Learning From Data
Lecture 7: K-Means Clustering & PCA

Yang Li yangli@sz.tsinghua.edu.cn

TBSI

October 29, 2020

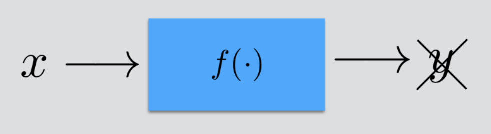
Today's Lecture

Unsupervised Learning

- Overview: the representation learning problem
- K-means clustering
- Principal component analysis

Written Assignment 2 is due today.

Unsupervised Learning



Similar to supervised learning, but without labels.

- Still want to learn the machine f
- Significantly harder in general

Unsupervised learning goal

Find **representations** of input feature x that can be used for reasoning, decision making, predicting things, comminicating etc.

The representation learning problem

(Y Bengio et. al. *Representation Learning: A Review and New Perspectives*, 2014)

Given input features x, find "simpler" features z that **preserve the same information** as x.

Example: Face recognition

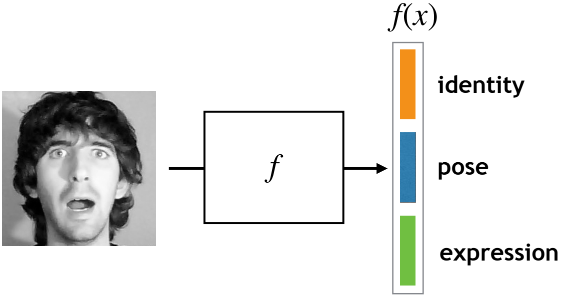
100×100

$$\rightarrow x = \left.\begin{bmatrix} 0.5 \\ 0 \\ \vdots \\ 0.3 \\ 1.0 \end{bmatrix}\right\} 10^4 \quad \rightarrow \qquad z = \begin{bmatrix} \vdots \end{bmatrix}$$

What information is in this picture? *identity, facial attributes, gender, age, sentiment, etc*

Characteristics of a good representation

- low dimensional: compress information to a smaller size $\rightarrow$ *reduce data size*
- sparse representation: most entries are zero for most data $\rightarrow$ *better interpretability*
- independent representations: disentangle the source of variations

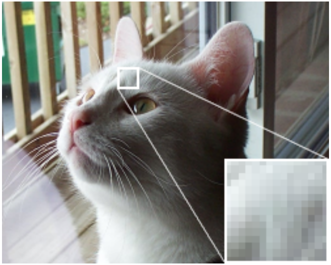


Uses of representation learning

- Data compression

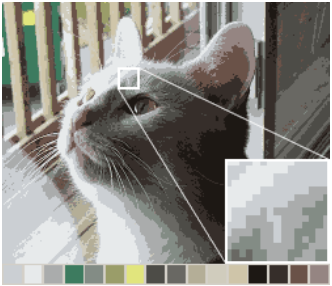
Example: Color image quantization. Each 24bit RGB color is reduced to a palette of 16 colors.

