COMPSCI 589
Lecture 21: Kernel Principal Components Analysis and Spectral Clustering

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

Slides by Benjamin M. Marlin (marlin@cs.umass.edu).
Created with support from National Science Foundation Award# IIS-1350522.

Outline

Limitations of LDR

- All the dimensionality reduction methods we've seen so far find optimal linear sub-spaces under different constraints.

Limitations of LDR

- All the dimensionality reduction methods we've seen so far find optimal linear sub-spaces under different constraints.
- **Question:** How can we move beyond linear sub-spaces?

Basis Expansion

- One way to move beyond linear dimensionality reduction is to first apply a non-linear basis expansion to the data vectors, and then apply linear dimensionality reduction.

Basis Expansion

- One way to move beyond linear dimensionality reduction is to first apply a non-linear basis expansion to the data vectors, and then apply linear dimensionality reduction.
- Any basis expansion can be applied including polynomials, etc., just as in the case of classification and regression.

Basis Expansion + SVD

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a basis expansion function $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ for $D' > D$, we obtain the following SVD-based algorithm:

Basis Expansion + SVD

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a basis expansion function $\phi : \mathbb{R}^{D} \rightarrow \mathbb{R}^{D'}$ for $D' > D$, we obtain the following SVD-based algorithm:

1. Compute $\mathbf{U}, \mathbf{S}, \mathbf{V} = \text{SVD}(\phi(\mathbf{X}))$

Basis Expansion + SVD

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a basis expansion function $\phi : \mathbb{R}^{D} \rightarrow \mathbb{R}^{D'}$ for $D' > D$, we obtain the following SVD-based algorithm:

1. Compute $\mathbf{U}, \mathbf{S}, \mathbf{V} = \text{SVD}(\phi(\mathbf{X}))$
2. Return $Z = \mathbf{US}$

Basis Expansion + PCA

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a basis expansion function $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ for $D' > D$, we obtain the following PCA-based algorithm:

Basis Expansion + PCA

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a basis expansion function $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ for $D' > D$, we obtain the following PCA-based algorithm:

1. Compute $\Sigma = (\phi(\mathbf{X}) - \mu)^T(\phi(\mathbf{X}) - \mu)$ where $\mu = 1/N \sum \phi(\mathbf{X}_i)$.

Basis Expansion + PCA

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a basis expansion function $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ for $D' > D$, we obtain the following PCA-based algorithm:

1. Compute $\Sigma = (\phi(\mathbf{X}) - \mu)^T(\phi(\mathbf{X}) - \mu)$ where $\mu = 1/N \sum \phi(\mathbf{X}_i)$.
2. Compute the K leading eigenvectors $w_1, ..., w_K$ of Σ where $\mathbf{w}_k \in \mathbb{R}^{D'}$.

Basis Expansion + PCA

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a basis expansion function $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ for $D' > D$, we obtain the following PCA-based algorithm:

1. Compute $\Sigma = (\phi(\mathbf{X}) - \mu)^T(\phi(\mathbf{X}) - \mu)$ where $\mu = 1/N \sum \phi(\mathbf{X}_i)$.
2. Compute the K leading eigenvectors $w_1, ..., w_K$ of Σ where $\mathbf{w}_k \in \mathbb{R}^{D'}$.
3. Stack the eigenvectors together into a $D' \times K$ matrix $\mathbf{W}$ where each column k of $\mathbf{W}$ corresponds to $\mathbf{w}_k$.

Basis Expansion + PCA

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a basis expansion function $\phi : \mathbb{R}^D \rightarrow \mathbb{R}^{D'}$ for $D' > D$, we obtain the following PCA-based algorithm:

1. Compute $\Sigma = (\phi(\mathbf{X}) - \mu)^T(\phi(\mathbf{X}) - \mu)$ where $\mu = 1/N \sum \phi(\mathbf{X}_i)$.
2. Compute the K leading eigenvectors $w_1, ..., w_K$ of Σ where $\mathbf{w}_k \in \mathbb{R}^{D'}$.
3. Stack the eigenvectors together into a $D' \times K$ matrix $\mathbf{W}$ where each column k of $\mathbf{W}$ corresponds to $\mathbf{w}_k$.
4. Project the matrix $\phi(\mathbf{X})$ into the rank-K sub-space of maximum variance by computing the matrix product $\mathbf{Z} = \phi(\mathbf{X})\mathbf{W}$.

