TYPES OF DATA SETS
[Record data; Graphs and networks; Ordered data {sequential}; Spatial, image and multimedia data]

STRUCTURED DATA CHARACTERISTICS
[Dimensionality; Resolution (分解); Sparsity; Distribution (e.g. centrality)]

Attributes -> Data Objects -> Data Sets
[Nominal; Binary; Ordinal; Numeric]

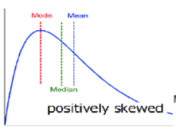
$$s^2=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2=\frac{1}{n-1}[\sum_{i=1}^{n}x_i^2-\frac{1}{n}(\sum_{i=1}^{n}x_i)^2]$$

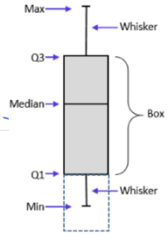
n - 1

$$\sigma^2=\frac{1}{N}\sum_{i=1}^{n}(x_i-\mu)^2=\frac{1}{N}\sum_{i=1}^{n}x_i^2-\mu^2$$

N

*mean – mode = 3 * (mean - median)*
±1 σ =68%;
±2 σ =95%;
±3 σ =97%.





GRAPHICS
Boxplot [Q_1 = 25[th] percentile; *Inter Quartile Range* = Q_3-Q_1; Outlier > 1.5 * IQR]; ***Histogram***; ***Quantile plot***; ***Quantile-Quantile (q-q) plot***; ***Scatter plot***.

VISUALIZATION
1. Pixel-oriented; ***2. Geometric projection*** *{direct; scatterplot; landscape; parallel}*; ***3. Icon-based*** *{Chernoff Faces; stick figures; shape coding; color icons; tile bars}*; ***4. Hierarchical*** *{dimensional stacking; Worlds-within-worlds; TreeMap; 3D Cone trees, InfoCube}*; ***5. Complex*** *{Tag Cloud; social network}*.

PROXIMITY (SIMILARITY) (dissimilarity = distance)
1. Nominal [m: matches, p: total variables]

$$d(i,j)=\frac{p-m}{p}$$

2. Binary *{Jaccard / coherence}*

	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

$$d(i,j)=\frac{r+s}{q+r+s+t}$$

$$d(i,j)=\frac{r+s}{q+r+s}$$

$$sim(i,j)=\frac{q}{q+r+s}$$

3. Numeric Standardizing:
Z-score or *Mean Absolute Deviation*

$$z=\frac{x-\mu}{\sigma}\qquad m_f=\frac{1}{n}(x_{1f}+x_{2f}+\ldots+x_{nf})\qquad z_{if}=\frac{x_{if}-m_f}{s_f}$$

$$s_f=\frac{1}{n}(|x_{1f}-m_f|+|x_{2f}-m_f|+\ldots+|x_{nf}-m_f|)$$

Minkowski distance [h = {1: Manhattan; 2: Euclidean; inf: supremum = max$|x_{if}-x_{jf}|$}]

$$d(i,j)=\sqrt[h]{|x_{i1}-x_{j1}|^h+|x_{i2}-x_{j2}|^h+\cdots+|x_{ip}-x_{jp}|^h}$$

Cosine

$$sim(A,B)=\cos(\theta)=\frac{A\cdot B}{\|A\|\|B\|}$$

Ordinal
[Map to [0, 1]]

4. Mixed Type:

$$d(i,j)=\frac{\sum_{f=1}^{p}\delta_{ij}^{(f)}d_{ij}^{(f)}}{\sum_{f=1}^{p}\delta_{ij}^{(f)}}$$

f is binary or nominal:
$d_{ij}^{(f)}=0$ if $x_{if}=x_{jf}$, or $d_{ij}^{(f)}=1$ otherwise
f is numeric: use the normalized distance
f is ordinal
- Compute ranks r_{if} and
- Treat z_{if} as interval-scaled

$$z_{if}=\frac{r_{if}-1}{M_f-1}$$

(Kullback-Leibler) KL Divergence:

$$D_{KL}(p(x)||q(x))=\sum_{x\in X}p(x)\ln\frac{p(x)}{q(x)}$$

$$=\int_{-\infty}^{\infty}p(x)\ln\frac{p(x)}{q(x)}dx$$

[上: Discrete; 下: Continuous]

{Attr 为0: 引入一个极小量 e=0.001}

MEASUREMENT FOR DATA QUALITY
[Accurate, Complete, Consistent, Timely, Believable, Interpretable]

