Separarea surselor

Motrescu Radu

Contents

1 Introducere in separarea surselor

Aceasta lucrare are scopul de a informa cititorul asupra unor metoda de analizare a componentelor unor date, fie acestea date economice sau imagini, pentru a putea fi interpretate mai usor, cat si metode de separare a componentelor unui set de date.

Algoritmii abordati sunt **PCA (Principal Component Analysis)**, **KPCA (Kernel Principal Component Analysis)** si **ICA (Independent Component Analysis)**.

O scurta descriere a cazurilor de aplicare a algoritmilor mentionati:

- PCA - Este folosit pentru a extrage informatiile cele mai relevante dintr-un set de date, adeseori cu dimensionalitate mare, cu scopul atat de a face interpretarea lor mai usoara, cat si pentru prelucrarea lor mai rapida.
- KPCA - Este o extensie a PCA, si acopera slabiciunile PCA, spre exemplu in seturi de date in care distributia valorilor nu poate fi analizata la fel de bine de PCA, si in cazuri de clustering.
- ICA - Aceasta abordare de analizare a componentelor are avantajul de le putea si separa, folosind atat unelte matematica cat si statistice, si este adeseori folosita in cazuri de separare a semnalelor de diferite tipuri, precum audio si video.

2 PCA

2.1 Libraria Accord.NET

Libraria Accord.Net contine clase pentru:

- Calcul stiintific: matematica, statistica si machine learning
- Procesare de imagini si semnale: imagini, semnale audio si recunoastere si urmarire faciala in timp real
- Biblioteci suport pentru controale specifice: histograme, scatter-plots, controale pentru fiecare clasa de procesare imagini si semnale

In contextul cerintelor pe care vrem noi sa le indeplinim, vom folosi clase dedicate matematicii, statisticii, si ale unor controale de afisare a datelor.

2.2 Ce este PCA?

PCA, sau Principal Component Analysis, sau pe romaneste, Analiza Componentelor Principale este o unealta matematica aplicata din algebra liniara.

Este o metoda non-parametrica (care nu depinde de statistici) de a extrage informatiile relevante dintr-un set de date complex sau confuz.

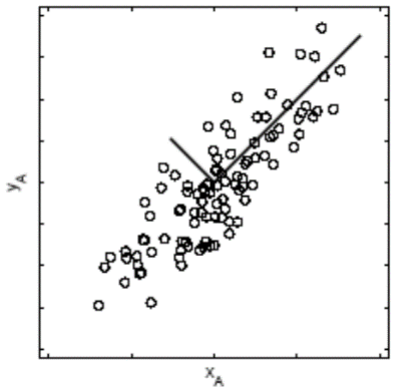
2.2.0.1 Un scurt exemplu: Sa presupunem ca am extras informatii pentru 100 de parametri pentru un student: inaltime, varsta, greutate, nota obtinuta la un test, culoarea parului etc. Vrem sa gasim cele mai importante caracteristici care definesc studentul. Cum facem asta? Folosim PCA pentru a selecta numai cele mai importante caracteristici.

- Ce parametri dorim sa indepartam:
  - Constanti: numarul de capete, care este 1 pentru toti studentii
  - Aproape constanti: grosimea firului de par: 0.003, 0.002, 0.0005 etc.
  - Care depind de alti parametri
- Ce parametri dorim sa pastram:
  - Care nu depind de alti parametri: culoarea ochilor
  - Care se schimba mult, au variatie mare: notele

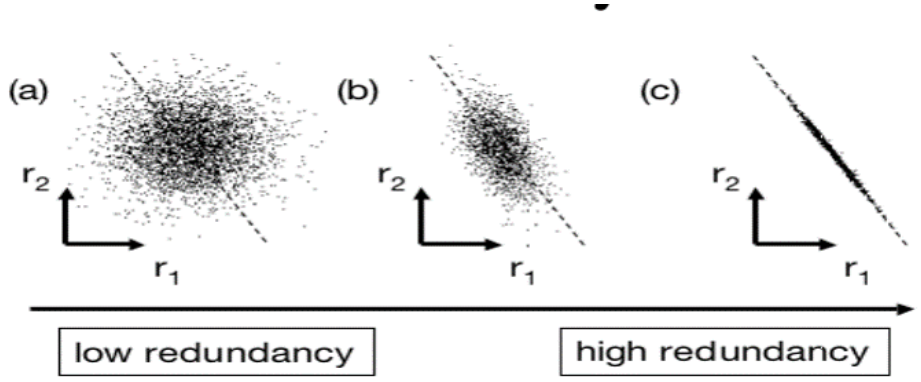
A putea elimina parametrii irelevanti si a-i pastra pe cei relevanti este usor pentru un om, el putand vedea clar care parametri nu exprima informatii relevante despre subiectul respectiv, dar cum putem face calculatorul sa isi dea seama de acesti parametri? Folosind matematica, desigur!

Dorim sa minimizam "sunetul de fundal" si redundanta datelor si sa maximizam variatia dintre parametri.

Se poate vedea in imaginea de mai jos maximizarea variatiei dintre axele norului de puncte respectiv.



In imaginile de mai jos se pot vedea inregistrarile unor informatii. In imaginea (a) si (b) se poate vedea cum informatiile nu sunt corelate, avand redunanta mica spre medie (ex: inaltimea unui student si media lui), iar in imaginea (c) se poate vedea o redundanta mare, insemnand ca ambii parametrii pot fi exprimati unul in functie de celalalt.



2.2.0.2 Variatia Este un mijloc de realizare a variabilitatii datelor dintr-un set de date cu media $\overline{X}$:

$$\sigma^2 = \frac{\sum_{i=1}^{n} \left(X_i - \overline{X}\right)^2}{n-1} \tag{1}$$

2.2.0.3 Covariatia Reprezinta variabilitatea fiecarei dimensiuni in relatie cu celelalte, si este masurata intre 2 dimensiuni pentru a se putea observa relatia dintre cele 2, spre exemplu numarul de ore studiate si nota obtinuta la examen.

$$\text{var}\left(X\right) = \frac{\sum_{i=1}^{n} \left(X_i - \overline{X}\right)\left(X_i - \overline{X}\right)}{n-1} \tag{2}$$

$$\text{cov}\left(X, Y\right) = \frac{\sum_{i=1}^{n} \left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{n-1} \tag{3}$$

2.2.0.4 Matricea de covariatie Consideram setul de date din care extragem valoarea medie (zero-mean data) si avand setul de vectori $\{x_1, x_2, ..., x_m\}$ care reprezinta liniile unei matrici $X_{m,n}$.

Fiecare linie a matricii reprezinta toate masuratorile unui anumit parametru, iar fiecare coloana reprezinta masuratorile care s-au intamplat la un moment dat.

De aici ajungem la definitia matricei de covariatie:

$$S_x \equiv \frac{1}{n-1} X X^T \text{ where } X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \tag{4}$$

S_x este o matrice simetrica $m \times m$, termenii de pe diagonala reprezentand variatia din acelasi parametru, iar termenii care nu sunt de pe diagonala reprezinta covariatia dintre parametri diferiti. Calculand S_x, cuantificam corelatia dintre toate posibilele perechi de masuratori. Observand elementele

din matrice, o cavariatie mare reprezinta un caz de redundanta mare, iar o covariatie egala cu 0 reprezinta date complet necorelate.

$$C = \begin{bmatrix} cov(X,X) & cov(X,Y) & cov(X,Z) \\ cov(Y,X) & cov(Y,Y) & cov(Y,Z) \\ cov(Z,X) & cov(Z,Y) & cov(Z,Z) \end{bmatrix} \tag{5}$$

2.2.0.5 Valorile si vectorii proprii Urmatorul pas in calcularea PCA este aflarea valorilor si a vectorilor proprii ale matricii de covariatie. Extragand aceste informatii, ele ne vor arata componentele principale ale setului de date: vectorul propriu cu cea mai mare valoare proprie este componenta principala a setului de date. Se obisnuieste sortarea vectorilor proprii in functie de valoarea proprie pentru a determina ordinea de semnificativitate.

Vectorii si valorile proprii reies din probleme de urmatoarea forma:

$$A.v = \lambda.v \tag{6}$$

A: matrice $m \times m$

v: vector $m \times 1$ nenul

λ: constanta

Pentru orice valoare a lui λ pentru care ecuatia are solutie se numeste valoarea proprie a lui A, si vectorul **v** care corespunde acestei valori se numeste vectorul propriu a lui A.

2.2.0.6 Pasul final PCA este sa aflam valorile finale ale setului de date. Aici avem optiunea sa ignoram o parte din dimensiunile pe care le avem, deoarece ele pot fi nesemnificative, avand valori proprii mici, sau dorim afisarea lor in 1, 2 sau 3 dimensiuni. Dupa stabilirea vectorilor proprii doriti, aflarea datelor finale este simpla, proiectam punctele in spatiul lor:

FinalData = RowFeatureVector $\times$ RowZeroMeanData

RowFeatureVector este matricea cu vectorii proprii transpusi, cu cel mai semnificati vector propriu pe prima linie.

RowZeroMeanData este matricea care contine setul de date initial din care s-a scazut valorea medie.

Matricea rezultata **FinalData** va contine setul de date dupa aplicarea algoritmului PCA.

2.3 Jurnalul dezvoltarii aplicatiei

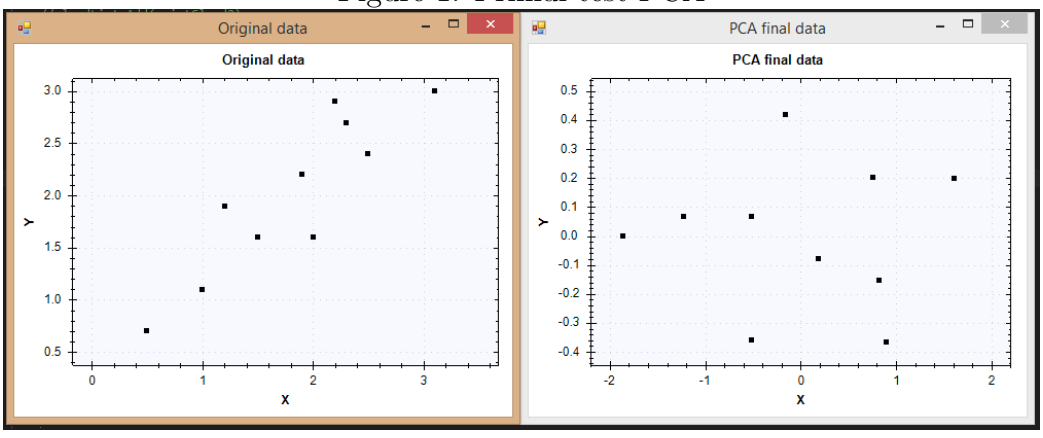
2.4 Aplicatia AccordPCA

Primii pasi pe care i-am facut in cadrul acestui proiect a fost implementarea algoritmului PCA. Acest lucru a fost realizat usor, folosind libraria Accord, mai specific, pentru a realiza diferitele operatii cu matrici.

```
public void Compute()
{
    mean = initialData.Mean(0);

    dataAdjusted = initialData.Subtract(mean, 0);
    covarianceMatrix = dataAdjusted.Covariance();

    evd = new EigenvalueDecomposition(covarianceMatrix);

    eigenvalues = evd.RealEigenvalues;
    eigenvectors = evd.Eigenvectors;

    eigenvectors = Matrix.Sort(eigenvalues, eigenvectors, new GeneralComparer(ComparerDirection.Descending, true));

    finalData = dataAdjusted.Dot(eigenvectors);

    // data2D = finalData.GetColumns(0, 1);

}
```

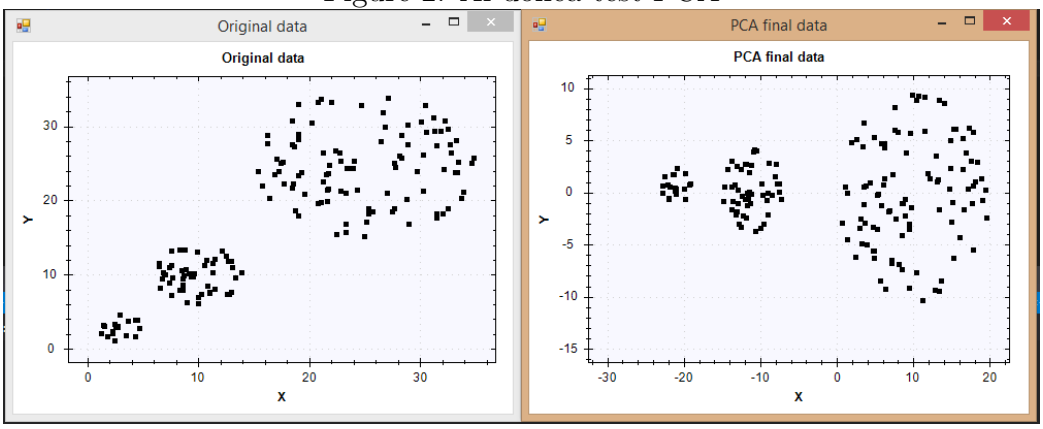
Pentru a testa acest algoritm, am introdus prima data un set de date deja testat de catre altcineva, unde se poate observa clar rezultatul algoritmului. (referinta catre http://dai.fmph.uniba.sk/courses/ml/sl/PCA.pdf)

Figure 1: Primul test PCA



Urmatorul lucru pe care l-am dorit sa il facem a fost sa vedem cum mai multe noruri de puncte, de diferite dimensiuni, se modifica dupa aplicarea algoritmului. Pentru acest lucru, am creat o clasa **PointCloud** care descrie un nor de puncte. In interiorul clasei am generat norul de puncte in jurul unor coordonate (x, y), sub forma unui disc, in care punctele au distributie normala. Rezultatele au fost urmatoarele:

Figure 2: Al doilea test PCA



La fel ca in primul test, se poate observa aranjarea dupa prima componenta principala a norilor.

2.5 Aplicatia PlotPCA

In acest moment, am decis sa realizam o aplicatie care are un GUI usor de folosit si unde fiecare nor de puncte este reprezentat in alta culoare, pentru a se vedea mai clar ce se intampla, acest lucru fiind mai important pentru seturile de date in care masuratorile pentru fiecare entitate sunt intrepatrunse.

