TP2. Traitement du signal.

Frédéric Richard

Cours apprentissage statistique et réseaux de neurones
Master mathématiques appliquées, statistique,
Parcours Data Science.

2025, AMU

1 Exercice 1. Propriétés des coefficients de Fourier.

1.1 Rappel de cours.

Soit $d > 0$ un entier représentant une dimension.

On considère une application g dans $L^2_P([0, 2\pi[^d)$, c'est à dire une application définie sur $\mathbb{R}^d$ à valeurs dans $\mathbb{C}$, 2π-périodique, de carré intégrable sur $[\not<, \not< \pi[$.

Pour $n \in \mathbb{Z}^d$, on note $c_n(g)$, le **coefficient de Fourier** de g, défini par

$$c_n(g) = \int_{[0,2\pi]^d} g(s) e^{-i\langle n,s\rangle} ds.$$

Dans $L^2_P([0, 2\pi[^d)$, on a la **série de Fourier**

$$g(y) = \sum_{n\in\mathbb{Z}^d} c_n(g)\, e^{i\langle n,y\rangle}.$$

Soieng g et h dans $L^2_P([0, 2\pi[^d)$. On note $g \circledast h$ le produit de convolution périodique de g et h, application définie, pour tout $y \in \mathbb{R}$, par

$$g \circledast h(y) = \int_{[0,2\pi[^d} g(s)h(y-s)ds = \int_{[0,2\pi[^d} g(y-s)h(s)ds.$$

1.2 Partie 1.

On fixe $d = 1$.

1. Soit $g \in L^2_P([0, 2\pi[$. On suppose que g est dérivable sur $\mathbb{R}$ et que $g' \in L^2_P([0, 2\pi[)$. Montrer que, pour tout $n \in \mathbb{Z}$,

$$c_n(g') = inc_n(g).$$

2. On suppose à présent que g est deux fois dérivable sur $\mathbb{R}$ et que $g'' \in L^2_P([0, 2\pi[)$. Montrer que, pour tout $n \in \mathbb{Z}$,

$$c_n(g'') = -n^2 c_n(g).$$

3. Soient f et g dans $L^2_P([0, 2\pi[)$. Etablir que, pour tout $n \in \mathbb{Z}$,

$$c_n(g * h) = c_n(g)\, c_n(h).$$

1.3 Partie 2.

On fixe $d = 2$.

Soit g dans $L^2_P([0, 2\pi[^2)$.

1. On suppose que g est différentiable sur $\mathbb{R}^2$, de dérivées partielles $\partial_j g$ dans $L^2_P([0, 2\pi[^2)$. Montrer que, pour $j \in [\![1, 2]\!]$ et $n \in \mathbb{Z}^2$,

$$c_n(\partial_j g) = i n_j c_n(g).$$

2. On suppose que g est deux fois différentiable sur $\mathbb{R}^2$ et de dérivées partielles secondes $\partial^2_{jk} g$ dans $L^2_P([0, 2\pi[^2)$. Montrer que, pour $j, k \in [\![1, 2]\!]$ et $n \in \mathbb{Z}^2$,

$$c_n(\partial^2_{jk} g) = -n_j n_k c_n(g).$$

2 Exercice 2. Mise en oeuvre de filtres linéaires.

2.1 Rappel de cours.

Soient d et N dans $\mathbb{N}^*$. On considère deux signaux discrets f et g, définies sur la grille uniforme $[\![0, N-1]\!]^d$. On note $f[n]$ et $g[n]$ les valeurs de ces signaux en une position n de la grille. On étend la définition de ces signaux à $\mathbb{Z}^d$ par périodisation.

On rappelle que

- le produit de convolution périodique discret $h = f \circledast g$ est le signal défini pour tout $n \in \mathbb{Z}^d$ par

$$f \circledast g[n] = \sum_{k \in [\![0, N-1]\!]^d} f[n-k] g[k].$$

- la transformée de Fourier discrète $\hat{f}$ du signal N-périodique f est le signal défini pour tout $n \in \mathbb{Z}^d$ par

$$\hat{f}[n] = \sum_{k \in [\![0, N-1]\!]^d} f[k] \exp\left(-i \frac{2\pi \langle k, n \rangle}{N}\right).$$

- la transformée de Fourier discrète inverse $\check{f}$ du signal N-périodique f est le signal défini pour tout $n \in \mathbb{Z}^d$ par

$$f[n] = \frac{1}{N^d} \sum_{k \in [\![0, N-1]\!]^d} f[k] \exp\left(i \frac{2\pi \langle k, n \rangle}{N}\right).$$

2.2 Partie 1. TFD et produit de convolution.

On fixe $d = 2$.

