Approximate Nearest Neighbor Search

Edo Liberty
Algorithms in Data mining

1 Recap

In the last class we discussed possible solutions for the nearest neighbor problem in low dimensions. The problem is defined below.

Definition 1.1. Nearest Neighbor Search: *Given a set of points* $\{x_1, \ldots, x_n\} \in \mathbb{R}^d$ *preprocess them into a data structure* X *of size* $\text{poly}(n, d)$ *in time* $\text{poly}(n, d)$ *such that nearest neighbor queries can be performed in logarithmic time. In other words, given a search point* q *a radius* r *and* X *one can return an* x_i *such the* $||q - x_i|| \le r$ *or nothing if no such point exists. The search for* x_i *should require at most* $\text{poly}(d, \log(n))$ *time.*

As we have seen, this problem is hard when the dimension d is large. It turns out that if we relax the problem a little bit, we can take a different tack at it.

Definition 1.2. Approximate Nearest Neighbor Search: *Given a set of points* $\{x_1, \ldots, x_n\} \in \mathbb{R}^d$ *preprocess them into a data structure* X *of size* $\text{poly}(n, d)$ *in time* $\text{poly}(n, d)$ *such that* ε*-close nearest neighbor queries can be performed in logarithmic time. In other words, given a search point* q *a radius* r *a constant* ε *and* X *one can do one of two things. If there exists a point* x_i *such that* $||q - x_i|| \le r$ *then return possibly another point* x_j *for which* $||q - x_j|| \le r(1 + \varepsilon)$*. If for all* i*,* $||q - x_i|| > r(1 + \varepsilon)$*, return no point. The search should require* $\text{poly}(\log(n), d)$ *time.*

2 Local Sensitive Hashing

In this section we will review ideas from [1] and [2]. We define a family $\mathcal{H}$ of functions as (r_1, r_2, p_1, p_2)-sensitive if:

$$\begin{aligned} ||x - y|| < r_1 \quad &\rightarrow \quad \Pr_{h \sim \mathcal{H}}(h(x) = h(y)) > p_1 \\ ||x - y|| > r_2 \quad &\rightarrow \quad \Pr_{h \sim \mathcal{H}}(h(x) = h(y)) < p_2 \end{aligned}$$

This is only meaningful if $r_1 < r_2$ and $p_1 > p_2$. Which means that if x and y are "close" then the probability that they hash to the same value is at least something, but if they are further away then it is smaller. Or, the probability of points being hashed to the same value decreases with their distance.

Let us assume such functions exist and give some intuition on how to use them. First we concatenate k different hash functions from $\mathcal{H}$ to construct a new hash function $g(x) = [h_1(x), \ldots, h_k(x)]$. We choose k such that $\Pr(g(x) = g(y)) \leq 1/n$ if $||x - y|| > r_2$. Using the (r_1, r_2, p_1, p_2)-sensitivity of $\mathcal{H}$ we will get that if $||x - y|| < r_1$ then $\Pr(g(x) = g(y)) \geq 1/n^\rho$ for some $\rho < 1$.

Now, if we generate $\ell = n^\rho$ different copies of g, $g_1, \ldots, g_\ell$, and consider every x in the data for which $g_i(x) = g_i(q)$ we will find every close point x with constant probability and consider only $O(n^\rho)$ far points.

Let us make this statement more precise. The preprocessing step is so.

```
1. $\rho \leftarrow \log(1/p_1)/\log(1/p_2)$
2. $\ell \leftarrow n^\rho$
3. $k \leftarrow \log(n)/log(1/p_2)$
4. for $\ell' \in \{1, \ldots, \ell\}$
5.     $g_{\ell'} \leftarrow [h_1(x), \ldots, h_k(x)]$
6. for $x \in X$
7.     for $\ell' \in \{1, \ldots, \ell\}$
8.         add $x$ to $T_{\ell'}(g_{\ell'}(x))$
```

The search stage is as follows:

```
1. $S \leftarrow \emptyset$
2. for $\ell' \in \{1, \ldots, \ell\}$
3.     add $T_{\ell'}(g_{\ell'}(x)))$ to $S$
4. if $|S| \leq 2n^\rho$
5.     for $x' \in S$
6.         if $||x' - q|| \leq r_2$
7.             return $x'$
```

Fact 2.1. *the number of points x such that $||x - q|| \geq r_2$ and $x \in S$ is smaller that $2 \cdot n^\rho$ with probability at least $1/2$.*

