Lecture 11. Advanced Topics on Pattern Discovery

Lecture 11. Advanced Topics on Pattern Discovery

❑ Frequent Pattern Mining in Data Streams

❑ Spatiotemporal and Trajectory Pattern Mining

❑ Pattern Discovery for Software Bug Mining

❑ Pattern Discovery for Image Analysis

❑ Pattern Mining and Society: Privacy Issues

❑ Looking Forward

Session 1. Frequent Pattern Mining in Data Streams

Challenges for Data Analysis in Data Streams

❑ Data Streams
  ❑ Features: Continuous, ordered, changing, fast, huge volumn
  ❑ Contrast with traditional DBMS (finite, persistent data sets)
❑ Characteristics
  ❑ Huge volumes of continuous data, possibly infinite
  ❑ Fast changing and requires fast, real-time response
  ❑ Data stream captures nicely our data processing needs of today
  ❑ Random access is expensive: single scan algorithm (*can only have one look*)
  ❑ Store only the summary of the data seen thus far
  ❑ Most stream data are at low-level and multi-dimensional in nature, needs multi-level and multi-dimensional processing

Architecture: Stream Data Processing

SDMS (Stream Data Management System)

User/Application

Continuous Query

Results

Multiple streams

Stream Query Processor
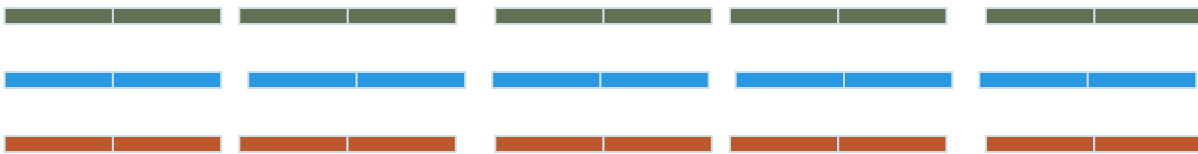
Q: How to mine frequent patterns in data streams?

Scratch Space
(Main memory and/or Disk)
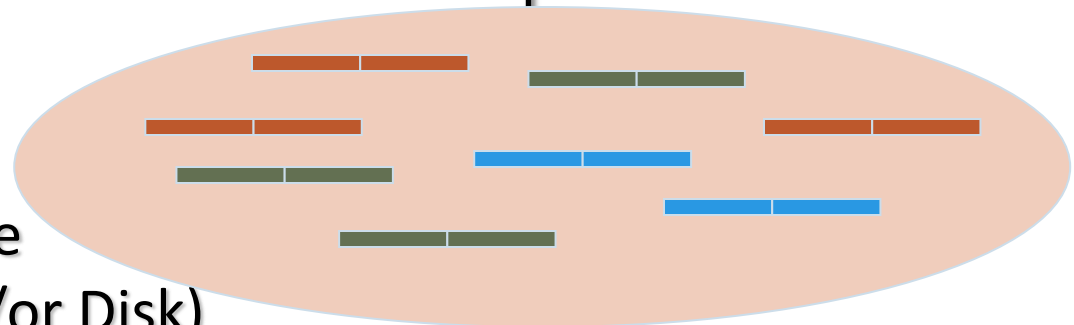
Stream Data Mining Tasks

❏   Stream mining vs. stream querying

  ❏   Stream mining shares many difficulties with stream querying

    ❏   E.g., single-scan, fast response, dynamic, …

    ❏   But often requires less "precision", e.g., no join, grouping, sorting

  ❏   Patterns are hidden and more general than querying

❏ Stream data mining tasks

  ❏   Pattern mining in data streams

  ❏   Multi-dimensional on-line analysis of streams

  ❏   Clustering data streams

  ❏   Classification of stream data

  ❏   Mining outliers and anomalies in stream data

Mining Approximate Frequent Patterns

- ❑ Mining precise frequent patterns in stream data: Unrealistic
  - ❑ Cannot even store them in a compressed form (e.g., FPtree)
- ❑ Approximate answers are often sufficient for pattern analysis
  - ❑ Ex.: A router
    - ❑ is interested in all flows whose frequency is at least 1% (σ) of the entire traffic stream seen so far
    - ❑ and feels that 1/10 of σ (ε = 0.1%) error is comfortable
- ❑ How to mine frequent patterns with good approximation?
  - ❑ Lossy Counting Algorithm (Manku & Motwani, VLDB'02)
    - ❑ Major ideas: Not to keep the items with very low support count
    - ❑ Advantage: Guaranteed error bound
    - ❑ Disadvantage: Keeping a large set of traces

Lossy Counting for Frequent Single Items

Bucket 1 | Bucket 2 | Bucket 3

Divide stream into 'buckets' (bucket size is $1/\varepsilon = 1000$)

First Bucket of the Stream

Empty (summary) +
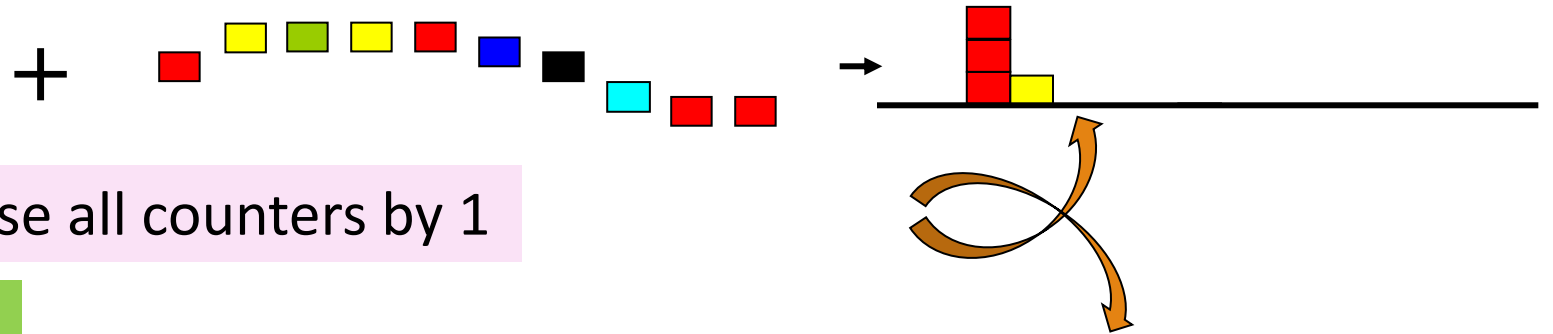
At bucket boundary, decrease all counters by 1

