

# On Coresets and Shape Fitting in High Dimensions

Sariel Har-Peled

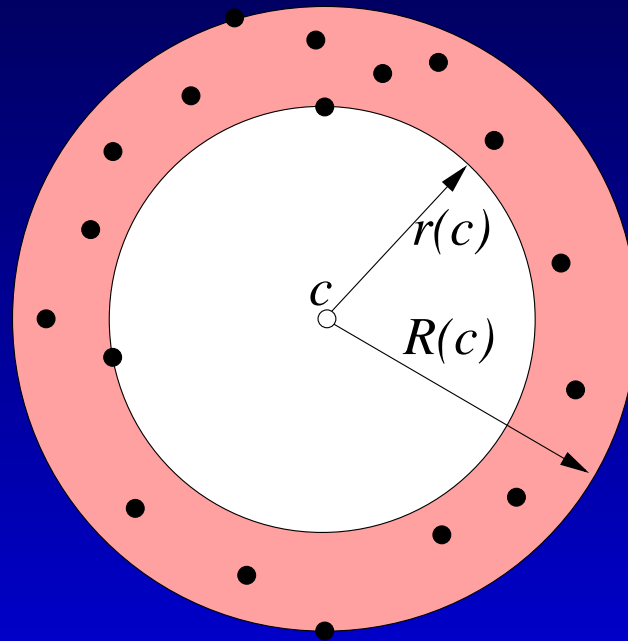
UIUC

# Based on...

1. M. Bădoiu, S. Har-Peled, and P. Indyk.  
Approximate clustering via core-sets. In *Proc. 34th Annu. ACM Sympos. Theory Comput.*, pages 250–257, 2002
2. S. Har-Peled and K. R. Varadarajan.  
Projective clustering in high dimensions using core-sets. In *Proc. 18th Annu. ACM Sympos. Comput. Geom.*, pages 312–318, 2002
3. S. Har-Peled and K.R. Varadarajan.  
High-dimensional shape fitting in linear time. In *Proc. 19th Annu. ACM Sympos. Comput. Geom.*, pages 39–47, 2003

# Problem

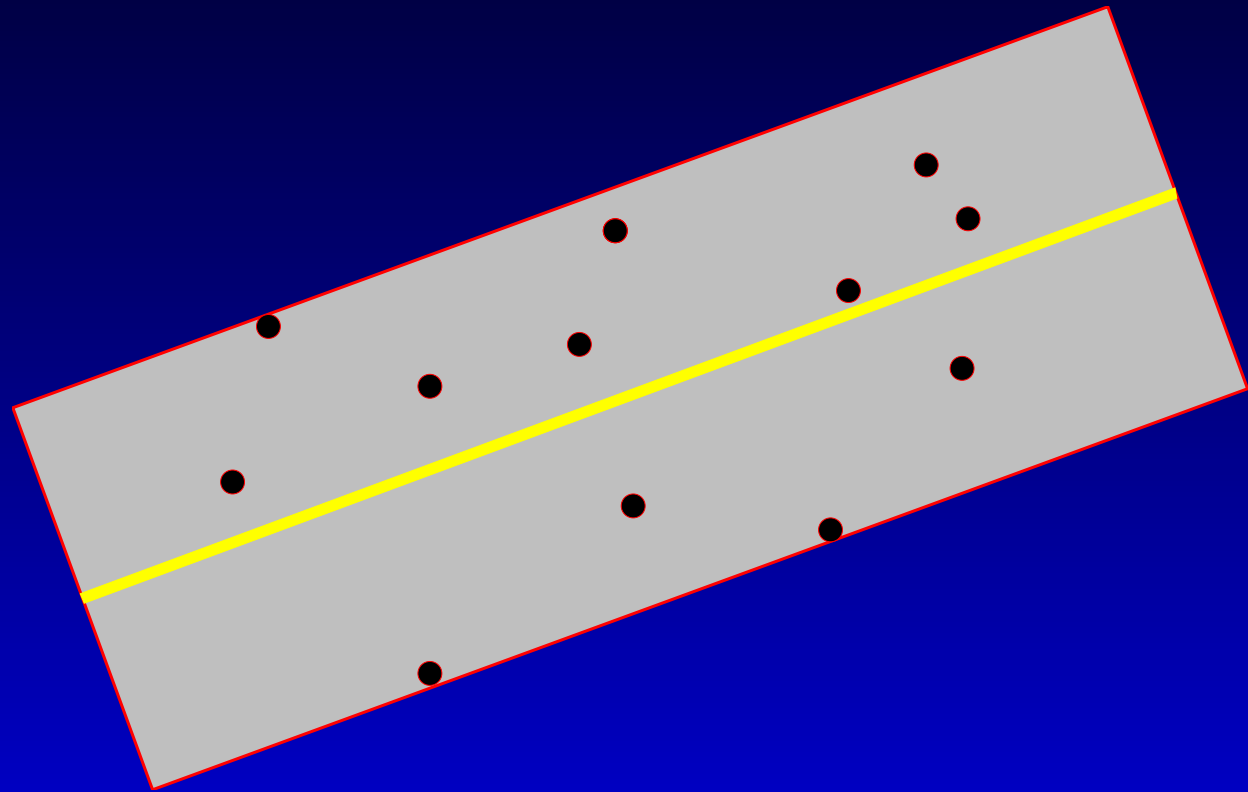
- $P$  - a set of  $n$  points in  $\mathbb{R}^d$
- $\mathcal{F}$  - a family of shapes
- **Target:** Find best shape in  $\mathcal{F}$  matching  $P$ .



- **Motivation:** Clustering, learning, graphics...

# Example: Find best line

- Finding min width strip/cylinder covering  $P$ .



- **Error:** Distance to furthest point.
- Fitting  $\Leftrightarrow$  covering problems.

# Problem: High Dim Shape Fitting

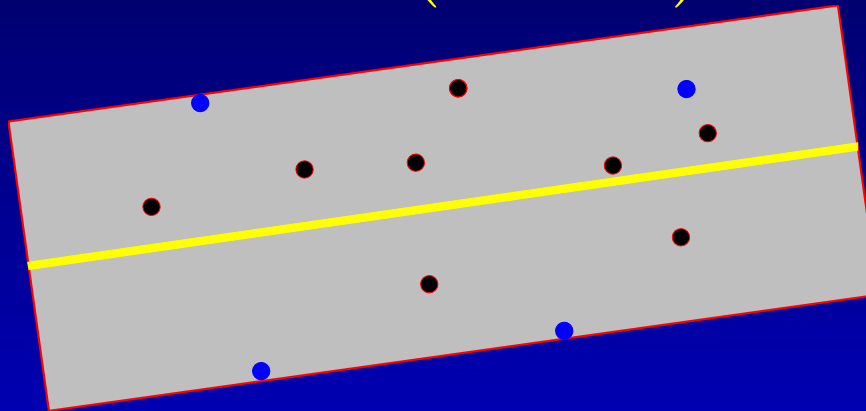
- $P$  - a set of  $n$  points in  $\mathbb{R}^d$
- **Target:** Find best shape matching  $P$ .
- Considerations:
  - Running time with poly dep. on dimension.
  - Can not use dimension reduction.

# Approx in low dim

- [Agarwal, Har-Peled, Varadarajan,01-03]  
Max shape fitting problems
  - $\varepsilon$ -approximation
  - Running time:  $O(n + 1/\varepsilon^{O(1)})$
  - Extracts a coresets of size  $1/\varepsilon^{O(d)}$ .
- **Problems:** Min ball, min cylinder, min vol bounding box, min width spherical shell, etc.

# What is a (low dim) coresheet?

- $S \subseteq P$
- $S$  is a  $\varepsilon$ -coresheet of  $P$ , if for any cover  $C$  of  $S$ :
  - $P \subseteq (1 + \varepsilon)C$
  - $S$  is “small” (i.e.,  $O(1/\varepsilon^{O(d)})$ ).



