Learning from Data
Lecture 10: Principal Component Analysis

Yang Li **yangli@sz.tsinghua.edu.cn**

TBSI

December 3, 2022

Today's Lecture

Unsupervised Learning (Part II): PCA

- Motivation
- Linear PCA
- Kernel PCA

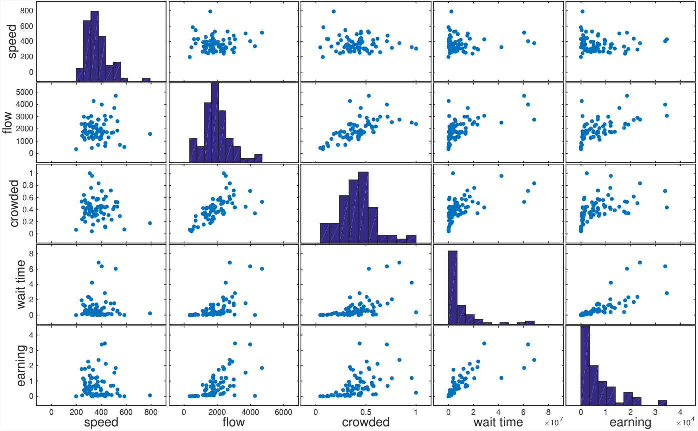
Motivation of PCA

Example: Analyzing San Francisco public transit route efficiency

features	notes
speed	average speed
flow	# boarding passengers per hour
crowded	% passenger capacity reached
wait time	average waiting time at bus stop
earning	net operation revenue
⋮	⋮

Motivation of PCA

Input features contain a lot of redundancy

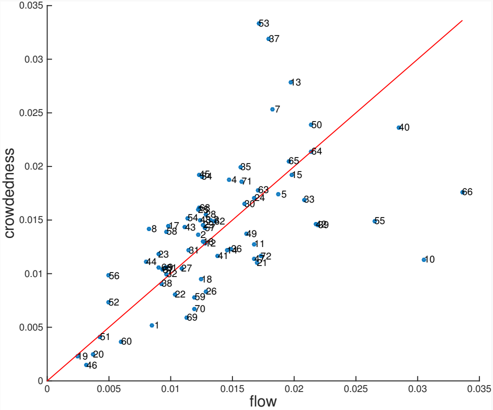


Scatter plot matrix reveals pairwise correlations among 5 major features

Motivation of PCA

Example of linearly dependent features

- Flow: average # boarding passengers per hour
- Crowdedness: $\frac{\text{average \# passengers on train}}{\text{train capacity}}$



How can we automatically detect and remove this redundancy?

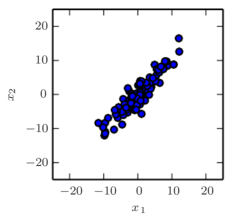
- geometric approach ← *start here!*
- diagonalize covariance matrix approach

How to remove feature redundancy?

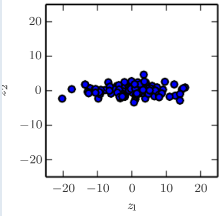
Given $\{x^{(1)}, \dots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^n$.

- Find a linear, orthogonal transformation $W : \mathbb{R}^n \to \mathbb{R}^k$ of the input data
- W aligns the direction of maximum variance with the axes of the new space.

Example: $n = 2$



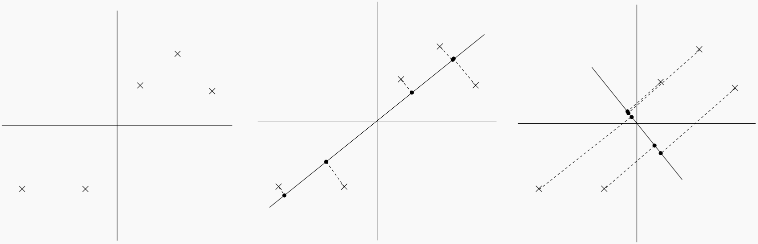
features x_1 and x_2 are strongly correlated



variations in $z = x^T W$ is mostly along the x-axis. *x can be represented in 1D!*

Direction of Maximum Variance

- Suppose $\mu = mean(x) = 0$, $\sigma_j = var(x_j) = 1$ (variance of jth feature)
- Find **major axis of variation** unit vector u:



input observations

projections on u have large variance

projections on u have small variance

u maximizes the variance of the projections

Principal Component Analysis (PCA)

Pearson, K. (1901), Hotelling, H. (1933) "Analysis of a complex of statistical variables into principal components". Journal of Educational Psychology.

