CS-67720 Metric Embedding Theory and Its Algorithmic Applications

Lecture 5

Lecturers: Yair Bartal and Nova Fandina *Fall 2018/19*
For comments contact fandina@gmail.com

5.1 Doubling Constant and Doubling Dimension

Recall a notion of 'dimension' for general metric spaces, we defined in the home work exercise:

Definition 5.1. *Let (X,d) be a metric space. The **doubling constant** of X is the minimal k such that for all $x \in X$, for all $r > 0$, $B(x,r)$ can be covered by at most k balls of radius $r/2$. The **doubling dimension** of X is $\dim(X) = \lceil \log_2(k) \rceil$, where k is the doubling constant of X.*

The following claim shows that for normed spaces, the doubling dimension captures the notion of a linear dimension of the space, up to a multiplicative factor.

Claim 5.1. *The doubling dimension of a d-dimensional normed space is $\Theta(d)$.*

Proof. As we have shown previously, in any ball $B(x,r)$, of radius r in a d-dimensional normed space, there is an $(r/2)$-net of the ball, of size $2^{O(d)}$. Hence, balls of radius $r/2$ around the points of the net will cover $B(x,r)$. Therefore, the doubling dimension is at most $O(d)$.

On the other hand, by the volume argument, in a d-dimensional normed space X, we need at least 2^d balls of radius $r/2$ to cover $B(x,r)$. Hence $\dim(X) \geq d$.

□

Many questions in metric embedding are formulated in terms of doubling dimension. Recall that Johnson-Lindenstrauss lemma allows to embed n points of ℓ_2 into ℓ_2 of dimension $O(\log n/\epsilon^2)$, with distortion $1+\epsilon$. This gives rise to the following questions, which are still open:

Open Problem 5.1. *Let $S \subset \ell_2$ be a finite set of n points. Can S be embedded into ℓ_2 of dimension $O(\dim(S)/\epsilon^2)$, and distortion $1+\epsilon$?*

A slightly weaker question:

Open Problem 5.2. *Can S be embedded into ℓ_2 of dimension $f(d)$ and distortion $g(d)$?*

On the positive side the following results have been shown.

Any n-point space X can be embedded into $\ell_p^{O(\log^2 n)}$ with distortion $O\left(\dim(X)^{1-1/p}\log^{1/p} n\right)$ [6]. This is tight for $p=2$: there exists a finite set with constant doubling dimension such that any embedding of it into ℓ_2 requires $\Omega\left(\sqrt{\log n}\right)$ distortion (Laakso graphs) [4]. Another related result, [1], states that any finite metric space X can be embedded into $\ell_p^{\tilde{O}\left(\frac{\dim(X)}{\delta}\right)}$ with distortion $O\left(\log^{(1+\delta)} n\right)$.

On the negative side, in [3] authors showed the impossibility result, for $p > 2$: There exists a finite $S \subset \ell_p$ with constant doubling dimension, such that any embedding $f : S \to \ell_p^d$, must have distortion at least $\Omega\left(\left(\frac{\log n}{d}\right)^{\frac{1}{2}-\frac{1}{p}}\right)$. Namely, there is no embedding of S with both distortion and dimension being dependent only on the doubling dimension of the space.

5.2 Nearest Neighbor Search

We continue our study with the very important problem of nearest neighbor search. We start with some approximate solutions that do not require embedding methods. Then, we will show that embeddings with relaxed grantees on distortion (so called, range preserving embedding) are very useful in this context.

Definition 5.2 (NNS problem)**.** *For a metric space (X, d) and a finite point subset $P \subseteq X$, propose a data structure that supports the nearest neighbor query: For a given $q \in X$, find $p \in P$ such that $d(q, p) = \min_{\bar{p} \in P} d(\bar{p}, q)$.*

We will focus on a very common setting in practice, where X is the ℓ_p^d normed space. For such setting, a naive brute-force approach is to store all the points of P in array. Then, for a given $q \in \mathbb{R}^d$, compute the distance $\|q - \bar{p}\|_p$ for each $\bar{p} \in P$ and return the minimal one. Space complexity/Preprocessing time and query time are $O(nd)$.

