Algorithms CS 5312

Dimensionality Reduction - Johnson-Lindenstrauss Lemma

Imdad Ullah Khan

Contents

1 Dimensionality Reduction

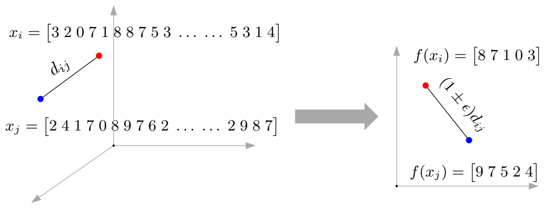
Having cursed dimensionality enough, now we finally start to do something about it. While we discussed many issues with large dimensions, we focus on computational aspect of the curse such as processing time, storage capacity and communication bandwidth. Since it is hard to work with high dimensional vectors (perhaps in tens or hundreds of thousands), for computational efficiency and quality of analytics, we would like to reduce the number of dimensions to a manageable number (in hundreds perhaps) without "distorting" the data too much. The specific definition and measure of distortion depends on the given data set and the target application. While there could be other objectives for dimensionality reduction, our goal is to reduce dimensionality of the dataset, while preserving pairwise distance.

In this course we will study many techniques for dimensionality reduction, namely, the Johnson-Lindenstrauss transform and (it's variations), the AMS transform (that is originally meant for something different), Locality Sensitive Hashing, and Principal Component Analysis. Before that, let's look at some similar problems.

1.1 Linear Dimensionality Reduction

Problem 1. *Given a point set $\mathcal{X} = \{x_1, \dots, x_n\} \subset \mathbb{R}^m$, Find a dimensionality reduction function $f : \mathbb{R}^m \mapsto \mathbb{R}^k$, $k \ll m$ such that $d(f(x_i), f(x_j)) \simeq d(x_i, x_j)$, i.e.*

$$\forall\, x_i, x_j \in \mathcal{X} \quad (1-\epsilon)d(x_i, x_j) \;\leq\; d(f(x_i), f(x_j)) \;\leq\; (1+\epsilon)d(x, y)$$



$x_i = [3\,2\,0\,7\,1\,8\,8\,7\,5\,3 \ldots \ldots 5\,3\,1\,4]$
d_{ij}
$x_j = [2\,4\,1\,7\,0\,8\,9\,7\,6\,2 \ldots \ldots 2\,9\,8\,7]$
$f(x_i) = [8\,7\,1\,0\,3]$
$(1 \pm \epsilon)d_{ij}$
$f(x_j) = [9\,7\,5\,2\,4]$

Figure 1

Figure 1 shows two points in a high dimensional space reduced to a 5 dimensional space.

If f is a linear function, it is called linear diensionality reduction and f can be represented by a linear transformation A, i.e. $f(\mathcal{X}) = A\mathcal{X}$, $\mathcal{X}$: the $n \times m$ data matrix with each $x_i \in \mathcal{X}$ as a row.

Note that dimensionality reduction is a special case of low distortion embedding since the distance measure d is the same in both domain and co-domain, i.e. the given and required distance measure is the same. Dimensionality reduction is, however, different from data compression since it does not require that $x \simeq f(x)$, but only that $d\big(f(x_i), f(x_j)\big) \simeq d(x_i, x_j)$ as shown in Figure 2.

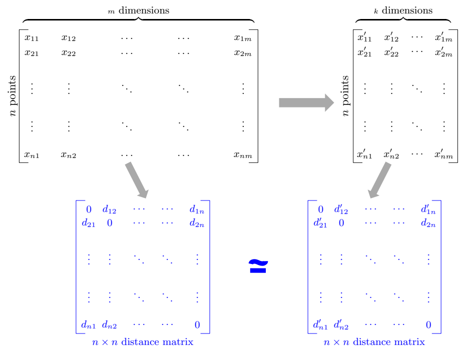


Figure 2

While the specific method of dimensionality reduction depends on the objective, all dimensionality reduction methods can be divided into two broad categories (as shown in Figure **??**)

1. Feature Selection: Select a few variables that are the *most relevant* and discard the rest
2. Feature Extraction - Create new features from data, which are usually are linear or non-linear combination of old ones. The objective is least reconstruction error or maximum inter-class separation.

