

# Reducción de la dimensionalidad

## Una introducción usando Python

Miguel González Duque - 26/07/22



# About me

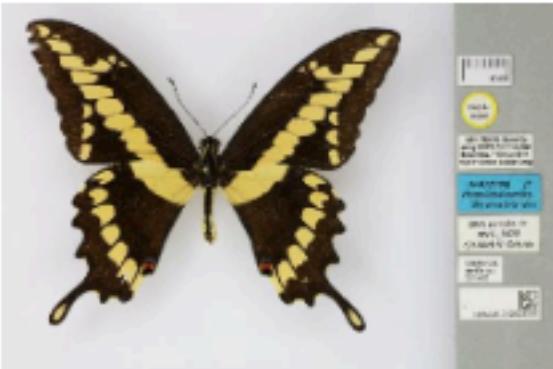


Pregrado y maestría en matemáticas de  
la Universidad Nacional de Colombia

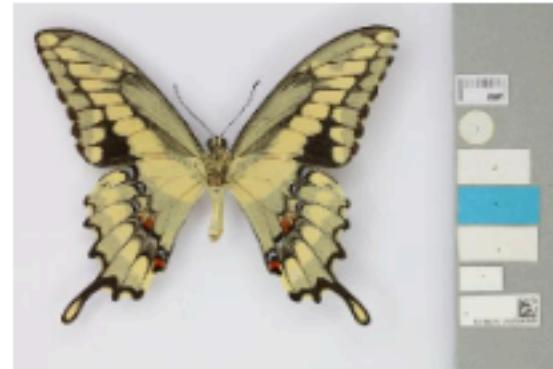
Estudiante de doctorado en la  
universidad IT de Copenhague

# Motivación

# Motivación



012824346\_Heraclides\_rumiko\_S...



012824346\_Heraclides\_rumiko\_S...



Anthene collinsi d' Abrera, 1980  
013376586\_Anthene\_collinsi\_D'A...



Anthene collinsi d' Abrera, 1980  
013376586\_Anthene\_collinsi\_D'A...



Aloeides thyra orientis Pringle, 1994  
013377027\_Aloeides\_thyra\_orienti...



Aloeides thyra orientis Pringle, 1994  
013377027\_Aloeides\_thyra\_orienti...



Aloeides monticola Pringle, 1994  
013377024\_Aloeides\_monticola\_...



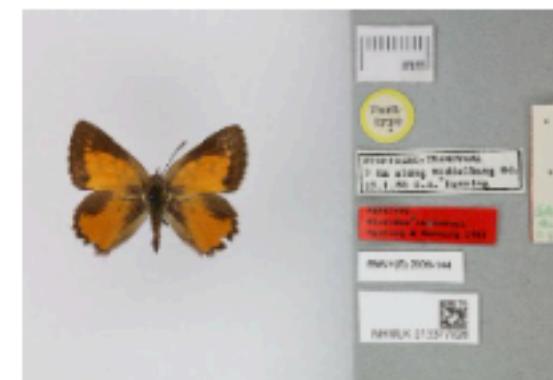
Aloeides monticola Pringle, 1994  
013377024\_Aloeides\_monticola\_...



Aloeides pallida juno Pringle, 1994  
013377025\_Aloeides\_pallida\_juno...



Aloeides pallida juno Pringle, 1994  
013377025\_Aloeides\_pallida\_juno...



Aloeides rossouwi Henning & Hen...  
013377026\_Aloeides\_rossouwi\_H...



Aloeides rossouwi Henning & Hen...  
013377026\_Aloeides\_rossouwi\_H...

NHM's catalogue of butterflies

¿Cómo visualizamos bases de datos enormes?

# Motivación



Cientos de imágenes, cada una representada por miles de números

(Canales RGB para cada pixel)

# Motivación



**MARIAN**

Visualizing 150000 butterflies from the Natural History Museum

December 26, 2019



[Click here for the interactive visualization.](#)

Tomada de: <https://marijan42.de/article/butterflies/>

Visualización: una de las motivaciones de la R.D.

