Análisis de componentes principales en imágenes de teledetección.

Mariana Vargas Vieyra
Director: Prof. Oscar Bustos

13 de diciembre de 2013

Agradecimientos.

En primer lugar a mi familia, mi papá, José, que es una gran inspiración para mi, mi mamá, Piqui, que siempre me ha mimado y contenido como sólo las madres saben, mis hermanos, Agustina, Joaquín, y Tomás, que me han hecho compañía y cebado mates durante las largas tardes de estudio, y también mientras escribía este trabajo, mis abuelos, los que están y los que no, que siempre creyeron en mi. Ellos son quienes han hecho todo esto posible. Sin su constante apoyo, su paciencia, y su incondicional contención no estaría hoy culminando mi carrera.

No puedo dejar fuera a mis profesores, que con mucho esmero me han educado y formado en la profesión a la que he elegido dedicarme. Un agradecimiento especial a mi director, Oscar Bustos, que con infinita dedicación me ha guiado a través de este último tramo.

Agradezco además a mi gran amigo Heber, que ha crecido conmigo y que es un hermano mellizo para mi, y a Carlos, a quien guardo un profundo afecto.

Gracias también a mis tíos, a mis profesores de danzas y de violín, dos actividades apasionantes que me han acompañado a lo largo de mi vida y mi carrera.

Por último pero no menos importante, gracias al pueblo argentino por la educación pública.

Índice

1. Introducción y motivaciones. **1**
1.1. Estructura del trabajo. 3

2. Elementos formales. **3**
2.1. Acercamiento intuitivo al PCA. 3
2.2. Repaso de estadística. 6
2.2.1. Varianza, covarianza, correlación, e independencia. 7
2.3. PCA propiamente dicho. 9
2.3.1. Derivación de componentes principales. 9
2.3.2. Propiedades algebraicas de componentes principales en poblaciones. 11
2.3.3. Propiedades algebráicas de componentes principales en muestras. 20
2.4. PCA en acción. 22

3. PCA y clasificación de imágenes. **27**
3.1. El problema de la clasificación. 27
3.2. Algunos algoritmos de clasificación. 29
3.3. PCA y Kmeans: ida y vuelta. 30
3.3.1. K=2. 32
3.3.2. Caso general. 35
3.3.3. ¿Quién es la transformación T? 40

4. Otras técnicas. **42**
4.1. Análisis discriminante. 42
4.1.1. Derivación de LDA 42
4.1.2. LDA y PCA. 43
4.2. Análisis de correlación canónica. 44
4.2.1. Derivación de CCA. 44
4.2.2. CCA y PCA. 46

5. Conclusiones. **46**

Apéndices **47**

A. Código. **47**
A.1. kmeans 47
A.2. PCA 50
A.3. Ejemplo de normal multivariada. 51
A.4. Versión rudimentaria de PCA. 52

Resumen

El análisis de componentes principales es una técnica estadística multivariada de análisis de datos de suma utilidad y amplio uso interdisciplinario. En particular ha sido de gran ayuda en el marco del procesamiento de imágenes. En este trabajo exploraremos los aspectos formales, algunos ejemplos sobre imágenes, y cómo contribuye al uso de otras técnicas multivariadas.

1. Introducción y motivaciones.

En la actualidad el procesamiento de imágenes es un instrumento clave para el desarrollo de diversas áreas. Pero antes de introducir lo que nos motiva a estudiar estas técnicas, debemos esclarecer algunos conceptos. Para empezar, ¿qué es exactamente el procesamiento de imágenes? En principio, el procesamiento de imágenes comprende todos aquellos procedimientos en los que el input es una imagen, y el output también. Según los autores González R. et al (2008) [3] esta primera definición es algo controversial, porque excluye muchas actividades en las que se procesan imágenes y de las que no necesariamente se obtienen otras como output. Es por eso que ellos proponen una clasificación en tres niveles de lo que es el espectro que se ubica entre entre el procesamiento de imágenes y las disciplinas que se encargan de emular la visión humana: en el nivel más bajo se ubican las operaciones más primitivas, tales como remoción de ruido y otras similares. En este caso el input y el output son ambos imágenes. En el nivel intermedio se encuentran tareas tales como la segmentación de imágenes en objetos o regiones, la descripción de estos, y su clasificación. Si bien el input de los procesos de nivel medio son imágenes, a diferencia del nivel más bajo, el output está constituído por atributos extraídos de estas. El nivel más alto involucra el reconocimiento y comprensión de objetos, y la implementación de funciones cognitivas asociadas a la visión.

