FP-Growth Algorithm, Rule Generation & Evaluation

ITCS 6163/8163
UNCC, Spring 2010

Outline

- Association rules & mining
- Finding frequent itemsets
  - Apriori algorithm
  - Compact representations of frequent itemsets
  - Alternative discovery methods
  - FP-Growth algorithm
- Generating association rules
- Evaluating discovered rules

Last time: Association rules & mining; Finding frequent itemsets (Apriori algorithm, Compact representations of frequent itemsets, Alternative discovery methods)

Today: FP-Growth algorithm; Generating association rules; Evaluating discovered rules

FP-Growth Algorithm: Key Ideas

- Limitations of Apriori (or generate-and-test paradigm)
  - Need to consider a large number of candidates (e.g., 1K frequent 1-itemsets ➔ ~1M 2-itemset candidates)
  - Need to scan the database many times (once each iteration)
- FP-Growth (frequent pattern growth)
  - Does not generate candidates
  - Typically just need to scan database twice

FP-Growth Algorithm

1. Build FP-Tree
   - Scan database to discover frequent 1-itemsets
   - Set the order of items in the transaction as the order of decreasing support, e.g., A, B, C, D (s(A) > s(B) > …)
   - Scan database again to build a compact representation of transactions in form of FP-Tree
2. Discover frequent itemsets using FP-Tree
   - Recursively find frequent itemsets with common suffix, ending with items having lower support first
   - E.g., finding itemsets ending with D, C, B, A, …

Why this order?

Why lower support first?

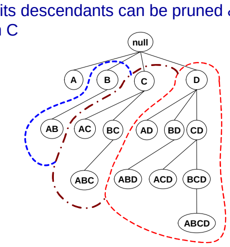
Why Order Items in Decreasing Support?

- So that transactions start with high-support items
  - Many transactions tend to share common prefix
  - Smaller branching factors, less bushy
  - ➔ Smaller tree (with fewer nodes)
- But this is just a heuristics & might not always work
  - Exercise: find an example where ordering items by increasing support produces smaller tree

Why Suffixes with Low Support First?

- Reverse order of items: D, C, B, A
  - Find all frequent itemsets ending w/ D, then C, B, and A
- More opportunities for pruning
  - If D has lower support than C, it is more likely to be pruned
  - If D is pruned, all its descendants can be pruned & D has more descendants than C

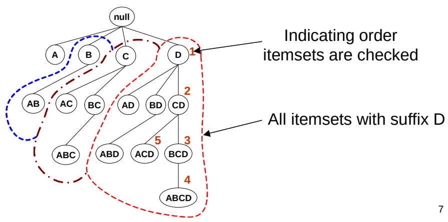
Item order:
A, B, C, D



Depth-First Traversal by FP-Algorithm

- Find all frequent itemsets ending w/ D, then C, B, and A
  - To find frequent itemsets ending w/ D, check if D is frequent; if yes, find frequent itemsets ending w/ CD, BD, and AD
  - To find frequent itemsets ending w/ CD, check if CD is frequent, if yes, find frequent itemsets ending w/ BCD and ACD
  - …

Item order:
A, B, C, D



Determine Item Order

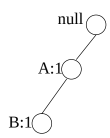
- Minimum support count = 2
- Scan database to find frequent 1-itemsets
  - s(A) = 8, s(B) = 7, s(C) = 5, s(D) = 5, s(E) = 3
- Item order (decreasing support): A, B, C, D, E

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

FP-Tree: Nodes & Paths

- Node = item, each transaction mapped to a path
- Consider node X whose path from root is p = $I_1I_2\ldots I_k$
  - Count associated with X = # of transactions with prefix p

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

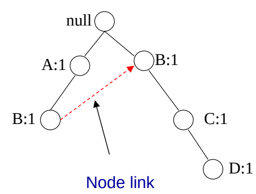


Tree after reading 1st transaction

FP-Tree: Node Links

- Nodes for the same item at different paths are connected via node links, to speed up computation of support counts

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

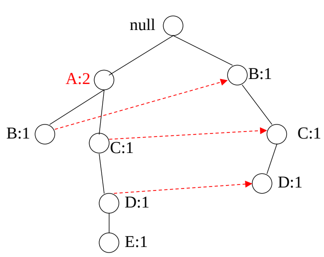


After reading 2nd transaction

Storing Transactions with Common Prefix

- Note the shared path for 1st & 3rd transaction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}