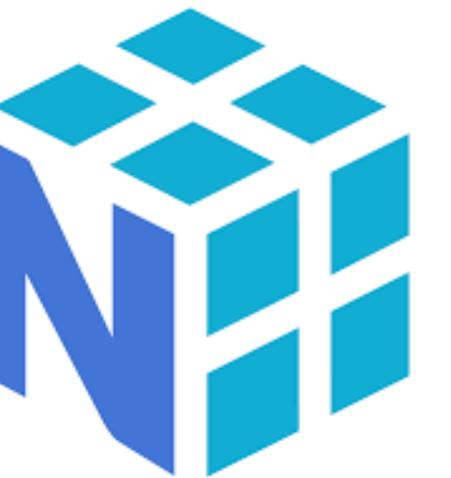
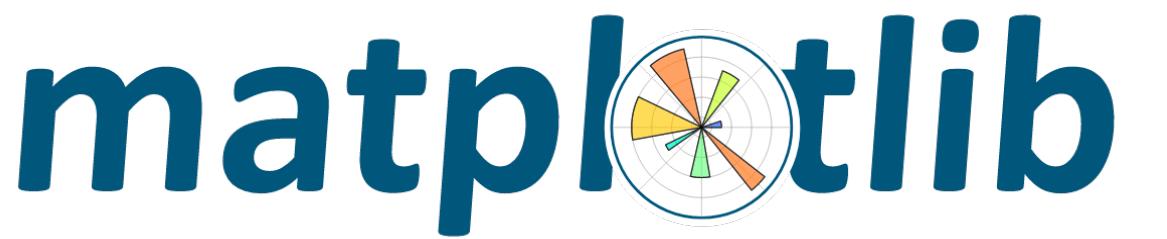
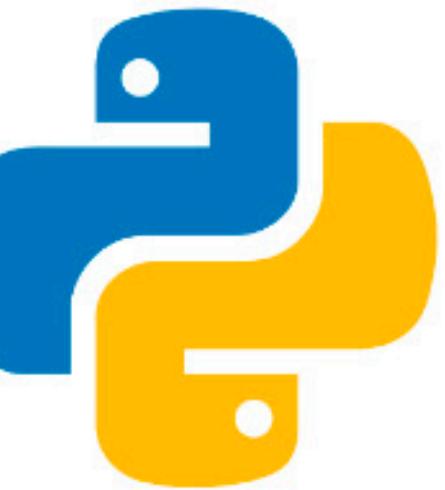
# Sobre hoy

¿Qué es?, ¿Para qué?

Algoritmos de reducción de la dimensionalidad (énfasis: PCA)

Un ejemplo en Python.

Tiempo: 30min/45min



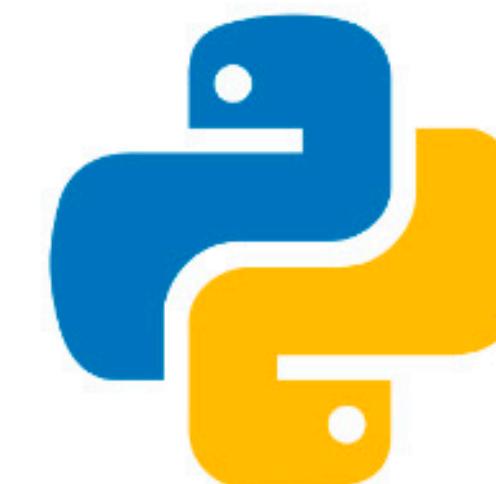
**¿Qué es la reducción de la dimensionalidad?**

# ¿Qué es?



¿Cuántos números necesitamos para [representar](#) esta imagen?

¿Necesitamos toda la información contenida en los pixeles?



# ¿Qué es?



¡No! Deberíamos poder describir con **menos números**.

→ [ ancho de las alas  
color primario en las alas  
:  
patrón de colores ]

# ¿Qué es?/¿Para qué?

Los algoritmos de R.D. comprimen los datos, o extraen variables importantes...

# ¿Qué es?/¿Para qué?

Los algoritmos de R.D. comprimen los datos, o extraen variables importantes...

Para visualizar

Para comprimir espacio

Para encontrar patrones y agrupar

# ¿Qué es?

Los algoritmos de R.D. comprimen los datos, o **extraen variables importantes**...

