ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Data Mining
Learning from Large Data Sets

Lecture 8 – Clustering large data sets

263-5200-00L

Andreas Krause

Announcements

- Homework 4 out tomorrow

Course organization

- **Retrieval**
  - Given a query, find "most similar" item in a large data set
  - Determine relevance of search results
  - *Applications*: GoogleGoggles, Shazam, …
- **Supervised learning** (Classification, Regression)
  - Learn a concept (function mapping queries to labels)
  - *Applications*: Spam filtering, predicting price changes, …
- **Unsupervised learning** (Clustering, dimension reduction)
  - Identify clusters, "common patterns"; anomaly detection
  - *Applications*: Recommender systems, fraud detection, …
- **Learning with limited feedback**
  - Learn to optimize a function that's expensive to evaluate
  - *Applications*: Online advertising, opt. UI, learning rankings, …

Unsupervised learning

- "Learning without labels"
- Typically useful for exploratory data analysis ("find patterns"; visualization; …)
- Most common methods:
  - Clustering (unsupervised classification)
  - Dimension reduction (unsupervised regression)

What is clustering?

- Given data points, group into clusters such that
  - *Similar* points are in the same cluster
  - *Dissimilar* points are in different clusters

- Points are typically represented either
  - in (high-dimensional) Euclidean space
  - in a metric space, given in terms of pairwise distances (Jaccard, cosine, …)

- Anomaly / outlier detection: Identification of points that "don't fit well in any of the clusters"

Examples of clustering

- Cluster
  - Documents based on the words they contain
  - Images based on image features
  - DNA sequences based on edit distance
  - Products based on which customers bought them
  - Customers based on their purchase history
  - Web surfers based on their queries / sites they visit
  - …

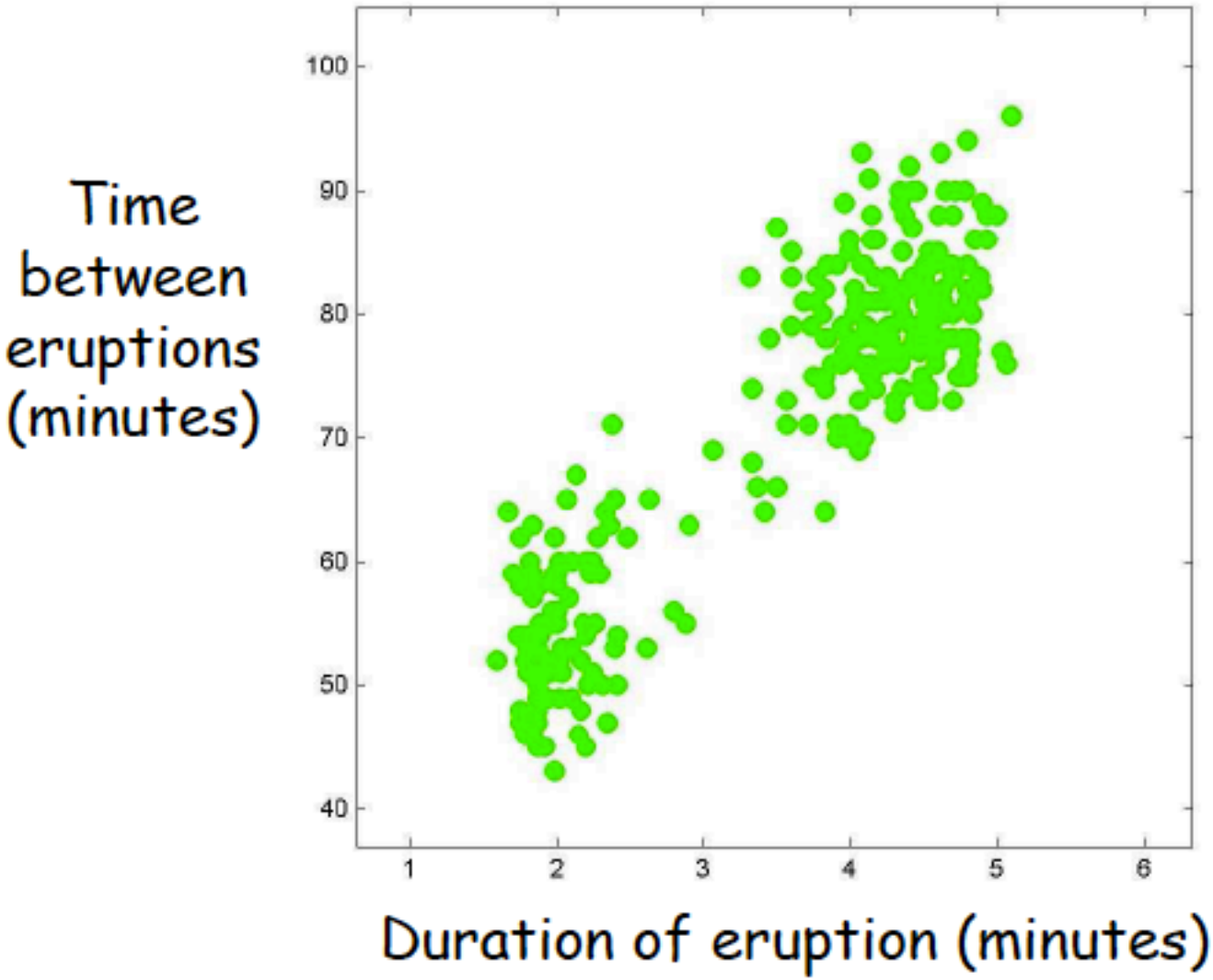
Standard approaches to clustering

- Hierarchical clustering
  - Build a tree (either bottom-up or top-down), representing the distances among the data points
  - Example: single-, average- linkage agglomerative clustering
- Partitional approaches
  - Define and optimize a notion of "goodness" defined over partitions
  - Example: Spectral clustering, graph-cut based approaches
- Model-based approaches
  - Maintain cluster "models" and infer cluster membership (e.g., assign each point to closest center)
  - Example: k-means, Gaussian mixture models, …

We will

- Review standard clustering algorithms
  - K-means
  - Probabilistic mixture models
- Discuss how to scale them to massive data sets and data streams

Clustering example



k-means

- Assumes points are in Euclidean space $\mathbf{x}_i \in \mathbb{R}^d$
- Represent clusters as centers $\mu_j \in \mathbb{R}^d$
- Each point is assigned to closest center

Goal: Pick centers to minimize average squared distance

$$L(\mu) = \sum_{i=1}^{N} \min_j ||\mu_j - \mathbf{x}_i||_2^2$$

- Non-convex optimization!
- NP-hard → can't solve optimally in general

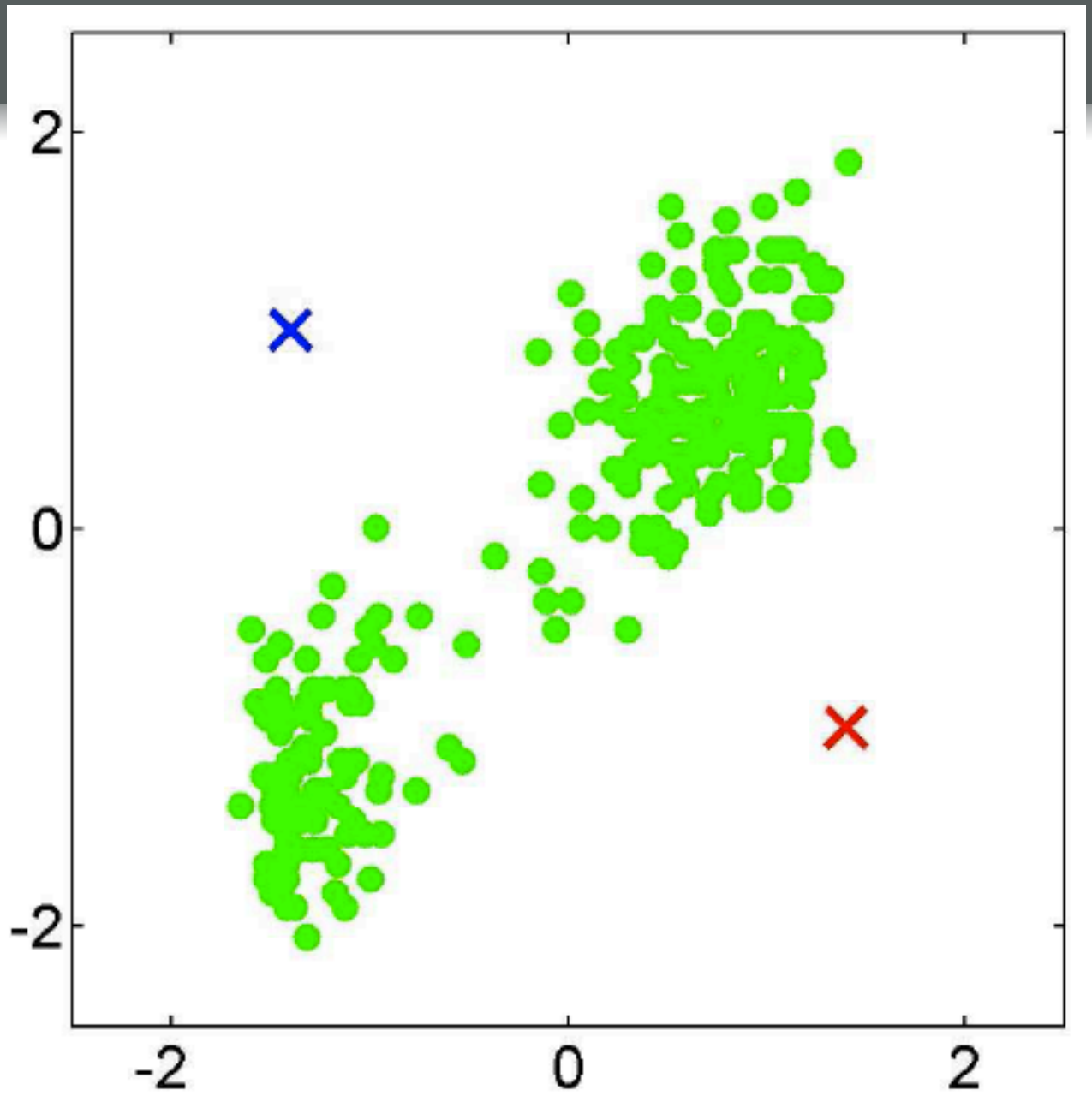
Classical k-means algorithm

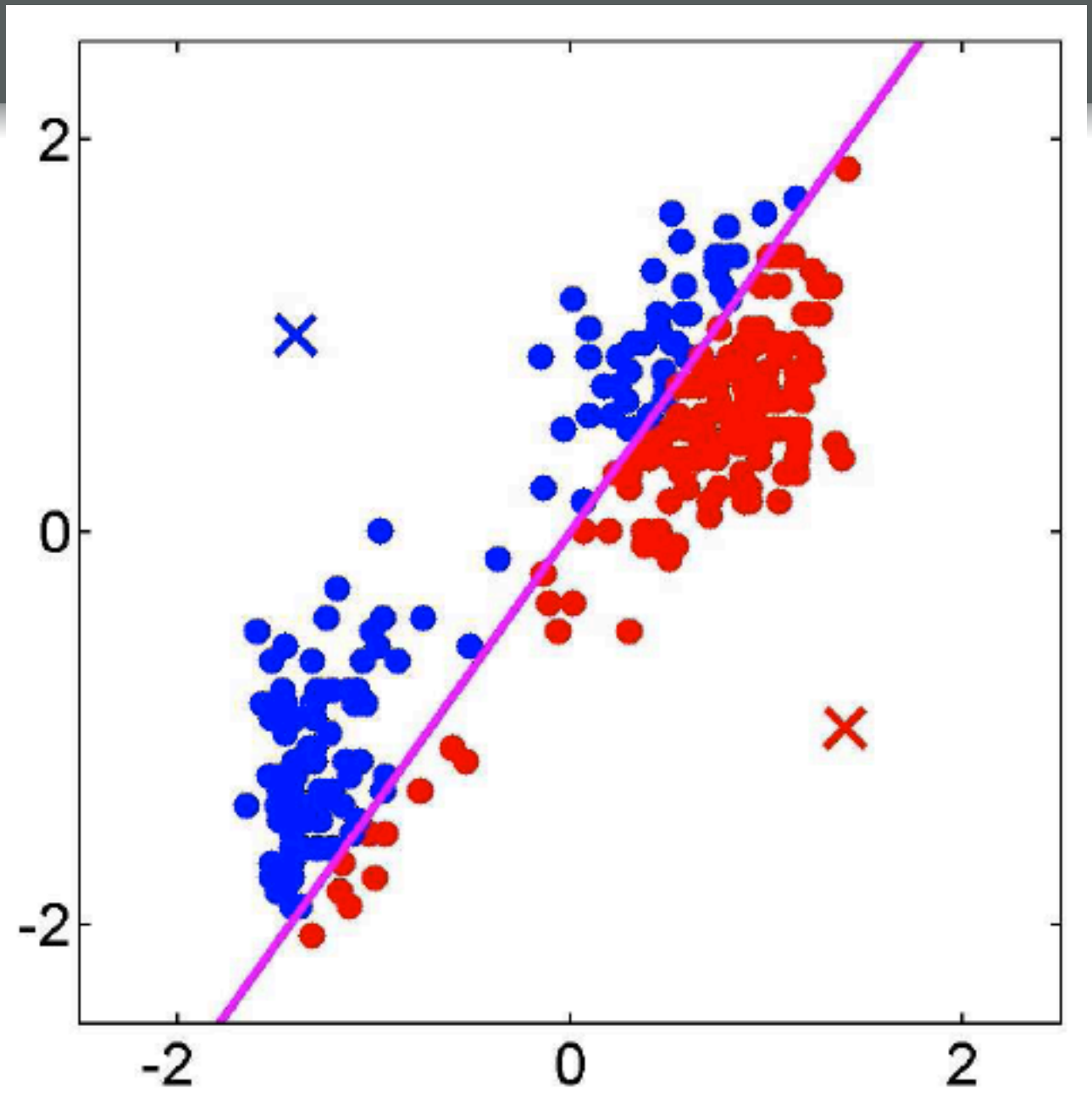
- Initialize cluster centers
  - E.g., pick one point at random, the other ones with maximum distance
- While not converged
  - Assign each point x_i to closest center

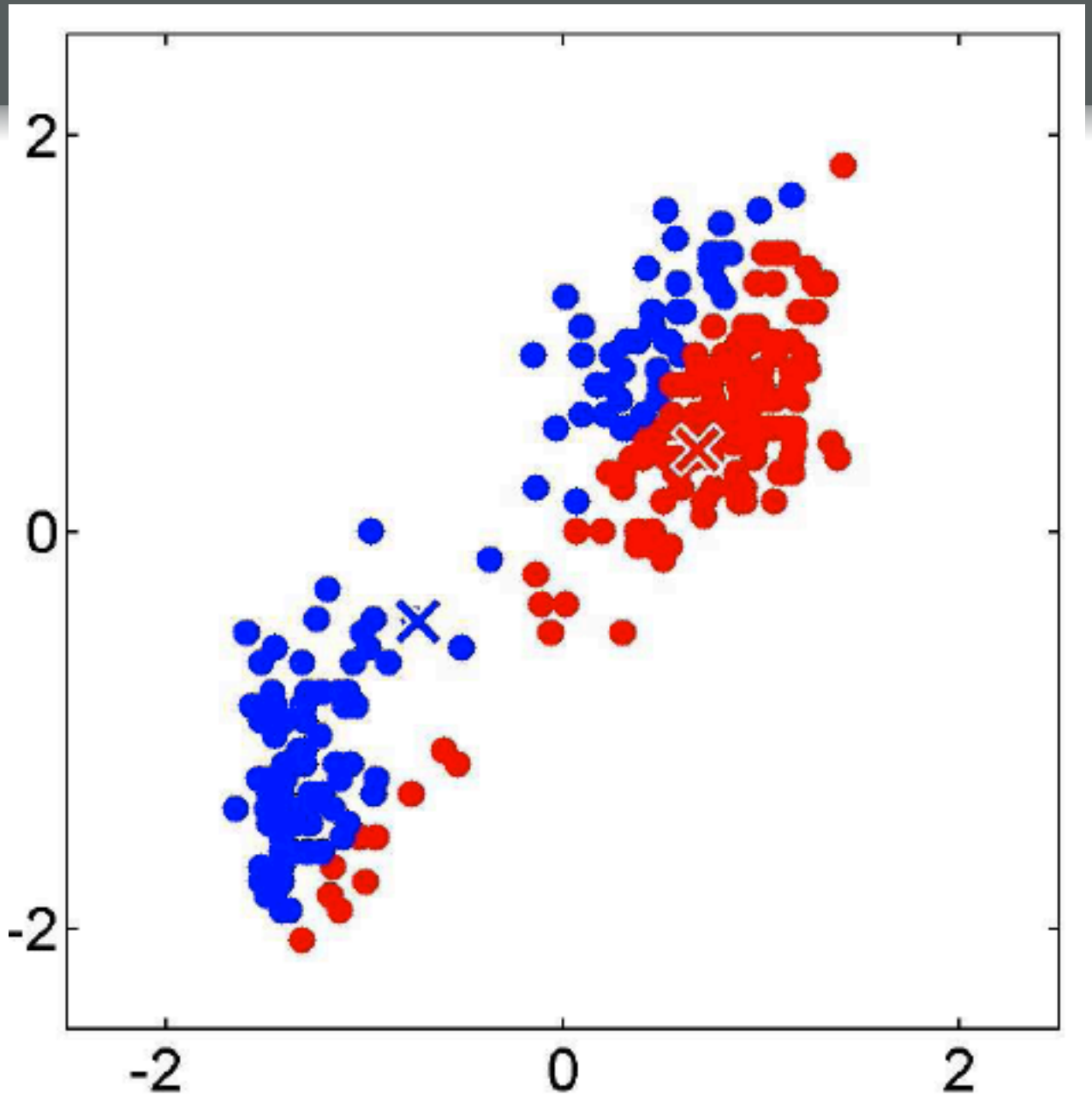
$$c_i \leftarrow \arg\min_j \|x_i - \mu_j\|_2^2$$

