ECE 1513: Introduction to Machine Learning

Lecture 3: Principle Component Analysis

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering

University of Toronto

Winter 2025

Quick Recap: *ML General Recipe*

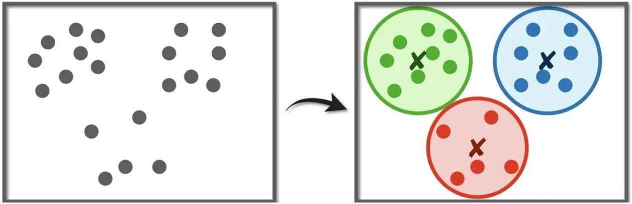
We defined ML as

> *the set of data-driven approaches that help us understand the environment and its behavior, and generalize it!*

Any learning task is accomplished by ML through *three major steps*

- Collect data
- Specify a model *that captures the pattern*
- Develop a learning algorithm

Quick Recap: *Clustering*



- *Data*
  - ↳ *Collection of samples* $\mathbb{D} = \{\boldsymbol{x}_n : n = 1, \ldots, N\}$
- *Model*
  - ↳ *A function mapping* $\boldsymbol{x}$ *to a cluster, e.g.,* K*-centroid model*
- *Learning algorithm*
  - ↳ *It takes* $\mathbb{D}$ *and returns a good clustering*

Quick Recap: *Clustering*

```
K-Means():
Initiate μ_1, ..., μ_K
while μ_1, ..., μ_K changing do
    Set C_1, ..., C_K ← Cluster_Assignment(μ_1, ..., μ_K)
    Update μ_1, ..., μ_K ← Centroid_Update(C_1, ..., C_K)
end while
Return μ_1, ..., μ_K
```

Quick Recap: *Density Estimation*

We look at the data as a stochastic process

- *We sample the dataset*

$$\mathbb{D} = \{\boldsymbol{x}_n : n = 1, \ldots, N\}$$

- *We know (assume) some distribution for the process*
  - ↳ Model: $P_{\boldsymbol{\theta}}$ for some unknown $\boldsymbol{\theta}$
- *Learning algorithm*
  - ↳ *Infers a good $\boldsymbol{\theta}$ by observing $\mathbb{D}$*

Quick Recap: *Maximum Likelihood*

The learning algorithm is maximum likelihood

$$
\begin{aligned}
\boldsymbol{\theta}^\star &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \mathcal{L}_{\mathbb{D}}\left(\boldsymbol{\theta}\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P_{\boldsymbol{\theta}}\left(\mathbb{D}\right) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, P_{\boldsymbol{\theta}}\left(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\right)
\end{aligned}
$$

Today's Agenda: *Dimensionality Reduction*

In today's lecture, we study our last *unsupervised* learning task, i.e.,

dimensionality reduction

through the following steps

- *Representing data in lower dimension*
  - ↳ *We review key notions in linear algebra*
- *Principle component analysis*
- *A look at particular applications*
  - ↳ *Image compression*
  - ↳ *Recommendation systems*
- *Wrapping up unsupervised learning*

Motivation: *Compression*

Say we have data samples in D-dimensional space

$$\mathbb{D} = \left\{ \boldsymbol{x}_n \in \mathbb{R}^D : n = 1, \ldots, N \right\}$$

? *Can we represent it in lower dimension?*

$$f : \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_D \end{bmatrix} \mapsto \boldsymbol{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_K \end{bmatrix}$$

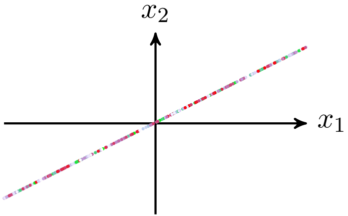
Example: *Image Compression*

Consider a 60K pixel image

? *Can we represent it with much less pixel while maintaining quality?*

Representing Data in Lower Dimensions: *Example I*

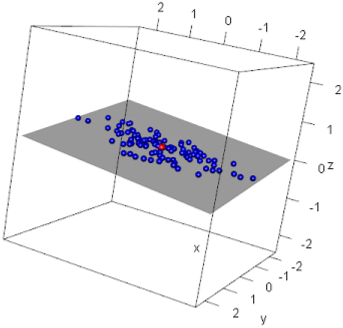
Consider the example where all samples lie on a line



! *Obviously, we can represent each sample with only one scalar!*

Representing Data in Lower Dimensions: *Example II*

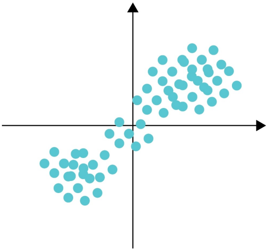
Consider the example where all samples lie on a 2D plane



