Institut Mines-Télécom

Processus et synthèse vocale sous Python

Roland Badeau

Voir page 5

TELECOM Paris

IP PARIS

1 Étude théorique : Analyse du mélange de deux processus

Soit un signal que l'on modélise par un processus X centré, réel et SSL. Un autre signal est lui aussi modélisé par un processus centré et SSL noté Y. Les processus X et Y sont supposés indépendants. Le processus X est filtré par un filtre dont la RI est h, que l'on ne connaît pas. Il s'ajoute à Y pour donner un processus Z que l'on mesure. Cela s'écrit :

$$Z = h * X + Y$$

On admet sans démonstration que pour toute suite sommable g, le processus $g * X$ est indépendant du processus Y.

On note $T = h * X$. Montrer que

$$S_Z(\nu) = S_Y(\nu) + S_T(\nu)$$

(pour cette preuve, il suffit d'utiliser le fait que T et Y sont indépendants, pas besoin de faire intervenir explicitement le filtre h).

On suppose que l'on a mesuré le processus X. Par exemple X est l'un des instruments participant à un orchestre devant lequel se trouve un micro qui ne mesurer que cet instrument. Z est le son produit par tout l'orchestre mesuré loin de tous les instruments. Et on veut annuler l'effet de l'instrument X.

Montrer que quelque soit la suite sommable h^1 et on notant Z^1 le processus $Z^1 = Z - h^1 * X$ on a

$$R_{Z^1}(0) \geq R_Y(0)$$

et que si $h^1 = h$ alors

$$R_{Z^1}(0) = R_Y(0)$$

Interprétation :

- On interprète Z^1 comme le processus Z débarrassé de tout écho dû à X (on a éliminé l'effet de l'un des instruments).
- On rappelle que $R_B(0)$ (où B est un processus SSL centré) est la puissance de ce processus. L'inégalité démontrée porte donc sur la puissance de Z^1 par rapport à celle de Y .
- Les inégalités démontrées permettent de dire que le problème de retrouver h se réduit à la minimisation de la puissance de $Z - h^1 * X$ où h^1 est l'inconnue.
- Il peut paraître étrange de supposer connu un processus, ici X, que l'on modélise par de l'aléa. Mais il y a des cas où cela est une hypothèse réaliste. Un autre exemple, à part l'orchestre de musique, est la communication entre un téléphone mobile et son antenne relais. Dans ce cas X est une séquence connue que l'antenne envoie au téléphone portable afin d'identifier le canal (les échos des ondes sur les bâtiment, véhicules...) c'est-à-dire trouver h. Et Y est un bruit électromagnétique inconnu.

Dans les sections suivantes nous allons mettre en pratique cette idée dans différents scénarios.

Un programme : Une brique fondamentale vous est donnée. Il s'agit de la fonction `moindres_carres`. Elle prend en entrée un entier p qui est la taille du h à trouver -1 (par souci d'homogénéité des notations avec la partie suivante). Elle prend aussi deux signaux x et z qui jouent les rôles des processus X et Z. Elle renvoie une séquence h^1 de taille $p + 1$ qui minimise l'énergie du signal $z - h^1 * x$.

Téléchargez le fichier **TP_Alea_Python.zip** depuis le Moodle du Master ATIAM. Il contient des données et du code Python. Vous travaillerez dans le répertoire résultant de sa décompression.

1.1 Première application : Identifier un canal

Dans le cas de l'identification de canal, nous ne connaissons pas le canal, c'est-à-dire la RI h. On se donne un signal X que l'on connaît et un signal Y parasite que l'on ne connaît pas. On observe

$$Z = h * X + Y$$

On veut retrouver h à partir de X et de Z. Cela s'écrit en Python :

```
h=np.asarray([1,1/2]) # la RI du canal, on fait semblant de ne pas la connaître...
N=1000
# taille des signaux
X=randn(N)
sigma=0.1
Y=sigma*randn(N) # un signal qui s'ajoute
Z=lfilter(h,[1],X)+Y # X subit l'effet du canal (convolution à h) et Y (inconnu ici) s'ajoute
hestime=moindres_carres(1,X,Z) #trouver h^1
hestime # on trouve le canal (sa RI)
```