- Approximation algorithm:
  - Extract coresheet.
  - Find optimal solution on coresheet.

# Low dim coresets

- Can do:

# Low dim coresets

- Can do:
  - Maintain approx under ins/del (polylog update time).

# Low dim coresets

- Can do:
  - Maintain approx under ins/del (polylog update time).
  - Maintain approx for streams (ins only) with small space.

# Low dim coresets

- Can do:
  - Maintain approx under ins/del (polylog update time).
  - Maintain approx for streams (ins only) with small space.
- Coresets exists for shape fitting:

# Low dim coresets

- Can do:
  - Maintain approx under ins/del (polylog update time).
  - Maintain approx for streams (ins only) with small space.
- Coresets exists for shape fitting:
  - # of outliers small (work with Yusu Wang).

# Low dim coresets

- Can do:
  - Maintain approx under ins/del (polylog update time).
  - Maintain approx for streams (ins only) with small space.
- Coresets exists for shape fitting:
  - # of outliers small (work with Yusu Wang).
  - For  $k$ -median and  $k$ -mean (in progress).

# Low dim coresets

- Can do:
  - Maintain approx under ins/del (polylog update time).
  - Maintain approx for streams (ins only) with small space.
- Coresets exists for shape fitting:
  - # of outliers small (work with Yusu Wang).
  - For  $k$ -median and  $k$ -mean (in progress).
- **Problem:** Coreset size exponential in dim.

# Low dim coresets

- Can do:
  - Maintain approx under ins/del (polylog update time).
  - Maintain approx for streams (ins only) with small space.
- Coresets exists for shape fitting:
  - # of outliers small (work with Yusu Wang).
  - For  $k$ -median and  $k$ -mean (in progress).
- **Problem:** Coreset size exponential in dim.
- **Q:** Coreset in high dim?

# High dim coresets

- Exponential lower bound.

# High dim coresets

- Exponential lower bound.
- Weaker coresets:
  - $S \subseteq P$
  - $B$  smallest enclosing ball of  $S$ .
  - $P \subseteq (1 + \varepsilon)B$ .

# High dim coresets

- Exponential lower bound.
- Weaker coresets:
  - $S \subseteq P$
  - $B$  smallest enclosing ball of  $S$ .
  - $P \subseteq (1 + \varepsilon)B$ .
- Q: Exist?

# High dim coresets

- Exponential lower bound.
- Weaker coresets:
  - $S \subseteq P$
  - $B$  smallest enclosing ball of  $S$ .
  - $P \subseteq (1 + \varepsilon)B$ .
- Q: Exist?
- Q: How to compute?

# High dim coresets

- Exponential lower bound.
- Weaker coresets:
  - $S \subseteq P$
  - $B$  smallest enclosing ball of  $S$ .
  - $P \subseteq (1 + \varepsilon)B$ .
- **Q:** Exist?
- **Q:** How to compute?
- **Target:** Size independent of dim.

# Intuition: Why should high dim coresets exist?

- High dim - glut of information
  - JL Lemma - dimension reduction.
- “Not all points are created equal, some are more equal than others”.
- Input dim vs. target dim.
  - Min ball - target dim 0
  - min cylinder - target dim 1.

# Min Ball

- $P$  - a set of points.
- **Problem:** Compute min enclosing ball of  $P$ .
- Algorithm Init:
  - $p \in P, q \leftarrow$  farthest neighbor of  $p$  in  $P$ .
  - $R$  - radius of min enclosing ball of  $P$ .
  - $S_0 \leftarrow \{p, q\}$
  - $B_0 \leftarrow \text{MinBallQPApprox}(S_0)$   
(MinBallQPApprox - using quadratic programming)
  - $r_0 = r(B_0)$ .
- **Claim:**  $R \leq 2r_0$ .

# MinBall Algorithm

- While  $P \not\subseteq (1 + \varepsilon)B_i$  do
  - $q_i \leftarrow \text{Furthest}_P(B_i)$
  - $S_{i+1} \leftarrow S_i \cup \{q_i\}$ .
  - $B_{i+1} \leftarrow \text{MinBallQPApprox}(S_{i+1})$
  - $i \leftarrow i + 1$
- Return  $B_i$ .

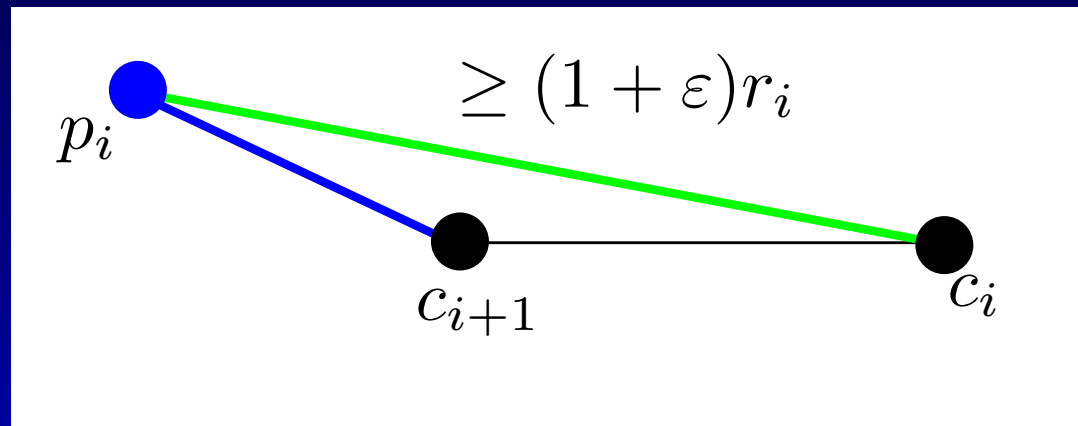
# MinBall # of iterations

- $r_{i+1} \leftarrow r(B_{i+1})$ .
- If  $r_{i+1} \geq (1 + \varepsilon^2/4)r_i$  we made progress.
- $\|c_i c_{i+1}\|$  - distance of centers of  $B_{i+1}$  and  $B_i$ .

