Principal Component Analysis (PCA)

TOP: Data Clustering 076/091

Instructor: Sayan Bandyapadhyay

Portland State University

Outline

2024 GDP Prediction of U.S.

- U.S. GDP for the first quarter of 2024
- U.S. GDP for the entirety of 2023, 2022, and so on.
- Unemployment rate
- Inflation rate
- Number of people work in each industry
- Number of members of the House and Senate belong to each political party
- Stock price data
- Number of CEOs seem to be mounting a bid for public office

2024 GDP Prediction of U.S.

- U.S. GDP for the first quarter of 2024
- U.S. GDP for the entirety of 2023, 2022, and so on.
- Unemployment rate
- Inflation rate
- Number of people work in each industry
- Number of members of the House and Senate belong to each political party
- Stock price data
- Number of CEOs seem to be mounting a bid for public office

That is a lot of parameters/features!

Curse of Dimensionality

Time complexity typically depends exponentially on the dimension

Curse of Dimensionality

Time complexity typically depends exponentially on the dimension

- k-means/median: $(1 + \epsilon)$-approximation in n^d time

Curse of Dimensionality

Time complexity typically depends exponentially on the dimension

- k-means/median: $(1 + \epsilon)$-approximation in n^d time
- For large d, distance computation is expensive: for $d = logn$, takes $O(\log n)$ time

Dimensionality Reduction

Dependent vs Independent features/variables

Dimensionality Reduction

Dependent vs Independent features/variables

- Feature elimination
  - Remove a subset of dependent features/variables
  - Information is lost

Dimensionality Reduction

Dependent vs Independent features/variables

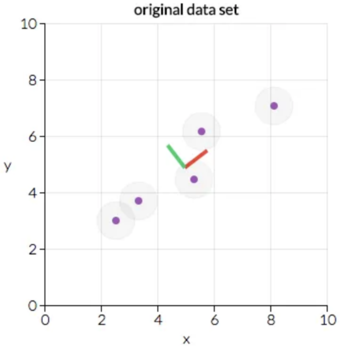
- Feature elimination
  - Remove a subset of dependent features/variables
  - Information is lost

- Feature extraction
  - New features are created that are independent
  - Old features are combined to create new features
  - Number of new features is small
  - New features are not interpretable
  - Information loss is controlled

Outline

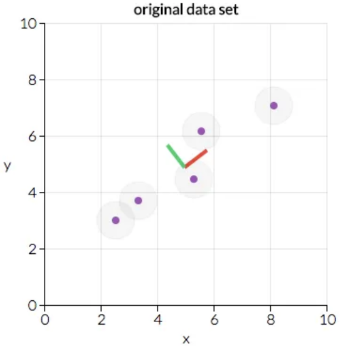
Feature Extraction using PCA

Finds main directions of the data (that have higher variance)



Feature Extraction using PCA

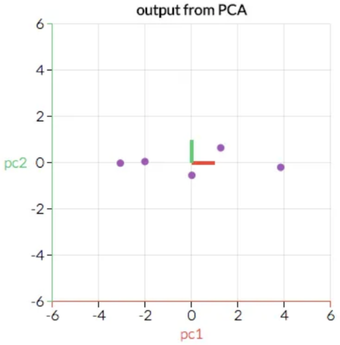
Finds main directions of the data (that have higher variance)



What line can be fit into this data?

Output of PCA

Transformed data by independent directions/vectors



We can drop the y direction without losing a lot of information (dimensionality reduction)

Outline

Eigenvectors and Eigenvalues

- Consider a square matrix A ($d \times d$)
- $\vec{v}$ ($d \times 1$) is an eigenvector if for some λ, $A\vec{v} = \lambda\vec{v}$

Eigenvectors and Eigenvalues

- Consider a square matrix A ($d \times d$)
- $\vec{v}$ ($d \times 1$) is an eigenvector if for some λ, $A\vec{v} = \lambda\vec{v}$
- λ is the eigenvalue corresponding to $\vec{v}$

Eigenvectors and Eigenvalues

- Consider a square matrix A ($d \times d$)
- $\vec{v}$ ($d \times 1$) is an eigenvector if for some λ, $A\vec{v} = \lambda\vec{v}$
- λ is the eigenvalue corresponding to $\vec{v}$
- $A = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix}$ and $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$

