Approximate Nearest Neighbor Problem: Improving Query Time

CS468, 10/9/2006

Outline

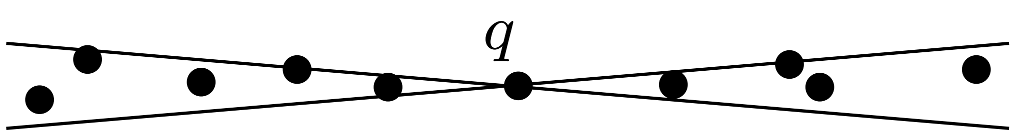
- Reducing the "constant" from $O\left(\epsilon^{-d}\right)$ to $O\left(\epsilon^{-(d-1)/2}\right)$ in query time
- Need to know ϵ ahead of time
  - Preprocessing time and storage feature $O(\epsilon^{-d})$, $O(\epsilon^{-(d-1)/2})$ etc.

Outline

- Reducing the "constant" from $O\left(\epsilon^{-d}\right)$ to $O\left(\epsilon^{-(d-1)/2}\right)$ in query time
- Need to know ϵ ahead of time
  - Preprocessing time and storage feature $O(\epsilon^{-d})$, $O(\epsilon^{-(d-1)/2})$ etc.
- Timothy M. Chan. *Approximate Nearest Neighbor Queries Revisited*. Discrete and Computational Geometry 1998.
  - Decomposition of space into cones
  - BBD-tree for range searching in $\mathbb{R}^{d-k}$ + point location in $\mathbb{R}^k$
- Kenneth Clarkson. *An Algorithm for Approximate Closest-point Queries*. SoCG 1994.
  - Additional $\log(\rho/\epsilon)$ in space complexity
  - Polytope approximation in $\mathbb{R}^{d+1}$

Chen's Algorithm: Motivation

$(1+\epsilon)$-ANN among (sorted) points in a narrow cone

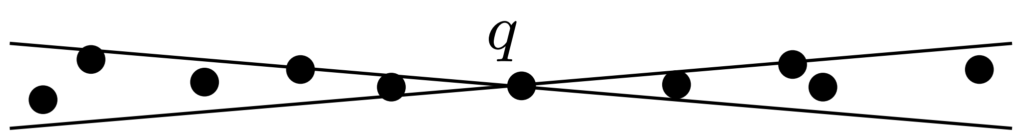


$O(\log n)$ by binary search

Need a data structure that returns a sorted points given q and a cone direction

Chen's Algorithm: Motivation

$(1+\epsilon)$-ANN among (sorted) points in a narrow cone

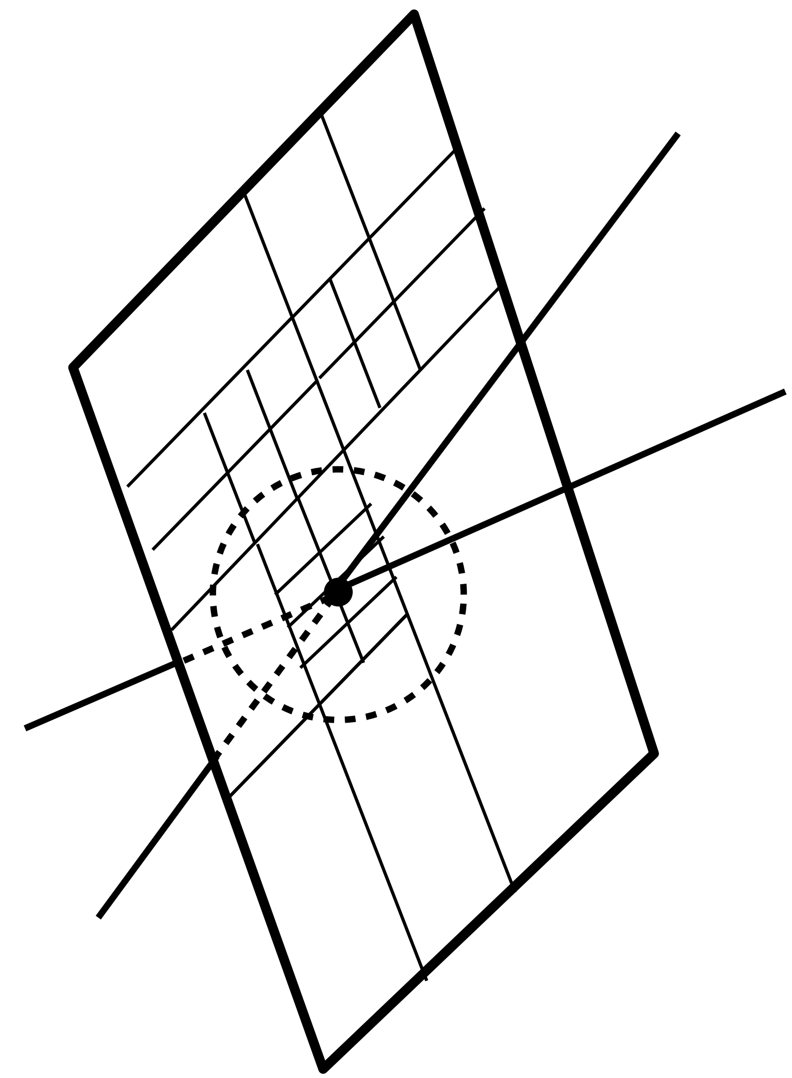


$O(\log n)$ by binary search

Need a data structure that returns a sorted points given q and a cone direction

Uses the BBD-tree data structure

Given a query point $q \in \mathbb{R}^d$ and a radius r
one can find $O(\log n)$ cells of the BBD-tree
which contain $B(q, r)$
and are contained in $B(q, 2r)$.
This takes $O(\log n)$ time



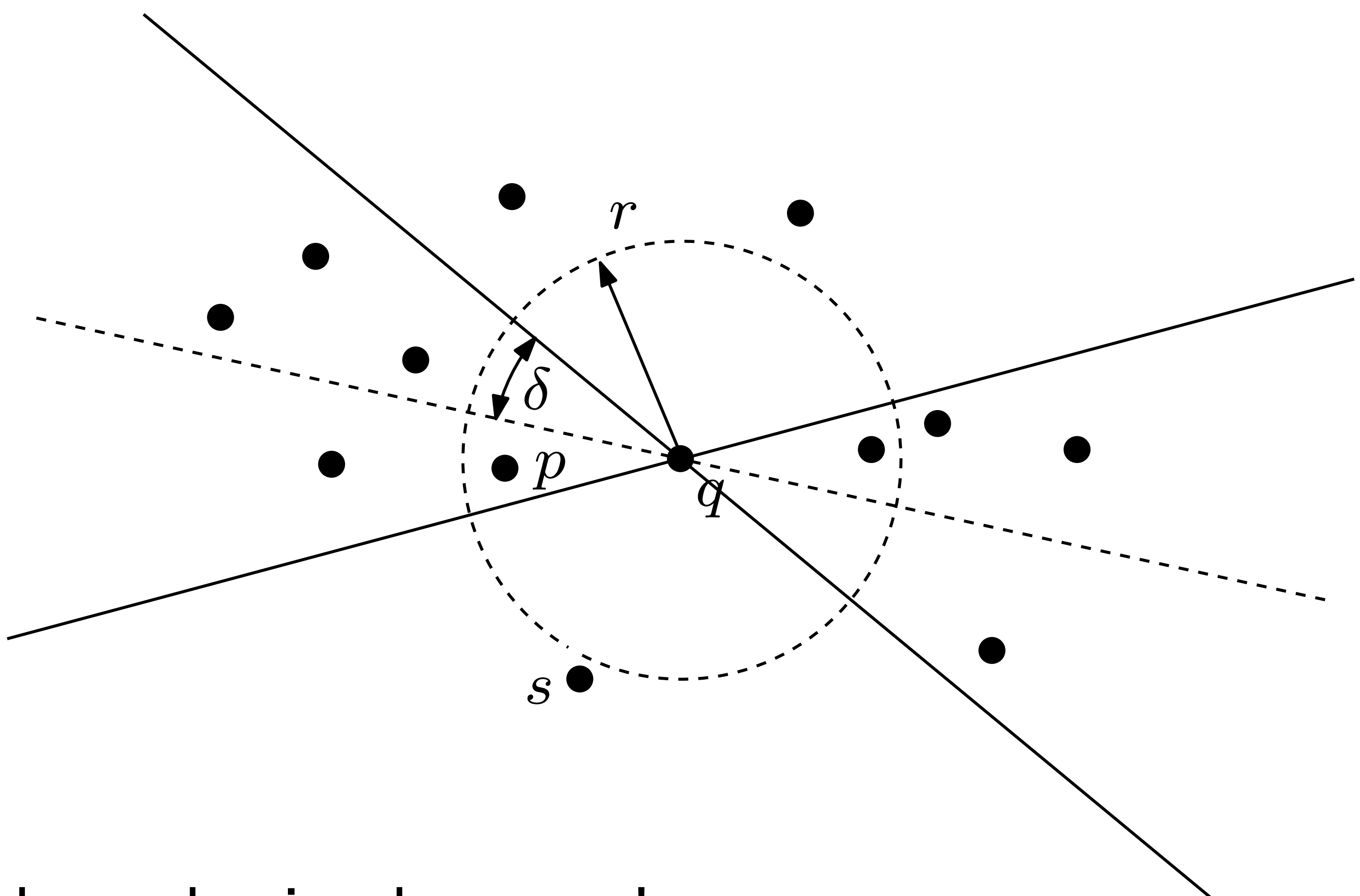
Use for approximate range searching in $\mathbb{R}^{d-1}$

Conic ANN (with a Hint)

Input: Query point q and a 2-approximation r to the NN distance
Output: A points s such that

$$||q - s|| \leq (1 + \epsilon)||q - p||$$

where p is the NN inside a cone with apex q and angle $\delta = \sqrt{\epsilon/16}$



Note: s need not be in the cone!

Note: The cone is fixed (not a part of input, mod. translation to q)

Main $(1+\epsilon)$-ANN Algorithm

Uses the "conic-ANN with a hint" as a subrotine
Query (given only q)

- Obtain r by [Arya and Mount 1998]
- Get one point per data structure, return the one closest to q

Main $(1+\epsilon)$-ANN Algorithm

Uses the "conic-ANN with a hint" as a subrotine

Query (given only q)

- Obtain r by [Arya and Mount 1998]
- Get one point per data structure, return the one closest to q

Preprocessing

- "Tile" $\mathbb{R}^d$ with $O(\epsilon^{-(d-1)/2})$ cones of angle $\delta = \Theta(\sqrt{\epsilon})$ ← "floating"
- Build a "conic-ANN" data structure for each cone

Main $(1+\epsilon)$-ANN Algorithm

Uses the "conic-ANN with a hint" as a subrotine

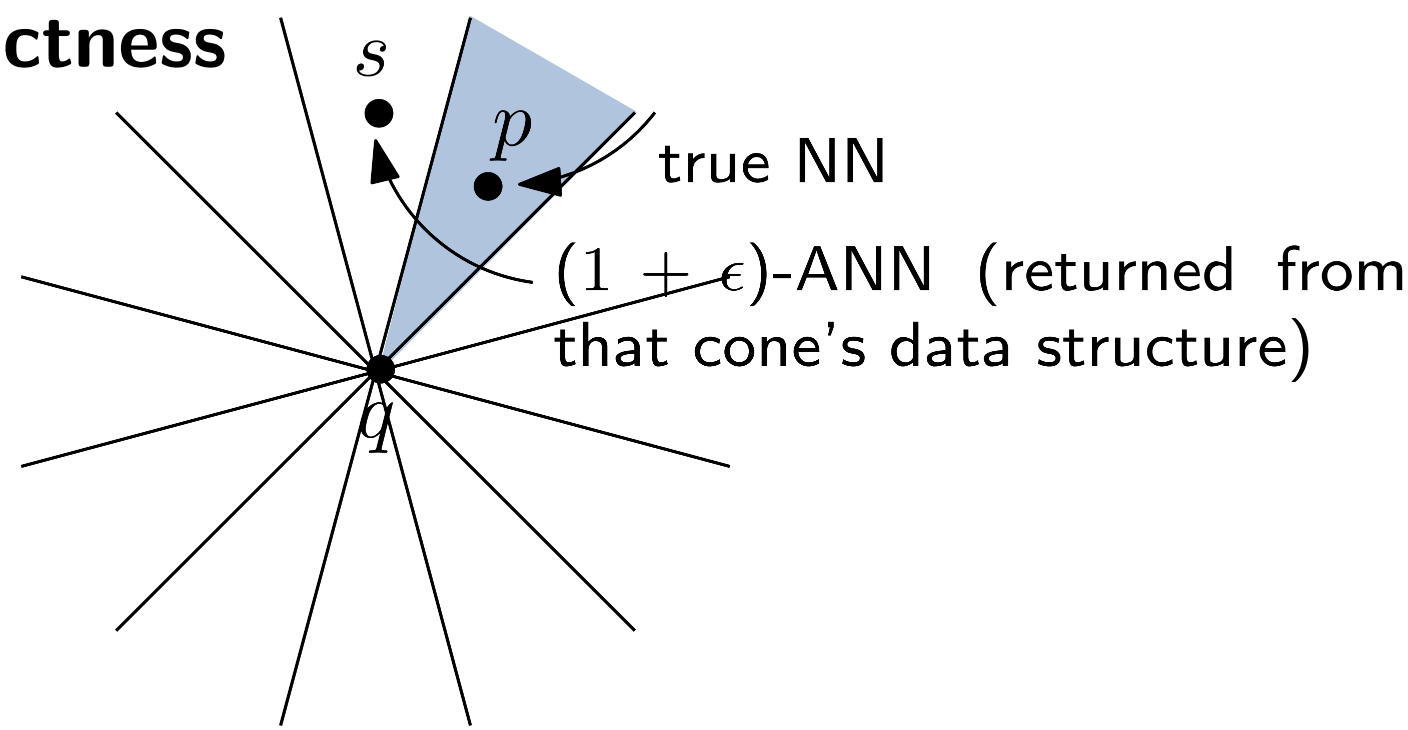
Query (given only q)

- Obtain r by [Arya and Mount 1998]
- Get one point per data structure, return the one closest to q

Preprocessing

- "Tile" $\mathbb{R}^d$ with $O(\epsilon^{-(d-1)/2})$ cones of angle $\delta = \Theta(\sqrt{\epsilon})$ ("floating")
- Build a "conic-ANN" data structure for each cone

Correctness



Main $(1+\epsilon)$-ANN Algorithm

Uses the "conic-ANN with a hint" as a subrotine

Query (given only q)

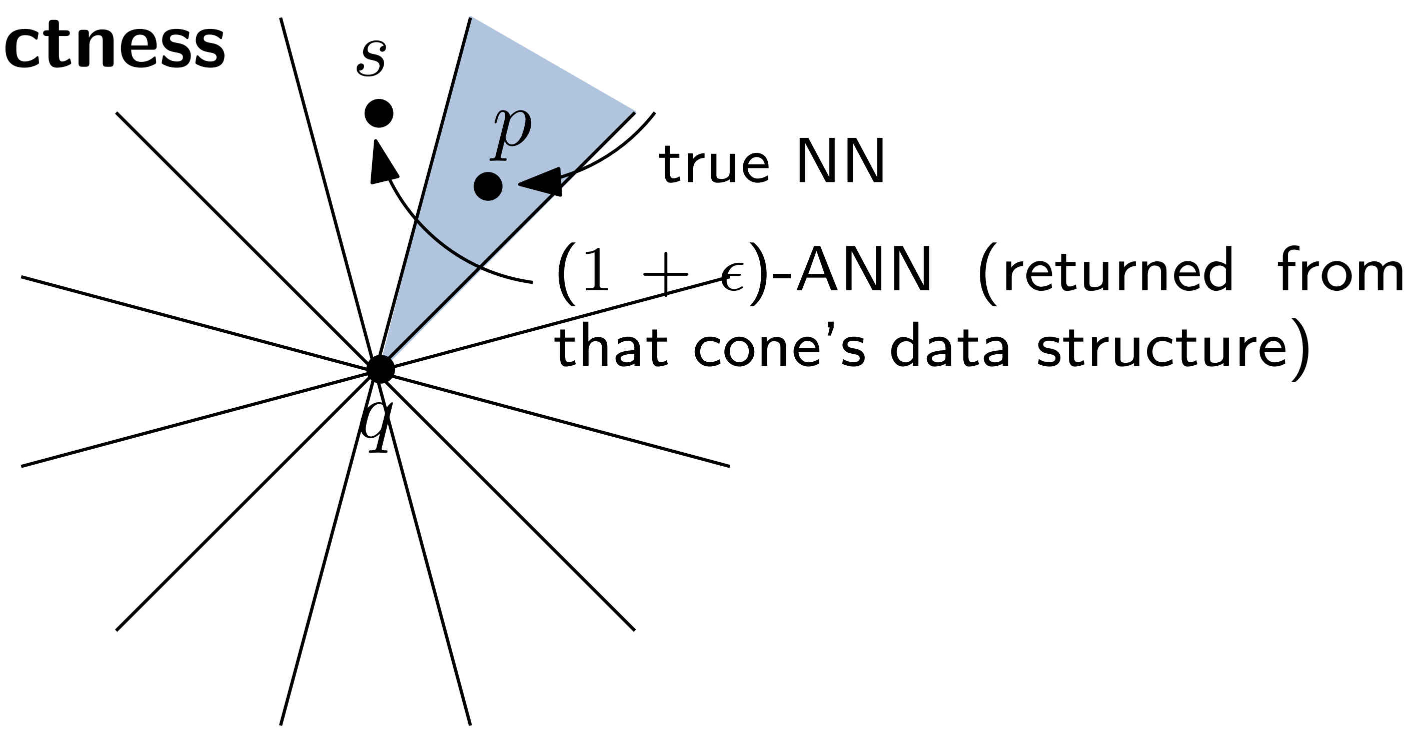
- Obtain r by [Arya and Mount 1998]
- Get one point per data structure, return the one closest to q

Preprocessing

"floating"

- "Tile" $\mathbb{R}^d$ with $O(\epsilon^{-(d-1)/2})$ cones of angle $\delta = \Theta(\sqrt{\epsilon})$
- Build a "conic-ANN" data structure for each cone

Correctness

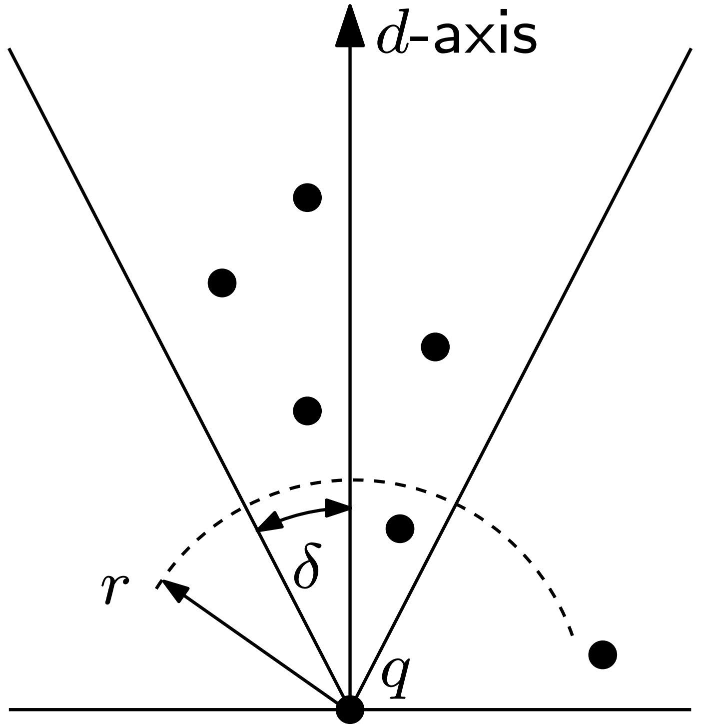


Query time

$O(\underbrace{\epsilon^{-(d-1)/2}}_{\text{[\# of cones]}} \underbrace{\log n}_{\text{[conic query]}})$

Conic-ANN Data Structure

For preprocessing given only direction of the cone (wlog: d-axis) and angle δ



Conic-ANN Data Structure

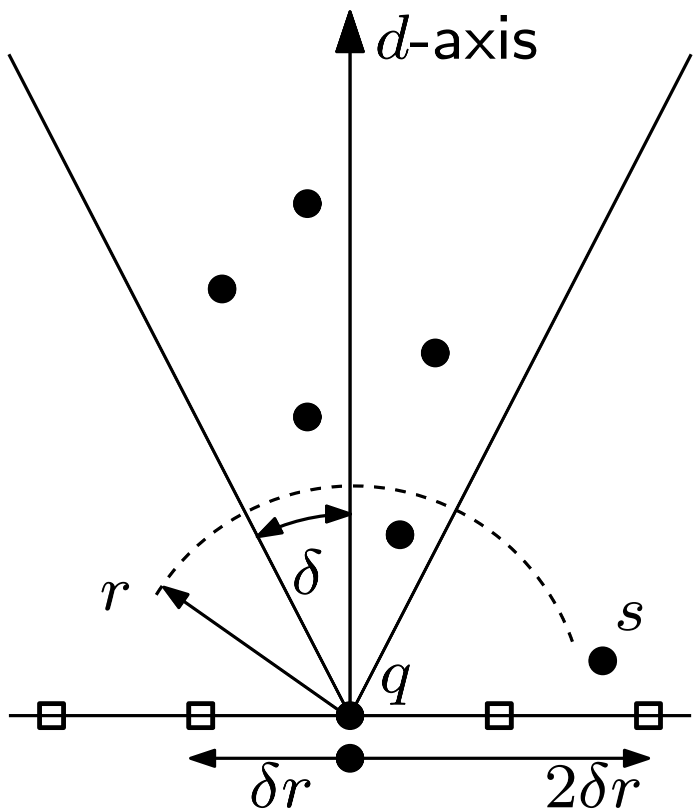
For preprocessing given only direction of the cone (wlog: d-axis) and angle δ

Query Algorithm (given q and r)

Approximate range query on the set of projections
$\{p' = [p_1\ p_2\ \cdots p_{d-1}]^T,\ p \in P\}$ with $B(q, \delta r)$

- returns $O(\log n)$ BBD-nodes (cells) in $O(\log n)$ time

$O(\log n)$ binary searches
Return the point s such that $|s_d - q_d|$ is min



Conic-ANN Data Structure

For preprocessing given only direction of the cone (wlog: d-axis) and angle δ

Query Algorithm (given q and r)

Approximate range query on the set of projections
$\{p' = [p_1\ p_2\ \cdots p_{d-1}]^T,\ p \in P\}$ with $B(q, \delta r)$

