Web Usage Mining

Bolong Zhang
3/27/2019

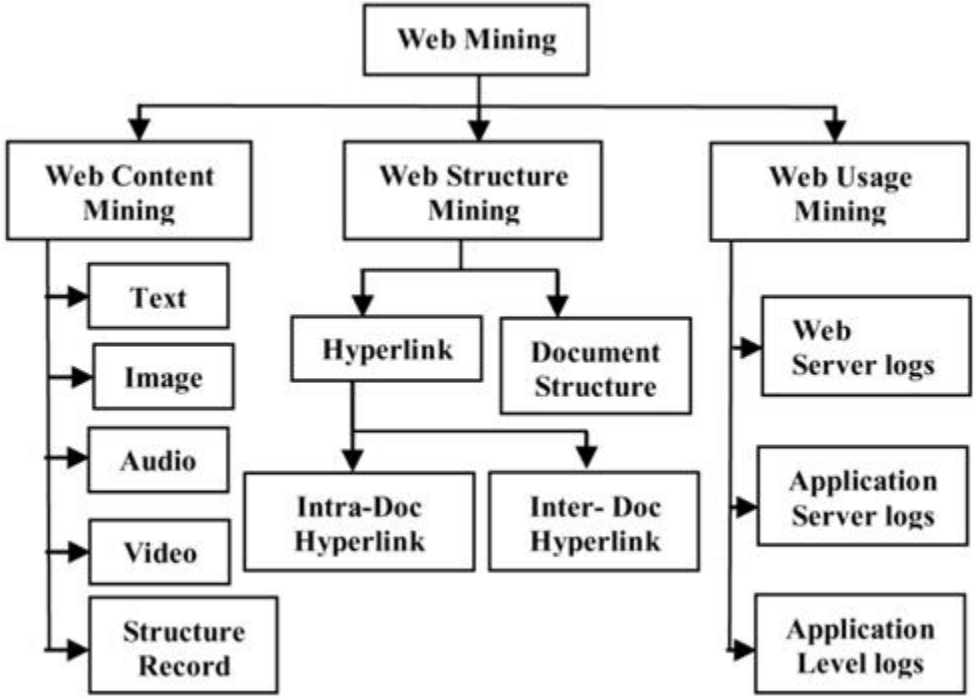
Outline

- Overview
- Aim & Obejective
- Different Levels
- Algorithm
- Clustering Techniques

