Space-efficient Online Approximation of Time Series Data: Streams, Amnesia, and Out-of-order

Luca Foschini

joint work with
Sorabh Gandhi and Subhash Suri

University of California Santa Barbara

ICDE 2010
Outline

1. Time Series Approximation
2. Contribution
3. Related Work
4. Algorithms and Analysis
5. Experimental Results
6. Conclusion and Future Work
Time Series

- ordered sequence of pairs

\[ S = \{(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\} \]
**Time Series**

- *ordered* sequence of pairs

\[ S = \{(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\} \]

- stream model: items arrive in order of timestamp \( t_i \)
Time Series

- *ordered* sequence of pairs

\[ S = \{(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\} \]

- stream model: items arrive in order of timestamp \( t_i \)

\[ v_1 \quad t_1 \quad v_2 \quad t_2 \quad v_3 \quad t_3 \quad v_4 \quad t_4 \quad v_5 \quad t_5 \quad v_6 \quad t_6 \quad v_7 \quad t_7 \quad v_8 \quad t_8 \quad v_9 \quad t_9 \quad v_{10} \quad t_{10} \]
Time Series

- *ordered* sequence of pairs
  
  \[ S = \{ (t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n) \} \]

- **stream model**: items arrive in order of timestamp \( t_i \)
  
  - hypothesis will be relaxed in the out-of-order model
Time Series

- *ordered* sequence of pairs

\[ S = \{(t_1, v_1), (t_2, v_2), \ldots, (t_n, v_n)\} \]

- **stream model**: items arrive in order of timestamp \( t_i \)
  - hypothesis will be relaxed in the out-of-order model

- **Goal**: maintain succinct representation of \( S \) online
Bucket Approximation

- $S$ partitioned into consecutive blocks $b_j$ called buckets.
Bucket Approximation

- $S$ partitioned into consecutive blocks $b_j$ called buckets.
- buckets approximated by a single value (PC), a line segment (PL), or other function segments.

Error of the approximation:

$$E_p = \left( \sum_{i=1}^{n} |v_i - \hat{v}_i|^p \right)^{1/p}$$
Bucket Approximation

- $S$ partitioned into consecutive blocks $b_j$ called buckets.
- Buckets approximated by a single value (PC), a line segment (PL), or other function segments.
- Error of the approximation: $L_p$ norm
  
  $$E_p = \left( \sum_{i=1}^{n} |v_i - \hat{v}_i|^p \right)^{1/p}$$
(α, β)-approximation

Optimal B-bucket partition

Given:
- a time series S
- a budget of B buckets
- a way to find the optimal approximation within a bucket

Online B-bucket partition is suboptimal
We content ourselves with a partition that:
- achieves no more than α times the optimal error with B buckets
- requires no more than βB buckets (α, β ⩾ 1)

Example
If the optimal partition for 10 buckets achieves error = 3, a (1.2, 1.5)-approximated partition achieves error ⩽ 3.6 using B’ ⩽ 15 buckets.
\[(\alpha, \beta)\text{-approximation}\]

**Optimal B-bucket partition**

Given:
- a time series \(S\)
- a budget of \(B\) buckets
- a way to find the optimal approximation within a bucket

Goal: find the partition of \(S\) into \(B\) buckets that minimizes the approximation error

Online \(B\)-bucket partition is suboptimal

We content ourselves with a partition that:
- achieves no more than \(\alpha\) times the optimal error with \(B\) buckets
- requires no more than \(\beta B\) buckets

\((\alpha, \beta) \geq 1\)

Example:
If the optimal partition for 10 buckets achieves error=3, a \((1.2, 1.5)\)-approximated partition achieves error \(\leq 3.6\) using \(B' \leq 15\) buckets.
(\(\alpha, \beta\))-approximation

Optimal B-bucket partition

Given:

- a time series \(S\)
- a budget of \(B\) buckets
- a way to find the optimal approximation within a bucket

Goal: find the partition of \(S\) into \(B\) buckets that minimizes the approximation error

Online B-bucket partition is suboptimal

We content ourselves with a partition that:

- achieves no more than \(\alpha\) times the optimal error with \(B\) buckets
- requires no more than \(\beta B\) buckets

\((\alpha, \beta \geq 1)\)

Example

If the optimal partition for 10 buckets achieves error=3, a \((1.2, 1.5)\)-approximated partition achieves error \(\leq 3.6\) using \(B' \leq 15\) buckets.
(α, β)-approximation

Optimal B-bucket partition

Given:
- a time series S
- a budget of B buckets
- a way to find the optimal approximation within a bucket

Goal: find the partition of S into B buckets that minimizes the approximation error

Online B-bucket partition is suboptimal

We content ourselves with a partition that:
- achieves no more than α times the optimal error with B buckets
- requires no more than βB buckets
(α, β ≥ 1)
(\(\alpha, \beta\))-approximation

**Optimal B-bucket partition**

**Given:**
- a time series \(S\)
- a budget of \(B\) buckets
- a way to find the optimal approximation within a bucket

**Goal:** find the partition of \(S\) into \(B\) buckets that minimizes the approximation error

**Online B-bucket partition is suboptimal**

We content ourselves with a partition that:
- achieves no more than \(\alpha\) times the optimal error with \(B\) buckets
- requires no more than \(\beta B\) buckets \((\alpha, \beta \geq 1)\)

**Example**

If the optimal partition for 10 buckets achieves error=3, a \((1.2, 1.5)\)-approximated partition achieves error \(\leq 3.6\) using \(B' \leq 15\) buckets.
Outline

1. Time Series Approximation
2. Contribution
3. Related Work
4. Algorithms and Analysis
5. Experimental Results
6. Conclusion and Future Work
A Generic Algorithmic Framework

- variation of well known greedy technique, but with provable memory-error bounds
Summary of Contribution

A Generic Algorithmic Framework

- variation of well known greedy technique, but with provable memory-error bounds
- achieves (2, 4) approximation for a broad class of error metrics.
A Generic Algorithmic Framework

- variation of well known greedy technique, but with provable memory-error bounds
- achieves (2, 4) approximation for a broad class of error metrics.
- query-ready (no post-processing needed)
A Generic Algorithmic Framework

- variation of well known *greedy* technique, but with provable memory-error bounds
- achieves (2, 4) approximation for a broad class of error metrics.
- query-ready (no post-processing needed)
- easy to implement, fast in update (\(O(\log B)\)) and \(O(B)\) storage for a broad class of error metrics.
Summary of Contribution

A Generic Algorithmic Framework

- variation of well known greedy technique, but with provable memory-error bounds
- achieves (2, 4) approximation for a broad class of error metrics.
- query-ready (no post-processing needed)
- easy to implement, fast in update (O(log B)) and O(B) storage for a broad class of error metrics.
- adaptable to other stream models:
  - amnesic (error tolerance depends on time)
  - out of order ((t_i, v_i) might arrive after (t_j, v_j), even if t_j > t_i)
Outline

1. Time Series Approximation
2. Contribution
3. Related Work
4. Algorithms and Analysis
5. Experimental Results
6. Conclusion and Future Work
Optimal Offline Algorithm

- dynamic programming, given in Jagadish et al. [1998], with time complexity $O(n^2B)$ and space $O(nB)$
- improved in Guha [2008] to $O(n^2B)$ time and $O(n)$ space.
Online Algorithms

- the best worst-case bounds due to Guha and his colleagues Gilbert et al. [2002], Guha [2008, 2009], Guha et al. [2001, 2006]
- for $L_2$ metric, $(1 + \epsilon, 1)$ approximation in $O(n + B^3\epsilon^{-2}\log^2 n)$ time with $O(B\epsilon^{-2}\log n \log \epsilon^{-1})$ space Guha [2008].
- improved in Guha [2009] to remove $\log n$ from space bounds.
- in Buragohain et al. [2007], lightweight algorithms for query-ready approximations, for $L_\infty$ error metric only
- survey in Guha [2008] for existing techniques.
Outline

1. Time Series Approximation
2. Contribution
3. Related Work
4. Algorithms and Analysis
5. Experimental Results
6. Conclusion and Future Work
Well-Behaved Error Metric

