Approximate Near Neighbor Search*

Aaron Geelon So

December 10, 2018

Given a set P of n data points in a metric space (X, D), your task is to build a data structure $\mathcal{A}$ that enables you to return the nearest neighbor in P for any query point $q \in X$. There are two naive solutions:

1. compute the distance $\mathsf{D}(p, q)$ for all $p \in P$ on the fly, or

2. precompute $\arg\min_{p \in P} \mathsf{D}(p, q)$ for all $q \in X$.

The former solution requires $O(n)$ space to hold P and $\Omega(n)$ query time. The latter requires $\Omega(|X|)$ space but $O(1)$ query time. These are two extremes of the *space-time tradeoff* in data structures. Often, giving up a little bit on one (space or time) can yield substantial improvement on the other. Further improvements may be made to both if we allow for an *approximate* solution and/or a *probabalistic* algorithm.

Today, we will discuss some methods and bounds on *approximate near neighbor* (ANN) *search*. These techniques are useful in tasks such as optical character recognition, searching for similar images (VisualRank), entity resolution, detecting (near) duplicates in a dataset, clustering, exploratory data analysis, and so on.

In the ANN problem, we are satisfied with any near neighbor, specifically, any point within some distance r away from the query point. We can call $r > 0$ the *scale parameter*. Furthermore, we allow for any approximate near neighbor: any point within cr from the query point. We can call $c > 1$ the *approximation factor*.

Definition 1 ((c, r, δ)-Approximate Near Neighbor)**.** *Let P be a set of n points in a metric space (X, D). Let $q \in X$ be a query point such that q is within a distance r from some $p \in P$. A* (c, r, δ)-approximate near neighbor (ANN) data structure $\mathcal{A}$ *is one that takes a query point q and with probability $1 - \delta$, returns a point $p \in P$ within cr of q.*

Note that if q is not near P, the behavior of $\mathcal{A}$ is undefined. Also, if $\delta = 0$, we omit δ and say that $\mathcal{A}$ is a (c, r)-ANN data structure.

1 Data-independent approach: dimensionality reduction

Let's consider $X = (\mathbb{R}^k, \ell_2)$. We can build a very simple $(1 + \epsilon, r)$-ANN data structure as follows: discretize $\mathbb{R}^k$ into cubes of side-lengths $\epsilon r/\sqrt{k}$. If a cube intersects with $B(p, r)$ for some $p \in P$, set the cube to be a key to p in a hash table. Now, given any query point q, we can just compute the cube that contains q, and check if the cube is saved in the dictionary.

Notice that because the diameter of these cubes are ϵr, we obtain a $1 + \epsilon$ approximation factor. Overall, each ball $B(p, r)$ is covered by $(1/\epsilon)^{\Omega(k)}$ cubes; the size of the hash table is at least $n \cdot (1/\epsilon)^{\Omega(k)}$. Unfortunately, this is exponential in dimension. But we can leverage the Johnson-Lindenstrauss lemma to make the number of entries in our hash table dimension-independent.

*This lecture follows [A+2018] pretty closely; I make almost no contribution in terms of pedagogy. See [A+2018] for a strictly better reference.

Recall that JL is a concentration bound on how much the length of a vector in $\mathbb{R}^d$ changes when projected to a random k-dimensional subspace:

Lemma 2 (Johnson-Lindenstrauss)**.** *Fix $d \geq 1$ and $k < d$. Let $A : \mathbb{R}^d \to \mathbb{R}^k$ be the projection of $\mathbb{R}^d$ onto a k-dimensional subspace, chosen uniformly at random. Let $f : \mathbb{R}^d \to \mathbb{R}^k$ be $f(x) = \frac{\sqrt{d}}{\sqrt{k}} Ax$. Then, there is a universal constant C such that for any $\epsilon \in (0, 1/2)$ and any $x, y \in \mathbb{R}^d$,*

$$\Pr_A \left[1 - \epsilon < \frac{\|f(x) - f(y)\|}{\|x - y\|} < 1 + \epsilon \right] \geq 1 - \exp\left(-C\epsilon^2 k\right).$$

We would like to approximately preserve $O(n)$ pairs of distances (between q and each $p \in P$). It follows that with probability $1 - \delta$, a random projection from $\mathbb{R}^d$ to a subspace $\mathbb{R}^k$ where $k = O_\delta\left(\frac{\log n}{\epsilon^2}\right)$ suffices (omitting a $\log \frac{1}{\delta}$ term). It follows that:

Theorem 3. *Fix $\epsilon \in (0, 1/2)$ and $d \geq 1$. There is a $(1+O(\epsilon), r, \delta)$-ANN data structure over $(\mathbb{R}^d, \ell_2)$ achieving $Q = O(d \cdot \frac{\log n}{\epsilon})$ query time and $S = n^{O(\log(1/\epsilon)/\epsilon^2)} + O(d(n + k))$ space. The time needed to build the data structure is $O(S + ndk)$.*

Still, this technique to achieve ANN through dimensionality reduction requires polynomial space in n. The following technique, *locality-sensitive hashing* (LSH), will decrease the space, although with increased query time n^ρ, where $\rho \in (0, 1)$.

