Tema: Clasificare de sunete folosind filtre Gammatone

Andrei Nicolicioiu
andrei.nicolicioiu@gmail.com

Publicat: 8 noiembrie 2021.
Update: 12 noiembrie. Clarificare ex 2.
Update: 13 noiembrie. Clarificare 5.b: return get_features.
Update: 20 noiembrie. Clarificare 5.b: aplicăm modul inainte de a calcula media.
Update: 20 noiembrie. Modificare în acuratețea asteptată.
Termen Limită: **21 noiembrie 2021 ora 23.59**

1 CLASIFICARE

Clasificarea unui sunet poate ajuta in diverse domenii [1; 2], de la identificarea genului unei melodii, la diferentierea intre tuse asociate diferitelor boli. În această temă vom implementa o metodă care va clasifica sunete din mai multe categorii ('Lătratul unui câine', 'Valuri', 'Ploaie' și altele), folosindune de datasetul ESC10 [3].

2 TRASĂTURI

În orice metodă de clasificare avem nevoie de un set de trăsături (eng. features) care caracterizează o entitate și fac posibilă distingerea între doua entități diferite. De exemplu, pentru diferențierea imaginilor cu căpșuni de imagini cu banane, culoarea va fi o bună trăsătură. Bineinteles, cu cât problema este mai grea, cu atât avem nevoie de trăsături mai complexe. 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 și ne putem folosi de acest lucru pentru a le distinge.

La curs am învățat să analizăm spectrul unui semnal cu ajutorul Transformatei Fourier. In cadrul acestei teme vom incerca să caracterizăm spectrul unui sunet în funcție de cum răspunde la diverse filtre. Vom proiecta o serie de filtre în domeniul timpului care au un comportament dorit în domeniul frecvențelor.

2.1 FILTRUL GAMMATONE

Filtrele Gammatone sunt folosite pentru a simula un proces similar cu cele petrecute în urechea umană [4] și sunt folosite în diverse aplicații audio [5]. Un filtru Gammatone este caracterizat in principal de o frecvență centrală și de o lățime de bandă la care răspunde. Precum o sa vedem în figura 2 filtrele Gammatone corespunzătoare frecvențelor joase au o lățime de bandă mică, iar cele înalte au o lățime de banda mai mare. Acest principiu este realizat pentru că in general, vrem să putem distinge mai fin frecvențele joase, pe când la frecvențele înalte nu avem nevoie de atât de mare precizie.

Vom învăța mai multe despre filtrare în cursurile și laboratoarele următoare. Una din cele mai simple operații de filtrare a unui semnal x cu un filtru h este filtrarea liniară, definita în felul următor:

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

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

Spunem ca semnalul de iesire y este obținut după de am filtrat semnalul de intrare x cu filtrul h.

Aceasta operație 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: trece-jos, trece-sus, trece-bandă.

Filtrul Gammatone este contruit să răspundă la semnale având frecvențe în jurul unei valori f_c date. Deci este un tip de filtru trece-bandă. Pentru a filtra un semnal esantionat la frecvența f_s filtrul este construit in felul următor în domeniul timp:

$$h(n) = at^3 e^{-2\pi bt} \cos(2\pi f_c t); \tag{3}$$

$$t = \frac{n}{f_s} = nT_s$$
$$a = f_c^{2.5}$$
$$b = 1.019 * 24.7 * (4.37 * \frac{f_c}{1000} + 1)$$

unde presupunum ca avem semnale esantionate la frecventa f_s iar esantionul n corespunde timpului t. Lățimea de bandă scade pe măsura ce frecvența creste și este controlata de parametrul b.

În figura 1 putem vedea un filtru corespunzător frecventei de 1000 Hz.

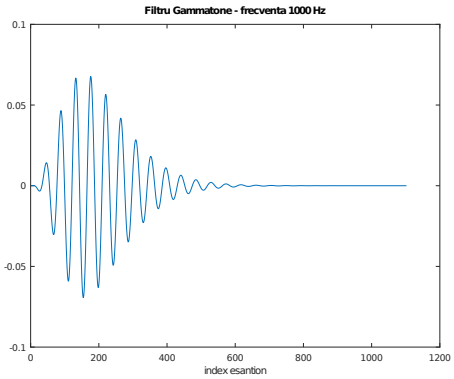

Figure 1: Filtru Gammatone corespunzător frecvenței de 1000 Hz.

