Dimensionality Reduction

Algorithms in Machine Learning, ISAE-SUPAERO

Fabrice Jimenez
Data Scientist & Artificial Intelligence Engineer
Airbus Commercial Aircraft

Dimensionality reduction: why?

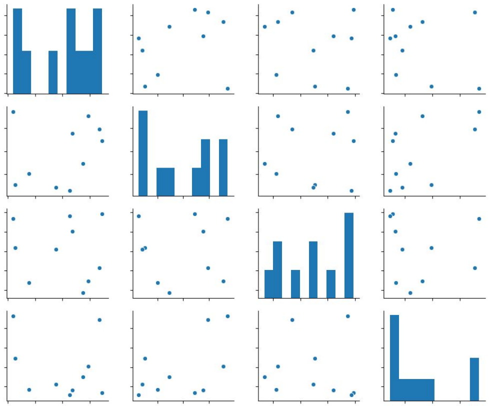
What is high dimension?

v1	v2	v3	v4	...	v365
8.4	15	2.2	0.5	...	65.8
9.1	10	5.1	-4.3	...	-7
...	...	...	...	...	...

Many, many features... Maybe more features than data points!

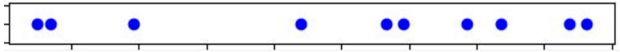
Let's consider 10 points in 20 dimensions v1 ... v20

4D?
nD??

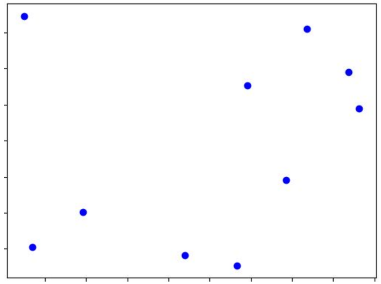


First issue: visualization

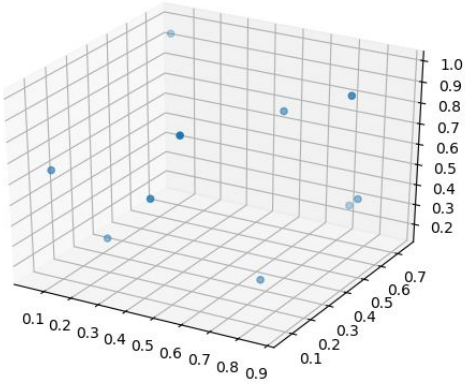
1D



2D



3D



The Curse of Dimensionality

One so-called “Curse”, several effects…

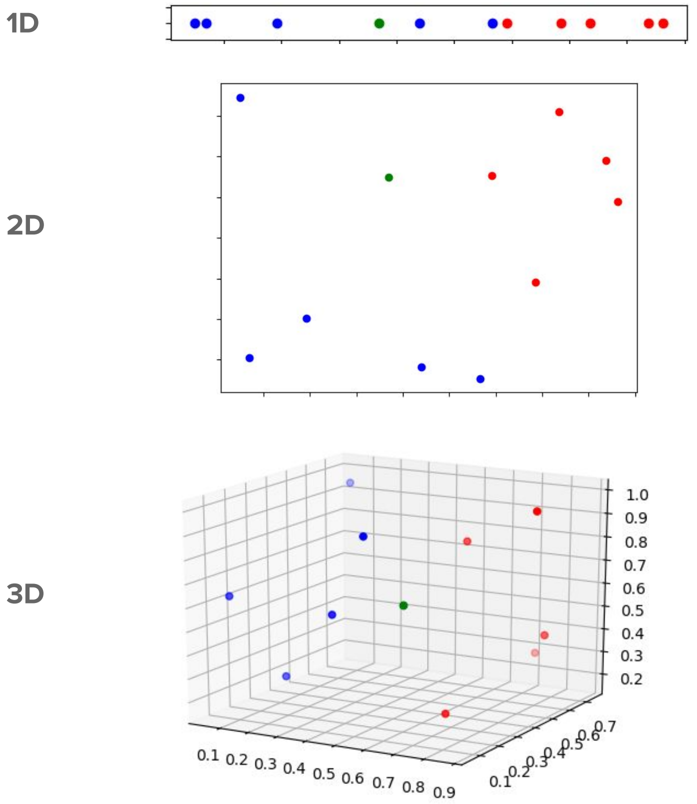
Effect n°1 - Data Coverage

Statistics, Machine Learning: generalization / estimation of population, from a reduced sample which is representative

Representative = training set covering “enough” portions of feature space

Imagine we need to train a binary classifier (red or blue points)…

How would you classify the new green point?

