

# From Large to Small Datasets: Size Generalization for Clustering Algorithm Selection



**Vaggos Chatziafratis**  
UC Santa Cruz



**Ishani Karmarkar**  
Stanford



**Ellen Vitercik**  
Stanford

# How to integrate **machine learning** into **discrete optimization**?



## **Algorithm configuration**

*How to tune an algorithm's parameters?*



## **Algorithm selection**

*Given a variety of algorithms, which to use?*



## **Algorithm design**

*Can machine learning guide algorithm discovery?*

# How to integrate **machine learning** into **discrete optimization**?



## **Algorithm configuration**

*How to tune an algorithm's parameters?*



## **Algorithm selection**

*Given a variety of algorithms, which to use?*



## **Algorithm design**

*Can machine learning guide algorithm discovery?*

# Algorithm configuration

## Example: **Integer programming solvers**

Most popular tool for solving combinatorial (& nonconvex) problems



Routing



Manufacturing



Scheduling



Planning



Finance

# Algorithm configuration

IP solvers (CPLEX, Gurobi) have a **ton** parameters

- CPLEX has **170-page** manual describing **172** parameters
- Tuning by hand is notoriously **slow, tedious,** and **error-prone**

CPX_PARAM_NODEFILEIND 100	CPX_PARAM_TRELIM 160	CPX_PARAM_RANDOMSEED 130	CPXPARAM_MIP_Pool_RelGap 148	CPX_PARAM_FLOWCOVERS 70	CPX_PARAM_BRDIR 39
CPX_PARAM_NODELIM 101	CPX_PARAM_TUNINGDETLIM 160	CPX_PARAM_REDUCE 131	CPXPARAM_MIP_Pool_Replace 151	CPX_PARAM_FLOWPATHS 71	CPX_PARAM_BTTOL 40
CPX_PARAM_NODESEL 102	CPX_PARAM_TUNINGDISPLAY 162	CPX_PARAM_REINV 131	CPXPARAM_MIP_Strategy_Branch 39	CPX_PARAM_FPHEUR 72	CPX_PARAM_CALCQCPCDUALS 41
CPX_PARAM_NUMERICALEMPHASIS 102	CPX_PARAM_TUNINGMEASURE 163	CPX_PARAM_RELAXPREIND 132	CPXPARAM_MIP_Strategy_MIQCPStrat 93	CPX_PARAM_FRACCAND 73	CPX_PARAM_CLIQUES 42
CPX_PARAM_NZREADLIM 103	CPX_PARAM_TUNINGREPEAT 164	CPX_PARAM_RELOBJDIF 133	CPXPARAM_MIP_Strategy_StartAlgorithm 139	CPX_PARAM_FRACCUTS 73	CPX_PARAM_CLOCKTYPE 43
CPX_PARAM_OBJDIF 104	CPX_PARAM_TUNINGTILIM 165	CPX_PARAM_REPAIRTRIES 133	CPXPARAM_MIP_Strategy_VariableSelect 166	CPX_PARAM_FRACPASS 74	CPX_PARAM_CLONELOG 43
CPX_PARAM_OBJLLIM 105	CPX_PARAM_VARSEL 166	CPX_PARAM_REPEATPRESOLVE 134	CPXPARAM_MIP_SubMIP_NodeLimit 155	CPX_PARAM_GUBCOVERS 75	CPX_PARAM_COEREDIND 44
CPX_PARAM_OBJULIM 105	CPX_PARAM_WORKDIR 167	CPX_PARAM_RINSHEUR 135	CPXPARAM_OptimalityTarget 106	CPX_PARAM_HEURFREQ 76	CPX_PARAM_COLREADLIM 45
CPX_PARAM_PARALLELMODE 108	CPX_PARAM_WORKMEM 168	CPX_PARAM_RLT 136	CPXPARAM_Output_WriteLevel 169	CPX_PARAM_IMPLBD 76	CPX_PARAM_CONFLICTDISPLAY 46
CPX_PARAM_PERIND 110	CPX_PARAM_WRITELEVEL 169	CPX_PARAM_ROWREADLIM 141	CPXPARAM_Preprocessing_Aggregator 19	CPX_PARAM_INTSOLFILEPREFIX 78	CPX_PARAM_COVERS 47
CPX_PARAM_PERLIM 111	CPX_PARAM_ZEROHALFCUTS 170	CPX_PARAM_SCAIND 142	CPXPARAM_Preprocessing_Fill 19	CPX_PARAM_INTSOLLIM 79	CPX_PARAM_CPUMASK 48
CPX_PARAM_POLISHAFTERDETTIME 111	CPXPARAM_Benders_Strategy 30	CPX_PARAM_SCRIND 143	CPXPARAM_Preprocessing_Linear 120	CPX_PARAM_ITLIM 80	CPX_PARAM_CRAIN 50
CPX_PARAM_POLISHAFTEREPAGAP 112	CPXPARAM_Benders_Tolerances_feasibilitycut 35	CPX_PARAM_SIFTALG 143	CPXPARAM_Preprocessing_Reduce 131	CPX_PARAM_LANDPCUTS 82	CPX_PARAM_CUTLO 51
CPX_PARAM_POLISHAFTEREPGAP 113	CPXPARAM_Benders_Tolerances_optimalitycut 36	CPX_PARAM_SIFTDISPLAY 144	CPXPARAM_Preprocessing_Symmetry 156	CPX_PARAM_LBHEUR 81	CPX_PARAM_CUTPASS 52
CPX_PARAM_POLISHAFTERINTSOL 114	CPXPARAM_Conflict_Algorithm 46	CPX_PARAM_SIFTITLIM 145	CPXPARAM_Read_DataCheck 54	CPX_PARAM_LPMETHOD 136	CPX_PARAM_CUTSFACTOR 52
CPX_PARAM_POLISHAFTERNODE 115	CPXPARAM_CPUmask 48	CPX_PARAM_SIMDISPLAY 145	CPXPARAM_Read_Scale 142	CPX_PARAM_MCF CUTS 82	CPX_PARAM_CUTUP 53
CPX_PARAM_POLISHAFTERTIME 116	CPXPARAM_DistMIP_Rampup_Duration 128	CPX_PARAM_SINGLIM 146	CPXPARAM_ScreenOutput 143	CPX_PARAM_MEMORYEMPHASIS 83	CPXPARAM_DATACHECK 54
CPX_PARAM_POLISHTIME (deprecated) 116	CPXPARAM_LPMethod 136	CPX_PARAM_SOLNPOOLGAP 146	CPXPARAM_Sifting_Algorithm 143	CPX_PARAM_MIPCBREDLP 84	CPX_PARAM_DEPIND 55
CPX_PARAM_POPULATELIM 117	CPXPARAM_MIP_Cuts_BQP 38	CPX_PARAM_SOLNPOOLCAPACITY 147	CPXPARAM_Sifting_Display 144	CPX_PARAM_MIPDISPLAY 85	CPX_PARAM_DETLIM 56
CPX_PARAM_PPRIND 118	CPXPARAM_MIP_Cuts_LocallyImplied 77	CPX_PARAM_SOLNPOOLREPLACE 151	CPXPARAM_Sifting_Iterations 145	CPX_PARAM_MIPEMPHASIS 87	CPX_PARAM_DISJ CUTS 57
CPX_PARAM_PREDUAL 119	CPXPARAM_MIP_Cuts_RLT 136	CPX_PARAM_SOLNPOOLINTENSITY 149	CPXPARAM_Simplex_Display 145	CPX_PARAM_MIPINTERVAL 88	CPX_PARAM_DIVETYPE 58
CPX_PARAM_PREIND 120	CPXPARAM_MIP_Cuts_ZeroHalfCut 170	CPX_PARAM_SOLNPOOLREPLACE 151	CPXPARAM_Simplex_Limits_Singularity 146	CPX_PARAM_MIPKAPPASTATS 89	CPX_PARAM_DPRIND 59
CPX_PARAM_PRLINEAR 120	CPXPARAM_MIP_Limits_CutsFactor 52	CPX_PARAM_SOLUTIONTARGET (deprecated: see CPXPARAM_OptimalityTarget 106)	CPXPARAM_SolutionType 152	CPX_PARAM_MIPORDIND 90	CPX_PARAM_EACHCUTLIM 60
CPX_PARAM_PREPASS 121	CPXPARAM_MIP_Limits_RampupDetTimeLimit 127	CPXPARAM_SOLUTIONTYPE 152	CPXPARAM_Threads 157	CPX_PARAM_MIPORDTYPE 91	CPX_PARAM_EPAGAP 61
CPX_PARAM_PRESLVND 122	CPXPARAM_MIP_Limits_RampupTimeLimit 128	CPX_PARAM_STARTALG 139	CPXPARAM_TimeLimit 159	CPX_PARAM_MIPSEARCH 92	CPX_PARAM_EPGAP 61
CPX_PARAM_PRICELIM 123	CPXPARAM_MIP_Limits_Solutions 79	CPX_PARAM_STRONGCANDLIM 154	CPXPARAM_Tune_DefTimeLimit 160	CPX_PARAM_MIQCPSTRAT 93	CPX_PARAM_EPINT 62
CPX_PARAM_PROBE 123	CPXPARAM_MIP_Limits_StrongCand 154	CPX_PARAM_STRONGCANDLIM 154	CPXPARAM_Tune_Display 162	CPX_PARAM_MIRCUTS 94	CPX_PARAM_EPMRK 64
CPX_PARAM_PROBEDETTIME 124	CPXPARAM_MIP_Limits_StrongIt 154	CPX_PARAM_STRONGITLIM 154	CPXPARAM_Tune_Measure 163	CPX_PARAM_MPSLONGNUM 94	CPX_PARAM_EPOPT 65
CPX_PARAM_PROBETIME 124	CPXPARAM_MIP_Limits_TreeMemory 160	CPX_PARAM_SUBALG 99	CPXPARAM_Tune_Repeat 164	CPX_PARAM_NETDISPLAY 95	CPX_PARAM_EPPER 65
CPX_PARAM_QPMAKEPSDIND 125	CPXPARAM_MIP_OrderType 91	CPX_PARAM_SUBMIPNODELIMIT 155	CPXPARAM_Tune_TimeLimit 165	CPX_PARAM_NETEPOPT 96	CPX_PARAM_EPRELAX 66
CPX_PARAM_QPMETHOD 138	CPXPARAM_MIP_Pool_AbsGap 146	CPX_PARAM_SYMMETRY 156	CPXPARAM_WorkDir 167	CPX_PARAM_NETEPRHS 96	CPX_PARAM_EPRHS 67
CPX_PARAM_QPNZREADLIM 126	CPXPARAM_MIP_Pool_Capacity 147	CPX_PARAM_THREADS 157	CPXPARAM_WorkMem 168	CPX_PARAM_NETFIND 97	CPX_PARAM_FEASOPTMODE 68
	CPXPARAM_MIP_Pool_Intensity 149	CPX_PARAM_TILIM 159	CraInd 50	CPX_PARAM_NETITLIM 98	CPX_PARAM_FILEENCODING 69
				CPX_PARAM_NETPPRIND 98	

