CS 6301.008.18S Lecture—April 10, 2018

Main topics are #metric_embeddings, the #JL_lemma, and #Bourgain's_theorem.

Prelude

- Please sign up for presentations.
- The projects do not need to be completely done by presentation time, but I do want to hear about the work you've done by then.

Metric Spaces

- Often we'll want some way to compare different objects to tell how similar they are.
- We've been dealing with these kinds of comparisons all semester with points. Usually we care about their Euclidean distance.
- We also saw an example with more complicated objects. For example, we looked at the Fréchet distance between two curves.
- These kinds of distances can be generalized into something we call a metric space.
- A *metric space* (X, d) consists of a (possibly infinite) set X and a *distance function* d : X x X ➡ R_≥0 such that
  - d(x, x) = 0 for all x in X (an element is distance 0 from itself)
  - d(x, y) > 0 for all distinct x, y in X (different element have non-zero distance)
  - d(x, y) = d(y, x) for x, y in X (distances are symmetric)
  - d(x, y) + d(y, z) ≥ d(x, z) for all x, y, z in X (distances follow the triangle inequality)
- Points in R^d form a metric space under Euclidean distance. Curves in R^d form a metric space under the Hausdorff and Fréchet distances.
- We often like to talk about general metric spaces (X, d), because the triangle inequality is enough to show some interesting results.
- For example, you can define the k-center problem for any metric space (X, d) where X is finite with n elements: just find a collection C in X of size k that minimizes the maximum distance from any element of X to an element of C.
- This problem is NP-hard, because it's a generalization of the version we saw where X was a subset of points in the plane.
- But, assuming you know the (n choose 2) distances, that algorithm by Gonzalez we saw works perfectly fine and gives you a 2-approximation.
- The PTAS requires more structure, though, so you really do need something like points in R^d.
- Similarly, you can approximately solve the traveling salesperson problem of visiting every element of a metric while trying to minimize the total distance between consecutive pairs

of elements from your sequence of visits. As long as you know the distances, you can get a 3/2-approximation.

Metric Embeddings

- But maybe you don't want to store, or even compute, all the (n choose 2) distances. If you're working with very high dimensional objects or points, these computations can be very expensive.
- One solution is to map the elements of your metric into elements of another one that's easier to work with. But then you usually distort the distances.
- Given two metric spaces (X, d_X) and (Y, d_Y), a function $f : X \rightarrow Y$ is called a D-embedding if there exists a scaling factor $r > 0$ such that for all x, x' in X
  - $r\, d_X(x, x') \leq d_Y(f(x), f(x')) \leq D\, r\, d_X(x, x')$.
- In other words, some distances may shrink and others may increase, but they all change the same up to a D factor.
- A 1-embedding scales all distances by exactly the same factor. It's essentially the same metric, so we call it an *isometric embedding*.
- Generally, you hope to find embeddings into much simpler metrics while keeping D is as small as possible.
- But that isn't always possible depending on what your requirements are.
- For example, consider the *graphic metric* (V, d) for a graph G = (V, E). Here, d(u, v) is the number of edges in the shortest path from u to v.
- Suppose we want to simplify an arbitrary graphic metric on a graph G by embedding it into some spanning tree of G.
- Well, if G is a cycle on n vertices, this is a problem. You'll have to remove an edge between some pair of vertices u and v. But now the distance between u and v has increased by a factor of (n-1)! On the other hand, many other distances are exactly preserved. You get an (n - 1)-embedding.
- There are cases where you get much better results, though.

The Johnson-Lindenstrauss Lemma

- Let's say we have n points in R^d where d is a large number.
- Working with such high dimensional points is difficult, so we'd like to map them to a lower dimensional Euclidean space, but we don't want to distort the distances much.
- Fortunately, there's a fairly simple randomized algorithm for finding a good mapping.
- "Lemma" (Johnson-Lindenstrauss): Let X be a set of n points in R^d, and fix $0 < \delta < 1$. There exists a $(1 + \delta)$-embedding of X into R^k (using Euclidean distances) with $k = O((\log n) / \delta^2)$.

