DIMENSIONALITY REDUCTION

Borrowing from:
Percy Liang
(Stanford)
Linear Dimensionality Reduction

Idea: Project high-dimensional vector onto a lower dimensional space

\[ x \in \mathbb{R}^{361} \]
\[ z = U^\top x \]
\[ z \in \mathbb{R}^{10} \]
Problem Setup

Given $n$ data points in $d$ dimensions: $x_1, \ldots, x_n \in \mathbb{R}^d$

\[ \mathbf{X} = \left( \begin{array}{c} x_1 \\ \vdots \\ x_n \end{array} \right) \in \mathbb{R}^{d \times n} \]
Problem Setup

Given $n$ data points in $d$ dimensions: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$

$$
\mathbf{X} = \begin{pmatrix} 
\mathbf{x}_1 & \cdots & \mathbf{x}_n 
\end{pmatrix} \in \mathbb{R}^{d \times n}
$$

Want to reduce dimensionality from $d$ to $k$

Choose $k$ directions $\mathbf{u}_1, \ldots, \mathbf{u}_k$

$$
\mathbf{U} = \begin{pmatrix} 
\mathbf{u}_1 & \cdots & \mathbf{u}_k 
\end{pmatrix} \in \mathbb{R}^{d \times k}
$$
Problem Setup

Given \( n \) data points in \( d \) dimensions: \( x_1, \ldots, x_n \in \mathbb{R}^d \)

\[
X = \begin{pmatrix}
    x_1 \\
    \vdots \\
    x_n
\end{pmatrix} \in \mathbb{R}^{d \times n}
\]

Want to reduce dimensionality from \( d \) to \( k \)

Choose \( k \) directions \( u_1, \ldots, u_k \)

\[
U = \begin{pmatrix}
    u_1 \\
    \vdots \\
    u_k
\end{pmatrix} \in \mathbb{R}^{d \times k}
\]

For each \( u_j \), compute “similarity” \( z_j = u_j^\top x \)
Problem Setup

Given \( n \) data points in \( d \) dimensions: \( \mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d \)

\[
\mathbf{X} = \begin{pmatrix}
\mathbf{x}_1 & \cdots & \mathbf{x}_n
\end{pmatrix} \in \mathbb{R}^{d \times n}
\]

Want to reduce dimensionality from \( d \) to \( k \)

Choose \( k \) directions \( \mathbf{u}_1, \ldots, \mathbf{u}_k \)

\[
\mathbf{U} = \begin{pmatrix}
\mathbf{u}_1 & \cdots & \mathbf{u}_k
\end{pmatrix} \in \mathbb{R}^{d \times k}
\]

For each \( \mathbf{u}_j \), compute “similarity” \( z_j = \mathbf{u}_j^\top \mathbf{x} \)

Project \( \mathbf{x} \) down to \( \mathbf{z} = (z_1, \ldots, z_k)^\top = \mathbf{U}^\top \mathbf{x} \)

How to choose \( \mathbf{U} \)?
Principal Component Analysis

Represent each face as a high-dimensional vector

\[ \mathbf{x} \in \mathbb{R}^{361} \]
\[ \mathbf{z} = \mathbf{U}^\top \mathbf{x} \]
\[ \mathbf{z} \in \mathbb{R}^{10} \]

How do we choose \( \mathbf{U} \)?

**Two Objectives**
1. Minimize the reconstruction error
2. Maximize the projected variance
PCA Objective 1: Reconstruction Error

\( \mathbf{U} \) serves two functions:

- **Encode**: \( \mathbf{z} = \mathbf{U}^\top \mathbf{x}, \quad z_j = \mathbf{u}_j^\top \mathbf{x} \)
PCA Objective 1: Reconstruction Error

\( \mathbf{U} \) serves two functions:

- **Encode:** \( \mathbf{z} = \mathbf{U}^\top \mathbf{x}, \quad \mathbf{z}_j = \mathbf{u}_j^\top \mathbf{x} \)

- **Decode:** \( \hat{\mathbf{x}} = \mathbf{U} \mathbf{z} = \sum_{j=1}^{k} \mathbf{z}_j \mathbf{u}_j \)

Want reconstruction error \( k \mathbf{x} \hat{\mathbf{x}} \) to be small

Objective: minimize total squared reconstruction error

\[
\min_{\mathbf{U}} \mathbf{R} \mathbf{d} \cdot \mathbf{k} \mathbf{x} \mathbf{x}^\top \mathbf{U}^\top \mathbf{x}
\]
PCA Objective 1: Reconstruction Error

\( \mathbf{U} \) serves two functions:

- **Encode**: \( \mathbf{z} = \mathbf{U}^\top \mathbf{x}, \quad z_j = \mathbf{u}_j^\top \mathbf{x} \)

