## Know your data(Chapter 2)

**1.Type:** **Nominal**：定类 **Binary**： **Ordinal:** 定序 *Numeric (**Interval-scaled**：定距，0 不代表没有 **Ratio-scaled**：定比，0 代表没有)

**2.Unimodal 单峰：** mean-mode=3*(mean-median)

**3.Symmetric:** mode=median=mean

**positively skewed:** < < **negatively skewed:** > >

**4. Variance:** **sample(n): population(N):**

$$s^2=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-\bar{x})^2=\frac{1}{n-1}\left[\sum_{i=1}^{n}x_i^2-\frac{1}{n}\left(\sum_{i=1}^{n}x_i\right)^2\right] \quad \sigma^2=\frac{1}{N}\sum_{i=1}^{n}(x_i-\mu)^2=\frac{1}{N}\sum_{i=1}^{n}x_i^2-\mu^2$$

**5.Q1:25% Q3:75% Inter-quartile range:**IQR=Q3-Q1

**6.Graphic Displays of Basic Statistical Descriptions:** **Boxplot:** min Q1 Q2/median Q3 max max-Q3/A1-min=whisker **Histogram:** x values, y frequencies, show distributions of variables & binned quantitative data. But bar charts: compare variables & categorical data **Quantile plot:** Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences). **Quantile-q plot:** graphs the quantiles of one univariant distribution against the corresponding quantiles of another **Scatter plot:** Provides a first look at bivariate data to see clusters of points, outliers

**7.Data Visualization: *Pixel oriented:** m dimensions with m windows. To save space and show the connections among multiple dimensions, circle segment ***Geometric Projection: (Direct Data Visualization: Scatterplot Matrices(2D):** k*(k-1)/2 **Landscape:** **Parallel Coordinates:** axes [minimum, maximum]) ***Icon-based:** features of icons. Shape/color/tile bars(**Chernoff Faces/Stick Figures**) ***Hierarchical: (Dimensional Stacking**: Partitioning of the attribute value ranges into classes. **Words within worlds:** Assign the function and two most important parameters to innermost world **Tree-Map: InfoCube:) Tag cloud: Social Networks**

**8.Similarity: *Z-score:** $z=\frac{x-\mu}{\sigma}$

**Minkowski Distance:** $d(i,j)=\sqrt[h]{|x_{i1}-x_{j1}|^h+|x_{i2}-x_{j2}|^h+\dots+|x_{il}-x_{jl}|^h}$

**Manhattan/City Block Distance:** p=1 **Euclidean Distance:** p=2

**Supremum D:** $d(i,j)=\lim_{h\to\infty}\left(\sum_{f=1}^{p}|x_{if}-x_{jf}|^h\right)^{\frac{1}{h}}=\max_f |x_{if}-x_{jf}|$

**9.JaccardCoefficient/Coherence(i,j):** j: 1 0

$J(A,B)=\frac{|A\cap B|}{|A\cup B|}=\frac{|A\cap B|}{|A|+|B|-|A\cap B|}$ $d(i,j)=\frac{r+s}{q+r+s+t}$ symmetric i:1 q r

$d(i,j)=\frac{r+s}{q+r+s}$ asymmetric 0 s t

**Cosine Similarity:** $cos(d_1,d_2)=\frac{d_1\bullet d_2}{\|d_1\|\times\|d_2\|}$

**Chi-square(X2):** $\chi^2=\sum_i\frac{(O_i-E_i)^2}{E_i}$ (observed, expected)

Null hypothesis: two distributions are independent 自由度=(行数-1) *(列数-1) 查表得到不相关的概率，卡方值越大说明关联性越强。

**10.Variance:** $\sigma^2=var(X)=E[(X-\mu)^2]=E[X^2]-\mu^2=E[X^2]-[E(x)]^2$

Sample var: Est avg == real avg, N; Est avg != real avg, N-1

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

Sample cov: (Total Cov/n). >0, pos;独立则==0(反不可); <0, neg

**11. Pearson Correlation**: normalize covariance with standard deviation [-1, 1]

$\rho_{12}=\frac{\sigma_{12}}{\sigma_1\sigma_2}=\frac{\sigma_{12}}{\sqrt{\sigma_1^2\sigma_2^2}}$ >0, pos correlated; ==0, independent; <0, neg

**12.KL Divergence:** measure the difference between two probability distributions over the same x. D(p(X)||q(X)), P true/observation, 1 model/approximation of p, from q(prior( to p(posterior). D>=0 and only p==q,D == 0. P:0,D:0;q:0,D:正无穷

$D_{KL}(p(x)\|q(x))=\sum_{x\in X}p(x)\ln\frac{p(x)}{q(x)}$ smoothing: not counting the possibility of unseen

**Value Range** jacc_coef,=[0,1]; cov=(-inf, inf); KL-div=[0, inf]; z=(-inf,inf); pearson cor coef=[-1, 1]; L_inf=[0, inf]=[min(s(ij)), max(s(ij))] s(ij)=supdist(I,j)

## Data Processing(Chapter3)

**1.Quality:** accuracy, completeness, consistency, timeliness, believability, interpretability

**Cleaning:** Incomplete-ignore, Noisy(Binning-Equal-frequency), Regression, Clustering-outlier), Inconsistent, Intentional;

**2.Integration:** schema integration( feature 一样但不同名)

**3.Data Reduction: Parametric(可以用 mode 表示)/Non-parametric Methods; dependent/response variable, independent/explanatory variable; Regression/ Log-Linear/ Histogram Analysis/ Clustering Analysis/ Sampling(**random sampling-equal probability, without replacement-selected & removed, stratified-partition/cluster first then sampling)/ **Data Cube aggregation/ Data Compression4. Data Transformation: Normalization(min-max/z-score/decimal scaling)**

**4.Discretization: Binning**-unsupervised (Equal-width/distance partition Equal-depth (frequency) partition, Top-down **Histogram**-un,td **Clustering**-un,td or bottom-up **Classification(Decision-Tree**-su,td) **Correlation(x^2)**-su, bu **Concept Hierarchy Generation**

**5.Dimensionality Reduction: Feature Selection:** subset; **Feature extraction: 降维** (**PCA**: linearly uncorrelated variables called principal components. Find the eigenvectors of the covariance matrix, and these eigenvectors define the new space. **Attribute Subset Selection**: Redundant attributes. Irrelevant attributes. **Attribute Creation**: (**Attribute extraction**-domain-specific, **Mapping data to new space-Fourier/wavelet transformation, Attribute Construction**-Combining features)

## Data Warehousing and OLAP for Data Mining (Chapter4)

**1.Basic Concepts:** Subject-oriented, Integrated, Time Variant, Nonvolatile( initial loading of data & access of data).