Overview

Web Mining

Finding information and patterns from the World Wide Web



Web Usage Mining

Discovering user's navigation pattern and **predicting user's behavior**

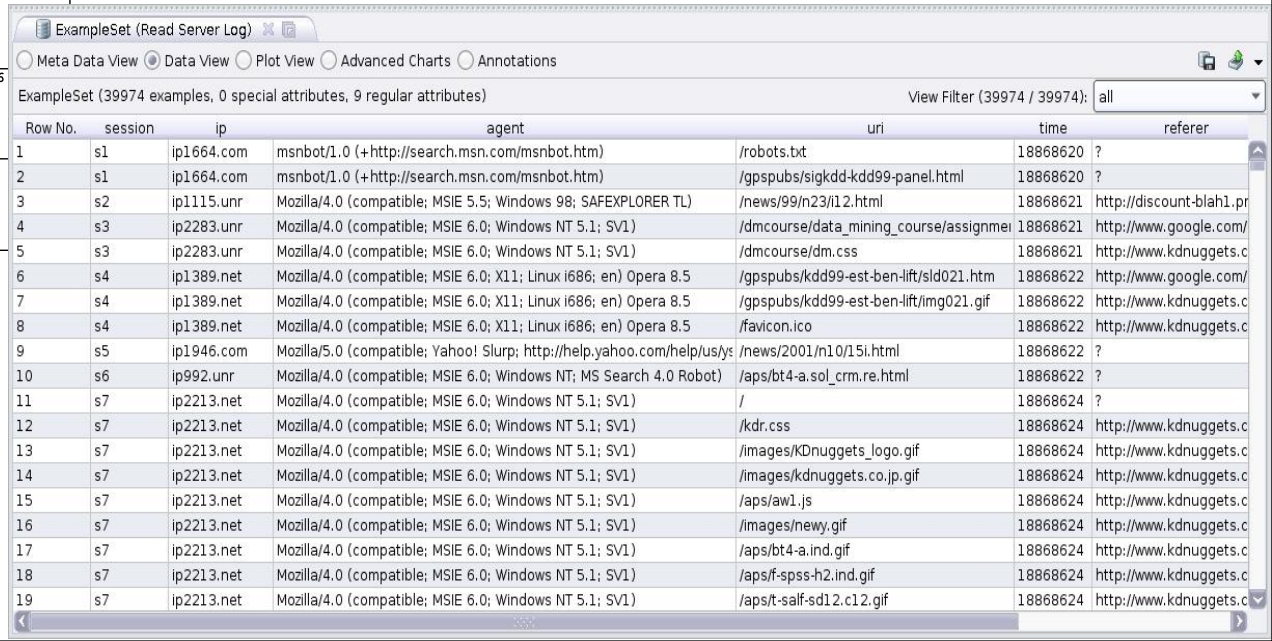
Web Server Logs

1	2006-02-01 00:08:43 1.2.3.4 - GET /classes/cs589/papers.html - 200 9221 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+2.0.50727) http://dataminingresources.blogspot.com/
2	2006-02-01 00:08:46 1.2.3.4 - GET /classes/cs589/papers/cms-tai.pdf - 200 4096 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1;+.NET+CLR+2.0.50727) http://maya.cs.depaul.edu/~classes/cs589/papers.html
3	2006-02-01 08:01:28 2.3.4.5 - GET /classes/ds575/papers/hyperlink.pdf - 200 318814 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1) http://www.google.com/search?hl=en&lr=&q=hyperlink+analysis+for+the+web+survey
4	2006-02-02 19:34:45 3.4.5.6 - GET /classes/cs480/announce.html - 200 3794 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1) http://maya.cs.depaul.edu/~classes/cs480/
5	2006-02-02 19:34:45 3.4.5.6 - GET /classes/cs480/styles2.css - 200 1636 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1) http://maya.cs.depaul.edu/~classes/cs480/announce.html
6	2006-02-02 19:34:45 3.4.5.6 - GET /classes/cs480/header.gif - 200 6027 HTTP/1.1 maya.cs.depaul.edu Mozilla/4.0+(compatible;+MSIE+6.0;+Windows+NT+5.1;+SV1) http://maya.cs.depaul.edu/~classes/cs480/announce.html

parameters of log files:
(1)User Name
(2)Visiting Path
(3)Time Stamp
(4)Page Last Visited
(5)Success Rate
(6)User Agent
(7)URL
(8)Request Type

records the browsing behavior of site visitors

<ip_addr> <base_url> - <date> <method> <file> <protocol> <code> <bytes> <referrer> <user_agent>


ExampleSet (Read Server Log)

Meta Data View | Data View | Plot View | Advanced Charts | Annotations

ExampleSet (39974 examples, 0 special attributes, 9 regular attributes) View Filter (39974 / 39974): all

Row No.	session	ip	agent	uri	time	referer
1	s1	ip1664.com	msnbot/1.0 (+http://search.msn.com/msnbot.htm)	/robots.txt	18868620	?
2	s1	ip1664.com	msnbot/1.0 (+http://search.msn.com/msnbot.htm)	/gpspubs/sigkdd-kdd99-panel.html	18868620	?
3	s2	ip1115.unr	Mozilla/4.0 (compatible; MSIE 5.5; Windows 98; SAFEXPLORER TL)	/news/99/n23/i12.html	18868621	http://discount-blah1.pr
4	s3	ip2283.unr	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/dmcourse/data_mining_course/assignmer	18868621	http://www.google.com/
5	s3	ip2283.unr	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/dmcourse/dm.css	18868621	http://www.kdnuggets.c
6	s4	ip1389.net	Mozilla/4.0 (compatible; MSIE 6.0; X11; Linux i686; en) Opera 8.5	/gpspubs/kdd99-est-ben-lift/sld021.htm	18868622	http://www.google.com/
7	s4	ip1389.net	Mozilla/4.0 (compatible; MSIE 6.0; X11; Linux i686; en) Opera 8.5	/gpspubs/kdd99-est-ben-lift/img021.gif	18868622	http://www.kdnuggets.c
8	s4	ip1389.net	Mozilla/4.0 (compatible; MSIE 6.0; X11; Linux i686; en) Opera 8.5	/favicon.ico	18868622	http://www.kdnuggets.c
9	s5	ip1946.com	Mozilla/5.0 (compatible; Yahoo! Slurp; http://help.yahoo.com/help/us/ys	/news/2001/n10/15i.html	18868622	?
10	s6	ip992.unr	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT; MS Search 4.0 Robot)	/aps/bt4-a.sol_crm.re.html	18868622	?
11	s7	ip2213.net	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/	18868624	?
12	s7	ip2213.net	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/kdr.css	18868624	http://www.kdnuggets.c
13	s7	ip2213.net	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/images/KDnuggets_logo.gif	18868624	http://www.kdnuggets.c
14	s7	ip2213.net	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/images/kdnuggets.co.jp.gif	18868624	http://www.kdnuggets.c
15	s7	ip2213.net	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/aps/aw1.js	18868624	http://www.kdnuggets.c
16	s7	ip2213.net	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/images/newy.gif	18868624	http://www.kdnuggets.c
17	s7	ip2213.net	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/aps/bt4-a.ind.gif	18868624	http://www.kdnuggets.c
18	s7	ip2213.net	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/aps/f-spss-h2.ind.gif	18868624	http://www.kdnuggets.c
19	s7	ip2213.net	Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	/aps/t-salf-sd12.c12.gif	18868624	http://www.kdnuggets.c

Processes

3 main stages

1. Preprocessing:

raw data -> data abstraction

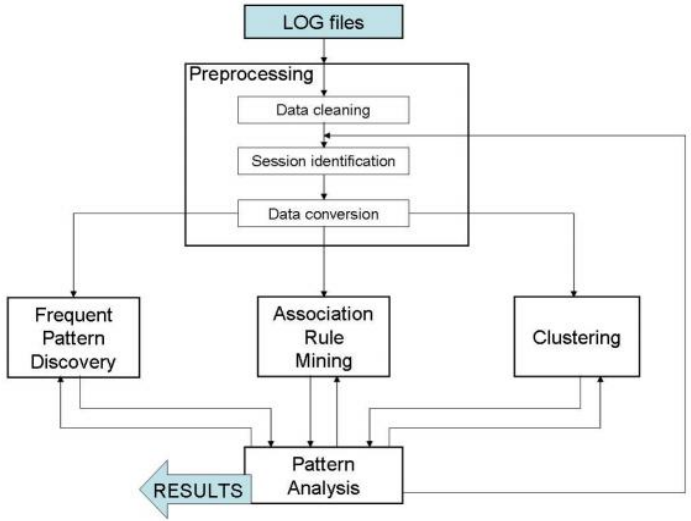
(users, sessions, episodes, clicktrea ms, and pageviews)

