

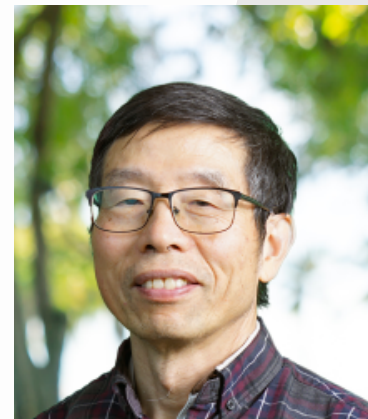
Byzantine-tolerant distributed learning of finite mixture models



Qiong Zhang
RUC

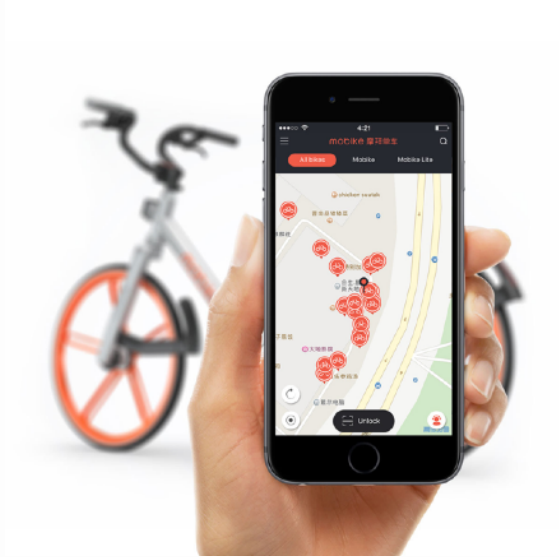


Yan Shuo Tan
NUS



Jiahua Chen
UBC

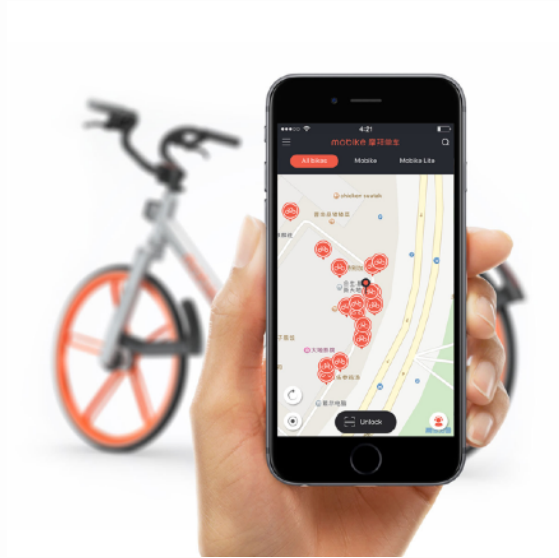
Bike share



Example credit: Trevor Campbell

Bike share

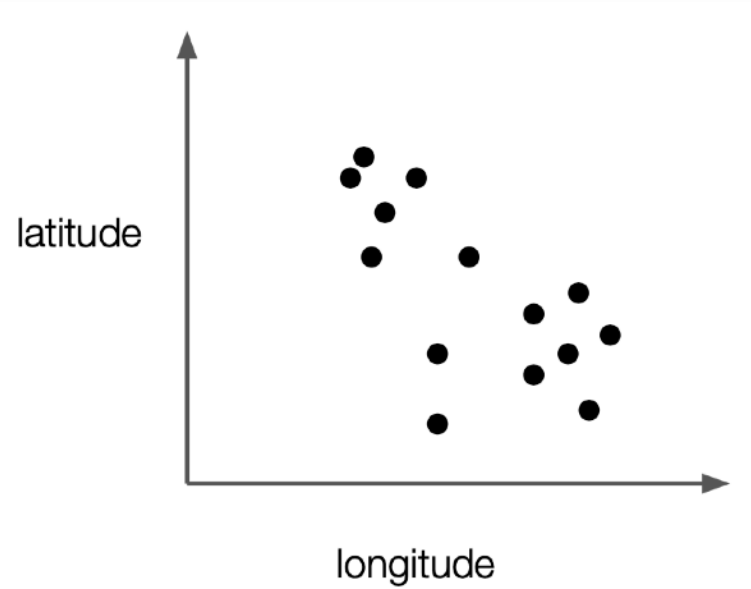
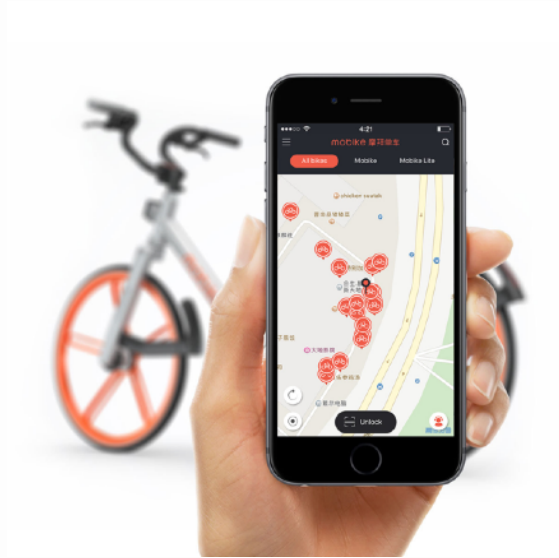
Need to rebalance bikes-where do customers leave them?



Example credit: Trevor Campbell

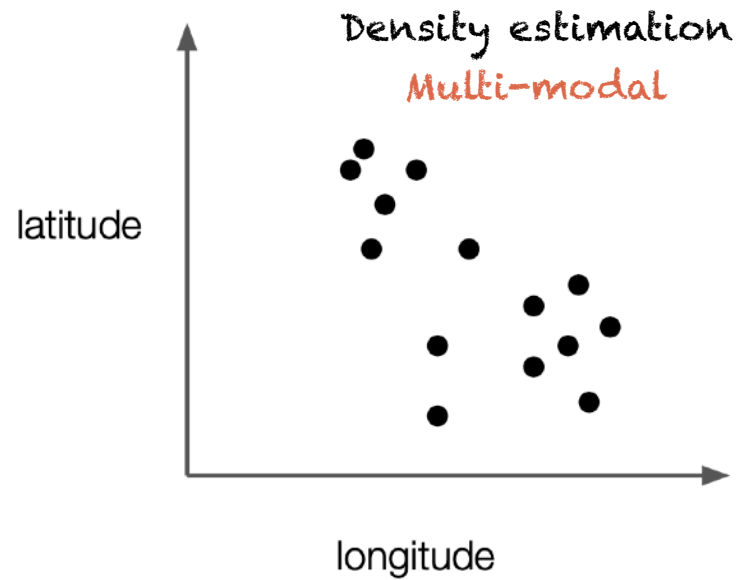
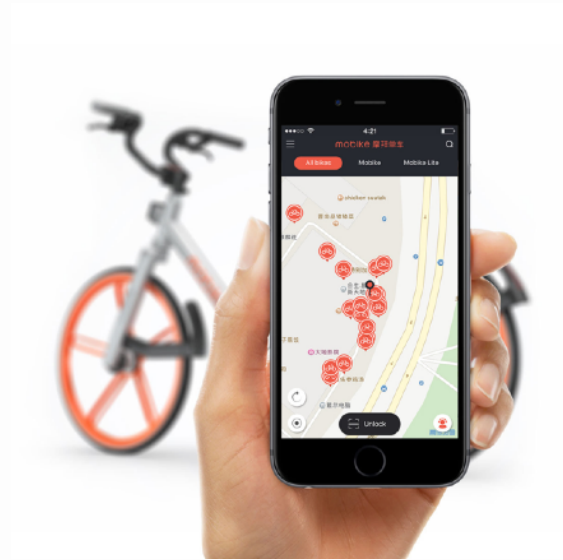
Bike share

Need to rebalance bikes-where do customers leave them?



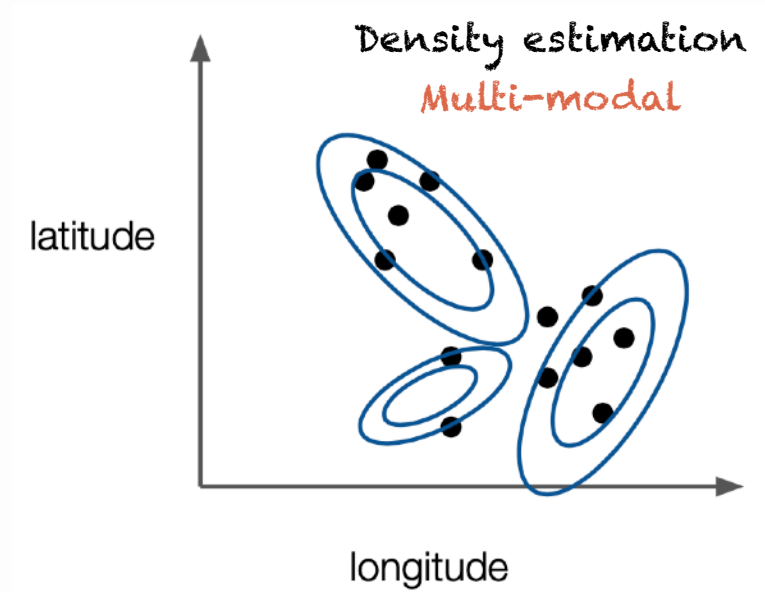
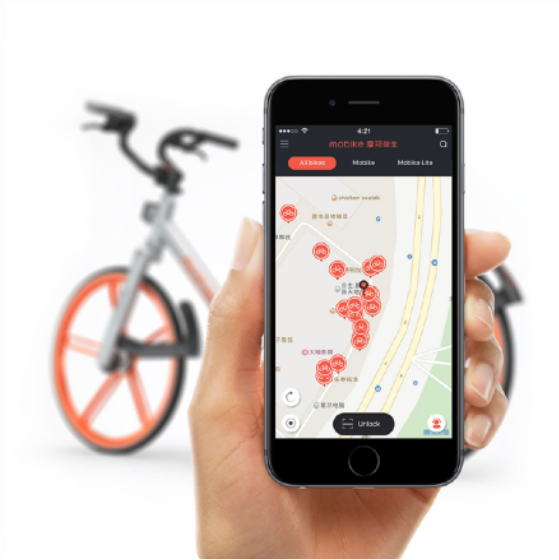
Bike share

Need to rebalance bikes-where do customers leave them?



Bike share

Need to rebalance bikes-where do customers leave them?



Example credit: Trevor Campbell

Human genetic clustering



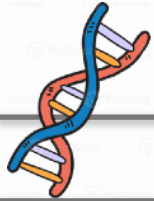
Human genetic clustering













Do you want to know your ancestry from your gene?

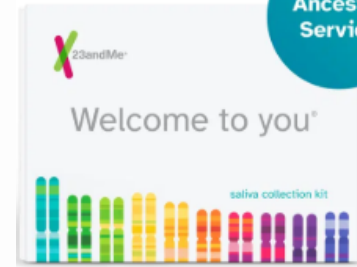


Human genetic clustering

Do you want to know your ancestry from your gene?



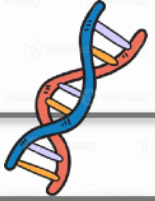
	SNP1	SNP2	SNPN
			
			
			















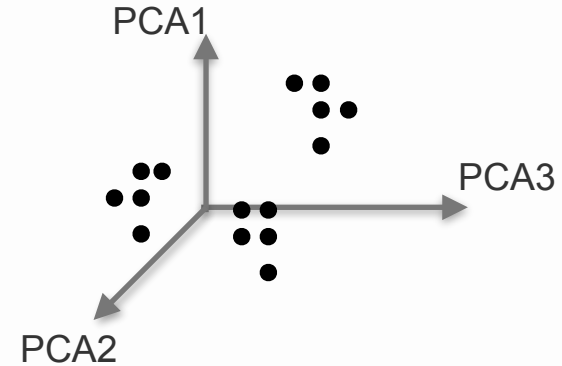
Ancestry
Service

Human genetic clustering

Do you want to know your ancestry from your gene?

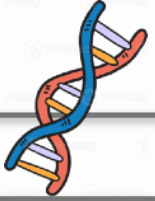














	SNP1	SNP2	SNPN
			
			
			

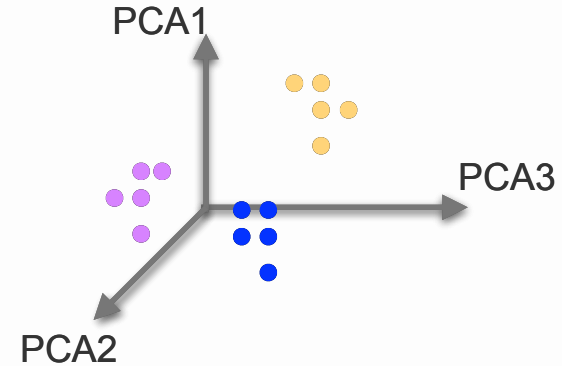


Human genetic clustering

Do you want to know your ancestry from your gene?



	SNP1	SNP2	SNPN
			
			
			



Finite mixture model

A family of distributions

Finite mixture model

A family of distributions

- Let $\mathcal{F} = \{f(x; \theta) : \theta \in \Theta\}$ be a parametric distribution family
- The finite mixture model of \mathcal{F} with order K has its density function:

$$f_G(x) := \int f(x; \theta) dG(\theta) = \sum_{k=1}^K w_k f(x; \theta_k)$$

Finite mixture model

A family of distributions

- Let $\mathcal{F} = \{f(x; \theta) : \theta \in \Theta\}$ be a parametric distribution family
- The finite mixture model of \mathcal{F} with order K has its density function:

$$f_{\boxed{G}}(x) := \int f(x; \theta) dG(\theta) = \sum_{k=1}^K w_k f(x; \theta_k)$$

Mixing distribution

$$G = \sum_k w_k \delta_{\theta_k}$$

Finite mixture model

A family of distributions

- Let $\mathcal{F} = \{f(x; \theta) : \theta \in \Theta\}$ be a parametric distribution family
- The finite mixture model of \mathcal{F} with order K has its density function:

$$f_G(x) := \int f(x; \theta) dG(\theta) = \sum_{k=1}^K w_k f(x; \theta_k)$$

Mixing distribution Subpopulation parameter

$$G = \sum_k w_k \delta_{\theta_k}$$

Mixing weight

Finite mixture model

A family of distributions

- Let $\mathcal{F} = \{f(x; \theta) : \theta \in \Theta\}$ be a parametric distribution family
- The finite mixture model of \mathcal{F} with order K has its density function:

$$f_G(x) := \int f(x; \theta) dG(\theta) = \sum_{k=1}^K w_k f(x; \theta_k)$$

Mixing distribution

Order (known)

Subpopulation parameter

$$G = \sum_k w_k \delta_{\theta_k}$$

Mixing weight

Finite mixture model

A family of distributions

- Let $\mathcal{F} = \{f(x; \theta) : \theta \in \Theta\}$ be a parametric distribution family
- The finite mixture model of \mathcal{F} with order K has its density function:

$$f_{\boxed{G}}(x) := \int f(x; \theta) dG(\theta) = \sum_{k=1}^{\boxed{K}} w_k f(x; \boxed{\theta_k}) \qquad G = \sum_k \boxed{w_k} \delta_{\theta_k}$$

Mixing distribution Order (known) Subpopulation parameter Mixing weight

- e.g., finite Gaussian mixture

$$\mathcal{F} = \{\phi(x; \mu, \Sigma) = |2\pi\Sigma|^{-1/2} \exp\{- (x - \mu)^\top \Sigma^{-1} (x - \mu)/2\} : \mu \in \mathbb{R}^d, \Sigma > 0\}$$

Finite mixture model

A family of distributions

- Let $\mathcal{F} = \{f(x; \theta) : \theta \in \Theta\}$ be a parametric distribution family
- The finite mixture model of \mathcal{F} with order K has its density function:

$$f_{\boxed{G}}(x) := \int f(x; \theta) dG(\theta) = \sum_{k=1}^{\boxed{K}} w_k f(x; \boxed{\theta_k}) \qquad G = \sum_k \boxed{w_k} \delta_{\theta_k}$$

Mixing distribution Subpopulation parameter Mixing weight

- e.g., finite Gaussian mixture

$$\mathcal{F} = \{\phi(x; \mu, \Sigma) = |2\pi\Sigma|^{-1/2} \exp\{- (x - \mu)^\top \Sigma^{-1} (x - \mu)/2\} : \mu \in \mathbb{R}^d, \Sigma > 0\}$$

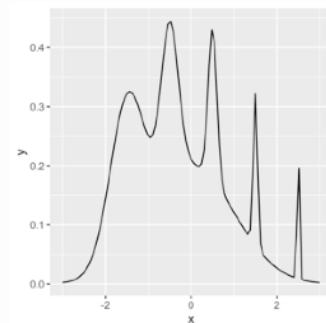
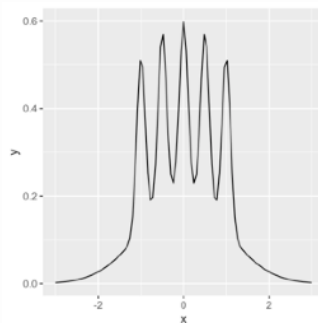
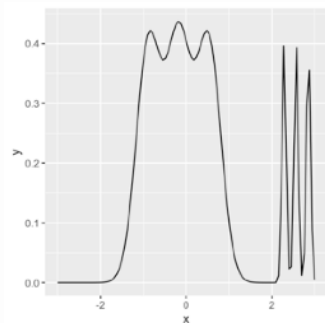
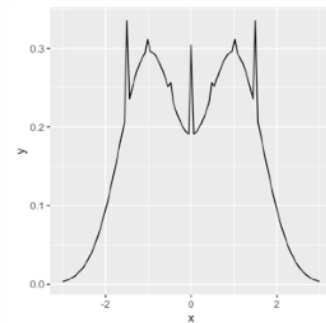
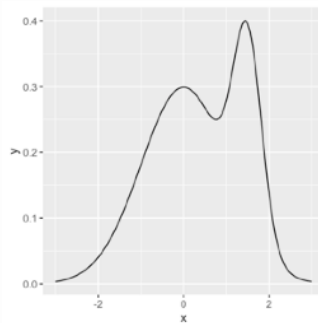
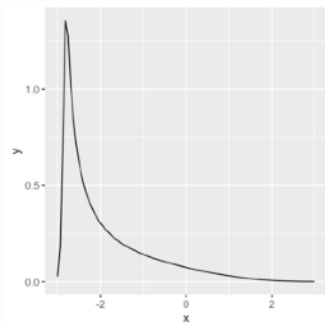
- Parameter space

$$\mathbb{G}_K = \left\{ G = \sum_{k=1}^K w_k \delta_{\theta_k} : \theta_k \in \Theta, w_k \in (0,1), \sum_k w_k = 1 \right\}$$

