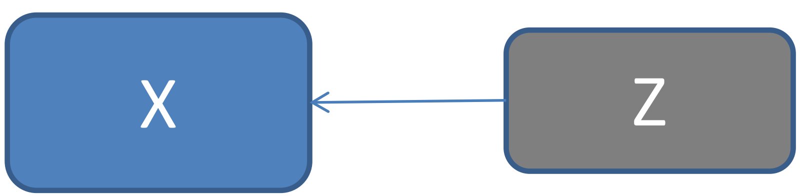
Dimensionality reduction

Lecture 2c

Latent variables

- Sometimes data depends on the variables we can not measure (or 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

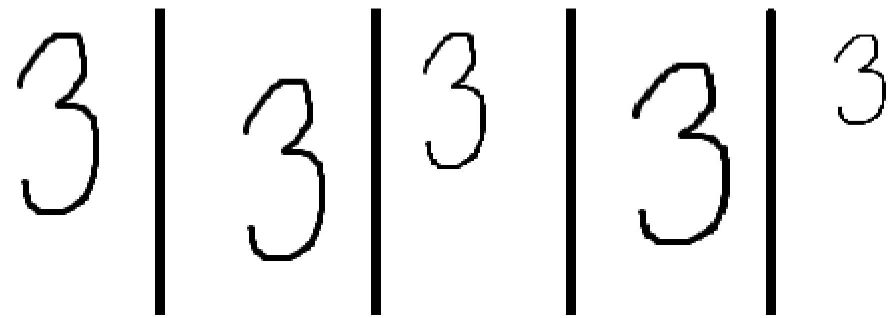


Source: Leadliaison.com

$$X = g(Z)$$

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 **feature reduction** / **representation learning** technique, aims to learn latent features from x: $\tilde{z} = f(x)$

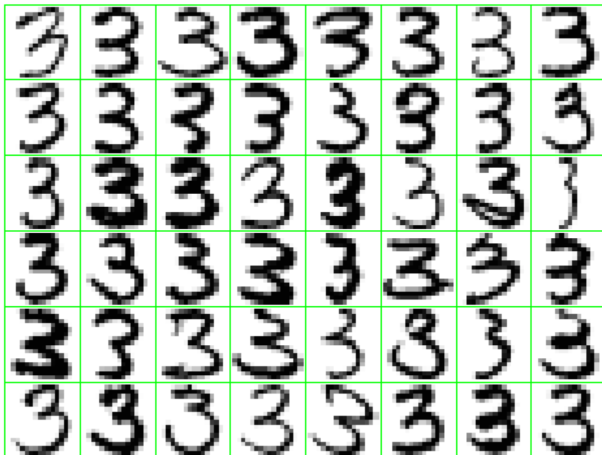
- Used to approximate high dimensional data with a few **informative** features –> much less data to store

Applications

- Industry (sensors)
- Medicine (genes)
- Text analysis (word counts)
- …

Principal Component Analysis (PCA)

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



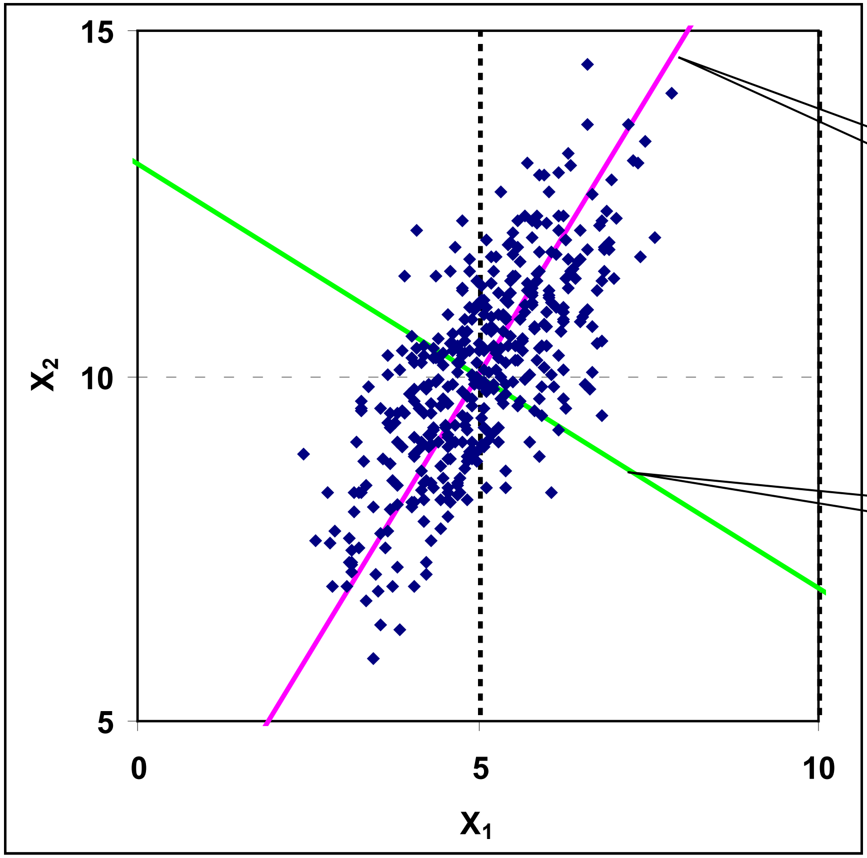
Principal components analysis

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



Principal component analysis

- Assume features have mean zero
- Aim: maximize variance of projected data
  - Sample covariance matrix $S = \frac{1}{n}X^T X$
- **Mathematical objective**

$$\max_{u^T u=1} u^T S u$$

- Optimal solution found by eigenvalue decomposition $Su = \lambda u$ with maximum λ