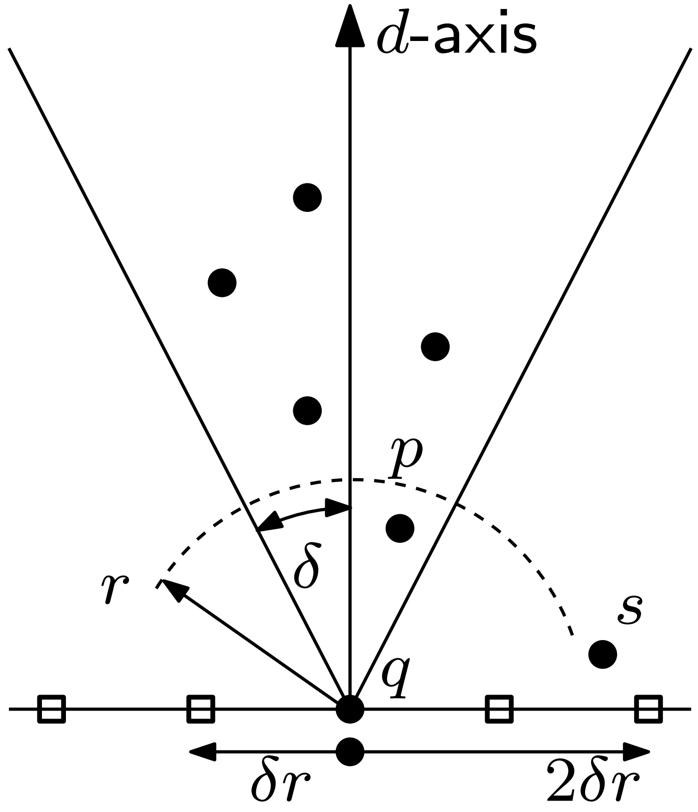
- returns $O(\log n)$ BBD-nodes (cells) in $O(\log n)$ time

$O(\log n)$ binary searches
Return the point s such that $|s_d - q_d|$ is min

Correctness (proof for $||q - s|| \leq (1 + \epsilon)||q - p||$)

$|s_d - q_d| \leq |p_d - q_d| \leq ||p - q||$
$|s' - q'| \leq 2\delta r \leq 4\delta||p - q||$

$||s - q|| \leq \sqrt{1 + 16\delta^2}||p - q|| = (1 + \epsilon)||p - q||$



Conic-ANN Data Structure

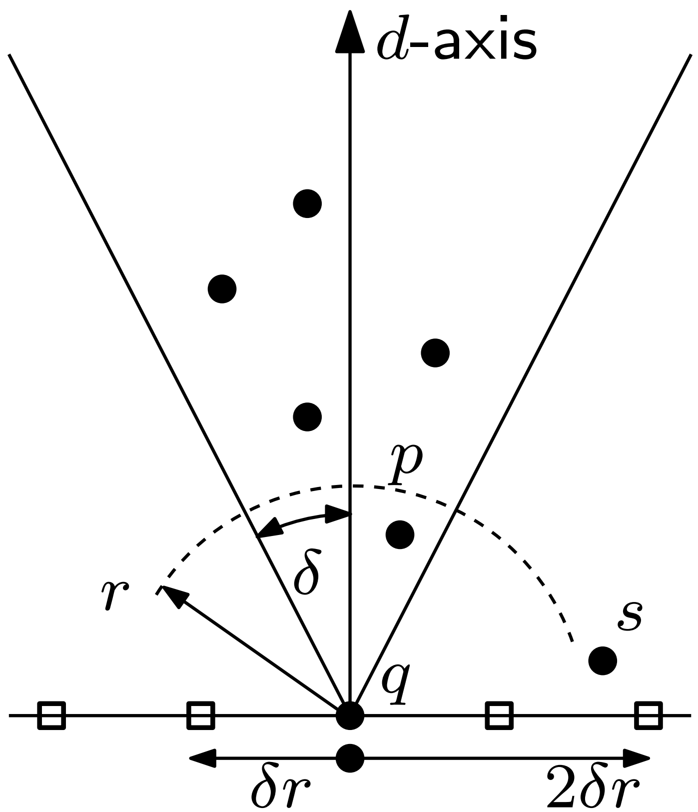
For preprocessing given only direction of the cone (wlog: d-axis) and angle δ

Query Algorithm (given q and r)

Approximate range query on the set of projections
$\{p' = [p_1 \ p_2 \ \cdots p_{d-1}]^T, \ p \in P\}$ with $B(q, \delta r)$

- returns $O(\log n)$ BBD-nodes (cells) in $O(\log n)$ time

$O(\log n)$ binary searches
Return the point s such that $|s_d - q_d|$ is min



Correctness (proof for $||q - s|| \leq (1 + \epsilon)||q - p||$)

$|s_d - q_d| \leq |p_d - q_d| \leq ||p - q||$
$|s' - q'| \leq 2\delta r \leq 4\delta||p - q||$

$||s - q|| \leq \sqrt{1 + 16\delta^2}||p - q|| = (1 + \epsilon)||p - q||$

Data structure
BBD-tree on the projection set
For every tree node v the associated list of points is sorted in the d coordinate

Conic-ANN Analysis

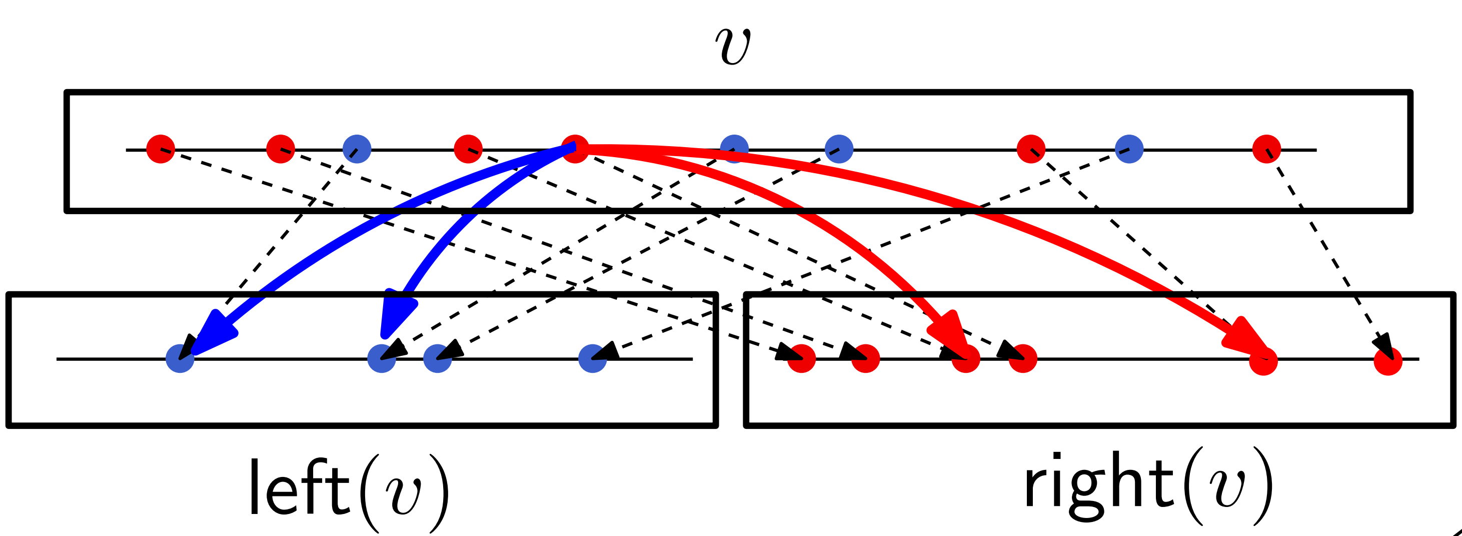
Construction (preprocessing)

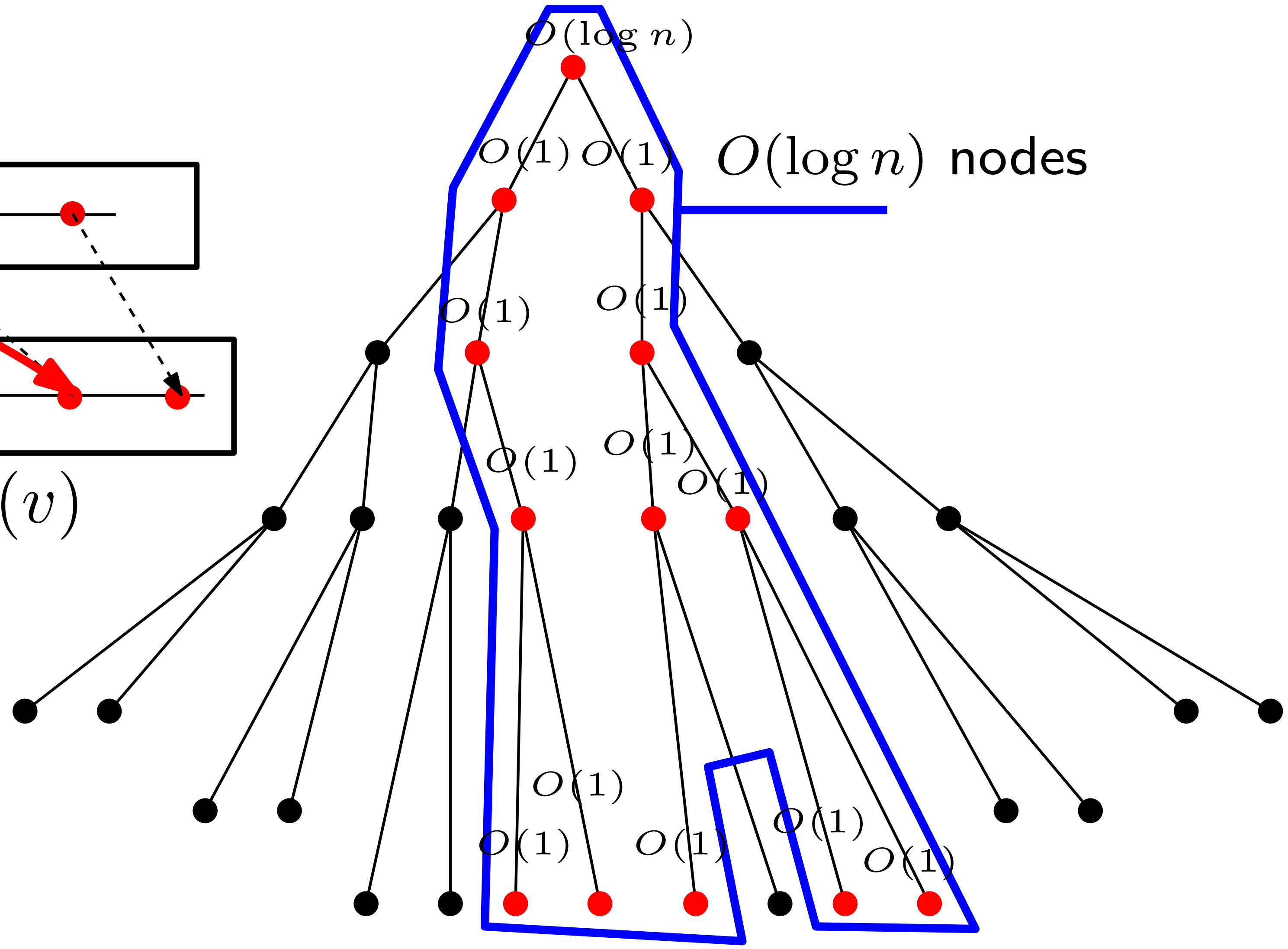
BBD-tree $O(n \log n)$ +sorting $O(n \log n) = O(n \log n)$

Query

Approximate range query $O(\log n)$ + bin. searches $O(\log^2 n) = O(\log^2 n)$

Improving query time by exploiting correlation [Lueker and Willard]





Summary and Remarks

Variant with projecting to $d-2$ dimensions

- BBD tree $+$ planar point location

Rough ($\approx d^{3/2}$) approximation algorithms

- Polynomial dependence on d

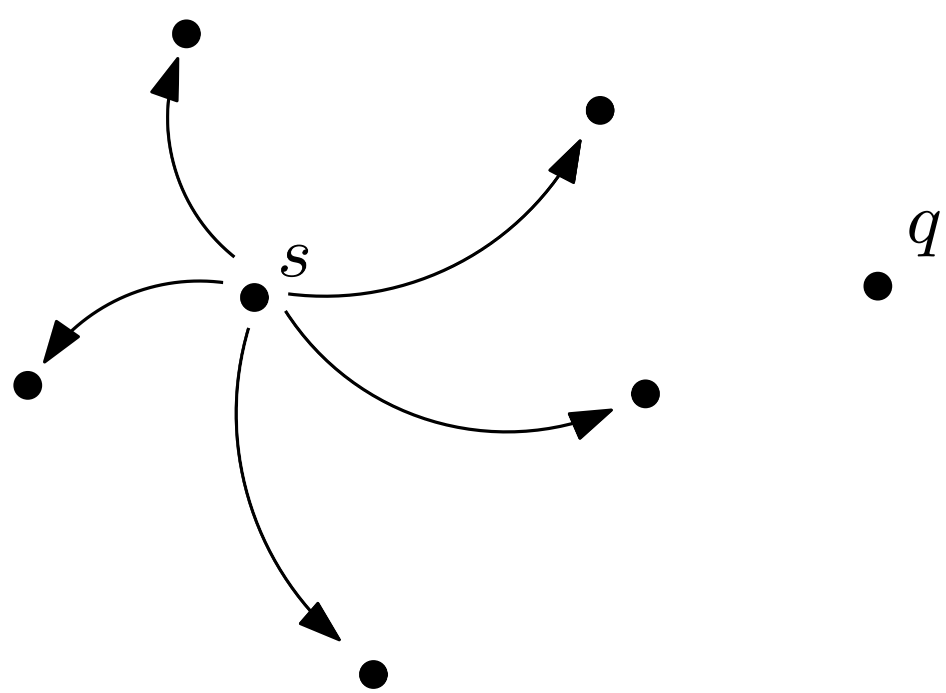
Clarkson's Algorithm: Iterative Improvement

Exact nearest neighbor problem

Data structure For each site s, a (small) list L_s of other sites such that

for **any** query point q

if s is not the nearest neighbor of q, then L_s contains a site closer to q



Clarkson's Algorithm: Iterative Improvement

Exact nearest neighbor problem

Data structure For each site s, a (small) list L_s of other sites such that

for **any** query point q

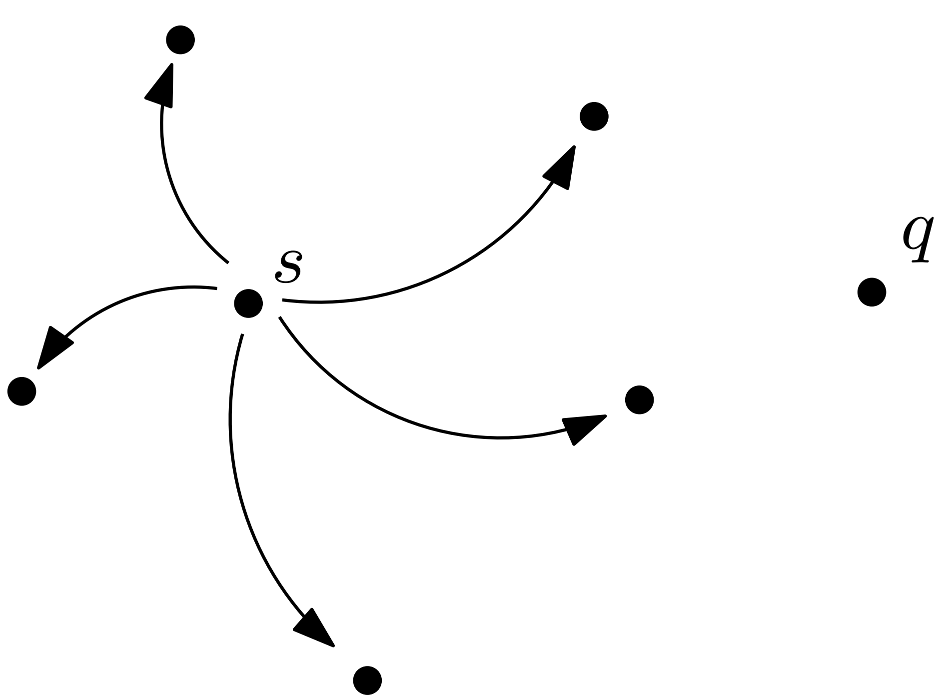
if s is not the nearest neighbor of q, then L_s contains a site closer to q

Algorithm

$s \leftarrow$ arbitrary site

while $\exists t \in L_s : \; ||t - q|| < ||s - q||$ do $s \leftarrow t$

return s



Clarkson's Algorithm: Iterative Improvement

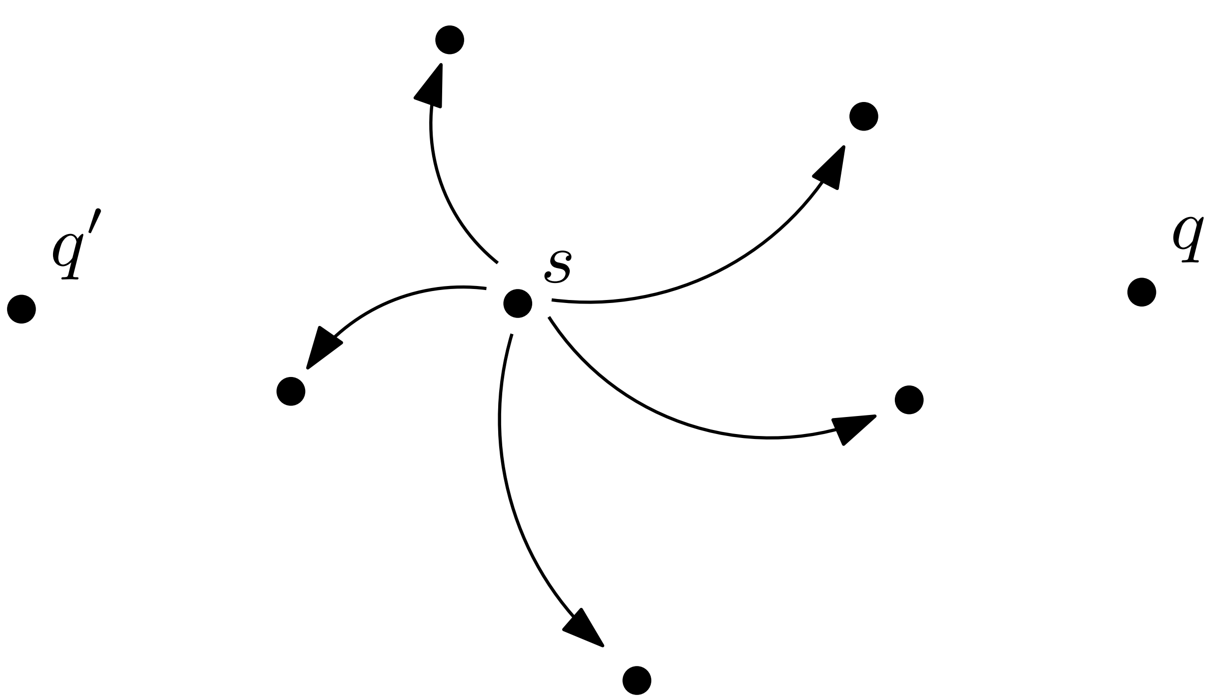
Exact nearest neighbor problem

Data structure For each site s, a (small) list L_s of other sites such that

for **any** query point q
if s is not the nearest neighbor of q, then L_s contains a site closer to q

Algorithm

$s \leftarrow$ arbitrary site
while $\exists t \in L_s : \; ||t - q|| < ||s - q||$ do $s \leftarrow t$
return s



Note
The same L_s valid for all q!

