Approximate Sparse Linear Regression

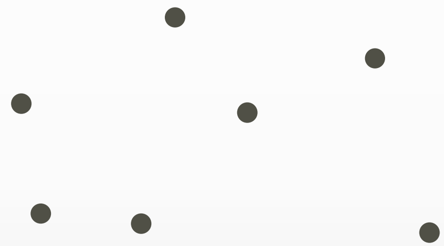
Sariel Har-Peled
UIUC

Piotr Indyk
MIT

Sepideh Mahabadi
Columbia U.

Nearest Neighbor

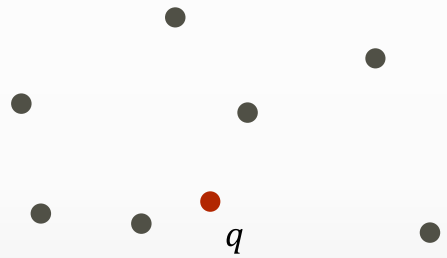
Dataset of n points P in a metric space, e.g. $\mathbb{R}^d$



Nearest Neighbor

Dataset of n points P in a metric space, e.g. $\mathbb{R}^d$

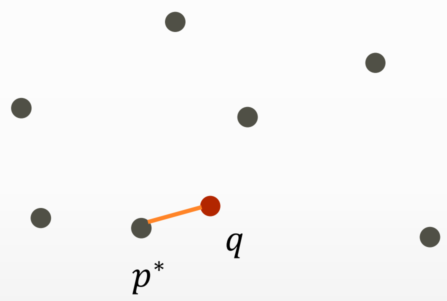
A query point q comes online



Nearest Neighbor

Dataset of n points P in a metric space, e.g. $\mathbb{R}^d$

A query point q comes online



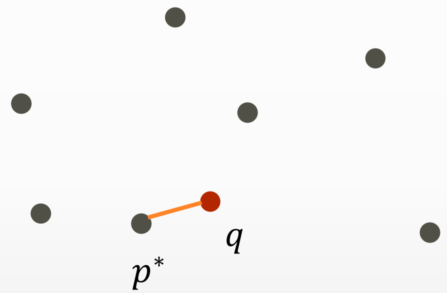
Goal:

- Find the nearest data point p^*

Nearest Neighbor

Dataset of n points P in a metric space, e.g. $\mathbb{R}^d$

A query point q comes online



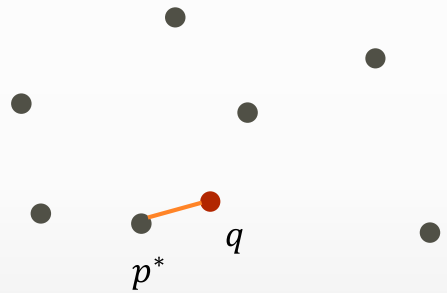
Goal:

- Find the nearest data point p^*
- Do it in sub-linear time and small space

Nearest Neighbor

Dataset of n points P in a metric space, e.g. $\mathbb{R}^d$

A query point q comes online



Goal:

- Find the nearest data point p^*
- Do it in sub-linear time and small space

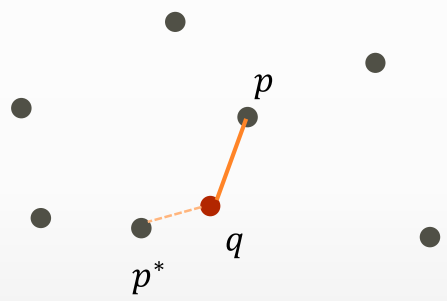
All existing algorithms for this problem

- Either space or query time depending exponentially on d
- Or assume certain properties about the data, e.g., bounded intrinsic dimension

Approximate Nearest Neighbor

Dataset of n points P in a metric space, e.g. $\mathbb{R}^d$

A query point q comes online



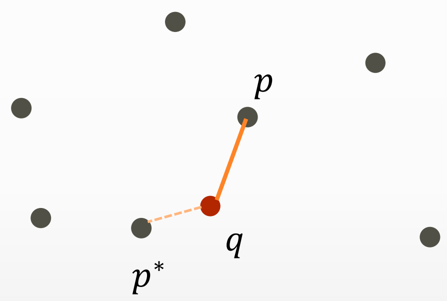
Goal:

- Find the nearest data point p^*
- Do it in sub-linear time and small space
- Approximate Nearest Neighbor
  - If optimal distance is r, report a point in distance cr for $\mathrm{c} = (1 + \epsilon)$

Approximate Nearest Neighbor

Dataset of n points P in a metric space, e.g. $\mathbb{R}^d$

A query point q comes online

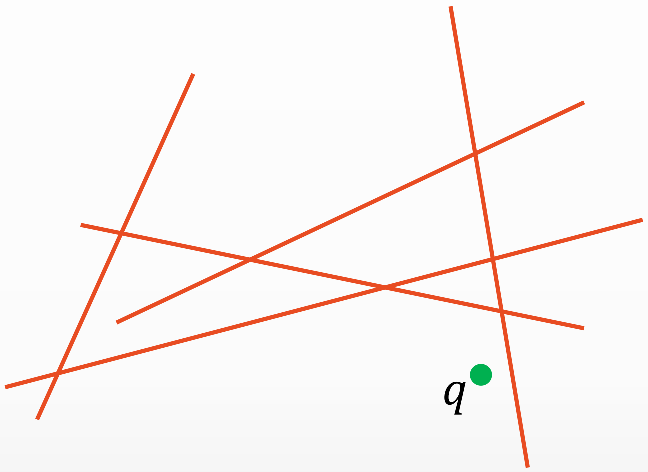


Goal:

- Find the nearest data point p^*
- Do it in sub-linear time and small space
- Approximate Nearest Neighbor
  - If optimal distance is r, report a point in distance cr for $c = (1 + \epsilon)$
  - For Hamming (and Manhattan) query time is $n^{1/O(c)}$ [IM98]
  - and for Euclidean it is $n^{\frac{1}{O(c^2)}}$ [AI08]

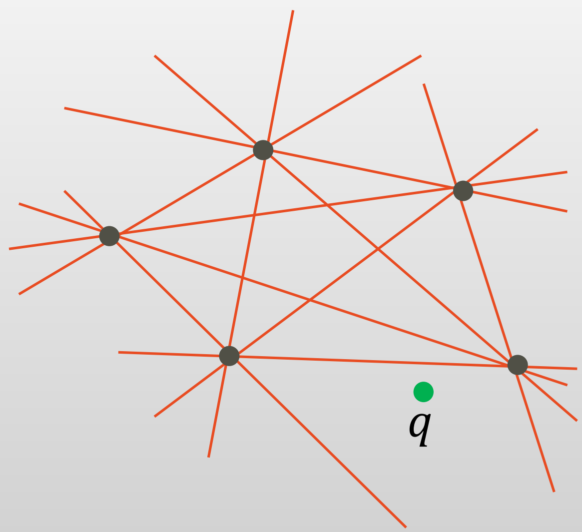
Nearest Hyperplane Search

- **Input:** a set of N hyperplanes of dimension k in $\mathbb{R}^d$
- **Query:** a point in $\mathbb{R}^d$

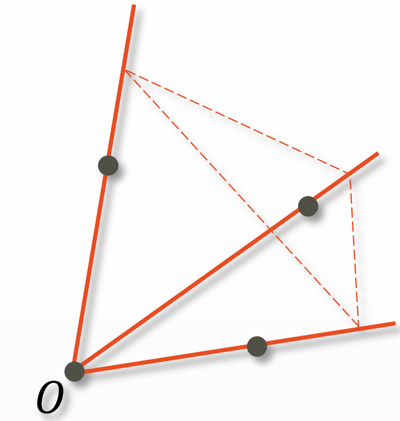


Induced Representation:

- **Input:** a set P of n points in $\mathbb{R}^d$
- **Search Space:** all $N = \binom{n}{k} \approx n^k$ hyperplanes defined by k points in P
- **Query:** a point in $\mathbb{R}^d$

