Data Mining: Foundation, Techniques and Applications

Lesson 1b :A Quick Overview of Data Mining

Li Cuiping(李翠平)
School of Information
Renmin University of China

Anthony Tung(鄧锦浩)
School of Computing
National University of Singapore

Why a quick overview ?

- Have a look at the general techniques before looking at the foundation.
  - Machine Learning & Statistics
  - Indexing
  - Pre-Computation
- Easier to explain and see how these foundation support the various techniques

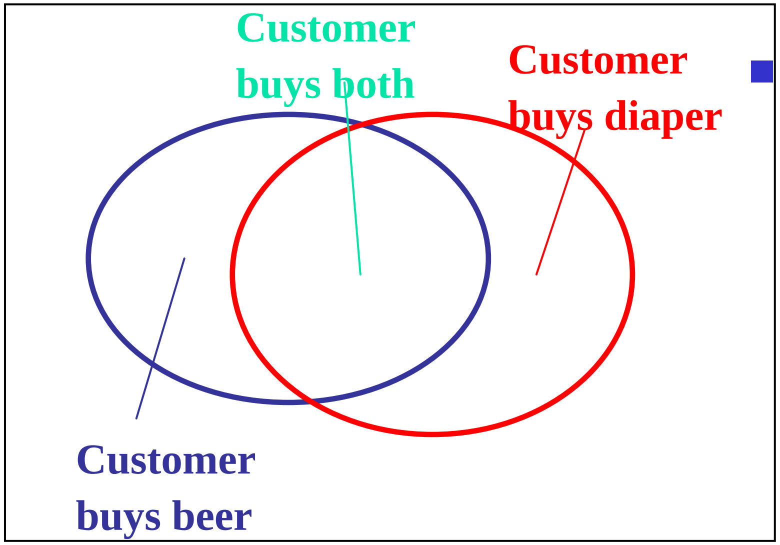
Outline

- Association Rules
  - The Apriori Algorithm
- Clustering
  - Partitioning
  - Density Based
- Classification
  - Decision Trees
- A General Framework for DM

Association Rule: Basic Concepts

- Given: (1) database of transactions, (2) each transaction is a list of items (purchased by a customer in a visit)
- Find: all rules that correlate the presence of one set of items with that of another set of items
  - E.g., *98% of people who purchase tires and auto accessories also get automotive services done*
- Applications
  - ** $\Rightarrow$ Maintenance Agreement* (What the store should do to boost Maintenance Agreement sales)
  - *Home Electronics $\Rightarrow$ ** (What other products should the store stocks up?)
  - Attached mailing in direct marketing

Rule Measures: Support and Confidence



- Find all the rules $X \,\&\, Y \Rightarrow Z$ with minimum confidence and support
  - support, s, probability that a transaction contains {X ^ Y ^ Z}
  - confidence, c, conditional probability that a transaction having {X ^ Y} also contains Z

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Let minimum support 50%, and minimum confidence 50%, we have

$A \Rightarrow C$ (50%, 66.6%)

$C \Rightarrow A$ (50%, 100%)

Mining Frequent Itemsets: the Key Step

- Find the *frequent itemsets*: the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - i.e., if {a b} is a frequent itemset, both {a} and {b} should be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to *k* (*k*-itemset)
- Use the frequent itemsets to generate association rules.

The Apriori Algorithm

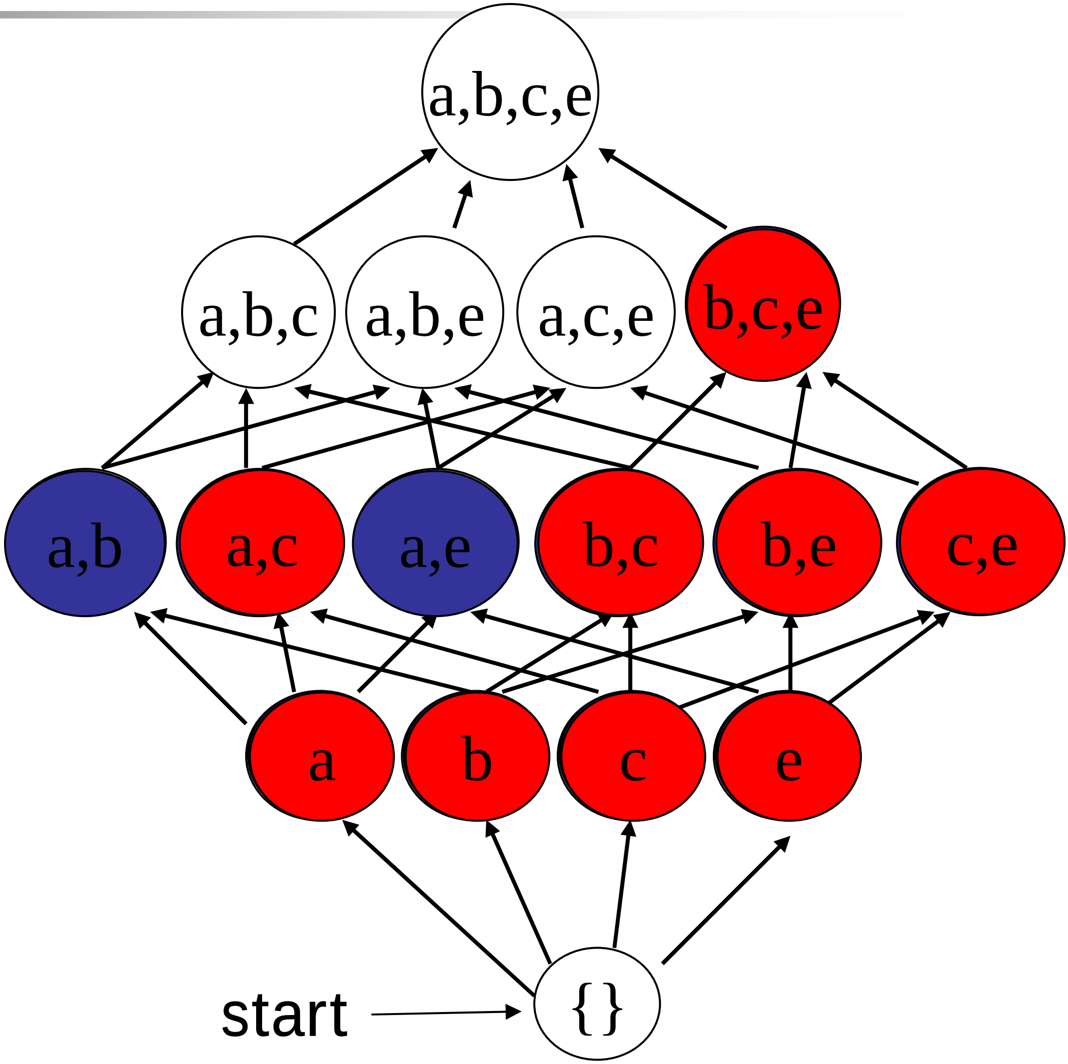
- **Join Step**: C_k is generated by joining L_{k-1} with itself
- **Prune Step**: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset
- Pseudo-code:

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

```
L_1 = {frequent items};
for (k = 1; L_k != ∅; k++) do begin
    C_k+1 = candidates generated from L_k;
    for each transaction t in database do
        increment the count of all candidates in C_k+1 that are contained in t
    L_k+1 = candidates in C_k+1 with min_support
    end
return ∪_k L_k;
```

The Apriori Algorithm

- Bottom-up, breadth first search
- Only read is perform on the databases
- Store candidates in memory to simulate the lattice search
- Iteratively follow the two steps:
  - generate candidates
  - count and get actual frequent items



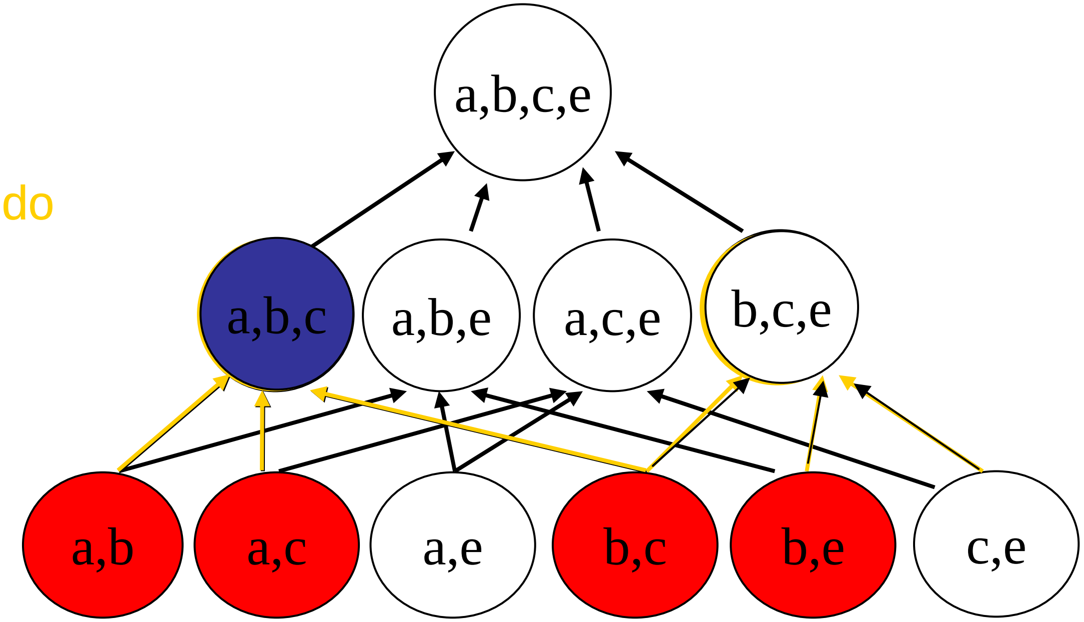
Candidate Generation and Pruning

- Suppose all frequent (k-1) items are in L_{k-1}
- Step 1: Self-joining L_{k-1}

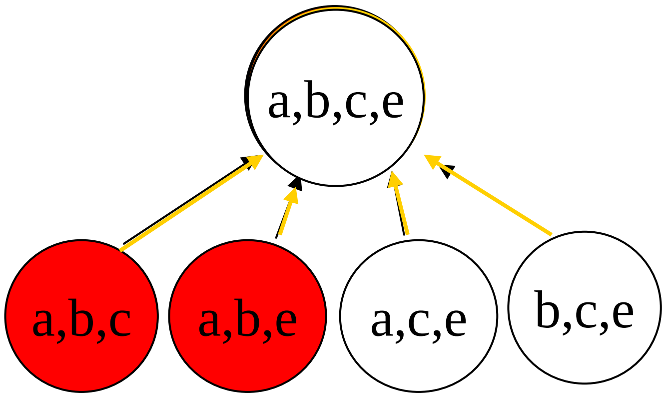
  insert into $\boldsymbol{C_k}$
  select $\boldsymbol{p.i_1, p.i_2, \ldots, p.i_{k-1}, q.i_{k-1}}$
  from $\boldsymbol{L_{k-1}\ p,\ L_{k-1}\ q}$
  where $\boldsymbol{p.i_1=q.i_1}$, … , $\boldsymbol{p.i_{k-2}=q.i_{k-2}}$, $\boldsymbol{p.i_{k-1} < q.i_{k-1}}$