2 Projection

Let $\vec{v}$ be a unit vector, let l be a line in the direction of $\vec{v}$ and let p be the point on l that is closest to a vector (point). The line connecting $\vec{u}$ to p is perpendicular to $\vec{v}$, otherwise p will not be the closest point (by the Pythagoras theorem). The point (vector) p is called the *projection* of $\vec{u}$ on $\vec{v}$. Finding projection of $\vec{u}$ on the standard basis vectors is easy as shown in Figure 3.

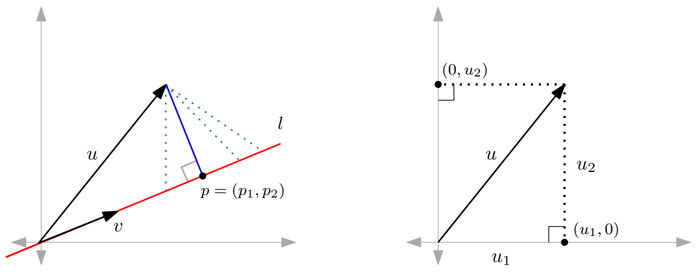


Figure 3

For general vectors we derive it from dot product. Note that p is just scaled vector $\vec{v}$, i.e. $p = a\,\vec{v}$. We need to find the scalar a. Since $\vec{u} - p = \vec{u} - a\vec{v}$ is perpendicular on $\vec{v}$, $\vec{v} \cdot (\vec{u} - a\,\vec{v}) = 0$. Hence, $\vec{v} \cdot \vec{u} - \vec{v} \cdot a\,\vec{v} = \vec{v} \cdot \vec{u} - a\,\vec{v} \cdot \vec{v} = 0$ which means $a\,\vec{v} \cdot \vec{v} = \vec{v} \cdot \vec{u}$ and therefore $a = \dfrac{\vec{v} \cdot \vec{u}}{\vec{v} \cdot \vec{v}} = \dfrac{\vec{v} \cdot \vec{u}}{\|\vec{v}\|}$

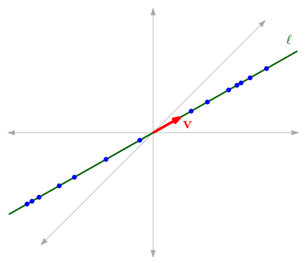


Figure 4

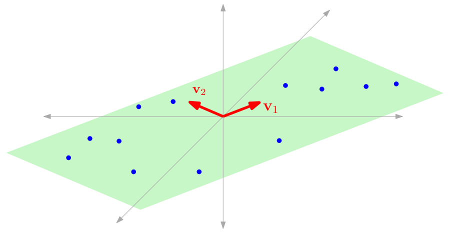


Figure 5

Suppose the m-D data lies on the line ℓ and let $\vec{v}$ be the unit vector in direction of ℓ as shown in Figure 4. For $\vec{x_i} \in \mathcal{X}$, let $f(\vec{x_i}) := \vec{v} \cdot \vec{x_i}$.

In this case, since $\vec{v} \cdot \vec{x}_i = \vec{x}_i$ (as $\vec{x}_i$ lies on ℓ), we get

$$\forall i,\, j \quad \|f(x_i) - f(x_j)\| = \|\vec{v} \cdot \vec{x}_i - \vec{v} \cdot \vec{x_j}\| = \|\vec{x}_i - \vec{x}_j\|$$

Now suppose the m-d data lies on a plane with orthonormal basis $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k$ as in Figure 5. Let $\vec{V}$ be the matrix with $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_k$ as columns. For $\vec{x_i} \in X$, let $f(\vec{x_i}) := \vec{x_i}\,\vec{V}$. Then,

$$\forall i,\, j \quad \|f(x_i) - f(x_j)\| = \|\vec{x}_i\vec{V} - \vec{x}_j\vec{V}\| = \|\vec{x}_i - \vec{x}_j\|$$

In both these cases,there is 0 error dimensionality reduction, i.e. without any distortion. However, the problem is that we do not know $\vec{V}$.

