Near Neighbor Problem Made Fair

Sariel Har-Peled
UIUC

Sepideh Mahabadi
TTIC
Nearest Neighbor Problems

- **Nearest Neighbor**: Given a set of objects, find the closest one to the query object.
Nearest Neighbor Problems

• **Nearest Neighbor:** Given a set of objects, find the closest one to the query object.
Nearest Neighbor Problems

• **Nearest Neighbor:** Given a set of objects, find the closest one to the query object.

• **Near Neighbor:** given a set of objects, find one that is close enough to the query object.
There are many applications of NN

Searching for the closest object
Near Neighbor

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

Dataset of \( n \) points \( P \) in a metric space, e.g. \( \mathbb{R}^d \), and a parameter \( r \)

A query point \( q \) comes online
Near Neighbor

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

A query point $q$ comes online

Goal:

• Find a point $p^*$ in the $r$-neighborhood
Near Neighbor

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

A query point $q$ comes online

Goal:

• Find a point $p^*$ in the $r$-neighborhood
• Do it in sub-linear time and small space
Near Neighbor

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

A query point $q$ comes online

Goal:
• Find a point $p^*$ in the $r$-neighborhood
• 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 Near Neighbor

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

A query point $q$ comes online

Goal:

• Find a point $p^*$ in the $r$-neighborhood
• Do it in sub-linear time and small space
• Approximate Near Neighbor
  – Report a point in distance $cr$ for $c > 1$
Approximate Near Neighbor

Dataset of \( n \) points \( P \) in a metric space, e.g. \( \mathbb{R}^d \), and a parameter \( r \)

A query point \( q \) comes online

Goal:
- Find a point \( p^* \) in the \( r \)-neighborhood
- Do it in sub-linear time and small space
- **Approximate Near Neighbor**
  - Report a point in distance \( cr \) for \( c > 1 \)
  - For Hamming (and Manhattan) query time is \( n^{O(1/c)} \) [IM98]
  - and for Euclidean it is \( n^{O(\frac{1}{c^2})} \) [AI08]
Fair Near Neighbor

Sample a neighbor of the query uniformly at random

- Individual fairness: every neighbor has the same chance of being reported.
- Remove the bias inherent in the NN data structure
Fair Near Neighbor

Sample a neighbor of the query uniformly at random

- Individual fairness: every neighbor has the same chance of being reported.
- Remove the bias inherent in the NN data structure

Applications:
- Removing noise, k-NN classification
- Anonymizing the data
- Counting the neighborhood size
Fair Near Neighbor

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

A query point $q$ comes online

Goal:

- Return each point $p$ in the neighborhood of $q$ with uniform probability
- Do it in sub-linear time and small space
Approximate Fair Near Neighbor

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

A query point $q$ comes online

Goal of Approximate Fair NN

- Any point $p$ in $N(q, r)$ is reported with “almost uniform” probability, i.e., $\lambda_q(p)$ where

$$\frac{1}{(1 + \epsilon)|N(q, r)|} \leq \lambda_q(p) \leq \frac{(1 + \epsilon)}{|N(q, r)|}$$
Results on $(1 + \epsilon)$-Approximate Fair NN

<table>
<thead>
<tr>
<th>Domain</th>
<th>Guarantee</th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Neighborhood $N(q, r)$</td>
<td>w.h.p</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN} \cdot \frac{</td>
</tr>
</tbody>
</table>

$S_{ANN}$ and $T_{ANN}$ are the space and query time of standard ANN
Results on $(1 + \varepsilon)$-Approximate Fair NN

<table>
<thead>
<tr>
<th>Domain</th>
<th>Guarantee</th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Neighborhood</td>
<td>w.h.p</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN} \cdot \frac{</td>
</tr>
<tr>
<td>$N(q, r)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approximate Neighborhood</td>
<td>In expectation</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN})$</td>
</tr>
<tr>
<td>$N(q, r) \subseteq S \subseteq N(q, cr)$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$S_{ANN}$ and $T_{ANN}$ are the space and query time of standard ANN.

Approximate neighborhood: a set $S$ such that $N(q, r) \subseteq S \subseteq N(q, cr)$
Results on $(1 + \epsilon)$-Approximate Fair NN

<table>
<thead>
<tr>
<th>Domain</th>
<th>Guarantee</th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Neighborhood $N(q, r)$</td>
<td>w.h.p</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN} \cdot \frac{</td>
</tr>
<tr>
<td>Approximate Neighborhood $N(q, r) \subseteq S \subseteq N(q, cr)$</td>
<td>In expectation</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN})$</td>
</tr>
</tbody>
</table>

- $S_{ANN}$ and $T_{ANN}$ are the space and query time of standard ANN
- Approximate neighborhood: a set $S$ such that $N(q, r) \subseteq S \subseteq N(q, cr)$
- Dependence on $\epsilon$ is $O(\log(\frac{1}{\epsilon}))$
## Results on \((1 + \epsilon)\)-Approximate Fair NN