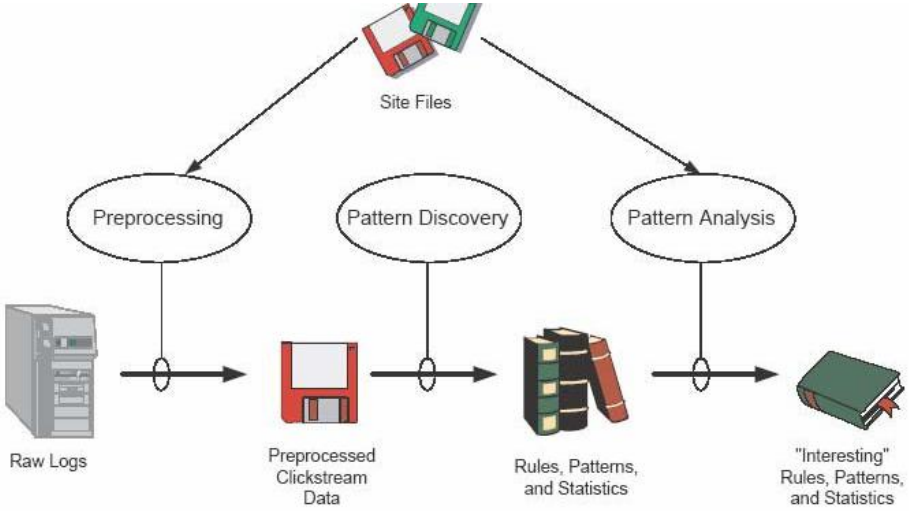
2. Pattern Discovery:

is the key component of WUM, whic h converges the algorithms and tech niques from data mining, machine le arning, statistics and pattern recogni tion etc. research categories.

3. Pattern Analysis:

Validation and interpretation of the m ined patterns





Preprocessing

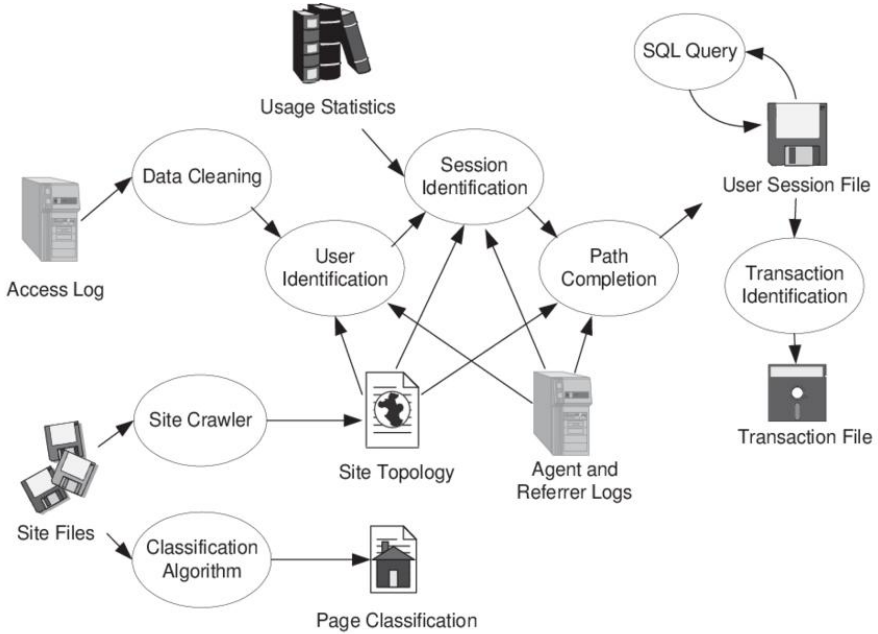
Data Cleaning:

User Identification:

Session Identification:

Path Completion:

Formatting:



Preprocessing

Data Cleaning:

Staus Codes:

Sever Error

Redirect:
300 Series

Success:
200 Series

Failures:
404 Page Not Found
401 Unauthorized
403 Forbidden

Algorithm: DataPreparation

1. Start
2. Check for data available in server log
3. If raw data is available goto step 4 else goto step 2
4. Cleaning data by removing gap, .jpg , .gif or sound file.
5. Execute UserIdentification.
6. Execute SessionIdentification.
7. Divide the session in transaction with a certain duration.
8. If any data available goto step 4 else goto step 9
9. exit

Input: log file
Output: cleaned log file
Step 1: Begin
Read records in log file
Step 2: For each record
Read (status code, method)
Step 3: If (status code= '40*' and method= '**')
Then, remove that status field
Get IP address and URL link
Step 4: If (suffix_URL_link= {.gif, .jpg, .css, .av}
&& request= "implicit")
Then, remove that URL_link
Else
Save IP address and URL_link
End if
End if
Step 5: If (status code! = '40*' and method= '**')
Then, Get IP address and URL link
If (suffix_URL_link= {.gif, .jpg, .css, .av}
&& request= "implicit")
Then, remove that URL_link
Else
Save IP address and URL_link
End if
End if
Next record
End for
End