Eigenvectors and Eigenvalues

- Consider a square matrix A ($d \times d$)
- $\vec{v}$ ($d \times 1$) is an eigenvector if for some λ, $A\vec{v} = \lambda\vec{v}$
- λ is the eigenvalue corresponding to $\vec{v}$
- $A = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix}$ and $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$
- $A\vec{v} = \begin{pmatrix} 3v_1 + v_2 \\ 2v_2 \end{pmatrix}$

Eigenvectors and Eigenvalues

- Consider a square matrix A ($d \times d$)
- $\vec{v}$ ($d \times 1$) is an eigenvector if for some λ, $A\vec{v} = \lambda\vec{v}$
- λ is the eigenvalue corresponding to $\vec{v}$
- $A = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix}$ and $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$
- $A\vec{v} = \begin{pmatrix} 3v_1 + v_2 \\ 2v_2 \end{pmatrix} = \begin{pmatrix} \lambda v_1 \\ \lambda v_2 \end{pmatrix}$

Eigenvectors and Eigenvalues

- Consider a square matrix A ($d \times d$)
- $\vec{v}$ ($d \times 1$) is an eigenvector if for some λ, $A\vec{v} = \lambda\vec{v}$
- λ is the eigenvalue corresponding to $\vec{v}$
- $A = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix}$ and $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$
- $A\vec{v} = \begin{pmatrix} 3v_1 + v_2 \\ 2v_2 \end{pmatrix} = \begin{pmatrix} \lambda v_1 \\ \lambda v_2 \end{pmatrix}$
- $v_1 = -1, v_2 = 1$

Eigenvectors and Eigenvalues

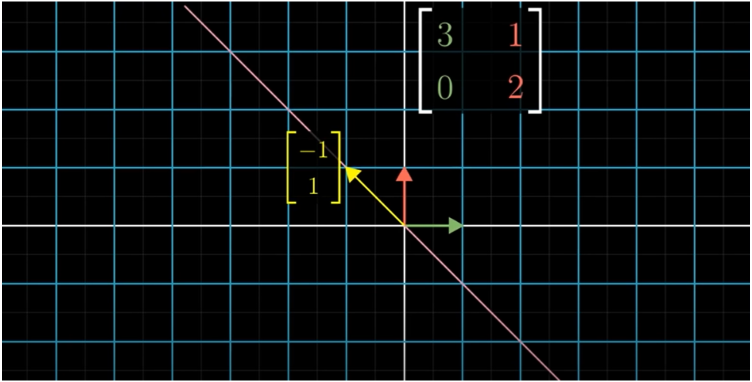
- Consider a square matrix A ($d \times d$)
- $\vec{v}$ ($d \times 1$) is an eigenvector if for some λ, $A\vec{v} = \lambda\vec{v}$
- λ is the eigenvalue corresponding to $\vec{v}$
- $A = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix}$ and $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$
- $A\vec{v} = \begin{pmatrix} 3v_1 + v_2 \\ 2v_2 \end{pmatrix} = \begin{pmatrix} \lambda v_1 \\ \lambda v_2 \end{pmatrix}$
- $v_1 = -1, v_2 = 1$
- $A\vec{v} = \begin{pmatrix} -2 \\ 2 \end{pmatrix} = 2\begin{pmatrix} -1 \\ 1 \end{pmatrix} = \lambda\vec{v}$

Eigenvectors and Eigenvalues

- Consider a square matrix A ($d \times d$)
- $\vec{v}$ ($d \times 1$) is an eigenvector if for some λ, $A\vec{v} = \lambda\vec{v}$
- λ is the eigenvalue corresponding to $\vec{v}$
- $A = \begin{pmatrix} 3 & 1 \\ 0 & 2 \end{pmatrix}$ and $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$
- $A\vec{v} = \begin{pmatrix} 3v_1 + v_2 \\ 2v_2 \end{pmatrix} = \begin{pmatrix} \lambda v_1 \\ \lambda v_2 \end{pmatrix}$
- $v_1 = -1, v_2 = 1$
- $A\vec{v} = \begin{pmatrix} -2 \\ 2 \end{pmatrix} = 2\begin{pmatrix} -1 \\ 1 \end{pmatrix} = \lambda\vec{v}$

