Locality sensitive hashing (LSH)
Nearest Neighbor

Given a set $P$ of $n$ points in $\mathbb{R}^d$
Nearest Neighbor

Want to build a data structure to answer nearest neighbor queries
Voronoi Diagram

Build a Voronoi diagram & a point location data structure
Curse of dimensionality

- In $\mathbb{R}^2$ the Voronoi diagram is of size $O(n)$
- Query takes $O(\log n)$ time
- In $\mathbb{R}^d$ the complexity is $O(n^{\lceil d/2 \rceil})$
- Other techniques also scale bad with the dimension
Problem has many variants

• Approximate (will talk about this soon)
• k nearest neighbors
• All neighbors at distance $\leq r$
All nearest neighbors

• Finding the nearest neighbors of every point in the data (near duplicate web pages)
Locality Sensitive Hashing

- We will use a family of hash functions such that close points tend to hash to the same bucket.

- Put all points of $P$ in their buckets, ideally we want the query $q$ to find its nearest neighbor in its bucket.
Locality Sensitive Hashing

A family $H$ of functions is $(d_1 < d_2, p_1 > p_2)$-sensitive if

$$d(p, q) \leq d_1 \Rightarrow \Pr[h(p) = h(q)] \geq p_1$$

$$d(p, q) \geq d_2 \Rightarrow \Pr[h(p) = h(q)] \leq p_2$$
Locality Sensitive Family for a distance function $d(x, y)$
LSF for hamming distance

Think of the points as strings of $m$ bits and consider the hamming distance $h(p, q)$

$H=\{h_i(p) = \text{the } i\text{-th bit of } p\}$ is locality sensitive wrt $h(p, q)$

$\Pr[h(p) = h(q)] = 1 - \frac{\text{ham}(p, q)}{m}$

So this family is $(d_1, d_2, 1 - \frac{d_1}{m}, 1 - \frac{d_2}{m})$-sensitive
Jaaccard distance and random permutations

Think of p and q as sets

\[ \text{jaccard}(p,q) = \frac{|p \cap q|}{|p \cup q|} \]

\( Jd(p,q) = 1 - \text{jaccard}(p,q) = 1 - \frac{|p \cap q|}{|p \cup q|} \)

\( H = \{ h_\pi(p) = \min \text{ in } \pi \text{ of the items in } p \} \)

\[ \Pr[h_\pi(p) = h_\pi(q)] = \text{jaccard}(p,q) \]

Efficiency: Need to pick \( \pi \) from a min-wise ind. family of permutations

This is \((d_1 = 1 - j_1, d_2 = 1 - j_2, j_1, j_2)\)-sensitive family
Jaaccard distance and minhash

Think of \( p \) and \( q \) as sets

\[
Jd(p,q) = 1 - \text{jaccard}(p,q) = 1 - \frac{|p \cap q|}{|p \cup q|}
\]

\[H = \{h_r(p) = \min r(x), x \in p, r \sim U[0,1]\}\]

\[
\Pr[h_r(p) = h_r(q)] = \text{jaccard}(p,q)
\]

Precision for \( r \) should avoid ties

This is \((d_1 = 1 - j_1, d_2 = 1 - j_2, j_1, j_2)\)-sensitive family
Cosine distance

$p$ and $q$ are vectors and $d(p, q) = \theta$
Cosine distance

\[ H = \{ h_r(p) = 1 \text{ if } r \cdot p > 0, \ 0 \text{ otherwise } \mid r \text{ is a random unit vector} \} \]
Cosine distance

\[ H = \{ h_r(p) = 1 \text{ if } r \cdot p > 0, \ 0 \text{ otherwise} \mid r \text{ is a random unit vector} \} \]

\[ \Pr[h_r(p) = h_r(q)] = ? \]
Cosine distance

$$H = \{ h_r(p) = 1 \text{ if } r \cdot p > 0, \ 0 \text{ otherwise } \mid r \text{ is a random unit vector} \}$$

$$\Pr[h_r(p) = h_r(q)] = 1 - \frac{\theta}{\pi}$$
Cosine distance

\[ H = \{ h_r(p) = 1 \text{ if } r \cdot p > 0, \ 0 \text{ otherwise } | \ r \text{ is a random unit vector}\} \]

