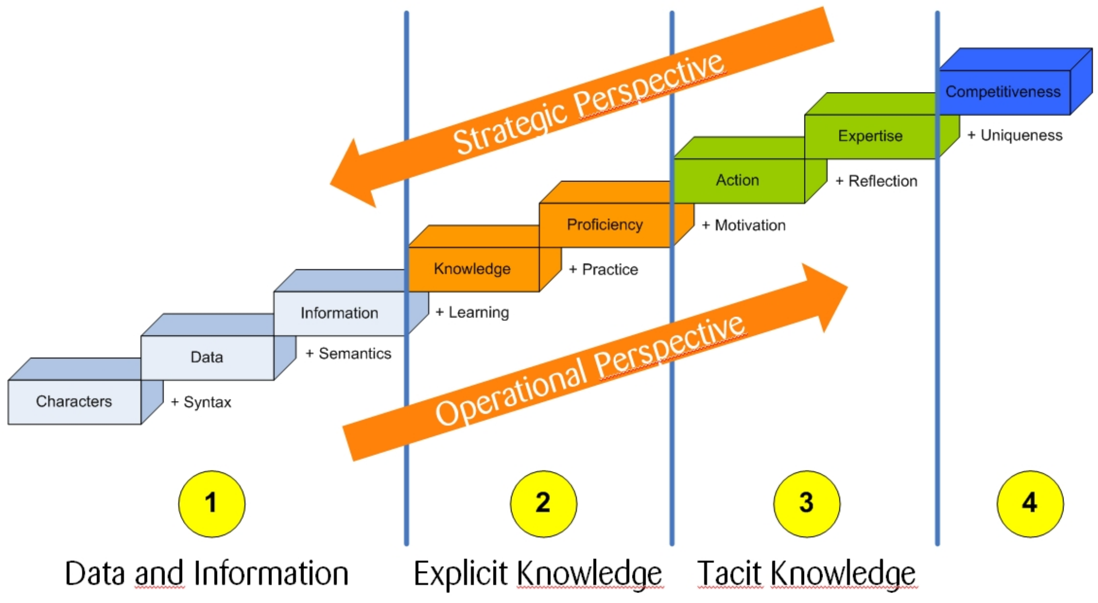
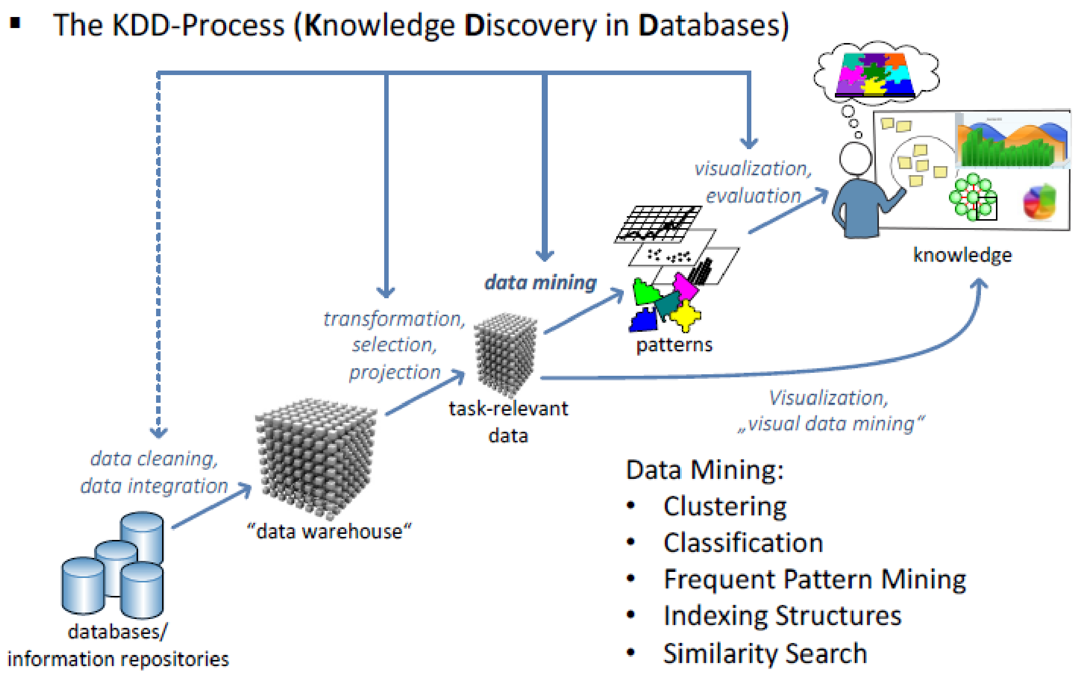
Data Mining Algorithms – summary

Contents

Introduction

- 

- 
data cleaning, data integration
transformation, selection, projection
data mining
visualization, evaluation
knowledge
patterns
task-relevant data
Visualization, „visual data mining"
"data warehouse"
databases/ information repositories

▪ The KDD-Process (**K**nowledge **D**iscovery in **D**atabases)

Data Mining:
- Clustering
- Classification
- Frequent Pattern Mining
- Indexing Structures
- Similarity Search

- 上图中第二步 transformation 包括 3 类：
  Normalization: e.g. 0 mean 1 variance
  Discretization: [0,100] -> low, medium, high
  Derive rows; attributes: total amount per month; diff = current – previous
- Prediction vs classification: numerical output (by regression)/ classes
- Data mining functionalities: characterization, discrimination(前俩是根据现实层级知识处理数据), association, classification, clustering, outlier and trend analysis, etc.

Data warehousing 1

- Values 有两类：
  Categorical:
    Nominal: No natural order among values
    Ordinal: Some ordering exists
  Numerical:
    Discrete: Possible values can be enumerated
    Continuous: Infinitely many possible values
- Dimensionality of data:
  1. One-dimensional
  2. Multi-dimensional
  3. High-dimensional
  4. No-dimensional: always talk about metric data

  Metric data is parallel to this categories.
- Metric data has a metric distance: e.g. images, graphs, itemset ..
  1. Symmetry: $d(p, q) = d(q, p)$
  2. Definiteness: $d(p, q) = 0 \Leftrightarrow p = q$
  3. Triangle inequality: $d(p, r) \leqslant d(p, q) + d(q, r)$

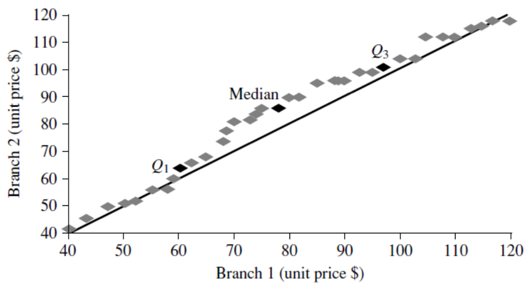
- Classes of measures:
    1. Distributive: same applied, e.g. count, min, sum, max
    2. Algebraic: can be computed, e.g. average
    3. Holistic: no bound on data needed, e.g. median, mode: value most often, rank: k-smallest
- Measures for central tendency
    1. Mean
    2. Midrange: (max - min)/2
    3. Median: apply also to ordinal; if even, median is usually defined to be the mean of the two middle values
    4. Mode: apply also to no ordering; most frequent data
- Measures for dispersion of data
    1. Quartiles: Q1, Q2 ...
    2. Inter-quartile range: Q3 - Q1
    3. Five number summary: min, Q1, median, Q3, max
    4. Boxplot: (plot outlier individually)
    5. Outlier: values that 1.5 * inter-quartile range below Q1 or above Q3
    6. Variance (scalable computation) and deviation:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2 = \left(\frac{1}{N}\sum_{i=1}^{N}x_i^2\right) - \bar{x}^2$$

       deviation measures spread about the mean and should be considered only when the mean is chosen as the measure of center.
- Visualization:
    1. Geometric: parallel coordinates, scatterplot
    2. Icon-based: Chernoff faces
    3. Pixel-oriented: recursive patterns
    4. Other: hierarchical, graph-based, hybrid
- Common ones:
    a. Box-plot: Minimum, Q1（四分位数第一个）, Median, Q3, Maximum
    b. Histogram
    c. Quantile plot: sort the data in ascending, 然后画图，横向均匀，递增 1/N，N 是数据个数

