

# Lecture 16: 3 June, 2021

Madhavan Mukund

<https://www.cmi.ac.in/~madhavan>

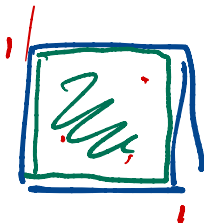
Data Mining and Machine Learning  
April–July 2021

# The curse of dimensionality

- ML data is often high dimensional — especially images
  - A  $1000 \times 1000$  pixel image has  $10^6$  features

# The curse of dimensionality

- ML data is often high dimensional — especially images
  - A  $1000 \times 1000$  pixel image has  $10^6$  features
- Data behaves very differently in high dimensions
  - $2D$  unit square,  $0.04\%$  probability of being near the border (within  $0.001$ )



# The curse of dimensionality

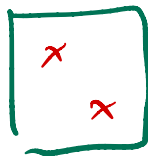
- ML data is often high dimensional — especially images
  - A  $1000 \times 1000$  pixel image has  $10^6$  features
- Data behaves very differently in high dimensions
  - $2D$  unit square,  $0.04\%$  probability of being near the border (within  $0.001$ )
  - $10^4 D$  hypercube,  $99.999999\%$  probability of being near the border

# The curse of dimensionality

- ML data is often high dimensional — especially images
  - A  $1000 \times 1000$  pixel image has  $10^6$  features
- Data behaves very differently in high dimensions
  - $2D$  unit square, 0.04% probability of being near the border (within 0.001)
  - $10^4 D$  hypercube, 99.999999% probability of being near the border
- Distances between items

# The curse of dimensionality

- ML data is often high dimensional — especially images
  - A  $1000 \times 1000$  pixel image has  $10^6$  features
- Data behaves very differently in high dimensions
  - $2D$  unit square,  $0.04\%$  probability of being near the border (within  $0.001$ )
  - $10^4 D$  hypercube,  $99.999999\%$  probability of being near the border
- Distances between items
  - $2D$  unit square, mean distance between  $2$  random points is  $0.52$



# The curse of dimensionality

- ML data is often high dimensional — especially images
  - A  $1000 \times 1000$  pixel image has  $10^6$  features
- Data behaves very differently in high dimensions
  - $2D$  unit square,  $0.04\%$  probability of being near the border (within  $0.001$ )
  - $10^4 D$  hypercube,  $99.999999\%$  probability of being near the border
- Distances between items
  - $2D$  unit square, mean distance between  $2$  random points is  $0.52$
  - $3D$  unit cube, mean distance between  $2$  random points is  $0.66$

# The curse of dimensionality

- ML data is often high dimensional — especially images
  - A  $1000 \times 1000$  pixel image has  $10^6$  features
- Data behaves very differently in high dimensions
  - $2D$  unit square,  $0.04\%$  probability of being near the border (within  $0.001$ )
  - $10^4 D$  hypercube,  $99.999999\%$  probability of being near the border
- Distances between items
  - $2D$  unit square, mean distance between  $2$  random points is  $0.52$
  - $3D$  unit cube, mean distance between  $2$  random points is  $0.66$
  - $10^6 D$  unit hypercube, mean distance between  $2$  random points is approximately  $408.25$



# The curse of dimensionality

- ML data is often high dimensional — especially images
  - A  $1000 \times 1000$  pixel image has  $10^6$  features
- Data behaves very differently in high dimensions
  - $2D$  unit square, 0.04% probability of being near the border (within 0.001)
  - $10^4 D$  hypercube, 99.999999% probability of being near the border
- Distances between items
  - $2D$  unit square, mean distance between 2 random points is 0.52
  - $3D$  unit cube, mean distance between 2 random points is 0.66
  - $10^6 D$  unit hypercube, mean distance between 2 random points is approximately 408.25
  - There's a lot of “space” in higher dimensions!
  - Higher danger of overfitting

