

Lecture 6 : Dimensionality Reduction

Lecturer: Kamesh Munagala

Scribe: Kamesh Munagala

In this lecture, we will consider the problem of mapping n points in a metric space with large dimension into a metric space with much smaller dimension, while preserving pairwise distances approximately.

Unbiased Estimators of Distance

The main idea comes from hashing. In standard hashing, the hash functions we used were oblivious to the relationship between different items. Can we make hash function behave in such a way that they map “nearby” objects (according to some similarity measure) into nearby buckets? Surprisingly, it turns out this can be done for many similarity (distance) measures.

Warm-up: Hamming Cube

Consider n points on the Hamming cube $\{0, 1\}^d$ in d dimensions. Suppose the distance function is the ℓ_1 -distance.

$$D(\vec{x}, \vec{y}) = \sum_{k=1}^d |x_k - y_k|$$

which simply counts the number of coordinates where the points differ. Consider now the following simple hash family:

$$\mathcal{H} = \{h_k \mid h_k(\vec{x}) = k^{\text{th}} \text{ bit of } \vec{x}\}$$

Such a hash function maps each item to one of two buckets. Let Z_{ij} denote the random variable that is d if items x_i and x_j map to different buckets and 0 otherwise, where the randomness is over the hash function chosen from \mathcal{H} . Then it is easy to check that

$$\mathbf{E}[Z_{ij}] = \frac{1}{d} \sum_{k=1}^d d \times \mathbf{1}_{h_k(x_i) \neq h_k(x_j)} = \frac{1}{d} \sum_{k=1}^d d \times |x_{ik} - x_{jk}| = D(x_i, x_j)$$

Therefore, a hash function from this family can be viewed as mapping the input points to $\{0, d\}$ so that the expected distance between the points is exactly preserved.

Of course, the expected distance does not mean much – if you choose any one hash function h_k , the n points map to one of two points, so the distance in the embedding is either 0 or d . Therefore, it is very likely *none* of the distances are preserved in this mapping – the distances are either compressed to 0 or expanded to d .

We can now use the standard trick – choose r hash functions $h_{k_1}, h_{k_2}, \dots, h_{k_p}$ at random from \mathcal{H} , and map \vec{x} to the p -dimensional point $\langle \frac{d}{p} h_{k_1}(\vec{x}), \dots, \frac{d}{p} h_{k_p}(\vec{x}) \rangle$, that is, map \vec{x} to d/p times the value of the k_m^{th} bit, for k_m being a random dimension for each $m = 1, 2, \dots, p$. By linearity of expectation applied to the previous argument, if Z_{ij} is the random variable denoting the ℓ_1 distance between the mapping of x_i and x_j , then $\mathbf{E}[Z_{ij}] = D(x_i, x_j)$. Since we are choosing the hash functions at random, it is unlikely that two input items x_i and x_j will map to the same bucket in *all* dimensions.

The question then becomes, how large should p be so that with large probability, all Z_{ij} for $i, j \in \{1, 2, \dots, n\}$ are close to their expected value? It is easy to check that this has to be as large as d . For instance, with $p = \sqrt{d}$, the smallest distance between two hashed vectors is \sqrt{d} , but it could be that these vectors were originally only distance 1 apart.

There is a moral to the above story: It is not merely sufficient to construct a hash function that in expectation preserves distance. It is also necessary to ensure the variance of this estimator is much smaller than the mean. Otherwise the trick of taking many independent copies of this hash function will end up requiring more dimensions than the initial space! However, there is a subtle point with the above hash function. It ensures that

$$\Pr[h_k(x) = h_k(y)] = 1 - \frac{D(x_i, x_j)}{d}$$

Though labeling the buckets as 0 and d and using the resulting distance introduces too much variance, surprisingly, we can use the fact that the probability of mapping to the *same* bucket depends on distance in order to beat brute force for similarity search. This will be the topic of the next lecture on LSH.