2 Data-independent approach: locality-sensitive hashing

As a high-level overview of the previous approach, we built a dictionary on $\mathbb{R}^k$ so that any q near point $p \in P$ get mapped to an approximately near neighbor. But because we needed on the order of $(1/\epsilon)^k$ cubes to cover each r-ball, our hash takes up a lot of space.

To improve this, we'll generalize this notion of 'mapping near points of p to p' and additionally introducing some randomness. We now define a LSH family, a collection of functions such that with high probability, a function drawn from the family will hash near points together while splitting apart far points:

Definition 4 (Locality-Sensitive Hashing (LSH) Family)**.** *Let (X, D) be a metric space. Let the scale parameter and approximation factor be $r > 0$ and $c > 1$. Let U a set. A distribution $\mathcal{H}$ over maps $h : X \to U$ is called (r, cr, p_1, p_2)-*sensitive *if for all $x, y \in X$,*

- *if $\mathsf{D}(x, y) \leq r$, then $\Pr_{h\sim\mathcal{H}}[h(x) = h(y)] \geq p_1$,*
- *if $\mathsf{D}(x, y) > cr$, then $\Pr_{h\sim\mathcal{H}}[h(x) = h(y)] \leq p_2$.*

The distribution $\mathcal{H}$ is called an LSH family*, and it has* quality $\rho = \frac{\log 1/p_1}{\log 1/p_2}$.

In order for the LSH family to be useful, we need $p_1 > p_2$. This implies $\rho < 1$. Also, notice that given a LSH family of (r, cr, p_1, p_2)-sensitive maps $h : X \to U$, we can construct a new LSH family of (r, cr, p_1^k, p_2^k)-sensitive maps $h : X \to U^k$, by just picking k i.i.d. hash functions $h_1, \ldots, h_k$, and defining:

$$g(x) = (h_1(x), \ldots, h_k(x)).$$

Note that the quality of this new hash family does not change.

Suppose that we had access to an LSH family. Then, we can apply a hash $h : X \to U$ onto all of P. Given a query $q \in X$ within r of some p^*, we just need check the distances between q and p for $p \in h^{-1}(P)$. Because the probability that a point cr away from q is hashed to $h(q)$, the expected number of points we need to check is np_2. However, because it's possible that p^* is the only near neighbor, h might not hash p^* into $h(q)$. In fact, the probability of success is then just p_1.

But we can boost the probability of success to be arbitrarily close to 1 by repeated trials; it is not too hard to derive from a tail concentration bound that with constant probability, it suffices to perform $L = O(1/p_1)$ trials. Because of our note that we can obtain a new LSH family by combining k i.i.d. hashes to get new probabilities p_1^k and p_2^k, we have more generally:

$$L = O\left(\frac{1}{p_1^k}\right) \text{ hash tables}$$

where each hash table stores n entries. Furthermore, if it takes τ time to compute $h(\cdot)$ and $O(\tau)$ time to compute the distance between two points, the expected query time is:

$$Q = O\left(L \cdot \left(k\tau + n \cdot p_2^k \cdot \tau\right)\right).$$

The value of k that minimizes Q is $k = \lceil \log_{1/p_2} n \rceil \leq \log_{1/p_2} n + 1$. This implies that $L = O(n^\rho / p_1)$. It follows that space is at least:

$$S = O\left(\frac{n^{\rho+1}}{p_1}\right).$$

Formally, we have:

Theorem 5. *Let (X, D) be a metric space, scale $r > 0$, approximation factor $c > 1$. Suppose the metric admits a (r, cr, p_1, p_2)-sensitive LSH family $\mathcal{H}$, where the map $h(\cdot)$ can be stored in σ space, and for any x, $h(x)$ can be computed in τ time. Suppose that computing distances takes $O(\tau)$ time.*

There exists a (c, r, δ)-ANN data structure over X achieving query time Q and space S:

$$Q = O\left(n^\rho \cdot \tau \frac{\log_{1/p_2} n}{p_1}\right) \qquad S = O\left(\frac{n^{1+\rho}}{p_1} + \frac{n^\rho}{p_1} \cdot \sigma \log_{1/p_2} n\right).$$

It takes $O(S \cdot \tau)$ time to build this data structure.

Example 6 (Hamming space)**.** *Let $X = \{0,1\}^d$ be the Boolean hypercube with the ℓ_1-distance. Let $\mathcal{H}$ be the LSH family of projections onto a random coordinate i, $\mathcal{H} = \{h_i : h_i(x) = x_i\}$.*

If two points are within r of each other, this means that all but r of their coordinates coincide. Thus, the probability that they get sent to the same bucket is $1 - r/d$. Similarly, if two points are at least cr from each other, they coincide on at most $d - cr$ points, so the probability they collide is $1 - cr/d$. It follows that $\mathcal{H}$ is $(r, cr, 1 - r/d, 1 - cr/d)$-sensitive and $\rho \leq 1/c$.

