Why Constraint-Based Mining?

Why Constraint-Based Mining?

- Finding all the patterns in a dataset autonomously – unrealistic!
  - Too many patterns but not necessarily user-interested!
- Pattern mining should be an interactive process
  - User directs what to be mined using a data mining query language (or a graphical user interface)
- Constraint-based mining
  - User flexibility: Provides constraints on what to be mined
  - Optimization: Explores such constraints for efficient mining
    - Constraint-based mining: Constraint-pushing, similar to push selection first in DB query processing

Constraints in General Data Mining

A data mining query can be in the form of a meta-rule or with the following language primitives

- Knowledge type constraint
  - Ex.: Classification, association, clustering, outlier finding, …
- Data constraint – using SQL-like queries
  - Ex.: Find products sold together in NY stores this year
- Dimension/level constraint
  - Ex.: In relevance to region, price, brand, customer category
- Rule (or pattern) constraint
  - Ex.: Small sales (price < \$10) triggers big sales (sum > \$200)
- Interestingness constraint
  - Ex.: Strong rules: min_sup $\geq$ 0.02, min_conf $\geq$ 0.6, min_correlation $\geq$ 0.7

Meta-Rule Guided Mining

- A meta-rule can contain partially instantiated predicates & constants
  - $P_1(X, Y) \wedge P_2(X, W) \Rightarrow buys(X, \text{"iPad"})$
- The resulting mined rule can be
  - age(X, "15-25") ^ profession(X, "student") ⇒ buys(X, "iPad")
- In general, (meta) rules can be in the form of
  - $P_1 \wedge P_2 \wedge \ldots \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_r$
- Method to find meta-rules
  - Find frequent (l + r) predicates (based on *min-support*)
  - Push constants deeply when possible into the mining process
    - Using constraint-push techniques introduced in this lecture
  - Also, push min_conf, min_correlation, and other measures as early as possible (measures acting as constraints)

Different Kinds of Constraints: Different Pruning Strategies

Different Kinds of Constraints Lead to Different Pruning Strategies

- Constraints can be categorized as
  - Pattern space pruning constraints vs. data space pruning constraints
- Pattern space pruning constraints
  - Anti-monotonic: If constraint *c* is violated, its further mining can be terminated
  - Monotonic: If *c* is satisfied, no need to check *c* again
  - Succinct: If the constraint *c* can be enforced by directly manipulating the data
  - Convertible: *c* can be converted to monotonic or anti-monotonic if items can be properly ordered in processing
- Data space pruning constraints
  - Data succinct: Data space can be pruned at the initial pattern mining process
  - Data anti-monotonic: If a transaction *t* does not satisfy *c*, then *t* can be pruned to reduce data processing effort

Constrained Mining with Pattern Anti-Monotonicity

Pattern Space Pruning with Pattern Anti-Monotonicity

- Constraint c is *anti-monotone*
  - If an itemset S **violates** constraint c, so does any of its superset
  - That is, mining on itemset S can be terminated
- Ex. 1: c_1: $sum(S.price) \leq v$ is anti-monotone
- Ex. 2: c_2: range(S.profit) $\leq$ 15 is anti-monotone
  - Itemset *ab* violates c_2 (range(ab) = 40)
  - So does every superset of *ab*
- Ex. 3. c_3: $sum(S.Price) \geq v$ is not anti-monotone
- Ex. 4. Is c_4: $support(S) \geq \sigma$ anti-monotone?
  - Yes! Apriori pruning is essentially pruning with an anti-monotonic constraint!

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

price(item)>0

Item	Profit
a	40
b	0
c	−20
d	−15
e	−30
f	−10
g	20
h	5

Constrained Mining with Pattern Monotonicity

Pattern Monotonicity and Its Roles

- A constraint c is *monotone*: If an itemset S **satisfies** the constraint c, so does any of its superset
  - That is, we do not need to check c in subsequent mining
- Ex. 1: c_1: $sum(S.Price) \geq v$ is monotone
- Ex. 2: c_2: $min(S.Price) \leq v$ is monotone
- Ex. 3: c_3: range(S.profit) $\geq$ 15 is monotone
  - Itemset *ab* satisfies c_3
  - So does every superset of *ab*

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

price(item)>0

Item	Profit
a	40
b	0
c	−20
d	−15
e	−30
f	−10
g	20
h	5

Constrained Mining with Data Anti-Monotonicity

Data Space Pruning with Data Anti-Monotonicity

- A constraint c is *data anti-monotone*: In the mining process, if a data entry *t* cannot satisfy a pattern *p* under *c*, *t* cannot satisfy *p*'s superset either
  - Data space pruning: Data entry *t* can be pruned
- Ex. 1: c_1: *sum*(*S.Profit*) $\geq v$ is data anti-monotone
  - Let constraint c_1 be: *sum*(*S.Profit*) ≥ 25
    - T_{30}: {b, c, d, f, g} can be removed since none of their combinations can make an S whose sum of the profit is ≥ 25
- Ex. 2: c_2: *min*(*S.Price*) $\leq v$ is data anti-monotone
    - Consider $v = 5$ but every item in a transaction, say T_{50}, has a price higher than 10
- Ex. 3: c_3: *range*(*S.Profit*) > *25* is data anti-monotone