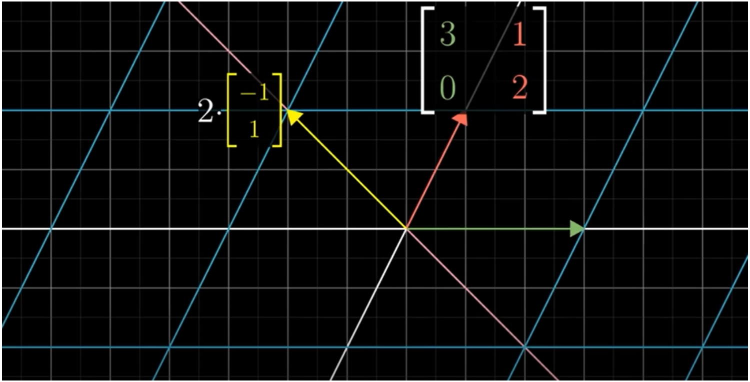
Link to youtube video

An Example



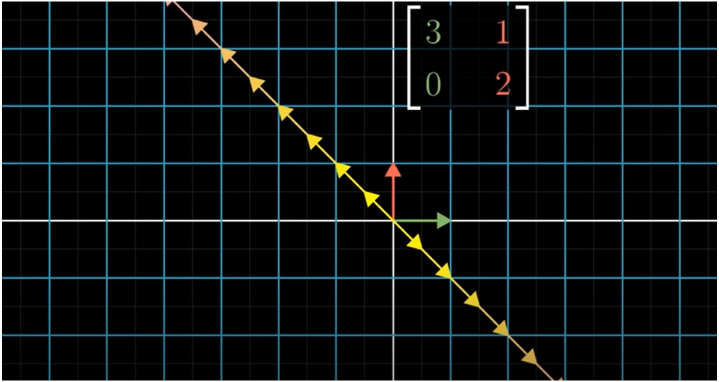
Before transformation

An Example



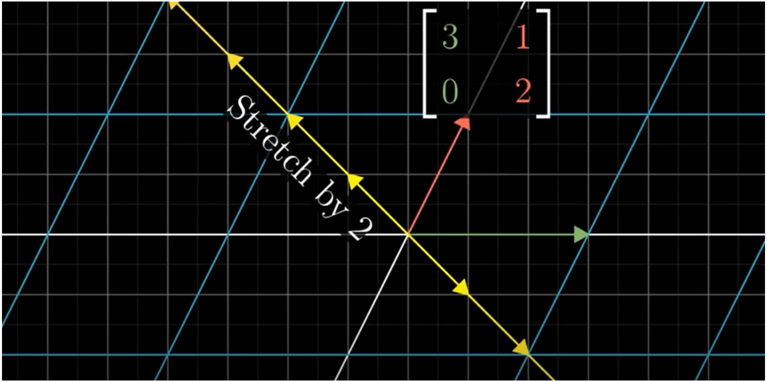
After transformation

An Example



Span of $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$ before

An Example



The span remains the same afterwards

Outline

PCA - Step 1 - Centering

Given the dataset $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$

- Create an $n \times d$ matrix X using the dataset: rows :: points, columns :: features

$$\begin{array}{c c} & \begin{array}{cccc} f_1 & f_2 & \cdots & f_d \end{array} \\ \begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} & \left[\begin{array}{cccc} \; & \; & \; & \; \\ \; & \; & \; & \; \\ \; & \; & \; & \; \\ \; & \; & \; & \; \end{array} \right] \end{array}$$

$$X_{n \times d}$$

PCA - Step 1 - Centering

Given the dataset $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$

- Create an $n \times d$ matrix X using the dataset: rows :: points, columns :: features

$$\begin{array}{c} \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{array} \begin{array}{c} \begin{array}{cccc} f_1 & f_2 & \cdots & f_d \end{array} \\ \left[\begin{array}{cccc} & & & \\ & & & \\ & & & \\ & & & \end{array} \right] \end{array}$$

$$X_{n \times d}$$

- For each column, subtract the mean of that column from each entry

PCA - Step 1 - Centering

Given the dataset $\{x_1, \ldots, x_n\}$ in $\mathbb{R}^d$

- Create an $n \times d$ matrix X using the dataset: rows :: points, columns :: features

$$
\begin{array}{c}
 \\
x_1 \\
x_2 \\
\vdots \\
x_n
\end{array}
\begin{array}{c}
\begin{array}{cccc} f_1 & f_2 & \cdots & f_d \end{array} \\
\left[\begin{array}{cccc} & & & \\ & & & \\ & & & \\ & & & \end{array}\right]
\end{array}
$$

$$X_{n \times d}$$

- For each column, subtract the mean of that column from each entry
- Each column has a mean of zero

