Algorithme de Kohonen, Reconnaissance automatique de chiffres manuscrits

Louis Douge & William Beng

ENSTA ParisTech

2 juin 2015

Plan de la présentation

La bibliothèque numpy

Acquérir des connaissances

- documentation officielle
- moteur de recherche, forum, tests personnels (console)...

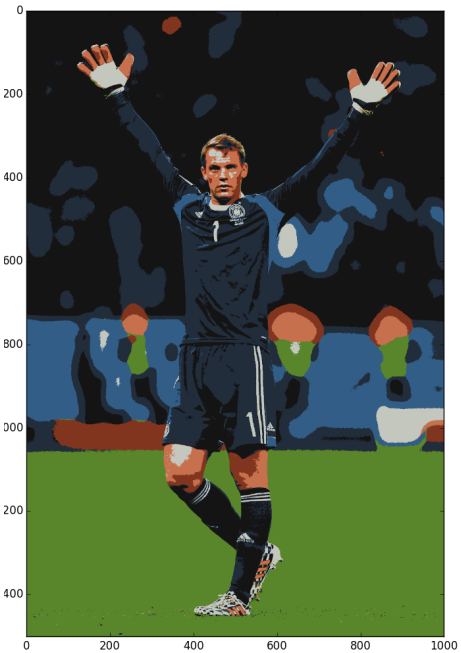
La bibliothèque numpy

Acquérir des connaissances

- documentation officielle
- moteur de recherche, forum, tests personnels (console)...

Pratiquer dès le début : exercice des K-moyennes

- manipulation de format similaire
- sorties graphiques



Avantages de numpy

- Manipulation matricielle

```
distances_matrix = numpy.sqrt(numpy.sum((
    vectors-input_vector)**2, axis=1))
```

- Fonctions usuelles

```
kernel = numpy.exp(-(X ** 2 + Y ** 2) / (2 *
    gaussian_sigma ** 2))
```

- Gaussienne échantillonnée

```
M, N = numpy.ogrid[0:space_shape[0], 0:
    space_shape[1]]
```

Adapter le calendrier des tâches

Calendrier initial à mettre à jour

- incertitudes de départ
- réutilisation des fichiers fournis à chaque séance

But

Créer un code capable de générer
des données en vue de son amélioration

Graphiquement : visualiser la carte entraînée et la carte labellisée.
Problème :

```
weights.plot = plt.figure('_')
ax_weights = weights_plot.add_subplot(111)
ax_weights.imshow(weights[image,:].reshape(
    data_shape[0]*map_shape[0],data_shape[1]*
    map_shape[1]), interpolation='nearest',
    cmap = plt.cm.bone)
```

```
# parcours de la COA en ajoutant un subplot
    pour chaque neurone
for image in range(map_shape[0]*map_shape[1]):
    ## cr ation d'un axe matplotlib
    ax_weights = weights_plot.add_subplot(
        map_shape[0],map_shape[1],image)
    ## chargement dans la figure du neurone
        n image comme matrice de pixels en
        niveau de gris
    ax_weights.imshow(weights[image,:].reshape(
        data_shape[0],data_shape[1]),
        interpolation='nearest', cmap = plt.cm.
        bone)
    ax_weights.axes.get_xaxis().set_visible(
        False)
    ax_weights.axes.get_yaxis().set_visible(
        False)
```

