Dimensionality Reduction and Principal Components

Nuno Vasconcelos

(Ken Kreutz-Delgado)

UCSD

Motivation

- Recall, in Bayesian decision theory we have:
  - World: States Y in {1, ..., M} and observations of X
  - Class conditional densities $P_{X|Y}(x|y)$
  - Class probabilities $P_Y(i)$
  - Bayes decision rule (BDR)

$$i^* = \arg\max_i P_{X|Y}(x|i) P_Y(i)$$

- We have seen that this procedure is truly optimal only if all probabilities involved are correctly estimated
- One of the most problematic factors in accurately estimating probabilities is the dimension of the feature space

Example

- Cheetah Gaussian classifier, DCT space

8 first DCT features

Prob. of error: 4%

all 64 DCT features

8%

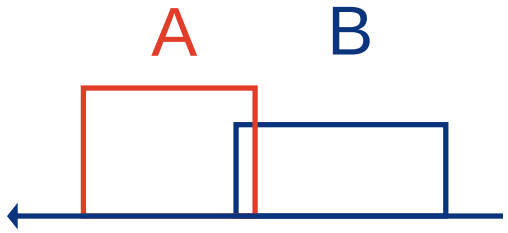
- Interesting observation: more features = higher error !

Comments on the Example

- The first reason why this happens is that things are not what we think they are in high dimensions
- one could say that high dimensional spaces are STRANGE!!!
- In practice, we invariably have to do some form of dimensionality reduction
- Eigenvalues play a major role in this
- One of the major dimensionality reduction techniques is Principal Component Analysis (PCA)

The Curse of Dimensionality

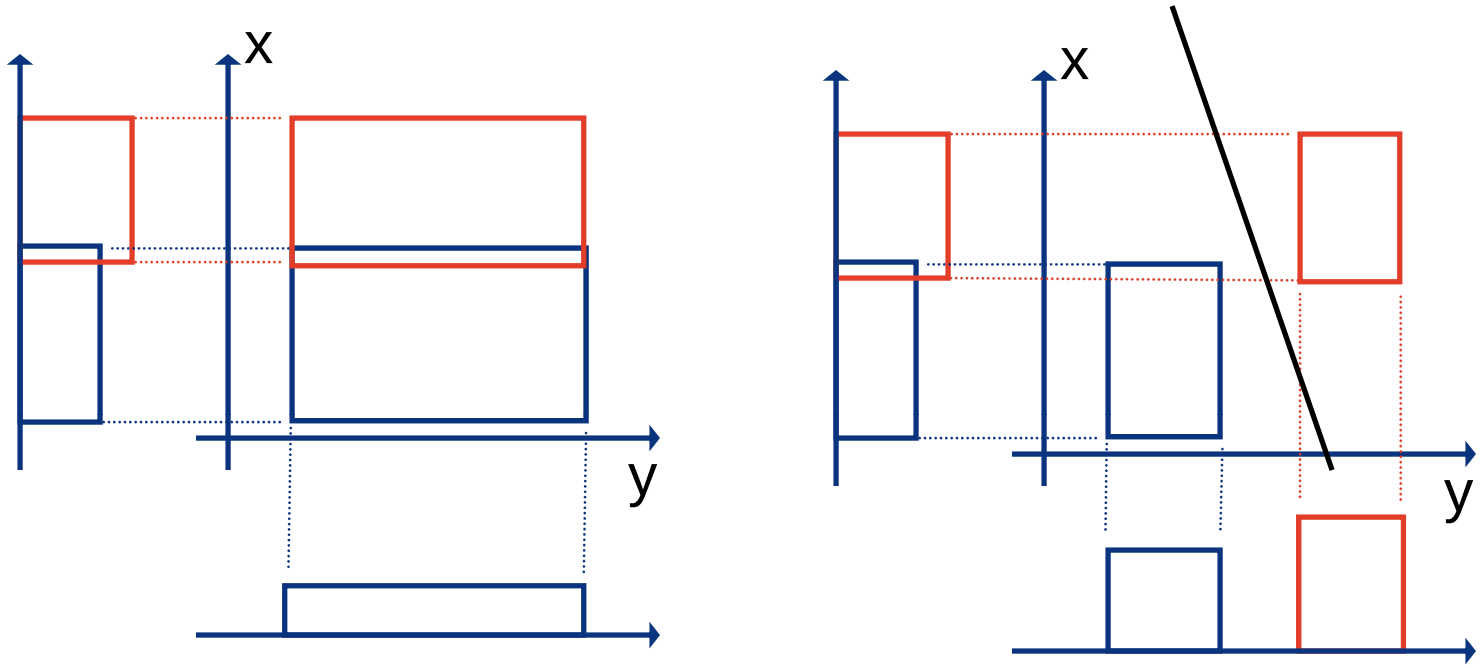
- Typical observation in Bayes decision theory:
  - Error increases when number of features is large
- This is unintuitive since *theoretically*:
  - If I have a problem in n-D I can always generate a problem in (n+1)-D without increasing the probability of error, and even often decreasing the probability of error
- E.g. two uniform classes in 1D



