TEXAS TECH UNIVERSITY

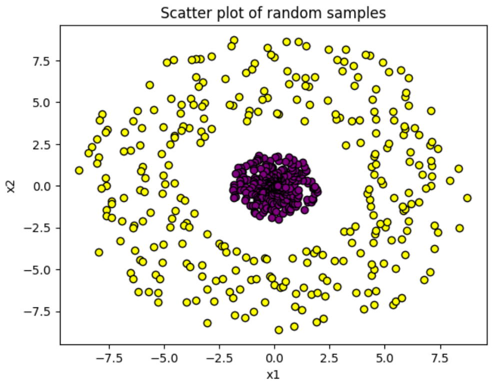
Kernelized Principal Component Analysis

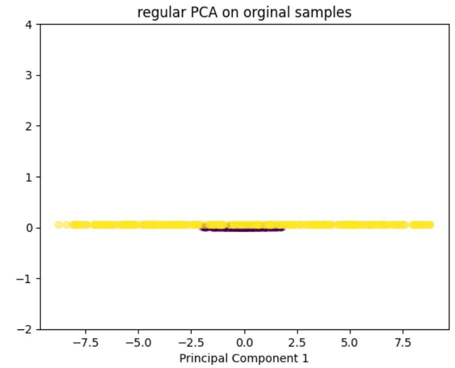
Tyreek Alexander
Ph.D. Student

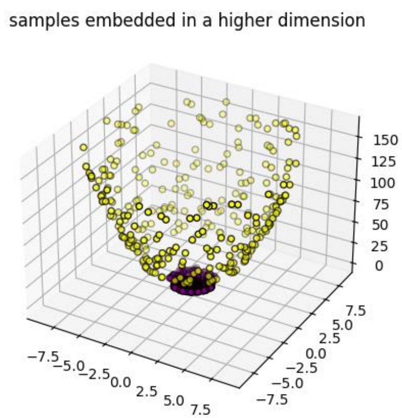
Friday, May 3rd, 2024

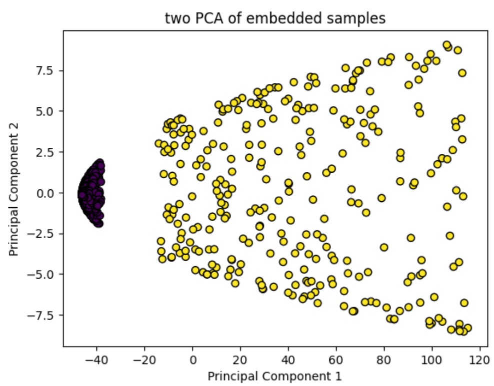
Kernel PCA

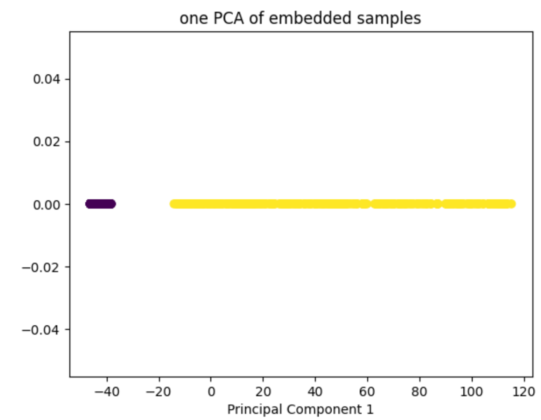
- Definition:
    - Given a set of samples in the original space, we project each sample onto a new embedding of higher dimensionality. We perform PCA in this higher dimension. The result is a non-linear projection of samples in the dimension of required principal components.











Kernel PCA Basics

- Computing the mean
  - Original space: $\mu = \frac{1}{N}\sum_{k=1}^{N} x^k$
  - Embedded space: $\overline{\Phi} = \frac{1}{N}\sum_{k=1}^{N} \Phi(x^k)$
- Covariance matrix
  - Original space: $\Sigma(x^k - \mu)(x^k - \mu)^T$
  - Embedded space: $\Sigma_\Phi = \left(\Phi(x^k) - \overline{\Phi}\right) \ \left(\Phi(x^k) - \overline{\Phi}\right)^T$
- Projection
  - Original space: $y^k = \left(x^k - \mu\right)^T \mathrm{e}$
  - Embedded space: $\Phi(y^k) = \left(\Phi(x^k) - \overline{\Phi}\right)^T e_\Phi$

