Lecture 15: 18 March, 2025

Madhavan Mukund

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

Data Mining and Machine Learning
January–April 2025

The curse of dimensionality

- ML data is often high dimensional — especially images

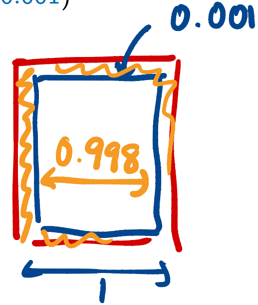
  8x8 → 64

  - A 1000×1000 pixel image has 10^6 features

The curse of dimensionality

- ML data is often high dimensional — especially images
  - A 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
  - $2D$ unit square, 0.4% probability of being near the border (within 0.001)

$$(1 \times 1) - (0.998)^2$$
$$\simeq 0.004$$



The curse of dimensionality

- ML data is often high dimensional — especially images
    - A 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
    - $2D$ unit square, 0.4% probability of being near the border (within 0.001)
    - $10^4 D$ hypercube, 99.999999% probability of being near the border

$$1 - (0.998)^{1000000} = 0.998 - - 9$$

The curse of dimensionality

- ML data is often high dimensional — especially images
  - A 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
  - $2D$ unit square, 0.4% 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

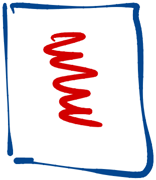
- ML data is often high dimensional — especially images
  - A 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
  - $2D$ unit square, 0.4% probability of being near the border (within 0.001)
  - $10^4 D$ hypercube, 99.999999% probability of being near the border
- 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 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
  - $2D$ unit square, 0.4% 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 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
  - $2D$ unit square, 0.4% 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

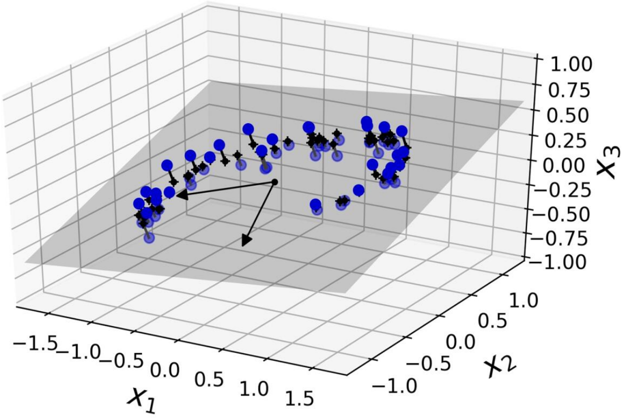
- ML data is often high dimensional — especially images
  - A 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
  - $2D$ unit square, 0.4% probability of being near the border (within 0.001)
  - $10^4 D$ hypercube, 99.999999% probability of being near the border
- 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

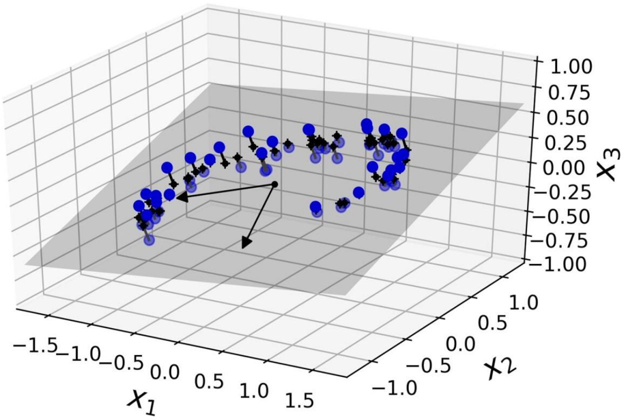
Dimensionality reduction

- 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

Singular Value Decomposition (SVD)

- Input matrix M, dimensions $n \times d$
  - Rows are items, columns are features
- Decompose M as $UDV^\top$
  - 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

d-dim $\rightarrow$ k-dim

M   d dim

k U n

D_k k

V^T $k \times d$

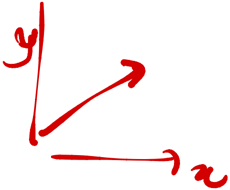
$V = d \times k$

1 2 .. k

1 ⋮ d

Singular Value Decomposition (SVD)

- Input matrix M, dimensions $n \times d$
  - Rows are items, columns are features
- Decompose M as $UDV^\top$
  - 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

