Forward Decay: A Practical Time Decay Model for Streaming Systems

Graham Cormode, Vladislav Shkapenyuk, Divesh Srivastava
AT&T Labs - Research

Bojian Xu
Iowa State University
Outline

• Background: Time-decayed data stream processing

• Previous time-decay model: “backward decay”

• Our new time-decay model: “forward decay”

• Data stream processing under “forward decay” model
  • Data aggregation
  • Reservoir sampling
Data Stream Processing

- Example I: Maintain the number of distinct source IP addresses in the packets received in the last one hour on a network link

- Processing Requirements
  - One pass processing
  - Small workspace: poly-logarithmic in the size of data
  - Fast processing time per element
  - Approximate answers are OK
Time-decayed Data Stream Processing

- Old data are less important
  → Assign recent data with higher weights (or importance).

- Example: Maintain the number of distinct source IP addresses in the packets received in the last one hour on a network link

```
Packets whose weights are 0

Packets received in the last one hour

The most recently received packet
```
Stream Model & Decayed Weight

**Stream:** \( R = \{(v_1, t_1), (v_2, t_2), \ldots, (v_n, t_n)\} \)
- \( v_i \): value
- \( t_i \): timestamp at which \( v_i \) was created

Decayed weight: weight of each \((v, t)\) decays over time.
Outline

- Background: Time-decayed data stream processing
- Previous time-decay model: “backward decay”
- Our new time-decay model: “forward decay”
- Data stream processing under “forward decay” model
“Backward” Decay and Examples

Decayed weight of \((v, t)\) at time \(T\): \(f(T - t)\)

- \(f\) is a **non-increasing** function of age \((T - t)\)

- Sliding window decay with window size 24 hours
  
  \[
  f(x) = \begin{cases} 
  1, & \text{if } x \leq 24 \text{ hours} \\
  0, & \text{otherwise}
  \end{cases}
  \]

- Exponential decay: 
  
  \[
  f(x) = \frac{1}{2^x}
  \]

- Polynomial decay: 
  
  \[
  f(x) = \frac{1}{(x+1)^2}
  \]

- No decay: 
  
  \[
  f(x) \equiv 1
  \]
Example Time-decayed Aggregates Using Backward Decay

**Data stream:** $(v_1, t_1), (v_2, t_2), ...$

- **Count:** $\sum_i f(T - t_i)$
- **Sum:** $\sum_i v_i \cdot f(T - t_i)$
- **$\phi$-Heavy hitters:** $\left\{ v \mid \frac{\sum_{v_i=v} f(T - t_i)}{\sum_i f(T - t_i)} \geq \phi \right\}$
In General, Data Aggregation under Backward Decay is Hard, Space-wise

Even a simple time-decayed count $\sum f(T-t_i)$ can become very hard

- If $f$ is sliding window decay: need to store all data in the window [Datar et al '02]
- If $f$ is polynomial decay: need to store the whole stream [Cohen&Strauss '03]
Our Contribution: “Forward Decay”

- Supports time-decayed weight based data aggregation and sampling
- Has little overhead (space cost, CPU load) compared with no-decay
- Better scalability than backward decay
Outline

- Background: Time-decayed data stream processing
- Previous time-decay model: “backward decay”
- Our new time-decay model: “forward decay”
- Data stream processing under “forward decay” model
“Forward Decay” and Examples

Decayed weight of \((v, t)\) at time \(T\):
\[
\frac{g(t-L)}{g(T-L)}
\]
- \(g\) is a **non-decreasing** function
- \(L < \min\{t_i|i \geq 1\}\) is a pre-selected landmark

- **Exponential decay** (coincide with the backward exponential decay)
  \[
g(x) = \exp(x) \Rightarrow \frac{\exp(t-L)}{\exp(T-L)} = \exp(t - T)
\]

- **Polynomial decay**: \(g(x) = x^2 \Rightarrow \frac{g(t-L)}{g(T-L)} = \frac{(t-L)^2}{(T-L)^2}\)

- **No decay**: \(g(x) \equiv 1 \Rightarrow \frac{g(t-L)}{g(T-L)} \equiv 1\)
Advantages of “Forward Decay”

\[ \frac{1}{g(T-L)} \text{ is shared by all elements at any time.} \]

\[ g(t - L) \text{ is fixed after element } (v, t) \text{ is received.} \]

These two observations lead to efficient time-decayed data stream processing

Bojian Xu, Iowa State University
Outline

• Background: Time-decayed data stream processing

• Previous time-decay model: “backward decay”

• Our new time-decay model: “forward decay”

• Data stream processing under “forward decay” model
  • Data aggregation
  • Random sampling
Forward Time-decayed Data Aggregation

Count: \[ \sum \frac{g(t_i - L)}{g(T - L)} = \frac{1}{g(T - L)} \sum g(t_i - L) \]

Sum: \[ \sum v_i \frac{g(t_i - L)}{g(T - L)} = \frac{1}{g(T - L)} \sum v_i g(t_i - L) \]

\( \phi \)-heavy hitters:

\[ \left\{ v \mid \frac{\sum v_i g(t_i - L)}{\sum_i g(t_i - L) / g(T - L)} \geq \phi \right\} = \left\{ v \mid \frac{\sum v_i g(t_i - L)}{\sum_i g(t_i - L)} \geq \phi \right\} \]
Forward Time-decayed Data Aggregation

