Lecture 2d

Latent variable models

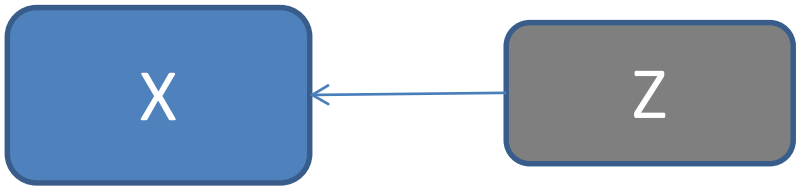
Overview

- Principal Component Analysis (PCA)
- Probabilistic PCA
- Independent component analysis (ICA)

Latent variables

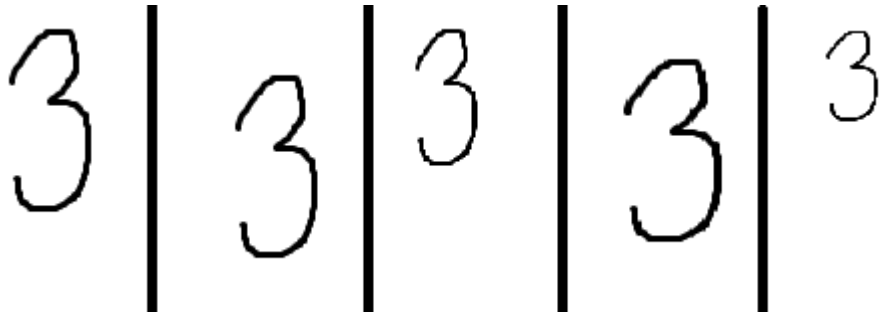
- Sometimes data depends on the variables we can not measure (hard to measure)
  - Answers on the test depend on Intelligence
  - Brain activity in the brain is measured by sensors
  - Stock prices depend on market confidence

Source: Leadliaison.com



Latent variables

- Latent factor discovered→ data storage may decrease a lot



- Latent factors
  - Center
  - Scaling
- Original vs compressed
  - 100x100x5=50000
  - 100x100+2*5+2*5=10020

Principal Component Analysis (PCA)

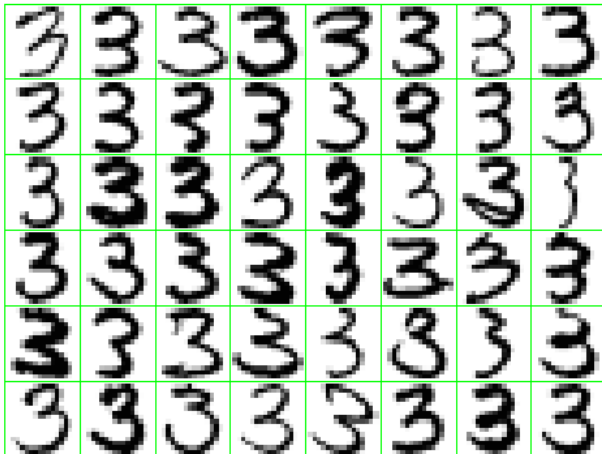
- *PCA* is a technique for reducing the complexity of high dimensional data
- It can be used to approximate high dimensional data with a few dimensions (latent features) –> much less data to store
- New variables might have a special interpretation

Applications

- Image recognition
- Information compression
- Subspace clustering
- …

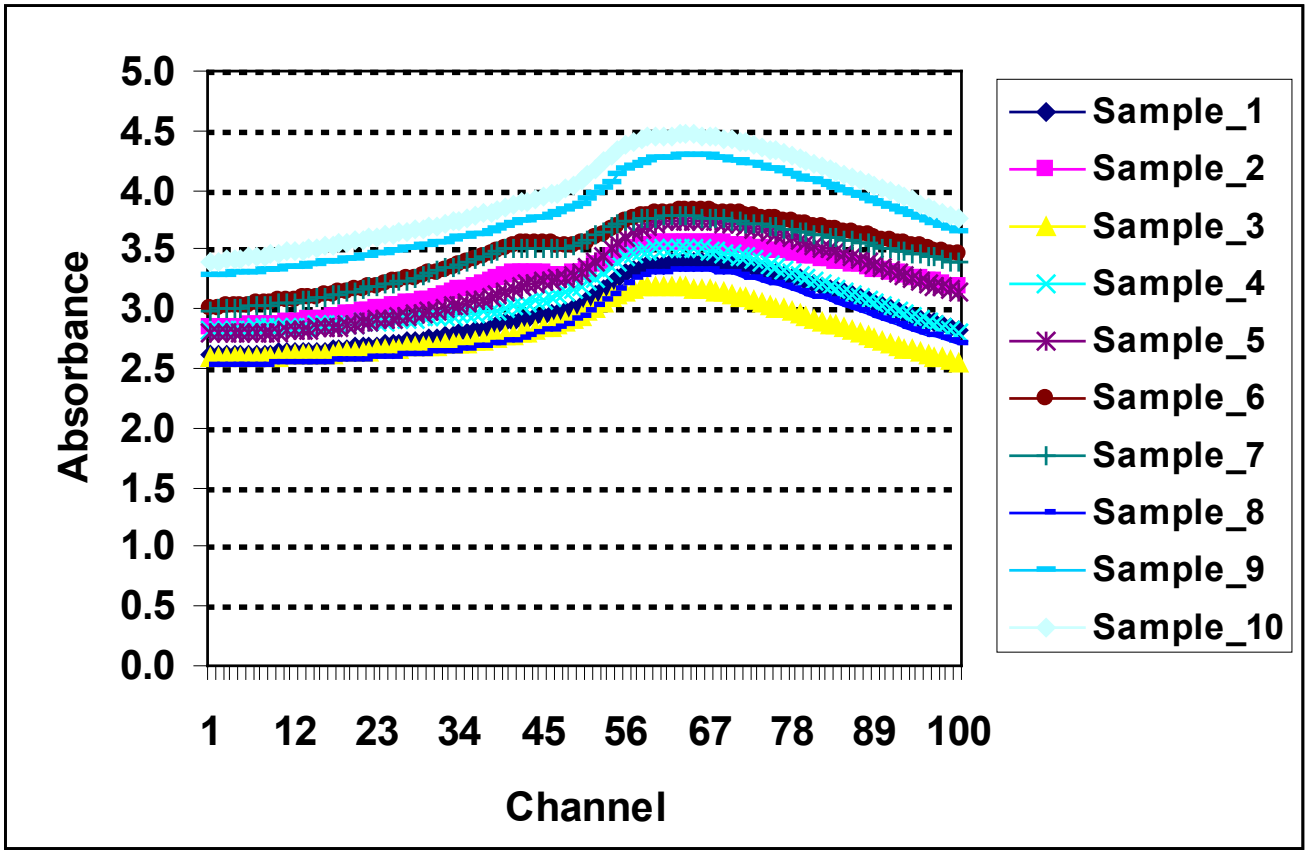
Principal Component Analysis (PCA)

- Example 1: Hadwritten digits
  - Can we get a more compact summary?



