Lecture 15: 17 March, 2026

Madhavan Mukund

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

Data Mining and Machine Learning
January–April 2026

Mixture models

- Probabilistic process — parameters Θ
  - Tossing a coin with $\Theta = \{Pr(H)\} = \{p\}$
- Perform an experiment
  - Toss the coin N times, $H\ T\ H\ H\ \cdots\ T$
- Estimate parameters from observations
  - From h heads, estimate $p = h/N$
  - Maximum Likelihood Estimator (MLE)
- What if we have a mixture of two random processes
  - Two coins, c_1 and c_2, with $Pr(H) = p_1$ and p_2, respectively
  - Repeat N times: choose c_i with probability $1/2$ and toss it
  - Outcome: N_1 tosses of c_1 interleaved with N_2 tosses of c_2, $N_1 + N_2 = N$
  - Can we estimate p_1 and p_2?

Expectation Maximization (EM)

- Iterative algorithm to estimate the parameters
  - Make an initial guess for the parameters
  - Compute a (fractional) labelling of the outcomes
  - Re-estimate the parameters
- $H\ T\ T\ H\ H\ T\ H\ T\ H\ H\ T\ H\ T\ H\ T\ H\ H\ T\ H\ T$
  - Initial guess: $p_1 = 1/2$, $p_2 = 1/4$
  - $Pr(c_1 = T) = q_1 = 1/2$, $Pr(c_2 = T) = q_2 = 3/4$
  - For each H, likelihood it was c_i, $Pr(c_i \mid H)$, is $p_i/(p_1 + p_2)$
  - For each T, likelihood it was c_i, $Pr(c_i \mid T)$, is $q_i/(q_1 + q_2)$
  - Assign fractional count $Pr(c_i \mid H)$ to each H: $2/3 \times c_1$, $1/3 \times c_2$
  - Likewise, assign fractional count $Pr(c_i \mid T)$ to each T: $2/5 \times c_1$, $3/5 \times c_2$

Expectation Maximization (EM)

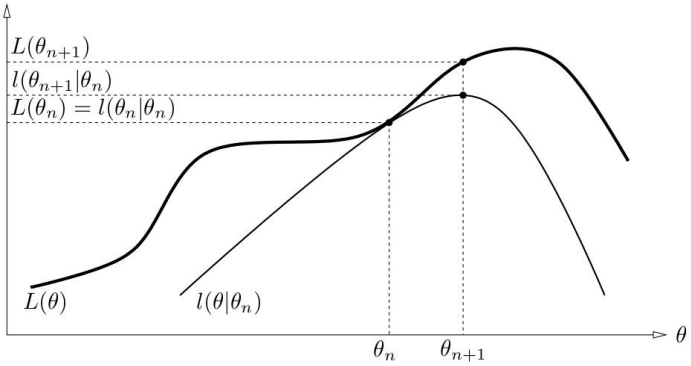
- $H\ T\ T\ H\ H\ T\ H\ T\ H\ H\ T\ H\ T\ H\ T\ H\ H\ T\ H\ T$
- Initial guess: $p_1 = 1/2$, $p_2 = 1/4$
- Fractional counts: each H is $2/3 \times c_1$, $1/3 \times c_2$, each T: $2/5 \times c_1$, $3/5 \times c_2$
- Add up the fractional counts
  - c_1: $11 \cdot (2/3) = 22/3$ heads, $9 \cdot (2/5) = 18/5$ tails
  - c_2: $11 \cdot (1/3) = 11/3$ heads, $9 \cdot (3/5) = 27/5$ tails
- Re-estimate the parameters
  - $p_1 = \dfrac{22/3}{22/3 + 18/5} = 110/164 = 0.67$, $q_1 = 1 - p_1 = 0.33$
  - $p_2 = \dfrac{11/3}{11/3 + 27/5} = 55/136 = 0.40$, $q_2 = 1 - p_2 = 0.60$
- Repeat until convergence

Expectation Maximization (EM)

- Mixture of probabilistic models $(M_1, M_2, \ldots, M_k)$ with parameters $\Theta = (\theta_1, \theta_2, \ldots, \theta_k)$
- Observation $O = o_1 o_2 \ldots o_N$
- Expectation step
  - Compute likelihoods $Pr(M_i|o_j)$ for each M_i, o_j
- Maximization step
  - Recompute MLE for each M_i using fraction of O assigned using likelihood
- Repeat until convergence
  - Why should it converge?
  - If the value converges, what have we computed?

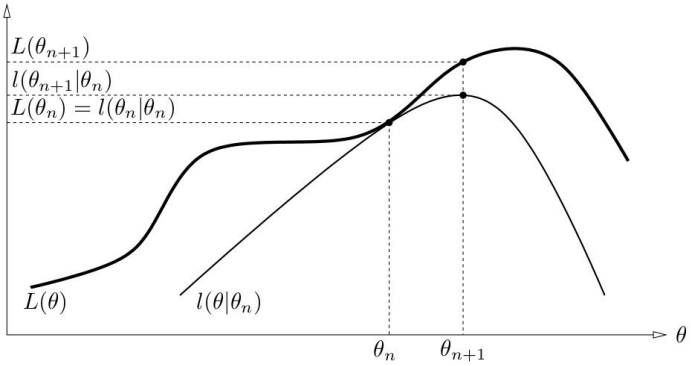
Theoretical foundations of EM

- Mixture of probabilistic models $(M_1, M_2, \ldots, M_k)$ with parameters $\theta = (p_1, p_2, \ldots, p_k)$



