PR414 / PR813 Lecture 1
Manipulating Feature Space

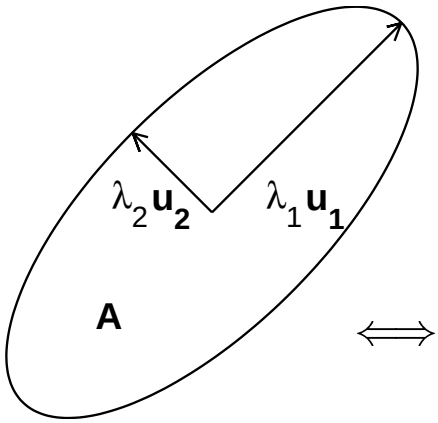
Ludwig Schwardt, Johan du Preez

University of Stellenbosch

15 February 2005

1. General Notes

- **Notation: x and N are scalars, $\boldsymbol{x}$ is a vector and $\boldsymbol{X}$ is a matrix**
- **All vectors are assumed to be *column vectors* $\Longrightarrow$ make sure this corresponds with your data!**
- **Double-check all data dimensions when doing Pattern Recognition $\Longrightarrow$ this reveals errors early**
- **Use singular value decomposition (SVD) instead of eigen-analysis to improve numerical stability $\Longrightarrow$ in Matlab, replace `[V,D] = eig(A)` with `[V,D,ignore] = svd(A)` (this has the advantage of also sorting the eigenvalues...)**
- **It is always a good idea to *plot* your feature data $\Longrightarrow$ this leads to better insights and catches stupid mistakes as well**
- **A square symmetric positive semi-definite matrix such as a covariance matrix has *orthonormal* eigenvectors and *non-negative* eigenvalues $\Longrightarrow$ always possible to represent it by an ellipsoid shape with scaled eigenvectors as axes**

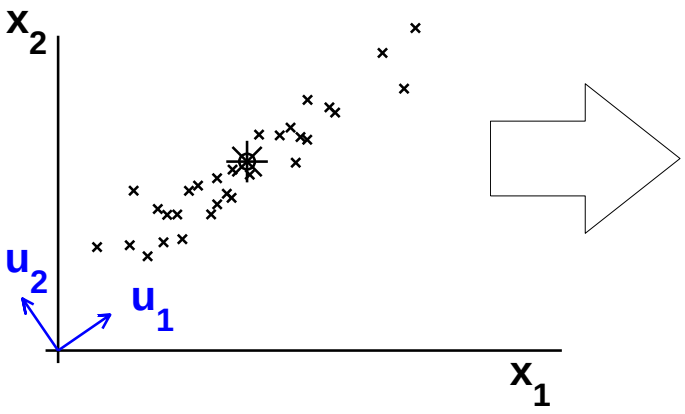


$$\Longleftrightarrow \boldsymbol{A} = [\boldsymbol{u}_1 \boldsymbol{u}_2] \times \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \times [\boldsymbol{u}_1 \boldsymbol{u}_2]^T$$

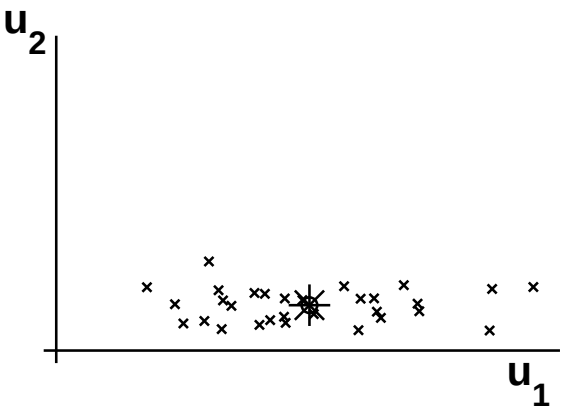
2. Principal Component Analysis (PCA)

- Also known as *Karhunen-Loéve transform* (KLT). It uses either the *correlation* or *covariance* matrices of the data to find a *linear subspace* efficient for *representation*.
- Based on the correlation matrix it finds a subspace that minimises the mean-square error between the feature vectors and their projections in this space. The principle axis then passes through the data mean.
- Based on the covariance matrix, the data mean is effectively shifted to the origin of the feature space. The mse solution now lines up with the primary directions of the data which *decorrelates* the feature data via rotation.

