

# Artificial Intelligence Programming

## Decision Trees

Chris Brooks

Department of Computer Science  
University of San Francisco

## Rule Learning

- Previously, we've assumed that background knowledge was given to us by experts.
  - Focused on how to use that knowledge.
- Today, we'll talk about how to acquire that knowledge from observation.
- Focus on learning propositional rules
  - $sunny \wedge warm \rightarrow PlayTennis$
  - $cool \wedge (rain \vee strongWind) \rightarrow \neg PlayTennis$

## Learning

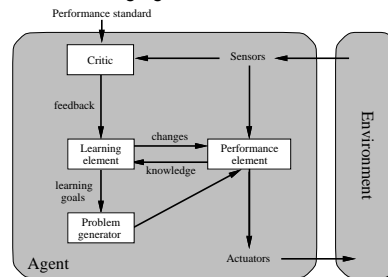
- What does it mean for an agent to learn?

## Learning

- What does it mean for an agent to learn?
- Agent acquires new knowledge
- Agent changes its behavior
- Agent improves its performance measure on a given task

## Learning Agents

- Recall that at the beginning of the semester we talked about *learning agents*



## Learning Agents

- A learning agent has a *performance element* and a *learning element*.
  - The performance element is what an agent uses to decide what to do.
  - This is what we've studied up to now.
- The learning element is what allows the agent to modify the performance element.
  - This might mean adding or changing rules or facts, modifying a heuristic, changing a successor function
  - In order to modify its behavior, an agent needs information telling it how well it is performing.
  - This information is called *feedback*.

## Deduction vs. Induction

- Up to now, we've looked at cases where our agent is given general knowledge and uses this to solve a particular problem.
  - Exactly two people like Homer, Suck always cleans a room, etc.
  - This general-to-specific reasoning is known as *deduction*.
  - Advantage: deduction is sound, assuming your knowledge is correct.

## Deduction vs. Induction

- Sometimes, you may not have general information about a problem.
- Instead, you might have *data* about particular instances of a problem.
- the problem then is to figure out a general rule from specific data.
- This is called *induction* - most learning is an inductive process.
  - Problem: induction is not sound.

## Example

- In the assignment, you'll work with the problem of an agent deciding whether we should play tennis on a given day.
- There are four observable percepts:
  - Outlook (sunny, rainy, overcast)
  - Temperature (hot, mild, cool)
  - Humidity (high, low)
  - Wind (strong, weak)
- We don't have a model, but we do have some data about past decisions.
- Can we induce a general rule for when to play tennis?

## Types of Learning Tasks

- There are essentially three categories of learning tasks, each of which provides different feedback.
- They vary in the amount of information that is available to our learning algorithm.
- Supervised learning.
  - In this case, an external source (often called a teacher) provides the agent with *labeled examples*
  - Agent sees specific actions/cases, along with their classification.
- D2 was Sunny, mild, high humidity and weak wind. We played tennis.

## Types of Learning Tasks

- Unsupervised Learning
  - In this case, there is no teacher to provide examples.
  - The agent typically tries to find a "concept" or pattern in data.
  - Statistical methods such as clustering fall into this category
  - Our agent might be told that day1, day 4 and day 7 are similar and need to determine what characteristics make these days alike.

## Types of Learning Tasks

- Reinforcement Learning
  - This is a particular version of learning in which the agent only receives a *reward* for taking an action.
  - May not know how optimal a reward is.
  - Will not know the "best" action to take
  - Our agent might be presented with a Sunny, Hot, Low humidity, Strong wind day and asked to choose whether to play tennis.
  - It chooses 'yes' and gets a reward of 0.3

## Supervised Learning

- Supervised learning is one of the most common forms of learning.
- Agent is presented with a set of labeled data and must use this data to determine more general rules.
- Examples:
  - List of patients and characteristics: what factors are correlated with cancer?
  - What factors make someone a credit risk?
  - What are the best questions for classifying animals?
  - Whose face is in this picture?
- This is the form of learning we will spend most of our time on.