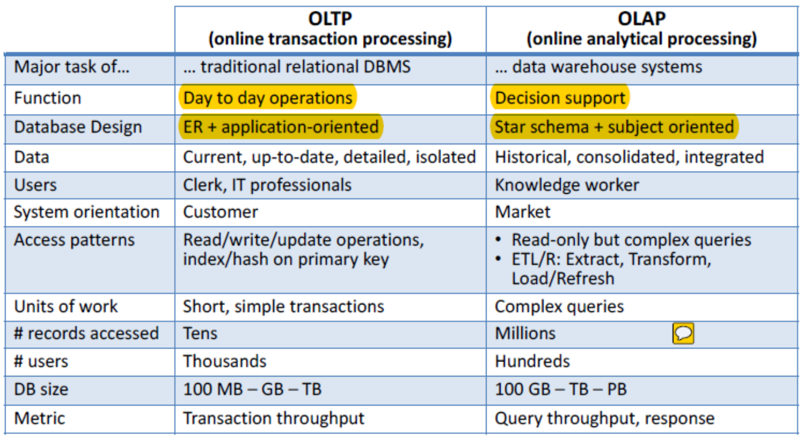
$$f_i = \frac{i - 0.5}{N}$$

    d. Quantile-quantile plot: 两个支部如果数量一样，直接画图，如果不一样，多的按照少的那个插值算一下，因为 q plot 中的比值都一样，所以可以省略掉



e. Scatter plot -> scatter plot matrix -> parallel coordinates(逐层递增展示高维的能力，但是 parallel 这个不能很好的展示大量数据 1), matrix and parallel 的 dimension reordering 是通过 quality metrics 来评价，可以 reduce clutter。
f. Parallel coordinates: min-max range in each axis, 不同 order 可以 focus on clusters or correlation
g. Loess curve (local regression; on scatter plot): two parameters: a smoothing parameter (0-1, percentage of near points to be used), and the degree of the polynomials that are fitted by the regression
h. Pixel-oriented visualization: we can use any 2-D space-filling curve, 更高层 pattern 自己排布; each data for a pixel; have sub windows for each dimension; Inside a window, the data values are arranged in some global order shared by all windows.
i. Chernoff faces

- Data warehouse is a decision support database, maintained separately, support information processing, primary source for data mining
- Warehouse characteristics:
    1. Subject-oriented: around major subjects, simple and concise, focus on decision makers
    2. Integrated: integrating databases, flat files, online records; clean and integration before moving in
    3. Time-variant: historical data, has element of time
    4. Nonvolatile: initial and access, no update, physically separate storage
- Data warehouse vs operational DBMS

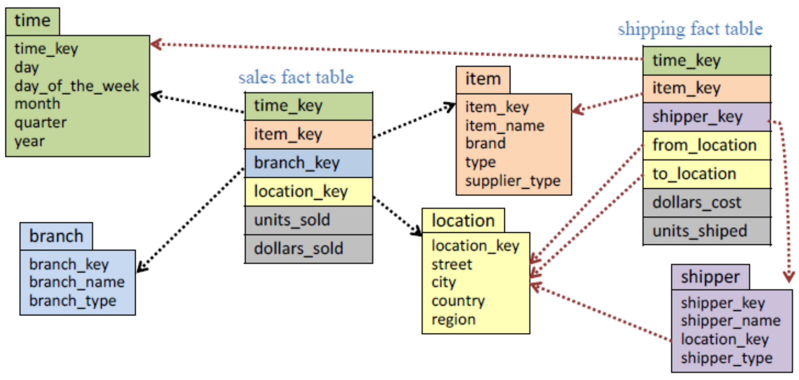


	OLTP **(online transaction processing)**	**OLAP** **(online analytical processing)**
Major task of…	… traditional relational DBMS	… data warehouse systems
Function	Day to day operations	Decision support
Database Design	ER + application-oriented	Star schema + subject oriented
Data	Current, up-to-date, detailed, isolated	Historical, consolidated, integrated
Users	Clerk, IT professionals	Knowledge worker
System orientation	Customer	Market
Access patterns	Read/write/update operations, index/hash on primary key	• Read-only but complex queries • ETL/R: Extract, Transform, Load/Refresh
Units of work	Short, simple transactions	Complex queries
# records accessed	Tens	Millions
# users	Thousands	Hundreds
DB size	100 MB – GB – TB	100 GB – TB – PB
Metric	Transaction throughput	Query throughput, response

- Modeling schemes for data warehouse
    1. Star schema

2. Snowflake schema
3. Fact constellation (星座)

- Star schema:
  A fact table (dimensions i.e. keys, and measures i.e. dependent attributes) in the middle connected to a set of dimension tables; each dimension is represented by only one table, and each table contains a set of attributes
- Snowflake schema: some dimensional hierarchy normalized into a set of smaller dimension tables
- Fact constellations: multiple fact tables



- Data warehouse vs data mart: data warehouse is enterprise-wise, has larger span, often use fact constellation schema; data mart is a department subset of the data warehouse, often uses star or snowflake schema.
- OLAP operations:
  1. Roll-up: either by climbing up a hierarchy (city to country) or by dimension reduction (remove time 那个方向，就剩下一个总和)
  2. Drill-down: stepping down a hierarchy or add dimension
  3. Slice: a selection on one dimension
  4. Dice: a selection on two or more dimensions
  5. Pivot (rotate):

Quarter 1			**Quarter 2**			**Quarter 3**		
TV	PC	VCR	TV	PC	VCR	TV	PC	VCR
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…

$\Downarrow$ **Pivot (rotate)**

TV			**PC**			**VCR**		
Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3
…	…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…	…

Rotate the axes in a 3-D cube, or transforming a 3-D cube into a series of 2-D planes.

