IFT 4030/7030,
Machine Learning for Signal Processing
Week5: Machine Learning 2, Non-linear Dimensionality Reduction

Cem Subakan

SpeechBrain

UNIVERSITÉ LAVAL

Mila

Admin

- Homework 1 is out, due on October 24th !!!
  - Notez que le devoir 1 est publié, dû à l'octobre 24!
- How are projects going? I will share the access information for the cluster soon.
  - Comment vont les projets? On va partager les infos pour le cluster bientot.

Admin

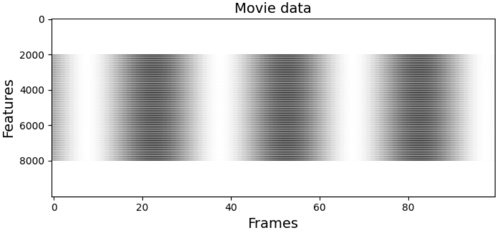
- Homework 1 is out, due on October 24th !!!
  - Notez que le devoir 1 est publié, dû à l'octobre 24!
- How are projects going? I will share the access information for the cluster soon.
  - Comment vont les projets? On va partager les infos pour le cluster bientot.
- Non-linear dimensionality reduction
  - Au'jourdhui: Reduction de dimensionalité non-linéaire

Why dimensionality reduction

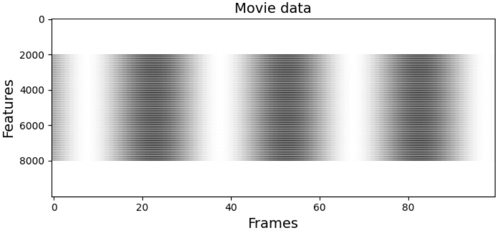
- How many real dimensions in this video? (We have 100 x 100 x 100 dimensions that we can see)
  - Combien de dimensions effectifs dans cette vidéo?

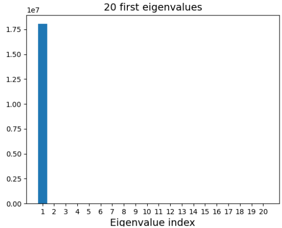
`Watch the video.`

Very low dimensionality



Very low dimensionality





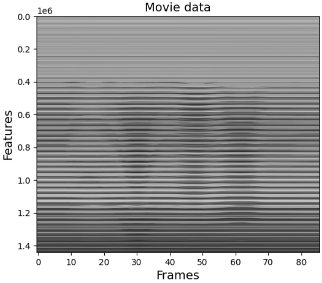
We only need one pixel to convey the same information / On a besoin d'un seul pixel pour le convoi de meme info!

How about this video?

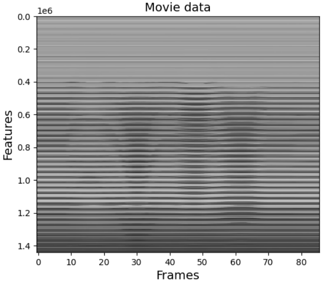
- What is the dimensionality now? / Quelle est la dimensionalité maintenant?
  - Large: $87 \times 800 \times 600 \times 3$.

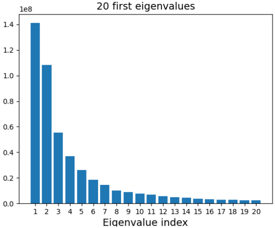
`Watch the video.`

More dimensions?



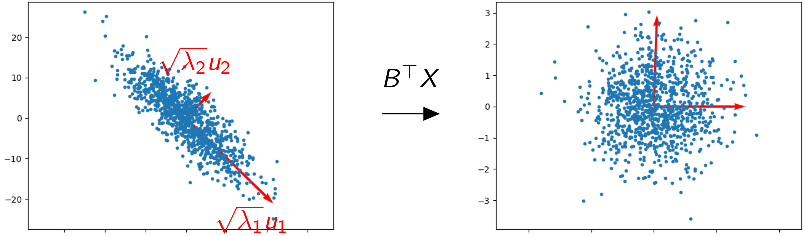
More dimensions?





We have more active dimensions according to PCA, but I think it should be less (4 could do it) / On a plus de dimensions qui sont actives selon PCA, mais je pense 4 serait suffisant.

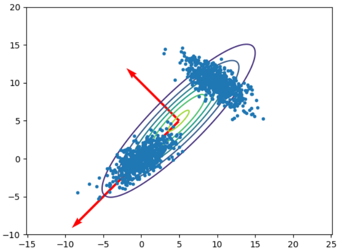
PCA on easy data



$$B^\top = \mathrm{diag}([\sqrt{\lambda_1}, \sqrt{\lambda_2}])^{-1} U^\top.$$

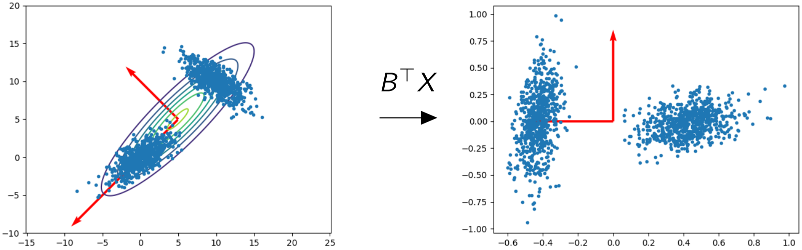
PCA on harder data

- Principal components are not really meaningful here / Les composants principaux n'ont pas du sense nécessairement içi!



PCA on harder data

- Principal components are not really meaningful here / Les composants principaux n'ont pas du sense nécessairement içi!



What's wrong with PCA?

- Principal components are linear, assume Gaussian distribution.
  - Les composants principaux sont linéaires, et supposent une distribution Gaussiénne.

What's wrong with PCA?

- Principal components are linear, assume Gaussian distribution.
  - Les composants principaux sont linéaires, et supposent une distribution Gaussiénne.
- Data is not always (often not in fact) Gaussian.
  - Les données ne sont pas souvent Gaussienné.

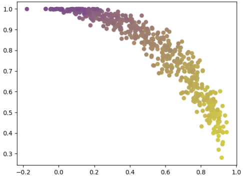
What’s wrong with PCA?

- Principal components are linear, assume Gaussian distribution.
  - Les composants principaux sont linéaires, et supposent une distribution Gaussiénne.
- Data is not always (often not in fact) Gaussian.
  - Les données ne sont pas souvent Gaussienné.
- Can we define non-linear components?
  - Peut-on définir des composants non-linéaires?

Table of Contents

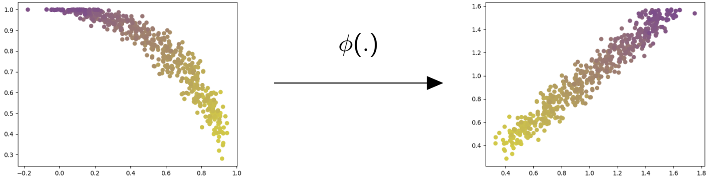
Linearizing the data

- Can we find a feature transformation $\phi()$ such that the data is more Gaussian?
  - Peut-on trouver une transformation $\phi()$ pour que les données sont plus Gaussiennes?



