
Tema: Clasificare și detecție de sunete

Autor: Andrei Nicolicioiu

Publicat: 8 noiembrie 2022.

Update: **13 noiembrie**. Modificare ecuațiile 1 și 3. Adăugare teste de verificare ex. 1 și 5.

Termen Limită: **21 noiembrie 2022 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 operația de *corelație* pentru a găsi și pentru a alinia diferite șabloane sau tipare (eng. patterns) din semnale și cum putem folosi această operație pentru a identifica anumite tipuri de sunete.

2 CLASIFICARE ȘI DETECȚIE

Clasificarea unui sunet constă în determinarea categoriei din care face parte acest sunet (ex. ce e acest sunet?). Detecția unui sunet constă în determinarea locației unde se află un anumit sunet într-un semnal mai îndelungat (când începe acest sunet). Ambele 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 datasetul ESC10 [3].

3 CORELAȚIA

O operație foarte utilă în procesarea de semnale este operația numită corelație¹. Pentru două semnale reale și discrete x și h , având același număr de elemente K , corelația este definită în felul următor:

$$y = (h \star x) = h(0)x(0) + h(1)x(1) + \dots + h(K-1)x(K-1)$$

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

Intuitiv, corelația ne arată cât de similare sunt două semnale. Pentru a obține o măsură validă, pe care o putem folosi pentru a compara similaritatea între diferite semnale, trebuie să folosim o normalizare.

$$y = (h \star x) = \frac{\sum_{k=0}^{K-1} h(k)x(k)}{\sqrt{\sum_{k=0}^{K-1} h^2(k)} \sqrt{\sum_{k=0}^{K-1} x^2(k)}} \quad (1)$$

¹<https://en.wikipedia.org/wiki/Cross-correlation>

În cazul general în care avem 2 semnale x , h de dimensiuni diferite $N > K$ operația de corelație este definită în mai multe puncte $n \in [1 \dots N - K + 1]$ între o parte a semnalului mare x și semnalul mic h .

$$y(n) = (h \star x)(n) = \sum_{k=0}^{K-1} h(k)x(n+k) \quad (2)$$

Atunci când vrem să folosim această operație pentru a calcula similaritatea între doua semnale, trebuie să folosim varianta normalizată:

$$y(n) = (h \star x)(n) = \frac{\sum_{k=0}^{K-1} h(k)x(n+k)}{\sqrt{\sum_{k=0}^{K-1} h^2(k)} \sqrt{\sum_{k=0}^{K-1} x^2(n+k)}} \quad (3)$$

unde $\mu_h = \frac{1}{K} \sum_{k=0}^{K-1} h(k)$ și $\mu_{x_n} = \frac{1}{K} \sum_{k=0}^{K-1} x(n+k)$ sunt mediile semnalelor h și respectiv a unei părți din x .

Putem folosi această operație pentru a "cauta" un semnal mic h într-un semnal mare x . Altfel spus, pentru fiecare locație (validă) din semnalul mare x vrem să aflăm cât de similar este h cu o bucată din x de la acea locație.

În Figura 1 putem vedea doua semnale x și h și rezultatul aplicării operației de corelație. Maximul corelație se atinge în dreptul punctului 300, ceea ce înseamnă ca la aceea locație x este foarte similar cu h (dacă mutăm semnalul h la aceea locație, cele doua semnale se vor suprapune perfect).

Același operație se poate folosi și pentru semnale mult mai complexe precum semnalele audio naturale pentru a cauta sunete similare. Însă pentru a realiza acest lucru avem nevoie de o reprezentare mai puternică a semnalului audio.

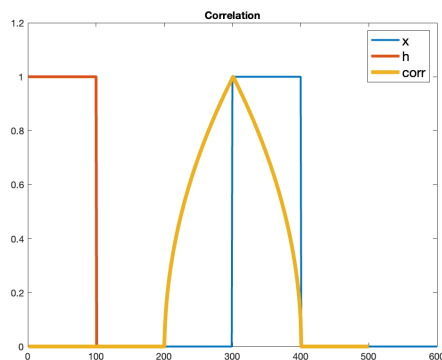


Figure 1: Corelație între semnalele x și h .

4 TRASĂTURI

Pentru a recunoaste 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 doua semnale de tip diferit. De exemplu, pentru diferențierea imaginilor cu căpșuni de imagini cu banane, culoarea va fi o trăsătură bună. 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 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.

4.1 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, definită în felul următor:

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

$$y(n) = (h * x)(n) = \sum_{k=0}^{K-1} h(k)x(n-k) \quad (4)$$

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

Aceasta operație mai poartă numele de *convoluție*. Putem observa că operația de convoluție este echivalentă cu operația de corelație dacă inversăm ordinea semnalului h .

$$h * x = \text{flip}(h) * x$$

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.