PCA goals

- Find principal components $u_1, \ldots, u_n$ that are mutually orthogonal (uncorrelated)
- Most of the variation in x will be accounted for by k principal components where $k \ll n$.

Main steps of (full) PCA:

1. Standardize x such that $Mean(x) = 0, Var(x_j) = 1$ for all j
2. Find projection of x, $u_1^T x$ with maximum variance
3. For $j = 2, \ldots, n$,
   Find another projection of x, $u_j^T x$ with maximum variance,
   where u_j is orthogonal to $u_1, \ldots, u_{j-1}$

Step 1: Standardize data

Normalize x such that $Mean(x) = 0$ and $Var(x_j) = 1$

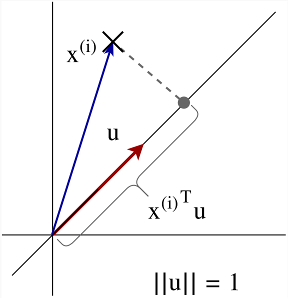
$$x^{(i)} := x^{(i)} - \mu \leftarrow \text{recenter}$$
$$x_j^{(i)} := x_j^{(i)} / \sigma_j \leftarrow \text{scale by } stdev(x_j)$$

Check:

$$var\left(\frac{x_j}{\sigma_j}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{x_j^{(i)} - \mu_j}{\sigma_j}\right)^2 = \frac{1}{\sigma_j^2}\frac{1}{m}\sum_{i=1}^{m}\left(x_j^{(i)} - \mu_j\right)^2$$
$$= \frac{1}{\sigma_j^2}\sigma_j^2 = 1$$

Step 2: Find Projection with Maximum Variance

Since $\|u\| = 1$, the length of $x^{(i)}$'s projection on u is $x^{(i)^T} u$.



Variance of the projections:

$$\frac{1}{m}\sum_{i=1}^{m}(x^{(i)^T} u - 0)^2 = \frac{1}{m}\sum_{i=1}^{m} u^T x^{(i)} x^{(i)^T} u$$
$$= u^T \left(\frac{1}{m}\sum_{i=1}^{m} x^{(i)} x^{(i)^T}\right) u$$
$$= u^T \Sigma u$$

Σ : the sample covariance matrix of $x^{(1)} \ldots x^{(m)}$.

1st Principal Component

Find unit vector u_1 that maximizes variance of projections:

$$u_1 = \underset{u:\|u\|=1}{\operatorname{argmax}} \; u^T \Sigma u \tag{1}$$

u_1 is the **1st principal component** of X

u_1 can be solved using optimization tools, but it has a more efficient solution:

Proposition 1

u_1 is the largest eigenvector of covariance matrix Σ

Proposition 1

u_1 is the largest eigenvector of covariance matrix Σ

Proof. Generalized Lagrange function of Problem 1:

$$L(u) = -u^T \Sigma u + \beta(u^T u - 1)$$

To minimize $L(u)$,

$$\frac{\delta L}{\delta u} = -2\Sigma u + 2\beta u = 0 \implies \Sigma u = \beta u$$

Therefore u_1 must be an eigenvector of Σ.
Let $u_1 = v_j$, the eigenvector with the jth largest eigenvalue λ_j,

$$u_1^T \Sigma u_1 = v_j^T \Sigma v_j = \lambda_j v_j^T v_j = \lambda_j.$$

Hence $u_1 = v_1$, the eigenvector with the largest eigenvalue λ_1. $\square$

Proposition 2

The jth **principal component** *of X, u_j is the jth largest eigenvector of Σ.*

Proof. Consider the case $j = 2$,

$$u_2 = \underset{u:\|u\|=1, u_1^T u=0}{\operatorname{argmax}} u^T \Sigma u \tag{2}$$

