Association Analysis

Naren Ramakrishnan

VirginiaTech

Recap

- Supervised learning
  - Classification, Regression
- Unsupervised learning
  - Clustering, Dimensionality Reduction
- Time Series Analysis
  - Both supervised and unsupervised learning

Today

- Association analysis
  - Primarily unsupervised learning
  - One of the “new age” data mining problems
- Goes by other names
  - Market basket analysis
  - Mining transaction datasets
  - Itemset mining
  - Association rule mining

Example

- 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

Market basket

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**

Examples of Association Rules

$\{Diaper\} \rightarrow \{Beer\}$,
$\{Milk, Bread\} \rightarrow \{Eggs,Coke\}$,
$\{Beer, Bread\} \rightarrow \{Milk\}$,

Implication means co-occurrence, not causality!

Another example

- Mining associations in electronic medical records

Property	Value
Number of patients	1,620,681
Number of diagnostic (ICD) codes	41,186,511
Number of procedure (CPT) codes	38,942,605
Max. number of codes in a record	10,430
Min. number of codes in a record	1
Max. span of a record in days	8202 days$\approx$ 22.5 years
Min. span of a record in days	1

Describing the Relationship between Cat Bites and Human Depression Using Data from an Electronic Health Record

David A. Hanauer, Naren Ramakrishnan, Lisa S. Seyfried

How do we find association rules?

- First
  - Find “frequent” itemsets {X,Y}
    - Defined by a support threshold
- Next
  - See if $X \rightarrow Y$ or $Y \rightarrow X$ hold
    - Defined by a confidence threshold

Frequent itemsets

- **Itemset**
  - A collection of one or more items
    - Example: {Milk, Bread, Diaper}
  - k-itemset
    - An itemset that contains k items
- **Support count (σ)**
  - Frequency of occurrence of an itemset
  - E.g. σ({Milk, Bread,Diaper}) = 2
- **Support**
  - Fraction of transactions that contain an itemset
  - E.g. s({Milk, Bread, Diaper}) = 2/5
- **Frequent Itemset**
  - An itemset whose support is greater than or equal to a *minsup* threshold

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

Example:

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

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

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

All this sounds good..

- But how do we mine association rules from a given database?
  - Keep in mind that the database is likely to have billions of transactions and potentially millions of items

Observation 1

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} $\rightarrow$ {Beer} (s=0.4, c=0.67)
{Milk,Beer} $\rightarrow$ {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} $\rightarrow$ {Milk} (s=0.4, c=0.67)
{Beer} $\rightarrow$ {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} $\rightarrow$ {Milk,Beer} (s=0.4, c=0.5)
{Milk} $\rightarrow$ {Diaper,Beer} (s=0.4, c=0.5)

