# Composable Core-sets for Determinant Maximization: A Simple Near-Optimal Algorithm


Piotr Indyk
MIT

**Sepideh Mahabadi**
TTIC

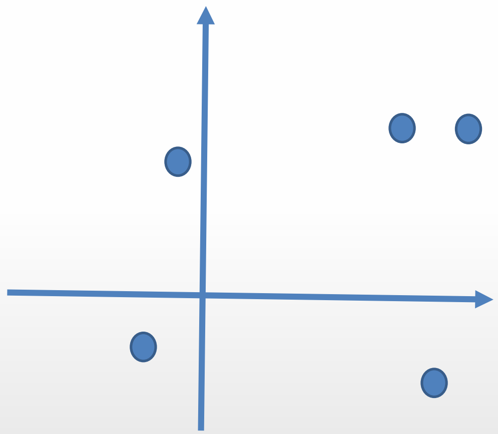
Shayan Oveis Gharan
U. of Washington

Alireza Rezaei
U. of Washington

# Volume (Determinant) Maximization Problem

**Input:** a set of $n$ vectors $\mathrm{V} \in \mathbb{R}^d$ and a parameter $k \leq d$,

$\boldsymbol{k = 2}$



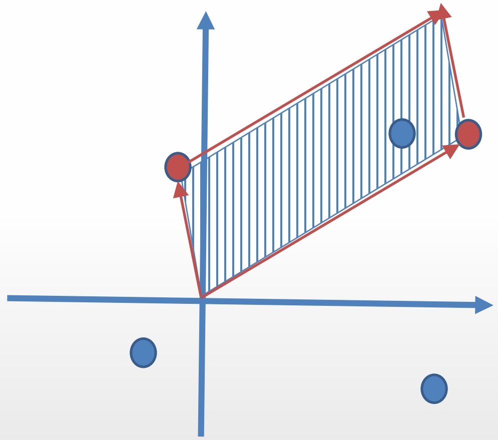
# Volume (Determinant) Maximization Problem

**Input:** a set of $n$ vectors $\mathrm{V} \in \mathbb{R}^d$ and a parameter $k \leq d$,

**Output:** a subset $S \subset V$ of size $k$ with the maximum volume

- Parallelepiped spanned by the points in $S$

$\boldsymbol{k = 2}$



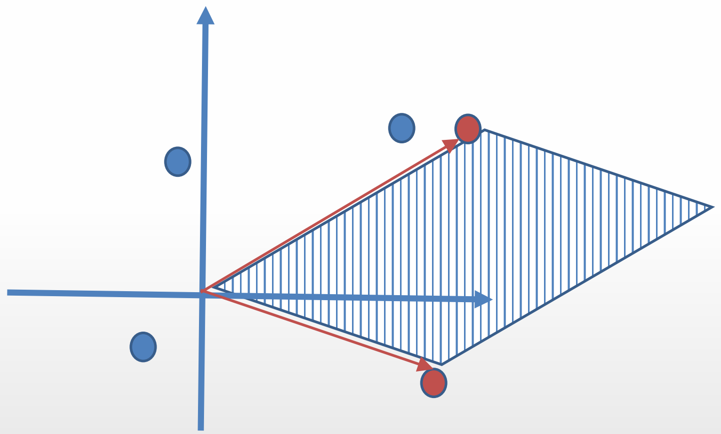
# Volume (Determinant) Maximization Problem

**Input:** a set of $n$ vectors $\mathrm{V} \in \mathbb{R}^d$ and a parameter $k \leq d$,

**Output:** a subset $S \subset V$ of size $k$ with the maximum volume

- Parallelepiped spanned by the points in $S$

$\boldsymbol{k = 2}$



# Determinant (Volume) Maximization Problem

**Input:** a set of $n$ vectors $\mathrm{V} \in \mathbb{R}^d$ and a parameter $k \leq d$,

**Output:** a subset $S \subset V$ of size $k$ with the maximum volume

- Parallelepiped spanned by the points in $S$

$$\begin{bmatrix} v_1 \; v_2 \dots v_n \end{bmatrix}$$

$$\boldsymbol{V}$$

**Equivalent Formulation:**

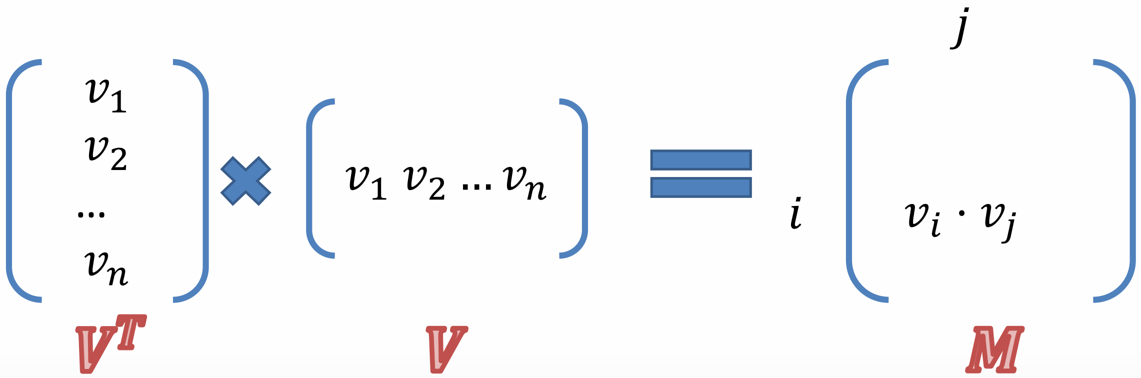
Reuse $V$ to denote the matrix where its columns are the vectors in $V$

# Determinant (Volume) Maximization Problem

**Input:** a set of $n$ vectors $\mathrm{V} \in \mathbb{R}^d$ and a parameter $k \leq d$,

**Output:** a subset $S \subset V$ of size $k$ with the maximum volume

- Parallelepiped spanned by the points in $S$




**Equivalent Formulation:**

Reuse $V$ to denote the matrix where its columns are the vectors in $V$

- Let $M$ be the gram matrix $V^T V$

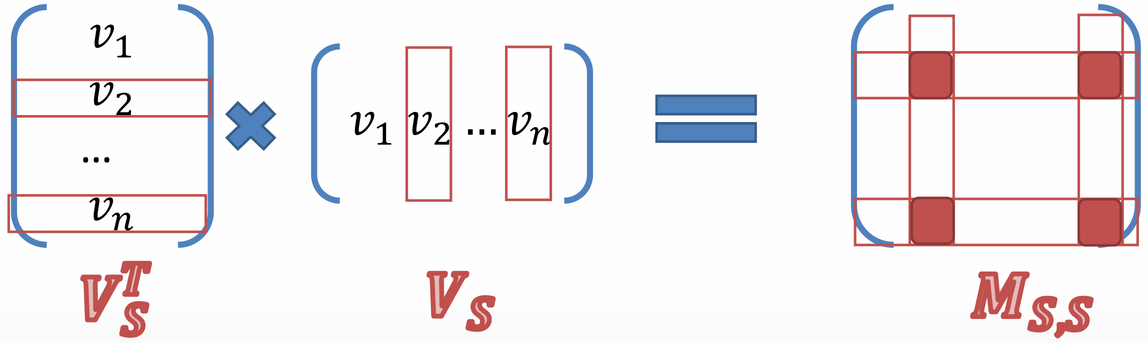
$M_{i,j} = v_i \cdot v_j$

# Determinant (Volume) Maximization Problem

**Input:** a set of $n$ vectors $\mathrm{V} \in \mathbb{R}^d$ and a parameter $k \leq d$,

**Output:** a subset $S \subset V$ of size $k$ with the maximum volume

- Parallelepiped spanned by the points in $S$




**Equivalent Formulation:**

Reuse $V$ to denote the matrix where its columns are the vectors in $V$

- Let $M$ be the gram matrix $V^T V$
- Choose $S$ such that $\det(M_{S,S})$ is maximized

$M_{i,j} = v_i \cdot v_j$

$\det(M_{S,S}) = Vol(S)^2$

# What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI'13]

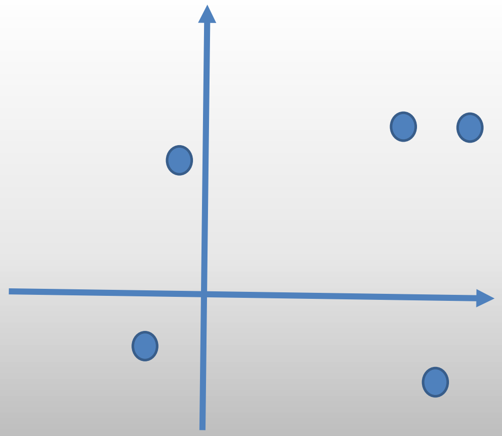
# What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI'13]
- Best algorithm: $e^k$-approximation [Nik'15]

# What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI'13]
- Best algorithm: $e^k$-approximation [Nik'15]
- **Greedy** is a popular algorithm: achieves approximation factor $k!$
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$

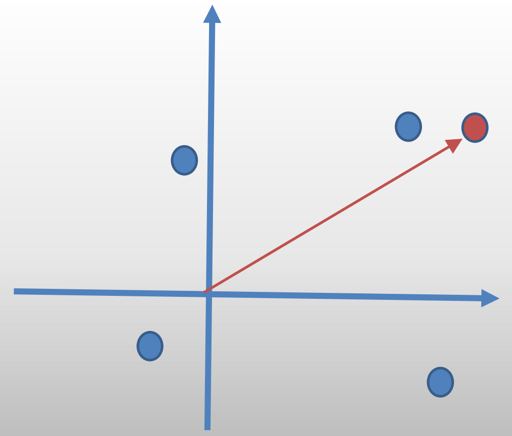
$\boldsymbol{k = 2}$



# What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI'13]
- Best algorithm: $e^k$-approximation [Nik'15]
- **Greedy** is a popular algorithm: achieves approximation factor $k!$
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$

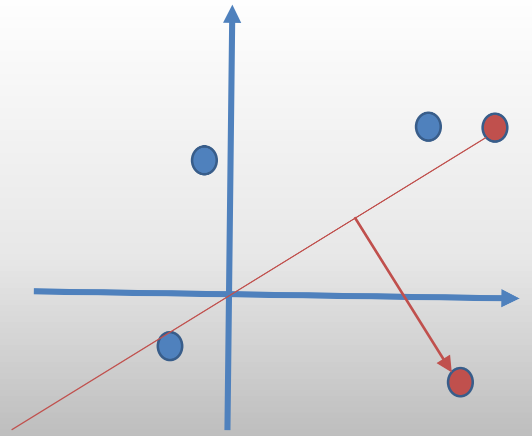
$\boldsymbol{k = 2}$



# What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI'13]
- Best algorithm: $e^k$-approximation [Nik'15]
- **Greedy** is a popular algorithm: achieves approximation factor $k!$
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$

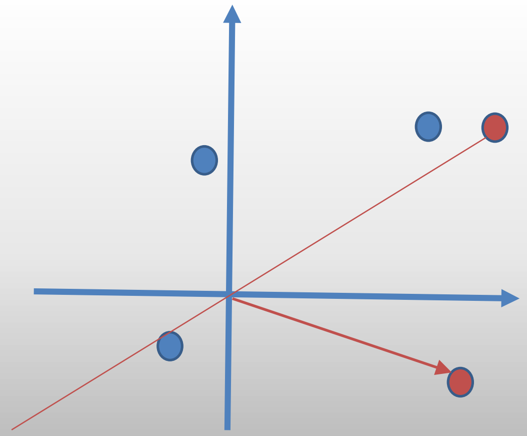
$k = 2$



# What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI'13]
- Best algorithm: $e^{k}$-approximation [Nik'15]
- **Greedy** is a popular algorithm: achieves approximation factor $k!$
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$

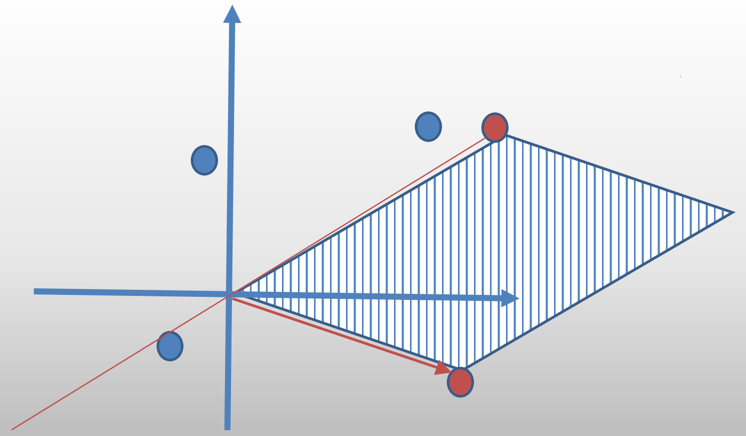
$k = 2$



# What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI'13]
- Best algorithm: $e^{k}$-approximation [Nik'15]
- **Greedy** is a popular algorithm: achieves approximation factor $k!$
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$

