Sensitivity Sampling with Predictions for k-Means Clustering

Cristian Boldrin Fabio Vandin

University of Padova, Italy

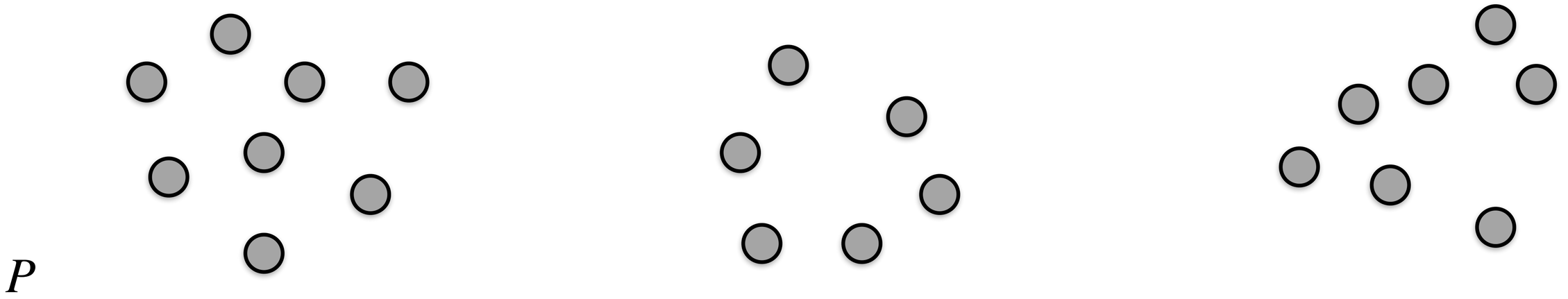
cristian.boldrin.2@phd.unipd.it

ECML PKDD
Naples
September, 2026

Problem: Euclidean k-Means Clustering

Problem: Euclidean k-Means Clustering

Input: set P of n points in $\mathbb{R}^d$; integer k



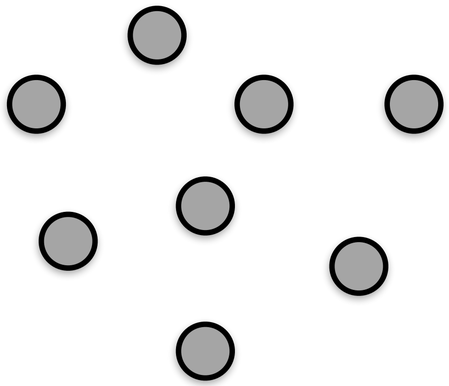
P

Problem: Euclidean k-Means Clustering

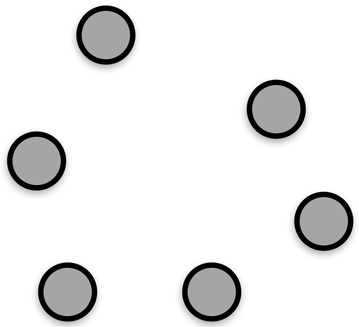
Input: set P of n points in $\mathbb{R}^d$; integer k

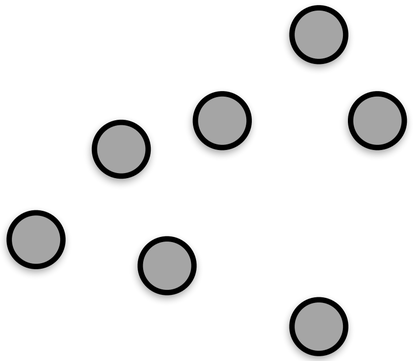
Output: set $S = \{c_1, c_2, \ldots, c_k\}$ of centers which minimizes:

$$\sum_{p \in P} \min_{c \in S} ||p - c||^2$$



P



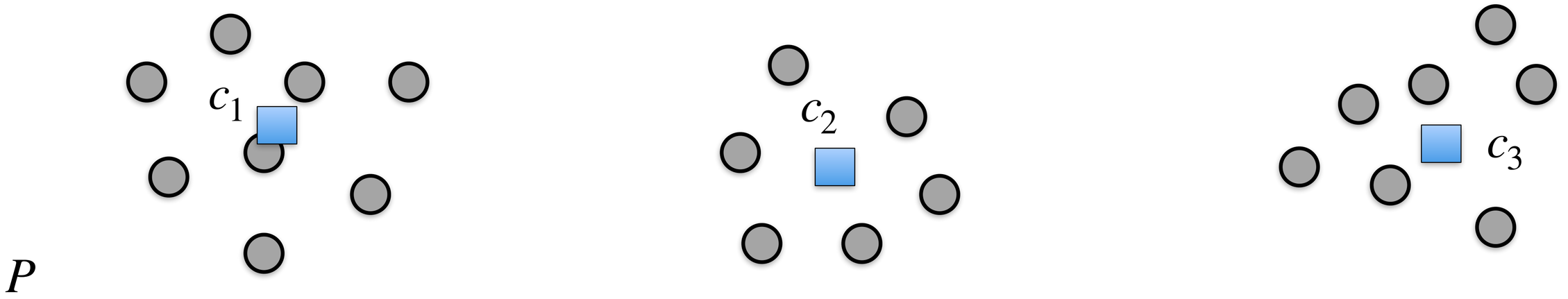


Problem: Euclidean k-Means Clustering

Input: set P of n points in $\mathbb{R}^d$; integer k

Output: set $S = \{c_1, c_2, \ldots, c_k\}$ of centers which minimizes:

$$\sum_{p \in P} \min_{c \in S} ||p - c||^2$$

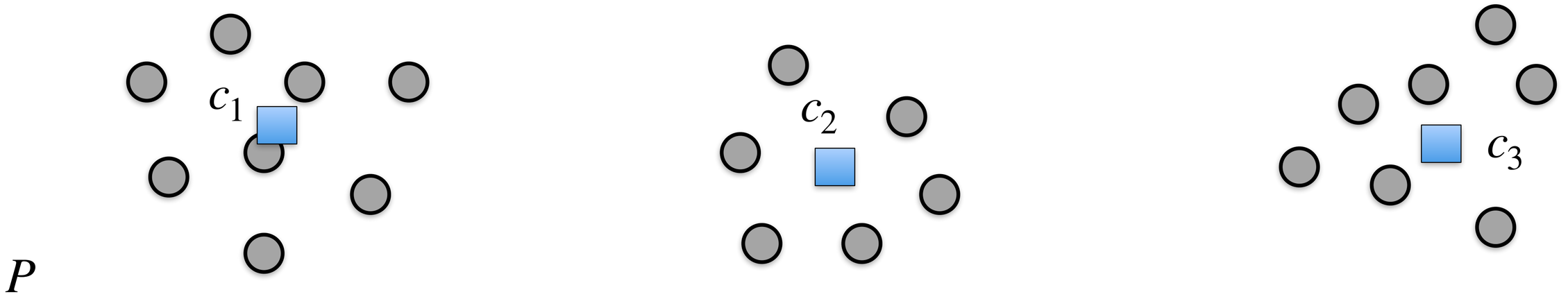


Problem: Euclidean k-Means Clustering

Input: set P of n points in $\mathbb{R}^d$; integer k

Output: set $S = \{c_1, c_2, \ldots, c_k\}$ of centers which minimizes:

$$cost(P, S) = \sum_{p \in P} cost(p, S)$$

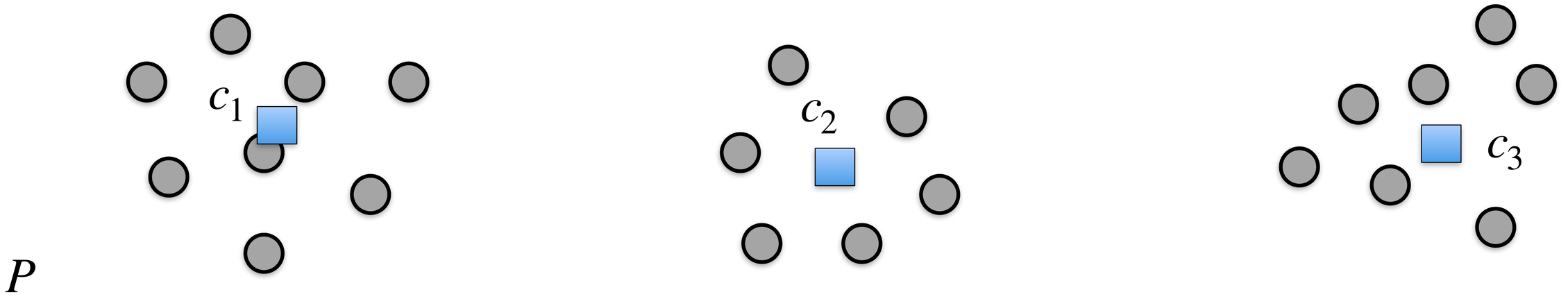


Problem: Euclidean k-Means Clustering

Input: set P of n points in $\mathbb{R}^d$; integer k

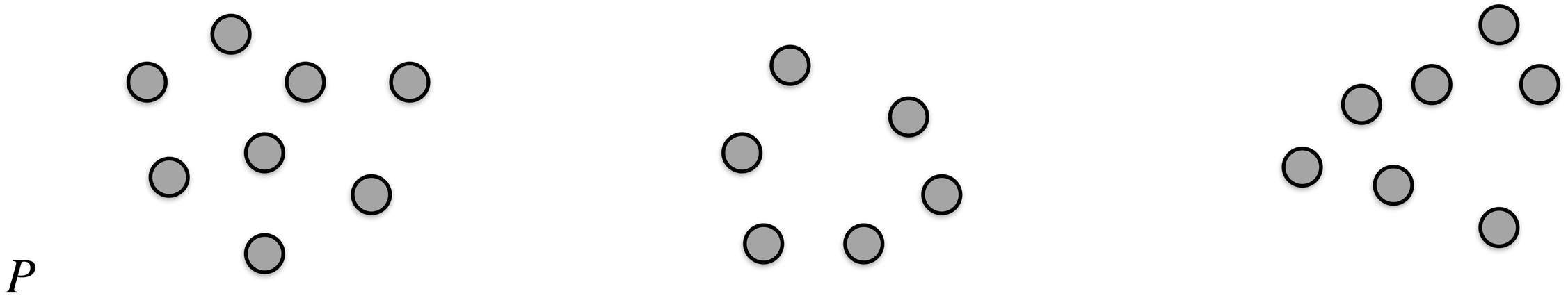
Output: set $S = \{c_1, c_2, \ldots, c_k\}$ of centers which minimizes:

$$OPT_k(P) = \min_{S:\ |S|=k} cost(P, S)$$



Coresets-based approach

- *Coreset:* Small weighted subset of original points which preserves *"cluster structure"*