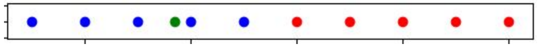


The Curse of Dimensionality

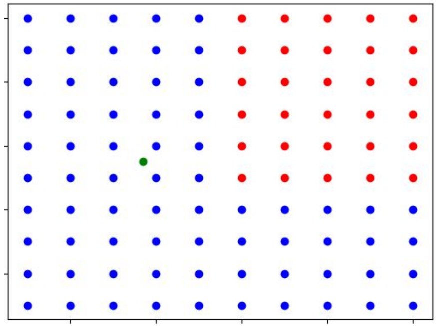
How would you classify the new green point?

One so-called “Curse”, several effects...

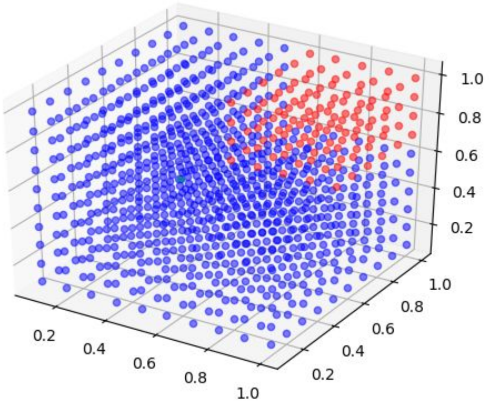
1D



2D



3D



Effect n°1 - Data Coverage

Statistics, Machine Learning: generalization / estimation of population, from a reduced sample which is representative

Representative = training set covering “enough” portions of feature space

Imagine we need to train a binary classifier (red or blue points)...

Ideally, training samples “uniformly” distributed on the feature space...

If in 1D you need ~10 points:
—> in 2D you need ~100 points
—> in 3D you need ~1000 points
—> in nD you need ~10^n points

For fixed number of samples, many dimensions = poor coverage
= classifier with poor performance!!!

The Curse of Dimensionality

More details:
On the surprising behavior of distance metrics in high dimensional space
Charu C. Aggarwal et. al.

One so-called “Curse”, several effects…

Effect n°2 - Distance concentration

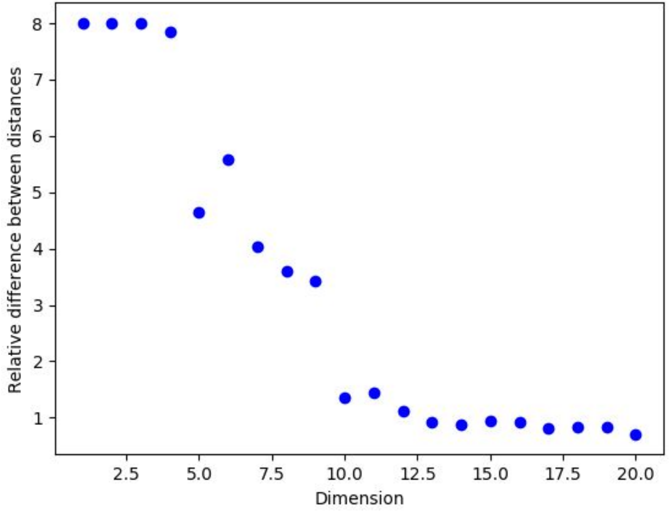
$$\lim_{d \to \infty} E\left(\frac{\text{dist}_{\max}(d) - \text{dist}_{\min}(d)}{\text{dist}_{\min}(d)}\right) \to 0$$

With high dimension, sparsity of points increases, and Euclidean distance becomes more and more equal between all points

Notion of distance important for many ML tasks: clustering, outlier detection…

Many dimensions = distance obsolete = results obsolete…

Let’s take our dataset with 10 points in 20 dimensions:
—> For each dimension d, keep the d first variables
—> Compute all pairwise distances
—> Study the relative difference between distances



Roughly, when d exceeds the number of points, distance becomes useless…

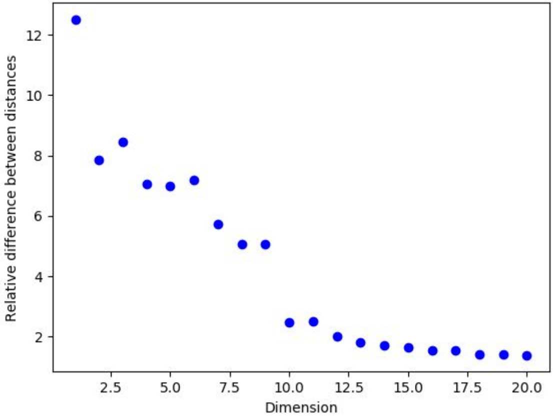
The Curse of Dimensionality

One so-called “Curse”, several effects...