Pour différentes valeurs du bruit estimez l'erreur commise sur l'estimation de h, en Python l'erreur relative commise s'écrit

```
norm(hestime-h)/norm(h)
```

(on prendra au moins $\sigma = 0.1, 1, 10$)

Que constatez vous lorsque σ devient trop grand ? Pour $\sigma = 10$, à partir de quelle valeur de N l'estimation de h redevient-elle acceptable (moins de 10% d'erreur relative, par exemple) ?

Dans le programme que nous venons de faire nous connaissons la longueur de la séquence h. Que proposez-vous si la longueur de h n'est pas connue ?

Comparez, pour $\sigma = 1$ l'efficacité du choix d'une séquence $\boxed{\mathrm{X = randn(1, N)}}$ avec l'approche intuitive consistant à prendre $X = \delta$, i.e introduire une impulsion dans le canal et écouter la sortie. À quoi attribuez-vous le manque d'efficacité de l'approche par envoie d'une impulsion ?

1.2 Seconde application : Suppression d'un son qui en parasite un autre.

Dans ce cas, Y est un son qui nous intéresse mais n'est pas connu. X est un son connu, dont on veut éliminer la présence dans Z. Étant donné Z et X on veut estimer h et en déduire une estimation de Y par l'équation

$$Y = Z - h * X$$

Après avoir chargé le fichier "parasite.mat", la variable **Xparasite** contient le signal X et la variable **Zparasite** contient le son mesuré Z dont on veut éliminer l'influence de X. En utilisant un programme sur le modèle suivant

```
hestim=moindres_carres(p_est_inconnu,Xparasite,Zparasite)
Y=Zparasite-lfilter(hestim,1,Xparasite)
```

Comment avez-vous fait pour estimer la taille du filtre ? Quel filtre h avez-vous trouvé ? Ecoutez **Zparasite** et **Y** par la commande $\boxed{\mathrm{play(VARIABLE,22050);}}$ [1] pour vous assurer que vous avez bien éliminé l'influence du son parasite.

1. les sons de ce TP sont échantillonnés à 22050 Hz

2 Le modèle auto-régressif pour la synthèse de parole

Dans le modèle auto-régressif on suppose que le signal enregistré est X et qu'il est obtenu à partir d'un bruit blanc ϵ suivant le filtre récursif suivant

$$X_n + h_1 X_{n-1} + \ + h_p X_{n-p} = \epsilon n$$

Ceci modélise le fait qu'un signal qui possède les statistiques d'un bruit blanc, ici ϵ passe dans un filtre qui se trouve être récursif. Dans ce problème il y a deux inconnus, les h_j et le signal ϵ.

Un autre programme : La fonction lpc_morceau calcule les coefficients h_j. Elle est très semblable à la fonction moindres_carres : elle minimise la norme de $h * X$, sauf qu'écrit comme cela il suffit de prendre $h = 0$ pour obtenir une solution imbattable. On force $h_0 = 1$ pour éviter une solution triviale et on minimise $\|h * X\|$.

2.1 Test du modèle et du programme fourni

À l'issue des commandes suivantes, on voudrait avoir **hestim=h**

```
N=1000
h=np.asarray([1,1/2])
sigma=1
epsilon=sigma*randn(N)
X=lfilter([1],h,epsilon) # help(lfilter) pour comprendre la position du h
hestim=lpc_morceau(1,X)
hestim
```

Retrouvez-vous bien le filtre h utilisé ? Que se passe-t-il sous Python lorsque la première ligne est remplacée par **h=[1 2];** ? Comment expliquez-vous cela ? (on peut se reporter à la partie TZ du polycopié).

Revenez à un filtre h qui n'a pas de pôles à l'extérieur du cercle unité (causal).