- **Decode**: \( \tilde{\mathbf{x}} = \mathbf{U} \mathbf{z} = \sum_{j=1}^{k} \tilde{z}_j \mathbf{u}_j \)

Want reconstruction error \( \|\mathbf{x} - \tilde{\mathbf{x}}\| \) to be small
PCA Objective 1: Reconstruction Error

**U** serves two functions:

- **Encode:** $z = U^T x$, $z_j = u_j^T x$
- **Decode:** $\tilde{x} = Uz = \sum_{j=1}^{k} z_j u_j$

Want reconstruction error $\|x - \tilde{x}\|$ to be small

**Objective:** minimize total squared reconstruction error

$$\min_{U \in \mathbb{R}^{d \times k}} \sum_{i=1}^{n} \|x_i - UU^T x_i\|^2$$
PCA Objective 2: Projected Variance

Empirical distribution: uniform over $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Expectation (think sum over data points):

$$\hat{E}[f(\mathbf{x})] = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i)$$

Variance (think sum of squares if centered):

$$\text{var}[f(\mathbf{x})] + (\hat{E}[f(\mathbf{x})])^2 = \hat{E}[f(\mathbf{x})^2] = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i)^2$$
PCA Objective 2: Projected Variance

Empirical distribution: uniform over \(x_1, \ldots, x_n\)

Expectation (think sum over data points):

\[
\hat{E}[f(x)] = \frac{1}{n} \sum_{i=1}^{n} f(x_i)
\]

Variance (think sum of squares if centered):

\[
\text{var}[f(x)] + (\hat{E}[f(x)])^2 = \hat{E}[f(x)^2] = \frac{1}{n} \sum_{i=1}^{n} f(x_i)^2
\]

Assume data is centered: \(\hat{E}[x] = 0\)
PCA Objective 2: Projected Variance

Empirical distribution: uniform over $\mathbf{x}_1, \ldots, \mathbf{x}_n$

Expectation (think sum over data points):

$$\hat{\mathbb{E}}[f(\mathbf{x})] = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i)$$

Variance (think sum of squares if centered):

$$\text{var}[f(\mathbf{x})] + (\hat{\mathbb{E}}[f(\mathbf{x})])^2 = \hat{\mathbb{E}}[f(\mathbf{x})^2] = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i)^2$$

Assume data is centered: $\hat{\mathbb{E}}[\mathbf{x}] = 0$ (what’s $\hat{\mathbb{E}}[\mathbf{U}^\top \mathbf{x}]$?)
PCA Objective 2: Projected Variance

Empirical distribution: uniform over $x_1, \ldots, x_n$

Expectation (think sum over data points):

$$\hat{E}[f(x)] = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$$

Variance (think sum of squares if centered):

$$\text{var}[f(x)] + (\hat{E}[f(x)])^2 = \hat{E}[f(x)^2] = \frac{1}{n} \sum_{i=1}^{n} f(x_i)^2$$

Assume data is centered: $\hat{E}[x] = 0$ (what’s $\hat{E}[U^\top x]$?)

Objective: maximize variance of projected data

$$\max_{U \in \mathbb{R}^{d \times k}, U^\top U = I} \hat{E}[\|U^\top x\|^2]$$
PCA Objective 2: Projected Variance

Empirical distribution: uniform over \( \mathbf{x}_1, \ldots, \mathbf{x}_n \)

Expectation (think sum over data points):

\[
\hat{E}[f(\mathbf{x})] = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i)
\]

Variance (think sum of squares if centered):

\[
\hat{\text{var}}[f(\mathbf{x})] + (\hat{E}[f(\mathbf{x})])^2 = \hat{E}[f(\mathbf{x})^2] = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i)^2
\]

Assume data is centered: \( \hat{E}[\mathbf{x}] = 0 \) (what’s \( \hat{E}[\mathbf{U}^\top \mathbf{x}] \)?)

Objective: maximize variance of projected data

\[
\max_{\mathbf{U} \in \mathbb{R}^{d \times k}} \text{subject to } \mathbf{U}^\top \mathbf{U} = \mathbf{I} \quad \hat{E}[\| \mathbf{U}^\top \mathbf{x} \|^2]
\]

\[
\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \delta_{i,j}
\]
Equivalence of two objectives

Key intuition:

\[
\text{variance of data} = \underbrace{\text{captured variance}}_{\text{fixed}} + \underbrace{\text{reconstruction error}}_{\text{want large}}
\]

\[
\text{want small}
\]
Equivalence of two objectives

Key intuition:
\[
\text{variance of data} = \underbrace{\text{captured variance}}_{\text{fixed}} + \underbrace{\text{reconstruction error}}_{\text{want small}}
\]

Pythagorean decomposition: \( x = \mathbf{U}\mathbf{U}^\top x + (I - \mathbf{U}\mathbf{U}^\top)x \)