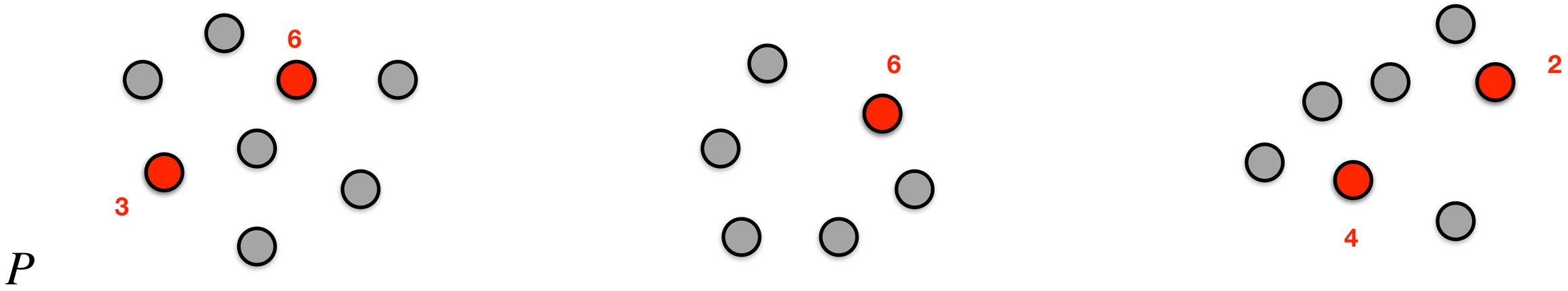


P

Coresets-based approach

- *Coreset:* Small weighted subset of original points which preserves *"cluster structure"*

Coreset $\Omega \subseteq P$

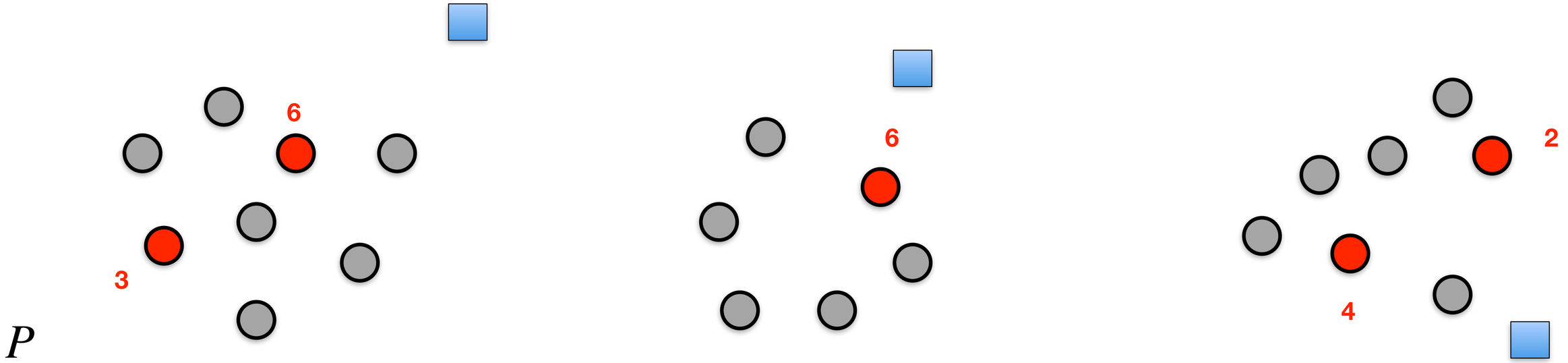


Coresets-based approach

- *Coreset:* Small weighted subset of original points which preserves *"cluster structure"*

Coreset $\Omega \subseteq P$

Set S of k centers



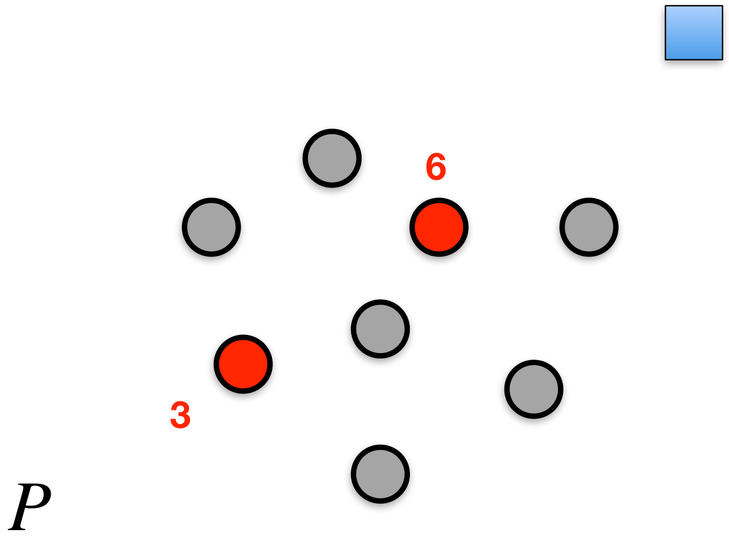
Coresets-based approach

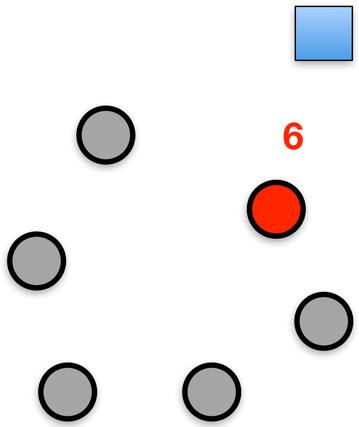
- *Coreset:* Small weighted subset of original points which preserves *"cluster structure"*

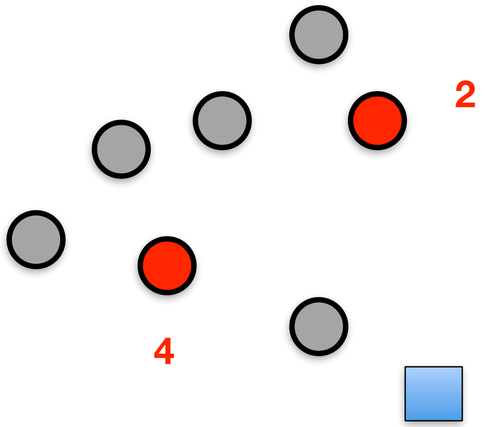
Coreset $\Omega \subseteq P$

Set S of k centers

Coreset cost: $cost(\Omega, S) = \sum_{q \in \Omega} w_q \, cost(q, S)$





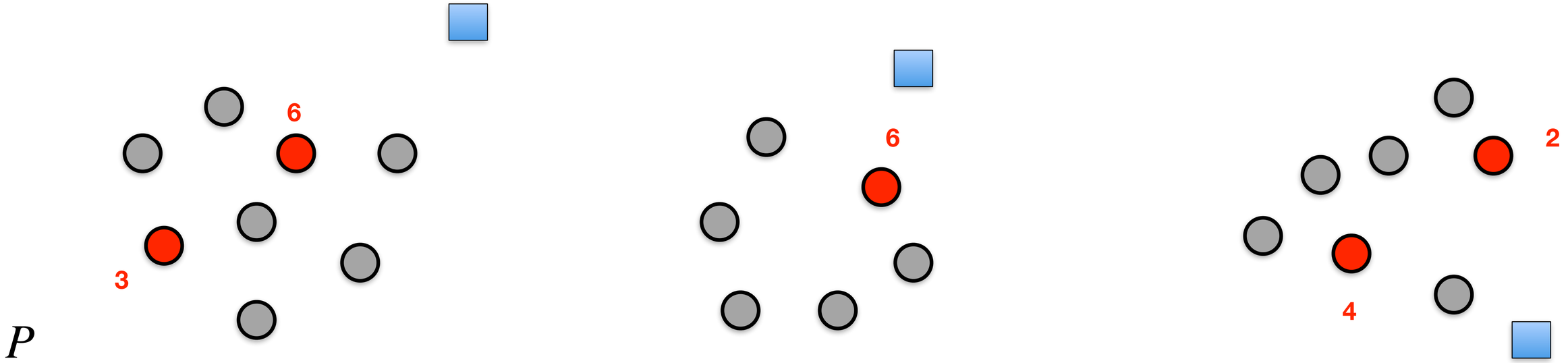


Definition: coreset

A weighted <u>subset</u> Ω of the input P which satisfies:

$$cost(\Omega, S) \in (1 \pm \varepsilon)\ cost(P, S)$$

simultaneously for all set S of k centers



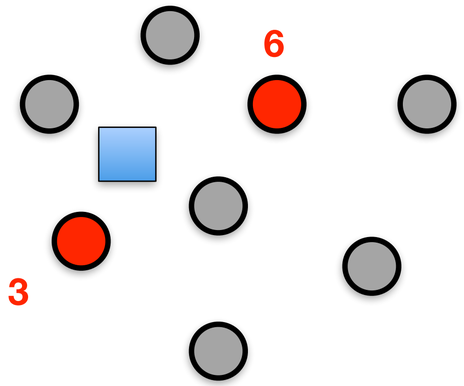
P

Definition: coreset

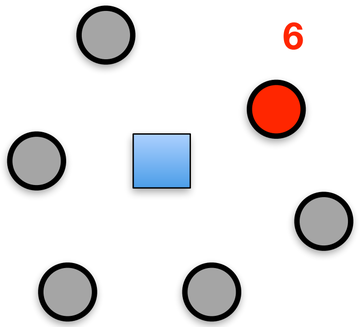
A weighted <u>subset</u> Ω of the input P which satisfies:

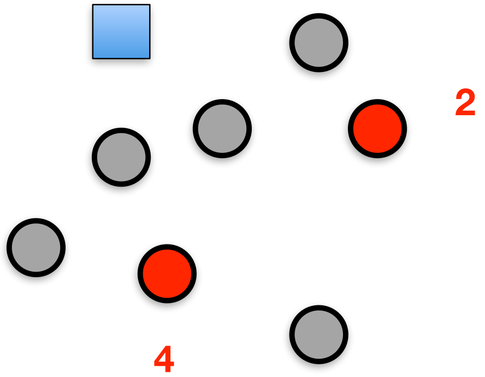
$$cost(\Omega, S) \in (1 \pm \varepsilon)\ cost(P, S)$$

simultaneously for all set S of k centers



P



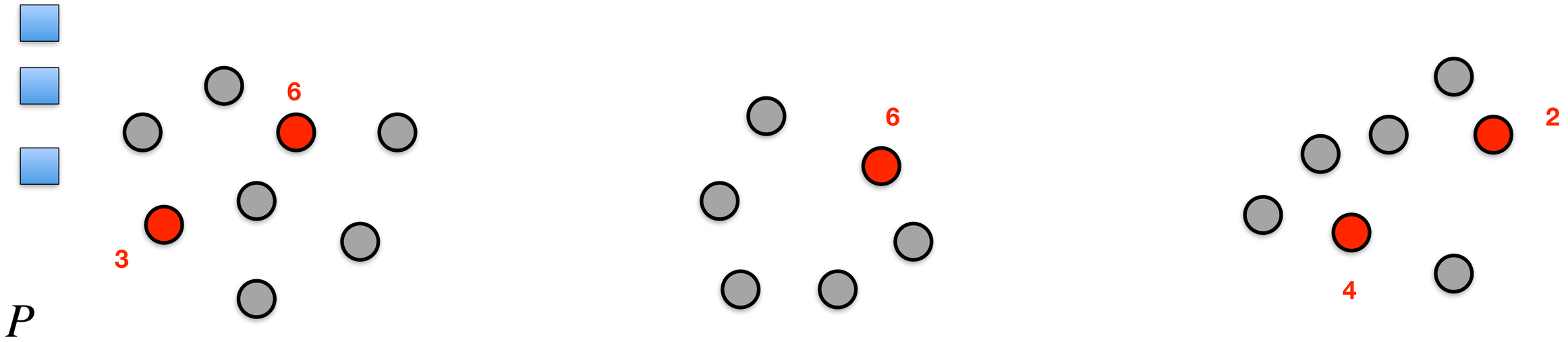


Definition: coreset

A weighted <u>subset</u> Ω of the input P which satisfies:

$$cost(\Omega, S) \in (1 \pm \varepsilon)\ cost(P, S)$$

simultaneously for all set S of k centers



Goal

Efficiently design algorithms to produce coresets of *small* size

Goal

Efficiently design algorithms to produce coresets of *small* size

Applications:

- Pre-processing step for speeding up clustering
- Memory constraints: distributed computing, streaming algorithms
- Dynamic algorithms

Goal

Efficiently design algorithms to produce coresets of *small* size

Applications:

- Pre-processing step for speeding up clustering
- Memory constraints: distributed computing, streaming algorithms
- Dynamic algorithms

We show that using *predictions* (e.g., information from historical data) *accelerates* the coreset construction!

First attempt: Uniform Sampling

Coreset Algorithm

- Sample m points independently and uniformly at random from P
- Assign weight n/m to each sampled point

First attempt: Uniform Sampling

Coreset Algorithm

- Sample m points independently and uniformly at random from P
- Assign weight n/m to each sampled point

How many points are required?

First attempt: Uniform Sampling

Coreset Algorithm

- Sample m points independently and uniformly at random from P
- Assign weight n/m to each sampled point

How many points are required?

P

First attempt: Uniform Sampling

Coreset Algorithm

- Sample m points independently and uniformly at random from P
- Assign weight n/m to each sampled point

How many points are required?

P

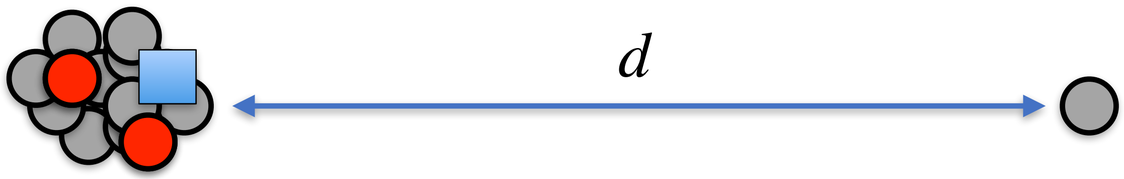
First attempt: Uniform Sampling

Coreset Algorithm

- Sample m points independently and uniformly at random from P
- Assign weight n/m to each sampled point

How many points are required?

Coreset must pick the point on the right (otw: *true cost* $\approx d^2$ but *coreset cost* ≈ 0)



P

First attempt: Uniform Sampling

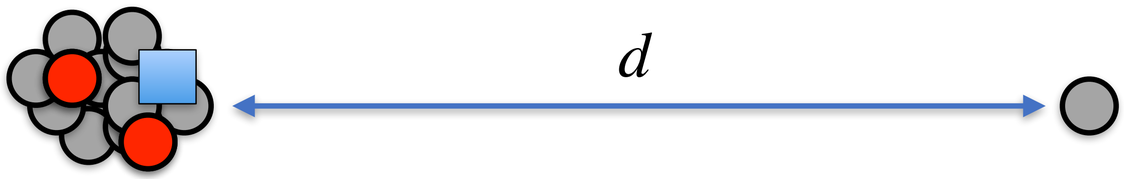
Coreset Algorithm

- Sample m points independently and uniformly at random from P
- Assign weight n/m to each sampled point

How many points are required?

Uniform Sampling:
need to take $\Omega(n)$ points!

Coreset must pick the point on the right (otw: *true cost* $\approx d^2$ but *coreset cost* ≈ 0)



P

First attempt: Uniform Sampling

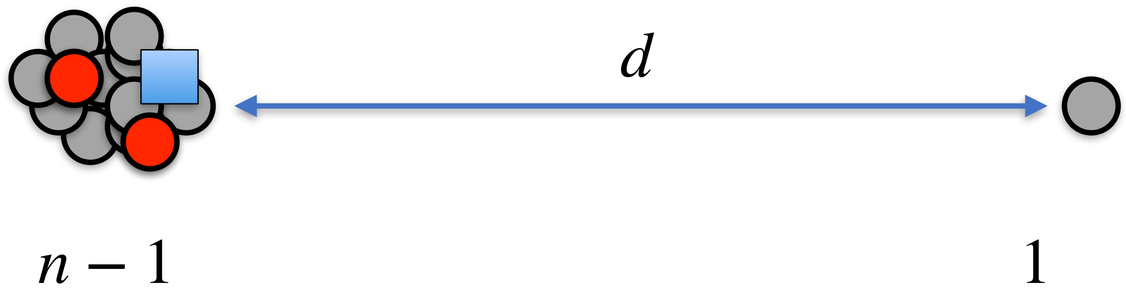
Coreset Algorithm

- Sample m points independently and uniformly at random from P
- Assign weight n/m to each sampled point

How many points are required?

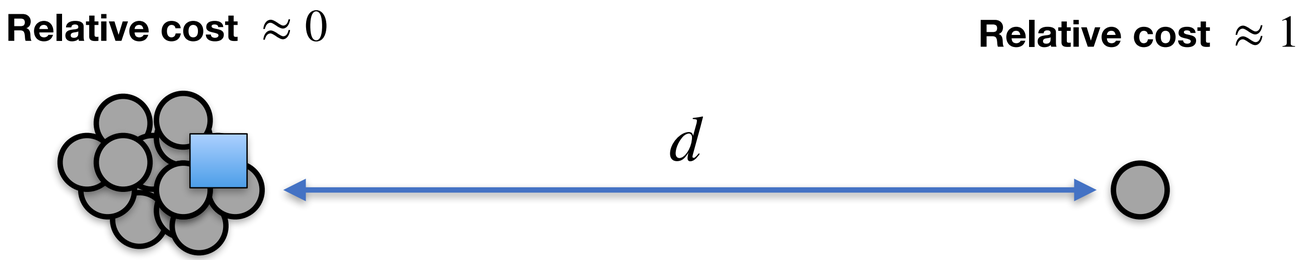
Takeaway: some points are more *"important"* than others

Coreset must pick the point on the right (otw: *true cost* $\approx d^2$ but *coreset cost* ≈ 0)



Identifying important points

Quantify *importance* of point p as "relative cost"



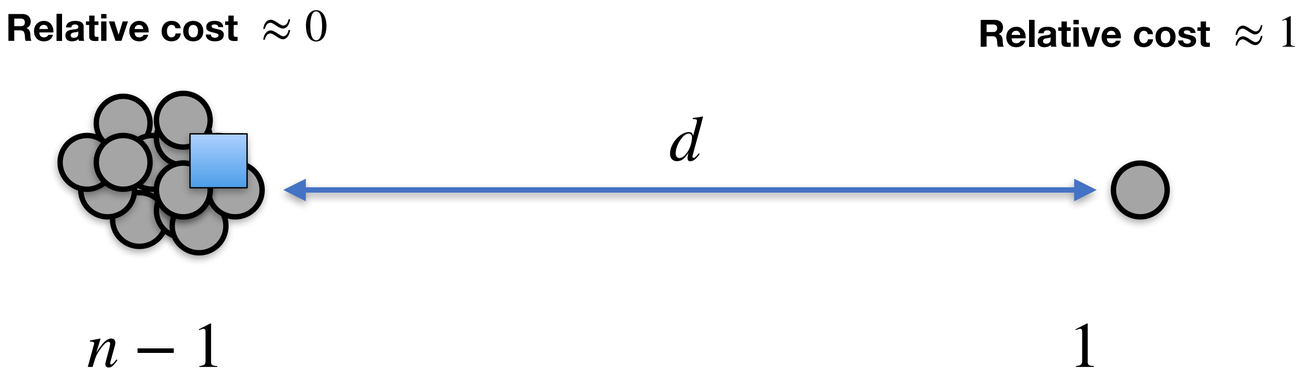
P

Identifying important points

Quantify *importance* of point p as "relative cost"