Not Useful for Exact NN

Reason 1: space complexity $\Omega(n^2)$

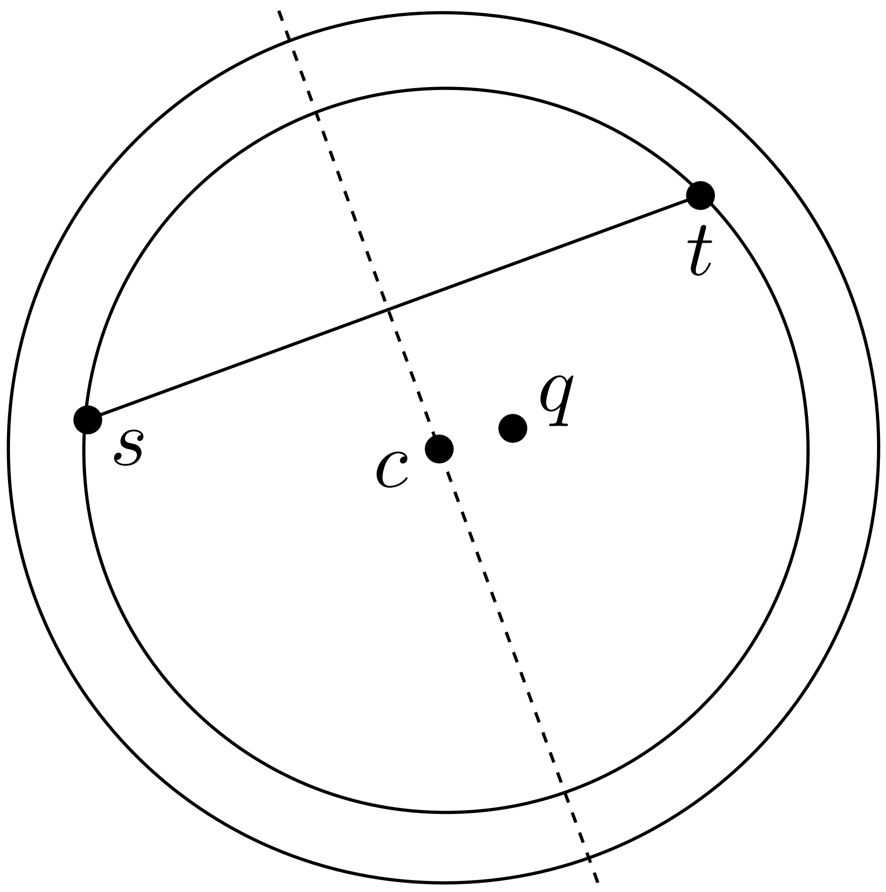
For all s, L_s has to include all Delaunay neighbors of s

For $d > 2$, Delaunay triangulation may have $\Omega(n^2)$ edges

Not Useful for Exact NN

Reason 1: space complexity $\Omega(n^2)$

For all s, L_s has to include all Delaunay neighbors of s



For $d > 2$, Delaunay triangulation may have $\Omega(n^2)$ edges

Proof:

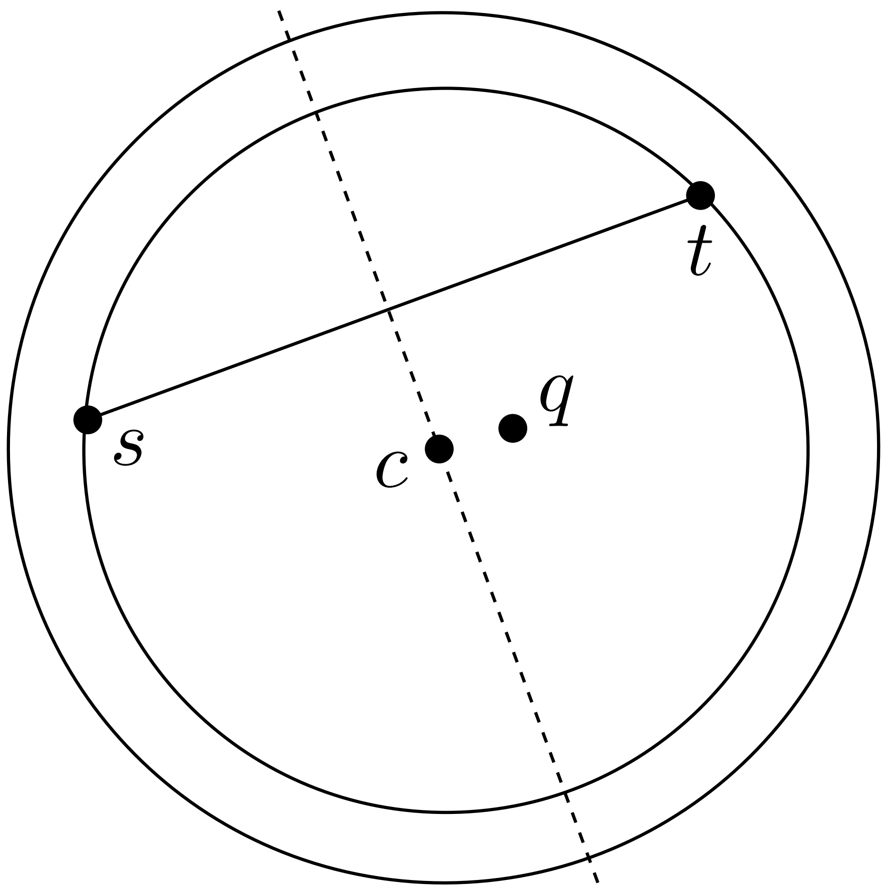
t Delaunay neighbor of s, but $t \notin L_s$

t is the only site closer to q than s

Not Useful for Exact NN

Reason 1: space complexity $\Omega(n^2)$

For all s, L_s has to include all Delaunay neighbors of s



For $d > 2$, Delaunay triangulation may have $\Omega(n^2)$ edges

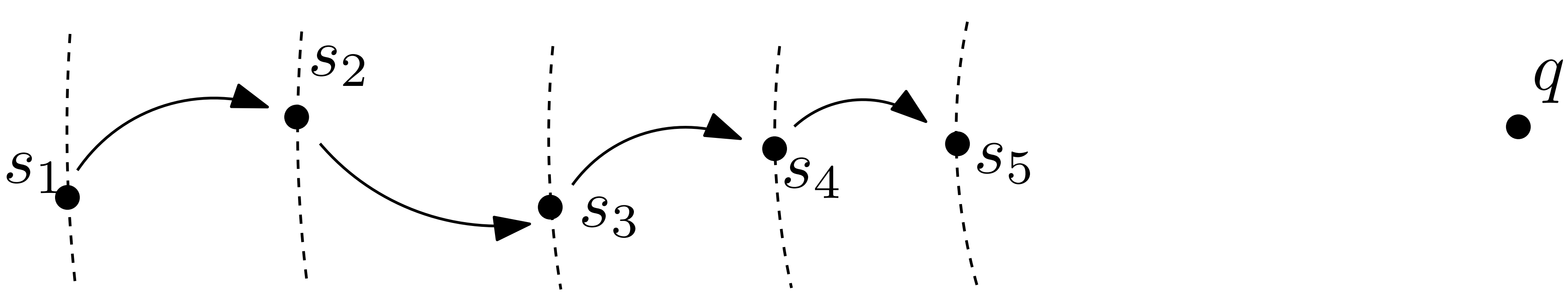
Proof:

t Delaunay neighbor of s, but $t \notin L_s$

t is the only site closer to q than s

Reason 2: query time $\Omega(n)$

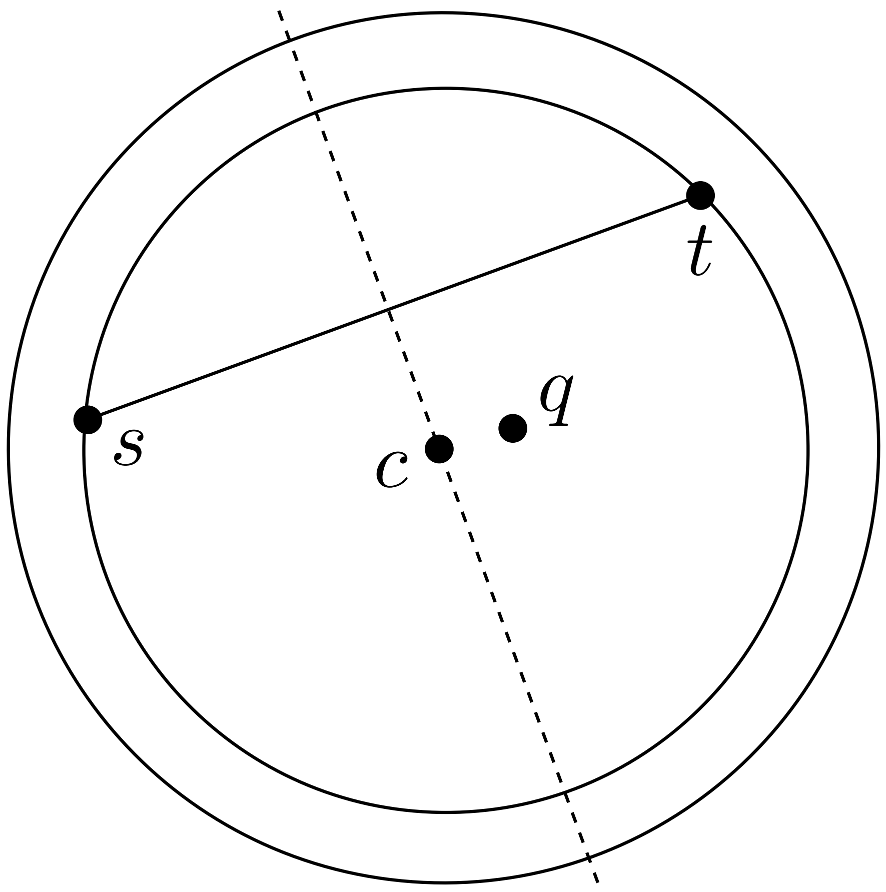
No "sufficient progress" guarantee, may have to visit all sites



Not Useful for Exact NN

Reason 1: space complexity $\Omega(n^2)$

For all s, L_s has to include all Delaunay neighbors of s



For $d > 2$, Delaunay triangulation may have $\Omega(n^2)$ edges

Proof:

t Delaunay neighbor of s, but $t \notin L_s$

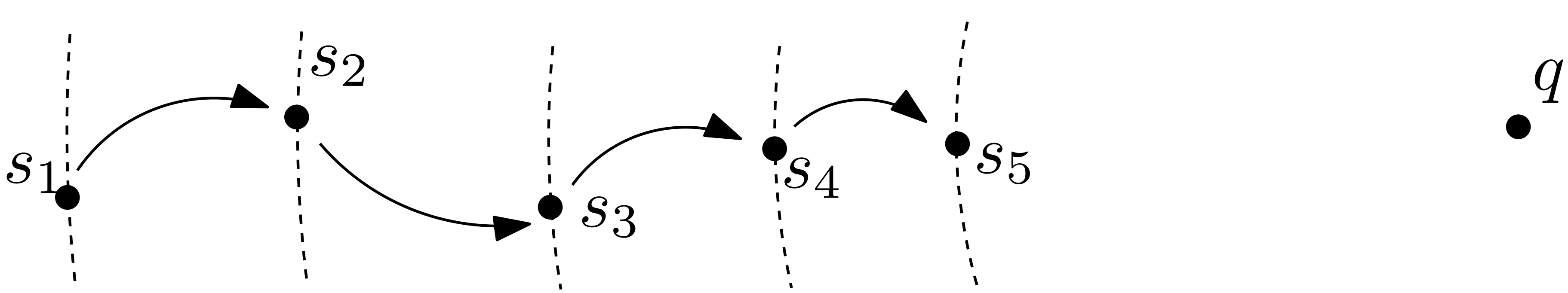
t is the only site closer to q than s

Conclusion

No improvement over the trivial algorithm!

Reason 2: query time $\Omega(n)$

No "sufficient progress" guarantee, may have to visit all sites

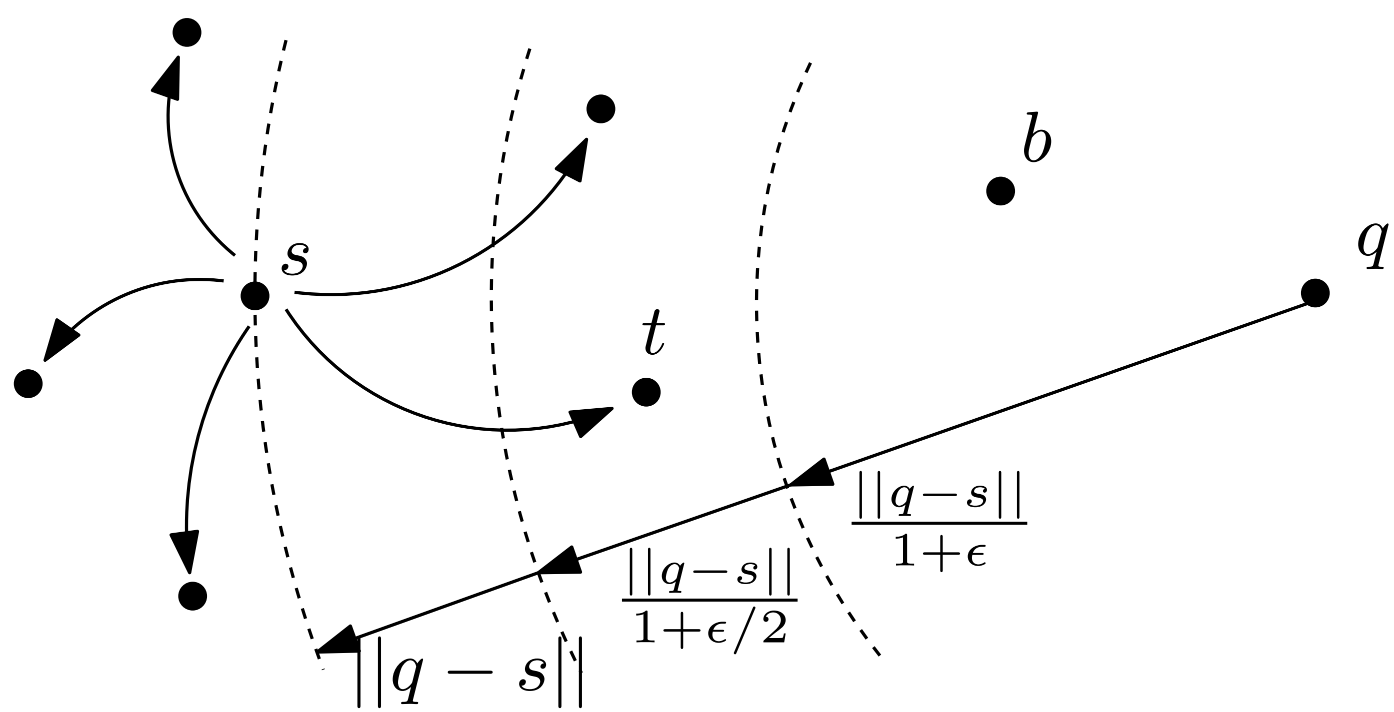


Modification for ANN

Data structure For each site s, a (small) list L_s of other sites such that

for **any** query point q

if s is not a $(1+\epsilon)$-ANN of q,
then L_s contains a site $(1+\epsilon/2)$-closer to q

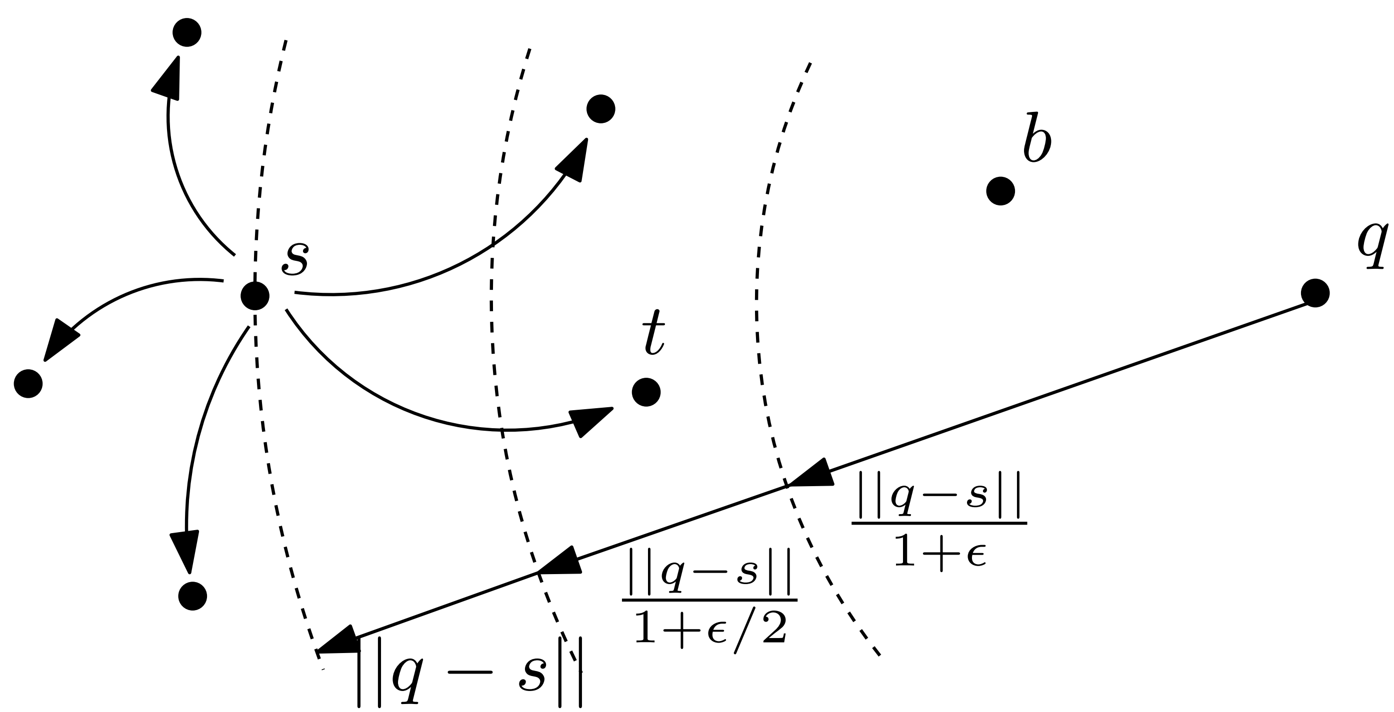


Modification for ANN

Data structure For each site s, a (small) list L_s of other sites such that

for **any** query point q

if s is not a $(1+\epsilon)$-ANN of q,
then L_s contains a site $(1+\epsilon/2)$-closer to q



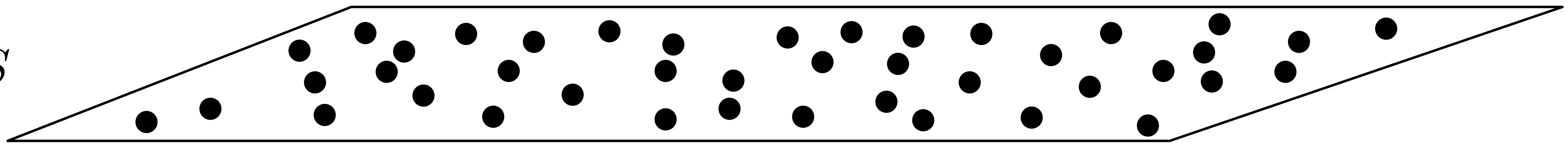
Algorithm (simple version)

$s \leftarrow$ arbitrary site
while $\exists t \in L_s$: $||q-t|| \leq \frac{||q-s||}{1+\epsilon/2}$ do $s \leftarrow t$
return s

Query Algorithm

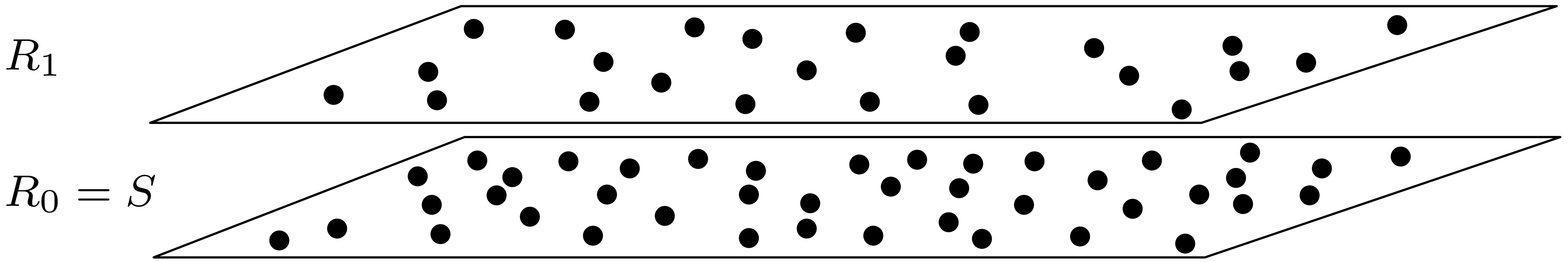
Skip list approach [Arya and Mount 1993]

$R_0 = S$



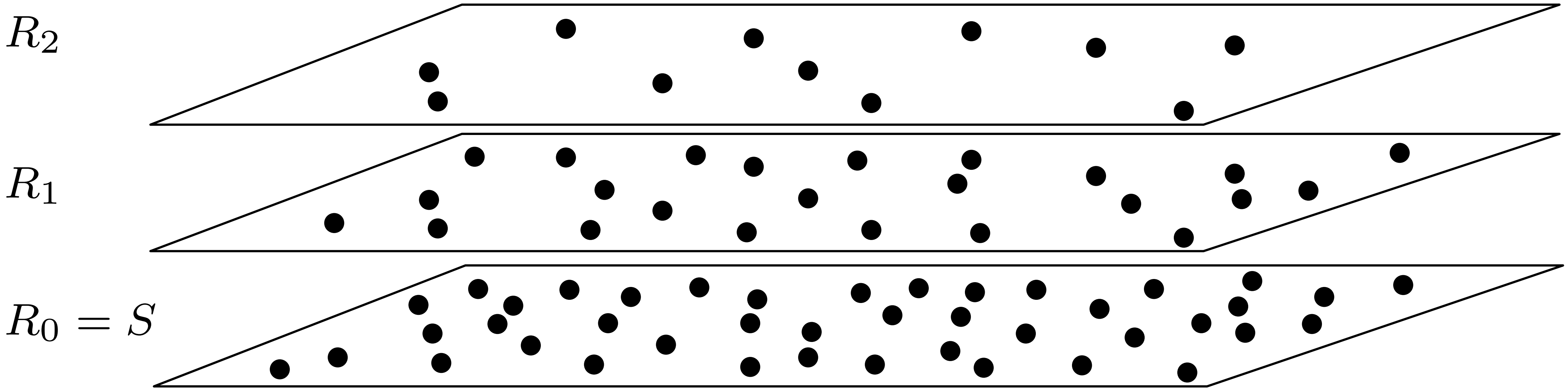
Query Algorithm

Skip list approach [Arya and Mount 1993]



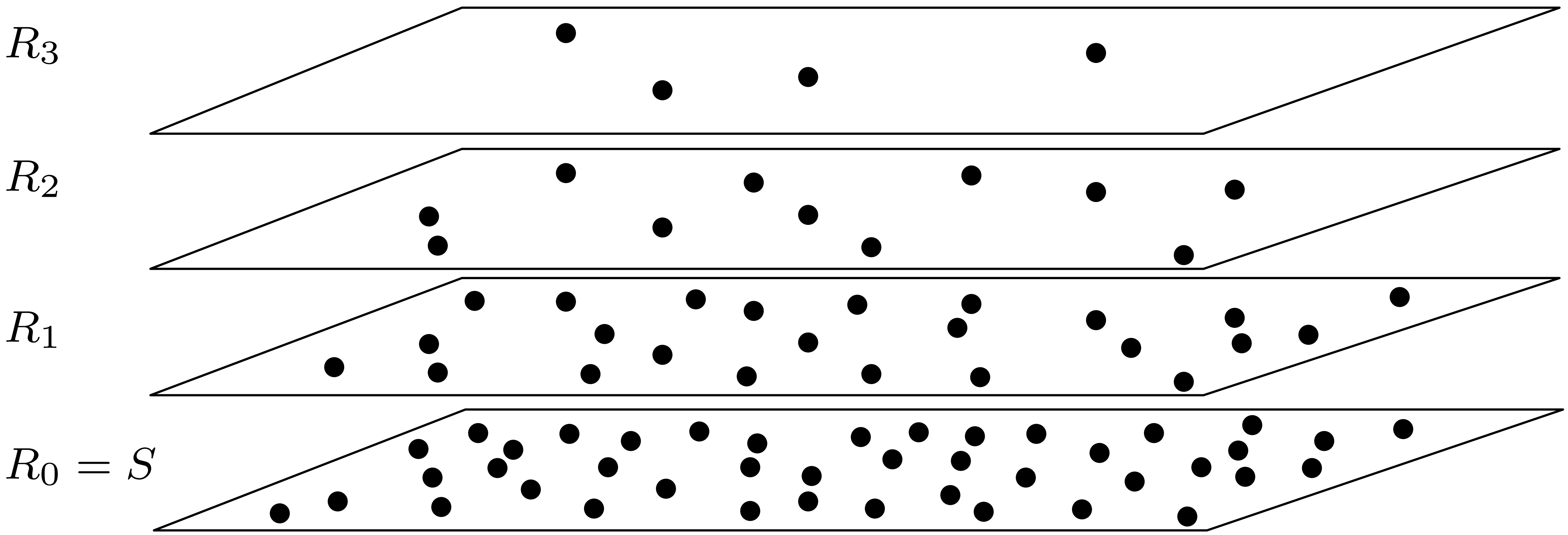
Query Algorithm

Skip list approach [Arya and Mount 1993]



