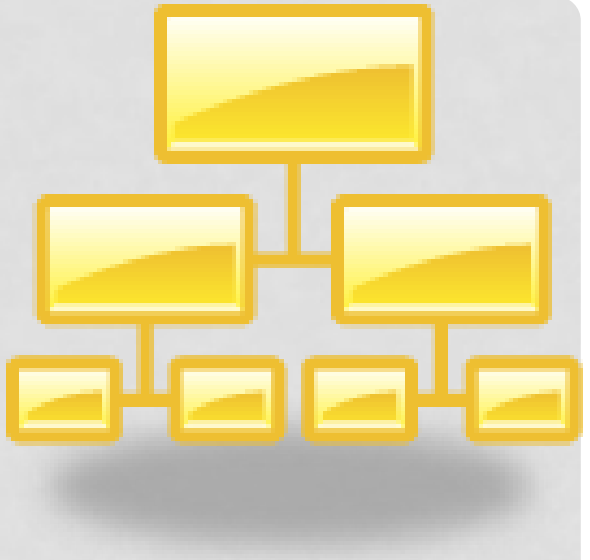


DECISION TREES

REFERENCES:

- "ARTIFICIAL INTELLIGENCE: A MODERN APPROACH, 3RD ED" (PEARSON) 18.3-18.4
- HTTP://ONLAMP.COM/PUB/A/PYTHON/2006/02/09/AI_DECISION_TREES.HTML
- HTTP://CHEM-ENG.UTORONTO.CA/~DATAMINING/DMC/DECISION_TREE_OVERFITTING.HTM

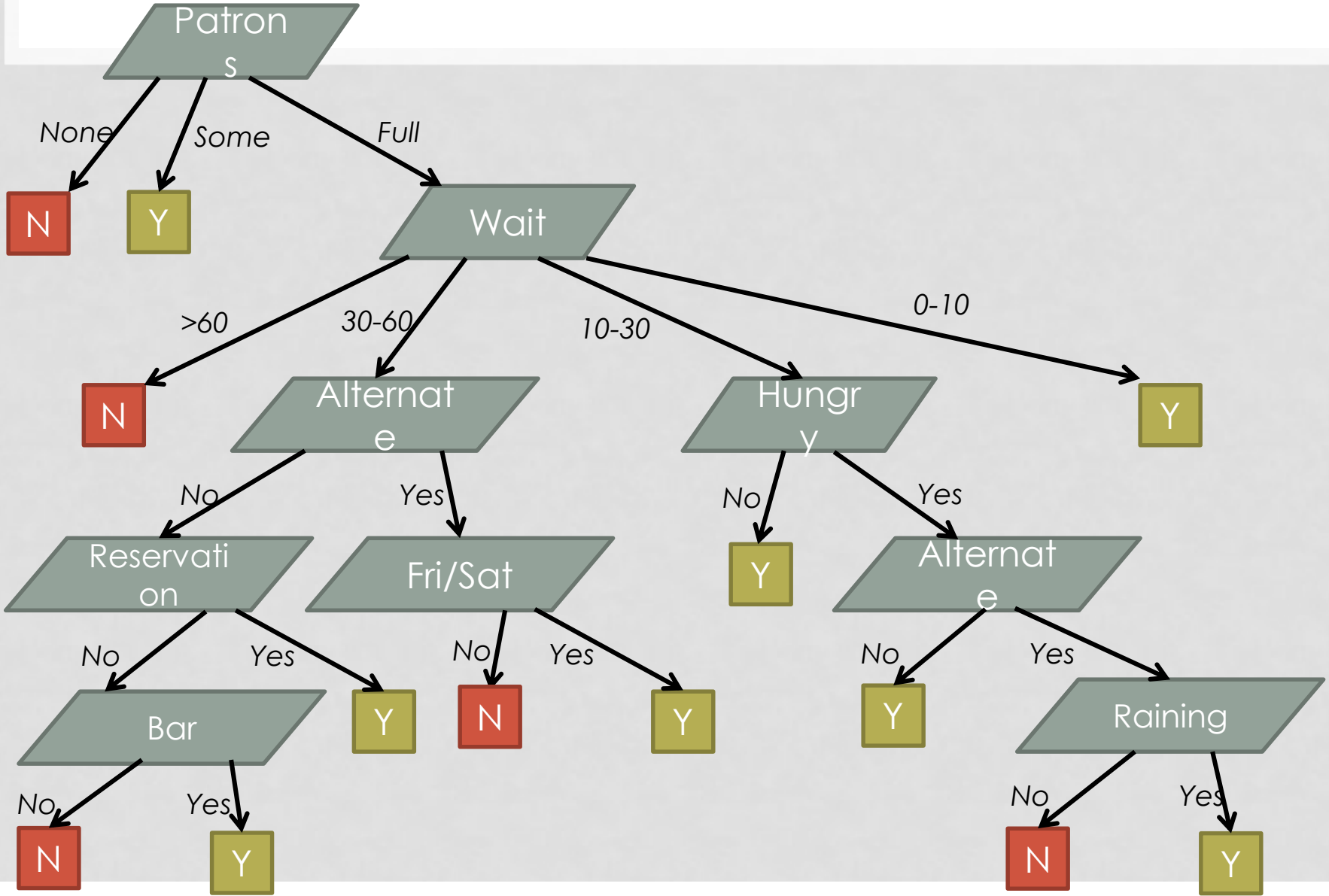
WHAT ARE THEY?

