Approximate Nearest Neighbor Search in High Dimensions

Piotr Indyk
MIT
Nearest Neighbor Search

• **Given:** a set $P$ of $n$ distinct points in a $d$-dimensional space $\mathbb{R}^d$ under some norm $\|\cdot\|$.

• **Goal:** build a data structure which, given any query $q \in \mathbb{R}^d$ returns a point $p \in P$ minimizing $\|p-q\|$

• **What is a data structure?**
  - A data structure of size $M$ is an array $D[1\ldots M]$ of numbers (“the memory”), together with an associated algorithm $A$ that, given a point $q$, returns a point in $P$ as specified above.
  - Example in a moment.
  - See [Fefferman-Klartag’09] for an exposition.

• **Want:**
  - Fast running time of the algorithm $A$
  - Small data structure size $M$
Nearest Neighbor Search

- Best match problem [Minsky-Papert’69], Post office problem [Knuth’73]
- Broad applications in computer science, machine learning etc
  - E.g., searching for similar audio files, images, videos, etc
  - Google “wiki” “nearest neighbor search”
  - Think $n \gg 10^6$, $d > 50$
- Many connections to geometric functional analysis, discrete metric spaces, etc.
Example: d=1

• Pointset $P$: $x_1 < x_2 \ldots < x_n$, $x_i \in \mathbb{R}$
• Query: $q \in \mathbb{R}$
• Nearest neighbor: equivalent to finding smallest $x_i$ greater than $q$ ("successor" of $q$)
• Performance:
  – Query time: $O(\log n)$ (binary search)
  – Space: $O(n)$ (suffices to store sorted input)
Example: d=2

- Space partitioning: Voronoi diagram
  - Combinatorial complexity $O(n)$
- Given $q$, find the cell $q$ belongs to (point location)
- Performance [Lipton-Tarjan’80]
  - Query time: $O(\log n)$
  - Space: $O(n)$
The case of $d>2$

- Voronoi diagram has size $n^{\lceil d/2 \rceil}$
  - $n^{O(d)}$ space, $(d+ \log n)^{O(1)}$ time [Dobkin-Lipton’78, Meiser’93, Clarkson’88]
- We can also perform a linear scan: $O(dn)$ space, $O(dn)$ time
  - Can speedup the scan time by roughly $O(n^{1/d})$
- These are pretty much the only known general solutions!
- In fact, exact algorithm with $n^{1-\beta}$ query time for some $\beta>0$ and $\text{poly}(n)$ preprocessing would violate certain complexity-theoretic conjecture (SETH)
  - See next lecture by V. V. Williams
Approximate Nearest Neighbor

- **Given:** a set $P$ of $n$ points in a $d$-dimensional space $\mathbb{R}^d$ under some norm $||.||$, parameter $c > 1$
- **Goal:** data structure which, given any query $q$ returns $p' \in P$, where

$$||p' - q|| \leq c \min_{p \in P} ||p - q||$$
(c,r)-Approximate Near Neighbor

- **Given:** a set $P$ of $n$ points in a $d$-dimensional space $\mathbb{R}^d$ under some norm $\|\cdot\|$, parameters $c>1$ and $r>0$
- **Goal:** build a data structure $D$ which, for any query $q$:
  - If there is $p \in P$ s.t. $\|q-p\| \leq r$,
  - Then return $p' \in P$ s.t. $\|q-p'\| \leq cr$
- Decision version of approximate nearest neighbor
  - Equivalent up to $(\log n)^O(1)$ factors in space and query time
- Randomized version $(c,r,\delta)$-ANN: for any query $q$
  $$\Pr_D[D \text{ answers } q \text{ as above}] > 1-\delta$$
Approximate Near(est) Neighbor Algorithms

• Space/time **exponential in** $d$ [Arya-Mount’93],[Clarkson’94], [Arya-Mount-Netanyahu-Silverman-Wu’98] [Kleinberg’97], [Har-Peled’02], ....