can be transformed into a 2D problem with the same error

- Just add a non-informative variable (extra feature dimensions) y

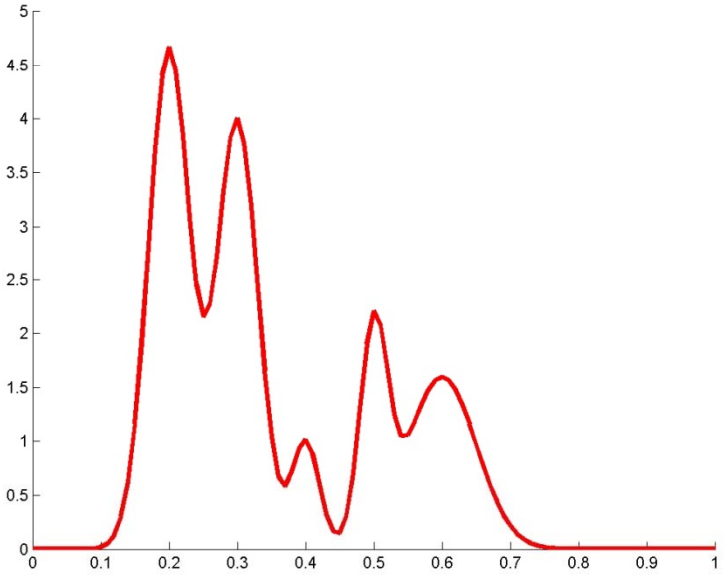
Curse of Dimensionality



- On the left, even with the new feature (dimension) y, there is no decision boundary that will achieve zero error
- On the right, the addition of the new feature (dimension) y allows a detection with has zero error

Curse of Dimensionality

- So why do we observe this curse of dimensionality?
- The problem is the quality of the density estimates
- BDR optimality assumes perfect estimation of the PDFs
- This is not easy:
  - Most densities are not simple (Gaussian, exponential, etc.) but a mixture of several factors
  - Many unknowns (# of components, what type),
  - The likelihood has multiple local minima, etc.
  - Even with algorithms like EM, it is difficult to get this right



Curse of dimensionality

The problem goes much deeper than this:

- 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

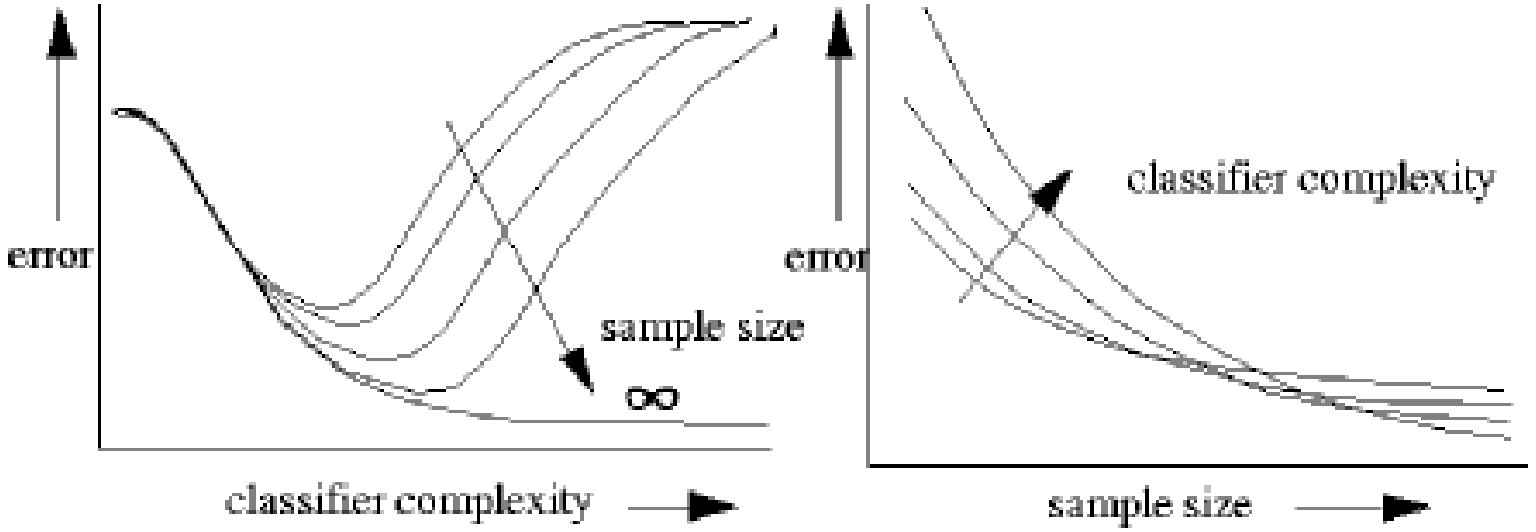
for uniform data you get, on average,

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

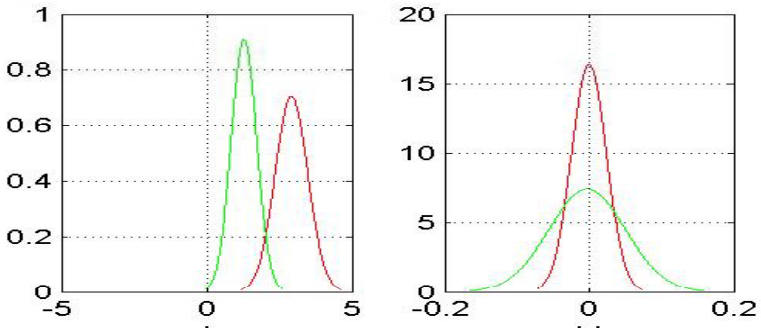
- This is the curse of dimensionality:
  - For a given classifier the number of examples required to maintain classification accuracy increases exponentially with the dimension of the feature space
- In higher dimensions the classifier has more parameters
  - Therefore: Higher complexity & Harder to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



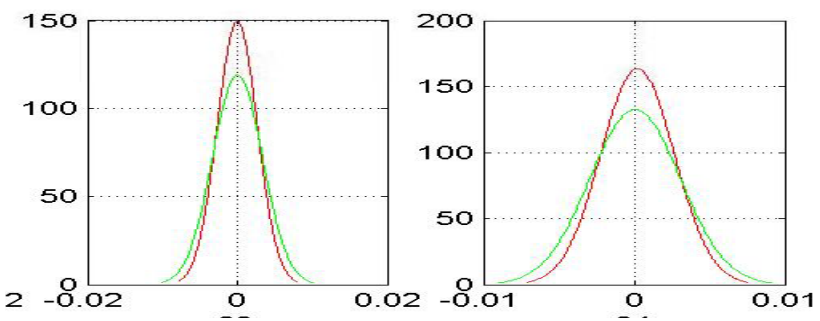
Dimensionality Reduction

- 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
- Non-discriminant means that they do not separate the classes well

discriminant



non-discriminant



Dimensionality Reduction

- Highly dependent features, even if very discriminant, are not needed - one is enough!
- E.g. data-mining company studying consumer credit card ratings:

  X = {salary, mortgage, car loan, # of kids, profession, ...}

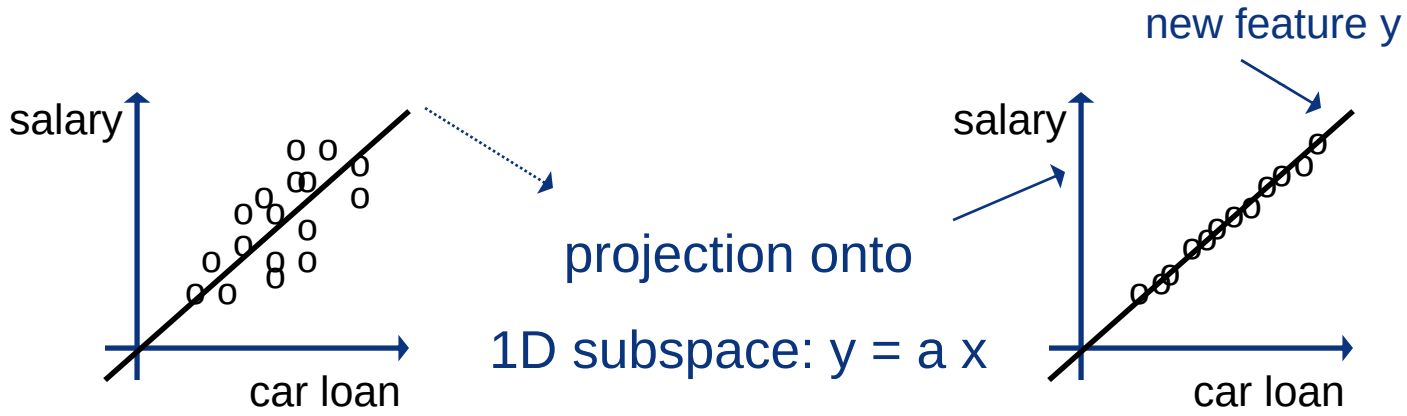
  The first three features tend to be highly correlated:

  - "the more you make, the higher the mortgage, the more expensive the car you drive"
  - from one of these variables I can predict the others very well

  Including features 2 and 3 does not increase the discrimination, but increases the dimension and leads to poor density estimates

Dimensionality Reduction

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

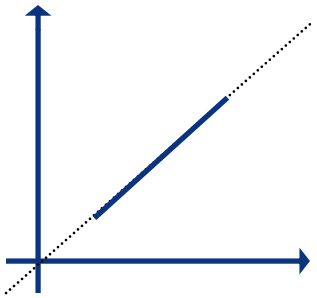


- In the example above we have a 3D hyper-plane in 5D
- If we can find this hyper-plane we can:
  - Project the data onto it
  - Get rid of two dimensions without introducing significant error

