Submodular function maximization

Hugo Jair Escalante

Outline

- Set functions
- Submodularity and monotonicity
- Maximization of submodular functions
- Applications: CSMMI
- Dealing with streams
- Experiments & results
- Discussion and final remarks

Set functions

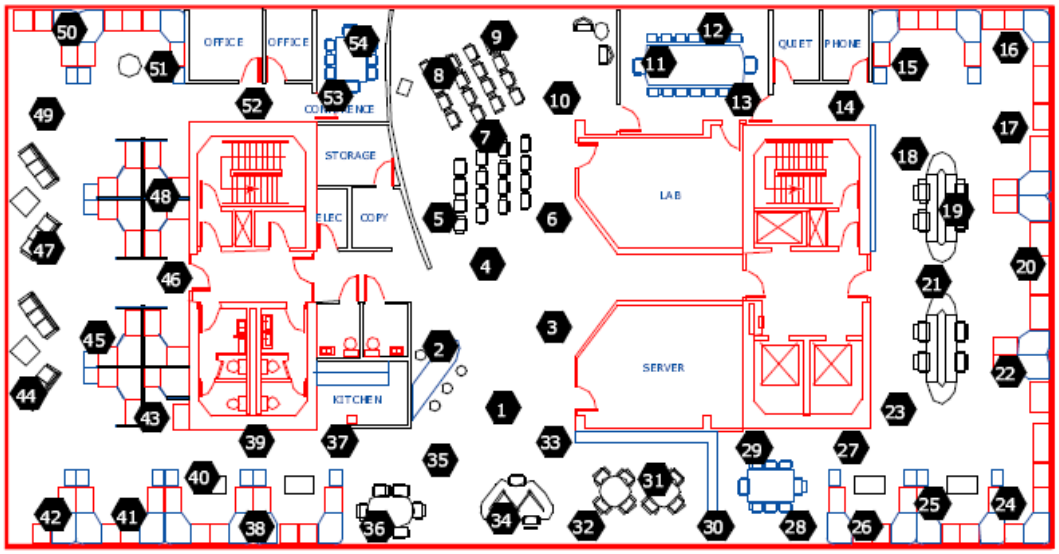
- Functions of the form:

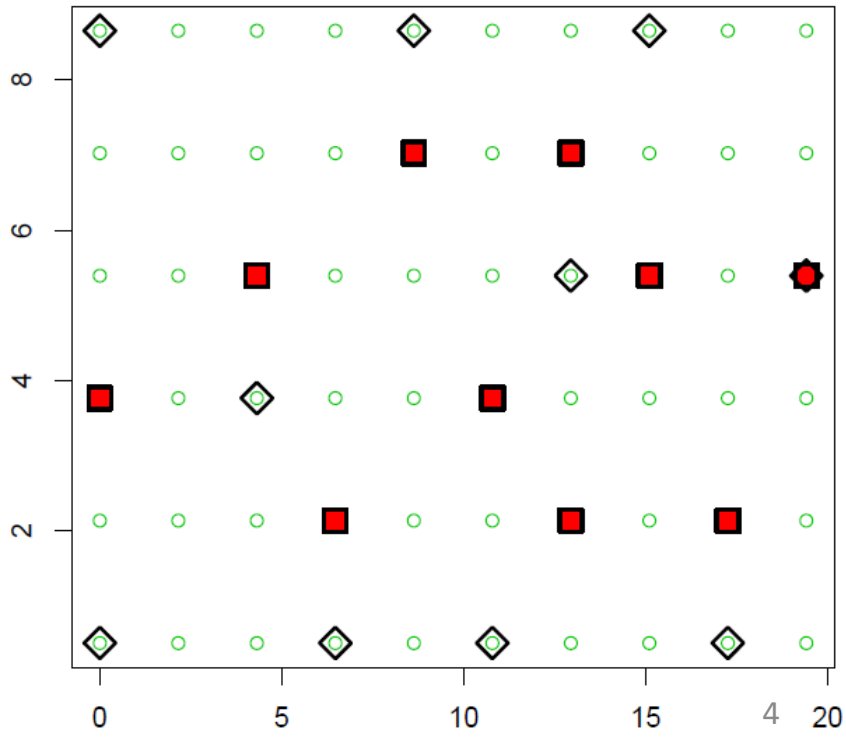
$$f : 2^V \rightarrow \mathbb{R}$$

  With V a finite set, and assuming $f(\Phi)=0$

- **Example:** consider the problem of placing sensors to cover some space represented by locations (V), and $f(S)$ the utility obtained when placing sensors at locations S

Set functions





Submodularity

Definition 1.1 (Discrete derivative) For a set function $f : 2^V \to \mathbb{R}$, $S \subseteq V$, and $e \in V$, let $\Delta_f(e \mid S) := f(S \cup \{e\}) - f(S)$ be the *discrete derivative* of f at S with respect to e.

Definition 1.2 (Submodularity) A function $f : 2^V \to \mathbb{R}$ is *submodular* if for every $A \subseteq B \subseteq V$ and $e \in V \setminus B$ it holds that

$$\Delta(e \mid A) \geq \Delta(e \mid B)\,.$$

Equivalently, a function $f : 2^V \to \mathbb{R}$ is *submodular* if for every $A, B \subseteq V$,

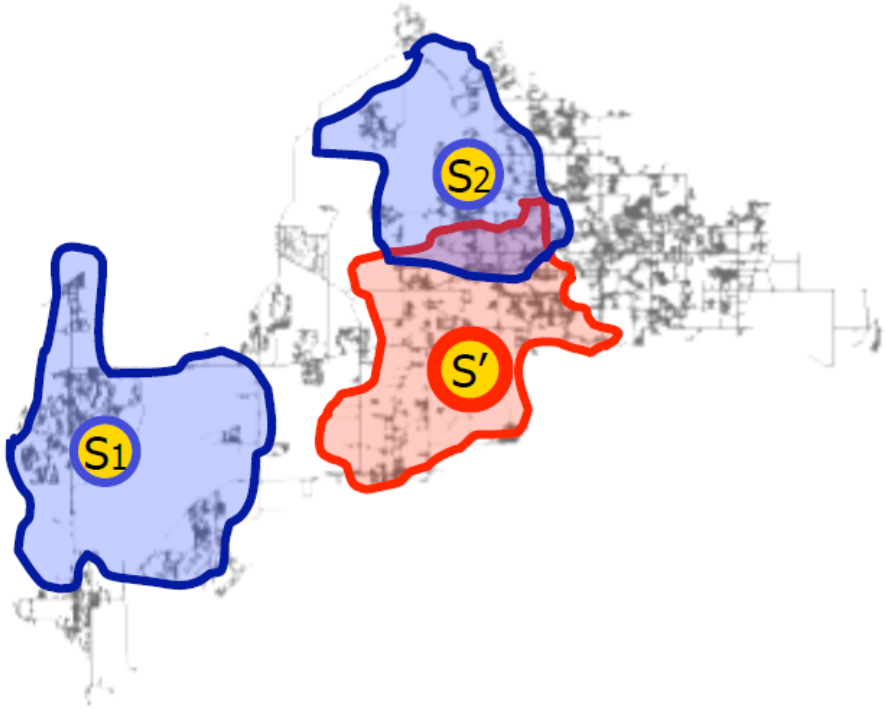
$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B).$$

Submodularity

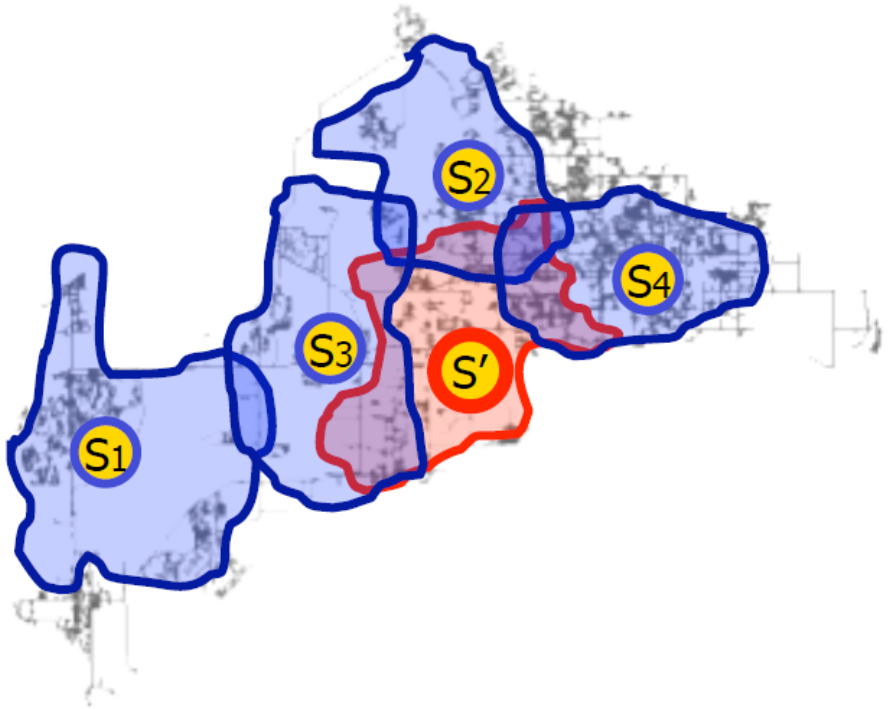
Definition 1.3 (Monotonicity) A function $f : 2^V \rightarrow \mathbb{R}$ is *monotone* if for every $A \subseteq B \subseteq V$, $f(A) \leq f(B)$.

