1.1.3

$Av = \lambda v$

$(1) \boldsymbol{v}^T \boldsymbol{A}\boldsymbol{v} = \boldsymbol{v}^T \lambda \boldsymbol{v} = \lambda \boldsymbol{v}^T \boldsymbol{v} = \lambda v^2 \quad \forall \boldsymbol{v} \neq \boldsymbol{0}$?

$\lambda_i(\boldsymbol{A}) > 0 => \lambda v^2 \not=> from(1) \boldsymbol{v}^T \boldsymbol{A}\boldsymbol{v} > 0 \quad \forall \boldsymbol{v} \neq \boldsymbol{0}$? -2

$\boldsymbol{v}^T \boldsymbol{A}\boldsymbol{v} > 0 => from(1) \lambda v^2 > 0 => \lambda_i(\boldsymbol{A}) > 0 \quad \forall \boldsymbol{v} \neq \boldsymbol{0}$

1.2.1

Prove th at:

1. The mean of $\mathcal{Z} = \{\boldsymbol{z}_i\}_{i=1}^N$ is zero, that is, $\boldsymbol{\mu}_z = \frac{1}{N}\sum_{i=1}^N \boldsymbol{z}_i = \boldsymbol{0}$.
2. The covariance of $\mathcal{Z}$ is diagonal, that is $\boldsymbol{\Sigma}_z$ is diagonal.
3. $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2 = \|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2$ for all i and j.

#1. ?

$\frac{1}{N}\sum_{i=1}^N \boldsymbol{z}_i = \boldsymbol{0} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{U}^T(\boldsymbol{x}_i - \boldsymbol{\mu}_x) = \frac{1}{N}\boldsymbol{U}^T\sum_{i=1}^N (\boldsymbol{x}_i - \boldsymbol{\mu}_x) = \boldsymbol{0}$

$^*\sum_{i=1}^N (\boldsymbol{x}_i - \boldsymbol{\mu}_x) = 0$

#2.

$\boldsymbol{\Sigma}_z = \mathbb{E}(ZZ^T) = \mathbb{E}((U^T(x_i - \mu_i))(U^T(x_i - \mu_i))^T) = \mathbb{E}(ZZ^T) = \mathbb{E}((U^T(x_i - \mu_i))((x_i - \mu_i)^T)U) = \mathbb{E}(ZZ^T) = U^T\mathbb{E}((x_i - \mu_i)(x_i - \mu_i)^T)U$
$= U^T\boldsymbol{\Sigma}_x U = U^T\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T U = \boldsymbol{\Lambda}$

$\boldsymbol{\Lambda}$ is diagonal and $\boldsymbol{\Sigma}_z = \boldsymbol{\Lambda} => \boldsymbol{\Sigma}_z$ is diagonal

#3. why?

$\|\boldsymbol{z}_i - \boldsymbol{z}_j\|_2 = \left\|\boldsymbol{U}^T(\boldsymbol{x}_i - \boldsymbol{\mu}_x) - \boldsymbol{U}^T(\boldsymbol{x}_j - \boldsymbol{\mu}_x)\right\|_2 = \|(\boldsymbol{U}^T\boldsymbol{x}_i - \boldsymbol{U}^T\boldsymbol{\mu}_x - \boldsymbol{U}^T\boldsymbol{x}_j + \boldsymbol{U}^T\boldsymbol{\mu}_x\|_2 = \|(\boldsymbol{U}^T\boldsymbol{x}_i - \boldsymbol{U}^T\boldsymbol{x}_j\|_2 = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$

-1

1.2.7

We'll use the hint and plug $\boldsymbol{M} = \boldsymbol{B}\boldsymbol{C}$ into the equivalent unconstrained problem while:

$\boldsymbol{M} \in \mathbb{R}^{D\times N}$ with rank $(\boldsymbol{M}) = d$

(a) $\boldsymbol{B} \in \mathbb{R}^{D\times d}$

(b) $\boldsymbol{C} \in \mathbb{R}^{d\times N}$

So, $\min_{\boldsymbol{M}\in\mathbb{R}^{D\times N}} \|\boldsymbol{A} - \boldsymbol{M}\|_F^2 = \min_{\cancel{\boldsymbol{M}\in\mathbb{R}^{D\times N}}} \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2$

Now, as we learnt in class, sience a solution is up to some matrix Y:

$BC = BYY^{-1}C$

so we add ortogonality constraint as follow:

$$\begin{cases} \min_{\boldsymbol{B}_d\in\mathbb{R}^{D\times d}, \boldsymbol{C}\in\mathbb{R}^{d\times N}} \|\boldsymbol{A} - \boldsymbol{B}_d C\|_F^2 \\ \text{s.t. } \boldsymbol{B}_d^T\boldsymbol{B}_d = \boldsymbol{I}_d \end{cases}$$

we assume that is B sem-orthogonal matrix

So now we want to find the optimal C as a function of A and B. We'll calculate the gradient:

$$\begin{aligned} \nabla_C \|A - \boldsymbol{B}_d C\|_F^2 &= \boldsymbol{0} \\ -2\boldsymbol{B}_d^T(\boldsymbol{A} - \boldsymbol{B}_d C) &= \boldsymbol{0} \\ \boldsymbol{B}_d^T\boldsymbol{A} - \underbrace{\boldsymbol{B}_d^T\boldsymbol{B}_d}_{=\boldsymbol{I}_d} C &= \boldsymbol{0} \\ C &= \boldsymbol{B}_d^T\boldsymbol{A} \end{aligned}$$

and in truncated form we'll define $\boldsymbol{B} = \boldsymbol{B}_d^T$ and $\boldsymbol{C}^T = \boldsymbol{C}$ and get: $\boldsymbol{A} = \boldsymbol{B}\Sigma\boldsymbol{C}^T$

B = ?

+1

B = ?

M = ?

+1

2.1.3

Show that if k can be written as an inner product, that is

$$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left\langle \phi\left(\boldsymbol{x}_i\right), \phi\left(\boldsymbol{x}_j\right)\right\rangle$$

for some ϕ, then, the matrix defined by:

$$\boldsymbol{K}_x[i,j] = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$$

is an SPSD matrix, namely, $\boldsymbol{K}_x \succeq 0$

$\boldsymbol{K}_x[i,j] = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ is SPSD matrix beacuse $\boldsymbol{K}_x$ is symtric $\boldsymbol{K}_x[i,j] = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left\langle \phi\left(\boldsymbol{x}_i\right), \phi\left(\boldsymbol{x}_j\right)\right\rangle = \left\langle \phi\left(\boldsymbol{x}_j\right), \phi\left(\boldsymbol{x}_i\right)\right\rangle = k\left(\boldsymbol{x}_j, \boldsymbol{x}_i\right) = \boldsymbol{K}_x[j,i]$

we will show that each $\cancel{\boldsymbol{K}_x[i,j] >= 0}$

$\boldsymbol{K}_x[i,j] = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left\langle \phi\left(\boldsymbol{x}_i\right), \phi\left(\boldsymbol{x}_j\right)\right\rangle \cancel{=} \phi^T\phi >= 0 => \boldsymbol{K}_x[i,j]$is SPSD matrix

-2

2.1.5

$$k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) := \left\langle \phi\left(\boldsymbol{x}_i\right), \phi\left(\boldsymbol{x}_j\right)\right\rangle$$

for some ϕ.

- Let:

$$\tilde{k}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) := \left\langle \phi\left(\boldsymbol{x}_i\right) - \boldsymbol{\mu}_\phi, \phi\left(\boldsymbol{x}_j\right) - \boldsymbol{\mu}_\phi\right\rangle$$

  be the centered version, where:

$$\boldsymbol{\mu}_\phi = \frac{1}{N}\sum_{i=1}^{N} \phi\left(\boldsymbol{x}_i\right)$$

$\tilde{k}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) := \left\langle \phi\left(\boldsymbol{x}_i\right) - \boldsymbol{\mu}_\phi, \phi\left(\boldsymbol{x}_j\right) - \boldsymbol{\mu}_\phi\right\rangle = \left\langle \phi\left(\boldsymbol{x}_i\right), \phi\left(\boldsymbol{x}_j\right)\right\rangle - \left\langle \phi\left(\boldsymbol{x}_i\right), \boldsymbol{\mu}_\phi\right\rangle - \left\langle \phi\left(\boldsymbol{x}_j\right), \boldsymbol{\mu}_\phi\right\rangle + \left\langle \boldsymbol{\mu}_\phi, \boldsymbol{\mu}_\phi\right\rangle = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$ $- \left\langle \phi\left(\boldsymbol{x}_i\right), \frac{1}{N}\sum_{j=1}^{N} \phi\left(\boldsymbol{x}_j\right)\right\rangle - \left\langle \phi\left(\boldsymbol{x}_j\right), \frac{1}{N}\sum_{i=1}^{N} \phi\left(\boldsymbol{x}_i\right)\right\rangle + \left\langle \frac{1}{N}\sum_{i=1}^{N} \phi\left(\boldsymbol{x}_i\right), \frac{1}{N}\sum_{j=1}^{N} \phi\left(\boldsymbol{x}_j\right)\right\rangle = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \frac{1}{N}\sum_{j=1}^{N}\left\langle \phi\left(\boldsymbol{x}_i\right), \phi\left(\boldsymbol{x}_j\right)\right\rangle - \frac{1}{N}$ $\sum_{i=1}^{N}\left\langle \phi\left(\boldsymbol{x}_j\right), \phi\left(\boldsymbol{x}_i\right)\right\rangle + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{i=1}^{N}\left\langle \phi\left(\boldsymbol{x}_i\right), \phi\left(\boldsymbol{x}_j\right)\right\rangle = k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \frac{1}{N}\sum_{j=1}^{N} k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \frac{1}{N}\sum_{i=1}^{N} k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{i=1}^{N} k\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right)$