- Step 2: pruning

  forall ***itemsets c in*** $\boldsymbol{C_k}$ do
  forall ***(k-1)-subsets s of c*** do
  if *(s is not in* L_{k-1}*)* **then**
  delete *c* **from** C_k



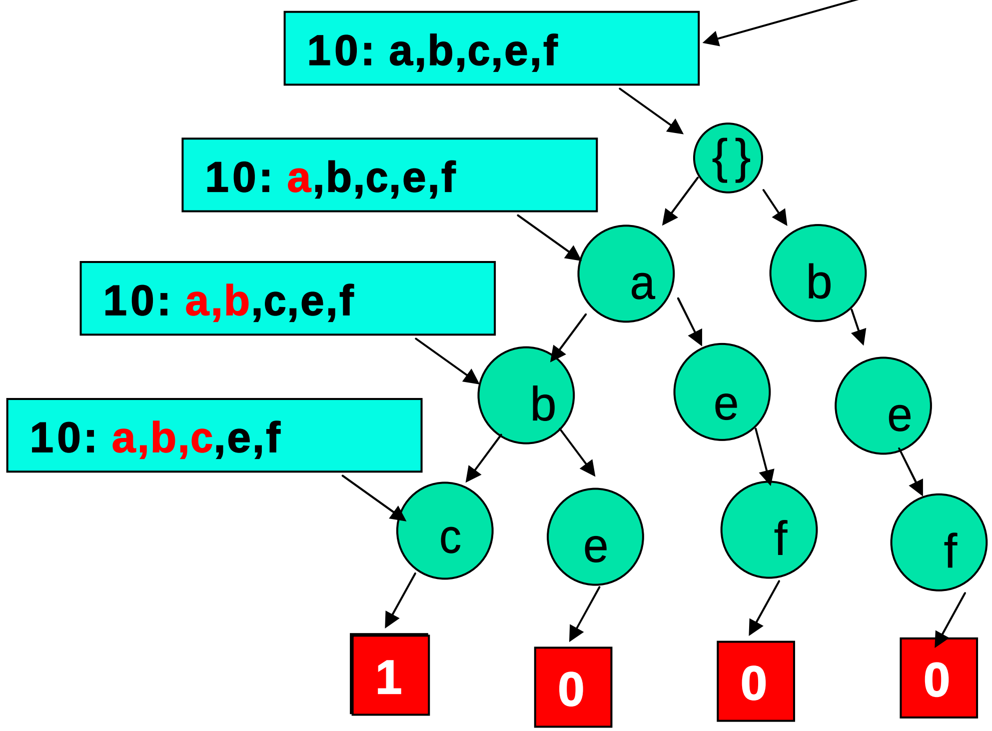
Candidate Generation and Pruning(another example)



Counting Supports

- Stored candidates itemsets in a trier structure for support counting

TID	Items
10	a, b, c, e, f
20	b, c, e
30	b, f, g
40	b, e
.	.
.	.



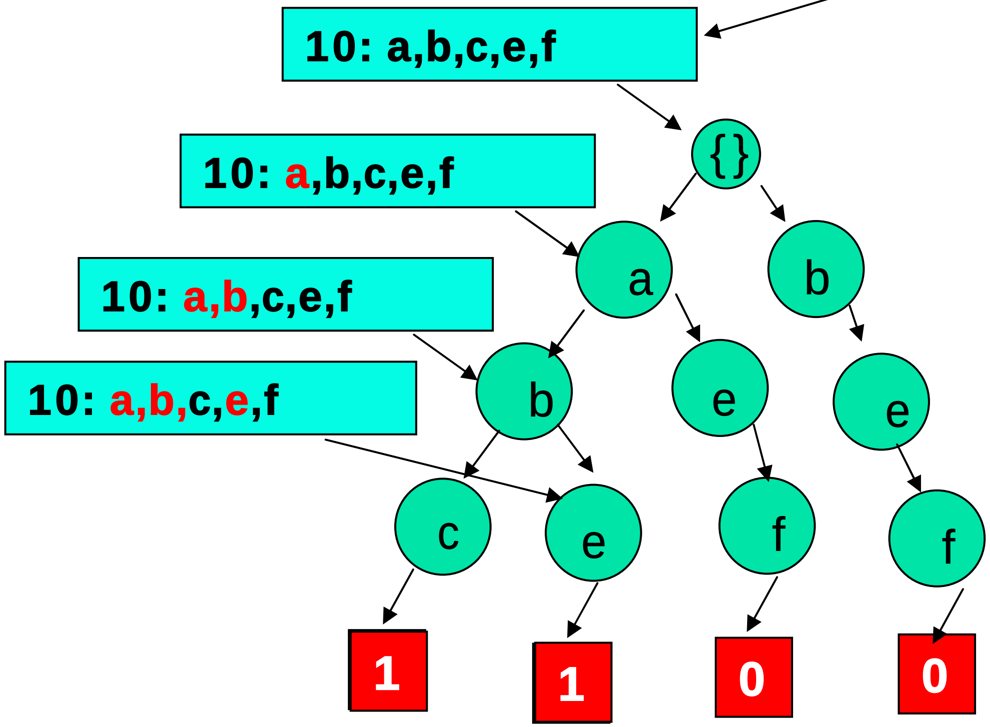
Candidate Itemsets:

{a,b,c}, {a,b,e}, {a,e,f}, {b,e,f}

Counting Supports

- Stored candidates itemsets in a trier structure for support counting

TID	Items
10	a, b, c, e, f
20	b, c, e
30	b, f, g
40	b, e
.	.
.	.



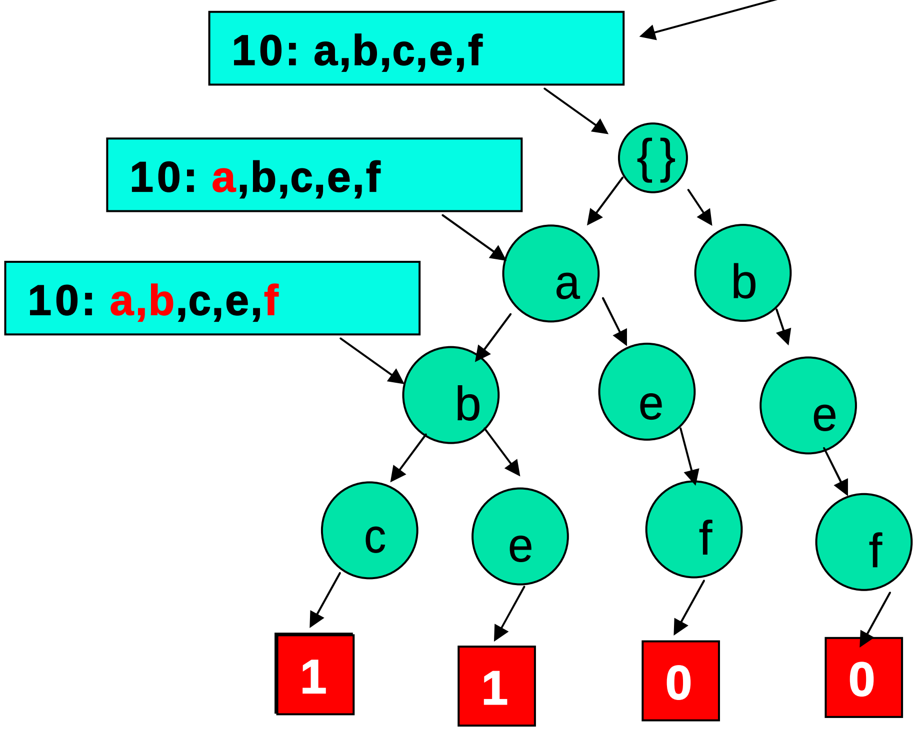
Candidate Itemsets:

{a,b,c}, {a,b,e}, {a,e,f}, {b,e,f}

Counting Supports

- Stored candidates itemsets in a trier structure for support counting

TID	Items
10	a, b, c, e, f
20	b, c, e
30	b, f, g
40	b, e
:	:



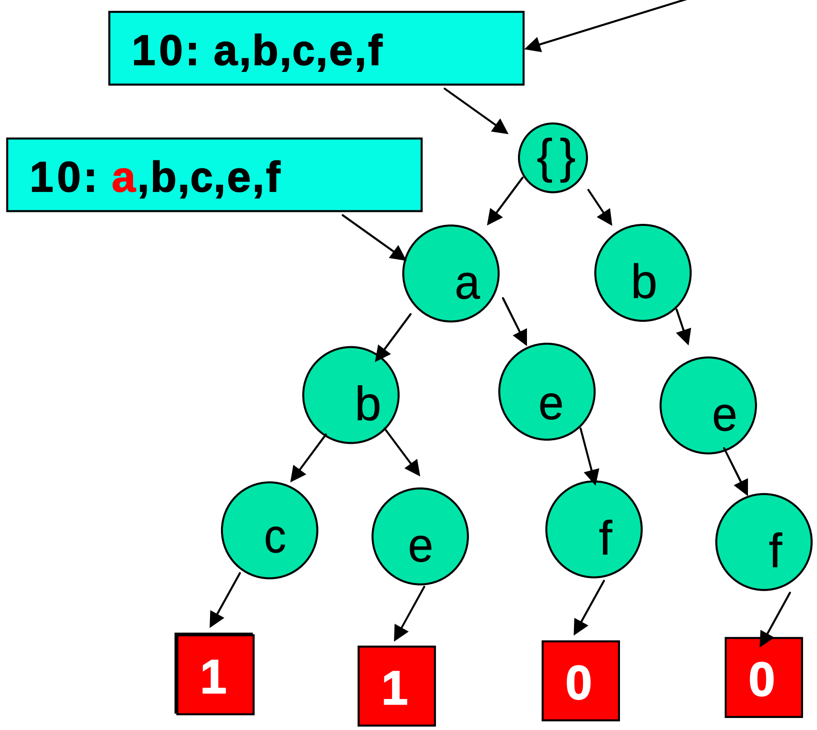
Candidate Itemsets:

{a,b,c}, {a,b,e}, {a,e,f}, {b,e,f}

Counting Supports

- Stored candidates itemsets in a trier structure for support counting

TID	Items
10	a, b, c, e, f
20	b, c, e
30	b, f, g
40	b, e
:	:



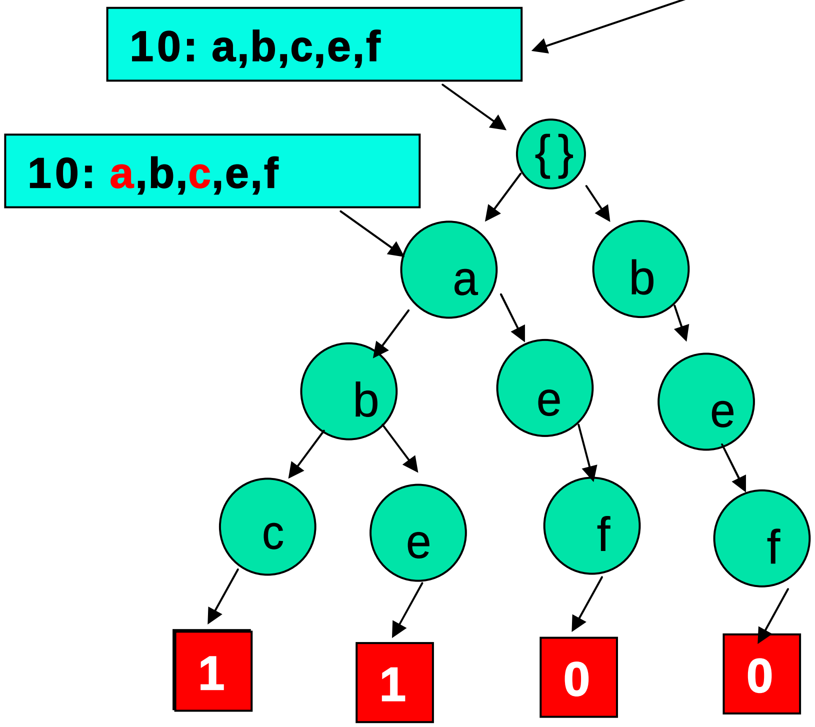
Candidate Itemsets:

{a,b,c}, {a,b,e}, {a,e,f}, {b,e,f}

Counting Supports

- Stored candidates itemsets in a trier structure for support counting

TID	Items
10	a, b, c, e, f
20	b, c, e
30	b, f, g
40	b, e
.	.
.	.



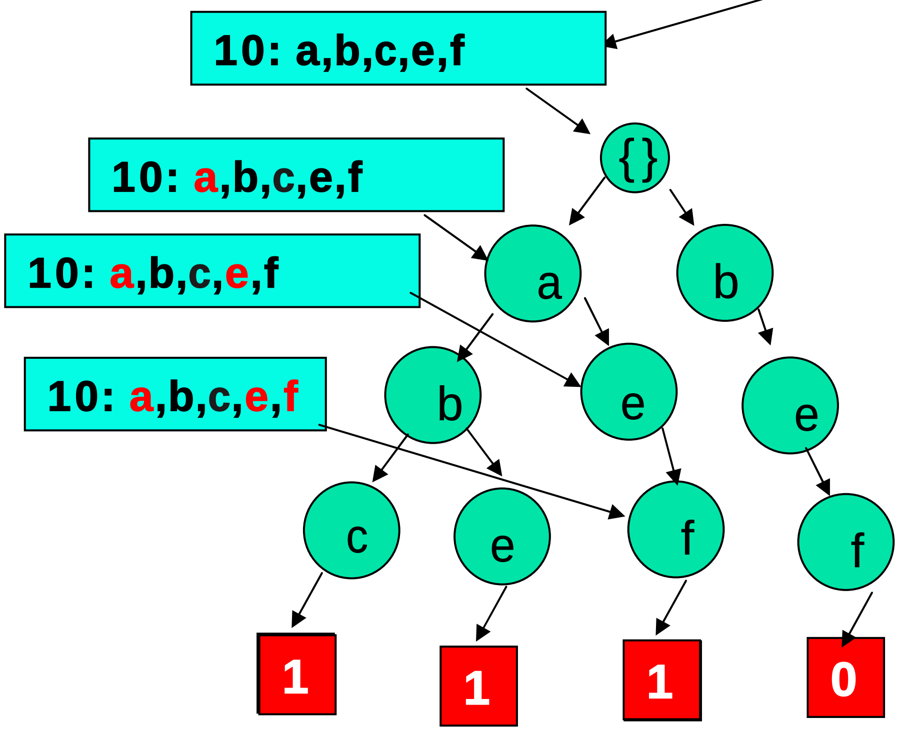
Candidate Itemsets:

{a,b,c}, {a,b,e}, {a,e,f}, {b,e,f}

Counting Supports

•Stored candidates itemsets in a trier structure for support counting

TID	Items
10	a, b, c, e, f
20	b, c, e
30	b, f, g
40	b, e
⋮	⋮



Candidate Itemsets:

{a,b,c}, {a,b,e}, {a,e,f}, {b,e,f}

Rules Generation

- Since the support of all the frequent itemsets are known, it is possible to derive all rules that satisfied the MINCONF threshold by making use of the support computed.
- Eg. If supp({a,b,c,d})=20 and supp({a,b})=50 then confidence for the rule {a,b=>c,d} is 20/50 = 40%.

Apriori Algorithm

- A level-wise, candidate-generation-and-test approach (Agrawal & Srikant 1994)

Database D

TID	Items
10	a, c, d
20	b, c, e
30	a, b, c, e
40	b, e

Min_sup=2

Scan D

1-candidates

Itemset	Sup
a	2
b	3
c	3
d	1
e	3

Freq 1-itemsets

Itemset	Sup
a	2
b	3
c	3
e	3

2-candidates

Itemset
ab
ac
ae
bc
be
ce

Scan D

Counting

Itemset	Sup
ab	1
ac	2
ae	1
bc	2
be	3
ce	2

Freq 2-itemsets

Itemset	Sup
ac	2
bc	2
be	3
ce	2

3-candidates

Itemset
bce

Scan D

Freq 3-itemsets

Itemset	Sup
bce	2

Outline

- Association Rules
  - The Apriori Algorithm
- Clustering
  - Partitioning
  - Density Based
- Classification
  - Decision Trees
- A General Framework for DM

What is Cluster Analysis?

- Cluster: a collection of data objects
  - Similar to one another within the same cluster
  - Dissimilar to the objects in other clusters
- Cluster analysis
  - Grouping a set of data objects into clusters
- Clustering is unsupervised classification: no predefined classes
- Typical applications
  - As a stand-alone tool to get insight into data distribution
  - As a preprocessing step for other algorithms

What Is Good Clustering?

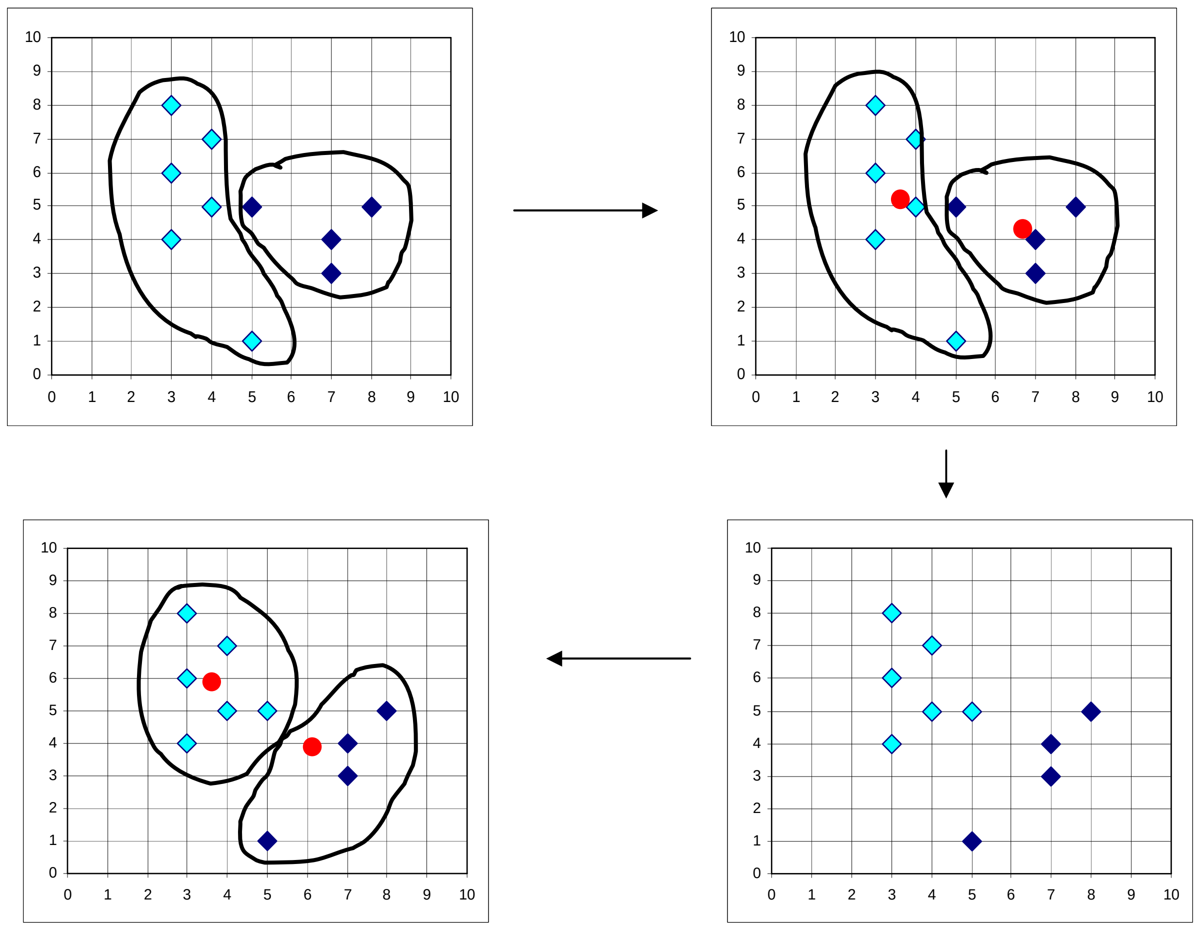
- A <u>good clustering</u> method will produce high quality clusters with
  - high <u>intra-class</u> similarity
  - low <u>inter-class</u> similarity
- The <u>quality</u> of a clustering result depends on both the similarity measure used by the method and its implementation.
- The <u>quality</u> of a clustering method is also measured by its ability to discover some or all of the <u>hidden patterns</u>.