The Lagrangian function:

$$L(u) = -u^T \Sigma u + \beta_1(u^T u - 1) + \beta_2(u_1^T u)$$

Minimizing $L(u)$ yields:

$$\beta_2 = 0, \Sigma u = \beta_1 u$$

To maximize $u^T \Sigma u = \lambda$, u_2 must be the eigenvector with the second largest eigenvalue $\beta_1 = \lambda_2$. The same argument can be generalized to cases $j > 2$. *(Use induction to prove for $j = 1 \dots n$)* □

Summary

We can solve PCA by solving an eigenvalue problem!
Main steps of (full) PCA:

1. Standardize x such that $Mean(x) = 0, Var(x_j) = 1$ for all j
2. Compute $\Sigma = cov(x)$
3. Find principal components $u_1, \ldots, u_n$ by eigenvalue decomposition: $\Sigma = U \Lambda U^T$. $\leftarrow$ *U is an orthogonal basis in $\mathbb{R}^n$*

Next we project data vectors x to this new basis, which spans the **principal component space**.

PCA Projection

- Projection of sample $x \in \mathbb{R}^n$ in the principal component space:

$$z^{(i)} = \begin{bmatrix} {x^{(i)}}^T u_1 \\ \vdots \\ {x^{(i)}}^T u_n \end{bmatrix} \in \mathbb{R}^n$$

- Matrix notation:

$$z^{(i)} = \begin{bmatrix} | & & | \\ u_1 & \dots & u_n \\ | & & | \end{bmatrix}^T x^{(i)} = U^T x^{(i)}, \text{ or } Z = XU$$

- The truncated transformation $Z_k = XU_k$ keeping only the first k principal components is used for **dimension reduction**.

Properties of PCA

- The variance of principal component projections are

$$\mathrm{Var}(x^T u_j) = {u_j}^T \Sigma u_j = \lambda_j \text{ for } j = 1, \ldots, n$$

- % of variance explained by the jth principal component: $\frac{\lambda_j}{\sum_{i=1}^{n} \lambda_i}$.
  i.e. projections are uncorrelated
- % of variance accounted for by retaining the first k principal components ($k \leq n$): $\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{n} \lambda_j}$

Another geometric interpretation of PCA is minimizing projection residuals. (see homework!)

Covariance Interpretation of PCA

PCA removes the "redundancy" (or noise) in input data X:

Let $Z = XU$ be the PCA projected data,

$$\operatorname{cov}(Z) = \frac{1}{m} Z^T Z = \frac{1}{m}(XU)^T(XU) = U^T \left(\frac{1}{m} X^T X\right) U = U^T \Sigma U$$

Since Σ is symmetric, it has real eigenvalues. Its eigen decomposition is

$$\Sigma = U \Lambda U^T$$

where

$$U = \begin{bmatrix} | & & | \\ u_1 & \dots & u_n \\ | & & | \end{bmatrix}, \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$$

Then

$$\operatorname{cov}(Z) = U^T (U \Lambda U^T) U = \Lambda$$

The principal component transformation XU diagonalizes the sample covariance matrix of X

Linear PCA Review

PCA Dimension reduction

- Find principal components $u_1, \ldots, u_n$ that are mutually orthogonal (uncorrelated)
- Most of the variations in x will be accounted for by k principal components where $k \ll n$.

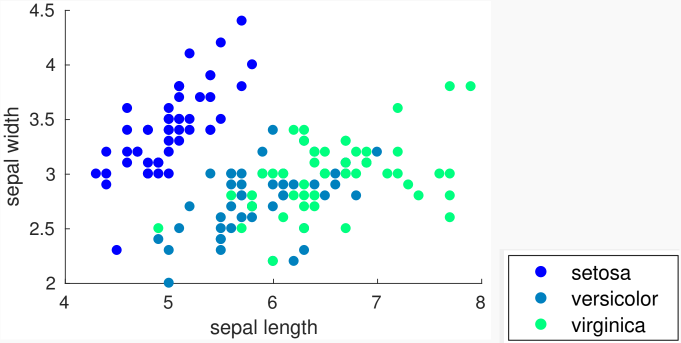
Main steps

1. Standardize x such that $Mean(x) = 0, Var(x_j) = 1$ for all j
2. Compute $\Sigma = cov(x)$
3. Find principal components $u_1, \ldots, u_n$ by eigenvalue decomposition: $\Sigma = U \Lambda U^T$. $\leftarrow$ *U is an orthogonal basis in $\mathbb{R}^n$*
4. Project data on first the k principal components: $z = [x^T u_1, \ldots, x^T u_k]^T$

