DAM lecture 6:

Dimensionality reduction 1
(and a bit of linear algebra)
25.02.2016

Aasa Feragen
aasa@diku.dk

Assignment 1 experiences

- I will make a poll on A1 workload – I hope you will participate
- Assignment 2 will have a considerably higher workload; start early!
- Save time – develop your code within the templates

Assignment 1 experiences

- CodeChecker experiences – let's learn for A2:
  - Develop your code within the template from the start! Trying to fit your existing code to the template gives hard-to-debug mistakes.

Assignment 1 experiences

- CodeChecker experiences – let's learn for A2:
  - Develop your code within the template from the start! Trying to fit your existing code to the template gives hard-to-debug mistakes.
  - Respect the input and output format of the template as described in the beginning of the template

Assignment 1 experiences

- CodeChecker experiences – let's learn for A2:
  - Develop your code within the template from the start! Trying to fit your existing code to the template gives hard-to-debug mistakes.
  - Respect the input and output format of the template as described in the beginning of the template
  - You can start checking your code at the first exercise – you don't need to finish it all

Assignment 1 experiences

- CodeChecker experiences – let's learn for A2:
  - Develop your code within the template from the start! Trying to fit your existing code to the template gives hard-to-debug mistakes.
  - Respect the input and output format of the template as described in the beginning of the template
  - You can start checking your code at the first exercise – you don't need to finish it all
  - Do not read the data within the code – this will overwrite alternative dataset input used for some exercises. That means, your uploaded code should not include any line of code that reads or loads the data file.

After today's lecture you should

- recall variance and covariance from lecture 2, and gain a deeper understanding of covariance through its eigenvalue decomposition
- know the definition of principal component analysis (PCA)
- be able to compute PCA using eigenvalue decomposition
- be able to use PCA for visualization of variation along principal components (PCs)
- be familiar with the equivalence definitions of PCA by least squares projection error minimization, projected variance maximization, and eigenvalue decomposition of the covariance matrix

Literature for today's lecture

- Chapters 4 and 10
- **Shlens tutorial:**
  Optional; fantastic intro to PCA with Matlab code
  (find it on Absalon)

Recall from Lecture 2: Data centering

- Given a data set $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$, replace each data point with the transformed

$$\boldsymbol{z}_n = \boldsymbol{x}_n - \bar{\boldsymbol{x}},$$

and do your analysis on the $\boldsymbol{z}_n$.

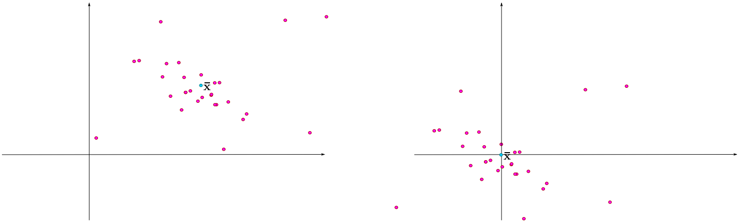


Figure : Left: Datapoints $\boldsymbol{x}_n$. Right: centered datapoints $\boldsymbol{z}_n = \boldsymbol{x}_n - \bar{\boldsymbol{x}}$.

Recall from Lecture 2: Data normalization

- A common preprocessing step is to *normalize* the features (the data coordinates) by dividing each coordinate by its standard deviation.
- That is, for a dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\}$, where $\boldsymbol{x}_n = (x_{n1}, x_{n2}, \ldots, x_{nd})$, replace $\boldsymbol{x}_n$ with $\boldsymbol{z}_n$, where

$$z_{ni} = \frac{x_{ni}}{s_{x_i}}$$

- What is the effect of normalization?

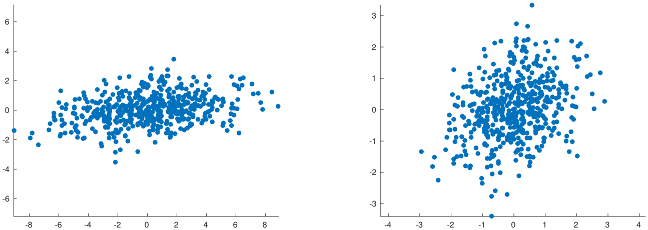


Figure : Before normalization, after normalization

Recall from Lecture 2: Covariance

- The **covariance** of the x- and y-coordinates in the sample $\{(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)\}$ of points in the plane $\mathbb{R}^2$ is:

$$cov(x, y) = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(y_n - \bar{y})$$

