Lesson 7 Decision Trees

Outline

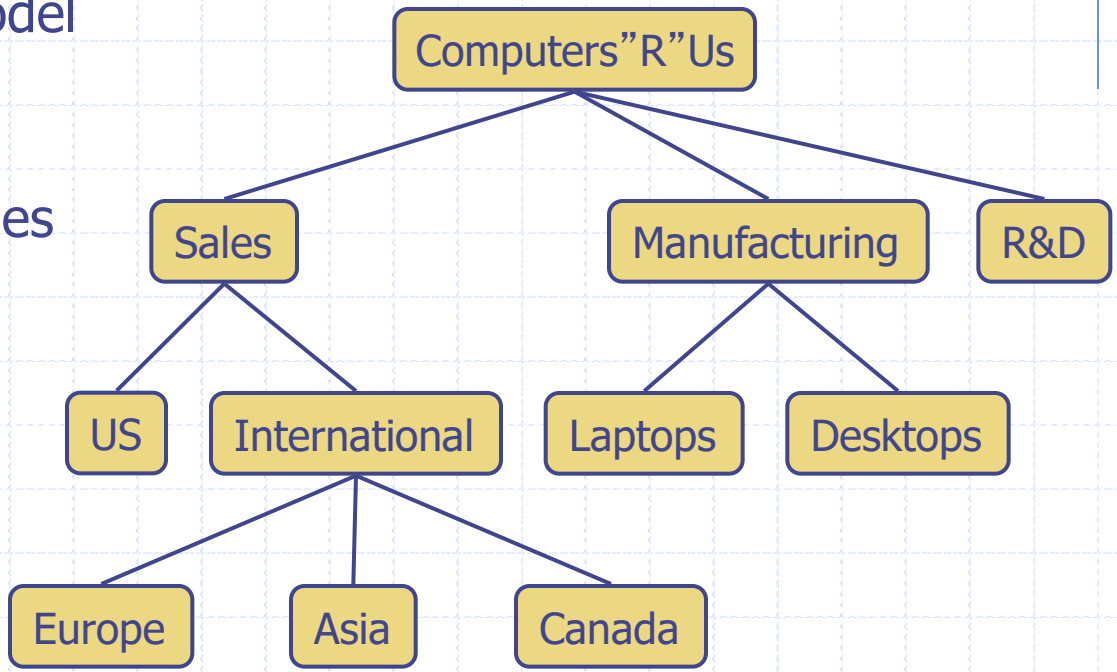
- Introduction to Decision Trees
- TDIDT: Top-Down Induction of Decision Trees
- Attribute selection
  - Entropy, Information Gain
  - Gain Ratio
- Decision Tree Learning
- Overfitting and Tree Pruning

Introduction to Decision Trees

- Decision tree algorithm generates a tree from the training set.
- It is a simple hierarchically structured way to guide one's path to a decision.
- Decision tree learning is one of the most widely used techniques for classification.
  - Its classification accuracy is competitive with other methods, and
  - When implemented correctly, it is very efficient. (https://stats.stackexchange.com/questions/105487/the-efficiency-of-decision-tree)

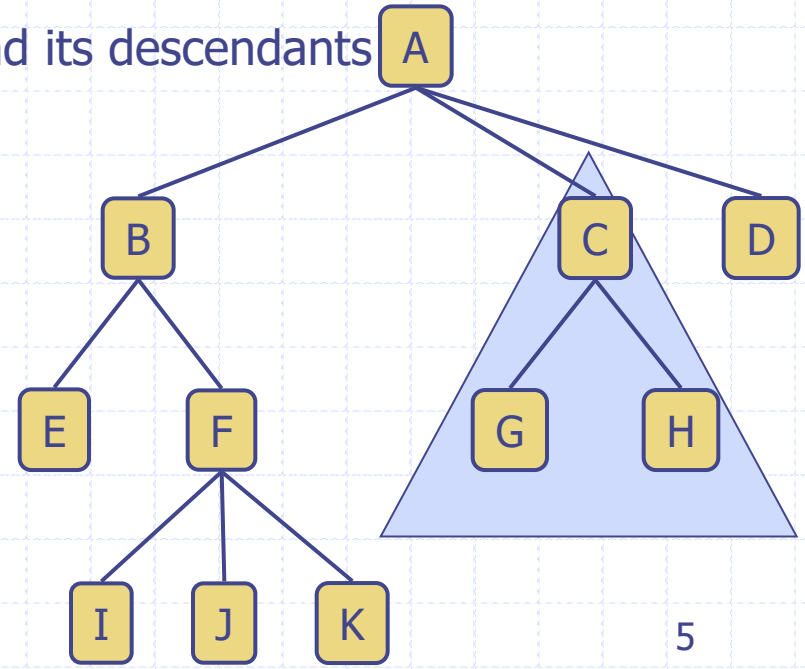
Trees

- In computer science, a tree is an abstract model of a hierarchical structure
- A tree consists of nodes with a parent-child relation
- Applications:
  - Organization charts
  - File systems