1. **Nonnegativity**: The error is always non-negative.

\[ e(S', b) \geq 0, \quad S' \subseteq S, \quad b \geq 1 \]  

(1)

2. **Monotonicity**: The error of approximating \( S_1 \cup S_2 \) using one bucket is \[ \geq \] the error in approximating \( S_1 \) and \( S_2 \) separately with one bucket each.

\[ e(S_1 \cup S_2, 1) \geq e(S_1, 1) + e(S_2, 1) \]  

(2)

3. **Additivity**: The error of a \( B \)-bucket approximation is the sum of the errors of the individual buckets. (works also for max)

\[ \]
**Generic-MinMerge (GM) Algorithm**

1: for next $v_i \in S$ do
2:  Allocate new bucket $u$
3:  if $|\mathcal{B}| \geq 4B$ then
4:   $\{b_a, b_b\} = \text{MINERR-ADJPAIR}(\mathcal{B})$
5:   $b_m = \text{MERGE}(b_a, b_b)$
6:   $\mathcal{B} = (\mathcal{B} \setminus \{b_a, b_b\}) \cup \{b_m\}$
7:  end if
8: end for

**Min-Merge Property**

$B$ obeys the Min-Merge property if for adjacent buckets $b_1$, $b_2$, $e(b_m) = \text{MERGE}(b_1, b_2) \geq \max_i e(b_i)$
**Genericial-MinMerge (GM) Algorithm**

1. **for** next \( v_i \in S \) **do**
2. Allocate new bucket \( u \)
3. **if** \( |B| \geq 4B \) **then**
4. \( \{b_a, b_b\} = \text{MINERR-ADJPAIR}(B) \)
5. \( b_m = \text{MERGE}(b_a, b_b) \)
6. \( B = (B \setminus \{b_a, b_b\}) \cup \{b_m\} \)
7. **end if**
8. **end for**

**Budget** \( B = 1 \)

Min-Merge Property

\( B \) obeys the Min-Merge property if for adjacent buckets \( b_1, b_2, \ldots, b_m \):

\[
e(\text{merge}(b_1, b_2)) \geq \max_i e(b_i)
\]
**Generic-MinMerge (GM) Algorithm**

1: **for** next $v_i \in S$ **do**
2: Allocate new bucket $u$
3: **if** $|\mathcal{B}| \geq 4\mathcal{B}$ **then**
4: \{b_a, b_b\} = MINERR-ADJPAIR(\mathcal{B})
5: $b_m = $ MERGE($b_a, b_b$)
6: $\mathcal{B} = (\mathcal{B} \setminus \{b_a, b_b\}) \cup \{b_m\}$
7: **end if**
8: **end for**

---

**Budget $B = 1$**

---

**Min-Merge Property**

$\mathcal{B}$ obeys the Min-Merge property if for adjacent buckets $b_1, b_2, e(b_m) = \text{MERGE}(b_1, b_2) \geq \max_i e(b_i)$
**Generic-MinMerge (GM) Algorithm**

1. **for** next $v_i \in S$ **do**
2. Allocate new bucket $u$
3. **if** $|B| \geq 4B$ **then**
4. $\{b_a, b_b\} = \text{MINERR-ADJPAIR}(B)$
5. $b_m = \text{MERGE}(b_a, b_b)$
6. $B = (B \setminus \{b_a, b_b\}) \cup \{b_m\}$
7. **end if**
8. **end for**
**Generic-MinMerge (GM) Algorithm**

```
1: for next \( v_i \in S \) do
2:   Allocate new bucket \( u \)
3:   if \( |B| \geq 4B \) then
4:     \( \{b_a, b_b\} = \text{MINERR-ADJPAIR}(B) \)
5:     \( b_m = \text{MERGE}(b_a, b_b) \)
6:     \( B = (B \setminus \{b_a, b_b\}) \cup \{b_m\} \)
7:   end if
8: end for
```
**Generic-MinMerge (GM) Algorithm**

