Separación de imágenes con técnicas de PCA e ICA

Pablo de Castro

January 2, 2015

1 Introducción

El objetivo de esta práctica es la aplicación de las técnicas de Análisis por Componentes Principales (*PCA, del inglés*) y de Análisis por Componentes Independientes (*ICA, del inglés*), para separas tres imágenes que se encuetran mezcladas para extraer las componentes y la matríz de mezcla.

2 Fundamento Teórico

2.1 Problema Propuesto

Se han provisto tres imágenes de 512×512 píxeles cada una conteniendo una mezcla lineal de tres imágenes originales. Como se mostrará en la siguiente sección, dos de las imágenes originales pueden distinguirse claramente (una mujer joven y un gato). Sin embargo, la tercera imágen no es claramente identificable por lo que está oculta.

Las imágenes a utilizar no contienen ningún componente de ruido. El modelo de mezcla utilizado se puede describir como:

$$\mathbf{Y} = \mathbf{AX}$$

en donde $\mathbf{Y}$ son las tres imágenes mezcladas (dimensión 3×512^2), $\mathbf{X}$ las tres imágenes originales (dimensión 3×512^2) y $\mathbf{A}$ es la matrix de mezcla (dimensión 3×3). Se desconoce tanto la matriz de mezcla $\mathbf{A}$ como las componentes originales $\mathbf{X}$. Este problema, en general, carece de solución determinada. Sin embargo, en este trabajo se utilizaran las técnicas de Análisis de Componentes Principales y Análisis por Componentes Independientes para estimar ambos elementos bajo ciertas condiciones impuestas. La aplicación de estos métodos también tiene como objetivo la identificación de la tercera imagen original que se encuentra oculta.

2.2 Análisis de Componentes Principales

El Análisis de Componentes Principales (PCA, del inglés), es un prodedimiento estadístico para transformar un conjunto de observaciones mediante una transformación ortogonal a otro conjunto de componentes que esten descorrelacionadas, denominadas componentes principales.

Esta tranformación se define de tal modo que cada componente tenga la máxima varianza posible bajo la condición de que la transfromación se ortogonal a las componentes anteriores. El procedimiento permite, de forma no paramétrica, la compresión o representación en un sistema de coordenadas que explique mejor la varianza de los datos.

Supóngase que se tiene una matríz de datos $\mathbf{Y}$ (con dimensiones (nº componentes) $\times$ (nº muestras)), el objetivo es encontrar un tranformación lineal ortonormal $\mathbf{P}$ que genere una nueva representación de los datos $\mathbf{Z}$ en los que las componentes esten descorrelacionadas:

$$\mathbf{Z} = \mathbf{PY}$$

La correlación entre las componentes se puede evaluar con el valor de los elementos no diagonales de la matriz de covarianza $\mathbf{C_Y}$, que para la matríz de datos $\mathbf{Y}$ una vez centralizada (restando la media de cada componente) se define (sin tener en cuenta los factores constantes) como:

$$\mathbf{C_Y} = \mathbf{Y}\mathbf{Y^T}$$

El objetivo es que en la representación $\mathbf{Z}$ las componentes esten descorrelacionadas, es decir que los elementos no diagonales de la matríz sean nulos [1]. La matrix de covarianza $\mathbf{C_Z}$ para matríz de datos $\mathbf{Z}$ se define como:

$$\mathbf{C_Z} = \mathbf{Z}\mathbf{Z^T} = \mathbf{P}\mathbf{Y}(\mathbf{P}\mathbf{Y})^T = \mathbf{P}\mathbf{Y}\mathbf{Y^T}\mathbf{P}^T = \mathbf{P}\mathbf{C_Y}\mathbf{P}^T$$

Dado que la matríz $\mathbf{P}$ es ortogonal, su tranversa es igual a su inversa $\mathbf{P^T} = \mathbf{P^{-1}}$, por lo que $\mathbf{P}$ es la matríz que diagonaliza la matríz de covarianza de los datos $\mathbf{Y}$. En resumen, el problema de encontrar los componentes principales se reduce a la búsqueda de los autovectores de $\mathbf{C_Y}$, para lo cuál se pueden utiliza diversos métodos (i.e. EVD y SVD) . La primera componente correspondera con el autovector del autovalor mayor en valor absoluto, la segunda componente al segundo mayor y así sucesivamente.

Este método es útil para la compresión y representación de datos, ya que permite cambiar a una base en la que las componentes estén descorrelacionadas mediante una tranformación lineal. Por lo tanto, cuando se aplique sobre los datos de este problema se espera que cada componente principal corresponda aproximadamente a una imágen inicial.

2.3 Análisis de Componentes Independientes

El Análisis por Componentes Independientes (*ICA, del inglés*) es un método computacional para separar componentes estadísticamente independientes presentes en un conjunto de datos. Este proceso también se conoce como separación de fuentes independientes y es una posible solución para el problema propuesto en esta práctica.