f is called monotone iff for all e and S it holds that: $\triangle_f(e|S) \geq 0$

Submodular functions



(a) *Adding* s' *to set* $\{s_1, s_2\}$



(b) *Adding* s' *to superset* $\{s_1, \ldots, s_4\}$

$$\Delta(s' \mid \{s_1, s_2\}) \geq \Delta(s' \mid \{s_1, \ldots, s_4\})$$

Diminishing returns effect in the problem of placing sensors in a water distribution network to detect contaminants: *if more sensors are already placed there is more overlap, and less gain utility. Selecting any given element earlier helps more than selecting it later.*

Submodular functions

- Some submodular functions:
  - Modular functions

for all $A, B \subseteq V$ it holds that $f(A) + f(B) = f(A \cup B) + f(A \cap B)$. $f(S) = \sum_{e \in S} w(e)$

  - Weighted coverage functions

$$f(S) := g\Big(\bigcup_{v \in S} v\Big) = \sum_{x \in \bigcup_{v \in S} v} w(x), \qquad g(A) = |A|$$

  - Facility location

$$f(S) = \sum_{i=1}^{m} \max_{j \in S} M_{i,j}.$$

  - Entropy

$$H(\mathbf{X}_S) = -\sum_{\mathbf{x}_S} P(\mathbf{x}_S) \log_2 P(\mathbf{x}_S)$$

  - Mutual information

$$f(S) = I(\mathbf{Y}; \mathbf{X}_S) = H(\mathbf{Y}) - H(\mathbf{Y} \mid \mathbf{X}_S)$$

Maximization of submodular functions

- Submodular functions arise in several domains and problems (e.g., max-coverage, facility location, mutual information, …)

- Consider the problem of **data summarization**: *selecting representative subsets of manageable size out of large data sets*
  - *Exemplar-based clustering, document and corpus summarization, recomender systems, active set selection*

Maximization of submodular functions

- Many data summarization tasks can be formulated as:

$$\max_{S \subseteq V} f(S) \quad \text{s.t.} \quad |S| \leq k$$

- Let S* denote the optimal solution, with value:

$$\texttt{OPT} = f(S^*)$$

- This problem is NP-hard for many classes of submodular functions.

f(*S*) a black-box utility function, for now

Maximization of submodular functions

- Nemhauser et al. showed that a simple greedy (polynomial time) algorithm is highly effective:

  - Start with the empty set $S_0=\Phi$
  - Iterate k-times over the whole data set:

$$S_i = S_{i-1} \cup \{\arg\max_{e \in V} \triangle_f(e|S_{i-1})\}$$

- Nemhauser et al. proved that: $f(S^g) \geq (1-1/e)\mathtt{OPT}$

  (the solution is provably within 63% of the optimal?)

For several classes of monotone submodular functions it is known that this is the best approximation guarantee that one can hope.

Maximization of submodular functions

- Proof: Nemhauser et al. 1978

Many data summarization

- Sample applications:
  - Outbreak detection
  - News article recommendation
  - Non-parametric learning
  - Document and corpus summarization
  - Network inference
  - Viral marketing
  - …

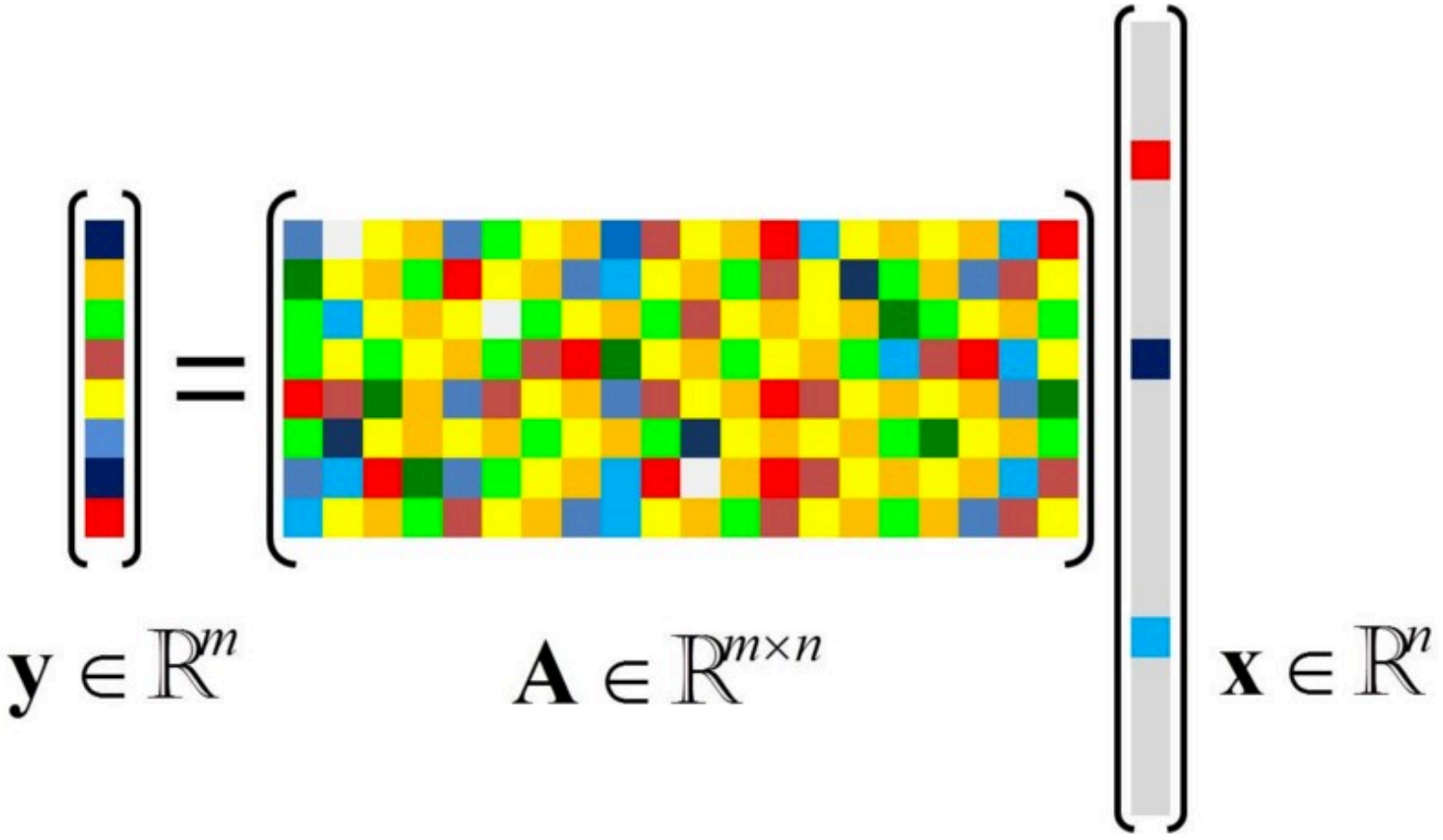
Application: CSMMI

- **Problem:** to learn a dictionary of codewords to be used for the SC representation of videos for action and gesture recognition

Jun Wan, Vassilis Athitsos, Pat Jangyodsuk, Hugo Jair Escalante, Qiuqi Ruan, Isabelle Guyon. **CSMMI: Class-Specific Maximization of Mutual Information for Action and Gesture Recognition.** IEEE Trans. Image Processing, Vol 23(7):3152--3165

Application: CSMMI

- Sparse representations: Idea, to learn a dictionary



Application: CSMMI

- Sparse representations: Idea, to learn a dictionary

$$\Phi_i^0 = [\phi_1, \ldots \phi_j, \ldots \phi_K], \phi_j \in \Re^n$$

- Such that:

$$\min_{\Phi_i^0, X_{\Phi_i^0}} \{\|Y_i - \Phi_i^0 X_{\Phi_i^0}\|_F^2\} \quad s.t. \ \|x_j\|_0 \leq T$$

