B4M36SAN
BE4M36SAN
Seminar/Tutorial

Alikhan Anuarbekov

Ali

anuarali@fel.cvut.cz

A little more details for Midterm test

3 Nonlinear Regression (3p.)

Write the definition of polynomial regression relation for a single independent variable. Explain its advantages and disadvantages. Describe how you would determine the optimal degree of the model. Could this model encounter the problem of multicollinearity, and why? Can we use more than one independent variable in this model? Can polynomial regression be applied to classification tasks?

Formula: $$Y = \beta_0 + \beta_1 \cdot X + \beta_2 \cdot X^2 + \beta_3 \cdot X^3 + \ldots + \beta_P \cdot X^P + \epsilon,$$
$$\epsilon \in N(0, \sigma^2)$$

Advantages: + You could represent any continuous non-linear shape with large enough degree

Disadvantages: - If shape is too complex, you will 100% get an over-fit (6 or larger degree)
- If shape is discontinuous, you will hardly model it with a simple polynomial
-

How to find an optimal degree? Recap previous seminar beginning:

An approach to select an optimal degree of the polynomial

4.2.3) Finding out what is the optimal model among possibilities:

Option B:

find REASONABLE models and test them in a for loop via some criterion (AIC, BIC, CrossValidation, R^2)

Given that you compute number of ***smooth optima/inflex points*** and ***not-so-smooth points***,

Try all of previous model types: linear, step, polynomial, spline, ….

But with **degree** and **knots** adjusted according to ***smooth/optima/inflex*** and ***not-so-smooth points***

For example:

Try one single polynomial with degree = smooth optima/inflex points

Divide smooth points to intervals of 2-3, e.g. for lets say 12 smooth points use 4 polynomials of degree 3

Multicollinearity makes coefficient unstable --> transform to orthogonal polynom (Just mention, not formula!)

$$X_1 = X$$
$$X_2 = X^2$$
$$\dots$$
$$X_P = X^P$$

$$\vec{x} = \begin{pmatrix} 1 \\ 2 \\ \dots \\ 100 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1^2 & 1^3 \\ 2 & 2^2 & 2^3 \\ .. & \dots & \dots \\ 100 & 100^2 & 100^3 \end{pmatrix}$$

But they are correlated!
May cause some problem,since:

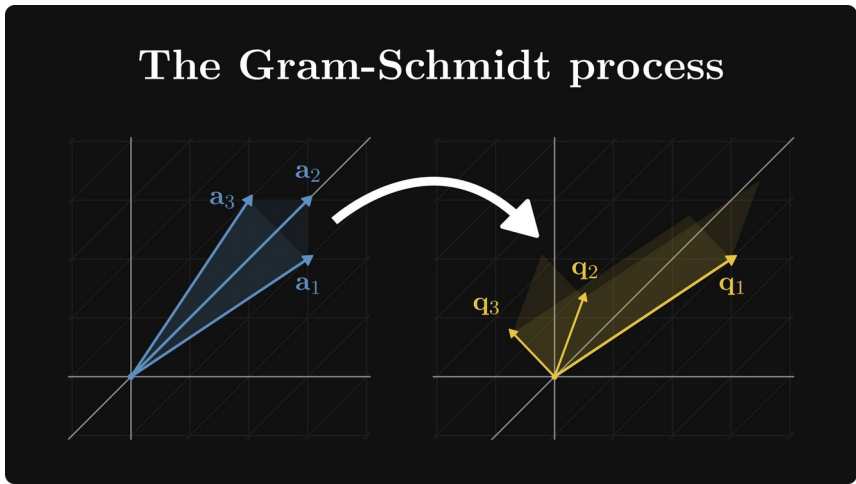
correlation = broken assumption of LM

$$\vec{x} = \begin{pmatrix} 1 \\ 2 \\ \dots \\ 100 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1^2 & 1^3 \\ 2 & 2^2 & 2^3 \\ .. & \dots & \dots \\ 100 & 100^2 & 100^3 \end{pmatrix} \rightarrow \text{Gramm-Schmidt ortog} \rightarrow \begin{pmatrix} -0.17 & 0.22 & -0.25 \\ .. & \dots & \dots \\ .. & \dots & \dots \\ 0.15 & 0.14 & 0.21 \end{pmatrix} = \vec{X}_{ortog}$$

Even though they are changed = changed coefficients,

but their predictions are the same!

predict(lm with x coefs) = predict(lm with x_ortog coefs)



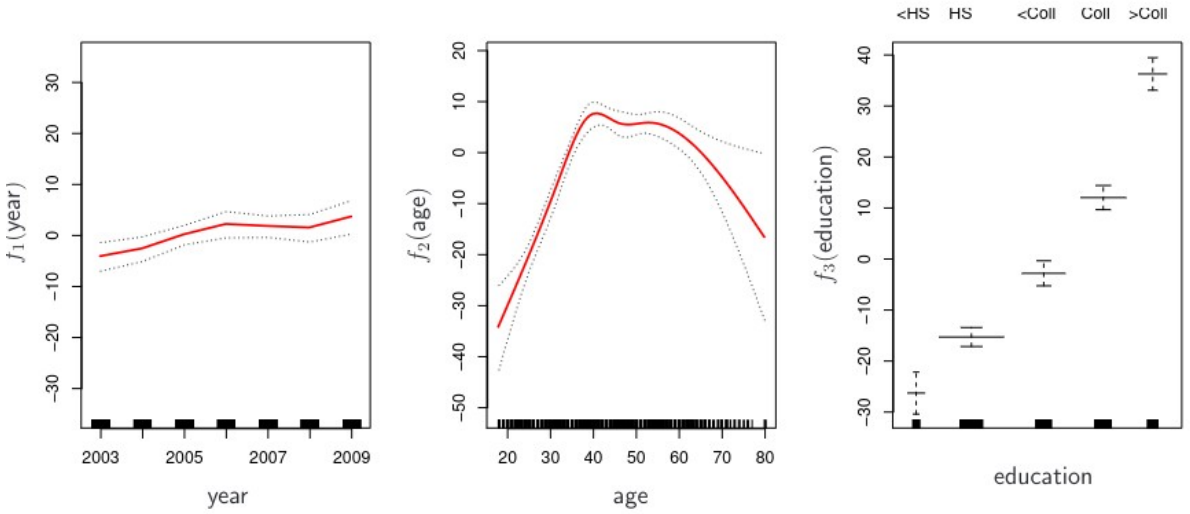
More than one variable in polynomial regression:

directly (+ possible interaction terms):

$$Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_1^2 + \beta_3 \cdot X_2 + \beta_4 \cdot X_2^2 + \ldots + \beta_P \cdot X_K^P + \epsilon,$$
$$\epsilon \in N(0, \sigma^2)$$

GAMs (+ possible interaction terms):

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i$$



Use of polynomial regression in classification:

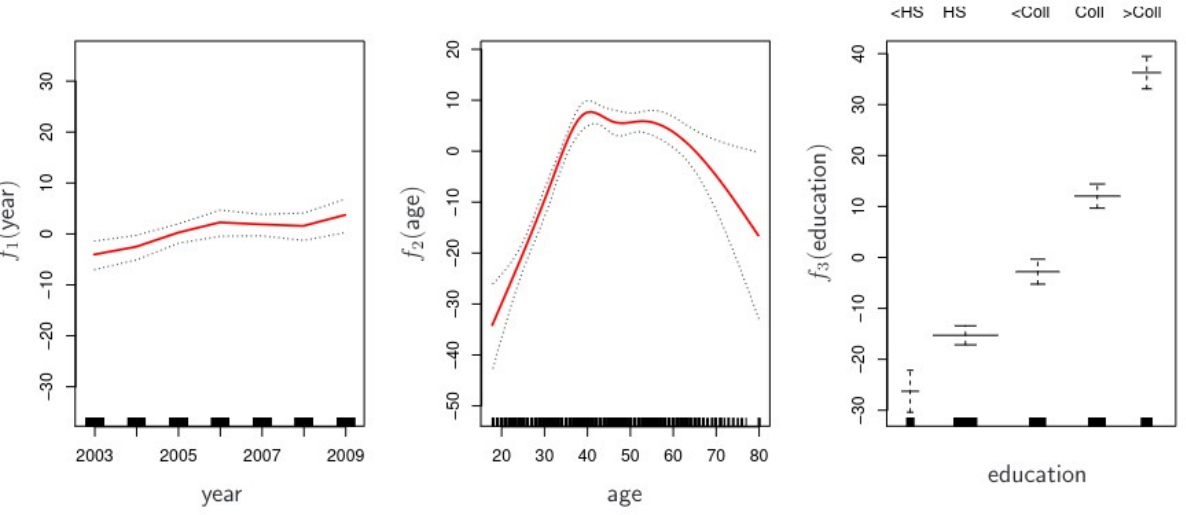
Just use logistic regression + GAM:

$$logit\, Y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_1^2 + \beta_3 \cdot X_2 + \beta_4 \cdot X_2^2 + \ldots + \beta_P \cdot X_K^P + \epsilon,$$

$$Y = \frac{1}{\exp\left(-1\left(\beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_1^2 + \beta_3 \cdot X_2 + \beta_4 \cdot X_2^2 + \ldots + \beta_P \cdot X_K^P\right)\right)} + \epsilon,$$

$$\epsilon \in Binomial\left(p\right)$$

$$y_i = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon_i$$

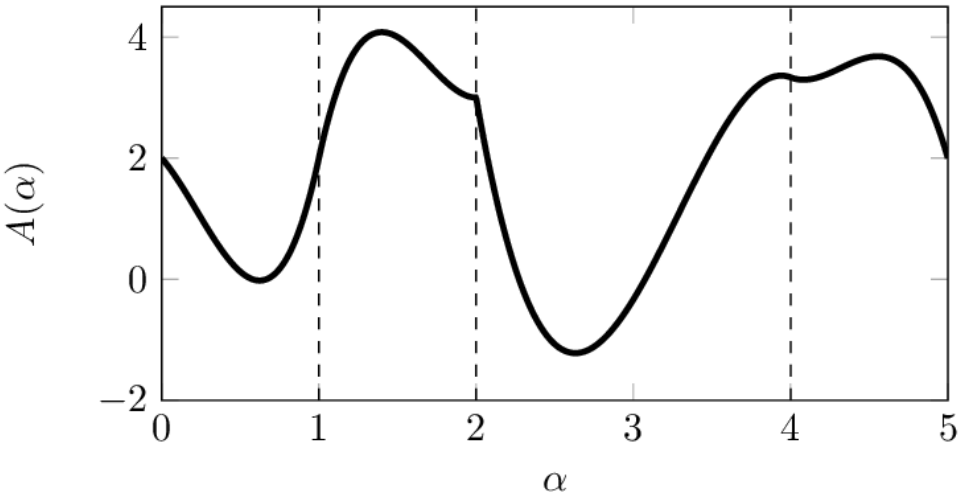


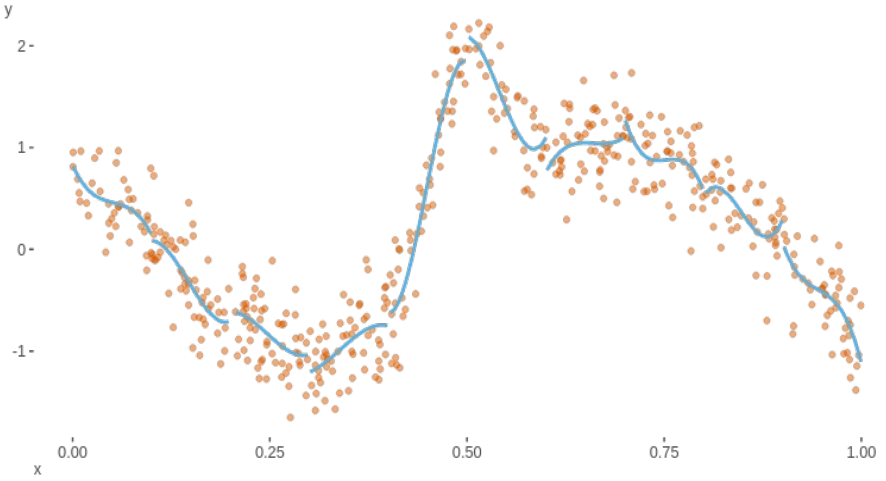
3 Nonlinear Regression (~~2p~~) 3p

Splines typically work with knots. First, explain what a knot is and why introducing knots can be advantageous. Then, explain how knots can be set/computed/estimated. Finally, name three approaches to nonlinear regression that do not use knots. When would you apply these methods in practice (consider them all as together first and then individually)?

Knot: A point that separates two intervals with different polynomial models. Typically knots are required to follow continuity from one model curve to another.

Knots are needed to overcome over-fit issues – instead of large, lets say 12th degree polynomial, we will fit 4 polynomials of 3th degree



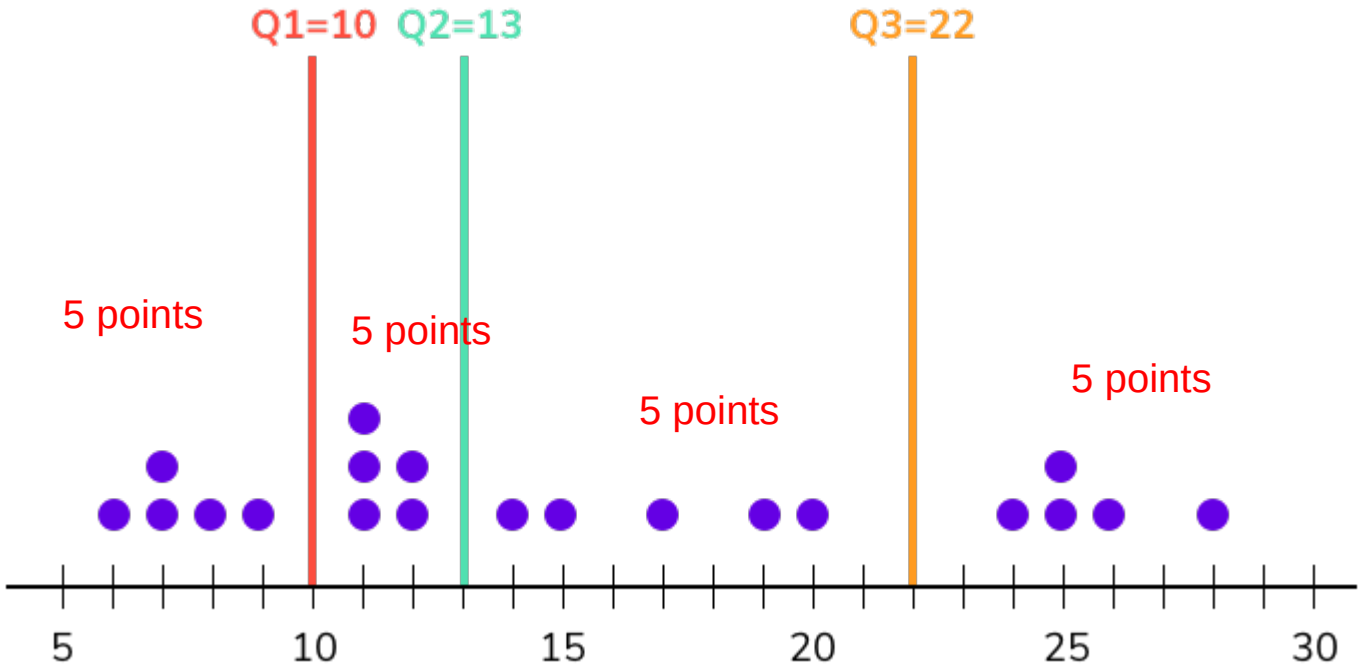


We can choose the position of knots if number of them is given:

quantiles =

such that each interval between knots has the same number of points

> optimality control via Cross-Validation



A bit problematic and ambiguous question, so I did not penalized it hardly

Nonlinear regression models:

WITH KNOTS: Piece-wise polynomial, Spline, Step function,
Smoothing spline (knots in every points, but I would have taken it as well as correct answer)

WITHOUT KNOTS: single Polynomial, local regression,
~ GAMs/GLMs also count

Additional tip:

- **Try not to answer with long text, I could read it and give you points for Midterm, but on Final exam you should be answering with formulas and short statements only!**
  - **Maybe only this question about methods without knots could be accepted in this text-discussion form**
- **This is a math course, you should answer formally and exactly**
  - **DEFINITION = short, formula + description of elements**
  - **EXAMPLES = short, only numbers/particular formula, not text**

Final Assignment, few comments

- **Today (17.11.2024) is the deadline for proposal = few sentences about what you are planning to do for Final assignment**
- **Next week (24.11.2024) is the deadline for Work Plan = detailed description of the task you are going to solve**
- **In two weeks (01.12.2024) is the deadline for consultation possibility**

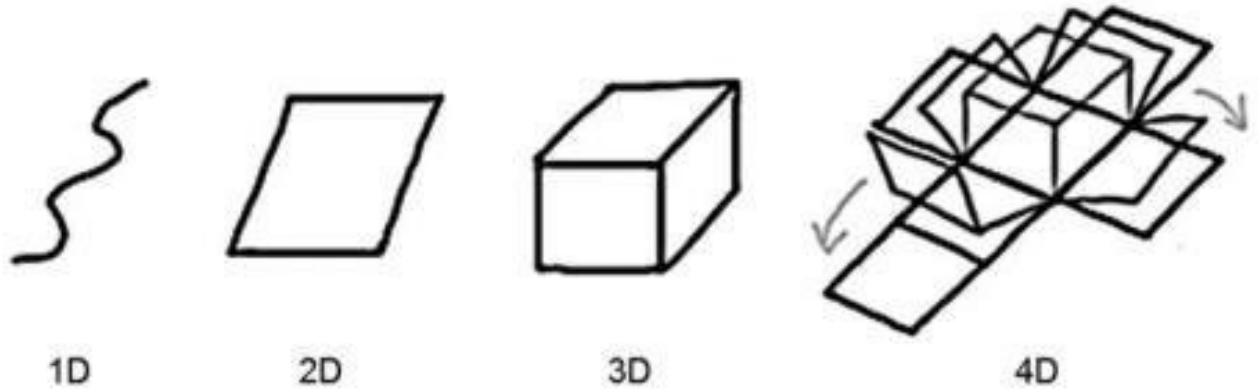
- **The deadlines are soft till 01.12.2024, you could submit it later than deadlines, but if you do not submit anything till 01.12.2024, you will get huge penalties**