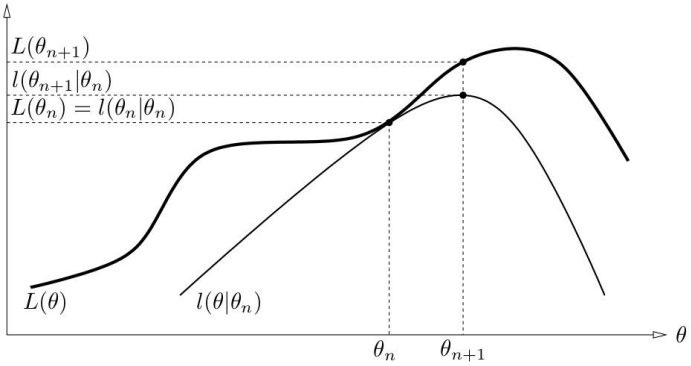
Theoretical foundations of EM

- Mixture of probabilistic models $(M_1, M_2, \ldots, M_k)$ with parameters $\theta = (p_1, p_2, \ldots, p_k)$
- Observation $O = o_1 o_2 \ldots o_N$



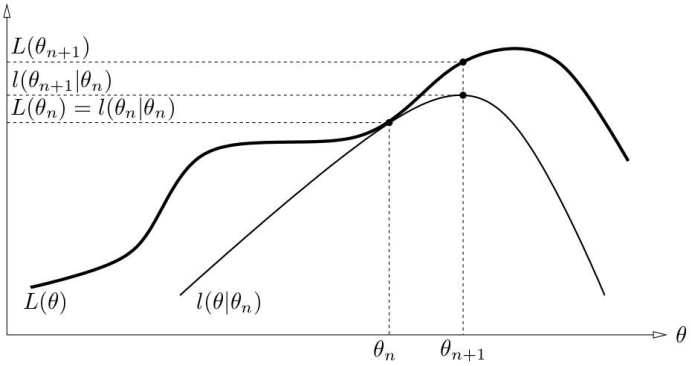
Theoretical foundations of EM

- Mixture of probabilistic models $(M_1, M_2, \ldots, M_k)$ with parameters $\theta = (p_1, p_2, \ldots, p_k)$
- Observation $O = o_1 o_2 \ldots o_N$
- EM builds a sequence of estimates $\theta_1, \theta_2, \ldots, \theta_n$



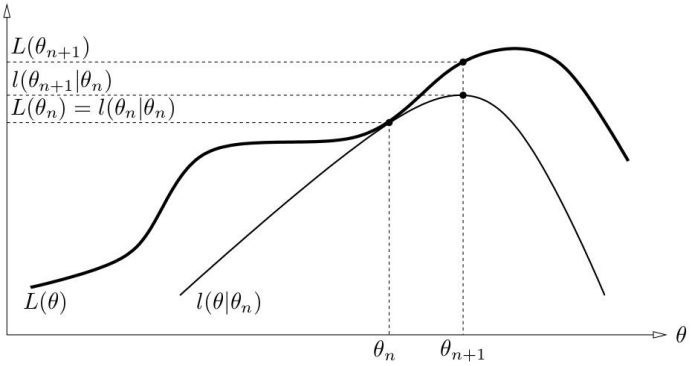
Theoretical foundations of EM

- Mixture of probabilistic models $(M_1, M_2, \ldots, M_k)$ with parameters $\theta = (p_1, p_2, \ldots, p_k)$
- Observation $O = o_1 o_2 \ldots o_N$
- EM builds a sequence of estimates $\theta_1, \theta_2, \ldots, \theta_n$
- $L(\theta_j)$ — log-likelihood function, $\ln Pr(O \mid \theta_j)$



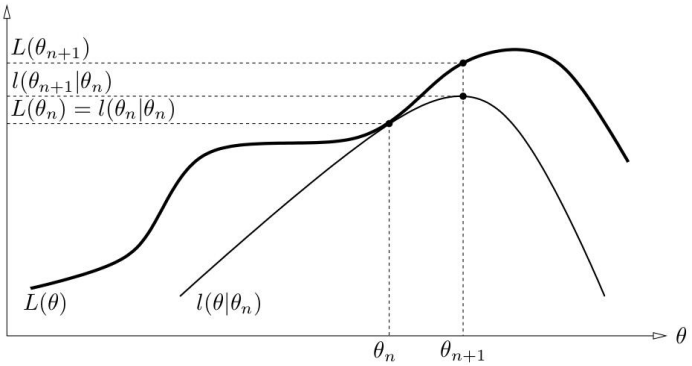
Theoretical foundations of EM

- Mixture of probabilistic models $(M_1, M_2, \ldots, M_k)$ with parameters $\theta = (p_1, p_2, \ldots, p_k)$
- Observation $O = o_1 o_2 \ldots o_N$
- EM builds a sequence of estimates $\theta_1, \theta_2, \ldots, \theta_n$
- $L(\theta_j)$ — log-likelihood function, $\ln Pr(O \mid \theta_j)$
- Want to extend the sequence with θ_{n+1} such that $L(\theta_{n+1}) > L(\theta_n)$



