Clustering Algorithms for Streaming and Online Settings

Claire Monteleoni

Computer Science

George Washington University

Big Data Challenges for ML

We face an explosion in data!

Internet transactions
DNA sequencing
Satellite imagery
Environmental sensors
…

Real-world data can be:

Vast
High-dimensional
Noisy, raw
Sparse
Streaming, time-varying
Sensitive/private

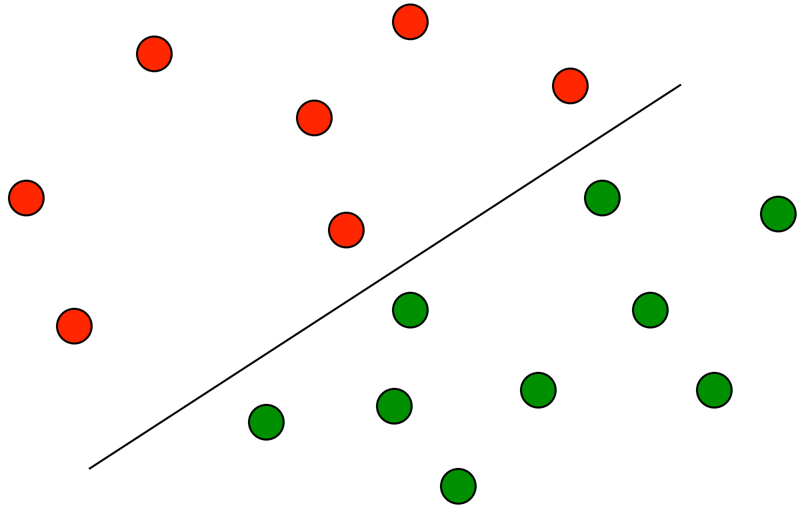
Machine Learning

Given labeled data points, find a good classification rule.

Describes the data

Generalizes well

E.g. linear classifiers:



Machine Learning algorithms for real data sources

Goal: design algorithms to detect patterns in real data sources.

Want efficient algorithms, with performance guarantees.

- Data streams
- Raw (unlabeled or partially-labeled) data
  - Active learning
  - Clustering
- Sensitive/private data
  - Privacy-preserving machine learning
- New applications of Machine Learning
  - Climate Informatics

Machine Learning algorithms for real data sources

Goal: design algorithms to detect patterns in real data sources.

Want efficient algorithms, with performance guarantees.

- Data streams
- Raw (unlabeled or partially-labeled) data
  - Active learning
  - Clustering

Scaling-up unsupervised learning to the velocity and volume of big data.

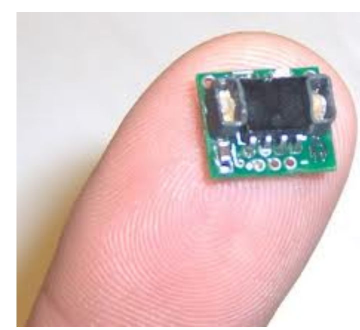
Data stream motivations

Data velocity: data arrives in a stream over time.

e.g. forecasting, real-time decision making, streaming data applications.

Data volume: data is large compared to memory or computation resources.



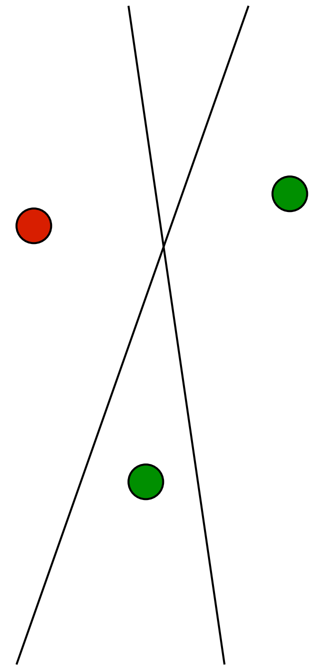


e.g. resource-constrained learning.

Learning from data streams

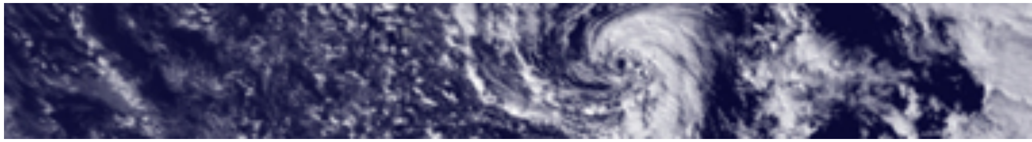
Data arrives in a stream over time.

E.g. linear classifiers:

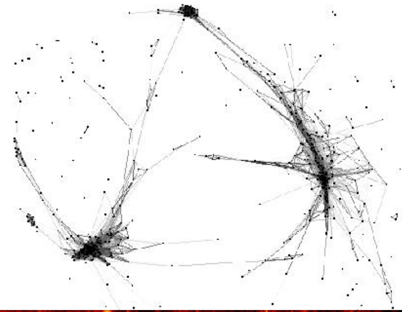


Clustering data streams: Motivations

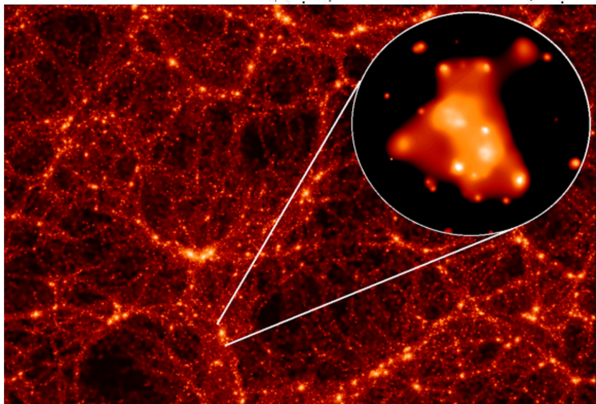
- Multimedia:
  - Aggregating and detecting topics in streaming media
    - e.g. clustering video, music, news stories



- Climate / weather:
  - Grouping / detecting spatiotemporal patterns
    - e.g. droughts, storms



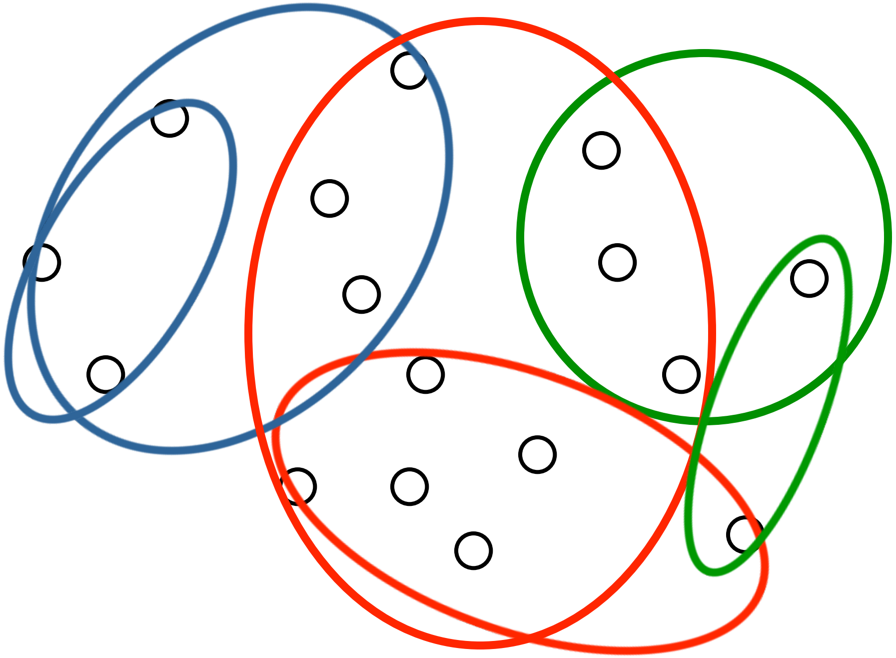
- Exploratory data analysis:
  - e.g. Neuroscience:
    - online spike classification
    - pattern detection in networks of neurons
    - network monitoring
  - Astronomy



Clustering

What can be done without any labels?

Unsupervised learning, Clustering.



How to evaluate a clustering algorithm?

k-means clustering objective

Clustering algorithms can be hard to evaluate without prior information or assumptions on the data.

With *no* assumptions on the data, one evaluation technique is w.r.t. some objective function.

A widely-cited and studied objective is the k-means clustering objective: Given set, $X \subset R^d$, choose $C \subset R^d$, $|C| = k$, to minimize:

$$\phi_C = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$$

k-means approximation

Optimizing k-means is NP-hard, even for k=2.
[Dasgupta '08; Deshpande & Popat '08].

Very few algorithms approximate the k-means objective.

Definition: b-approximation: $\phi_C \leq b \cdot \phi_{OPT}$

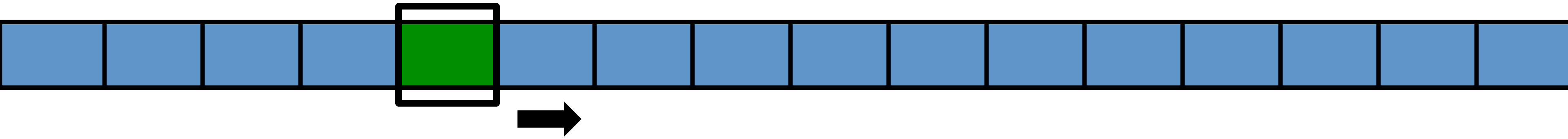
Definition: Bi-criteria (a,b)-approximation guarantee: a·k centers,

b-approximation.

Even "the k-means algorithm" [Lloyd 1957] does not have an approximation guarantee. Can suffer from bad initialization.

Goal: approximate the k-means clustering objective with streaming or online clustering algorithms [Open problems, Dasgupta '08]

Learning from data streams



“Streaming” model:

- Stream of of known length n.
- Memory available is o(n)
- Tested only at the end
- A (small) constant number of passes allowed

“Online” model:

- Endless stream of data
- Fixed amount of memory
- Tested at every time step
- Each point in stream is seen only once

Outline

Streaming clustering

[Ailon, Jaiswal & M, NIPS 2009]

Online clustering

[Choromanska & M, AISTATS 2012]

Streaming k-means approximation

[Ailon, Jaiswal & M, NIPS 2009]:

Goal: approximate the k-means objective with a one-pass streaming clustering algorithm

Related work:

[Arthur & Vassilvitskii, SODA '07]: *k*-means++, a batch clustering algorithm with O(log *k*)-approx. of *k*-means.

