Stabbing the Sky: Efficient Skyline Computation over Sliding Windows

Outline

• Introduction
• n-of-N Queries
• \( (n_1, n_2) \)-of-N Queries
• Performance Evaluation
• Conclusions

Skyline

Skyline Query:
• Input: a set of points in \( d \)-dimensional space.
• Output: points not dominated by another point.

\((x_1, x_2, \ldots, x_d)\) dominates \((y_1, y_2, \ldots, y_d)\) iff \(x_i \leq y_i\) \((1 \leq i \leq d)\) \& \(\exists k, x_k < y_k\)

Applications

Multi-criteria decision making...

Stock Trading Example:
• What are the top deals?

Skyline Query Over Sliding Window

Stock Trading Example
• Top deals of a stock in the last 5 mins? last 4 mins, ...
• Top deals of a stock in the last 10K deals? ...

Queries:
• n-of-N model \((\forall n \leq N)\): the most recent \( n \) elements
• \((n_1, n_2)\)-of-N model
• One-time queries
• Continuous queries

Challenges

Insertions & deletions (possibly high speed).

On-line information
– memory requirement
– processing speed

Existing techniques do not support n-of-N:
[Borzsonyi et al (ICDE01), Tan et al (VLDB01), Kossman et al (VLDB 02), Papadias et al (SIGMOD03), Kapoor (SIAM J. comp00)]
– support the computation of whole dataset
– \(O(n \log^{d/2} n)\) for \(d \geq 4\) & \(O(n \log n)\) otherwise
Results

n-of-N:
• keep \(N'\) (\(N' \leq N\)) elements where \(N' = O \left( \log^d N \right)\) if data distribution on each dimension is independent.
• a novel encoding scheme, with \(O \left( N' \right)\) space, leads to n-of-N query time \(O \left( \log N' + s \right)\) instead of \(O \left( n \log^{d-2} n \right)\).
• a new trigger based technique for continuously processing an n-of-N query.
  – trigger update time: \(O \left( \log s \right)\).
  – result update time: \(O \left( \log \delta \right)\) where \(\delta\) is a result change.
(n_1, n_2)-of-N: similar results.

n-of-N Queries

• is redundant Point \(e\) in \(P_N\) (the most recent \(N\) elements) iff
  – \(e\) expires w.r.t \(P_N\), or
  – \(\exists e'\) s.t. \(e' \rightarrow e\), and \(e'\) is younger than \(e\).

Optimality

Theorem: Non-redundant Points (\(R_N\)) vs. n-of-N Skyline Query Result (\(Q_{n,N}\))
  – \((P_N - R_N)\) does not appear in any \(Q_{n,N}\)
  – \(Q_{n,N}\) must be a subset of \(R_N\)
  – \(\forall x \in R_N \rightarrow \exists n, x \in Q_{n,N}\)
  – \(R_N = O(\log^{d-1} N)\) for “independent” distributions
• Only need to keep \(R_N\) — the minimum number of elements to be kept.

Querying \(R_N\)

\(e \in Q_{n,N}\) iff
• \(e\) is a root in \(G_{R_N}\) or
• \(e' \rightarrow e\) in \(G_{R_N}\) & \(e'\) has expired \(\Leftrightarrow t(e') < M - n + 1 \leq t(e)\)

Querying \(R_N\): Optimal Algorithm

To answer an n-of-N Query, encode the \(G_{R_N}\) using intervals:
• Stab the intervals by \((M-n+1)\).
• For all returned intervals \((x,y)\), return point whose timestamp is \(y\)
• Technique: Use an interval tree index to achieve optimal \(O(\log |R_N| + s)\) query time
Maintaining $R_N$

new element $e_{\text{new}}$ arrives:
- If the oldest $e_{\text{old}} \in R_N$ expires, remove $e_{\text{old}}$ and update $R_N$ and $G_{\text{old}}$ (interval tree).
- find $D \subseteq R_N$ dominated by $e_{\text{new}}$, update $R_N$ and $G_{\text{new}}$
- Depth-first search on a R-tree of $R_N$
- find $e \not\in c_{\text{new}}$, update $G_{\text{new}}$
- Best-first search on the R-tree of $R_N$