PCA: computations

Data

$$T = \|\boldsymbol{x}^{\mathbf{1}} \ \dots \ \boldsymbol{x}^{\boldsymbol{p}}\|, \quad \boldsymbol{x}^{j} = (x_{1j}, \dots, x_{nj})$$

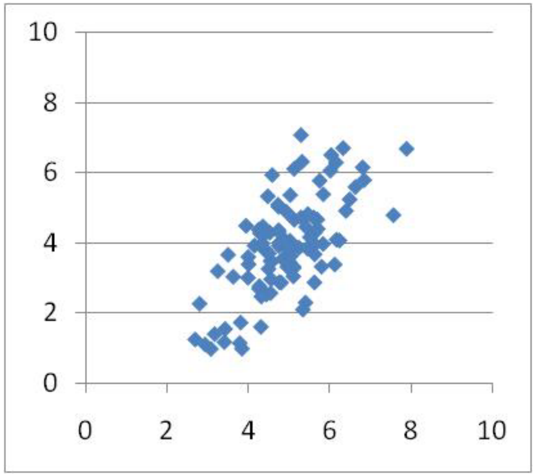
1. Centred data

$$\boldsymbol{X} = \|\boldsymbol{x}^1 \ - \overline{\boldsymbol{x}}^1 \ \boldsymbol{x}^2 \ - \overline{\boldsymbol{x}}^2 \ \dots \ \boldsymbol{x}^p \ - \overline{\boldsymbol{x}}^p \ \|,$$