Finite mixture for density estimation

Finite mixture for density estimation

Finite mixture can be used to approximate density functions with various shapes



Finite mixture for model-based clustering

Finite mixture for model-based clustering

- Latent variable representation (Z not observed)

$$\begin{cases} X|Z = k \sim f(x; \theta_k), \\ P(Z = k) = w_k, k \in [K] = 1, \dots, K \end{cases}$$

Finite mixture for model-based clustering

- Latent variable representation (Z not observed)

$$\begin{cases} X|Z = k \sim f(x; \theta_k), \\ P(Z = k) = w_k, k \in [K] = 1, \dots, K \end{cases}$$

- Marginal of X is a mixture of order K

Finite mixture for model-based clustering

- Latent variable representation (Z not observed)

$$\begin{cases} X|Z = k \sim f(x; \theta_k), \\ P(Z = k) = w_k, k \in [K] = 1, \dots, K \end{cases}$$

- Marginal of X is a mixture of order K
- Posterior distribution of the latent variable

$$P(Z = k | X = x) \propto w_k f(x; \theta_k)$$

Finite mixture for model-based clustering

- Latent variable representation (Z not observed)

$$\begin{cases} X | Z = k \sim f(x; \theta_k), \\ P(Z = k) = w_k, k \in [K] = 1, \dots, K \end{cases}$$

- Marginal of X is a mixture of order K
- Posterior distribution of the latent variable

$$P(Z = k | X = x) \propto w_k f(x; \theta_k)$$

- Clustering(maximize posterior)

$$\kappa(x; G) = \operatorname{argmax}_{j \in [K]} w_j f(x; \theta_j)$$

Finite mixture for model-based clustering

- Latent variable representation (Z not observed)

$$\begin{cases} X | Z = k \sim f(x; \theta_k), \\ P(Z = k) = w_k, k \in [K] = 1, \dots, K \end{cases}$$

- Marginal of X is a mixture of order K
- Posterior distribution of the latent variable

$$P(Z = k | X = x) \propto w_k f(x; \theta_k)$$

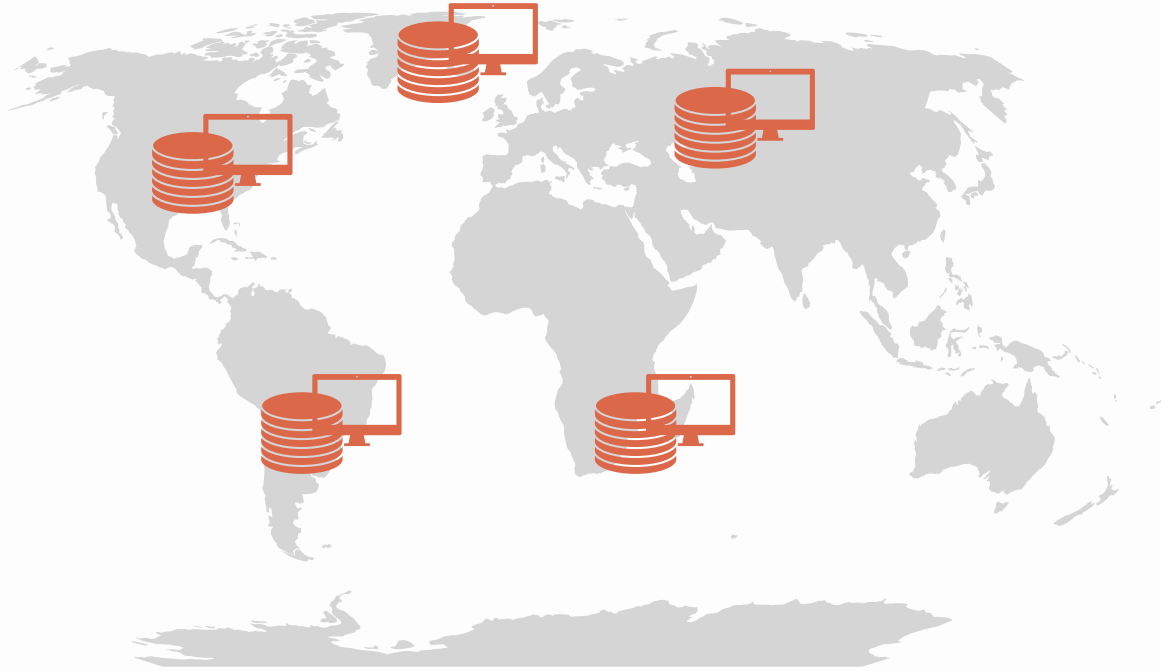
- Clustering(maximize posterior)

$$\kappa(x; G) = \operatorname{argmax}_{j \in [K]} w_j f(x; \theta_j)$$

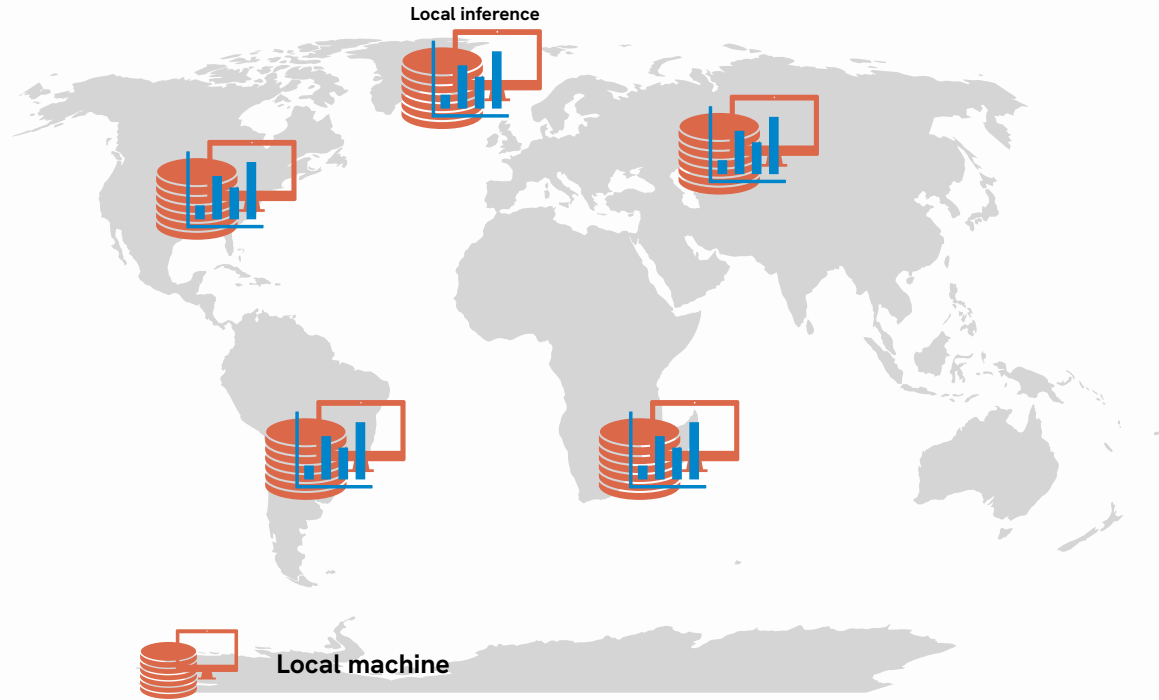


How to estimate G from data?

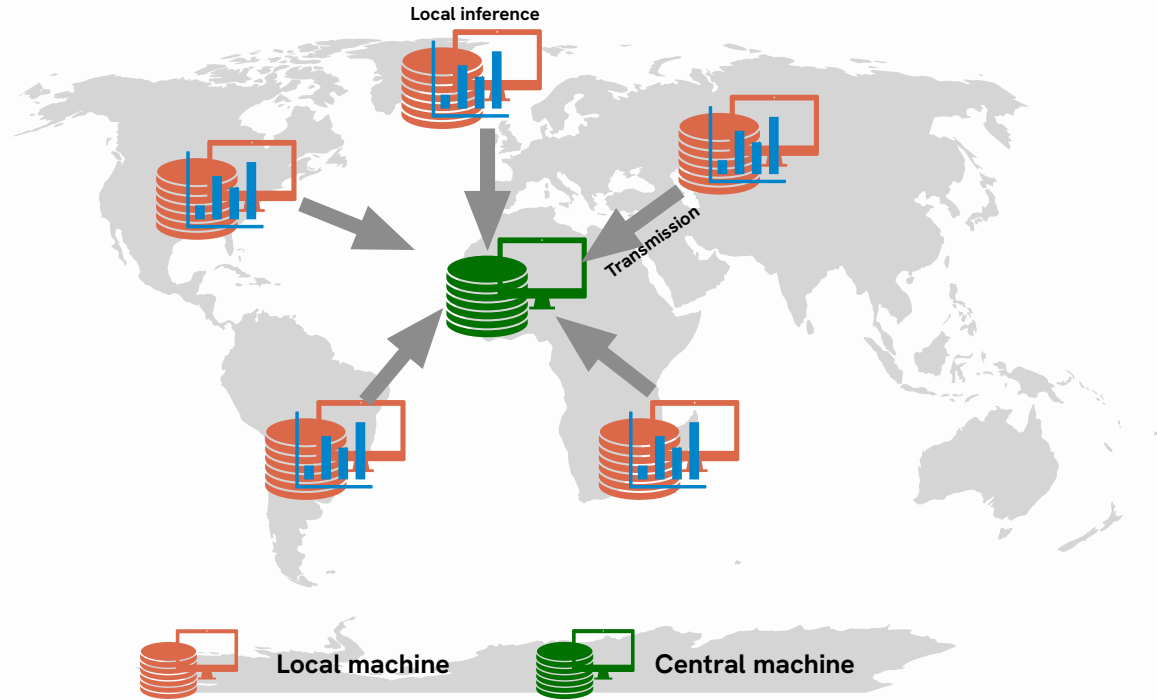
Distributed data storage & split-and-conquer



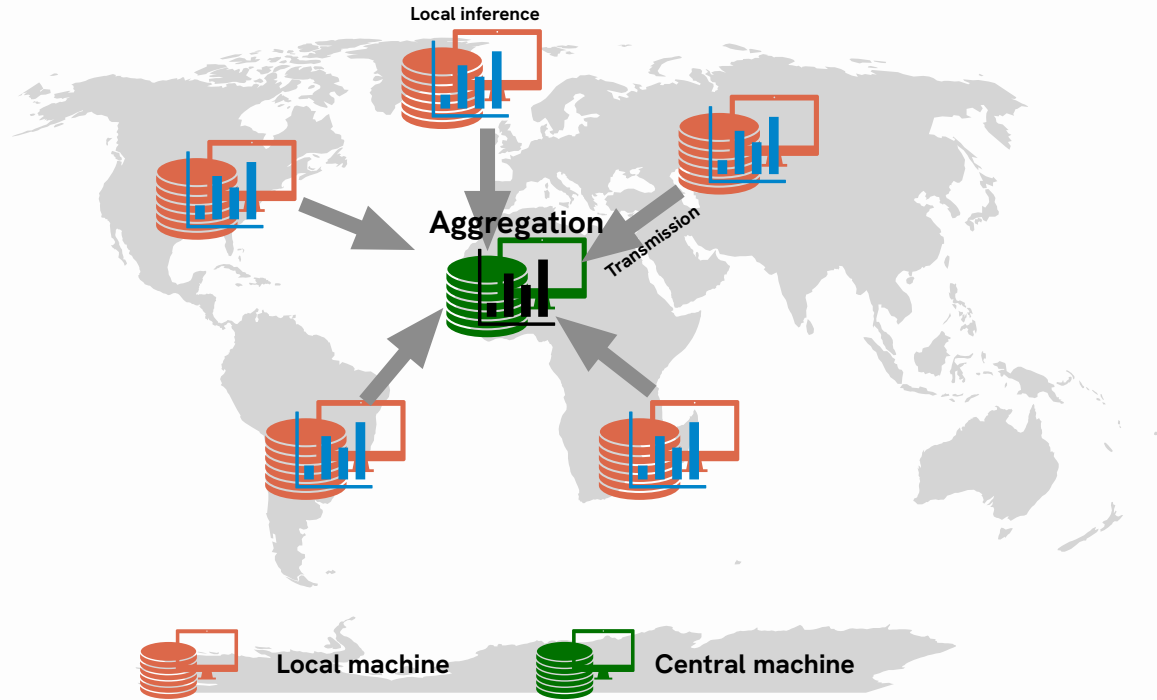
Distributed data storage & split-and-conquer



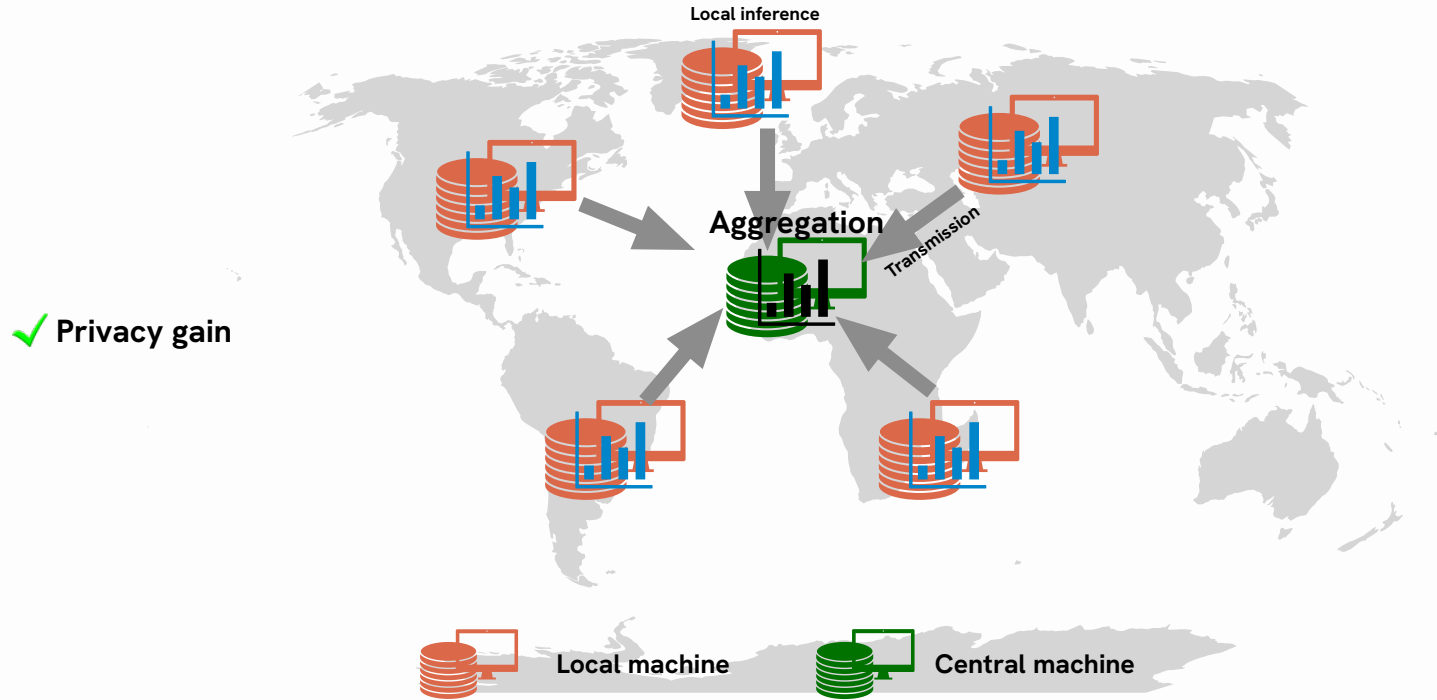
Distributed data storage & split-and-conquer



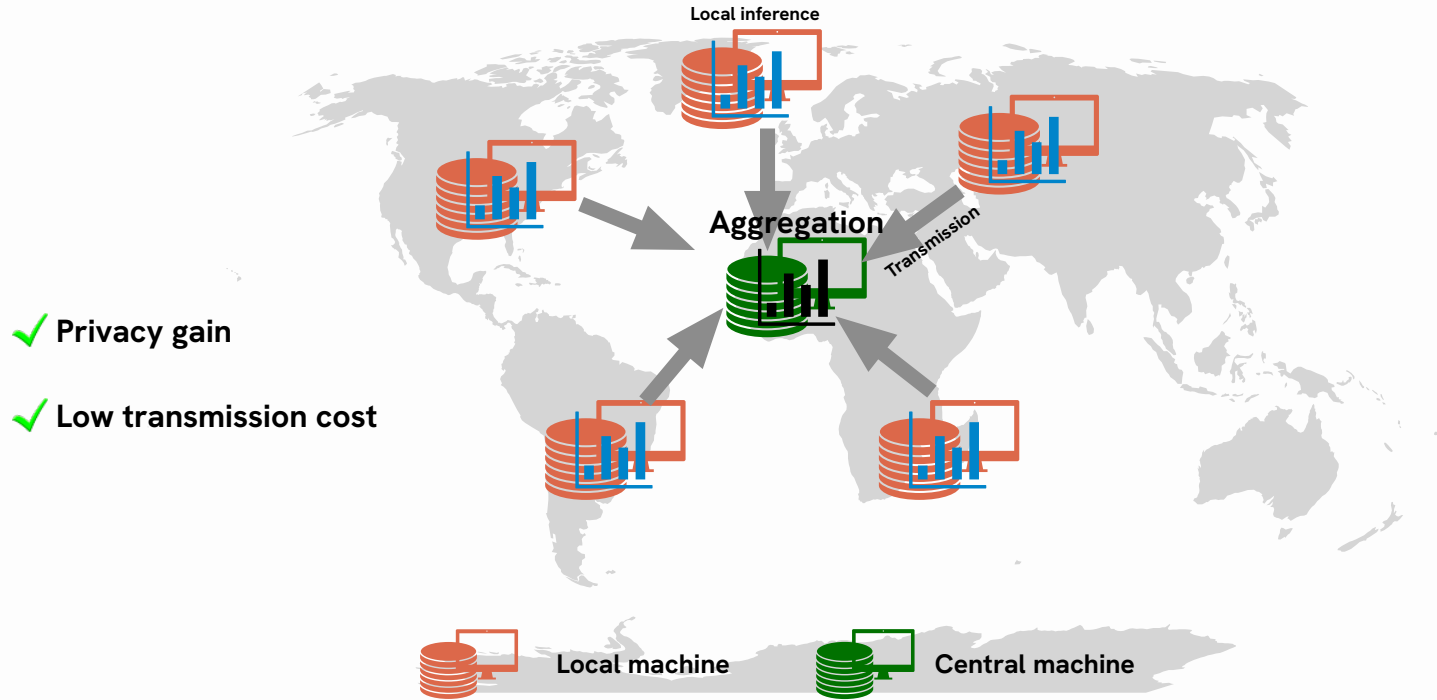
Distributed data storage & split-and-conquer



