Lecture 15: 21 March, 2022

Madhavan Mukund

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

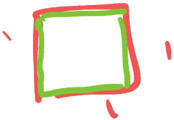
Data Mining and Machine Learning
January–May 2022

The curse of dimensionality

- ML data is often high dimensional — especially images
  - 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)



The curse of dimensionality

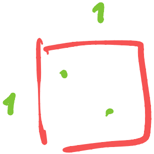
- ML data is often high dimensional — especially images
  - A 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
  - $2D$ unit square, 0.4% probability of being near the border (within 0.001)
  - $10^4 D$ hypercube, 99.999999% probability of being near the border

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

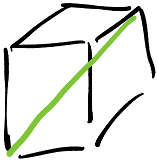
The curse of dimensionality

- 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

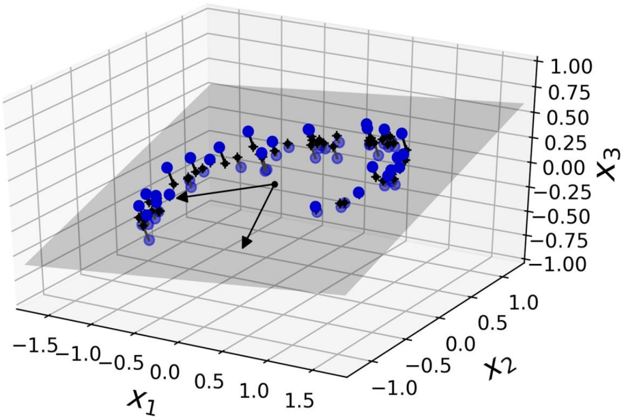
- ML data is often high dimensional — especially images
  - A 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
  - $2D$ unit square, 0.4% probability of being near the border (within 0.001)
  - $10^4 D$ hypercube, 99.999999% probability of being near the border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

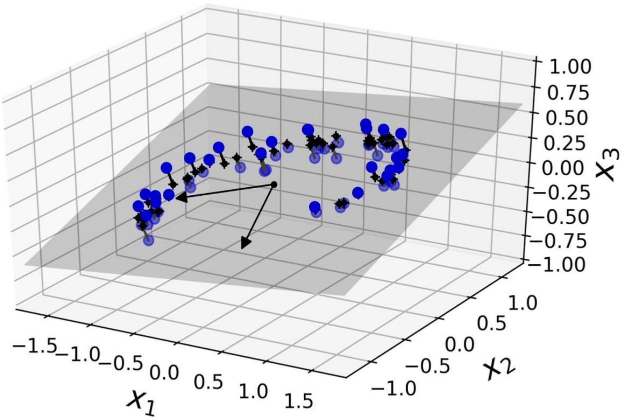
Dimensionality reduction

- 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

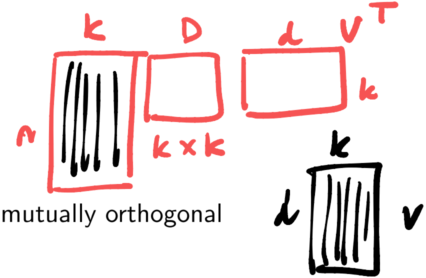


Singular Value Decomposition (SVD)

- 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



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 terms 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 terms across features
  - $M = \sum_i D_{ii}(\mathbf{u}_i \cdot \mathbf{v}_i^\top)$

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 terms 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

nxk kxd

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 terms 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
  - $\mathbf{u}_i \cdot \mathbf{v}_i^\top$ describes components of rows of M along direction $\mathbf{v}_i$

Singular vectors

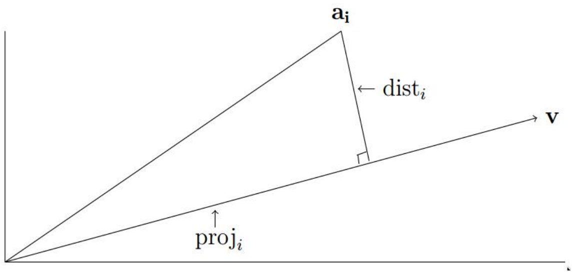
- Unit vectors passing through the origin

Singular vectors

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

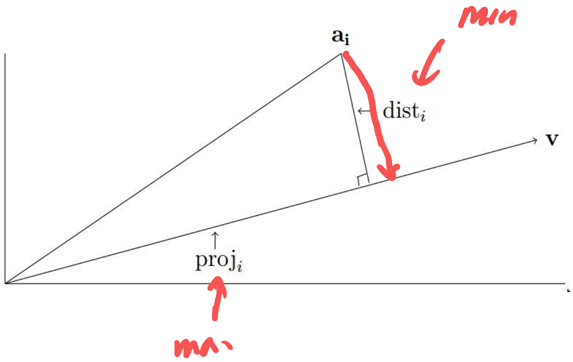
Singular vectors

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



Singular vectors . . .