## Classification

- the particular learning problem we are focusing on is sometimes known as *classification*
  - For a given input, determine which class it belongs to.
- Programs that can perform this task are referred to as *classifiers*

## Defining the Learning Problem

- We can phrase the learning problem as that of estimating a function  $f$  that tells us how to classify a set of inputs.
- An example is a set of inputs  $x$  and the corresponding  $f(x)$  - the class that  $x$  belongs to.
  - $\langle \langle \text{Overcast, Cool, Low, Weak} \rangle, \text{playTennis} \rangle$
- We can define the learning task as follows:
  - Given a collection of examples of  $f$ , find a function  $H$  that approximates  $f$  for our examples.
  - $H$  is called a *hypothesis*.

## Induction

- We would like  $H$  to *generalize*
  - This means that  $H$  will correctly classify unseen examples.
- If the hypothesis can correctly classify all of the training examples, we call it a *consistent* hypothesis.
- Goal: find a consistent hypothesis that also performs well on unseen examples.
- We can think of learning as search through a space of hypotheses.

## Inductive Bias

- Notice that induction is not sound.
- In picking a hypothesis, we make an educated guess.
- The way in which we make this guess is called a *bias*.
- All learning algorithms have a bias; identifying it can help you understand the sorts of errors it will make.
- Examples:
  - Occam's razor
  - Most specific hypothesis.
  - Most general hypothesis.
  - Linear function

## Observing Data

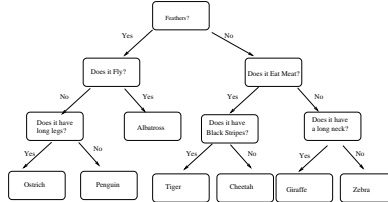
- Agents may have different means of observing examples of a hypothesis.
- A *batch* learning algorithm is presented with a large set of data all at once and selects a single hypothesis.
- An *incremental* learning algorithm receives examples one at a time and continually modifies its hypothesis.
  - Batch is typically more accurate, but incremental may fit better with the agent's environment.
- An *active* learning agent is able to choose examples.
- A *passive* learning agent has examples presented to it by an outside source.
  - Active learning is more powerful, but may not fit with the constraints of the domain.

## Online vs Offline

- An *offline* learning algorithm is able to separate learning from execution.
  - Learning and performance are separate
  - Batch learning is easier, computational complexity is less of a factor.
- An *online* learning algorithm allows an agent to mix learning and execution.
  - Agent takes actions, receives feedback, and updates its performance component.
  - Incremental learning makes more sense, fast algorithms a requirement.
- We will worry about both training time (time needed to construct a hypothesis) and classification time (time needed to classify a new instance).

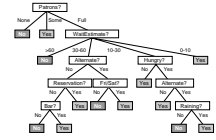
## Learning Decision Trees

- Decision trees are data structures that provide an agent with a means of classifying examples.
- At each node in the tree, an attribute is tested.



## Another Example

- R & N show a decision tree for determining whether to wait at a busy restaurant.
- The problem has the following inputs/attributes:
  - Alternative nearby
  - Has a bar
  - Day of week
  - Hungriness
  - Crowd
  - Price
  - Raining?
  - Reservation
  - Type of restaurant
  - Wait estimate



- Note that not all attributes are used.

## Trees as rules

- A decision tree is just a compiled set of rules.
- We can rewrite the tree as a clause in which the path to the leaf is on the left, and the leaf is on the right.
  - $Wait_{30} \wedge reservation \rightarrow Stay$
  - $Wait_{10} - 30 \wedge Hungry \wedge \neg Alternate \rightarrow Stay$
  - $NoPatrons \rightarrow \neg Stay$
- The tree gives us a more efficient way of determining what to do - we're able to check several rules at once.

