Dimensionality Reduction

CASA0006: Data Science for Spatial Systems

Huanfa Chen

CASA0006

Outline
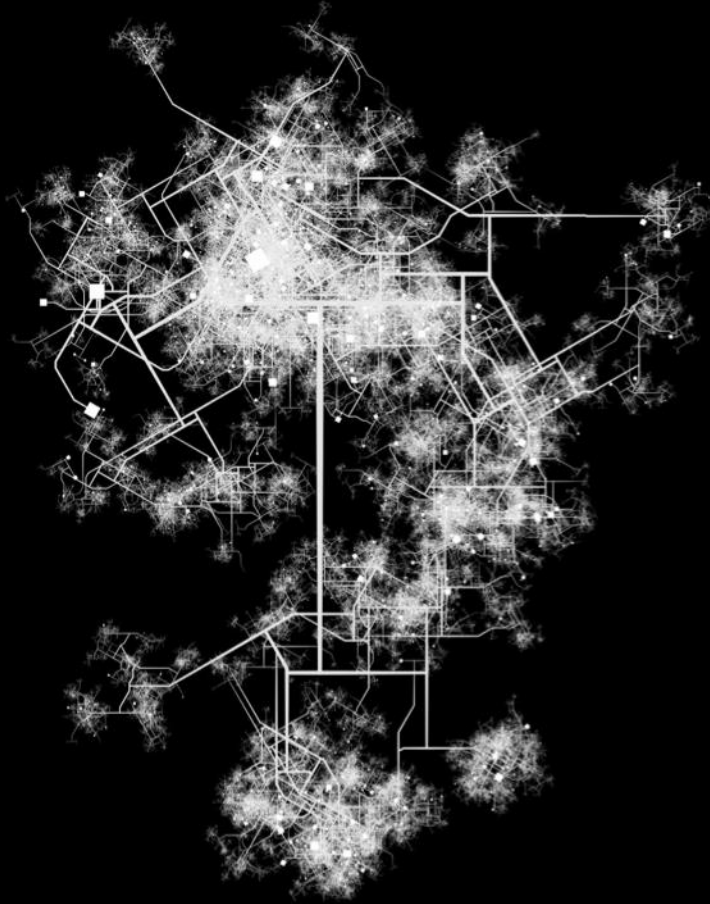
1. Dimensionality Reduction

2. Curse of dimensionality

3. Methods

   a. PCA
   b. Kernel PCA
   c. LLE

4. Summary

Dimensionality reduction

Dimensionality Reduction
New representations in lower-dim space

- The process of reducing the number of variables under considerations by obtaining a set of relevant factors

- It is *unsupervised learning*, so there is no ground truth to validate the result

Input (n columns) → DR → *Output (k columns)*

a	b	c	d

e	f
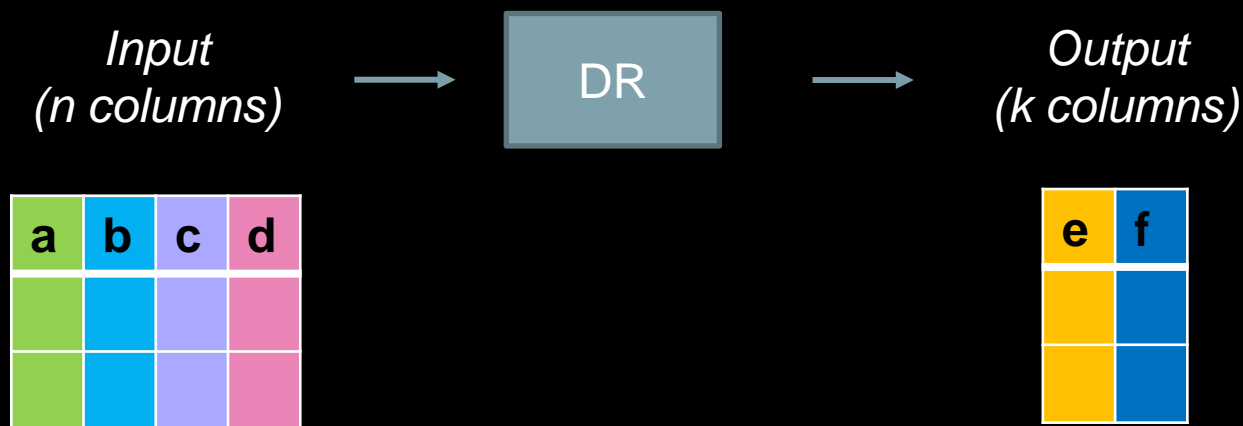
Dimensionality Reduction

Two classes of DR

1. Linear DR: each new dimension is a linear function of the original dimensions. Example: PCA

2. Non-linear DR: kernel PCA, LLE

Dimensionality Reduction

Key questions for a DR algorithm:

1. Is it linear or non-linear?
2. What is the objective of this algorithm?
3. What is the application of the outputs?
4. What are the hyperparameters? How to tune them?

Dimensionality Reduction
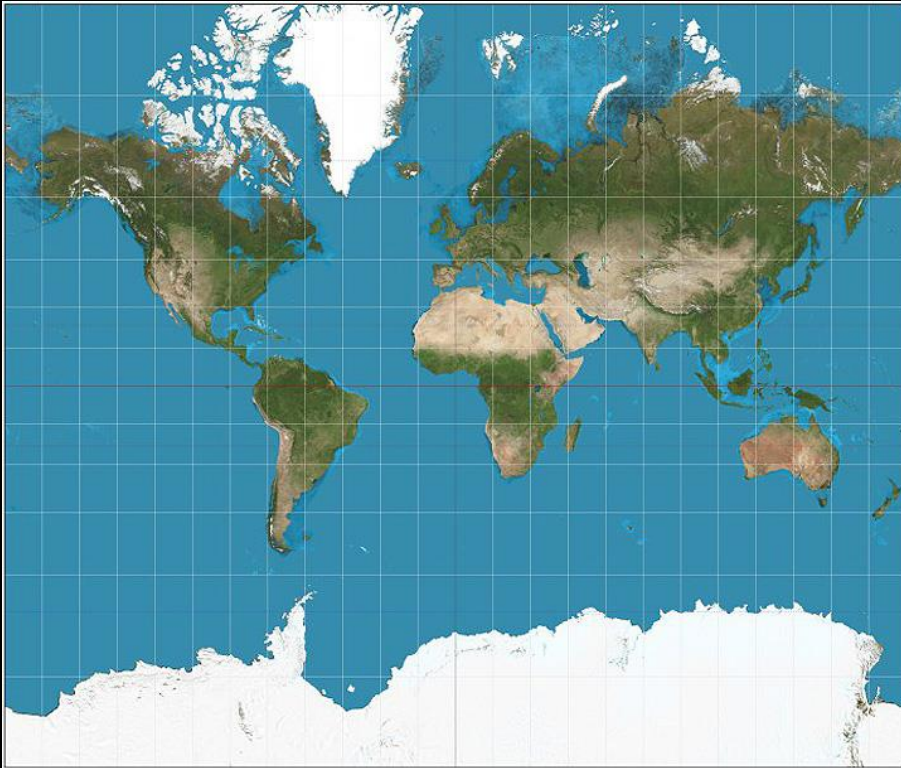
DR is similar with Map projection (e.g. Mercator)