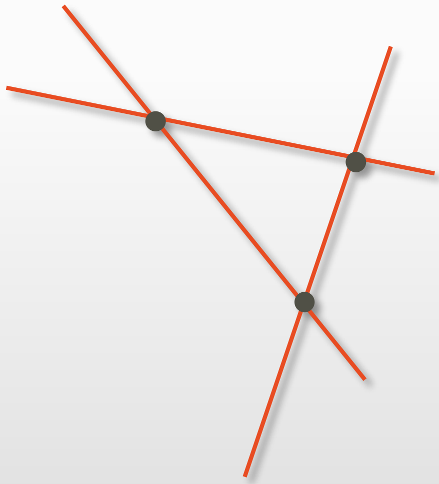


Our Problems



1. **Nearest Induced Subspace**

   Closest k- subspace passing through the origin and k points of P



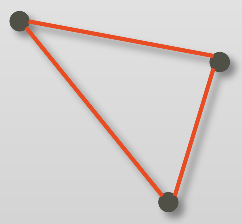
2. **Nearest Induced Flat**

   Closest $(k-1)$-flat (affine subspace) passing through k points of P

3. **Nearest Induced Simplex**

   Closest $(k-1)$-simplex passing through the origin and k points of P

$\boldsymbol{n = 3,}$
$\boldsymbol{k = 2}$



Connection to Sparse Linear Regression

Sparse Linear Regression:

- **Given** a matrix $M \in \mathbb{R}^{d \times n}$ and a d dimensional query q
- **Find** a $\boldsymbol{k}$**-sparse** n dimensional vector τ which **minimizes** $\|M\tau - q\|_2$

➢ Equivalent to the **Nearest Induced Subspace**

$$\begin{array}{cc} & n \\ d & \begin{bmatrix} \quad & M & \quad \end{bmatrix} \end{array} \begin{bmatrix} \tau \end{bmatrix} = \begin{bmatrix} q \end{bmatrix}$$

❖ If we require $\|\tau\|_1 = 1$,

❖ If we further require $\tau_i \geq 0$,

Nearest Induced Flat

Nearest Induced Simplex

✓ Applications in Machine learning, compressed sen

Connection to Sparse Linear Regression

Sparse Linear Regression:

- **Given** a matrix $M \in \mathbb{R}^{d \times n}$ and a d dimensional query q
- **Find** a $\boldsymbol{k}$**-sparse** n dimensional vector τ which **minimizes** $\|M\tau - q\|_2$

$$\begin{array}{cc} & n \\ d & \left[\quad M \quad\right] \end{array} \left[\tau\right] = \left[q\right]$$

➢ Equivalent to the **Nearest Induced Subspace**

❖ If we require $\|\tau\|_1 = 1$, $\Rightarrow$ **Nearest Induced Flat**

❖ If we further require $\tau_i \geq 0$, $\Rightarrow$ **Nearest Induced Simplex**

✓ Applications in Machine learning, compressed sensing, computer vision, etc.

Results – Algorithms

1. **Approximate Nearest Induced Problems (Online)**

Problem	Equivalent problem	Space	Query
ANI **Subspace**	SLR	$n^{k-1} \cdot S_{ANN}$	$n^{k-1} \cdot T_{ANN}$
ANI **Flat**	Affine SLR	$n^{k-1} \cdot S_{ANN}$	$n^{k-1} \cdot T_{ANN}$
ANI **Simplex**	Convex SLR	$n^{k-1} \cdot \log^{k} n \cdot S_{ANN}$	$n^{k-1} \cdot \log^{k} n \cdot T_{ANN}$

2. **Special case: convex variant of $k = 2$, Nearest Induced**

Problem	Approximation	Space	Query
Online	$(1+\epsilon), \epsilon \leq 1$	$n \cdot \log n \cdot S_{ANN}$	$n^{k-1} \cdot \epsilon^{-2} \log n \cdot T_{ANN}$
Offline	$2(1+\epsilon)$	$n^{k-1} \cdot S_{ANN}$	$n^{k-1} \cdot T_{ANN}$

Results – Conditional Lower Bounds

1. **Assuming "Affinely Degenerate Conjecture"**

   Our data structure in the **online settings** provides an optimal trade-off up to polylog factors:

   - No algorithm can improve both preprocessing from $\tilde{O}(n^k)$ and query time from $\tilde{O}(n^{k-1})$ by much.

2. **Assuming hardness for the k-sum problem**

   Solving all three variants of the problem in the **offline setting** requires $\tilde{\Omega}(\frac{n^{k/2}}{e^k})$ times.