- The covariance between i^{th} and j^{th} coordinate for the sample $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_d\} \subset \mathbb{R}^d$ is:

$$cov(x_i, x_j) = \frac{1}{N} \sum_{n=1}^{N} (x_{n,i} - \bar{x}_{\cdot,i})(x_{n,j} - \bar{x}_{\cdot,j}))$$

- NB! $cov(x_i, x_i) = var(x_i)$

Recall from Lecture 2: Covariance matrix

- Still working with a sampled dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$, we can define its $d \times d$ *covariance matrix* Σ by setting

$$\Sigma_{i,j} = cov(x_i, x_j).$$

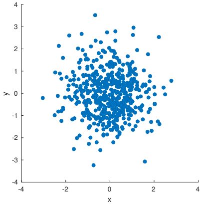
- What does it mean if Σ is diagonal? What are the diagonal elements?

$$\Sigma = \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \lambda_d \end{pmatrix}$$

- The variance of each coordinate is found along the diagonal! $s_{x_i}^2 = \lambda_i$!

Recall from Lecture 2:
2D case: Diagonal elements of Σ and dataset "shape"

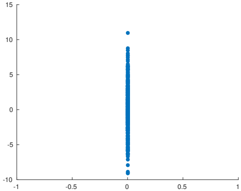
- If the diagonal elements are identical, then the dataset is "circular"



$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Recall from Lecture 2:
2D case: Diagonal elements of Σ and dataset "shape"

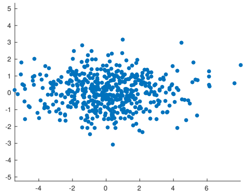
- If the diagonal elements are identical, then the dataset is "circular"
- What is the dataset like if one f the diagonal elements is 0? The dataset lies on a lower-dimensional linear subspace (a line)



$$\Sigma = \begin{pmatrix} 0 & 0 \\ 0 & 10 \end{pmatrix}$$

Recall from Lecture 2:
2D case: Diagonal elements of Σ and dataset "shape"

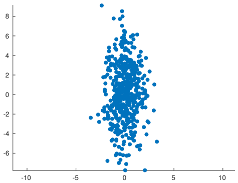
- If the diagonal elements are identical, then the dataset is "circular"
- What is the dataset like if one f the diagonal elements is 0?
  The dataset lies on a lower-dimensional linear subspace (a line)
- What is the dataset like if $0 < \lambda_2 < \lambda_1$?
  The dataset can be approximated by an ellipse with eccentricity $\frac{\sqrt{\lambda_1}}{\sqrt{\lambda_2}}$.



$$\Sigma = \begin{pmatrix} 5 & 0 \\ 0 & 1 \end{pmatrix}$$

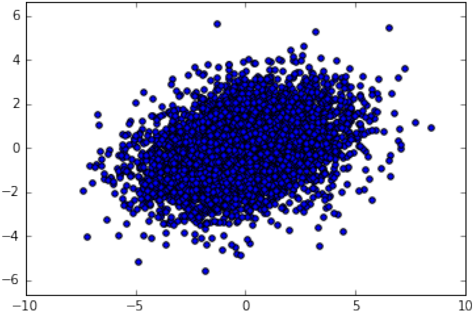
Recall from Lecture 2:
2D case: Diagonal elements of Σ and dataset "shape"

- If the diagonal elements are identical, then the dataset is "circular"
- What is the dataset like if one f the diagonal elements is 0?
  The dataset lies on a lower-dimensional linear subspace (a line)
- What is the dataset like if $0 < \lambda_2 < \lambda_1$?
  The dataset can be approximated by an ellipse with eccentricity $\frac{\sqrt{\lambda_1}}{\sqrt{\lambda_2}}$.
- What is this dataset like?



$$\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & 10 \end{pmatrix}$$

But what about those covariance matrices that are not diagonal?



But what about those covariance matrices that are not diagonal?