• Space/time **polynomial in** $d$ [Indyk-Motwani’98], [Kushilevitz-Ostrovsky-Rabani’98], [Indyk’98], [Gionis-Indyk-Motwani’99], [Charikar’02], [Datar-Immorlica-Indyk-Mirrokni’04], [Chakrabarti-Regev’04], [Panigrahy’06], [Ailon-Chazelle’06], [Andoni-Indyk’06],….., [Andoni-Indyk-Nguyen-Razenshteyn’14], [Andoni-Razenshteyn’15] [Andoni-Indyk-Laarhoven-Razenshteyn-Schmidt’15], [Andoni-Nguyen-Nikolov-Razenshteyn-Waingarten’17], [Andoni-Naor-Nikolov-Razenshteyn-Waingarten’18], …
Plan

• Non-adaptive approach: \( l_1 \), \( l_2 \) and friends
  – Dimensionality reduction
  – Randomized space partitions (a.k.a. Locality-Sensitive Hashing)

• Adaptive approach: faster, more general
Non-adaptive data structures
Dimensionality reduction

- Consider approximation $c = 1 + \varepsilon \leq 2$

- Two steps:
  - Design a data structure with
    - Space: $(1/\varepsilon)^O(d)$
    - Query time: $O(d)$
  - Use random projection [Johnson-Lindenstrauss’84]
    - Dimension: $d \rightarrow O(\log(n)/\varepsilon^2)$
    - All distances preserved up to $1 \pm \varepsilon$ (in $l_2$)

- Yields space $n^{O(1/\varepsilon^2)}$ and query time $O(d \log(n)/\varepsilon^2)$ [Ostrovski-Rabani’98]

- Space too large to be practical
Locality-Sensitive Hashing (LSH)

• A family $H$ of functions $h: \mathbb{R}^d \rightarrow U$ is called $(P_1, P_2, r, cr)$-sensitive for $||.||$, if for any pair of points $p, q$:
  - If $||p - q|| \leq r$ then $\Pr[h \in H \mid h(p) = h(q)] \geq P_1$
  - If $||p - q|| \geq cr$ then $\Pr[h \in H \mid h(p) = h(q)] \leq P_2$

• Theorem [Indyk-Motwani'98]: Suppose there is $H$ as above. Then there is a $(c, r, 0.1)$-ANN data structure with:
  - Space: $O(\frac{dn + nL}{r})$
  - Time: $O(\frac{dL}{r})$

where $L = \frac{n}{P_1}$, $r = \log(P_1)/\log(P_2)$
Locality-Sensitive Hashing

- A family $H$ of functions $h: \mathbb{R}^d \rightarrow U$ is called $(P_1, P_2, r, cr)$-sensitive for $\|\cdot\|$, if for any pair of points $p, q$:
  - If $\|p-q\| \leq r$ then $Pr_{h \in H} [ h(p)=h(q) ] \geq P_1$
  - If $\|p-q\| \geq cr$ then $Pr_{h \in H} [ h(p)=h(q) ] \leq P_2$

- Theorem [Indyk-Motwani'98]: Suppose there is an $H$ as above. Then there is a $(c, r, 0.1)$-ANN data structure with space $O(dn+nL)$ and time $O(dL)$ where $L=n^\rho/P_1$, $\rho=\log(P_1)/\log(P_2)$

- Non-adaptive: the memory cells accessed to answer queries depend on query $q$ but not on data set $P$
LSH: examples

• $\{0,1\}^d$ under $\|\cdot\|_1$:
  – $H=\{h_i : h_i(p) = p_i, i=1..d\}$
  – $\Pr_{h \in H}[h_i(p) = h_i(q)] = 1 - \|p-q\|_1 / d$
  – Yields exponent $\rho = 1/c$