! *We can represent each sample with a 2D vector*

Representing Data in Lower Dimensions: *General Form*

In reality, the samples might be more spread in one direction



! *We can have an approximate low-dimensional representation*

Orthonormal Bases

$\mathbf{u}_i \in \mathbb{R}^D$ form an orthonormal base if

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_D] \rightsquigarrow \begin{cases} \mathbf{u}_j^\mathsf{T} \mathbf{u}_j = 0 \\ \|\mathbf{u}_i\| = 1 \end{cases} \rightsquigarrow \mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}_D$$

Classic Base

Classic base is given by the identity matrix $\mathbf{I}_D$

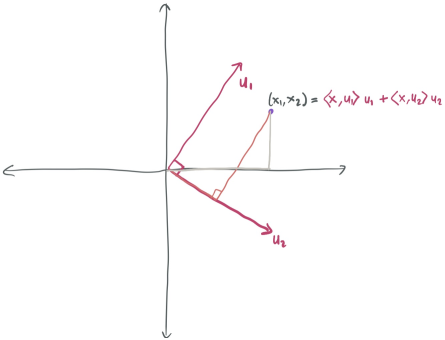
Key Feature

Orthonormal base only rotates and does not change the norm

$$\|\mathbf{U}\boldsymbol{x}\| = \|\boldsymbol{x}\|$$

Orthonormal Bases

We can use orthonormal bases to represent a vector



Recall

In D-dimensional space, we have only D orthogonal vectors

Eigenvalues and Eigenvectors

Say we have a square matrix $\mathbf{A} \in \mathbb{R}^{D \times D}$

Eigenvalue and Eigenvector

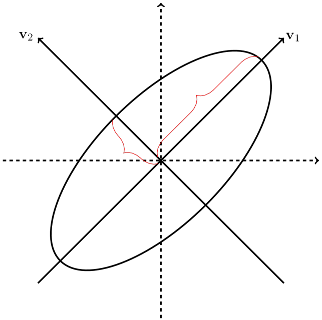
$(\lambda, \mathbf{v})$ *is an eigenvalue and eigenvector pair if*

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}$$

Recall

We consider eigenvectors to be unit-norm, i.e., $\|\mathbf{v}\| = 1$

Eigenvalues and Eigenvectors: *Visualization*



! *Eigenvectors describe an orthonormal base*

Matrix Decomposition

Let's put everything in a matrix form

$$\begin{aligned}\mathbf{A}\begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_D\end{bmatrix} &= \begin{bmatrix}\lambda_1\mathbf{v}_1 & \cdots & \lambda_D\mathbf{v}_D\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{v}_1 & \cdots & \mathbf{v}_D\end{bmatrix}\begin{bmatrix}\lambda_1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \lambda_D\end{bmatrix}\end{aligned}$$

Eigendecomposition

We can hence say that

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{T}}$$

where $\mathbf{V}$ is an orthonormal base and $\mathbf{\Lambda}$ is diagonal matrix of eigenvalues

Key Features

- *We can compute determinant from eigenvalues*

$$\det \mathbf{A} = \prod_{i=1}^{D} \lambda_i$$

- *We can compute trace from eigenvalues*

$$\operatorname{tr}\{\mathbf{A}\} = \sum_{i=1}^{D} \lambda_i$$

- *A positive semi-definite matrix has only non-negative eigenvalues*

$$\mathbf{A} \succeq 0 \rightsquigarrow \lambda_i \geqslant 0$$

↳ *Famous example of a positive semi-definite matrix*

$$\mathbf{A} = \mathbf{X}\mathbf{X}^{\mathsf{T}}$$

Dimensionality Reduction: *Problem Formulation*

Single direction $\mathbf{u} \in \mathbb{R}^D$

$$\hat{\boldsymbol{x}}_n = z_n \mathbf{u} + \boldsymbol{\mu}$$

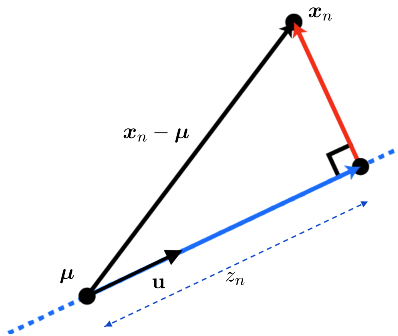
Of course $\mathbf{u}$ is a base, i.e.,

$$\|\mathbf{u}\| = 1$$

Dimensionality Reduction: *Problem Formulation*

How to find z_n?

$$z_n = \mathbf{u}^{\mathsf{T}} \left(\boldsymbol{x}_n - \boldsymbol{\mu} \right)$$



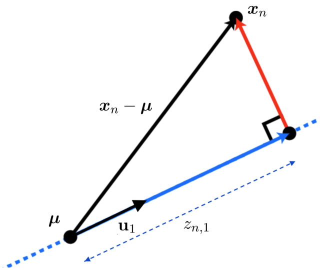
Dimensionality Reduction: *Problem Formulation*