Carte labellisée

Terminal

Fichier Édition Affichage Rechercher Terminal Aide

```
douge@mufasa01:~/IN104/BENGDOUGE$ ls
COA_sur_MNIST.py   final_weights.npy  LAB_sur_MNIST.py   __pycache__
COA_sur_MNIST.py~  label_card.npy     LAB_sur_MNIST.py~  weights
DECISION.py        labelled_card.npy  parametres.py
DECISION.py~       labelled_map       parametres.py~
douge@mufasa01:~/IN104/BENGDOUGE$ python3 LAB_sur_MNIST.py
Traceback (most recent call last):
  File "LAB_sur_MNIST.py", line 13, in <module>
    import kohonen
ImportError: No module named 'kohonen'
douge@mufasa01:~/IN104/BENGDOUGE$ ls
COA_sur_MNIST.py   final_weights.npy  LAB_sur_MNIST.py   __pycache__
COA_sur_MNIST.py~  label_card.npy     LAB_sur_MNIST.py~  weights
DECISION.py        labelled_card.npy  parametres.py
DECISION.py~       labelled_map       parametres.py~
douge@mufasa01:~/IN104/BENGDOUGE$ cd ..
douge@mufasa01:~/IN104$ cd TESTS/
douge@mufasa01:~/IN104/TESTS$ python3 LAB_sur_MNIST.py
douge@mufasa01:~/IN104/TESTS$ python3 COA_sur_MNIST.py
douge@mufasa01:~/IN104/TESTS$ cd ..
douge@mufasa01:~/IN104$ cd TD5
douge@mufasa01:~/IN104/TD5$ cd Douge\'s\ work/
douge@mufasa01:~/IN104/TD5/Douge's work$ cd Douge\'s\ work\ 19mai1622
douge@mufasa01:~/IN104/TD5/Douge's work/Douge's work 19mai1622$ python3 LAB_sur_
MNIST.py
[[6 6 6 0 0 0 0 0 5 3]
 [6 6 6 2 0 0 0 0 5 3]
 [6 6 2 2 2 3 0 5 5 3]
 [4 2 2 2 2 3 3 3 3 3]
 [4 4 2 2 2 3 3 3 3 8]
 [9 4 8 8 8 8 5 5 8 1]
 [9 4 4 8 8 8 5 6 1 1]
 [9 4 4 4 8 5 5 5 1 1]
 [7 9 9 9 4 4 5 1 1 1]
 [7 7 7 7 7 7 1 1 1 1]]
douge@mufasa01:~/IN104/TD5/Douge's work/Douge's work 19mai1622$
```

numpy : load & save

But

Réutiliser des résultats intermédiaires pour limiter les calculs :

- carte entraînée
- carte labellisée

On génère des fichiers intermédiaires.

```
## sauvegarde de la derni re COA
numpy.save("weights/final_weights_%d"%(iterations),
    weights)
```

"Prototypes des fonctions" supervisées par le fichier script final.py :

```
COA_sur_MNIST.COA(data_path, iterations, nb_map,
    sigma_max_value, sigma_min_value, eta_max_value,
    eta_min_value, decay_start_iter,decay_stop_iter,
    affichage_graphique)
LAB_sur_MNIST.LAB(data_path, iterations, nb_map,
    LAB_all, LAB_nb)
error_rate = DECISION.DEC(data_path, iterations, nb_map
    , LAB_all, DEC_all, LAB_nb)
```

Principe de la photographie

But

Prélever un couple (COA, carte de label) à une certaine fréquence
=> calcul du taux d'erreur

```
if (curr_iter == sampling_iter):
    ## On sauve la COA ainsi obtenue
    numpy.save("weights/final_weights_%d"%(
        curr_iter),weights)
    sampling_iter += iterations/nb_map
```

Influence du nombre d'itération sur le taux d'erreur

On utilise toutes les cartes entraînées et leurs scores correspondant

```
#LAB_all=True si nb_map =! 1 et que l'on souhaite
    etudier toutes les cartes produites
LAB_all = True
#DEC_all=True si nb_map =! 1 et que l'on souhaite
    etudier toutes les cartes produites
DEC_all = True
# On souhaite visualiser la COA produite
affichage_graphique=True
```

Utilisation de toutes les cartes

```
# Cas ou l'on souhaite labelliser toutes les cartes
if LAB_all == True:
    for weights_nb in range(1,nb_map+1):
        #=============
        # Chargement de la COA
        #=============
            weights=numpy.load("weights/
                final_weights_%d.npy"%(weights_nb*
                iterations/nb_map))
        ...
```

Utilisation de la carte finale

```
if LAB_nb == None:
    #=================================
    # Chargement de la COA
    #=================================

    weights=numpy.load("weights/final_weights_%
        d.npy"%(iterations))

    ...
```