Effect n°3 - Noise pollution

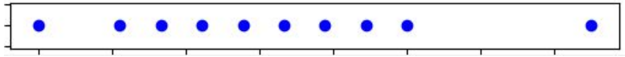
When looking for a pattern in low dimension, if you add many dimensions irrelevant to this pattern: noise conceals the actual information

Relative difference between distances:
➔ Main jump between 1D and 2D (introduction of noise)
➔ Second jump roughly when d > number of points (effect n°2)

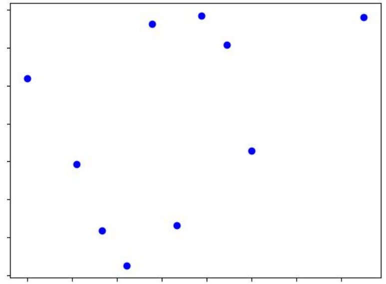


Can you identify the pattern?

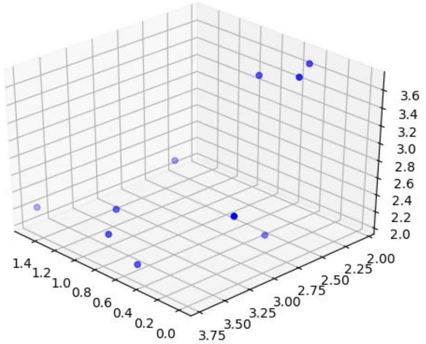
1D



2D



3D

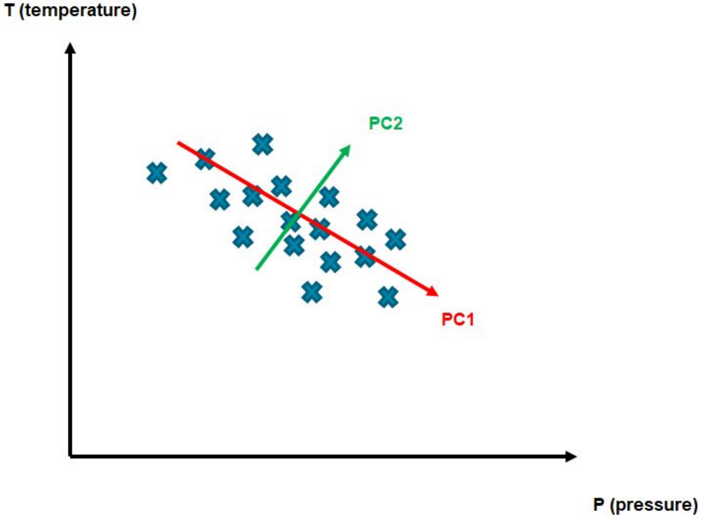


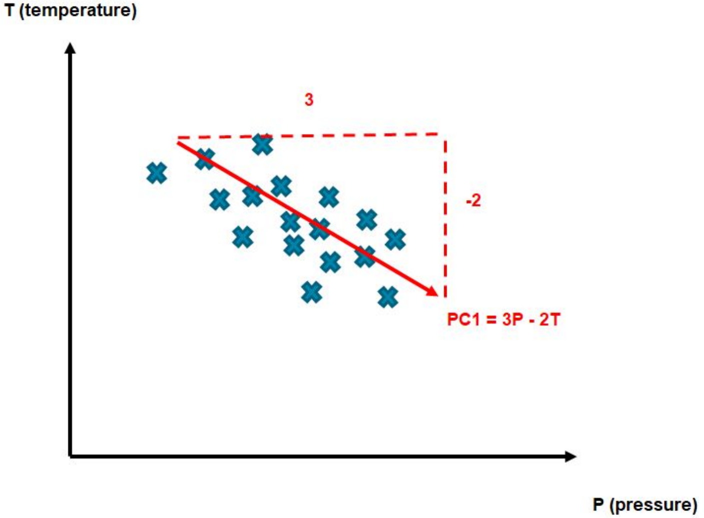
PCA and information pattern

Effects especially valid if dimensions are drawn from independent distributions
Correlations reinforce patterns visibility, but how to distinguish correlation information, noise, and pattern information ?

