Randomized Algorithms 2023A – Lecture 6
Least Squares Regression and Probabilistic Embedding into Dominating Trees*

Robert Krauthgamer

1 Least Squares Regression

Problem definition: In *Least Squares Regression*, the input is a matrix $A \in \mathbb{R}^{n\times d}$ and a vector $b \in \mathbb{R}^n$, and the goal is to find $\text{argmin}\{\|Ax^* - b\| : x^* \in \mathbb{R}^d\}$.

Informally, when solving a system $Ax^* = b$ that is over-constrained ($n \gg d$), we do not expect to find an exact solution, and we want to minimize the sum of squared errors $\sum_i (A_i x^* - b_i)^2$.

We shall consider $(1+\varepsilon)$-approximation, i.e., finding $x' \in \mathbb{R}^d$ such that

$$\|Ax' - b\| \le (1+\varepsilon) \min_{x^*\in\mathbb{R}^d} \|Ax^* - b\|. \tag{1}$$

Theorem: Let $S \in \mathbb{R}^{s\times n}$ be an $(\varepsilon, \delta, d+1)$-OSE matrix. Then for every regression instance $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, with high probability, an optimal solution x' (or even $(1+\varepsilon)$-approximation) to the regression instance $\langle SA, Sb\rangle$ is a $(1+O(\varepsilon))$-approximation to the instance $\langle A, b\rangle$, i.e., such x' satisfies (1).

This theorem essentially reduces a regression problem with n constraints to regression with s constraints, but we should take into account also the time to compute SA.

Proof: As explained in class, it follows from applying the OSE guarantee to the linear subspace spanned by the columns of A and by b (total of $d+1$ vectors), and then

$$(1-\varepsilon)\|Ax' - b\| \le \|SAx' - Sb\| = \min_{x\in\mathbb{R}^d} \|SAx - Sb\| \le (1+\varepsilon) \min_{x^*\in\mathbb{R}^d} \|Ax^* - b\|.$$

2 Metric Embeddings

Definition (metric space): We say that (X, d) is a *metric space*, if X is a set (of points), and $d : X \times X \to \mathbb{R}_+$ (a distance function) is symmetric, non-negative (with 0 only between a point

*These notes summarize the material covered in class, usually skipping proofs, details, examples and so forth, and possibly adding some remarks, or pointers. The exercises are for self-practice and need not be handed in. In the interest of brevity, most references and credits were omitted.

and itself), and satisfies the triangle inequality.

Prime examples: A simple example is the Euclidean space $\mathbb{R}^d$. Or one can take a subset of its points.

Given a graph with positive (or non-negative) edge weights $G = (V, E, w)$, its shortest-path metric d_G is a metric on the vertex set V. Or one can take a subset $V' \subset V$.

Optimization problems: Many optimization problems are naturally defined on metric spaces, for example TSP and k-median. (The input may specify a subset of the points to be visited, clustered, potential centers, etc.)

Definition (embedding): An *embedding* of a metric space (X, d_X) into a metric space (Y, d_Y) is a map $f : X \to Y$. Its *distortion* is the least $D = D_1 D_2 \geq 1$ such that

$$\forall x, x' \in X, \quad \frac{1}{D_1} d_X(x, x') \leq d_Y(f(x), f(x')) \leq D_2 \cdot d_Y(d_X(x, x')).$$

Remark: In many cases, we can scale distances in Y and thus assume WLOG that $D_1 = 1$ (or alternatively $D_2 = 1$).

Definition (tree metric): A metric space (X, d) is called a *tree metric* if there exists a tree G such that

Exer: Show that a metric space (X, d) is a tree metric if and only if it satisfies the following (called 4-point condition)

$$\forall x, y, z, w \in X, \quad d(w, x) + d(y, z) \leq \max\{(d(w, y) + d(x, z), d(w, z) + d(x, y)\}.$$

Many optimization problems can be solved in polynomial time in tree metrics, including TSP and k-median (hint: use dynamic programming).

Observation: Given a metric space (X, d_X) and a distortion-D embedding of it into a tree metric (Y, d_Y), one can compute a D-approximate solution for TSP and k-median.

This promising approach has the following serious obstacle, which we will bypass using randomization.

Theorem 1 [Rabinovich and Raz, 1998]: Every embedding of the shortest-path metric of C_n, an unweighted n-cycle, into a tree metric has distortion $\Omega(n)$.

Remark: This special case where the tree metric is a spanning tree of C_n is easy, the general case requires a proof.

Example [Karp]: Let T be a spanning tree of C_n that is obtained by removing uniformly random edge. Then for all $x, y \in C_n$,

$$d_T(x, y) \geq d_{C_n}(x, y).$$
$$\mathbb{E}[d_T(x, y)] \leq 2 d_{C_n}(x, y).$$

Remark: Extends to a cycle with edge lengths by sampling proportionally to the edge lengths.

2.1 Probabilistic Embedding

Probabilistic embedding into trees: A *probabilistic embedding* of a metric (X, d) into trees is a probability distribution over mappings $f : X \to T$ and tree metrics (T, d_T).

The tree T is called *dominating* if

$$\forall x, y \in X, \qquad d_T(f(x), f(y)) \geq d(x, y).$$

The probabilistic embedding has *distortion* $D \geq 1$ if

$$\forall x, y \in X, \qquad d_T(f(x), f(y)) \leq D \cdot d(x, y).$$

Remark 1: As we saw above, the n-cycle C_n admits a probabilistic embedding into dominating trees with distortion 2.

Remark 2: T is random (not fixed) and may contain Steiner points (points that are not images under f).

2.2 Probabilistic Embedding into Dominating Trees

Theorem 2 [Bartal'96, Fakcharoenphol-Rao-Talwar'03]: Every n-point metric admits a probabilistic embedding into dominating trees with distortion $O(\log n)$.

