Kernel PCA

- idea: get a non-linear coordinate transform by first mapping the data non-linearly into a augmented feature space $\tilde{x}_i = \varphi(x_i)$ ← non-linear

  then get new features z_i by PCA on $\tilde{x}$ space

- kernel trick: compute the new features without explicit construction of $\tilde{x}_i$

⇒ need to reformulate PCA such that it only accesses data via scalar products $x_i x_{i'}^T$, i.e. via Gram matrix $G = X X^T$ (X is centered)

- we did PCA as eigen decomposition of $S = X^T X$

$$X^T X v^T = \lambda v^T \qquad \underbrace{X \cdot X^T}_{=G} \cdot \underbrace{X v^T}_{=u} = X \cdot (\lambda v^T) = \lambda \underbrace{X v^T}_{u}$$

$G u = \lambda u$ ⇒ eigenvalues of S are also eigenvalues of G

eigenvectors are related as $u = X \cdot v^T = z$

⇒ we can compute the new features by eigen decomposition of G instead of S

But: z are not unit vectors, but have variance $\mathrm{var}_i(z_{ij}) = \lambda_j$

define $z = u \cdot \Lambda^{\frac{1}{2}}$ ← unit vector (eigenvector of G)

$$\Lambda^{1/2} = \begin{pmatrix} \sqrt{\lambda_1} & & 0 \\ & \ddots & \\ 0 & & \sqrt{\lambda_D} \end{pmatrix}$$

solution in terms of scalar products:

$G = X X^T \qquad G u = \lambda u$

$z = u \cdot \Lambda^{1/2}$ for the first D' eigenvectors

- kernel trick: replace G with a kernel matrix $K_{ii'} = \underbrace{k(x_i, x_{i'})}_{\text{kernel function}}$

  and compute eigendecomposition of K

- <u>catch</u>: PCA requires the data to be centered, but the K-matrix is not centered: $K_{ii'} = \varphi(x_i) \cdot \varphi(x_{i'})^T$ for some $\varphi(x)$

  $\varphi(x)$ a is implicitly defined by $k(x_i, x_{i'})$ and generally not centered

- Q: How can we center K without computing $\bar{\varphi}(x) = \frac{1}{N} \sum_i \varphi(x_i)$

A: we can compute a centered Gram matrix (kernel matrix) from <u>uncentered data</u>!

Let X' be uncentered data, ~~$X \to X' - \bar{X}'$~~ $X = X' - \mathbb{1} \cdot \bar{X}'$ ← column vector of all 1's

G' uncentered Gram matrix $G' = X' X'^T$

G centered Gram matrix $G = X X^T = (X' - \mathbb{1}\bar{X}')(X' - \mathbb{1}\bar{X}')^T$

$$G = \underbrace{X' X'^T}_{=G'} - \mathbb{1}\bar{X}' X'^T - X'(\mathbb{1}\bar{X}')^T + \mathbb{1}\bar{X}'\bar{X}'^T\mathbb{1}^T$$

[define average in matrix notation: $\bar{X}' = \frac{1}{N}\sum_i X_i' = \frac{1}{N}(\mathbb{1}^T X')$]

$$= G' - \mathbb{1}\cdot\frac{1}{N}\mathbb{1}^T \underbrace{X' X'^T}_{=G'} - X' \underbrace{\bar{X}'^T}_{\frac{1}{N}X'^T\mathbb{1}} \mathbb{1}^T + \mathbb{1}\cdot\frac{1}{N}\mathbb{1}^T X' \underbrace{\left(\frac{1}{N} X'^T \mathbb{1}\right)}_{G'} \mathbb{1}^T$$

$$= G' - \mathbb{1}\underbrace{\left(\frac{1}{N}\mathbb{1}^T G'\right)}_{\bar{G}'} - \underbrace{\left(\frac{1}{N} G' \mathbb{1}\right)}_{\bar{G}'^T}\mathbb{1}^T + \mathbb{1}\underbrace{\left(\frac{1}{N^2}\mathbb{1}^T G' \mathbb{1}\right)}_{\gamma}\mathbb{1}^T$$