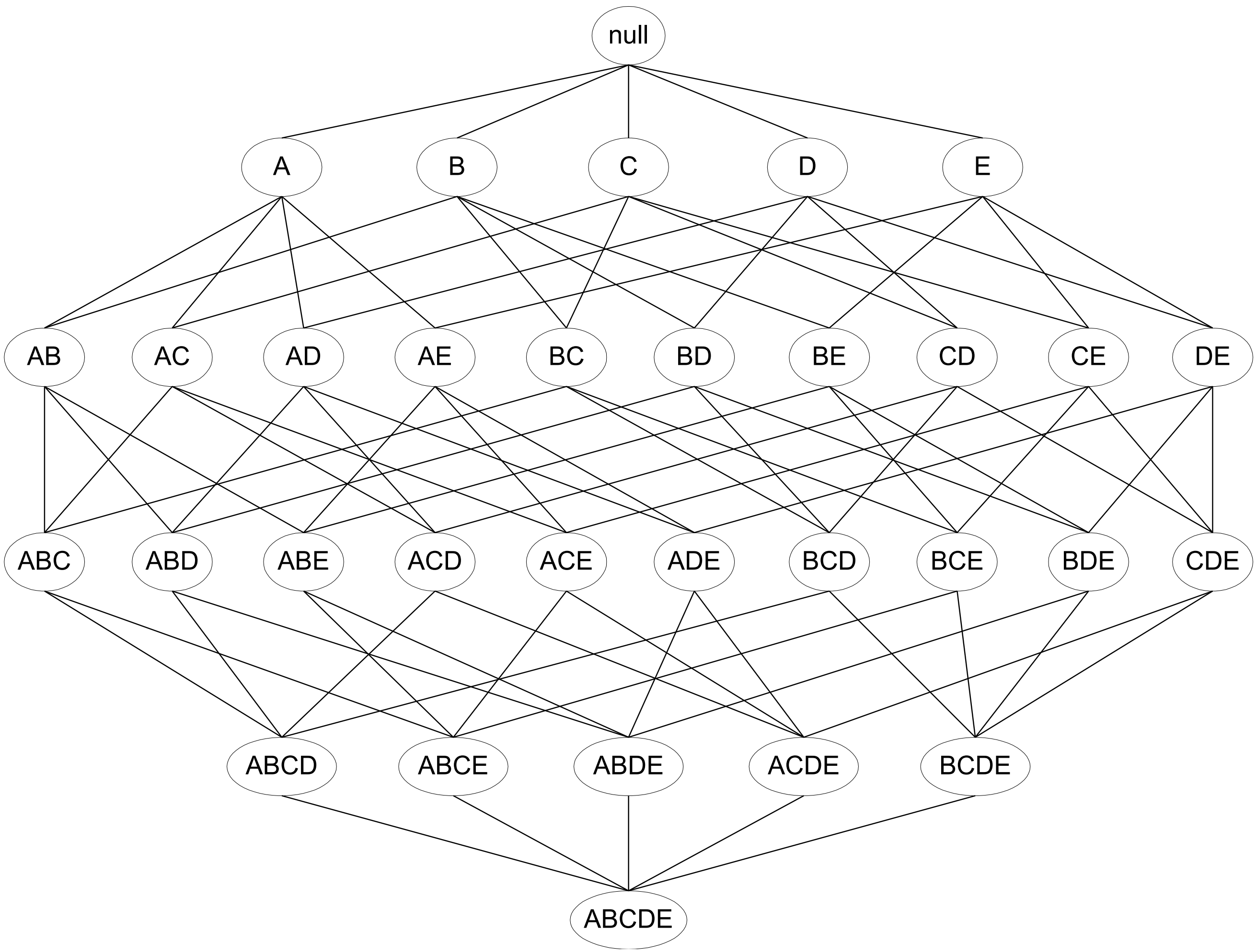
Observations:

- 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
- Thus, we may decouple the support and confidence requirements