Two directions $\mathbf{u}_1$ and $\mathbf{u}_2 \in \mathbb{R}^D$

$$
\begin{aligned}
\hat{\boldsymbol{x}}_n &= z_{n,1}\mathbf{u}_1 + z_{n,2}\mathbf{u}_2 + \boldsymbol{\mu} \\
&= [\mathbf{u}_1, \mathbf{u}_2] \begin{bmatrix} z_{n,1} \\ z_{n,2} \end{bmatrix} + \boldsymbol{\mu} = \mathbf{U}\boldsymbol{z}_n + \boldsymbol{\mu}
\end{aligned}
$$

How to find $\boldsymbol{z}_n$?

$$z_{n,1} = \mathbf{u}_1^{\mathsf{T}} \left(\boldsymbol{x}_n - \boldsymbol{\mu} \right) \qquad z_{n,2} = \mathbf{u}_2^{\mathsf{T}} \left(\boldsymbol{x}_n - \boldsymbol{\mu} \right)$$



Dimensionality Reduction: *Problem Formulation*

So we can write

$$
\begin{aligned}
\boldsymbol{z}_n = \begin{bmatrix} z_{n,1} \\ z_{n,2} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_1^{\mathsf{T}} \left(\boldsymbol{x}_n - \boldsymbol{\mu}\right) \\ \mathbf{u}_2^{\mathsf{T}} \left(\boldsymbol{x}_n - \boldsymbol{\mu}\right) \end{bmatrix} &= \begin{bmatrix} \mathbf{u}_1^{\mathsf{T}} \\ \mathbf{u}_2^{\mathsf{T}} \end{bmatrix} \left(\boldsymbol{x}_n - \boldsymbol{\mu}\right) \\
&= \left[\mathbf{u}_1, \mathbf{u}_2\right]^{\mathsf{T}} \left(\boldsymbol{x}_n - \boldsymbol{\mu}\right) \\
&= \mathbf{U}^{\mathsf{T}} \left(\boldsymbol{x}_n - \boldsymbol{\mu}\right)
\end{aligned}
$$

Dimensionality Reduction: *Problem Formulation*

$\mathbf{u}_1$ and $\mathbf{u}_2$ are bases, i.e.,

$$\|\mathbf{u}_1\| = \|\mathbf{u}_2\| = 1 \qquad\qquad \mathbf{u}_1^\mathsf{T}\mathbf{u}_2 = 0$$

So we can say

$$\mathbf{U}^\mathsf{T}\mathbf{U} = \begin{bmatrix} \|\mathbf{u}_1\|^2 & \mathbf{u}_1^\mathsf{T}\mathbf{u}_2 \\ \mathbf{u}_1^\mathsf{T}\mathbf{u}_2 & \|\mathbf{u}_2\|^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \mathbf{I}_2$$

Orthonormal bases!

Dimensionality Reduction: *Problem Formulation*

$K < D$ directions $\mathbf{u}_1, \dots, \mathbf{u}_K \in \mathbb{R}^D$

$$\begin{aligned}
\hat{\boldsymbol{x}}_n &= \sum_{k=1}^{K} z_{n,k} \mathbf{u}_k + \boldsymbol{\mu} \\
&= [\mathbf{u}_1, \dots, \mathbf{u}_K] \begin{bmatrix} z_{n,1} \\ \vdots \\ z_{n,K} \end{bmatrix} + \boldsymbol{\mu} \\
&= \mathbf{U} \boldsymbol{z}_n + \boldsymbol{\mu}
\end{aligned}$$

Similarly, we can find

$$\boldsymbol{z}_n = \mathbf{U}^{\mathsf{T}} \left(\boldsymbol{x}_n - \boldsymbol{\mu} \right)$$

Dimensionality Reduction: *Problem Formulation*

We know that $\mathbf{U}$ contains orthonormal bases

$$\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_K] \rightsquigarrow \begin{cases} \mathbf{u}_k^{\mathsf{T}} \mathbf{u}_j = 0 \\ \|\mathbf{u}_k\| = 1 \end{cases} \rightsquigarrow \mathbf{U}^{\mathsf{T}} \mathbf{U} = \mathbf{I}_K$$

Attention

Note that $\mathbf{U} \in \mathbb{R}^{D \times K}$ *and that*

$$\mathbf{U}\mathbf{U}^{\mathsf{T}} \neq \mathbf{I}_D$$

Latent Space $\leftrightsquigarrow$ Reconstruction

Dimensionality Reduction: *Latent Variable*

Dimensionality is reduced linearly via $\mathbf{U}^{\mathsf{T}} : \mathbb{R}^D \mapsto \mathbb{R}^K$ *as*

$$\boldsymbol{z}_n = \mathbf{U}^{\mathsf{T}} \left(\boldsymbol{x}_n - \boldsymbol{\mu} \right)$$

