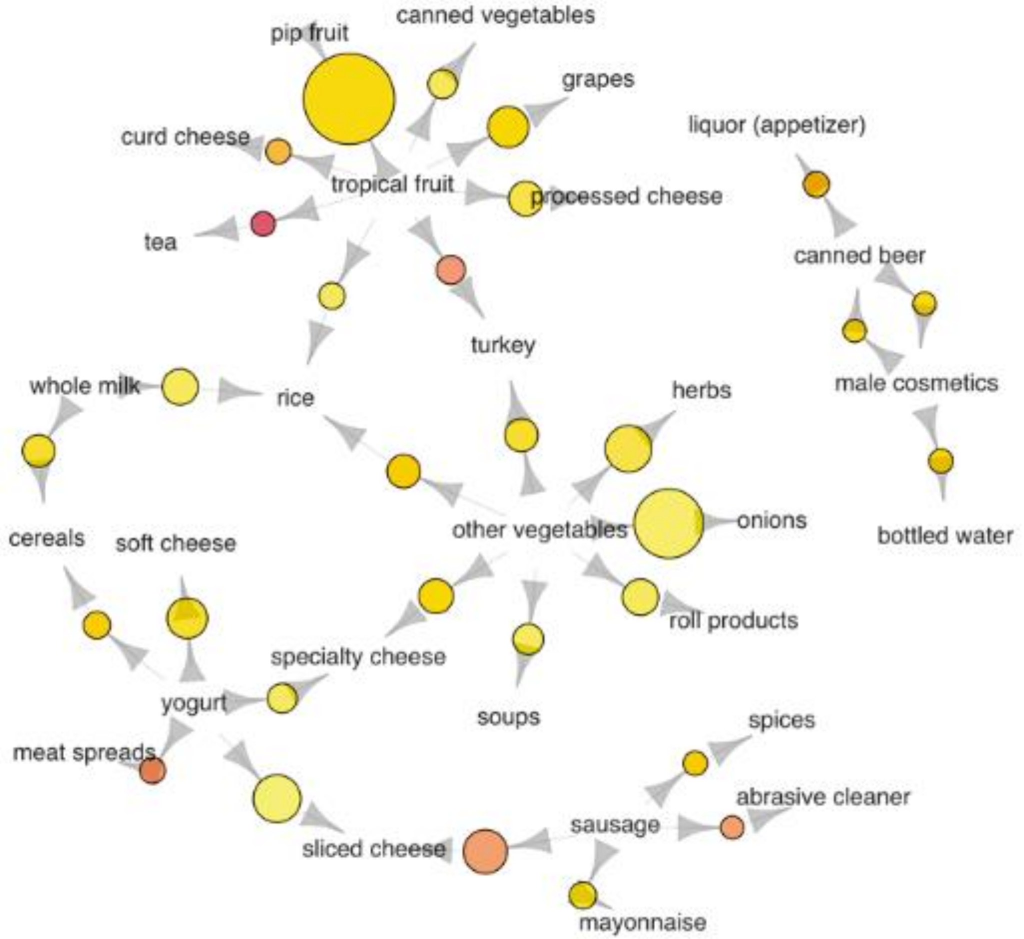
Association mining

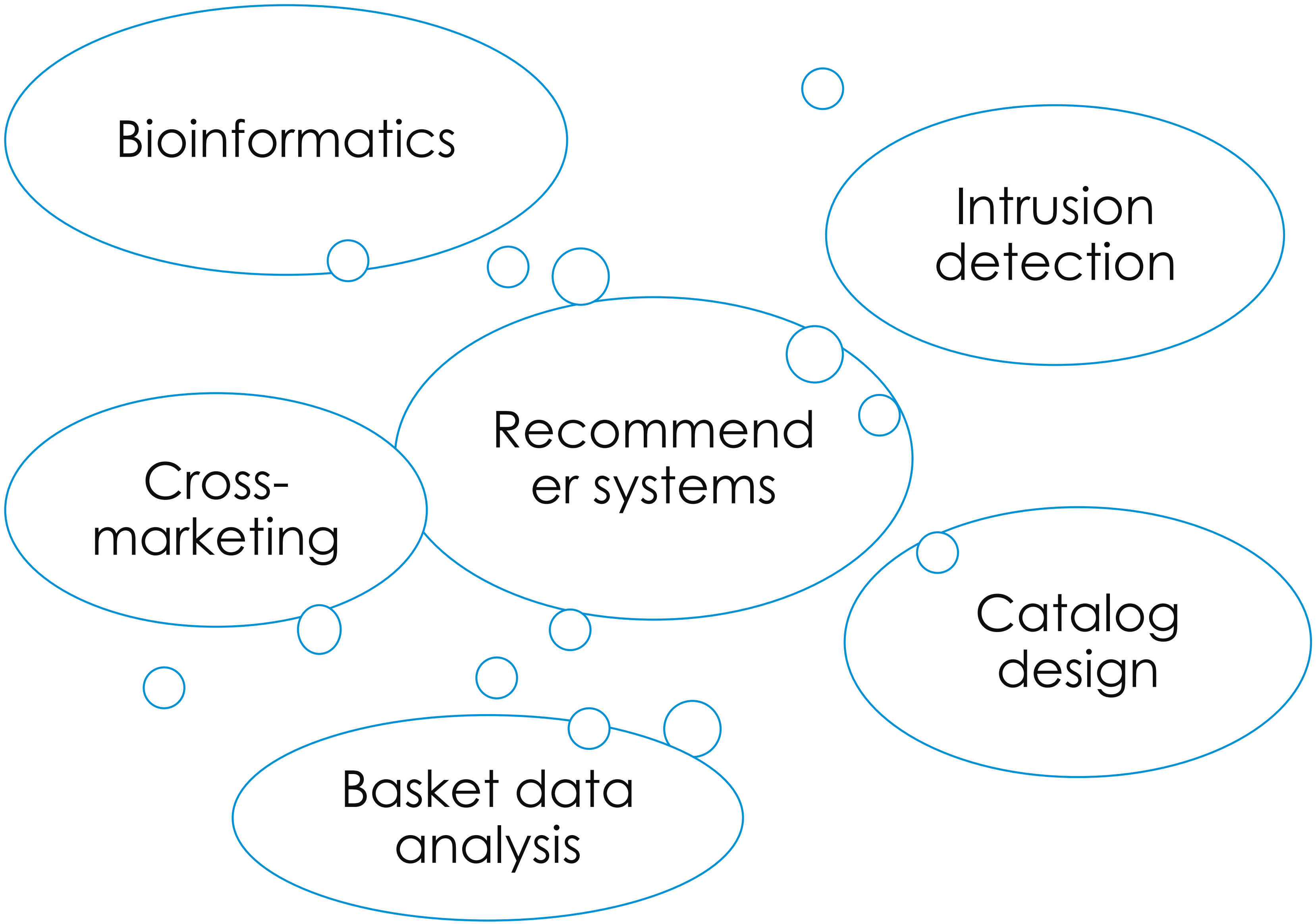
Data Mining for Business and Governance

Dr. Gonzalo Nápoles

What is association mining?

- Association rules mining is about finding
  - Association
  - Correlation
  - Causal structures
- Among a set of items or objects in
  - Transaction databases
  - Relational databases
  - Example datasets
  - Other sources of information




Bioinformatics
Intrusion detection
Recommend er systems
Cross-marketing
Catalog design
Basket data analysis

What is an association rule?

- An association rules is an implication with the form:

$$Antecedent \rightarrow Consequent\ [support, confidence]$$

- Example:
  - Given a database of customer transactions where each transaction is a list of items purchased by a customer in a visit

$$buys(x, "diapers") \rightarrow buys(x, "beer")[0.5\%, 60\%]$$

0.5% of customer transactions include diapers

60% of customers that buy diapers also buy beer

What is an association rule?

- How to use them?
  - Find all rules that associate the presence of one set of items with that of another set of items. Any number of items, in principle.

$$buys(x, "bread") \,\&\, buys(x, "butter") \rightarrow buys(x, "milk")[30\%, 60\%]$$

  - We can specify constraints on rules:
    - Example: "*find only rules involving home laundry appliances*"

Measures: support and confidence

- We need a binary set of attributes $I = \{i_1, \dots, i_N\}$ called *items* AND a set of transactions $T = \{t_1, \dots, t_M\}$ called *database*.