Definition (*sensitivity*):

A point $p \in P$ has sensitivity $\sigma(p) = \sup_{S:\ |S| = k} \dfrac{cost(p, S)}{cost(P, S)}$



Sensitivity Sampling

Coreset Algorithm

Sensitivity: $\sigma(p) = \sup_{S:\ |S|=k} \frac{cost(p, S)}{cost(P, S)}$

Sensitivity Sampling

Coreset Algorithm

Sensitivity: $\sigma(p) = \sup_{S:\ |S|=k} \frac{cost(p,S)}{cost(P,S)}$

- Sample m points independently with probability proportional to $\mathbb{P}[p] \propto \sigma(p)$

Sensitivity Sampling

Sensitivity: $\sigma(p) = \sup_{S:\ |S|=k} \frac{cost(p,S)}{cost(P,S)}$

Coreset Algorithm

- Sample m points independently with probability proportional to $\mathbb{P}[p] \propto \sigma(p)$
- Assign weight $1/\left(m\ \mathbb{P}[p]\right)$ to each sampled point

Sensitivity Sampling

Sensitivity: $\sigma(p) = \sup_{S:\ |S|=k} \frac{cost(p,S)}{cost(P,S)}$

Coreset Algorithm

- Sample m points independently with probability proportional to $\mathbb{P}[p] \propto \sigma(p)$
- Assign weight $1/\left(m\ \mathbb{P}[p]\right)$ to each sampled point

Computing (exact) sensitivities is *intractable*

Sensitivity Sampling

Sensitivity: $\sigma(p) = \sup_{S:\ |S|=k} \frac{cost(p,S)}{cost(P,S)}$

Coreset Algorithm

- Sample m points independently with probability proportional to $\mathbb{P}[p] \propto \sigma(p)$
- Assign weight $1/\left(m\ \mathbb{P}[p]\right)$ to each sampled point

Computing (exact) sensitivities is *intractable*

➡ Work with *approximation* of sensitivities

Sensitivity Sampling

Define (α, β) bi-criteria approximation A:

- Yields α-approximation, i.e., $cost(P, A) \leq \alpha \cdot OPT_k(P)$
- Uses at most βk centers

State-of-the-art [Bansal et al., FOCS 2024]

Compute $\left(O(1), O(1)\right)$ bi-criteria approximation A.

State-of-the-art [Bansal et al., FOCS 2024]

Compute $\left(O(1), O(1)\right)$ bi-criteria approximation A. For C_j's induced by A and Δ_j its average cost, setting for each $p \in C_j$:

$$\mathbb{P}[p] \propto \frac{cost(p, A)}{cost(P, A)} + \frac{1}{|A| \cdot |C_j|} + \frac{cost(p, A)}{|A| \cdot cost(C_j, A)} + \frac{\Delta_j}{cost(P, A)}$$

State-of-the-art [Bansal et al., FOCS 2024]

Compute $\left(O(1), O(1)\right)$ bi-criteria approximation A. For C_j's induced by A and Δ_j its average cost, setting for each $p \in C_j$:

$$\mathbb{P}[p] \propto \frac{cost(p, A)}{cost(P, A)} + \frac{1}{|A| \cdot |C_j|} + \frac{cost(p, A)}{|A| \cdot cost(C_j, A)} + \frac{\Delta_j}{cost(P, A)}$$

yields (theoretically *optimal*) coreset size of

$m = \widetilde{O}\left(k\varepsilon^{-2} \cdot \min\left(\sqrt{k}, \varepsilon^{-2}\right)\right)$ with constant probability.

State-of-the-art [Bansal et al., FOCS 2024]

Compute $\left(O(1), O(1)\right)$ bi-criteria approximation A.

In practice: run *kmeans*++ with $2k$ centers

Bottleneck of the coreset algorithm:

$$\Theta(nkd)$$

Sensitivity Sampling with Predictions

Compute $\left(O(1), O(1)\right)$ bi-criteria approximation A.

Our Algorithm:

Predictions: A is given in input as set of centers

Sensitivity Sampling with Predictions

Compute $\left(O(1), O(1)\right)$ bi-criteria approximation A.

Our Algorithm:

Predictions: A is given in input as set of centers

Time complexity is still $\Theta(nkd)$ (distance computation) but in practice (4-5x) faster

Sensitivity Sampling with Predictions

Compute $\left(O(1), O(1)\right)$ bi-criteria approximation A.

Our Algorithm:

Predictions: A is given in input as set of centers

(Natural setting in applications in which we can use previous information)

Sensitivity Sampling with Predictions

Setting: clustering over related datasets. Assume $P_1, P_2, \ldots$ with points drawn i.i.d. from the same data-generating distribution.

Sensitivity Sampling with Predictions

Setting: clustering over related datasets. Assume $P_1, P_2, \ldots$ with points drawn i.i.d. from the same data-generating distribution.

Theorem (informal):

Let $A \subseteq P_i$ be a (α, β) bi-criteria approximation for P_i .

Then, if P_i and P_j are *"large enough"*, A is a $(O(\alpha), \beta)$ bi-criteria approximation for P_j with constant probability (over the choices of P_i, P_j).

Sensitivity Sampling with Predictions

Setting: clustering over related datasets. Assume $P_1, P_2, \ldots$ with points drawn i.i.d. from the same data-generating distribution.

Theorem (informal):

Let $A \subseteq P_i$ be a (α, β) bi-criteria approximation for P_i .

Then, if P_i and P_j are *"large enough"*, A is a $(O(\alpha), \beta)$ bi-criteria approximation for P_j with constant probability (over the choices of P_i, P_j).

"Good" centers from historical data serve as "good" predictions for future data!

Sensitivity Sampling with Predictions

Even *noisy predictions* suffice!

Sensitivity Sampling with Predictions

Even *noisy predictions* suffice!

Theorem:

If the predicted set A of centers induce an (α, β) bi-criteria approximation and $\alpha, \beta = O\left(poly \log\left(k, \varepsilon^{-1}\right)\right)$, then sensitivity sampling with predictions achieves *optimal* coreset bound.

Sensitivity Sampling with Predictions

Clustering algorithm using coresets.

Sensitivity Sampling with Predictions

Clustering algorithm using coresets.

When clustering datasets represented as sequence of (large enough) snapshots:

Sensitivity Sampling with Predictions

Clustering algorithm using coresets.

When clustering datasets represented as sequence of (large enough) snapshots:

- Compute bi-criteria approximation A on the *first* snapshot

Sensitivity Sampling with Predictions

Clustering algorithm using coresets.

When clustering datasets represented as sequence of (large enough) snapshots:

- Compute bi-criteria approximation A on the *first* snapshot
- Reuse A as predictions to build a (theoretically optimal) coreset for each subsequent snapshot

Sensitivity Sampling with Predictions

Clustering algorithm using coresets.

When clustering datasets represented as sequence of (large enough) snapshots:

- Compute bi-criteria approximation A on the *first* snapshot
- Reuse A as predictions to build a (theoretically optimal) coreset for each subsequent snapshot
- Run clustering algorithm (e.g., Lloyd's) on such (small) coreset to get a solution with guarantees

Experiments

Experiments

Datasets considered.

Dataset	Total Points	# Snapshots	Aggregation	n	d
Twitter	$14M$	7	Daily	$2.2M$	3
IntelLab	$2.2M$	34	Daily	$101k$	6
Taxi	$1.7M$	12	Monthly	$161k$	2
NYC TLC	$47M$	4	Monthly	$12.7M$	16
NYC TLC	$743M$	10	Yearly	$146M$	14

Table 1: Datasets' statistics: total number of points across snapshots; number of snapshots; aggregation period; maximum number n of points over all the snapshots; number d of features.

Experiments

Algorithms considered.

- Baseline: *Uniform-Sampling*

Experiments

Algorithms considered.

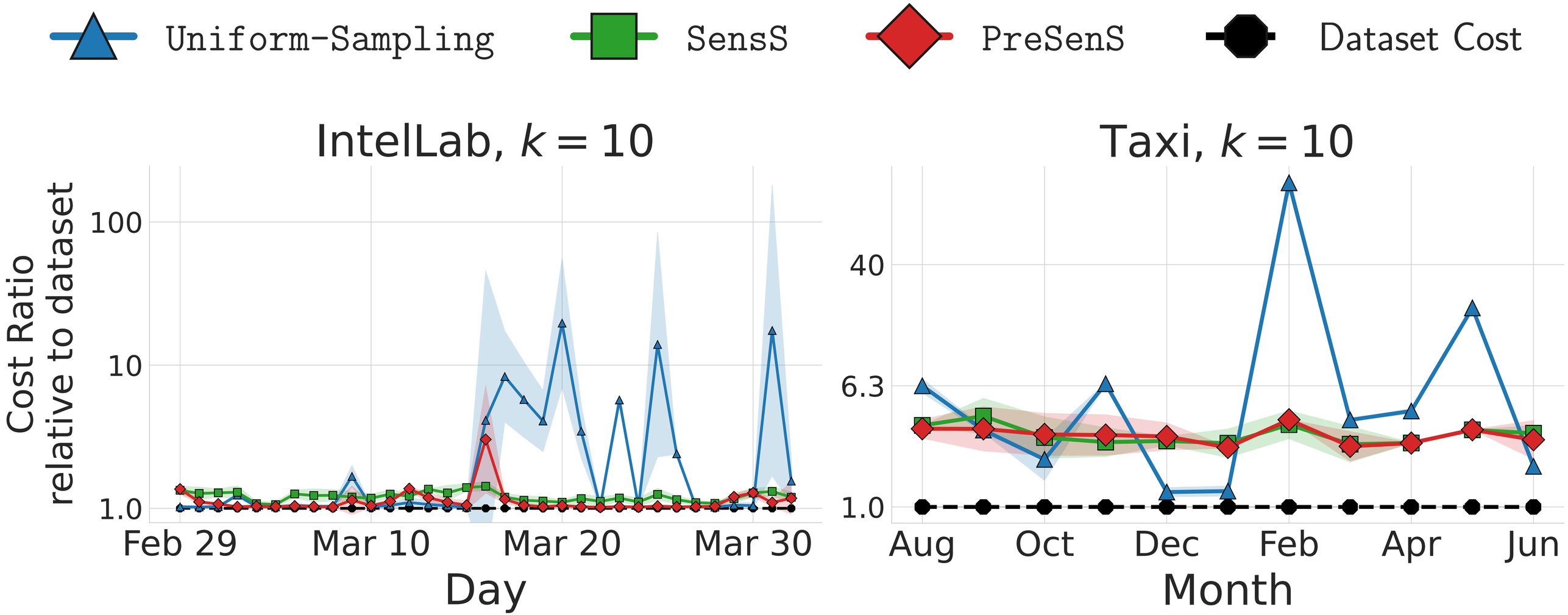
- Baseline: *Uniform-Sampling*
- SOTA: *SensS* [Bansal et al., FOCS 2024] using *kmeans*++ with $2k$ centers for computing approximation (from scratch)

Experiments

Algorithms considered.