Original



(0-255,0-255,0-255)
24bit x 300 x 400

Compressed

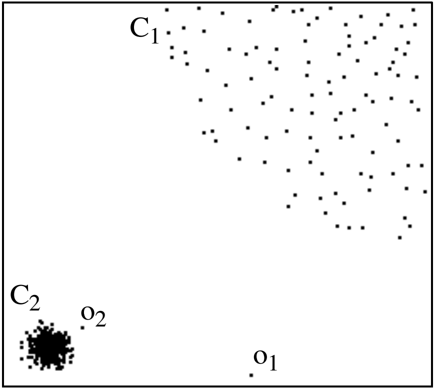


0-15
4bit x 300 x 400 + 16 x24bit
6 times smaller

Uses of representation learning

- Abnormality (outlier, novelty) detection

Example: local density-based outlier detection

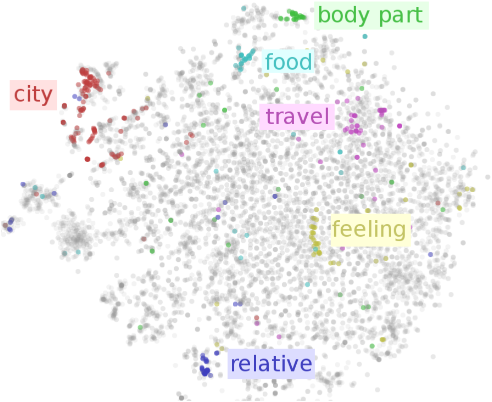


o_1 and o_2 are the detected outliers

Uses of representation learning

- Knowledge representation based on human perception

Example: word embedding

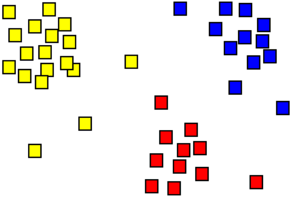


http://ruder.io/word-embeddings-1/

Each word is represented by a 2D vector. Words in the same semantic category are grouped together

Clustering analysis

Given input features $\{x^{(1)}, \ldots, x^{(m)}\}$, group the data into a few *cohesive* "clusters".



- Objects in the same cluster are more similar to each other than to those in other clusters

The k-means clustering problem

Given input data $\{x^{(1)}, \ldots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^d$, **k-means clustering** partition the input into $k \leq m$ sets $C_1, \ldots, C_k$ to minimize the within-cluster sum of squares (WCSS).

$$\underset{C}{\operatorname{argmin}} \sum_{j=1}^{k} \sum_{x \in C_j} \|x - \mu_j\|^2$$

Equivalent definitions:

- minimizing the within-cluster variance: $\sum_{j=1}^{k} |C_j| \operatorname{Var}(C_j)$
- minimizing the pairwise squared deviation between points in the same cluster: *(homework)*

$$\sum_{i=1}^{k} \frac{1}{2|C_i|} \sum_{x, x' \in C_i} \|x - x'\|^2$$

- maximizing between-cluster sum of squares (BCSS) *(homework)*

K-Means Clustering Algorithm

- Optimal k-means clustering is NP-hard in Euclidean space.
- Often solved via a heuristic, iterative algorithm

Lloyd's Algorithm (1957,1982)

Let $c^{(i)} \in \{1, \ldots, k\}$ be the cluster label for $x^{(i)}$

```
Initialize cluster centroids μ_1,...μ_k ∈ R^n randomly
Repeat until convergence{
  For every i,
     c^(i) := argmin_j ||x^(i) − μ_j||^2      ← assign x^(i) to the cluster
                                                 with the closest centroid
  For each j
     μ_j := Σ_{i=1}^m 1{c^(i)=j} x^(i) / Σ_{i=1}^m 1{c^(i)=j}    ← update centroid
}
```

$$c^{(i)} := \operatorname{argmin}_j \left\|x^{(i)} - \mu_j\right\|^2$$

$$\mu_j := \frac{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)}=j\}x^{(i)}}{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)}=j\}}$$

Demo:http://stanford.edu/class/ee103/visualizations/kmeans/kmeans.html

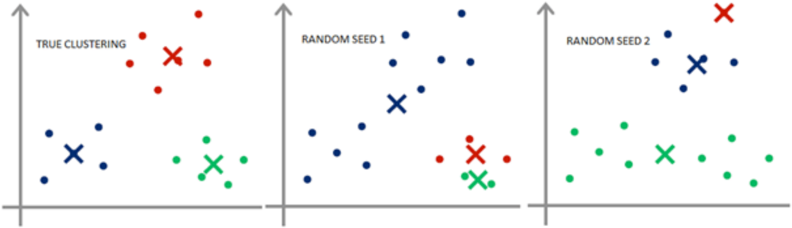
Lloyd, Stuart P. (1982). "Least squares quantization in PCM". IEEE Transactions on Information Theory

K-Means clustering discussion

- K-Means learns a k-dimensional *sparse* representation.
  i.e. $x^{(i)}$ is transformed into a "one-hot" vector $z^{(i)} \in \mathbb{R}^k$:

$$z_j^{(i)} = \begin{cases} 1 & \text{if } c^{(i)} = j \\ 0 & \text{otherwise} \end{cases}$$

- Only converges to a local minimum: initialization matters!