**Count:** \[ \sum \frac{g(t_i - L)}{g(T - L)} = \frac{1}{g(T - L)} \sum g(t_i - L) \]

**Sum:** \[ \sum \frac{v_i g(t_i - L)}{g(T - L)} = \frac{1}{g(T - L)} \sum v_i g(t_i - L) \]

\(\phi\)-heavy hitters:

\[ \left\{ v \mid \frac{\sum v_i \frac{g(t_i - L)}{g(T - L)}}{\sum v_i g(t_i - L)/g(T - L)} \geq \phi \right\} = \left\{ v \mid \frac{\sum v_i g(t_i - L)}{\sum v_i g(t_i - L)} \geq \phi \right\} \]
Forward Time-decayed Data Aggregation

Count: \[ \sum \frac{g(t_i-L)}{g(T-L)} = \frac{1}{g(T-L)} \sum g(t_i - L) \]

Sum: \[ \sum \frac{v_i g(t_i-L)}{g(T-L)} = \frac{1}{g(T-L)} \sum v_i g(t_i - L) \]

Can be maintained in one pass using small space

factoring out

\(\phi\)-heavy hitters:

\[ \left\{ v \left| \frac{\sum_{i:v_i=v} g(t_i-L)}{\sum_i g(t_i-L)} \geq \phi \right. \right\} = \left\{ v \left| \frac{\sum_{i:v_i=v} g(t_i-L)}{\sum_i g(t_i-L)} \geq \phi \right. \right\} \]
Forward Time-decayed Data Aggregation

Count: \[ \sum \frac{g(t_i - L)}{g(T-L)} = \frac{1}{g(T-L)} \sum g(t_i - L) \]

Sum: \[ \sum v_i \frac{g(t_i - L)}{g(T-L)} = \frac{1}{g(T-L)} \sum v_i g(t_i - L) \]

\(\phi\)-heavy hitters:

\[ \left\{ v \mid \frac{\sum_{i=v} g(t_i - L)}{\sum_i g(t_i - L)} \geq \phi \right\} = \left\{ v \mid \frac{\sum_{i=v} g(t_i - L)}{\sum_i g(t_i - L)} \geq \phi \right\} \]

Can be maintained in one pass using small space

factoring out

Cancelled out
Forward Time-decayed Data Aggregation

Count: \[ \sum \frac{g(t_i - L)}{g(T - L)} = \frac{1}{g(T - L)} \sum g(t_i - L) \]

Sum: \[ \sum v_i \frac{g(t_i - L)}{g(T - L)} = \frac{1}{g(T - L)} \sum v_i g(t_i - L) \]

\( \phi \)-heavy hitters:

\[ \left\{ v \left| \frac{\sum v_i = v g(t_i - L)}{\sum_i g(t_i - L)} \right. \geq \phi \right\} = \left\{ v \left| \frac{\sum v_i = v g(t_i - L)}{\sum_i g(t_i - L)} \right. \geq \phi \right\} \]

Can be maintained in one pass using small space

factoring out

Cancelled out

This is **static** weight based heavy hitters. Solutions are available in [Metwally, Agrawal & Abbadi'05, Cormode, Korn & Tirthapura'08]

Bojan Xu, Iowa State University
Time-decayed Weight Based Reservoir Sampling

- Requirements
  - Items of larger weights get higher chances to be selected
  - Fixed sample size - reservoir

- Static weight based reservoir sampling exists:
  - (Weighted) Reservoir sampling [Vitter'85, Efraimidis & Spirakis'06]
  - Priority sampling [Alon, Duffield, Lund & Thorup'05]

- What about time-decayed weight based reservoir sampling?
### Time-decayed Weight Based Reservoir Sampling

- **Requirements**
  - Items of larger weights get higher chances to be selected
  - Fixed sample size - reservoir

- **Static weight** based reservoir sampling exists:
  - (Weighted) Reservoir sampling [Vitter'85, Efraimidis & Spirakis'06]
  - Priority sampling [Alon, Duffield, Lund & Thorup'05]

- **What about time-decayed weight based reservoir sampling?**

- **In forward decay:**
  \[
  \frac{g(t-L)}{g(T-L)} \quad \text{static for any particular element (v,t)}
  \]
  
  \[
  \text{shared by all the elements and can be cancelled out}
  \]
Time-decayed Weight Based Reservoir Sampling

- **Requirements**
  - Items of larger weights get higher chances to be selected
  - Fixed sample size - reservoir

- **Static weight** based reservoir sampling exists:
  - (Weighted) Reservoir sampling [Vitter'85, Efraimidis & Spirakis'06]
  - Priority sampling [Alon, Duffield, Lund & Thorup'05]

- **What about time-decayed weight based reservoir sampling?**

- **In forward decay:**
  \[
  \frac{g(t-L)}{g(T-L)} \]

- **So, static weight based reservoir sampling algorithms can be directly used for any forward decay based weighted reservoir sampling.**

---

Bojian Xu, Iowa State University
Experimental Setup

- GS streaming system (formerly called Gigascope system)

- Compare performance/cost between:
  - Different decay models
    - no decay
    - Backward decay
    - Forward decay
  - Various aggregates and sampling
Experimental Results: Count, Sum

Forward Decay has little overhead and good scalability with data rate
Experimental Results: Heavy Hitters

Forward decay has little overhead and better scalability than backward decay.
Experimental Results: Sampling

- Forward decay has little overhead
- Exponential backward has little overhead, because it coincides to forward exponential decay
Summary: Forward Decay

- A new time decay model for temporal streaming data analysis
- Supports time-decayed weight based data aggregation and sampling
- Has little overhead (space cost, CPU load) compared with no-decay
- Better scalability than backward decay
Thank You!

Q & A