DATA CLEANING
1. Incomplete/Missing {global constant; attr mean (global/same-class); most probable}
2. Noisy {Binning (smooth by bin means / median / boundaries); regression; clustering (rm outliers); semi-supervised}
3. Discrepancy detection {metadata; overload; uniqueness / consecutive / null rule}

$$T=\prod_{i=1}^{n}(L_i+1)$$

DATA INTEGRATION
1. Data integration; ***2. Schema integration (metadata)***;
3. Entity identification; ***4. Data conflicts***;
5. Redundancy [Object identification; Derivative data]

$$\chi^2=\sum_{i}^{n}\frac{(O_i-E_i)^2}{E_i}$$

Correlation analysis
X^2(chi-square) test:

$$\hat{\sigma}^2=\frac{1}{n}\sum_{i=1}^{n}(x_i-\hat{\mu})^2$$

n

Covariance analysis
Single variable: ***sample***:

$$\sigma^2=\mathrm{var}(X)=E[(X-\mu)^2]=E[X^2]-\mu^2=E[X^2]-[E(x)]^2$$

Two variables: ***sample***:

$$\hat{\sigma}_{12}=\frac{1}{n}\sum_{i=1}^{n}(x_{i1}-\hat{\mu}_1)(x_{i2}-\hat{\mu}_2)$$

$$\sigma_{12}=E[(X_1-\mu_1)(X_2-\mu_2)]=E[X_1X_2]-\mu_1\mu_2=E[X_1X_2]-E[X_1]E[X_2]$$

彼此独立可以推出 $\sigma_{12}=0$，反之不成立！

$$\Sigma=E[(\mathbf{X}-\mu)(\mathbf{X}-\mu)^T]=\begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{21} & \sigma_2^2\end{pmatrix}$$

Two variable correlation $\rho_{12}=\frac{\sigma_{12}}{\sigma_1\sigma_2}=\frac{\sigma_{12}}{\sqrt{\sigma_1^2\sigma_2^2}}$
****sample:*** (上下的 n 抵消了)
ρ_{12} = 0: 彼此独立;
>0: 正相关; <0: 负相关

Covariance Matrix

$$\hat{\rho}_{12}=\frac{\hat{\sigma}_{12}}{\hat{\sigma}_1\hat{\sigma}_2}=\frac{\sum_{i=1}^{n}(x_{i1}-\hat{\mu}_1)(x_{i2}-\hat{\mu}_2)}{\sqrt{\sum_{i=1}^{n}(x_{i1}-\hat{\mu}_1)^2\sum_{i=1}^{n}(x_{i2}-\hat{\mu}_2)^2}}$$

DATA REDUCTION
1. Regression & Log-linear models [最小二乘; Parametric: 假设符合模型, 估算参数, 只存储参数, 舍弃除 outlier 的数据; *Y: dependent / response / measurement, X: independent / explanatory / predictors*] *{Linear reg.; Nonlinear reg.; Multiple reg.; Log-linear reg.}*
2. Histograms *{Equal-width; Equal-freq/depth}*, ***Clustering***, ***Sampling*** *{Simple random ~; ~ w/o or w/ replacement; stratified ~}*
4. Data cube aggregation
5. Data compression *{String ~ (lossless); Audio/Video ~ (lossy)}*
e.g. *Wavelet transform* [O(N); length 必须二次方]

Resolution	Averages	Detail Coefficients
8	[2, 2, 0, 2, 3, 5, 4, 4]	
4	[2, 1, 4, 4]	[0, −1, −1, 0]
2	$[1\frac{1}{2}, 4]$	$[\frac{1}{2}, 0]$
1	$[2\frac{3}{4}]$	$[-1\frac{1}{4}]$

Given a database of T tuples, D dimensions, and F shell fragment size, the fragment cubes' space requirement is:

$$O\left(T\left\lceil\frac{D}{F}\right\rceil(2^F-1)\right)$$

DATA TRANSFORMATION
1. Smoothing [Rm noise from data];
2. Attr / feature construction (new from old);
3. Aggregation *{Summarization; Data cube}*;
4. Normalization { $v'=\frac{v-min_A}{max_A-min_A}(new\_max_A-new\_min_A)+new\_min_A$
Min-max ~;
Z-score ~;

$$v'=\frac{v-\mu_A}{\sigma_A}\qquad v'=\frac{v}{10^j}$$

Decimal scaling ~ [j: smallest integer such that Max(|v'|) < 1]};
5. Discretization *{Binning* {Equal-width (distance), Equal-depth (freq.); Smoothing: by bin mean/boundary}; *Histogram*; *Clustering*; *DT*; *Correlation* {Chi-merge (χ^2-based discretization): [Bottom-up merge] {Find best neighboring intervals (those having similar distributions of classes, i.e., low χ^2 values) to merge}}