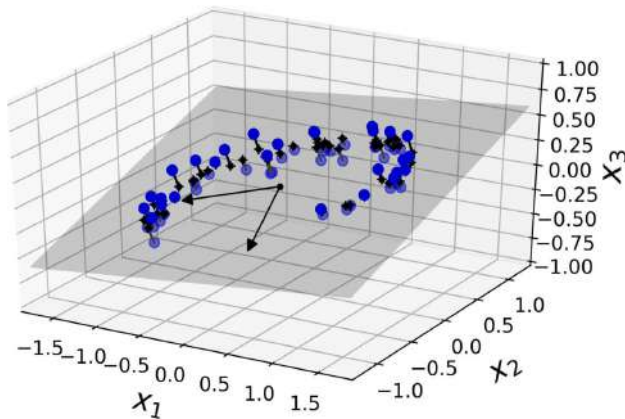
# Dimensionality reduction

- Remove unimportant features by projecting to a smaller dimension



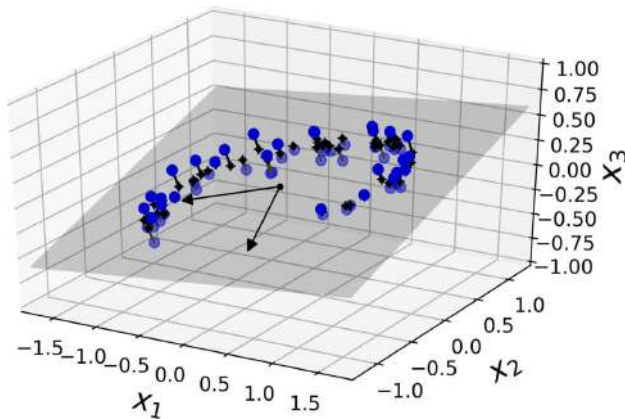
# Dimensionality reduction

- Remove unimportant features by projecting to a smaller dimension
- Example: project blue points in 3D to black points in 2D plane



# Dimensionality reduction

- Remove unimportant features by projecting to a smaller dimension
- Example: project blue points in 3D to black points in 2D plane
- **Principal Component Analysis** — transform  $d$ -dimensional input to  $k$ -dimensional input, preserving essential features



# Singular Value Decomposition (SVD)

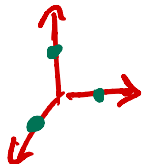
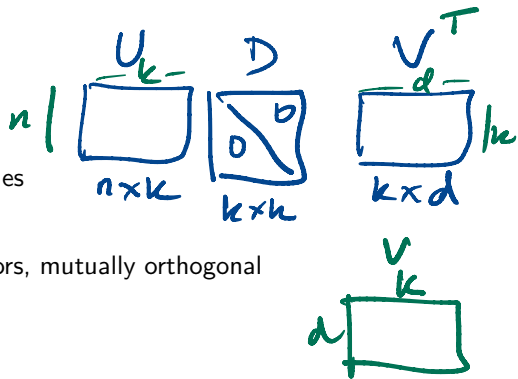
- Input matrix  $M$ , dimensions  $n \times d$ 
  - Rows are items, columns are features

Handwritten representation of the input matrix  $M$  with dimensions  $n \times d$ .

	Features			
	$x_1$	$x_2$	--	$x_d$
Items	$v_1$	$v_2$	--	$v_d$
	$\vdots$			
	$\vdots$			
	$v_n$	$v_2^n$	--	$v_d^n$

# Singular Value Decomposition (SVD)

- Input matrix  $M$ , dimensions  $n \times d$ 
  - Rows are items, columns are features
- Decompose  $M$  as  $UDV^T$ 
  - $D$  is a  $k \times k$  diagonal matrix, positive real entries
  - $U$  is  $n \times k$ ,  $V$  is  $d \times k$
  - Columns of  $U$ ,  $V$  are **orthonormal** — unit vectors, mutually orthogonal