Linearizing the data

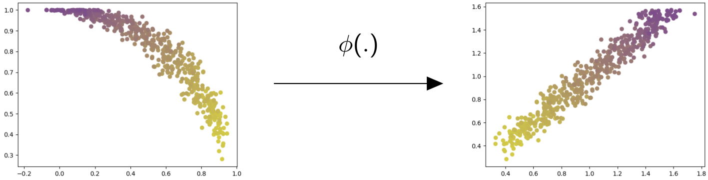
- Can we find a feature transformation $\phi()$ such that the data is more Gaussian?
  - Peut-on trouver une transformation $\phi()$ pour que les données sont plus Gaussiennes?



- Now we can do PCA! / Maintenant on peut faire PCA!

Linearizing the data

- Can we find a feature transformation $\phi()$ such that the data is more Gaussian?
  - Peut-on trouver une transformation $\phi()$ pour que les données sont plus Gaussiennes?



- Now we can do PCA! / Maintenant on peut faire PCA!
- Note that we can use a $\Phi(.)$ function that increases the dimensionality as well. / Notez qu'on peut aussi utiliser une fonction $\Phi(.)$ qui augmente la dimensionnalité.

From PCA to Kernel PCA

- In regular PCA we do / Dans le PCA régulier on fait:

$$C = \text{cov}(X)$$
$$Cu_i = \lambda_i u_i$$

From PCA to Kernel PCA

- In regular PCA we do / Dans le PCA régulier on fait:

$$C = \text{cov}(X)$$
$$Cu_i = \lambda_i u_i$$

- In Kernel PCA we will do / Dans Kernel PCA on fera:

$$C := \text{cov}(\phi(x))$$
$$Cu_i = \lambda_i u_i$$

Why do we call it 'Kernel' PCA?

- Let's do some thinking / Réflechissons sur le sujet

$$C = \frac{1}{N} \sum_{n=1}^{N} \phi(x_n)\phi(x_n)^\top$$

  - Let's get the eigenvectors into the picture / Mettons les valeurs propres dans les équations

$$Cu_i = \frac{1}{N} \sum_{n=1}^{N} \phi(x_n) \left(\phi(x_n)^\top u_i \right) = \lambda_i u_i$$

Why do we call it 'Kernel' PCA?

- Let's do some thinking / Réflechissons sur le sujet

$$C = \frac{1}{N}\sum_{n=1}^{N}\phi(x_n)\phi(x_n)^\top$$

  - Let's get the eigenvectors into the picture / Mettons les valeurs propres dans les équations

$$Cu_i = \frac{1}{N}\sum_{n=1}^{N}\phi(x_n)\left(\phi(x_n)^\top u_i\right) = \lambda_i u_i$$

- We kinda see that $u_i = \sum_n a_{in}\phi(x_n)$. Let's substitute! / On fait cette observation, substitions!

$$\frac{1}{N}\sum_{n=1}^{N}\phi(x_n)\phi(x_n)^\top\sum_{n'}a_{in'}\phi(x_{n'}) = \lambda_i\sum_n a_{in}\phi(x_n)$$

Why do we call it 'Kernel' PCA? (continued)

- Let's multiply both sides by $\phi(l)^\top$, and re-arrange. Multiplions avec $\phi(l)^\top$ de deux cotés et re-arrengons:

$$\frac{1}{N}\sum_{n=1}^{N}\phi(x_l)\phi(x_n)^\top\sum_{n'}a_{in'}\phi(x_n)\phi(x_n') = \lambda_i\sum_{n}a_{in}\phi(x_l)\phi(x_n)^\top$$
$$\rightarrow \frac{1}{N}\sum_{n=1}^{N}k(x_l,x_n)\sum_{n'}a_{in'}k(x_n,x_n') = \lambda_i\sum_{n}a_{in}k(x_l,x_n)$$

- Let's switch to array notation: (remember from lecture 1 that we can do it) / Switchons à la notation d'array (souvenez le cours 1)

$$K^2 a_i = \lambda_i N K a_i$$
$$K a_i = \lambda_i N a_i$$

- We can therefore simply find the eigenvectors of K. Assuming that $\phi(x)$ is zero mean! If not, we need to do more work to remove the mean.
  - On peut alors tout simplement trouver les vecteurs propres de K. Mais notez qu'on a supposé que $\phi(x)$ est zero-mean! On doit travailler davantage si on veut substraitre le moyen.

Projection on to principal components

- Let's think about projection on to eigenvalue v_i of cov($\phi(x)$ / Considérons la projection sur la valeur propre v_i de cov($\phi(x)$).

$$w_i(x) = \phi(x)^\top u_i$$

Projection on to principal components

- Let's think about projection on to eigenvalue v_i of $\mathrm{cov}(\phi(x)$ / Considérons la projection sur la valeur propre v_i de $\mathrm{cov}(\phi(x))$.

$$\begin{aligned} w_i(x) &= \phi(x)^\top u_i \\ &= \sum_n a_{in} \phi(x) \phi(x_n)^\top \end{aligned}$$

Projection on to principal components

- Let's think about projection on to eigenvalue v_i of cov($\phi(x)$ / Considérons la projection sur la valeur propre v_i de cov($\phi(x)$).

$$
\begin{aligned}
w_i(x) =& \phi(x)^\top u_i \\
&= \sum_n a_{in} \phi(x) \phi(x_n)^\top \\
&= \sum_n a_{in} k(x, x_n)
\end{aligned}
$$

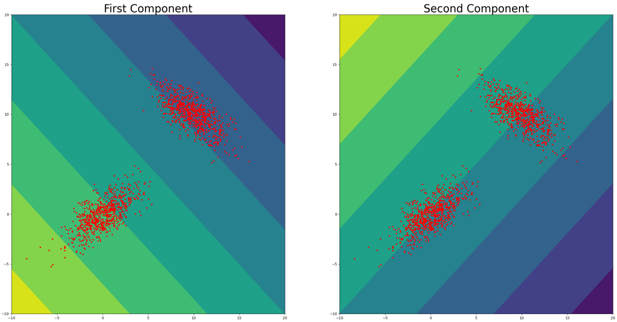
Projection on to principal components

- Let's think about projection on to eigenvalue v_i of $\text{cov}(\phi(x)$ / Considérons la projection sur la valeur propre v_i de $\text{cov}(\phi(x))$.

$$
\begin{aligned}
w_i(x) =& \phi(x)^\top u_i \\
=& \sum_n a_{in}\phi(x)\phi(x_n)^\top \\
=& \sum_n a_{in}k(x, x_n)
\end{aligned}
$$

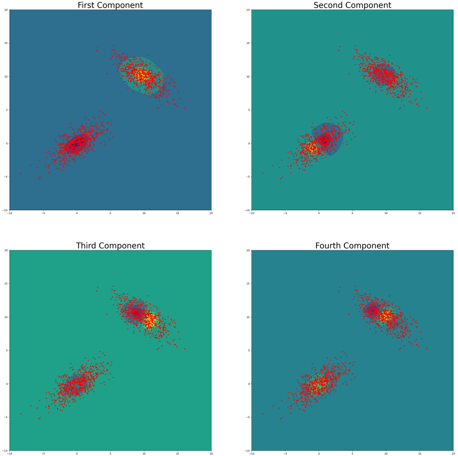
- Notice that the $i'th$ eigenvector a_i of K is of length N, and K is of size $N \times N$, (as opposed to $\text{cov}(x)$ which is $L \times L$). For large datasets this is a limiting factor.
  - Notez que la valeur propre i de K est de longeur N, et K est de taille $N \times N$. (Notez que $\text{cov}(x)$ est de taille $L \times L$) Pour des datasets qui sont large, c'est quelque chose limitante.