2. Covariance matrix

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

3. Search for eigenvectors and eigenvalues of **S**
   – Equivalent: SVD of $\boldsymbol{X}$

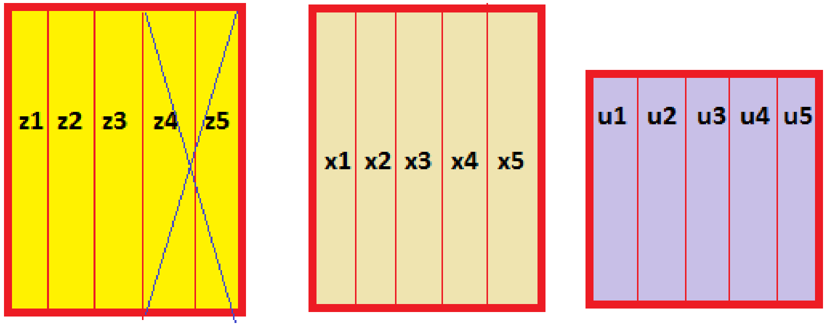


	Column 1	*Column 2*
Column 1	0.951	0.905
Column 2	0.905	1.883

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 *q:*
   $$Z = X\ U_q$$

6. New data will have dimensions n x q instead of n x p

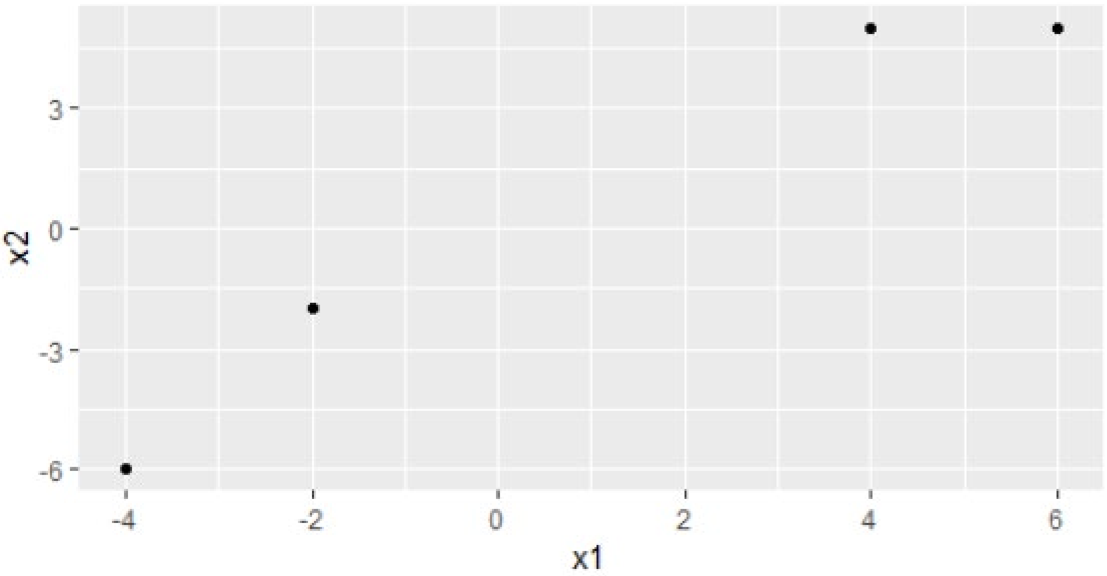
Store: n x q+ p x q
instead n x p

100*50 vs
100*4+50*4

Getting approximate original data:
$$\tilde{X} = ZU_q^T + \|\bar{\boldsymbol{x}}^1 \quad \bar{\boldsymbol{x}}^2 \quad \ldots \quad \bar{\boldsymbol{x}}^p\ \|$$

Example

x_1	x_2
-4	-6
-2	-2
4	5
6	5



Example

- Centered data

x_1	x_2
-5	-6.5
-3	-2.5
3	4.5
5	4.5

- Eigenvectors of S

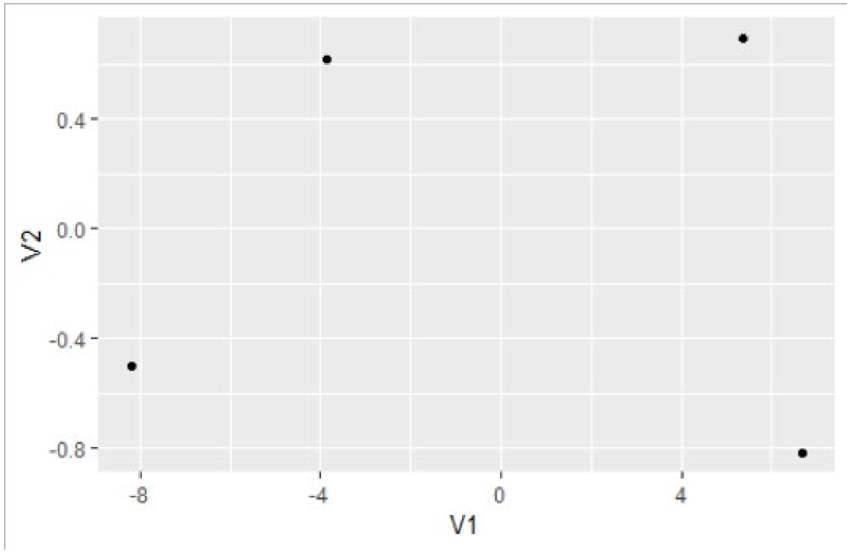
```
> eigen(S)
eigen() decomposition
$values
[1] 38.8054751  0.4445249

$vectors
          [,1]       [,2]
[1,] 0.6569407 -0.7539423
[2,] 0.7539423  0.6569407
```

Example

- Expressing new coordinates
  - $PC1: z_1 = 0.66x_1 + 0.75x_2$
  - $PC2: z_2 = -0.75x_1 + 0.66x_2$
  - Which component has largest contribution to 1st PC?

- Scores

```
> Z
          [,1]       [,2]
[1,] -8.185328 -0.5004029
[2,] -3.855678  0.6194752
[3,]  5.363562  0.6944062
[4,]  6.677444 -0.8134784
```



Example

- Discarding PC2
  - New data

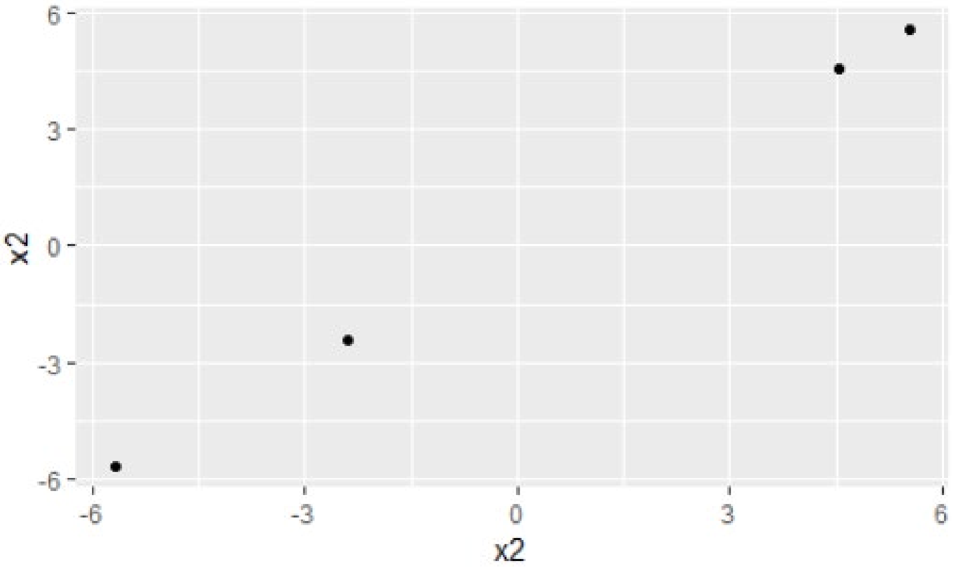
z_1
-8.19
-3.86
5.36
6.68

- How much do we store now?
  - 4*1+2*1=6 (and two numbers for mean values)
  - Have we reduced the storage?