Application: CSMMI

- The reconstruction error with class specific dictionaries is used as classifier

$$i_Y = \underset{i \in [1,2,\ldots,C]}{\arg\min} \; \| Y - \Phi_i^* \widehat{X_{Y_i}} \|_2^2$$

$$\widehat{X_{Y_i}} = \underset{X_{Y_i}}{\arg\min} \; \| Y - \Phi_i^* X_{Y_i} \|_F^2 \quad s.t. \; \| x \|_0 \leq T$$

Application: CSMMI

- Related works, learn dictionaries regardless of the classes (starting from KSVD)

- CSMMI: to learn class specific dictionaries, maximizing MI

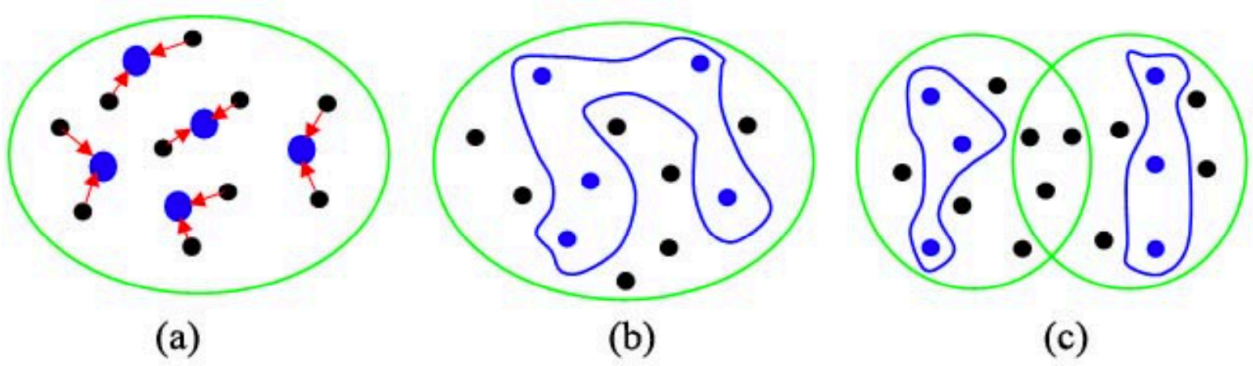


Fig. 1. (a) *Liu-Shah* [6]; (b) *Qiu-Jiang* [3], (c) *CSMMI* (our method). Each green circle denotes the region of an initial dictionary. The black points denote the initial dictionary items and the blue points represent the selected dictionary items. In the methods of *Liu-Shah* and *Qiu-Jiang*, the shared dictionary makes it difficult to distinguish which dictionary item is important to a specific class.

Application: CSMMI

- Key idea:
  1. Learn class specific dictionaries of size K (KSVD)

$$\Phi^0 = [\Phi_1^0, \Phi_2^0, \ldots, \Phi_C^0]. \quad \Phi_i^0 = [\phi_1, \ldots \phi_j, \ldots \phi_K], \phi_j \in \Re^n$$

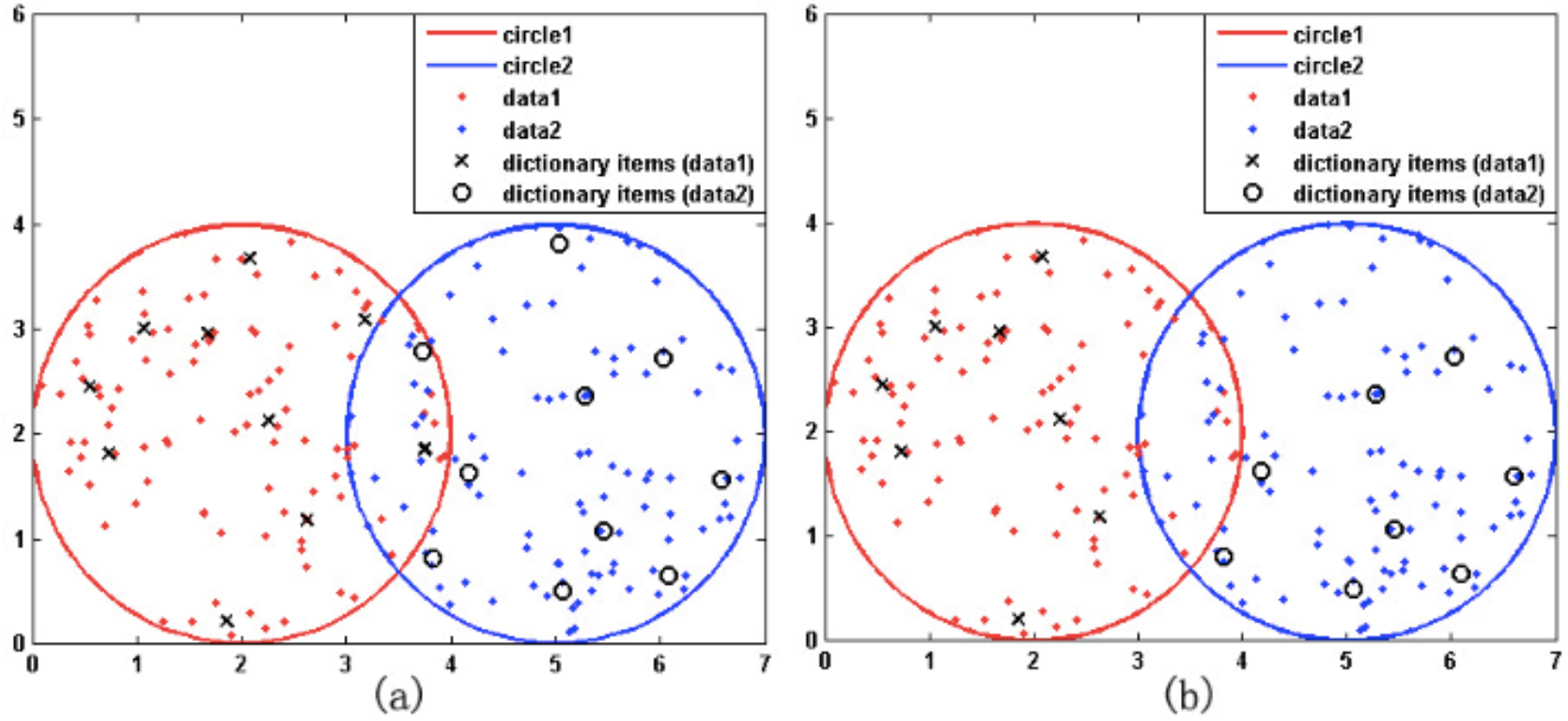
  2. Select a subset of codewords , k<K for each class

$$\Phi_1^*, \Phi_2^*, \ldots, \Phi_C^* \ (|\Phi_i^*| = k, \ k < K).$$

$$\arg\max_{\phi_i \in \Phi_i^0 \setminus \Phi_i^*} \overbrace{I(\Phi_i^* \cup \phi_i; \Phi_i^0 \setminus (\Phi_i^* \cup \phi_i)) - I(\Phi_i^*; \Phi_i^0 \setminus \Phi_i^*)}^{intra-class\ MI\ term(\tau_1)}$$

$$\overbrace{-[I(\Phi_i^* \cup \phi_i; \Phi^0 \setminus \Phi_i^0) - I(\Phi_i^*; \Phi^0 \setminus \Phi_i^0)]}^{inter-class\ MI\ term(\tau_2)} \quad (3)$$

