CPSC 340:
Machine Learning and Data Mining

Association Rules

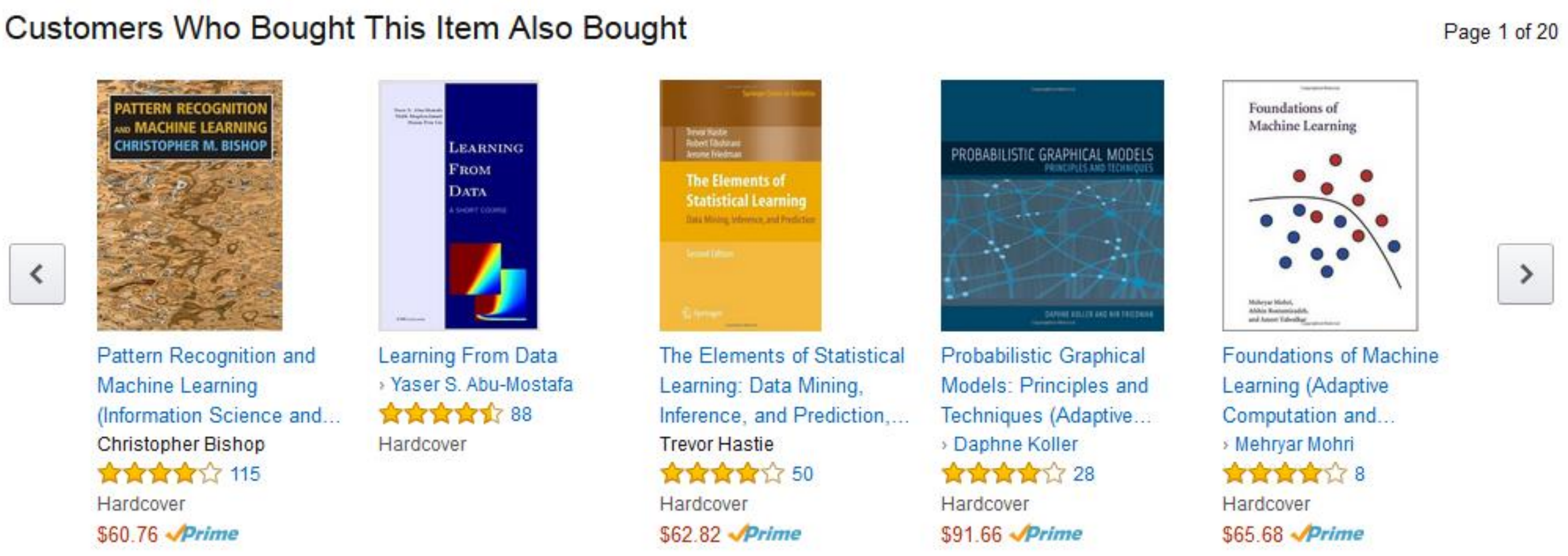
Fall 2015

Admin

- Assignment 2 due now.
- Assignment 3 out Monday.
  - Change the due date to have an extra tutorial?
  - Probably means moving the midterm back.
- Review your calculus and linear algebra before Monday!

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}.
- An association rule (S => T) has the interpretation:
  - 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 | S) is 'high'.
- Association rules are directed but not necessarily causal:
  - Buying suncreen doesn't necessarily imply buying sunglasses/sandals:
    - $p(T \mid S) \neq p(S \mid T)$.
  - You are not necessarily buying sunscreen because you bought sunglasses/sandals:
    - There is a common cause: you are going to the beach.