# MinBall # of iters - short case

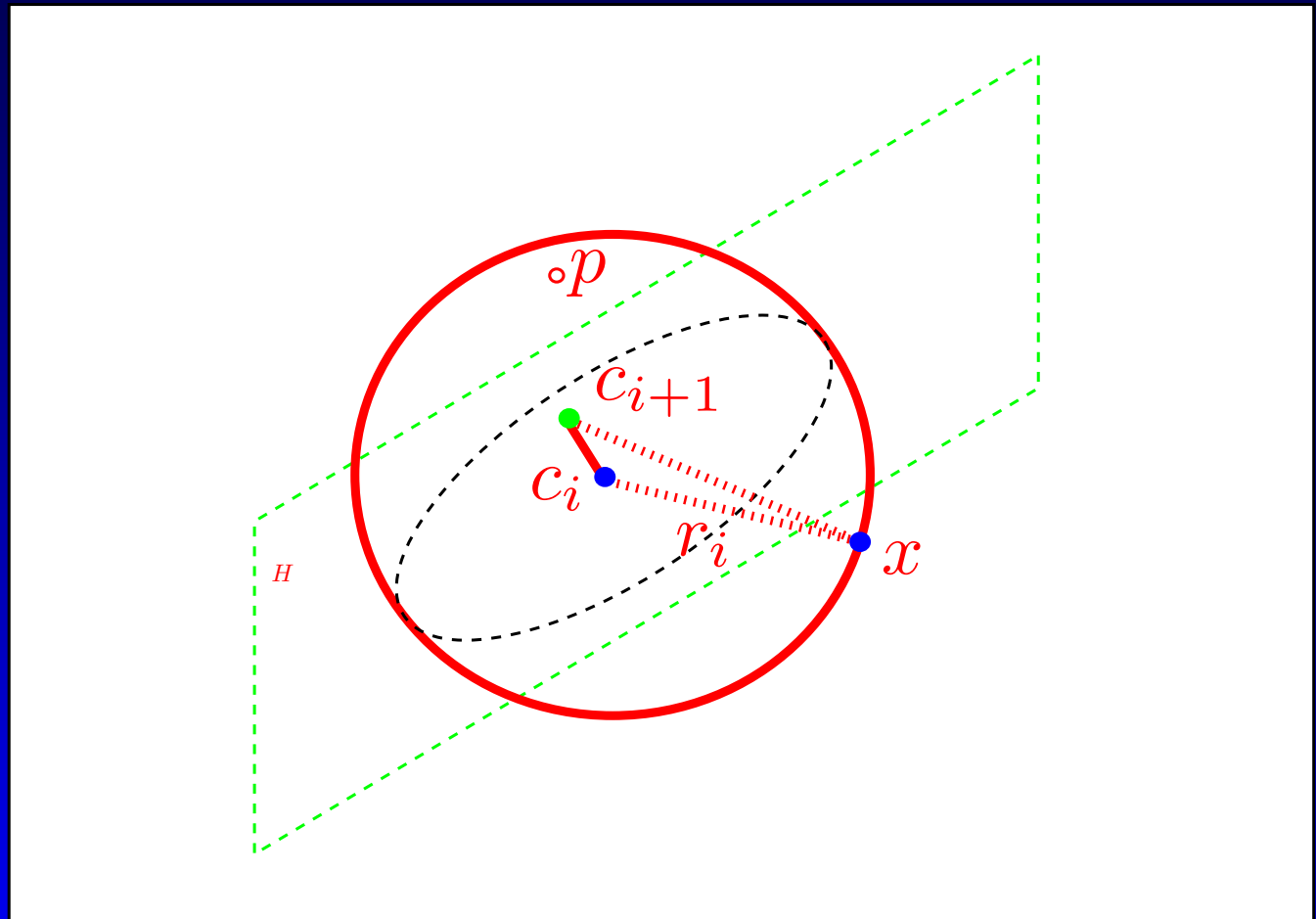
If  $\|c_i c_{i+q}\| < \varepsilon r_i / 2$  then

$$r_{i+1} \geq (1 + \varepsilon)r_i - \|c_i c_{i+1}\| \geq (1 + \varepsilon/2)r_i.$$



# MinBall # of iters - long case

- If  $\|c_i c_{i+1}\| > \varepsilon r_i$  then
$$r_{i+1} \geq \sqrt{r_i^2 + (\varepsilon r_i)^2} \geq (1 + \varepsilon^2/4)r_i.$$



# MinBall Result

## Theorem:

MinBall algorithm terminates after  $O(1/\varepsilon^2)$  iterations.

[Bădoiu, Clarkson, 02]

Number of iterations  $O(1/\varepsilon)$ .

# MinBall Algorithm

- $O(1/\varepsilon)$  iterations. Every iteration:
  - Scan points - linear time.
  - Quadratic programming for MinBallQPApprox:  $O(1/\varepsilon^9)$  time.
- Overall running time  $O(nd/\varepsilon + 1/\varepsilon^{10})$ .
- Big speed gain in avoiding quadratic programming for whole pointset.

# $k$ -center clustering

- **Target:** Cover points with  $k$  balls.
- Want to run  $k$  parallel copies of `MinBall`
- At every iteration:
  - $\mathcal{B}_i$  - set of  $k$  current balls.
  - If  $P \subseteq (1 + \varepsilon)\mathcal{B}_i$  - done.
  - $q_i$  - point furthest away from  $\mathcal{B}_i$ .
- **Q:** To which of  $k$  balls to add  $q_i$ ?  
**A:** Ask an oracle.

# $k$ -center clustering cont'

**Theorem:** Compute  $(1 + \varepsilon)opt$   $k$ -center clustering in  $O(ndk/\varepsilon + k/\varepsilon^{10})$  time. Requires  $O(k/\varepsilon)$  oracle queries.

Removing the oracle:

**Theorem:** Compute  $(1 + \varepsilon)opt$   $k$ -center clustering in  $k^{O(k/\varepsilon)}(nd/\varepsilon + k/\varepsilon^{10})$  time.

# Min Ball with Outliers

- $P$  - set of  $n$  points.
- A large subset of  $P$  are outliers.
- **Q**: Find smallest ball that contains  $m$  points of  $P$ .
- $Q \subseteq P$  - required set.
- $S \subseteq Q$  - coresset of MinBall of  $Q$  of size  $O(1/\epsilon)$ .

# Min Ball with Outliers

- **Algorithm:**
  - Try all subsets  $\subseteq P$  of size  $O(1/\varepsilon)$ :
    - Compute  $B \leftarrow \text{MinBall}(S)$
    - Compute  $\#$  points of  $P$  covered by  $B$
  - Return best ball computed
- Running time  $n^{O(1/\varepsilon)}$ .

# 1-Median

- A random sample span a subspace that contains a good center.  
Yield a fast algorithm for approximate 1-median...
- Can be extended to  $k$ -median. Either:
  - All clusters are “big”.  
All clusters can be sampled using random sampling. And solved on subspace.
  - Sample large cluster, remove it, and recurse on remaining points.

# $k$ -Median - Result

**Theorem:** For any point-set  $P \subset \mathbb{R}^d$ ,  $\varepsilon < 1$ ,  $k$ , a  $(1 + \varepsilon)$ -approximate  $k$ -median for  $P$  in time

$$2^{(k/\varepsilon)^{O(1)}} d^{O(1)} n \log^{O(k)} n$$

Polynomial dependency on dimension.

# Min Radius Cylinder

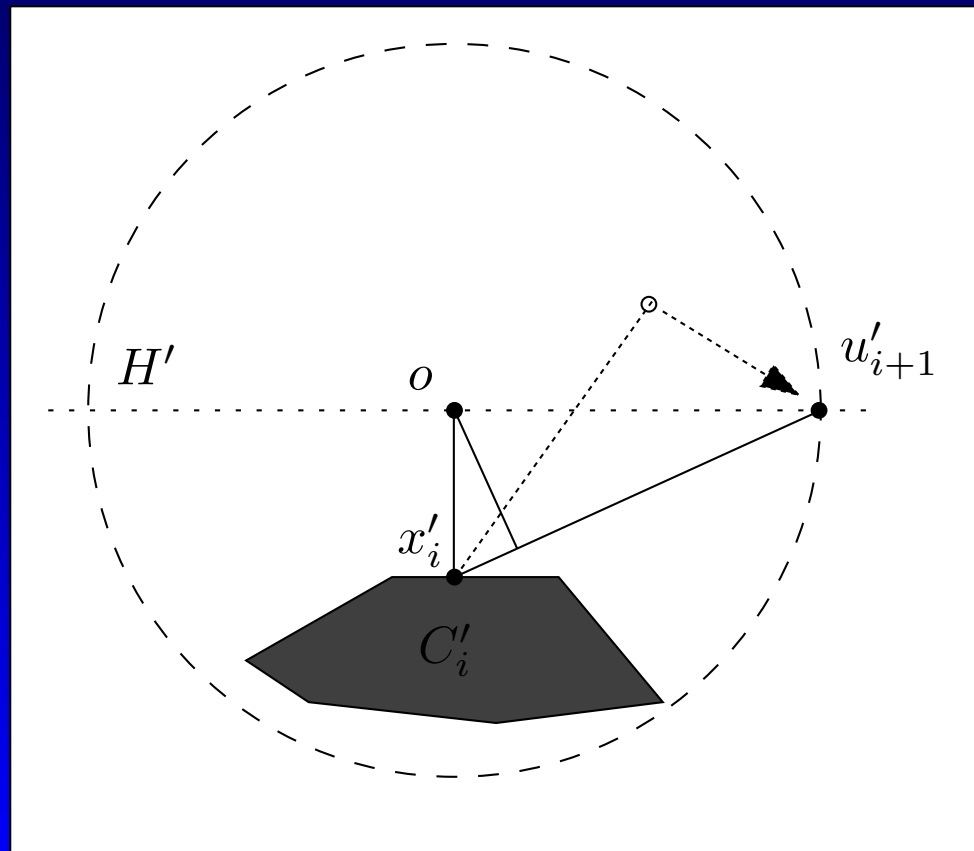
- Find a line close to the optimal center line.
- Idea: Find two points close to center line - use spanned line as center.
- **Q:** How to find points?