- Each *transaction* t has a unique ID and contains a subset of items in I. A *rule* can be represented as $X \rightarrow Y : X, Y \subseteq I$.

$$Antecedent \rightarrow Consequent\ [support, confidence]$$

Measures: support and confidence

- Support: can be defined as the proportion (**probability**) of transactions t that contains itemset X in T. That is to say:

$$supp(X) = \frac{|t \in T : X \subseteq t|}{|T|}$$

- Confidence: can be defined as the ratio (**conditional probability)** of transaction having X also contains Y.

$$conf(X \rightarrow Y) = supp(X \cup Y)/supp(X)$$

Mining association rules

- Given a set of transactions T, the goal of association rule mining is to find all rules having
  - support ≥ minimum support (minsup) threshold
  - confidence ≥ minimum confidence (minconf) threshold

ID	Items
1	A, B, C
2	A, C
3	A, D
4	B, E, F

With minimum support 50% and minimum confidence 50%, we have:

$$\boxed{A \rightarrow C\ [50\%, 66.6\%]}$$

$$\boxed{C \rightarrow A\ [50\%, 100\%]}$$

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!

Mining association rules

Two-step approach:

Still computation ally expensive

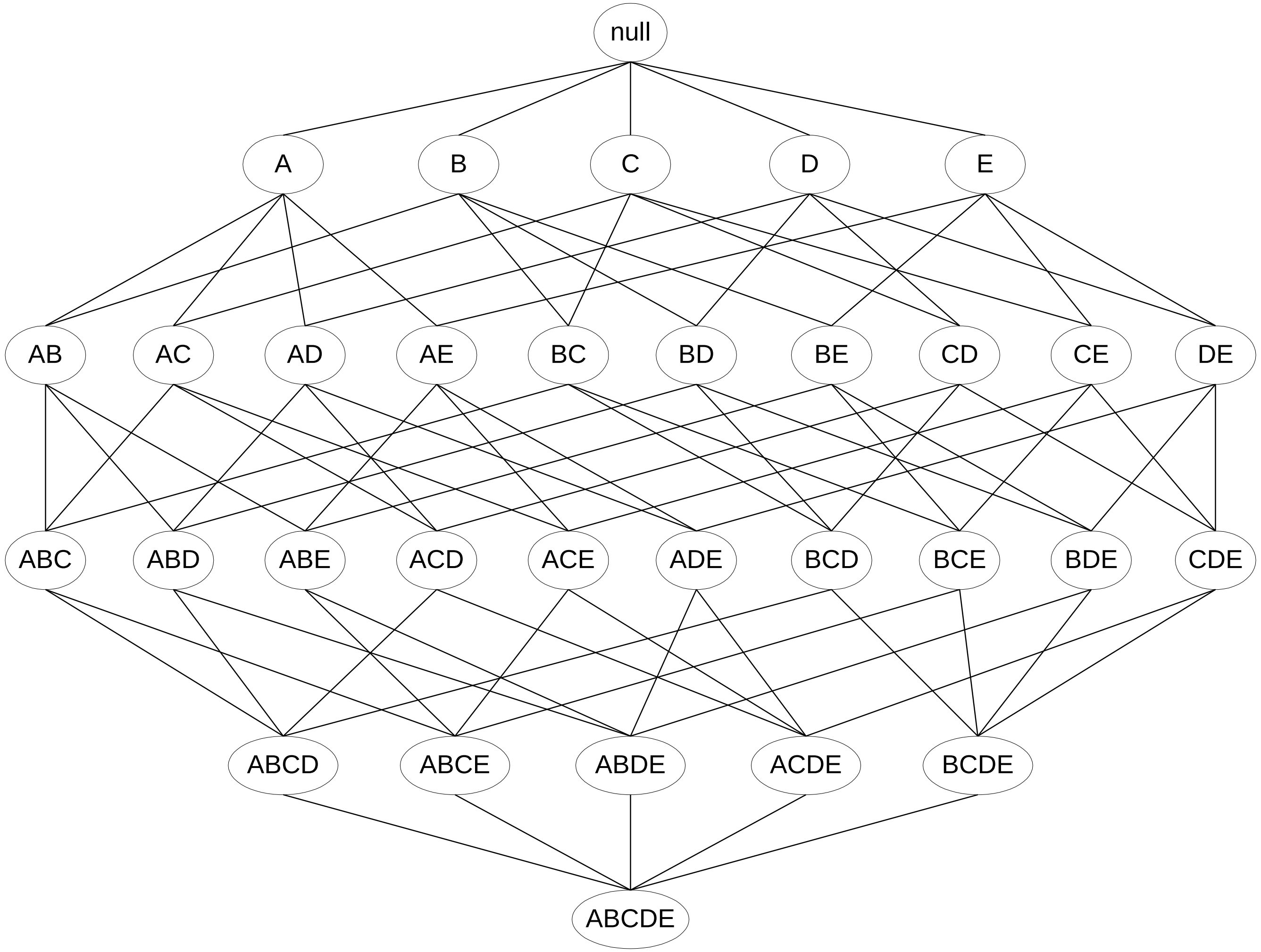
1. Frequent Itemset Generation

Generate all itemsets whose support ≥ minsup

2. Rule Generation

Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent itemset generation



Given N items, there are 2^N -1 possible itemsets

Mining association rules

Transaction ID	Items
1	A, B, C
2	A, C
3	A, D
4	B, E, F

minsup = 50%
minconf = 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

For rule $A \rightarrow C$:

support = $supp(\{A, C\}) = 50\%$
confidence = $supp(\{A, C\})/supp(\{A\}) = 66.6\%$

The Apriori principle

Any subset of a frequent itemset must be frequent

Relevance

Subsets with non-frequent items
are not interesting!

Mining frequent itemsets

The key step

- Find the *frequent itemsets*: the sets of items with minimum support
  - A subset of a frequent itemset must also be a frequent itemset, i.e., if {AB} is a frequent itemset, both {A} and {B} must be a frequent itemset
  - Iteratively find frequent itemsets with cardinality from 1 to k (k-itemset) (usually k is not larger than 7)
- Use the frequent itemsets to generate association rules.

Apriori algorithm