- A "flowchart" of logic
- Example

THE CLASSIC EXAMPLE

- **Goal**: Decide if we'll wait for a table at a restaurant
- Factors:
  - *Alternate*: Is there another restaurant nearby?
  - *Bar*: Does the restaurant have a bar?
  - *Fri / Sat*: Is it a Friday or Saturday?
  - *Hungry*: Are we hungry?
  - *Patrons*: How many people {None, Some, Full}
  - *Price*: Price Range {$, $$, $$$}
  - *Raining*: Is it raining?
  - *Reservation*: Do we have a reservation?
  - *Type*: {French, Italian, Thai, Burger}
  - *Wait*: {0-10, 10-30, 30-60, >60}

POSSIBLE DECISION TREE



ANALYSIS

- Pluses:
  - Easy to traverse
  - Naturally expressed as if/else's
- Negatives:
  - how do we build an **optimal** tree?
    - What is an optimal tree?

SAMPLE DATASET

#	0Alt	1Bar	2Fri	3Hun	4Pat	5Pr	6Ran	7Res	8Type	9Wait	xWait?
0	Y	N	N	Y	S	$$$	N	Y	Fr	0-10	Y
1	Y	N	N	Y	F	$	N	N	Th	30-60	N
2	N	Y	N	N	S	$	N	N	Bu	0-10	Y
3	Y	N	Y	Y	F	$	Y	N	Th	10-30	Y
4	Y	N	Y	N	F	$$$	N	Y	Fr	>60	N
5	N	Y	N	Y	S	$$	Y	Y	It	0-10	Y
6	N	Y	N	N	N	$	Y	N	Bu	0-10	N
7	N	N	N	Y	S	$$	Y	Y	Th	0-10	Y
8	N	Y	Y	N	F	$	Y	N	Bu	>60	N
9	Y	Y	Y	Y	F	$$$	N	Y	It	10-30	N
10	N	N	N	N	N	$	N	N	Th	0-10	N
11	Y	Y	Y	Y	F	$	N	N	Bu	30-60	Y

SAMPLE INPUT, CONT

- We can also think of these as "training data"
  - For a decision tree we want to model
  - In this context, the input:
    - is that of "Experts"
    - exemplifies the thinking you want to encode
    - is raw data we want to *mine*
    - …
- Note:
  - **Doesn't** contain all possibilities
  - There might be **noise**

PART I (CONSTRUCTING A TREE)

BUILDING A TREE

- So how do we build a decision tree from input?
- A lot of possible trees:
  - $O(m^n)$
    - m is the # of choices in each column
    - n is the number of columns
  - Some are good, some are bad:
    - good == shallowest
    - bad == deepest
  - Intractable to find the best
    - Using a greedy algorithm, we can find a pretty-good one…

ID3 ALGORITHM

- By Ross Quinlan (RuleQuest Research)
- A recursive *greedy algorithm*
- Basic idea:
  - Choose the *best* attribute, i # More on this later.
  - Create a tree with n children
    - n is the number of values for attribute i
  - Divide the training set into n sub-sets
    - Where all items in a subset have the same value for attribute i.
    - If all items in the subset have the same output value, make this a leaf node.
    - If not, recursively create a new sub-tree
      - Only use those training examples in this subset
      - Don't consider attribute i any more.