# Min Radius Cylinder

Finding one point close to center line

**Observation:**

One point easy by similar argument to MinBall coresets...



# Min Radius Cylinder

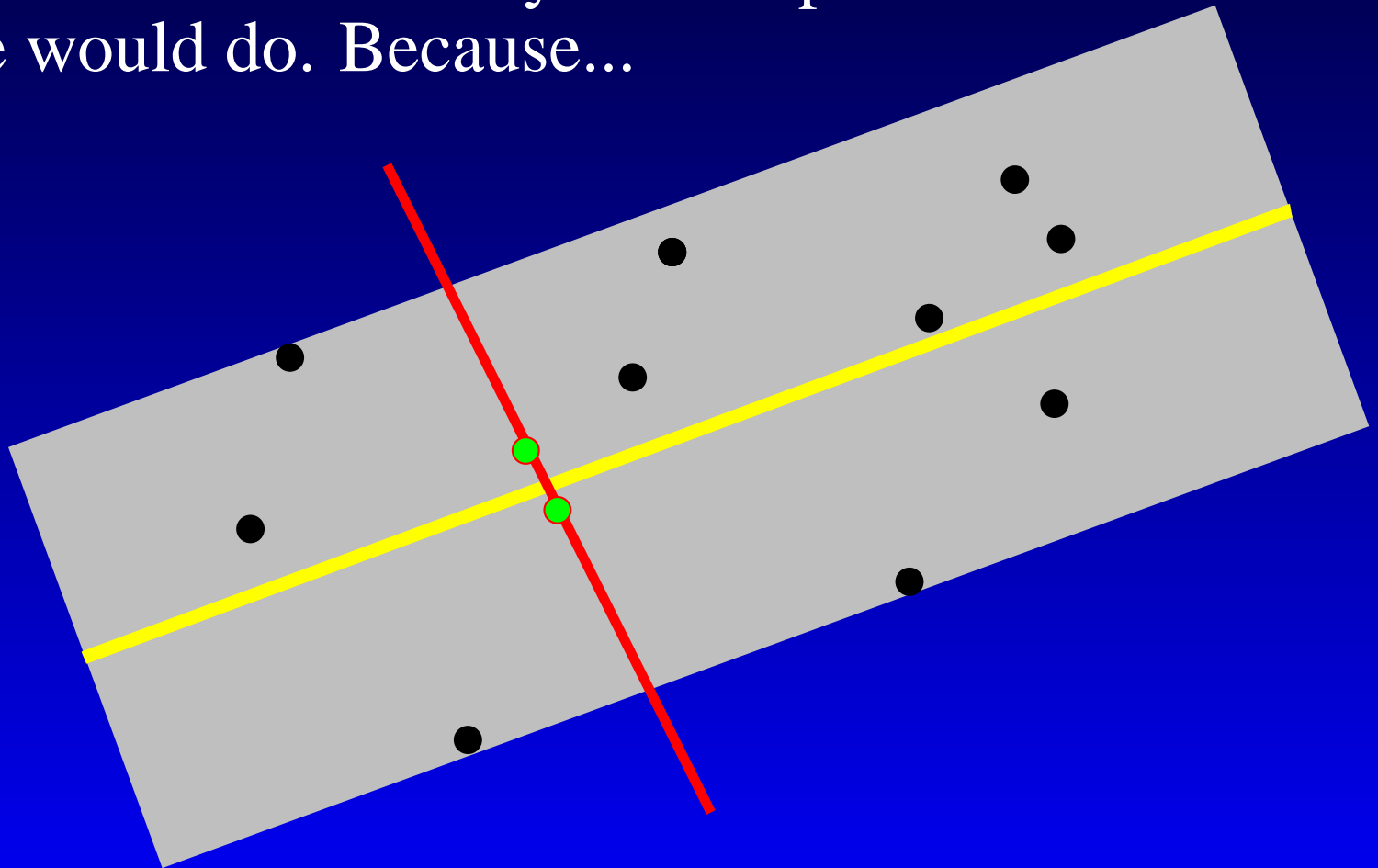
Finding a good second point?

**Observation:** Not any second point close to center line would do. Because...

# Min Radius Cylinder

Finding a good second point?

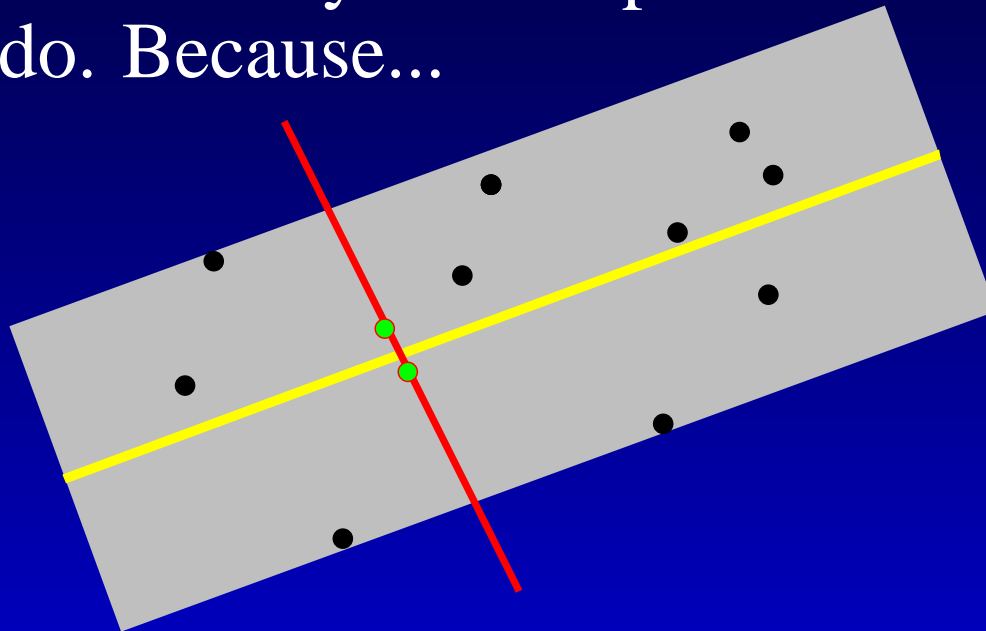
**Observation:** Not any second point close to center line would do. Because...



# Min Radius Cylinder

Finding a good second point?

**Observation:** Not any second point close to center line would do. Because...



**Solution:** Find two faraway points...

# Min Radius Cylinder

Finding a good second point?