Principal Components Analysis → Linear combination of initial features

- **→ First components maximize the projected variability of the dataset = spread distances = often main information patterns**
- **→ Correlated features are gathered in the same components: removes correlation redundancy**





Defeat the Curse of Dimensionality? Compute PCA and keep the first principal components might be a solution…

PCA Computation

Dataset Matrix

$$M = \begin{bmatrix} X_{11} & \dots & X_{1p} \\ \dots & \dots & \dots \\ X_{n1} & \dots & X_{np} \end{bmatrix}$$

Centering

$$\bar{M} = \begin{bmatrix} X_{11} - \bar{X}_1 & \dots & X_{1p} - \bar{X}_p \\ \dots & \dots & \dots \\ X_{n1} - \bar{X}_1 & \dots & X_{np} \bar{X}_p \end{bmatrix}$$

Scaling

$$\tilde{M} = \begin{bmatrix} \frac{X_{11} - \bar{X}_1}{\sigma(X_1)} & \dots & \frac{X_{1p} - \bar{X}_p}{\sigma(X_p)} \\ \dots & \dots & \dots \\ \frac{X_{n1} - \bar{X}_1}{\sigma(X_1)} & \dots & \frac{X_{np} \bar{X}_p}{\sigma(X_p)} \end{bmatrix}$$

Covariance matrix: $$cov(M) = \frac{1}{n} \times \bar{M}^T . \bar{M}$$

Correlation matrix: $$cor(M) = \frac{1}{n} \times \tilde{M}^T . \tilde{M}$$

Eigenvalue decomposition: $A = V^T . D . V$ **where** $D = diag\left(\lambda_1, \lambda_2, ..., \lambda_n\right)$

- **applies to square matrices n x n which are diagonalizable**
- **V: columns of V are eigenvectors Vi for the corresponding eigenvalues** λ_i **i.e** $AV_i = \lambda_i V_i$

PCA Computation:

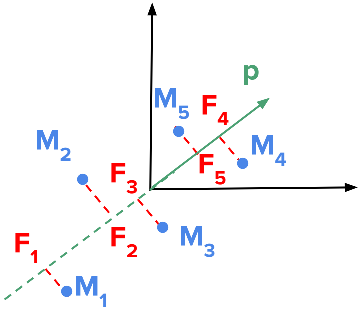
- **Correlation matrix is square, symmetric, real:** *diagonalizable in orthonormal basis (Spectral theorem…)*
- **Principal Components:** *eigenvectors of correlation matrix*
- **Explained variance by component i:** *ith eigenvalue*

Exercise: PCA and diagonalization

Demonstrate that principal components of the PCA are the eigenvectors of the correlation matrix

1/ Consider principal component with direction vector p
Our dataset M projected on this component becomes F such as:

$$F = \tilde{M}p \qquad \begin{cases} \bar{F} = 0 \\ \|p\| = 1 \end{cases}$$



2/ Write the explained variance on this component:

$$var\left(F\right) = \frac{1}{n}\sum_{i=1}^{n}\left(Fi - \bar{F}\right)^2 = \frac{1}{n}\sum_{i=1}^{n} Fi^2 = \frac{1}{n}F^T F$$

$$= \frac{1}{n}\left(\tilde{M}p\right)^T \tilde{M}p$$

$$= \frac{1}{n}p^T\left(\tilde{M}^T\tilde{M}\right)p$$

$$var\left(F\right) = p^T cor\left(M\right)p$$

3/ We want the component to explain a maximum of variance, we need to formulate the following maximization problem:

$$\begin{cases} maximize\left[p^T cor\left(M\right)p\right] \\ \|p\| = p^T p = 1 \end{cases} \Longleftrightarrow \begin{cases} \mathcal{L}\left(p\right) = p^T cor\left(M\right)p - \mu\left(p^T p - 1\right) \\ \dfrac{\partial\mathcal{L}}{\partial p}\left(p\right) = 2cor\left(M\right)p - 2\mu p = 0 \end{cases}$$

$$\Longleftrightarrow cor\left(M\right)p = \mu p$$

p must be eigenvector of the correlation matrix

4/ Explained variance on the component is then:

$$var\left(F\right) = p^T cor\left(M\right)p = p^T\mu p = \mu$$

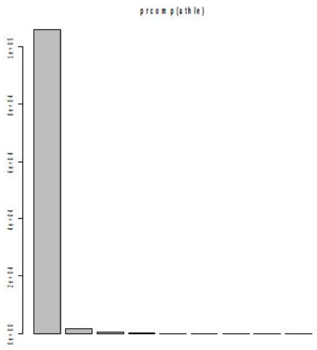
Explained variance is eigenvalue of the correlation matrix

PCA Interpretation

Be careful with quick interpretations!

Example 1:

One of the variables "draws" all the variance of the dataset: careful with scaling!



No scaling: *1 variable with high variance "draws" all PCA effect to itself*
Scaling: *1 noise variable will have same variance as information variable*

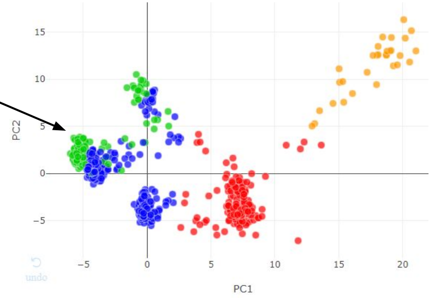
If different units: *scaling mandatory, otherwise does not make sense!*

Think about these examples when you get your results...

