Approximate nearest neighbors: Towards removing the curse of dimensionality

Ioannis Iakovidis
iakoviid.github.io

National and Kapodistrian University of Athens

December 2, 2021

Introduction

Problem: Nearest neighbor (NN)

Given a set P of n distinct points in R^d under some norm $\|\cdot\|$.

Goal

Given some query points $q \in R^d$. Output the point $p \in P$ that minimizes $\|p - q\|$.

- As we will answer a number of queries for set P. It is beneficial to construct a data structure and use it for efficient search given a query.
- So we have trade offs between the query processing time, the space of our data structure and the preprocessing time.

Introduction

- The algorithm and especially the k-Nearest Neighbohrs version has a lot of applications in Machine Learning and Data Processing.
- Typically in those application we have problems where n and d are large (n is in the millions).
- This problem has a naive solution: Compute the distances between all the points and find the minimum.
- The query time for this algorithm is $O(n \cdot d)$. This is not efficient as n is large. Also we have not used any preprocessing.

Algorithms for Low Dimensional Settings

For the case where d=1 all points lie on the axis and so the problem can be efficiently solved with binary search.

space: $O(n)$ query time: $O(logn)$.

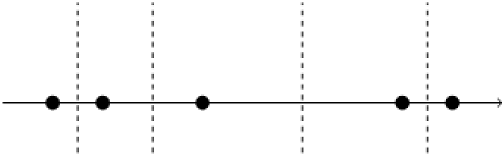


Figure 1: 1D case

Algorithms for Low Dimensional Settings

For the case of d=2 we can have a generalization of this idea using Voronoi Diagrams. And there are multidimensional equivalents of the binary search tree for this problem, k-d tree (k-dimensional tree) and Vantage Point Tree.

Figure 2: Voronoi diagram

Curse Of Dimensionality

- But as d grows the complexity of the space partition is grows exponentially. The Voronoi diagram has size $O(n^{\lceil d/2 \rceil})$
- In fact there is a hardness results that states: An exact algorithm with query time n^{1-b} for some $b > 0$ and $poly(n)$ preprocessing would violate SETH.
- This is the case for many algorithms in data processing. Usually their complexity depends exponentially with the dimension of the data. This problem is called curse of dimensionality.

Approximate Nearest Neighbors

For obtaining better guaranties we will relax the problem to output an approximate solution.

Problem: Approximate Nearest neighbor (ANN)

Given a set P of n distinct points in R^d under some norm $\|\cdot\|$. Construct a data structure that given a point q, returns a point $p \in P$ such that $d(p, q) \leq c \cdot \min_{p' \in P} d(p', q)$

Point Location in Equal Balls

Problem: Point Location in Equal Balls (PLEB)

Given a set P of n distinct points in R^d under some norm $\|\cdot\|$.
Construct a data structure that given a point q:

- if there is a $p \in P$ such that $d(p, q) \leq r_1$, return **YES** and any point $p' \in P$ with $d(p', q) \leq r_2$.
- if there is no point $p \in P$ such that $d(p, q) \leq r_2$, return **NO**.

Reduction from ANN to PLEB

Theorem: Reduction from ANN to PLEB

If for every r there is a data structure with space and time bound S and T, that solves $(r, (1+\epsilon)r)$-PLEB. Then there is an algorithm for $(1+\epsilon)^2$-ANN with space bound $O(S \cdot \log_{1+\epsilon} \frac{D_{max}}{D_{min}})$ and query time $O(T \cdot \log\log_{1+\epsilon} \frac{D_{max}}{D_{min}})$, where D_{max} and D_{min} the largest and smallest interpoint distances.

There are more efficient reductions use $(r, (1+\epsilon r))$-PLEB and solve $(1+\epsilon)$-ANN. But outside of the scope of this presentation.

Reduction from ANN to PLEB

Proof.

Assuming that we are looking for the nearest point p of q in a set P. Let the sequence $R = \{\frac{D_{min}}{2}, (1+\epsilon)\frac{D_{min}}{2}, \ldots, (1+\epsilon)^k \frac{D_{min}}{2}\}$ where $k \in \mathrm{N}$ such that $(1+\epsilon)^k \frac{D_{min}}{2} \geq D_{max} \implies k \geq \log_{1+\epsilon} \frac{2D_{max}}{D_{min}}$.

Reduction from ANN to PLEB

Proof.

Assuming that we are looking for the nearest point p of q in a set P. Let the sequence $R = \{\frac{D_{min}}{2}, (1+\epsilon)\frac{D_{min}}{2}, \ldots, (1+\epsilon)^k\frac{D_{min}}{2}\}$ where $k \in \mathrm{N}$ such that $(1+\epsilon)^k\frac{D_{min}}{2} \geq D_{max} \implies k \geq \log_{1+\epsilon}\frac{2D_{max}}{D_{min}}$.
Find r^* the $\min\{r \in R : PLEB(r, (1+\epsilon)r) = \text{YES}\}$.
Because $PLEB(r^*, (1+\epsilon)r^*) = \text{YES}$ we know that $d(p', q) \leq (1+\epsilon)r^*$ also because $PLEB(r^*/(1+\epsilon), r^*) = \text{NO}$ $d(p, q) \geq r^*/(1+\epsilon)$.
So the output point of $PLEB(r^*, (1+\epsilon)r^*)$ p' is a $(1+\epsilon)^2$-approximation for p.