It is possible to find the low dimensional space to which the data is close by, with an approach similar to (multiple) linear regression. However, the error here would be perpendicular distance

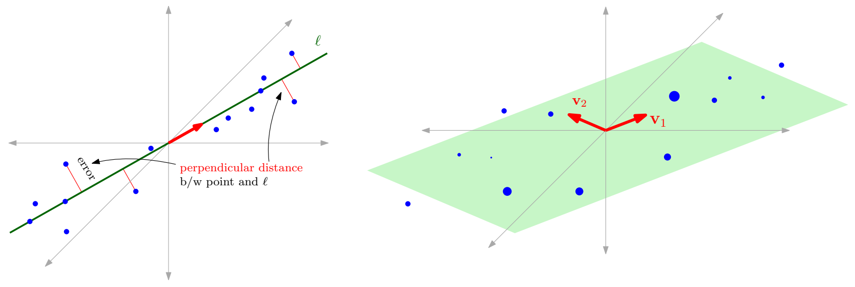


Figure 6

not vertical distance as shown in Figure 6. Secondly, whereas the goal in linear regression is to minimize SSE, in dimensionality reduction it would be to minimize pairwise distances .

This approach can be taken with modified goals and is called *Principal Component Analysis* (PCA) which is a form of feature extraction and is data dependent dimensionality reduction.

3 The Johnson-Lindenstrauss Lemma

The correct answer is a classic result in functional analysis - the Johnson-Lindenstrauss Lemma. Its statement is as follows:

Theorem 1. *Given* $\mathcal{X} = \{x_1, \dots, x_n\} \subset \mathbb{R}^m$. *For* $\epsilon \in (0, 1/2)$, *there exists a linear map* $f : \mathbb{R}^m \to \mathbb{R}^k$, $k = c \log n/\epsilon^2$ *such that for any* $x_i, x_j \in \mathcal{X}$

$$(1-\epsilon)\|x_i - x_j\|_2 \le \|f(x_i) - f(x_j)\|_2 \le (1+\epsilon)\|x_i - x_j\|_2$$

Note that this lemma works only for ℓ_2 distance. This lemma has found many applications in algorithms. For example, both of our generic proximity computation problems have reduced time complexity. The distance matrix can now be computed in $O(n^2 \frac{\log n}{\epsilon^2})$ instead of $O(n^2 \frac{\log n}{\epsilon^2})$ and the nearest neighbor computation can be done in $O(n \frac{\log n}{\epsilon^2})$ instead of $O(nm)$.

3.1 Proof

A constructive proof of Johnson-Lindenstrauss lemma is as follows.

Suppose we project $\mathcal{X}$ onto k random directions. In order to do, choose k random unit vectors $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_k \in \mathbb{R}^m$. Let $\mathcal{V}$ be the $m \times k$ matrix with $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_k$ as columns. Each row of $\mathcal{XV}$ is the reduced dimensional version of x_i as shown in Figure 7

Recall our discussion on generating random unit vectors $\vec{v} = (\mathcal{N}(0,1), \mathcal{N}(0,1), \dots, \mathcal{N}(0,1))$ consisting of m-coordinates, normalized by $\|v\|$ is a provably random unit vector (a point on the surface of the unit m-ball). We also discussed that the more discrete version $\vec{v} \in [-1,1]^m$ is a good enough approximation of a random unit vector. Recall approximately generating unit directions towards corners of the m-cubes $[-1,1]^m$. For $m \gg 1$, these 2^m directions approximately

