CSE517A Machine Learning **Spring 2020**

Lecture 13: Dimensionality Reduction

Instructor: Marion Neumann *Scribe: Jingyu Xin*

Reading: FCML 7.1 , 7.2 (PCA); LFD eCH 9.1, 9.2 (PCA, SVD)

1 Introduction

Goal: represent data points $\mathbf{x}_i \in \mathbb{R}^d$ in D in a lower dimensional (sub)space $\mathbb{R}^r$ $(r < d)$

Notation: $X = [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{d \times n}$ with $\mathbf{x}_i \in \mathbb{R}^d$

Why?:

- data compression
- data denoising (similar to quantization)
- data decorrelation
- visualization

Main Techniques:

- linear: PCA/SVD/MDS (all the same!), data whitening
- non-linear: kernel PCA, GPLVM, Isomap (also referred to as *manifold learning*)

Similar to clustering (and also to most supervised learning problems) there are *heuristic* (*non-probabilistic*) methods and *probabilistic* methods for dimensionality reduction, cf. Table 1 for an overview.

	Clustering	Dimensionality Reduction
heuristic	k-means, kernel k-means	PCA, kernel PCA
probabilistic	GMM	probabilistic PCA, GPLVM

Table 1: Probabilistic and non-probabilistic models for unsupervised machine learning.

2 Principal Component Analysis (PCA)

Goal: project $\mathbf{x}_i \in \mathbb{R}^d$ to first r $(r < d)$ *principal components* (directions of maximum variance)

Task: find $U \in \mathbb{R}^{d \times r}$ such that $\{\mathbf{v}_i\}_{i=1}^n$ with $\mathbf{v}_i = U^\top \mathbf{x}_i$ have maximum spread

Two (equivalent) ways of formulating this problem:

- maximize the projected variance (we are using this one)
- minimize the squared reconstruction $(UU^\top \mathbf{x}_i)$ error

Data statistics:

- (sample) mean: $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$
- (sample) co-variance: $C = \frac{1}{n} \sum_{i=1}^n (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$

2.1 Derivation

Step 1: **Center the data:** $\mathbf{x}_i \leftarrow \mathbf{x}_i - \bar{\mathbf{x}}$ (for the rest of this lecture we assume that the $\mathbf{x}_i$ are *centered*)

Step 2: **Find subspace of maximum spread.** For simplicity let's consider a one-dimensional subspace first. To find the subspace of maximum spread you could imagine to project the data into each possible one-dimensional subspace, then measure the spread of the projected data for each subspace, and then pick the subspace with the largest spread. To project the data we can use *vector projection*: $\mathbf{u}^\top \mathbf{x}_i$ with $\mathbf{u} \in \mathbb{R}^d$ and $\mathbf{u}^\top \mathbf{u} = 1$ and to measure the spread we can compute the sum of squared projection values. So, we actually want to solve the following problem:

$$\begin{aligned}
\mathbf{u} &= \arg\max_{\mathbf{u}:\mathbf{u}^\top\mathbf{u}=1} \frac{1}{n}\sum_{i=1}^{n}(\mathbf{u}^\top \mathbf{x}_i)^2 \\
&= \arg\max_{\mathbf{u}:\mathbf{u}^\top\mathbf{u}=1} \mathbf{u}^\top \underbrace{\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^\top}_{=C}\mathbf{u} \\
&= \arg\max_{\mathbf{u}:\mathbf{u}^\top\mathbf{u}=1} \mathbf{u}^\top C\mathbf{u}
\end{aligned} \tag{1}$$

Now, we want to solve the optimization problem

$$\max_{\mathbf{u}} \mathbf{u}^\top C\mathbf{u} \quad \text{s.t. } \mathbf{u}^\top\mathbf{u} = 1 \tag{2}$$

Apply Lagrange multiplier, we have:

$$\begin{aligned}
\alpha(\mathbf{u}, \lambda) &= \mathbf{u}^\top C\mathbf{u} - \lambda(\mathbf{u}^\top\mathbf{u} - 1) \\
\frac{\partial\alpha}{\partial\mathbf{u}} &= 2C\mathbf{u} - 2\lambda\mathbf{u} \triangleq 0 \\
&\iff C\mathbf{u} = \lambda\mathbf{u}
\end{aligned} \tag{3}$$