Next Bucket of the Stream

+
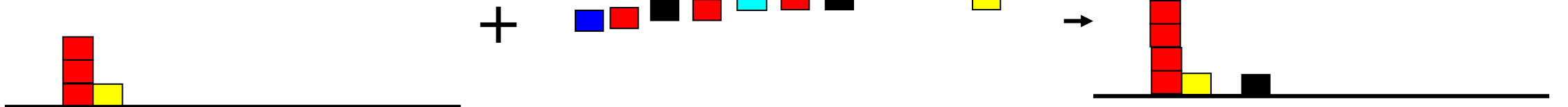
Approximation Guarantee

❑ Given: (1) support threshold: σ, (2) error threshold: ε, and (3) stream length N

❑ Output: items with frequency counts exceeding (σ – ε) N

❑ How much do we undercount?

If stream length seen so far = N and bucket-size = 1/ε

then frequency count error ≤ # of buckets

$$= N/bucket\text{-}size = N/(1/ε) = εN$$

❑ Approximation guarantee

   ❑ No false negatives

   ❑ False positives have true frequency count at least (σ–ε)N

   ❑ Frequency count underestimated by at most εN

Other Issues and Recommended Readings

❑ Other issues on pattern discovery in data streams

    ❑ Space-saving computation of frequent and top-k elements (Metwally, Agrawal, and El Abbadi, ICDT'05)

    ❑ Mining approximate frequent k-itemsets in data streams

    ❑ Mining sequential patterns in data streams

❑ Recommended Readings

    ❑ G. Manku and R. Motwani, "Approximate Frequency Counts over Data Streams", VLDB'02

    ❑ A. Metwally, D. Agrawal, and A. El Abbadi, "Efficient Computation of Frequent and Top-k Elements in Data Streams", ICDT'05

Session 2. Spatiotemporal and Trajectory Pattern Mining

Spatial Patterns and Associations

- ❑ Spatial frequent patterns and association rule: $A \Rightarrow B$ [*s%, c%*]
  - ❑ A and B are sets of spatial or non-spatial predicates, e.g.,
    - ❑ Topological relations: *intersects, overlaps, disjoint,* etc.
    - ❑ Spatial orientations: *left_of, west_of, under,* etc.
    - ❑ Distance information: *close_to, within_distance,* etc.
  - ❑ *s%*: support, and *c%*: confidence of the rule
- ❑ Example: Rules likely to be found
  - ❑ *is_a(x, large_town) ^ intersect(x, highway) → adjacent_to(x, water) [7%, 85%]*
- ❑ Explore *spatial autocorrelation*: Spatial data tends to be highly self-correlated (*nearby things are more related than distant ones*)
  - ❑ E.g., neighborhood, temperature

Mining Spatial Associations: Progressive Refinement

- Hierarchy of spatial relationship:
  - *g_close_to*: *near_by*, *touch*, *intersect*, *contain*, etc.
  - Progressive refinement: First search for rough relationship and then refine it
- Two-step mining of spatial association:
  - Step 1: Rough spatial computation (as a filter)
    - Using MBR (Minimum Bounding Rectangle) or R-tree for rough estimation
  - Step2: Detailed spatial algorithm (as refinement)
    - Apply only to those objects which have passed the rough spatial association test (no less than *min_support*)
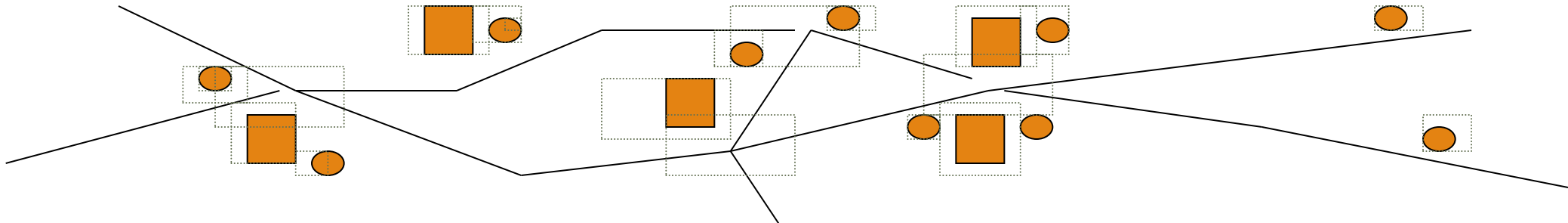
Mining Relative Movement Patterns

❑ **Flock:** At least m entities are within a circular region of *radius r* and move in the same direction

❑ **Convoy:** Uses *density-based clustering* at each timestamp; no need to be a rigid circle

❑ Flock and convoy: Both require k consecutive time stamps

❑ **Swarm:** Moving objects may not be close to each other for all the consecutive time stamps