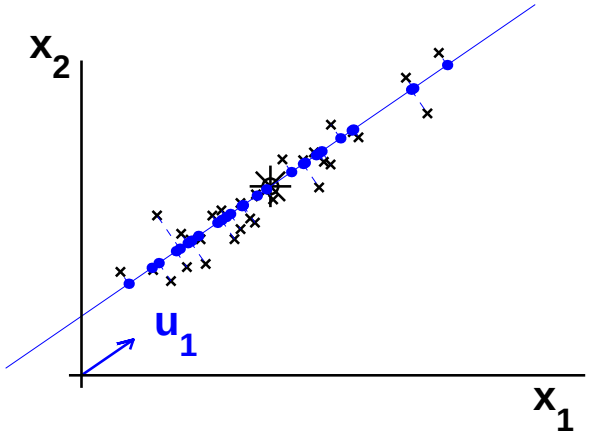
Feature data (* = mean)



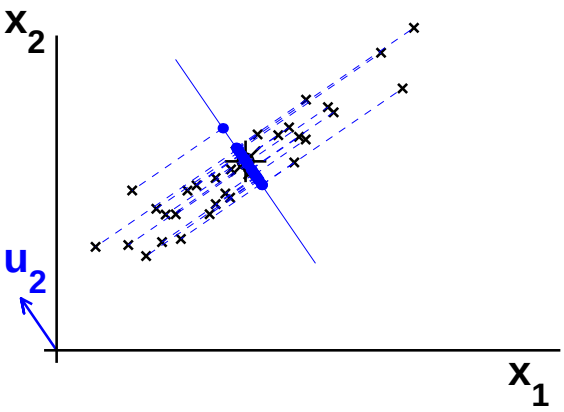
After PCA



Best 1−dim linear subspace



Worst 1−dim linear subspace



3. Mathematical basis for PCA

- Consider y with covariance C_y. Project it on a normalised basis Q to get the projected vector $x = Q'y$ (from linear algebra).
- Its covariance is given by $C_x = \mathbf{E}[(x - \overline{x})(x - \overline{x})'] = Q'C_yQ$ (from multi-dimensional probability theory).
- Now do an eigen decomposition of the covariance: $C_yV = VD$ (matrix formulation of eigen decomposition). V is orthonormal and its columns are the eigenvectors. This means that $V'V = I$. D is diagonal and contains the eigenvalues. Therefore $V'C_yV = D$.
- But, from the previous paragraph, if we should project the vector y on the orthonormal basis V, the resultant covariance matrix will be $C_x = V'C_yV = D$. In other words, by projecting the vector y on the eigenvectors of its own covariance matrix, we decorrelate the random variables on the various dimensions of y!

4. Calculating PCA

1. **Calculate covariance matrix Σ (Matlab `cov`):**

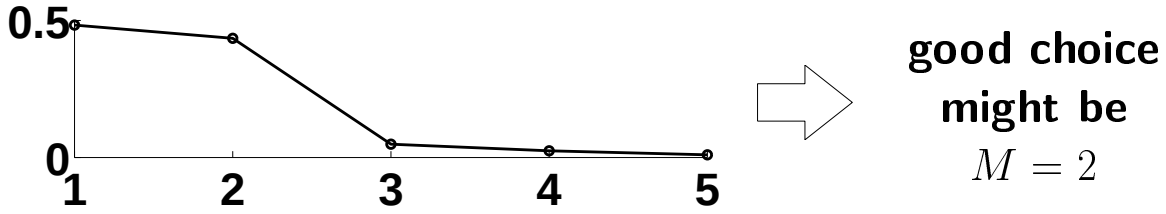
$$\boldsymbol{\mu} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n$$

$$\boldsymbol{\Sigma} = \frac{1}{N-1}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T$$