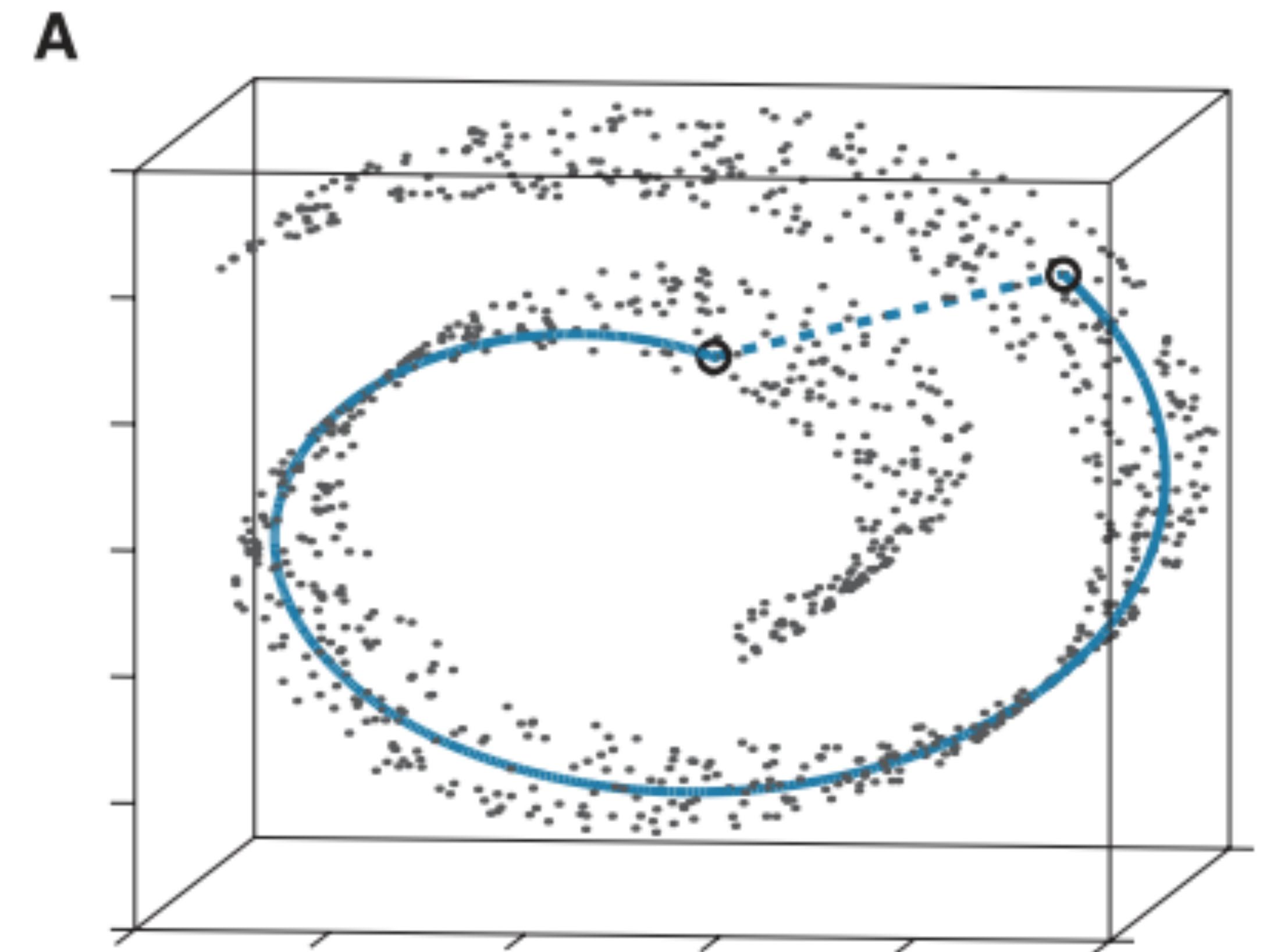


→ [ ancho de las alas  
color primario en las alas  
:  
patrón de colores ]

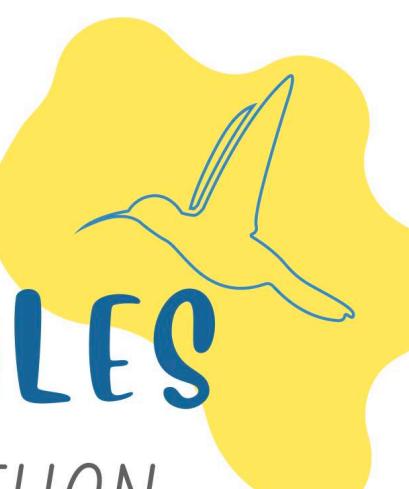
# ¿Qué es?

Otra perspectiva:

**La hipótesis de la superficie**  
(los datos viven cerca a una  
superficie bajo-dimensional)



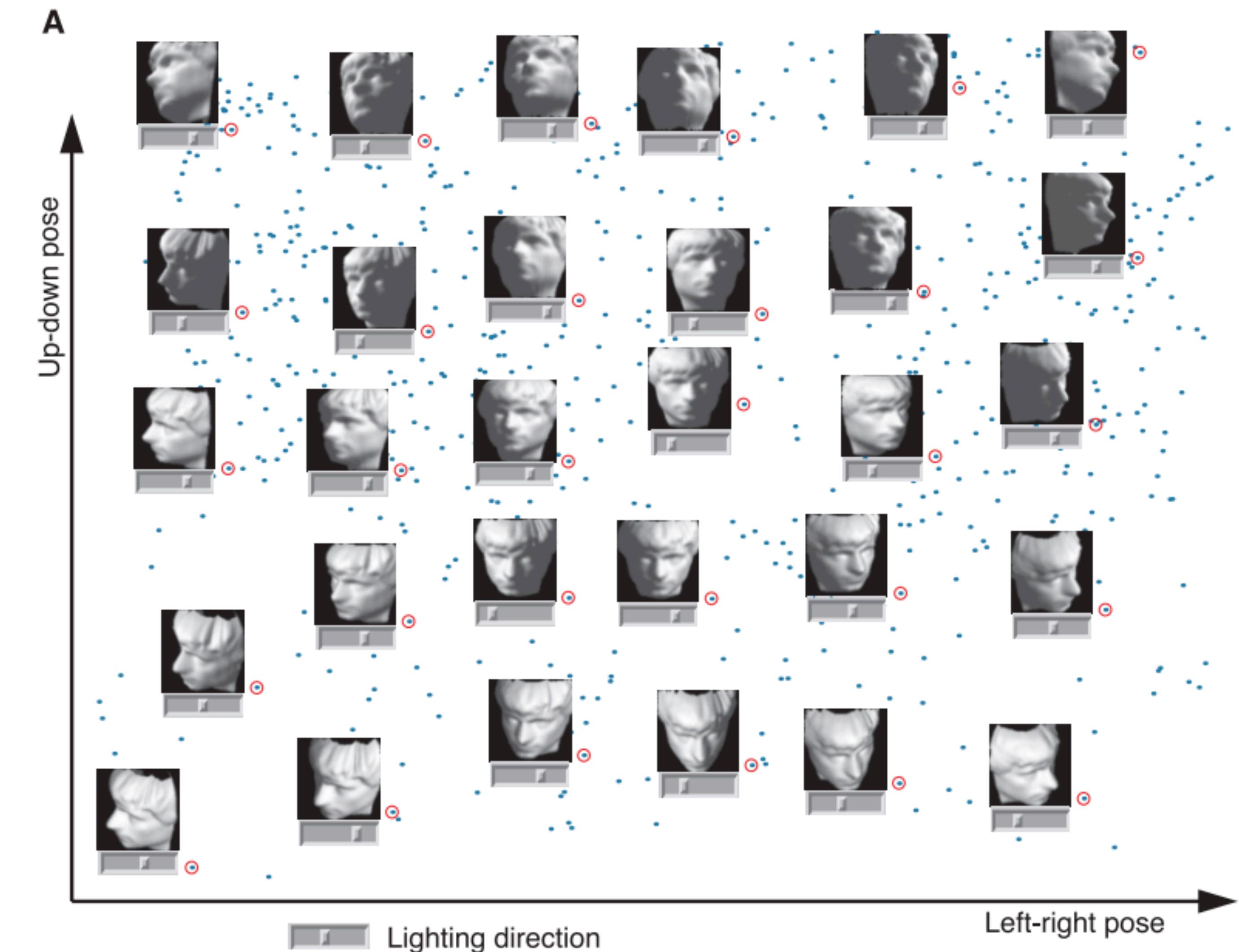
Tomada de: artículo de isomap



# ¿Qué es?

Tomada de: artículo de isomap

Otra perspectiva:  
Encontrar **variables latentes**.



**¿Cómo hacer reducción de la dimensionalidad?**

¿Cómo hacer reducción de la dimensionalidad?

¡Hay muchas formas!

