COMPSCI 532: Design and Analysis of Algorithms	November 11, 2015
Lecture 20	
Lecturer: Debmalya Panigrahi	*Scribe: Allen Xiao*

1 Overview

In this lecture, we introduce the idea of dimensionality reduction and specifically the Johnson-Lindenstrauss Lemma. The analysis we give follows that of Dasgupta and Gupta [DG03], which we recommend as a reference.

2 Dimensionality Reduction

In many applications today, data is drawn from a high-dimensional feature space, where the dimension d is incredibly high. That is, we view each data point as a vector in $\mathbb{R}^d$. A few examples of high dimensional data:

- DNA sequencing: each nucleotide in the sequence is a feature.
- Health records: various measurements like weight, blood pressure, diagnosed diseases, medications, nutrition, etc.
- Computer vision: every pixel of an image/video is a feature.

A problem with high dimensional data is that many algorithms we use to extract higher-order information (clustering, nearest-neighbors, etc.) are severely impacted by high dimension; something like n^d in the runtime is not uncommon. There are a few strategies for making high-dimensional data palatable for these algorithms:

1. Restrict to important dimensions (pick some $k \ll d$ dimensions). This is difficult to do in general, since it may not be clear which features are "important" to preserving the core characteristics of the data set.
2. Transform into a low-dimensional space, without changing pairwise distances too much, called a *low-distortion embedding* of the data. May sacrifice interpretability of the projected vectors.

Most work goes into studying the second idea. In this lecture, we present a remarkable result of the second idea by Johnson and Lindenstrauss [JL84].

3 The Johnson-Lindenstrauss Lemma

Theorem 1 (Johnson-Lindenstrauss Lemma)**.** *Let $A = a_1, \ldots, a_n$ be a set of n points in $\mathbb{R}^d$. For $k = O(\log n/\epsilon^2)$ and $\epsilon \in (0, 1)$, there exists a function $f : \mathbb{R}^d \to \mathbb{R}^k$ mapping points in $\mathbb{R}^d$ onto a k-dimensional subspace while:*

$$(1 - \epsilon)\|a_i - a_j\|^2 \leq \|f(a_i) - f(a_j)\|^2 \leq (1 + \epsilon)\|a_i - a_j\|^2$$

and $f(\cdot)$ can be found in randomized polynomial time.

Here $\|\cdot\|$ is the standard Euclidean norm. To see why this result is so powerful, notice that the dimension of the destination space is $k \ll n, d$, and that k is *independent* of d. This saves space as well, since the projected vectors require about $\log n$ space to represent rather than d. Additionally, the proof of this subspace is constructive – not only does the subspace exist, we also have an efficient procedure for finding it.

We will use a randomized procedure for finding a k-dimensional subspace which satisfies the distance-preserving property. Essentially, a random k-dimensional subspace is likely to be a satisfying one with probability at least $1/2$. By repeating this construction some $O(\log n)$ times, one can expect to find a distance-preserving subspace with high probability. Let us first go over an outline of the analysis, since it takes a few twists.

1. First, since we are interested only in the pairwise distances, let $v_{11}, v_{12}, \ldots, v_{nn}$ be the n^2 vectors:
$$v_{ij} = a_i - a_j$$
Preserving all pairwise distances $\|a_i - a_j\|$ is equivalent to preserving the vector *lengths* $\|v_{ij}\|$.

2. Instead of preserving lengths for all n^2 vectors at once, we will show that a random k-dimensional subspace (with k sufficiently large) preserves the length of *any* single vector v_{ij} with probability $1 - O(1/n^2)$. Then, take a union bound over the n^2 vectors for constant probability of success.