• Works for $\mathbb{R}^d$ under $\|\cdot\|_p$, $p \in [1,2]$
LSH: examples

- $\mathbb{R}^d$ under $\|\cdot\|_2$ [Datar-Indyk-Immorlica-Mirrokni’04]
  - Project on a random 1-dimensional space and round
  - Yields exponent $\rho < 1/c$

\[ LSH: \text{examples} \]

\[ \begin{align*}
\text{Year} & \quad \text{Explained} \\
1998 & \quad 0.5 \\
2004 & \quad 0.45
\end{align*} \]
LSH: examples

- $\mathbb{R}^d$ under $\|\cdot\|_2$
- Project (on a $t$-space) and round
  - [Charikar et al’98, Andoni-Indyk’06]
  - Intervals $\rightarrow$ lattice of balls
  - Can hit empty space, so hash until a ball is hit
  - Yields exponent $\rho \rightarrow 1/c^2$ as $t \rightarrow \infty$

$\rho$ for $l_2$
c=2

0.5
0.45
0.25

1998 2004 2006

[Motwani-Naor-Panigrahy’06, O’Donnell-Wu-Zhou’09]:
Any LSH in $l_2$ must have $\rho \geq 1/c^2 - o(1)$ or $P_1 < \exp(-a d)$ for some $a>0$
Adaptive data structures
The “idea”

- Why is the answer not obvious?
- It is often possible to get a data structure that works well when the data has some structure (clusters, low-dimensional subspace, i.i.d. from some distribution, etc).
- The tricky part is what to do when the data does not have that structure, or any structure in particular.

What if the data structure depended on …the data?
The actual idea

• Every point-set has some structure that can be exploited algorithmically

• Details depend on the context/problem, but at a high level:
  – Either there is dense cluster of small radius, or
  – Points are “spread” out

• Applications:
  – Faster algorithms for $l_1$, $l_2$
  – Algorithms for general norms
Faster Algorithms
**Basic Data Adaptive Method**

P= input pointset, r = radius, c = approximation

**Preprocessing:**
1. As long as there is a ball $B_i$ of radius $O(cr)$ containing $T$ points in $P$
   - $P = P - B_i$
   - $i = i + 1$
2. Build LSH data structure on $P$
   No dense clusters – most points are $>> cr$ from $q$
3. For each ball $B_i$ build a specialized data structure for $B_i \cap P$
   Diameter bounded by $O(cr)$ – better LSH functions

**Query procedure:**
1. Query the main data structure
2. Query all data structures for balls that are “close” to the query
Results (for $l_2$)

- For $c$-approximation:

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Query Time</th>
<th>Index Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-adaptive LSH</td>
<td>$dn^{1/c^2}$</td>
<td>$n^{1+1/c^2}$</td>
</tr>
<tr>
<td>Andoni, Indyk, Nguyen, Razenshteyn’14</td>
<td>$dn^{0.87/c^2+O(1)/c^3}$</td>
<td>$n^{1+0.87/c^2+O(1)/c^3}$</td>
</tr>
<tr>
<td>Andoni-Razenshteyn’15</td>
<td>$dn^{1/(2c^2-1)}$</td>
<td>$n^{1+1/(2c^2-1)}$</td>
</tr>
</tbody>
</table>

Graph showing $\rho$ for $l_2$ and $c=2$.
More general algorithms
Generality

• Non-adaptive methods:
  – Dimensionality reduction: mostly $l_2$
    • No dimensionality reduction in $l_1$ [Brinkman-Charikar’03, Lee-Naor’04]
    • Any space supporting dimensionality reduction with low distortion is “very close” to $l_2$ [Johnson-Naor’09]
  – Locality-sensitive hashing: $l_p$ for $p \in [1,2]$, Jaccard coefficient, Angular distance etc
    • Does not work e.g., for $l_\infty$
  – Reductions:
    • Small powers of the above
    • Low-distortion embeddings into the above (edit distance, Ulam metric, transportation norm,..)