4.2 TRĂSĂTURI OBȚINUTE PRIN FILTRARE

Putem caracteriza un semnal după răspunsul său la diferite filtre. De exemplu, dacă avem un set de M filtre ($\{b_1, b_2, \dots, b_M\}$) trece-bandă, fiecare centrat în altă frecvență, putem filtra un semnal de intrare x de M ori.

$$\begin{aligned} y_1 &= b_1 * x \\ y_2 &= b_2 * x \\ &\dots \\ y_M &= b_M * x \end{aligned} \quad (5)$$

Semnalul de ieșire y poate fi caracterizat în fiecare punct n de cele M activări ale filtrelor: $o(n) = [y_1(n), y_2(n), \dots, y_M(n)]$. La fiecare pas de timp n , ieșirea va fi caracterizată de vectorul de trăsături (*features*) $o(n)$. 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.

5 METODA CELOR MAI APROPIAȚI K VECINI

Scopul nostru este să clasificăm sunete, adică să spunem din ce categorie / clasă face parte fiecare sunet. Pentru aceasta, vom folosi o bază de date de sunete, iar pentru fiecare sunet ar trebui să obținem un vector de trăsături folosind filtrarea descrisă anterior. Baza de date folosită de noi are 10 tipuri de sunete iar noi vom folosi o metodă simplă pentru a determina clasa unui semnal nou, pe baza similarității cu alte semnale cunoscute.

În general, când vrem să clasificăm o mulțime de date, folosim 2 subseturi distincte. Primul este setul de învățare (antrenare) care conține date despre care știm din ce clasă fac parte. Al doilea set, denumit setul de testare, va fi folosit pentru a măsura cât de bine funcționează o metodă de clasificare.

Presupunem că setul de învățare conține D_1 sunete, fiecare caracterizat de un semnal (vector) de dimensiune $2M$. La fel, setul de testare conține D_2 sunete, fiecare caracterizat de un vector de dimensiune $2M$.

Pentru a determina clasa unui sunet din setul de test, vom folosi metoda celor mai apropiați k vecini (eng. k -nearest neighbors). Aceasta constă în asignarea unei clase pentru un semnal nou h , din setul de test folosindu-ne de clasele celor mai apropiate (cele mai similare) semnale din setul de învățare. Aceasta presupune determinarea similarității dintre semnalul de test h și toate semnalele din setul de învățare, folosind Ecuația 1. Apoi alegem cele mai similare k semnale, cărora le știm clasele. Semnalul analizat h va primi clasa majoritară între cele k găsite anterior. Trebuie să repetăm acest procedeu pentru fiecare semnal din setul de test.

6 IMPLEMENTARE

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

7 CERINTE

1. Implementare corelație [2 puncte]. Implementați operația de corelație între 2 filtre x și h , precum este definită în Ecuația 3. Creați o funcție care implementează aceasta operație folosind **exact** următorul antet:

```
function y = my_corr(x, h)
```

Observație: Test necesar (dar nu suficient) pentru a vă asigura că funcția implementată este corectă

```
y = my_corr(h, h);  
% rezultat: y = 1. In acest caz rezultatul are o singura valoare pentru  
ca cele 2 semnale au aceeasi lungime)
```

2. Detectie [1 punct]. Fie semnalele $h \in \mathbb{R}^{100}$ de dimensiune 100, $h(n) = 1, \forall n \in [1, 100]$ și $x \in \mathbb{R}^{600}$ de dimensiune 600,

$$x(n) = \begin{cases} 1 & \text{pentru } n \in [300, 400] \\ 0 & \text{in rest} \end{cases}$$

Creați o funcție cu antetul de mai jos care primește două semnale x și h , calculează corelația dintre semnale cu funcția definită precedent, plotează rezultatul ca în Figura 1 și în final returnează indicele unde corelația este maximă.

```
function ind = detect(x, h, plot)
```

Plotați rezultatele doar dacă `plot=true`. Apelați această funcție pentru semnalele de mai sus.

3. Afișați spectrul filtrelor [1 punct]. În această temă vom folosi o mulțime de $M = 12$ filtre precalculate și încărcate în scheletul de cod, pe care le vom folosi apoi pentru a calcula trasături ale sunetelor. Afișați primele 2 filtre, care ar trebui să arate precum în următoarea figură.

```
figure, plot(filters(:,1));  
figure, plot(filters(:,2));
```

Calculați folosind Transformata Fourier Discretă (implementată ca Fast Fourier Transform - `fft`) spectrul fiecărui filtru. Afișați magnitudinea spectrului corespunzător frecvențelor pozitive, aflat în prima jumătate a răspunsului dat de funcția `fft`. Rezultatul trebuie să arate precum Figura 3.