We call $\boldsymbol{z}_n$ *latent variable*

Higher Dimensional Recovery: *Reconstruction*

We reconstruct a sample $\boldsymbol{x}_n \in \mathbb{R}^D$ *from its latent variable* $\boldsymbol{z}_n \in \mathbb{R}^K$ *as*

$$\hat{\boldsymbol{x}}_n = \mathbf{U} \boldsymbol{z}_n + \boldsymbol{\mu}$$

Dimensionality Reduction as *Learning Task*

- Data
  - ↳ Collection of samples $\mathbb{D} = \{\boldsymbol{x}_n : n = 1, \ldots, N\}$
- Model
  - ↳ Linear transform $\boldsymbol{z} = \mathbf{U}^{\mathsf{T}} (\boldsymbol{x} - \boldsymbol{\mu})$
- Learning algorithm
  - ↳ We look for algorithm $\mathcal{A} : \mathbb{D} \mapsto \mathbf{U}^\star, \boldsymbol{\mu}^\star$
  - ↳ $\mathbf{U}^\star, \boldsymbol{\mu}^\star$ are good ones!

? *How can we define a good transform?*

! *It should not kill to much information!*

Preserving Information Maximally

Say we specify $\mathbf{U}$ and $\boldsymbol{\mu}$ and compute latent variable $\boldsymbol{z}$ of sample $\boldsymbol{x}$: *if we want the data sample back, we reconstruct from the latent space as*

$$\hat{\boldsymbol{x}} = \mathbf{U}\boldsymbol{z} + \boldsymbol{\mu}$$

The information on $\boldsymbol{x}_n$ is preserved if

$$\hat{\boldsymbol{x}} \overset{!}{=} \boldsymbol{x} \leftrightsquigarrow \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2 \overset{!}{=} 0$$

But, we know that it's not happening perfectly; thus, we try

$$\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2 \overset{!}{\approx} 0$$

- ? *We cannot check it for every data sample!*
- ! *But, we can check it for samples in $\mathbb{D}$!*

Information Preservation $\equiv$ *Minimum Representation Error*

We want $\mathbf{U}$ and $\boldsymbol{\mu}$ to make $\|\hat{\boldsymbol{x}}_n - \boldsymbol{x}_n\|^2$ as small as possible for $\boldsymbol{x}_n \in \mathbb{D}$: *so we could try to make the average*

$$\mathcal{J}\left(\mathbf{U}, \boldsymbol{\mu}\right) = \frac{1}{N} \sum_{n=1}^{N} \|\hat{\boldsymbol{x}}_n - \boldsymbol{x}_n\|^2$$

is minimized $\rightsquigarrow$ *if* $\mathcal{J}\left(\mathbf{U}, \boldsymbol{\mu}\right) = 0$; *then,* $\hat{\boldsymbol{x}}_n = \boldsymbol{x}_n$ *for all* n

Dimensionality Reduction via *Minimum Representation Error*

Optimal $\mathbf{U}^\star$ *and* $\boldsymbol{\mu}^\star$ *are defined as*

$$\mathbf{U}^\star, \boldsymbol{\mu}^\star = \operatorname*{argmin}_{\mathbf{U}, \boldsymbol{\mu}} \mathcal{J}\left(\mathbf{U}, \boldsymbol{\mu}\right)$$

Optimal Bias $\equiv$ *Data Centroid*

Let's start with simpler one: *we want to find* $\boldsymbol{\mu}^\star$

$$\mathcal{J} = \frac{1}{N}\sum_{n=1}^{N} \|\hat{\boldsymbol{x}}_n - \boldsymbol{x}_n\|^2 = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{U}\boldsymbol{z}_n + \boldsymbol{\mu} - \boldsymbol{x}_n\|^2$$

$$= \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{U}\mathbf{U}^\mathsf{T}(\boldsymbol{x}_n - \boldsymbol{\mu}) + \boldsymbol{\mu} - \boldsymbol{x}_n\|^2$$

$$= \frac{1}{N}\sum_{n=1}^{N} \left\|\left(\mathbf{U}\mathbf{U}^\mathsf{T} - \mathbf{I}_D\right)(\boldsymbol{x}_n - \boldsymbol{\mu})\right\|^2$$

Let's call $\mathbf{A} = \mathbf{U}\mathbf{U}^\mathsf{T} - \mathbf{I}_D$*: we want to find* $\boldsymbol{\mu}^\star$ *which minimizes*

$$\mathcal{J} = \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{A}\boldsymbol{x}_n - \mathbf{A}\boldsymbol{\mu}\|^2$$

Optimal Bias $\equiv$ *Data Centroid*

If we call $\boldsymbol{y}_n = \mathbf{A}\boldsymbol{x}_n$ *and* $\mathbf{a} = \mathbf{A}\boldsymbol{\mu}$*; then, we look for*

$$\mathbf{a}^\star = \operatorname*{argmin}_{\mathbf{a}} \frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{y}_n - \mathbf{a}\|^2$$

which is its centroid, i.e.,

$$\mathbf{a}^\star = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n$$

So, we have

$$\mathbf{A}\boldsymbol{\mu}^\star = \frac{1}{N}\sum_{n=1}^{N} \mathbf{A}\boldsymbol{x}_n \rightsquigarrow \boldsymbol{\mu}^\star = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_n$$