Digression: Central Limit Theorem

Before proceeding further, we will present a rough statement of the Central Limit Theorem. Let X_1, X_2, \dots, X_d be independent random variables, each with mean μ and standard deviation τ . Then under mild restrictions on higher moments of these variables, in the limit of large d , the distribution of $Z = \frac{\sum_{k=1}^d X_k}{d}$ converges to $\mathcal{N}(\mu, \sigma^2)$, where $\sigma = \frac{\tau}{\sqrt{d}}$, and $\mathcal{N}(\mu, \sigma^2)$ is the standard Normal distribution with mean μ and standard deviation σ .

What is a normal distribution? The distribution $\mathcal{N}(\mu, \sigma^2)$ has the density function

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

For a Normal distribution $Y \sim \mathcal{N}(\mu, \sigma^2)$, the deviation around the mean exponentially falls off, so that roughly speaking, assuming $k \geq 1$,

$$\Pr[|Y_i - \mu| \geq k\sigma] \leq \frac{2}{\sqrt{2\pi}} \frac{e^{-k^2/2}}{k} \leq e^{-k^2/2}$$

Therefore, the probability that the random variable deviates significantly from the mean is very small, provided we take deviations relative to the standard deviation. For instance, “six sigma” means six standard deviations from the mean, whose probability looks like $1/e^{-18} \approx 10^{-9}$, a really small number!

Euclidean Spaces

The question that arises from the above discussion is: Is there some other similarity measure in which we can construct an unbiased estimator with low variance? The answer surprisingly is “yes” for the Euclidean space. The hash family is

$$\mathcal{H} = \{h_{\vec{r}} \mid h_{\vec{r}}(\vec{x}) = \vec{r} \cdot \vec{x} \text{ and } \|\vec{r}\|_2 = 1\}$$

In other words, choose unit vector \vec{r} at random from the unit sphere; the hash value is the length of the projection of \vec{x} in the direction of \vec{r} .

Random Vectors. In order to understand what the above procedure achieves, we should first understand how to generate a random unit vector. We will show that the following process obeys a distribution that looks the same in all directions: For each coordinate $r_k, k = 1, 2, \dots, d$, generate r_k independently from a Normal distribution with mean 0 and variance 1, that is, from $\mathcal{N}(0, 1)$. This vector will not have unit norm, but the vector $\vec{r}' = \vec{r}/\|\vec{r}\|_2$ has unit norm, and points in the same direction as \vec{r} .

Let us write the closed form for the density of \vec{r} . Let $f(r_1, r_2, \dots, r_d)$ denote the density function. Since $r_1 \sim \mathcal{N}(0, 1)$, and independently $r_2 \sim \mathcal{N}(0, 1)$, and so on, the joint density is simply the product of the densities of r_1, r_2, \dots, r_d . This means

$$f(r_1, r_2, \dots, r_d) = \prod_{k=1}^d \frac{1}{\sqrt{2\pi}} e^{-r_k^2/2} = \frac{1}{(2\pi)^{d/2}} e^{-\sum_{k=1}^d r_k^2/2} = \frac{1}{(2\pi)^{d/2}} e^{-\|\vec{r}\|_2^2/2}$$

The density only depends on the length of \vec{r} and not on its direction! This means that for a given length, all directions have the same density; In other words, this process generates a vector whose direction is random.

Properties of Normal Distributions. In applying the Central Limit Theorem, we used the fact that the Normal distribution's probability of deviation from the mean falls off exponentially – the probability of being k standard deviations away drops off roughly as e^{-k^2} . In understanding the above hashing scheme, we need a different property of Normal distributions. The property is the following:

Claim 1. *If $X \sim \mathcal{N}(\mu, \sigma^2)$ and $a \geq 0$ is a constant, then $aX \sim \mathcal{N}(a\mu, a^2\sigma^2)$. Furthermore, if $Y \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $Z \sim \mathcal{N}(\mu_2, \sigma_2^2)$ are independent random variables, then $Y + Z \sim \mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$.*

The proof follows by writing out the respective density functions and checking. The key point is that taking linear combinations of independent Normal random variables yields a Normal random variable.