- **You do not have to participate in personal consultations, I will send you the comments and feedback after you submit the proposal/work plan in BRUTE**

- **If you are doing Cybersecurity, everything (grading, format, etc.) is done by Tomas Pevny, only final presentation would be with everybody else**

Dimensional Reduction

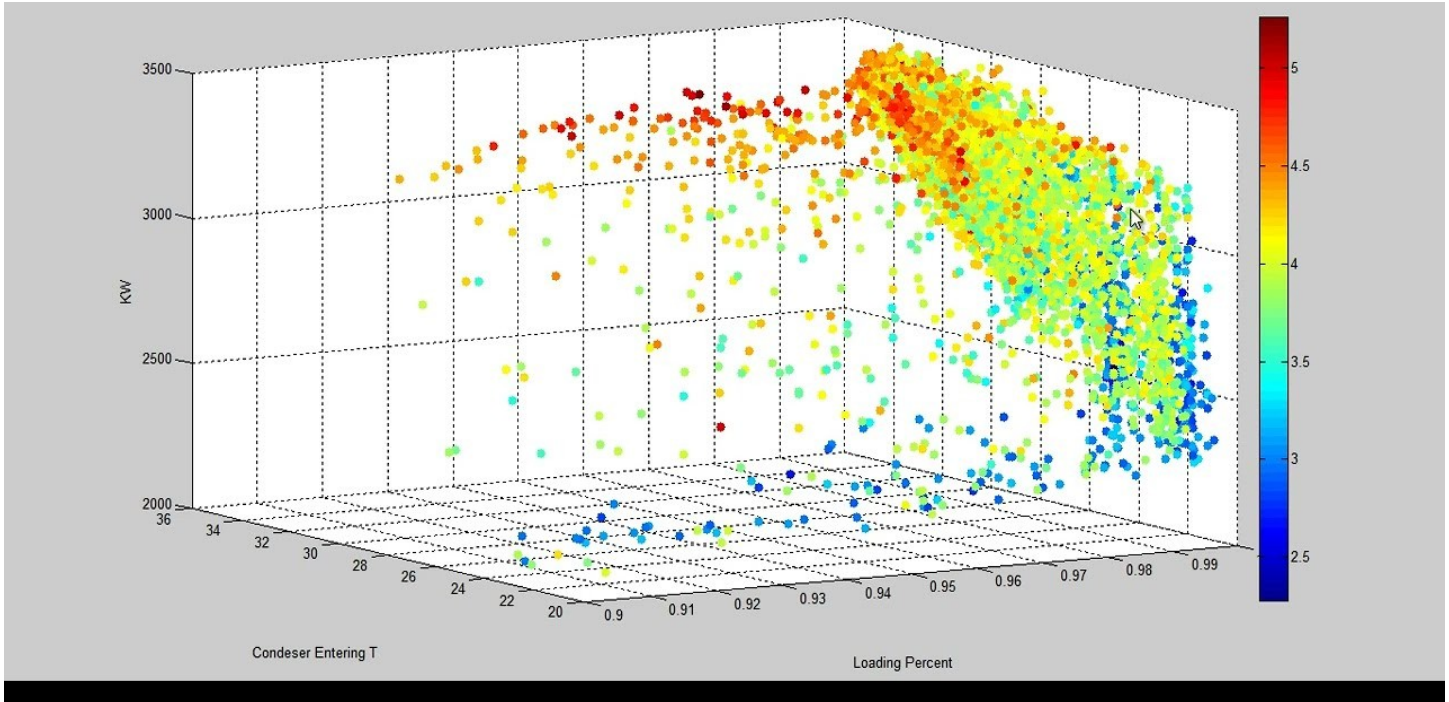
If we have many X variables, we can't even visualize them in dimension larger than 3D (4D = color trick)



In practice, I can observe
10D space pretty easily:
(+ 1D for each property of an object)

Sales order fact

Sales id
Product id
Customer id
Quantity
Price
Date of order
Date of delivery
Date of transaction
Date of cancellation
Year



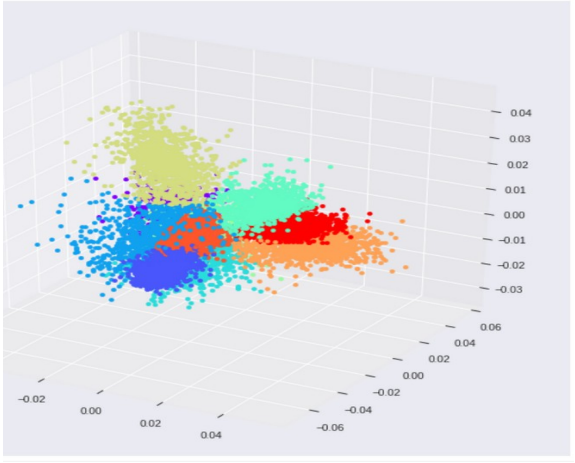
But what if we have more than 2 X variables? We can't even visualize them in dimension larger than 3D

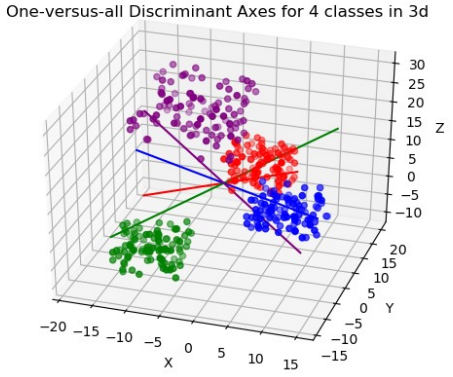
If we assume that all classes have same $LDA: \sum_1 = \sum_2$

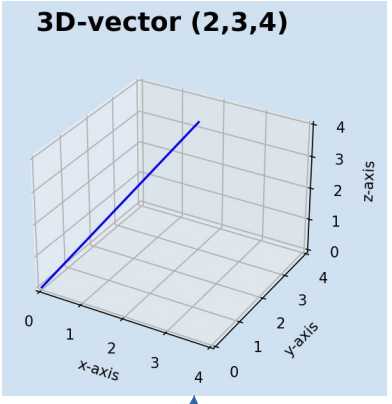
Fisher's discriminant plot
(Linear dimension reduction, HW 2)

Then we can take every pair and visualize their decision boundary:

Eigenvector/ Eigenvalue (linear algebra)







For every pair, take their mean centers and connect with a vector

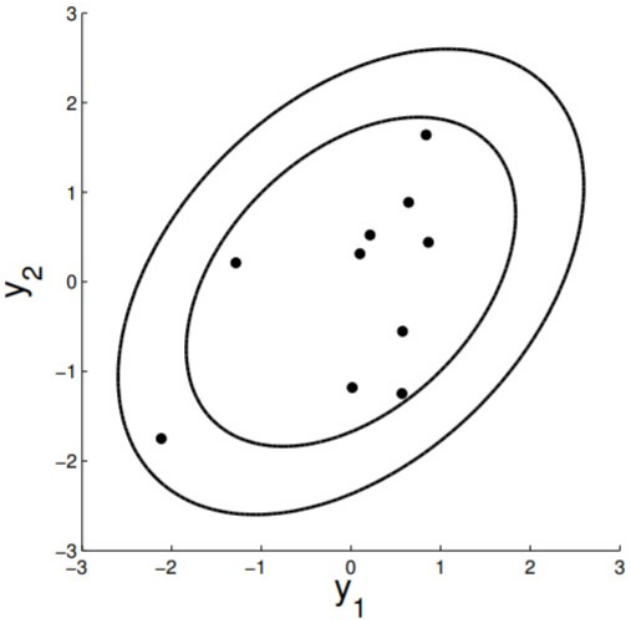
Each connection is a line in D dimensional space