Concept Hierarchy [Organizes concepts (i.e. attr. val.) hierarchically; usually associated with each dim. in a data warehouse] {Recursively reduce the data by collecting and replacing low level concepts (e.g. age numeric val.) by higher level concepts (e.g. youth, adult, or senior)}

DIMENSIONALITY REDUCTION
(Reduce the number of random variables under consideration, via obtaining a set of principal variables) [Avoid the curse of dim.; Eliminate irrelevant features, reduce noise; Reduce time & space required; Allow easier visualization]
1. Feature selection (find a subset);
Attribute Subset Selection {Redundant, Irrelevant};
Heuristic Search in Attribute Selection {Best single attribute under the attribute independence assumption (choose by significance tests); Best step-wise feature selection (best); Step-wise attribute elimination (worse); Best combined attribute selection and elimination}; Optimal branch and bound (Use attribute elimination and backtracking);}
2. Feature extraction (transform the space);
Principal Component Analysis [*covariance matrix 的特征向量]

DATA WAREHOUSE
[A *Subject-oriented*, *Integrated* {multiple, heterogeneous}, *Time-variant* (t > operational system), *Non-volatile*: [independent; static (initial loading, access of data)] collection of data in support of management's decision-making process]
Models: {Enterprise warehouse; Data mart; Virtual warehouse}
(Extraction Transform Loading)
Conceptual Model {*Star schema* (1-N); *Snow-flake schema* (1-N-Ms); *Fact constellations* (1-N-1s)}
Design Process {Top-down / bottom-up / combination; Software Engineering (waterfall; spiral)} 模型
Usage {Info processing; Analytical processing; DM}

OLTP (OnLine Transactional Processing) [smaller DB size; smaller #records accessed; more users] *VS* **OLAP (OnLine Analytical Processing)** {extraction, cleaning, transformation, load}

OLAP Server Architecture {***Relational OLAP*** [Greater scalability]; ***Multidimensional OLAP*** [Sparse array-based multi-dim. storage engine; Fast indexing to pre-computed summarized data]; ***Hybrid OLAP*** [Flexibility (low-level: relational, high-level: array)] {e.g. SQLServer}; ***Specialized SQL servers*** (e.g., Redbricks)}

Indexing OLAP Data {***Bitmap Index***} {Each value in the column has a bit vector: bit-op is fast; The length of the bit vector: # of records in the base table; The i-th bit is set if the i-th row of the base table has the value for the indexed column; not suitable for high cardinality domains}

Base table

Cust	Region	Type
C1	Asia	Retail
C2	Europe	Dealer
C3	Asia	Dealer
C4	America	Retail
C5	Europe	Dealer

Index on Region

RecID	Asia	Europe	America
1	1	0	0
2	0	1	0
3	1	0	0
4	0	0	1
5	0	1	0

Index on Type

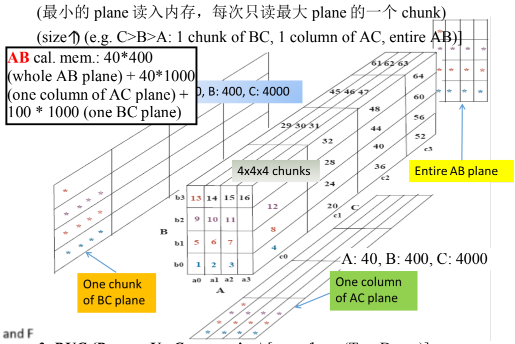
RecID	Retail	Dealer
1	1	0
2	0	1
3	0	1
4	1	0
5	0	1

DATA CUBE
(A lattice of cuboids (0-D: apex cuboid (parent); n-D: base cuboid))
Measure {*Distributive* (若应用到 aggregate value 与应用到全部数据的结果相同 *(可分布式计算并汇总)*) {count / sum, min / max}; *Algebraic* (用有限个 Distributive 几何运算得到); *Holistic* (描述子集所需内存没有常数上限) {median, mode, rank)}
OLAP Operation {Roll/Drill-up/down (summarize / detailize); Slice(去掉整个维度); Dice (只取一部分); Pivot (旋转)}
E.g. "SELECT item, city, year, SUM (amount) FROM SALES CUBE BY item, city, year", Need compute the following Group-Bys: {(date, product, customer), (date, product),(date, customer), (product, customer), (date), (product), (customer), ()}
Close cube (if there exists no cell *d*, such that *d* is a descendant of *c*, and *d* has the same measure value as *c*)
Cube shell (The cuboids involving a small # of dimensions)

