COMS 4995-8: Advanced Algorithms (Spring'21) Feb 4, 2021

Lecture 8: Nearest Neighbor Search

Instructor: *Alex Andoni* Scribes: *Sixuan Wang, Grisha Temchenko*

1 Reminder: Dimension reduction

Last time we talked about dimension reduction, in particular we stated a foundational lemma called Johnson–Lindenstrauss lemma. What it shows is that there exists a distribution over ϕ, which maps d-dimensional euclidean space into k-dimensional euclidean space, where we usually think of k is being much smaller than d.The property is that for any x and y, we have:

$$Pr[\|\phi(x) - \phi(y)\|] = (1 \pm \epsilon)\|x - y\| \geq 1 - e^{-\frac{\epsilon^2 k}{9}}$$

Importantly, there is no dependency on the original dimension here. When we care about some end points, for example, if we take $k = D(\frac{\log n}{\epsilon^2})$, then we have:

$$Pr[\|\phi(x) - \phi(y)\|] \geq 1 - \frac{1}{n^3}$$

As a result, we can embed endpoints in $\mathbb{R}^d$ into $\mathbb{R}^k$, preserving the distances up to $1 \pm \epsilon$, where k is only $O(\frac{\log(n)}{\epsilon^2})$.

2 Nearest Neighbor Search

Definition 1. *Giving D, which is a data set of points, $D \subset \mathbb{R}^d$, $\mid D \mid = n, q \subset \mathbb{R}^d$, so that we can find the closest $p^* \subset D$. **Note:**"Closest" p here means*

$$argmin_{p \in D}\|q - p\|$$

Motivation: Look at high dimension key when d is high, so it's the high dimensional case. To illustrate, say we are in domain of images and there are two images which are of 20 * 20 pixels. How do we compare these two images?
One way to declare whether these two images are similar or not is by mapping them into a vector, let's say a 400 dimensional vector where each coordinate corresponds to some pixel in the image.

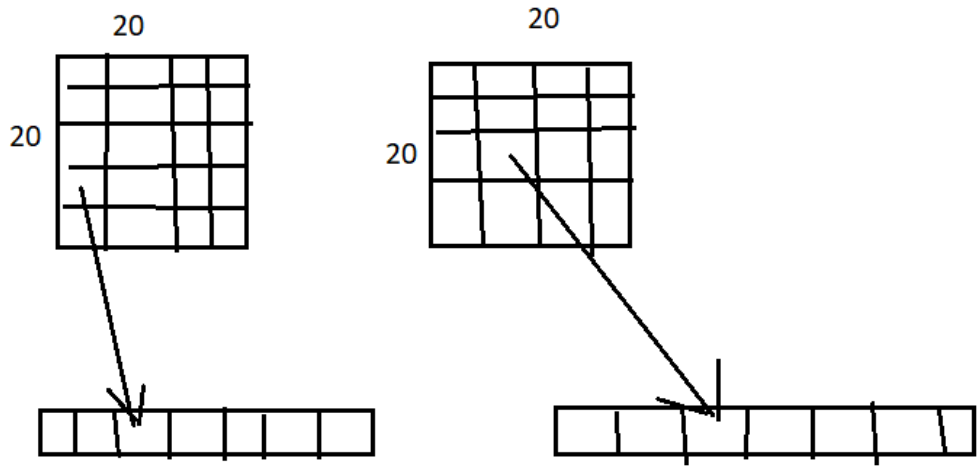

Figure 1: The visualization of how two images mapping into two vectors.

It's reasonable to say that the distance between these two vectors represents some dissimilarity measure between images. The smaller the distance, the more similar the images are. It's the fundamental method of representation and there are more fancy ways to do a representation of images. One popular way today is that we'll take an image fit into a big black box, which is known as neural network, it gives a representation which is a vector whose dimension is the number of neurons we have in penultimate level.

What we are thinking here is not about how to do the representation, we are more from the perspective that if we are done the representation already, how do we solve the problem of finding the nearest neighbor quickly.

Performance measure of NNS:
- Space
- Query time

Solution 0: just iterate over all points and calculate the distance. Pick the point with minimal distance.
- Space: $O(n \cdot d)$
- Query time: $O(n \cdot d)$

The space complexity here is good though we do want to improve query time.

