Data Mining

and

Concept Learning

Fall 2012

Stan Matwin

stan@site.uottawa.ca

Distinguished University Professor

Electrical Engineering and Computer Science

University of Ottawa

Canada

Goals of this course

- to convey and teach basic concepts of Machine Learning (ML) and Data Mining (DM)
- to outline interesting areas of current and future research
- to provide hands-on DM experience to students
- to present successful applications
- to suggest method selection principles for a given application
- to enable advanced self-study on ML/DM

Course outline

1. Machine Learning/Data Mining: basic terminology.
2. Symbolic learning: Decision Trees; learning as search; basic Evaluation of learning/mining results;
3. Theoretical aspects of ML. Bias, variance, learnability: the PAC framework.
4. Probabilistic learning: Bayesian learning.
5. Bayesian methods for learning from text data
6. Kernel-based methods, Support Vector Machine. Applications in Machine Vision and bioinformatics

7. **Ensembles of learners: boosting, bagging, random forest**

8. **Advance evaluation measures – ROC, AUC**

9. **Pre- and post-processing: Feature selection; combining classifiers; discretization**

10. **Distance-based classification. Clustering**

11. **Data mining concepts and techniques**

12. **Semi-supervised learning: co-training**

13. **Data mining and privacy**

NOTE: if time allows, we will cover advanced methods for sequential data (HMM, Conditional Random Fields, Gibbs sampling)

1. Machine Learning / Data Mining: basic terminology

- Machine Learning:
  - given a certain task, and a data set that constitutes the task,
  - ML provides algorithms that resolve the task based on the data, and the solution improves with time
- Examples:
  - predicting lottery numbers next Saturday
  - Detecting oil spills on sea surface
  - Assisting Systematic Reviews
  - Profiling
    - Dreams
    - Digital game players
    - Fraudulent CC use

- **Data Mining: extracting regularities from a VERY LARGE dataset/database as part of a business/application cycle**
- **examples:**
  - **customer churn profiling**
  - **direct mail targeting/ cross sell**
  - **security applications: monitoring of**
    - » **Social networks**
    - » **Computer networks**
  - **prediction of aircraft component failures**
  - **clustering of genes wrt their beavior**

Basic ML tasks

- Supervised learning
  - classification/concept learning
  - estimation: essentially, extrapolation
- Unsupervised learning:
  - clustering: finding groups of “similar” objects
  - associations: in a database, finding that some values of attributes go with some other

Concept learning (also known as *classification*): a definition

- Data are given as vectors of attribute values, where the domain of possible values for attribute *j* is denoted as *Aj*, for 1 <= *j* <= *N*. Moreover,a set *C* = {*c*1,., *ck*} of *k* classes is given; this can be seen as a special attribute or label for each record. Often *k* = 2, in which case we are learning a binary classifier.
- Inducing, or learning a classifier, means finding a mapping

  $F: A_1 \times A_2 \times A_N \rightarrow C,$

  given a finite <u>training</u> <u>set</u> $X1 = \{<x_{ij}, c_i>, 1 <= j <= N, c_i \in C, 1 <= i <= M\}$ of M labeled examples [comment on noise]