2.5.1 Setul 3-cloud-points

Primele teste pe care le-am facut, au fost pe 3 nori de puncte generati aleator, la fel ca in exemplul de mai sus, si se poate vedea mai clar separarea

norilor atat pe in spatiu 2D, cat si pe axa primei componente, cea cu cea mai mare variatie. De asemenea, atunci cand norii se suprapun putin de-a lungul axei Y, algoritmul PCA reuseste sa separe norii complet de-a lungul axei primei componente. Pe de alta parte, daca norii se suprapun mai mult, atunci se poate observa ca algoritmul PCA nu mai poate separa norii la fel de bine ca si pana acum.

Figure 3: Nori separati total

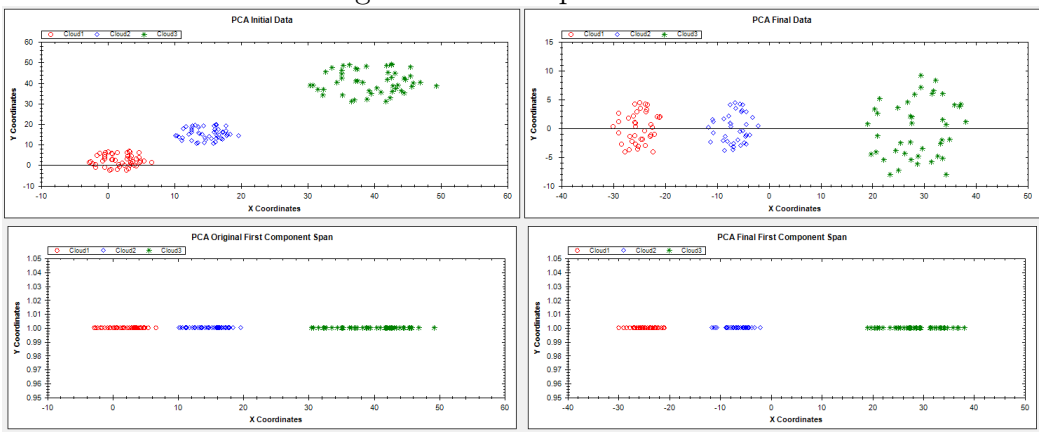


Figure 4: Nori interesectati partial

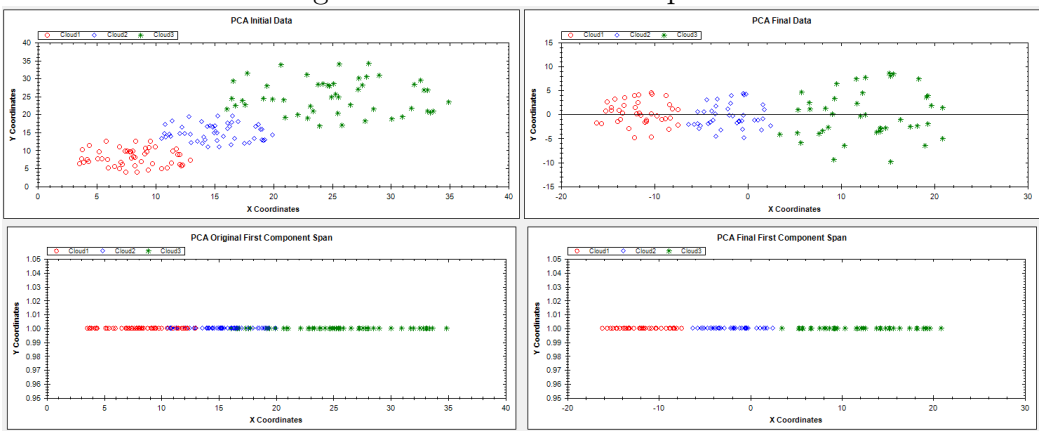
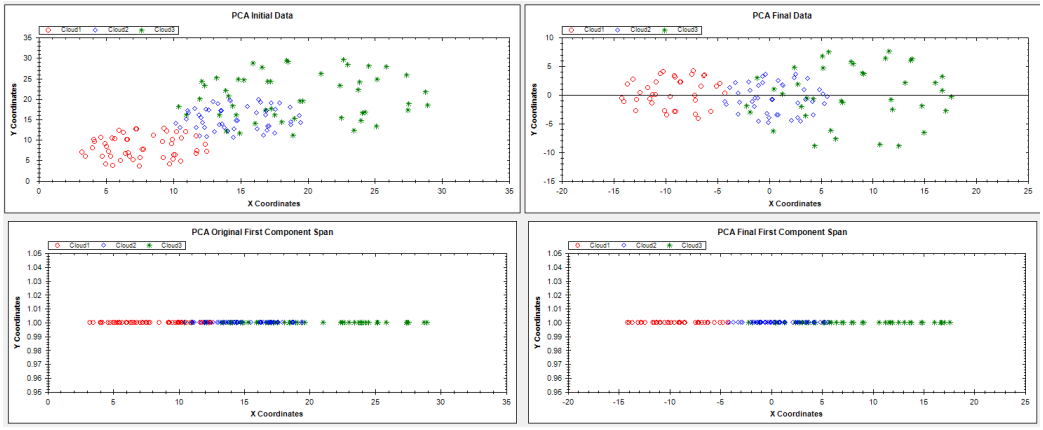


Figure 5: Norul albastru si verde intersectati aproape total



3 Kernel PCA

3.1 Setul two-moon si abordarea Kernel PCA

Pentru a aborda problema seturilor de date liniar inseparabile, nu putem folosi abordarea PCA simpla, rezultatul nu va fi cel dorit, dupa cum se poate vedea aplicand algoritmul PCA pentru setul two-moon: cele doua seturi de puncte se amesteca de-a lungul primei componente principale, astfel o clasificare a unui nou punct proiectat in spatiul respectiv va fi dificila, deoarece nu putem seta un punct de delimitare intre cei doi nori, acestia fiind intrepatrunsi.

Abordarea Kernel PCA rezolva acest lucru prin proiectarea datelor intr-un spatiu dimensional mai mare, unde ele vor deveni liniar separabile, iar acest lucru se face cu o ”functie kernel”.

Pentru a implementa Kernel PCA, vom avea in considerare urmatoarele 2 lucruri:

1. Calcularea matricii kernel

Pentru fiecare pereche de puncte se calculeaza :

$$k(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|_2^2) \tag{7}$$

Daca avem un set de date cu 100 de mostre, matricea kernel va fi o matrice simetrica 100×100.

Alegerea valorii lui γ este foarte importanta, in functie de ea rezultatul va fi cel dorit sau nu.

2. Calcularea vectorilor si a valorilor proprii

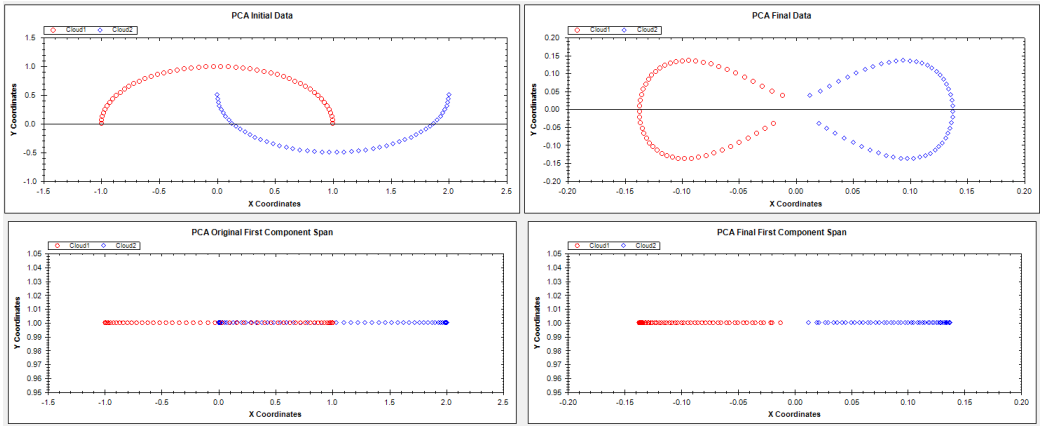
Deoarece nu putem garanta ca matricea e centrata, vom aplica urmatoarea formula pentru a o centra:

$$K' = K - 1_N K - K 1_N + 1_N K 1_N \tag{8}$$

unde 1_N este o matrice $N \times N$ cu toate valorile egale cu $\frac{1}{N}$. Acum, putem afla vectorii si valorile proprii pentru matricea centrata, iar vectorii proprii vor reprezenta setul de date proiectat pe componentele principale respective.

In imaginile urmatoare se poate observa cum algoritmul PCA si algoritmul KPCA afecteaza setul de date two-moon, rezultatul obtinut prin folosirea KPCA este mult mai bun.

Figure 6: Setul two-moon proiectat cu ajutorul algoritmul KPCA, $\gamma = 15$



3.2 Cros-validare 10-fold

Pentru a ne asigura ca algoritmul se comporta corect si in alte situatii, am decis sa facem o cros-validare 10-fold (10-fold crossvalidation). Acest lucru implica urmatoarele:

-avem 100 de puncte, distribuite in setul de date two-moon

-din acestea, vom alege pe rand, cate 10 puncte, diferite de fiecare data, si vom retine norul din care face parte fiecare

-cu celelalte 90 de puncte vom "antrena" algoritmul pentru a putea proiecta celelalte 10 puncte pe componentele principale respective, in cazul nostru, dorim sa facem proiectarea pe prima componenta principala

-avand cele 90 de puncte proiectate in spatiul nou, putem determina din nou granitele celor 2 nori de puncte, iar mijlocul acestor granite va fi punctul nostru de separare a norilor

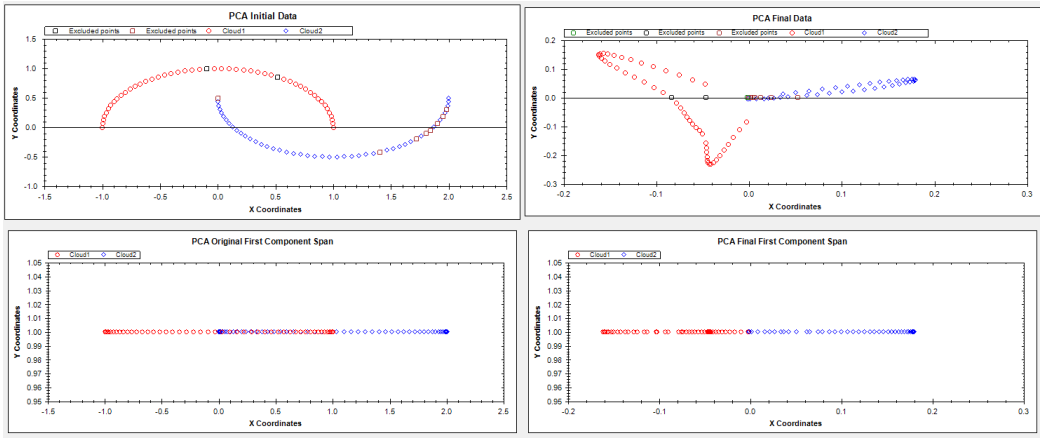
-cu punctul de separare gasit, cele 10 puncte proiectate pe prima componenta principala pot fi categorizate: daca se afla in stanga punctului de separare el va fi in norul albastru de puncte, iar daca este la dreapta punctul de separare, va apartine norului rosu

-stiind din ce nori au provenit punctele si in ce nori au fost proiectati, putem verifica eroarea algoritmului pentru cele 10 puncte: $\frac{nr.\ puncte\ proiectate\ gresit}{nr.\ puncte\ totale}$

-repetam acest proces pana cand toate punctele au fost proiectate, iar eroarea de final este cea cautata

In cazul testelor noastre, eroarea a variat intre 1% si 4%, fiind un rezultat foarte bun.

Figure 7: Exemplu de validare 10-fold



3.3 Aplicatia Eigenfaces

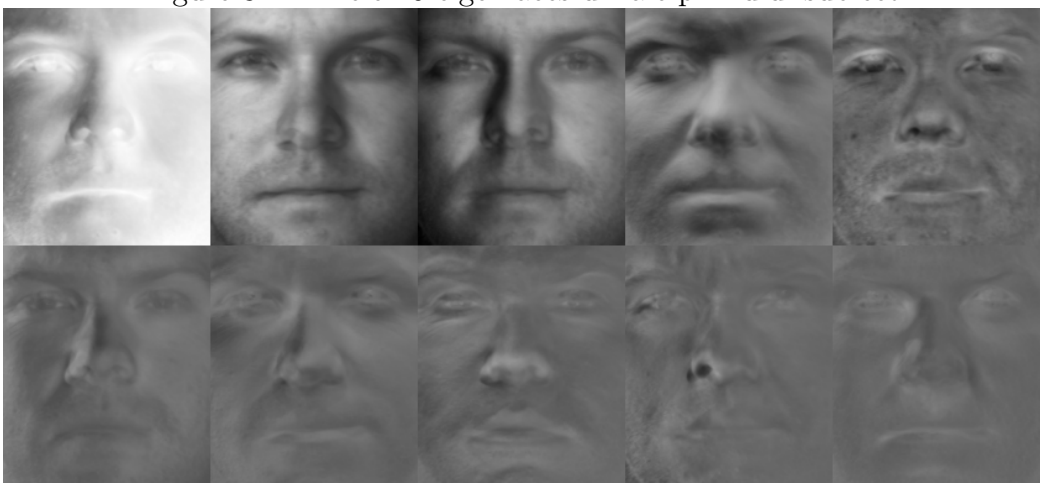
Dupa ce am realizat testele KPCA, urmatorul pas a fost sa crestem dimensiunile cu care lucram. Mai exact, am inceput sa folosim imagini pentru a vedea cum actioneaza algoritmul PCA pe ele.

Imaginile folosite au fost cele din baza de date Yale, care a fost folosita in multe alte cercetari de genul acesta, continand destui subiecti si destule imagini pentru a putea testa orice algoritm.

Pentru a putea procesa aceste imagini, ele trebuiau sa aiba toate aceeasi dimensiune, in cazul nostru 168×192, iar fiecare imagine va deveni un rand din matricea care va fi prelucrata de algoritmul PCA.

Fetele rezultate din acest algoritm reprezinta toti vectorii proprii care contribuie la spatiul fetelor intr-un mod semnificativ, si sunt sortati dupa valoarea proprie, si dupa cum se poate observa, aceste imagini devin din ce in ce mai nesemnificative:

Figure 8: Primele 10 eigenfaces-uri ale primului subiect



3.3.1 Proiectarea unei imagini noi

Pentru a putea proiecta o imagine noua in spatiul fetelor, tot ce trebuie sa facem este:

$$X = X - avg(X) \tag{9}$$

$$Y = X^T \times E_0 \tag{10}$$

X = este vectorul imaginii pe care o vom proiecta

E_0 = este vectorul propriu asociat valorii proprii maxime

Y = este imaginea proiectata.