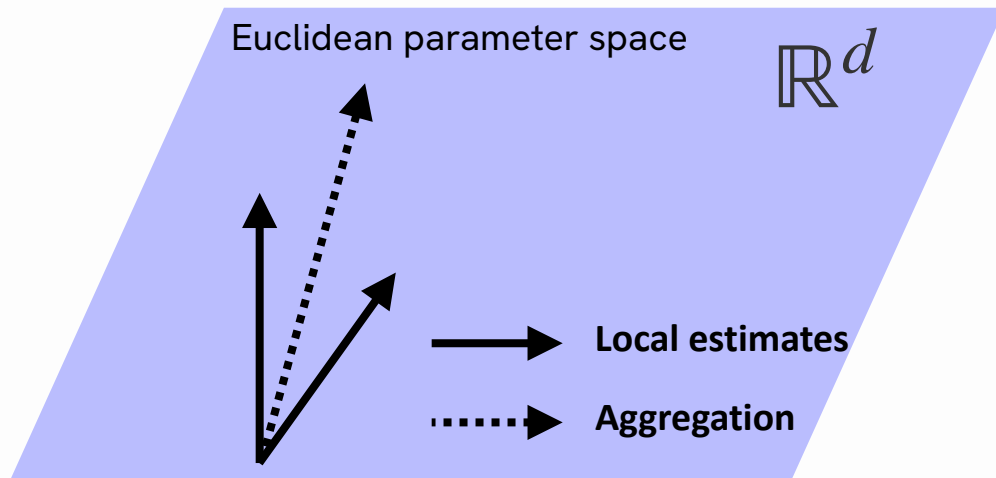
Distributed data storage & split-and-conquer



Distributed data storage & split-and-conquer





SC learning under **Euclidean** parameter space




SC learning under **Euclidean** parameter space

Local datasets

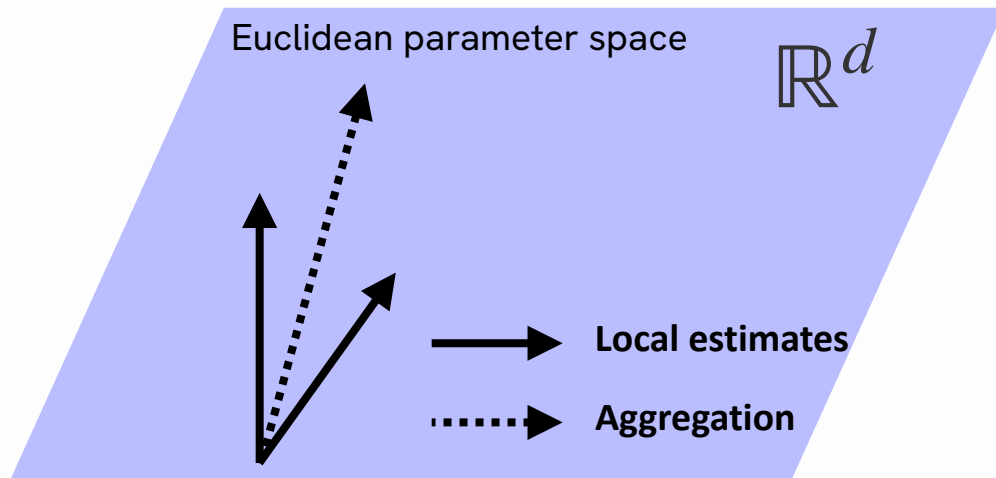
\mathcal{X}_1 

\mathcal{X}_2 

\mathcal{X}_3 

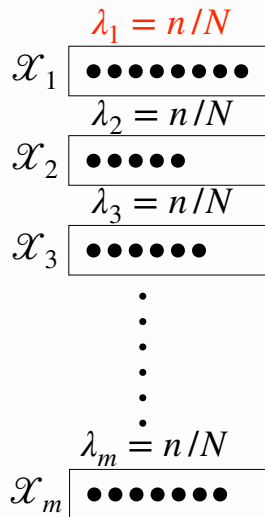
\vdots

\mathcal{X}_m 

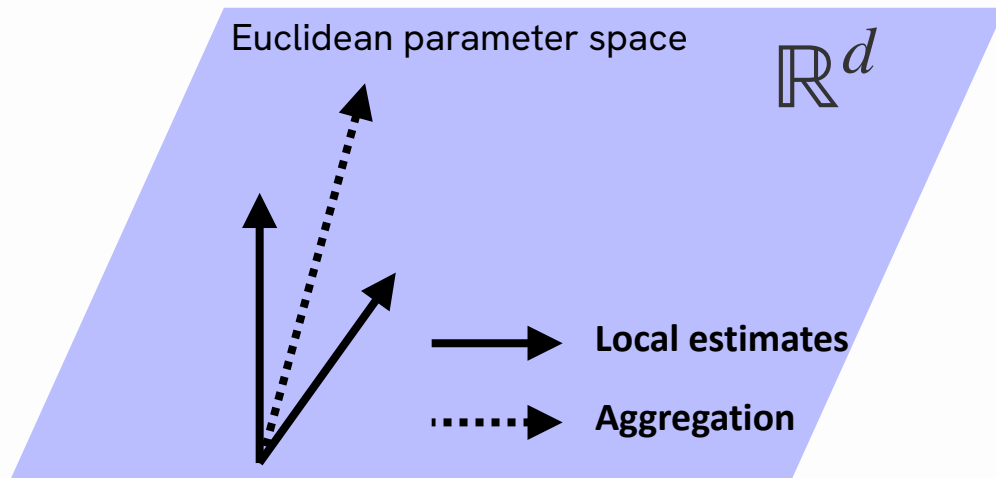


SC learning under **Euclidean** parameter space

Local datasets

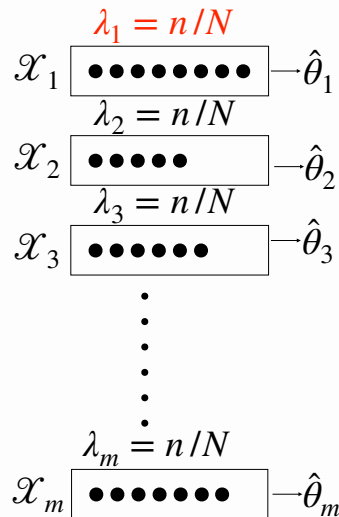


IID observations from $f(x; \theta^*)$

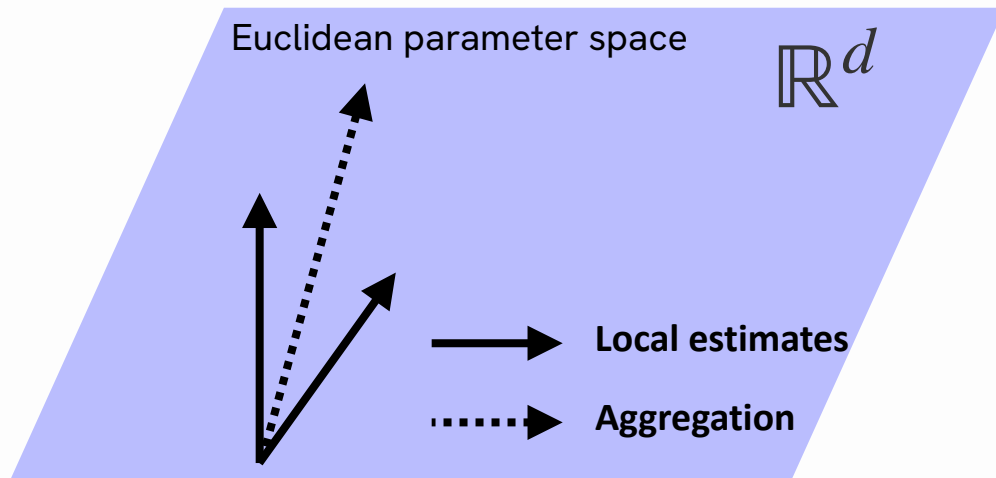


SC learning under **Euclidean** parameter space

Local datasets Local estimates

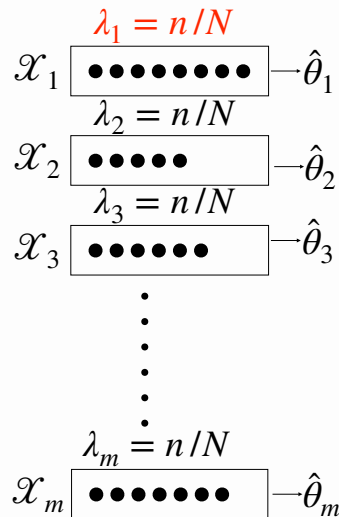


iID observations from $f(x; \theta^*)$

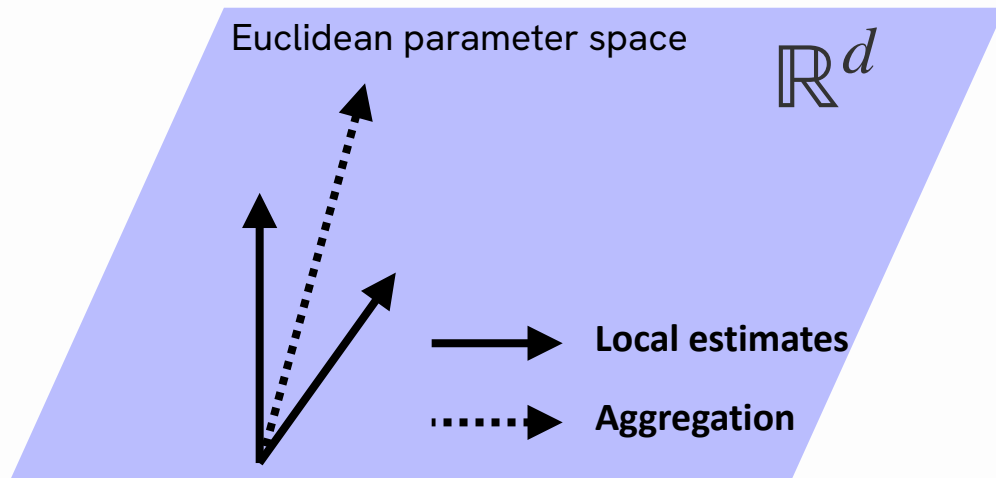


SC learning under Euclidean parameter space

Local datasets Local estimates



i.i.d. observations from $f(x; \theta^*)$

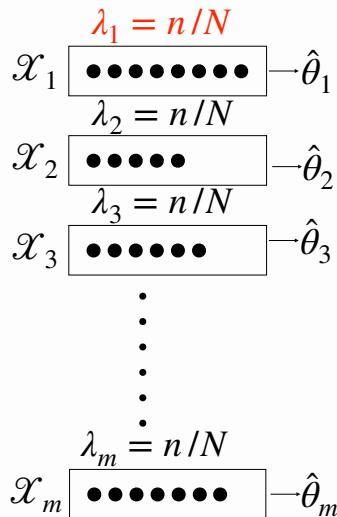


Estimator

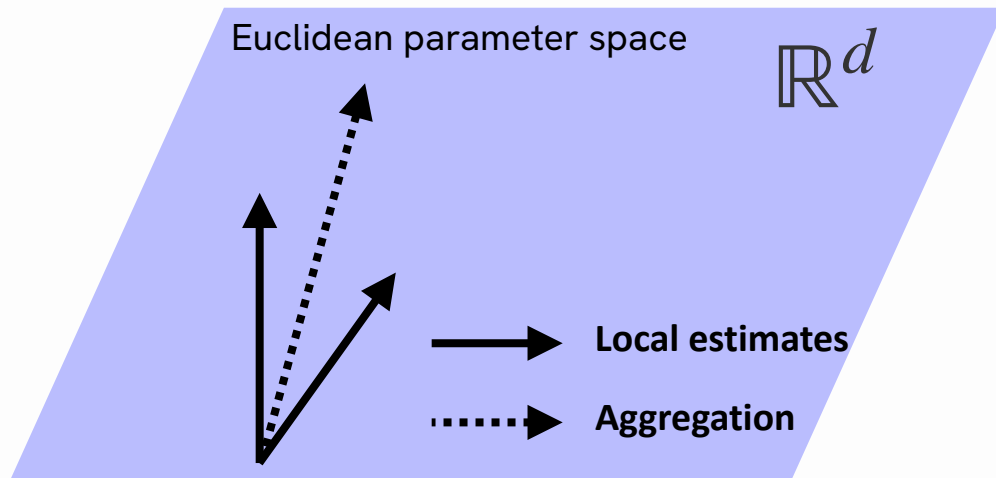
$$\boxed{\bar{\theta}} = \sum_{j=1}^m \lambda_j \hat{\theta}_j$$

SC learning under Euclidean parameter space

Local datasets Local estimates



IID observations from $f(x; \theta^*)$



Estimator

$$\boxed{\bar{\theta}} = \sum_{j=1}^m \lambda_j \hat{\theta}_j$$

n : Local sample size
 m : Number of machines
 N : Total sample size

Why finite mixture is special?

Parameter space is **non-Euclidean**

Why finite mixture is special?

Parameter space is **non-Euclidean**

Parameterization by a **vector** has **non-identifiability** issue

Why finite mixture is special?

Parameter space is **non-Euclidean**

Parameterization by a **vector** has **non-identifiability** issue

- Consider $f(x; G) = 0.4f(x; -1) + 0.6f(x; 1)$

Why finite mixture is special?

Parameter space is **non-Euclidean**

Parameterization by a **vector** has **non-identifiability** issue

- Consider $f(x; G) = 0.4f(x; -1) + 0.6f(x; 1)$
- Let $\mathbf{G}_1 = (0.4, -1, 0.6, 1)$ and $\mathbf{G}_2 = (0.6, 1, 0.4, -1)$

Why finite mixture is special?

Parameter space is **non-Euclidean**

Parameterization by a **vector** has **non-identifiability** issue

- Consider $f(x; G) = 0.4f(x; -1) + 0.6f(x; 1)$
- Let $\mathbf{G}_1 = (0.4, -1, 0.6, 1)$ and $\mathbf{G}_2 = (0.6, 1, 0.4, -1)$
- Non-identifiable: $\mathbf{G}_1 \neq \mathbf{G}_2$ but $f(x; \mathbf{G}_1) = f(x; \mathbf{G}_2)$

Why finite mixture is special?

Parameter space is **non-Euclidean**

Parameterization by a **vector** has **non-identifiability** issue

- Consider $f(x; G) = 0.4f(x; -1) + 0.6f(x; 1)$
- Let $\mathbf{G}_1 = (0.4, -1, 0.6, 1)$ and $\mathbf{G}_2 = (0.6, 1, 0.4, -1)$
- **Non-identifiable: $\mathbf{G}_1 \neq \mathbf{G}_2$ but $f(x; \mathbf{G}_1) = f(x; \mathbf{G}_2)$**
- The mixing distribution G as a distribution **does not** have this issue

Why finite mixture is special?

Parameter space is **non-Euclidean**

Parameterization by a **vector** has **non-identifiability** issue

- Consider $f(x; G) = 0.4f(x; -1) + 0.6f(x; 1)$
- Let $\mathbf{G}_1 = (0.4, -1, 0.6, 1)$ and $\mathbf{G}_2 = (0.6, 1, 0.4, -1)$
- **Non-identifiable: $\mathbf{G}_1 \neq \mathbf{G}_2$ but $f(x; \mathbf{G}_1) = f(x; \mathbf{G}_2)$**
- The mixing distribution G as a distribution **does not** have this issue

- $$\mathbb{G}_K = \left\{ G = \sum_{k=1}^K w_k \delta_{\theta_k} : \theta_k \in \Theta, w_k \in (0,1), \sum_k w_k = 1 \right\}$$

Why finite mixture is special?

Parameter space is **non-Euclidean**

Parameterization by a **vector** has **non-identifiability** issue

- Consider $f(x; G) = 0.4f(x; -1) + 0.6f(x; 1)$
- Let $\mathbf{G}_1 = (0.4, -1, 0.6, 1)$ and $\mathbf{G}_2 = (0.6, 1, 0.4, -1)$
- **Non-identifiable: $\mathbf{G}_1 \neq \mathbf{G}_2$ but $f(x; \mathbf{G}_1) = f(x; \mathbf{G}_2)$**
- The mixing distribution G as a distribution **does not** have this issue

- $\mathbb{G}_K = \left\{ G = \sum_{k=1}^K w_k \delta_{\theta_k} : \theta_k \in \Theta, w_k \in (0,1), \sum_k w_k = 1 \right\}$

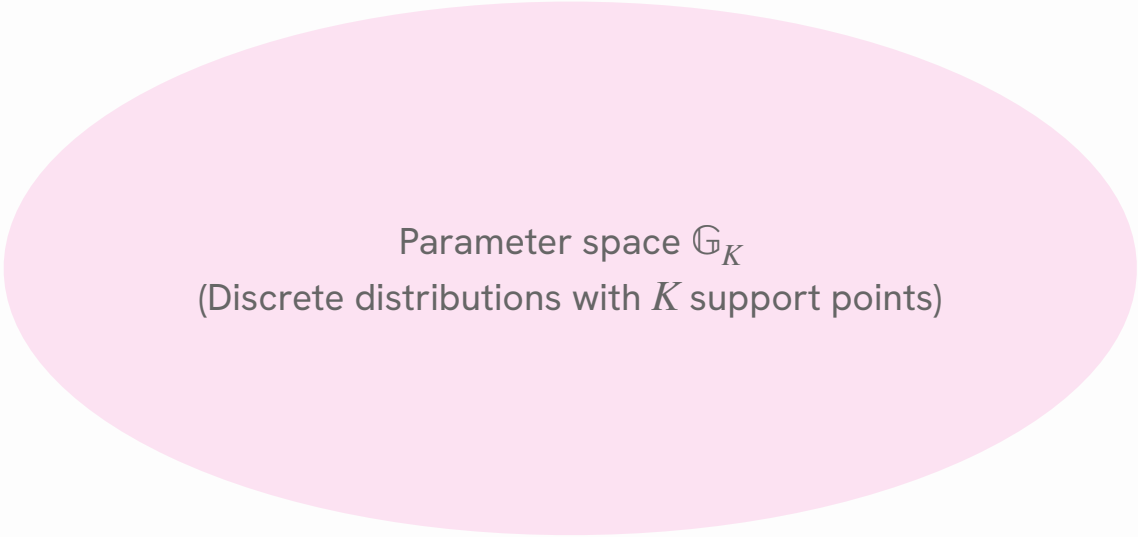
Parameter space: Discrete distribution with at most K support points

Why finite mixture is special?