- It's going to be helpful to work with a set V of "vectors" instead of "points".
- We'll specifically show if you fix a 0 < eps < 1, and let k ≥ 4 * (eps^2 / 2 - eps^3 / 3)^(-1) * ln n, then there is an embedding where (1 - eps)||u - v||^2 ≤ ||f(u) - f(v)||^2 ≤ (1 + eps)||u - v||^2 for any pair u, v in V where f : R^d → R^k.
- Choose eps correctly in relation to delta to get the JL-lemma.
- The transformation is fairly simple. We'll going to choose a random k-dimensional subspace S of R^d and find the projection of every vector in V to this subspace. Then we'll just scale things outward again so the expected squared length of a vector remains is equal to its squared length before the projection.
- We'll show that an arbitrary vector has its squared length off from the expected value by a factor more than (1 ± eps) with probability at most O(1 / n^2). There's only (n choose 2) differences vectors for members of V, so there is a less than 1 probability than *any* of the squared lengths we care about will change. Meaning, some projection is fine.
- To talk about length changes, its enough to discuss the distribution for the squared length of an arbitrary *unit vector* to a random subspace.
- But that's difficult to reason about, so instead let's fix the subspace and look at the squared length of a random unit vector projected to the subspace. It's the same distribution. For simplicity, we'll use the space spanned by the first k coordinates, so the projection is just those first k coordinates.
- Alright, so how do you choose a random vector? Let X_1, …, X_d be d independent Gaussian N(0, 1) random variables, and let Y = 1 / ||X|| (X_1, …, X_d). I'm going to claim without proof that Y is a point chosen uniformly at random from the d-dimensional sphere S^{d-1}.
- Let Z in R^k be the projection of Y onto its first k coordinates (so the projection onto our fixed subspace).
- Let L = ||Z^2||, the squared length we care about.
- Let mu = E[L]. We have mu
  - = E[X_1^2 + … + X_k^2] / E[X_1^2 + … + X_d^2]
  - = (E[X_1^2] + … + E[X_k^2]) / (E[X_1^2 + … + E[X_d^2])
  - = k /d, because each X_i comes from the same distribution.
- So, now we need to argue that the distribution of L is tightly concentrated around k / d. This requires some subtle probability arguments, but they ultimately lead to this lemma:
- Lemma: Let k < d, then
  - If beta < 1, Pr[L ≤ beta k / d] ≤ exp((k / 2)(1 - beta + ln beta),
  - or if beta > 1, Pr[L ≥ beta k /d] ≤ exp((k / 2)(1 - beta + ln beta)
  - where exp(.) = e^(.).
- So as beta gets away from 1, the expression 1 - beta + ln beta gets more and more negative, and quickly. Choosing larger values of k just causes that negativity to grow even faster.

- And since its all in an exponent, that means the probability drops very fast as L gets away from the mean of k / d.
- OK, why do we choose k ≥_ 4 * (eps^2 / 2 - eps^3 / 3)^(-1) * ln n?
- To define our map f : R^d ➡ R^k, let S be the random k-dimensional subspace we're going to project into, and let v_i' be the projection of each v_i in V to S.
- Now, consider any pair v_i, v_j, let L' = ||v_i' - v_j'||^2, and let mu' = (k / d)||v_i - v_j||^2 be the expected value of L'.
- By the lemma, Pr[L ≤ (1 - eps)mu')
  - ≤ exp((k / 2) (1 - (1 - eps) + ln(1 - eps)))
  - ≤ exp((k / 2)(eps - (eps + eps^2 / 2)))
    - (here, I used the first couple terms of the Taylor expansion for ln implying ln(1 - x) ≤_-x - x^2 / 2 for all 0 ≤_ x < 1)
  - = exp(- k eps^2 / 4)
  - ≤ exp(2 ln n) = 1 / n^2
- Similarly, Pr[L ≥_(1 + eps)mu')
  - ≤ exp((k / 2)(1 - (1 + eps) + ln(1 + eps)))
  - ≤ exp((k / 2)(-eps + (eps - eps^2 / 2 + eps^3 / 3)))
    - (ln(1 + x) ≤_ x - x^2 / 2 + x^3 / 3 for all x ≥_ 0)
  - = exp(-(k(eps^2 / 2 - eps^3 / 3)) / 2)
  - ≤ exp(-2 ln n) = 1 / n^2
- We'll use the map f(v_i) = (sqrt(d / k)) v_i'.
- We just argued that for any pair of vectors v_i v_j in V, the probability that ||f(v_i) - f(v_j)||^2 / ||v_i - v_j||^2 does not lie in the range [1 - eps, 1 + eps] is at most 2 / n^2.
- There are (n choose 2) pairs, so the probability that *some* pair suffers large distortion is at most (n choose 2) * 2 / n^2 = 1 - 1/n.
- By sticking a larger constant next to the ln, you can make the probability of failure as low as 1 / n^c for any constant c you desire, meaning we even have a randomized Monte Carlo algorithm for computing a good projection.
- If you have time to test projection quality, then the algorithm is Las Vegas instead.

Bourgain's Theorem

- I'd like to finish by discussing something a bit weaker, but more general.
- Let (X, d) be any metric space over n elements. Bourgain's theorem says there is an O(log n)-embedding of X into O(log^2 n)-dimensional Euclidean space.
- I won't go into the proof at all, but I'll give the surprisingly simple construction.
- For every 1 ≤ i ≤_ c log n (for sufficiently large c), for every 1 ≤ j ≤ ceil(log n), independently construct a set A_{i j} where each element in X is selected with probability 2^{-j}.
- Now, define d(x, A_{i j}) = min_{y in A_{i j}} d(x, y) to be the distance from x to the subset

A_{i j}.

- Finally, let f(x) = < d(x, A_{i j}) | 1 ≤ i ≤_ c log n, 1 ≤ j ≤ ceil(log n)> which is a vector in O(log^2 n)-dimensional space.
- Again, I won't prove it, but f is an O(log n)-embedding with non-zero probability if c is sufficiently large.