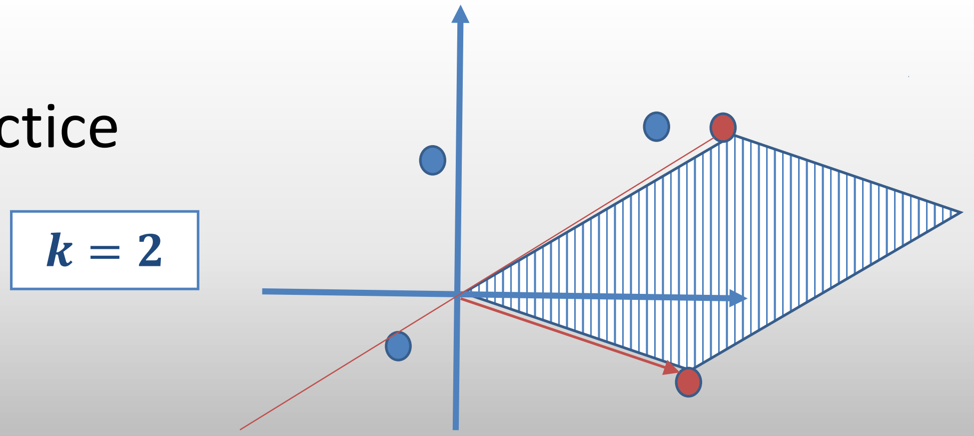
$k = 2$



# What is known?

- Hard to approximate within a factor of $2^{ck}$ [CMI'13]
- Best algorithm: $e^{k}$-approximation [Nik'15]
- **Greedy** is a popular algorithm: achieves approximation factor $k!$
  - $U \leftarrow \emptyset$
  - For $k$ iterations,
    - Add to $U$ the farthest point from the subspace spanned by $U$

- Greedy performs very well in practice



# Determinantal Point Processes (DPP)

**DPP:** Very popular probabilistic model, where given a set of vectors $V$, **samples** any $k$-subset $S$ with probability proportional to this determinant.

- Maximum a posteriori (MAP) decoding is determinant maximization
- Volume/determinant is a notion of **diversity**

- **NeurIPS'18 Tutorial,** *Negative Dependence, Stable Polynomials, and All That*, Jegelka, Sra
- **ICML'19 Workshop,** *Negative Dependence: Theory and Applications in Machine Learning,* Gartrell, Gillenwater, Kulesza, Mariet

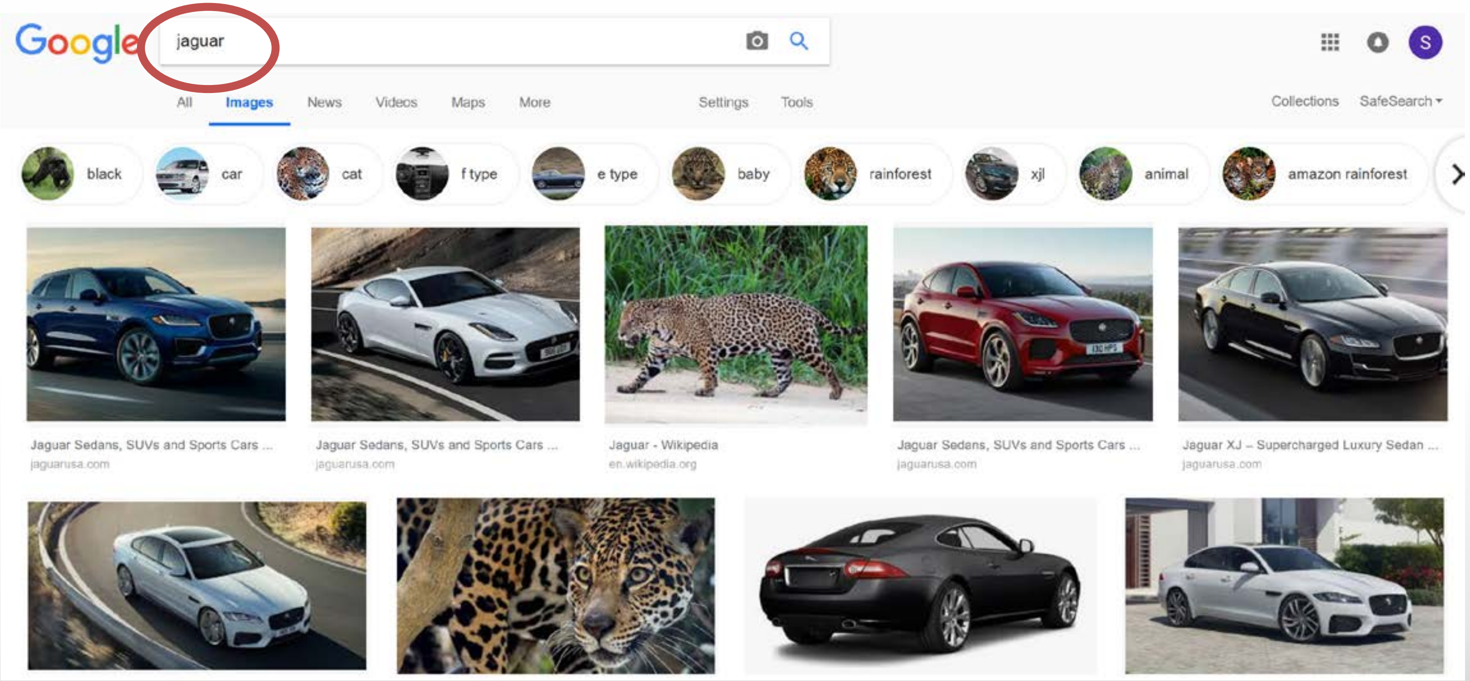
# Application: Diversity Maximization

**Given a set of objects, how to pick a few of them while maximizing diversity?**

# Application: Diversity Maximization

**Given a set of objects, how to pick a few of them while maximizing diversity?**

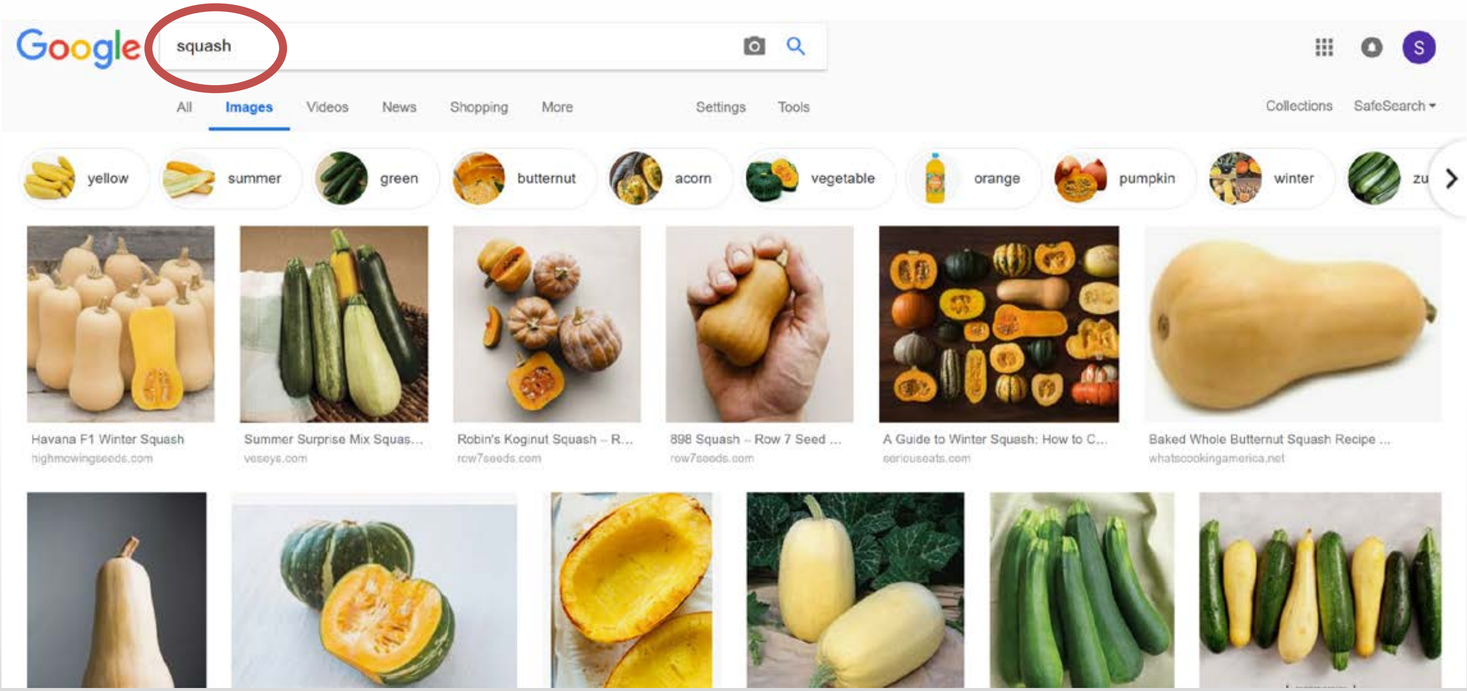
- **Searching**



# Application: Diversity Maximization

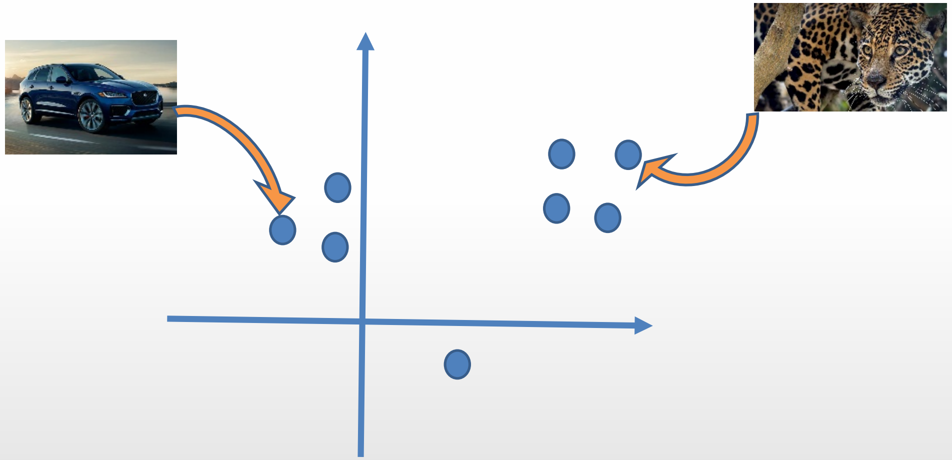
**Given a set of objects, how to pick a few of them while maximizing diversity?**

- **Searching**



# Application: Diversity Maximization

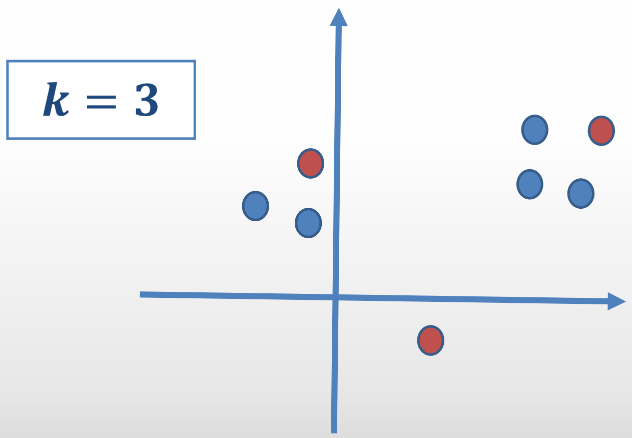
**Objects (documents, images, etc)** → Feature Vectors → **Points in a high dimensional space**



# Application: Diversity Maximization

**Input:** a set of $n$ vectors $V \subset \mathbb{R}^d$ and a parameter $k$,

**Goal:** pick $k$ points while maximizing "diversity".



# Determinantal Point Processes (DPP)

**DPP:** Very popular probabilistic model, where given a set of vectors $V$, **samples** any $k$-subset $S$ with probability proportional to this determinant.

- Maximum a posteriori (MAP) decoding is determinant maximization
- Volume/determinant is a notion of **diversity**

**Applications**

[MJK'17,GCGS'14] Video summarization
[KT+'12, CGGS'15,KT'11] Document summarization
[YFZ+'16] Tweet generation
[LCYO'16] Object detection
….

# Determinantal Point Processes (DPP)

**DPP:** Very popular probabilistic model, where given a set of vectors $V$, **samples** any $k$-subset $S$ with probability proportional to this determinant.

- Maximum a posteriori (MAP) decoding is determinant maximization
- Volume/determinant is a notion of **diversity**

Applications

[MJK'17,GCGS'14] Video summarization
[KT+'12, CGGS'15,KT'11] Document summarization
[YFZ+'16] Tweet generation
[LCYO'16] Object detection
….