PCA - Step 2 - Interrelating the Features

- Transpose X and multiply by X to create a matrix $C = X^T X$

PCA - Step 2 - Interrelating the Features

- Transpose X and multiply by X to create a matrix $C = X^T X$
- X^T is $d \times n$: columns :: points, rows :: features

$$
\underbrace{\begin{array}{c|c}
 & x_1 \; x_2 \; \ldots \; x_n \\ \hline
\begin{array}{c} f_1 \\ f_2 \\ \vdots \\ f_d \end{array} & \left[\begin{array}{cccc} \; & \; & \; & \; \\ & & & \\ & & & \\ & & & \end{array}\right]
\end{array}}_{X^T_{d\times n}}
\times
\underbrace{\begin{array}{c|c}
 & f_1 \; f_2 \; \ldots \; f_d \\ \hline
\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} & \left[\begin{array}{cccc} \; & \; & \; & \; \\ & & & \\ & & & \\ & & & \end{array}\right]
\end{array}}_{X_{n\times d}}
=
\underbrace{\left[\begin{array}{ccc}
f_1^T f_1 & \ldots. & f_1^T f_d \\
f_2^T f_1 & & f_2^T f_d \\
\vdots & f_i^T f_j & \vdots \\
f_d^T f_1 & \ldots. & f_d^T f_d
\end{array}\right]}_{C_{d\times d}}
$$

PCA - Step 2 - Interrelating the Features

- Transpose X and multiply by X to create a matrix $C = X^T X$
- X^T is $d \times n$: columns :: points, rows :: features

$$\underbrace{\begin{array}{c|c} & x_1 \; x_2 \; \dots \; x_n \\ \hline \begin{array}{c} f_1 \\ f_2 \\ \vdots \\ f_d \end{array} & \left[\begin{array}{c} \quad \\ \quad \\ \quad \\ \quad \end{array} \right] \end{array}}_{X^T_{d\times n}} \times \underbrace{\begin{array}{c|c} & f_1 \; f_2 \; \dots \; f_d \\ \hline \begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_n \end{array} & \left[\begin{array}{c} \quad \\ \quad \\ \quad \\ \quad \end{array} \right] \end{array}}_{X_{n\times d}} = \underbrace{\begin{bmatrix} f_1^T f_1 & \dots & f_1^T f_d \\ f_2^T f_1 & & f_2^T f_d \\ \vdots & f_i^T f_j & \vdots \\ f_d^T f_1 & \dots & f_d^T f_d \end{bmatrix}}_{C_{d\times d}}$$

- So, C is basically the covariance matrix of X

PCA - Step 2 - Interrelating the Features

- Transpose X and multiply by X to create a matrix $C = X^T X$
- X^T is $d \times n$: columns :: points, rows :: features

$$\underset{X^T_{d\times n}}{\begin{array}{cc} & \begin{matrix} x_1 & x_2 & \ldots & x_n \end{matrix} \\ \begin{matrix} f_1 \\ f_2 \\ \vdots \\ f_d \end{matrix} & \left[\begin{matrix} \quad & \quad & \quad & \quad \\ & & & \\ & & & \\ & & & \end{matrix}\right] \end{array}} \times \underset{X_{n\times d}}{\begin{array}{cc} & \begin{matrix} f_1 & f_2 & \ldots & f_d \end{matrix} \\ \begin{matrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{matrix} & \left[\begin{matrix} \quad & \quad & \quad & \quad \\ & & & \\ & & & \\ & & & \end{matrix}\right] \end{array}} = \underset{C_{d\times d}}{\left[\begin{matrix} f_1^T f_1 & \ldots\ldots & f_1^T f_d \\ f_2^T f_1 & & f_2^T f_d \\ \vdots & f_i^T f_j & \vdots \\ f_d^T f_1 & \ldots\ldots & f_d^T f_d \end{matrix}\right]}$$

