PCA (Wrap-up) and Nonlinear Dimensionality Reduction via Kernel PCA

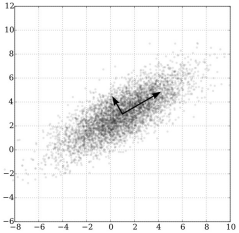
Piyush Rai

Machine Learning (CS771A)

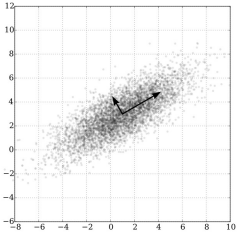
Sept 7, 2016

Recap/Wrap-up of PCA

Principal Component Analysis (PCA)

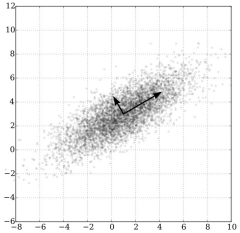


Principal Component Analysis (PCA)



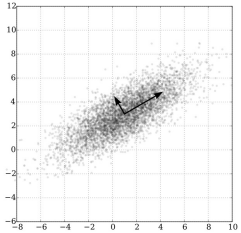
- PCA basically does the following

Principal Component Analysis (PCA)



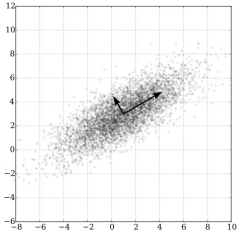
- PCA basically does the following
  - Learns the most important directions (new basis vectors) in the data

Principal Component Analysis (PCA)



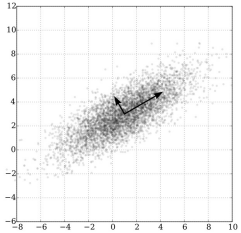
- PCA basically does the following
  - Learns the most important directions (new basis vectors) in the data
  - *Re*-represents data using the new basis vectors (change of basis)

Principal Component Analysis (PCA)



- PCA basically does the following
  - Learns the most important directions (new basis vectors) in the data
  - *Re*-represents data using the new basis vectors (change of basis)
  - Throws away "uninteresting" directions without much loss of info. This gives a new lower-dimensional representation of the original data

Principal Component Analysis (PCA)



- PCA basically does the following
  - Learns the most important directions (new basis vectors) in the data
  - *Re*-represents data using the new basis vectors (change of basis)
  - Throws away "uninteresting" directions without much loss of info. This gives a new lower-dimensional representation of the original data
- PCA uses "amount of data variance captured" to define important directions

Principal Component Analysis (PCA)

- How does PCA *find* the "maximum variance" directions (last class)?

Principal Component Analysis (PCA)

- How does PCA *find* the "maximum variance" directions (last class)?
  - Using eigen-decomposition of the **covariance matrix** of data

Principal Component Analysis (PCA)

- How does PCA *find* the "maximum variance" directions (last class)?
  - Using eigen-decomposition of the **covariance matrix** of data
  - Each eigenvector represents one such direction

Principal Component Analysis (PCA)

- How does PCA *find* the "maximum variance" directions (last class)?
  - Using eigen-decomposition of the **covariance matrix** of data
  - Each eigenvector represents one such direction
  - First (top) eigenvector is the direction that captures the largest variance

Principal Component Analysis (PCA)

- How does PCA *find* the "maximum variance" directions (last class)?
  - Using eigen-decomposition of the **covariance matrix** of data
  - Each eigenvector represents one such direction
  - First (top) eigenvector is the direction that captures the largest variance
  - Each subsequent eigenvector is the next best as per this criterion

Principal Component Analysis (PCA)

- How does PCA *find* the "maximum variance" directions (last class)?
  - Using eigen-decomposition of the **covariance matrix** of data
  - Each eigenvector represents one such direction
  - First (top) eigenvector is the direction that captures the largest variance
  - Each subsequent eigenvector is the next best as per this criterion
- **Steps in Principal Component Analysis**
  - Compute the covariance matrix $\mathbf{S}$ using the centered data as

$$\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top \qquad \text{(note: } \mathbf{X} \text{ assumed } D \times N \text{ here)}$$

Principal Component Analysis (PCA)

- How does PCA *find* the "maximum variance" directions (last class)?
  - Using eigen-decomposition of the **covariance matrix** of data
  - Each eigenvector represents one such direction
  - First (top) eigenvector is the direction that captures the largest variance
  - Each subsequent eigenvector is the next best as per this criterion
- **Steps in Principal Component Analysis**
  - Compute the covariance matrix $\mathbf{S}$ using the centered data as

$$\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top \qquad \text{(note: } \mathbf{X} \text{ assumed } D \times N \text{ here)}$$

  - Do an eigen-decomposition of $\mathbf{S}$. This will give D eigenvectors.

Principal Component Analysis (PCA)

- How does PCA *find* the "maximum variance" directions (last class)?
  - Using eigen-decomposition of the **covariance matrix** of data
  - Each eigenvector represents one such direction
  - First (top) eigenvector is the direction that captures the largest variance
  - Each subsequent eigenvector is the next best as per this criterion
- **Steps in Principal Component Analysis**
  - Compute the covariance matrix $\mathbf{S}$ using the centered data as

$$\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top \qquad \text{(note: } \mathbf{X} \text{ assumed } D \times N \text{ here)}$$

  - Do an eigen-decomposition of $\mathbf{S}$. This will give D eigenvectors.
  - Take top K leading eigenvectors $\{\boldsymbol{u}_k\}_{k=1}^K$ with eigenvalues $\{\lambda_k\}_{k=1}^K$

Principal Component Analysis (PCA)

- How does PCA *find* the "maximum variance" directions (last class)?
  - Using eigen-decomposition of the **covariance matrix** of data
  - Each eigenvector represents one such direction
  - First (top) eigenvector is the direction that captures the largest variance
  - Each subsequent eigenvector is the next best as per this criterion
- **Steps in Principal Component Analysis**
  - Compute the covariance matrix $\mathbf{S}$ using the centered data as

