Coresets for k-Segmentation of Streaming Data

{ Guy Rosman, Mikhail Volkov, Dan Feldman, John W. Fisher III and Daniela Rus } MIT / CSAIL

Introduction

We propose a new coreset for k-segmentation of high-dimensional data.
k-segment coreset P - approximates the data D such that for any k-segment f, $\text{cost}(P,f)$ is ε-approximately $\text{cost}(D,f)$.

- Insertion time per new observation and required memory is linear in both the dimension d of the data, and the number k of segments.
- Coreset in streaming mode (with $O(\log n)$ insertion time/space).
- Supports efficient k-segmentation (cost minimization), in addition to approximating k-segment costs.
- k-segment coresets for high-dimensional data - linearity in d allows k-segmentation of videos.

A New Coreset for High-Dimensional Segmentation

Existing algorithms for k-segmentation are at least quadratic in d, or cubic in k [2, 1].
α-approximation for the k-segment mean of P ($\alpha \geq 1$) is a k-segment f such that

$$\text{cost}(P,f) \leq \alpha \cdot \text{cost}(P,f^*).$$

(α,β)-approximation ($\alpha,\beta > 0$) for the k-segment mean is a $(k\cdot\beta)$-segment g such that

$$\text{cost}(P,g) \leq \alpha \cdot \text{cost}(P,f^*).$$

1-segment mean A sufficient statistic for the purpose of evaluating 1-segments on the data is given by SVD of the matrix $(X)_i = (1\ t_i\ (p_i)_1\ \cdots\ (p_i)_d)$
k-segment mean - a k-segment $f^* : \mathbb{R} \to \mathbb{R}^d$ that minimizes $\text{cost}(P,f) = \sum_{(t,p)\in P} \|(p - f(t))\|^2$,

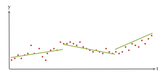


Figure 1: A k-segment for a 1D data

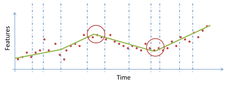


Figure 2: A balanced partition has few segments and small collision size.

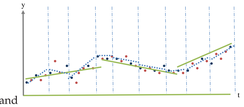


Figure 3: Weak coreset

Observation: The number of collisions is no more than k. Keeping coreset segments "small enough" allows us to bound the error.
Extension to k-segments For k segments, if we segment the data finely, there are no "big" segment collisions. This proposes:

- Estimate the complexity of the data by an (α,β)-approximation.
- Segment the data in a balanced way. Approximate each segment by SVD.

Weak Coreset

- Allows optimization with bounded error (not just cost estimation for solutions).
- Augmenting each coreset segment with sampling points, we get an ε approximation with $O(\frac{\log N}{\varepsilon})$ complexity increase.

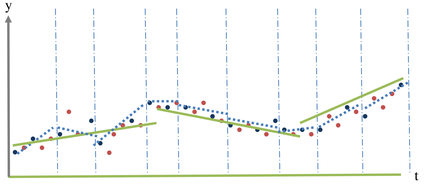


Figure 4: Weak coreset

References

[1] D. Feldman, A. Sugaya, and D. Rus. An effective coreset compression algorithm for large scale sensor networks. In *IPSN*, pages 257–268, 2012.

[2] D. Feldman, C. Sung, and D. Rus. The single pixel gps: learning big data signals from tiny coresets. In *Intl. Conf. on Advances in Geographic Information Systems*, pages 23–32. ACM, 2012.

[3] M. Volkov, G. Rosman, D. Feldman, J. W. F. III, and D. Rus. submitted paper. In *ICRA*, 2015.

Algorithm

- Run a *Bicriteria* algorithm, estimate the complexity of the data.
- Segment the data and approximate each segment using the *BalancedPartition* algorithm.

An approximation to the optimal cost can be constructed recursively and efficiently.

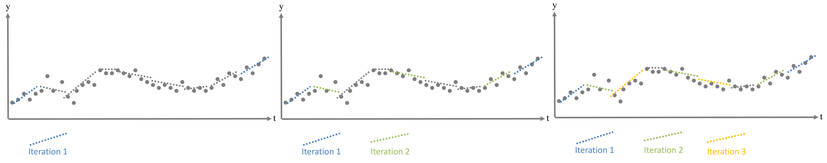


Figure 5: Recursively constructing a bicriteria appoximation for the cost.

Bicriteria

Input: A set $P \subseteq \mathbb{R}^{d+1}$ and an integer $k \geq 1$
Output: An $(O(\log n), O(\log n))$-approximation to the k-segment mean of P.
if $n \leq 2k+1$ **then**
  $f :=$ a 1-segment mean of P **return** f
Set $t_1 \leq \cdots \leq t_n$ and $p_1,\cdots,p_n \in \mathbb{R}^d$ such that $P = \{(t_1,p_1),\cdots,(t_n,p_n)\}$
Partition P into $4k$ sets such that for every $i \in [2k-1]$:

(i) $|\{t \mid (t,p) \in P_i\}| = \left\lfloor \frac{m}{4k} \right\rfloor$, $m = |P|$,and

(ii) segments are monotonous, not necessarily contiguous.

