Chapter 18:

Data Mining

Database System Concepts, 7th Ed.

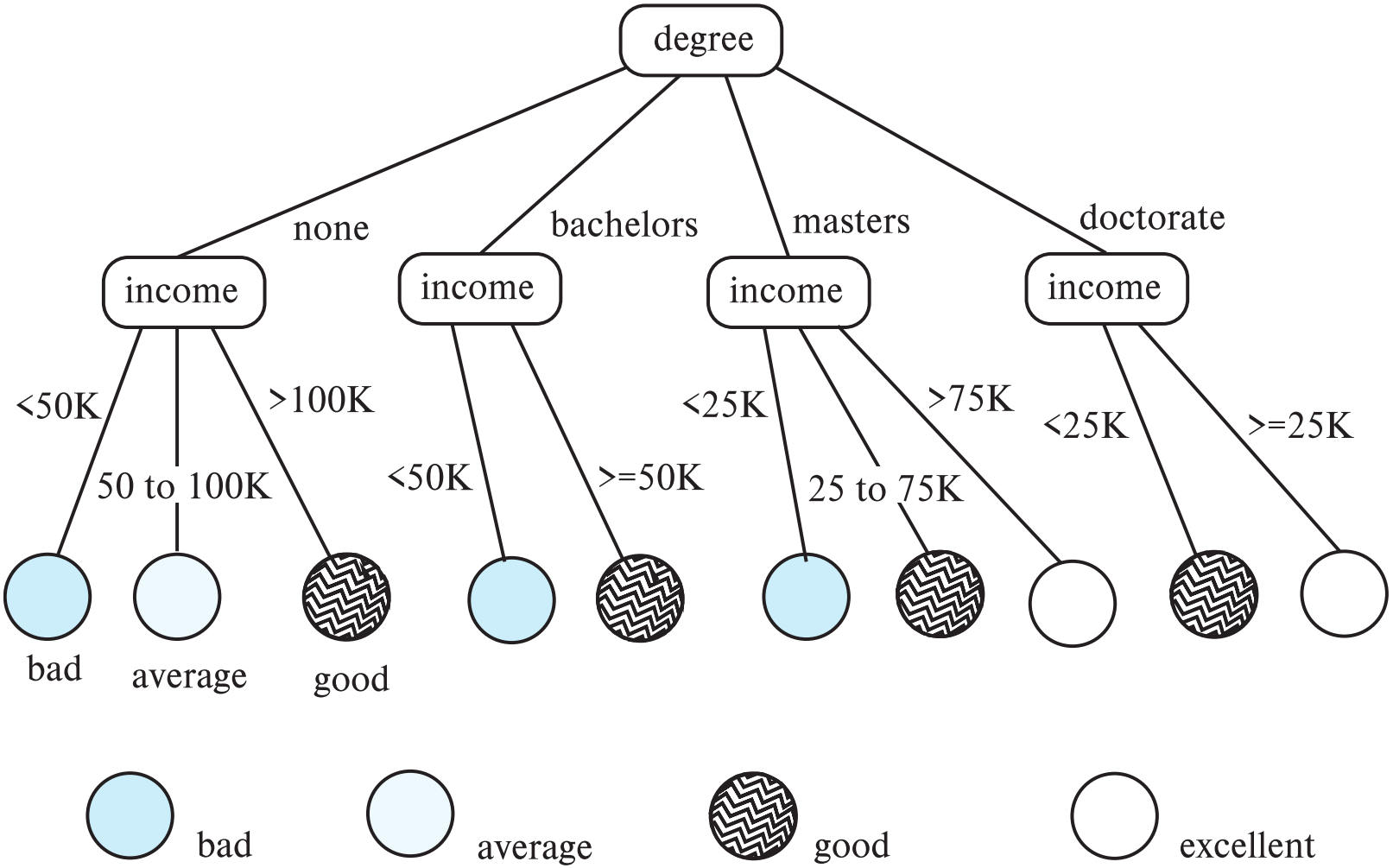
Data Mining

- **Data mining** is the process of semi-automatically analyzing large databases to find useful patterns
  - Similar goals to machine learning, but on very large volumes of data
- Also called **knowledge discovery in databases** (**KDD**)
- Some types of knowledge can be represented as rules
- More generally, knowledge is discovered by applying machine learning techniques on past instances of data, to form a **model**
  - Model is then used to make predictions for new instances

Types of Data Mining Tasks

- Examples of data mining tasks:
  - **Classification**
    - Items (with associated attributes) belong to one of several classes
    - **Training instances** have attribute values and classes provided
    - Given a new item whose class is unknown, predict to which class it belongs based on its attribute values
  - **Association**
    - Find books that are often bought by “similar” customers. If a new such customer buys one such book, suggest the others too.
    - Associations may be used as a first step in detecting **causation**
      - E.g., association between exposure to chemical X and cancer
  - **Clustering**
    - E.g., typhoid cases were clustered in an area surrounding a contaminated well
    - Detection of clusters remains important in detecting epidemics