3.3.2 Clasificarea unei imagini noi

Pentru a vedea daca o imagine nou proiectata apartine sau nu setului initial de imagini, avem nevoie mai intai de imaginea proiectata, si de o granita, la fel ca la abordarea KPCA.

Modul prin care vom afla daca o imagine apartine setului este prin calcularea distantei de la aceasta imagine la reprezentarea setului de imagini in spatiul respectiv, iar daca valorea aceasta este mai mica decat granita mentionata mai sus, imaginea va fi clasificata ca va apartine setului initial.

Pentru a afla aceasta distanta, vom efectua urmatoarele operatii:

$$D = W_i - Y \tag{11}$$

$$N = \|D_i\| \tag{12}$$

$$v = min(N) \tag{13}$$

W = reprezinta fetele setului de date initial proiectate in spatiul fetelor

D = diferenta dintre toate liniile lui W si Y

N = norma fiecarei linii din D

v = este distanta dintre imaginea nou proiectata si spatiul fetelor

Pentru a afla granita care delimiteaza imaginile din spatiul fetelor de celelalte, vom folosi aceleasi operatii ca si mai sus, dar aplicate pe toate imaginile setului initial, si vom cauta $v = avg(N)$, distanta medie dintre toate fetele proiectate si spatiul fetelor. Am ales sa facem acest lucru si sa nu cautam $v = max(N)$ deoarece se poate intampla ca o imagine din setul initial sa aiba distanta fata de spatiul fetelor disproportional mai mare fata de celelalte, si astfel granita de clasificare ar fi prea mare, incluzand orice imagine pe care incercam sa o proiectam.

Acestea fiind zise, am facut urmatorul test: din cele 65 de imagini ale subiectului 1 din baza de date Yale, primele 44 au fost folosite pentru a "antrena" algoritmul, iar ultimele 21 au fost folosite pentru clasificare. Eroarea rezultata a fost de $\sim 50\%$, imaginile care nu au fost clasificate ca facand parte in set au fost cele in care subiectul are numai jumatate de fata iluminata, numarul acestor imagini fiind $\sim 50\%$ din ele, deci putem trage concluzia ca algoritmul face ceea ce isi propune.

4 ICA

4.1 Ce este ICA?

Una dintre problemele din lumea reala este gasirea unei reprezentari potrivite pentru date multivariate, adica date care au multe dimensiuni, si care de multe ori sunt aleatoare. Din cauza problemelor de performanta, orice fel de reprezentare este deobicei interpretata ca o transformare liniara a datelor originale. Printre aceste metode de analiza a transformarilor liniare se numara si PCA, despre care am vorbit pana acum. ICA (Independent Component Analysis) sau Analiza Componentelor Individuale este o abordare noua care are rolul de a reprezentarile liniare ale unor date cu distributie non-gaussiana, astfel incat componentele sa fie cat mai independente din punct de vedere statistic. Acest tip de separare este dorita deoarece captureaza structura esentiala a datelor in multe tipuri de aplicatii, precum extragerea caracteristicilor (feature extraction) sau separarea semnalelor (signal separation).

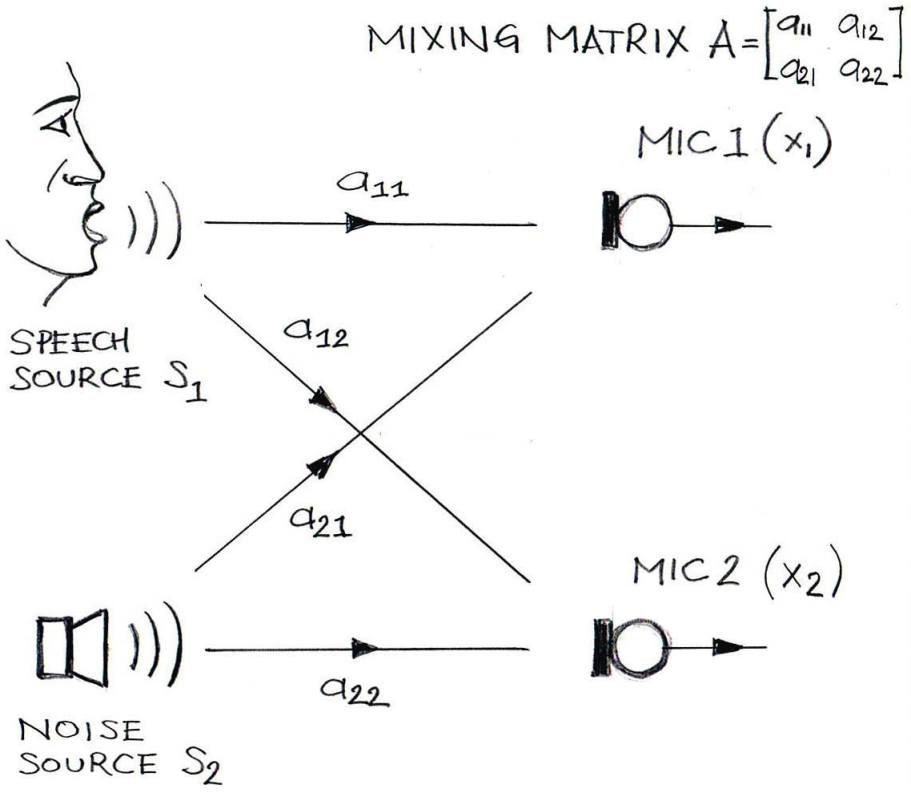
Pentru a putea vizualiza aceasta idee, putem analiza o problema din lumea reala. Sa spunem ca suntem intr-o camera in care 2 oameni vorbesc simultan. Avem 2 microfoane pe care le pozitionam in locuri diferite in camera, si ele ne dau 2 inregistrari ale semnalelor. Aceste semnale sunt $x_1(t)$ si $x_2(t)$, cu x_1 si x_2 functii de amplitudine in functie de t. Fiecare inregistrare este o suma ponderata a semnalelor emise de cei doi oameni care vorbesc, pe care le notam $s_1(t)$ si $s_2(t)$. Pentru claritate le punem sub forma de ecuatie liniara:

$$x_1(t) = a_{11}s_1 + a_{12}s_2 \tag{14}$$

$$x_2(t) = a_{21}s_2 + a_{22}s_2 \tag{15}$$

unde $a_{11}, a12, a_{21}, a_{22}$ sunt niste parametrii care determina distanta dintre microfoane si vorbitori. Avand aceste date, daca am putea estima cele doua semnale initiale $s_1(t)$ si $s_2(t)$ folosind doar semnalele inregistrate $x_1(t)$ si $x_2(t)$, am reusi sa rezolvam multe probleme din lumea reala. Tipul de problema prezentat mai sus este denumit si *cocktail-party problem*.

In imaginea urmatoare avem o reprezentare vizuala a problemei:



Stiind x_1 si x_2, daca am cunoaste elementele matricii A, atunci am putea afla s_1 si s_2 prin mijloace normale, fiind o ecuatie liniara. Dar, formuland problema ca mai inainte, informatiile initiale sunt cele 2 semmnale inregistrate de la cele 2 microfoane.

O abordare de rezolvare a problemei ar putea fi sa incercam sa ne folosim de proprietatiile statistice ale semnalelor $s_1(t)$ si $s_2(t)$ pentru a estima matricea A. Tot ce trebuie sa facem este sa presupunem ca semnalele $s_i(t)$ sunt independente statistic la fiecare moment t.

4.2 Definitia riguroasa ICA

Pornim de la presupunerea ca observam n amestecari liniare $x_1, x_2, \ldots, x_n$ a n componente independente:

$$x_j = a_{j1}s_1 + a_{j2}s_2 + \ldots + a_{jn}s_n, \text{ pentru toate } j. \tag{16}$$

Se observa ca am renuntat la indicele de timp t, deoarece ICA presupune ca atat fiecare amestecare x_j cat si fiecare sursa s_k sunt variable aleatoare, lucru care o sa fie folositor cand vom vorbi despre distributia datelor. De asemenea, ne vom asigura ca variabilele observate x_j sunt centrate, scazand media esantioanelor.

Este convenient sa ne folosim de notatia vector-matrice, in locul sumelor precum cea precedenta. De acum vom nota $\mathbf{x}$ vectorul al carui elemente sunt amestecarile x_j, si la fel pentru $\mathbf{s}$, ale carui elemente sunt s_j. De asemenea, vom nota $\mathbf{A}$ matricea de elemente a_{ij}. Folosind aceasta notatie, modelul de amestecare se scrie astfel:

$$\mathbf{x} = \mathbf{A}\mathbf{s}. \tag{17}$$

Modelul prezentat este modelul ICA, si este un model generativ, descris in statistica astfel:

$$P(X|Y = y) \tag{18}$$

unde X este o variabila observabila, iar Y este variabila cautata, ceea ce inseamna ca descrie cum datele observate, x_i, sunt generate de un proces aplicat asupra datelor initiale, in cazul nostru componentele s_i. Componentele independente sunt variabile latente, nu sunt observate direct, ci sunt deduse din alte variable. De asemenea, presupunem ca matricea de amestecare A nu este cunoscuta. Tot ce observam este vectorul aleator $\mathbf{x}$, si trebuie sa estimam atat $\mathbf{A}$ cat si $\mathbf{s}$. Acest lucru trebuie facut sub anumite ipoteze.

Punctul de inceput al ICA este ipoteza ca componentele independente s_i sunt independente statistic, lucru pe care il vom defini mai incolo. Vom presupune deasemenea ca acestea au distributie *nongaussiana*, si ca matricea A de amestecare este patrata. Dupa ce am estimat matricea A, ii putem calcula inversa, sa zicem $\mathbf{W}$, si obtinem componentele independente astfel:

$$\mathbf{s} = \mathbf{W}\mathbf{x} \tag{19}$$

4.3 Limitarile ICA

Avand in vedere modelul dat, se pot observa 2 limitari:

1. Nu putem determina *energiile* componentelor independente.

Motivul este ca, atat **s** cat si **A** sunt necunoscute, orice multiplicator scalar al unei dintre surse s_i ar putea fi negat prin impartirea coloanei corespunzatoare a_i al **A** cu acest scalar. Acelasi lucru se intampla cu semnul variabilelor, acesta se poate inmulti cu -1 fara a afecta modelul. Din fericire, aceasta limitare nu este semnificativa in majoritatea cazurilor.

2. Nu putem determina ordinea componentelor independente:

Motivul este ca, din nou, atat **s** cat si **A** fiind necunoscute, putem schimba ordinea termenilor din suma precedenta, oricare dintre componente putand fi prima. Formal, avand o matrice de permutare **P** si inversa ei, modelul poate fi formulat astfel:

$$x = AP^{-1}Ps. \tag{20}$$

Elementele lui **Ps** sunt componentele independente originale, dar in ordine diferita, iar matricea $\mathbf{AP}^{-1}$ este doar o noua matrice de amestecare necunoscuta, care poate fi aflata.

4.4 Independenta

4.4.1 Definitie

Ca sa putem intelege conceptul de independenta, consideram 2 variabile scalare aleatoare, y_1 si y_2. Variabilele y_1 si y_2 sunt independente daca informatia din y_1 nu influnteaza informatia din y_2 si invers. Acest lucru este valabil pentru variabilele precedente s_1, s_2, dar nu si pentru amestecarile x_1, x_2.

Independenta poate fi definita prin probabilitati de densitate. O functie de densitate a unei probabilitati este folosita pentru a specifica probabilitatea unei variabile aleatoare de a avea valori dintr-un anumit interval. Probabilitatea este data prin integrarea functiei cu limitele intervalului. Sa notam $p(y-1, y_2)$ functia de densitate cumulata a y_1 si y_2, si $p_1(y_1)$ functia de densitate marginala a y_1, care reprezeinta functia de densitate a variabilei y_1:

$$p_1(y_1) = \int p(y_1, y_2) dy_2, \tag{21}$$

si similar pentru y_2. Acestea fiind spuse, definim y_1 si y_2 ca fiind independente daca:

$$p(y_1, y_2) = p_1(y_1) p_2(y_2) \tag{22}$$

Definitia aceasta poate fi folosita pentru a demonstra una dintre cele mai importante proprietati ale variabilelor aleatoare independente. Folosind functii h_1 si h_2 de mai sus, avem:

$$Eh_1(h_1), h_2(y_2) = Eh_1(y_1) Eh_2(y_2) \tag{23}$$

Putem demonstra astfel:

$$\begin{aligned} E\{h_1(y_1), h_2(y_2)\} &= \iint h_1(y_1) h_2(y_2) p(y_1, y_2) dy_1 dy_2 \\ &= \iint h_1(y_1) p_1(y_1) h_2(y_2) p_2(y_2) dy_1 dy_2 \\ &= \int h_1(y_1) p_1(y_1) dy_1 \int h_2(y_2) p_2(y_2) dy_2 \\ &= E\{h_1(y_1)\} E\{h_2(y_2)\} \end{aligned} \tag{24}$$

4.4.2 Corelatia si independenta

O forma mai slaba de independenta este necorelatia. Doua variabile independente y_1 si y_2 se numeste necorelate daca covariatia lor este 0:

$$E\{y_1 y_2\} - E\{y_1\}E\{y_2\} = 0 \tag{25}$$

Daca variabilele sunt independente, ele sunt necorelate, rezultand $h_1(y_1) = y_1$ si $h_2(y_2) = y_2$ din ecuatia precedenta. Pe de alta parte, necorelarea nu implica independenta. Sa presupunem ca (y_1, y_2) sunt valori discrete si au o distributie astfel incat perechea are $1/4$ sanse sa ia una dintre valorile urmatoare: $(0, 1), (0, -1), (1, 0), (-1, 0)$. Atunci y_1 si y_2 sunt necorelate, dar:

$$E\{y_1^2, y_2^2\} = 0 \neq \frac{1}{4} = E\{y_1^2\}E\{y_2^2\} \tag{26}$$

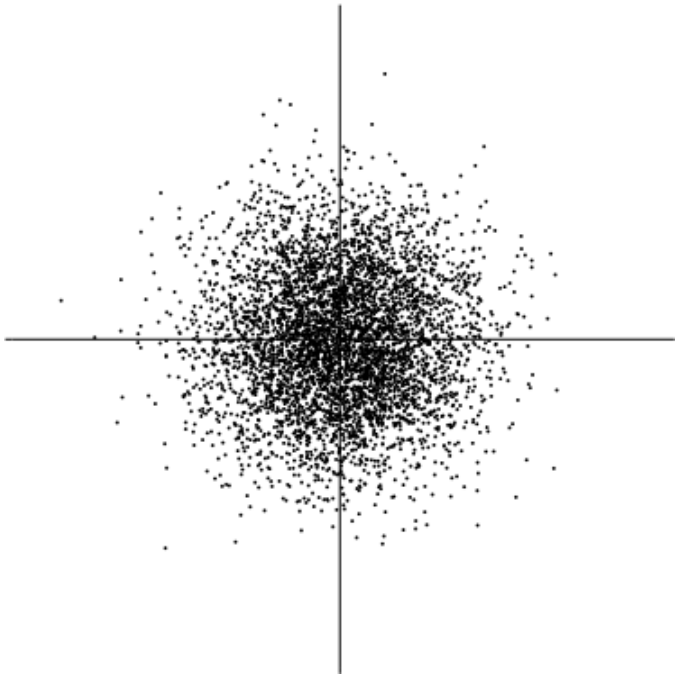
si astfel conditia de mai sus nu este respectata, variabilele nefiind independente.