[Guha, Meyerson, Mishra, Motwani, & O'Callaghan, TKDE '03]: Divide and conquer streaming (a,b)-approximate *k*-medoid clustering.

Contributions to streaming clustering

Extend *k*-means++ to *k*-means#, an (O(log *k*), O(1))-approximation to k-means, in batch setting.

Analyze Guha *et al.* divide and conquer algorithm, using (a,b)-approximate *k*-means clustering.

Use Guha *et al.* with k-means# and then k-means++ to yield a one-pass O(log k)-approximation algorithm to *k*-means objective.

Analyze multi-level hierarchy version for improved memory vs. approximation tradeoff.

Experiments on real and simulated data.

k-means++

Algorithm:

```
Choose first center c₁ uniformly at random from X,
and let C = {c₁}.
Repeat (k-1) times:
   Choose next center cᵢ = x' ∈ X with prob. D(x',C)² / Σ_{x∈X} D(x,C)²
   C ← C ∪ {cᵢ}
```

Choose first center c_1 uniformly at random from X, and let $C = \{c_1\}$.

Repeat $(k-1)$ times:

- Choose next center $c_i = x' \in X$ with prob. $\frac{D(x', C)^2}{\sum_{x \in \mathcal{X}} D(x, C)^2}$
- $C \leftarrow C \cup \{c_i\}$

where $D(x, C) = \min_{c \in C} \|x - c\|$

Theorem (Arthur & Vassilvitskii '07): Returns an O(log *k*)-approximation, in expectation.

k-means#

Idea: *k*-means++ returns *k* centers, with O(log *k*)-approximation. Can we design a variant that returns O(*k* log *k*) centers, but constant approximation?

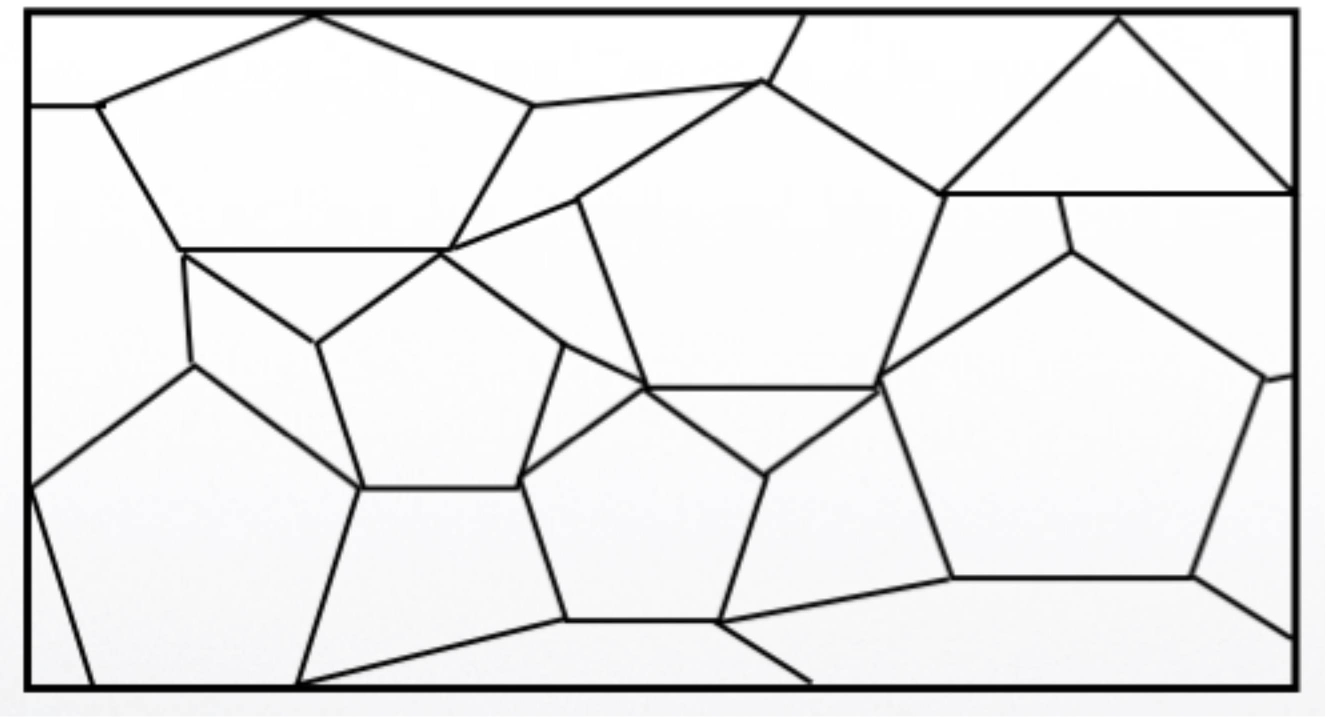
Algorithm:

```
Initialize C={}.
Choose 3·log(k) centers independently and uniformly
at random from X, and add them to C.
Repeat (k-1) times:
   Choose 3·log(k) centers indep. with prob. D(x',C)^2 / Σ_{x∈X} D(x,C)^2
   and add them to C.
```

$$\frac{D(x',C)^2}{\sum_{x\in\mathcal{X}} D(x,C)^2}$$

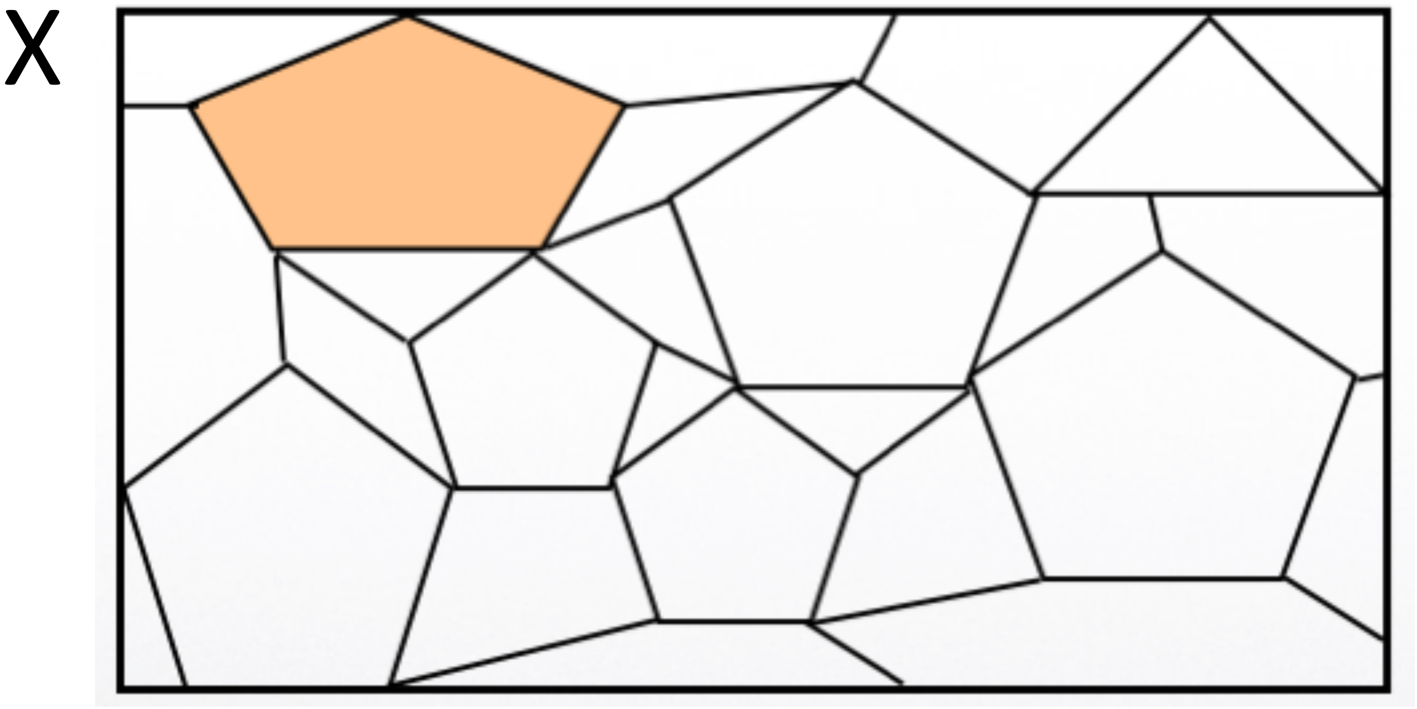
k-means# proof idea

X



The clustering (partition) induced by OPT.