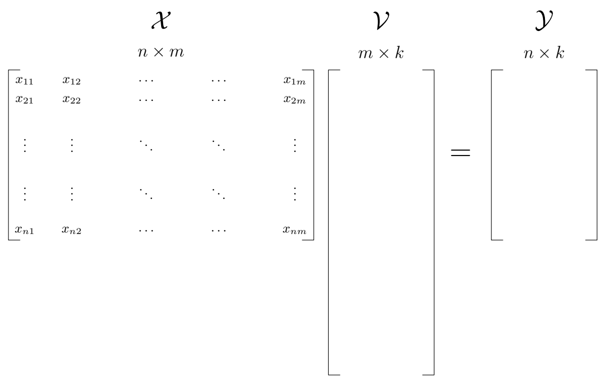

$\mathcal{X}$
$n \times m$
$\mathcal{V}$
$m \times k$
$\mathcal{Y}$
$n \times k$
x_{11} x_{12} $\cdots$ $\cdots$ x_{1m}
x_{21} x_{22} $\cdots$ $\cdots$ x_{2m}
x_{n1} x_{n2} $\cdots$ $\cdots$ x_{nm}
$=$

Figure 7

cover surface of m-ball [1]. Hence, we give the sketch of the constructive proof of JL-Lemma by projecting on such random unit vectors.

Suppose we project two vectors $x, y \in \mathbb{R}^m$ in a random direction, i.e. take their dot products with a randomly generated direction $v \in \{-1, 1\}^m$. For $\vec{x} \in \mathcal{X}$ let $f_v(\vec{x}) = \langle \vec{x}, \vec{v} \rangle = \vec{x} \cdot \vec{v} = \sum_{i=1}^{m} \vec{x}_i \vec{v}_i$. Then,

$$E\big[(f_v(\vec{x}) - f_v(\vec{y}))^2\big] = E\Bigg[\Bigg(\sum_{i=1}^{m} \vec{v}_i\vec{x}_i - \sum_{i=1}^{m} \vec{v}_i\vec{y}_i\Bigg)^2\Bigg] = E\Bigg[\sum_{i=1}^{m} \vec{v}_i^2(\vec{x}_i - \vec{y}_i)^2\Bigg] = \sum_{i=1}^{m}(\vec{x}_i - \vec{y}_i)^2 E[\vec{v}_i^2]$$

Note that dimensionality of $\vec{x}$ and $\vec{y}$ is reduced to only 1, and thus we get an unbiased estimator of the square of Euclidean distance. This means that since $E[\vec{v}_i^2] = 1$, on expectation, the squared ℓ_2 distance between $f_v(\vec{x})$ and $f_v(\vec{y})$ is the same as the squared ℓ_2 distance between $\vec{x}$ and $\vec{y}$, i.e. $E\big[\|f_v(\vec{x}) - f_v(\vec{y})\|^2\big] = \|\vec{x} - \vec{y}\|^2$

3.2 Enhancing Results

There are two major issues with the above result

1. We want to preserve distances almost surely, not in expectation only
2. We want guarantee on distances not squared distances, i.e. If $E[X^2] = \mu^2$, it does not necessarily mean that $E[X] = \mu$. For example, if $X = 0$ with probability $1/2$ and 2 otherwise, then $E[X] = 1$ but $E[X^2] = 2$ and $\sqrt{2} \neq 1$.

Both of these problems can be resolved with probability amplification or repeated independent trials, i.e. we take average of k independent random projections. We choose k unit vectors $v^1, v^2, \ldots, v^k$ (each approximately a random direction), where each coordinate of v^i is chosen uniformly at random from $\{-1, 1\}^m$ and scaled by $1/\sqrt{k}$.