Continuous n-of-N Query

Trigger-based algorithm:
- Deletion: $Q_{n,N} - \{e_{\text{old}}\}$ and $Q_{n,N} - \{D\}$
- Insertion: $Q_{n,N} \cup \{e_{\text{new}}\}$ if $\neg (\exists e' \not\in c_{\text{new}} \land t(e') \geq M-n+1)$
- Maintain a min-heap of $Q_{n,N}$ for efficiency

Continuous (n1,n2)-of-N Query

More complicated than n-of-N Query
- $P_N$ needs to be kept!
- (Old) critical dominance: $t(ae) = \max \{ t(e'): e' \rightarrow e \land t(e') < t(e) \}$
- backward critical dominance: $t(be) = \min \{ t(e'): e' \rightarrow e \land t(e') > t(e) \}$
- $e \in Q(n_1,n_2,N)$ iff $ae < M-n_2+1 \leq e \leq M-n_1+1 < be$
- $e_{\text{old}} = 3$, $D = \{6\}$, $e_{\text{new}} = 8$
- $e' = 4$
- $M = 8$, $n = 4$, $N = 5$
- $Q_{4,5} = \{4,7\}$, $Q_{5,5} = \{5,7,8\}$
- $M = 7$, $n = 4$, $N = 5$
- $Q_{4,5} = \{4,7\}$, $Q_{5,5} = \{5,7,8\}$

Processing (n1,n2)-of-N Query

Encode the CBC dominance graph:
- $e \leftrightarrow ((ae, e], be)$
- Build an interval tree on $(ae, e]$, only
- Stab using $M-n_2+1$ against the interval tree and check $e \leq M-n_1+1 < be$
- $O(\log N + s^*), \text{sub-optimal}$

More on (n1,n2)-of-N Query

Maintenance: Similar to that of n-of-N query, but
- Always expires the oldest element in $P_{\text{old}}$ and maintain the interval tree and the R-tree on $R_{\text{old}}$.
- Implementation-wise: Use two interval trees to index $R_{\text{new}}$ and $P_{\text{new}}$.

Continuous queries
- More complicated
  - A new skyline point might not be a skyline in the previous result.
  - not critically dominated by a skyline point in the previous result.
  - nor a newly arrived point.
  - Basic idea
  - Maintain additional Candidate Solutions (minimization) & triggers
  - Details in the full paper

Experiment Setup

- Hardware
  - P4 2.8G CPU, 1G Memory
- Datasets
  - Correlated, independent, and anti-correlated
  - $d = 2 \text{ to } 5$, $N = 10^6$
- Algorithms
  - KLP, nN, mnN, cnN, n12N, mn12N
- Metrics
  - Processing time => Streaming rate
n-of-N Query

- Varying dimensionality
  - \( M \) up to 2M, \( N = 1M, n \) uniformly from [1K, 1M], #queries = 1000

Maintenance Costs

2d and 5d datasets, measure average and max time, \( N = i \times 10^5 \)

Scalability

\( M \) (total number) = 2M, \( N = 1M, \) #queries = 2M

Continuous n-of-N Queries

- 2d & 5d datasets
- \( N = 10K \) and 1M
- 10 queries with \( n = r(N/10) \)
- measures cnN avg, cnN max, nN avg, nN max

(n1, n2)-of-N Queries

Varying dimensionality
- \( M \) up to 2M, \( N = 1M, \) #queries = 1000
- restricting \( n_1 - n_2 \geq 500 \)

Scalability

\( M \) = 2M, \( N = 1M, \) #queries = 2M
Maintenance

- 2d and 5d datasets
- measure average and max time
- \( N = i \times 10^5 \)

Conclusions

- Efficient algorithms for various sliding windows skyline queries
  - Keep only minimum number of points
  - Encode and index those points
  - Maintain all the data structures
- The proposed solutions
  - have theoretical guarantee on the performance, and
  - have demonstrated efficiency and scalability in the experiments
- Future work
  - Improve the current solution for \((n_1, n_2)\)-of-\(N\) queries
  - Approximate skyline queries

Q&A

Thank You!

Reference