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

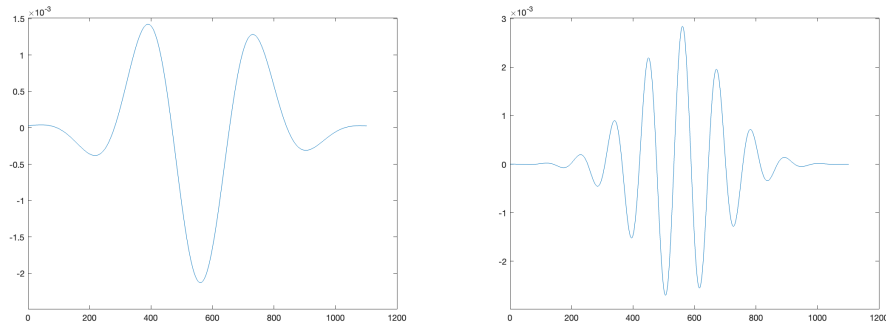


Figure 2: Primele 2 filtre din setul de filtre folosit pentru a calcula trăsături ale semnalelor audio.

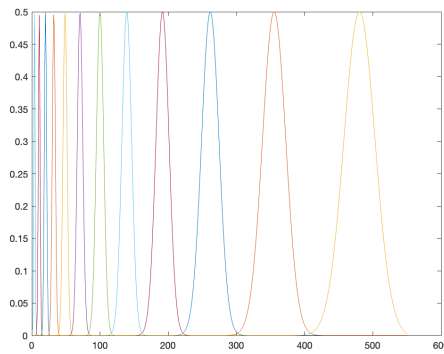


Figure 3: Spectrele fiecărui filtru din setul dat.

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

4.a Creare ferestre [1 punct]. Un fisier audio conține N esantioane, unde N este de ordinul 10^6 . Vom selecta un număr de $F \ll N$ ferestre, adică grupuri consecutive de esantioane, fiecare având K esantioane. Alegem numărul de esantioane K să fie egal cu dimensiunea filtrelor încărcate anterior.

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

4.b Filtrare ferestre [3 punct]. Vom aplica Ecuația 4 pentru fiecare fereastră. Vom folosi filtrele aceiași dimensiune K cu a ferestrelor create. Practic aplicăm Ecuația 4 o singură dată pentru $n = K$ 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 fisier audio și toate filtrele printr-o singură înmulțire de matrici.

Aplicând această operație pentru toate elementele din dataset vom obține pentru fiecare fisier o matrice $o \in \mathbb{R}^{F \times M}$ de dimensiune $F \times M$. Aplicăm modul peste această matrice. Pentru fiecare fisier audio, vom calcula media și deviația standard a reprezentării o de dimensiune $F \times M$ obținând un vector de dimensiune $2M$.

Implementați aceste operații în cadrul funcției `get_features` din scheletul de cod. Pentru fiecare fisier 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 $D \times 2M$.

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

5. Clasificare folosind cei mai apropiați k vecini [2 puncte]. Folosind trasăturile sunetelor din seturile de învățare și testare calculate anterior (`feat_train`, `feat_test`) estimați clasa fiecărui sunet din setul de test folosind metoda celor mai apropiați vecini, așa cum este descrisă în Secțiunea 5.

Implementați o funcție cu exact următorul antet care implementează algoritmul din Secțiunea 5.

```
function [similarities, predictions] = knn(labels_train, feat_train,
    feat_test, top_k)
```

Mai precis, `similarities` va conține similaritățile dintre toate sunetele caracterizate de `feat_test` și toate sunetele caracterizate de `feat_train` (în cazul nostru o matrice 100×300). Vectorul `predictions` va conține clasa fiecărui sunet corespunzător `feat_test`

Observație: Pentru o implementare rapidă, încercați să implementați Ecuația 3 pentru toate perechile de semnale din seturile de antrenare și testare folosind o singură înmulțire de matrici.

Observație: Test necesar (dar nu suficient) pentru a vă asigura că funcția implementată este corectă

```
[sim_train, pred_train] = knn(labels_train, feat_train, feat_train, 1);
% sim_train va conține pe diagonala principala valoarea 1 aproape mereu
% valoare media a elementelor de pe diagonala va fi aprox 0.98
% mean(diag(sim_train)) aprox 0.9833
```

Matricea `sim_train` va conține doar elemente 1 pe diagonala principală (există excepții pentru cazurile când semnalul inițial conține doar zero). Valorile de 1 de pe diagonala sunt obținute pentru că, în acest caz, reprezintă corelația între un semnal și el însuși (ex. primul element din diagonală este corelația între primul semnal de antrenare și el însuși).

Folosind metoda descrisă, putem obține un clasificator cu acuratete de aproximativ 35 – 60% pe setul de test, folosind toate datele.

8 PREDARE TEMĂ

În implementarea temei nu modificați scheletul de cod și implementați funcțiile cerute în enunț exact cu antetul dat. 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 acuratete ați obținut.

Uploadați o arhivă denumită `nume_prenume_grupa.zip` pe `курс.upb.ro` până la data menționată mai sus. Arhiva **nu** trebuie să conțină și fișierele de date (`data.mat` sau `filters.mat`).

REFERENCES

- [1] Jens Schröder, Jorn Anemüller, 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.