Solution 1: d = 1
Algorithm: Sort all points in D and at the query q, we do binary search.
- Space: $O(n)$
- Query time: $O(\log n)$

2.1 Voronoi diagram

Solution 2: d = 2
Voronoi diagram:
We can try to partition the entire space according to what is the right answer for that part of space. For example, giving these five points, which are part of the data set. In the diagram, the central part is for all queries which the answer is the central point, the other part acts in the similar way. We have:

$$size = O(n)$$

That means the number of lines and points in the diagram is n. What we can do is what is called point location, which says that when given such diagram and a query q, we can efficiently determine which part we fall into.
- Space: $O(n \log(n)$
- Query time: $O(\log n)$

Doubt 1: Does Voronoi diagram still perform well when we have $d \gg 2$?

Claim 2. *No. Note we have: Voronoi diagram size* $= n\lfloor \frac{d}{2} \rfloor$.

This means if the dimension is relatively high, Voronoi diagram's size grows exponentially faster in dimension. This kind of dependence, generally speaking, exponential dependence on the dimension is what is called course of dimensionality.

Theorem 3. *If we can solve NNS in dimension* $d = (\log^2(n))$ *with* $O(1)$ *pre-processing time and space and query time* $n^{0.99}$*, then SETH (strong exponential time hypothesis) is false.*

That is, we can't solve a general CNF formula on variables in time $\ll 2^n$.

3 Approximate NNS in high dimensions

What SETH means for us is that we don't really hope to solve Nearest Neighbor Search in polynomial space and linear time. Instead we will use an approximation.

Definition 4. *(r-near neighbor): Given* $r > 0$*, and a data set* $D \subset \mathbb{R}^d$*, build data structure such that given query* $q \in R^d$*, report any* $p \in D$ *such that* $||q - p|| \leq r$

This a "threshold" version of NNR in a sense: we report only points within certain radius r.

Definition 5. *(c-approximation of r-near neighbor, c-ANN): Given* $c > 1, r > 0$*, data set* $D \subset \mathbb{R}^d, n = |D|$*, build a data structure on* D *such that it can answer given* $q \in \mathbb{R}^d$*:*

- *if there exists* $\overset{*}{p} \in D$ *such that* $||\overset{*}{p} - q|| \leq r$*, then report* $p \in D$ *such that* $||p - q|| \leq c \cdot r$
  (in case of randomized algorithms, p is reported with probability at least 90% per point)
- *if doesn't exist* $\overset{*}{p} \in D$ *such that* $||\overset{*}{p} - q|| \leq r$*, then may or may not report anything*

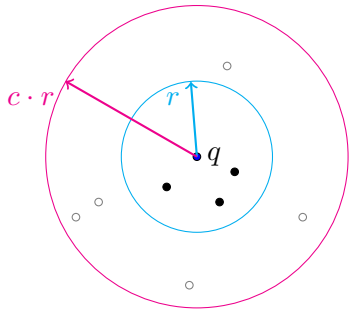

Figure 2: Because there are points within radius r, any of the black or grey points can be reported, even though only one of the points is the closest. If there were not black points within the radius r, c-ANN may or may not report the gray points.

We can think about approximations $1 \pm \varepsilon$, but we can also think about higher approximations.

Remarks:

1. There exists a formal reduction from approximate *nearest* neighbor to approximate *near* neighbor. We do not formally define the approximate nearest neighbor, but roughly, if there exists a closest point at distance r it reports a point at distance $c \cdot r$ (as opposed to a fixed threshold).

   It turns out we can do a reduction from the *nearest* neighbor to the *near* neighbor. Roughly the idea is to guess a threshold r, or enumerate all possible thresholds, and build a data structure for all possible r. In practice we usually know the threshold that we care about.

2. Usually, the algorithms for ANN can be modified to report a set $L \subset D$ such that:

   - if $p \in D$ is $||p - q|| \leq r$, then $p \in L$
   - if $||p - q|| \geq c \cdot r$, then $p \notin L$

3. Randomized algorithms report p with probability at least 90% per point if there exists $\overset{*}{p}$ (in definition 5 the first case)

So in c-ANN we are trying to improve over naive space $O(nd)$ and query time $O(nd)$.