Alternative Formulation

Let's now have a few observations: *call the error of sample n*

$$\mathbf{e}_n = \hat{\boldsymbol{x}}_n - \boldsymbol{x}_n$$

We compute the inner product of $\mathbf{e}_n$ and $\hat{\boldsymbol{x}}_n - \boldsymbol{\mu}^\star$, i.e.,

$$\phi_n = \left(\hat{\boldsymbol{x}}_n - \boldsymbol{\mu}^\star \right)^\mathsf{T} \mathbf{e}_n$$

We know that $\hat{\boldsymbol{x}}_n - \boldsymbol{\mu}^\star = \mathbf{U}\boldsymbol{z}_n$, so we can write

$$\phi_n = \left(\hat{\boldsymbol{x}}_n - \boldsymbol{\mu}^\star \right)^\mathsf{T} \mathbf{e}_n = \boldsymbol{z}_n^\mathsf{T} \mathbf{U}^\mathsf{T} \mathbf{e}_n$$

We can also open $\mathbf{e}_n$ as

$$\begin{aligned} \mathbf{e}_n &= \mathbf{U}\boldsymbol{z}_n + \boldsymbol{\mu}^\star - \boldsymbol{x}_n \\ &= \mathbf{U}\boldsymbol{z}_n - \left(\boldsymbol{x}_n - \boldsymbol{\mu}^\star \right) \end{aligned}$$

Alternative Formulation

So, the inner product reads

$$\begin{aligned}\phi_n &= \boldsymbol{z}_n^\mathsf{T} \mathbf{U}^\mathsf{T} \mathbf{e}_n \\ &= \boldsymbol{z}_n^\mathsf{T} \mathbf{U}^\mathsf{T} \left(\mathbf{U} \boldsymbol{z}_n - (\boldsymbol{x}_n - \boldsymbol{\mu}^\star) \right) \\ &= \boldsymbol{z}_n^\mathsf{T} \mathbf{U}^\mathsf{T} \mathbf{U} \boldsymbol{z}_n - \boldsymbol{z}_n^\mathsf{T} \mathbf{U}^\mathsf{T} \left(\boldsymbol{x}_n - \boldsymbol{\mu}^\star \right)\end{aligned}$$

Recall that $\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}_K$*, so we can write*

$$\boldsymbol{z}_n^\mathsf{T} \mathbf{U}^\mathsf{T} \mathbf{U} \boldsymbol{z}_n = \boldsymbol{z}_n^\mathsf{T} \boldsymbol{z}_n$$

Also we know that $\mathbf{U}^\mathsf{T} \left(\boldsymbol{x}_n - \boldsymbol{\mu}^\star \right) = \boldsymbol{z}_n$*, so we could say*

$$\boldsymbol{z}_n^\mathsf{T} \mathbf{U}^\mathsf{T} \left(\boldsymbol{x}_n - \boldsymbol{\mu}^\star \right) = \boldsymbol{z}_n^\mathsf{T} \boldsymbol{z}_n$$

This means that

$$\phi_n = \boldsymbol{z}_n^\mathsf{T} \boldsymbol{z}_n - \boldsymbol{z}_n^\mathsf{T} \boldsymbol{z}_n = 0$$

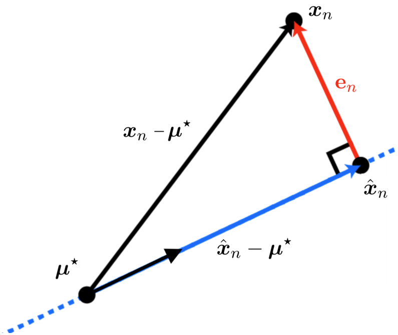
Alternative Formulation

Orthogonality Principle

The error $\mathbf{e}_n$ and unbiased estimate $\hat{\boldsymbol{x}}_n - \boldsymbol{\mu}^\star$ are orthogonal

We can hence write

$$\left\| \hat{\boldsymbol{x}}_n - \boldsymbol{\mu}^\star \right\|^2 + \left\| \mathbf{e}_n \right\|^2 = \left\| \boldsymbol{x}_n - \boldsymbol{\mu}^\star \right\|^2$$



Alternative Formulation

We have seen that

$$\|\hat{\boldsymbol{x}}_n - \boldsymbol{\mu}^\star\|^2 = \|\mathbf{U}\boldsymbol{z}_n\|^2 = \|\boldsymbol{z}_n\|^2$$

and also know that $\|\boldsymbol{x}_n - \boldsymbol{\mu}^\star\|^2$ *has nothing to do with* $\mathbf{U}$*; thus,*

$$\|\boldsymbol{z}_n\|^2 + \|\mathbf{e}_n\|^2 = \textit{constant}$$

If we average over n*, we conclude*

$$\frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{z}_n\|^2 + \mathcal{J} = \textit{constant} \rightsquigarrow \mathcal{J} = \textit{constant} - \frac{1}{N}\sum_{n=1}^{N}\|\boldsymbol{z}_n\|^2$$

