Vocubulary

- an attribute of an object: variable, field, characteristic, feature, predictor
- collection of attributes describing an object: record, point, case, sample, entity, entry, instance

variable type

1. categorical

- Ordinal

> = 2 categories, with clear ordering eg. low, medium, high; grades, clothing size

- Nominal

> = 2 categories, no intrinsic ordering eg. ID number, eye color, zip code

2. Interval temperatire in Celsius intervals between values are equally spaced eg. $10000, $15000, $20000
3. Ratio eg. temperature in Kelvin, length, time, counts

- Notes: average of categorical variable not make sense average of educational experience -> nonsense (spacing not even) average required to be on interval

types of data sets

1. Record collection of record, each having fixed set of attribute
2. Data matrix objects with only numeric attributes can be represented by m*n matrix points in multi-dimensional space
3. Document data document becomes a "term" vector
4. Transaction Data special type of record data
5. Graph data
6. Ordered Data eg. genomic sequence data, spatial-temporal data

Association rule mining

find pattern, association, corelation usage:

- association, causality analysis
- sequential pattern
- pattern analysis
- classification
- cluster analysis
- data warehousing (iceberg cube) objective: find all rules that correlate 1 set of items with another set

support and confidence

find all rules X => Z with min confidence and support

- support: P(X & Z) statistical significant
- confidence: P(Z|X) rule's strength

variation

1. Boolean vs quantitative buys(x,"SQLServer") ^ buys(x, "DMBook") => buys(x, "PC")[1%, 75%] age(x, "30..39") ^ income(x, "42.48K") => ...
2. single vs multiple dimension
3. single vs multiple level
4. various extensions

Apriori principle

any subset of frequent item set must be frequent

Algorithm

1. apply apriori principle, iteratively find frequent itemsets with cardinality from 1->k
2. use itesets found, repeat previous step
3. if R satisfies min confidence, then R is strong association rule => output

L3 = {2,3,5} -> {2},{3},{5},{2,3},{2,5},{3,5}

improve Apriori's efficiency

1. hash based itemset counting
2. filter transaction not having frequent itemset
3. itemset at least frequent in 1 partition of DB
4. Sampling: mine on subset of data
5. add new iteset only when all subset frequent

other measuers: Interest

lift ratio = Interest() = P(A^B) / [P(A)P(B)] for A => B

Sequential ARM

goal

mine sequential association rules computing frequent sequences data consist of `<s1, s2, ... ,sn>` itemsets (items bought together) in time order

Customer sequence: all transactions of customer = seq ordered by increasing transaction time

steps

1. sort pahse: convert database to sequences
2. freq itemset phase: pick frequent itemsets
3. transformation phase: denote frequent itemsets as other symbols

4. sequence phase: find desired sequences using {ApioriAll, AprioriSome, DynamicSome}
5. maximal phase: find max sequences among set of frequent sequences

pick frequent itemsets

```
sequenced itemsets => frequent => denote:
<30,90>                  => <30,90>                  => <1,5>
<(10,20),30,(40,60,70)> => <30,{40,70,(40,70)}>    => <1,{2,3,4}>
<(30,50,70)>             => <{30,70}>                => <1,3>
<30,(40,70),90>          => <30,{40,70,(40,70)},90> => <1,{2,3,4}>
<90>                     => <90>                     => <5>

for min support=25%, obtain
<30,90>, <30,(40,70)>

denote as:
(30)    -> 1
(40)    -> 2
(70)    -> 3
(40,70) -> 4
(90)    -> 5
```

ApriorиAll

L1 -> L2 -> L3 -> L4

find max seqs: for(k=n; k>1; k--) forEach k-seq s_k delete items matching subsequences of s_k [so that item seqs will not overlap each other] get max supported seq in each k (except 1)

generating candidate

abc abd acd ace bcd ==> abc + abd --> abcd / abdc(X) acd + ace --> acde(X) / aced(X) ==> only {abcd} left

forming rules

```
<1 2 3 4> => (1 -> 2,3,4), (1,2 -> 3,4), (1,2,3 -> 4)
```

multi-level / generalised association rule

(1) milk => wheat bread (2) 2% milk => wheat bread

- rule (1) is ancestor of rule (2), (2) is redundant

multi-dimensional association

- single-dimension rule buys(X, "milk") => buys(X, "bread")