**2.Type:Enterprise warehouse:** collect all of the information about subject spanning the entire organization **Data Mart:** A subset of corporate-wide data that is of value to a specific groups of users **Virtual warehouse:** a set of views over operational databases

**3.Operations:Data extraction/Data cleaning/ Load/ Data transformation**(host format to warehouse format)/ **Refresh**

**4.Dimension Tables**-item; **Fact Table**-measures; **Data Cube**-(nD: base cuboid, 0D: apex cuboid)

**5.Modeling**:**Star Schema**: A fact table in the middle connected to a set of dimension tables **Snowflake schema:** dimensional hierarchy **Fact constellations:** Multiple fact tables

**6.OLAP Operations: Roll up/Drill-up:** summarize, dimension reduction; **Drill down/roll down:** new dimensions **Slice:** dimension's subset **Dice:** data's subset **Pivot/rotate: Drill Across:** involving across more than one fact table **Drill through:** Through the bottom level of cube to its back-end relational tabl

**7.Measures: Distributive:** if the result derived by applying the function to n aggregate values is the same as that derived by applying the function on all the data without partitioning E.g., count(), sum(), min(), max() **Algebraic:** if it can be computed by an algebraic function with M arguments (where M is a bounded integer), each of which is obtained by applying a distributive aggregate functionavg(x) = sum(x) / count(x), average, std, maxN, minN, CenterOfMass **Holistic:** if there is no constant bound on the storage size needed to describe a subaggregate .E.g., median(), mode(), rank(),Q1, Q3

**8.Materialization: Full/No/Partial Materialization**

**9.How many cuboids in an n-dimensional cube with L levels:**

$T=\prod_{i=1}^{n}(L_i+1)$ 3 dimensions: 3 levels + 4 levels + 5 levels T = (3+1) (4+1) (5+1)

**10.Bitmap Index:** speed up + reduce storage

**11.Architectures: Relational(ROLAP):**use relational or extended-relational DBMS **Multidimensional(MOLAP):** sparse array-based multidimensional storage engine **Hybrid(HOLAP):** low level: relational, high-level: array

*** P:** 和顺序有关 P(n,m) = n! / (n-m)!

*** C:** 和顺序无关 C(n,m)=n! / [(n-m)! * m!]

## Data Cube Technology (Chapter 5)

1. **Base Cell / Aggregate Cell; Full Cube/Iceberg Cube**

**2.Close Cube:** a cell c is closed if there exists no cell d, such that d is a descendant of c, and d has the same measure value as c

**3. Computation Methods:**

- Example: A cube with 100 dimensions
  - Suppose it contains only 2 base cells: $\{(a_1, a_2, a_3, \dots, a_{100}), (a_1, a_2, b_3, \dots, b_{100})\}$
  - How many aggregate cells if "having count >= 1"?
    - Answer: $(2^{101}-2)-4$ (Why?!)

Let cube P have only 2 base cells: $\{(a_1, a_2, a_3, \dots, a_{100}):10, (a_1, a_2, b_3, \dots, b_{100}):10\}$

- How many cells will the iceberg cube contain if "having count(*) ≥ 10"?

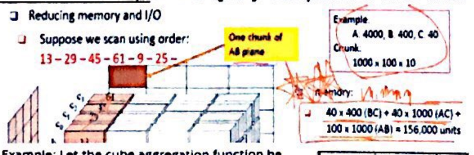
A9 Answer: $2^{101}-4$ (still too big!)

**General Heuristics** (share-sorts/share-partitions **Aggregate** may be computed from previously computed aggregates: **Smallest-child/Cache-results/Amortize-scans)**

**Multi-Way Array Aggregation** (Bottom-up): A>B>C for ABC

**BUC**(Top-down): **A Shell-Fragment Approach** (High-dimensional OLAP) **Online Query Computation with Shell-**

- Reducing memory and I/O
- Suppose we scan using order: 13 – 29 – 45 – 61 – 9 – 25 –




Example: Let the cube aggregation function be count

| 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 |

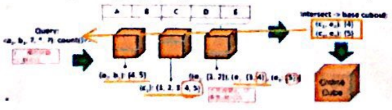
| Attribute Value | TID List | List Size |
|---|---|---|
| a1 | 1 2 3 | 3 |
| a2 | 4 5 | 2 |
| b1 | 1 4 5 | 3 |
| b2 | 2 3 | 2 |
| c1 | 1 2 3 4 5 | 5 |
| d1 | 1 3 4 5 | 4 |
| d2 | 2 | 1 |
| e1 | 1 2 | 2 |
| e2 | 3 4 | 2 |
| e3 | 5 | 1 |

Divide the 5-D table into 2 shell fragments: (A, B, C) and (D, E)

Build traditional invert index or RID list (1-D)

**Fragments**: Frag-Shells -> Offline & Online

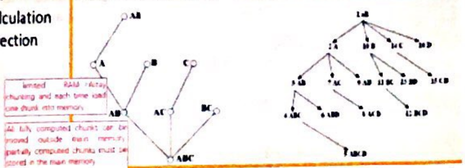
**总结**: **Multiway:** dense full cube. no iceberg, no high-D; **BUC:** ice, no high-D; **shell:** high-D, no dense full cube, no large cube




Given a database of T tuples, D dimensions, and F shell fragment size, the fragment cubes' space requirement is: $\boxed{O\left(T\left\lceil\frac{D}{F}\right\rceil(2^F-1)\right)}$

For F < 5, the growth is sub-linear

| | multiway | BUC |
|---|---|---|
| Input format | Multi-dimensional array | Relational database |
| Good for | Full cube | Iceberg cube |
| Key idea | Simultaneously Aggregation | Partition and sort |
| Calculation direction | | |




1) Suppose a data relation has 100 attributes and 106 tuples. Each attribute has 50 distinct values. Suppose each cell takes 16 bytes of space, and the shell-fragments are all 4 dimensions. (i) [5] What is the size (in bytes) of one pre-computed shell-fragment of size 4? Answer: 16 * (1 + 4 *50 + 6*50^2 + 4* 50^3) Bytes. For the 0-D (apex) cuboid, there is 1 cell. For each 1-D cuboid, there are 50 *1*1*1 = 50 cells. There are 4 such cuboids. For each 2-D cuboid, there are 50*50*1*1 = 502 cells. There are 6 such cuboids. For each 3-D cuboid, there are 50*50*50*1 = 503 cells. There are 4 such cuboids. The total size of these cells is 16* (1 + 4*50 + 6*50^2 + 4* 50^3), or equivalently, 16* ((50 + 1)4- 50*4). Note that if we also count the base cells, since the number of tuples is smaller than 504, we can store the data in 106 cells. We will add 16*106 to the size. (ii) [4] If an OLAP query contains 2 instantiated variables and 6 inquired variables, what is the number of shell fragments this query must access in the best case and in the worst case, respectively? Answer: Best case: 2, worst case: 8. There are 8 dimensions we care about. In the best case, these 8 dimensions happen to be in 2 shell fragments since there are 4 dimensions each. In the worst case, all these 8 dimensions are in different shell fragments and we need to access 8 shell fragments.

2)Cannot support the following operations efficiently? i. Computing an iceberg cube [[Multiway cannot support, aggregation bottom-up and Apriori principle/pruning cannot. ii. Processing an OLAP query on 30 dimensions [Multiway and BUC – 30 dimensions is too many]

3)list one best method and another worst (a) computing a dense full cube of low dimensionality (e.g., less than 6 dimensions), B: multiway. W: shell-fragment (since most part of cube is not precomputed) (b)performing OLAP operations in a high-dimensional database (e.g., over 50 dimen-sions) B: shell-fragment, W: the other (c)computing a large iceberg cube of around 10 dimensions. B: BUC W: multiway

4)CubeSparse Sparse array compression: Use chuck to partition the data, and use (chunk id, offset) to store only those cells contain (nonempty) values.