  6. Drill-across: executes queries involving more than one fact table
  7. Drill-through: uses relational SQL facilities to drill through the bottom level of a data cube down to its back-end relational tables

- A starnet query model consists of radial lines (a concept hierarchy for a dimension), each abstraction level in a hierarchy is called a footprint (showed as small dots)

Data warehousing 2

- Data reduction:
  1. Numerosity reduction
  2. Dimensionality reduction
  3. Discretization (i.e. quantization) and concept hierarchy
  4. generalizaition
- Numerosity reduction:
  1. Parametric methods: estimate model parameters, store them, discard data
  2. Non-parametric methods: sampling, histograms, cluster (the cluster representations of the data are used to replace the actual data.)
- Dimensionality reduction, i.e. feature selection
- Heuristic feature selection methods:
  1. Best single feature under feature independence assumption: significance test
  2. Step-wise best feature selection: pick the best conditioned to chosen features
  3. Step-wise feature elimination: repeatedly eliminate worst
  4. Best combined feature selection and elimination
  5. Optimal branch and bound: feature elimination and backtracking
- Transform feature space basis:
  1. Data dependent: PCA
  2. Data independent: random projections
- PCA:
  1. Computation of centered data matrix: adapt value range and subtract mean
  2. Compute covariance matrix
  3. Eigen decomposition (注意，矩阵转置然后一乘，就是 cov)

$$\frac{1}{N}\tilde{A}^T \cdot \tilde{A} = \overbrace{\boxed{Cov = \underset{\substack{\text{eigenvector matrix} \\ (V \cdot V^T = V^T \cdot V = Id)}}{\boxed{V}} \cdot D \cdot V^T}}^{\text{diagonalization}}, \quad D = \begin{pmatrix} \overset{\text{eigenvalues}}{\boxed{v_1}} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & v_d \end{pmatrix}$$

  4. Transformation of data (normalized 的数据右乘矩阵即可)

$$Cov = V \cdot D \cdot V^T \quad \rightarrow \quad \tilde{A} \cdot V$$

  5. Truncation (截断) of transformed data (就是从前往后选取几列就成)
- PCA properties:
  1. High complexity O(nd^2 + d^3) (covariance matrix + diagonalization) for n points and d dimensions
  2. Optimal reconstruction w.r.t. mean-square error
  3. Lost 'semantics', i.e. hard to interpret about original dimensions

- Random projection:
  Randomly choose k vectors in d dimensions (k is old, d is new size)
  No need orthogonal
  But normalized in O(dk)
  Application complexity O(ndk) for n data
- Achlioptas's simple approach:
  R,R' ∈ R $^{k\times d}$

$$R[i,j] = \begin{cases} +1 & with\ probability\ 1/2 \\ -1 & with\ probability\ 1/2 \end{cases}$$

$$R'[i,j] = \sqrt{3} \cdot \begin{cases} +1 & with\ probability\ 1/6 \\ 0 & with\ probability\ 2/3 \\ -1 & with\ probability\ 1/6 \end{cases}$$

  Application on each data, each row in R or R' has length to 1

$$g(x) = \frac{1}{\sqrt{r}} \cdot x \cdot R$$

$$g'(x) = \frac{1}{\sqrt{r}} \cdot x \cdot R'$$

- Concept hierarchies: from low level concepts to higher level
- Discretization (quantization): divide range into intervals
  1. Binning
  2. Clustering (can be hierarchical clustering)
  3. Entropy-based discretization
- Binning: divide data into bins and store a representative (average, median…) for each bin
  1. Equi-width: easy; but outliers dominate (因为 bin 个数事先确定)
  2. Equi-height: good data scaling, more robust to outliers; but bad when a single value's frequency is high (intervals complete and disjoint)
  3. V-optimal: minimize variance within bin;注意 bound 不是真实数据取的值，而是范围，也就是要 complete，虽然总体的头尾确实是 min, max 附近的数

$$\sum_{i=1}^{N} \sum_{j=lb_i}^{ub_i} (f(j) - avg_i)^2$$

where:
N = number of buckets
lb_i, ub_i = lower and upper bounds of *i-th* bucket
$f(j)$ = number of occurrence of the value *j*
avg_i = average of frequencies occurring in *i-th* bucket

stogram:

- Entropy-based discretization: partition into two (at a time), minimize the entropy

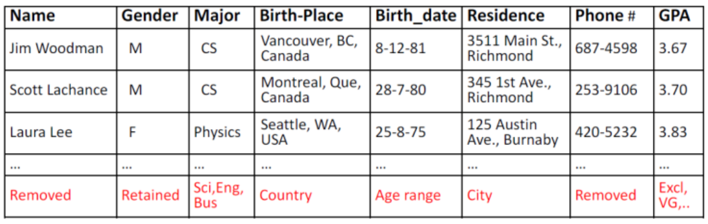
$$E(S,T) = \frac{|S_1|}{|S|} Ent(S_1) + \frac{|S_2|}{|S|} Ent(S_2) \qquad Ent(S) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

  实际上，需要两个变量：也就是要被分的，和参照的 class
  http://kevinmeurer.com/a-simple-guide-to-entropy-based-discretization/
  m 就是参照的 class 的个数, p 就是那个 bin 内部各 class 的概率, log2 是因为 bit
- Generalization:
  Data cube (OLAP) approach; manual
  Attribute-oriented induction approach; automated
- OLAP approach

Pros: can be performed on data cube by roll-up and drill-down

Cons: handle simple nonnumeric or simple aggregated numeric values; cannot tell what level should reach

Example:



Name	Gender	Major	Birth-Place	Birth_date	Residence	Phone #	GPA
Jim Woodman	M	CS	Vancouver, BC, Canada	8-12-81	3511 Main St., Richmond	687-4598	3.67
Scott Lachance	M	CS	Montreal, Que, Canada	28-7-80	345 1st Ave., Richmond	253-9106	3.70
Laura Lee	F	Physics	Seattle, WA, USA	25-8-75	125 Austin Ave., Burnaby	420-5232	3.83
...	...	...	...	...	...	...	...
Removed	Retained	Sci,Eng, Bus	Country	Age range	City	Removed	Excl, VG,..

Then we can sum and count

- Attribute-oriented induction (AOI)
  By attribute removal or attribute generalization
  Put threshold on distinct values of an attribute or control the size of final relation
- Data cleaning
  1. Fill in missing values
  2. Identify outliers and smooth out noisy data
  3. Correct inconsistent data
- Missing data
  1. Ignore the tuple
  2. Fill in manually
  3. Use a default constant
  4. Use mean
  5. Use mean of same class
  6. Use most probable value (e.g. by decision tree)
- Noisy data and inconsistent data
  Binning (then smooth)
  Clustering
  Combined computer(detect) and human(correct)
  Regression
- Data cleaning is not always necessary, some data mining can deal with uncertain data

Frequent pattern mining

- Frequent itemset notation:
  Items I = {i1, i2, i3, …}: set of all possible items
  k-itemset itemset of length k
  support number of transactions in database has this itemset
  frequent itemset support(X) ≥ minSup
- Apriori principle:
  1. Any non-empty subset of a frequent itemset is frequent, too!
  2. Any superset of a non-frequent itemset is non-frequent, too!
- Apriori algorithm:

- Variables
  - C_k: candidate itemsets of size k
  - L_k: frequent itemsets of size k

```
$L_1$ = {frequent items}
for ($k$ = 1; $L_k$ != ∅; $k$++) do begin
    join $L_k$ with itself to produce $C_{k+1}$
    discard ($k$+1)-itemsets from $C_{k+1}$ that contain non-frequent
        $k$-itemsets as subsets
    $C_{k+1}$ = candidates generated from $L_k$

    for each transaction $t$ in database do
        Increment the count of all candidates in $C_{k+1}$
            that are contained in $t$
    $L_{k+1}$ = candidates in $C_{k+1}$ with $minSup$
return $\cup_k L_k$
```

