Autori: Andra Băltoiu, Silviu Gheorghe, Gabriel Majeri, Rebeca Oprea

Procesarea Semnalelor
Laboratorul 9
Antrenarea dicționarelor

1 Eliminarea zgomotului dintr-o imagine

Scopul acestui laborator este de a elimina zgomotul dintr-o imagine (*denoising*), prin antrenarea unui dicționar pentru reprezentări rare.

Algoritmii și structurile de date care vă vor ajuta în rezolvarea temei se regăsesc în biblioteca `dictlearn`[1]. Documentația oficială a acesteia este disponibilă aici.

Figura 1: Eliminarea zgomotului. a) Imaginea originală. b) Imaginea alterată de zgomot. c) Imaginea rezultată.

[1] *Dictionary Learning Toolbox*. Autori: Paul Irofti, Denis Ilie-Ablachim și Bogdan Dumitrescu

Privire de ansamblu

În paragrafele de mai jos, vom folosi notația/denumirile vectorilor din documentația oficială `dictlearn`.

Generarea imaginii cu zgomot

În primă fază, veți altera conținutul imaginii `I` (figura 1, a)), prin modificarea valorii fiecărui pixel: veți adăuga la fiecare pixel un zgomot (un număr), generat după o distribuție normală de medie 0 și deviație standard σ. Veți obține astfel imaginea `Inoisy` (figura 1), b)).

Descompunerea în *patch*-uri

Pentru eliminarea zgomotului veți utiliza metoda *patch*-urilor distincte și care nu se suprapun.

Pentru a obține semnalele **Y** ce vor fi utilizate ulterior în antrenare, trebuie mai întâi să **extragem *patch*-urile** din imagine. Fiecare semnal va fi un *patch* de dimensiune $p \times p$, vectorizat.

Antrenarea dicționarului

Eliminarea zgomotului presupune învățarea unui dicționar **D** și a reprezentării rare corespunzătoare **X**, care să reprezinte cât mai bine datele inițiale **Y**.

Intuiția este că dicționarul învățat va putea reprezenta numai tipare recurente din imagini, neputând învăța zgomotul. Astfel, la reconstrucția datelor inițiale pe baza dicționarului învățat, se va elimina zgomotul.

Dicționarul **D** se inițializează aleator și se normalizează, fiecare din cele n coloane este un vector de medie 0 și normă (lungime) 1. Dicționarul **D** este redundant, conține mult mai multe coloane decât cele $m = p \times p$ coloane necesare unei baze din $\mathbb{R}^m$.

Calcularea reprezentării rare

Învățarea reprezentării rare **X** și a dicționarului **D** se realizează utilizând algoritmii OMP și K-SVD.

La finalul antrenării, veți calcula reprezentarea rară a imaginii cu zgomot, folosind dicționarul antrenat anterior. Pentru aceasta puteți folosi metoda `omp`, ce găsește reprezentarea rară a semnalelor pe baza nivelului de sparsitate impus s.

Obțineți astfel reprezentarea $\mathbf{X_c}$, în baza căreia reconstruiți imaginea `Ic`, care nu ar trebui să conțină zgomot.

Figura 1, c) prezintă imaginea din care a fost eliminat zgomotul, `Ic`.

Evaluarea performanței

Pentru a evalua performanța soluției, veți folosi *Peak Signal to Noise Ratio* (PSNR). Veți determina PSNR între imaginea *denoised* și imaginea originală, respectiv între imaginea *noisy* și cea originală. O valoare mai mare în primul caz indică o recuperare fidelă a imaginii inițiale.

2 Ghid Python

În acest laborator veți avea nevoie de bibliotecile `numpy`, `matplotlib`, `sklearn` și `dictlearn`.

`scikit-learn` este o bibliotecă generalistă de învățare automată. O vom folosi doar pentru funcționalitățile utilitare de prelucrare a datelor. Se poate instala prin comanda `pip install sklearn`.

Pentru a instala pachetul *Dictionary Learning*, utilizați comanda `pip install dictlearn`. Ulterior îl puteți încărca prin secvența

```
from dictlearn import DictionaryLearning
```

iar prin secvența

```
from dictlearn import methods
```

puteți încărca și modulul ce conține metode pentru algoritmii de reprezentări rare și pentru actualizarea dicționarului.

În implementarea voastră, utilizați următoarele valori ale parametrilor:

```
p = 8              # dimensiunea unui patch (numar de pixeli)
s = 6              # sparsitatea
N = 1000           # numarul total de patch-uri
```

```
n = 256            # numarul de atomi din dictionar
K = 50             # numarul de iteratii DL
sigma = 0.075      # deviatia standard a zgomotului
```

Pentru a încărca o imagine în memorie, importați mai întâi submodulul `image` din biblioteca `matplotlib`,

```
from matplotlib import image
```

iar apoi folosiți secvența de cod

```
I = image.imread('nume_imagine.png')
```

Dacă imaginea **nu** este deja reprezentată ca *float* (verificați `I.dtype`), cu intensitățile pixelilor cuprinse între 0 și 1, o puteți normaliza prin secvența

```
I = I / 255.0
```

Pentru a vizualiza o imagine citită, puteți încărca biblioteca `matplotlib.pyplot` iar apoi să folosiți `imshow`:

```
import matplotlib.pyplot as plt
plt.imshow(I, cmap='gray')
```

Pentru a adăuga zgomot distribuit normal în fiecare pixel al imaginii normalizate, puteți folosi o secvență similară cu:

```
Inoisy = I + sigma*np.random.randn(I.shape[0], I.shape[1])
```

Pentru a extrage *patch*-uri dintr-o imagine și apoi pentru a reconstrui o imagine din patch-uri folosiți

```
from sklearn.feature_extraction.image \
import extract_patches_2d, reconstruct_from_patches_2d
```

Documentația acestor două funcții se găsește aici. Prima primește ca parametrii imaginea și dimensiunea patch-urilor (în cazul vostru, un tuplu de 2 elemente, reprezentând cele 2 dimensiuni ale patch-ului), iar a doua primește ca parametrii lista de patch-uri și dimensiunea imaginii finale (tot ca tuplu).