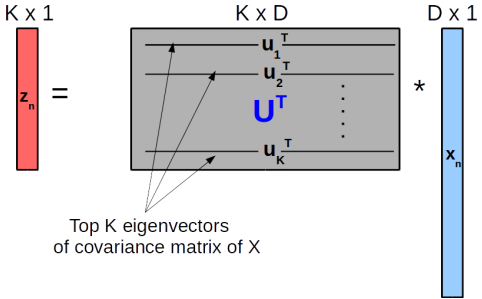
$$\mathbf{S} = \frac{1}{N}\mathbf{X}\mathbf{X}^\top \qquad \text{(note: } \mathbf{X} \text{ assumed } D \times N \text{ here)}$$

  - Do an eigen-decomposition of $\mathbf{S}$. This will give D eigenvectors.
  - Take top K leading eigenvectors $\{\boldsymbol{u}_k\}_{k=1}^K$ with eigenvalues $\{\lambda_k\}_{k=1}^K$
  - $\mathbf{U} = [\boldsymbol{u}_1 \ \ldots \ \boldsymbol{u}_K]$ is $D \times K$ matrix (each column is a projection direction)

PCA as Linear Projection

- Can use $\mathbf{U}$ to linearly project each $\boldsymbol{x}_n \in \mathbb{R}^D$ to a K-dim subspace as

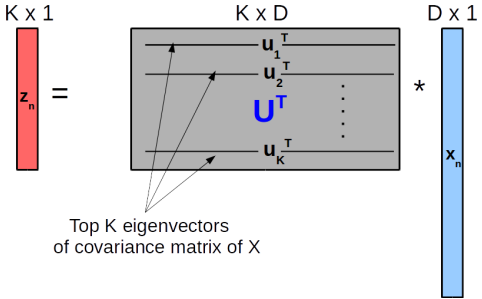
$$\boldsymbol{z}_n = \mathbf{U}^\top \boldsymbol{x}_n = [\boldsymbol{u}_1^\top \boldsymbol{x}_n \ \boldsymbol{u}_2^\top \boldsymbol{x}_n \ \ldots \ \boldsymbol{u}_K^\top \boldsymbol{x}_n]^\top$$



PCA as Linear Projection

- Can use $\mathbf{U}$ to linearly project each $\boldsymbol{x}_n \in \mathbb{R}^D$ to a K-dim subspace as

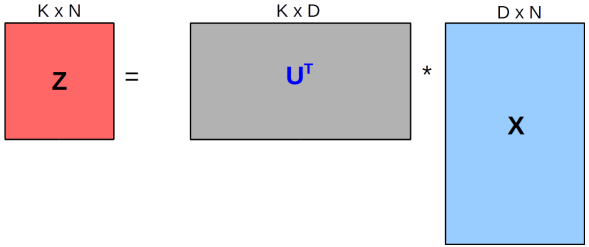
$$\boldsymbol{z}_n = \mathbf{U}^\top \boldsymbol{x}_n = [\boldsymbol{u}_1^\top \boldsymbol{x}_n \ \boldsymbol{u}_2^\top \boldsymbol{x}_n \ \ldots \ \boldsymbol{u}_K^\top \boldsymbol{x}_n]^\top$$



- $\boldsymbol{z}_n \in \mathbb{R}^K$ is also called low-dimensional "embedding" of $\boldsymbol{x}_n \in \mathbb{R}^D$

PCA as Linear Projection

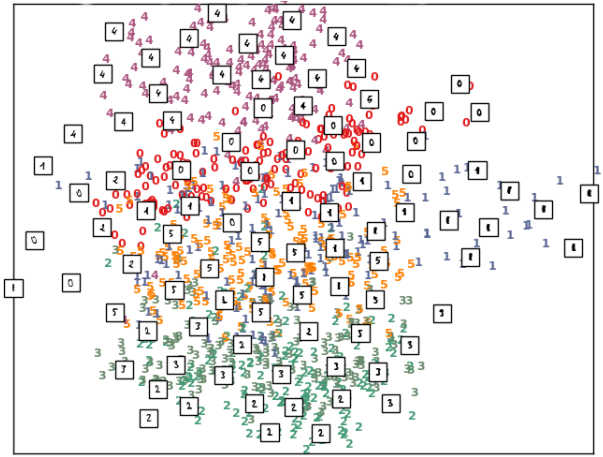
- $\mathbf{Z} = [\boldsymbol{z}_1\ \boldsymbol{z}_2 \ldots\ \boldsymbol{z}_n]$ is the $K \times N$ matrix of embeddings of all the N examples



- $\mathbf{Z}$ can also be thought of as a new, compact feature representation of $\mathbf{X}$

PCA based Embeddings of Handwritten Digits

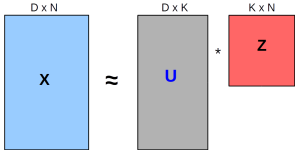
- Shown below are 2-dim embeddings of PCA on a set of handwritten digits (each digit image was originally 8×8, i.e., 64 dimensional)



PCA: Data as Combination of Basis Vectors

- PCA to K-dims is also akin to saying $\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{u}_k$. Thus

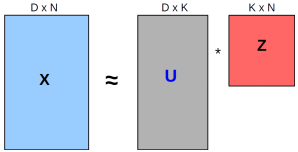
$$\mathbf{X} \approx \mathbf{UZ} \qquad \text{(matrix factorization)}$$



PCA: Data as Combination of Basis Vectors

- PCA to K-dims is also akin to saying $\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{u}_k$. Thus

$$\mathbf{X} \approx \mathbf{UZ} \qquad \text{(matrix factorization)}$$

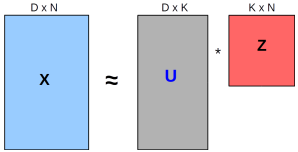


- Example: Each face in a collection can be represented as a combination of a small no of "eigenfaces" ("template" faces)

PCA: Data as Combination of Basis Vectors

- PCA to K-dims is also akin to saying $\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{u}_k$. Thus

$$\mathbf{X} \approx \mathbf{UZ} \qquad \text{(matrix factorization)}$$



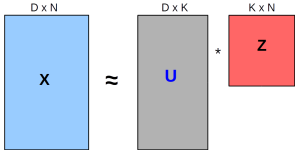
- Example: Each face in a collection can be represented as a combination of a small no of "eigenfaces" ("template" faces)