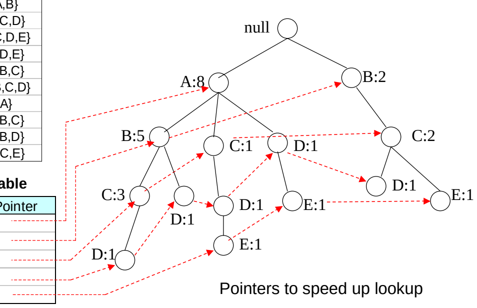
After reading the first 3 transactions

Complete FP-Tree for Sample Transactions

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{A}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Header table

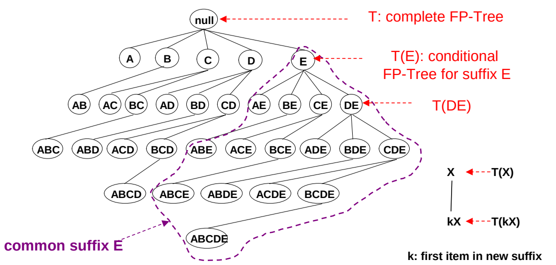
Item	Pointer
A	
B	
C	
D	
E	



Pointers to speed up lookup

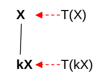
Discover Frequent Itemsets w/ FP-Tree

- T for discovering all frequent itemsets, T(E) for discovering all frequent itemsets ending with E, …



Obtaining Conditional FP-Trees

- Obtain T(kX) from T(X): conditional FP-Tree for suffix X
  - Obtain T'(kX) from T(X) by retaining only paths ending with k
  - Compute support count of k. If k is infrequent, stop & backtrack
  - Update counts in T' to consider only paths ending with k
  - Remove leaf nodes (k's)
  - Remove nodes with insufficient support

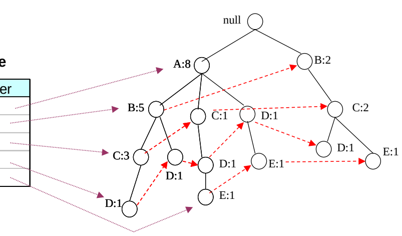


Example: Obtain T(E) from T

- Use header table to find the first path ending with E, find rest by following node links

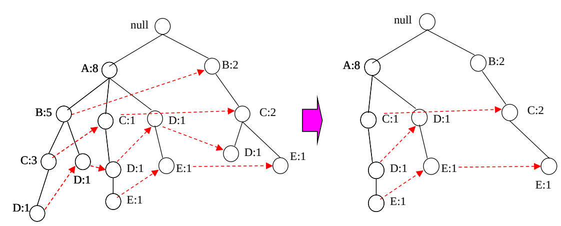
Header table

Item	Pointer
A	
B	
C	
D	
E	



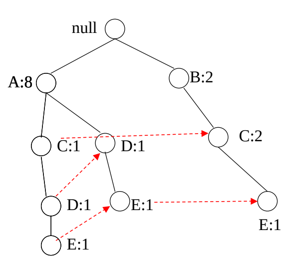
Retain Only Prefix Paths

- Obtain T' from T by retaining only paths ending with E
  - These paths are often called prefix paths



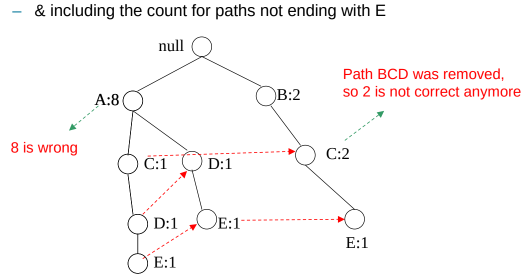
Determine if E is Frequent

- Support count of E = sum of counts of all E nodes
  - suppose minimum support count is 2
  - support count of E = 3, so E is frequent



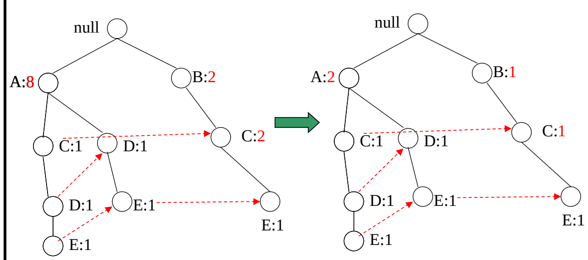
Update Counts

- Counts for internal nodes may not be correct
  - due to removal of paths which do not have E's
  - & including the count for paths not ending with E