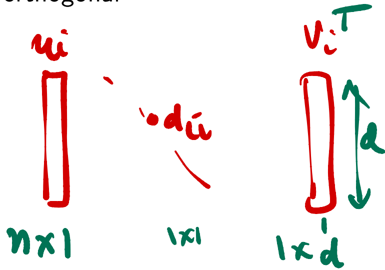


Singular Value Decomposition (SVD)

- Input matrix M, dimensions $n \times d$
  - Rows are items, columns are features
- Decompose M as $UDV^\top$
  - 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 items across features

Singular Value Decomposition (SVD)

- Input matrix M, dimensions $n \times d$
  - Rows are items, columns are features
- Decompose M as $UDV^\top$
  - 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 items across features
  - $M = \sum_{i=1}^{k} D_{ii}(\mathbf{u}_i \cdot \mathbf{v}_i^\top)$



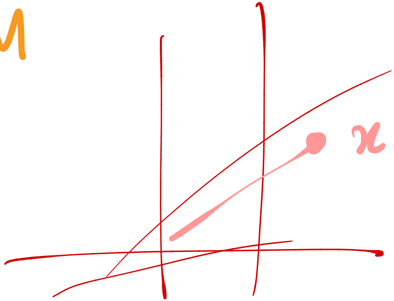
$$\overset{M\cdot d_1}{d_{11} u_1 v_1^T} + \overset{M\cdot d_2}{d_{22} u_2 v_2^T} + \cdots + \overset{M\cdot d_k}{u_k v_k^T d_{kk}}$$

$$\Downarrow$$

$$M$$

$\Rightarrow$ new basis

$v_1, \ldots, v_d$



$x = (x_1, \ldots, x_d)$

$(x\cdot v_1, x\cdot v_2, \ldots, x\cdot v_d)$

Singular Value Decomposition (SVD)

- Input matrix M, dimensions $n \times d$
  - Rows are items, columns are features
- Decompose M as $UDV^\top$
  - 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 items across features
  - $M = \sum_i D_{ii}(\mathbf{u}_i \cdot \mathbf{v}_i^\top)$
  - For columns $\mathbf{u}_i$ of U and $\mathbf{v}_i$ of V, $\mathbf{u}_i \cdot \mathbf{v}_i^\top$ is an $n \times d$ matrix, like M

Singular Value Decomposition (SVD)

- Input matrix M, dimensions $n \times d$
  - Rows are items, columns are features
- Decompose M as $UDV^\top$
  - 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 items across features
  - $M = \sum_i D_{ii}(\boldsymbol{u}_i \cdot \boldsymbol{v}_i^\top)$
  - For columns $\boldsymbol{u}_i$ of U and $\boldsymbol{v}_i$ of V, $\boldsymbol{u}_i \cdot \boldsymbol{v}_i^\top$ is an $n \times d$ matrix, like M
  - $\boldsymbol{u}_i \cdot \boldsymbol{v}_i^\top$ describes components of rows of M along direction $\boldsymbol{v}_i$

Singular vectors

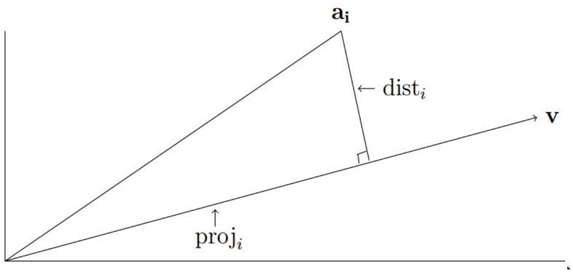
- Unit vectors passing through the origin

Singular vectors

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

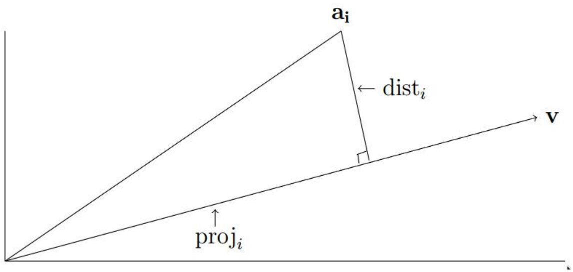
Singular vectors

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



Singular vectors

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



Singular vectors . . .