"BEST" ATTRIBUTE

- **Entropy** [on board]

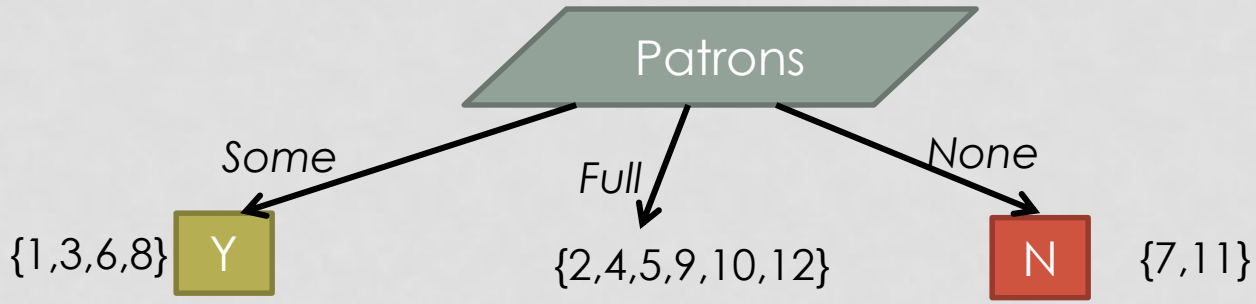
$$H(T) = -\sum_{k=1}^{q} P(k) * log_2\big(P(k)\big)$$

INFORMATION GAIN

- The reduction in entropy
- In the ID3 algorithm:
- $IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$
  - A is the attribute we're considering
  - S is the set of training cases we're considering
  - T is the set of all subsets of S obtained by splitting on attribute A
  - p(t) is the #elements in t divided by sizeof(T)
  - H(t) is the entropy of the subset of cases t.

ORIGINAL EXAMPLE, CONT.

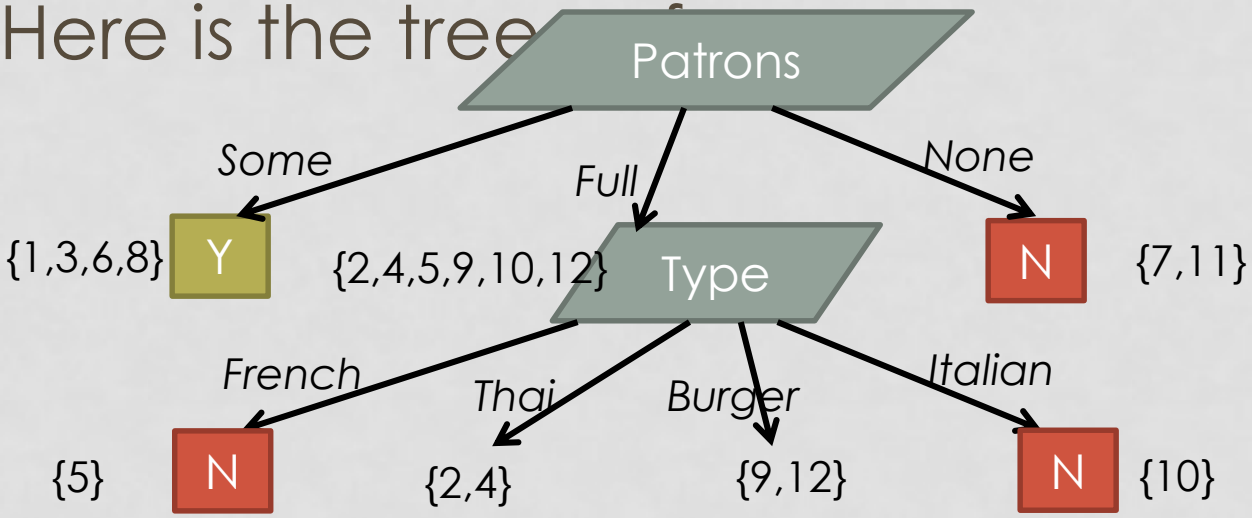
- Pat is much better (0.541 gain vs. 0.21 gain)
- Here is the tree so far:



- Now we need a subtree to handle the case where Patrons==Full
  - Note: The training set is smaller now (6 vs. 12)

ORIGINAL EXAMPLE, CONT.