Application: CSMMI



Application: CSMMI

- Experimental results

COMPARISON ON THE WEIZMANN ACTION DATASET

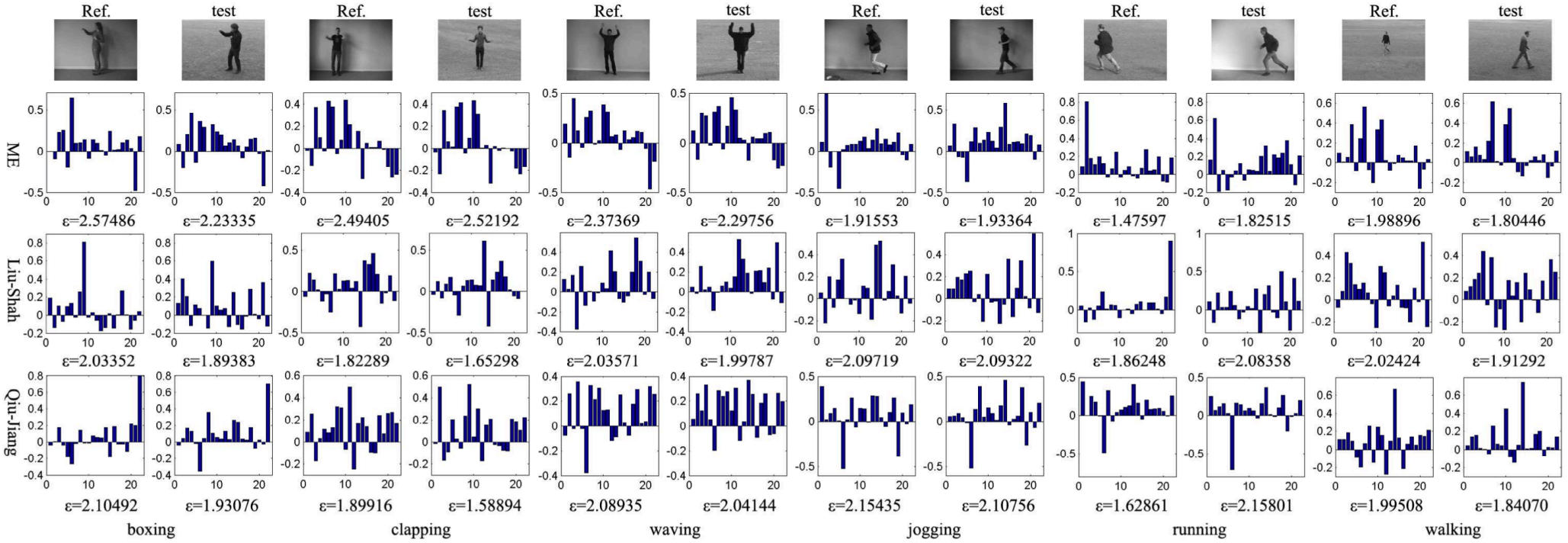
Papers	Methods	Dictionary Size	Average Accuracy
[38]	Space-time shape	—	97.83%
[39]	Multiple instance learning +kinematic feature	—	95.75%
[37]	Sparse linear approximation + feature covariance matrices	—	100%
[26]	prototype trees	—	100%
[22]	pLSA+cuboid	1200	90%
[1]	Concatenated dictionary +LMP	256	98.9%
[40]	Self-Similarities	—	95.3%
our method	*CSMMI*+STIP	140	**100%**

COMPARISON ON THE KTH ACTION DATASET

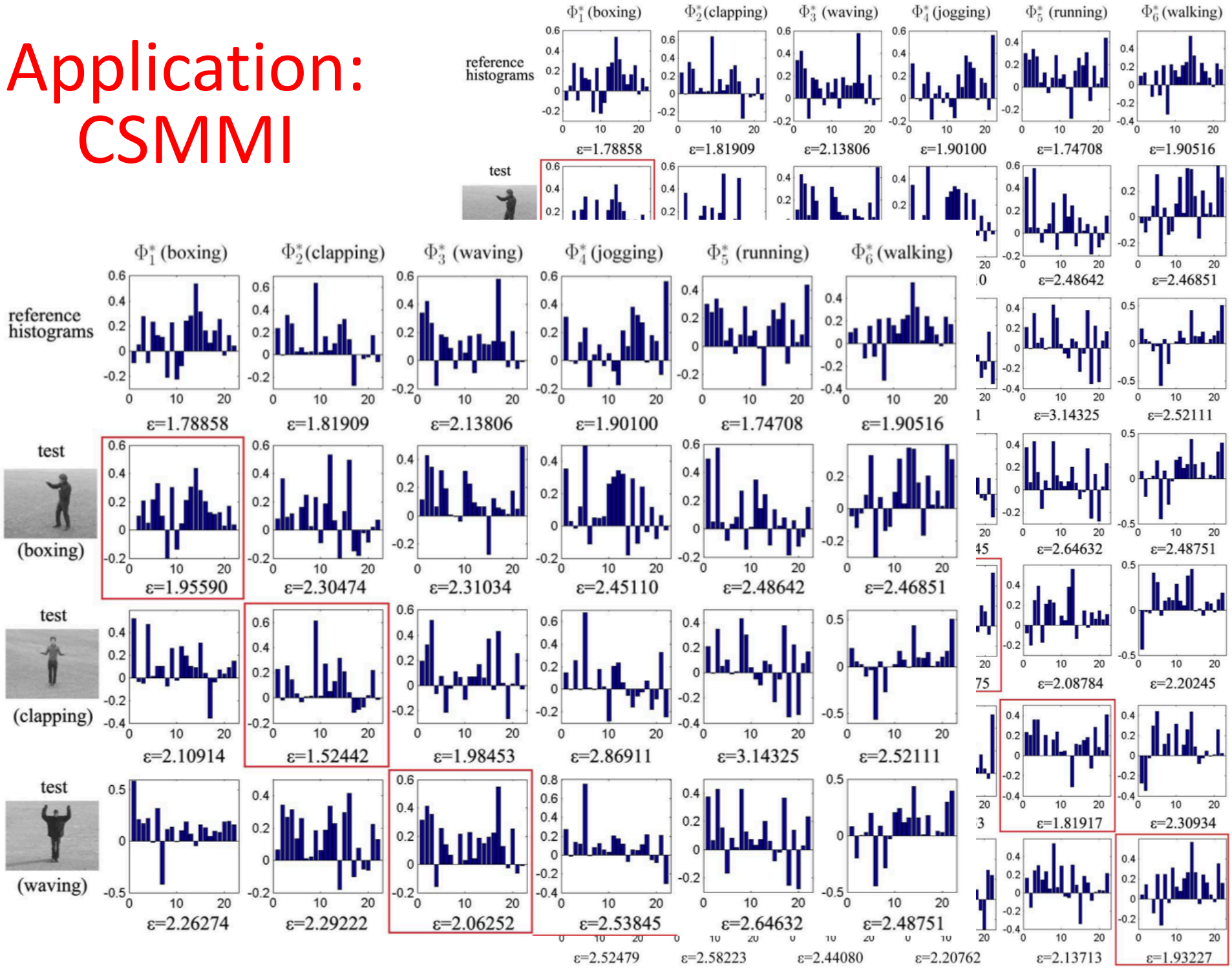
Papers	Methods	Dictionary Size	Average Accuracy
[29]	non-linear SVM+STIP	4000	91.8%
[39]	multiple instance learning +kinematic feature	—	87.7%
[41]	probabilistic spatiotemporal voting	—	88.0%
[37]	sparse linear approximation +Feature Covariance Matrices	—	97.4%
[26]	prototype trees	—	95.77%
[42]	Independent subspace analysis	—	93.9%
our method	*CSMMI*+STIP	365	**98.33%**

Application: CSMMI

- Experimental results



Application: CSMMI



Application: CSMMI

- Experimental results

TABLE III

COMPARISON ON THE UCF SPORTS ACTION DATASET

Papers	Methods	Dictionary Size	Average Accuracy
[30]	Maximum Average Correlation Height	—	69.2%
[42]	Independent subspace analysis	—	86.5%
[1]	class-specific dictionary +cuboid	256	83.8%
[43]	hierarchy of discriminative shape and motion features	300	87.27%
[44]	hough transform-based voting	—	86.6%
[3]	*ME*+STIP	325	81.33%
[6]	*Liu-Shah*+STIP	250	84%
[3]	*Qiu-Jiang*+STIP	308	85.33%
our method	*CSMMI*+STIP	469/250	**98.0%**/87.33%

TABLE IV

COMPARISON ON THE UCF YOUTUBE ACTION DATASET

Papers	Methods	Dictionary Size	Average Accuracy
[31]	cuboid+difussion maps	1000	70.4%
[45]	hybrid features	2000	71.2%
[42]	Independent Subspace Analysis	—	75.8%
[3]	ME+STIP	715	71.1%
[6]	*Liu-Shah*+STIP	624	72.7%
[3]	*Qiu-Jiang*+STIP	678	73.3%
our method	*CSMMI*+STIP	721	**78.6%**

TABLE V

COMPARISON ON THE HOLLYWOOD2 ACTION DATASET

Papers	Methods	Dictionary Size	mAP
[47]	dense+HOG/HOF	4000	47.4%
[48]	dense trajectories	—	58.3%
[42]	independent subspace analysis	—	53.3%
[46]	compensated descriptors +VLAD representation	—	**62.5%**
[3]	ME+STIP	329	41.3%
[6]	*Liu-Shah*+STIP	415	41.9%
[3]	*Qiu-Jiang*+STIP	394	43.2%
our method	*CSMMI*+STIP	437	**62.1%**

TABLE VI

COMPARISON ON THE KECK ACTION DATASET

Papers	Methods	Static setting	Dynamic setting
[26]	prototype trees	95.2%	91.07%
[49]	Product Manifolds	94.4%	92.3%
[3]	*ME*+shape-motion	91.2%	89.3%
[6]	*Liu-Shah*+shape-motion	94.2%	90.7%
[3]	*Qiu-Jiang*+shape-motion	94.9	92.7%
[3]	*Qiu-Jiang**+shape-motion	97%	—
our method	*CSMMI*+shape-motion	**95.1%**	**93.2%**

Many data summarization

- In many contemporary applications, running the standard greedy algorithm is computationally prohibitive:
  - The data set does not fit in main memory
  - Data itself arrives in a stream, possibly cannot be stored