1. Both are reduction of dimensions
2. There are various methods, depending on applications
3. Both would lead to some information loss

Image Credit

Dimensionality Reduction
Motivations

Perspective	Details
Visualisation	To visualise the data when reduced to low dimensions such as 2D or 3D
Computation	To reduce the time and storage space
Modelling	To reduce number of features and avoid overfitting
Others	To avoid the curse of dimensionality

Curse of dimensionality

Curse of Dimensionality
Difficulty of high dimensions

1. Possibilities are exponential in the dimensions
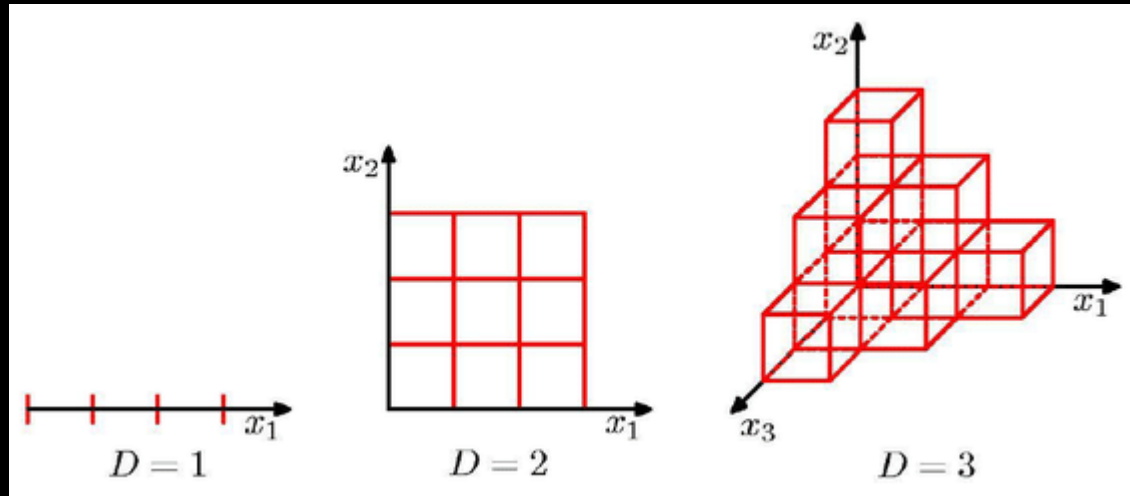
Possible values 3^1 3^2 3^3 3^k

Each additional dimension triples the effort to grid search all combinations.

Curse of Dimensionality
Difficulty of high dimensions

2. High dimensions cause overfitting in machine learning

- An enormous amount of training data is required to ensure that there are several samples with each combination of values.

- When we have more features than records, we run the risk of massively overfitting our model
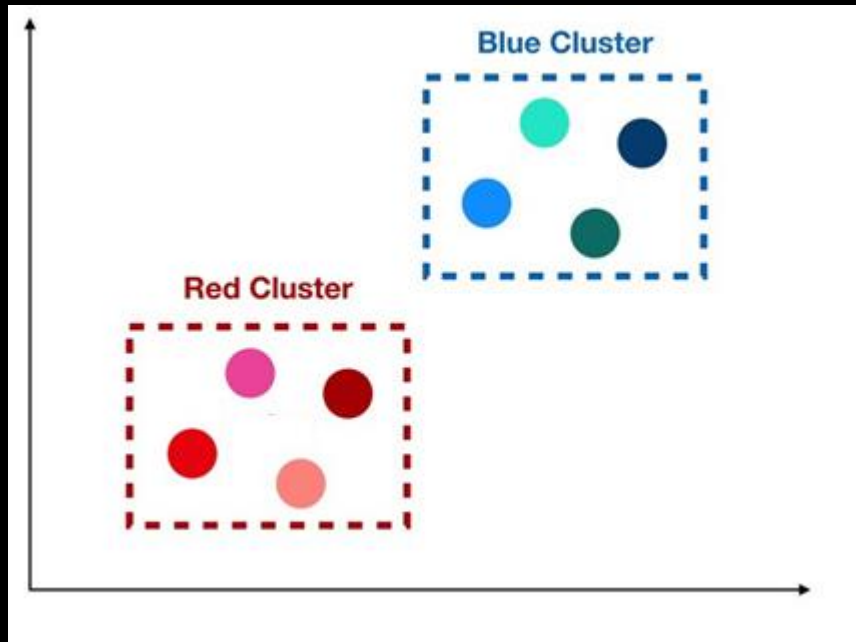
Curse of Dimensionality
Distance functions

3. Distance functions become meaningless in high dimensions

- Every observation in the dataset appears 'equidistant' from all the others

- No meaningful clusters can be found.

https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e

Curse of Dimensionality
Distance functions

Example: Clustering of candies from colours

Visual observations:
there are two clusters of candies

14

Curse of Dimensionality
Distance functions

Colour definition using 8 colours

- What is the Euclidean distance between each pair?
- How many clusters?

