EECS 545: Machine Learning

Lecture 16. Unsupervised Learning: EM & PCA

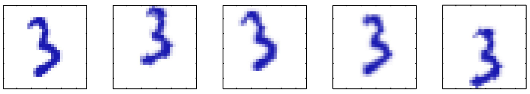
Honglak Lee

3/12/2025

Principal Component Analysis (PCA)

High-Dimensional Data

- ... may have low-dimensional structure.



(above: images of digit "3" via translations and rotations)

- The data is 100x100-dimensional.
- But there are only three degrees of freedom, so it lies on a 3-dimensional subspace (x, y, angle).
  - (on a non-linear manifold, in this case)

Principal Component Analysis

- Given a set of $\{\mathbf{x}^{(n)}\}_{n=1,\ldots,N}$ of observations
  - in a space of dimension *D*, i.e., $\mathbf{x}^{(n)} \in \mathbb{R}^D$
  - find a subspace of dimension *M* < *D*
  - that captures most of its *variability*. (i.e., approximate $x^{(n)}$'s using principal components as basis vectors)

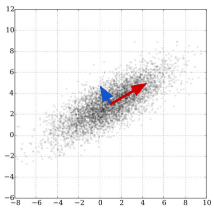


Image source: Wikipedia

Principal Component Analysis

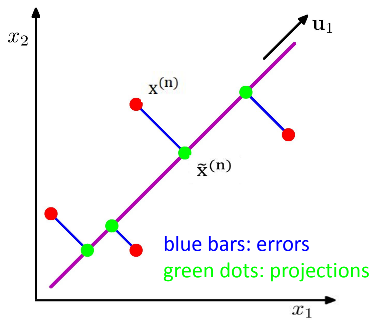
- Given a set of $\{\mathbf{x}^{(n)}\}_{n=1,\ldots,N}$ of observations
  - in a space of dimension *D*,
  - find a subspace of dimension *M* < *D*
  - that captures most of its *variability*. (i.e., approximate $x^{(n)}$'s using principal components as basis vectors)
- PCA can be described as either:
  - maximizing the variance of the projection, or
  - minimizing the squared approximation error.
  - (both are equivalent; see the next slide)

Two Descriptions of PCA

Approximate the data with *projection*

(i.e., for each $x^{(n)}$, find closest point on on the subspace spanned by principal components):

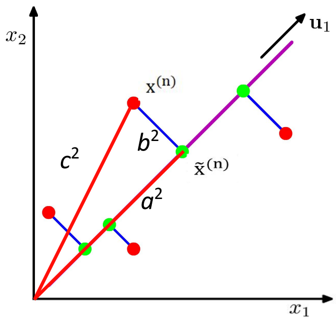
- Maximize variance, or
- Minimize squared error



Main idea: We want to find a basis vector (e.g. $\mathbf{u}_1$) (= principal component) that does the best approximation or best preserves the variance when projected

Equivalent Descriptions

- With mean at the origin $c_i^2 = a_i^2 + b_i^2$
- With constant $\sum_i c_i^2$
  - Minimizing $\sum_i b_i^2$
  - Maximizes $\sum_i a_i^2$
  - … and vice versa



Note: without loss of generality, here we assume that the input data x has zero mean.

First Principal Component

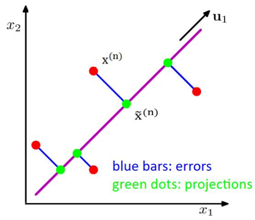
- Given data points $\{\mathbf{x}^{(n)}\}$ in a D-dim space,
  - Mean $\bar{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N} \mathbf{x}^{(n)}$
  - Data covariance $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N} (\mathbf{x}^{(n)} - \bar{\mathbf{x}})(\mathbf{x}^{(n)} - \bar{\mathbf{x}})^\top$

D x D matrix

First Principal Component

- Given data points $\{\mathbf{x}^{(n)}\}$ in a D-dim space,
- Let $\mathbf{u}_1$ be the PC maximizing variance of projection:
  - It should have length 1: $\mathbf{u}_1^\top \mathbf{u}_1 = 1$
  - Projection of $\mathbf{x}^{(n)}$ to $\mathbf{u}_1$ subspace:

$$(\mathbf{u}_1^\top \mathbf{x}^{(n)})\mathbf{u}_1$$



- Remark: More generally, projection of $\mathbf{x}^{(n)}$ to subspace spanned by $\mathbf{u}_1,\ldots,\mathbf{u}_M$: $\sum_{j=1}^{M}(\mathbf{u}_j^\top \mathbf{x}^{(n)})\mathbf{u}_j$