- **Streaming algorithms:** Access only a small fraction of data at any point in time and provide approximate solutions

Streaming submodular maximization algorithms

Streaming submodular maximization

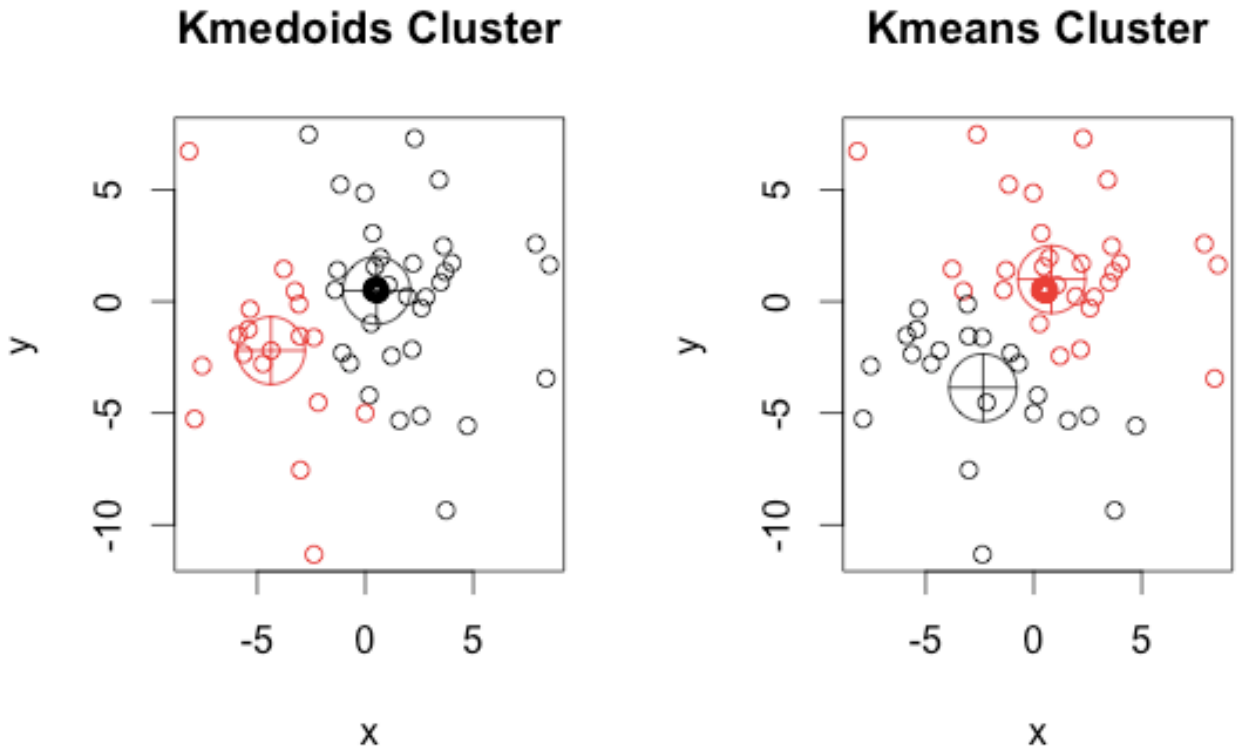
- Assumptions:
  - The ground set is ordered (arbitrarily) and any streaming algorithm must process V in the given order $V = \{e_1, \ldots, e_n\}$
  - At each iteration *t* the algorithm may maintain a memory $M_t \subset V$ of points and must be ready to output a candidate feasible solution $S_t \subset M_t$ of size at most $|S_t| \leq k$
  - When a new point arrives from the stream, the algorithm may elect to remember it
    * if the memory exceeds a specified capacity bound, it must discard elements before accepting new ones. T

Streaming submodular maximization

- The performance of a streaming algorithm is measured by:
  - Number of passes the algorithm needs to make over the stream
  - Memory required by the algorithm
  - Running time of the algorithm
  - Approximation ratio: $f(S_T)/\texttt{OPT}$

Sample applications (1)

- **Exemplar based clustering**: Select a set of exemplars that better represent a massive data set.



Sample applications (1)

- **Exemplar based clustering**: Select a set of exemplars that better represent a massive data set.

- K-medoid problem: $L(S) = \frac{1}{|V|} \sum_{e \in V} \min_{v \in S} d(e, v).$

- Introducing an auxiliary element e_0 we can turn L into a monotone submodular function

$$f(S) = L(\{e_0\}) - L(S \cup \{e_0\}).$$

Sample applications (2)

- **Large-scale nonparametric learning** (Active set selection): select a small representative subset of instances and only work with a kernel matrix restricted to this subset

$$K_{V,V} = \begin{pmatrix} \mathcal{K}_{e_1,e_1} & \cdots & \mathcal{K}_{e_1,e_n} \\ \vdots & & \vdots \\ \mathcal{K}_{e_n,e_1} & \cdots & \mathcal{K}_{e_n,e_n} \end{pmatrix}.$$

Sample applications (2)

- The informative vector machine criterion for Gaussian processes

$$f(S) = \frac{1}{2} \log \det(\mathbf{I} + \sigma^{-2} \Sigma_{S,S}),$$

$$f(S) = I(\mathbf{Y}_S; \mathbf{X}_V) = H(\mathbf{X}_V) - H(\mathbf{X}_V | \mathbf{Y}_S) = \tfrac{1}{2} \log \det(\mathbf{I} + \sigma^{-2} \Sigma_{S,S})$$

$$K_{V,V} = \begin{pmatrix} \mathcal{K}_{e_1,e_1} & \cdots & \mathcal{K}_{e_1,e_n} \\ \vdots & & \vdots \\ \mathcal{K}_{e_n,e_1} & \cdots & \mathcal{K}_{e_n,e_n} \end{pmatrix}.$$

Many data summarization

- Naïve approximations:
  - Greedy algorithm: too comp. expensive and cannot run in real streams
  - Maintaining in memory the k-best elements: performance degrades arbitrarily with k

	# passes	approx. guarantee	memory	update time
Standard Greedy [27]	$O(k)$	$(1-1/e)$	$O(k)$	$O(k)$
GREEDY-SCALING [20]	$O(1/\delta)$	$\delta/2$	$kn^{\delta}\log n$	?
STREAM-GREEDY [14]	multiple	$(1/2-\varepsilon)$	$O(k)$	$O(k)$
SIEVE-STREAMING	1	$(1/2-\varepsilon)$	$O(k\log(k)/\varepsilon)$	$O(\log(k)/\varepsilon)$

The Sieve-streaming algorithm

- Key observations:
  - Knowing OPT helps
  - Knowing $m = \max_{e \in V} f(\{e\})$ is enough
  - Lazy updates (approximate m)

The Sieve-streaming algorithm

$$S_i = S_{i-1} \cup \{\arg\max_{e \in V} \triangle_f(e|S_{i-1})\}$$

- Knowing $\texttt{OPT}$ helps.
  - If S_i is the set of the first i elements picked by the greedy algorithm, then the marginal value:

$$\triangle_f(e_{i+1}|S_i)$$

  - Of the next element added is at least:

$$(\texttt{OPT} - f(S_i))/k$$

- **Idea:** identify elements with similarly high marginal value, under a lowered threshold:

$$\beta\texttt{OPT}/k$$

The Sieve-streaming algorithm

$$S_i = S_{i-1} \cup \{\arg\max_{e \in V} \triangle_f(e|S_{i-1})\}$$

- Suppose we know OPT up to a constant factor α, i.e., we have a value *v* such that:

$$\texttt{OPT} \geq v \geq \alpha \cdot \texttt{OPT} \qquad 0 \leq \alpha \leq 1$$

- The algorithm starts with $S_0 = \emptyset$, and then after observing each element, it adds it to S if the marginal value is at least:

$$(v/2 - f(S))/(k - |S|)$$

and we are still below the cardinality constraint

The Sieve-streaming algorithm

Algorithm 1 SIEVE-STREAMING-KNOW-OPT-VAL

Input: v such that $\texttt{OPT} \geq v \geq \alpha\ \texttt{OPT}$

1: $S = \emptyset$
2: **for** $i = 1$ to n **do**
3: **if** $\triangle_f(e_i \mid S) \geq \frac{v/2 - f(S)}{k - |S|}$ and $|S_v| < k$ **then**
4: $S := S \cup \{e_i\}$
5: **return** S

PROPOSITION 5.1. *Assuming input v to algorithm 1 satisfies $\texttt{OPT} \geq v \geq \alpha\ \texttt{OPT}$, the algorithm satisfies the following properties*

- *It outputs a set S such that $|S| \leq k$ and $f(S) \geq \frac{\alpha}{2}\texttt{OPT}$*
- *It does 1 pass over the data set, stores at most k elements and has $O(1)$ update time per element.*