2.2 TRĂSĂTURI DIN FILTRE GAMMATONE

Putem caracteriza un semnal după răspunsul său la diferite filtre Gammatone. 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:

$$\begin{aligned} y_1 &= b_1 * x \\ y_2 &= b_2 * x \\ &... \\ y_M &= b_M * x \end{aligned} \tag{4}$$

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 trebui 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 Gammatone [3 puncte]. Implementati o functie care crează un filtru Gammatone de dimensine, frecvență date. Urmăriți ecuația 3 unde n va lua valori intre 1 si dimensiunea dorita. Creați funcția care returneaza un filtru având **exact** următorul antet:

```
function h = gammatone_filter(size, fc, fs)
```

2. Set de frecvențe. [1 punct]. Vom construi un set de filtre centrate intr-un set de frecvențe egal depărtate pe scala Bark. Pentru a trece din scala normală in scala Bark folosim transformarea:

$$f_b = 6 * \text{asinh}(f/600);$$

și invers, din scala Bark in scala normală:

$$f = 600 * \sinh(f_b/6);$$

unde sinh si asinh sunt funcțiile sin hiperbolic si inversa sa (arcsin hiperbolic).

Creați o funcție care calculeaza M+1 frecvente egal distribuite pe scala Bark, începând cu zero, până la $\frac{f_s}{2}$ si returnează pe primele M. Folosiți următorul antet:

```
function freqs = get_equal_bark_freqs(M,fs)
```

3. Creați un set de filtre [1 punct]. Creați un set de $M = 12$ filtre Gammatone folosind frecvențele create la punctul precedent. Acestea ar trebui sa aiba dimensiunea egala cu numarul de esantioane din 25 ms de fisier audio esantionat cu f_s. Apoi creați și afișați un filtru corespunzător frecvenței $f_c = 1000$ Hz. Ar trebui să arate precum cel din Figura 1.

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

```
    coefs = fft(h);
```

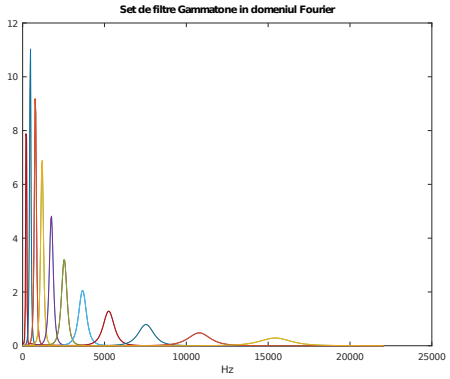

Figure 2: Spectrul setului de Filtre Gammatone.

5. Filtrare rapidă [4 puncte]. Pentru a filtra semnalul cu fiecare filtru din set, trebuie sa aplicam Ecuația 1 in fiecare punct al semnalului de intrare. Dar pentru că sunetul este esantionat foarte des, rezultatul filtrării a unor esantioane aproapiate este aproximativ același. Așa că vom aplica operația de filtrare definită de Ecuația 1 doar în anumite puncte.

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 Ecuația 1 pentru fiecare fereastră. Vom folosi filtrele aceeași dimensiune K cu a ferestrelor create. Practic aplicam Ecuația 1 o singură dată pentru $n = K$ obținând câte un singur scalar pentru fiecare fereastră.

Putem observa ca această operație se reduce la un produs scalar între fereastră și filtrul inversat. Pentru o implementare mai eficientă putem să efectuam toate produsele scalare dintre toate ferestrele corespunzatoare unui fisier audio si toate filrele printr-o singură înmultire de matrici.

Aplicând această operație pentru toate elementele din dataset vom obține pentru fiecare fisier o matrice o de dimensiune $F \times M$. Aplicăm modul peste aceasta matrice. Pentru fiecare fisier audio, vom calcula media și deviația standard a reprezentării o de dimensiune $F \times M$ obținand un vector de dimensiune $2M$ care trebuie returnat de funcția *get_features* din scheletul de cod. De asemenea această funcție trebuie să returneze și setul de M filtre de dimensiune K folosite, reprezentate ca o matrice $K \times M$.

```
[filters, feat_train] = get_features(audio_train, fs, plot_figs)
```

Folosind metoda descrisa, putem obține un clasificator cu acuratețe de aproximativ 40 - 65% pe setul de test, folosind toate datele.

6 PREDARE TEMĂ

În implementarea temei nu modificați scheletul de cod și implementați funcțiile cerute în enunt exact cu antetul dat. Codul va fi însoțit de un fisier readme de câteva rânduri în care să prezentați structura implementării (ex. ce face fiecare funcție) și câteva detalii de implementare. Pentru a nu rata ceva la corectare, vă rugăm să menționați în readme câte cerințe ați rezolvat și ce acuratețe ați obținut.

Uploadați o arhivă denumita nume_prenume_grupa.zip pe curs.upb.ro până la data mentionată mai sus. Arhiva nu trebuie să conțină și fisierele de date (data.mat).

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.

[4] Yang Shao, Zhaozhang Jin, DeLiang Wang, and Soundararajan Srinivasan. An auditory-based feature for robust speech recognition. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4625–4628. IEEE, 2009.

[5] Jun Qi, Dong Wang, Ji Xu, and Javier Tejedor Noguerales. Bottleneck features based on gammatone frequency cepstral coefficients. In *Interspeech*. International Speech Communication Association, 2013.