CPSC 340: Machine Learning and Data Mining

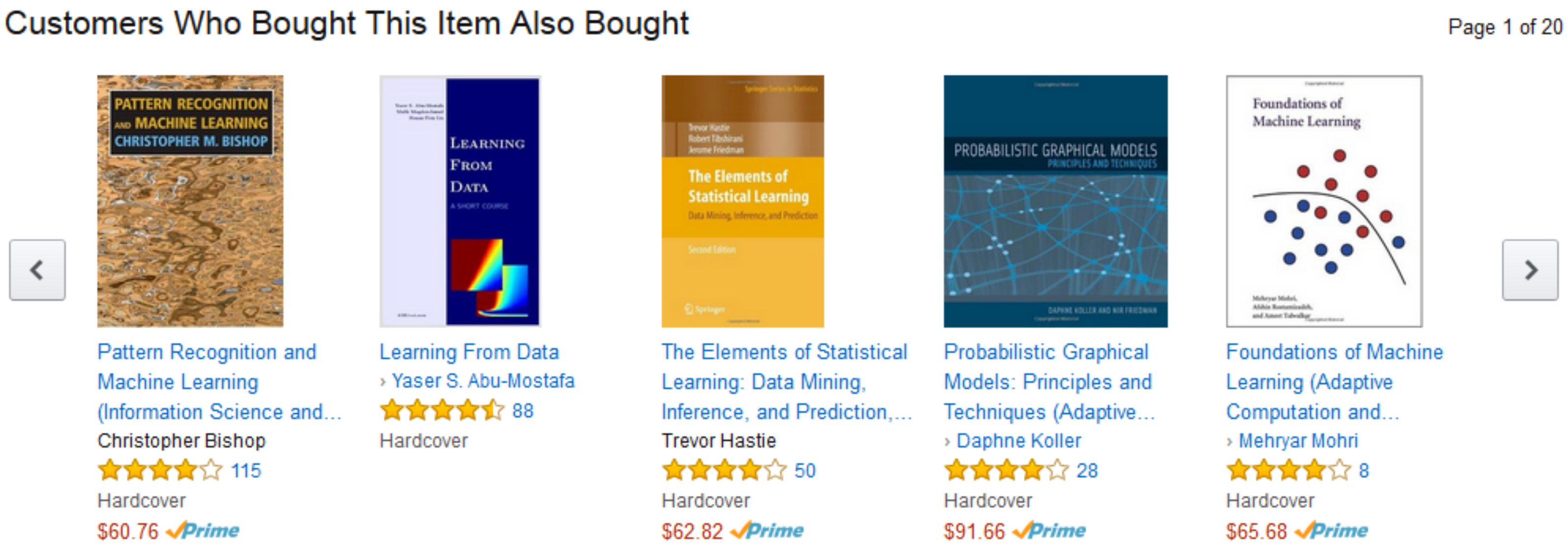
Association Rules

Admin

- Assignment 2 is due Sunday:
  - You should already be started!
- Tutorials today & tomorrow
  - hw2 python code, vector quantization

Motivation: Product Recommendation

- We want to find items that are frequently 'bought' together.



- With this information, you could:
  - Put them close to each other in the store.
  - Make suggestions/bundles on a website.

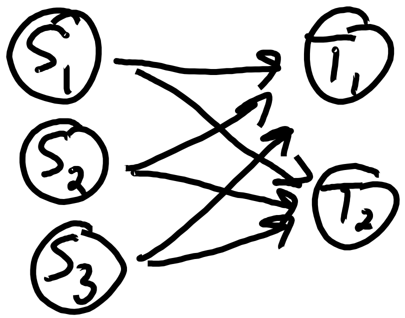
Association Rules

- Consider two sets of items 'S' and 'T':
  - For example: S = {sunglasses, sandals} and T = {sunscreen}.
- We're going to consider association rules (S => T):
  - If you buy all items 'S', you are likely to also buy all items 'T'.
  - E.g., if you buy sunglasses and sandals, you are likely to buy sunscreen.

Association Rules

- Interpretation in terms of conditional probability:
  - The rule (S => T) means that p(T = 1 | S = 1) is 'high'.

    I'm using p(T = 1 | S = 1) for p(T_1 = 1, T_2 = 1,…, T_k = 1 | S_1 = 1, S_2 = 1,…, S_c = 1).
- Association rules are directed but not necessarily causal:



  - p(T | S) ≠ p(S | T).
    - E.g., buying sunscreen doesn't necessarily imply buying sunglasses/sandals:
  - The correlation could be backwards or due to a common cause.
    - E.g., the common cause is that you are going to the beach.