Para la aplicación este procedimiento, se supone que las fuentes (i.e. componentes de $\mathbf{X}$) son estadísticamente independientes, que tienen distribuciones no gaussianas (aunque una sola componente si que puede ser gaussiana) y que a matriz de mezcla $\mathbf{A}$ es invertible. Bajo dichas condiciones el sistema es identificable y sus componentes independientes pueden determinarse salvo permutaciones y factores multiplicativos.

Existen diversos métodos para la obtención de las componentes principales, incluyendo la maximización de la no-gaussianidad (i.e. kurtosis o negentropía), máxima verosimilitud, mínima información nula, métodos tensoriales y de decorrelación no lineal. Muchos de los métodos mencionados requiren la aplicación de PCA como un primer paso para la preparación de los datos, que se denomina blanqueado (*withening*). El objetivo último de todos los métodos es la busqueda de una matríz de componentes $\mathbf{W}$, que es la inversa de la matriz de mezcla $\mathbf{A}$, de modo que las componentes de $\mathbf{X}$ sean estadísticamente independientes.

3 Desarrollo

3.1 Software Utilizado

El análisis de las imágenes en esta práctica se ha realizado en el entorno de computación interactivo *IPython Notebook* [2]. Este mismo documento puede ser visualizado o descargado en formato digital en http://nbviewer.ipython.org/github/pablodecm/SepFuentes/blob/master/SepFuentes.ipynb.

Se han utilizado las bibliotecas de software *open source*: *numpy* [**?**] para la manipulación de arrays y matrices, *matplotlib* [3] para la visualización de imágenes, *scikit-learn* [4] para las implementaciones de PCA e ICA utilizadas y *sympy* [5] para el formato de la salida de datos de matrices.

A continuación, se importan todos los modulos necesarios de la bibliotecas que serán de utilidad en los apartados posteriores.

```
In [1]: %matplotlib inline
        import numpy as np
```

```
import matplotlib.pyplot as plt
from sklearn import decomposition
import sympy
sympy.init_printing(use_latex='mathjax')
```

3.2 Importación y Visualización de las Imágenes

Las imágenes mezcladas se encuentran en los archivos de texto:

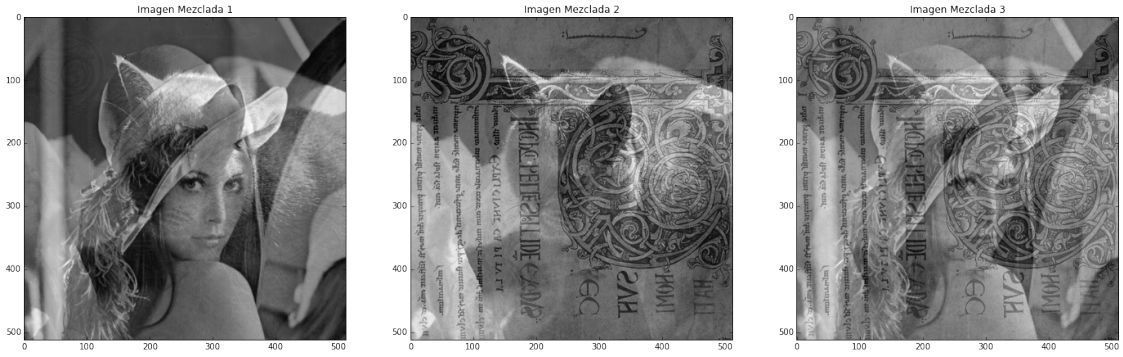
- imagen_mezclada_uno.dat
- imagen_mexclada_dos.dat
- imagen_mexclada_tres.dat

En este apartado, se van a importar las imágenes a un formato de array multidimensional y se van a visualizar:

In [2]:
```
img_1 = np.genfromtxt('imagen_mezclada_uno.dat', dtype='float64')
img_2 = np.genfromtxt('imagen_mexclada_dos.dat', dtype='float64')
img_3 = np.genfromtxt('imagen_mexclada_tres.dat', dtype='float64')

f, (ax1,ax2,ax3) = plt.subplots(1,3)
f.set_size_inches((24,40))
ax1.imshow(img_1, cmap=plt.cm.gray)
ax1.set_title("Imagen Mezclada 1")
ax2.imshow(img_2, cmap=plt.cm.gray)
ax2.set_title("Imagen Mezclada 2")
ax3.imshow(img_3, cmap=plt.cm.gray)
ax3.set_title("Imagen Mezclada 3")
```

Out[2]:
```
<matplotlib.text.Text at 0x1089bea10>
```



Se aprecia que las imagenes provistas contienen tres imágenes mezcladas. La visualización permite distinguir visualmente tres objetos identificables a una mujer jover, un gato y lo que parece una página de un libro con texto e ilustraciones.