Theoretical foundations of EM

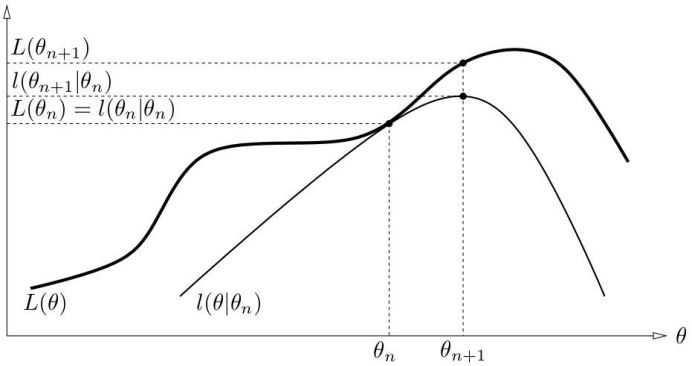
- Mixture of probabilistic models $(M_1, M_2, \ldots, M_k)$ with parameters $\theta = (p_1, p_2, \ldots, p_k)$
- Observation $O = o_1 o_2 \ldots o_N$
- EM builds a sequence of estimates $\theta_1, \theta_2, \ldots, \theta_n$
- $L(\theta_j)$ — log-likelihood function, $\ln Pr(O \mid \theta_j)$
- Want to extend the sequence with θ_{n+1} such that $L(\theta_{n+1}) > L(\theta_n)$



- EM performs a form of gradient descenct

Theoretical foundations of EM

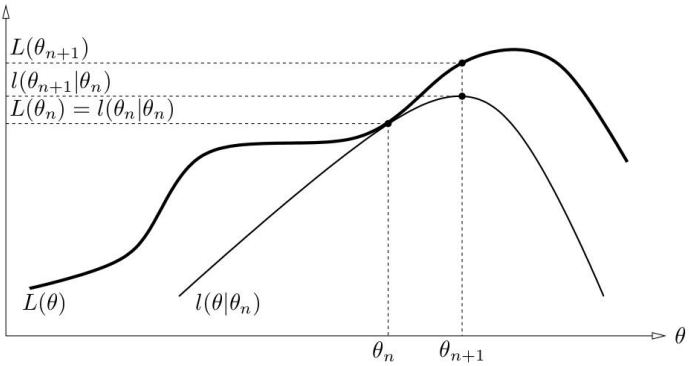
- Mixture of probabilistic models $(M_1, M_2, \ldots, M_k)$ with parameters $\theta = (p_1, p_2, \ldots, p_k)$
- Observation $O = o_1 o_2 \ldots o_N$
- EM builds a sequence of estimates $\theta_1, \theta_2, \ldots, \theta_n$
- $L(\theta_j)$ — log-likelihood function, $\ln Pr(O \mid \theta_j)$
- Want to extend the sequence with θ_{n+1} such that $L(\theta_{n+1}) > L(\theta_n)$



- EM performs a form of gradient descenct
- If we update θ_n to θ' we get a new likelihood $L(\theta_n) + \Delta(\theta', \theta_n)$ which we call $\ell(\theta' \mid \theta_n)$

Theoretical foundations of EM

- Mixture of probabilistic models $(M_1, M_2, \ldots, M_k)$ with parameters $\theta = (p_1, p_2, \ldots, p_k)$
- Observation $O = o_1 o_2 \ldots o_N$
- EM builds a sequence of estimates $\theta_1, \theta_2, \ldots, \theta_n$
- $L(\theta_j)$ — log-likelihood function, $\ln Pr(O \mid \theta_j)$
- Want to extend the sequence with θ_{n+1} such that $L(\theta_{n+1}) > L(\theta_n)$



- EM performs a form of gradient descenct
- If we update θ_n to θ' we get a new likelihood $L(\theta_n) + \Delta(\theta', \theta_n)$ which we call $\ell(\theta' \mid \theta_n)$
- Choose θ_{n+1} to maximize $\ell(\theta' \mid \theta_n)$

Semi-supervised learning

- Supervised learning requires labelled training data

Semi-supervised learning

- Supervised learning requires labelled training data
- What if we don't have enough labelled data?

Semi-supervised learning

- Supervised learning requires labelled training data
- What if we don't have enough labelled data?
- For a probabilistic classifier we can apply EM

Semi-supervised learning

- Supervised learning requires labelled training data
- What if we don't have enough labelled data?
- For a probabilistic classifier we can apply EM
  - Use available training data to assign initial probabilities

Semi-supervised learning

- Supervised learning requires labelled training data
- What if we don't have enough labelled data?
- For a probabilistic classifier we can apply EM
  - Use available training data to assign initial probabilities
  - Label the rest of the data using this model — fractional labels

Semi-supervised learning

- Supervised learning requires labelled training data
- What if we don't have enough labelled data?
- For a probabilistic classifier we can apply EM
  - Use available training data to assign initial probabilities
  - Label the rest of the data using this model — fractional labels
  - Add up counts and re-estimate the parameters

Semi-supervised topic classification

- Each document is a multiset or bag of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$

Semi-supervised topic classification

- Each document is a multiset or bag of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Each topic c has probability $Pr(c)$