4.4.3 Variabile nongaussiene

Restrictia fundamentala a ICA este ca componentele independente sa fie nongaussiane pentru ca separarea sa se intample. Ca sa vedem de ce acest lucru este necesar, sa presupunem ca sursele s_1, s_2 au distributie gaussiana, iar A este o matrice ortonormala. Din asta rezulta ca x_1, x_2 vor fi gaussiene, necorelate si au variatie unitara. Densitatea cumulata a x_1, x_2 este:

$$p(x_1, x_2) = \frac{1}{2\pi} \exp(-\frac{x_1^2 + x_2^2}{2}) \tag{27}$$

Distributia aceasta este ilustrata mai jos:



Se poate observa cum densitatea este complet simetrica. Asta inseamna ca nu contine informatii cu privire la directiile coloanelor matricii de amestecare A. Din acest motiv, A nu poate fi estimat.

4.5 Principiile estimarii ICA

4.5.1 Nongaussianitate inseamna independenta

Daca luam in calcul cele spuse pana acum, putem deduce ca esenta estimarii unui model ICA sta in nongaussianitate.

Teorema limita centrala, in versiunea clasica, stabileste ca sumele partiale "normalizate" ale unui sir de variabile aleatoare independente si identic distribuite, cu dispersie finita, tind in distributie catre legea normala (gaussiana).

Sa presupunem ca datele din vectorul x este distribuit asemenea mod-

elului ICA prezentat mai devreme, adica ca o amestecare de componente independente, si distributia componentelor independente este identica. Pentru a estima componentele independente, trebuie sa aplicam operatii liniare asupra x_i:

$$y = w^T x = \sum_i w_i x_i, \text{unde } w \text{ este vectorul pe care il cautam.} \tag{28}$$

Daca w ar fi un rand al inversei lui A, aceasta combinatie liniara ar fi egala cu una dintre componentele independente. Ca sa vedem cum teorema limita centrala este folosita pentru a gasi un w ca sa respecte conditia precedenta, putem face o schimbare de variabile, definind $z = A^T w$. Apoi avem:

$$y = w^T x = w^T A s = z^T s \tag{29}$$

Se observa ca y este o combinatie liniara de s_i, cu ponderile date de z_i. Deoarece suma suma celor doua variabile independente este mai gaussiana decat variabilele originale, $z^T s$ este mai gaussian decat oricare s_i, si devine mai putin gaussian atunci cand este s_i, in acest caz, doar un element al lui z este diferite de 0.

Cautam w astfel incat $w^T x$ sa fie cat mai nongaussian. Un astfel de vector ar corespunde unui z care are doar o componenta diferita de zero. Asta inseamna ca $w^T x = z^T s$ este egal cu una dintre componentele independente.

Maximizand nongaussianitatea lui $w^T x$ ne da una dintre componentele independente. Spatiul de cautare pentru optimizarea nongaussianitatii intr-un spatiul n-dimensional al vectorilor w are $2n$ maxime locale, 2 pentru fiecare componente independenta, s_i si $-s_i$. Ca sa gasim mai multe componente independente, trebuie sa gasim toate maximele locale.

4.6 Masuratori ale nongaussianitatii

Pentru a folosi nongaussianitatea in estimarea ICA, avem nevoie de o masura cantitativa a nongaussinitatii unei variabile aleatoare, y. Vom presupune ca y este centrat, adica are media egala cu zero, iar variatia ei este unu.

4.6.1 Kurtosis / Aplatizarea

O masura clasica a nongaussinitatii este aplatizarea, sau al patrulea moment central al unei variabile aleatoare, si este definit astfel:

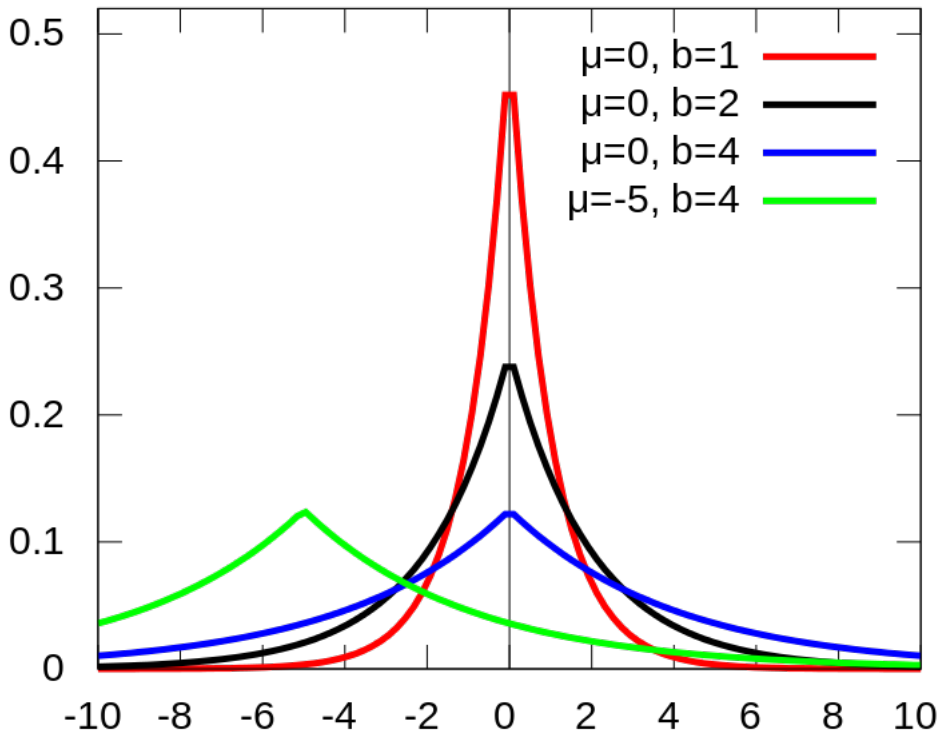
$$kurt(y) = E\{y^4\} - 3(E\{y^2\})^2 \tag{30}$$

Deoarece am stabilitia ca variatia este unu, atunci formula se simplifica la:

$$kurt(y) = E\{y^4\} - 3 \tag{31}$$

Acest lucru arata defapt ca aplatizarea reprezinta al patrulea moment, $E\{y^4\}$. Pentru un y gaussian, al patrulea moment este definit ca $3(E\{y^2\})^2$, deci aplatizarea pentru un asemenea y este 0. Pentru majoritatea variabilelor aleatoare nongaussiene, aplatizarea este diferita de zero.

Aplatizarea poate fi pozitiva sau negativa, variabilele aleatoare care au aplatizare negativa numinduse subgaussiene, iar cele care au aplatizare pozitiva se numest supergaussiene. Variabilele aleatoare supergaussiene corespund cu distributii de tip Laplace, care au un varf ascutit, si coada alungita:



Variabilele aleatoare subgaussiene corespund cu distributiile uniforme, acestea fiind plate in reprezentare:

Valoarea absoluta a aplatizarii, a fost folosita ca masura a nongaussianitatii in ICA, deoarece este simplu atat teoretic cat si computational. Pentru a calcula aplatizarea, putem folosi pur si simplu al patrulea moment al datelor avute. Analiza teoretica este simplificata datorita proprietatii de liniaritate. Daca avem doua variabile aleatoare independente, atunci:

$$kurt(x_1 + x_2) = kurt(x_1) + kurt(x_2) \tag{32}$$

$$kurt(\alpha x_1) = \alpha^4 kurt(x_1) \tag{33}$$

Pentru a demonstra cum arata spatiul de optimizare pentru aplatizare, si cum se pot afla componentele independente pot fi gasite prin maximizarea sau minimizarea aplatizarii. Avand modelul $x = As$ ca mai inainte, si componentele independente s_1 si s_2, cu valorile aplatizarii $kurt(s_1)$ si $kurt(s_2)$, ambele diferite de zero, cautam componentele independente prin $y = w^T x$. Facem din nou transformarea $z = A^T w$. Apoi avem:

$$y = w^T x = w^T As = z^T s = z_1 s_1 + z_2 s_2 \tag{34}$$

Folosind proprietatiile aplatizarii mentionate mai devreme avem:

$$kurt(y) = kurt(z_1 s_1) + kurt(z_2 s_2) = z_1^4 kurt(s_1) + z_2^4 kurt(s_2) \tag{35}$$

Mai devreme am spus ca variatia lui y este unu, deoarece s_1 si s_2 au variatia unu. Acest lucru implica faptul ca:

$$E\{y^2\} = z_1^2 + z_2^2 = 1 \tag{36}$$

ceea ce inseamna ca z este restrans la un cerc cu raza unu in spatiul 2D. Problema de optimizare este urmatoarea: care este maximul functiei $|kurt(y)| = |z_1^4 kurt(s_1) + z_2^4 kurt(s_2)|$ pentru acest tip de cerc?

Maximele in acest caz sunt in punctele unde exact unul dintre elementele vectorului z este zero si celalalt este diferit de zero, si deoarece ne aflam intr-un cerc cu raza unu, elementul diferit de zero este egal cu -1 sau 1. Putem observa ca aceste puncte sunt cele in care y este egal cu unul dintre componentele independente $\pm s_i$, si problema este rezolvata.

In practica am incepe de la un w aleator, am calcula directia in care aplatizarea lui $y = w^T x$ creste cel mai rapid, daca aplatizarea este pozitiva sau scade cel mai rapid, daca aplatizarea este negativa, in functie de esantioanele lui $x(i)$ al vectorului de mixturi x, si folosind o metoda de tip gradient pentru a gasi un nou vector w.

In practica, aceasta metoda are cateva dezavantaje. Aplatizarea este sensibila la valori extreme setului de date, ceea ce ar insemna ca toata aproximarea ar depinde de aceste valori, lucru care cauzeaza rezultate eronate.

4.6.2 Negentropia

Negentropia este a doua metoda foarte importanta pentru masurarea nongaussianitatii unor variabile aleatoare independente. Din punctul de vedere al teoriei informatiei, negentropia se bazeaza pe cantitatea de entropie diferentiala. Conceptul de entropie este un concept de baza in teoria informatiei. Aceasta reprezinta gradul de informatiei pe care observarea unei variabile o da. Cu cat variabila este mai putin structurata si mai putin previzibila, cu atat mai mare este entropia variabilei. Entropia H a unei variabile aleatoare discrete Y este:

$$H(Y) = -\sum_i P(Y = a_i) log P(Y = a_i) \tag{37}$$

unde a_i reprezinta valorile posibile alea lui Y. Aceasta definitie poate fi generalizata pentru variabile aleatoare continue, in acest caz, numinduse entropie diferentiala:

$$H(Y) = -\int f(y) log f(y) dy \tag{38}$$

unde $f(y)$ este functia de densitate a unui vector aleator y.

Un rezultat fundamental al teoriei informatiei este stabilirea faptului ca variabilele gaussiene au cea mai mare entropie, comparativ cu alte variabile cu variatie egala. Asta inseamna ca entropia poate fi folosita ca si masura pentru nongaussianitate, aratand ca variabilele cu distributie gaussiana sunt cele mai putin structurate dintre toate distributiile. Entropia este mica pentru variabilele ale caror distributii sunt concentrate pe anumite valori, fiind

analog unor functii cu varfuri ascutite, de tip Laplace. Pentru a obtine o masura a nongaussianitatii care este zero pentru o variabila aleatoare gaussiana si tot timpul pozitiva, se foloseste o versiune modificata a definitiei entropiei diferentiale, numita negentropie. Negentropia J este definitia astfel:

$$J(y) = H(y_{gauss}) - H(y) \tag{39}$$

unde y_{gauss} este variabila aleatoare gaussiana care are aceasi matrice de covariatie ca si y. Datorita proprietatilor de mai sus, negentropia este pozitiva, si zero atunci cand variabila y are distributie gaussiana.

La fel ca la aplatizare, aproximarea negentropiei are un dezavantaj major, acela ca este greu de calculat din punct de vedere computational, deoarece ar fi nevoie de o estimare a functiei de densitate, lucru care nu poate fi facut precis.

4.6.3 Aproximarea negentropiei

Dupa cum am mentionat mai devreme, aproximarea negentropiei este dificila. In practica, se fac anumite aproximari, iar tipurile de aproximari abordate se va face aici.

$$J(y) \approx \frac{1}{12} E\{y^3\}^2 + \frac{1}{48} kurt(y)^2 \tag{40}$$

Prima abordare este cea bazata pe momentele statistice: Variabila y se presupune ca are media egala cu zero, si variatie unitara. Se poate observa ca aceata abordare sufera de aceleasi limitari precum aproximarea folosind aplatizarea. Pentru a evita acest lucru, s-au dezvoltat aproximari bazate pe principiul entropiei maxime, cu forma:

$$J(y) \approx \sum_{i=1}^{p} k_i [E\{G_i(y)\}] - E\{G_i(v)\}]^2 \tag{41}$$

unde k_i reprezinta constante pozitive, si v este o variabila gaussiana, iar atat v cat si y au media zero si variatie unitara, functiile G_i fiind niste functii nonpatratice. In cazul in care folosim o singura functie nonpatratica, aproximarea devine:

$$J(y) \propto [E\{G(y)\}] - E\{G(v)\}]^2 \tag{42}$$

Alegand G bine, putem obtine aproximari mult mai bune decat cele de mai devreme, spre exemplu, daca alegem G care nu creste repede, precum:

$$G_1(u) = \frac{1}{a_1} \log \cosh a_1 u_1, G_2(u) = -\exp(\frac{-u^2}{2}) \tag{43}$$

unde $1 \leq a_1 \leq 2$ este o constanta. Cu toate ca nu am folosit aceste tipuri de aproximari, ele sunt un compromis bun intre metodele de masurare a nongaussianitatii abordate mai devreme, aplatizarea si negentropia.