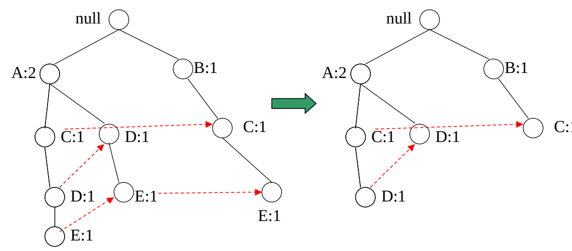
Update Counts

- Start from leaves, going upward
  - if node X has only one child Y, count of X = count of Y
  - otherwise, count of X = sum of counts of X's children



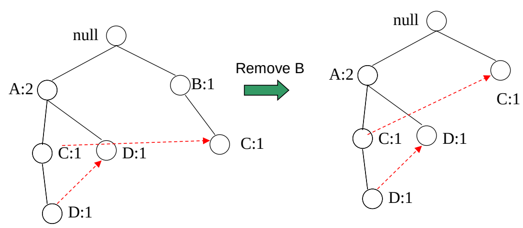
Remove E Nodes

- E nodes can now be removed
  - counts of internal nodes have been updated
  - not needed for solving: DE, CE, …



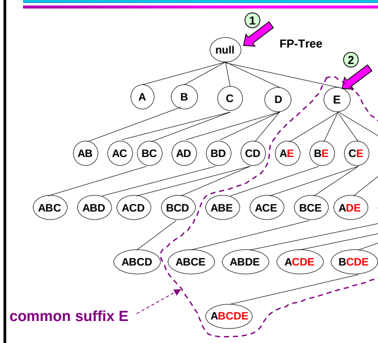
Remove Nodes with Insufficient Support

- If sum of counts of all X nodes < minimum support count, remove X nodes, since XE can not be frequent



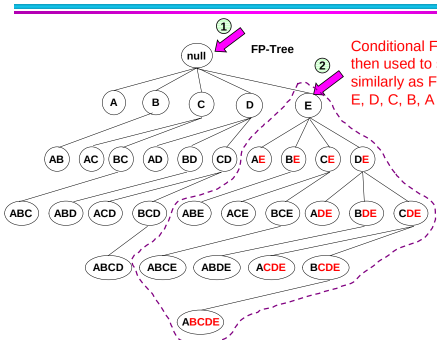
T(E): conditional FP-Tree for suffix E

Current Position in Processing



Sub-tree for suffix E turned into conditional FP-Tree for suffix E
- counts updated
- leaves (E's) removed
- infrequent itemsets pruned

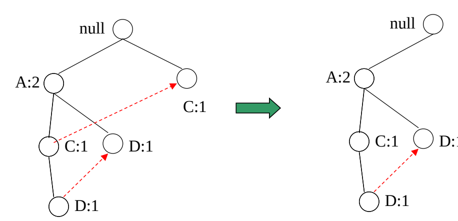
Current Position in Processing



Conditional FP-Tree for suffix E then used to solve DE, CE, BE, AE similarly as FP-Tree used to solve E, D, C, B, A

Obtain T(DE) from T(E)

- Retaining only prefix paths ending with D

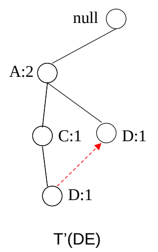


T(E): conditional FP-Tree for suffix E

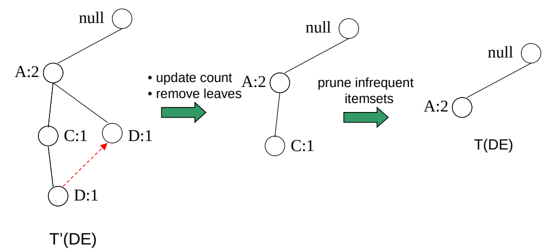
T'(DE)