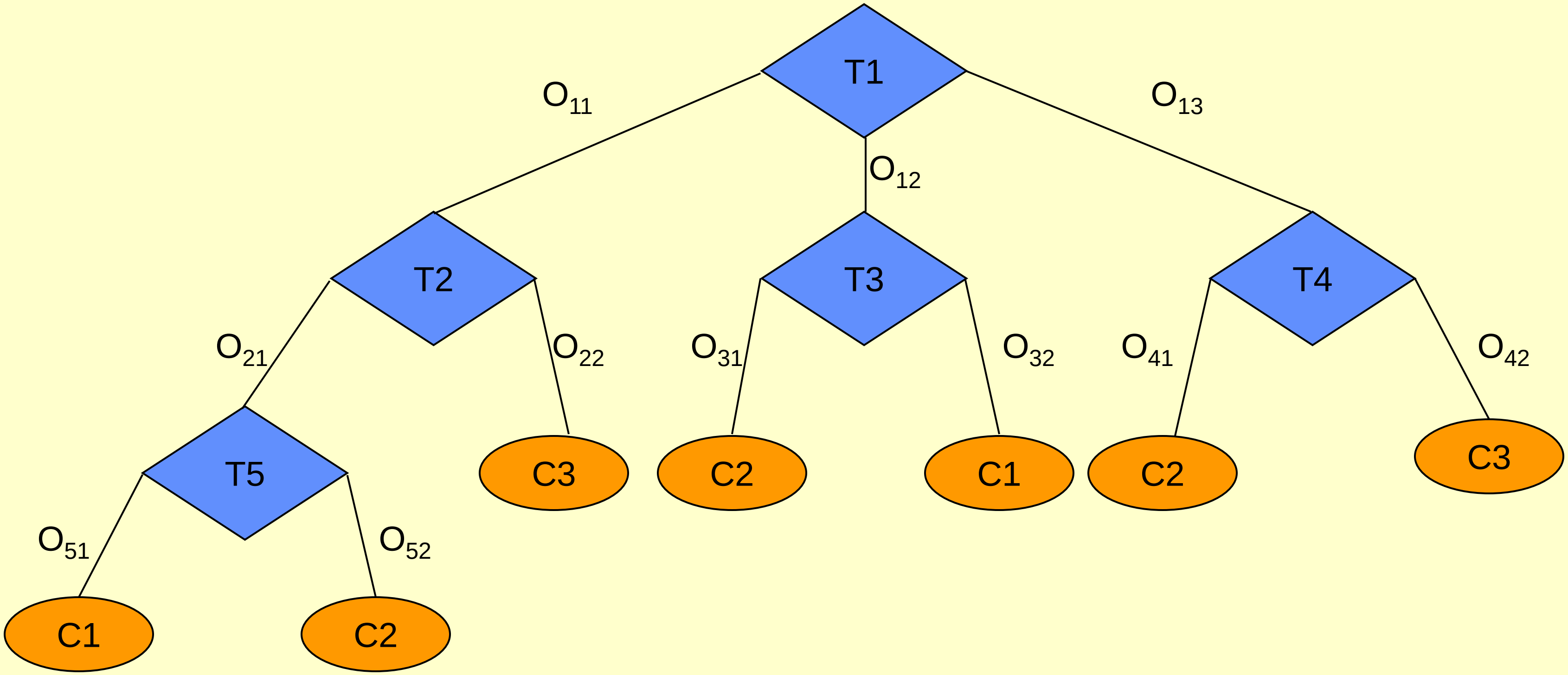
- We assume that data is represented as fixed size vectors of attributes (AVL representation): eg all patients are represented by the same 38 attributes, perhaps in conceptual groupings into personal, social, medical
- F belongs to a fixed language, e.g. F can be
  - a set of n -1 dimensional hyperplanes partitioning an n-dimensional space into k subspaces, or
  - a decision tree with leaves belonging to C, or
  - a set of rules with consequents in C.
- We also want F to perform well, in terms of its predictive power on (future) data not belonging to X1 [predictive power]

- In data base terminology, we “model” one relation
- There are methods that deal with multi-relational representations (multiple tables), - multi-relational learning AKA Inductive Logic Programming

Data Mining Process

- **An iterative process which includes the following steps**
  - **Formulate the problem e.g. Classification/Numeric Prediction**
  - **Collect the relevant data (No data, no model)**
  - **Represent the Data in the form of *labeled* examples (a.k.a instances) to be learned from**
  - **Learn a model/predictor**
  - **Evaluate the model**
  - **Fine tune the model as needed**

Decision Trees



Tests: T1, …, T5

Test Outcomes: O_{11}, …, O_{52}

Predictions: C1, …, C3

Example 1: Are We Going to Play Outdoors

- Predict whether there will be an outdoor game depending on the existing weather conditions
- Classification: yes (play), no (don’t play)
- Data:A collection of past observation about the days that there was or was not an out door game. The data is already collected and labeled.