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

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

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

If $|Mv_3| > |Mv_2|$ we would have picked v_3 before v_2

- 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 $\displaystyle\max_{\mathbf{v}\perp\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_r;\ |\mathbf{v}|=1} |M\mathbf{v}| = 0$
  - 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

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

Shrink M from d dimension to r dimension

- 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
- Corresponding left singular vectors are given by $\mathbf{u}_i = \dfrac{1}{\sigma_i} M\mathbf{v}_i$

singular values

$M = U D V^T$ ← right sing. vectors

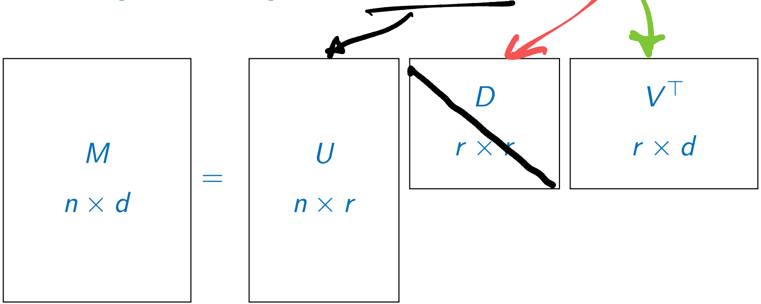
left sing vectors — $n \times k$ $V: d \times k$

Singular vectors ...

- With each singular vector $\mathbf{v}_j$, associated singular value is $\sigma_j = |M\mathbf{v}_j|$
- Repeat r times till $\max_{\mathbf{v}\perp\mathbf{v}_1,\mathbf{v}_2,\ldots,\mathbf{v}_r;\ |\mathbf{v}|=1} |M\mathbf{v}| = 0$
  - 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
- Corresponding left singular vectors are given by $\mathbf{u}_i = \frac{1}{\sigma_i} M\mathbf{v}_i$
- Can show that $\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\}$ are also orthonormal

Singular Value Decomposition

- M, dimension $n \times d$, of rank r uniquely decomposes as $M = UDV^\top$
  - $V = [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_r]$ are the right singular vectors
  - D is a diagonal matrix with $D[i,i] = \sigma_i$, the singular values
  - $U = [\mathbf{u}_1 \ \mathbf{u}_2 \ \cdots \ \mathbf{u}_r]$ 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 — $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$

$$\sigma_j = |M \cdot v_j|$$

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

$k \leq r$

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
- 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

$M = U_r D_r V_r^T$

Rank-k approximation

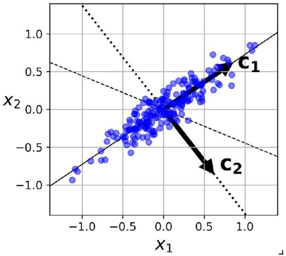
- M has rank r, SVD gives rank r decomposition
- Singular values are non-increasing — $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$
- Suppose we retain only k largest ones
- We have
  - Matrix of first k right singular vectors $V_k = [\mathbf{v}_1 \; \mathbf{v}_2 \; \cdots \; \mathbf{v}_k]$,
  - Corresponding singular values $\sigma_1, \sigma_2, \ldots, \sigma_k$
  - Matrix of k left singular vectors $U_k = [\mathbf{u}_1 \; \mathbf{u}_2 \; \cdots \; \mathbf{u}_k]$
- 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

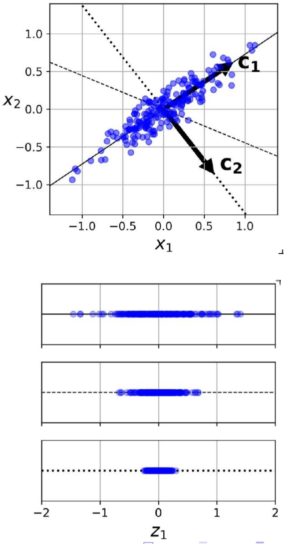
PCA and variance

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



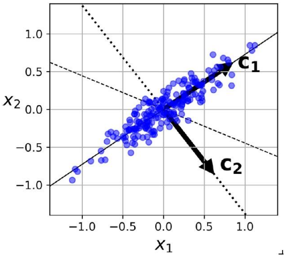
PCA and variance

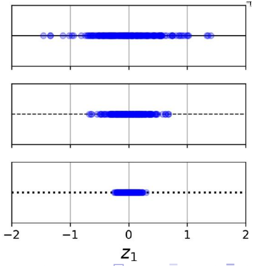
- Interpret PCA in terms of preserving variance
- Different projections have different variance
- SVD orders projections in decreasing order of variance