- So, C is basically the covariance matrix of X
- Shows how every variable in X relates to every other variable in X

PCA - Step 3 - Eigendecomposition

- Calculate the eigenvectors and their corresponding eigenvalues of C

PCA - Step 3 - Eigendecomposition

- Calculate the eigenvectors and their corresponding eigenvalues of C
- C can be decomposed into PDP^{-1}, where P is the matrix of eigenvectors and D is the diagonal matrix with eigenvalues

$$\underbrace{\begin{bmatrix} \vdots & \vdots & & \vdots \\ e_1 & e_2 & \cdots & e_d \\ \vdots & \vdots & & \vdots \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_d \end{bmatrix}}_{D}$$

PCA - Step 3 - Eigendecomposition

- Calculate the eigenvectors and their corresponding eigenvalues of C
- C can be decomposed into PDP^{-1}, where P is the matrix of eigenvectors and D is the diagonal matrix with eigenvalues

$$\underbrace{\begin{bmatrix} \vdots & \vdots & & \vdots \\ e_1 & e_2 & \cdots & e_d \\ \vdots & \vdots & & \vdots \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & 0 \\ & & \ddots & \\ 0 & & & \lambda_d \end{bmatrix}}_{D}$$

- Eigenvectors represent directions - principal components

PCA - Step 3 - Eigendecomposition

- Calculate the eigenvectors and their corresponding eigenvalues of C
- C can be decomposed into PDP^{-1}, where P is the matrix of eigenvectors and D is the diagonal matrix with eigenvalues

$$\underbrace{\begin{bmatrix} \vdots & \vdots & & \vdots \\ e_1 & e_2 & \cdots & e_d \\ \vdots & \vdots & & \vdots \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & 0 \\ & & \ddots & \\ 0 & & & \lambda_d \end{bmatrix}}_{D}$$

- Eigenvectors represent directions - principal components
- Eigenvalues represent magnitude, or importance
- Bigger eigenvalues correlate with more important directions

PCA - Step 4 - Sorting Eigenvectors

- Take the eigenvalues $\lambda_1, \ldots, \lambda_d$ and sort them from largest to smallest

PCA - Step 4 - Sorting Eigenvectors

- Take the eigenvalues $\lambda_1, \ldots, \lambda_d$ and sort them from largest to smallest
- Sort the eigenvectors in $P_{d \times d}$ accordingly

PCA - Step 4 - Sorting Eigenvectors

- Take the eigenvalues $\lambda_1, \ldots, \lambda_d$ and sort them from largest to smallest
- Sort the eigenvectors in $P_{d \times d}$ accordingly
- Call this sorted matrix of eigenvectors $P^*_{d \times d}$

PCA - Step 4 - Sorting Eigenvectors

- Take the eigenvalues $\lambda_1, \ldots, \lambda_d$ and sort them from largest to smallest
- Sort the eigenvectors in $P_{d \times d}$ accordingly
- Call this sorted matrix of eigenvectors $P^*_{d \times d}$
- Eigenvectors are independent of one another

PCA - Step 5 - Projecting the Data Points

- Calculate $X^*_{n \times d} = X_{n \times d} P^*_{d \times d}$

PCA - Step 5 - Projecting the Data Points

- Calculate $X^*_{n \times d} = X_{n \times d} P^*_{d \times d}$
- Each row of X^* is a transformed point

PCA - Step 5 - Projecting the Data Points

- Calculate $X^*_{n \times d} = X_{n \times d} P^*_{d \times d}$
- Each row of X^* is a transformed point
- Each point is a combination of the original variables, where the weights are determined by the eigenvectors

PCA - Step 5 - Projecting the Data Points

- Calculate $X^*_{n \times d} = X_{n \times d} P^*_{d \times d}$
- Each row of X^* is a transformed point
- Each point is a combination of the original variables, where the weights are determined by the eigenvectors
- Dimension is still d
- No information loss so far

Summary of PCA

1. Create the matrix X whose each column has a mean of zero
2. Compute the covariance matrix $C = X^T X$
3. Calculate the eigenvectors and their corresponding eigenvalues of C
4. Compute the sorted matrix of eigenvectors P^* from P, where $C = PDP^{-1}$
5. Calculate $X^* = XP^*$

