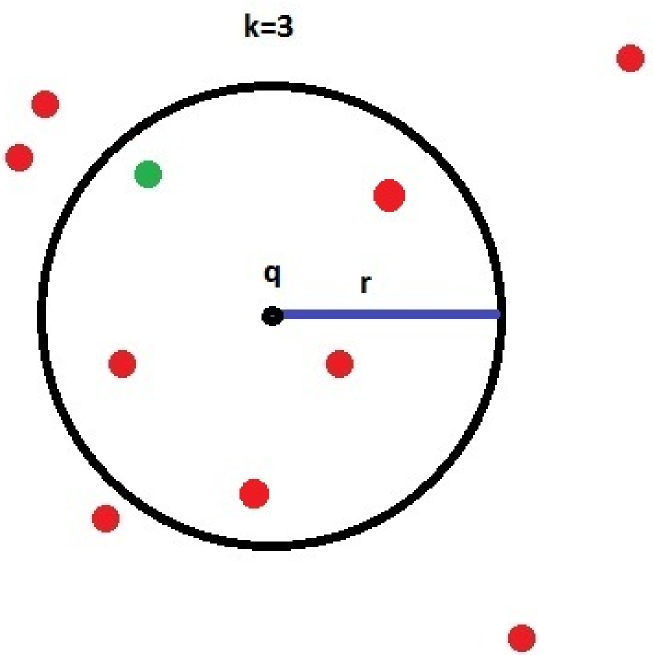
Diverse Near Neighbor Problem

Sofiane Abbar (QCRI)
Sihem Amer-Yahia (CNRS)
Piotr Indyk (MIT)
Sepideh Mahabadi (MIT)
Kasturi R. Varadarajan (UIowa)

Near Neighbor Problem

- **Definition**
  - Set of n points $\boldsymbol{P}$ in d-dimensional space
  - Query point $\boldsymbol{q}$
  - Report one neighbor of $\boldsymbol{q}$ if there is any



- **Neighbor:** A point within distance r of query

- **Application**
  - Major importance in databases (document, image, video), information retrieval, pattern recognition
    - Object of interest as point
    - Similarity is measured as distance.

Motivation

Search: How many answers?

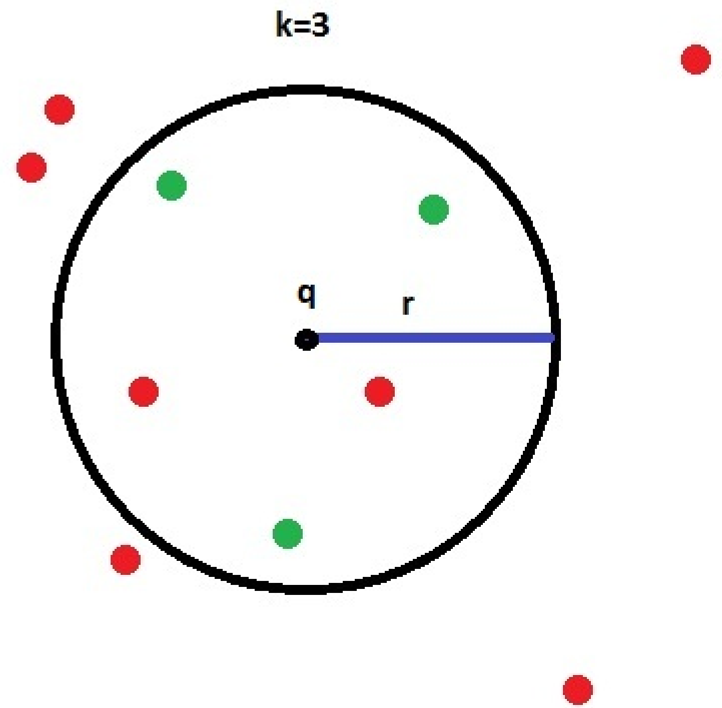
- Small output size, e.g. 10
  - Reporting k Nearest Neighbors may not be informative (could be identical texts)
- Large output size
  - Time to retrieve them is high