	Red	Maroon	Pink	Flamingo	Blue	Turquoise	Seaweed	Ocean
●	1	0	0	0	0	0	0	0
●	0	1	0	0	0	0	0	0
●	0	0	1	0	0	0	0	0
●	0	0	0	1	0	0	0	0
●	0	0	0	0	1	0	0	0
●	0	0	0	0	0	1	0	0
●	0	0	0	0	0	0	1	0
●	0	0	0	0	0	0	0	1

Example credit (Adjusted)

Curse of Dimensionality

Using DR to learn the new dimensions

	Red	Blue
Red	1.00	0
Maroon	1.20	-0.10
Pink	1.00	0.20
Flamingo	0.80	0
Blue	0	1.00
Turquoise	0.25	0.90
Seaweed	0.15	1.00
Ocean	-0.10	1.20

Curse of Dimensionality

Transform the candies using the new dimensions
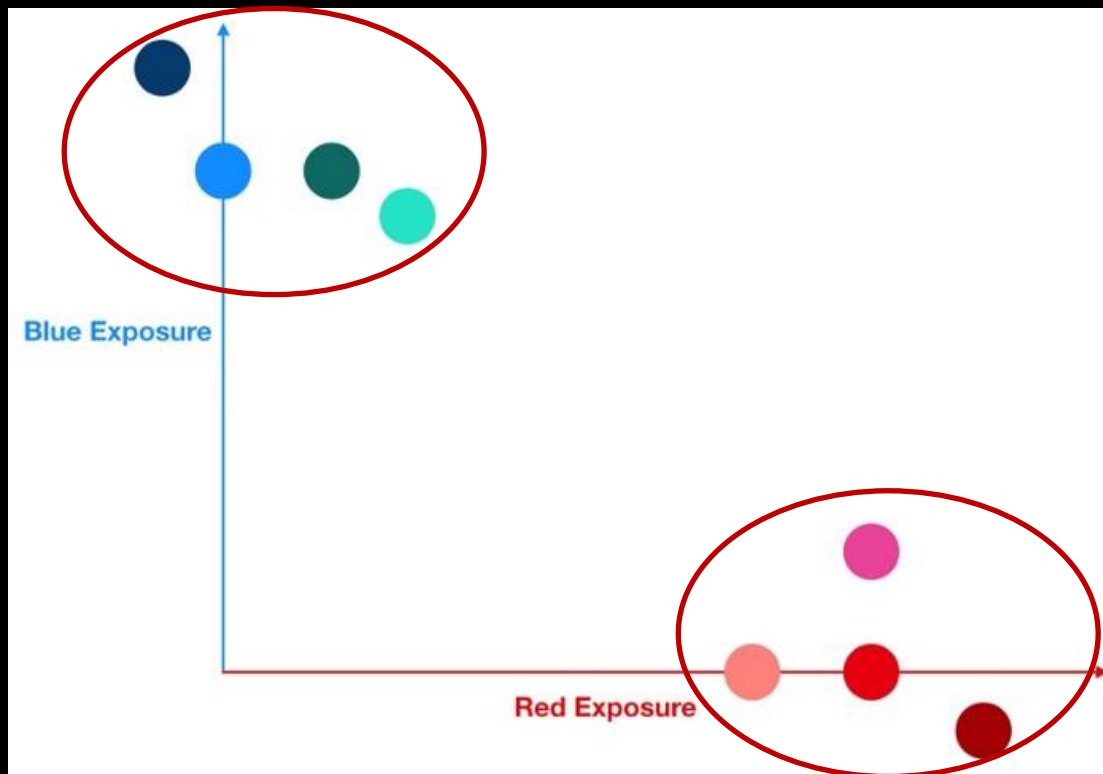
Two clusters are identified, which is consistent with the visual judgement

Curse of Dimensionality
Implications

In a high dimension space, it is (very) likely

1. Many features are almost constant, and some are highly correlated.

2. Most observations lie within (or close to) a much lower-dimensional subspace

3. DR algorithms aim to learn the low-dimensional subspace

Principal Component Analysis

Principal Component Analysis

Linear combination of features

- Steps ("*Keep the largest variance*")
    1. Find a new set of dimensions (principal components, or PC). Each PC is a linear combination of the original dims
    2. Rank all PC according to the variance of data. The larger variance, the higher importance.
    3. Keep the first k PC (using rules to select)
    4. Project the data into the new space.

a	b	c	d

→
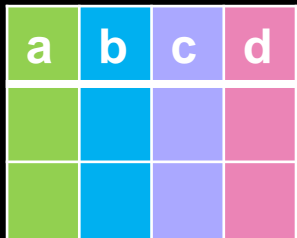
Name	New dim	combination	Variance
1st component	e	0.5a + 0.6b + 0.1c + 0.3d	0.8
2nd component	f	0.6a + 0.1b + 0.2c + 0.5d	0.1
	g		0.05
	h		0.05

Variance of data

- Variance of a list $X = [X_1, X_2, \ldots, X_m]$ in one dimension

$$Var(X) = \sum_{i=1}^{m} \frac{(X_i - \bar{X})^2}{m}$$
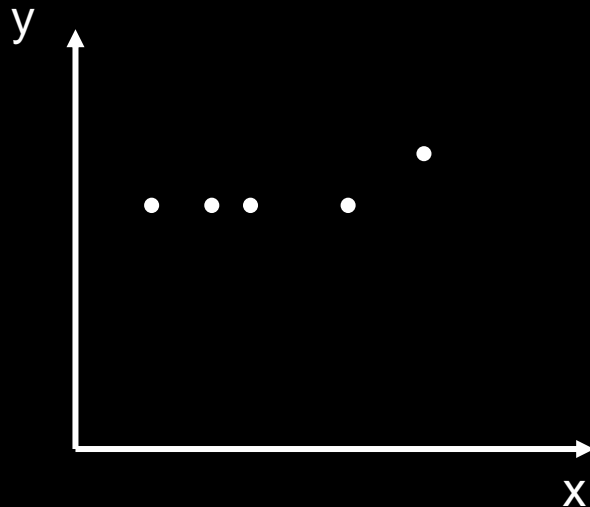
On which of the two axes (x or y), the data have a larger variance?

Variance of data

- If we want to find a line (or a projection) that keeps the most variance of points, which line should be selected?