1. Vérifier que $f \circledast g$ et $\hat{f}$ sont tous les deux des signaux N-périodiques.
2. Montrer que, pour tout $n \in \mathbb{Z}^2$,

$$\widehat{f \circledast g}[n] = \hat{f}[n]\, \hat{g}[n].$$

3. Vérifier que, pour tout $n \in \mathbb{Z}^2$,

$$f[n] = (\hat{f})^\vee[n].$$

4. En déduire une méthode basée sur la transformée de Fourier discrète permettant de calculer le produit de convolution périodique discret.

2.3 Partie 2. Filtre gaussien.

Soit N un entier pair et $s > 0$. On considère un filtre linéaire dont la fonction de transfert est le signal gaussien discret N-périodique défini pour $n \in [\![-\frac{N}{2}, \frac{N}{2} - 1]\!]^2$ par

$$\hat{g}[n] = \frac{1}{Z_{N,s}} \exp\left(-s|n|^2\right) = \frac{1}{Z_{N,s}} \exp\left(-s(n_1^2 + n_2^2)\right),$$

où $Z_{N,s}$ est une constante de normalization donnée par

$$Z_{N,s} = \sum_{n \in [\![-N/2, N/2-1]\!]^2} \exp\left(-s|n|^2\right),$$

1. En complétant la fonction ci-dessous, écrire une fonction python qui étant donnés les paramètres N et s renvoie la fonction de transfert $\hat{g}$.

[1]:
```
from numpy import floor, exp, power, arange, reshape

def Gaussien(N=512, s=0.01):
    """Define a bi-dimensional N-periodic Gaussian kernel.

   Parameters
    ----------
    N : int
        image size. The default is 512.
    s : positive float, optional
        scale parameter. The default is 1.
```

```
    Returns
    -------
    The Gaussian kernel
    """
    M = floor(N / 2)

    # Definition of an unidimensional Gaussian kernel.
    g = reshape(exp(- s * power(arange(- M, M, 1), 2)), (N, 1))

    # Extension to a bi-dimensional Gaussian kernel.
    ## To be completed.

    return(g)
```

2. Afficher la fonction de transfert $\hat{g}$ avec la commande *imshow* du module **matplotlib.pyplot**. Donner une interprétation du filtre.
3. Appliquer le filtre de réponse de transfert $\hat{g}$ à l'image *baboon.jpg*. On utilisera les méthodes *fft2* et *ifft2* du module **numpy.fft** pour réaliser les transformées de Fourier discrètes (directe et inverse). Avant d'appliquer le filtre, veiller à replacer le pixel $n = [0, 0]$ de la fonction de transfert à l'indice (0, 0) de la matrice en opérant une translation avec la commande *fftshift* de **numpy.fft**.
4. Faire varier les valeurs de s et commenter l'effet de ce paramètre sur la sortie du filtre.

2.4 Partie 3. Filtres dérivatifs.

Soit N un entier pair et $s > 0$. On considère un filtre linéaire dont la fonction de transfert est le signal gaussien discret N-périodique défini pour $n \in [\![-N/2, N/2-1]\!]^2$ par

$$\hat{g}[n] = \frac{1}{Z_{N,s}} \exp\left(-s|n|^2\right) = \frac{1}{Z_{N,s}} \exp\left(-s(n_1^2 + n_2^2)\right),$$

où $Z_{N,s}$ est une constante de normalization définie par

$$Z_{N,s} = \sum_{n \in [\![-N/2, N/2-1]\!]^2} \exp\left(-s|n|^2\right),$$

On considère trois nouvelles fonctions de transfert définies pour $n = (n_1, n_2) \in [\![-N/2, N/2-1]\!]^2$ de la manière suivante:

- pour $j \in [\![1, 2]\!]$

$$\widehat{\partial_j g}[n] = i n_j \hat{g}[n]$$

et

$$\widehat{\Delta g}[n] = -|n|^2 \hat{g}[n].$$

1. En vous référant à l'exercice 1, donner une interprétation des filtres associés à ces fonctions de transfert.
2. Créer une fonction python qui définit ces fonctions de transfert. Visualiser ces fonctions de transfert et indiquer si les filtres associés sont passe-haut, passe-bas ou passe-bande ?
3. Appliquer les filtres associés à ces fonctions de transfert à l'image *baboon.jpg*. On notera

- $\partial_j f = f \circledast \partial_j g$, pour $j \in [\![1, 2]\!]$,
- $\Delta f = f \circledast \Delta g$.