Small output size which is

- **Relevant** and **Diverse**
- Good to have result from each cluster, i.e. should be diverse

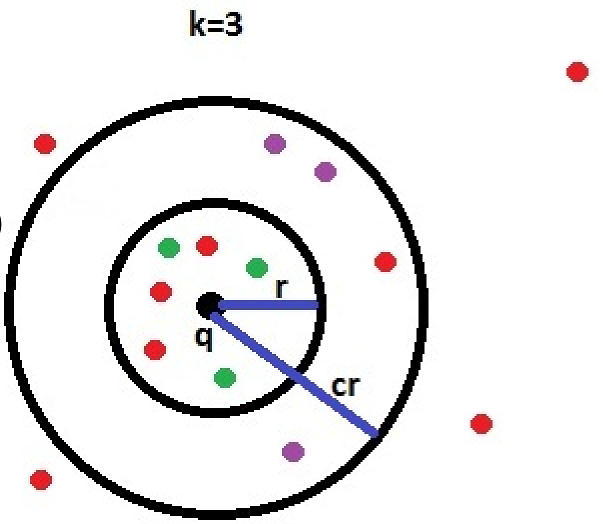
Diverse Near Neighbor Problem

- **Definition**
  - Set of n points $\boldsymbol{P}$ in d-dimensional space
  - Query point $\boldsymbol{q}$
  - Report the **k** most diverse neighbors of q
- **Neighbor:**
  - Points within distance r of query
  - We use Hamming distance
- **Diversity:**
  - $\mathrm{div}(S) = \min_{p,q \in S} |p - q|$
- **Goal:** report **Q** (green points), s.t.
  - $Q \subseteq P \cap B(q, r)$
  - |Q| = k
  - $div(Q)$ is maximized



Approximation

- Want sublinear query time, so need to approximate
- Approximate NN:
  - $B(q,r) \rightarrow B(q,cr)$ for some value of $c > 1$
  - **Result:** query time of $O(dn^{\frac{1}{c}})$
- Approximate Diverse NN:
  - **Bi-criterion** approximation: distance and diversity
  - $(\mathbf{c}, \boldsymbol{\alpha})$-Approximate k-diverse Near Neighbor
  - Let Q^* (green points) be the optimum solution for $B(q,r)$
    - Report approximate neighbors Q (purple points) $Q \subseteq B(q,cr)$
    - Diversity approximates the optimum diversity $$div(Q) \geq \frac{1}{\alpha} div\,(Q^*)\,,\ \alpha \geq 1$$



Results

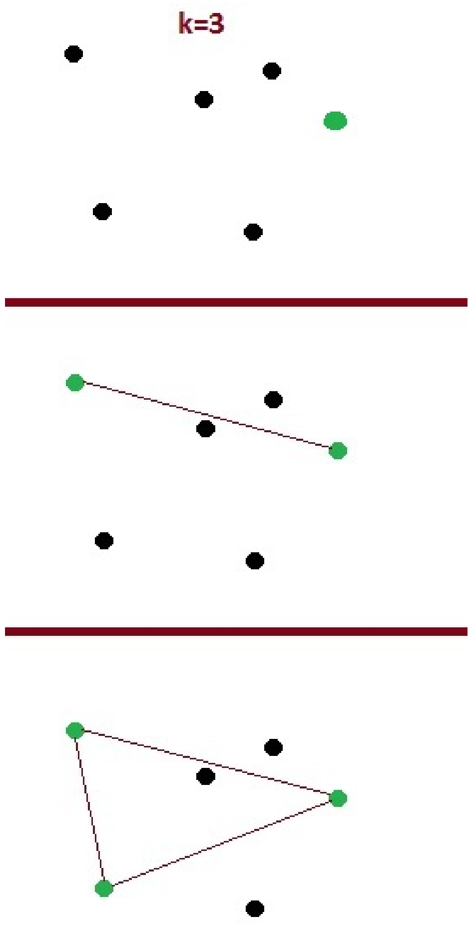
	Algorithm A	Algorithm B
Distance Apx. Factor	c > 2	c >1
Diversity Apx. Factor α	6	6
Space	$(n \log k)^{1+1/(c-1)} + nd$	$\log k * n^{1+1/c} + nd$
Query Time	$\left(k^2 + \frac{\log n}{r}\right) d \ (\log k)^{c/(c-1)} n^{1/(c-1)}$	$\left(k^2 + \frac{\log n}{r}\right) d * \log k * n^{1/c}$

