Separarea surselor

Motrescu Radu

Contents

1 Introducere in separarea surselor //TODO

2 PCA

3 Kernel PCA

3.1 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.

3.2 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.

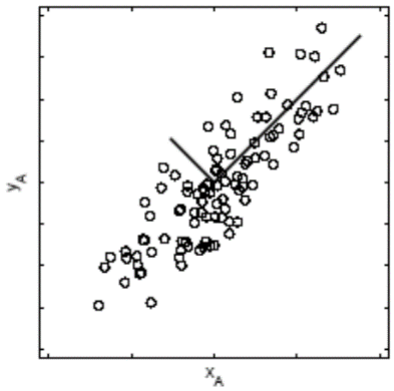
3.2.0.1 Un scurt exemplu: Sa presupunem ca am extras informatii pentru 100 de parametri pentru un student: inaltime, varsta, greutate, nota obtinuta la un test, culoarea parului etc. Vrem sa gasim cele mai importante caracteristici care definesc studentul. Cum facem asta? Folosim PCA pentru a selecta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

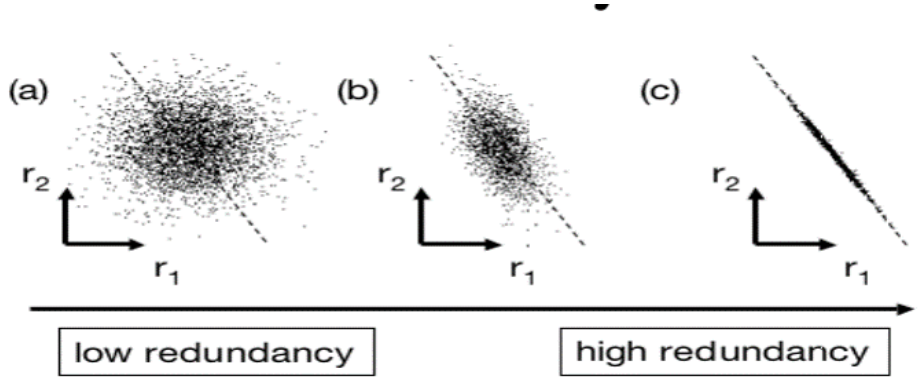
A putea elimina parametrii irelevanti si a-i pastra pe cei relevanti este usor pentru un om, el putand vedea clar care parametri nu exprima informatii relevante despre subiectul respectiv, dar cum putem face calculatorul sa isi dea seama de acesti parametri? Folosind matematica, desigur!

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.



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

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

3.2.0.3 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.

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

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

3.2.0.4 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.

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

3.2.0.5 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.

3.2.0.6 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

RowFeatureVectore 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.

3.3 Jurnalul dezvoltarii aplicatiei

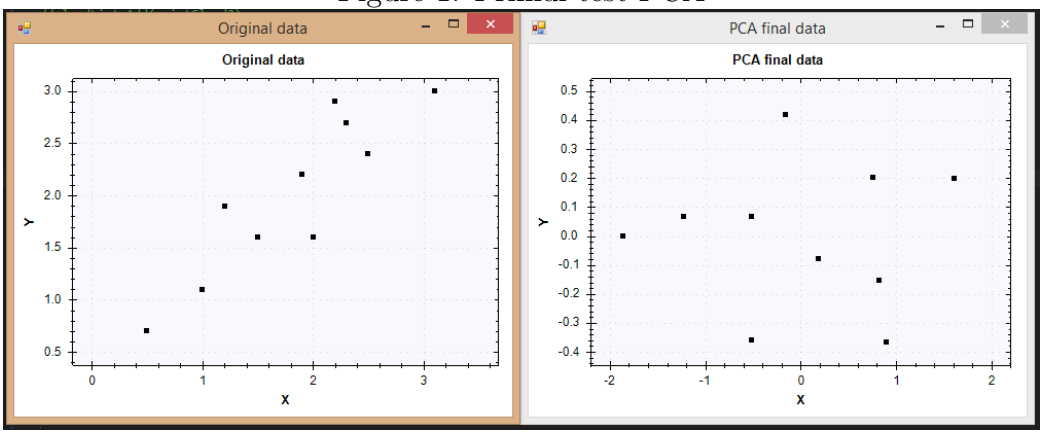
3.4 Aplicatia AccordPCA

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);