- Sum of squares of lengths of projections of all rows in M onto $\mathbf{v}$ — $|M\mathbf{v}|^2$

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

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

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

Not unique

v, -v both work

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

Greedy construction

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

- 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 $\max\limits_{\mathbf{v} \perp \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r;\ |\mathbf{v}|=1} |M\mathbf{v}| = 0$
  - r turns out to be the rank of M
  - Vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r\}$ 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 $\displaystyle\max_{\mathbf{v}\perp\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_r;\ |\mathbf{v}|=1} |M\mathbf{v}| = 0$
  - r turns out to be the rank of M
  - Vectors $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_r\}$ 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

$\sigma_1 = |Mv_1|$

$\sigma_2 = |Mv_2|$

$\sigma_3 = |Mv_3|$

$\vdots$

$\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \ldots$

Singular vectors . . .

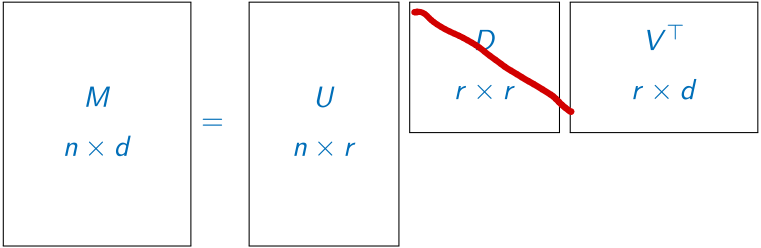
- With each singular vector $\boldsymbol{v}_j$, associated singular value is $\sigma_j = |M\boldsymbol{v}_j|$
- Repeat r times till $\displaystyle\max_{\boldsymbol{v}\perp\boldsymbol{v}_1,\boldsymbol{v}_2,\ldots,\boldsymbol{v}_r;\ |\boldsymbol{v}|=1} |M\boldsymbol{v}| = 0$
  - r turns out to be the rank of M
  - Vectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r\}$ 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 $\boldsymbol{u}_i = \dfrac{1}{\sigma_i} M\boldsymbol{v}_i$

Singular vectors . . .

- With each singular vector $\boldsymbol{v}_j$, associated singular value is $\sigma_j = |M\boldsymbol{v}_j|$
- Repeat r times till $\displaystyle\max_{\boldsymbol{v}\perp\boldsymbol{v}_1,\boldsymbol{v}_2,\ldots,\boldsymbol{v}_r;\ |\boldsymbol{v}|=1} |M\boldsymbol{v}| = 0$
  - r turns out to be the rank of M
  - Vectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r\}$ 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 $\boldsymbol{u}_i = \dfrac{1}{\sigma_i} M\boldsymbol{v}_i$
- Can show that $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_r\}$ are also orthonormal

Singular Value Decomposition

- M, dimension $n \times d$, of rank r uniquely decomposes as $M = UDV^\top$
  - $V = [\boldsymbol{v}_1 \ \boldsymbol{v}_2 \ \cdots \ \boldsymbol{v}_r]$ are the right singular vectors
  - D is a diagonal matrix with $D[i,i] = \sigma_i$, the singular values
  - $U = [\boldsymbol{u}_1 \ \boldsymbol{u}_2 \ \cdots \ \boldsymbol{u}_r]$ are the left singular vectors



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 — $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$

Rank-k approximation

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

Rank-k approximation

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

Rank-k approximation

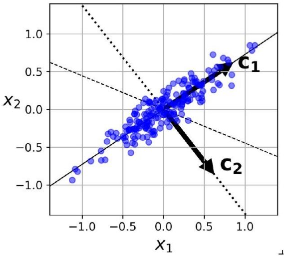
- M has rank r, SVD gives rank r decomposition
- Singular values are non-increasing — $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$
- Suppose we retain only k largest ones
- We have
  - Matrix of first k right singular vectors $V_k = [\boldsymbol{v}_1 \ \boldsymbol{v}_2 \ \cdots \ \boldsymbol{v}_k]$,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

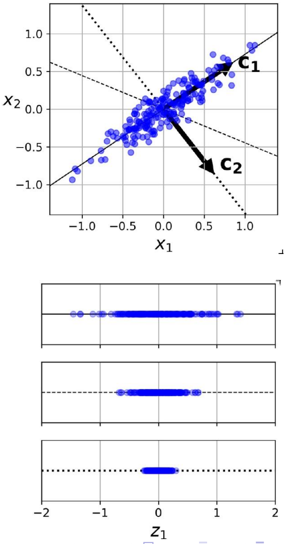
PCA and variance

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



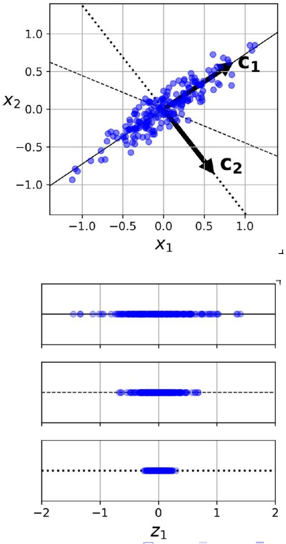
PCA and variance

- 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"