<table>
<thead>
<tr>
<th>Domain</th>
<th>Guarantee</th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Neighborhood (N(q, r))</td>
<td>w.h.p</td>
<td>(O(S_{ANN}))</td>
<td>(\tilde{O}(T_{ANN} \cdot \frac{</td>
</tr>
<tr>
<td>Approximate Neighborhood (N(q, r) \subseteq S \subseteq N(q, cr))</td>
<td>In expectation</td>
<td>(O(S_{ANN}))</td>
<td>(\tilde{O}(T_{ANN}))</td>
</tr>
</tbody>
</table>

- \(S_{ANN}\) and \(T_{ANN}\) are the space and query time of standard ANN
- Approximate neighborhood: a set \(S\) such that \(N(q, r) \subseteq S \subseteq N(q, cr)\)
- Dependence on \(\epsilon\) is \(O(\log(\frac{1}{\epsilon}))\)
- Experiments
Results on $(1 + \varepsilon)$-Approximate Fair NN

<table>
<thead>
<tr>
<th>Domain</th>
<th>Guarantee</th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Neighborhood $N(q, r)$</td>
<td>w.h.p</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN} \cdot \frac{</td>
</tr>
<tr>
<td>Approximate Neighborhood $N(q, r) \subseteq S \subseteq N(q, cr)$</td>
<td>In expectation</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN})$</td>
</tr>
</tbody>
</table>

- $S_{ANN}$ and $T_{ANN}$ are the space and query time of standard ANN
- Approximate neighborhood: a set $S$ such that $N(q, r) \subseteq S \subseteq N(q, cr)$
- Dependence on $\varepsilon$ is $O(\log(\frac{1}{\varepsilon}))$
- Experiments
- Recent paper [Aumuller, Pagh, Silvestry’19] defining the same notion
Locality Sensitive Hashing (LSH)

One of the main approaches to solve the Nearest Neighbor problems
Locality Sensitive Hashing (LSH)

Hashing scheme s.t. close points have higher probability of collision than far points
Locality Sensitive Hashing (LSH)

Hashing scheme s.t. close points have higher probability of collision than far points

Hash functions: $g_1, \ldots, g_L$

- $g_i$ is an independently chosen hash function
Locality Sensitive Hashing (LSH)

Hashing scheme: s.t. close points have higher probability of collision than far points

Hash functions: \( g_1, \ldots, g_L \)
- \( g_i \) is an independently chosen hash function

If \( ||p - p'|| \leq r \), they collide w.p. \( \geq P_{\text{high}} \)
If \( ||p - p'|| \geq cr \), they collide w.p. \( \leq P_{\text{low}} \)

For \( P_{\text{high}} \geq P_{\text{low}} \)
Retrieval: [Indyk, Motwani’98]

- The union of the query buckets is roughly the neighborhood of $q$

- $\bigcup_i B_i(g_i(q))$ is roughly the neighborhood
Locality Sensitive Hashing (LSH)

**Retrieval:** [Indyk, Motwani’98]

- The union of the query buckets is roughly the neighborhood of $q$

- $U_i B_i(g_i(q))$ is roughly the neighborhood

- How to report a uniformly random neighbor from union of these buckets?
Locality Sensitive Hashing (LSH)

**Retrieval:** [Indyk, Motwani’98]

- The union of the query buckets is roughly the neighborhood of $q$

$$U_i B_i(g_{i(q)})$$ is roughly the neighborhood

- How to report a uniformly random neighbor from union of these buckets?

- Collecting all points might take $O(n)$ time
Approaches
Approach 1: Uniform/Uniform

How to output a random neighbor from $U_i B_i(g_i(q))$:

1. Choose a uniformly random bucket
2. Choose a uniformly random point in the bucket
Approach 2: Weighted/Uniform

How to output a random neighbor from $U_i B_i(g_{i(q)})$:

1. Choose a random bucket proportional to its size
2. Choose a random point in the bucket
Approach 2: Weighted/Uniform

How to output a random neighbor from $\bigcup_i B_i(g_{i(q)})$:

1. Choose a random bucket proportional to its size
2. Choose a random point in the bucket
   - Each point $p$ in the neighborhood is picked w.p. proportional to its degree $d_p$

Number of buckets that $p$ appears in
Approach 3: Optimal

How to output a random neighbor from $U_i B_i(g_i(q))$:

1. Choose a random bucket proportional to its size
2. Choose a random point in the bucket
   ➢ Each point $p$ in the neighborhood is picked w.p. proportional to its degree $d_p$
3. Keep $p$ with probability $\frac{1}{d_p}$, o.w. repeat
1. Choose a random bucket proportional to its size

2. Choose a random point in the bucket
   - Each point \( p \) in the neighborhood is picked w.p. proportional to its degree \( d_p \)

3. Keep \( p \) with probability \( \frac{1}{d_p} \), o.w. repeat
   - Uniform probability
1. Choose a random bucket proportional to its size

2. Choose a random point in the bucket
   - Each point $p$ in the neighborhood is picked w.p. proportional to its degree $d_p$

3. Keep $p$ with probability $\frac{1}{d_p}$, o.w. repeat
   - Uniform probability
   - Need to spend $O(L)$ to find the degree

Approach 3: Optimal
Approach 3: Optimal

1. Choose a random bucket proportional to its size
2. Choose a random point in the bucket
   - Each point $p$ in the neighborhood is picked w.p. proportional to its degree $d_p$
3. Keep $p$ with probability $\frac{1}{d_p}$, o.w. repeat
   - Uniform probability
   - Need to spend $O(L)$ to find the degree
   - Might need $O(d_{max}) = O(L)$ samples
   - Total time is $O(L^2)$
Approximate the degree $d_p$

Sample $O\left(\frac{L}{d_p \cdot \varepsilon^2}\right)$ buckets out of $L$ buckets to $(1 + \varepsilon)$-approximate the degree.