- Let $k = 1$
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
  - Generate length $(k + 1)$ candidate itemsets from length k frequent itemsets
  - Prune candidate itemsets containing subsets of length k that are infrequent
  - Count the support of each candidate by scanning the database
  - Eliminate candidates that are infrequent, thus retaining only those that are frequent

Apriori algorithm – an example

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

minsup=2

Itemset	Support
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

Itemset	Support
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

Itemset	Support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

Itemset	Support
{2 3 5}	2

How to generate candidates?

- Suppose the items in L_{k-1} are listed in an order
- Step 1: self-joining L_{k-1}

  insert into C_k

  select $p.item_1$, $p.item_2$, …, $p.item_{k-1}$, $q.item_{k-1}$

  from L_{k-1} p, L_{k-1} q

  where $p.item_1=q.item_1$, …, $p.item_{k-2}=q.item_{k-2}$, $p.item_{k-1} < q.item_{k-1}$
- Step 2: pruning

  for all *itemsets c in* C_k do

      for all *(k-1)-subsets s of c* do

          if *(s is not in* L_{k-1}*)* then delete c from C_k

Apriori algorithm - example

Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

1st scan

C_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Itemset	Support
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

2nd scan

L_2

Itemset	Support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

Itemset	Support
{2 3 5}	2

3rd scan

L_3

Itemset	Support
{2 3 5}	2

minsup=2

More considerations

- **Choice of minimum support threshold**:
  - lowering support threshold results in more frequent itemsets
  - this may increase number of candidates and max length of frequent itemsets

- **Dimensionality (number of items) of the data set**:
  - more space is needed to store support count of each item
  - if number of frequent items also increases, both computation and I/O costs may also increase

More considerations

- **Size of database**
  - Apriori makes multiple passes, thus the execution time of algorithm may increase significantly.

- **Average transaction width**
  - Transaction width increases with denser databases.
  - This may increase max length of frequent itemsets.

