Principal Component Analysis

Nuno Vasconcelos

(Ken Kreutz-Delgado)

UCSD

Curse of dimensionality

- Typical observation in Bayes decision theory:
  - **Error increases when number of features is large**
- Even for simple models (e.g. Gaussian) we need a large number of examples *n* to have good estimates
- **Q:** what does “large” mean? This depends on the dimension of the space
- The best way to see this is to think of an histogram
  - suppose you have 100 points and you need at least 10 bins per axis in order to get a reasonable quantization

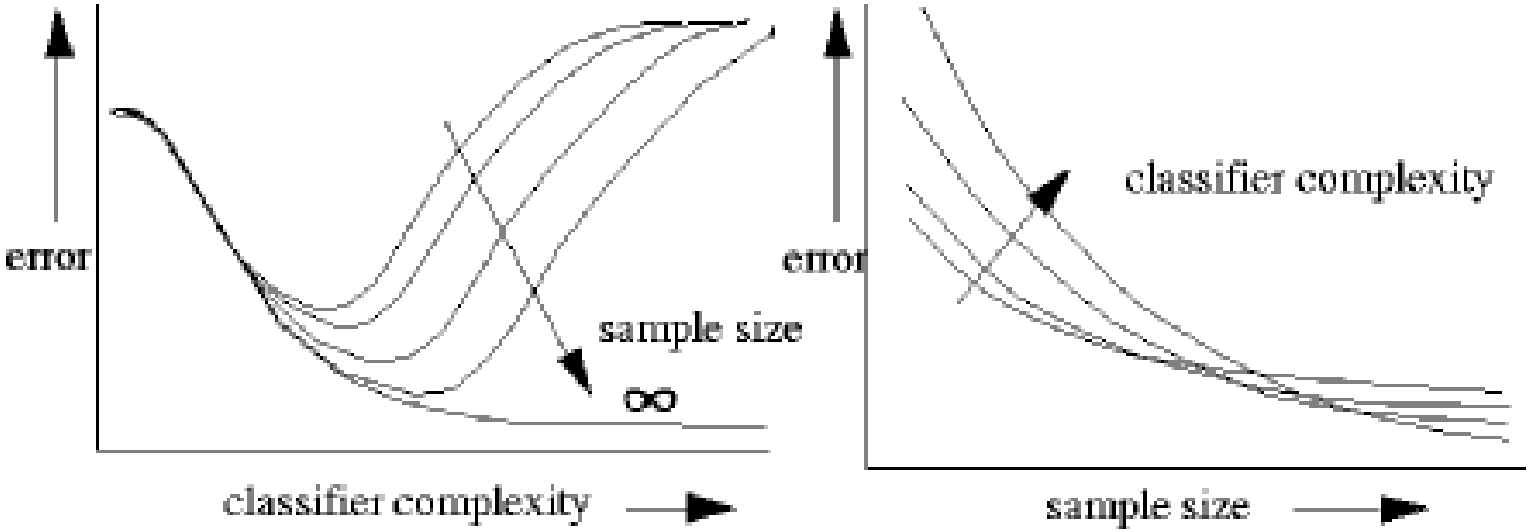
for uniform data you get, on average,

dimension	1	2	3
points/bin	10	1	0.1

which is decent in1D, bad in 2D, terrible in 3D
(9 out of each10 bins are empty!)

Curse of Dimensionality

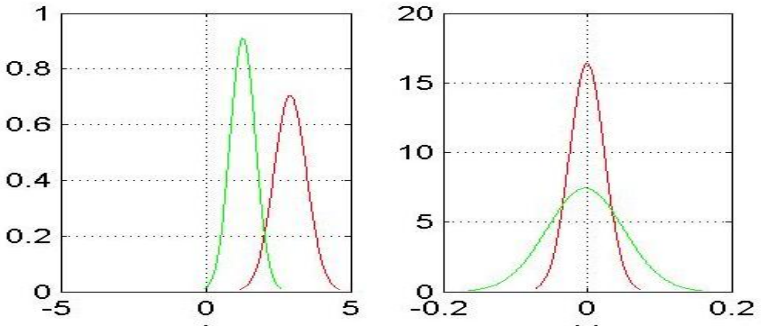
- This is the **curse of dimensionality:**
  - For a **given classifier** the **number of examples required to maintain classification accuracy increases exponentially with the dimension of the feature space**
- In higher dimensions the classifier has more parameters
  - **Therefore: Higher complexity & Harder to learn**



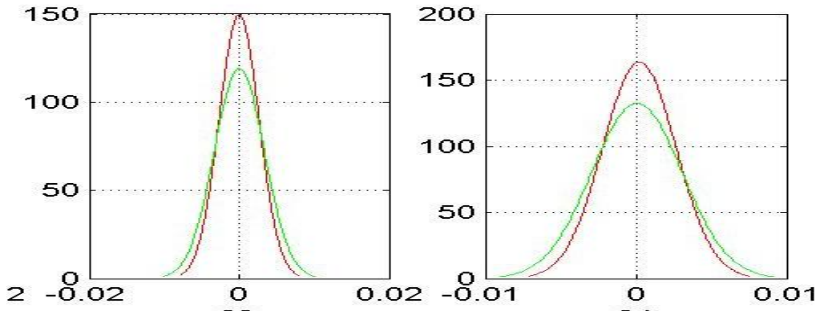
Dimensionality Reduction

- What do we do about this? **Avoid unnecessary dimensions**
- "Unnecessary" features arise in two ways:
  1. features are not discriminant
  2. features are not independent (are highly correlated)
- **Non-discriminant** means that they do not separate the classes well

discriminant

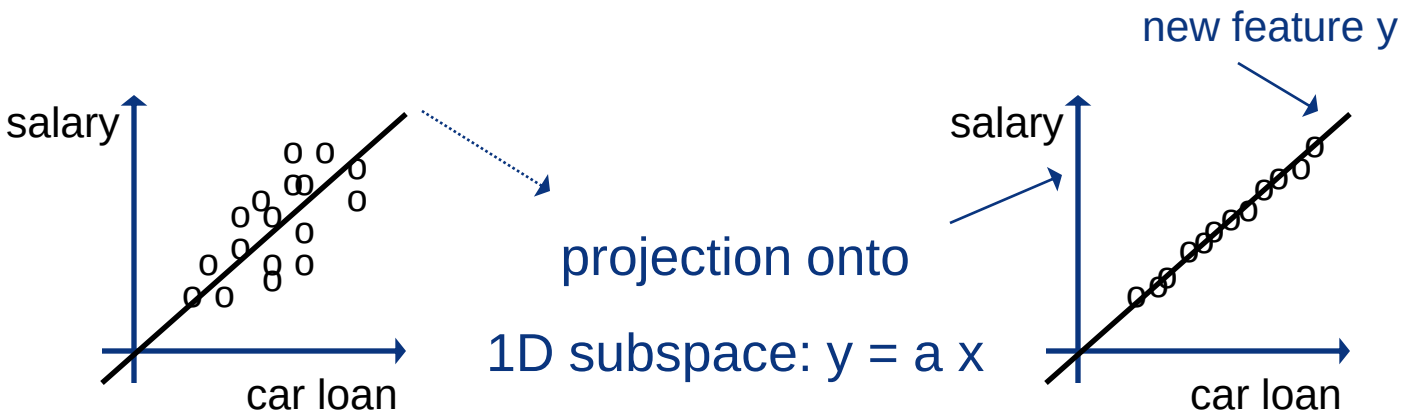


non-discriminant



Dimensionality Reduction

- **Q:** How do we detect the presence of feature correlations?
- **A:** The data “lives” in a low dimensional subspace (up to some amounts of noise). E.g.