- Can thus approximately reconstruct the matrix $\mathbf{X}$ using $\mathbf{UZ}$: Do PCA on the $N \times D$ data matrix $\mathbf{X}$, keep $\mathbf{U}$ $(D \times K)$ and $\mathbf{Z}$ $(K \times N)$ and throw away $\mathbf{X}$

PCA: Data as Combination of Basis Vectors

- PCA to K-dims is also akin to saying $\boldsymbol{x}_n \approx \sum_{k=1}^{K} z_{nk} \boldsymbol{u}_k$. Thus

$$\mathbf{X} \approx \mathbf{UZ} \qquad \text{(matrix factorization)}$$



- Example: Each face in a collection can be represented as a combination of a small no of "eigenfaces" ("template" faces)

- Can thus approximately reconstruct the matrix $\mathbf{X}$ using $\mathbf{UZ}$: Do PCA on the $N \times D$ data matrix $\mathbf{X}$, keep $\mathbf{U}$ $(D \times K)$ and $\mathbf{Z}$ $(K \times N)$ and throw away $\mathbf{X}$
  - Substantial storage saving if $K \ll D$

PCA: Data as Combination of Basis Vectors

- Consider doing PCA on a (words $\times$ documents) matrix $\mathbf{X}$

PCA: Data as Combination of Basis Vectors

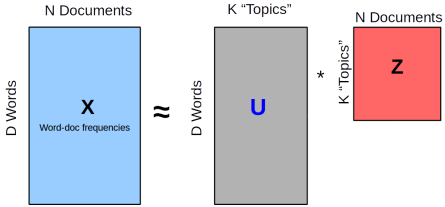
- Consider doing PCA on a (words $\times$ documents) matrix $\mathbf{X}$
- Each entry X_{dn} in $\mathbf{X}$ is the frequency of word d in document n

PCA: Data as Combination of Basis Vectors

- Consider doing PCA on a (words $\times$ documents) matrix $\mathbf{X}$
- Each entry X_{dn} in $\mathbf{X}$ is the frequency of word d in document n
- PCA on $\mathbf{X}$ will give us eigenvectors that correspond to "topics" or concepts

PCA: Data as Combination of Basis Vectors

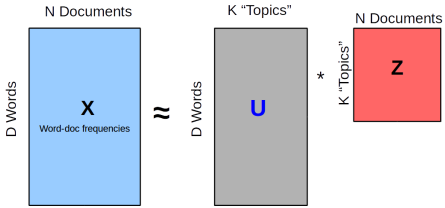
- Consider doing PCA on a (words $\times$ documents) matrix $\mathbf{X}$
- Each entry X_{dn} in $\mathbf{X}$ is the frequency of word d in document n
- PCA on $\mathbf{X}$ will give us eigenvectors that correspond to "topics" or concepts



- Each document is like a weighted combination of these topics

PCA: Data as Combination of Basis Vectors

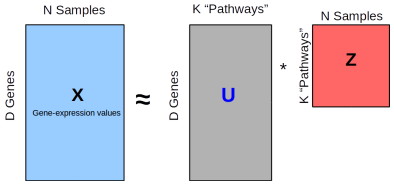
- Consider doing PCA on a (words $\times$ documents) matrix $\mathbf{X}$
- Each entry X_{dn} in $\mathbf{X}$ is the frequency of word d in document n
- PCA on $\mathbf{X}$ will give us eigenvectors that correspond to "topics" or concepts



- Each document is like a weighted combination of these topics
- This is similar to "Latent Semantic Analysis" (LSA), a well-known document dimensionality reduction technique in information retrieval (basically, LSA does SVD on $\mathbf{X}$, which is equivalent to doing PCA on $\mathbf{X}$)

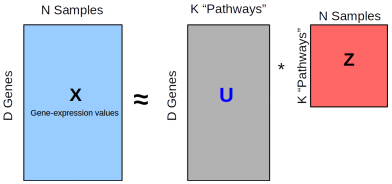
PCA: Data as Combination of Basis Vectors

- **X** is (genes $\times$ samples) matrix: Each sample (expression values of a set of D genes) is a weighted combination of K biological "pathways"

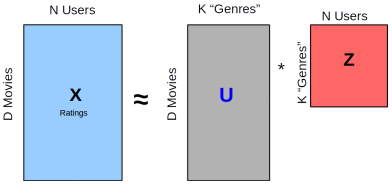


PCA: Data as Combination of Basis Vectors

- **X** is (genes $\times$ samples) matrix: Each sample (expression values of a set of D genes) is a weighted combination of K biological "pathways"

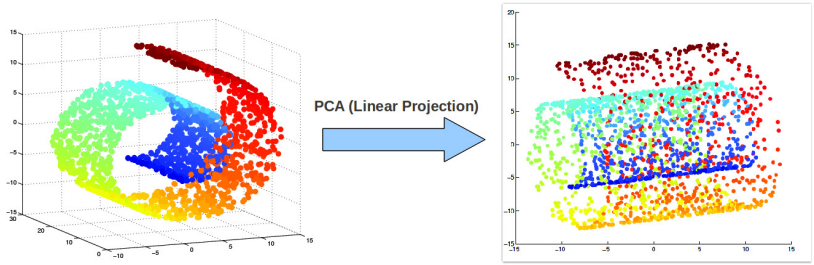


- **X** is (movies $\times$ users) matrix: Each user (represented by the vector of his/her ratings of D movies) is a weighted combination of K genres



Beyond Linear Projections..

