# Introduction

https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction (https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction)
High-dimensional data can be difficult to interpret. One approach to simplification is to assume that the data of interest lie on an **embedded non-linear manifold within the higher-dimensional space**. If the manifold is of low enough dimension, the data can be visualized in the low-dimensional space.

Many non-linear dimensionality reduction (NLDR) methods are related to the linear methods listed below. Non-linear methods can be broadly classified into two groups: those that provide a mapping (either from the high-dimensional space to the low-dimensional embedding or vice versa), and those that just give a visualization. In the context of machine learning, mapping methods may be viewed as a **preliminary feature extraction step**, after which pattern recognition algorithms are applied. Typically those that just give a visualization are based on proximity data - that is, distance measurements.

**A side note:**

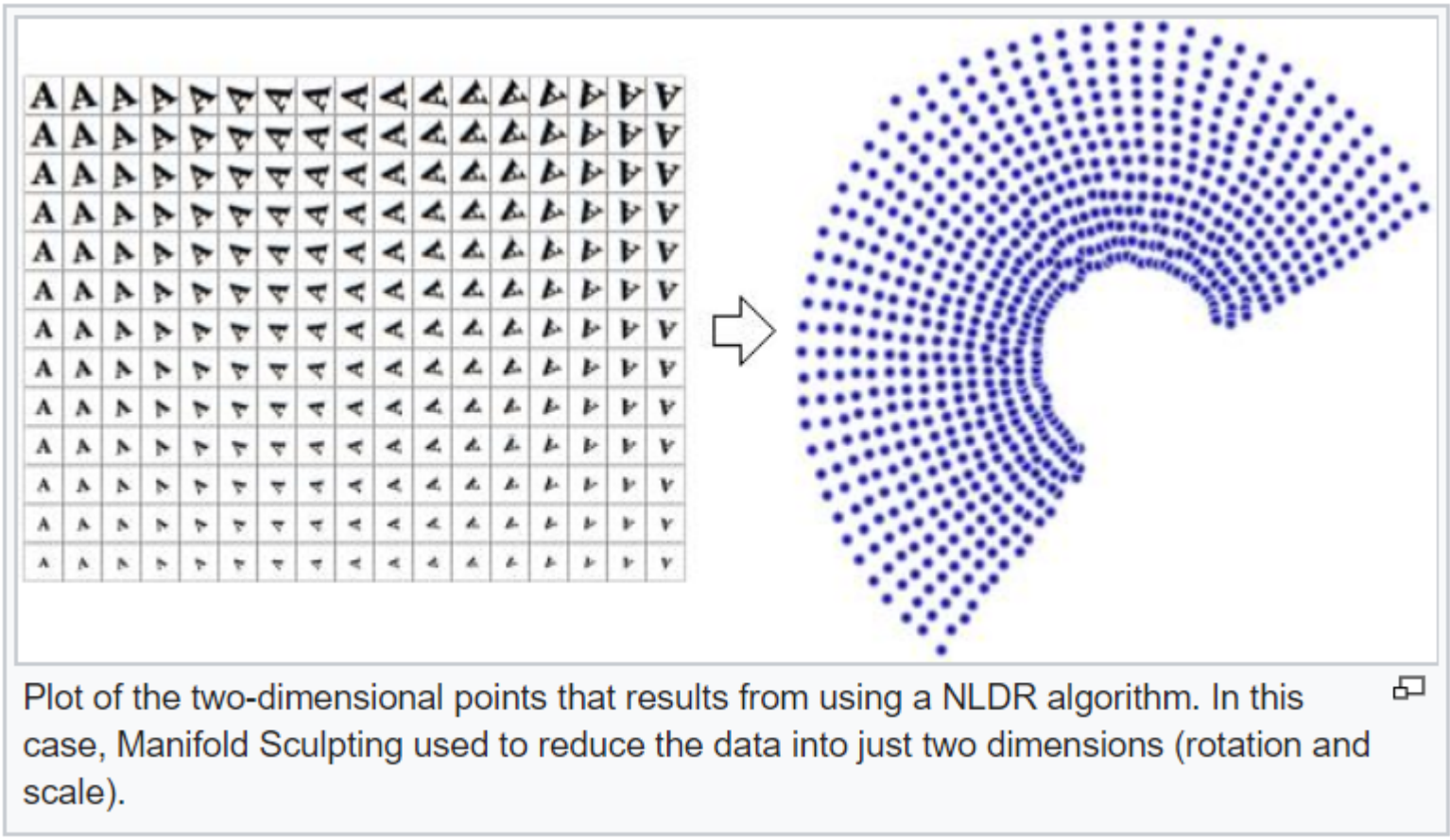
- The most important task for NLDR is to find the most relevant (meaning catching most important features) and most economic (meaning no redundant dimension) representation of the original data. In the resulting lower dimension vector, each entry might be a linear and often a nonlinear combination of the original data.
- In mathematics, a manifold is a topological space that locally resembles Euclidean space near each point. One-dimensional manifolds include lines and circles. Two-dimensional manifolds are also called surfaces. Examples include the plane, the sphere, and the torus, which can all be **embedded (formed without self-intersections)** in three dimensional real space.
- The nonlinear dimension reduction here refers to the transformation from one basis to another is not linear. That is the transformation cannot be through a basis as usual (such as in PCA). Therefore, the nonlinearity is not related to the nonlinear relation between features and target.
- In principle, NLDR such as using neural network can replace the LDR such as using PCA, but not in the other around. However, classical LDR has clear pictures, while NLDR normally is compacted in a black box.