  ❑ Efficient pattern mining algorithms for uncovering such swarm patterns
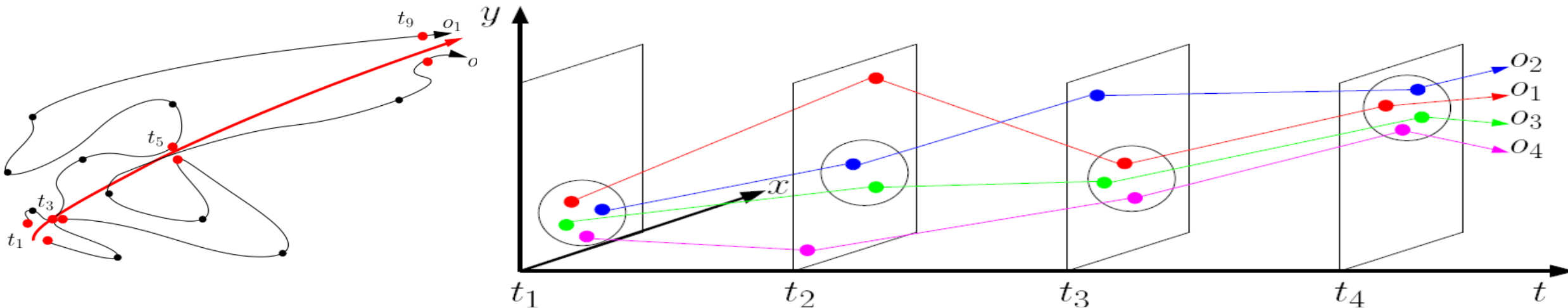
Mining Periodic Patterns with Sparse Data
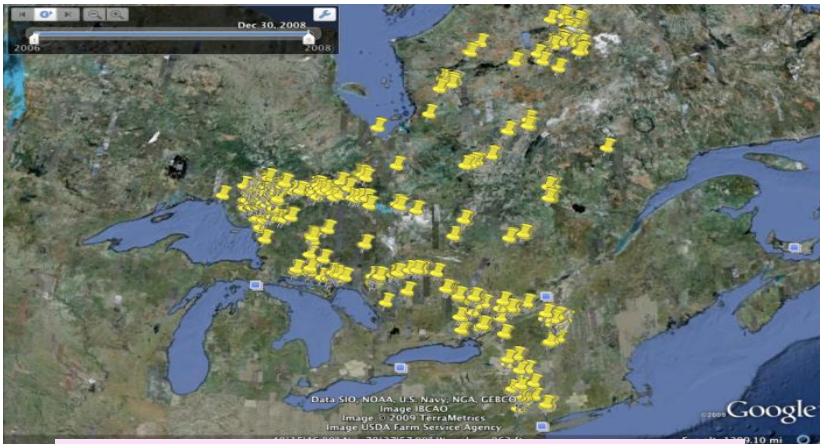
3-yr Bird migration data: very sparse

Jan.-Mar.   Apr.-June   July-.Oct   Nov.   Dec.
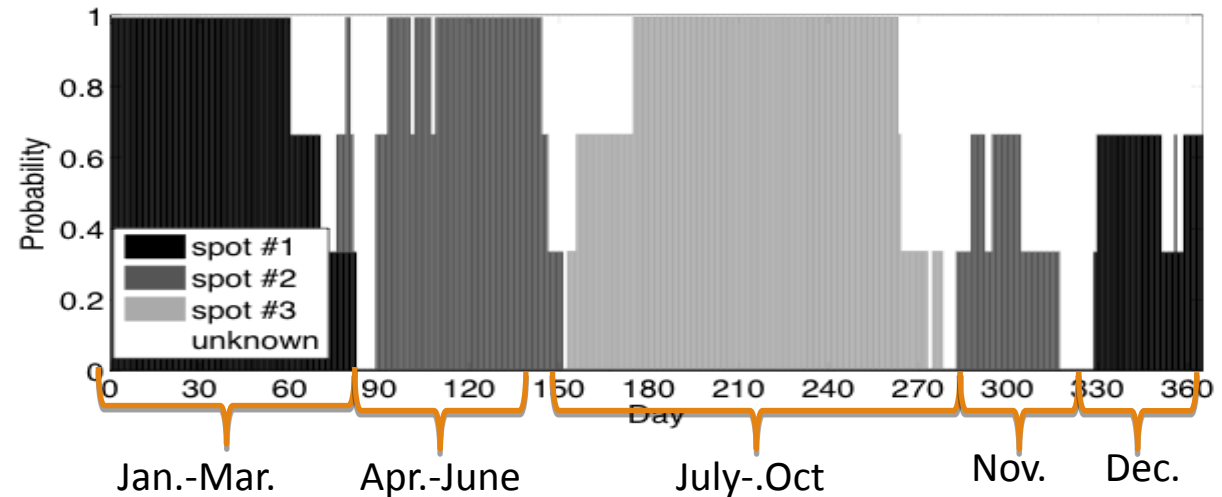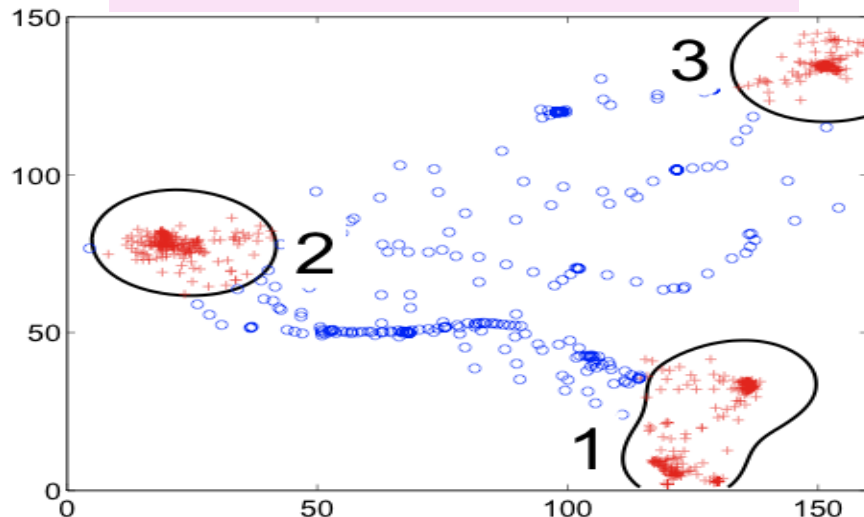
- ❑ **Detecting periods**: Cluster to find reference "points" and then detect multiple interleaved periods by Fourier Transform and auto-correlation
- ❑ **Summarizing periodic patterns:** By clustering and pattern discovery

