Jeudi 4 novembre 2021

# Compte-rendu 1 : Méthodes statistiques pour la classification dans les chaînes de Markov cachées

Alexandre Chaussard

Antoine Klein

# I) Utilisation du critère du maximum de vraisemblance

*Mise en contexte :* L'objectif de cette partie est de classifier un signal bruité en deux catégories par une méthode statistique paramétrique ; celle du maximum de vraisemblance. Pour rappel, la méthode consiste à déterminer les paramètres de la loi qui maximisent la vraisemblance des données observées.

Nous partons donc d'un signal (signal.npy) que nous bruitons par un bruit gaussien indépendant du signal original, modélisé par une variable aléatoire. Remarquons que dans cette étude nous connaissons les paramètres qui caractérisent le bruit (moyenne et variance) ; ce qui n'est jamais le cas en pratique. En revanche, cela permet de mettre en concurrence les modèles sur un bruit identique et pilotable : on parle de démarche supervisée.

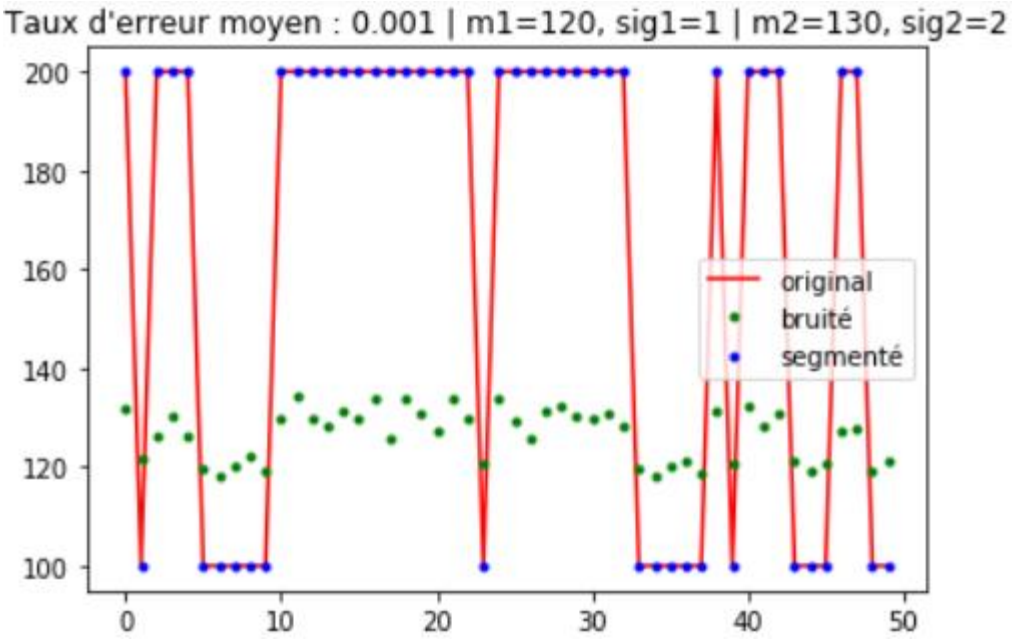


*Figure 1 :* Une réalisation d'un signal bruité et de sa classification

On cherche par la suite à classifier ce signal bruité en deux classes (cl1 et cl2) ; en l'occurrence deux réels, respectivement 100 et 200. Intuitivement, plus les paramètres des classes de bruit sont distants les uns des autres et plus la classification sera efficace (la frontière de décision sera loin des lobes principaux des gaussiennes). Voyons si cela se retrouve confirmé par la pratique.

A présent, nous calculons l'erreur commise par le modèle pour mettre en valeur l'impact des paramètres de bruit sur l'efficacité de la méthode. Si l'erreur dépend de la réalisation de variables aléatoires, la loi des grands nombres nous assure que la moyenne empirique du taux d'erreur moyen s'approche du vrai taux d'erreur moyen à mesure que le nombre de réalisations (=T) augmente.

Les résultats sont donc les suivants :