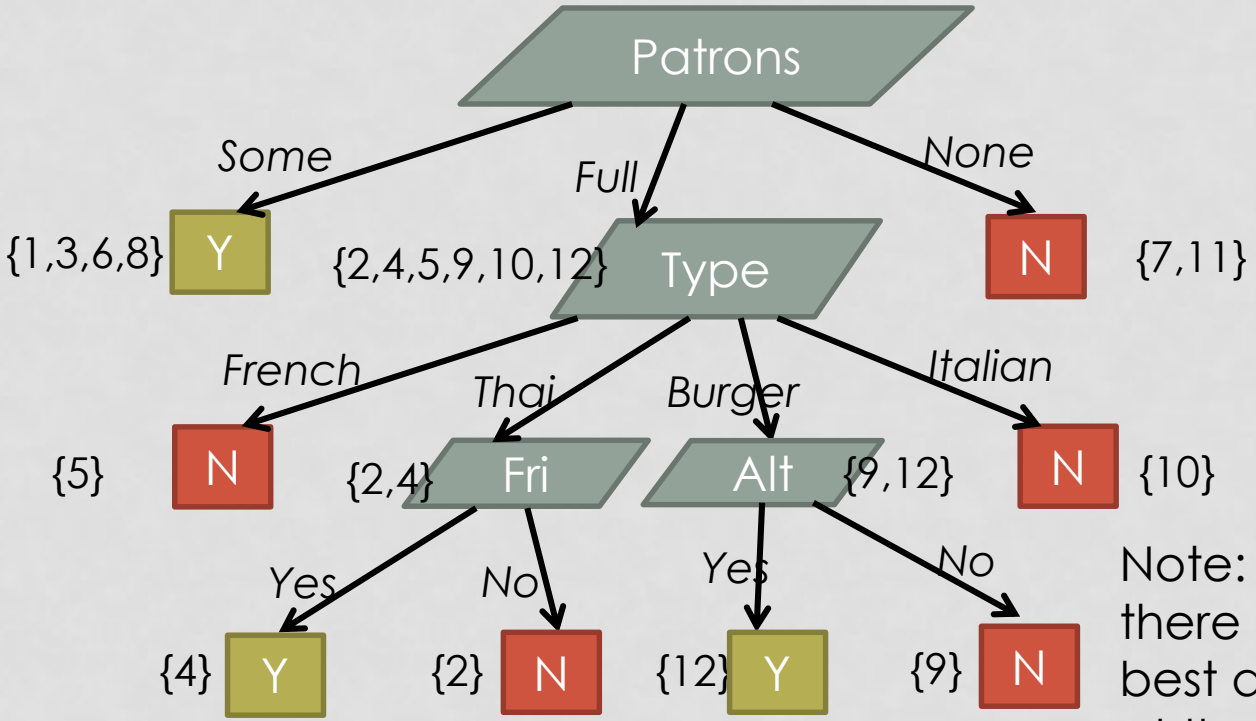
- Type is better (0.251 gain vs. 0.109 gain)
  - Hungry, Price, Reservation, Est would give you same gain.
- Here is the tree so far:



- Recursively make two more sub-trees…

ORIGINAL EXAMPLE, CONT.

- Here's one possibility (skipping the details):



Note: on this data-set, there are a few ties for best attribute. Just look at the depth – it should be close to mine.