Determine if DE is Frequent

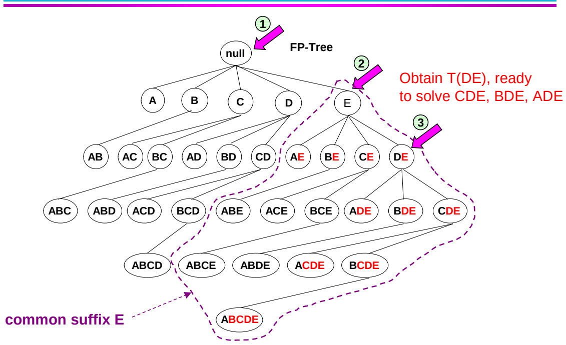
- Support count of DE = 2 (sum of counts of all D's)
  - DE is frequent, need to solve: CDE, BDE, ADE if they exist



Preparing for Solving CDE, BDE, ADE

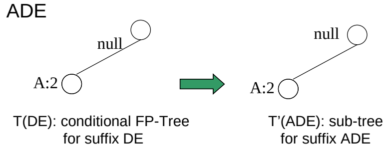


Current Position of Processing



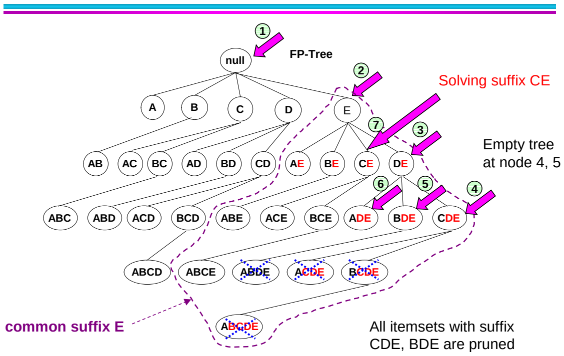
Solving CDE, BDE, ADE

- Sub-trees for both CDE and BDE are empty
  - no prefix paths ending with C or B
- Working on ADE

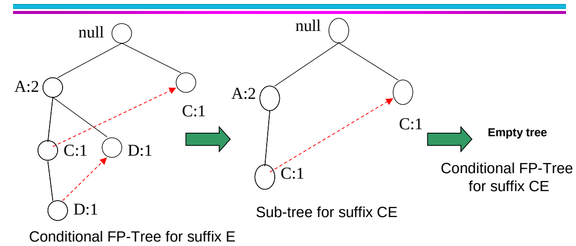


- ADE (support count = 2) is frequent
  - but no more subproblem for ADE, backtrack
  - & no more subproblem for DE, backtrack
  - → solving next subproblem CE

Current Position in Processing

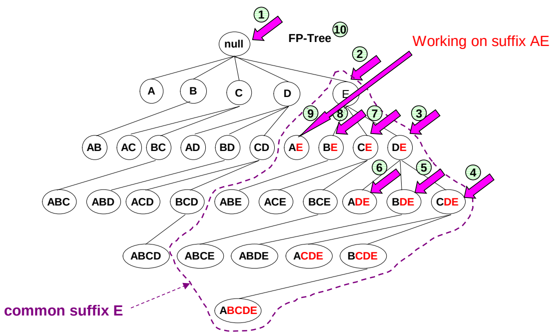


Solving for Suffix CE

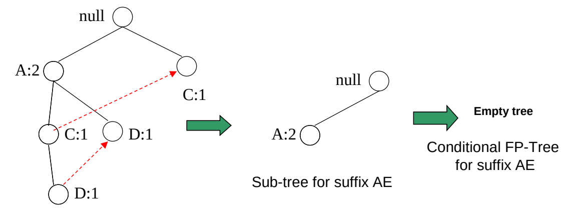


- CE is frequent (support count = 2)
- No more subproblems for CE (empty conditional FP-Tree, why?), so done with CE
- Work on next subproblems: BE (no support), AE