- multi-dimension age(X, "19-25") ^ occupation(X, "student") => buys(X, "coke") age(X, "19-25") ^ buys(X, "popcorn") => buys(X, "coke")

data distribution: bynamic discretization clustering: distance-based association

spatial association analysis:

- intersect, overlap ...
- left_of, under ...
- close_to, within_distance ...

how to apply association rule mining?

eg. for image data what is a transaction? 1 image what is an item? 1 image feature(color) what is a customer (seq)? sequence of fashion design images

==========

Classification

definition

classification: predict categorical class labels, model based on value in class labels prediction: model continuous valued functions (eg. predict unknown values)

model onstruciton: describe set of predetermined class model usage: classify future / unknown objects

supervied VS unsupervised

- supervision (classification) training data accomapnied by labels indicating class of observations
- unsupervised (clustering) class label unknown; given set of measurements, observations to establish classes

data preparation

data cleaning, feature selection, data transformation

evaluation of classification

accuracy, speed, scalability, interpretability, goodness of rule

decision tree

tree construction: all data at root -> partition recursively tree pruning: identify, remove branches

1 rule is created for each path from root to leaf, in form of `if-then` rule leaf node holds class prediction

algorithm

greedy algorithm: top-down recursive divide-and-conquer

- each step choose largest gain, do not consider optimal global gain

attribute are categorical / discrete set of continuous-valued selected on basis of heuristic(啟發法) / statistical measure

- stop partitioning: all samples for given node belong to same class no remaining attribute for further partitioning no samples left

attribute selection measure

information gain (discrete)

attributes assumed to be categorical (can modify to continuous)

p elements in class P, n elements in class N I(p,n) = -p/(p+n)*log(p/(p+n)) - n/(p+n)*log(n/(p+n)) <-- log of base 2

- count all p,n with this attribute as decision
- amount of information needed to decide if arbitrary example belong to P/N
- lowest when either class has 0 element, highest(1) when equal distribution

entropy of p_i of P and n_i of N: E(A) = sum(v; i=1){ (p_i + n_i)/(p + n) * I(p_i, n_i) }

- count by portion of sample it contain for this attr value
- higher purity of data less information to describe
- entropy = 0 when all data identical, entropy=1 when data half different

Gain(A) = I(p,n) - E(A)

- encoding info gained by branching on A
- the lower entropy (more purity) after branching, the more information gain

avoid overfit

overfit reason: curve try to reflect all anomalies

- prepruning: halt tree construction early, don't split node cause goodness measure below threshold
- postpruning: remove branches, get sequence of progressively pruned tree use different dataset from training data to decide which is best pruned tree

cross validation (eg. 10-fold cross validation) use minimum description length

1. min sample for node split
2. min sample for leaf
3. max depth of tree
4. max num of terminal nodes
5. max features to consider split

Gini index (continuous)

attibutes are continuous-valued several possible split values for each attribute may need other tools (eg. clustering) to split value

- info gain can have multiple nodes VS Gini can only produce 2 branches (T/F)

gini_index = 1- sum(P(A)^2 + P(B)^2) weighted_GI = num(A)/total*GI(A) + num(B)/total*GI(B)

choose classification with higher weighted gini index

CART tree

support both regression and classification

regression tree

similar: continuous outcome, ~ classification tree, many splits attempt difference:

- prediction = average of numerical target
- impurity = sum of squared deviation
- performance = RMSE

coefficient of variation = SD/average(x) SDR(T,X) = S(T) - S(T,X) => then select attribute with largest SDR

a good if-then partition encourage higher SD in population and slower SD in individual group

many classfication model have regression/prediction variant: Decision tree -> Regression tree Bayesian classifier -> Bayesian regressor ...

association-based classification

mine high support and high ocnfidence rule in form of "cond_set => y"

Instance-based classification

store training sample until new instance must be classified

- k-nearest neighbor
- case based reasoning

lazy evaluation VS eager evaluation

lazy evaluation: evaluate new record only when need to spend less time training need to store previous instance for comparison

eager evaluation: consider all records at first, has chosen global assumption spend more time training eg. Decision tree

Naive Bayesian classification

use Bayes theorem P(C|X) = P(X|C) x P(C) / P(X)

since P(X|C) cannot be calculated, assume attribute independence {P(x1|C), P(x2|C) ...} => P(X|C) = P(x1|C) x P(x2|C) ... x P(xn|C)

categorical: relateive frequency continuous: gaussian density function

example

there's class (p,n)