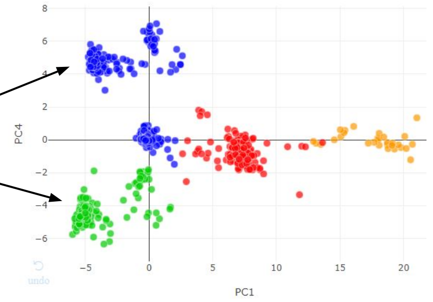
Example 2:

Be careful with proximity of points in projection graph

Green and blue points very close...



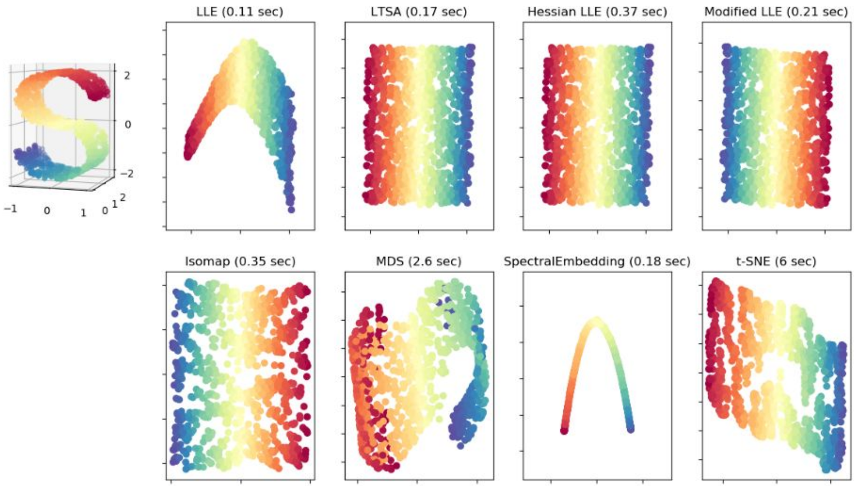
Actually no!
But not visible in the projection PC1 - PC2



PCA vs Manifold Learning

PCA is nice when target pattern is linked to variance in a linear direction…

Unfortunately, it's not always the case: patterns may be non-linear. Non-linear projection methods exist to reduce dimension:



(source: scikit-learn)

Manifold Learning

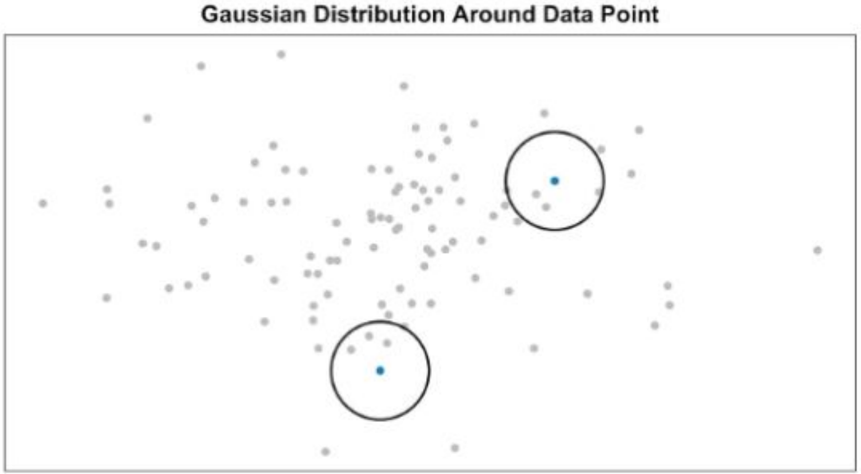
We will see 1 example in this course: t-SNE (t-distributed Stochastic Neighbor Embedding)

t-distributed Stochastic Neighbor Embedding (t-SNE)

Particular technique of dimensionality reduction: it transcribes the similarities between points in a lower dimension

More details on t-SNE:
Visualizing data using t-SNE
Laurens van der Maaten et. al.