The *K-Means* Clustering Method

- Given *k*, the *k-means* algorithm is implemented in 4 steps:
  - Partition objects into *k* nonempty subsets
  - Compute seed points as the centroids of the clusters of the current partition. The centroid is the center (mean point) of the cluster.
  - Assign each object to the cluster with the nearest seed point.
  - Go back to Step 2, stop when no more new assignment.

The *K-Means* Clustering Method

- Example



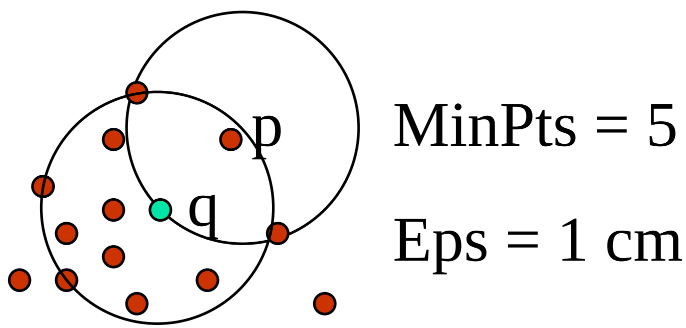
Density-Based Clustering Methods

- Clustering based on density (local cluster criterion), such as density-connected points
- Major features:
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters as termination condition
- Several interesting studies:
  - <u>DBSCAN:</u> Ester, et al. (KDD'96)
  - <u>OPTICS</u>: Ankerst, et al (SIGMOD'99).
  - <u>DENCLUE</u>: Hinneburg & D. Keim (KDD'98)
  - <u>CLIQUE</u>: Agrawal, et al. (SIGMOD'98)

Density-Based Clustering: Background

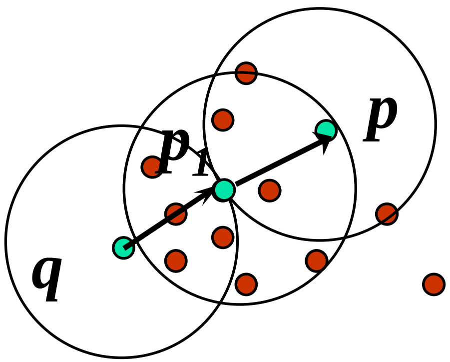
- Two parameters:
  - ***Eps***: Maximum radius of the neighbourhood
  - ***MinPts***: Minimum number of points in an Eps-neighbourhood of that point
- ***$N_{Eps}(p)$***: ***{q belongs to D | dist(p,q) <= Eps}***
- Directly density-reachable**:** A point ***p*** is directly density-reachable from a point ***q*** wrt. ***Eps***, ***MinPts*** if
  - 1) ***p*** belongs to ***$N_{Eps}(q)$***
  - 2) core point condition:

$$|N_{Eps}(q)| >= MinPts$$

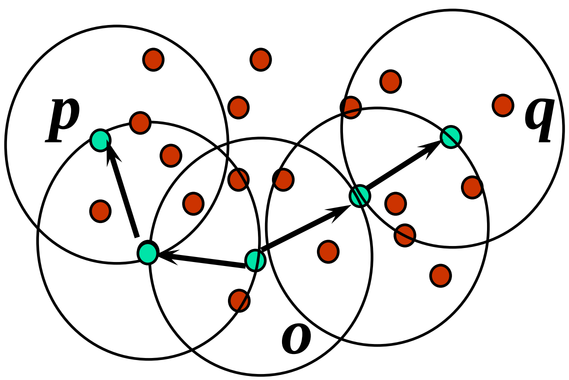


Density-Based Clustering: Background (II)

- Density-reachable:
  - A point p is density-reachable from a point q wrt. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i

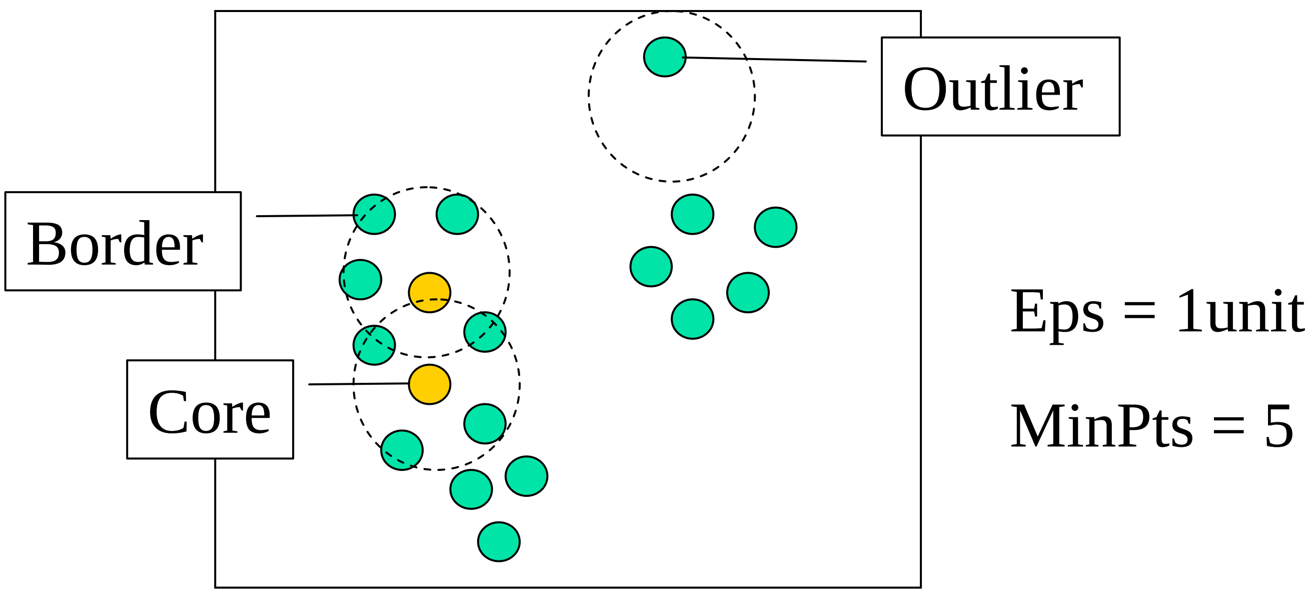


- Density-connected
  - A point p is density-connected to a point q wrt. *Eps*, *MinPts* if there is a point o such that both, p and q are density-reachable from o wrt. *Eps* and *MinPts*.



DBSCAN: Density Based Spatial Clustering of Applications with Noise

- Relies on a *density-based* notion of cluster: A *cluster* is defined as a maximal set of density-connected points
- Discovers clusters of arbitrary shape in spatial databases with noise



DBSCAN: The Algorithm

- Arbitrary select a point ***p***
- Retrieve all points density-reachable from ***p*** wrt ***Eps*** and ***MinPts***.
- If ***p*** is a core point, a cluster is formed.
- If ***p*** is a border point, no points are density-reachable from ***p*** and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.

Outline

- Association Rules
  - The Apriori Algorithm
- Clustering
  - Partitioning
  - Density Based
- Classification
  - Decision Trees
- A General Framework for DM

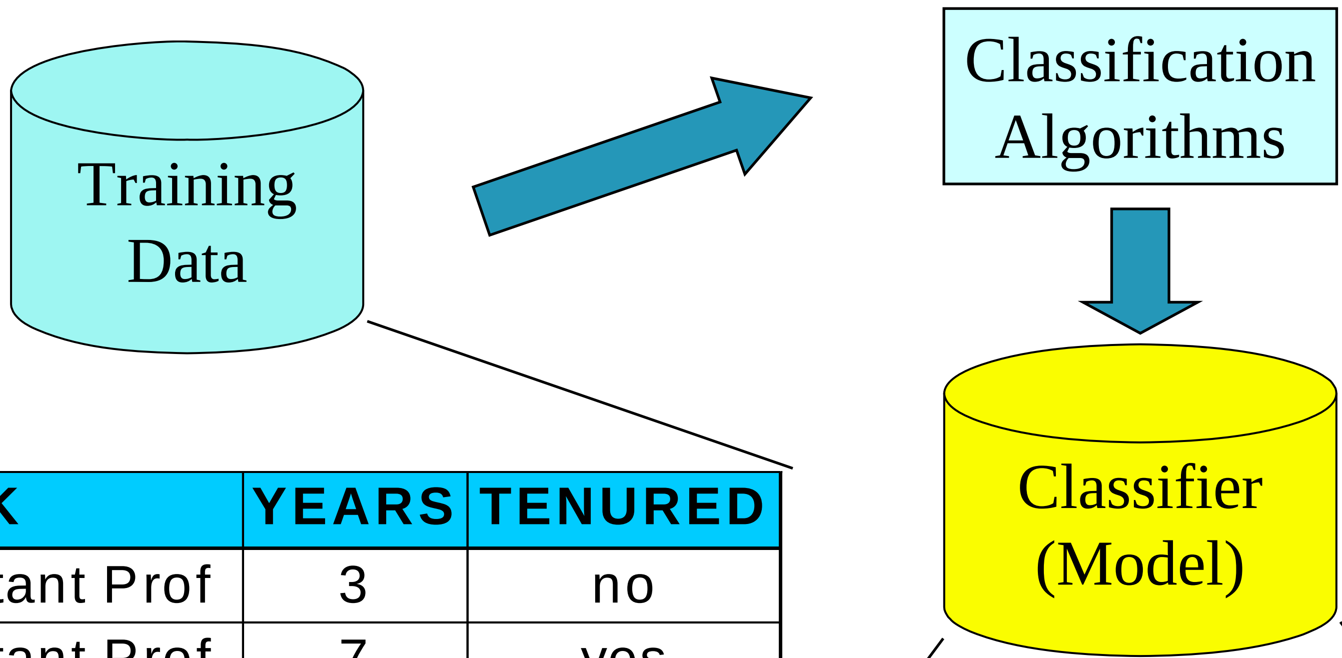
What is Classification ?