2. **Do eigenanalysis of $\Sigma = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ (Matlab `eig`)**
3. **Select the M biggest eigenvalues $\lambda_1 \cdots \lambda_M$**
4. **Pack their associated eigenvectors into the transformation matrix $\boldsymbol{W} = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_M]$**
5. **Transform the data via $\boldsymbol{y} = \boldsymbol{W}^T\boldsymbol{x}$**

Take note:

- **Typically all data are pooled together for PCA, regardless of class labels**
- **Choice of M is *black art* $\Longrightarrow$ useful tool is plot of sorted normalised eigenvalues $\lambda_m / \sum_{m=1}^{M}\lambda_m$ (look for knee):**



5. Verify your code with a test case

With the following data (feature vectors as rows):

1	0	2
2	1	4
2	4	1
1	2	2
1	-1	1
-2	-2	-2

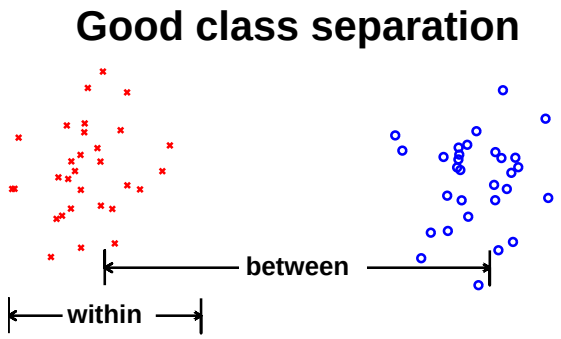
you should get an unreduced principal axes system (basis vectors in the columns):

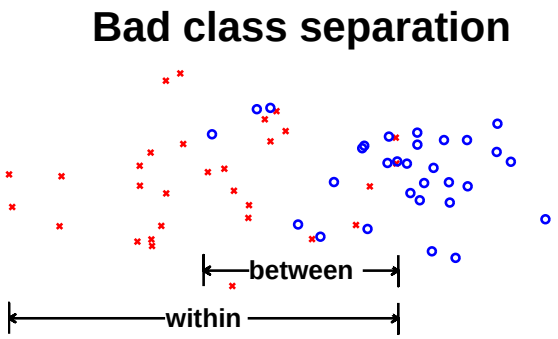
-4.923122e-01	1.391148e-01	-8.592297e-01
-6.510149e-01	-7.140919e-01	2.573954e-01
-5.777615e-01	6.860902e-01	4.421219e-01

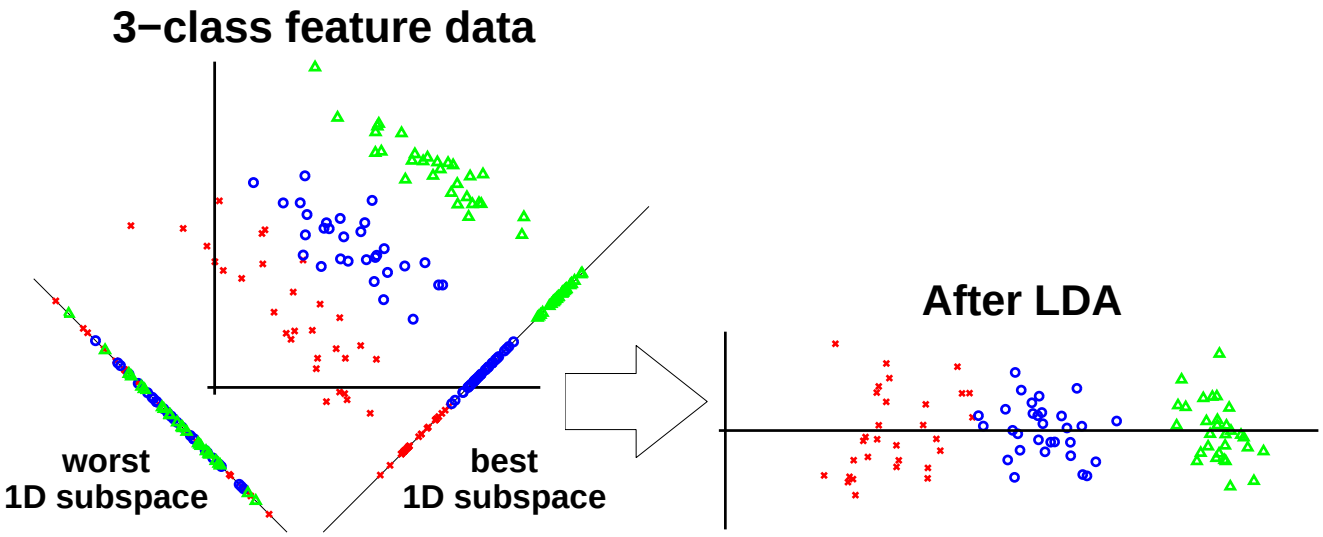
If you reduce dimension the columns should disappear from the right.