}
```

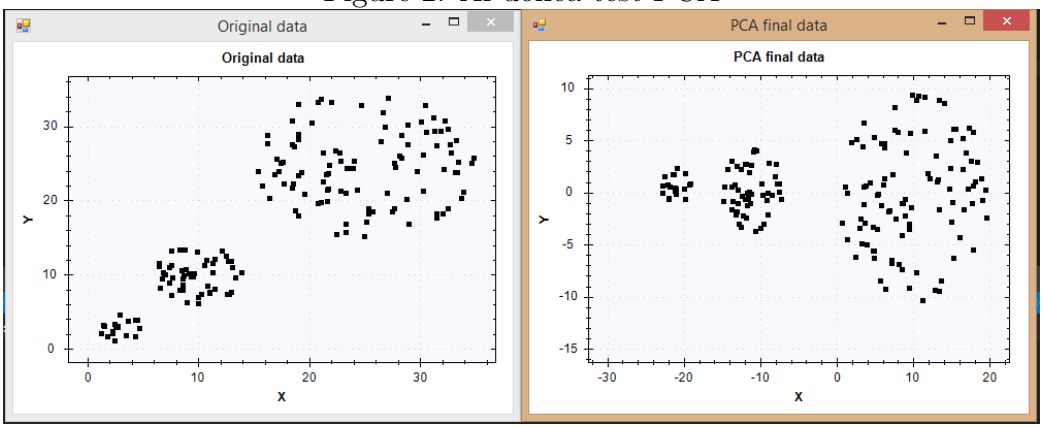
Pentru a testa acest algoritm, am introdus prima data un set de date deja testat de catre altcineva, unde se poate observa clar rezultatul algoritmului. (referinta catre http://dai.fmph.uniba.sk/courses/ml/sl/PCA.pdf)

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.

3.5 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.

3.5.1 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

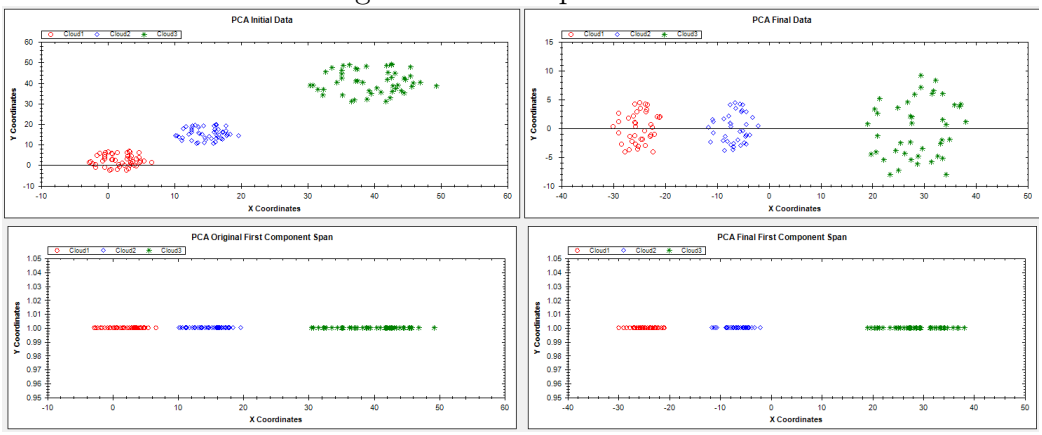


Figure 4: Nori interesectati partial

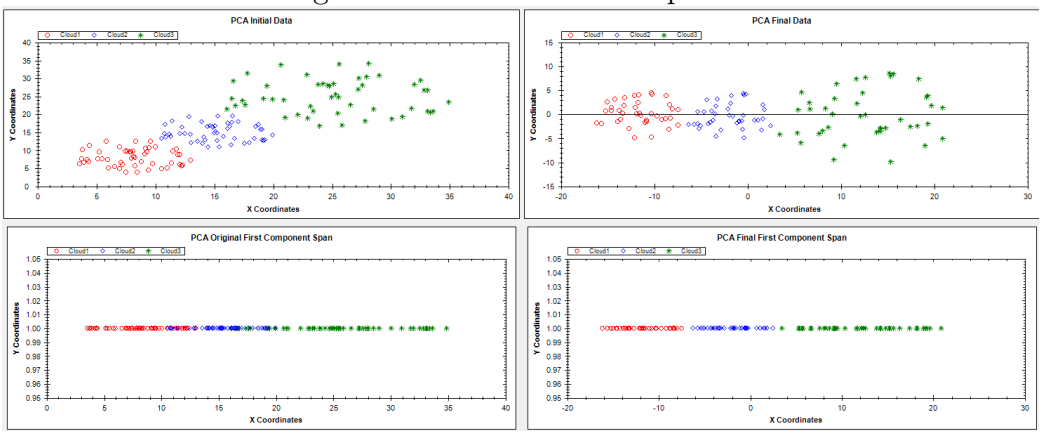
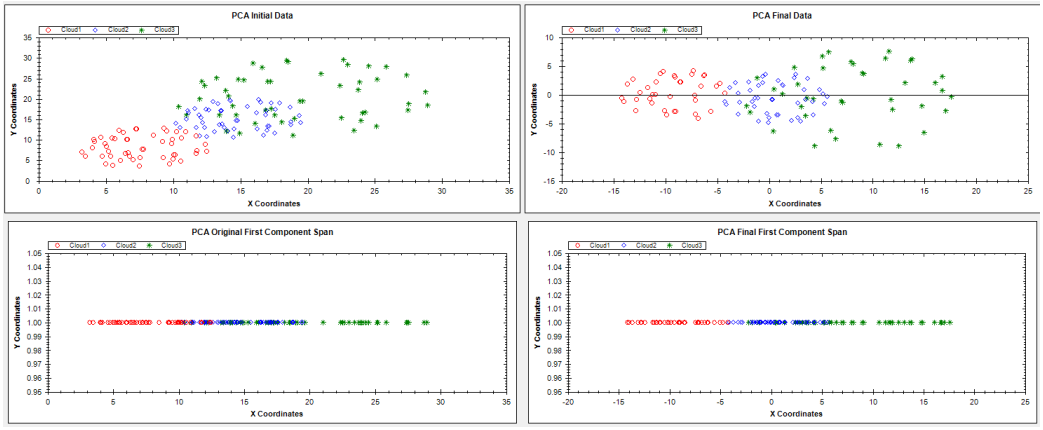