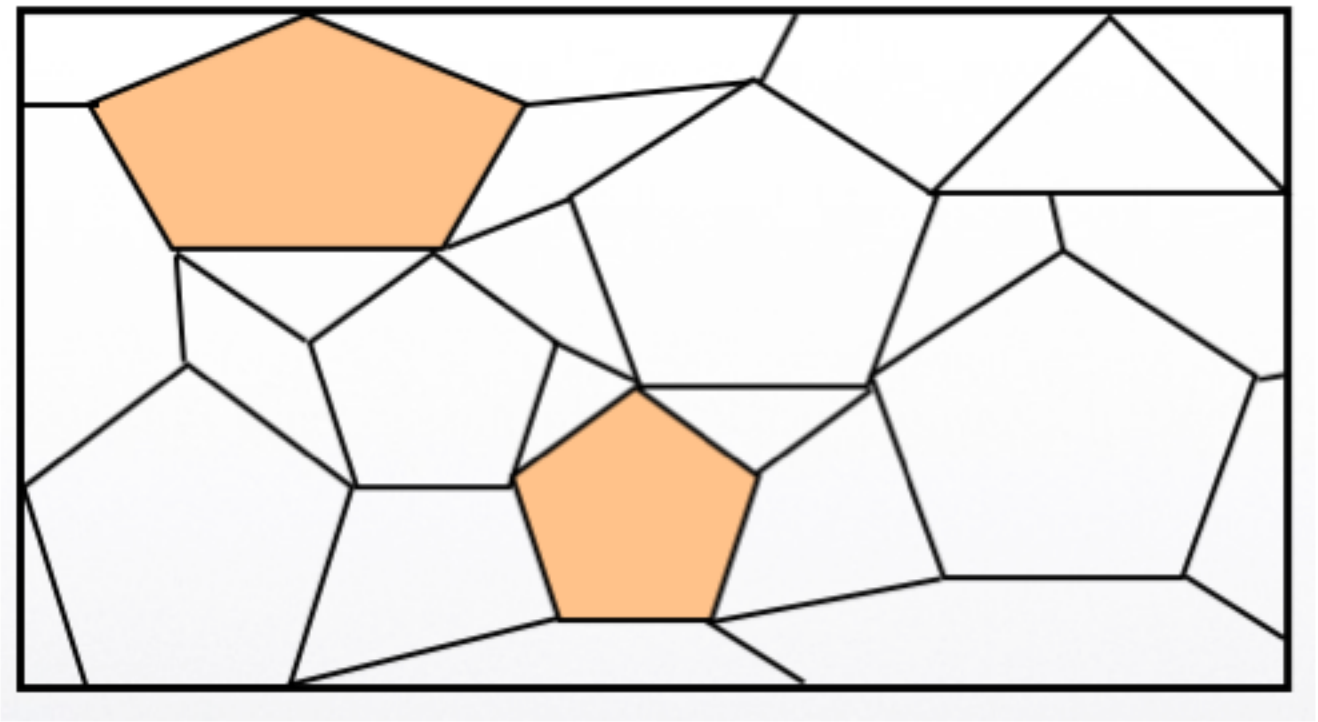
k-means# proof idea



The clustering (partition) induced by OPT.

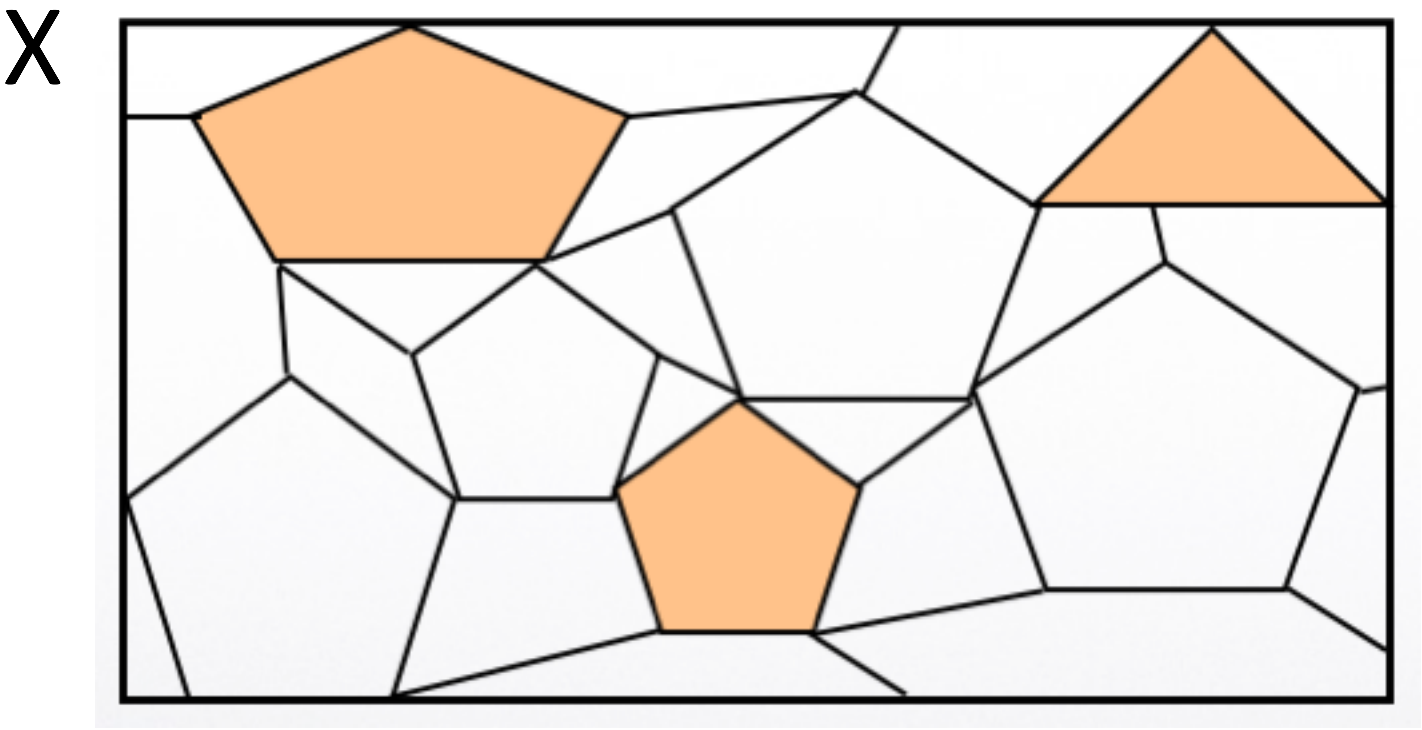
k-means# proof idea

X



The clustering (partition) induced by OPT.

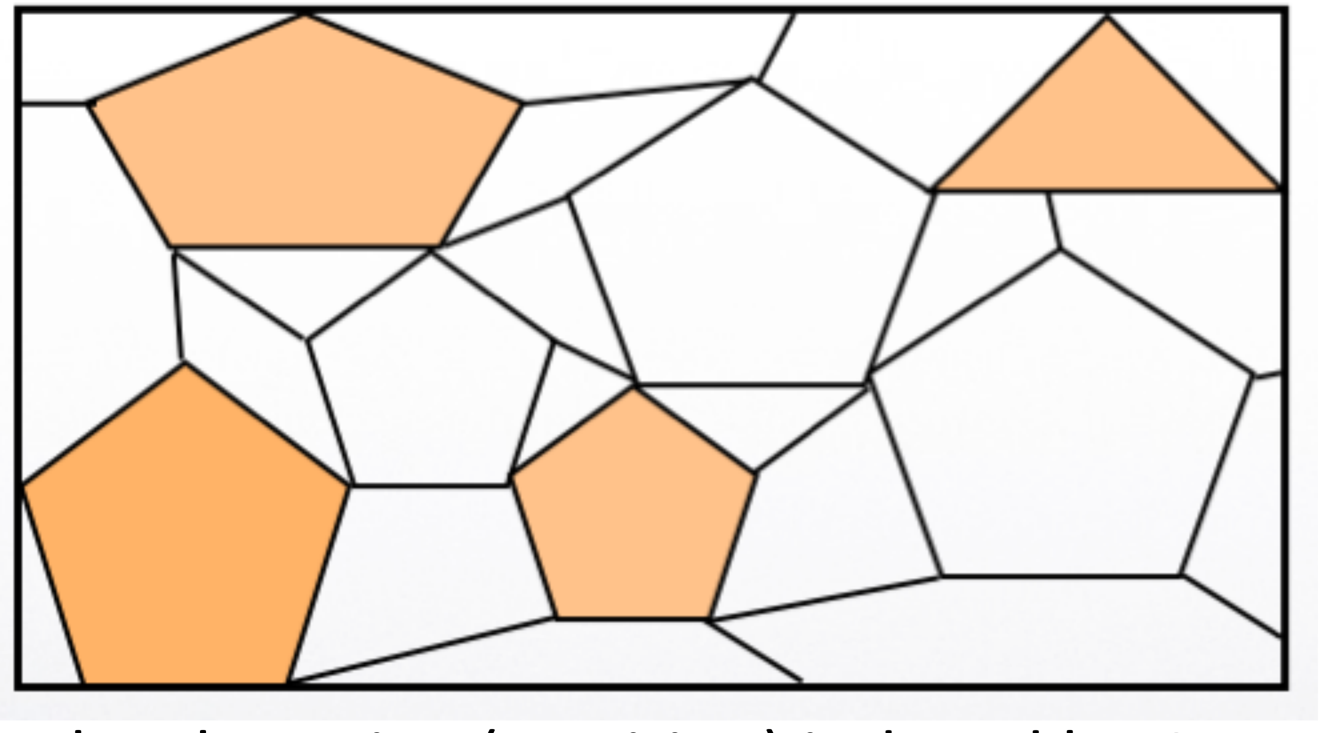
k-means# proof idea



The clustering (partition) induced by OPT.

k-means# proof idea

X



The clustering (partition) induced by OPT.

→ We cover the *k* clusters in OPT, after choosing *O(k log k)* centers.

k-means#

Theorem: With probability at least 1/4, *k*-means# yields an O(1)-approximation, on O(*k* log *k*) centers.

Proof outline: Definition "covered": cluster A ∈ OPT is covered if:

$\phi_C(A) < 32 \cdot \phi_{OPT}(A)$, where $\phi_C(A) = \sum_{x \in A} D(x, C)^2$.

Define $\{X_c, X_u\}$: the partition of X into covered, uncovered.

- In first round we cover one cluster in OPT.
- In any later round, either:

Case 1: $\phi_C(X_c) > \phi_C(X_u)$: We are done. (Reached 64-approx.)

Case 2 : $\phi_C(X_c) \leq \phi_C(X_u)$: We are likely to hit and cover another uncovered cluster in OPT.

We show *k*-means# is a (3 · log(*k*), 64)-approximation to k-means.

k-means# proof: First round

Fix any point x chosen in the first step. Define A as the unique cluster in OPT, s.t. x $\in$ A.

Lemma (AV '07): Fix A $\in$ OPT, and let C be the 1-clustering with the center chosen uniformly at random from A. Then $E[\phi_C(A)] = 2 \cdot \phi_{OPT}(A)$.

Corollary: $Pr[\phi_C(A) < 8 \cdot \phi_{OPT}(A)] \geq 3/4$. Pf. Apply Markov's inequality.

After 3 · log(*k*) random points, probability of hitting a cluster A with a point that is good for A is at least $1 - (1/4)^{3 \log k} \geq 1 - 1/k$.

So after first step, w.p. at least (1-1/k), at least 1 cluster is covered.

k-means# proof: Case 1

Case 1: $\phi_C(X_c) > \phi_C(X_u)$:

Since X= $X_c \cup X_u$ and by definition of ϕ,

$$\phi_C(X) = \phi_C(X_c) + \phi_C(X_u) \leq 2 \cdot \phi_C(X_c) \leq 64 \cdot \phi_{OPT}(X_c) \leq 64 \cdot \phi_{OPT}(X)$$

by definition of Case 1, and definition of covered.

