Tema: Clasificare și detecție de sunete

Autor: Andrei Nicolicioiu
Publicat: 03.11.2024.

Termen Limită: **17.11.2024 ora 23.59**

1 SCOP

În această temă vom vedea câteva operații de bază pentru analiza semnalelor, în special pentru analiza semnalelor de tip audio. Vom vedea cum putem folosi filtrarea de sunete pentru a putea distinge diferite sunete.

2 CLASIFICARE

Clasificarea unui sunet constă în determinarea categoriei din care face parte acest sunet (ex. ce reprezintă acest sunet? ex. zgomot de valuri? voce umană? lătrat de câine?). Aceste probleme apar frecvent în cazuri practice, de exemplu clasificarea unui sunet poate ajuta în diverse domenii [1; 2], de la identificarea genului unei melodii, la diferențierea între 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), folosindu-ne de setul de date ESC10 [3].

3 TRĂSĂTURI

Pentru a recunoaște semnale mai complexe, cum ar fi sunetele naturale, nu putem folosi direct semnalul direct (raw), ci avem nevoie să îl procesăm / transformăm într-o formă mai ușor de prelucrat. Pentru aceasta va trebui să extragem un set de trăsături (eng. features) care caracterizează un semnal și fac posibilă distingerea între două semnale de tip diferit. De exemplu, pentru diferențierea imaginilor cu căpșuni de imagini cu banane, culoarea va fi o trăsătură bună. Bineînțeles, 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 trăsături care surprind bine spectrul sunetelor. De exemplu, în plânsul unui copil există frecvențe mai mari decât există în 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. În cadrul acestei teme vom încerca să caracterizăm spectrul unui sunet în funcție de cum răspunde la diverse filtre trece-bandă care surprind aspecte date de anumite zone din spectrul de frecvențe.

3.1 FILTRAREA

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. Pentru un semnal de intrare (vector de dimensiune $[1 \times N]$) și un filtru (vector de dimensiune $[1 \times K]$), operația de filtrare liniară este definită în felul următor:

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

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

Spunem că semnalul de ieșire y este obținut după ce am filtrat semnalul de intrare x cu filtrul h. Observăm că dacă încercăm să calculăm valoarea lui y(0) avem nevoie de valoarea elementelor $x(-1), x(-2), \ldots, x(-K+1)$. Pentru a avea o operație validă, putem considera ieșirea ca începând la $n = K - 1$, iar rezultatul obținut este un vector y de dimensiune $[1 \times (N - K + 1)]$).

Această operație mai poartă numele de ***convoluție***: $y = h * x$

Operația de filtrare este definită de elementele filtrului h. În funcție de valorile lui h putem defini diferite tipuri de filtrări: trece-jos, trece-sus, trece-bandă. În continuare vom vedea exemple de filtre utile.

3.2 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.

Unul dintre cele mai simple filtre îl reprezintă filtrul Gaussian, definit în felul următor:

$$g_{\mu,\sigma}(n) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(n-\mu)^2}{2\sigma^2}} \tag{2}$$

Acesta este definit de media μ și deviația standard σ care controlează locația respectiv lățimea filtrului. Pentru un filtru de lungime $size = S$ vom considera $n \in \{0...S-1\}$. Definim un filtru gaussian g_σ cu deviația standard σ și având lungimea $size = S$, ca fiind centrat in mijlocul ferestrei, adică media sa este $\mu = S/2$ ca: $g_\sigma(n) = g_{S/2,\sigma}(n)$ pentru $n \in \{0...S-1\}$. Figura 1 ilustrează un filtru gaussian de lungime $S = 101$.