Current Position in Processing



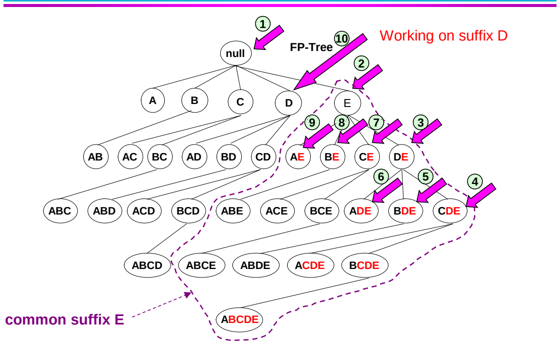
Solving for Suffix AE



T(E): conditional FP-Tree for suffix E

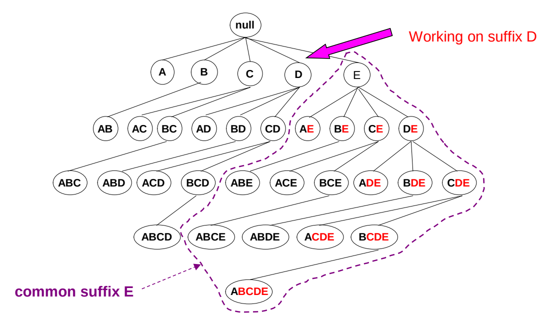
- AE is frequent (support count = 2)
- Done with AE, backtrack; done with E, backtrack
- Work on next subproblem: suffix D

Current Position in Processing



Found Frequent Itemsets with Suffix E

- E, DE, ADE, CE, AE discovered in this order



Outline

- Association rules & mining
- Finding frequent itemsets
  - Apriori algorithm
  - Compact representations of frequent itemsets
  - Alternative discovery methods
  - FP-Growth algorithm
- Generating association rules ⬅
- Evaluating discovered rules

Rule Generation

- Given a frequent itemset X, find all non-empty subsets $Y \subset X$ such that $Y \rightarrow X - Y$ satisfies the minimum confidence requirement
  - If X = {A,B,C,D}, we have these candidate rules:

ABC →D,	ABD →C,	ACD →B,	BCD →A,
A →BCD,	B →ACD,	C →ABD,	D →ABC
AB →CD,	AC → BD,	AD → BC,	BC →AD,
BD →AC,	CD →AB,		

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

Rule Generation

- How to efficiently generate rules from frequent itemsets?
- Recall that support is anti-monotonic:
  - Support of an itemset never exceeds support of its subset
  - E.g., s(ABCD) ≤ s(ABD)
- But in general, confidence is not anti-monotonic
  - e.g., c(ABC →D) can be larger or smaller than c(AB →D). Why?

$$c(ABC \rightarrow D) = \frac{\sigma(ABCD)}{\sigma(ABC)} \qquad c(AB \rightarrow D) = \frac{\sigma(ABD)}{\sigma(AB)}$$

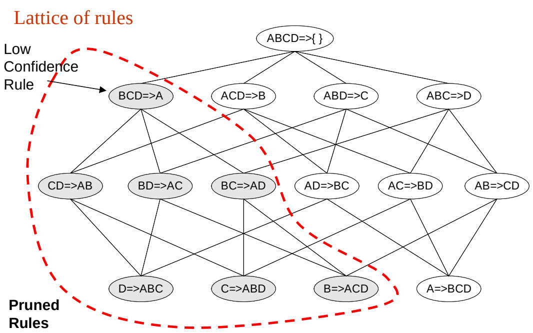
Rule Generation

- But confidence of rules generated from the same itemset has an anti-monotone property
  - E.g., X = {A,B,C,D}, c(BCD → A) ≥ c(CD → AB) ≥ c(D → ABC)

$$\frac{\sigma(ABCD)}{\sigma(BCD)} \geq \frac{\sigma(ABCD)}{\sigma(CD)} \geq \frac{\sigma(ABCD)}{\sigma(D)}$$

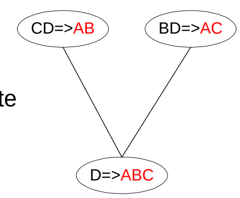
- Confidence is anti-monotonic w.r.t. the number of items on the right-hand side of the rule
  - More items on the right → lower/equal confidence
- Or monotonic w.r.t. the number of items on left-hand side
  - More items on the left → larger/equal confidence

Rule Generation for Apriori Algorithm



Rule Generation for Apriori Algorithm

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



Outline

- Association rules & mining
- Finding frequent itemsets
  - Apriori algorithm
  - Compact representations of frequent itemsets
  - Alternative discovery methods
  - FP-Growth algorithm
- Generating association rules
- Evaluating discovered rules ⬅