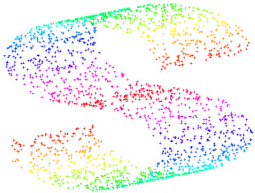
- Consider the swiss-roll dataset (points lying close to a manifold)



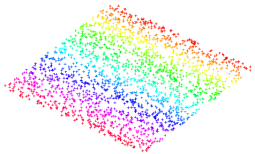
- Linear projection methods (e.g., PCA) can't capture intrinsic nonlinearities

Nonlinear Dimensionality Reduction

- Given: Low-dim. surface embedded **nonlinearly** in high-dim. space
  - Such a structure is called a **Manifold**

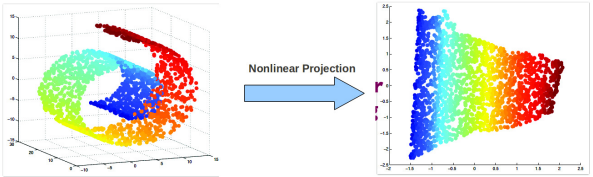


- Goal: Recover the low-dimensional surface



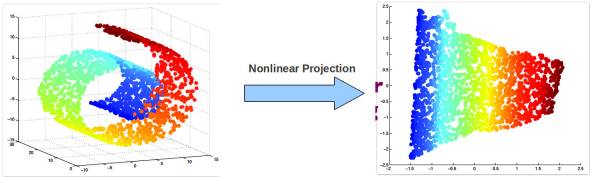
Nonlinear Dimensionality Reduction

- We want to a learn nonlinear low-dim projection



Nonlinear Dimensionality Reduction

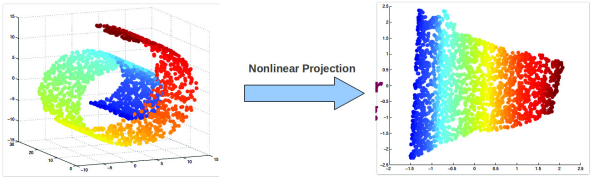
- We want to a learn nonlinear low-dim projection



- Usually two ways of doing this

Nonlinear Dimensionality Reduction

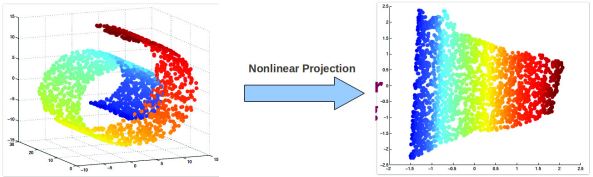
- We want to a learn nonlinear low-dim projection



- Usually two ways of doing this
  - Nonlinearize a linear dimensionality reduction method. E.g.:
    - **Kernel PCA (nonlinear PCA)**

Nonlinear Dimensionality Reduction

- We want to a learn nonlinear low-dim projection



- Usually two ways of doing this
  - Nonlinearize a linear dimensionality reduction method. E.g.:
    - **Kernel PCA (nonlinear PCA)**
  - Using manifold based methods that intrinsically preserve nonlinear geometry
    - Locally Linear Embedding (LLE)
    - Isomap
    - Maximum Variance Unfolding
    - Laplacian Eigenmaps
    - And others (tSNE, Hessian LLE, etc.)

Kernel PCA

- Recall PCA: Given N observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $\forall \boldsymbol{x}_n \in \mathbb{R}^D$, we define the $D \times D$ covariance matrix (assuming centered data $\sum_n \boldsymbol{x}_n = \boldsymbol{0}$)

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top$$

Kernel PCA

- Recall PCA: Given N observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $\forall \boldsymbol{x}_n \in \mathbb{R}^D$, we define the $D \times D$ covariance matrix (assuming centered data $\sum_n \boldsymbol{x}_n = \boldsymbol{0}$)

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top$$

- PCA computes eigenvectors $\boldsymbol{u}_i$ which satisfy $\mathbf{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \ \forall i = 1, \ldots, D$

Kernel PCA

- Recall PCA: Given N observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $\forall \boldsymbol{x}_n \in \mathbb{R}^D$, we define the $D \times D$ covariance matrix (assuming centered data $\sum_n \boldsymbol{x}_n = \boldsymbol{0}$)

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top$$

- PCA computes eigenvectors $\boldsymbol{u}_i$ which satisfy $\mathbf{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$ $\forall i = 1, \ldots, D$
- Let's assume a kernel k with associated M dimensional nonlinear map ϕ

Kernel PCA

- Recall PCA: Given N observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $\forall \boldsymbol{x}_n \in \mathbb{R}^D$, we define the $D \times D$ covariance matrix (assuming centered data $\sum_n \boldsymbol{x}_n = \boldsymbol{0}$)

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top$$

- PCA computes eigenvectors $\boldsymbol{u}_i$ which satisfy $\mathbf{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \ \forall i = 1, \ldots, D$
- Let's assume a kernel k with associated M dimensional nonlinear map ϕ
- $M \times M$ covariance matrix **in this space** (assume centered data $\sum_n \phi(\boldsymbol{x}_n) = \boldsymbol{0}$)

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n) \phi(\boldsymbol{x}_n)^\top$$

Kernel PCA

- Recall PCA: Given N observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $\forall \boldsymbol{x}_n \in \mathbb{R}^D$, we define the $D \times D$ covariance matrix (assuming centered data $\sum_n \boldsymbol{x}_n = \boldsymbol{0}$)

$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top$$

- PCA computes eigenvectors $\boldsymbol{u}_i$ which satisfy $\mathbf{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \ \forall i = 1, \ldots, D$
- Let's assume a kernel k with associated M dimensional nonlinear map ϕ
- $M \times M$ covariance matrix **in this space** (assume centered data $\sum_n \phi(\boldsymbol{x}_n) = \boldsymbol{0}$)

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n) \phi(\boldsymbol{x}_n)^\top$$

- Kernel PCA: Compute eigenvectors $\boldsymbol{v}_i$ satisfying: $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \ \forall i = 1, \ldots, M$

Kernel PCA

- Recall PCA: Given N observations $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $\forall \boldsymbol{x}_n \in \mathbb{R}^D$, we define the $D \times D$ covariance matrix (assuming centered data $\sum_n \boldsymbol{x}_n = \boldsymbol{0}$)

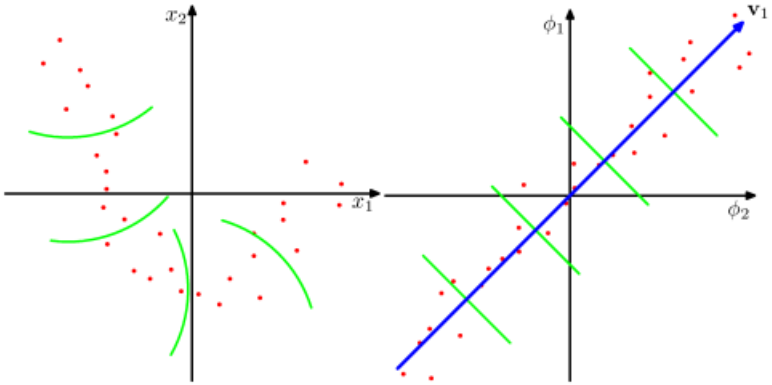
$$\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \boldsymbol{x}_n^\top$$