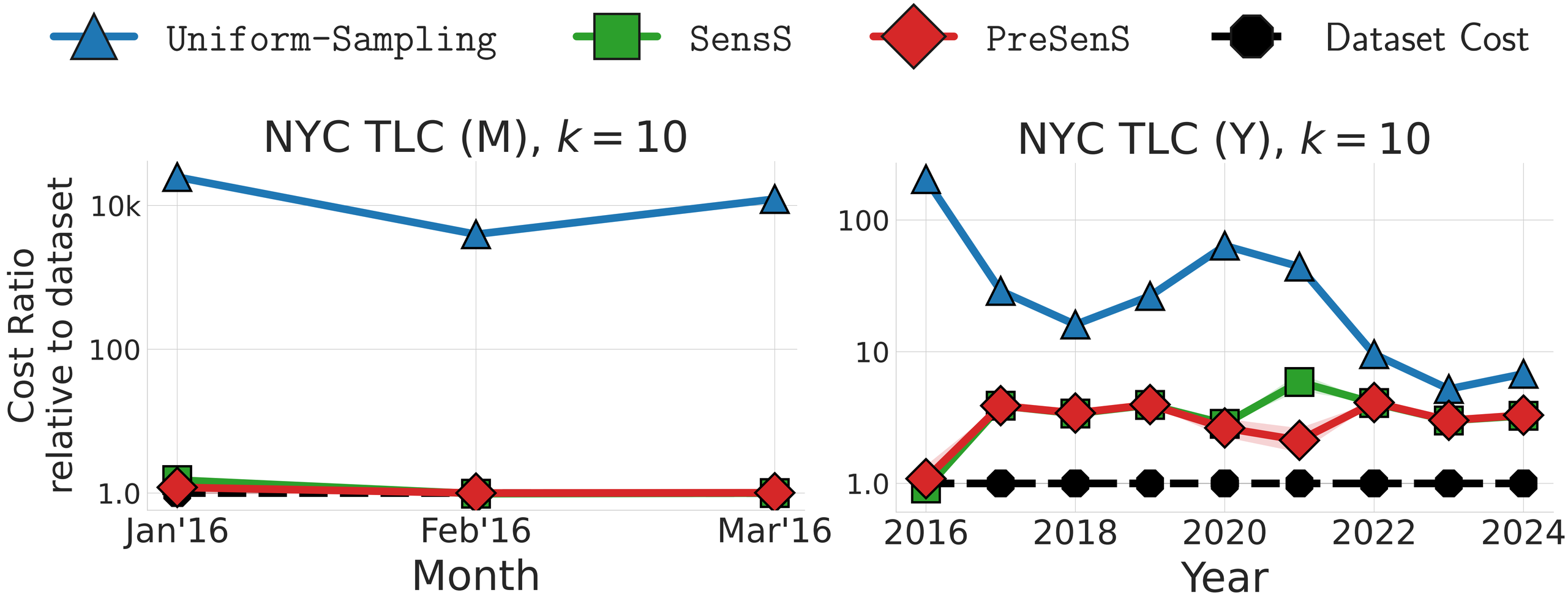
- Baseline: *Uniform-Sampling*
- SOTA: *SensS* [Bansal et al., FOCS 2024] using *kmeans*++ with $2k$ centers for computing approximation (from scratch)
- (Ours) *PreSenS*, where predictions are computed on the *first* snapshot and reused on *all* subsequent ones