First Principal Component

- Maximize the projection variance:

$$\frac{1}{N}\sum_{n=1}^{N}(\mathbf{u}_1^\top \mathbf{x}^{(n)} - \mathbf{u}_1^\top \overline{\mathbf{x}})^2 = \mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1$$

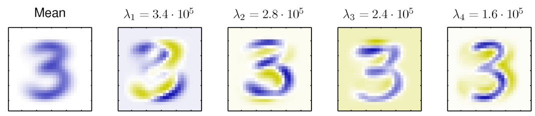
- Use a Lagrange multiplier to enforce $\mathbf{u}_1^\top \mathbf{u}_1 = 1$
- Maximize: $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^\top \mathbf{u}_1)$
- Derivative is zero when $\mathbf{S}\mathbf{u}_1 = \lambda_1 \mathbf{u}_1$
  - That is, $\mathbf{u}_1^\top \mathbf{S}\mathbf{u}_1 = \lambda_1$
- So $\mathbf{u}_1$ is eigenvector with largest eigenvalue.

PCA by Maximizing Variance

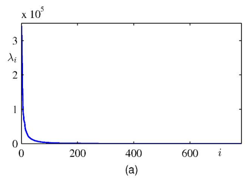
- Repeat to find the *M* eigenvectors of the data covariance matrix **S** corresponding to the *M* largest eigenvalues.
  - The *total variance* is the sum of variances of all individual principal components
  - Principal components are orthogonal to each other
- We can also do the same thing from a "minimizing (projection) squared error" viewpoint.

Digit Image Example

- The mean and first four PCA eigenvectors.

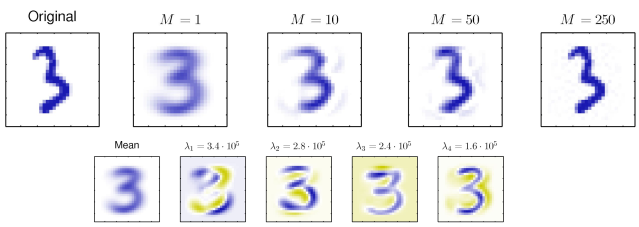


- The eigenvalue spectrum:



Reconstructing the Image

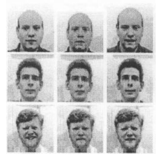
- Compress the image representation by using only first M eigenvectors, and discarding the less important information.



Learning features via PCA

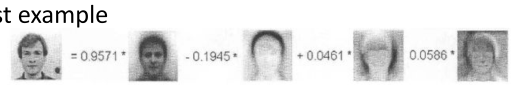
- Example: Eigenfaces

Training face images



Learned PCA bases

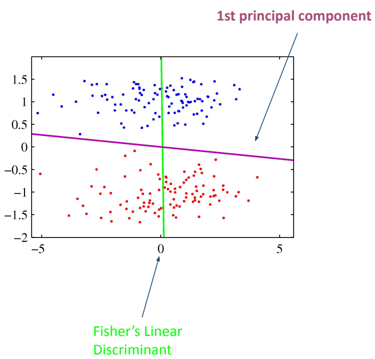
Test example



Images from www.cse.unr.edu/~bebis/CS485/Lectures/Eigenfaces.ppt

Limits to PCA

- Maximizing variance is not always the best way to make the structure visible.
- PCA vs Fisher's linear discriminant



Probabilistic PCA

- We can view PCA as solving a probabilistic latent variable problem.
- Describe a distribution $p(\mathbf{x})$ in *D*-dimensional space, in terms of a latent variable $\mathbf{z}$ in *M*-dimensional space.

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \qquad p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I}) \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$$

- **W** is a *D* by *M* linear transformation from **z** to **x**

$$p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2 \mathbf{I})$$

Probabilistic PCA

- Given the generative model

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$$

- we can infer

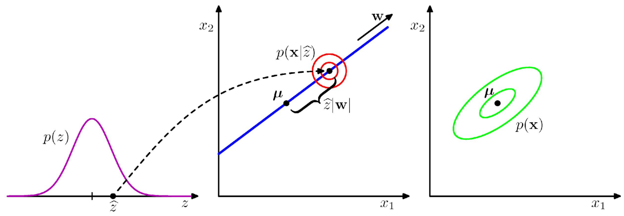
$$\begin{aligned} \mathbb{E}[\mathbf{x}] &= \mathbb{E}[\mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}] = \boldsymbol{\mu} \\ \mathrm{cov}[\mathbf{x}] &= \mathbb{E}[(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})(\mathbf{W}\mathbf{z} + \boldsymbol{\epsilon})^\top] \\ &= \mathbb{E}[\mathbf{W}\mathbf{z}\mathbf{z}^\top\mathbf{W}^\top] + \mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top] = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I} \end{aligned}$$