- Algorithm A was earlier introduced in [Abbar, Amer-yahia, Indyk, Mahabadi, WWW'13]

Techniques

Compute k-diversity: GMM

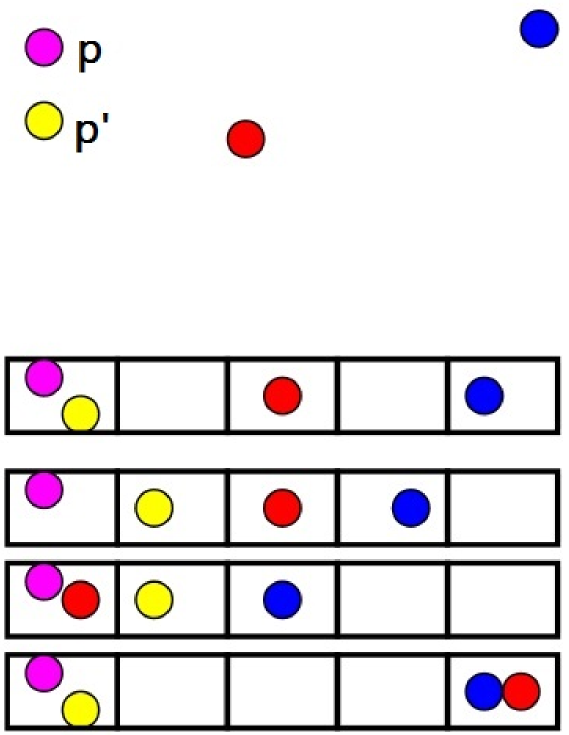
- Have n points, compute the subset with maximum diversity.
- Exact : **NP-hard** to approximate better than 2 [Ravi et al.]
- **GMM** Algorithm [Ravi et al.] [Gonzales]
  - Choose an arbitrary point
  - Repeat k-1 times
    - Add the point whose minimum distance to the currently chosen points is maximized



- Achieves approximation factor **2**
- Running time of the algorithm is O(kn)

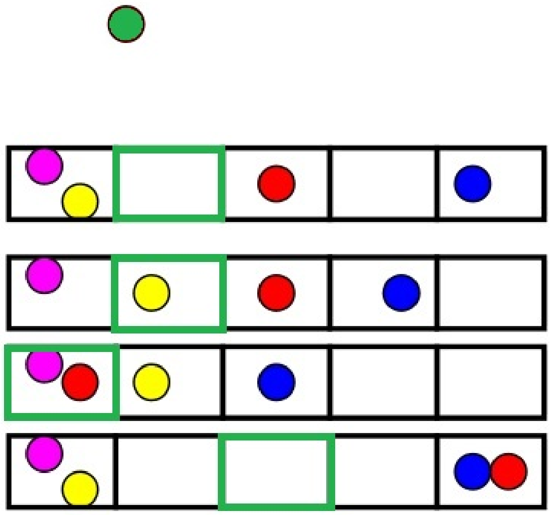
Locality Sensitive Hashing (LSH)