Experiments



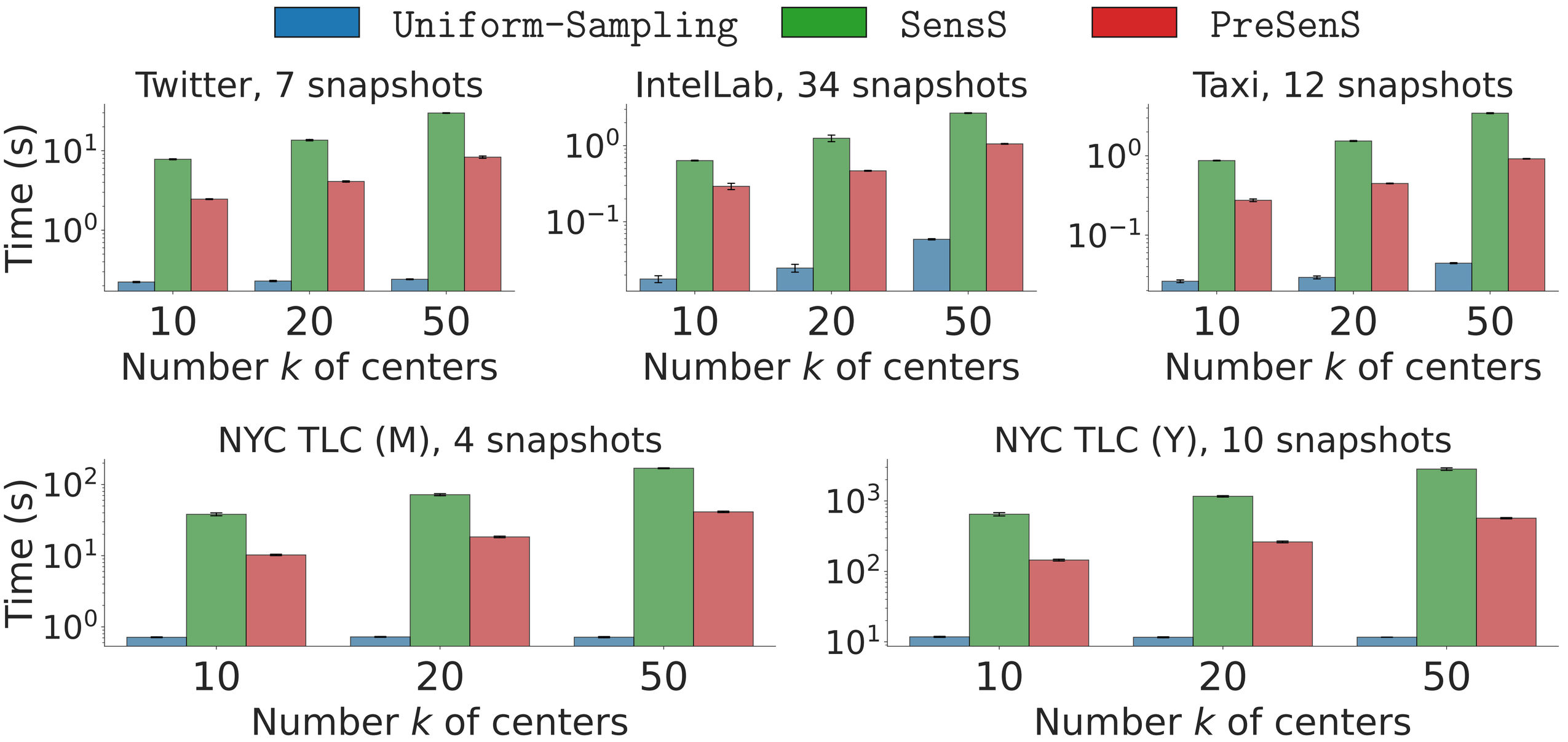
$$m = 50 \cdot k$$

Experiments

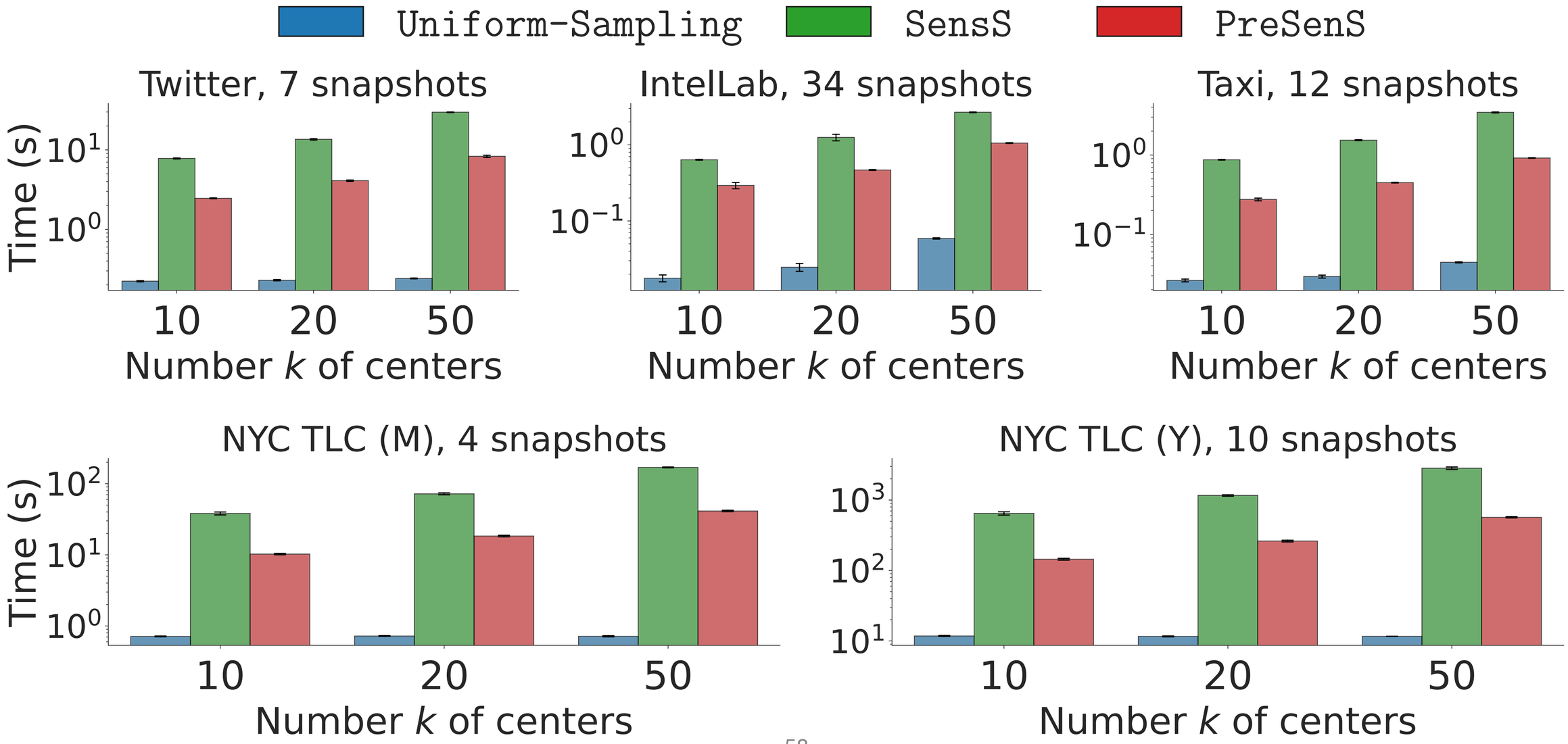


$$m = 50 \cdot k$$

Experiments



Predictions *accelerate* coreset construction by *3.6x* (on average) and up to *5.2x* on largest dataset, while being *comparable* in terms of *clustering quality*



Conclusion

Our contributions.

- *PreSenS,* first approach combining coresets and *predictions* for *speeding up* the runtime of sensitivity-based k-means clustering
- Predictions can be derived *efficiently* with *guarantees* in the context of dynamic application
- *Robustness:* Coreset size matches the state-of-the-art even for inaccurate predictions
- Empirically *PreSenS* is the best algorithm in terms of clustering quality vs runtime

Conclusion

Our contributions.

- *PreSenS,* first approach combining coresets and predictions for speeding up the runtime of sensitivity-based k-means clustering
- Predictions can be derived *efficiently* with *guarantees* in the context of dynamic application
- *Robustness:* Coreset size matches the state-of-the-art even for inaccurate predictions
- Empirically *PreSenS* is the best algorithm in terms of clustering quality vs runtime

Progetto “National Centre for HPC, Big Data and Quantum Computing”, CN00000013 (approvato nell’ambito del Bando M42C – Investimento 1.4 – Avvisto “Centri Nazionali” – D.D. n. 3138 del 16.12.2021, ammesso a finanziamento con Decreto del MUR n. 1031 del 17.06.2022)

C. Boldrin and F. Vandin, **“Sensitivity Sampling with Predictions for k-Means Clustering”**, ECML-PKDD 2026.

Code and extended version of the paper:

arXiv https://arxiv.org/abs/2607.04949

GitHub https://github.com/VandinLab/PreSenS

cristian.boldrin.2@phd.unipd.it

References

Bansal, N., Cohen-Addad, V., Prabhu, M., Saulpic, D., Schwiegelshohn, C.: Sensitivity sampling for k-means: Worst case and stability optimal coreset bounds. FOCS (2024)

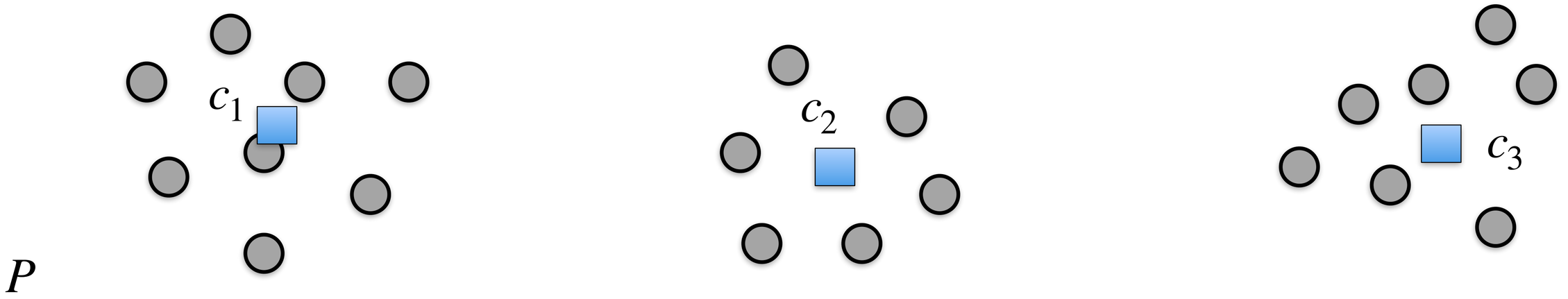
Draganov, A., Saulpic, D., Schwiegelshohn, C.: Settling time vs. accuracy tradeoffs for clustering big data. PACMMOD (2024)

Feldman, D., Langberg, M.: A unified framework for approximating and clustering data. STOC (2011)

Schwiegelshohn, C., Sheikh-Omar, O.A.: An empirical evaluation of k-means coresets. ESA (2022)

Problem: Euclidean k-Means Clustering

- *NP-Hard,* even for $k = 2$ *[Dasgupta, 2008]*
- *NP-Hard* to get approximation ratio < 1.07 *[Cohen-Addad and Karthik, FOCS 2019]*
- Best approximation ratio: 5.912 *[Cohen-Addad et al., STOC 2022]* based on primal-dual and quasi-independent sets (<u>not scalable</u>)



Sensitivity Sampling

Define $\left(\alpha, \beta\right)$ bi-criteria approximation A:

- Yields α-approximation, i.e., $cost(P, A) \leq \alpha \cdot OPT_k(P)$
- Uses at most βk centers

[Feldman and Langberg, STOC 2011]: Compute $\left(O(1), O(1)\right)$ bi-criteria approximation A. Then, setting $\mathbb{P}[p] \propto \frac{cost(p, A)}{cost(P, A)} + \frac{1}{n}$ yields coreset size of $m = \widetilde{O}\left(kd\varepsilon^{-4}\right)$ with constant probability.

Sensitivity Sampling with Predictions

"Large enough": $\Omega\left(\frac{\beta k}{\varepsilon^2 \, OPT_{\beta k}\left(\mathscr{D}\right)}\right)$

Optimum over the distribution:

$$OPT_k\left(\mathscr{D}\right) = \min_{S:\ |S| = k} \mathbb{E}_{p \sim \mathscr{D}}\left[cost(p, S)\right]$$

Relaxing Fixed Distribution Assumption

Setting: clustering over related datasets with points drawn *dynamic* data-generating distribution.

Uniform convergence applies by paying an extra *additive term* depending on the discrepancy (*drift*) of distribution.

Our result still holds when such additive term is *small* compared to the target optimum on the "generalisation" dataset, i.e., $\boldsymbol{OPT}_k\left(P_j\right)$.

Additional Experiments: Estimated Distortion

Evaluate distortion of coresets.

To empirically probe *strong* coreset guarantees, we generate in $\mathscr{C}$ many sets of centers (via *kmeans++*, *random*, *CH*, *MEB*).

We compute the estimated distortion as:

$$\max_{S \in \mathscr{C}} \max \left(\frac{cost(P,S)}{cost_{\Omega}(P,S)}, \frac{cost_{\Omega}(P,S)}{cost(P,S)} \right)$$

Experiments

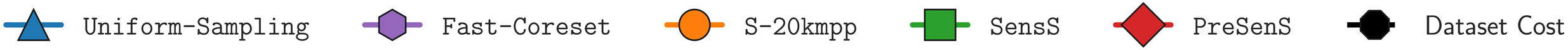
Results.