The Sieve-streaming algorithm

- Obtaining a good approximation to $\texttt{OPT}$ is not straightforward
-
-

Ain't a very useful estimate!

with $v=km$ and $\alpha=1/k$, we get a guarantee:

$$\texttt{OPT}/2k$$

$$m \leq \texttt{OPT} \leq k \cdot m.$$

Equivalently, a function $f : 2^V \to \mathbb{R}$ is *submodular* if for every $A, B \subseteq V$,

$$f(A \cap B) + f(A \cup B) \leq f(A) + f(B).$$

The Sieve-streaming algorithm

- Idea: refining the threshold. Consider the set:

$$O = \{(1+\epsilon)^i | i \in \mathbb{Z}, m \leq (1+\epsilon)^i \leq k \cdot m\}$$

- There should exist at least some $v \in O$ such that: $(1-\epsilon)\texttt{OPT} \leq v \leq \texttt{OPT}$

The Sieve-streaming algorithm

Algorithm 2 SIEVE-STREAMING-KNOW-MAX-VAL

Input: $m = \max_{e \in V} f(\{e\})$

1: $O = \{(1+\epsilon)^i | i \in \mathbb{Z}, m \leq (1+\epsilon)^i \leq k \cdot m\}$
2: For each $v \in O, S_v := \emptyset$
3: **for** $i = 1$ to n **do**
4: **for** $v \in O$ **do**
5: **if** $\triangle_f(e_i \mid S_v) \geq \frac{v/2 - f(S_v)}{k - |S_v|}$ and $|S_v| < k$ **then**
6: $S_v := S_v \cup \{e_i\}$
7: **return** $\text{argmax}_{v \in O_n} f(S_v)$

PROPOSITION 5.2. *Assuming input m to Algorithm 2 satisfies $m = \max_{e \in V} f(\{e\})$, the algorithm satisfies the following properties*

- *It outputs a set S such that $|S| \leq k$ and $f(S) \geq \left(\frac{1}{2} - \epsilon\right) OPT$*
- *It does 1 pass over the data set, stores at most $O\left(\frac{k \log k}{\epsilon}\right)$ elements and has $O\left(\frac{\log k}{\epsilon}\right)$ update time per element.*

The Sieve-streaming algorithm

- Final algorithm: relax the assumption we need to know the maximum value of all singletons:
  - Maintain an auxiliary variable *m* which holds the current maximum singleton element
  - Initiate thresholds for an increased range:

$$v = (1+\epsilon)^i, m \leq (1+\epsilon)^i \leq 2 \cdot k \cdot m$$

The Sieve-streaming algorithm

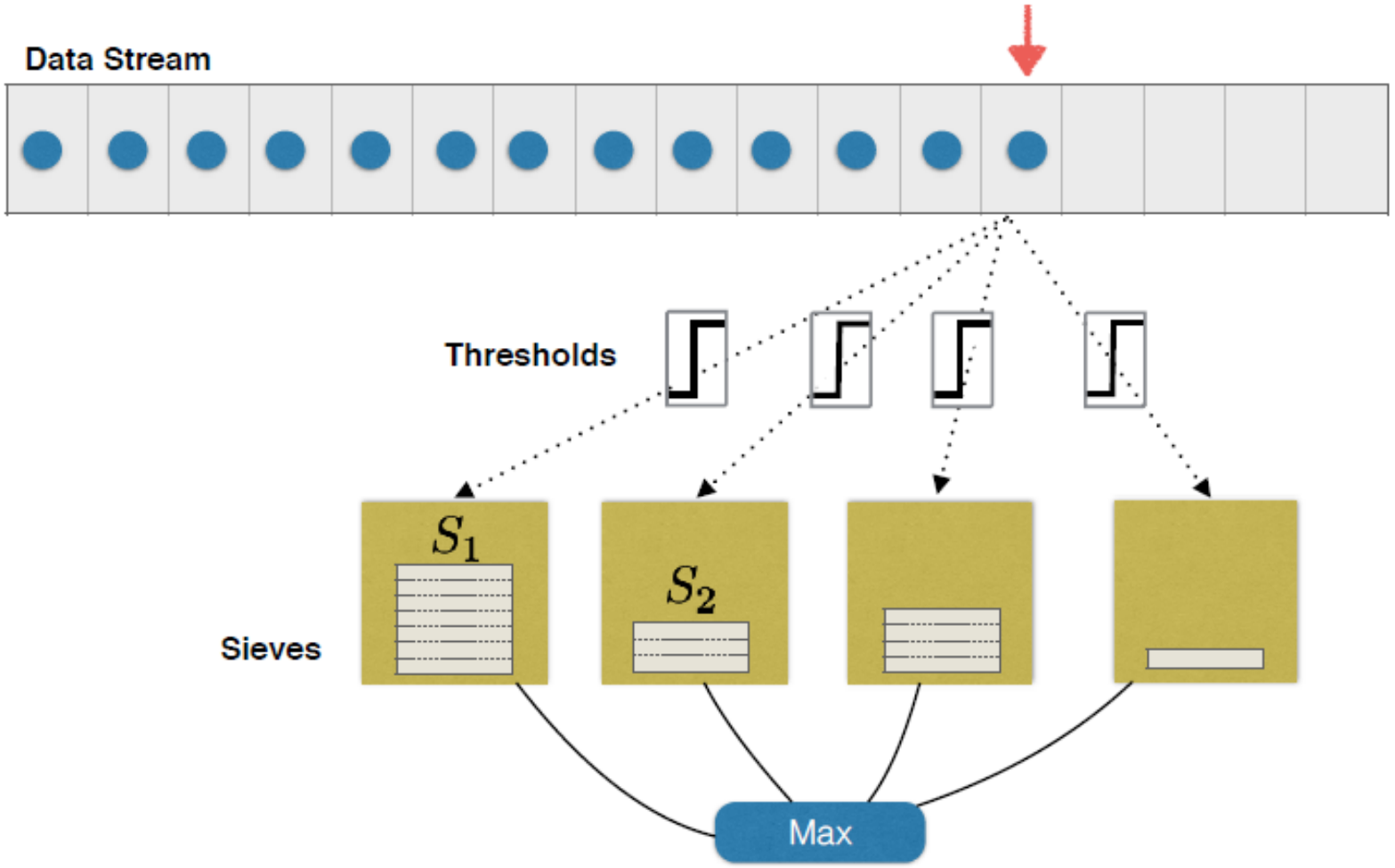
Algorithm 3 SIEVE-STREAMING

$O = \{(1+\epsilon)^i | i \in \mathbb{Z}\}$
For each $v \in O, S_v := \emptyset$ (maintain the sets only for the necessary v's lazily)
$m := 0$
for $i = 1$ to n **do**
  $m := \max(m, f(\{e_i\}))$
  $O_i = \{(1+\epsilon)^i | m \leq (1+\epsilon)^i \leq 2 \cdot k \cdot m\}$
  Delete all S_v such that $v \notin O_i$.
  for $v \in O_i$ **do**
    if $\triangle_f(e_i \mid S_v) \geq \frac{v/2 - f(S_v)}{k - |S_v|}$ and $|S_v| < k$ **then**
      $S_v := S_v \cup \{e_i\}$
return $\text{argmax}_{v \in O_n} f(S_v)$

THEOREM 5.3. SIEVE-STREAMING *(Algorithm 3) satisfies the following properties*

- *It outputs a set S such that $|S| \leq k$ and $f(S) \geq (\frac{1}{2} - \epsilon)$ OPT*
- *It does 1 pass over the data set, stores at most $O\left(\frac{k \log k}{\epsilon}\right)$ elements and has $O\left(\frac{\log k}{\epsilon}\right)$ update time per element.*

