Dimensionality Reduction

Steve Avsec

Illinois Institute of Technology

April 15, 2024

Overview

Johnson-Lindenstrauss Lemma

Given $0 < \varepsilon < 1$ and m points in $\mathbb{R}^N$, and an integer $n > \frac{C \log(m)}{\varepsilon^2}$, there exists a linear transformation $A : \mathbb{R}^N \to \mathbb{R}^n$ such that

$$(1+\varepsilon)^{-1} \|A\mathbf{x} - A\mathbf{y}\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_2 \leq (1-\varepsilon)^{-1} \|A\mathbf{x} - A\mathbf{y}\|_2$$

Johnson-Lindenstrauss Lemma

Given $0 < \varepsilon < 1$ and m points in $\mathbb{R}^N$, and an integer $n > \frac{C\log(m)}{\varepsilon^2}$, there exists a linear transformation $A : \mathbb{R}^N \to \mathbb{R}^n$ such that

$$(1+\varepsilon)^{-1}\|A\mathbf{x} - A\mathbf{y}\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_2 \leq (1-\varepsilon)^{-1}\|A\mathbf{x} - A\mathbf{y}\|_2$$

Here ε is controlling the amount of ”distortion” that is allowed with 0 representing no distortion and 1 a lot of distortion.

Johnson-Lindenstrauss Lemma

Given $0 < \varepsilon < 1$ and m points in $\mathbb{R}^N$, and an integer $n > \frac{C \log(m)}{\varepsilon^2}$, there exists a linear transformation $A : \mathbb{R}^N \to \mathbb{R}^n$ such that

$$(1+\varepsilon)^{-1} \|A\mathbf{x} - A\mathbf{y}\|_2 \leq \|\mathbf{x} - \mathbf{y}\|_2 \leq (1-\varepsilon)^{-1} \|A\mathbf{x} - A\mathbf{y}\|_2$$

Here ε is controlling the amount of "distortion" that is allowed with 0 representing no distortion and 1 a lot of distortion.

Notice that the ambient dimension N does not factor into the dimension being projected onto.

How to construct A

Originally: Let $\mathbf{u}_1, \ldots, \mathbf{u}_n$ be iid random vectors pulled from $\mathcal{N}(0, I_N)$.

How to construct A

Originally: Let $\mathbf{u}_1, \ldots, \mathbf{u}_n$ be iid random vectors pulled from $\mathcal{N}(0, I_N)$.

Let
$$P = \sum_{j=1}^{n} u_j u_j^t.$$

P is by expectation an orthogonal projection (e.g. $P = P^t$ and $E[P^2] = E[P]$).

How to construct A

Originally: Let $\mathbf{u}_1, \ldots, \mathbf{u}_n$ be iid random vectors pulled from $\mathcal{N}(0, I_N)$.

Let

$$P = \sum_{j=1}^{n} u_j u_j^t.$$

P is by expectation an orthogonal projection (e.g. $P = P^t$ and $E[P^2] = E[P]$).

Using a well-known probability phenomenon called "concentration of measure", there's a non-zero probability that each pairwise distance is distorted by a constant factor c. So then $A = P/c$.

How to construct A

Originally: Let $\mathbf{u}_1, \ldots, \mathbf{u}_n$ be iid random vectors pulled from $\mathcal{N}(0, I_N)$.

Let

$$P = \sum_{j=1}^{n} u_j u_j^t.$$

P is by expectation an orthogonal projection (e.g. $P = P^t$ and $E[P^2] = E[P]$).

Using a well-known probability phenomenon called "concentration of measure", there's a non-zero probability that each pairwise distance is distorted by a constant factor c. So then $A = P/c$.

If your first draw is unsuccessful, redraw with a new P.

Extensions

There's nothing particularly special about $\mathcal{N}(0, I_N)$ random vectors. What matters is that the vectors have expected length 1, expectation 0, and covariance 0.

Extensions

There's nothing particularly special about $\mathcal{N}(0, I_N)$ random vectors. What matters is that the vectors have expected length 1, expectation 0, and covariance 0.

The constant C changes depending on the probabilistic model chosen. For normal r.v., this C is known to be 8.

Extensions

There's nothing particularly special about $\mathcal{N}(0, I_N)$ random vectors. What matters is that the vectors have expected length 1, expectation 0, and covariance 0.

The constant C changes depending on the probabilistic model chosen. For normal r.v., this C is known to be 8.

C increases as the 4th moment of the random vectors increases. One common choice is to create random vectors using rvs like

$$X = \begin{cases} -\frac{1}{\sqrt{2pN}} & \text{with probability } p \\ 0 & \text{with probability } 1-2p \\ \frac{1}{\sqrt{2pN}} & \text{with probability } p \end{cases}$$

Continued

N-dimensional vectors whose components are iid copies of X make good JL projections, and as a bonus can be made sparse.

Continued

N-dimensional vectors whose components are iid copies of X make good JL projections, and as a bonus can be made sparse.

The fourth moment of X is given by

$$E[X^4] = 2\left(\frac{1}{\sqrt{2pN}}\right)^4 = \frac{1}{2p^2N^2}$$

so the sparser the matrix, the larger the 4th moment and larger n has to be.