Parameter space is **non-Euclidean**, conventional method does not apply

Why finite mixture is special?

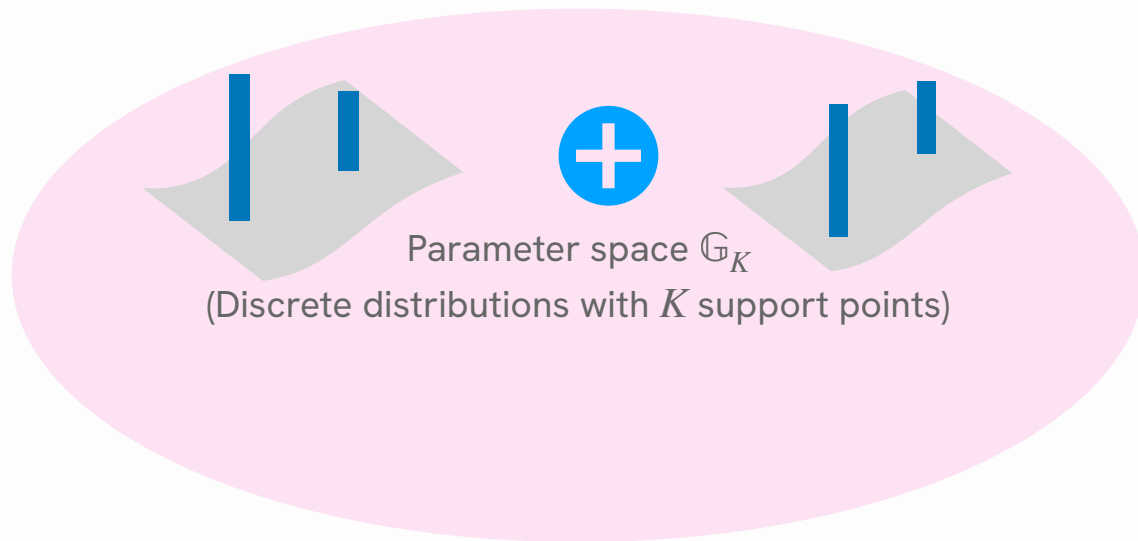
Parameter space is **non-Euclidean**, conventional method does not apply



Parameter space \mathbb{G}_K
(Discrete distributions with K support points)

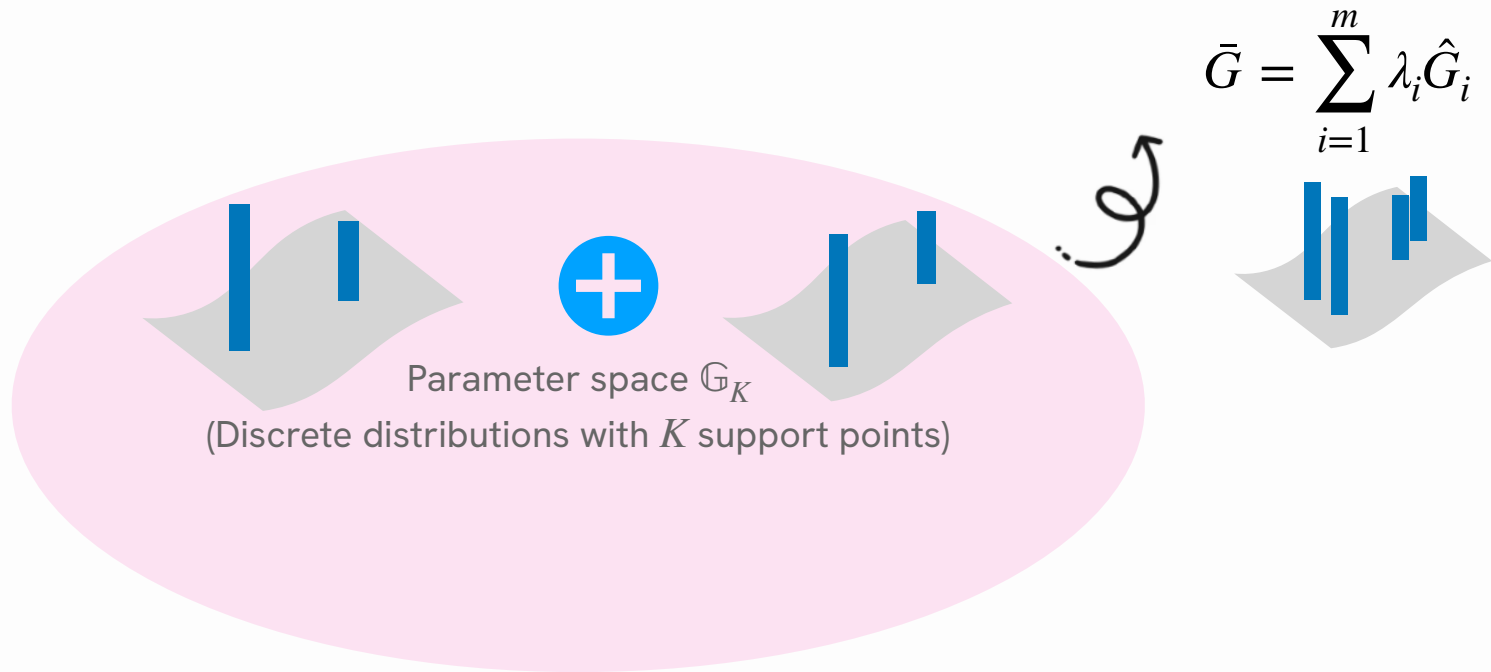
Why finite mixture is special?

Parameter space is **non-Euclidean**, conventional method does not apply



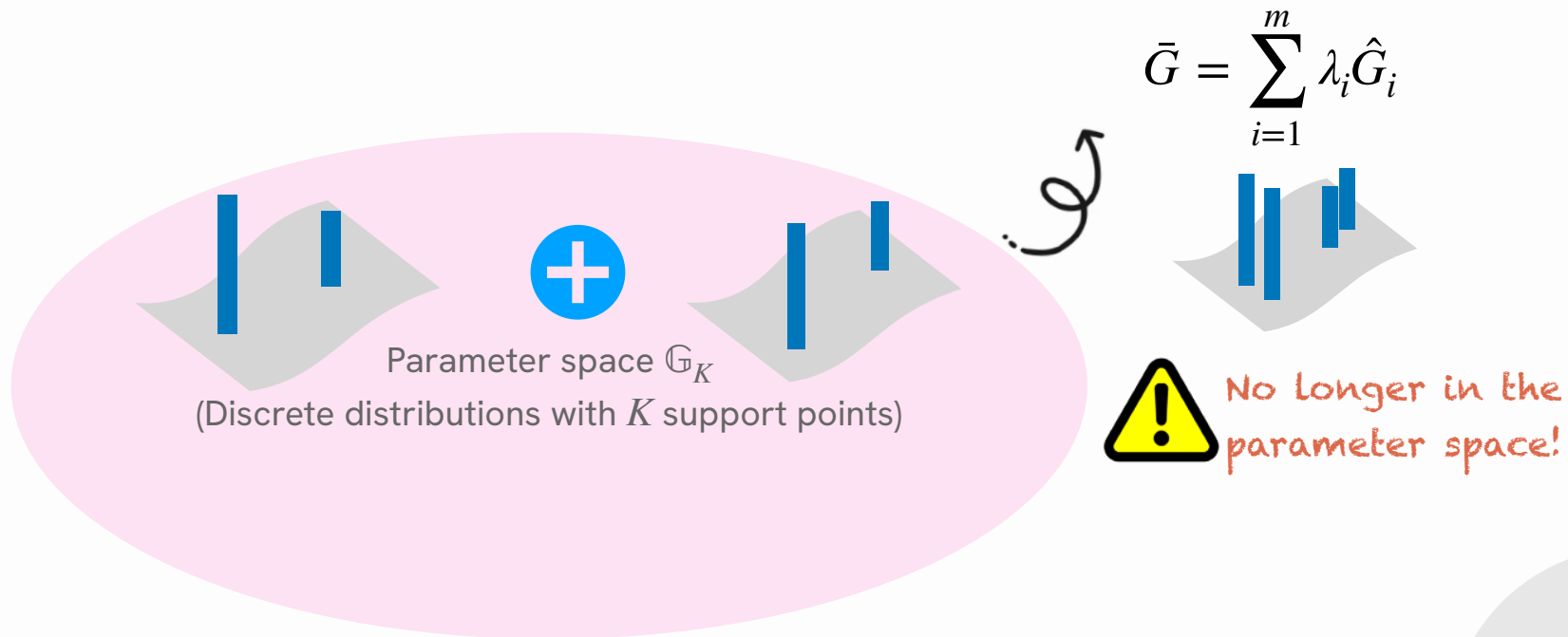
Why finite mixture is special?

Parameter space is **non-Euclidean**, conventional method does not apply



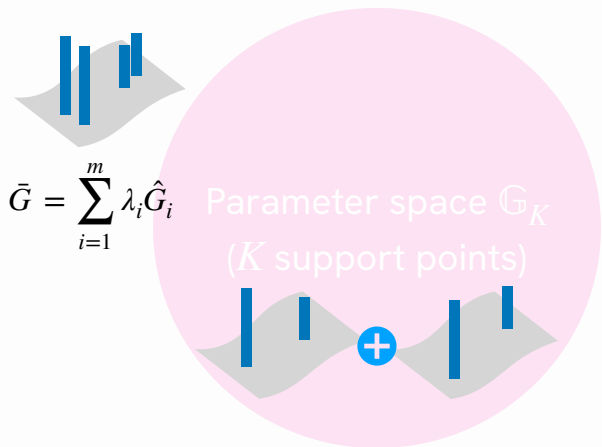
Why finite mixture is special?

Parameter space is **non-Euclidean**, conventional method does not apply



Our reduction based aggregation

Paper Link



Our reduction based aggregation

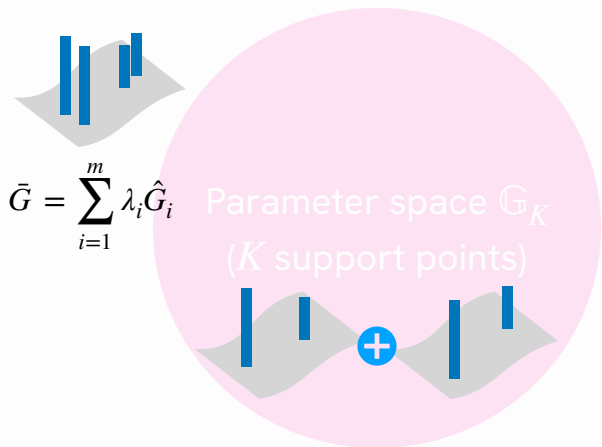
Zhang, Q., & Chen, J. (2022). Distributed learning of finite Gaussian mixtures. *JMLR*.

Paper Link



- Zhang and Chen (JMLR 2022): reduction approach

$$\bar{G}^R = \operatorname{arginf}_{G \in \mathbb{G}_K} \rho(\bar{G}, G)$$



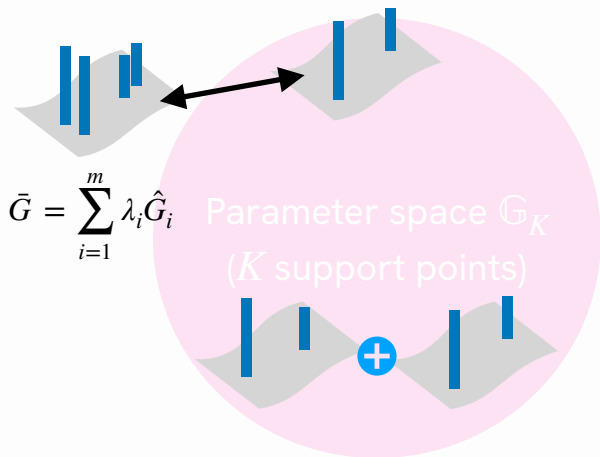
Our reduction based aggregation

Zhang, Q., & Chen, J. (2022). Distributed learning of finite Gaussian mixtures. *JMLR*.

Paper Link



**Smallest composite
transportation divergence**



- Zhang and Chen (JMLR 2022): reduction approach

$$\bar{G}^R = \operatorname{arginf}_{G \in \mathbb{G}_K} \rho(\bar{G}, G)$$

- $\rho(\cdot, \cdot)$: composite transportation divergence (for efficient computation)

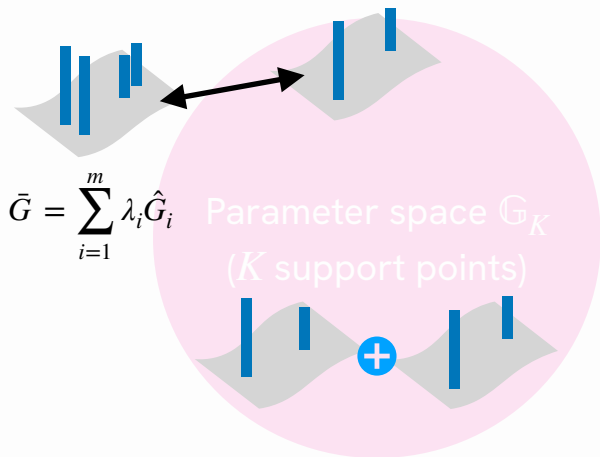
Our reduction based aggregation

Zhang, Q., & Chen, J. (2022). Distributed learning of finite Gaussian mixtures. *JMLR*.

Paper link



**Smallest composite
transportation divergence**



- Zhang and Chen (JMLR 2022): reduction approach

$$\bar{G}^R = \operatorname{arginf}_{G \in \mathbb{G}_K} \rho(\bar{G}, G)$$

- $\rho(\cdot, \cdot)$: composite transportation divergence (for efficient computation)
- \bar{G}^R is $O_P(N^{-1/2})$ when $n \geq m$; $\sqrt{N}(\bar{G}^R - G^*) \rightarrow N(0, I^{-1}(G^*))$ when $m = o(n)$

*The asymptotic results are represented after some ordering of the G into vectors

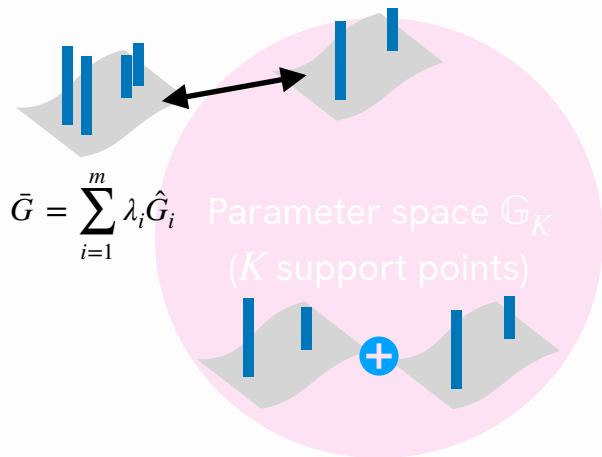
Our reduction based aggregation

Zhang, Q., & Chen, J. (2022). Distributed learning of finite Gaussian mixtures. *JMLR*.

Paper link



**Smallest composite
transportation divergence**



- Zhang and Chen (JMLR 2022): reduction approach

$$\bar{G}^R = \operatorname{arginf}_{G \in \mathbb{G}_K} \rho(\bar{G}, G)$$

- $\rho(\cdot, \cdot)$: composite transportation divergence (for efficient computation)
- \bar{G}^R is $O_P(N^{-1/2})$ when $n \geq m$; $\sqrt{N}(\bar{G}^R - G^*) \rightarrow N(0, I^{-1}(G^*))$ when $m = o(n)$
- An efficient MM algorithm: K -means clustering on \mathcal{F}

*The asymptotic results are represented after some ordering of the G into vectors

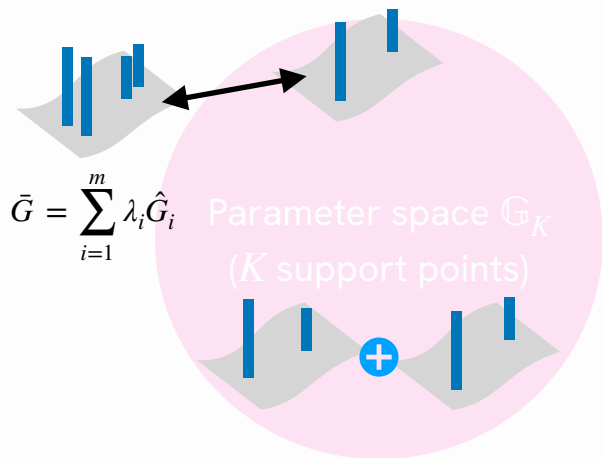
Our reduction based aggregation

Zhang, Q., & Chen, J. (2022). Distributed learning of finite Gaussian mixtures. *JMLR*.

Paper Link



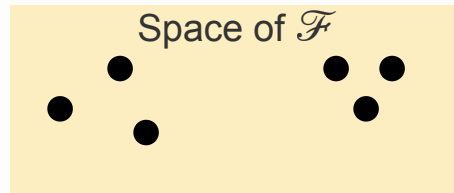
**Smallest composite
transportation divergence**



- Zhang and Chen (JMLR 2022): reduction approach

$$\bar{G}^R = \operatorname{arginf}_{G \in \mathbb{G}_K} \rho(\bar{G}, G)$$

- $\rho(\cdot, \cdot)$: composite transportation divergence (for efficient computation)
- \bar{G}^R is $O_P(N^{-1/2})$ when $n \geq m$; $\sqrt{N}(\bar{G}^R - G^*) \rightarrow N(0, I^{-1}(G^*))$ when $m = o(n)$
- An efficient MM algorithm: K -means clustering on \mathcal{F}



Demo:
estimate 2-
component
mixture with 3
machines

*The asymptotic results are represented after some ordering of the G into vectors

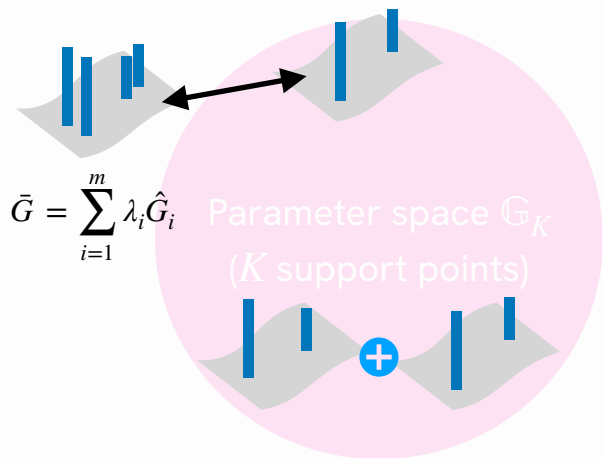
Our reduction based aggregation

Zhang, Q., & Chen, J. (2022). Distributed learning of finite Gaussian mixtures. *JMLR*.