Tree Terminology

- **Root:** node without parent (A)
- **Inner node**: node with at least one child (A, B, C, F)
- **Leaf node:** node without children (E, I, J, K, G, H, D)
- **Ancestors of a node:** parent, grandparent, grand-grandparent, etc. (ancestors of K: F, B, A)
- **Descendant of a node:** child, grandchild, grand-grandchild, etc. (descendants of B are E, F, I, J, K)
- ***Subtree***: tree consisting of a node and its descendants
- **Edge:** Link between two nodes. (CG, AB…)



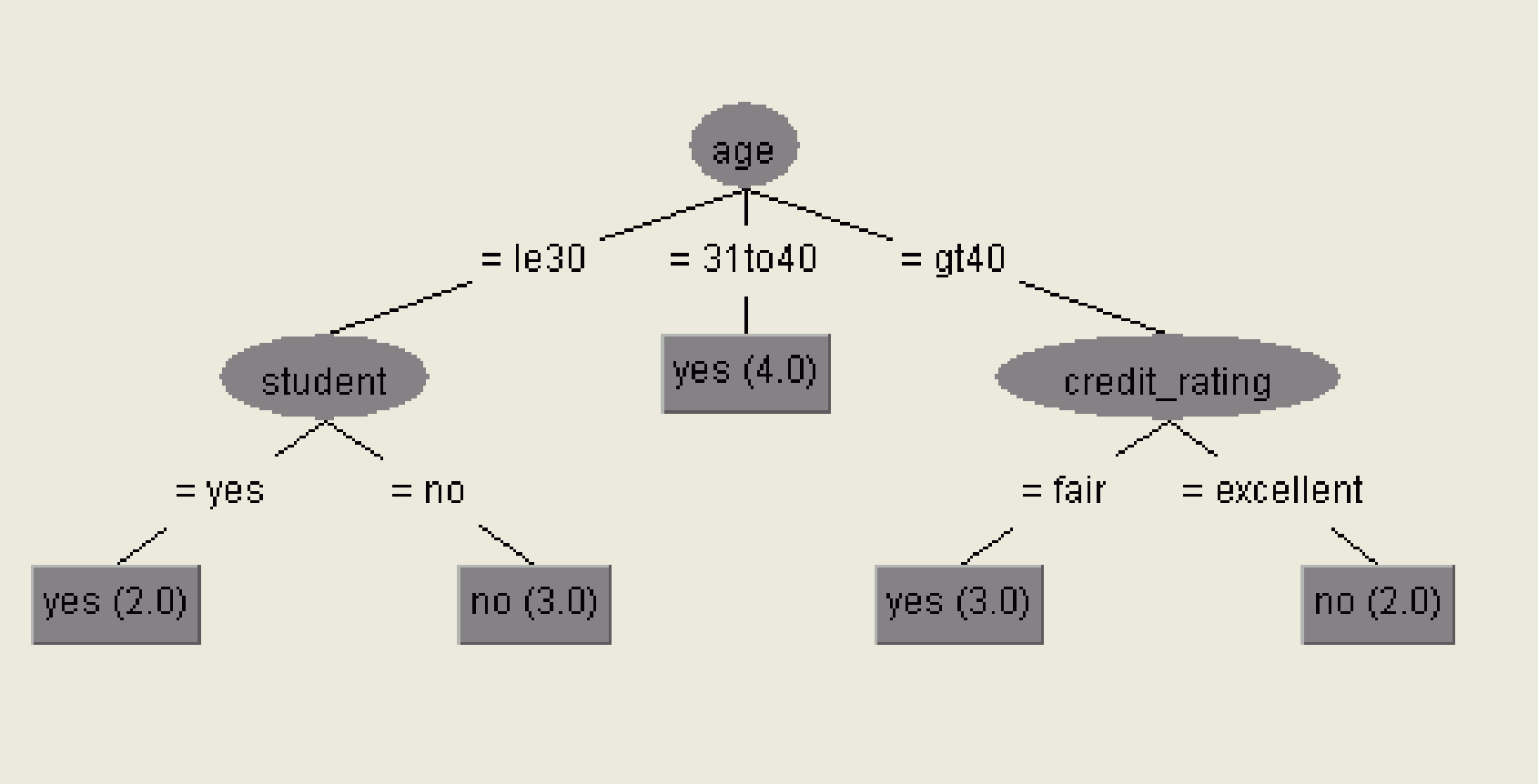
Anatomy of the Decision Tree

- A decision tree is a tree with the following properties:
  - An inner node represents an attribute.
  - An edge represents a test on the attribute of the parent node.
  - A leaf represents one of the classes.
- Construction of a decision tree
  - Based on the training data
  - Top-Down strategy

Training Data

Age	Income	Student	Credit Rating	Buys Computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

A Decision Tree for the data



Question: Can you come up with another decision tree for the training data?