Alternative Formulation: *Maximal Representation Variance*

Dimensionality Reduction via *Minimum Representation Error*

Optimal $\mathbf{U}^\star$ *is given by*

$$\mathbf{U}^\star = \operatorname*{argmin}_{\mathbf{U}} \mathcal{J}$$

Dimensionality Reduction via *Maximal Representation Variance*

Optimal $\mathbf{U}^\star$ *is given by*

$$\mathbf{U}^\star = \operatorname*{argmax}_{\mathbf{U}} \frac{1}{N} \sum_{n=1}^{N} \|\boldsymbol{z}_n\|^2$$

Notion of Correlation in Data

? *Why do we call it variance?*

We know that $\mathbf{U}^\mathsf{T} \left(\boldsymbol{x}_n - \boldsymbol{\mu}^\star \right) = \boldsymbol{z}_n$, so can say

$$\begin{aligned} \| \boldsymbol{z}_n \|^2 = \| \mathbf{U}^\mathsf{T} \boldsymbol{x}_n - \mathbf{U}^\mathsf{T} \boldsymbol{\mu}^\star \|^2 &= \| \mathbf{U}^\mathsf{T} \boldsymbol{x}_n - \frac{1}{N} \sum_{n=1}^{N} \mathbf{U}^\mathsf{T} \boldsymbol{x}_n \|^2 \\ &= \| \mathbf{U}^\mathsf{T} \boldsymbol{x}_n - \mathtt{avg} \left(\mathbf{U}^\mathsf{T} \mathbb{D} \right) \|^2 \end{aligned}$$

Therefore, we could say

$$\begin{aligned} \frac{1}{N} \sum_{n=1}^{N} \| \boldsymbol{z}_n \|^2 &= \mathtt{var} \left(\mathbf{U}^\mathsf{T} \mathbb{D} \right) \\ &= \textit{variance in the latent space} \end{aligned}$$

Notion of Variance in Data

? *How can we maximize the variance?*

Say $K = 1$, i.e., $\mathbf{U} = \mathbf{u} \rightsquigarrow z_n = \mathbf{u}^\mathsf{T} \left(\boldsymbol{x}_n - \boldsymbol{\mu}^\star \right)$

$$\begin{aligned} \frac{1}{N} \sum_{n=1}^{N} |z_n|^2 &= \frac{1}{N} \sum_{n=1}^{N} \left| \mathbf{u}^\mathsf{T} \left(\boldsymbol{x}_n - \boldsymbol{\mu}^\star \right) \right|^2 \\ &= \frac{1}{N} \| \mathbf{u}^\mathsf{T} \tilde{\mathbf{X}} \|^2 \end{aligned}$$

where we define

$$\tilde{\mathbf{X}} = \left[\boldsymbol{x}_1 - \boldsymbol{\mu}^\star, \ldots, \boldsymbol{x}_N - \boldsymbol{\mu}^\star \right] \in \mathbb{R}^{D \times N}$$

Sample Covariance

Sample Covariance

The sample covariance of the dataset is given by

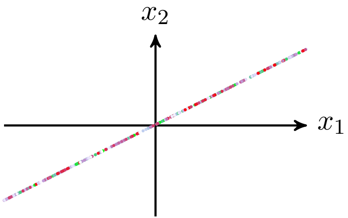
$$\mathbf{\Sigma} = \frac{1}{N}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\mathsf{T}$$

In case of $K = 1$, we can write the variance as

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}|z_n|^2 &= \frac{1}{N}\mathbf{u}^\mathsf{T}\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\mathsf{T}\mathbf{u} \\ &= \mathbf{u}^\mathsf{T}\mathbf{\Sigma}\mathbf{u}\end{aligned}$$

Sample Covariance

Let's compute sample covariance for the line example

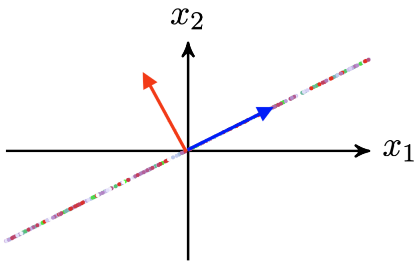


$$\tilde{\mathbf{X}} = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 20 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