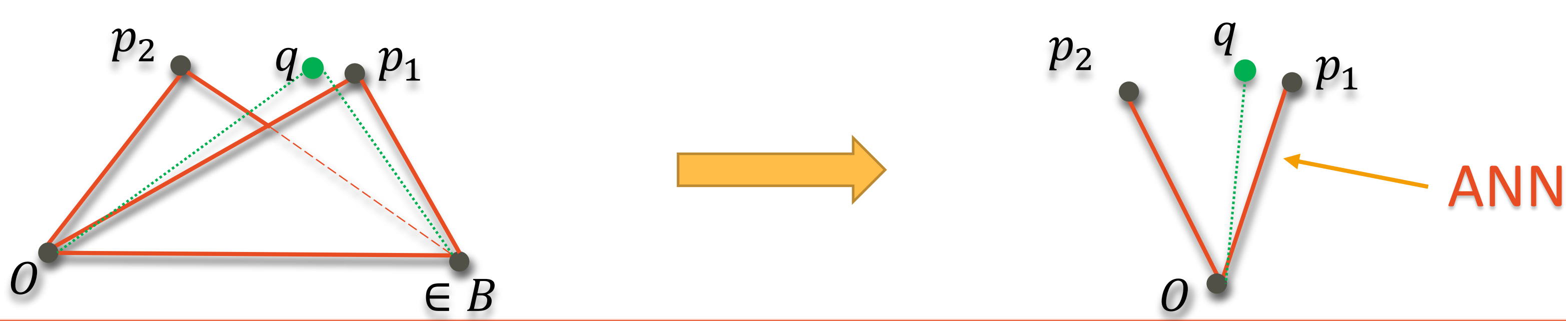
Algorithms

Nearest Induced Subspace

1. Fix a subset B of $k-1$ points → $\approx n^{k-1}$ such subsets
2. Search among all k-subspaces $\mathcal{F}$ that include the points in B. → $n-k+1$ such subspaces
   - ➢ **Use a single ANN query** → Total time is $\tilde{O}(n^{k-1} \cdot T_{ANN})$

- Use the projection Π which maps the points in B into the origin.
  - $\mathcal{F}$ becomes a set of vectors $\boldsymbol{V} = \boldsymbol{\mathcal{F}_\Pi} \in \mathbb{R}^{d-k+1}$
  - Project the query as well to get $\boldsymbol{q_\Pi} \in \mathbb{R}^{d-k+1}$
- Normalize all vectors to be on the unit sphere

➢ The closest subspace in $\mathcal{F}$ to q corresponds to the closest vector in V to q_Π



Nearest Induced Flat

1. Fix a subset B of $k-1$ points
2. Search among all k-subspaces $\mathcal{F}$ that include the points in B.
   - **Use a single ANN query**

$\approx n^{k-1}$ such subsets

$n-k+1$ such subspaces

Total time is $\tilde{O}(n^{k-1} \cdot T_{Ann})$

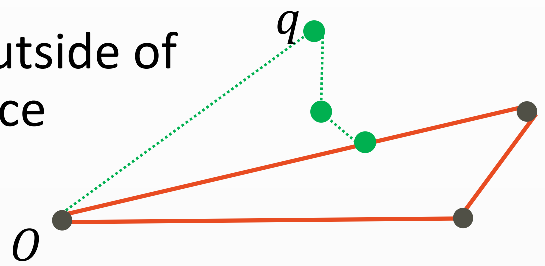
- Use the projection Π which maps the points in B into the origin.
  - $\mathcal{F}$ becomes a set of vectors $\boldsymbol{V} = \boldsymbol{\mathcal{F}_{\Pi}} \in \mathbb{R}^{d-k+1}$
  - Project the query as well to get $\boldsymbol{q_{\Pi}} \in \mathbb{R}^{d-k+1}$
- Normalize all vectors to be on the unit sphere

➢ The closest subspace in $\mathcal{F}$ to q corresponds to the closest vector in V to q_{Π}

✓ Similar approach works for Nearest Induced **Flat**

Nearest Induced Simplex

- Why not the same approach?
  - ✗ Projection of q onto the nearest flat might fall outside of the simplex corresponding to the nearest subspace



Intermediate goal: retrieve all **"feasible simplices"**

- Find all $p \subset P \setminus B$ s.t. projection of q on to the flat formed by $p \cup B$, i.e., $\mathcal{F}_{p\cup B}$, falls inside the simplex $\Delta_{p\cup B}$.