Semi-supervised topic classification

- Each document is a multiset or bag of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Each topic c has probability $Pr(c)$
- Each word $w_i \in V$ has conditional probability $Pr(w_i \mid c_j)$, for $c_j \in C$
  - Note that $\sum_{i=1}^{m} Pr(w_i \mid c_j) = 1$

Semi-supervised topic classification

- Each document is a multiset or bag of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Each topic c has probability $Pr(c)$
- Each word $w_i \in V$ has conditional probability $Pr(w_i \mid c_j)$, for $c_j \in C$
  - Note that $\sum_{i=1}^{m} Pr(w_i \mid c_j) = 1$
- Assume document length is independent of the class

Semi-supervised topic classification

- Each document is a multiset or bag of words over a vocabulary $V = \{w_1, w_2, \ldots, w_m\}$
- Each topic c has probability $Pr(c)$
- Each word $w_i \in V$ has conditional probability $Pr(w_i \mid c_j)$, for $c_j \in C$
  - Note that $\sum_{i=1}^{m} Pr(w_i \mid c_j) = 1$
- Assume document length is independent of the class
- Only a small subset of documents is labelled
  - Use this subset for initial estimate of $Pr(c)$, $Pr(w_i \mid c_j)$

Semi-supervised topic classification

- Current model $Pr(c)$, $Pr(w_i \mid c_j)$

Semi-supervised topic classification

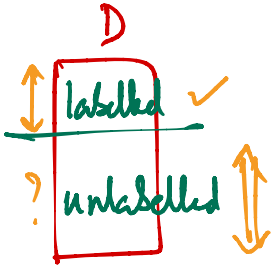
- Current model $Pr(c)$, $Pr(w_i \mid c_j)$
- Compute $Pr(c_j \mid d)$ for each unlabelled document d
  - Normally we assign the maximum among these as the class for d
  - Here we keep fractional values

Semi-supervised topic classification

- Current model $Pr(c)$, $Pr(w_i \mid c_j)$
- Compute $Pr(c_j \mid d)$ for each unlabelled document d
  - Normally we assign the maximum among these as the class for d
  - Here we keep fractional values
- Recompute $Pr(c_j) = \frac{\sum_{d \in D} Pr(c_j \mid D)}{|D|}$
  - For labelled d, $Pr(c_j \mid d) \in \{0, 1\}$
  - For unlabelled d, $Pr(c_j \mid d)$ is fractional value computed from current parameters

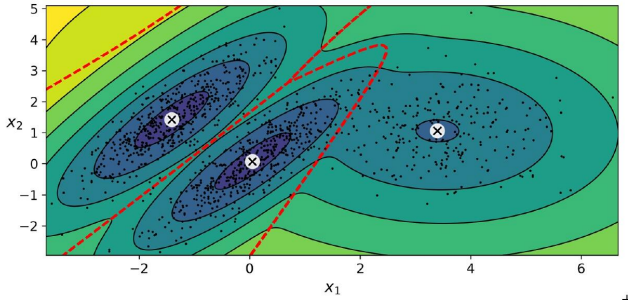
Semi-supervised topic classification

- Current model $Pr(c)$, $Pr(w_i \mid c_j)$
- Compute $Pr(c_j \mid d)$ for each unlabelled document d
  - Normally we assign the maximum among these as the class for d
  - Here we keep fractional values
- Recompute $Pr(c_j) = \dfrac{\sum_{d \in D} Pr(c_j \mid D)}{|D|}$
  - For labelled d, $Pr(c_j \mid d) \in \{0, 1\}$
  - For unlabelled d, $Pr(c_j \mid d)$ is fractional value computed from current parameters
- Recompute $Pr(w_i \mid c_j)$ — fraction of occurrences of w_i in documents labelled c_j
  - n_{id} — occurrences of w_i in d
  - $Pr(w_i \mid c_j) = \dfrac{\sum_{d \in D} n_{id} \; Pr(c_j \mid d)}{\sum_{t=1}^{m} \sum_{d \in D} n_{td} \; Pr(c_j \mid d)}$



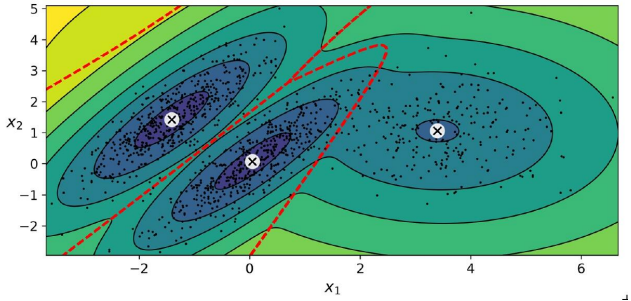
Clustering

- Data points from a mixture of Gaussian distributions



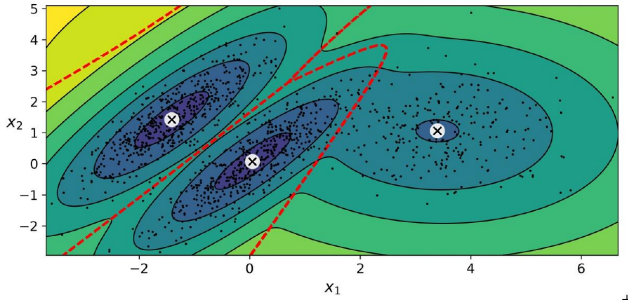
Clustering

- Data points from a mixture of Gaussian distributions
- Use EM to estimate the parameters of each Gaussian distribution