Decision Trees – General Idea

- It is a divide and conquer algorithm
- Recursively divides a training set:
  1. Create a root node and assign all of the training data to it
  2. Select the best splitting attribute (how?)
  3. Add a branch to the root node for each value of the split. Split the data into mutually exclusive subsets along the lines of the specific split
  4. Repeat steps 2 and 3 for each leaf node until the stopping criteria are satisfied (when to stop?)

ID3 Algorithm: Pseudo-code

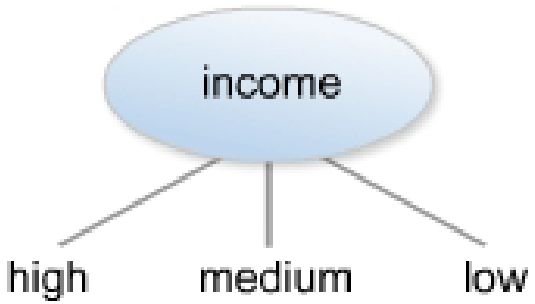
Algorithm ID3(S)
Input Data Set S
Output Decision Tree DT
  if all data in S belong to the same class c **then**
    return a new leaf and label it with c
  else
    Select an attribute A according to some heuristic function
    Generate a new node DT with A as test
    for each value v_i of A do
      S_i <- all data in S with A = v_i
      DT_i <- ID3(S_i) //use ID3 to construct a decision tree DT_i for S_i
      Generate an edge that connects DT and DT_i
  return DT

Issues from the Algorithm

- Which attribute to select as the root?
- Which attribute to select along the tree?
- How does one specify the attribute test condition?
- How does one determine the best split?
- When to stop? What are the stopping criteria?

Specifying the Test Condition

- How the attribute is split depends upon the type of the attribute (*categorical* or *continuous*) and the number of ways to split at the node (*2-way* split or *multi-way* split). In the case of categorical attributes, when multi-way split is used, the node is split into as many partitions as there are distinct values of that attribute.

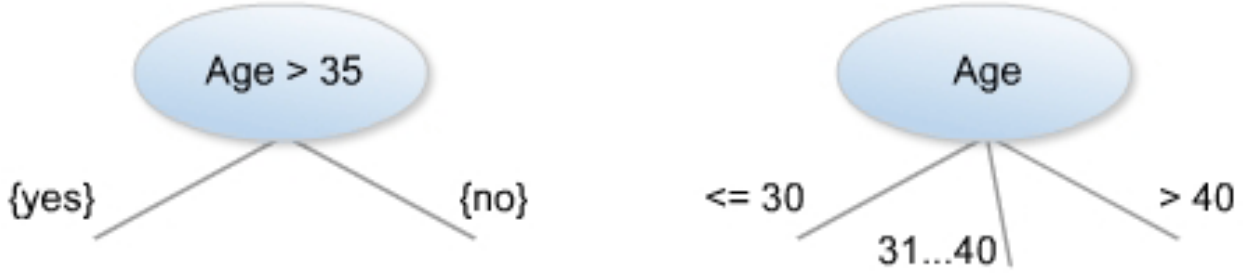


Specifying the Test Condition

- In the case of binary split, the data samples are partitioned into two parts. Optimal partitioning needs to be found based on the number of distinct values of this attribute. (Example below, but there are other possibilities.)

