DECISION TREES

- Nod intern = test pe un atribut
- Arc = rezultatul unui test
- Frunza (nod terminal) = o eticheta de clasa

- Algoritmi:
  - ID3, C4.5, CART
  - Abordare greedy, top-down
  - Setul initial de inregistrari (training set) este partitionat in subseturi mai mici pe masura ce arborele este construit

- Parametri
  - D = set de inregistrari curent (initial, este setul total de inregistrari avand etichetele de clasa asociate)
  - Lista de atribute (care descriu inregistrarile)
  - Metoda de selectie a atributului folosit pentru splitting
    - Aceasta metoda decide daca arborele este binar sau nu

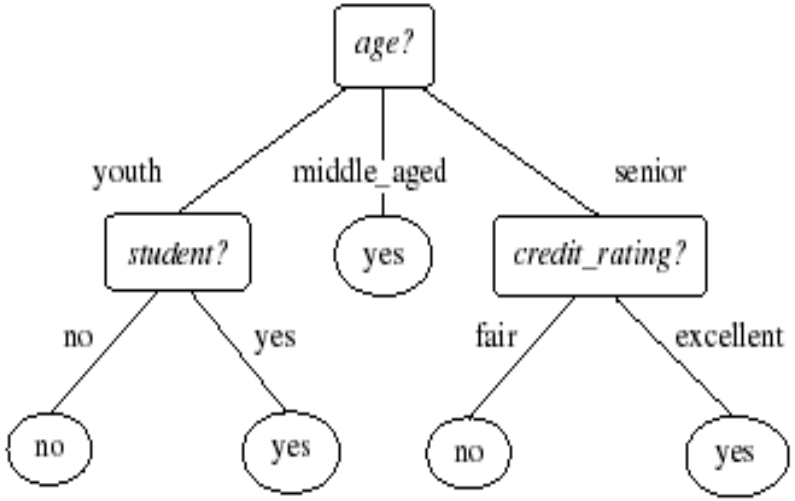

Figure 6.2 A decision tree for the concept *buys_computer*, indicating whether a customer at *AllElectronics* is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).

Referinta figura: J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, 2nd Edition, Morgan Kaufmann, 2006.

