New Algorithms for Similarity Search in High Dimensions

Piotr Indyk
MIT
Given: a set $P$ of $n$ points in a $d$-dimensional space

**Nearest Neighbor:** for any query $q$, returns a point $p \in P$ minimizing $||p-q||$

Want:
- Fast query time
- Use small extra space
Example application

- Near-duplicate Retrieval

To be or not to be

(... , 1, ..., 4, ..., 2, ..., 2, ...)

(... , 6, ..., 1, ..., 3, ..., 6, ...)

(... , 1, ..., 3, ..., 7, ..., 5, ...)

(... , 2, ..., 2, ..., 1, ..., 1, ...)
The case of $d=2$

- Compute Voronoi diagram
- Given $q$, find the cell $q$ belongs to (point location)
- Performance:
  - Space: $O(n)$
  - Query time: $O(\log n)$
The case of $d>2$

- Voronoi diagram has size $n^{[d/2]}$
  - [Dobkin-Lipton’78, Meiser’93, Clarkson’88]: $n^{O(d)}$ space, $(d + \log n)^{O(1)}$ time

- We can also perform a linear scan: $O(dn)$ space and time
  - Can speedup the scan by roughly $O(n^{1/d})$

- These are pretty much the only known general solutions!
  - Kd-trees, Cover Trees, LSH, Spill trees, etc all denegerate to one of the two extremes in the worst case

- In fact:
  - If there was an algorithm that performs $n$ queries in $O(dn^{1+\alpha})$ time for $\alpha < 1$ (incl. preprocessing)
  - Then there would be an algorithm solving satisfiability in (slightly) sub-exponential time [Williams’04]
Approximation to the Rescue

- c-Approximate Nearest Neighbor: build data structure which, for any query \( q \)
  - returns \( p' \in \mathbb{P}, \|p-q\| \leq cr \),
  - where \( r \) is the distance to the nearest neighbor of \( q \)
Approximate \textbf{Near Neighbor}

- \textit{c-Approximate r-Near Neighbor}: build data structure which, for any query \(q\):
  - If there is a point \(p \in P\), \(||p-q|| \leq r\)
  - it returns \(p' \in P\), \(||p-q|| \leq cr\)

- \textit{Most algorithms randomized}:
  - For each query \(q\), the probability (over the randomness used to construct the data structure) is at least 90%
Approximate Near Neighbor Algorithms

• Huge area [Arya-Mount-et al’94], [Kleinberg’97], [Har-Peled’02], [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], and many more

• In this talk we focus on:
  – Euclidean norm
  – Polynomial dependence on dimension
  – Near-linear space: $dn + n^{1+\rho(c)}$, $\rho(c)<1$
  – Sub-linear query time: $dn^{\rho(c)}$
Locality-Sensitive Hashing

• Idea: construct hash functions $g: \mathbb{R}^d \rightarrow U$ such that for any points $p, q$:
  
  – If $||p-q|| \leq r$, then $\Pr[g(p)=g(q)]$ is “high” “not-so-small”
  
  – If $||p-q|| > cr$, then $\Pr[g(p)=g(q)]$ is “small”

• Then we can solve the problem by hashing
LSH - Formally

• A family $H$ of functions $h: \mathbb{R}^d \rightarrow U$ is called $(P_1,P_2,r,cr)$-sensitive for distance $D$, if for any pair of points $p,q$:
  – If $D(p,q) < r$ then $\Pr[ h(p)=h(q) ] > P_1$
  – If $D(p,q) > cr$ then $\Pr[ h(p)=h(q) ] < P_2$
Basic algorithm

• We use combine several LSH functions to form
  \[ g(p) = \langle h_1(p), h_2(p), \ldots, h_k(p) \rangle \]

• Preprocessing:
  – Select \( g_1 \ldots g_L \)
  – For all \( p \in P \), hash \( p \) to buckets \( g_1(p) \ldots g_L(p) \)

• Query:
  – Retrieve the points from buckets \( g_1(q), g_2(q), \ldots \), until
    • Either the points from all \( L \) buckets have been retrieved, or
    • Total number of points retrieved exceeds \( 3L \)
  – Answer the query based on the retrieved points
  – Total time: \( O(dL) \)
LSH functions?