Which projection keeps more variance of points?
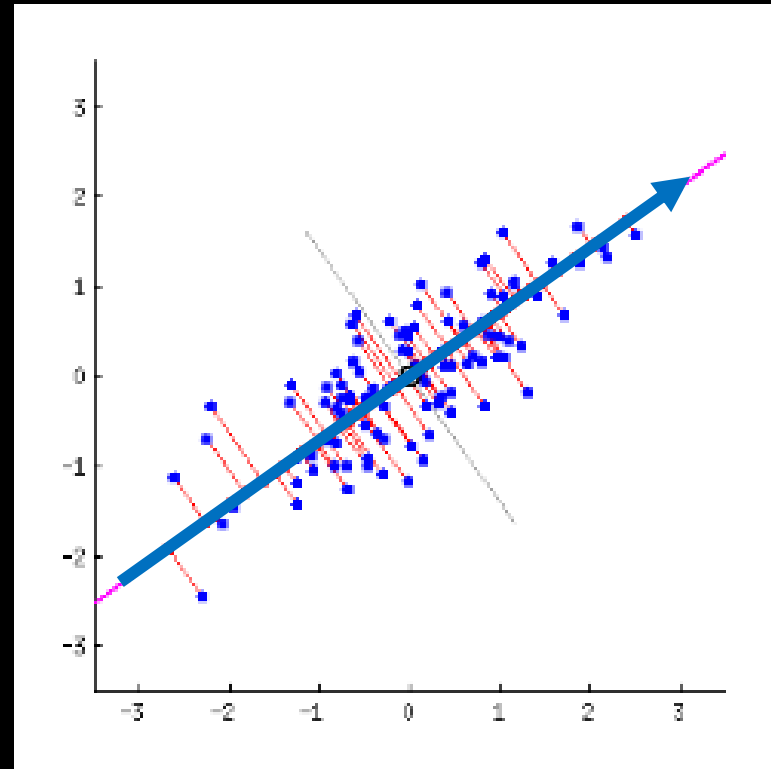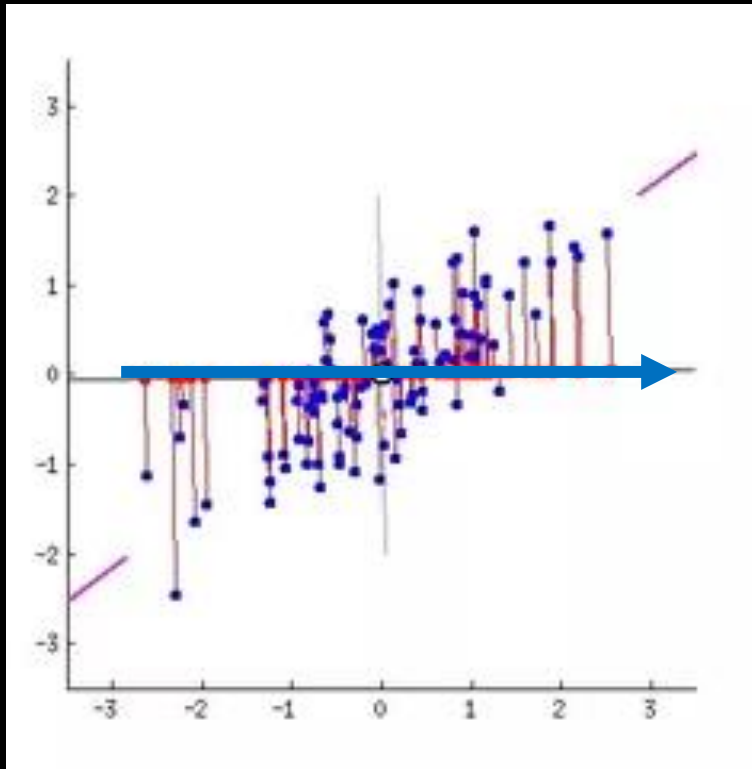
Illustration of PCA
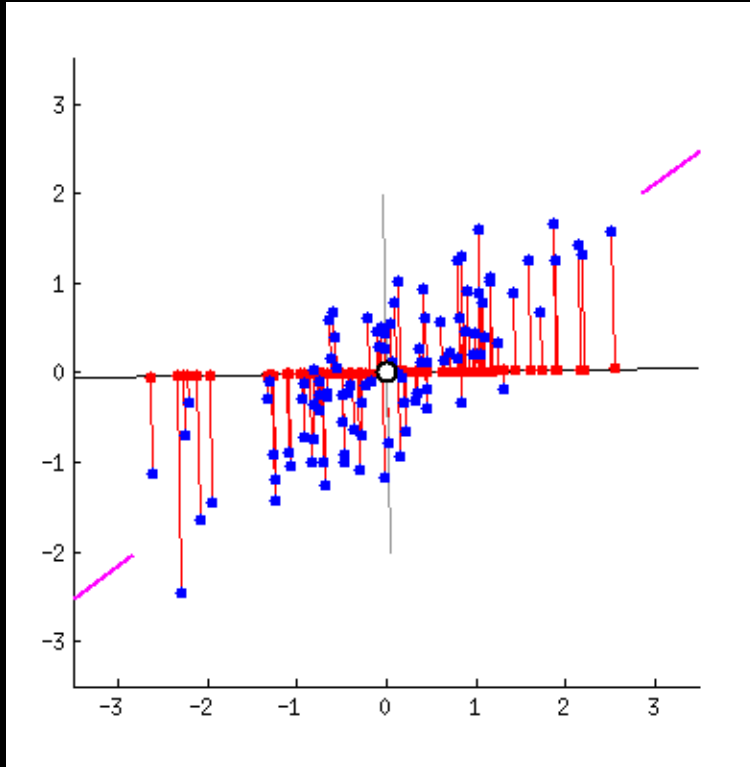
- PCA aims to search for new dimensions that are uncorrelated and most of the information (or variance) within the initial dataset is stored into the first k components

- It can be described by math, using covariance matrix, eigenvectors, and eigenvalues (see link below)

- Simply speaking, eigenvalues are proportional to the explained variance of data on this PC

Good introduction to math behind PCA:
https://towardsdatascience.com/pca-eigenvectors-and-eigenvalues-1f968bc6777a

How many factors of PCA to retain?

Three rules to choose k and you can use one of them.