Absorbance records for ten samples of chopped meat

Parallel coordinate plot for "FAT"



1 target (fat)

100 features (absorbance at 100 wavelengths or channels)

The features are strongly correlated to each other

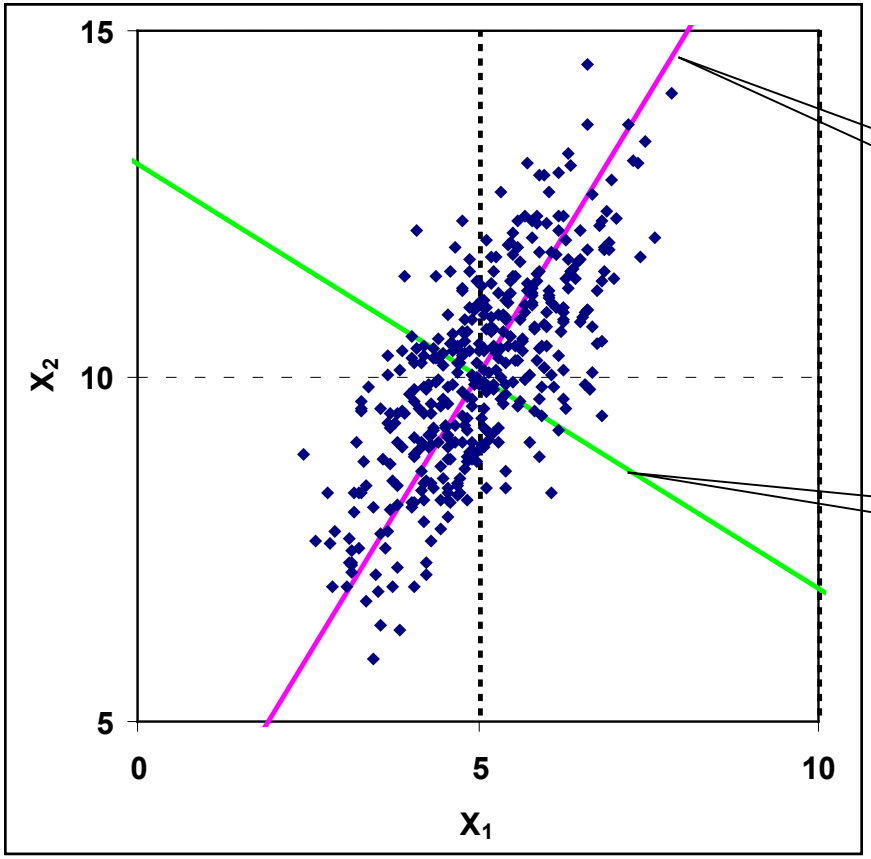
Principal components analysis

Idea: Introduce a new coordinate system (PC1, PC2, …) where

- The first principal component (PC1) is the direction that maximizes the variance of the projected data
- The second principal component (PC2) is the direction that maximizes the variance of the projected data after the variation along PC1 has been removed
- The third principal component (PC3) is the direction that maximizes the variance of the projected data after the variation along PC1 and PC2 has been removed
- ….

In the new coordinate system, coordinates corresponding to the last principal components are very small → can take away these columns