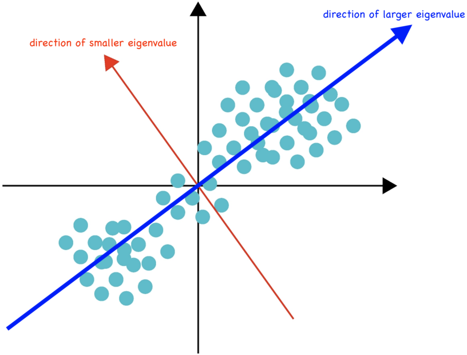
Sample Covariance

The large eigenvalue shows us where the data spans more



Sample Covariance

This applies to any dataset



! *It can tell us right direction for dimensionality reduction!*

Optimal Transform $\equiv$ *Principle Components of Covariance*

$\boldsymbol{\Sigma}$ is positive semi-definite: *we thus have*

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_D \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_D \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^\mathsf{T} \\ \vdots \\ \mathbf{v}_D^\mathsf{T} \end{bmatrix}$$

with $\lambda_1, \ldots, \lambda_D \geqslant 0$. *Let's replace in the variance expression*

$$\begin{aligned} \frac{1}{N} \sum_{n=1}^{N} |z_n|^2 &= \mathbf{u}^\mathsf{T} \boldsymbol{\Sigma} \mathbf{u} \\ &= \mathbf{u}^\mathsf{T} \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_D \end{bmatrix} \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_D \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^\mathsf{T} \\ \vdots \\ \mathbf{v}_D^\mathsf{T} \end{bmatrix} \mathbf{u} \\ &= \sum_{i=1}^{D} \lambda_i \left(\mathbf{u}^\mathsf{T} \mathbf{v}_i \right)^2 \end{aligned}$$

Optimal Transform $\equiv$ *Principle Components of Covariance*

We want to find optimal $\mathbf{u}$: *it should maximize the following term*

$$\frac{1}{N}\sum_{n=1}^{N}|z_n|^2 = \sum_{i=1}^{D}\lambda_i\left(\mathbf{u}^\mathsf{T}\mathbf{v}_i\right)^2$$

- $\mathbf{v}_i$*'s are bases; thus, we always have*

$$\left(\mathbf{u}^\mathsf{T}\mathbf{v}_i\right)^2 \leqslant 1$$

- $[\mathbf{v}_1 \cdots \mathbf{v}_D]$ *is an orthonormal matrix; thus,*

$$\left\|\mathbf{u}^\mathsf{T}[\mathbf{v}_1 \cdots \mathbf{v}_D]\right\| = \sum_{i=1}^{D}\left(\mathbf{u}^\mathsf{T}\mathbf{v}_i\right)^2 = 1$$

Optimal Transform $\equiv$ *Principle Components of Covariance*

We want to find optimal $\mathbf{u}$: *its should be set to*

$$\mathbf{u} = \mathbf{v}_{\max}$$

where $\mathbf{v}_{\max}$ *corresponds to the maximum eigenvalue* $\lambda_{\max}$

$$\frac{1}{N}\sum_{n=1}^{N} |z_n|^2 = \lambda_{\max}$$

Optimal Transform $\equiv$ *Principle Components of Covariance*

Say $K = 2$, i.e., $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2] \rightsquigarrow z_{n,1}, z_{n,2}$

$$\frac{1}{N}\sum_{n=1}^{N} \|\boldsymbol{z}_n\|^2 = \frac{1}{N}\sum_{n=1}^{N} |z_{n,1}|^2 + \frac{1}{N}\sum_{n=1}^{N} |z_{n,2}|^2$$
$$= \mathbf{u}_1^\mathsf{T} \boldsymbol{\Sigma} \mathbf{u}_1 + \mathbf{u}_2^\mathsf{T} \boldsymbol{\Sigma} \mathbf{u}_2$$

With same lines of derivations, we could say

- *Optimal* $\mathbf{u}_1$ *is* $\mathbf{v}_{\max_1}$ *corresponding to the largest eigenvalue*
- *Optimal* $\mathbf{u}_2$ *is* $\mathbf{v}_{\max_2}$ *corresponding to the second largest eigenvalue*

Principle Component Analysis: *General Algorithm*

`PCA():`

1: Set $\boldsymbol{\mu}$ to the centroid of dataset

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n$$

2: Construct the sample covariance matrix

$$\boldsymbol{\Sigma} = \frac{1}{N} \tilde{\mathbf{X}} \tilde{\mathbf{X}}^{\mathsf{T}}$$

3: Decompose $\boldsymbol{\Sigma}$ in terms of its eigenvalues an eigenvectors as $\boldsymbol{\Sigma} = \mathbf{V} \boldsymbol{\Lambda} \mathbf{V}^{\mathsf{T}}$

4: Find the K largest eigenvalues and set $\mathbf{u}_k = \mathbf{v}_{\max_k}$

5: Set the projection matrix to

$$\mathbf{U} = [\mathbf{u}_1, \dots, \mathbf{u}_K]$$

Formulating the Problem

We can look at an image as a data sample

This is a 300×200 *image*

we can look at it as a 60000*-dimensional sample* $\boldsymbol{x}_n$

Formulating the Problem

? *How many samples then we need?*

! *Obviously large enough to build sample covariance* $\mathbf{\Sigma}$

Recall that the sample covariance is of the size $\mathbf{\Sigma} \in \mathbb{R}^{D\times D}$

