Візуалізація даних

**Лабораторна робота №10**

# Зниження розмірності

Для зниження розмірності використовуйте бібліотеку scikit-learn <https://scikit-learn.org/stable/modules/manifold.html>

Для реалізації завдання використовуйте наступний приклад

<https://scikit-learn.org/stable/auto_examples/manifold/plot_lle_digits.html#sphx-glr-auto-examples-manifold-plot-lle-digits-py>

# Authors: Fabian Pedregosa <fabian.pedregosa@inria.fr>

# Olivier Grisel <olivier.grisel@ensta.org>

# Mathieu Blondel <mathieu@mblondel.org>

# Gael Varoquaux

# License: BSD 3 clause (C) INRIA 2011

**print(**\_\_doc\_\_**)**

**from** time **import** time

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

**from** matplotlib **import** offsetbox

**from** sklearn **import** **(**manifold**,** datasets**,** decomposition**,** ensemble**,** discriminant\_analysis**,** random\_projection**)**

digits **=** datasets**.**load\_digits**(**n\_class**=**6**)**

X **=** digits**.**data

y **=** digits**.**target

n\_samples**,** n\_features **=** X**.**shape

n\_neighbors **=** 30

#---------------------------------------------------------------

# Scale and visualize the embedding vectors

**def** plot\_embedding**(**X**,** title**=None):**

x\_min**,** x\_max **=** np**.**min**(**X**,** 0**),** np**.**max**(**X**,** 0**)**

X **=** **(**X **-** x\_min**)** **/** **(**x\_max **-** x\_min**)**

plt**.**figure**()**

ax **=** plt**.**subplot**(**111**)**

**for** i **in** range**(**X**.**shape**[**0**]):**

plt**.**text**(**X**[**i**,** 0**],** X**[**i**,** 1**],** str**(**y**[**i**]),**

color**=**plt**.**cm**.**Set1**(**y**[**i**]** **/** 10.**),**

fontdict**={**'weight'**:** 'bold'**,** 'size'**:** 9**})**

**if** hasattr**(**offsetbox**,** 'AnnotationBbox'**):**

# only print thumbnails with matplotlib > 1.0

shown\_images **=** np**.**array**([[**1.**,** 1.**]])** # just something big

**for** i **in** range**(**X**.**shape**[**0**]):**

dist **=** np**.**sum**((**X**[**i**]** **-** shown\_images**)** **\*\*** 2**,** 1**)**

**if** np**.**min**(**dist**)** **<** 4e-3**:**

# don't show points that are too close

**continue**

shown\_images **=** np**.**r\_**[**shown\_images**,** **[**X**[**i**]]]**

imagebox **=** offsetbox**.**AnnotationBbox**(**

offsetbox**.**OffsetImage**(**digits**.**images**[**i**],** cmap**=**plt**.**cm**.**gray\_r**),**

X**[**i**])**

ax**.**add\_artist**(**imagebox**)**

plt**.**xticks**([]),** plt**.**yticks**([])**

**if** title **is** **not** **None:**

plt**.**title**(**title**)**

#----------------------------------------------------------------------

# Plot images of the digits

n\_img\_per\_row **=** 20

img **=** np**.**zeros**((**10 **\*** n\_img\_per\_row**,** 10 **\*** n\_img\_per\_row**))**

**for** i **in** range**(**n\_img\_per\_row**):**

ix **=** 10 **\*** i **+** 1

**for** j **in** range**(**n\_img\_per\_row**):**

iy **=** 10 **\*** j **+** 1

img**[**ix**:**ix **+** 8**,** iy**:**iy **+** 8**]** **=** X**[**i **\*** n\_img\_per\_row **+** j**].**reshape**((**8**,** 8**))**

plt**.**imshow**(**img**,** cmap**=**plt**.**cm**.**binary**)**

plt**.**xticks**([])**

plt**.**yticks**([])**

plt**.**title**(**'A selection from the 64-dimensional digits dataset'**)**

#---------------------------------------------------------------

# Random 2D projection using a random unitary matrix

**print(**"Computing random projection"**)**

rp **=** random\_projection**.**SparseRandomProjection**(**n\_components**=**2**,** random\_state**=**42**)**

X\_projected **=** rp**.**fit\_transform**(**X**)**

plot\_embedding**(**X\_projected**,** "Random Projection of the digits"**)**

#---------------------------------------------------------------

# Projection on to the first 2 principal components

**print(**"Computing PCA projection"**)**

t0 **=** time**()**

X\_pca **=** decomposition**.**TruncatedSVD**(**n\_components**=**2**).**fit\_transform**(**X**)**

plot\_embedding**(**X\_pca**,**

"Principal Components projection of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#----------------------------------------------------------------------

# Projection on to the first 2 linear discriminant components

**print(**"Computing Linear Discriminant Analysis projection"**)**

X2 **=** X**.**copy**()**

X2**.**flat**[::**X**.**shape**[**1**]** **+** 1**]** **+=** 0.01 # Make X invertible

t0 **=** time**()**

X\_lda **=** discriminant\_analysis**.**LinearDiscriminantAnalysis**(**n\_components**=**2**).**fit\_transform**(**X2**,** y**)**

plot\_embedding**(**X\_lda**,**

"Linear Discriminant projection of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#---------------------------------------------------------------