# Singular Value Decomposition (SVD)

- Input matrix  $M$ , dimensions  $n \times d$  *original features*
  - Rows are items, columns are features
- Decompose  $M$  as  $UDV^T$ 
  - $D$  is a  $k \times k$  diagonal matrix, positive real entries
  - $U$  is  $n \times k$ ,  $V$  is  $d \times k$  *new features*
  - Columns of  $U$ ,  $V$  are *orthonormal* — unit vectors, mutually orthogonal
- Interpretation
  - Columns of  $V$  correspond to new abstract features

# Singular Value Decomposition (SVD)

- Input matrix  $M$ , dimensions  $n \times d$ 
  - Rows are items, columns are features
- Decompose  $M$  as  $UDV^T$ 
  - $D$  is a  $k \times k$  diagonal matrix, positive real entries
  - $U$  is  $n \times k$ ,  $V$  is  $d \times k$
  - Columns of  $U$ ,  $V$  are **orthonormal** — unit vectors, mutually orthogonal
- Interpretation
  - Columns of  $V$  correspond to new abstract features
  - Rows of  $U$  describe decomposition of terms across features



# Singular Value Decomposition (SVD)

- Input matrix  $M$ , dimensions  $n \times d$ 
  - Rows are items, columns are features
- Decompose  $M$  as  $UDV^T$ 
  - $D$  is a  $k \times k$  diagonal matrix, positive real entries
  - $U$  is  $n \times k$ ,  $V$  is  $d \times k$
  - Columns of  $U$ ,  $V$  are **orthonormal** — unit vectors, mutually orthogonal
- Interpretation
  - Columns of  $V$  correspond to new abstract features
  - Rows of  $U$  describe decomposition of terms across features
  - $M = \sum_i D_{ii}(\mathbf{u}_i \cdot \mathbf{v}_i^T)$

# Singular Value Decomposition (SVD)

- Input matrix  $M$ , dimensions  $n \times d$ 
  - Rows are items, columns are features
- Decompose  $M$  as  $UDV^T$ 
  - $D$  is a  $k \times k$  diagonal matrix, positive real entries
  - $U$  is  $n \times k$ ,  $V$  is  $d \times k$
  - Columns of  $U$ ,  $V$  are **orthonormal** — unit vectors, mutually orthogonal
- Interpretation
  - Columns of  $V$  correspond to new abstract features
  - Rows of  $U$  describe decomposition of terms across features
  - $M = \sum D_{ii} (u_i \cdot v_i^T)$
  - For columns  $u_i$  of  $U$  and  $v_i$  of  $V$ ,  $u_i \cdot v_i^T$  is an  $n \times d$  matrix, like  $M$

$u_i: n \times 1$      $v_i: d \times 1$      $v_i^T: 1 \times d$

# Singular Value Decomposition (SVD)

- Input matrix  $M$ , dimensions  $n \times d$ 
  - Rows are items, columns are features
- Decompose  $M$  as  $UDV^T$ 
  - $D$  is a  $k \times k$  diagonal matrix, positive real entries
  - $U$  is  $n \times k$ ,  $V$  is  $d \times k$
  - Columns of  $U$ ,  $V$  are **orthonormal** — unit vectors, mutually orthogonal
- Interpretation
  - Columns of  $V$  correspond to new abstract features
  - Rows of  $U$  describe decomposition of terms across features
  - $M = \sum_i D_{ii}(\mathbf{u}_i \cdot \mathbf{v}_i^T)$
  - For columns  $\mathbf{u}_i$  of  $U$  and  $\mathbf{v}_i$  of  $V$ ,  $\mathbf{u}_i \cdot \mathbf{v}_i^T$  is an  $n \times d$  matrix, like  $M$
  - $\mathbf{u}_i \cdot \mathbf{v}_i^T$  describes components of rows of  $M$  along direction  $\mathbf{v}_i$

# Singular vectors

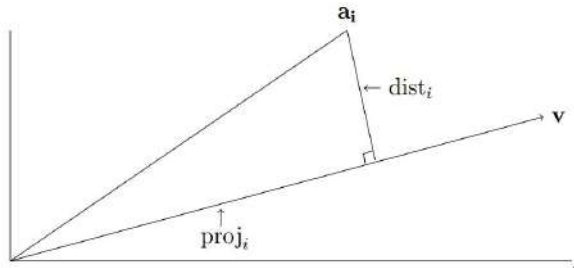
- Unit vectors passing through the origin