Practical considerations

- Replicate clustering trails and choose the result with the smallest WCSS
- How to initialize centroids μ_j's ?
  - Uniformly random sampling ☹
  - Distance-based sampling e.g. kmeans++ [Arthur & Vassilvitskii SODA 2007] ☺
- How to choose k?
  - Cross validation (later lecture)
  - G-Means [Hamerly & Elkan, NIPS 2004]
- How to improve k-means efficiency?
  - Elkan's algorithm [Elkan, ICML 2003]
  - Mini-batch k-means [D. Sculley, WWW 2010]

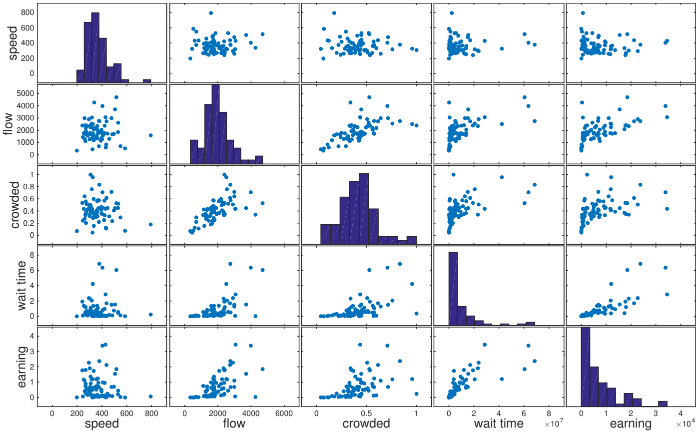
Motivation of PCA

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

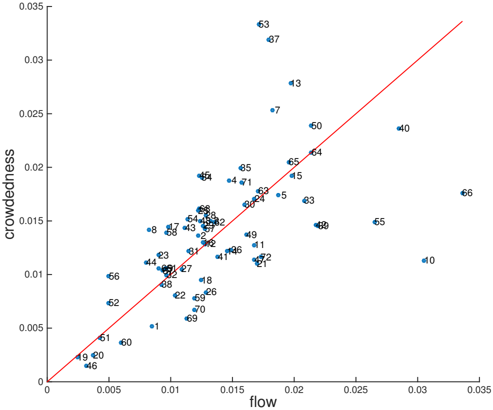


Scatter plot matrix reveals pairwise correlations among 5 major features

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?

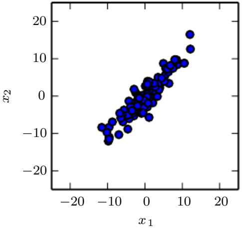
- geometric approach
  ← *start here!*
- diagonalize covariance matrix approach

How to removing feature redundancy?

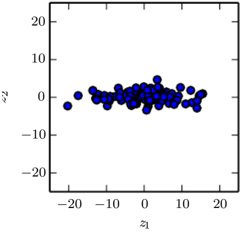
Given $\{x^{(1)}, \ldots, x^{(m)}\}$, $x^{(i)} \in \mathbb{R}^n$.

- 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: $n = 2$



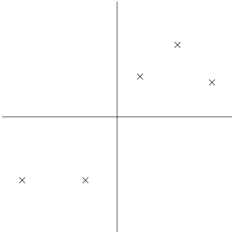
features x_1 and x_2 are strongly correlated



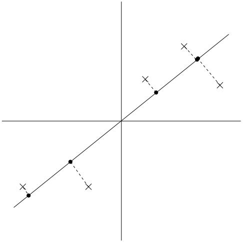
variations in $z = x^T W$ is mostly along the x-axis. *x can be represented in 1D!*

Direction of Maximum Variance

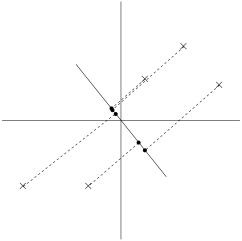
- Suppose $\mu = mean(x) = 0$, $\sigma_j = var(x_j) = 1$ (variance of jth feature)
- Find **major axis of variation** unit vector u:



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 $Var(x_j) = 1$

$$x^{(i)} := x^{(i)} - \mu \quad \leftarrow \text{recenter}$$

$$x_j^{(i)} := x_j^{(i)}/\sigma_j \quad \leftarrow \ \text{scale by } stdev(x_j)$$

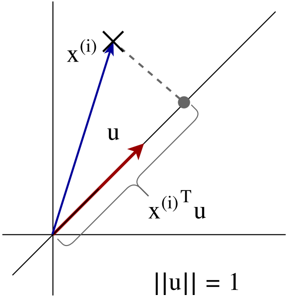
Check:

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

$$= \frac{1}{\sigma_j^2}\sigma_j^2 = 1$$

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:

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

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

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

A Review on Eigenvalue Problem

The Eigenvalue Problem

Nonzero vector $u \in \mathbb{R}^n$ is an **eigenvector** of matrix $A \in \mathbb{R}^{n \times n}$ if

$$Au = \lambda u$$

for some $\lambda \in \mathbb{R}$. We call λ the **eigenvalue** corresponding to u.