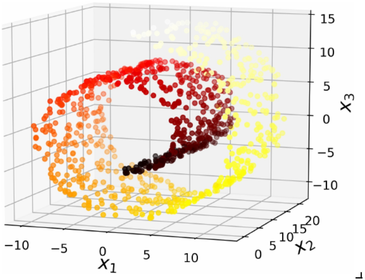


Manifold learning

- Projection may not always help

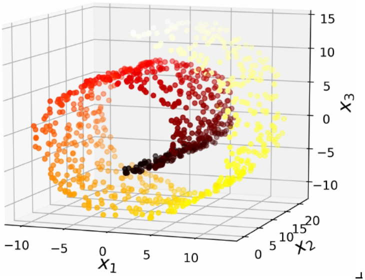
Manifold learning

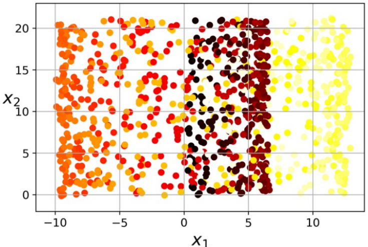
- Projection may not always help
- Swiss roll dataset



Manifold learning

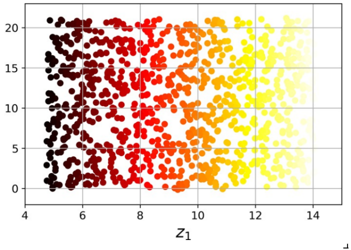
- Projection may not always help
- Swiss roll dataset
- Projection onto 2 dimesions is not useful

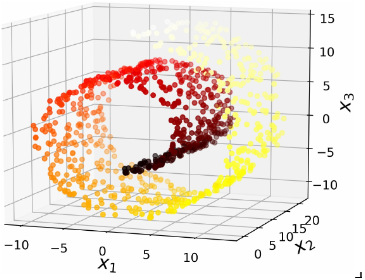


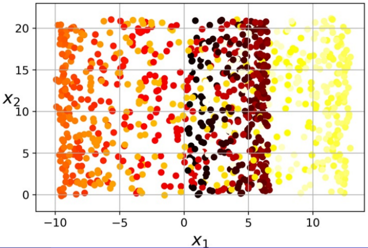


Manifold learning

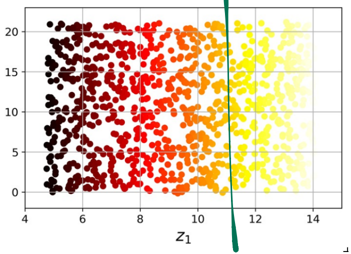
- Projection may not always help
- Swiss roll dataset
- Projection onto 2 dimesions is not useful
- Better to unroll the image



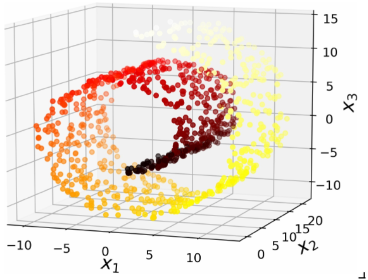


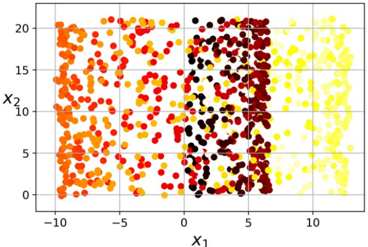


- Projection may not always help
- Swiss roll dataset
- Projection onto 2 dimesions is not useful
- Better to unroll the image



- Discover the manifold along which the data lies





Locally linear embeddings (LLE)

- Describe each point x_i as a linear combination of k nearest neighbours, assume weight 0 for other neighbours

m points

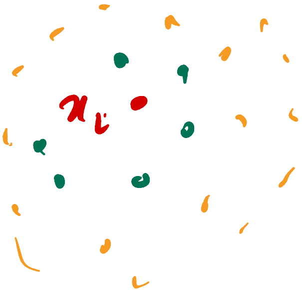
Represent x_i as $\sum_{j=1}^{m} w_{ij} x_j$