# Singular vectors

- Unit vectors passing through the origin
- Want to find “best”  $k$  singular vectors to represent feature space

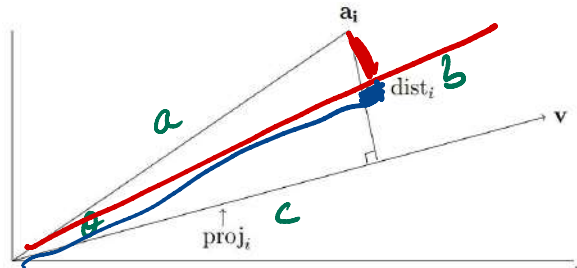
# Singular vectors

- Unit vectors passing through the origin
- Want to find “best”  $k$  singular vectors to represent feature space
- Suppose we project  $\mathbf{a}_i = (a_{i1}, a_{i2}, \dots, a_{id})$  onto  $\mathbf{v}$  through origin



# Singular vectors

- Unit vectors passing through the origin
- Want to find “best”  $k$  singular vectors to represent feature space
- Suppose we project  $\mathbf{a}_i = (a_{i1}, a_{i2}, \dots, a_{id})$  onto  $\mathbf{v}$  through origin
- Minimizing distance of  $\mathbf{a}_i$  from  $\mathbf{v}$  is equivalent to maximizing the projection of  $\mathbf{a}_i$  onto  $\mathbf{v}$
- Length of the projection is  $\mathbf{a}_i \cdot \mathbf{v}$



$$\sin \theta = \frac{b}{a} \rightarrow 0 \quad \theta \rightarrow 0$$

$$\cos \theta = \frac{c}{a} \quad \cos \theta \rightarrow 1$$

# Singular vectors ...

- Sum of squares of lengths of projections of all rows in  $M$  onto  $v$  —  $|Mv|^2$

Projector  $a \cdot v$

$$M = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \cdot \begin{bmatrix} v \\ \vdots \end{bmatrix} \quad \begin{matrix} a_1 \cdot v \\ a_2 \cdot v \\ \vdots \\ a_n \cdot v \end{matrix}$$

$$M \cdot v \quad |Mv|$$

$$\sqrt{(a_1 \cdot v)^2 + (a_2 \cdot v)^2 + \dots + (a_n \cdot v)^2}$$



# Singular vectors ...

- Sum of squares of lengths of projections of all rows in  $M$  onto  $\mathbf{v}$  —  $|M\mathbf{v}|^2$
- First singular vector — unit vector through origin that maximizes the sum of projections of all rows in  $M$

$$\mathbf{v}_1 = \arg \max_{|\mathbf{v}|=1} |M\mathbf{v}|$$

# Singular vectors ...

- Sum of squares of lengths of projections of all rows in  $M$  onto  $\mathbf{v}$  —  $|M\mathbf{v}|^2$
- First singular vector — unit vector through origin that maximizes the sum of projections of all rows in  $M$

$$\mathbf{v}_1 = \arg \max_{|\mathbf{v}|=1} |M\mathbf{v}|$$

- Second singular vector — unit vector through origin, perpendicular to  $\mathbf{v}_1$ , that maximizes the sum of projections of all rows in  $M$

$$\mathbf{v}_2 = \arg \max_{\mathbf{v} \perp \mathbf{v}_1; |\mathbf{v}|=1} |M\mathbf{v}|$$

