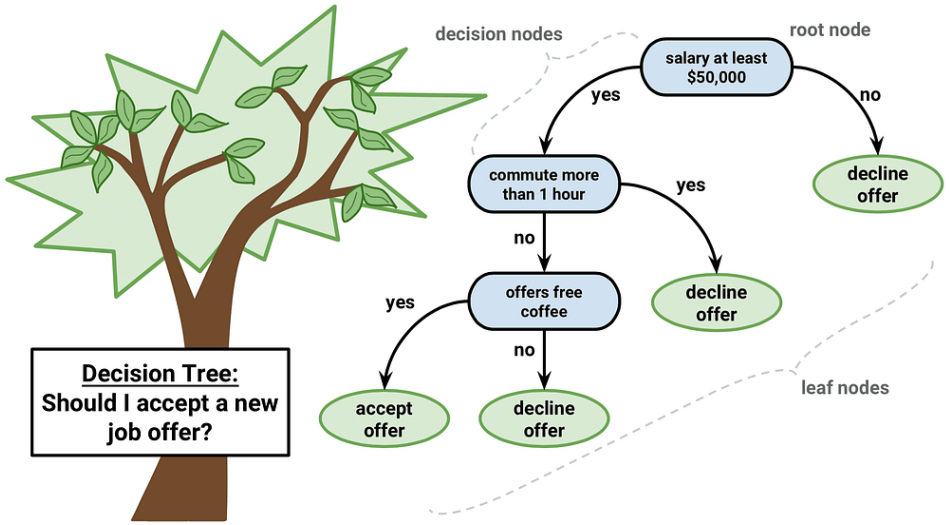
On Decision Trees

Alexander G. Ororbia II

Introduction to Machine Learning

CSCI-335

4/8/2026



The Decision Tree Hypothesis

Decision Tree Hypotheses

Propose a function representing the mapping for *(input, output)* pairs in the training data.

- Classification: (attributes, class), e.g. 'restaurant'
- Regression: (attributes, value), e.g. $ prediction

Geometry of Decision Boundaries

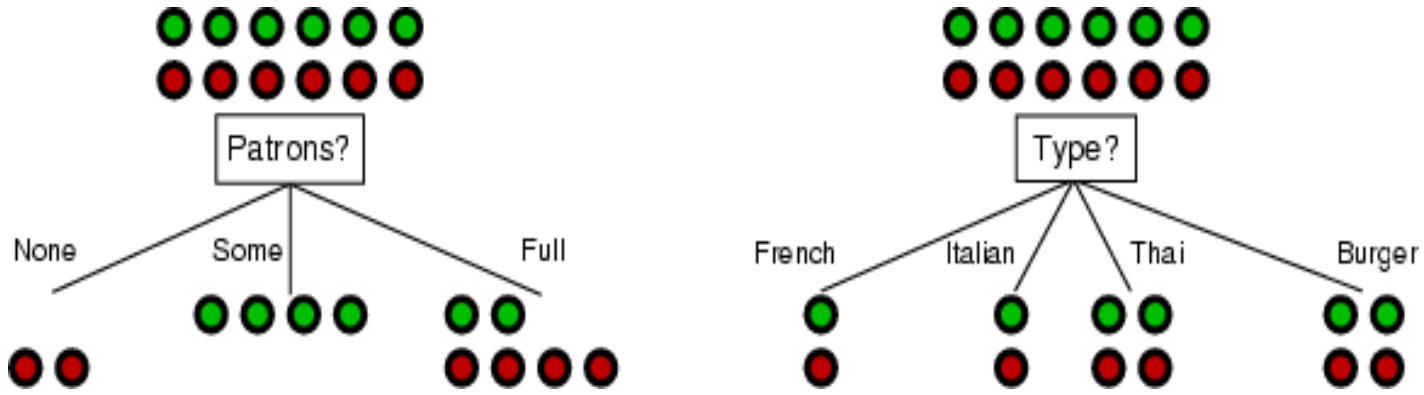
Our decision trees *split* individual attribute domains, producing axis-aligned rectangular decision regions

Some attributes are *correlated* (i.e. vary together), leading to non-rectangular regions (e.g. for classes)

However, even with an arbitrary linear separator, we cannot always represent the target function *f*.