Continued

N-dimensional vectors whose components are iid copies of X make good JL projections, and as a bonus can be made sparse.

The fourth moment of X is given by

$$E[X^4] = 2\left(\frac{1}{\sqrt{2pN}}\right)^4 = \frac{1}{2p^2N^2}$$

so the sparser the matrix, the larger the 4th moment and larger n has to be.

There are many improvements and extensions of JL, but the thrust is that you can do a lot of dimensionality reduction just using random projections.

A Minimization Problem

Solve the minimization problem:

$$\min \|X - X_k\|_2$$

subject to X_k of rank k.

A Minimization Problem

Solve the minimization problem:

$$\min \|X - X_k\|_2$$

subject to X_k of rank k.

Recall that

$$\|X\|_2^2 = Tr(X^t X)$$

so

$$\|X - X_k\|_2^2 = Tr(X^t X) - Tr(X^t X_k) - Tr(X_k^t X) + Tr(X_k^t X_k)$$

A Replacement

We can replace X_k with $XD_kD_k^t$ where D_k is a rank k projection (so $D_k^tD_k = I$), and we get

$$Tr(X^tX) - Tr(X^tXD_kD_k^t) - Tr(D_kD_k^tX^tX) + Tr(D_kD_k^tX^tXD_kD_k^t)$$

A Replacement

We can replace X_k with $XD_kD_k^t$ where D_k is a rank k projection (so $D_k^tD_k = I$), and we get

$$Tr(X^tX) - Tr(X^tXD_kD_k^t) - Tr(D_kD_k^tX^tX) + Tr(D_kD_k^tX^tXD_kD_k^t)$$

Since $Tr(XY) = Tr(YX)$ and $D_k^tD_k = I$, the last two terms are equal with opposite signs, so we can reduce this to:

$$Tr(X^tX) - Tr(D_k^tX^tXD_k)$$

A Replacement

We can replace X_k with $XD_kD_k^t$ where D_k is a rank k projection (so $D_k^tD_k = I$), and we get

$$Tr(X^tX) - Tr(X^tXD_kD_k^t) - Tr(D_kD_k^tX^tX) + Tr(D_kD_k^tX^tXD_kD_k^t)$$

Since $Tr(XY) = Tr(YX)$ and $D_k^tD_k = I$, the last two terms are equal with opposite signs, so we can reduce this to:

$$Tr(X^tX) - Tr(D_k^tX^tXD_k)$$

Notice that $Tr(X^tX)$ does not depend on k.

Final Redux

So our minimization is equivalent to

$$\max_{D_k} Tr(D_k^t X^t X D_k)$$

Final Redux

So our minimization is equivalent to

$$\max_{D_k} Tr(D_k^t X^t X D_k)$$

Replacing X with its SVD, we have

$$\max_{D_k} Tr(D_k^t V \Sigma^2 V^t D_k)$$

Final Redux

So our minimization is equivalent to

$$\max_{D_k} Tr(D_k^t X^t X D_k)$$

Replacing X with its SVD, we have

$$\max_{D_k} Tr(D_k^t V \Sigma^2 V^t D_k)$$

One can use some induction on k and good, old vasioned calculus to show that D_k is just the first k columns of V.

How to Tune k

The ”simple” answer is to plot the singular values of X and look for an ”elbow”.

How to Tune k

The "simple" answer is to plot the singular values of X and look for an "elbow".

What to do if there is no clear elbow: Think of X as decomposing into

$$X = S + E$$

where S stands for some structure in the data and E denotes some noise.

How to Tune k

The "simple" answer is to plot the singular values of X and look for an "elbow".

What to do if there is no clear elbow: Think of X as decomposing into

$$X = S + E$$

where S stands for some structure in the data and E denotes some noise.

The singular values of E tend to be governed by the *Marchenko-Pastur Law* which is given by

$$d\nu_\lambda(x) = C\frac{\sqrt{(\lambda_+ - x)(x - \lambda_-)}}{\lambda x}\chi_{[\lambda_-,\lambda_+]}(x)\,dx$$

where $\lambda_\pm = \sigma^2(1 \pm \sqrt{\lambda})^2$ and $\lambda \approx \frac{d}{N}$.

Some Observations

1. Where JL was entirely probabilistic and did not actually depend on the data in any way, PCA is entirely driven by the data matrix.

Some Observations

1. Where JL was entirely probabilistic and did not actually depend on the data in any way, PCA is entirely driven by the data matrix.

2. These are two ends of a spectrum with many possibilities in between depending on trade-offs between speed, "training" size, and ultimately the dimension of the reduced data.

Some Extensions

- Kernel PCA: Handles non-linearity in the data.

Some Extensions

- Kernel PCA: Handles non-linearity in the data.
- Sparse PCA: Preserves sparseness in the data, but computationally tricky (espensive).

Some Extensions

- Kernel PCA: Handles non-linearity in the data.
- Sparse PCA: Preserves sparseness in the data, but computationally tricky (espensive).
- Independent Component Analysis: ICA is a ”deepening” of PCA to find additional structure and not just maximum variance.