# Singular vectors ...

- Sum of squares of lengths of projections of all rows in  $M$  onto  $\mathbf{v}$  —  $|M\mathbf{v}|^2$
- First singular vector — unit vector through origin that maximizes the sum of projections of all rows in  $M$

$$\mathbf{v}_1 = \arg \max_{|\mathbf{v}|=1} |M\mathbf{v}|$$

- Second singular vector — unit vector through origin, perpendicular to  $\mathbf{v}_1$ , that maximizes the sum of projections of all rows in  $M$

$$\mathbf{v}_2 = \arg \max_{\mathbf{v} \perp \mathbf{v}_1; |\mathbf{v}|=1} |M\mathbf{v}|$$

- Third singular vector — unit vector through origin, perpendicular to  $\mathbf{v}_1$ ,  $\mathbf{v}_2$ , that maximizes the sum of projections of all rows in  $M$

$$\mathbf{v}_3 = \arg \max_{\mathbf{v} \perp \mathbf{v}_1, \mathbf{v}_2; |\mathbf{v}|=1} |M\mathbf{v}|$$

## Singular vectors ...

- With each singular vector  $\mathbf{v}_j$ , associated singular value is  $\sigma_j = |M\mathbf{v}_j|$

# Singular vectors ...

- With each singular vector  $\mathbf{v}_j$ , associated singular value is  $\sigma_j = |M\mathbf{v}_j|$
- Repeat  $r$  times till 
$$\max_{\mathbf{v} \perp \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r; |\mathbf{v}|=1} |M\mathbf{v}| = 0$$
  - $r$  turns out to be the rank of  $M$
  - Vectors  $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r\}$  are orthonormal right singular vectors

# Singular vectors ...

- With each singular vector  $\mathbf{v}_j$ , associated singular value is  $\sigma_j = |M\mathbf{v}_j|$
- Repeat  $r$  times till 
$$\max_{\mathbf{v} \perp \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r; |\mathbf{v}|=1} |M\mathbf{v}| = 0$$
  - $r$  turns out to be the rank of  $M$
  - Vectors  $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r\}$  are orthonormal **right singular vectors**
- Our greedy strategy provably produces “best-fit” dimension  $r$  subspace for  $M$ 
  - Dimension  $r$  subspace that maximizes content of  $M$  projected onto it

# Singular vectors ...

- With each singular vector  $\mathbf{v}_j$ , associated singular value is  $\sigma_j = |M\mathbf{v}_j|$
- Repeat  $r$  times till  $\max_{\mathbf{v} \perp \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r; |\mathbf{v}|=1} |M\mathbf{v}| = 0$ 
  - $r$  turns out to be the rank of  $M$
  - Vectors  $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r\}$  are orthonormal **right singular vectors**
- Our greedy strategy provably produces “best-fit” dimension  $r$  subspace for  $M$ 
  - Dimension  $r$  subspace that maximizes content of  $M$  projected onto it
- Corresponding **left singular vectors** are given by  $\mathbf{u}_i = \frac{1}{\sigma_i} M\mathbf{v}_i$



$$\text{SVD: } M = U D V^T$$

$\underbrace{\hspace{1.5cm}}_{n \times d} \quad \underbrace{\hspace{1.5cm}}_{n \times k} \quad \underbrace{\hspace{1.5cm}}_{k \times k} \quad \underbrace{\hspace{1.5cm}}_{k \times d}$

$n$  items  
 $d$  features

Diag  
 Items in terms  
 of new features

Old-new  
 $V: d \times k$   
 new features

$M$

$$\sum_{i=1}^k D_{ii} U_i \cdot V_i^T \quad \leadsto \quad n \times d$$

$n \times 1 \quad d \times 1 \quad 1 \times d$



# Singular vectors ...

- With each singular vector  $\mathbf{v}_j$ , associated singular value is  $\sigma_j = |M\mathbf{v}_j|$
- Repeat  $r$  times till  $\max_{\mathbf{v} \perp \mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r; |\mathbf{v}|=1} |M\mathbf{v}| = 0$ 
  - $r$  turns out to be the rank of  $M$
  - Vectors  $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_r\}$  are orthonormal **right singular vectors**
- Our greedy strategy provably produces “best-fit” dimension  $r$  subspace for  $M$ 
  - Dimension  $r$  subspace that maximizes content of  $M$  projected onto it
- Corresponding **left singular vectors** are given by  $\mathbf{u}_i = \frac{1}{\sigma_i} M\mathbf{v}_i$
- Can show that  $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_r\}$  are also orthonormal