3. Notice that the dot product (i.e. projection) is symmetric.
$$\|\operatorname{proj}_b(a)\|^2 = \langle a, b\rangle = \langle b, a\rangle = \|\operatorname{proj}_a(b)\|^2$$
If a is a fixed vector and b is random, then the distribution of lengths is same whether we project a onto b or b onto a. Extending this argument, the distribution of lengths for a unit vector projected onto a random k-dimensional subspace is the same as the distribution of lengths of a random unit vector projected onto a fixed k-dimensional subspace. We will prove that the lengths of any v_{ij} are preserved with high probability after random projection using the second distribution.

For the concentration of lengths, we will only prove one side. The opposite side is very similar.

Lemma 2. *Let Y be a uniform random point on the unit sphere in $\mathbb{R}^d$, and let $g(Y)$ be the projection of Y onto its first $k = O(\log n/\epsilon^2)$ coordinates, then:*
$$\Pr\left(\|g(Y)\|^2 > (1+\epsilon)\frac{k}{d}\right) \le O(1/n^2)$$

Proof. A common construction for a spherically symmetric random vector is to take a vector with independent standard Gaussian coordinates, and normalize.
$$\begin{aligned} X &= (X_1, X_2, \ldots, X_d) \qquad \forall i, X_i \sim \mathcal{N}(0,1) \\ Y &= \frac{1}{\|X\|}(X_1, X_2, \ldots, X_d) \end{aligned}$$
After normalizing, Y is a uniform random point on the unit sphere. Before scaling, expected length of X is $\sqrt{d}$.
$$\mathbb{E}\left[\|X\|^2\right] = \mathbb{E}\left[\sum_{i=1}^{d} X_i^2\right] = \sum_{i=1}^{d} \mathbb{E}\left[X_i^2\right] = \sum_{i=1}^{d} 1 = d$$

For simplicity of the proof, we will use X without rescaling. Let $Z \in \mathbb{R}^k$ be the projection of X onto its first k coordinates. The expected length of Z, for the same reasons as X, is $\sqrt{k}$. The probabilistic statement in the lemma is (before scaling):

$$\Pr\left(\|Z\|^2 > (1+\epsilon)\mathbb{E}\left[\|Z\|^2\right]\right) = \Pr\left(\sum_{i=1}^{k} X_i^2 > (1+\epsilon)k\right)$$

We perform a matrix Chernoff bound to bound the probability. Many of the steps are the same as the standard Chernoff bound, and we will highlight the differences. Let $t > 0$ be a parameter we decide later.

$$\begin{aligned}
\Pr\left(\sum_{i=1}^{k} X_i^2 > (1+\epsilon)k\right) &= \Pr\left(e^{t\sum_{i=1}^{k} X_i^2} > e^{t(1+\epsilon)k}\right) \\
&\leq \frac{\mathbb{E}\left[e^{t\sum_{i=1} kX_i^2}\right]}{e^{t(1+\epsilon)k}} \\
&= \frac{\prod_{i=1}^{k}\mathbb{E}\left[e^{tX_i^2}\right]}{e^{t(1+\epsilon)k}}
\end{aligned}$$

Since X_i are normally distributed, we can compute the expectation of in the numerator using the equation for the normal distribution. We will omit the work, but for $t < 1/2$, it holds that:

$$\mathbb{E}\left[e^{tX_i^2}\right] \leq \left(\frac{1}{\sqrt{1-2t}}\right)$$

Now (assuming we choose $t < 1/2$ later) we can write the bound as:

$$\Pr\left(\sum_{i=1}^{k} X_i^2 > (1+\epsilon)k\right) \leq \left(\frac{1}{\sqrt{1-2t}}\right)^k \left(\frac{1}{e^{t(1+\epsilon)}}\right)^k$$

Choose $t = \epsilon/(2(1+\epsilon))$. This satisfies $t < 1/2$.

