Tema: Clasificare de sunete folosind filtre Gabor

Andrei Nicolicioiu
andrei.nicolicioiu@gmail.com

Publicat: 2 noiembrie 2020.
Update: -
Termen Limită: **17 noiembrie 2020**

1 CLASIFICARE

Clasificarea de sunete, adică prezicerea tipului unui sunet este foarte utilă în diferite domenii [1; 2]. În această temă vom implementa o metodă care va clasifica sunete din mai multe categorii ('Lătratul unui câine', 'Valuri', 'Ploaie' și altele), folosindu-ne de datale din subsetul ESC10 din datasetul [3]

2 TRASĂTURI

În orice metodă de clasificare avem nevoie de un set de trăsături (feature-uri) care caracterizează o entitate și fac posibilă distingerea între doua entități diferita. De exemplu, pentru diferențierea imaginilor cu căpșuni de imagini cu banane, culoarea va fi o bună trăsătură. Pentru distingerea sunetelor este foarte important spectrul lor de frecvențe, așa că vom încerca să ne construim niște feature-uri care surprind bine spectrul sunetelor. De exemplu, în plânsul unui copil există frecvențe mai mari decât există in sunetul valurilor.

La curs am învățat să analizăm spectrul unui semnal cu ajutorul Transformatei Fourier. In cadrul acestei teme vom proiecta o serie de filtre în domeniul timpului care au un comportament dorit în domeniul frecvențelor.

2.1 FILTRUL GABOR

Filtrele Gabor sunt folosite de obicei pentru analiza semnalelor bidimensionale precum imaginile, însă noi le vom folosi în varianta unidimensională. Ele sunt folosite pentru a găsi regiuni locale dintr-un semnal care au anumite frecvențe.

Vom învăța mai multe despre filtrare în cursurile și laboratoarele următoare. O operație de filtrare liniară a unui semnal x este definită în felul următor:

$$y(n) = h(0)x(n) + h(1)x(n-1) + ... + h(1)x(n-K)$$

$$y(n) = \sum_{k=0}^{K-1} h(k)x(n-k) \tag{1}$$

Aceasta mai poartă numele de convolutie:

$$y = h * x \tag{2}$$

Operația este definită de elementele filtrului h. În funcție de valorile lui h putem defini diferite tipuri de filtrări. Unul dintre cele mai polulare filtre îl reprezintă filtrul Gaussian, definit în felul următor:

$$g(k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(k-\mu)^2}{2\sigma^2}} \tag{3}$$

Acesta este definit de media μ și deviația standard σ care controlează locația respectiv lățimea filtrului.

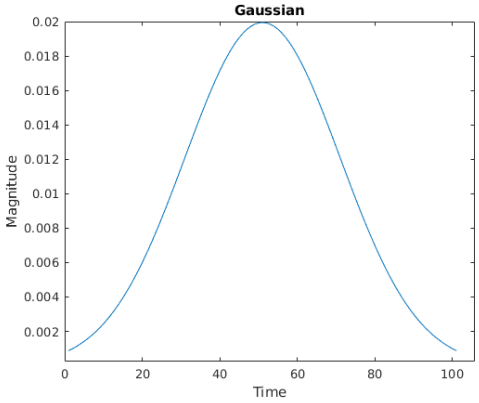

Figure 1: Gaussian Filter.

Filtrul Gaussian este un filtru trece-jos, lasă sa treacă frecvențele joase nealterate iar frecvențele înalte sunt amortizate. Ne dorim însă un filtru care să răspundă la anumite frecvențe.

Filtrul Gabor este contruit să răspundă la semnale având frecvențe în jurul unei valori f_0 date. Deci este un tip de filtru trece-bandă. Este contruit prin înmulțirea unul filtru Gaussian cu un semnal sinusoidal de o anumită frecvență f_0.

$$b(n) = g(n)\cos(2\pi f_0 n) \tag{4}$$

Modulând filtrul gaussian folosind functiile $\cos$ și $\sin$ obținem 2 filtre ortogonale. Ne rezumăm în descriere la semnalul definit cu funcția $\cos$, celalalt fiind definit echivalent.

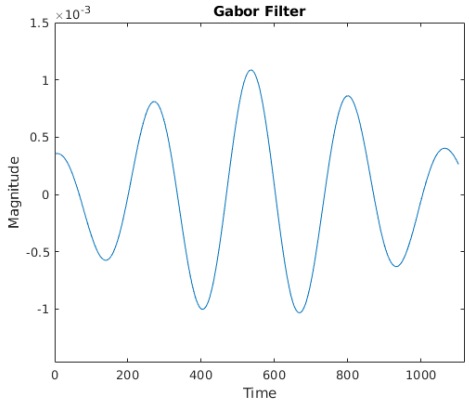

Figure 2: Filtru Gaussian.