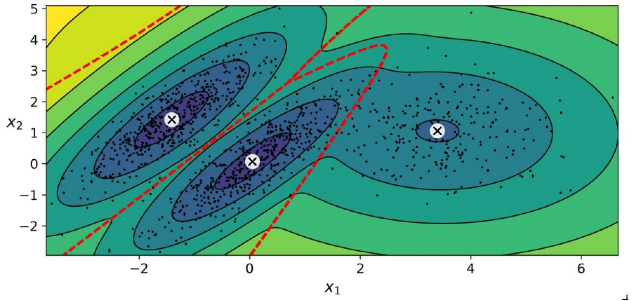
Clustering

- Data points from a mixture of Gaussian distributions
- Use EM to estimate the parameters of each Gaussian distribution
- Assign each point to "best" Gaussian



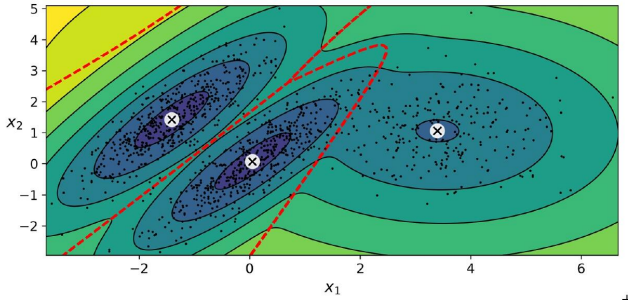
Clustering

- Data points from a mixture of Gaussian distributions
- Use EM to estimate the parameters of each Gaussian distribution
- Assign each point to "best" Gaussian
- Can tweak the shape of the clusters by constraining the covariance matrix



Clustering

- Data points from a mixture of Gaussian distributions
- Use EM to estimate the parameters of each Gaussian distribution
- Assign each point to "best" Gaussian
- Can tweak the shape of the clusters by constraining the covariance matrix
- Outliers are those that are outside $k\sigma$ for all the Gaussians



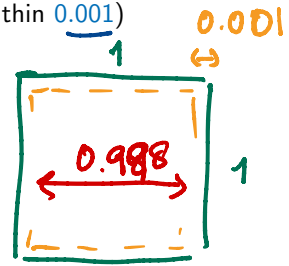
The curse of dimensionality

- 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)

Roughly $0.001 \times 1 \times 4 \approx 0.004$



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

Cube $1 - (0.998)^3$

$1 - (0.998 \times 0.998)$

$\simeq 0.004$

$(0.998)^{10,000} \simeq 10^{-44}$

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

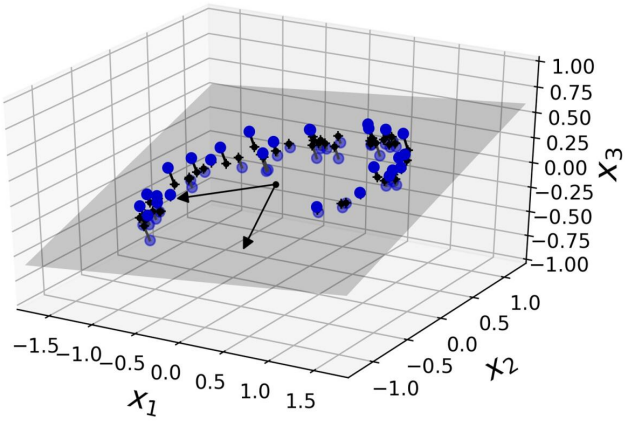
- ML data is often high dimensional — especially images
  - A 1000×1000 pixel image has 10^6 features
- Data behaves very differently in high dimensions
  - $2D$ unit square, 0.4% probability of being near the border (within 0.001)
  - $10^4 D$ hypercube, 99.999999% probability of being near the border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

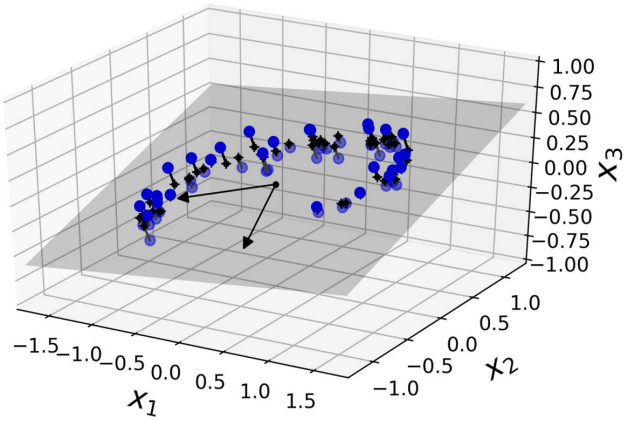
Dimensionality reduction

- 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



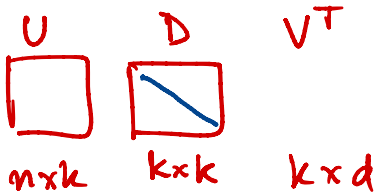
Singular Value Decomposition (SVD)

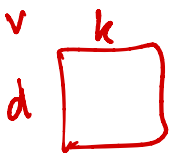
- Input matrix M, dimensions $n \times d$
  - Rows are items, columns are features

n points
d "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 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
  - $M = \sum_i D_{ii}(\boldsymbol{u}_i \cdot \boldsymbol{v}_i^\top)$