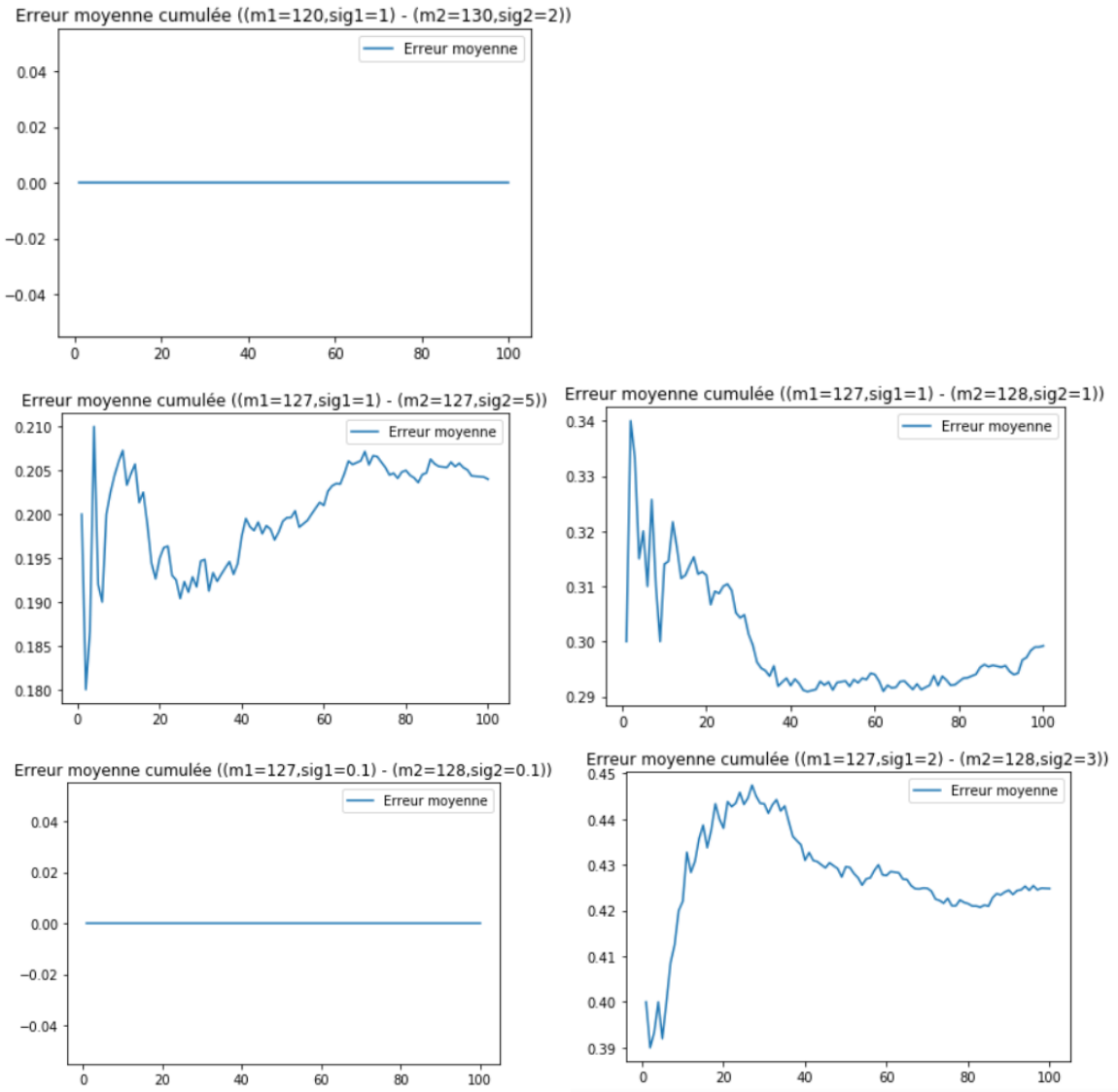


*Figure 2 :* Moyenne empirique du taux d'erreur en fonction du nombre de réalisations (T) pour chacun des 5 bruits proposés

On remarque que l'erreur moyenne empirique converge vers l'erreur moyenne. (C'est bien ce qui est attendu de la loi forte des grands nombres) Avec 100 réalisations nous observons une approximation acceptable.

On constate aussi que l'erreur moyenne est d'autant plus faible que les paramètres du bruit sont distincts (écart de moyenne ou écart de variance). En bref, plus les lobes des gaussiennes sont éloignés, meilleure est la classification.

Un bruit est donc dit fort s'il dégrade considérablement le signal et si son erreur moyenne est élevée indépendamment de sa stratégie de classification. (Propriété intrinsèque du bruit, ne dépendant que des paramètres des classes de bruit).

Un bruit est dit faible s'il ne dégrade que peu le signal et si son erreur moyenne est faible. (Classes de bruit distinctes)

On note les différents bruits selon le tableau suivant :

| Nom du bruit | Moyenne du bruit M1 | Moyenne du bruit M2 | Ecart-type du bruit SIG1 | Ecart-type du bruit SIG2 |
|---|---|---|---|---|
| B1 | 120 | 130 | 1 | 2 |
| B2 | 127 | 127 | 1 | 5 |
| B3 | 127 | 128 | 1 | 1 |
| B4 | 127 | 128 | 0,1 | 0,1 |
| B5 | 127 | 128 | 2 | 3 |

On classe les bruits selon leur impact sur la qualité de la segmentation du signal :

| | B4 | B1 | B2 | B3 | B5 |
|---|---|---|---|---|---|
| Taux d'erreur à $10^{-2}$ | 0% | 0% | 20% | 31% | 44% |
| Impact | faible | faible | fort | fort | fort |

*Figure 3 :* Nos taux d'erreurs moyens pour les différents bruits

*Conclusion :* La première partie traite de la classification par le maximum de vraisemblance. Cette méthode paramétrique a mis en lumière l'impact des classes de bruit sur l'efficacité de la classification. Retenons que plus les classes de bruit sont distinctes et meilleure sera la classification. Cela donne lieu à des bruits forts/faibles dont la difficulté de la classification est intrinsèquement élevée/aisée.