Pattern Evaluation

- Algorithm might produce a large number of rules
  - Many of which are not "interesting"
- Original formulation of association rules
  - Based on support & confidence threshold
  - But sometimes high confidence rules are interesting
- Additional interestingness measures may be used to further prune/rank discovered rules
  - Lift
  - Cosine
  - Jaccard coefficient

Computing Interestingness Measure

- Given a rule X → Y, information needed to compute rule interestingness can be obtained from a contingency table

Contingency table for X → Y

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	|T|

f_{11}: support count of X and Y
f_{10}: support count of X and $\overline{Y}$
f_{01}: support count of $\overline{X}$ and Y
f_{00}: support count of $\overline{X}$ and $\overline{Y}$

Transaction does not contain X

Used to define various measures
- support, confidence, lift, cosine, Jaccard coefficient, etc

Computing Interestingness Measure

- Given a rule X → Y, information needed to compute rule interestingness can be obtained from a contingency table

Contingency table for X → Y

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	|T|

confidence (X → Y)

$= \frac{\sigma(X \cup Y)}{\sigma(X)} = \frac{P(X,Y)}{P(X)} = P(Y \mid X)$

$= f_{11} / (f_{11} + f_{10})$

$= f_{11} / f_{1+}$

Drawback of Confidence

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea → Coffee

Confidence= P(Coffee|Tea) = 0.75, P(Coffee) = 0.9 > .75

➔ Drinking tea actually decreases probability of drinking coffee

➔ Although confidence is high, rule is misleading

Computing Interestingness Measure

- Given a rule X → Y, information needed to compute rule interestingness can be obtained from a contingency table

Contingency table for X → Y

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	|T|

confidence (X → Y)

$= \frac{\sigma(X \cup Y)}{\sigma(X)} = \frac{P(X,Y)}{P(X)} = P(Y \mid X)$

based on P(X,Y) & P(X), but also need to consider P(Y)

➔ need to measure correlation of X & Y

Measuring Correlation

- Population of 1000 students
  - 600 students know how to swim (S)
  - 700 students know how to bike (B)
  - 420 students know how to swim and bike (S,B)

  - $P(S \wedge B) = 420/1000 = 0.42$
  - $P(S) \times P(B) = 0.6 \times 0.7 = 0.42$

  - $P(S \wedge B) = P(S) \times P(B)$ => Independent
  - $P(S \wedge B) > P(S) \times P(B)$ => Positively correlated
  - $P(S \wedge B) < P(S) \times P(B)$ => Negatively correlated

Measuring Correlation

- Population of 1000 students
  - 600 students know how to swim (S)
  - 700 students know how to bike (B)
  - 420 students know how to swim and bike (S,B)

$$Lift(S,B) = \frac{P(B \mid S)}{P(B)} = \frac{P(S,B)}{P(S)P(B)}$$

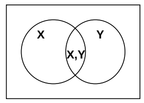
  - $P(S \wedge B) = P(S) \times P(B)$ ➔ lift = 1 ➔ S & B independent
  - $P(S \wedge B) > P(S) \times P(B)$ ➔ lift > 1 ➔ positive-correlated
  - $P(S \wedge B) < P(S) \times P(B)$ ➔ lift < 1 ➔ negative-correlated

Measures on Correlation

- The larger the value, usually the more likely the two variables/events are correlated

$$Lift(X,Y)=\frac{P(X,Y)}{P(X)P(Y)}$$

$$Cosine(X,Y)=\frac{P(X,Y)}{\sqrt{P(X)P(Y)}}$$



$$Jaccard(X,Y)=\frac{P(X,Y)}{P(X)+P(Y)-P(X,Y)}$$

... many others

Computing Lift from Contingency Table

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

$$Lift(X,Y)$$

$$=\frac{P(X,Y)}{P(X)P(Y)}=\frac{\frac{f_{11}}{N}}{\frac{f_{1+}}{N}\frac{f_{+1}}{N}}=\frac{Nf_{11}}{f_{1+}f_{+1}}$$

Lift Example

	Coffee	$\overline{\text{Coffee}}$	
Tea	15	5	20
$\overline{\text{Tea}}$	75	5	80
	90	10	100

Association Rule: Tea → Coffee

Confidence= P(Coffee|Tea) = 0.75

but P(Coffee) = 0.9

⇒ Lift = 0.75/0.9= 0.8333 (< 1, therefore negatively associated)