$$\begin{aligned}
\Pr\left(\sum_{i=1}^{k} X_i^2 > (1+\epsilon)k\right) &\leq \left(\frac{1}{\sqrt{1-2t}}\right)^k \left(\frac{1}{e^{t(1+\epsilon)}}\right)^k \\
&= \left(\frac{1}{\sqrt{1-\frac{\epsilon}{1+\epsilon}}}\right)^k \left(\frac{1}{e^{\epsilon/2}}\right)^k \\
&= \left(\frac{1}{1+\epsilon}\right)^{k/2} \left(\frac{1}{e^{\epsilon}}\right)^{k/2} \\
&= \left(e^{-\epsilon+\ln(1+\epsilon)}\right)^{k/2}
\end{aligned}$$

Using the following Taylor series bound:

$$\ln(1+\epsilon) \leq \epsilon - \epsilon^2/2 + \epsilon^3/3$$

We obtain:

$$\begin{aligned}\Pr\left(\sum_{i=1}^{k} X_i^2 > (1+\epsilon)k\right) &= \left(e^{-\epsilon+\ln(1+\epsilon)}\right)^{k/2} \\ &\leq \left(e^{-\epsilon+\epsilon-\epsilon^2/2+\epsilon^3/3}\right)^{k/2} \\ &\leq \left(e^{-\epsilon^2/6}\right)^{k/2} \\ &= e^{-\epsilon^2 k/12}\end{aligned}$$

Choose $k = 24\ln(4n)/\epsilon^2$.

$$\Pr\left(\sum_{i=1}^{k} X_i^2 > (1+\epsilon)k\right) \leq e^{-\epsilon^2 k/12} = \frac{1}{4n^2}$$

To complete the proof, notice that, while the projection of X onto its first k coordinates had expected length $\sqrt{k}$, the projection of Y onto the first k coordinates would instead have expected length $\sqrt{k/d}$. Scaling each term in $\sum_{X_i^2}$ by k/d will scale the mean by the same quantity, therefore the statement above is equivalently:

$$\Pr\left(\|g(Y)\|^2 > (1+\epsilon)\frac{k}{d}\|Y\|^2\right) \leq \frac{1}{4n^2}$$

□

Lemma 3. *Let S be a random k-dimensional subspace of $\mathbb{R}^d$, and for any $v \in \mathbb{R}^d$, let $h(v)$ be its projection onto S. For any set of n^2 points $v_{ij} \in \mathbb{R}^d$, their squared lengths are all preserved within $(1 \pm \epsilon)k/d$ factor with probability at least $1/2$.*

Proof. Begin with the result of the previous lemma. The probability that any single v_{ij} fails on either $\pm$ side is is no more than $1/(2n^2)$. By the union bound, the probability that at least one fails is no more than:

$$\sum_{i,j} \frac{1}{2n^2} = \frac{n^2}{2n^2} = \frac{1}{2}$$

Therefore, the probability that none fail is at least $1/2$. □

Corollary 4. *Let S be a random k-dimensional subspace of $\mathbb{R}^d$, and for any $v\mathbb{R}^d$, let $h(v)$ be its projection onto S. The function:*

$$f(v) = \sqrt{\frac{d}{k}}h(v)$$

satisfies the requirements of the main theorem with probability at least $1/2$.

To complete our analysis, we invoke the probabilistic argument. Any event with positive probability has an outcome in which it occurs.

Corollary 5. *By the probabilistic argument, there exists a function $f(\cdot)$ which satisfies the requirements of the main theorem for any set of n points in $\mathbb{R}^d$.*

Again, repeating the one-time procedure some $O(\log n)$ times gives high probability of finding a satisfying $f(\cdot)$. The procedure we described to derive $f(\cdot)$ takes randomized polynomial time.

Corollary 6. *$f(\cdot)$ satisfying the statement of the theorem can be found in randomized polynomial time.*

3.1 Dimension reduction for other distances

The Johnson-Lindenstrauss lemma states the existence low-distortion embedding for data in $\mathbb{R}^d$ under the Euclidean norm (ℓ_2), using only $O(\log n/\epsilon^2)$ dimensions. A natural question to ask is whether this result, or ones like it, exist for other distance functions (for example, other ℓ_p norms). A result by Brinkman and Charikar [BC05] answers this question in the negative: under Manhattan distance (the ℓ_1 norm), one can construct examples of n points which requires $\Omega(\text{poly}(n))$ dimension for constant distortion of ℓ_1 distances, versus the $O(\log n)$ given by JL.