Decision Tree Classifiers

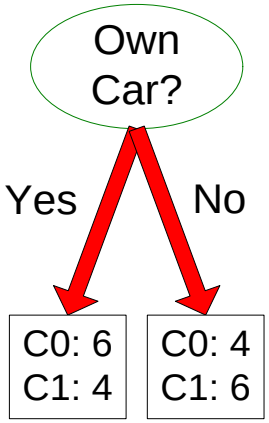


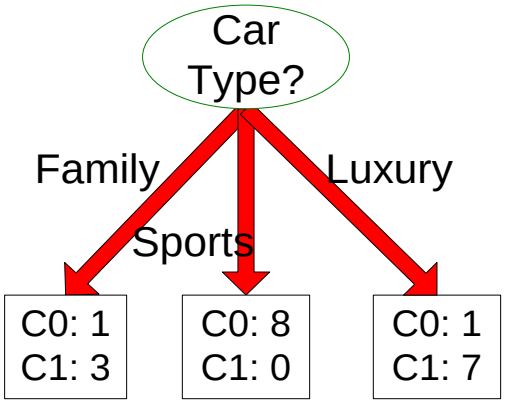
Decision Trees

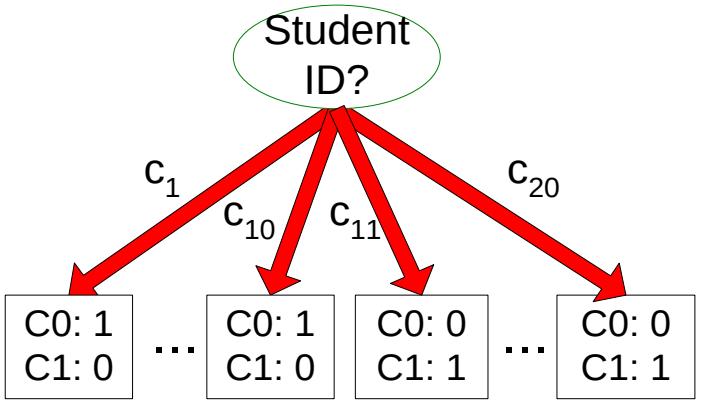
- Each internal node of the tree partitions the data into groups based on a **partitioning or splitting attribute**, and a **partitioning condition** for the node
- Node Purity:
  - all (or most) of the items at the node belong to the same class
- Traverse tree from top to make a prediction

Splitting

Before Splitting: 10 records of class 0 (C0),
10 records of class 1 (C1)







Which is the best for predicting credit worthiness?

How to Determine a Good Split

- Greedy approach:
  - Nodes with **homogeneous** or **pure** class distribution are preferred
- Need a measure of node impurity:

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous	Homogeneous
High degree of impurity	Low degree of impurity

Measure of Impurity: GINI

- Gini Index for a given node t,

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

where $p_i(t)$ is the relative frequency of class i at node t, and c is the total number of classes

- For a 2-class problem (p, q), where p represents the relative frequency of Class 1, and q = (1–p)
  - GINI = 1 – p^2 – (1 – p)^2 = 2p (1-p) = 2pq
- Maximum (1 – 1/c) when records are equally distributed among all classes, implying least beneficial situation for classification
- Minimum (0) when all records belong to one class, implying most beneficial situation for classification

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Computing Gini Index of a Single Node

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)^2 – P(C2)^2 = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)^2 – (5/6)^2 = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)^2 – (4/6)^2 = 0.444

Splitting Based on GINI

- When a node *p* is split into *k* partitions (children), the quality of split is computed as,

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

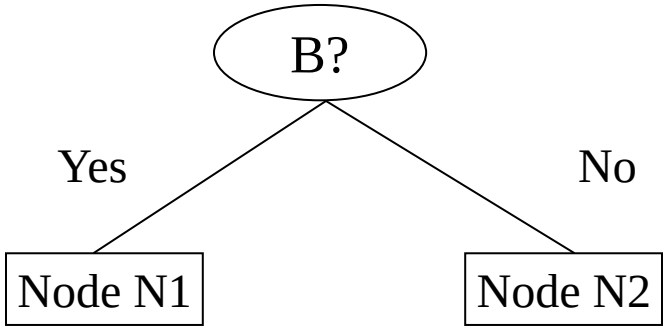
where, n_i = number of records at child *i*,

n = number of records at node *p*.

Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for – reduction in GINI

	Parent
C1	**6**
C2	**6**
Gini = 0.500	



C1:5; C2:2

C1:1; C2:4

Gini(N1)
= 1 – $(5/7)^2$ – $(2/7)^2$
= 1-0.51 – 0.082

= 0.408

Gini(N2)
= 1 – $(1/5)^2$ – $(4/5)^2$
= 1 – 0.04 – 0.64

= 0.32

	N1	N2
C1	**5**	**1**
C2	**2**	**4**
Gini=0.371		

Gini (Children)
= 7/12 * 0.408 +
5/12 * 0.32
= 0.238 + 0.133

= 0.371 (reduced from 0.5)

Splitting Criteria Based on Entropy

- Entropy at a given node *t*,

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

  where $p_i(t)$ is the relative frequency of class i at node t, and c is the total number of classes