Dropping Dimensions

1. Arbitrarily select how many dimensions to keep - useful for visual representation

Dropping Dimensions

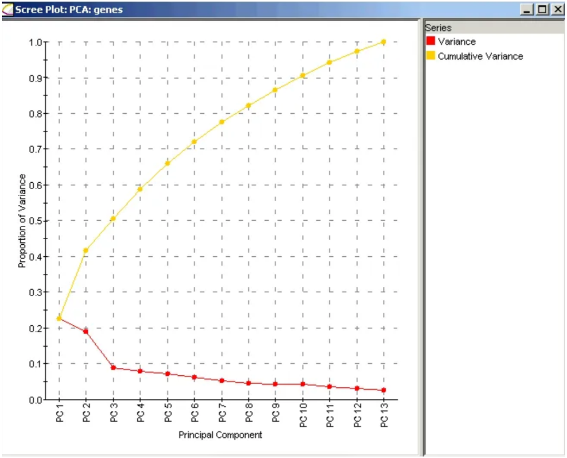
1. Arbitrarily select how many dimensions to keep - useful for visual representation

2. Add features based on their importance (eigenvalues) until a set threshold is achieved
$\lambda_1/(\sum_i \lambda_i) + \lambda_2/(\sum_i \lambda_i) + \dots$

Dropping Dimensions

1. Arbitrarily select how many dimensions to keep - useful for visual representation
2. Add features based on their importance (eigenvalues) until a set threshold is achieved
   $\lambda_1/(\sum_i \lambda_i) + \lambda_2/(\sum_i \lambda_i) + \dots$
3. Add features until a significant drop in the eigenvalue - Elbow method

An Example


Scree Plot: PCA: genes
Series
Variance
Cumulative Variance
Proportion of Variance
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
PC 1
PC 2
PC 3
PC 4
PC 5
PC 6
PC 7
PC 8
PC 9
PC 10
PC 11
PC 12
PC 13
Principal Component

Outline

Johnson Lindenstrauss (JL) Lemma ’84

For any set of n data points in $\mathbb{R}^d$, there is a map/projection f to $\mathbb{R}^k$ for $k = O(\log n)$ with small distortion of the distances.

Johnson Lindenstrauss (JL) Lemma '84

For any set of n data points in $\mathbb{R}^d$, there is a map/projection f to $\mathbb{R}^k$ for $k = O(\log n)$ with small distortion of the distances.

- f maps $x_i \in X$ to $f(x_i) \in \mathbb{R}^k$

Johnson Lindenstrauss (JL) Lemma '84

For any set of n data points in $\mathbb{R}^d$, there is a map/projection f to $\mathbb{R}^k$ for $k = O(\log n)$ with small distortion of the distances.

- f maps $x_i \in X$ to $f(x_i) \in \mathbb{R}^k$
- d is assumed to be much larger than $\log n$

Johnson Lindenstrauss (JL) Lemma '84

For any set of n data points in $\mathbb{R}^d$, there is a map/projection f to $\mathbb{R}^k$ for $k = O(\log n)$ with small distortion of the distances.

- f maps $x_i \in X$ to $f(x_i) \in \mathbb{R}^k$
- d is assumed to be much larger than $\log n$
- Let $\epsilon \in (0, 1)$
- For any two $x_i, x_j \in X$, distortion $\frac{||f(x_i)-f(x_j)||}{||x_i-x_j||} \in [1 - \epsilon, 1 + \epsilon]$

Johnson Lindenstrauss (JL) Lemma '84

For any set of n data points in $\mathbb{R}^d$, there is a map/projection f to $\mathbb{R}^k$ for $k = O(\log n)$ with small distortion of the distances.

- f maps $x_i \in X$ to $f(x_i) \in \mathbb{R}^k$
- d is assumed to be much larger than $\log n$
- Let $\epsilon \in (0, 1)$
- For any two $x_i, x_j \in X$, distortion $\frac{||f(x_i) - f(x_j)||}{||x_i - x_j||} \in [1 - \epsilon, 1 + \epsilon]$
- $k = O(\log n / \epsilon^2)$

Johnson Lindenstrauss (JL) Lemma ’84

For any set of n data points in $\mathbb{R}^d$, there is a map/projection f to $\mathbb{R}^k$ for $k = O(\log n)$ with small distortion of the distances.