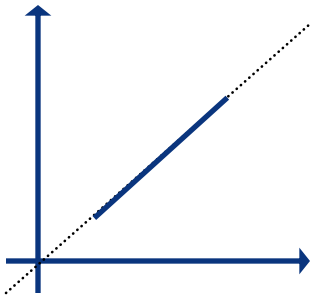


- In the example above we have a 3D hyper-plane in 5D
- If we can find this hyper-plane we can:
  - Project the data onto it
  - Get rid of two dimensions without introducing significant error

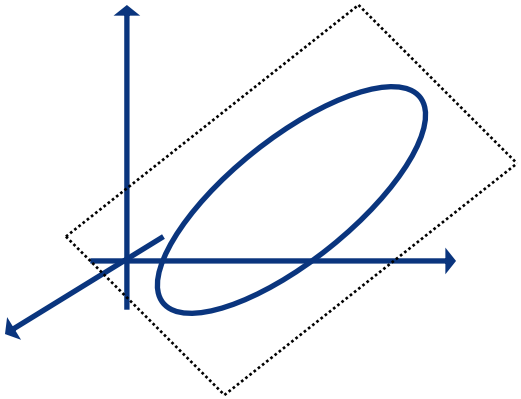
Principal Components

- **Basic idea:**
  - If the data lives in a (lower dimensional) subspace, it is going to look very flat when viewed from the full space, e.g.

1D subspace in 2D



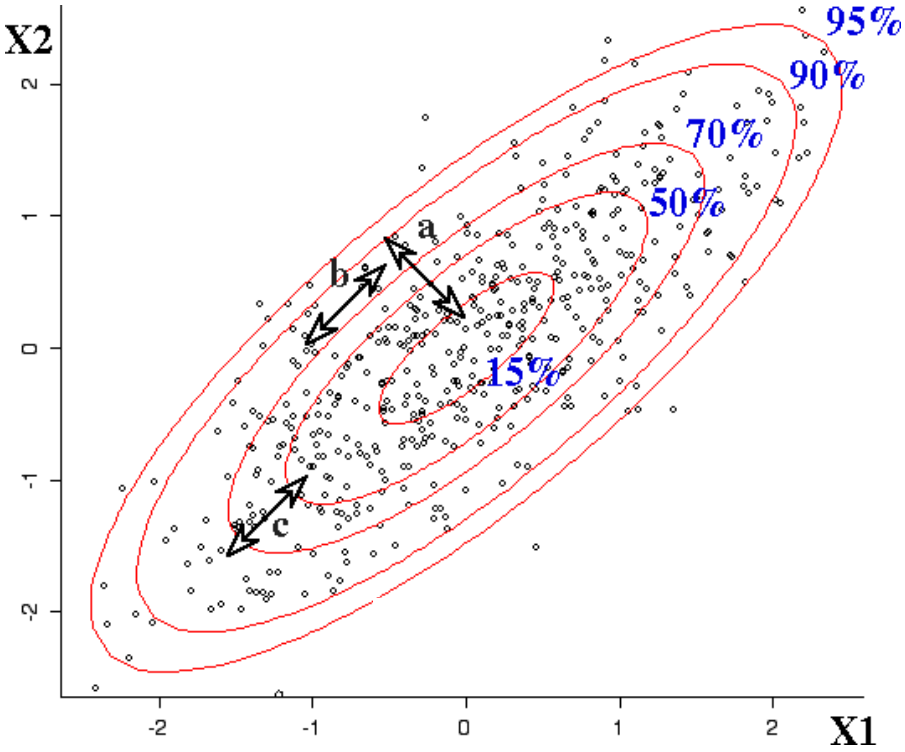
2D subspace in 3D



- This means that:
  - **If we fit a Gaussian to the data the iso-probability contours are going to be highly skewed ellipsoids**
  - **The directions that explain most of the variance in the fitted data give the Principle Components of the data.**

Principal Components

- **How do we find these ellipsoids?**
- When we talked about metrics we said that the
  - **Mahalanobis distance** measures the "natural" units for the problem because it is "adapted" to the covariance of the data
- We also know that
  - What is special about it is that **it uses Σ^{-1}**
- Hence, information about possible subspace structure must be in the covariance matrix Σ



$$d^2(x,\mu) = (x-\mu)^T \Sigma^{-1} (x-\mu)$$

Multivariate Gaussian Review

- The **equiprobability contours** (level sets) of a Gaussian are the points such that

$$(x-\mu)^T\Sigma^{-1}(x-\mu) = K$$

- Let's consider the **change of variable *z* = *x*-μ**, which only moves the origin by μ. The equation

$$z^T\Sigma^{-1}z = K$$

is **the equation of an ellipse** (a hyperellipse).

- This is **easy to see when Σ is diagonal:**

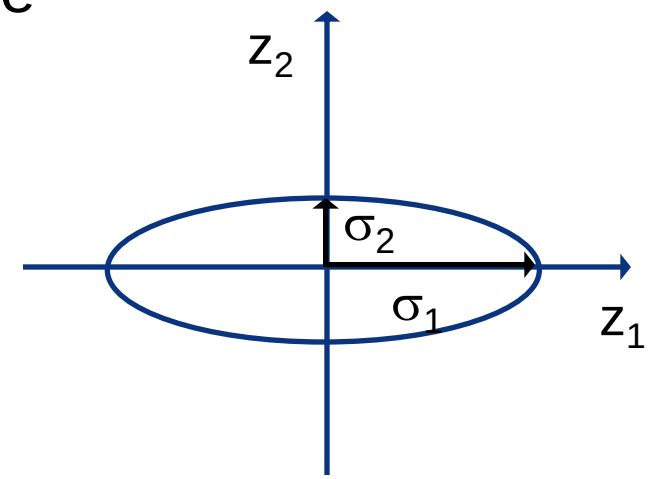
$$\Sigma = \Lambda = diag(\sigma_1^2,\ldots,\sigma_d^2) \Rightarrow z^T\Sigma^{-1}z = \sum_i \frac{z_i^2}{\sigma_i^2} = K$$

Gaussian Review

- This is the equation of an **ellipse with principal lengths σ_i**
  - E.g. when *d = 2*

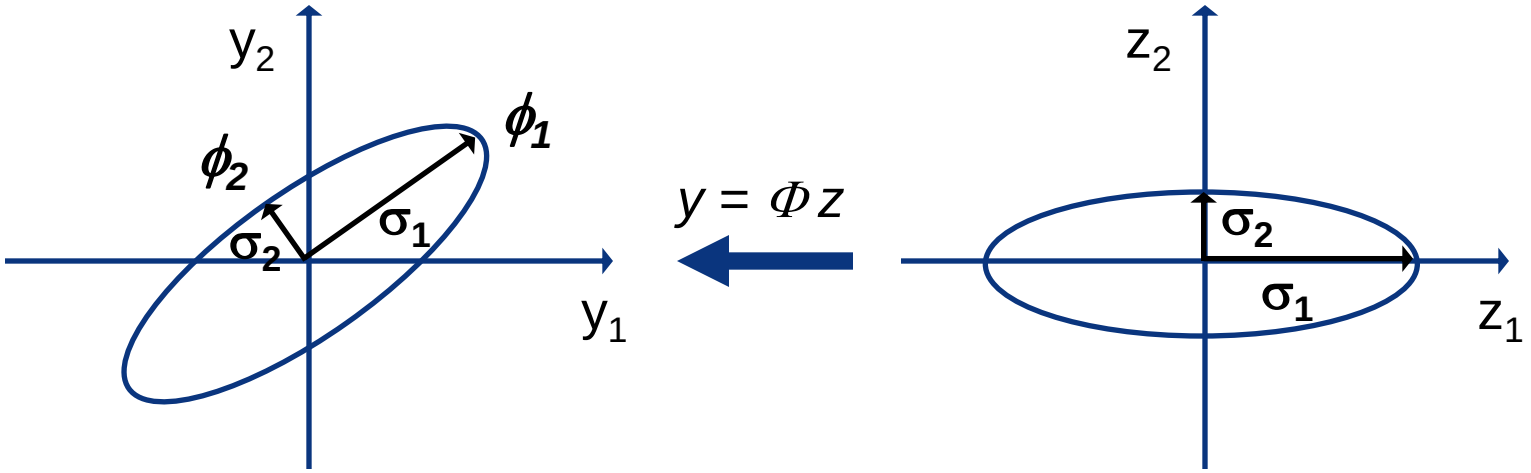
$$\frac{z_1^2}{\sigma_1^2} + \frac{z_2^2}{\sigma_2^2} = 1$$