6. Linear Discriminant Analysis (LDA)

- Also known as *class-based KLT*, *Fisher discriminants*
- Finds a *linear subspace* that maximises class separability among the feature vector projections in this space
- Popular separability criterion is ratio of *between-class* scatter and *within-class* scatter
- LDA seeks directions efficient for *discrimination*







7. Calculating LDA

1. **For each class $i = 1 \ldots C$, calculate mean $\boldsymbol{\mu}_i$, covariance matrix $\boldsymbol{\Sigma}_i$ and *a priori* class probability P_i**

$$P_i = \frac{N_i}{N}, \quad \text{with } N_i \text{ number of vectors in class } i$$

2. **Calculate scatter matrices $\boldsymbol{S}_W$ and $\boldsymbol{S}_B$:**

$$\boldsymbol{S}_W = \sum_{i=1}^{C} P_i \boldsymbol{\Sigma}_i, \quad \boldsymbol{\mu} = \sum_{i=1}^{C} P_i \boldsymbol{\mu}_i,$$

$$\boldsymbol{S}_B = \sum_{i=1}^{C} P_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$$

3. **Do eigenanalysis of $\boldsymbol{S}_W = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^T$ (Matlab `eig`)**
4. **Discard any zero eigenvalues of $\boldsymbol{S}_W$ with their eigenvectors**
5. **Form whitening transform $\boldsymbol{B} = \boldsymbol{V}\boldsymbol{D}^{-1/2}$**
6. **Obtain new $\boldsymbol{S}'_B = \boldsymbol{B}^T \boldsymbol{S}_B \boldsymbol{B}$**
7. **Do eigenanalysis of $\boldsymbol{S}'_B = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ (Matlab `eig`)**
8. **Select the M biggest eigenvalues $\lambda_1 \cdots \lambda_M$**
9. **Pack their associated eigenvectors into the transformation matrix $\boldsymbol{W} = \boldsymbol{B} \times [\boldsymbol{u}_1 \cdots \boldsymbol{u}_M]$**
10. **Transform the data via $\boldsymbol{y} = \boldsymbol{W}^T \boldsymbol{x}$**

8. Notes on LDA

- **Maximum number of dimensions after LDA is** $C-1$
- **Unlike PCA,** $\boldsymbol{W}$ **is not just rotation** $\Longrightarrow$ **scales as well**
- **Separability criterion is ratio of between-class scatter matrix** $\boldsymbol{S}_B$ **and within-class scatter matrix** $\boldsymbol{S}_W$ **of *transformed data*, as a function of transformation** $\boldsymbol{W}$**, given by**