Semantic Trajectory Pattern Mining

❑ Semantic trajectory: Trajectory carries semantics (e.g., category)
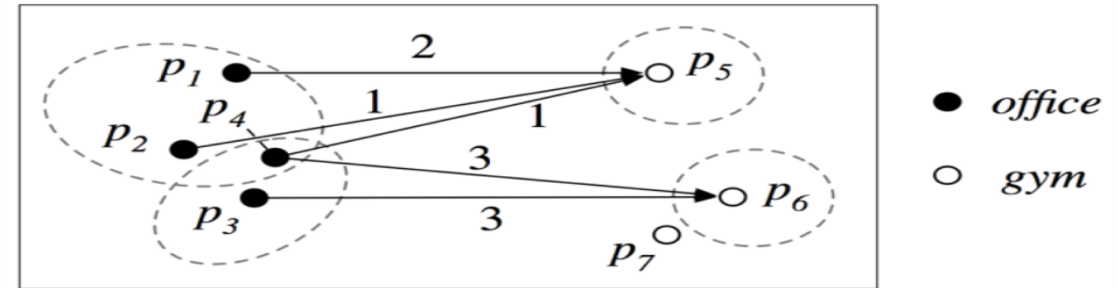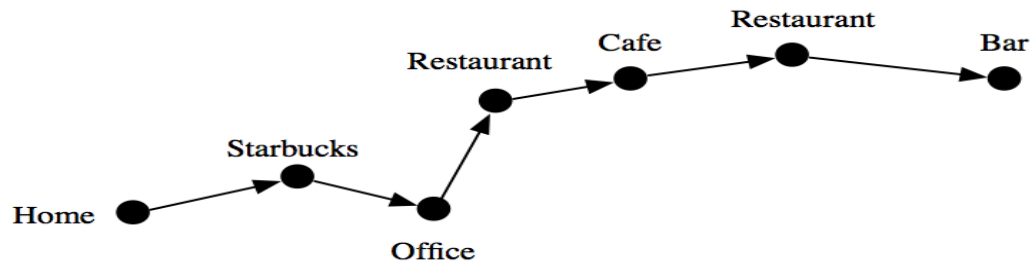
❑ Meaningful sequential patterns: 3 constraints: (i) semantic consistency; (ii) spatial compactness; and (iii) temporal continuity

❑ Method: A two-step approach

  ❑ 1: Mining coarse patterns that satisfy the semantic and temporal constraints (e.g., office → gym)

    ❑ First, mine semantically meaningful patterns (e.g., categories)

    ❑ Then, detect dense and compact clusters in the high-dimensional space

  ❑ 2: Splitting coarse patterns into fine-grained ones to meet the spatial constraint (e.g., people working in which office tend to go to which gym)

Recommended Readings
on Spatiotemporal and Trajectory Pattern Mining

❑ Y. Huang, S. Shekhar, H. Xiong, Discovering colocation patterns from spatial data sets: A general approach, IEEE Trans. on Knowledge and Data Engineering, 16(12), 2004

❑ K. Koperski, J. Han, "Discovery of Spatial Association Rules in Geographic Information Databases", SSD'95

❑ Z. Li, B. Ding, J. Han, R. Kays, "Swarm: Mining Relaxed Temporal Moving Object Clusters", VLDB'10

❑ Z. Li, B. Ding, J. Han, Roland Kays, Peter Nye, "Mining Periodic Behaviors for Moving Objects", KDD'10

❑ C. Zhang, J. Han, L. Shou, J. Lu, T. La Porta, "Splitter: Mining Fine-Grained Sequential Patterns in Semantic Trajectories", VLDB'14

❑ Y. Zheng and X. Zhou, Computing with Spatial Trajectories, Springer, 2011

Session 3. Pattern Discovery for Software Bug Mining

Pattern Discovery for Software Bug Mining

❑ Software is complex, and its runtime data is larger and more complex!

❑ Finding bugs is challenging: Often no clear specifications or properties; need substantial human efforts in analyzing data

❑ Software reliability analysis

    ❑ Static bug detection: Check the code

    ❑ Dynamic bug detection or testing: Run the code

    ❑ Debugging: Given symptoms or failures, pinpoint the bug locations in the code

❑ Why pattern mining?—Code or running sequences contain hidden patterns

    ❑ Common patterns → likely specification or property

    ❑ Violations (anomalies comparing to patterns) → likely bugs

    ❑ Mining patterns to narrow down the scope of inspection

        ❑ Code locations or predicates that happen more in failing runs but less in passing runs are suspicious bug locations

Typical Software Bug Detection Methods

- **Mining rules from source code**

  - Bugs as deviant behavior (e.g., by statistical analysis)

  - Mining programming rules (e.g., by frequent itemset mining)

  - Mining function precedence protocols (e.g., by frequent subsequence mining)