- A has at most n distinct eigenvalues

Eigenvalue Decomposition

Let $U = [u_1, \ldots, u_n]$ be the matrix of n linearly independent eigenvectors of A and $\Lambda = diag([\lambda_1, \ldots, \lambda_n])$, then

$$A = U \Lambda U^{-1}$$

- If A is symmetric, A can be decomposed as $A = U \Lambda U^T$ where U is an orthogonal matrix ($U^T U = I$).

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 j**th principal component** *of* X , u_j *is the* j*th 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 $j = 1 \ldots n$)*

$\square$

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,

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

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

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

where

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

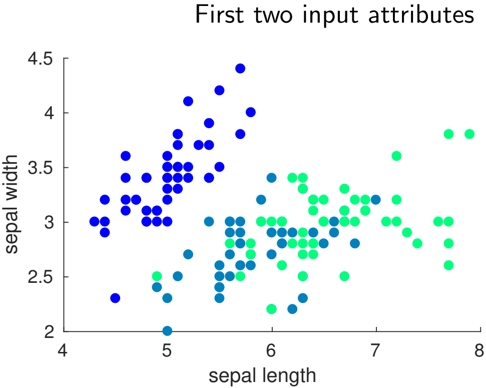
Then

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

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

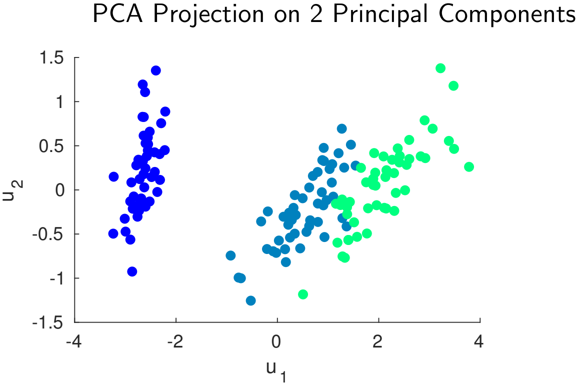
PCA Example: Iris Dataset

- 150 samples
- input feature dimension: 4



PCA Example: Iris Dataset

- 150 samples
- input feature dimension: 4



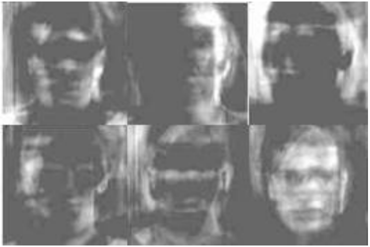
% of variance explained by PC1: 73%,   by PC2: 22%

PCA Example: Eigenfaces

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

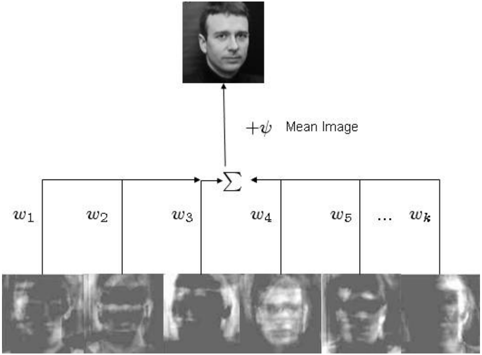
Training data

Eigenfaces: k principal components



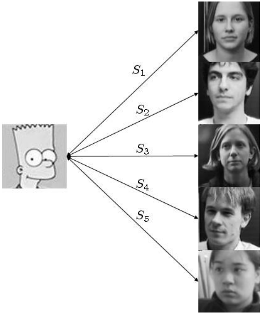
PCA Example: Eigenfaces

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

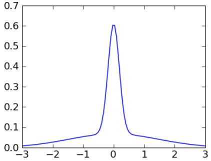


PCA Limitations

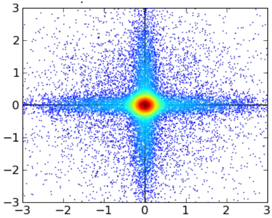
- Only considers linear relationships in data (see kernel PCA)
- Assumes input data is real and continuous
- Assumes **approximate normality** of input space (but may still work well on non-normally distributed data in practice)