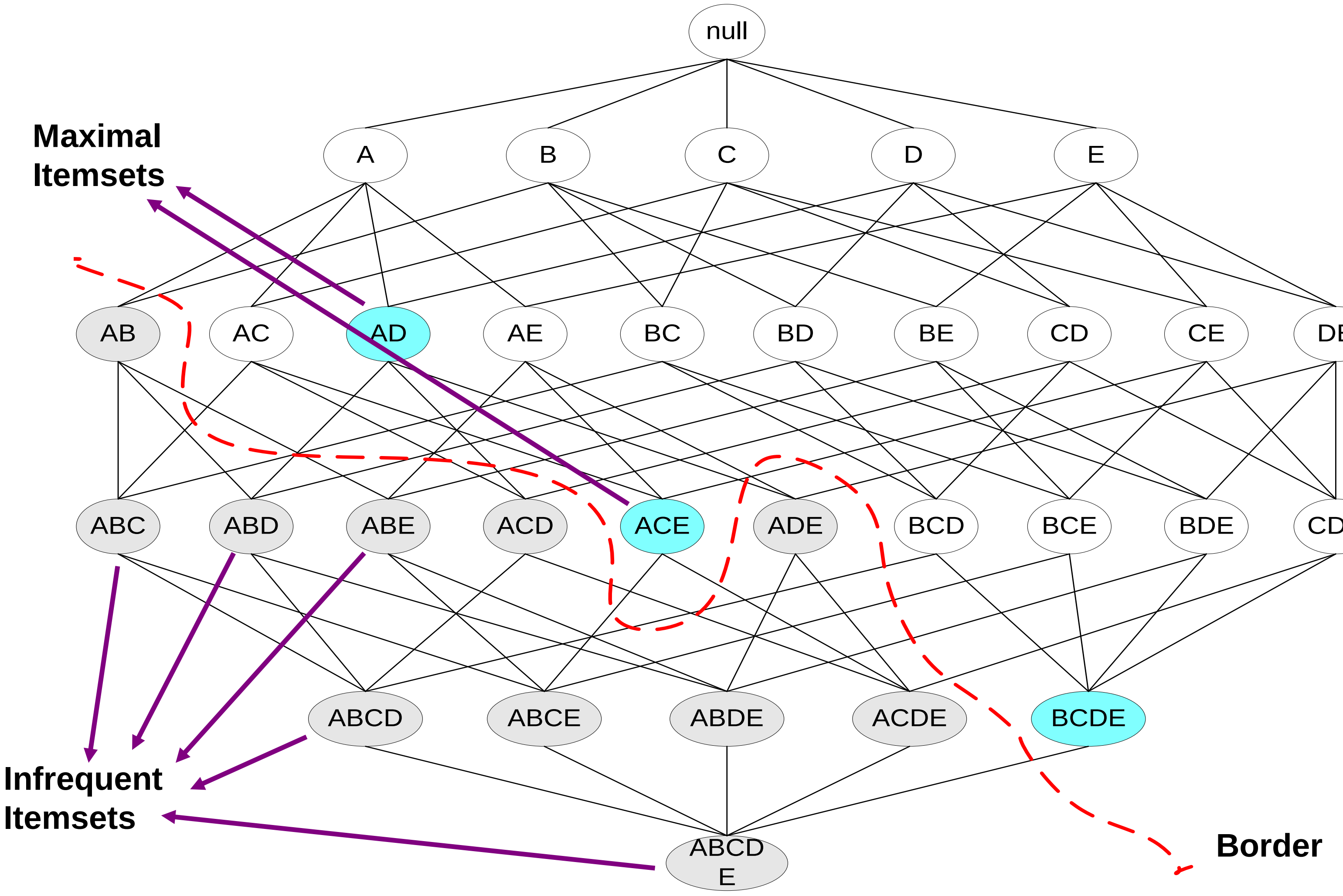
Maximal frequent itemset

If we have an itemset of length N, the a superset is an itemset with length greater than N.

An immediate superset adds an item to the previous itemset.

The itemset {ABC} is an immediate superset of the itemset {AB}

An itemset if maximal frequent if a) it is frequent and b) we cannot build an immediate superset that is also frequent.



An itemset is maximal frequent if no immediate superset is frequent

Closed itemset

What if an immediate superset has the same support as the current itemset?

It would be useful to distinguish those cases.

An itemset if closed if a) it is frequent and b) we cannot build an immediate superset with the same support than it (the current itemset).

However, the superset can still be frequent.