Attribute Selection for Decision Tree Learning

- A Good Attribute
  - Splits examples into (***ideally***) "all positive" or "all negative" subsets

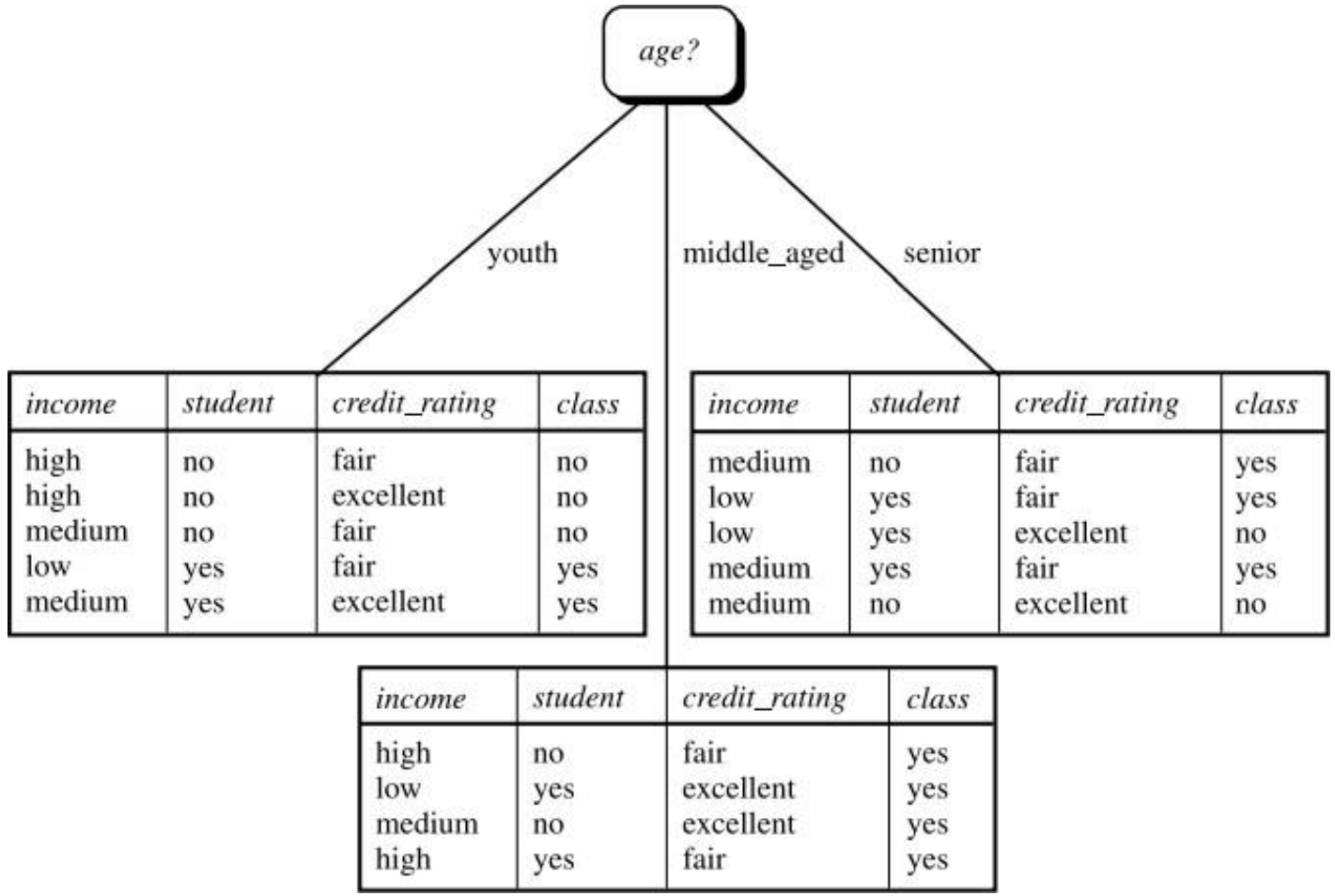


- For Example Above (Selecting Tree Root Node)
  - *Patrons?* is a better choice than *Type?* as attribute to select

It's white board time! Crafting ***top-down induction*** *(ID3)!*

Class-Labeled Training Tuples from the *AllElectronics* Customer Database

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no



income	*student*	*credit_rating*	*class*
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	*student*	*credit_rating*	*class*
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	*student*	*credit_rating*	*class*
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