$\bar{G}'$ row vector of column averages

γ: average value of G'

post-centering of Gram matrix:

(1) compute $G' = X' X'^T$ uncentered Gram matrix

(2) compute row vector of column averages $\bar{G}' = \frac{1}{N} \mathbb{1}^T G'$ and total average

and total average $\gamma = \frac{1}{N^2} \mathbb{1}^T G' \mathbb{1}$

(3) compute centered Gram matrix $\boxed{G = G' - \mathbb{1} \bar{G}' - \bar{G}'^T \mathbb{1}^T + \gamma \mathbb{1} \mathbb{1}^T}$

This also works when $G'_{ij} = k(x_i, x_j)$ with kernel trick

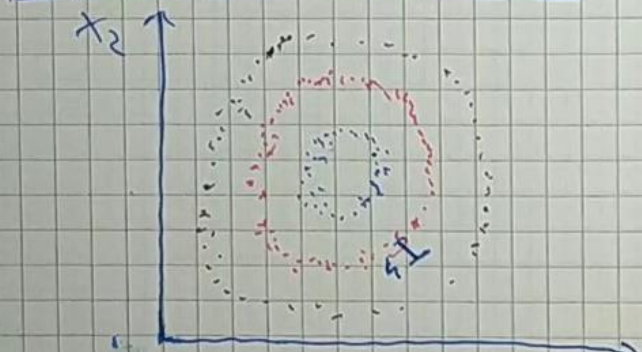
⇒ kernel PCA:

(1) define G' from $k(x_i, x_j)$

(2) center: $G = G' - \dots$ (as above)

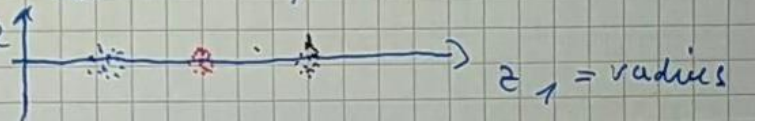
(3) define new features $z_j = u_j \Lambda_j^{1/2}$ with $G u = \lambda u$ the eigen-decomp.

why is this useful?



PCA has no effect: data form a circle ⇒ features x_1 and x_2 are already uncorrelated, eigenvectors V of Σ are unit vectors, $z = x$

kernel PCA with squared exponential kernel with correct bandwidth h (larger than dist within rings, but smaller than distance between rings)



x_1 ⇒ learns "polar coordinates"

Dimension reduction: compute new features $z = \varphi(x)$

$\dim(z_i) \leq \dim(x_i)$, φ linear (PCA), non-linear (k-PCA)

- purposes: - use z as input of some other ML algorithm,
  instead of x => may run faster, because lower dimension,
  may be more accurate, because z contains the essence of x but not its noise
  - use z for visualization: $\dim(z) = 2$ or 3, draw as picture
    => the essential behavior of x may still be visible (e.g. clusters)

- further non-linear algorithms: Local Linear Embedding (LLE)
  - idea: - find for each i the k nearest neighbors according to distance $d(x_i, x_{i'})$ (chosen by designer)
    - express x_i as a linear combination of its neighbors:

$$x_i = \sum_{i' \in Nei(x_i)} w_{ii'} \cdot x_{i'} \quad \text{with} \sum_{i'} w_{ii'} = 1$$

    - choose new coordinates z_i such that the relation with neighbors is approximately preserved: $z_i \approx \sum_{i' \in Nei(x_i)} w_{ii'} z_{i'}$

=> effect: local relationships between instances are preserved, long-distance relations may change a lot.