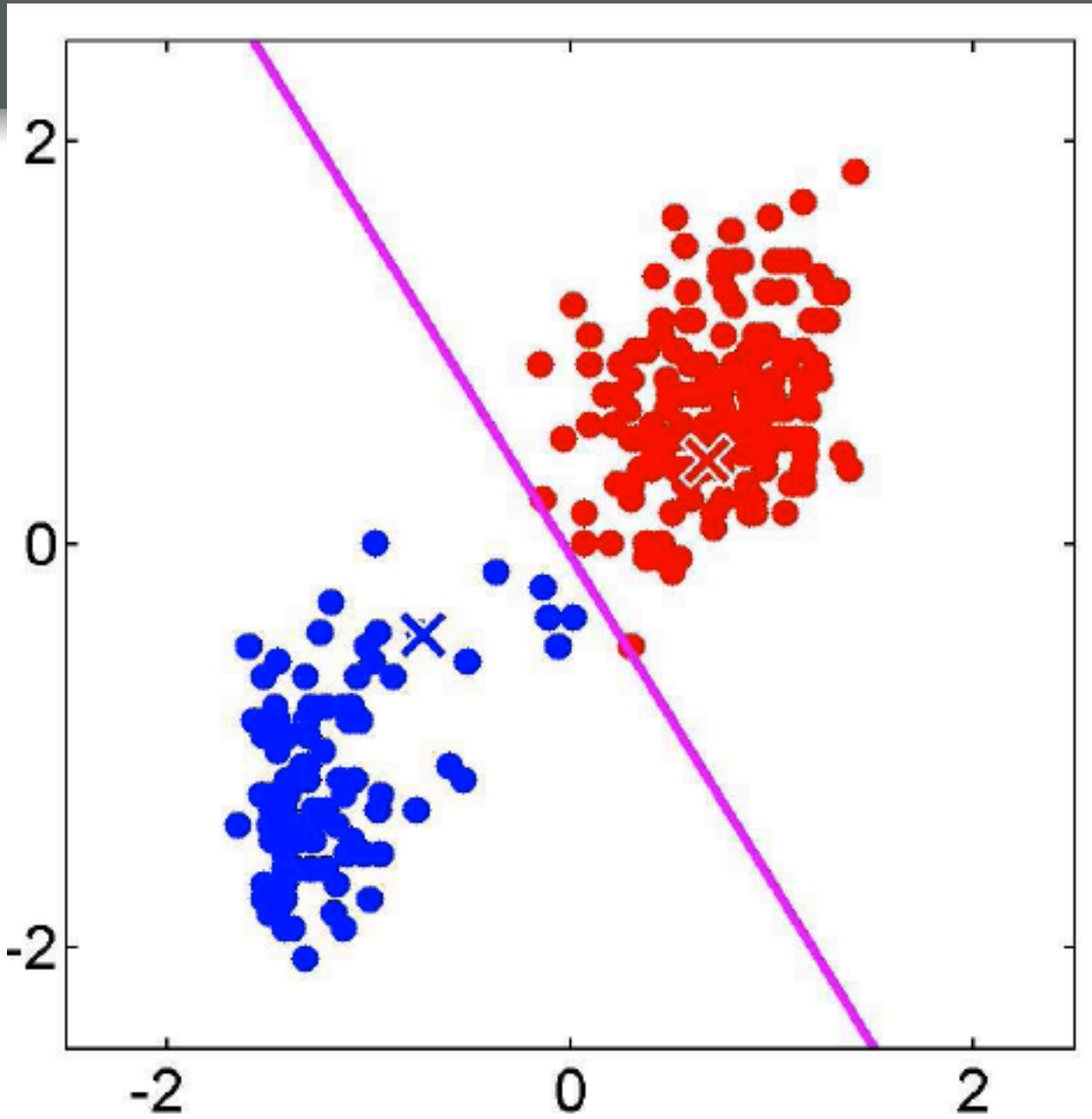
  - Update center as mean of assigned data points

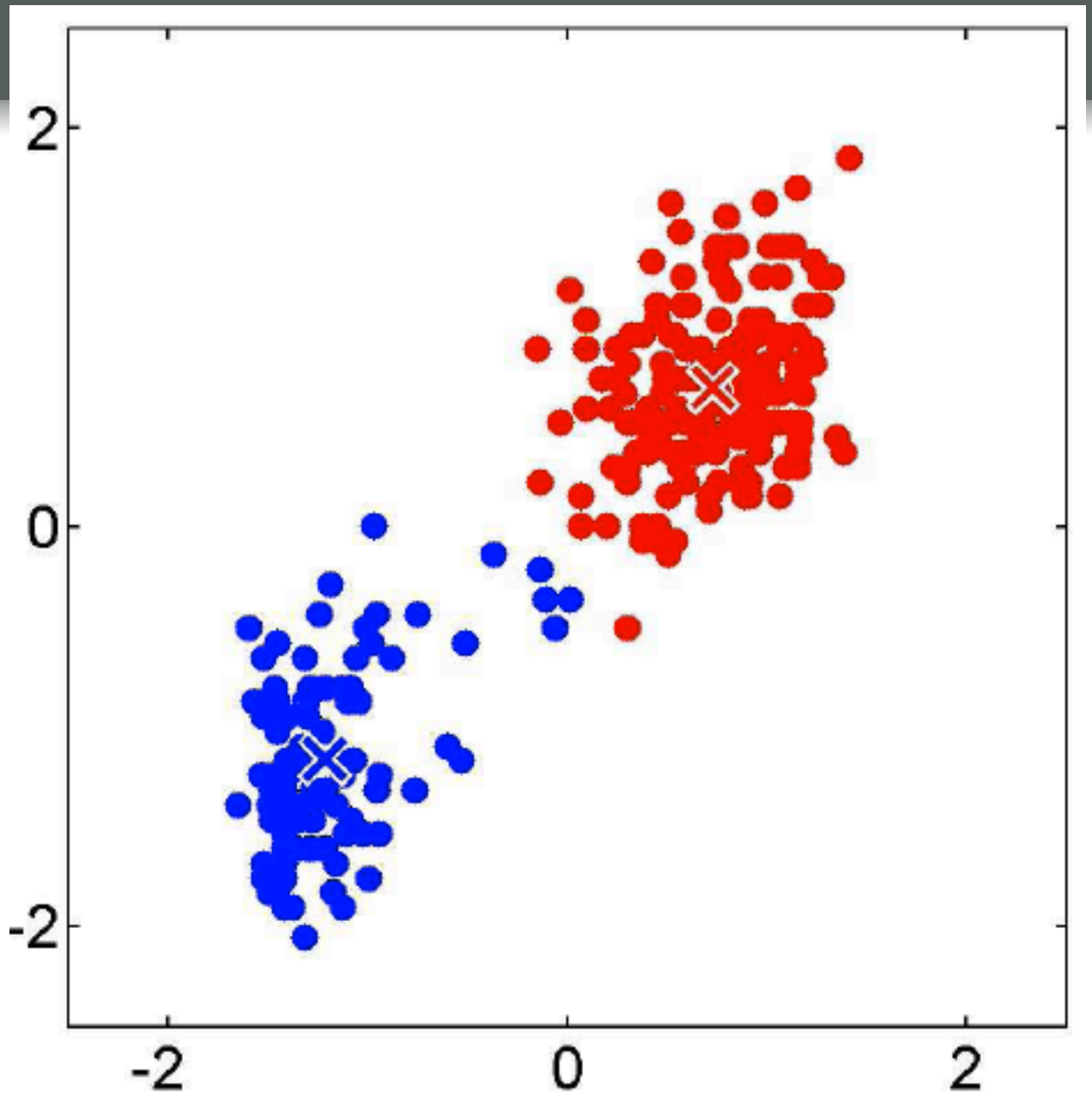
$$\mu_j = \frac{1}{n_j} \sum_{i: c_i = j} x_i$$

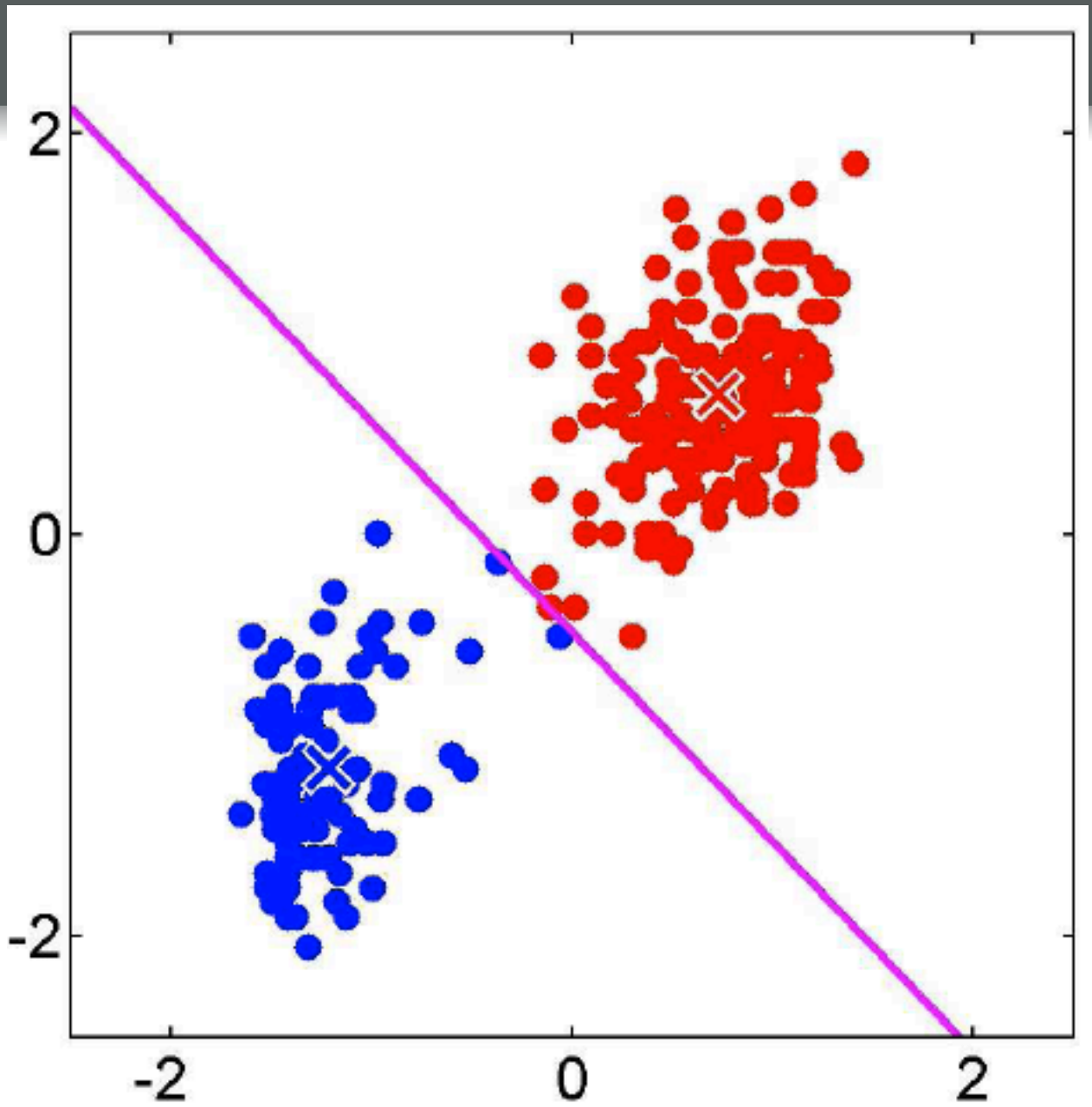

2
0
-2
-2
0
2

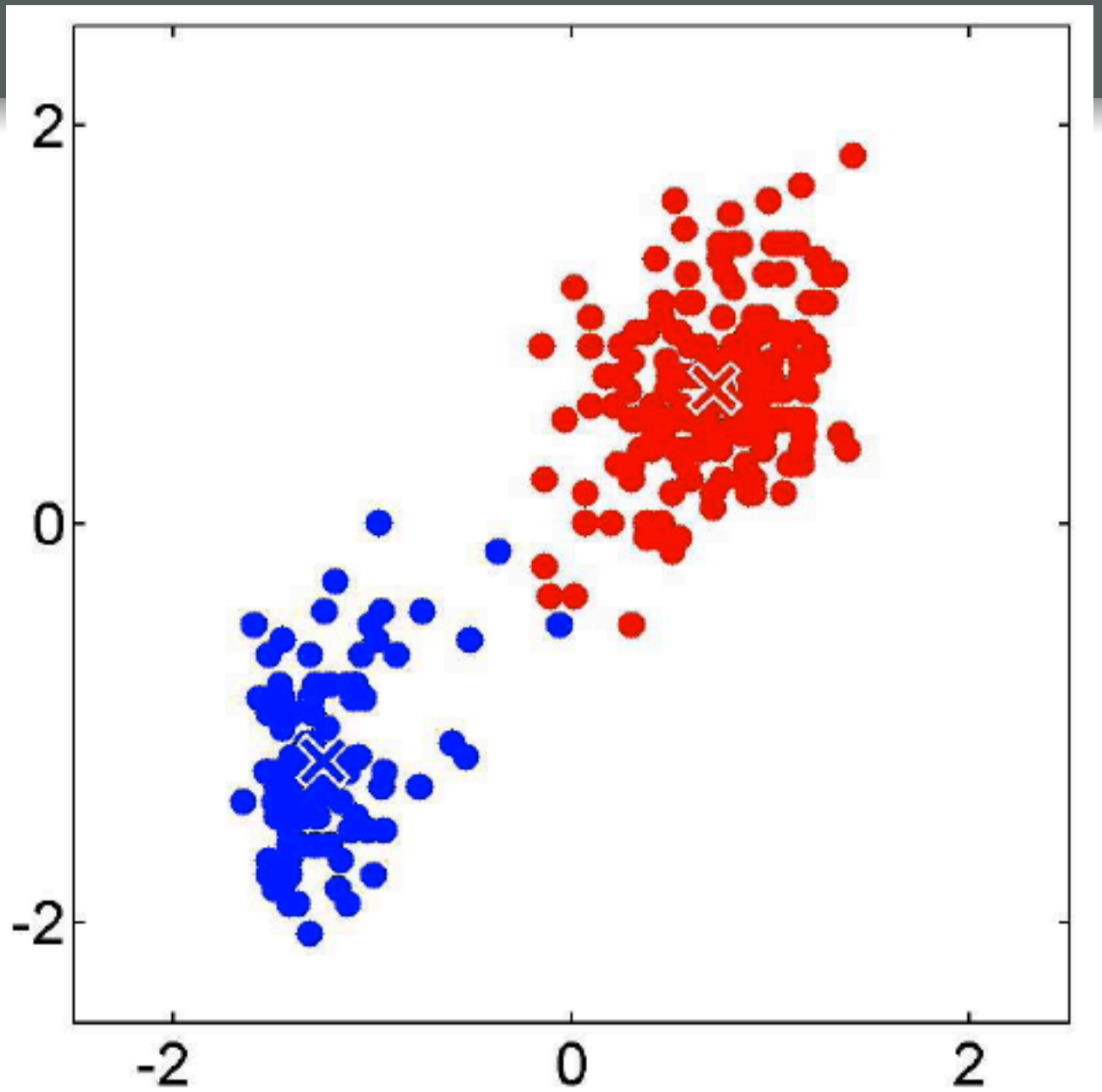

2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2


2
0
-2
-2
0
2

Properties of k-means

- Guaranteed to monotonically decrease average squared distance in each iteration