Association Rules vs. Clustering

- Clustering:
  - Which objects are related?
  - Grouping rows together.

"These rows are in cluster 1"

$X =$

Sunglasses	Sandals	Sunscreen	Snorkel
1	1	1	0
0	0	1	0
1	0	1	0
0	1	1	1
1	0	0	0
1	1	1	1
0	0	0	0

Association Rules vs. Clustering

- Clustering:
  - Which objects are related?
  - Grouping rows together.
- Association rules:
  - Which features occur together?
  - Relating groups of columns.

X =

Sunglasses	Sandals	Sunscreen	Snorkel
1	1	1	0
0	0	1	0
1	0	1	0
0	1	1	1
1	0	0	0
1	1	1	1
0	0	0	0

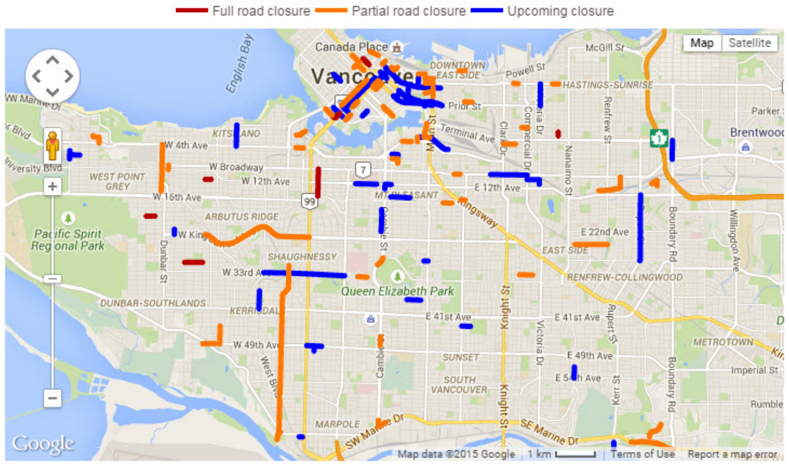
$\underbrace{\text{Sunglasses, Sandals}}_{S}$ $\underbrace{\text{Sunscreen}}_{T}$

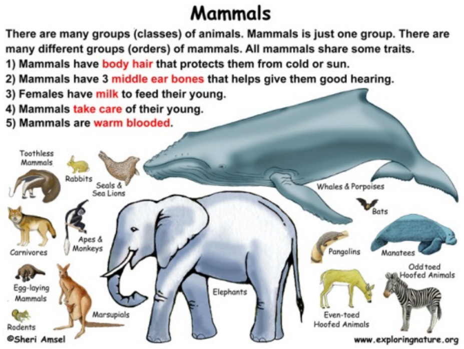
If these two columns are "1"

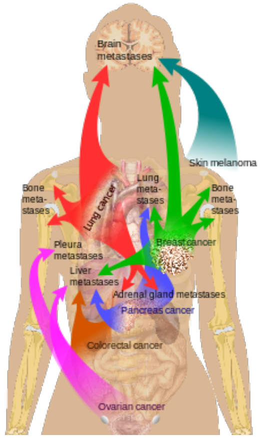
Then this value is probably "1"

Applications of Association Rules

- Which foods are frequently eaten together?
- Which genes are turned on at the same time?
- Which traits occur together in animals?
- Where do secondary cancers develop?
- Which traffic intersections are busy/closed at the same time?
- Which players outscore opponents together?





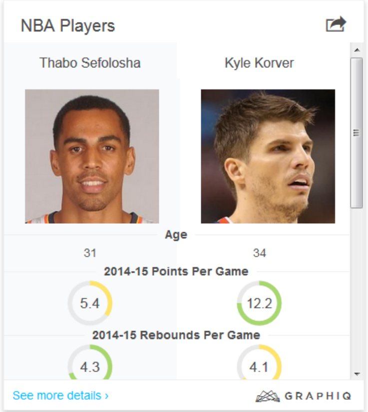


Atlanta Hawks #1

Minutes played together: 398
Combined net rating (per 48 minutes): 23.8
Overall rank among two-man lineups: 1st

Reaction: Against all odds, the most efficient tandem in the NBA is a pair of thirty-something wings. Kyle Korver and Thabo Sefolosha complement each other perfectly, with Korver providing the scoring punch and Sefolosha taking on the toughest defensive assignment for the Hawks.