Proof. $x \in S$ is for some ℓ' we have $g_{\ell'}(q) = g_{\ell'}(x)$ for x such that $||x - q|| > r_2$ this happens with probability $p_2^{log(n)/log(1/p_2)} = 1/n$. Thus, the expected total number of such points x is 1. Since we have $\ell = n^\rho$ different g functions the total expected number of such points is n^ρ. Due to the above and Markov's inequality $\Pr[|S| > 2n^\rho] \leq \Pr[|S| > 2E[|S|]] \leq 1/2$. □

Fact 2.2. *If $||x - q|| \leq r_1$ then with constant probability $x \in S$*

Proof. By the (r_1, r_2, p_1, p_2)-sensitivity of H

$$\Pr[g(x) = g(q)] \geq p_1^k = p_1^{\log(n)/\log(1/p_2)} = n^{-\log(1/p1)/\log(1/p_2)} = n^{-\rho}$$

Since we repeat this $\ell = n^\rho$ times independently, we have that $g_{\ell'}(x) \neq g_{\ell'}(q)$ for all ℓ' with probability at most $(1 - n^{-\rho})^{n^\rho} < e^{-1}$ □

Thus, both events happen with probability at least $1 - 1/2 - e^{-1} = const$. We can duplicate the entire data structure $O(\log(1/\delta))$ time to achieve success probability $1 - \delta$ in the cost of an $O(\log(1/\delta))$ factor in data storage and search time. This means that the searching running time is $O(dn^\rho)$.

3 LSH functions

3.1 $\{0,1\}^d$ with the Hamming distance

The humming distance between points are $x, y \in \{0,1\}^d$ is defined as the number of coordinates for which x and y defer. We claim that choosing a random coordinate from each vector is a local sensitive function and examine its parameters.

Fact 3.1. *let $\mathcal{H}$ be a family of d functions for which $h_i(x) = x_i$. Then, $\mathcal{H}$ is $(r, (1+\varepsilon)r, 1 - \frac{r}{d}, 1 - \frac{(1+\varepsilon)r}{d})$-sensitive.*

Fact 3.2. *If $r \leq d/\log(n)$ then $\rho = \log(1/p_1)/\log(1/p_2) \leq 1/(1+\varepsilon)$*

Proof. See Fact 3 in [2]. Moreover, assuming $r \leq d/\log(n)$ is harmless since we can always extend each vector by $d\log(n)$ zeros which does not change their distances and guaranties that $r \leq d/\log(n)$. □

Remark 3.1. *This results is also applicable to the Euclidian distance setting because it is possible to map ℓ_2^d into $\ell_1^{O(d)}$ and also trivially possible to map $\ell_1^d = \{0,1\}^{O(d/\varepsilon)}$ with distortion ε for bounded valued vectors.*

Thus, the running time of $O(n^\rho)$ is in fact $O(n^{1/(1+\varepsilon)})$. In other words, to find a the closest neighbor up to a factor of 2 in this distance is possible while examining only $O(\sqrt{n})$ data points. This, however, does not achieve the bound of $O(\text{poly}(d, \log(n)))$.

3.2 Searching with similarities

Note that in the above we never used the fact that the distance function is a metric. Indeed, it is possible to search though items as long as we can produce a local sensitive hashing. In [1] Charikar defined Local sensitive hashing as:

$$\Pr_h[h(x) = h(y)] = sim(x, y)$$

For example, let x and y be sets of items. Their set similarity can be defined as $\frac{|x \cap y|}{|x \cup y|}$. Here we can use a famous trick. We will map $h(x) \to \arg\min_{x_i \in x} g(x_i)$ when g is a random permutation over the entire universe or a random function into $[0, 1]$ for example. The reason this holds true is because the minimal value of g in $|x \cup y|$ might accidentally be also in $|x \cap y|$ but since the distribution is uniform, the probability of this event is $\frac{|x \cap y|}{|x \cup y|}$.

3.3 LSH for points in $\mathbb{S}^{d-1}$

The set of unit length vectors in $\mathbb{R}^d$ is called the d dimensional unit sphere and is denoted by $\mathbb{S}^{d-1}$ (the power is $d-1$ to denote that it is actually a $d-1$ dimensional manifold. Do not be confused, the points are still in $\mathbb{R}^d$) For these points, we can define the distance as the angle between the vectors $d(x, y) = cos^{-1}(x^T y)$. We can thus define a hash function $h(x) = sign(u^T x)$ for a vector u chosen uniformly at random from $\mathbb{S}^{d-1}$. It is immediate to show that h is local sensitive to the angular distance with parameters similar to the previous subsection.

References

[1] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.

[2] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.