Preprocessing

User Identification:

associate page references with different users

Input: cleaned log file
Output: Unique Users file
Step 1: Begin
Initialize IP_List=0; Users List=0; Browser List=0; No-Of-users=0;
Read Record From cleaned log file
Step 2: For each page in log file
Read Record.IP address and Record. Browser
Step 3: If Record.IP address is not in IP_List
Then, add Record.IP address in to IP_List
Add Record. Browser in to Browser List
No-Of-users++
Add new user in to User List.
Else
If (Record.IPaddress is present in IP_List and Record. Browser not in Browser List)
Then, No-Of-users++
Add new user in to User List.
End If
End for
End

Method	Description	Privacy Concerns	Advantages	Disadvantages
IP Address + Agent	Assume each unique IP address/Agent pair is a unique user	Low	Always available. No additional technology required.	Not guaranteed to be unique. Defeated by rotating IPs.
Embedded Session Ids	Use dynamically generated pages to associate ID with every hyperlink	Low to medium	Always available. Independent of IP addresses.	Cannot capture repeat visitors. Additional overhead for dynamic pages.
Registration	User explicitly logs in to the site.	Medium	Can track individuals not just browsers	Many users won't register. Not available before registration.
Cookie	Save ID on the client machine.	Medium to high	Can track repeat visits from same browser.	Can be turned off by users.
Software Agents	Program loaded into browser and sends back usage data.	High	Accurate usage data for a single site.	Likely to be rejected by users.

Time	IP	URL	Ref	Agent
0:01	1.2.3.4	A	-	IE5;Win2k
0:09	1.2.3.4	B	A	IE5;Win2k
0:10	2.3.4.5	C	-	IE6;WinXP;SP1
0:12	2.3.4.5	B	C	IE6;WinXP;SP1
0:15	2.3.4.5	E	C	IE6;WinXP;SP1
0:19	1.2.3.4	C	A	IE5;Win2k
0:22	2.3.4.5	D	B	IE6;WinXP;SP1
0:22	1.2.3.4	A	-	IE6;WinXP;SP2
0:25	1.2.3.4	E	C	IE5;Win2k
0:25	1.2.3.4	C	A	IE6;WinXP;SP2
0:33	1.2.3.4	B	C	IE6;WinXP;SP2
0:58	1.2.3.4	D	B	IE6;WinXP;SP2
1:10	1.2.3.4	E	D	IE6;WinXP;SP2
1:15	1.2.3.4	A	-	IE5;Win2k
1:16	1.2.3.4	C	A	IE5;Win2k
1:17	1.2.3.4	F	C	IE6;WinXP;SP2
1:26	1.2.3.4	F	C	IE5;Win2k
1:30	1.2.3.4	B	A	IE5;Win2k
1:36	1.2.3.4	D	B	IE5;Win2k

User 1

0:01	1.2.3.4	A	-
0:09	1.2.3.4	B	A
0:19	1.2.3.4	C	A
0:25	1.2.3.4	E	C
1:15	1.2.3.4	A	-
1:26	1.2.3.4	F	C
1:30	1.2.3.4	B	A
1:36	1.2.3.4	D	B

User 2

0:10	2.3.4.5	C	-
0:12	2.3.4.5	B	C
0:15	2.3.4.5	E	C
0:22	2.3.4.5	D	B

User 3

0:22	1.2.3.4	A	-
0:25	1.2.3.4	C	A
0:33	1.2.3.4	B	C
0:58	1.2.3.4	D	B
1:10	1.2.3.4	E	D
1:17	1.2.3.4	F	C

Preprocessing

Algorithm: SessionIdentificaton

1. Start
2. Take time of the first log entries.
3. Calculate the threshold time from the starting time.
4. if threshold >30 min
   session change
   else same session
5. exit

Session Identification:

divide all pages accessed by users into sessions

Time oriented heuristics consider boundaries on time spent on individual pages or in the entire a site during a single visit

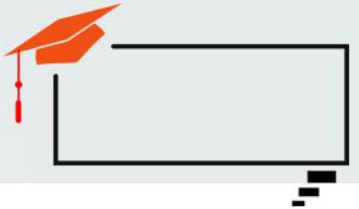
1. sort users
2. sessionize using heuristics: time interval as heuristics

0:01	1.2.3.4	A	-	IE5;Win2k
0:09	1.2.3.4	B	A	IE5;Win2k
0:19	1.2.3.4	C	A	IE5;Win2k
0:25	1.2.3.4	E	C	IE5;Win2k
1:15	1.2.3.4	A	-	IE5;Win2k
1:26	1.2.3.4	F	C	IE5;Win2k
1:30	1.2.3.4	B	A	IE5;Win2k
1:36	1.2.3.4	D	B	IE5;Win2k

0:01	1.2.3.4	A	-	IE5;Win2k
0:09	1.2.3.4	B	A	IE5;Win2k
0:19	1.2.3.4	C	A	IE5;Win2k
0:25	1.2.3.4	E	C	IE5;Win2k

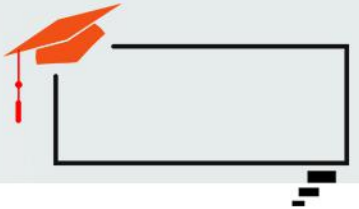
1:15	1.2.3.4	A	-	IE5;Win2k
1:26	1.2.3.4	F	C	IE5;Win2k
1:30	1.2.3.4	B	A	IE5;Win2k
1:36	1.2.3.4	D	B	IE5;Win2k

Pattern Discovery



- Statistical Analysis
- Clustering
- Classification
- Association Rules
- Sequential Patterns

Pattern Discovery



- **Statistical Analysis**

Page views, viewing time, length of navigational path

Frequency , mean, median....