With Sefolosha still getting back up to speed after a calf injury sidelined him for two months, the Hawks should probably just attach him to Korver until the two can get their chemistry back to how it was. Because any combination of players that can help a team outscore its opponents by 23.8 points per game is probably one worth exploring further.



http://www.exploringnature.org/db/view/624
https://en.wikipedia.org/wiki/Metastasis
http://basketball-players.pointafter.com/stories/3791/most-valuable-nba-duos#30-atlanta-hawks
http://modo.coop/blog/tips-from-our-pros-avoiding-late-charges-during-summer

Support and Confidence

- We "score" rule (S => T) by "support" and "confidence".
  - Running example: {sunglasses,sandals} => suncreen.
- Support:
  - How often does 'S' happen?
    - How often were sunglasses and sandals bought together?
  - Marginal probability: p(S = 1).

$p(S_1 = 1, S_2 = 1, \ldots, S_k = 1)$

- Confidence:
  - When 'S' happens, how often does 'T' happen?
  - When sunglasses+sandals were bought, how often was sunscreen bought?
  - Conditional probability: p(T = 1| S = 1).

Woes with notation/definitions

- In some books/sources, support is defined as on the previous slide
  - For example Wikipedia or *Mining of Massive Datasets*
- In other sources, support is defined as
  - How often does $S \cup T$ (S and T) happen?
    - How often were sunglasses, sandals and sunscreen bought together?
  - Joint probability: p(S = 1,T=1).
  - For example in *Database Management Systems* by Ramakrishnan & Gehrke
- Furthermore, in some texts, for a rule S=>T, T must be a single item. In other cases it can itself be a set.

Support and Confidence

- We're going to look for rules that:
  1. Happen often (high support), $p(S = 1) \geq$'s'.
  2. Are reliable (high confidence), $p(T = 1 | S = 1) \geq$ 'c'.

- Association rule learning problem:
  - Given support 's' and confidence 'c'.
  - Output all rules with support at least 's' and confidence at least 'c'.

- A common variation is to restrict size of sets:
  - Returns all rules with $|S| \leq k$ and/or $|T| \leq k$.
  - Often for computational reasons.

Finding Sets with High Support

- First let's focus on finding sets 'S' with high support ("frequent itemsets")
- How do we compute p(S = 1)?
  - If S = {bread, milk}, we count proportion of times they are both "1".

Bread	Eggs	Milk	Oranges	
1	1	1	0	→ yes
0	0	1	0	→ no
1	0	1	0	→ yes
0	1	0	1	→ no
…	…	…	…	⋮

$$p(S=1) = \frac{\#\text{ times all elements of 'S' are '1'}}{n}$$

Challenge in Learning Association Rule

- Consider the problem of finding all sets 'S' with $p(S = 1) \geq s$.
  - With 'd' features there are 2^d-1 possible sets.

For d=4 we have {1}, {2}, {3}, {4}, {1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {3,4}, {1,2,3}, {1,2,4}, {1,3,4}, {2,3,4}, {1,2,3,4}

- It takes too long to even write all sets unless 'd' is tiny.
- Can we avoid testing all sets?
  - Yes, using a basic property of probabilities…

Upper Bound on Joint Probabilities

- Suppose we know that $p(S = 1) \geq s$.
- Can we say anything about $p(S = 1, A = 1)$?
  - Probability of buying all items in 'S', plus another item 'A'.
- Yes, $p(S = 1, A = 1)$ cannot be bigger than $p(S = 1)$.

Because probabilities are non-negative $p(S=1, A=1) \leq p(S=1, A=1) + p(S=1, A=0)$