Unbiased Estimator. Why is the above fact relevant? Consider our hash function

$$h_{\vec{r}}(\vec{x}) = \vec{r} \cdot \vec{x} = \sum_{k=1}^d r_k x_k$$

Since $r_k \sim \mathcal{N}(0, 1)$, the first part of the above claim implies $r_k x_k \sim \mathcal{N}(0, x_k^2)$. Since the r_k 's in different dimensions are independent, the above claim also implies

$$h_{\vec{r}}(\vec{x}) = \sum_{k=1}^d r_k x_k \sim \mathcal{N}\left(0, \sum_{k=1}^d x_k^2\right) = \mathcal{N}\left(0, \|\vec{x}\|_2^2\right)$$

This shows that the hash value is a Normal distribution with zero mean, and variance equal to the squared norm of \vec{x} . For a random variable with mean zero, the expectation of the square equals the variance. (Show this!) This means:

$$\mathbf{E}[(h(\vec{x}))^2] = \|\vec{x}\|_2^2$$

where the expectation is over the choice of \vec{r} . Thus we have an unbiased estimator of the squared norm of \vec{x} : Generate a random vector \vec{r} by choosing each coordinate from a $\mathcal{N}(0, 1)$ distribution independently; take the square of the length of the projection of \vec{x} onto \vec{r} .

The properties of Normal distributions implies the above holds even for the *difference* of two vectors, so that

$$\mathbf{E} \left[(h(\vec{x}) - h(\vec{y}))^2 \right] = \mathbf{E} \left[\left(\sum_{k=1}^d r_k (x_k - y_k) \right)^2 \right] = \|\vec{x} - \vec{y}\|_2^2$$

Therefore, if we are given n points $D = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$, if we project these points onto a random vector, the squared distance between any two points is the same as the expectation of the squared difference in their projections. So far so good. But had something similar even for Hamming spaces; the problem was that the variance of the estimator there was too large for it to be useful. What about in this case?

Bounding the Variance. We perform the same trick as before: In order to reduce variance in our estimator, we choose p hash functions at random from \mathcal{H} (each of this is a random vector generated independently of the others), and map \vec{x} to the point $\Pi(\vec{x})$ in p -dimensional space:

$$\Pi(\vec{x}) = \left\langle \frac{1}{\sqrt{p}} \vec{x} \cdot \vec{r}_1, \frac{1}{\sqrt{p}} \vec{x} \cdot \vec{r}_2, \dots, \frac{1}{\sqrt{p}} \vec{x} \cdot \vec{r}_p \right\rangle$$

Each dimension of $\Pi(\vec{x})$ is distributed as $\mathcal{N}\left(0, \frac{\|\vec{x}\|_2^2}{p}\right)$, where the variance is divided by p because we scaled down each dimension by a factor of \sqrt{p} . Similarly, $\Pi(\vec{x}) - \Pi(\vec{y})$ has each of its coordinates distributed as $\mathcal{N}\left(0, \frac{\|\vec{x} - \vec{y}\|_2^2}{p}\right)$.

By the same argument as before,

$$\mathbf{E} \left[(\Pi(\vec{x}) - \Pi(\vec{y}))^2 \right] = \sum_{s=1}^p \frac{\|\vec{x} - \vec{y}\|_2^2}{p} = \|\vec{x} - \vec{y}\|_2^2$$

All we need to show is that the random variable $(\Pi(\vec{x}) - \Pi(\vec{y}))^2$, which is the sum of the squares of p independent random variables each distributed as $\mathcal{N}\left(0, \frac{\|\vec{x} - \vec{y}\|_2^2}{p}\right)$, is tightly concentrated around its expectation, for reasonable values of p .

Sums of Squares of Normals. We will need to know what an **Exponential** distribution is. This distribution has a parameter λ , and has the density function

$$f(x) = \lambda e^{-\lambda x}$$

This distribution has mean $\frac{1}{\lambda}$, and standard deviation $\frac{1}{\lambda}$.

Intuitively, think about tossing a coin with bias p till a heads is obtained. The random variable X denoting the number of failed coin tosses before success. This is a **Geometric** distribution that satisfies $\Pr[X = k] = (1 - p)p^k$. Intuitively, the **Exponential** distribution is the continuous version of the **Geometric** distribution, where you make the success probability become smaller and smaller, and squish time so there are a large number of coin tosses in a unit time interval.