# Hay muchos algoritmos

Escalamiento  
multidimensional

Análisis de  
componentes  
independientes

Factorización  
positiva de  
matrices

Autocodificador

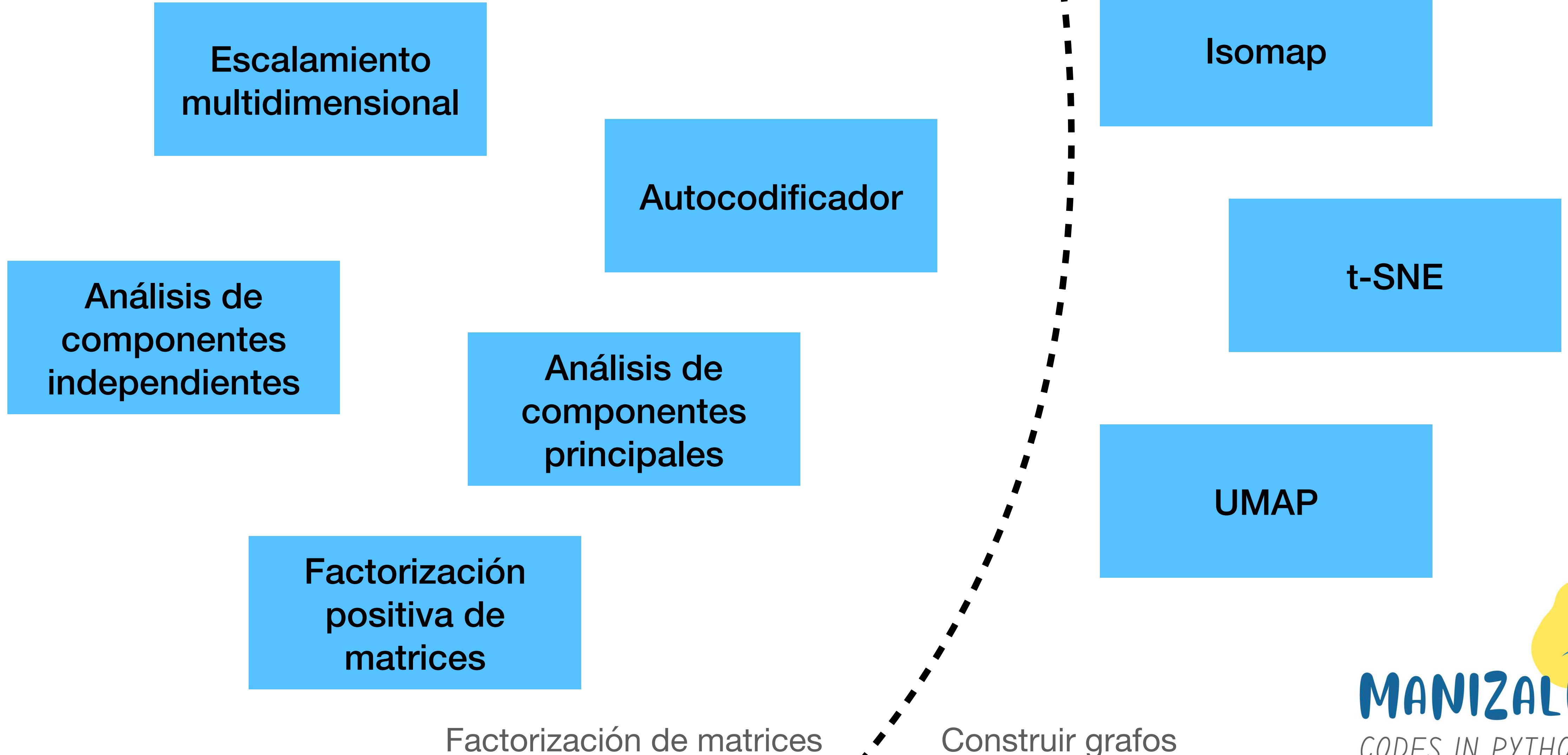
Análisis de  
componentes  
principales

Isomap

t-SNE

UMAP

# Hay dos familias



# Hoy nos enfocaremos en...

Escalamiento  
multidimensional

Análisis de  
componentes  
independientes

Factorización  
positiva de  
matrices

Autocodificador

Análisis de  
componentes  
principales

Factorización de matrices

Isomap

t-SNE

UMAP

Construir grafos

# Hoy nos enfocaremos en...

Disponible en



Escalamiento  
multidimensional

Análisis de  
componentes  
independientes

Factorización  
positiva de  
matrices

Autocodificadores

Análisis de  
componentes  
principales

Factorización de matrices

Isomap

t-SNE

UMAP

Construir grafos