Category	Description
General statistics	1) Total number of hits 2) Total number of visitors 3) Different errors 4) Successful visits 5) Incomplete visits 6) Error reports
Access statistics	Request Hit and Miss count based on 1) IP address 2) URL
Periodical statistics	Access of web pages according to period of time e.g. daily, monthly, yearly.

Pattern Discovery

- **Clustering**

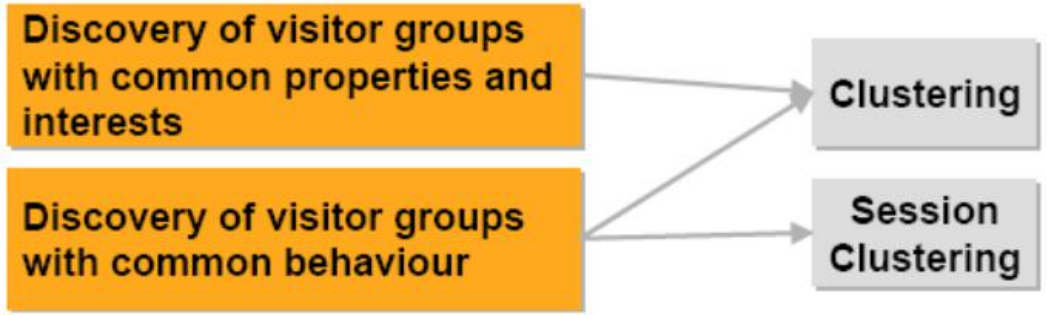
Objects:

1. Users

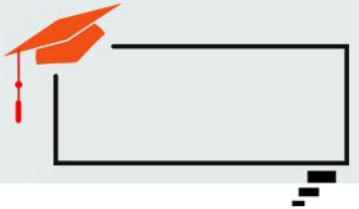
similar navigation patterns

2. Pages

related content



Pattern Discovery



- **Clustering Algorithm**

Density-based algorithms : DBSCAN(common), OPTICS

Grid-based algorithms : STING, CLIQUE, WaweCluster.

Model-based algorithms : MCLUST

Fuzzy algorithms : FCM (Fuzzy CMEANS)

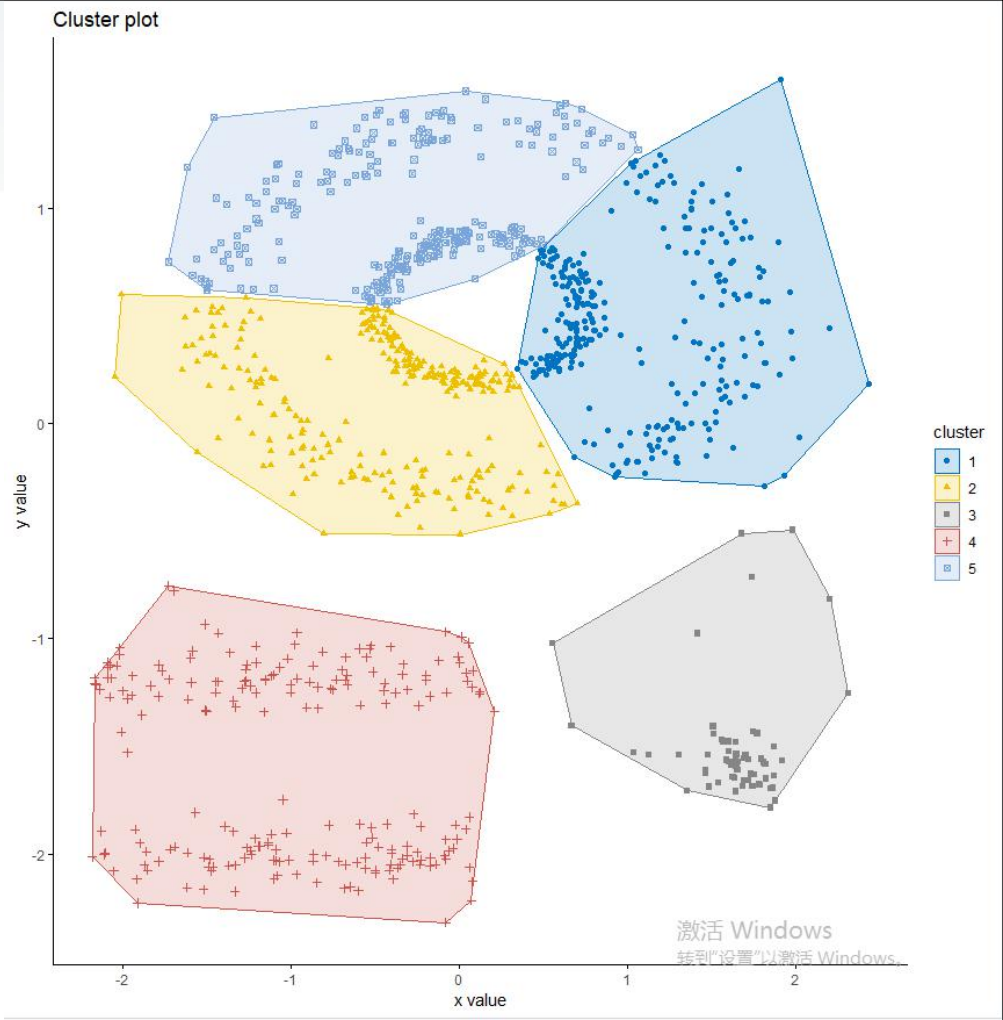
Why not distance-based algorithm ?