# An example of the application of NLDR

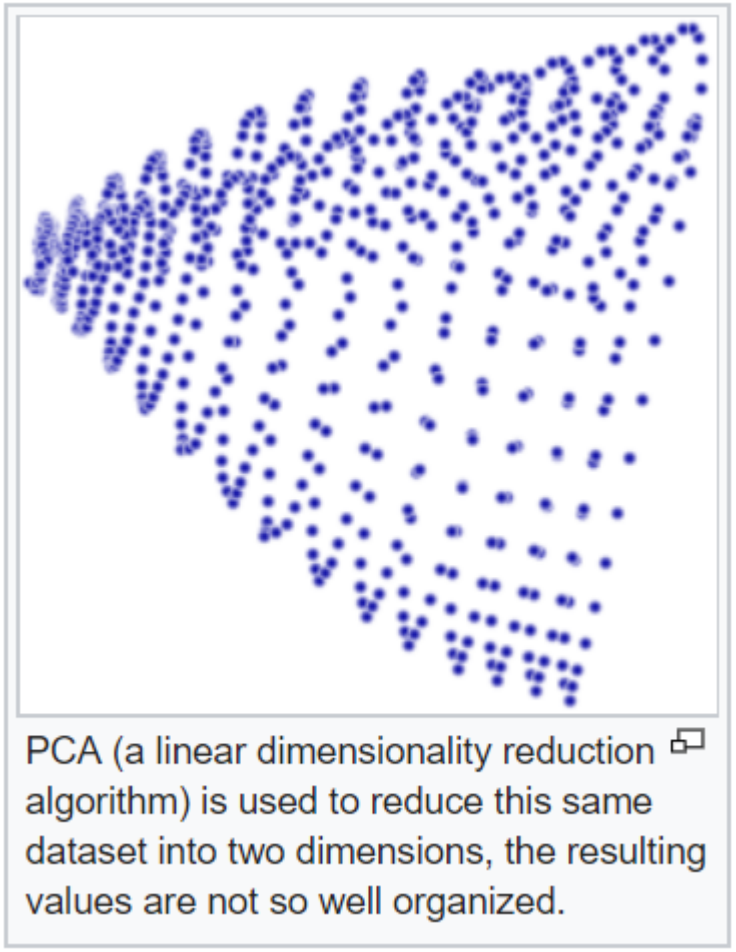
Consider a dataset represented as a matrix such that each row represents a set of features that describe a particular instance of something. If the number of features is large, then the space of unique possible rows is exponentially large. Thus, the larger the dimensionality, the more difficult it becomes to sample the space. Many machine learning algorithms, for example, struggle with high-

