Introducción al Análisis de Componente Principal y al Análisis de Componente Independiente usando Scilab

Raúl Ramírez Ramírez[*]

Instituto Politecnico Nacional
IPN

Unidad Profesional
Interdisciplinaria de Ingenieria y Tecnologias Avanzadas
UPIITA

Verano de Investigación 2008

Resumen

El Análisis de Componente Principal (PCA), y el Análisis de Componente Independiente (ICA) son dos técnicas que han ayudado a realizar avances en diferentes áreas de investigación; el presente artículo pretende dar a conocer al lector éstos métodos así como la forma en que se aplican, la efectividad que tiene cada uno y mensionar algunos casos de aplicación. Además se presentará el uso de un software matemático bajo la licencia GNU/GPL llamado Scilab, mediante el cual se implementarán las operaciones necesarias para llevar a cabo cada uno de los métodos anteriores.

[*] rorjc2000@hotmail.com

1. Introducción

En cuestiones de investigación, suele ser necesario realizar el analisis de señales o arreglos de datos que han sido recavados de manera estadística, presentandoce en los datos diferentes tipos ruido debido a cuestiones ambientales, estado y calidad de los instrumentos de adquisición y/o medición, así como a condiciones propias del sistema objeto de dicha investigación. Debido a lo anterior, ha sido necesario implementar diferentes métodos y algoritmos, para eliminar ruido de las señales muestreadas, pero además éstos algoritmos, no sólo son útiles para eliminar señales no deseadas, sino que tambien se tiene la posibilidad de reconocer vestigios correspondientes a otro tipo de datos intrínsecos en el muestreo original de la señal, la aplicación de los métodos de Análisis de Componente Principal y Análisis de Componente Independiente (de aqui en adelante se hara referencia a cada uno de los métodos por PCA e ICA respectivamente, debido a sus siglas en ingles) ha resultado de interés para varios investigadores en diferentes ramas, debido a la efectividad que cada una ha logrado en diferentes aplicaciones y los metodos estadísticos en los que se basan; existen diferentes tipos de software que pueden resultar muy útiles como por ejemplo, MATLAB, MATHEMATICA, OCTAVE y SCILAB, funcionando los dos primeros bajo licencias de propietario (software no libre) y los dos últimos bajo la licencia GNU/GPL (software libre), siendo de particular interes en este articulo el último, debido a su tipo de licencia y no presentar algunos inconveientes en sus funciones de graficación como Octave, aunque cabe mencionar que a diferencia de Scilab, Octave es más compatible con Matlab. Teniendo en cuenta lo anterior, se introducirá al lector en el conocimiento de estos dos métodos, al tiempo que se familiariza con la forma de trabajo de Scilab.

2. Sobre Scilab

Scilab es un software matemático, de uso general y de libre acceso, ya que ha sido desarrollado bajo la licencia GNU/GPL. En el podemos llevar a cabo desde las más sencillas operaciones hasta los cálculos más robustos que podamos imaginarnos. Comparado con Octave y Matlab, su sintaxis y varias de sus funciones son las mismas, he incluso permite guardar las variables con las que se esta trabajando en un formato propio de Matlab para poder compartirlas con éste programa. Si el lector está familiarizado con alguno de éstos programas, no le será difícil acostumbrarse a Scilab.

Se mencionarán algunas funciones que se utilizarán a lo largo del artículo con el fin de que los lectores que no estén familiarizados con ellos, puedan entender mejor como es que se llevan a cabo las operaciones en el programa.

2.1. Asignación

Se lleva a cabo con el signo =
A=2

2.2. Ocultar operación ;

Se utiliza generalmente al final de una línea de comandos, ésta puede usarse o no, si es usada al final de cada instrucción, ocultará el resultado de ésta; si no se utiliza, imprimirá en pantalla el resultado de la operación, esto nos permite decidir que salidas ver en la pantalla para tener cierto control sobre el programa cuando aún nos encontramos probando éste.