5) ** Cube & Cuboid** Suppose the base cuboid of a data cube contains two cells(a1; a2; a3; a4; : : : ; a10) : 1, (a1; b2; a3; b4; : : : ; b10) : 1 where ai 6= bi for any I i. [3] How many nonempty cuboids are there in this data cube? Answer: 2^10. Since we have 10 dimensions with no concept hierarchy, there are 2^10 cuboids and all of them should not be empty. ii. [3] How many (nonempty) aggregate closed cells are there in this data cube? Answer: 1. There are 3 closed cells, including the two base cells and (a1; _; a3; _; a5; _; a7; _; a9; _). But only the latter one is a aggregated closed cell. iii. [3] How many (nonempty) aggregate cells are there in this data cube? Answer: 2014. For each base cell, there are 2^10 1 aggregated cells. However, there are 25 cells that are counted twice since there are 5 common dimensions. Therefore, the total number of nonempty aggregate cells is 2*(2^10-1)-2^5 = 2014. iv. [3] If we set minimum support = 2, how many (nonempty) aggregate cells are there in the corresponding iceberg cube? Answer: 2^5. These two base cells have common value in 5 dimensions; therefore, there are 2^5 nonempty cells with support = 2 and all of them are aggregate cells.

6) A cell c is closed if any descendent cell d have a smaller measure than c. measure(closed cell) <= #(base cell)
(a1,a2,c3,...,ck),(a1,b2,c3,...,ck),(b1,a2,c3,...,ck), (b1,b2,c3,...,ck)

* Cuboids = 2^k, given k dimensions

*aggregate Closed cells: (a1,*,c3,...,ck) : 2, (b1,*,c3,...,ck) : 2, (*,a2,c3,...,ck) : 2, (*,b2,c3,...,ck) : 2, (*,*,c3,...,ck) : 4

*aggregate cells: 4×(2^k-1)−4×2^(k−2)−3×2^(k−2)

Hint: "(a1,*,c3,...,ck) : 2" merges 2 × 2k−2 cells into 1 × 2k−2 cells and thus net loss is 2k−2 cells. There are 4 such cells. Thus their total loss is 4 × 2k−2 cells. "(*,*,c3,...,ck) : 4" merges 4 × 2k −2 cells into one 2k−2 cells and thus net loss is 3 × 2k−2.

*If we set min_support = 2, how many aggregate cells are there in the corresponding iceberg cube? 5 × 2^(k−2) since there are 5 aggregate closed cells, each will cover 2k−2 aggregate cells.

7) Explain why good intra-cuboid expansion may enhance the quality of drill-down in sampling cube, but a bad intra-cuboid expansion may reduce its quality. [Enhance quality: if we expand on a dimension that is not correlated with the target measure, we have more data available (so we can have tighter confidence bounds) Reduce quality: if we expand on a dimension correlated with the target measure, the new data will skew the target measure. We effectively end up answering a different question than the one posed]

8) What are the major challenges to support OLAP on sampling data? [Outline a method that may support such operations effectively. Major challenges: Some drilling down cells may contain few or no data. Method: Intra or inter-cuboid expansion to combine with other cells whose attributes has low correlations with the dimensions interested.]

## Mining Frequent Patters, Association and Correlations(char6)

**1.Transactional Database(TDB): itemset, k-itemset, absolute/relative support. Frequent:** if the support of X is no less than a minsup threshold. **Association Rule:**

- We first compute the following two metrics, s and c.
  - Support of $X \cup Y$
    - Ex. s{Diaper, Beer} = 3/5 = 0.6 (i.e., 60%)
  - Confidence of $X \rightarrow Y$
    - The conditional probability that a transaction containing X also contains Y
      $c = sup(X \rightarrow Y) / sup(X)$
    - Ex. c = sup{Diaper, Beer}/sup{Diaper} = ¾ = 0.75

≠

Solution 1 **Closed patterns**: A pattern (itemset) X is closed if X is *frequent*, and there exists *no super-pattern* $Y \supset X$, *with the same support* as X

Ex. $TDB_1$: $T_1$: {$a_1$, ..., $a_{50}$}; $T_2$: {$a_1$, ..., $a_{100}$}

Suppose *minsup* = 1. How many closed patterns does $TDB_1$ contain?

- Two: $P_1$: "{$a_1$, ..., $a_{50}$}: 2"; $P_2$: "{$a_1$, ..., $a_{100}$}: 1"

X {a}   {a, b}, super pattern {a, c}

Solution 2 **Max-patterns**: A pattern X is a max-pattern if X is frequent and there exists no frequent super-pattern $Y \supset X$, ~~same support~~

Difference from close-patterns?

Do not care the real support of the sub-patterns of a max-pattern

Let Transaction DB $TDB_1$: $T_1$: {$a_1$, ..., $a_{50}$}; $T_2$: {$a_1$, ..., $a_{100}$}

Suppose *minsup* = 1. How many max-patterns does $TDB_1$ contain?

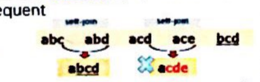
- One: P: "{$a_1$, ..., $a_{100}$}: 1"

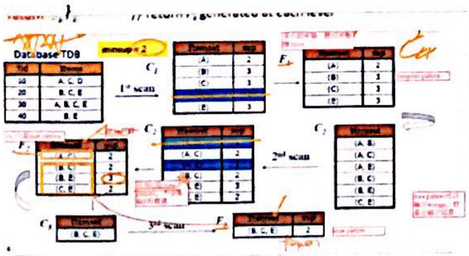
{$a_6$, $a_{55}$} is freq

closed pattern: lossless compression/ Max-pattern: lossy

**2.Downward Closure Property Apriori:** any subset of a frequent itemset must be frequent

```
// Step 1: Joining
for each p in F_{k-1}
   for each q in F_{k-1}
      if p.item_1 = q.item_1, ..., p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1} {
         c = join(p, q)
// Step 2: pruning
         if has_infrequent_subset(c, F_{k-1})
            continue   // prune
         else add c to C_k
```



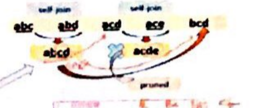


How to generate candidates?

- Step 1: self-joining $F_k$
- Step 2: pruning

Example of candidate-generation

- $F_3$ = {abc, abd, acd, ace, bcd}
- Self-joining: $F_3$*$F_3$
  - abcd from abc and abd
  - acde from acd and ace
- Pruning:
  - acde is removed because ade is not in $F_3$
- $C_4$ = {abcd}



**Partitioning**: any itemset that is potentially frequent in TDB must be frequent in at least one of the partitions of TDB

**Direct Hashing and Pruning:** different itemsets may have the same hash value, can speed up while the size of buckets 重要

**3.Vertical Data Format:**

Using diffset to accelerate mining

reduce storage

- Only keep track of differences of tids
- t(e) = {$T_{10}$, $T_{20}$, $T_{30}$}, t(ce) = {$T_{10}$, $T_{30}$} → Diffset (ce, e) = {$T_{20}$}

Method: Scan DB twice (A. Savasere, E. Omiecinski and S. Navathe, *VLDB'95*)

Scan 1: Partition database so that each partition can fit in main memory (why?)