# Algorithm configuration

IP solvers (CPLEX, Gurobi) have a **ton** parameters

- CPLEX has **170-page** manual describing **172** parameters
- Tuning by hand is notoriously **slow, tedious**, and **error-prone**

What's the best **configuration** for the application at hand?



Best configuration for **routing** problems  
likely not suited for **scheduling**



# How to integrate **machine learning** into **discrete optimization**?

- **Algorithm configuration**  
*How to tune an algorithm's parameters?*
- **Algorithm selection**  
*Given a variety of algorithms, which to use?*
- **Algorithm design**  
*Can machine learning guide algorithm discovery?*

# Example: Clustering

Many different algorithms

K-means



Mean shift



Ward



Agglomerative



Birch



How to **select** the best algorithm for the application at hand?

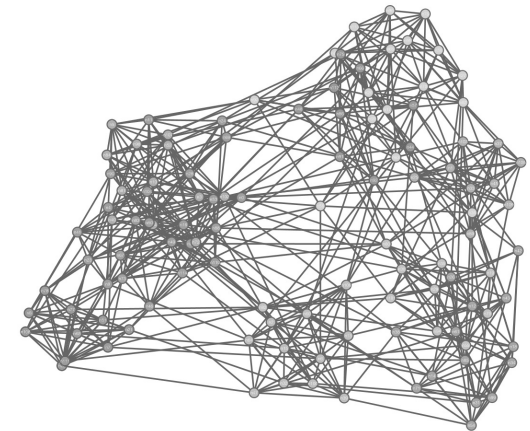


# Algorithm selection in theory

**Worst-case analysis** has been the main framework for decades  
*Has led to beautiful, practical algorithms*

Worst-case analysis's approach to **algorithm selection**:  
Select the algorithm that's best in worst-case scenarios

Worst-case instances **rarely occur in practice**



# How to integrate **machine learning** into **discrete optimization**?

Answer to this question is built on a key observation:

**In practice, we have data about  
the application domain**

A stack of several cardboard boxes is shown, with a person's hands visible at the bottom right, holding one of the boxes. The boxes are brown and have red and white striped string wrapped around them. Each box has a 'FRAGILE' label with three icons: a vertical line with an upward arrow, a wine glass, and an umbrella. The background is blurred, showing what appears to be a warehouse or shipping area with other boxes and equipment.

**In practice, we have data about  
the application domain**

Routing problems a shipping company solves

**In practice, we have data about  
the application domain**



Clustering problems a biology lab solves



**In practice, we have data about  
the application domain**



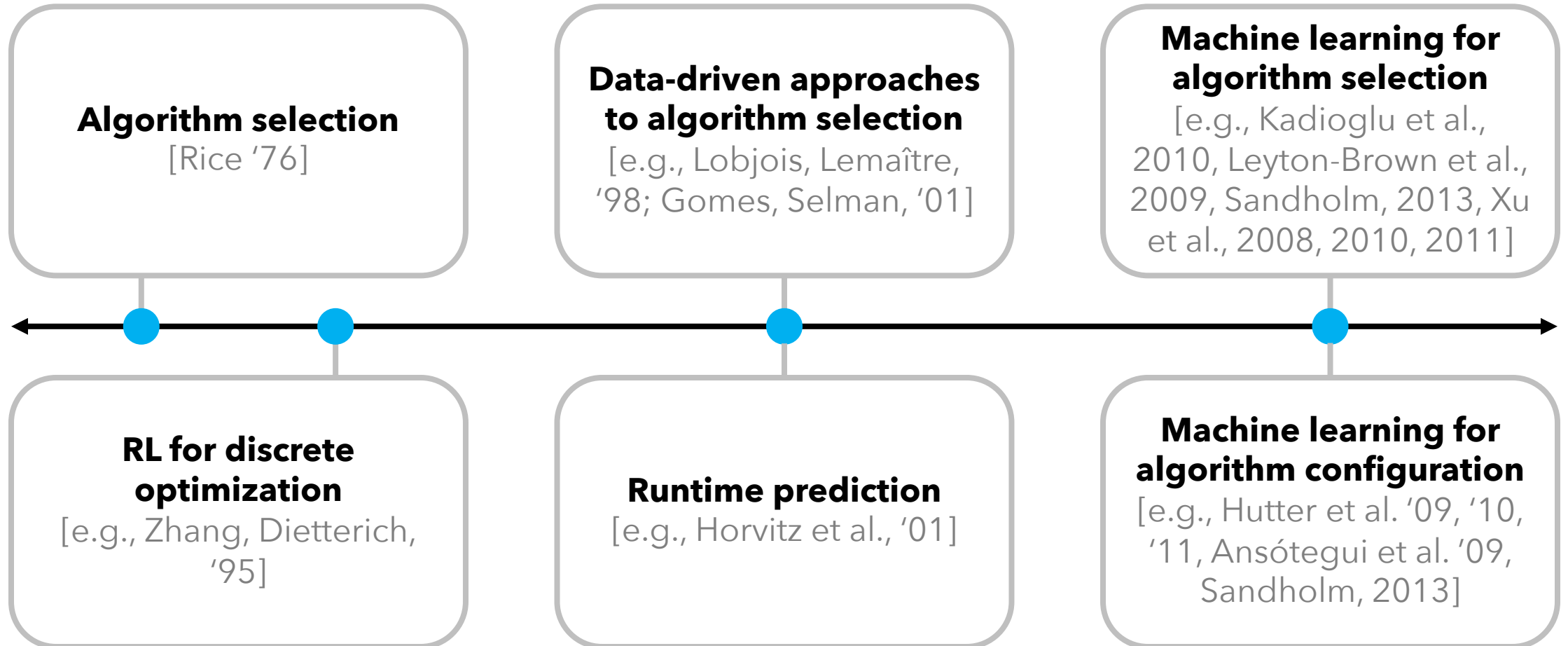
Scheduling problems an airline solves

# In practice, we have data about the application domain

How can we use this data to guide:

- **Algorithm configuration**  
*How to tune an algorithm's parameters?*
- **Algorithm selection**  
*Given a variety of algorithms, which to use?*
- **Algorithm design**  
*Can machine learning guide algorithm discovery?*

# A bit of history



# A bit of history



## **Late 2010s-present:**

- Tons of work integrating modern ML models into discrete optimization  
[e.g., surveys by Bengio et al., '18; Cappart et al., '23; ...]
- Theoretical guarantees  
[e.g., book chapters by Balcan, '20; Mitzenmacher, Vassilvitskii, '20; ...]



# Conventional data-driven pipeline

1. Gather **historical** problem instances
2. Identify the algorithm (and configuration) with the **best average performance**
3. Hope (or prove) it will have (nearly) best **future** performance

## **Key scalability challenge:**

Evaluating an alg's performance on a combinatorial problem...  
Typically requires **solving** that combinatorial problem!

# Size generalization in practice

Applied research circumvents this challenge:

Use a **distribution** to generate small & large problems

*E.g., Erdős-Rényi graphs*



**Small** instances are used for **training**



**Large** instances are used for **testing**

E.g.:

Dai, Khalil, et al., NeurIPS'17

Li et al., NeurIPS'18

Gasse et al., NeurIPS'19

Veličković et al., ICLR'20

Veličković et al., ICML'20

Tang et al., ICML'20

Gupta et al., NeurIPS'20

Ibarz et al., LoG'22

Chmiela et al., NeurIPS'21

Huang et al., ICML'23

Alomrani et al., TMLR'23

...

# Size generalization in practice

Applied research circumvents this challenge:

Use a **distribution** to generate small & large problems

*E.g., Erdős-Rényi graphs*



**Small** instances are used for **training**



**Large** instances are used for **testing**

However:

- Practical problems don't have a **known** distribution
- Practitioners simply have massive problems they must solve

# Size generalization: algorithm selection

Given a massive combinatorial problem, can we:

1. "**Shrink**" it
2. Evaluate **candidate algorithms** on the smaller instance
3. Provably guarantee:  
The best algorithm on the **small** instance  
...is also best on the original **large** instance?