is the ellipse



Gaussian Review

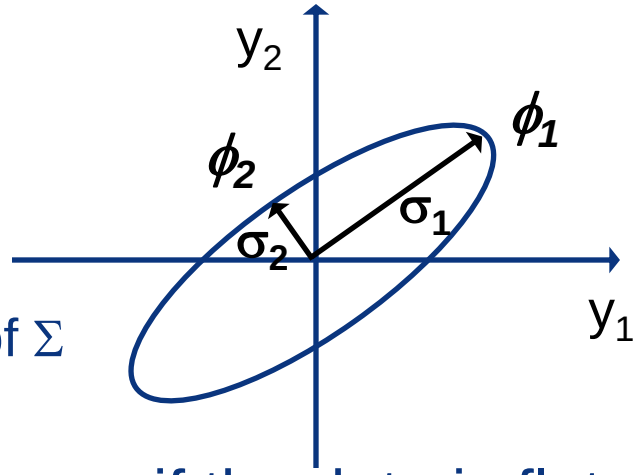
- Introduce a **transformation $y = \Phi z$**
- Then y has covariance $\Sigma_y = \Phi \Sigma_z \Phi^T = \Phi \Lambda \Phi^T$
- If Φ is proper orthogonal this is just a rotation and we have



- We obtain **a rotated ellipse** with principal components ϕ_1 and ϕ_2 which are the columns of Φ
- Note that $\Sigma_y = \Phi \Lambda \Phi^T$ is the eigendecomposition of Σ_y

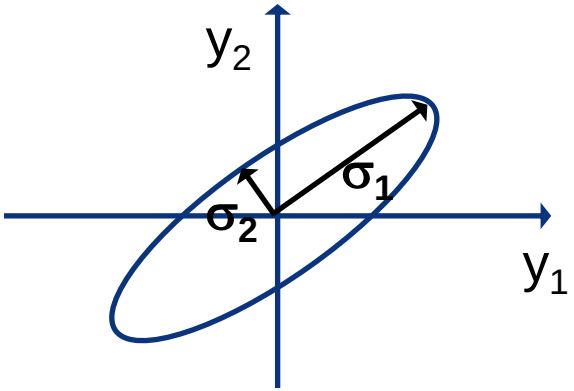
Principal Component Analysis (PCA)

- If y is Gaussian with covariance Σ, the **equiprobability contours** are the **ellipses** whose
  - **Principal Components** ϕ_i are the **eigenvectors** of Σ
  - **Principal Values** (lengths) σ_i are the **square roots of the eigenvalues** λ_i of Σ

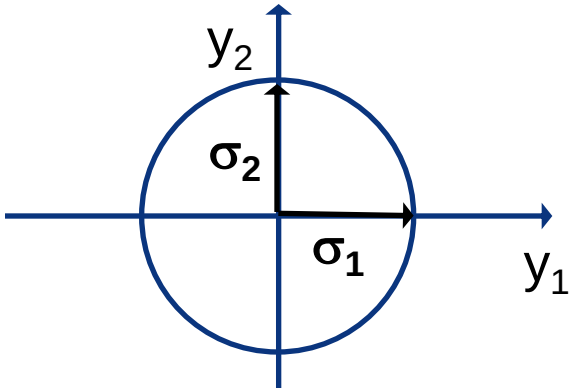


- By computing the eigenvalues we know if the data is flat

$\sigma_1 >> \sigma_2$: flat



$\sigma_1 = \sigma_2$: not flat



Learning-based PCA

- Given sample $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $x_i \in \mathcal{R}^d$

  - compute sample mean: $\hat{\mu} = \frac{1}{n}\sum_i(\mathbf{x}_i)$
  - compute sample covariance: $\hat{\Sigma} = \frac{1}{n}\sum_i(\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T$
  - compute eigenvalues and eigenvectors of $\hat{\Sigma}$

    $$\hat{\Sigma} = \Phi\Lambda\Phi^T, \ \Lambda = diag(\sigma_1^2, \ldots, \sigma_n^2) \ \ \Phi^T\Phi = I$$

  - order eigenvalues $\sigma_1^2 > \ldots > \sigma_n^2$
  - if, for a certain k, $\sigma_k << \sigma_1$ eliminate the eigenvalues and eigenvectors above k.

Learning-based PCA

- Given principal compoenents $\phi_i, i \in 1, \ldots, k$ and a test sample $\mathcal{T} = \{\mathbf{t}_1, \ldots, \mathbf{t}_n\}, \ t_i \in \mathcal{R}^d$

  - subtract mean to each point $\mathbf{t}'_i = \mathbf{t}_i - \hat{\mu}$

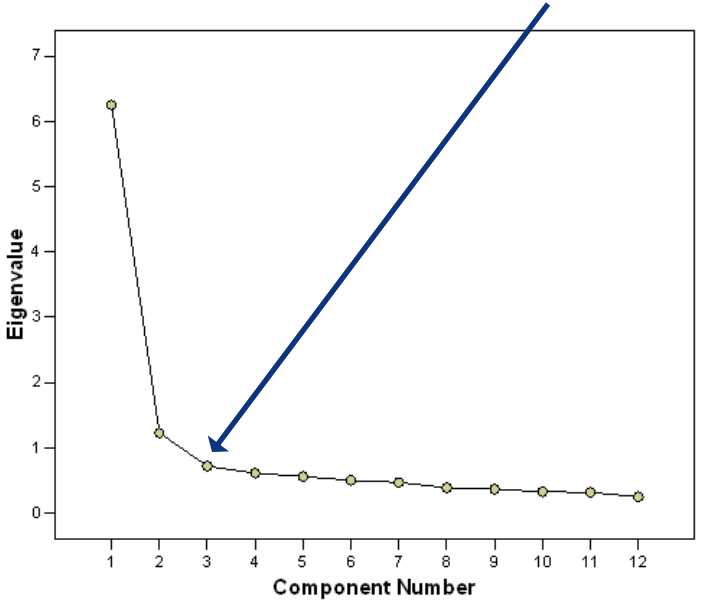
  - project onto eigenvector space $\mathbf{y}_i = \mathbf{A}\mathbf{t}'_i$ where