- Mine local frequent patterns in this partition

Scan 2: Consolidate global frequent patterns

- Find global frequent itemset candidates (those frequent in at least one partition)
- Find the true frequency of those candidates, by scanning TDB one more time

**A transaction DB in Horizontal Data Format**

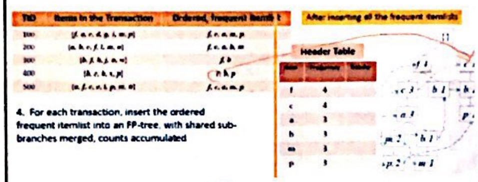
| Tid | Itemset |
|---|---|
| 10 | a, c, d, e |
| 20 | a, b, e |
| 30 | b, c, e |

**The transaction DB in Vertical Data Format**

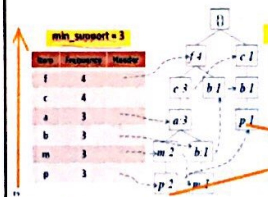
| Item | TidList |
|---|---|
| a | 10, 20 |
| b | 20, 30 |
| c | 10, 30 |
| d | 10 |
| e | 10, 20, 30 |

**4.FP-Tree**

1. Scan DB once, find single item frequent pattern:
   Let min_support = 3
   f:4, a:3, c:4, b:3, m:3, p:3
2. Sort frequent items in frequency descending order, f-list   F-list = f-c-a-b-m-p
3. Scan DB again, find the ordered frequent itemlist for each transaction



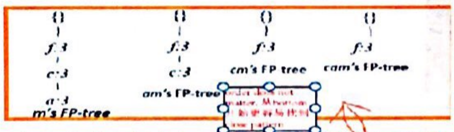
4. For each transaction, insert the ordered frequent itemlist into an FP-tree, with shared sub-branches merged, counts accumulated



**Conditional database of each pattern**

| Item | Conditional database |
|---|---|
| c | f:3 |
| a | fc:3 |
| b | fca:1, f:1, c:1 |
| m | fca:2, fcab:1 |
| p | fcam:2, cb:1 |

Mine Each Conditional Database recursively:

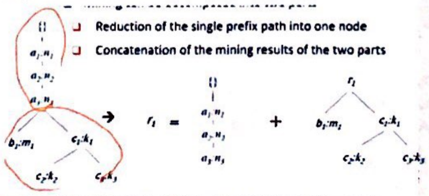


Actually, for single branch FP-tree, all the frequent patterns can be generated in one shot

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

Single Prefix Path in FP-tree;

- Reduction of the single prefix path into one node
- Concatenation of the mining results of the two parts



Scaling FP-growth by Item-Based Data Projection

Closet" mining closed itemsets by pattern-growth

**5.Pattern Evaluation: Strong Association Rule**

**6.Interstingess Measure:**

**Lift**(dependent/correlated)

$$lift(B, C) = \frac{c(B \rightarrow C)}{s(C)} = \frac{s(B \cup C)}{s(B) \times s(C)}$$ X2

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

Lift(B, C) may tell how B and C are correlated

- Lift(B, C) = 1: B and C are independent
- \> 1: positively correlated
- < 1: negatively correlated

| | B | ¬B | $\Sigma_{row}$ |
|---|---|---|---|
| C | 400 (450) | 350 (300) | 750 |
| ¬C | 200 (150) | 50 (100) | 250 |
| $\Sigma_{col}$ | 600 | 400 | 1000 |

**7.Null-Invariance:** value does not change with the # of null-transactions。改变 null-transactions 时 lift 和 x2 会变，所以不是 null-invariant. Not good to evaluate data.

**8. Imbalance Ratio with Kulczynski Measure:**

$$IR(A, B) = \frac{|s(A) - s(B)|}{s(A) + s(B) - s(A \cup B)}$$

**Null invariant** Suppose a transaction database contains N transactions, ct transactions contain both coffee and tea, ct transactions contain coffee but not tea, ct transactions contain tea but not coffee, and ct transactions contain neither tea nor coffee. At what condition that Lift may not be a good measure to indicate correlations between coffee and tea? and why? 【】

Condition: If the number of ct transactions is substantially larger than the remaining # of transactions, LIFT is not a good measure. / Reasoning: Lift is not null-invariant. That is, although LIFT is affected by the number of ct transactions, the correlation should not be (because transactions that do not contain c or t tell us nothing about the correlation - so they shouldn't affect the measure).

** **Null invariant** **Explain why null-invariance property is important in the study which authors are"correlated? 【】 Since most authors are not coauthoring papers, null value is very high. Null-invariance is critical otherwise the "correlation value could be greatly influenced by null values.

(1) mine FPs from the new DB; (2) get counts of those patterns frequent in new DB but not in old DB by scanning the old DB once, (3) get counts of those that are frequent in old DB by not in new DB. The merge counts will be used to just which patterns are globally frequent

| Measure | Definition | Range | Null-Invariant? |
|---|---|---|---|
| $\chi^2(A, B)$ | $\sum_{i,j} \frac{(e(a_i, b_j) - o(a_i, b_j))^2}{e(a_i, b_j)}$ | [0, ∞] | No |
| $Lift(A, B)$ | $\frac{s(A \cup B)}{s(A) \times s(B)}$ | [0, ∞] | 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 |
| $MaxConf(A, B)$ | $max\{\frac{s(A \cup B)}{s(A)}, \frac{s(A \cup B)}{s(B)}\}$ | [0, 1] | Yes |

Let n be the total number of transactions, and count(¬(A ∪ B)) be the number of null-

distance

| Measure | Input | Range | Pros/Cons |
|---|---|---|---|
| Minkowski | $\left(\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p\right)^{1/p}$ | 0 → ∞ | Pros: Most commonly used distance for numerical data; Positivity/Symmetry/Triangle Inequality |
| Manhattan | Minkowski, p = 1; $\sum_{i=1}^{n} \lvert x_i - y_i \rvert$ | 0 → ∞ | Pros: Not sensitive to outliers. Cons: Non differentiable |
| Euclidean | Minkowski, p = 2; $\left(\sum \lvert x_i - y_i \rvert^2\right)^{1/2}$ | 0 → ∞ | Pros: differentiable. Cons: Sensitive to outliers |
| Supremum | Minkowski, p → ∞; $\max_{f=1} \lvert x_{if} - y_{jf} \rvert$ | 0 → ∞ | |

normalization

| Measure | Input | Range | Pros/Cons |
|---|---|---|---|
| Z-Score | $z = \frac{X - \mu}{\sigma}$ | [−∞, +∞] But scores outside [−3, 3] are likely to be outliers | Pros: Easy to calculate; Good for outlier detection. Cons: Small data sets skew the results |
| Mean Absolute Deviance | $\frac{\sum_{i=1}^{n} \lvert x_i - \bar{x} \rvert}{n}$ | [0, +∞] | |
| Min/Max Normalization | $v_i' = \frac{v_i - min_A}{max_A - min_A} \times (new\_max_A - new\_min_A) + new\_min_A$ | new_min_A → new_max_A | Pros: Allows for custom range of data |

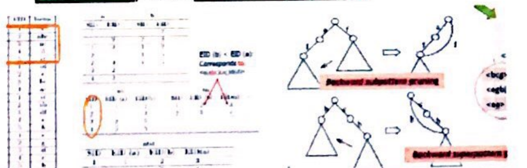
## Advanced Frequent Pattern Mining(Chapter 7)