For $\vec{x} \in \mathcal{X}$, let $f(\vec{x}) = (f_{v^1}(x), f_{v^2}(x), \ldots, f_{v^k}(x))$ i.e. $f(x)$ is a k-dimensional vector, where the ith coordinate is the projection of $\vec{x}$ onto v^i and $f(\vec{x})[i] = \vec{x} \cdot \vec{v}^i$. Then,

$$E(\|f(\vec{x}) - f(\vec{y})\|^2) = E\Bigg(\sum_{j=1}^{k} \left(f_{v^j}(\vec{x}) - f_{v^j}(\vec{y})\right)^2\Bigg) = \sum_{j=1}^{k} E\Bigg(\sum_{i=1}^{m} (\vec{v}_i^j)^2 (\vec{x}_i - \vec{y}_i)^2\Bigg)$$

$$E\left(\sum_{i=1}^{m} (\vec{v}_i^j)^2 (\vec{x}_i - \vec{y}_i)^2\right) = \sum_{i=1}^{n} (\vec{x}_i - \vec{y}_i)^2 E((\vec{v}_i^j)^2) = \frac{\|x-y\|^2}{k}$$

$$\text{Thus} \qquad E[\|f(x) - f(y)\|^2] = \|x-y\|^2$$

In expectation, mapping f preserves the squared ℓ_2 distance between a pair. Next, we will show that with high probability, the squared ℓ_2 distance in reduced dimensions is concentrated around its mean. For this, we use the Hoeffding's inequality (where the interval for X_j is hidden in the constants).

There exists some constant c_1, such that $Pr\left(\|f(x) - f(y)\|^2 \geq (1+\epsilon)\|x-y\|^2\right) \leq e^{-c_1\epsilon^2 k}$

Similarly, we get a bound on the lower tail (as in the last set of notes). There exists some constant c_2 such that $Pr\left(\|f(x) - f(y)\|^2 \leq (1-\epsilon)\|x-y\|^2\right) \leq e^{-c_2\epsilon^2 k}$

Hence, there is some constant c such that

$$Pr\left((1-\epsilon)\|x-y\|^2 \leq \|f(x) - f(y)\|^2 \leq (1+\epsilon)\|x-y\|^2\right) \geq 1 - e^{-c\epsilon^2 k}$$

We choose k such that $e^{-c\epsilon^2 k} < \frac{1}{n^3}$, so when union bound is applied, the probability that some pair is 'bad' is at most $\frac{1}{n}$. This means that for $k \geq {}^1/_{\epsilon^2}(\log(n) + \log(1/c))$, with high probability of $1 - {}^1/_n$, squared ℓ_2 distance is preserved for all pairs.

3.3 Remarks

1. The original Johnson-Lindenstrauss lemma actually used each coordinate of all the vectors r^j's to be random number from $\mathcal{N}(0,1)$ (actually it was even different than that).

2. The dimensionality of the resulting space, k, is ${}^1/_{\epsilon^2}(\log(n) + \log(1/c))$.

3. k does not depend on m (the original dimensions) and depends on n only

4. $k \propto \epsilon$ (the error margin), i.e. as we require less error, k would naturally grow.

5. It is known that ohnson Lindenstrauss lemma is essentially the best one can do in terms of linear maps for dimensionality reduction.[4]. In fact, even other maps can't do much better [3].

6. The matrix $\mathcal{V}$ can be precomputed since this technique is data oblivious. Note that $\mathcal{V}$ does not depend on the data at all and only the number of columns is a function of the number of original points (n).

7. Storing the matrix $\mathcal{V}$ is not necessary. As we will discuss in streaming algorithms, it can be generated it using a random number generator with fixed seeds or hash functions.

8. The Johnson-Lindenstrauss lemma works only for the ℓ_2 (Euclidean) distance and this technique of random projections may not work for other distance measures. In fact, it is known that, in order to preserve the ℓ_1 (Manhattan) distance within $(1 \pm \epsilon)$ error, the number of dimensions required k (in the reduced dimensional space) is at least $n^{1/2 - O(\epsilon \log(1/\epsilon))}$. [2]

References

[1] Dimitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003. Special Issue on PODS 2001.

[2] Bo Brinkman and M. Charikar. On the impossibility of dimension reduction in l_1. pages 514–523, 11 2003.

[3] K. G. Larsen and J. Nelson. Optimality of the johnson-lindenstrauss lemma. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 633–638, 2017.

[4] Kasper Green Larsen and Jelani Nelson. The johnson-lindenstrauss lemma is optimal for linear dimensionality reduction. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 82:1–82:11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.