Regular PCA components are not very informative



The contours show/ Les contours montre $w_i^\top \begin{bmatrix} x \\ y \end{bmatrix}$.

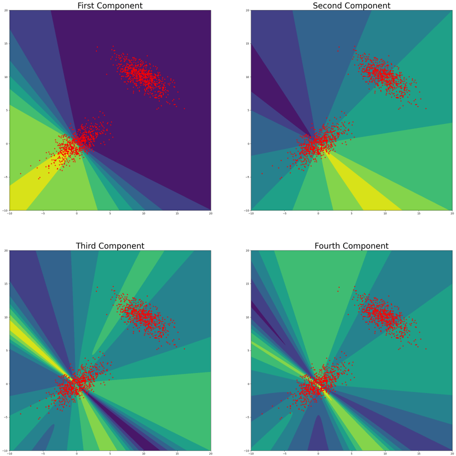
Kernel PCA components



The contours show/ Les contours montre $w_i^\top K\left(\begin{bmatrix} x \\ y \end{bmatrix}, X\right)$.

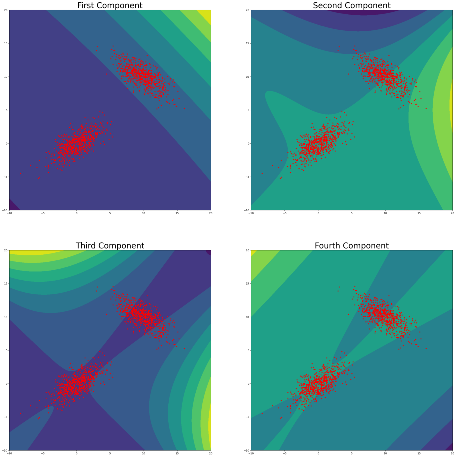
$K = \exp(-\gamma\|x-y\|^2)$ is rbf kernel.

Kernel PCA components



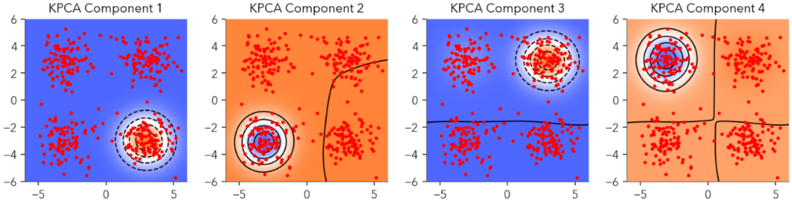
The contours show/ Les contours montre $w_i^\top K\left(\begin{bmatrix} x \\ y \end{bmatrix}, X\right)$. K is sigmoid kernel.

Kernel PCA components



The contours show/ Les contours montre $w_i^\top K\left(\begin{bmatrix}x\\y\end{bmatrix}, X\right)$. K is polynomial kernel.

Kernel PCA Better Performance



with / avec RBF Kernel
Taken from UIUC MLSP class

Kernel PCA Summary

- Performance is highly dependent on the Kernel. / La performance est dependent sur le choix du noyau.

Kernel PCA Summary

- Performance is highly dependent on the Kernel. / La performance est dependent sur le choix du noyau.
- Kernel PCA reveals more interesting variations in the data if the Kernel choice is good / Kernel PCA rèvele des variations plus intéssentes dans les données, si le choix du Kernel est bon.

Kernel PCA Summary

- Performance is highly dependent on the Kernel. / La performance est dependent sur le choix du noyau.
- Kernel PCA reveals more interesting variations in the data if the Kernel choice is good / Kernel PCA rèvele des variations plus intéssentes dans les données, si le choix du Kernel est bon.
- Kernel PCA is more expensive and not suitable for large datasets / Kernel PCA est computationnalement cher large datasets.

Table of Contents

Multi Dimensional Scaling (MDS)

- MDS tries to find a low dimensional embedding such that the pairwise-distances are preserved.
  - MDS trouve un embedding à bas dimension telle que les distances pairwises sont préservés.
- We construct a matrix of pairwise distances / On construit une matrice des distances pairwises

$$d_{i,j} = \|x_i - x_j\|^2$$

- We want to estimate $\widehat{x}$ such that / on veut estimer $\widehat{x}$ telle que

$$\|\widehat{x}_i - \widehat{x}_j\| \approx d_{ij}$$

- How do we do it? / Comment est-ce qu'on fait ça?

Multi Dimensional Scaling (MDS)

- MDS tries to find a low dimensional embedding such that the pairwise-distances are preserved.
  - MDS trouve un embedding à bas dimension telle que les distances pairwises sont préservés.
- We construct a matrix of pairwise distances / On construit une matrice des distances pairwises

$$d_{i,j} = \|x_i - x_j\|^2$$

- We want to estimate $\widehat{x}$ such that / on veut estimer $\widehat{x}$ telle que

$$\|\widehat{x}_i - \widehat{x}_j\| \approx d_{ij}$$

- How do we do it? / Comment est-ce qu'on fait ça?
  - Eigenvectors / Vecteurs Propres

Let's write it out

- $d_{i,j} = \|x_i - x_j\|^2 = x_i^\top x_i + x_j^\top x_j - 2x_i^\top x_j$
- In matrix form / Dans la forme matricielle

$$D = \underbrace{\text{diag}(X^\top X)1_N^\top}_{\text{copy across columns}} + \underbrace{1_N\text{diag}(X^\top X)^\top}_{\text{copy across rows}} - 2X^\top X$$

- Next we make the columns and rows zero mean
  - On fait les colonnes et les lignes zero-moyenne $Z := I_N - 1_N 1_N^\top / N$
  - DZ removes mean column, ZD removes mean row.
  - We apply this on D, and we can show that / On applique ça sur D et on peut montrer que
- $ZDZ = -2ZX^\top XZ = -2(XZ)^\top(XZ) = -2(X - \mathbb{E}[x])^\top(X - \mathbb{E}[x])$

In other words

- Or define the similarity matrix / définissions la matrice de similarité

$$S := -\frac{1}{2}ZDZ = -\frac{1}{2}\left(D - \frac{1}{N}1_{N\times N}D - \frac{1}{N}D1_{N\times N} + \frac{1}{N^2}1_{N\times N}D1_{N\times N}\right)$$

- Then do an eigen decomposition on $S = (X - \mathbb{E}[x])^\top (X - \mathbb{E}[x]) = U\Lambda U^\top$, Set $\widehat{X} = \Lambda^{-1/2}U^\top$, which is an embedding / qui est un embedding.
- For low-dim embeddings, just use low dimensional version of $\widehat{X}$.

MDS with 3dims

Pretty good!