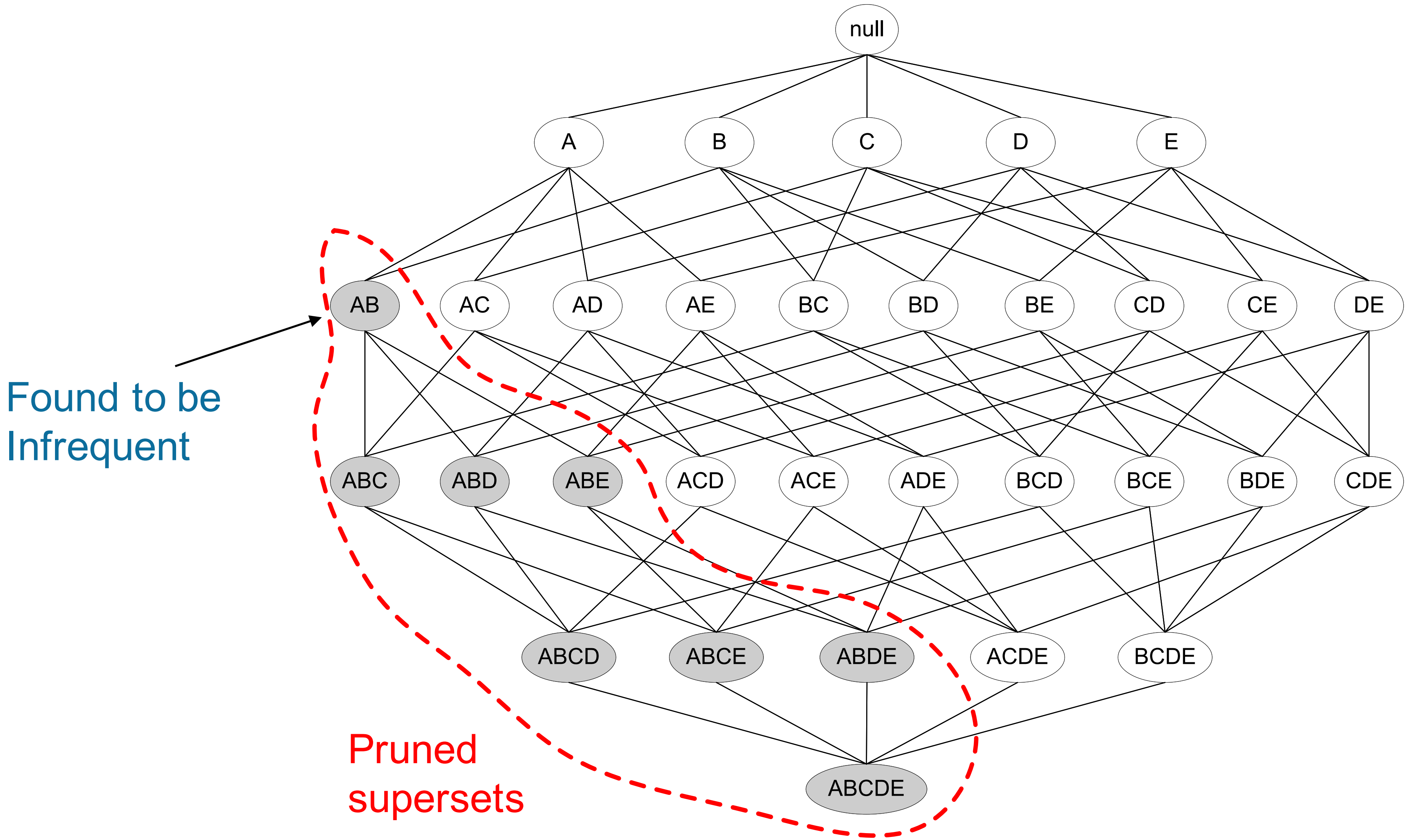
Two step approach

- Two-step approach:
  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 is still computationally expensive!

Searching for sets



Observation 2



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

Triplets (3-itemsets)

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

If every subset is considered,

$^6C_1 + {}^6C_2 + {}^6C_3 = 41$

With support-based pruning,

$6 + 6 + 1 = 13$

More formally

- Support is ***anti-monotone***
  - If an itemset X does not have support, no superset of X can have support

The 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 DB
  - Eliminate candidates that are infrequent, leaving only those that are frequent

The 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 – HOW?
  - Prune candidate itemsets containing subsets of length k that are infrequent
  - Count the support of each candidate by scanning the DB
  - Eliminate candidates that are infrequent, leaving only those that are frequent

Generating (k+1) itemsets from k-itemsets

- To generate a (k+1) itemset
  - Pick 2 k-itemsets that have the same (k-1) prefix
  - Merge them!
- Example
  - (A,B,C) and (A,B,D) are merged to form (A,B,C,D)

Rule generation

- Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement
  - If {A,B,C,D} is a frequent itemset, candidate rules:

$ABC \rightarrow D$,	$ABD \rightarrow C$,	$ACD \rightarrow B$,	$BCD \rightarrow A$,
$A \rightarrow BCD$,	$B \rightarrow ACD$,	$C \rightarrow ABD$,	$D \rightarrow ABC$
$AB \rightarrow CD$,	$AC \rightarrow BD$,	$AD \rightarrow BC$,	$BC \rightarrow AD$,
$BD \rightarrow AC$,	$CD \rightarrow AB$,		

- If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \emptyset$ and $\emptyset \rightarrow L$)

Rule generation

- How to efficiently generate rules from frequent itemsets?
  - In general, confidence does not have an anti-monotone property