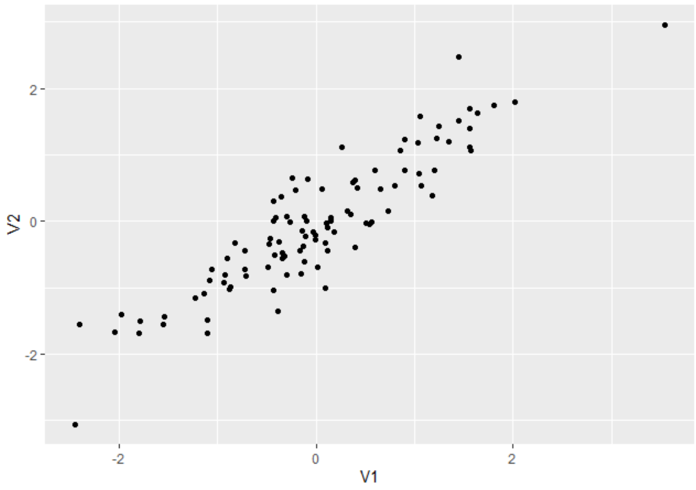
Example

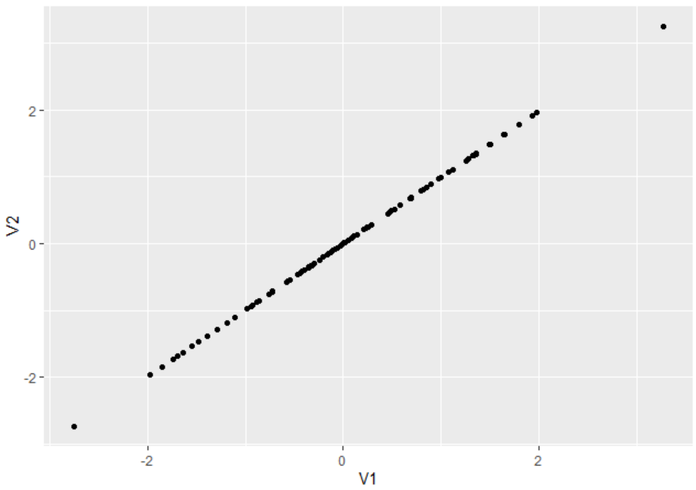
- Approximate original data
  - $\bar{x}^1 = 1, \bar{x}^2 = 0.5$

```
           [,1]      [,2]
[1,] -4.377275 -5.671265
[2,] -1.532951 -2.406958
[3,]  4.523542  4.543816
[4,]  5.386684  5.534407
```



Example: more data



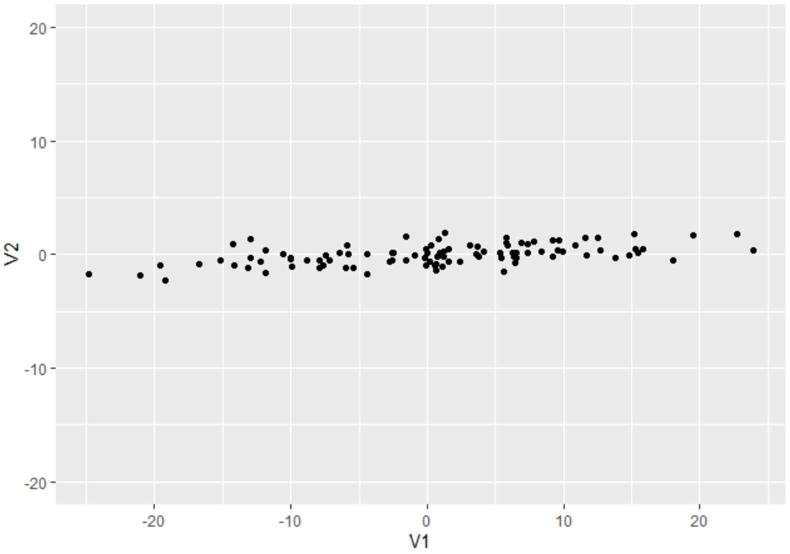


- Data became approximate (but less data to store)