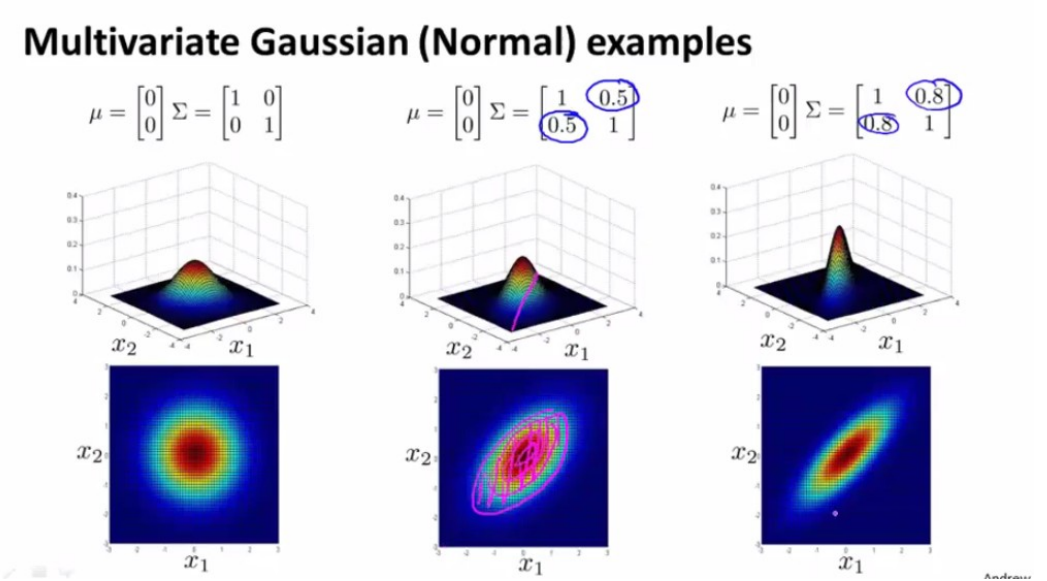
If we project along 1-2 largest eigenvectors, Then we can get a clean projection in 2D/3D

Alternatively, **PCA = Principle Component Analysis**

Remember Gaussian covariance(normalized correlation) matrix rotation?

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^\intercal\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .4 \\ .4 & 1 \end{bmatrix}$$





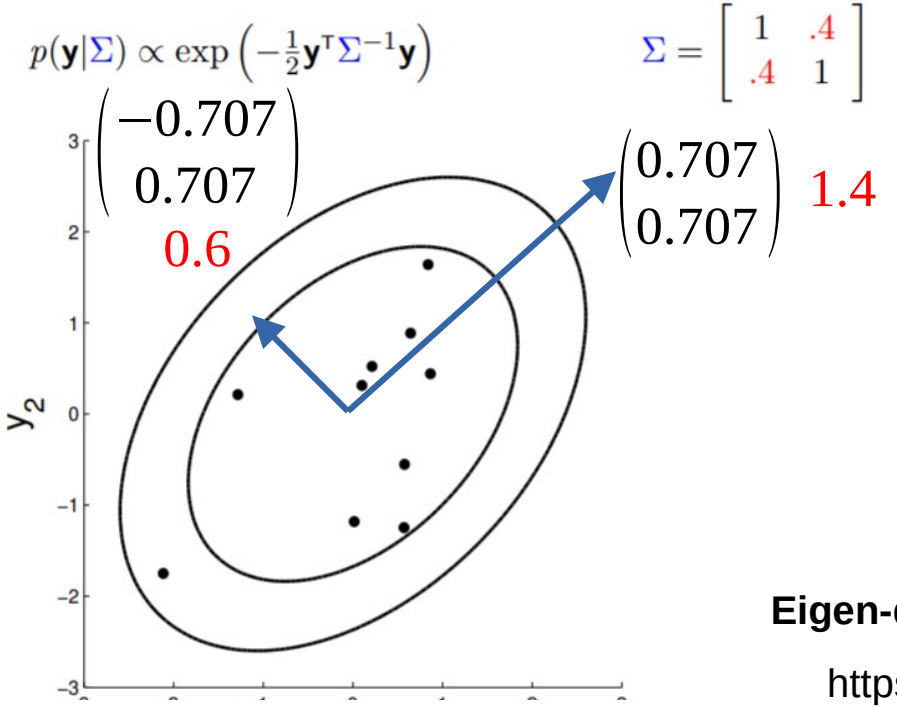
Lets just say that any data we observe are from an unknown Gaussian

Alternatively, **PCA = Principle Component Analysis**

I estimate: **cov(X) = X^T * X** and perform an **eigendecomposition** (diagonalization via eigenvectors)
> **Warning**! All points should be centered around their density/variance center

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^\intercal\Sigma^{-1}\mathbf{y}\right)$$

$$\Sigma = \begin{bmatrix} 1 & .4 \\ .4 & 1 \end{bmatrix}$$




```
> cv.mat <- matrix(c(1, 0.4, 0.4, 1), 2, 2)
> eigen(cv.mat)
eigen() decomposition
$values
[1] 1.4 0.6

$vectors
           [,1]       [,2]
[1,] 0.7071068 -0.7071068
[2,] 0.7071068  0.7071068
```

Eigen-decomposition tutorials:

https://www.youtube.com/watch?v=PFDu9oVAE-g

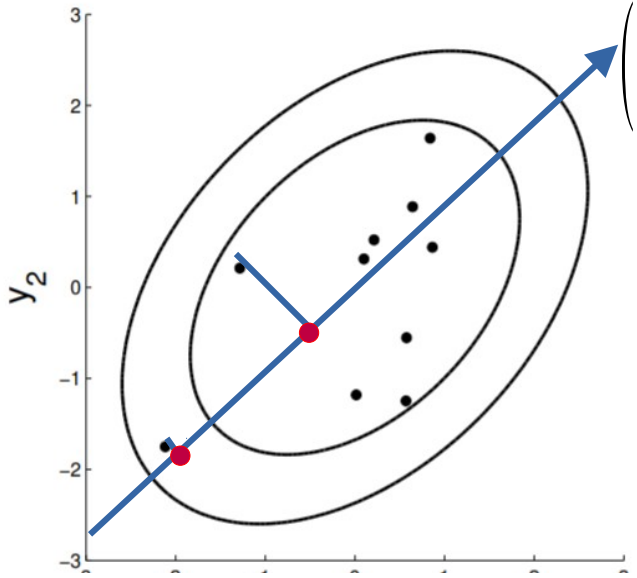
https://www.youtube.com/watch?v=KTKAp9Q3yWg

https://math.fel.cvut.cz/en/people/velebil/files/lag_2023_podzim/lag_handout08a.pdf

Alternatively, **PCA = Principle Component Analysis**

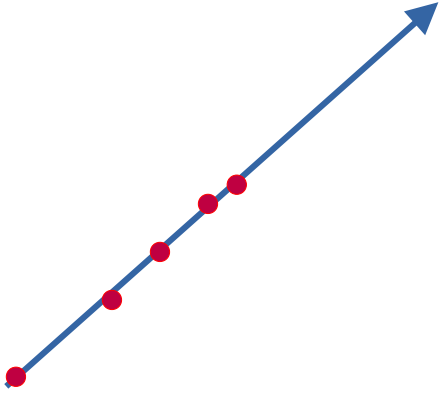
Use only K largest eigenvalues = largest variance axes, where K is a needed reduced dimension(typically K = 2)

$$p(\mathbf{y}|\Sigma) \propto \exp\left(-\tfrac{1}{2}\mathbf{y}^{\mathsf{T}}\Sigma^{-1}\mathbf{y}\right) \qquad \Sigma = \begin{bmatrix} 1 & .4 \\ .4 & 1 \end{bmatrix}$$



$$\begin{pmatrix} 0.707 \\ 0.707 \end{pmatrix} \quad 1.4$$

Take $K = 1$ and project onto it

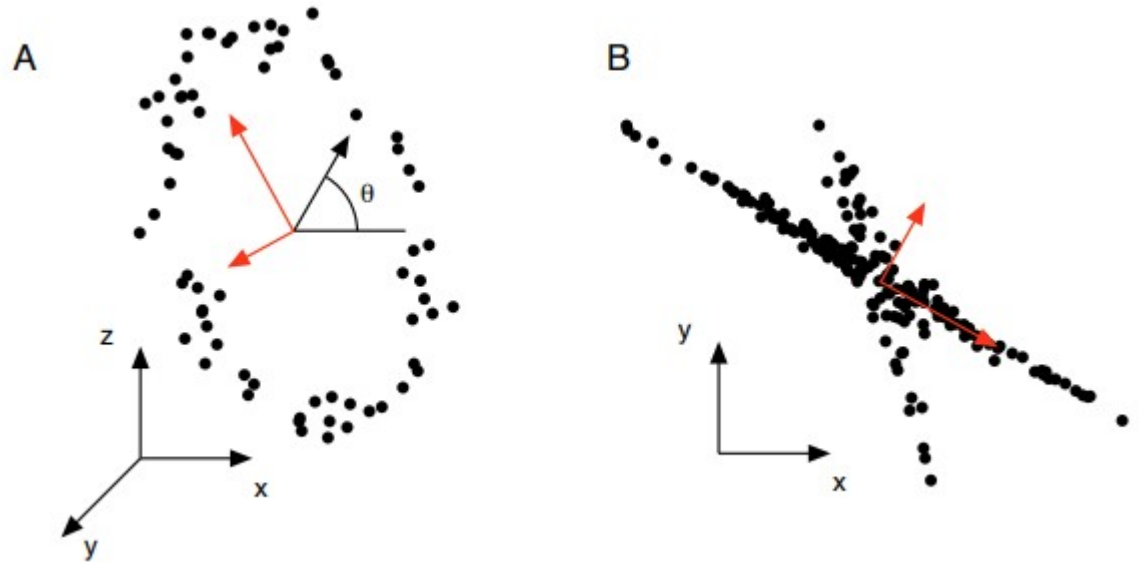


> On the Final exam you may get a question of PCA computation! This is a short description of its principle