Pentru a normaliza dicționarul, utilizați

```
from sklearn.preprocessing import normalize
D0 = normalize(D0, axis=0, norm='max')
```

Pentru a antrena un dicționar pe datele $\mathbf{Y}$, utilizați secvența

```
dl = DictionaryLearning(
    n_components=n,
    max_iter=K,
    fit_algorithm='ksvd',
    n_nonzero_coefs=s,
    code_init=None,
    dict_init=D0,
    params=None,
    data_sklearn_compat=False
)
dl.fit(Y)
D = dl.D_
```

Secvența creează o instanță a clasei `DictionaryLearning`, folosind constructorul cu parametrii, apoi antrenează dicționarul pe semnalele `Y` și în final memorează dicționarul în variabila `D`.

Pentru a calcula reprezentarea rară a unui semnal, utilizați

```
X, err = methods.omp(Ynoisy, D, n_nonzero_coefs=s)
```

Pentru a calcula valoarea PSNR între două imagini, folosiți următoarea funcție:

```
def psnr(img1, img2):
    mse = np.mean((img1 - img2) ** 2)
    if(mse == 0):
        return 0
    max_pixel = 255
    psnr = 20 * np.log10(max_pixel / np.sqrt(mse))
    return psnr
```

3 Exerciții

1. Pregătirea imaginii.
   (a) Încărcați imaginea. Veți obține imaginea `I` de dimensiune $m_1 \times m_2$.
   (b) Adăugați zgomot cu dispersie σ imaginii.
   (c) Extrageți *patch*-urile din imaginea `Inoisy` și memorați-le în variabila `Ynoisy`.

Funcția `extract_patches_2d` va returna colecția de *patch*-uri, însă pentru a putea fi utilizate în continuare în antrenarea dicționarului e nevoie de ajustarea dimensiunii, respectiv de vectorizarea *patch*-urilor.

Afișați întâi dimensiunea `Ynoisy`. Pentru a vectoriza *patch*-urile folosiți

```
Ynoisy = Ynoisy.reshape(Ynoisy.shape[0], -1)
```

Afișați din nou dimensiunea `Ynoisy`, pentru a observa rezultatul vectorizării.

Transpuneți matricea (`np.transpose`), apoi calculați media semnalelor pe linii (axa 0; puteți folosi `np.mean`), reprezentând media *patch*-urilor. Scădeți-o din `Ynoisy`.

(d) Selectați `N` *patch*-uri de dimensiune `p*p` la întâmplare din imagine, obținând astfel semnalele `Y`.

Pentru aceasta, generați un set de N indici

```
indices = numpy.random.choice(Ynoisy.shape[1], N)
```

cu care să indexați în `Ynoisy`

```
Y = Ynoisy[:, indices]
```

2. Antrenarea dicționarului.

(a) Generați un dicționar aleator (de exemplu, tot cu ajutorul funcției `numpy.random.randn`) și normați coloanele, obținând dicționarul `D0`. Reamintim că dicționarul trebuie să fie, conform documentației, o matrice de dimensiuni $p^2 \times n$.

(b) Antrenați dicționarul `D` pornind de la dicționarul `D0` inițializat mai sus, pentru `K` iterații, utilizând *patch*-urile selectate `Y` ca semnale de antrenare.

Observație: antrenarea s-ar putea să dureze până la câteva minute, în funcție de puterea calculatorului vostru.

3. Calcul reprezentării rare și reconstrucția imaginii.

(a) Calculați reprezentarea rară a tuturor semnalelor `Ynoisy` (folosind algoritmul OMP), obținând astfel `Xc`.

Observație: calculul reprezentării rare s-ar putea să dureze până la câteva minute, în funcție de puterea calculatorului vostru.

(b) Obțineți *patch*-urile curate, `Yc`, utilizând dicționarul `D` și reprezentarea `Xc`, apoi adăugați media pe linii pe care ați scăzut-o anterior.

Reamintim că înmulțirea între matricea dicționarului și matricea care conține semnalele rare se poate obține prin expresia `D @ Xc`.

(c) Transpuneți matricea `Yc` și schimbați forma pentru a anula efectele vectorizării:

```
Yc = Yc.reshape(Yc.shape[0], p, p)
```

(d) Reconstruiți imaginea din patch-urile `Yc`, obținând imaginea curată `Ic`.

4. Evaluarea performanței.

(a) Vizualizați cele trei imagini (originală, alterată de zgomot și curățată de zgomot).

(b) Calculați *PSNR* pentru a măsura reducerea zgomotului. Calculați atât *PSNR* între imaginea originală și cea afectată de zgomot, cât și între cea originală și cea în care ați eliminat zgomotul.

Dacă valoarea obținută pentru imaginea *denoised* este **mai mare** decât pentru cea *noisy*, metoda și-a îndeplinit scopul.