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"



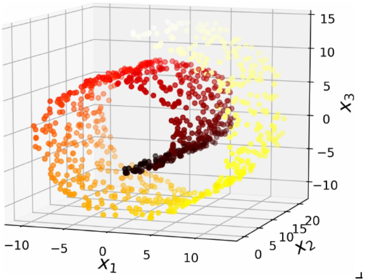


Manifold learning

- Projection may not always help

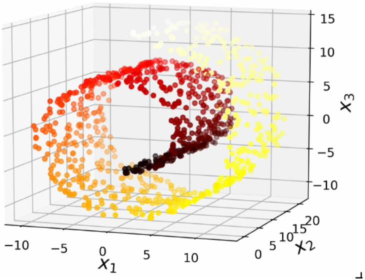
Manifold learning

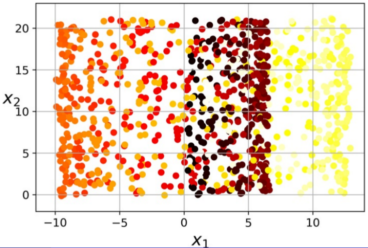
- Projection may not always help
- Swiss roll dataset



Manifold learning

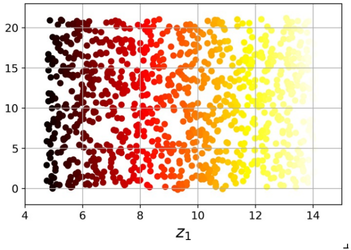
- Projection may not always help
- Swiss roll dataset
- Projection onto 2 dimesions is not useful

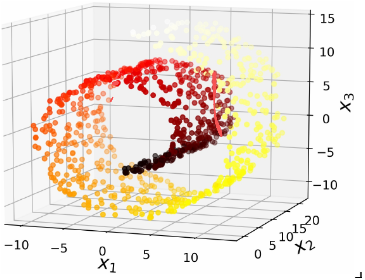


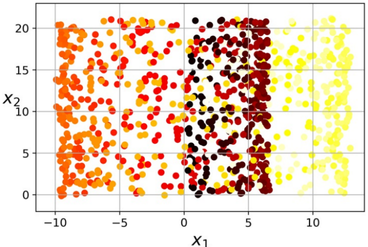


Manifold learning

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

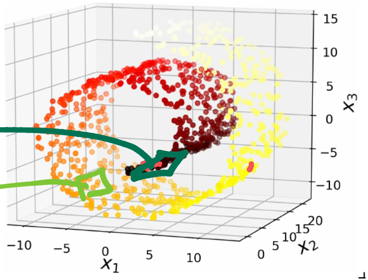


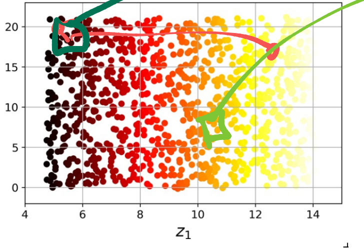


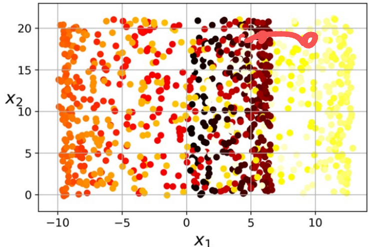


Manifold learning

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







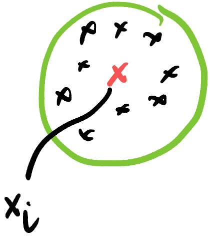
- Discover the manifold along which the data lies

Locally learning embeddings (LLE)

linear

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

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



Locally learning embeddings (LLE)

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

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

- 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 learning embeddings (LLE)