# Outline

1. Introduction
  - a. Size generalization motivation
  - b. Clustering algorithm selection**
2. Center-based clustering
3. Single-linkage
4. Conclusions and future directions

# Semi-supervised clustering

Assume there's a **ground truth** clustering of a massive dataset

- Accessible through **expensive** ground-truth oracle queries
- Models interactions with a **domain expert**
- Basu et al., KDD'04; Zhu, '05; Kulis, ICML'05; Chen, Feng, Neurocomputing '12; Balcan, Nagarajan, White, **V**, COLT'17; ...

## Applications:

- Image recognition [Boom et al., ICPR'12]
- Medical diagnostics [Ershadi, Seifi, Applied Soft Computing, '22]



# Clustering algorithm selection

Given a set of **candidate algorithms**:

select algorithm that will best recover the ground truth using

- Low runtime
- Few ground-truth queries

In practice, clustering algorithm selection is often done *"in a very **ad hoc**, if not completely random, manner,"* which is regrettable *"given the **crucial effect** of [algorithm selection] on the resulting clustering."*

[Ben-David, AAAI'18]

# Notation

$\mathcal{G} = \{G_1, \dots, G_k\}$  is the ground truth clustering of  $\mathcal{X} \subset \mathbb{R}^d$

**Ground-truth oracle**  $\tau: \tau(x) = i$  if  $x \in G_i$

$\mathcal{C} = \{C_1, \dots, C_k\}$  is a clustering of  $\mathcal{X}' \subseteq \mathcal{X}$

$$\text{cost}_{\mathcal{G}}(\mathcal{C}; \mathcal{X}') = \frac{1}{|\mathcal{X}'|} \min_{\sigma \in \Sigma^k} \sum_{x \in \mathcal{X}'} \sum_{j=1}^k \mathbf{1}\{x \in C_{\sigma(j)} \text{ and } x \notin G_j\}$$

[e.g., Ashtiani, Ben-David, UAI'15]

**Distance oracle** returns  $d(x, y)$  for  $x, y \in \mathcal{X}$




# Size generalization for clustering

Given a huge clustering dataset  $\mathcal{X}$ , can we:

1. Subsample  $\mathcal{X}$  (uniformly at random)
2. Evaluate a set of candidate algorithms on the subsample
3. Prove the algorithm with lowest cost on the subsample  
...will have low cost on  $\mathcal{X}$ ?

**Goal:** minimize


- 
1. Runtime
  2. Number of ground-truth oracle queries
  3. Number of distance oracle queries

# Size generalization for clustering

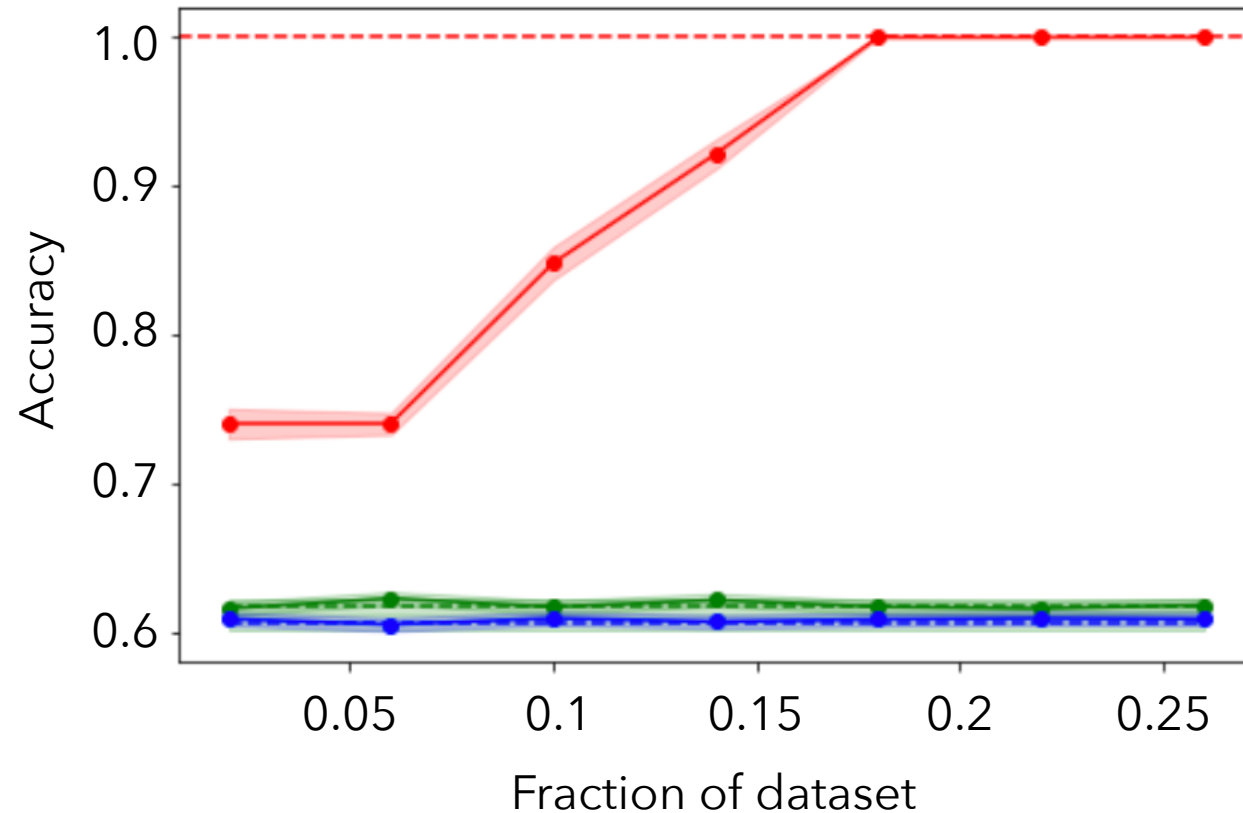
Given a huge clustering dataset  $\mathcal{X}$ , can we:

1. Subsample  $\mathcal{X}$  (uniformly at random)
2. Evaluate a set of candidate algorithms on the subsample
3. Prove the algorithm with lowest cost on the subsample  
...will have low cost on  $\mathcal{X}$ ?

Answer this question in  
the affirmative for

- 
1. Gonzalez's k-centers heuristic\*
  2. k-means++
  3. Single-linkage clustering

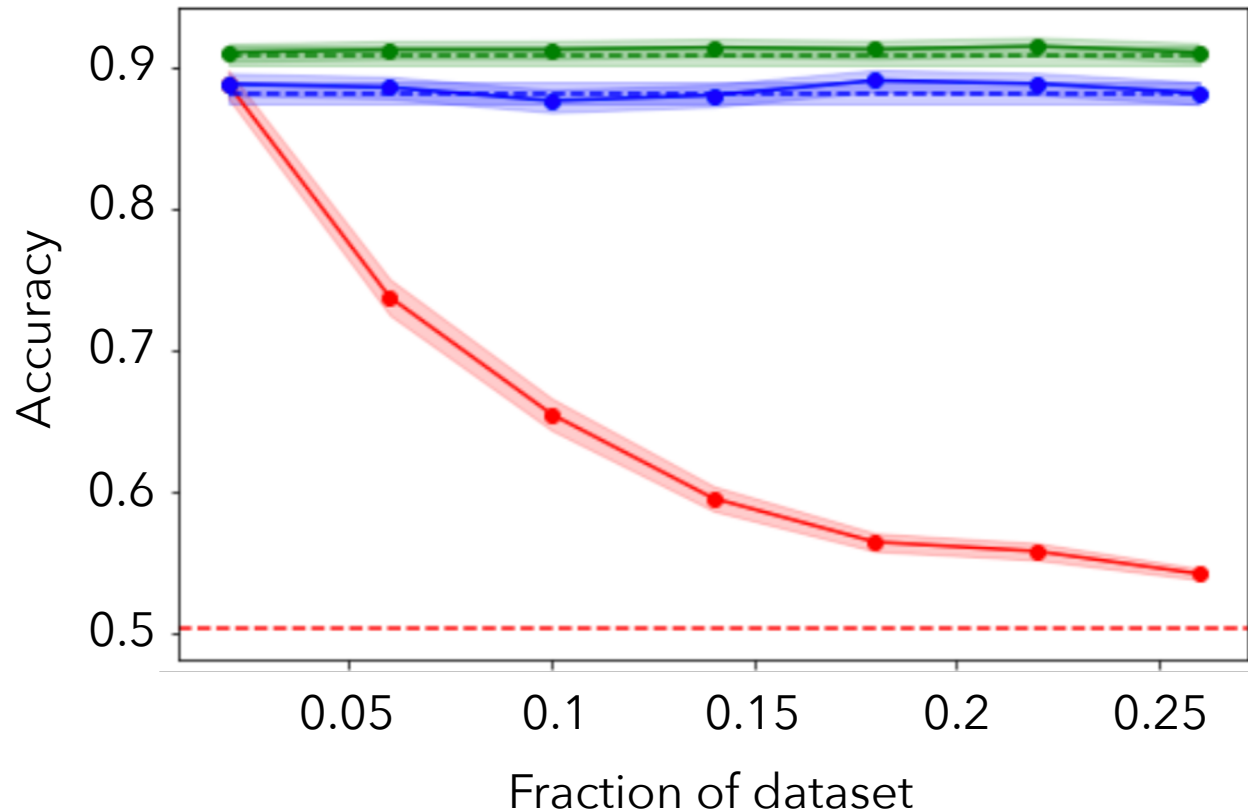
# Empirical motivation



- Single linkage
- Subsampled single linkage
- $k$ -means++
- Subsampled  $k$ -means++
- $k$ -centers heuristic
- Subsampled  $k$ -centers heuristic

