Coresets for Nonparametric Estimation –
the Case of DP-Means

Olivier Bachem (ETH Zurich)
joint work with Mario Lucic and Andreas Krause
Introduction

Bayesian nonparametric models

• Infer model complexity from data
• Model complexity grows with increasing amounts of data

But scaling nonparametric estimation is notoriously hard

Our contribution: Novel approach using coresets
What are coresets?

- **Idea**: Approximate the full dataset by a smaller weighted subset

- **Coreset**: weighted subset with strong theoretical guarantees on approximation quality

- **Previous work**: coreset constructions for parametric models such as K-Means, K-Median and Gaussian mixture models

- ✔ Fast inference by solving on (small) coreset
Why coresets?

Example data

Large cluster
Small cluster

Uniform subsample

Bad approximation:
Only points from large cluster

Coreset (non-uniform)

Good approximation:
Points from both clusters
Non-uniform weights
DP-Means clustering as a prototype

“K-Means with flexible number of centers”

- Minimize the DP-Means objective function:

\[ \text{cost}_{DP}(\mathcal{P}, Q) = \sum_{p \in \mathcal{P}} \min_{q \in Q} \text{dist}(p, q)^2 + |Q|\lambda \]

- Solution \( Q \) can consist of unbounded number of cluster centers
- Natural motivations ("limit" of DPMM, Elbow method, AIC/BIC)

Ideal prototype to apply coresets to nonparametric estimation
Coresets for DP-Means

Coreset definition for DP-Means (simplified)

• $\mathcal{C}$ is a $\epsilon$-coreset of $\mathcal{P}$ if for all solutions $Q$:

$$|\text{cost}_{DP}(\mathcal{P}, Q) - \text{cost}_{DP}(\mathcal{C}, Q)| \leq \epsilon \text{cost}_{DP}(\mathcal{P}, Q)$$

✔ Guarantees $(1 \pm \epsilon)$ multiplicative approximation of objective function
Solving on the coreset

- Coreset property needs to hold **uniformly** for all solutions $Q$

**Theoretical guarantee**

\[
\text{cost}_{DP}(\mathcal{P}, Q(C)) \leq \frac{1 + \epsilon}{1 - \epsilon} \text{cost}_{DP}(\mathcal{P}, Q(\mathcal{P}))
\]

Optimal solution on coreset  Optimal solution on full dataset

✓ Solving on the coreset leads to provably good solutions
## Why is it hard to extend to NP?

1. Coreset property needs to hold uniformly for all solutions $Q$

| ✓ Parametric (K-Means): cardinality of solutions is **bounded** |
| ! Nonparametric (DP-Means): cardinality of solutions is **unbounded** |

2. Coreset constructions (generally) require rough approximations

| ✓ Parametric (K-Means): use solutions of larger complexity |
| ! Nonparametric (DP-Means): we do not even know size of solution! |
Practical coreset construction: step 1

1. Obtain rough approximation: DP-Means++ algorithm

- **D²-sampling**: iteratively sample data points as new centers proportional to squared distance to existing cluster centers
- **K-Means++**: sample $k$ centers
- **DP-Means++**: sample centers while

$$
\sum_{p \in \mathcal{P}} \text{dist}(p, C)^2 > 16\lambda |C|(\log_2 |C| + 2)
$$

✔  DP-Means++ yields provably competitive solutions
Example

Dataset

Rough solution using DP-Means++
Practical coreset construction: step 2

2 Coreset using importance sampling

- **Idea**: sample points with high impact on objective more frequently

- **Sampling distribution** (see paper for details):

  \[
  q(p) \propto \frac{2\alpha \text{dist}(p, A)^2}{\bar{c}} + \frac{4\alpha \sum_{p' \in P_a} \text{dist}(p', A)^2}{|P_a|\bar{c}} + \frac{4|P|}{|P_a|} + 1
  \]

- **Coreset size**: sample \(\mathcal{O}\left(\frac{dk'^3 \log k'}{\epsilon^2}\right)\) points to obtain a coreset

- **Coreset is sublinear in the number of data points**
Example (continued)

Sampling probabilities

Coreset
Example (continued)

Solution and coreset

Solution and full dataset
Advantages of coresets in practice

- Simple, practical algorithms despite non-trivial analysis
- Flexibly choose coreset size and thus computational complexity
- Theoretical guarantees as opposed to uniform subsampling
Experiment 1: approximation quality of coresets

- **Goal**: Check how well coresets approximate the objective function
- **How?**: Evaluate random solutions on coresets of different sizes and compare to true values on full data set
- **Baseline**: Compare to uniform subsampling
Experiment 1: approximation quality of coresets

![Graph showing the approximation quality of coresets for different subsample sizes and variance values. The graph compares KDD, Uniform, and Coreset methods.](image)
Experiment 1: approximation quality of coresets

Coresets provide better approximation of objective function
Experiment 2: solving on the coreset

- **Goal:** solution quality when solving on coreset
Coresets leads to better solutions than uniform subsampling
Conclusion

- First coreset construction for nonparametric estimation
- Theoretical existence results for DP-Means
- Practical algorithm with promising experimental results
- First step towards coresets for Bayesian nonparametrics
Theoretical existence result

Theorem 3.2

Let \( \epsilon > 0 \), and let \( \mathcal{P} \) be a dataset. Then there exist \( \epsilon \)-coresets of \( \mathcal{P} \) with size

\[
O \left( \frac{d^d k^* \log n}{\epsilon^d} \right)
\]

where \( k^* \) is the optimal number of cluster centers.

There exists coresets for solution sets of unbounded size

Exponential grid and knowledge of optimal solution required
Appendix: speedup vs. full dataset

### KDD with subsample size 5000

<table>
<thead>
<tr>
<th></th>
<th>Uniform</th>
<th>Coreset</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sampling time (s)</td>
<td>0.01</td>
<td>0.49</td>
<td>-</td>
</tr>
<tr>
<td>Solver time (s)</td>
<td>13.33</td>
<td>13.49</td>
<td>635.07</td>
</tr>
<tr>
<td>Total time (s)</td>
<td>13.33</td>
<td>13.98</td>
<td>635.07</td>
</tr>
<tr>
<td>Speedup</td>
<td>47.6x</td>
<td>45.4x</td>
<td>1.0x</td>
</tr>
<tr>
<td>DP-Means cost (10^9)</td>
<td>299.54</td>
<td>250.36</td>
<td>244.57</td>
</tr>
<tr>
<td>Relative error η</td>
<td>22.5%</td>
<td>2.4%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Significant speedup of up to 45x at relative error of 2.4%
Appendix: speedup vs. uniform

Speedup of up to an order of magnitude compared to uniform