From Eq. (3), we know that $\mathbf{u}$ is an eigenvector of C. Now,

$$\max \mathbf{u}^\top C\mathbf{u} \iff \max \mathbf{u}^\top\lambda\mathbf{u} \iff \max \lambda\underbrace{\mathbf{u}^\top\mathbf{u}}_{=1} \iff \max\lambda \tag{4}$$

So, to project $\mathbf{x}_i \in \mathbb{R}^d$ into r dimensions, pick the r eigenvectors with the largest eigenvalues (*principal components*)

$$U = [\mathbf{u}_1, \ldots, \mathbf{u}_r] \in \mathbb{R}^{d\times r} \tag{5}$$

Matrix Algebra Note: C is *symmetric* $\Rightarrow$ $\mathbf{u}_j$'s are *orthogonal* ($\mathbf{u}_j^\top\mathbf{u}_k = 0$). Since we assumed that $\mathbf{u}_j$ have *unit lengths* ($\mathbf{u}_j^\top\mathbf{u}_j = 1$) they are *orthonormal*.

2.2 PCA Algorithm and Data Projection

The PCA algorithm as stated in Algorithm 1 computes the eigenvectors of C and returns the the top r eigenvectors.

Algorithm 1 PCA

Input $\mathbf{x}_1, \ldots, \mathbf{x}_n, r$
$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$
$C = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top$
$\mathbf{u}_1, \ldots, \mathbf{u}_d \rightarrow$ EVs of C, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$
return $U = [\mathbf{u}_1, \ldots, \mathbf{u}_r] \in \mathbb{R}^{d\times r}$

To reduce the dimensionality of our data we can now project the $\mathbf{x}_i$'s into an r-dimensional subspace using the the matrix U. In the following, we assume $\bar{\mathbf{x}} = \mathbf{0}$ (our input data is centerd).
Data Projection: $\mathbf{v}_i = U^\top \mathbf{x}_i$ (for single data point) or $V = U^\top X$ (for all input points in D)

Note, that we can write the data projection as

$$\mathbf{v} = U^\top \mathbf{x} = \begin{bmatrix} \mathbf{u}_1^\top \mathbf{x} \\ \mathbf{u}_2^\top \mathbf{x} \\ \vdots \\ \mathbf{u}_r^\top \mathbf{x} \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_r \end{bmatrix} \tag{6}$$

2.3 Reconstruction Error

Reconstruction: $\hat{\mathbf{x}}_i = U\mathbf{v}_i$

Note, that we can write the reconstruction as

$$\hat{\mathbf{x}} = U\mathbf{v} = \begin{bmatrix} u_{11}v_1 + u_{12}v_2 + \cdots + u_{1r}v_r \\ u_{21}v_1 + u_{22}v_2 + \cdots + u_{2r}v_r \\ \vdots \\ u_{d1}v_1 + u_{d2}v_2 + \cdots + u_{dr}v_r \end{bmatrix} = \sum_{\alpha=1}^{r} v_\alpha \mathbf{u}_\alpha \tag{7}$$

Now, let's look at the *reconstruction error*, which is defined as $\|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$.

Let $\tilde{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_d] \in \mathbb{R}^{d \times d}$ (use all eigenvectors). Then using Eq. (7), the original $\mathbf{x}$ can be expressed as

$$\mathbf{x} = \tilde{U}\mathbf{v} = \sum_{\alpha=1}^{d} v_\alpha \mathbf{u}_\alpha. \tag{8}$$