PCA Example: Iris Dataset

- 150 samples
- input feature dimension: 4

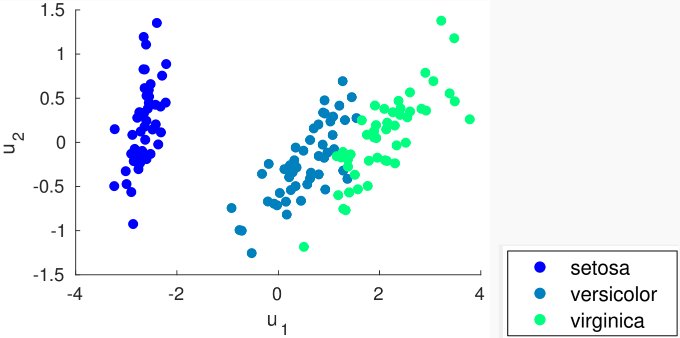
First two input attributes



PCA Example: Iris Dataset

- 150 samples
- input feature dimension: 4

PCA Projection on 2 Principal Components



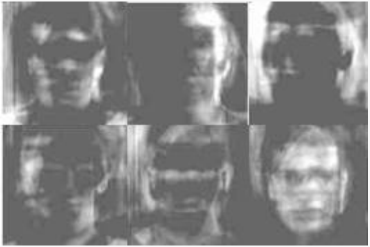
% of variance explained by PC1: 73%,   by PC2: 22%

PCA Example: Eigenfaces

Learning image representations for face recognition using PCA [Turk and Pentland CVPR 1991]

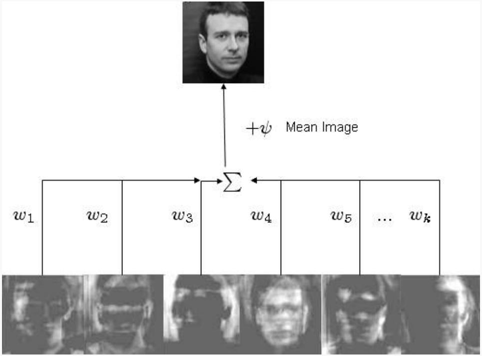
Training data

Eigenfaces: k principal components



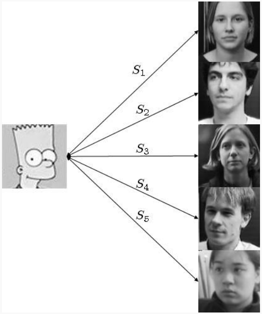
PCA Example: Eigenfaces

Each face image is a linear combination of the **eigenfaces** (principal components)



Each image is represented by k weights

Recognize faces by classifying the weight vectors. e.g. k-Nearest Neighbor



Kernel PCA

Feature extraction using PCA

$$x^{(i)} \xrightarrow{\text{PCA}} Wx^{(i)} \longrightarrow \underbrace{\boxed{f}}_{\text{e.g. k-means}} \longrightarrow c^{(i)}$$

Linear PCA assumes data are separable in $\mathbb{R}^n$

A non-linear generalization

- Project data into higher dimension using feature mapping $\phi : \mathbb{R}^n \to \mathbb{R}^d$ $(d \geq n)$
- Feature mapping is defined by a kernel function $K\left(x^{(i)}, x^{(j)}\right) = \phi(x^{(i)})^T \phi(x^{(j)})$ or kernel matrix $K \in \mathbb{R}^{m \times m}$
- We can now perform standard PCA in the feature space

Kernel PCA

(Bernhard Schoelkopf, Alexander J. Smola, and Klaus-Robert Mueller. 1999. *Kernel principal component analysis*. In Advances in kernel methods) Sample covariance matrix of feature mapped data (assuming $\phi(x)$ is centered)

$$\Sigma = \frac{1}{m}\sum_{i=1}^{m}\phi(x^{(i)})\phi(x^{(i)})^T \in \mathbb{R}^{d\times d}$$

Let $(\lambda_k, u_k), k = 1, \ldots, d$ be the eigen decomposition of Σ:

$$\Sigma u_k = \lambda_k u_k$$

PCA projection of $x^{(l)}$ onto the *kth* principal component u_k:

$$\phi(x^{(l)})^T u_k$$

How to avoid evaluating $\phi(x)$ explicitly?