## An example training set

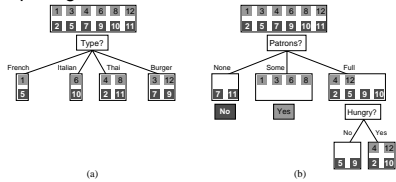
Ex	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	\$	Rain	Res	Type	Est	Wait?
X <sub>1</sub>	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
X <sub>2</sub>	T	F	F	T	Full	\$	F	F	Thai	30-60	F
X <sub>3</sub>	F	T	F	F	Some	\$	F	F	Burger	0-10	T
X <sub>4</sub>	T	F	T	T	Full	\$	F	F	Thai	10-30	T
X <sub>5</sub>	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X <sub>6</sub>	F	T	F	T	Some	\$\$	T	T	Italian	0-10	T
X <sub>7</sub>	F	T	F	F	None	\$	T	F	Burger	0-10	F
X <sub>8</sub>	F	F	F	T	Some	\$\$	T	T	Thai	0-10	T
X <sub>9</sub>	F	T	T	F	Full	\$	T	F	Burger	>60	F
X <sub>10</sub>	T	T	T	T	Full	\$\$\$	F	T	Italian	10-30	F
X <sub>11</sub>	F	F	F	F	None	\$	F	F	Thai	0-10	F
X <sub>12</sub>	T	T	T	T	Full	\$	F	F	Burger	30-60	T

## Inducing a decision tree

- An example is a specific set of attributes, along with a classification (Wait or not)
- Examples where Wait is true are called positive examples
- Examples where Wait is false are called negative examples
- The set of labeled examples is called the *training set*.
- We want to have a tree that:
  - Classifies the training set correctly
  - Accurately predicts unseen examples
  - is as small as possible (Occam's razor)
- What if we construct a tree with one leaf for each example?

## Choosing useful attributes

- Intuitively, we would like to test attributes that 'split' the training set.
- Splitting on restaurant type is not useful - positive and negative examples are still clustered together.
  - Equal number of 'yes' and 'no' answers
- Splitting on crowdedness is more effective.



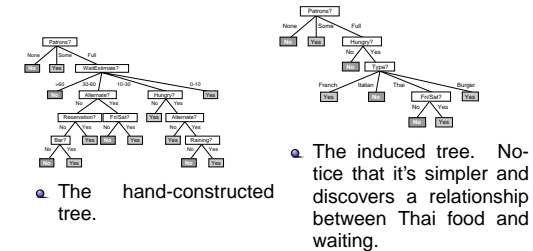
Department of Computer Science — University of San Francisco — p.24/97

## Constructing a decision tree

- We can construct a decision tree recursively:
  - Base cases:** If all examples are positive or negative, we are done.
  - If there are no examples left, then we haven't seen an instance of this classification, so we use the majority classification of the parent.
  - If there are no attributes left to test, then we have instances with the same description, but different classifications.
    - Insufficient description
    - Noisy data, nondeterministic domain
    - Use majority vote
  - Recursive step** Else, pick the best attribute to split on and recursively construct subtrees.

Department of Computer Science — University of San Francisco — p.25/97

## An example tree



Department of Computer Science — University of San Francisco — p.26/97

## Using the tree to make decisions

- We can now use this tree to make decisions, or to *classify* new instances.
- Suppose we have a new instance:  $Alt = F$ ,  $Bar = F$ ,  $Fri = T$ ,  $Hungry = T$ ,  $Patrons = Full$ ,  $Rain = F$ ,  $Reservations = F$ ,  $Type = Burger$ ,  $EstimatedTime = 10 - 30$ .
- We traverse the tree from the root, following the appropriate 'Full', 'Hungry', and 'Type' branches.
- According to the tree, we should wait.

Department of Computer Science — University of San Francisco — p.27/97

## Choosing an Attribute

- The key to constructing a compact and efficient decision tree is to effectively choose attributes to test.
- Intuition: we want to choose tests that will separate our data set into positive and negative examples.
- We want to measure the amount of *information* provided by a test.
- This is a mathematical concept that characterizes the number of bits needed to answer a question or provide a fact.

Department of Computer Science — University of San Francisco — p.28/97

## Information

- Example:
  - In the vacuum world, rooms can be either clean or dirty. This requires one bit to represent.
  - What if a room could take on four states? Eight?
  - What if a room could only be in one state? How many bits would we need to represent this?