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

min_sup = 2

price(item) > 0

Item	Profit
a	40
b	0
c	−20
d	−15
e	−30
f	−10
g	20
h	5

Data Space Pruning Should Be Explored Recursively

- Example. c_3: *range*(*S.Profit*) > *25*
  - We check b's projected database
    - But item "a" is infrequent (sup = 1)
  - After removing "a (40)" from T_{10}
    - T_{10} cannot satisfy c_3 any more
      - Since "b (0)" and "c (−20), d (−15), f (−10), h (5)"
    - By removing T_{10}, we can also prune "h" in T_{20}

b's-proj. DB

TID	Transaction
10	a, c, d, f, h
20	c, d, f, g, h
30	c, d, f, g

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

Item	Profit
a	40
b	0
c	−20
d	−15
e	−30
f	−10
g	20
h	5

min_sup = 2

price(item) > 0

Constraint: range{S.profit} > 25

b's-proj. DB

TID	Transaction
~~10~~	~~a, c, d, f, h~~
20	c, d, f, g, ~~h~~
30	c, d, f, g

Recursive Data Pruning

b's FP-tree

single branch: cdfg: 2

Only a single branch "cdfg: 2" to be mined in b's projected DB

- Note: c_3 prunes T_{10} effectively only after "a" is pruned (by min-sup) in b's projected DB

Constrained Mining with Succinct Constraints

Succinctness: Pruning Both Data and Pattern Spaces

- Succinctness: If the constraint *c* can be enforced by directly manipulating the data
- Ex. 1: To find those patterns without item *i*
  - Remove *i* from DB and then mine (pattern space pruning)
- Ex. 2: To find those patterns containing item *i*
  - Mine only *i*-projected DB (data space pruning)
- Ex. 3: c_3: $min(S.Price) \le v$ is succinct
  - Start with only items whose price $\le$ v and remove transactions with high-price items only (pattern + data space pruning)
- Ex. 4: c_4: $sum(S.Price) \ge v$ is not succinct
  - It cannot be determined beforehand since sum of the price of itemset S keeps increasing

Constrained Mining with Convertible Constraints

Convertible Constraints: Ordering Data in Transactions

- Convert tough constraints into (anti-)monotone by proper ordering of items in transactions
- Examine c_1: avg(*S*.profit) > 20
  - Order items in value-descending order
    - <*a, g, f, b, h, d, c, e*>
  - An itemset *ab* violates c_1 (avg(ab) = 20)
    - So does *ab** (i.e., *ab-projected DB)*
    - C_1: anti-monotone if patterns grow in the right order!
- Can item-reordering work for Apriori?
  - Does not work for level-wise candidate generation!
  - *avg*(*agf*) = 23.3 > 20, but avg(*gf*) = 15 < 20

min_sup = 2

price(item)>0

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	a, c, e, f, g

Item	Profit
a	40
b	0
c	−20
d	−15
e	−30
f	10
g	20
h	−5

Handling Multiple Constraints

How to Handle Multiple Constraints?

- It is beneficial to use multiple constraints in pattern mining
- But different constraints may require potentially conflicting item-ordering
  - If there exists an order *R* making both c_1 and c_2 convertible, try to sort items in the order that benefits pruning most
  - If there exists conflict ordering between c_1 and c_2
    - Try to sort data and enforce *one constraint* first (which one?)
    - Then enforce the other when mining the projected databases
- Ex. c_1: avg(*S*.profit) > 20, and c_2: avg(S.price) < 50
  - Sort in profit descending order and use c_1 first (assuming c_1 has more pruning power)
  - For each project DB, sort trans. in price ascending order and use c_2 at mining

Constraint-Based Sequential-Pattern Mining

Constraint-Based Sequential-Pattern Mining

- Share many similarities with constraint-based itemset mining
- Anti-monotonic: If S violates *c*, the super-sequences of S also violate *c*
  - sum(S.price) < 150; min(S.value) > 10
- Monotonic: If S satisfies *c*, the super-sequences of S also do so
  - element_count (S) > 5; S $\supseteq$ {PC, digital_camera}
- Data anti-monotonic: If a sequence s_1 with respect to S violates c_3, s_1 can be removed
  - c_3: sum(S.price) $\geq$ v
- Succinct: Enforce constraint c by explicitly manipulating data
  - S $\supseteq$ {i-phone, MacAir}
- Convertible: Projection based on the sorted value not sequence order
  - value_avg(S) < 25; profit_sum (S) > 160
  - max(S)/avg(S) < 2; median(S) – min(S) > 5

Timing-Based Constraints in Seq.-Pattern Mining

- Order constraint: Some items must happen before the other
  - {algebra, geometry} → {calculus} (where "→" indicates ordering)
  - Anti-monotonic: Constraint-violating sub-patterns pruned
- Min-gap/max-gap constraint: Confines two elements in a pattern
  - E.g., mingap = 1, maxgap = 4
  - Succinct: Enforced directly during pattern growth
- Max-span constraint: Maximum allowed time difference between the 1st and the last elements in the pattern
  - E.g., maxspan (S) = 60 (days)
  - Succinct: Enforced directly when the 1st element is determined
- Window size constraint: Events in an element do not have to occur at the same time: Enforce max allowed time difference
  - E.g., window-size = 2: Various ways to merge events into elements

Episodes and Episode Pattern Mining

- Episodes and regular expressions: Alternative to seq. patterns
  - Serial episodes: A $\rightarrow$ B
  - Parallel episodes: A | B

    Indicating partial order relationships
  - Regular expressions: (A|B)C*(D $\rightarrow$ E)
- Methods for episode pattern mining
  - Variations of Apriori/GSP-like algorithms
  - Projection-based pattern growth
    - Q_1: Can you work out the details?
  - Q_2: What are the differences between mining episodes and constraint-based pattern mining?