Can also be viewed as:

1. Join: p and q are joined if they share the same first k-1 items (sort in some order, only if firsts are the same)
2. Prune: delete k-subset that is not frequent
3. Scan: count support

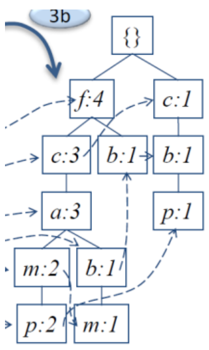
- Hash-tree
  1. Leaf nodes of hash-tree contain lists of itemset and their support (i.e., counts)
  2. Interior nodes contain hash tables

  Construction: search and insert, if overflow, transform leaf node to internal node

  Counting: 使用时只按照包含的子树来查看，(1,3,7,9,12)找 3-itemset，那么 root 只看前三位，之后也是只看能包含到的位置

- FP-tree

  Construction:
  1. Scan DB, find frequent 1-itemsets
  2. Retain only frequent ones in transaction and order transaction in descending order
  3. Scan DB again, construct FP-tree



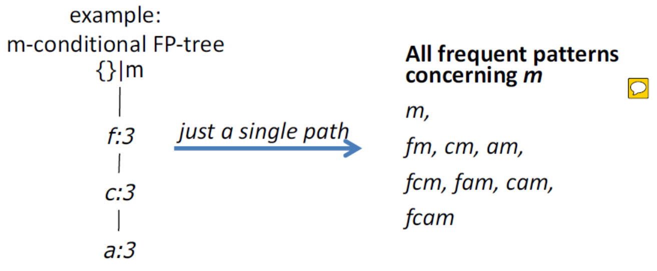
  Mine FP-tree:
  1. Construct conditional pattern base for each node in the FP-tree
  2. Construct conditional FP-tree for each conditional pattern base (会去掉不够 frequent 的，就是正常的构造 tree，不过记得头部加上 condition)
  3. Recursively mine, if single path or empty, enumerate all

conditional pattern base:

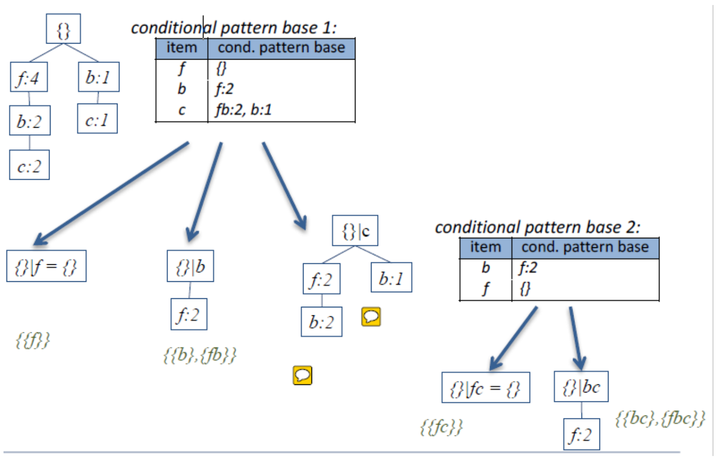
item	cond. pattern base
f	*{}*
c	*f:3, {}*
a	*fc:3*
b	*fca:1, f:1, c:1*
m	*fca:2, fcab:1*
p	*fcam:2, cb:1*

注意数字是自己 node 中的数

Single path enumerate (总是包含 condition，别的任选):



说白了，就是不断 tree base 转化，因为每次 tree 都是单个 base 的，所以简化



Pros:

1. No candidate generation
2. Compact data structure
3. Eliminate DB scan

- Closed frequent itemset: no proper super-itemset with the same support (complete information)
- Maximal frequent itemset: no proper super-itemset with support ≥ minSup (not complete information)
- Association rule: An association rule is an implication of the form $X \Rightarrow Y$ where $X, Y \subseteq I$ are two itemsets with $X \cap Y = \emptyset$.
  注意下式 support 尾巴并不对，除以|D|不能省略，记住两个都小于 1，X 并 Y 更大，更少见

$$support(X \Rightarrow Y) = P(X \cup Y) = \frac{|\{T \in D | X \cup Y \subseteq T\}|}{|D|} = support(X \cup Y)$$

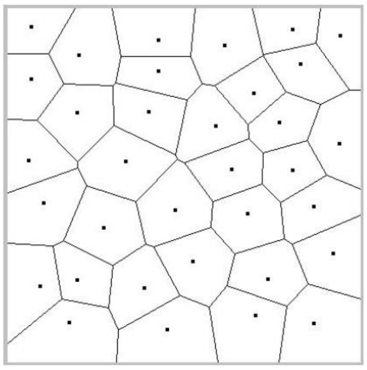
$$confidence(X \Rightarrow Y) = P(Y|X) = \frac{|\{T \in D | X \cup Y \subseteq T\}|}{|\{T \in D | X \subseteq T\}|} = \frac{support(X \cup Y)}{support(X)}$$

Note that $X \cup Y$ is the join of the two, and support become smaller

- Generate association rules:
    1. Find frequent itemsets
    2. For each itemset, divide it into 2 subsets, check minConf ($2^{|X|}$ -2 possibilities, include both way a => b and b => a)
- Find rules that are unexpected and actionable
- Association rules can be misleading, add correlation A=>B [supp, conf, corr]

Clustering 1

- Major clustering approaches:
    1. Partitioning (k partitions)
    2. Probabilistic model-based clustering (EM)
    3. Density-based
    4. Hierarchical
    5. other
- Voronoi model for convex cluster regions (都是垂直平分):



- K-Means clustering

  Objective: Minimize within-cluster squared distances between mean and elements

  Measure of compactness (sum of squared error):

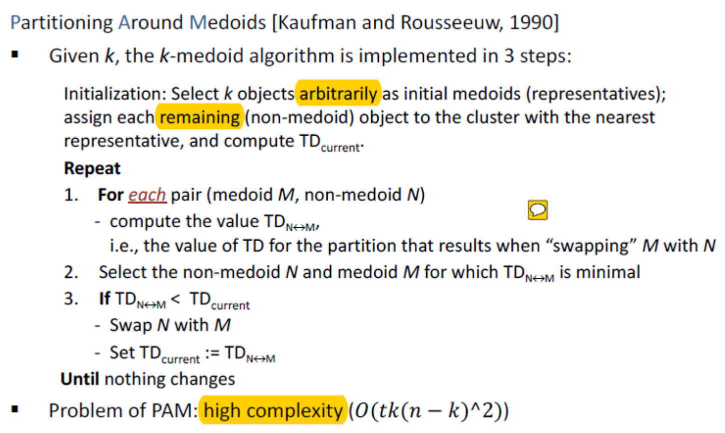
$$SSE(\mathcal{C}) = \sum_{C_j \in \mathcal{C}} SSE(C_j) = \sum_{p \in DB} dist(p, \mu_{C(p)})^2$$

  (above 1st by sum over clusters, 2nd by sum over all points)

  Computation: NP-hard, so use heuristic algorithms
- K-Means algorithm
    1. Initialization
    2. Repeat until no change:
        a. Assign object to nearest representative
        b. Compute centroids
- K-Medoid

  Use the absolute error (total distance)

PAM algorithm:



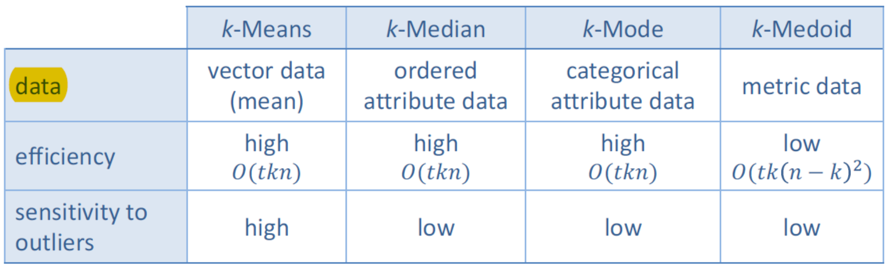
Partitioning Around Medoids [Kaufman and Rousseeuw, 1990]

- Given k, the k-medoid algorithm is implemented in 3 steps:

```
Initialization: Select $k$ objects arbitrarily as initial medoids (representatives);
assign each remaining (non-medoid) object to the cluster with the nearest
representative, and compute $TD_{current}$.
Repeat
1. For each pair (medoid $M$, non-medoid $N$)
   - compute the value $TD_{N \leftrightarrow M}$,
     i.e., the value of TD for the partition that results when "swapping" $M$ with $N$
2. Select the non-medoid $N$ and medoid $M$ for which $TD_{N \leftrightarrow M}$ is minimal
3. If $TD_{N \leftrightarrow M} < TD_{current}$
   - Swap $N$ with $M$
   - Set $TD_{current} := TD_{N \leftrightarrow M}$
Until nothing changes
```

- Problem of PAM: high complexity ($O(tk(n-k)$^2))

- K-Mode (First Approach)
  Distance is sum of Hamming distance
  Mode is not necessarily an instance
  Choose the one with highest frequency
  Actually, applying K-Means on categorical data
  Mode of a dataset might no be unique
- K-Median
  Median in each dimension independently, not an instance
- Comparison



	k-Means	*k*-Median	*k*-Mode	*k*-Medoid
data	vector data (mean)	ordered attribute data	categorical attribute data	metric data
efficiency	high $O(tkn)$	high $O(tkn)$	high $O(tkn)$	low $O(tk(n-k)^2)$
sensitivity to outliers	high	low	low	low

  tk(t-k)^2 is easy, t iterations, k(t-k) possible swaps, (t-k) distance need to be recalculated
- Pros:
  Easy implementation
- Cons:
  Need to specify k
  Forced to convex space partitions
  Depend on initial partition, local optimum
- Initialization of k clusters
  a. Choose samples, cluster them, use the centers
  b. Choose m sets of samples, cluster each of them to get k centers, so k*m centers, then cluster these k*m points for m times, each time with initialization of one set of centers, find the best result centers
- Choice of k
  Choose by clustering from 2 to n-1
  Use silhouette-coefficient
- Silhouette coefficient is not monotonic over k

$$a(o) = \frac{1}{|C(o)|} \sum_{p \in C(o)} dist(o,p)$$

(note that not consider o itself, and divide by |C(o)|-1)

$$b(o) = \min_{C_i \neq C(o)} \left(\frac{1}{|C_i|} \sum_{p \in C_i} dist(o,p) \right)$$

$$s(o) = \begin{cases} 0 & if\ a(o) = 0, e.g. |C_i| = 1 \\ \dfrac{b(o) - a(o)}{\max\{a(o), b(o)\}} & else \end{cases}$$

单个点如上，对一个 cluster 就是求和算平均，对所有 clusters 也是，注意设置为 0 是避免全都是一个点的 cluster

Range -1 to 1

$$silh(C_i) = \frac{1}{|C_i|} \sum_{o \in C_i} s(o) \qquad silh(\mathcal{C}) = \frac{1}{|D|} \sum_{o \in D} s(o)$$

Explanation of silhouette coefficient:

Let $a(o) \neq 0$， s(o) ~1: good; ~0: in-between; ~-1: bad

s_C of a clustering

1. $0.7 < s_C \le 1.0$ strong structure
2. $0.5 < s_C \le 0.7$ medium structure
3. $0.25 < s_C \le 0.5$ weak structure
4. $s_C \le 0.25$ no structure

Clustering 2

- Probabilistic model-based clusters: EM
  1. Define clusters as probability distributions
  2. Improve the parameters of each distribution
- EM not restricted to Gaussian, but as example here

$$p(x|\underbrace{\mu}_{\text{mean} \in \mathbb{R}}, \underbrace{\sigma^2}_{\text{variance} \in \mathbb{R}}) = \mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{1}{2\sigma^2}\cdot(x-\mu)^2}$$

-

$$p(\boldsymbol{x}|\underbrace{\boldsymbol{\mu}}_{\text{mean vector} \in \mathbb{R}^d}, \underbrace{\boldsymbol{\Sigma}}_{\text{covariance matrix} \in \mathbb{R}^{d\times d}}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d|\boldsymbol{\Sigma}|}} \cdot e^{-\frac{1}{2}\cdot(\boldsymbol{x}-\boldsymbol{\mu})^T\cdot(\boldsymbol{\Sigma})^{-1}\cdot(\boldsymbol{x}-\boldsymbol{\mu})}$$

-
- Mixture model

$$p(\boldsymbol{x}|\Theta) = \sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

x is a d-dimensional vector, or a data

π_k sum up to 1, each in [0,1]

$$p(\mathbf{X}|\theta) = \prod_{n=1}^{N} p(x_n|\theta)$$ (for N data points)

Maximize log-likelihood:

$$\Theta_{ML} = \arg\max_{\Theta}\{\log p(\mathbf{X}|\Theta)\}$$

$$\log p(\mathbf{X}|\theta) = \log \prod_{n=1}^{N}\sum_{k=1}^{K} \pi_k \cdot \mathrm{p}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \cdot \mathrm{p}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

上面对 μ_j 求偏导，令为 0，就得到 EM 中的式子

- No closed-form solution to Gaussian, mutual dependency, so apply iteratively
- EM algorithm

1. Initialize means $\boldsymbol{\mu}_j$, covariances $\boldsymbol{\Sigma}_j$, and mixing coefficients π_j and evaluate the initial log likelihood.
2. <u>**E-step**:</u> Evaluate the responsibilities using the current parameter values:

$$\gamma_j^{new}(\boldsymbol{x}_n) = \frac{\pi_j \cdot \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^{K} \pi_k \cdot \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

3. <u>**M-step**:</u> Re-estimate the parameters using the current responsibilities:

$$\boldsymbol{\mu}_j^{new} = \frac{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)\, \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)}$$

$$\boldsymbol{\Sigma}_j^{new} = \frac{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)(\boldsymbol{x}_n - \boldsymbol{\mu}_j^{new})(\boldsymbol{x}_n - \boldsymbol{\mu}_j^{new})^T}{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)}$$

$$\pi_j^{new} = \frac{\sum_{n=1}^{N} \gamma_j^{new}(\boldsymbol{x}_n)}{\sum_{k=1}^{K}\sum_{n=1}^{N} \gamma_k^{new}(\boldsymbol{x}_n)}$$

4. Evaluate the new log likelihood $\log p(\mathbf{X}|\Theta^{\text{new}})$ and check for convergence of parameters of log likelihood ($|\log p(\mathbf{X}|\Theta^{\text{new}})$ - $\log p(\mathbf{X}|\Theta)| \leq \epsilon$).
If the convergence criterion is not satisfied, set $\Theta = \Theta^{\text{new}}$ and go to step 2.