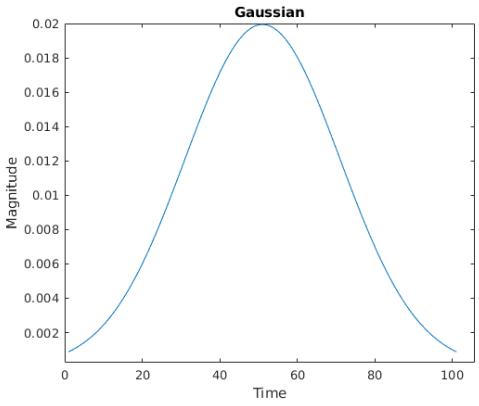

Figure 1: Gaussian Filter.

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 un anumit interval de 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 unui filtru Gaussian cu un semnal sinusoidal de o anumită frecvență f_0. Pentru un filtru de lungime S:

$$b_{f_0,\sigma}(n) = g_\sigma(n)\cos(2\pi f_0 n), \qquad \forall n \leq S \tag{3}$$

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

3.3 Trăsături obținute folosind un set de 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 $\{f_0, f_1, ..., f_M\}$. Dacă filtrăm un semnal

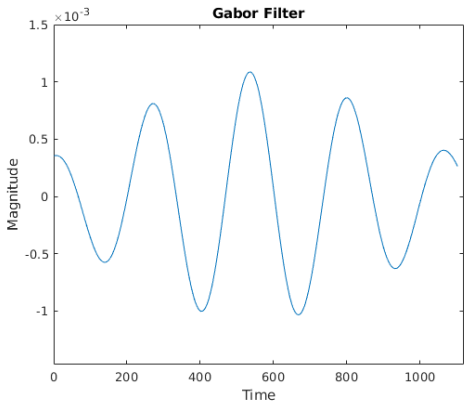

Figure 2: Filtru Gabor.

cu un filtru Gabor (trece-bandă) de o anumită frecvență, atunci vom păstra în semnalul de ieșire predominant frecvențe apropiate de frecvența aleasă. Dacă semnalul rezultat e suficient de puternic, atunci putem spune că acea frecvență este importantă pentru semnal. Putem repeta același lucru pentru mai multe frecvențe. De exemplu, dacă avem un set de M filtre ($\{b_1, b_2, ...b_M\}$) trece-bandă, fiecare centrat în altă frecvență, putem filtra un semnal de intrare x de M ori.

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

Putem concatena toate semnalele rezultate, punând fiecare semnal y_m pe linia m a unei matrici Y de dimensiune $[M \times (N - K + 1)]$.

La fiecare pas de timp n, semnalul de ieșire $Y(n)$ va fi caracterizat de cele M activări ale filtrelor: $Y(n) = [y_1(n), y_2(n), ...y_M(n)]$. Acest vector $Y(n)$ poartă denumirea de vector de trăsături (eng. *features*). Cel mai simplu mod de a caracteriza întreg semnalul de ieșire, 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.

4 Implementare

În implementarea temei veți pleca de la un schelet de cod, pe care va trebui să îl urmați. Toate figurile generate trebuie salvate ca *id_nume_figura.png* unde *id=nume_prenume_grupa*.

Scheletul de cod încarcă datele precum și valoarea frecvenței de esantionare f_S.

5 Cerințe

1. Implementare filtru Gabor [3 puncte]. Implementați o funcție care creează un filtru Gabor de dimensiune, deviație standard și frecvență date. Asemănător cu Ecuația 3, creați 2 variante de filtre, modulând filtrul gaussian cu următoarele funcții sinusoidale: $\cos(x)$ și $\sin(x)$.

```
def gabor_filter(size, sigma, freq):
    ...
    return cos_h, sin_h
```

2. Creați un set de filtre [1 punct]. Vom crea un set de filtre Gabor centrate în anumite frecvențe. Pentru aceasta vom folosi scala Mel [1], care este concepută astfel încât frecvențele egal depărtate pe scala Mel, sunt percepute de oameni ca egal depărtate auditiv. Vom crea parametrii filtrelor în felul următor: creăm $M = 12$ segmente egale între frecvențele A și B de pe scala Mel corespunzătoare 0 Hz și $f_S/2$ pe scala normală. In scala normală, acestea vor avea segmente corespunzătoare, cu centre in $\{c_1, c_2, ..., c_M\}$, iar lungimea fiecărui segment, pe scala normală va fi $\{l_1, l_2, ..., l_M\}$. Creați M filtre folosind funcția gabor_filter(size, σ_i, f_i), size= 1102.