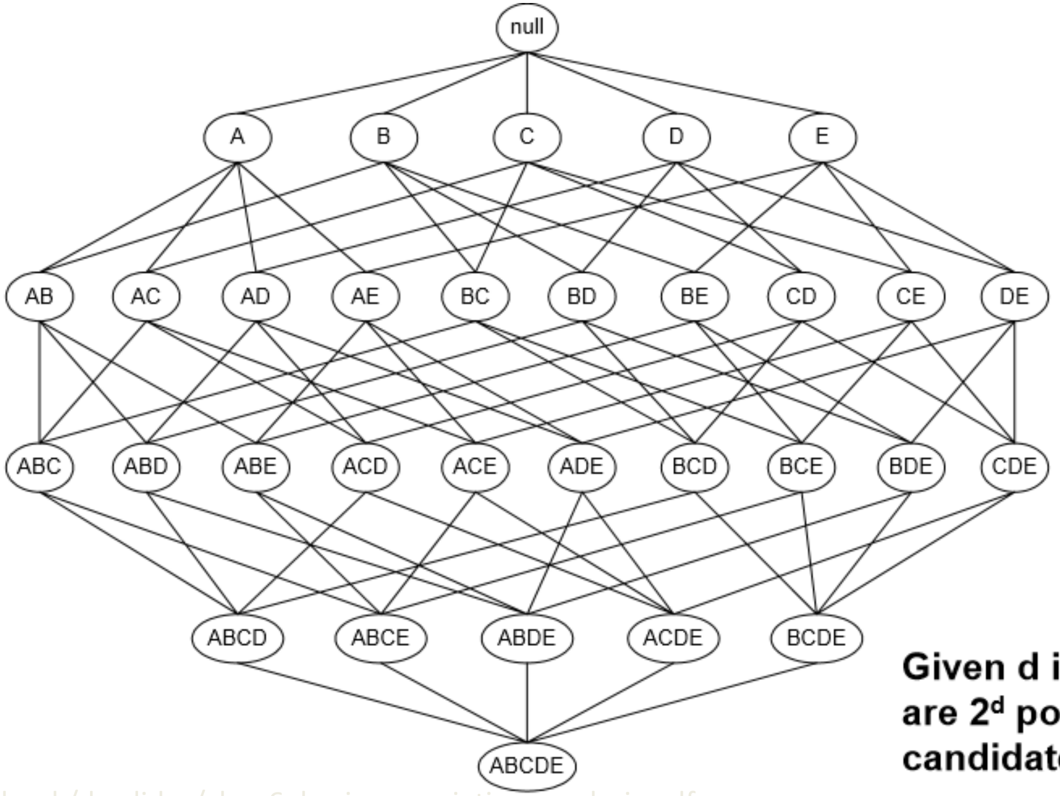
By the marginalization rule $p(S=1) = p(S=1, A=1) + p(S=1, A=0)$

Putting these together gives $p(S=1, A=1) \leq p(S=1)$

- E.g., probability of rolling {4,5} on 2 dice (1/36) is less than rolling 4 on one die (1/6).

Support Set Pruning

- This property means that $p(S_1 = 1) < s$ implies $p(S_1 = 1, S_2 = 1) < s$.
  - If p(sunglasses=1) < 0.1, then p(sunglasses=1,sandals=1) is less than 0.1.
  - We never consider $p(S_1 = 1, S_2 = 1)$ if $p(S_1 = 1)$ has low support.