- Most applications deal with **massive data**
- Lots of effort for solving the problem in massive data models of computation [MJK'17, WIB'14, PJG+'14, MKSK'13, MKBK'15, MZ'15, MZ'15, BENW'15]
- e.g. streaming, distributed, parallel

# Determinantal Point Processes (DPP)

**DPP:** Very popular probabilistic model, where given a set of vectors $V$, **samples** any $k$-subset $S$ with probability proportional to this determinant.

- Maximum a posteriori (MAP) decoding is determinant maximization
- Volume/determinant is a notion of **diversity**

**Applications**

[MJK'17,GCGS'14] Video summarization
[KT+'12, CGGS'15,KT'11] Document summarization
[YFZ+'16] Tweet generation
[LCYO'16] Object detection
....

- Most applications deal with **massive data**
- Lots of effort for solving the problem in massive data models of computation [MJK'17, WIB'14, PJG+'14, MKSK'13, MKBK'15, MZ'15, MZ'15, BENW'15]
- e.g. streaming, distributed, parallel

**Composable Core-sets**

# Core-sets

**Core-sets** [AHV'05]**:** a subset $U$ of the data $V$ that represents it well

Solving the problem over $U$ gives a good approximation of solving the problem over $V$

# Composable Core-sets

**Core-sets** [AHV'05]**:** a subset $\boldsymbol{U}$ of the data $\boldsymbol{V}$ that represents it well

**Composable Core-sets** [AAI**M**V'13 and I**M**MM'14]**:**

The union of coresets is a coreset for the union

# Composable Core-sets

**Core-sets** [AHV'05]**:** a subset $\boldsymbol{U}$ of the data $\boldsymbol{V}$ that represents it well

**Composable Core-sets** [AAI**M**V'13 and I**M**MM'14]**:**

The union of coresets is a coreset for the union

- Let $\boldsymbol{f}$ be an optimization function
  - E.g. $\boldsymbol{f}(\boldsymbol{V})$: solution to $k$ determinant maximization

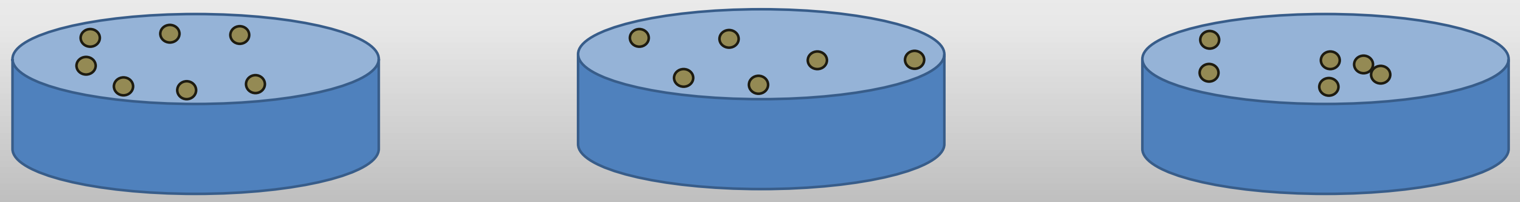
# Composable Core-sets

**Core-sets** [AHV'05]**:** a subset $\boldsymbol{U}$ of the data $\boldsymbol{V}$ that represents it well

**Composable Core-sets** [AAI**M**V'13 and I**M**MM'14]**:**

The union of coresets is a coreset for the union

- Let $\boldsymbol{f}$ be an optimization function
  - E.g. $\boldsymbol{f}(\boldsymbol{V})$: solution to $k$ determinant maximization
- Multiple data sets $\boldsymbol{V_1}, \cdots, \boldsymbol{V_m}$ and their coresets $\boldsymbol{U_1} \subset \boldsymbol{V_1}, \cdots, \boldsymbol{U_m} \subset \boldsymbol{V_m}$,
  - $\boldsymbol{f}(\boldsymbol{U_1} \cup \cdots \cup \boldsymbol{U_m})$ approximates $\boldsymbol{f}(\boldsymbol{V_1} \cup \cdots \cup \boldsymbol{V_m})$ by a factor $\boldsymbol{\alpha}$



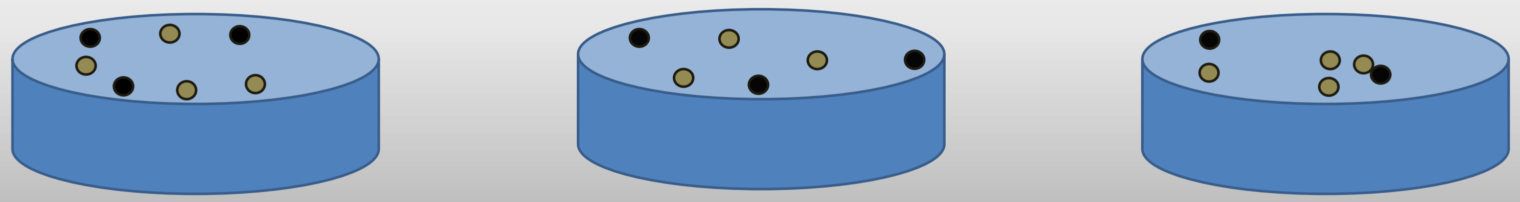
# Composable Core-sets

**Core-sets** [AHV'05]**:** a subset $\boldsymbol{U}$ of the data $\boldsymbol{V}$ that represents it well

**Composable Core-sets** [AAIMV'13 and IMMM'14]**:**

The union of coresets is a coreset for the union

- Let $\boldsymbol{f}$ be an optimization function
  - E.g. $\boldsymbol{f}(\boldsymbol{V})$: solution to $k$ determinant maximization
- Multiple data sets $\boldsymbol{V}_1, \cdots, \boldsymbol{V}_m$ and their coresets $\boldsymbol{U}_1 \subset \boldsymbol{V}_1, \cdots, \boldsymbol{U}_m \subset \boldsymbol{V}_m$,
  - $\boldsymbol{f}(\boldsymbol{U}_1 \cup \cdots \cup \boldsymbol{U}_m)$ approximates $\boldsymbol{f}(\boldsymbol{V}_1 \cup \cdots \cup \boldsymbol{V}_m)$ by a factor $\boldsymbol{\alpha}$



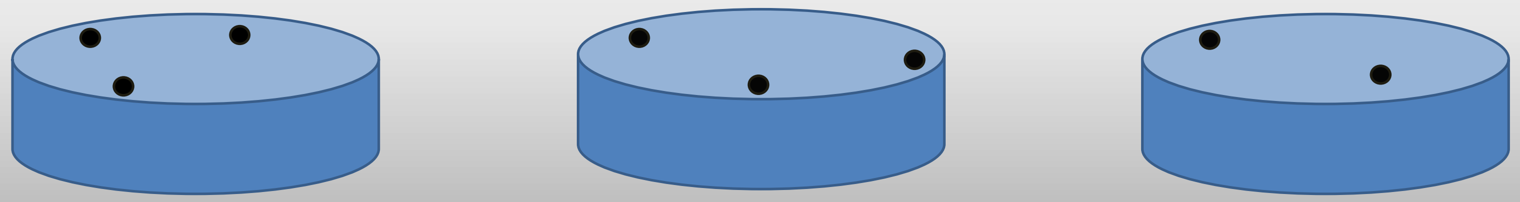
# Composable Core-sets

**Core-sets** [AHV'05]**:** a subset $\boldsymbol{U}$ of the data $\boldsymbol{V}$ that represents it well

**Composable Core-sets** [AAI**M**V'13 and I**M**MM'14]**:**

The union of coresets is a coreset for the union

- Let $\boldsymbol{f}$ be an optimization function
  - E.g. $\boldsymbol{f}(\boldsymbol{V})$: solution to $k$ determinant maximization
- Multiple data sets $\boldsymbol{V_1}, \cdots, \boldsymbol{V_m}$ and their coresets $\boldsymbol{U_1} \subset \boldsymbol{V_1}, \cdots, \boldsymbol{U_m} \subset \boldsymbol{V_m}$,
  - $\boldsymbol{f}(\boldsymbol{U_1} \cup \cdots \cup \boldsymbol{U_m})$ approximates $\boldsymbol{f}(\boldsymbol{V_1} \cup \cdots \cup \boldsymbol{V_m})$ by a factor $\boldsymbol{\alpha}$



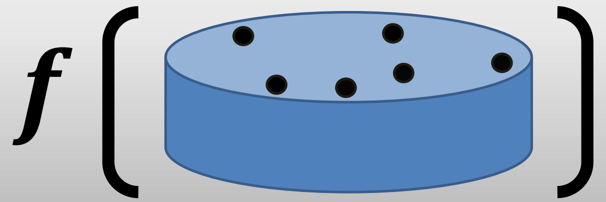
# Composable Core-sets

**Core-sets** [AHV'05]: a subset $\boldsymbol{U}$ of the data $\boldsymbol{V}$ that represents it well

**Composable Core-sets** [AAI**M**V'13 and I**M**MM'14]:

The union of coresets is a coreset for the union

- Let $\boldsymbol{f}$ be an optimization function
  - E.g. $\boldsymbol{f}(\boldsymbol{V})$: solution to $k$ determinant maximization
- Multiple data sets $\boldsymbol{V_1}, \cdots, \boldsymbol{V_m}$ and their coresets $\boldsymbol{U_1} \subset \boldsymbol{V_1}, \cdots, \boldsymbol{U_m} \subset \boldsymbol{V_m}$,
  - $\boldsymbol{f}(\boldsymbol{U_1} \cup \cdots \cup \boldsymbol{U_m})$ approximates $\boldsymbol{f}(\boldsymbol{V_1} \cup \cdots \cup \boldsymbol{V_m})$ by a factor $\boldsymbol{\alpha}$



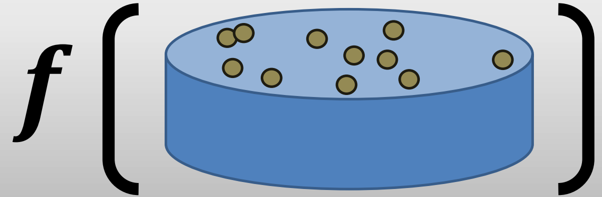
# Composable Core-sets

**Core-sets** [AHV'05]**:** a subset $U$ of the data $V$ that represents it well

**Composable Core-sets** [AAI**M**V'13 and I**M**MM'14]**:**

The union of coresets is a coreset for the union

- Let $f$ be an optimization function
  - E.g. $f(V)$: solution to $k$ determinant maximization
- Multiple data sets $V_1, \cdots, V_m$ and their coresets $U_1 \subset V_1, \cdots, U_m \subset V_m$,
  - $f(U_1 \cup \cdots \cup U_m)$ approximates $f(V_1 \cup \cdots \cup V_m)$ by a factor $\alpha$



# Composable Core-sets

**Core-sets** [AHV'05]**:** a subset $\boldsymbol{U}$ of the data $\boldsymbol{V}$ that represents it well

**Composable Core-sets** [AAI**M**V'13 and I**M**MM'14]**:**

The union of coresets is a coreset for the union

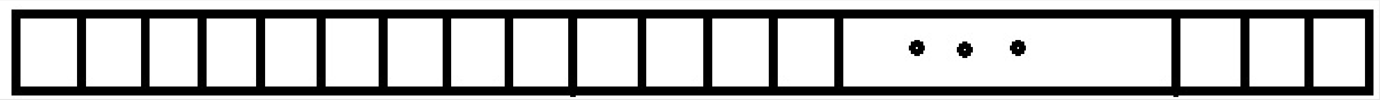
- Let $\boldsymbol{f}$ be an optimization function
  - E.g. $\boldsymbol{f}(\boldsymbol{V})$: solution to $k$ determinant maximization
- Multiple data sets $\boldsymbol{V_1}, \cdots, \boldsymbol{V_m}$ and their coresets $\boldsymbol{U_1} \subset \boldsymbol{V_1}, \cdots, \boldsymbol{U_m} \subset \boldsymbol{V_m}$,
  - $\boldsymbol{f}(\boldsymbol{U_1} \cup \cdots \cup \boldsymbol{U_m})$ approximates $\boldsymbol{f}(\boldsymbol{V_1} \cup \cdots \cup \boldsymbol{V_m})$ by a factor $\boldsymbol{\alpha}$

✓ Composable Core-sets have been studied for the **diversity Maximization** problems, for other notions of diversity: minimum pairwise distance, sum of pairwise distances, etc.

