COMS W6998-5: Algorithms through Geometric Lens (Fall'17) Nov 15, 2017

# Lecture 20 – Metric Embeddings into Trees, Distance Oracles

Instructors: *Alex Andoni, Ilya Razenshteyn* Scribes: *Xingyu Zhang*

## 1 Recap and Today's Agenda

Last time, we proved the following theorem:

**Theorem 1.** *Given $\Delta > 0$. Every n-point set $X$ admits a random partition $\mathcal{P}$ s.t.*

1. *Diameter of all parts is $\leq \Delta$.*
2. *$\forall x \in X$, $t \leq \frac{\Delta}{8}$.*

*we have $P(B(x,t) \subseteq \mathcal{P}) \geq (\frac{|B(x,\Delta/8)|}{|B(x,\Delta)|})^{\frac{8t}{\Delta}}$.*

Today, we will be studying the following topics:

1. Approximating a metric using trees
2. $l_1$-subsets of metrics
3. Using data structures to build "distance oracles"

## 2 Approximating a metric with trees

We will first prove the following theorem:

**Theorem 2.** *$\forall n$-point sets $X$, there exists a distribution over a set of weighted trees $T$ s.t.*

1. *$P_T[d_T(x_1,x_2) \geq d_X(x_1,x_2)] = 1, \quad \forall x_1, x_2$*
2. *$\forall x_1, x_2$, $E_T[d_T(x_1,x_2)] \leq O(logn) d_X(x_1,x_2)$*

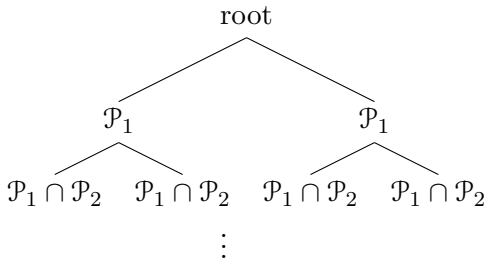
The previous theorem states that we can embed any metric on a finite set into a metric on a tree, with distortion at most $O(logn)$ in expectation. This result has many applications in online algorithms.

We now describe the procedure by which we randomly construct the trees. Let $diam(X)$ denote the largest pairwise distance between any two points in $X$. Define:

$$\Delta_1 = diam(X)/8$$
$$\Delta_2 = diam(X)/64$$
$$\vdots$$
$$\Delta_k = diam(X)/8^k$$
$$\vdots$$
$$\Delta_L = diam(X)/8^L$$

We stop the procedure at $L$, where $\Delta_L <$ any pairwise distance. So $L \approx log(diam(X))$. We invoke Theorem 1 to get a partition $P_k$ for each $\Delta_k$ for $1 \leq k \leq L$.

We construct the following tree:




At level $i$, the weights of the edges are $4\Delta_i$. The last level are a bunch of leaf nodes and there is one node for each point of $X$. This tree gives information about what partitions each vertex belongs to. For example, if they're both descendants of a common vertex, then they belong to the same partitions indicated by the vertex. Now we will prove the Theorem 2.

*Proof.* First, we will prove that $d_T(x_1, x_2) \geq d(x_1, x_2)$. Find the least common ancestor of $x_1$ and $x_2$ and suppose it is at level $u$. The downward edges from the least common ancestor have weight $4\Delta_u$, thus any distance from $x_1$ to $x_2$ in the tree must have weight at least $8\Delta_u$. However, $x_1$ and $x_2$ are part of the same partition $\mathcal{P}_{u-1}$, thus, $d(x_1, x_2) \leq \Delta_{u-1} = 8\Delta_u$. Hence we prove the first claim.

Next, we will prove that $E_T(d_T(x_1, x_2)) \leq O(logn)d(x_1, x_2)$. Let $i^*$ be such that $\Delta_{i^*} = \frac{diam(X)}{8^{i^*}} \approx 100d(x_1, x_2)log(n)$. Then notice that

$$
\begin{aligned}
E_T(d_T(x_1,x_2)) &\le O(1)E[\Delta_u] \\
&= O(1)\sum_{i=1}^{L}\Delta_i P(u=i) \\
&= O(1)(\sum_{i=1}^{i^*}\Delta_i P(u=i)+\sum_{i=i^*}^{L}\Delta_i P(u=i)) \\
&\le O(1)(\sum_{i=1}^{i^*}\Delta_i P(u=i)+\sum_{i=i^*}^{L}\Delta_i) \\
&\le O(logn)d(x_1,x_2)+O(1)\sum_{i=1}^{i^*}\Delta_i P(u=i)
\end{aligned}
$$