- optimization problem: $\hat{w}^{(i)} = \arg\min_w \| x_i - w x \|^2$ s.t. $w_{ii'}^{(i)} = 0$ for $i' \notin Nei(x_i)$

and $\sum_{i'} w_{ii'}^{(i)} = 1$

simplify by slicing $\tilde{X}^{(i)}$: matrix of only neighbors of X_i

ex. $\sum_{i'} w_{ii'} = 1 \iff w \cdot \mathbb{1} = 1$

$$G^{(i)} = (\tilde{X}^{(i)} - X_i)(\tilde{X}^{(i)} - \mathbb{1} X_i)^T$$

centering of $\tilde{X}^{(i)}$ around X_i

Gram matrix centered around X_i

$$\hat{w}^{(i)} = \underset{w,\lambda}{\operatorname{argmin}}\ w G^{(i)} w^T + \lambda (1 - w \mathbb{1})$$

$$\frac{\partial}{\partial w}: \quad 2 G^{(i)} w^T - \lambda \mathbb{1} \stackrel{!}{=} 0 \qquad u = \frac{2 w^T}{\lambda}$$

$G^{(i)} u = \mathbb{1}$ linear system, solve with standard methods

$$\hat{u}^{(i)} = (G^{(i)})^{-1} \cdot \mathbb{1}$$

$$\hat{w}^{(i)} = \frac{(\hat{u}^{(i)})^T}{(\hat{u}^{(i)})^T \cdot \mathbb{1}}$$

repeat for every i

embedding: $\hat{z} = \underset{z}{\operatorname{argmin}} \sum_i \| z_i - \hat{w}^{(i)} \tilde{z}^{(i)} \|^2$ $\quad \tilde{z}^{(i)}$: only neighbors of i

rewrite in matrix notation: $\tilde{w}^{(i)}$: $\hat{w}^{(i)}$ filled with zeros for non-neighbors

$\tilde{W}$: matrix of $\tilde{w}^{(i)}$

$$\hat{z} = \underset{z}{\operatorname{argmin}}\ tr\, (\mathbb{1} z - \tilde{W} z)^T (z - \tilde{W} z)$$

$$= \underset{z}{\operatorname{argmin}}\ tr\, (z^T (\mathbb{1} - \tilde{W})^T (\mathbb{1} - \tilde{W}) z)$$

$$= \underset{z}{\operatorname{argmin}}\ tr\, (z^T M z) \qquad M = (\mathbb{1} - \tilde{W})^T (\mathbb{1} - \tilde{W})$$

$$= U \Lambda U^T \quad \text{eigen-decomp.}$$

assume that eigenvalues are sorted $\Rightarrow u_D$ is the eigenvector for λ_D, smallest eigenvalue

$\lambda_D = 0$ because M has rank $D-1$ ($M \cdot \mathbb{1} = 0 \cdot \mathbb{1}$)

$\lambda_{D-1} \neq 0$ $\Rightarrow$ this minimizes the objective

when $\dim(z) = D'$ $\Rightarrow$ use the columns $u_{D-D'}, \dots u_{D-1}$ as new features

$$z_{ij} = u_{i,D-D'-1+j}$$

- embed new data: x_{new}:
  - find the k nearest neighbors of x_{new} in TS
  - find w_{new} such that $w_{new} = \arg\min_w \| x_{new} - w \tilde{x}^{(new)} \|^2$ ← neighbors
  - define $z^{(new)} = w_{new} \cdot \tilde{z}^{(new)}$ ← neighbors of x_{new} in z-space

- example

$\Rightarrow z_1$ LLE unrolls the spiral (k-PCA cannot do this!)

- in practice: t-SNE is most popular $\stackrel{!}{=}$ improvement of LLE

---

new instance x_{new}: row vector of size D

w_{new}: row vector of size k

z_{new}: row vector of size D', $\tilde{z}^{(new)} \in \mathbb{R}^{k \times D'}$ ← new features of k nearest neighbors from TS

$\tilde{x}^{(new)} \in \mathbb{R}^{k \times D}$ ← k rows for k nearest neighbors.

$\tilde{x}^{(new)}_{j1} \in TS$ s.t. $d(x_{new}, \tilde{x}^{(new)}_{j1})$ is small