Principal Component Analysis
- two inputs

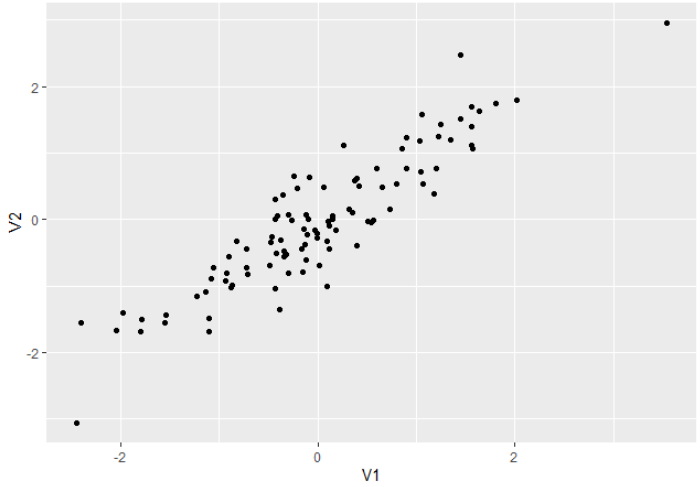


PC1

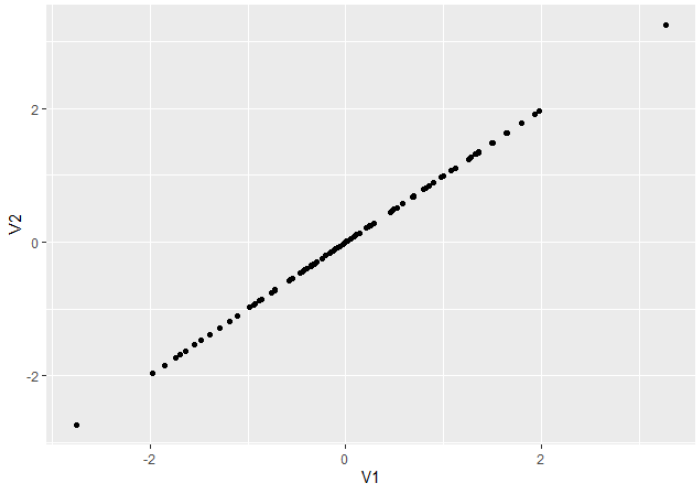
PC2

PCA- after reducing dimensionality

Original data



After compression

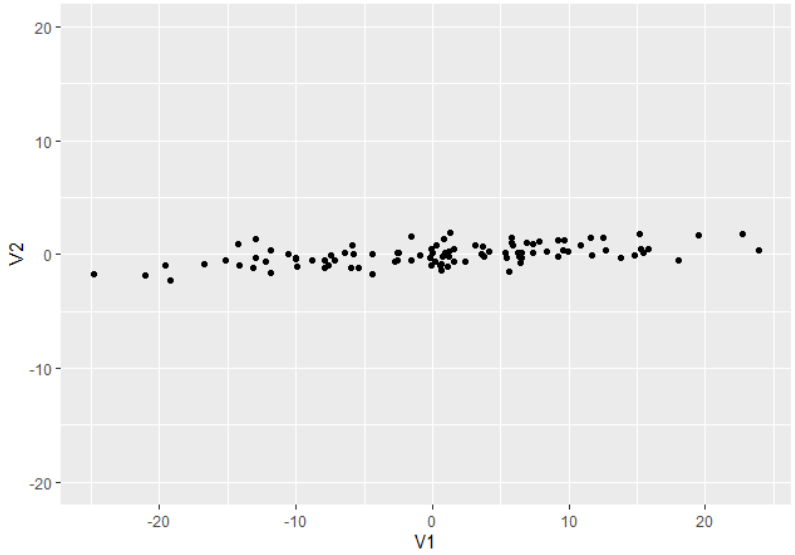


- Data became approximate (but less data to store)
- $PC_1, \ldots PC_M$ are actually eigenvections of **sample covariance** (first largest eigenvalue,…,Mth largest egenvalue)

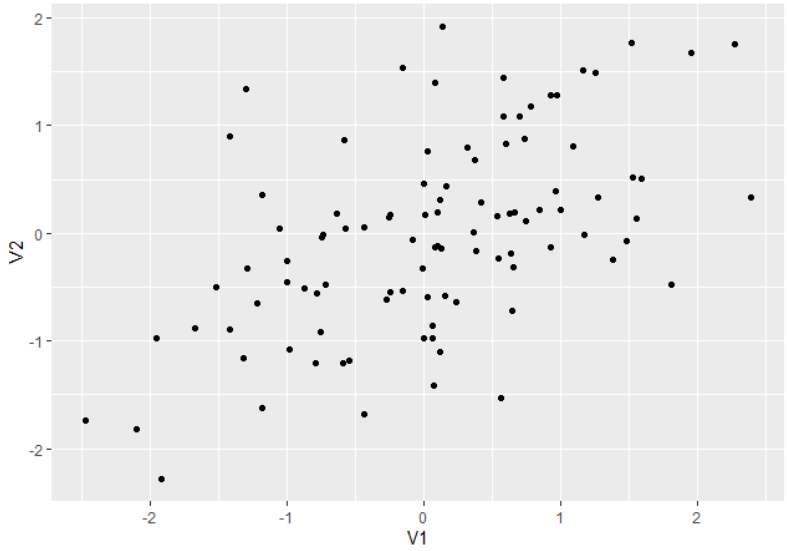
PCA and scaling

- Do we need to scale features?

Without scaling



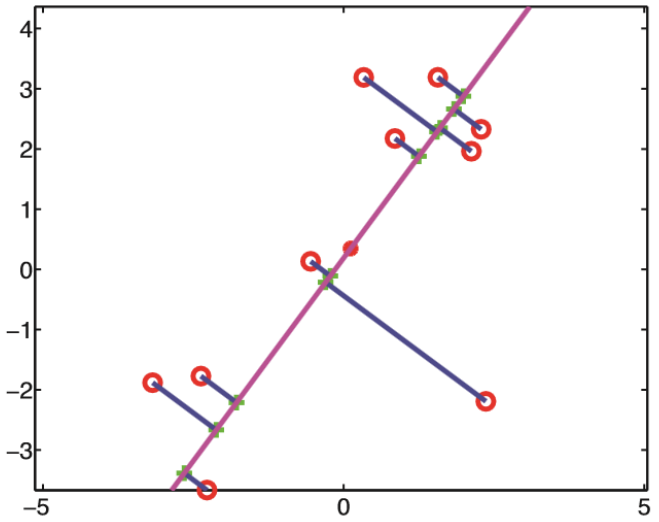
After scaling



PCA: another view

- Aim: minimize the distance between the original and projected data

$$\min_{U_M} \sum_{i=1}^{N} \|x_n - \tilde{x}_n\|^2$$



Source: Murphy

PCA: computations

Data $D = \left\| \mathbf{x}_1 \ \mathbf{x}_2 \ \ldots \ \mathbf{x}_p \right\|, \quad \mathbf{x}_i = \left(x_{i1}, \ldots, x_{in} \right)$

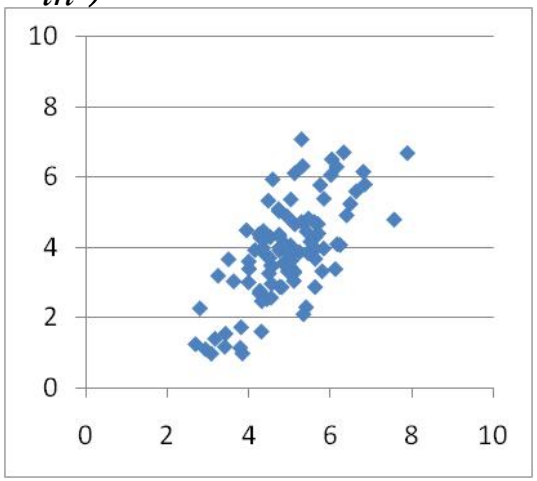
1. Centred data

$$X = \left\| \mathbf{x}_1 - \overline{\mathbf{x}}_1 \ \mathbf{x}_2 - \overline{\mathbf{x}}_2 \ \ldots \ \mathbf{x}_p - \overline{\mathbf{x}}_p \right\|,$$