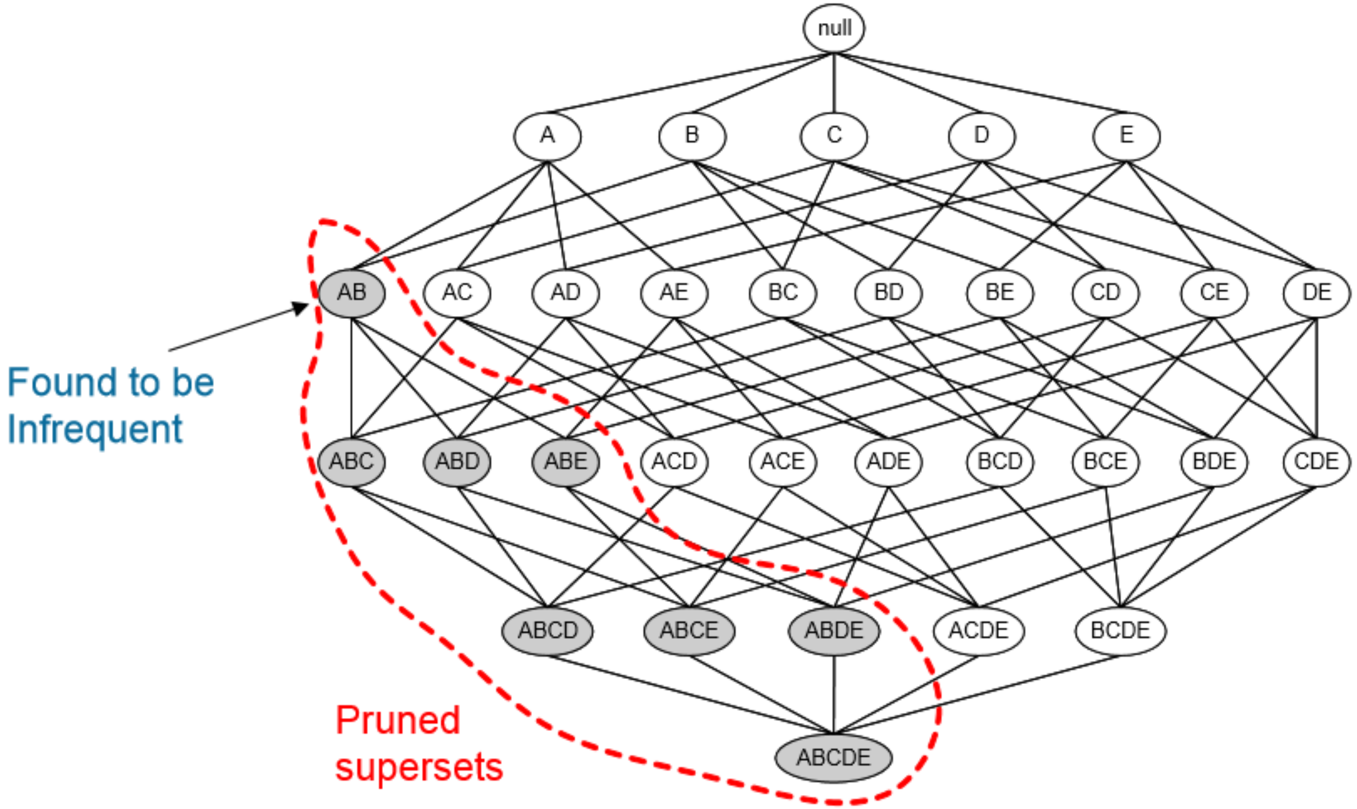


Given d items, there are 2^d possible candidate itemsets

http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap6_basic_association_analysis.pdf

Support Set Pruning

- This property means that $p(S_1 = 1) < s$ implies $p(S_1 = 1, S_2 = 1) < s$.
  - If p(sunglasses=1) < 0.1, then p(sunglasses=1,sandals=1) is less than 0.1.
  - We never consider $p(S_1 = 1, S_2 = 1)$ if $p(S_1 = 1)$ has low support.



$p(E)$ must be greater than

$p(D,E)$ which must be greater than

$p(C,D,E)$ which must be greater than

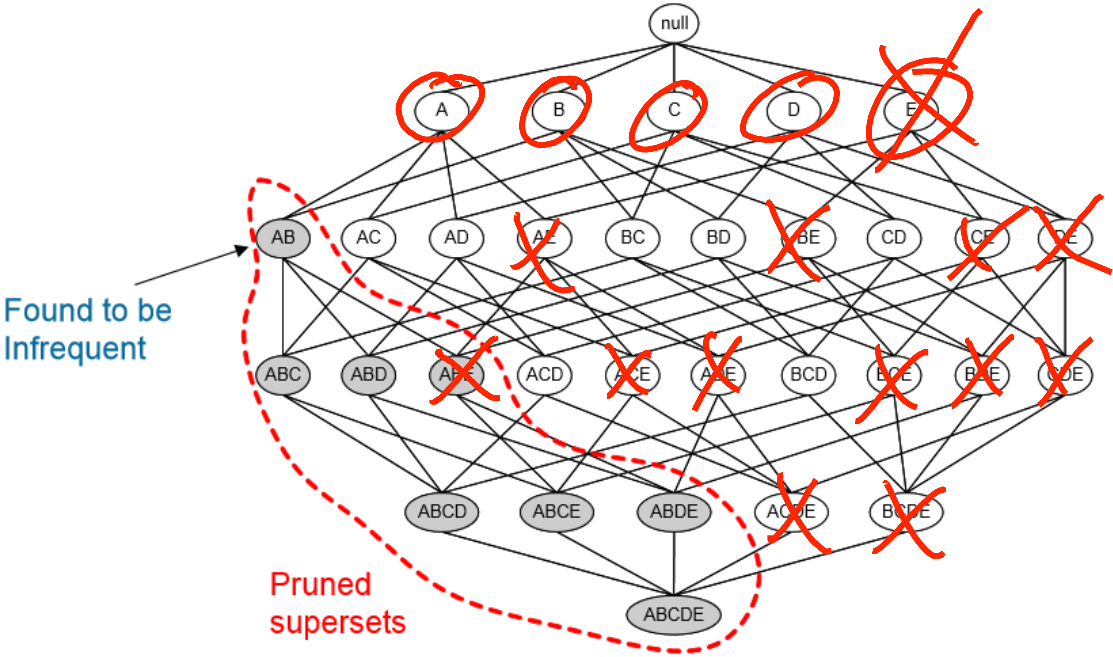
$p(B,C,D,E)$ which must be greater than

$p(A,B,C,D,E)$

http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap6_basic_association_analysis.pdf