- **LSH**
  - close points have higher probability of collision than far points
  - **Hash functions:** $g_1 , \dots , g_L$
    - $g_i = < h_{i,1}, \dots, h_{i,t} >$
    - $h_{i,j} \in \mathcal{H}$ is chosen randomly
    - $\mathcal{H}$ is a family of hash functions which is (P_1, P_2, r, cr)-sensitive:
      - If $||p - p'|| \leq r$ then $\Pr[h(p) = h(p')] \geq P_1$
      - If $||p - p'|| \geq cr$ then $\Pr[h(p) = h(p')] \leq P_2$
    - Example: Hamming distance:
      - $h(p) = p_i$, i.e., the ith bit of p
      - Is $(1 - \frac{r}{d}, 1 - \frac{rc}{d}, r, rc)$-sensitive
  - $\boldsymbol{L}$ and $\boldsymbol{t}$ are parameters of LSH



LSH-based Naïve Algorithm

- [Indyk, Motwani] Parameters L and t can be set s.t. With constant probability
  - Any neighbor of q falls into the same bucket as q in at least one hash function
  - Total number of **outliers** is at most $3L$
  - **Outlier** : point farther than cr from the query point



Algorithm

- Arrays for each hash function $A_1, \ldots, A_L$
- For a query $\boldsymbol{q}$ compute
  - Retrieve the possible neighbors $S = \bigcup_{i=1}^{L} \boldsymbol{A}[g_i(q)]$
  - Remove the outliers $\mathrm{S} = \mathrm{S} \cap \mathrm{B}(\mathrm{q}, \mathrm{cr})$
  - Report the approximate k most diverse points of S, or GMM(S)

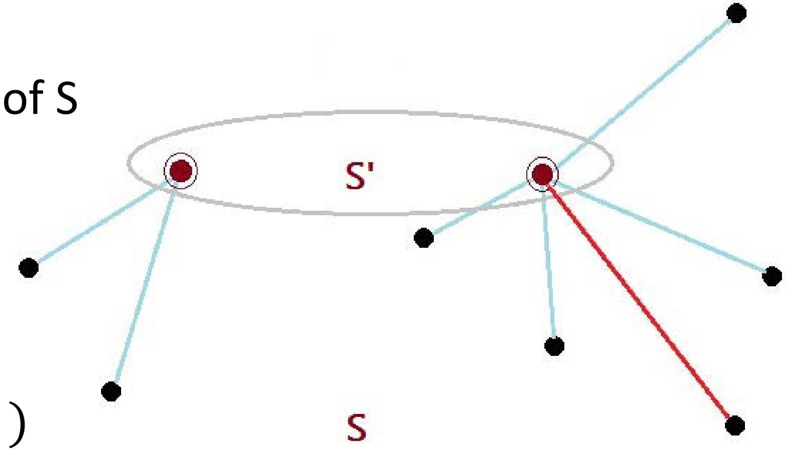
- Achieves (c,2)-approximation

- Running time may be linear in n ☹
  - Should prune the buckets before collecting them

Core-sets

- **Core-sets** [Agarwal, Har-Peled, Varadarajan]**:** subset of a point set **S** that represents it.
  - Approximately determines the solution to an optimization problem
  - Composes: A union of coresets is a coreset of the union
- **β– core-set:** Approximates the cost up-to a factor of β

- **Our Optimization problem:**
  - Finding the k-diversity of S.
  - Instead we consider finding **K-Center Cost** of S
    - $KC(S,S') = \max_{p \in S} \min_{p' \in S'} |p - p'|$
    - $KC_k(S) = \min_{S' \subseteq S, |S'| = k} KC(S,S')$
  - **KC cost 2-approximates diversity**
    - $KC_{k-1}(S) \leq div_k(S) \leq 2.KC_{k-1}(S)$