2. Covariance matrix

$$\boldsymbol{S} = \frac{\mathbf{1}}{\boldsymbol{N}} \boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X}$$

3. Search for eigenvectors and eigenvalues of **S**



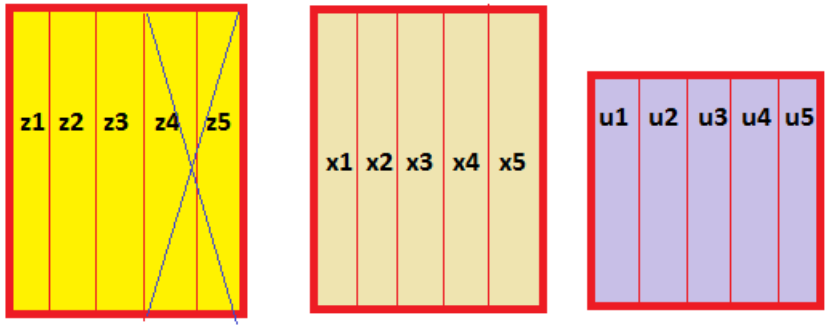
	Column 1	*Column 2*
Column 1	0.951	0.905
Column 2	0.905	1.883

PCA: computations

4. Coordinates of any data point $x=(x_1 \ldots x_p)$ in the new coordinate system:

$$z = (z_1, \ldots z_n),\ z_i = x^T u_i$$

Matrix form: $Z = X\ U$



5. Discard principle components after some *M:*

$$Z = X\ U_M$$

Store: N x M+ p x M instead N x p

6. New data will have dimensions N x M instead of N x p

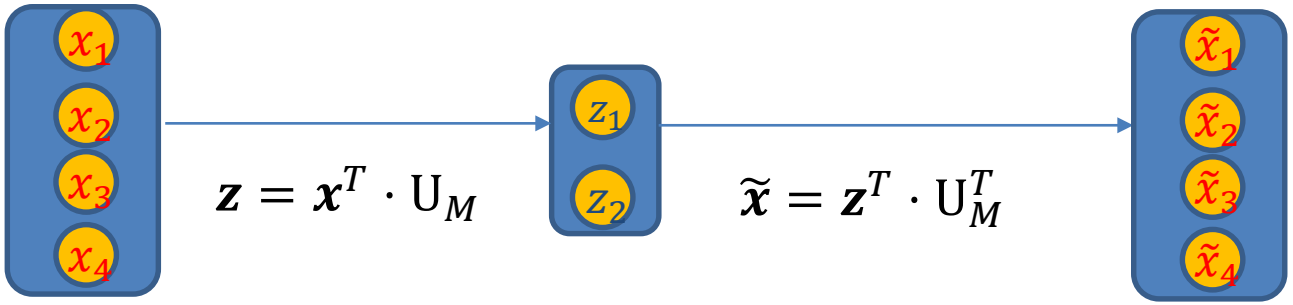
100*50 vs 100*4+50*4

Getting approximate original data:

$$\tilde{X} = ZU_M^T$$

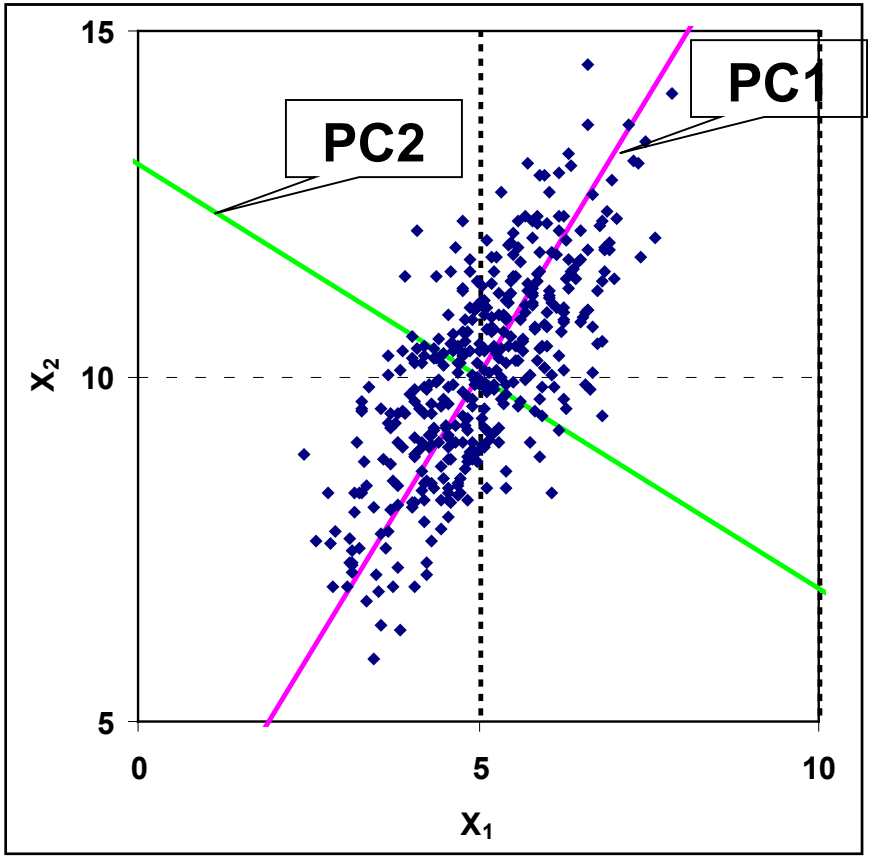
PCA: computations

- PCA makes a **linear** compression of features



$$\min_{U_M} \sum_{i=1}^{N} \|x_n - \tilde{x}_n\|^2$$

Principal Component Analysis



Eigenanalysis of the Covariance Matrix

Eigenvalue	2.8162	0.3835
Proportion	0.880	0.120
Cumulative	0.880	1.000

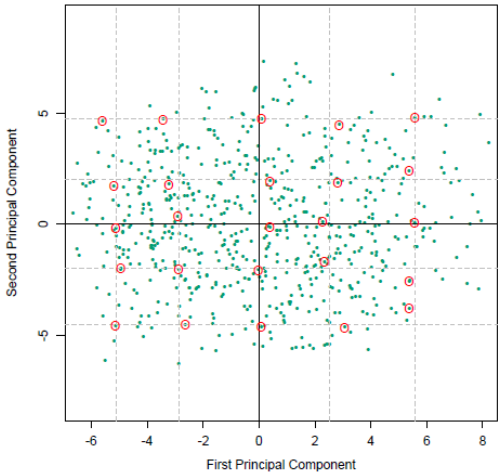
Variable	PC1	PC2
X1	0.523	0.852
X2	0.852	-0.523