Last inequality is by $X_c \subseteq X$, and definition of ϕ (each term in sum is nonnegative).

k-means# proof: Case 2

Case 2: $\phi_C(X_c) \leq \phi_C(X_u)$:

The probability of picking a point in X_u at the next round is:

$$\frac{\sum_{x \in X_u} D(x,C)^2}{\sum_{x \in X} D(x,C)^2} = \frac{\phi_C(X_u)}{\phi_C(X_u) + \phi_C(X_c)} \geq \frac{1}{2}$$

Lemma (AV '07): Fix A ∈ OPT, and let C be any clustering. If we add a center to C, sampled randomly from the D^2 weighting over A, yielding C' then: $E[\phi_{C'}(A)] \leq 8 \cdot \phi_{OPT}(A)$.

Corollary: $Pr[\phi_{C'}(A) < 32 \cdot \phi_{OPT}(A)] \geq 3/4$. By Markov's inequality.

So, w.p. $\geq \frac{1}{2} \cdot \frac{3}{4} = \frac{3}{8}$ we pick a point in X_u that covers a new cluster in OPT.

After $3 \cdot \log(k)$ picks, prob. of covering a new cluster is at least (1-1/k).

k-means# proof summary

For the first round, prob. of covering a cluster in OPT is at least (1-1/k).

For the k-1 remaining rounds, either Case 1 holds, and we have achieved a 64-approximation, or Case 2 holds, and the probability of covering a new cluster in OPT, in the next round, is at least (1-1/k).

So the probability that after k rounds there exists an uncovered cluster in OPT is $\leq 1-(1-1/k)^k \leq 3/4$.

Thus the algorithm achieves a 64-approximation on $3k \cdot \log(k)$ centers, with probability at least 1/4.

k-means#

Theorem: With probability at least 1/4, k-means# yields an O(1)-approximation, on O(k log k) centers.

Corollary: With probability at least 1-1/n, running k-means# for $3 \cdot \log n$ independent runs yields an O(1)-approximation (on O(k log k) centers).

Proof: Call it repeatedly, $3 \cdot \log n$ times, independently, and choose the clustering that yields the minimum cost. Corollary follows, since

$$\left(1-(3/4)^{3\log n}\right) \geq \left(1-\frac{1}{n}\right) .$$

Divide and conquer clustering

S - stream

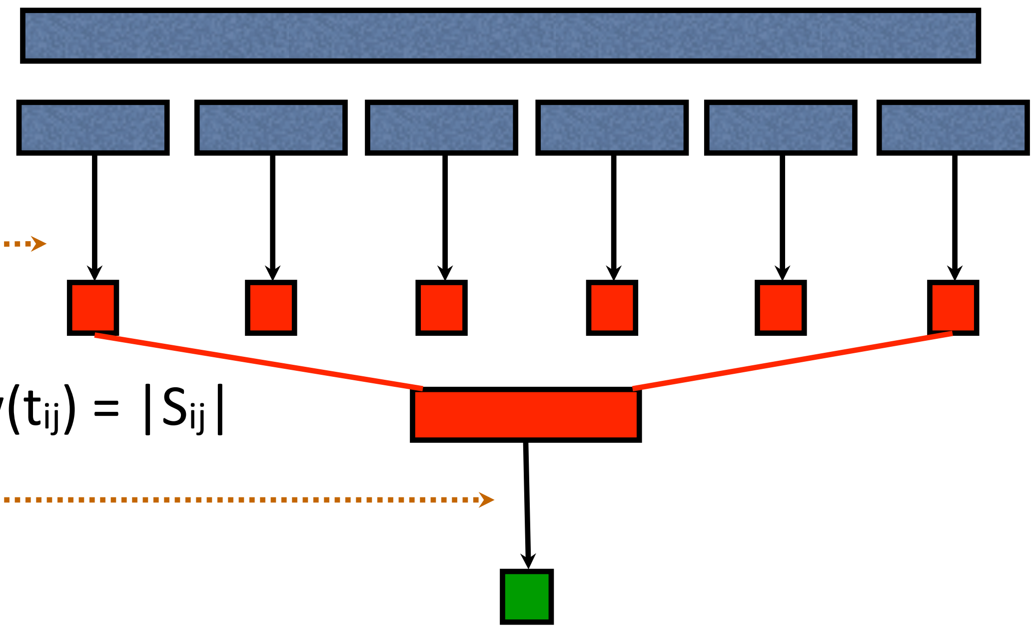
$\{S_i\}$ – partition

(a,b)-clustering

$\{T_i\}$ - sets of 'centers'

S_w - $\cup_i T_i$ with weights $w(t_{ij}) = |S_{ij}|$

(a',b')-clustering



T – final "centers"

[Guha *et al.* '03] analyzed this template for *k*-medoid clustering:

(a', O(bb'))-approximation.

One-pass *k*-means approximation

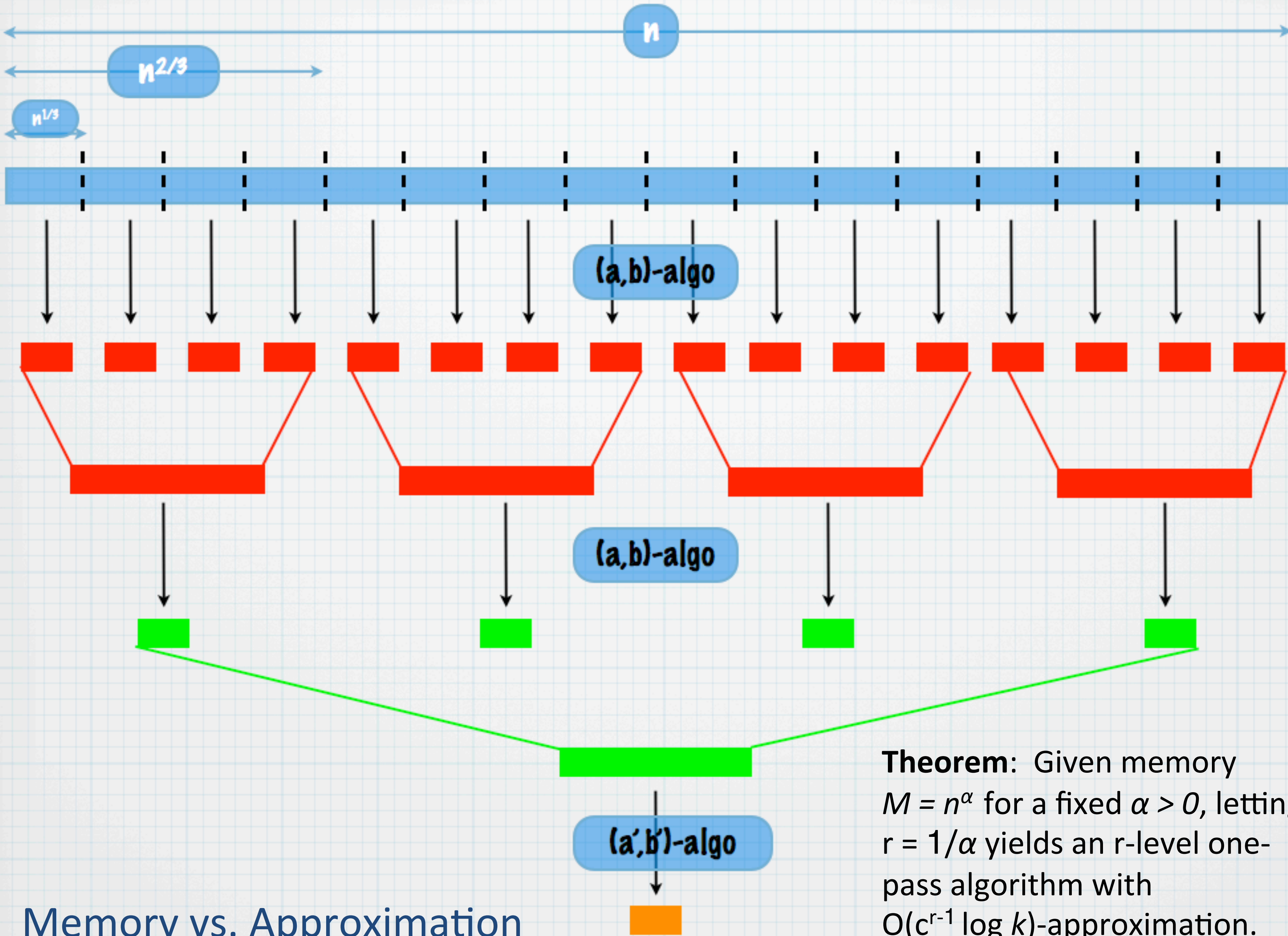
We analyze the Guha *et al.* scheme for (a,b)-approximation algorithms w.r.t. *k*-means: yields a one-pass (a', O(bb'))-approximation algorithm.

Our algorithm:

For the (a,b) algorithm, use (repeated) k-means#: a = O(log *k*), b = O(1).

For the (a',b') algorithm, use k-means++: a' = 1, b' = O(log k)

So the combined algorithm is a (1, O(log k))-approximation to *k*-means.



Memory vs. Approximation

Theorem: Given memory $M = n^{\alpha}$ for a fixed $\alpha > 0$, letting $r = 1/\alpha$ yields an r-level one-pass algorithm with $O(c^{r-1} \log k)$-approximation.