A Priori Algorithm

- A priori algorithm for finding all subsets with p(S = 1) >= s.
  1. Generate list of all sets ‘S’ that have a size of 1.
  2. Set k = 1.
  3. Prune candidates ‘S’ of size ‘k’ where p(S = 1) < s.
  4. Add all sets of size (k+1) that have all subsets of size k in current list.
  5. Set k = k + 1 and go to 3.



http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap6_basic_association_analysis.pdf

A Priori Algorithm

Bread	Coke	Milk	Beer	Diaper	Eggs
1	0	1	0	1	0
0	1	0	1	1	1
1	0	1	0	1	1
⋮	⋮	⋮	⋮	⋮	⋮

Let's take minimum support as s = 0.30.

First compute probabilities for sets of size k = 1:

Item S	p(S = 1)
Bread	0.4
Coke	0.2
Milk	0.4
Beer	0.3
Diaper	0.4
Eggs	0.1

Bread, milk, diaper, beer have support at least 's'!

A Priori Algorithm

Bread	Coke	Milk	Beer	Diaper	Eggs
1	0	1	0	1	0
0	1	0	1	1	1
1	0	1	0	1	1
⋮	⋮	⋮	⋮	⋮	⋮

Let's take minimum support as s = 0.30.

First compute probabilities for sets of size k = 1:

Item S	p(S=1)
Bread	0.4
Coke	0.2
Milk	0.4
Beer	0.3
Diaper	0.4
Eggs	0.1

Bread, milk, diaper, beer have support at least 's'!

Combine sets of size k=1 with support 's' to make sets of size k = 2:

We don't check rules with coke or eggs.

Itemset S	p(S=1)
{Bread, Milk}	0.3
{Bread, Beer}	0.2
{Bread, Diaper}	0.3
{Milk, Beer}	0.2
{Milk, Diaper}	0.3
{Beer, Diaper}	0.3

{Bread, Milk}, {Bread, Diaper}, {Milk, Diaper}, {Beer, Diaper} have support at least 's'!

A Priori Algorithm

Bread	Coke	Milk	Beer	Diaper	Eggs
1	0	1	0	1	0
0	1	0	1	1	1
1	0	1	0	1	1
⋮	⋮	⋮	⋮	⋮	⋮

Let's take minimum support as s = 0.30.

First compute probabilities for sets of size k = 1:

Item S	$p(S=1)$
Bread	0.4
Coke	0.2
Milk	0.4
Beer	0.3
Diaper	0.4
Eggs	0.1

Bread, milk, diaper, beer have support at least 's'.

Combine sets of size k=1 with support 's' to make sets of size k = 2:

We don't check rules with coke or eggs.

Itemset S	$p(S=1)$
{Bread, Milk}	0.3
{Bread, Beer}	0.2
{Bread, Diaper}	0.3
{Milk, Beer}	0.2
{Milk, Diaper}	0.3
{Beer, Diaper}	0.3

{Bread, Milk}, {Bread, Diaper}, {Milk, Diaper}, {Beer, Diaper} have support at least 's'.

Check sets of size k = 3 where all subsets of size k = 2 have high support:

Itemset	$p(S=1)$
{Bread, Milk, Diaper}	0.3

(All other 3-item and higher-item counts are < 0.3)

(We only considered 13 out 63 possible rules.)

A Priori Algorithm Discussion

- Some implementations only return 'Maximal frequent subsets':
  - Only return sets S with $p(S = 1) \geq s$ where no superset S' has $p(S' = 1) \geq s$.
  - E.g., don't return {break,milk} if {bread, milk, diapers} also has high support.

- Number of rules we need to test is hard to quantify:
  - Need to test more rules for small 's'.
  - Need to test more rules as counts increase.
- Computing $p(S = 1)$ if S has 'k' elements costs $O(nk)$.
  - But there is some redundancy:
    - Computing $p(\{1,2,3\})$ and $p(\{1,2,4\})$ can re-use some computation.
  - Hash trees can be used to speed up various computations.