  - Revealing neglected conditions (e.g., by frequent itemset/subgraph mining)

- **Mining rules from revision histories**

  - By frequent itemset mining

- **Mining copy-paste patterns from source code**

- Find copy-paste bugs (e.g., CP-Miner [Li et al., OSDI'04]) (to be discussed here)

  - Reference: Z. Li, S. Lu, S. Myagmar, Y. Zhou, "CP-Miner: A Tool for Finding Copy-paste and Related Bugs in Operating System Code", OSDI'04

Mining Copy-and-Paste Bugs

- Copy-pasting is common
  - 12% in Linux file system
  - 19% in X Window system
- Copy-pasted code is error-prone
- Mine *"forget-to-change"* bugs by sequential pattern mining
  - Build a sequence database from source code
  - Mining sequential patterns
  - Finding mismatched identifier names & bugs

```c
void __init prom_meminit(void)
{
    ......
    for (i=0; i<n; i++) {
        total[i].adr = list[i].addr;
        total[i].bytes = list[i].size;
        total[i].more = &total[i+1];
    }
    ......
```

```c
for (i=0; i<n; i++) {
    taken[i].adr = list[i].addr;
    taken[i].bytes = list[i].size;
    taken[i].more = &total[i+1];
}
```

Code copy-and-pasted but **forget to change** "id"!

(Simplified example from *linux-2.6.6/arch/sparc/prom/memory.c*)

Building Sequence Database from Source Code

- Statement $\rightarrow$ (mapped to) number

- Tokenize each component

  - Different operators, constants, key words $\rightarrow$ different tokens

  - Same type of identifiers $\rightarrow$ same token

- Program $\rightarrow$ A long sequence

  - Cut the long sequence by blocks

Map a statement to a number

old = 3; new = 3;

Tokenize

5 61 20   5 61 20

Hash

16   **16**

Hash values

65	for (i=0; i<n; i++) {
16	total[i].adr = list[i].addr;
16	total[i].bytes = list[i].size;
71	total[i].more = &total[i+1];
	}
...	......
65	for (i=0; i<n; i++) {
16	taken[i].adr = list[i].addr;
16	taken[i].bytes = list[i].size;
71	taken[i].more = &total[i+1];
	}

Final sequence DB:
(65)
(16, 16, 71)
…
(65)
(16, 16, 71)
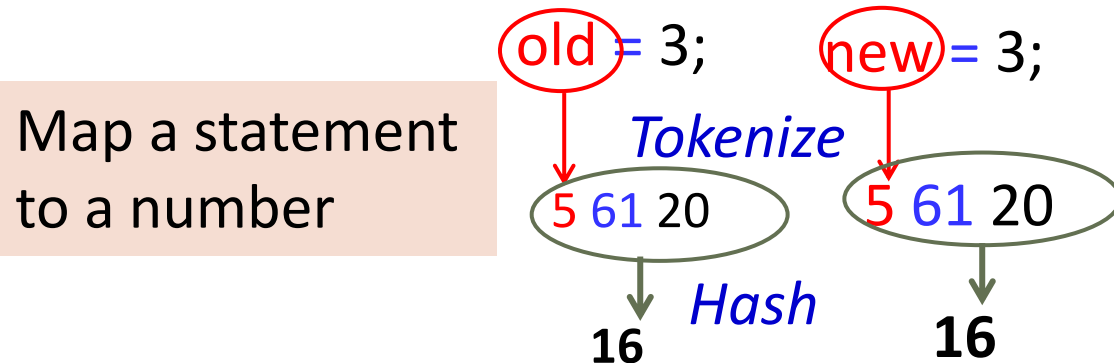
Sequential Pattern Mining & Detecting "Forget-to-Change" Bugs

❑ Modification to the *sequence pattern mining algorithm*

    ❑ Constrain the max gap

<div style="color:red">(16, 16, 71)
……
(16, 16, 10, 71)</div>

> Allow a maximal gap: inserting statements in copy-and-paste

❑ Composing Larger Copy-Pasted Segments

    ❑ Combine the neighboring copy-pasted segments repeatedly

```
f (a1);        f1 (b1);
f (a2);        f1 (b2);
f (a3);        f2 (b3);
```

conflict

❑ Find conflicts: Identify names that cannot be mapped to the corresponding ones

    ❑ E.g., 1 out of 4 "**total**" is unchanged, *unchanged ratio* = 0.25

    ❑ If $0 < unchanged\ ratio < threshold$, then report it as a bug

❑ CP-Miner reported many C-P bugs in Linux, Apache, … out of millions of LOC (lines of code)
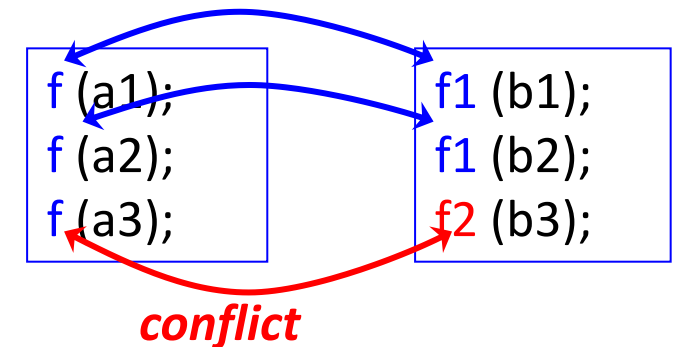
23

Courtesy of Yuanyuan Zhou@UCSD

Session 4. Pattern Discovery for Image Analysis

Image Representation for Visual Pattern Discovery

❑ An image can be characterized by visual primitives, e.g., interest points

  ❑ Each visual primitive can be described by visual feature, e.g., a high-dimensional feature vector