UDV^T

$u_i \cdot v_i^T$ $n \times 1$ $1 \times d$ $= n \times d$

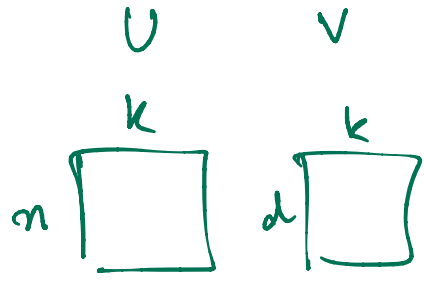
$u_1 v_1$

$M = D_{11} u_1 v_1^T$

$+ D_{22} u_2 v_2^T$

$\vdots$

$+ D_{kk} u_k v_n^T$



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

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

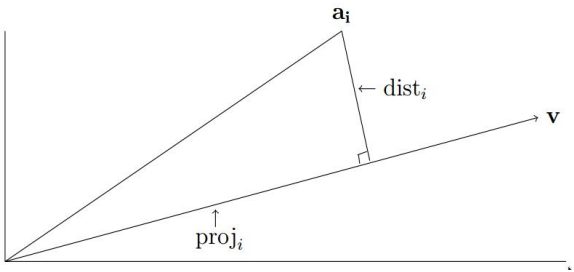
- Unit vectors passing through the origin

Singular vectors

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

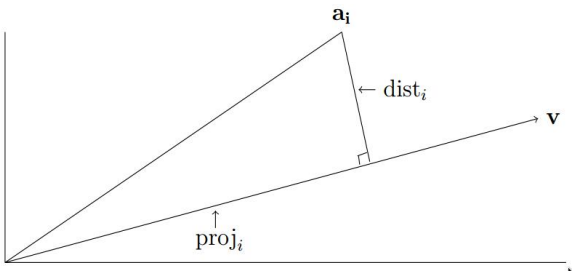
Singular vectors

- 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



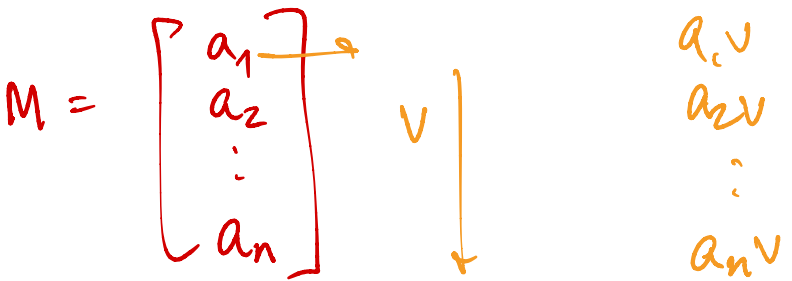
Singular vectors

- Unit vectors passing through the origin
- Want to find "best" k singular vectors to represent feature space
- Suppose we project $\boldsymbol{a}_i = (a_{i1}, a_{i2}, \ldots, a_{id})$ 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 $\boldsymbol{v}$ — $|M\boldsymbol{v}|^2$



Singular vectors . . .

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

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

$v_1 - v$

Singular vectors . . .

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

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

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

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

Singular vectors . . .

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

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

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

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

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

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

Singular vectors . . .

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

$$v_1 \longrightarrow \sigma_1 = |Mv_1|$$
$$v_2 \longrightarrow \sigma_2 = |Mv_2|$$
$$\vdots$$

Singular vectors . . .

- With each singular vector $\boldsymbol{v}_j$, associated singular value is $\sigma_j = |M\boldsymbol{v}_j|$
- Repeat r times till $\max_{\boldsymbol{v} \perp \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r;\ |\boldsymbol{v}|=1} |M\boldsymbol{v}| = 0$
  - 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

Singular vectors . . .

- With each singular vector $\boldsymbol{v}_j$, associated singular value is $\sigma_j = |M\boldsymbol{v}_j|$
- Repeat r times till $\max_{\boldsymbol{v} \perp \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r;\ |\boldsymbol{v}|=1} |M\boldsymbol{v}| = 0$
  - 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

Singular vectors . . .

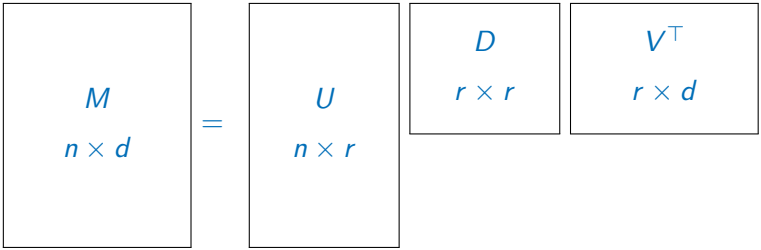
- With each singular vector $\boldsymbol{v}_j$, associated singular value is $\sigma_j = |M\boldsymbol{v}_j|$
- Repeat r times till $\max_{\boldsymbol{v} \perp \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r;\ |\boldsymbol{v}|=1} |M\boldsymbol{v}| = 0$
  - r turns out to be the rank of M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

Singular vectors . . .