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 Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)     if Dj 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(Dj, attribute_list) to node N;
     endfor
(15) return N;
```

Referinta figura: J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, 2nd Edition, Morgan Kaufmann, 2006.

Descriere algoritm:

- Arborele porneste cu un singur nod N care contine setul initial de inregistrari
- Daca toate inregistrarile din D apartin aceleasi clase, atunci N devine nod frunza si i se pune eticheta acelei clase
- Altfel, se aplica metoda de determinare a celui mai potrivit atribut pentru splitting
  - Atribut pentru splitting
  - Punct de splitting
  - Subset de splitting
- Scopul este ca partitiile astfel rezultate sa fie cat mai pure
  - O partitie este **pura** daca toate inregistrarile sale apartin aceleasi clase

- Nodul N este etichetat cu criteriul de splitting (test)
- Rezulta cate o ramura pentru fiecare rezultat posibil al testului
- Inregistrarile sunt partitionate in functie de rezultatul testului
- Scenarii posibile de splitting (fie A atributul selectat pentru splitting)
  - 1. valori discrete – cate o ramura pentru fiecare valoare posibila; ramura este etichetata cu acea valoare aj; partitia Dj este subsetul de inregistrari care au valoarea aj pentru atributul A
  - 2. valori continue – testul are 2 iesiri posibile: A<=split point respectiv A>split point
  - Punctul de splitting este returnat de catre metoda de selectie a atributului (de obicei este considerat a fi mijlocul distantei dintre 2 valori adiacente, deci nu este o valoare pe care o ia A)
  - 3. valori discrete dar se doreste un arbore binar
- Atributul A este scos din lista de atribute

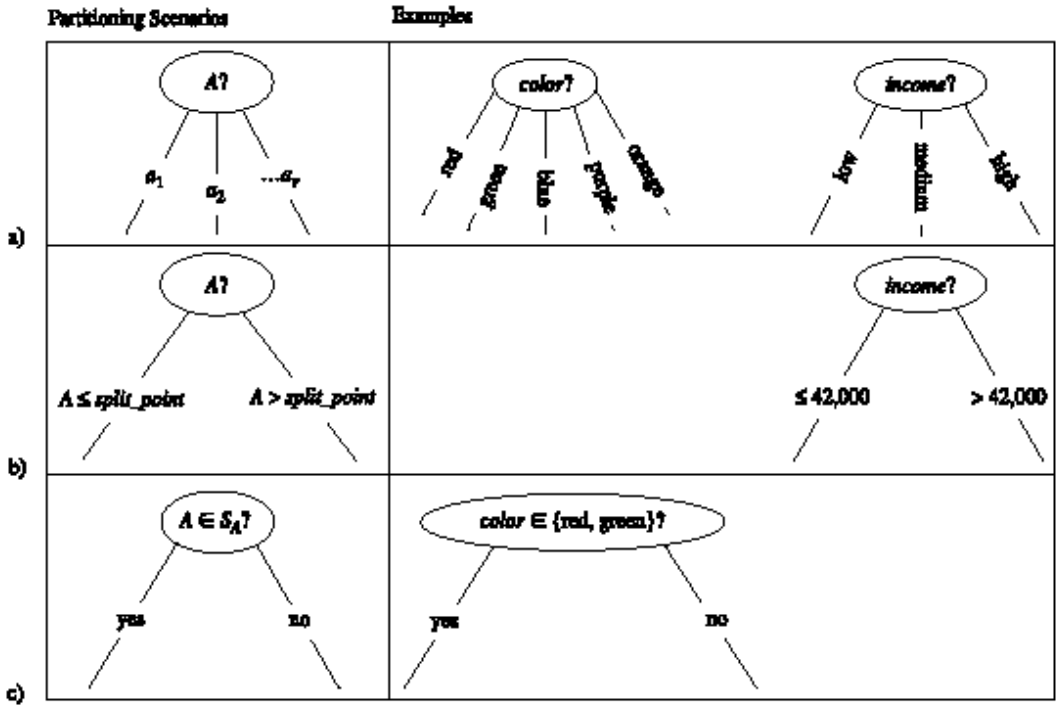

Figure 6.4 Three possibilities for partitioning tuples based on the splitting criterion, shown with examples. Let A be the splitting attribute. (a) If A is discrete-valued, then one branch is grown for each known value of A. (b) If A is continuous-valued, then two branches are grown, corresponding to $A \leq split\_point$ and $A > split\_point$. (c) If A is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where S_A is the splitting subset for A.

Referinta figura: J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, 2nd Edition, Morgan Kaufmann, 2006.

- Criterii de terminare
  - Toate inregistrarile dintr-un subet apartin aceleasi clase
  - Nu mai exista atribute dupa care putem sa continuam partitionarea
    - In acest caz N devine nod frunza si se eticheteaza cu clasa majoritara (sau se poate retine distributia in clase a inregistarilor)
  - Nu exista inregistrari pentru o anumita ramura, adica o partitie Dj este vida => se creeaza un nod frunza cu clasa majoritara din D

- Masuri de selectie a atributelor (reguli de splitting)
  - Se calculeaza pentru fiecare atribut si se alege atributul cu cel mai mare scor
  - D – training set
  - Atributul de clasa (cel care trebuie clasificat) are m valori distincte => m clase distincte Ci
  - C(i,D) = setul de inregistrari de clasa Ci din D

- **Information gain**
  - Folosit de ID3
  - Se bazeaza pe teoria informatiei
  - Informatia necesara pentru a clasifica o inregistrare din D

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

  - pi =probabilitatea ca o inregistrare arbitrara din D sa apartina clasei Ci
  - Info(D)=entropia