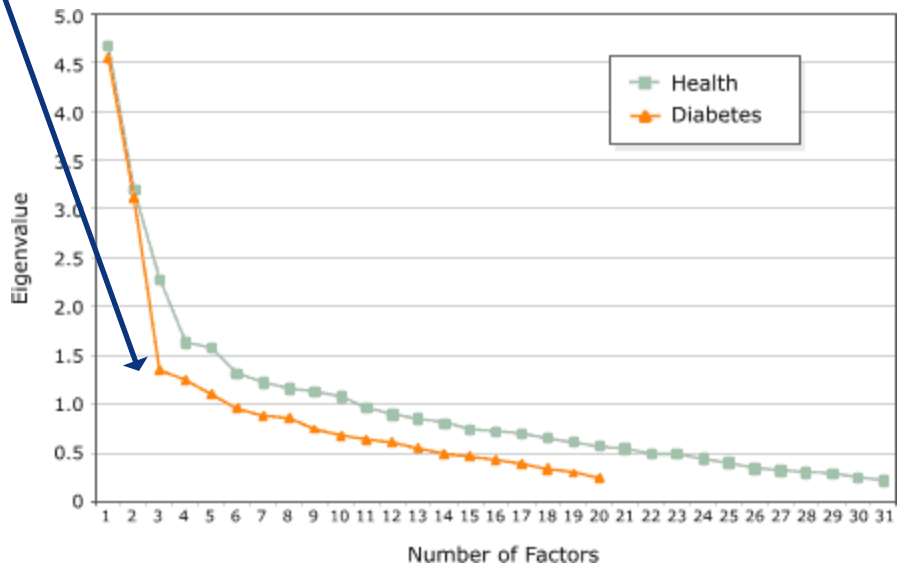
$$\mathbf{A} = \begin{bmatrix} \phi_1^T \\ \vdots \\ \phi_k^T \end{bmatrix}$$

  - use $\mathcal{T}' = \{\mathbf{y}_1, \ldots \mathbf{y}_n\}$ to estimate class conditional densities and do all further processing on $\mathbf{y}$.

Principal Component Analysis

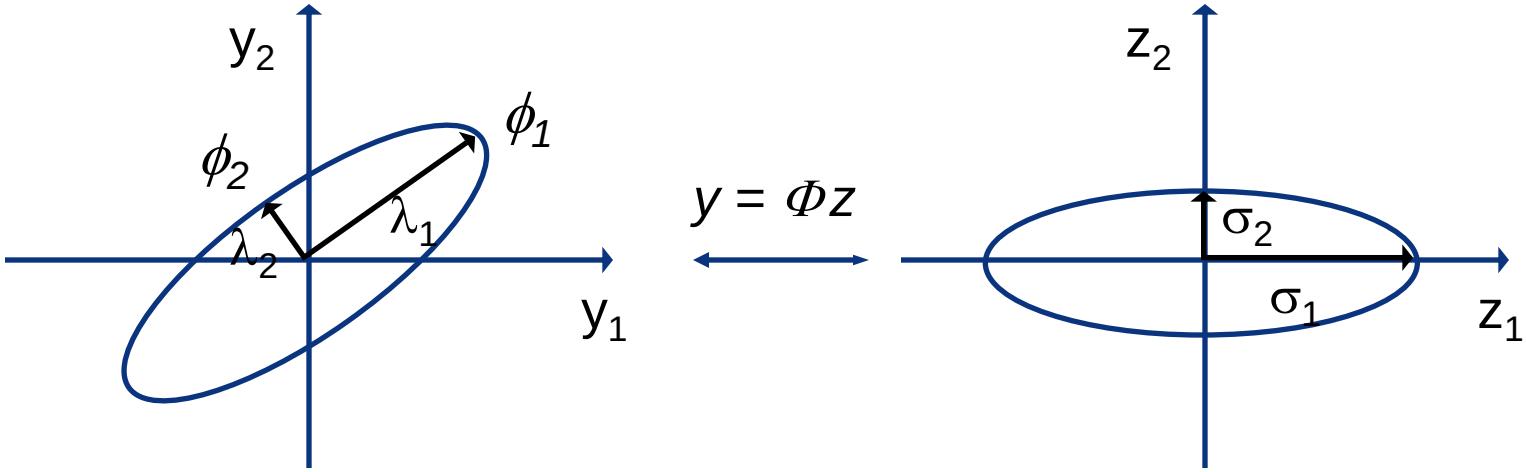
- How to **determine the number of eigenvectors** to keep?
  One possibility is to **plot eigenvalue magnitudes**
  - This is called a **Scree Plot**
  - Usually there is a fast decrease in the eigenvalue magnitude followed by a flat area
  - One good choice is the knee of this curve





Principal Component Analysis

- Another possibility: **Percentage of Explained Variance**
  - Remember that **eigenvalues are a measure of variance along the principle directions (eigenvectors)**

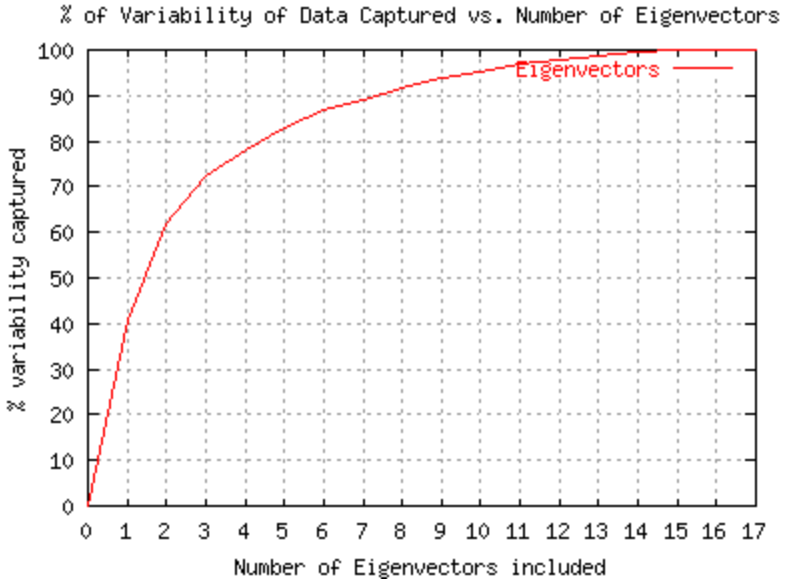


  - Ratio r_k measures **% of total variance** contained in the **top k eigenvalues**
  - Measure of the **fraction of data variability along the associated eigenvectors**

$$r_k = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2}$$

Principal Component Analysis

- Given r_k a natural measure is to pick the eigenvectors that explain p% of the data variability
  - This can be done **by plotting the ratio r_k as a function of k**



$$r_k = \frac{\sum_{i=1}^{k} \sigma_i^2}{\sum_{i=1}^{n} \sigma_i^2}$$

- **E.g. we need 3 eigenvectors to cover 70% of the variability of this dataset**

PCA by SVD

- There is an **alternative way** to compute the principal components, based on the singular value decomposition
- **("Condensed") Singular Value Decomposition (SVD):**
  - **Any** full-rank ***n*** x ***m*** matrix (***n*** >***m***) can be decomposed as

$$A = \mathbf{M}\,\Pi\,\mathbf{N}^T$$

  - **M** is a ***n*** x ***m*** (nonsquare) column orthogonal matrix of **left singular vectors** (columns of **M**)
  - Π is an ***m*** x ***m*** (square) diagonal matrix containing the ***m*** **singular values** (which are nonzero and strictly positive)
  - **N** an ***m*** x ***m*** row orthogonal matrix of **right singular vectors** (columns of **N** = rows of $\mathbf{N}^T$)

$$\mathbf{M}^T\mathbf{M} = I_{m\times m} \qquad \mathbf{N}^T\mathbf{N} = \mathbf{N}\mathbf{N}^T = I_{m\times m}$$