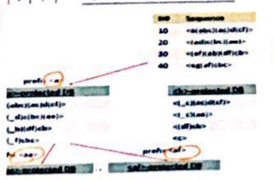
**1. Constrain-Based Frequent Pattern Mining(Rule/Pattern Constraint):** **anti-monotone** sum(S.price) <=v, range(S.profit)<=15, min(S.value) > 10, sum(S.Price)>=v, support(S)>=σ **Monotone** sum(S.Price) >=v, min(S.Price)<=v, range(S.profit)>=15, element_count (S) > 5; S 属于{PC, digital_camera} **Convertible** e.g. avg(S.profit) > 20 Order items in (profit) value-descending order <a, g, h, b, f, d, c, e>. An itemset ab violates c1 (avg(ab) = 20), So does ab* (i.e., ab-projected DB), C1: anti-monotone if patterns grow in the right order! (item-reordering can't work for Apriori) value_avg(S) < 25; profit_sum (S) > 16, max(S)/avg(S) < 2; median(S) – min(S) > 5 **Succintness** min(S.Price)<=v, S 属于 {i-phone, MacAir}, sum(S.Price)v is not succinct **Data Anti-Monotone** sum(S.Profit) >= v is data anti-monotone. Let constraint c1 be: sum(S.Profit) ≥ 25, T30: {b, c, d, f, g} can be removed since none of their combinations can make an S whose sum of the profit is ≥ 25; min(S.Price)<=v, Consider v = 5 but every item in a transaction, say T50 , has a price higher than 10; ange(S.Profit) > 25

| ce pruning constraints | Data space pruning constraints |
|---|---|
| i-monotonic: If constraint c is violated, its further ing can be terminated<br>notonic: If c is satisfied, no need to check c again<br>nvertible: c can be converted to monotonic or i-monotonic if items can be properly ordered in cessing<br>cinct: If the constraint c can be enforced by ectly manipulating the data | ■ Data succinct: Data space can be pruned at the initial pattern mining process<br>■ Data anti-monotonic: If a transaction t does not satisfy c, then t can be pruned to reduce data processing effort |

Apriori + Succinct Constraint


**2. Sequential Pattern Mining:** **GSP** Apriori-Based Sequential Pattern Mining (If a subsequence s1 is infrequent, none of s1's super-sequences can be frequent) **SPADE** Sequential Pattern Mining in Vertical Data Format **PrefixSpan** Sequential Pattern Mining by Pattern-Growth **CloSpan** Mining Closed Sequential Patterns, e.g. NO: <abc>: 20, YES: <abcd>:20, <abcde>: 15 **Pseudo-Projection** pointer **Physical Projection**

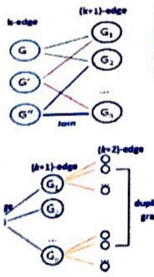




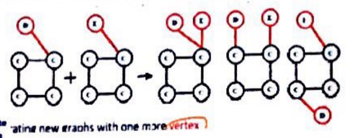
**Order constraint** Some items must happen before the other **Min-gap/max-gap constraint** Confines two elements in a pattern **Max-span constraint** Maximum allowed time difference between the 1st and the last elements in the pattern

E.g., maxspan (S) = 60 (days) **Window size constraint** Events in an element do not have to occur at the same time: Enforce max allowed time difference

**3. Sequential Pattern Mining:** **Frequent/Sub Graph Patterns** **gSPAN: Graph Pattern Growth in Order/Right most path extension** A pattern-Growth-Based Patterns the path from root to the right-most leaf, choose the vertex with the smallest index at each step; **CloseGraph** G and G1 are frequent, and G is a subgraph of G1. If in any part of the graph in the dataset where G occurs, G1 also occurs, then we need not grow G

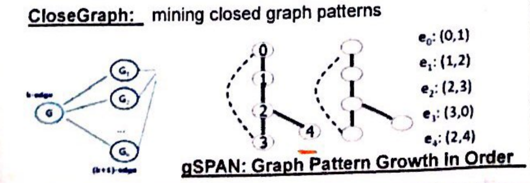


**Apriori-Based Approach** mining methods **Vertex Growing; Edge Growing**



**Pattern-Growth Approach**

**CloseGraph:** mining closed graph patterns




gSPAN: Graph Pattern Growth in Order

If feature $x_k$ is continuous-valued, $p(x_k = v_k|C_i)$ is usually computed based on Gaussian distribution with a mean μ and standard deviation σ

$$p(x_k = v_k|C_i) = N(x_k|\mu_{C_i}, \sigma_{C_i}) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{(x-\mu_{C_i})^2}{2\sigma^2}}$$

Use **Laplacian correction** (or Laplacian estimator)

- *Adding 1 to each case*
  Prob(income = low) = 1/(1000 + 3)

$$a_{n+1} = a_n - \gamma \nabla F(a_n)$$

- Calculation: For a discrete random variable Y taking m

$$H(Y) = -\sum_{i=1}^{m} p_i \log(p_i) \quad where\ p_i = P(Y = y_i)$$

- Interpretation
  - Higher entropy → higher uncertainty
  - Lower entropy → lower uncertainty
- Conditional entropy

$$H(Y|X) = \sum_x p(x) H(Y|X = x)$$

- Select the attribute with the highest information gain (used in typical decision tree induction algorithm: ID3/C4.5)
- Let $p_i$ be the probability that an arbitrary tuple in D belongs to class $C_i$, estimated by $|C_{i,D}|/|D|$
- Expected information (entropy) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

- Information gained by branching on attribute A

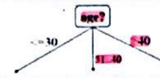
$$Gain(A) = Info(D) - Info_A(D)$$

### Example: Attribute Selection with Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

| age | $p_i$ | $n_i$ | $I(p_i, n_i)$ |
|---|---|---|---|
| <=30 | 2 | 3 | 0.971 |
| 31..40 | 4 | 0 | 0 |
| >40 | 3 | 2 | 0.971 |




$$Info_{age}(D) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.694$$

$\frac{5}{14} I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

- Gain ratio: Overcomes the problem (as a normalization to information gain)

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$$

- GainRatio(A) = Gain(A)/SplitInfo(A)
- The attribute with the maximum gain ratio is selected as the splitting attribute
- Gain ratio is used in a popular algorithm C4.5 (a successor of ID3) by R. Quinlan
- Example
  - $SplitInfo_{income}(D) = -\frac{4}{14}\log_2\frac{4}{14} - \frac{6}{14}\log_2\frac{6}{14} - \frac{4}{14}\log_2\frac{4}{14} = 1.557$
  - GainRatio(income) = 0.029/1.557 = 0.019

$$gini_A(D) = \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

- Reduction in Impurity:
  - $\Delta gini(A) = gini(D) - gini_A(D)$
- The attribute provides the smallest $gini_{split}(D)$ (or the largest reduction in impurity) is chosen to split the node (*need to enumerate all the possible spl.*
- Example: D has 9 tuples in buys_computer = "yes" and 5 in "no"

$$gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

- Suppose the attribute income partitions D into 10 in $D_1$: {low, medium} and 4 in $D_2$

$$gini_{income \in \{low, medium\}}(D) = \frac{10}{14} gini(D_1) + \frac{4}{14} gini(D_2)$$

$$= \frac{10}{14}\left(1 - \left(\frac{7}{10}\right)^2 - \left(\frac{3}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2\right) = 0.443$$

