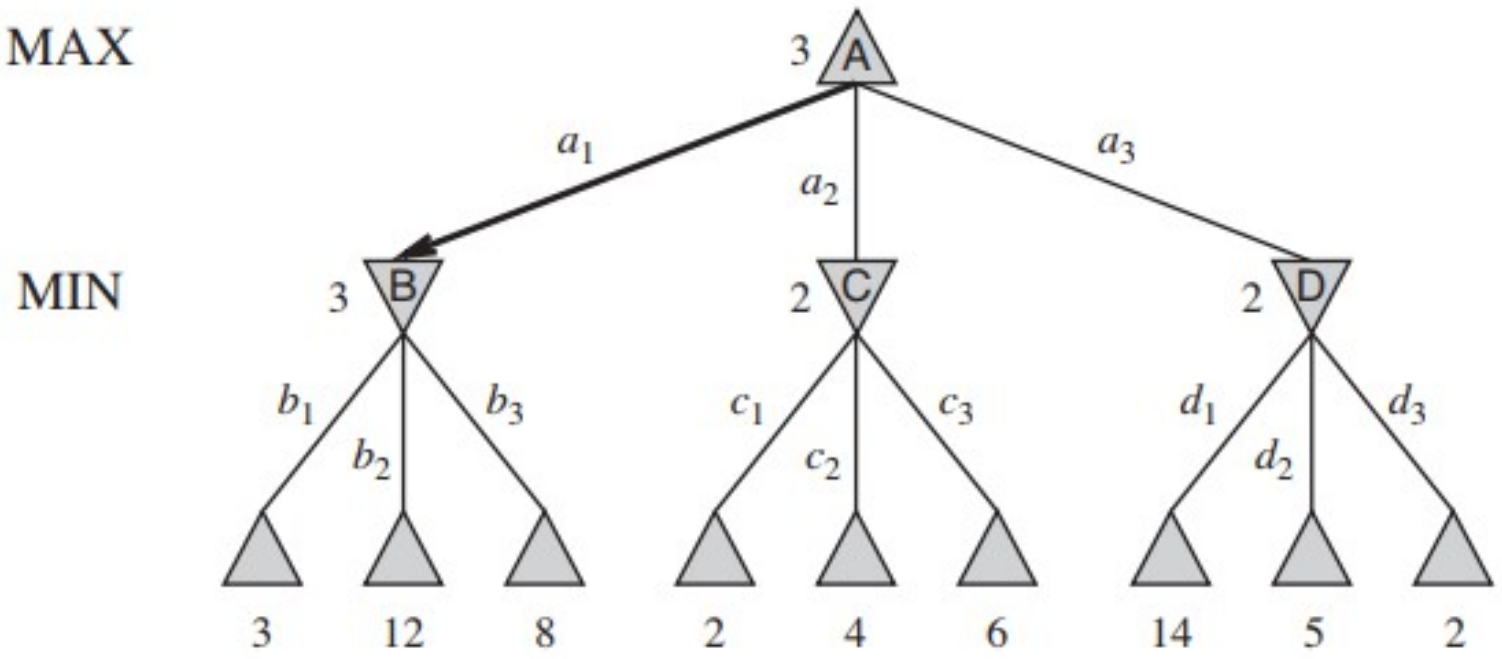
Lecture 7:

Decision Trees and Inductive Learning

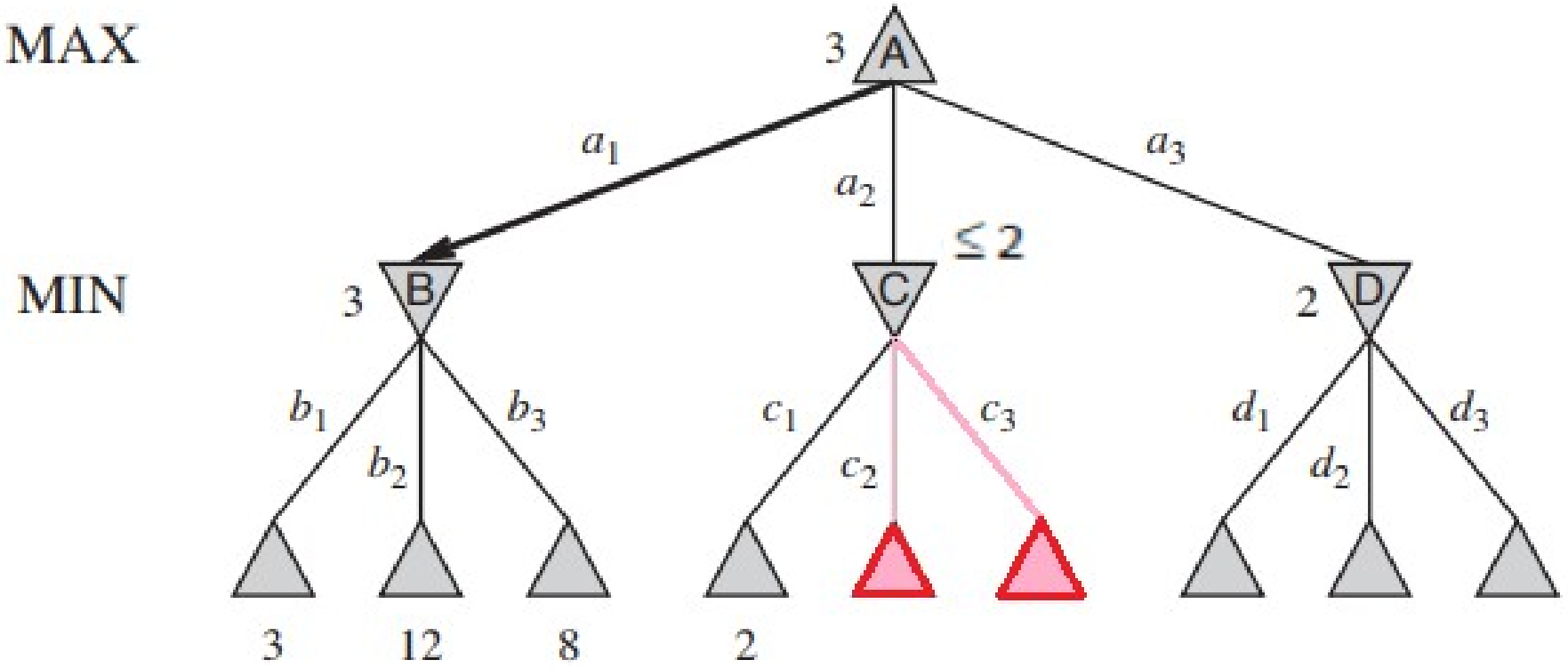
11 February 2020

Minimax search example



- Look ahead *d* moves
- Compute h of all leaves
- Inductively compute values for internal nodes from values of children
  - Max Step (agent's turn): Set values to be max value of children
  - Min Step (opponent's turn): Set value to min value of children
- Select child of root that has highest value
  - This is the action the agent should make

Alpha/beta pruning



- Expand noes in depth first ordering
- Expand action c_1 and get 2
- What can you say about about state C and action a_2?
  - At state C opponent will be able to get to a state with a value ≤ 2
  - Already know that a_1 gives you a value of 3
  - Never want to take action a_2
- Don't even need to consider other children of C

Decision Trees

- Nodes are states
  - At each state you need to make a decision specific to that state
  - Edge for each choice
  - Edge leads to state obtained by selecting choice
- Effectively, actions are choices
  - Different from other examples because actions are different for each state
- Example - navigating a road map
  - At intersection of Main and Oak
    - Turn right on Oak
    - Turn Left on Oak
    - Stay on Main Street
  - Options available different for different intersections