Take expectations; note rotation \( \mathbf{U} \) doesn’t affect length:
\[
\hat{\mathbb{E}}[\|x\|^2] = \hat{\mathbb{E}}[\|\mathbf{U}^\top x\|^2] + \hat{\mathbb{E}}[\|x - \mathbf{U}\mathbf{U}^\top x\|^2]
\]
Equivalence of two objectives

Key intuition:

\[
\text{variance of data} = \text{captured variance} + \text{reconstruction error}
\]

Pythagorean decomposition:

\[
x = UU^T x + (I - UU^T)x
\]

Take expectations; note rotation \( U \) doesn’t affect length:

\[
\hat{E}[\|x\|^2] = \hat{E}[\|UU^T x\|^2] + \hat{E}[\|x - UU^T x\|^2]
\]

Minimize reconstruction error \( \leftrightarrow \) Maximize captured variance
Finding one principal component

Objective: maximize variance of projected data

Input data:

\[ \mathbf{X} = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix} \]
Finding one principal component

Input data:

\[ \mathbf{X} = \begin{pmatrix} x_1 & \ldots & x_n \end{pmatrix} \]

Objective: maximize variance of projected data

\[ \text{max} \ \mathbb{E}[\mathbf{u}^\top \mathbf{x}^2] \]

\( \mathbf{C} \) is the covariance matrix of data

\[ \text{def} = \frac{1}{n} \mathbf{X}^\top \mathbf{X} \]

\[ \mathbf{u}^\top = \text{largest eigenvalue of } \mathbf{C} \]
Finding one principal component

**Input data:**

\[
X = \begin{pmatrix}
  x_1 \\
  \vdots \\
  x_n
\end{pmatrix}
\]

**Objective:** maximize variance of projected data

\[
\begin{align*}
  &= \max_{\|u\|=1} \mathbb{E}[(u^\top x)^2] \\
  &= \max_{\|u\|=1} \frac{1}{n} \sum_{i=1}^{n} (u^\top x_i)^2
\end{align*}
\]
Finding one principal component

Input data:

\[ X = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix} \]

Objective: maximize variance of projected data

\[
= \max_{\|u\|=1} \mathbb{E}[u^\top x]^2 \\
= \max_{\|u\|=1} \frac{1}{n} \sum_{i=1}^{n} (u^\top x_i)^2 \\
= \max_{\|u\|=1} \frac{1}{n} \|u^\top X\|^2
\]

\[ u \] = largest eigenvalue of \[ C \]

def \[ = \frac{1}{n} \|X\|^2 \]

Principal component analysis (PCA) / Basic principles 12
Finding one principal component

Input data:

\[ \mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_n \end{pmatrix} \]

Objective: maximize variance of projected data

\[
\begin{aligned}
&= \max_{\|\mathbf{u}\|=1} \mathbb{E}[\mathbf{u}^\top \mathbf{x}^2] \\
&= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{u}^\top \mathbf{x}_i)^2 \\
&= \max_{\|\mathbf{u}\|=1} \frac{1}{n} \|\mathbf{u}^\top \mathbf{X}\|^2 \\
&= \max_{\|\mathbf{u}\|=1} \mathbf{u}^\top \left( \frac{1}{n} \mathbf{X} \mathbf{X}^\top \right) \mathbf{u}
\end{aligned}
\]
Finding one principal component

Input data: 

\[ X = \begin{pmatrix} x_1 & \ldots & x_n \end{pmatrix} \]

Objective: maximize variance of projected data

\[
= \max_{\|u\|=1} \mathbb{E}[(u^\top x)^2] \\
= \max_{\|u\|=1} \frac{1}{n} \sum_{i=1}^{n} (u^\top x_i)^2 \\
= \max_{\|u\|=1} \frac{1}{n} \|u^\top X\|^2 \\
= \max_{\|u\|=1} u^\top \left( \frac{1}{n}XX^\top \right) u \\
= \text{largest eigenvalue of } C \overset{\text{def}}{=} \frac{1}{n}XX^\top
\]
Finding one principal component

Input data:

\[ X = \begin{pmatrix} x_1 \ldots x_n \end{pmatrix} \]

Objective: maximize variance of projected data

\[
\begin{align*}
&= \max_{\|u\|=1} \mathbb{E}[(u^\top x)^2] \\
&= \max_{\|u\|=1} \frac{1}{n} \sum_{i=1}^{n} (u^\top x_i)^2 \\
&= \max_{\|u\|=1} \frac{1}{n} \|u^\top X\|^2 \\
&= \max_{\|u\|=1} u^\top \left( \frac{1}{n} XX^\top \right) u \\
&= \text{largest eigenvalue of } C \overset{\text{def}}{=} \frac{1}{n} XX^\top \\
(C \text{ is covariance matrix of data})
\]
How many components?