Theorem (Eigenvalue decomposition)

If Σ is a $d \times d$ matrix with linearly independent eigenvectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$, with corresponding eigenvalues $\lambda_1, \ldots, \lambda_d$, then Σ has a decomposition

$$\Sigma = Q \begin{pmatrix} \lambda_1 & 0 & \ldots & 0 \\ 0 & \lambda_2 & \ldots & 0 \\ \vdots & \vdots & \ddots \vdots & \\ 0 & 0 & \ldots & \lambda_d \end{pmatrix} Q^{-1},$$

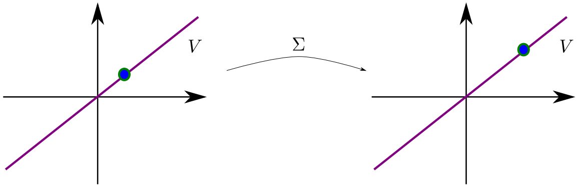
where the columns of Q are the eigenvectors $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d$.

Timeout: What is an eigenvalue and its eigenvector?

- An *eigenvector* for a matrix Σ is a vector $\boldsymbol{e}$ such that

$$\Sigma \boldsymbol{e} = \lambda \boldsymbol{e}$$

for some real number λ, called the *eigenvalue* of $\boldsymbol{e}$. In this course, we further ask that $\|\boldsymbol{e}\| = 1$.

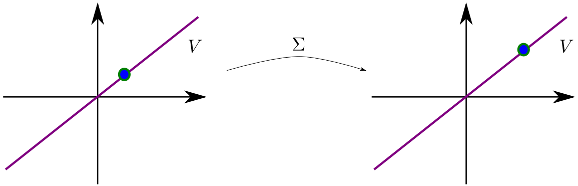


Timeout: What is an eigenvalue and its eigenvector?

- An *eigenvector* for a matrix Σ is a vector $\boldsymbol{e}$ such that

$$\Sigma \boldsymbol{e} = \lambda \boldsymbol{e}$$

  for some real number λ, called the *eigenvalue* of $\boldsymbol{e}$. In this course, we further ask that $\|\boldsymbol{e}\| = 1$. (Why is that a possible requirement)?

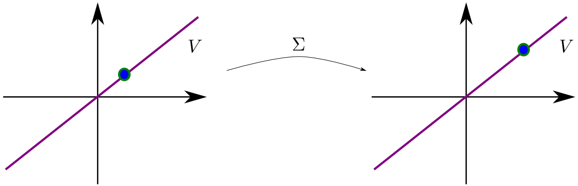


Timeout: What is an eigenvalue and its eigenvector?

- An *eigenvector* for a matrix Σ is a vector $\boldsymbol{e}$ such that

$$\Sigma \boldsymbol{e} = \lambda \boldsymbol{e}$$

  for some real number λ, called the *eigenvalue* of $\boldsymbol{e}$. In this course, we further ask that $\|\boldsymbol{e}\| = 1$. (Why is that a possible requirement)?
- What does this mean?

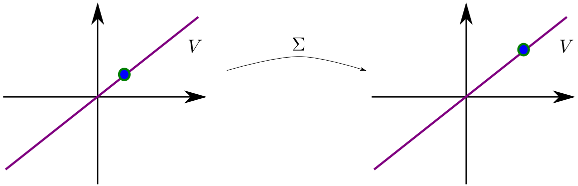


Timeout: What is an eigenvalue and its eigenvector?

- An *eigenvector* for a matrix Σ is a vector $\boldsymbol{e}$ such that

$$\Sigma \boldsymbol{e} = \lambda \boldsymbol{e}$$

  for some real number λ, called the *eigenvalue* of $\boldsymbol{e}$. In this course, we further ask that $\|\boldsymbol{e}\| = 1$. (Why is that a possible requirement)?
- What does this mean?
- The eigenvector $\boldsymbol{e}$ spans a subspace V which is left invariant by Σ – that is, for any $\boldsymbol{v} \in V$, we also have $\Sigma(\boldsymbol{v}) \in V$.

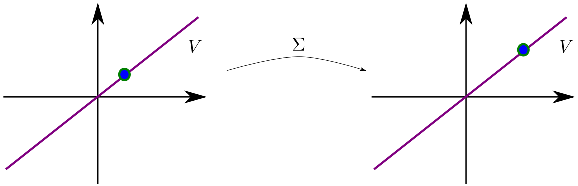


Timeout: What is an eigenvalue and its eigenvector?