- Noisy circles dataset  
[Pedregosa, et al., '11]
- 500 points in original instance

# Empirical motivation



- Single linkage
- Subsampled single linkage
- $k$ -means++
- Subsampled  $k$ -means++
- $k$ -centers heuristic
- Subsampled  $k$ -centers heuristic

- Gaussian mixtures
- 500 points in original instance

# Related research

The clustering theory literature:

- Often implicitly assumes ground truth minimizes some  $h$ 
  - E.g.,  $k$ -means or -centers
- Many algorithms to (approximately) minimize  $h$ 
  - E.g., algorithms based on **coresets**:

Subsets  $\mathcal{X}_c \subseteq \mathcal{X}$  such that for any set of  $k$  centers  $\mathcal{C}$ ,  $h(\mathcal{C}, \mathcal{X}) \approx h(\mathcal{C}, \mathcal{X}_c)$

E.g.,  $k$ -means objective given  
**dataset**  $\mathcal{X}$  and centers  $\mathcal{C}$

E.g.,  $k$ -means objective given  
**coreset**  $\mathcal{X}_c$  and centers  $\mathcal{C}$

# Related research

The clustering theory literature:

- Often implicitly assumes ground truth minimizes some  $h$ 
  - E.g.,  $k$ -means or -centers
- Many algorithms to (approximately) minimize  $h$ 
  - E.g., algorithms based on **coresets**:

Subsets  $\mathcal{X}_c \subseteq \mathcal{X}$  such that for any set of  $k$  centers  $\mathcal{C}$ ,  $h(\mathcal{C}, \mathcal{X}) \approx h(\mathcal{C}, \mathcal{X}_c)$

However:

- Identifying  $h$  may be as hard as identifying the ground truth
- Ground truth **need not align** with any previously studied  $h$

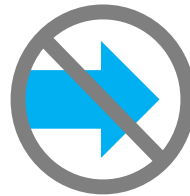
# Related research

The clustering theory literature:

- Often implicitly assumes ground truth minimizes some  $h$
- Many algorithms to (approximately) minimize  $h$ 
  - E.g., algorithms based on **coresets**:  
Subsets  $\mathcal{X}_c \subseteq \mathcal{X}$  such that for any set of  $k$  centers  $C$ ,  $h(C, \mathcal{X}) \approx h(C, \mathcal{X}_c)$

*Even if the ground truth is known to align with some  $h$ :*

Good approximation  
with respect to  $h$   
 $h(C, \mathcal{X}) \approx h(C, \mathcal{X}_c)$

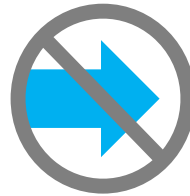


Low error with respect  
to the ground truth  
 $\text{cost}_g(C, \mathcal{X}) \approx \text{cost}_g(C, \mathcal{X}_c)$

# Related research

Even if the ground truth is known to align with some  $h$ :

Good approximation  
with respect to  $h$   
 $h(C, \mathcal{X}) \approx h(C, \mathcal{X}_c)$



Low error with respect  
to the ground truth  
 $\text{cost}_g(C, \mathcal{X}) \approx \text{cost}_g(C, \mathcal{X}_c)$

At the heart of an **important gap** between theory and practice

[Blum, '09; von Luxburg et al., '12; Balcan et al., JACM'13; Ben-David, AAAI'18; ...]

Also related: Ashtiani and Ben-David [UAI'15]

- Also study semi-supervised clustering
- Use samples to learn a data representation for downstream clustering



# Outline

1. Introduction
- 2. Center-based clustering**
3. Single-linkage
4. Conclusions and future directions

# Center-based clustering algorithms

Algorithms return centers  $\mathcal{C} = \{c_1, \dots, c_k\} \subset \mathbb{R}^d$

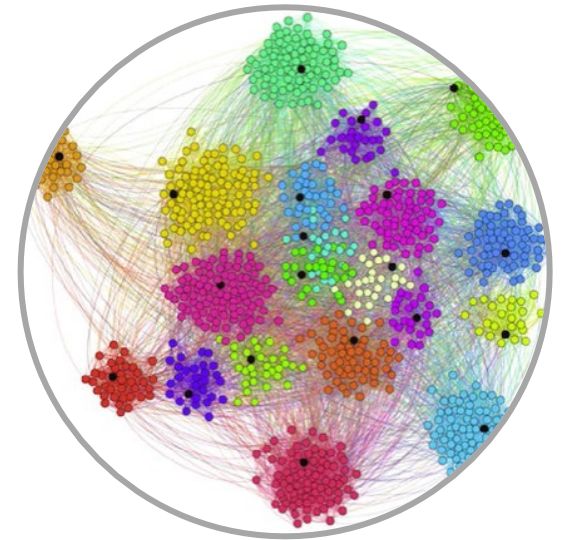
Assign points to nearest center:

$$S_i = \left\{ x \in \mathcal{X} : i = \operatorname{argmin}_{j \in [k]} d(x, c_j) \right\}$$

Notation:  $d_{\text{center}}(x, \mathcal{C}) = \min_{c \in \mathcal{C}} d(x, c)$

**k-means** objective: minimize  $\sum_{x \in \mathcal{X}} d_{\text{center}}(x, \mathcal{C})^2$

**k-centers** objective: minimize  $\max_{x \in \mathcal{X}} d_{\text{center}}(x, \mathcal{C})$



# Center seeding algorithm

## SEEDING

Choose  $c_1 \sim \text{Unif}(\mathcal{X})$ , set  $\mathcal{C}^1 = \{c_1\}$

For  $i \in \{2, \dots, k\}$ :

Sample  $c_i$  with probability  $\propto f(d_{\text{center}}(c_i, \mathcal{C}^{i-1}); \mathcal{X})$

Set  $\mathcal{C}^i = \mathcal{C}^{i-1} \cup \{c_i\}$

$k$ -means++ [Arthur, Vassilvitskii, SODA'07]:

$$f(d_{\text{center}}(c_i, \mathcal{C}^{i-1}); \mathcal{X}) = d_{\text{center}}(c_i, \mathcal{C}^{i-1})^2$$

log  $k$ -approximation algorithm for the  $k$ -means objective

*This 1-step version; additional Lloyd iterations can further improve objective*

# Center seeding algorithm

## SEEDING

Choose  $c_1 \sim \text{Unif}(\mathcal{X})$ , set  $\mathcal{C}^1 = \{c_1\}$

For  $i \in \{2, \dots, k\}$ :

Sample  $c_i$  with probability  $\propto f(d_{\text{center}}(c_i, \mathcal{C}^{i-1}); \mathcal{X})$

Set  $\mathcal{C}^i = \mathcal{C}^{i-1} \cup \{c_i\}$

Gonzalez's  $k$ -centers heuristic [TCS'85]:

$$f(d_{\text{center}}(c_i, \mathcal{C}^{i-1}); \mathcal{X}) = \mathbf{1} \left\{ c_i = \underset{x \in \mathcal{X}}{\operatorname{argmax}} d_{\text{center}}(x, \mathcal{C}^{i-1}) \right\}$$

Selects the point that's furthest from current centers  $\mathcal{C}^{i-1}$

# Center seeding algorithm

## SEEDING

Choose  $c_1 \sim \text{Unif}(\mathcal{X})$ , set  $\mathcal{C}^1 = \{c_1\}$

For  $i \in \{2, \dots, k\}$ :

Sample  $c_i$  with probability  $\propto f(d_{\text{center}}(c_i, \mathcal{C}^{i-1}); \mathcal{X})$

Set  $\mathcal{C}^i = \mathcal{C}^{i-1} \cup \{c_i\}$

Gonzalez's  $k$ -centers heuristic [TCS'85]:

$$f(d_{\text{center}}(c_i, \mathcal{C}^{i-1}); \mathcal{X}) = \mathbf{1} \left\{ c_i = \operatorname{argmax}_{x \in \mathcal{X}} d_{\text{center}}(x, \mathcal{C}^{i-1}) \right\}$$

2-approximation to the  $k$ -centers objective

# Center seeding algorithm: **APXSEEDING**

Sample  $\mathcal{X}' = \{x_1, \dots, x_{mk}\} \sim \text{Unif}(\mathcal{X})^{mk}$

Set  $\mathcal{C}^1 = \{x_1\}$  and  $\ell = 2$

//  $\ell$  is a counter for stepping through the sample  $\mathcal{X}'$

For  $i \in \{2, \dots, k\}$ :

Set  $x = x_\ell; \ell++$

//  $x$  is the candidate for the  $i^{\text{th}}$  center

For  $j \in \{2, \dots, m\}$ :

// Metropolis-Hastings procedure to update  $x$

Set  $y = x_\ell; \ell++$

If  $\frac{f(d_{\text{center}}(y, \mathcal{C}^{i-1}); \mathcal{X}')}{f(d_{\text{center}}(x, \mathcal{C}^{i-1}); \mathcal{X}')} > \text{Unif}([0,1])$ : set  $x = y$

Set  $\mathcal{C}^i = \mathcal{C}^{i-1} \cup \{x\}$

# Center seeding algorithm: **APXSEEDING**

Sample  $\mathcal{X}' = \{x_1, \dots, x_{mk}\} \sim \text{Unif}(\mathcal{X})^{mk}$