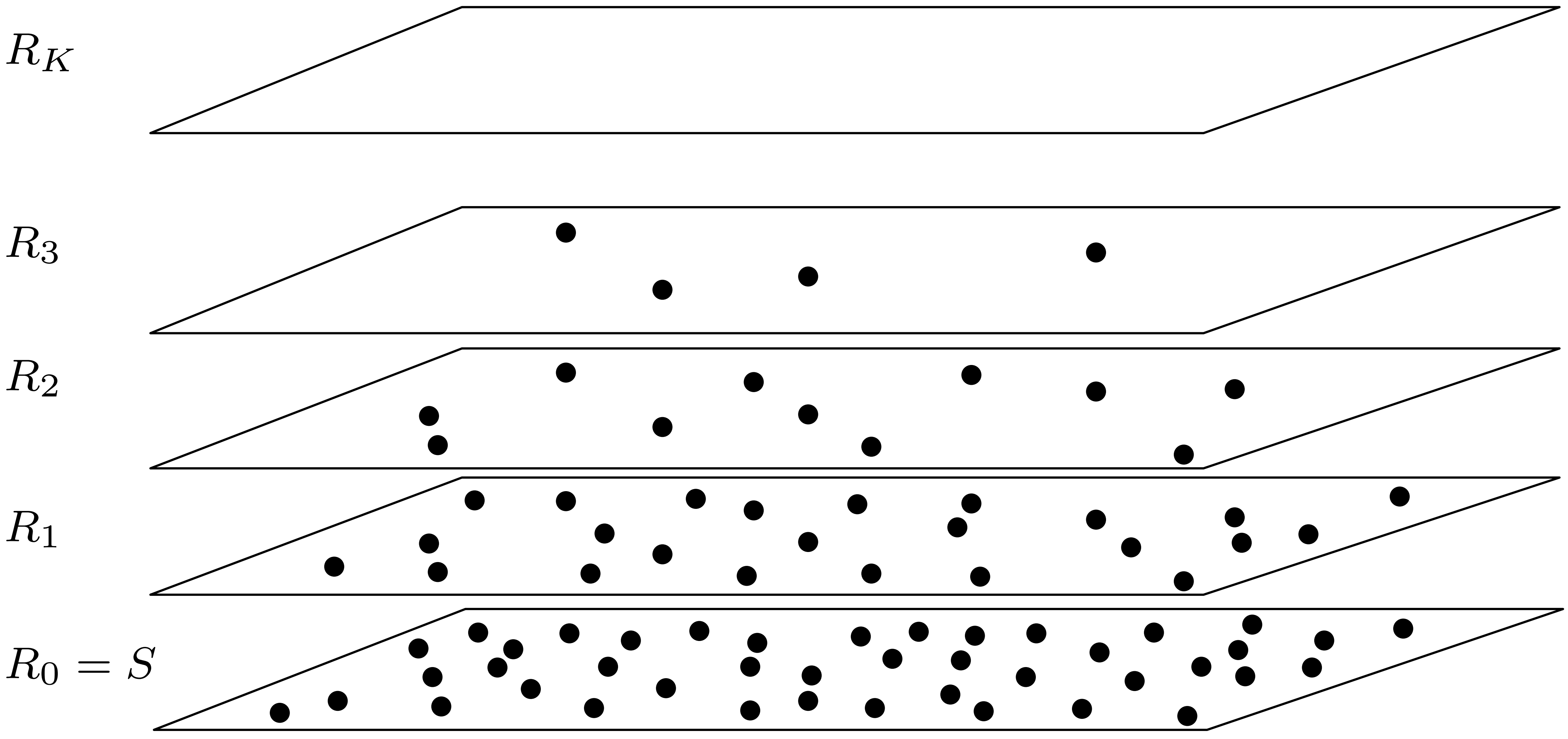
Query Algorithm

Skip list approach [Arya and Mount 1993]



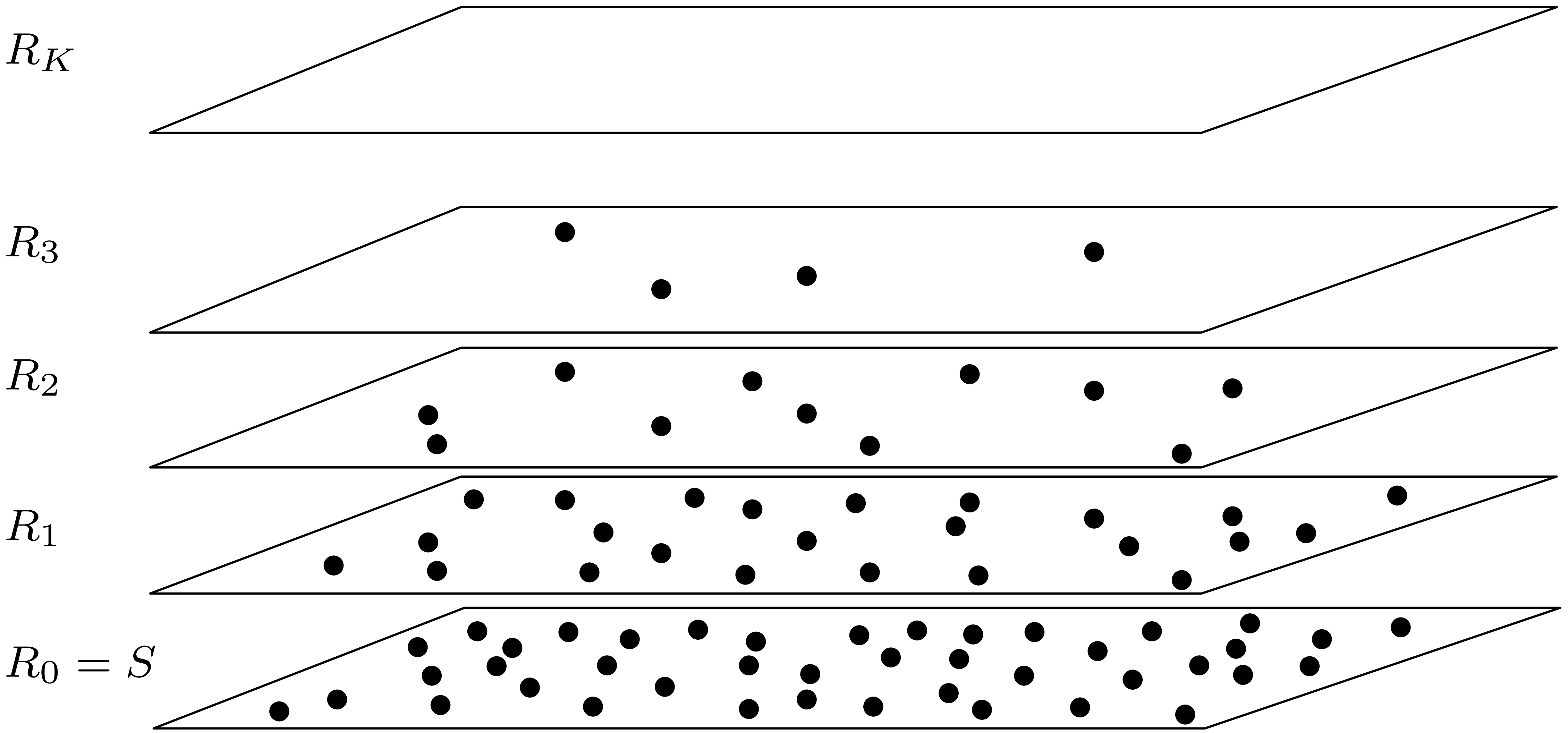
Query Algorithm

Skip list approach [Arya and Mount 1993]



Query Algorithm

Skip list approach [Arya and Mount 1993]



Algorithm

- start with any $t_{K-1} \in R_{K-1}$
- for $j = K-2, K-3, \ldots, 0$
  - find $t_j =$ $(1+\epsilon)$-ANN of q in R_j starting from t_{j+1} [using naive algorithm]
- return t_0

Query Time Analysis

Suppose that any node's list size is at most c

Observation: Query time $= c\cdot$ number of visited nodes

Compare with a regular path

- Visit nodes in the order of proximity to q, then go to the lower level

Query Time Analysis

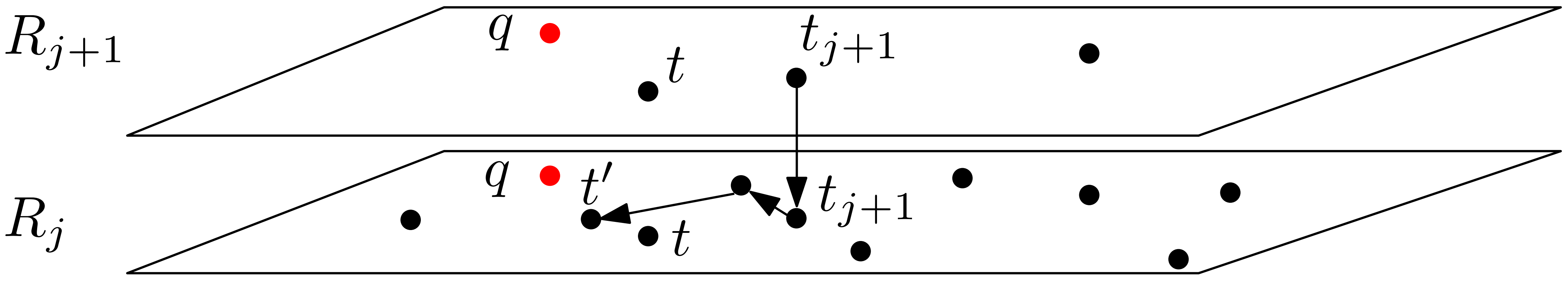
Suppose that any node's list size is at most c

Observation: Query time $= c\cdot$ number of visited nodes

Compare with a regular path

- Visit nodes in the order of proximity to q, then go to the lower level

Claim: Our path visits at most $2K$ nodes more



$$(1+\epsilon/2)^2 \geq 1+\epsilon \qquad \Rightarrow \qquad ||q-t'|| \leq ||q-t||$$

Query Time Analysis

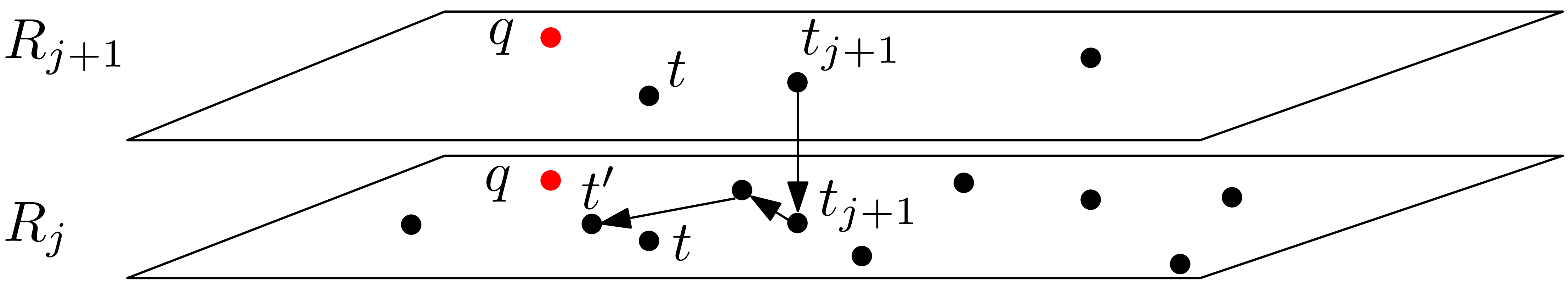
Suppose that any node's list size is at most c

Observation: Query time $= c\cdot$ number of visited nodes

Compare with a regular path

- Visit nodes in the order of proximity to q, then go to the lower level

Claim: Our path visits at most $2K$ nodes more



$$(1+\epsilon/2)^2 \geq 1+\epsilon \quad \Rightarrow \quad ||q-t'|| \leq ||q-t||$$

$\Pr[\text{regular path length} \geq C\log n] \leq O(n^{-C})$

[distribution of points across levels]
[starting search point]

Query Time Analysis

What about any q?

Skip list

n possible search targets
Probability of failure $n \cdot O(n^{-C}) = O(n^{-(C-1)})$

Query Time Analysis

What about any q?

Skip list

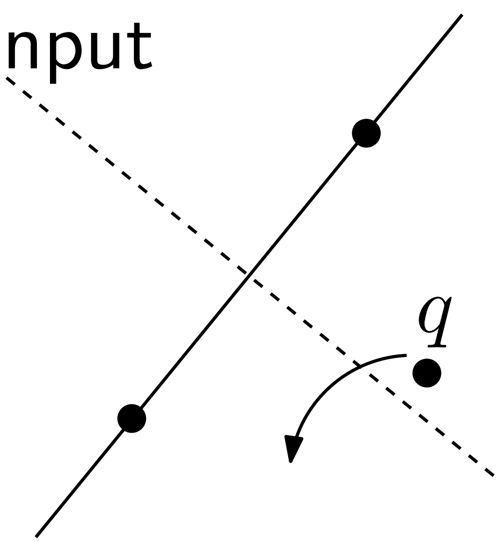
n possible search targets
Probability of failure $n \cdot O(n^{-C}) = O(n^{-(C-1)})$

Only $n^{O(d)}$ "combinatorially distinct" regular paths

- If q_1 and q_2 incude the same distance ordering on the input sites, their regular paths are the same
- Arrangement of $\binom{n}{2}$ bisecting hyperplanes has

$$\binom{\binom{n}{2}}{d} \leq (n^2)^d = n^{2d}$$

d-dimensional cells



Query Time Analysis

What about any q?

Skip list

n possible search targets
Probability of failure $n \cdot O(n^{-C}) = O(n^{-(C-1)})$

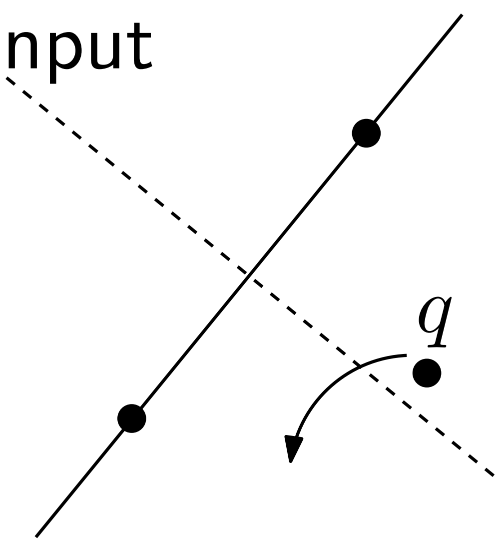
Only $n^{O(d)}$ "combinatorially distinct" regular paths

- If q_1 and q_2 incude the same distance ordering on the input sites, their regular paths are the same

- Arrangement of $\binom{n}{2}$ bisecting hyperplanes has

$$\binom{\binom{n}{2}}{d} \leq (n^2)^d = n^{2d}$$

d-dimensional cells



Setting $C = 2d + C'$

$$\Pr[\text{regular path length} \leq O(d) \log n] = O(n^{-C'})$$

Weighted Voronoi Diagrams

Goal For each site s, compute L_s such that

$\forall q \in \mathbb{R}^d$

$$\forall b \in S: \ ||q-b|| \geq \frac{||q-s||}{1+\epsilon} \qquad \Leftarrow \qquad \forall t \in L_s: \ ||q-t|| \geq \frac{||q-s||}{1+\epsilon/2}$$

[s is an $(1+\epsilon)$-ANN of q] [no "improvement" in L_s]

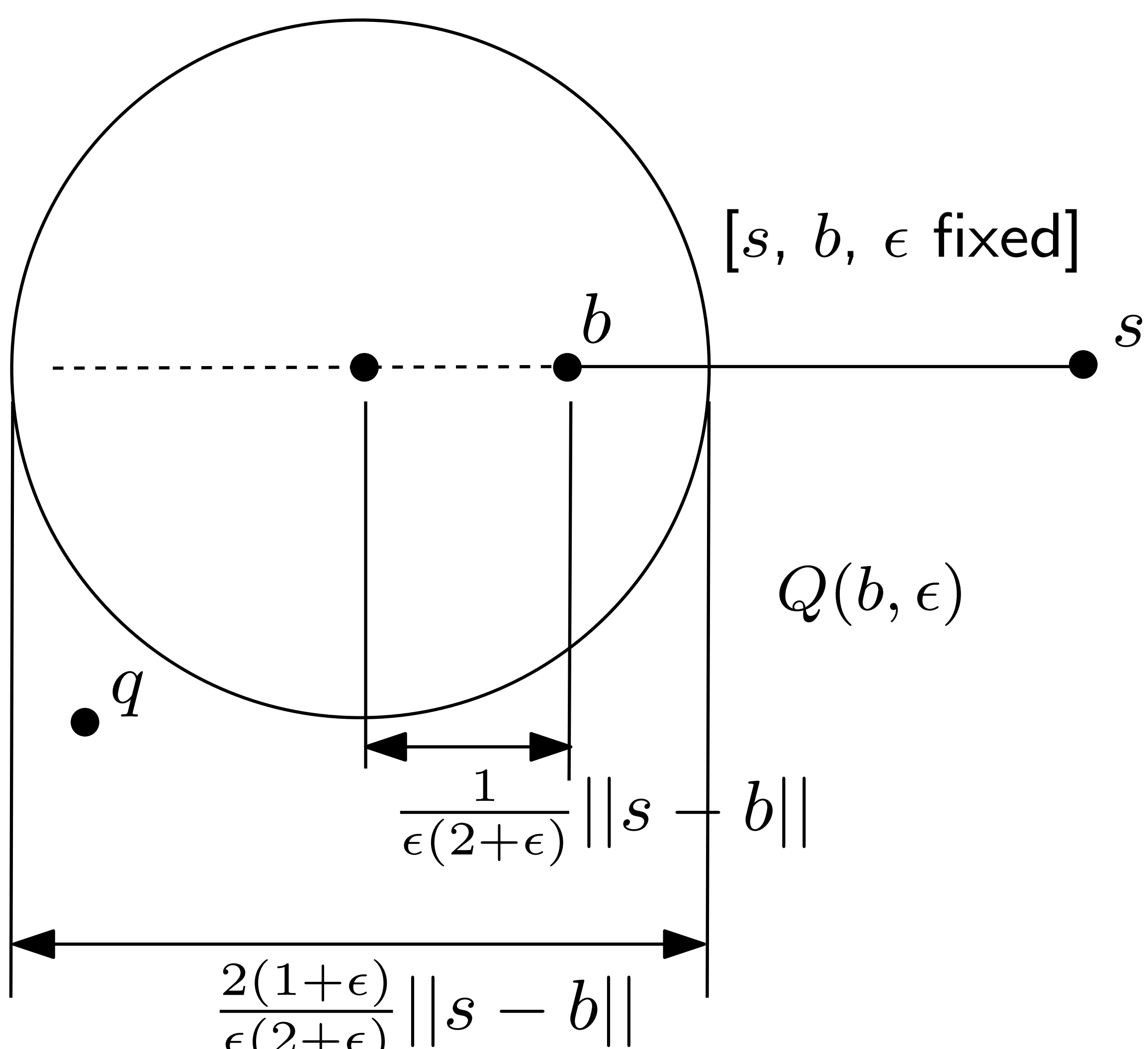
Weighted Voronoi Diagrams

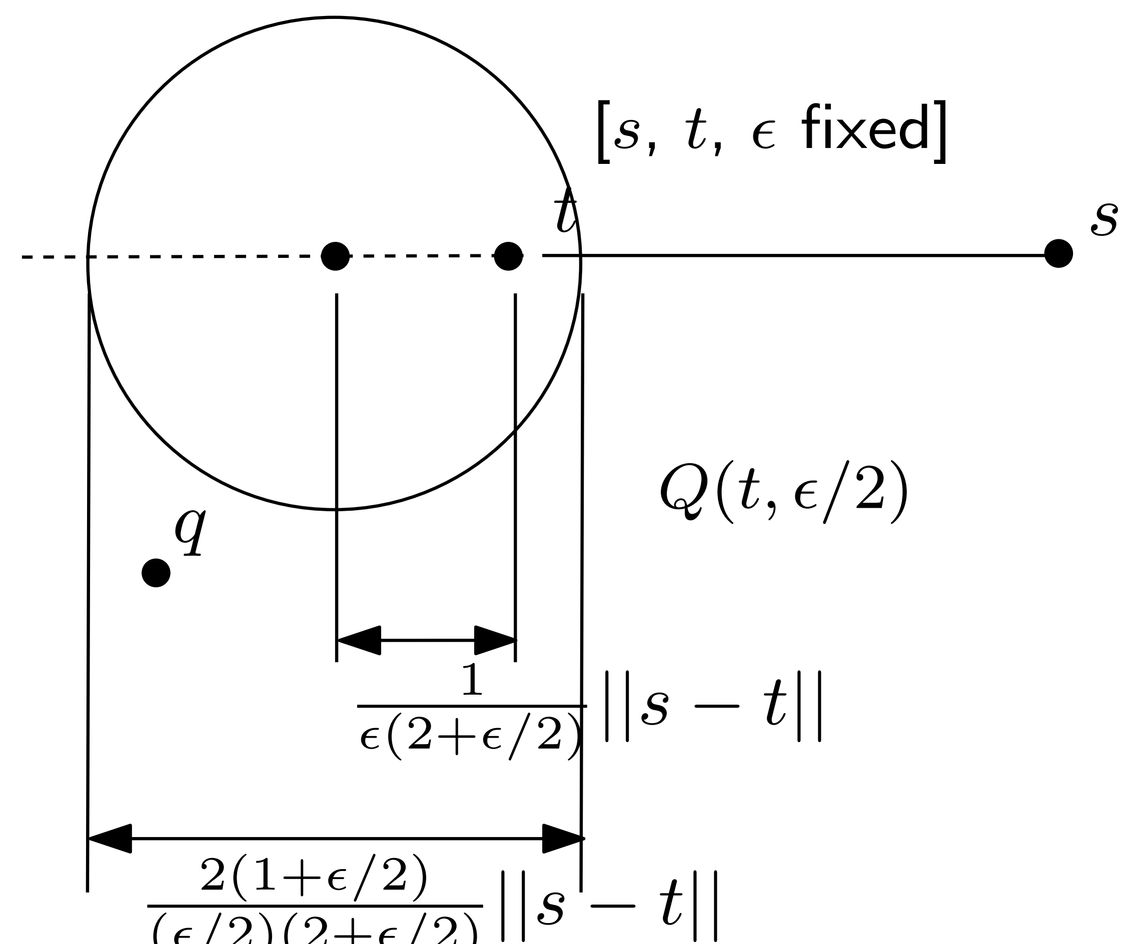
Goal For each site s, compute L_s such that

