large training sets with many attributes
- ID3 tries to generate fairly simple trees, but is not guaranteed to produce the best one
- Two of the most widely used decision tree algorithms, C4.5 and C5.0, are descendants of ID3

ID3

Main question answered by the ID3 algorithm is:
Given a set of training examples, how do we choose the next feature to place in the decision tree being constructed
Possible choices are:

- Random: any attribute selected at random

- Least-Values: the attribute with the smallest number of possible values is selected

- Most-Values: the attribute with the largest number of possible values is selected

- Max-Gain: the attribute that has the largest expected information gain is selected, i.e., trying to select an attribute that will result in the smallest expected size of the sub-trees rooted at its children.

The ID3 algorithm uses the MAX-GAIN method of selecting the best attribute.

Decision tree learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of

```
function DTL(examples, attributes, default) returns a decision tree
    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 MODE(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 ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

Decision Trees on Real Problems

Must consider the following issues

- Multi-class problems
- Alternative splitting criterion
- Noise in data
- Real-valued attributes
- Missing values
- Attributes with cost

Overfitting

We say overfitting has occurred if the learned concept is too specific to the training data. Various reasons can lead to over fitting:

Noise
Lack of enough training data

For example, in one study of 5 learning tasks, overfitting decreased the accuracy of decision trees by 10-25%

Thus overfitting is a real problem

Avoiding Overfitting

- **Prepruning**: Stop Growing the tree when there is not enough data to make reliable decisions
  - prepruning: easier and more intuitive

- **Postpruning:** Grow the full decision tree and then remove nodes for which there is not sufficient evidence
  - postpruning: (generally) works better in practice

Avoiding Overfitting

- **Prepruning**: Stop Growing the tree when there is not enough data to make reliable decisions
  - prepruning: easier and more intuitive

- **Postpruning:** Grow the full decision tree and then remove nodes for which there is not sufficient evidence
  - postpruning: (generally) works better in practice

Evaluation Methods for Pruning

Validation Methods: Reserving part of the training data as validation set. 2 Common approaches are:

- using a single training set and a single validation set
- Cross-validation: the training data is divided into N partitions. Then N experiments are done and each partition is used once as the validation set, and the the other N-1 partitions are used as training set

Statistical Analysis: Use statistical tests to estimate whether pruning/expanding a node is likely to produce an improvement beyond the training data or not.

Advantages of Decision Trees

- Highly expressive
- Easy to generate simple algorithm
- Easy to read small trees, which can be converted to rule set
- Relatively fast to construct; classification is very fast
- Can achieve good performance on many tasks
- A wide variety of problems can be recast as classification problems

Weaknesses of decision trees

- Can be hard to understand
- Not always easy to learn complex concepts
- Methods for handling missing attribute values are somewhat clumsy
- Some problems with continuously-valued attributes or classes may not be easily discredited.