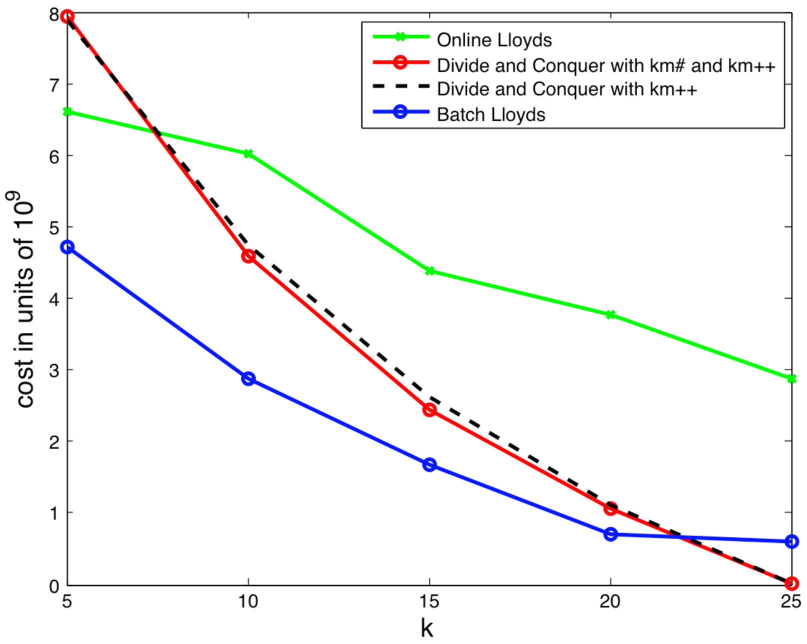
Experiments

k	BL	OL	DC-1	DC-2	BL	OL	DC-1	DC-2
5	$4.7254 \cdot 10^9$	$6.5967 \cdot 10^9$	$7.9336 \cdot 109$	$7.8752 \cdot 109$	5.80	1.44	16.95	12.22
10	$2.8738 \cdot 10^9$	$6.0146 \cdot 10^9$	$4.5968 \cdot 10^9$	$4.7288 \cdot 10^9$	7.33	2.76	53.10	24.74
15	$1.6753 \cdot 10^9$	$4.3743 \cdot 10^9$	$2.4338 \cdot 10^9$	$2.6280 \cdot 10^9$	8.85	4.00	112.68	36.86
20	$7.0016 \cdot 10^8$	$3.7794 \cdot 10^9$	$1.0661 \cdot 10^9$	$1.1017 \cdot 10^9$	11.75	6.04	250.21	48.57
25	$6.0011 \cdot 10^8$	$2.8859 \cdot 10^9$	$2.7493 \cdot 10^5$	$2.7906 \cdot 10^5$	13.83	7.00	403.81	60.96

Table 1: norm25 dataset. (columns 2-5 has the clustering cost and columns 6-9 has time in sec.)

Mixture of 25 Gaussians:

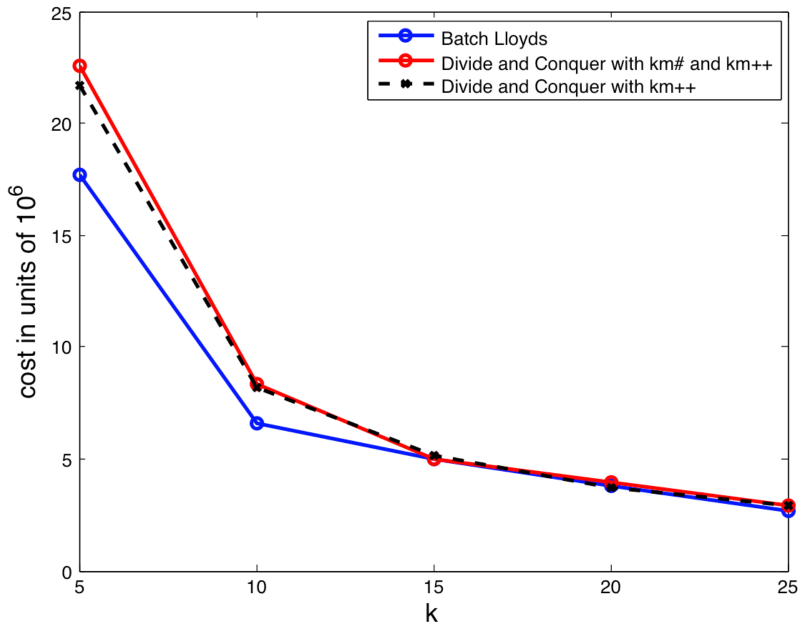
10K points sampled from a mixture of 25 Gaussians chosen at random from 15 dimensional hypercube (side 500).

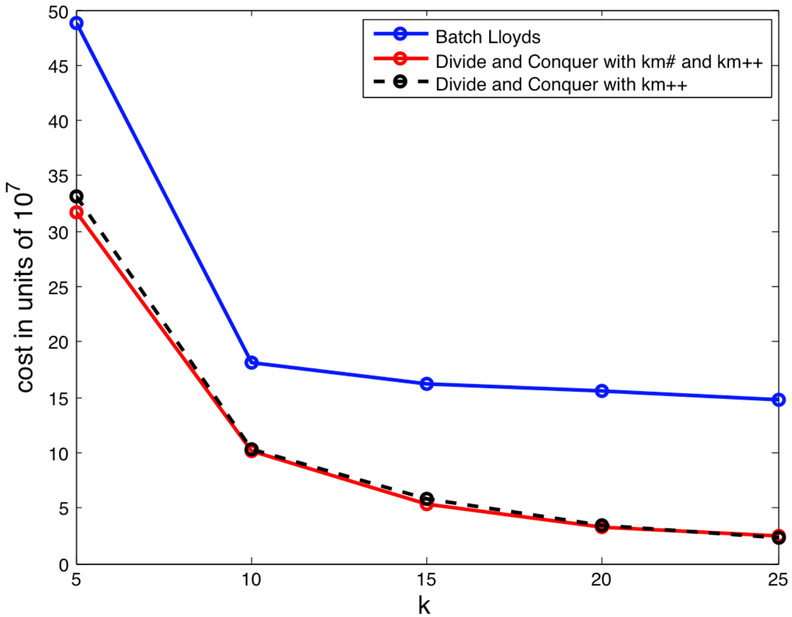


Experiments

k	BL	OL	DC-1	DC-2	BL	OL	DC-1	DC-2
5	1.7713 · 107	1.2401 · 108	2.2582 · 107	2.1683 · 107	1.78	0.15	2.30	1.10
10	$6.5871 \cdot 10^6$	$8.5684 \cdot 10^7$	$8.3452 \cdot 10^6$	$8.2037 \cdot 10^6$	2.27	0.31	7.45	2.40
15	$4.9851 \cdot 10^6$	$8.4633 \cdot 10^7$	$4.9935 \cdot 10^6$	$5.1391 \cdot 10^6$	3.42	0.45	13.34	3.32
20	$3.7836 \cdot 10^6$	$6.5110 \cdot 10^7$	$3.9289 \cdot 10^6$	$3.7279 \cdot 10^6$	3.38	0.59	32.42	5.00
25	$2.6363 \cdot 10^6$	$6.3758 \cdot 10^7$	$2.8899 \cdot 10^6$	$2.9470 \cdot 10^6$	4.54	0.62	46.45	5.89

Table 2: Cloud dataset. (columns 2-5 has the clustering cost and columns 6-9 has time in sec.)





k	BL	OL	DC-1	DC-2	BL	OL	DC-1	DC-2
5	$4.8769 \cdot 10^8$	$1.7001 \cdot 10^9$	$3.1770 \cdot 10^8$	$3.3191 \cdot 10^8$	3.74	0.87	14.60	6.53
10	$1.8169 \cdot 10^8$	$1.6930 \cdot 10^9$	$1.0104 \cdot 10^8$	$1.0271 \cdot 10^8$	5.59	1.66	47.92	12.17
15	$1.6227 \cdot 10^8$	$1.4762 \cdot 10^9$	$5.3517 \cdot 10^7$	$5.7865 \cdot 10^7$	7.04	2.19	86.54	17.53
20	$1.5580 \cdot 10^8$	$1.4766 \cdot 10^9$	$3.2577 \cdot 10^7$	$3.4155 \cdot 10^7$	9.87	2.83	218.95	25.70
25	$1.4704 \cdot 10^8$	$1.4754 \cdot 10^9$	$2.3981 \cdot 10^8$	$2.2735 \cdot 10^8$	13.26	4.41	331.77	40.64

Table 3: Spambase dataset. (columns 2-5 has the clustering cost and columns 6-9 has time in sec.)

UCI data: Clouds and Spambase.

Outline

Streaming clustering

[Ailon, Jaiswal & M, NIPS 2009]

Online clustering

[Choromanska & M, AISTATS 2012]

Open problems posed by Dasgupta

Provide an online algorithm for k-means clustering endless streams in either framework [Dasgupta, Spring '08, Lecture 6]:

1. At time t, algorithm sees data point x_t, and outputs the set of k centers C_t. For some constant $\alpha \geq 1$ and for all t:

$$\texttt{cost}(C_t) \leq \alpha \cdot OPT_t.$$

where $OPT_t = \texttt{cost(best } k \texttt{ centers for } x_1, \ldots, x_t\texttt{)}.$

2. At time t, algorithm announces set of k centers C_t, then sees x_t and incurs loss equal to cost of x_t under C_t: the squared distance from x_t to closest center in C_t. Goal: bound the regret G, between cumulative loss at time T, and OPT for the stream seen so far:

$$L_T(\texttt{alg}) = \sum_{t \leq T} \min_{c \in C_t} \|x_t - c\|^2 \leq OPT_T + G$$

Online clustering with experts

[Choromanska & M, AISTATS 2012]

Goal: approximate the k-means clustering objective with an online clustering algorithm

- A new evaluation framework, extending Dasgupta's
  - Bound variant of 2 w.r.t. performance of a set of experts: clustering algorithms
- A new family of online clustering algorithms
  - Extend algorithms for online learning with experts
- Performance guarantees with no data assumptions
  - Regret bounds
  - Novel form of online clustering approximation guarantees, w.r.t. OPT!
- Encouraging experimental performance

Contributions to online clustering

- Extend online learning algorithms from [Herbster & Warmuth '98] and [M & Jaakkola '03] to clustering setting.
  - Instead of using prediction errors to update weights over experts, use a proxy for k-means cost obtained so far.

- Prove (c,η)-realizability of our clustering and loss function.
  - Allows us to extend regret bounds from [HW98] and [MJ03].