    $c(ABC \rightarrow D)$ can be larger or smaller than $c(AB \rightarrow D)$

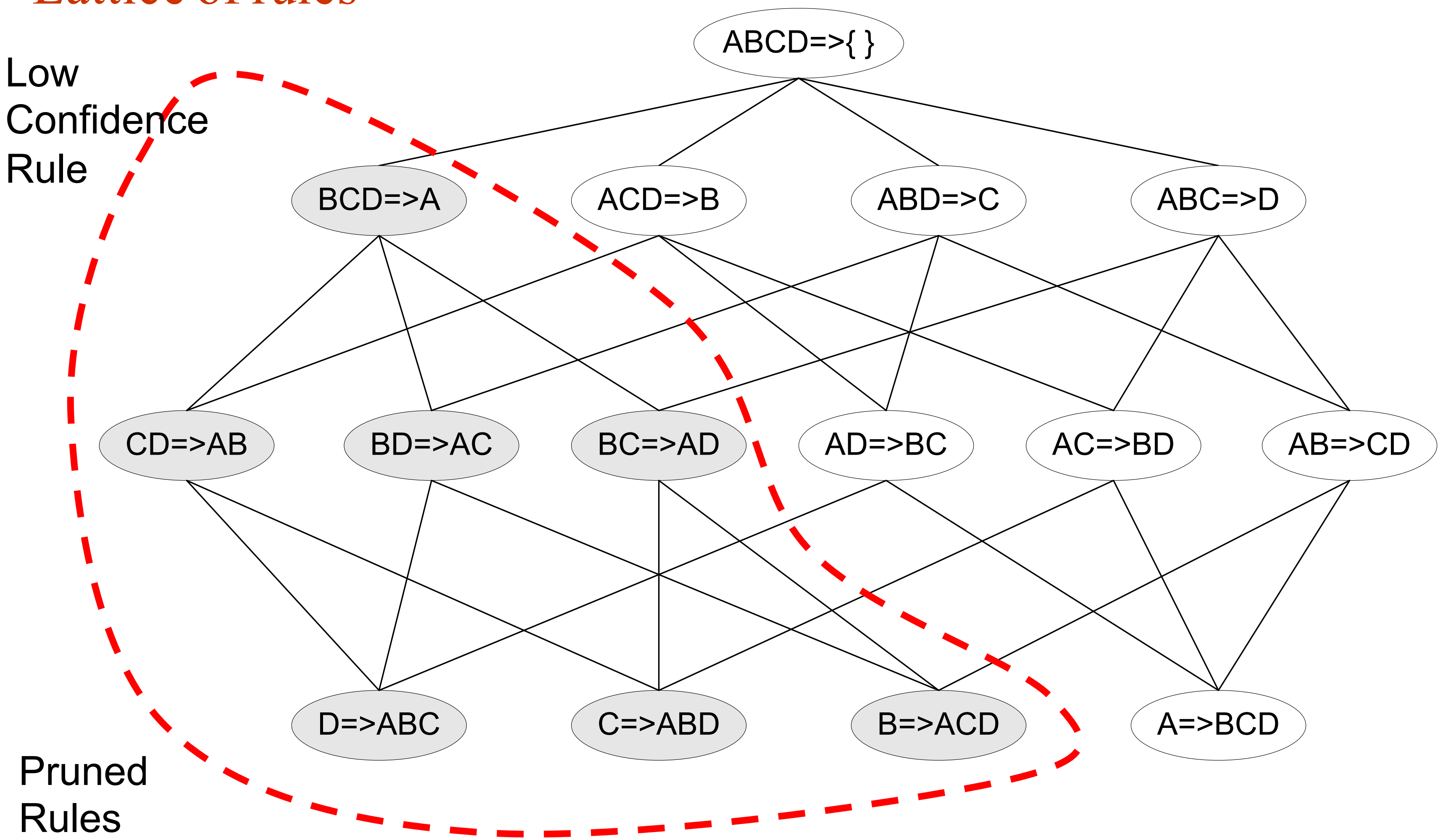
  - But confidence of rules generated from the same itemset has an anti-monotone property
  - e.g., $L = \{A,B,C,D\}$:

$$c(ABC \rightarrow D) \geq c(AB \rightarrow CD) \geq c(A \rightarrow BCD)$$

  - Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

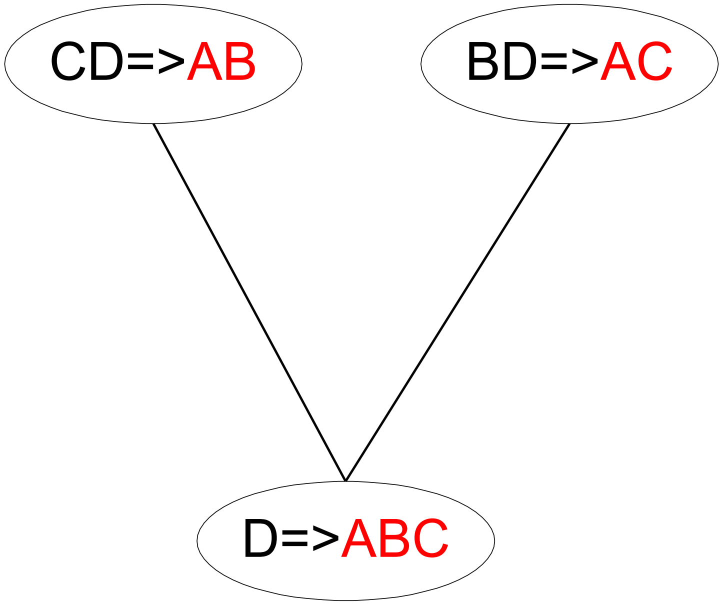
Rule generation (from a single itemset)

Lattice of rules



Rule generation details

- Candidate rule is generated by merging two rules that share the same prefix in the rule consequent

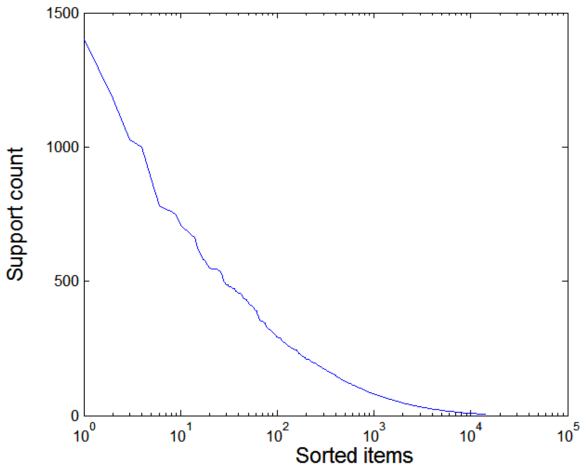


- join(CD=>AB,BD=>AC) would produce the candidate rule D => ABC

- Prune rule D=>ABC if its subset AD=>BC does not have high confidence

Practical issues

- Many real datasets have skewed support distributions
  - Too small support vs
  - Too large support