✓ If we accomplish this, we can use the algorithm for finding the closest flat among this set.

✓ What if the closest point on the closest simplex lies on the boundary?

  - The boundary is a lower dimensional object which can be checked by brute force in time $\widetilde{\boldsymbol{O}}(\boldsymbol{n}^{\boldsymbol{k}-\boldsymbol{1}})$

Characterizing Feasibility

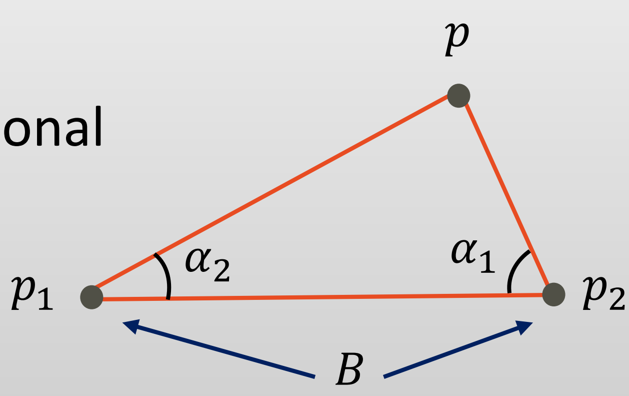
"One can detect whether the simplex $\Delta_{p \cup B}$ is feasible or not without having full knowledge of p and q."

More specifically it is enough to have

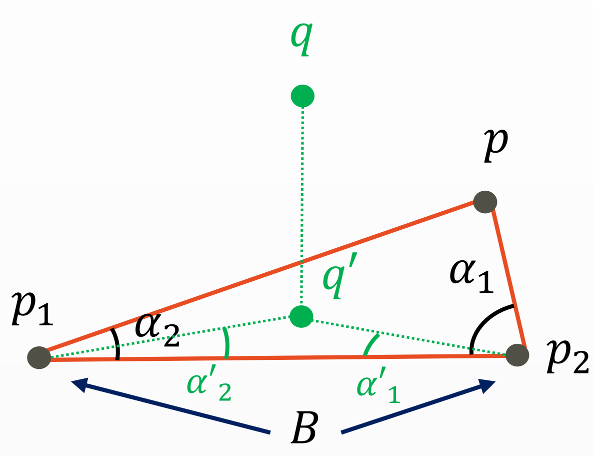
- The distance (denoted by $\boldsymbol{r}$) from q to the flat $\mathcal{F}_{p \cup B}$
- The relative positioning of p with respect to B
- The relative positioning of q with respect to B

Relative Positioning of a point p with respect to B is $(\alpha_1, \cdots, \alpha_{k-1})$ where

- α_i is the angle between $(k-2)$-dimensional flats $\mathcal{F}_B$ and $\mathcal{F}_{B \cup \{p\} \setminus \{p_i\}}$
- p_i is the i-th point in B



Characterizing Feasibility



Let

- $(\alpha_1, \cdots, \alpha_{k-1})$: relative positioning of p w.r.t. B
- $(\alpha'_1, \cdots, \alpha'_{k-1})$: relative positioning of q' w.r.t. B
  - Where q' is the projection of q on $\mathcal{F}_{B\cup\{p\}}$

Observation: $\Delta_{B\cup\{p\}}$ is feasible iff $\alpha'_i \le \alpha_i$ for every $i < k$

Let $(\alpha^q_1, \cdots, \alpha^q_{k-1})$ be the relative positioning of q with respect to B

Lemma: one can compute $(\alpha'_1, \cdots, \alpha'_{k-1})$ given

1. $(\alpha^q_1, \cdots, \alpha^q_{k-1})$
2. and $\boldsymbol{r}$: the distance from q to $\mathcal{F}_{B\cup\{p\}}$

We can detect Feasibility using Range search trees☺

We don't know $\boldsymbol{r}$ in advance ☹

Feasibility is Monotone

Given

- $(\alpha_1, \cdots, \alpha_{k-1})$: relative positioning of p w.r.t. B
- $(\alpha_1^q, \cdots, \alpha_{k-1}^q)$: relative positioning of q w.r.t. B

Monotonicity Property: if for a parameter r the corresponding simplex is feasible, then for all $r' \geq r$, the corresponding simplex is feasible too.