Representation for Outdoor Game Prediction

- **Each example records the following information (a.k.a *features* or *attributes*)**
- **outlook {sunny, overcast, rainy}**
- **temperature real**
- **humidity real**
- **windy {TRUE, FALSE}**
- **play {yes, no} -> the label**

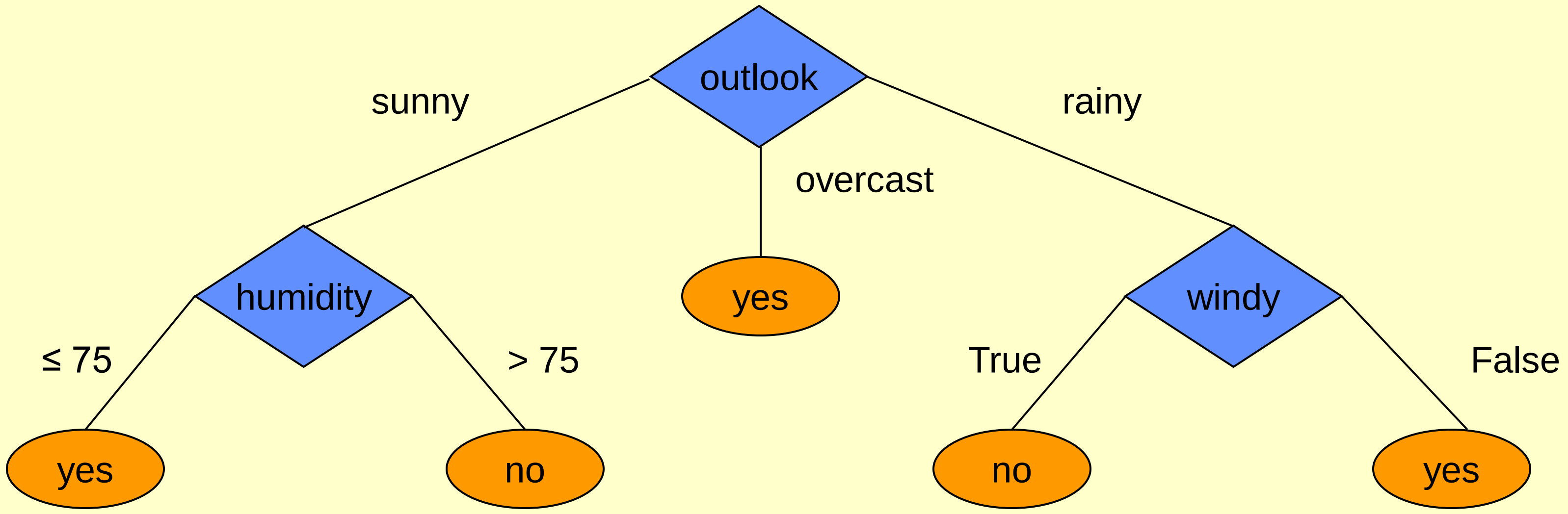
Outdoor Game Play Prediction Data for WEKA

```
% ARFF file for the weather data with some numeric features
%
@relation weather

@attribute outlook { sunny, overcast, rainy }
@attribute temperature numeric
@attribute humidity numeric
@attribute windy { true, false }
@attribute play? { yes, no }

@data
%
% 14 instances
%
sunny, 85, 85, false, no
sunny, 80, 90, true, no
overcast, 83, 86, false, yes
rainy, 70, 96, false, yes
rainy, 68, 80, false, yes
rainy, 65, 70, true, no
overcast, 64, 65, true, yes
sunny, 72, 95, false, no
sunny, 69, 70, false, yes
rainy, 75, 80, false, yes
sunny, 75, 70, true, yes
overcast, 72, 90, true, yes
overcast, 81, 75, false, yes
rainy, 71, 91, true, no
```

Outdoor Game Play Decision Tree



Outdoor Game Play Model (As output by WEKA)

```
J48 pruned tree
------------------

outlook = sunny
|   humidity <= 75: yes (2.0)
|   humidity > 75: no (3.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  :      5

Size of the tree :   8
```

Confusion Matrix

Correctly Classified Instances	9	64.2857 %
Incorrectly Classified Instances	5	35.7143 %

```
=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 3 2 | b = no
```

Correctly classified entries (row&column): aa, bb

Incorrectly Classified entries (row&column): ab, ba

False Positive/Days incorrectly classified as play: ba

False Negative/Days incorrectly classified as no play: ab

Example 2: Who would survive Titanic’s sinking

- Predict whether a person on board would have survived the tragic sinking
- Classification: yes (survives), no (does not survive)
- Data:The data is already collected and labeled for all 2201 people on board the Titanic.