DATA CUBE COMPUTATION
1. General Heuristics {Smallest-child; cache-results; amortize-scans; share-sorts; share-partitions}
2. MOLAP (Multi-way Array Aggregation) [BottomUp] [*PRO*: 计算小维度 full cube 高效; 同时多维度 aggr.; 中间 aggr 值复用于 ancestor cuboids] [*CON*: 不能 Apriori pruning: no iceberg optimi-zation] {Parition arrays into chunks; Compute aggregates in "multiway" by visiting cube cells in the order which minimizes the # of times to visit each cell, and reduces memory access and storage cost

(最小的 plane 读入内存，每次只读最大 plane 的一个 chunk)
(size↑ (e.g. C>B>A: 1 chunk of BC, 1 column of AC, entire AB))

AB cal. mem.: 40*400 (whole AB plane) + 40*1000 (one column of AC plane) + 100 * 1000 (one BC plane)



3. BUC (Bottom-Up Computation) [apex→base (Top-Down)]
[*PRO*: 适合 large-dim.(与 16Fall-Mid-Sol 矛盾?); Iceberg pruning {If a partition < minsup, its descendants pruned}]
4. Semi-Online Computational Model
[Tradeoff: amount of pre-computation VS speed of online computation]
[*PRO*: Offline + online OLAP; High-dim.; Lossless reduction]
{将 dimension 分成 ***shell fragment*** (不必 disjoint); Compute data cubes for each shell fragment while retaining *inverted indices* or *value-list indices*; Given the pre-computed fragment cubes, dynamically compute cube cells of the high-dimensional data cube online;}

tid	A	B	C	D	E
1	a1	b1	c1	d1	e1
2	a1	b2	c1	d2	e1
3	a1	b2	c1	d1	e2
4	a2	b1	c1	d1	e2
5	a2	b1	c1	d1	e3

Attribute Value	TID List	List Size
a1	1 2 3	3
a2	4 5	2
b1	1 4 5	3
b2	2 3	2
c1	1 2 3 4 5	5
d1	1 3 4 5	4
d2	2	1
e1	1 2	2
e2	3 4	2
e3	5	1

Cell	Intersection	TID List	List Size
a1 b1	1 2 3 ∩ 1 4 5	1	1
a1 b2	1 2 3 ∩ 2 3	2 3	2
a2 b1	4 5 ∩ 1 4 5	4 5	2
a2 b2	4 5 ∩ 2 3	ϕ	0

Online Query Computation with Shell-Fragments
[Query form: <a1, a2, …, an: M>, each a_i has 3 possible values: {Instantiated value; Aggregate * function; Inquire ? Function}] {(e.g., <3, ?, ?, *, 1: count> 返回 a 2-D data cube)}

FREQUENT ITEMSETS / PATTERNS (support ≥ 大于等于！)
Association Rule (X →Y) [support = sup (X∪Y), confidence = sup(X∪Y) / sup(X)]; ***Closed Pattern*** [If X is *frequent*, AND there exists no *super-pattern* Y ⊃ X, *with the same support* as X; Lossless compression]; ***Max Pattern*** [if X is frequent AND there exists no frequent *super-pattern* Y ⊃ X; Lossy compression]

PATTERN MINING METHODS
1. Downward Closure / Apriori
{Reduce passes of transaction database scans: {***Partitioning*** [任何可能频繁@TDB 的 itemset 必然在至少一个 TDB's partition 里频繁] {Scan DB only twice; Consolidate global FP}; *Dynamic itemset counting*}; Shrink the number of candidates {***Direct Hashing and Pruning*** (减少候选项数量) [相应 hashing bucket count 低于 threshold 的 k-itemset 必不频繁]; *Pruning by support lower bounding*; *Sampling*}; Exploring special data structures {*Tree rojection*; *H-miner*; *Hypecube decomposition*}}

- Candidates: a, b, c, d, e
- Hash entries
- {ab, ad, ae}
- {bd, be, de}
- …
- Frequent 1-itemset: a, b, d, e
- ab is not a candidate 2-itemset if the sum of count of {ab, ad, ae} is below support threshold

Itemsets	Count
{ab, ad, ae}	35
{bd, be, de}	298
……	…
{yz, qs, wt}	58

Hash Table

****ECLAT (Equivalent CLAss Transformation)***
(DFS using set intersection) [Tid-List:包含一个 itemset 的 transac.-ids 列表] [t(X) = t(Y): X and Y 总是一起发生); t(X) ⊂ t(Y): 包含 X 的交易总是包含 Y] {(e.g., Vertical format: t(e) = {T_{10}, T_{20}, T_{30}}; t(a) = {T_{10}, T_{20}}; t(ae) = {T_{10}, T_{20}}; t(ac) = t(d), t(ac) ⊂ t(ce)}
**使用 diffset 加速* [*Only keep track of diff. of tids] {t(e) = {T_{10}, T_{20}, T_{30}}, t(ce) = {T_{10}, T_{30}} → Diffset(ce, e) = {T_{20}}}*