- Still if the degree is low this takes $O(L)$ samples.
Approximate the degree $d_p$

Sample $O\left(\frac{L}{d_p \cdot \epsilon^2}\right)$ buckets out of $L$ buckets to $(1 + \epsilon)$-approximate the degree.

Still if the degree is low this takes $O(L)$ samples.

Case 1: Small degree $d_p$:
- More samples are required to estimate
- Reject with lower probability -> Fewer queries of this type

Case 2: Large degree $d_p$:
- Fewer samples are required to estimate
- Reject with higher probability -> More queries of this type
Approximate the degree $d_p$

Sample $O\left(\frac{L}{d_p \cdot \epsilon^2}\right)$ buckets out of $L$ buckets to $(1 + \epsilon)$-approximate the degree. 

- Still if the degree is low this takes $O(L)$ samples.

Case 1: Small degree $d_p$:
- More samples are required to estimate
- Reject with lower probability -> Fewer queries of this type

Case 2: Large degree $d_p$:
- Fewer samples are required to estimate
- Reject with higher probability -> More queries of this type

- This decreases $O(L^2)$ runtime to $\tilde{O}(L)$
- Large dependency on $\epsilon$ of the form $O\left(\frac{1}{\epsilon^2}\right)$
- Via a different sampling approach we show how to reduce the dependency to logarithmic $O\left(\log \frac{1}{\epsilon}\right)$. 
Experiments

Setup

• Take MNIST as the data set
• Ask a query several times and compute the empirical distribution of the neighbors.
• Compute the statistical distance of the empirical distribution to the uniform distribution
Experiments

Setup

• Take MNIST as the data set
• Ask a query several times and compute the empirical distribution of the neighbors.
• Compute the statistical distance of the empirical distribution to the uniform distribution

Comparison

• Our algorithm performs 2.5 times worse than the optimal algorithm, but the other two perform 7 and 10 times worse than the optimal.
• Four times faster than the optimal but 15 times slower than the other two
## Conclusion

<table>
<thead>
<tr>
<th>Domain</th>
<th>Guarantee</th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Neighborhood (N(q,r))</td>
<td>w.h.p</td>
<td>(O(S_{ANN}))</td>
<td>(\tilde{O}(T_{ANN} \cdot \frac{</td>
</tr>
<tr>
<td>Approximate Neighborhood (N(q,r) \subseteq S \subseteq N(q,cr))</td>
<td>In expectation</td>
<td>(O(S_{ANN}))</td>
<td>(\tilde{O}(T_{ANN}))</td>
</tr>
</tbody>
</table>

- The dependence on the parameter \(n\) matches the standard Nearest Neighbor.  
- We get an independent near neighbor each time we draw a sample.  
- More generally the approach works for sampling form a sub-collection of sets.
Conclusion

<table>
<thead>
<tr>
<th>Domain</th>
<th>Guarantee</th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Neighborhood $N(q, r)$</td>
<td>w.h.p</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN} \cdot \frac{</td>
</tr>
<tr>
<td>Approximate Neighborhood $N(q, r) \subseteq S \subseteq N(q, cr)$</td>
<td>In expectation</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN})$</td>
</tr>
</tbody>
</table>

✓ The dependence on the parameter $n$ matches the standard Nearest Neighbor.
✓ We get an independent near neighbor each time we draw a sample.
✓ More generally the approach works for sampling from a sub-collection of sets.

Open Problem:

- Finding the optimal dependency on the density parameter: $\frac{|N(q, cr)|}{|N(q, r)|}$
Conclusion

<table>
<thead>
<tr>
<th>Domain</th>
<th>Guarantee</th>
<th>Space</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Neighborhood $N(q, r)$</td>
<td>w.h.p</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN} \cdot \frac{</td>
</tr>
<tr>
<td>Approximate Neighborhood $N(q, r) \subseteq S \subseteq N(q, cr)$</td>
<td>In expectation</td>
<td>$O(S_{ANN})$</td>
<td>$\tilde{O}(T_{ANN})$</td>
</tr>
</tbody>
</table>

- The dependence on the parameter $n$ matches the standard Nearest Neighbor.
- We get an independent near neighbor each time we draw a sample.
- More generally the approach works for sampling from a sub-collection of sets.

Open Problem:
- Finding the optimal dependency on the density parameter: $\frac{|N(q, cr)|}{|N(q, r)|}$

Thanks
Questions?