Si bien no hay restricciones en cuanto al formato de imagen que se toma como input para ser procesada, actualmente se usan imágenes digitales por su eficacia en cuanto a almacenamiento. Las técnicas ya definidas restringidas a esta clase de imágenes se llama "procesamiento de imágenes digitales". En esta instancia será necesario que nos detengamos un poco en lo que es la representación computacional de una imagen.

A continuación, introducimos dos definiciones de suma importancia.

Definición 1. *Una* imagen digital *será una función* f*:* $\mathbb{N} \times \mathbb{N} \to \mathbb{N}^n$ *que para cada par ordenado del dominio asigna un valor o vector de valores, cada uno de los cuales representa la intensidad de gris.*

En realidad es válido decir que una imagen es una función $f\colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}^n$, pero ante la necesidad de representarla computacionalmente se vuelve necesaria su discretización, por eso la definición anterior.

Definición 2. *Un* píxel *será un elemento del dominio de una imágen digital.*

En otras palabras, una imagen digital consiste en un conjunto de píxeles, cada uno de los cuales tiene asociado un vector de cierta dimensión (3 en imágenes de color, o de dimensión mayor en imágenes satelitales multiespectrales), formado por valores enteros que, generalmente, están en el rango {0, ..., 255}.

En este trabajo usaremos la terminología "procesamiento de imágenes" para referirnos al procesamiento de imágenes digitales.

Ya que hemos ganado intuición acerca de estos conceptos, podemos comenzar a hablar de su importancia. El procesamiento de imágenes es importante por muchas razones. Para empezar, es una herramienta ampliamente utilizada en numerosas y diversas disciplinas, con múltiples propósitos. Por ejemplo, en el campo de la agroindustria el uso de la teledetección espacial ha facilitado y profundizado el estudio del suelo. Por otro lado, hoy en día la tecnología nos permite

obtener imágenes mediante sensores que son capaces de recopilar información a lo largo de todo el espectro electromagnético, y almacenarla en múltiples bandas. Tales sensores se llaman sensores hiperespectrales, y las imágenes así adquiridas son de gran utilidad en muchas áreas por la gran cantidad de datos que aportan.

Además, nos permite analizar elementos y eventos microscópicos, como el comportamiento de algunas bacterias, y macroscópicos, como fenómenos meteorológicos o incluso el estudio del cosmos.

Como ya dijimos, el abanico de áreas de trabajo que ofrecen estas técnicas es amplísimo, muchas de ellas no serán tratadas en este trabajo, sólo hay una en particular que es de nuestro interés: la teledetección.

Por teledetección entendemos el conjunto de técnicas utilizadas para el estudio de las características de un objeto o fenómeno mediante mediciones realizadas por un sensor que no está en contacto con éste. Este concepto engloba tanto la obtención de imágenes como el tratamiento de los datos adquiridos y de la información posteriormente elaborada. Como ejemplo de estos sensores tenemos los Landsat, que son una serie de satélites puestos en órbita por Estados Unidos entre los años 1972, cuando se lanzó el primero de todos, y 1999, cuando se lanzó el séptimo y último de la serie hasta ahora. La figura 1 muestra una imagen tomada por el Landsat 5 del desierto de Las Vegas, Estados Unidos.