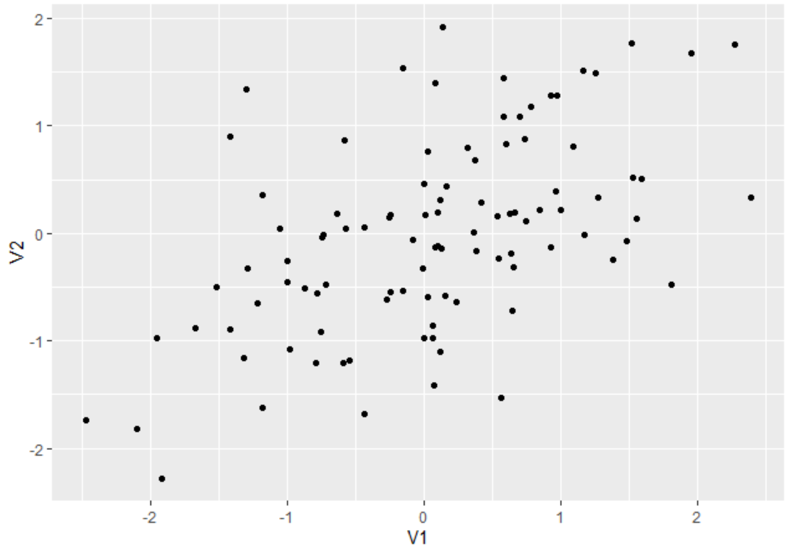
PCA and scaling

- Do we need to scale features?

Without scaling

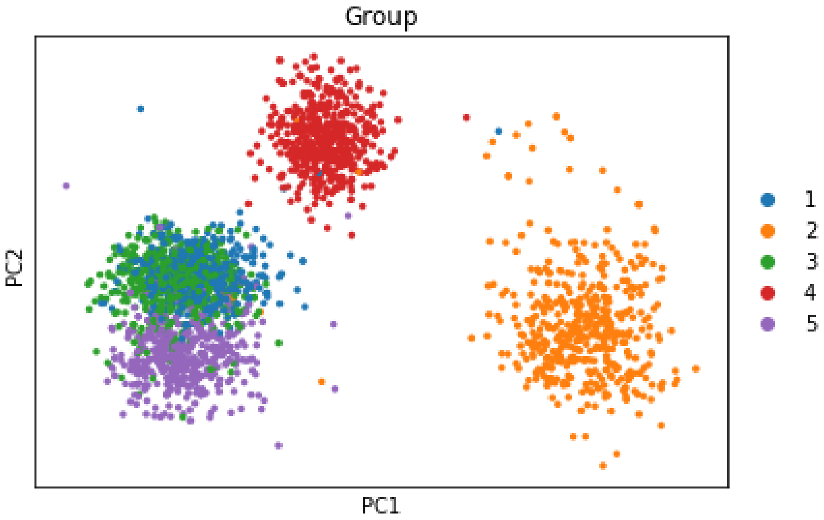


After scaling



PCA

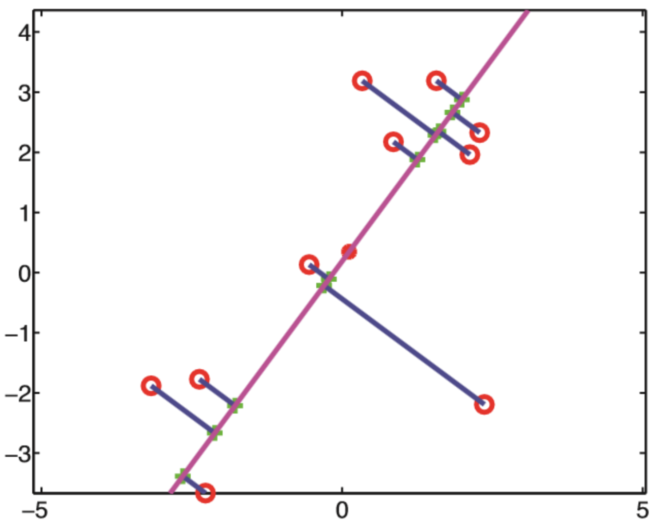
- Reducing into 2 dim can enable studying structures
  - Example: gene expression data (20000 genes, 2500 cells)



PCA: equivalent formulation

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

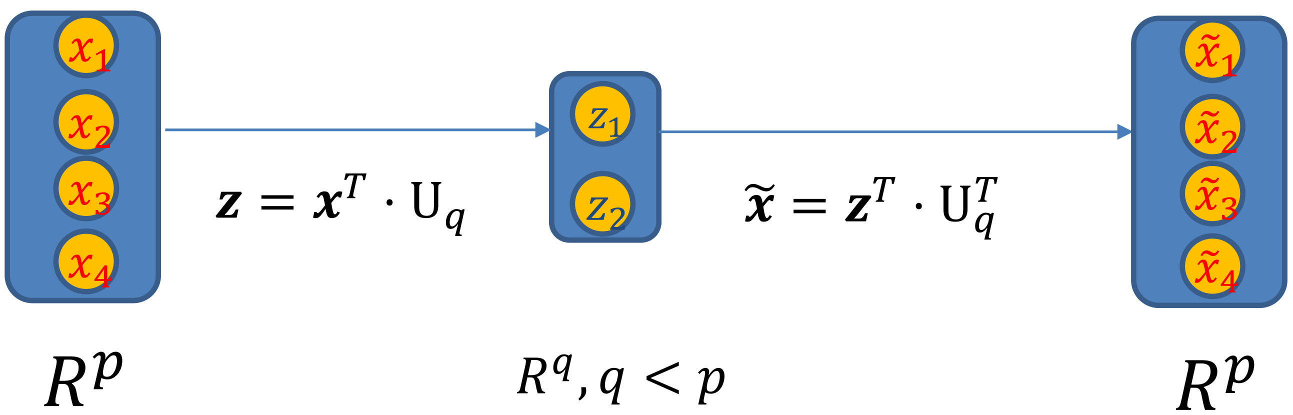
$$\min_{U_M} \sum_{i=1}^{N} \|x_n - \tilde{x}_n\|^2$$