Paper link



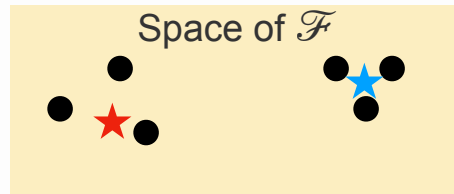
**Smallest composite
transportation divergence**



- Zhang and Chen (JMLR 2022): reduction approach

$$\bar{G}^R = \operatorname{arginf}_{G \in \mathbb{G}_K} \rho(\bar{G}, G)$$

- $\rho(\cdot, \cdot)$: composite transportation divergence (for efficient computation)
- \bar{G}^R is $O_P(N^{-1/2})$ when $n \geq m$; $\sqrt{N}(\bar{G}^R - G^*) \rightarrow N(0, I^{-1}(G^*))$ when $m = o(n)$
- An efficient MM algorithm: K -means clustering on \mathcal{F}



Demo:
estimate 2-
component
mixture with 3
machines

Initialization

*The asymptotic results are represented after some ordering of the G into vectors

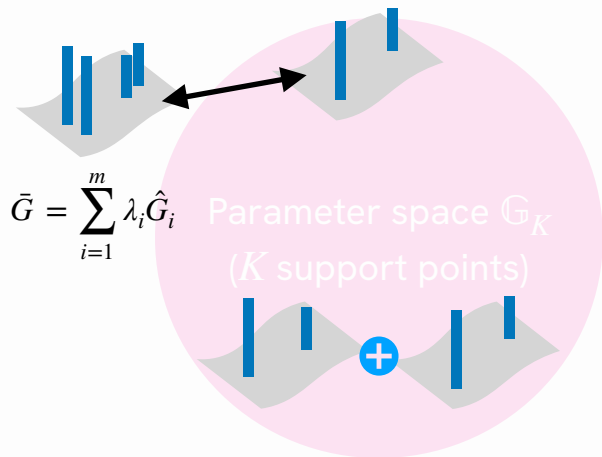
Our reduction based aggregation

Zhang, Q., & Chen, J. (2022). Distributed learning of finite Gaussian mixtures. *JMLR*.

Paper link



**Smallest composite
transportation divergence**



- Zhang and Chen (JMLR 2022): reduction approach

$$\bar{G}^R = \operatorname{arginf}_{G \in \mathbb{G}_K} \rho(\bar{G}, G)$$

- $\rho(\cdot, \cdot)$: composite transportation divergence (for efficient computation)
- \bar{G}^R is $O_P(N^{-1/2})$ when $n \geq m$; $\sqrt{N}(\bar{G}^R - G^*) \rightarrow N(0, I^{-1}(G^*))$ when $m = o(n)$
- An efficient MM algorithm: K -means clustering on \mathcal{F}



Demo:
estimate 2-
component
mixture with 3
machines

Initialization \longrightarrow Majorization

*The asymptotic results are represented after some ordering of the G into vectors

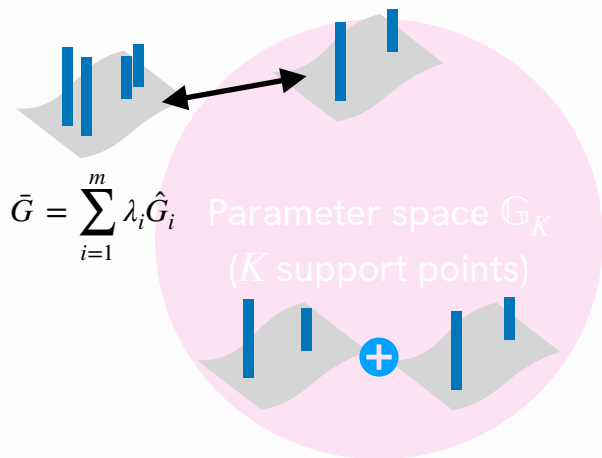
Our reduction based aggregation

Zhang, Q., & Chen, J. (2022). Distributed learning of finite Gaussian mixtures. *JMLR*.

Paper link



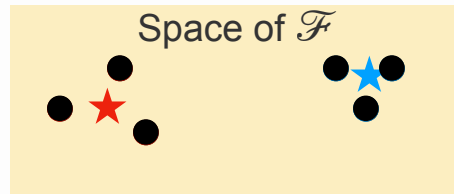
**Smallest composite
transportation divergence**



- Zhang and Chen (JMLR 2022): reduction approach

$$\bar{G}^R = \operatorname{arginf}_{G \in \mathbb{G}_K} \rho(\bar{G}, G)$$

- $\rho(\cdot, \cdot)$: composite transportation divergence (for efficient computation)
- \bar{G}^R is $O_P(N^{-1/2})$ when $n \geq m$; $\sqrt{N}(\bar{G}^R - G^*) \rightarrow N(0, I^{-1}(G^*))$ when $m = o(n)$
- An efficient MM algorithm: K -means clustering on \mathcal{F}



Demo:
estimate 2-
component
mixture with 3
machines

Initialization \longrightarrow Majorization \longrightarrow Minimization

*The asymptotic results are represented after some ordering of the G into vectors

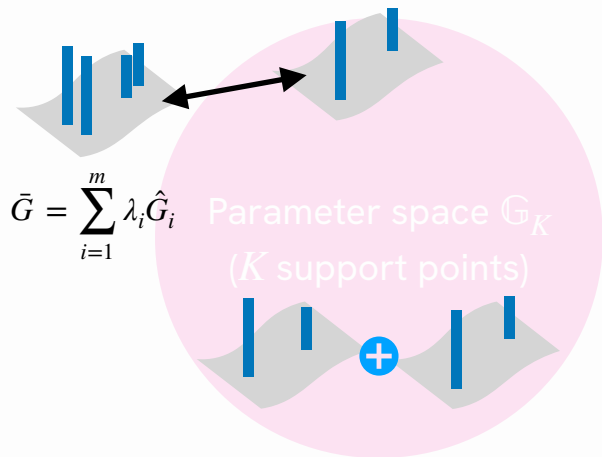
Our reduction based aggregation

Zhang, Q., & Chen, J. (2022). Distributed learning of finite Gaussian mixtures. *JMLR*.

Paper link



**Smallest composite
transportation divergence**



- Zhang and Chen (JMLR 2022): reduction approach

$$\bar{G}^R = \operatorname{arginf}_{G \in \mathbb{G}_K} \rho(\bar{G}, G)$$

- $\rho(\cdot, \cdot)$: composite transportation divergence (for efficient computation)
- \bar{G}^R is $O_P(N^{-1/2})$ when $n \geq m$; $\sqrt{N}(\bar{G}^R - G^*) \rightarrow N(0, I^{-1}(G^*))$ when $m = o(n)$
- An efficient MM algorithm: K -means clustering on \mathcal{F}



Demo:
estimate 2-
component
mixture with 3
machines

Initialization → Majorization → Minimization → Majorization

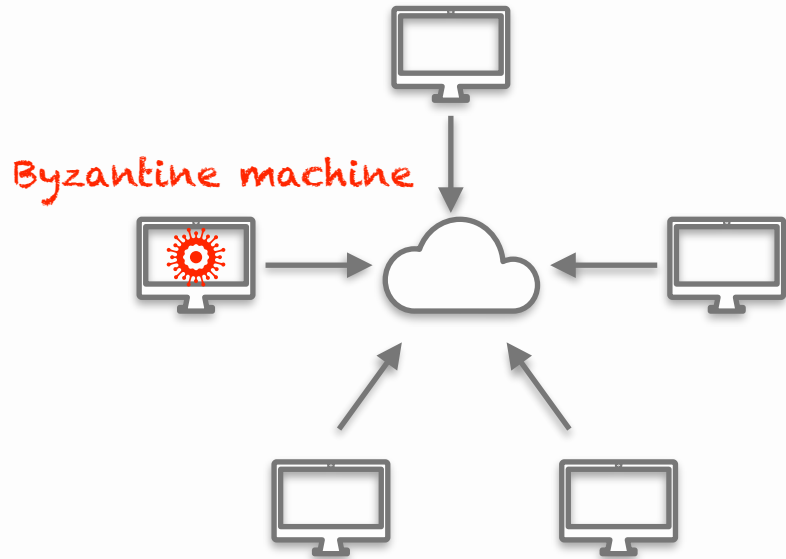
*The asymptotic results are represented after some ordering of the G into vectors

What is Byzantine failure?

A subset of these machines (*Byzantine machines*) may transmit arbitrary or malicious messages to the central machine.

What is Byzantine failure?

A subset of these machines (*Byzantine machines*) may transmit arbitrary or malicious messages to the central machine.



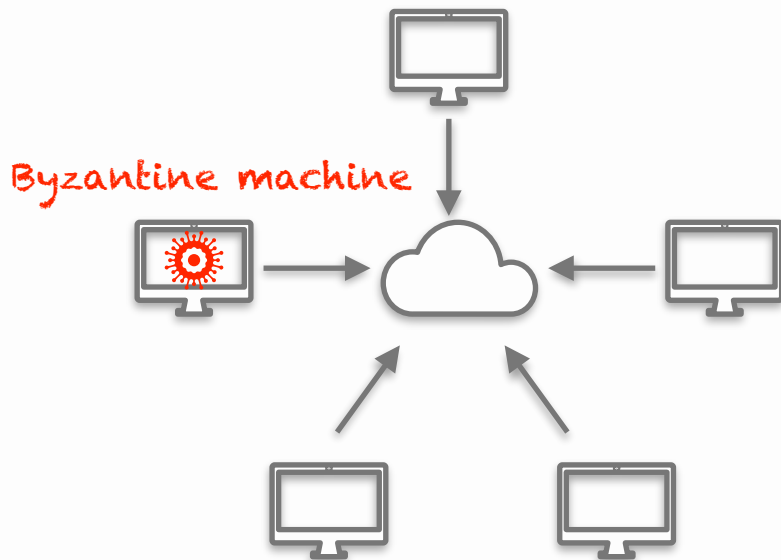
What is Byzantine failure?

A subset of these machines (*Byzantine machines*) may transmit arbitrary or malicious messages to the central machine.

The server receives:

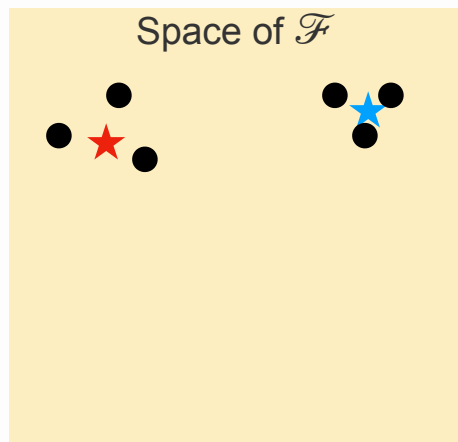
$$\tilde{G}_j = \begin{cases} \hat{G}_j, & \text{when } j \notin \mathbb{B} \\ \xi_i, & \text{when } j \in \mathbb{B} \end{cases}$$

Arbitrary mixing distribution

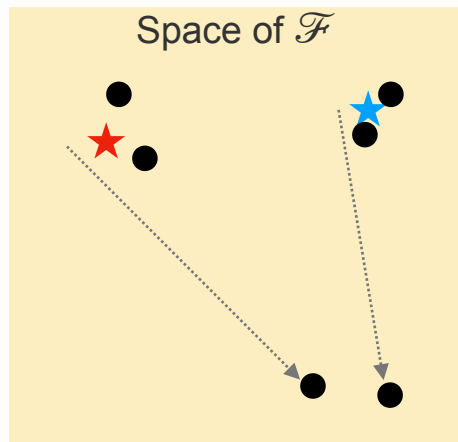


What is the consequence?

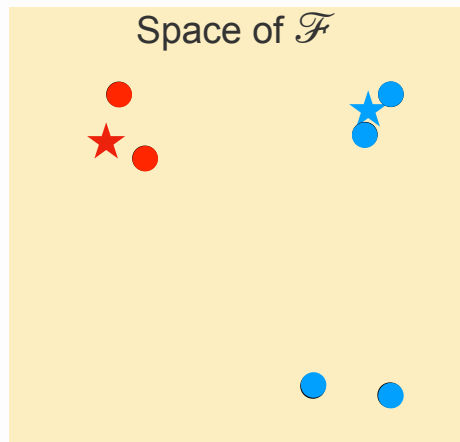
What is the consequence?



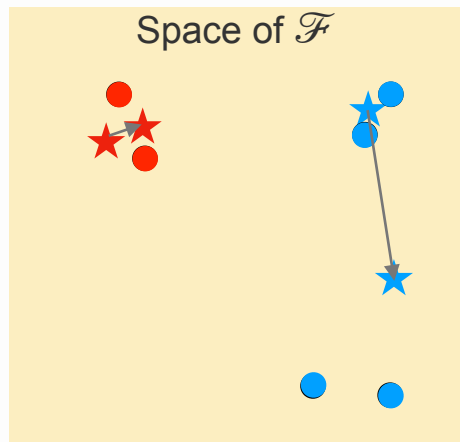
What is the consequence?



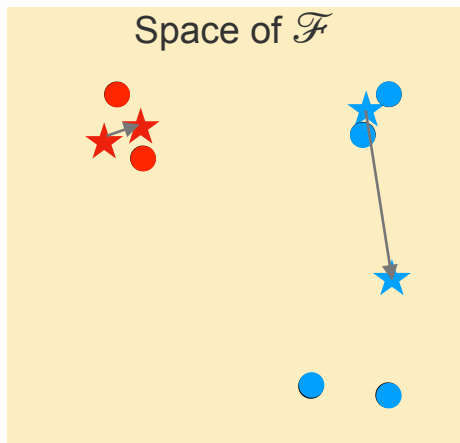
What is the consequence?



What is the consequence?



What is the consequence?



Distorted aggregation result

Existing Byzantine-tolerant aggregation methods

Robust alternative of mean such as:

- *Coordinate-wise median (Yin et al., 2018)*
- *Geometric median (Lai et al., 2016; Steinhardt, 2019)*
- *Trimmed mean (Yin et al., 2018)*
- *Median of means (Lugosi and Mendelson, 2019)*
- *Filtering (Diaklnikolas et al., 2017, 2019; Steinhardt et al., 2017; Zhu et al., 2021, Zhu et al., 2023)*
- *Krum (Blanchard et al., 2017; Chen et al., 2018; El El Mhamdi et al., 2018)*
- *No-regret (Zhu et al., 2021; Hopkins et al., 2020, Zhu et al., 2023)*
- *GAN (Zhu et al., 2022; Gao et al., 2020, Zhu et al., 2023)*
- ...

Existing Byzantine-tolerant aggregation methods

Developed for **Euclidean** parameter space, does not apply under mixture

Robust alternative of mean such as:

- Coordinate-wise median (*Yin et al., 2018*)
- Geometric median (*Lai et al., 2016; Steinhardt, 2019*)
- Trimmed mean (*Yin et al., 2018*)
- Median of means (*Lugosi and Mendelson, 2019*)
- Filtering (*Diaklnikolas et al., 2017, 2019; Steinhardt et al., 2017; Zhu et al., 2021, Zhu et al., 2023*)
- Krum (*Blanchard et al., 2017; Chen et al., 2018; El El Mhamdi et al., 2018*)
- No-regret (*Zhu et al., 2021; Hopkins et al., 2020, Zhu et al., 2023*)
- GAN (*Zhu et al., 2022; Gao et al., 2020, Zhu et al., 2023*)
- ...

Existing Byzantine-tolerant aggregation methods

Developed for **Euclidean** parameter space, does not apply under mixture

Robust alternative of mean such as:

- Coordinate-wise median (*Yin et al., 2018*)
- Geometric median (*Lai et al., 2016; Steinhardt, 2019*)
- Trimmed mean (*Yin et al., 2018*)
- Median of means (*Lugosi and Mendelson, 2019*)
- Filtering (*Diaklnikolas et al., 2017, 2019; Steinhardt et al., 2017; Zhu et al., 2021, Zhu et al., 2023*)
- Krum (*Blanchard et al., 2017; Chen et al., 2018; El El Mhamdi et al., 2018*)
- No-regret (*Zhu et al., 2021; Hopkins et al., 2020, Zhu et al., 2023*)
- GAN (*Zhu et al., 2022; Gao et al., 2020, Zhu et al., 2023*)
- ...