Kernel PCA

- As in the classification case, it becomes very expensive to use an explicit basis function expansion that maps data into a high-dimensional space.

Kernel PCA

- As in the classification case, it becomes very expensive to use an explicit basis function expansion that maps data into a high-dimensional space.
- In the basic SVD-based algorithm, there's no way to avoid this problem.

Kernel PCA

- As in the classification case, it becomes very expensive to use an explicit basis function expansion that maps data into a high-dimensional space.
- In the basic SVD-based algorithm, there's no way to avoid this problem.
- In the PCA-based algorithm, it's not hard to show that the sample covariance matrix obtained after the basis expansion depends only on inner products of the form $\langle \phi(\mathbf{X}_i), \phi(\mathbf{X}_j) \rangle$.

Kernel PCA

- As in the classification case, it becomes very expensive to use an explicit basis function expansion that maps data into a high-dimensional space.
- In the basic SVD-based algorithm, there's no way to avoid this problem.
- In the PCA-based algorithm, it's not hard to show that the sample covariance matrix obtained after the basis expansion depends only on inner products of the form $\langle \phi(\mathbf{X}_i), \phi(\mathbf{X}_j) \rangle$.
- Kernel functions can often provide a much more efficient computation of these inner products without explicitly computing the basis expansion: $\mathcal{K}(\mathbf{X}_i, \mathbf{X}_j) = \langle \phi(\mathbf{X}_i), \phi(\mathbf{X}_j) \rangle$

Kernel PCA Algorithm

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a kernel function $\mathcal{K}$, Kernel PCA can be computed as follows:

Kernel PCA Algorithm

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a kernel function $\mathcal{K}$, Kernel PCA can be computed as follows:

1. Compute $\mathbf{K}_{ij} = \mathcal{K}(\mathbf{X}_i, \mathbf{X}_j)$ for all i, j

Kernel PCA Algorithm

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a kernel function $\mathcal{K}$, Kernel PCA can be computed as follows:

1. Compute $\mathbf{K}_{ij} = \mathcal{K}(\mathbf{X}_i, \mathbf{X}_j)$ for all i, j
2. Compute $\Sigma = (I - 1_N)\mathbf{K}(I - 1_N)$ where 1_N is an NxN matrix where every entry is $1/N$.

Kernel PCA Algorithm

Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a kernel function $\mathcal{K}$, Kernel PCA can be computed as follows:

1. Compute $\mathbf{K}_{ij} = \mathcal{K}(\mathbf{X}_i, \mathbf{X}_j)$ for all i, j
2. Compute $\Sigma = (I - 1_N)\mathbf{K}(I - 1_N)$ where 1_N is an NxN matrix where every entry is $1/N$.
3. Compute the K leading eigenvectors $w_1, ..., w_K$ of Σ where $\mathbf{w}_k \in \mathbb{R}^N$ along with their eigenvalues $\lambda_1, ..., \lambda_K$

Kernel PCA Algorithm

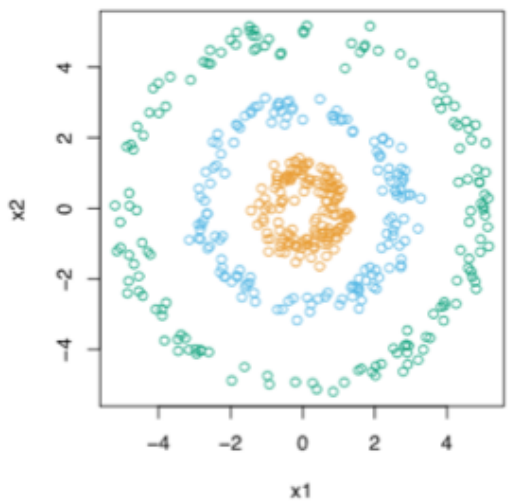
Given a data set $\mathbf{X} \in \mathbb{R}^{N \times D}$ and a kernel function $\mathcal{K}$, Kernel PCA can be computed as follows:

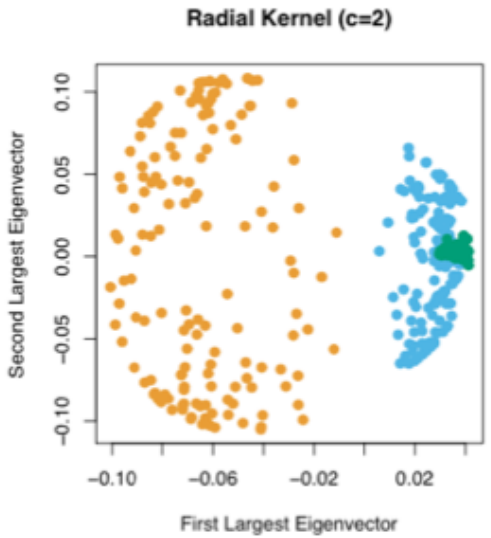
1. Compute $\mathbf{K}_{ij} = \mathcal{K}(\mathbf{X}_i, \mathbf{X}_j)$ for all i, j
2. Compute $\Sigma = (I - 1_N)\mathbf{K}(I - 1_N)$ where 1_N is an NxN matrix where every entry is $1/N$.
3. Compute the K leading eigenvectors $w_1, ..., w_K$ of Σ where $\mathbf{w}_k \in \mathbb{R}^N$ along with their eigenvalues $\lambda_1, ..., \lambda_K$
4. Compute the elements of the matrix of projected data vectors using

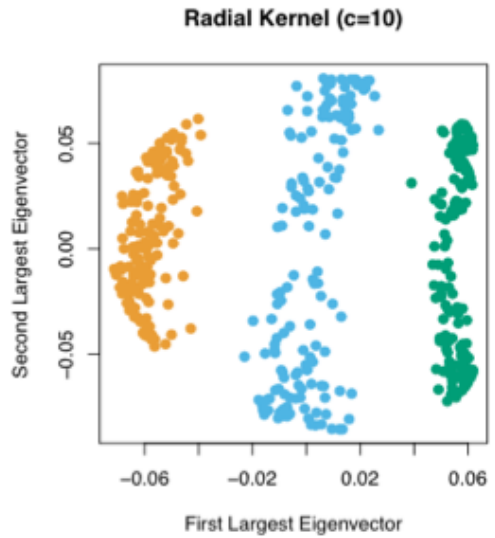
$$\mathbf{Z}_{nk} = \sum_{i=1}^{N} \frac{\mathbf{w}_{ik}}{\lambda_k} \mathcal{K}(\mathbf{X}_n, \mathbf{X}_i)$$

Example

Consider the case of using the radial basis function kernel $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \exp(-\frac{1}{c}||\mathbf{x} - \mathbf{y}||_2^2)$.







Summary

- Kernel PCA provides a non-linear dimensionality reduction method that can be used to extract directions of variation after applying high-dimensional basis function expansions, without explicitly performing the basis expansion.

Summary

- Kernel PCA provides a non-linear dimensionality reduction method that can be used to extract directions of variation after applying high-dimensional basis function expansions, without explicitly performing the basis expansion.
- Kernel PCA can be thought of as looking for directions of variation in the space of similarities between data cases, and can extract components that correspond to fairly complex structures.

Summary

- Kernel PCA provides a non-linear dimensionality reduction method that can be used to extract directions of variation after applying high-dimensional basis function expansions, without explicitly performing the basis expansion.
- Kernel PCA can be thought of as looking for directions of variation in the space of similarities between data cases, and can extract components that correspond to fairly complex structures.
- When a good kernel is known a priori, kernel PCA can provide an effective pre-processing step for clustering methods as well as linear classification and regression methods.

Summary

- Kernel PCA provides a non-linear dimensionality reduction method that can be used to extract directions of variation after applying high-dimensional basis function expansions, without explicitly performing the basis expansion.
- Kernel PCA can be thought of as looking for directions of variation in the space of similarities between data cases, and can extract components that correspond to fairly complex structures.
- When a good kernel is known a priori, kernel PCA can provide an effective pre-processing step for clustering methods as well as linear classification and regression methods.
- However, exact computation of kernel PCA can be expensive because the size of the matrix that is decomposed is NxN.

Outline

Spectra Clustering

- Applying kernel PCA followed by K-means clustering is closely related to a clustering method called spectral clustering.

Spectra Clustering

- Applying kernel PCA followed by K-means clustering is closely related to a clustering method called spectral clustering.
- Spectral clustering works by defining a weighted similarity graph on the data cases.