Alternatively, **PCA = Principle Component Analysis**

Of course, the data does not have to be from Gaussian distribution to be reduced by PCA

> Mean and Covariance Matrix could be computed for any data with same formula as with Gaussian from LDA



PCA

LDA

max scatter
of the **entire data set**

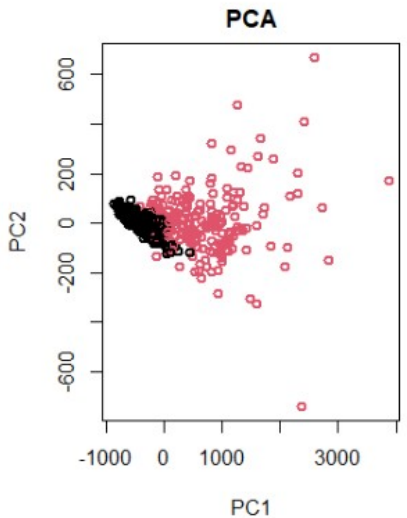
Finds axes/directions of:

max scatter **between**
AND
min scatter **within** classes

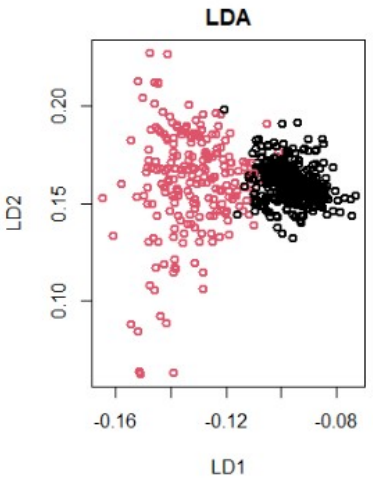
$Cov(X)$

Eigenproblem leading to the new axes

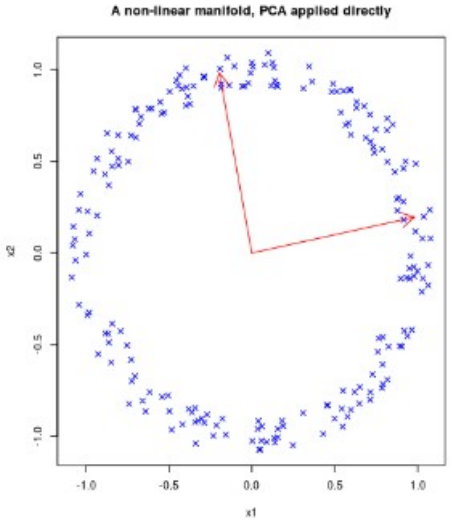
$S_w^{-1}S_B$

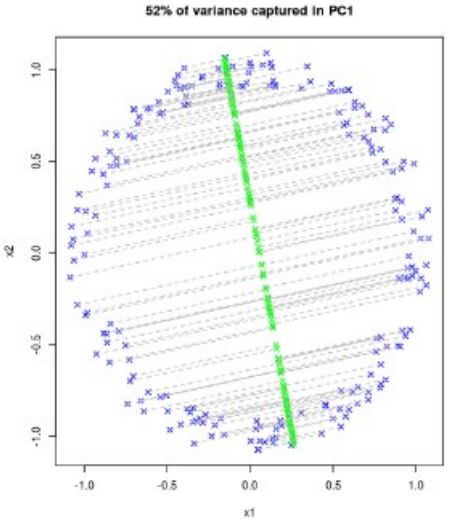


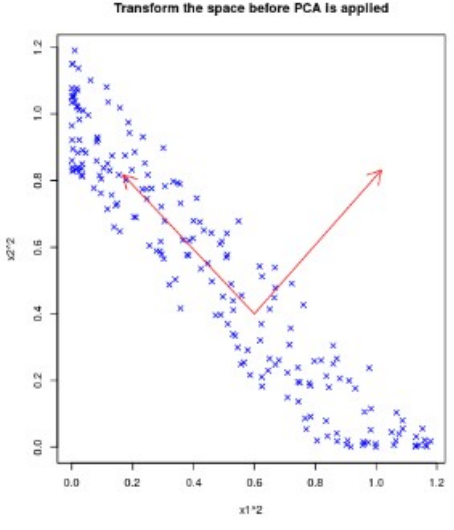
Projection (*BreastCancer*)

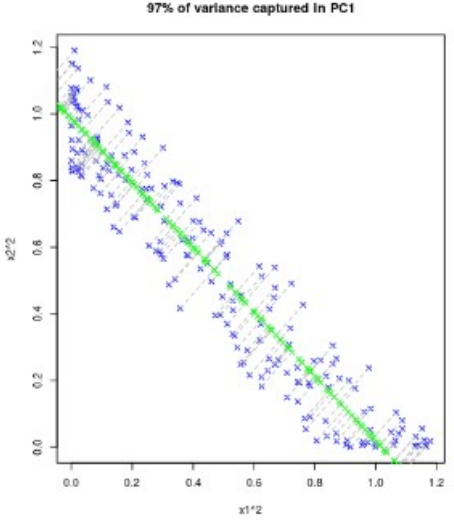


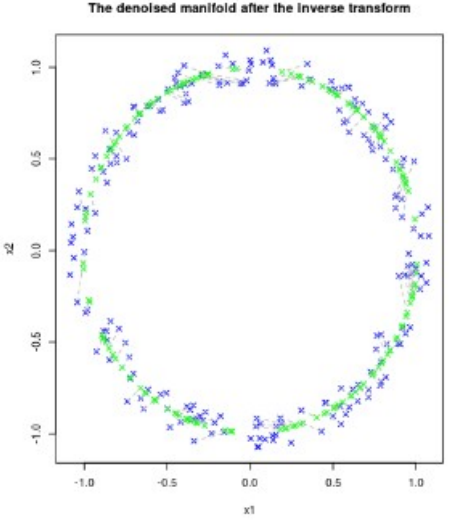
The need for non-linear methods











Non-linear Dimension Reduction: PCA/LDA not helping. What is a Manifold?

2D: Points + distances between points on map

1D Manifold:
Path and distance between stops

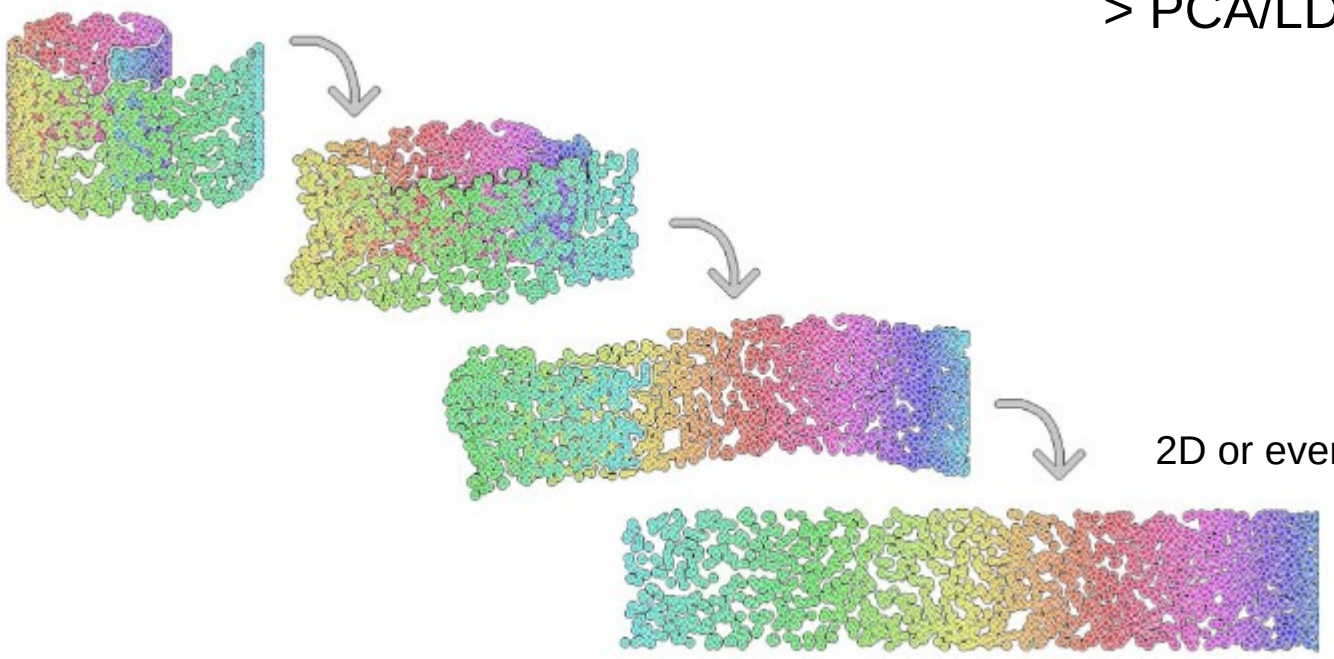
Non-linear intuition = straight line is not the shortest distance between points

> For example: Physical laws of matter prevents us from going through earth (you can’t simply pierce through land and water)

Surface is non-linear = shortest path is a parabola

Non-linear Dimension Reduction:
Manifold: same principle

3D



> PCA/LDA cannot find this!