  ❑ Each image is a collection of visual primitives
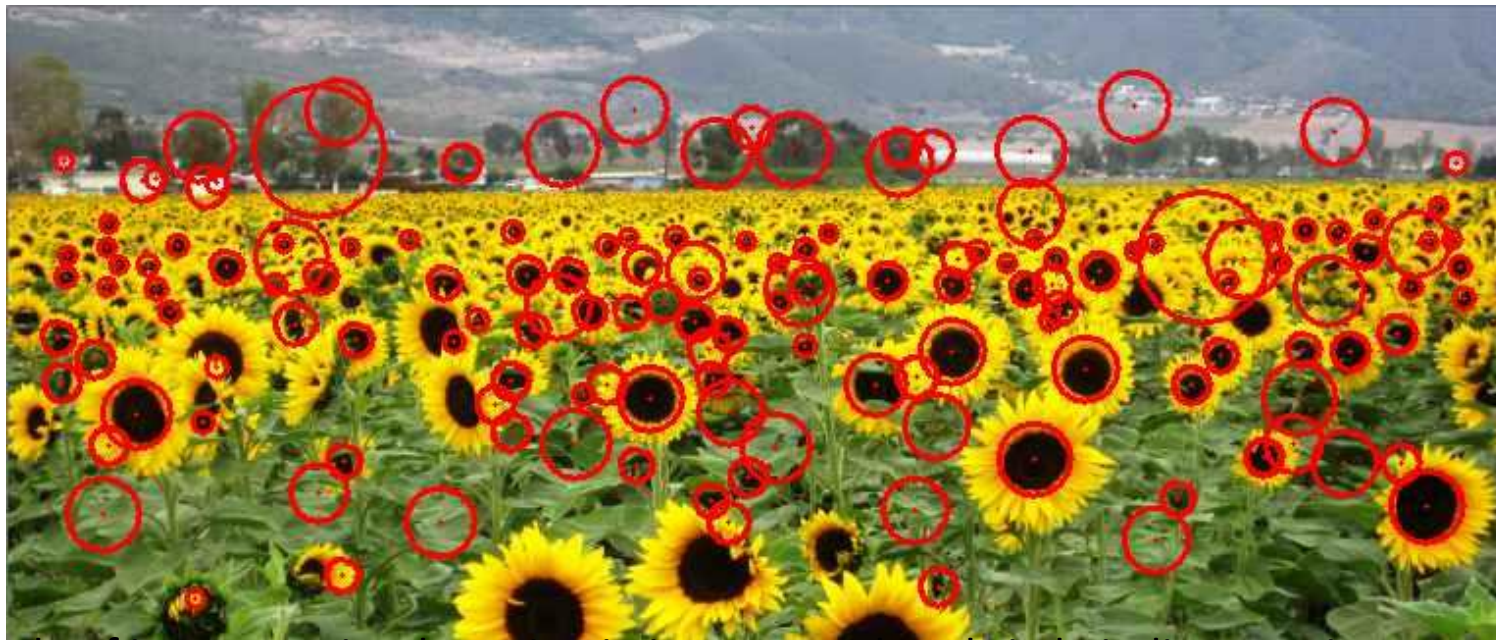
An example of interest point detection in images. Each red circle indicate an interest point.
Image courtesy from boofCV http://boofcv.org/index.php?title=Example_Detect_Interest_Points

Courtesy of Junsong Yuan @ NTU.SG, Ying Wu @Northwestern

Visual Patterns Discovery

Images

Feature Extraction

Visual Primitives

Feature Quantization

Induced Visual Transactions

Pattern Discovery

Visual Patterns
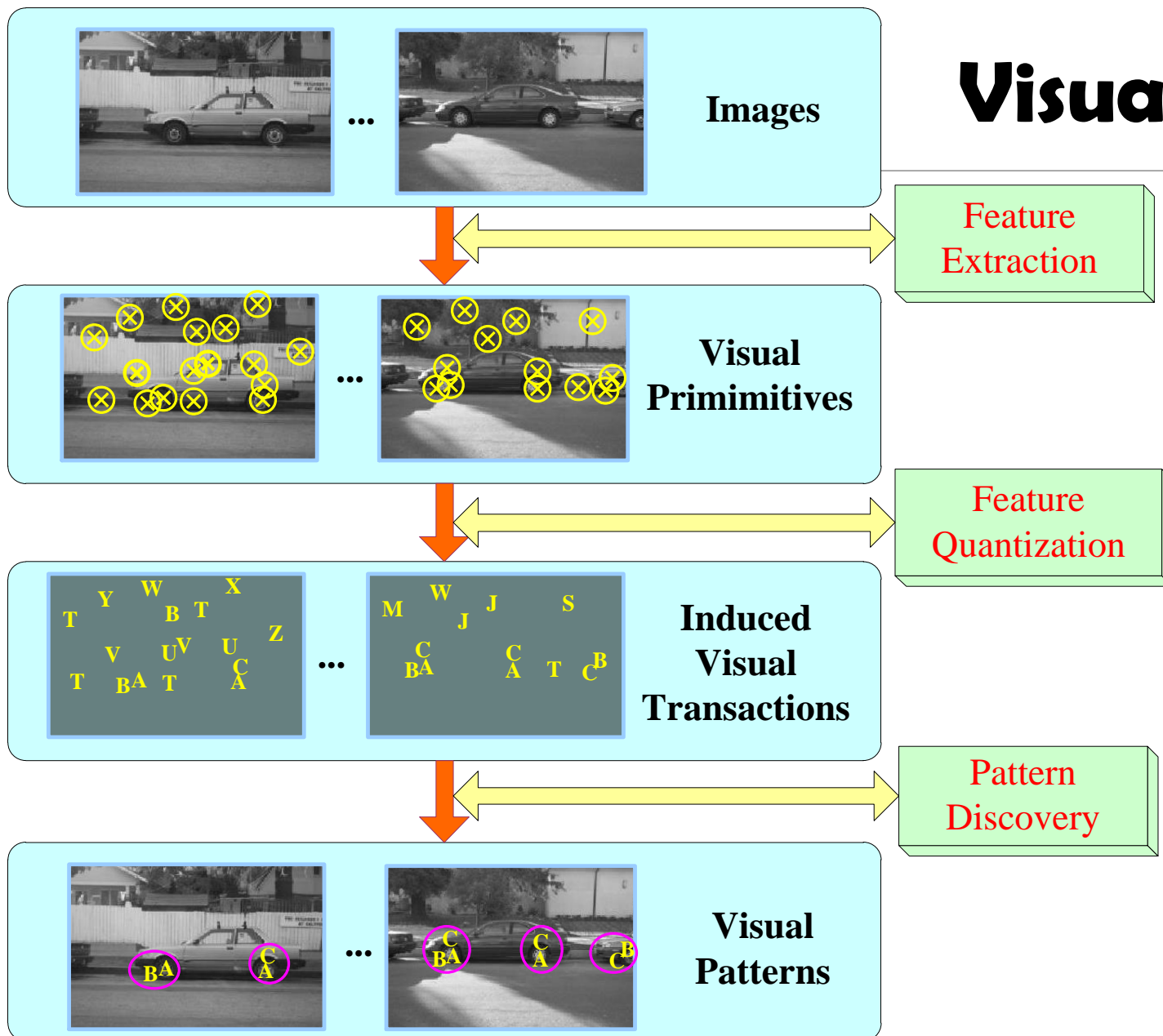
- ❑ Visual primitives can be clustered into visual "items"
  - ❑ Similar visual primitives belong to the same item
- ❑ Each visual primitive finds its k-nearest-neighbor in the image to form a visual "transaction"
  - ❑ An image can generate a number of transactions, i.e., induced visual transactions
- ❑ Mining "frequent itemsets" leads to semantically meaningful visual patterns