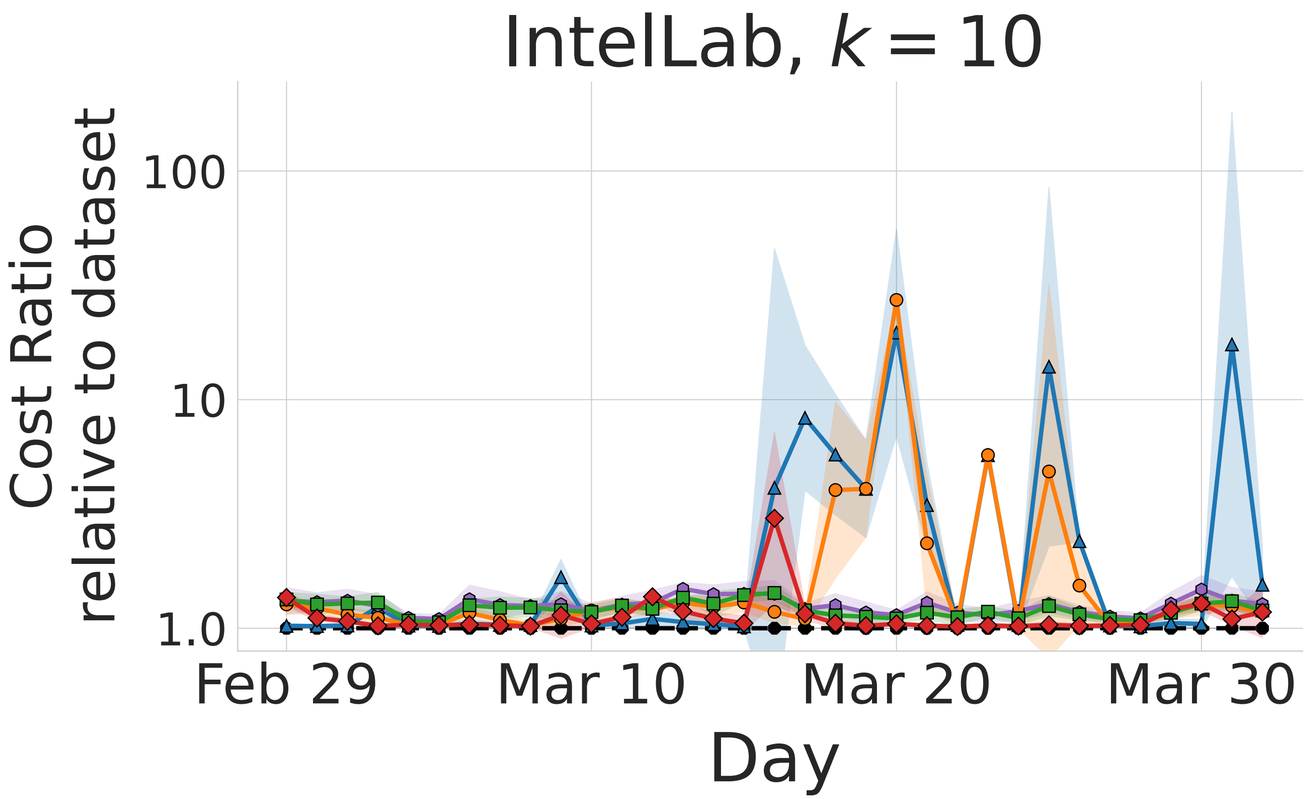
As in previous work [Schwiegelshohn and Sheikh-Omar, ESA 2022]

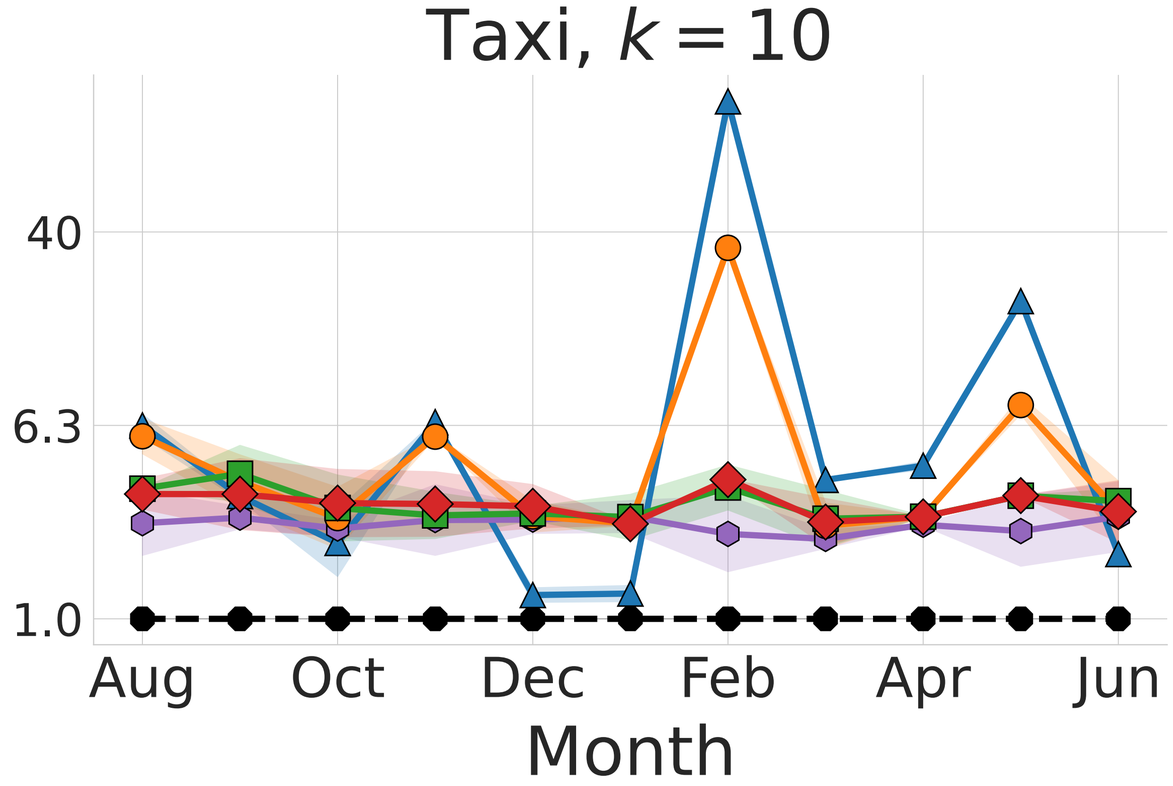
- Run algorithms with number k of centers $k \in \{10, 20, 30, 50\}$
- Run algorithms for fixed coreset size $m \in \{50, 200, 500\} \cdot k$