Pattern Discovery

- **Clustering Algorithm**

k- means

DBSCAN can find **non-linearly separable clu sters.**



Pattern Discovery

- **Clustering Algorithm**

Density-based algorithms : DBSCAN, OPTICS

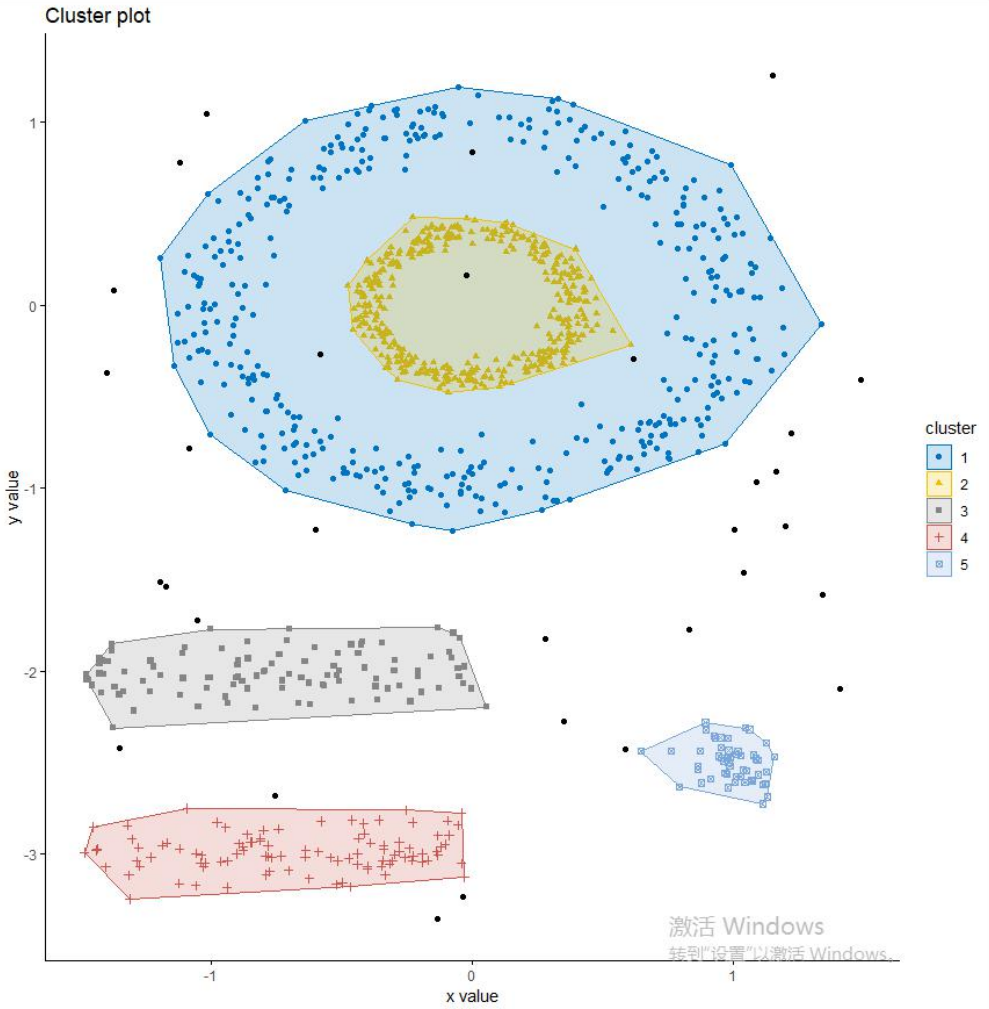
Advantages:

1. Not specify the number of clusters.

2. Any shapes.

3. Identify outliers.

4. Large



Pattern Discovery

- **DBSCAN**

D

k

Eps

MinPts

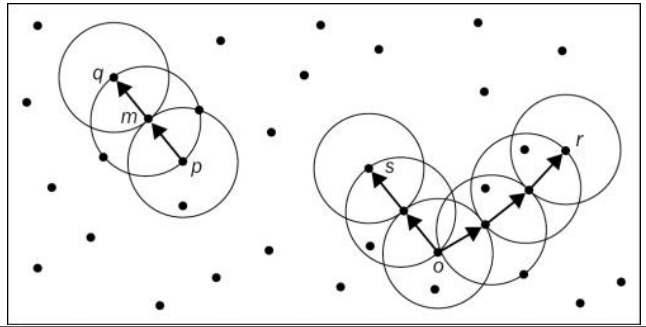
Eps as radius, minpt as neighborhood density thr
eshold. An object is noise only if there is no clust
er that contains