- An *eigenvector* for a matrix Σ is a vector $\boldsymbol{e}$ such that

  $$\Sigma \boldsymbol{e} = \lambda \boldsymbol{e}$$

  for some real number λ, called the *eigenvalue* of $\boldsymbol{e}$. In this course, we further ask that $\|\boldsymbol{e}\| = 1$. (Why is that a possible requirement)?
- What does this mean?
- The eigenvector $\boldsymbol{e}$ spans a subspace V which is left invariant by Σ – that is, for any $\boldsymbol{v} \in V$, we also have $\Sigma(\boldsymbol{v}) \in V$.
- The eigenvalue λ tells you how much Σ *stretches* V.



Timeout: What is a basis?

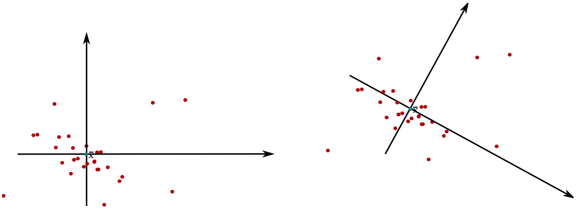
- Given a vector space (like $\mathbb{R}^d$), an "orthonormal basis" consists of a set of unit length vectors

$$\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_d$$

  which are all at 90 deg angle with each other
- These vectors define a coordinate system in $\mathbb{R}^d$.
- Example: The standard basis

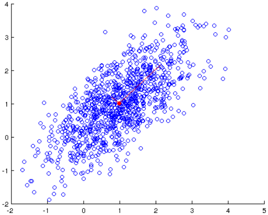
$$(1,0,0,\ldots,0),(0,1,0,\ldots,0),(0,0,1,\ldots,0),\ldots,(0,0,0,\ldots,1)$$

  defines the usual x, y, z, etc axes as a coordinate system.



What does the eigenvalue decomposition of the covariance matrix mean?

- What does
$$\Sigma = Q \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_d \end{pmatrix} Q^{-1}$$
mean?
- Q is a change of bases, re-expressing the covariance matrix in the basis defined by the eigenvectors.

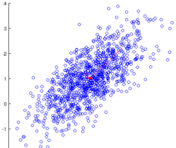


What does the eigenvalue decomposition of the covariance matrix mean?

- The diagonal matrix

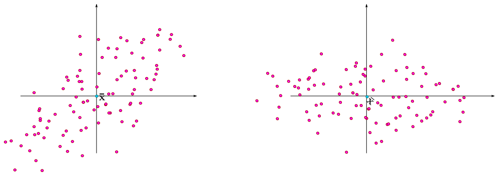
$$D = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_d \end{pmatrix}$$

  is the covariance of the dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$, expressed in the new basis.
- What do you see?
  - The coordinates of the data points in the new basis are independent!
  - The variance of each coordinate is found along the diagonal!



What does the change of basis do?

- Align principal components with axes in the new coordinate system
- The intrinsic geometry of the data is unchanged! Only rotation and reflection.
  (Because eigenvectors are orthonormal)



Useful fact: The multivariate Gaussian distribution

- The multivariate Gaussian distribution on $\mathbb{R}^d$, that is over d-dimensional vectors $\boldsymbol{x}$, is given by the probability density function

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}$$

- One can prove that if X is a random variable sampled from f, then $E[X] = \mu$ and that the covariance matrix of the distribution f is Σ.
- This is useful because it lets us visualize covariance matrices by sampling from f!

2D case: Eigenvalues of the covariance and dataset "shape"

Let's turn to an iPython notebook example

Summary: Interpreting covariance

- The *covariance matrix* Σ describes dependencies between coordinates
- Eigenvalue decomposition: $\Sigma = QDQ^{-1}$ defines a new basis of eigenvectors, in which the covariance matrix of the data is the diagonal matrix

$$D = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_d \end{pmatrix}$$

- The λ_i correspond to the projected variance along the principal components defined by the eigenvectors e_i

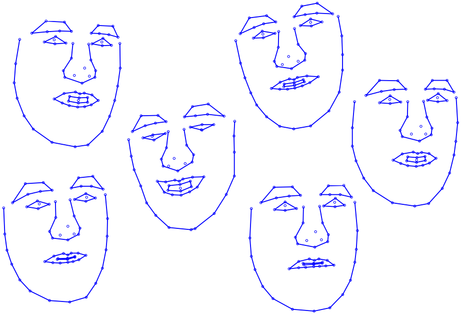
Example: Face shape[1]

- Here's an image of a man's face
- How do you detect what the face looks like? Can a computer learn to do the same?