Expliquez la valeur affichée par la suite de commandes (pour cela il faut bien lire l'aide de la commande lfilter, "help lfilter")

```
tmp=lfilter(hestim,1,X); #Remarquer la position de hestim
sum(tmp**2)/(N*sigma**2)
```

2.2 Application du modèle à un signal de parole

La fonction lpc découpe un signal en trames d'une taille prescrite et recherche pour chaque trame les coefficients h qui expliquent le mieux le signal observé. Elle renvoie un tableau dont chaque ligne est les coefficients de h pour la trame correspondante et un vecteur qui contient l'écart-type de ϵ pour chacune des trames.

La fonction joue_lpc fabrique un son en générant un bruit blanc ϵ de même écart-type que celui calculé par le programme lpc puis le fait passer dans le filtre spécifié pour chaque trame.

Chargez le son aeiou (voix) qui se trouve dans le fichier modelear.mat.

```
[lpcs,res]=lpc(aeiou,15,2000);
joue_lpc(lpcs,res,2000,22050);
```

Qu'entendez-vous ? Que se passe-t-il si on élimine le préfiltrage au début du programme lpc ?

Remarquez-vous une grande amélioration/dégradation lorsque le paramètre p varie ?

Modifiez la fonction joue_lpc afin qu'elle regénère le son non pas en introduisant un bruit blanc mais un signal formé d'impulsions éloignées de 0.01 secondes entre elles (on rappelle que les sons traités ici sont échantillonnés à 22050Hz).

En utilisant la fonction affiche_spectrogramme dire à quelle fréquence les impulsions devraient être envoyées dans le filtre.

Que pouvez-vous suggérer pour améliorer la qualité de la restitution sonore ? (voir l'alternative dans le programme joue_lpc)

3 Estimation de la densité spectrale de puissance : périodogramme

On cherche à estimer la densité spectrale de puissance (DSP) d'un processus X_n stationnaire au sens large et de moyenne nulle, observé sur un intervalle $n \in [0 \ldots N-1]$, à l'aide de la transformée de Fourier discrète (TFD), implémentée dans la fonction Numpy `fft.fft`. On rappelle que la DSP $S_X(e^{2i\pi\nu})$ est définie comme la transformée de Fourier à temps discret (TFTD) de la fonction d'autocovariance $R_X(k)$: $S_X(e^{2i\pi\nu}) = \sum_{k\in\mathbb{Z}} R_X(k)e^{-2i\pi\nu k}$.

1. Soit $\hat{R}_N(k)$ l'estimateur biaisé de la fonction d'autocovariance du processus X_n. On rappelle que le périodogramme de X, que nous noterons $\widehat{S}_N(e^{2i\pi\nu})$, est défini comme $\widehat{S}_N(e^{2i\pi\nu}) = \sum_{k=-N+1}^{N-1} \hat{R}_N(k)e^{-2i\pi\nu k}$. Exprimer $\widehat{S}_N(e^{2i\pi\nu})$ en fonction de la TFTD du signal $X_0 \ldots X_{N-1}$.
2. En déduire un algorithme de calcul de $\widehat{S}_N(e^{2i\pi k/M})$ pour $k \in [0 \ldots M-1]$, où $M \geq N$ est l'ordre de la TFD utilisée, que vous implémenterez en Python. Testez-le sur un bruit blanc.
3. Comment obtenir la séquence $\hat{R}_N(k)$ de l'estimateur biaisé de la fonction d'autocovariance à partir de $\widehat{S}_N(e^{2i\pi k/M})$? Quelle précaution faut-il prendre sur M ? Écrire une fonction qui calcule $\hat{R}_N(k)$ en exploitant cette propriété. Testez-la sur un bruit blanc. Comparer sa complexité algorithmique à celle d'un calcul direct de $\hat{R}_N(k)$.
4. Toujours dans le cas du bruit blanc, estimer la variance du périodogramme pour plusieurs horizons d'observation N, comme fonction de la fréquence réduite ν. Qu'observez-vous ?
5. Implémentez la méthode du périodogramme fenêtré présentée en cours, que vous testerez sur un bruit blanc. Estimez la variance de ce périodogramme fenêtré pour plusieurs horizons d'observation N. Qu'en concluez-vous ?