Spectra Clustering

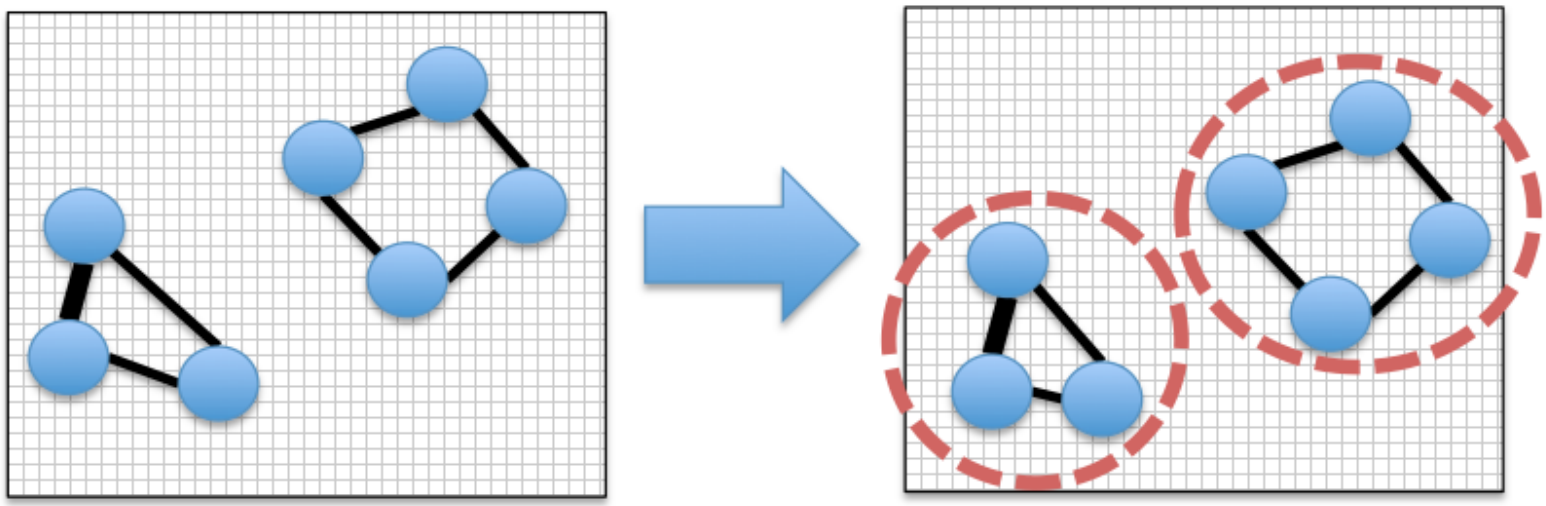
- Applying kernel PCA followed by K-means clustering is closely related to a clustering method called spectral clustering.
- Spectral clustering works by defining a weighted similarity graph on the data cases.
- It's very common to use $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \exp(-\frac{1}{c}||\mathbf{x} - \mathbf{y}||_2^2)$ as the weighting function and to include edges corresponding to the K nearest neighbors of each point.

Spectra Clustering

- Applying kernel PCA followed by K-means clustering is closely related to a clustering method called spectral clustering.
- Spectral clustering works by defining a weighted similarity graph on the data cases.
- It's very common to use $\mathcal{K}(\mathbf{x}, \mathbf{y}) = \exp(-\frac{1}{c}||\mathbf{x} - \mathbf{y}||_2^2)$ as the weighting function and to include edges corresponding to the K nearest neighbors of each point.
- Unlike the kernel PCA case, the similarity values for edges not included in the graph are set to 0. The matrix of edge weights is denoted by $\mathbf{W}$.