- PCA computes eigenvectors $\boldsymbol{u}_i$ which satisfy $\mathbf{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i \ \forall i = 1, \ldots, D$
- Let's assume a kernel k with associated M dimensional nonlinear map ϕ
- $M \times M$ covariance matrix **in this space** (assume centered data $\sum_n \phi(\boldsymbol{x}_n) = \boldsymbol{0}$)

$$\mathbf{C} = \frac{1}{N} \sum_{n=1}^{N} \phi(\boldsymbol{x}_n) \phi(\boldsymbol{x}_n)^\top$$

- Kernel PCA: Compute eigenvectors $\boldsymbol{v}_i$ satisfying: $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \ \forall i = 1, \ldots, M$
- We would like to do this without having to compute $\mathbf{C}$ or $\phi(\boldsymbol{x}_n)$'s

Kernel PCA



Right figure: After mapping the data via ϕ, data is now close to a linear subspace

Kernel PCA

- Goal: Compute eigenvectors $\boldsymbol{v}_i$, i.e., $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, each $\boldsymbol{v}_i$ is M dimensional

Kernel PCA

- Goal: Compute eigenvectors $\boldsymbol{v}_i$, i.e., $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, each $\boldsymbol{v}_i$ is M dimensional
- Plugging in the expression for $\mathbf{C}$, we have

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$$

Kernel PCA

- Goal: Compute eigenvectors $\boldsymbol{v}_i$, i.e., $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, each $\boldsymbol{v}_i$ is M dimensional
- Plugging in the expression for $\mathbf{C}$, we have

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$$

- Denoting $\alpha_{in} = \frac{1}{\lambda_i N}\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i$, $\boldsymbol{v}_i = \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_n)$ (also recall Rep. Thm.)

Kernel PCA

- Goal: Compute eigenvectors $\boldsymbol{v}_i$, i.e., $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, each $\boldsymbol{v}_i$ is M dimensional
- Plugging in the expression for $\mathbf{C}$, we have

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$$

- Denoting $\alpha_{in} = \frac{1}{\lambda_i N}\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i$, $\boldsymbol{v}_i = \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_n)$ (also recall Rep. Thm.)
- Thus we can get $\boldsymbol{v}_i$ by finding $\boldsymbol{\alpha}_i = [\alpha_{i1} \ \ldots \ \alpha_{iN}]$

Kernel PCA

- Goal: Compute eigenvectors $\boldsymbol{v}_i$, i.e., $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, each $\boldsymbol{v}_i$ is M dimensional
- Plugging in the expression for $\mathbf{C}$, we have

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$$

- Denoting $\alpha_{in} = \frac{1}{\lambda_i N}\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i$, $\boldsymbol{v}_i = \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_n)$ (also recall Rep. Thm.)
- Thus we can get $\boldsymbol{v}_i$ by finding $\boldsymbol{\alpha}_i = [\alpha_{i1} \ \ldots \ \alpha_{iN}]$
- Plugging this back in the eigenvector equation $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \sum_{m=1}^{N}\alpha_{im}\phi(\boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_n)$$

Kernel PCA

- Goal: Compute eigenvectors $\boldsymbol{v}_i$, i.e., $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$, each $\boldsymbol{v}_i$ is M dimensional
- Plugging in the expression for $\mathbf{C}$, we have

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$$

- Denoting $\alpha_{in} = \frac{1}{\lambda_i N}\phi(\boldsymbol{x}_n)^\top \boldsymbol{v}_i$, $\boldsymbol{v}_i = \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_n)$ (also recall Rep. Thm.)
- Thus we can get $\boldsymbol{v}_i$ by finding $\boldsymbol{\alpha}_i = [\alpha_{i1} \ \ldots \ \alpha_{iN}]$
- Plugging this back in the eigenvector equation $\mathbf{C}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_n)\phi(\boldsymbol{x}_n)^\top \sum_{m=1}^{N}\alpha_{im}\phi(\boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_n)$$

- Pre-multiplying both sides by $\phi(\boldsymbol{x}_\ell)^\top$ and re-arranging

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\boldsymbol{x}_\ell)^\top\phi(\boldsymbol{x}_n)\sum_{m=1}^{N}\alpha_{im}\phi(\boldsymbol{x}_n)^\top\phi(\boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N}\alpha_{in}\phi(\boldsymbol{x}_\ell)^\top\phi(\boldsymbol{x}_n)$$

Kernel PCA

- Using $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, we get

$$\frac{1}{N} \sum_{n=1}^{N} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n) \sum_{m=1}^{N} \alpha_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N} \alpha_{in} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)$$

Kernel PCA

- Using $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, we get

$$\frac{1}{N}\sum_{n=1}^{N} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n) \sum_{m=1}^{N} \alpha_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N} \alpha_{in} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)$$

- Define $\boldsymbol{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$

Kernel PCA

- Using $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, we get

$$\frac{1}{N} \sum_{n=1}^{N} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n) \sum_{m=1}^{N} \alpha_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N} \alpha_{in} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)$$

- Define $\boldsymbol{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$
  - $\boldsymbol{K}$ is the similarity of two examples $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ in the ϕ space

Kernel PCA

- Using $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, we get

$$\frac{1}{N} \sum_{n=1}^{N} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n) \sum_{m=1}^{N} \alpha_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N} \alpha_{in} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)$$

- Define $\boldsymbol{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$
  - $\boldsymbol{K}$ is the similarity of two examples $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ in the ϕ space
  - ϕ is implicitly defined by kernel function k (which can be, e.g., RBF kernel)

Kernel PCA

- Using $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, we get

$$\frac{1}{N}\sum_{n=1}^{N} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n) \sum_{m=1}^{N} \alpha_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N} \alpha_{in} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)$$

- Define $\boldsymbol{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$
  - $\boldsymbol{K}$ is the similarity of two examples $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ in the ϕ space
  - ϕ is implicitly defined by kernel function k (which can be, e.g., RBF kernel)
- Define $\boldsymbol{\alpha}_i$ as the $N \times 1$ vector with elements α_{in}