DBSCAN $(\mathbf{D}, \varepsilon, minpts)$:

1 $Core \leftarrow \emptyset$
2 **foreach** $\mathbf{x}_i \in \mathbf{D}$ **do**
3 Compute $N_\epsilon(\mathbf{x}_i)$
4 $id(\mathbf{x}_i) \leftarrow \emptyset$
5 **if** $N_\varepsilon(\mathbf{x}_i) \geq minpts$ **then** $Cores \leftarrow Cores \cup \{\mathbf{x}_i\}$
6 $k \leftarrow 0$
7 **foreach** $\mathbf{x}_i \in Core$, *such that* $id(\mathbf{x}_i) = \emptyset$ **do**
8 $k \leftarrow k + 1$
9 $id(\mathbf{x}_i) \leftarrow k$
10 DENSITYCONNECTED $(\mathbf{x}_i, k)$
11 $\mathcal{C} \leftarrow \{C_i\}_{i=1}^{k}$, where $C_i \leftarrow \{\mathbf{x} \in \mathbf{D} \mid id(\mathbf{x}) = i\}$
12 $Noise \leftarrow \{\mathbf{x} \in \mathbf{D} \mid id(\mathbf{x}) = \emptyset\}$
13 $Border \leftarrow \mathbf{D} \setminus \{Core \cup Noise\}$
14 **return** $\mathcal{C}, Core, Border, Noise$

DENSITYCONNECTED $(\mathbf{x}, k)$:

15 **foreach** $\mathbf{y} \in N_\varepsilon(\mathbf{x})$ **do**
16 $id(\mathbf{y}) \leftarrow k$
17 **if** $\mathbf{y} \in Core$ **then** DENSITYCONNECTED $(\mathbf{y}, k)$

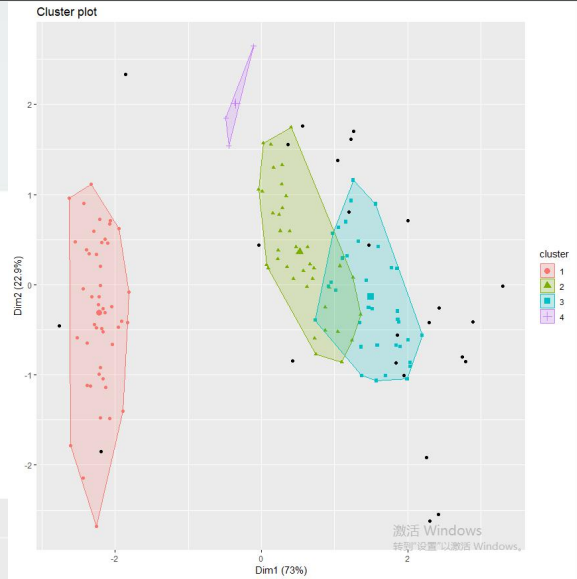


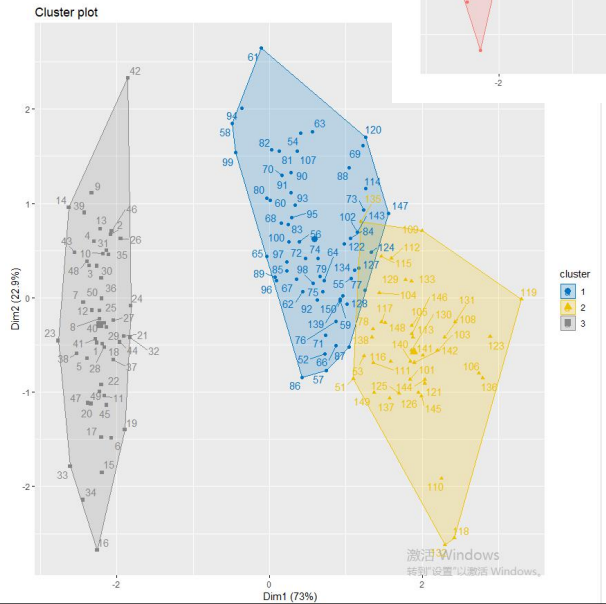
Pattern Discovery

- **Clustering Algorithm**

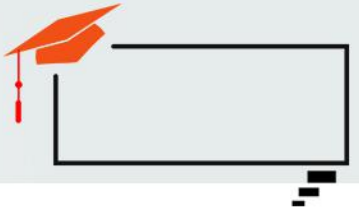
Fuzzy algorithms : FCM (Fuzzy C MEANS)

Like k-means, however, each point has a weighting associated with a particular cluster





Pattern Discovery



- **Association Rules -** correlation between users

Frequent itemsets

Apriori algorithm:

- A subset of a frequent itemset must also be a frequent itemset
  - i.e., if {AB} is a frequent itemset, both {A} and {B should be a frequent itemset