Association Rules vs. Clustering

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

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

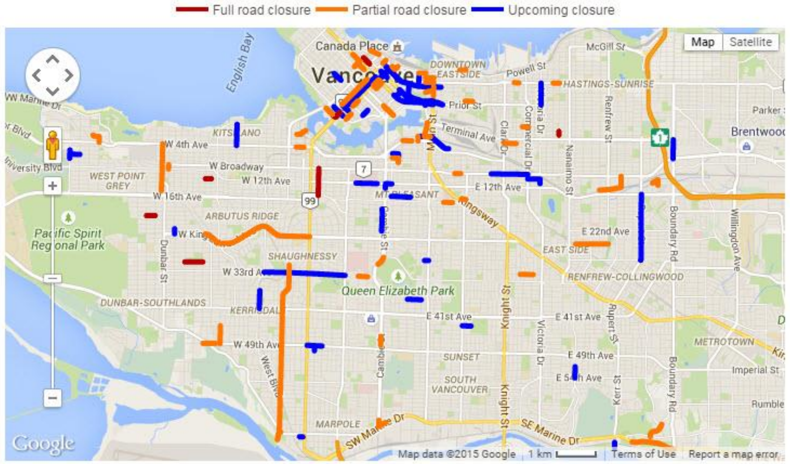
Association Rules vs. Clustering

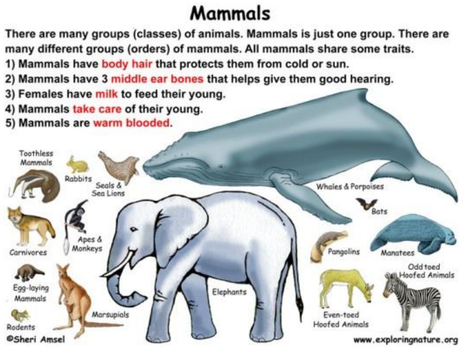
- Clustering:
  - Which objects are related?
  - Grouping rows together.
- Association rules:
  - Which features occur together?
  - Relationship between columns.

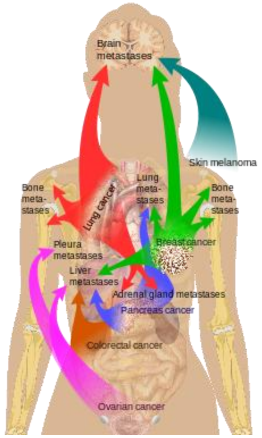
Sunglasses	Sandals	Sunscreen	Snorkel
1	1	1	0
0	0	1	0
1	0	1	0
0	1	1	1
1	0	0	0
1	1	1	1
0	0	0	0

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?





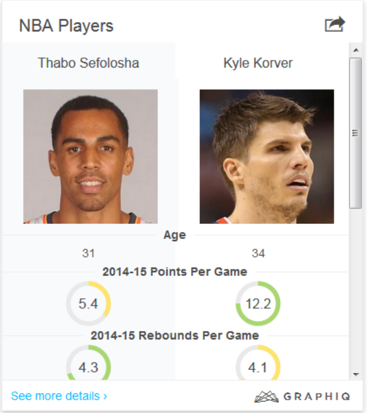


Atlanta Hawks #1

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 measure ‘strength’ of rule (S => T) by ‘support’ and ‘confidence’.
  - Running example: ({sunglasses,sandals} => suncreen).
- Support:
  - How often does ‘S’ happen?
  - In example: how often were sunglasses and sandals bought together?
  - Marginal probability: $p(S)$.
- Confidence:
  - When ‘S’ happens, how often does ‘T’ happen?
  - In example: when sunglasses and sandals were bought together, how often was sunscreen also bought?
  - Conditional probability: $p(T \mid S)$.

Support and Confidence

- Support: does ‘S’ happen enough to be worth considering?
- Confidence: how often is (S => T) true?

- Association rule learning algorithm:
  - Input: minimum support ‘s’ and minimum 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 $|T| \leq k$.