- f maps $x_i \in X$ to $f(x_i) \in \mathbb{R}^k$
- d is assumed to be much larger than $\log n$
- Let $\epsilon \in (0, 1)$
- For any two $x_i, x_j \in X$, distortion $\frac{||f(x_i)-f(x_j)||}{||x_i-x_j||} \in [1-\epsilon, 1+\epsilon]$
- $k = O(\log n/\epsilon^2)$
- k cannot be arbitrarily small

Partial Proof of JL Lemma

Proof is by construction of a map $f \Longrightarrow$ An algorithm for computing f

- There are many proofs – we discuss one

Partial Proof of JL Lemma

Proof is by construction of a map $f \Longrightarrow$ An algorithm for computing f

- There are many proofs – we discuss one
- We construct a $k \times d$ matrix M

Partial Proof of JL Lemma

Proof is by construction of a map $f \Rightarrow$ An algorithm for computing f

- There are many proofs – we discuss one
- We construct a $k \times d$ matrix M
- Each entry is an independent sample from the standard normal $N(0, 1)$ distribution

Partial Proof of JL Lemma

Proof is by construction of a map $f \Longrightarrow$ An algorithm for computing f

- There are many proofs – we discuss one
- We construct a $k \times d$ matrix M
- Each entry is an independent sample from the standard normal $N(0, 1)$ distribution
- $f(x) := \frac{1}{\sqrt{k}} M_{k \times d} x_{d \times 1}$

Partial Proof of JL Lemma

Proof is by construction of a map $f \Rightarrow$ An algorithm for computing f

- There are many proofs – we discuss one
- We construct a $k \times d$ matrix M
- Each entry is an independent sample from the standard normal $N(0, 1)$ distribution
- $f(x) := \frac{1}{\sqrt{k}} M_{k \times d} x_{d \times 1}$
- The proof uses properties of $N(0, 1)$ – Random projections work

Proof: Chapter 2.7

Terminal Embedding (Narayanan and Nelson '19)

JL Lemma only preserves distances between points in X

- This is good for applications such as Nearest Neighbor search

Terminal Embedding (Narayanan and Nelson '19)

JL Lemma only preserves distances between points in X

- This is good for applications such as Nearest Neighbor search
- But, not suitable for clustering – we need distances between centers and points to be preserved

Terminal Embedding (Narayanan and Nelson ’19)

JL Lemma only preserves distances between points in X

- This is good for applications such as Nearest Neighbor search
- But, not suitable for clustering – we need distances between centers and points to be preserved
- This is possible – Terminal embedding

Terminal Embedding (Narayanan and Nelson ’19)

JL Lemma only preserves distances between points in X

- This is good for applications such as Nearest Neighbor search
- But, not suitable for clustering – we need distances between centers and points to be preserved
- This is possible – Terminal embedding
- For any two $x_i \in \mathbb{R}^d, x_j \in X$, distortion $\frac{||f(x_i)-f(x_j)||}{||x_i-x_j||} \in [1-\epsilon, 1+\epsilon]$

Terminal Embedding (Narayanan and Nelson '19)

JL Lemma only preserves distances between points in X

- This is good for applications such as Nearest Neighbor search
- But, not suitable for clustering – we need distances between centers and points to be preserved
- This is possible – Terminal embedding
- For any two $x_i \in \mathbb{R}^d, x_j \in X$, distortion $\frac{||f(x_i)-f(x_j)||}{||x_i-x_j||} \in [1-\epsilon, 1+\epsilon]$
- Proof is along the same line, but much more complicated

Terminal Embedding (Narayanan and Nelson '19)

JL Lemma only preserves distances between points in X

- This is good for applications such as Nearest Neighbor search
- But, not suitable for clustering – we need distances between centers and points to be preserved
- This is possible – Terminal embedding
- For any two $x_i \in \mathbb{R}^d, x_j \in X$, distortion $\frac{||f(x_i)-f(x_j)||}{||x_i-x_j||} \in [1-\epsilon, 1+\epsilon]$
- Proof is along the same line, but much more complicated

For k-means/median, the dimension can be improved from $O(\log n)$ to $O(\log k)$ to preserve the cost of all clustering