- Similar to question of “How many clusters?”
- Magnitude of eigenvalues indicate fraction of variance captured.

Eigenvalues on a face image dataset:

2 3 4 5 6 7 8 9 10 11

287.1 553.6 820.1 1086.7 1353.2

Eigenvalues typically drop sharply, so don’t need that many.

Of course variance isn’t everything...
How many components?

- Similar to question of “How many clusters?”
- Magnitude of eigenvalues indicate fraction of variance captured.
- Eigenvalues on a face image dataset:

![Graph showing eigenvalues]
How many components?

- Similar to question of “How many clusters?”
- Magnitude of eigenvalues indicate fraction of variance captured.
- Eigenvalues on a face image dataset:

![Graph showing eigenvalues](image)

- Eigenvalues typically drop off sharply, so don’t need that many.
- Of course variance isn’t everything...
Computing PCA

Method 1: eigendecomposition

\[ \mathbf{U} \text{ are eigenvectors of covariance matrix } C = \frac{1}{n} \mathbf{XX}^\top \]
Computing PCA

Method 1: eigendecomposition

\[ \mathbf{U} \text{ are eigenvectors of covariance matrix } \mathbf{C} = \frac{1}{n} \mathbf{X} \mathbf{X}^\top \]

Computing \( \mathbf{C} \) already takes \( O(nd^2) \) time (very expensive)
Computing PCA

**Method 1**: eigendecomposition

\[ U \text{ are eigenvectors of covariance matrix } C = \frac{1}{n}XX^{\top} \]

Computing \( C \) already takes \( O(n d^2) \) time (very expensive)

**Method 2**: singular value decomposition (SVD)

Find \( X = U_{d \times d} \Sigma_{d \times n} V_{n \times n}^{\top} \)

where \( U^{\top} U = I_{d \times d}, \quad V^{\top} V = I_{n \times n}, \quad \Sigma \) is diagonal
Computing PCA

Method 1: eigendecomposition

$U$ are eigenvectors of covariance matrix $C = \frac{1}{n}XX^\top$

Computing $C$ already takes $O(nd^2)$ time (very expensive)

Method 2: singular value decomposition (SVD)

Find $X = U_{d \times d} \Sigma_{d \times n} V_{n \times n}^\top$

where $U^\top U = I_{d \times d}, V^\top V = I_{n \times n}, \Sigma$ is diagonal

Computing top $k$ singular vectors takes only $O(ndk)$
Computing PCA

**Method 1: eigendecomposition**

*\( \mathbf{U} \) are eigenvectors of covariance matrix \( \mathbf{C} = \frac{1}{n} \mathbf{X} \mathbf{X}^\top \)

Computing \( \mathbf{C} \) already takes \( O(nd^2) \) time (very expensive)

**Method 2: singular value decomposition (SVD)**

Find \( \mathbf{X} = \mathbf{U}_{d \times d} \Sigma_{d \times n} \mathbf{V}_{n \times n}^\top \)

where \( \mathbf{U}^\top \mathbf{U} = I_{d \times d}, \mathbf{V}^\top \mathbf{V} = I_{n \times n}, \Sigma \) is diagonal

Computing top \( k \) singular vectors takes only \( O(ndk) \)

**Relationship between eigendecomposition and SVD:**

Left singular vectors are principal components \( (\mathbf{C} = \mathbf{U} \Sigma^2 \mathbf{U}^\top) \)
• $d =$ number of pixels
• Each $\mathbf{x}_i \in \mathbb{R}^d$ is a face image
• $x_{ji} =$ intensity of the $j$-th pixel in image $i$

• $d =$ number of pixels
• Each $x_i \in \mathbb{R}^d$ is a face image
• $x_{ji} =$ intensity of the $j$-th pixel in image $i$

\[
X_{d \times n} \approx U_{d \times k} \begin{pmatrix} z_1 \\ \vdots \\ z_n \end{pmatrix}
\]

Idea: $z_i$ more "meaningful" representation of $i$-th face than $x_i$

Can use $z_i$ for nearest-neighbor classification

Much faster: $O(dk + nk)$ time instead of $O(dn)$ when $n, d \ll k$

Why no time savings for linear classifier? Principal component analysis (PCA) / Case studies 18

- $d =$ number of pixels
- Each $x_i \in \mathbb{R}^d$ is a face image
- $x_{ji} =$ intensity of the $j$-th pixel in image $i$

\[
\begin{pmatrix}
X_{d \times n} \\
\end{pmatrix} \approx
\begin{pmatrix}
U_{d \times k} \\
\end{pmatrix} \begin{pmatrix}
Z_{k \times n} \\
\end{pmatrix}
\]