Figure 5: Norul albastru si verde intersectati aproape total



3.5.2 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

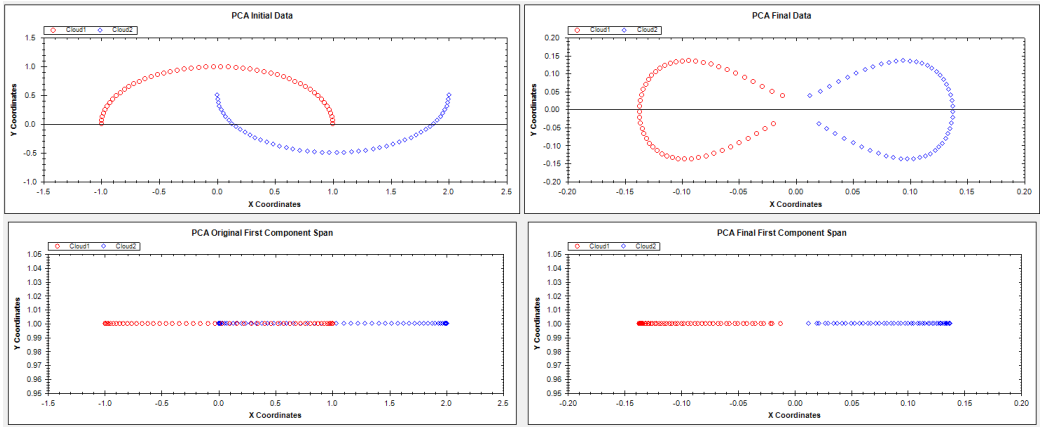
Deoarece nu putem garanta ca matricea e centrata, vom aplica urmatoarea formula pentru a o centra:

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

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, $\gamma = 15$



3.5.3 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

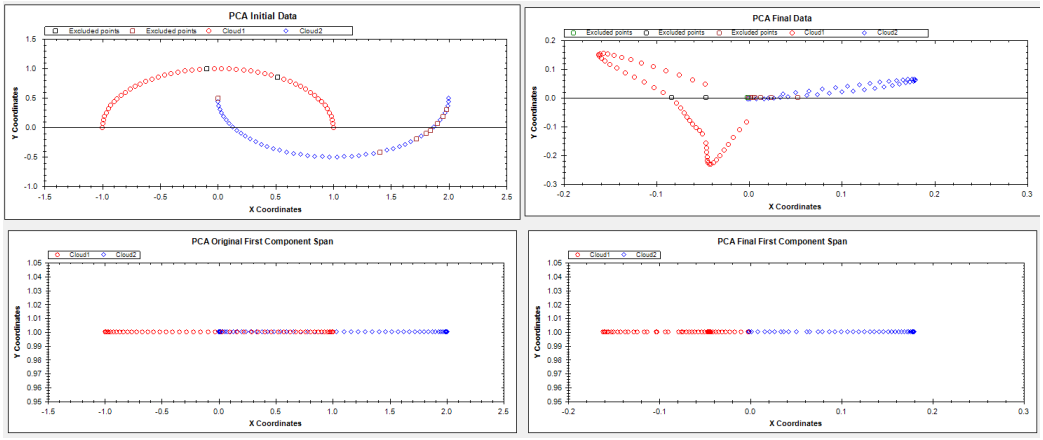
-cu punctul de separare gasit, cele 10 puncte proiectate pe prima componenta principala pot fi categorizate: daca se afla in stanga