Similarities in initial space: normal probability density around each point



Perplexity: parameter ruling the coverage
➜ to change sigma

$$P_{ij} = \frac{exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(-||x_i - x_k||^2 / 2\sigma_i^2)}$$

then

$$P_{ij} = \frac{P_{ij} + P_{ji}}{2n}$$

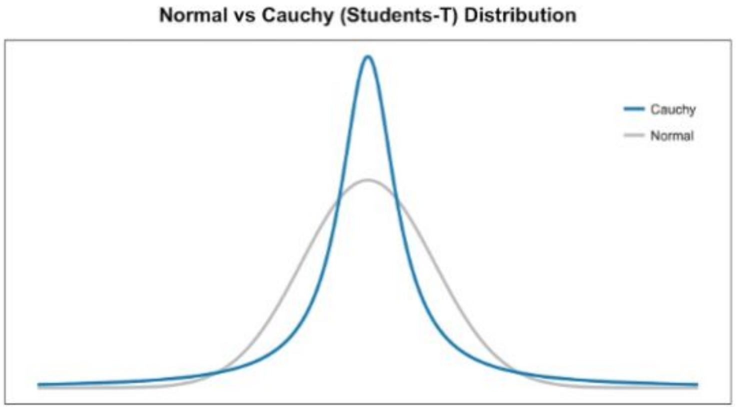
t-distributed Stochastic Neighbor Embedding (t-SNE)

Particular technique of dimensionality reduction: it transcribes the similarities between points in a lower dimension

Similarities in reduced space: Cauchy probability density around each point

$$Q_{ij} = \frac{(1+||y_i-y_j||^2)^{-1}}{\sum_{k\neq i}(1+||y_i-y_k||^2)^{-1}}$$

Learning rate and stopping criteria: parameters impacting accuracy of the projection



Objective:
minimize difference between distance distributions in initial and reduced space

How:
optimization (gradient descent or others) on the Kullback-Leibler divergence KL(P,Q)

Going from normal to Cauchy distribution: stretches the distances → highly differentiated clusters (sometimes too much!)

Pros: high transcription capability of complex patterns

Cons: sensitive to tuning (optimization algorithm)

Dimensionality Reduction: Application

It's time to play on your own...

Main interest = discover the dataset, play with PCA, t-SNE parameters, and take a step back for interpretation

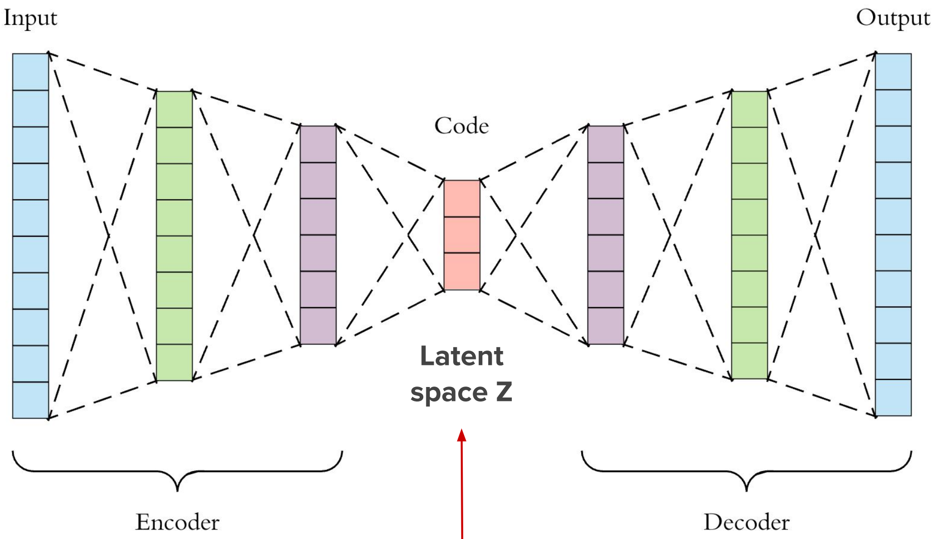
Autoencoders

PCA is powerful, but only for linear behaviors. t-SNE is adapted for non-linear setup, but not scalable…

What about neural networks for dimensionality reduction? Adapted for non-linear setup, and handles well a lot of data…