- Measures lack of homogeneity (or disorderliness) of a node
  - Maximum ($\log_2 c$) when records are equally distributed among all classes implying the least beneficial situation for classification
  - Minimum (0) when all records belong to one class, implying the most beneficial situation for classification
- From $\log_b a * \log_a x = \log_b x$, we have $\log_2 x = \log_2 e * \log_e x = 1.44 \log_e x$
- Entropy based computations are similar to the GINI index computations

Computing Entropy

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

Note that $\log_2 x = \log_2 e * \log_e x = 1.44 \log_e x$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 $\log_2 0$ – 1 $\log_2 1$ = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (5/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Measure of Impurity: Misclassification Error

- Classification error at a node t

$$Error(t) = 1 - \max_i [p_i(t)]$$

where $p_i(t)$ is the relative frequency of class i at node t, and the maximum is taken over all classes *c*

- We want $\max_i[p_i(t)]$ to be large so that 1- $\max_i[p_i(t)]$ is small
- Maximum of (1 – 1/*c*) when records are equally distributed among all classes, implying the least beneficial situation for classification
- Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification

Computing Error of a Single Node

$$Error(t) = 1 - \max_i [p_i(t)]$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

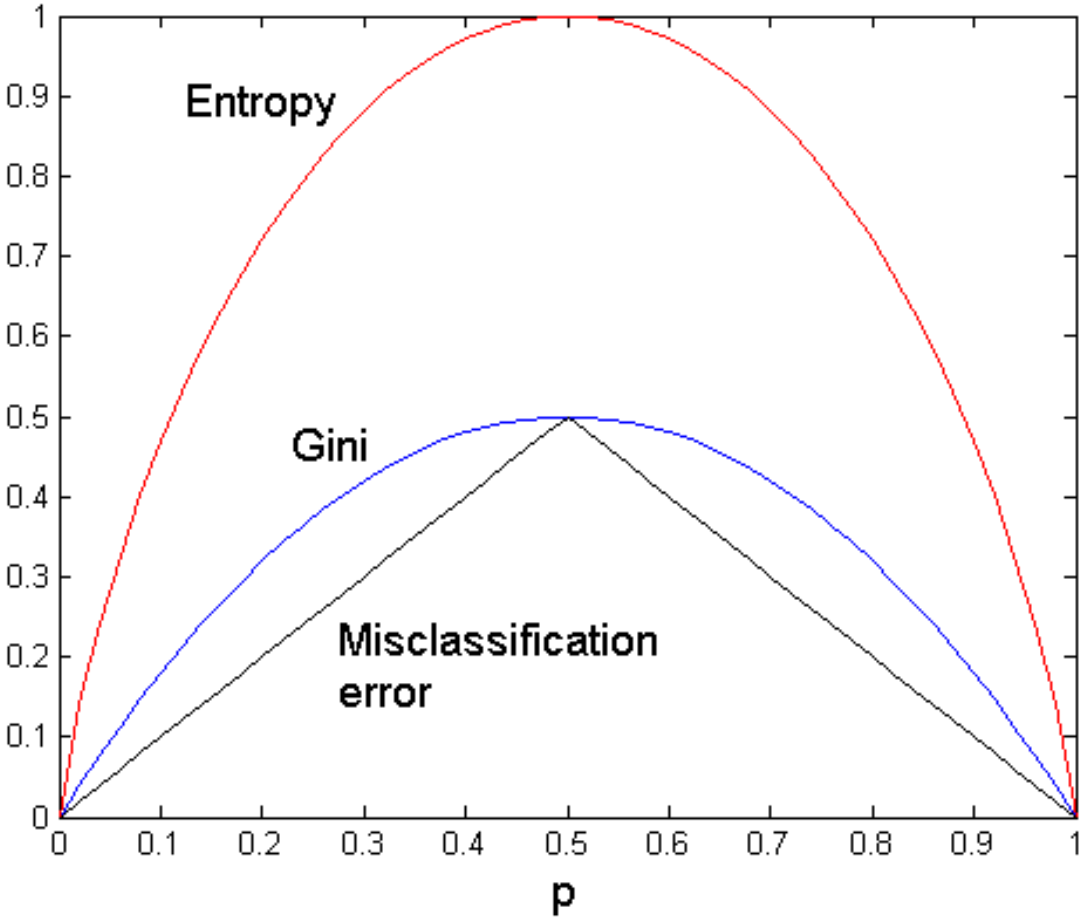
C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

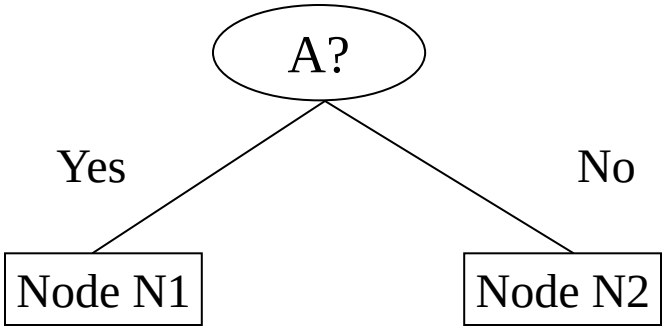
Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Impurity Measures