-1

2.1.6

DisProve

if every $\cancel{K[i,j] = 3}$ then the mean of the colums is 3 and the mean of the rows is 3 and after we substract the mean we can see that not every $k[i,j] <= 0$ and ~~that why~~ the metrix is not SPD matrix.

-2

2.1.7

$\phi = \phi(X)$

$K_x^* = \phi^T\phi$

$\widetilde{\Phi}^T \tilde{\phi}^\star \mapsto \tilde{k}x = J\left(kx - \frac{1}{N}K_x\mathbf{1}_N\right)$

$Z = \Sigma_d^{-1} V_d^T \cancel{\phi^T}\phi$

-2

2.1.8

We knowing that:

The training encoding $Z = \Sigma_d V_d^T$

The OOSE $Z^* = \Sigma_d^{-1} V_d^T \bar{k}_x$

Where

$\bar{k}_x = V_d \Sigma_d^2 V_d^T$

and

$V^T V = I$

Therefore:

$\underline{\underline{X}}^*$ is a subset of $\underline{\underline{X}}$

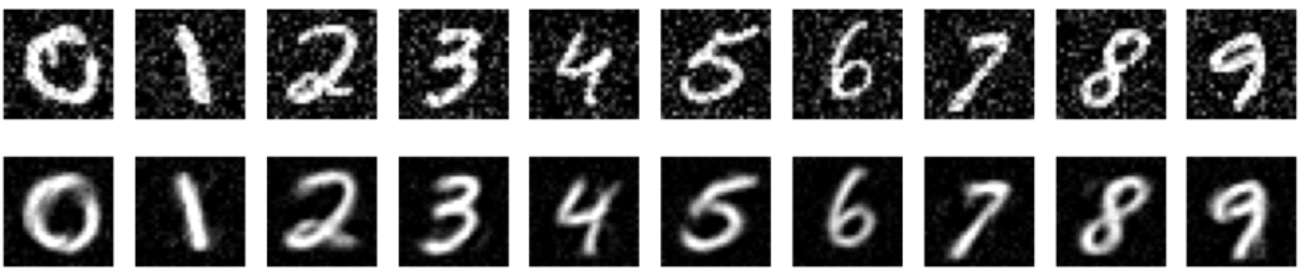
-1

$$Z^* =$$
$$\Sigma_d^{-1} V_d^T \bar{k}_x =$$
$$\Sigma_d^{-1} V^T V \Sigma_d^2 V_d^T =$$
$$\Sigma_d^{-1} \Sigma_d^2 V_d^T =$$
$$\Sigma_d V_d^T = Z$$

```
def test_d_pca(d):

    count = 0
    for i in range(10):
        x = mX[:, np.where(vY == i)[0]]
        my_pca = PCA(d)
        my_pca.Fit(x)
        mZ = my_pca.Encode(x)
        mHatX = my_pca.Decode(mZ)
        vIdx[i] = count
        if i == 0:
            result = mHatX
        else:
            result = np.column_stack((result, mHatX))
        count += mHatX.shape[1]


    plot_mX(result)

for d in range(20, 160, 20):

    test_d_pca(d)

plt.tight_layout()
plt.show()

mX.shape, vY.shape
```

2.

+5



```python
import scipy.stats as st
from scipy.spatial.distance import cdist

def k_1(x1, x2):

    return x1.T @ x2

def k_2(x1, x2):
    p = 2
    return np.power(1+ x1.T @ x2, p)

def k_3(x1, x2):

    dist_matrix = cdist(x1.T, x2.T)
    sigma = np.median(dist_matrix)
    b  = 2*sigma**2
    return np.exp(-dist_matrix/b)
```

1. **sigma must be a fixed value**
2. **dist_matrix**2**

-2

a