Now, we can write the reconstruction error as

$$\begin{aligned} \|\mathbf{x} - \hat{\mathbf{x}}\|^2 &= \| \sum_{\alpha=1}^{d} v_\alpha \mathbf{u}_\alpha - \sum_{\alpha=1}^{r} v_\alpha \mathbf{u}_\alpha \|^2 \\ &= \| \sum_{\alpha=r+1}^{d} v_\alpha \mathbf{u}_\alpha \|^2 = \sum_{\alpha=r+1}^{d} v_\alpha^2, \end{aligned} \tag{9}$$

since the $\mathbf{u}_\alpha$'s are *orthonormal* ($\Rightarrow \mathbf{u}_\alpha^\top \mathbf{u}_\alpha = 1$, $\mathbf{u}_\alpha^\top \mathbf{u}_\beta = 0$). Looking at all data points $\mathbf{x}_i \in D$, we get:

$$\begin{aligned} \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 &= \sum_{\alpha=r+1}^{d} \frac{1}{n} \sum_{i=1}^{n} \underbrace{[\mathbf{v}_i]_\alpha^2}_{(\mathbf{u}_\alpha^\top \mathbf{x}_i)^2} \\ &= \sum_{\alpha=r+1}^{d} \frac{1}{n} \sum_{i=1}^{n} \mathbf{u}_\alpha^\top \mathbf{x}_i \mathbf{x}_i^\top \mathbf{u}_\alpha \\ &= \sum_{\alpha=r+1}^{d} \mathbf{u}_\alpha^\top \underbrace{C\mathbf{u}_\alpha}_{=\lambda \mathbf{u}_\alpha} \\ &= \sum_{\alpha=r+1}^{d} \lambda \underbrace{\mathbf{u}_\alpha^\top \mathbf{u}_\alpha}_{=1} \\ &= \sum_{\alpha=r+1}^{d} \lambda_\alpha \end{aligned} \tag{10}$$

So, the reconstruction error for the entire dataset D can be computed from the eigenvalues of the eigenvectors that were not considered for projection.

2.4 Remarks and Summary

How to pick r?

(1) drop in *eigenspectrum*

(2) given a reconstruction error we can select r using Eq. (10) or given a reconstruction error tolerance ε, pick r s.t. $\frac{\sum_{j=r+1}^{d} \lambda_j}{\sum_{j=1}^{d} \lambda_j} < \varepsilon$

Note:

- PCA decorrelates data: $C^{new} = \frac{1}{n} \sum \mathbf{v}_i \mathbf{v}_i^\top = \frac{1}{n} \sum U^\top \mathbf{x}_i \mathbf{x}_i^\top U = U^\top C U = U^\top \boldsymbol{\lambda} U = \boldsymbol{\lambda} I$
- PCA is optimal: no other linear method can produce projections with a smaller reconstruction error.
- PCA can be computed in $\mathcal{O}(d^2 n + d^3)$
  - $\mathcal{O}(d^2 n)$ for computing C
  - $\mathcal{O}(d^3)$ for computing the eigenvectors of C.

3 Singular Value Decomposition and Multidimensional Scaling

Singular Value Decomposition (SVD) is another (more efficient and numerically more stable) way to compute U.

Matrix notation:

$$C = \frac{1}{n} X X^\top \quad X = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d \times n} \tag{11}$$

PCA decomposes C:

$$X X^\top = U \Lambda U^\top \quad \text{with } \Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_d \end{bmatrix}$$

3.1 SVD

Any matrix X can be decomposed into

$$X = U D V^\top \tag{12}$$

with $U \in \mathbb{R}^{d \times d}$, $V \in \mathbb{R}^{n \times n}$ and $D = \left[\begin{array}{ccc|c} d_1 & \cdots & 0 & \\ \vdots & \ddots & \vdots & 0 \\ 0 & \cdots & d_d & \end{array}\right] \in \mathbb{R}^{d \times n}$ (for $d < n$).

The following properties hold for U, V, and D:

- d_i are the *singular values* of X ($d_i = \sqrt{\lambda_i}$)
- U contains the *left singular vectors* = EVs of $X X^\top$
- V contains the *right singular vectors* = EVs of $X^\top X$
- U and V are *orthogonal*: $U^\top U = I$, $U U^\top = I$, $U^\top = U^{-1}$

Exercise 3.1. Prove that U is the PCA solution, i.e., show that Eq. (11) is equivalent to Eq. (3).