For a 2-class problem, with *p* giving the relative frequency of Class 1



Misclassification Error vs Gini Index



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

Gini(N1)
= 1 – $(3/3)^2$ – $(0/3)^2$
= 0

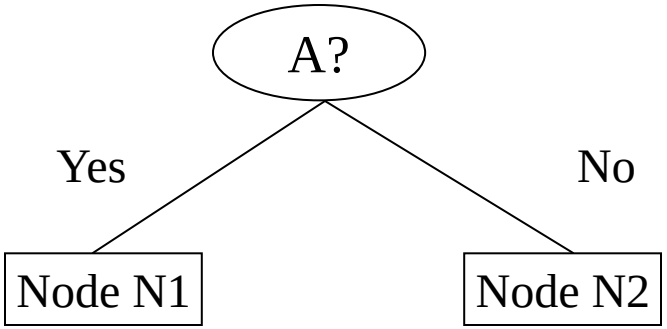
Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

	N1	N2
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

Gini improves but
what about the error?

Misclassification Error vs Gini Index



	Parent
C1	7
C2	3

Error = 0.3

Error(N1)
= 1 – max[(3/3), (0/3)]
= 0

Error(N2)
= 1 – max[(4/7), (3/7)]
= 3/7

	N1	N2
C1	3	4
C2	0	3

Error = 0.3

Error(Parent)
= 1 – max[(7/10), (3/10)] =
0.3

Error(Children)
= 3/10 * 0 + 7/10 * 3/7
= 0.3

Gini improves but error remains the same!!

Bayesian Classifiers

- Bayesian classifiers use **Bayes theorem**,

$$p(c_j \mid d) = p(d \mid c_j) * p(c_j) / p(d)$$

where

$p(c_j \mid d)$ = probability of instance d being in class c_j,
$p(d \mid c_j)$ = probability of generating instance d given class c_j,
$p(c_j)$ = probability of occurrence of class c_j, and
$p(d)$ = probability of instance d occurring

- For example: d may represent an individual, and the classes may be "buy computer" and "not buy computer"

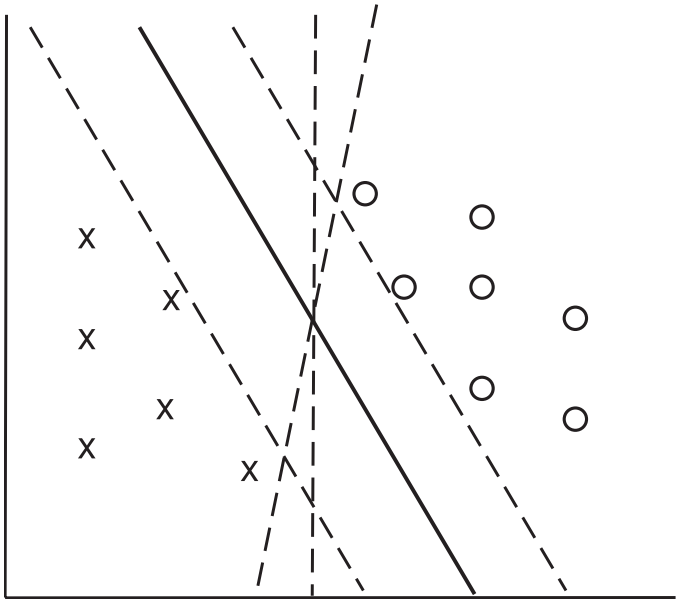
  d = (*age = youth, income = medium, student = yes, credit_rating = fair*)

- To simplify the task, **Naïve Bayesian classifiers** assume attributes have independent distributions, and thereby estimate

$$p(d \mid c_j) = p(d_1 \mid c_j) * p(d_2 \mid c_j) * \ldots * p(d_n \mid c_j)$$

Support Vector Machine Classifiers

- Simple 2-dimensional example:
  - Points are in two classes
  - Find a line (**maximum margin line**) s.t. line divides two classes, and distance from nearest point in either class is maximum



Support Vector Machine

- In *n*-dimensions points are divided by a plane, instead of a line
- SVMs can be used separators that are curve, not necessarily linear, by transforming points before classification
  - Transformation functions may be non-linear and are called kernel functions
  - Separator is a plane in the transformed space, but maps to curve in original space
- There may not be an exact planar separator for a given set of points
  - Choose plane that best separates points

Association Rules

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction
- Retail shops are often interested in associations between different items that people buy.
  - Someone who buys bread is quite likely also to buy milk
  - A person who bought the book *Database System Concepts* is quite likely also to buy the book *Operating System Concepts*
- Associations information can be used in several ways.
  - E.g., when a customer buys a particular book, an online shop may suggest associated books
- **Association rules:**

  bread $\rightarrow$ *milk* *DB-Concepts, OS-Concepts* $\rightarrow$ Networks

  - Left hand side: **antecedent**, right hand side: **consequent**
  - An association rule must have an associated **population**; the population consists of a set of **instances**
    - E.g., each transaction (sale) at a shop is an instance, and the set of all transactions is the population