dimensional data. This has become known as the curse of dimensionality. Reducing data into fewer dimensions often makes analysis algorithms more efficient and more stable. Moreover, humans often have difficulty comprehending data in many dimensions. Thus, reducing data to a small number of dimensions is useful for visualization purposes.

The reduced-dimensional representations of data are often referred to as "intrinsic variables", whicn imply that these are the values from which the data was produced. For example, consider a dataset that contains images of a letter 'A', which has been scaled and rotated by varying amounts. Each image has 32x32 pixels. Each image can be represented as a vector of 1024 pixel values. Each row is a sample on a **two-dimensional manifold in 1024-dimensional space** (a Hamming space). The intrinsic dimensionality is two, because two variables (rotation and scale) were varied in order to produce the data. Information about the shape or look of a letter 'A' is not part of the intrinsic variables because it is the same in every instance. Nonlinear dimensionality reduction will discard the correlated information (the letter 'A') and recover only the varying information (rotation and scale). The image to the right shows sample images from this dataset (not all input images are shown), and a plot of the two-dimensional points that results from using a NLDR algorithm (in this case, Manifold Sculpting was used) to reduce the data into just two dimensions.



Plot of the two-dimensional points that results from using a NLDR algorithm. In this case, Manifold Sculpting used to reduce the data into just two dimensions (rotation and scale).



PCA (a linear dimensionality reduction algorithm) is used to reduce this same dataset into two dimensions, the resulting values are not so well organized.

By comparison, if Principal component analysis, which is a linear dimensionality reduction algorithm, is used to reduce this same dataset into two dimensions, the resulting values are not so well organized. **This demonstrates that the high-dimensional vectors (each representing a letter 'A') that sample this manifold vary in a non-linear manner.**

## Manifold learning algorithms

https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction (https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction)
Some manifold learning algorithms are listed below.

- SDD Maps

- Isomap
- Locally-linear embedding
- Laplacian eigenmaps
- Sammon's mapping
- Self-organizing map
- Principal curves and manifolds
- Autoencoders
- Gaussian process latent variable models
- Contagion maps
- Curvilinear component analysis
- Curvilinear distance analysis
- Diffeomorphic dimensionality reduction
- Kernel principal component analysis
- Manifold alignment
- Diffusion maps
- Hessian Locally-Linear Embedding (Hessian LLE)
- Modified Locally-Linear Embedding (MLLE)
- Relational perspective map
- Local tangent space alignment
- Local multidimensional scaling
- Maximum variance unfolding
- Nonlinear PCA
- Data-driven high-dimensional scaling
- Manifold sculpting
- t-distributed stochastic neighbor embedding
- RankVisu
- Topologically constrained isometric embedding
- Uniform Manifold Approximation and Projection

Below are few examples.