Set  $C^1 = \{x_1\}$  and  $\ell = 2$

For  $i \in \{2, \dots, k\}$ :

Set  $x = x_\ell; \ell++$

For  $j \in \{2, \dots, m\}$ :

Set  $y = x_\ell; \ell++$

If  $\frac{f(d_{\text{center}}(y, C^{i-1}); \mathcal{X}')}{f(d_{\text{center}}(x, C^{i-1}); \mathcal{X}')} > \text{Unif}([0, 1])$ : set  $x = y$

Set  $C^i = C^{i-1} \cup \{x\}$

- Requires only  $O(mk^2)$  distance oracle queries
- SEEDING requires  $O(|\mathcal{X}|k)$  queries

# Connection to prior research

APXSEEDING generalizes an approach by Bachem et al. [AAAI'16]

Use MCMC to obtain a sublinear-time  $k$ -means approximation

We generalize their framework to:

1. Work with **general functions  $f$**  (beyond  $k$ -means)
2. Give **accuracy** guarantees (instead of approximation)



# APXSEEDING guarantees

Guarantees depend on a parameter  $\zeta_{k,f}(\mathcal{X})$

- Measures the **smoothness** of SEEDING's distribution over centers
- As the distribution approaches uniform,  $\zeta_{k,f}(\mathcal{X}) \rightarrow 1$

$$\zeta_{k,f}(\mathcal{X}) = \max_{Q \subseteq \mathcal{X}, |Q| \leq k} \max_{x \in \mathcal{X}} \frac{|\mathcal{X}| f(d_{\text{center}}(x, Q); \mathcal{X})}{\sum_{y \in Q} f(d_{\text{center}}(y, Q); \mathcal{X})}$$

# APXSEEDING guarantees

## Theorem:

- Let  $S$  be the partition of  $\mathcal{X}$  induced by SEEDING
- Let  $S'$  be the partition of  $\mathcal{X}$  induced by APXSEEDING with  $O\left(\zeta_{k,f}(\mathcal{X}) \cdot k \cdot \log \frac{k}{\epsilon}\right)$  uniform samples
- For any ground truth clustering  $\mathcal{G}$ ,  
$$|\mathbb{E}[\text{cost}_{\mathcal{G}}(S; \mathcal{X})] - \mathbb{E}[\text{cost}_{\mathcal{G}}(S'; \mathcal{X})]| \leq \epsilon$$

- $\zeta_{k,f}(\mathcal{X}) \in [1, |\mathcal{X}|]$
- Smaller is better

# APXSEEDING guarantees

## Theorem:

- Let  $S$  be the partition of  $\mathcal{X}$  induced by SEEDING
- Let  $S'$  be the partition of  $\mathcal{X}$  induced by APXSEEDING with  $O\left(\zeta_{k,f}(\mathcal{X}) \cdot k \cdot \log \frac{k}{\epsilon}\right)$  uniform samples
- For any ground truth clustering  $\mathcal{G}$ ,  
$$|\mathbb{E}[\text{cost}_{\mathcal{G}}(S; \mathcal{X})] - \mathbb{E}[\text{cost}_{\mathcal{G}}(S'; \mathcal{X})]| \leq \epsilon$$

## Coming up:

Under natural assumptions,  $\zeta_{k,f}(\mathcal{X})$  is independent of  $|\mathcal{X}|$  for  $k$ -means++ and (a smoothed version) of Gonzalez's heuristic

# APXSEEDING guarantees

## Theorem:

- Let  $S$  be the partition of  $\mathcal{X}$  induced by SEEDING
- Let  $S'$  be the partition of  $\mathcal{X}$  induced by APXSEEDING with  $O\left(\zeta_{k,f}(\mathcal{X}) \cdot k \cdot \log \frac{k}{\epsilon}\right)$  uniform samples

- For any ground truth clustering  $\mathcal{G}$ ,  
$$\left| \mathbb{E}[\text{cost}_{\mathcal{G}}(S; \mathcal{X})] - \mathbb{E}[\text{cost}_{\mathcal{G}}(S'; \mathcal{X})] \right| \leq \epsilon$$

Can be estimated to error  $\epsilon$  using  $O\left(\frac{k}{\epsilon^2}\right)$  ground-truth queries

# Outline

1. Introduction
2. Center-based clustering
  - a. APXSEEDING
  - b.  $k$ -centers**
  - c.  $k$ -means
3. Single-linkage
4. Conclusions and future directions

# Gonzalez's k-centers heuristic

## First obstacle:

$$f(d_{\text{center}}(x, C); \mathcal{X}) = \mathbf{1} \left\{ x = \operatorname{argmax}_{y \in \mathcal{X}} d_{\text{center}}(y, C) \right\}$$

is deterministic, so  $\zeta_{k,f}(\mathcal{X}) = |\mathcal{X}|$

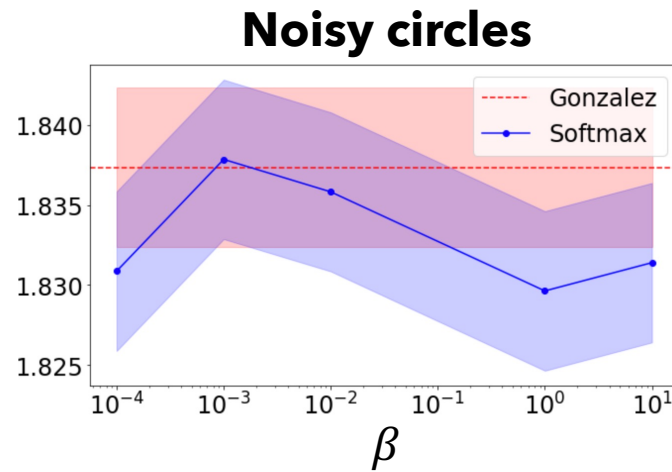
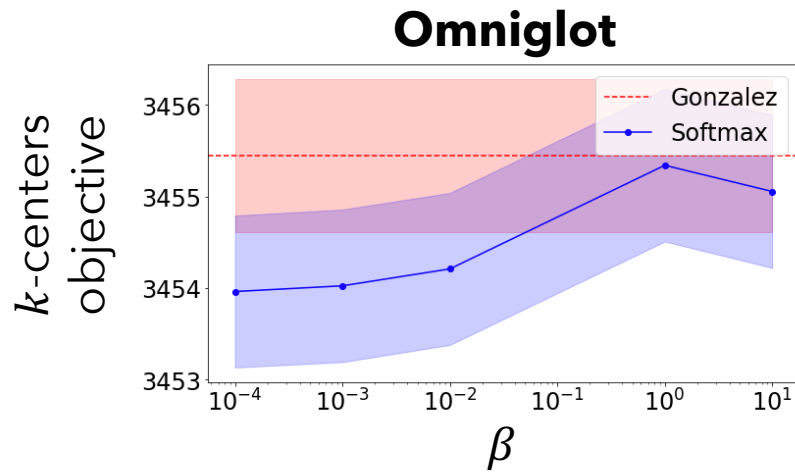
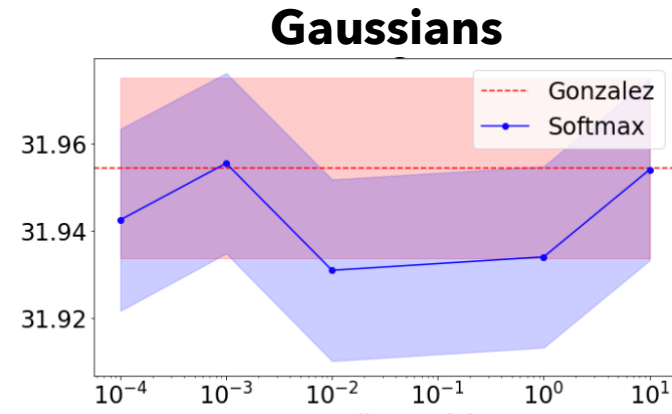
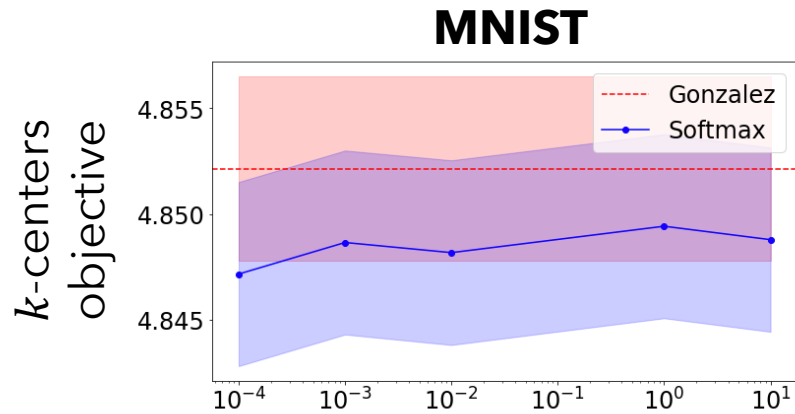
Instead, we'll study a **smoothed** version of the heuristic:

$$f_{\text{softmax}}(d_{\text{center}}(x, C); \mathcal{X}) = \exp(\beta d_{\text{center}}(x, C))$$

# Outline

1. Introduction
2. Center-based clustering
  - a. APXSEEDING
  - b.  $k$ -centers
    - a. **Justification of smoothed  $k$ -centers**
    - b. Size generalization for smoothed  $k$ -centers
  - c.  $k$ -means
3. Single-linkage
4. Conclusions and future directions