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

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

- Choose weights to minimize the sum square distance

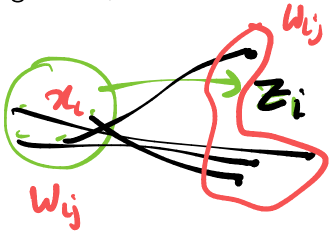
$$\hat{W} = \arg\min_{W} \sum_{i=1}^{m} \left(x_i - \sum_{j=1}^{m} w_{ij} x_j \right)^2$$

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

Locally learning embeddings (LLE)

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

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



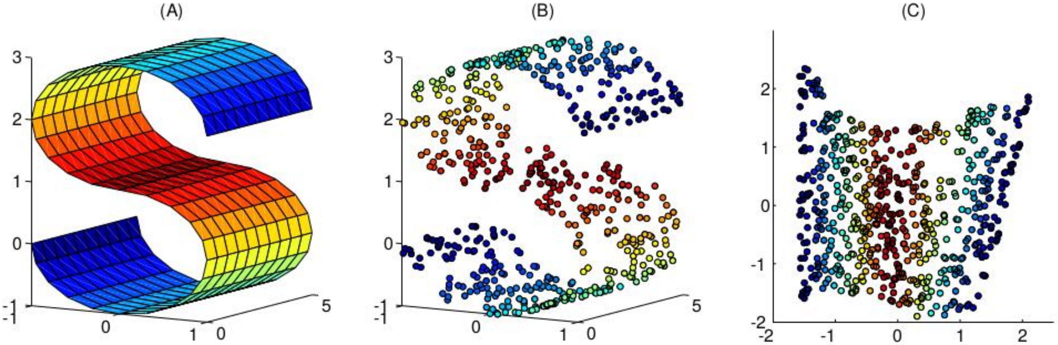
- Choose weights to minimize the sum square distance

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

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

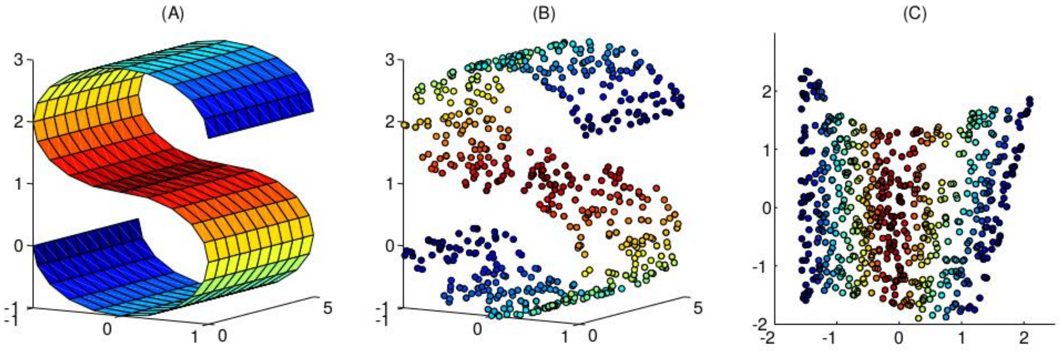
$$\hat{Z} = \arg\min_{Z} \sum_{i=1}^{m} \left(z_i - \sum_{j=1}^{m} w_{ij} z_j \right)^2$$

Locally learning embeddings (LLE)



Original image | Sampled points | LLE reconstruction

Locally learning embeddings (LLE)



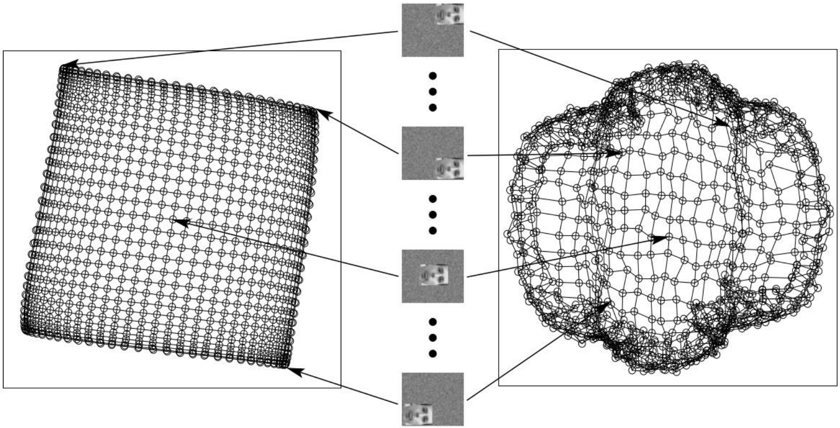
Original image

Sampled points

LLE reconstruction

- Need enough samples to discover the “curves”

Locally learning 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