P(p) = 9/14 P(rain|p)=3/9, P(hot|p)=2/9, P(high|p)=3/9, P(false|p)=6/9 => X = {rain, hot, high, false} condition P(X|p) = P(rain|p) x P(hot|p) x P(high|p) x P(false|p)

P(X|p) x P(p) = 0.0105 VS P(X|n) x P(n) = 0.0183 so sample X classified as class 'n' ignore P(X)

==========

Clustering

collection of data objects -> high intra-class similarity, low inter-class similarity unsupervised classification: no predefined class

application:

- spatial data analysis, eg. GIS
- detect spatial cluster
- image processing

good clustering

result quality

- depends on similarity measure by method and implementation clustering method quality
- ability to find hidden pattern

quality

similarity expressed in term of distance function d(i,j) function different for interval-scaled, boolean, categorical, ordinary variables weights on different variables based on applications and data semantics

Minkowski distance = q_root(|x1-y1|^q + |x2-y2|^q + ... + |x_p-y_p|^q) Manhattan distance = |x1-y1| + |x2-y2| + ... + |x_p-y_p|

distance for binary variables

	1	**0**	**sum**
1	a	b	a+b
0	c	d	c+d
sum	a+c	b+d	p

- simple matching coefficient (invariant,symmetric bianry)
- eg. gender d(i,j) = (b+c) / (a+b+c+d)
- Jaccard coefficient (non-invariant, asymmetric binary) d(i,j) = (b+c) / (a+b+c)

distance for nominal variables

1. m = num of matches, p = total num of variables d(i,j) = (p-m) / p
2. create new binary variable for each M nominal states

how to choose dissimilarity metric?

Numeric data:

1. Euclidean
2. Squared euclidean distance
3. Manhattan distance
4. Bray and Curtis

Categorical data

- Mismatch value

Similarity = numerical measure how alike 2 data objects are, range [0,1] Dissimilarity = numerical measure how different 2 data objects are, min=0, upper limit varies

type	**dissimilarity**	**similarity**
Nominal	d = 0 if p=q	s = 1 if p=q
	d = 1 if p!=q	s = 0 if p!=q
Ordinal	d = |p-q| /(n-1)	s = 1 - |p-q|/(n-1)
	[values mapped to integers 0 to n-1]	
Interval / Ratio	d = |p-q|	s = -d, s = q/(1+d) or
		s = 1 - (d - min_d)/(max_d - min_d)

clustering methods

partitioning

construct various partitions, evaluate them by some criteria

- density-based
- grid-based
- model-based

given particular k, partition n object

k-means

1. partition object into k non-empty subsets
2. compute seed point as centroid of clusters
3. assign objects to nearest seed point

4. back to step2, stop when no new assignment

O(#iterations * #clusters * #objects) need to specify k, unable to handle noise and outliers

k-modes

handle categorical values using dissimilarity measures frequency based method to update modes of clusters

k-medoids

choose a point that is closest to all points

- Partitioning around medoids (PAM) start from initial set of medoids, iteratively replace by other points works effectively for small data sets, not large

CLARA: deal with larger data sets than PAM

hierarchical clustering

single-linkage clustering

nearest neighbour technique: distance = closest pair grouping clusters in bottom up fashion, each step combine 2 clusters contain closest pair of element

1. group closest 2+ elements
2. recalc distance from group to other nodes (take min)
3. back to step 1, repeat

Complete Linkage Method

furthest neighbor technique: distance = most distant pair other same as single-linkage

Group average

distance = average of all pairs of individuals

Centroid clustering

groups = mean values computed for each attribute (mean vector) distance = distance between 2 mean vector

==========

Data Preprocessing

why

dirty data

- incomplete, noisy, inconsistent quality data => quality minging result

Data quality

criteria

- accuracy, consistency, completeness
- timeliness(time data capture - time event captured)
- believability (property of being worthy of being believed by rational and informed user)
- value added
- interpretability
- accessibility

category

- intrinsic
- contextual
- representational
- accessibility

Process

Data cleaning

fill missing, smooth noisy data, remove outlier, resolve inconsistency

handle missing data

1. ignore tuple, not good for attribute value
2. global constant (eg. unknown, new class)
3. attribute mean
4. attribute mean of same class
5. most probable value inferred by Bayesian / decision tree

nosiy data

random error handle:

1. Binning method
    - sort, partition into bins
    - smooth by bin means/median/boundaries (change num to closer min/max value)

        > sort by equal width/depth

2. clustering
    - remove outlier
3. combined computer and human inspection
4. regression
    - fit data into regression function

Data integration

integrate multiple database, files detect and resolve conflict (eg. different scales, representations) remove redundant data

Data transformation

- smoothing remove noise
- aggregation construct data cube
- generalization concept hierarchy climbing (different level of abstraction on data n-dimensional cube)
- normalization min-max, z-score([v-mean]/stddev), decimal scaling
- attribute/feature construction

Data reduction

obtain reduced representation in volume, produce similar analytical result

dimensionality reduction

- feature selection min set of feature st. probability distribution of classes ~ original distribution reduce # patterns
- heuristic method decision-tree: choose attributes for decision
- parametric method assume data fit some regression model, estimate model parameter, discard data
- non-parametric method not assume model

> histogram eg. divide data into buckets, store avg/sum for each bucket use dynamic programming clustering partition data into cluster, store cluster representation only sampling choose representative subset of data stratified sampling: approx percentage of each class

Data discretization

part of data reduction, for numerical data

- discretization replace actual data with interval labels
- concept hierarchies replace low level concept(numeric age) with high level one(young, middle-aged...)

Data warehouse

why

information gap by: systems designed for transaction processing, not query analysis unified access to data, collect info, provide unifom UI

what

decision support database, separate from operational database support information processing

subject-oriented

around major subjects model, analyse data exclude not useful data

integrated

integrate multiple different data source (db, file, records) data converted, processed

time-variant

time horizon significantly longer, eg. 5-10 years

non-volatile

no operational update, no need concurrency control only need initial data loading, data access

Data warehouse VS DBMS

- DBMS, OLTP (online transaction processing) wrapper when query posed to client, dict used to translate query -> appropriate queries traditional DBMS, day-to-day operation up to date, RW application oriented
- data warehouse, OLAP (online analytical processing) information integrated in advance decision making

Data Cubes

base cuboid: n-D top most 0-D cuboid: apex cuboid lattice form data cube

- different combination of attributes (1/2/3/4.. num of attributes together)

schema

- star schema simple db design, good for ad-hoc query

item table: item_key, item_name, ... location table: location_key, street ... sales fact table: item_key, location_key, units_sold ...

fact table: factual/quantitative data (measure) dimension table: descriptive data about business [many records]

- galaxy schema multiple fact tables diferent fact tables for each levels of aggregation
- snowflake schema expanded version of star schema that tables are fully normalized sub-dimension table under higher dimension table
- distributive count, sum, min, max
- algebraic avg, min_N
- holistic median, mode, rank

OLAP operations

roll up drill down slice, dice pivot drill across drill through

data warehouse application

information processing analytical processing data mining

OLAM (online analytical mining)

mining with drilling, dicing, pivoting

design

- top-down view select relevant info for data warehouse
- data source view expose info being captured, stored, managed
- data warehouse view contain fact table, dimension table
- business query view from end-user view

design process

1. choose business process to model
2. grain of process
3. dimension apply to each fact table
4. measure populate each fact table

warehouse model

- enterprise warehouse enterpreise wide data multiple subject area multiple data source large memory needed
- data mart to specific group of user department wide single subject limited data source occupy limited memory
- virtual warehouse views over operational DB

ROLAP (Relational) MOLAP (Multi-dimensional) [Array based] HOLAP (Hybrid) SQL server

view materialization

allow fast answer materialized view / snapshot = cache query result DW = materialized view of operational DB and external data source

materialize small, carefully chosen set of views that can evaluate majority of important queries view refreshing = make view consistent with DW base table

view update: immediate / deferred aggregates:

- distributive refresh without any problem
- algebraic easily refreshed if contain all other necessary data
- holistic hard to refresh

data extraction

through gateway (ODBC, OLE, JDBC)

data cleaning

data loading

sorting, summarization, aggregation, building indexes, materialized views full incremental loading during refresh

data refreshing

frequency procedure

- data shipping
- transaction shipping

Cube computation

data cube = lattice of cuboids, from 0-L to n-L can materialize every cuboid, none, some new operation `cube by` SQL like, generate all possible attribute combination by selection

1. drill, roll -> SQL and/or1. OLAP operations
2. determine which materialized cuboid operation apply
3. index, compressed VS dense array