$$L(\mu^{(t)}) = \sum_{i=1}^{N} \min_j ||\mu_j^{(t)} - \mathbf{x}_i||_2^2$$

$$L(\mu^{(t+1)}) \leq L(\mu^{(t)})$$

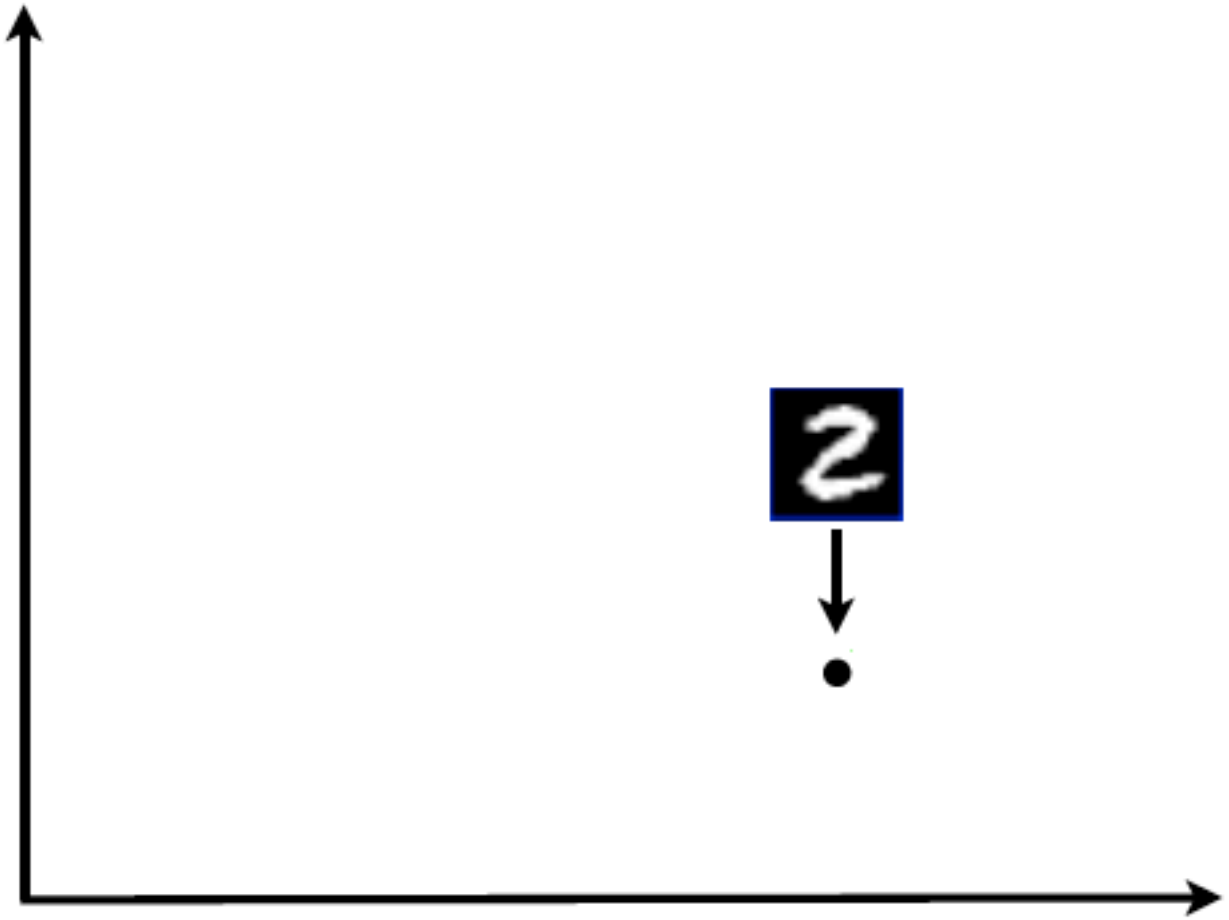
- Converges to a local optimum
- Complexity:
  - Per iteration $n\ d\ k$

- Have to process entire data set in each iteration

K-means for large data sets / streams

- What if data set does not fit in main memory?
  - In principle not a problem (why?)
  - But each iteration still requires an entire pass through the data set
- Recall supervised learning (online SVM, etc.)
  - There we were able to process one data point at a time
  - Get (provably) good solutions from a single pass through the data
  - Could even do it in parallel!
- Can we do the same thing for clustering??

Streaming clustering



- How should me maintain clusters as new data arrives?

Recall online SVM

- Recall Online SVMs (& stochastic gradient descent)
- Loss function *decomposes additively* over data set

$$L(\mathbf{w}) = \sum_{i=1}^{N} \underbrace{\text{hinge}(\mathbf{x}_i; y_i, \mathbf{w})}_{f_i(w)}$$

- Can take a (sub-)gradient step for each data point

$$w^{(t+1)} = w^{(t)} - \eta_t \nabla f_t(w^{(t)})$$

Online k-means

- For k-means, loss function *also* decomposes additively over data set

$$L(\mu) = \sum_{i=1}^{N} \underbrace{\min_j ||\mu_j - \mathbf{x}_i||_2^2}_{f_i(\mu)}$$

- Let's try take a (sub-)gradient step for each data point

$$\mu^{(t+1)} \leftarrow \mu^{(t)} + \eta_t \nabla f_t(\mu^{(t)})$$

Calculating the gradient

$$L(\mu) = \sum_{i=1}^{N} \underbrace{\min_j ||\mu_j - \mathbf{x}_i||_2^2}_{f_i(\mu)}$$

$\mu = [\mu_1 \cdots \mu_k] \in \mathbb{R}^{d \times k}$

$$\frac{\partial f_i(\mu)}{\partial \mu_j} = \begin{cases} 0 & \text{if } j \neq \operatorname{argmin} \|\mu_j - x_i\|_2^2 \\ \frac{\partial \|\mu_j - x_i\|_2^2}{\partial \mu_j} = 2(\mu_j - x_i) & \text{otherwise} \end{cases}$$

Online k-means algorithm

- Initialize centers randomly
- For t = 1:N
  - Find $c = \arg\min_j ||\mu_j - \mathbf{x}_t||_2$
  - Set $\mu_c \leftarrow \mu_c + \eta_t(\mathbf{x}_t - \mu_c)$

- To converge to local optimum, need that

$$\sum_t \eta_t = \infty \qquad \sum_t \eta_t^2 < \infty$$

E.g. $\eta_t = \frac{1}{t}$

Practical aspects

- Generally works best if data is «randomly» ordered (like stochastic gradient descent)
- Typically, want to choose larger value for k
- How can one implement multiple random restarts in one pass?

Problems with online k-means

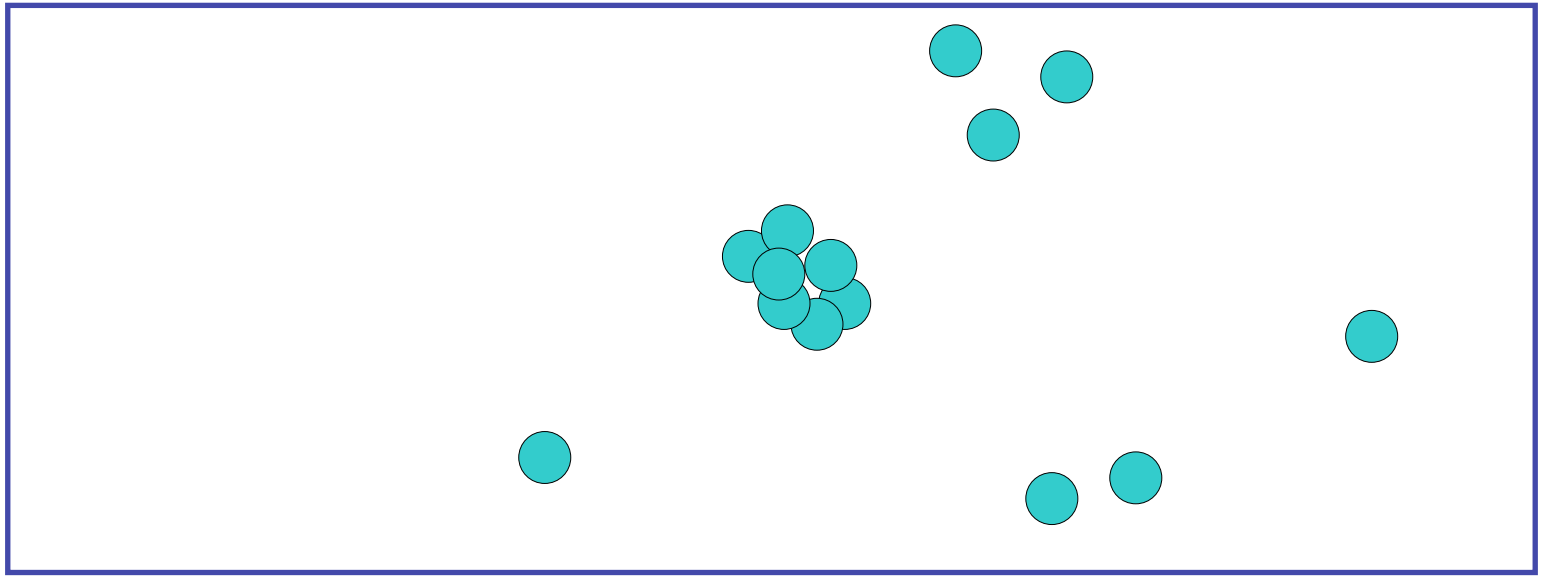
- Have to commit to "k" in advance
- Objective function non-convex (and problem NP-hard)
  → guarantees for online convex programming / SGD do not apply!
- Not clear how to parallelize

Alternative: Summarizing large data sets

- Idea:
  - Efficiently construct a *compact version* C of the data set D such that solving k-means on C gives a good solution to D

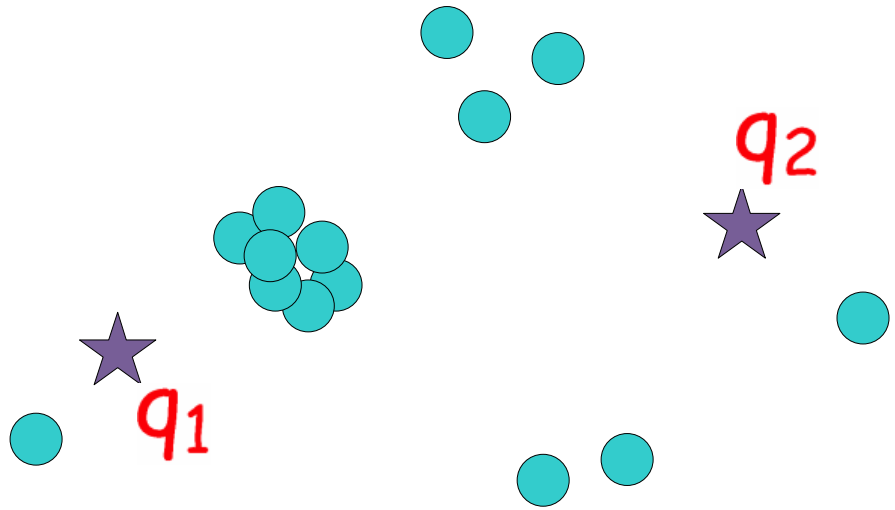
- Approach:
  - First construct C such that it allows *approximately answer* "k-means queries"
    i.e., approximately evaluate $L(\mu) = \sum_{i=1}^{N} \min_j ||\mu_j - \mathbf{x}_i||_2^2$
  - Then solve k-means using the approximate loss function