A transaction DB in Horizontal Data Format

Tid	Itemset
10	a, c, d, e
20	a, b, e
30	b, c, e

The transaction DB in Vertical Data Format

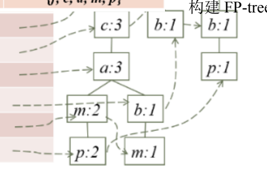
Item	TidList
a	10, 20
b	20, 30
c	10, 30
d	10
e	10, 20, 30

2. FPGrowth

TID	Items in the Transaction	Ordered, frequent items
100	{f, a, c, d, g, i, m, p}	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o, w}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}

{扫描 DB 一次, 找到 single item FP; 按 freq 降序排列 frequent item, 得到 f-list; 再扫 DB 一遍, 构建 FP-tree}

f	4
c	4
a	3
b	3
m	3
p	3



For each conditional pattern-base
Mine single-item patterns
Construct its FP-tree & mine it
p-conditional PB: fcam:2, cb:1 → c: 3
m-conditional PB: fca:2, fcab:1 → fca: 3
b-conditional PB: fca:1, f:1, c:1 → ϕ

8 = 1 *(m)* + 2^3 *(fca)*, i.e.

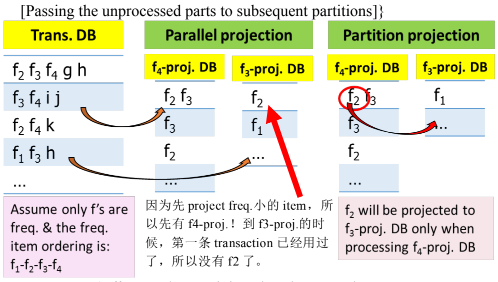
Conditional pattern bases

min_support = 3

Item	Conditional pattern base
c	f:3
a	fc:3
b	fca:1, f:1, c:1
m	fca:2, fcab:1
p	fcam:2, cb:1

m: 3; fm: 3, cm: 3, am: 3
fcm: 3, fam:3, cam: 3; fcam: 3

DB Projection (若 FPTree 放不下内存, scale FPGrowth)
{*Parallel projection* (project. DB on each freq. item) [Space costly, 所有 partitions 可以并行处理]; *Partition projection* (partition DB in order) [Passing the unprocessed parts to subsequent partitions]}



CLOSET+ (*Efficient*, *direct* mining closed patterns by Pattern-Growth) [*Itemset merging*: 若 X 出现的地方 Y 也都出现，那么 merge Y with X]
其他: {*Hybrid tree projection {Bottom-up physical ~; Top-down pseudo ~} Sub-itemset pruning; Item skipping; Efficient subset checking*}

PATTERN EVALUATION

[*Interestingness Measure*: {Objective (sup., conf., corr.); Subjective (Query-based; Knowledge-base; Visualization)]
1. Lift [=1: 独立; >1: 正相关; <1: 负相关]; $lift(B,C) = \frac{c(B \to C)}{s(C)} = \frac{s(B \cup C)}{s(B) \times s(C)}$
2. Chi-Square [=0: 独立; ≠0: 正或负相关]} $\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$

Null invariance (Value does not change with the # of null-transactions)

Measure	Definition	Range	Null-Invariant
$\chi^2(A,B)$	$\sum_{i,j=0,1} \frac{(e(a_i b_j) - o(a_i b_j))^2}{e(a_i b_j)}$	$[0, \infty]$	No
$Lift(A,B)$	$\frac{s(A \cup B)}{s(A) \times s(B)}$	$[0, \infty]$	No
$AllConf(A,B)$	$\frac{s(A \cup B)}{max\{s(A), s(B)\}}$	$[0, 1]$	Yes
$Jaccard(A,B)$	$\frac{s(A \cup B)}{s(A) + s(B) - s(A \cup B)}$	$[0, 1]$	Yes
$Cosine(A,B)$	$\frac{s(A \cup B)}{\sqrt{s(A) \times s(B)}}$	$[0, 1]$	Yes
$Kulczynski(A,B)$	$\frac{1}{2}\left(\frac{s(A \cup B)}{s(A)} + \frac{s(A \cup B)}{s(B)}\right)$	$[0, 1]$	Yes
$MaxConf(A, B)$	$max\{\frac{s(A)}{s(A \cup B)}, \frac{s(B)}{s(A \cup B)}\}$	$[0, 1]$	Yes