- $Gini_{\{low,high\}}$ is 0.458; $Gini_{\{medium,high\}}$ is 0.450
- Thus, split on the {low,medium} (and {high}) since it has the lowest Gini index
- If a data set $D$ contains examples from $n$ classes, gini index, $gini(D)$ is defined
  - $gini(D) = 1 - \sum_{j=1}^{n} p_j^2$ [0,1) $1 - 0^2 - 1^2 = 0$ — pure, 1 class
  - $p_j$ is the relative frequency of class $j$ in $D$
- What is the range of Gini index?
  - The minimum= 0, meaning pure
  - The maximum= ? What is the case that Gini index reach the maximum?

$$p(B) = \sum_i p(B|A_i)p(A_i)$$

Bayes' Theorem:

$$p(H|X) = \frac{p(X|H)P(H)}{p(X)} \propto p(X|H)\,P(H)$$

posteriori probability · likelihood · prior probability

| | Cancer (1%) | No Cancer (99%) |
|---|---|---|
| Test Pos | 80% | 9.6% |
| Test Neg | 20% | 90.4% |

- Only 1% people have cancer
- How accurate is the test?
  - 80%? 99%? 1%?

- P(XH) = P(X|H)P(H)
- Chance of true positive is thus 1%*80% = 0.008
- According to Bayes' Theorem, P(H|X) = P(X|H)P(H)/P(X), the chance of having a cancer given positive test results is

True pos / (True pos + False pos) = 0.008 / (0.008+0.09504) = 7.76%

## Classification (Chapter 8)

**1. Decision Tree: Pros:** Easy to explain, Easy to implement (many software), Efficient, Can tolerant missing data, White box, No need to normalize data, Non-parametric: No assumption on data distribution, no assumption on, attribute independencyCan work on various attribute types

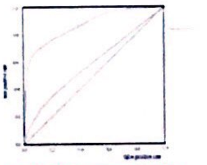
**Cons:** Unstable. Sensitive to noise, Accuracy may be not good enough (depending on your data), The optimal splitting is NP. Greedy algorithms are used, Overfitting **Information Gain/Entropy:** biased towards multivalued attributes **Gain Ratio:** tends to prefer unbalanced splits in which one partition is much smaller than the others **Gini Index:** biased to multivalued attributes, has difficulty when # of classes is large, tends to favor tests that result in equal-sized partitions and purity in both partitions **Pre-prunning Post-Pruning** Relatively fast learning speed, Convertible to simple and easy to understand classification rules, Easy to be adapted to database system implementations (e.g., using SQL). Comparable classification accuracy with other methods,Easy to ensemble, i.e., random forests, xgboost

**2. Bayes Classification Methods: Pros:** Performance: A naive Bayesian classifier, has comparable performance with decision tree and selected neural network classifiers; Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct—prior knowledge can be combined with observed data **Cons:** attributes conditional independence, therefore loss of accuracy

**3. Linear Classifier: Logistic Regression: Pros:** handle multiple types of features, Fast and easy, more robust and better performance than tree, both weights and predicted value **Cons:** if the decision boundary is not linear, then LR is not good **Generative Classifier:** Naïve Bayes, Bayesian Network **Discriminative Classifier:** Logistic Regression, Support Vector Machine Pro: accuracy is generally high, Robust, Fast evaluation of the learned target function Cons: Long training time, Difficult to understand the learned function (weights), Not easy to incorporate domain knowledge

**4. Model Evaluation and Selection:**

**Holdout; Cross-Validation; Bootstrap; ROC:** The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model **Oversampling:** Oversample the minority class. **Under-sampling:** Randomly eliminate tuples from majority class **Synthesizing:** Synthesize new minority classes **Threshold-moving; Class weight adjusting; Ensemble techniques:**



Sigmoid function:

- $\sigma(y) = \frac{1}{1+e^{-y}} = \frac{e^y}{e^y+1}$
- Projects $(-\infty, +\infty)$ to $[0, 1]$

- Vertical axis represents the true positive rate (TP/P)
- Horizontal axis rep. the false positive rate (FP/N)

- Use the same confusion matrix, calculate the measure just introduced

| Actual Class\Predicted class | cancer = yes | cancer = no | Total |
|---|---|---|---|
| cancer = yes | 90 TP | 210 FN | 300 |
| cancer = no | 140 FP | 9560 TN | 9700 |
| Total | 230 | 9770 | 10000 |

- Sensitivity = TP/P = 90/300 = 30%
- Specificity = TN/N = 9560/9700 = 98.56%
- Accuracy = (TP + TN)/All = (90+9560)/10000 = 96.50%
- Error rate = (FP + FN)/All = (140 + 210)/10000 = 3.50%
- Precision = TP/(TP + FP) = 90/(90 + 140) = 90/230 = 39.13%
- Recall = TP/ (TP + FN) = 90/(90 + 210) = 90/300 = 30.00%
- F1 = 2 P × R /(P + R) = 2 × 39.13% × 30.00%/(39.13% + 30%) = 33.9

| Measure | Input | Range | Pros/Cons |
|---|---|---|---|
| Symmetric Binary Variable | $\frac{r+s}{q+r+s+t}$ | [0, 1] | • Null variant<br>• if 0 and 1 are equally important |
| Asymmetric Binary Variable | $\frac{r+s}{q+r+s}$ | [0, 1] | • Null invariant<br>• If 0 is not important (such as meaning did appear, too common in data, ...) |
| Jaccard Coefficient / Coherence | $\frac{q}{q+r+q+s-q}$ | [0, 1] | • This is a similarity measure<br>• The higher the value, the more similar the vector |

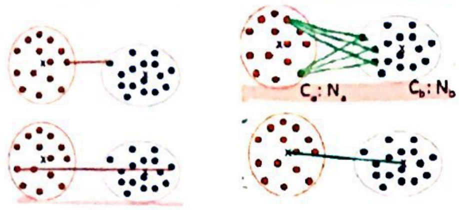
| | 1 | 0 | sum |
|---|---|---|---|
| 1 | q | r | q + r |
| 0 | s | t | s + t |
| sum | q + s | r + t | p |

## Cluster Analysis (Chapter 10)

**1. Partition Methods: KMeans/ K-median** **Pros:** efficient **Cons:** terminate at a local optimal, need to specify k, sensitive to noisy data and outliers

**2. Hierarchical Methods: AGNES** **Pros:** efficient **Cons:** terminate at a local optimal, need to specify k, sensitive to noisy data and outliers

- Single link (nearest neighbor)
- Average link (group average)
- Complete link (diameter)
- Centroid link (centroid similarity)




Single link performs well on nonglobular data, but it performs poorly in the presence of noise.; Average and Complete Linkage performs well on globular data, but have mixed results otherwise.; Ward is the most effective method for noisy data.