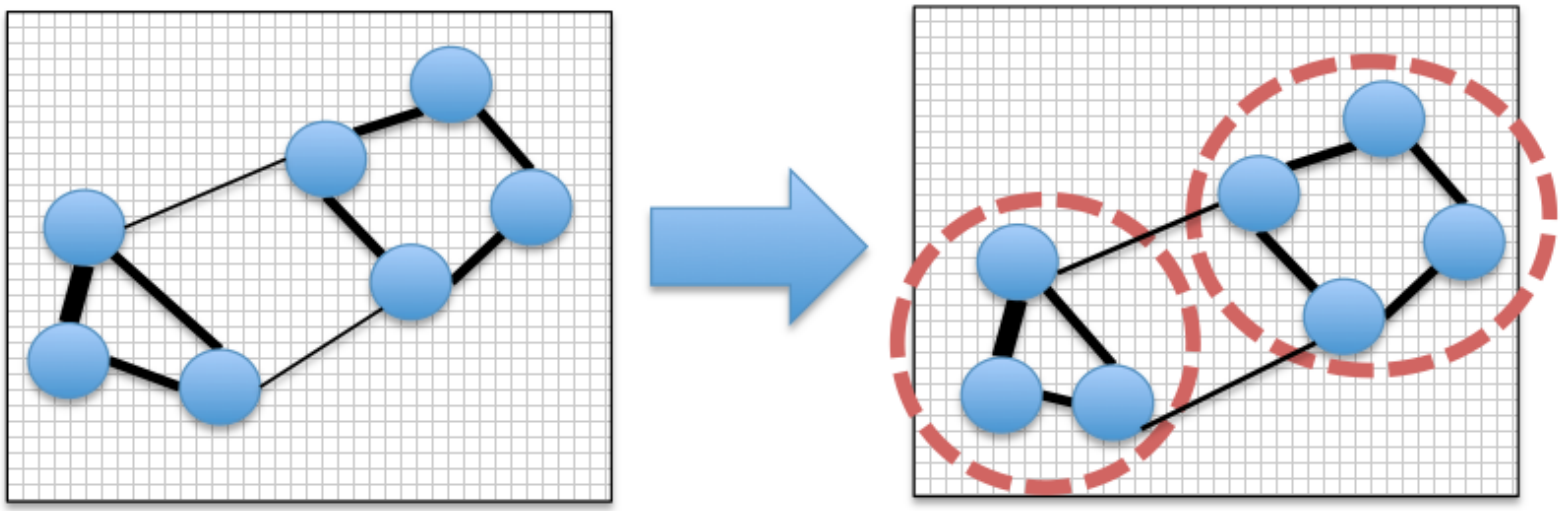
Connected Components

- In the simple case, the resulting graph has multiple connected components, and we can create one cluster for each connected component.



Minimum Weight Balanced Cut

- In the more complex case, the graph has only one connected component, and the goal is to identify a minimum weight balanced cut:



Spectral Clustering Algorithm

- Define the weighted node degree matrix $\mathbf{D}$ such that $\mathbf{D}_{ii} = \sum_n \mathbf{W}_{in}$.

Spectral Clustering Algorithm

- Define the weighted node degree matrix $\mathbf{D}$ such that $\mathbf{D}_{ii} = \sum_n \mathbf{W}_{in}$.
- Define the graph laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$.

Spectral Clustering Algorithm

- Define the weighted node degree matrix $\mathbf{D}$ such that $\mathbf{D}_{ii} = \sum_n \mathbf{W}_{in}$.
- Define the graph laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$.
- Compute the eigendecomposition of $\mathbf{L}$ and extract the M eigenvectors corresponding to the M smallest eigenvalues.

Spectral Clustering Algorithm

- Define the weighted node degree matrix $\mathbf{D}$ such that $\mathbf{D}_{ii} = \sum_n \mathbf{W}_{in}$.
- Define the graph laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$.
- Compute the eigendecomposition of $\mathbf{L}$ and extract the M eigenvectors corresponding to the M smallest eigenvalues.
- Just like in kernel PCA, each eigenvector is length N. The collection of M of them forms an alternative dimensionality reduced representation for $\mathbf{X}$, which we can call $\mathbf{Z}$.

Spectral Clustering Algorithm

- Define the weighted node degree matrix $\mathbf{D}$ such that $\mathbf{D}_{ii} = \sum_n \mathbf{W}_{in}$.
- Define the graph laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$.
- Compute the eigendecomposition of $\mathbf{L}$ and extract the M eigenvectors corresponding to the M smallest eigenvalues.
- Just like in kernel PCA, each eigenvector is length N. The collection of M of them forms an alternative dimensionality reduced representation for $\mathbf{X}$, which we can call $\mathbf{Z}$.
- Run K-means on $\mathbf{Z}$ to extract K clusters.

Spectral Clustering Algorithm

- Define the weighted node degree matrix $\mathbf{D}$ such that $\mathbf{D}_{ii} = \sum_n \mathbf{W}_{in}$.
- Define the graph laplacian $\mathbf{L} = \mathbf{D} - \mathbf{W}$.
- Compute the eigendecomposition of $\mathbf{L}$ and extract the M eigenvectors corresponding to the M smallest eigenvalues.
- Just like in kernel PCA, each eigenvector is length N. The collection of M of them forms an alternative dimensionality reduced representation for $\mathbf{X}$, which we can call $\mathbf{Z}$.
- Run K-means on $\mathbf{Z}$ to extract K clusters.
- This algorithm is an approximation to the exact K-way min-cut in the graph, which has exponential complexity to compute.