The Sieve-streaming algorithm



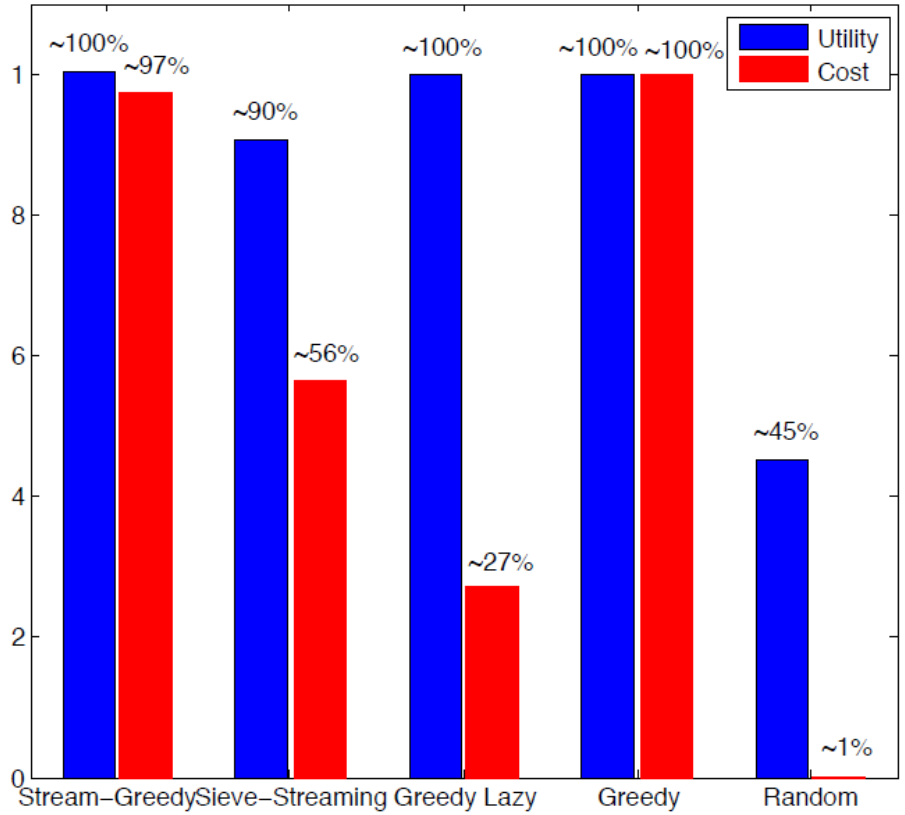
Experimental results

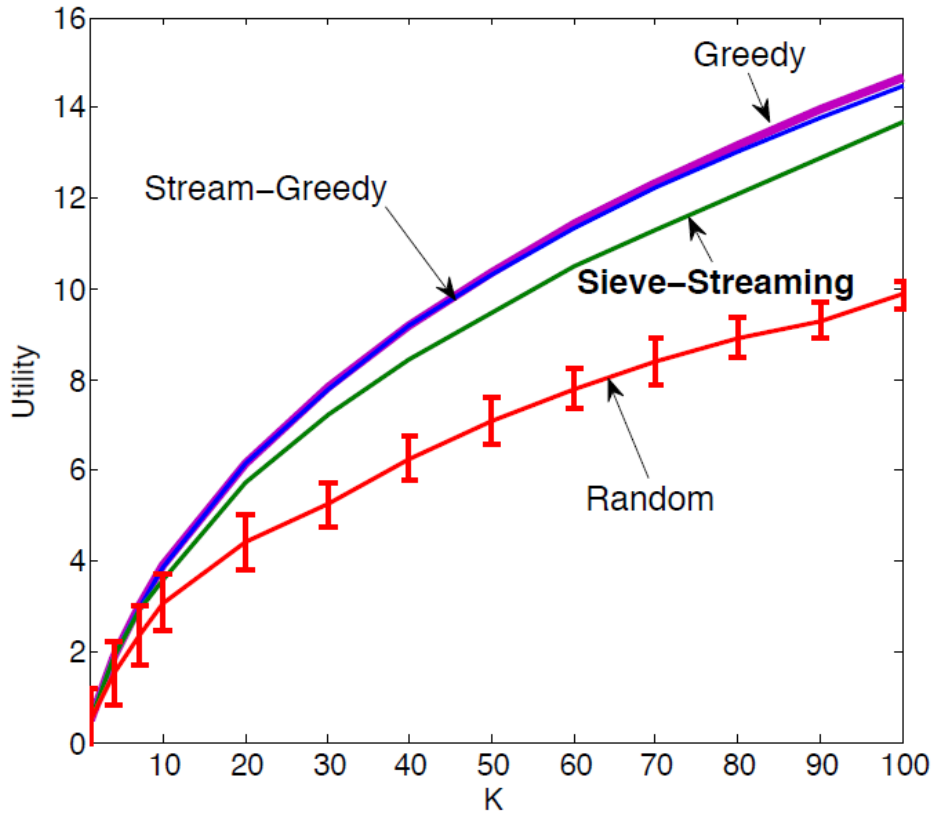
- Two applications:
  - Exemplar-based clustering
  - Active set selection for nonparametric learning

- Baselines vs. the Sieve-streaming algorithm
  - Random selection
  - Standard greedy
  - Lazy greedy
  - Stream greedy

Many data summarization

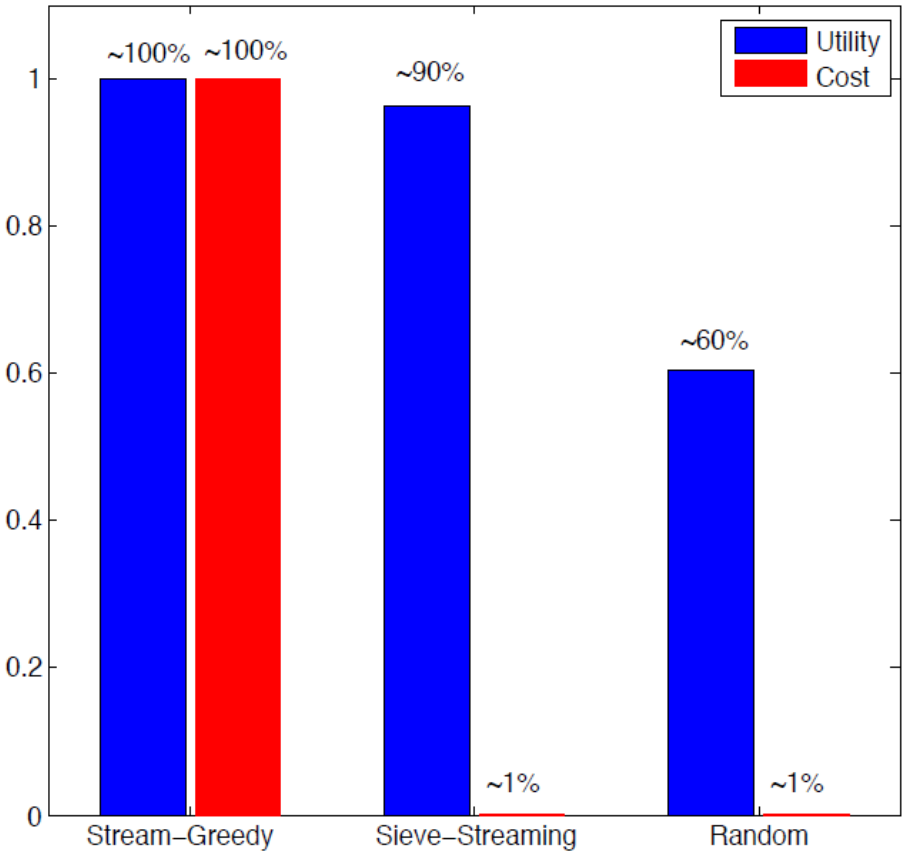
- Active set selection (5875 inst. 22 feats)