4.7 Abordarea Maximum Likelihood

O abordare foarte populara pentru rezolvarea modelului ICA este *maximum likelihood estimation*. Aceasta metoda estimeaza valorile parametrului unui model. Acesti parametri sunt gasiti astfel inca estimarea lor sa mareasca sansa ca modelul sa produca datele observate. Pentru cazul particular al modelului nostru ICA, avand $W = (w_1, \ldots, w_n)^T$ ca A^{-1}, *log-likelihood-ul* ia forma:

$$L = \sum_{t=1}^{T} \sum_{i=1}^{n} \log f_i(w_i^T x(t)) + T \log |\det W| \tag{44}$$

unde f_i este functia de densitate a lui s_i, aici presupunand ca o stim, iar $x(t)$ reprezinta esantioanele amestecarilor x_i. Termenul $\log |\det W|$ vine din faptul ca pentru un vector aleator x cu densitatea p_x, si pentru orice matrice W:

$$y = Wx, \text{data de } p_x(Wx)|\det W|. \tag{45}$$

4.8 Metoda Gradient Descent

Metoda gradient descent este una dintre cele mai folosite metode pentru aproximarea maximului sau minimului pentru o functie complexa. Acesta functioneaza facand pasi in directia gradientului functiei in punctul curent, astfel se ajunge la minimul functiei. Daca in schimb, pasii sunt in directia pozitiva gradientului, se ajunge la maximul functiei. Gradientul unei functii reprezinta derivata partiala a functiei pe care incercam sa o minimizam/maximizam.

Vom folosi aceasta metoda pentru a incerca sa aproximam matricea A^{-1}, sau W, aceasta fiind matricea care ne va ajuta sa separam componentele, asa cum am mentionat pana acum. Functia de update pentru o abordare de tip gradient descent pentru optimizarea maximum likelihood este:

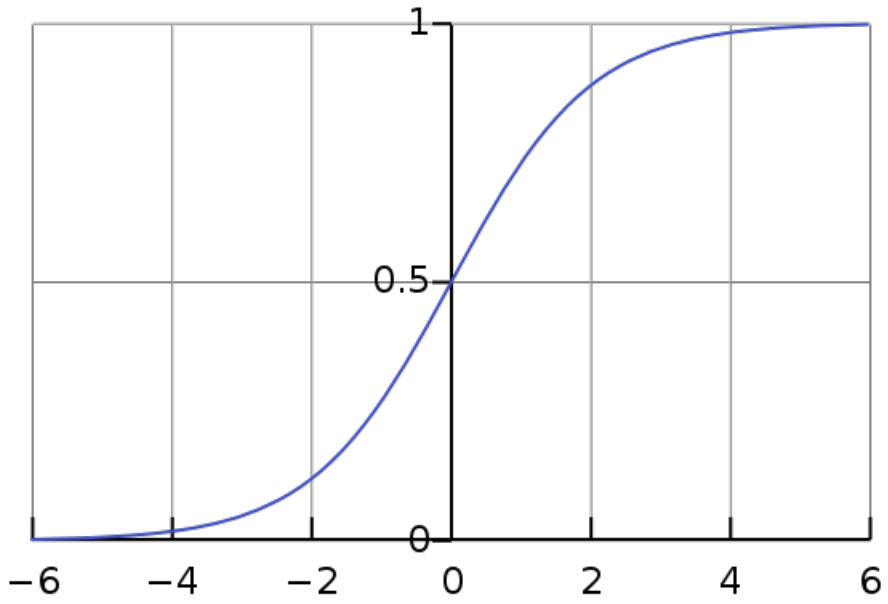
$$W^+ = W + \mu[I + g(y)y^T]W \tag{46}$$

unde μ este rata de invatare a algoritmului, iar g este functia de densitate a componentelor independente.

Un factor important in succesul acestui algoritm este alegerea functiei de densitate. Pana acum am presupus ca se cunoaste functia de densitate a componentelor independente, dar in practica, acest lucru nu poate fi garantat, iar din aceasta cauza, va fi nevoie de o aproximare. S-a observat ca o functie potrivita acestei probleme este functia sigmoida, deoarece avand o distributie continua, calculand derivata acestei functii ajungem la o functie de densitate, care poate fi folosita pentru aproximarea densitatii componentelor independente. Aici vom defini g ca fiind functia sigmoida, in practica s-a demonstrat ca este suficient de buna:

$$g_i(x) = \frac{1}{1 + e^{-x}} \tag{47}$$

Functia sigmoida are urmatoarea distributie:



4.9 Metodologia si algoritmul

4.9.1 Jupyter Notebook

Implementarea algoritmului s-a facut in limbajul de programare Python, folosind platforma Jupyter Notebook. Am ales aceasta platforma deoarece face ca pasii algoritmului sa poata fi mai bine urmariti, iar afisarea rezultatelor sa se faca intr-o maniera usoara si concisa. Aplicatia functioneaza pe un server local, iar codul este scris intr-un *notebook (caiet)*, prezent intr-o pagina web, codul fiind impartit in *cells (celule)*, in fiecare celula putem scrie o bucata de cod, care mai apoi poate fi executata. Acest stil de lucru este de folos deoarece permite executarea fiecarei celule individual, scurtand timpul dedicat intretinerii caietului. De asemenea, in aceste celule putem amplasa grafice, histograme, elemente de tip *mark-up* pentru prezentarea unor informatii suplimentare, si elemente de tip audio sau video.

4.9.2 Semnalele audio testate

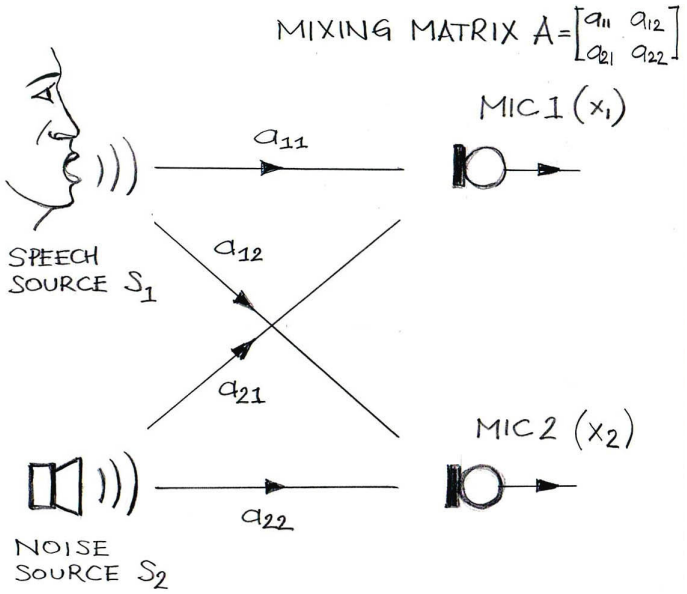
Pentru testarea algoritmului, am ales mai multe seturi de semnale audio, fiecare cu proprietati diferite. Astfel, am realizat patru teste:

- Semnale primitive: pentru inceput, am testat algoritmul pe doua tipuri de semnale primitive, generate in notebook, unul de tip sinus, si unul de tip fierastrau.
- Voce si muzica: pentru a testa un caz de separare unde se poate face clar distinctia intre cele doua semnale amestecate, am ales o voce care spunea cifrele in engleza, si o melodie.
- Doua voci: acesta a fost testul care simula cel mai bine realitatea, doua voci, una care spunea cifrele in engleza, si una care spunea cifrele in spaniola.
- Trei semnale: acesta a fost testul care demonstra faptul ca se pot separa mai mult de doua componente independente amestecate, asa cum era ipoteza in teorie. Semnalele alese au fost cele reprezentand un aspirator in functiune, un voce umana, si un grup de oameni ce aplauda.

4.9.3 Pre-procesarea datelor

Pentru toate semnalele audio inafara de cele primitive, a fost nevoie sa centram datele, la fel ca in cazul PCA, pentru standardizare.

Pornind de la semnalele audio initiale, vom simula amestecarea lor, folosind operatorul de amestecare, A. In realitate, acest A ar fi dat de distantele dintre sursele de semnal si dispozitivele care le inregistreaza. In cazul nostru, A va fi o matrice patratica, fie de 2×2, fie de 3×3, in functie de caz. Acest proces arata ca mai inainte:



Pentru fiecare caz testat, am ales initial matricea A aleator, cu valorile distribuite uniform intre 0.01 si 0.1. Deoarece acest lucru a facut verificarea rezultatelor si modificarea algoritmului destul de greu, am ales dintre matricile aleatoare A cele care au demonstrat cel mai bine eficacitatea algoritmului. O astfel de matrice este:

$$\begin{matrix} 0.56804456 & 0.92559664 \\ 0.07103606 & 0.08712930 \end{matrix}$$

Vom prezenta mai tarziu rezultatele fiecarui test, iar acolo vor fi prezentate toate variabilele folosite pentru a se ajunge la rezultatele noastre.

4.9.4 Aplicarea algoritmului Gradient Descent

Avand sursele de semnal amestecate, trebuie sa aplicam algoritmul gradient descent prezentat mai devreme pentru a incerca sa gasim componentele independente initiale. Pentru acest lucru, vom porni de la o matrice A^{-1}, sau W, cu valori aleatoare uniform distribuite intre 0.01 si 0.1. Acest interval s-a observat ca fiind unul bun pentru cazurile noastre, dar el trebuie adaptat in functie de situatie.

Se calculeaza:

$$Y = WX \tag{48}$$

unde Y reprezinta banuiala algoritmului cu privire la componentele independente. Functia de actualizare a lui W, definita mai sus, va gasi un W mai bun, luand in calcul Y. In faza initiala, acest pas se executa pentru un numar finit de pasi, ales in functie de situatie. Dupa terminarea executarii acestui numar de iteratii, W actualizat ar trebui sa fie cel care poate sa separe componentele. Acest lucru nu este intotdeauna adevarat, ceea ce inseamna ca ne trebuie o alta metoda de a opri cautarea solutiei optime.

4.9.5 Conditia de oprire

Algoritmii care aplica metoda gradient descent, deobicei sunt opriti atunci cand functia de cost ajunge sau este foarte aproape de minim. In general, se considera ca functia a ajuns la costul minim atunci cand diferentele dintre valorile consecutive obtinute de algoritm sunt mai mici decat un anumit *threshold (prag)*. Alegere acestui prag este foarte importanta, deoarece un prag prea mic va duce la castiguri foarte mici de precizie dupa multe operatii, iar un prag mare poate va opri algoritmul inainte de a ajunge la valoarea minima. Cu toate ca aceasta idee pare destul de simpla, balansul dintre precizia rezultatului obtinut si viteza algoritmului este ceva ce trebuie masurat si ajustat de la caz la caz. Functia de cost calculata este:

$$f_{cost} = \log g(Y)(1 - g(Y)) \tag{49}$$

care va fi o matrice patratica, cu numarul de linii egal cu numarul de esantioane al lui Y. Pentru ca trebuie sa calculam diferenta dintre costurile iteratiilor consecutive, si pentru ca nu putem face asta usor pentru matrici, vom modifica functia de cost astfel incat ea sa devina suma elementelor matricii rezultate din functia f_{cost}.

Aceasta abordare este cea de preferat in cazul algoritmilor de tip gradient descent, deoarece avem un mod obiectiv de a masura performanta algoritmului, si pentru ca functia de cost este legata direct de restul operatiilor. In realitate, am descoperit ca metoda aceasta de a opri algoritmul nu este posibila atunci cand lucram cu semnale audio lungi, care pot avea pana la 16000 de esantioane pe secunda, matricea pentru care trebuie sa calculam suma

elementelor ar avea dimensiuni foarte mari, spre exemplu 120000×120000 elemente, care nu incape in memoria RAM a calculatorului.

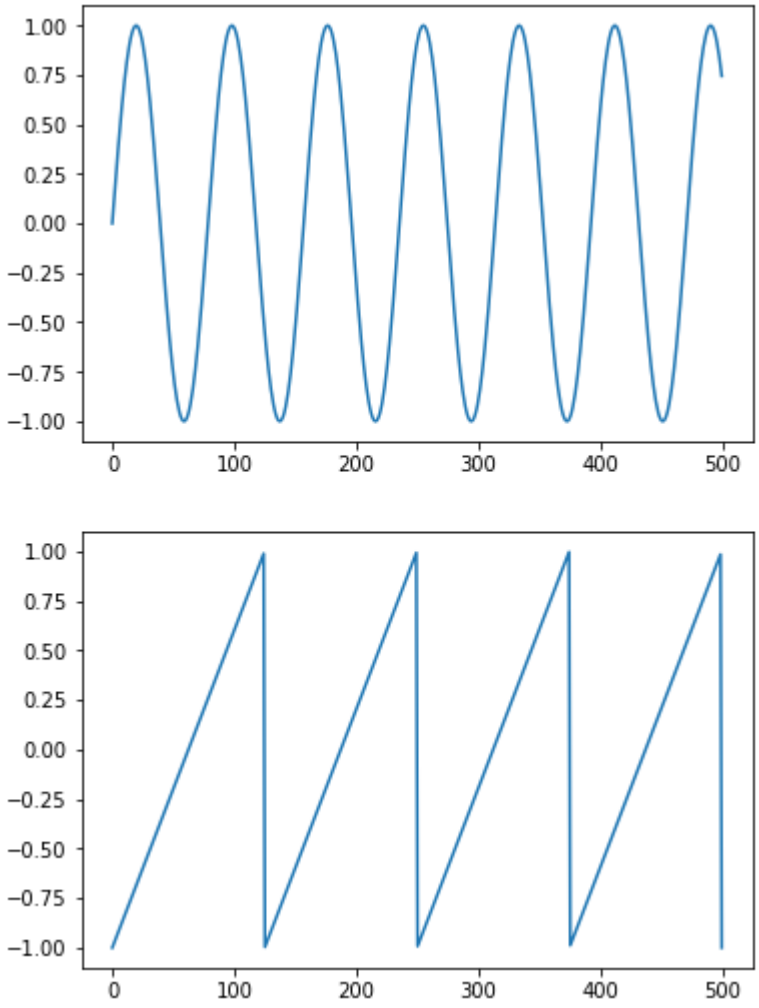
O alta abordare pentru a masura independenta este necorelatia, asa cum am mentionat intr-un capitol mai devreme. Acolo am spus ca daca doua variabile sunt independente, ele sunt necorelate, dar afirmatia inversa nu este intotdeauna adevarata. Deoarece abordarea care implica functia de cost nu era potrivita pentru semnale audio de dimensiuni mari, pentru testele care implicau acest tip de componente, am decis sa folosim necorelatia pentru a masura independenta componentelor. Dupa fiecare iteratie a algoritmului, am calculat corelatia intre componentele rezultate, si daca se afla sub un anumit prag, deobicei foarte mic, spre exemplu $3e^{-6}$. Aceasta abordare s-a dovedit a fi una reusita, era consistenta de-a lungul testelor, si poate fi aplicata pentru orice fel de semnale.