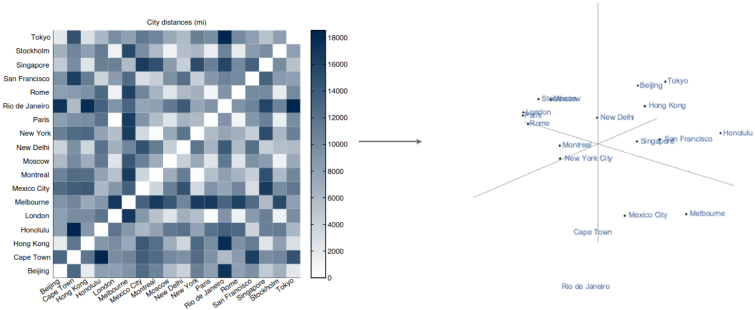


Image taken from UIUC MLSP class

MDS with 2dims

Still pretty good!

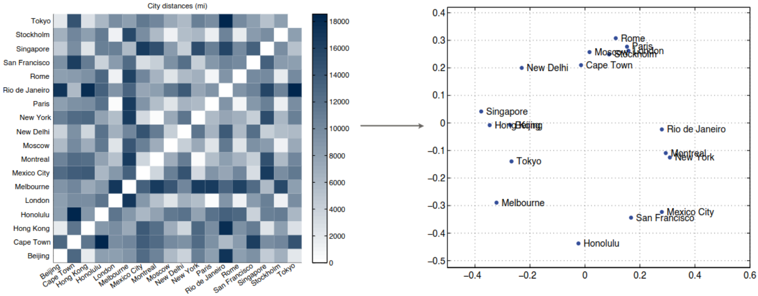


Image taken from UIUC MLSP class

MDS variants

- MDS which uses cosine similarity instead of distances
  - MDS qui utilise la similarité cosinus au lieu de distances
- MDS which uses ranking
  - MDS qui utilise des rankings
- MDS which uses a kernel
  - MDS qui utilise un noyau
- Using only local data
  - Juste usage des données locales

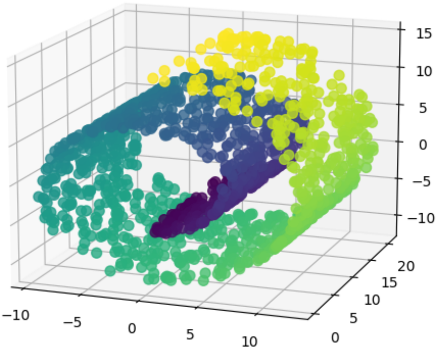
Why MDS?

- MDS finds the implied geometry of the dataset
  - MDS trouve la géometrie de notre dataset.
    - Not obvious in large dimensions / Pas évident si le donnée a beaucoup de dimensions.
- Sort of like PCA
  - Decompose $X^\top X$ au lieu de $XX^\top$.
- Sort of like kPCA also
  - Decompose $X^\top X$
- It's a linear decomposition in the end like PCA and kPCA.
  - Fin de la journée c'est une décomposition linéaire comme PCA et kPCA.

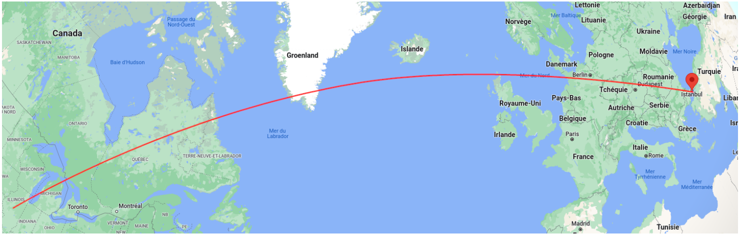
Table of Contents

Actual dimensionality can be misleading

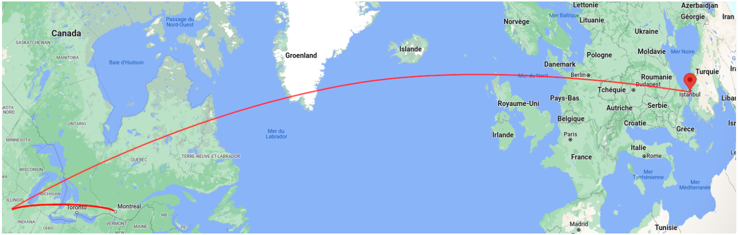
Actual dimensionality can be misleading



Local vs Global

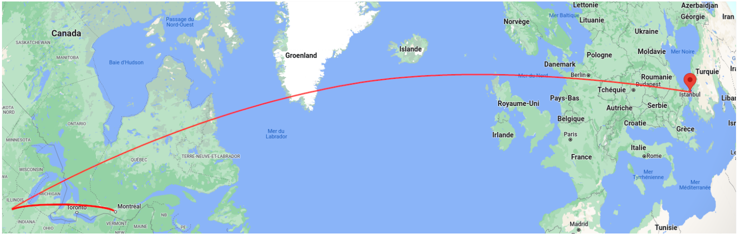


Local vs Global

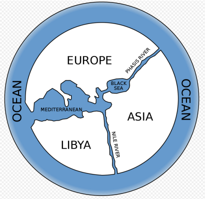


Earth is round, but locally flat (/flatter)! / La terre est ronde mais localement plate ou plus plate

Local vs Global



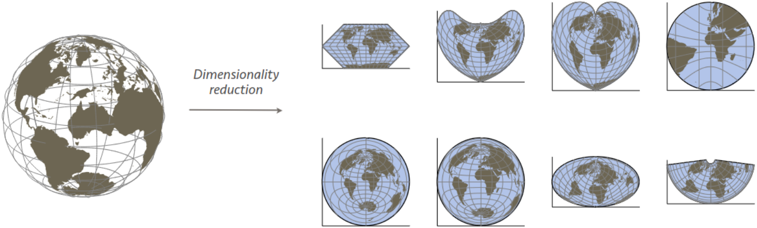
Earth is round, but locally flat (/flatter)! / La terre est ronde mais localement plate ou plus plate



I DON'T MEAN THIS

Map projections

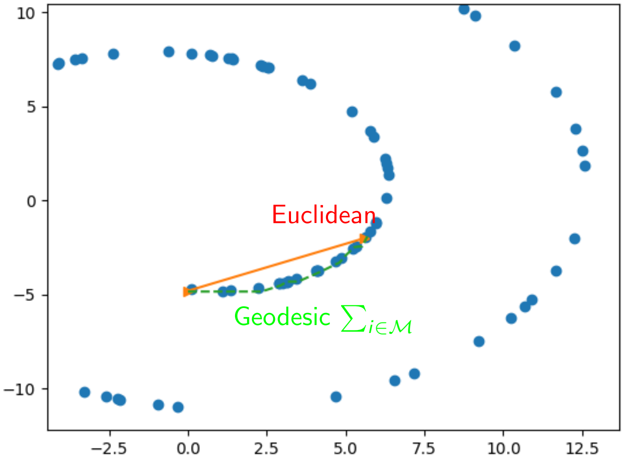
- Many ways to go from 3D to 2D / Plusieurs façons de réduire la dimensionnalité de 3 à 2.



Taken from UIUC MLSP class

Table of Contents

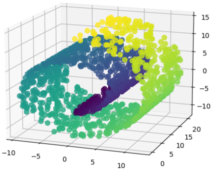
Euclidean vs Geodesic Distance

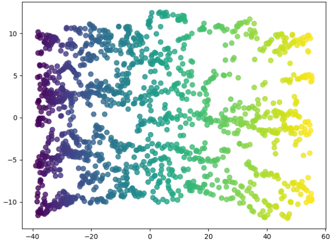


ISOMAP

- MDS with geodesic distances / MDS avec des distances géodesiques.
- Results in embeddings that are more structure aware / Comme ça on obtiens des embeddings qui sont plus consients des structure du manifold.