Kernel PCA

- Using $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, we get

$$\frac{1}{N}\sum_{n=1}^{N} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n) \sum_{m=1}^{N} \alpha_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N} \alpha_{in} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)$$

- Define $\boldsymbol{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$
  - $\boldsymbol{K}$ is the similarity of two examples $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ in the ϕ space
  - ϕ is implicitly defined by kernel function k (which can be, e.g., RBF kernel)
- Define $\boldsymbol{\alpha}_i$ as the $N \times 1$ vector with elements α_{in}
- Using $\boldsymbol{K}$ and $\boldsymbol{\alpha}_i$, the eigenvector equation becomes:

$$\boldsymbol{K}^2\boldsymbol{\alpha}_i = \lambda_i N \boldsymbol{K}\boldsymbol{\alpha}_i \quad \Rightarrow \quad \boxed{\boldsymbol{K}\boldsymbol{\alpha}_i = \lambda_i N \boldsymbol{\alpha}_i}$$

Kernel PCA

- Using $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, we get

$$\frac{1}{N}\sum_{n=1}^{N} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n) \sum_{m=1}^{N} \alpha_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N} \alpha_{in} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)$$

- Define $\boldsymbol{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$
  - $\boldsymbol{K}$ is the similarity of two examples $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ in the ϕ space
  - ϕ is implicitly defined by kernel function k (which can be, e.g., RBF kernel)
- Define $\boldsymbol{\alpha}_i$ as the $N \times 1$ vector with elements α_{in}
- Using $\boldsymbol{K}$ and $\boldsymbol{\alpha}_i$, the eigenvector equation becomes:

$$\boldsymbol{K}^2 \boldsymbol{\alpha}_i = \lambda_i N \boldsymbol{K} \boldsymbol{\alpha}_i \quad \Rightarrow \quad \boxed{\boldsymbol{K}\boldsymbol{\alpha}_i = \lambda_i N \boldsymbol{\alpha}_i}$$

- Thus $\boldsymbol{\alpha}_i$ is an eigenvector of the $N \times N$ kernel matrix $\mathbf{K}$

Kernel PCA

- Using $\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$, we get

$$\frac{1}{N}\sum_{n=1}^{N} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n) \sum_{m=1}^{N} \alpha_{im} k(\boldsymbol{x}_n, \boldsymbol{x}_m) = \lambda_i \sum_{n=1}^{N} \alpha_{in} k(\boldsymbol{x}_\ell, \boldsymbol{x}_n)$$

- Define $\boldsymbol{K}$ as the $N \times N$ **kernel matrix** with $K_{nm} = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$
  - $\boldsymbol{K}$ is the similarity of two examples $\boldsymbol{x}_n$ and $\boldsymbol{x}_m$ in the ϕ space
  - ϕ is implicitly defined by kernel function k (which can be, e.g., RBF kernel)
- Define $\boldsymbol{\alpha}_i$ as the $N \times 1$ vector with elements α_{in}
- Using $\boldsymbol{K}$ and $\boldsymbol{\alpha}_i$, the eigenvector equation becomes:

$$\boldsymbol{K}^2\boldsymbol{\alpha}_i = \lambda_i N \boldsymbol{K}\boldsymbol{\alpha}_i \quad \Rightarrow \quad \boxed{\boldsymbol{K}\boldsymbol{\alpha}_i = \lambda_i N \boldsymbol{\alpha}_i}$$

- Thus $\boldsymbol{\alpha}_i$ is an eigenvector of the $N \times N$ kernel matrix $\mathbf{K}$
- Note: Since $\boldsymbol{v}_i^\top \boldsymbol{v}_i = 1$ and $\boldsymbol{v}_i = \sum_{n=1}^{N} \alpha_{in}\phi(\boldsymbol{x}_n)$, we have $\boldsymbol{\alpha}_i^\top \mathbf{K}\boldsymbol{\alpha}_i = 1$, which means $\boldsymbol{\alpha}_i^\top \lambda_i N \boldsymbol{\alpha}_i = 1$ and $\boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_i = 1/(\lambda_i N)$. Thus the original solution with $\boldsymbol{\alpha}_i^\top \boldsymbol{\alpha}_i = 1$ (eigenvec with norm 1) needs to be re-scaled as $\tilde{\alpha}_{in} = \alpha_{in}/\sqrt{\lambda_i N}$

Kernel PCA: Centering the Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\boldsymbol{x}_n) = \phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)$$

Kernel PCA: Centering the Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\boldsymbol{x}_n) = \phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)$$

- Each element of the centered kernel matrix

Kernel PCA: Centering the Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\boldsymbol{x}_n) = \phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)$$

- Each element of the centered kernel matrix

$$\tilde{K}_{nm} = \tilde{\phi}(\boldsymbol{x}_n)^\top \tilde{\phi}(\boldsymbol{x}_m)$$

Kernel PCA: Centering the Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\boldsymbol{x}_n) = \phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)$$

- Each element of the centered kernel matrix

$$\begin{aligned}\tilde{K}_{nm} &= \tilde{\phi}(\boldsymbol{x}_n)^\top \tilde{\phi}(\boldsymbol{x}_m) \\ &= \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)^\top \phi(\boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_j)^\top \phi(\boldsymbol{x}_\ell)\end{aligned}$$

Kernel PCA: Centering the Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\boldsymbol{x}_n) = \phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)$$

- Each element of the centered kernel matrix

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\phi}(\boldsymbol{x}_n)^\top \tilde{\phi}(\boldsymbol{x}_m) \\
&= \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)^\top \phi(\boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_j)^\top \phi(\boldsymbol{x}_\ell) \\
&= k(\boldsymbol{x}_n, \boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_n, \boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_\ell)
\end{aligned}$$

Kernel PCA: Centering the Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\boldsymbol{x}_n) = \phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)$$

- Each element of the centered kernel matrix

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\phi}(\boldsymbol{x}_n)^\top \tilde{\phi}(\boldsymbol{x}_m) \\
&= \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)^\top \phi(\boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_j)^\top \phi(\boldsymbol{x}_\ell) \\
&= k(\boldsymbol{x}_n, \boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_n, \boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_\ell)
\end{aligned}$$