?

d dims

$\downarrow$

k dims



Locally linear embeddings (LLE)

- Describe each point x_i as a linear combination of k nearest neighbours, assume weight 0 for other neighbours

$$\text{Represent} \quad x_i \quad \text{as} \quad \sum_{j=1}^{m} w_{ij} x_j$$

- Choose weights to minimize the sum square distance

$$\hat{W} = \arg\min_{W} \sum_{i=1}^{m} \left(x_i - \sum_{j=1}^{m} w_{ij} x_j \right)^2$$

Locally linear embeddings (LLE)

- Describe each point x_i as a linear combination of k nearest neighbours, assume weight 0 for other neighbours

Represent x_i as $\sum_{j=1}^{m} w_{ij} x_j$

- Choose weights to minimize the sum square distance

$$\hat{W} = \arg\min_{W} \sum_{i=1}^{m} \left(x_i - \sum_{j=1}^{m} w_{ij} x_j \right)^2$$

- Normalize weights — captures "local" geometry upto rotation, reflection, scaling

Locally linear embeddings (LLE)

- Describe each point x_i as a linear combination of k nearest neighbours, assume weight 0 for other neighbours

d

Represent x_i as $\sum_{j=1}^{m} w_{ij} x_j$

- Choose weights to minimize the sum square distance

$$\hat{W} = \arg\min_{W} \sum_{i=1}^{m} \left(x_i - \sum_{j=1}^{m} w_{ij} x_j \right)^2$$

$x_i \longmapsto z_i$

$d \qquad J$

- Normalize weights — captures "local" geometry upto rotation, reflection, scaling
- Re-express each point in J dimensions, $x_i \mapsto z_i$

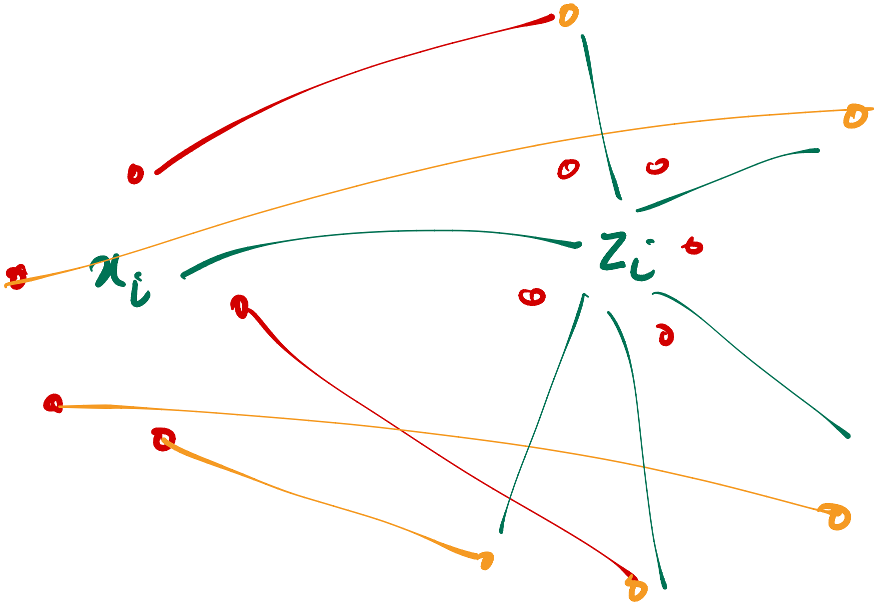
$$\hat{Z} = \arg\min_{Z} \sum_{i=1}^{m} \left(z_i - \sum_{j=1}^{m} w_{ij} z_j \right)^2$$