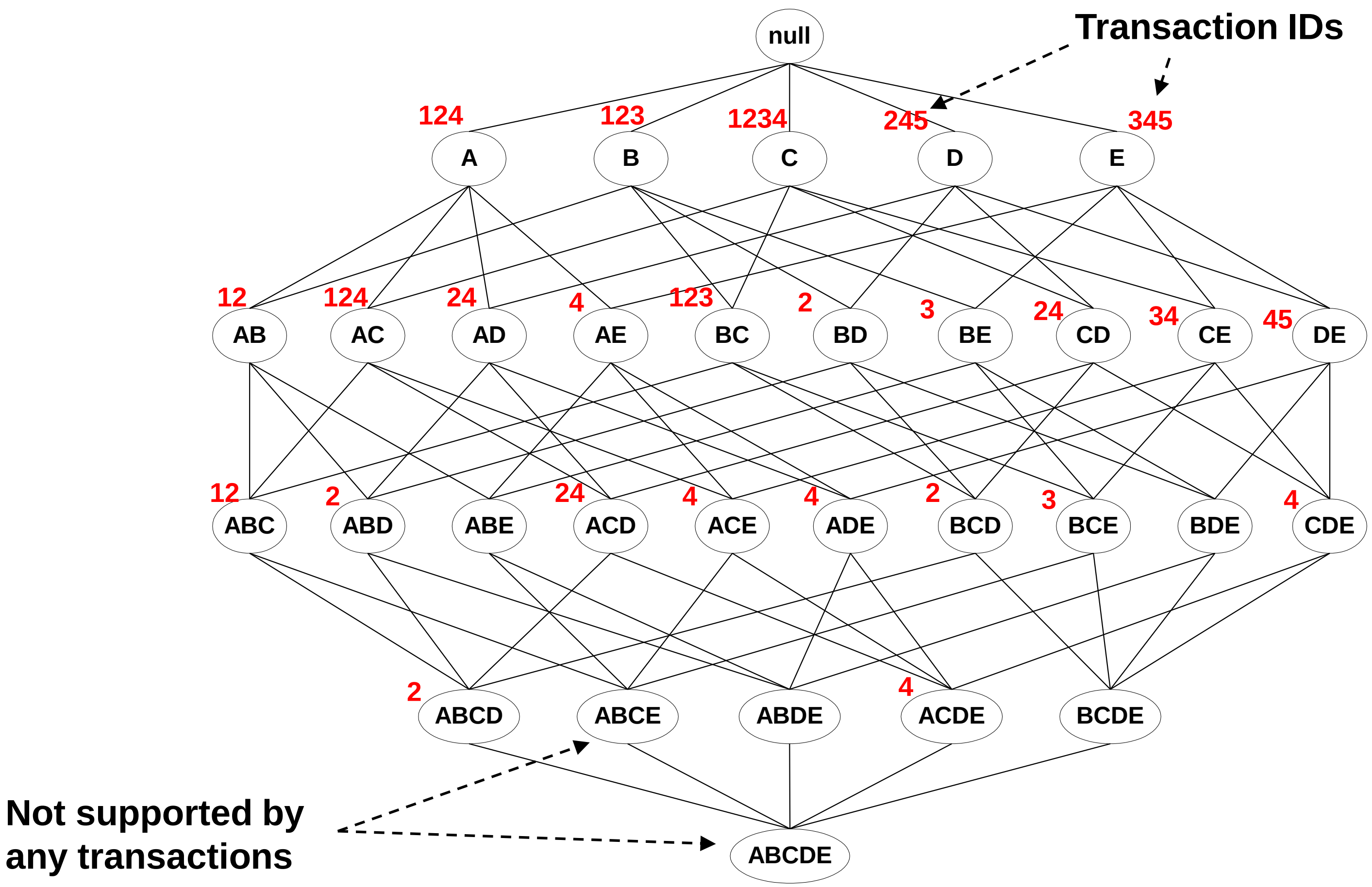
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,B,C,D}
4	{A,B,D}
5	{A,B,C,D}

Itemset	Support
{A}	4
{B}	5
{C}	3
{D}	4
{A,B}	4
{A,C}	2
{A,D}	3
{B,C}	3
{B,D}	4
{C,D}	3