Compute an approximation g_i for each segment P_i
$Q :=$ the union of $k+1$ signals P_i with the smallest cost among $i \in [2k]$.
$h :=$ Bicriteria$(P \setminus Q, k)$
Set

$$f(t) := \begin{cases} g_i(t) & \exists (t,p) \in P_i \text{ such that } P_i \subseteq Q \\ h(t) & \text{otherwise} \end{cases}.$$

return f

BalancedPartition

Input: A set $P = \{(1,p_1),\cdots,(n,p_n)\}$ in $\mathbb{R}^{d+1}$
an error parameters $\varepsilon \in (0,1/10)$ and $\sigma > 0$.
Output: A set D of coreset segments.
$Q := \emptyset$; $D = \emptyset$;
p_{n+1}:= an arbitrary point in $\mathbb{R}^d$ **for** $i := 1$ *to* $n+1$ **do**
  $Q := Q \cup \{(i,p_i)\}$;
  $f^* :=$ a 2-approximation to the 1-segment mean of Q. $\lambda := \text{cost}(Q,f^*)$;
  if $\lambda > \sigma$ **or** $i = n+1$ **then**
    /* Define the new coreset segment data up to i */
    /* Use SVD to approximate the data */
    /* Start a new segment */
return D

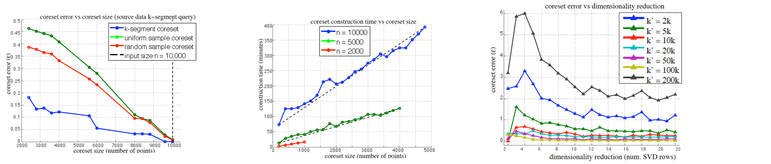


Figure 6: (a) Approximation error as a function of coreset size. (b) Coreset construction time vs. coreset size. (c) Reducing the dimension of per-segment approximation.

Streaming

Streaming coresets construction Given two coresets from consecutive times, they can be merged into a coreset. This can be shown by repeating the **Bicriteria** and the **BalancedPartition** algorithms in terms of coreset segments from the child coresets.

This can be used in a streaming mode to form a tree, computed in real-time. The tree can be augmented to allow fast retrieval and localization.

Large-scale, efficient, implementation We compress a 256,000 frame BOWs stream in 20 minutes on a 255 Amazon EC2 vCPU nodes cluster, demonstrating a near-perfect parallelism.

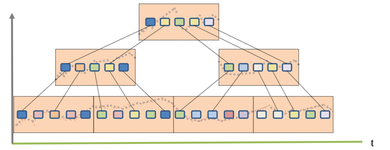
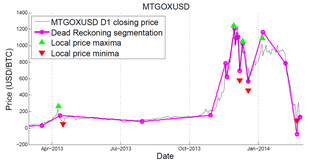
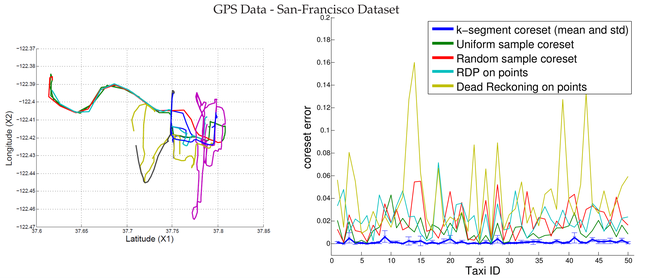


Figure 7: Streaming coresets

Results 1

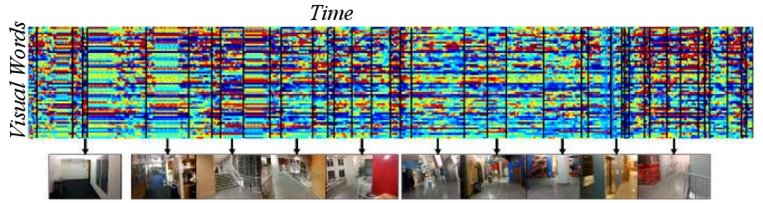
Average ε	Bitcoin data	GPS data
k-segment coreset	0.0092	0.0014
Uniform sample coreset	1.8726	0.0121
Random sample coreset	8.0110	0.0214
RDP on original data	0.0366	0.0231
RDP on k-segment	0.0335	0.0051
DeadRec on original data	0.0851	0.0417
DeadRec on k-segment	0.0619	0.0385





Results 2

- Bag-of-words represetation for each frame - a VQ of 5000 SURF+color descriptors.
- **Segmentation results** – ran our algorithm on a 3000 frames sequence from a wearable camera video, and compared to human annotators' segmentation. 25% improvement in RAND score compared to uniform.
- **Large-scale / performance** – real-time processing speeds on large videos (we tested on 6 hours of footage).



Acknowledgements

Guy Rosman was partially supported by MIT-Technion fellowship. Support for this research has been provided by Hon Hai/Foxconn Technology Group and MIT Lincoln Laboratory. The authors are grateful for this support.