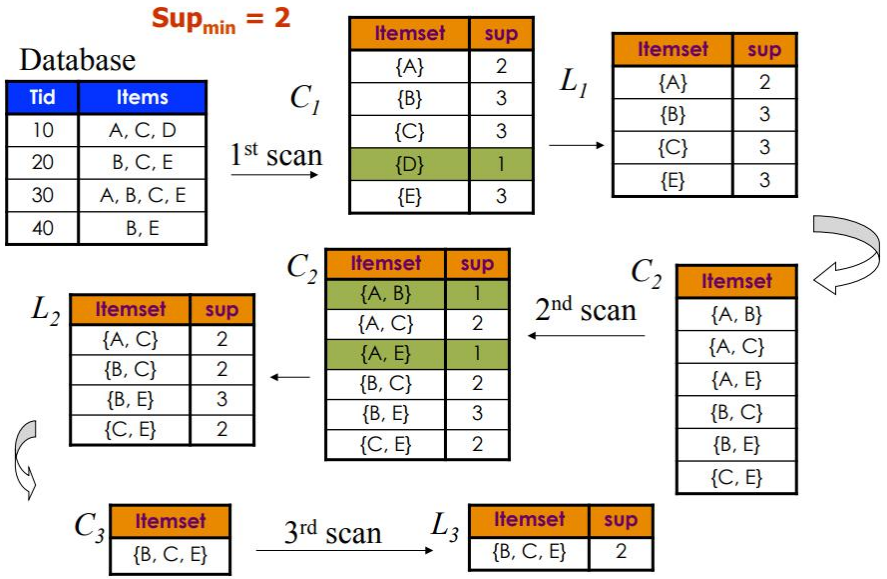
– Iteratively find frequent itemsets with cardinality from 1 to k (k-itemset)

Pattern Discovery

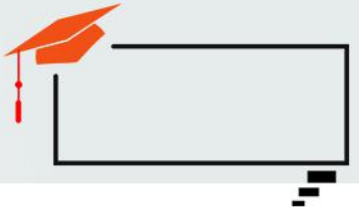
- **Association Rules**

C_k: Candidate itemset of size k
L_k : frequent itemset of size k

L_1 = {frequent items};
for (k = 1; L_k !=∅; k++) **do begin**
  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 min_support
  end
return $\cup_k L_k$;



Pattern Discovery



- **Association Rules**

Candidate Generation

-step 1 : self-joining Lk

-step 2 : pruning

Example: Suppose we have the following frequent 3-itemsets and we would like to generate the 4-itemsets ca ndidates

L3={{I1, I2, I3} , {I1, I2, I4}, {I1, I3, I4}, {I1, I3, I5}, {I2,I3,I4}}

Remove duplicate

- Self-joining: L3*L3 gives:

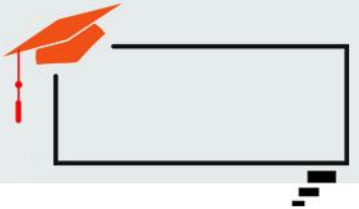
  {I1,I2,I3,I4} from {I1, I2, I3} , {I1, I2, I4}, and {I2,I3,I4}

  {I1,I3,I4,I5} from {I1, I3, I4} and {I1, I3, I5}

Pruning: {I1,I3,I4,I5} is removed because {I1,I4,I5} is not in L3

L4={I1,I2,I3,I5}

Pattern Discovery



- **Association Rules**

- Once the frequent itemsets have been found, it is straightforward to generate strong association rules that satisfy:

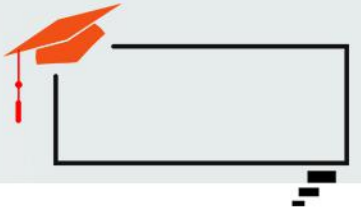
- minimum support
- minimum confidence

- Relation between Support and Confidence

$$\text{Confidence}(X \Rightarrow Y) = P(Y \mid X) = \frac{\text{support\_count}(X \cup Y)}{\text{support\_count}(X)}$$

support_count(X) is the number of transactions containing the itemset X

Pattern Discovery



- **Association Rules**

- For each frequent itemset L, generate all non empty subsets of L
- For every non empty subset S of L, output the rule:

If (support_count(L)/support_count(S)) >= min_conf

$$S \Rightarrow (L - S)$$

a simple correlation measure - Lift

$$\text{Lift}(X, Y) = \frac{P(X \cup Y)}{P(X)P(Y)}$$

> 1, X, Y positively correlated ; = 1 Independent; <1 negatively correlated

Pattern Discovery

- **Classification**

Classification is done to identify the characteristics that indicate the group to which each case belongs.

K-nearest neighbour

Distance:

(1) Euclidean Distance:

$$D(x, y) = ((\sum_{i=1}^{m} |x_i - y_i|)^2)^{1/2}$$

(2) Manhattan Distance:

$$D(x, y) = \sum |x_i - y_i|$$

(3) Minkowski Distance

$$D(x, y) = ((\sum_{i=1}^{m} |x_i - y_i|)^r)^{1/r}$$

(4) Cityblock, Canberra.....

Input Parameters: Data set, k

Output: Class membership

Step 1: Store all the training tuples.

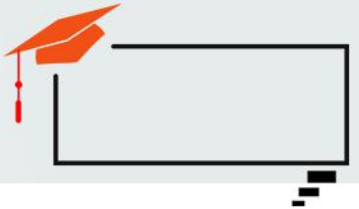
Step 2: For each unseen tuple which is to be classified

A Compute distance of it with all the training tuples using Euclidean Distance.

B. Find the k nearest training tuples to the unseen tuple.

C. Assign the class which is most common in the k nearest training tuples to the unseen tuple.

End for



Thanks

Any quenstions ?