Visualiser les réponses de ces filtres. Quelles informations apportent-elles ?

4. Pour une image donnée f, former l'image h définie pour tout $n \in \mathbb{Z}^2$ par

$$h[n] = |f \circledast \partial_1 g[n]|^2 + |f \circledast \partial g_2[n]|^2.$$

Que représente cette image ? A quoi correspondent les zones où les valeurs de cette image sont élevées ? Pour s'en rendre compte, on pourra seuiller l'image, c'est à dire définir l'image binaire

$$h_\eta[n] = \begin{cases} 1 & \text{si } h[n] > \eta, \\ 0 & \text{sinon} \end{cases}$$

pour η fixé convenablement.

3 Exercice 3. Détection de contours dans une image.

3.1 Introduction.

Les contours d'une image sont des courbes du domaine de l'image où sont localisées des fortes variations de niveaux de gris de l'image. Ces contours peuvent indiquer des frontières entre des objets présents dans l'image.

Pour détecter des zones où les variations des niveaux gris sont élevées, on peut utiliser la norme du gradient de l'image. En effet, une norme de gradient élevée est un indicateur de fortes variations des niveaux de gris. Ainsi, en seuillant la norme du gradient de l'image, il est possible de repérer des zones de fortes variations.

Cependant, les zones ainsi détectées ne sont généralement pas des courbes. Elles forment le plus souvent des bandes. Bien que ces bandes contiennent les contours, elles sont plus "larges" que ces derniers.

Pour réduire ces bandes à des courbes, on se restreint généralement aux pixels où la norme du gradient est localement maximale dans la direction du gradient. Pour déceler cette maximalité locale de la norme du gradient, Marr et Hildreth (1980) ont proposé de répérer les passages par zéro du laplacien de l'image.

3.2 Algorithme.

De cette approche, on tire l'algorithme de détection de contours suivant.

- Etape 1: pour une image donnée f, on calcule le laplacien de l'image Δf définie, pour tout $n \in \mathbb{Z}^2$, par

$$\Delta f[n] = f \circledast \Delta g[n],$$

où g est un filtre gaussien discret et ∇g son laplacien.

- Etape 2: On détecte les passages par zéro de Δf:
  - Pour cela, on forme l'image du signe k_0 de Δf:

$$k_0[n] = \begin{cases} 1 & \text{si } \Delta f[n] > 0, \\ -1 & \text{sinon.} \end{cases}$$

  - On calcule la réponse $\tilde{k}$ de l'image k_0 au filtre dont le noyau est à support sur $[\![-1,1]\!]^2$ et a pour valeurs sur $[\![-1,1]\!]^2$

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & -1 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

- Etape 3: On calcule la norme (au carré) du gradient de h:

$$h[n] = |f \circledast \partial_1 g[n]|^2 + |f \circledast \partial g_2[n]|^2.$$

- Etape 4: Les contours sont définis par les pixels n tels que $\tilde{k}[n] > 0$ et $h[n] > \eta$, pour $\eta > 0$ fixé convenablement.

3.3 Mise en oeuvre.

1. Mettre en oeuvre la méthode et l'appliquer à l'image *baboon.jpg*.
2. Faire varier les paramètres s et η de la méthode et analyser leur effets sur la détection de contours.

4 Exercice 4. Déflouage d'image.

4.1 Introduction.

A l'acquisition, une image peut subir des dégradations telles que du flou ou du bruit. Dans certains cas, on peut modéliser ces dégradations au moyen du modèle suivant.

$$h = f \circledast g + b,$$

où - h est l'image observée, - b est un bruit additif gaussien indépendant de f et g, - f est une image non observée sans dégradation ni bruit, - g est un noyau de convolution qui, appliquée à f, la dégrade. Par exemple, si g est la réponse impulsionnelle d'un filtre passe-bas, la convolution de f avec g va rendre l'image f floue.

Déconvoluer l'image consiste à retrouver f à partir de l'image dégradée h. On parle de déflouage d'image lorsque l'image a été dégradée par un filtre passe-bas.