[1] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Face shape[1]

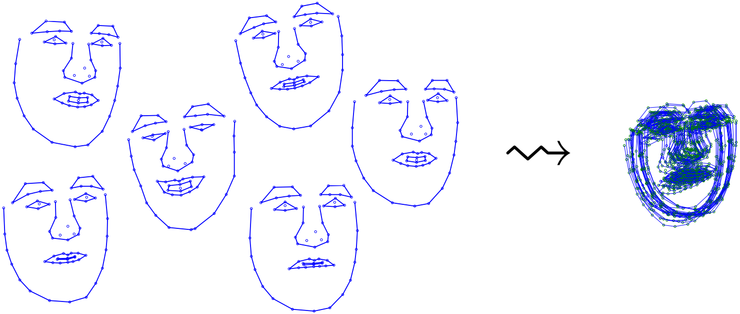
- Here are a set of connect-the-dots-figures describing the man's face while he is talking.
- How can I describe the variation in the face?



[1] All face data from Tim Cootes' talking face dataset,
http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Face shape[1]

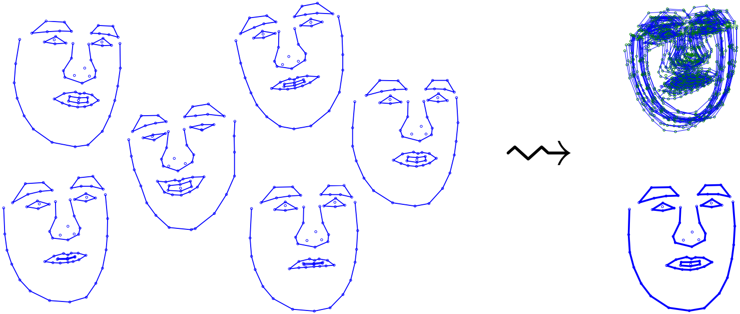
- Here are a set of connect-the-dots-figures describing the man's face while he is talking.
- How can I describe the variation in the face?



[1] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Face shape[1]

- Here are a set of connect-the-dots-figures describing the man's face while he is talking.
- How can I describe the variation in the face?



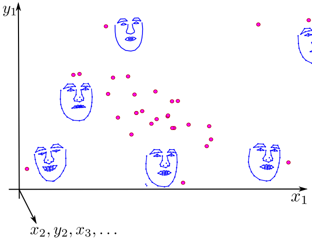
[1] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Face shape[1]

- Each face is described by 68 landmark points
- Consider a high-dimensional "face space" consisting of vectors

$$(x_1, y_1, x_2, y_2, \ldots, x_{68}, y_{68}).$$

- Not all vectors in $\mathbb{R}^{2\times 68}$ result in natural-looking faces!



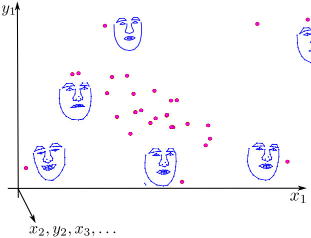
[1] All face data from Tim Cootes' talking face dataset,
http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Face shape[1]

- Each face is described by 68 landmark points
- Consider a high-dimensional "face space" consisting of vectors

$$(x_1, y_1, x_2, y_2, \ldots, x_{68}, y_{68}).$$

- Not all vectors in $\mathbb{R}^{2\times 68}$ result in natural-looking faces!
- Often high-dimensional data has a low intrinsic dimensionality or few degrees of freedom.
- Talking face degrees of freedom:
  - **Easy:** Translation (2) and rotation (1)
  - **Complicated:** Variability in movement while talking.

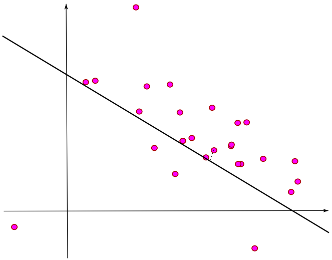


[1] All face data from Tim Cootes' talking face dataset,
http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Let's take a look at our iPython notebook

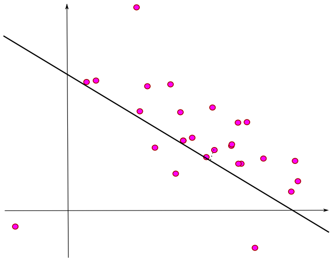
Principal component analysis (PCA)

- **Task of today:** Given a dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$, where d is potentially very large, find a low(er)-dimensional linear subspace $V \subset \mathbb{R}^d$ that is "very close to" the dataset.