Probabilistic PCA

- The generative model

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}$$

can be illustrated as:



Likelihood of Probabilistic PCA

- (Marginal) likelihood:

$$\begin{aligned} &\log p(\mathbf{x} \mid \mathbf{W}, \boldsymbol{\mu}, \sigma^2) \\ &= \sum_i p(\mathbf{x}^{(i)} \mid \mathbf{W}, \boldsymbol{\mu}, \sigma^2) \\ &= -\frac{ND}{2}\log 2\pi - \frac{N}{2}\log|C| - \frac{1}{2}\sum_i (\mathbf{x}^{(i)} - \boldsymbol{\mu})^\top C^{-1}(\mathbf{x}^{(i)} - \boldsymbol{\mu}) \end{aligned}$$

where $\quad C = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}$

- We can simply maximize this likelihood function with respect to $\mathbf{W}, \boldsymbol{\mu}, \sigma^2$.

Maximum Likelihood Parameters

- Mean: $\quad \boldsymbol{\mu} = \overline{\mathbf{x}}$
- Noise: $\quad \sigma^2_{\mathrm{ML}} = \dfrac{1}{D-M}\displaystyle\sum_{i=M+1}^{D} \lambda_i$
- **W**: $\quad \mathbf{W}_{\mathrm{ML}} = \mathbf{U}_M(\mathbf{L}_M - \sigma^2\mathbf{I})^{1/2}\mathbf{R}$

where

- $\mathbf{L}_M$ is diag with the *M* largest eigenvalues
- $\mathbf{U}_M$ is the M corresponding eigenvectors
- **R** is an arbitrary *M* by *M* orthogonal matrix (rotation matrix) (i.e., **z** can be defined by rotating "back")

Maximum likelihood by EM

- Latent variable model

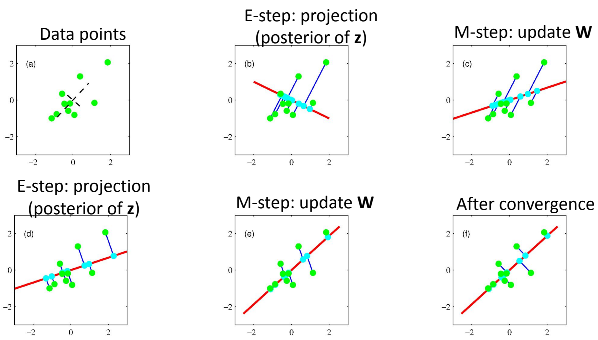
$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z} \mid \mathbf{0}, \mathbf{I})$$

$$p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}(\mathbf{x} \mid \mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \sigma^2\mathbf{I})$$

- E-step: Estimate the posterior $q(\mathbf{z}) = p(\mathbf{z} \mid \mathbf{x})$
  - Use linear Gaussian
- M-step: Maximize the data-completion likelihood given $q(\mathbf{z})$

$$\mathrm{maximize}_{\theta=\{\mathbf{W},\boldsymbol{\mu},\sigma\}} \sum_i \sum_{\mathbf{z}^{(i)}} q(\mathbf{z}^{(i)}) \log p_\theta(\mathbf{x}^{(i)}, \mathbf{z}^{(i)})$$

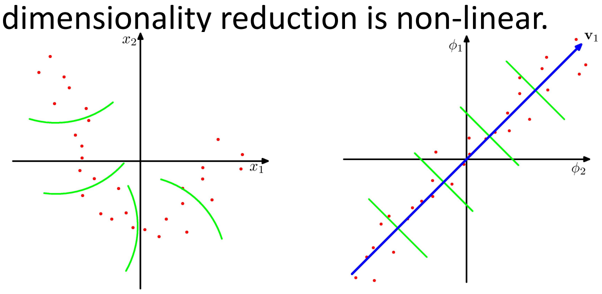
Finding PCA params by EM



- Illustrating EM on simulated data

Kernel PCA

- Suppose the regularity that allows dimensionality reduction is non-linear.



Kernel PCA

- As with regression and classification, we can transform the raw input data $\{\mathbf{x}^{(n)}\}$ to a set of feature values

$$\{\mathbf{x}^{(n)}\} \rightarrow \{\ \phi(\mathbf{x}^{(n)})\ \}$$

- Linear PCA (on the nonlinear feature space) gives us a linear subspace in the feature value space, corresponding to nonlinear structure in the data space.

Kernel PCA

- Define a kernel, to avoid having to evaluate the feature vectors explicitly.