$x_1 \longmapsto z_1$

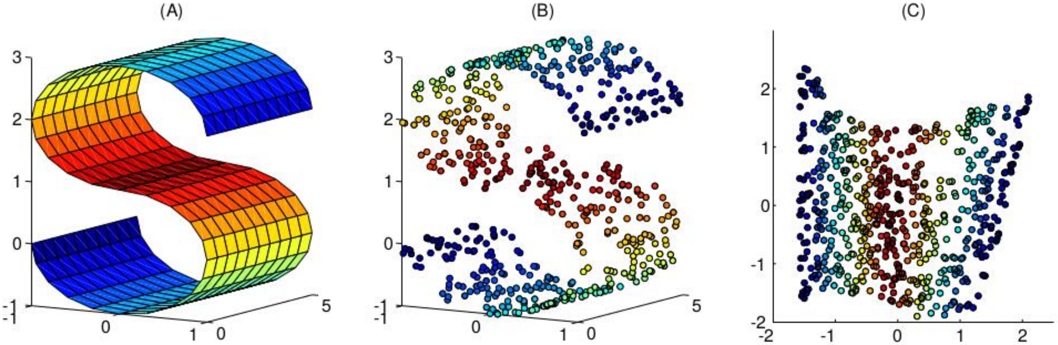
$x_2 \longmapsto z_2$

$\vdots$

$x_m \longmapsto z_m$

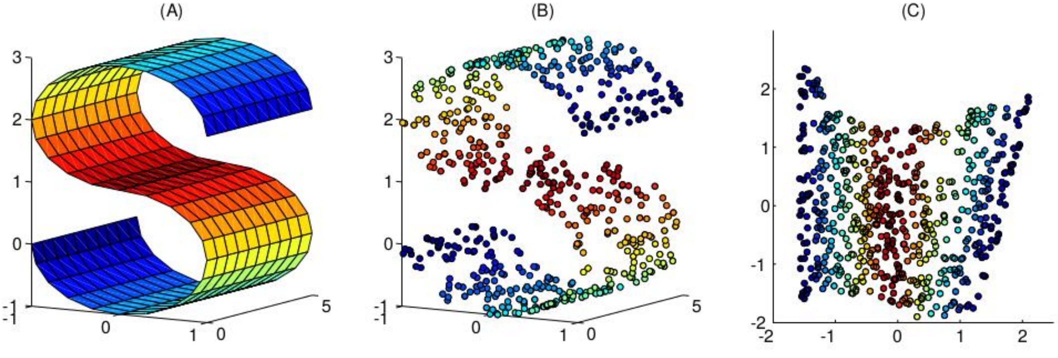

x_i
z_i

Locally linear embeddings (LLE)



Original image　　Sampled points　　LLE reconstruction

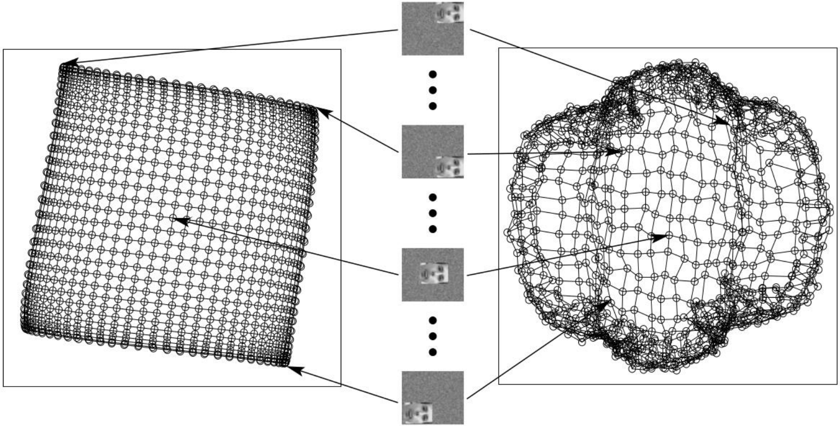
Locally linear embeddings (LLE)



Original image | Sampled points | LLE reconstruction

- Need enough samples to discover the "curves"

Locally linear embeddings (LLE)

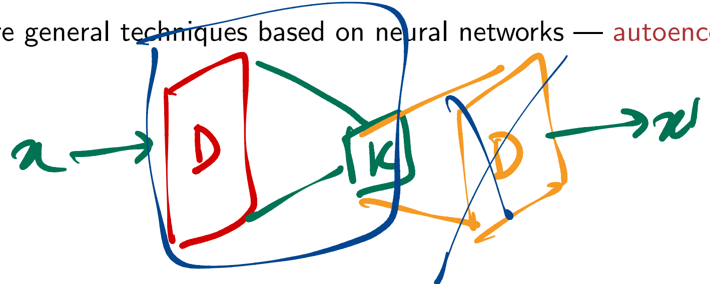


LLE reconstruction preserves neighbourhood structure

PCA distorts geometry

Summary

- Singular Value Decomposition (SVD) finds best fit k-dimensional subspace for any matrix M
- Principal Component Analysis uses SVD for dimensionality reduction
- Unsupervised technique — often helps simplify the problem, but may not
- SVD/PCA can only compress features that have a linear relationship
- More general techniques based on neural networks — autoencoders



Loss (x, x')