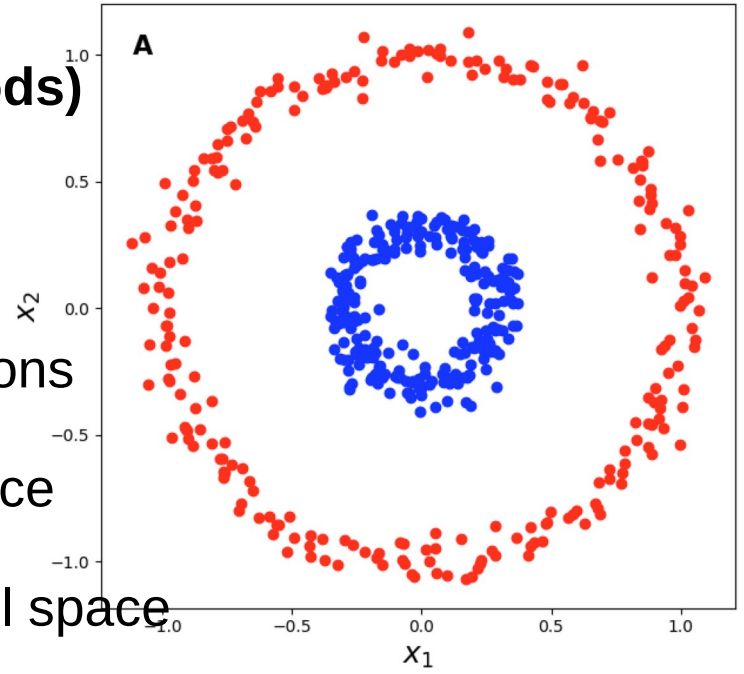
2D or even 1D manifold

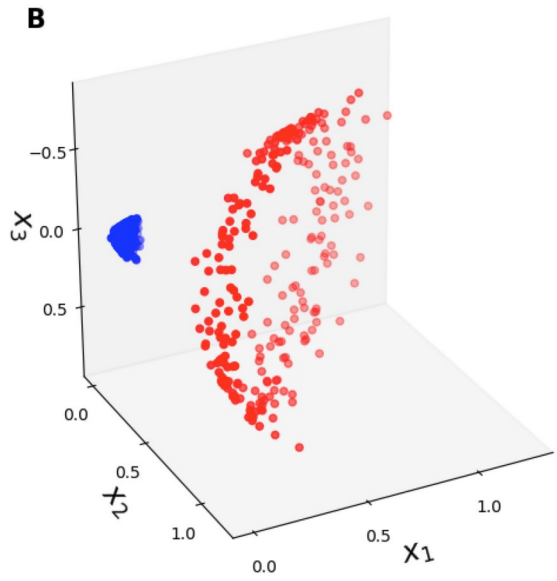
Kernel Trick (PCA/other methods)

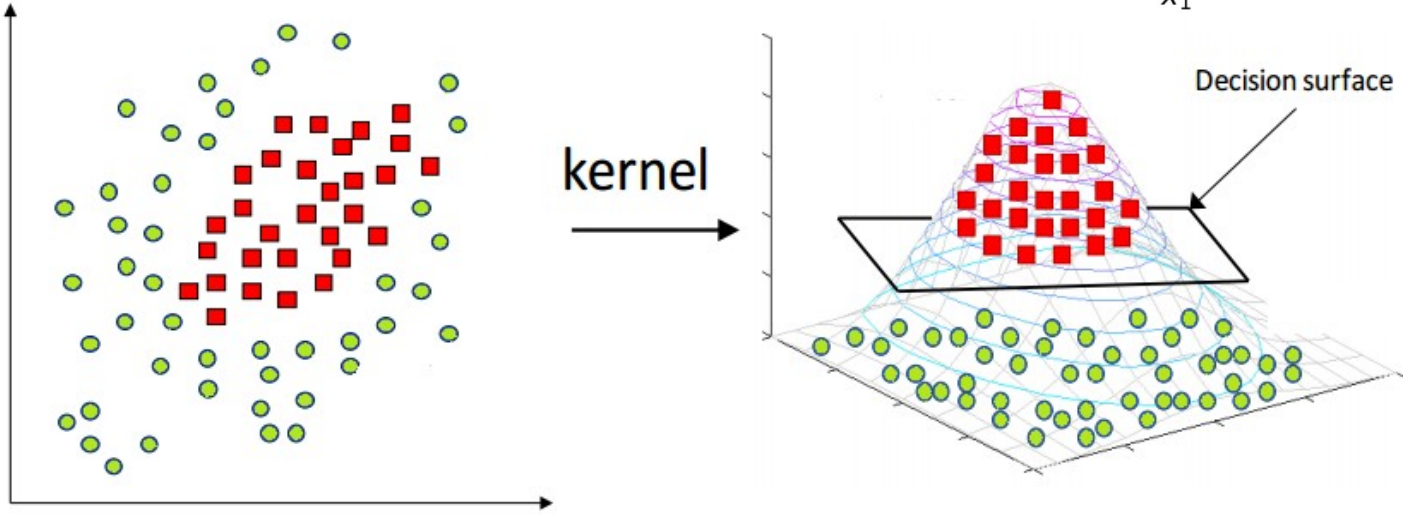
> Non-linear in D dimensions

> But is linear in (D+1)* dimensions
* can be lower/higher,
generally = in some other space