4 Applications of Dimensionality Reduction

In this final section, we discuss some problems which use dimensionality reduction in their solutions. These are: data streaming (specifically the result by Alon, Matias, and Szegedy [AMS96]), and approximate nearest neighbors (using the algorithm by Har-Peled [HP01]).

4.1 Data streaming

In the data stream model, data x_i arrive in an online fashion and our goal is to maintain a data structure to answer a fixed query on the data stream so far, while minimizing space and update time. For this section, let us consider $x_i \in \{1, 2, \ldots n\}$, for m samples $x_1, \ldots, x_m$. For an idea of the kind of situation this could model, suppose we are a router on a network with n senders who send a total of m packets, and we wish to compute network statistics (like who sent the most packets).

In order to compute these statistics, a naive solution would be to maintain a histogram (collection of counters) over all n possible data values. Each time $x_i = a$ arrives, we increment the counter for a by 1. This solution takes $O(1)$ update time and $O(n \log m)$ space to store, for n counters which count up to m. Our goal will be to specialize this "histogram" data structure in order to compute a *specific* statistic using only logarithmic space. The construction will depend on the stastitic we choose.

We can represent the histogram after the i-th data point as a vector of counts $\mathbf{v}_i$ (initially, $\mathbf{v}_0 = 0$).

$$\mathbf{v}_i = \mathbf{v}_{i-1} + \mathbf{e}_{x_i}$$

Where $\mathbf{e}_j$ is the indicator vector with 1 in the j-th entry and 0 everywhere else. One common statistic to compute is the ℓ_p norm of the counts so far.

$$\|\mathbf{v}_i\|_p = \left(\sum_{j=1}^{n} |v_{ij}|^p \right)^{1/p}$$

Some choices of p which appear often are:

- ℓ_1 norm: The sum of absolute values of entries.

- ℓ_2 norm: The sum of squares of entries (Euclidean norm).
- ℓ_∞ norm: The maximum entry of the vector. To see why, notice that as $p \to \infty$, the maximum entry of $\mathbf{v}_i$ is represented more and more in the sum.

In this example, we will maintain the ℓ_2 norm using logarithmic space. For simplicity of notation, we will use the square of the ℓ_2 norm.

$$\|\mathbf{v}_i\|_2^2 = \sum_{j=1}^{n} |v_{ij}|^2$$

We will try to approach this problem by applying the JL lemma, projecting the m points down into a $k \approx \log n$ dimensional space.

Let $R_{k\times n}$ be the projection matrix from $\mathbb{R}^n$ into a random subspace $\mathbb{R}^k$. Recall from the JL lemma construction that we can design such an R as:

$$R = \sqrt{\frac{n}{k}} \begin{bmatrix} \mathcal{N}(0,1) & \cdots & \mathcal{N}(0,1) \\ \vdots & \ddots & \vdots \\ \mathcal{N}(0,1) & \cdots & \mathcal{N}(0,1) \end{bmatrix}_{k\times n}$$

Instead of maintaining $\mathbf{v}_i$, we will maintain $\mathbf{u}_i = R\mathbf{v}_i$, which is only dimension k. This type of procedure, which maintains a linear function of $\mathbf{v}_i$, is called a *linear sketch*. Since R is a linear operator, updating $\mathbf{u}_i$ is straightforward:

$$\begin{aligned} \mathbf{u}_i &= R\mathbf{v}_i \\ &= R(\mathbf{v}_{i-1} + \mathbf{e}_{x_i}) \\ &= R\mathbf{v}_{i-1} + R\mathbf{e}_{x_i} \\ &= \mathbf{u}_{i-1} + R_{x_i} \end{aligned}$$