$\forall q \in \mathbb{R}^d$

$$\forall b \in S: \ ||q-b|| \geq \frac{||q-s||}{1+\epsilon} \qquad \Leftarrow \qquad \forall t \in L_s: \ ||q-t|| \geq \frac{||q-s||}{1+\epsilon/2}$$

[s is an $(1+\epsilon)$-ANN of q] [no "improvement" in L_s]





Weighted Voronoi Diagrams

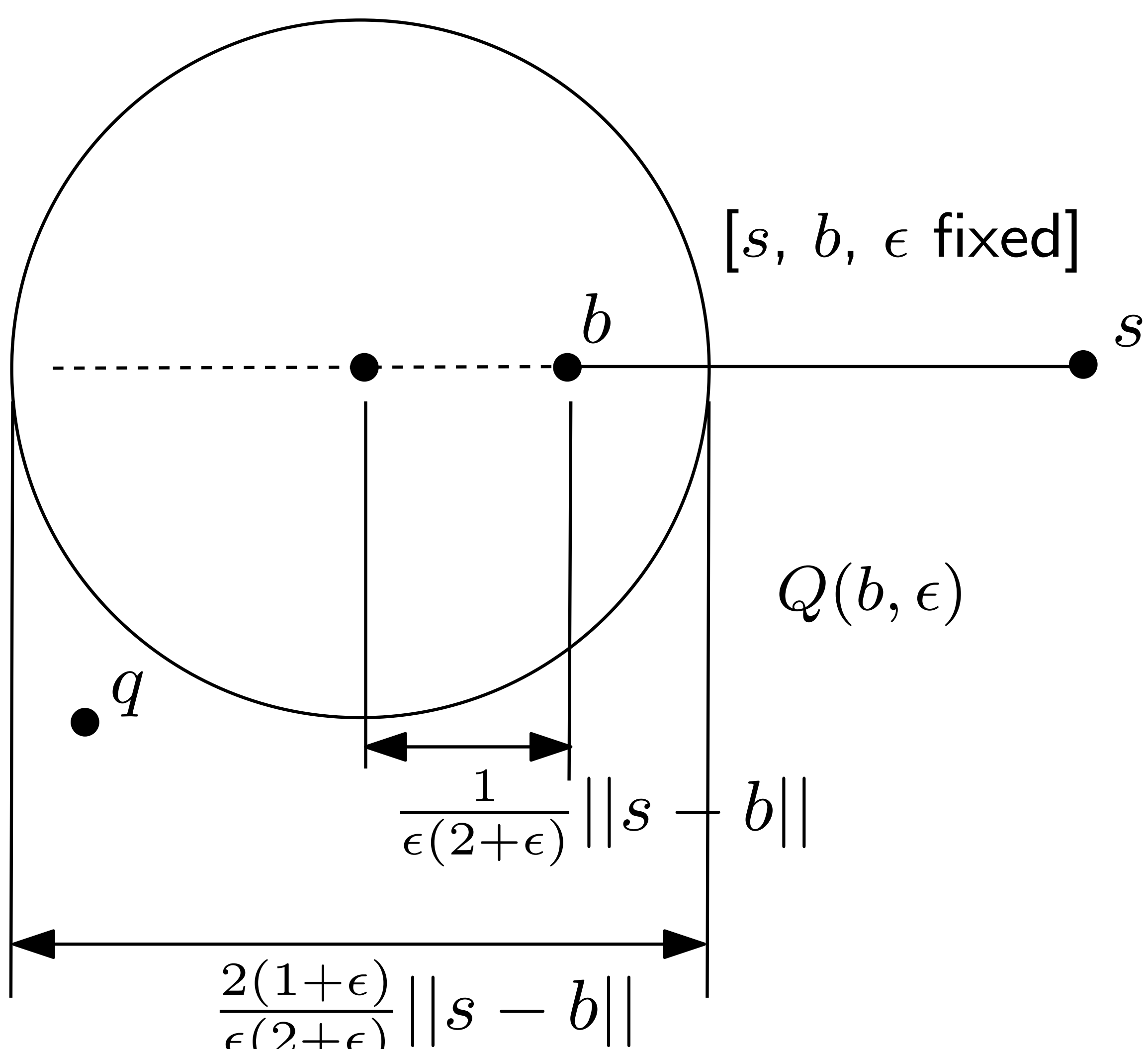
Goal For each site s, compute L_s such that

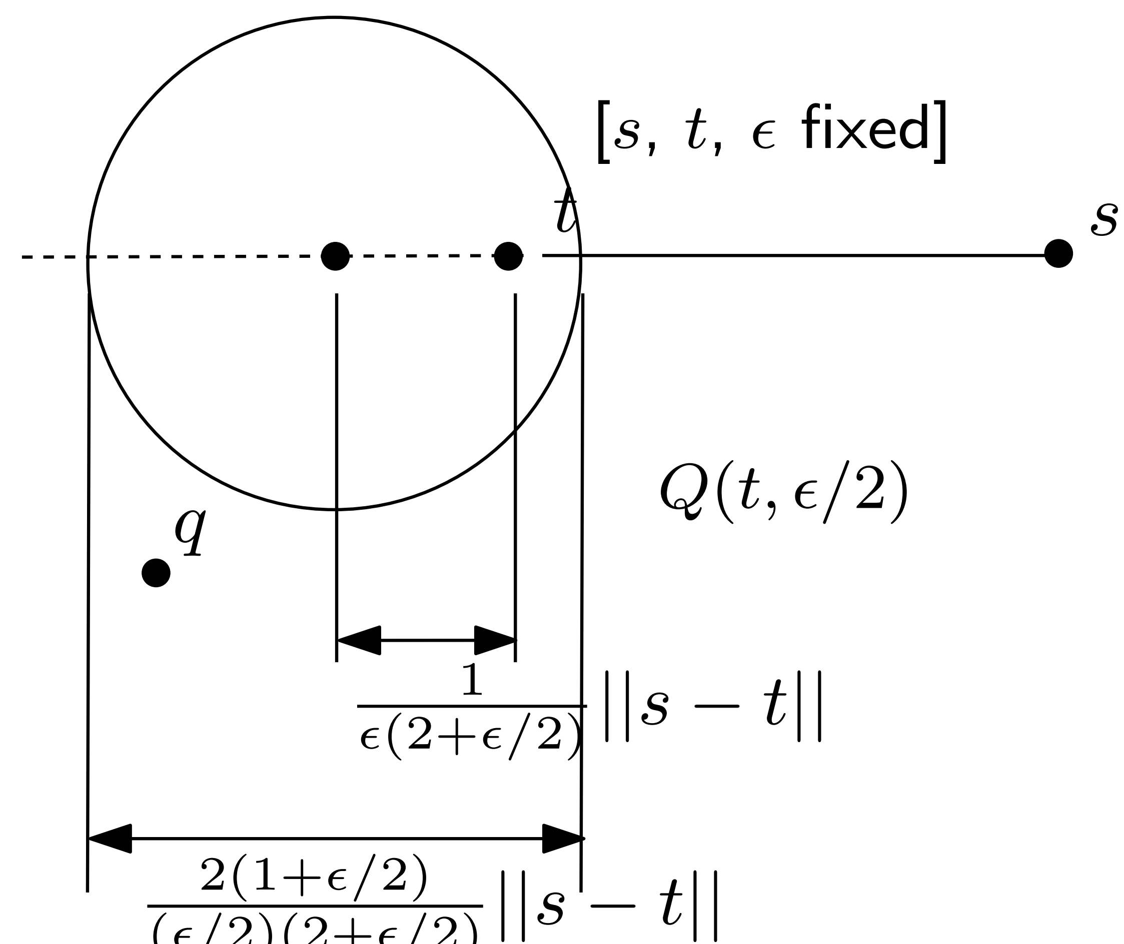
$\forall q \in \mathbb{R}^d$

$$\forall b \in S: \ ||q-b|| \geq \frac{||q-s||}{1+\epsilon} \quad \Leftarrow \quad \forall t \in L_s: \ ||q-t|| \geq \frac{||q-s||}{1+\epsilon/2}$$

[s is an $(1+\epsilon)$-ANN of q] [no "improvement" in L_s]

$$\forall b \in S: \ q \in Q(b,\epsilon) \quad \Leftarrow \quad \forall t \in L_s: \ q \in Q(t,\epsilon/2)$$





Weighted Voronoi Diagrams

Goal For each site s, compute L_s such that

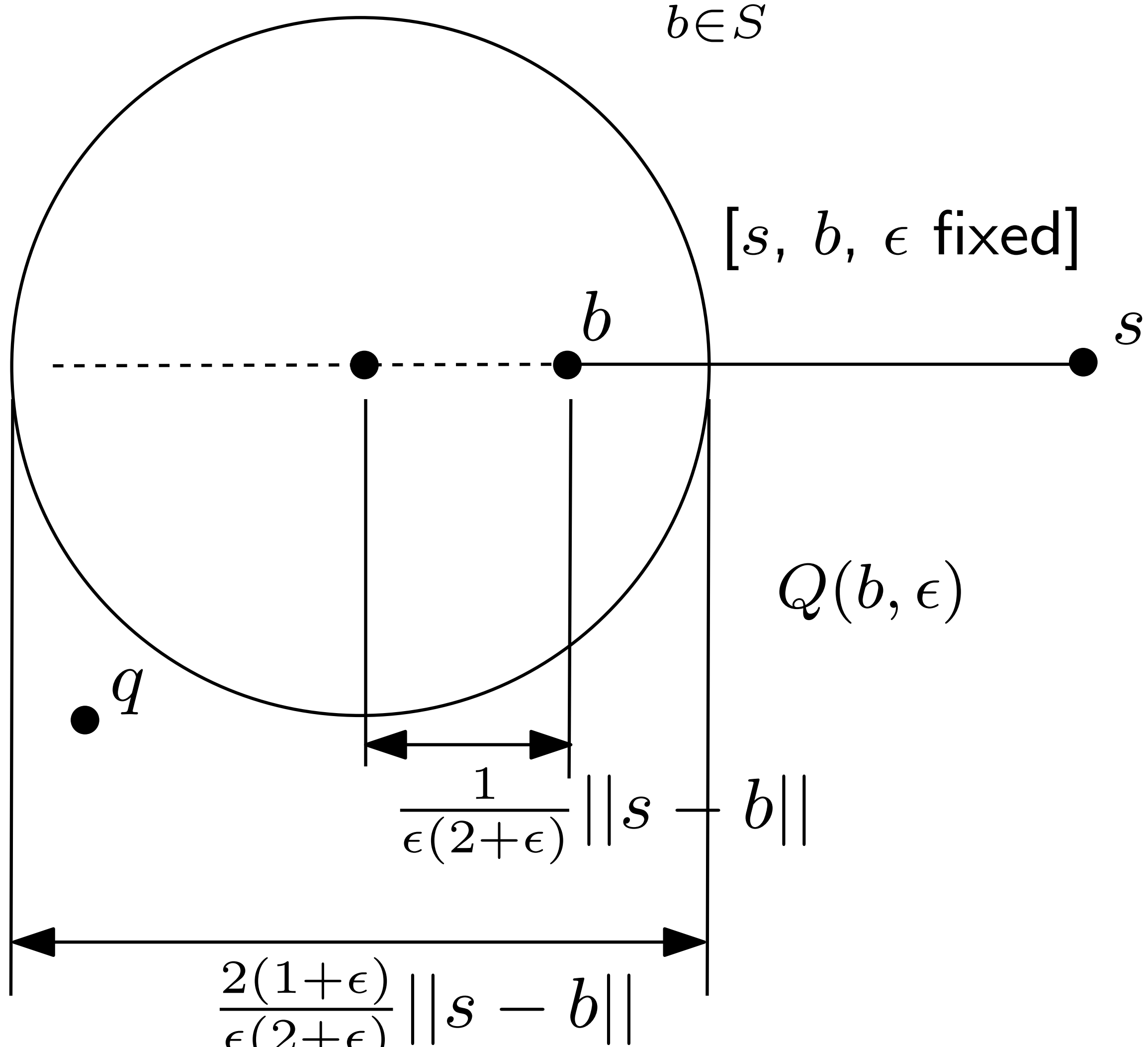
$\forall q \in \mathbb{R}^d$

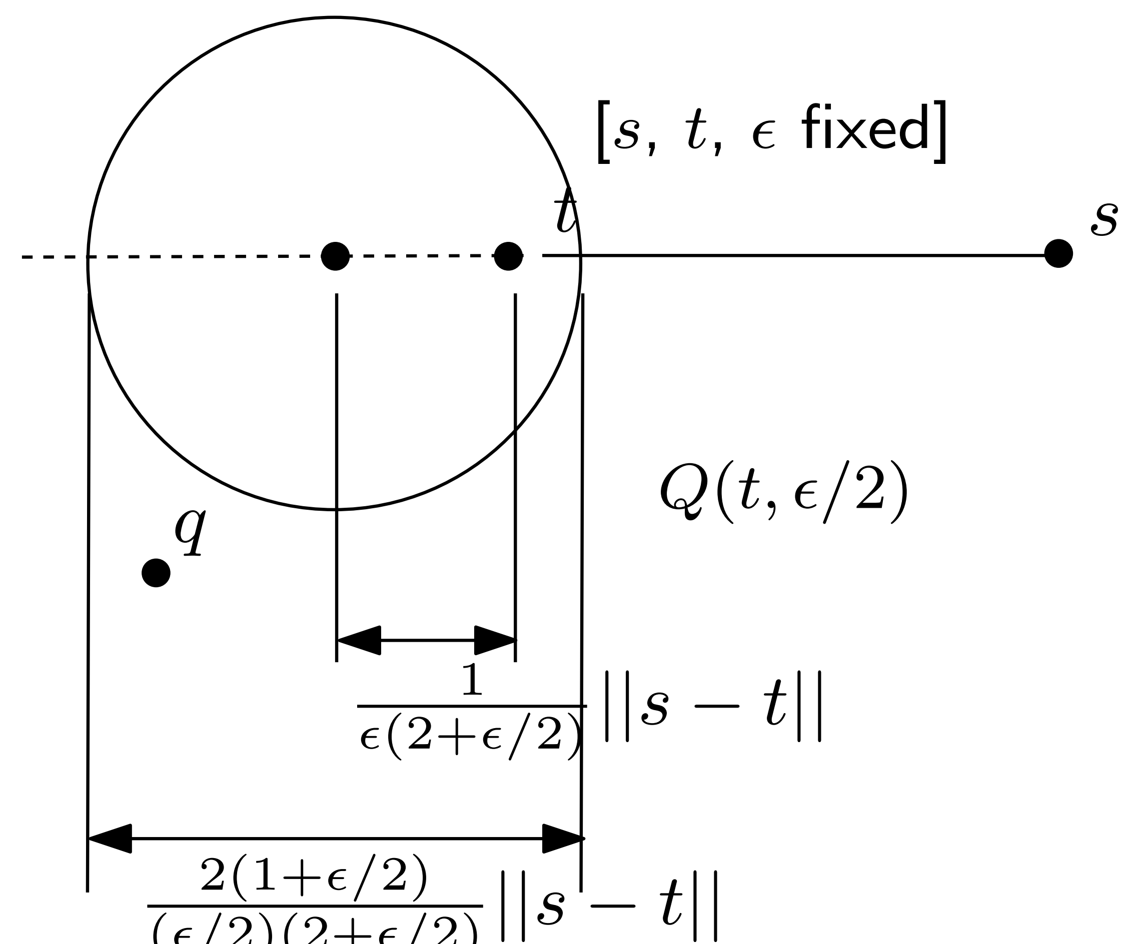
$$\forall b \in S: \ ||q-b|| \geq \frac{||q-s||}{1+\epsilon} \quad \Leftarrow \quad \forall t \in L_s: \ ||q-t|| \geq \frac{||q-s||}{1+\epsilon/2}$$

[s is an $(1+\epsilon)$-ANN of q] [no "improvement" in L_s]

$$\forall b \in S: \ q \in Q(b,\epsilon) \quad \Leftarrow \quad \forall t \in L_s: \ q \in Q(t,\epsilon/2)$$

$$\bigcap_{b \in S} Q(b,\epsilon) \quad \supseteq \quad \bigcap_{t \in L_s} Q(t,\epsilon/2)$$





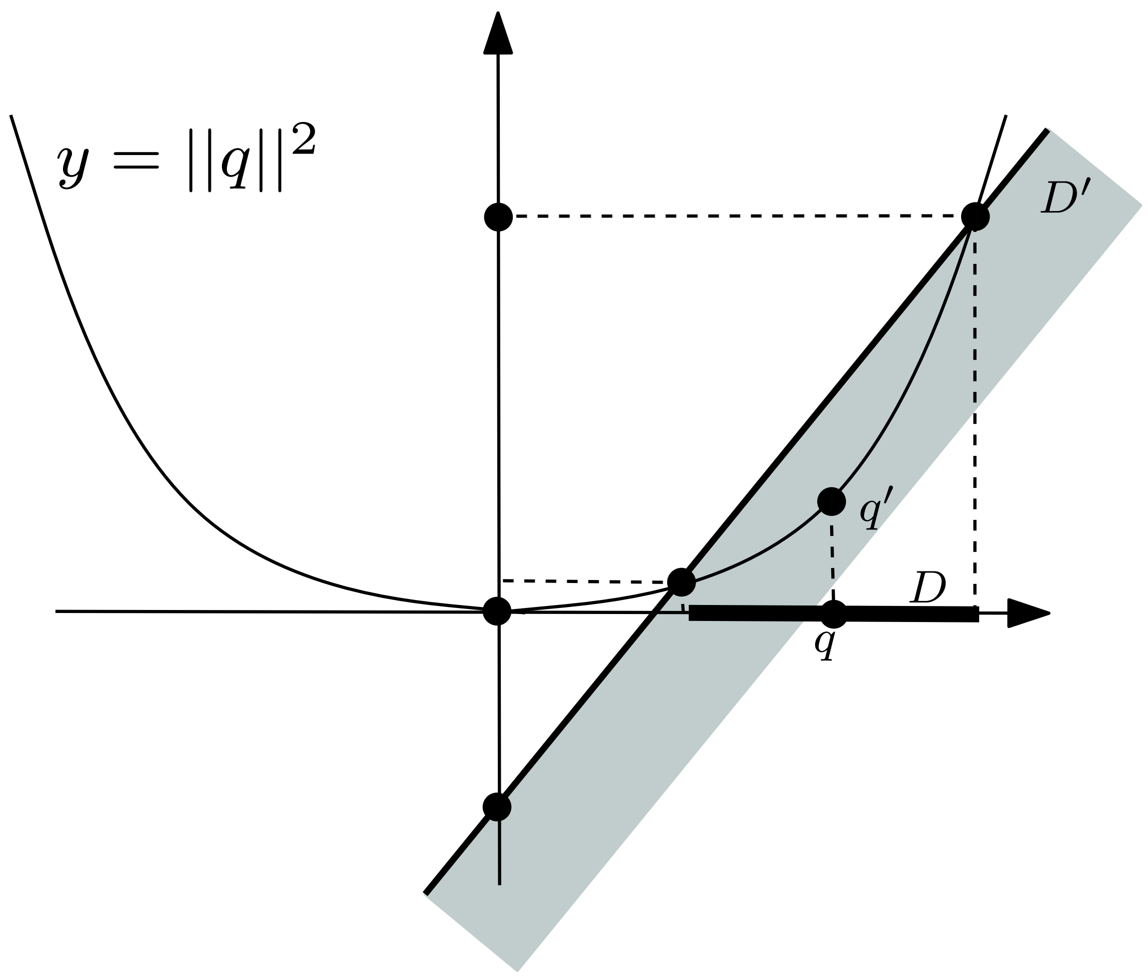
Linearization ("Lifting")

A point inside/outside a sphere in $\mathbb{R}^d$?

$\Updownarrow$

A point above/below a hyperplane in $\mathbb{R}^{d+1}$?