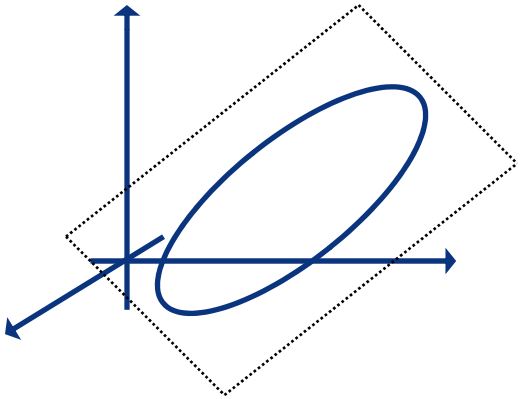
Principal Components

- **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



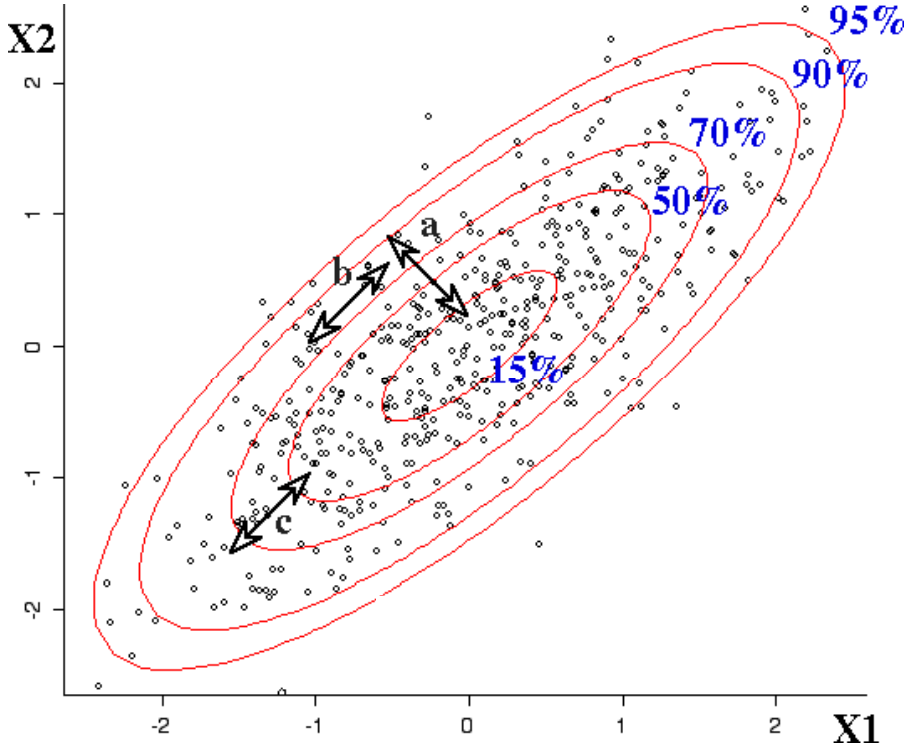
2D subspace in 3D



- 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 Principal 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