k-mean queries



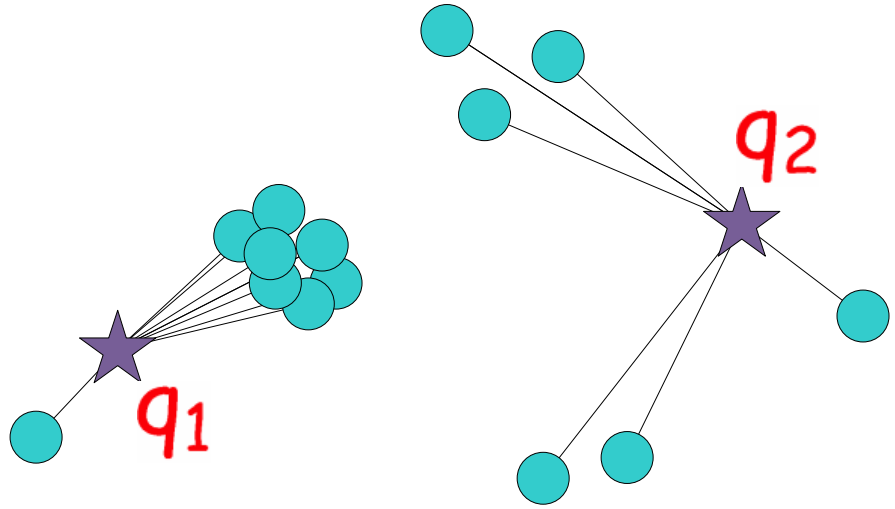
$k = 2$

$Q = \{q_1, q_2\}$

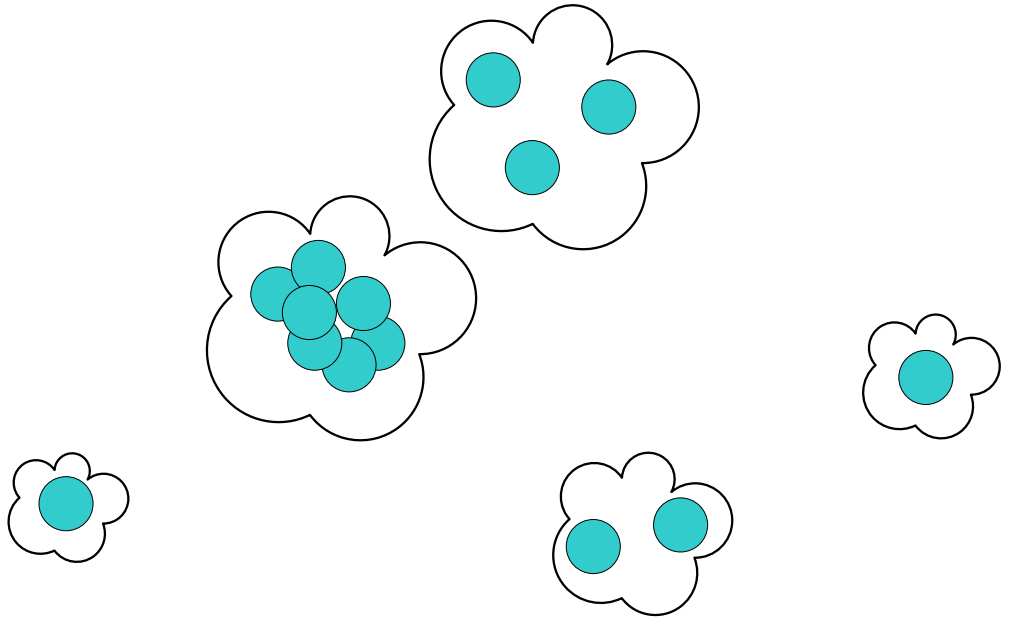


$k = 2$

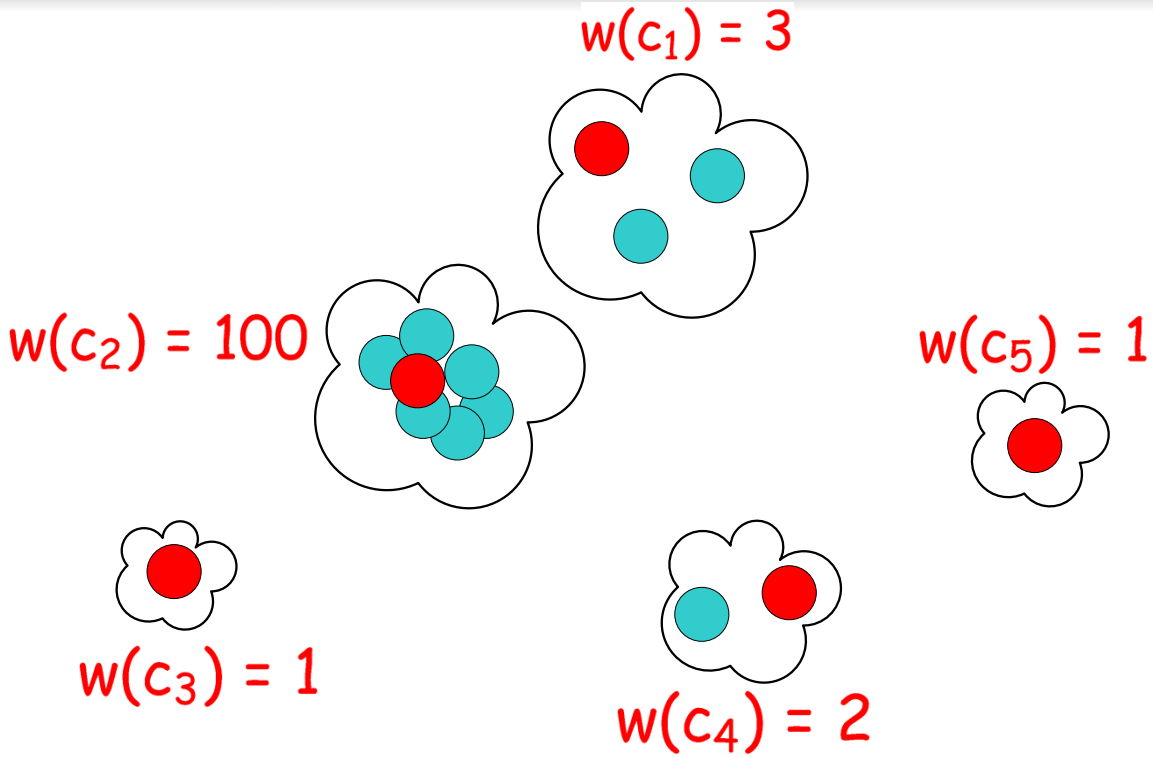
$Q = \{q_1, q_2\}$



Data set summarization for k-means



Data set summarization for k-means

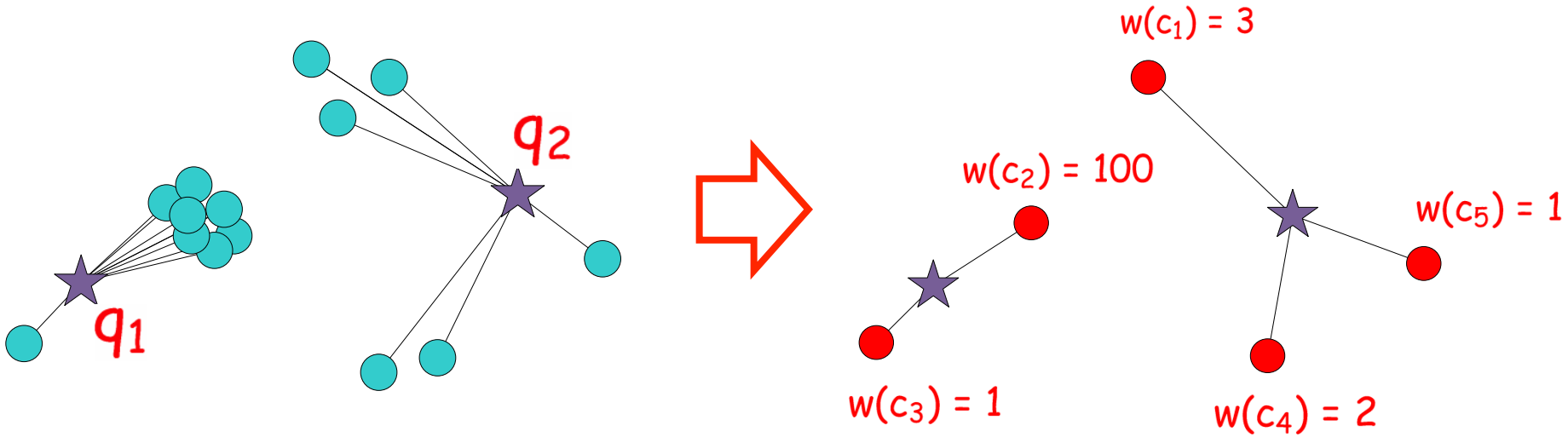


- **Key idea**: Replace many points by one weighted representative

$$L_k(\mu; C) = \sum_{(w,\mathbf{x}) \in C} w \min_{j \in \{1,\ldots,k\}} ||\mu_j - \mathbf{x}||_2^2$$

Coresets

$$L_k(\mu; C) = \sum_{(w,\mathbf{x})\in C} w \min_{j\in\{1,\ldots,k\}} ||\mu_j - \mathbf{x}||_2^2$$



C is called a (k,ε)-coreset for data set D, if

$$(1-\varepsilon)L_k(\mu; D) \le L_k(\mu; C) \le (1+\varepsilon)L_k(\mu; D)$$

Constructing coresets

- Suppose for all pairs of points: $||\mathbf{x}_i - \mathbf{x}_j||_2 \leq 1$
- First attempt: Random sampling
  - Pick n << N points C uniformly at random from D

$$\mathbb{E}[L_k(\mu; C)] = L_k(\mu; D)$$

  - Hoeffding's inequality gives

$$\Pr\Big(|L_k(\mu; C) - L_k(\mu; D)| \geq \varepsilon\Big) \leq 2\exp\left(-2\varepsilon^2 n\right)$$

  - Thus need $\mathcal{O}\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$ points to ensure

absolute error at most ε with probability at least 1-δ

Constructing coresets

- Suppose for all pairs of points: $||\mathbf{x}_i - \mathbf{x}_j||_2 \leq 1$
- First attempt: Random sampling
  - Pick n << N points C uniformly at random from D
  - Assign uniform weights N/n

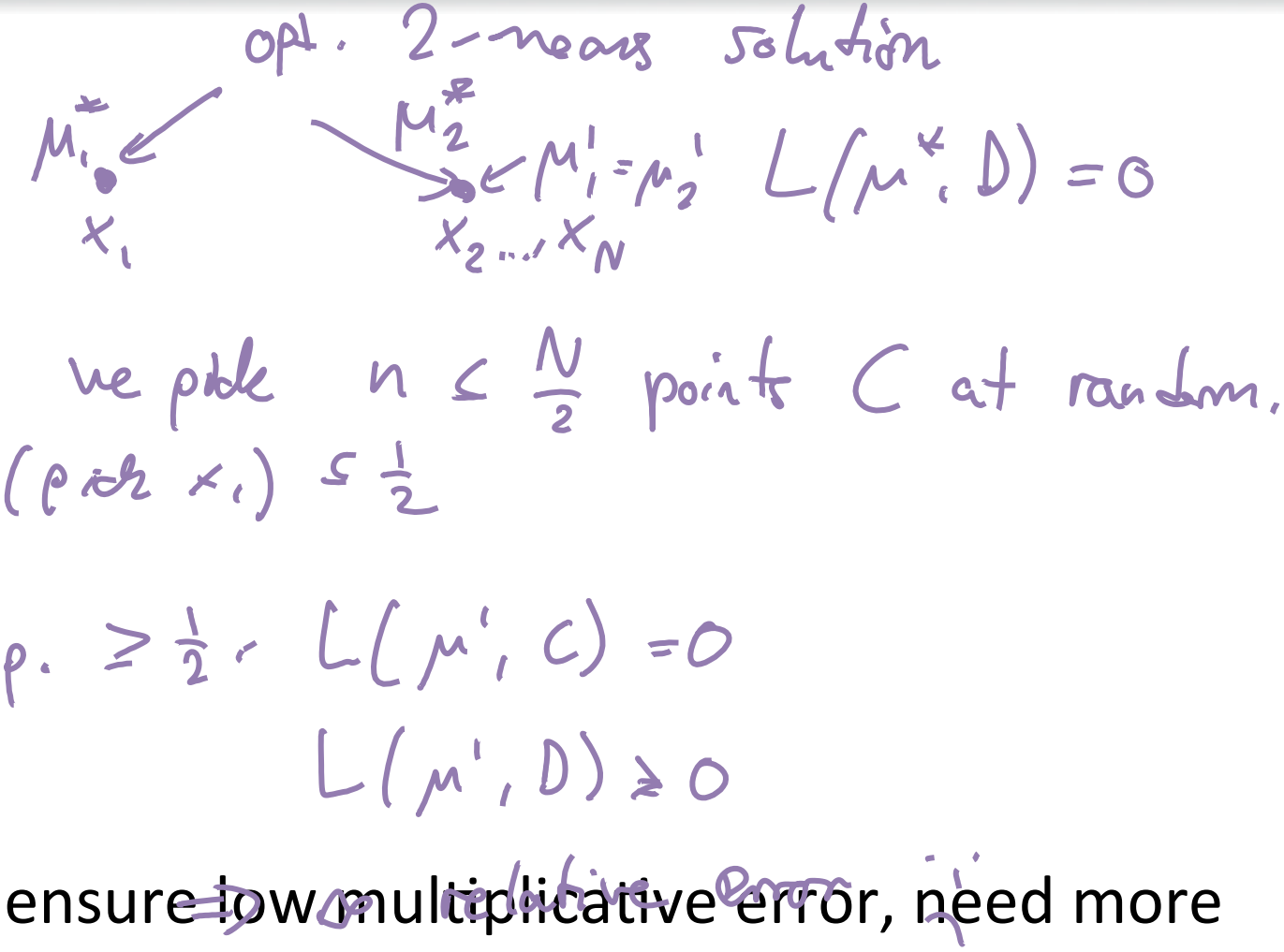
$R_i \sim U(\{1 \dots N\})$

$$L(\mu; C) = \sum_{i=1}^{n} \frac{N}{n} \min_j \|x_{R_i} - \mu_j\|_2^2$$

$$\mathbb{E}[L(\mu; C)] = \sum_{i=1}^{n} \frac{N}{n} \underbrace{\mathbb{E}_R\left[\min_j \|x_{R_i} - \mu_j\|_2^2\right]}_{\sum_{t=1}^{N} \frac{1}{N} \min_j \|x_t - \mu_j\|_2^2}$$

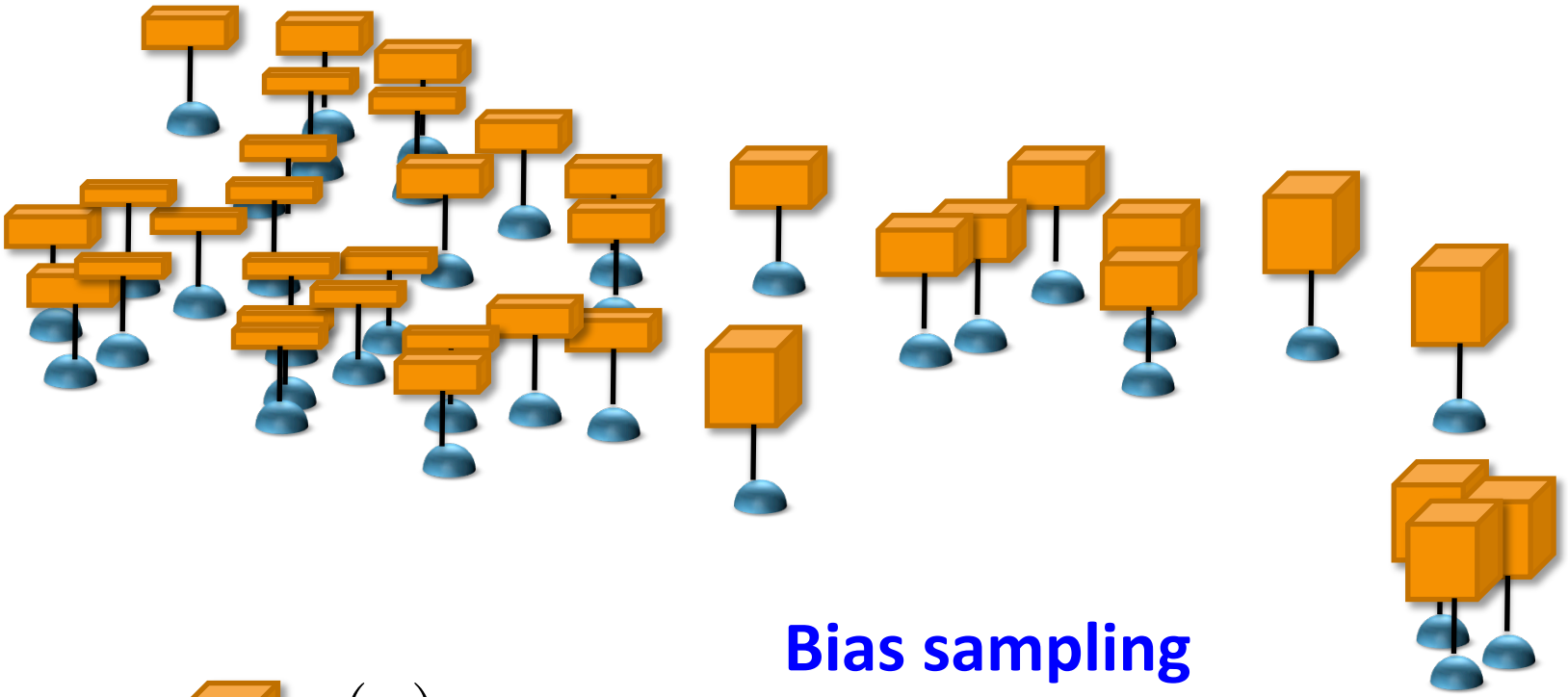
$$= \sum_{t=1}^{N} \min_j \|x_t - \mu_j\|_2^2 = L(\mu, 0)$$

Can we get small relative error?

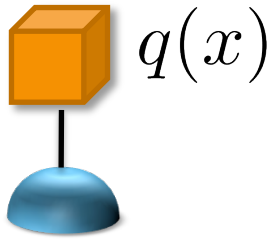


- To ensure low multiplicative error, need more complex construction
- ➔ will use non-uniform sampling!