So, instead of performing PCA which computes C first and then decomposes it (computes U), we decompose X directly using SVD which scales as $\mathcal{O}(d^2 n)$ if $d < n$ (or $\mathcal{O}(n^2 d)$ if $n < d$). This saves use $\mathcal{O}(d^3)$ over PCA.

Besides the use cases of dimensionality reduction discussed above, SVD has other practical uses:

[U1] Matrix inversion or computing the pseudo inverse:
e.g. for MLE/MAP computation: $(XX^\top)^{-1} = (U\Lambda U^\top) = (U^\top)^{-1}(U\Lambda)^{-1} = U\Lambda^{-1}U^{-1} = U\Lambda^{-1}U^\top$

[U2] Matrix approximation (low-rank approximation)

3.2 MDS

Yet, there is another way to compute the same dimensionality reduction called Multidimensional Scaling (MDS). Here, we use the eigenvectors $\{\mathbf{v}_i\}$ of the *Gram matrix* $X^\top X = G \in \mathbb{R}^{n\times n}$ that correspond to the r largest eigenvalues of G. Using $V = [\mathbf{v}_1, \ldots, \mathbf{v}_r]$, the **projection** is simply given as

$$\hat{X} = DV^\top$$

Exercise 3.2. Using the SVD and MDS definitions, show that $V = X^\top UD^{-1}$.

Theorem: MDS and PCA are equivalent: PCA $\iff$ MDS.

$$U^\top X = DV^\top$$

Proof:

$\{\mathbf{v}_i\}$ are EVs of G. So, $GV = V\tilde{\Lambda}$ where $\tilde{\Lambda} = \begin{bmatrix} \tilde{\lambda}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \tilde{\lambda}_r \end{bmatrix}$

$$\begin{aligned}
& GV = V\tilde{\Lambda} \\
\iff\ & X^\top XV = V\tilde{\Lambda} \\
\iff\ & X^\top XV = X^\top UD^{-1}\tilde{\Lambda} \\
\iff\ & XX^\top XV = XX^\top UD^{-1}\tilde{\Lambda} \\
\iff\ & XV = UD^{-1}\tilde{\Lambda} \\
\iff\ & U^\top XV = \underbrace{U^\top U}_{=I} D^{-1}\tilde{\Lambda} \\
\iff\ & U^\top X = D^{-1}\tilde{\Lambda}V^\top \\
\iff\ & U^\top X = DV^\top
\end{aligned}$$

Note:

- use the ordered, scaled, and truncated eigenvectors of G to yield a low dimensional embedding
- the eigenvalues measure how each dimension contributes to the dot products $\mathbf{x}^\top\mathbf{x}'$
- we can estimate the new dimensionality using the number of *significant* (non-negative) eigenvalues
- MDS can be computed in $\mathcal{O}(n^2d + n^3)$
  - $\mathcal{O}(n^2d)$ for computing G
  - $\mathcal{O}(n^3)$ for computing the eigenvectors of G

Question: When to use which method?
Use MDS, if $d > n$; use PCA, if $n > d$ or use SVD anyways!

Notation Summary:

- X = centered data matrix
- U = projection matrix
- D = singular values of X
- Λ = eigenvalues of $C = \frac{1}{n} X X^\top$
- $\tilde{\Lambda}$ = eigenvalues of $G = X^\top X$
- $DV^\top$ = MDS embedding of X

3.3 Data Whitening

(Statistical) whitening is a **data preprocessing** step that can be useful for many ML methods:

$$\tilde{X} = D^{-1} U^\top X \quad \text{with } r = d \tag{13}$$

$\Rightarrow$ rescale all dimensions to have variance 1 and covariance entries of 0. This data processing technique is also called *data standardization*.

Many ML methods assume same variance in all dimensions (k-NN, standard RBF kernel etc.)
Other data/input processing methods are *data centering* and *data normalization*. Fig. 1 illustrates unsupervised data processing strategies that can be used to prepare the training input for supervised machine learning.