26

Challenges of Visual Pattern Discovery

ID	Transactions
G12:	**EDABC**
G13:	**HTABC**
G67:	**CSABK**
G78:	**ZTABM**
G79:	**KFABV**
G112:	**KSABT**
G113:	**KAABE**
G198:	**ILABO**
G215:	**TRABW**
G216:	**...**

AB
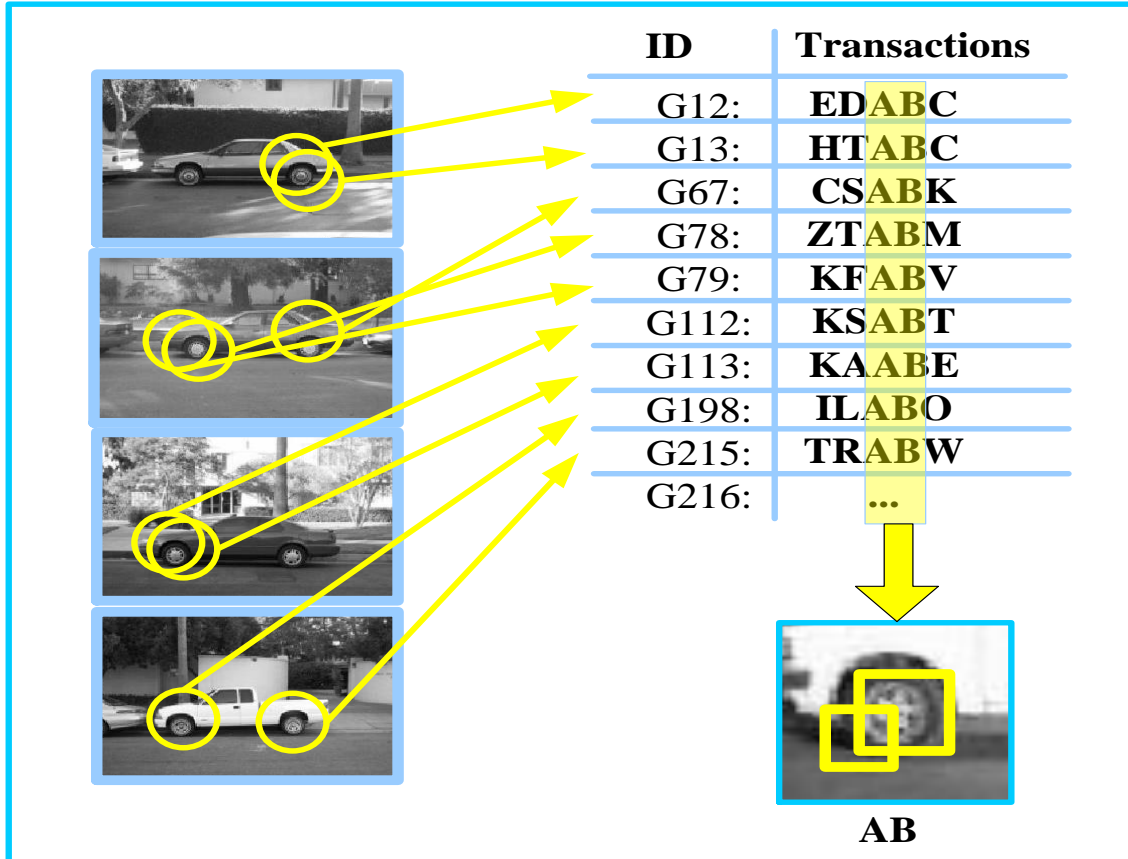
- Images are spatial data
  - Spatial configuration among the visual items matters
  - Induced transactions may overlap with each other, thus one needs to address the over counting problem
- Uncertainties of visual items and patterns
  - Noisy clustering of visual primitives into visual items affects visual pattern discovery
  - Visual synonym and polysemy