Loadings (U)

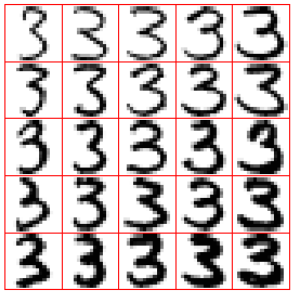
Principal Component Analysis

- Digits: two eigenvectors extracted

- Interptretation of eigenvectiors



Lower tail length

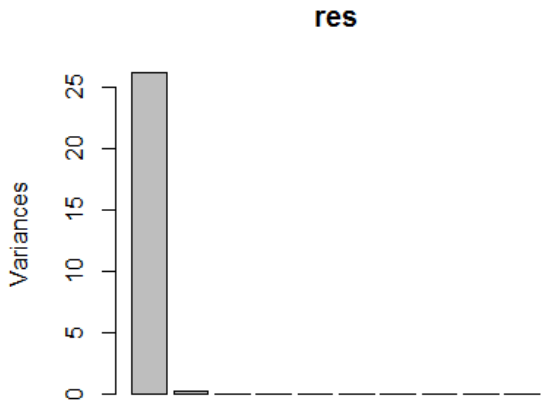


thickness

PCA in R

- Prcomp(), biplot(), screeplot()

```
mydata=read.csv2("tecator.csv")
data1=mydata
data1$Fat=c()
res=prcomp(data1)
lambda=res$sdev^2
#eigenvalues
lambda
#proportion of variation
sprintf("%2.3f",lambda/sum(lambda)*100)
screeplot(res)
```




```
> lambda
  [1] 2.612713e+01 2.385369e-01 7.844883e-02 3.018501e-
  [7] 2.052212e-04 1.084213e-04 2.077326e-05 1.150359e-
```

```
> sprintf("%2.3f",lambda/sum(lambda)*100)
  [1] "98.679" "0.901"  "0.296"  "0.114"  "0.006
  [9] "0.000"  "0.000"  "0.000"  "0.000"  "0.000
```

Only 1 component captures the 99% of variation!

PCA in R

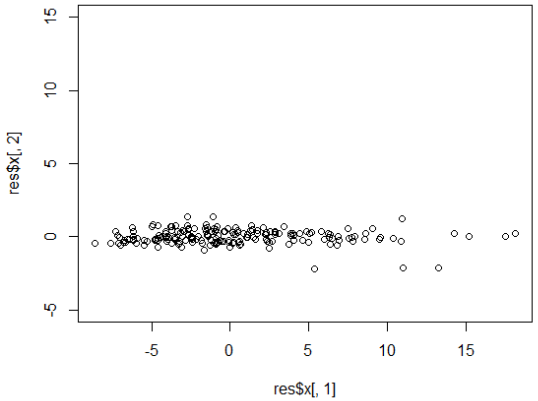
- Principal component **loadings (U)**

```
U=res$rotation
head(U)
```

```
> head(U)
                PC1       PC2        PC3
Channel1 0.07938192 0.1156228 0.08073156 -0.0927
Channel2 0.07987445 0.1170972 0.07887873 -0.0981
Channel3 0.08036498 0.1185571 0.07702127 -0.1031
Channel4 0.08085611 0.1200006 0.07515015 -0.1077
Channel5 0.08135022 0.1214075 0.07323819 -0.1119
Channel6 0.08184896 0.1227491 0.07125048  0.1156
```

- Data in (PC1, PC2) – **scores (Z)**

```
plot(res$x[,1], res$x[,2], ylim=c(-5,15))
```

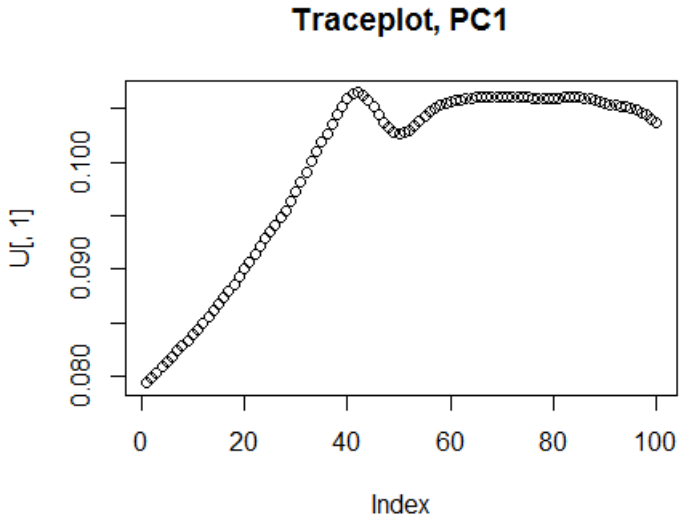


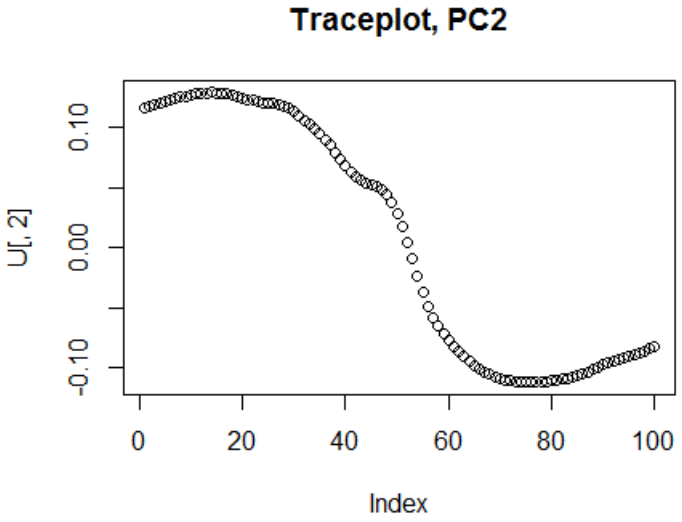
Do we need second dimension?