? *This would be a huge matrix to decompose!*

We can alternatively look at the image as

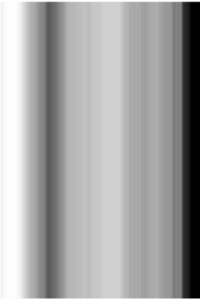
$\mathbf{X} \in \mathbb{R}^{300\times 200}$ *and treat each column as a single* 300*-dimensional sample*

So, the image itself is a dataset!

$$\mathbf{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{200}]$$

Using PCA to Compress Images

- ? *How well we can compress?*
- ! *Let's look at the case with $K = 0$!*



Using PCA to Compress Images

? *How well we can compress?*

! *Let's look at the case with $K = 10$!*

Using PCA to Compress Images

- ? *How well we can compress?*
- ! *Let's look at the case with* $K = 20$*!*

Using PCA to Compress Images

? *How well we can compress?*

! *Let's look at the case with $K = 30$!*

Using PCA to Compress Images

- ? *How well we can compress?*
- ! *Let's look at the case with* $K = 40$*!*

Using PCA to Compress Images

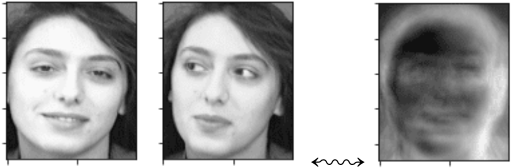
- ? *How well we can compress?*
- ! *Let's look at the case with $K = 50$!*

Sounds to be good enough!

Sample Example: *Face Recognition*

One classic application of PCA is face recognition

- *We compress high-dimensional images via PCA to latent space*
- *We use the latent representations to compare two pictures*
  ↳ *If they belong to the same person, latent representations should be close*
- *If we choose K properly then we could have right latent representation*



Formulating the Problem

? *How to find missing rating?*

Formulating the Problem as PCA

? *How to find missing rating?*

We can look at it as a completion problem

$$\text{rating user } n = \boldsymbol{x}_n = \begin{bmatrix} \text{movie } 1 \\ \vdots \\ \text{movie } D \end{bmatrix} = \sum_{k=1}^{K} z_{n,k} \mathbf{u}_k$$

- *There are K eigenratings!*
- *Each user is almost perfectly by its latent representation $\boldsymbol{z}_n$*

It sounds like PCA!

Sample Example: *Recommendation System*

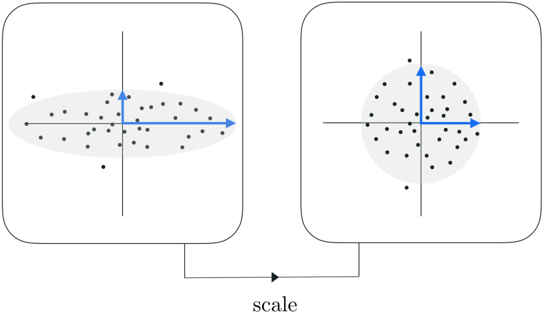
This approach is used in many recommendation systems

- *Our ratings in online shops are used to find the eigenratings*
- *Eigenratings approximate our true ratings from latent space*
- *The shop will the decide what to advertise*

In 2009, this idea won the \$ 1M Netflix Prize

Sensitivity to Scaling

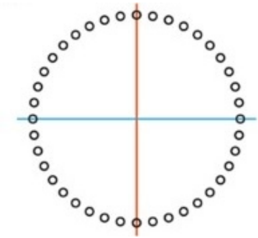
- ! *PCA is sensitive to scaling*



- ! *We typically normalize the data before applying PCA*

PCA is Linear!

- *PCA cannot capture nonlinear patterns*



- *Extending PCA to nonlinear patterns gave birth to Autoencoders*

Further Read

- Bishop
  - ↳ Chapter 12: *Section 12.1* — ***PCA***
- ESL
  - ↳ Chapter 14: *Section 14.1* — ***Principle Components***
- Goodfellow
  - ↳ Chapter 2 — ***Review on Linear Algebra***
  - ↳ Chapter 5: *Section 5.8.1* — ***PCA***
- MacKay
  - ↳ Chapter 34 — ***Latent Space Design***

Where are We?

We studied three major unsupervised learning tasks

- *Clustering*
  - ↳ *Data*
  - ↳ *Model: K-centroid*
  - ↳ *Learning algorithm: K-means clustering*
- *Distribution Learning*
  - ↳ *Data*
  - ↳ *Model: a distribution with unknowns*
  - ↳ *Learning algorithm: maximum likelihood*
- *Dimensionality Reduction*
  - ↳ *Data*
  - ↳ *Model: projection*
  - ↳ *Learning algorithm: PCA*

Next Stop

We next look at supervised learning

- *Linear regression*
- *Linear classification*
- *Support vector machines*
- *Neural networks*