2.2 Trăsături din filtre Gabor

Putem caracteriza un semnal după răspusul său la diferite filtre Gabor. Ne construim întâi un set de filtre (*filter bank*), definite printr-un set de frecvențe $\{f_0, f_1, ..., f_M\}$. Obținem M semnale filtrate:

$$\begin{aligned} y_1 &= b_1 * x \\ y_2 &= b_2 * x \\ &... \\ y_M &= b_M * x \end{aligned} \tag{5}$$

Semnalul de iesire y poate fi caracterizat în fiecare punct n de cele M activări ale filtrelor: $o(n) = [y_1(n), y_2(n), ...y_M(n)]$. La fiecare pas de timp n, iesirea va fi caracterizată de vectorul de trasături (*features*) $o(n)$. Cel mai simplu mod de a caracteriza întreg semnalul de iesire, va fi să calculăm media peste timp a acestor trăsături, obținând un singur vector de dimensiune M pe care îl putem folosi pentru a putea analiza întreg semnalul.

3 ANTRENARE

Pentru fiecare sunet din baza de date, ar trrebui să obținem în modul descris un vector de trasături. Baza de date folosită de noi are 10 tipuri de sunete iar noi vom folosi o metodă de învățare automată (*machine learning*) pentru a le putea distinge.

În general se folosesc 2 subseturi distincte de date. Primul este setul de învățare (antrenare) pe care îl folosim ca să analizăm datele și sa ne antrenăm un clasificator care să diferențieze cât mai bine datele. Pentru ca vrem ca un clasificator să generalizeze, adică să poată face predictii corecte despre fisiere audio pe care nu le-a văzut la antrenare, vom măsura performanța clasificatorului pe un alt subset din date, denumit setul de testare.

In scheletul temei veți folosi direct o metoda implementată in pachetele octave.

4 IMPLEMENTARE

În implementarea temei veți pleca de la un schelet de cod, pe care va trebui să îl urmați.

Install Instalați octave:

```
sudo apt-get install liboctave-dev
```

Instalați pachetul octave *nan* din octave command prompt. Aceasta ne oferă access la un classificator de tip Linear discriminant analysis (LDA) implementat în funcția *train_sc*.

```
pkg install nan -forge
pkg load nan
```

5 CERINTE

1. Implementare filtru Gabor [3 puncte]. Implementati o functie care crează un filtru Gabor de domensine, deviatie standard și frecvență date. Creați 3 variante de filtre, modulând filtrul gaussian cu urmatoarele funcții sinusoidale: $\cos(x)$ și $\sin(x)$ și sinusoidala complexă $e^{-ix} = \cos(x) + i\sin(x)$.

```
function [complex_h, cos_h, sin_h] = gabor_filter(size, sigma, freq)
```

2. Creați un set de filtre [1 punct]. Creați un set de $M = 12$ filtre Gabor folosind următorii parametrii:

```
    size  = 1102

    freq = [0.0027, 0.0089, 0.0173, 0.0284, 0.0433, 0.0632, 0.0898,
    0.1254, 0.1730, 0.2365, 0.3215, 0.4350]

    sigma = [187.2109  140.0663  104.7939, 78.4041, 58.6599, 43.8878,
    32.8357, 24.5668, 18.3803, 13.7516, 10.2886, 7.6977]
```

Afisați filtrele de tip $\cos$ și $\sin$ cu primii parametri.

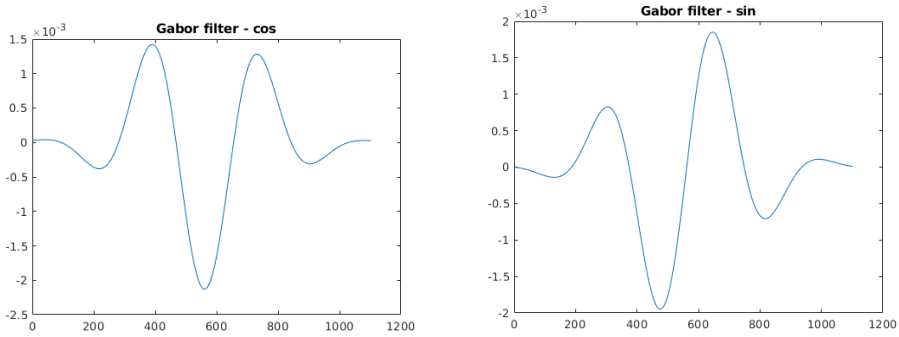

Figure 3: Set de Filtre Gabor.