✓ Determinant maximization is a "higher order" notion of diversity

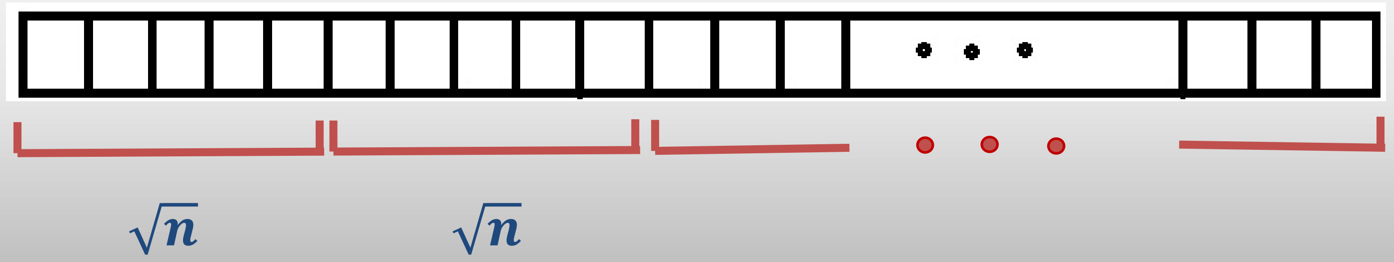
# Applications: Streaming Computation

- **Streaming Computation:**
  - Processing sequence of $n$ data elements “on the fly”
  - limited Storage



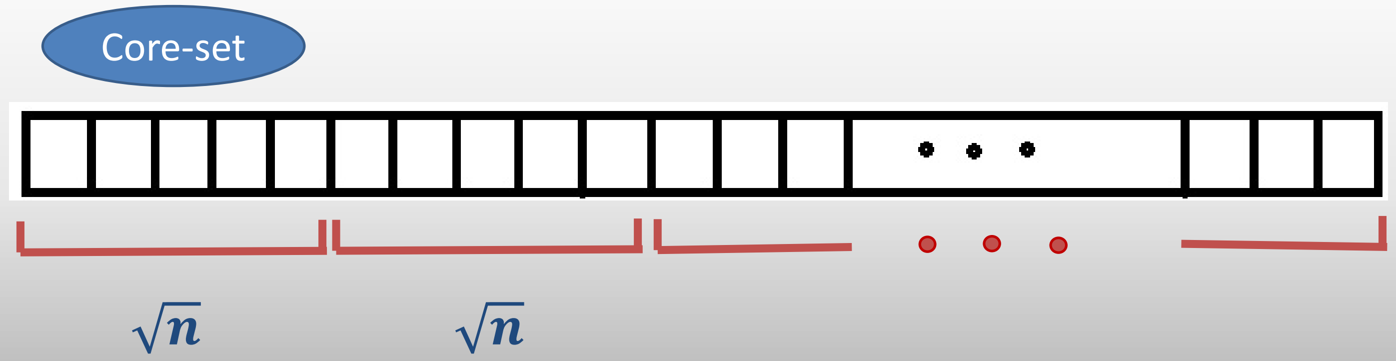
# Applications: Streaming Computation

- **Streaming Computation:**
  - Processing sequence of $n$ data elements “on the fly”
  - limited Storage
- **Composable Core-set**
  - Divide into chunks



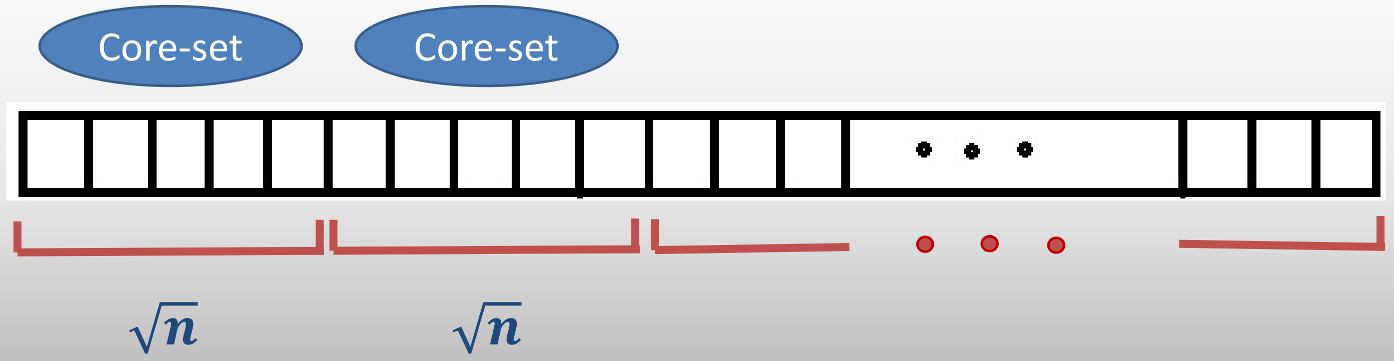
# Applications: Streaming Computation

- **Streaming Computation:**
  - Processing sequence of $n$ data elements "on the fly"
  - limited Storage
- **Composable Core-set**
  - Divide into chunks
  - Compute Core-set for each chunk as it arrives



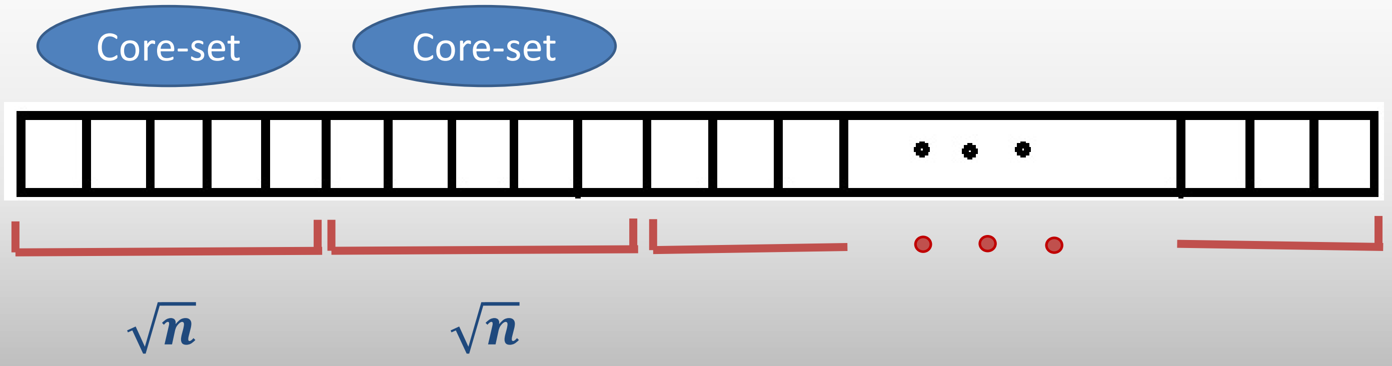
# Applications: Streaming Computation

- **Streaming Computation:**
  - Processing sequence of $n$ data elements "on the fly"
  - limited Storage
- **Composable Core-set**
  - Divide into chunks
  - Compute Core-set for each chunk as it arrives



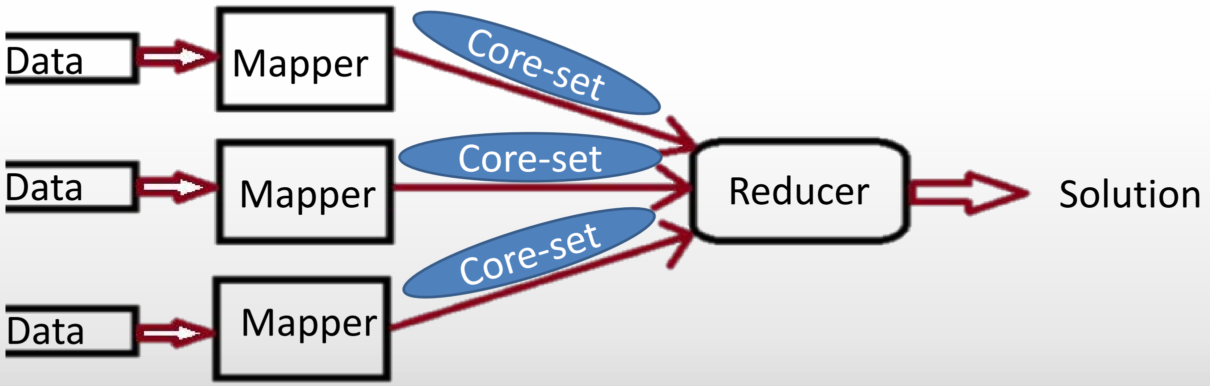
# Applications: Streaming Computation

- **Streaming Computation:**
  - Processing sequence of $n$ data elements "on the fly"
  - limited Storage
- **Composable Core-set**
  - Divide into chunks
  - Compute Core-set for each chunk as it arrives
  - Space goes down from $n$ to $\sqrt{n}$



# Applications: Distributed Computation

- Streaming Computation
- **Distributed System:**
  - Each machine holds a block of data.
  - A composable core-set is computed and sent to the server



# Applications: Improving Runtime

- Streaming Computation
- Distributed System
- **Similar framework for improving the runtime**

Can we get a composable core-set of small size for the determinant maximization problem?

# Composable Core-sets for Volume Maximization

| | | [IMOR'18] |
|---|---|---|
| **Approximation** | | $\tilde{O}(k)^{k/2}$ |
| **Core-set Size** | | $\tilde{O}(k)$ |
| **Simple?** | | × |

**LP-based Optimal Approximation Algorithm of [IMOR'18]:**

There exists a polynomial time algorithm for computing an $\tilde{O}(k)^{k/2}$ -composable core-set of size $\tilde{O}(k)$ for the volume maximization problem.

# Composable Core-sets for Volume Maximization

| | Lower Bound | [IMOR'18] |
|---|---|---|
| **Approximation** | $\Omega(k)^{\frac{k}{2}-o(k)}$ | $\tilde{O}(k)^{\frac{k}{2}}$ |
| **Core-set Size** | $k^{O(1)}$ | $\tilde{O}(k)$ |
| **Simple?** | | × |

**Lower bound [IMOR'18]:**

Any composable core-set of size $k^{O(1)}$ for the volume maximization problem must have an approximation factor of $\Omega(k)^{\frac{k}{2}(1-o(1))}$.

# Our Results

| | Lower Bound | [IMOR'18] | Greedy |
|---|---|---|---|
| Approximation | $\Omega(k)^{\frac{k}{2}-o(k)}$ | $\widetilde{O}(k)^{\frac{k}{2}}$ | $O(C^{k^2})$ |
| Core-set Size | $k^{O(1)}$ | $\widetilde{O}(k)$ | $k$ |
| Simple? | | × | ✓ |

**The widely used Greedy algorithm** produces a composable core-set of size $k$ with approximation factor $O(C^{k^2})$.

# Our Results

| | Lower Bound | [IMOR'18] | Greedy | Local Search |
|---|---|---|---|---|
| Approximation | $\Omega(k)^{\frac{k}{2}-o(k)}$ | $\tilde{O}(k)^{\frac{k}{2}}$ | $O(C^{k^2})$ | $O(k^k)$ |
| Core-set Size | $k^{O(1)}$ | $\tilde{O}(k)$ | $k$ | $k$ |
| Simple? | | × | ✓ | ✓ |

**The Local Search Algorithm** produces a composable core-set of size $k$ with approximation factor $O(k)^{2k}$.

# This Talk

**The Local Search Algorithm** produces a composable core-set of size $k$ with approximation factor $O(k)^k$ for the volume maximization problem.

# This Talk

**The Local Search Algorithm** produces a composable core-set of size $k$ with approximation factor $O(k)^k$ for the volume maximization problem.