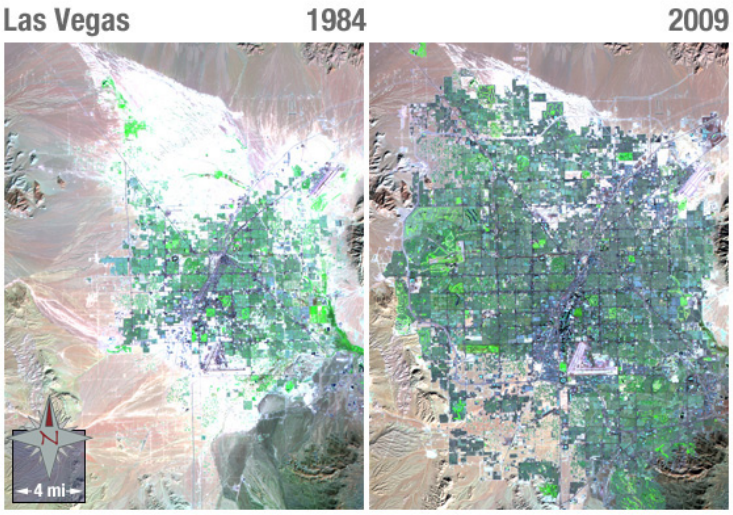


Figura 1: Población de Las Vegas en 1984 y en 1999. El crecimiento de ésta es evidente.

La teledetección es sólo una rama del espectro de posibilidades del procesamiento de imágenes, sin embargo seremos aún más específicos: nosotros trataremos el análisis de componentes principales (PCA por sus siglas en inglés *Principal Component Analysis*) aplicado a imágenes de teledetección, y con esto hemos delimitado nuestro campo de estudio. La importancia del PCA yace en que simplifica la estructura de los datos que se procesan, en particular de las imágenes, facilitando la comprensión de la información. Por ejemplo, consideremos la figura 1. A diferencia de nosotros, los seres humanos, una computadora no comprende que en esa imagen se pueden distinguir principalmente el desierto y una ciudad, y que esta última ha crecido hasta alcanzar lo que parece ser el

doble de su tamaño en quince años, sólo ve una matriz de vectores numéricos. Para obtener información a partir de esos datos crudos, la computadora debe poder hacer la distinción que nosotros hacemos naturalmente, debe poder clasificar. Si bien en la imagen hay mucho más que sólo desierto y ciudad, como rocas, vegetación, bloques de casas, etc., a los fines de medir el crecimiento poblacional nos interesa que se distinga el asentamiento humano de lo que no lo es. A esto podríamos lograrlo con PCA y así facilitar la clasificación.

Los detalles del funcionamiento de esta herramienta serán discutidos a partir de la sección 2.

1.1. Estructura del trabajo.

Lo que resta del trabajo se divide en cuatro secciones.

En la segunda sección estudiaremos los elementos matemáticos formales que hacen al marco teórico en el que se sustenta nuestro objeto de estudio, esto abarca conceptos estadísticos básicos tales como media, varianza, covarianza, matriz de covarianzas, entre otros, y en más detalle el Análisis de Componentes Principales.

En la tercera sección explicaremos el importante papel que juega el PCA en el uso del algoritmo *Kmeans* para clasificar una imagen.

La cuarta sección trata acerca del uso de PCA combinado con otras técnicas de análisis multivariado.

En la quinta y última sección describiremos las conclusiones a las que hemos arribado.

2. Elementos formales.

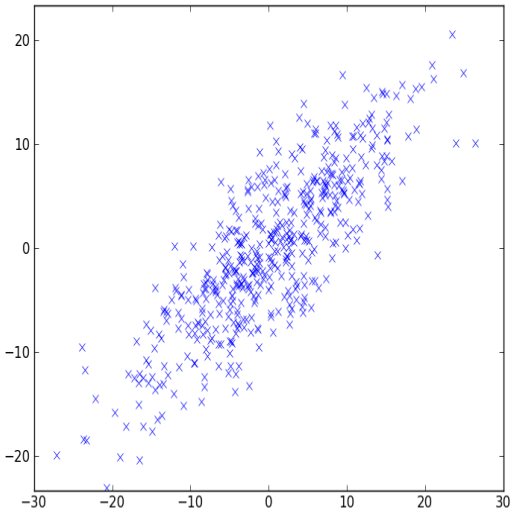
En la introducción hemos mencionado que en este trabajo estudiaremos el análisis de componentes principales (PCA) aplicado a imágenes de teledetección. Para ello nos centraremos a continuación en los aspectos teóricos del PCA.