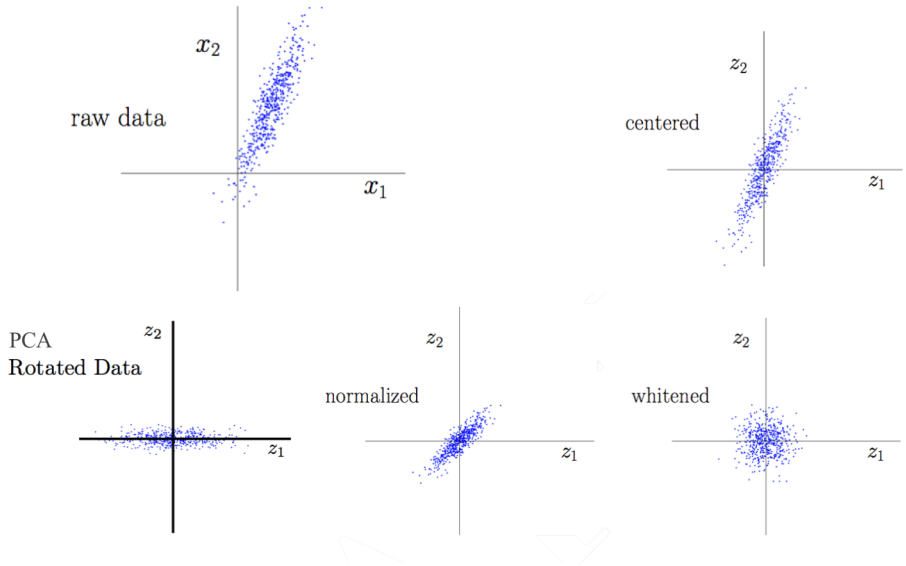
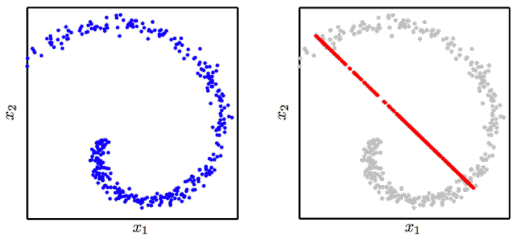


Figure 1: Illustration of different iput data processing transformations.

4 Non-linear Dimensionality Reduction

Let's look at non-linear methods.



(a) Data in $\mathcal{X}$ space (b) Top-1 PCA reconstruction

Figure 2: Illustration of PCA on spiral dataset

4.1 MDS

Recall that via MDS, the (PCA) data projection, cf. Eq. (6), can be computed as DV^T, where $V = [\mathbf{v}_1, \ldots, \mathbf{v}_r]$ are the EVs of $X^\top X = G$ corresponding to the largest eigenvalues of G: $\tilde{\lambda}_1 \geq \tilde{\lambda}_2 \geq \cdots \geq \tilde{\lambda}_r$.

$\Rightarrow$ kernelize MDS by replacing the Gram matrix $G = X^\top X$ by a kernel matrix K.

4.2 Kernel PCA

Let's begin by looking at the covariance matrix in transformed feature space:

$$C = \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top \tag{14}$$

So, we have: $C\mathbf{u}_\alpha = \lambda_\alpha \mathbf{u}_\alpha$, where $\mathbf{u}_\alpha$ are the eigenvectors of C in this transformed feature space.

Goal: Solve this without having to compute $\phi(\mathbf{x}_i)$ and $\mathbf{u}_\alpha$'s. Assume $\sum_i \phi(\mathbf{x}_i) = 0$ (zero-mean in feature space)

$$\frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{x}_i) \underbrace{\phi(\mathbf{x}_i)^\top \mathbf{u}_\alpha}_{=a_{\alpha i}} = \lambda_\alpha \mathbf{u}_\alpha \tag{15}$$

for $\lambda_\alpha > 0$, we have $\mathbf{u}_\alpha$ as a linear combination of $\phi(\mathbf{x}_i)$:

$$\mathbf{u}_\alpha = \sum_{i=1}^{n} a_{\alpha i} \phi(\mathbf{x}_i) \tag{16}$$

$$\begin{aligned}
\text{Eq. (15)} &\iff \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top \sum_{j=1}^{n} a_{\alpha j} \phi(\mathbf{x}_j) = \lambda_\alpha \sum_{i=1}^{n} a_{\alpha i} \phi(\mathbf{x}_i) \\
&\iff \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{x}_l)^\top \phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^\top \sum_{j=1}^{n} a_{\alpha j} \phi(\mathbf{x}_j) = \lambda_\alpha \sum_{i=1}^{n} a_{\alpha i} \phi(\mathbf{x}_l)^\top \phi(\mathbf{x}_i) \\
&\iff \frac{1}{n} \sum_{i=1}^{n} \phi(\mathbf{x}_l)^\top \phi(\mathbf{x}_i) \sum_{j=1}^{n} a_{\alpha j} \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) = \lambda_\alpha \sum_{i=1}^{n} a_{\alpha i} \phi(\mathbf{x}_l)^\top \phi(\mathbf{x}_i) \\
&\iff \frac{1}{n} \sum_{i=1}^{n} \mathrm{k}(\mathbf{x}_l, \mathbf{x}_i) \sum_{j=1}^{n} a_{\alpha j} \mathrm{k}(\mathbf{x}_i, \mathbf{x}_j) = \lambda_\alpha \sum_{i=1}^{n} a_{\alpha i} \mathrm{k}(\mathbf{x}_l, \mathbf{x}_i)
\end{aligned}$$

In matrix notation this is equivalent to

$$K^2\mathbf{a}_\alpha = \lambda_\alpha n K \mathbf{a}_\alpha \iff K\mathbf{a}_\alpha = \lambda_\alpha n \mathbf{a}_\alpha$$

Surprise: this is an eigenvector problem. So, $\mathbf{a}_\alpha$'s are the eigenvectors of K.

Now, cast the projection in terms of the kernel:

$$[\mathbf{v}_i]_\alpha = \mathbf{u}_\alpha^\top \phi(\mathbf{x}_i) = \phi(\mathbf{x}_i)^\top \mathbf{u}_\alpha = \sum_{j=1} a_{\alpha j} \underbrace{\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)}_{\mathrm{k}(\mathbf{x}_i, \mathbf{x}_j)} = \mathbf{a}_\alpha^\top K_{:i} \tag{17}$$