2.3. Arreglos-[ ]

Para realizar un arreglo de cualquier tipo debemos ponerlo dentro de *[ ]*.
Para Generar una matrizdebe seguirse la siguiente estructura:

A=[a,b c; d,e f; g h,i]

La instrucción anterior generará una matriz de tres por tres, la coma y el espacio, son utilizados para separar un elemento de otro y pueden ser usados insdistintamente como se muestra, cada quien es libre de elegir usarlos o no; por otro lado, el punto y coma hace referencia a un fin de fila y al inicio de una nueva fila

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \tag{1}$$

2.4. size()

Al aplicar éste comando a una variable, Scilab devolverá las dimenciones de dicha variable, si se desae utilizar dichas dimenciones para algún fin en particular, podemos capturarlas asignando un arreglo al comando *size* de la forma:

[f c]=size(A)

Ésta instrucción devolverá las dimensiones del arreglo A en número de filas y columnas, en las variables f y c respectivamente, pero ésto sólo funcionará si el arreglo es de dos dimensiones.

2.5. zeros()

Genera un arreglo lleno de ceros, su sintaxis es la siguiente:

A=zeros(m,n)

En este caso A representará un arreglo de $m*n$ lleno de ceros.

2.6. ones()

Genera un arreglo lleno de unos, su sintaxis es la siguiente:

A=ones(m,n)

En este caso A representará un arreglo de $m*n$ lleno de unos.

2.7. Manejo de arreglos

Sea A un arreglo de *3*3*, para manejar el contenido del arreglo basta con mensionar que se puede hacer referencia mediante la sintaxis:

A(m,n)

siendo m un número de fila y n un número de columna, obteniendo así el valor contenido en esta posicion del arreglo.

2.8. El comodín - :

Este comodin resulta de gran utilidad cuado se desea utilizar TODA una fila o TODA una columna de un arreglo, la forma de implementarlo es la siguiente:

y=A(:,n)

La asignación anterior crea un nuevo arreglo llamado y formado por todas las filas de la columna n, dicho de otra manera, por toda la coumna n.

También puede ser interpretado como un 'A' como se muestra a continuación:

w=[0:9]

El código anterior crea un vector fila llamado w que contiene los números de 0 HASTA 9. En el manejo de un arreglo se haría referencia a las filas o columnas según la posición en la que se encuentre.

y=A(1:2,n)

En éste caso y contiene las dos filas de la columna n del arreglo A. Puede emplearse durante una asignación como se ilustra a continuación:

y(1:4,:)=A(5:8,:)

Aquí se están asignando todas las columnas de las filas 5 a 8 de A en las filas 1 a 4 de y. Es facil despues de éstos ejemplos imaginar porque el pseudónimo de 'comodín'.

2.9. sum()

Obtiene la suma de un arreglo que le sea asignado:

sum(A)

Realiza la suma de todos los elementos de A.

sum(A(n,:))

Realiza la suma de todos los elementos de la fila n de A.

2.10. max()

Obtiene el valor máximo de un arreglo dado.

max(A)

2.11. min()

Obtiene el valor máximo de un arreglo dado.

min(A)

2.12. inv()

Obtiene la inversa de un arreglo dado.

inv(A)

2.13. clear

Borra una variable o elemento especificado.

clear A

2.14. El operador - ’

Calcula la matriz transpuesta del operador al que se aplica
A’

2.15. for

Estructura del cilo for
for {condiciones}
. {sentencias}
end

2.16. if

Estructura del cilo if
if {condiciones}
. {sentencias}
end

3. Análisis de Componente Principal

Un repaso de los conocimientos previos sobre Estadística y Álgebra Matricial utilizados en estos métodos se puede encontrar de manera clara y concreta en [1] capítulo dos y de manera independiente y más extensa en [2], [3] y [4], [5] respectivamente.

Como ya se mencionó, el PCA es una forma de identificar patrones en datos y representar los datos en una forma tal que sus semajanzas y diferencias sean resaltadas. Éste método resulta de gran utilidad cuando es necesario buscar cierto tipo de información como ruido u otras señales de interés en muestreos de grandes dimensiones.

Una ventaja más es que una vez que se han encontrado los patrones en los datos, es posible comprimir éstos, a través de una reducción del número de dimensiones sin perder demasiada información.