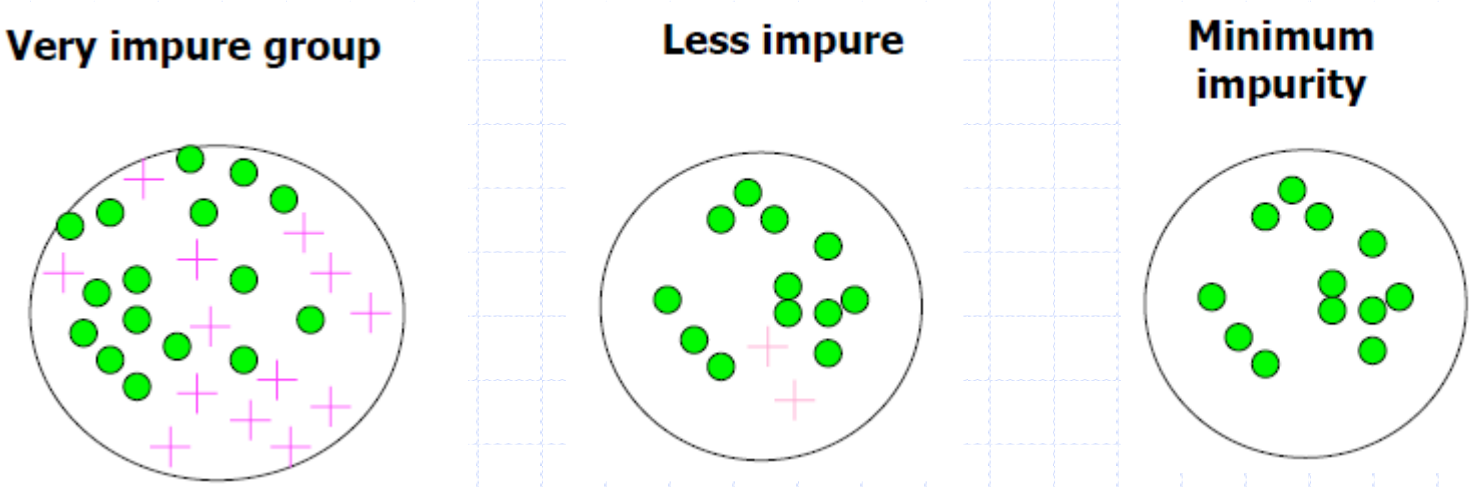
Specifying the Test Condition

- In the case of continuous attributes, the attribute can be discretized to a few categories. Otherwise, using numeric values, a binary split can be made using the best possible split value.



Determining the Best Split

- We want to grow a simple tree that prefers attributes to split the data so that each successor node is as *pure* as possible
  - *Purity* means that the distribution of examples in each node is such that each node mostly contains examples of only a single class



Entropy

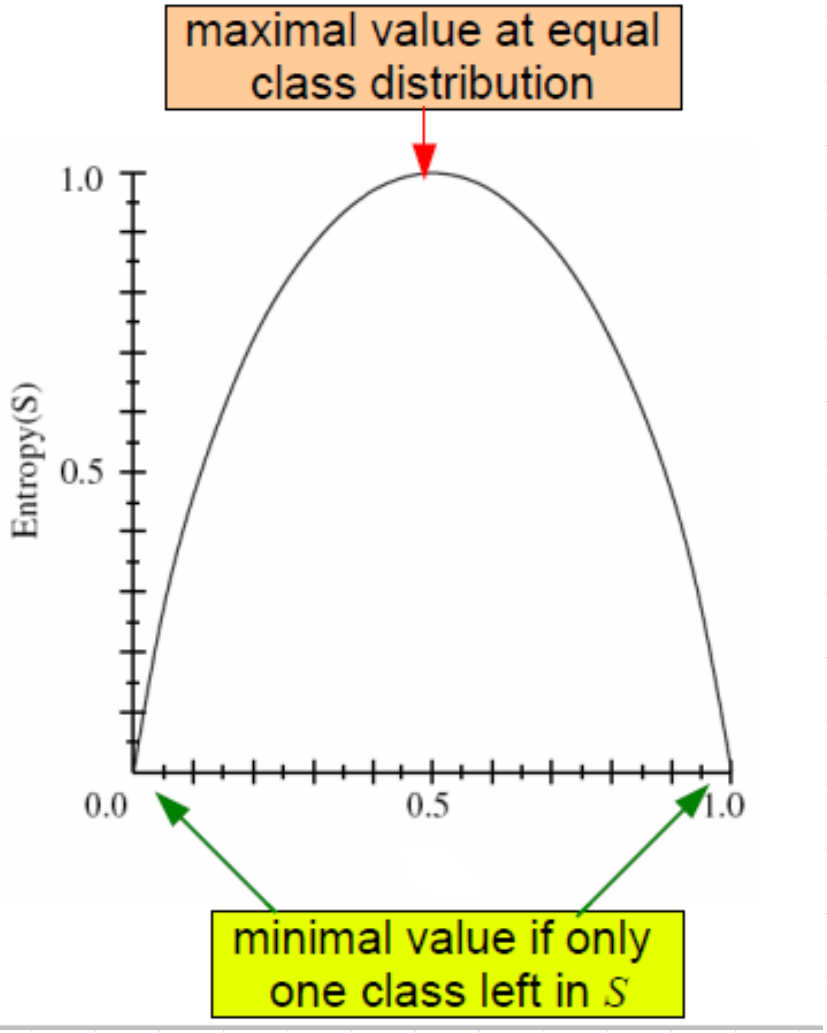
- *Information gain* is how we can measure purity after split.
- First, we need to know a concept called *Entropy* or the *Expected Information* associated with each node.
- Entropy at a given node *t* is given by the following formula: (m is the number of classes, $p(C_j|t)$ is the probability that an arbitrary data item in note t belongs to the class C_j.)

$$\text{Entropy(t)} = -\sum_{j=1\ldots m} p(C_j|t)\log p(C_j|t)$$

Entropy for two classes

x axis represents probability that one class is in the node t. y axis represents the entropy of the node t.