- Sa presupunem ca am partitiona setul D dupa atributul A care are n valori distincte {a1, … , an} => partitiile {D1,…Dn}
- Dj = acele inregistrari pentru care atributul A ia valoarea aj
- Se doreste ca fiecare partitie sa fie cat mai pura
- Informatia necesara pentru a ajunge la o clasificare exacta este:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

- Cu cat este mai mica, cu atat e mai mare puritatea
- Atributul cu cel mai mare castig de informatie este selectat pentru splitting

$$Gain(A) = Info(D) - Info_A(D).$$

Table 6.1 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

$$Info(D) = -\frac{9}{14}\log_2\left(\frac{9}{14}\right) - \frac{5}{14}\log_2\left(\frac{5}{14}\right) = 0.940 \text{ bits.}$$

$$\begin{aligned} Info_{age}(D) &= \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) \\ &+ \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4}\right) \\ &+ \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right) \\ &= 0.694 \text{ bits.} \end{aligned}$$

$$Gain(age) = Info(D) - Info_{age}(D) = 0.940 - 0.694 = 0.246 \text{ bits.}$$

Referinta figura: J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, 2nd Edition, Morgan Kaufmann, 2006.

- Pentru valori continue
  - Trebuie sa determinam cel mai bun splitting point
  - Se sorteaza valorile lui A in ordine crescatoare
  - Se considera mijlocul distantei pentru fiecare pereche de valori adiacente => pentru n valori posibile vom avea n-1 posibile puncte de splitting
  - Pentru fiecare punct se evalueaza InfoA(D) (cu 2 partitii)
  - Se alege punctul cu aceasta valoare minima

- **Gain ratio**
  - Information Gain favorizeaza testele cu mai multe iesiri posibile (de ex, pentru atributul ID vom avea doar seturi pure, fiecare continand o singura inregistrare)
  - C4.5, succesorul lui ID3, foloseste o extensie a acestei masuri pentru a rezolva aceasta problema, aplicand o normalizare

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

  - Se tine cont de numarul de inregistrari cu valoare aj pentru atributul A relativ la numarul total de inregistrari, si nu de clasificare
  - Se alege atributul cu Gain Ratio maxim

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}.$$

- **Indexul Gini**
  - Folosit de CART
  - Acest index masoara impuritatea lui D

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

  - pi=probabilitatea ca o inregistrare in D sa apartina clasei Ci
  - Are loc un splitting binar
  - Pentru partitionarea lui D in D1 si D2 (in functie de valorile atributului A) se calculeaza suma proportionala a impuritatilor partitiilor rezultate

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2).$$

  - Scopul este reducerea impuritatii

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

- Reducerea arborilor (tree prunning)
  - Se obtin arbori mai simpli, mai mici => mai usor de inteles

- Exista 2 tipuri de abordari
  - Se opreste construirea unei ramuri in timpul generarii arborelui
    SAU
  - Se sterg ramuri din arbori deja construiti
- Decizia se bazeaza pe masuri ca: semnificatie statistica, castig de informatie, index Gini, … (daca splitting-ul rezulta intr-o masura sub un anumit prag, atunci se decide oprirea dezvoltarii)
- Pragul potrivit nu este usor de ales

Tree prunning

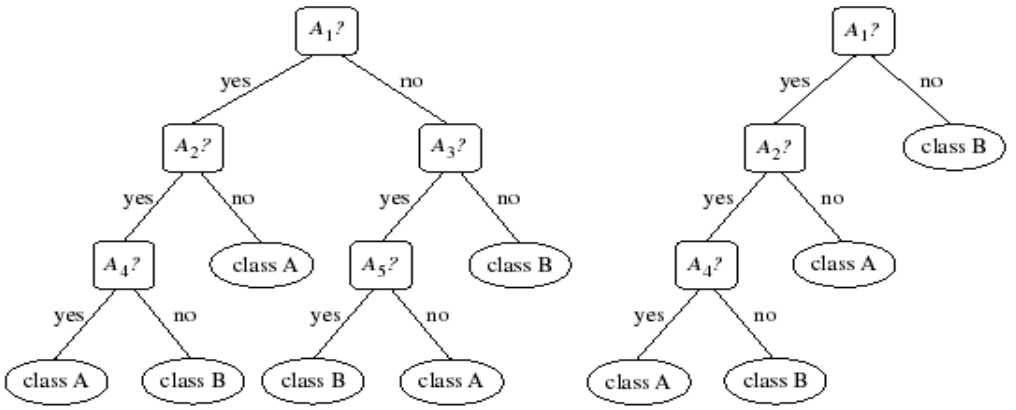

Figure 6.6 An unpruned decision tree and a pruned version of it.

Referinta figura: J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, 2nd Edition, Morgan Kaufmann, 2006.