3.1 Solution #1 for c-ANN

First idea is to use the previously seen dimensional reduction (Johnson–Lindenstrauss).

Algorithm 1: c-ANN using dimensional reduction

Pick $\varphi : \mathbb{R}^d \to \mathbb{R}^k$ at random, $k = O(\frac{\log n}{\varepsilon^2})$
Store $\varphi(p)$ for all $p \in D$
At query q:

- Compute $||\varphi(p) - \varphi(q)||$ for all $p \in D$
- Return the closest p

Even though we still iterate over all points, we reduced the size of the data set to k dimensions, so now the computation is less expensive. It's not as important that we reduced the space, but it is important

is that we somehow compressed the information about every point, so now computing the distances is faster.

New **space**: $O(nk) = O(n\frac{\log n}{\varepsilon^2})$

New query **time**: $O(dk) + O(nk) = O(n\frac{\log n}{\varepsilon^2})$

(Approximate $c = 1 + \varepsilon$, assuming $n >> d$)

Correctness analysis: Using Corollary of Johnson–Lindenstrauss on points $D \bigcup \{q\}$ it implies that distances between $p \in D$ and q are preserved up to $(1+\varepsilon)$ factor with probability at least $1-1/N \geq 1-1/n$ for $k = O(\frac{\log N}{\varepsilon^2}) = O(\frac{\log n}{\varepsilon^2})$, where $N = n + 1$.

A crucial aspect is that φ is *oblivious*, meaning that φ does not depend on the actual points. We pick the function φ without looking at any data sets. It is important because when we pick φ we don't know any queries yet.

Another way to look at how this algorithm works is that there exists some φ that mapped our original space and metric into something that is lower complexity, such that φ is defined on the entire metric and for all p, q have the distance between $\varphi(p)$ and $\varphi(q)$ approximately the same as the distance between p and q with probability at least $1 - 1/n^3$. In case of Johnson–Lindenstrauss it maps $\mathbb{R}^d$ to lower dimensional $\mathbb{R}^k$, nearly preserving the distances: $\varphi : (\mathbb{R}^d, \ell_2) \to (\mathbb{R}^k, \ell_2)$.

It is enough to use a sketching map instead of dimension reducing map. For example, for ℓ_1 we saw that we can't do dimension reduction, but we don't need to. Instead, we can do something different.

4 Sketching for ℓ_1

Theorem 6. *There exists a distribution over φ such that $\varphi : \mathbb{R}^d \to \mathbb{R}^k$ and estimator algorithm $E : \mathbb{R}^k \times \mathbb{R}^k \to \{0, 1\}$, and for all $x, y \in \mathbb{R}^d$:*

- *if $||x - y|| \leq r$ then $E(\varphi(x), \varphi(y)) = 0$*
- *if $||x - y|| \geq (1 + \varepsilon)r$ then $E(\varphi(x), \varphi(y)) = 1$*

with probability at least $1 - \delta$, $k = O(\frac{\log 1/\delta}{\varepsilon^2})$

(Proof is left for the next homework)

4.1 Hamming space

Consider $\{0, 1\}^d$ with ℓ_1 distance, which is called the Hamming space. All vector coordinates in this space are either 0 or 1. The distance between two vectors in this space is the number of different coordinates.

Theorem 7. *Fix $d, r \geq 1$, approximation $1 + \varepsilon$. There exists a distribution $\varphi : \{0, 1\}^d \to \{0, 1\}^k, k = O(\frac{\log n}{\varepsilon^2})$ such that:*

- *if $||x - y|| < r$ then $||\varphi(x) - \varphi(y)|| \leq \tau k$*
- *if $||x - y|| \geq (1 + \varepsilon)r$ then $||\varphi(x) - \varphi(y)|| > (\tau + \frac{\varepsilon}{9})k$*

where τ is a universal constant.

(Proof in the next lecture)

This can be seen almost like a dimension reduction from d dimensions to k, but with particular scale r. This implies $(1 + \varepsilon)$-ANN for Hamming space with the same parameters as Solution #1 for ℓ_2.