From Theorem 1, we also get that

$$
\begin{aligned}
P(u=i) &\le P(\mathcal{P}_i \text{ separates } x_1,x_2) \\
&\le 1-P(B(x_1,d(x_1,x_2))\subseteq \mathcal{P}_i(x_1)) \\
&\le 1-(\frac{|B(x_i,\Delta_{i+1})|}{|B(x_1,\Delta_i)|})^{\frac{8d(x_1,x_2)}{\Delta_i}} \\
&\approx O(\frac{d(x_1,x_2)log(\frac{|B(x_i,\Delta_{i+1})|}{|B(x_1,\Delta_i)|})}{\Delta_i})
\end{aligned}
$$

Using this, we can finish the proof:

$$
\begin{aligned}
E_T(d_T(x_1,x_2)) &\le O(logn)d(x_1,x_2)+O(1)\sum_{i=1}^{i^*}\Delta_i P(u=i) \\
&\le O(logn)d(x_1,x_2)+O(1)\sum_{i=1}^{i^*}d(x_1,x_2)log(\frac{|B(x_i,\Delta_{i+1})|}{|B(x_1,\Delta_i)|} \\
&\le O(logn)d(x_1,x_2)
\end{aligned}
$$

□

# 3 Embedding into $l_1$

**Theorem 3.** *For every n-point metric $X$, $\forall\epsilon>0$, $\exists X'\subset X$, $|X'|=n^{1-\epsilon}$ s.t. $X$ embeds into $l_1/l_2$ with distortion $O(1/\epsilon)$.*

**Definition 4.** *Let $0<\alpha<1/8$. $x\in X$ is $\alpha$-padded if $\forall i$, $B(x,\alpha\Delta_i)\subseteq\mathcal{P}_i(x)$.*

**Claim 5.** *$\forall x$, $0<\alpha<1/8$, $P(x \text{ is } \alpha-padded)\ge n^{-8\alpha}$.*

*Proof.*

$$\begin{aligned} P(x \text{ is } \alpha - \text{padded}) &= \prod_{i=1}^{L} P[B(x, \alpha\Delta_i) \subseteq \mathcal{P}_i(x)] \\ &\geq \prod_{i=1}^{L} (\frac{|B(x, \Delta_{i+1})|}{|B(x, \Delta_i)|})^{8\alpha} \\ &\geq n^{-8a} \end{aligned}$$

□

**Claim 6.** *$\forall x_1, x_2 \in X$ s.t. $x_1$ is $\alpha$-padded, we have $d_T(x_1, x_2) \leq O(1/\alpha) d(x_1, x_2)$.*

Now we prove the theorem:

*Proof.* Here is the construction:

1. Sample $T$
2. $E$[number of $\alpha$-padded points] $\geq n^{1-8\alpha}$ (set $\alpha = \epsilon/8$).
3. In the tree, $O(1/\alpha)$ preserves distances between $\alpha$-padded points.

The theorem follows from the following claim:

**Claim 7.** *Any tree metric embeds into $l_1$ with distortion 1.*

□

# 4 Distance Oracles

Suppose we have $X$ which is an $n - point$ and we wish to store distances between points. The naive approach is to store every pair of points, which would require $n^2$ space.

We will see that distance oracles allow one to save space but at the cost of approximation.

**Theorem 8.** *Let $\epsilon > 0$ be given. $\forall$ n-point $X$, there exists a data structure of size $O(n^{1+\epsilon})$ such that given two points $x_1, x_2$, you can approximate $d(x_1, x_2)$ up to $O(1/\epsilon)$ in time $O(1)$.*

The idea is to sample $O(n^\epsilon)$ trees $T_i$ and then collapse all the "trivial" paths. Since $P$(x is $\Theta(\epsilon)$ − padded) $\geq n^{-\epsilon}$, every $x \in X$ is $\Theta(\epsilon)$-padded in $T_i$. Record this i.

Remains to do: Given a weighted $T$ on $O(n)$ nodes, preprocess the tree to get a data-structure of size $O(nlogn)$ such that it just takes $O(1)$ time to compute distances. Here is how you do it:

First run DFS to traverse all the nodes and record the order of traversal. For example:

$$DFS : 1, 2, 4, 5, 2, 1, 3, 6, 3, 1$$

This array is linear size. The least common ancestor of two nodes is the number in between the two nodes in the DFS array which is closest to the root. To find this, we store an array of depth. For example:

$$d_i : 0, 1, 2, 1, 2, 1, 0, 1, 2, 1, 0$$

From the array of depth, we wish to choose the smallest number in between. The problem of doing this is called the range minimum query problem. The solution to this problem is to store $Ans(i, j)$ which is the minimum of $d_i, d_{i+1}, \ldots, d_{i+2^j-1}$. This takes $O(nlogn)$ space and time. With this, we can find the minimum over any sequence, since it is the union of subsequences of length of a power of 2.

The distance between $x_1$ and $x_2$ can be approximated by $l(x_1) + l(x_2) - 2l(lca(x_1, x_2))$.

Next time, we will talk about the following theorem:

**Theorem 9.** *For any n-point metric $k \geq 2$, there exists a data structure of size $O(n^{1+1/k})$, (2k-1)-approximation, $O(k)$ query time.*