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^\top \phi(\mathbf{x}')$$

- Express PCA in terms of the kernel,
  – Some care is required to centralize the data.

$$K_{nm} = \phi(\mathbf{x}^{(n)})^\top \phi(\mathbf{x}^{(m)}) = k(\mathbf{x}^{(n)}, \mathbf{x}^{(m)})$$

Kernel PCA

- Assume that $\{\phi(\mathbf{x}^{(n)})\}$ have zero mean.
- Sample covariance matrix: $S = \frac{1}{N}\sum_{n=1}^{N} \phi(\mathbf{x}^{(n)})\phi(\mathbf{x}^{(n)})^\top = \frac{1}{N}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}$
- Let **v** be an eigenvector for *S*

$$S\mathbf{v} = \lambda\mathbf{v} \implies \lambda\mathbf{v} = \boldsymbol{\Phi}^\top \left(\frac{1}{N}\boldsymbol{\Phi}\mathbf{v}\right)$$

$$\therefore\ \mathbf{v} = \boldsymbol{\Phi}^\top\boldsymbol{\alpha} \quad \text{for some } \boldsymbol{\alpha} \in \mathbb{R}^N$$

- Thus, $S\mathbf{v} = \lambda\mathbf{v} \implies \lambda\boldsymbol{\Phi}^\top\boldsymbol{\alpha} = \frac{1}{N}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\alpha} = \frac{1}{N}\boldsymbol{\Phi}^\top K\boldsymbol{\alpha}$
- Multiply Φ on both sides and cancel out $K = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$

$$\lambda N\boldsymbol{\alpha} = K\boldsymbol{\alpha} \implies \boldsymbol{\alpha} \text{ is an eigenvector of } K$$

Kernel PCA

- We thus have $\mathbf{v} = \boldsymbol{\Phi}^\top\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is eigenvector of the kernel matrix *K*.
- Now, $\|\mathbf{v}\| = 1 \implies \boldsymbol{\alpha}^\top K\boldsymbol{\alpha} = \boldsymbol{\alpha}^\top\lambda_K\boldsymbol{\alpha} = 1 \implies \|\boldsymbol{\alpha}\| = \lambda_K^{-1/2}$
- It is often infeasible to obtain $\mathbf{v}$ (depends on dim of Φ), but we can compute projections:

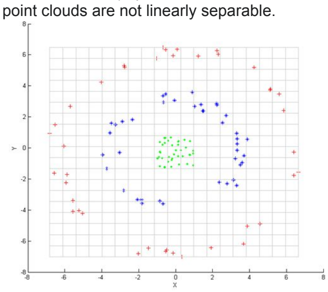
$$\mathbf{v}^\top\phi(\mathbf{x}) = \boldsymbol{\alpha}^\top\boldsymbol{\Phi}\phi(\mathbf{x}) = \boldsymbol{\alpha}^\top k(\mathbf{x}) \quad \text{where} \quad k(\mathbf{x}) = \left[k(\mathbf{x}^{(1)}, \mathbf{x}), \ldots, k(\mathbf{x}^{(N)}, \mathbf{x})\right]$$

- Finally, some care is required to centralize data (to ensure that features have zero mean):

$$K' = K - \mathbf{1}_N K - K\mathbf{1}_N + \mathbf{1}_N K \mathbf{1}_N$$
$$\text{where} \quad \mathbf{1}_N \in \mathbb{R}^{N\times N} \quad \text{is a matrix of ones.}$$

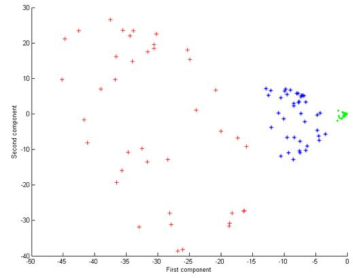
Kernel PCA

Linear PCA operates only in the given (in this case two-dimensional) space, in which these concentric point clouds are not linearly separable.



Data

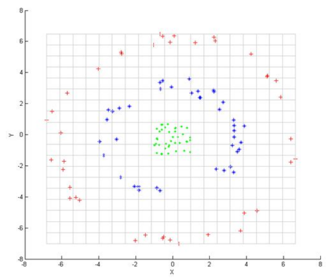
The first principal component is enough to distinguish the three different groups



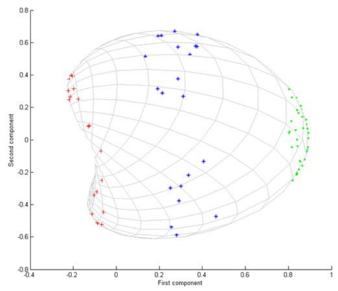
Kernel PCA with $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^\top\mathbf{y} + 1)^2$

Source: https://en.wikipedia.org/wiki/Kernel_principal_component_analysis

Kernel PCA



Data



Kernel PCA with Gaussian kernel

Source: https://en.wikipedia.org/wiki/Kernel_principal_component_analysis