• What about general norms?
General norms

• Every $d$-dimensional normed space is within $\sqrt{d}$ from $\ell_2^d$ (after a linear transformation) [John’48]
  – Yields approximation factor of $O(\sqrt{d})$ – pretty large
• Low-distortion embedding of any symmetric norm [Andoni-Nguyen-Nikolov-Razenshteyn-Waingarten’17]
  – Embedding into $\bigoplus \ell_\infty \bigoplus \ell_1 \ell_\infty$
  – Yields approximation factor of $\text{poly}(\log \log n)$
• Algorithms for any norm via cutting modulus [Andoni-Naor-Nikolov-Razenshteyn-Waingarten’18]
  – Yields approximation factor of $O(\log d)$
  – The algorithm operates in the ”cell-probe” model (counts only memory accesses, not computation)
  – Can be converted into an “actual” algorithm for specific norms or with a weaker guarantee
Cutting modulus

- Parameter $\Xi(M, \alpha)$ defined for any metric space $M = (X, D)$ and “error parameter” $\alpha > 0$
- It is at most $O(\log(d)/\alpha^2)$ for any normed space $\|\|\|\|_\cdot$ over $\mathbb{R}^d$ [Naor’17]
- Related to non-linear spectral gaps
  - See the talk by Assaf Naor next week
The core partitioning procedure

- Theorem:
  - Let $M = (X, D)$ with $|X| = N$ and take $\alpha, r > 0$
  - There is a “small” collection $\mathcal{F} \subset 2^X$ s.t. for every $n$-point dataset $P \subset X$:
    - Either there exists a ball of radius $\lesssim \mathbb{E}(M, \alpha) \cdot r$ with $\Omega(n)$ points
    - Or there is a distribution $\mathcal{D}$ over “few” sets from $\mathcal{F}$ that partition $P$ (approximately) evenly and, for every $x_1, x_2 \in X$ with $D(x_1, x_2) \leq r$:
      $$\Pr_{A \sim \mathcal{D}}[A \text{ separates } x_1 \text{ and } x_2] \lesssim \alpha$$
  - ANN data structure can be constructed using divide and conquer approach
Conclusions + Open Problems

• Approximate Nearest Neighbor Search
  – Non-adaptive approach: $l_1$, $l_2$ and friends
  – Adaptive approach: faster, more general
• Connections to geometric and metric functional analysis
• Open questions:
  – Deterministic algorithms? Very little known
  – Better data structure for edit distance?
    • No poly(n) space, $n^{1-\beta}$ query time, $\text{poly}(\log(d))$-approx. known
  – Same for transportation norm, but replace $\text{poly}(\log(d))$ with $\text{poly}(\log\log(d))$
• Software: google “FALCONN”

THANK YOU!
LEFTOVERS
ANN-Benchmarks (third party)

Info
ANN-Benchmarks is a benchmarking environment for approximate nearest neighbor algorithms search. This website contains the current benchmarking results. Please visit http://github.com/maumueller/ann-benchmarks/ to get an overview over evaluated data sets and algorithms. Make a pull request on Github to add your own code or improvements to the benchmarking system.

Benchmarking Results
Results are split by distance measure and dataset. In the bottom, you can find an overview of an algorithm’s performance on all datasets. Each dataset is annotated by \( k = \ldots \), the number of nearest neighbors an algorithm was supposed to return. The plot shown depicts \textit{Recall} (the fraction of true nearest neighbors found, on average over all queries) against \textit{Queries per second}. Clicking on a plot reveals detailed interactive plots, including approximate recall, index size, and build time.

Machine Details
All experiments were run in Docker containers on Amazon EC2 \textit{c4.2xlarge} instances that are equipped with Intel Xeon E5-2666v3 processors (4 cores available, 2.90 GHz, 25.6MB Cache) and 15 GB of RAM running Amazon Linux. For each parameter setting and dataset, the algorithm was given thirty minutes to build the index and answer the queries.

Raw Data & Configuration
Please find the raw experimental data here (13 GB). The query set is available queries-sisap.tar (7.5 GB) as well. The algorithms used the following parameter choices in the experiments: \( k = 10 \) and \( k=100 \).