- In matrix notation, the centered $\tilde{\boldsymbol{K}} = \boldsymbol{K} - \boldsymbol{1}_N\boldsymbol{K} - \boldsymbol{K}\boldsymbol{1}_N + \boldsymbol{1}_N\boldsymbol{K}\boldsymbol{1}_N$

Kernel PCA: Centering the Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\boldsymbol{x}_n) = \phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)$$

- Each element of the centered kernel matrix

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\phi}(\boldsymbol{x}_n)^\top \tilde{\phi}(\boldsymbol{x}_m) \\
&= \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)^\top \phi(\boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_j)^\top \phi(\boldsymbol{x}_\ell) \\
&= k(\boldsymbol{x}_n, \boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_n, \boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_\ell)
\end{aligned}$$

- In matrix notation, the centered $\tilde{\boldsymbol{K}} = \boldsymbol{K} - \boldsymbol{1}_N\boldsymbol{K} - \boldsymbol{K}\boldsymbol{1}_N + \boldsymbol{1}_N\boldsymbol{K}\boldsymbol{1}_N$
- $\boldsymbol{1}_N$ is the $N \times N$ matrix with every element $= 1/N$

Kernel PCA: Centering the Data

- In PCA, we centered the data before computing the covariance matrix
- For kernel PCA, we need to do the same

$$\tilde{\phi}(\boldsymbol{x}_n) = \phi(\boldsymbol{x}_n) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)$$

- Each element of the centered kernel matrix

$$\begin{aligned}
\tilde{K}_{nm} &= \tilde{\phi}(\boldsymbol{x}_n)^\top \tilde{\phi}(\boldsymbol{x}_m) \\
&= \phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_n)^\top \phi(\boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_\ell)^\top \phi(\boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}\phi(\boldsymbol{x}_j)^\top \phi(\boldsymbol{x}_\ell) \\
&= k(\boldsymbol{x}_n, \boldsymbol{x}_m) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_n, \boldsymbol{x}_\ell) - \frac{1}{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{\ell=1}^{N}k(\boldsymbol{x}_\ell, \boldsymbol{x}_\ell)
\end{aligned}$$

- In matrix notation, the centered $\tilde{\boldsymbol{K}} = \boldsymbol{K} - \boldsymbol{1}_N\boldsymbol{K} - \boldsymbol{K}\boldsymbol{1}_N + \boldsymbol{1}_N\boldsymbol{K}\boldsymbol{1}_N$
- $\boldsymbol{1}_N$ is the $N \times N$ matrix with every element $= 1/N$
- Eigen-decomposition is then done for the centered kernel matrix $\tilde{\boldsymbol{K}}$

Kernel PCA: Computing the Embeddings

- Suppose $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_L\}$ are the top L eigenvectors of kernel matrix $\tilde{\boldsymbol{K}}$
- The L-dimensional KPCA projection $\boldsymbol{z}_m = [z_{m1}, \ldots, z_{mL}]$ of a point $\boldsymbol{x}_m$:

$$z_{m\ell} = \phi(\boldsymbol{x}_m)^\top \boldsymbol{v}_\ell \qquad \forall \ell = 1, \ldots, L$$

Kernel PCA: Computing the Embeddings

- Suppose $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_L\}$ are the top L eigenvectors of kernel matrix $\tilde{\boldsymbol{K}}$
- The L-dimensional KPCA projection $\boldsymbol{z}_m = [z_{m1}, \ldots, z_{mL}]$ of a point $\boldsymbol{x}_m$:

$$z_{m\ell} = \phi(\boldsymbol{x}_m)^\top \boldsymbol{v}_\ell \qquad \forall \ell = 1, \ldots, L$$

- Using the definition of $\boldsymbol{v}_\ell$, i.e., $\boldsymbol{v}_\ell = \sum_{n=1}^N \alpha_{\ell n} \phi(\boldsymbol{x}_n)$, we have

$$z_{m\ell} = \phi(\boldsymbol{x}_m)^\top \boldsymbol{v}_\ell = \sum_{n=1}^N \alpha_{\ell n} k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

Kernel PCA: Computing the Embeddings

- Suppose $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_L\}$ are the top L eigenvectors of kernel matrix $\tilde{\boldsymbol{K}}$
- The L-dimensional KPCA projection $\boldsymbol{z}_m = [z_{m1}, \ldots, z_{mL}]$ of a point $\boldsymbol{x}_m$:

$$z_{m\ell} = \phi(\boldsymbol{x}_m)^\top \boldsymbol{v}_\ell \qquad \forall \ell = 1, \ldots, L$$

- Using the definition of $\boldsymbol{v}_\ell$, i.e., $\boldsymbol{v}_\ell = \sum_{n=1}^{N} \alpha_{\ell n} \phi(\boldsymbol{x}_n)$, we have

$$z_{m\ell} = \phi(\boldsymbol{x}_m)^\top \boldsymbol{v}_\ell = \sum_{n=1}^{N} \alpha_{\ell n} k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

- Note: Cost of computing the embeddings scales in N

Kernel PCA: Computing the Embeddings

- Suppose $\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_L\}$ are the top L eigenvectors of kernel matrix $\tilde{\boldsymbol{K}}$
- The L-dimensional KPCA projection $\boldsymbol{z}_m = [z_{m1}, \ldots, z_{mL}]$ of a point $\boldsymbol{x}_m$:

$$z_{m\ell} = \phi(\boldsymbol{x}_m)^\top \boldsymbol{v}_\ell \qquad \forall \ell = 1, \ldots, L$$

- Using the definition of $\boldsymbol{v}_\ell$, i.e., $\boldsymbol{v}_\ell = \sum_{n=1}^{N} \alpha_{\ell n} \phi(\boldsymbol{x}_n)$, we have

$$z_{m\ell} = \phi(\boldsymbol{x}_m)^\top \boldsymbol{v}_\ell = \sum_{n=1}^{N} \alpha_{\ell n} k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

- Note: Cost of computing the embeddings scales in N
- Note: For linear kernel, KPCA reduces to PCA (but more efficient if $N < D$)