A grandes rasgos, el algoritmo PCA toma una arreglo de variables y define nuevas variables que son combinaciones lineales de las variables originales. Éste método supone que las variables de entrada son correlacionadas.

3.1. Descripción del Método

El primer paso, es obtener la media correspondiente a cada una de las variables del arreglo de datos, y llevar a cabo un ajuste de los datos, restando a cada uno la media correspondiente a su variable, lo siguiente es formar una matriz de covarianza, en seguida, se procede a calcular los eigenvectores y los eigenvalores de la matriz de covarinza, una vez que se tienen éstos, debemos construir un vector característico que es sólo otra manera de representar la matriz de eigenvectores organizando éstos de mayor a menor a manera de columnas según los eigenvalores de la matriz de covarianza; otra opción para llevar a cabo lo anterior consiste en despreciar los eigenvectores menos significativos (el realizar lo anterior se verá reflejado en una reducción de dimensiones, lo que implicará perdida de datos, en éste caso los menos significativos). El paso final consiste en obtener la matriz transpuesta del vector característico y multiplicar ésta por la transpuesta de la matriz que contiene los datos iniciales ajustados, esta nueva matriz se llamará matriz de datos finales, matriz final de transformación o simplemente matriz final.

Para obtener nuevamente los datos iniciales, obtenemos, la inversa de la matriz transpuesta de los vectores característicos y multiplicamos ésta a la derecha de la matriz de final, pero además debemos recordar sumar la media de los datos originales, ya que como se mencionó, ésta se resta en al inicio del procedimiento.

Ahora llevaremos a cabo la implementación del algoritmo para un arreglo de tres dimensiones.

Sea entonces, A nuestro arreglo inicial de datos:

$$A = \begin{array}{c} \begin{matrix} \text{x} & \text{y} & \text{z} \end{matrix} \\ \begin{bmatrix} 0{,}7417 & 0{,}8145 & 1{,}7603 \\ 2{,}6537 & 0{,}8115 & 1{,}5333 \\ 0{,}0007 & 0{,}7597 & 0{,}9452 \\ 1{,}1594 & 3{,}1006 & 2{,}2203 \\ 2{,}3354 & 2{,}2903 & 1{,}4222 \\ 2{,}2056 & 1{,}0797 & 3{,}2238 \\ 2{,}9826 & 3{,}2746 & 0{,}1535 \\ 2{,}4069 & 0{,}7532 & 1{,}6912 \\ 3{,}0825 & 1{,}0973 & 0{,}9264 \\ 0{,}2399 & 1{,}2693 & 1{,}4559 \\ 1{,}9685 & 1{,}0257 & 0{,}9850 \\ 2{,}3248 & 1{,}9881 & 0{,}4493 \\ 2{,}5494 & 1{,}6940 & 2{,}7318 \\ 0{,}6967 & 1{,}1659 & 0{,}7437 \\ 1{,}9103 & 2{,}0832 & 0{,}3935 \end{bmatrix} \end{array} \tag{2}$$

Éste arreglo se ha obtenido mediante la función *rand(n,m)* de Scilab, la cual genera números aleatorios según le sea especificado, ésta función se utiliza como se explica a continuación.

La función *rand(3)* generará la siguiente salida en la pantalla de Scilab:

ans=

. *0.6672110*

Ésta salida representa el cálculo del tercer número aleatorio genereado por la función *rand()* mientras que *rand(1,3)* generará un vector fila compuesto por tres números aleatorios.

ans=

. *0.6857310 0.00071234 0.33405671*

Por tanto la funcion *rand(N,M)* en general regresará una matriz de NxM números aleatorios; para nuestro caso, se ha utlizado *rand(15,3)*.