Updates
- 18-10-2017: Included FAISS-IVF

Contact
ANN-Benchmarks has been developed by Martin Aumueller (maau@itu.dk), Erik Bernhardsson (mail@erikbern.com), and Alec Faitfull (alef@itu.dk). Please use Github to submit your implementation or improvements.
ANN-Benchmarks (third party)

Fig. 4. Recall-QPS (1/s) tradeoff - up and to the right is better. Top: GLOVE, bottom: SIFT; left: 10-NN, right: 100-NN.

Aumuller, Bernhardsson, Faithfull, SISAP’17
Locality-Sensitive Hashing

• A family $H$ of functions $h : \mathbb{R}^d \rightarrow U$ is called $(P_1, P_2, r, cr)$-sensitive for $\| . \|$, if for any pair of points $p, q$:
  – If $\| p - q \| \leq r$ then $\Pr_{h \in H} [ h(p) = h(q) ] \geq P_1$
  – If $\| p - q \| \geq cr$ then $\Pr_{h \in H} [ h(p) = h(q) ] \leq P_2$

• Theorem [Indyk-Motwani’98]: Suppose there is an $H$ as above. Then there is a $(c, r, 0.1)$-ANN data structure with space $O(dn + nL)$ and time $O(dL)$ where $L = n^\rho / P_1$, $\rho = \log(P_1) / \log(P_2)$

• Proof outline:
  – By “powering” $H$, set $P_2 = \frac{1}{2^n}$, so the expected number of “far” points from $P$ colliding with $q$ under $h$ is $< 1/2$
  – Probability of $q$ colliding with a “close” point is $P_1 = P_2^\rho = \left( \frac{1}{2^n} \right) ^\rho$
  – Repeat $O(n^\rho)$ to ensure good probability of success
  – Non-adaptive - the memory cells accessed to answer queries depend on query $q$ but not on data set $P$
Results

• Theory (c-approximation):

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Query Time</th>
<th>Index Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data-oblivious LSH(…FOCS’06)</td>
<td>$dn^{1/c^2}$</td>
<td>$n^{1+1/c^2}$</td>
</tr>
<tr>
<td>Andoni, Indyk, Nguyen, Razenshteyn, SODA’14</td>
<td>$dn^{0.87/c^2 +o(1/c^3)}$</td>
<td>$n^{1+0.87/c^2 +o(1/c^3)}$</td>
</tr>
<tr>
<td>Andoni-Razenshteyn, STOC’15</td>
<td>$dn^{1/(2c^2-1)}$</td>
<td>$n^{1+1/(2c^2-1)}$</td>
</tr>
</tbody>
</table>

• Practice:
  – Andoni-Indyk-Laarhoven-Razenshteyn-Schmidt (NIPS’15)
  – FALCONN (Razenshteyn-Schmidt)
    • Cross-polytope LSH, plus multi-probe, fast random projections,…
    • https://github.com/FALCONN-LIB/FALCONN
Conclusions

• Data helps designing data structures (duh…)
• ..provably, for general high-dimensional pointsets
• Code:
  – FALCONN: https://github.com/FALCONN-LIB/FALCONN
  – Quadsketch: https://github.com/talwagner/quadsketch
• Open problem: make these data structures dynamic (a.k.a. “real-time”)
Cutting modulus

- Parameter of any metric space $M = (X, D)$
- It is at most $O(\log d)$ for any normed space $\|\cdot\|_2$ over $\mathbb{R}^d$ [Naor’17] the cutting modulus $\Xi(M, \varepsilon)$ is the smallest number such that, for any graph embedded into $M$ with edges of “length” at most $K$
  - Either there is a ball of radius $K \cdot \Xi(M, \varepsilon)$ containing $\Omega(n)$ vertices
  - Or the graph has an $\varepsilon$-sparse cut
- Actual definition involves weighted edges
  - See talk by Assaf Naor next week