Se han importado y visualizado las imágenes, sin embargo, el formato de matriz no es adecuado para la aplicación de PCA e ICA. Por dicho motivo, se van a transformar la matrices 512×512 en vectores de 262144 elementos. Las tres componentes mezcladas a su vez se apilan verticalmente para formar la matríz $\mathbf{Y}$, de dimensiones 3×262144.

In [3]:
```
# each image is flattened and all three are stacked together
Y = np.vstack((img_1.flatten(),img_2.flatten(),img_3.flatten()))
Y.shape
```

Out[3]:

$$(3, \quad 262144)$$

3.3 Aplicación del método PCA

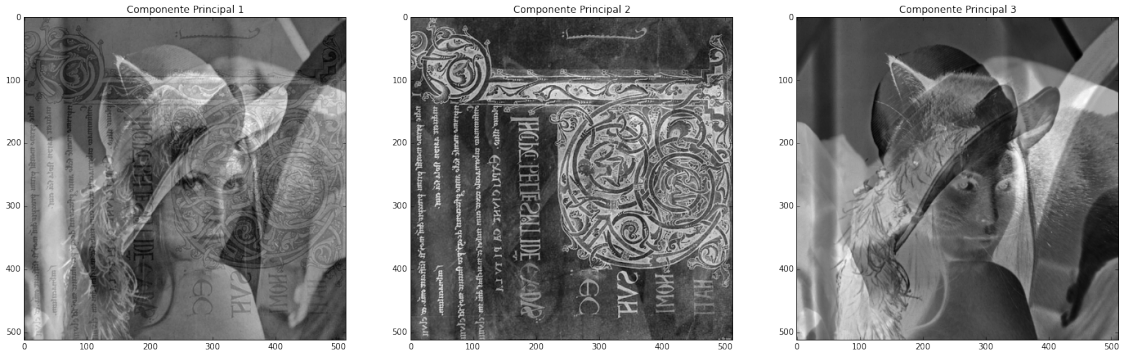
La impleméntacion del método de componenetes principales utilizada es la provista por el módulo *decomposition.PCA* de *scikit-learn*. Se basa en la el método de descomposición en valores singulares (SVD en inglés).

A continuación, se aplica el método a la matríz $\mathbf{Y}$ y se visualizan las componentes principales de los datos de nuevo en forma de imagen:

In [4]:
```
pca = decomposition.PCA(n_components=3)
img_pca= pca.fit_transform(Y.T)

f, (ax1,ax2,ax3) = plt.subplots(1,3)
f.set_size_inches((24,40))
ax1.imshow(img_pca[:,0].reshape((512,512)), cmap=plt.cm.gray)
ax1.set_title("Componente Principal 1")
ax2.imshow(img_pca[:,1].reshape((512,512)), cmap=plt.cm.gray)
ax2.set_title("Componente Principal 2")
ax3.imshow(img_pca[:,2].reshape((512,512)), cmap=plt.cm.gray)
ax3.set_title("Componente Principal 3")
```

Out[4]:
```
<matplotlib.text.Text at 0x108c19f90>
```



Se observa que las componentes obtenidas son más facilmente distinguibles que en el caso anterior. Sin embargo, como se esperaba, el PCA no ha conseguido separar las imágenes en su totalidad. Esto se aprecia claramente para la tercera componente que es una combinación de la imagen de mujer joven y de la imagen del gato. Es posible también obtener la varianza (autovalores)explicada por cada componente en porcentaje:

In [5]:
```
sympy.Matrix(pca.explained_variance_ratio_.round(decimals=4))
```

Out[5]:

$$\begin{bmatrix}0.8389 & 0.1334 & 0.0278\end{bmatrix}$$

La primera componenente explica más del 83% de la varianza total, mientras que los porcentajes de las otras dos son mucho más pequeños. Se puede obtener la matríz $\mathbf{P}$ con los autovectores de la matríz de covarianza.

In [6]:
```
sympy.Eq( sympy.MatrixSymbol("\mathbf{P}",3,3),
          sympy.Matrix(pca.components_.round(decimals=4)))
```

Out[6]:

$$\mathbf{P} = \begin{bmatrix} 0.6591 & 0.1426 & 0.7384 \\ 0.7432 & -0.2742 & -0.6104 \\ 0.1154 & 0.951 & -0.2867 \end{bmatrix}$$

Una primera estimación de la matríz de mezcla $\mathbf{A}_{PCA}$ puede ser obtenida como la inversa de la tranformación $\mathbf{P}$ obteniendo:

```
In [7]: A_pca = np.linalg.inv(pca.components_)
        sympy.Eq(sympy.MatrixSymbol("\mathbf{A}_{PCA}",3,3),
                 sympy.Matrix(A_pca.round(decimals=4)))
```