Department of Computer Science — University of San Francisco — p.29/97

## Information Theory

- More formally, let's say there are  $n$  possible answers  $v_1, v_2, \dots, v_n$  to a question, and each answer has probability  $P(v_n)$  of occurring.
- The *information content* of the answer to the question is:  

$$I = \sum_{i=1}^n -P(v_i) \log_2 P(v_i)$$
- For a coin, this is:  $-\frac{1}{2} \log_2 \frac{1}{2} + -\frac{1}{2} \log_2 \frac{1}{2} = 1$ .
- Questions with one highly likely answer will have low information content. (if the coin comes up heads 99/100 of the time,  $I = 0.08$ )
- Information content is also referred to as entropy.
  - This is often used in compression and data transfer algorithms

## Using Information Theory

- For decision trees, we want to know how valuable each possible test is, or how much information it yields.
- We can estimate the probabilities of possible answers from the training set.
- Usually, a single test will not be enough to completely separate positive and negative examples.
- Instead, we need to think about how much better we'll be after asking a question.
- This is called the *information gain*.

## Information Gain

- If a training set has  $p$  positive examples and  $n$  negative examples, its entropy is:  

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$
- We want to find the attribute that will come the closest to separating the positive and negative examples.
- We begin by computing the remainder - this is the information still in the data after we test attribute  $A$ .
- Say attribute  $A$  can take on  $v$  possible values. This test will create  $v$  new subsets of data, labeled  $E_1, E_2, \dots, E_v$ .
- The remainder is the sum of the information in each of these subsets.
- $Remainder(A) = \sum_{i=1}^v \frac{p_i+n_i}{p+n} * I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$

## Information Gain

- Information gain can then be quantified as the difference between the original information (before the test) and the new information (after the test).
- $Gain(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - Remainder(A)$
- Heuristic: Always choose the attribute with the largest information gain.
  - Question: What kind of search is this?

## Decision tree pseudocode

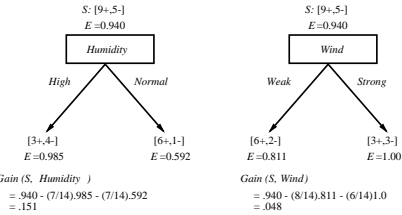
```
def makeTree(dataset) :
    if all data are in the same class :
        return a single Node with that classification
    if there are no attributes left to test:
        return a single Node with majority classification
    else :
        select the attribute that produces the largest information gain
        split the dataset according to the values of this attribute to create v smaller datasets.
        create a new Node - each child will be created by calling makeTree with one on the v subsets.
```

## Example

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
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

## Example

Which attribute is the best classifier?



## Noise

- If there are two examples that have the same attributes but different values, a decision tree will be unable to classify them separately.
- We say that this data is *noisy*.
- Solution: Use majority rule at the parent node.

## Overfitting

- A common problem in learning algorithms occurs when there are random or anomalous patterns in the data.
  - For example, in the tennis problem, by chance it might turn out that we always play on Tuesdays.
- The phenomenon of learning quirks in the data is called *overfitting*.
- In decision trees, overfitting is dealt with through pruning.
- Once the tree is generated, we evaluate the significance of each node.

## Pruning

- Assume that the test provides no information. (null hypothesis)
- Does the data in the children look significantly different from this assumption?
- Use a chi-square test.
- If not, the node is removed and examples moved up to the parent.

## Continuous-valued inputs

- Decision trees can also be extended to work with integer and continuous-valued inputs.
- Discretize the range into, for example,  $< 70$  and  $> 70$ .
- Challenge: What value yields the highest information gain?
- Use a hill-climbing search to find this value.
- In the homework, you can just select a size for the range. (I like bins of size 10)

## Features of Decision Trees

- Work well with symbolic data
- Use information gain to determine what questions are most effective at classifying.
  - Greedy search
- Produce a human-understandable hypothesis
- Fixes needed to deal with noise or missing data
- Can represent all Boolean functions

## Uses of Decision Trees

- Management
- Automated help systems
- Microsoft's online help
- Data mining