Challenge in Learning Association Rule

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

For d=4, 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 would take too long to even write all sets unless 'd' is tiny.
- Can we somehow avoid testing all sets?
- Yes, using 'downward-closure'/'anti-monotonicity' property:
  - Fancy DM names for basic property of probabilities…

Upper Bound on Joint Probailities

- Consider a set 'S', where we have computed support p(S).
- Now we want to know support of S plus a new variable A, p(S,A).
- We can derive an upper bound on p(S,A) in terms p(S):

$$p(S) = \sum_{x} p(S, x)$$ (marginalization rule)

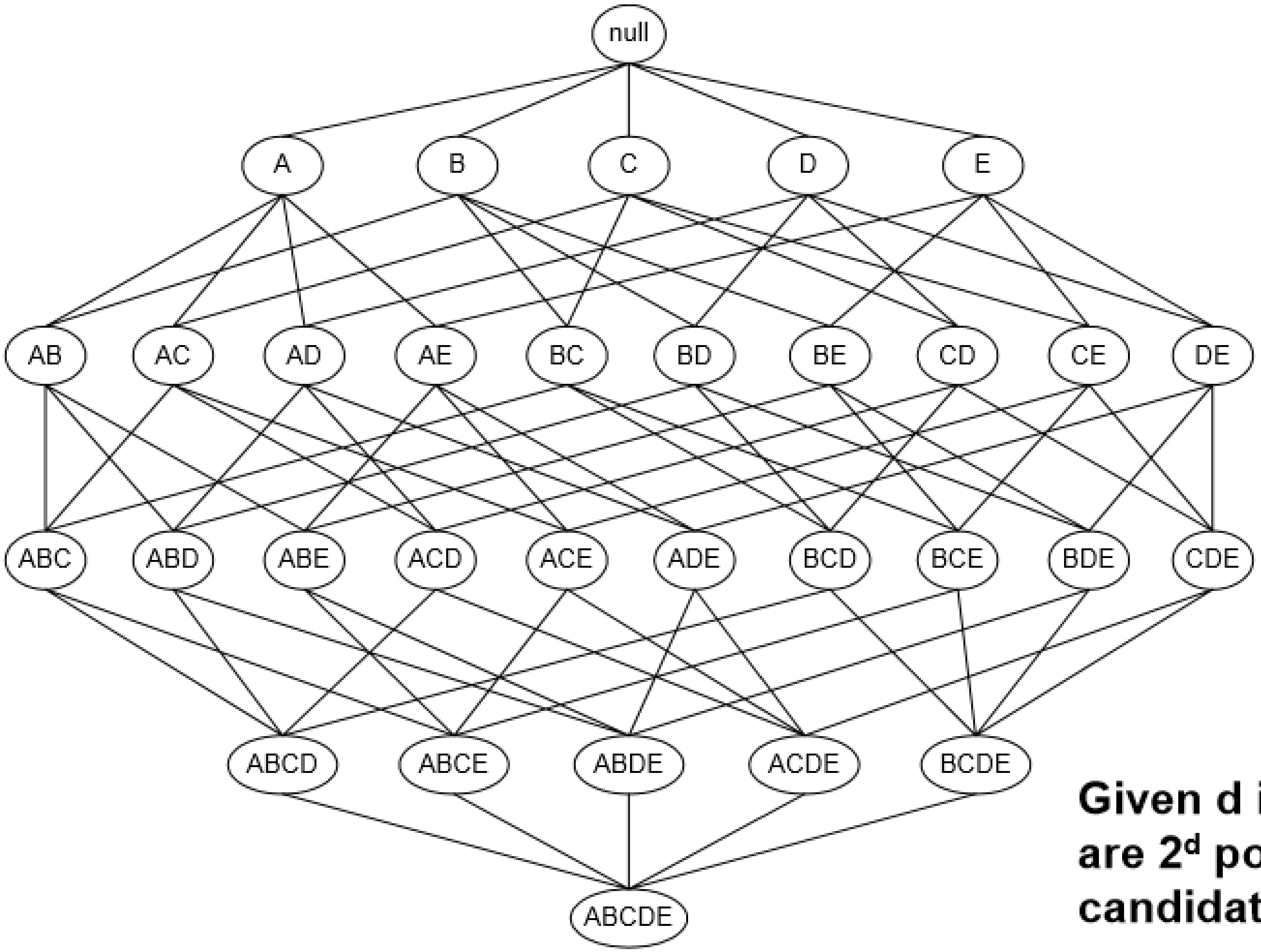
For any 'A' $$\sum_{x} p(S,x) \geq p(S,A)$$ (probabilities are non-negative, and A is a 'specific' x)