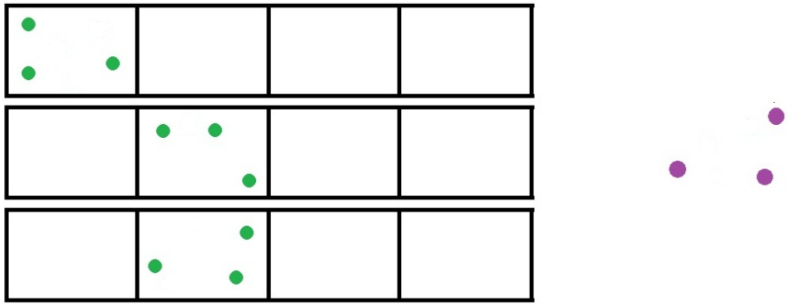
- **GMM** computes a 1/3-Coreset for KC-cost

Algorithms

Algorithm A

- Parameters L and t can be set s.t. with constant probability
  - Any neighbor of q falls into the same bucket as q in at least one hash function
  - There is no outlier

- No need to keep all the points in each bucket,
- just keep a coreset!
  - $\boldsymbol{A'}_{\boldsymbol{i}}[\boldsymbol{j}] = \boldsymbol{GMM}(\boldsymbol{A_i}[\boldsymbol{j}])$
  - Keep a 1/3 coreset of $\boldsymbol{A_i}[\boldsymbol{j}]$



- Given query $\boldsymbol{q}$
  - Retrieve the coresets from buckets $\mathrm{S} = \bigcup_{i=1}^{L} \boldsymbol{A'}[g_i(q)]$
  - Run GMM(S)
  - Report the result

Analysis

- Achieves (c,6)-Approx
  - Union of 1/3 coresets is a 1/3 coreset for the union
  - The last GMM call, adds a 2 approximation factor

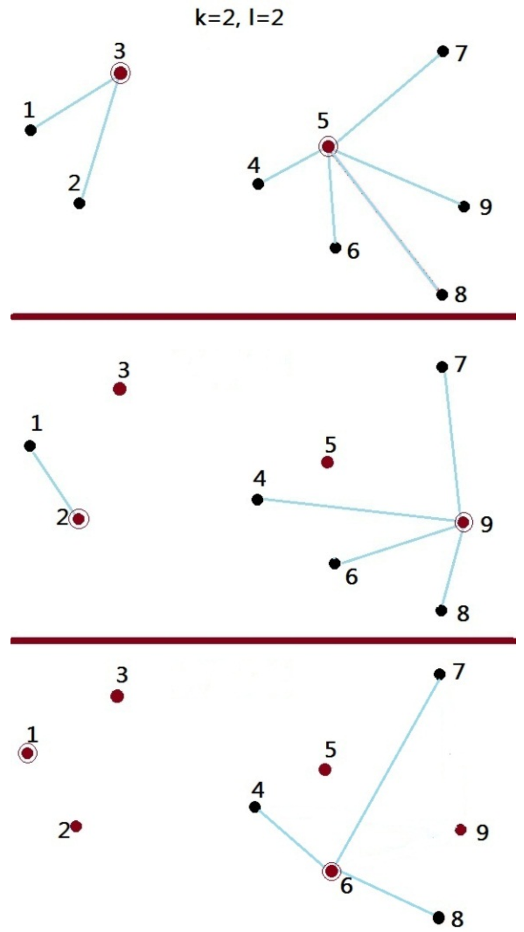
- **Only works** if we set L and t s.t. there is **no outlier** in S with constant probability
  - Space: $O(nL) = O((n \log k)^{1+1/(c-1)} + nd)$
  - Time: $O(Lk^2) = O(\left(k^2 + \frac{\log n}{r}\right) d \, (\log k)^{c/(c-1)} n^{1/(c-1)})$
  - Only makes sense for $c > 2$

- Not optimal:
  - ANN query time is $O(dn^{\frac{1}{c}})$
  - So if we want to improve over these we should be able to deal with outliers.