Example 2: Representation for the Titanic Survivor Prediction

- **Each example records the following *attributes***
- **social class {first class,second class, third class, crew member}**
- **age {adult, child}**
- **sex {male, female}**
- **survived {yes, no}**

Titanic Survivor model

J48 pruned tree
----

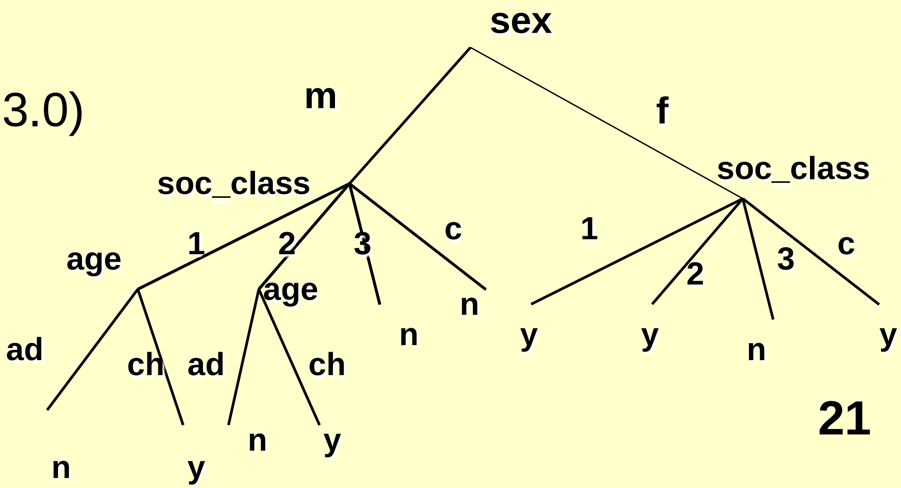
sex = male
| social_class = first
| | age = adult: no (175.0/57.0)
| | age = child: yes (5.0)
| social_class = second
| | age = adult: no (168.0/14.0)
| | age = child: yes (11.0)
| social_class = third: no (510.0/88.0)
| social_class = crew: no (862.0/192.0)
sex = female
| social_class = first: yes (145.0/4.0)
| social_class = second: yes (106.0/13.0)
| social_class = third: no (196.0/90.0)
| social_class = crew: yes (23.0/3.0)

Number of Leaves :   10
Size of the tree :   15

Correctly Classified Instances
1737   78.9187 %
Incorrectly Classified Instances
464   21.0813 %

=== Confusion Matrix ===

a   b   <-- classified as
267 444 | a = yes
20 1470 | b = no



Induction of decision trees: an algorithm building a DT from data…

building a *univariate (single attribute is tested)* decision tree from a set T of training cases for a concept C with classes $C_1, \ldots C_k$

Consider three possibilities:

- **T contains 1 or more cases all belonging to the same class C_j. The decision tree for T is a leaf identifying class C_j**
- **T contains no cases. The tree is a leaf, but the label is assigned heuristically, e.g. the majority class in the parent of this node**

- T contains cases from different classes. T is divided into subsets that seem to lead towards collections of cases. A test t based on a single attribute is chosen, and it partitions T into subsets $\{T_1,\ldots,T_n\}$. The decision tree consists of a decision node identifying the tested attribute, and one branch for ea. outcome of the test. Then, the same process is applied recursively to ea.T_i

Choosing the test

- **why not explore all possible trees and choose the simplest (Occam's razor)? But this is an NP complete problem. E.g. in the 'Titanic' example there are millions of trees consistent with the data**

- **idea: to choose an attribute that best separates the examples according to their class label**
- **This means to maximize the difference between the info needed to identify a class of an example in T, and the same info after T has been partitioned in accordance with a test X**
- **Entropy is a measure from information theory [Shannon] that measures the quantity of information**