Principal component analysis (PCA)

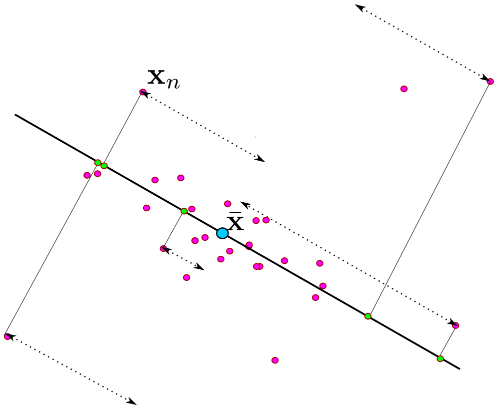
- **Task of today:** Given a dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$, where d is potentially very large, find a low(er)-dimensional linear subspace $V \subset \mathbb{R}^d$ that is "very close to" the dataset.
- **Question for you:** How would you do this / how would you make the question more precise?



Principal component analysis (PCA)

Possible formulation: Find the linear subspace $V \subset \mathbb{R}^d$ that *maximizes the variance* of the projected dataset

- **Question:** Is this V unique?
- Do you know how to find such a V?



PCA and eigenvalue decomposition of the covariance matrix

Theorem

Let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$ be a dataset. Let Σ be its covariance matrix, with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$ and corresponding eigenvectors $e_1, e_2, \ldots, e_d$.

The k-dimensional linear subspace V_k whose projected variance is maximized, is spanned by the eigenvectors $e_1, e_2, \ldots, e_k$.

PCA and eigenvalue decomposition of the covariance matrix

Theorem

Let $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^d$ be a dataset. Let Σ be its covariance matrix, with eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$ and corresponding eigenvectors $e_1, e_2, \ldots, e_d$.

The k-dimensional linear subspace V_k whose projected variance is maximized, is spanned by the eigenvectors $e_1, e_2, \ldots, e_k$.

Based on this, we call:

- The subspace spanned by e_1, the first principal component of the dataset (or PC1)
- The subspace spanned by e_2, the second principal component of the dataset (PC2)
- etc...

Computing PCA

The theorem on the previous slide gives an algorithm for computing PCA:

- Center the data:
  $\boldsymbol{x}_n \leftarrow \boldsymbol{x}_n - \bar{\boldsymbol{x}}$ for all $n = 1 \dots N$
- Compute the covariance matrix: $cov(X)$, where X is the data matrix (with centered data points $\boldsymbol{x}_n$)

$$X = \begin{pmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N \end{pmatrix}$$

- Compute the eigenvalues and eigenvectors of $cov(X)$
- Order eigenvalues and eigenvectors such that
  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$
- Return: The i^{th} PC is spanned by the eigenvector e_i.

Let's take a look at our iPython notebook

Variance captured by the principal components

- The principal components (eigenvectors of Σ) form a basis in which the covariance matrix is

$$\Sigma_{eig} = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_d \end{pmatrix}$$

- The diagonal elements of Σ_{eig} are the variances of the dataset projected onto the principal components.
- Note that for vectors, the sum of component variances is the variance of the vectors:
- Thus,

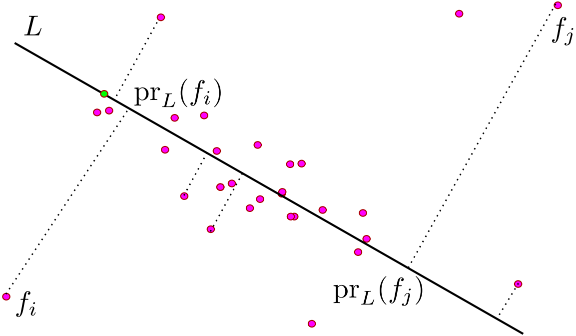
$$\sum_{i=1}^{k} \lambda_i$$

is the variance of the dataset projected onto $V_k = span(PC1, PC2, \dots, PCk)$, or the *variance captured* by the first k principal components.