What is the connection between **Exponential** and Normal distributions? Here's a math fact whose proof is tedious algebra:‘

Claim 2. *Suppose $X \sim \mathcal{N}(0, \sigma^2)$ and $Y \sim \mathcal{N}(0, \sigma^2)$ are independent random variables, then*

$$X^2 + Y^2 \sim \text{Exponential} \left(\frac{1}{2\sigma^2} \right)$$

Consider some two dimensions of $\Pi(\vec{x}) - \Pi(\vec{y})$. Let the values here be X and Y . Then, we know that $X, Y \sim \mathcal{N}\left(0, \frac{\|\vec{x} - \vec{y}\|_2^2}{p}\right)$. This means

$$X^2 + Y^2 \sim \text{Exponential}\left(\frac{p}{2\|\vec{x} - \vec{y}\|_2^2}\right)$$

This distribution has mean $\frac{2\|\vec{x} - \vec{y}\|_2^2}{p}$, and the same value as standard deviation. This is the crux – the standard deviation of our estimator of squared length is comparable to the mean. This was not true for Hamming spaces where it could have been d times larger!

The quantity $\Pi(\vec{x}) - \Pi(\vec{y})$ has squared length equal to the sum of $p/2$ independent copies of this random variable. This means its mean is $p/2$ times larger, and its standard deviation is $\sqrt{p/2}$ times larger. Furthermore, by CLT, the distribution becomes approximately normal.

This means the squared length of $\Pi(\vec{x}) - \Pi(\vec{y})$ is approximately normal with mean $\|\vec{x} - \vec{y}\|_2^2$, and standard deviation $\|\vec{x} - \vec{y}\|_2^2 \times \sqrt{\frac{2}{p}}$. Note that the standard deviation is now much smaller than the mean! Suppose we choose $p \approx 6 \log n$, where n is the number of points in our database. Then the probability that we deviate by more than $k = \sqrt{p/2}$ times the standard deviation is at most

$$e^{-k^2} = e^{-p/2} \approx \frac{1}{n^3}$$

But k times the standard deviation is at most $\|\vec{x} - \vec{y}\|_2^2$, which is at most the mean. This means that with very high probability, the squared length is at most twice the original squared length. By union bounds over all the n^2 pairs of points, all the distances are preserved to within this factor with probability at least $1 - n^2 \times 1/n^3 = 1 - 1/n$.

The argument is somewhat rough, but it is reasonably complete and can be extended to show that by choosing slightly more hash functions, the distances are in fact very close to the true value with very high probability. We finally have the following theorem, which shows that we can reduce the dimension to roughly $\log n$ while preserving distances among n input points.

Theorem 1 (Johnson-Lindenstrauss). *Given any n points in Euclidean space, for any $\epsilon > 0$, there is a mapping to $p = O\left(\frac{\log n}{\epsilon^2}\right)$ dimensions so that with probability at least $1 - \frac{1}{n}$, the distance between any pair of points is preserved to within a factor of $1 \pm \epsilon$.*

Note that the quantification in the above theorem is crucial: If we randomly project onto $O\left(\frac{\log n}{\epsilon^2}\right)$ dimensions, then with very high probability, *all* pairwise distances between n input points are preserved. This is what makes the result algorithmically useful.

Furthermore, the neat feature is that the scheme is *oblivious to the input* – for any set S of n input points, the random directions are chosen from the *same* distribution. This has applications in settings where data is scanned one input at a time. We will see another such method, the Fourier Transform, a bit later in the course. Of course, it is conceivable that a dimension reduction scheme that depends on the input points needs fewer dimensions to preserve salient properties of the data. Maybe all data lies on a 2-dimensional subspace to start with. In such a case, an oblivious scheme such as the above still requires $\log n$ dimensions, but a scheme that depends on the data could identify the subspace and project onto it. We will consider this when we discuss algebraic methods such as the PCA.