Robust Core-sets

- S' is an l-robust β-coreset for S if
  - for any set O of outliers of size at most l
  - $(S' \backslash O)$ is a β-coreset for S
- Peeling Algorithm [Agarwal, Har-peled, Yu,'06][Varadarajan, Xiao, '12]:
  - Repeat $(l + 1)$ times
    - Compute a β-coreset for S
    - Add them to the coreset S'
    - Remove them from the set S

Note: if we order the points in S' as we find them, then the first $(l' + 1)k$ points also form an l'-robust β-coreset.



2 robust coreset: S'= {3, 5; 2, 9; 1, 6}

1 robust coreset

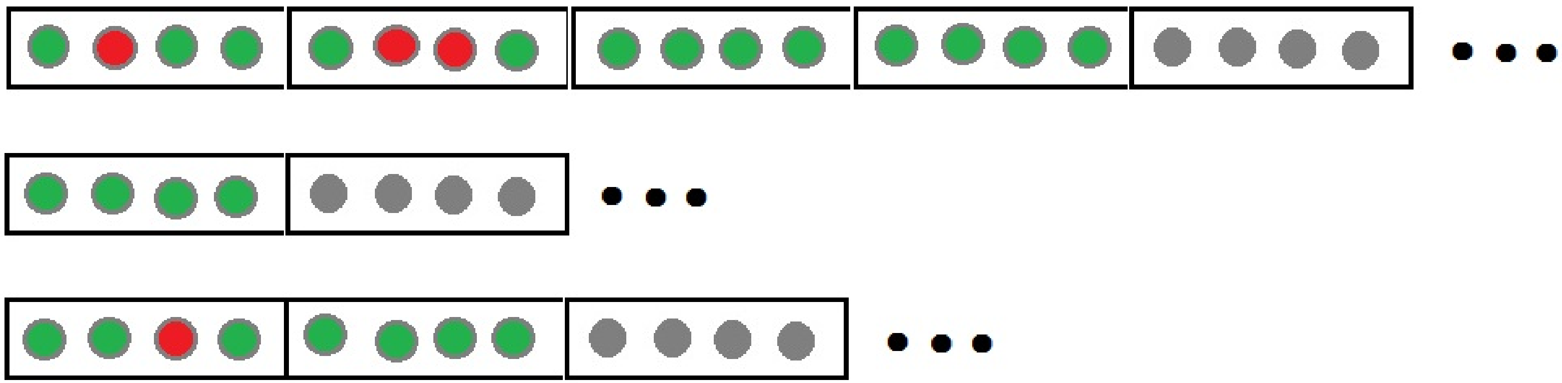
Algorithm B

- Parameters L and t can be set s.t. With constant probability
  - Any neighbor of q falls into the same bucket as q in at least one hash function
  - Total number of **outliers** is at most $3L$

- For each bucket $A_i[j]$ keep an $3L$-robust 1/3-coreset in $A'_i[j]$ which has size $(3L+1)k$

- For query q
  - For each bucket $A'[g_i(q)]$
    - Find smallest l s.t. the first (kl) points contains less than l outliers
    - Add those kl points to S
  - Remove outliers from S
  - Return $GMM(S)$

Example and Analysis



- Total # outliers $\leq 3L$, $|S| < O(Lk)$
- Time: $O(Lk^2) = O(\left(k^2 + \frac{\log n}{r}\right) d * \log k * n^{\frac{1}{c}})$
- Space: $O(nL) = O(\log k * n^{1+1/c} + nd)$
- Achieves (c,6)-Approx for the same reason

Conclusion

	Algorithm A	Algorithm B	ANN
Distance Apx. Factor	c > 2	c >1	c >1
Diversity Apx. Factor α	6	6	-
Space	$\sim n^{1+\frac{1}{c-1}}$	$\sim n^{1+\frac{1}{c}}$	$n^{1+\frac{1}{c}}$
Query Time	$\sim d\ n^{\frac{1}{c-1}}$	$\sim d\ n^{\frac{1}{c}}$	$d\ n^{\frac{1}{c}}$

Further Work

- Improve diversity factor α
- Consider other definitions of diversity , e.g., sum of distances

Thank You!