Reduction from ANN to PLEB

Proof.

Assuming that we are looking for the nearest point p of q in a set P. Let the sequence $R = \{\frac{D_{min}}{2}, (1+\epsilon)\frac{D_{min}}{2}, \ldots, (1+\epsilon)^k \frac{D_{min}}{2}\}$ where $k \in \mathrm{N}$ such that $(1+\epsilon)^k \frac{D_{min}}{2} \geq D_{max} \implies k \geq \log_{1+\epsilon} \frac{2D_{max}}{D_{min}}$.
Find r^* the $\min\{r \in R : PLEB(r, (1+\epsilon)r) = \text{YES}\}$.
Because $PLEB(r^*, (1+\epsilon)r^*) = \text{YES}$ we know that $d(p', q) \leq (1+\epsilon)r^*$ also because $PLEB(r^*/(1+\epsilon), r^*) = \text{NO}$ $d(p, q) \geq r^*/(1+\epsilon)$.
So the output point of $PLEB(r^*, (1+\epsilon)r^*)$ p' is a $(1+\epsilon)^2$-approximation for p.Finding r^* can be done with binary search with $\log k$ PLEB calls so the query time is $O(T \log k) = O(T \cdot \log \log_{1+\epsilon} \frac{D_{max}}{D_{min}})$. While data structures are created for every possible PLEB call in preprocessing so we use space $O(sk) = O(S \cdot \log_{1+\epsilon} \frac{D_{max}}{D_{min}})$. □

Locality Sensitive Hashing

Definition: Locality Sensitive Hash (LSH)

A hash family $H = \{h : U \to S\}$ is called (r_1, r_2, p_1, p_2)-locally sensitive if for all points $p, p' \in U$,

- if $d(p, p') \leq r_1$, then $Pr[h(p) = h(p')] \geq p_1$
- if $d(p, p') \geq r_2$, then $Pr[h(p) = h(p')] \leq p_2$

Example: Let $U = \{0, 1\}^d$ and our notion of distance be the hamming distance, $d(p, p') = |\{i : p(i) \neq p'(i)\}|$. Then $H = \{h_i : h_i(p) = p(i), i \in [d]\}$ is $(r, cr, 1 - \frac{r}{d}, 1 - \frac{cr}{d})$ locality sensitive.

Solving PLEB using LSH

Theorem: Locality Sensitive Hash (LSH) to PLEB

Suppose that there is some (r_1, r_2, p_1, p_2)-LSH $H = \{h : U \to S\}$. Then there is an algorithm for (r_1, r_2)-PLEB which uses:

- $O(dn + n^{1+\rho})$ space
- $O(n^\rho)$ query time, measured in hash evaluations

where $\rho = \frac{\ln 1/p_1}{\ln 1/p_2}$. This algorithm succeds with constant probability.

Solving PLEB using LSH

Algorithm: Locality Sensitive Hash (LSH) to PLEB

Let k and l parameters. From H we define an other function family $G = \{g : U \to S^k | g(p) = (h_1, \ldots, h_k), h_i \in H), \forall i \in [k]\}$.
Preprocessing: 1) Choose l hash functions from G $g_1, \ldots, g_l$ independently uniformly at random.
2) Store all $p \in P$ to buckets $g_1(p), \ldots, g_l(p)$ retaining only non empty buckets.

Solving PLEB using LSH

Algorithm: Locality Sensitive Hash (LSH) to PLEB

Let k and l parameters. From H we define an other function family $G = \{g : U \to S^k | g(p) = (h_1, \ldots, h_k), h_i \in H), \forall i \in [k]\}$.
Preprocessing: 1) Choose l hash functions from G $g_1, \ldots, g_l$ independently uniformly at random.
2) Store all $p \in P$ to buckets $g_1(p), \ldots, g_l(p)$ retaining only non empty buckets.
For query q: 1) Search through the buckets $g_1(q), \ldots, g_l(q)$ and stop after the first $2l$ points.
2) If any of these points p has $d(p, q) \leq r_2$, return p with YES, otherwise, return NO.

Solving PLEB using LSH

Proof: Locality Sensitive Hash (LSH) to PLEB

We will show that the following hold with constant probability:
1) If there exists $p \in P : d(p, q) \leq r_1$ then there exists a $j \in [l] : g_j(p) = g_j(q)$.
2) There are at most $2l - 1$ points $p \in P : d(p, q) \geq r_2$ and there exists a $j \in [l] : g_j(p) = g_j(q)$.