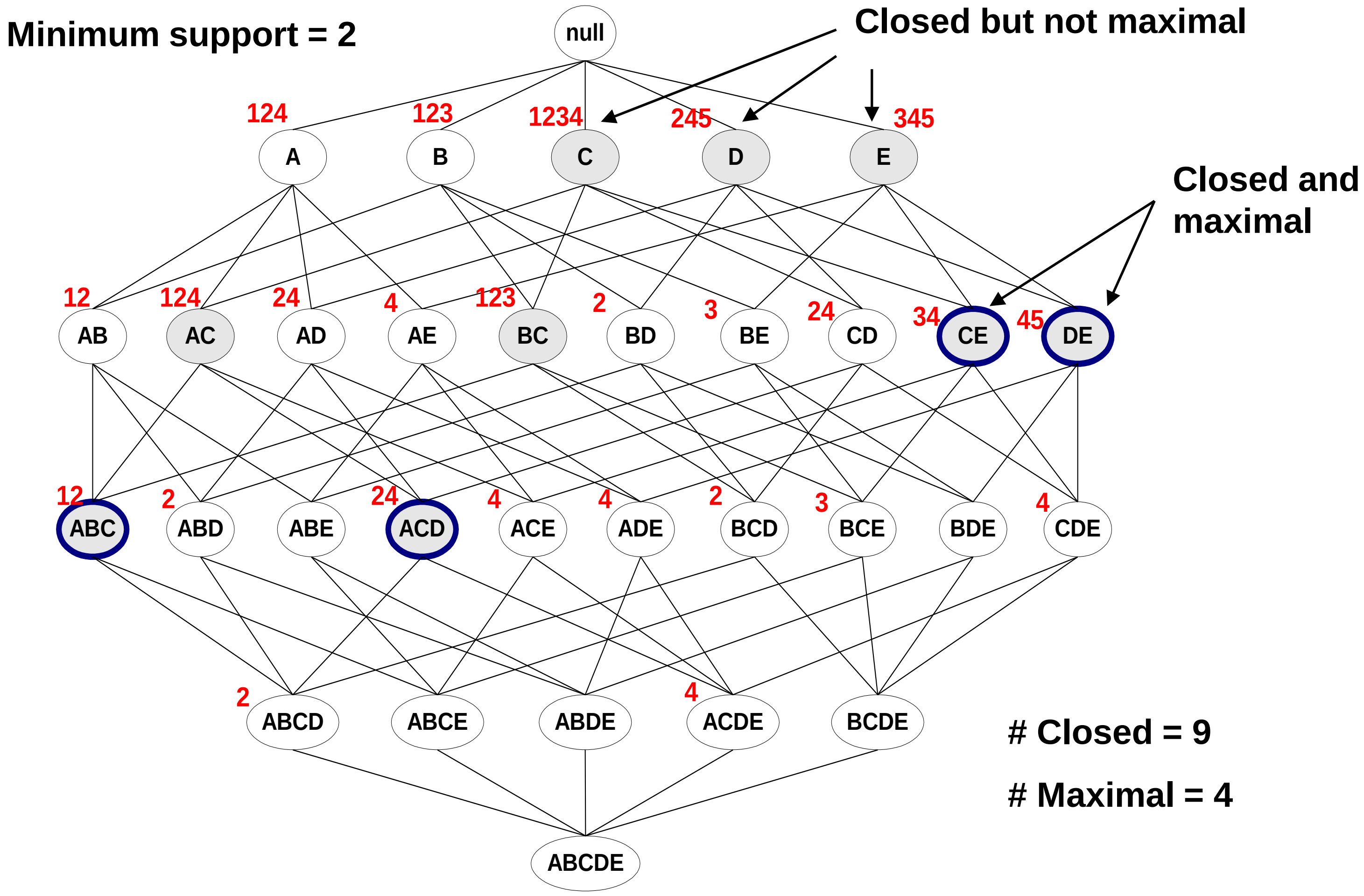
Itemset	Support
{A,B,C}	2
{A,B,D}	3
{A,C,D}	2
{B,C,D}	3
{A,B,C,D}	2

An itemset is closed if no immediate superset has the same support as the itemset.