Example application I: Metric TSP:

Given a TSP instance which is an n-point metric space (X, d), apply the theorem to randomly construct a tree T with metric d_T. Now solve TSP on this tree optimally by going around the tree twice (assuming all leaves are point in X, otherwise we can prune such vertices). Finally, output the same tour (same permutation of points) as a solution to TSP on (X, d).

Analysis: First bound the algorithm's performance

$$ALG(X, d) \leq ALG(X, d_T) = TSP(X, d_T),$$

then bound the expectation of the optimum in the tree

$$\mathbb{E}[TSP(X, d_T)] \leq O(\log n) TSP(X, d).$$

Key property: the objective is linear in the distances.

Remark: It works similarly even with $O(1)$-approximation for TSP in trees.

Remark: There is a much better algorithm for metric TSP (approximation 2 by twice MST, and even 3/2 by Christofides), but this approach works also for generalizations like vehicle routing.

Example application II: k-median:

Given an n-point metric (X, d), find a set $S \subset X$ of k points (called medians) that minimizes $\sum_{x \in X} d(x, S)$.

Again, apply the theorem to construct a tree T with metric d_T, and solve the instance optimally using dynamic programming along the tree. The analysis is similar.

Another example: min-sum clustering (again break X into k sets, but now the objective is the sum of distances among all pairs inside the same set).

Proof of Theorem 2:

Assume WLOG that the minimum interpoint distance in X is 2, and denote the maximum as $\Delta = \text{diam}(X)$, and $\delta = \lceil \log_2 \Delta \rceil$.

We may refer to X as a complete graph, to every pair of points (x, y) as an edge.

The main usage of this theorem is that it "reduces" problems about X to problems about a tree (metric), which is usually easier. We will see/discuss these applications in the next class.

Definition (hierarchical decomposition): A *hierarchical decomposition* of X is a sequence $P_L, \ldots, P_1, P_0$ of partitions of X, such that

a) $P_L = \{X\}$ (the trivial partition)

b) each P_i is a refinement of P_{i+1}, i.e., each element of P_i, referred to as a *cluster* $S \subseteq X$, is contained entirely in some cluster of P_{i+1}.

c) all clusters in P_i have diameter at most 2^i. Thus, $P_0 = \big\{\{x\} : x \in X\big\}$ (all clusters are singletons).

Building a tree: Given a hierarchical decomposition, we build a tree metric T with $L+1$ levels, where the vertices at level i are the clusters of P_i. Start with a root that corresponds to the single cluster X of P_L. Let each cluster of P_i be the child of the cluster in P_{i+1} that contains it, and let the edge between them have length 2^i. The leaves correspond to clusters that are singletons, and we can thus let the embedding f map each $x \in X$ to the leaf which is the singleton cluster $\{x\}$.

Exer: Extend the proof below to obtain a tree T' whose vertex set is exactly X (without additional vertices).

Hint: Get rid of non-leaf vertices in T by "mapping" them to leaves.

Lemma 3: For every two points $x, y \in X$ there is a unique integer i such that x, y are in the same cluster of P_{i+1} but not of P_i. Moreover, $d_T(x, y) \in [2 \cdot 2^i, 4 \cdot 2^i)$.

Proof: immediate.

Lemma 4: This (hierarchical) tree metric d_T dominates (X, d).

Proof: immediate from Lemma 3 (and seen in class).

Lemma 5: Suppose the hierarchical decomposition is randomized and guarantees, for a certain $\alpha > 0$, that

$$\forall x, y \in X, \forall i, \quad \Pr[x, y \text{ are in different clusters of } P_i] \leq \alpha \cdot \tfrac{d(x,y)}{2^{i+1}}.$$

Then the embedding has distortion $O(\alpha \log \Delta)$, i.e., $\mathbb{E}[d_T(x, y)] \leq O(\alpha \log \Delta) d(x, y)$.

Remark: This is weaker than Theorem 2, and we will later show a stronger bound.

Proof: Was seen in class.

2.3 Randomized Decomposition

Intuition: We start with a randomized algorithm that partitions X into clusters of diameter 2^i (without a hierarchy).

Algorithm A (partitioning X at a given scale 2^i):

1. choose a random permutation $\pi : [n] \to X$ and a random $\beta \in [1, 2]$

2. initialize $P \leftarrow \emptyset$

3. for $l = 1$ to n do

4. add to P a new cluster consisting of all point in X that are within distance $\beta_i = \beta 2^{i-2}$ from $\pi(l) \in X$ and are not already in any cluster of P.

Observations:

a) Every cluster has a "center" point $\pi(l)$, but it need not contain the center.

b) We can think of lines (3-4) as if each vertex in X assigns itself to the first center, according to the order π, within distance β_i.

c) Every cluster has diameter at most $2\beta_i \leq 2^i$.

d) The algorithm may create empty clusters but we can discard them.

Algorithm B (for hierarchical partitioning of X):

1. choose a random permutation $\pi : [n] \to X$ and a random $\beta \in [1, 2]$

2. initialize $P_L \leftarrow \{X\}$

3. for $i = L - 1$ down to 0 do

4. let $P_i \leftarrow \emptyset$

5. for $l = 1$ to n do

6. for every cluster $S \in P_{i+1}$

7. add to P_i a new cluster consisting of all points in S that are within distance $\beta_i = \beta 2^{i-2}$ from $\pi(l)$ and are not already in any cluster of P_i.

Observation: This is like applying Algorithm A recursively to partition each $S \in P_{i+1}$, except that the "centers" are taken from all of X and not only from S. Another difference is that all scales use the same π and β.

We will analyze this algorithm and finish the proof of Theorem 2 next time.