## Classification (Chapter 9)

**1.Ensemble Methods:**

Bagging: Trains each model using a subset of the training set, and models learned in parallel resample

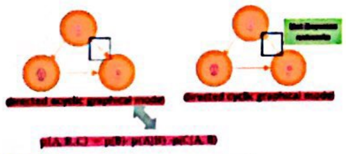
Boosting: Trains each new model instance to emphasize the training instances that previous models mis-classified, and models learned in order

Forest-RI (random input selection): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size

Forest-RC (random linear combinations): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)

Pros: good performance for tabular data(random forest XGBoost), can scale to large datasets, can handle missing data Cons: can overfit, lack of interpretability

**2.Bayesian Networks:** feature is independent(directed acyclic graph)

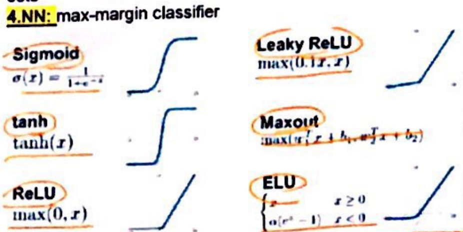


**3.SVM:** max-margin classifier MMH

- $\xi$: slack variable
- C (> 0): Controls the trade-off between the penalty and the margin
  - Smaller C: allow more mistake
  - Larger C: allow less mistake
  - This is the widely used *soft-margin SVM*

Pros: Elegant mathematical formulation, guaranteed global optimal with optimization, trains well on small data sets, flexibility through kernel functions, conformity with semi-supervise training Cons: not naturally scalable to large data sets

**4.NN:** max-margin classifier




2. **Compute Gradient**: For each parameter $\theta_i$ (weights, bias), compute its gradient $\frac{\partial}{\partial \theta_i} l_\theta$ Explained later
3. **Update Parameter**: $\theta_i = \theta_i - \alpha \cdot \frac{\partial}{\partial \theta_i} l_\theta$

Pros: very good performance on certain tasks, little feature engineering Cons: huge amounts of computation power, black box model, hard to tune the architecture and hyperparameters

**Lazy learning** (e.g., instance-based learning): Simply stores training data (or only minor processing) and waits until it is given a test tuple

**Eager learning** (the above discussed methods): Given a set of training tuples, constructs a classification model before receiving new (e.g., test) data to classify

Lazy: knn, locally weighted regression, case-based reasoning

KNN: small k: overfitting, big K: too many irrelevant points

**3.Gaussian Mixture Model & EM:** Hard Clustering: Soft Clustering GMM. **pros:** more general, small number of parameters, satisfy the statistical assumptions **cons:** converge to local optimal, computationally more expensive, hard to estimate number of clusters, spherical cluster

$$P(x|c_j,\theta_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}}$$

- Iteratively do the following two steps
  - E-Step: Evaluate the soft clustering probability according to $\mu_j^t, \sigma_j^t, w_j^t$
    - $w_{ij}^{t+1} = \frac{w_j^t P(x_i|\mu_j^t, \sigma_j^t)}{\sum_k w_k^t P(x_i|\mu_k^t, \sigma_k^t)}$
  - M-Step: Find the new parameters $\mu_j^t, \sigma_j^t$ that maximize log likelihood. In Gaussian distribution, this is equivalent to do parameter estimation when each data point has a weight.
    - $\mu_j^{t+1} = \frac{\sum_i w_{ij}^{t+1} x_i}{\sum_i w_{ij}^{t+1}}$, $(\sigma_j^2)^{t+1} = \frac{\sum_i w_{ij}^{t+1}(x_i - \mu_j^{t+1})^2}{\sum_i w_{ij}^{t+1}}$ ← Weighted average mean and variance
    - $w_j^{t+1} = \frac{\sum_i w_{ij}^{t+1}}{n}$

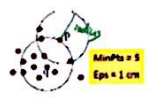
**4.Density-Based Methods:**

Eps ($\varepsilon$): Maximum radius of the neighborhood

MinPts: Minimum number of points in the Eps-neighborhood of a point

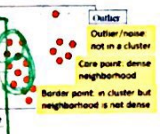
The Eps($\varepsilon$)-neighborhood of a point $q$:

$N_{Eps}(q)$: {p belongs to D | dist(p, q) ≤ Eps}




**Algorithm**

- Arbitrarily select a point $p$
- Retrieve all points density-reachable from $p$ w.r.t. *Eps* and *MinPts*
  - If $p$ is a core point, a cluster is formed
  - If $p$ is a border point, no points are density-reachable from $p$, and DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed




**Directly density-reachable:**

A point $p$ is directly density-reachable from a point $q$ w.r.t. *Eps* ($\varepsilon$), *MinPts* if

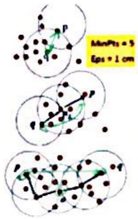
- $p$ belongs to $N_{Eps}(q)$
- core point condition: $|N_{Eps}(q)| \ge MinPts$

**Density-reachable:**

A point $p$ is density-reachable from a point $q$ w.r.t. *Eps*, *MinPts* if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that $p_{i+1}$ is directly density-reachable from $p_i$

**Density-connected:**

A point $p$ is density-connected to a point $q$ w.r.t. *Eps*, *MinPts* if there is a point $o$ such that both $p$ and $q$ are density-reachable from $o$ w.r.t. *Eps* and *MinPts*




**5.Density and Grid Based Methods:** pros: find subspaces of the highest dimensionality, insensitive to the order of records, scales linearly with the size cons: sensitive to parameters

| Measure | Input | Range | Pros/Cons |
|---|---|---|---|
| Cosine Similarity | $\cos(d_1, d_2) = \frac{d_1 \bullet d_2}{\|d_1\| \times \|d_2\|}$ | $[-1, 1]$ In many applications, $d_i$ are all positive, then $[0, 1]$ | Commonly used in text mining<br>1-> similar<br>0-> irrelevant<br>-1-> opposite |
| Chi-Squared Test | $\chi^2 = \sum_{i}^{n} \frac{(O_i - E_i)^2}{E_i}$ | $[0, +\infty]$ | Correlation measure for categorical data<br>Higher value->strong correlation |
| Variance / Covariance | $\sigma_{12} = E[(X_1 - \mu_1)(X_2 - \mu_2)]$ | $[-\infty, +\infty]$ | Correlation measure for continuous data<br>High positive value->strong positive correlation<br>Very negative value->strong negative correlation |
| Correlation coefficient | $\rho_{12} = \frac{\sigma_{12}}{\sigma_1 \sigma_2} = \frac{\sigma_{12}}{\sqrt{\sigma_1^2 \sigma_2^2}}$ | $[-1, 1]$ | Correlation measure for continuous data<br>High positive value->strong positive correlation<br>Very negative value->strong negative correlation |

**6.Evaluation of Clustering:**

**External**: Supervised, employ criteria not inherent to the dataset

- Compare a clustering against prior or expert-specified knowledge (i.e., the ground truth) using certain clustering quality measure

**Internal**: Unsupervised, criteria derived from data itself

- Evaluate the goodness of a clustering by considering how well the clusters are separated and how compact the clusters are, e.g., silhouette coefficient

**Relative**: Directly compare different clusterings, usually those obtained via different parameter settings for the same algorithm

### Relative Measure

Relative measure: Directly compare different clusterings, usually those obtained via different parameter settings for the same algorithm