$\forall\ \alpha = 1, \ldots, r.$

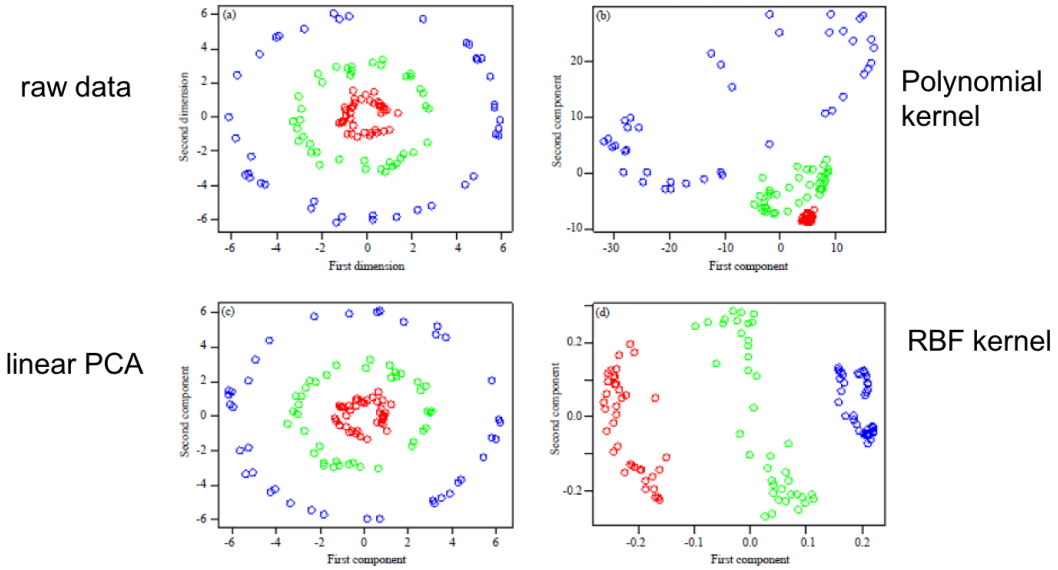


Figure 3: Projections of data using different kernels

4.3 Other Non-linear Dimensionality Reduction Techniques

- Manifold learning: Isomap, Laplacian Eigenmaps (both use distance of pairs of points)
- Maximum variance unfolding
- Topology constraint embedding
- Autoencoders (we'll cover those later in this course)

5 [optional] Probabilistic PCA

Let's model dimensionality reduction as a latent model

$$\mathbf{x}_i = U\mathbf{v}_i + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{18}$$

where $U \in \mathbb{R}^{d\times r}$, $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{v_i} \in \mathbb{R}^r$ $(r < d)$, and $\boldsymbol{\epsilon} \sim N(0, \sigma^2 I_d)$. $\mathbf{v}_i$ is the latent variable that is never observed and $\boldsymbol{\mu}$ allows bias for non-zero mean.

Assumptions:

$$\mathbf{v} \sim N(\mathbf{0}, I_r)$$
$$p(\mathbf{x} \mid \mathbf{v}) \sim N(\mathbf{x} \mid U\mathbf{v} + \boldsymbol{\mu}, \sigma^2 I_d)$$

Generative model:

(1) pick $\mathbf{v}_i$ from r-dimensional Gaussian

(2) project $\mathbf{v}_i$ to $\mathbf{x}_i$ with shift $\boldsymbol{\mu}$

(2) construct the d-dimensional Gaussian distribution of $\mathbf{x}_i$

Note:

- Other model assumptions are possible. i.e. non-linear mappings, GP etc.
- GPLVM: place a prior $\mathbf{u}$ and derive marginal likelihood $p(\mathbf{x} \mid \theta) = N(\mathbf{x} \mid \mathbf{0}, X^\top X + \sigma^2 I)$

Now, marginalize out $\mathbf{v}$:

$$\begin{aligned} p(\mathbf{x}) &= \int_{\mathbf{v}} p(\mathbf{v}) p(\mathbf{x} \mid \mathbf{v}) \mathrm{d}\mathbf{v} \\ &= N(\mathbf{x} \mid \boldsymbol{\mu}, \underbrace{UU^\top}_{\in \mathbb{R}^{d \times d}} + \sigma^2 I_d) \\ &= N(\boldsymbol{\mu}, C) \\ &= p(\mathbf{x} \mid \Theta) \end{aligned} \tag{19}$$

where $C = UU^\top + \sigma^2 I_d$ and $\Theta = \{U, \boldsymbol{\mu}, \sigma^2\}$. See the derivations in Section 6 (Appendix).

Now we need to learn Θ, there are different possibilities:

(1) Posterior over latent variables $p(\mathbf{v} \mid \mathbf{x})$ (similar to GPs)

(2) Variational Bayes (general method, cf. FCML 7.4, 7.5)

(3) Maximize the log likelihood

Here we try to maximize the log likelihood:

$$\begin{aligned} l(D, \Theta) &= \log p(X \mid \Theta) \\ &= \log \prod_{i=1}^{n} p(\mathbf{x}_i \mid \Theta) \\ &= \sum_{i=1}^{n} \log p(\mathbf{x}_i \mid \Theta) \\ &= -\frac{n}{2} \underbrace{r \log(2\pi)}_{=constant} - \frac{n}{2} \log |C| - \frac{1}{2} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})^\top C^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) \\ &= -\frac{n}{2} \log |C| - \frac{1}{2} \sum_{i=1}^{n} Tr((\mathbf{x}_i - \boldsymbol{\mu})^\top C^{-1} (\mathbf{x}_i - \boldsymbol{\mu})) \\ &= -\frac{n}{2} \log |C| - \frac{1}{2} Tr(C^{-1} * n * \underbrace{\frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top}_{\text{sample covariance } S}) \\ &= -\frac{n}{2} [\log |C| + Tr(C^{-1} S)] \end{aligned} \tag{20}$$