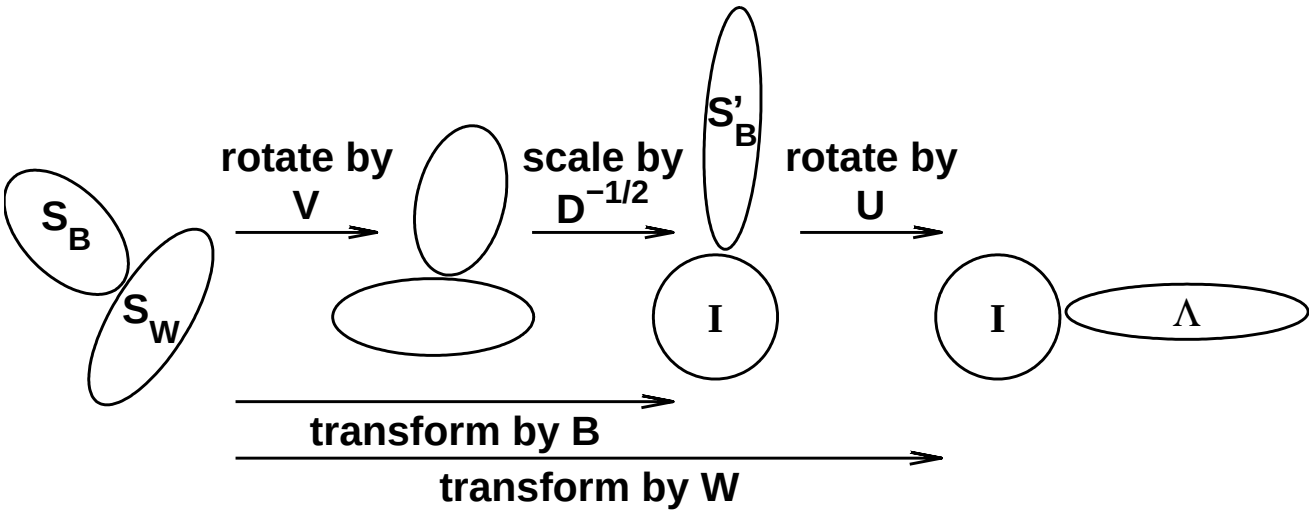
$$J(\boldsymbol{W}) = \frac{\left|\boldsymbol{W}^T\boldsymbol{S}_B\boldsymbol{W}\right|}{\left|\boldsymbol{W}^T\boldsymbol{S}_W\boldsymbol{W}\right|} \quad \textbf{or} \quad \mathrm{tr}\left[(\boldsymbol{W}^T\boldsymbol{S}_W\boldsymbol{W})^{-1}(\boldsymbol{W}^T\boldsymbol{S}_B\boldsymbol{W})\right]$$

- **Maximising** $J(\boldsymbol{W})$ **leads to *generalised eigenvalue* problem**

$$\boldsymbol{S}_B\boldsymbol{W} = \boldsymbol{S}_W\boldsymbol{W}\boldsymbol{\Lambda}$$

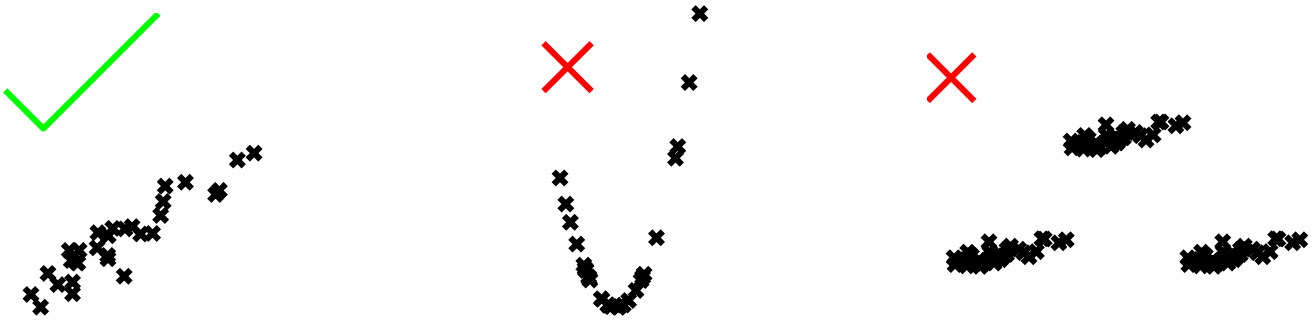
- **In Matlab this is** `eig(Sb,Sw)` $\Longrightarrow$ **same as steps 3-9...**
- **This is also known as *simultaneous diagonalisation*, since** $\boldsymbol{W}$ **diagonalises both** $\boldsymbol{S}_W$ **and** $\boldsymbol{S}_B$ **as**