Influence de σ et η sur le taux d'erreur

On utilise uniquement la carte finale et son score correspondant

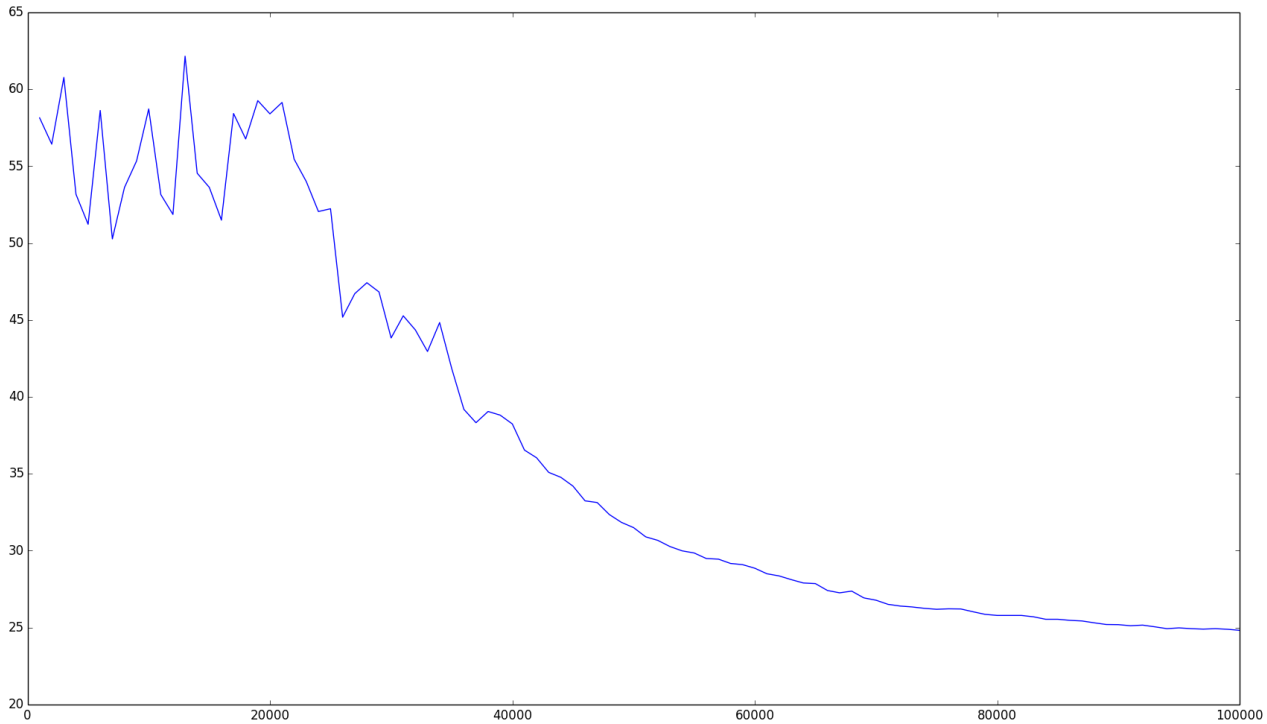
```
LAB_all = False
DEC_all = False
affichage_graphique=False
```