Frequent Item Set

Itemset

- A collection of one or more items
  - Example: {Milk}, {Milk, Bread, Diaper}
- ***k*-itemset**
  - An itemset that contains *k* items

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Support count (σ) or absolute support

- Frequency of occurrence of an itemset
- E.g. σ({Milk, Bread, Diaper}) = 2

Support (s) or relative support

- Fraction of transactions that contain an itemset
- $s = \sigma / |T|$, where $|T|$ is the number of transactions
- E.g. s({Bread, Milk, Diaper}) = 2/5

Frequent (or Large) Itemset

- An itemset whose support is greater than or equal to a ***minsup*** threshold, where ***minsup*** is a given minimum support

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Association Rules

Association Rule

- An implication expression of the form X → Y, where X and Y are itemsets
- Arrow means co-occurrence, not causality
- Example:
  {Milk, Diaper} → {Beer}

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Rule Evaluation Metrics

- Support (s) of an association rule
  - Fraction of transactions that contain both X and Y
- Confidence (c) of an association rule
  - Measures how often items in Y appear in transactions that contain X

Example:

$$\{\text{Milk, Diaper}\} \rightarrow \{\text{Beer}\}$$

$$s = \frac{\sigma(\text{Milk, Diaper, Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk, Diaper, Beer})}{\sigma(\text{Milk, Diaper})} = \frac{2}{3} = 0.67$$

Mining Association Rules

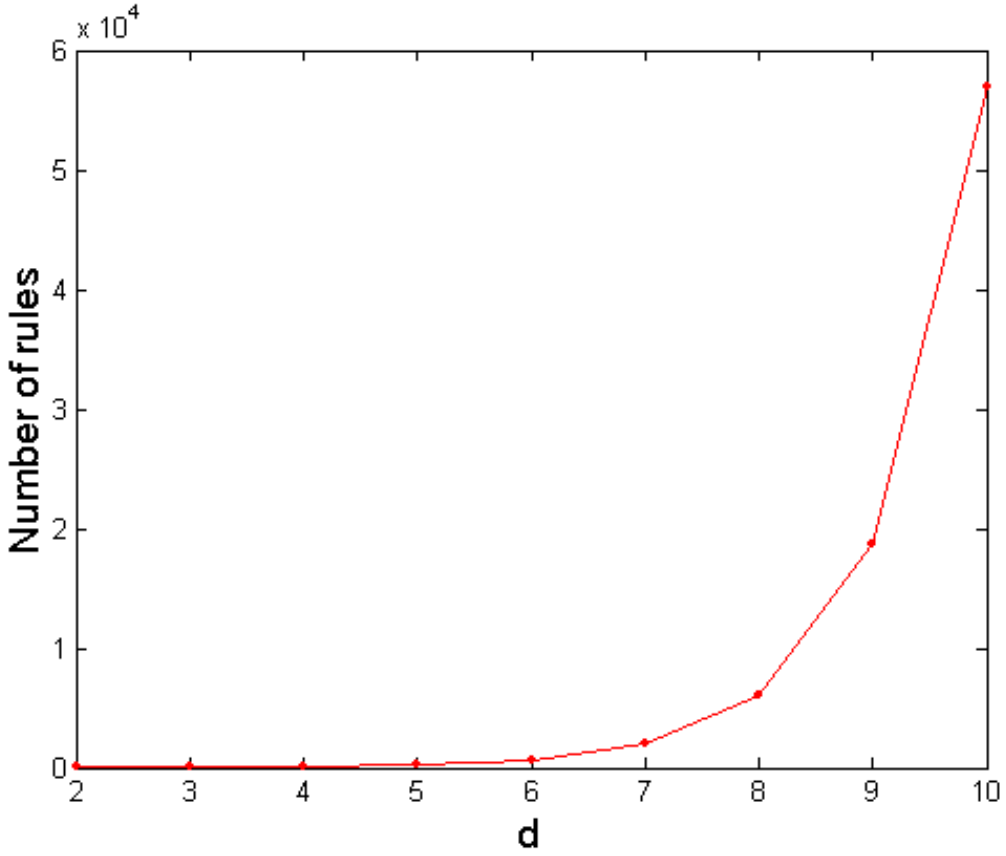
- Given a set of transactions *T*, the goal of association rule mining is to find all rules having
  - Support (*s*) ≥ *minsup* threshold
  - Confidence (*c*) ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

  ⇒ Computationally prohibitive!