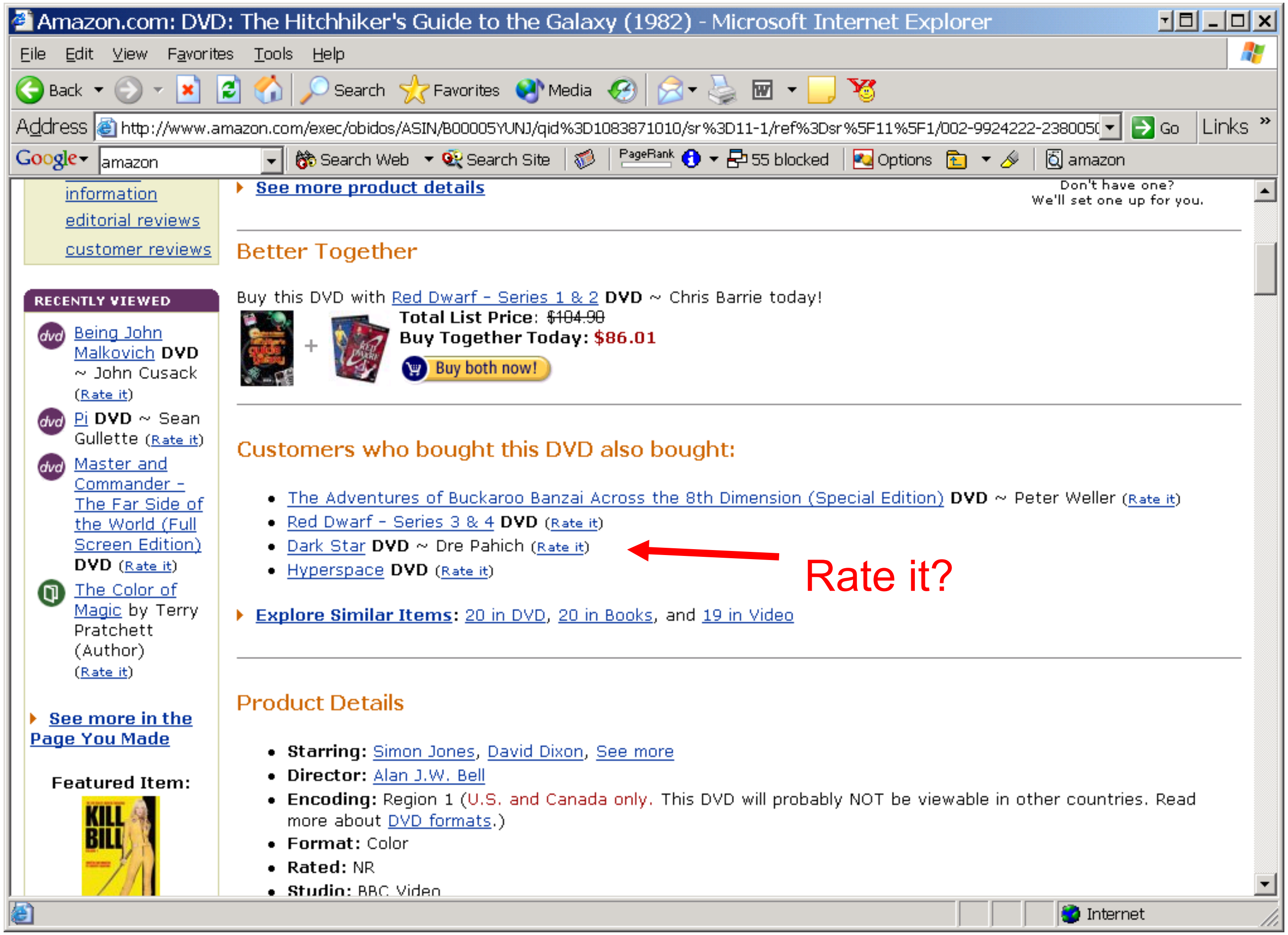
- Assocation analysis tends to produce too many rules!
  - Use interestingness measures to prune/select rules