Generating Rules

- A priori algorithm gives all 'S' with $p(S = 1) \geq s$.
- To generate the rules, we consider subsets of frequent itemsets
  - If {1,2,3} is a frequent itemset, candidate rules involving these items are:
    - {1} => {2,3}, {2} => {1,3}, {3} => {1,2}, {1,2} => {3}, {1,3} => {2}, {2,3} => {1}.
  - There is an exponential number of subsets.
- But we can again prune using rules of probability:

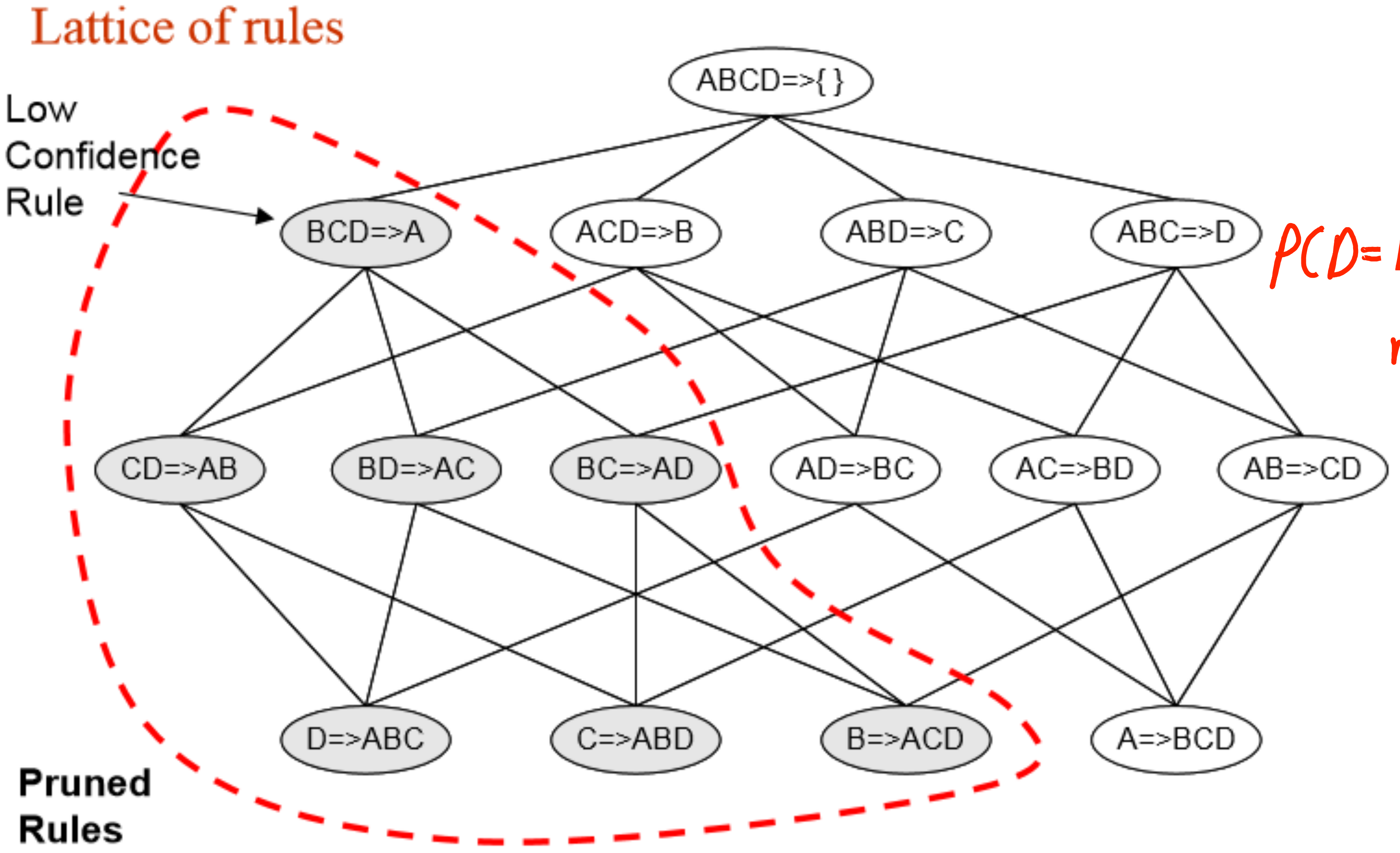
By definition of conditional probability we have $p(T=1 \mid S=1) = \frac{p(S=1, T=1)}{p(S=1)}$

And since $p(S=1) \leq 1$ we have $p(T=1 \mid S=1) \geq p(S=1, T=1)$

By the same logic we have $p(T=1, R=1 \mid S=1, Q=1) \geq p(T=1, R=1, Q=1 \mid S=1)$

- E.g., probability of rolling 2 sixes is higher if you know one die is a 6.