Source: Murphy

PCA: computations

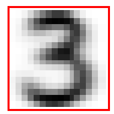
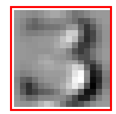
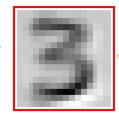
- PCA makes a **linear** compression of features



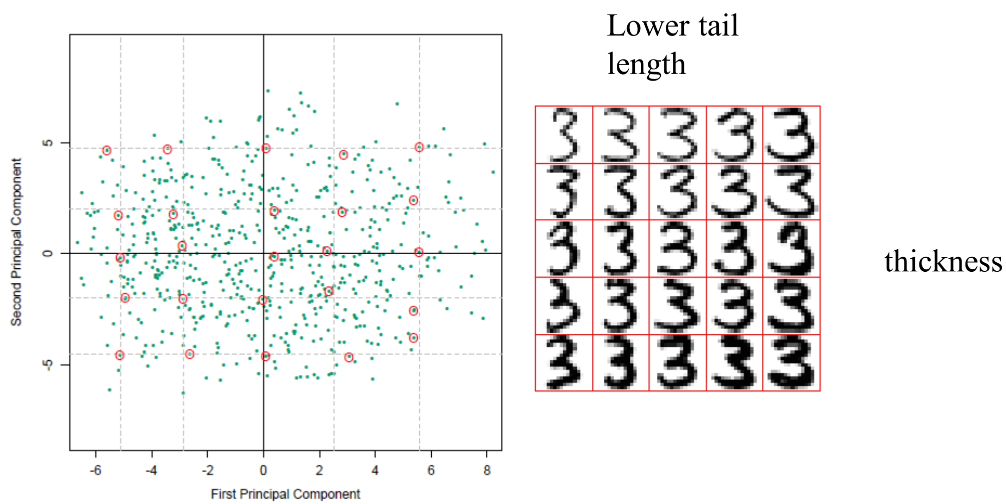
$$\min_{U_q} \sum_{i=1}^{n} \|x_n - \tilde{x}_n\|^2$$

Principal Component Analysis

- Digits: two eigenvectors extracted

x =  + z1 ·  + z2 ·  .