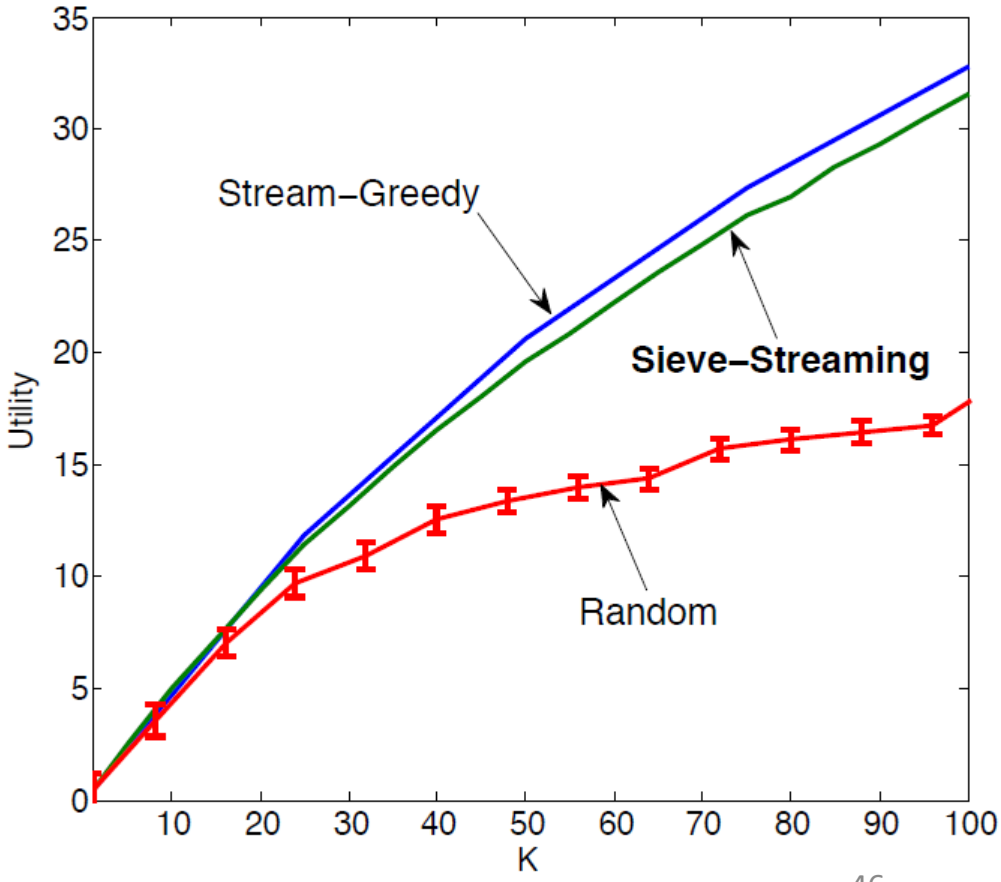




Many data summarization

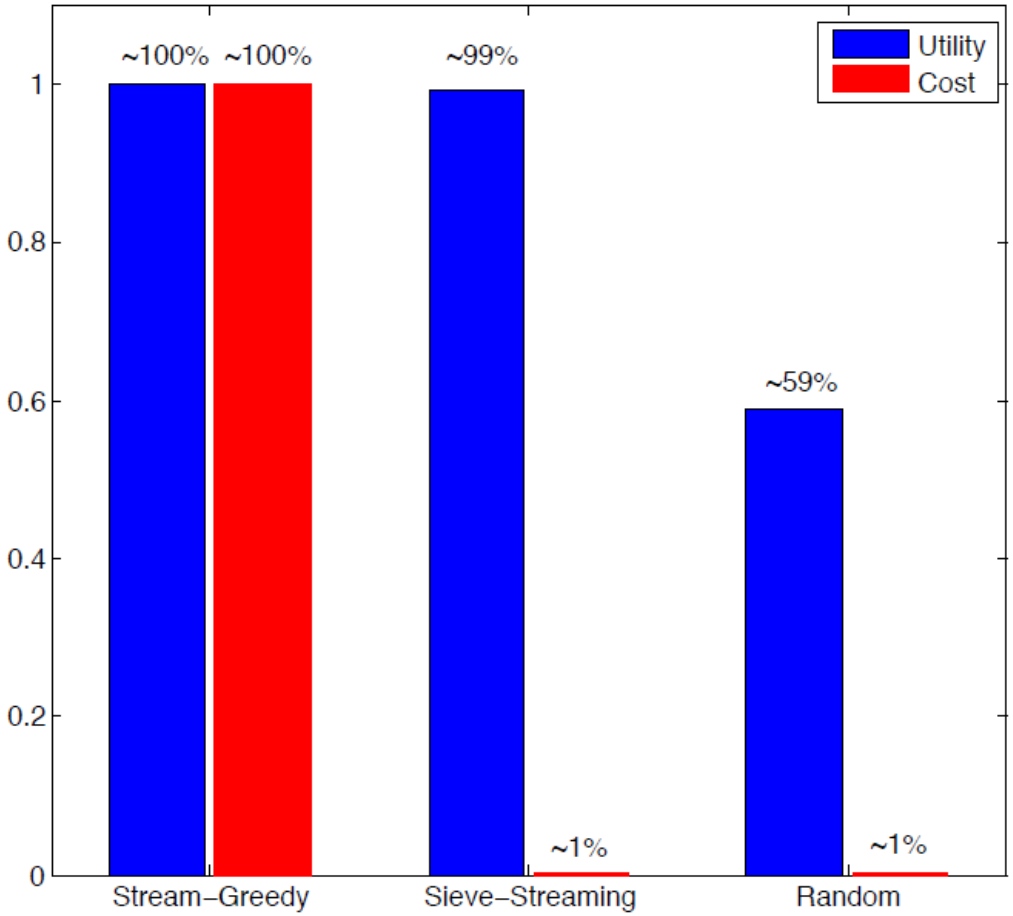
- Active set selection (+45 million inst. 6 feats)

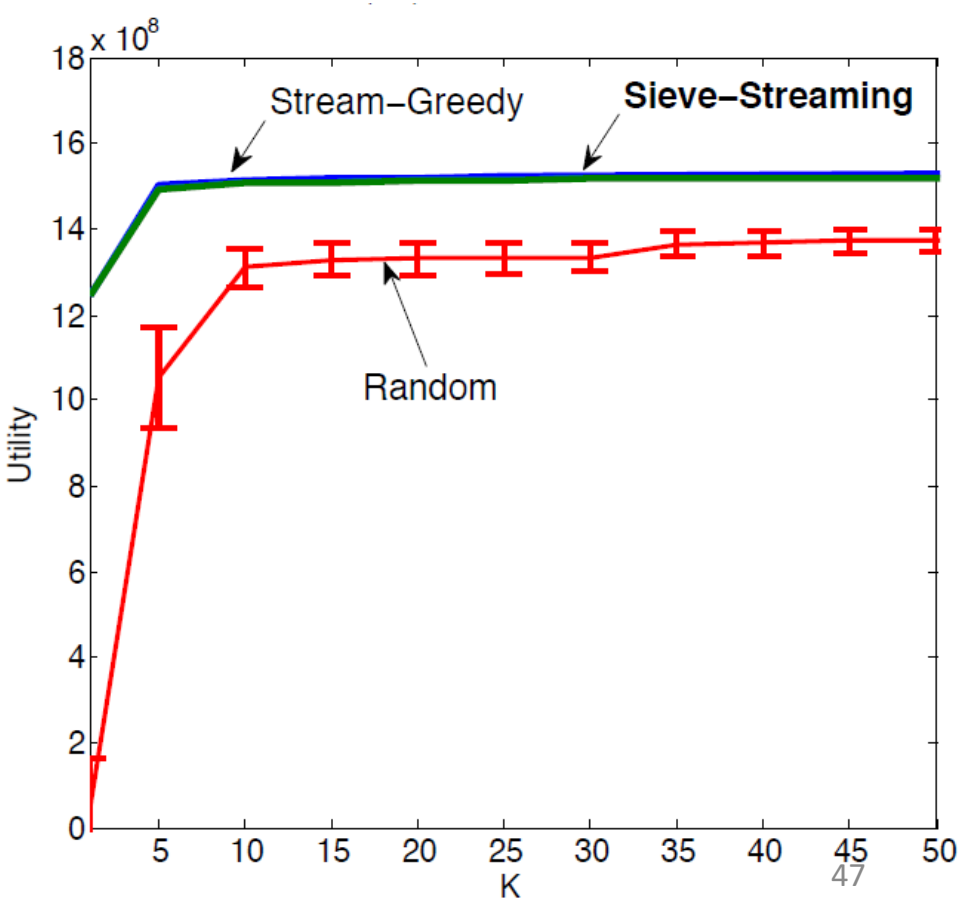




Many data summarization

- Exemplar-based clustering (2.5 million inst. 68 feats.)





Discussion

- Nice algorithms:
  - Easy to implement
  - Performance guarantees
  - Too many applications
- But, still have to be evaluated in the corresponding tasks
- Main challenge on using SFM: proof your objective function is monotone submodular
- How good is the guarantee for different tasks?
- There is already a lot of (ongoing) work on the use of SFM for diverse tasks
- **Matlab Toolbox for Submodular Function Optimization (v 2.0)**

http://las.ethz.ch/sfo/

Final remarks

- Research opportunities sith SFM:
  - NMF on a budget with SFM: what criterion?
  - Prototype selection/generation for NN classification (instance selection)
  - Vocabulary learning/construction for BoVWs: replace k-means with SFM of a supervised criterion
  - Multimodal document summarization / multimodal snippet generation: define appropriate criteria for SFM

References

- G. L. Nemhauser and L. A. Wolsey. **Best algorithms for approximating the maximum of a submodular set function.** Math. Oper. Research, 1978.
- Badanidiyuru, Ashwinkumar, et al. **Streaming submodular maximization: massive data summarization on the fly.** *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.*
- Andreas Krause, Daniel Golovin, **Submodular Function Maximization,** *Chapter in Tractability: Practical Approaches to Hard Problems, Cambridge University Press, 2014.*
- Andreas Krause, Ajit Singh, Carlos Guestrin, **Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies**, *In Journal of Machine Learning Research (JMLR), vol. 9, pp. 235-284, 2008.*
- Krause, Andreas, and Ryan G. Gomes. **Budgeted nonparametric learning from data streams.** *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010.
- Mirzasoleiman, Baharan, et al. **Distributed submodular maximization: Identifying representative elements in massive data.** *Advances in Neural Information Processing Systems*. 2013.
- Jun Wan, Vassilis Athitsos, Pat Jangyodsuk, Hugo Jair Escalante, Qiuqi Ruan, Isabelle Guyon. **CSMMI: Class-Specific Maximization of Mutual Information for Action and Gesture Recognition.** IEEE Trans. Image Processing, Vol 23(7):3152--3165