2.1. Acercamiento intuitivo al PCA.

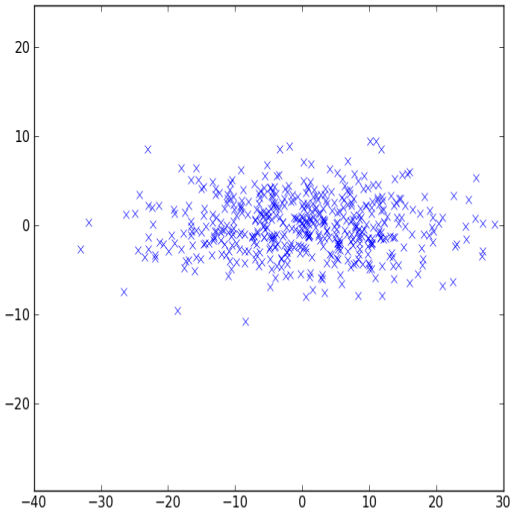
Antes de empezar a describir los tecnicismos que circundan al PCA, le echaremos un vistazo a qué hace exactamente esta herramienta para hacernos una idea de cómo funciona.

Tomemos un vector de variables aleatorias $\vec{x} = (x_1, x_2)$. Hemos escrito un sencillo programa en Python que genera una muestra aleatoria de una distribución normal multivariada de quinientos elementos.

Graficando los datos según los ejes x_1, x_2, vemos algo así:



A simple vista podemos ver que x_1 y x_2 no son las direcciones que mejor explican la variación de las observaciones, pues los datos lucen como una guinda ubicada en diagonal. De hecho querríamos que los datos se representen en función de estos nuevos ejes, lo cual luciría así:



Como podemos ver, este nuevo eje horizontal es en el que más variación ocurre. Este cambio de bases es parte de lo que obtenemos del PCA. Generalmente se trabaja con vectores mucho más grandes, en nuestro ejemplo usamos uno de

dos variables con el fin de poder graficar en dos dimensiones.

Básicamente, una imagen es un conjunto de datos que puede ser pensado como un vector de variables aleatorias, una por cada capa de la imagen (por ejemplo, tres si la imagen está en formato RGB), que adquieren ciertos valores, tantas veces como píxeles hay en la imagen. De este modo, si tuviésemos, por ejemplo, una imagen de tamaño 50×50 en RGB (por sus siglas en inglés *Red, Green, Blue*), la representación correspondiente al input del algoritmo de PCA sería una lista de 2500 vectores de tamaño tres, cada uno de los cuales contendría los valores de cada píxel. Por ejemplo, tomemos la imagen que se ve a continuación:

Figura 2: Imagen de una torta selva negra en RGB.

Esta imagen está en RGB y tiene 389 filas de píxeles, y 500 columnas, o sea que tenemos 194500 vectores de tamaño tres. Para conocerlos, hemos escrito un pequeño programa en Python que imprime algunos de los valores numéricos de la imagen tomados al azar. Estos son los resultados arrojados:

```
[188 138  87]
[190 134  85]
[66 45 24]
[58 53 49]
[32 15 21]
[38 14 10]
[54 44 35]
[35 24 22]
[23 18 14]
```

La primera columna corresponde al color rojo, la segunda al verde, y la tercera al azul.