# Singular Value Decomposition

- $M$ , dimension  $n \times d$ , of rank  $r$  uniquely decomposes as  $M = UDV^T$ 
  - $V = [v_1 \ v_2 \ \cdots \ v_r]$  are the right singular vectors
  - $D$  is a diagonal matrix with  $D[i, i] = \sigma_i$ , the singular values
  - $U = [u_1 \ u_2 \ \cdots \ u_r]$  are the left singular vectors

$$\begin{array}{|c|} \hline M \\ \hline n \times d \\ \hline \end{array} = \begin{array}{|c|} \hline U \\ \hline n \times r \\ \hline \end{array} \begin{array}{|c|} \hline D \\ \hline r \times r \\ \hline \end{array} \begin{array}{|c|} \hline V^T \\ \hline r \times d \\ \hline \end{array}$$

$$\sigma_1 \approx |Mv_1|$$

$$\sigma_2 \approx |Mv_2|$$

$$\sigma_1 \geq \sigma_2$$

$$\sigma_i < \sigma_{i+1}$$

$$\times \sigma_1 < \sigma_2$$

$$v_i \perp v_1, \dots, v_{i-1} \quad v_{i+1} \perp v_1, \dots, v_i$$

# Rank- $k$ approximation

- $M$  has rank  $r$ , SVD gives rank  $r$  decomposition

# Rank- $k$ approximation

- $M$  has rank  $r$ , SVD gives rank  $r$  decomposition
- Singular values are non-increasing —  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$

# Rank- $k$ approximation

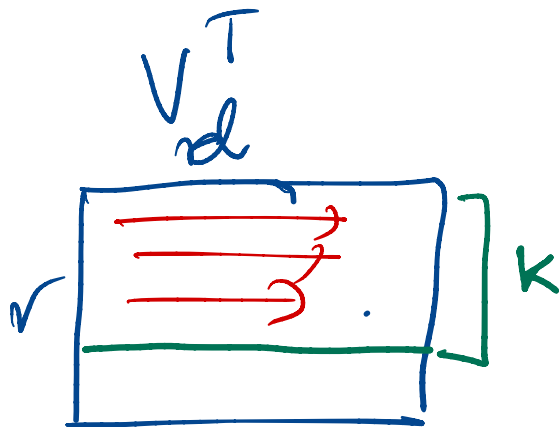
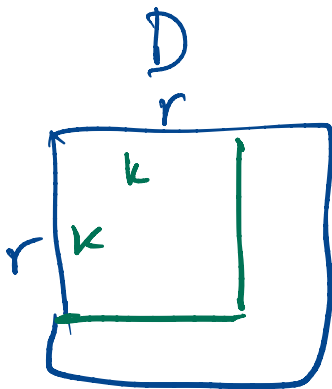
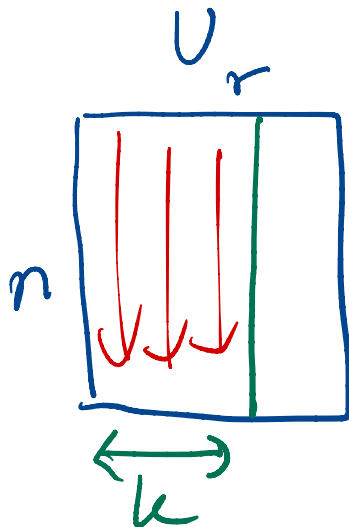
- $M$  has rank  $r$ , SVD gives rank  $r$  decomposition
- Singular values are non-increasing —  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$
- Suppose we retain only  $k$  largest ones