PCA in R

- Trace plots

```
U= res$rotation
plot(U[,1], main="Traceplot, PC1")
plot(U[,2],main="Traceplot, PC2")
```

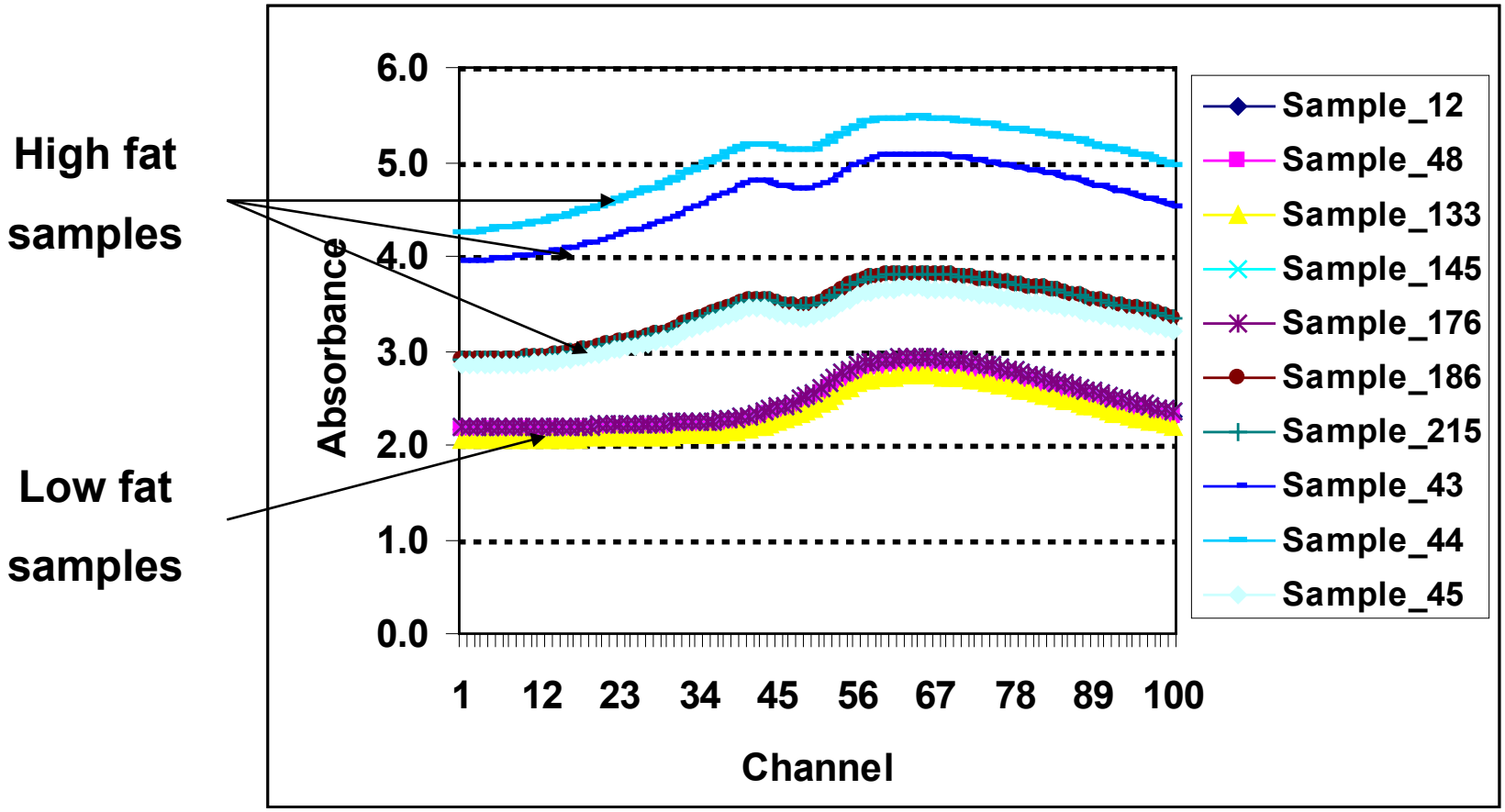




Which components contribute to PC1-2?

Absorbance records for ten samples of chopped meat

PCA2 captures the most of remaining variation



Probabilistic PCA

- z_i-latent variables, x_i- observed variables

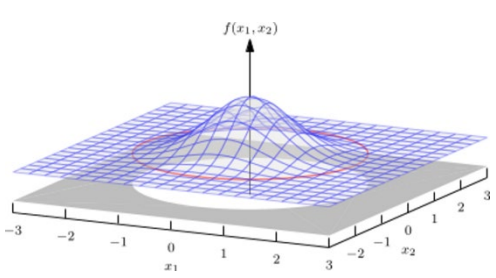
$$\boldsymbol{z} \sim \boldsymbol{N}(0, \boldsymbol{I})$$

$$\boldsymbol{x}|\boldsymbol{z} \sim \boldsymbol{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2 \boldsymbol{I})$$

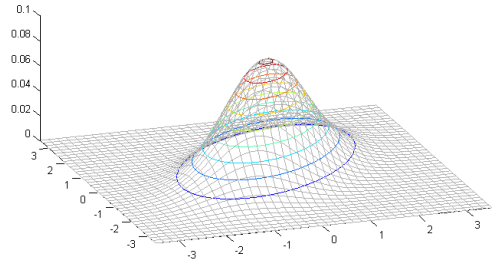
- Alternatively

$$\boldsymbol{z} \sim \boldsymbol{N}(0, \boldsymbol{I}), \boldsymbol{x} = \boldsymbol{\mu} + \boldsymbol{W}\boldsymbol{z} + \boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \boldsymbol{N}(\boldsymbol{0}, \boldsymbol{\sigma^2 I})$$

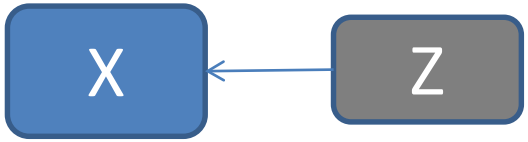
- **Interpretation**: Observed data (X) is obtained by rotation, scaling and translation of standard normal distribution (Z) and adding some noise.