- Project onto $\mathbb{R}^1$, discretize into intervals [Datar-Indyk-Immorlica-Mirrokni, SoCG’04]
- Project onto $\mathbb{R}^t$, for constant $t$ [Andoni-Indyk, FOCS’06]
- Intervals $\rightarrow$ lattice of balls [Charikar et al’98]
  - Can hit empty space, so hash until a ball is hit
- Analysis:
  - $\rho \rightarrow 1/c^2$ as $t \rightarrow \infty$
  - [Motwani-Naor-Panigrahya’06] [O’Donnell-Wu-Zhou’09]: best exponent $\rho \geq 1/c^2$
- End of story!
- …or so we thought
Data-dependent hashing

[Andoni-Indyk-Nguyen-Razenshtyten, SODA’14]
[Andoni-Razenshtyten, STOC’15]
Data-dependent hashing

• The aforementioned LSH schemes are optimal for \textit{data oblivious} hashing
  – Select $g_1 \ldots g_L$ independently at random
  – For all $p \in \mathcal{P}$, hash $p$ to buckets $g_1(p) \ldots g_L(p)$
  – Retrieve the points from buckets $g_1(q), g_2(q), \ldots$

• For this model, the “lattice of balls” hash functions are optimal

• The new schemes use hash functions that are \textit{data dependent}
Basic Algorithm

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

Preprocessing:
1. As long as there is a ball $B_i$ of radius $O(cr)$ containing $n^{1-\beta}$ points in $P$
   - $P=P-B_i$
   - $i=i+1$
2. Build the basic LSH data structure on $P$
   No dense clusters – better performance
3. For each ball $B_i$ build a specialized data structure for $B_i \cap P$
   Diameter bounded by $O(cr)$ – better performance

Query procedure:
1. Query the main data structure
2. Query all data structures for balls that are “close” to the query
Practical variants
[Andoni-Indyk-Laarhoven-Razenshtyen-Schmidt,NIPS’15]
Lattice of balls LSH

- We have $\rho \to 1/c^2$ as $t \to \infty$
- In fact: $\rho = 1/c^2 + O(\log t / t^{1/2})$
  - Need $t \approx 30$ to see some gains
- Time to hash is $t^{O(t)}$
- Not practical 😞
  - In practice still $\rho(2) = 0.45$
Making it practical: Cross-polytope LSH

• Cross-polytope LSH introduced by [Terasawa, Tanaka 2007]:
  - To hash $p$, apply a random rotation $S$ to $p$
  - Set hash value to a vertex of a cross-polytope $\{\pm e_i\}$ closest to $Sp$
• Can replace rotation by pseudo-random rotation
  - $O(d \log d)$ time
  (see Rachel Ward’s talk)
• We show this has almost the same quality as lattice of balls
  (for $d$ large enough)
Second idea: Improve on memory consumption

- *LSH consumes lots of memory:* myth or reality?
- For $n = 10^6$ random points and queries within 45 degrees, need 725 tables for success probability 0.9 (if using Hyperplane LSH)
- Can be reduced substantially via **Multiprobe LSH** [Lv, Josephson, Wang, Charikar, Li 2007]
- **Our contribution:** Multiprobe for Cross-polytope LSH
FALCONN
[Razenshteyn-Schmidt’05]

- **Glove** [Pennington, Socher, Manning 2014] $n = 1.2M$, $d = 100$, aim at **10 nearest neighbors**
  - 16-bit hashes
  - 1...1400 tables
  - Single probe
  - Accuracy 0.016...0.99
  - 10µs to 8.5ms query
  - From 5 Mb to 7 Gb
Conclusions

- Overview of LSH
- Substantial progress over the years (in theory and in practice)

Extensions (with Sepideh Mahabadi et al)
- Structured NN (query points are related somehow)
- Diversity-aware NN (diversity of answers matter)

Hope this is not the end of the story…