- Add assumptions that experts are b-approximation algorithms w.r.t. k-means objective, to extend regret bounds

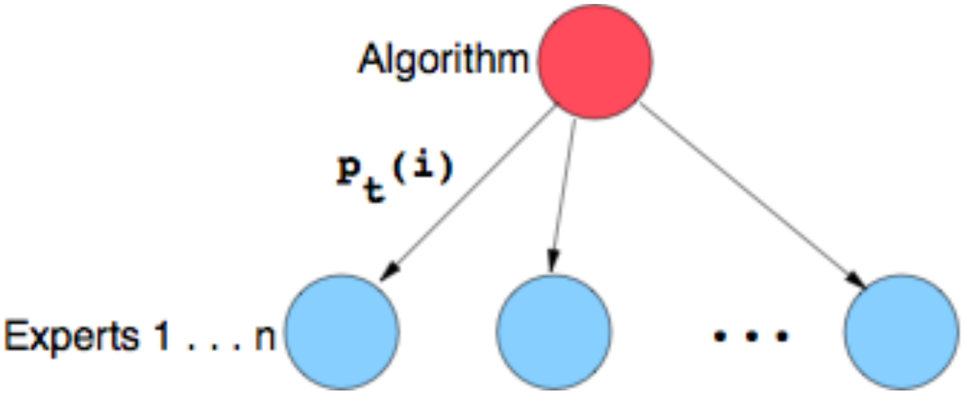
→ Novel online approximation bounds w.r.t. OPT for the entire stream!

Online learning (supervised setting)

- Learning proceeds in stages.
  - Algorithm first predicts a label for the current data point.
  - Loss is then computed: function of predicted and observed label.
  - Learner can update its hypothesis (usually taking into account loss).
- Framework models regression, or classification
  - By varying choice of loss function:
    - Many hypothesis classes
    - Problem need not be separable
- Non-stochastic setting: no statistical assumptions.
  - No assumptions on observation sequence.
  - Observations can even be generated online by an adaptive adversary.
- Analyze regret: difference in cumulative loss from that of the optimal comparator algorithm for the observed sequence (computed in hindsight) .

Online learning with experts

Learner maintains distribution over *n* “experts.”



Experts are black boxes: need not be good algorithms, can vary with time, and depend on one another.

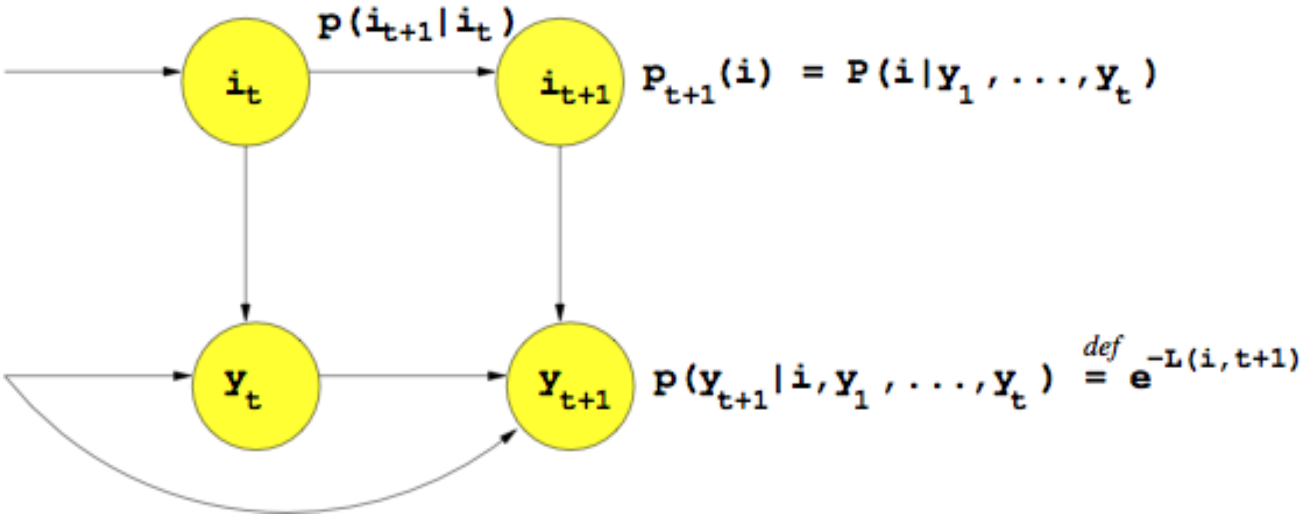
Learner informs prediction using a probability distribution $p_t(i)$ over experts, i, depending on L(i,t), loss of expert i’s output w.r.t. observation – defined per problem.

Different algorithms to update p_t (i) - based on the model of time-varying data.

Shifting algorithms

To handle changing observations, maintain $p_t(i)$ via an HMM.

Hidden state: identity of the current best expert.



[M&Jaakkola'03]: Performing Bayesian updates on this HMM yields existing online learning algorithms.

$$p_{t+1}(i) \propto \sum_j p_t(j) e^{-L(j,t)} p(i|j)$$

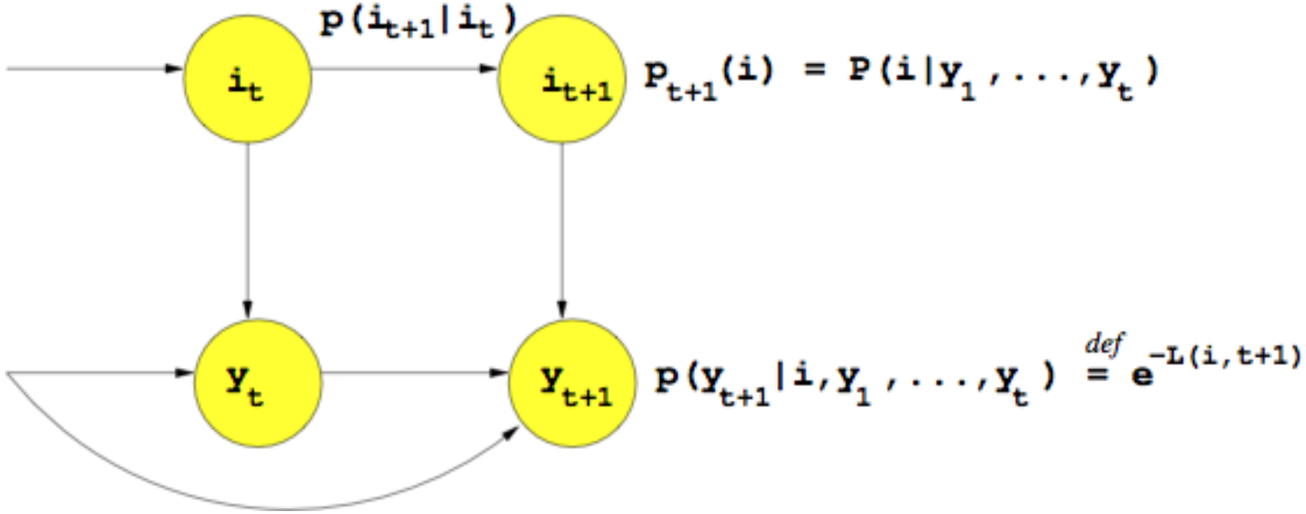
Static update, $P(i \mid j) = \delta(i,j)$ gives [Littlestone&Warmuth'89] algorithm: The Weighted Majority Algorithm, a.k.a. Static-Expert.

$$p_{t+1}(i) \propto p_t(i) e^{-L(i,t)}$$

Shifting algorithms

To handle changing observations, maintain $p_t(i)$ via an HMM.

Hidden state: identity of the current best expert.



Performing Bayesian updates on this HMM yields existing OL algorithms.

$$p_{t+1}(i) \propto \sum_j p_t(j) e^{-L(j,t)} p(i|j)$$

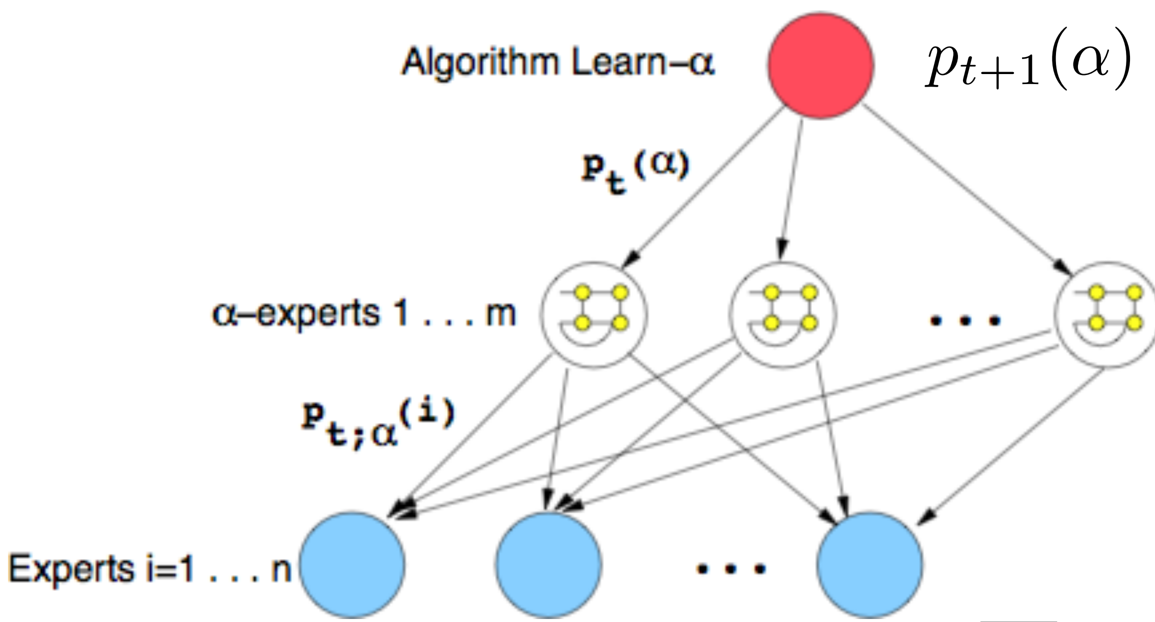
[Herbster&Warmuth'98]
Model shifting concepts via:

$$P(i|j;\alpha) = \begin{cases} (1-\alpha) & i = j \\ \frac{\alpha}{n-1} & i \neq j \end{cases}$$

Learn-α algorithm

[M, 2003] [M & Jaakkola, NIPS 2003]

Learn-α algorithm: Learn the α parameter using α–experts, each updating with different value of α. Use Bayesian updates to track the best α.