We consider Byzantine-tolerant distributed learning of finite mixture models

Our method at a glance

We consider the majority of the machines are failure-free, $|\mathbb{B}| = \alpha m$ with $\alpha < 1/2$

Our method at a glance

We consider the majority of the machines are failure-free, $|\mathbb{B}| = \alpha m$ with $\alpha < 1/2$

Step 1

Use distance based
method to filter
out "bad" estimators

Our method at a glance

We consider the majority of the machines are failure-free, $|\mathbb{B}| = \alpha m$ with $\alpha < 1/2$

Step 1

Use distance based
method to filter
out "bad" estimators

Step 2

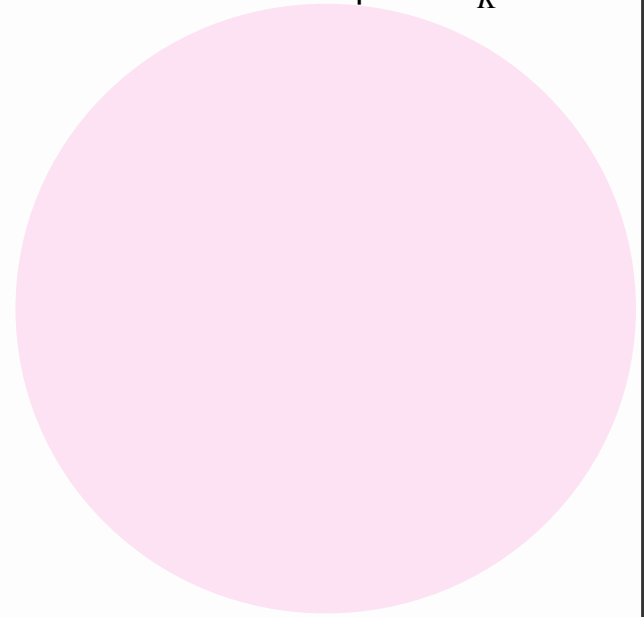
Aggregate the remaining
use the reduction
method introduced
earlier in this talk

Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$
distance from G^* in L^2

Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2



Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

Parameter space \mathbb{G}_K

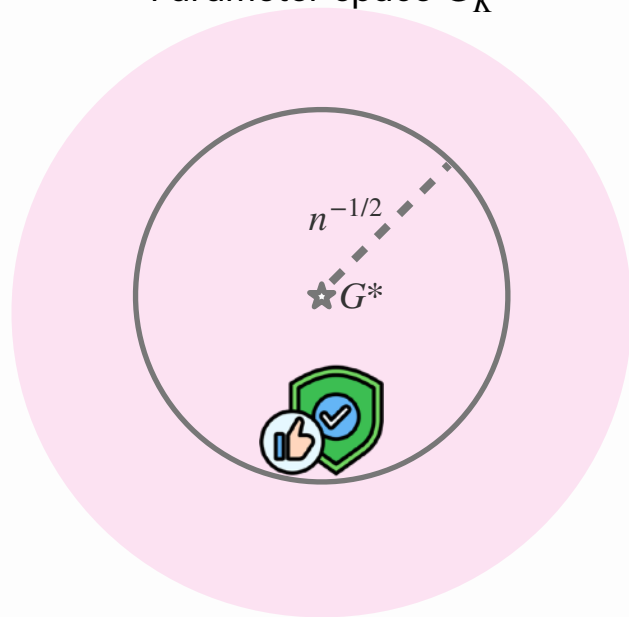
★ G^*



Intuition for our method

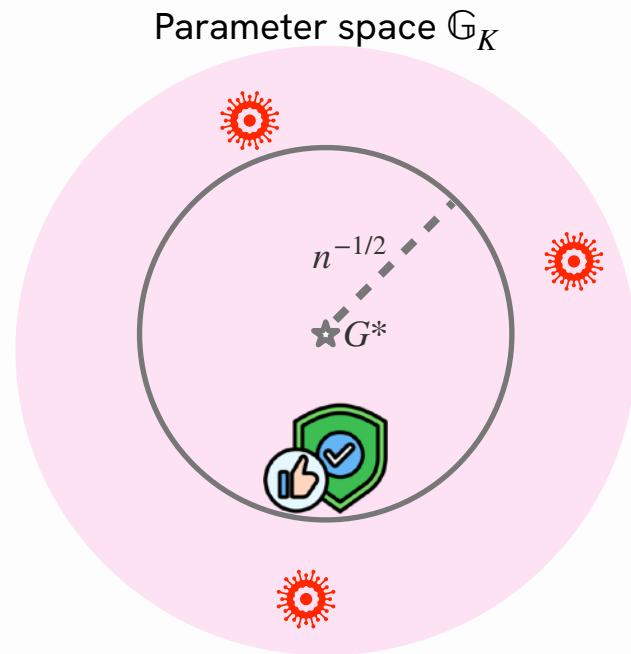
The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

Parameter space \mathbb{G}_K



Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

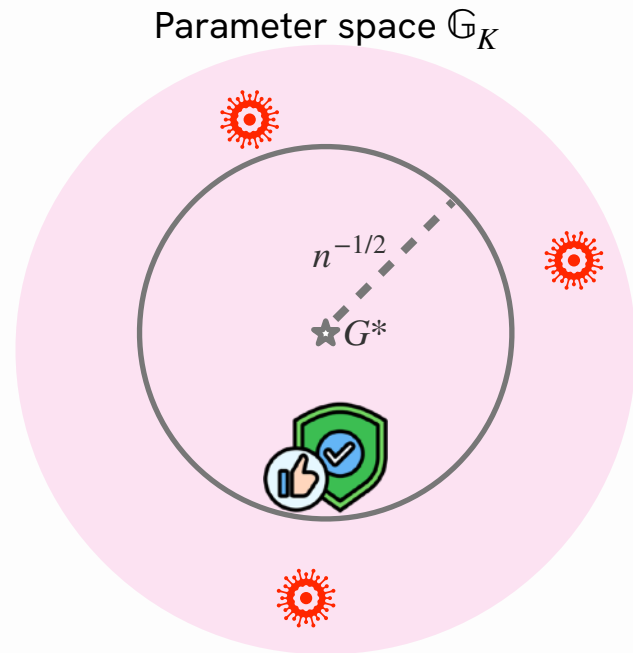


Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

- As both $G, G' \rightarrow G^*$, we have

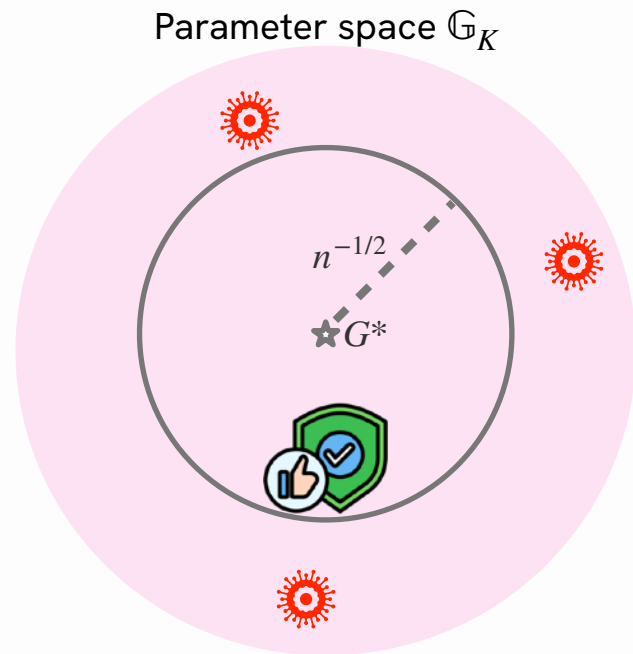
$$nL^2(G, G') \approx \sqrt{n}(G - G')^\top H^* \sqrt{n}(G - G')$$



Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

- As both $G, G' \rightarrow G^*$, we have **Asymp. generalized χ^2**
 $nL^2(G, G') \approx \sqrt{n}(G - G')^\top H^* \sqrt{n}(G - G')$

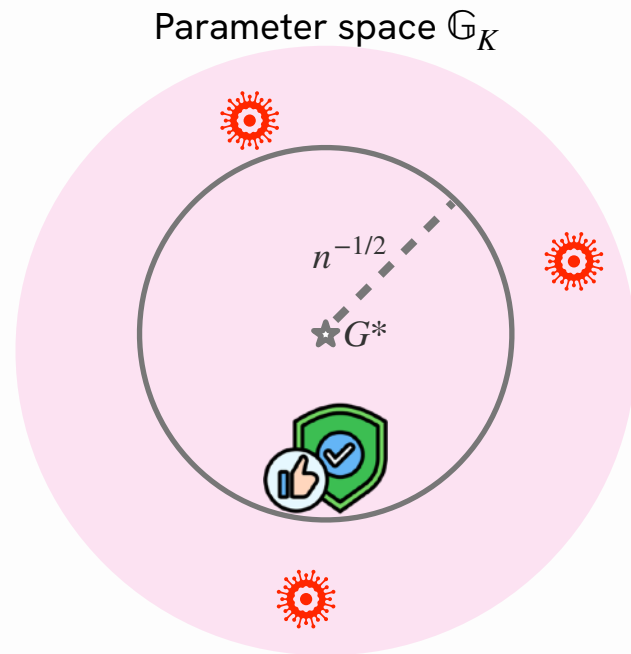


Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

- As both $G, G' \rightarrow G^*$, we have **Asymp. generalized χ^2**
$$nL^2(G, G') \approx \sqrt{n}(G - G')^\top H^* \sqrt{n}(G - G')$$

- For **failure-free machine** estimates



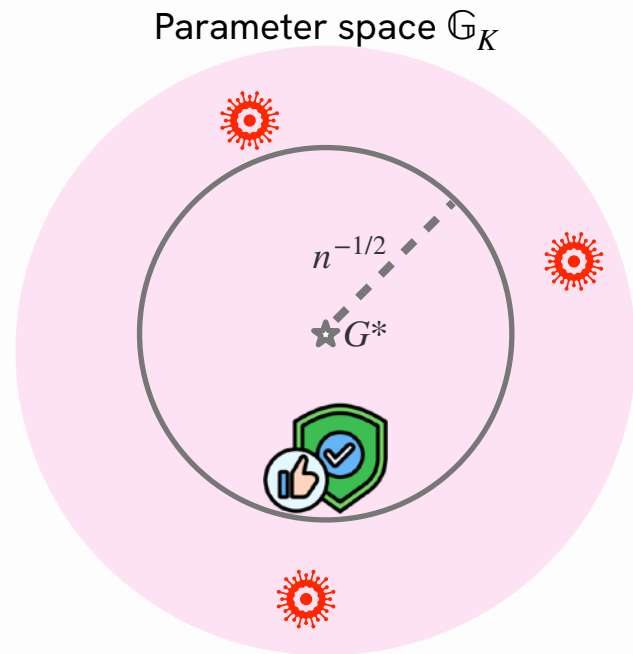
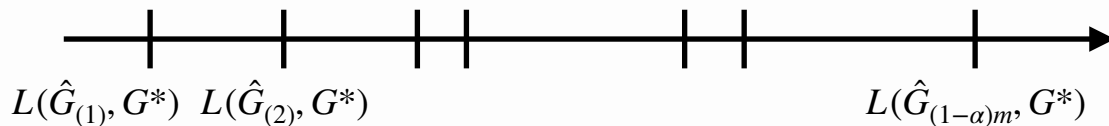
Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

- As both $G, G' \rightarrow G^*$, we have **Asymp. generalized χ^2**

$$nL^2(G, G') \approx \sqrt{n}(G - G')^\top H^* \sqrt{n}(G - G')$$

- For **failure-free machine** estimates



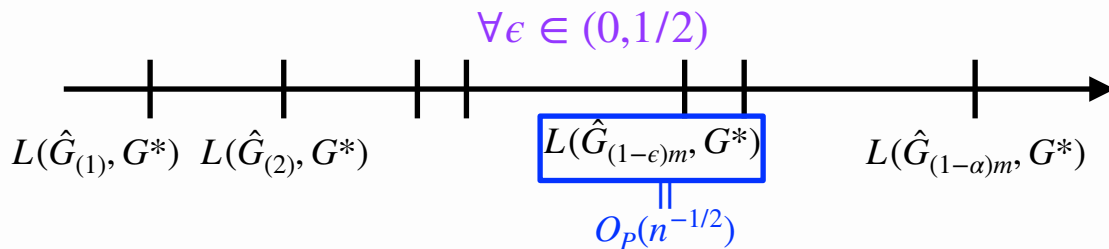
Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

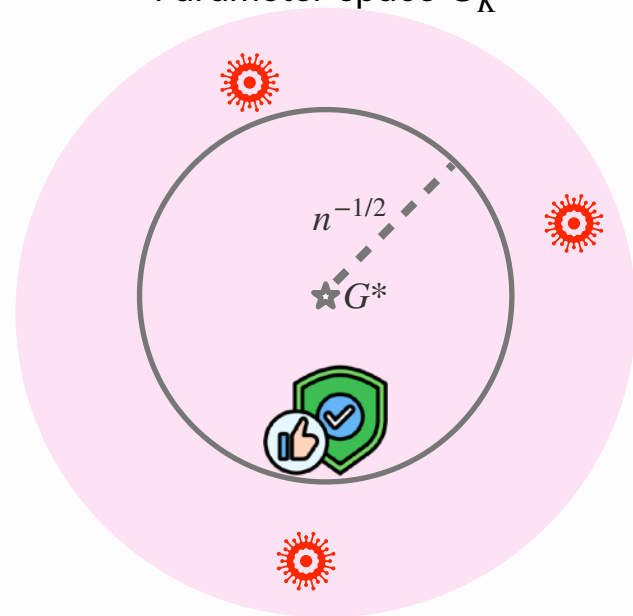
- As both $G, G' \rightarrow G^*$, we have **Asymp. generalized χ^2**

$$nL^2(G, G') \approx \sqrt{n}(G - G')^\top H^* \sqrt{n}(G - G')$$

- For **failure-free machine** estimates



Parameter space \mathbb{G}_K



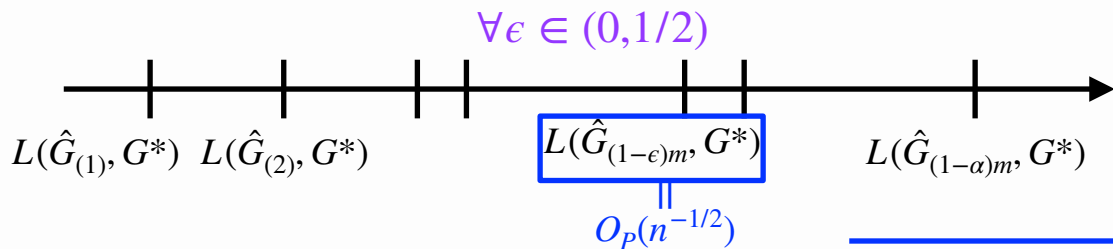
Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

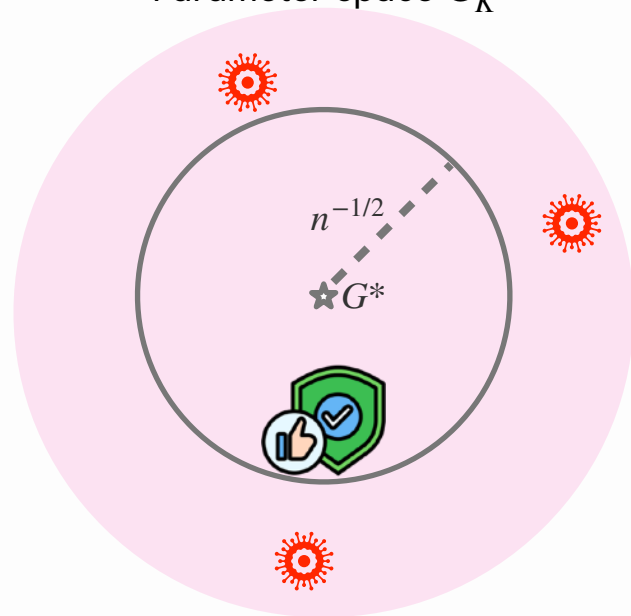
- As both $G, G' \rightarrow G^*$, we have **Asymp. generalized χ^2**

$$nL^2(G, G') \approx \sqrt{n}(G - G')^\top H^* \sqrt{n}(G - G')$$

- For **failure-free machine** estimates



Parameter space \mathbb{G}_K



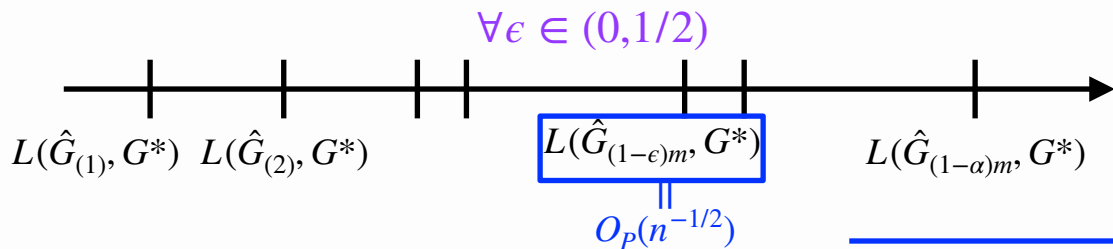
Intuition for our method

The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

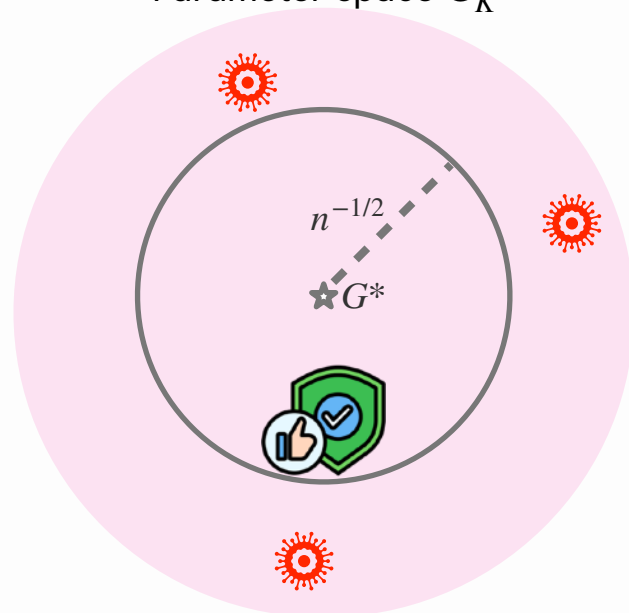
- As both $G, G' \rightarrow G^*$, we have **Asymp. generalized χ^2**

$$nL^2(G, G') \approx \sqrt{n}(G - G')^\top H^* \sqrt{n}(G - G')$$

- For **failure-free machine** estimates



Parameter space \mathbb{G}_K



As $n, \rho_n \rightarrow \infty$

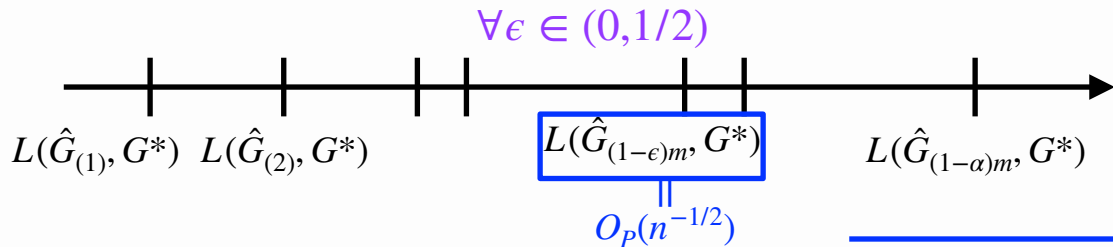
$$P(L(\hat{G}_i, G^*) \geq \rho_n n^{-1/2}) = O(\rho_n^{-8})$$