SUBSETS WITH NO MORE ATTRIBUTES

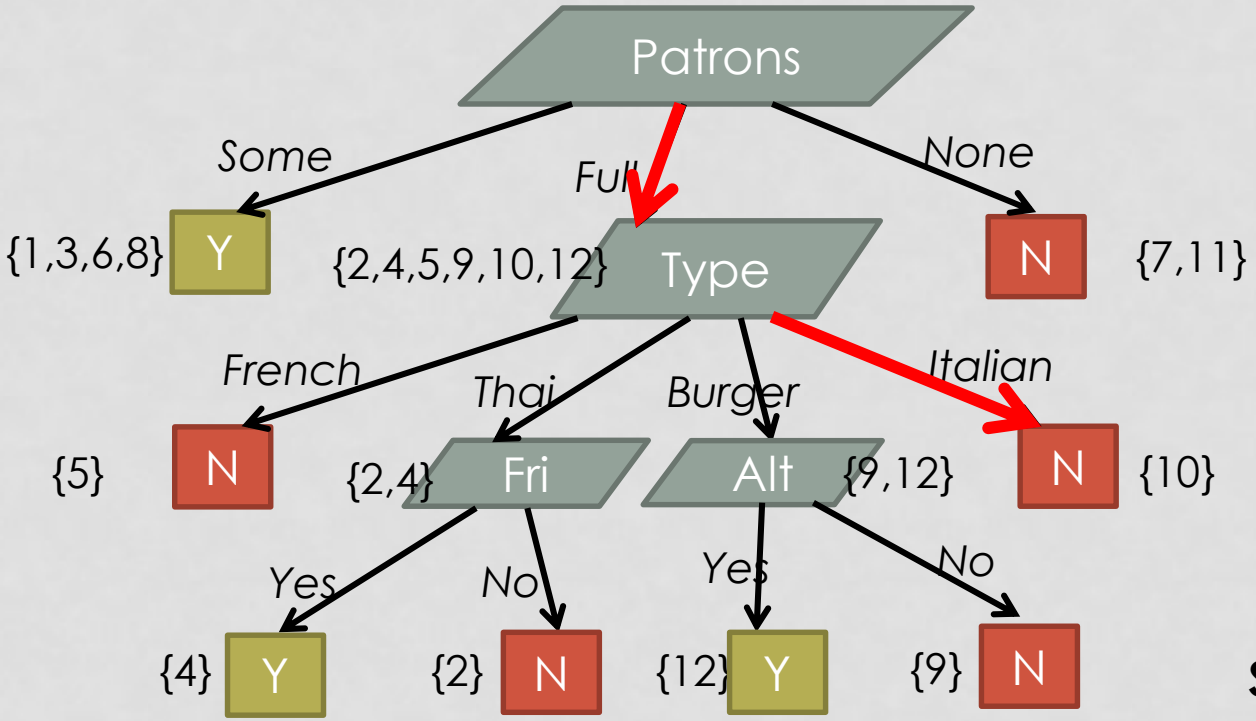
USING A DECISION TREE

- This algorithm will *perfectly* match all training cases.
- The *hope* is that this will generalize to novel cases.
- Let's take a new case (not found in training)
  - Alt="No", Bar="Yes", Fri="No", Pat="Full"
  - Hungry="Yes", Price="$$", Rain=Yes
  - Reservation="Yes", Type="Italian", Est="30-60"
- Will we wait?

ORIGINAL EXAMPLE, CONT.

- Here's the decision process:

- Alt="No"
- Bar="Yes"
- Fri="No"
- Pat="Full"
- Hungry="Yes"
- Price="$$"
- Rain=Yes
- Reservation="Yes"
- Type="Italian"
- Est="30-60"

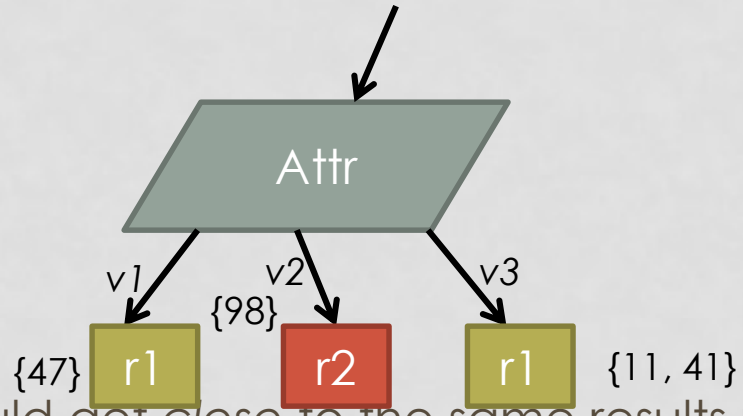


So…No, we won't wai

PART II (PRUNING)

PRUNING

- Sometimes an exact fit is not necessary
  - The tree is too big (deep)
  - The tree isn't generalizing well to new cases (over-fitting)
  - We don't have a lot of training cases:



    - We would get *close* to the same results removing the attr node, and labeling it as a leaf (r1)
- [Ask Jason (after you've got a good start) if you're interested]