**Observation:** Not any second point close to center line would do. Because...

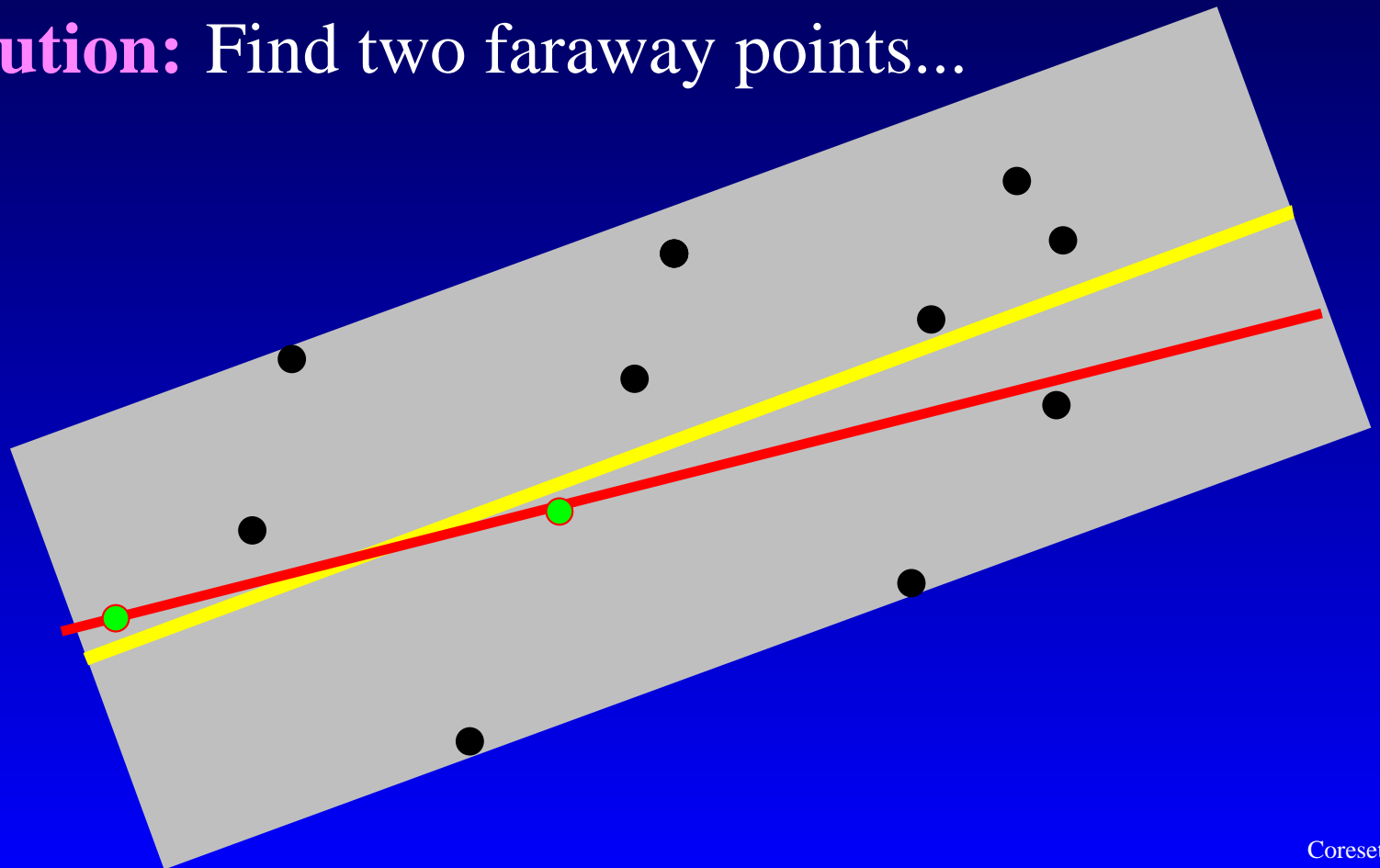
**Solution:** Find two faraway points...

# Min Radius Cylinder

Finding a good second point?

**Observation:** Not any second point close to center line would do. Because...

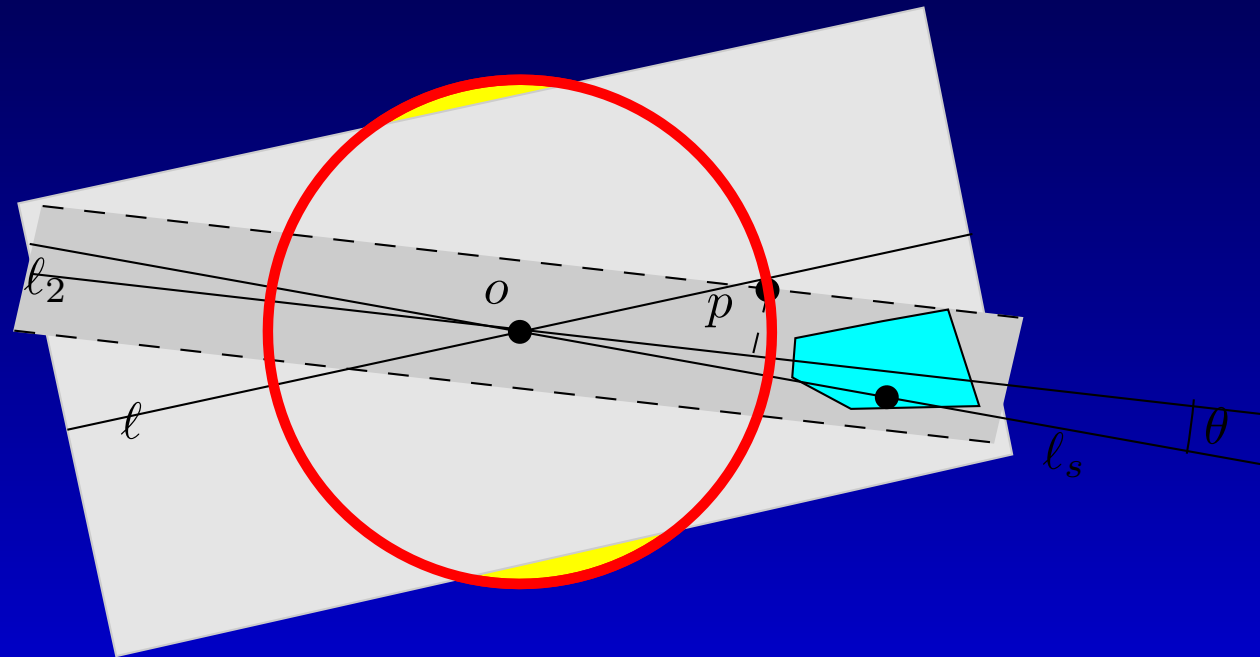
**Solution:** Find two faraway points...



# Min Radius Cylinder - cont'

$o$  - point on center line of cylinder (via coresets)

- Short case:  $\text{diam}(P) \leq r_{\text{opt}}(P)/\varepsilon$ .



# Min Radius Cylinder - cont'

- Long case:  $\text{diam}(P) > r_{\text{opt}}(P)/\varepsilon$

Remove points inside  $\text{ball}(o, \varepsilon \text{diam}(P))$ , and use similar argument to short case.

# Min Radius Cylinder - cont'

**Theorem:** Given  $P \subseteq \mathbb{R}^d$ , one can find a coresets  $S$  of size  $O(1/\varepsilon^5 \log 1/\varepsilon)$ . Such that  $\text{span}(S)$  contains  $(1 + \varepsilon)$ -approximate line center.

- Enumerate all such coresets:  $n^\rho$

$$\rho = O\left(\frac{1}{\varepsilon^5} \log \frac{1}{\varepsilon}\right).$$

- Compute for each such coresets its line.
- Return best line out of all of them.

# Min Radius Cylinder - cont'

1. **Theorem:** Given a pointset  $P$  in  $\mathbb{R}^d$ , one can compute a cylinder covering  $P$  of radius  $\leq (1 + \varepsilon)r_{\text{opt}}(P)$ , in  $dn^{O(1/\varepsilon^5 \log 1/\varepsilon)}$  time.
2. **[Bădoiu, Clarkson, 02]**  
Coreset of cylinder of size  $O(1/\varepsilon^4)$ .
3. Find min-cylinder with outliers:  $n^{O(1/\varepsilon^4)}$ .
4. Find cover by  $j$  min-cylinders:  $n^{O(j/\varepsilon^4)}$ .