Autoencoders

Input X



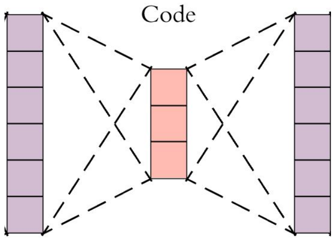
Latent space Z

Target X

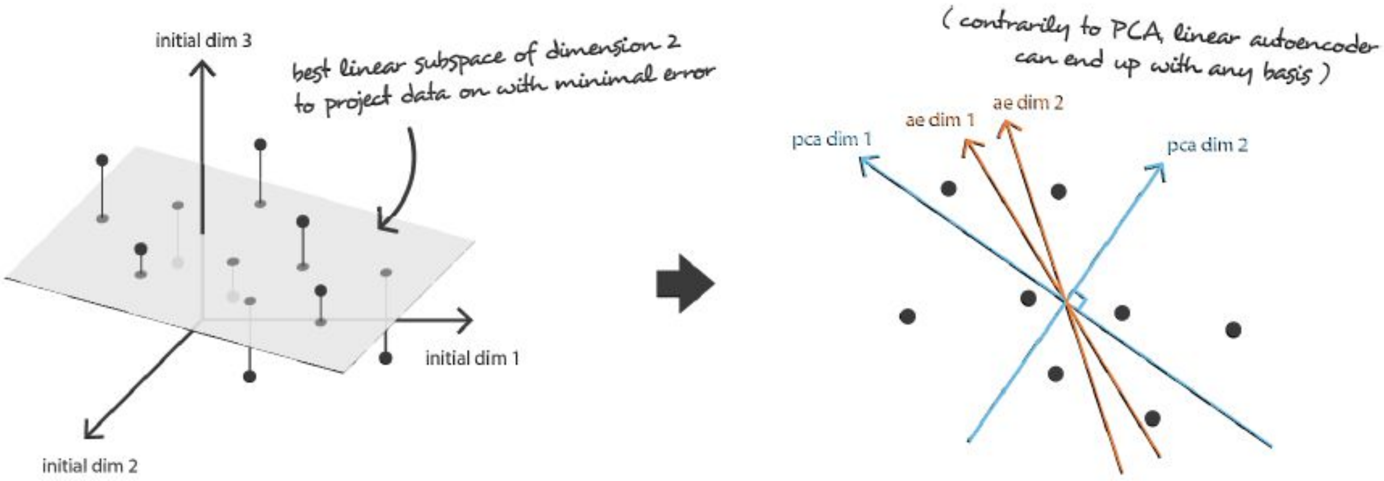
Loss function:
- **Continuous: MSE**
- **Categorical: Cross-entropy**

Lower dimensionality → as much information as possible to reconstruct X

Autoencoders: link with PCA



Linear autoencoder (1 hidden layer and no activation function) equivalent to PCA, but less constraints...



Data in the full initial space

In order to reduce dimensionality, PCA and linear autoencoder target, in theory, the same optimal subspace to project data on...

Data projected on the best linear subspace

... but not necessarily with the same basis due to different constraints (in PCA the first component is the one that explains the maximum of variance and components are orthogonal)

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Autoencoders: avoid overfitting!

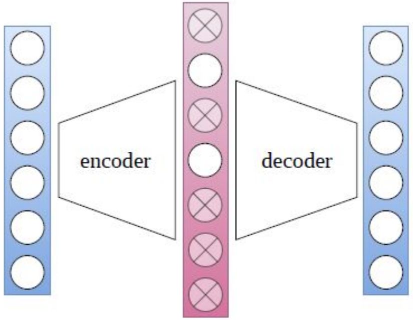
Overfitting = learning the identity function. In general, dim(latent space) << dim(input X) forces to compress information, but...

Even small latent spaces can “remember” an entire training set... So it might be useful to apply other regularization, for example:

Sparse autoencoders

Can have more latent units than inputs, but only a few are allowed to activate together:

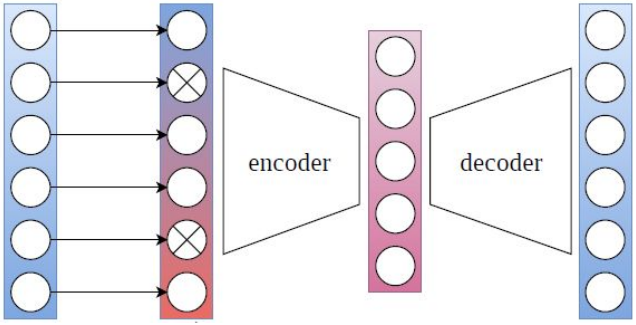
- **L1 penalty**
- **Fixed proportion with KL divergence penalty**
- **Fixed number k**



Denoising autoencoders

Trained to reconstruct corrupted versions of the input, generated for example with:

- **Adding noise with given random distribution**
- **Turning-off some inputs randomly**