**MANIZALES**  
CODES IN PYTHON



# ¿Cuál deberíamos usar?

Disponible en



Escalamiento  
multidimensional

Análisis de  
componentes  
independientes

Factorización  
positiva de  
matrices

Análisis de  
componentes  
principales

Autocodificadores

Factorización de matrices

Construir grafos

Isomap

t-SNE

UMAP

MANIZALES  
CODES IN PYTHON



# ¿Cuál deberíamos usar?

Disponible en



Escalamiento  
multidimensional

Isomap

Análisis de  
componentes  
independientes

componentes  
principales

Factorización  
positiva de  
matrices

UMAP

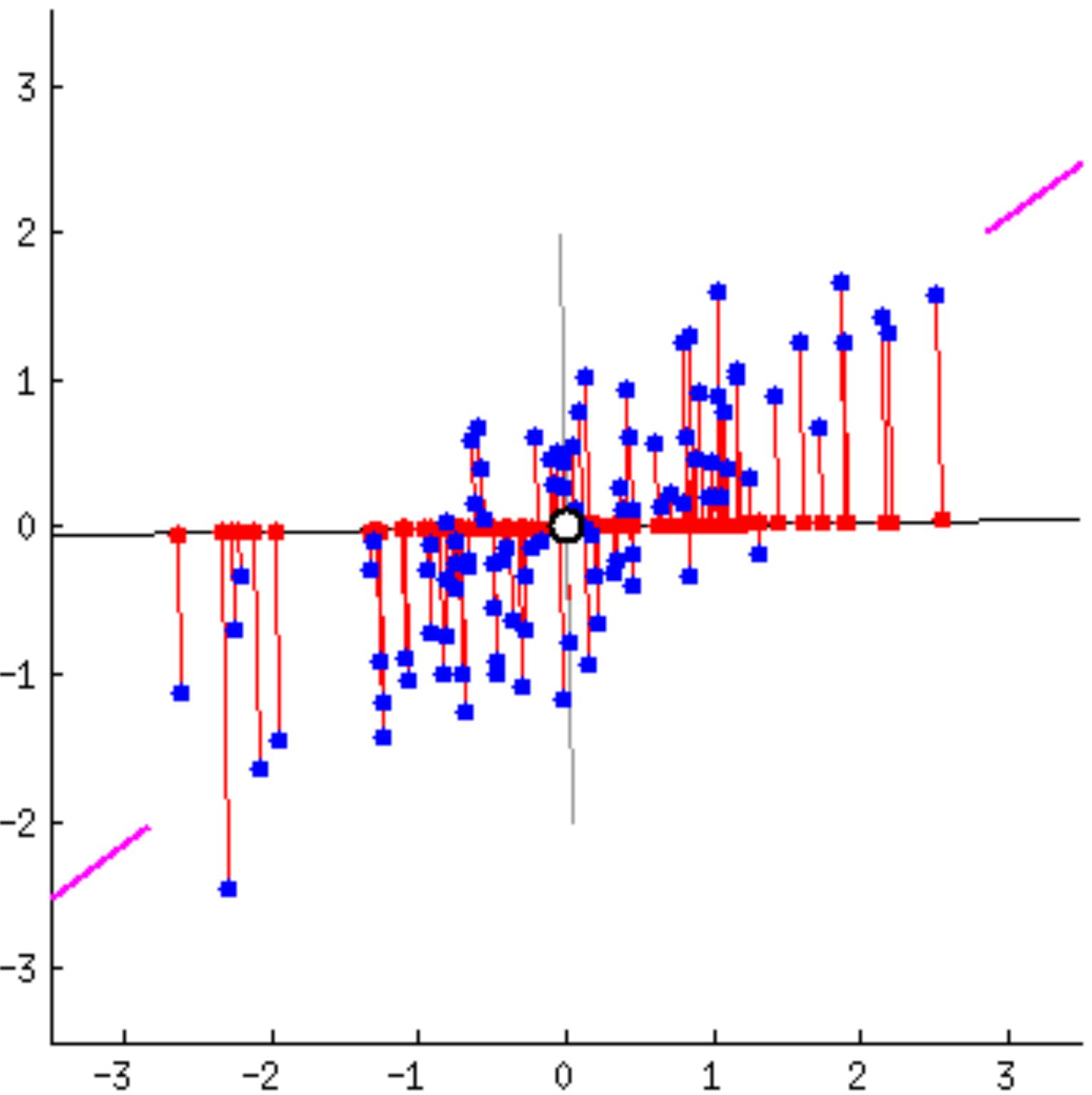
Factorización de matrices

Construir grafos

MANIZALES  
CODES IN PYTHON



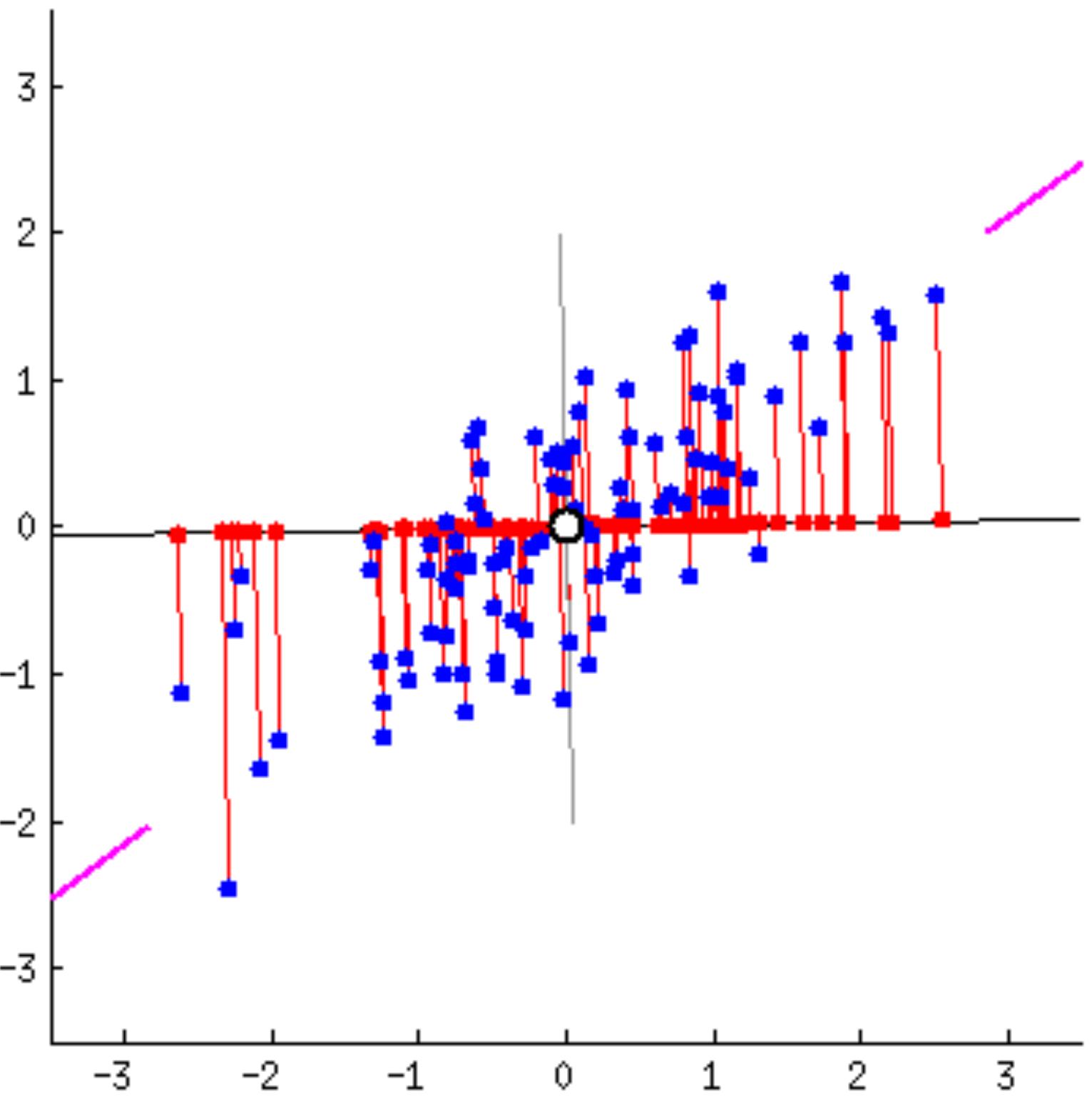
# Análisis de componentes principales (PCA)



La idea intuitiva: minimizar el error al proyectar

Gif tomado de:

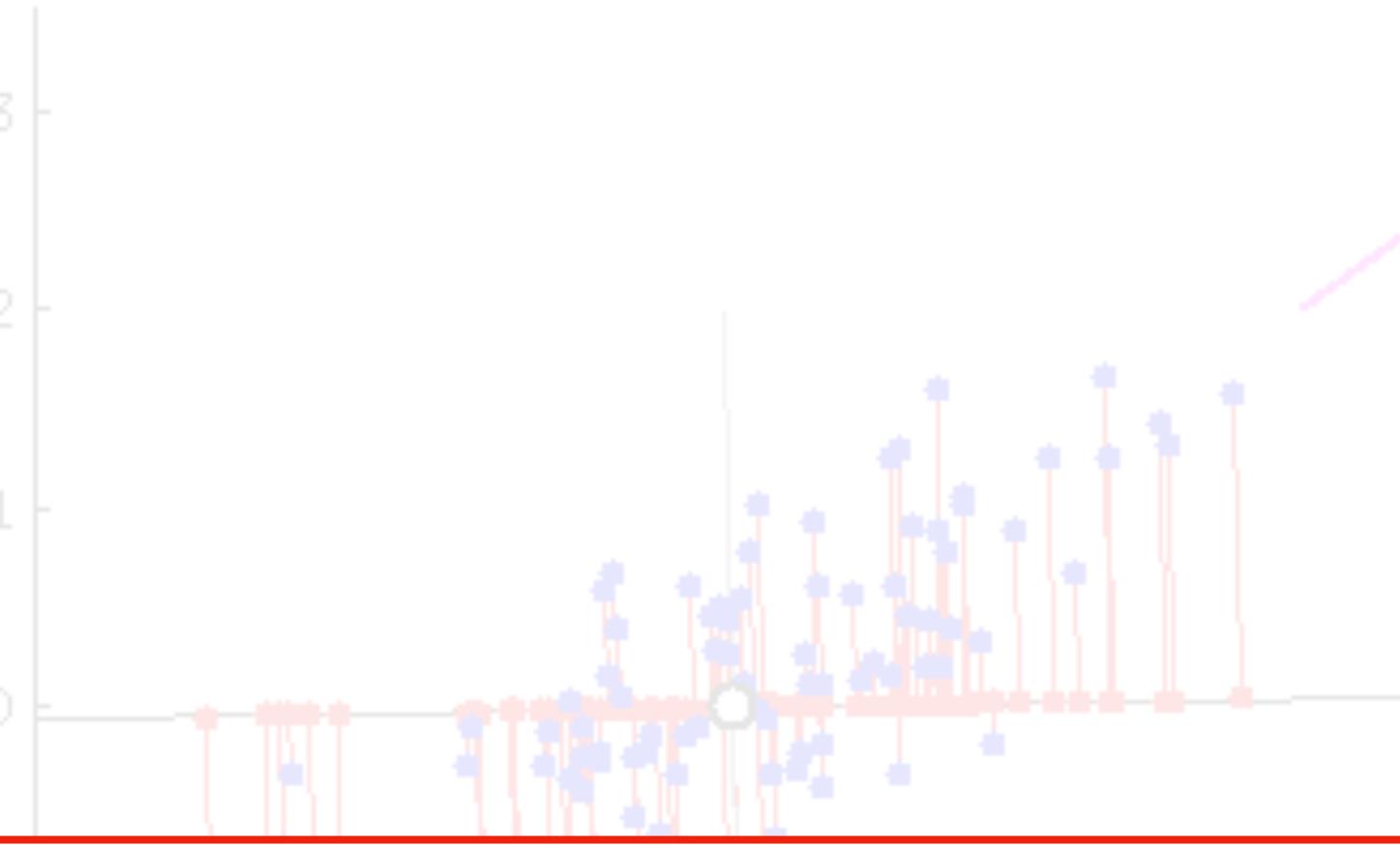
<https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues>



Equivalentemente: maximizar la varianza de los datos proyectados

Gif tomado de:

<https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues>



Si estás interesada/o en las matemáticas del asunto, ¡muy recomendadas las referencias en la descripción!

Equivalentemente: maximizar la varianza de los datos proyectados

Gif tomado de:

<https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues>

```
from sklearn.decompose import PCA

from utils import load_dataset

def fit_PCA_to_data():
    # We load the numpy arrays
    data = load_dataset()

    # We create a PCA object from scikit-learn
    pca = PCA(n_components=2)

    # We fit it to our data and get latent codes.
    latent_codes = pca.fit_transform(data)

    return latent_codes
```



nos permite usar  
PCA fácil

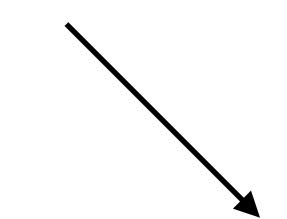
Imagen creada usando:

<https://carbon.now.sh/>

```
from sklearn.decompose import PCA  
  
from utils import load_dataset  
  
def fit_PCA_to_data():  
    # We load the numpy arrays  
    data = load_dataset()  
  
    # We create a PCA object from scikit-learn  
    pca = PCA(n_components=2)  
  
    # We fit it to our data and get latent codes.  
    latent_codes = pca.fit_transform(data)  
  
    return latent_codes
```



nos permite usar  
~~PCA~~ fácil



t-SNE, Isomap, MDE...