**In comparison to the optimal core-set algorithm**

- Approximation $O(k)^{\boldsymbol{k}}$ as opposed to $O(k \log k)^{\boldsymbol{k/2}}$
- Smaller Size $k$ as opposed to $O(k \log k)$
- Simpler to implement (similar to Greedy)
- Better performance in practice

# The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

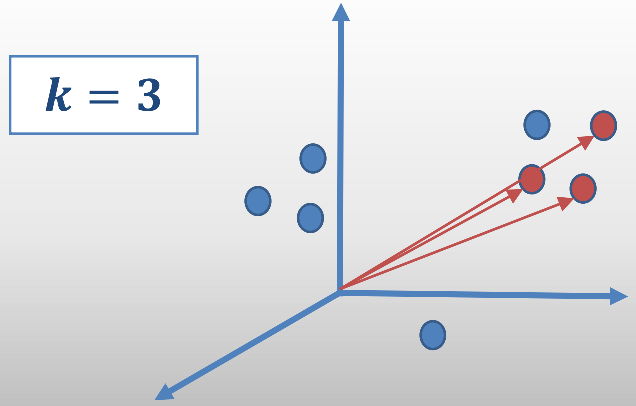
1. Start with an arbitrary subset of $k$ points $S \subseteq V$
2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$

# The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

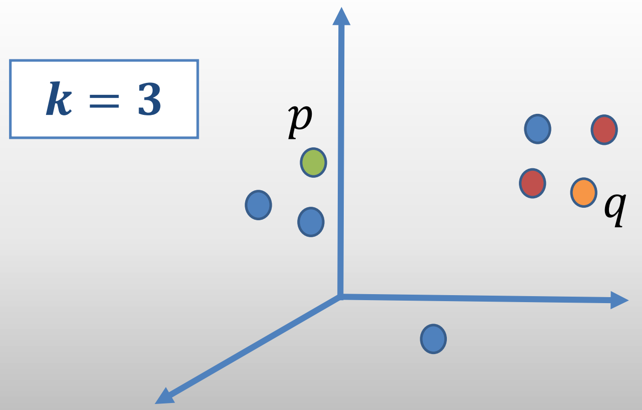
2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$



# The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$
2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$

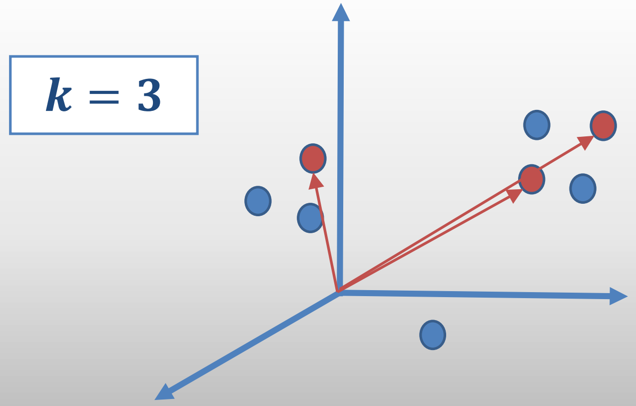


# The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$

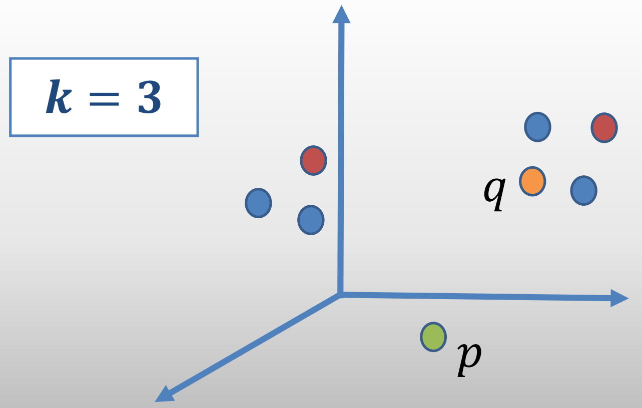


# The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$

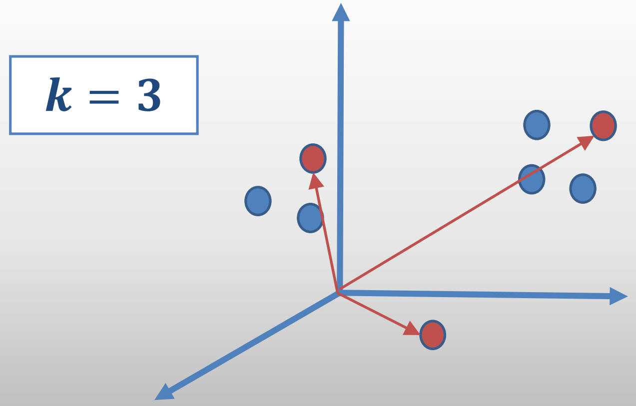


# The Local Search Algorithm

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ increases the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$



# To bound the run time

**Input:** a set $V$ of $n$ points a

Start with a crude approximation (Greedy algorithm)

1. Start with an **arbitrary** subset of $k$ points $S \subseteq V$

2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ **increases** the volume, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$

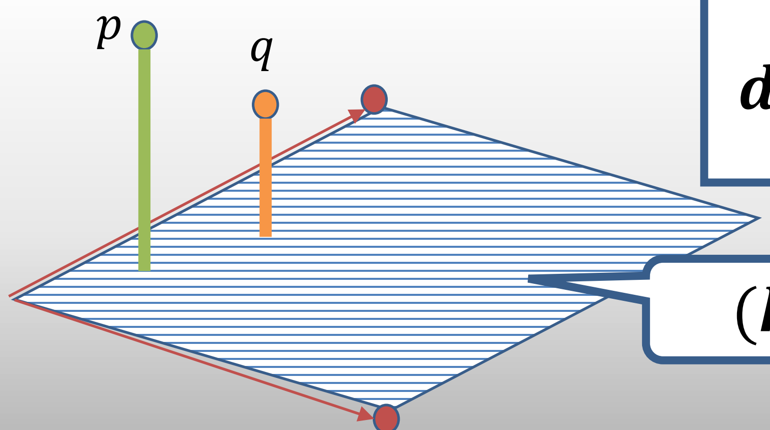
If it increases by at least a factor of $(1 + \epsilon)$

# Checking the condition

**Input:** a set $V$ of $n$ points and a parameter $k$

1. Start with an arbitrary subset of $k$ points $S \subseteq V$
2. While there exists a point $p \in V \setminus S$ and $q \in S$ s.t. replacing $q$ with $p$ **increases the volume**, then swap them, i.e., $S = S \cup \{p\} \setminus \{q\}$

$$\boldsymbol{dist}(\boldsymbol{p}, \boldsymbol{H}_{\boldsymbol{S}\setminus\{\boldsymbol{q}\}}) > \boldsymbol{dist}(\boldsymbol{q}, \boldsymbol{H}_{\boldsymbol{S}\setminus\{\boldsymbol{q}\}})$$




**$(k-1)$-dimensional Subspace**

**Main Lemma [informal]:**

Local Search preserves maximum distance to "all" subspaces of dimension $\boldsymbol{k-1}$

**Main Lemma [informal]:**
Local Search preserves maximum distance to "all" subspaces of dimension $\boldsymbol{k-1}$

- $\boldsymbol{V}$ is the point set
- $\boldsymbol{S} = LS(V)$ is the core-set produced by local search

**Main Lemma [informal]:**
Local Search preserves maximum distance to “all” subspaces of dimension $\boldsymbol{k-1}$

- $\boldsymbol{V}$ is the point set
- $\boldsymbol{S} = LS(V)$ is the core-set produced by local search

**Main Lemma [formal]:**
For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{q \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)$$

**Main Lemma [formal]:**

For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)$$

**Proof.**

- Let $p \in V$ be a point

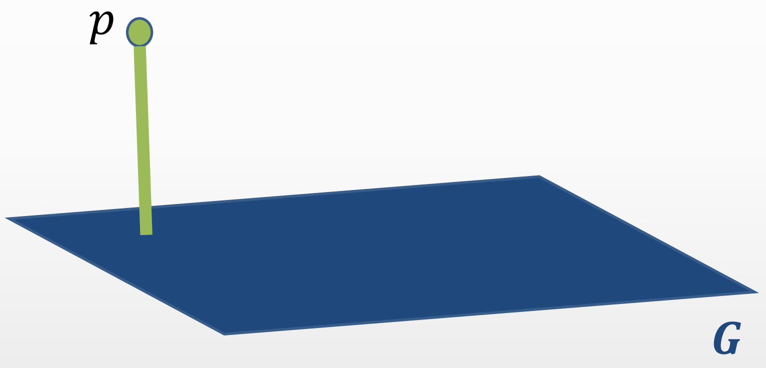
**Main Lemma [formal]:**

For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)$$

**Proof.**

- Let $p \in V$ be a point
- Let $G$ be a $(k-1)$-dimensional subspace.



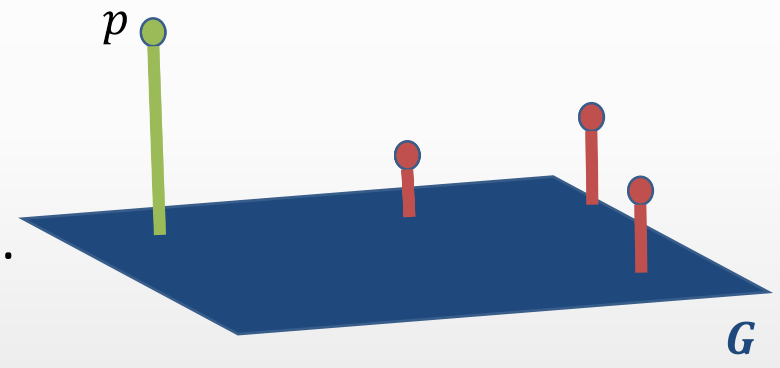
**Main Lemma [formal]:**

For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)$$

**Proof.**

- Let $p \in V$ be a point
- Let $G$ be a $(k-1)$-dimensional subspace.
- Assume for any $q \in S$, $d(q, G) \leq x$



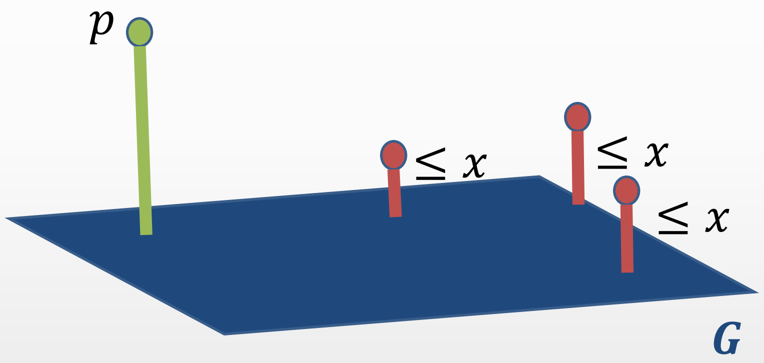
**Main Lemma [formal]:**

For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s\in S} dist(q,G) \geq \frac{1}{2k} \cdot \max_{p\in V} dist(p,G)$$

**Proof.**

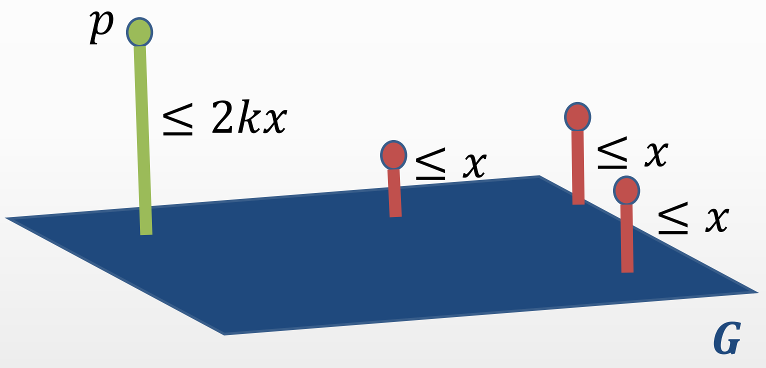
- Let $p \in V$ be a point
- Let $G$ be a $(k-1)$-dimensional subspace.
- Assume for any $q \in S$, $d(q,G) \leq x$



**Main Lemma [formal]:**
For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)$$

**Proof.**



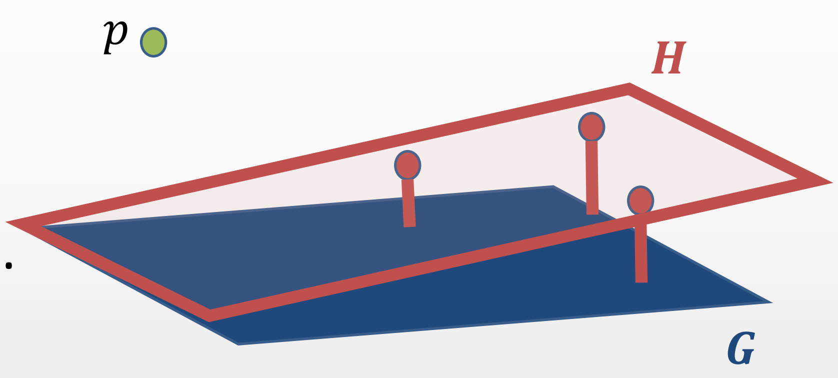

- Let $p \in V$ be a point
- Let $G$ be a $(k-1)$-dimensional subspace.
- Assume for any $q \in S$, $d(q, G) \leq x$
- **Goal:** $\boldsymbol{d(p, G) \leq 2kx}$

**Main Lemma [formal]:**

For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)$$

**Proof.**



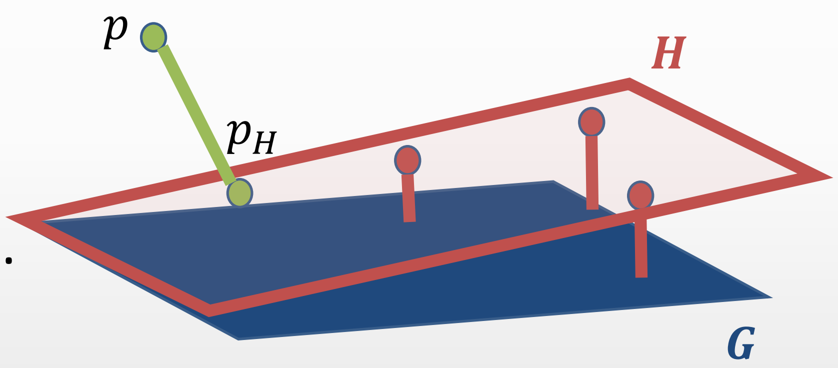

- Let $p \in V$ be a point
- Let $G$ be a $(k-1)$-dimensional subspace.
- Assume for any $q \in S$, $d(q, G) \leq x$
- **Goal:** $\boldsymbol{d(p, G) \leq 2kx}$
- $H := H_S$ be the subspace passing through $S$