This is \((\theta_1, \theta_2, 1 - \frac{\theta_1}{\pi}, 1 - \frac{\theta_2}{\pi})\)-sensitive family
Cosine distance

\[ H = \{ h_r(p) = 1 \text{ if } r \cdot p > 0, \ 0 \text{ otherwise } | \ r \text{ is a random unit vector} \} \]

For binary vectors (like term-doc incidence vectors):

\[ \theta = \cos^{-1} \left( \frac{|A \cap B|}{\sqrt{|A||B|}} \right) \]

This is \( \left( \theta_1, \theta_2, 1 - \frac{\theta_1}{\pi}, 1 - \frac{\theta_2}{\pi} \right) \)-sensitive family
Combining by “AND”

Reduce the number of false positives by concatenating hash function to get a new family of hash functions

\[ h(p) = h_1(p)h_2(p)h_3(p)h_4(p)\ldots \ldots h_k(p) = 00101010 \]

We get a new family of hash functions

\[ h(p) = h(q) \text{ iff } h_i(p) = h_i(q) \quad \forall i \]

If the original family is \((d_1, d_2, p_1, p_2)\)-sensitive
then the new family is \((d_1, d_2, (p_1)^k, (p_2)^k)\)-sensitive
Combining by “OR”

Reduce the number of false negatives

\[ h(p) = h_1(p), h_2(p), h_3(p), h_4(p), \ldots, h_L(p) = 0, 0, 1, 0, 1, 0, 1, 0 \]

We get a new family of hash functions

\[ h(p) = h(q) \text{ iff } \exists i \text{ s.t. } h_i(p) = h_i(q) \]

If the original family is \((d_1, d_2, p_1, p_2)\)-sensitive then the new family is

\((d_1, d_2, 1 - (1 - p_1)^L, 1 - (1 - p_2)^L)\)-sensitive
“And k” followed by “Or L”

\((d_1, d_2, p_1, p_2)\)-sensitive \(\Rightarrow\)

\((d_1, d_2, 1 - (1 - (p_1)^k)^L, 1 - (1 - (p_2)^k)^L)\)-sensitive

What does this do?
The function $1 - (1 - p^k)^L$

- $k=5$, $L=20$

For example if $(p_1, p_2)$ were $(0.6, 0.4)$ then now they are $(0.802, 0.186)$
A theoretical result on NN
(r,\varepsilon)-neighbor problem

If there is a neighbor \( p \), such that \( d(p,q) \leq r \), return \( p' \), s.t. \( d(p',q) \leq (1+\varepsilon)r \).
(r, \varepsilon)-neighbor problem

- Lets construct a data structure that succeeds with constant probability

- Focus on the hamming distance first
NN using locality sensitive hashing

- Take a \((r_1, r_2, p_1, p_2) = (r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)\) - sensitive family

- If there is a neighbor at distance \(r\) we catch it with probability \(p_1\)
NN using locality sensitive hashing

• Take a \((r_1, r_2, p_1, p_2) = (r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)\) - sensitive family

• If there is a neighbor at distance \(r\) we catch it with probability \(p_1\) so to guarantee catching it we need to “or” \(1/p_1\) functions.
NN using locality sensitive hashing

- Take a \((r_1, r_2, p_1, p_2) = (r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)\) - sensitive family

- If there is a neighbor at distance \(r\) we catch it with probability \(p_1\) so to guarantee catching it we need to "or" \(1/p_1\) functions.

- But we also get false positives, how many?
NN using locality sensitive hashing

• Take a \((r_1, r_2, p_1, p_2) = (r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)\) - sensitive family

• If there is a neighbor at distance \(r\) we catch it with probability \(p_1\) so to guarantee catching it we need to “or” \(1/p_1\) functions..