Confident Set Pruning



$P(D=1 | A=1, B=1, C=1)$

must be greater than

$P(C=1, D=1 | A=1, B=1)$

- Or… computation is very fast if T can only be a single item

http://www-users.cs.umn.edu/~kumar/dmbook/dmslides/chap6_basic_association_analysis.pdf

Association Rule Mining Issues

- Spurious associations:
  - Can it return rules by chance?
- Alternative scores:
  - Support score seems reasonable.
  - Is confidence score the right score?
- Faster algorithms than a priori:
  - ECLAT/FP-Growth algorithms.
  - Generate rules based on subsets of the data.
  - Cluster features and only consider rules within clusters.
  - Amazon's recommendation system.

Spurious Associations

- For large ‘d’, high probability of returning spurious associations:
  - With random data, one of the 2^d rules is likely to look strong.

- Classical story:
  - *In 1992, Thomas Blischok, manager of a retail consulting group at Teradata, and his staff prepared an analysis of 1.2 million market baskets from about 25 Osco Drug stores. Database queries were developed to identify affinities. The analysis "did discover that between 5:00 and 7:00 p.m. that consumers bought beer and diapers". Osco managers did NOT exploit the beer and diapers relationship by moving the products closer together on the shelves.*

Problem with Confidence

- Consider the “Sunscreen Store”:
  - Most customers go there to buy sunscreen.
- Now consider rule (sunglasses => sunscreen).
  - If you buy sunglasses, it could mean you weren’t there for sunscreen:
    - p(sunscreen = 1| sunglasses = 1) < p(sunscreen = 1).
  - So (sunglasses => sunscreen) could be a misleading rule:
    - You are less likely to buy sunscreen if you buy sunglasses.
  - But the rule could have high confidence.
- Example:
  - p(sunscreen) = 0.9 (marginal probability)
  - p(sunglasses) = 0.2 (marginal probability)
  - p(sunscreen and sunglasses) = 0.1 (joint probability)
  - This means p(sunscreen | sunglasses) = 0.1/0.2 = 0.5 (conditional probability)


Customers who bought sunglasses
Customers who didn’t buy sunglasses


Customers who bought sunglasses
Customers who bought sunscreen
Customers who didn't buy sunglasses
Most customers buy sunscreen

Customers who bought sunglasses are still likely to buy sunscreen.

Most customers buy sunscreen.

Customers who bought sunglasses are still likely to buy sunscreen.

Most customers buy sunscreen

But knowing that they bought sunglasses make it less likely they bought sunscreen.

Customers who bought sunglasses are still likely to buy sunscreen.

Most customers buy sunscreen.

But knowing that they bought sunglasses make it <u>less</u> likely they bought sunscreen.

Normalize by probability of buying if you <u>don't</u> know 'S'.

- One alternative to confidence is “lift”:
  - How much more likely does ‘S’ make us to buy ‘T’?

Confidence

$$\text{Lift}(S \Rightarrow T) = \frac{p(T=1 \mid S=1)}{p(T=1)}$$

Amazon Recommendation Algorithm

- How can we scale to millions of users and millions of products?
  - Only consider rules (S => T) where S and T have a size of 1.
  - For each item, construct bag of users vector x_i.
  - Recommend items ‘j’ with high cosine similarity:

$$\cos(x_i, x_j) = \frac{\sum_{k=1}^{d} x_{ik} x_{jk}}{\|x_i\| \, \|x_j\|}$$

    - If $\cos(x_i,x_j) = 1$, products were bought by exact same users.

$$X = \underbrace{\left[\quad \right]}_{\text{user}} \Big\} \text{product}$$

Summary

- Association Rules: (S => T) means seeing S means T is likely.
- Support: measure of how often we see S.
- Confidence: measure of how often we see T, given we see S.
- A priori algorithm: use inequalities to prune search for rules.
- Amazon product recommendation:
  - Simpler method used for huge datasets in practice.

- Next time: how do we do supervised learning with a *continuous* y_i?

Bonus Slide: Sequential Pattern Analysis

- Finding patterns in data organized according to a sequence:
  - Customer purchases:
    - ‘Star Wars’ followed by ‘Empire Strikes Back’ followed by ‘Return of the Jedi’.
  - Stocks/bonds/markets:
    - Stocks going up followed by bonds going down.
- In data mining, called sequential pattern analysis:
  - If you buy product A, are you likely to buy product B at a later time?
- Similar to association rules, but now order matters.
  - Many issues stay the same.
- Exist sequential versions of many association rule methods:
  - Generalized sequential pattern (GSP) algorithm is like a priori algorithm.