4.10 Rezultate

Dupa cum am mentionat mai sus, am realizat patru teste. Le vom prezenta aici pe fiecare, impreuna cu rezultatele obtinute.

4.10.1 Semnale primitive

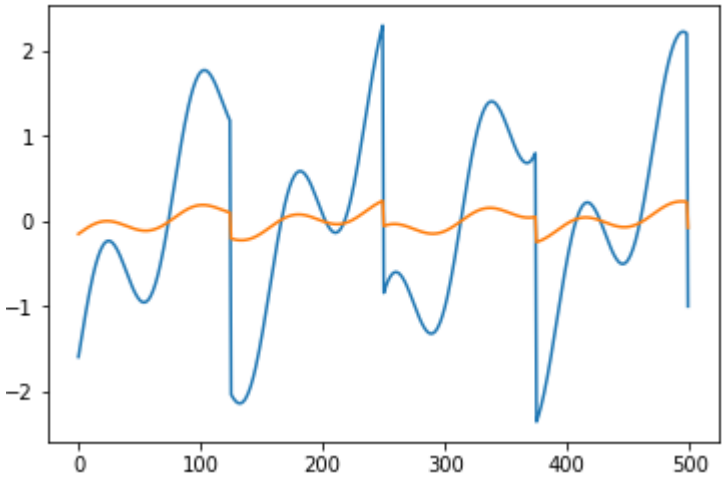
Primul test a fost realizat cu doua semnale generate prin intermediul codului Python, unul de tip sinus, si unul de tip fierastrau:



Matricea de amestecare A aleasa a fost:

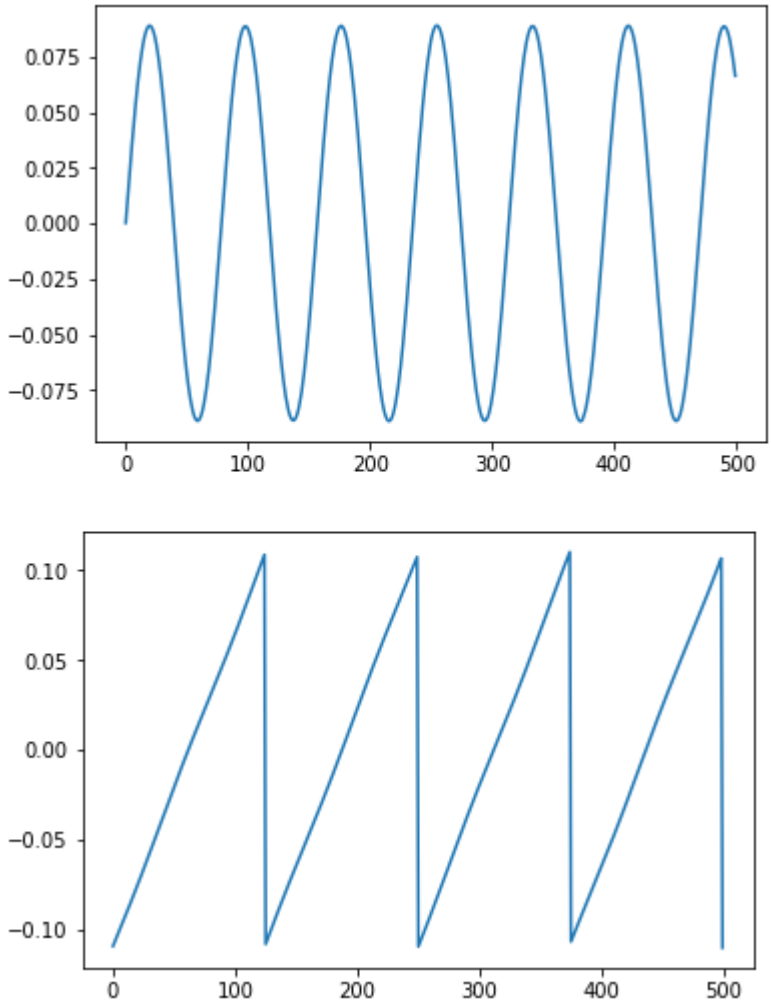
$$\begin{matrix} 0.56804456 & 0.92559664 \\ 0.07103606 & 0.08712930 \end{matrix}$$

Realizarea acestei amestecari arata astfel:



Am ales aceasta prezentare a semnalelor deoarece se poate observa cum nu se poate distinge diferenta dintre semnalele initiale, desi se pot observa asemanari cu acestea, precum varfurile ascutite ale semnalului de tip fierastrau si curbele semnalului sinus.

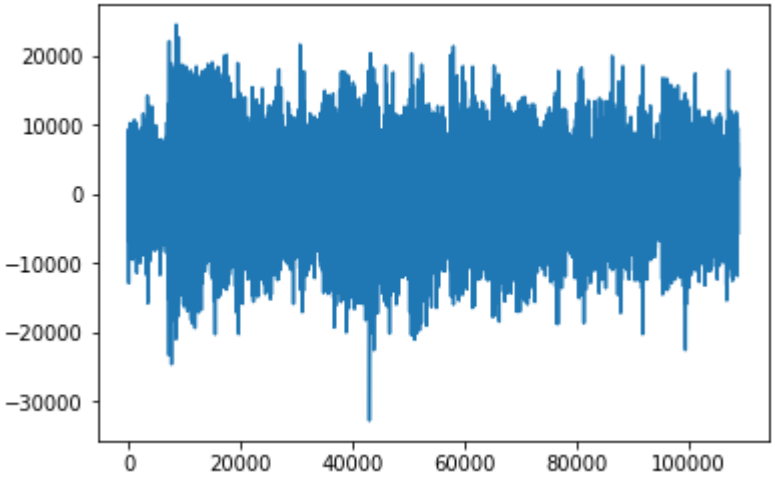
Dupa executarea algoritmului pe semnalele amestecate, se ajunge la urmatoarea separare:

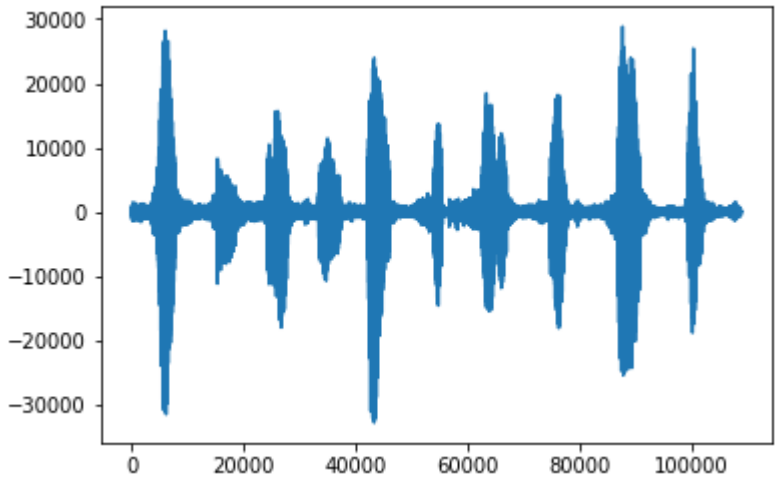


Se poate observa cum semnalele au fost complet separate, lucru care ne indica ca teoria prezentata pana acum este corecta. Coeficientul de corelatie al semnalelor separate este 0.000515439559418. Pentru acest caz simplu, ICA isi face treaba foarte bine, iar timpul de executie este acceptabil, pana in 10 secunde.

4.10.2 Semnale audio: voce si muzica

Urmatorul test a fost efectuat cu doua semnale audio preluate de pe disc. Un semnal continea muzica, si un semnal continea o voce de barbat care spunea cifrele in engleza:

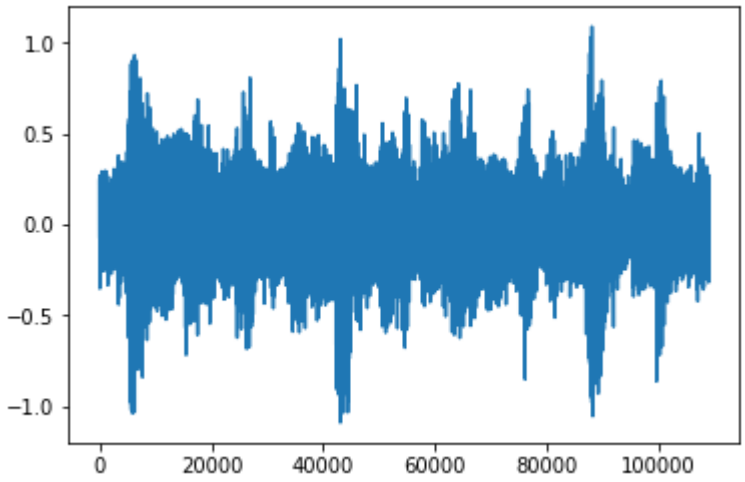


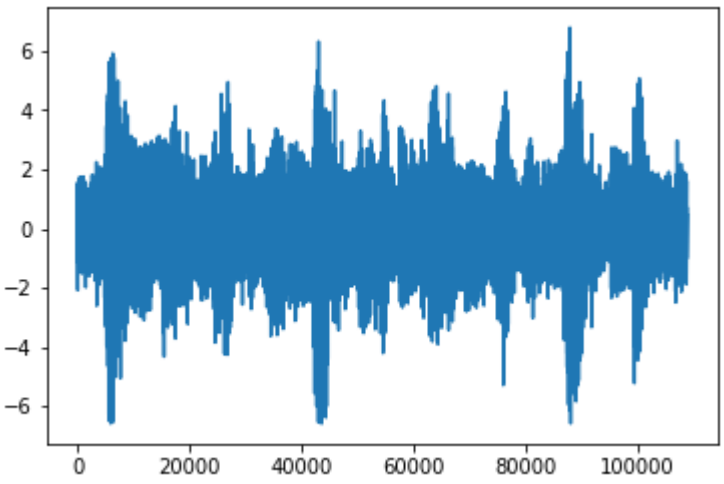


Matricea de amestecare A a fost:

$$\begin{matrix} 0.15270211 & 0.08406566 \\ 0.90514896 & 0.53725471 \end{matrix}$$

Semnalele amestecate rezultate sunt:

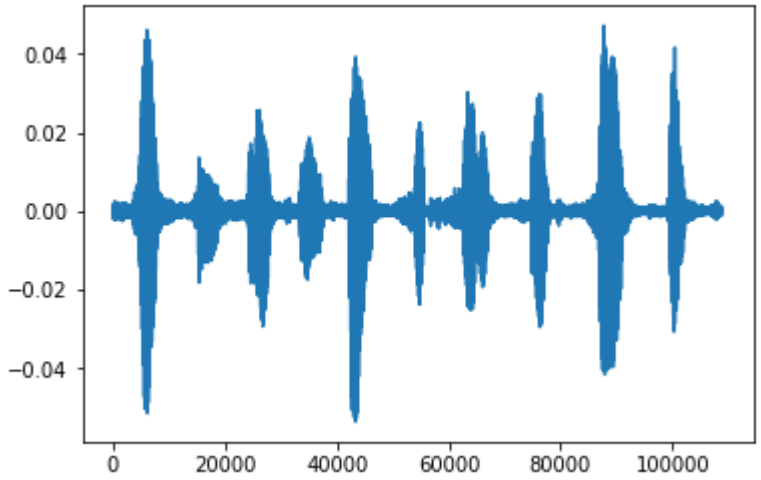


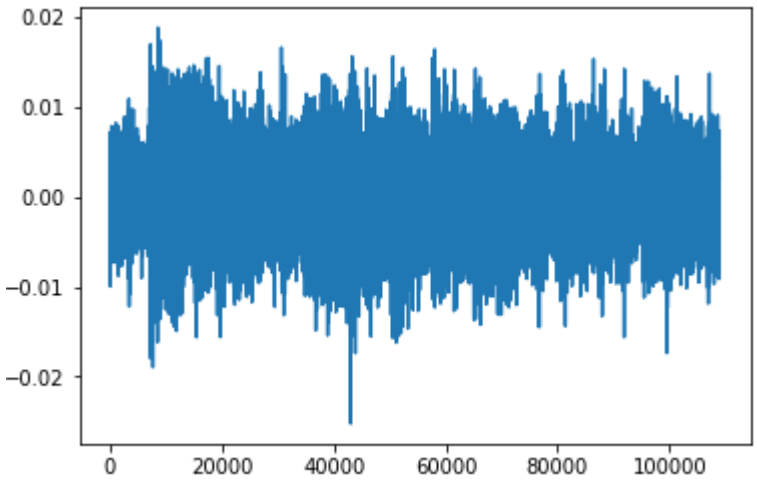


Am ales prezentarea lor separata deoarece avand magnitudini diferite, al doilea semnal l-ar fi acoperit pe primul in cazul unei afisari combinate. Acest

caz de amestecare ar simula un caz din viata reala in care unul dintre microfoane este mai aproape de cele doua surse de semnal, iar celalalt mai departe. Se observa cum cele doua semnale sunt aproape identice, daca ignoram factorul magnitudinii.