Ahora debemos obtener las tres medias correspondientes a cada una de las variables de nuestro arreglo, denominaremos *medx* a la media correspondiente de la variable x, *medy* a la media correspondiente de la variable y, *medz* a la media correspondiente de la variable z, pero primero debemos conocer el número de datos totales por columna. La sintaxis en Scilab sería:

```
[f c]= size(A)
```

De ésta manera obtenemos en la variable f el número de filas de la matriz y en c el número de columnas de la misma. Continuando con lo anterior, se recomienda realizar un programa que lleve a cabo el calculo de las medias para un arreglo cual quiera que éste sea mediante la utilizacion de ciclos *for* como se muestra acontinuacón:

```
[f c]=size(A);              //aquí se obtiene la dimensión de la matriz filas-columnas

. medias=zeros(1,c)      //se crea un vector que contendrá todas las medias

. for i=1:c               //este ciclo for es el encargado de llenar la matriz
.   medias(1,i)=sum(A(:,i))/f;                //de las medias con los valores
. end                     //correspondientes a cada variable
```

Este es el resultado en la consola de Scilab:

$$medias = \begin{pmatrix} 1{,}8172 & 1{,}5471 & 1{,}3756 \end{pmatrix} \tag{3}$$

A continuación se restan las medias a su columna correspondiente para obtener de ésta manera el ajuste de los datos, aplicando ésto mediante un ciclo *for* nuevamente; en éste punto creamos una matriz llamada Ad identica a A, que contendrá los datos ajustados.

```
Ad=A;

for i=1:c
.    Ad(:,i)=Ad(:,i)-medias(i);
end
```

$$Ad = \begin{bmatrix} -1{,}1075 & -0{,}7326 & 0{,}3846 \\ 0{,}8364 & -0{,}7356 & 0{,}1576 \\ -1{,}8165 & -0{,}7874 & -0{,}4304 \\ -0{,}6578 & 1{,}5534 & 0{,}8446 \\ 0{,}5181 & 0{,}7431 & 0{,}0465 \\ 0{,}3883 & -0{,}4674 & 1{,}8481 \\ 1{,}1653 & 1{,}7274 & -1{,}2221 \\ 0{,}5896 & -0{,}7939 & 0{,}3155 \\ 1{,}2652 & -0{,}4498 & -0{,}4492 \\ -1{,}5773 & -0{,}2778 & 0{,}0802 \\ 0{,}1512 & -0{,}5214 & -0{,}3906 \\ 0{,}5075 & 0{,}4409 & -0{,}9263 \\ 0{,}7321 & 0{,}1468 & 1{,}3561 \\ -1{,}1205 & -0{,}3812 & -0{,}6319 \\ 0{,}0930 & 0{,}5360 & -0{,}9821 \end{bmatrix} \tag{4}$$

Ahora debemos calcular la matriz de covarianza, nuevamente implementando en Scilab el algoritmo para el cálculo de ésta, obtenemos el siguiente código:

```
cov=zeros(c,c);
. for j=1:c
.   for i=1:c
.     cov(j,i)=sum(Ad(:,j).*Ad(:,i))/f;
.   end
. end
```

$$cov = \begin{pmatrix} 0{,}9255 & 0{,}1952 & -0{,}0110 \\ 0{,}1952 & 0{,}6464 & -0{,}1366 \\ -0{,}0110 & -0{,}1366 & 0{,}7003 \end{pmatrix} \tag{5}$$

Para obtener los eigenvalores y los eigenvectores de la matriz de covarianza usamos el comando

```
[eigevec, eigval]=spec(cov);
```

Obteniendo en *eigvec* los eigenvectores y en *eigval* los eigenvalores:

$$eigval = \begin{pmatrix} 0{,}4813 & 0 & 0 \\ 0 & 0{,}7484 & 0 \\ 0 & 0 & 1{,}0426 \end{pmatrix} \tag{6}$$

$$eigvec = \begin{pmatrix} -0{,}3416 & 0{,}4184 & -0{,}8415 \\ 0{,}8050 & -0{,}3316 & -0{,}4917 \\ 0{,}4848 & 0{,}8455 & 0{,}2235 \end{pmatrix} \tag{7}$$

Necesitamos ahora reordenar nuestra matriz de eigenvectores de mayor a menor según los eigenvalores, para generar nuestro vector característico. Hasta éste punto es evidente que la columna perteneciente a z tiene el mayor eigenvalor, mientras que la columna de x posé el menor, sin embargo para generalizar el método, necesitamos crear un código que realice ésto sin importar en que posición se encuentren los eigenvalores ni la dimensión de la matriz *eigvec*.