- **information measure (in bits) of a message is $-\log_2$ of the probability of that message**
- **notation: S: set of the training examples; $freq(C_i, S)$ = number of examples in S that belong to C_i;**

selecting 1 case and announcing its class has info measure - $\log_2(\text{freq}(C_i, S)/|S|)$ bits

to find information pertaining to class membership in all classes: $\text{info}(S) = -\sum_1 (\text{freq}(C_i, S)/|S|)*\log_2(\text{freq}(C_i, S)/|S|)$

after partitioning according to outcome of test X:

$\text{info}_X(T) = \sum |T_i|/|T|*\text{info}(T_i)$

$\text{gain}(X) = \text{info}(T) - \text{info}_X(T)$ measures the gain from partitioning T according to X

We select X to maximize this gain

Other splitting criteria

Note that the information gain formula is a special case of a more general criterion:

$I(s) = P(L)f(P(+|L_s),P(-|L_s))+P(R)f(P(+|Rs),P(-|Rs))$

where *P(L), P(R)* are probabilities of an example going left or right, and assuming two classes – and +

And *f(a, b)* is the impurity function

For $f(a,b) = a\log_2(a)+b\log_2(b)$ we have info gain, for $f(a,b) = 2ab$ we have a so called Gini criterion. For $f(a,b)=\sqrt{ab}$ we have a criterion known as DKM. It is known as the most stable of the three.

Data for learning the weather (play/don't play) concept (Witten p. 10)

Day	Outlook	Temp	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Ovcst	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Col	Normal	Weak	yes
6	Rain	Cool	Normal	Strong	No
7	Ovcst	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Ovcst	Mild	High	Strong	Yes
13	Ovcst	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

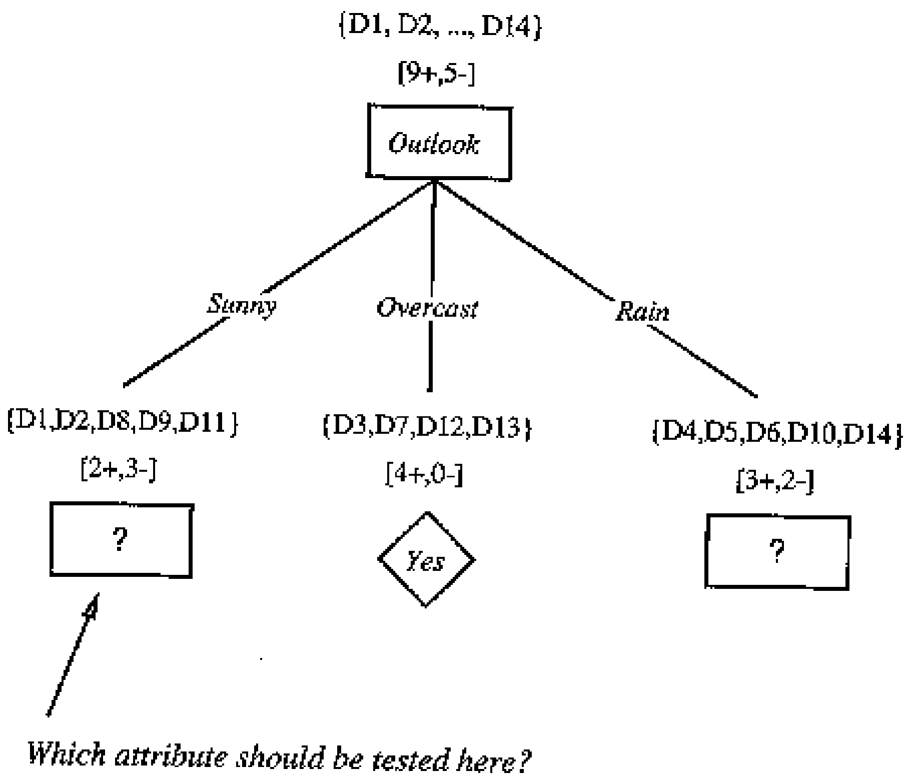
$$\text{Info}(S) = -(9/14)\log_2(9/14)-(5/14)\log_2 (5/14) = 0.940$$

Selecting the attribute

- Gain(S, Outlook) = 0.246
- Gain(S, Humidity) = 0.151
- Gain(S, Wind) = 0.048
- Gain(S, Temp) = 0.029

- Choose Outlook as the top test

How does info gain work?