# Isomap projection of the digits dataset

**print(**"Computing Isomap embedding"**)**

t0 **=** time**()**

X\_iso **=** manifold**.**Isomap**(**n\_neighbors**,** n\_components**=**2**).**fit\_transform**(**X**)**

**print(**"Done."**)**

plot\_embedding**(**X\_iso**,**

"Isomap projection of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#---------------------------------------------------------------

# Locally linear embedding of the digits dataset

**print(**"Computing LLE embedding"**)**

clf **=** manifold**.**LocallyLinearEmbedding**(**n\_neighbors**,** n\_components**=**2**,**

method**=**'standard'**)**

t0 **=** time**()**

X\_lle **=** clf**.**fit\_transform**(**X**)**

**print(**"Done. Reconstruction error: %g" **%** clf**.**reconstruction\_error\_**)**

plot\_embedding**(**X\_lle**,**

"Locally Linear Embedding of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#---------------------------------------------------------------

# Modified Locally linear embedding of the digits dataset

**print(**"Computing modified LLE embedding"**)**

clf **=** manifold**.**LocallyLinearEmbedding**(**n\_neighbors**,** n\_components**=**2**,**

method**=**'modified'**)**

t0 **=** time**()**

X\_mlle **=** clf**.**fit\_transform**(**X**)**

**print(**"Done. Reconstruction error: %g" **%** clf**.**reconstruction\_error\_**)**

plot\_embedding**(**X\_mlle**,**

"Modified Locally Linear Embedding of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#---------------------------------------------------------------

# HLLE embedding of the digits dataset

**print(**"Computing Hessian LLE embedding"**)**

clf **=** manifold**.**LocallyLinearEmbedding**(**n\_neighbors**,** n\_components**=**2**,**

method**=**'hessian'**)**

t0 **=** time**()**

X\_hlle **=** clf**.**fit\_transform**(**X**)**

**print(**"Done. Reconstruction error: %g" **%** clf**.**reconstruction\_error\_**)**

plot\_embedding**(**X\_hlle**,**

"Hessian Locally Linear Embedding of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#---------------------------------------------------------------

# LTSA embedding of the digits dataset

**print(**"Computing LTSA embedding"**)**

clf **=** manifold**.**LocallyLinearEmbedding**(**n\_neighbors**,** n\_components**=**2**,**

method**=**'ltsa'**)**

t0 **=** time**()**

X\_ltsa **=** clf**.**fit\_transform**(**X**)**

**print(**"Done. Reconstruction error: %g" **%** clf**.**reconstruction\_error\_**)**

plot\_embedding**(**X\_ltsa**,**

"Local Tangent Space Alignment of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#---------------------------------------------------------------

# MDS embedding of the digits dataset

**print(**"Computing MDS embedding"**)**

clf **=** manifold**.**MDS**(**n\_components**=**2**,** n\_init**=**1**,** max\_iter**=**100**)**

t0 **=** time**()**

X\_mds **=** clf**.**fit\_transform**(**X**)**

**print(**"Done. Stress: %f" **%** clf**.**stress\_**)**

plot\_embedding**(**X\_mds**,**

"MDS embedding of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#---------------------------------------------------------------

# Random Trees embedding of the digits dataset

**print(**"Computing Totally Random Trees embedding"**)**

hasher **=** ensemble**.**RandomTreesEmbedding**(**n\_estimators**=**200**,** random\_state**=**0**,**

max\_depth**=**5**)**

t0 **=** time**()**

X\_transformed **=** hasher**.**fit\_transform**(**X**)**

pca **=** decomposition**.**TruncatedSVD**(**n\_components**=**2**)**

X\_reduced **=** pca**.**fit\_transform**(**X\_transformed**)**

plot\_embedding**(**X\_reduced**,**

"Random forest embedding of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#---------------------------------------------------------------

# Spectral embedding of the digits dataset

**print(**"Computing Spectral embedding"**)**

embedder **=** manifold**.**SpectralEmbedding**(**n\_components**=**2**,** random\_state**=**0**,**

eigen\_solver**=**"arpack"**)**

t0 **=** time**()**

X\_se **=** embedder**.**fit\_transform**(**X**)**

plot\_embedding**(**X\_se**,**

"Spectral embedding of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

#---------------------------------------------------------------

# t-SNE embedding of the digits dataset

**print(**"Computing t-SNE embedding"**)**

tsne **=** manifold**.**TSNE**(**n\_components**=**2**,** init**=**'pca'**,** random\_state**=**0**)**

t0 **=** time**()**

X\_tsne **=** tsne**.**fit\_transform**(**X**)**

plot\_embedding**(**X\_tsne**,**

"t-SNE embedding of the digits (time %.2fs)" **%**

**(**time**()** **-** t0**))**

plt**.**show**()**

# Варіанти завдань

Для відповідного датасету згідно з варіантом виконати пониження розмірності даних до просторів з розмірностями два та три (2D та 3D). Для пониження розмірності використовуйте всі доступні методи бібліотеки scikit-learn для зниження розмірності. Порівняйте результати та визначте яким методом було досягнуто найкращий результат.