Note: You can think of purity as being the opposite of entropy.



Entropy: Example

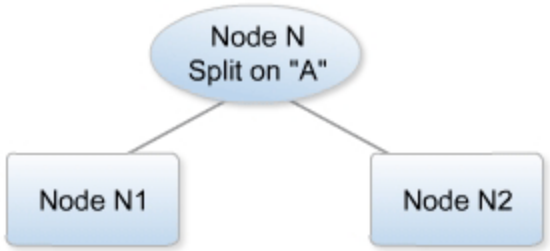
- Consider the following node, N, before selecting an attribute for splitting, where there are n1 data items that belong to class C1 and n2 items in class C2.
- The probability that a data item belongs to class C1 in this node is $\frac{n1}{n1+n2}$. Similarly, the probability that a data item belongs to class C2 in this node is $\frac{n2}{n1+n2}$.

$$\text{Entropy(N)} = -\frac{n1}{n1+n2}\log\left(\frac{n1}{n1+n2}\right) - \frac{n2}{n1+n2}\log\left(\frac{n2}{n1+n2}\right)$$

C1	**n1**
C2	**n2**

Selecting the attribute

- Problem:
  - Entropy(N) only tells us about the distribution of classes at a particular node N, but does not tell us about possible split.
  - How can we compute the quality of the entire split?
- Solution:
  - Compute the weighted average over all sets resulting from the split. ($\text{Entropy}_\text{A}(\text{N})$ represents the average entropy for attribute A, and x is the number of nodes after splitting.)
    - $\text{Entropy}_\text{A}(\text{N}) = \sum_{i=1\ldots x} \frac{|Ni|}{|N|} Entropy(Ni)$



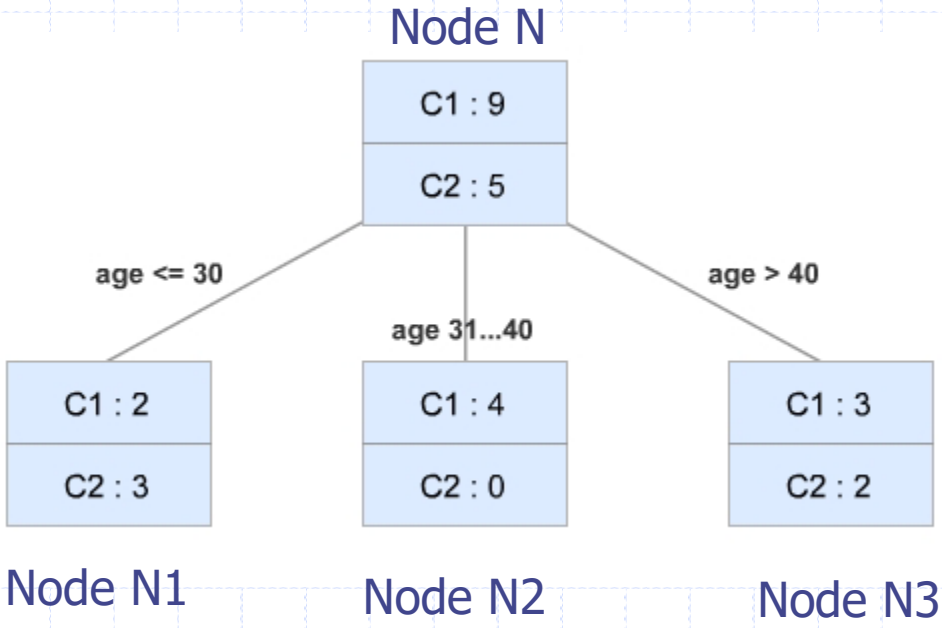
Selecting the attribute: Information Gain

- When a decision has to be made to select an attribute to split the data items in this node, the attribute that results in the highest *information gain* is the one that is selected.
- Information gain is defined as:

$$\text{Gain}(N, A) = \text{Entropy}(N) - \text{Entropy}_A(N)$$

- Note: Maximizing information gain is equivalent to minimizing average entropy.