Data set	mc	$\neg mc$	$m \neg c$	$\neg m \neg c$	*AllConf*	*Jaccard*	*Cosine*	*Kulc*	*MaxConf*
D_1	10,000	1,000	1,000	100,000	0.91	0.83	0.91	0.91	0.91
D_2	10,000	1,000	1,000	100	0.91	0.83	0.91	0.91	0.91
D_3	100	1,000	1,000	100,000	0.09	0.05	0.09	0.09	0.09
D_4	1,000	1,000	1,000	100,000	0.5	0.33	0.5	0.5	0.5
D_5	1,000	100	10,000	100,000	0.09	0.09	0.29	0.5	0.91
D_6	1,000	10	100,000	100,000	0.01	0.01	0.10	0.5	0.99

3. IR (Imbalance Ratio): $IR(A,B) = \frac{|s(A) - s(B)|}{s(A) + s(B) - s(A \cup B)}$

	milk	$\neg$*milk*	Σ_{row}
coffee	mc	$\neg mc$	c
$\neg$*coffee*	$m \neg c$	$\neg m \neg c$	$\neg c$
Σ_{col}	m	$\neg m$	Σ

Data set	mc	$\neg mc$	$m \neg c$	$\neg m \neg c$	Jaccard	Cosine	Kulc	IR
D_1	10,000	1,000	1,000	100,000	0.83	0.91	0.91	0
D_2	10,000	1,000	1,000	100	0.83	0.91	0.91	0
D_3	100	1,000	1,000	100,000	0.05	0.09	0.09	0
D_4	1,000	1,000	1,000	100,000	0.33	0.5	0.5	0
D_5	1,000	100	10,000	100,000	0.09	0.29	0.5	0.89
D_6	1,000	10	100,000	100,000	0.01	0.10	0.5	0.99

MINING MULTI-LEVEL ASSOCIATIONS

1. Shared Multi-level Mining [用最低的 min-sup.来 pass down 候选项集]; ***2. Redundancy Filtering*** [*Redundant rule*: sup≈祖先的期望值 *AND* conf≈祖先;] ***3. Use group-based "individualized" minsup***

MINING MULTI-DIMENSIONAL ASSOCIATIONS

Multi-dimensional Rules (Items in ≥ 2 dimensions OR predicates)
[*Inter-dimension association rules* (no repeated predicates)
(e.g. age("18-25") ∧ job("student") ⇒ buys("coke"));
Hybrid-dimension association rules (repeated predicates)
(e.g. age("18-25") ∧ buys("popcorn") ⇒ buys("coke"))]

MINING QUANTITATIVE ASSOCIATIONS

(Mining associations with num. attrs.) ***1. Static discretization based on predefined concept hierarchies*** {Data cube-based aggregation};
2. Dynamic discretization based on data distribution; ***3. Clustering*** {First one-dimensional clustering, then association}; ***4. Deviation analysis*** (e.g. Gender = F ⇒ Wage: mean=$7/hr (overall mean = $9))}

*** Mining Extraordinary Phenomena in Quantitative Associations**
[Rule: accepted ONLY IF a stat. test confirms the inference with high conf.; Subrule: Highlights the extraordinary behavior of a subset of the population of the super rule]

MINING NEGATIVE CORRELATIONS

[Rare patterns (sup.很低但有趣); Negative Patterns (负相关: 很少一起发生); Negatively Correlated (A&B 频繁 AND sup(A∪B) << sup(A)×sup(B))] {*Kulczynski measure-based* [(P(A|B)+P(B|A))/2 < ϵ, ϵ: negative pattern threshold]};

MINING COMPRESSED PATTERNS

[*Pattern dist. measure*: $Dist(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$
δ-clustering [对每个 pattern P: 找到所有可以用 P 表达 AND 距离 P 在 δ 之内 (δ-cover)}

MINING REDUNDANCY-AWARE PATTERNS

(High significance AND low redundancy) {***Maximal Marginal Significance***: measure combined significance of a pattern set}

MINING CONSTRAINT-BASED FP

Constraints [*Knowledge type* ~ (classification; association; clustering; outlier); *Data* ~ (SQL-like query); *Dimension / level* ~ (region, price, brand, cat.); *Rule / Pattern* ~ (如 price < 10 ⇒ sum > 200); *Interestingness* ~]

META-RULE GUIDED MINING

[Meta rules: 总体上可以用 "$P_1 \wedge P_2 \wedge \ldots \wedge P_l \Rightarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_r$"的形式表示] {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)}