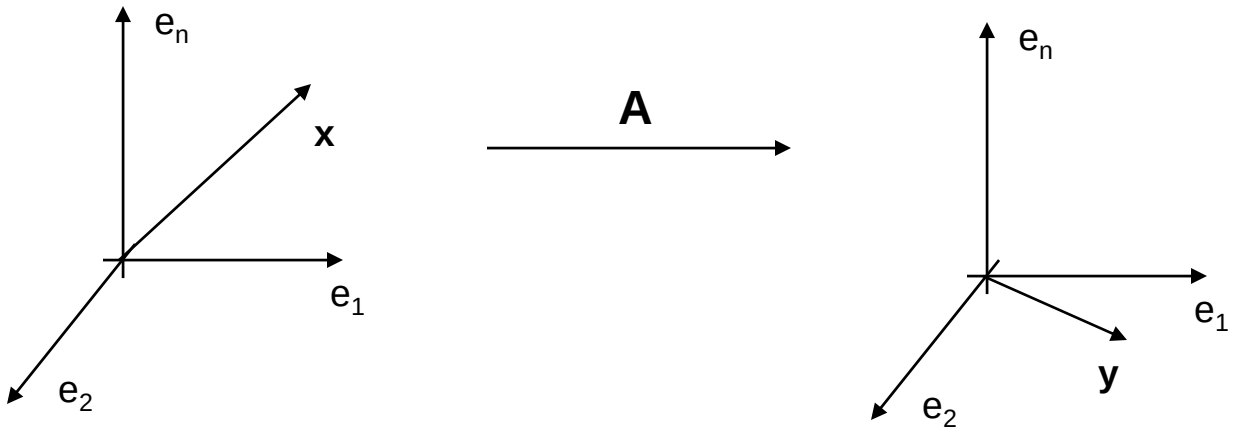


$$d^2(x,\mu) = (x-\mu)^T \Sigma^{-1} (x-\mu)$$

Principal Components & Eigenvectors

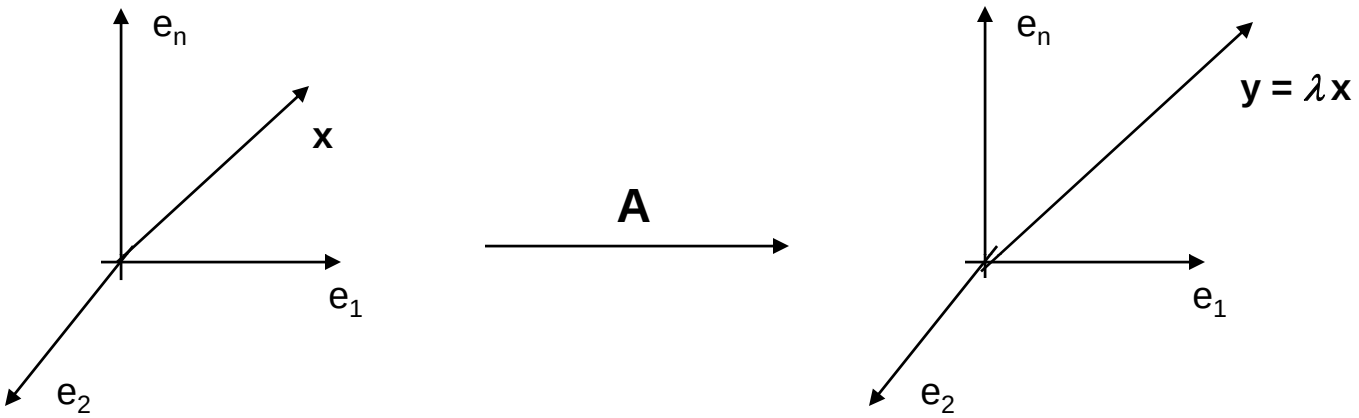
- It turns out that all relevant information is stored in the eigenvalue/vector decomposition of the covariance matrix
- So, let's start with a brief review of eigenvectors
  - Recall: a *n x n* (square) matrix can represent a linear operator that maps a vector from the space R^n back into the same space (when the domain and codomain of a mapping are the same, the mapping is an automorphism).
  - E.g. the equation y = Ax represents a linear mapping that sends *x* in R^n to *y also* in R^n

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$



Eigenvectors and Eigenvalues

- What is amazing is that there exist special ("eigen") vectors which are simply scaled by the mapping:



- These are the eigenvectors of the *n* x *n* matrix A
  - They are the solutions ϕ_i to the equation

$$A\varphi_i = \lambda_i \varphi_i$$

  where the scalars λ_i are the *n* eigenvalues of A

- For a *general* matrix A, there is NOT a full set of *n* eigenvectors

Eigenvector Decomposition

- However, If *A* is *n* x *n*, real *and* symmetric, it has *n* real eigenvalues and *n* orthogonal eigenvectors.
- Note that these can be written all at once

$$\begin{bmatrix} | & & | \\ A\varphi_1 & \cdots & A\varphi_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ \lambda_1\varphi_1 & \cdots & \lambda_n\varphi_n \\ | & & | \end{bmatrix}$$

or, using the tricks that we reviewed in the 1st week

$$A\begin{bmatrix} | & & | \\ \varphi_1 & \cdots & \varphi_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ \varphi_1 & \cdots & \varphi_n \\ | & & | \end{bmatrix}\begin{bmatrix} \lambda_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \lambda_n \end{bmatrix}$$

- **I.e.**

$$A\Phi = \Phi\Lambda$$

$$\Phi = \begin{bmatrix} | & & | \\ \varphi_1 & \cdots & \varphi_n \\ | & & | \end{bmatrix} \qquad \Lambda = \begin{bmatrix} \lambda_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & \lambda_n \end{bmatrix}$$

Symmetric Matrix Eigendecomposition

- The *n* real orthogonal eigenvectors of real $A = A^T$ can be taken to have unit norm, in which case Φ is orthogonal

$$\Phi\Phi^T = \Phi^T\Phi = I \quad \Leftrightarrow \quad \Phi^{-1} = \Phi^T$$

so that

$$A\Phi = \Phi\Lambda \quad \Leftrightarrow \quad A = \Phi\Lambda\Phi^T$$

- This is called the eigenvector decomposition, or eigendecomposition, of the matrix *A.* Because *A* is real and symmetric, it is a special case of the SVD
- This factorization of *A* allows an alternative geometric interpretation to the matrix operation:

$$y = Ax = \Phi\Lambda\Phi^T x$$

Eigenvector Decomposition

- This can be seen as a sequence of three steps
  - 1) Apply the inverse orthogonal transformation Φ^T

$$x' = \Phi^T x$$

    - This is a transformation to a rotated coordinate system (plus a possible reflection)
  - 2) Apply the diagonal operator Λ
    - This is just component-wise scaling in the rotated coordinate system:

$$x'' = \Lambda x' = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} x' = \begin{bmatrix} \lambda_1 x'_1 \\ \vdots \\ \lambda_n x'_n \end{bmatrix}$$