Example 7 (Euclidean space)**.** *On $(\mathbb{R}^d, \ell_2)$, it is possible to obtain LSH quality $\rho = 1/c^2 + \frac{O(\log\log n)}{(\log n)^{1/3}}$. The high level picture is to perform* ball carving, *where we cover the whole space with a sequence of balls of radius wr, for some parameter $w > 1$. The hash of q returns the index of the first ball that contains q. Of course, the space $\mathbb{R}^d$ is not compact, so it seems that we can't cover the whole space in finite time. To get around this, let $s \in [0, 4w]^d$ be a vector chosen uniformly at random. Then, we cover $\mathbb{R}^d/(s + \mathbb{Z}^d)$. This can be further improved by performing dimensionality reduction with JL first.*

It turns out that these two examples are near-optimal:

Theorem 8. *Fix dimension $d \geq 1$ and approximation factor $c \geq 1$. Let $\mathcal{H}$ be a (r, cr, p_1, p_2)-sensitive LSH family over the Hamming space, and suppose $p_2 \geq 2^{-o(d)}$. Then, $\rho \geq 1/c - o_d(1)$.*

And because $\|x - y\|_1 = \|x - y\|_2^2$ for Boolean vectors, this implies a lower bound $\rho \geq 1/c^2 - o(1)$ for Euclidean space.

3 Data-dependent approach

So far, we've looked only at data-independent approaches, i.e. the hashes were oblivious to the data P. It turns out to be possible to improve on the bounds by taking the data into account.

Theorem 9. *For every $c > 1$, there exists a data structure for (c, r, δ)-ANN over $(\mathbb{R}^d, \ell_2)$ with space $n^{1+\rho} + O(nd)$ and query time $n^\rho + dn^{o(1)}$, where:*

$$\rho \leq \frac{1}{2c^2 - 1} + o(1).$$

So, for $c = 2$, data-dependence improves query time from $n^{1/4+o(1)}$ to $n^{1/7+o(1)}$ while using less memory. To do this, we'll first look at data-independent LSH on the sphere.

3.1 Data-independent LSH for a sphere

On the unit sphere (S^{d-1}, ℓ_2), we can somewhat extend the ball-carving idea. What we'll do is take a sequence of random directions and carve out a half-space that's lifted off the origin by some parameter η. More formally, consider a sequence of i.i.d. Gaussians $g_1, g_2, \ldots, \sim \mathcal{N}(0, I_{d\times d})$. The hash maps a point $x \in S^{d-1}$ to:

$$h(x) = \min_t \{t \geq 1 : \langle x, g_t \rangle \geq \eta\}.$$

It turns out that this LSH family yields:

$$\rho = \frac{4 - c^2 r^2}{4 - r^2} \cdot \frac{1}{c^2} + \delta(r, c, \eta),$$

where $\delta(r, c, \eta) > 0$ and $\delta(r, c, \eta) \to 0$ as $\eta \to \infty$. There are three main regimes to consider here:

- $r = o(1)$, so $\rho = \frac{1}{c^2} + o(1)$
- $r \approx 2/c$, so ρ is close to 0; notice that any point serve as an answer to any valid query
- $r \approx \frac{\sqrt{2}}{c}$, where $\rho \approx \frac{1}{2c^2-1}$.

Notice that on the sphere, the distance between two random points is $\sqrt{2}$ with high probability. Thus, in the last regime, we are essentially asking for a neighbor that is slightly closer than the 'typical' point.

3.2 Data-dependent LSH for a sphere

The main idea for data-dependent LSH is to recursively remove *dense low-diameter clusters*. In particular, iteratively find points $u_t \in S^{d-1}$ such that the ball $B(u, \sqrt{2} - \epsilon) \cap P_{t-1}$ contains at least a τ-fraction of P,

$$|B(u, \sqrt{2} - \epsilon) \cap P_{t-1}| \geq \tau n,$$

where $P_t := P_t \setminus B(u, \sqrt{2} - \epsilon)$ and $P_0 := P$. Continue carving out S^{d-1} until there are no dense clusters left.

For now, suppose that our query q is close to near some point in P_T, with scale $r < \sqrt{2}/c$. We can now perform the data-independent LSH on this sphere for P_T; there are at most τn points within $\sqrt{2} - \epsilon$ of the query, and so the hash will have in expectation at most $(\tau + p_2)n$ points.

References

[A+2018] Andoni, A., Indyk, P., Razenshteyn, I. *"Approximate nearest neighbor search in high dimensions."* arXiv preprint arXiv:1806.09823 (2018).