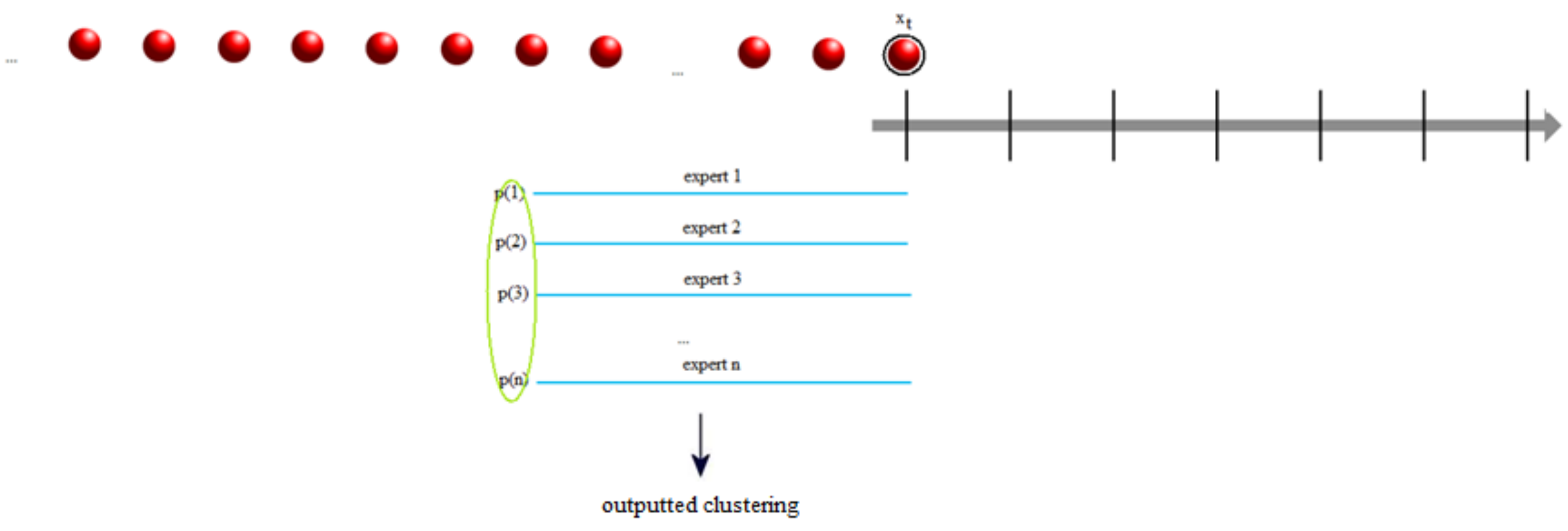
$$p_{t+1}(\alpha) \propto p_t(\alpha) e^{-L(\alpha,t)}$$

$$p_{t+1;\alpha}(i) \propto \sum_j p_t(j) e^{-L(j,t)} p(i|j;\alpha)$$

Online clustering with experts

Algorithm produces clustering informed by experts' clusterings:

- Clustering "experts" output centers at each time t.
- Approximation assumptions on the batch clustering algorithms used as experts yields novel online approximation guarantees.
- At time t, algorithm receives experts' clusterings and outputs a clustering informed by experts.



Analysis ideas

- Prove clustering analogs of regret bounds.
  - Define clustering and loss functions. $L(x_t, c_t) = \left\| \frac{x_t - c_t}{2R} \right\|^2$

    $clust(t) = \sum_{i=1}^{n} p_t(i) c_t^i.$
  - Prove (c,η)-realizability to relate our loss to log-loss.

- Instantiate experts as (batch) clustering algorithms with b-approximation assumptions, run on sliding window
  - Starting from regret bounds, extend with approximation assumptions to yield novel online approximation guarantees.

Performance Guarantees

- Static expert: $L_T(\boldsymbol{alg}) \leq L_T(a_i*) + 2\log n$

$$L_T(alg) \leq \frac{bW}{4R^2} OPT_T + 2\log n$$

- Fixed-share: $L_T^{\log}(\alpha) \leq L_T^{\log}(\alpha^*) + (T-1)D(\alpha^* \| \alpha)$

$$L_T(\alpha) \leq \frac{bW}{4R^2} OPT_T + 2(T-1)D(\alpha^* \| \alpha)$$

- Learn-α:

$$L_T^{\log}(\boldsymbol{alg}) \leq L_T^{\log}(\alpha^*) + (T-1) \min_{\{\alpha_j\}} D(\alpha^* \| \alpha_j) + \log m$$

$$L_T(\boldsymbol{alg}) \leq \frac{bW}{4R^2} OPT_T + 2(T-1) \min_{\{\alpha_j\}} D(\alpha^* \| \alpha_j) + \log m$$

Results: final k-means cost

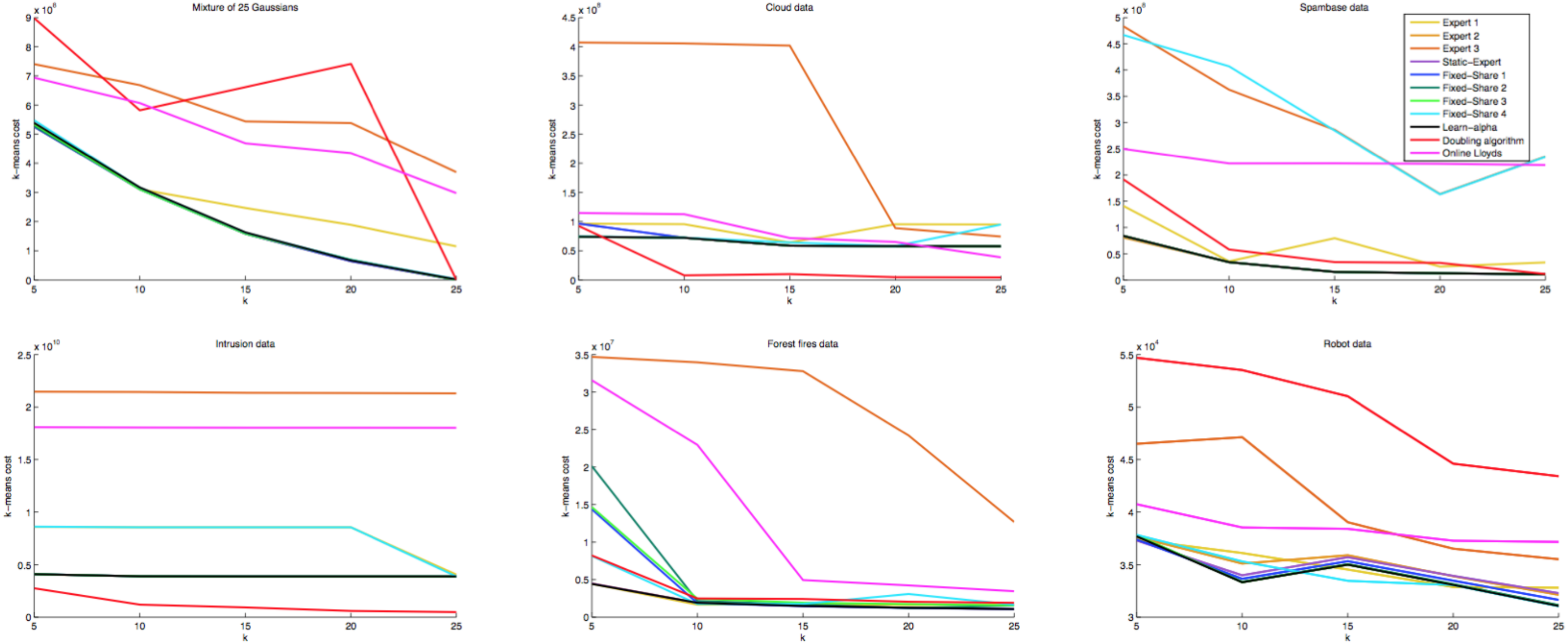


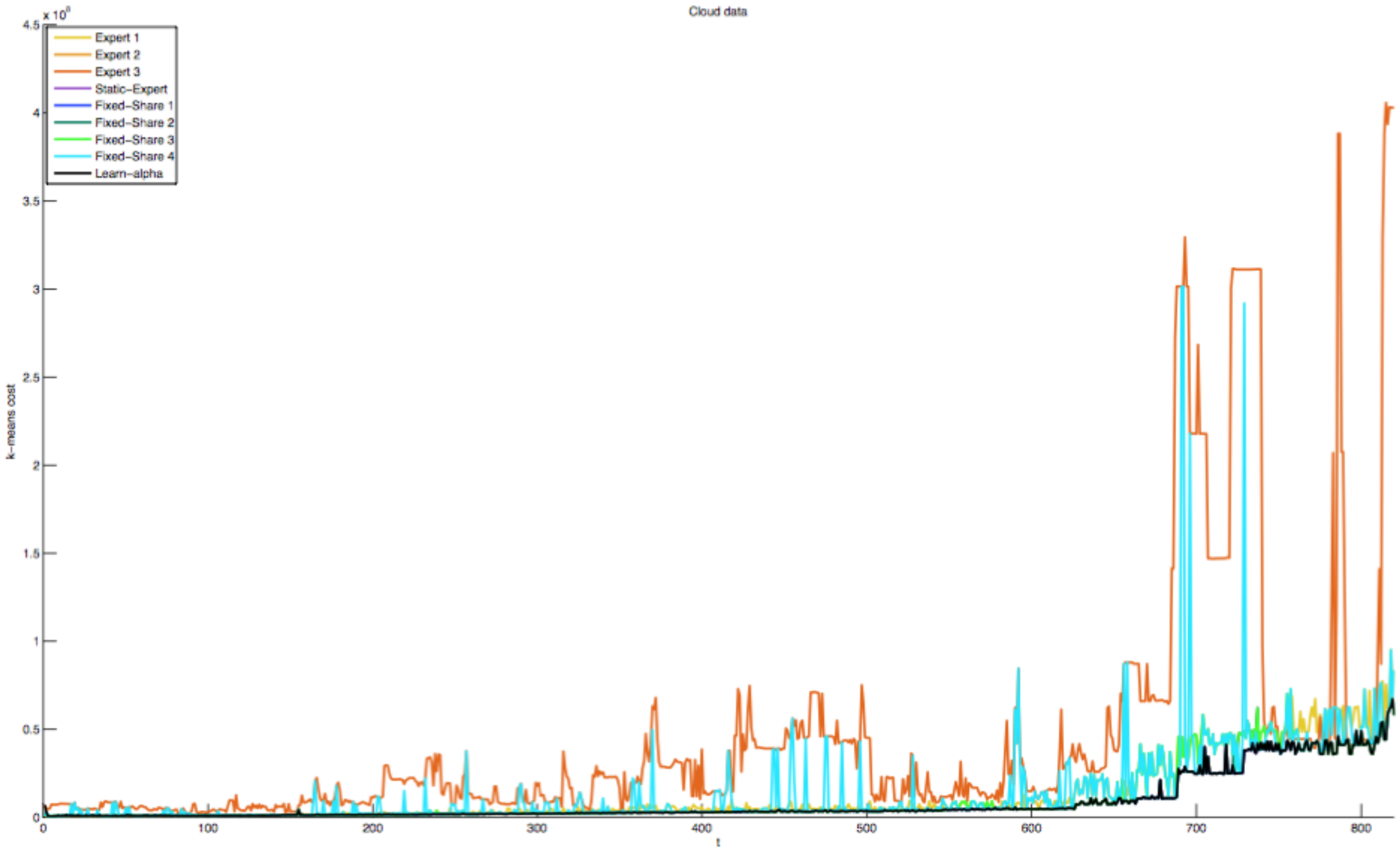
Figure 1: k-means cost on the entire sequence, versus k, per experiment. Legend in upper right.