$$\boldsymbol{W}^T\boldsymbol{S}_W\boldsymbol{W} = \boldsymbol{I} \quad \textbf{and} \quad \boldsymbol{W}^T\boldsymbol{S}_B\boldsymbol{W} = \boldsymbol{\Lambda}$$

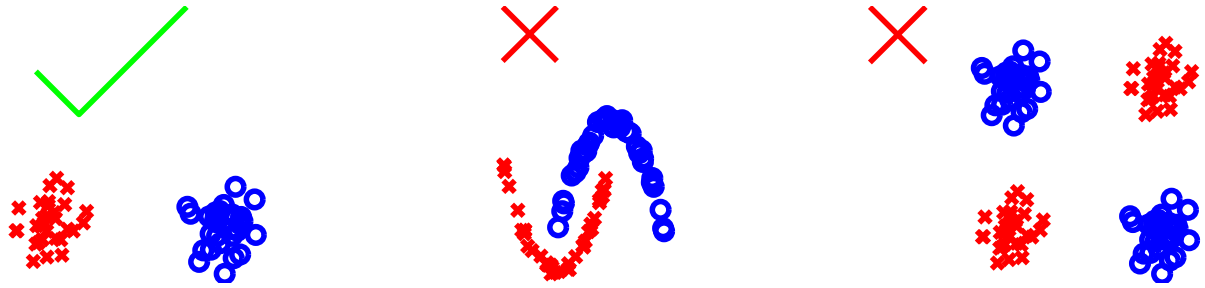


9. Limitations

- PCA cannot exploit the extra compression associated with nonlinear relationships and multimodal patterns



- LDA assumes that *class means* convey most of class information $\Longrightarrow$ this favours unimodal data distributions
- LDA therefore cannot enhance nonlinearly separable data sets and classes with the same mean



- Nonlinear extensions to PCA and LDA very popular $\Longrightarrow$ especially *kernel-based* methods look very promising

10. Verify your code with a test case

With the following data (feature vectors as rows):

Class 1:

$$\begin{matrix} 1 & 0 & 2 \\ 2 & 1 & 4 \\ 1 & 0 & 2 \end{matrix}$$

Class 2:

$$\begin{matrix} 2 & 4 & 1 \\ 1 & 2 & 2 \end{matrix}$$

Class 3:

$$\begin{matrix} 1 & -1 & 1 \\ -2 & -2 & -2 \\ 1 & -1 & 1 \end{matrix}$$

and a reduction to 2 dimensions, you should get a final transformation matrix (basis vectors in the columns):

$$\begin{matrix} -2.195944e{+}00 & 6.580001e{-}01 \\ 2.276487e{+}00 & 2.143681e{-}01 \\ 1.200649e{+}00 & -1.113292e{+}00 \end{matrix}$$

This is made up as the product of an initial whitening transformation:

$$\begin{matrix} -3.677406e\text{-}01 & -3.113625e\text{-}01 & -2.678662e{+}00 \\ -1.813796e\text{-}01 & -8.972333e\text{-}01 & 2.120238e{+}00 \\ -3.947572e\text{-}01 & 7.023062e\text{-}01 & 1.521150e{+}00 \end{matrix}$$

and a reduced axes system for the transformed means vectors (here we first rescaled the inter-class scatter matrix so that the cov function effectively divided by N instead of N-1):

$$\begin{matrix} -2.448890e\text{-}01 & 4.896354e\text{-}01 \\ -3.695377e\text{-}01 & -8.450977e\text{-}01 \\ 8.963656e\text{-}01 & -2.146324e\text{-}01 \end{matrix}$$

(Remember to take the prior probabilities of each class into account in each case.)