# Rank- $k$ approximation

- $M$  has rank  $r$ , SVD gives rank  $r$  decomposition
- Singular values are non-increasing —  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$
- Suppose we retain only  $k$  largest ones
- We have
  - Matrix of first  $k$  right singular vectors  $V_k = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_k]$ ,
  - Corresponding singular values  $\sigma_1, \sigma_2, \dots, \sigma_k$
  - Matrix of  $k$  left singular vectors  $U_k = [\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_k]$
- Let  $D_k$  be the  $k \times k$  diagonal matrix with entries  $\sigma_1, \sigma_2, \dots, \sigma_k$
- Then  $U_k D_k V_k^T$  is the best fit rank- $k$  approximation of  $M$

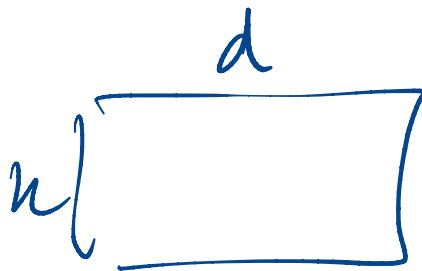
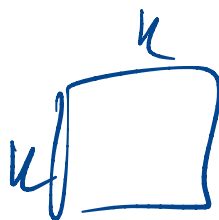
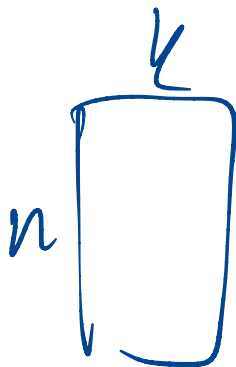
$$M = U D V^T$$

$M$



~~size~~

$$M = n \times d$$



# Rank- $k$ approximation

- $M$  has rank  $r$ , SVD gives rank  $r$  decomposition
- Singular values are non-increasing —  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$
- Suppose we retain only  $k$  largest ones
- We have
  - Matrix of first  $k$  right singular vectors  $V_k = [\mathbf{v}_1 \ \mathbf{v}_2 \ \dots \ \mathbf{v}_k]$ ,
  - Corresponding singular values  $\sigma_1, \sigma_2, \dots, \sigma_k$
  - Matrix of  $k$  left singular vectors  $U_k = [\mathbf{u}_1 \ \mathbf{u}_2 \ \dots \ \mathbf{u}_k]$
- Let  $D_k$  be the  $k \times k$  diagonal matrix with entries  $\sigma_1, \sigma_2, \dots, \sigma_k$
- Then  $U_k D_k V_k^T$  is the best fit rank- $k$  approximation of  $M$
- In other words, by truncating the SVD, we can focus on  $k$  most significant features implicit in  $M$

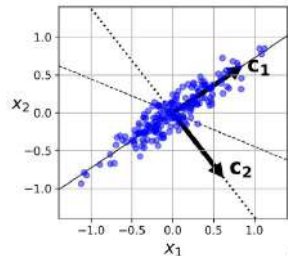


# PCA and variance

- Interpret PCA in terms of preserving variance

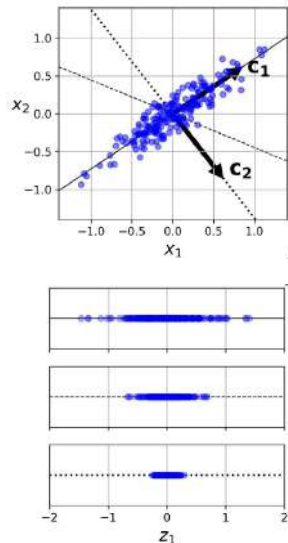
# PCA and variance

- Interpret PCA in terms of preserving variance
- Different projections have different variance



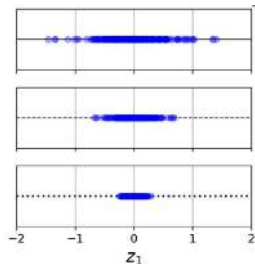
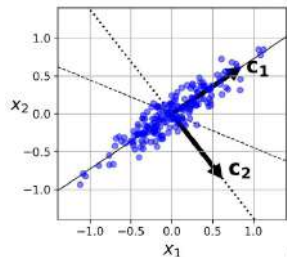
# PCA and variance