# Gonzalez versus softmax k-centers



García-Díaz et al. [J. of Heuristics, '17] also observe a smoother heuristic can yield better performance



# Softmax $k$ -centers approximation bound

## Theorem:

- $C_{\text{OPT}}$  = optimal  $k$ -centers solution
  - Induces partition  $S_1, \dots, S_k$  of  $\mathcal{X}$
  - Suppose partition is balanced:  $\mu_\ell |\mathcal{X}| \leq |S_i| \leq \mu_u |\mathcal{X}|$  for all  $i$
- $C$  = centers returned by Softmax  $k$ -centers
- With high probability,

$$\max_{x \in \mathcal{X}} d_{\text{center}}(x, C) \leq 4 \max_{x \in \mathcal{X}} d_{\text{center}}(x, C_{\text{OPT}}) + \frac{1}{\beta} \log \frac{k \mu_u}{\mu_\ell}$$

# Outline

1. Introduction
2. Center-based clustering
  - a. APXSEEDING
  - b.  $k$ -centers
    - a. Justification of smoothed  $k$ -centers
    - b. Size generalization for smoothed  $k$ -centers**
  - c.  $k$ -means
3. Single-linkage
4. Conclusions and future directions

# Sample complexity bound

**Lemma:** If  $\max_{x,y \in \mathcal{X}} d(x,y) \leq R$ , then  $\zeta_{k, f_{\text{softmax}}}(\mathcal{X}) \leq \exp(2\beta R)$

*Exist instances where this is tight*

**Connecting the dots:**  $O\left(\exp(\beta R) \cdot k \cdot \log \frac{k}{\epsilon}\right)$  samples sufficient to ensure  $|\mathbb{E}[\text{cost}_g(S; \mathcal{X})] - \mathbb{E}[\text{cost}_g(S'; \mathcal{X})]| \leq \epsilon$

Partition of  $\mathcal{X}$  induced by SEEDING

Partition of  $\mathcal{X}$  induced by APXSEEDING

# Sample complexity bound

**Lemma:** If  $\max_{x,y \in \mathcal{X}} d(x,y) \leq R$ , then  $\zeta_{k, f_{\text{softmax}}}(\mathcal{X}) \leq \exp(2\beta R)$

*Exist instances where this is tight*

**Connecting the dots:**  $O\left(\exp(\beta R) \cdot k \cdot \log \frac{k}{\epsilon}\right)$  samples sufficient to ensure  $|\mathbb{E}[\text{cost}_g(S; \mathcal{X})] - \mathbb{E}[\text{cost}_g(S'; \mathcal{X})]| \leq \epsilon$

$\beta \geq \frac{1}{\gamma} \log \frac{k\mu_u}{\mu_\ell}$  sufficient for  $k$ -centers  $(4, \gamma)$ -approximation  
 $\Rightarrow$  number of samples doesn't depend on  $|\mathcal{X}|$

# Sample complexity bound

**Lemma:** If  $\max_{x,y \in \mathcal{X}} d(x,y) \leq R$ , then  $\zeta_{k, f_{\text{softmax}}}(\mathcal{X}) \leq \exp(2\beta R)$

*Exist instances where this is tight*

**Connecting the dots:**  $O\left(\exp(\beta R) \cdot k \cdot \log \frac{k}{\epsilon}\right)$  samples sufficient to ensure  $|\mathbb{E}[\text{cost}_g(S; \mathcal{X})] - \mathbb{E}[\text{cost}_g(S'; \mathcal{X})]| \leq \epsilon$

Experiments indicate  $\beta$  can be set much smaller, e.g.,  $\beta = \frac{1}{R}$   
 $\Rightarrow$  number of samples is  $O\left(k \cdot \log \frac{k}{\epsilon}\right)$

# Outline

1. Introduction
2. Center-based clustering
  - a. APXSEEDING
  - b.  $k$ -centers
  - c.  $k$ -means**
3. Single-linkage
4. Conclusions and future directions

# $k$ -means summary

$$f(d_{\text{center}}(x, C); \mathcal{X}) = d_{\text{center}}(x, C)^2$$

Assume  $\mathcal{X}$  drawn from some distribution

- Support contained in ball of radius  $R$
- Distribution satisfies other mild non-degeneracy assumptions

Results by Bachem et al. [AAAI'16] imply (informally) that

$\zeta_{k,f}(\mathcal{X})$  grows **linearly** in  $R^2$  and  $k$

$\Rightarrow$  number of samples doesn't depend on  $|\mathcal{X}|$

# Outline

1. Introduction
2. Center-based clustering
- 3. Single-linkage**
4. Conclusions and future directions



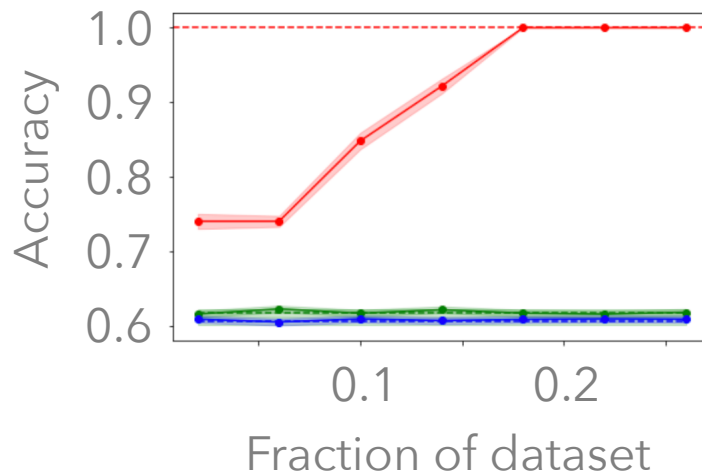
# Instability of single-linkage

Single linkage is known to be **unstable**

[Balcan et al., JMLR'14; Chaudhuri et al., IEEE Trans. Inf. Theory '14]

But in our experiments,

we find its accuracy **can be estimated on a subsample**



- Single linkage
- Subsampled single linkage
- *k*-means++
- Subsampled *k*-means++
- *k*-centers heuristic
- Subsampled *k*-centers heuristic

# Instability of single-linkage

Single linkage is known to be **unstable**

[Balcan et al., JMLR'14; Chaudhuri et al., IEEE Trans. Inf. Theory '14]

But in our experiments,

we find its accuracy **can be estimated on a subsample**

albeit, a larger sample

**Goal of this section** (a more philosophical goal 🤔):

We characterize which **property** of the dataset  $\mathcal{X}$  either:

- Allows for size generalization when this property holds, or
- Prohibits size generalization when it does not

# Single-linkage algorithm

Each point is initially in its own cluster:  $\mathcal{C}^0 = \{\{x_1\}, \dots, \{x_n\}\}; i = 0$

While  $|\mathcal{C}^i| > k$ :

$$d_i = \min_{A, B \in \mathcal{C}^i} \min_{x \in A, y \in B} d(x, y)$$

**Intercluster distance**  $d(A, B) := \min_{x \in A, y \in B} d(x, y)$



# Single-linkage algorithm

While  $|\mathcal{C}^i| > k$ :

$$d_i = \min_{A, B \in \mathcal{C}^i} d(A, B)$$

while there exist  $A, B \in \mathcal{C}^i$  with  $d(A, B) = d_i$ :

replace  $A$  and  $B$  with the merged cluster  $A \cup B$  in  $\mathcal{C}^i$



# Single-linkage algorithm

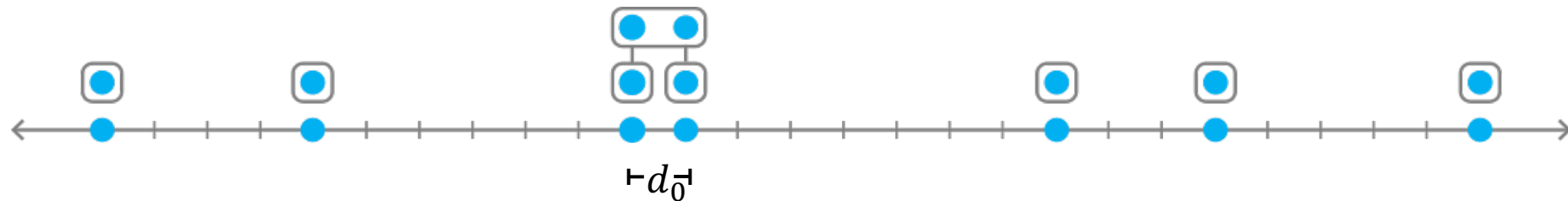
While  $|\mathcal{C}^i| > k$ :

$$d_i = \min_{A, B \in \mathcal{C}^i} d(A, B)$$

while there exist  $A, B \in \mathcal{C}^i$  with  $d(A, B) = d_i$ :