Example for d=1



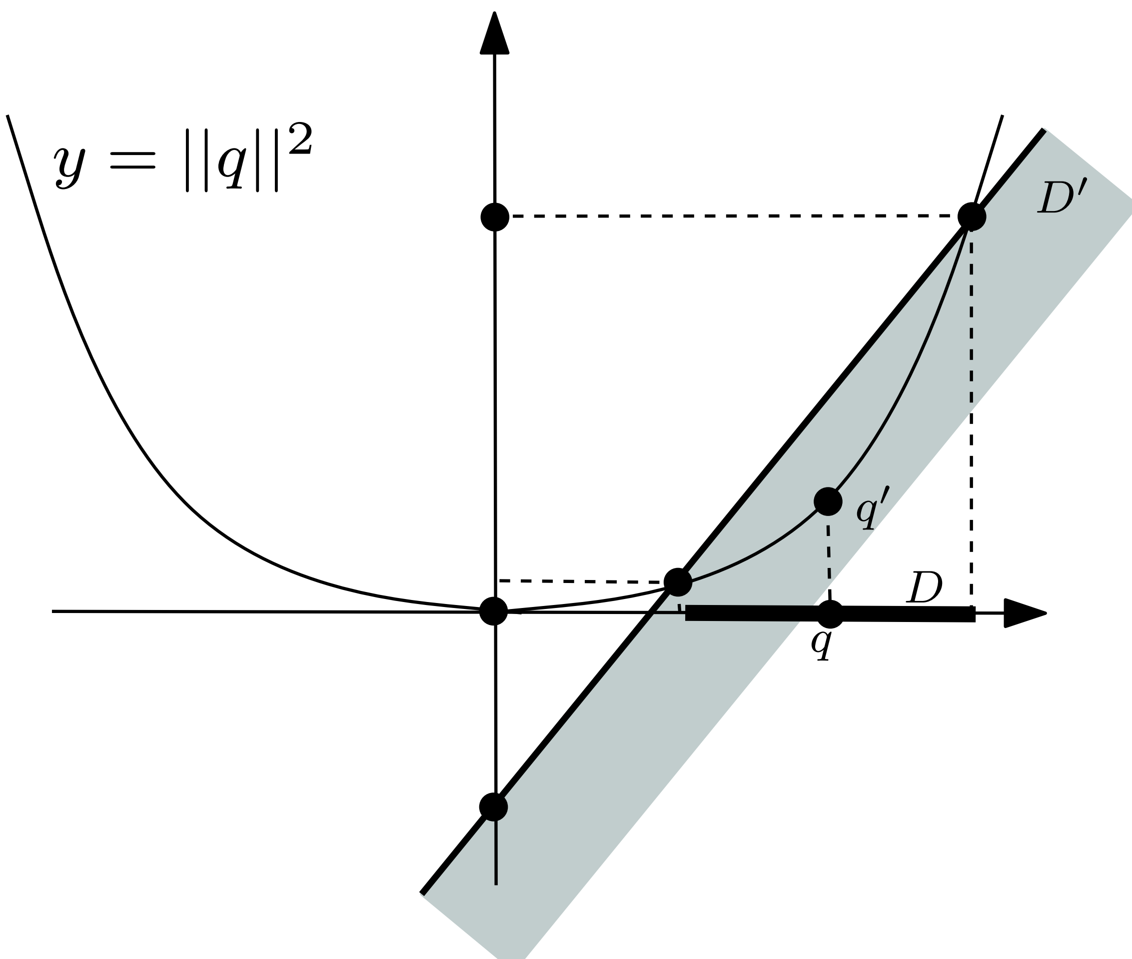
Linearization ("Lifting")

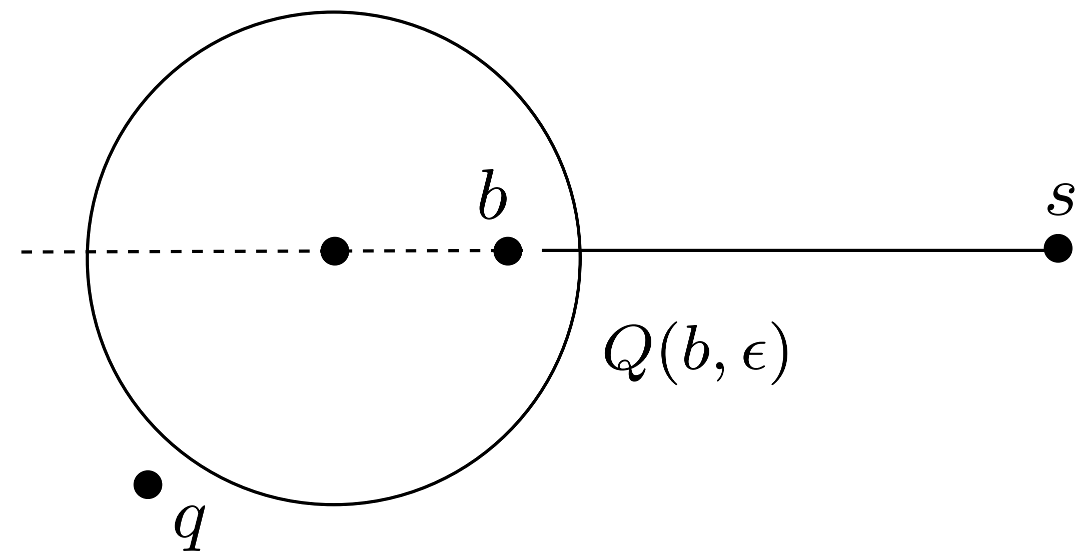
A point inside/outside a sphere in $\mathbb{R}^d$?

$\Updownarrow$

A point above/below a hyperplane in $\mathbb{R}^{d+1}$?

Example for d=1





$$Q(b,\epsilon) = \{q \in \mathbb{R}^d : \; ||q-s|| \leq (1+\epsilon)||q-b||\}$$

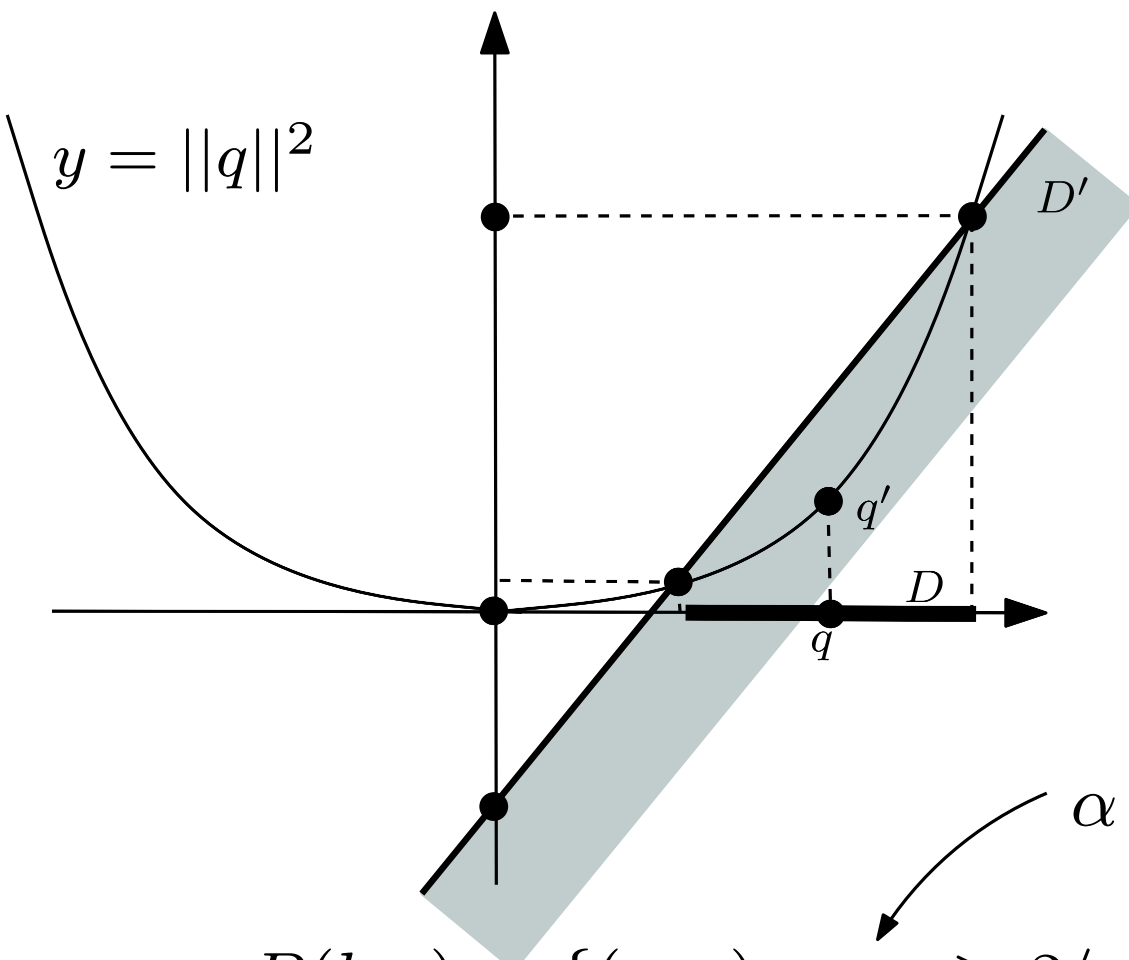
Linearization ("Lifting")

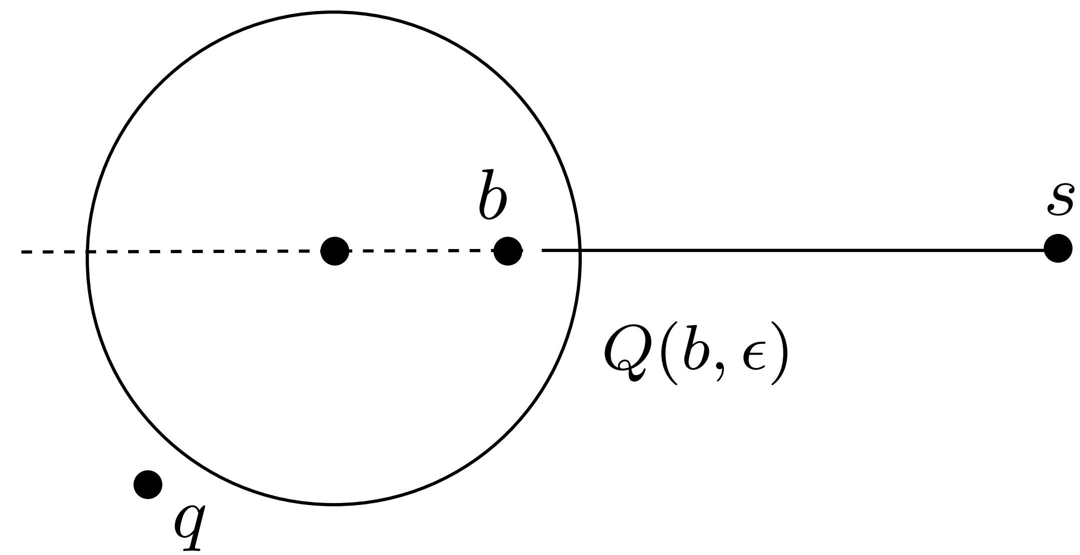
A point inside/outside a sphere in $\mathbb{R}^d$?

$\Updownarrow$

A point above/below a hyperplane in $\mathbb{R}^{d+1}$?

Example for d=1





$$Q(b, \epsilon) = \{q \in \mathbb{R}^d : \ ||q - s|| \leq (1 + \epsilon)||q - b||\}$$

$\alpha \approx 2\epsilon$

$$P(b, \epsilon) = \underbrace{\{(q, y) : \ \alpha y \geq 2\langle q, b\rangle - ||b||^2\}}_{H(b,\epsilon)} \cap \underbrace{\{(q, y) : \ y = ||q||^2\}}_{\Psi}$$

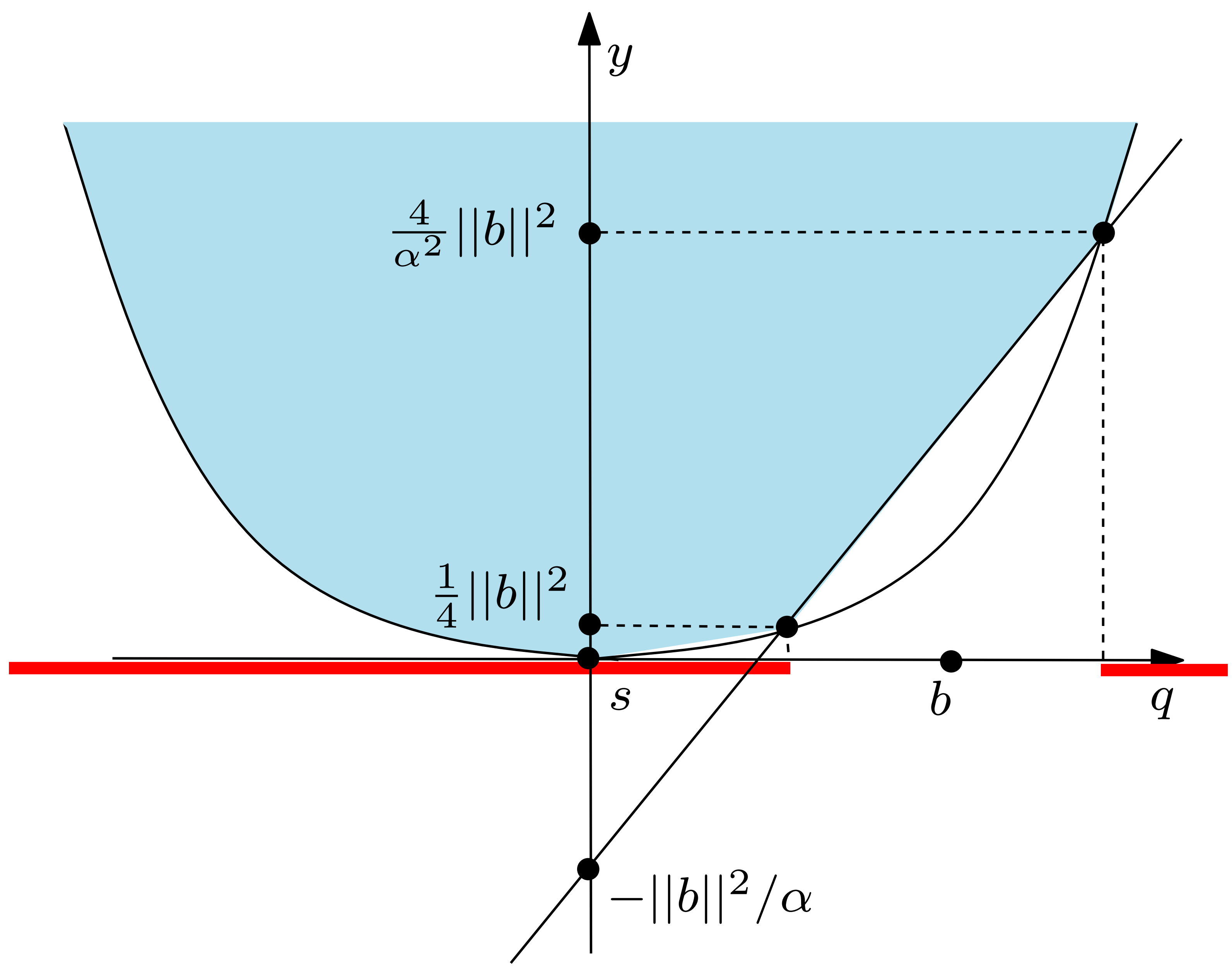
$H(b, \epsilon)$, halfspace in $\mathbb{R}^{d+1}$
(**note:** contains the origin)

Ψ, standard paraboloid in $\mathbb{R}^{d+1}$
(**note:** independent of b, ϵ)

Final Formulation

Paraboloid

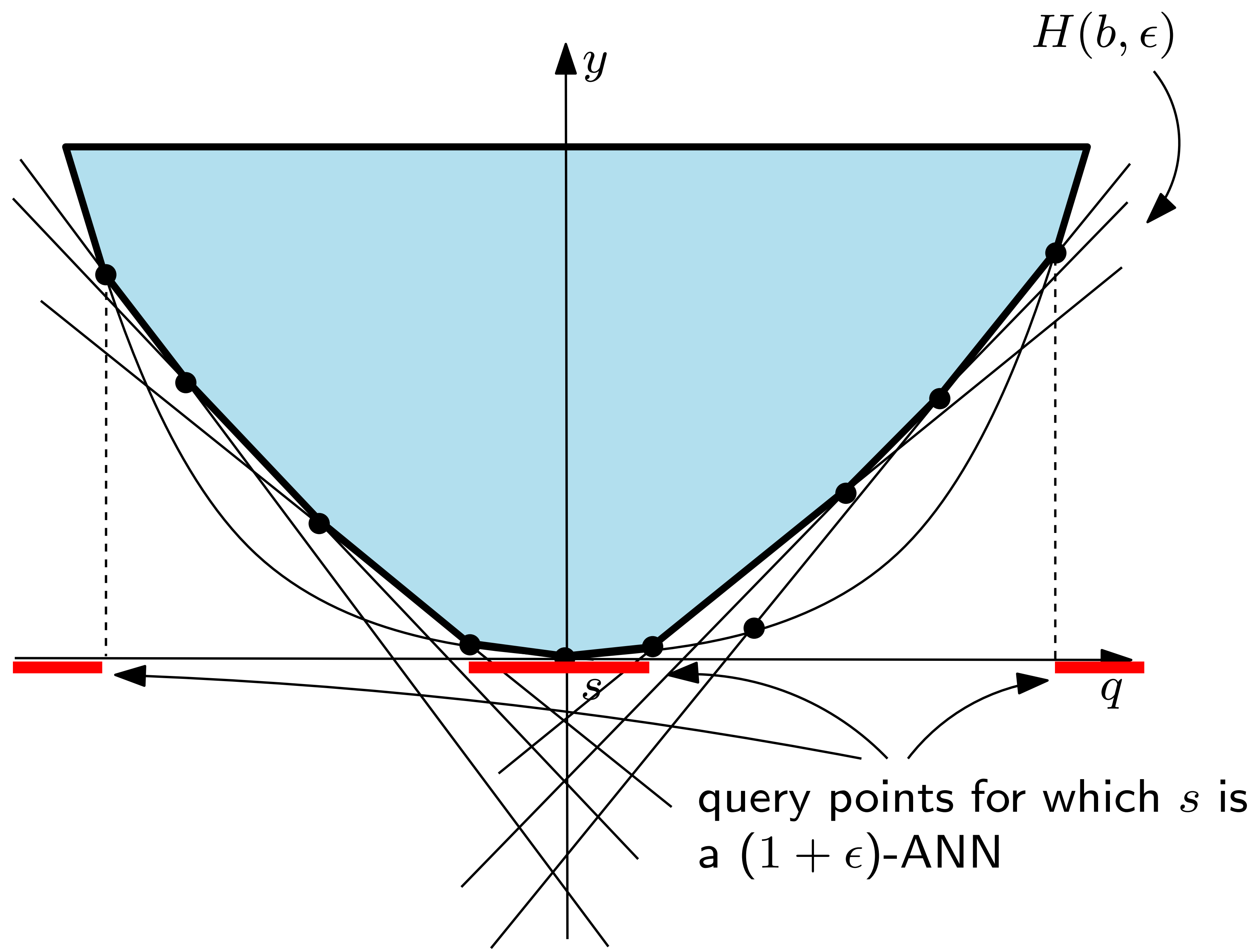
$\Psi = \{(q, y) : \; y = ||q||^2\}$



Final Formulation

Paraboloid
$\Psi = \{(q, y) : \; y = ||q||^2\}$
Halfspaces
$H(b, \epsilon) = \{(q, y) : \; \alpha y \geq 2\langle b, q\rangle - ||b||^2\}$
for all $b \in S$

[can compute using S and ϵ]



Final Formulation

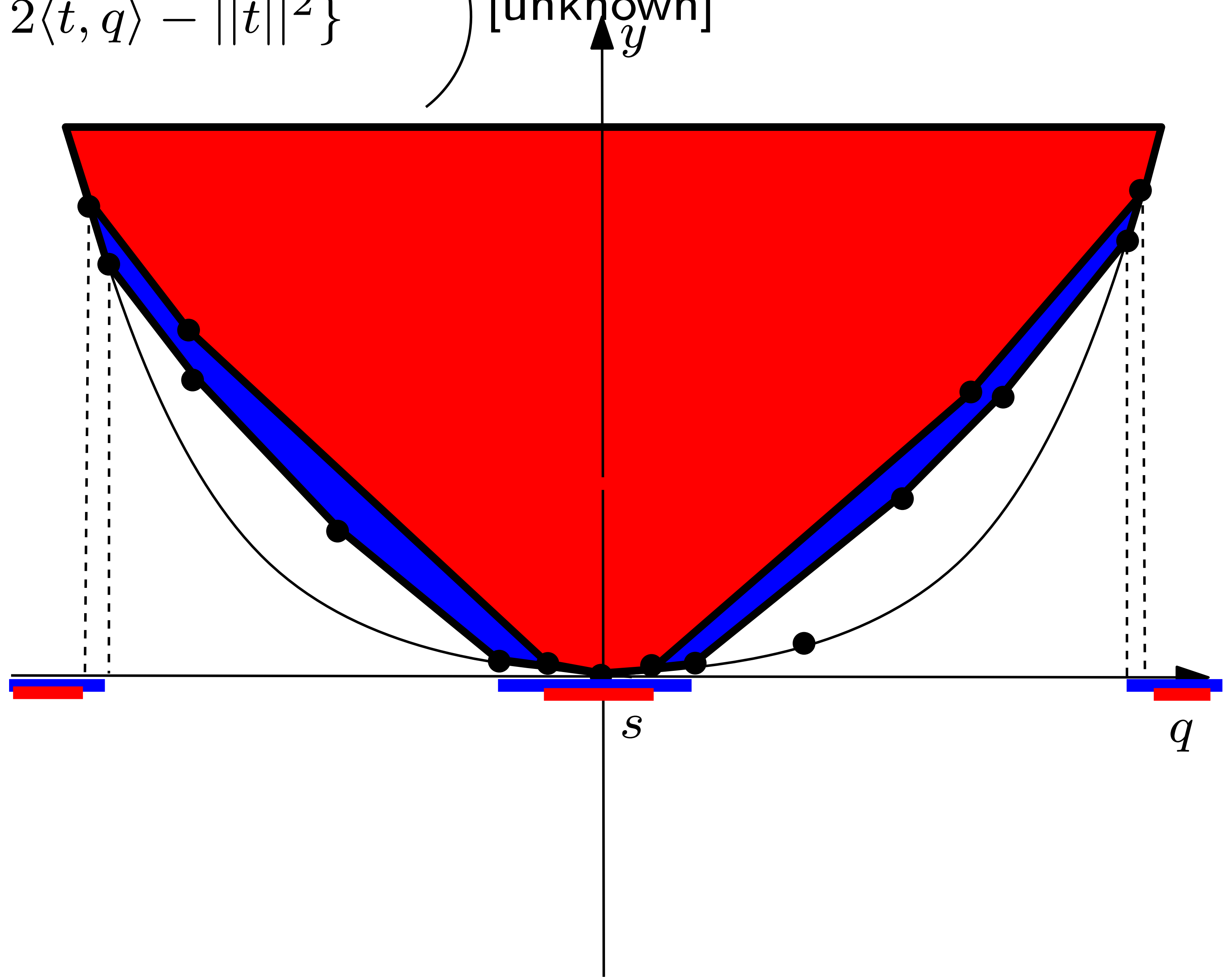
Paraboloid
$\Psi = \{(q, y) : \ y = ||q||^2\}$

Halfspaces
$H(b, \epsilon) = \{(q, y) : \ \alpha y \geq 2\langle b, q\rangle - ||b||^2\}$
for all $b \in S$

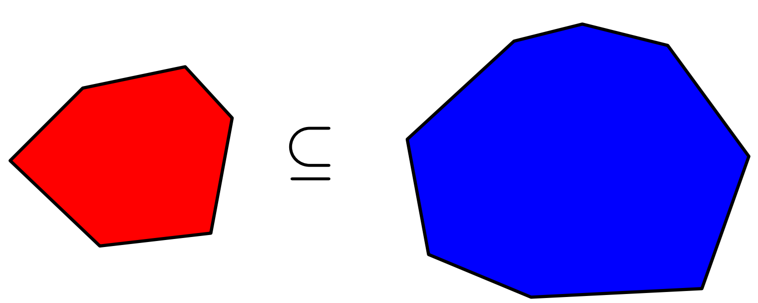
[can compute using S and ϵ]