Building Decision Trees: Example

- Problem
  - Given a list of choices construct a decision tree
  - Given a list of restaurants
    - Construct a decision tree
      - Decide what type of restaurant to go to

	Type	Price	Crowded	Time
McDonald's	Fast Food	$	Yes	< 10 min
Burger King	Fast Food	$	Yes	< 10 min
Chipotle	Mexican	$$	Yes	< 10 min
Jennie's	Diner	$$$	Yes	< 10 min
Fred's Fried Fish	Seafood	$	No	< 10 min
Captain John's	Seafood	$$$	No	20 min
Henri's	French	$$$	No	30 min
Pinocchio's	Italian	$$	Yes	20 min
Taste of Italy	Italian	$$	No	20 min

What is a good attribute?

- We want to grow a simple tree

  → a good attribute prefers attributes that split the data so that each successor node is as *pure* as posssible

  - i.e., the distribution of examples in each node is so that it mostly contains examples of a single class

- In other words:
  - We want a measure that prefers attributes that have a high degree of „order“:
    - Maximum order: All examples are of the same class
    - Minimum order: All classes are equally likely

  → Entropy is a measure for (un-)orderedness

  - Another interpretation:
    - Entropy is the amount of information that is contained
    - all examples of the same class → no information

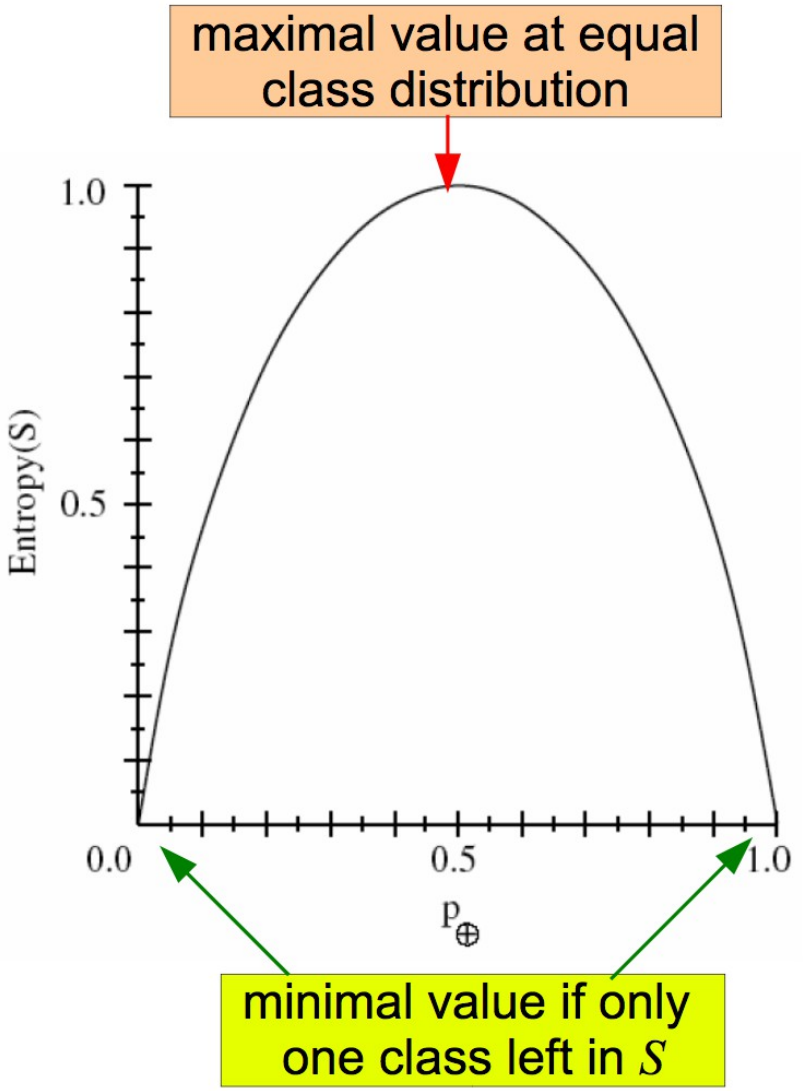
Entropy for Binary Classification

- S is a set of examples
- $p_{\oplus}$ is the proportion of examples in class $\oplus$
- $p_{\ominus} = 1 - p_{\oplus}$ is the proportion of examples in class $\ominus$