- Classification:
  - predicts categorical class labels
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Typical Applications
  - credit approval
  - target marketing
  - medical diagnosis
  - treatment effectiveness analysis

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction: training set
  - The model is represented as classification rules, decision trees, or mathematical formulae
- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set, otherwise over-fitting will occur

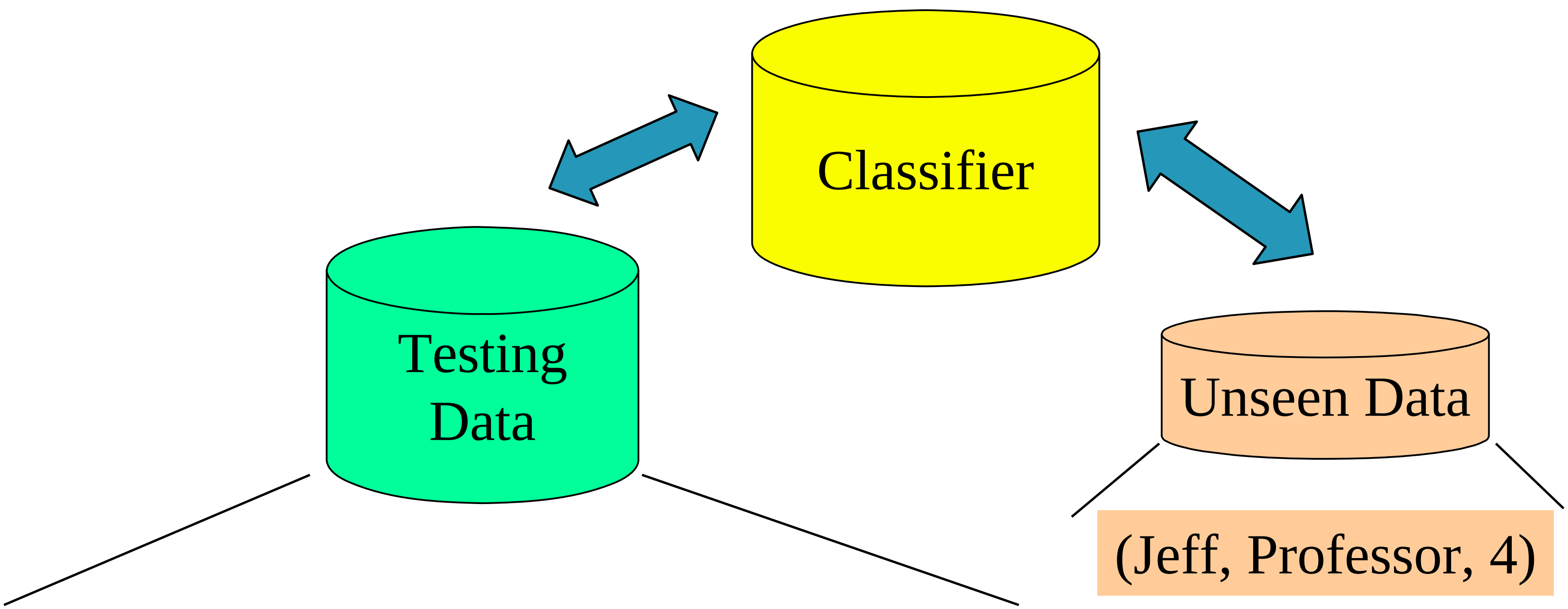
Classification Process (1): Model Construction



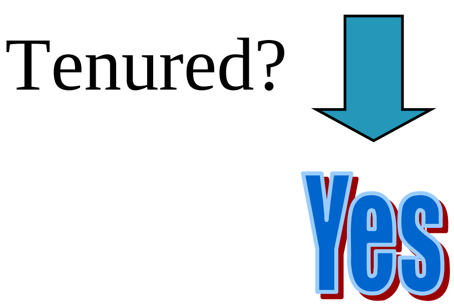
NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = ‘professor’
OR years > 6
THEN tenured = ‘yes’

Classification Process (2): Use the Model in Prediction



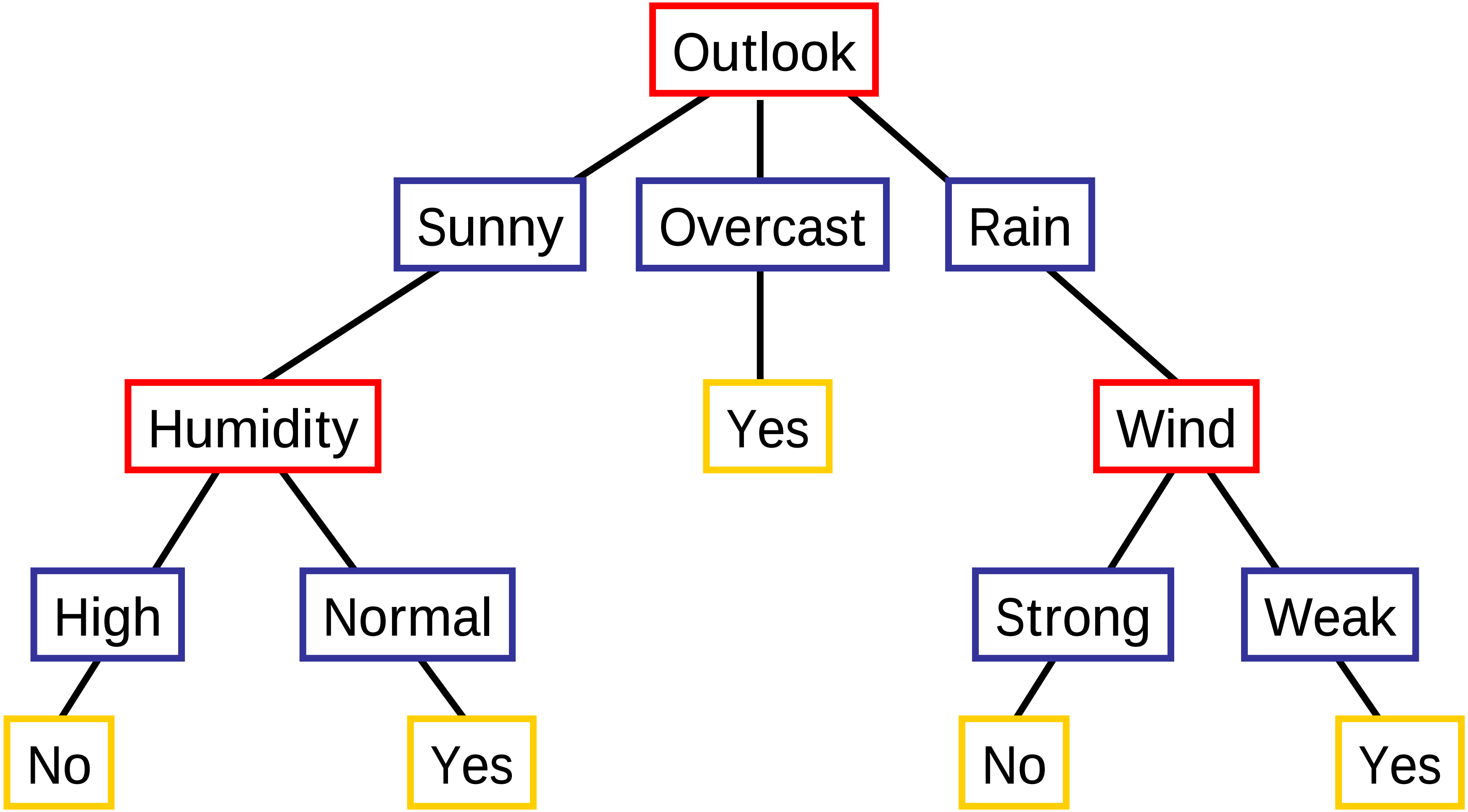
NAME	RANK	YEARS	TENURED
Tom	Assistant Prof	2	no
Merlisa	Associate Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes



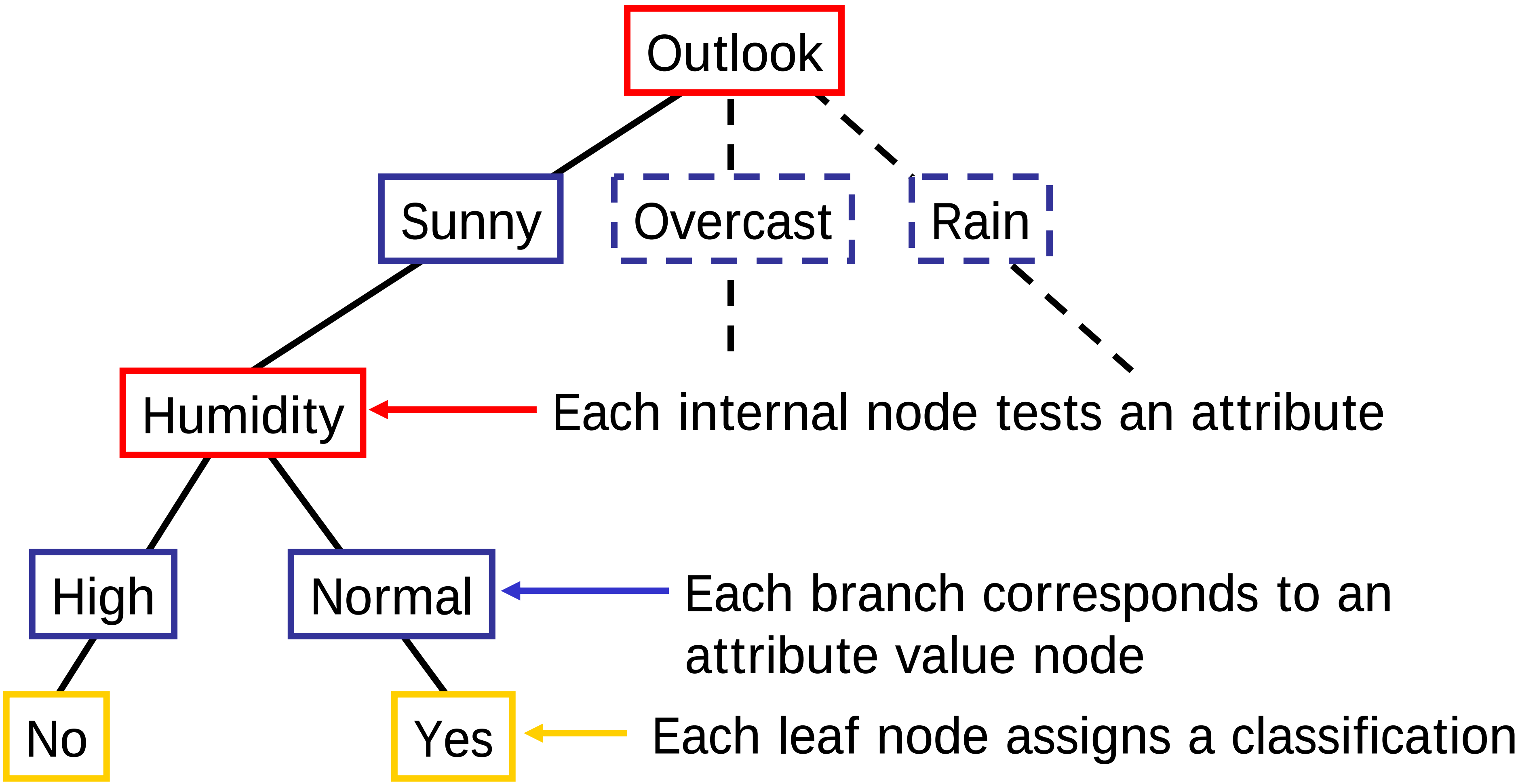
Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by labels indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

Decision Tree for PlayTennis



Decision Tree for PlayTennis

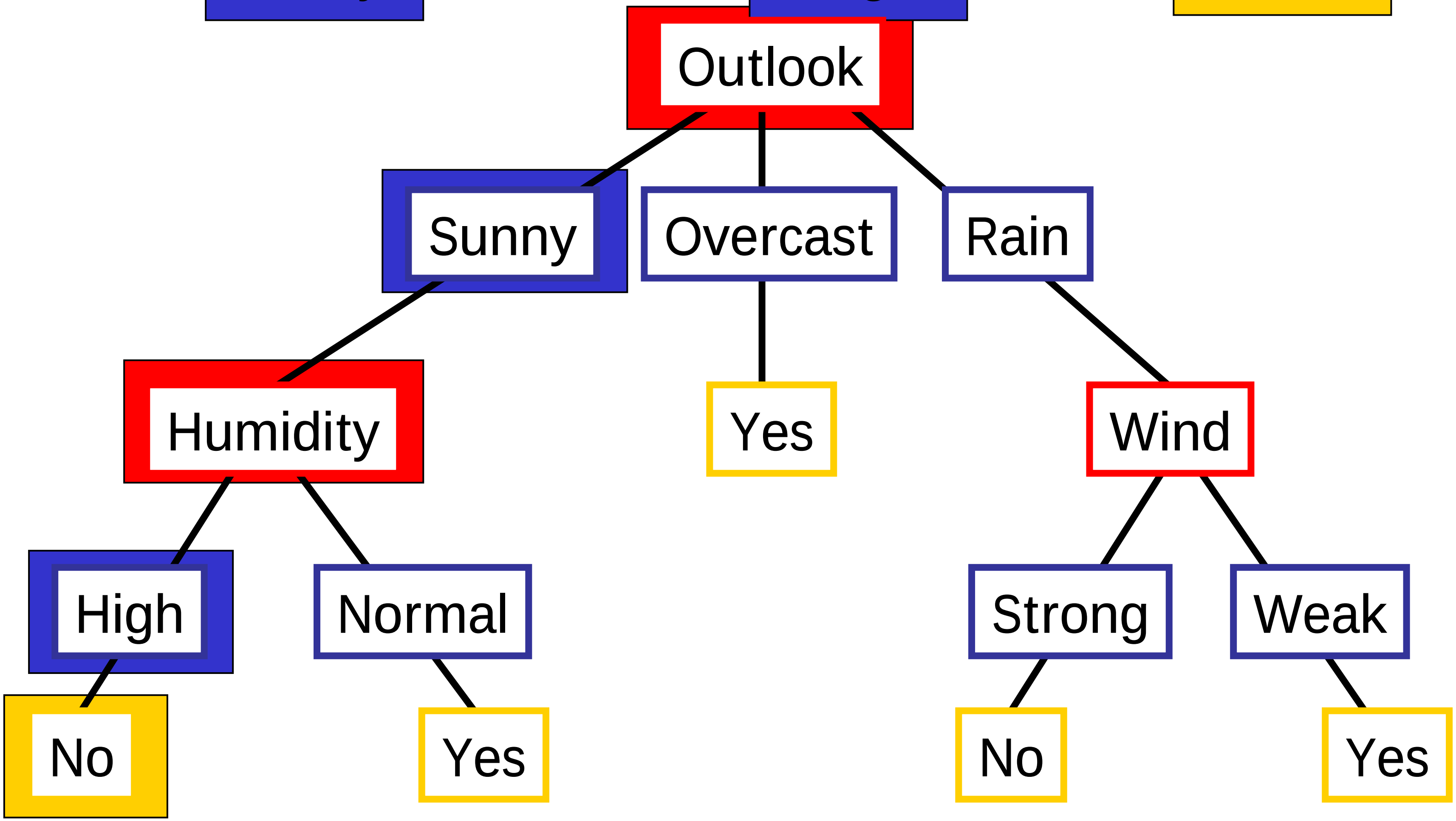


Training Dataset

Outlook	Temp	Humid	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Decision Tree for PlayTennis

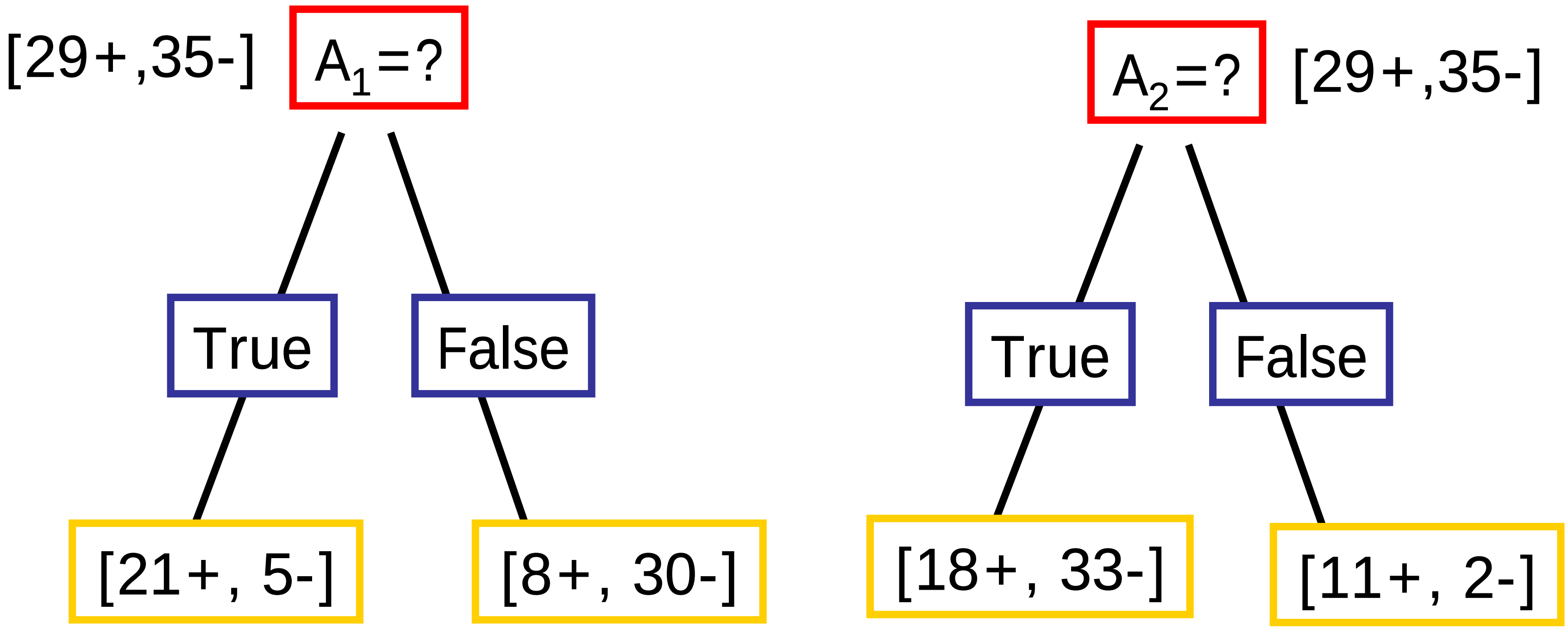
Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	?No



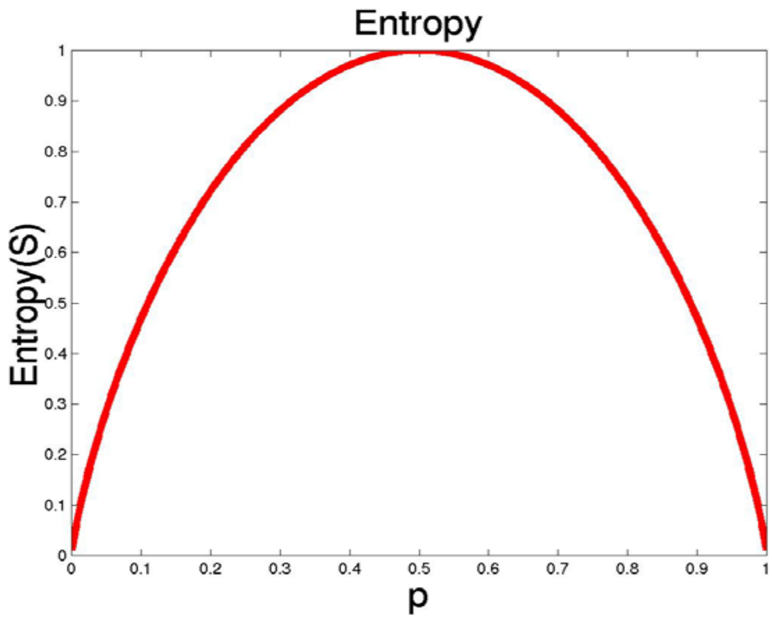
Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

Which Attribute is ”best”?