Sampling Distribution

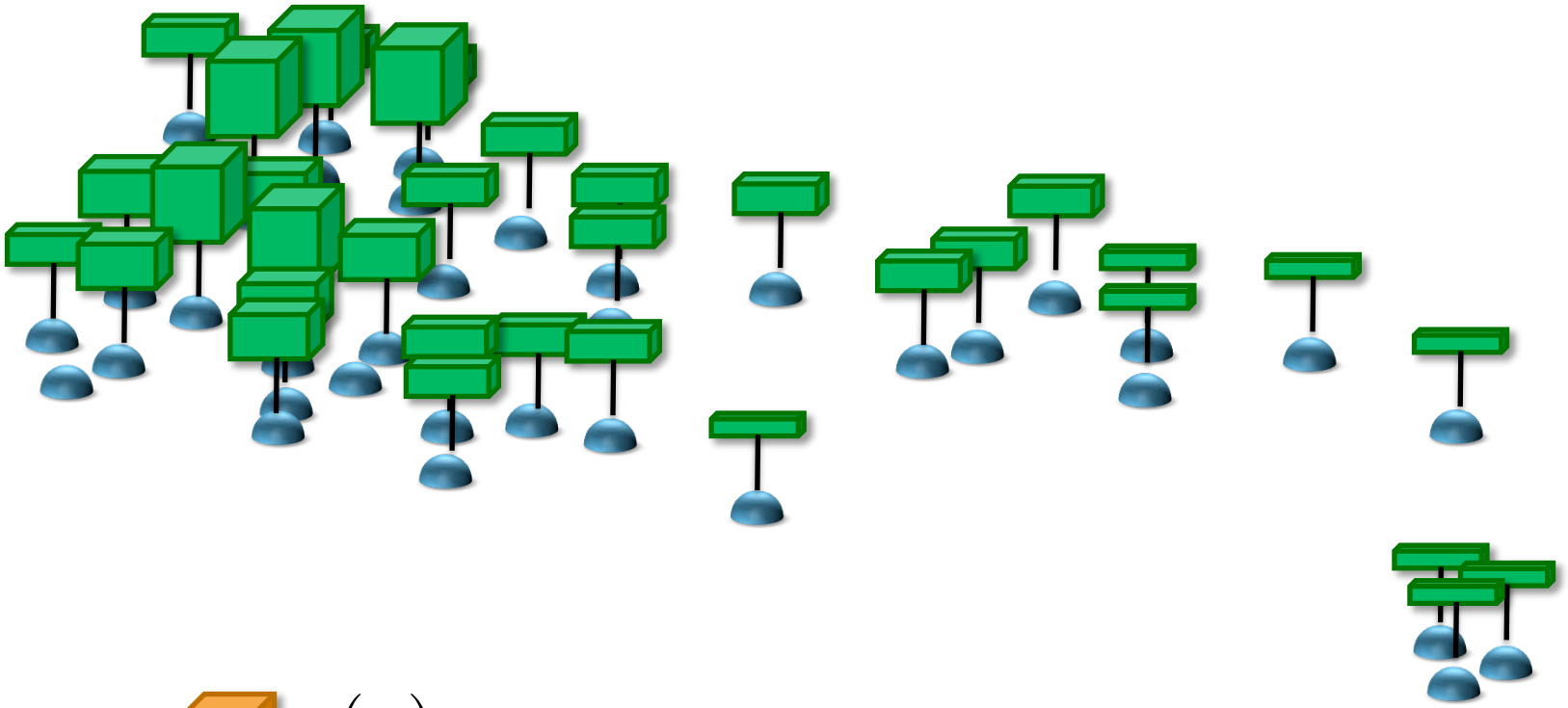


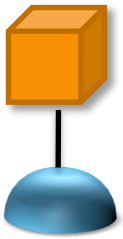
Bias sampling towards small clusters



Sampling distribution

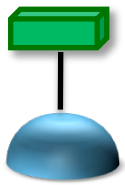
Importance Weights





$q(x)$

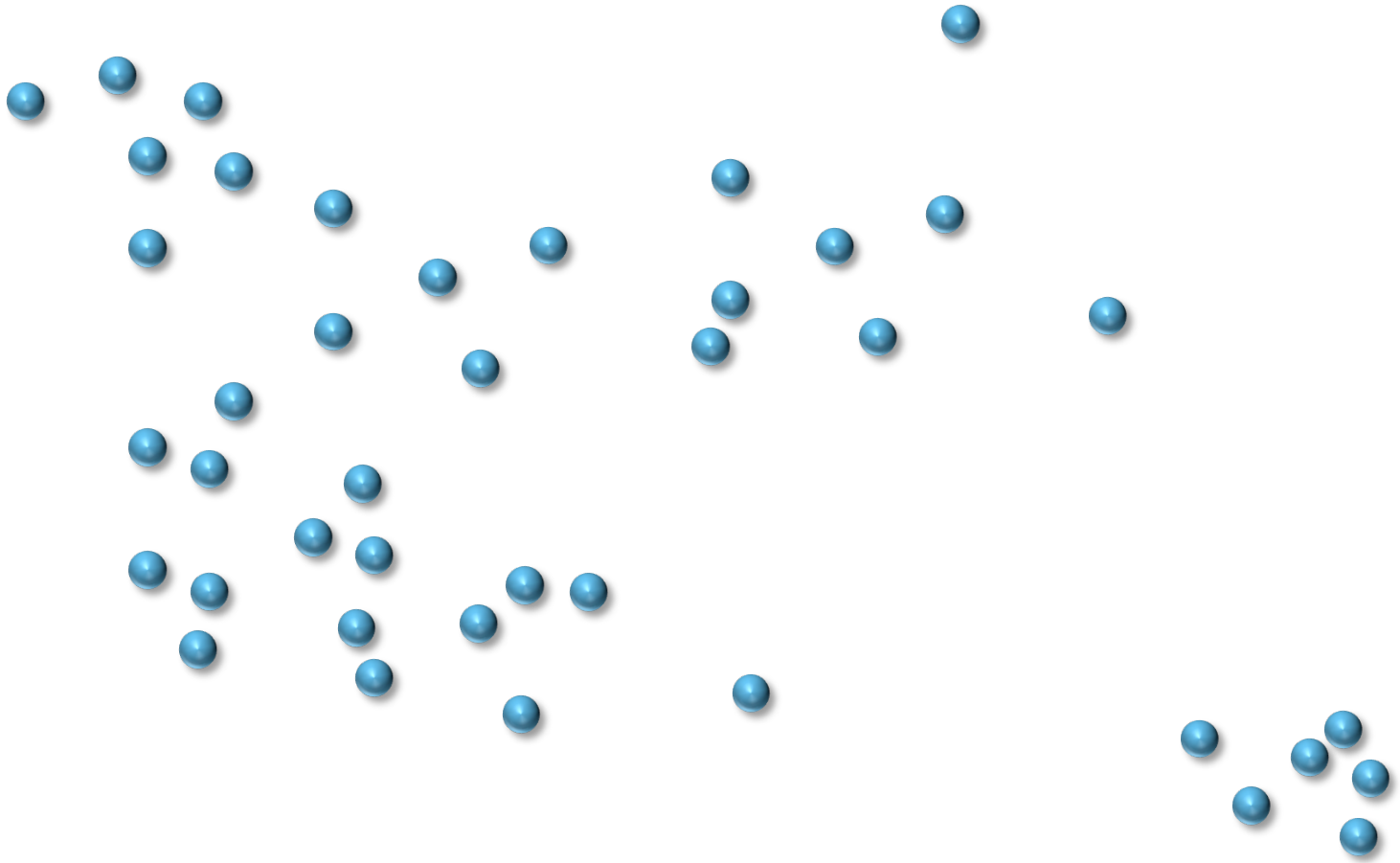
Sampling distribution



$\gamma(x)$

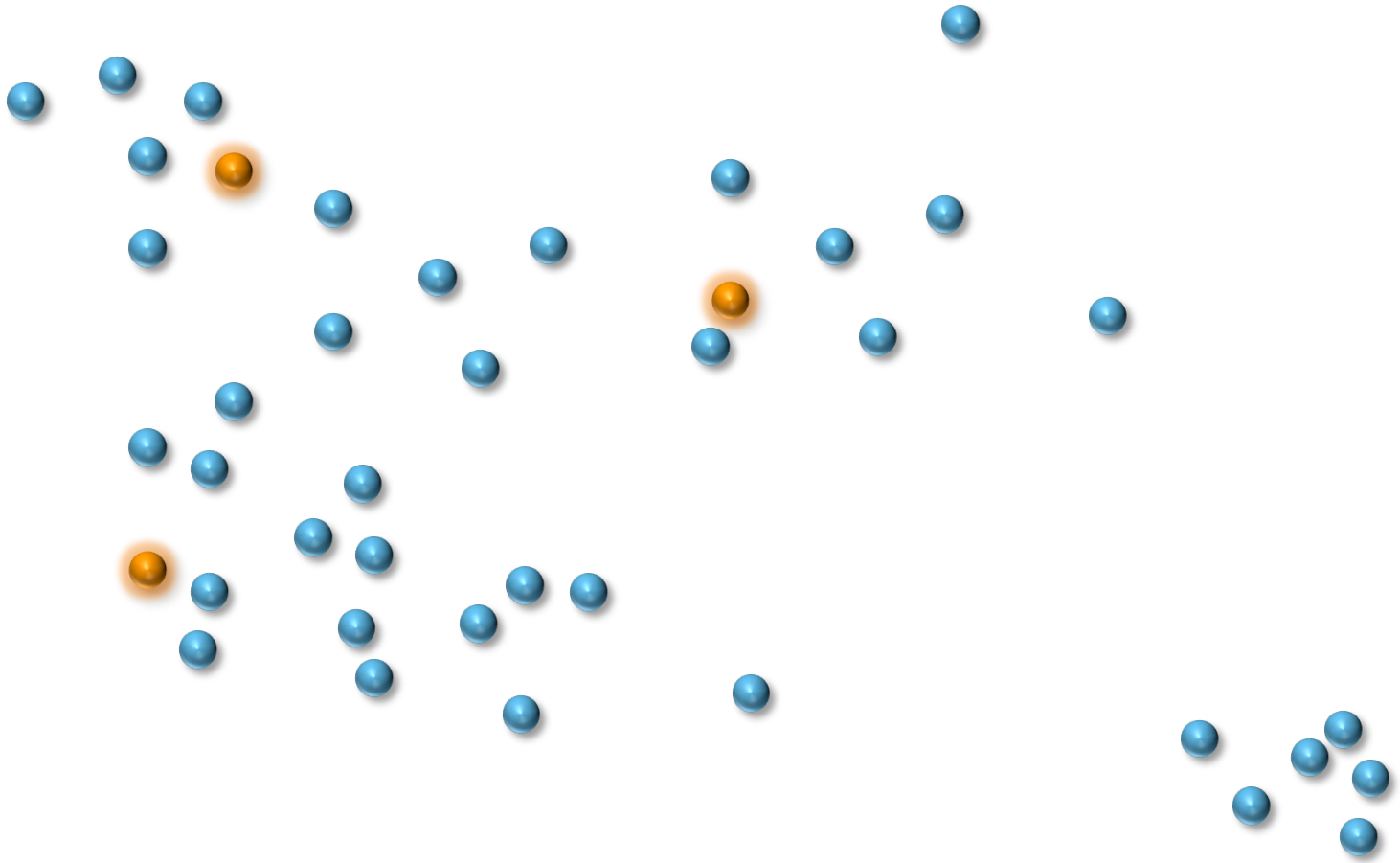
Weights $\gamma \propto \frac{1}{q}$

Creating a Sampling Distribution



Iteratively find representative points

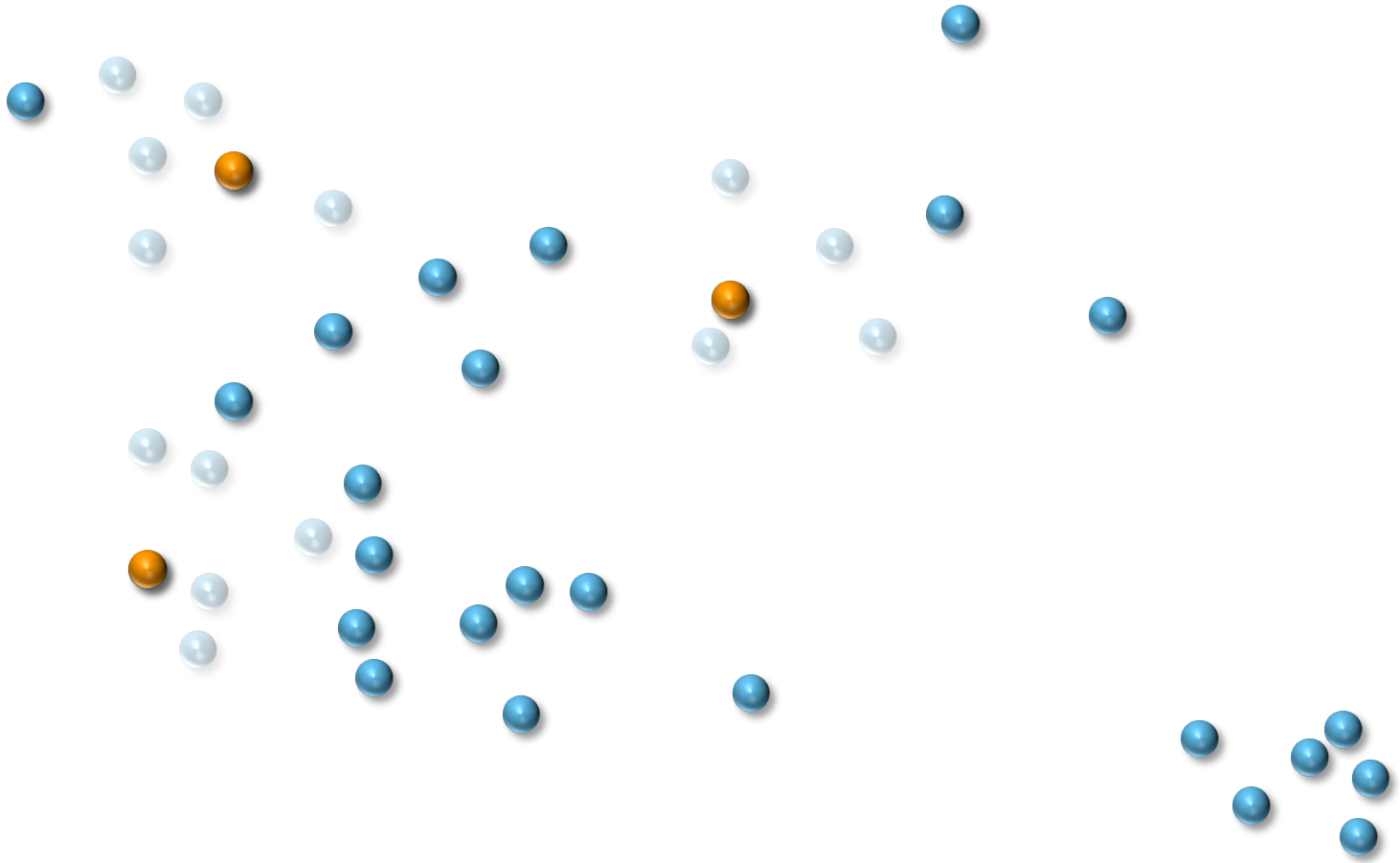
Creating a Sampling Distribution



Iteratively find representative points

- Sample a small set uniformly at random

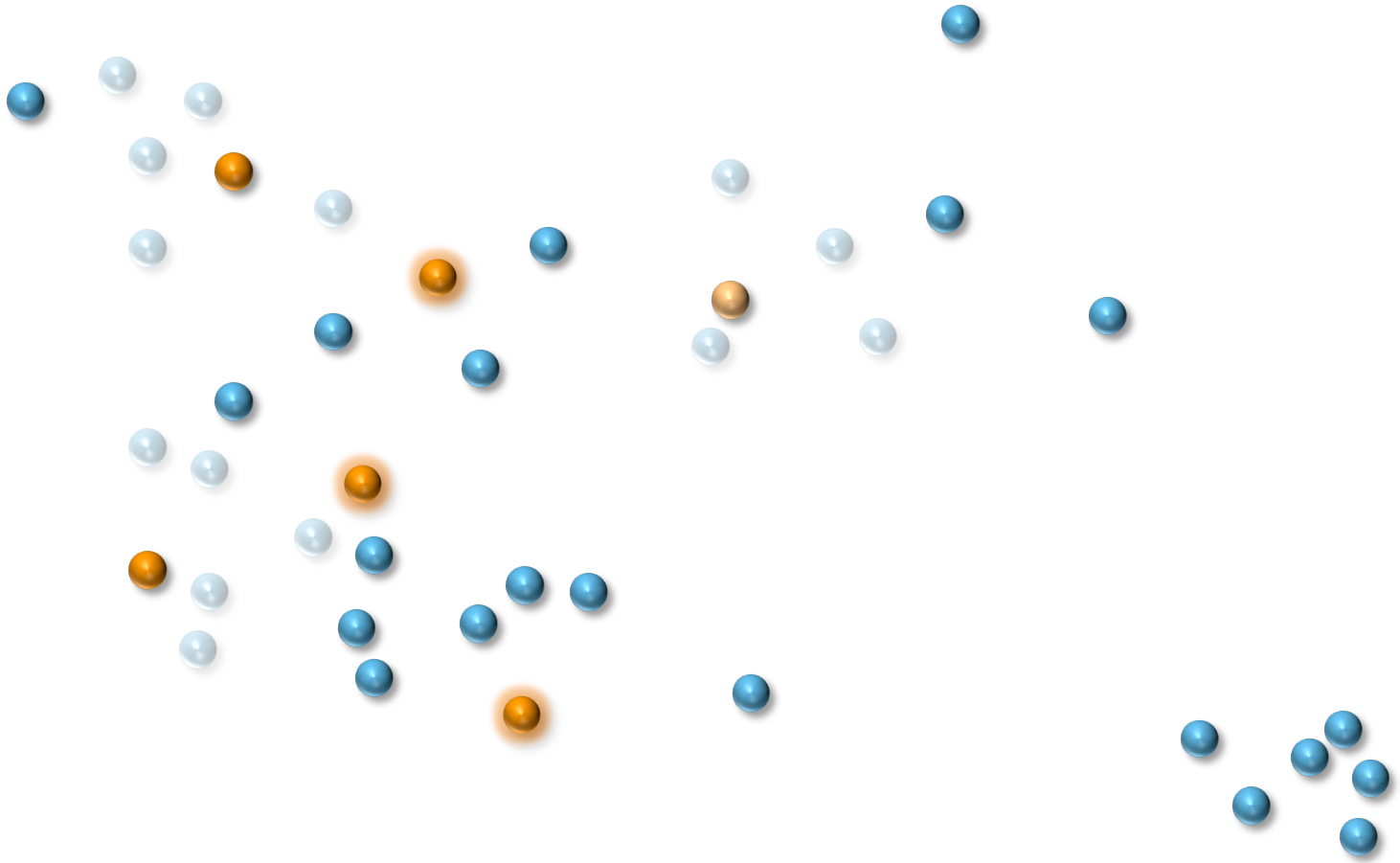
Creating a Sampling Distribution



Iteratively find representative points

- Sample a small set uniformly at random
- Remove half the blue points nearest the samples