PCA by SVD

- To relate this to PCA, we construct the ***d* x *n* Data Matrix**

$$X = \begin{bmatrix} | & & | \\ x_1 & \dots & x_n \\ | & & | \end{bmatrix}$$

- The **sample mean** is

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{1}{n}\begin{bmatrix} | & & | \\ x_1 & \dots & x_n \\ | & & | \end{bmatrix}\begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \frac{1}{n} X 1$$

PCA by SVD

- We **center the data** by subtracting the mean from each column of **X**
- This yields the ***d* x *n* Centered Data Matrix**

$$X_c = \begin{bmatrix} | & & | \\ x_1 & \cdots & x_n \\ | & & | \end{bmatrix} - \begin{bmatrix} | & & | \\ \mu & \cdots & \mu \\ | & & | \end{bmatrix}$$

$$= X - \mu 1^T = X - \frac{1}{n} X 1 1^T = X\left(I - \frac{1}{n} 1 1^T\right)$$

PCA by SVD

- The Sample Covariance is the *d* x *d* matrix

$$\Sigma = \frac{1}{n}\sum_i \left(x_i - \mu\right)\left(x_i - \mu\right)^T = \frac{1}{n}\sum_i x_i^c \left(x_i^c\right)^T$$

where x_i^c is the i^{th} column of X_c

- This can be written as

$$\Sigma = \frac{1}{n}\begin{bmatrix} | & & | \\ x_1^c & \cdots & x_n^c \\ | & & | \end{bmatrix}\begin{bmatrix} - & x_1^c & - \\ & \vdots & \\ - & x_n^c & - \end{bmatrix} = \frac{1}{n} X_c X_c^T$$

PCA by SVD

- The centered data matrix

$$X_c^T = \begin{bmatrix} - & x_1^c & - \\ & \vdots & \\ - & x_n^c & - \end{bmatrix}$$

is *n* x *d*. Assuming it has rank = *d*, it has the SVD:

$$X_c^T = \mathrm{M}\Pi\mathrm{N}^T$$

$$\mathrm{M}^T\mathrm{M} = I \qquad \mathrm{N}^T\mathrm{N} = I$$

- This yields:

$$\Sigma = \frac{1}{n} X_c X_c^T = \frac{1}{n} \mathrm{N}\Pi\mathrm{M}^T\mathrm{M}\Pi\mathrm{N}^T = \frac{1}{n} \mathrm{N}\Pi^2\mathrm{N}^T$$

PCA by SVD

$$\Sigma = \mathbf{N}\left(\frac{1}{n}\Pi^2\right)\mathbf{N}^T$$

- Noting that **N** is ***d*** x ***d*** and orthonormal, and Π^2 diagonal, shows that this is just the eigenvalue decomposition of Σ
- It follows that
  - **The eigenvectors of Σ are the columns of N**
  - **The eigenvalues of Σ are**

$$\lambda_i = \sigma_i^2 = \frac{\pi_i^2}{n}$$

- **This gives an alternative algorithm for PCA**

PCA by SVD

Summary of Computation of PCA by SVD:

- Given $\boldsymbol{X}$ with one example per column
  - **1) Create the (transposed) Centered Data-Matrix:**

$$\boldsymbol{X}_c^T = \left(\boldsymbol{I} - \frac{1}{\boldsymbol{n}} \mathbf{1}\mathbf{1}^T \right) \boldsymbol{X}^T$$

  - **2) Compute its SVD:**

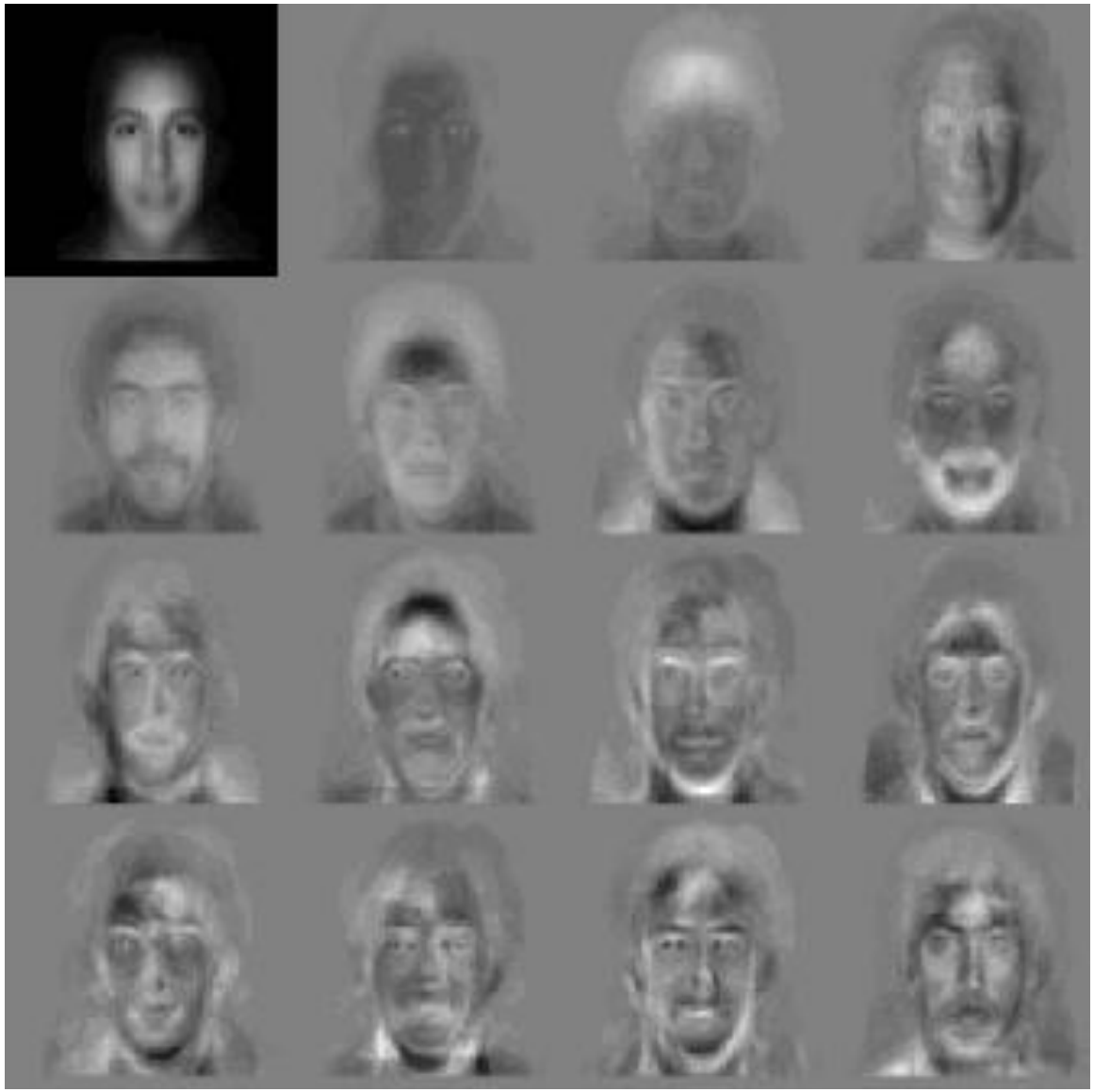
$$\boldsymbol{X}_c^T = \mathbf{M}\boldsymbol{\Pi}\mathbf{N}^T$$