For sake of time, only *partial* results are presented, with $m = 50 \cdot k$

Experiments

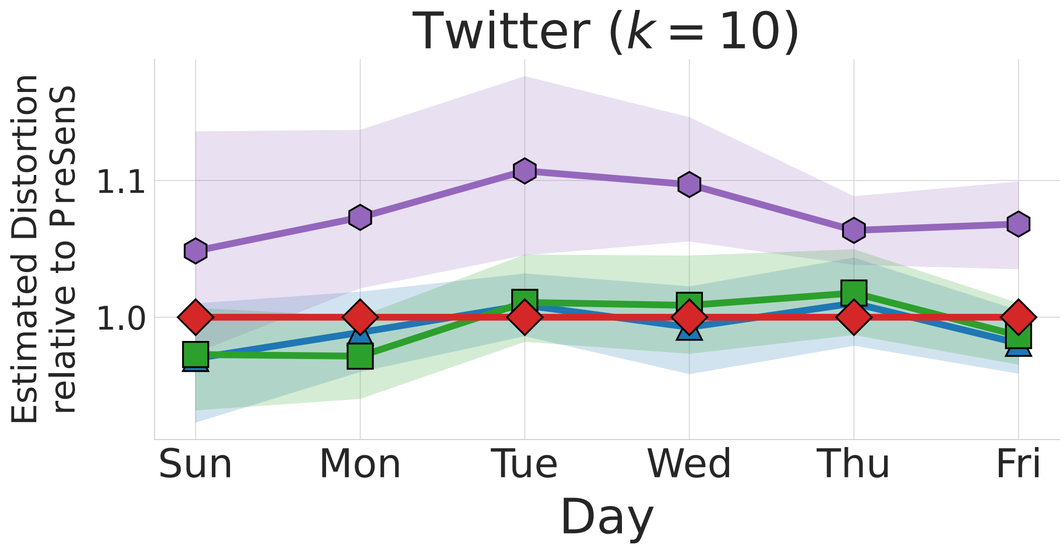


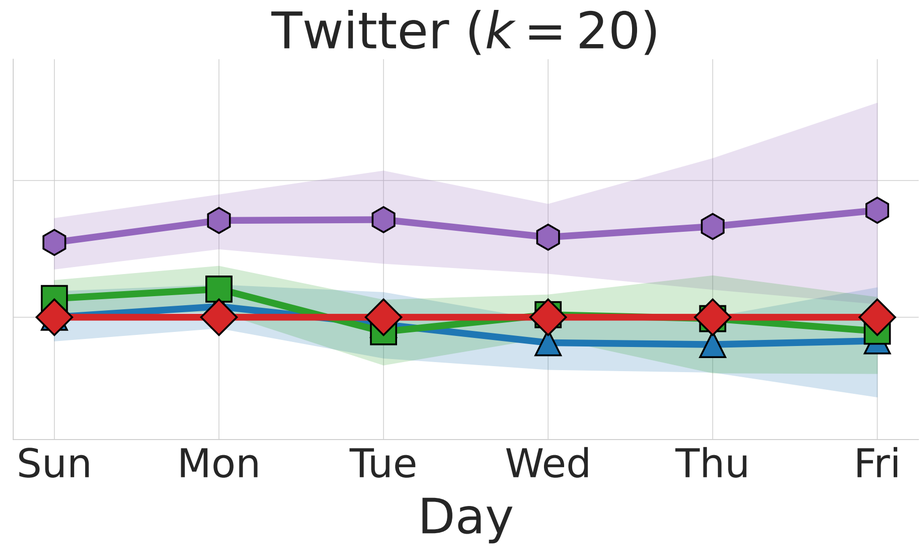


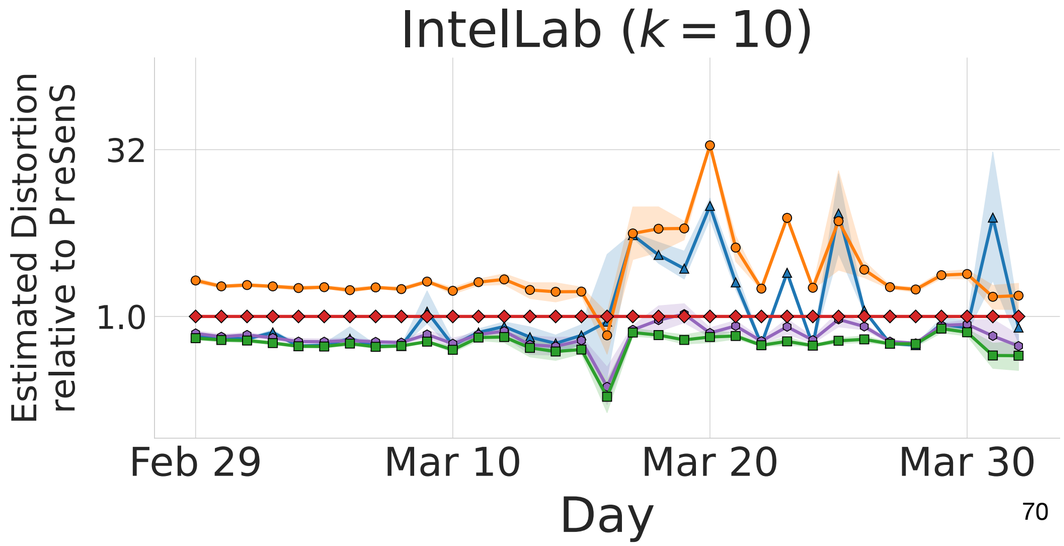


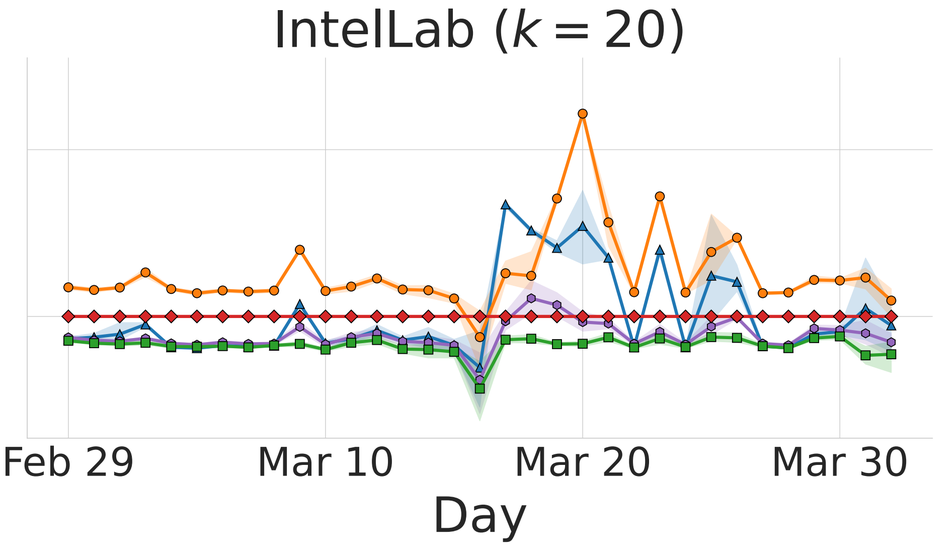
$$m = 50 \cdot k$$

Additional Experiments: Estimated Distortion

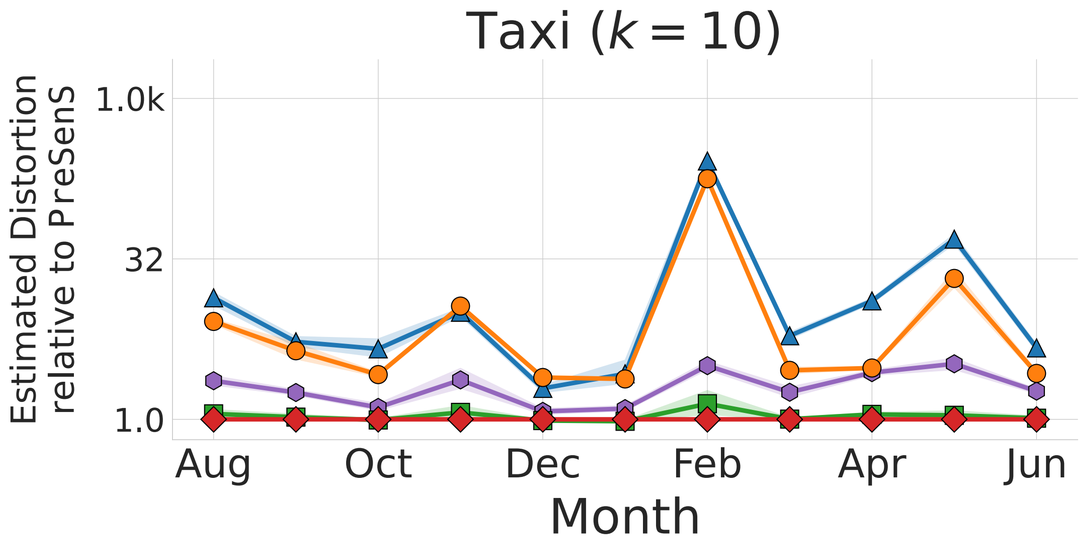


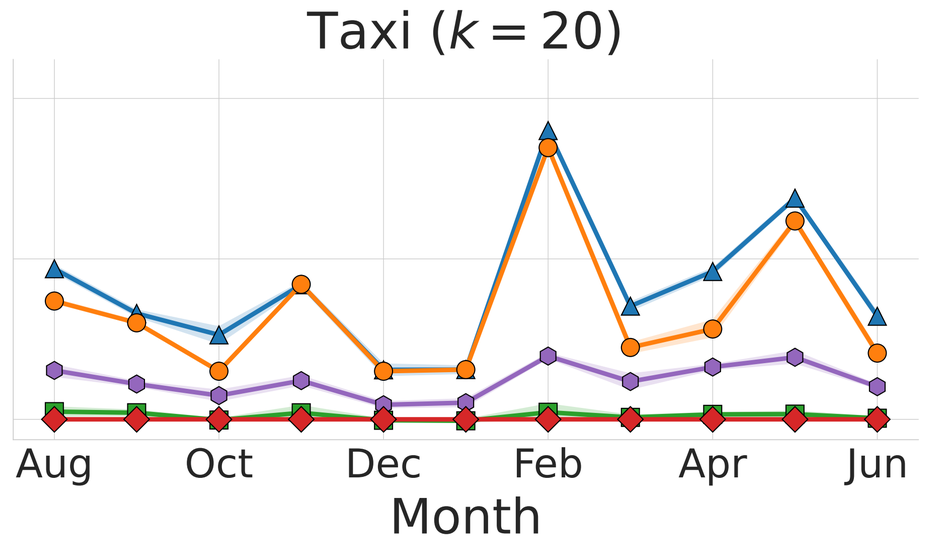


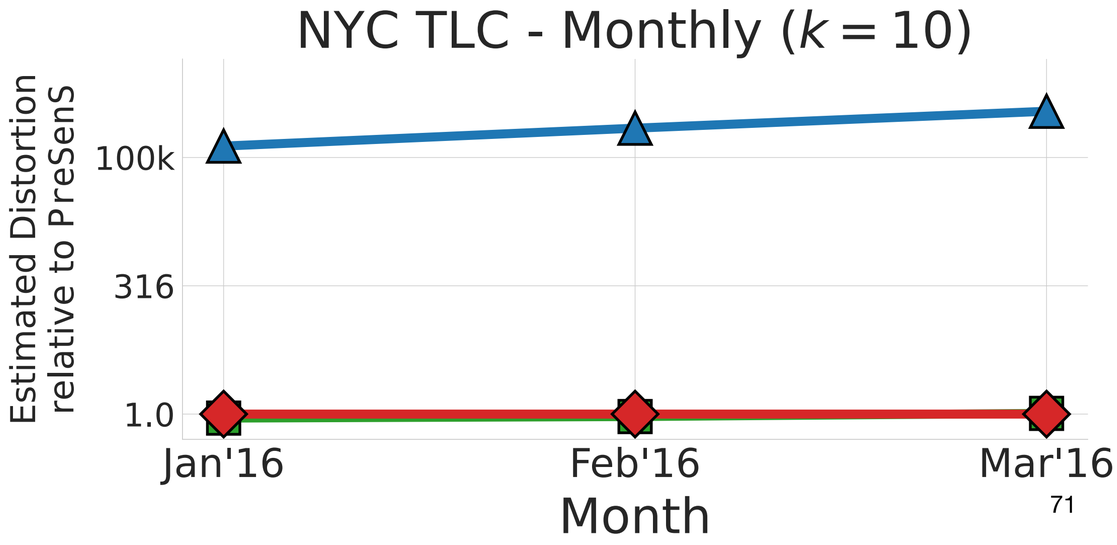


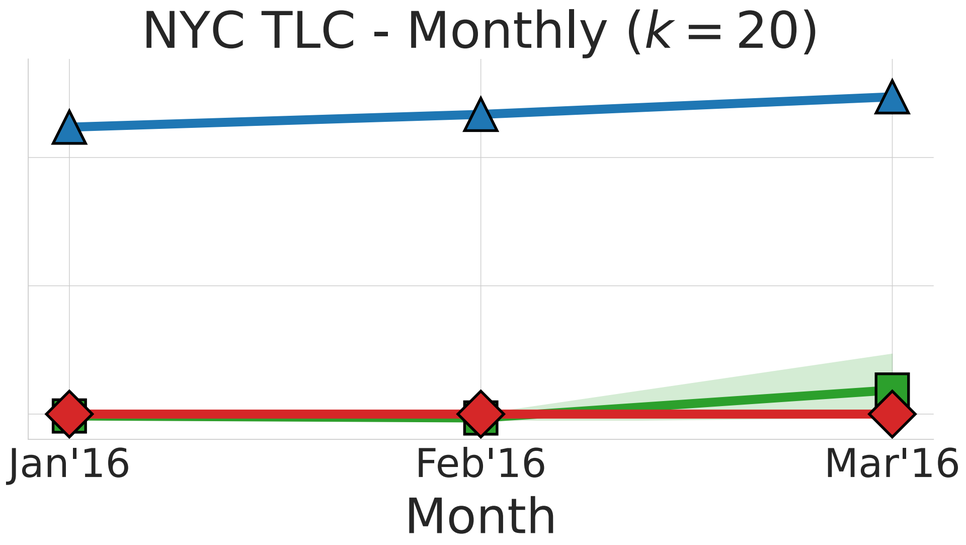


Additional Experiments: Estimated Distortion









Additional Experiments: Size of Coresets

Number of **unique** points in *PreSensS*'s coresets over Number of **unique** points in *Uniform-Sampling*'s coresets.

Dataset	$m = 50k$		
	$k = 10$	$k = 20$	$k = 50$
Twitter	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
IntelLab	0.97 ± 0.04	0.98 ± 0.03	0.99 ± 0.02
Taxi	0.64 ± 0.16	0.58 ± 0.14	0.53 ± 0.14
NYT (M)	0.22 ± 0.14	0.38 ± 0.19	0.53 ± 0.20
NYT (Y)	0.55 ± 0.18	0.52 ± 0.24	0.57 ± 0.25