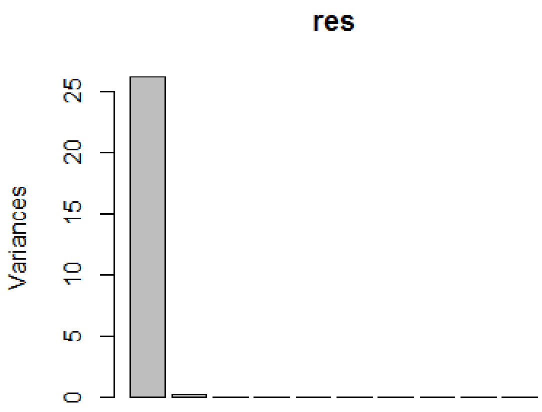
- Interpretation of eigenvectors



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
```

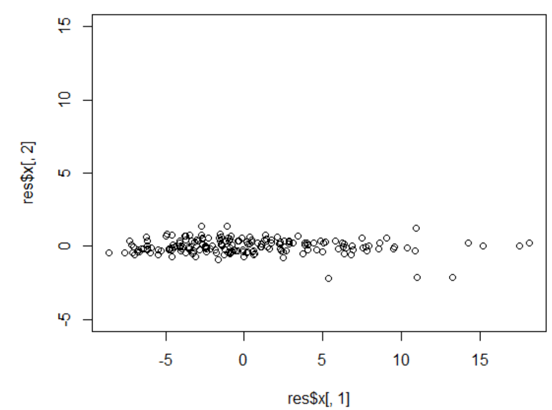
Only 1 component captures the 99% of variation!

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)**

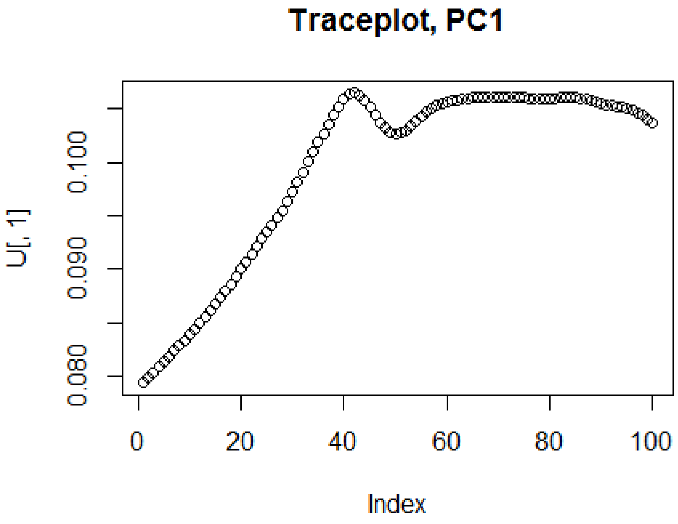
plot(res$x[,1], res$x[,2], ylim=c(-5,15))

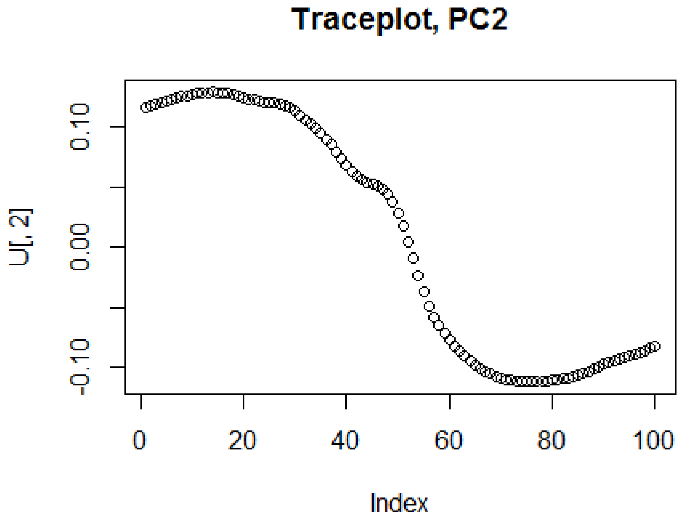
Do we need the second dimension?

PCA in R

- **Trace plots**

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

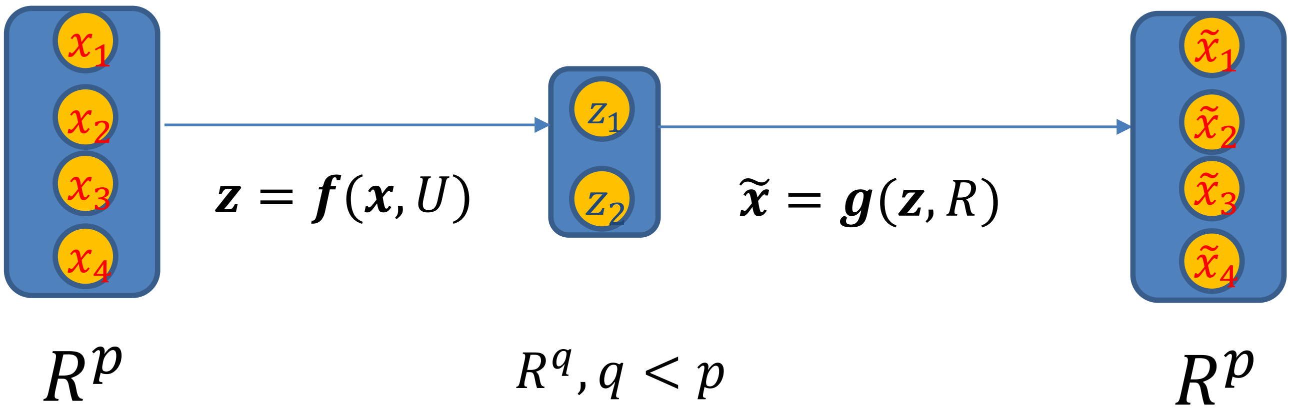




Which components contribute to PC1-2?

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 cost function

Other linear representation learning methods

- Probabilistic PCA
  - Similar to PCA but has more opportunities
    - Can be used to handle missing values directly
    - Can be easily embedded in Bayesian ML models
    - Can be used to generate new data