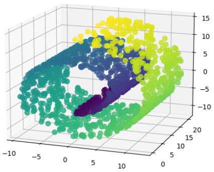
Unroll the swiss roll - ISOMAP

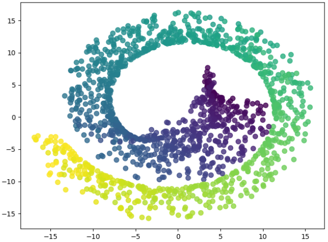




- Pretty good! Maps the geodesic distances well into the euclidean space
  - On est capable de mapper les distances géodesic bien à l'espace euclidienne.

Unroll the swiss roll - MDS



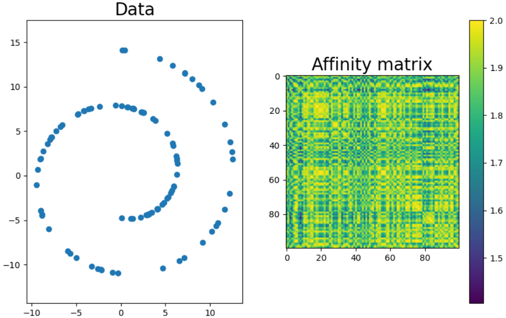


- This doesn't map as well / Ici ça ne map pas aussi bon.

Table of Contents

Neighborhood Approach

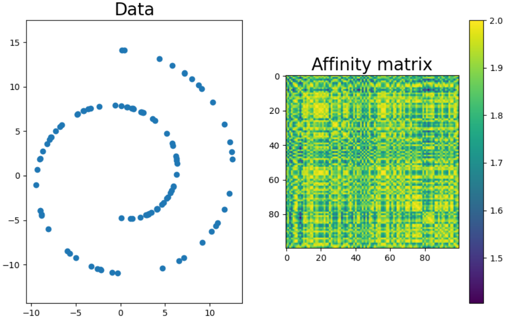
- Make a note of N nearby points and supress the rest, / Trouve N points qui sont plus proches, ignore les autres.



- Affinity Matrix $w_{i,j} = \exp(-\|x_i - x_j\|)$

Neighborhood Approach

- Make a note of N nearby points and supress the rest, / Trouve N points qui sont plus proches, ignore les autres.



- Affinity Matrix $w_{i,j} = \exp(-\|x_i - x_j\|)$
- (The data is not ordered / Le data n'est pas en ordre)

Laplacian Eigenmap

- Minimize / Minimisons:

$$E_{LE} = \min_{y} \sum_{i,j} \|y_i - y_j\|^2 w_{i,j}$$

- We want / On veut:

	Large $\|y_i - y_j\|$	Small $\|y_i - y_j\|$
Large w_{ij}	**Bad**	Good
Small w_{ij}	Good	Don't care

- In matrix form / Dans la forme matricielle:

$$E_{LE} = 2Y^\top LY$$
$$L = \underbrace{W - \text{diag}(1^\top W)}_{\text{Graph Laplacian}} = W - R$$

Laplacian Eigenmap Solution

- The solution (Note that we normalize the Laplacian) / Notez qu'on normalise la Laplacienne.

$$\min_Y Y^\top \underbrace{R^{-1/2} L R^{-1/2}}_{\text{normalized Laplacian}} Y$$

$$s.t.\ Y^\top R^2 T = I$$

- What will be the solution? / C'est quoi la solution? (On a déjà vu des choses similaires)

Laplacian Eigenmap Solution

- The solution (Note that we normalize the Laplacian) / Notez qu'on normalise la Laplacienne.

$$\min_{Y} Y^\top \underbrace{R^{-1/2} L R^{-1/2}}_{\text{normalized Laplacian}} Y$$
$$s.t.\ Y^\top R^2 T = I$$

- What will be the solution? / C'est quoi la solution? (On a déjà vu des choses similaires)
- $\widehat{L} = U \Sigma U^\top$. $\widehat{Y} = U \Sigma^{-1/2}$.

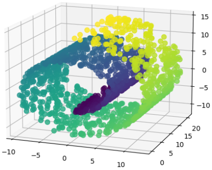
Laplacian Eigenmap Solution

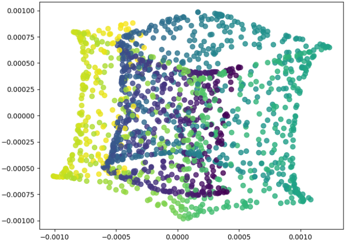
- The solution (Note that we normalize the Laplacian) / Notez qu'on normalise la Laplacienne.

$$\min_Y Y^\top \underbrace{R^{-1/2} L R^{-1/2}}_{\text{normalized Laplacian}} Y$$
$$s.t.\ Y^\top R^2 T = I$$

- What will be the solution? / C'est quoi la solution? (On a déjà vu des choses similaires)
- $\widehat{L} = U \Sigma U^\top$. $\widehat{Y} = U \Sigma^{-1/2}$.
- But this time we take the smallest eigenvalues. (We are minimizing)
  - Mais cette fois on prends les valeur propres qui sont petits. (On est en train de minimiser)

Unroll the swiss roll - Laplacian Eigenmap





- The Kernel distance isn't as good as geodesic distance / La distance de Kernel n'est pas aussi bon pour le problème.

Table of Contents

TSNE

- This is a very popular visualization tool / Un outil très populaire
- The idea is to define inter-sample similarity metrices in high and low dimensional spaces / L'idée est de définir des mésures de similarités dans l'espace originale et la projection

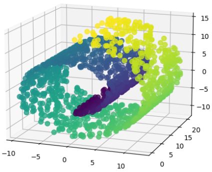
$$\text{Original space} \rightarrow p_{i,j} \approx \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}$$

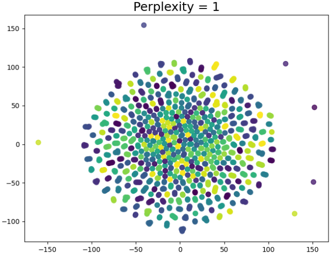
$$\text{Low-dim space } \rightarrow q_{i,j} \approx \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l\neq k}(1+\|y_l - y_k\|^2)^{-1}}$$

- Minimize via gradient descent wrt y: / Minimisons pour y avec gradient descent:

$$KL(p||q(y)) = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}(y)}$$

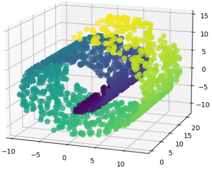
Unroll the swiss roll - TSNE

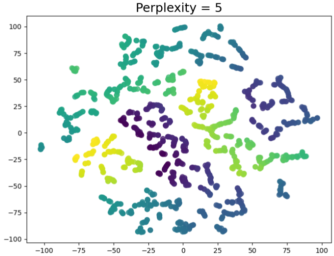




- Perplexity ((σ_i) related to number of neighbors) changes results a lot! / Perplexité (lié à la taille du voisinage) affecte les résultats beaucoup!