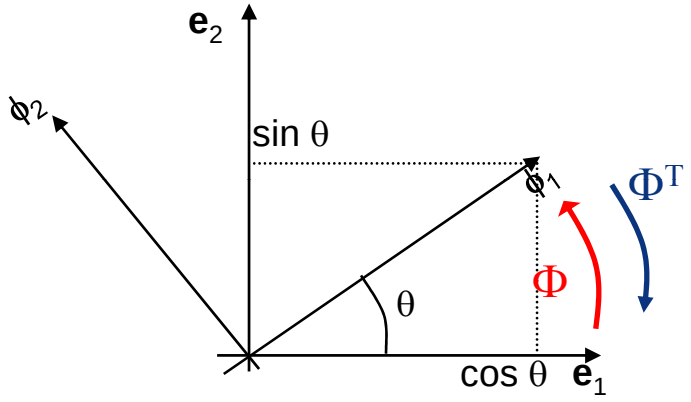
  - 3) Apply the orthogonal transformation Φ
    - This is a rotation back to the initial coordinate system

$$y = \Phi x''$$

Orthogonal Matrices

- Remember that orthogonal matrices are best understood by considering how the matrix operates on the vectors of the canonical basis (equivalently, on the unit hypersphere)
  - Note that Φ sends e_1 to ϕ_1

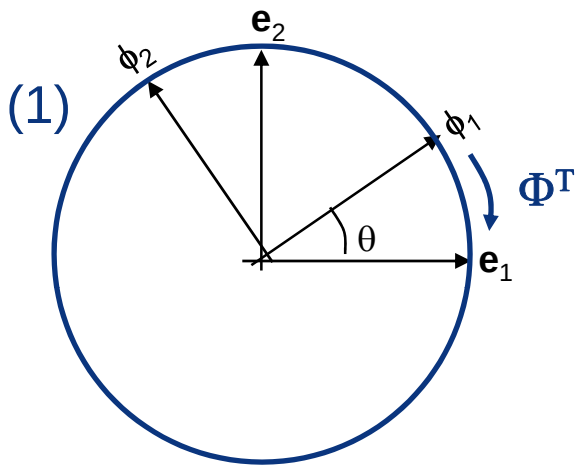
$$\varphi_1 = \begin{bmatrix} | & & | \\ \varphi_1 & \cdots & \varphi_n \\ | & & | \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ 0 \end{bmatrix}$$

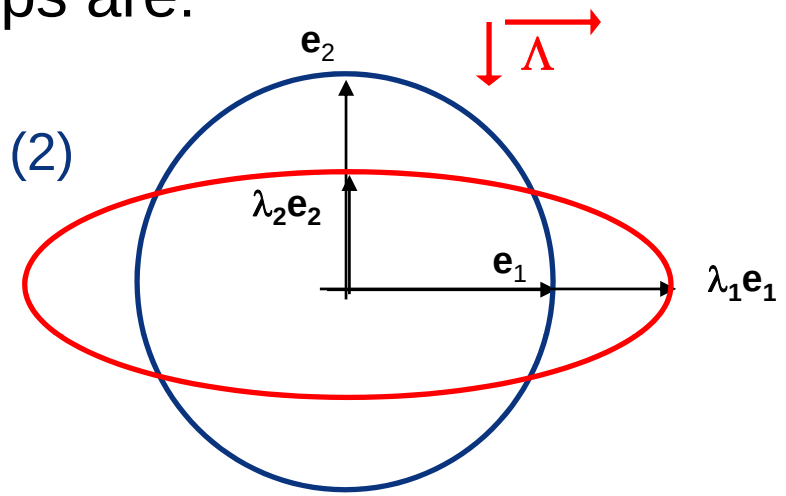


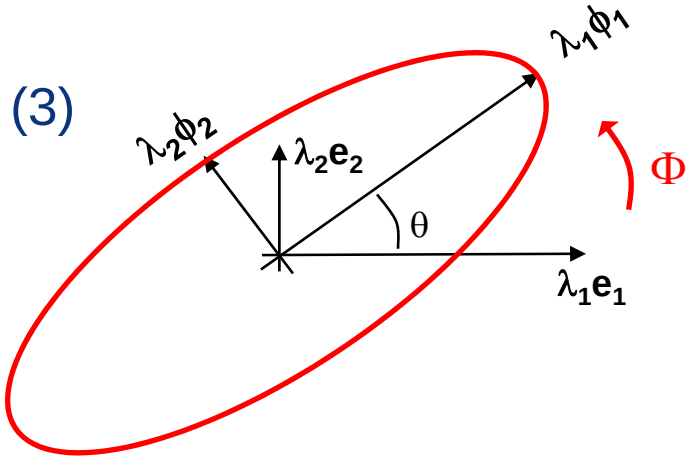
  - Since Φ^T is the inverse rotation (ignoring reflections), it sends ϕ_1 to e_1
- Hence, the sequence of operations is
  - 1) Rotate (ignoring reflections) ϕ_i to e_i (the canonical basis)
  - 2) Scale e_i by the eigenvalue λ_i
  - 3) Rotate scaled e_i back to the initial direction along ϕ_i