Therefore, $$\boxed{p(S) \geq p(S,A)}$$

Support Set Pruning

- Because we have $p(S) \geq p(S,A)$ we have the following property:
  - If support of 'S' is not big enough, support of 'S' with 'A' is not big enough.
  - If $(p(S) < s)$, then $(p(S,A) < s)$.
  - If $p(\text{sunglasses}) < 0.1$, then $p(\text{sunglasses},\text{sandals})$ must be less than 0.1.
- We never need to consider $p(S,A)$ if $p(S)$ has low support.

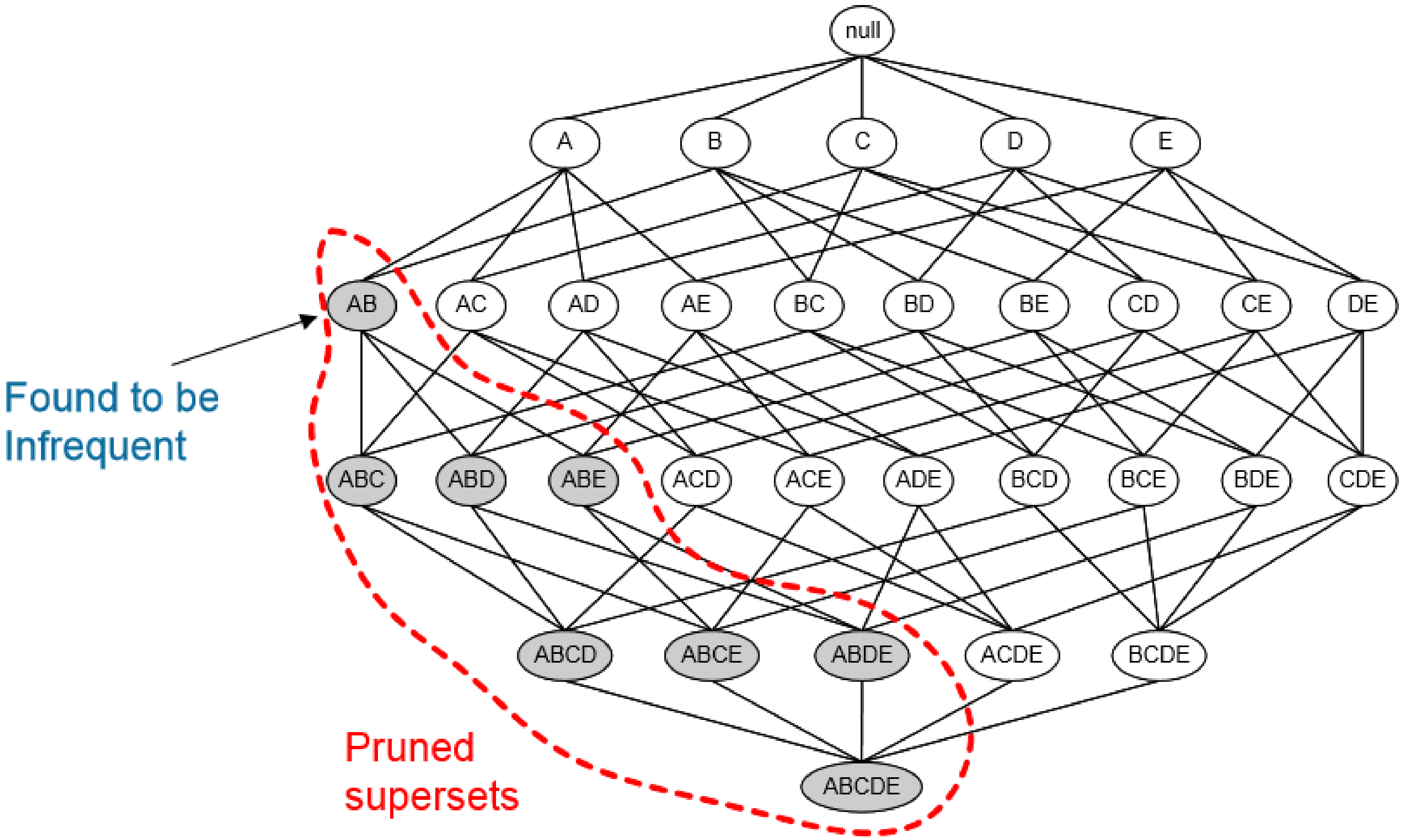
Support Set Pruning



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



A Priori Algorithm

- A priori algorithm for finding all subsets with p(S) >= 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) < s.
  4. Generate 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.

A Priori Algorithm

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

Minimum Support = 3

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

(No need to generate candidates involving Coke or Eggs)

Itemset	Count
{Bread,Milk,Diaper}	3