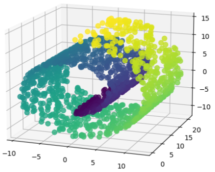
Unroll the swiss roll - TSNE

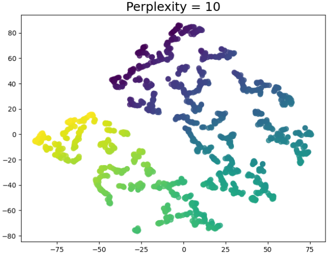




- Perplexity ((σ_i) related to number of neighbors) changes results a lot! / Perplexité (lié à la taille du voisinage) affecte les résultats beaucoup!

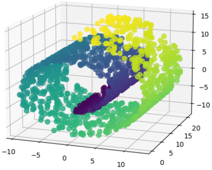
Unroll the swiss roll - TSNE

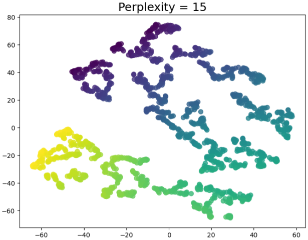




- Perplexity ((σ_i) related to number of neighbors) changes results a lot! / Perplexité (lié à la taille du voisinage) affecte les résultats beaucoup!

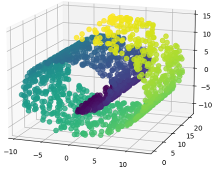
Unroll the swiss roll - TSNE

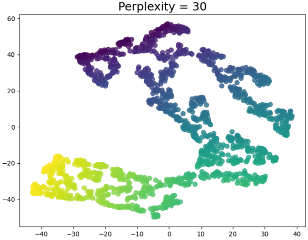




- Perplexity ((σ_i) related to number of neighbors) changes results a lot! / Perplexité (lié à la taille du voisinage) affecte les résultats beaucoup!

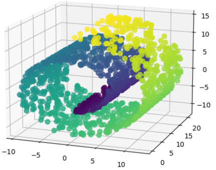
Unroll the swiss roll - TSNE

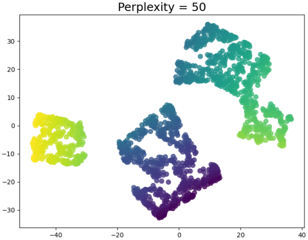




- Perplexity ((σ_i) related to number of neighbors) changes results a lot! / Perplexité (lié à la taille du voisinage) affecte les résultats beaucoup!

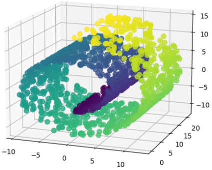
Unroll the swiss roll - TSNE

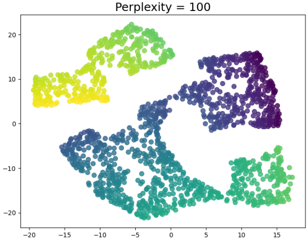




- Perplexity ((σ_i) related to number of neighbors) changes results a lot! / Perplexité (lié à la taille du voisinage) affecte les résultats beaucoup!

Unroll the swiss roll - TSNE





- Perplexity ((σ_i) related to number of neighbors) changes results a lot! / Perplexité (lié à la taille du voisinage) affecte les résultats beaucoup!

Table of Contents

Locally Linear Embedding

- Consider local neighboorhood of each point / Considérons le voisinage locale des points
  - Assume each neihborhood is linear / Supposons que le chaque voisinage est linéaire

$$x_i \approx \sum_{j \in \mathcal{N}(i)} w_{ij} x_j$$

- Then we solve the following problem / Puis on résouds la problème suivante:

$$\min_W \sum_i \|x_i - \sum_{j \in \mathcal{N}(i)} w_{ij} x_j\|^2$$
$$s.t. \sum_j w_{ij} = 1$$

- It's a convex problem, and can be solved easily.

Locally Linear Embedding - Finding y

- The weights will work just as well / Les poids qu'on avait trouvé fonctionnera dans faible dimensionnalité aussi
- We fix w_{ij}, and then solve, / On fixe w_{ij} et pis résouds,

$$\min_y \sum_i \|y_i - \sum_{j \in \mathcal{N}(i)} w_{ij} y_j\|^2$$

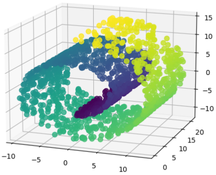
Locally Linear Embedding - Finding y

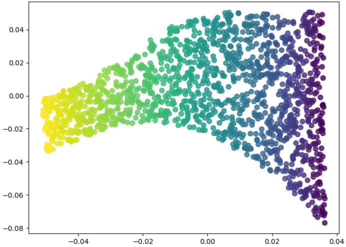
- The weights will work just as well / Les poids qu'on avait trouvé fonctionnera dans faible dimensionnalité aussi
- We fix w_{ij}, and then solve, / On fixe w_{ij} et pis résouds,

$$\min_y \sum_i \|y_i - \sum_{j \in \mathcal{N}(i)} w_{ij} y_j\|^2$$

- And (not surprisingly) , the solution is an eigendecomposition problem. / La solution est encore une fois décomposition des valeurs propres.
- We eigendecompose $(I - W)^\top (I - W)$, and take the eigenvectors that correspond to the smallest eigenvalues.... / On decompositionne $(I - W)^\top (I - W)$, pis prends les vecteurs propres qui correspond aux valeurs propres plus petits.

Unroll the swiss roll - LLE

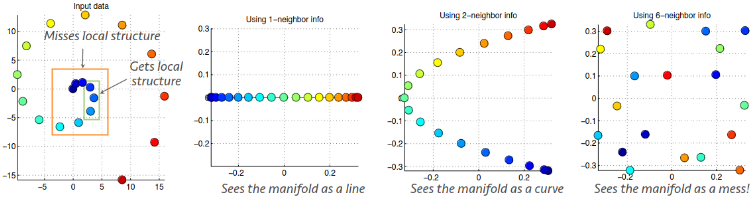




- Not that bad! / Pas si mauvais!

A note on the manifold methods

- Modeling the local structure is important! If you use too large neighborhoods, you will lose the structure.
  - C'est important d'etre capable à modéliser la structure locale. Si on utilise des voisinages qui sont trop larges, on perdra la structure.



Taken from UIUC MLSP class

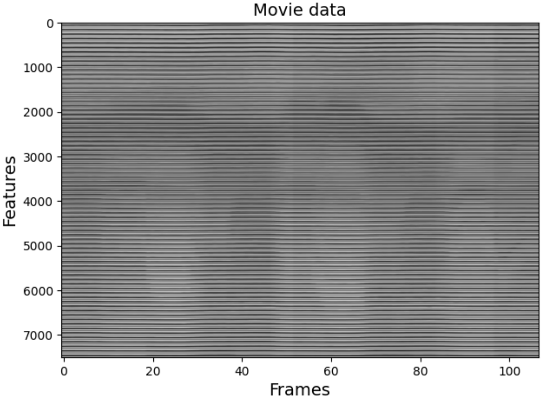
A video example

- High dimensional input $100 \times 75 = 7500$ dimensions / Une entrée à haute dimensions.
- Low dimensional structure, Moving lips around
  - Structure de petite dimensionnalité. Je bouge les levres.
- Can we simplify this? Peut-on simplifier cela?