Z



X



Probabilistic PCA

- **Aim**: extract Z from X
- Distribution of x:

$$\boldsymbol{x} \sim \boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{C})$$
$$\boldsymbol{C} = \boldsymbol{W}\boldsymbol{W}^{\boldsymbol{T}} + \boldsymbol{\sigma}^{\boldsymbol{2}}\boldsymbol{I}$$

- Rotation invariance
  - Assume that x was generated from $z' = Rz, RR^T = I,\ p(x)$ does not change!

$$\boldsymbol{x}|\boldsymbol{z}' \sim \boldsymbol{N}(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z}' + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$$

  - **Model will not be able find latent factors uniquely! ☹**
    - It does not distinguish z from z'

Probabilistic PCA

- Estimation of parameters: ML

Theorem. ML estimates are given by

$$\mu_{ML} = \bar{x}$$

$$W_{ML} = U_M(L_M - \sigma_{ML}^2 I)^{\frac{1}{2}}R$$

$$\sigma_{ML}^2 = \frac{1}{p-M}\sum_{i=M+1}^{p}\lambda_i$$

- U_M matrix of M eigenvectors
- L_M diagonal matrix of M eigenvalues
- R any orthogonal matrix

Probabilistic PCA

- Estimation of Z
  - Use mean of posterior

$$\hat{z} = (W_{ML}^T W_{ML} + \sigma_{ML}^2 I)^{-1} W_{ML}^T (x - \mu)$$

- Connection to standard PCA
  - Assume $R = I, \sigma^2 = 0$→get standard PCA components scaled by inverse root of eigenvalues

$$Z = XUL^{-\frac{1}{2}}$$

Advantages of probabilistic PCA

- More settings to specify→ more flexible
- Can be faster when M<<p
- Missing values can be handled
- M can be derived if a Bayesian version is used
- Probabilistic PCA can be applied to classification problems directly
- Probabilistic PCA can generate new data

Probabilistic PCA in R

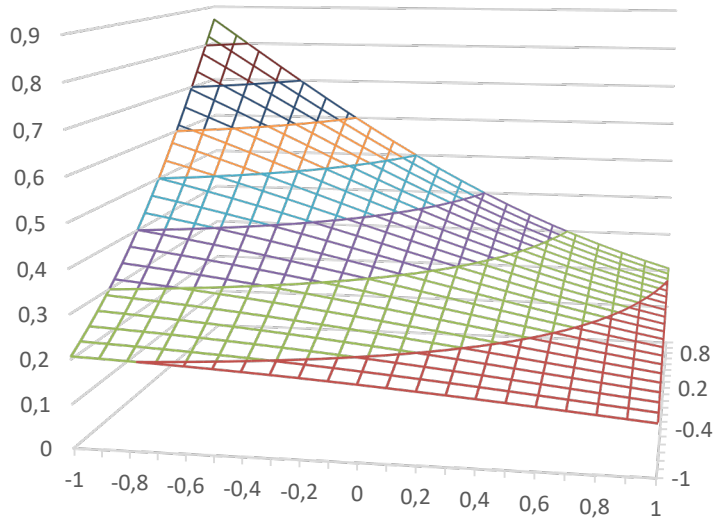
- Use **pcaMethods** from Bioconductor
- Install
  - source("https://bioconductor.org/biocLite.R")
  - biocLite("pcaMethods")

Ppca(data, nPcs,…)

Results: scores, loadings…

Independent component analysis (ICA)

- Probabilistic PCA does not capture latent factors
  - Rotation invariance
- Let's choose distribution which is not rotation invariant→will get unique latent factors
- Choose non-Gaussian $\mathrm{p}(z_i)$
- Assuming latent features are **independent**



$$p(z) = \prod_{i=1}^{M} p(z_i)$$

$$p(z_i) = \frac{2}{\pi(e^{z_i} + e^{-z_i})}$$

ICA

- Model

$$x = \mu + Wz + \epsilon, \qquad \epsilon \sim \boldsymbol{N}(0, \Sigma)$$

- **Estimation : Maximum likelihood** ($V = W^{-1}$)
  - Assuming noise-free x

$$\max_{V} \sum_{i=1}^{n} \sum_{j=1}^{p} \log\left(p_j\left(v_j^T x_i\right)\right)$$

$$\text{Subject to } \left|\left|v_i\right|\right| = 1$$

ICA: estimation algorithm

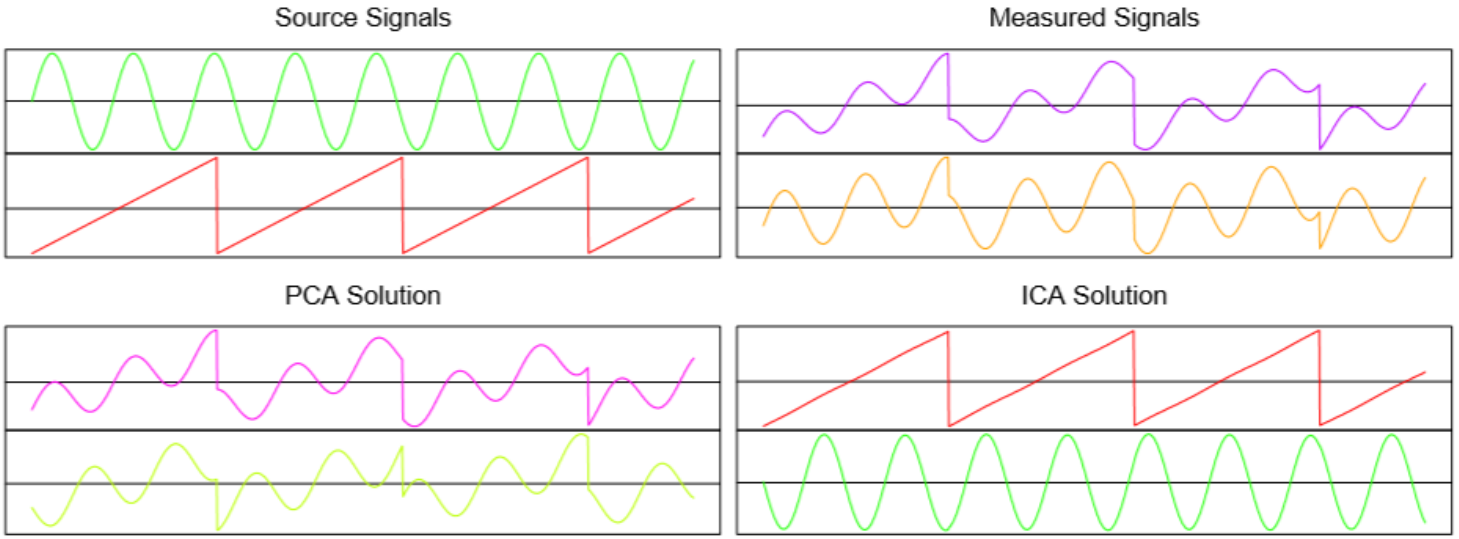
1. Estimate V by maximum likelihood
2. Compute $Z = X'V$