  - **3) Principal Components are columns of N; Principle Values are:**

$$\sigma_i = \sqrt{\lambda_i} = \frac{\pi_i}{\sqrt{n}}$$

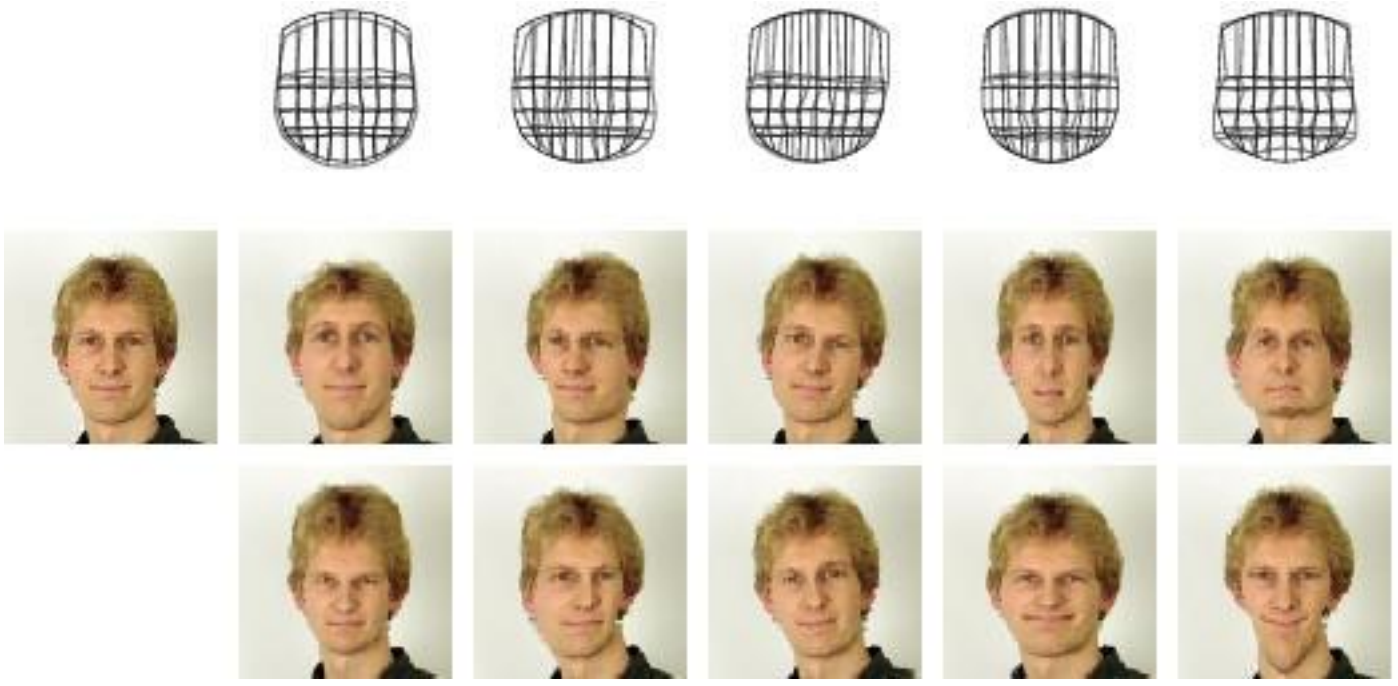
Principal Component Analysis

- Principal components are often quite **informative about the structure of the data**
- **Example:**
  - **Eigenfaces,** the principal components for the space of images of faces
  - The figure only show the first 16 eigenvectors (eigenfaces)
  - Note lighting, structure, etc



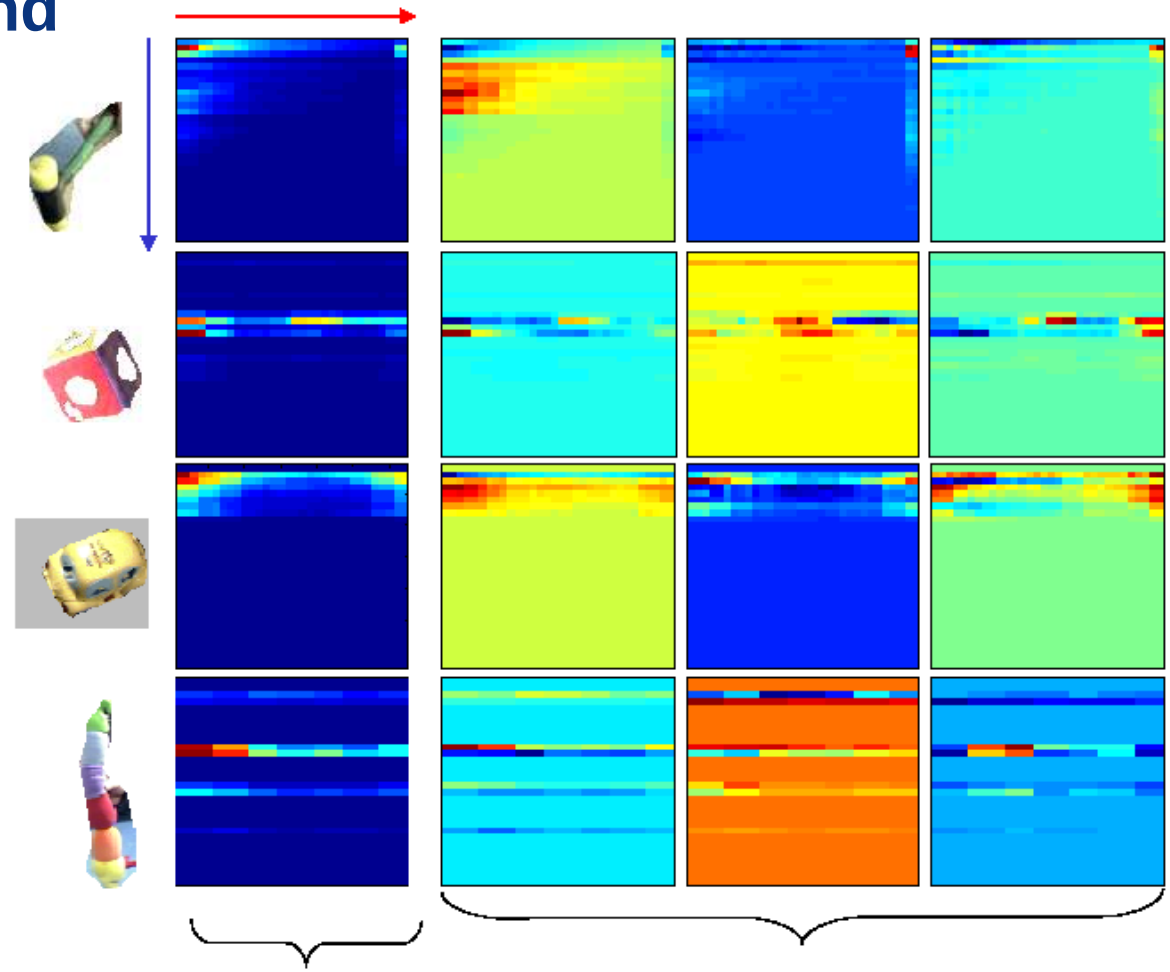
Principal Components Analysis

- PCA has been applied to virtually all learning problems
- E.g. **eigenshapes for face morphing**



