TOP-K FREQUENT ITEMSETS
VIA DIFFERENTIALLY PRIVATE FP-TREES

AUGUST 26TH, 2014

Jaewoo Lee and Chris Clifton
Dept. of Computer Science / CERIAS
FREQUENT ITEMSET

• Task
  • \( I = \{I_1, I_2, \cdots, I_m\} \)
  • An itemset \( X \subseteq I \) is frequent if \( \sigma(X) \geq \tau \)
  • Given D, find top \( k \) frequent itemsets in D

• Challenge
  • Search space
  • Privacy budget
  • Search order

\[ \varepsilon = \sum_{i=1}^{n} \varepsilon_i \]

\[ \sigma(X_1) + noise \geq \tau ? \]
  Yes

\[ \sigma(X_2) + noise \geq \tau ? \]
  No

\[ \sigma(X_3) + noise \geq \tau ? \]
  No

\[ \vdots \]

\[ \vdots \]
## Our Method

- **NoisyCut**

<table>
<thead>
<tr>
<th>Phase</th>
<th>Frequent Itemset Discovery</th>
<th>Support Derivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 1</td>
<td>Find frequent itemsets without worrying about their supports</td>
<td>Given $\mathcal{L} = {X_1, ..., X_{</td>
</tr>
<tr>
<td></td>
<td>A variant of <em>sparse vector algorithm</em></td>
<td><strong>Sensitivity</strong> of insert operation?</td>
</tr>
<tr>
<td></td>
<td>$\sigma(X) + noise \geq \tau + noise$?</td>
<td>Imposing consistency</td>
</tr>
</tbody>
</table>
• Intuition
  • itemsets whose supports are far away from $\tau$ remain frequent/infrequent
  • blur the borderline

• Support of an itemset between two neighboring database
  • $\text{sup}_2(X) = \text{sup}_1(X)$
  • $\text{sup}_2(X) = \text{sup}_1(X) + 1$
Algorithm 1:

\[ \hat{\tau} = \tau + \text{Lap}(\lambda) \]

for each itemset \( X \) & \(|\{ i \mid v_i = 1\}| < k \)
if \( \text{sup}(X) + \text{Lap}(\lambda) \geq \hat{\tau} \) then
\[ v_i = 1 \]
else
\[ v_i = 0 \]

(frequent) (infrequent)

return \( v = (v_1, \ldots, v_t) \)
• Two neighboring database

$$\Pr[\mathcal{M}(D_1) = v] \leq e^\epsilon \frac{\Pr[\mathcal{M}(D_2) = v]}{\Pr[\mathcal{M}(D_1) = v]}$$
• Noisy support + Noisy Threshold

\[ \Pr_{D_1}[\nu_1 = 1, \nu_3 = 1] = \int_{-\infty}^{\infty} \Pr[\hat{\tau}_1 = x] \prod_i \Pr[\hat{\sigma}_1(X_i) \geq x] \, dx \]

If \( \sigma_1(X_i) = \sigma_2(X_i) \)

\[ \leq e^\epsilon \int_{-\infty}^{\infty} \Pr[\hat{\tau}_2 = x] \prod_i \Pr[\hat{\sigma}_2(X_i) \geq x] \, dx \]

If \( \sigma_2(X_i) = \sigma_1(X_i) + 1 \)

\[ \leq e^\epsilon \int_{-\infty}^{\infty} \Pr[\hat{\tau}_2 = x + 1] \prod_i \Pr[\hat{\sigma}_2(X_i) \geq x + 1] \, dx \]

\[ = \Pr_{D_2}[\nu_1 = 1, \nu_3 = 1] \]
Algorithm 2

- Given a set of large itemsets $\mathcal{L}$, find MFIs
- For each MFI $M$, build an FP-tree

```
<table>
<thead>
<tr>
<th>TID</th>
<th>Itemset</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{a, c, d, e}</td>
</tr>
<tr>
<td>2</td>
<td>{a, c, d}</td>
</tr>
<tr>
<td>3</td>
<td>{b, d, e}</td>
</tr>
<tr>
<td>4</td>
<td>{a, c}</td>
</tr>
<tr>
<td>5</td>
<td>{b, e}</td>
</tr>
</tbody>
</table>
```
• Update tree
  • for each \( t \in D, t' = t \cap M \)

TID | Itemset
---|---
1 | \{a, c, d, e\}
2 | \{a, c, d\}
3 | \{b, d, e\}
4 | \{a, c\}
5 | \{b, e\}
Noise propagation

- add children’s count to that of their parents
- remaining steps are the same with FP-growth algorithm
**CONSISTENCY**

- **Constraints**
  - \( x_p \geq \sum x_{child} \)
  - \( x_i \geq 0 \)
  - \( x_1 \geq x_4 + x_7 \)
  - \( x_2 \geq x_5 \)
  - \( x_4 \geq x_6 \)

\[ C\vec{x} \leq \vec{x} \]

minimize \( ||\vec{x} - \hat{x}||^2 \)

subject to \( C\vec{x} \leq \vec{x} \)
**EXPERIMENTS**

- **Datasets**

| Dataset       | |D| | |J| | max|t| | avg|t| |
|---------------|----------|--------|---------|---------|---------|---------|---------|
| mushroom      | 8,124    | 119    | 23      | 23      |
| pumsb star    | 49,046   | 2,088  | 63      | 50.5    |
| retail        | 88,162   | 16,470 | 76      | 10.3    |
| kosarak       | 990,002  | 41,270 | 2,498   | 8.1     |
| aol           | 647,377  | 2,290,685 | 48,070 | 34.9    |
| BMS-POS       | 515,597  | 1,657  | 164     | 6.5     |
| BMS-WebView1  | 59,602   | 497    | 267     | 2.5     |
| BMS-WebView2  | 77,512   | 3,340  | 161     | 5.0     |

- **Prior Solutions**
  - **PrivBasis (PB)**: approximate the longest frequent itemset based on frequent 1-itemset, 2-itemsets.
  - **SmartTrunc (ST)**: truncate the transactions to reduce the sensitivity
• F-Score=$2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$

(a) mushroom  (b) pumsb star  (c) retail

(d) BMS-POS  (e) BMS-WV1  (f) BMS-WV2
EXPERIMENTS

\[
\text{RE} = \text{median}_X \left( \frac{|\hat{\sigma}(X) - \sigma(X)|}{\sigma(X)} \right)
\]

(a) mushroom  (b) pumsb star  (c) retail
(d) BMS-POS  (e) BMS-WV1  (f) BMS-WV2
CONCLUSION

• Proposed algorithm
  • releases frequent itemsets with small noise
  • can also be useful to build a model for other data mining tasks (e.g., sequential pattern, clustering, etc.)