**Main Lemma [formal]:**

For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)$$

**Proof.**



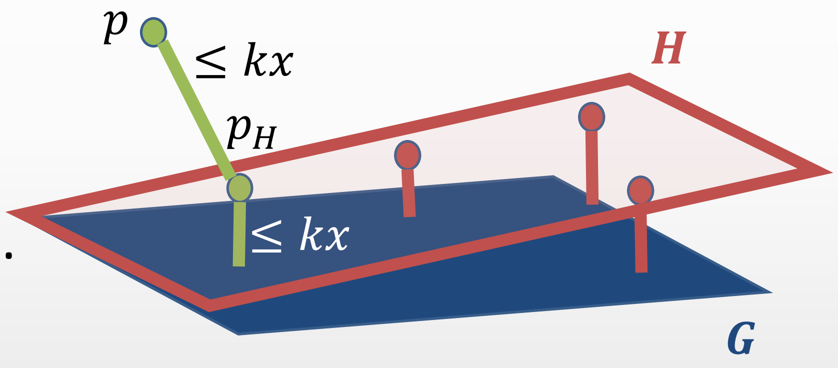

- Let $p \in V$ be a point
- Let $G$ be a $(k-1)$-dimensional subspace.
- Assume for any $q \in S$, $d(q, G) \leq x$
- **Goal:** $\boldsymbol{d(p, G) \leq 2kx}$
- $H := H_S$ be the subspace passing through $S$
- Let $p_H$ be the projection of $p$ onto $G$

**Main Lemma [formal]:**

For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s\in S} dist(q,G) \geq \frac{1}{2k} \cdot \max_{p\in V} dist(p,G)$$

**Proof.**




- Let $p \in V$ be a point
- Let $G$ be a $(k-1)$-dimensional subspace.
- Assume for any $q \in S$, $d(q,G) \leq x$
- **Goal:** $\boldsymbol{d(p,G) \leq 2kx}$
- $H := H_S$ be the subspace passing through $S$
- Let $p_H$ be the projection of $p$ onto $G$

**Lemma 1:** $\boldsymbol{d(p,p_H) \leq kx}$

**Lemma 2:** $\boldsymbol{d(p_H,G) \leq kx}$

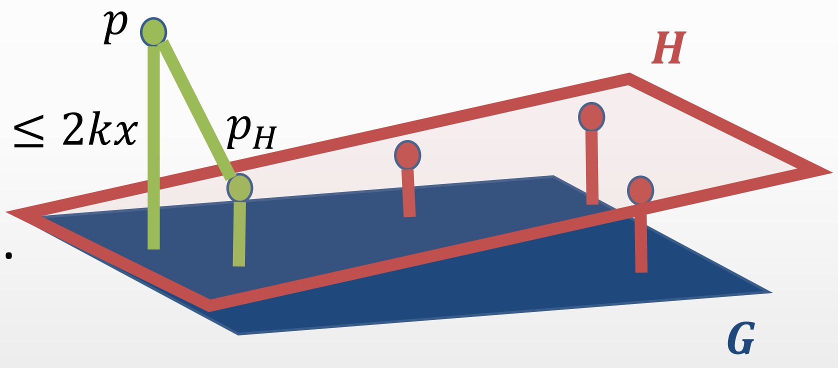
**Main Lemma [formal]:**

For any $(k-1)$-dimensional subspace $G$, the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)$$

**Proof.**

- Let $p \in V$ be a point
- Let $G$ be a $(k-1)$-dimensional subspace.
- Assume for any $q \in S$, $d(q, G) \leq x$
- **Goal:** $\boldsymbol{d(p, G) \leq 2kx}$
- $H := H_S$ be the subspace passing through $S$
- Let $p_H$ be the projection of $p$ onto $G$



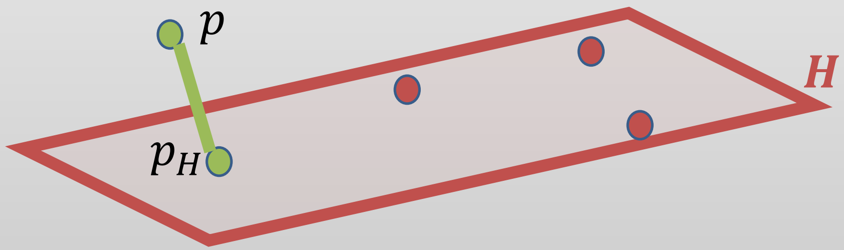

**Lemma 1: $\boldsymbol{d(p, p_H) \leq kx}$**

**Lemma 2: $\boldsymbol{d(p_H, G) \leq kx}$**

# Lemma 2: $\boldsymbol{d(p_H, G) \leq kx}$

**Claim:**

We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

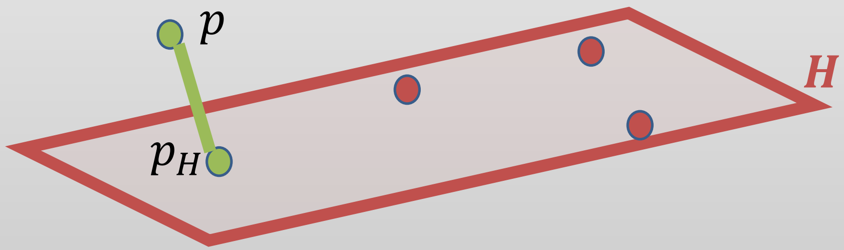


# Lemma 2: $\boldsymbol{d(p_H, G) \leq kx}$

**Claim:**
We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Proof.**

- Since $H$ has dimension $k$, we can write $\boldsymbol{p_H = \sum_{i=1}^{k} \alpha_i q_i}$

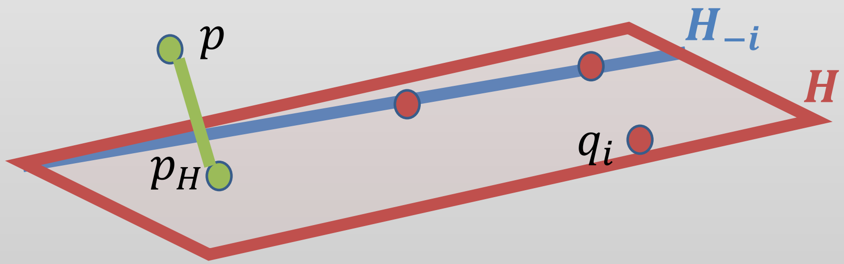


# Lemma 2: $\boldsymbol{d(p_H, G) \le kx}$

**Claim:**
We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \le 1$

**Proof.**

- Since $H$ has dimension $k$, we can write $\boldsymbol{p_H = \sum_{i=1}^{k} \alpha_i q_i}$
- Let $\boldsymbol{H_{-i} \coloneqq H_{S \setminus \{q_i\}}}$

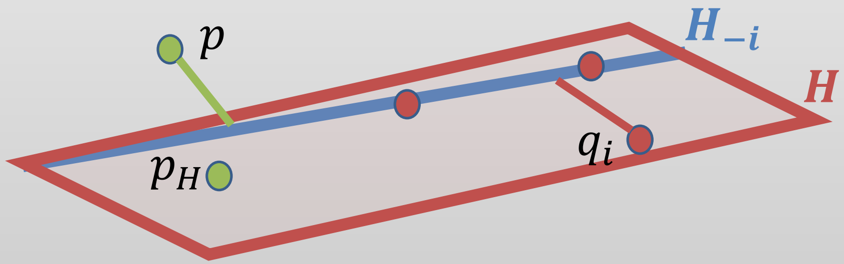


# Lemma 2: $\boldsymbol{d(p_H, G) \leq kx}$

**Claim:**
We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Proof.**

- Since $H$ has dimension $k$, we can write $\boldsymbol{p_H = \sum_{i=1}^{k} \alpha_i q_i}$
- Let $\boldsymbol{H_{-i} \coloneqq H_{S \setminus \{q_i\}}}$
- Since we did not choose $p$ in LS, $\boldsymbol{dist(p, H_{-i}) \leq dist(q_i, H_{-i})}$

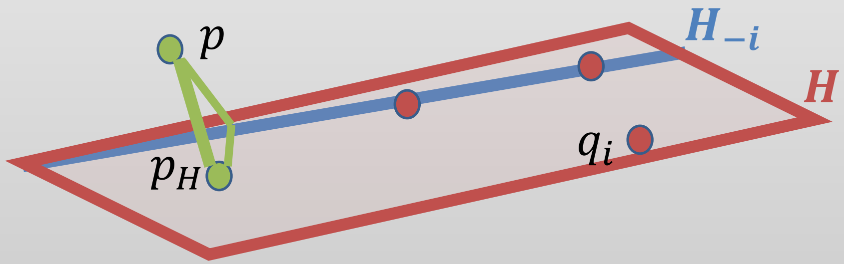


# Lemma 2: $\boldsymbol{d(p_H, G) \leq kx}$

**Claim:**
We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Proof.**

- Since $H$ has dimension $k$, we can write $\boldsymbol{p_H = \sum_{i=1}^{k} \alpha_i q_i}$
- Let $\boldsymbol{H_{-i} := H_{S \setminus \{q_i\}}}$
- Since we did not choose $p$ in LS, $\boldsymbol{dist(p, H_{-i}) \leq dist(q_i, H_{-i})}$
- Since $p_H$ is a projection of $p$ onto $H$, $\boldsymbol{dist(p_H, H_{-i}) \leq dist(p, H_{-i})}$
-

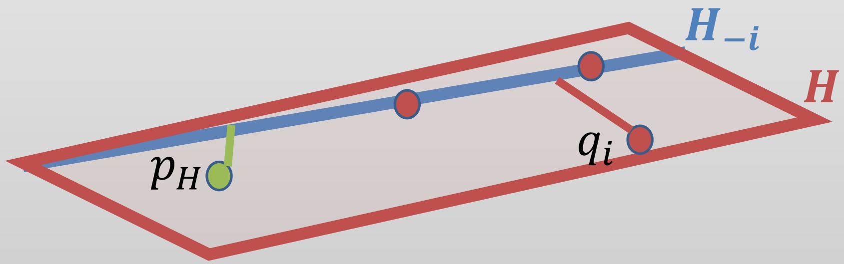


# Lemma 2: $\boldsymbol{d(p_H, G) \leq kx}$

**Claim:**
We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Proof.**

- Since $H$ has dimension $k$, we can write $\boldsymbol{p_H = \sum_{i=1}^{k} \alpha_i q_i}$
- Let $\boldsymbol{H_{-i} := H_{S\setminus\{q_i\}}}$
- Since we did not choose $p$ in LS, $\boldsymbol{dist(p, H_{-i}) \leq dist(q_i, H_{-i})}$
- Since $p_H$ is a projection of $p$ onto $H$, $\boldsymbol{dist(p_H, H_{-i}) \leq dist(p, H_{-i})}$
- Thus $\boldsymbol{dist(p_H, H_{-i}) \leq dist(q_i, H_{-i})}$

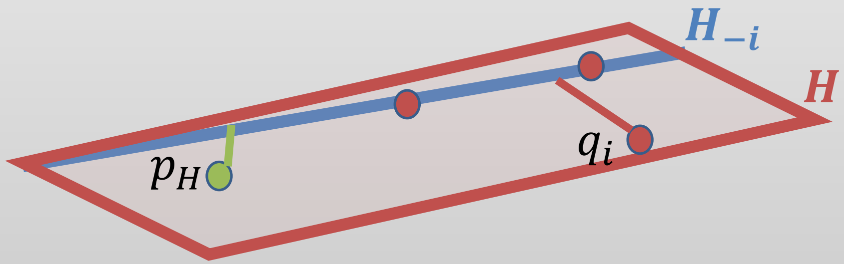


# Lemma 2: $\boldsymbol{d(p_H, G) \le kx}$

**Claim:**
We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \le 1$

**Proof.**

- Since $H$ has dimension $k$, we can write $\boldsymbol{p_H = \sum_{i=1}^{k} \alpha_i q_i}$
- Let $\boldsymbol{H_{-i} \coloneqq H_{S \setminus \{q_i\}}}$
- Since we did not choose $p$ in LS, $\boldsymbol{dist(p, H_{-i}) \le dist(q_i, H_{-i})}$
- Since $p_H$ is a projection of $p$ onto $H$, $\boldsymbol{dist(p_H, H_{-i}) \le dist(p, H_{-i})}$
- Thus $\boldsymbol{dist(p_H, H_{-i}) \le dist(q_i, H_{-i})}$
- **Thus** $\boldsymbol{|\alpha_i| \le 1}$