# Autoencoders

## Introduction

- An autoencoder is a **unsupervised** neural network which is trained to approximate the identity function. That is, it is trained to map from a vector of values to the same vector. When used for dimensionality reduction purposes (also do opposite), one of the hidden layers in the network is limited to contain only a small number of network units.
- As compared to PCA, autoencoders has the following features in terms of dimensionality reduction:
  - An autoencoder can learn non-linear transformations.
  - It doesn't have to learn dense layers. It can use convolutional layers to learn which is better for video, image and series data.
  - It is more efficient to learn several layers with an autoencoder rather than learn one huge transformation with PCA.
  - An autoencoder provides a representation of each layer as the output.
  - It can make use of pre-trained layers from another model to apply transfer learning to enhance the encoder/decoder.
- Application examples
  - Image Coloring: Autoencoders are used for converting any black and white picture into a colored image.
  - Feature variation: It extracts only the required features of an image and generates the output by removing any noise or unnecessary interruption.
  - Dimensionality Reduction: The reconstructed image is the same as our input but with reduced dimensions. It helps in providing a similar image with a reduced pixel value. The small size can help search image from a big picture database.
  - Denoising Image: The input seen by the autoencoder is not the raw input but a stochastically corrupted version. A denoising autoencoder is thus trained to reconstruct the original input from the noisy version.
  - Anomaly detection
  - Information retrieval
- Variation of Autoencoders
  - Convolution Autoencoders
  - Sparse Autoencoders
  - Deep Autoencoders
  - Contractive Autoencoders
  - variational autoencoders

See some details about the mathematical implementation of autoencoders in the following links. There are various ways to implement antoencoders. https://www.jeremyjordan.me/autoencoders/ (https://www.jeremyjordan.me/autoencoders/)
https://www.jeremyjordan.me/variational-autoencoders/ (https://www.jeremyjordan.me/variational-autoencoders/)
https://davidstutz.de/the-mathematics-of-variational-auto-encoders/ (https://davidstutz.de/the-mathematics-of-variational-auto-encoders/)

## t-distributed stochastic neighbor embedding

See details in t-SNE in the "t-distributed stochastic neighbor embedding t-SNE.pdf" under the /algorithm folder.

# Why nonlinear basis change is not common?

https://math.stackexchange.com/questions/1096325/change-of-basis-with-a-nonlinear-operator
(https://math.stackexchange.com/questions/1096325/change-of-basis-with-a-nonlinear-operator)

Given a vector space $V$ and its two basis: $B$ given by vectors $\{e_i\}$ and $B'$ given by vectors $e_i'$, why are the two basis connected necessarily by a linear transformation

$$e_i' = \Lambda_{ij} e_j$$

Why do we exclude the possibility of a nonlinear transformation such as

$$e_i' = \Lambda_{ijk} e_j e_k$$

- Is possible but isn't terribly useful. The important fact is that always exists a linear transformation between two bases. Also, a nonsingular linear transformation always transforms a basis in a basis. With nonlinear transformations this fails.
- A non-linear change of basis takes us out of linear algebra and the crucial identification between vector and coordinates becomes unworkable in the linear algebraic context. See details in the link above.

**Comments**

- After a nonlinear transformation, we may have, e.g., $z_1 = x^2$, $z_2 = xy$ and $z_3 = y^2$ (from a 2D two 3D transformation).
- Thus a signal $s = az_1 + bz_2 + cz_3$ ($a, b, c$ are constants) is formally in linear format in the new basis, but nonlinear in the original basis.
- Although mathematically it is not so common to do a nonlinear basis change, and it is not very useful, in machine learning, the nonlinear basis change can be very useful. As shown in many nonlinear dimension reduction algorithms shown earlier, representing the original data in a nonlinear basis (often we don't even know what it is like ) can help resolve many machine learning problems. For example, face encoding, which represents a person's image with a much lower dimension (learned from the nonlinear convolutional neural network) can significantly increase the accuracy of face recognition. However, this type of work cannot be easily achieved by the conventional linear dimension reduction techniques such as PCA.

# Explore the linearity/nonlinearity of the data

- For a 1D or 2D regression problems, it is easy to just plot the $y(x)$ relation to check linearity/nonlinearity visually. For 1D to 2D or even 3D classification problems, it is possible to check the linearity by visualization.
- Randomly choose one or two dimensions of data and do exploratory data analysis, preferably graph analysis.
- Is it possible to do a nonlinear mapping (such as manifold sculpting or t-SNE, etc.) and then compare it with a linear mapping (such as PCA), and then choose the linear or nonlinear by judging which mapping is better?