punctului de separare el va fi in norul albastru de puncte, iar daca este la dreapta punctul de separare, va apartine norului rosu

-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



3.6 Aplicatia Eigenfaces

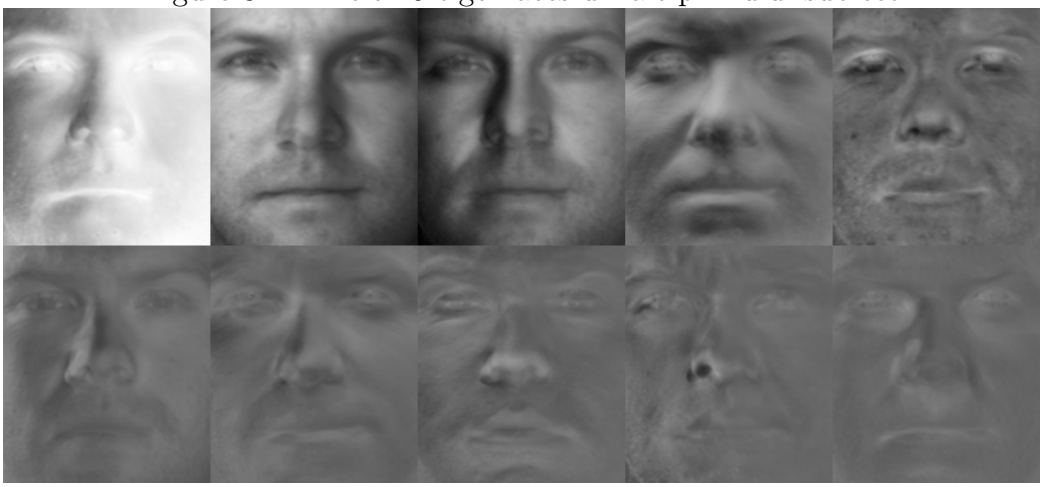
Dupa ce am realizat testele KPCA, urmatorul pas a fost sa crestem dimensiunile cu care lucram. Mai exact, am inceput sa folosim imagini pentru a vedea cum actioneaza algoritmul PCA pe ele.

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



3.6.1 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.

3.6.2 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.