Let's take a look at our iPython notebook

Example: Visualizing variance in face shape[2]

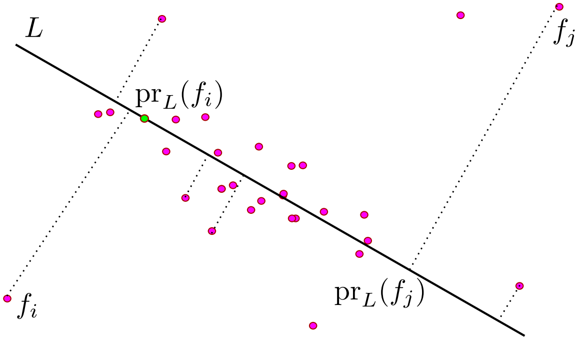
- Visualizing dataset variation along PCs
- The PCs approximate the face data – we expect to find faces along the first PCs!
- Pick samples (not datapoints) along the PC to visualize the variation



[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Visualizing variance in face shape[2]

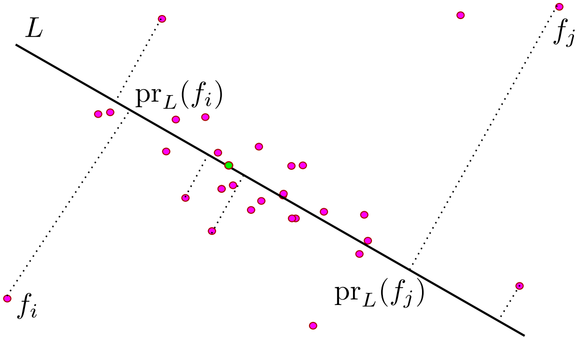
- Visualizing dataset variation along PCs
- The PCs approximate the face data – we expect to find faces along the first PCs!
- Pick samples (not datapoints) along the PC to visualize the variation



[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Visualizing variance in face shape[2]

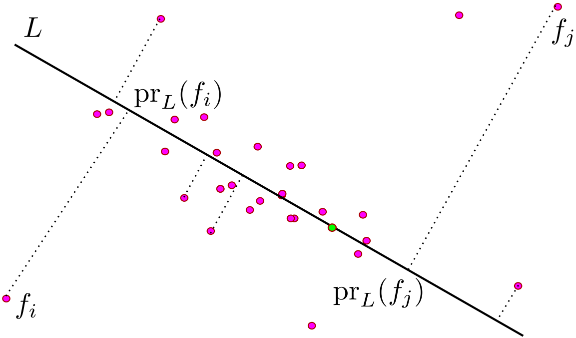
- Visualizing dataset variation along PCs
- The PCs approximate the face data – we expect to find faces along the first PCs!
- Pick samples (not datapoints) along the PC to visualize the variation



[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Visualizing variance in face shape[2]

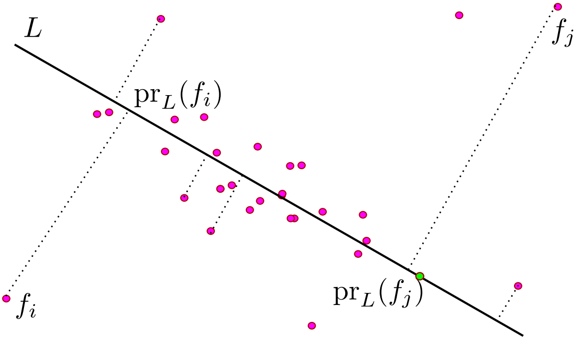
- Visualizing dataset variation along PCs
- The PCs approximate the face data – we expect to find faces along the first PCs!
- Pick samples (not datapoints) along the PC to visualize the variation



[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Visualizing variance in face shape[2]

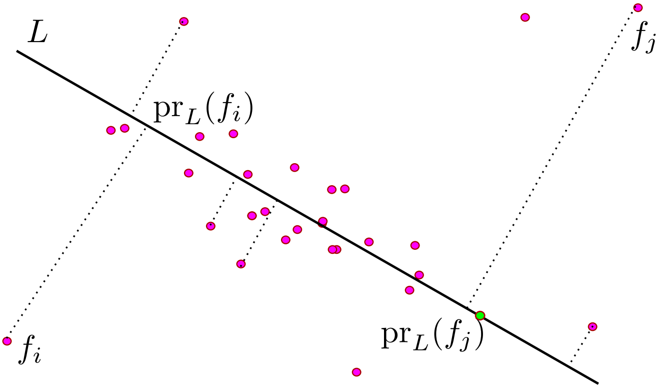
- Visualizing dataset variation along PCs
- The PCs approximate the face data – we expect to find faces along the first PCs!
- Pick samples (not datapoints) along the PC to visualize the variation



[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Visualizing variance in face shape[2]

Sampling the PC densely, we get a movie of the shape variation along PC1, PC2, etc



[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Visualizing variance in face shape[2]

Let's take a look at our iPython notebook

[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Visualizing variance in face shape[2]

- Visualizing through plots and playing the videos, what do you see?