4.2 Partie 1. Floutage d'une image.

On prend un filtre gaussien discret g comme noyau de convolution

1. Flouter l'image *baboon.jpg* en lui appliquant le filtre g.
2. Ajouter un bruit gaussien à l'image floutée.

4.3 Partie 2. Méthode d'inversion directe.

1. Dans un premier temps, on considère le modèle de dégradation sans bruit ($b = 0$). On suppose que la TFD $\hat{g}$ de g ne s'annule pas. Exprimer la TFD de f en fonction de celles de h et g.
2. En déduire une méthode pour déconvoluer h en utilisant la TFD et la TFDI.
3. Appliquer cette méthode à l'image de *baboon.jpg* convoluée avec le filtre gaussien g. Evaluer l'erreur d'approximation de f en calculant la différence quadratique entre l'image déconvoluée et l'image originale.
4. Ajouter un bruit dans le modèle de dégradation. Appliquer la méthode d'inversion précédente à cette nouvelle image dégradée. Que constatez-vous et comment expliquez-vous ce résultat ?

4.4 Partie 3. Méthode par résolution de problème inverse.

On peut également déconvoluer une image dégradée h en minimisant un problème d'optimisation un critère de la forme

$$J(f) = \frac{1}{2} \sum_{n \in [\![0, N-1]\!]^2} |f \circledast g[n] - h[n]|^2 + \lambda R(f),$$

Dans ce critère, le premier terme mesure l'écart entre l'image observée h et la valeur prédite de h à partir de f. Il s'agit d'un **terme d'attache aux données**. Le second terme induit des solutions f régulières. Il permet au problème d'optimisation d'être bien posé. Il s'agit d'un **terme de régularisation**. La valeur $\lambda > 0$ est un paramètre qui permet de régler le compromis entre l'attache aux données et la régularisation. Une régularisation typique consiste à pénaliser le gradient ∇f de f avec un terme de la forme

$$R(f) = \frac{1}{2} \sum_{n \in [\![0, N-1]\!]^2} |\nabla f[n]|^2 = \frac{1}{2} \sum_{n \in [\![0, N-1]\!]^2} \left((\partial_1 f[n])^2 + (\partial_2 f[n])^2 \right).$$

Le critère J peut s'écrire sous une forme matricielle. On "aplatie" les images f et h en concaténant leurs colonnes en un seul vecteur. Le critère J se présente alors sous la forme

$$\tilde{J}(\tilde{f}) = \frac{1}{2} |G\tilde{f} - \tilde{h}|^2 + \frac{\lambda}{2} \left(|D_1 \tilde{f}|^2 + |D_2 \tilde{f}|^2 \right),$$

où - $\tilde{f}$ et $\tilde{h}$ sont les versions aplaties de f et h, respectivement, - G est une matrice de taille $N^2 \times N^2$ qui permet de réaliser le produit de convolution de f avec le noyau g, - D_1 et D_2 sont des matrices

de taille $N^2 \times N^2$ qui permettent de calculer les dérivées de f selon les lignes et les colonnes, respectivement.

1. En vous aidant de la forme matricielle de J, montrer que le problème d'optimisation admet une solution et l'expliciter en fonction de $\tilde{h}$.
2. Mettre en oeuvre et évaluer la méthode de déflouage par résolution du problème inverse. On pourra s'aider des fonctions ci-dessous qui permettent de mettre le problème d'optimisation sous une forme matricielle.

[3]:
```python
def create_2d_convolution_matrix(v):
    """Crée une matrice circulante 2D (bloc-circulante) V à partir du noyau 2D v.

    Parameters:
        v (numpy array): Noyau de convolution 2D.

    Returns:
        V (numpy array): Matrice circulante 2D basée sur v.
    """
    M, N = v.shape  # Dimensions du noyau de convolution

    padded_v = np.zeros((M, N))
    padded_v[:, :] = v

    # Création de la matrice circulante 2D (bloc-circulante)
    V = np.zeros((M * N, M * N))
    for i in range(M):
        for j in range(N):
            # Décalage circulant de g
            shifted_g = np.roll(np.roll(padded_g, i, axis=0), j, axis=1)
            V[i * N + j, :] = shifted_g.flatten()

    return V

def flatten_signal(f):
    """Applatie le signal 2D f."""
    return f.flatten()
```