- EM apply to partitioning:

$$\text{Cluster}(\boldsymbol{x}_n) = argmax_{k\in\{1,\dots,K\}}\{\gamma_k(\boldsymbol{x}_n)\}$$

- Properties:
  Better than K-Means on clusters of varying size and differing variances
  Local maximum
  O(tnk); however t tends to be high
- Initialization for centers:
  1. Multiple random starts
  2. Use K-Means centers
- Choose k
  - Silhouette coefficient only works for partitioning
  - Maximum likelihood estimation is non-decreasing on k

  Deterministic (add a function increase on k): $\log p(\mathbf{X}|\Theta_K) + \mathcal{P}(K)$

  Stochastic: MCMC

Clustering 3

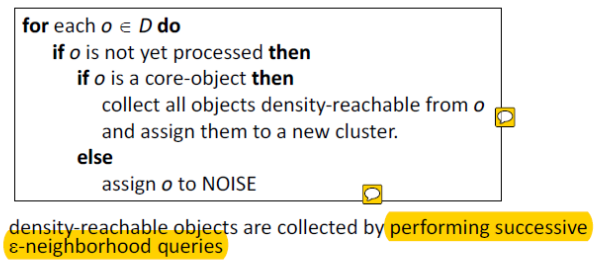
- Partitioning and hierarchical methods are designed to find spherical-shaped clusters, use density-based for arbitrary shape
- 1. Core object is those has at least MinPts within circle ϵ-radius, including itself: $|N_\epsilon(q)| \geq$ MinPts

  2. p directly density-reachable from q: q is core, $p \in N_\epsilon(q)$

  3. p density-reachable from q: q is core, p within a chain of directly density-reachable; not symmetric

  4. p density-connected q: symmetric, can be both non-core, general than density-reachable; c(o1,o2) and c(o2,o3) => c(o1,o3); DBSCAN use this
- Density-based cluster:
  1. maximality: p in S and q is density-reachable from p, then q in S
  2. connectivity: each density-connected to all others in S

  Density-based clustering:
  1. Density-based clusters and noise
- DBSCAN algorithm:



```
for each o ∈ D do
    if o is not yet processed then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

– density-reachable objects are collected by performing successive ε-neighborhood queries

每次一网打尽整个 cluster，assign noise 是暂时的，可以被 assign 到新 cluster
书中一个点不会被放到多个 cluster，ppt 没有明白表示

- K-distance(p): distance from p to its k-nearest neighbor (including itself)
- K-distance plot: k-distance of all objects, sorted in descending
- Setting of ϵ and MinPts:
  a. Fix MinPts (default: 2*dimension - 1)
  b. User selects ϵ from MinPts-distance plot (by looking for dramatically drop)
- Pros:
  1. Clusters of arbitrary shape (not convex)
  2. Number of cluster no need to decide
  3. Separate noise
  4. Use spatial index structures O(n log n), without $O(n^2)$
  5. N_ϵ-query: O(n)
- Cons:
  1. Difficult to determine parameters (比如 distance plot 整个平滑，所以就有了 hierarchical clustering)
  2. Sensitive to parameter setting
- Hierarchical clustering

  Dendrogram(系统树)

  Can be agglomerative or divisive

Leaf is a single object

Internal node is the union of two sub-trees

Height of internal node represents distance of two children

- Agglomerative hierarchical clustering

  1. Initially, each object forms its own cluster
  2. Consider all pairwise distances between the initial clusters (objects)
  3. Merge the closest pair (A, B) in the set of the current clusters into a new cluster C = A ∪ B
  4. Remove A and B from the set of current clusters; insert C into the set of current clusters
  5. If the set of current clusters contains only C (i.e., if C represents all objects from the database): STOP
  6. Else: determine the distance between the new cluster C and all other clusters in the set of current clusters; go to step 3.

  就是挑最相近的两个 cluster 合，然后替换成新的，计算新的到别的 cluster 的距离

- Distance for clusters

  Single-Link: $dist\_sl(X,Y) = \min_{x \in X, y \in Y} dist(x,y)$

  Complete-Link: $dist\_cl(X,Y) = \max_{x \in X, y \in Y} dist(x,y)$

  Average-Link: $dist\_al(X,Y) = \frac{1}{|X| \cdot |Y|} \cdot \sum_{x \in X, y \in Y} dist(x,y)$

- Divisive hierarchical clustering (DIANA)

  Select the cluster C with largest diameter for splitting

  Search the most disparate observation o in C (highest average dissimilarity)

  – $SplinterGroup := \{o\}$

  – Iteratively assign the $o' \in C \backslash SplinterGroup$ with the highest $D(o') > 0$ to the splinter group until for all $o' \in C \backslash SplinterGroup$: $D(o') \leq 0$

  $D(o') = \sum_{o_j \in C \backslash SplinterGroup} \frac{d(o', o_j)}{|C \backslash SplinterGroup|} - \sum_{o_i \in SplinterGroup} \frac{d(o', o_i)}{|SplinterGroup|}$

  挑最与众不同的点，然后看 cluster 中和它平均距离比到补集近的点，排序挑下一个最刺头的过去，直到剩下的到 splinter 的距离比到补集的远

- Agglomerative vs divisive

  Divisive is conceptually more complex

  Agglomerative on local patterns

  Divisive use complete information

- Core-distance:

  If q is core object w.r.t. ϵ and MinPts, then smallest distance such that o is a core object

  Else undefined

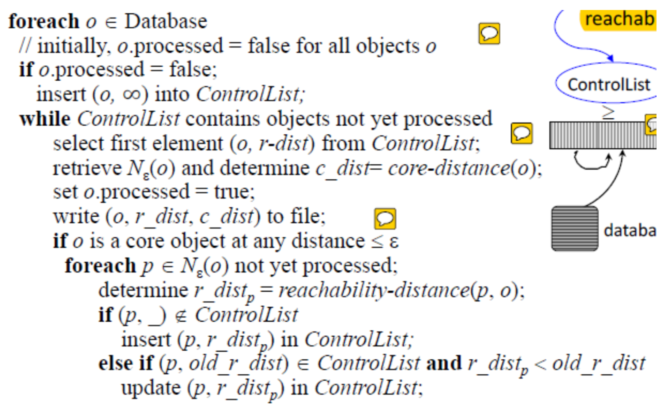
- Reachability-distance: minimum radius value that makes p ***directly*** density-reachable from q

  If q is core object w.r.t. ϵ and MinPts, then max{core-distance(q), dist(p, q)}

  Else ∞

- OPTICS:

  the core distance is also exported, but this is not required for further processing controlList is sorted with reachability-distance in ascending