Intuition for our method

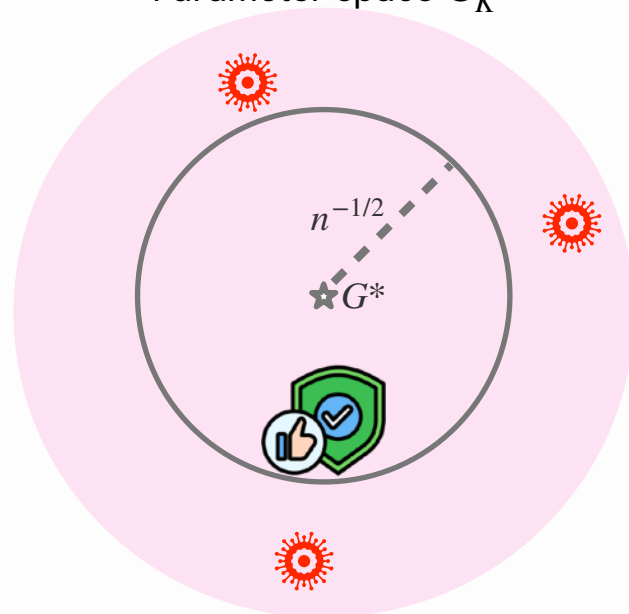
The majority of failure free local estimators are within $O(n^{-1/2})$ distance from G^* in L^2

- As both $G, G' \rightarrow G^*$, we have **Asymp. generalized χ^2**
 $nL^2(G, G') \approx \sqrt{n}(G - G')^\top H^* \sqrt{n}(G - G')$

- For **failure-free machine** estimates



Parameter space \mathbb{G}_K



As $n, \rho_n \rightarrow \infty$

$$P(L(\hat{G}_i, G^*) \geq \rho_n n^{-1/2}) = O(\rho_n^{-8})$$

A slightly inflated ball of radius $O(\rho_n n^{-1/2})$ around a good initial estimate contain almost all of the failure-free local estimates

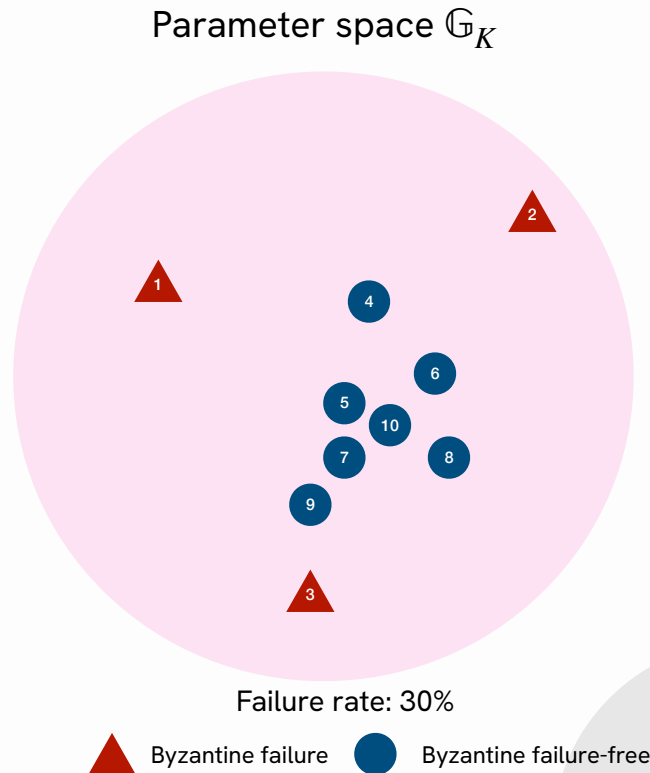
Proposed method: distance filtered mixture reduction

Proposed method: distance filtered mixture reduction

- We pick **Center of attention (COAT)** as the initial estimate

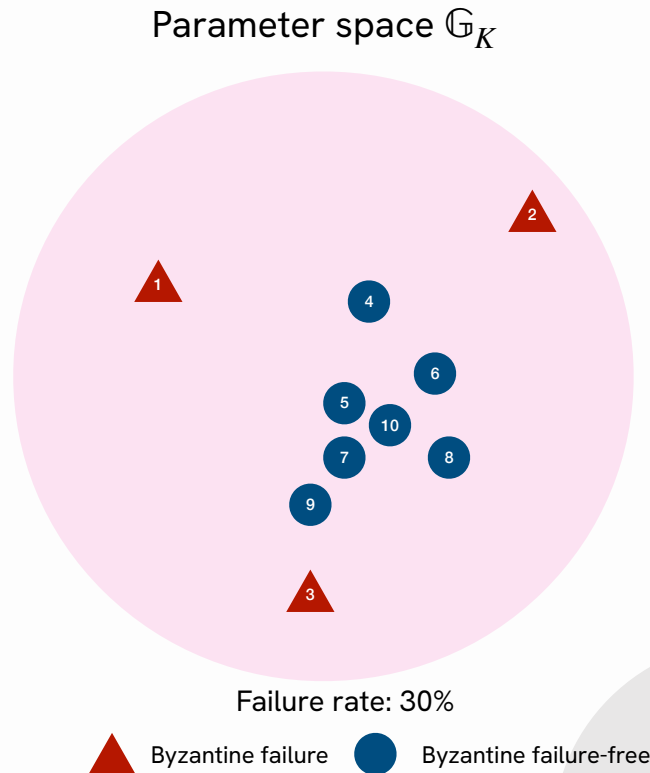
Proposed method: distance filtered mixture reduction

- We pick **Center of attention (COAT)** as the initial estimate



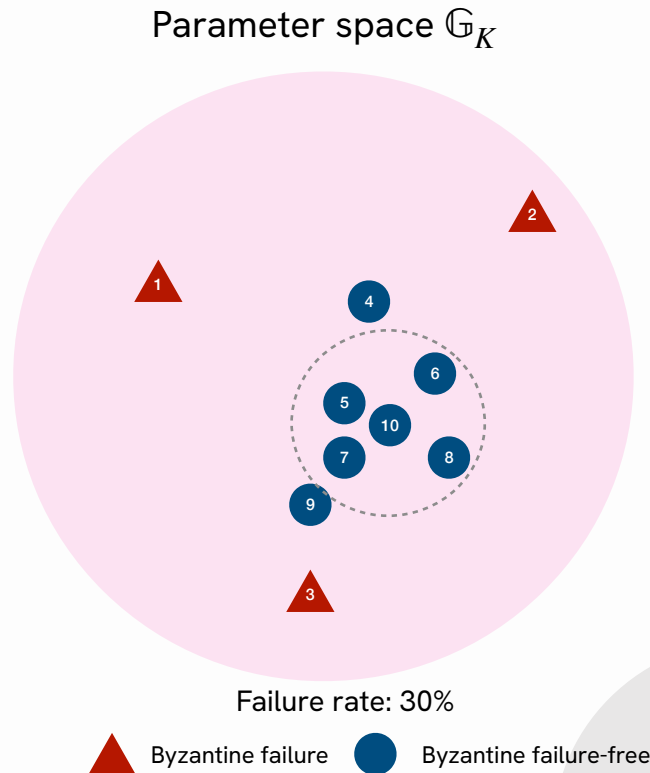
Proposed method: distance filtered mixture reduction

- We pick **Center of attention (COAT)** as the initial estimate
 - The centre of the smallest ball that contains 50% of all local estimates: \hat{G}^{COAT}
 - We denote the corresponding radius as r^{COAT}



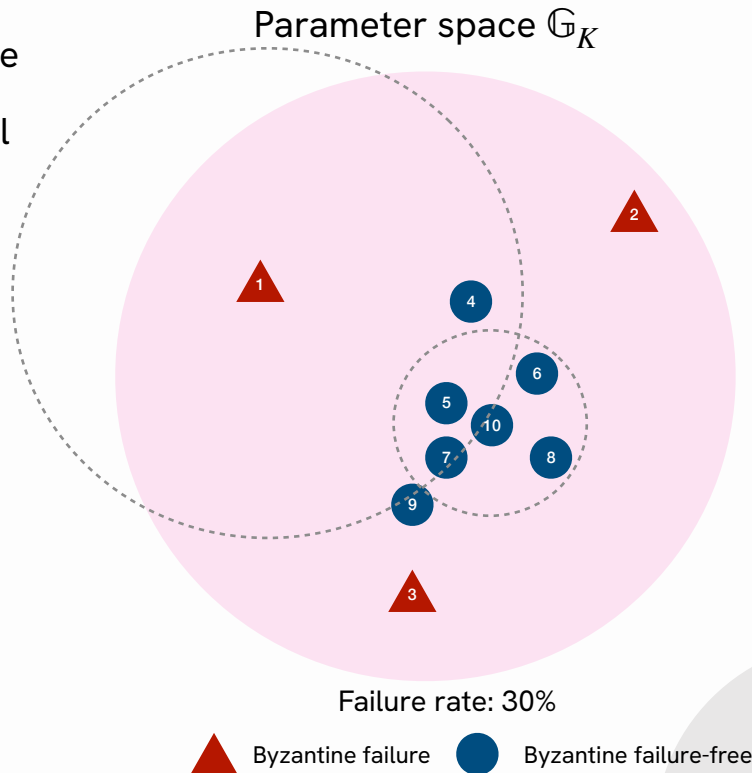
Proposed method: distance filtered mixture reduction

- We pick **Center of attention (COAT)** as the initial estimate
 - The centre of the smallest ball that contains 50% of all local estimates: \hat{G}^{COAT}
 - We denote the corresponding radius as r^{COAT}



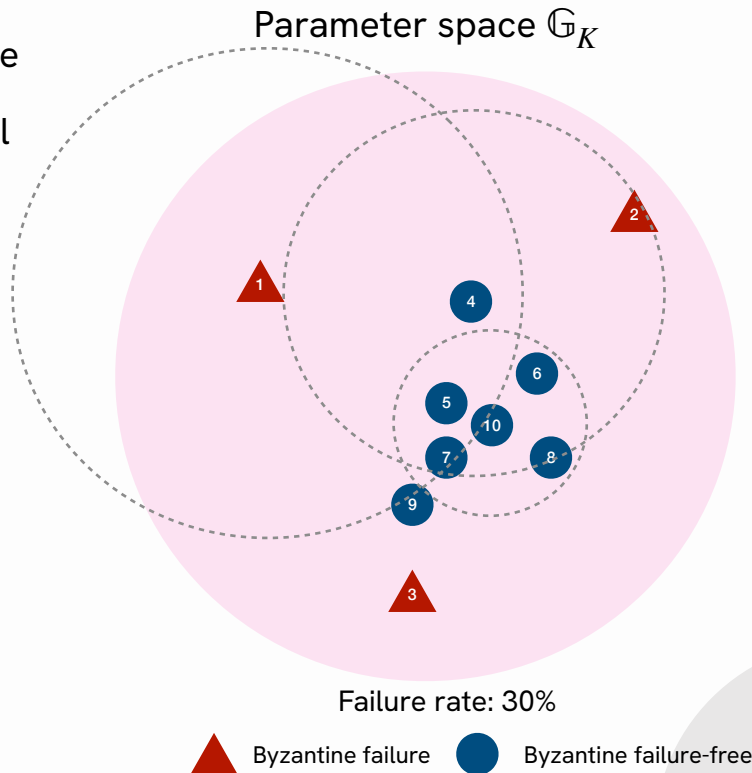
Proposed method: distance filtered mixture reduction

- We pick **Center of attention (COAT)** as the initial estimate
 - The centre of the smallest ball that contains 50% of all local estimates: \hat{G}^{COAT}
 - We denote the corresponding radius as r^{COAT}



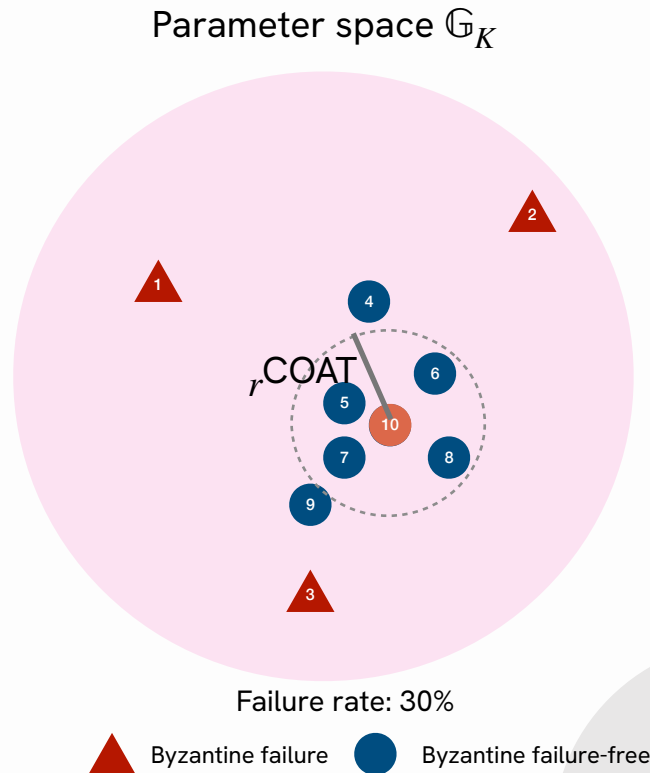
Proposed method: distance filtered mixture reduction

- We pick **Center of attention (COAT)** as the initial estimate
 - The centre of the smallest ball that contains 50% of all local estimates: \hat{G}^{COAT}
 - We denote the corresponding radius as r^{COAT}



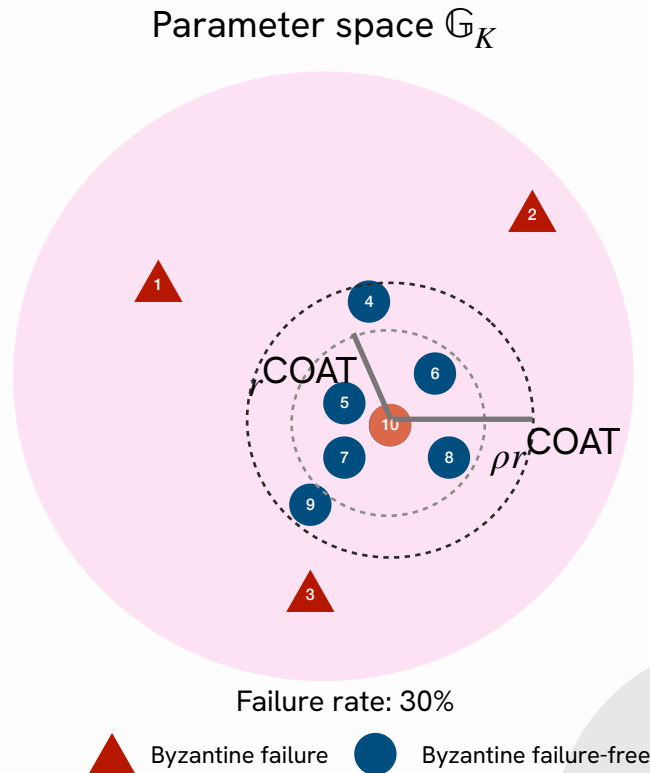
Proposed method: distance filtered mixture reduction

- We pick **Center of attention (COAT)** as the initial estimate
 - The centre of the smallest ball that contains 50% of all local estimates: \hat{G}^{COAT}
 - We denote the corresponding radius as r^{COAT}



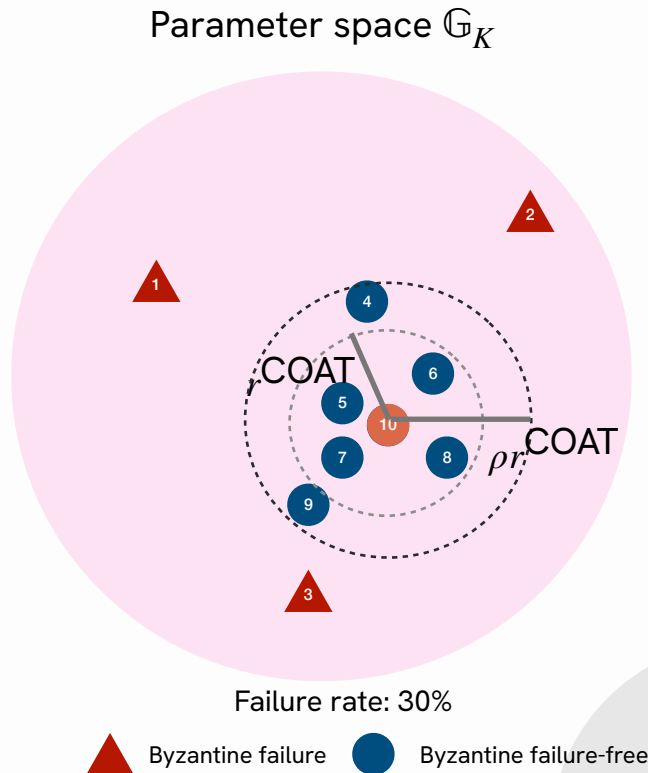
Proposed method: distance filtered mixture reduction

- We pick **Center of attention (COAT)** as the initial estimate
 - The centre of the smallest ball that contains 50% of all local estimates: \hat{G}^{COAT}
 - We denote the corresponding radius as r^{COAT}
- Select $\mathbb{S}_\rho = \{i : L(\hat{G}^{\text{COAT}}, \widetilde{G}_i) \leq \rho r^{\text{COAT}}\}$



Proposed method: distance filtered mixture reduction

- We pick **Center of attention (COAT)** as the initial estimate
 - The centre of the smallest ball that contains 50% of all local estimates: \hat{G}^{COAT}
 - We denote the corresponding radius as r^{COAT}
- Select $\mathbb{S}_\rho = \{i : L(\hat{G}^{\text{COAT}}, \widetilde{G}_i) \leq \rho r^{\text{COAT}}\}$
- Aggregation all local estimates in \mathbb{S}_ρ
- $\rho = 1$: 50% of local estimates are aggregated
- $\rho > 1$: more than 50% local estimates are aggregated



Statistical guarantees

Statistical guarantees

- We establish the theoretical results under some regularity conditions

Statistical guarantees

- We establish the theoretical results under some regularity conditions
- Properties of the initial estimate
 - $L(\hat{G}^{\text{COAT}}, G^*) = O_P(n^{-1/2})$ (mixture density)
 - When strongly identifiable: $\|\hat{G}^{\text{COAT}} - G^*\| = O_P(n^{-1/2})$ (mixing distribution as vector)