# Min Radius $k$ -flat

1.  $v_1, \dots, v_{k-1}, v_k$  vectors spanning optimal  $k$ -flat.
  2. Collapse space by projecting along  $v_1, v_2, \dots, v_{k-1}$ .
  3. Optimal  $k$ -flat - optimal cylinder in collapsed space.
  4. By coresets find approx cylinder line  $\ell$ .
  5. Collapse original space along  $\ell$ .
  6. Repeat argument on collapsed space.
- $\exists$  a coreset of size  $O(k/\varepsilon^4)$  for min radius  $k$ -flat.

# Min Radius $k$ -flat

1. Coreset of min width  $k$ -flat of size  $O(k/\varepsilon^4)$ .  
(dim independent!)
2. Find min radius  $k$ -flat + outliers:  $n^{O(k/\varepsilon^4)}$ .
3. Find cover by  $j$  min radius  $k$ -flats:

$$n^{O(jk/\varepsilon^4)}.$$

Running time not exciting, can we do better?

# Min Radius Cylinder Lower Bound

By extending a lower bound of Megiddo (Meg90):

- Unless  $\mathbb{P} = \text{NP}$ .
- There is no algorithm:
  - Input:  $P$  - set of  $n$  points in  $\mathbb{R}^d$ ,  $0 < \varepsilon \leq 1$ .
  - Output:  $r$ , such that
$$r_{\text{opt}}(P) \leq r \leq (1 + \varepsilon)r_{\text{opt}}(P).$$
  - runs in time polynomial in  $n$ ,  $d$ , and  $1/\varepsilon$ .

# Min Radius Cyl. in Linear Time

- **Idea:** Extend the algorithm for MinBall.
  - Start from two good points in coresets.
  - While cover for coresets not  $(1 + \varepsilon)$ -cover
    - Add furthest pnt from curr line to coresets.
    - Recompute cover of coresets by a cylinder.
- If algorithm do  $O(1/\varepsilon^{O(1)})$  iters, then...
- Running time:  $O(nd/\varepsilon^{O(1)} + \text{noise}(1/\varepsilon))$ .
- **Conjecture:** This algorithm works.

# Min Radius Cylinder

## Quantifying Progress

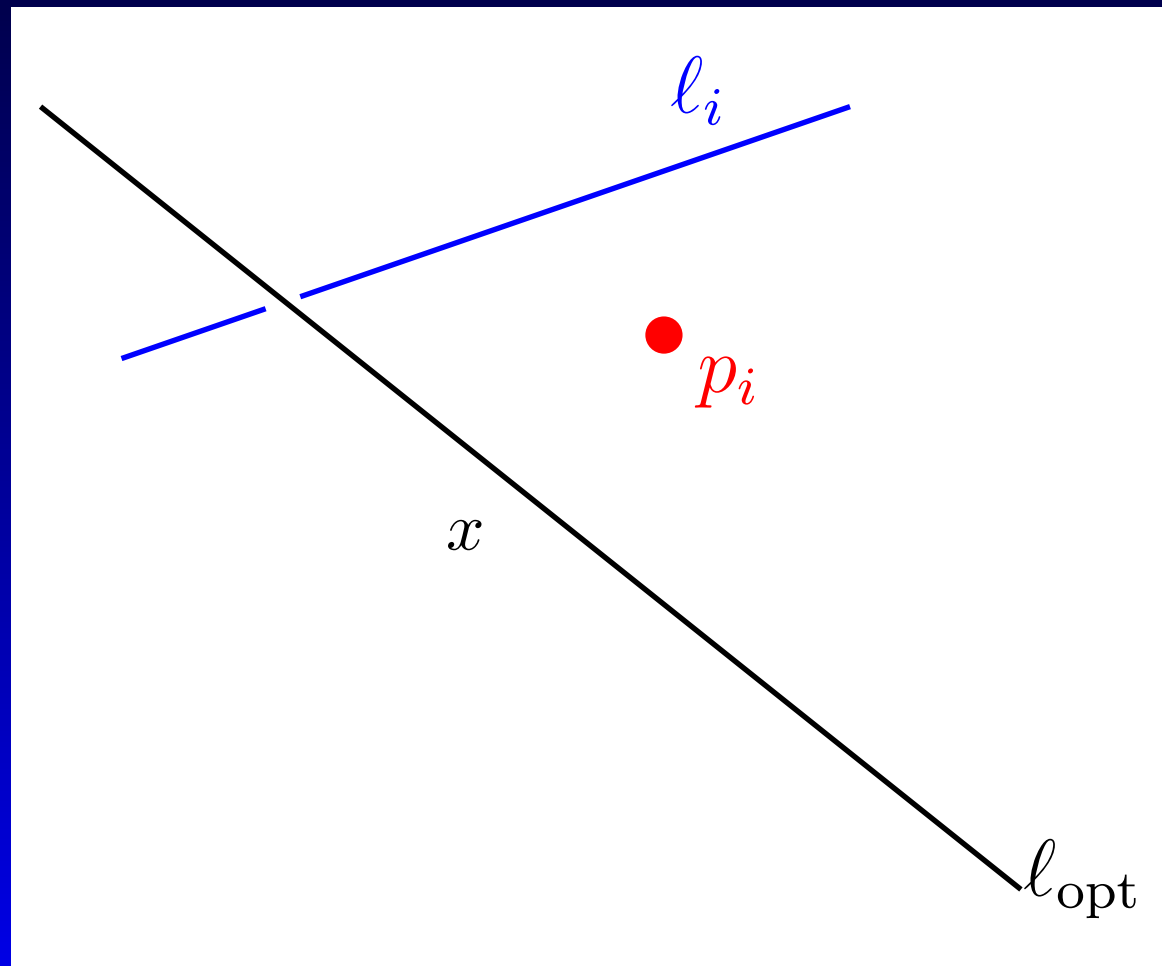
- `MinBall` was analyzable because argument showed progress in every step.
- $\ell_{\text{opt}}$  optimal line center.
- Observation: If a line  $\ell$  is close to  $\ell_{\text{opt}}$  in “interesting region”, then  $\ell$  good approximation.
- $I_{\text{opt}}$  - interval of  $\ell_{\text{opt}}$  containing projection of  $P$ .
- **Lemma:** If  $\forall x \in I_{\text{opt}}$  we have:  
$$\text{dist}(x, \ell) \leq \varepsilon r_{\text{opt}}$$
then  $\ell$  required approximation.

# Min Radius Cylinder Algorithm

- $\ell_0$  - line passing through a point  $x \in P$ , and its furthest neighbor in  $P$ .
- $R \leftarrow$  Guess optimal radius.
- While  $\text{dist}(\ell_i, P) \geq (1 + \varepsilon)R$  do
  - $p_i \leftarrow$  furthest point in  $P$  from  $\ell_i$ .
  - $H_i \leftarrow \text{span}(\ell_i, p_i)$ .
  - $\ell_{i+1} \leftarrow$  Projection of  $\ell_{\text{opt}}$  to  $H_i$ .
  - $i \leftarrow i + 1$
- Return  $\ell_i$ .