• But we also get false positives, how many?

\[
n \frac{p_2}{p_1} \approx n \left(1 - \left(1 - p_2 \right)^{\frac{1}{p_1}}\right)
\]
NN using locality sensitive hashing

• Take a \((r_1, r_2, p_1, p_2) = (r, (1+\varepsilon)r, 1-r/m, 1-(1+\varepsilon)r/m)\) - sensitive family

• Make a new function by concatenating ("and") \(k\) of these basic functions

• We get a \((r_1, r_2, (p_1)^k, (p_2)^k)\) sensitive family

• If there is a neighbor at distance \(r\) we catch it with probability \((p_1)^k\) so to guarantee catching it we need \(L=1/(p_1)^k\) functions. We get a \((r_1, r_2, 1-(1-(p_1)^k)^L, 1-(1-(p_2)^k)^L)\) sensitive family

• But we also get false positives in our \(1/(p_1)^k\) buckets, how many? \(n(p_2)^k/(p_1)^k\)
(r,ε)-Neighbor with constant prob

Scan the first $4n(p_2)^k/(p_1)^k$ points in the buckets and return the closest

A close neighbor ($\leq r_1$) is in one of the buckets with probability $\geq 1-(1/e)$

There are $\leq 4n(p_2)^k/(p_1)^k$ false positives with probability $\geq 3/4$

⇒ Both events happen with constant prob.
Analysis

Total query time: \( \frac{k}{(p_1)^k} + n\left(\frac{p_2}{p_1}\right)^k \)

We want to choose \( k \) to minimize this.
Analysis

Total query time: 
(Each op takes time prop. to the dim.)

We want to choose $k$ to minimize this:

$$\frac{k}{(p_1)^k} + n\left(\frac{p_2}{p_1}\right)^k$$

$$k = n\left(\frac{p_2}{p_1}\right)^k \iff \frac{n}{k} = \left(\frac{1}{p_2}\right)^k$$

$$k = \log_{\frac{1}{p_2}}(n) - \Theta(\log \log n)$$
Summary

Total query time: \( \frac{k}{(p_1)^k} + n\left(\frac{p_2}{p_1}\right)^k \)

Put: \( k = \left\lfloor \log_{1/p_2}(n) \right\rfloor \)

\[
t \leq \left(\frac{1}{p_1}\right)^{\left\lfloor \log_{1/p_2}(n) \right\rfloor} \leq \frac{1}{p_1} \left(\frac{1}{p_1}\right)^{\log_{1/p_2}(n)} = \frac{1}{p_1} n^{\frac{\log(1/p_1)}{\log(1/p_2)}} = \frac{1}{p_1} n^\rho
\]

space: \( n \frac{1}{p_1} n^\rho = \frac{1}{p_1} n^{1+\rho} \)
What is $\rho$?

$$\rho = \frac{\log \left( \frac{1}{p_1} \right)}{\log \left( \frac{1}{p_2} \right)} = \frac{\log(p_1)}{\log(p_2)} = \frac{\log \left( 1 - \frac{r}{m} \right)}{\log \left( 1 - \frac{(1+\varepsilon)r}{m} \right)} \approx \frac{1}{1 + \varepsilon}$$
(1+\varepsilon)-approximate NN

• Given \( q \) find \( p \) such that \( \forall p' \neq p \)
  \[ d(q,p) \leq (1+\varepsilon)d(q,p') \]

• We can use our solution to the \((r,\varepsilon)\)-neighbor problem
(1+\epsilon)-approximate NN vs (r,\epsilon)-neighbor problem

- If we know \( r_{\text{min}} \) and \( r_{\text{max}} \) we can find (1+\epsilon)-approximate NN using \( \log\left(\frac{r_{\text{max}}}{r_{\text{min}}}\right) \) (\( r,\epsilon' \approx \epsilon/2 \))-neighbor problems
LSH using $p$-stable distributions

Definition: A distribution $D$ is 2-stable if when $X_1, \ldots, X_d$ are drawn from $D$, 
$\sum \nu_i X_i = \|\nu\|X$ where $X$ is drawn from $D$.

So what do we do with this?

$h(p) = \sum p_i X_i$

$h(p) - h(q) = \sum p_i X_i - \sum q_i X_i = \sum (p_i - q_i) X_i = \|p - q\|X$
LSH using \textit{p-stable} distributions

So what do we do with this?

\[ h(p) = \left\lfloor (pX)/r \right\rfloor \]

Pick \( r \) to maximize \( \rho \)...

\[ r \]
Bibliography

- M. Charikar: Similarity estimation techniques from rounding algorithms. STOC 2002: 380-388