In the case of $d = 1$, there are more efficient solutions. For example, we can sort the points of P in $O(n \log n)$ time, and run a binary search when answering a query. In the case of $d = 2$, the Voronoi Diagram provides solution with the same parameters.

For higher dimensions there are no known solutions for the problem with both preprocessing and query time being sublinear on n and polynomial on d. There are billions of algorithms and heuristics for this problem, however all the known algorithms are of two types: 1) sublinear prepocessing cost, but query time is linear in n and exponential in d; 2) query time is subliniar in n and polynomial in d, but the preprocessing time is exponential in d (in terms of n^d factor). This phenomenon is known as a *curse of dimensionality*. See a very recent excellent review [2] of the state of the art in the (approximate) nearest neighbor search.

However, it is possible to reduce the strength of the curse by considering an approximate version of the NNS problem.

Definition 5.3 (α-ANN problem)**.** *For a metric space (X, d), a finite point subset $P \subseteq X$, and a parameter $\alpha \geq 1$, propose a data structure that supports the following query: Given $q \in X$, find $p \in P$, s.t $d(q, p) \leq \alpha \cdot \min_{\bar{p} \in P}\{d(\bar{p}, q)\}$.*

For a special case of $X = \ell_2^d$, we can use a tool we already have: the JL dimension reduction. Particularly, we can embed an n-point P into ℓ_2^k, with $k = O\left(\frac{\log n}{\epsilon^2}\right)$, and distortion $1 + \epsilon$. To answer a query $q \in \mathbb{R}^d$, first embed it to $\acute{q} \in \ell_2^k$ (with the same random bits as before) and then find a closest point to it in the image set of P. With constant probability, it is a $1+\epsilon$ approximation to the nearest neighbor on the original space. The query time is $O(\frac{\log n}{\epsilon^2} d) + O(n\frac{\log n}{\epsilon^2})$. However, this solution has preprocessing time $O(nd\frac{\log n}{\epsilon^2})$ (the space complexity is $O(n\frac{\log n}{\epsilon^2})$).

Other solutions to the $(1+\epsilon) - ANN$ are not that straightforward. Indyk and Motwani [5] proposed solution with query time $O(d \log n)$, and $O(n \log^2 n) \times (O\left(\frac{1}{\epsilon^d}\right))$ time for preprocessing, for any $p \geq 1$. For $p \in [1, 2]$ the authors developed a data structure with $O(d \log n)$ query time, and $(nd)^{O(1)}$ preprocessing time, where the power is $O(\frac{1}{\epsilon^2})$. We will show a slightly weaker result based on these constructions.

Particularity, for any $p \geq 1$, we present a $(1+\epsilon)$-approximation data structure with $O(n \log \Phi) \cdot O(\frac{1}{\epsilon^d})$ preprocessing time, and $O_\epsilon(d \log \log \Phi)$ query time, where $\Phi = d_{max}P/d_{min}P$ is the aspect ratio of P.

5.2.1 Reducing Approximate Nearest Neighbor

We start with a relaxation to our problem, an Approximate Range Nearest Neighbor problem.

Definition 5.4 ($(\alpha, r) - ARNN$ Problem)**.** *For a metric space (X, d), a finite point subset $P \subseteq X$, and parameters $r \geq 0, \alpha \geq 1$, propose a data structure that supports the following query: Given a query point $q \in X$, find $p \in P$ such that:*

- *if* $d(q,P) \leq r$, *then* $d(q,p) \leq \alpha \cdot r$;
- *if* $d(q,P) > r$, *then return either* $NONE$ *or some* $p \in P$, *such that* $d(q,p) \leq \alpha \cdot r$.