Halfspaces
$G(t, \epsilon' = \epsilon/2) = \{(q, y) : \ \alpha' y \geq 2\langle t, q\rangle - ||t||^2\}$
for all $t \in L_s$

[unknown]



Goal
It suffices to make sure that



Preprocessing

initialize the weight of all sites to 1

repeat

pick a (weighted) random sample $R \subseteq S$ of size $C_1 cd \log c$

if $\bigcap_{t \in R} G(t, \epsilon/2) \cap \Psi \subseteq \bigcap_{b \in S} H(b, \epsilon)$

return R

else

$v =$ a violating vertex of $\bigcap_{t \in R} G(t, \epsilon/2) \cap \Psi$

double the weight of $V = \{t \in S \setminus R : \; v \notin G(t, \epsilon/2)\}$

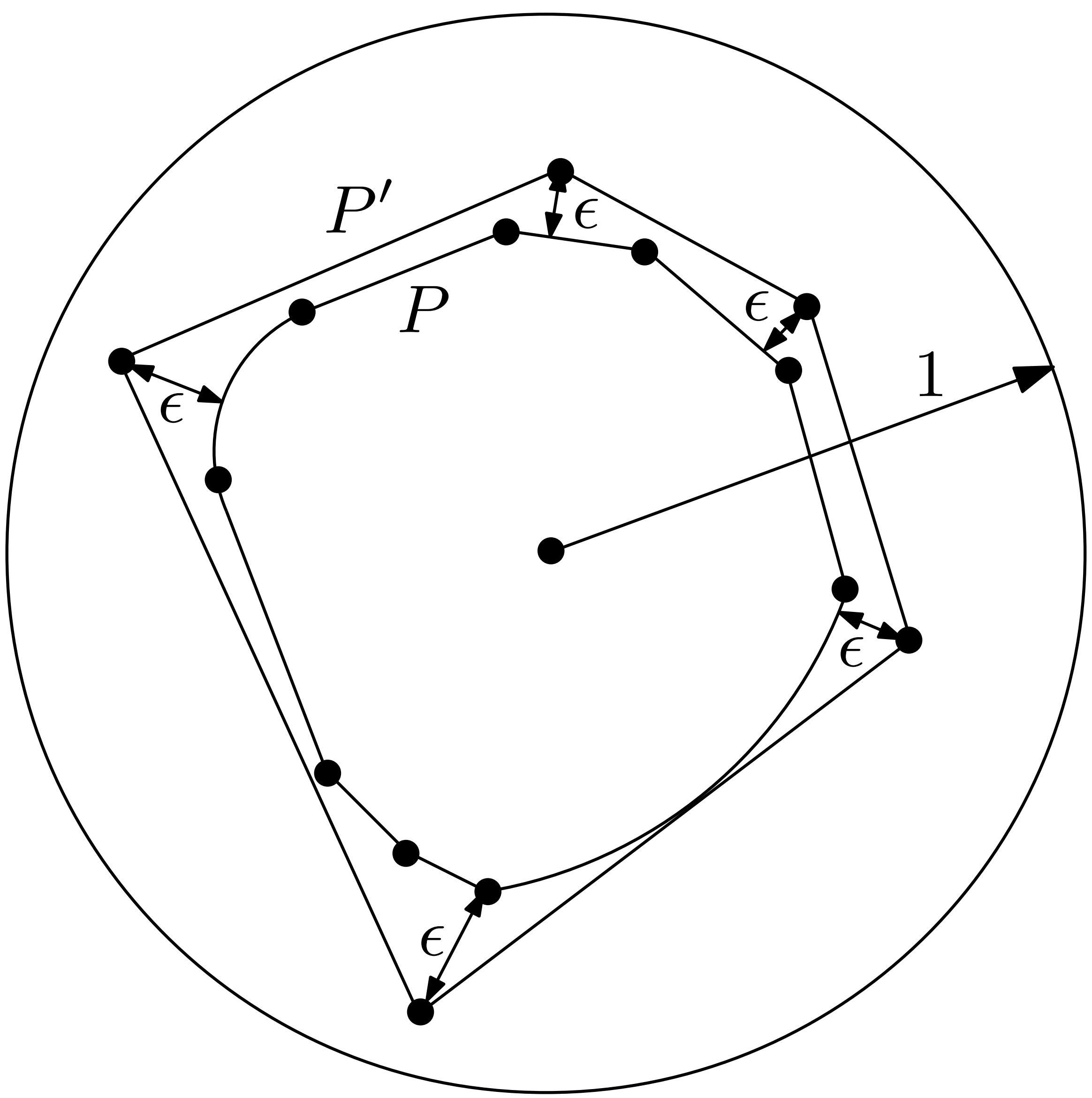
The sample size depends on c, the **optimal** size of L_s

Next we bound c using polytope approximation

Size of L_s

Exhibit a list of size $O\left(\epsilon^{-(d-1)/2}\log\frac{\rho}{\epsilon}\right)$, where $\rho = \frac{\max_{s,t\in S}||s-t||}{\min_{s,t\in S}||s-t||}$

Lemma For any convex and compact set $P \subset \mathbb{R}^d$ contained in the unit sphere and any $\epsilon \in (0,1)$, there is a polytope $P' \supset P$ with at most $O(\epsilon^{(d-1)/2})$ facets which is in the ϵ-neighborhood of P.

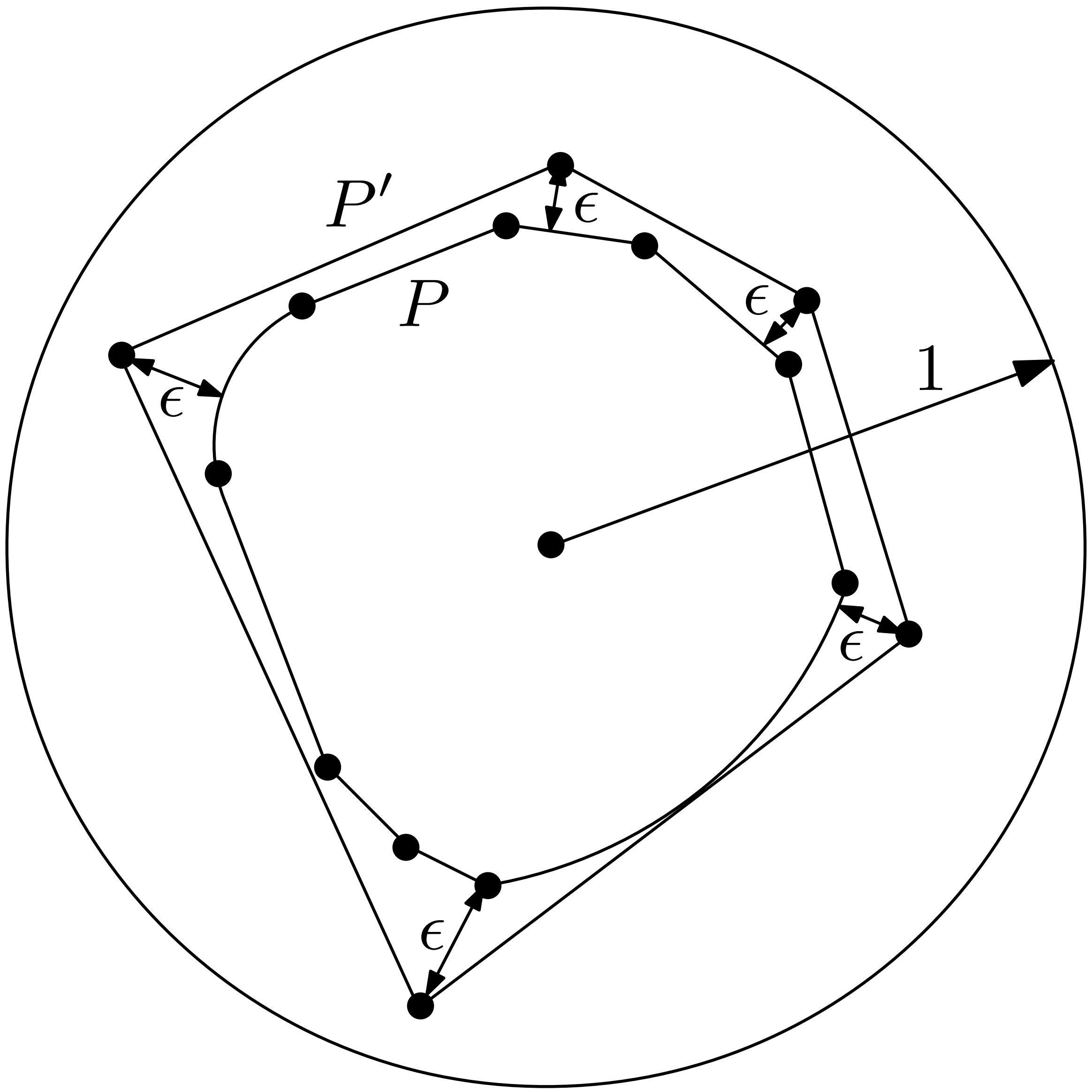


Note Always "outer" approximation

Size of L_s

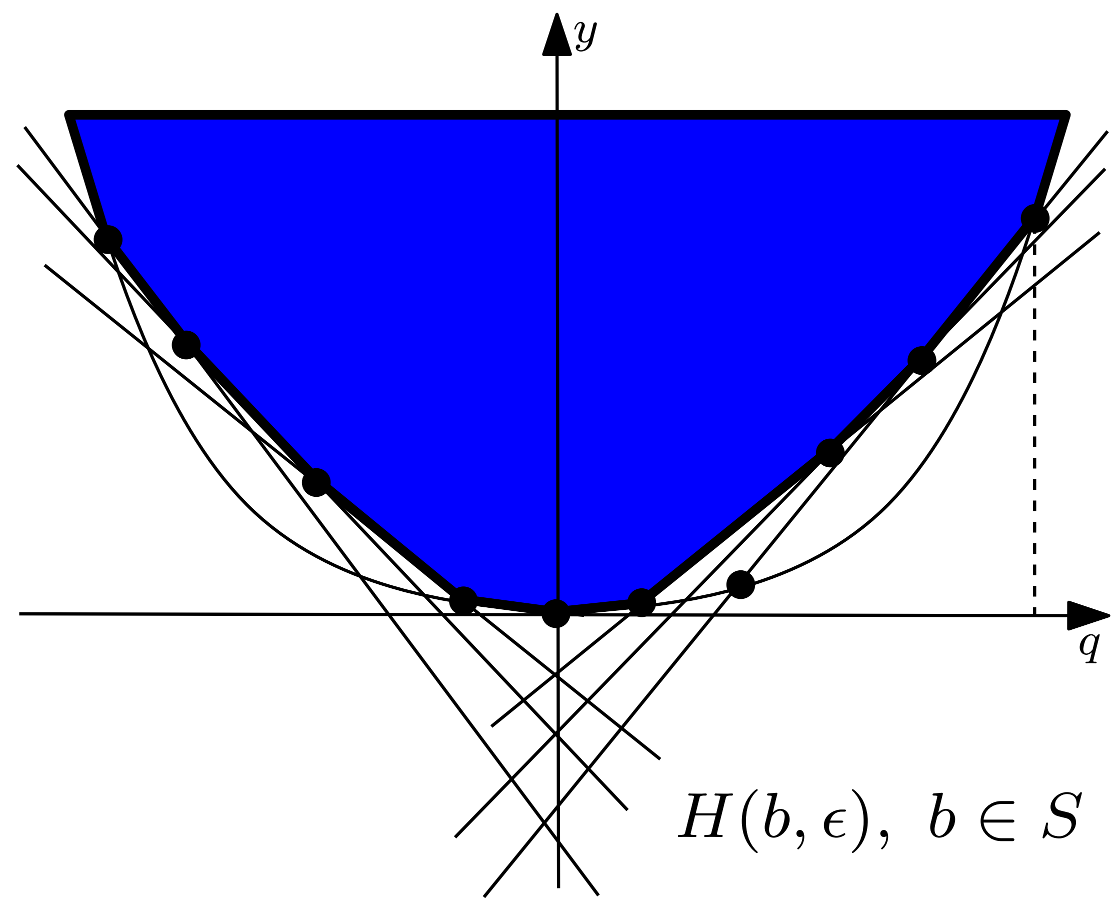
Exhibit a list of size $O\left(\epsilon^{-(d-1)/2}\log\frac{\rho}{\epsilon}\right)$, where $\rho = \frac{\max_{s,t\in S}||s-t||}{\min_{s,t\in S}||s-t||}$

Lemma For any convex and compact set $P \subset \mathbb{R}^d$ contained in the unit sphere and any $\epsilon \in (0,1)$, there is a polytope $P' \supset P$ with at most $O(\epsilon^{(d-1)/2})$ facets which is in the ϵ-neighborhood of P.