**Claim:**

We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Claim:**

We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Assumption:** $dist(q_i, G) \leq x$

**Claim:**

We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Assumption:** $dist(q_i, G) \leq x$

**Lemma2:** $dist(p_H, G) \leq \sum_{i=1}^{k} \alpha_i dist(q_i, G) \leq k \cdot x \leq kx$

**Claim:**

We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Assumption:** $dist(q_i, G) \leq x$

**Lemma2:** $dist(p_H, G) \leq \sum_{i=1}^{k} \alpha_i dist(q_i, G) \leq k \cdot x \leq kx$

**Lemma 1:** $d(p, p_H) \leq kx$

**Claim:**

We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Assumption:** $dist(q_i, G) \leq x$

**Lemma2:** $dist(p_H, G) \leq \sum_{i=1}^{k} \alpha_i dist(q_i, G) \leq k \cdot x \leq kx$

**Lemma 1:** $d(p, p_H) \leq kx$

**Goal:** $d(p, G) \leq 2kx$

**Claim:**
We can write $p_H = \sum_{i=1}^{k} \alpha_i q_i$ s.t. all $|\alpha_i| \leq 1$

**Assumption:** $dist(q_i, G) \leq x$

**Lemma2:** $dist(p_H, G) \leq \sum_{i=1}^{k} \alpha_i dist(q_i, G) \leq k \cdot x \leq kx$

**Lemma 1:** $d(p, p_H) \leq kx$

**Goal:** $d(p, G) \leq 2kx$

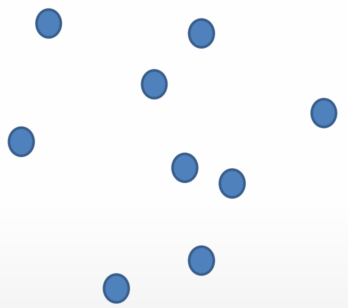
**Main Lemma [formal]:**
For any $(k-1)$-dimensional subspace $G,$ the maximum distance of the point set to $G$ is approximately preserved

$$\max_{s \in S} dist(q, G) \geq \frac{1}{2k} \cdot \max_{p \in V} dist(p, G)$$

**Main Theorem**

Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

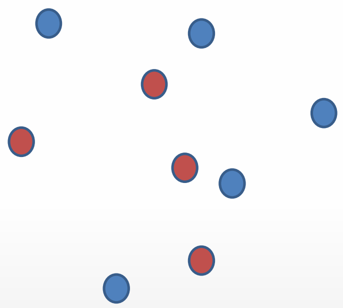


**Main Theorem**

Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets



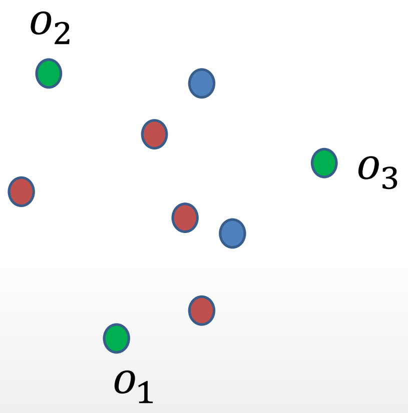
**Main Theorem**

Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume



## Main Theorem

# Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

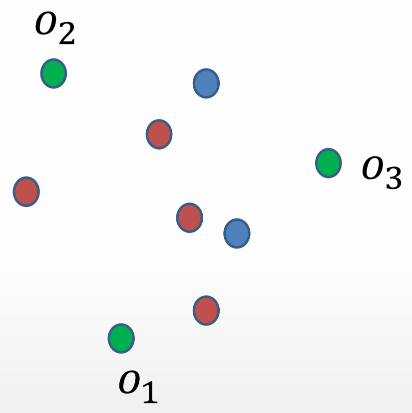
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1\ to\ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

# Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

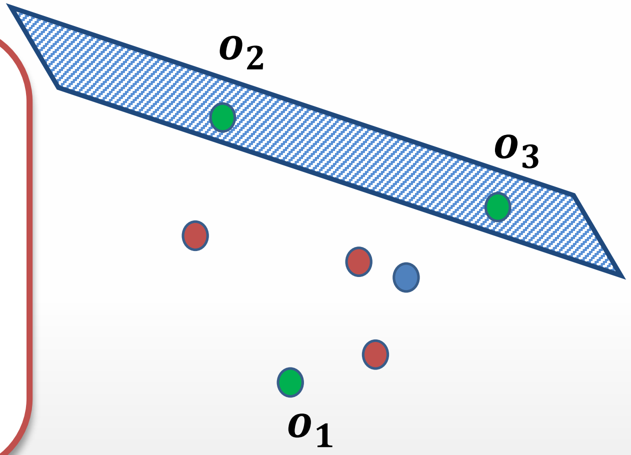
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1\ to\ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



**Main Theorem**

Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

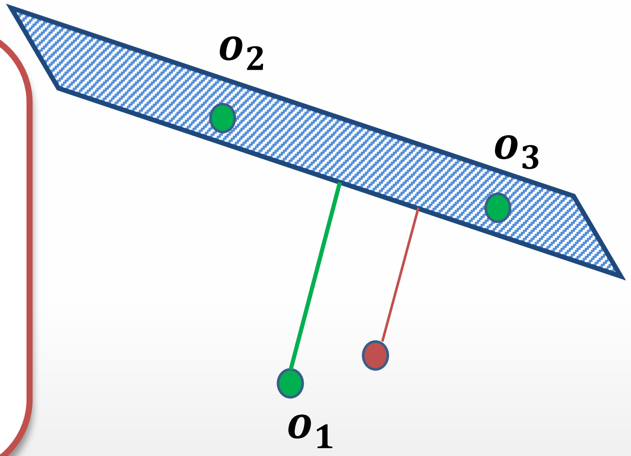
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1\ to\ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

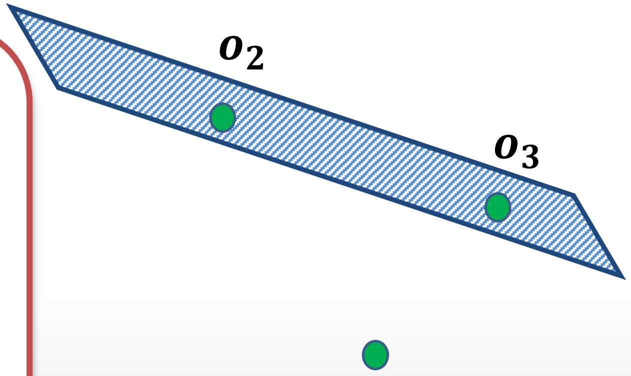
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1 \ to\ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

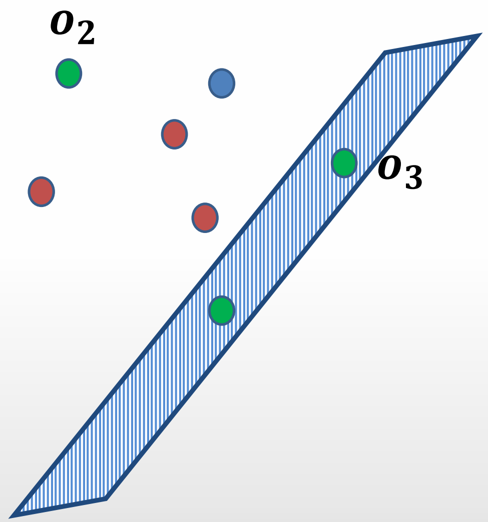
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1$ $to$ $k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

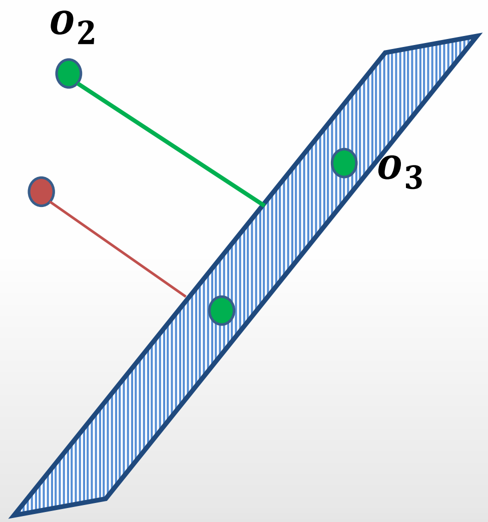
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1 \; to \; k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

# Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

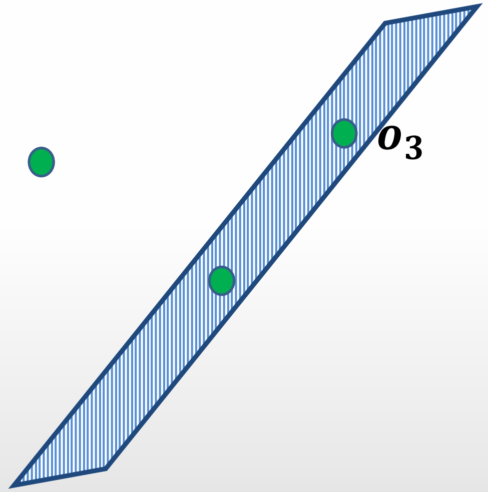
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1, \dots, o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1\ to\ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



**Main Theorem**

Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

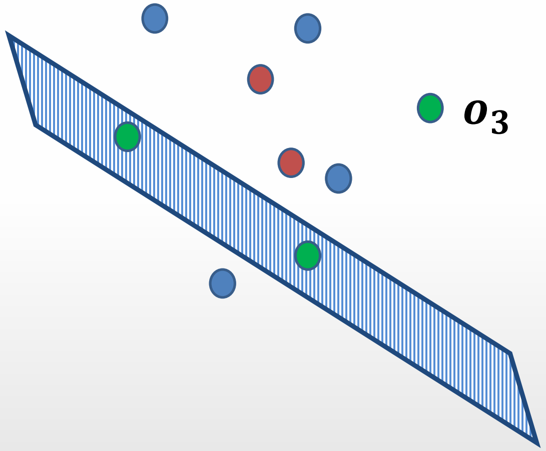
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1$ $to$ $k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

# Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \cup_i V_i$ be the union of the point sets

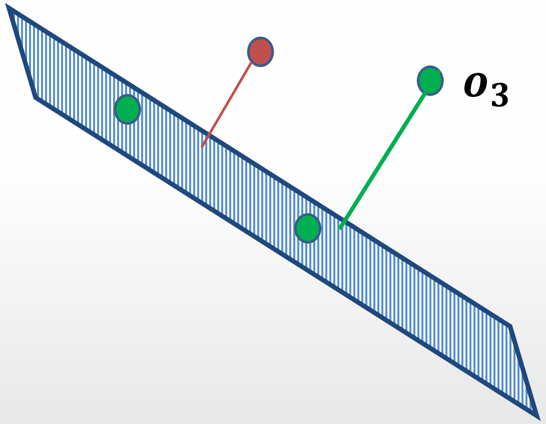
Let $\boldsymbol{S} = \cup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \ldots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1\ to\ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

# Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

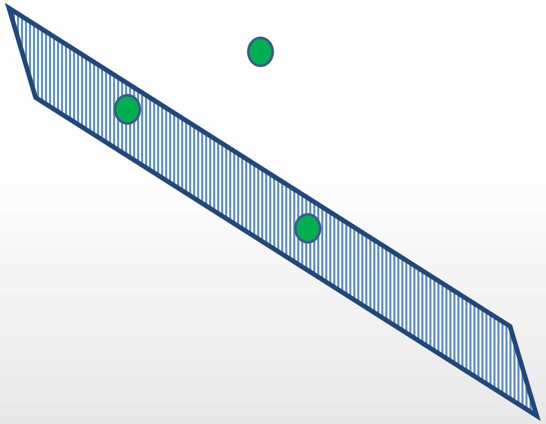
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1 \ to \ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

# Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

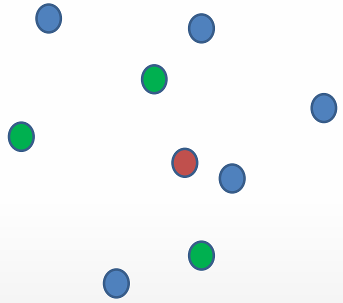
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1\ to\ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \bigcup_i V_i$ be the union of the point sets

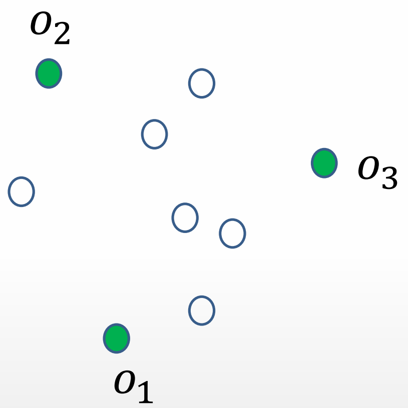
Let $\boldsymbol{S} = \bigcup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \dots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1\ to\ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