[1] [2]

4 ICA //TODO

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).

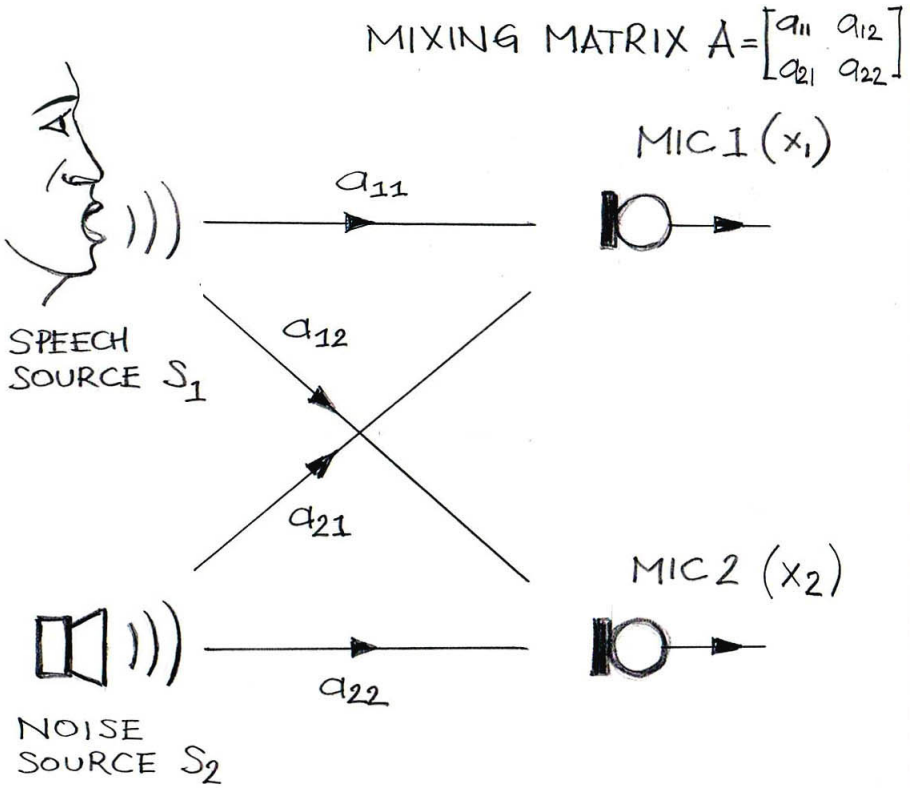
Pentru a putea vizualiza aceasta idee, putem analiza o problema din lumea reala. Sa spunem ca suntem intr-o camera in care 2 oameni vorbesc simultan. Avem 2 microfoane pe care le pozitionam in locuri diferite in camera, si ele ne dau 2 inregistrari ale semnalelor. Aceste semnale sunt $x_1(t)$ si $x_2(t)$, 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.

5 Concluzii //TODO

6 Bibliografie //TODO

References

[1] Nobody Jr. My article, 2006.

[2] Mr. X. Something great. 2005.