Rezultatele separarii:



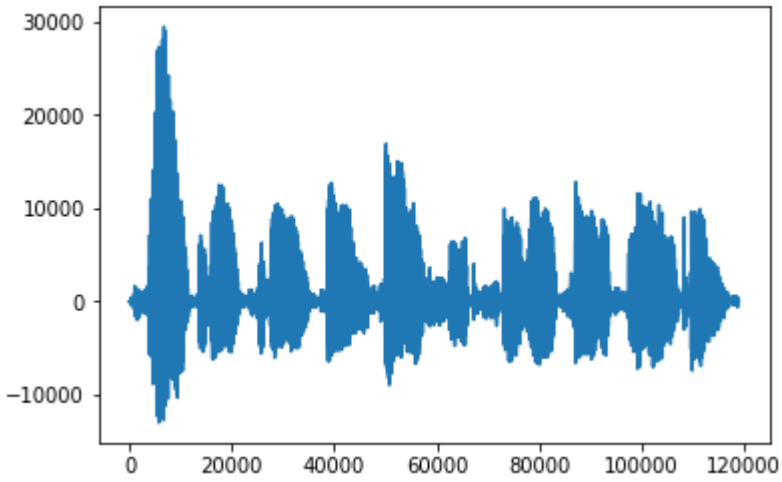


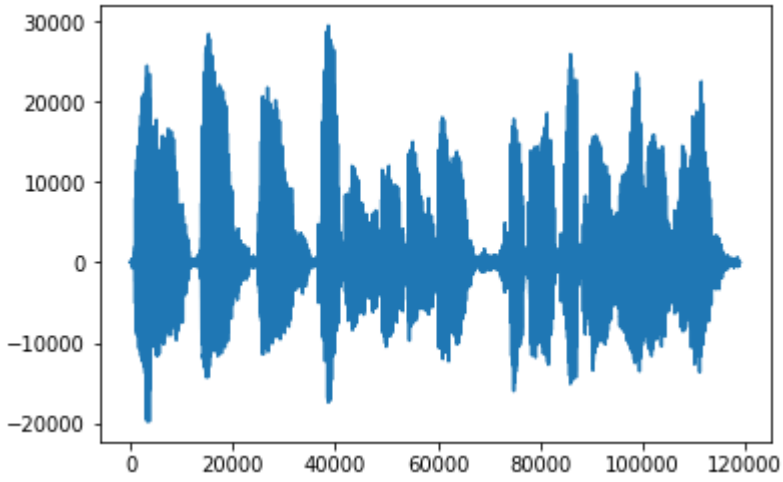
Din nou, se observa o separare foarte buna semnalelor amestecate, coe-

ficientul de corelatie fiind $-3.11671198288e-06$, acest exemplu fiind foarte expresiv in acest sens, se poate face foarte clar diferenta dintre semnalul cu voce si cel cu muzica.

4.10.3 Semnale audio: doua voci

Urmatorul este a fost efectuat cu doua semnale audio, ambele contineau voci de barbat, unul spunea cifrele in engleza, iar celalalt spunea cifrele in spaniola:

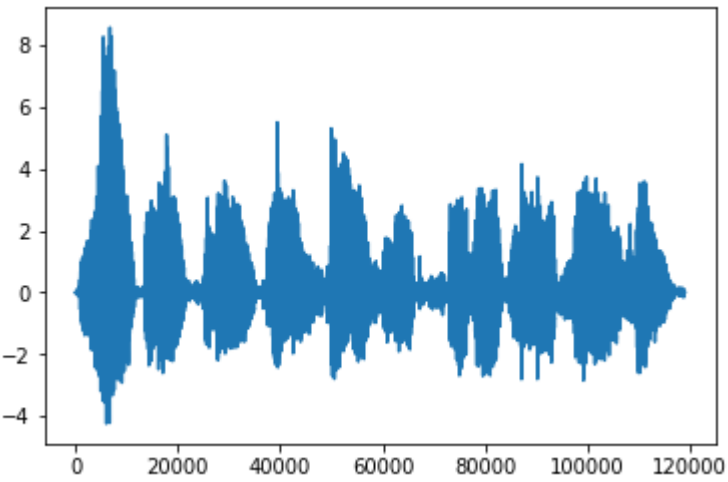


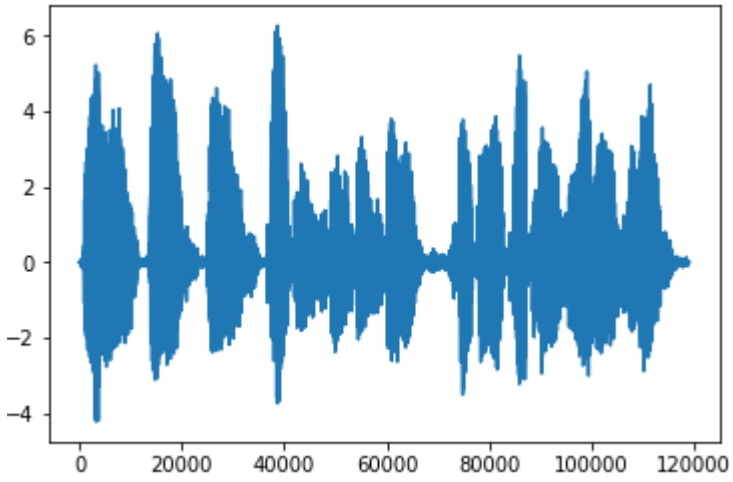


Matricea de amestecare A este:

$$\begin{matrix} 0.82583118 & 0.39868634 \\ 0.12657259 & 0.96097025 \end{matrix}$$

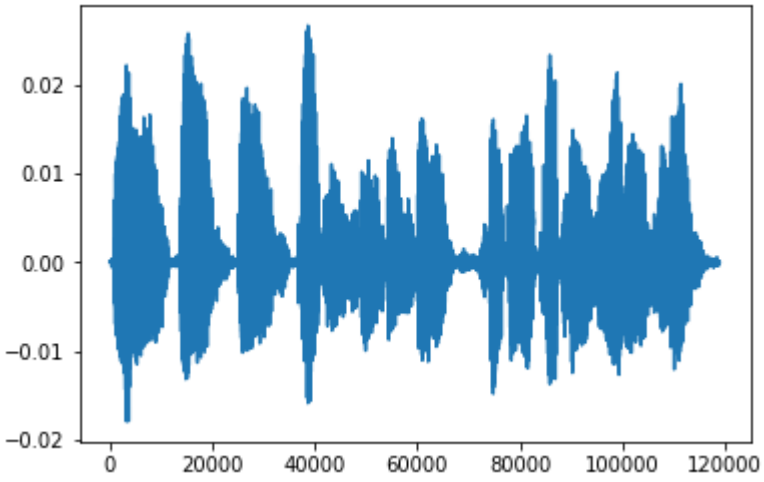
Rezultatul amestecarii este:

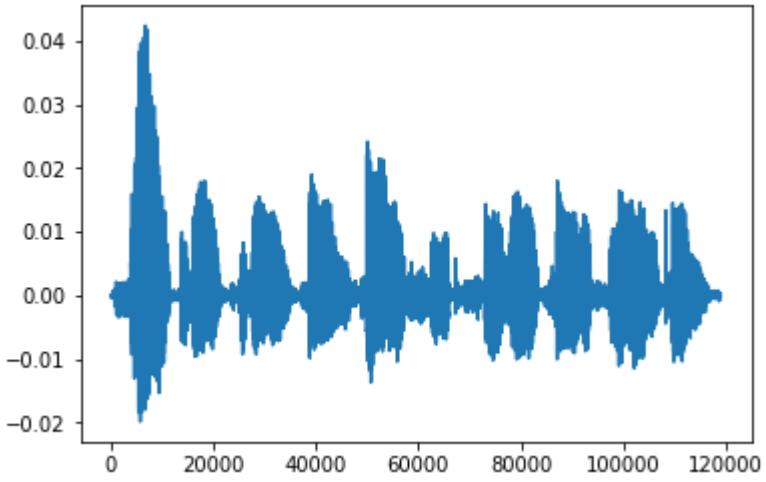




Aceasta amestecare simuleaza un caz din viata reala cand primul microfon se afla la o distanta echidistanta de cele doua surse de semnal, iar al doilea microfon se afla mai aproape de semnalul cu vocea spaniola, pe fundal auzindu-se si vocea engleza.

Rezultatul separarii este:

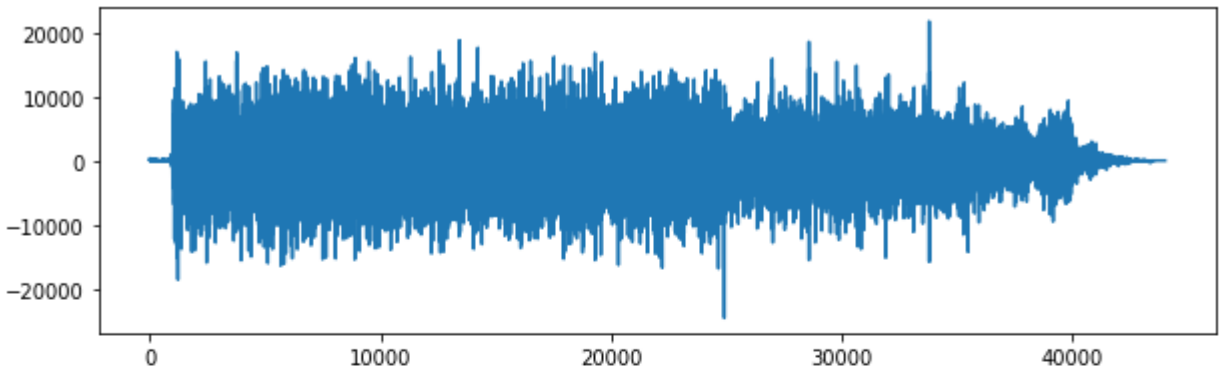


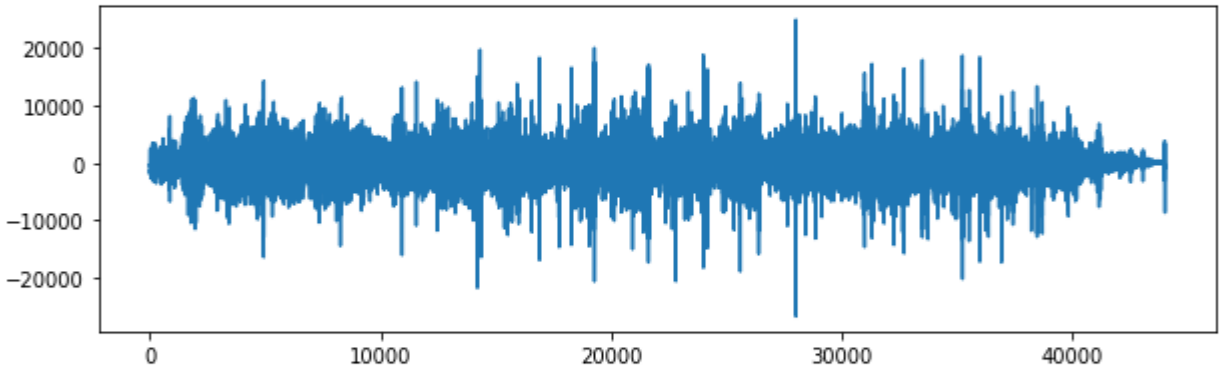


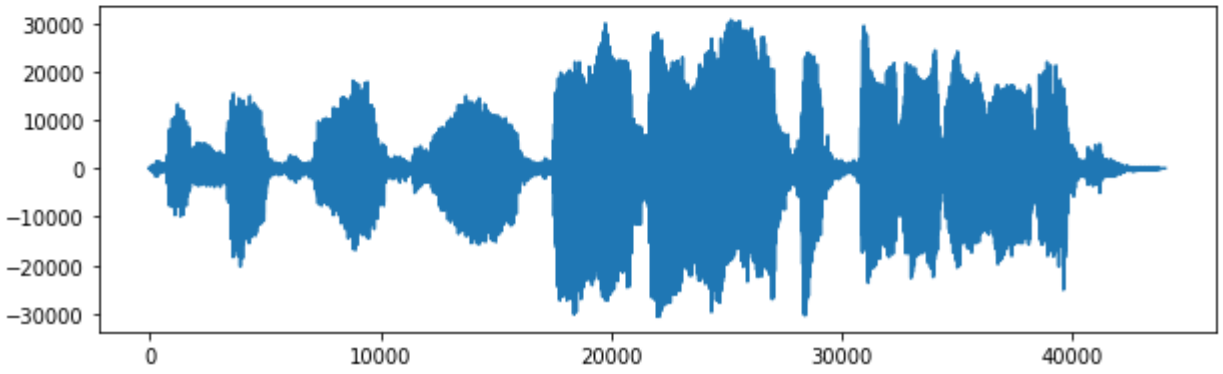
Coeficientul de corelatie al surselor separate este $5.62035243305e-08$, un coeficient foarte mic, care arata o separare buna, acest lucru se observa si din imagini, se poate face clar diferenta intre cele doua voci, si se vede asemanarea cu semnalele originale. De asemenea, se poate vedea ca algoritmul nu pastreaza ordinea componentelor initiale, ele ies din algoritm in ordine inversa, lucru sustinut de teorie.