Idea: $z_i$ more “meaningful” representation of $i$-th face than $x_i$
Can use $z_i$ for nearest-neighbor classification

- $d =$ number of pixels
- Each $x_i \in \mathbb{R}^d$ is a face image
- $x_{ji} =$ intensity of the $j$-th pixel in image $i$

\[
X_{d\times n} \approx U_{d\times k} \cdot Z_{k\times n}
\]

Idea: $z_i$ more “meaningful” representation of $i$-th face than $x_i$
Can use $z_i$ for nearest-neighbor classification
Much faster: $O(dk + nk)$ time instead of $O(dn)$ when $n, d \gg k$

- $d = \text{number of pixels}$
- Each $\mathbf{x}_i \in \mathbb{R}^d$ is a face image
- $x_{ji} = \text{intensity of the } j\text{-th pixel in image } i$

$$
\mathbf{X}_{d \times n} \approx \mathbf{U}_{d \times k} \mathbf{Z}_{k \times n}
$$

Idea: $\mathbf{z}_i$ more “meaningful” representation of $i$-th face than $\mathbf{x}_i$

Can use $\mathbf{z}_i$ for nearest-neighbor classification

Much faster: $O(dk + nk)$ time instead of $O(dn)$ when $n, d \gg k$

Why no time savings for linear classifier?
Latent Semantic Analysis [Deerwater 1990]

- \( d \) = number of words in the vocabulary
- Each \( \mathbf{x}_i \in \mathbb{R}^d \) is a vector of word counts
Latent Semantic Analysis [Deerwater 1990]

- $d =$ number of words in the vocabulary
- Each $x_i \in \mathbb{R}^d$ is a vector of word counts
- $x_{ji} =$ frequency of word $j$ in document $i$

\[
\begin{aligned}
X_{d \times n}
& \approx \begin{pmatrix}
\text{stocks:} & 2 & \cdots & 0 \\
\text{chairman:} & 4 & \cdots & 1 \\
\text{the:} & 8 & \cdots & 7 \\
\vdots & \vdots & \ddots & \vdots \\
\text{wins:} & 0 & \cdots & 2 \\
\text{game:} & 1 & \cdots & 3
\end{pmatrix}

& \approx \begin{pmatrix}
0.4 & \cdots & -0.001 \\
0.8 & \cdots & 0.03 \\
0.01 & \cdots & 0.04 \\
\vdots & \ddots & \vdots \\
0.002 & \cdots & 2.3 \\
0.003 & \cdots & 1.9
\end{pmatrix}

& \begin{pmatrix}
Z_1 \\
\vdots \\
Z_n
\end{pmatrix}
\end{aligned}
\]
Latent Semantic Analysis [Deerwater 1990]

• $d =$ number of words in the vocabulary
• Each $x_i \in \mathbb{R}^d$ is a vector of word counts
• $x_{ji} =$ frequency of word $j$ in document $i$

\[
X_{d \times n} \approx U_{d \times k} Z_{k \times n}
\]

How to measure similarity between two documents?
$z_1^T z_2$ is probably better than $x_1^T x_2$
Latent Semantic Analysis [Deerwater 1990]

- \( d \) = number of words in the vocabulary
- Each \( \mathbf{x}_i \in \mathbb{R}^d \) is a vector of word counts
- \( \mathbf{x}_{ji} = \) frequency of word \( j \) in document \( i \)

\[
\mathbf{X}_{d \times n} \approx \mathbf{U}_{d \times k} \mathbf{Z}_{k \times n}
\]

\[
\begin{pmatrix}
\text{stocks:} & 2 & \cdots & 0 \\
\text{chairman:} & 4 & \cdots & 1 \\
\text{the:} & 8 & \cdots & 7 \\
\vdots & \vdots & \ddots & \vdots \\
\text{wins:} & 0 & \cdots & 2 \\
\text{game:} & 1 & \cdots & 3
\end{pmatrix}
\begin{pmatrix}
0.4 & \cdots & -0.001 \\
0.8 & \cdots & 0.03 \\
0.01 & \cdots & 0.04 \\
\vdots & \ddots & \vdots \\
0.002 & \cdots & 2.3 \\
0.003 & \cdots & 1.9
\end{pmatrix}
\begin{pmatrix}
\mathbf{z}_1 \\
\vdots \\
\mathbf{z}_n
\end{pmatrix}
\]

How to measure similarity between two documents?
\[
\mathbf{z}_1^\top \mathbf{z}_2 \text{ is probably better than } \mathbf{x}_1^\top \mathbf{x}_2
\]

Applications: information retrieval
Note: no computational savings; original \( \mathbf{x} \) is already sparse
Network anomaly detection [Lakhina 2005]

\[ x_{ji} = \text{amount of traffic on link } j \text{ in the network during each time interval } i \]
Network anomaly detection [Lakhina 2005]

\[ x_{ji} = \text{amount of traffic on link } j \text{ in the network during each time interval } i \]