Imagen creada usando:

<https://carbon.now.sh/>

## PCA

```
from sklearn.decompose import PCA  
  
from utils import load_dataset  
  
def fit_PCA_to_data():  
    # We load the numpy arrays  
    data = load_dataset()  
  
    # We create a PCA object from scikit-learn  
    pca = PCA(n_components=2)  
  
    # We fit it to our data and get latent codes.  
    latent_codes = pca.fit_transform(data)  
  
    return latent_codes
```

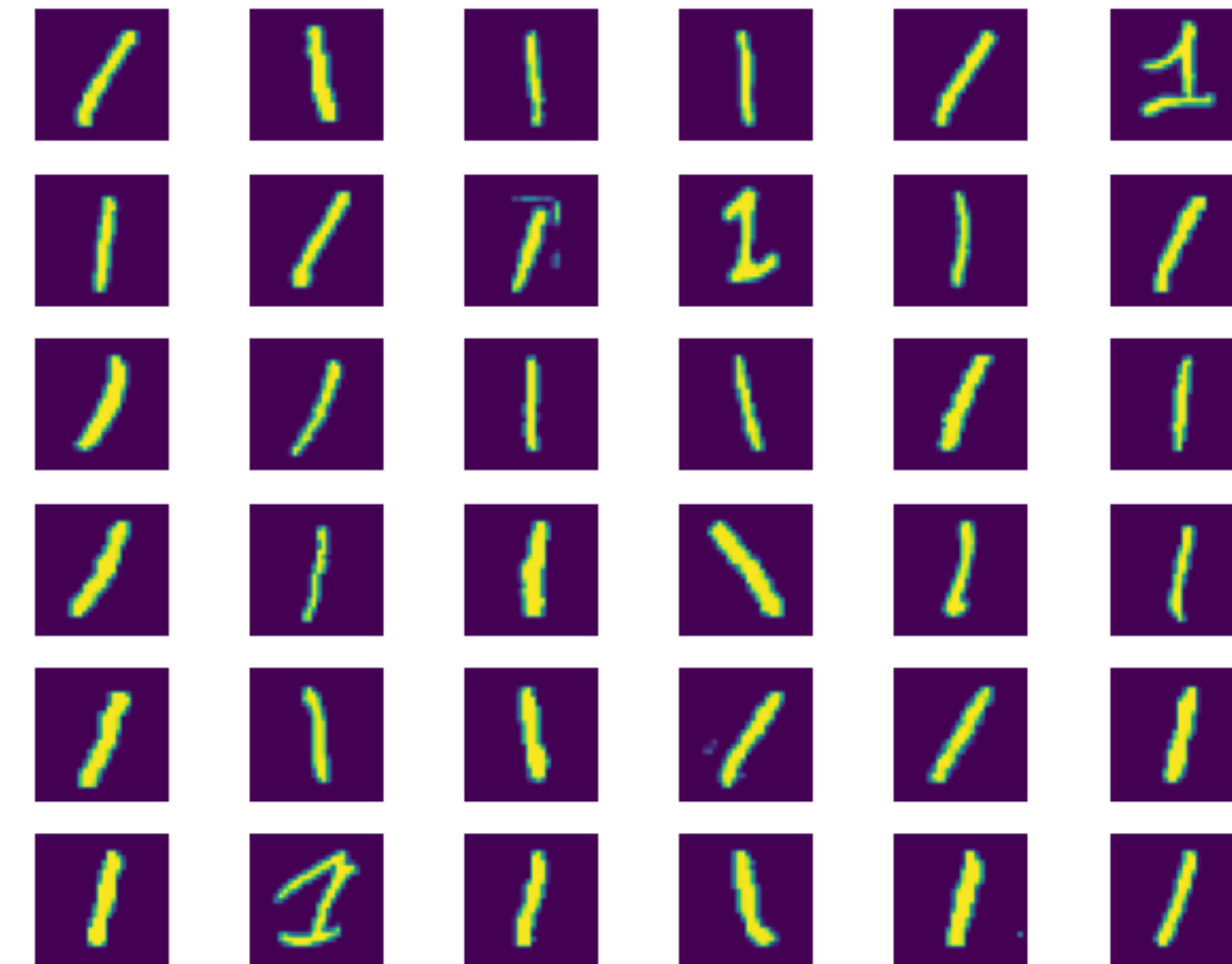
## t-SNE

```
from sklearn.manifold import TSNE  
  
from utils import load_dataset  
  
def fit_tSNE_to_data():  
    # We load the numpy arrays  
    data = load_dataset()  
  
    # We create a t-SNE object from scikit-learn  
    tsne = TSNE(n_components=2)  
  
    # We fit it to our data and get latent codes.  
    latent_codes = tsne.fit_transform(data)  
  
    return latent_codes
```

...y la interfaz está bien diseñada

Imágenes creadas usando:

<https://carbon.now.sh/>



Un ejemplo en Python