4.10.4 Semnale audio: trei semnale

Acest test a fost facut pentru a demonstra faptul ca ICA poate separa mai mult de doua surse independente, si a fost realizat cu trei semnale audio: un aspirator in functiune, o serie de aplaude si o voce:



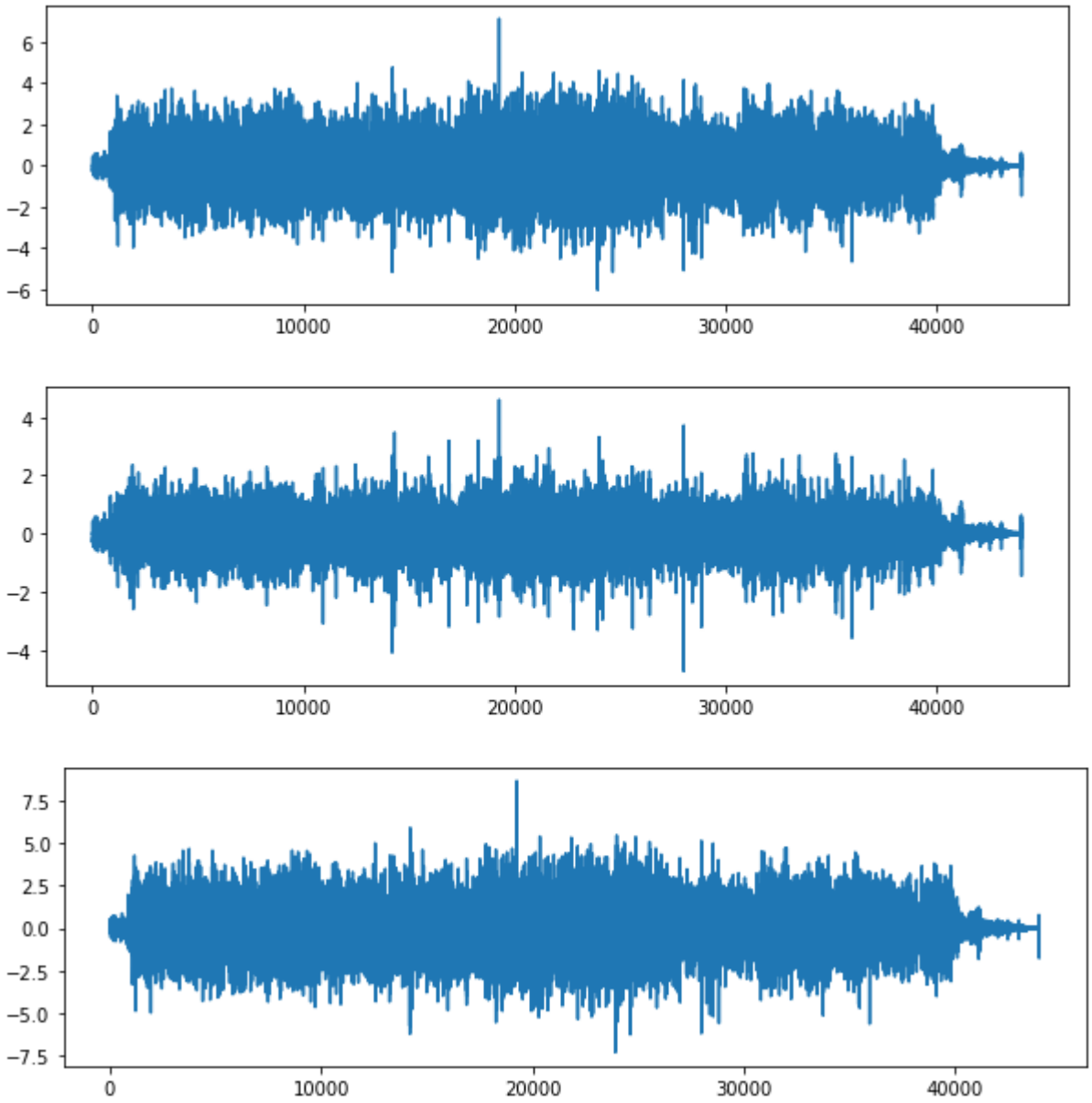




Matricea de amestecare A este:

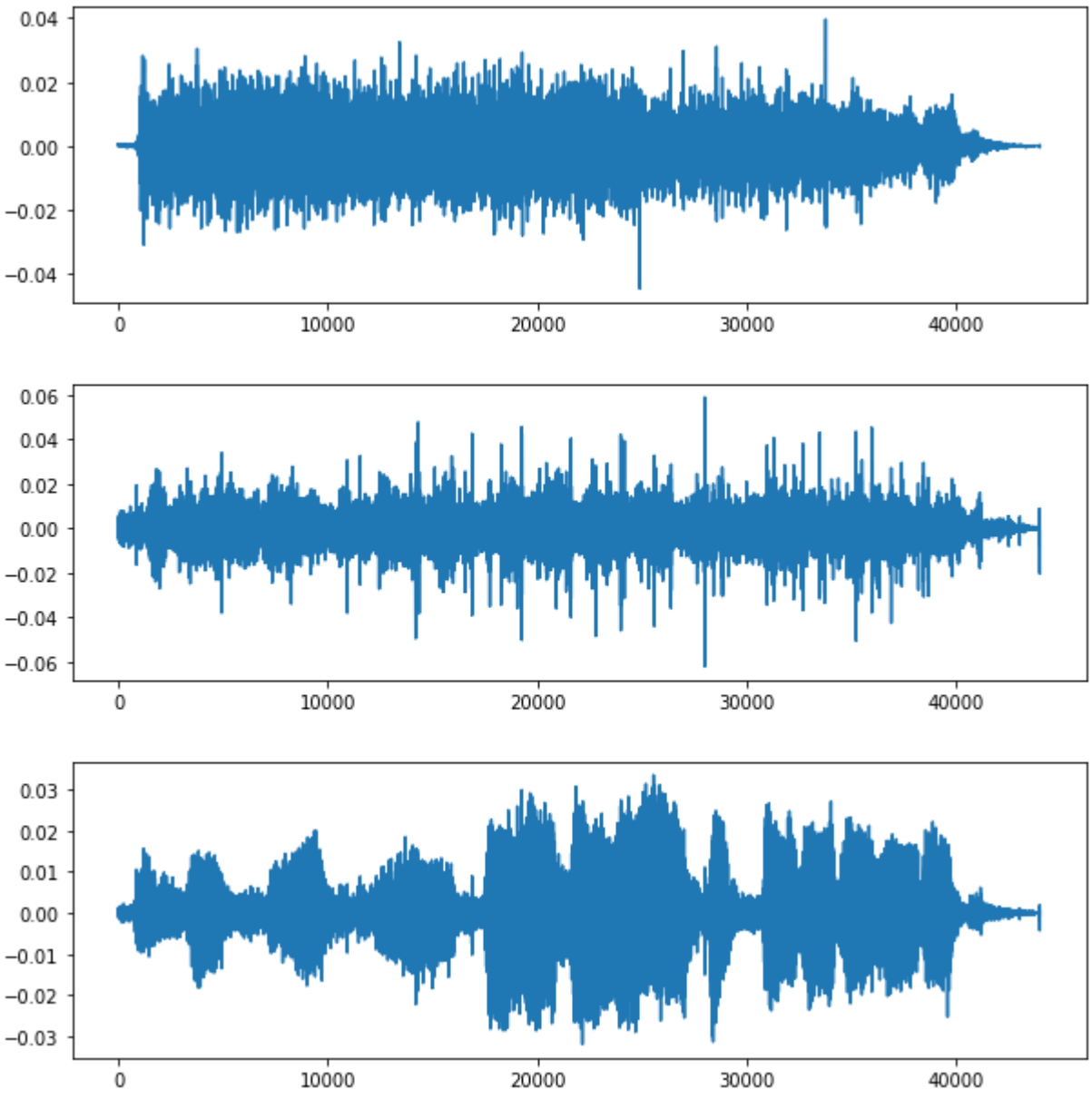
$$\begin{matrix} 0.77191 & 0.4752 & 0.58699 \\ 0.33712 & 0.47563 & 0.20836 \\ 0.96878 & 0.57618 & 0.65625 \end{matrix}$$

Rezultatul amestecarii este:



La fel ca in testele de pana acum, amestecarea rezulta in semnale care sunt

foarte similare. Dupa executarea algoritmului, semnalele separate rezultate sunt:

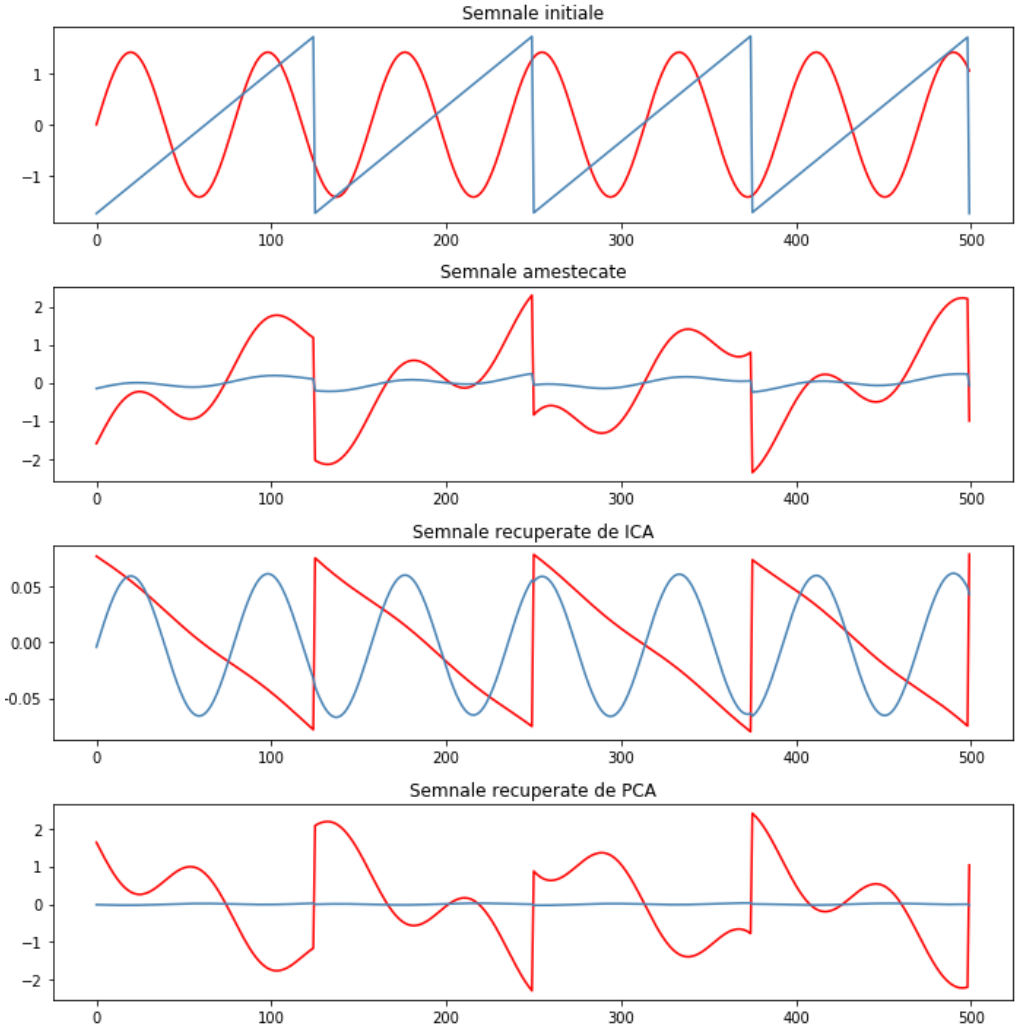


4.10.5 Analiza coeficientilor de corelatie a testelor

Uitandu-ne la coeficientii de corelatie ale surselor separate, se poate observa ca necesitatea ca ei sa fie mai mici, pentru o separare mai buna, creste, si asta deoarece complexitatea surselor amestecate este mai mare. Daca pentru semnalele primitive a rezultat un coeficient de corelatie de ordinul $5e^{-4}$, pentru semnalele audio reale acesta a fost si mai mic, $3e^{-6}$ in cazul muzicii amestecate cu voce, si $5e^{-8}$ in cazul vocilor amestecate. Asta arata ca cu cat semnalele initiale sunt considerate mai *asemanatoare* (putem spune ca doua voci sunt mai asemanatoare decat o voce si o melodie acustica), cu atat indicele de corelatie al semnalelor separate este mai mic.

4.10.6 ICA versus PCA

Deoarece am abordat si tema PCA in aceasta lucrare, am dorit sa efectuam un test in care comparam puterea de separare dintre ICA si PCA, in contextul componentelor independente. Din fericire, cei care au realizat biblioteca scikit-learn pentru Python au efectuat aceasta comparatie pentru noi. Acestia au folosit implementarile lor pentru algoritmii PCA si ICA, si deoarece am demonstrat ca implementarile noastre functioneaza conform teoriei, am decis ca nu mai este necesar sa efectuam acest test si cu algoritmii nostrii. Acestea sunt rezultatele:



Se poate observa cum abordarea PCA nu reuseste sa separe componentele independente, in asemenea cazuri, abordarea ICA este mai buna.

4.10.7 Problema ICA in lumea reala

Pana acum, ipoteza teoriei descrise a fost faptul ca semnalele sunt amestecate instant, in diferite proportii. Acest lucru in lumea reala ar insemna ca semnalele audio pornesc de la sursa si ajung la receptori instant, luandu-se

in calcul distanta. Se poate observa clar problema: in lumea reala exista un timp de parcurgere al semnalului, care variaza atat in functie de distanta dintre surse si receptori, cat si in functie de tipul mediului prin care se face propagarea. Aceasta intarziere poate fi de la mai putin de o milisecunda, pana la mai multe milisecunde. Avand in considerare faptul ca o secunda inregistrata de un receptor audio poate contine 16000 de esantioane, o intarziere de o milisecunda intre ajungerea unui semnal la cei doi receptori duce la o defazare de 16 esantioane intre inceputul inregistrarilor semnalului. Aceasta diferenta este destul de mare incat rezultatul obtinut de algoritm sa nu fie cel dorit.

4.11 Aplicatii ale ICA

4.11.1 Separarea artefactelor din inregistrari MEG

Magnetoencefalografia (MEG) este o procedure neinvaziva in care este monitorizata activitatea neuronilor corticali cu o precizie foarte buna din punct de vedere al timpului, si cu precizie moderata din punct de vedere spatial. Cand se analizeaza date MEG, o problema comuna este extractia caracteristicilor semnalelor neuromagnetice in prezenta artefactelor. Amplitudinea artefactelor ar putea fi mai mare decat cele ale semnalelor creierului, acestea putand fi interepretate ca semnale patologice, adica ca semnale care pot cauza sau descriu o anumita afectiune.

O posibila rezolvare a acestei probleme este considerarea faptului ca activitatea cerebrala si artefactele, fie ele miscari ale ochilor sau senzori defecti, sunt procese separate din punct de vedere anatomic si fiziologic, si aceasta separare este data de independenta statistica dintre semnalele magnetice generate de aceste procese.

Semnalele MEG au fost inregistrate intr-o camera izolata magnetic, cu un neuromagnetometru *Neuromag-122* cu 122 de canale, care acopera intreg scalpul pacientului. Aparatul inregistreaza date din 61 de locatii. Persoana de test a fost rugata sa clipeasca si sa faca miscari rapide si bruste cu ochii, pentru a produce artefacte oculare tipice. Pentru a produce artefacte musculare, acesta a fost rugat sa muste in gol, pana la 20 de secunde. Pe langa

asta, un alt artefact a fost creat plasand un ceas digital la un metru de casca, inauntrul camerei izolate.

5 Concluzii //TODO

6 Bibliografie //TODO

References

[1] Aapo Hyvärinen and Erkki Oja. Independent component analysis: Algorithms and applications. 2000.

[1]