Model assumption: total traffic is sum of flows along a few “paths”

Apply PCA: each principal component intuitively represents a “path”

Anomaly when traffic deviates from first few principal components

[Diagram showing normal and anomalous traffic patterns]
Multi-task learning [Ando & Zhang 2005]

- Have $n$ related tasks (classify documents for various users)
- Each task has a linear classifier with weights $x_i$
- Want to share structure between classifiers
Multi-task learning [Ando & Zhang 2005]

- Have $n$ related tasks (classify documents for various users)
- Each task has a linear classifier with weights $x_i$
- Want to share structure between classifiers

One step of their procedure:
- Given $n$ linear classifiers $x_1, \ldots, x_n$, run PCA to identify shared structure:

$$X = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix} \approx UZ$$

Each principal component is a eigen-classifier
Multi-task learning [Ando & Zhang 2005]

• Have \( n \) related tasks (classify documents for various users)
• Each task has a linear classifier with weights \( x_i \)
• Want to share structure between classifiers

One step of their procedure:

given \( n \) linear classifiers \( x_1, \ldots, x_n \),
run PCA to identify shared structure:

\[
X = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix} \approx UZ
\]

Each principal component is a eigen-classifier

Other step of their procedure:
Retrain classifiers, regularizing towards subspace \( U \)
PCA Summary

- **Intuition**: capture variance of data or minimize reconstruction error

- **Algorithm**: find eigendecomposition of covariance matrix or SVD

- **Impact**: reduce storage (from $O(nd)$ to $O(nk)$), reduce time complexity

- **Advantages**: simple, fast

- **Applications**: eigen-faces, eigen-documents, network anomaly detection, etc.
Generative Model [Tipping and Bishop, 1999]:

For each data point $i = 1, \ldots, n$:
- Draw the latent vector: $z_i \sim \mathcal{N}(0, I_{k \times k})$
- Create the data point: $x_i \sim \mathcal{N}(Uz_i, \sigma^2 I_{d \times d})$

PCA finds the $U$ that maximizes the likelihood of the data

$$\max_U p(X \mid U)$$
Probabilistic Interpretation

Generative Model [Tipping and Bishop, 1999]:

For each data point \( i = 1, \ldots, n \):
- Draw the latent vector: \( \mathbf{z}_i \sim \mathcal{N}(0, I_{k \times k}) \)
- Create the data point: \( \mathbf{x}_i \sim \mathcal{N}(\mathbf{Uz}_i, \sigma^2 I_{d \times d}) \)

PCA finds the \( \mathbf{U} \) that maximizes the likelihood of the data

\[
\max_{\mathbf{U}} p(\mathbf{X} \mid \mathbf{U})
\]

Advantages:
- Handles missing data (important for collaborative filtering)
- Extension to factor analysis: allow non-isotropic noise (replace \( \sigma^2 I_{d \times d} \) with arbitrary diagonal matrix)
Limitations of Linearity

PCA is effective

PCA is ineffective
Limitations of Linearity

Problem is that PCA subspace is linear:

\[ S = \{ \mathbf{x} = \mathbf{Uz} : \mathbf{z} \in \mathbb{R}^k \} \]

In this example:

\[ S = \{ (x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1 \} \]
Nonlinear PCA

We want desired solution:

\[ S = \{(x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1^2\} \]

Broken solution

Desired solution
Nonlinear PCA

We want desired solution:

\[ S = \{(x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1^2\} \]

We can get this:

\[ S = \{\phi(x) = Uz\} \text{ with } \phi(x) = (x_1^2, x_2)^\top \]

Problems:

1. Ad-hoc and tedious
2. Large, computationally expensive
Nonlinear PCA

We want desired solution: \( S = \{(x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1^2 \} \)

We can get this: \( S = \{\phi(x) = Uz\} \) with \( \phi(x) = (x_1^2, x_2)^\top \)

Linear dimensionality reduction in \( \phi(x) \) space
\[ \uparrow \]
Nonlinear dimensionality reduction in \( x \) space
Nonlinear PCA

We want desired solution: \( S = \{(x_1, x_2) : x_2 = \frac{u_2}{u_1} x_1^2\} \)

We can get this: \( S = \{\phi(x) = Uz\} \) with \( \phi(x) = (x_1^2, x_2)^\top \)

Idea: Use kernels
Kernel PCA

Representer theorem:

\[ \mathbf{XX}^\top \mathbf{u} = \lambda \mathbf{u} \quad \mathbf{u} = \mathbf{X} \alpha = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i \]
**Kernel PCA**

Representer theorem:

\[ XX^T u = \lambda u \quad u = X\alpha = \sum_{i=1}^{n} \alpha_i x_i \]