3. Afișați spectrul filtrelor [1 punct]. Calculati folosind Transformata Fourier Discretă (implementată ca Fast Fourier Transform - fft) spectrul fiecarul filtru Gabor definit cu funcția cos și funcția sin. Afisați magnitudinea spectrului corespunzator frecvențelor pozitive, aflat în prima jumătate a răspunsului dat de funcția *fft*.

```
    coefs = fft(cos_h);
```

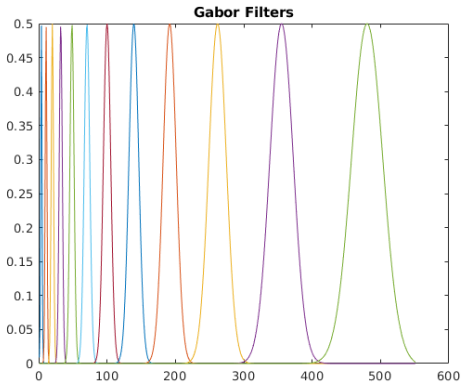

Figure 4: Spectru Set de Filtre Gabor.

4. Convoluție [1 punct]. Putem folosi direct filtrele create pentru a procesa inputul precum este prezentat in Sectiunea 2.2. Fom folosi doar filtrele de tip sin pentru acest task. Asta presupune aplicarea unei convolutii dintre fiecare fisier de input și fiecare filtru din setul de filtre de tip sin. Putem folosi direct funcția *conv* din Octave.

Pentru fiecare fisier audio, grupând rezultatele celor M filtre, vom obține un rezultat de dimensiune (aproximativ) $N \times M$, unde N este numărul de esantioane ale fisierului audio. Pentru a obține un descriptor al fisierului, vom calcula media si deviatie standard pe cei N pasi de timp. In final vom obține câte un vector de dimensiune $2M$ pentru fiecare fisier audio, pe care îi returna din funcția *get_features* din scheletul de cod.

5. Filtrare rapidă [4 puncte]. Operația de filtrare dată de convoluție este oarecum ineficientă pentru că are multe operații redundante. Pentru că sunetul este esantionat foarte des, rezultatul filtrării a unor esantioane aproapiate este apriximativ acelasi. Asa că vom aplica operația de filtrare definită de Equația 1 doar în anumite puncte. Această operație se mai numește și convoluție cu stride.

5.a Creare ferestre [1 punct]. Un fisier audio contine N esantioane, unde N este de ordinul 10^6. Vom selecta un numar de $F << N$ ferestre, adica grupuri consecutive de esantioane, fiecare avand

K esantioane. Stiind ca fisierele audio sunt esantionate la frecvența f_s, creăm ferestele astfel încât să corespundă la 25 ms cu 10 ms distanță între ele.

5.b Filtrare ferestre [3 punct]. Vom aplica Equația 1 pentru fiecare fereastră. Vom folosi filtre de tip $\sin$ cu aceeași dimensiune K cu a ferestrelor create. Prin această operație, din fiecare fereastră obținem un scalar. Putem observa ca această operație se reduce la un produs scalar între fereastră și filtrul inversat. De asemenea putem observă ca filtrul este simetric și inversarea este redundantă.

Aplicănd această operație pentru toate elementele din dataset vom obține pentru fiecare fisier un tensor o_c de dimensiune $F \times M$.

Repetăm aceleași operații, dar cu seturile de filtre $\sin$ si sinusoid complex pentru a obține o_s si o_e.

Calculați $o = \sqrt{o_c^s + o_s^2}$ si afisati media diferenței dintre aceasta și $|o_s|$ [1 punct]. Ar trebui ca cele 2 să fie egale, aratând ca filtrarea cu 2 filre $\cos$ si $\sin$ e echivalentă cu filtrarea cu filtrul sinusoidal complex echivalent $o = o_s$.

La fel ca la subpunctul precedent, pentru fiecare fisier audio, putem calcula media și deviația standard a reprezentarii o de dimensiune $F \times M$ obținand un vector de dimensiune $2M$ folosit în funcția *get_features* din scheletul de cod.

Folosind metoda descrisă, putem obține un clasificator cu acuratețe de aproximativ $55 - 70\%$.

[Bonus] Încercați să obțineți o performanță cât mai bună prin varierea numarului de filtre folosite și a parametrilor filtrelor.

REFERENCES

[1] Jens Schröder, Jorn Anemiiller, and Stefan Goetze. Classification of human cough signals using spectro-temporal gabor filterbank features. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6455–6459. IEEE, 2016.

[2] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[3] Karol J Piczak. Esc: Dataset for environmental sound classification. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 1015–1018, 2015.