- **With prewhitening**
  1. Convert X into PCA coordinate system (do not remove dimensions): $X' = XU$
  2. Estimate V by maximum likelihood in ICA
  3. Estimate final scores $Z = X'V$

  – Note: full transformation matrix is $U_{ICA} = U \cdot V$

ICA

- Example

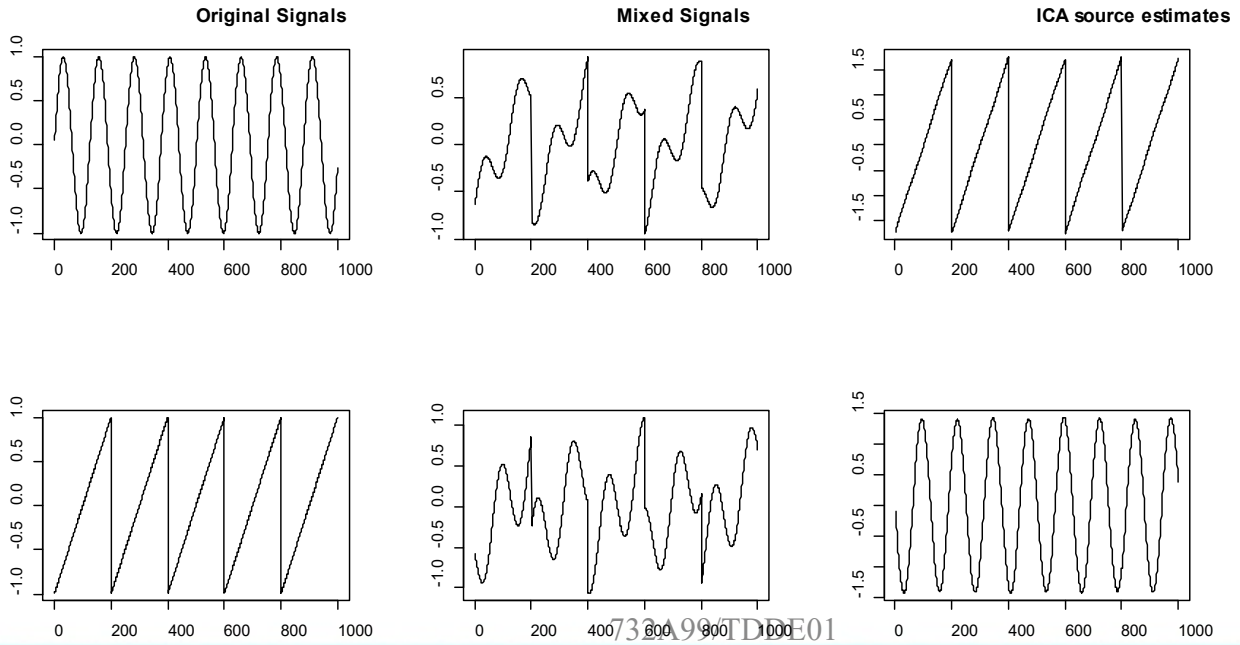


Source: Elem of stat learn by Hastie

Independent component analysis: R

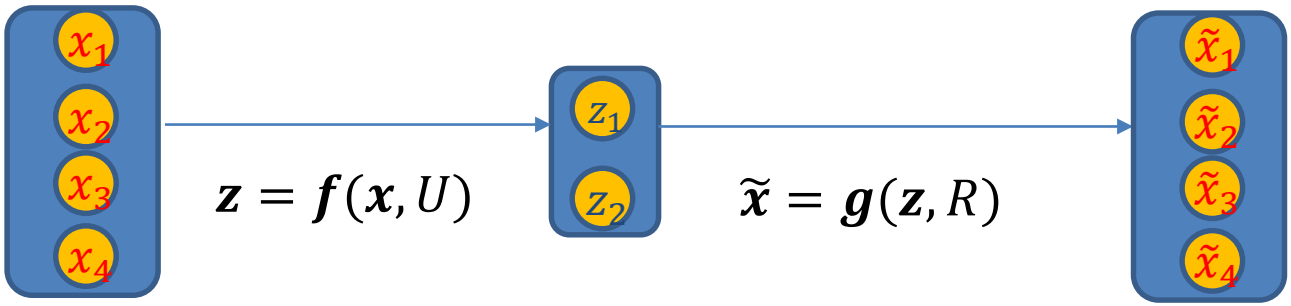
R package: fastICA

```
S <- cbind(sin((1:1000)/20), rep((((1:200)-100)/100), 5))
A <- matrix(c(0.291, 0.6557, -0.5439, 0.5572), 2, 2)
X <- S %*% A #mixing signals
a <- fastICA(X, 2) #now separate them
```



Autoencoders (nonlinear PCA)

- Why linear transformations? Take nonlinear instead!
- $f()$ and $g()$ are typically Neural Networks



$$\min_{U,R} \sum_{i=1}^{N} \|x_n - \tilde{x}_n\|^2$$

…or some other loss function