Para poder manejar mejor la matriz de eigenvalores con el fin de crear el vector característico, transformaremos ésta en un vector que contendrá los eigenvalores.

```
auxeigvec=zeros(1,c);
. for i=1:c
.     auxeigval(1,i)=sum(eigval(:,i));
.  end
```

Se ha declarado una variable auxiliar (*auxeigval*), en la cual se guardarán los eigenvalores a manera de vector una vez terminado el ciclo *for*.

$$auxeigval = \begin{pmatrix} 0{,}4813 & 0{,}7484 & 1{,}0426 \end{pmatrix} \tag{8}$$

Para ordenar la matriz necesitaremos de una variable auxiliar más (*aux1*), la cual deberá tener el mismo número de columnas que la matriz de eigenvectores, ya que en ésta se guardará de manera ordenada, las posiciones correspondientes a cada eigenvalor en *eigval*, ordenados de mayor a menor. Al final podremos borrar si así lo deseamos la variable *auxeigval* ya que se encontrará vacía y no la volveremos a usar.

```
aux1=zeros(1,c);
. for j=1:c
.     for i=1:c
.         if max(auxeigval)== auxeigval(i)
.             aux1(j)=i;
.         end
.     end
.     auxeigval(aux1(j))=0;
. end
. clear(auxeigval);
```

$$aux1 = \begin{pmatrix} 3 & 2 & 1 \end{pmatrix} \tag{9}$$

A partir de aquí, ya podemos formar nuestro vector característico, reordenando de mayor a menor los eigenvectores, segun la magnitud de sus eigenvalores; sea entonces *vectcar* nuestro vector característico. El siguiente código se encarga de llevar a cabo éste ordenamiento.

```
vectcar=zeros(size(eigvec));
```

```
.   for i=1:c
.      vectcar(:,i)=eigvec(:,aux1(i));
.   end
```

El vector característico, luce de la siguiente manera:

$$vectcar = \begin{pmatrix} -0{,}8415 & 0{,}4184 & -0{,}3416 \\ -0{,}4917 & -0{,}3316 & 0{,}8050 \\ 0{,}2235 & 0{,}8455 & 0{,}4848 \end{pmatrix} \tag{10}$$

Es en éste punto del procedimiento donde se decide si es conveniente llevar a cabo una reducción de dimenciones o no, esto se realiza eliminando la ó las columnas menos significativas según nuestro critério y conveniencia, para el caso particular expuesto en el presente artículo, no se llevara a cabo dicha reducción por el momento, pero se realizará despues a manera demostrativa, eliminando la última columna.

De ésta manera, hemos llegado al último paso del algoritmo, donde sólo es necesario obtener la matriz transpuesta del vector característico (vectcar) y multiplicarla por la matriz transpuesta de los datos iniciales ajustados (Ad) para así obtener la matriz final de datos (mfd).

```
mfd=vectcar' * Ad';
```

$$mfd' = \begin{pmatrix} 1{,}35136 & 0{,}1181 & -0{,}0359 \\ -0{,}30693 & 0{,}7272 & -0{,}8016 \\ 1{,}81968 & -0{,}8628 & -0{,}2221 \\ -0{,}02153 & -0{,}0763 & 1{,}8849 \\ -0{,}79112 & 0{,}0096 & 0{,}4437 \\ 0{,}31617 & 1{,}8801 & 0{,}3870 \\ -2{,}10343 & -1{,}1187 & 0{,}3999 \\ -0{,}03527 & 0{,}7768 & -0{,}6876 \\ -0{,}94400 & 0{,}2987 & -1{,}0123 \\ 1{,}48194 & -0{,}5000 & 0{,}3540 \\ 0{,}04178 & -0{,}0940 & -0{,}6609 \\ -0{,}85108 & -0{,}7171 & -0{,}2676 \\ -0{,}38522 & 1{,}4042 & 0{,}5256 \\ 0{,}98917 & -0{,}8767 & -0{,}2305 \\ -0{,}56150 & -0{,}9693 & -0{,}0765 \end{pmatrix} \tag{11}$$