Results: mean cost over sequence

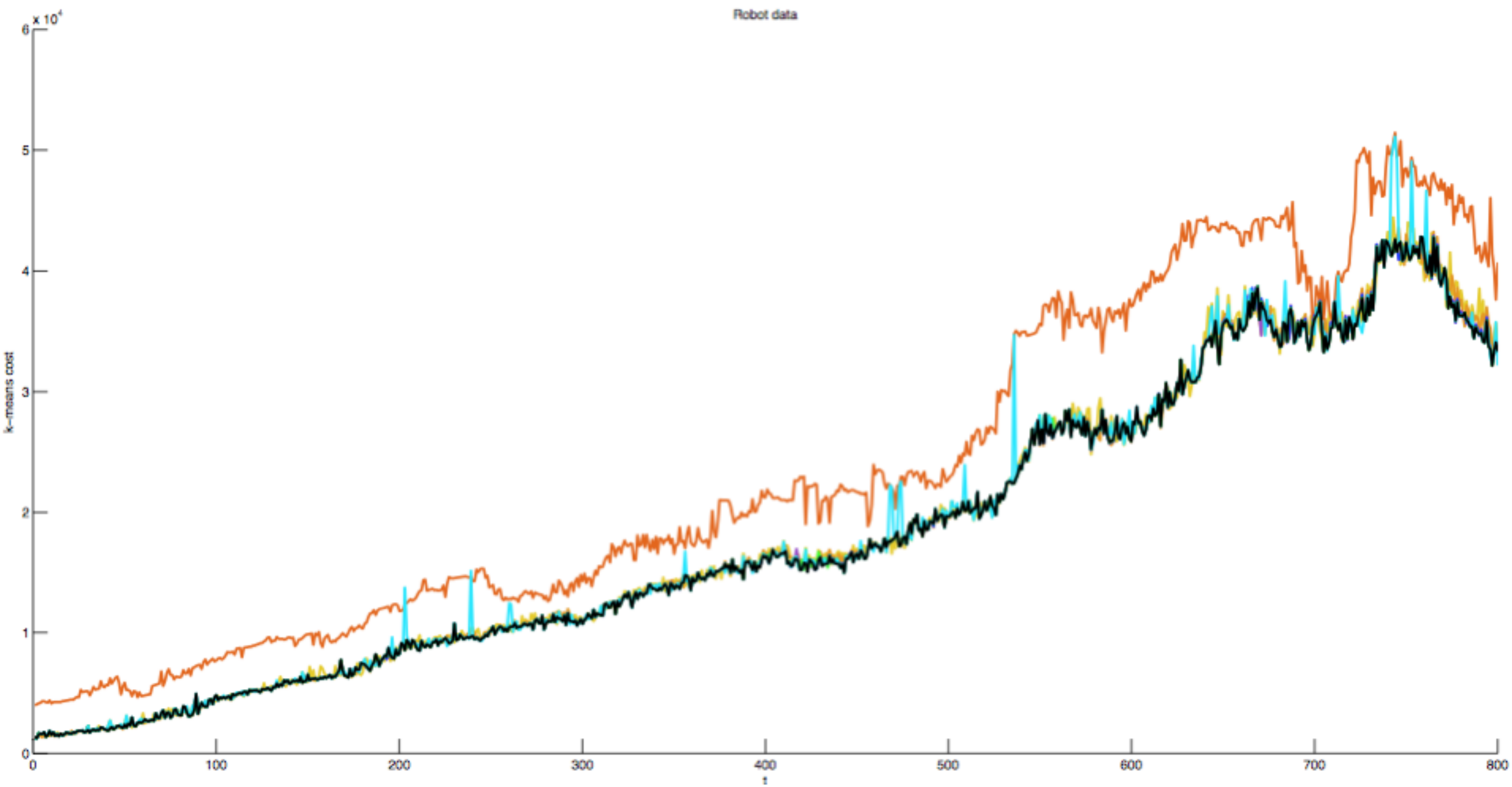
	25 Gaussians	Cloud $\times 10^7$	Spam $\times 10^8$	Intrus. $\times 10^{10}$	For. fire $\times 10^6$	Robot $\times 10^4$
e_1	0.6193±0.3195×10^8	1.3180±1.9395	0.2706±0.2793	0.1988±0.2104	0.7766±0.6413	1.8362±1.2172
e_2	**0.0036**±0.0290×10^7	**0.8837**±1.3834	**0.1042**±0.1463	**0.0743**±0.1041	**0.6616**±0.4832	**1.8199**±1.2102
e_3	2.0859±0.9204×10^8	4.6601±7.8013	1.6291±1.3292	0.7145±0.5376	7.1172±7.6576	2.3590±1.4070
da	0.0179±0.0723×10^8	**0.5285**±0.2959	0.1971±0.0826	**0.0050**±0.0529	1.4496±0.6484	2.5514±1.4239
ol	1.7714±0.6888×10^8	4.2322±2.4965	0.8222±0.7619	1.3518±0.3827	2.9617±1.3006	1.9806±1.0160
se	**0.0014**±0.0143×10^8	**0.8855**±1.3824	**0.1059**±0.1469	**0.0778**±0.1094	0.6620±0.4831	1.8139±1.2032
f_1	**0.0014**±0.0143×10^8	**0.8855**±1.3823	**0.1059**±0.1469	0.0779±0.1100	**0.6614**±0.4819	1.8137±1.2032
f_2	**0.0014**±0.0143×10^8	0.9114±1.4381	**0.1059**±0.1470	**0.0778**±0.1099	0.7008±0.5382	**1.8134**±1.2031
f_3	**0.0014**±0.0143×10^8	1.0715±1.6511	**0.1059**±0.1470	0.0779±0.1099	0.6996±0.5361	1.8145±1.2031
f_4	0.0124±0.0193×10^8	1.4806±2.6257	0.3723±0.7351	0.1803±0.2358	1.0489±1.4817	1.8334±1.2212
f_5	1.3811±1.0881×10^8	3.0837±6.3553	0.8212±1.1583	0.4126±0.5040	4.4481±6.2816	2.2576±1.3849
la	**0.0012**±0.0136×10^8	0.8862±1.3920	**0.1076**±0.1483	0.0785±0.1108	**0.6616**±0.4805	**1.8130**±1.2026
e_4	**7.3703**±4.2635×10^3	**0.6742**±1.2301	**0.0687**±0.1355	**0.0704**±0.1042	**0.2316**±0.2573	**1.3667**±1.0176
e_5	8.2289±4.4386×10^3	0.6833±1.2278	0.0692±0.1356	**0.0704**±0.1042	0.2625±0.2685	1.4385±1.0495
e_6	9.8080±4.7863×10^3	0.7079±1.2364	0.0710±0.1360	0.0705±0.1042	0.3256±0.2889	1.5713±1.1011
se	**0.1360**±1.4323×10^6	**0.6743**±1.2300	**0.0687**±0.1355	**0.0705**±0.1045	0.2322±0.2571	1.3642±1.0138
f_1	**0.1360**±1.4323×10^6	**0.6743**±1.2300	**0.0687**±0.1355	**0.0705**±0.1045	0.2322±0.2571	1.3640±1.0135
f_2	0.1361±1.4322×10^6	0.6746±1.2298	**0.0687**±0.1355	**0.0705**±0.1045	0.2322±0.2572	1.3636±1.0130
f_3	0.1364±1.4322×10^6	**0.6743**±1.2300	**0.0687**±0.1355	0.0711±0.1055	**0.2321** ±0.2570	**1.3634**±1.0127
f_4	0.0027±0.0144×10^8	0.7207±1.3025	**0.0707**±0.1357	0.0773±0.1203	0.2776±0.4917	1.3963±1.0339
f_5	1.4039±1.0790×10^8	3.0786±6.4109	0.7155±1.0650	0.4227±0.5179	4.6103±6.3019	2.3142±1.4127
la	**0.0012**±0.0134×10^8	**0.6742**±1.2300	**0.0687**±0.1355	**0.0708**±0.1046	**0.2318**±0.2573	**1.3632**±1.0128

Table 1: Mean and standard deviation, over the sequence, of k-means cost on points seen so far. $k = 25$ for Gaussians, $k = 15$ otherwise. The best expert and the best 2 scores of the algorithms, per experiment, are bold. Below the triple lines, 3 more experts are added to the ensemble.

Clustering analogs to learning curves



Clustering analogs to learning curves



Future work on clustering data streams

- Online clustering with experts, where experts need not be clustering algorithms

- A negative result for Dasgupta's conjecture (framework 1).

- Other open problems in online clustering
  - Online spectral clustering
  - Hierarchical clustering with k-means approximation guarantees for all k simultaneously
  - How to allow k to vary with time-varying data
  - Your suggestions?

Thank you!

And many thanks to my coauthors:

"Streaming k-means approximation"

Nir Ailon, Technion

Ragesh Jaiswal, IIT Delhi

"Online Clustering with Experts"

Anna Choromanska, Columbia