We can use **binary search** to find the correct value of r ☺

Algorithm Outline

Data Structure:

- Construct a **range search tree** on the $(k-1)$-tuples $(\alpha_1^p, \cdots, \alpha_{k-1}^p)$ for all points $p \in P \setminus B$
- For each node T in the tree, construct a data structure on the vectors corresponding to T, for retrieving the nearest **flat** $\mathcal{F}_{B\cup\{p\}}$ for $p \in T$

Query Processing:

- Given q, compute $\left(\alpha_1^q, \cdots, \alpha_{k-1}^q\right)$
- Use Binary Search on r
  - Compute $(\alpha_1', \cdots, \alpha_{k-1}')$ using $\left(\alpha_1^q, \cdots, \alpha_{k-1}^q\right)$ and r
  - Use the range search tree to retrieve all points p whose simplices would have been feasible if at distance r
    - This step will return polylog nodes T from the tree whose union is our desired set of points.
  - Use their corresponding data structures to retrieve the nearest flat $\mathcal{F}_{ANN}$ among those.
  - If the distance between q and $\mathcal{F}_{ANN}$ was less than r, then continue with a smaller value of r, otherwise continue with a larger value.

Conditional Lower Bounds

Affinely Degenerate Conjecture

Affinely Degenerate Conjecture: given a set P of n points in $\mathbb{R}^d$, checking whether they are in general position requires $\Omega(n^d)$ time [Erickson, Seidel'95].

- **General position:** all subsets of $d + 1$ points are affinely independent.

The Reduction

- Let $\boldsymbol{k} = \boldsymbol{d}$ and construct our data structure for **Nearest Induced Flat** with **space $\boldsymbol{O(n^k) = O(n^d)}$** and a **query time of $\boldsymbol{\widetilde{O}(n^{d-1})}$**
  - Use ANN of [AMNSW'98]: $S_{ANN} = O(n)$ and $T_{ANN} = O(\log n)$

- Now for each input point p, we query the closest $(d-1)$-dimensional induced flat passing through d points of $\boldsymbol{P \setminus \{p\}}$ to P
  - Our data structure can be modified to handle this type of query with an extra log factor.

- If any of the queries reported a 0, the point set is not in general position. Total time is $\boldsymbol{O(n^d)}$.

The k-sum Problem

k-sum problem: given n integer numbers $a_1, \dots, a_n$, does there exist a subset of size k whose sum is 0?

- Conjectured to require $\widetilde{\Omega}(n^{\lceil k/2 \rceil})$ time

  [Patrascu, Williams'10].

The Reduction

$$v_i = \begin{bmatrix} a_i \\ 0 \\ \dots \\ 0 \\ 1 \\ 0 \\ \dots \\ 1 \end{bmatrix}$$

- For each integer a_i assign a $(k+1)$ dimensional vector v_i
  - First coordinate of v_i is a_i
  - All other coordinates are 0, except for one randomly chosen coordinate which is 1
- The **query** is $\left[\mathbf{0}, \frac{\mathbf{1}}{\boldsymbol{k}}, \dots, \frac{\mathbf{1}}{\boldsymbol{k}}\right]^{\mathbf{T}}$ and let $v_{i_1}, \dots, v_{i_k}$ be the points corresponding to the nearest flat.
  - If the distance of query to the flat is 0 return $a_{i_1}, \dots, a_{i_k}$
    - This means $q = c_1 v_{i_1} + \dots + c_k v_{i_k}$ for some coefficients
  - Otherwise return not possible
- With probability e^{-k} the positions of 1's in $v_{i_1}, \dots, v_{i_k}$ form a permutation. Therefore their coefficients should be equal and thus $\sum_{j \le k} a_{i_j} = q_1 = 0$
- Similar argument for the reverse direction.

Open Problems

- Finding other optimal trade-offs: can we achieve much lower query time at a cost of increasing the preprocessing time /space?
- Shaving polylog factors from the current results.
- Proving unconditional lower bound for the problem.

Open Problems

- Finding other optimal trade-offs: can we achieve much lower query time at a cost of increasing the preprocessing time /space?

- Shaving polylog factors from the current results.

- Proving unconditional lower bound for the problem.

Thanks

Questions?