**Silhouette coefficient** as an **internal measure**: Check cluster cohesion and separation

- For each point $x_i$, its silhouette coefficient $s_i$ is: $\frac{\mu_{out}^{min}(x_i) - \mu_{in}(x_i)}{\max\{\mu_{out}^{min}(x_i), \mu_{in}(x_i)\}}$
  where $\mu_{in}(x_i)$ is the mean distance from $x_i$ to points in its own cluster
  $\mu_{out}^{min}(x_i)$ is the mean distance from $x_i$ to points in its closest cluster
- Silhouette coefficient (SC) is the mean values of $s_i$ across all the points: $\frac{1}{n}\sum_{i=1}^{n} s_i$
- SC close to +1 implies good clustering
  - Points are close to their own clusters but far from other clusters

**Silhouette coefficient as a relative measure**: Estimate the # of clusters in the data

$SC_i = \frac{1}{n_i} \sum_{x_j \in C_i} s_j$ Pick the $k$ value that yields the best clustering, i.e., yielding high values for $SC$ and $SC_i$ ($1 \le i \le k$)

**Purity**: Quantifies the extent that cluster $C_i$ contains points only from one (ground truth) partition $purity = \frac{1}{n}\sum_{i=1}^{r} \max_j \{n_{ij}\}$

- Total purity of clustering C
- Perfect clustering if purity = 1 and $r = k$ (the number of clusters obtained is the same as that in the ground truth)
- Ex. 1 (green or orange): $purity_1 = 30/50$, $purity_2 = 20/25$, $purity_3 = 25/25$, $purity = (30 + 20 + 25)/100 = 0.75$
- Two clusters may share the same majority partition

| C\T | $T_1$ | $T_2$ | $T_3$ | Sum |
|---|---|---|---|---|
| $C_1$ | 0 | 20 | 30 | 50 |
| $C_2$ | 0 | 20 | 5 | 25 |
| $C_3$ | 25 | 0 | 0 | 25 |
| $m_j$ | 25 | 40 | 35 | 100 |

| C\T | $T_1$ | $T_2$ | $T_3$ | Sum |
|---|---|---|---|---|
| $C_1$ | 0 | 30 | 20 | 50 |
| $C_2$ | 0 | 20 | 5 | 25 |
| $C_3$ | 25 | 0 | 0 | 25 |
| $m_j$ | 25 | 50 | 25 | 100 |

- **Maximum matching**: Only one cluster can match one partition
  - Match: Pairwise matching, weight $w(e_{ij}) = n_{ij}$, $w(M) = \sum w(e)$
  - Maximum weight matching: $match = \arg\max \frac{w(M)}{n}$
  - Ex2. (green) $match = purity = 0.75$; (orange) $match = 0.65 > 0.6$

- **Precision**: The fraction of points in $C_i$ from the majority partition $T_{j_i}$ (i.e., the same as purity), where $j_i$ is the partition that contains the maximum # of points from $C_i$ $prec_i = \frac{1}{n_i}\max_{j=1}^{k}\{n_{ij}\} = \frac{n_{ij_i}}{n_i}$
  - Ex. For the green table
    - $prec_1 = 30/50$; $prec_2 = 20/25$; $prec_3 = 25/25$
- **Recall**: The fraction of point in partition $T_{j_i}$ shared in common with cluster $C_i$, where $m_{j_i} = |T_{j_i}|$ $recall_i = \frac{n_{ij_i}}{|T_{j_i}|} = \frac{n_{ij_i}}{m_{j_i}}$
  - Ex. For the green table
    - $recall_1 = 30/35$; $recall_2 = 20/40$; $recall_3 = 25/25$
- **F-measure** for $C_i$: The harmonic means of $prec_i$ and $recall_i$: $F_i = \frac{2 n_{ij_i}}{n_i + m_{j_i}}$
- F-measure for clustering C: average of all clusters: $F = \frac{1}{r}\sum_{i=1}^{r} F_i$
  - Ex. For the green table
    - $F_1 = 60/85$, $F_2 = 40/65$; $F_3 = 1$ $F = 0.774$ $F_1$: precision + Recall $\frac{30}{50} + \frac{30}{35} = \frac{60}{85}$

| C\T | $T_1$ | $T_2$ | $T_3$ | Sum |
|---|---|---|---|---|
| $C_1$ | 0 | 20 | 30 | 50 |
| $C_2$ | 0 | 20 | 5 | 25 |
| $C_3$ | 25 | 0 | 0 | 25 |
| $m_j$ | 25 | 40 | 35 | 100 |

- Entropy of clustering C: $H(C) = -\sum_{i=1}^{r} p_{C_i} \log p_{C_i}$ $p_{C_i} = \frac{n_i}{n}$ (i.e., the probability of cluster $C_i$)
- Entropy of partitioning T: $H(T) = -\sum_{j=1}^{k} p_{T_j} \log p_{T_j}$
- Entropy of T with respect to cluster $C_i$: $H(T|C_i) = -\sum_{j=1}^{k}\left(\frac{n_{ij}}{n_i}\right)\log\left(\frac{n_{ij}}{n_i}\right)$
- Conditional entropy of T with respect to clustering C: $H(T|C) = \sum_{i=1}^{r} \frac{n_i}{n} H(T|C_i) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij} \log\left(\frac{p_{ij}}{p_{C_i}}\right)$
  - The more a cluster's members are split into different partitions, the higher the conditional entropy
  - For a perfect clustering, the conditional entropy value is 0

$$H(T|C) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij}(\log p_{ij} - \log p_{C_i}) = -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij}\log p_{ij} + \sum_{i=1}^{r}\left(\log p_{C_i}\sum_{j=1}^{k} p_{ij}\right)$$

$$= -\sum_{i=1}^{r}\sum_{j=1}^{k} p_{ij}\log p_{ij} + \sum_{i=1}^{r} p_{C_i}\log p_{C_i} = H(C,T) - H(C)$$

- Given a clustering $C = \{C_1, \ldots, C_k\}$ with $k$ clusters, cluster $C_i$ containing $n_i = |C_i|$ points
  - Let $W(S, R)$ be sum of weights on all edges with one vertex in S and the other in R
  - The sum of all the intra-cluster weights over all clusters: $W_{in} = \frac{1}{2}\sum_{i=1}^{k} W(C_i, C_i)$
  - The sum of all the inter-cluster weights: $W_{out} = \frac{1}{2}\sum_{i=1}^{k} W(C_i, \overline{C_i}) = \sum_{i=1}^{k-1}\sum_{j>i} W(C_i, C_j)$
  - The number of distinct intra-cluster edges: $N_{in} = \sum_{i=1}^{k}\binom{n_i}{2}$
  - The number of distinct inter-cluster edges: $N_{out} = \sum_{i=1}^{k-1}\sum_{j=i+1}^{k} n_i n_j$

- **Beta-CV measure**: $BetaCV = \frac{W_{in}/N_{in}}{W_{out}/N_{out}}$
  - The ratio of the mean intra-cluster distance to the mean inter-cluster distance
  - The smaller, the better the clustering

- **Jaccard coefficient**: Fraction of true positive point pairs, but after ignoring the true negatives (thus asymmetric)
  - $Jaccard = TP/(TP + FN + FP)$ [i.e., denominator ignores $TN$]
  - Perfect clustering: $Jaccard = 1$
- **Rand Statistic**
  - $Rand = (TP + TN)/N$
  - Symmetric; perfect clustering: $Rand = 1$
- **Fowlkes-Mallow Measure**
  - Geometric mean of precision and recall
    $FM = \sqrt{prec \times recall} = \frac{TP}{\sqrt{(TP+FN)(TP+FP)}}$