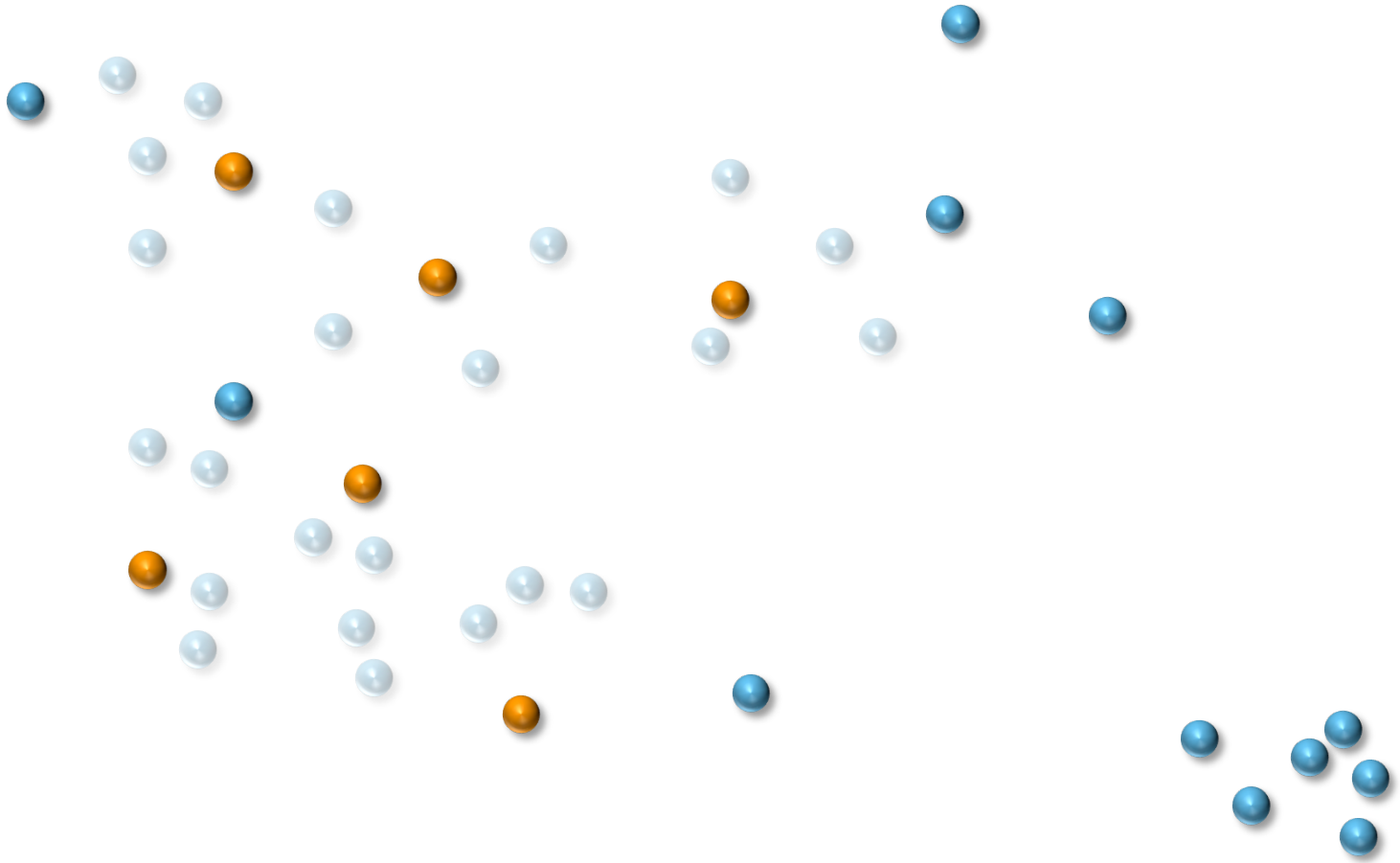
Creating a Sampling Distribution



Iteratively find representative points

- Sample a small set uniformly at random
- Remove half the blue points nearest the samples

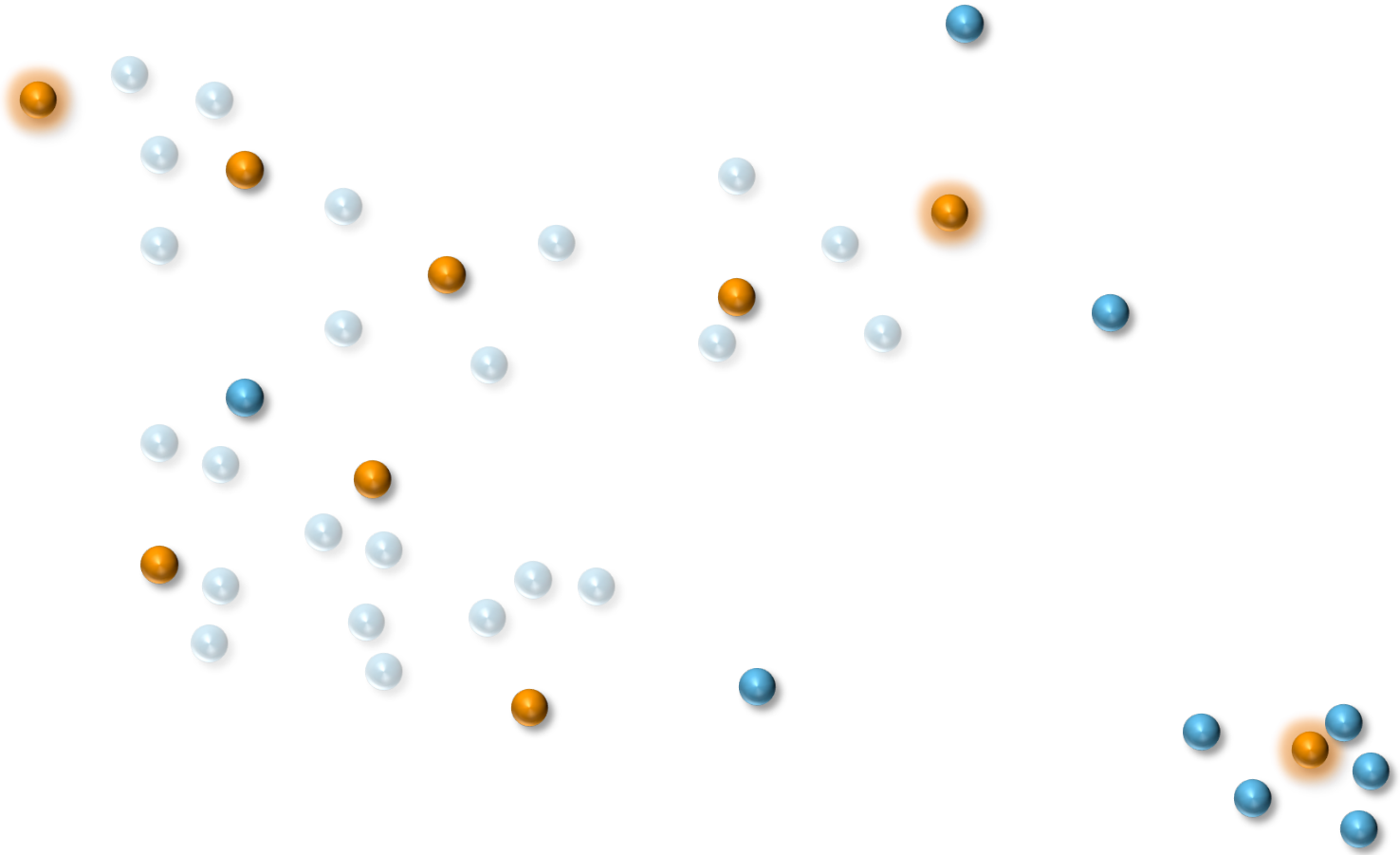
Creating a Sampling Distribution



Iteratively find representative points

- Sample a small set uniformly at random
- Remove half the blue points nearest the samples

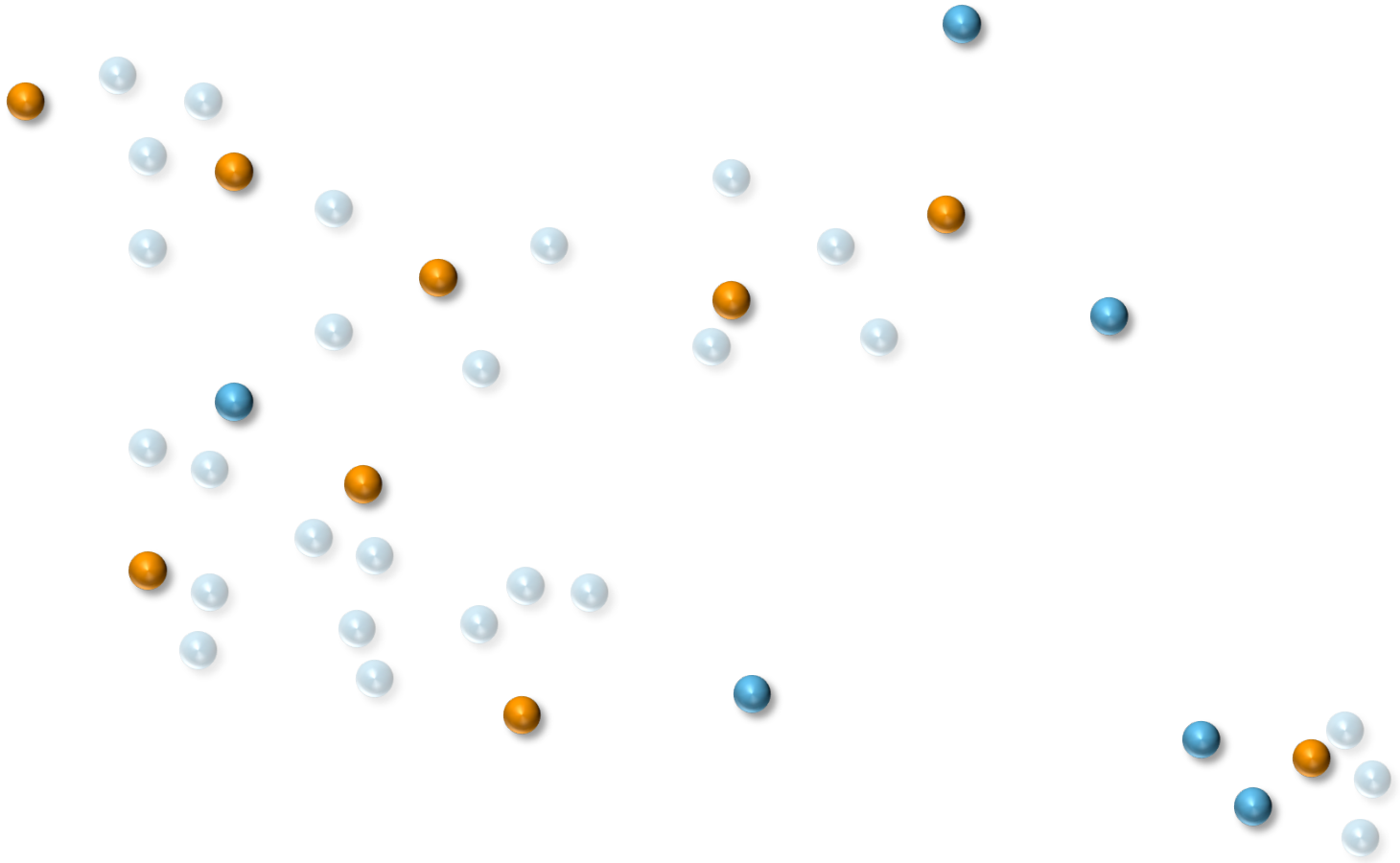
Creating a Sampling Distribution



Iteratively find representative points

- Sample a small set uniformly at random
- Remove half the blue points nearest the samples

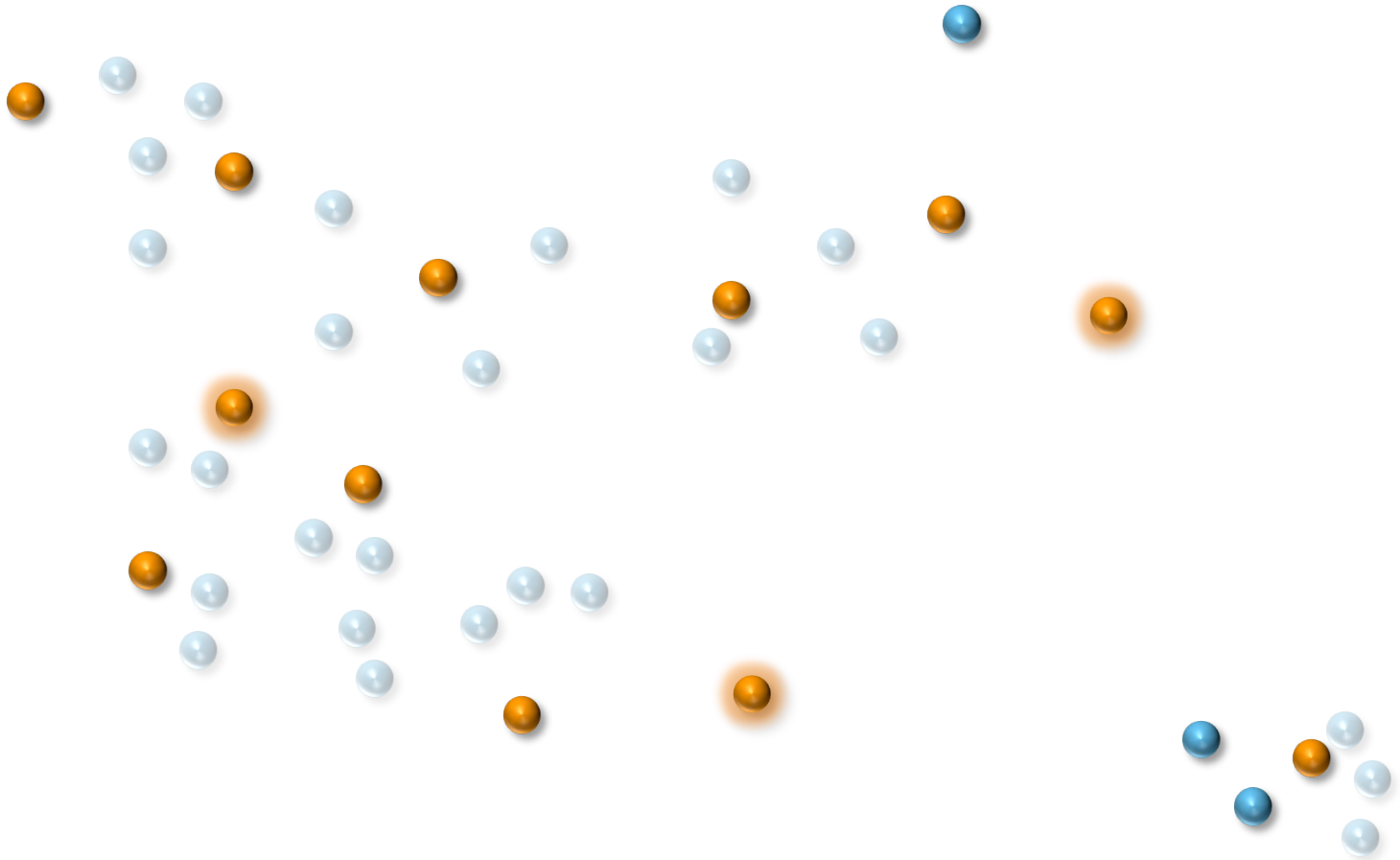
Creating a Sampling Distribution



Iteratively find representative points

- Sample a small set uniformly at random
- Remove half the blue points nearest the samples