1: **for** next $v_i \in S$ **do**
2: Allocate new bucket $u$
3: **if** $|\mathcal{B}| \geq 4\mathcal{B}$ **then**
4: \{ $b_a, b_b$ \} = MINERR-ADJPAIR($\mathcal{B}$)
5: $b_m = $ MERGE($b_a, b_b$)
6: $\mathcal{B} = (\mathcal{B} \setminus \{b_a, b_b\}) \cup \{b_m\}$
7: **end if**
8: **end for**

**Budget $\mathcal{B} = 1$**

---

Luca Foschini (UCSB)  
Time Series Approximation  
ICDE 2010
**GENERIC-MINMERGE (GM) Algorithm**

1: for next $v_i \in S$ do
2:   Allocate new bucket $u$
3:   if $|B| \geq 4B$ then
4:     $\{b_a, b_b\} = \text{MINERR-ADJPAIR}(B)$
5:     $b_m = \text{MERGE}(b_a, b_b)$
6:     $B = (B \setminus \{b_a, b_b\}) \cup \{b_m\}$
7:   end if
8: end for

**Budget $B = 1$**

**Min-Merge Property**

$B$ obeys the Min-Merge property if for adjacent buckets $b_1, b_2, e(b_m) = \text{MERGE}(b_1, b_2) \geq \max_i e(b_i)$
**Generic-MinMerge (GM) Algorithm**

**Generic-MinMerge**

1: for next \( v_i \in S \) do
2: Allocate new bucket \( u \)
3: if \(|B| \geq 4B\) then
4: \( \{b_a, b_b\} = \text{MINERR-ADJPAIR}(B) \)
5: \( b_m = \text{MERGE}(b_a, b_b) \)
6: \( B = (B \setminus \{b_a, b_b\}) \cup \{b_m\} \)
7: end if
8: end for

**Min-Merge Property**

\( B \) obeys the Min-Merge property if for adjacent buckets \( b_1, b_2, \)
\( e(b_m) = \text{MERGE}(b_1, b_2) \geq \max_i e(b_i) \)
Theorem

For any well-behaved error metric, \textsc{Generic-MinMerge} achieves a \((2, 4)\)-approximation.
Analysis of GM (memory-error tradeoff)

**Theorem**

For any well-behaved error metric, $\text{Generic-MinMerge}$ achieves a $(2, 4)$-approximation.

**Proof sketch**

For any well-behaved error metric, $\text{Generic-MinMerge}$ achieves a $(2, 4)$-approximation.
Analysis of GM (memory-error tradeoff)

**Theorem**

For any well-behaved error metric, \texttt{Generic-MinMerge} achieves a \((2, 4)\)-approximation.

**Proof sketch**

\[ \sum_x e(X) \leq E^* \]

Luca Foschini (UCSB)

Time Series Approximation

ICDE 2010 10 / 21
Analysis of GM (memory-error tradeoff)

Theorem

For any well-behaved error metric, Generic-MinMerge achieves a (2, 4)-approximation.

Proof sketch

\[ \sum_x e(x) \leq E^* \]
Analysis of GM (memory-error tradeoff)

**Theorem**

For any well-behaved error metric, **Generic-MinMerge** achieves a (2, 4)-approximation.

**Proof sketch**

\[
\sum_x e(X) \leq E^* \\
\text{charge each } O \text{ to adjacent } X \text{s} \\
\text{(3B } X \text{s are sufficient)}
\]
Theorem

For any well-behaved error metric, **Generic-MinMerge** achieves a (2, 4)-approximation.

Proof sketch

\[ \sum_x e(x) \leq E^* \]

charge each \( O \) to adjacent \( X \)s

\( 3B \ X \)s are sufficient

\[ \sum_O e(O) \leq E^* \]
Analysis of GM (memory-error tradeoff)

Theorem

For any well-behaved error metric, \textsc{Generic-MinMerge} achieves a \((2, 4)\)-approximation.