[设定：在 *C* 的限定条件下，从交易 *T* 里挖掘当前的 *FP / P*]

PATTERN SPACE PRUNING (Prune 掉整个 pattern)

[*Anti-monotonic* (c 被违反则可以停止深入; Itemset S 违反 C, 则 S 超集均违反); *Monotonic* (Itemset S 符合 C, S 超集也均符合);
Succinct (c can be enforced by directly manipulating the data);
Convertible (通过排列 transaction 里的 item 顺序，可转化成其他种类的条件, 如 avg()→降序)]

DATA SPACE PRUNING (Prune 掉整个 transaction)

[*Data succinct* (Can be pruned at the initial process); *Data anti-monotonic* (若一个 data entry 不能满足 pattern P, 它也不能满足 P 的所有超集, 因此可以被 prune)]
[应当被 explored recursively]

* **Succinctness** (Pruning both Data and Pattern Spaces)
{Ex. 1: To find patterns without item *i*: {Remove *i* from DB and then mine (pattern space pruning)}; Ex. 2: To find 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 ≤ 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};}

* **Multiple Constraints:**
Ex. c_1: avg(S.profit) > 20, and c_2: avg(S.price) < 50

Sorted 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

MINING LONG PATTERNS

[挑战: Curse of "downward closure" property of frequent patterns (FP's children 也是 FP, 于是 len 大的会衍生太多子孙)]

Pattern Fusion
[Fuse small patterns together in one step ("short-cuts") to generate new pattern candidates of significant sizes ("leaps")]
[*PRO*: Strive for mining almost complete and representative colossal patterns]
[*CON*: Not strive for completeness]
[*Core patterns of a colossal pattern α*: a set of subpatterns of α that cluster around α by sharing a similar support;
Core patterns: 给定 FP α, 其 subpattern β 是 **τ-core** pattern of α, 若 β shares a similar support set with α (见右边条件, τ: core ratio, $|D_\alpha|$: 数据库 D 里包含 α 的 pattern 数量)] $\frac{|D_\alpha|}{|D_\beta|} \ge \tau \quad 0 < \tau \le 1$

(d, τ)-robustness: 若最多可以去掉 *d* 个 item，且剩下的 pattern 是其 **τ-core**; 一个(d, τ)-robust pattern α 含有 $\Omega(2^d)$ core patterns; Colossal pattern 倾向于含有比 small patterns 多得多的 core patterns]

{在每次迭代中: 从当前 pattern pool 里随机选 K 个 seed patterns; 对于每个 seed pattern: 找到所有以其为中心的 bounding ball 内的 patterns; 将所有这些找到的 patterns 融合(fuse)到一起来生成 a set of super-patterns; 所有生成的 super-patterns 形成一个新的 pool 用于下次迭代; 在迭代开始时, 若当前 pool 包含小于等于 K patterns 则终止}

SEQUENTIAL PATTERN MINING

(Given a set of sequences, find the complete set of frequent subsequences)

A *sequence database*

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

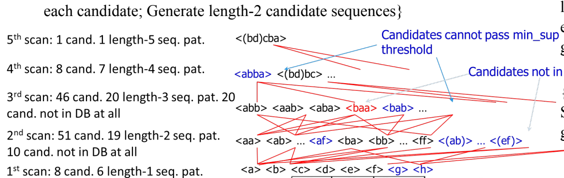
A *sequence*: < (ef) (ab) (df) c b >

- An *element* may contain a set of *items* (also called *events*)
- Items within an element are unordered and we list them alphabetically

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

- Given *support threshold* *min_sup* = 2, <(ab)c> is a *sequential pattern*

1. GSP (Generalized Sequential Patterns) [Apriori based] {Initial candidates: All singleton sequences; Scan DB once, count support for each candidate; Generate length-2 candidate sequences}



2. SPADE (Sequential PAttern Discovery using Equivalent Class) [Vertical format-based] {A sequence database is mapped to: <SeqID, EleID> (用以判断 candidate 是否存在及其顺序); Grow the subsequences (patterns) one item at a time by Apriori candidate generation}

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e

a		b		···
SID	EID	SID	EID	···
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			

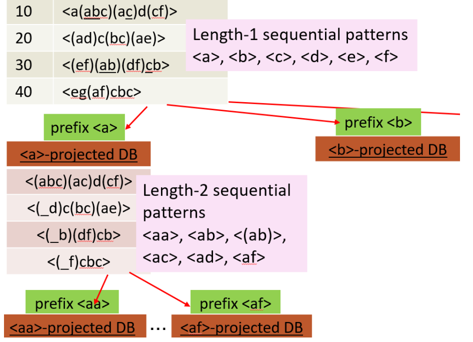
ab			ba		
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)
1	1	2	1	2	3
2	1	3	2	3	4
3	2	5			
4	3	5			