Kernel PCA: Summary of the algorithm

- Construct the $N \times N$ kernel matrix $\mathbf{K}$

Kernel PCA: Summary of the algorithm

- Construct the $N \times N$ kernel matrix $\mathbf{K}$
- Center $\mathbf{K}$ as follows $\tilde{\boldsymbol{K}} = \boldsymbol{K} - \mathbf{1}_N \boldsymbol{K} - \boldsymbol{K}\mathbf{1}_N + \mathbf{1}_N \boldsymbol{K} \mathbf{1}_N$, where $\mathbf{1}_N$ is an $N \times N$ matrix of all $1/N$

Kernel PCA: Summary of the algorithm

- Construct the $N \times N$ kernel matrix $\mathbf{K}$
- Center $\mathbf{K}$ as follows $\tilde{\boldsymbol{K}} = \boldsymbol{K} - \mathbf{1}_N \boldsymbol{K} - \boldsymbol{K}\mathbf{1}_N + \mathbf{1}_N \boldsymbol{K} \mathbf{1}_N$, where $\mathbf{1}_N$ is an $N \times N$ matrix of all $1/N$
- Do eigen-decomposition of $\tilde{\boldsymbol{K}}$ and find top L eigenvecs $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_L$ with eigenvals $\lambda_1, \lambda_2, \ldots, \lambda_L$

Kernel PCA: Summary of the algorithm

- Construct the $N \times N$ kernel matrix $\mathbf{K}$
- Center $\mathbf{K}$ as follows $\tilde{\boldsymbol{K}} = \boldsymbol{K} - \mathbf{1}_N \boldsymbol{K} - \boldsymbol{K}\mathbf{1}_N + \mathbf{1}_N \boldsymbol{K} \mathbf{1}_N$, where $\mathbf{1}_N$ is an $N \times N$ matrix of all $1/N$
- Do eigen-decomposition of $\tilde{\boldsymbol{K}}$ and find top L eigenvecs $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_L$ with eigenvals $\lambda_1, \lambda_2, \ldots, \lambda_L$
- Re-scale each eigenvector as $\tilde{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_i / \sqrt{\lambda_i N}$, $\forall i = 1, \ldots, L$

Kernel PCA: Summary of the algorithm

- Construct the $N \times N$ kernel matrix $\mathbf{K}$
- Center $\mathbf{K}$ as follows $\tilde{\boldsymbol{K}} = \boldsymbol{K} - \mathbf{1}_N \boldsymbol{K} - \boldsymbol{K}\mathbf{1}_N + \mathbf{1}_N \boldsymbol{K} \mathbf{1}_N$, where $\mathbf{1}_N$ is an $N \times N$ matrix of all $1/N$
- Do eigen-decomposition of $\tilde{\boldsymbol{K}}$ and find top L eigenvecs $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_L$ with eigenvals $\lambda_1, \lambda_2, \ldots, \lambda_L$
- Re-scale each eigenvector as $\tilde{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_i / \sqrt{\lambda_i N}$, $\forall i = 1, \ldots, L$
- Finally, compute embedding $\boldsymbol{z}_m \in \mathbb{R}^L$ of any point $\boldsymbol{x}_m$ as

$$z_{m\ell} = \sum_{n=1}^{N} \tilde{\alpha}_{\ell n} k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

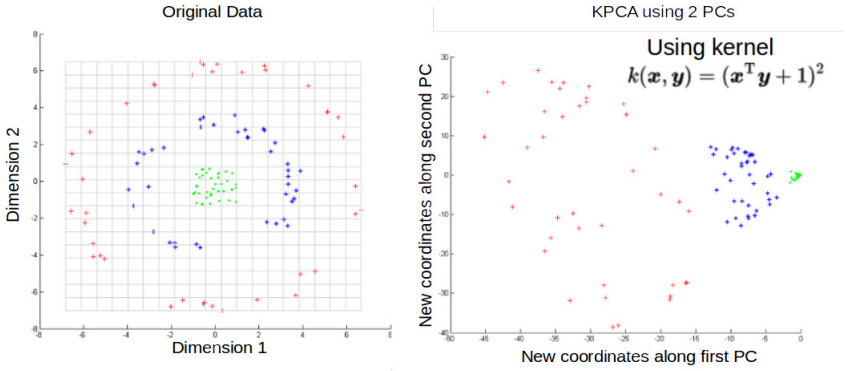
Kernel PCA: Summary of the algorithm

- Construct the $N \times N$ kernel matrix $\mathbf{K}$
- Center $\mathbf{K}$ as follows $\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_N\mathbf{K} - \mathbf{K}\mathbf{1}_N + \mathbf{1}_N\mathbf{K}\mathbf{1}_N$, where $\mathbf{1}_N$ is an $N \times N$ matrix of all $1/N$
- Do eigen-decomposition of $\tilde{\mathbf{K}}$ and find top L eigenvecs $\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_L$ with eigenvals $\lambda_1, \lambda_2, \ldots, \lambda_L$
- Re-scale each eigenvector as $\tilde{\boldsymbol{\alpha}}_i = \boldsymbol{\alpha}_i / \sqrt{\lambda_i N}$, $\forall i = 1, \ldots, L$
- Finally, compute embedding $\boldsymbol{z}_m \in \mathbb{R}^L$ of any point $\boldsymbol{x}_m$ as

$$z_{m\ell} = \sum_{n=1}^{N} \tilde{\alpha}_{\ell n} k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

- Note: For compactness, the $L \times N$ matrix of all L eigenvectors (each is N dimensional) can be written as $\tilde{\boldsymbol{\alpha}} = [\tilde{\boldsymbol{\alpha}}_1 \ \tilde{\boldsymbol{\alpha}}_2 \ \ldots \ \tilde{\boldsymbol{\alpha}}_L]^\top$. Thus we can also write embedding $\boldsymbol{z}_m \in \mathbb{R}^L$ as $\boldsymbol{z}_m = \tilde{\boldsymbol{\alpha}}\boldsymbol{k}_m$ where $\boldsymbol{k}_m$ is $N \times 1$ vector of kernelized similarities of $\boldsymbol{x}_m$ with all the N training data points

Kernel PCA: An Example



Note that even if we throw away the 2nd PC, we get a good 1-D embedding