Proof sketch

\[
\sum_x e(X) \leq E^* \\
\sum_O e(O) \leq E^* \\
e(GM) \leq 2E^*
\]
Analysis of GM (runtime)

\{ b_a, b_b \} = \text{MINERR-ADJPAIR}(\mathcal{B})

b_m = \text{MERGE}(b_a, b_b)

runtime depends on \text{MINERR-ADJPAIR} and \text{MERGE}
Analysis of GM (runtime)

\[ \{b_a, b_b\} = \text{MINERR-ADJPAIR}(B) \]
\[ b_m = \text{MERGE}(b_a, b_b) \]

- runtime depends on MINERR-ADJPAIR and MERGE

- Requires computing the error for the union of two buckets \textit{without explicit knowledge} of the values in them.
Analysis of GM (runtime)

\[
\{b_a, b_b\} = \text{MINERR-ADJPAIR}(B)
\]
\[
b_m = \text{MERGE}(b_a, b_b)
\]

- Requires computing the error for the union of two buckets \textit{without explicit knowledge} of the values in them
- for L_2 error each bucket
  - stores the \textit{least squares line} fitting the set of values \((t_i, v_i)\) in the bucket
  - slope and intercept is maintained in \(O(1)\) space, \text{MINERR-ADJPAIR} and \text{MERGE} are \(O(1)\) time.

runtime depends on \text{MINERR-ADJPAIR} and \text{MERGE}
Analysis of GM (runtime)

\( \{ b_a, b_b \} = \text{MINERR-ADJPAIR}(B) \)

\( b_m = \text{MERGE}(b_a, b_b) \)

- Requires computing the error for the union of two buckets \textit{without explicit knowledge} of the values in them
- for L_2 error each bucket
  - stores the \textit{least squares line} fitting the set of values \((t_i, v_i)\) in the bucket
  - slope and intercept is maintained in \(O(1)\) space, \text{MINERR-ADJPAIR} and \text{MERGE} are \(O(1)\) time.

\textbf{Theorem}

\textit{We can compute a streaming} \((\sqrt{2}, 4)\) \textit{approximation of its B-bucket PC or PL histogram under the L_2 norm using} \(O(B)\) \textit{space and} \(O(\log B)\) \textit{update time for each value.}
Amnesic model

- recent data are approximated more faithfully than older data.
- the approximation error is allowed to grow with the age of the data.
recent data are approximated more faithfully than older data.

the approximation error is allowed to grow with the age of the data.

proposed in Palpanas et al. [2004], very good results in practice, no worst-case guarantees
Amnesic model

- recent data are approximated more faithfully than older data.
- the approximation error is allowed to grow with the age of the data.
- proposed in Palpanas et al. [2004], very good results in practice, no worst-case guarantees

Theorem

The Amnesic-Merge algorithm achieves a \((1 + \varepsilon, 3)\) approximation, uses \(O(\varepsilon^{-1/2}B^* \log(1/\varepsilon))\) space and \(O(\log B^* + \varepsilon^{-1/2} \log(1/\varepsilon))\) update time for each value, where \(B^*\) is the number of buckets in the optimal solution.
Out-of-Order Stream Model

- relaxation of ordering of timestamps
Out-of-Order Stream Model

- relaxation of ordering of timestamps
- motivated by the asynchrony of distributed data processing applications: Cormode et al. [2008], Li et al. [2008, 2007]
- must be tolerant of a small number of gaps in the arrival sequence.
Out-of-Order Stream Model

- relaxation of ordering of timestamps
- motivated by the asynchrony of distributed data processing applications: Cormode et al. [2008], Li et al. [2008, 2007]
- must be tolerant of a small number of *gaps* in the arrival sequence.
Out-of-Order Stream Model

- relaxation of ordering of timestamps
- motivated by the asynchrony of distributed data processing applications: Cormode et al. [2008], Li et al. [2008, 2007]
- must be tolerant of a small number of gaps in the arrival sequence.

many heuristics proposed in the literature: Babcock et al. [2002], Tucker et al. [2003], Busch and Tirthapura [2007], Cormode et al. [2008].
GM can be modified to perform 0, 1, or 2 merges to deal with gaps being filled.
GM can be modified to perform 0, 1, or 2 merges to deal with gaps being filled.