`Watch`

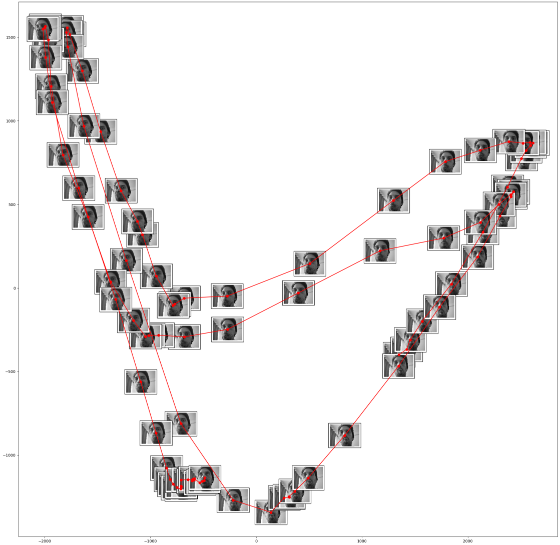
Frames

Unraveled

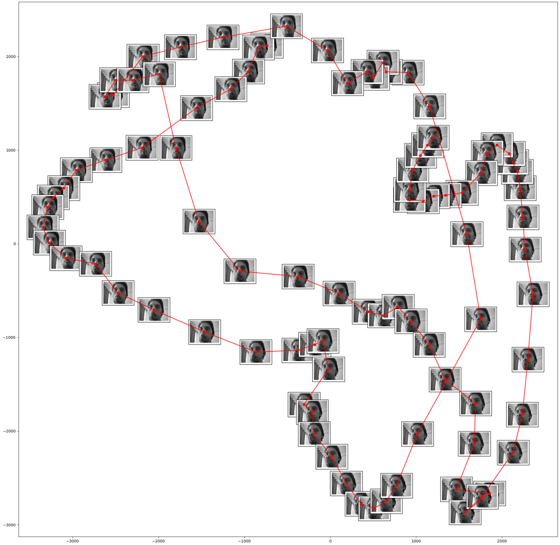


Movie data
Features
Frames
0
1000
2000
3000
4000
5000
6000
7000
0
20
40
60
80
100