Statistical guarantees

- We establish the theoretical results under some regularity conditions
- Properties of the initial estimate
 - $L(\hat{G}^{\text{COAT}}, G^*) = O_P(n^{-1/2})$ (mixture density)
 - When strongly identifiable: $\|\hat{G}^{\text{COAT}} - G^*\| = O_P(n^{-1/2})$ (mixing distribution as vector)
- Properties of DFMR(ρ)
 - When $\rho = \Omega(m^{1/14+\delta})$ for any $\delta > 0$, $n \geq m$, and strongly identifiable,
 $\|\hat{G}^{\text{DFMR}} - G^*\| = O_P(N^{-1/2} + \tilde{\alpha}_m \rho n^{-1/2})$ where $\tilde{\alpha}_m$ the proportion of failure estimates within $2\rho n^{-1/2}$ distance from G^*
 - If $P(L(\xi_i, G^*) \leq r) = O(r^3)$ as $r \rightarrow 0$, we have $\hat{G}^{\text{DFMR}} = \hat{G}^{\text{oracle}} + o_P(N^{-1/2})$

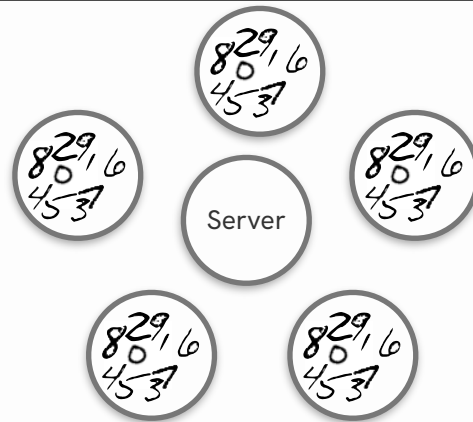
Real data: NIST clustering

Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$

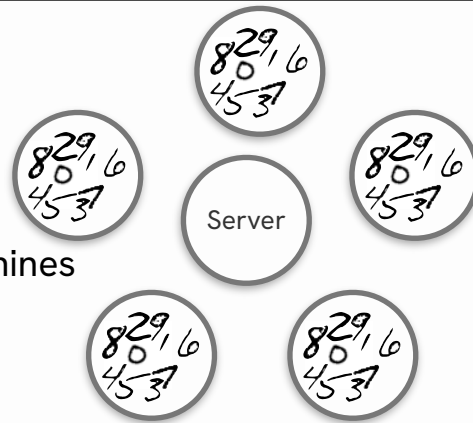
Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$



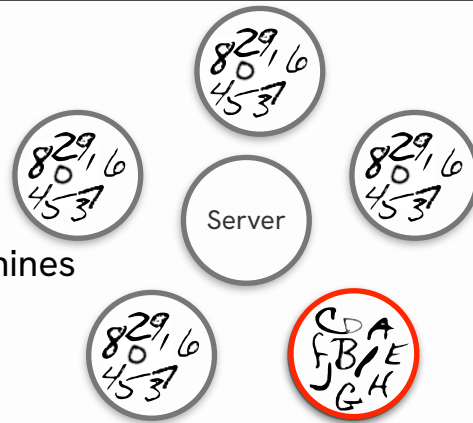
Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$
- Byzantine failures: replace digits features with letter features on failure machines
- Each machine fit a $K = 10$ Gaussian mixture



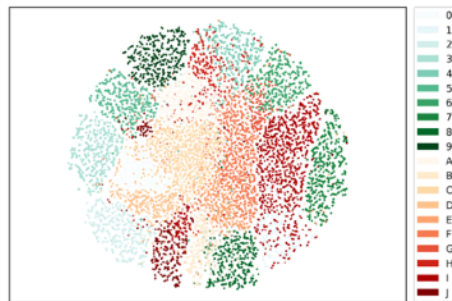
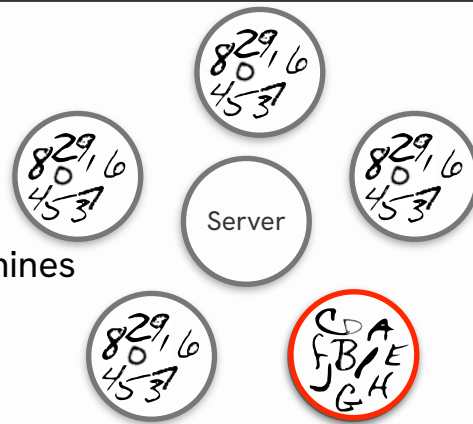
Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$
- Byzantine failures: replace digits features with letter features on failure machines
- Each machine fit a $K = 10$ Gaussian mixture



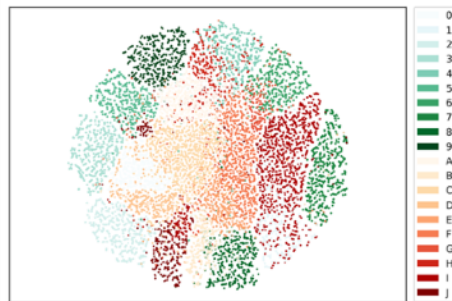
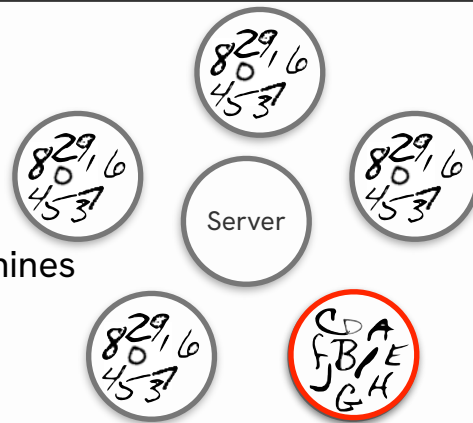
Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$
- Byzantine failures: replace digits features with letter features on failure machines
- Each machine fit a $K = 10$ Gaussian mixture



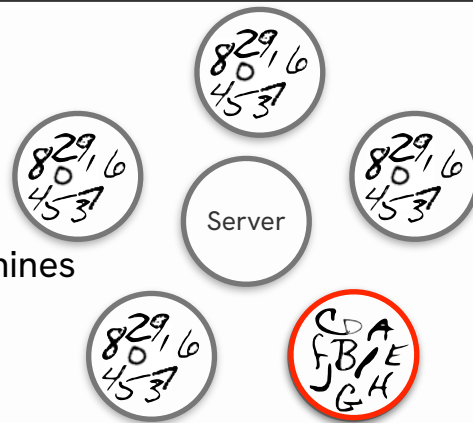
Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$
- Byzantine failures: replace digits features with letter features on failure machines
- Each machine fit a $K = 10$ Gaussian mixture
- Clustering performance: (the higher the better)

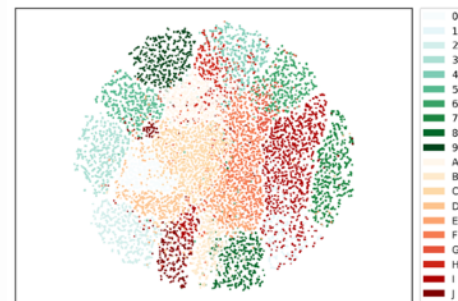


Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$
- Byzantine failures: replace digits features with letter features on failure machines
- Each machine fit a $K = 10$ Gaussian mixture
- Clustering performance: (the higher the better)



α	Oracle	DFMR(ρ)	DFMR(1)	Trim	COAT	Vanilla
0.0	0.9195 (0.0014)	0.9195 (0.0014)	0.9186 (0.0018)	0.9034 (0.0116)	0.8896 (0.0108)	0.9195 (0.0014)
0.1	0.9193 (0.0015)	0.9194 (0.0014)	0.9185 (0.0018)	0.9035 (0.0118)	0.8898 (0.0106)	0.9043 (0.0050)
0.2	0.9192 (0.0015)	0.9194 (0.0013)	0.9186 (0.0020)	0.9042 (0.0112)	0.8898 (0.0106)	0.9046 (0.0044)
0.3	0.9189 (0.0017)	0.9194 (0.0015)	0.9186 (0.0018)	0.9040 (0.0107)	0.8898 (0.0104)	0.9041 (0.0046)
0.4	0.9189 (0.0017)	0.9195 (0.0014)	0.9186 (0.0018)	0.9037 (0.0117)	0.8892 (0.0110)	0.9042 (0.0049)

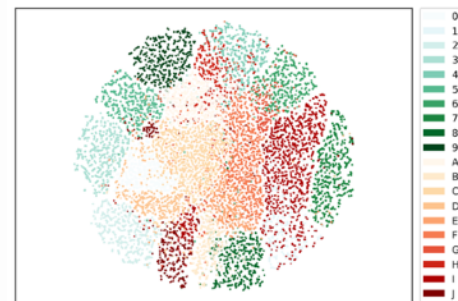
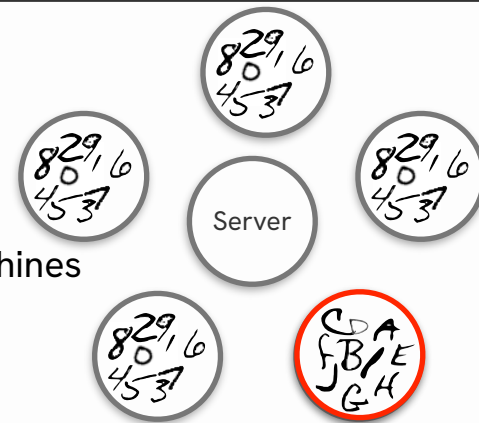


Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$
- Byzantine failures: replace digits features with letter features on failure machines
- Each machine fit a $K = 10$ Gaussian mixture
- Clustering performance: (the higher the better)

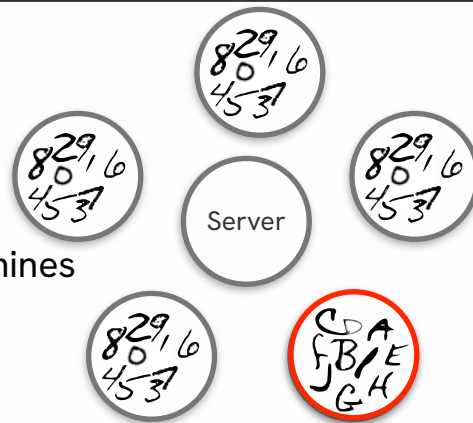
Barrio et al. (2019) Single machine Aggregate all

α	Oracle	DFMR(ρ)	DFMR(1)	Trim	COAT	Vanilla
0.0	0.9195 (0.0014)	0.9195 (0.0014)	0.9186 (0.0018)	0.9034 (0.0116)	0.8896 (0.0108)	0.9195 (0.0014)
0.1	0.9193 (0.0015)	0.9194 (0.0014)	0.9185 (0.0018)	0.9035 (0.0118)	0.8898 (0.0106)	0.9043 (0.0050)
0.2	0.9192 (0.0015)	0.9194 (0.0013)	0.9186 (0.0020)	0.9042 (0.0112)	0.8898 (0.0106)	0.9046 (0.0044)
0.3	0.9189 (0.0017)	0.9194 (0.0015)	0.9186 (0.0018)	0.9040 (0.0107)	0.8898 (0.0104)	0.9041 (0.0046)
0.4	0.9189 (0.0017)	0.9195 (0.0014)	0.9186 (0.0018)	0.9037 (0.0117)	0.8892 (0.0110)	0.9042 (0.0049)

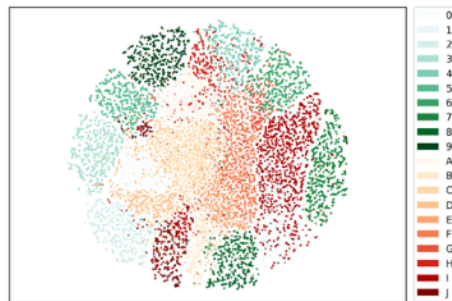


Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$
- Byzantine failures: replace digits features with letter features on failure machines
- Each machine fit a $K = 10$ Gaussian mixture
- Clustering performance: (the higher the better)

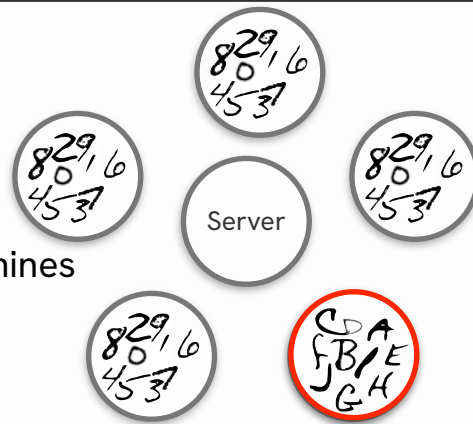


α	Oracle	Our method		Barrio et al. (2019)		Single machine	Aggregate all
		DFMR(ρ)	DFMR(1)	Trim	COAT	Vanilla	
0.0	0.9195 (0.0014)	0.9195 (0.0014)	0.9186 (0.0018)	0.9034 (0.0116)	0.8896 (0.0108)	0.9195 (0.0014)	
0.1	0.9193 (0.0015)	0.9194 (0.0014)	0.9185 (0.0018)	0.9035 (0.0118)	0.8898 (0.0106)	0.9043 (0.0050)	
0.2	0.9192 (0.0015)	0.9194 (0.0013)	0.9186 (0.0020)	0.9042 (0.0112)	0.8898 (0.0106)	0.9046 (0.0044)	
0.3	0.9189 (0.0017)	0.9194 (0.0015)	0.9186 (0.0018)	0.9040 (0.0107)	0.8898 (0.0104)	0.9041 (0.0046)	
0.4	0.9189 (0.0017)	0.9195 (0.0014)	0.9186 (0.0018)	0.9037 (0.0117)	0.8892 (0.0110)	0.9042 (0.0049)	

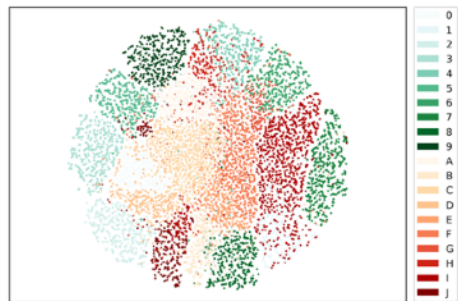


Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$
- Byzantine failures: replace digits features with letter features on failure machines
- Each machine fit a $K = 10$ Gaussian mixture
- Clustering performance: (the higher the better)



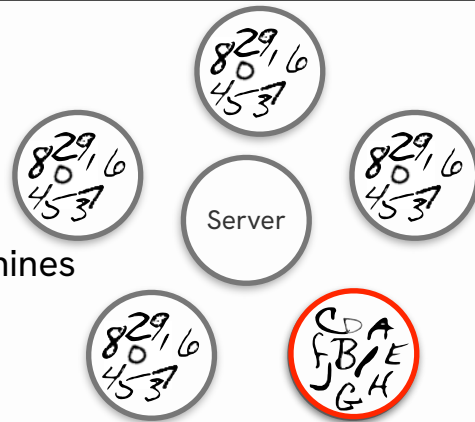
α	Oracle	Our method		Barrio et al. (2019)			Aggregate all	
		DFMR(ρ)	DFMR(1)	Trim	COAT	Vanilla		
0.0	0.9195 (0.0014)	0.9195 (0.0014)	0.9186 (0.0018)	0.9034 (0.0116)	0.8896 (0.0108)	0.9195 (0.0014)		
0.1	0.9193 (0.0015)	0.9194 (0.0014)	0.9185 (0.0018)	0.9035 (0.0118)	0.8898 (0.0106)	0.9043 (0.0050)		
0.2	0.9192 (0.0015)	0.9194 (0.0013)	0.9186 (0.0020)	0.9042 (0.0112)	0.8898 (0.0106)	0.9046 (0.0044)		
0.3	0.9189 (0.0017)	0.9194 (0.0015)	0.9186 (0.0018)	0.9040 (0.0107)	0.8898 (0.0104)	0.9041 (0.0046)		
0.4	0.9189 (0.0017)	0.9195 (0.0014)	0.9186 (0.0018)	0.9037 (0.0117)	0.8892 (0.0110)	0.9042 (0.0049)		



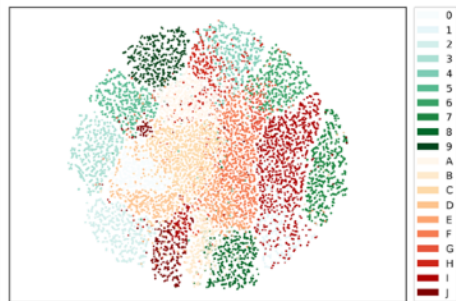
- DFMR(1): select 50% local estimates for aggregation; DFMR(ρ) selects $> 50\%$

Real data: NIST clustering

- Extract image features from pre-trained CNN in $d = 50, m = 50$
- Byzantine failures: replace digits features with letter features on failure machines
- Each machine fit a $K = 10$ Gaussian mixture
- Clustering performance: (the higher the better)



α	Oracle	Our method		Barrio et al. (2019)			Aggregate all	
		DFMR(ρ)	DFMR(1)	Trim	COAT	Vanilla		
0.0	0.9195 (0.0014)	0.9195 (0.0014)	0.9186 (0.0018)	0.9034 (0.0116)	0.8896 (0.0108)	0.9195 (0.0014)		
0.1	0.9193 (0.0015)	0.9194 (0.0014)	0.9185 (0.0018)	0.9035 (0.0118)	0.8898 (0.0106)	0.9043 (0.0050)		
0.2	0.9192 (0.0015)	0.9194 (0.0013)	0.9186 (0.0020)	0.9042 (0.0112)	0.8898 (0.0106)	0.9046 (0.0044)		
0.3	0.9189 (0.0017)	0.9194 (0.0015)	0.9186 (0.0018)	0.9040 (0.0107)	0.8898 (0.0104)	0.9041 (0.0046)		
0.4	0.9189 (0.0017)	0.9195 (0.0014)	0.9186 (0.0018)	0.9037 (0.0117)	0.8892 (0.0110)	0.9042 (0.0049)		



- DFMR(1): select 50% local estimates for aggregation; DFMR(ρ) selects $> 50\%$
- DFMR(ρ) with $\rho \in [1.35, 3]$ is **as good as** the Oracle; DFMR(1) is comparable

Summary

Paper Link



- Distributed learning of finite mixture is difficult due to the well-known “label switching problem”
- The above issue makes existing aggregation approaches and their Byzantine-tolerant inapplicable
- We design the first Byzantine-tolerant aggregation method for distributed learning of finite mixture models
- We demonstrate that DFMR is both computationally efficient and statistically sound.

Summary

- Distributed learning of finite mixture is difficult due to the well-known “label switching problem”
- The above issue makes existing aggregation approaches and their Byzantine-tolerant inapplicable
- We design the first Byzantine-tolerant aggregation method for distributed learning of finite mixture models
- We demonstrate that DFMR is both computationally efficient and statistically sound.

Paper Link