MLE Solution

Set $\frac{\partial l}{\partial \boldsymbol{\mu}} = 0$, $\frac{\partial l}{\partial U} = 0$, $\frac{\partial l}{\partial \sigma^2} = 0$, we get the MLE solution:

$$\boldsymbol{\mu}_{MLE} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i \tag{21}$$

$$\sigma^2_{MLE} = \frac{1}{d-r}\sum_{\alpha=r+1}^{d}\lambda_\alpha \tag{22}$$

$$U_{MLE} = W(\Lambda - \sigma^2 I)^{1/2} \tag{23}$$

where

$$W = [\mathbf{w}_1, \ldots, \mathbf{w}_r] \in \mathbb{R}^{d\times r}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_r \end{bmatrix}$$

Here $\mathbf{w}_\alpha$ are the principal EVs of S with $S\mathbf{w}_\alpha = \lambda_\alpha \mathbf{w}_\alpha$ for $\alpha = 1, \ldots, r$ and $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r$. Computing eigenvalues of S takes $\mathcal{O}(d^2 n)$ $(d < n)$. See this paper (especially Appendix A) for a derivation: Probabilistic Principal Component Analysis by Tipping and Bishop[1].

EM Approximation

More efficient computation via expectation maximization, which takes $\Theta(drn)$ time; cf. Appendix B in the Tipping and Bishop paper.

Complete data likelihood:

$$\mathcal{L}(X, V \mid \Theta) = \sum_{i=1}^{n}\log p(\mathbf{x}_i, \mathbf{v}_i) = \sum_{i=1}^{n}\log p(\mathbf{x}_i \mid \mathbf{v}_i)p(\mathbf{v}_i) \tag{24}$$

E-step: compute expectation of $\mathcal{L}(X, V \mid \Theta)$
M-step: maximize $\mathcal{L}(X, V \mid \Theta)$ w.r.t W, σ^2.

Some Notes:

- for $\sigma^2 \to 0$, we get standard PCA
- related: Factor Analysis, Independent Component Analysis, Gaussian Process Latent Variable Models (GPLVM)
- GPLVM is a generalization of PPCA. In kernel PCA, we model $\mathbf{x}_i \to \mathbf{v}_i$; In PPCA or GPLVM, we model $\mathbf{v}_i \to \mathbf{x}_i$

6 Appendix

Affine transformation of a multivariate Gaussian:

If $p(\mathbf{v}) \sim \mathcal{N}(\mathbf{m}, S)$, then $p(A\mathbf{v} + b) \sim \mathcal{N}(A\mathbf{m} + b, ASA^\top)$.

[1] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/bishop-ppca-jrss.pdf

Rewrite $\mathbf{x} \mid \mathbf{v}$ as a sum using $p(\mathbf{x} \mid \mathbf{v}) = \mathcal{N}(U\mathbf{v} + \boldsymbol{\mu}, \sigma^2 I)$:

$$\mathbf{x} \mid \mathbf{v} = U\mathbf{v} + \boldsymbol{\mu} + \epsilon$$

where $(U\mathbf{v} + \boldsymbol{\mu}) \sim \mathcal{N}(U\mathbf{m} + \boldsymbol{\mu}, USU^\top)$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I)$.

For $p(v) = \mathcal{N}(\mathbf{0}, I)$, $(U\mathbf{v} + \boldsymbol{\mu}) \sim \mathcal{N}(U\mathbf{m} + \boldsymbol{\mu}, USU^\top) = \mathcal{N}(\boldsymbol{\mu}, UU^\top)$
$\Rightarrow p(\mathbf{x} \mid \mathbf{v}) = N(\boldsymbol{\mu} + \mathbf{0}, UU^\top + \sigma^2 I)$