Drawback of Lift

	Y	$\overline{Y}$	
X	10	0	10
$\overline{X}$	0	90	90
	10	90	100

	Y	$\overline{Y}$	
X	90	0	90
$\overline{X}$	0	10	10
	90	10	100

$$Lift=\frac{0.1}{(0.1)(0.1)}=10 \quad >> \quad Lift=\frac{0.9}{(0.9)(0.9)}=1.11$$

X & Y seldom co-occur — X & Y frequently co-occur

Confidence might be better in this case (both have confidence 1)

Compute Cosine from Contingency Table

- Image X, Y are binary vectors
  - Its i-th position = 1, if X occur in i-th transaction

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

$$Cosine(X,Y)$$

$$=\frac{f_{11}}{\sqrt{f_{1+}}\sqrt{f_{+1}}}\left(=\frac{\frac{f_{11}}{N}}{\sqrt{\frac{f_{1+}}{N}\frac{f_{+1}}{N}}}=\frac{P(X,Y)}{\sqrt{P(X)P(Y)}}\right)$$

Cosine vs. Lift

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

$$Cosine(X,Y)=\frac{P(X,Y)}{\sqrt{P(X)P(Y)}}=\frac{f_{11}}{\sqrt{f_{1+}f_{+1}}}$$

$$Lift(X,Y)=\frac{P(X,Y)}{P(X)P(Y)}=\frac{Nf_{11}}{f_{1+}f_{+1}}$$

- If X and Y are independent, lift(X,Y) = 1, but Cosine(X,Y) = $\sqrt{P(X)P(Y)}$
- Cosine does not depend on N (& f_{00}), unlike Lift

Computing Jaccard Coefficient

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

$$Jaccard(X,Y)$$
$$= \frac{P(X,Y)}{P(X)+P(Y)-P(X,Y)} = \frac{f_{11}}{f_{11}+f_{10}+f_{01}}$$

Property under Null Addition

- Add more transactions that do not have X or Y

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00}	f_{o+}
	f_{+1}	f_{+0}	N

⇒

	Y	$\overline{Y}$	
X	f_{11}	f_{10}	f_{1+}
$\overline{X}$	f_{01}	f_{00} + s	f_{o+} + s
	f_{+1}	f_{+0} + s	N+s

Invariant measures:

- confidence, Cosine, Jaccard, etc

Non-invariant measures:

- support, lift, etc

Property under Variable Permutation

	B	$\overline{B}$
A	p	q
$\overline{A}$	r	s

⇒

	A	$\overline{A}$
B	p	r
$\overline{B}$	q	s

Does M(A,B) = M(B,A)?

If yes, M is symmetric; otherwise asymmetric

For example, c(A→B) = c(B→A)?

$$c(A \rightarrow B) = \frac{\sigma(A \cup B)}{\sigma(A)} \qquad c(B \rightarrow A) = \frac{\sigma(B \cup A)}{\sigma(B)}$$

Property under Variable Permutation

	B	$\overline{B}$
A	p	q
$\overline{A}$	r	s

⇒

	A	$\overline{A}$
B	p	r
$\overline{B}$	q	s

Does M(A,B) = M(B,A)?

If yes, M is symmetric; otherwise asymmetric

lift(A→B) = lift(B→A)?

$$lift(A \rightarrow B) = \frac{P(A,B)}{P(A)P(B)} \qquad lift(B \rightarrow A) = \frac{P(B,A)}{P(B)P(A)}$$

Property under Variable Permutation

	B	$\overline{B}$
A	p	q
$\overline{A}$	r	s

⇒

	A	$\overline{A}$
B	p	r
$\overline{B}$	q	s

Does M(A,B) = M(B,A)?

If yes, M is symmetric; otherwise asymmetric

Symmetric measures:

- support, lift, Cosine, Jaccard, etc

Asymmetric measures:

- confidence, etc

Applying Interestingness Measures

	B	$\overline{B}$
A	p	q
$\overline{A}$	r	s

⇒

	A	$\overline{A}$
B	p	r
$\overline{B}$	q	s

Association rule (A→B) directional

- confidence (asymmetric) is intuitive
- but confidence does not capture correlation

→ Use additional measures such as lift to rank discovered rules & further prune those with low ranks

References

- Introduction to Data Mining by Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. Addison-Wesley, 2006.
  - Chapter 6