replace  $A$  and  $B$  with the merged cluster  $A \cup B$  in  $\mathcal{C}^i$

$\mathcal{C}^{i+1} = \mathcal{C}^i; i++$



# Single-linkage algorithm

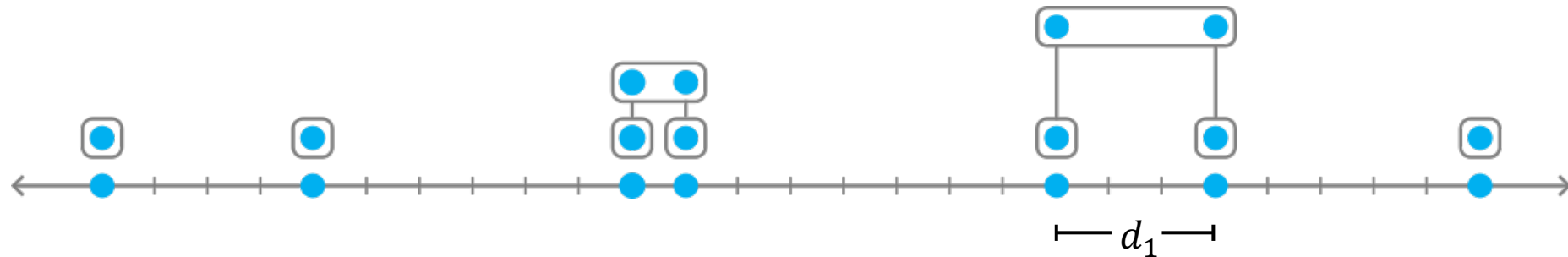
While  $|\mathcal{C}^i| > k$ :

$$d_i = \min_{A, B \in \mathcal{C}^i} d(A, B)$$

while there exist  $A, B \in \mathcal{C}^i$  with  $d(A, B) = d_i$ :

replace  $A$  and  $B$  with the merged cluster  $A \cup B$  in  $\mathcal{C}^i$

$\mathcal{C}^{i+1} = \mathcal{C}^i; i++$



# Single-linkage algorithm

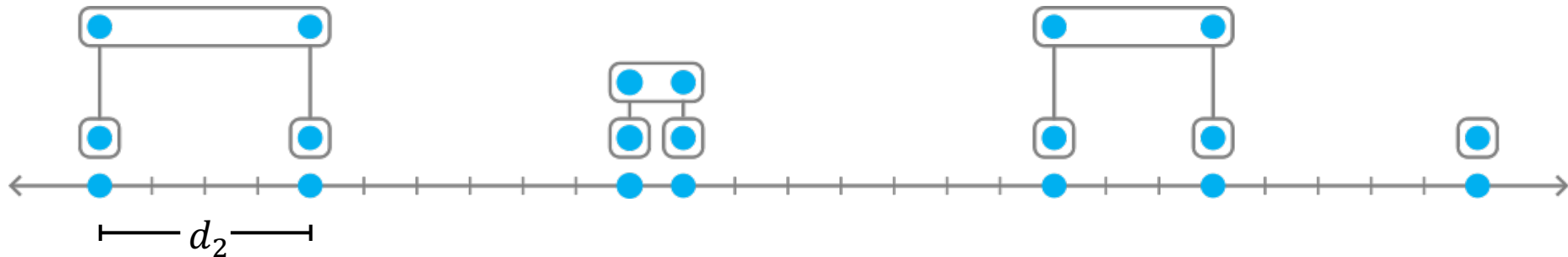
While  $|\mathcal{C}^i| > k$ :

$$d_i = \min_{A, B \in \mathcal{C}^i} d(A, B)$$

while there exist  $A, B \in \mathcal{C}^i$  with  $d(A, B) = d_i$ :

replace  $A$  and  $B$  with the merged cluster  $A \cup B$  in  $\mathcal{C}^i$

$\mathcal{C}^{i+1} = \mathcal{C}^i; i++$



# Single-linkage algorithm

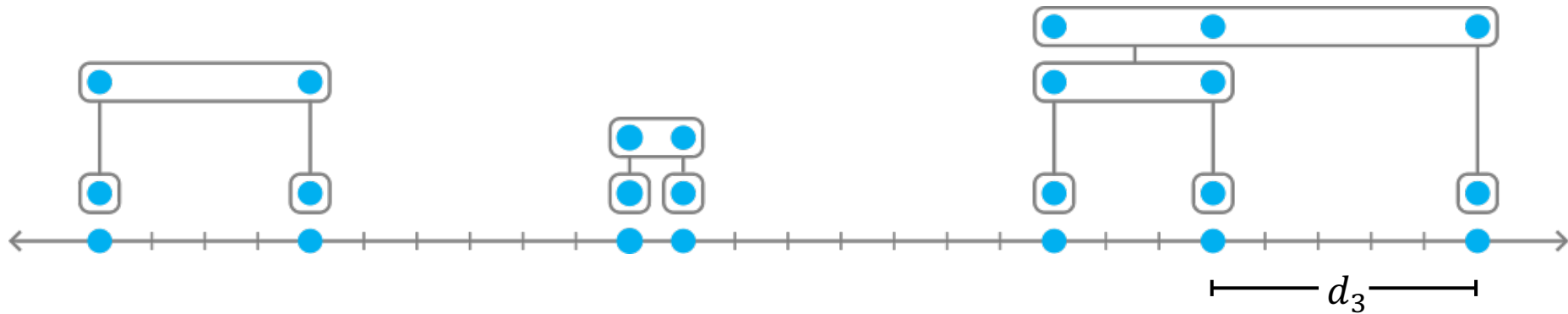
While  $|\mathcal{C}^i| > k$ :

$$d_i = \min_{A, B \in \mathcal{C}^i} d(A, B)$$

while there exist  $A, B \in \mathcal{C}^i$  with  $d(A, B) = d_i$ :

replace  $A$  and  $B$  with the merged cluster  $A \cup B$  in  $\mathcal{C}^i$

$\mathcal{C}^{i+1} = \mathcal{C}^i; i++$



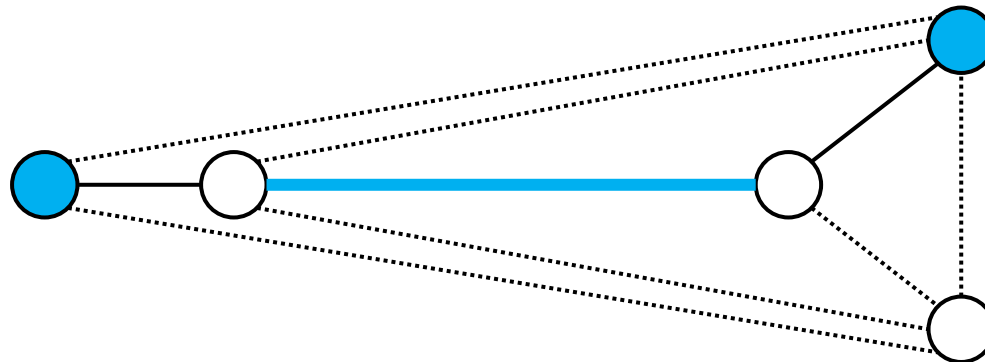


# Min-max distance

**Min-max distance** (or **bottleneck cost**) between  $x, y \in \mathcal{X}$ :

$$d_{\text{mm}}(x, y; \mathcal{X}) = \min_{\mathbf{p}} \max_i d(p_i, p_{i+1})$$

Taken over all simple paths  $\mathbf{p} = (p_1 = x, p_1, \dots, p_t = y)$  in complete graph over  $\mathcal{X}$  with edge weights  $d(x, y)$

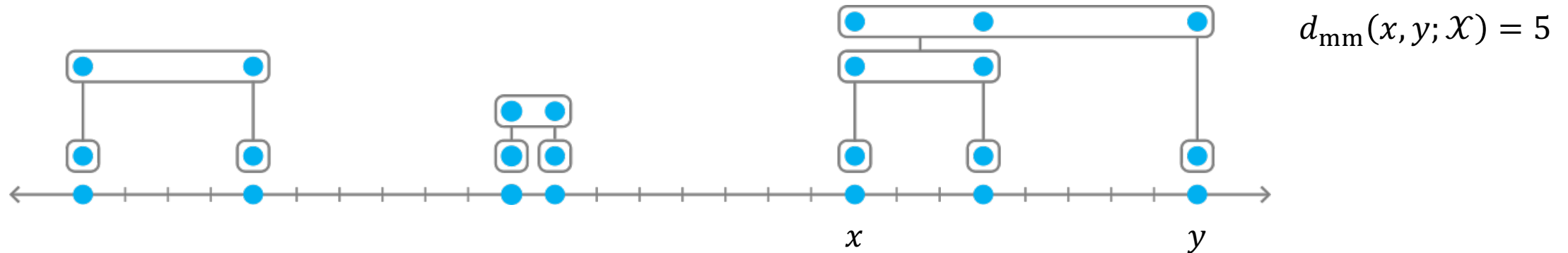


# Min-max distance

**Min-max distance** (or **bottleneck cost**) between  $x, y \in \mathcal{X}$ :

$$d_{\text{mm}}(x, y; \mathcal{X}) = \min_{\mathbf{p}} \max_i d(p_i, p_{i+1})$$

Taken over all simple paths  $\mathbf{p} = (p_1 = x, p_1, \dots, p_t = y)$  in complete graph over  $\mathcal{X}$  with edge weights  $d(x, y)$