```
foreach $o \in$ Database
  // initially, $o$.processed = false for all objects $o$
  if $o$.processed = false;
    insert $(o, \infty)$ into ControlList;
  while ControlList contains objects not yet processed
      select first element $(o, r\text{-}dist)$ from ControlList;
      retrieve $N_\varepsilon(o)$ and determine $c\_dist = core\text{-}distance(o)$;
      set $o$.processed = true;
      write $(o, r\_dist, c\_dist)$ to file;
      if $o$ is a core object at any distance $\le \varepsilon$
        foreach $p \in N_\varepsilon(o)$ not yet processed;
            determine $r\_dist_p = reachability\text{-}distance(p, o)$;
            if $(p, \_) \notin$ ControlList
               insert $(p, r\_dist_p)$ in ControlList;
            else if $(p, old\_r\_dist) \in$ ControlList and $r\_dist_p < old\_r\_dist$
               update $(p, r\_dist_p)$ in ControlList;
```

可以看成波的传播

ControlList 可能会剩下，只看有没有还没 process 的

Plot in the order of file, against reachability-distance

图里对于 undefined, ∞就是画满 y 轴

Performance 和 DBSCAN 类似， O(n log n) with spatial indexed support, O(n^2) without

Insensitive to parameter settings, need MinPts, and ϵ to be large

- Hierarchical clustering discussion

  Pros:

  1. No number of clusters
  2. No or robust parameters
  3. Hierarchy of clusters

  Cons:

  1. Runtime for agglomerative O(n^2 log n)总共要 n 轮，每次都排序挑最小, divisive O(2^n), OPTICS is better
  2. User has to choose final clustering

- Evaluation of clustering result

  Expert

  Internal: like silhouette or sum of square errors (或者说 variance)

$$SSE(\mathcal{C}) = \frac{1}{|DB|}\sum_{C_i \in \mathcal{C}} \sum_{p \in C_i} dist\big(p, \mu(C_i)\big)^2$$

  External: need true clusters

- External measures (我的是 C,真相是 G)

  Recall: $rec(C_i, G_j) = \frac{|C_i \cap G_j|}{|G_j|}$ Precision: $prec(C_i, G_j) = \frac{|C_i \cap G_j|}{|C_i|}$

  F-Measure: $F(C_i, |G_j) = \frac{2 * rec(C_i, G_j) * prec(C_i, G_j)}{rec(C_i, G_j) + prec(C_i, G_j)}$

  Purity (P): $P(\mathcal{C}, \mathcal{G}) = \sum_{C_i \in \mathcal{C}} \frac{|C_i|}{|DB|} pur(C_i, \mathcal{G})$ $pur(C_i, \mathcal{G}) = \max_{G_j \in \mathcal{G}} prec(C_i, G_j)$

  Recall 就是我取得了多少， precision 就是我自己命中多少

  F 是合并上两个

  Purity 是每个 cluster 看最高可能的命中，然后平均

  Random index:

  a 是所有同时在 C 某个 cluster 里，也同时在 G 某 cluster 里的 pair b 是处于 C 中不同 cluster，也处于 G 中不同 cluster 的 pair

(n 2)是所有可能的对, n(n-1)/2

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}}$$

或者用 mutual information (i.e. information gain): H 就是熵

$$I(\mathcal{C},\mathcal{G}) = H(\mathcal{C}) - H(\mathcal{C}|\mathcal{G}) = H(\mathcal{G}) - H(\mathcal{G}|\mathcal{C})$$

Normalized information gain:

$$NMI(\mathcal{C},\mathcal{G}) = \frac{I(\mathcal{C},\mathcal{G})}{\sqrt{H(\mathcal{C})H(\mathcal{G})}}$$

Classification 1

- Classification vs prediction
  Classification on categorical class labels
  Prediction on continuous-values functions (usually with regression)
- Nearest neighbor classifier
  1. NN classifier: 1-neighbor
  2. k-NN
  3. Weighted k-NN: use 3 kinds of weight
  4. Mean-based NN: training 时计算每个 class 的 mean，分类就看离哪个 mean 近
- 3 weights
  Standard: equal
  Distance: 1/(distance^2)
  a-priori: class frequency, 越少见，越金贵
- Pros:
  1. Can be applied to non-vector data
  2. Only requires similarity function (i.e. metric space)
  3. High accuracy
  4. Easy to add new training data
  5. Robust to noisy data

  Cons:
  1. Implementation can be inefficient: should create index structure in training phase
  2. No explicit knowledge
  3. Curse of dimensionality
- Instance-based learning (lazy evaluation): k-nearest neighbor
  Eager evaluation: decision tree, Bayes classifier

Classification 2

Decision tree:

- Tree construction
  Greedy algorithm

Condition to stop:

Belong to same class

No remaining attributes – majority voting

No samples left (nothing has this as value, use majority of all examples this round)

Algorithm:

```
ID3(*Examples*, *ClassLabels*, *Attributes*)
Create a *Root* node for the tree;
If all *Examples* have the same *ClassLabel*, return *Root* with corresponding label;
If *Attributes*=$\emptyset$, return *Root* with label = most common value of *ClassLabels* in *Examples*;
Else
  $A$=the 'best' decision attribute for next node
  Assign A as decision attribute for *Root*
  For each possible value $v_i$ of *A*:
    Generate branch corresponding to test $A = v_i$;
    $Examples_{v_i}$= examples that have value $v_i$ for $A$;
    If $Examples_{v_i} = \emptyset$, add leaf node with label = most common value of *ClassLabels* in *Examples*;
    Else add subtree ID3($Examples_{v_i}$, *ClassLabels*, *Attributes*\{*A*});
```

每一次挑最好的区分，然后分

- Split strategies:

  Need to be disjoint and complete

- Information gain:

$$entropy(T) = -\sum_{i=1}^{k} p_i \cdot \log_2 p_i$$

for *k* classes c_i with frequencies p_i

注意是 2 为底的

$$information\ gain(T, A) = entropy(T) - \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot entropy(T_i)$$

- Gini index

$$gini(T) = 1 - \sum_{i=1}^{k} p_i^2$$

for *k* classes c_i with frequencies p_i

$$gini_A(T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot gini(T_i)$$

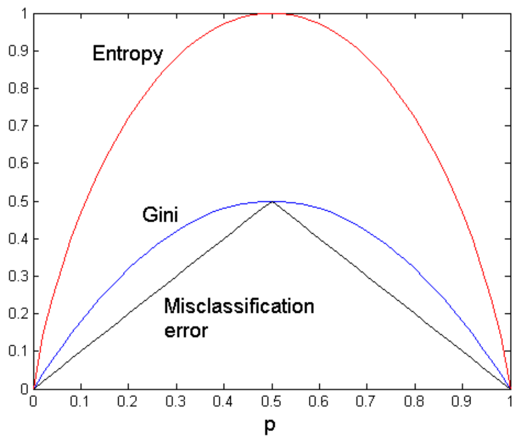
- Misclassification error

$$Error(T) = 1 - \max_{c_i} p_i$$

for *k* classes c_i with frequencies p_i

$$Error_A(T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot Error(T_i)$$

- Summary: all of the three need to be the small the better



2-class case

- Types of split

Categorical: 1. equal 2. subset

Numerical: 1. cut with one < (can order test samples and consider every mean of adjacent two samples) 2. Intervals

- Tree pruning

1. Prepruning:

Halt before goodness measure falling below a threshold

Like using minimum support: minimum number of data objects a leaf node contain

Minimum confidence: minimum fraction of the majority class in a leaf node

But prepruning is difficult to set appropriate thresholds and has less information for decision than postpruning, so worse quality

2. Postpruning

Reduced-error pruning:算法 1

- Decompose classified data into training set and test set
- Create a decision tree *E* for the training set
- Prune *E* by using the test set *T*
  - determine an interior node v of *E* whose pruning reduces the number of misclassified data points on *T* the most (i.e., replace the subtree S of node v by a leaf. Determine the value of the leaf by majority voting)
  - prune
  - finish if no such interior node exists

就是每个节点，看 prune 它成 majority vote 会不会变好，最好的那个就变，直到没有能变好的为止

Minimal cost complexity pruning:算法 2

Notation: size |E| is the number of leaf nodes

Cost-complexity: measure both classification error F_T and size of tree:

$$CC_T(E,\alpha) = F_T(E) + \alpha \cdot |E|$$

α 是调节树展开程度的，对于树 Ee 和{e}单个 node 这两株树

for small values of α: $CC_T(E_e, \alpha) < CC_T(\{e\}, \alpha)$

for large values of α: $CC_T(E_e, \alpha) > CC_T(\{e\}, \alpha)$

α_{crit}: $CC_T(E_e, \alpha_{crit}) = CC_T(\{e\}, \alpha_{crit})$ 这里 α_{crit} 就是判断 tree weak 程度的标志越小越 weak，越先被 prune 掉

Algorithms:

- Start with a complete tree *E*
- Iteratively remove the weakest link from the current tree
- If there are several weakest links, remove them all in the same step
- Result: sequence of pruned trees
  - $E(\alpha_1) > E(\alpha_2) > \ldots > E(\alpha_m)$ where $\alpha_1 < \alpha_2 < \ldots < \alpha_m$
- Selection of the best E(α_i)
  - Estimate the classification error on the overall data set by an l-fold cross validation on the training set

本质就是随着 α 的升高，tree 越来越简化，我们得到 a sequence of 不同程度 pruned tree，从中挑最好的。这个不像之前，不用额外的 test set

- Classification rule from tree:

  IF forecast = 'rainy' AND wind = 'weak' THEN playing_tennis = 'yes'
- Pros:
  1. Convertible to rules, represent explicit knowledge, intuitive to users
  2. Can be categorical or continuous-valued
  3. Hierarchical and linear
  4. Fast learning speed
  5. Fast classification speed
  6. Accuracy is good

  Cons:
  1. Not stable, small changes of data, large change of tree

Bayesian classifier

- For an object o, see it possibility in class Cj

$$\underset{c_j \in C}{\operatorname{argmax}}\{p(c_j|o)\} = \underset{c_j \in C}{\operatorname{argmax}}\left\{\frac{p(o|c_j)\cdot p(c_j)}{p(o)}\right\} = \underset{c_j \in C}{\operatorname{argmax}}\{p(o|c_j)\cdot p(c_j)\}$$

Value of $p(o)$ is constant and does not change the result

p(cj)就是这个 class 在所有数据里面的比例

p(o|cj)这个才需要建立模型或者是 count frequency

- p(o|cj)的三种估计方法

  parametric method: single gaussian distribution

  non-parametric methods: kernel methods

  mixture models: mixture of Gaussian

  但是上面的多维高斯会 curse of dimensionality， naive Bayes 应运而生
- Naive Bayes classifier:

data has d dimensions, need to be conditionally independent, i.e.

$$p(o|c_j) = p(o_1, \dots, o_d|c_j) = \prod_{i=1}^{d} p(o_i|c_j)$$

那么接下来就是求 p(oi|Cj)

Categorical: relative frequency

Continuous: possibility distribution， 比如在某个 single Gaussian 中

$$p(o_i|C_j) = \frac{1}{\sqrt{2\pi}\sigma_{i,j}} e^{-\frac{1}{2}\left(\frac{o_i-\mu_{i,j}}{\sigma_{i,j}}\right)^2}$$

然后用 Bayes 来分别算就成了

Calculate the probabilities for both classes:

With:
$1 = p(high|q) + p(low|q)$

$$p(high|q) = \frac{p(q|high)\cdot p(high)}{p(q)}$$
$$= \frac{p(age = 60|high)\cdot p(car\ type = family|high)\cdot p(\max speed = 190|high)\cdot p(high)}{p(q)}$$
$$= \frac{N(27.67, 13.61|60)\cdot \frac{1}{3}\cdot N(222,36.49|190)\cdot \frac{3}{5}}{p(q)} = 15.32\%$$

这里应该没有算 p(q)，直接用和为 1 算两个比例

- Bayesian classifier (之前的是 naive 的，其实没必要，只要能算概率)
  比如全都变成相关 Gaussian

$$P(o|C_j) = \frac{1}{(2\pi)^{d/2}|\Sigma_j|^{1/2}} e^{-\frac{1}{2}(o-\mu_j)\Sigma_j^{-1}(o-\mu_j)^T}$$

  |Σ j| is determinant of Σ j

  Two limits:

  几个 class 概率都很低就不分类了

  概率相同，则不知如何
- Pros:
  a. High accuracy
  b. Adopted new training objects easily
  c. Can incorporate expert knowledge to the prior of P(Ci)

  Cons:
  a. Often conditional probabilities not available
  b. Curse of dimension
- Independence hypothesis

  Pros:

  Efficient computation

  Optimal classifiers

  Break limitation: Bayesian networks, decision trees
- Evaluation of classification

  Separate data to 2 sets, one for training, one for testing

If not possible, then

m-fold cross validation:

a. equally into m subsets
b. iteratively use m-1 to train, 1 to test
c. combine m accuracy values to an overall accuracy, combine m generated model to an overall model

leave-one-out:

a. special case of cross validation with m equal to size of data set |O| = N
b. so, left only 1 to be test, accuracy = right/|O|
c. good for nearest-neighbor classifiers

- Measurements

  Classification Accuracy:

$$G_T(K) = \frac{|\{o \in T, K(o) = C(o)\}|}{|T|}$$

Classification Error:

$$F_T(K) = \frac{|\{o \in T, K(o) \neq C(o)\}|}{|T|}$$

就是正误比例，和为 1

Resubstitution error:

$$F_{TR}(K) = \frac{|\{o \in TR, K(o) \neq C(o)\}|}{|TR|}$$

(true) classification error:

$$F_{TE}(K) = \frac{|\{o \in TE, K(o) \neq C(o)\}|}{|TE|}$$

就是在训练集和测试集山的错误率

- Overfitting

  Reasons:

  a. bad quality of training data
  b. different statistical characteristics of training data and test data

  to avoid it:

  a. removal of noisy and erroneous training data
  b. appropriate size of training set
  c. appropriate samples

- underfitting
- Summary:

	Decision Trees	k-NN classifier	Bayes classifier
Compactness	Compact if pruned	No model	Model dependent
Interpretability of model	Good	-	Model dependent
Explanation of decision	Good rules for decision known	Medium-Good decision object set known	Medium-Good probabilities of decision are given
Training time	Low-Medium	No training	Model dependent
Test time	Low	Low (index) Very high	Model dependent but often Low
Scalability	Good	Good (index) Bad	Model dependent but often Good
Robustness	Low	High	High
Data types	Categorical and vector	Arbitrary data (need distance function)	Arbitrary data (need probability distribution)
Model	Set of (axis parallel) hyperplanes	Model free	Statistical density distribution