

Disclaimer: *These are rough notes, with some exercises.*

25.1 Dimension Reduction: another (easier) application of projection.

Say we are given n points in R^d . Say d is very large, say the points correspond to images, in which case, the dimension d is the number of pixels.

Question: Can we represent the distances between the n points in a small dimension? How would we do this?

Maybe. We would clearly just project onto k randomly chosen directions.

Question: How would we analyze this?

The expected projection length of a length ℓ vector is ℓ/\sqrt{d} for a single random direction. Moreover, the For this process the length for k trials can be expressed as

$$\sqrt{\sum_i Y_i^2},$$

where Y_i is a random variable that corresponds to the projection length onto the i th vector. Or one, just say the squared length is

$$\sum_i Z_i,$$

(with Z_i being Y_i^2) and (ignoring square roots) hope it is within $1 + / - \epsilon$ of what it should be. For small ϵ the square root only changes the error (ϵ) only changes things by a constant factor (around 2.)

Question: How would we analyze this?

This is the sum of a set of k independent random variables that have a Gaussian distribution. Thus, the variance is now $k\sigma^2$ and the mean is $k\mu$ where the variance of one variable is σ^2 and its mean is μ .

This again is distribution according to a gaussian distribution.

That is,

$$Pr[|X - E[X]| \geq t\sqrt{Var[X]}] \leq e^{-t^2/2}.$$

Question: Can we bound the probability that it varies from its expectation by a ϵ factor?

We wish to set t so that

$$t\sqrt{\text{Var}[X]} \leq \epsilon E[X]$$

and we wish that

$$e^{-t^2/2} \leq 1/n^3$$

so that every pair's distance is preserved to within an $1 + \epsilon$ factor.

The latter inequality holds when $t = \sqrt{6 \ln n}$.

Question: How do we enforce the former?

We'll plug in. $E[X] = k\mu$, $\text{Var}[X] = k\sigma^2$. Moreover, $\sigma^2 = \mu^2$ for our individual variables.

Thus, we have

$$t\sqrt{k}\mu \leq \epsilon k\mu$$

, and plugging in $t = \ln n$ (ignoring the constant 6 above) we get

$$\sqrt{kl\ln n}\mu \leq \epsilon k\mu.$$

Solving for k , we get that everything we want holds when

$$k \geq \ln n / \epsilon^2.$$

25.2 Projection for computer scientists.

We presumably like the hamming metric. How many bits do we differ in. (Again with the hypercube.) Points could be files, or b/w images, and thus may have large dimension.

Question: Can we project?

Well, not exactly. Indeed, one can show that projection can't work (not too easily though) without significant distortion.

Question: Can we project?

(This is like my kids, not really hearing the answer No.)

Well, ok then. Sort of. (No wonder my kids keep asking...)

Let's say we would just like to tell whether two points are within Δ or outside of say $(1 + \epsilon)\Delta$.

Then, we can simply have $O(\log D)$ such datastructures and basically "represent" the distances, where the farthest pair is D apart.

Question: Can we do this?

Sure. First, what is “this”. Find an embedding into a small dimensional embedding (say into $O(\log n)$ dimensions) where two nodes that are within distance Δ are close in this embedding and where two that are farther than $(1 + \epsilon)\Delta$ away are far in this embedding.

Consider a particular Δ which is less than d , of course.

Say, we pick n/Δ random coordinates, and take the parity of the values, and output the result.

For two nodes that differ in Δ places, what is the probability that the result is different?

Well, the probability that we choose no bits in which they differ is at most $(1 - \Delta/n)^{n/\Delta}$. Ack..it's $1/e$ again.

Ok, so with probability at least $1 - 1/e$ we get at least one bit in which they differ. But, the parity may be the same. That is, there may be an even number of difference.

Question: How do we deal with this?

Well, now jitter the thing by dropping half the coordinates. This has the affect of making the number of differences being even or odd being equally likely.

Question: Overall, how do we reduce the dimension?

So, we construct a coordinate by choosing n/Δ random coordinates, and then dropping half of them.

Question: What property do we have for the coordinate?

If a pair of nodes is at distance at least Δ , then the probability that they agree on a coordinate is at most

$$\frac{1}{2}(1 - 1/e).$$

Question: Closer nodes?

What about nodes that are closer by ϵ ? The probability that the disagree is at most

$$(1 - (1 - \epsilon)\Delta/n)^{n/\Delta} \approx e^{-1 - \epsilon} = e^{-1} e^{-\epsilon}.$$

And thus, the probability that the agree is at least

$$\frac{1}{2}((1 - 1/e) + \epsilon e^{-1}).$$

(Note: I apologize that I switched from Δ versus $(1 + \epsilon)\Delta$ to $(1 - \epsilon)\Delta$ versus Δ . It is easy enough to see that it makes no difference.)

Question: So?

Now, we get a random variable where for close pairs the expectation is different than for far pairs.

Exercise 1: What is the magnitude of the difference in terms of ϵ with a constant factor? (Hint: it is linear in ϵ . That's the answer..justify it.)

Question: What now?

If we repeat $\Theta(\log n/\epsilon^2)$ times, and using Chernoff bounds(since the variables are 0-1) we get a test that behaves differently with high probability for the two cases!

Exercise 2: Prove that with $O(\log n/\epsilon^2)$ dimensions any two close nodes are closer than some threshold (what is the threshold) and any two far nodes are farther than this threshold with probability at least $1 - 1/n$

That is, the distance in k of these coordinates between any two nodes of distance $< (1 - \epsilon)\Delta$ is always less than some threshold, while the distance between any two nodes with distance $> \Delta$ is always greater than that threshold.

Question: What have we done?

Reduced dimension for one level of distances.

25.3 Nearest neighbor search.

Question: What does this mean for nearest neighbor search?

We can reduce the comparison to $O(\log n/\epsilon^2)$ from $O(d)$. So if d very large, we do better. But, we do still perhaps need to scan over $O(n)$ points.

Question: Can we use more space, and reduce the number of points that we need to scan?

We can store the answer for each query.

Question: You must be kidding?

Well, now the number of possible queries (at each level) is $2^{O(\log n/\epsilon^2)}$, or $n^{O(1/\epsilon^2)}$.

Thus, with polynomial space we can get $O(1)$ lookup time.

Question: Hashing? When are the random choices made?

Well, the hashing scheme is the set of random choices to make the coordinates. For each query, we make the coordinates in the previously chosen (albeit randomly chosen) fashion to do the “hashing.”