(We only considered 13 out 64 possible rules.)

A Priori Algorithm

- A priori algorithm for finding all subsets with $p(S) \geq s$.
  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 $p(S) < s$.
  4. Generate 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.

- We might prune the output:
  - 'Maximal frequent subsets':
    - sets S with $p(S) \geq s$ but no superset S' has $p(S') \geq s$.
  - 'Closed frequent subsets':
    - sets S with $p(S) \geq s$ but no superset S' has $p(S') = p(S)$.

Cost of A Priori Algorithm

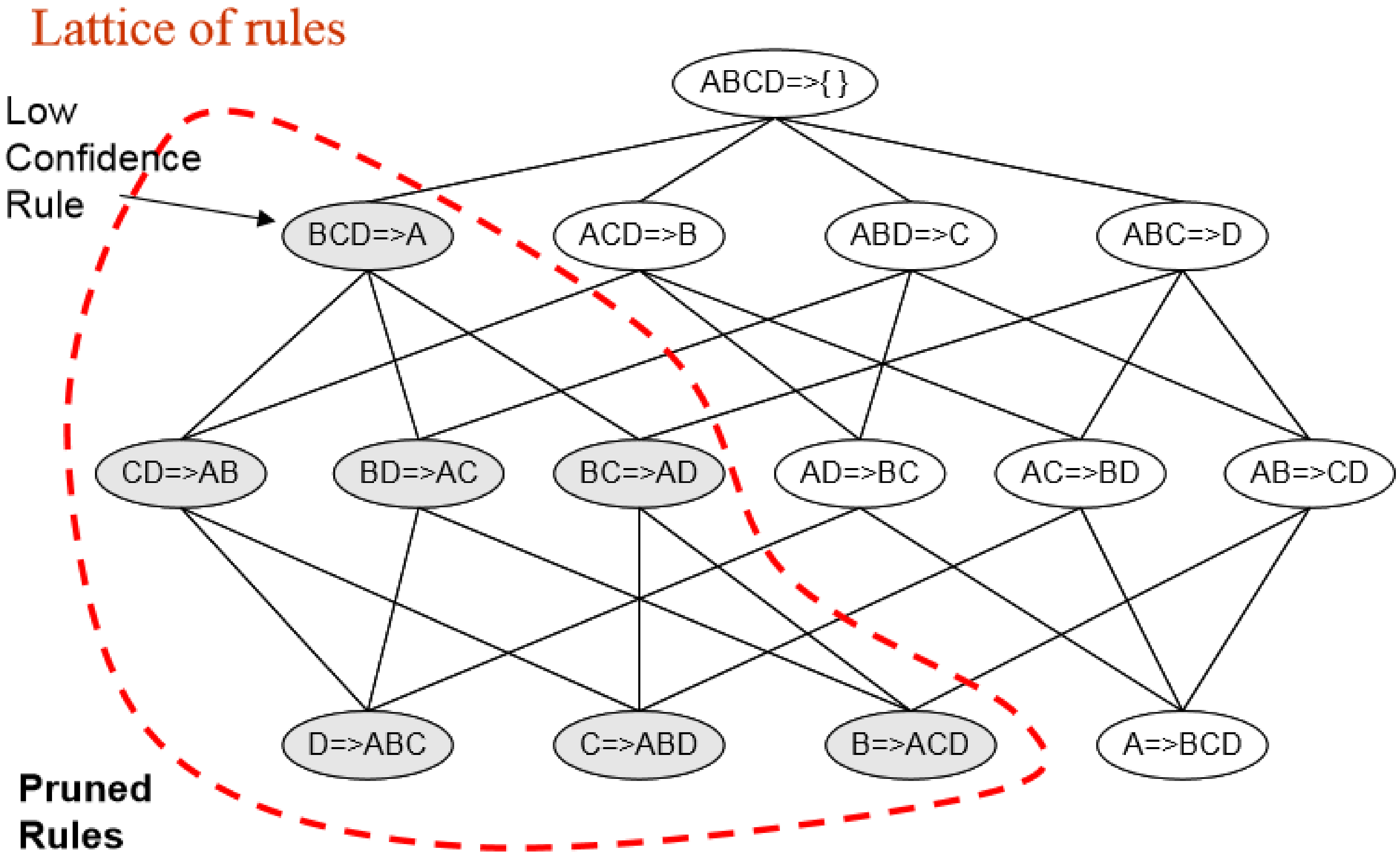
- Number of rules is hard to quantify:
  - But number of rules decreases as support threshold ‘s’ increases.
  - It also decrease as the number zeroes in each row increases.
- Computing p(S) if S has ‘k’ elements costs O(nk).
  - But there is some redundancy:
    - Computing p({1,2,3}) and p({1,2,4}) can re-use some computation.
  - Hash trees can be used to speed up the computation.
  - Hash tree can also be used to speed up finding all the subsets.