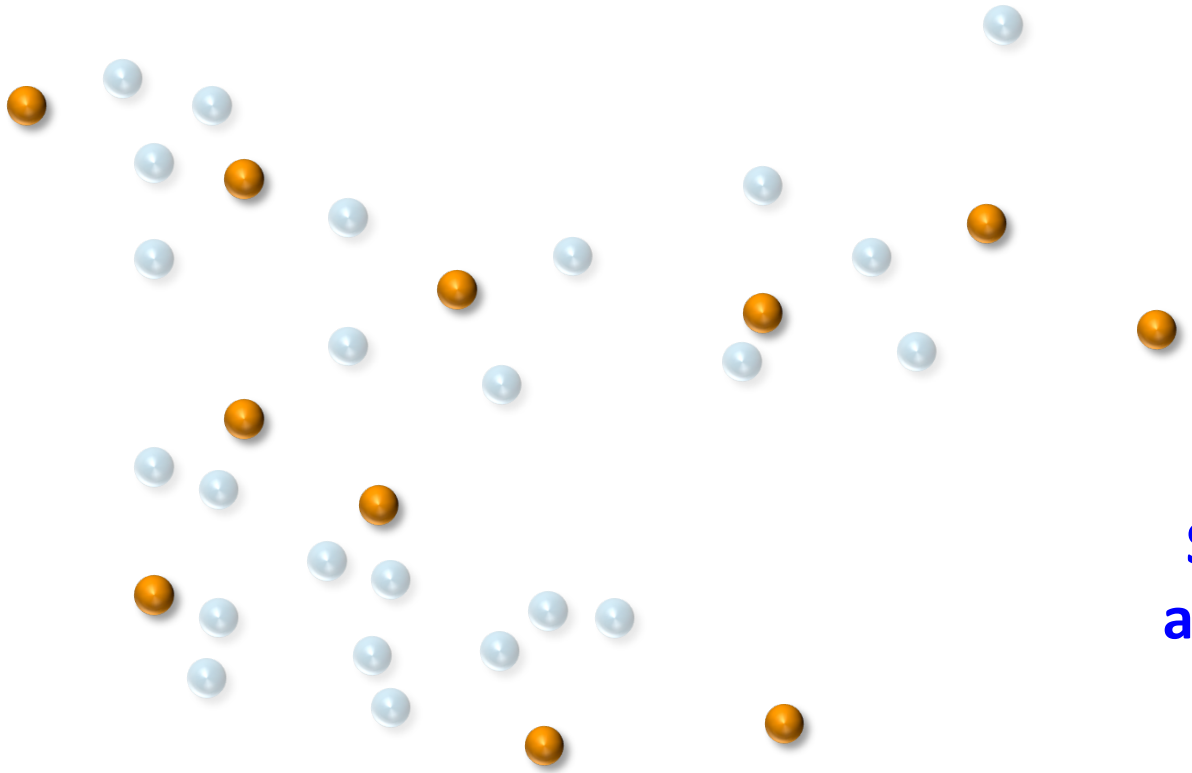
Creating a Sampling Distribution



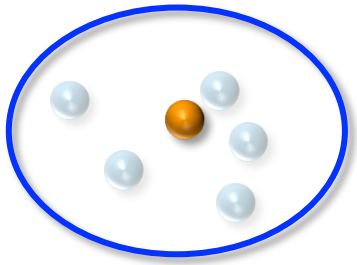
Iteratively find representative points

- Sample a small set uniformly at random
- Remove half the blue points nearest the samples

Creating a Sampling Distribution



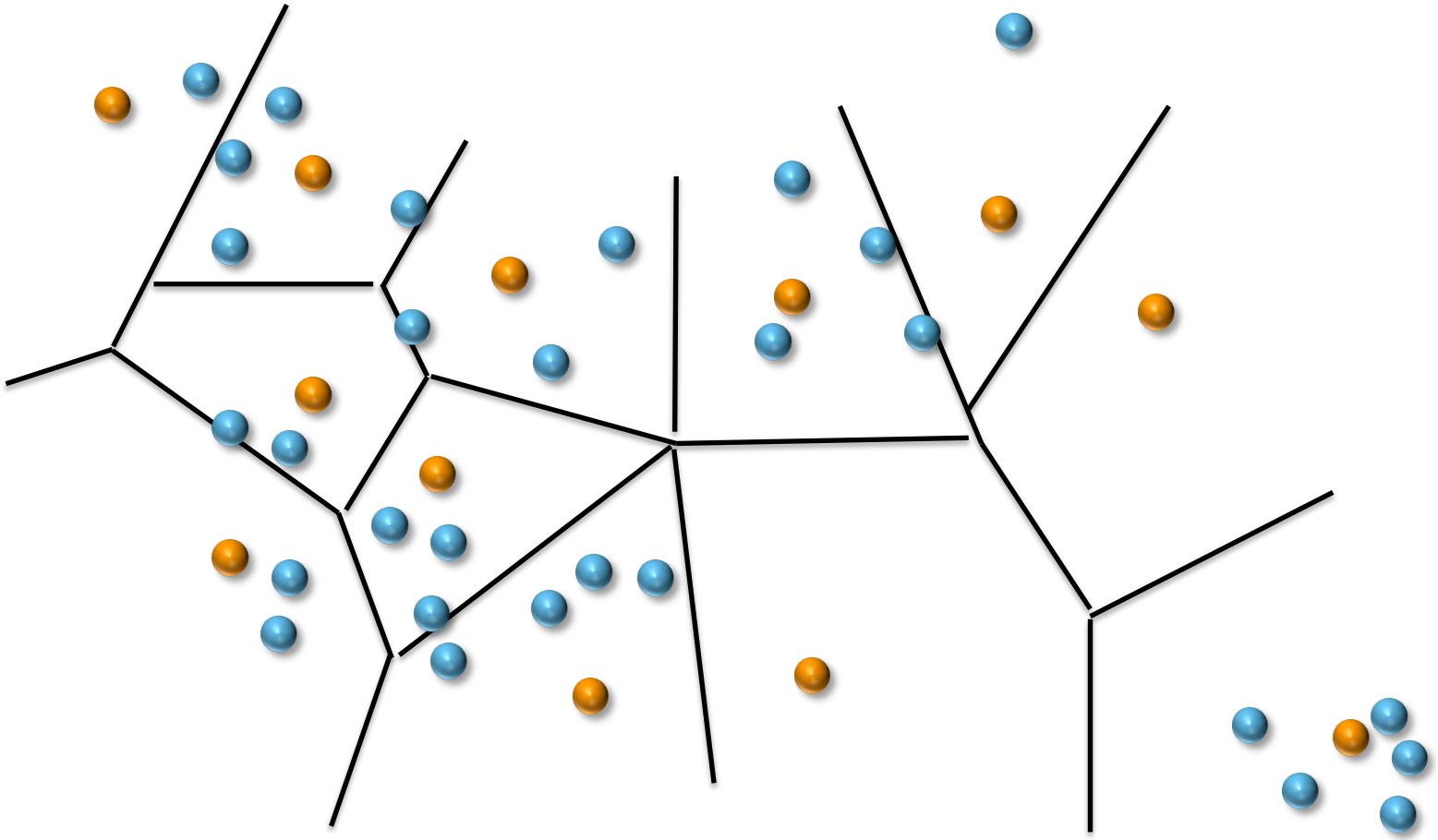
Small clusters are represented



Iteratively find representative points

- Sample a small set uniformly at random
- Remove half the blue points nearest the samples

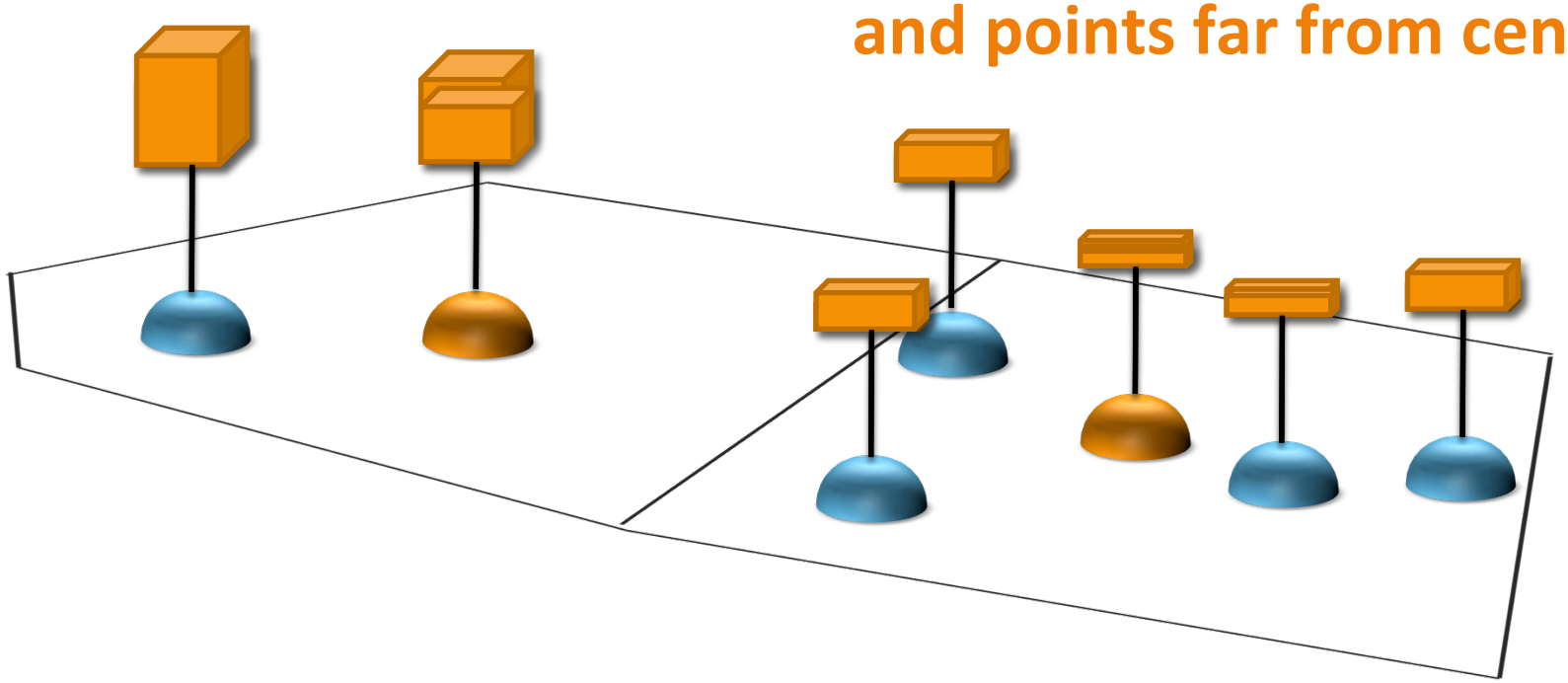
Creating a Sampling Distribution

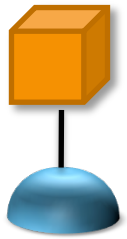


Partition data via a Voronoi diagram centered at points

Creating a Sampling Distribution

Points in sparse cells get more mass and points far from centers



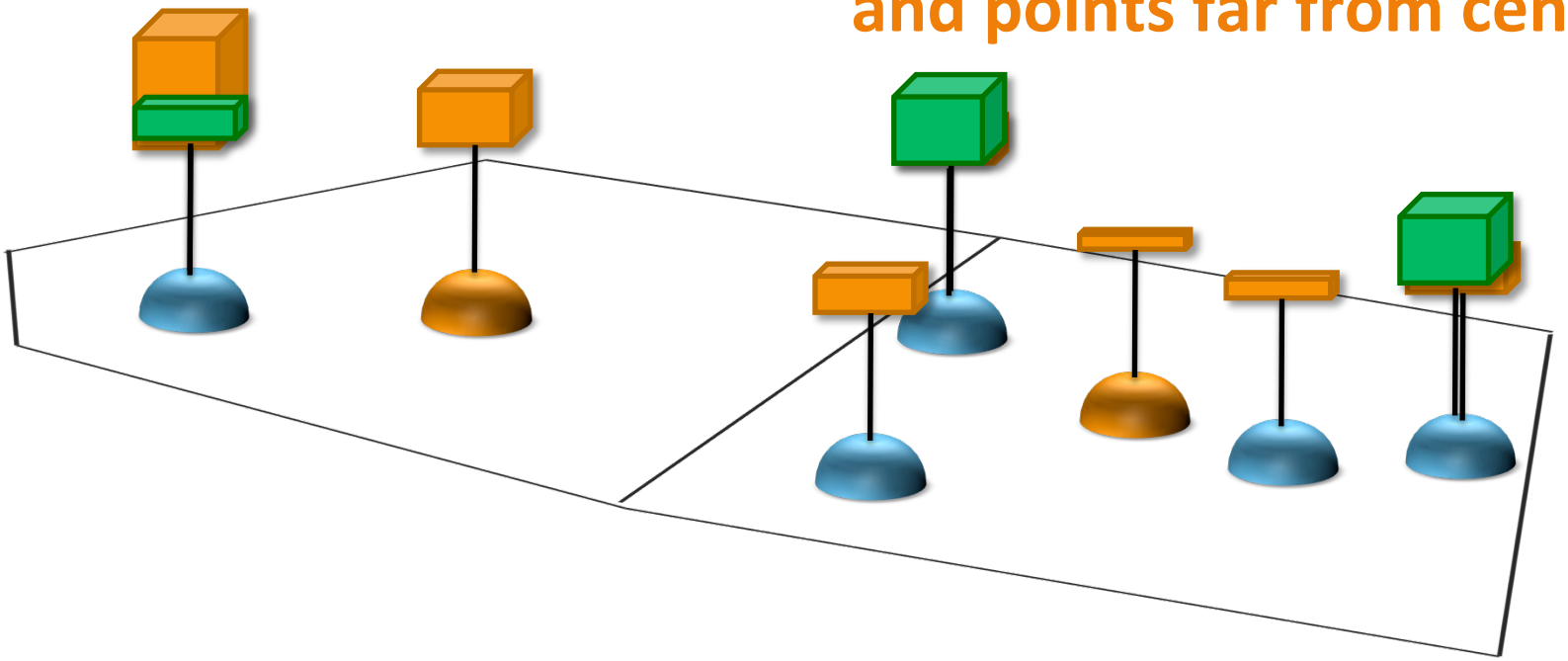


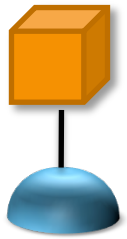
$q(x)$

Sampling distribution q

Importance Weights

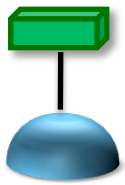
Points in sparse cells get more mass and points far from centers





$q(x)$

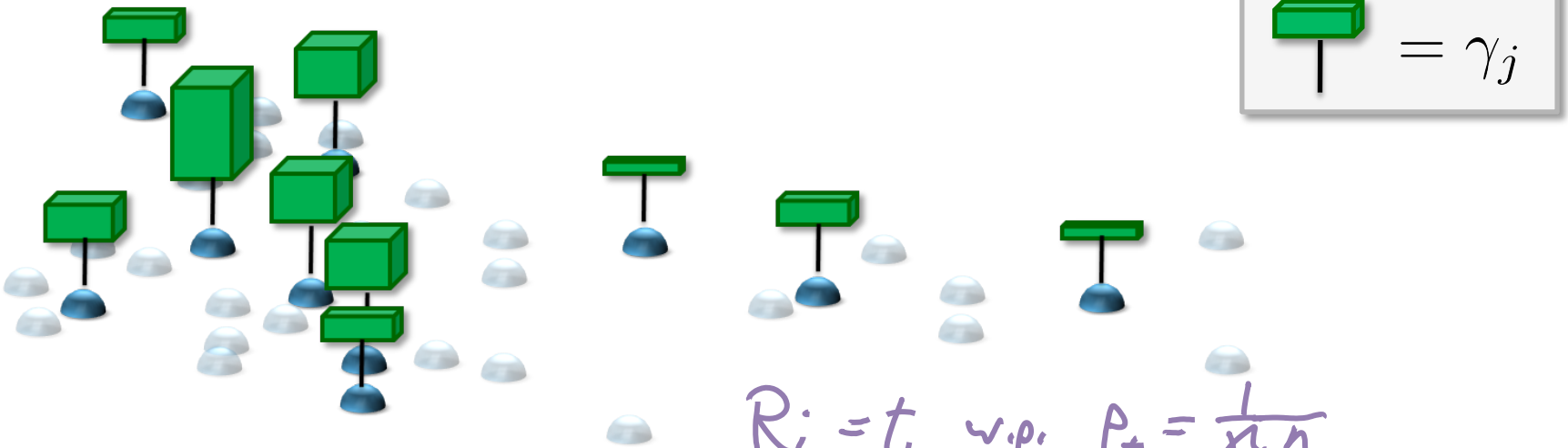
Sampling distribution q



$\gamma(x)$

Weights $\gamma \propto \frac{1}{q}$

Non-uniform sample



$R_i = t$ w.p. $p_t = \frac{1}{\gamma_t n}$

$$L(\mu; C) = \sum_{i=1}^{n} \gamma_{x_i} \min_j \| x_{R_i} - \mu_j \|_2^2$$

$$\mathbb{E}[L(\mu; C)] = \sum_{i=1}^{n} \sum_{t=1}^{N} \left(p_t \frac{1}{n p_t} \right) \min_j \| x_t - \mu_j \|_2^2$$

$$= \sum_{t=1}^{N} \min_j \| x_t - \mu_j \|_2^2 = L(\mu; D)$$

Coresets via Adaptive Sampling

$B \leftarrow \emptyset \quad D' \leftarrow D$

while $D' \neq \emptyset$

$S \leftarrow$ uniformly sample $10dk \ln(\frac{1}{\epsilon})$ points from D'

Remove $\frac{|D'|}{2}$ points nearest to S from D'

$B \leftarrow B \cup S$

Partition D into Voronoi cells D_b centered at $b \in B$

$$q(x) \propto \lceil \frac{5}{|D_b|} + \frac{\mathrm{dist}(x,B)^2}{\sum_{x'} \mathrm{dist}(x',B)^2} \rceil, \qquad \gamma(x) = \frac{1}{|C|q(x)}$$

$C \leftarrow$ sample $10\lceil dk \log^2 n \log(\frac{1}{\delta})/\epsilon^2 \rceil$ from D via q

C is (k,ε)-coreset of size polynomial in d,k,log n, 1/ε, 1/δ

with prob ≥ 1−δ

Can do better: Coresets for k-means

- **Theorem** [Har-Peled and Kushal, '05]
  One can find efficiently a (k,ε)-coreset for k-means of size

$$\mathcal{O}\left(k^3/\varepsilon^{d+1}\right)$$

- **Theorem** [Feldman et al '07]
  One can efficiently find a weak (k,ε)-coreset of size

$$\mathcal{O}\left(\mathrm{poly}(k, 1/\varepsilon)\right)$$

- Allows PTAS for k-means!