Recommended Readings

❏ Hongxing Wang, Gangqiang Zhao, Junsong Yuan, Visual pattern discovery in image and video data: a brief survey, Wiley Interdisciplinary Review: Data Mining and Knowledge Discovery 4(1): 24-37 (2014)

❏ Hongxing Wang, Junsong Yuan, Ying Wu, Context-Aware Discovery of Visual Co-Occurrence Patterns. IEEE Transactions on Image Processing 23(4): 1805-1819 (2014)

❏ Gangqiang Zhao, Junsong Yuan, Discovering Thematic Patterns in Videos via Cohesive Sub-graph Mining. ICDM 2011: 1260-1265

❏ Junsong Yuan, Ying Wu, Ming Yang, From frequent itemsets to semantically meaningful visual patterns. KDD 2007: 864-873

Session 5. Pattern Mining and Society: Privacy Issues

Pattern Mining and Society: Privacy Issues

❑ A potential adverse side-effect of data mining—Privacy could be compromised

   ❑ Privacy and accuracy are typically contradictory in nature

   ❑ Improving one often incurs a cost on the other

❑ Three categories on privacy issues arising out of data mining

   ❑ Input privacy (or data hiding)

     ❑ Distort or hide data to prevent the miners from reliably extracting confidential or private information

   ❑ Output privacy (or knowledge hiding)

     ❑ No disclosure of sensitive patterns or knowledge from datasets

   ❑ Owner privacy

     ❑ Does not allow any party to reliably learn the data or sensitive information that the other owners hold (i.e., the source of the data)

Ensuring Input Privacy

❑ Approach 1: Service provider anonymizes user's private information

    ❑ B2B (business-to-business) environment

       ❑ Service provider-to-data miner

    ❑ Do you really trust them?

❑ Approach 2: Data anonymized/perturbed at the data source itself

    ❑ B2C (business-to-customer) environment: Anonymized likely by a 3rd-party vendor

    ❑ Methods: Data perturbation, transformation, or hiding (hide sensitive attributes)

❑ K-anonymity privacy requirement and subsequent studies

    ❑ *k-anonymity*: Each equivalent class contains at least *k* records

    ❑ It is still not sufficient, thus leads to further studies, such as *ℓ-diversity*, *t-closeness*, and *differential privacy*

ID	ZIP	Age	Disease
1	61801	45	Heart
2	61848	49	Cancer
3	61815	41	Flu
4	61804	32	Diabetes
5	61802	38	Diabetes
6	61808	39	Flu

ID	ZIP	Age	Disease
1	618**	4*	Heart
2	618**	4*	Cancer
3	618**	4*	Flu
4	618**	3*	Diabetes
5	618**	3*	Diabetes
6	618**	3*	Flu

Data Perturbation for Privacy-Preserving Pattern Mining

❑ Statistical distortion: Using randomization algorithms

   ❑ Independent attribute perturbation: Values in each attribute perturbed independently

   ❑ Dependent attribute perturbation: Take care of correlations across attributes

❑ MASK [Rizvi & Haritsa VLDB'02]

   ❑ Flip each 0/1 bit with a probability p (Note: this may increase a lot of items)

   ❑ Tune p carefully to achieve acceptable average privacy and good accuracy

❑ Cut and paste (C&P) operator [Evfimievski et al. KDD'02]

   ❑ Uniform randomization: Each existing item in the real transaction is, with a probability p, *replaced* with a new item not present in the original transaction

   ❑ Methods developed on how to select items to improve the worst-case privacy

   ❑ Experiments show mining a C&P randomized DB correctly identifies 80-90% of "short" (length ≤ 3) frequent patterns, but how to effectively mine long patterns remains an open problem

Recommended Readings

- R. Agrawal and R. Srikant, Privacy-preserving data mining, SIGMOD'00

- C. C. Aggarwal and P. S. Yu, Privacy-Preserving Data Mining: Models and Algorithms, Springer, 2008

- C. Dwork and A. Roth. The Algorithmic Foundations of Differential Privacy. Foundations and Trends in Theoretical Computer Science. 2014

- A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In KDD'02

- A. Gkoulalas-Divanis, J. Haritsa and M. Kantarcioglu, Privacy in Association Rule Mining, in C. Aggarwal and J. Han (eds.), Frequent Pattern Mining, Springer, 2014 (Chapter 15)

- N. Li, T. Li, S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. ICDE'07

- A. Machanavajjhala, D. Kifer, J. Gehrke, M. Venkitasubramaniam, l-diversity: Privacy beyond k-anonymity, TKDD 2007

- S. Rizvi and J. Haritsa. Maintaining data privacy in association rule mining. VLDB'02

- J. Vaidya, C. W. Clifton and Y. M. Zhu, Privacy Preserving Data Mining, Springer, 2010

Session 6. Looking Forward

Looking Forward

- Lots of research issues on pattern discovery are still waiting to be solved
- Application exploration: **Invisible Pattern Mining** (i.e., built into various search, ranking, mining and other functional units)

   - Discriminative-pattern-based classification

   - Indexing and retrieval (e.g., graph indexing and similarity search)

   - Text mining: Phrase mining and topic modeling

   - Software bug mining and mining software specifications

   - Spatiotemporal and trajectory data mining

   - Image and multimedia data mining

   - Biological and chemical data analysis: DNA, graphs

   - Subspace clustering (to be covered in Clustering in Data Mining)

   - Web logs → Click stream patterns → Recommendation