We show a reduction of α-ANN problem to (α, r)-$ARNN$ problem.

Claim 5.2. *Assume there is an efficient algorithm for* (α, r)-$ARNN$ *problem, for any* α *and* r, *then, for any parameter* $\beta > 1$, *there is an efficient algorithm for* $(\beta \cdot \alpha)$-ANN.

The idea is to perform a search on the values of r, i.e. we build a number of data structures for (α, r_i)-$ARNN$ with carefully chosen values r_i, and use them to answer the $\beta \cdot \alpha$ query.

Proof. Let $d_{min} = \min\{d(x,y) | x \neq y \in P\}$, and $d_{max} = \max\{d(x,y) | x \neq y \in P\}$. Define the following sequence of ranges: $r_1 = d_{min}/(2\alpha)$, $r_2 = \beta \cdot r_1$, $r_3 = \beta^2 \cdot r_1$, ..., $r_l = \beta^{l-1} \cdot r_1$, where l is minimal such that $r_l \geq \frac{d_{max}}{\beta\alpha - 1}$. Namely, $l = \lceil (\frac{\log(\Phi \frac{2\alpha}{\beta\alpha-1})}{\log \beta}) \rceil + 1$. Next we build the data structure for $(\alpha \cdot \beta)$-ANN.

The data structure for $(\alpha \cdot \beta)$**-**ANN. The data structure is just the collection of (α, r_i)-ARNN data structures. Next, we describe a simpler algorithm for supporting query, and then we explain how to speed it up.

The query algorithm is the following: for a given $q \in X$, *sequentially* query (α, r_i)-$ARNN$ on q, starting from $i = 1$. Once a subroutine returns a point z, stop and output z. Otherwise, stop after l steps and output any point $z \in P$.

If query on the data structure (α, r_1)-ARNN results in a point z, then $d(q,z) \leq \alpha r_1 = d_{min}/2$, and this implies that z is a nearest neighbor of q, as otherwise, if by contradiction, there is $p^* \in P$ such that $d(q,p^*) < d(q,z)$, then $d(z,p^*) \leq d(q,z) + d(q,p) < 2d(q,z) \leq d_{min}$.

Let $1 < t < l$ be such that queries on (α, r_i)-ARNN for all $1 \leq i \leq t$ are NONE, and for (α, r_{t+1})-ARNN the result is a point z. Then, $d(q,P) > r_t$ and $d(q,z) \leq \alpha r_{t+1} = \alpha\beta r_t$. Thus z is the $\beta\alpha$ approximation to the nearest neighbor of q, as required.

To complete the proof, note that if query on $(\alpha, r_l = d_{max}/(\beta\alpha - 1))$-ARNN returns NONE, then for $p* := argmin\ d(q,P)$, we have $d(q,p*) > r_l = d_{max}/(\beta\alpha - 1)$, i.e. $d_{max} < (\beta\alpha - 1) \cdot d(q,p^*)$. In addition, by the triangle inequality, for any $z \in P$, we have $d(q,z) \leq d(q,p^*) + d(p^*,z) \leq d(q,p^*) + d_{max} \leq (\alpha\beta) \cdot d(q,p^*)$.

Note, that we can perform a binary search on the first value of r_i that results in a point on applying (α, r_i)-ARNN (instead of a naive sequential search). This improves the query time to $(\log l) \times Q(n)$, where $Q(n)$ is the query time of one (α, r)-ARNN instance. □

We continue with an algorithm for the α-ANN problem. We have seen the reduction of this problem to the (α, r)-ARNN problem. Particularly, we have shown that for any $\alpha \geq 1$, and any $\beta > 1$, the $(\alpha\beta)$-ARNN problem can be solved by $O_{\alpha,\beta}(\log\log\Phi)$ calls to the instances of (α, r)-ARNN problem. Thus, it remains to present an algorithm for the (α, r)-ARNN itself (we focus on a d dimensional ℓ_p space).