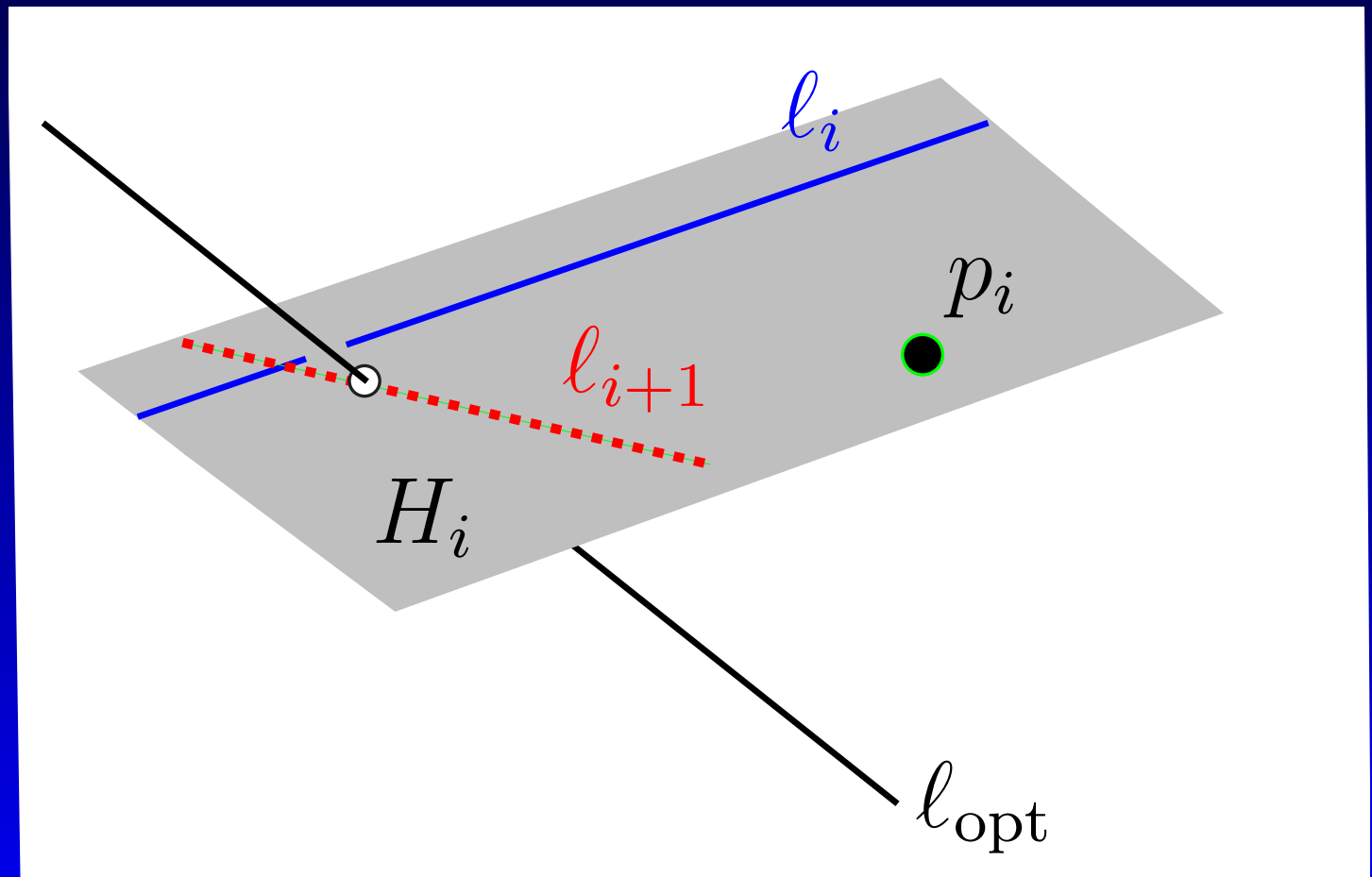
# Min Radius Cylinder - Iteration

Pick Furthest point



# Min Radius Cylinder - Iteration

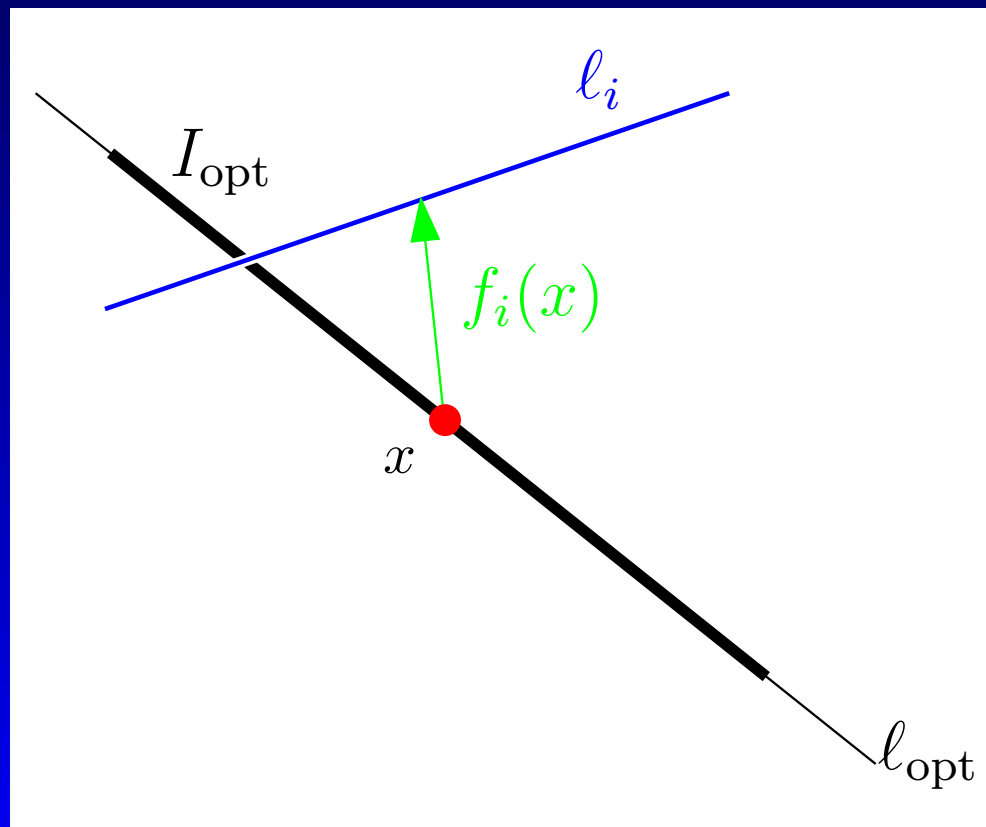
Generate Next Line



# Min Radius Cyl.

## Distance Function

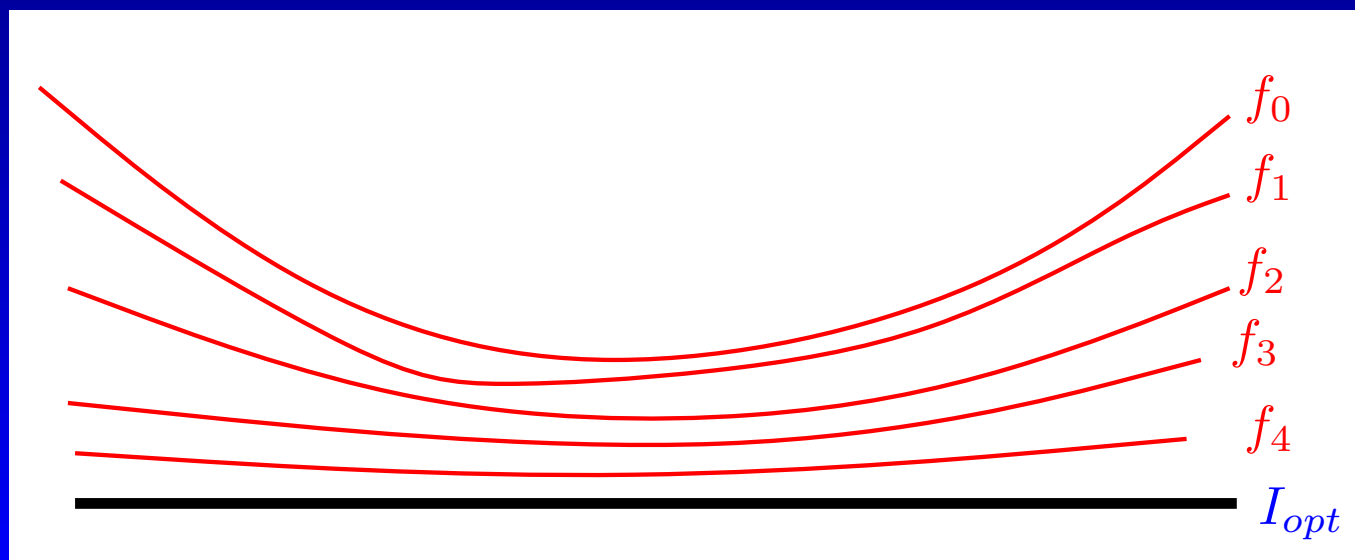
- For a line  $\ell_i$ , define distance function:
  - For  $x \in I_{\text{opt}}$ ,  $f_i(x) = \text{dist}(x, \ell_i)$ .