- With each singular vector $\boldsymbol{v}_j$, associated singular value is $\sigma_j = |M\boldsymbol{v}_j|$
- Repeat r times till $\max_{\boldsymbol{v} \perp \boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r;\ |\boldsymbol{v}|=1} |M\boldsymbol{v}| = 0$
  - 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 $\boldsymbol{u}_i = \frac{1}{\sigma_i} M\boldsymbol{v}_i$
- 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 — $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$

Pick v_1 to max $|Mv|$

Pick $v_2 \perp v_1$ max $|Mv|$

$\sigma_1 = Mv_1$

$\sigma_2 = Mv_2$

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

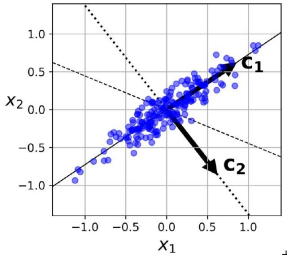
- M has rank r, SVD gives rank r decomposition
- Singular values are non-increasing — $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$
- Suppose we retain only k largest ones
- We have
  - Matrix of first k right singular vectors $V_k = [\boldsymbol{v}_1 \ \boldsymbol{v}_2 \ \cdots \ \boldsymbol{v}_k]$,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

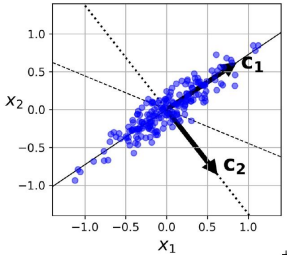
PCA and variance

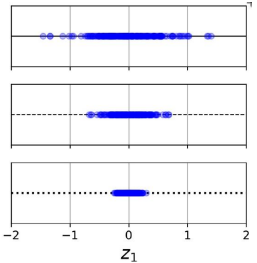
- Interpret PCA in terms of preserving variance
- Different projections have different variance



PCA and variance

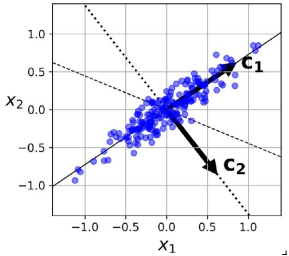
- Interpret PCA in terms of preserving variance
- Different projections have different variance
- SVD orders projections in decreasing order of variance

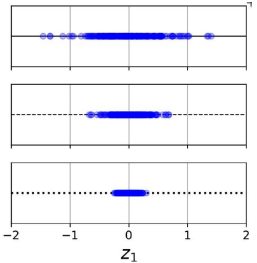




PCA and variance

- 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"



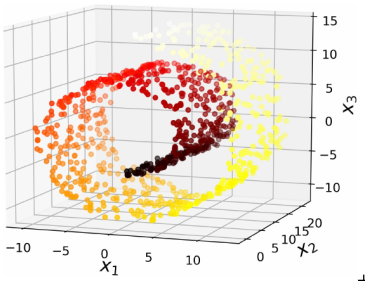


Manifold learning

- Projection may not always help

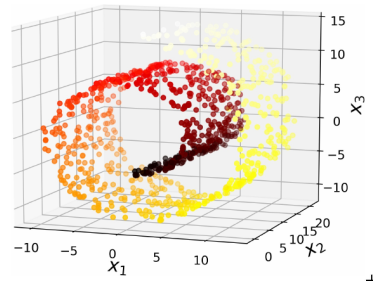
Manifold learning

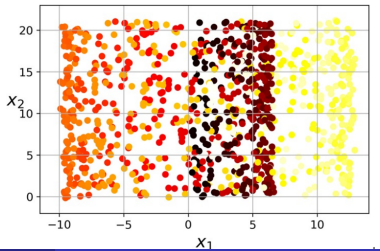
- Projection may not always help
- Swiss roll dataset



Manifold learning

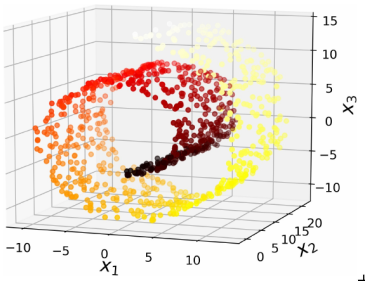
- Projection may not always help
- Swiss roll dataset
- Projection onto 2 dimesions is not useful