> do PCA/other method in kernel space





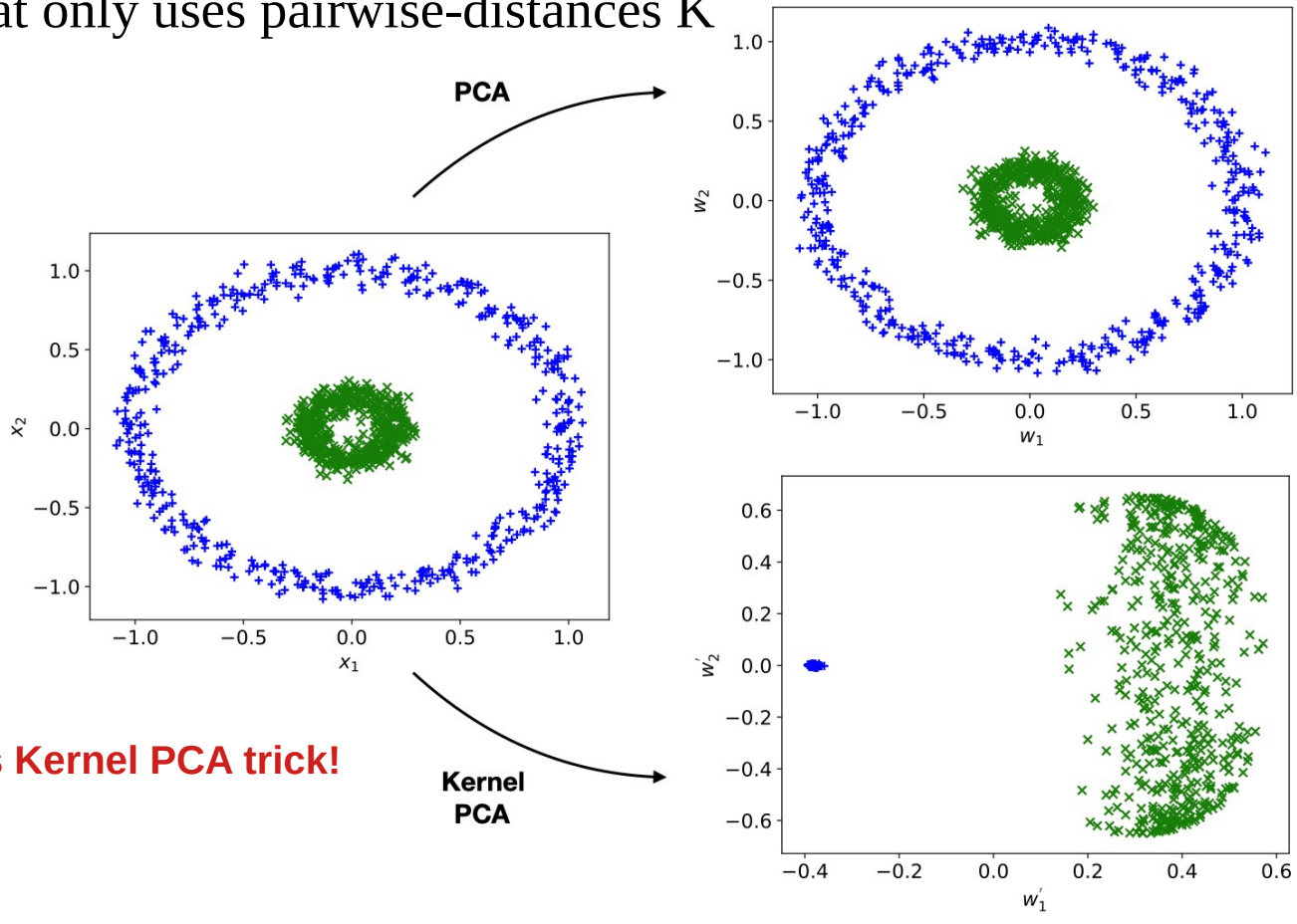


$\phi(X) =$ cannot compute, some complicated formula

$\phi(X)^T \phi(Y) =$ metric $=$ distance in complicated space

$K(X, Y) = \phi(X)^T \phi(Y) =$ it is enough to compute distance for each pair

$cov = X^T X \rightarrow cov\ \phi =$ formula that only uses pairwise-distances K



> On Final exam you may get a question:
Please show the exact computation of this Kernel PCA trick!

Multidimensional scaling (MDS)

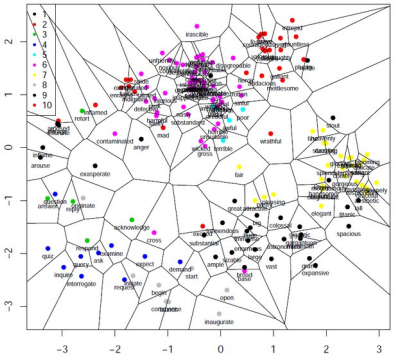
- The main idea
  - points close together in $\mathcal{X}$ should be mapped close together in $\mathcal{T}$,
- minimizes the **stress function**

$$stress(\mathbf{T}, f) = \sqrt{\frac{\sum_{i,j=1}^{m} (f(\delta_{ij}) - d_{ij})^2}{\sum_{i,j=1}^{m} d_{ij}^2}}$$

  - $\delta_{ij} = d_{\mathcal{X}}(\mathbf{x_i}, \mathbf{x_j})$, $d_{ij} = d_{\mathcal{T}}(\mathbf{t_i}, \mathbf{t_j})$ – typically Euclidean,
  - f is a proximity transformation function (e.g., identity, monotonic $\rightarrow$ metric, ordinal),
- whole class of methods that differ in
  - the method for calculation of proximities δ,
  - the parametrization of stress function,
  - the method that minimizes the stress function (e.g., gradient descent, Newton).

https://www.youtube.com/watch?v=NEaUSP4YerM

Assignment 4: Isomap implementation



Introduction

The goal of this tutorial is to get familiar with some basic methods for dimensionality reduction, complete you own implementation of the **Isomap algorithm** (in cooperation with *Multidimensional scaling*), experiment with its parameters and compare with other techniques of dimensionality reduction (**PCA**, **t-SNE**).

> You get high-dimensional vectors of words and 10 categories of these words (for example: mood/emotion words, asking/question-related prepositions etc.)

> You will have to show that these words are clustered together by this logic

> You can check if your solution is correct by comparing it to ground-truth result: (**graph_iso.pdf**)

Some R technical issue in assignment 4:

> If you have a wrong version of deldir library, you may get an error:

> Error in eval(ccc) : object 'voronoi' not found

> You can solve it by changing (<<- is double <, global variable):

voronoi <- deldir(X[,1], X[,2])

to

voronoi <<- deldir(X[,1], X[,2])

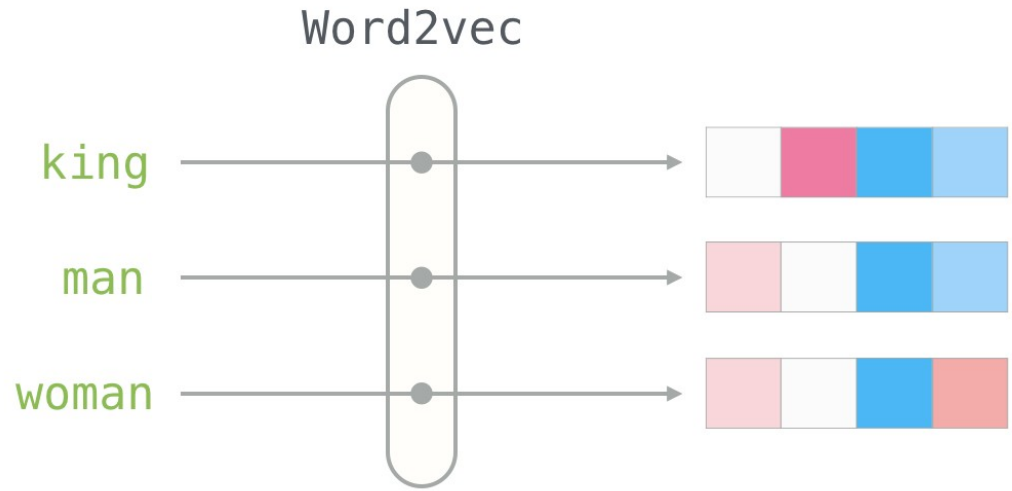
Word2vec data representation

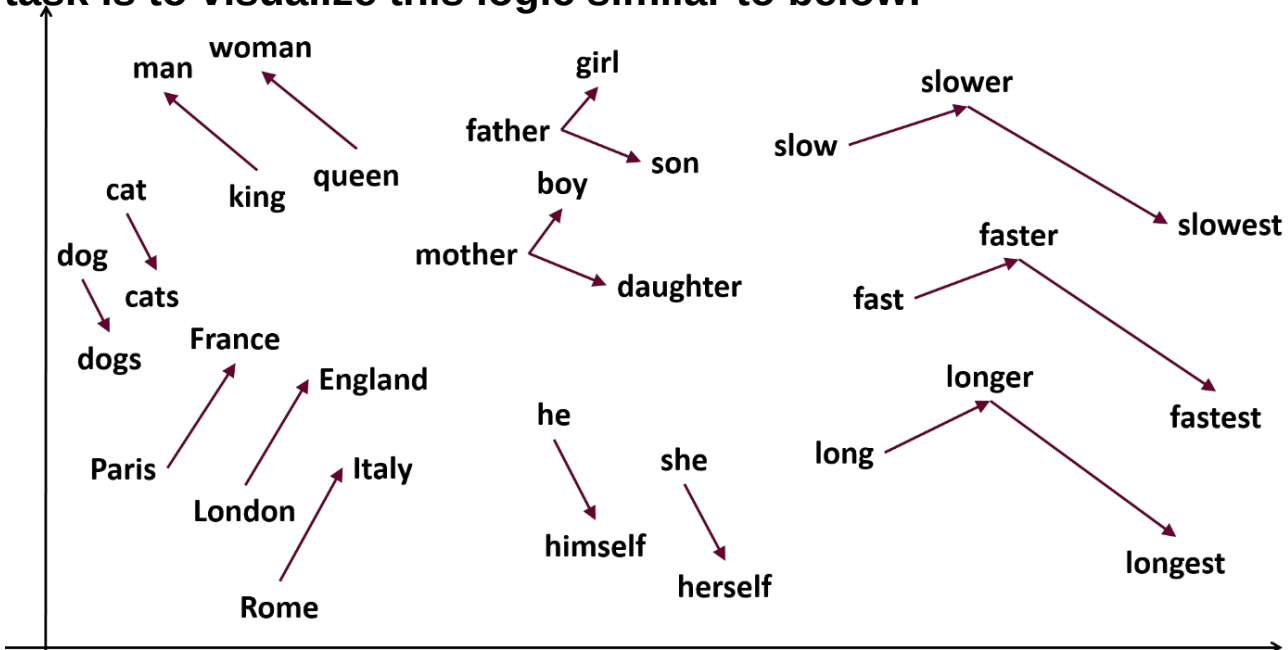
> ChatGPT for example uses this

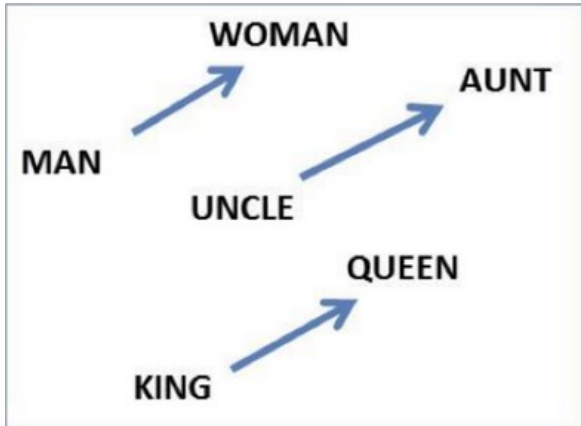
> Each word is a vector/point
in 256/1024 dimensional space