3. PrefixSpan (Prefix-projected Sequential PAttern Mining)
[Given <a(abc)(ac)d(cf)>: Prefixes: <a>, <aa>, <a(ab)>, <a(abc)>, …
Suffix: Prefixes-based projection] [*PRO*: No candidate subseqs. to be generated; Projected DBs keep shrinking] [*CON*: Major cost of constructing projected DBs]

Prefix	Suffix (Projection)
<a>	<(abc)(ac)d(cf)>
<aa>	<(_bc)(ac)d(cf)>
<ab>	<(_c)(ac)d(cf)>

{Find length-1 sequential patterns; Divide search space and mine each projected DB}

SID	Sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>



* ***Pseudo-Projection*** [If DB does not fit in memory]
* ***Physical-Projection*** [Used when DB can be held in main memory]
{No physically copying suffixes; Pointer to the sequence; Offset of the suffix}

4. CLOSPAN (Mining Closed Sequential Patterns)
[*PRO*: Efficiency!; Reduce # of (redundant) patterns; Attain the same expressive power]
[*closed sequential pattern*: There exists no super-pattern s' s.t. s' ⊃ s AND s' and s have same sup.; If $s \supset s_1$, s is closed iff. two project DBs have the same size] {Explore *Backward Subpattern* and *Backward Superpattern* pruning to prune redundant search space}

CONSTRAINT-BASED SEQUENTIAL-PATTERN MINING

[*Anti-monotonic* (If S violates c, the super-sequences of S also violate c); *Monotonic* (If S satisfies c, the super-sequences of S also do so); *Data anti-monotonic* (If a sequence s1 with respect to S violates c3, s1 can be removed); *Succinct* (Enforce constraint c by explicitly manipulating data); *Convertible* (Projection based on the sorted value not sequence order]

TIMING-BASED CONSTRAINTS IN SEQ.-PATTERN MINING

[*Order constraint* (Some items must happen before the other); *Anti-monotonic* (Constraint-violating sub-patterns pruned); *Min-gap/max-gap constraint* (Confines two elements in a pattern); *Succinct* (Enforced directly during pattern growth); *Max-span constraint* (Maximum allowed time difference between the 1st and the last elements in the pattern); *Window size constraint* (Events in an element don't have to occur at the same time: Enforce max allowed time difference)]

GRAPH PATTERN MINING

[Given a labeled graph dataset D = {G_1, G_2, …, G_n), the supporting graph set of a subgraph ***g*** is D_g = {G_i | ***g*** ⊆ G_i, G_i ∈D}] [support(g) = $|D_g|$/ |D|]
{Generation of candidate subgraphs: {Apriori vs. pattern growth (e.g., FSG vs. gSpan)}; Search order: {Breadth vs. depth}; Elimination of duplicate subgraphs: {Passive vs. active (e.g., gSpan (Yan&Han'02))}; Support calculation: {Store embeddings (e.g., GASTON, FFSM, MoFa)}; Order of pattern discovery {Path → tree → graph (e.g., GASTON}}

1. gSPAN (Graph Pattern Growth in Order) [*Completeness*: The enumeration of graphs using *right-most path extension* is complete; *DFS Code*: Flatten a graph into a sequence using depth-first search] {*Right-most path extension* in subgraph pattern growth {*Right-most path*: The path from root to the right-most leaf (choose the vertex w. the smallest index at each step); Reduce generation of duplicate subgraphs}}

2. CloseGraph (Mines closed graph patterns directly) [*PRO*: Lossless compression; Extension of gSpan] {Suppose G and G_1 are frequent, and G is a subgraph of G_1; If in any part of the graph in the dataset where G occurs, G_1 also occurs, then we need not grow G, since none of G's children will be closed except those of G_1;}

3. SpiderMine (Mining Top-K Large Structural Patterns in a Massive Network) [Large patterns are composed of a number of small components ("spiders") which will eventually connect together after some rounds of pattern growth] [*r-Spider*: An r-spider is a frequent graph pattern P such that there exists a vertex u of P, and all other vertices of P are within distance r from u]
[*PRO*: Good for mining large patterns: {Small patterns are much less likely to be hit in the random draw; Even if a small pattern is hit, it is even less likely to be hit multiple times; The larger the pattern, the greater the chance it is hit and saved}]

{Graph indexing: {***gIndex*** (Indexing Frequent and Discriminative Substructures)}; Support substructure similarity search {Keep the graph index structure, but select features in the query space}}