## *Annexe :* Nos codes Python

```python
from scipy.stats import norm
import numpy as np

# bruite le vecteur X avec un bruit gaussien indep.
def bruit_gauss2(X, cl1, cl2, m1, sig1, m2, sig2):

    Y = []

    for x in X:

        u = np.random.random()

        if x == cl1:
            Y.append(np.random.normal(loc=m1, scale=sig1))
        else:
            Y.append(np.random.normal(loc=m2, scale=sig2))

    return Y
```

```python
# Construit le signal segmenté S en classant les données du signal bruité
def classif_gauss2(Y, cl1, cl2, m1, sig1, m2, sig2):

    S = []

    for y in Y:

        if norm.pdf(y, m1, sig1) > norm.pdf(y, m2, sig2):
            S.append(cl1)
        else:
            S.append(cl2)

    return S
```

```python
import matplotlib.pyplot as plt

# Calcul du taux d'erreur
def taux_erreur(A, B):

    if len(A) != len(B):
        print("Impossible de comparer des signaux n'ayant pas le même nombre d'échantillons")
        return

    # Index des signaux différents
    i = 0

    for k in range(0, len(A)):
        if A[k] != B[k]:
            i+=1

    return i/len(A)

# Script première idée
def premiere_idee(m1, sig1, m2, sig2):

    # Chargement des données
    X = np.load("signal.npy")

    # Construction de l'abscisse
    abscisse = []
    for i in range(0, len(X)):
        abscisse.append(i)

    # Récupération des classes
    cl1, cl2 = np.unique(X)

    # Bruitage du signal
    Y = bruit_gauss2(X, cl1, cl2, m1, sig1, m2, sig2)

    # Segmentation
    S = classif_gauss2(Y, cl1, cl2, m1, sig1, m2, sig2)

    # Plot
    fig, ax = plt.subplots()

    ax.plot(abscisse, X, 'r-', label="original")
    ax.plot(abscisse, Y, 'g.', label="bruité")
    ax.plot(abscisse, S, 'b.', label="segmenté")
    ax.legend()

    plt.title("Taux d'erreur moyen : " + str(erreur_moyenne_MV(100, m1, sig1, m2, sig2)) + " | m1="
              + str(m1) + ", sig1=" + str(sig1) + " | m2=" + str(m2) + ", sig2=" + str(sig2))

    plt.show()
```

```python
# Calcul moyen du taux d'erreur
def erreur_moyenne_MV(T, m1, sig1, m2, sig2, toPlot=False, forceX=np.load("signal.npy")):

    # Somme du calul de la moyenne
    tau = 0

    # Calcul de l'erreur partielle pour le plot
    erreur_partielle = []
    abscisse = []

    # Chargement des données
    X = forceX

    # Récupération des classes
    cl1, cl2 = np.unique(X)

    for k in range(0, T):

        # Bruitage du signal
        Y = bruit_gauss2(X, cl1, cl2, m1, sig1, m2, sig2)

        # Segmentation
        S = classif_gauss2(Y, cl1, cl2, m1, sig1, m2, sig2)

        # Construction de la somme
        tau += taux_erreur(X, S)

        # Ajout du taux d'erreur partiel
        erreur_partielle.append(tau/(k+1))
        abscisse.append(k+1)

    if toPlot:
        # Plot de l'erreur partielle
        fig, ax = plt.subplots()
        ax.plot(abscisse, erreur_partielle, label="Erreur moyenne")
        ax.legend()
        plt.title("Erreur moyenne cumulée ((m1=" + str(m1) + ",sig1="+ str(sig1) +") -"
                  + " (m2=" + str(m2) + ",sig2="+ str(sig2) +"))")
        plt.show()

    return tau/T

# Interprétation lorsque T devient grand :
```

```python
# Test de la méthode avec les 6 signaux

M1 = [120, 127, 127, 127, 127]
M2 = [130, 127, 128, 128, 128]
SIG1 = [1, 1, 1, 0.1, 2]
SIG2 = [2, 5, 1, 0.1, 3]

for k in range(0, len(M1)):
    premiere_idee(M1[k], SIG1[k], M2[k], SIG2[k])
    erreur_moyenne_MV(100, M1[k], SIG1[k], M2[k], SIG2[k], toPlot=True)
```