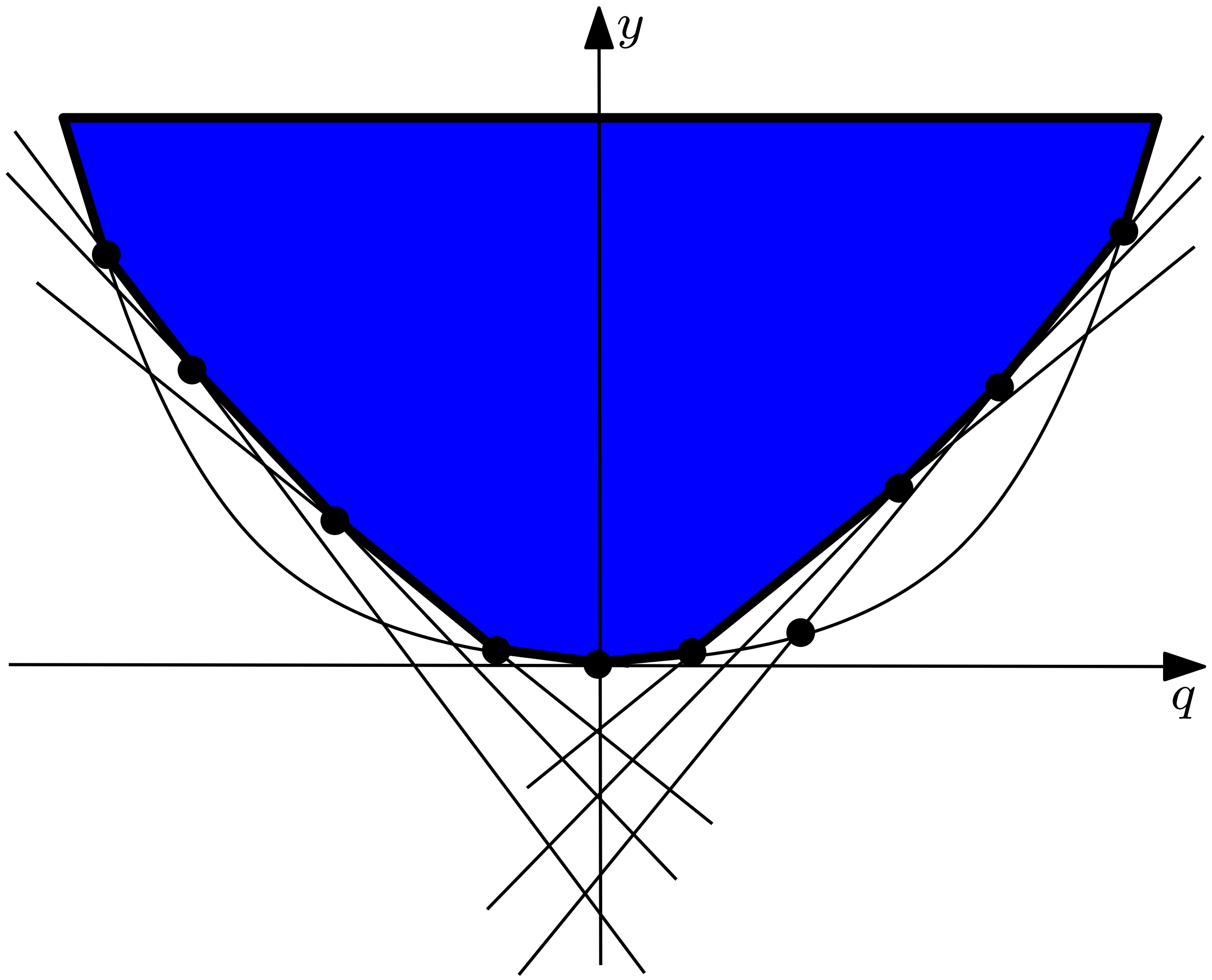
Note Always "outer" approximation

Recall We need an "inner" approximation of this



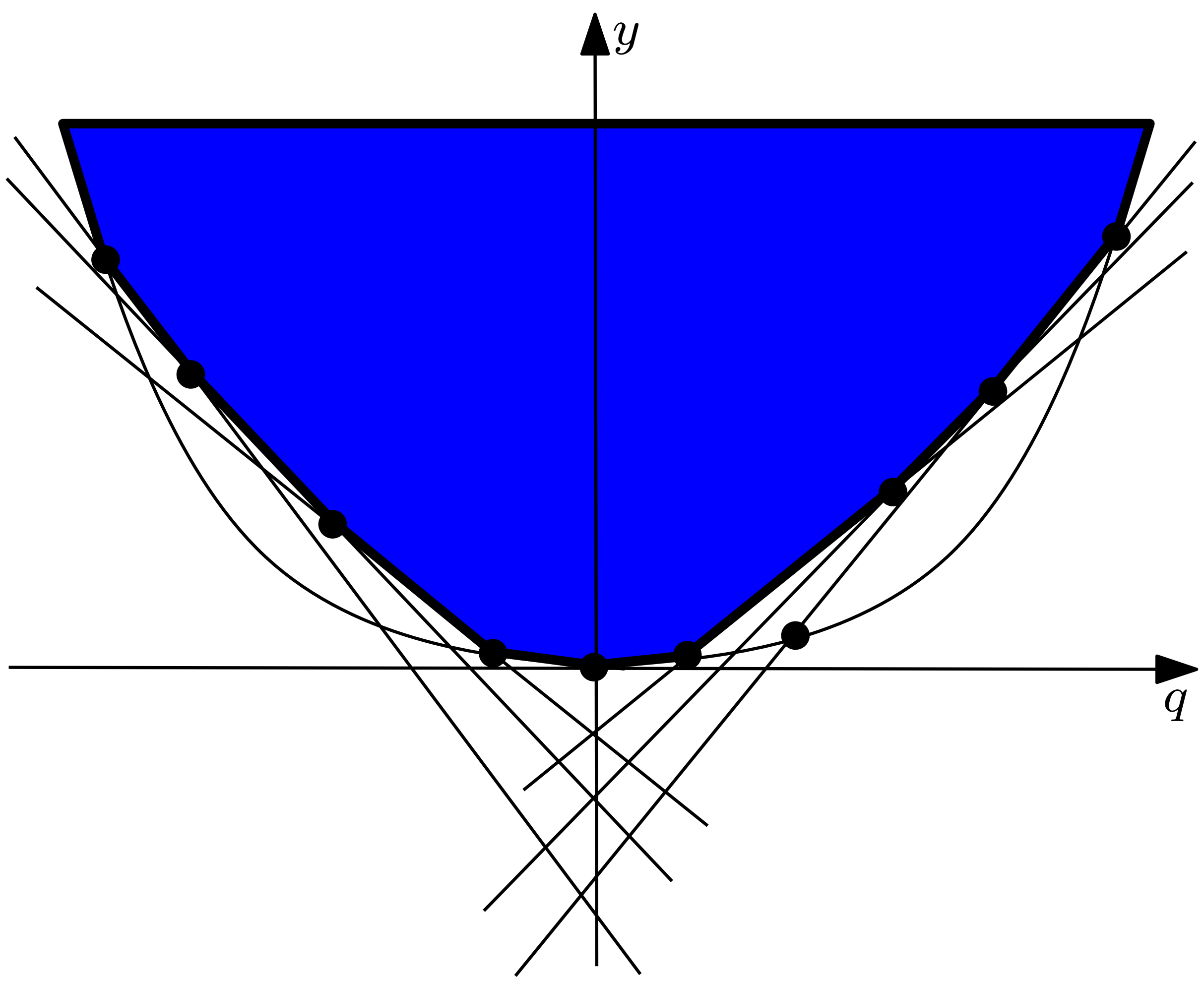
Size of L_s

Want an "inner" approximation of this

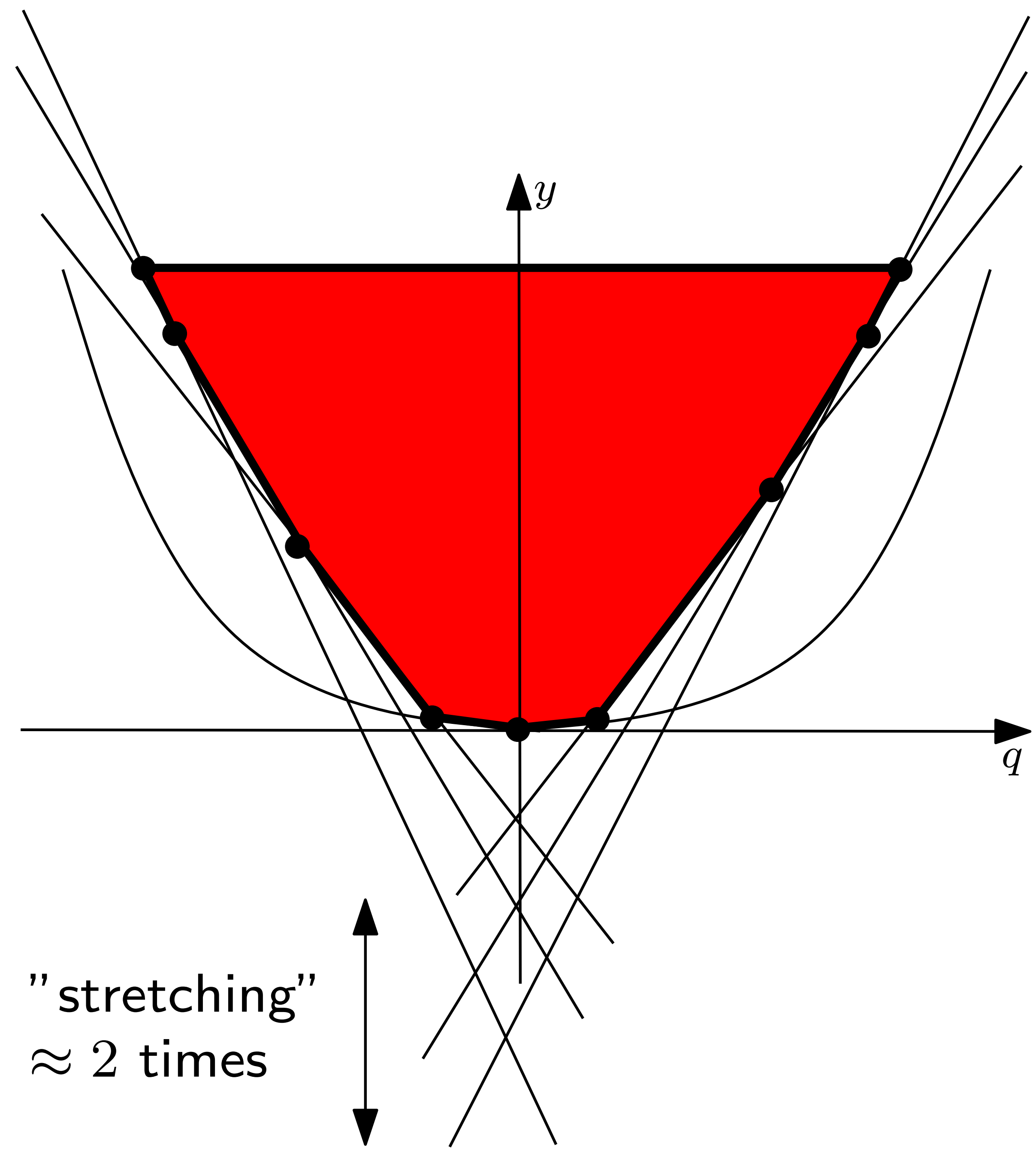


Size of L_s

Want an "inner" approximation of this

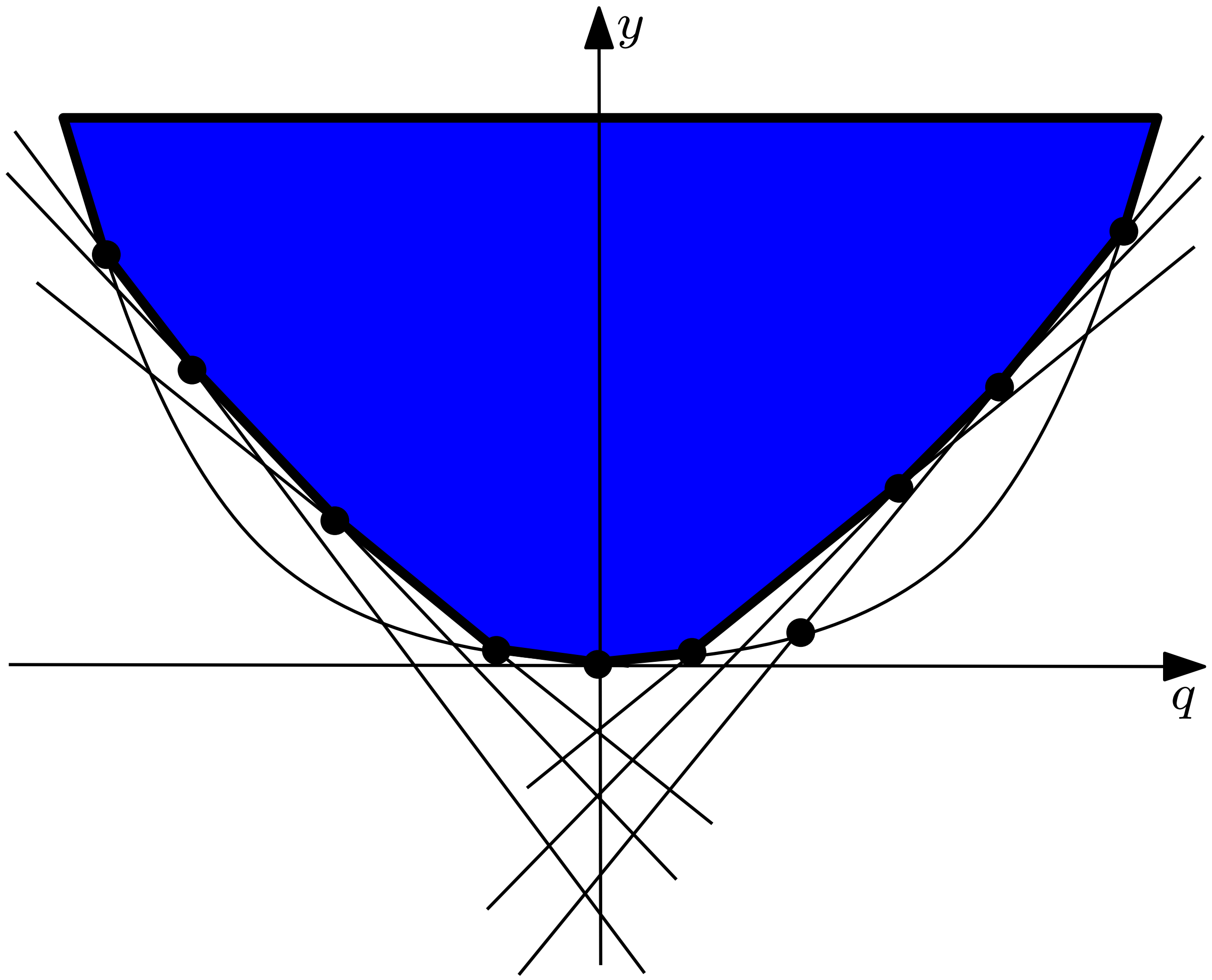


using only these hyperplanes as potential facets

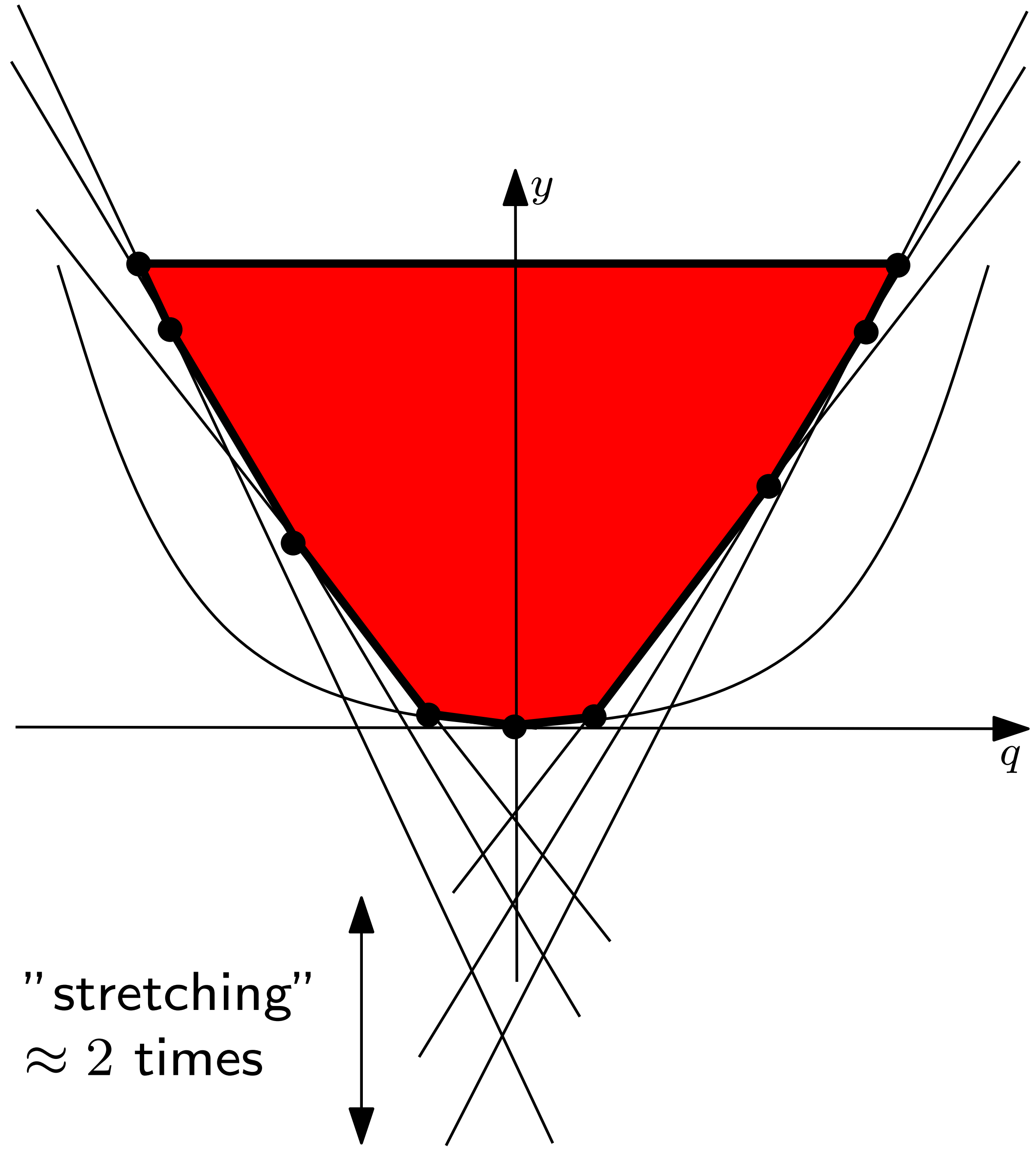


Size of L_s

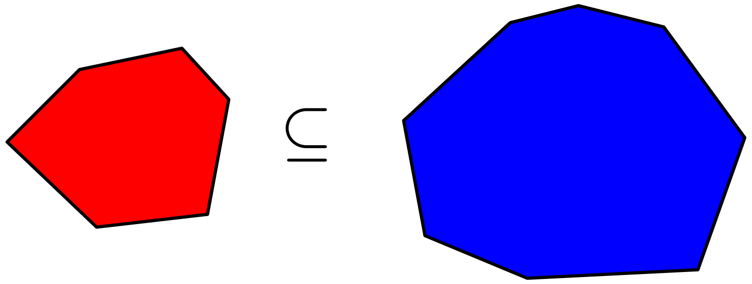
Want an "inner" approximation of this



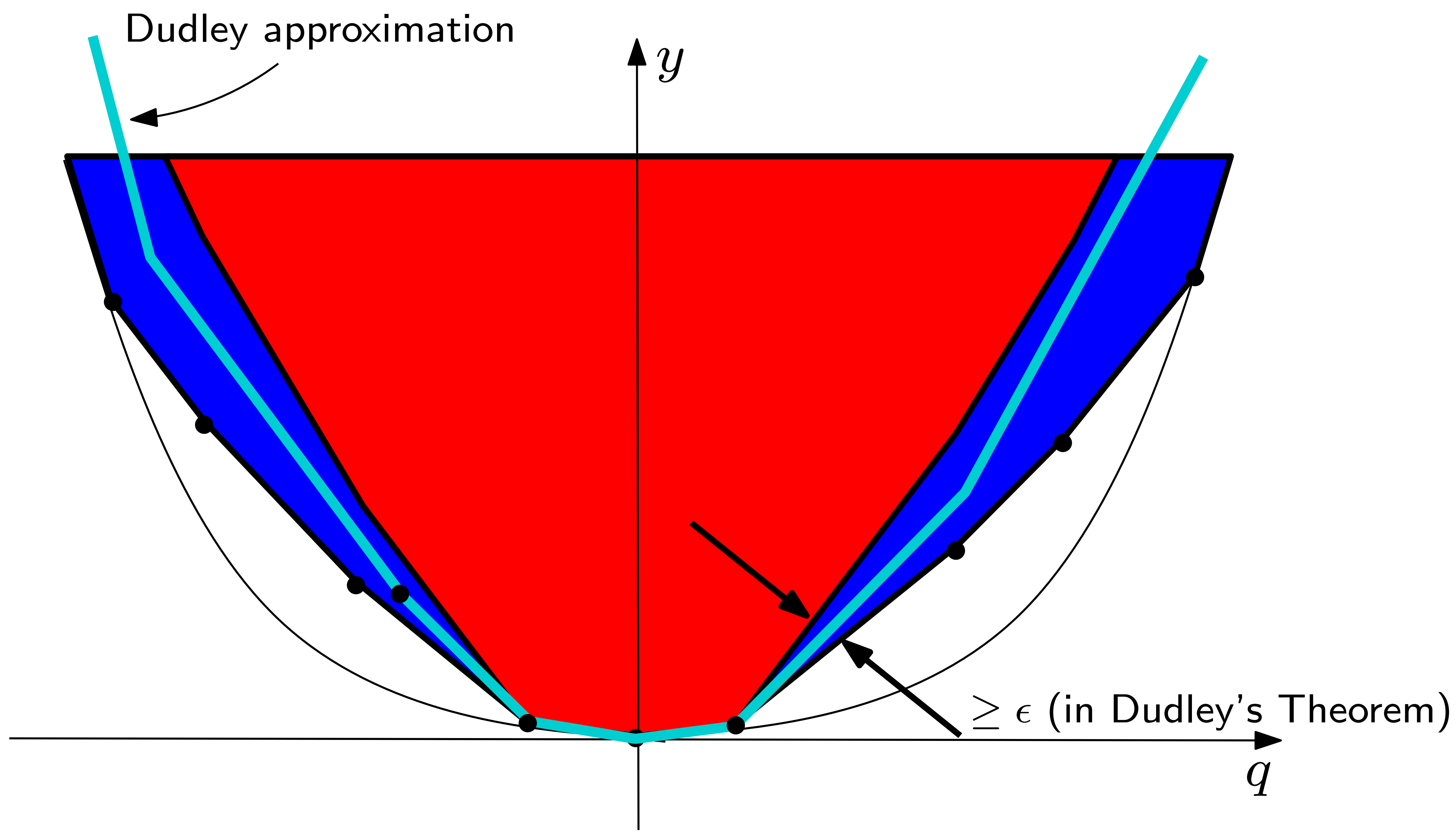
using only these hyperplanes as potential facets



Goal: Subsample (as much as possible) the hyperplanes on the right so that



Size of L_s

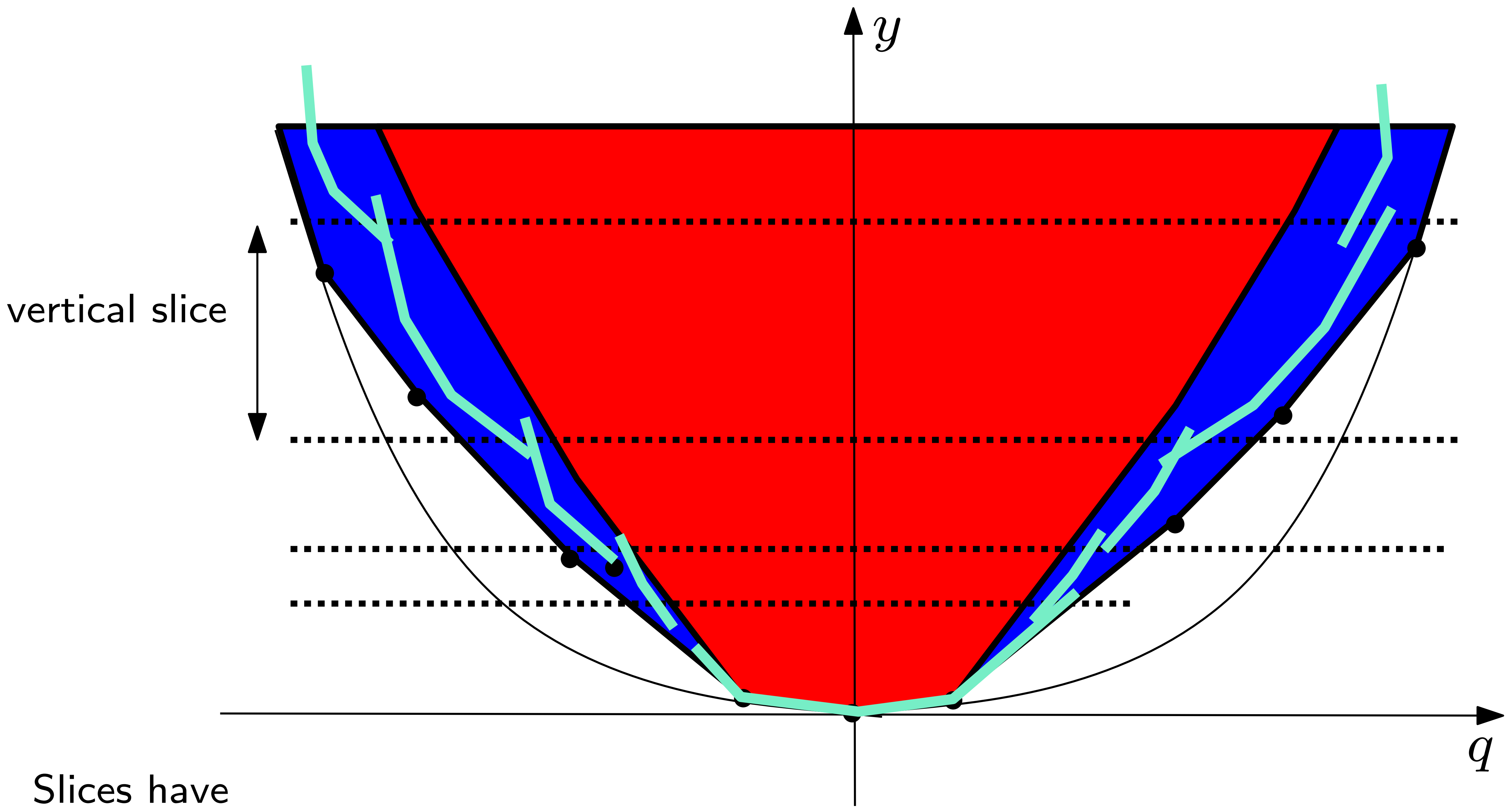


Straightforward application of Dudley's Theorem does not work!

The value of ϵ dictated by the smallest scale

Size of L_s

Solution: height-dependent slicing, per-slice Dudley approximations



Slices have

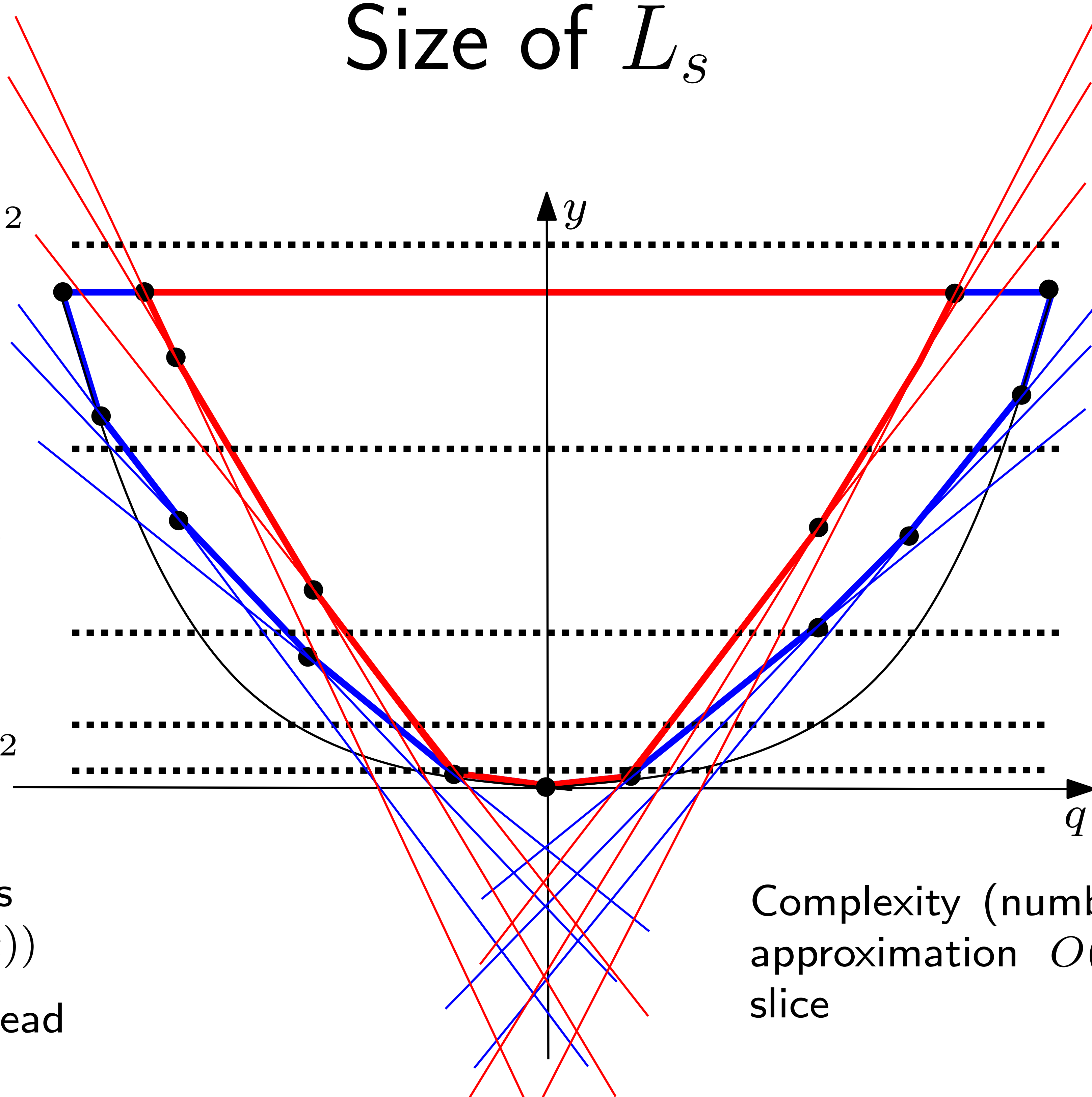
- geometrically increasing height
- "constant" gap

Size of L_s

$d_m > \frac{4}{\alpha^2} \max_{b \in S} ||b||^2$

$d_i = \frac{3}{2} d_{i-1}$

$d_0 = \frac{1}{4} \min_{b \in S} ||b||^2$



Number of slices
$m = O(\log(\rho/\alpha))$

Recall: ρ – spread

Complexity (number of facets) of approximation $O(\epsilon^{-(d-1)/2})$ per slice

Key fact
Red and blue projections into the q-hyperplane **within one slice** are at least a factor of $1 + \epsilon$ apart, so the same ϵ can be used in all approximations

Clarkson's Algorithm: Summary

- Improved query time at the expense of specifying ϵ in advance
- $O(\epsilon^{-(d-1)/2})$ instead of $O(\epsilon^{-d})$
- Express the condition on L_s in the form of $P(S, \epsilon) \supseteq Q(L_s, \epsilon/2)$
- Preprocessing by iterative random sampling from S and checking the containment condition
- Query procedure using
  - top-down search on a skip list
  - iterative improvement algorithm within one level