Entropy



- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Entropy

- Entropy(S)= expected number of bits needed to encode class (+ or -) of randomly drawn members of S (under the optimal, shortest length-code)

Why?

- Information theory optimal length code assign

  $-\log_2 p$ bits to messages having probability p.

- So the expected number of bits to encode

  (+ or -) of random member of S:

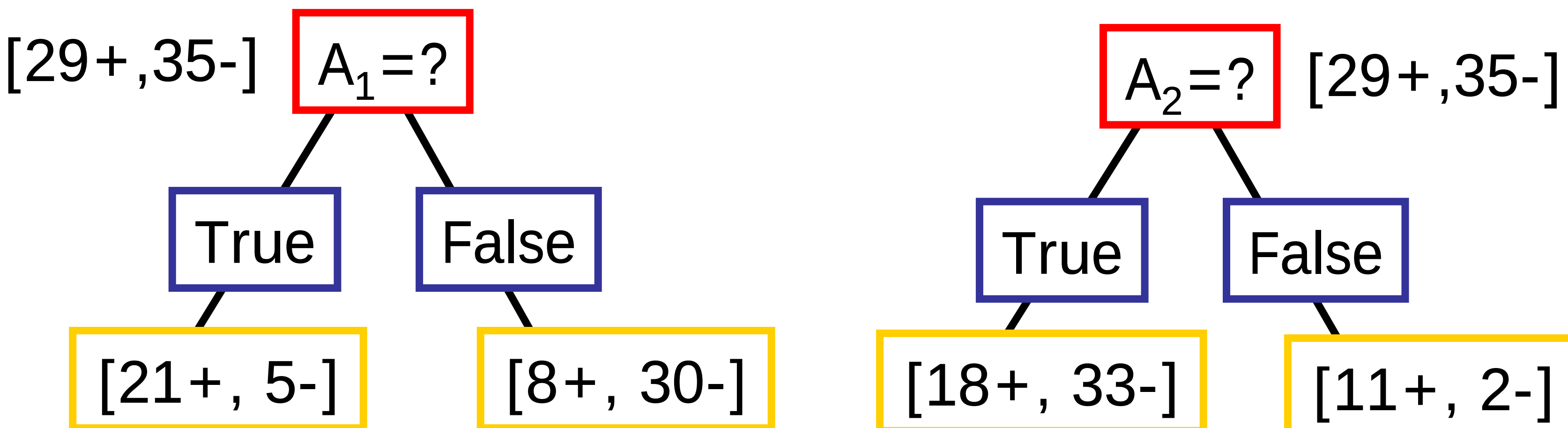
$$-p_+ \log_2 p_+ - p_- \log_2 p_-$$

Information Gain

- Gain(S,A): expected reduction in entropy due to sorting S on attribute A

$$Gain(S,A) = Entropy(S) - \sum_{v \in values(A)} |S_v|/|S| \; Entropy(S_v)$$

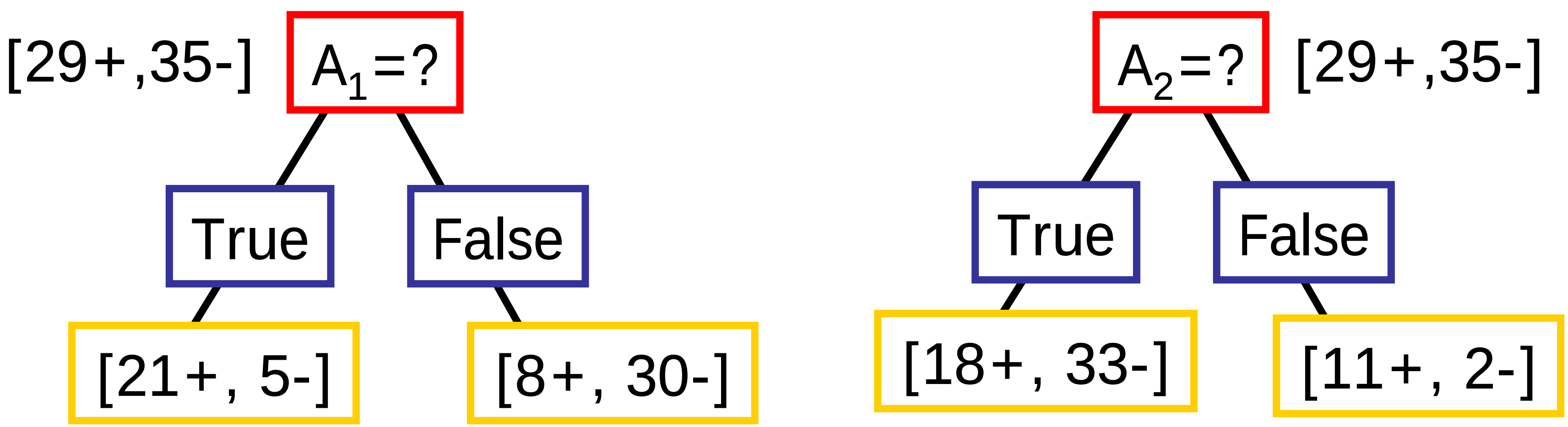
$$Entropy([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64$$
$$= 0.99$$



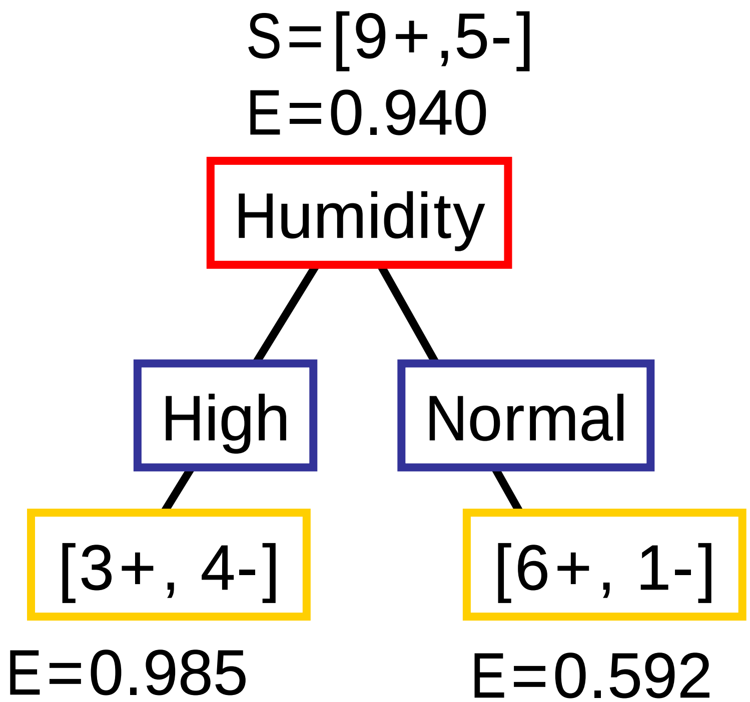
Information Gain

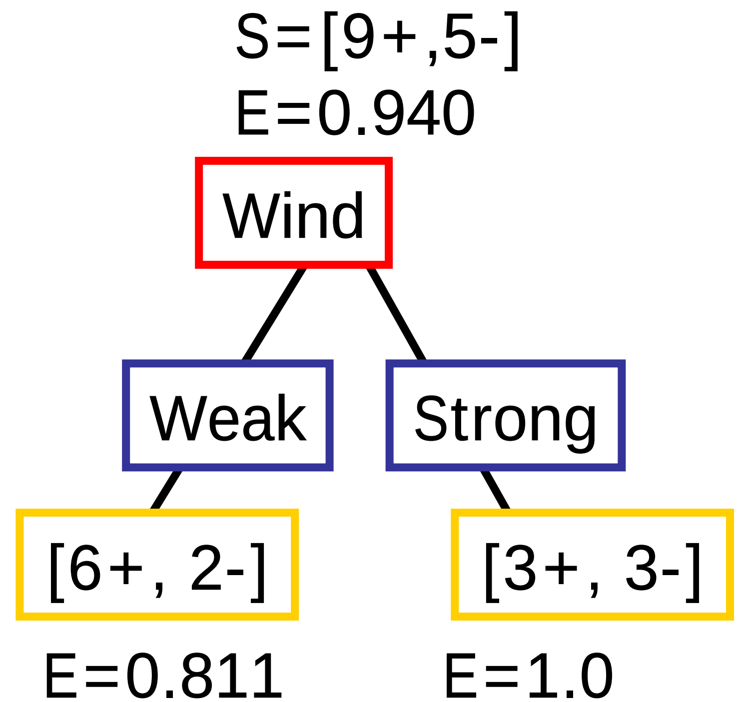
Entropy([21+,5-]) = 0.71
Entropy([8+,30-]) = 0.74
Gain(S,A_1)=Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
=0.27

Entropy([18+,33-]) = 0.94
Entropy([8+,30-]) = 0.62
Gain(S,A_2)=Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
=0.12