Eigenvector Decomposition

- Graphically, these three steps are:







This means that:

A) ϕ_i are the axes of the ellipse

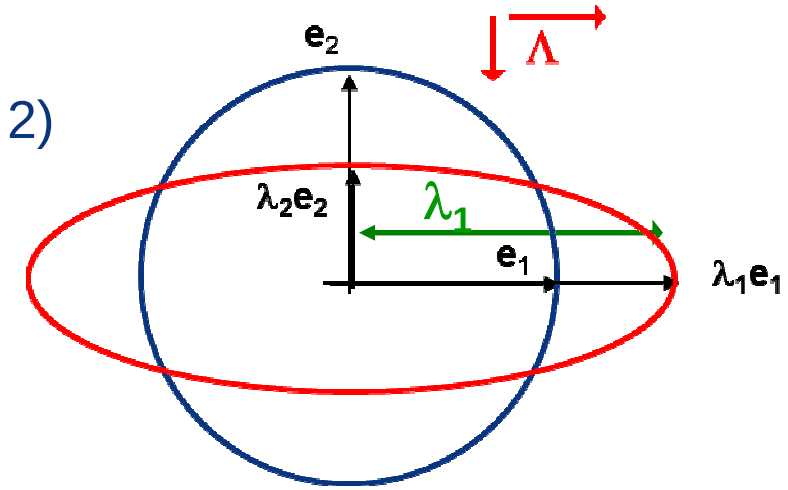
B) The width of the ellipse depends on the amount of "stretching" by λ_i

Eigendecomposition

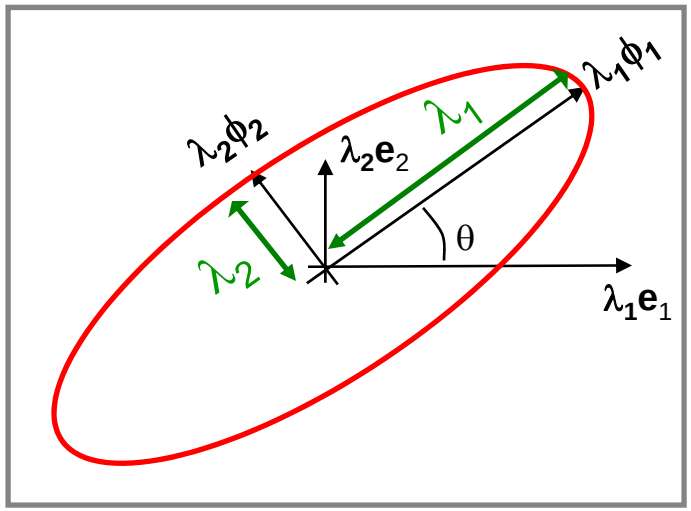
- Note that the stretching is done in Step (2):

$$x'' = \Lambda x' = \begin{bmatrix} \lambda_1 x_1' \\ \vdots \\ \lambda_n x_n' \end{bmatrix}$$

for $x' = e_i$, the length of x'' is λ_i



- Hence, the overall picture is:
  - The axes are given by the ϕ_i
  - These have length λ_i
- This decomposition can be used to find ``optimal'' lower dimensional subspaces



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 $z = x-\mu$, 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:

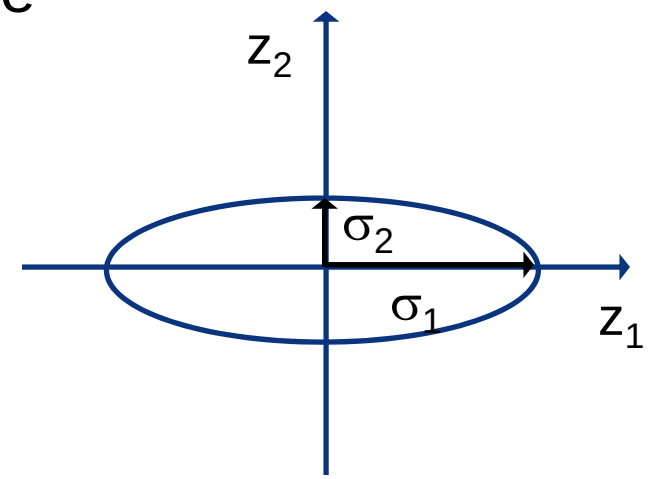
$$\Sigma = \Lambda = diag(\sigma_1^2, \ldots, \sigma_d^2) \Rightarrow z^T\Sigma^{-1}z = \sum_i \frac{z_i^2}{\sigma_i^2} = K$$