1. For visualisation, k=2 or 3 (Why? It is hard to visualise more than 3 dimensions on a paper or webpage)

2. To retain components with eigenvalues greater than one (would fail if all/most eigenvalues are smaller than one)

If Rule 2 is followed in this example, then we choose k = 1

Table 8.2 Variance explained by each component

	Total variance explained					
	Extraction sums of squared loadings			Rotation sums of squared loadings		
Component	Total	% of variance	Cumulative %	Total	% of variance	Cumulative %
1	2.602	52.032	52.032	1.035	20.707	20.707
2	.957	19.149	71.181	1.032	20.637	41.344
3	.741	14.826	86.007	1.018	20.358	61.702
4	.362	7.244	93.251	1.005	20.110	81.812
5	.337	6.749	100.000	.909	18.188	100.000

24

Rogerson, P.A. (2020) Statistical Methods for Geography: A Student's Guide, Fifth. Sage, London, pp 242-243

How many factors of PCA to retain?

3. To plot the eigenvalues on the y axis and the factor number on the x axis of a graph (termed a *scree plot*), and then locate a point just before the graph flattens out (like the elbow method)

Rule 3: choose k = 2;

Rogerson, P.A. (2020) Statistical Methods for Geography: A Student's Guide, Fifth. Sage, London, pp 242-243
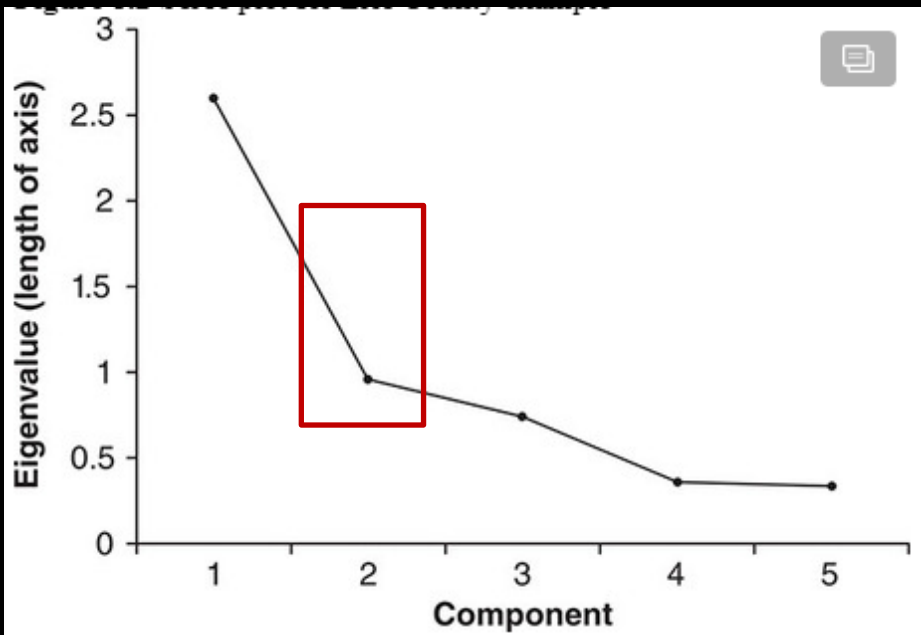
Summary of PCA

- PCA requires data standardisation, as it is sensitive to the relative scales of different variables.

- Good interpretation: each component is a linear combination of features

- It is guaranteed that the new features are uncorrelated – no more multicollinearity concerns.

- Common use: visualising the data; checking the clustering results

- The PCA outputs can be used as an input to clustering/classification/regression.

Principal Component Analysis
Problems

- PCA does not work well for 'twisted' dataset. In this case, non-linear DR or manifold learning is useful.
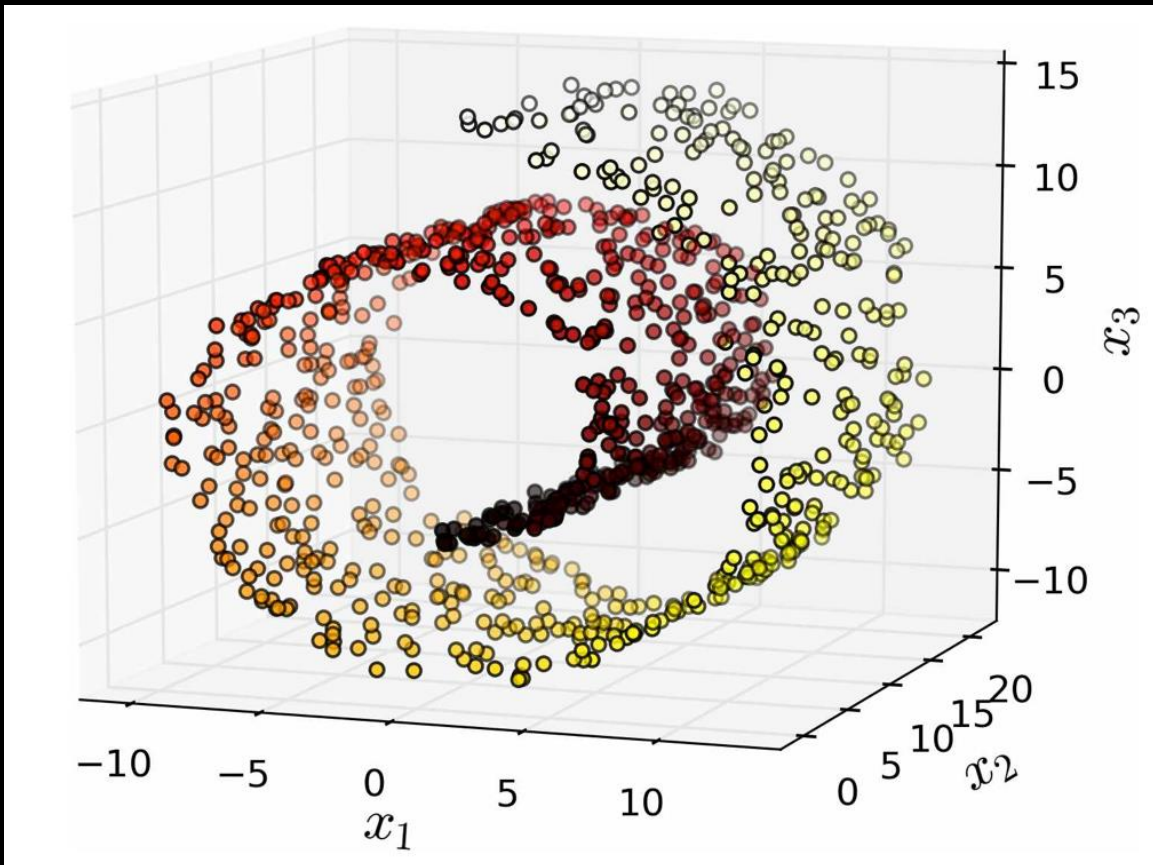
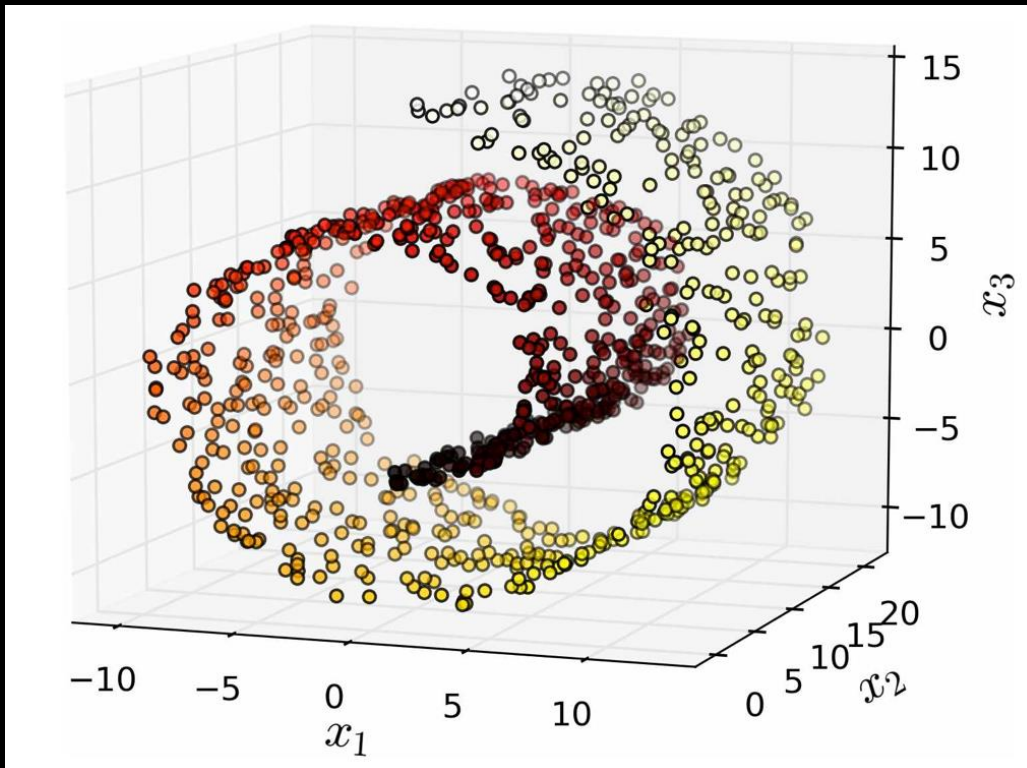Example: a 3-D twisted data (a swiss roll)