Solving PLEB using LSH

Proof: Locality Sensitive Hash (LSH) to PLEB

We will show that the following hold with constant probability:
1) If there exists $p \in P : d(p, q) \leq r_1$ then there exists a $j \in [l] : g_j(p) = g_j(q)$.
2) There are at most $2l - 1$ points $p \in P : d(p, q) \geq r_2$ and there exists a $j \in [l] : g_j(p) = g_j(q)$.
The expected number of points satisfying (2) is $l \cdot n \cdot {p_2}^k = l$ if k is set to $\log_{1/p_2} n$. So by Markov $Pr[|\{p : \text{satify } (2)\}| \geq 2l] \leq \frac{1}{2}$.
Therefore the $Pr[(2) \text{ holds}] > \frac{1}{2}$.

Solving PLEB using LSH

Proof: Locality Sensitive Hash (LSH) to PLEB

We will show that the following hold with constant probability:
1) If there exists $p \in P : d(p, q) \leq r_1$ then there exists a $j \in [l] : g_j(p) = g_j(q)$.
2) There are at most $2l - 1$ points $p \in P : d(p, q) \geq r_2$ and there exists a $j \in [l] : g_j(p) = g_j(q)$.
The expected number of points satisfying (2) is $l \cdot n \cdot {p_2}^k = l$ if k is set to $\log_{1/p_2} n$. So by Markov $Pr[|\{p : \text{satify } (2)\}| \geq 2l] \leq \frac{1}{2}$.
Therefore the $Pr[(2) \text{ holds}] > \frac{1}{2}$.
Let $p : d(p, q) \leq r_1$ then $Pr(g_j(p) = g_j(q)) = {p_1}^k = n^{-\rho}$.
So $Pr[(1) \text{ hold}] = 1 - (1 - n^{-\rho})^l = 1 - (1 - n^{-\rho})^{n^\rho} \geq 1 - \frac{1}{e} > \frac{1}{2}$.
If we set l to n^ρ.

LSH for l_2

Locality Sensitive Hash family for l_2

A (r_1, r_2, p_1, p_2)-Locality Sensitive hash family for l_2 is the following $H = \{h_{g,a}(x) = \lfloor \frac{gx+a}{w} \rfloor\}$. Where g a vector of iid normal random variables ($g_i \sim N(0,1)$) and $a \sim [0, w]$. We can show that the collision probability given $\|p - q\| < s$ is:

$$p(s) = \int_0^w \frac{1}{s} f(\frac{t}{s})(1 - \frac{t}{w})dt$$

LSH for l_2

Proof.

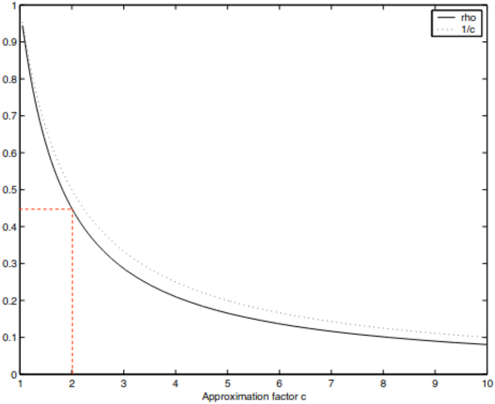
Given x, q let's calculate the probability of collision: $Pr[h_{g,a}(x) = h_{g,a}(q)]$ let $s = \|x - q\|$. We get a collision if $|gx - gq| < w$ and a divider does not fall between gx and gq.
But because g is a vector of iid normal r.v. we have that $|g(x - q)| = sZ, Z \sim N(0, 1)$.
Also that a divider falls beteen gx and gq is their interval length divided by w.
Thus the probability of collision is:

$$p(s) = \int_0^{w/s} f(z)(1 - \frac{zs}{w})dz = \int_0^{w} \frac{1}{s} f(\frac{t}{s})(1 - \frac{t}{w})dt, t = zs$$

For fixed parameter w the probability monotonically decreases with $s = \|x - q\|$. w is typically set to r_1. □

LSH for l_2

Recall that for $p_1 = p(r)$ and $p_2 = p(cr)$ we are interested in $\rho = \frac{\ln 1/p_1}{\ln 1/p_2}$ it is true that in this case $\rho < 1/c$.



Johnson–Lindenstrauss lemma

Theorem: Johnson–Lindenstrauss lemma

Given $0 < \epsilon < 1$, a set X of m points in $\mathbb{R}^N$, and a number $b > 8\ln(m)/\epsilon^2$, there is a linear map $f : \mathbb{R} \to \mathbb{R}^n$ such that:

$$(1-\epsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 < (1+\epsilon)\|u-v\|^2$$

for all $u, v \in X$.

Thank you for your attention.
Are there any questions?

References

References used for this talk:

1. Approximate nearest neighbors: Towards removing the curse of dimensionality, Piotr Indyk and Rajeev Motwani.
2. Similarity Search in High Dimensions via Hashing, Aristides Gionis Piotr Indyk and Rajeev Motwani.
3. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions Mayur Datar, Nicole Immorlica, Piotr Indyk and Vahab S. Mirrokni