Generating Rules

- A priori algorithm gives all 'S' with p(S) ≥ s.
- Given such an S, generate candidate rules as subsets.
  - If S = {1,2,3}, candidate rules 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 probabilistic inequality:

$p(A|B) = \frac{p(A,B)}{p(B)} \geq p(A,B)$, and by same logic: $p(\{1,2\}|\{3,4\}) \geq p(\{1,2,3\}|\{4\})$.

Confident Rule Pruning



Association Rule Mining Issues

- Spurious associations:
  - Can it return rules by chance?
- Alternative measures:
  - Support p(S) seems reasonable.
  - Is confidence p(T | S) the right score?
- Faster algorithms than 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 correlations:
  - Even 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."

Alternatives to Confidence

- Consider the ‘sunscreen’ store:
  - Most customers go there to buy sunscreen.
  - But they also sell other beach related items.
- Consider the rule (sunglasses => sunscreen).
  - This could have high support and confidence.
  - But if you buy sunglasses, it could you mean weren’t there for sunscreen:
    - $p(\text{sunscreen} \mid \text{sunglasses}) < p(\text{sunscreen})$.
  - This is a bad rule.
- One of the (many) alternatives to confidence is ‘lift’:
  - $\text{Lift}(S \Rightarrow T) = p(S,T)/(p(S)p(T))$.
  - “How often they occur together vs. how often they would if independent.”

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.
  - Only consider sets S and T that have previously been bought together.
  - For each item, construct 'bag of users' vector x_i.
  - Recommend items with highest cosine similarity:

$$\cos(x_i, x_j) = \frac{x_i^T x_j}{||x_i|| \cdot ||x_j||}$$

(Maximum value of 1 means that products bought by exact same users.)

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's product recommendation: simpler methods are used for huge datasets in practice.

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