What we have learnt thus far

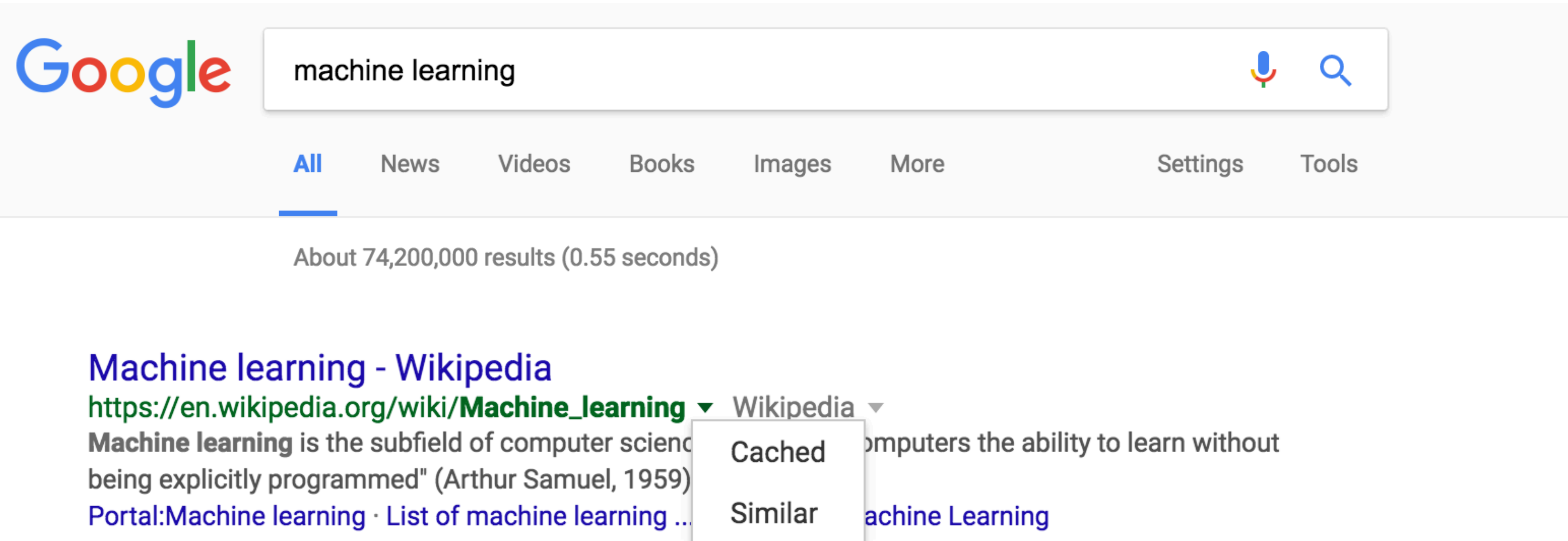
- Association analysis
  - A new age data mining problem
  - Data mining is simply “smart counting and book-keeping”
    - Efficiency of search is important!

Recommender systems

Recommender systems



Recommender systems



How do they work?

- Two basic threads
  - Content-based filtering
    - Tom likes aviation movies, so Tom will like Sully
  - Collaborative filtering
    - Tom likes the movies that Sally likes and Sally liked Sully, so Tom will like Sully
      - Suffers from “cold start” problems

Collaborative filtering

- Two basic flavors
  - User-based
  - Item-based
- Both are types of nearest neighbor reasoning! ☺

Example

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

First measure similarities between users

A popular similarity measure in user-based CF: Pearson correlation

$$sim(a,b) = \frac{\sum_{p \in P}(r_{a,p} - \bar{r}_a)(r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P}(r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P}(r_{b,p} - \bar{r}_b)^2}}$$

a, b : users

$r_{a,p}$: rating of user a for item p

P : set of items, rated both by a and b

Possible similarity values between -1 and 1; $\overline{r_a}, \overline{r_b}$ = user's average ratings

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

sim = 0,85
sim = 0,70
sim = -0,79

Make a prediction

- To predict the rating for user a for product p, find others who have rated p and scale their ratings by their similarity to a

$$pred(a,p) = \overline{r_a} + \frac{\sum_{b \in N} sim(a,b) * (r_{b,p} - \overline{r_b})}{\sum_{b \in N} sim(a,b)}$$

Item-based CF

- User-based CF has scalability issues if there are many more users than items
- Alternative idea
  - Find similarities between items

How this works

- Example
  - Look for items similar to item5
  - Take Alice's ratings for these items to predict her rating for item5

	Item1	Item2	Item3	Item4	Item5
Alice	5	3	4	4	?
User1	3	1	2	3	3
User2	4	3	4	3	5
User3	3	3	1	5	4
User4	1	5	5	2	1

How Amazon works

- Purportedly uses item-based collaborative filtering
- Pre-compute item similarities
  - They are more stable than user similarities
  - Neighborhood used at run-time is small since each user has rated only a small number of items

What we have learnt thus far

- Two broad classes of association methods
  - Itemset mining
  - Recommender systems
- We have seen only the most basic/vanilla versions of these methods
  - Significant variations and optimizations abound!