Out[7]:

$$\mathbf{A}_{PCA} = \begin{bmatrix} 0.6591 & 0.7432 & 0.1154 \\ 0.1426 & -0.2742 & 0.951 \\ 0.7384 & -0.6104 & -0.2867 \end{bmatrix}$$

3.4 Aplicación del método ICA

La aplicación del método de componentes independientes utizada es provista en la biblioteca *scikit-learn* y utiliza el algorímtmo de maximización de la no-gaussianidad *FastICA* [6]. En concreto, se basa en la maximización de la negentropía a partir de un esquema iterativo de punto fijo. Internamente, la implementación aplica PCA en el preprocesamieto para blaquear los datos.

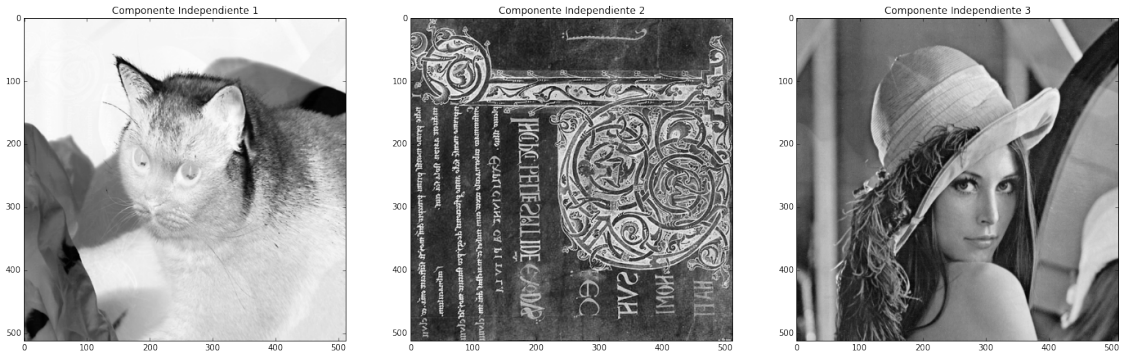
El resultado de la aplicación este método para las imágenes provistas se muestran el siguientes lineas:

```
In [8]: np.random.seed(27)

        ica = decomposition.FastICA(n_components=3)
        img_ica= ica.fit_transform(Y.T)

        f, (ax1,ax2,ax3) = plt.subplots(1,3)
        f.set_size_inches((24,40))
        ax1.imshow(img_ica[:,0].reshape((512,512)), cmap=plt.cm.gray)
        ax1.set_title("Componente Independiente 1")
        ax2.imshow(img_ica[:,1].reshape((512,512)), cmap=plt.cm.gray)
        ax2.set_title("Componente Independiente 2")
        ax3.imshow(img_ica[:,2].reshape((512,512)), cmap=plt.cm.gray)
        ax3.set_title("Componente Independiente 3")
```

Out[8]: <matplotlib.text.Text at 0x109729410>



Se aprecia claramente que el método ICA ha permitido separar las componentes independientes que estaban mezcladas en las imágenes provistas. Las tres imagenes son perfectamente identificables despues de la tranformación. Se aprecia una inversión del color segunda componente, lo que pueden explicarse dado que las componentes por ICA pueden incluir la multiplicación por un un escalar (negativo como en este caso) o comutaciones.La matríz de mezcla $\mathbf{A}_{ICA}$ en este caso tambien se puede estimar como:

```
In [9]: sympy.Eq( sympy.MatrixSymbol("\mathbf{A}_{ICA}",3,3),
                  sympy.Matrix(ica.mixing_.round(decimals=4)))
```

Out[9]:

$$\mathbf{A}_{ICA} = \begin{bmatrix} -566.9308 & 75.3859 & 675.8539 \\ -241.3357 & -183.6017 & -37.1631 \\ -419.1685 & -580.6624 & 630.7346 \end{bmatrix}$$

4 Conclusiones

Se han utilizado los métodos de análisis por Componentes Principales y Componentes Independientes para intentar separar tres imágenes mezcladas linealmente en otras tres imágenes.

Los resultados obtenidos por el método de Componentes Principales han permitido descubrir tranformar los datos a una nueva base en la que las nuevas componentes estan descorrelacionadas. El resultado de este método perimite una mayor distinción entre las imágenes pero se sigue observando una mezcla parcial.

El método de componentes independientes, sim embargo, se adecúa mucho mas al problema propuesto y es capaz de reproducir las componentes originales salvo factores y conmutaciones con exactitud.

References

[1] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[2] Fernando Pérez and Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007.

[3] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] SymPy Development Team. *SymPy: Python library for symbolic mathematics*, 2014.

[6] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):411–430, 2000.