Pourquoi ce choix

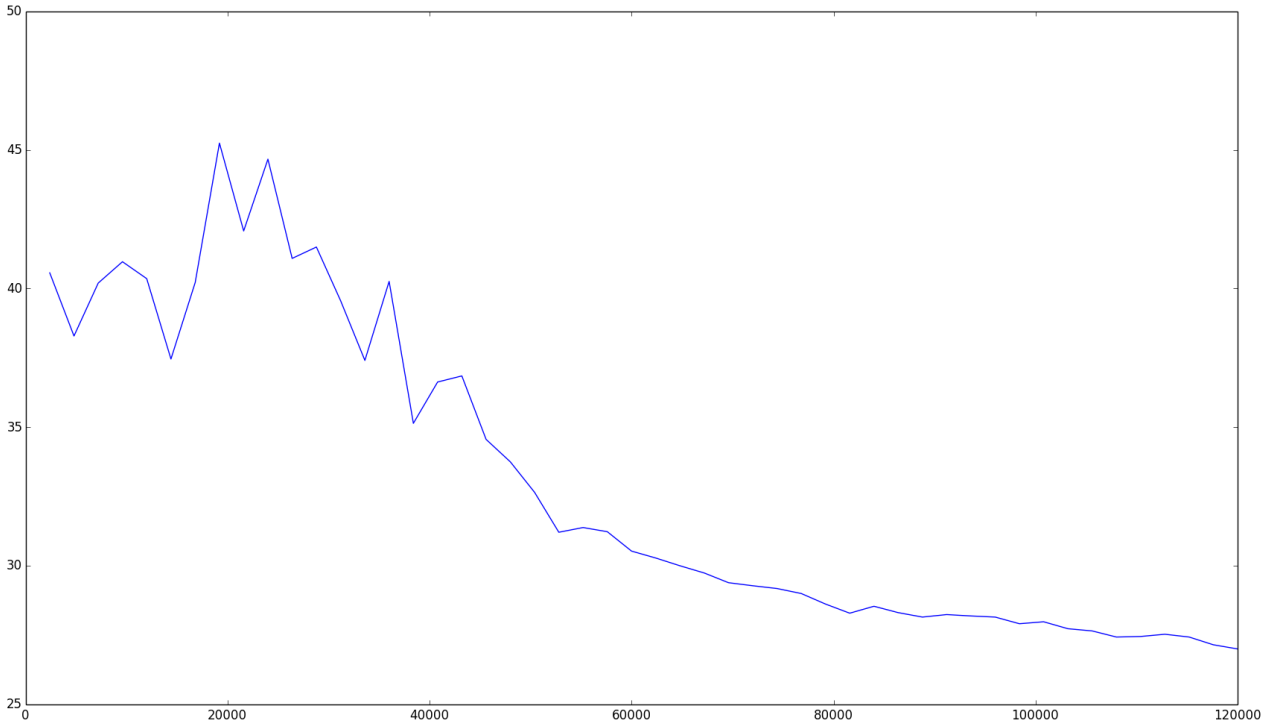
- plus clair pour le client
- double fonction du code.

Déterminer un nombre d'itération efficace

Taux d'erreur en fonction du nombre d'itérations (->100 000)