Principal Component Analysis

- Sound



average sound images

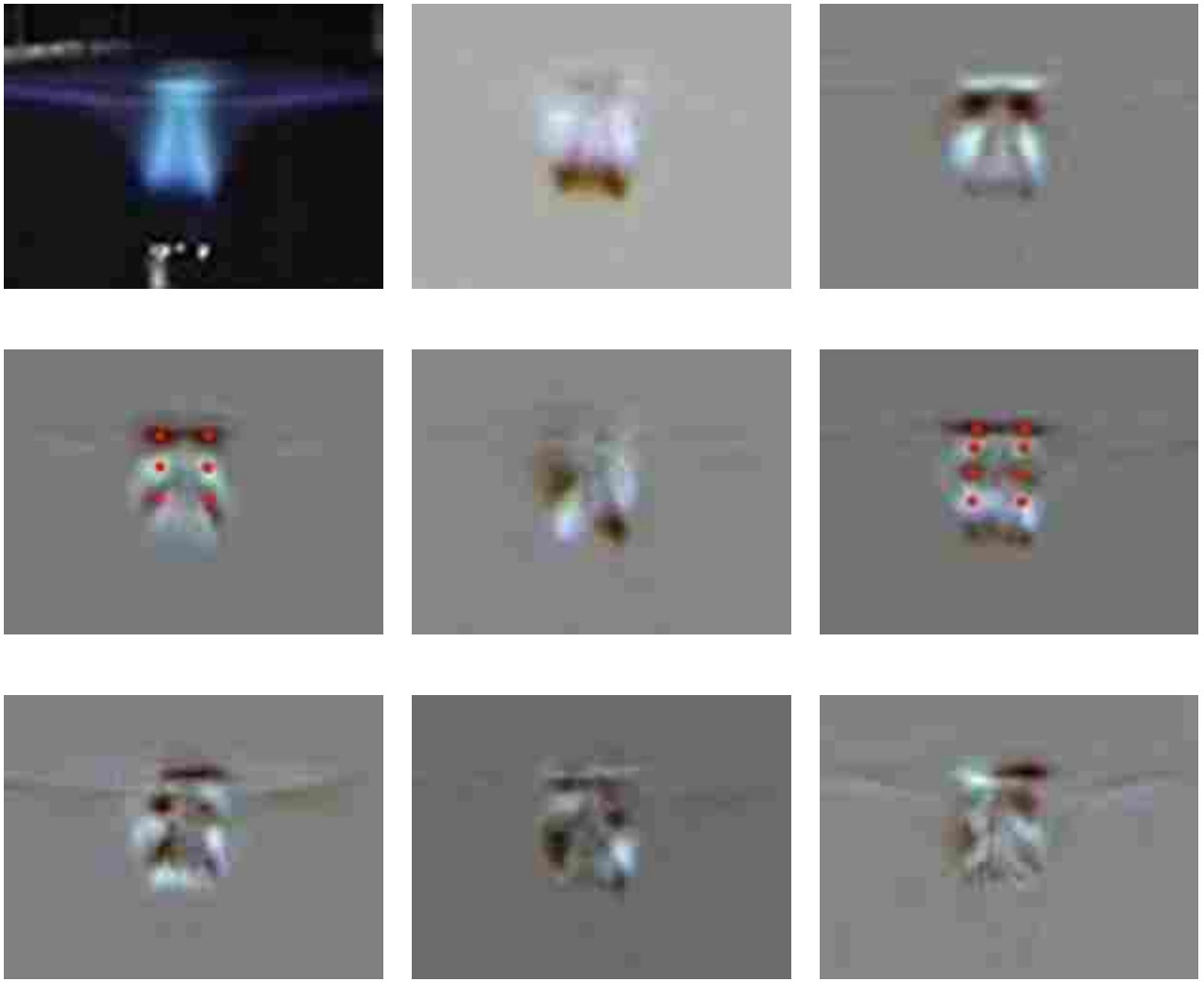
Eigensounds corresponding to the three highest eigenvalues