Example of strongly non-normal distributed input:

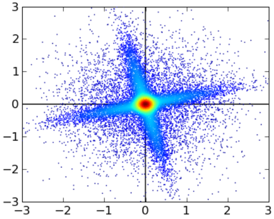
PDF



Original Input



PCA Projection



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

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

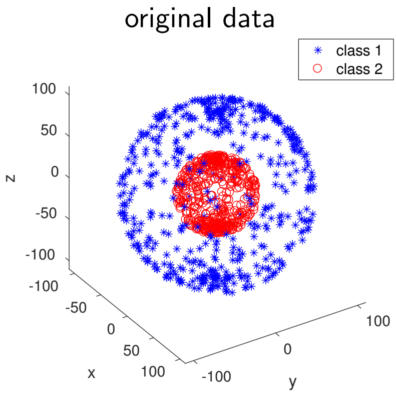
In matrix notation:

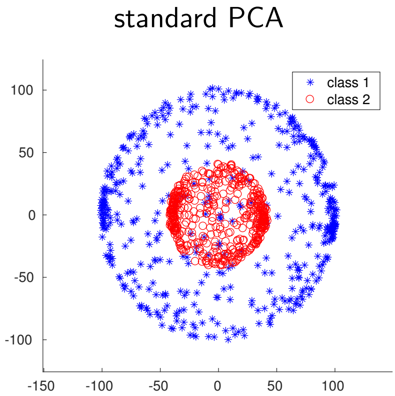
$$\widetilde{K} = K - \mathbf{1}_m K - K\mathbf{1}_m + \mathbf{1}_m K \mathbf{1}_m$$

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

Use $\widetilde{K}$ to compute PCA

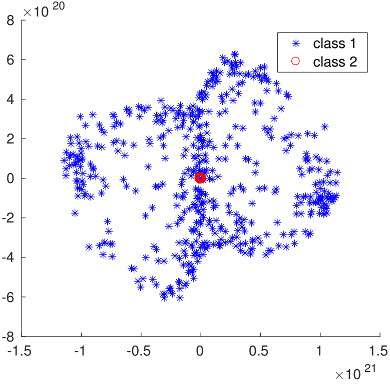
Kernel PCA Example





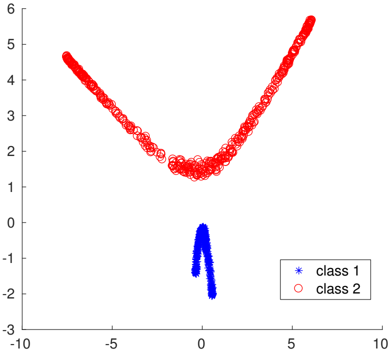
Kernel PCA Example

Polynomial kernel PCA



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

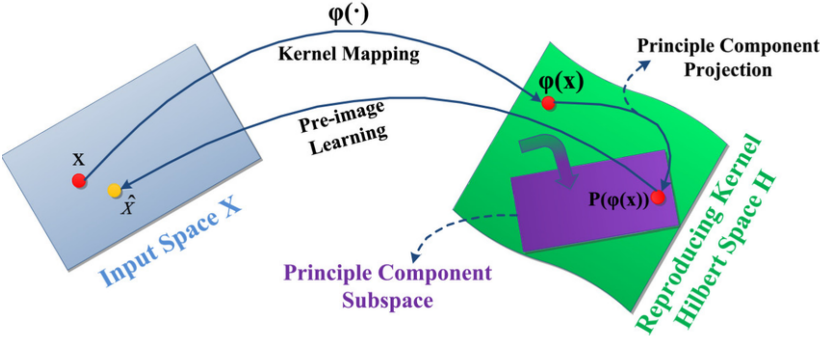
Gaussian kernel PCA



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

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 $\phi(x)$

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		✓	
PCA	✓		✓