- Independent component analysis (ICA)
  - Sometimes shows better emprical results compared to PCA

Probabilistic PCA

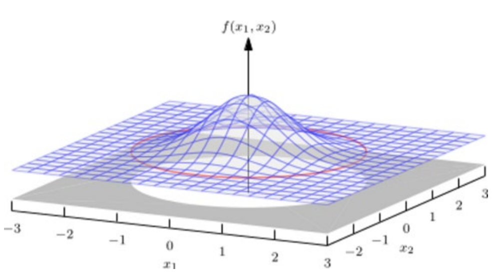
- z_i-latent variables, x_i- observed variables

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

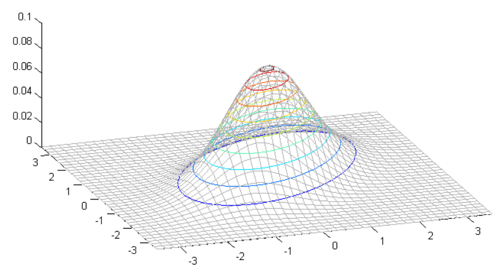
- Alternatively

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

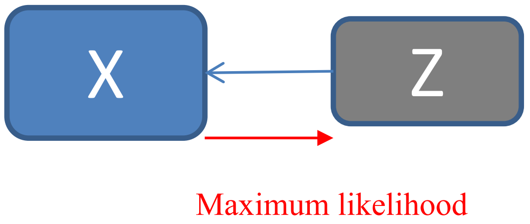
- **Interpretation**: Observed data (X) is obtained by rotation, scaling and translation of standard normal distribution (Z) and adding some noise.



Z

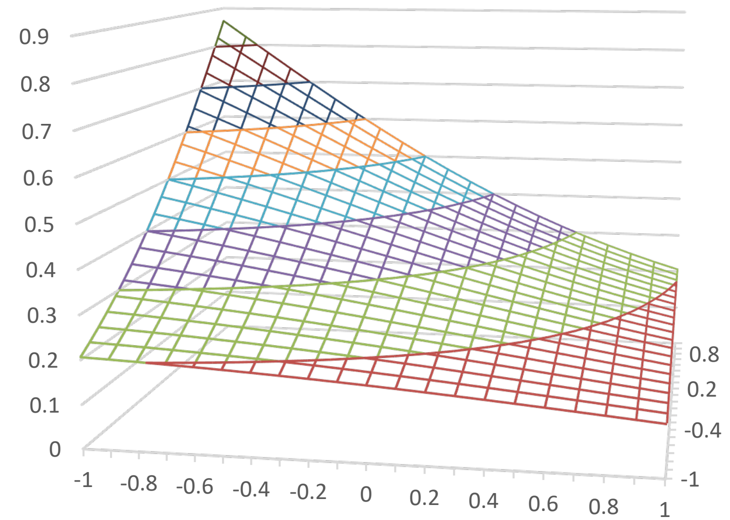


X



Independent component analysis (ICA)

- Probabilistic PCA does not capture latent factors uniquely
  - 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**

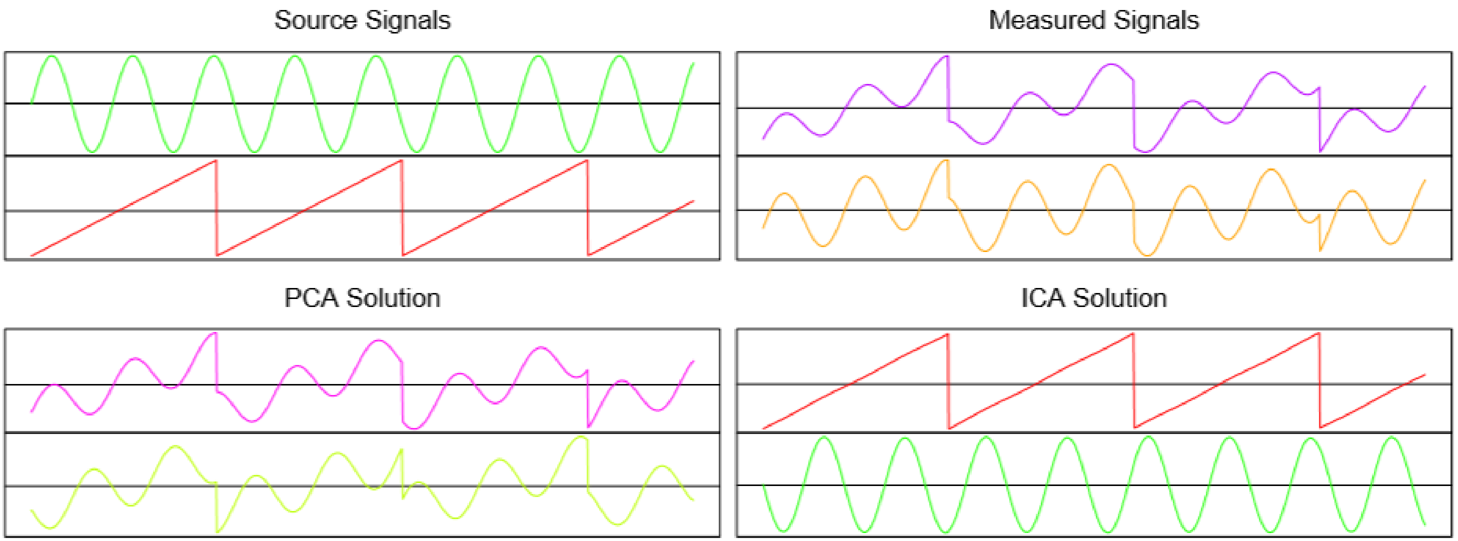


$$p(z) = \prod_{i=1}^{M} p(z_i)$$

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

ICA

- Example



Source: Elem of stat learn by Hastie