Entropy:

$$E(S) = -p_{\oplus} \cdot \log_2 p_{\oplus} - p_{\ominus} \cdot \log_2 p_{\ominus}$$

- Interpretation:
  - amount of unorderedness in the class distribution of S



Entropy

- For a complex (non-binary) set Entropy is given by the following

$$E(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Average Entropy of a Split

- **Problem:**
  - Entropy only computes the quality of a single (sub-)set of examples
    - corresponds to a single value
  - How can we compute the quality of the entire split?
    - corresponds to an entire attribute
- **Solution:**
  - Compute the weighted average over all sets resulting from the split
    - weighted by their size

$$I(S,A)=\sum_i \frac{|S_i|}{|S|}\cdot E(S_i)$$

- **Example:**
  - Average entropy for attribute *Outlook*:

$$I(\text{Outlook})=\tfrac{5}{14}\cdot 0.971+\tfrac{4}{14}\cdot 0+\tfrac{5}{14}\cdot 0.971=0.693$$

Information Gain

- When an attribute A splits the set S into subsets S_i
  - we compute the average entropy
  - and compare the sum to the entropy of the original set S

Information Gain for Attribute A

$$Gain(S, A) = E(S) - I(S, A) = E(S) - \sum_i \frac{|S_i|}{|S|} \cdot E(S_i)$$

- The attribute that maximizes the difference is selected
  - i.e., the attribute that reduces the unorderedness most!

- **Note**:
  - maximizing information gain is equivalent to minimizing average entropy, because $E(S)$ is constant for all attributes A

Building Decision Trees

- Greedy approach
  - Recursively select trait that maximizes information gain.

Build_Decision_Tree(S, Attributes)

- If S is monotone (all same type) or Attributes = {}
  - return
- //Find best attribute
- IG_{best}=-1
- for all attributes a
  - *S'* = set of subsets from dividing S by a
    - if IG(*S'*) > IG_{best}
      - IG_{best} = IG(*S*)
      - a_{best} = a
- *S'* = set of subsets from dividing S by a
- For all S' in
  - Build_Decision_Tree(S', Attributes \ a_{best})

Building Decision Trees: Example

- Recursively subdivide using attribute that maximizes information gain

$$E(S) - \sum_i \frac{|S_i|}{|S|} \cdot E(S_i) \quad \text{where} \quad \mathrm{E}(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

- Intuitively, which division looks the best?

	Type	Price	Crowded	Time
McDonald's	Fast Food	$	Yes	< 10 min
Burger King	Fast Food	$	Yes	< 10 min
Chipotle	Mexican	$$	Yes	< 10 min
Jennie's	Diner	$$$	Yes	< 10 min
Fred's Fried Fish	Seafood	$	No	< 10 min
Captain John's	Seafood	$$$	No	20 min
Henri's	French	$$$	No	30 min
Pinocchio's	Italian	$$	Yes	20 min
Taste of Italy	Italian	$$	No	20 min

Building Decision Trees: Example

$$E(S) - \sum_i \frac{|S_i|}{|S|} \cdot E(S_i)$$ where $$E(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

- After divide by price

	Type	Crowded	Time
McDonald's	Fast Food	Yes	< 10 min
Burger King	Fast Food	Yes	< 10 min
Fred's Fried Fish	Seafood	No	< 10 min
Chipotle	Mexican	Yes	< 10 min
Pinocchio's	Italian	Yes	20 min
Taste of Italy	Italian	No	20 min
Jennie's	Diner	Yes	< 10 min
Captain John's	Seafood	No	20 min
Henri's	French	No	30 min