Information Gain Example

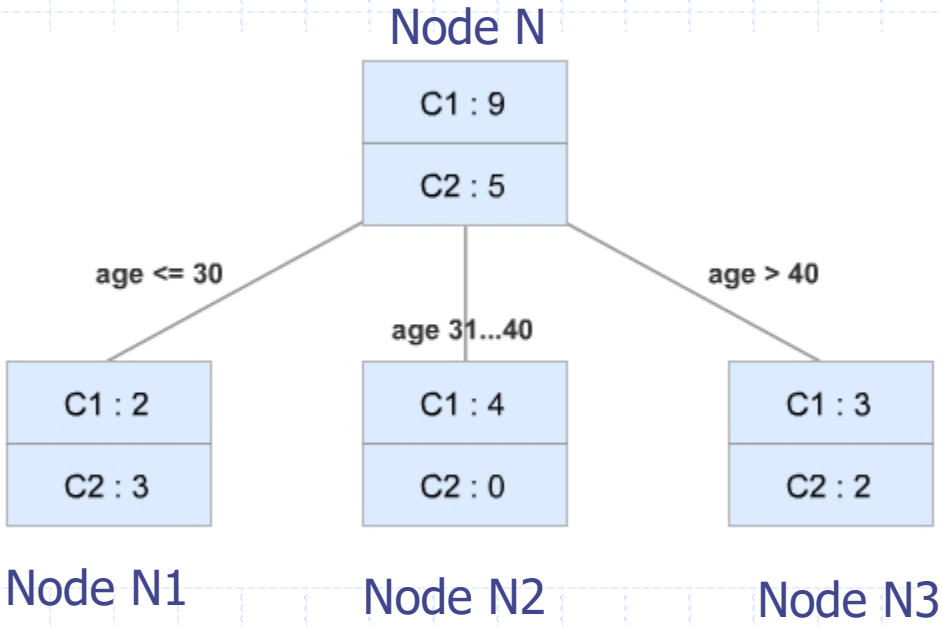


At the node N:

- P(C1|N) : P(BuysComputer = yes) = 9/14
- P(C2|N) : P(BuysComputer = no) = 5/14

Entropy(N)=−9/14log(9/14)−5/14log(5/14) = 0.94

Information Gain Example



Similarly:

Entropy(N1)=−2/5log(2/5)−3/5log(3/5)=0.971

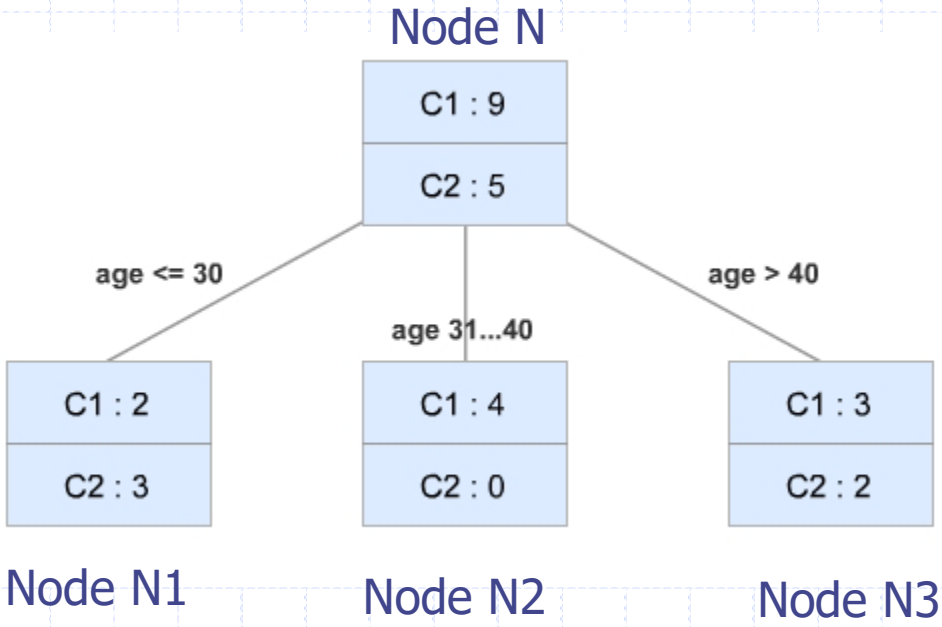
Entropy(N2)=−4/4log(4/4)−0/4log(0/4)=0

Entropy(N3)=−3/5log(3/5)−2/5log(2/5)=0.971

The average entropy for attribute age:

$Entropy_{age}(N)=5/14Entropy(N1)+4/14Entropy(N2)+5/14Entropy(N3)=0.694$

Information Gain Example



Finally, to compute information gain:

$Gain(N,age)=Entropy(N)-Entropy_{age}(N)=0.94-0.694=0.246$

Selecting the attribute

- Similarly, the gains associated with selecting the other attributes can be calculated:
  - Gain(N, income)=0.029
  - Gain(N, student)=0.151
  - Gain(N, CreditRating)=0.048
- Based on the above calculations, the *age* attribute results in the highest information gain when the node is split and so the *age* attribute is chosen to split the node N. The above calculations are repeated at each node during the decision tree building process whenever selecting an attribute.