Here, R_{x_i} is the x_i-th column of R. By the JL lemma, $\mathbf{u}_i$ should preserve the ℓ_2 norm of every projected vector, if k is sufficiently large.

$$\Pr\left(\|R\mathbf{v}\|_2^2 \in (1 \pm \epsilon)\|\mathbf{v}\|_2^2\right) \geq 1 - \exp(-O(\epsilon^2 k))$$

Once more, choosing $k = O(\log n/\epsilon^2)$ gives high probability of the projection approximately preserving the ℓ_2 norm. The bit size of the new histogram $\mathbf{u}_i$ is $O(k \log m) = O(\log n \log m/\epsilon^2)$. The update seems to be $O(k)$ time (vector addition), if we know R.

This is a problem however: R has size $k \times n$, which means storing R explicitly will use poly(n) space. Notice that we only require a single column of R each iteration; if we could *generate* the columns of R, we would be fine. Note that it is insufficient to sample new random column vector, since we need the columns of R to be consistent across iterations (if R_j is requested on different iterations, they should be the same).

Re-generating the columns of R is possible using a class of functions called *pseudorandom generators* (PRG), which are beyond the scope of this class, but have strong implications in complexity theory and derandomization. Briefly, a PRG takes a small number of random bits (a *seed*) to generate a larger number of random bits which "fool" a class of algorithms. That is, to this class of algorithms, the PRG output appears to be from a random source, even though the PRG itself only

used a relatively small amount of randomness. Often, the proof will require the class of algorithms to be restricted, e.g. run quickly, or use little space.

Luckily for us, a pseudorandom generator by Nisan [Nis90] provably fools algorithms which use logarithmic space and read their input only once (our streaming algorithm is one of these). Instead of storing R explicitly, we will seed Nisan's PRG and use it to deterministically generate the "random" bits for the j-th column. After each use, we can restart the PRG with the same seed and generate the same sequence of random bits. Instead of storing R, we need only store the seed which is $O(\log n)$ in size.

To conclude, this streaming result introduced the streaming model, and its authors were awarded the 2005 Gödel prize for their work. The original analysis did not use an application of the JL lemma, and instead a probabilistic analysis of random ± 1 vectors. The results are similar, however: If the number of ± 1 vectors is at least $\Omega(\log n/\epsilon^2)$, one could obtain a $(1 \pm \epsilon)$ approximation with high probability.

4.2 Nearest neighbor search

Suppose we have a fixed set of m points $p_i \in \mathbb{R}^n$. In the nearest neighbor search (NNS) problem, we are given a query point $q \in \mathbb{R}^n$, and wish to report the p_i minimizing the Euclidean distance to q among all p_i.

$$\min_{1 \le i \le n} \|q - p_i\|_2 = \min_{1 \le i \le n} \sqrt{\sum_{j=1}^{n} (p_{ij} - q_j)^2}$$

Again, we wish to maintain a data structure which makes this query fast. A naive solution is to simply check the distance between q and every p_i, but this takes time $O(mn)$. We will show that, by applying the JL lemma, we can approximate the closest p_i with a logarithmic query time. The approximate problem is called ϵ-NNS, where the returned p_i must be within $(1+\epsilon)$ the minimum. We will solve ϵ-NNS using an equivalent problem called *point location in equal balls* (PLEB).

The PLEB problem is roughly as follows. Suppose we have a fixed set of points $p_1, \ldots p_m \in \mathbb{R}^n$ and a fixed radius r. Given a query point q, return a p_i such that q lies in the ball of radius r about p_i. If no such p_i exists, return NO.