Gaussian Review

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

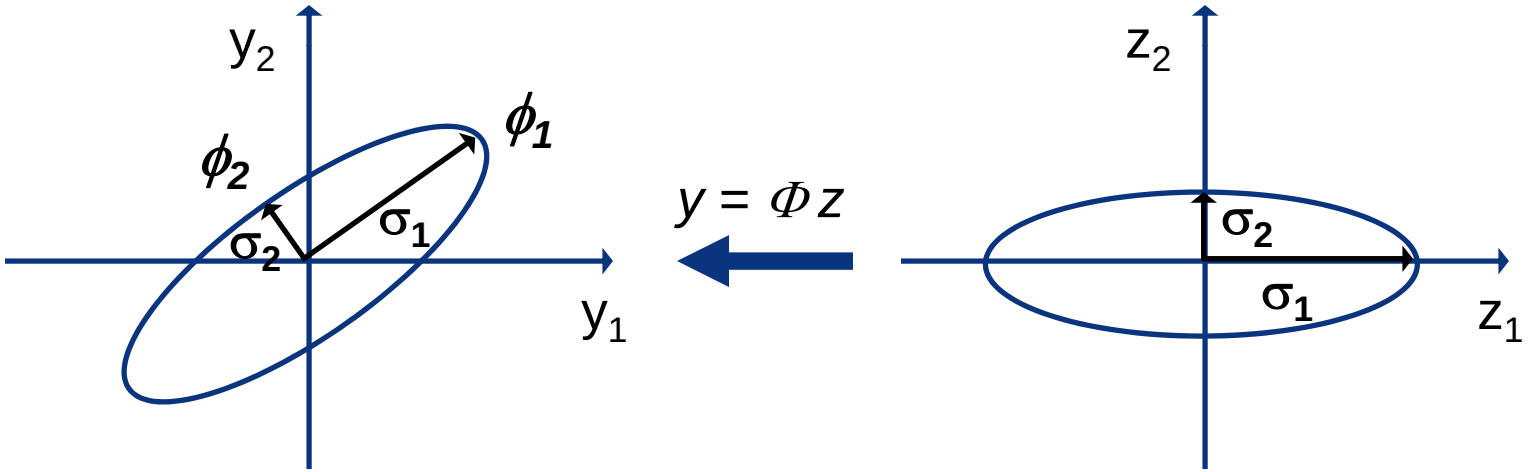
$$\frac{z_1^2}{\sigma_1^2} + \frac{z_2^2}{\sigma_2^2} = 1$$

is the ellipse



Gaussian Review

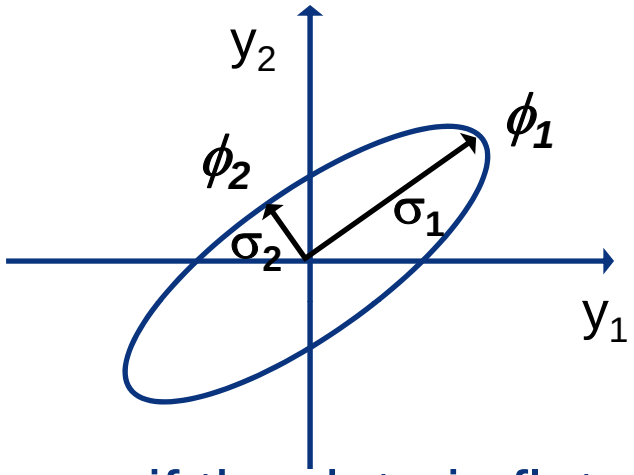
- Introduce a transformation $y = \Phi z$
- Then y has covariance $\Sigma_y = \Phi \Sigma_z \Phi^T = \Phi \Lambda \Phi^T$
- If Φ is proper orthogonal this is just a rotation and we have



- 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

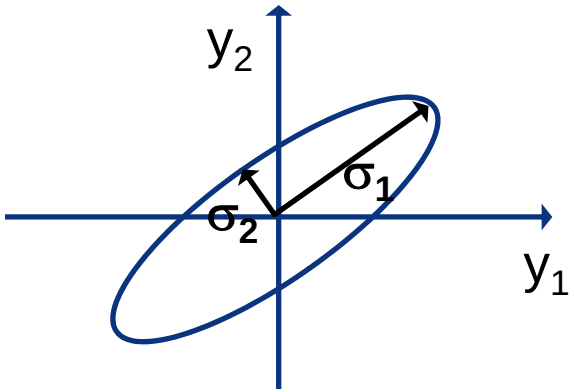
Principal Component Analysis (PCA)

- 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

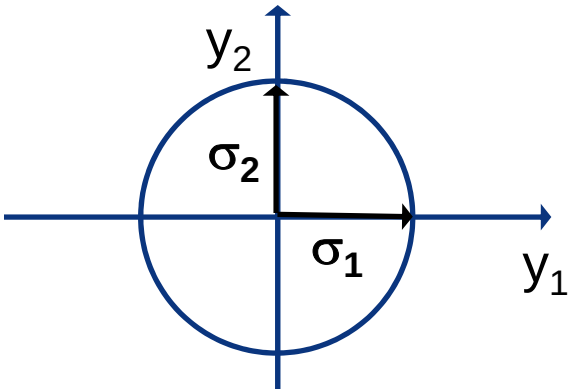


- By computing the eigenvalues we know if the data is flat

$\sigma_1 >> \sigma_2$: 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 $\mathbf{y}$.

END