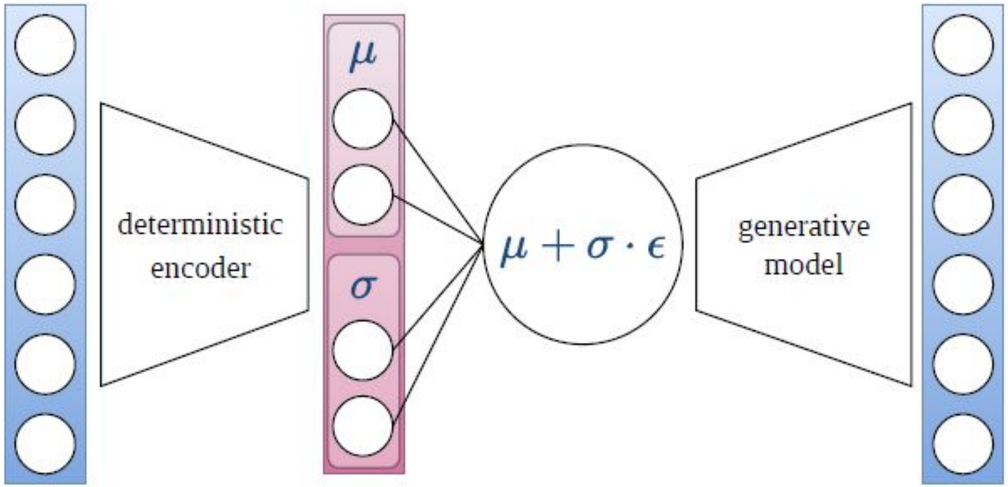


Variational Autoencoders

Limitation of standard Autoencoders: latent space without structure and may not be continuous, overfitting, no sampling possible...

Variational autoencoders: instead of raw units for latent space → parameterized probabilistic model, and we sample the output

Instead of encoding values, we encode gaussian distributions. The decoder becomes a generative model

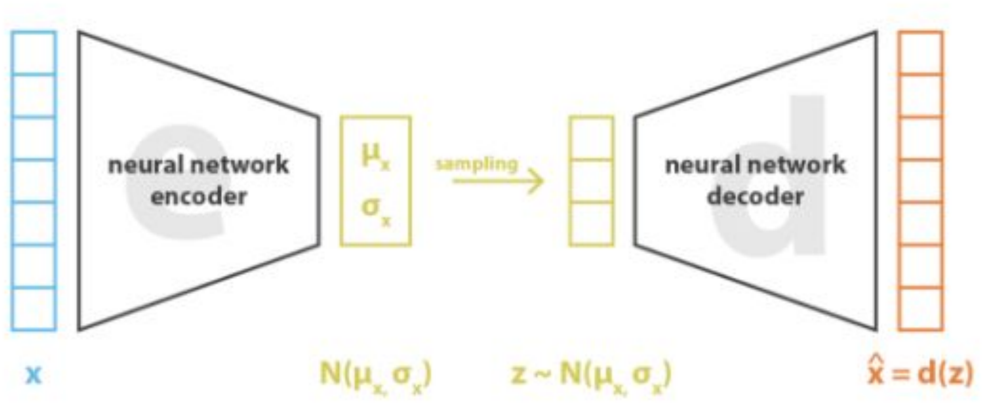


Variational Autoencoders: the loss

But encoding a distribution does not prevent the network to learn values (and put all sigma=0 for example) ➔ just like standard AE!

Trick to force the network to learn distributions: adding a term to the loss

➔ Kullback-Leibler divergence between the learned distribution and a standard Gaussian



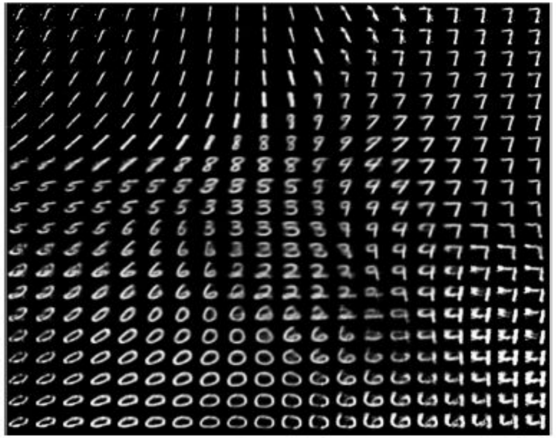
Encourages to learn means that are not too far apart, and variances that are not too small (no punctual distributions)
➔ ensures continuity of latent space
➔ natural regularization

$$\text{loss} = \| x - \hat{x} \|^2 + KL[\, N(\mu_x, \sigma_x), N(0, I)\,] = \| x - d(z) \|^2 + \boxed{KL[\, N(\mu_x, \sigma_x), N(0, I)\,]}$$

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Variational Autoencoders: illustration

Now the latent space has a structure, it is continuous, regularized, and we can sample from it to generate new individuals



jmetzen.github.io/2015-11-27/vae.html, ermongroup.github.io/cs228-notes/extras/vae,
github.com/davidsandberg/facenet

Autoencoders: Application

It’s time to play on your own...

Main interest = build your own network, gain intuition on tuning and performance evaluation

Questions?