Computational Complexity

- Given *d* unique items:
  - Total number of itemsets = 2^d
  - Total number of possible association rules:



$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$

$$= 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules

Computational Complexity (Proof)

Suppose there are d items. We first choose k of the items to form the left-hand side of the rule. There are $\binom{d}{k}$ ways for doing this. After selecting the items for the left-hand side, there are $\binom{d-k}{i}$ ways to choose the remaining items to form the right hand side of the rule, where $1 \leq i \leq d-k$. Therefore the total number of rules (R) is:

$$
\begin{aligned}
R &= \sum_{k=1}^{d} \binom{d}{k} \sum_{i=1}^{d-k} \binom{d-k}{i} \\
&= \sum_{k=1}^{d} \binom{d}{k} \left(2^{d-k} - 1\right) \\
&= \sum_{k=1}^{d} \binom{d}{k} 2^{d-k} - \sum_{k=1}^{d} \binom{d}{k} \\
&= \sum_{k=1}^{d} \binom{d}{k} 2^{d-k} - \left[2^d + 1\right],
\end{aligned}
$$

where

$$
\sum_{i=1}^{n} \binom{n}{i} = 2^n - 1.
$$

Computational Complexity (Proof Continued)

Since

$$(1+x)^d = \sum_{i=1}^{d} \binom{d}{i} x^{d-i} + x^d,$$

substituting $x = 2$ leads to:

$$3^d = \sum_{i=1}^{d} \binom{d}{i} 2^{d-i} + 2^d.$$

Therefore, the total number of rules is:

$$R = 3^d - 2^d - \left[2^d + 1\right] = 3^d - 2^{d+1} + 1.$$

Computational Complexity (Direct Combinatorial Argument)

Let there be d items. For a given item, it may be placed at the LHS, RHS, of the rule, or being left out altogether, and this accounts for 3 possibilities for a given item, which gives 3^d possibilities for all the d items.

But the above includes rules with a blank LHS or a blank RHS, which are not valid rules. The number of rules with a blank RHS is 2^d, which includes also a rule with blank on both sides; excluding this one rule gives $2^d - 1$, which represents rules with a non-blank LHS but a blank RHS.

Similarly considering non-blank RHS but a blank LHS accounts for another $2^d - 1$ rules.

Combining the above gives $2 \times (2^d - 1) = 2^{d+1} - 2$ rules with either a blank LHS or a blank RHS, but not blank on both sides.

Adding the one rule with a blank LHS as well as a blank RHS gives $2^{d+1} - 2 + 1 = 2^{d+1} - 1$, which represents the total number of rules to be excluded. These are the (invalid) rules with a blank LHS, or a blank RHS, or blank on both sides.

Subtracting these rules from the unrestricted possibilities of 3^d, we obtain the total number of valid rules as $3^d - 2^{d+1} + 1$, which is in agreement with the analytical proof above.

Mining Association Rules

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence

Reducing the Search Space

- The itemsets that have a support that exceeds the threshold minimum are called *large* (or *frequent*) itemsets, where large here means large support
- Discovering all large itemsets together with the value for the support is a major problem if the cardinality of the set of items is very high
- A typical supermarket has thousands of items
  - The number of distinct (non-empty) itemsets is 2^m, where m is number of items, and counting support for all possible itemsets becomes computation-intensive.
- To reduce the combinatorial search space, algorithms for finding association rules utilize two properties