The Top-Down Induction Algorithm

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition, *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

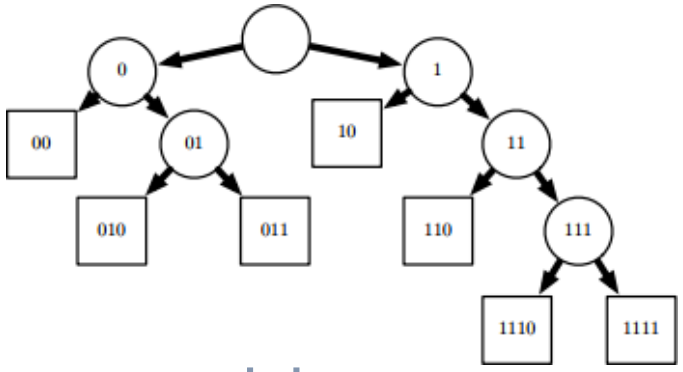
Method:

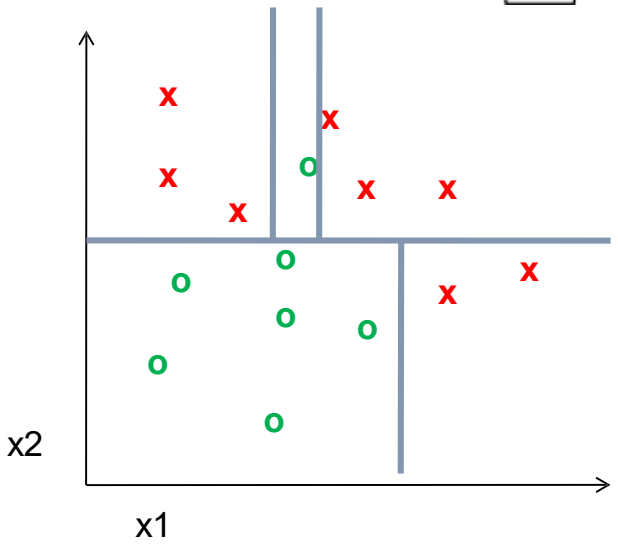
(1) create a node *N*;
(2) **if** tuples in *D* are all of the same class, *C*, **then**
(3) return *N* as a leaf node labeled with the class *C*;
(4) **if** *attribute_list* is empty **then**
(5) return *N* as a leaf node labeled with the majority class in *D*; // majority voting
(6) apply **Attribute_selection_method**(*D*, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node *N* with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
multiway splits allowed **then** // not restricted to binary trees
(9) *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome *j* of *splitting_criterion*
// partition the tuples and grow subtrees for each partition
(11) let D_j be the set of data tuples in *D* satisfying outcome *j*; // a partition
(12) **if** D_j is empty **then**
(13) attach a leaf labeled with the majority class in *D* to node *N*;
(14) **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node *N*;
endfor
(15) return *N*;

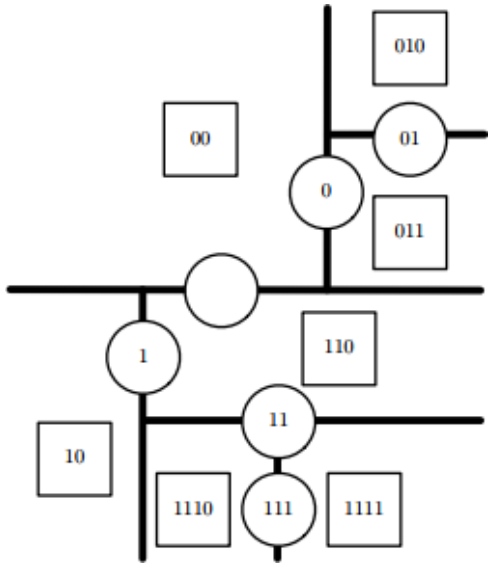
On Continuous Variables & Gain Ratio (C4.3)

(Yay, More White Board!)

Decision Trees – Aggregated Piecewise Functions







Divides (input) space into regions (nodes on lines categorize samples, leaf nodes on regions correspond to examples received). Piecewise-constant function (cannot learn function with more local maxima than number of training samples).

QUESTIONS?