The Kernel Trick

Represent projection $\phi(x^{(l)})^T u_k$ using kernel function K:

- Write u_k as a linear combination of $\phi(x^{(1)}), \ldots, \phi(x^{(m)})$:

$$u_k = \sum_{i=1}^{m} \alpha_k^i \phi(x^{(i)})$$

- PCA projection of $x^{(l)}$ using kernel function K:

$$\phi(x^{(l)})^T u_k = \phi(x^{(l)})^T \sum_{i=1}^{m} \alpha_k^i \phi(x^{(i)}) = \sum_{i=1}^{m} \alpha_k^i K(x^{(l)}, x^{(i)})$$

How to find α_k^i's directly ?

The Kernel Trick

Kth eigenvector equation:

$$\Sigma u_k = \left(\frac{1}{m}\sum_{i=1}^{m}\phi(x^{(i)})\phi(x^{(i)})^T\right)u_k = \lambda_k u_k$$

- Substitute $u_k = \sum_{i=1}^{m}\alpha_k^{(i)}\phi(x^{(i)})$, we obtain

$$K\alpha_k = \lambda_k m\alpha_k$$

where $\alpha_k = \begin{bmatrix}\alpha_k^1 \\ \vdots \\ \alpha_k^m\end{bmatrix}$ can be solved by eigen decomposition of K

- Normalize α_k such that $u_k^T u_k = 1$:

$$u_k^T u_k = \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_k^i\alpha_k^j\phi(x^{(i)})^T\phi(x^{(j)}) = \alpha_k^T K\alpha_k = \lambda_k m(\alpha_k^T\alpha_k)$$

$$\|\alpha_k\|^2 = \frac{1}{\lambda_k m}$$

Kernel PCA

When $\mathbb{E}[\phi(x)] \neq 0$, we need to center $\phi(x)$:

$$\widetilde{\phi}(x^{(i)}) = \phi(x^{(i)}) - \frac{1}{m}\sum_{l=1}^{m}\widetilde{\phi}(x^{(l)})$$

The "centralized" kernel matrix is

$$\tilde{K}_{i,j} = \widetilde{\phi}(x^{(i)})^T\widetilde{\phi}(x^{(j)})$$

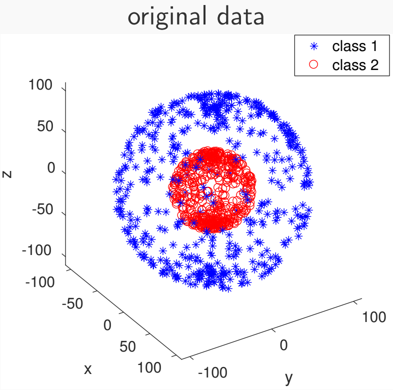
In matrix notation:

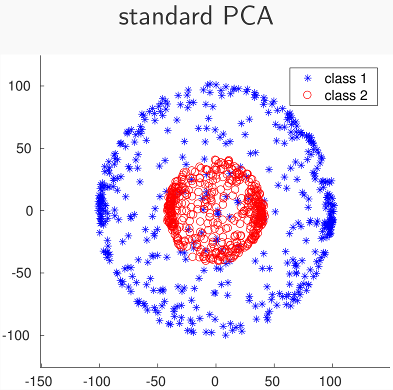
$$\widetilde{K} = K - 1_m K - K 1_m + 1_m K 1_m$$

where $1_m = \begin{bmatrix} 1/m & \dots & 1/m \\ \vdots & \ddots & \vdots \\ 1/m & \dots & 1/m \end{bmatrix} \in \mathbb{R}^{mxm}$