Si se hubiera llevado a cabo una reducción de dimensiones eliminando la columna de x, *mfd'* no contendría la última columna.

$$mfd' = \begin{pmatrix} 1{,}35136 & 0{,}1181 \\ -0{,}30693 & 0{,}7272 \\ 1{,}81968 & -0{,}8628 \\ -0{,}02153 & -0{,}0763 \\ -0{,}79112 & 0{,}0096 \\ 0{,}31617 & 1{,}8801 \\ -2{,}10343 & -1{,}1187 \\ -0{,}03527 & 0{,}7768 \\ -0{,}94400 & 0{,}2987 \\ 1{,}48194 & -0{,}5000 \\ 0{,}04178 & -0{,}0940 \\ -0{,}85108 & -0{,}7171 \\ -0{,}38522 & 1{,}4042 \\ 0{,}98917 & -0{,}8767 \\ -0{,}56150 & -0{,}9693 \end{pmatrix} \tag{12}$$

Como se mensionó, "Para obtener nuevamente los datos iniciales, obtenemos, la inversa de la matriz transpuesta de los vectores característicos y multiplicamos ésta a la derecha de la matriz de final, pero además debemos recordar sumar la media de los datos originales, ya que como se mencionó, ésta se resta en al inicio del procedimiento.". Aplicando ésto en Scilab.

*Ad1=inv(vectcar')*mfd*

Lo anterior devuelve una matriz identica a *Ad'*, es decir *Ad1'=Ad*, comprovando de ésta manera que el algoritmo se llevó cabo con exito.

3.2. Casos de aplicación

Se utiliza principalmente en el campo d visión artificial para reconocimiento de patrones, reconocimeintode cara , compresión de imagenes, reducción de ruido en EEG, mejoramiento de muestras de sonido, imagenes y otros tipos de datos, a través de la reducción de dimenciones para disminuir el ruido original.

3.3. Codigo fuente

```
    A=
0.7417 0.8145 1.7603;
2.6537 0.8115 1.5333;
0.0007 0.7597 0.9452;
1.1594 3.1006 2.2203;
2.3354 2.2903 1.4222;
2.2056 1.0797 3.2238;
2.9826 3.2746 0.1535;
2.4069 0.7532 1.6912;
3.0825 1.0973 0.9264;
0.2399 1.2693 1.4559;
1.9685 1.0257 0.9850;
2.3248 1.9881 0.4493;
2.5494 1.6940 2.7318;
0.6967 1.1659 0.7437;
1.9103 2.0832 0.3935 ];

    [f c]=size(A);

    medias=zeros(1,c);

    for i=1:c
.     medias(1,i)=sum(A(:,i))/f;
.   end

    Ad=A;

    for i=1:c
.     Ad(:,i)=Ad(:,i)-medias(i);
.   end

    cov=zeros(c,c);

    for j=1:c
.     for i=1:c
.       cov(j,i)=sum(Ad(:,j).*Ad(:,i))/f;
.     end
.   end
```

```
[eigvec, eigval]=spec(cov);

auxeigvec=zeros(1,c);

for i=1:c
.   auxeigval(1,i)=sum(eigval(:,i));
. end

aux1=zeros(1,c);

for j=1:c
.   for i=1:c
.     if max(auxeigval)== auxeigval(i)
.       aux1(1,j)=i;
.     end
.   end
.   auxeigval(aux1(j))=0;
. end

clear auxeigval;

vectcar=zeros(c,c);

for i=1:c
.   vectcar(:,i)=eigvec(:,aux1(i));
. end

mfd=vectcar'*Ad';
```

4. Análisis de Componente Independiente

4.1. Descripción del Método

4.2. Casos de aplicación

5. "Extended Infomax Algorithm"

5.1. Breve introducción

6. Conclusión

Referencias

[1] Lindsay I. Smith *A Tutorial on Principal Component Analisys*,(2002).

[2] Alguien11 *Algo11*,(year11).

[3] Alguien12 *Algo12*,(year12).

[4] James Pole *Algebra vectorial*,(year21).

[5] K. R. Matthews *Linear Algebra, Second Online Version*,(December 1998).