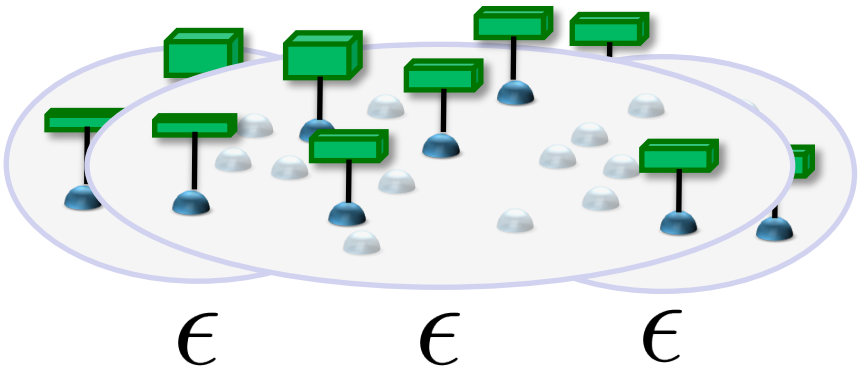
Coresets exist for

- K-means, K-median
- K-line means / median
- PageRank
- SVMs
- Diameter of a point set
- Matrix low-rank approximation
- …

Composition of Coresets

[c.f. Har-Peled, Mazumdar 04]

Merge The union of two (k, ϵ)-coresets is a (k, ϵ)-coreset.

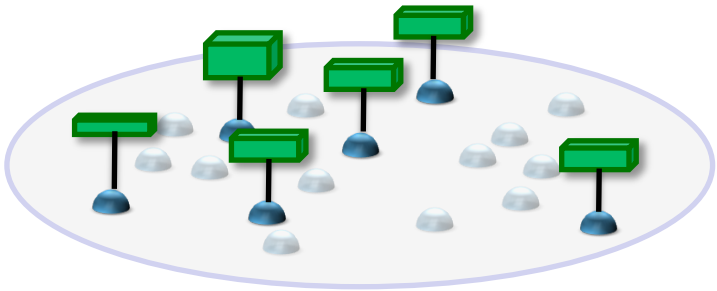


Composition of Coresets

[Har-Peled, Mazumdar 04]

Merge The union of two (k, ϵ)-coresets is a (k, ϵ)-coreset.

Compress A (k, δ)-coreset of a (k, ϵ)-coreset is a $(k, \epsilon + \delta + \epsilon\delta)$-coreset



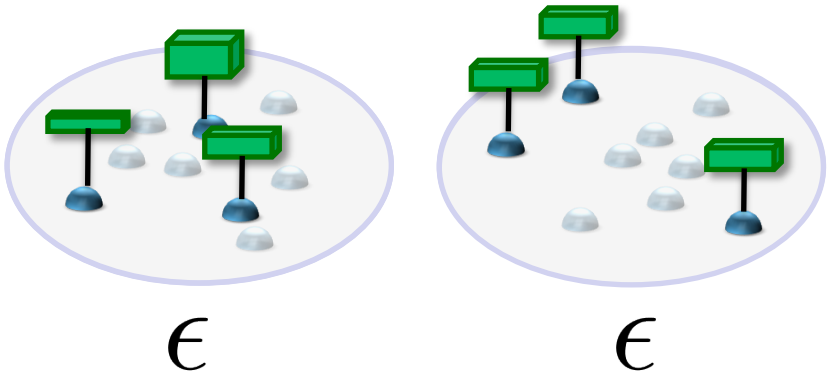
$$\epsilon + \delta + \epsilon\delta$$

Coresets on Streams

[Har-Peled, Mazumdar 04]

Merge The union of two (k, ϵ)-coresets is a (k, ϵ)-coreset.

Compress A (k, δ)-coreset of a (k, ϵ)-coreset is a $(k, \epsilon + \delta + \epsilon\delta)$-coreset

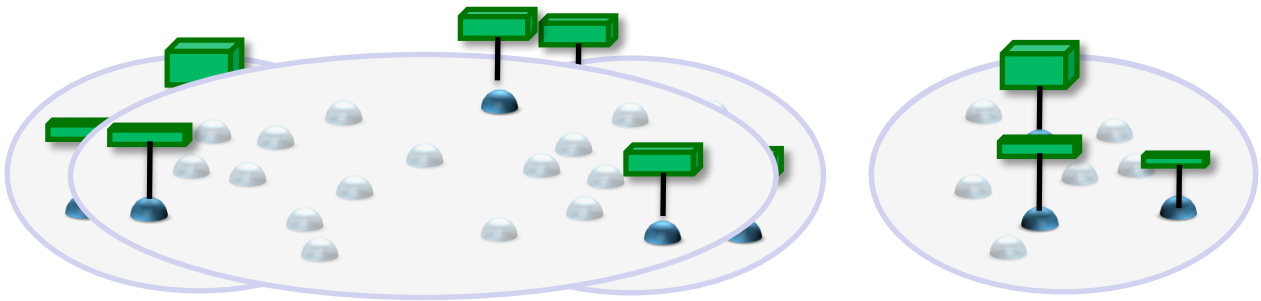


Coresets on Streams

[Har-Peled, Mazumdar 04]

Merge The union of two (k, ϵ)-coresets is a (k, ϵ)-coreset.

Compress A (k, δ)-coreset of a (k, ϵ)-coreset is a $(k, \epsilon + \delta + \epsilon\delta)$-coreset



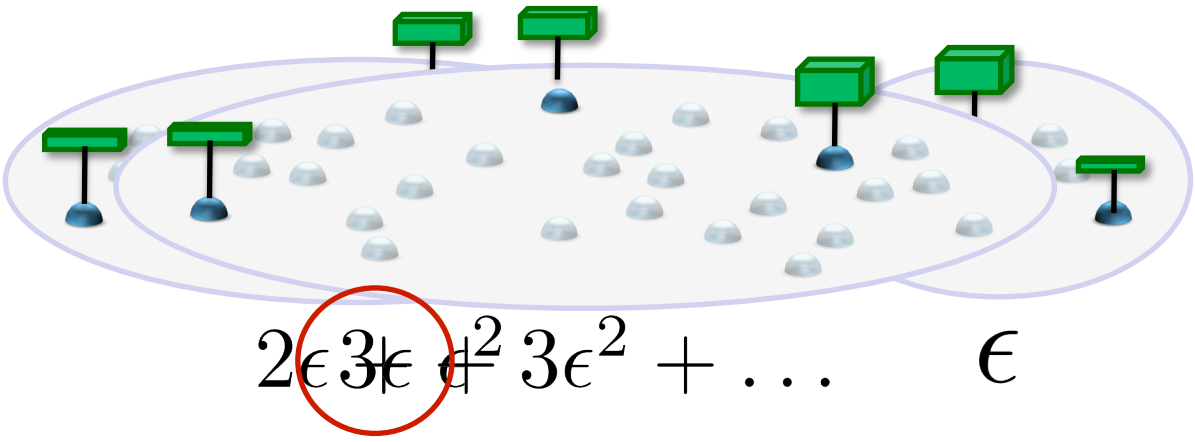
$\epsilon \quad 2\epsilon + \epsilon^2 \quad \epsilon \qquad\qquad \epsilon$

Coresets on Streams

[Har-Peled, Mazumdar 04]

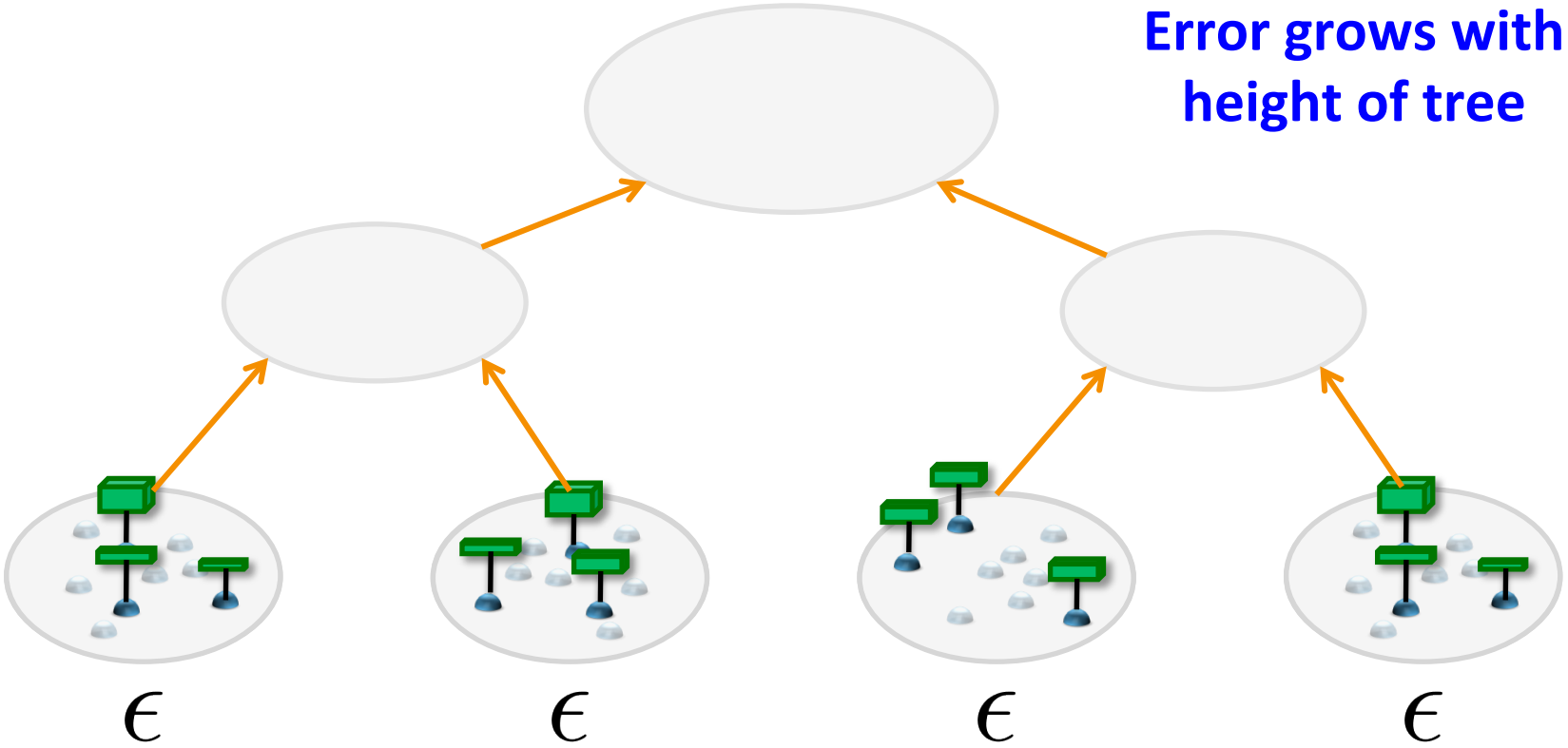
Merge The union of two (k, ϵ)-coresets is a (k, ϵ)-coreset.

Compress A (k, δ)-coreset of a (k, ϵ)-coreset is a $(k, \epsilon + \delta + \epsilon\delta)$-coreset

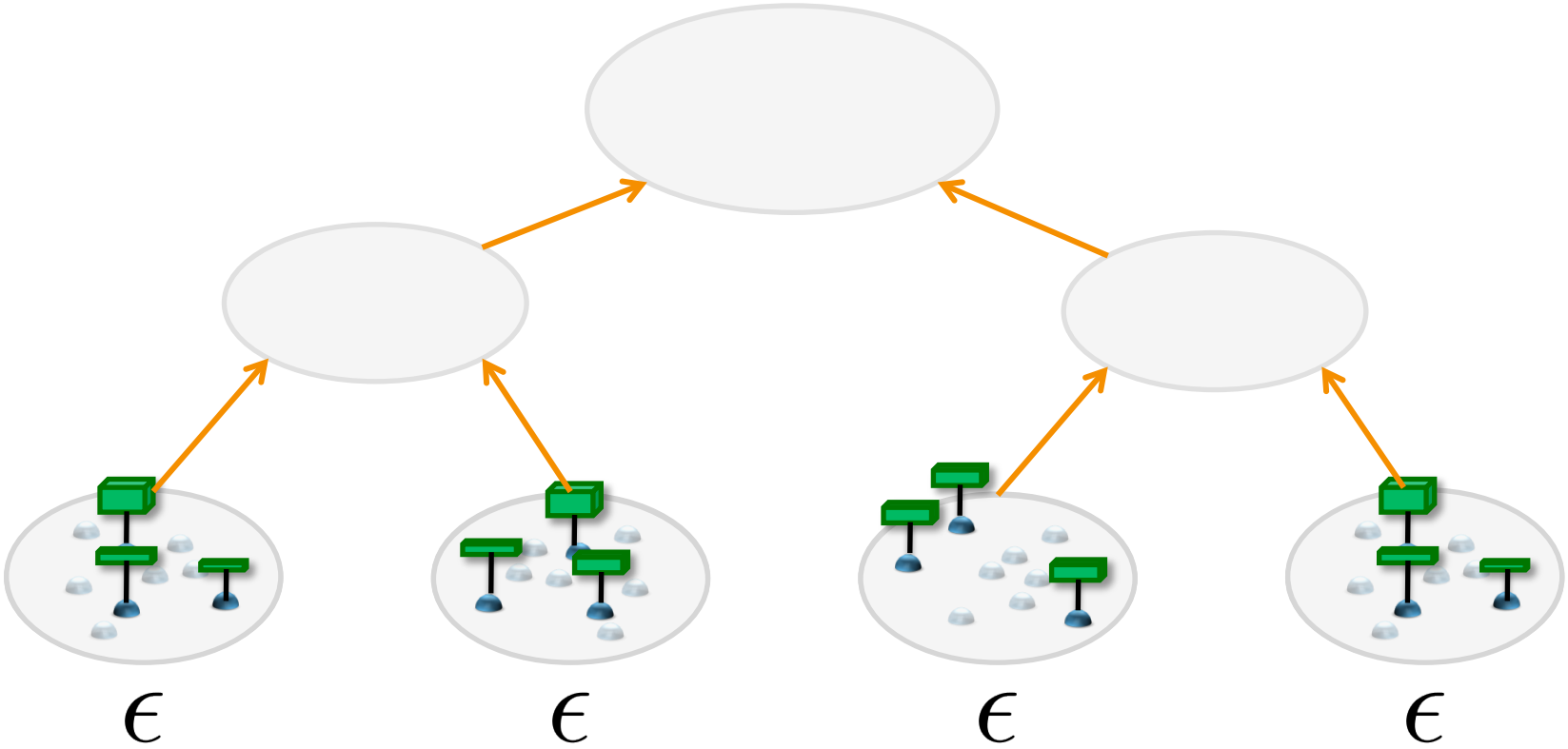


Error grows linearly with number of compressions

Coresets on Streams



Coresets in Parallel



k-means clustering with coresets

- Given data set D, desired number of clusters k, precision ε
- Construct (k, ε) - coreset C
  - E.g., in parallel using MapReduce
- Solve k-means on coreset
  - If coreset small, can even do exhaustive search!
  - In practice, run k-means with many restarts
- Resulting solution will be (1+ε)-optimal for D
- ➔ Provably near-optimal solution!

Summary so far

- Clustering is a central problem in unsupervised learning
- Two main classes of approaches
  - Hierarchical (difficult to scale)
  - Assignment based
- Discussed k-means algorithm
  - Widely used clustering algorithm
  - "Non-linear" versions available
  - Can scale to large data sets using online optimization and coreset constructions

Acknowledgments

- The slides are partly based on material By Chris Bishop, Andrew Moore and Danny Feldman