The approximation version of PLEB is called ϵ-PLEB. Formally, let $B(p_i, r) = \{x \in \mathbb{R}^n \mid \|x - p_i\|_2 \le r\}$, the ball of radius r about p_i. Given a query point q, we would like to report:

1. If $\exists i$ such that $q \in B(p_i, r)$, then report YES and output the satisfying p_i.
2. Otherwise, if $q \notin B(p_i, (1+\epsilon)r)$ for all p_i, then report NO.
3. Otherwise, either report NO, or YES with a p_i such that $q \in B(p_i, (1+\epsilon)r)$.

In particular, if q is not in any of the radius r ball but does lie in some radius $(1+\epsilon)r$ ball, it is fine to return one of those balls or fail. If it is in one of the radius r balls or in none of them, we must return the same result as PLEB.

To solve ϵ-PLEB, we will use a refinement of the space. We will divide $\mathbb{R}^n$ into hypercubes of side length $s = \epsilon r/\sqrt{n}$. The longest distance within one of these hypercubes is the diameter, which is at most:

$$\sqrt{n} \cdot s = \epsilon r$$

We label each cube C with a p_i where $B(p_i, r)$ which intersects C (perhaps none), and maintain the paired (C, p_i) in a hash table. When a query comes in for point q:

1. Locate the hypercube C containing q. This can be done in $O(n)$ time by examining the coordinates of q.

2. If C was labeled with a p_i, then by our argument above, q must be within $r + \epsilon r = (1+\epsilon)r$ of p_i, therefore we can return p_i.

3. If C is not labeled, then no radius r ball intersects C, therefore q is definitely within no r-ball and it is safe to return NO.

The query time is $O(n) + O(1) = O(n)$ (time to locate C, plus time to check the hash table). How large is the data structure? To bound the size of the hash table, we need to bound the number of cubes which can intersect the all m balls. For this, we use a simple volume argument. The volume of a n-dimensional sphere of radius r is approximately:

$$\frac{2^{O(n)} r^n}{n^{n/2}}$$

The volume of each of our hypercubes is exactly:

$$\left(\frac{\epsilon r}{\sqrt{n}}\right)^n$$

So the number of intersected cubes is no more than:

$$\frac{2^{O(n)} r^n / n^{n/2}}{\epsilon r/\sqrt{n}} = \left(\frac{O(1)}{\epsilon}\right)^n$$

At this point, we apply the JL lemma to reduce the data structure size to polynomial in n. Let the target dimension be $k = O(\log m/\epsilon^2)$, then by the JL lemma, we can preserve distances between all m points p_i with high probability. If the number of query points is within $\text{poly}(m)$, the distances to all q are also preserved with high probability. The space complexity of the hash table becomes $n^{O(\log(1/\epsilon)/\epsilon^2)}$. Additionally, projecting each query point raises the query time to $O(n \log n/\epsilon^2)$.

Finally, we must solve ϵ-NNS using ϵ-PLEB. First, scale all distances to be bounded within $[1, M]$. We will use a set of $O(\log_{1+\epsilon} M)$ ϵ-PLEB of radii:

$$r_t = (1+\epsilon)^{t-1}$$

For a single ϵ-NNS query, we report the ϵ-PLEB with the smallest r which reports YES. We can do this logarithmic time by binary search. The final query time is $O(n \log\log_{1+\epsilon} R)$, while the combined size of all data structures is $O(n^{\log(1/\epsilon)/\epsilon^2} \log_{1+\epsilon} R)$.

References

[AMS96] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM Symposium on Theory of Computing*, pages 20–29. ACM, 1996.

[BC05] Bo Brinkman and Moses Charikar. On the impossibility of dimension reduction in ℓ_1. *Journal of the ACM (JACM)*, 52(5):766–788, 2005.

[DG03] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms*, 22(1):60–65, 2003.

[HP01] Sariel Har-Peled. A replacement for Voronoi diagrams of near linear size. In *FOCS*, page 94. IEEE, 2001.

[JL84] William Johnson and Joram Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Math*, 26, 1984.

[Nis90] N Nisan. Using hard problems to create pseudorandom generators, acm distinguished dissertation, 1990.