S_{sunny} = {D1,D2,D8,D9,D11}

$Gain(S_{sunny}, Humidity)$ = .970 - (3/5) 0.0 - (2/5) 0.0 = .970

$Gain(S_{sunny}, Temperature)$ = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570

$Gain(S_{sunny}, Wind)$ = .970 - (2/5) 1.0 - (3/5) .918 = .019

See on p. 27

Gain ratio

- info gain favours tests with many outcomes (patient id example)
- consider split info(X) = $\sum |T_i|/|T|*\log(|T_i|/|T|)$

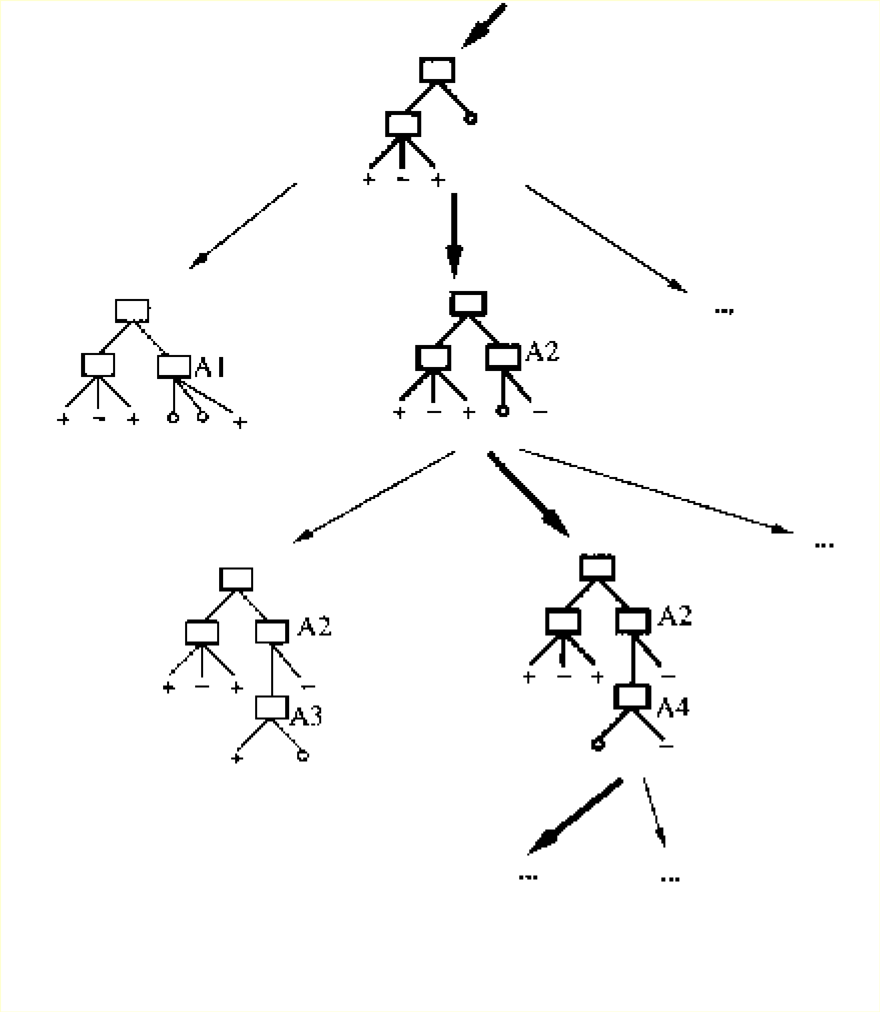
measures potential info. generated by dividing T into n classes (without considering the class info)

gain ratio(X) = gain(X)/split info(X)

shows the proportion of info generated by the split that is useful for classification: in the example (Witten p. 96), log(k)/log(n)

maximize gain ratio

In fact, learning DTs with the gain ratio heuristic is a search:



- Hill-climbing search
- Info gain is the search heuristic
- Covering the examples is the search criterion
- Inductive bias: sorter trees are preferred