Kernel function: \( k(x_1, x_2) \) such that

\( K \), the kernel matrix formed by \( K_{ij} = k(x_i, x_j) \),

is positive semi-definite
Kernel PCA

Representer theorem:

$$\mathbf{X} \mathbf{X}^\top \mathbf{u} = \lambda \mathbf{u} \quad \mathbf{u} = \mathbf{X} \mathbf{\alpha} = \sum_{i=1}^{n} \alpha_i \mathbf{x}_i$$

Kernel function: $k(\mathbf{x}_1, \mathbf{x}_2)$ such that $K$, the kernel matrix formed by $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, is positive semi-definite

$$\max_{\|\mathbf{u}\| = 1} \mathbf{u}^\top \mathbf{X} \mathbf{X}^\top \mathbf{u} = \max_{\mathbf{\alpha}^\top \mathbf{x}^\top \mathbf{x} \mathbf{\alpha} = 1} \mathbf{\alpha}^\top (\mathbf{X}^\top \mathbf{X})(\mathbf{X}^\top \mathbf{X}) \mathbf{\alpha} = \max_{\mathbf{\alpha}^\top \mathbf{K}^2 \mathbf{\alpha} = 1} \mathbf{\alpha}^\top \mathbf{K}^2 \mathbf{\alpha}$$
Kernel PCA

Direct method:
Kernel PCA objective:
\[
\max_{\alpha^\top K \alpha = 1} \alpha^\top K^2 \alpha
\]

\[\Rightarrow \text{kernel PCA eigenvalue problem: } X^\top X \alpha = \lambda' \alpha\]

Modular method (if you don’t want to think about kernels):
Find vectors \(x'_1, \ldots, x'_n\) such that
\[x'_i^\top x'_j = K_{ij} = \phi(x_i)^\top \phi(x_j)\]

Key: use any vectors that preserve inner products
One possibility is Cholesky decomposition \(K = X^\top X'\)
Kernel PCA
Canonical Correlation Analysis (CCA)
Often, each data point consists of two views:

- **Image retrieval**: for each image, have the following:
  - $x$: Pixels (or other visual features)
  - $y$: Text around the image
Motivation for CCA [Hotelling 1936]

Often, each data point consists of two views:

- **Image retrieval**: for each image, have the following:
  - $x$: Pixels (or other visual features)
  - $y$: Text around the image
- **Time series**:
  - $x$: Signal at time $t$
  - $y$: Signal at time $t + 1$
Motivation for CCA [Hotelling 1936]

Often, each data point consists of two views:

- **Image retrieval**: for each image, have the following:
  - $x$: Pixels (or other visual features)
  - $y$: Text around the image

- **Time series**:
  - $x$: Signal at time $t$
  - $y$: Signal at time $t + 1$

- **Two-view learning**: divide features into two sets
  - $x$: Features of a word/object, etc.
  - $y$: Features of the context in which it appears
Motivation for CCA [Hotelling 1936]

Often, each data point consists of two views:

- **Image retrieval**: for each image, have the following:
  - \( x \): Pixels (or other visual features)
  - \( y \): Text around the image

- **Time series**:
  - \( x \): Signal at time \( t \)
  - \( y \): Signal at time \( t + 1 \)

- **Two-view learning**: divide features into two sets
  - \( x \): Features of a word/object, etc.
  - \( y \): Features of the context in which it appears

**Goal**: reduce the dimensionality of the two views jointly
CCA Example

Setup:

Input data: \((x_1, y_1), \ldots, (x_n, y_n)\) (matrices \(X, Y\))

Goal: find pair of projections \((u, v)\)
CCA Example

Setup:
- Input data: \((x_1, y_1), \ldots, (x_n, y_n)\) (matrices \(X, Y\))
- Goal: find pair of projections \((u, v)\)

Dimensionality reduction solutions:
- Independent
- Joint

\(x\) and \(y\) are paired by brightness
CCA Definition

Definitions:

Variance: $\hat{\text{var}}(u^\top x) = u^\top XX^\top u$

Covariance: $\hat{\text{cov}}(u^\top x, v^\top y) = u^\top XY^\top v$

Correlation: $\frac{\hat{\text{cov}}(u^\top x, v^\top y)}{\sqrt{\hat{\text{var}}(u^\top x)} \sqrt{\hat{\text{var}}(v^\top y)}}$

Objective: maximize correlation between projected views

$$\max_{u,v} \hat{\text{corr}}(u^\top x, v^\top y)$$

Properties:

- Focus on how variables are related, not how much they vary
- Invariant to any rotation and scaling of data
From PCA to CCA

PCA on views separately: no covariance term

\[
\max_{u,v} \frac{u^\top XX^\top u}{u^\top u} + \frac{v^\top YY^\top v}{v^\top v}
\]