Principal Component Analysis

- Turbulence

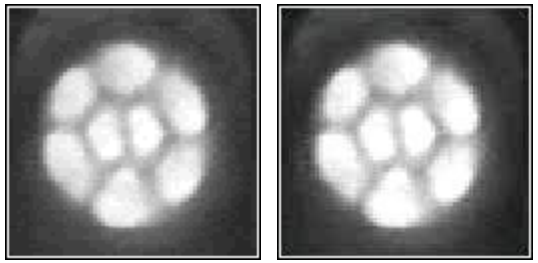
Flames

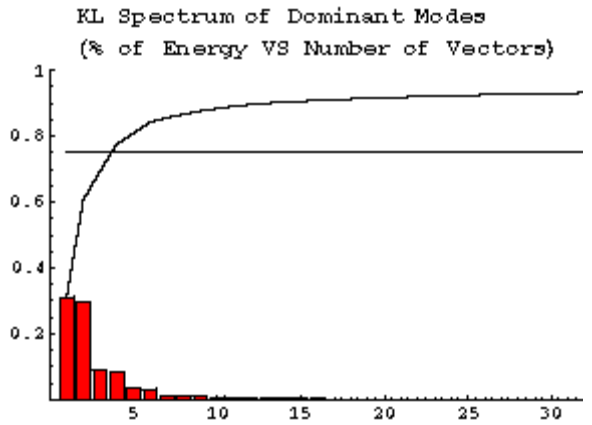
Eigenflames



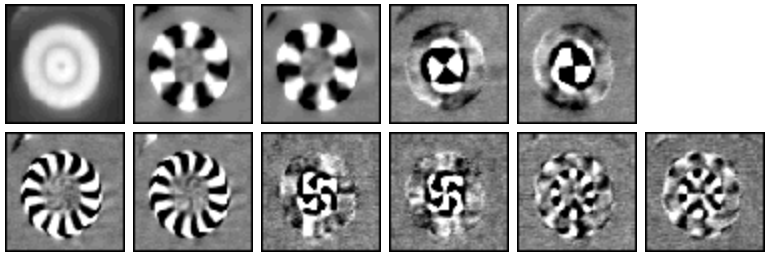
Principal Component Analysis

- Video

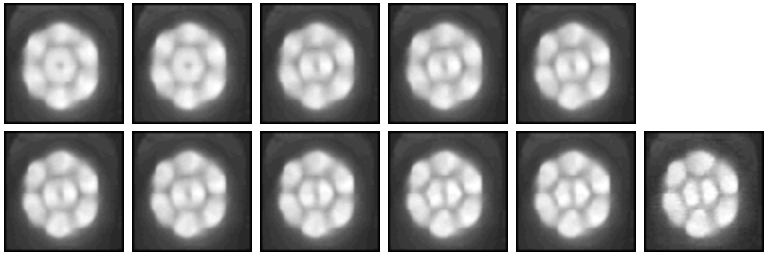


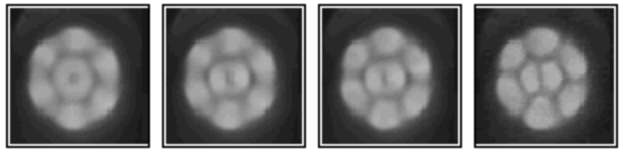


Eigenrings



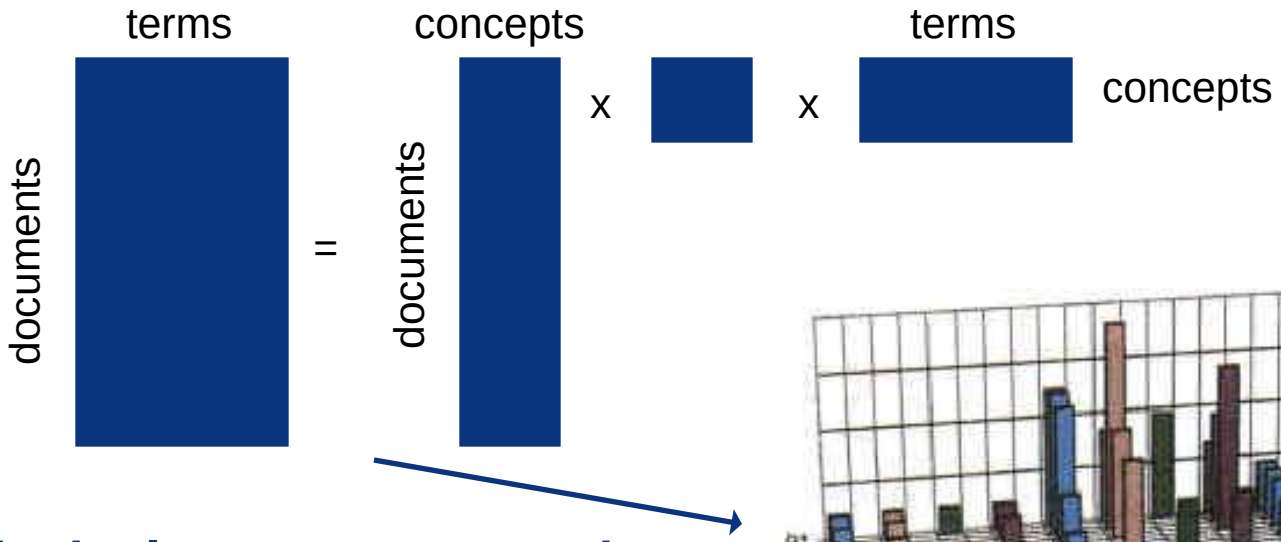
reconstruction

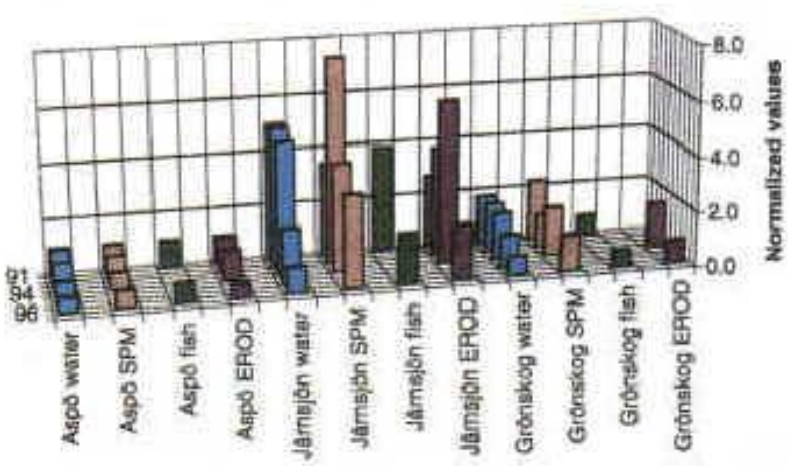




Principal Component Analysis

- **Text: Latent Semantic Indexing**
  - Represent each document by a word histogram
  - **Perform SVD on the *document* x *word* matrix**





  - **Principal components as the directions of semantic concepts**

Latent Semantic Analysis

- **Applications:**
  - document classification, information
- **Goal: solve two fundamental problems in language**
  - **Synonymy:** different writers use different words to describe the same idea.
  - **Polysemy:** the same word can have multiple meanings
- **Reasons:**
  - **Original term-document matrix is too large** for the computing resources
  - **Original term-document matrix is *noisy*:** for instance, anecdotal instances of terms are to be eliminated.
  - **Original term-document matrix overly sparse** relative to "true" term-document matrix. E.g. lists only words actually *in* each document, whereas we might be interested in all words *related to* each document-- much larger set due to synonymy

Latent Semantic Analysis

- **After PCA some dimensions get "merged":**
  - {(car), (truck), (flower)} --> {(1.3452 * car + 0.2828 * truck), (flower)}
- This **mitigates synonymy**,
  - Merges the dimensions associated with terms that have similar meanings.
- And **mitigates polysemy**,
  - Components of polysemous words that point in the "right" direction are added to the components of words that share this sense.
  - Conversely, components that point in other directions tend to either simply cancel out, or, at worst, to be smaller than components in the directions corresponding to the intended sense.

Extensions

- Soon we will talk about **kernels**
  - It turns out that **any algorithm which depends on the data through dot-products only**, i.e. the matrix of elements

$$x_i^T x_j$$

    can be kernelized
  - This is usually beneficial, we will see why later
  - For now we look at the question of whether PCA can be written in the inner product form mentioned above

- Recall the **data matrix** is

$$X = \begin{bmatrix} | & & | \\ x_1 & \dots & x_n \\ | & & | \end{bmatrix}$$

Extensions

- **Recall** the centered data matrix, covariance, and SVD:

$$X_c = X\left(I - \frac{1}{n}11^T\right) \qquad \Sigma = \frac{1}{n}X_c X_c^T \qquad X_c^T = \mathbf{M}\mathbf{\Pi}\mathbf{N}^T$$

- **This yields:**

$$X_c^T X_c = \mathbf{M}\mathbf{\Pi}^2\mathbf{M}^T, \quad \Phi = \mathbf{N} = X_c\mathbf{M}\mathbf{\Pi}^{-1}, \quad \Lambda = \frac{1}{n}\mathbf{\Pi}^2$$

- **Hence, solving for the *d* positive (nonzero) eigenvalues of the inner product matrix $X_c^T X_c$, and for their associated eigenvectors, provides an alternative way to compute the eigendecomposition of the sample covariance matrix needed to perform an SVD.**

Extensions

- In summary, we have

$$\Sigma = \Phi \Lambda \Phi^T \qquad \Phi = X_c \mathbf{M} \Pi^{-1}$$

$$\frac{1}{n} X_c^T X_c = \mathbf{M} \left(\frac{1}{n} \Pi^2 \right) \mathbf{M}^T = \mathbf{M} \Lambda \mathbf{M}^T$$

- This means that **we can obtain PCA** by
  - **1) Assembling the inner-product matrix $X_c^T X_c$**
  - **2) Computing its eigendecomposition $(\Pi^2, \mathbf{M})$**
- PCA
  - The **principal components** are then given by $\Phi = \mathbf{X_c M} \Pi^{-1}$
  - The **eigenvalues** are given by $\Lambda = (1/\mathbf{n}) \Pi^2$

Extensions

- What is interesting here is that **we only need the matrix**

$$K_c = X_c^T X_c = \begin{bmatrix} - & x_1^c & - \\ & \vdots & \\ - & x_n^c & - \end{bmatrix} \begin{bmatrix} | & & | \\ x_1^c & \cdots & x_n^c \\ | & & | \end{bmatrix}$$

$$= \begin{bmatrix} & \vdots & \\ \cdots & \left(x_n^c\right)^T x_n^c & \cdots \\ & \vdots & \end{bmatrix}$$

- This is the **inner product matrix** of "dot-products" of the centered data-points
- **Notice that you don't need the points themselves**, **only their dot-products (similarities)**

Extensions

- **In summary, to get PCA**
  - **1) Compute the dot-product matrix $K_c = X_c^T X_c$**
  - **2) Compute its eigendecomposition** (Π^2, M)
- **PCA:** For a covariance matrix $\Sigma = \Phi \Lambda \Phi^T$
  - **Principal Components** are given by $\Phi = \mathbf{X_c} \mathrm{M}\, \Pi^{-1}$
  - **Eigenvalues** are given by $\Lambda = (1/\mathbf{n})\Pi^2$
  - **Projection of the centered data-points onto the principal components** is given by

$$X_c^T \Phi = X_c^T X_c \mathbf{M} \Pi^{-1} = K_c \mathbf{M} \Pi^{-1}$$

- This allows the computation of the eigenvalues and PCA coefficients when we only have access to the dot-product (inner product) matrix K_c

END