Pentru a converti din scala normală în scala Mel folosiți:

$$f_{mel} = 1127 * ln(1 + \frac{f}{700}) \tag{5}$$

$$f = 700\left(e^{\frac{f_{mel}}{1127}} - 1\right) \tag{6}$$

În figura 3 putem observa o mapare între scala Mel și scala normală pentru câteva valori. Va trebui să faceți această mapare pentru a obține valorile $\{c_1, c_2, ..., c_M\}$ (exemplificate cu verde in figură) și $\{l_1, l_2, ..., l_M\}$.

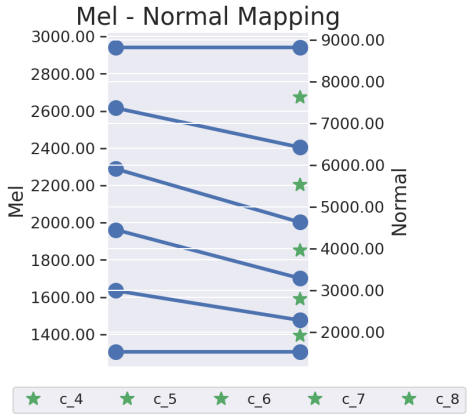

Figure 3: Mapare intre scala mel si scala normala.

În cazul semnalelor discrete, ne raportăm mereu la frecvența de esantionare f_S. Așa că frecvențele și deviațiile standard necesare pentru a crea filtrele Gabor vor fi: $f_i = \frac{c_i}{f_S}$ și $\sigma_i = \frac{f_S}{l_i}$.

Salvați filtrele de tip cos și sin folosind parametrii corespunzători primului segment ($f_i = 0.00267$, $\sigma_1 = 187.21221$) folosind denumirile *id_gabor_cos.png* și *id_gabor_sin.png*.

3. Afișați spectrul filtrelor [1 punct]. Calculați folosind Transformata Fourier Discretă (implementată ca Fast Fourier Transform - fft) spectrul fiecărui filtru Gabor definit cu funcția cos și funcția sin. Salvați o figură (*id_spectru_filtre.png*) cu magnitudinea spectrului corespunzător frecvențelor pozitive, aflat în prima jumătate a răspunsului dat de funcția *fft*. Rezultatul trebuie să arate precum Figura. 5.

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

4. Filtrare rapidă. Pentru a filtra semnalul cu fiecare filtru din set, trebuie să aplicăm Ecuația 1 în fiecare punct al semnalului de intrare. Dar pentru că sunetul este eșantionat foarte des, rezultatul filtrării a unor eșantioane apropiate este aproximativ același. Așa că vom aplica operația de filtrare definită de Ecuația 1 doar în anumite puncte, precum vom vedea în continuare.

[1] https://en.wikipedia.org/wiki/Mel_scale

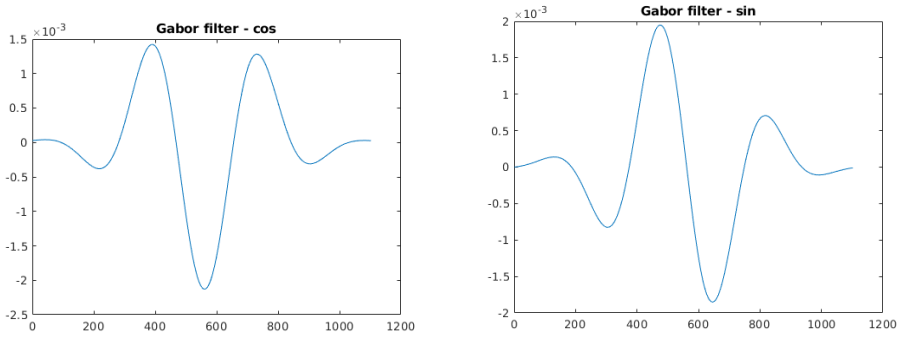

Figure 4: Perechea de filtre Gabor de aceeași frecvență, definite cu funcțiile cos și sin.