PCA on concatenation \((X^\top, Y^\top)^\top\): includes covariance term

\[
\max_{u,v} \frac{u^\top XX^\top u + 2u^\top XY^\top v + v^\top YY^\top v}{u^\top u + v^\top v}
\]
From PCA to CCA

PCA on views separately: no covariance term

$$\max_{u,v} \frac{u^T XX^T u}{u^T u} + \frac{v^T YY^T v}{v^T v}$$

PCA on concatenation \((X^T, Y^T)^T\): includes covariance term

$$\max_{u,v} \frac{u^T XX^T u + 2u^T XY^T v + v^T YY^T v}{u^T u + v^T v}$$

Maximum covariance: drop variance terms

$$\max_{u,v} \frac{u^T XY^T v}{\sqrt{u^T u} \sqrt{v^T v}}$$
From PCA to CCA

PCA on views separately: no covariance term

\[
\max_{u,v} \frac{u^\top XX^\top u}{u^\top u} + \frac{v^\top YY^\top v}{v^\top v}
\]

PCA on concatenation \((X^\top, Y^\top)^\top\): includes covariance term

\[
\max_{u,v} \frac{u^\top XX^\top u + 2u^\top XY^\top v + v^\top YY^\top v}{u^\top u + v^\top v}
\]

Maximum covariance: drop variance terms

\[
\max_{u,v} \frac{u^\top XY^\top v}{\sqrt{u^\top u} \sqrt{v^\top v}}
\]

Maximum correlation (CCA): divide out variance terms

\[
\max_{u,v} \frac{u^\top XY^\top v}{\sqrt{u^\top XX^\top u} \sqrt{v^\top YY^\top v}}
\]
Importance of Regularization

Extreme examples of degeneracy:

• If $x = Ay$, then any $(u, v)$ with $u = Av$ is optimal (correlation 1)
• If $x$ and $y$ are independent, then any $(u, v)$ is optimal (correlation 0)

Solution: regularization (interpolate between maximum covariance and maximum correlation)
Importance of Regularization

Extreme examples of degeneracy:

• If \( x = Ay \), then any \((u, v)\) with \( u = Av \) is optimal (correlation 1)

• If \( x \) and \( y \) are independent, then any \((u, v)\) is optimal (correlation 0)

Problem: if \( X \) or \( Y \) has rank \( n \), then any \((u, v)\) is optimal (correlation 1) with \( u = X^\top Y v \) \( \Rightarrow \) CCA is meaningless!
Importance of Regularization

Extreme examples of degeneracy:

- If $\mathbf{x} = \mathbf{A}\mathbf{y}$, then any $(\mathbf{u}, \mathbf{v})$ with $\mathbf{u} = \mathbf{A}\mathbf{v}$ is optimal (correlation 1)
- If $\mathbf{x}$ and $\mathbf{y}$ are independent, then any $(\mathbf{u}, \mathbf{v})$ is optimal (correlation 0)

Problem: if $\mathbf{X}$ or $\mathbf{Y}$ has rank $n$, then any $(\mathbf{u}, \mathbf{v})$ is optimal (correlation 1) with $\mathbf{u} = \mathbf{X}^\top\mathbf{Y}\mathbf{v} \Rightarrow$ CCA is meaningless!

Solution: regularization (interpolate between maximum covariance and maximum correlation)

$$\max_{\mathbf{u}, \mathbf{v}} \frac{\mathbf{u}^\top \mathbf{X}\mathbf{Y}^\top \mathbf{v}}{\sqrt{\mathbf{u}^\top (\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}) \mathbf{u}} \sqrt{\mathbf{v}^\top (\mathbf{Y}\mathbf{Y}^\top + \lambda \mathbf{I}) \mathbf{v}}}$$
Canonical Correlation Forests

We introduce canonical correlation forests (CCFs), a new decision tree ensemble method for classification. Individual canonical correlation trees are binary decision trees with hyperplane splits based on canonical correlation components. Unlike axis-aligned alternatives, the decision surfaces of CCFs are not restricted to the coordinate system of the input features and therefore more naturally represent data with correlation between the features. Additionally we introduce a novel alternative to bagging, the projection bootstrap, which maintains use of the full dataset in selecting split points. CCFs do not require parameter tuning and our experiments show that they out-perform axis-aligned random forests, other state-of-the-art tree ensemble methods and all of the 179 popular classifiers considered in a recent extensive survey.

Example: RF that uses CCA to determine axis for splits
Summary

Framework: \( z = U^\top x, \ x \approx Uz \)

Criteria for choosing \( U \):

- PCA: maximize projected variance
- CCA: maximize projected correlation

Algorithm: generalized eigenvalue problem

Extensions:
- non-linear using kernels (using same linear framework)
- probabilistic, sparse, robust (hard optimization)