This data is synthesised to illustrate the DR methods.

Colour is used to represent clusters.

<u>Image Credit</u>
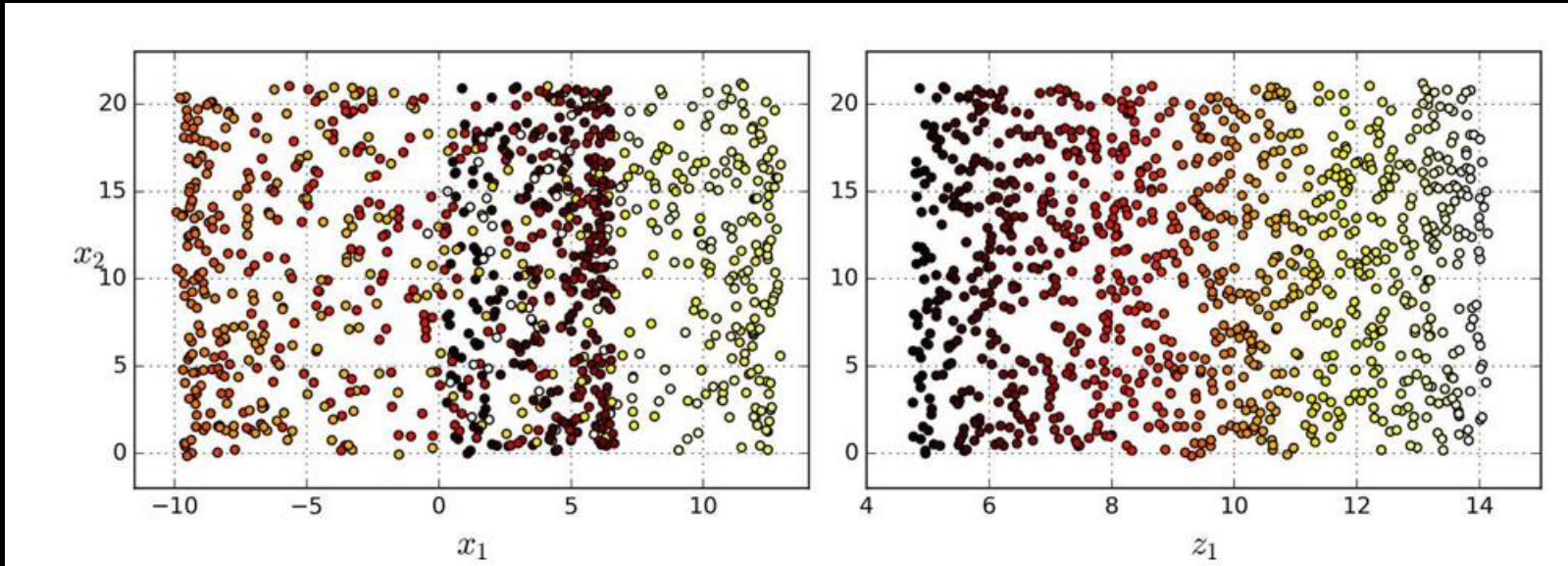
Manifold learning

- It relies on the <u>manifold assumption</u>: most real-world high-dimensional datasets lie close to a much lower-dimensional manifold.

- This assumption is very often empirically observed.