Determining the Split Value for a Continuous Attribute

- Standard method: binary splits
  - Multiway Splits is more involved so we will not discuss in this course.
- Unlike categorical attributes, every attribute has many possible split points
- Solution is straightforward extension:
  - for every "possible" split point of attribute
  - evaluate info gain
  - choose "best" split point
- Computationally more demanding

Determining the Split Value for a Continuous Attribute

- Efficient computation needs only one scan through the values
  - sort the values
  - take the split values as the mid points between adjacent values
  - each time updating the count matrix and computing the evaluation measure
  - choose the split position that has the "best" value

	Taxable Income (in 000s)																			
Sorted values	60		70		75		85		90		95		100		120		125		220	
Id	6		3		9		8		10		5		2		4		1		7	
Cheat	No		No		No		Yes		Yes		Yes		No		No		No		No	
Split value	55000		65000		72500		80000		87500		92500		97500		110000		122500		172500		230000	
	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Entropy	0.881		0.826		0.764		0.690		0.875		0.846		*0.600*		0.690		0.764		0.826		0.881	

Other Measures for Attribute Selection

- *Gain ratio*: C4.5 (a successor of ID3) uses gain ratio. Weka's J48 is an implementation of C4.5.

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2\left(\frac{|D_j|}{|D|}\right)$$

  - Here, attribute A has v distinct values, $|D_j|$ is the number of tuples with the j-th value.
  - GainRatio(A) = Gain(A)/$SplitInfo_A(D)$
  - Gain(A) is the information gain on attribute A in previous slides.
  - The attribute with the maximum gain ratio is selected as the splitting attribute

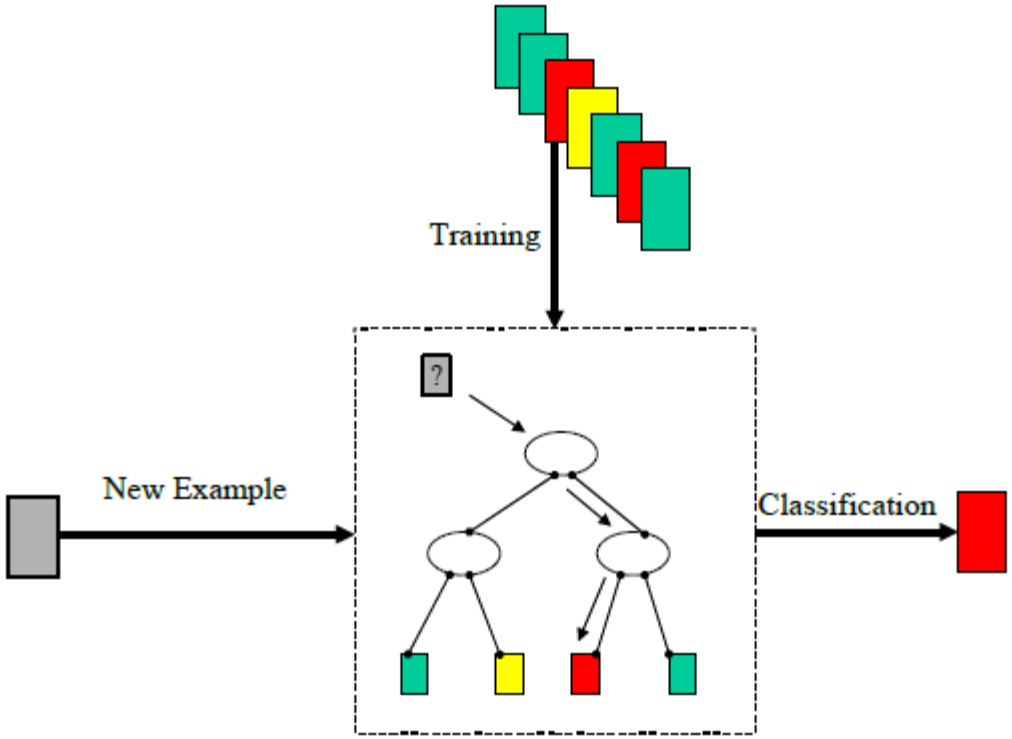
Other Measures for Attribute Selection

- GINI index
- CHAID
- C-SEPG-statisticMDL (Minimal Description Length) principle
- Multivariate splits (partition based on multiple variable combinations)
  - Example: CART
- Which attribute selection measure is the best?
  - Most give good results, none is significantly superior to others