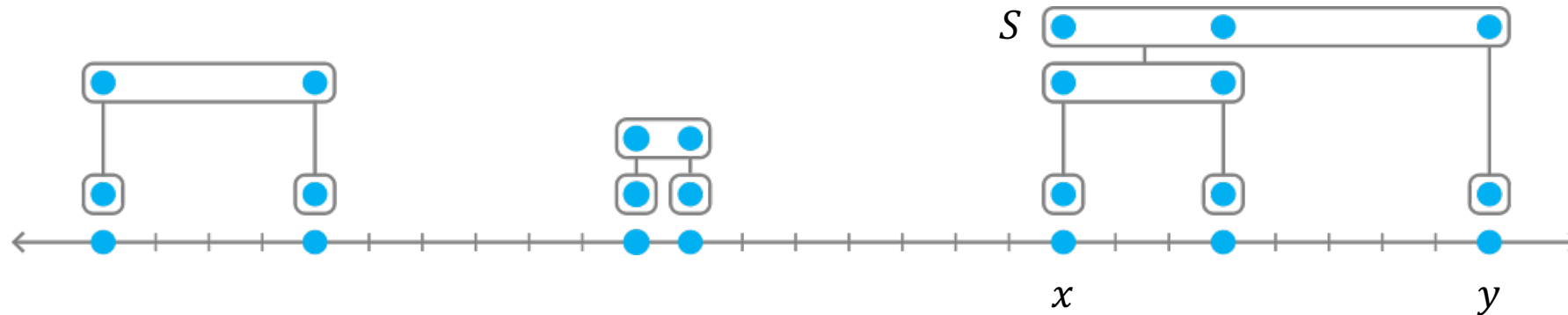
# Min-max distance

**Min-max distance** (or **bottleneck cost**) between  $x, y \in \mathcal{X}$ :

$$d_{\text{mm}}(x, y; \mathcal{X}) = \min_p \max_i d(p_i, p_{i+1})$$

For  $S \subseteq \mathcal{X}$ :

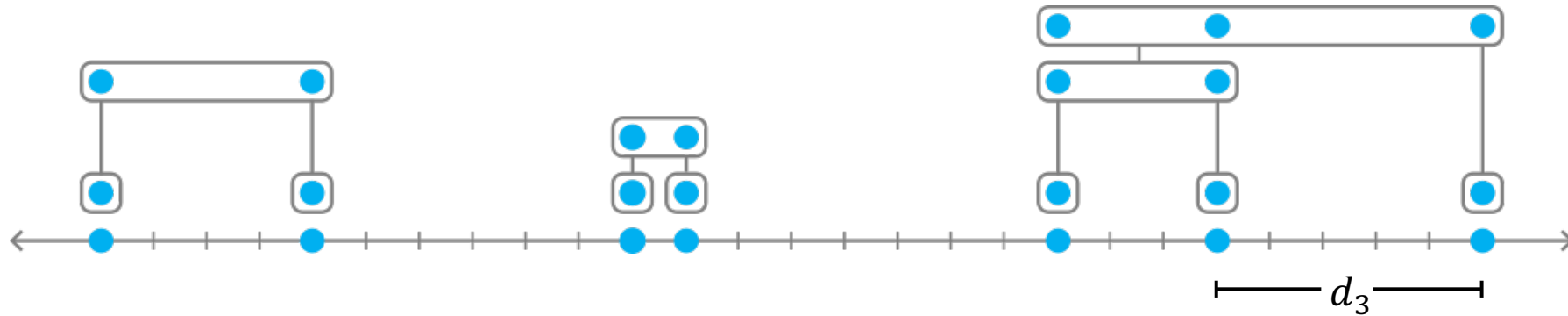
$$d_{\text{mm}}(S; \mathcal{X}) = \max_{x, y \in S} d_{\text{mm}}(x, y; \mathcal{X})$$



$$d_{\text{mm}}(x, y; \mathcal{X}) = 5$$
$$d_{\text{mm}}(S; \mathcal{X}) = 5$$

# Key lemma

**Lemma:**  $x, y$  are merged by round  $\ell$  if and only if  
$$d_{\text{mm}}(x, y; \mathcal{X}) \leq d_\ell$$



# Min-max distance distortion

Suppose we run SL on subsample  $\mathcal{X}_m$  of  $\mathcal{X}$  of size  $m$

Clusters will be merged based on  $d_{\text{mm}}(x, y; \mathcal{X}_m) \geq d_{\text{mm}}(x, y; \mathcal{X})$



If  $d_{\text{mm}}(x, y; \mathcal{X}_m) \gg d_{\text{mm}}(x, y; \mathcal{X})$ :

Clustering may be highly **distorted** on subsample



If  $d_{\text{mm}}(x, y; \mathcal{X}_m) \approx d_{\text{mm}}(x, y; \mathcal{X})$  for all  $x, y$ :

Clustering on  $\mathcal{X}_m$  should be **similar** to clustering on  $\mathcal{X}$

# Min-max distance distortion

**Goal of this section** (a more philosophical goal 🤔):

Characterize which property of the dataset  $\mathcal{X}$  either:

- Allows for size generalization when this property holds, or
- Prohibits size generalization when it does not

$C_1, \dots, C_k$  are the clusters returned by single linkage on  $\mathcal{X}$

$$\zeta_{k,SL}(\mathcal{X}) = |\mathcal{X}| \left[ \frac{\min_{i,j \in [k]} d_{mm}(C_i \cup C_j; \mathcal{X}) - \max_{t \in [k]} d_{mm}(C_t; \mathcal{X})}{\max_{t \in [k]} d_{mm}(C_t; \mathcal{X})} \right]^{-1}$$

Minimum intercluster distance

Maximum intracluster distance

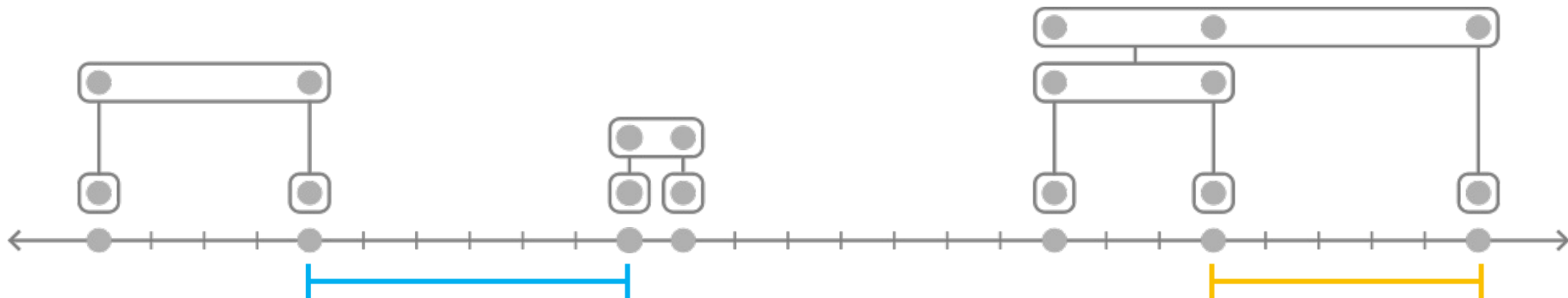
# Min-max distance distortion

$C_1, \dots, C_k$  are the clusters returned by single linkage on  $\mathcal{X}$

$$\zeta_{k,SL}(\mathcal{X}) = |\mathcal{X}| \left| \frac{\min_{i,j \in [k]} d_{mm}(C_i \cup C_j; \mathcal{X}) - \max_{t \in [k]} d_{mm}(C_t; \mathcal{X})}{\max_{t \in [k]} d_{mm}(C_t; \mathcal{X})} \right|^{-1}$$

Minimum intercluster distance

Maximum intracluster distance



# Single-linkage: Main results

## Theorem:

- $\mathcal{C} = \{C_1, \dots, C_k\}$  are the clusters returned by single linkage on  $\mathcal{X}$
- $\mathcal{C}'$  are the clusters returned by single linkage on  $\mathcal{X}_m$  with

$$m = \tilde{O} \left( \frac{k}{\epsilon^2} + \frac{|\mathcal{X}|}{\min |C_i|} + \zeta_{k,SL}(\mathcal{X}) \right) \text{ uniform samples}$$

- For any ground truth clustering  $\mathcal{G}$ , with high probability,  
$$|\text{cost}_{\mathcal{G}}(\mathcal{C}; \mathcal{X}) - \text{cost}_{\mathcal{G}}(\mathcal{C}'; \mathcal{X}_m)| \leq \epsilon$$

Can be computed using  $m^2$  distance queries and  $m$  ground truth queries



# Single-linkage: Main results

## Theorem (informal):

Also construct instances where

$$m = \Omega\left(\frac{|\mathcal{X}|}{\min |C_i|}\right)$$

and

$$m = \Omega\left(\zeta_{k,SL}(\mathcal{X})\right)$$

samples are **necessary** to ensure with constant probability

$$|\text{cost}_g(\mathcal{C}; \mathcal{X}) - \text{cost}_g(\mathcal{C}'; \mathcal{X}_m)| \leq \text{constant}$$

# Outline


1. Introduction
2. Center-based clustering
3. Single-linkage
- 4. Conclusions and future directions**

# Summary

Given a massive combinatorial problem, can we:

1. "Shrink" it
2. Evaluate candidate algorithms on the smaller instance
3. Provably guarantee:  
The best algorithm on the small instance  
...is also best on the original large instance?

Answer this question in  
the affirmative for

- 
1. Gonzalez's k-centers heuristic\*
  2. k-means++
  3. Single-linkage clustering

# Future directions

Given a massive combinatorial problem, can we:

1. "Shrink" it
2. Evaluate candidate algorithms on the smaller instance
3. Provably guarantee:  
    The best algorithm on the small instance  
    ...is also best on the original large instance?

**For what other problems can we answer this question?**

Graph algorithms, integer programming, ...?

# From Large to Small Datasets: Size Generalization for Clustering Algorithm Selection



**Vaggos Chatziafratis**  
UC Santa Cruz



**Ishani Karmarkar**  
Stanford



**Ellen Vitercik**  
Stanford