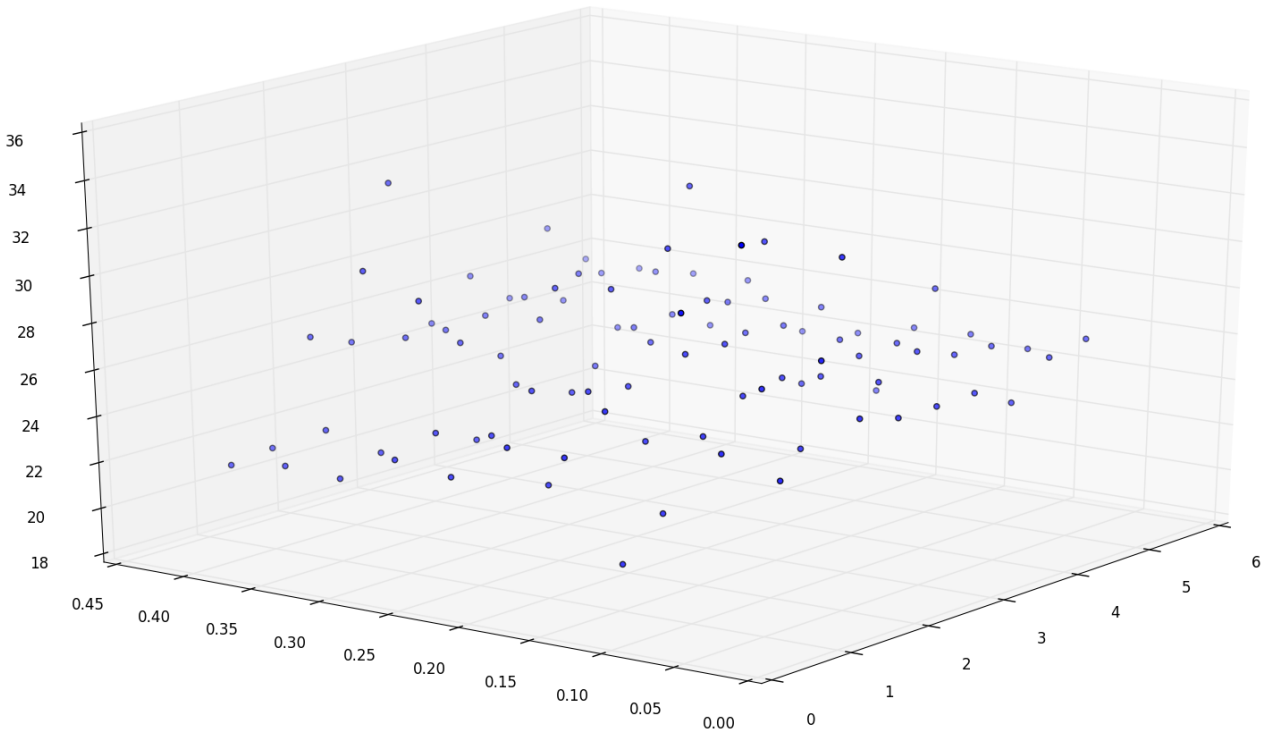


Taux d'erreur en fonction du nombre d'itérations (-> 120 000).

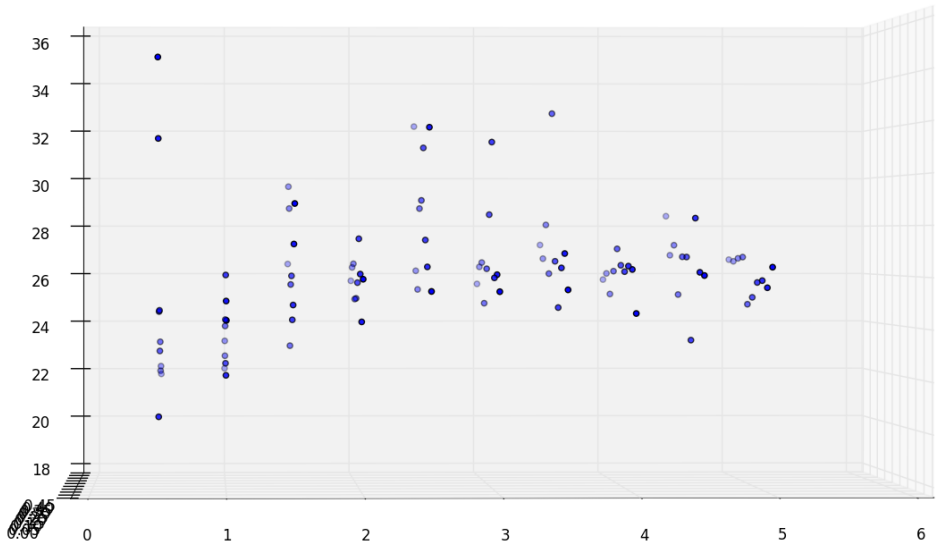


On choisit un nombre d'itérations égal à 100 000 pour la suite.

Taux d'erreur en fonction de σ et η (2h de calculs).



Taux d'erreur en fonction de σ et η (2h de calculs).



On pourra choisir (σ,η)=(0,5 ; 0,10).

Conclusion

- Le choix judicieux du nombre d'itérations, de σ et η améliore le résultat, mais...

Conclusion

- Le choix judicieux du nombre d'itérations, de σ et η améliore le résultat, mais...
- Il est plus efficace de simplement augmenter la taille de la carte.

Conclusion

- Le choix judicieux du nombre d'itérations, de σ et η améliore le résultat, mais...
- Il est plus efficace de simplement augmenter la taille de la carte.
- Après ces réglages, il faut vérifier le fonctionnement concret du code.

Fin