<u>Image Credit</u>

28

PCA vs. Manifold learning

PCA: fail to identify point clusters in the original space

Manifold learning

Kernel PCA

kPCA

- kPCA is one type of manifold learning or non-linear DR
- <u>Kernel trick</u>: a mathematical technique that implicitly maps instances into a very high-dimensional space, enabling non-linear classification/regression

- The idea of kPCA is: it firstly uses a kernel function to map observations into a higher dimension, then do DR in the higher dimension space.
- The benefits are that it would find patterns that are not identified using linear DR.

kPCA

Common kernels

	Formula	Parameters	Merits
Linear	$K(x, x_i) = x \cdot x_i$	/	It is only used when the sample is separable in low dimensional space.
Polynomial	$K(x, x_i) = [\gamma * (x \cdot x_i) + coef]^d$	$\gamma, coef, d$	global kernels
RBF	$K(x, x_i) = \exp(-\gamma * \|x - x_i\|^2)$	$\gamma.$	good local performance
Sigmoid	$K(x, x_i) = \tanh(\gamma(x \cdot x_i) + coef)$	$\gamma, coef$	needs to meet certain conditions

TABLE I.   DIFFERENT KERNEL FUNCTIONS OF SVM

Image Credit

kPCA

- Hyperparameters of kPCA
  - Which kernel to use: linear kernel, rbf, sigmoid kernel, etc
  - Hyperparameters of kernels: gamma of rbf

```
k_pca = KernelPCA(n_components = 2, kernel="rbf", gamma=0.0433, fit_inverse_transform=True)
```

- Hyperparameter tuning: similar with many ML algorithms (K-means or random forest)
- You can use a performance measure and ground-truth data (e.g. classification, or MSE of fit) to tune the hyperparameters.

kPCA

- Example: using kPCA to reduce the twisted data from 3D to 2D.
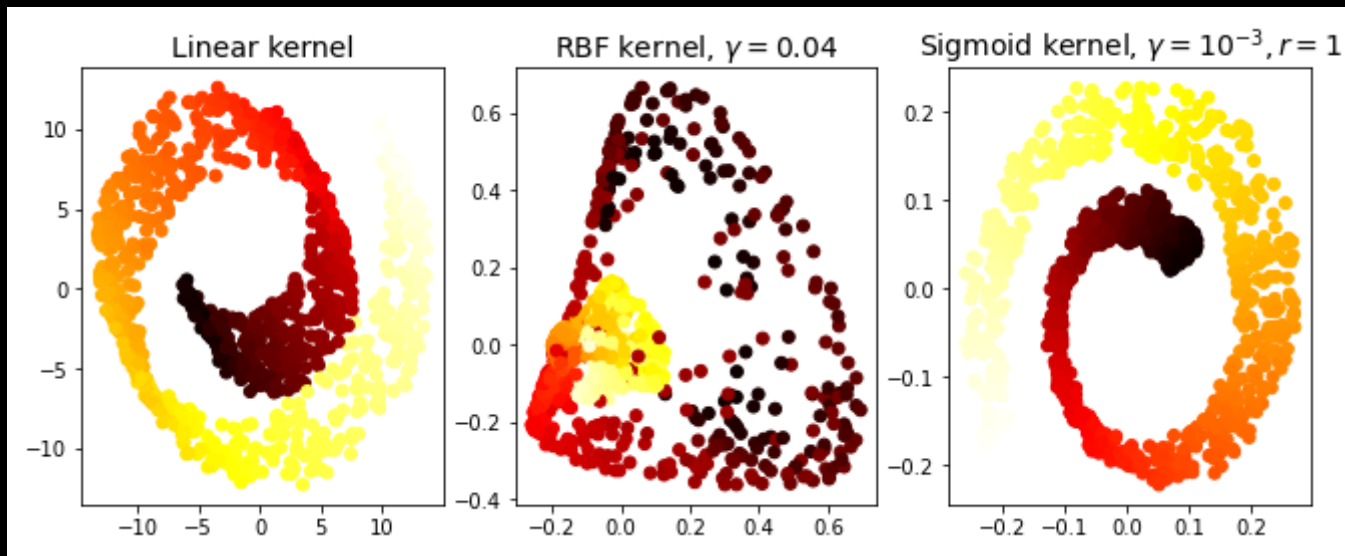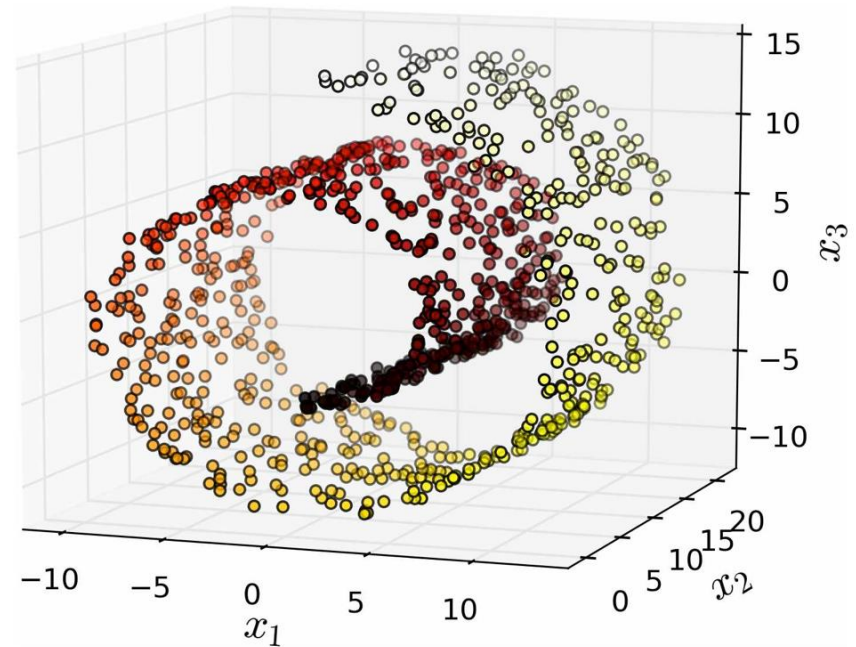- The clusters are roughly kept in the kPCA results.

Image Credit

Locally Linear Embedding
(LLE)

LLE

- One type of <u>manifold learning</u> or non-linear DR
- The principle is to preserve local relations (contrast to preserving global variance in PCA)

- Two steps (hyperparameter: k or n_neighbors)
  1. Measure how each instance relates to closest neighbors (weighted sum)
  2. Look for low-dimension representation where local relations are best preserved.