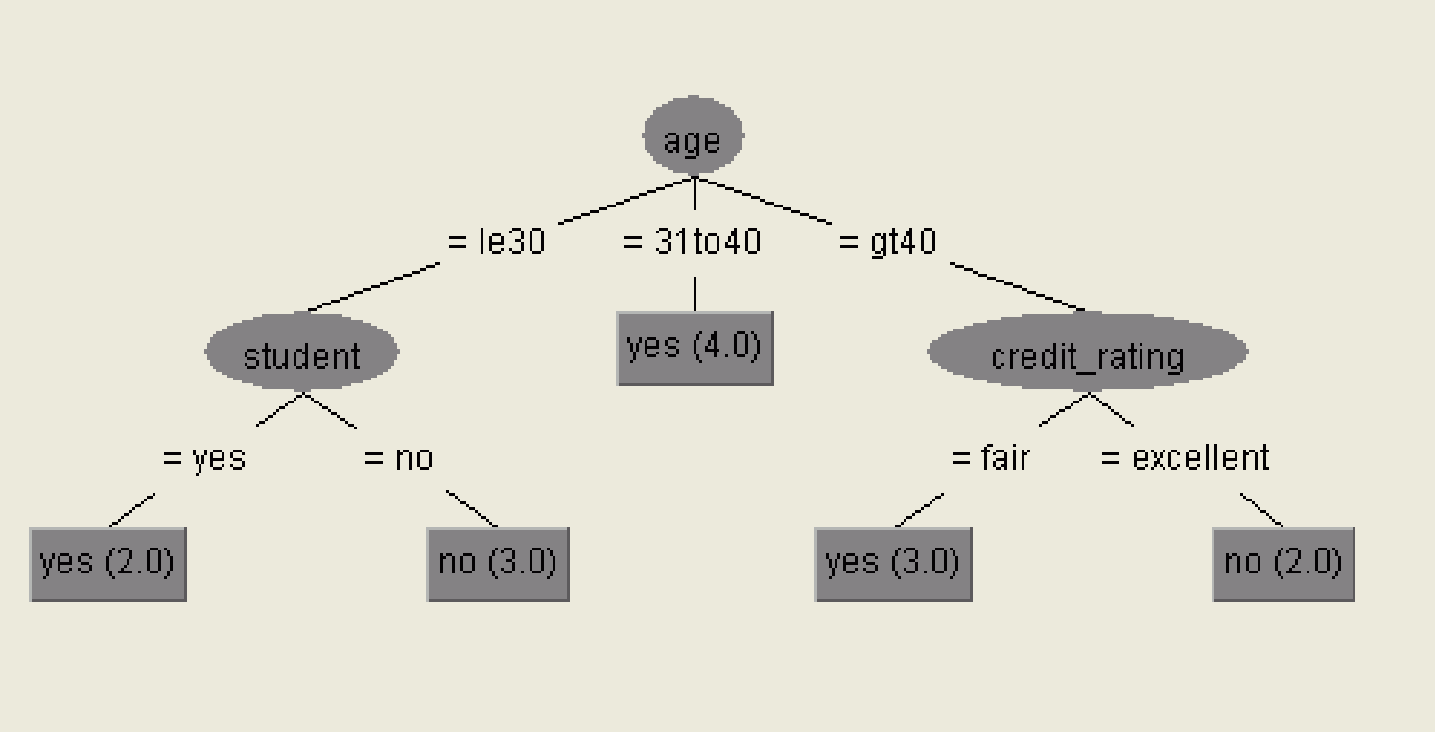
Decision Tree Learning

The training examples are used for choosing appropriate tests in the decision tree. Typically, a tree is built from top to bottom, where tests that maximize the information gain/gain ratio about the classification are selected first.

In Decision Tree Learning, a new example is classified by submitting it to a series of tests that determine the class label of the example.



Use Decision Tree for Classification



Age	Income	Student	Credit Rating	Buys Computer
> 40	high	no	fair	?

R code (See demo code)

```
library(rpart)
library(rpart.plot)

#recursive partition and regression trees algorithm
#rpart() function uses the Gini impurity measure to split the node
tree <- rpart(Species ~
              Petal.Length+Petal.Width+Sepal.Width+Sepal.Length, train)

rpart.plot(tree, extra = 1)
predict <-predict(tree, test, type = 'class')

table(predict, test$Species)
```

Overfitting

- Overfitting arises when the learning model fits the given training data so much that it reflects every detail of the training dataset, even an anomaly or noise.
- The model becomes complex.
- Results in high accuracy relative to the training dataset and low accuracy relative to the test dataset.
- So, an overfitted model does not generalize well.

Avoid Overfitting in Classification: Tree Pruning

- Two approaches to avoid overfitting
  - *Pre-pruning*: Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - *Post-pruning* (more common): generate a tree in full and then remove parts of it.
    - One of the post-pruning algorithms is to start from the bottom of the tree, and for each node N, compute the cost complexity of the subtree at N and the cost complexity of the subtree at N if it were to be pruned. Compare the two values - if pruning the subtree would result in a smaller cost complexity, then the subtree is pruned. Otherwise it is kept.
    - Cost complexity: a function of the number of leaves in the tree and the error rate (percentage of tuples misclassified) of the tree.