Kernel PCA Mathematical Foundation

- The choice of embedding has significant impact on computational complexity. Example: polynomial embedding.

$$\Phi[x_1, x_2, \dots, x_d] = \left[x_1^n, x_1^{n-1}x_2\ , \dots, x_1^{n-1}x_d\ , \dots, x_1^{n-2}x_2^2, \dots, x_d^n\right]$$

- $\Phi[X]$ is an $[m \times n]$ matrix where m is the number of features/dimensions in the embedded space.
- Increasing the degree of the polynomial, n, increases m.

$$m = {}^{d+n-1}_{\quad\ \ d}C$$

- If m is large the computation is impractical. Σ_Φ is an $[m \times m]$ matrix.

$$\Sigma_\Phi = \Phi\Phi^T$$

- How can we achieve efficient PCA in the embedded space?
- If we proceed and try to find the eigenvectors:

$$\Phi\Phi^T e = \lambda e$$

For all $\lambda > 0$: $\qquad \Phi\dfrac{(\Phi^T e)}{\lambda} = e,$

$$\text{let } \frac{(\Phi^T e)}{\lambda} = w$$

Kernel PCA Mathematical Foundation

- The result is $\Phi w = e$ this tells us that the eigenvectors we want, e, is always a linear combination of the columns of the embedded data matrix.
- Eigenvectors of Σ_Φ that correspond to $\lambda > 0$ are in the column span of Φ.
$$w = \lambda^{-1}\Phi^T e$$
- w is the vector of coefficients.
- If we take the definition of w
$$\Phi^T\Phi w = \Phi^T\Phi(\lambda^{-1}\Phi^T e)$$
$$= \Phi^T e$$
$$\Phi^T\Phi w = \lambda w$$
- Initially $\Phi\Phi^T$was too large $[m \times m]$ but now we have $\Phi^T\Phi$ which is $[n \times n]$, where $n \ll m$
- Instead of computing eigenvectors of Σ_Φ; $\Phi\Phi^T e = \lambda e$, we can find eigenvectors w, since the eigenvalues λ are essentially the same where $\Phi^T\Phi w = \lambda w$

Note: $\|w\|^2 = w^T w = u^T \Phi\Phi^T u \left(\frac{1}{\lambda^2}\right) = u^T(\lambda u)\left(\frac{1}{\lambda^2}\right) = \frac{1}{\lambda}$

Kernel PCA Mathematical Foundation

- Solving the eigen problem $\Phi^T \Phi w = \lambda w$, we get w and λ. This will yield $\|w\|^2 = 1$ since the norm of symmetric
- We then normalize w so that $\|w\|^2 = \frac{1}{\lambda}$ then e the eigenvector we want is equal to $\Phi w = e$.

$$\tilde{w} = \frac{1}{\sqrt{\lambda}}$$

- If we define our K= $\Phi^T \Phi$ as our Kernel matrix, we can find the projection y of the original data x

$$y = e^T \Phi$$
$$y = w^T \Phi^T \Phi; \quad e = \Phi w$$
$$y = \tilde{w}^T K$$

.

Kernel PCA Algorithm

1. Given samples $[x_1, x_2, \dots, x_n]$, a Kernel function $K(x, y)$, and number of principal components d.
2. Calculate $K_{ij} = K\left(x_i, x_j\right), \widetilde{K} = JKJ^T$:
3. Perform an Eigen Decomposition of $\widetilde{K} \approx W\Lambda W^{-1}$ for the top d eigenvectors.
4. $\boldsymbol{Y} = \widetilde{\boldsymbol{W}}^{\boldsymbol{T}}\boldsymbol{K}$

Kernel PCA extensions

- Kernel recompositing of original samples
  - Unlike standard PCA, the transformed data in the higher-dimensional space cannot be directly mapped back to the original feature space.
- Kernelizing Out of sample data:
  - If we want to project a new data point, we can use the kernel trick to calculate the kernel between the new data point and all the samples:

$$y^* = w^T K_x$$

$$\text{Where } K_x = [K(x, x_1), K(x, 2), \ldots K(x, n)]^T$$

- Using Label-Based Weights
  - One way of improving the separation of data points belonging to different classes is to modify the kernel matrix by using the labels.
  - A weight, α_{ij}, proportional to the difference in label values, can be inserted into the kernel function.

$$K_{ij} = \exp(-\alpha_{ij}\gamma \|x_i - x_j\|^2)$$

- An embedding exist for any random Kernel.