Summary of Results

<table>
<thead>
<tr>
<th>Error metric</th>
<th>approximation</th>
<th>update</th>
<th>working memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL L∞</td>
<td>$(1 + \varepsilon, 3)$</td>
<td>$O(\log B + \varepsilon^{-1/2} \log(1/\varepsilon))$</td>
<td>$O(\varepsilon^{-1/2} B \log(1/\varepsilon))$</td>
</tr>
<tr>
<td>PC L∞</td>
<td>$(1, 3)$</td>
<td>$O(\log B)$</td>
<td>$O(B)$</td>
</tr>
<tr>
<td>PC,PL L₂</td>
<td>$(\sqrt{2}, 5)$</td>
<td>$O(\log B)$</td>
<td>$O(B)$</td>
</tr>
</tbody>
</table>
Outline

1. Time Series Approximation
2. Contribution
3. Related Work
4. Algorithms and Analysis
5. Experimental Results
6. Conclusion and Future Work
Steamgen (Keogh et al. [2006]): The set has 9600 measurements of steam generator drum pressure.

Random (Keogh et al. [2006]): Random walk data, 65536 values.

DJIA: Dow Jones Industrial Average index, 25737 values.

Hum: 69652 values, representing humidity measurements every 30 seconds.

Temp: This set of 93861 temperature readings.

Precip: 1M annual gage-corrected precipitation recorded worldwide.
Simulations II

Implementation

- all algorithms implemented in Python or C++,
- source code available at www.cs.ucsb.edu/~foschini/files/ooodelay.zip
- System: GNU/Linux 64bit Intel Core Duo Processor (@ 2.00GHz) equipped with 2GB RAM.

Algorithms Implemented

- **Generic-MinMerge**, **Amnesic-Merge**, **Fragment-Merge**
- **optimal Dynamic Programming** described in Jagadish et al. [1998]
- **SpaceApxWaveHist Algorithm**: described in Guha [2008], currently the best theoretical approximation bound.
- **Optimal Amnesic Approximation**
PC (left), PL (right) approximation of 4K data, HUM dataset B-GM refers to the B-bucket approximation, 4B-GM refers to the 4B-bucket approximation (as needed for the theorem).
Per-item processing time vs. the size of the time stream, for $B = 1000, 5000, 9000$ for the PRECIP dataset.
The approximation error (left) of B-bucket GM is comparable to the B-bucket SPACEAPXWAVEHIST, with the latter given twice the working space.

GM is significantly faster, due to the $O(B^3)$ term in SPACEAPXWAVEHIST.
Results IV

Amnesic and Out-of-Order

- **Amnesic-Merge** vs. optimal, DJIA dataset.
- **Fragment-Merge** in the Bursty, Perturbation and the Ordered stream models for $L_2$ (left) and $L_\infty$ (right) error metrics.

Luca Foschini (UCSB)
Outline

1. Time Series Approximation
2. Contribution
3. Related Work
4. Algorithms and Analysis
5. Experimental Results
6. Conclusion and Future Work
Conclusion

- presented a generic framework for online approximation of time-series data for several models: data streams, amnesic, and out-of-order
- proved space-quality approximation bounds for a popular greedy merge scheme for commonly used error metrics, such as $L_2$ or $L_\infty$
- highly practical implementations, require very small memory footprints, and run extremely fast
• presented a generic framework for online approximation of time-series data for several models: data streams, amnesic, and out-of-order

• proved space-quality approximation bounds for a popular greedy merge scheme for commonly used error metrics, such as $L_2$ or $L_\infty$

• highly practical implementations, require very small memory footprints, and run extremely fast

Open problems and future work

• we show that GM algorithm achieves constant factor guarantees. One can show a $1 + \epsilon$ approximation to optimal $B$ bucket $L_2$ error in $O(B/\epsilon^2)$ space is also possible.

• make GM efficient for other metrics such as $L_1$
References


S. Guha. Tight results for clustering and summarizing data streams. In *Departmental Papers, Department of Computer and Information Science, University of Pennsylvania*, 2009. URL http://repository.upenn.edu/cis_papers/394/.