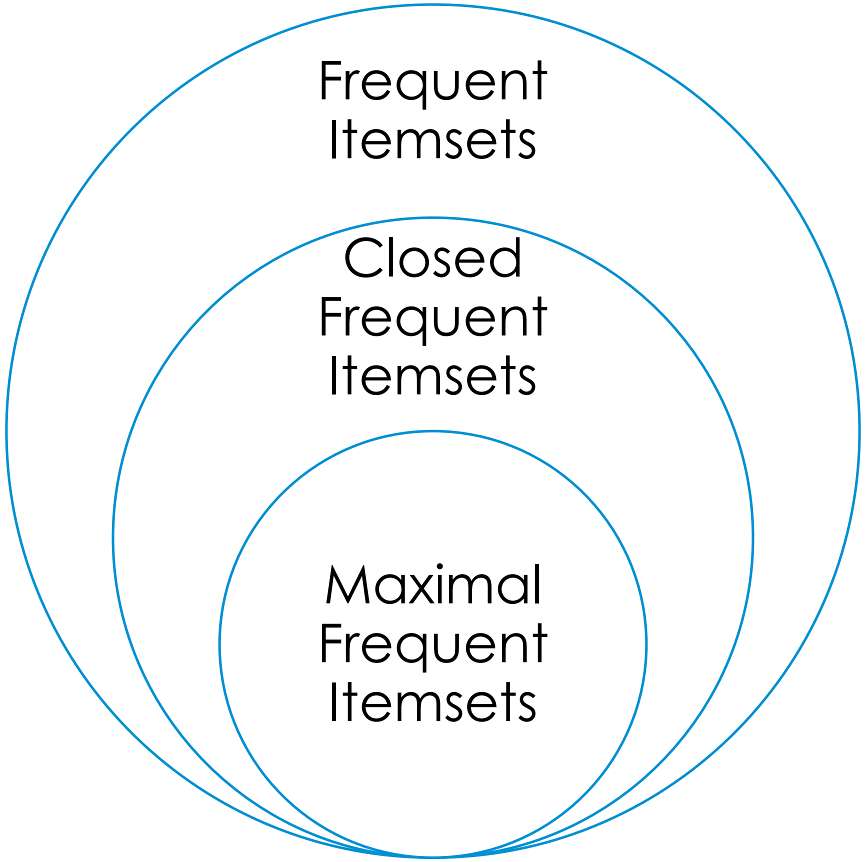
Maximal versus closed



Maximal versus closed



Maximal versus closed



Effect of support distribution

- How to set the appropriate *minsup* threshold?
    - If ***minsup*** is set too high, we could miss itemsets involving interesting rare items (e.g., expensive products)
    - If ***minsup*** is set too low, it is computationally expensive and the number of itemsets is very large
- Using a single minimum support threshold may not be effective.

Quantitative association rules

Record ID	Age	Married	NumCars
100	23	No	1
200	25	Yes	1
300	29	No	0
400	34	Yes	2
500	38	Yes	2

Sample Rules	Support	Confidence
<age:30..39> and <married:yes> → <numCars:2>	40%	100%
<numCars:0..1> → <married:no>	40%	66.70%

Mapping quantitative to Boolean

- Map the problem to the Boolean association rules by:
  - discretizing a non-categorical attribute to intervals
    - Age [20,29], [30,39],...
  - forming Boolean records
    - categorical attributes: each value becomes one item
    - non-categorical attributes: each interval becomes one item

Record ID	Age	Married	Cars
100	23	No	1
500	38	Yes	2

Record ID	Age: 20..29	Age: 30..39	Married: yes	Married: no	Cars: 0	Cars: 1	Cars: 2
100	1	0	0	1	0	1	0
500	0	1	1	0	0	0	1

Mining quantitative association rules

- Problems with the mapping
  - too few (large) intervals: loss of useful information and low confidence
  - too many (small) intervals: not enough support
- Solutions
  - using the supports of an itemset and its generalizations to determine the intervals
  - using interest measure to control the number of association rules
- However, this is still very much an open problem…

Pattern evaluation

- Association rule algorithms tend to produce too many rules
  - many of them are uninteresting or redundant
  - Redundant if $\{A,B,C\} \rightarrow \{D\}$ and $\{A,B\} \rightarrow \{D\}$ have same support & confidence

- We can design other measures to prune/rank the derived patterns and replace support & confidence.

Association mining

Data Mining for Business and Governance

Dr. Gonzalo Nápoles