Selecting the Next Attribute

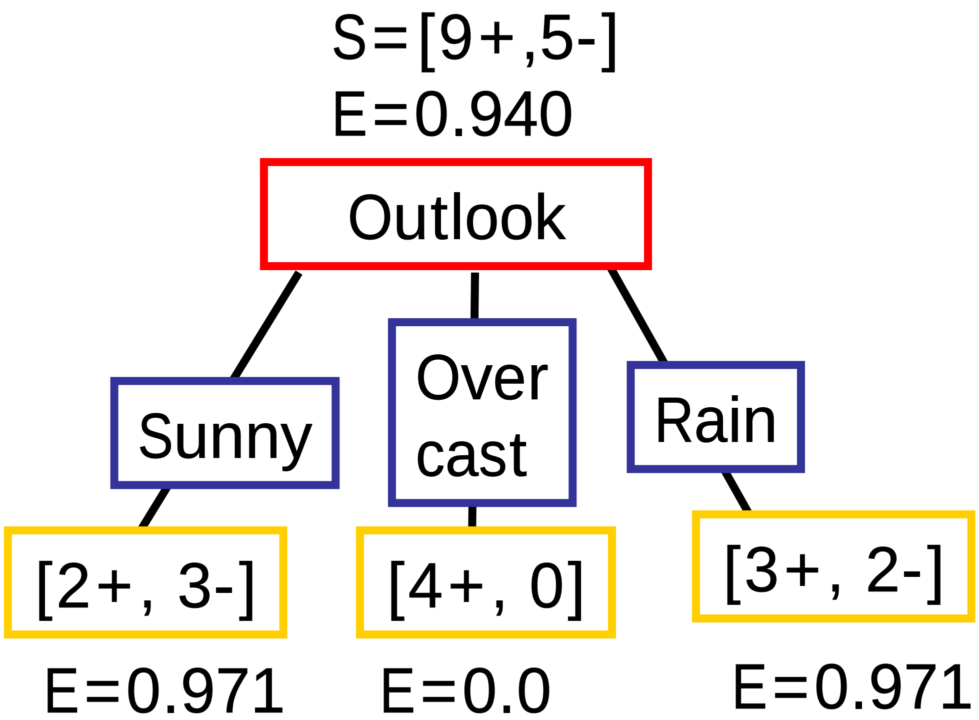




Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Selecting the Next Attribute



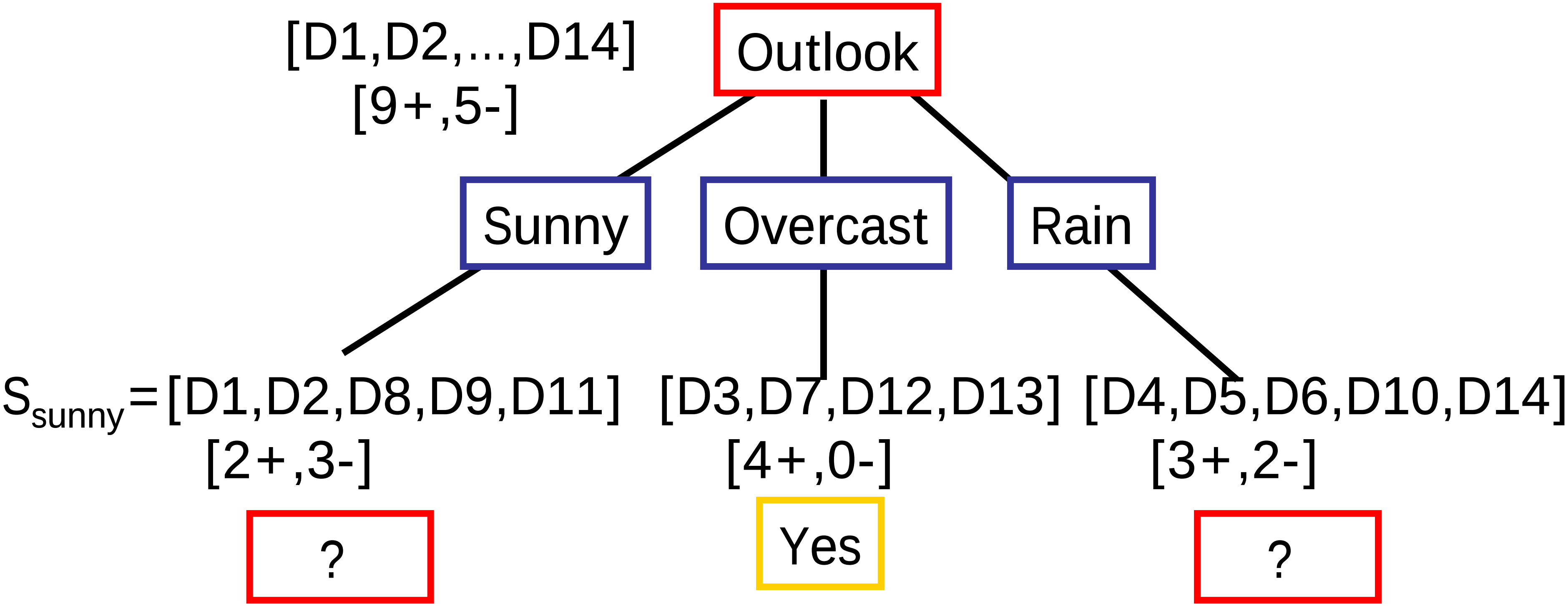
Gain(S,Outlook)

=0.940-(5/14)*0.971

-(4/14)*0.0 – (5/14)*0.0971

=0.247

ID3 Algorithm

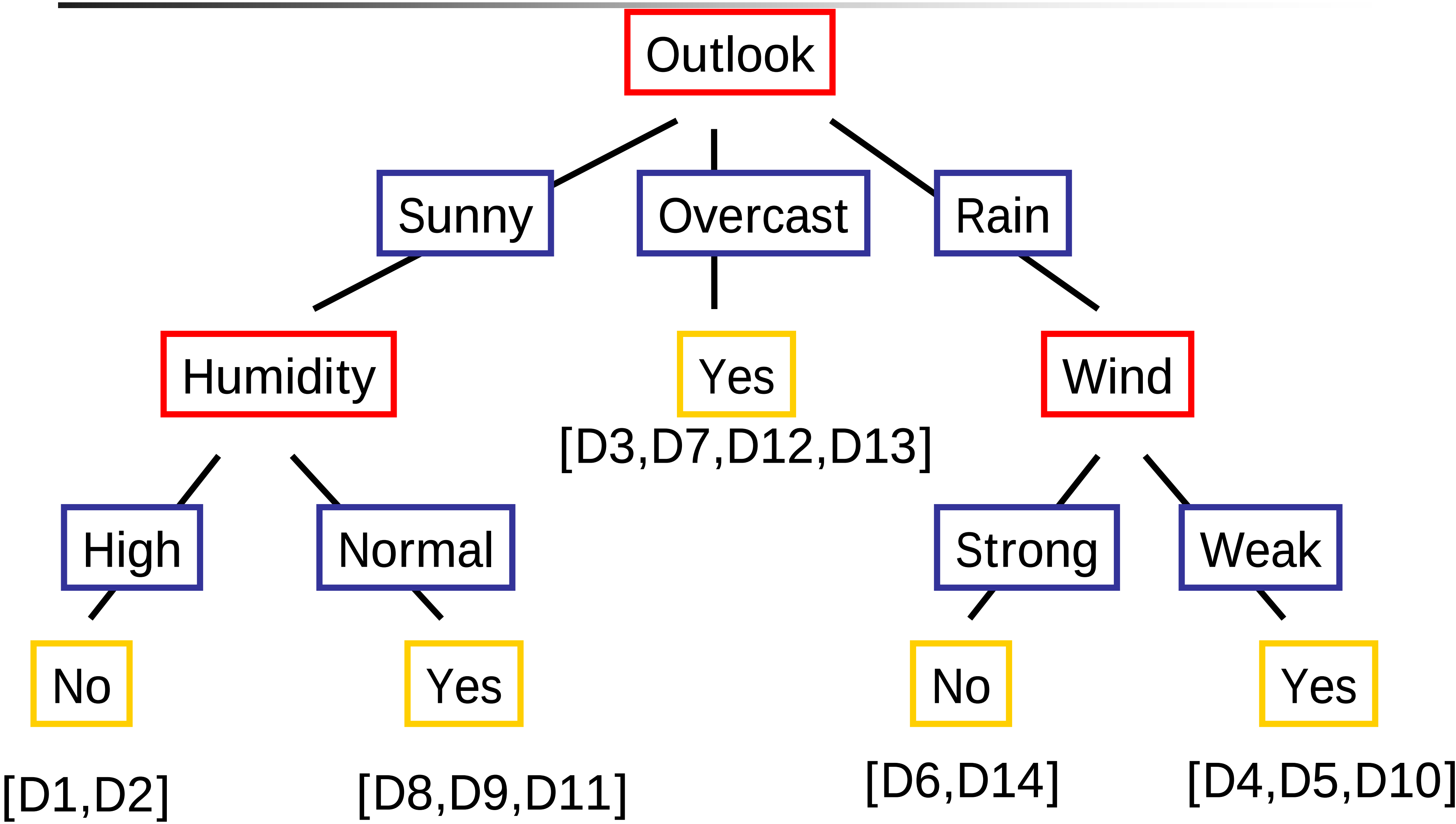


Gain(S_{sunny} , Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970

Gain(S_{sunny} , Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570

Gain(S_{sunny} , Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019

ID3 Algorithm



Avoid Overfitting in Classification

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples
- Two approaches to avoid overfitting
  - Prepruning: 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
  - Postpruning: Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

Outline

- Association Rules
  - The Apriori Algorithm
- Clustering
  - Partitioning
  - Density Based
- Classification
  - Decision Trees
- A General Framework for DM

A Generalized View of DM

1. The *task* the algorithm is used to address (e.g. classification, clustering, etc.)
2. The *structure* of the model or pattern we are fitting to the data (e.g. a linear regression model)
3. The *score function* used to judge the quality of the fitted models or patterns (e.g. accuracy, BIC, etc.)
4. The *search or optimization method* used to search over parameters and structures (e.g. steepest descent, MCMC, etc.)
5. The *data management technique* used for storing, indexing, and retrieving data (critical when data too large to reside in memory)

Can we fit what we learn into the framework?

	Apriori	K-means	ID3
task	rule pattern discovery	clustering	classification
structure of the model or pattern	association rules	clusters	decision tree
search space	lattice of all possible combination of items size= 2^m	choice of any k points as center size=infinity	all possible combination of decision tree size= potentially infinity
score function	support, confidence	square error	accuracy, information gain
search / optimization method	breadth first with pruning	gradient descent	greedy
data management technique	TBD	TBD	TBD