5.2.2 Solution to $(\alpha, r) - ARNN$

Let $X = \ell_2^d$, and $P \subset X$ be an n-point subset. For any $r \geq 0$ we build an $(1+\epsilon, r)$-ARNN. We note that this construction easily generalizes to any ℓ_p^d, $p \geq 1$.

Claim 5.3. *There is a data structure for* $(1+\epsilon, r)$-$ARNN$ *problem, with preprocessing time* $n \cdot (1/\epsilon)^{O(d)}$, *and with query time* $O(d)$.

Proof. Without loss of generality, assume $r = 1$ (otherwise, rescale the instance). Impose a grid on $\mathbb{R}^d$, with the edge length $\epsilon/d^{\frac{1}{2}}$. Thus, the distance between any two points in the same grid-cell is at most ϵ. Note that we can uniquely index each cell of the grid, for example by the most right upper corner of the cell. Moreover, there exists an algorithm that in $O(d)$ time computes the index of the grid-cell a given point belongs to.

For all $p_i \in P$, let $B_i = B(p_i, 1)$, and let K_i be the union of all the grid-cells that intersect the ball B_i. Store all the cells of K_i, together with the information that indicates the ball to which they belong, in a hash table.

A query $q \in \ell_2^d$ processed as follows: Compute the index of the grid-cell q belongs to, and check whether this cell is a part of some ball B_i. If the answer is positive, the algorithm returns p_i. Indeed, in that case we get $d(q, p_i) \leq 1 + \epsilon$, as required. Otherwise, the cell of q does not intersect any of the balls B_i, thus the algorithm returns $NONE$.

The preprocessing time of the above algorithm is equivalent to the number of the grid-cells that cover all the balls B_i. The number of the cells that cover one ball is bounded by $\left(\frac{1}{\epsilon}\right)^{O(d)}$ (volume of the d-dimensional ball of radius $1+\epsilon$ divided by the volume of the grid cell with edge length $\epsilon/\sqrt{d}$. Note that the volume of the ball of radius r in ℓ_p is given by $V(p, r) = c_{d,p} \cdot r^d$, where $c_{d,p} \sim \frac{(const)^d}{d^{d/p}}$). Therefore, in total, preprocessing time (and the size) is $T = n \cdot \left(\frac{1}{\epsilon}\right)^{O(d)}$. The query time is $Q = O(d)$, assuming $O(1)$ time for hash table look-up. □

Getting back to the problem of our interest, we conclude:

Theorem 5.4. *There is a data structure for $(1+\epsilon)$-ANN in ℓ_p^d, with query time $O_\epsilon(d \log\log \Phi(P))$, and preprocessing time $O(n \log \Phi(P)) \times \left(\frac{1}{\epsilon}\right)^{O(d)}$.*

Proof. Take $\beta = 1 + \epsilon$, and note that the number of copies of $((1 + \epsilon), r_i)$-ARNN's is bounded by $l = O_\epsilon(\log \Phi)$, and the number of query calls is bounded by $O_\epsilon(\log\log \Phi)$. □

Note the dependency of the running times on the dimension of the instance.

References

[1] Ittai Abraham, Yair Bartal, and Ofer Neiman. Advances in metric embedding theory. STOC '06, pages 271–286, 2006.

[2] A. Andoni, P. Indyk, and I. Razenshteyn. Approximate Nearest Neighbor Search in High Dimensions. *ArXiv e-prints*, June 2018.

[3] Yair Bartal, Lee-Ad Gottlieb, and Ofer Neiman. On the impossibility of dimension reduction for doubling subsets of ℓ_p. *SIAM J. Discrete Math.*, 29(3):1207–1222, 2015.

[4] A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science*, pages 534–543, October 2003.

[5] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.

[6] Robert Krauthgamer, James R. Lee, Manor Mendel, and Assaf Naor. Measured descent: A new embedding method for finite metrics. *CoRR*, abs/cs/0412008, 2004.