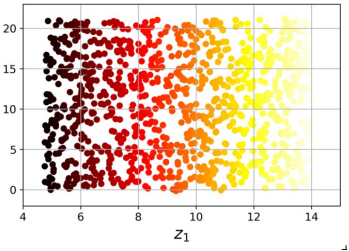


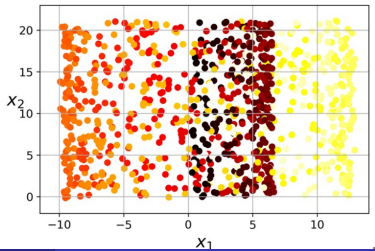


Manifold learning

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

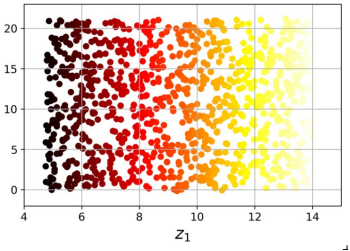


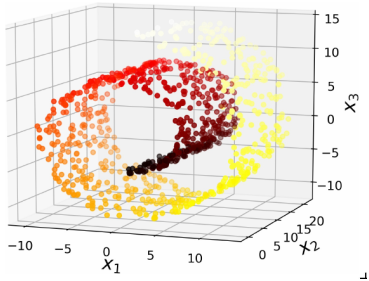


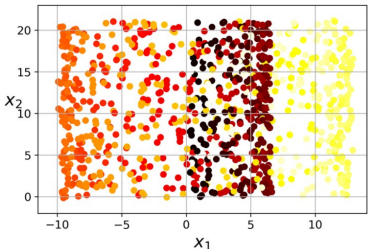


Manifold learning

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







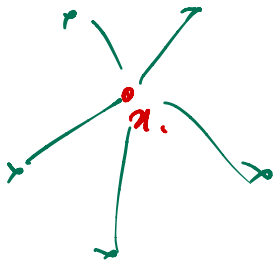
- Discover the manifold along which the data lies

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

$$\sum_j |w_{ij}| = 1$$



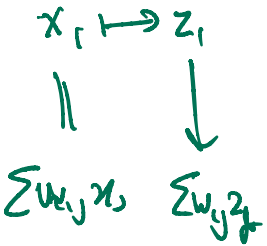
Locally linear embeddings (LLE)

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

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

- Choose weights to minimize the sum square distance

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



Locally linear embeddings (LLE)

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

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

- 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

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

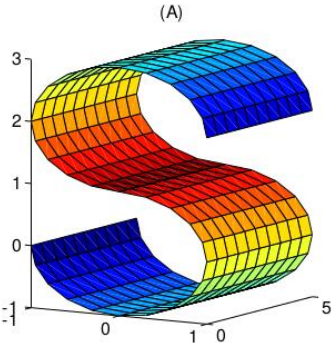
- Choose weights to minimize the sum square distance

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

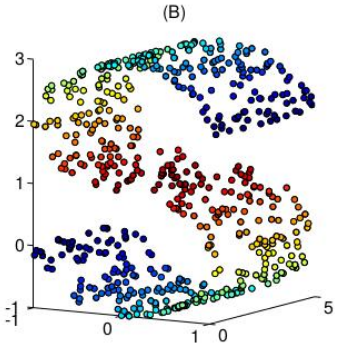
- Normalize weights — captures "local" geometry upto rotation, reflection, scaling
- Re-express each point in J dimensions, $x_i \mapsto z_i$

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

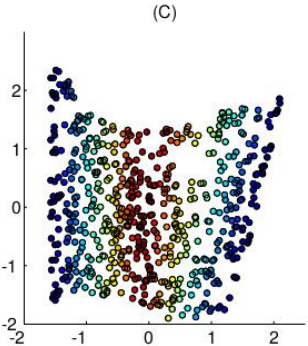
Locally linear embeddings (LLE)



Original image

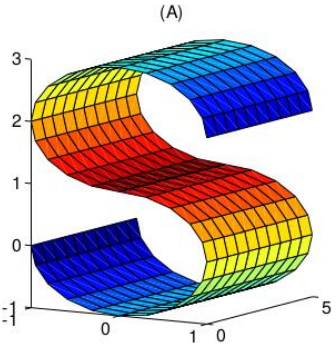


Sampled points

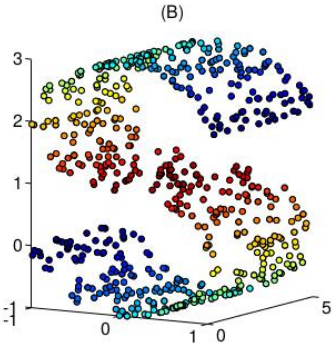


LLE reconstruction

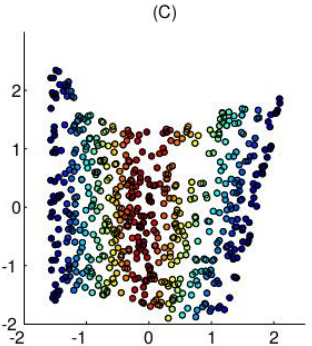
Locally linear embeddings (LLE)



Original image



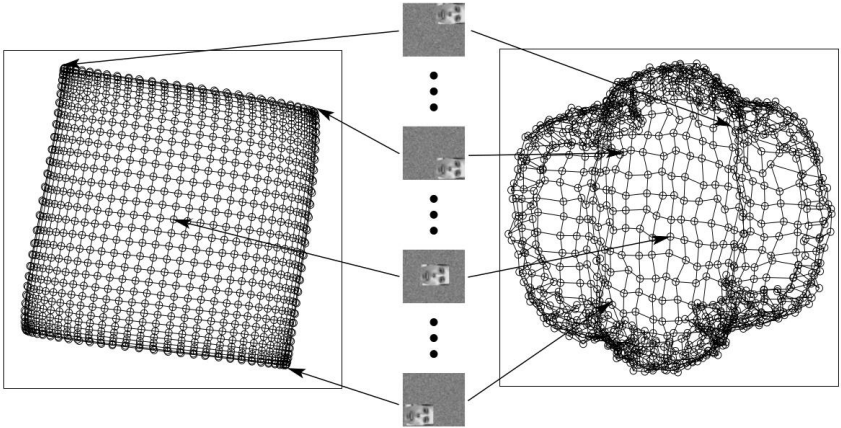
Sampled points



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