Use $\widetilde{K}$ to compute PCA

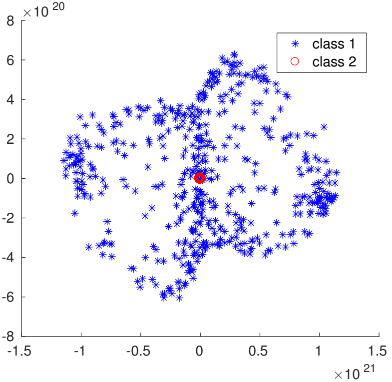
Kernel PCA Example





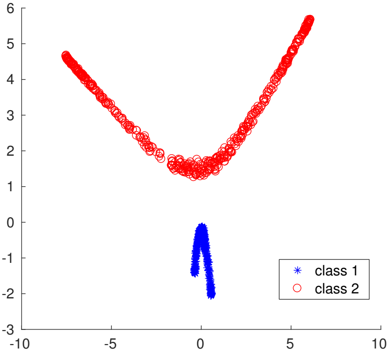
Kernel PCA Example

Polynomial kernel PCA



$$k(x,x') = (x \cdot x' + 1)^5$$

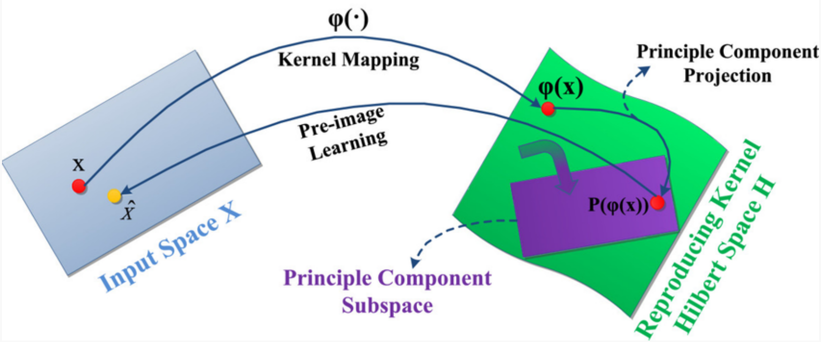
Gaussian kernel PCA



$$k(x,x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$$

Discussions of kernel PCA

- Often used in clustering, abnormality detection, etc
- Requires finding eigenvectors of $m \times m$ matrix instead of $n \times n$
- Dimension reduction by projecting to k-dimensional principal subspace is generally not possible



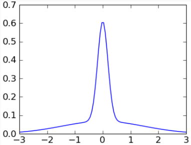
The Pre-Image problem: reconstruct data in input space x from feature space vectors $\phi(x)$

PCA Limitations

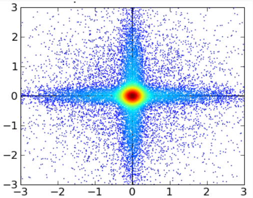
- Assumes input data is real and continuous
- Assumes **approximate normality** of input space (but may still work well on non-normally distributed data in practice) *← sample mean & covariance must be sufficient statistics*

Example of strongly non-normal distributed input:

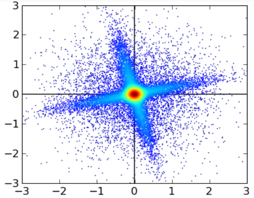
PDF



Original Input



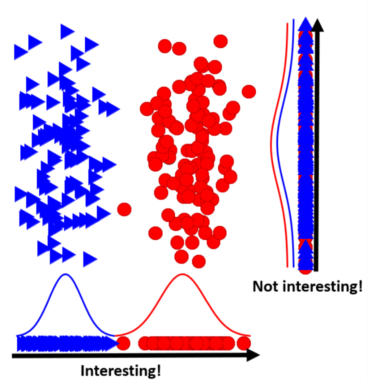
PCA Projection

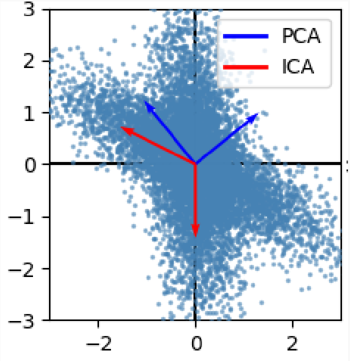


PCA Limitations

PCA results may not be useful when

- Axes of larger variance is less 'interesting' than smaller ones.
- Axes of variations are not orthogonal;





Summary

Representation learning

- Transform input features into “simpler” or “interpretable” representations.
- Used in feature extraction, dimension reduction, clustering etc

Unsupervised learning algorithms:

	low dimension	sparse	disentangle variations
k-means	✓	✓	
spectral embedding	✓		✓
PCA	✓		✓