LLE

- ## Step 1
  - For each training instance x_i, LLE identifies its k closest neighbors
  - Then, LLE tries to reconstruct x_i as a linear function of these neighbors $x_{NN_{ij}}$

$$x_i = \sum_{j=1}^{k} w_{ij} x_{NN_{ij}}$$

where NN_{ij} is the index of the j-th neighbour of x_i

Weights w_{ij} are selected such that

  - The squared distance between x_i and $\sum_{j=1}^{k} w_{ij} NN_{ij}$ is minimised.

 - Weights are normalised $\sum_{j=1}^{k} w_{ij} = 1$ for any x_i

The output is the weights w_{ij}, which represents the similarity between instance i and its j-th neighbour

LLE

- Step 2
  - Map the observations into a lower d-dimensional space such that these local relationships are preserved as much as possible.

  - If z_i is the d-space equivalent of x_i, then we want $z_i - \sum_{j=1}^{k} w_{ij} z_{NN_{ij}}$ to be minimized, given the w_{ij} from Step 1

  - The output of Step 2 is the coordinates of observations in the new d-dim space

LLE

- Example (using LLE to unroll the Swiss roll)
  - The neighbour relation (with similar colours) is kept in the new space
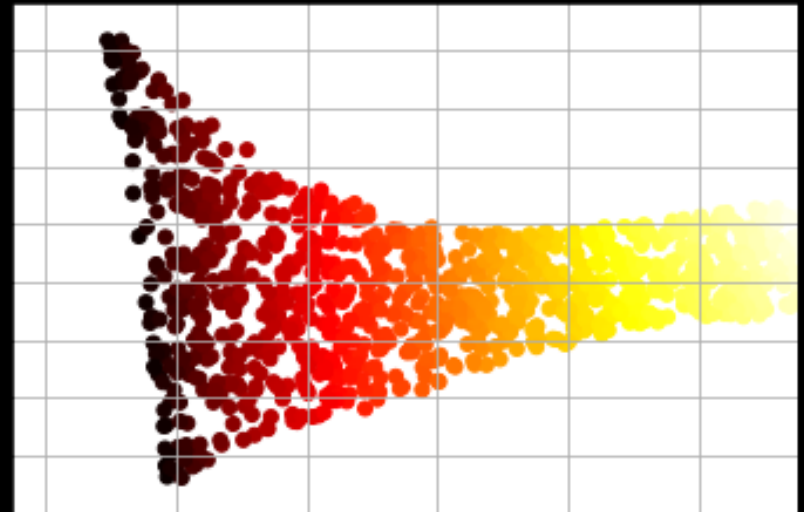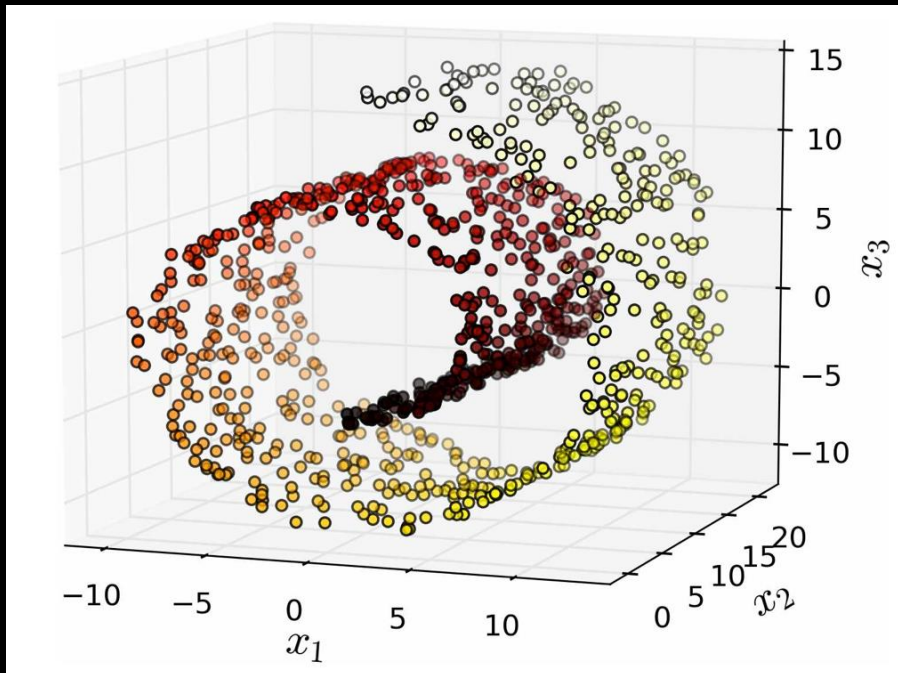
Comparison
Comparing PCA, kPCA, LLE

	PCA	kPCA	LLE
Principle	Preserving global variance	Preserving global variance	Preserving local relation (nearest neighbours)
Meaning of new feature	Linear combination of original dims	Unclear	Unclear
Flexibility	Results are deterministic and can't be adjusted	Can be adjusted using the hyperparameters	Can be adjusted using the hyperparameters. n_neighbors reflects a trade-off between local or global patterns.
As input to other analysis?	Yes	Yes	Yes
Computation cost	Normally low, but high for really high dims	Normally low	Normally low

Suggestions of using DR

- First choice: PCA is the baseline DR algorithm and the first choice.

- If PCA is really slow or does not yield good results (visually, or from low variation explained by selected PCs), then try kPCA or LLE.

Other DR methods

- t-Distributed Stochastic Neighbour Embedding (tSNE): preserves similar instances that are close and pushes dis-similar instances apart

- Multi-D Scaling (MDS): tries to reduce D while keep instance distance the same

- ISOMAP: connects each instance to its neighbours then preserves the geodesic distance between instances

Summary

- Dimensionality Reduction

- Curse of dimensionality

- PCA is a linear DR algorithm, which is often used for visualisation. The output of PCA can be used as input to further analysis.

- kPCA and LLE are non-linear and powerful DR. The new features are not easy to interpret.

Thank You
Questions?

Huanfa Chen

huanfa.chen@ucl.ac.uk

44

Workshop
Dimension reduction

- Weekly quiz on Moodle: please finish them before the workshop and we will discuss the quiz in the workshop

- Python notebooks for workshop: will be ready by 5pm Thursday.

- See you in the workshop on Friday 1-3pm