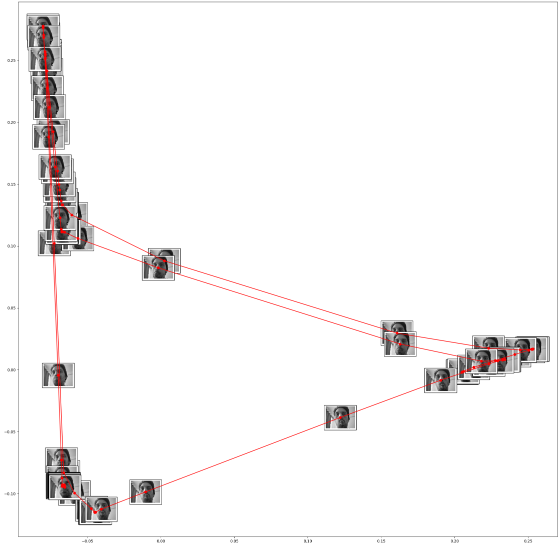
PCA



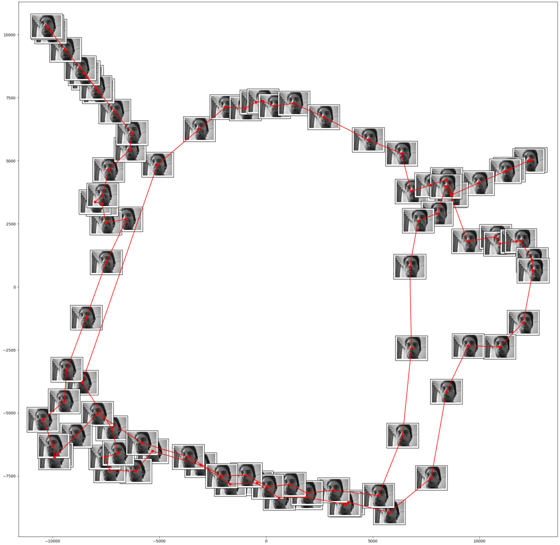
MDS



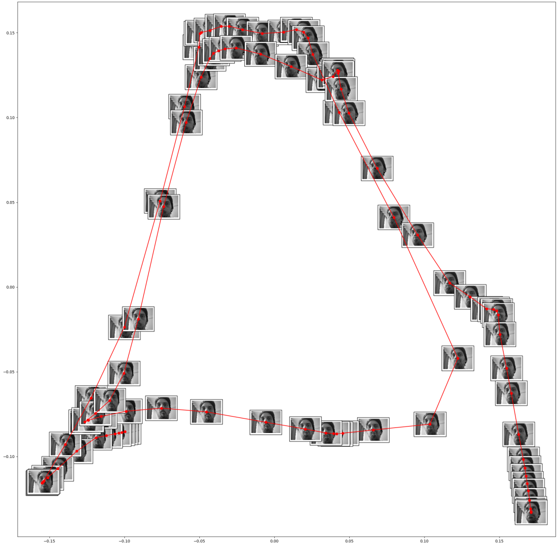
Laplacian Eigenmaps



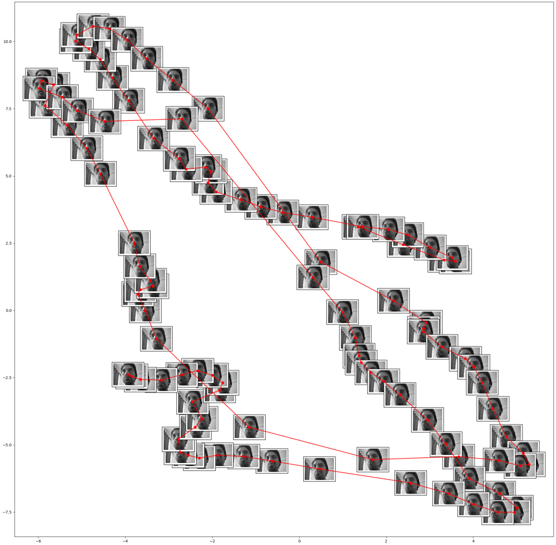
ISOMAP



LLE



TSNE

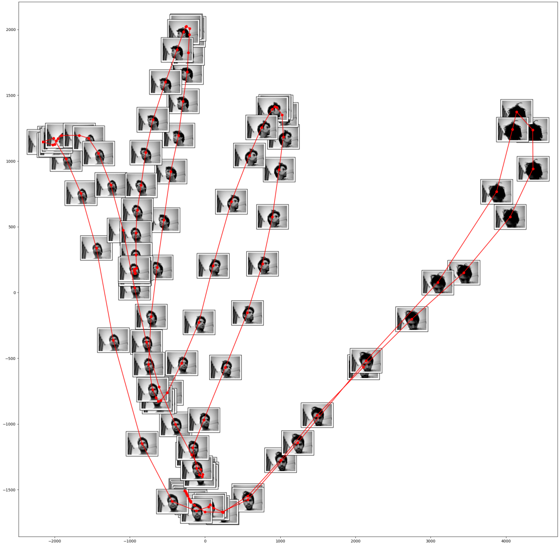


Another example

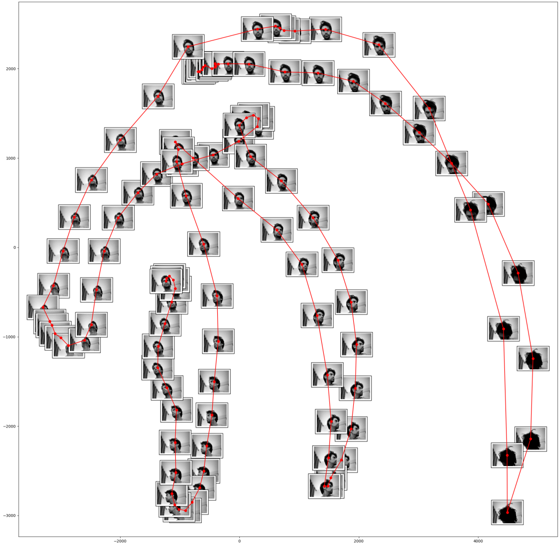
- This time we have distinct axes, left,right,up,down head movements
  - Cette fois-ci on a des mouvements distinctes, gauche, droite, haut, bas

Watch

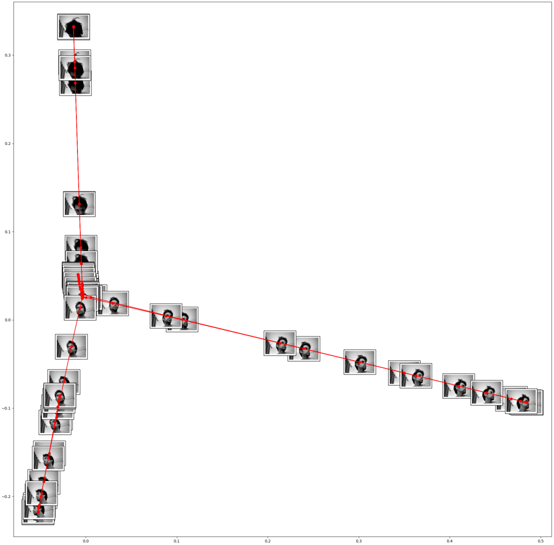
PCA


2000
1500
1000
500
0
−500
−1000
−1500
−2000
−1000
0
1000
2000
3000
4000

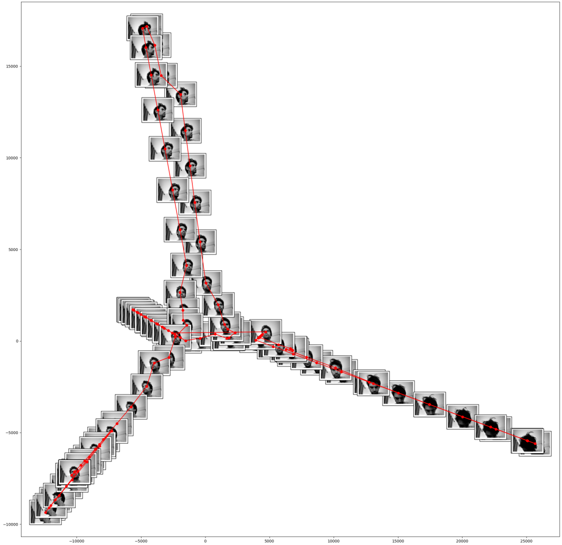
MDS



Laplacian Eigenmaps

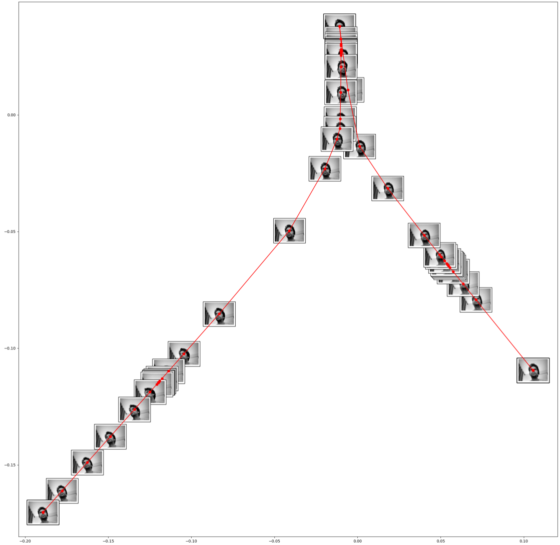


ISOMAP

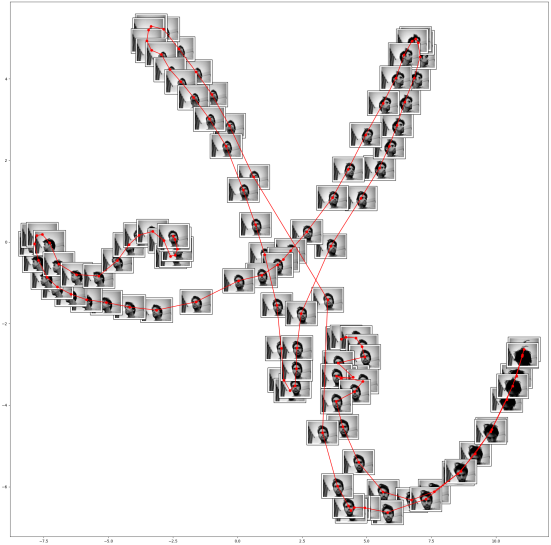


15000
10000
5000
0
−5000
−10000
−10000
−5000
0
5000
10000
15000
20000
25000

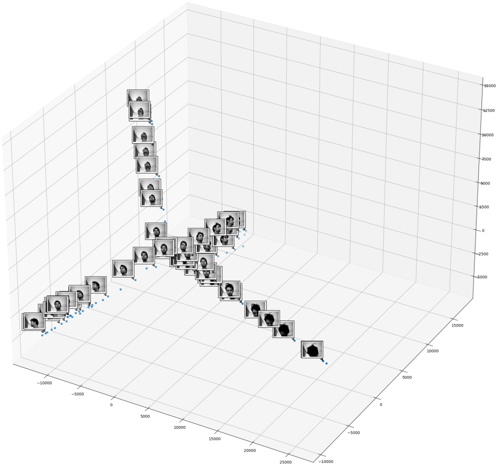
LLE



TSNE



ISOMAP



Even better! – learns 4 axes. / on apprend 4 axes

So, which method do we use?

- It depends. Pays off to experiment with different hyperparameters.
    - Ça depends. C'est bien d'experimenter avec différents approches.

Recap

- Kernel PCA: Map to another space where things are more appropriate for PCA. / On map à une autre espace plus approprié pour PCA.
- MDS: A transform which presevers pairwise distances.
- Manifolds
  - ISOMAP
  - Laplacian Eigenmaps
    - Locally Linear Embeddings

Suggested Reading

- LLE: https://cs.nyu.edu/~roweis/lle/papers/lleintro.pdf
- Kernel PCA: Bishop Chapter 12, https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=476974F6AA53BD038615E67656102714?doi=10.1.1.128.7613&rep=rep1&type=pdf
- Laplacian Eigenmaps: https://papers.nips.cc/paper/2001/file/f106b7f99d2cb30c3db1c3cc0fde9ccb-Paper.pdf

Next class

- Classification / Deep Learning (this time for real)