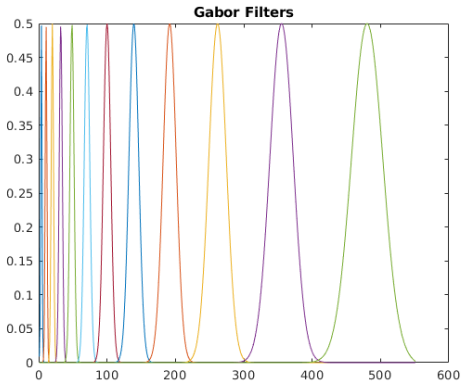

Figure 5: Spectrul Setului de Filtre Gabor.

4.a Creare ferestre [2 puncte]. Un fisier audio conține N eșantioane, unde N este de ordinul 10^6. Vom selecta un număr de $F << N$ ferestre, adică grupuri consecutive de eșantioane, fiecare având K eșantioane. Alegem numărul de eșantioane K să fie egal cu dimensiunea filtrelor create anterior.

Știind că fisierele audio sunt eșantionate la frecvența f_S, creăm ferestre de dimensiune K cu 12ms distanță între două ferestre.

4.b Filtrare ferestre [2 puncte]. Vom aplica Ecuația 1 pentru fiecare fereastră. Vom folosi filtrele cu aceeași dimensiune K cu a ferestrelor create. Practic aplicăm Ecuația 1 o singură dată pentru $n = K - 1$ obținând câte un singur scalar pentru fiecare fereastră.

Putem observa că această operație se reduce la un produs scalar între fereastră și filtrul inversat. Pentru o implementare mai eficientă putem să efectuăm toate produsele scalare dintre toate ferestrele corespunzătoare unui fișier audio și toate filtrele printr-o singură înmulțire de matrici. Pentru un fișier audio, mulțimea de F ferestre de dimensiune K poate fi reprezentată de o matrice de dimensiune $F \times K$. Mulțimea de M filtre este reprezentă de o matrice de dimensiune $M \times K$. Puteți folosi aceste matrici pentru a efectua operația de filtrare.

Vom obține astfel, pentru fiecare fișier audio o matrice $o \in \mathbb{R}^{F \times M}$ de dimensiune $F \times M$. Aplicăm modul peste această matrice. Pentru fiecare fișier audio, vom calcula rezultatul mediu al unei ferestre, precum și deviația standard. Astfel din matricea o de dimensiune $F \times M$ obținem un vector de dimensiune $2M$.

Implementați aceste operații în cadrul funcției *get_features* din scheletul de cod. Pentru fiecare fișier din input, trebuie să returnăm un vector de dimensiune $2M$, deci luând ca input o mulțime de D fișiere audio, rezultatul trebuie să fie o matrice de dimensiune $D \times 2M$.

```
def get_features(audio_train, fs):
```

```
    ...
    return feat_train
```

5. Clasificare [1 punct]. Folosind trăsăturile descrise mai sus, scheletul de cod implementează un clasificator simplu care poate diferenția între diferite tipuri de sunete. Acuratețea acestui clasificator pe setul de test trebuie să fie de aproximativ: $55 - 68\%$.

6 PREDARE TEMĂ

În implementarea temei nu modificați scheletul de cod și implementați funcțiile cerute în enunț exact cu antetul dat, fiecare într-un fisier separat. Codul va fi însoțit de un fișier 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 denumită nume_prenume_grupa.zip pe curs.upb.ro până la data menționată mai sus. Arhiva **nu** trebuie să conțină și fișierele 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.