# Local Search produces a core-set for volume maximization

Let $\boldsymbol{V} = \cup_i V_i$ be the union of the point sets

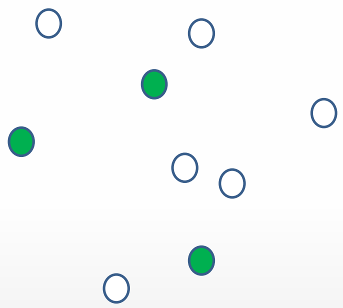
Let $\boldsymbol{S} = \cup_i S_i$ be the union of core-sets

Let $\boldsymbol{Opt} = \{\boldsymbol{o_1}, \ldots, \boldsymbol{o_k}\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1\ to\ k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



## Main Theorem

# Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, \dots, o_k\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1 \; to \; k$

- Let $q_i \in S$ be the point that is fa
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$

Since local search preserve maximum distances to subspaces

➢ Lose a factor of at most $2k$ at each iteration

## Main Theorem

# Local Search produces a core-set for volume maximization

Let $V = \bigcup_i V_i$ be the union of the point sets

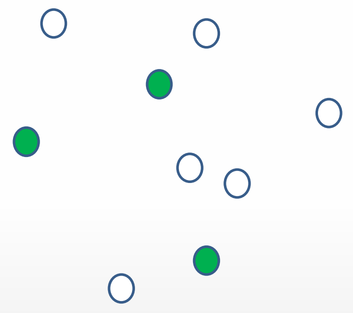
Let $S = \bigcup_i S_i$ be the union of core-sets

Let $Opt = \{o_1, \ldots, o_k\} \subset V$ be the optimal subset of points maximizing the volume

$Sol \leftarrow Opt$

For $i = 1$ $to$ $k$

- Let $q_i \in S$ be the point that is farthest away from $H_{Sol \setminus \{o_i\}}$
- $Sol \leftarrow Sol \cup \{q_i\} \setminus \{o_i\}$



- Lose a factor of at most $2k$ at each iteration
- Total approximation factor $(2k)^k$

# Empirical Results

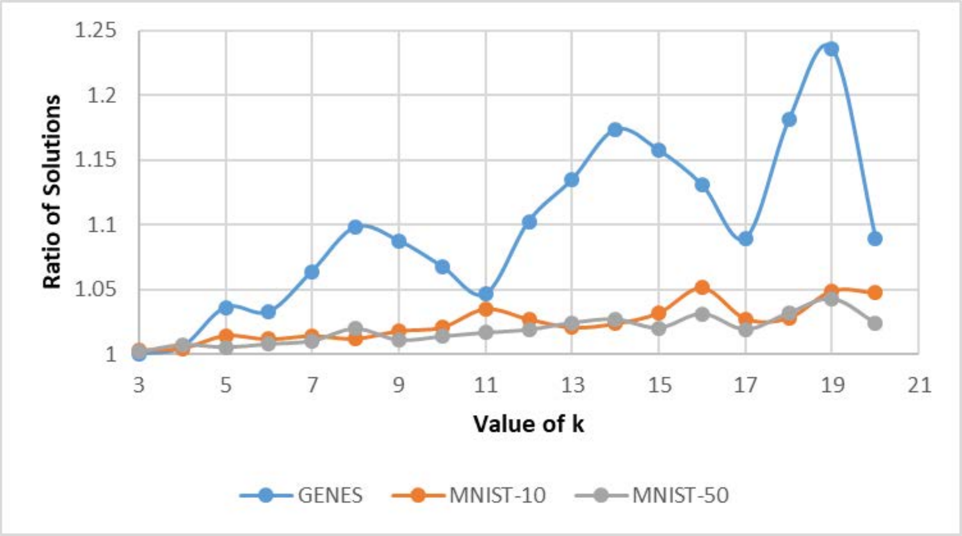
### Data Set

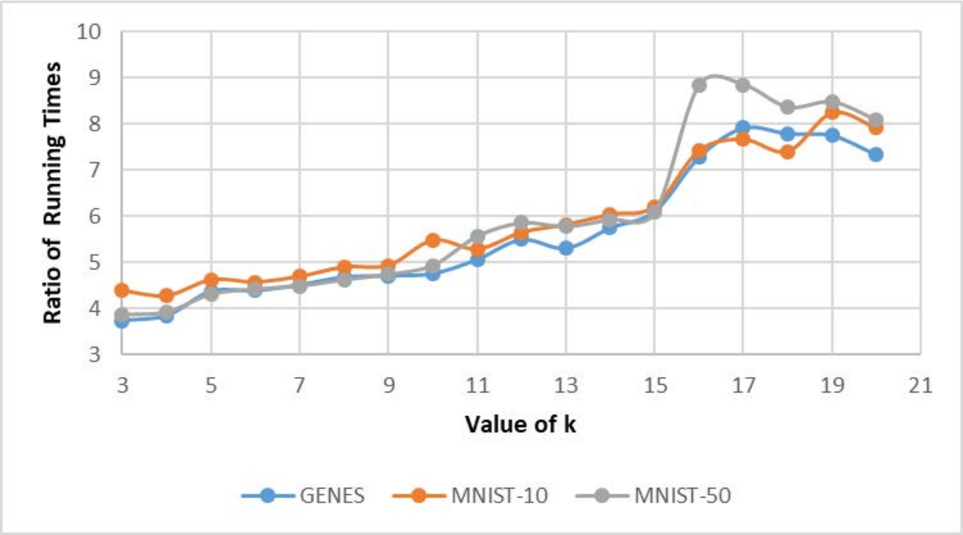
- MNIST, with number of parts = 10
- MNIST, with number of parts = 50
- GENES, with number of parts = 10

### Process

- Partition the data set randomly into parts
- Compute a core-set using one of the algorithms: **Greedy, Local Search, LP-Based algorithm of [IMOR'18]**
- Use greedy on the union of the coresets

# Local Search vs Greedy



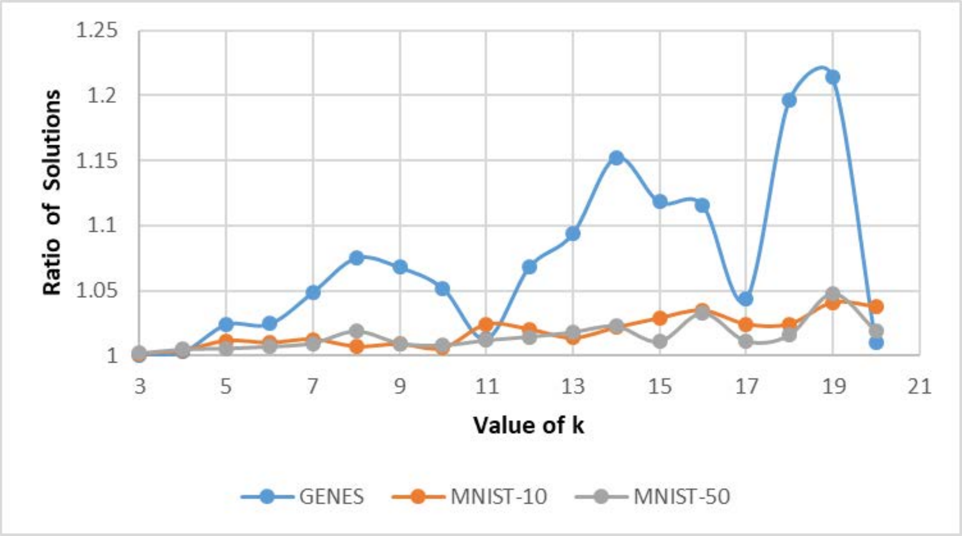

**Improvement of the solution of Local Search over Greedy**

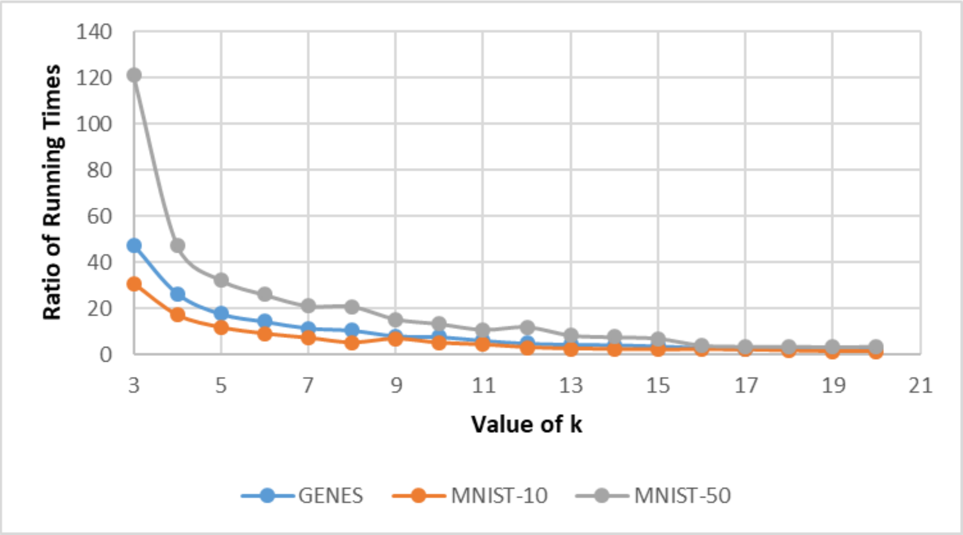
- On average, 1.2%, 2.5%, and 9.6% improvement
- Some cases up to 58% improvement

**Ratio of runtime of Local Search over Greedy**

- On average, 6 times slower

# Local Search vs. LP-based Algorithm of [IMOR'18]




**Improvement of the solution of Local Search over [IMOR'18]**

- On average, 1.4%, 1.8%, and 7.3% improvement
- Some cases up to 63% improvement

**Ratio of runtime of Local Search over [IMOR'18]**

- For lower values of k, Local Search is up to 50 times faster.

## Summary

- Volume/Determinant Maximization Problem
- Notion of composable core-sets
- Algorithms that find composable core-sets for volume/determinant maximization

| | [IMOR'18] | Greedy | Local Search |
|---|---|---|---|
| Approximation | $O(k\log k)^{k/2}$ | $O(C^{k^2})$ | $O(k^k)$ |
| Core-set Size | $O(k\log k)$ | $k$ | $k$ |
| Simple? | × | ✓ | ✓ |
| Empirical Approximation | | | Performs Best |
| Empirical Runtime | Slowest | Fastest | 4 times slower than Greedy. |

## Summary

- Volume/Determinant Maximization Problem
- Notion of composable core-sets
- Algorithms that find composable core-sets for volume/determinant maximization

| | [IMOR'18] | Greedy | Local Search |
|---|---|---|---|
| **Approximation** | $O(k\log k)^{k/2}$ | $O(C^{k^2})$ | $O(k^k)$ |
| **Core-set Size** | $O(k\log k)$ | $k$ | $k$ |
| **Simple?** | × | ✓ | ✓ |
| **Empirical Approximation** | | | **Performs Best** |
| **Empirical Runtime** | **Slowest** | **Fastest** | **4 times slower than Greedy.** |

## Conclusion

- Local Search might be the right algorithm to use in massive data models of computation.

## Summary

- Volume/Determinant Maximization Problem
- Notion of composable core-sets
- Algorithms that find composable core-sets for volume/determinant maximization

| | [IMOR'18] | Greedy | Local Search |
|---|---|---|---|
| **Approximation** | $O(k \log k)^{k/2}$ | $O(C^{k^2})$ | $O(k^k)$ |
| **Core-set Size** | $O(k \log k)$ | $k$ | $k$ |
| **Simple?** | × | ✓ | ✓ |
| **Empirical Approximation** | | | **Performs Best** |
| **Empirical Runtime** | **Slowest** | **Fastest** | **4 times slower than Greedy.** |

## Conclusion

- Local Search might be the right algorithm to use in massive data models of computation.

## Open Problem

- Tight analysis of Greedy: does it also provide approximation $k^{\mathrm{O(k)}}$?

# THANK YOU!

## Summary

- Volume/Determinant Maximization Problem
- Notion of composable core-sets
- Algorithms that find composable core-sets for volume/determinant maximization

| | [IMOR'18] | Greedy | Local Search |
|---|---|---|---|
| **Approximation** | $O(k \log k)^{k/2}$ | $O(C^{k^2})$ | $O(k^k)$ |
| **Core-set Size** | $O(k \log k)$ | $k$ | $k$ |
| **Simple?** | × | ✓ | ✓ |
| **Empirical Approximation** | | | **Performs Best** |
| **Empirical Runtime** | **Slowest** | **Fastest** | **4 times slower than Greedy.** |

## Conclusion

- Local Search might be the right algorithm to use in massive data models of computation.

## Open Problem

- Tight analysis of Greedy: does it also provide approximation $k^{O(k)}$?