> The idea is that logic in language
is observed in geometrical logic

> The task is to visualize this logic similar to below:



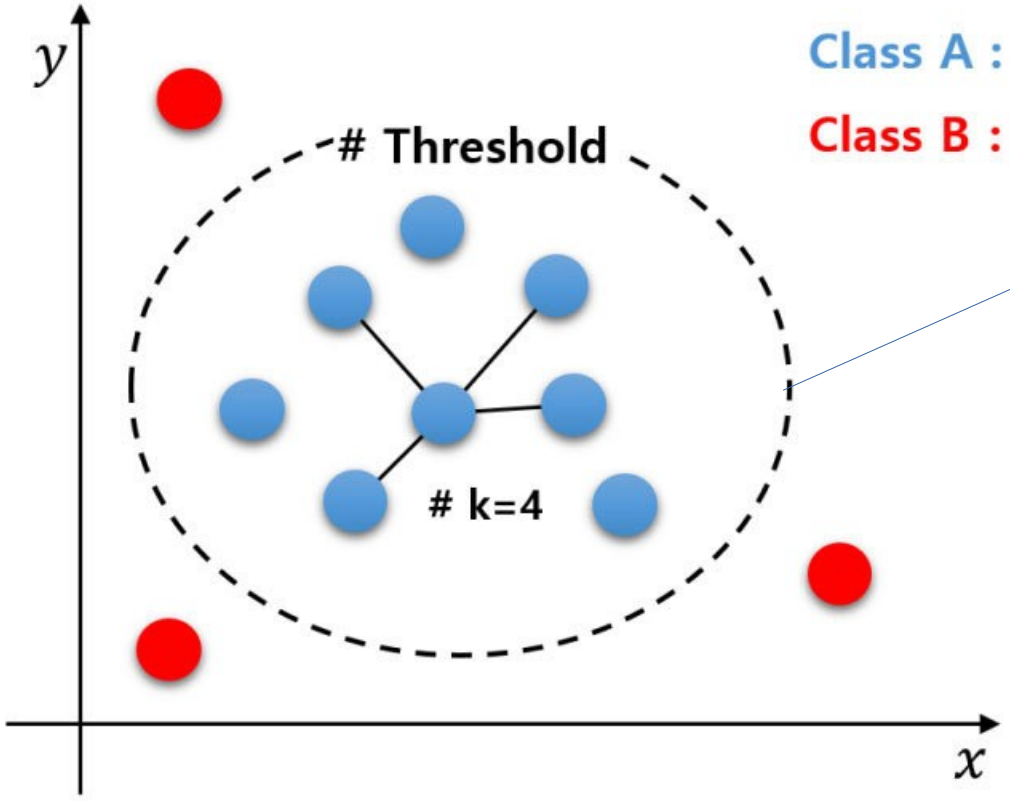




K Nearest Neighbors in Isomap

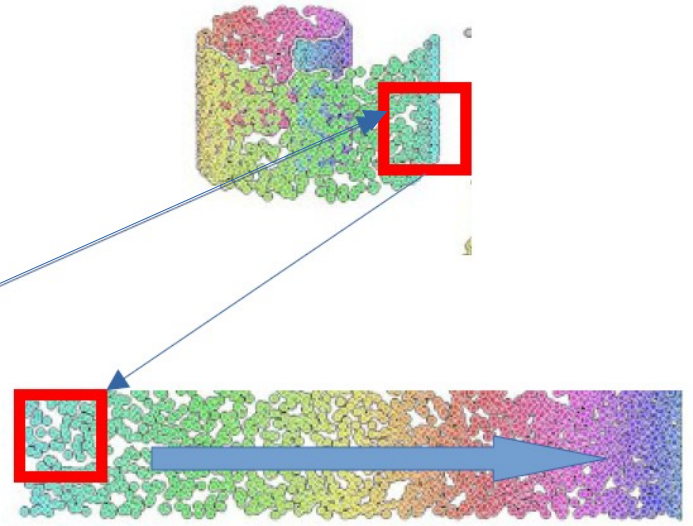
Use KNN to generate a local plane of neighbors

It uses the manifold's local linearity/local euclidean geometry



Class A : Inliers

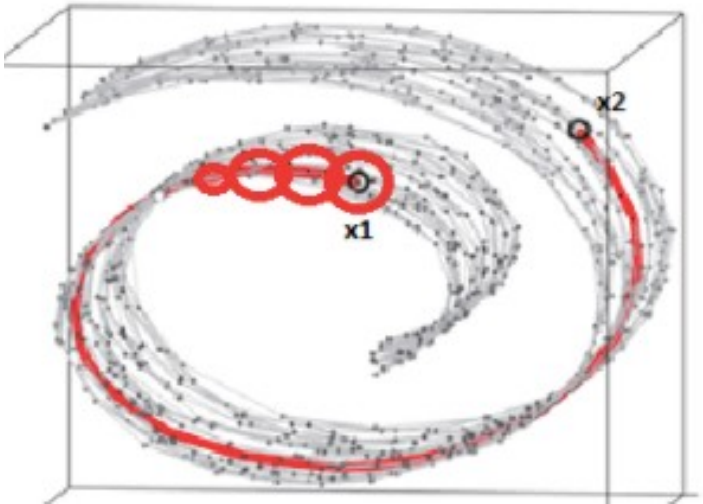
Class B : Outliers

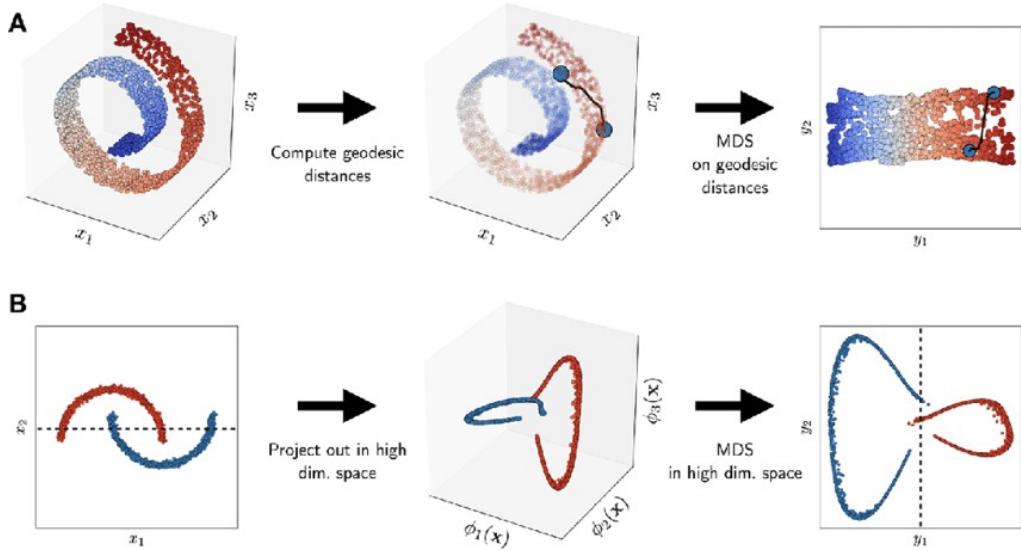


Find shortest path around such local planes,
That way we can "unroll" it
and generate such 2d representation

2. **Implement ISO-MAP dimensionality reduction procedure**.

- Use *k*-NN method to construct the neighborhood graph (sparse matrix).
  - For simplicity, you can use `get.knn` method available in `FNN` package.
- Compute shortest-paths (geodesic) matrix using your favourite algorithm.
  - Tip: Floyd-Warshall algorithm can be implemented easily here.
- Project the geodesic distance matrix into 2D space with (Classical) Multidimensional Scaling (`cmdscale` functions in R).
- Challenge: you may simply use PCA to do the same, but be careful to account for a proper normalization (centering) of the geodesic (kernel) matrix (see Kernel PCA for details).





Bonus homework: dimred_artifical + dimred_breast_cancer

9	18.11.	JB, AA, JK	Dimension reduction	san_dimred.zip, assignment4.zip

> Fill some gaps in code

> I will give you a 0.5 points for every file (0.5 for artificial, 0.5 for breast cancer)

```
# Now plot the first two columns
plot(___,
     col=rainbow(nrow(plane)),
     ylab = "PC2")
```

```
# Use the eigen() and
transform_mat <- ___

my_x <- ___       # Tra
```