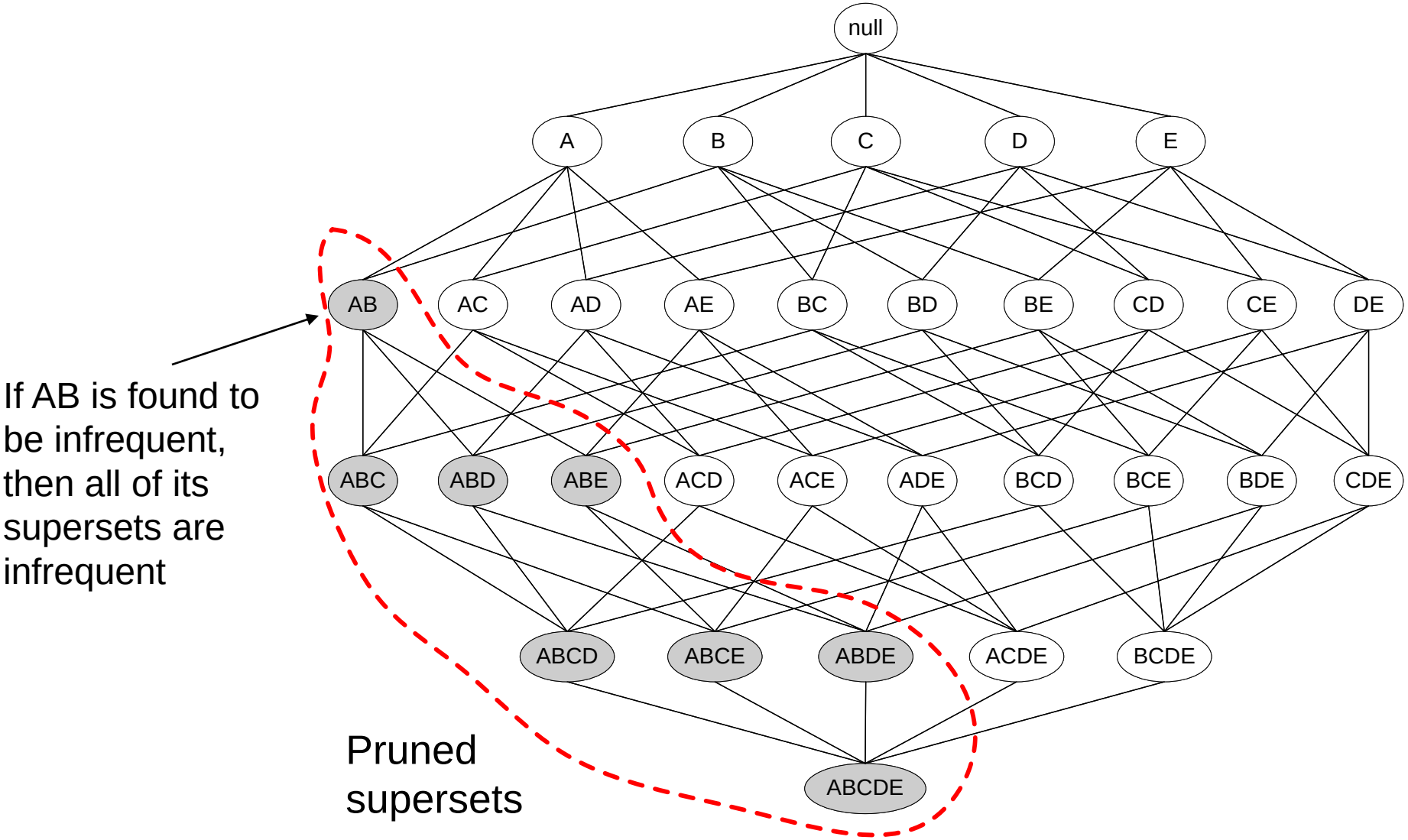
Reducing the Search Space

- **Apriori Principle**:
  - If an itemset is frequent, then all of its subsets must also be frequent
  - Support of an itemset never exceeds the support of its subsets

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

- *Downward Closure*
  - A subset of a large itemset must also be large (that is, each subset of a large itemset exceeds the minimum required support).
- *Antimonotonicity*
  - Conversely, a superset of a small itemset is also small (implying that it does not have enough support).
- Thus, once an itemset is found to be small (not a large itemset), then any extension to that itemset, formed by adding one or more items to the set, will also yield a small itemset

Reducing the Search Space



Candidate Generation: $F_{k-1} \times F_{k-1}$ Method

- Merge two frequent (k-1)-itemsets if their first (k-2) items are identical

- F_3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}
  - Merge(**AB**C, **AB**D) = **AB**CD
  - Merge(**AB**C, **AB**E) = **AB**CE
  - Merge(**AB**D, **AB**E) = **AB**DE

  - Do not merge(**A**BD,**A**CD) because they share only prefix of length 1 instead of length 2

Illustrating the Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

If every subset is considered,

$${}^6C_1 + {}^6C_2 + {}^6C_3$$

$$6 + 15 + 20 = 41$$

With support-based pruning,

$$6 + 6 + 1 = 13$$

Triplets (3-itemsets)

Itemset	Count
{Bread, Diaper, Milk}	**2**

Use of $F_{k-1} \times F_{k-1}$ method for candidate generation results in only one 3-itemset. This is eliminated after the support counting step.

Multidimensional Associations

- Consider a file of customer transactions with three dimensions
  - Transaction-Id
  - Time
  - Items Bought
- The following rule is an example where we include the label of the single dimension:

  Items-Bought(milk) $\Rightarrow$ Items-Bought(juice)
- Sometimes, it may be of interest to find association rules that involve multiple dimensions, e.g.,

  Time(6:30...8:00) $\Rightarrow$ Items-Bought(milk)
- Rules like these are called multidimensional association rules

Clustering

- It is often useful to partition data without having training samples
  - this is an instance of *unsupervised learning*
- In business, it may be important to determine groups of customers who have similar buying patterns, or in medicine, it may be important to determine groups of patients who show similar reactions to prescribed drugs.
- The goal of clustering is to place records into groups, such that records in a group are similar to each other and dissimilar to records in other groups, and the groups are usually disjoint.

Clustering

- An important aspect of clustering is the similarity function used.
- The Euclidean distance is often used to measure similarity:

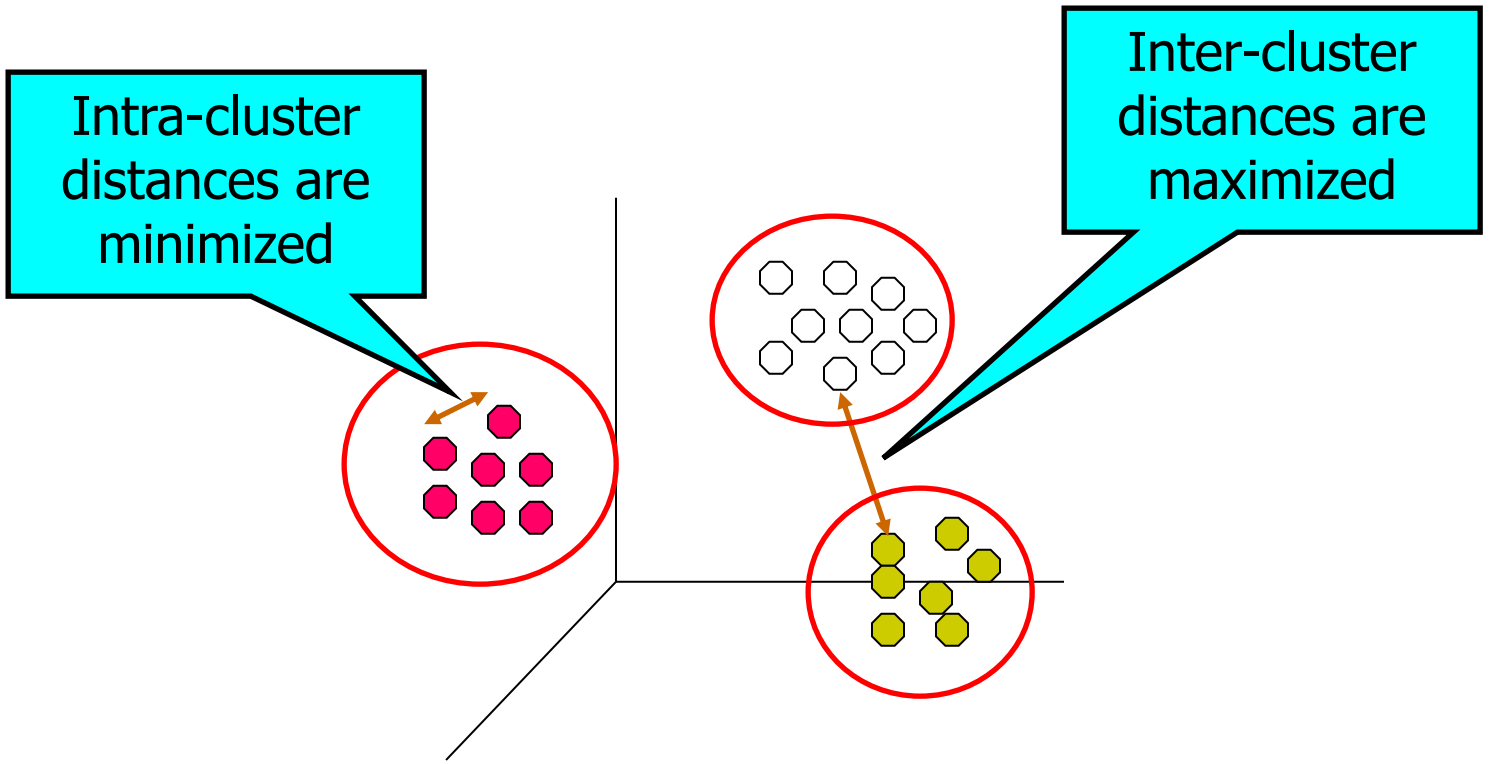
  Consider two *n*-dimensional data records as points ***x*** and ***y*** in *n*-dimensional space. We can consider the value for the *i*th dimension as x_i and y_i for the two records. The Euclidean distance between points $\boldsymbol{x} = (x_1,\ldots,x_n)$ and $\boldsymbol{y} = (y_1,\ldots,y_n)$ in *n*-dimensional space is

$$\rho(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}$$

  The smaller the distance between two points, the greater is the similarity

Cluster Analysis

- Finding groups of objects such that the objects in a group will be similar to one another and different from the objects in other groups



K-Means Clustering

- Each cluster is associated with a **centroid** (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, *K*, must be specified
- The basic algorithm is very simple

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K-Means Clustering

Consider the following table

Record	Age	Years of Service
1	30	5
2	50	25
3	50	15
4	25	5
5	30	10
6	55	25

Assume that the number of desired clusters *K* is 2. Let the algorithm choose records with Record 3 for cluster C_1 and Record 6 for cluster C_2 as the initial cluster centroids. The remaining records will be assigned to one of those clusters during the first iteration of the repeat loop.

K-Means Clustering

Record 1 has a distance from C_1 of $\sqrt{(20^2 + 10^2)} = 22.4$ and a distance from C_2 of 32.0, so it joins cluster C_1. Record 2 has a distance from C_1 of 10.0 and a distance from C_2 of 5.0, so it joins cluster C_2 and so on.

Record	Age	Years of Service	Dist from 3	Dist from 6
1	30	5	22.4	32.0
2	50	25	10.0	5.0
3	**50**	**15**	0	-
4	25	5	25.5	36.6
5	30	10	20.6	29.2
6	**55**	**25**	-	0

Thus we have the clusters

C_1 = {Record 1, **Record 3**, Record 4, Record 5}

C_2 = {Record 2, **Record 6**}.

K-Means Clustering

Next, compute the new centroids:

The new centroid for C_1 is ((30+50+25+30)/4, (5+15+5+10)/4)

= (33.75, 8.75)

The new centroid for C_2 is ((50+55)/2, (25+25)/2) = (52.5, 25)

In the second iteration the six records are placed into the two clusters as follows:

C'_1 = {Record 1, Record 4, Record 5}

C'_2 = {Record 2, Record 3, Record 6}.

The mean for C'_1 and C'_2 is re-computed as (28.3, 6.7) and (51.7, 21.7).

In the next iteration, all records stay in their previous clusters and the algorithm terminates.