- Interpret PCA in terms of preserving variance
- Different projections have different variance
- SVD orders projections in decreasing order of variance



# PCA and variance

- Interpret PCA in terms of preserving variance
- Different projections have different variance
- SVD orders projections in decreasing order of variance
- Criterion for choosing when to stop
  - Choose  $k$  so that a desired fraction of the variance is “explained”

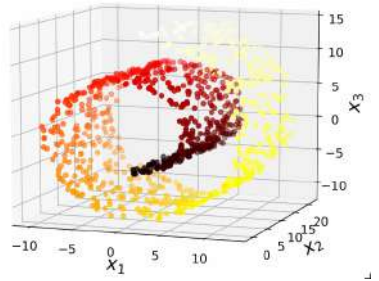


# Manifold learning

- Projection may not always help

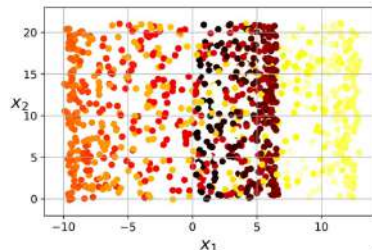
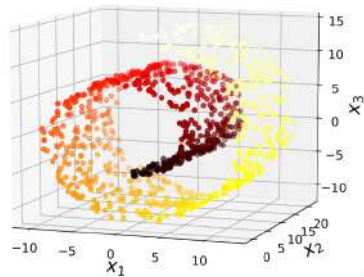
# Manifold learning

- Projection may not always help
- Swiss roll dataset



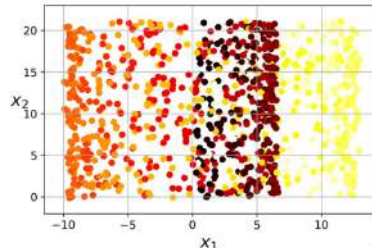
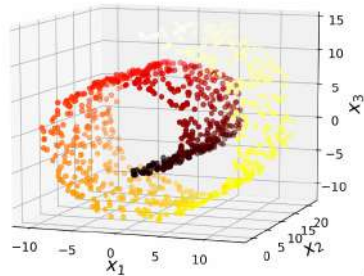
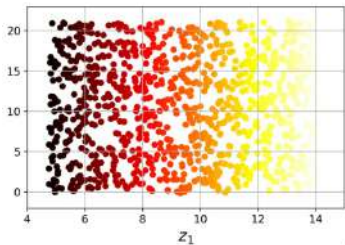
# Manifold learning

- Projection may not always help
- Swiss roll dataset
- Projection onto 2 dimesions is not useful



# Manifold learning

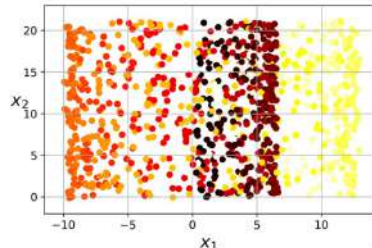
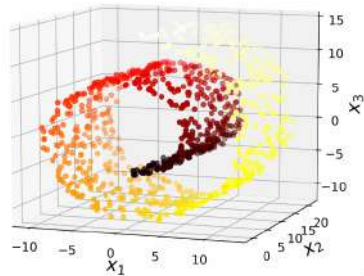
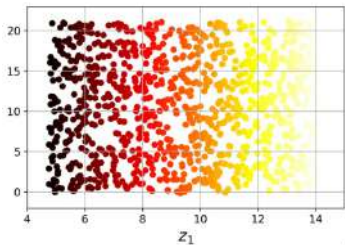
- Projection may not always help
- Swiss roll dataset
- Projection onto 2 dimesions is not useful
- Better to **unroll** the image





# Manifold learning

- Projection may not always help
- Swiss roll dataset
- Projection onto 2 dimesions is not useful
- Better to **unroll** the image



- Discover the **manifold** along which the data lies