[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Visualizing variance in face shape[2]

- Visualizing through plots and playing the videos, what do you see?
- If you want to capture emotion or make an automatic speech recognition system based on mouth movement, how does PCA help you?

[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Example: Visualizing variance in face shape[2]

- Visualizing through plots and playing the videos, what do you see?
- If you want to capture emotion or make an automatic speech recognition system based on mouth movement, how does PCA help you?
- The first two PCs captured 95% of the variation. What does that tell you about interesting variation?

[2] All face data from Tim Cootes' talking face dataset, http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html

Dimensionality reduction

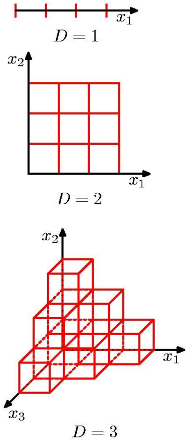
- PCA is an example of *dimensionality reduction*
- Dimensionality reduction refers to the process of reducing the dimensionality in your data representation.
- More precisely: Given a dataset $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^{d_1}$, finding a representation of your dataset

$$\{\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_N)\} \subset \mathbb{R}^{d_2}$$

  where $d_2 < d_1$, and where you retain the properties of your dataset as well as possible.
- Why is this useful?

The curse of dimensionality[3]

- In order to sample the interval $[0, 1]$ with density 0.1, I need 10 points.
- In order to sample the cube $[0, 1] \times [0, 1]$ with the same density, I need 100 points.
- etc
- The more dimensions, the more data you need for drawing conclusions.



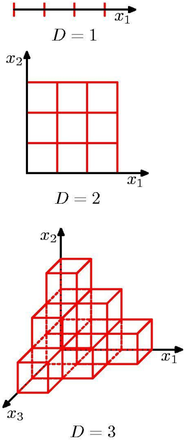
[3] Figure from Bishop: Pattern Recognition and Machine Learning

The curse of dimensionality[3]

- Consider the d-cube $[-1, 1]^d$.
- The distance from the center to a corner is

$$\sqrt{d} \to \infty \text{ as } d \to \infty$$

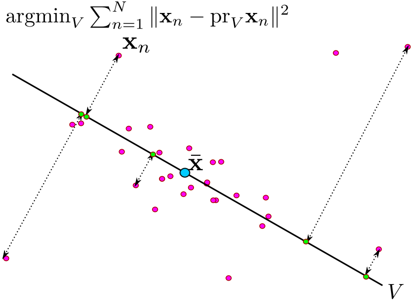
- When d gets large, everything gets large – including noise effects!
- By extracting the essential dimensions, we can avoid using unnecessary dimensions.



[3] Figure from Bishop: Pattern Recognition and Machine Learning

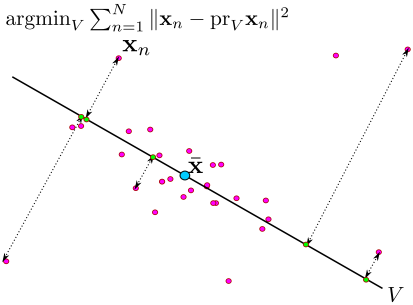
Equivalence of projection error minimization / projected variance maximation

- PCA equivalently formulated as minimizing squared projection error
- A least squares problem
- Equivalent to variance maximization *up to projection* (**why?**)



Equivalence of projection error minimization / projected variance maximation

- PCA equivalently formulated as minimizing squared projection error
- A least squares problem
- Equivalent to variance maximization *up to projection* (**why?**)
- For equivalence: Ask variance maximizing subspace to pass through the mean



Note

- The book uses a different formulation of PCA through optimization (gradient descent)
- We will discuss this, its pros, cons and applicability, next Thursday

Next lectures:

- Tuesday: kNN with Christian
- Thursday: Dimensionality reduction 2
  - PCA and visualization of global dataset structure
  - PCA through optimization (gradient descent)
  - Different formulations of PCA, and corresponding insight