# Min Radius Cyl.

## Convergence of Dist. Funcs

- For  $x \in I_{\text{opt}}$ :
  - $f_i(x) \leq f_{i-1}(x)$ .
  - $f_0(x) \leq 5r_{\text{opt}}$ .
- If  $f_i(x) \leq (1 - \varepsilon)f_{i-1}(x)$  then bound # iters by  $1/\varepsilon^{O(1)}$ .



# Min Radius Cyl.

## Convergence via a witness set

- At every iteration, a portion of  $I_{\text{opt}}$  is “hit”.
- $T \subseteq I_{\text{opt}}$ ,  $|T| = 1/\varepsilon^{O(1)}$ .
- There exists a point  $t_i \in T$  s.t.
  - $f_{i-1}(t_i) \leq (1 - \varepsilon)f_i(t_i)$ .
- After  $m = 1/\varepsilon^{O(1)}$  iterations

$$\forall t \in T \quad f_m(t) \leq \varepsilon r_{\text{opt}}.$$

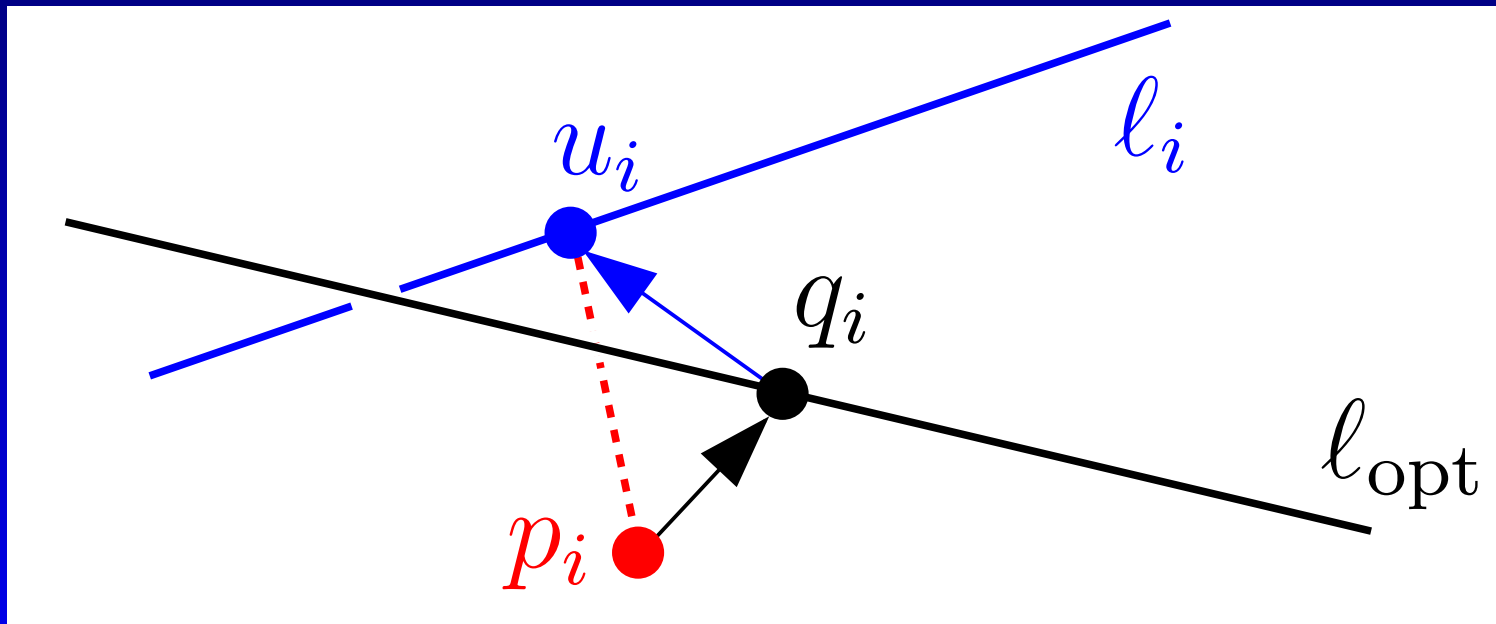
By convexity  $\forall x \in I_{\text{opt}} \quad f_m(x) \leq \varepsilon r_{\text{opt}}.$

- Found required approximation.

# Why is a point being hit?

- $p_i$  - furthest point in  $P$  from  $l_i$ .
- $q_i$  - projection of  $p_i$  on optimal line  $l_{\text{opt}}$ .
- $u_i$  - projection of  $q_i$  to  $l_i$ .

Then  $f_{i+1}(q_i) \leq (1 - \varepsilon/2)f_i(q_i)$ .



# Why a point is being hit? Cont'

We know that:

- $\|p_i q_i\| \leq r_{\text{opt}}$ .
- $\|p_i u_i\| \geq (1 + \varepsilon)r_{\text{opt}}$ .
- $p_i u_i \subseteq H_i$ .

one can show

$$\begin{aligned} f_{i+1}(q_i) &= \text{dist}(q_i, H_i) \\ &\leq \text{dist}(q_i, p_i u_i) \\ &\leq (1 - \varepsilon/2) \text{dist}(q_i, u_i) \\ &= (1 - \varepsilon/2) f_i(q_i). \end{aligned}$$

# Removing the oracle

Without a little help from my friend

**An anti-beetle slide**

- Oracle project opt line to curr plane.
- Instead use oracle that project line approx.
- small  $\#$  of candidate lines.
- Generate all possible oracle guesses.

# Linear Time Min Cylinder

- Running time of algorithm with oracle:  
 $O(nd/\varepsilon^3 \log(1/\varepsilon) + noise)$ .
- Oracle provides  $O(1/\varepsilon^3 \log^2(1/\varepsilon))$  random bits.
- Brute force enumeration, gives an algorithm with

$$nd \cdot \exp\left(O\left(\frac{1}{\varepsilon^3} \log^2 \frac{1}{\varepsilon}\right)\right)$$

running time.

- Outputs a cylinder of radius  $\leq (1 + \varepsilon)r_{\text{opt}}$ .

# Linear Time Min Radius $k$ -Flat

- A tedious extension of Cylinder alg.
- Algorithm with

$$n \cdot d \cdot \exp\left(e^{O(k^2)} / \varepsilon^{2k+3}\right)$$

running time.

- output a  $k$  flat of radius  $\leq (1 + \varepsilon)^k r_{\text{opt}}$ .

# Open problems

- Big discrepancy between algorithm with outliers and without outliers.
- A near linear time algorithm for the case of  $k$ -cylinders.
- An efficient approximate to the  $k$ -slab problem in 3d.
- Other distance functions: sum, sum sq.
- Other shapes: ring.
- Direct coresets for  $k$ -median?
- Practical algorithms?