Ahora viene la parte fina: si las variables involucradas tuviesen una distribución uniforme, veríamos que no están relacionadas de modo alguno, y por lo tanto, todas ellas tendrían "la misma importancia". Sin embargo, nos encontramos con que en el espacio de imágenes naturales los cambios en la vecindad de un determinado píxel son "suaves". Es decir, dado un píxel, podemos predecir de alguna manera los valores que tomarán todos aquellos que lo rodean, bajo determinados criterios. Dando estos datos al algoritmo de PCA obtenemos dos cosas: por un lado un cambio de ejes, es decir, el conjunto de datos será expresado en términos de variables que no están relacionadas entre sí, y por otro lado una reducción dimensional dado que nos quedamos con aquellas variables que explican la mayor parte de la variabiliad de la información.

En este acercamiento intuitivo al análisis de componentes principales hemos incorporado conceptos estadísticos clave, como "variabilidad" y "correlación", entre otros, que son los que detallaremos en esta sección.

2.2. Repaso de estadística.

Comenzaremos entonces recordando algunos conceptos estadísticos básicos.

Definición 3. *Un* espacio muestral*, denotado con la letra griega* Ω*, es el conjunto de todos los resultados posibles de un experimento aleatorio.*

Aquí cometemos un abuso de notación: en realidad el espacio muestral, al que nos referiremos como "muestra", estará compuesto por los valores que pueden tomar los elementos de los vectores píxel, que van de 0 a 255, y además, no habrá aleatoriedad en lo que respecta a imágenes naturales, como ya dijimos antes. Cabe destacar que al extraer la media de una imagen, los píxeles tomarán valores en el rango de $-127{,}5$ a $127{,}5$.

Definición 4. *Una* variable aleatoria *X es una función X:* $\Omega \to \mathbb{R}$*. Un* vector de variables aleatorias *será un vector cuyas componentes son variables aleatorias.*

Definición 5. *Dada una variable aleatoria X:* $\Omega \to \mathcal{A}$*, con* $\mathcal{A} \subseteq \mathbb{R}$*, una* función de probabilidad $\mathbf{f}$*:* $\mathcal{A} \to$ *[0, 1] está definida como*

$$\mathbf{f}(x) = Pr(X = x),$$

esto es, la probabilidad de que la variable aleatoria X adquiera el valor x, el cual pertenece al espacio muestral.

Definición 6. *La* esperanza *de una variable aleatoria x con función de probabilidad f, denotada por la letra E, está definida como sigue:*

$$E(x) = \int f(x)x \, \mathrm{d}x.$$

Para un vector de variables aleatorias $\vec{x} = (x_1, ..., x_n)$ *con funciones de probabilidad* $f_1, ..., f_n$ *para cada* $x_1, ..., x_n$ *respectivamente, la esperanza se calcula componente por componente, de este modo:*

$$E(\vec{x}) = \begin{pmatrix} E(x_1) \\ \vdots \\ E(x_n) \end{pmatrix} = \begin{pmatrix} \int f_1(x_1)x_1 \, \mathrm{d}x_1 \\ \vdots \\ \int f_n(x_n)x_n \, \mathrm{d}x_n \end{pmatrix}$$

Definición 7. *La* media muestral *de una muestra de m elementos se calcula del siguiente modo:*

$$\frac{1}{m}\sum_{i=1}^{m} x_i$$

Ahora estamos en condiciones de definir la varianza y la covarianza, las cuales son clave para explicar matemáticamente el PCA.

2.2.1. Varianza, covarianza, correlación, e independencia.

La importancia que tienen los siguientes conceptos amerita que nos detengamos un poco para comprenderlos ampliamente. Comencemos con la varianza.

Definición 8. *Dada una variable aleatoria X con media μ, la* varianza *de X, denotada var(X), se define como*

$$var(X) = E((X - \mu)^2).$$

La varianza es una medida de dispersión que mide cuánto se separa un valor de la media, y en el caso de las imágenes y a los fines de realizar PCA, cobra una particular importancia. Esto se debe a que es el criterio que se utilizará para seleccionar las componentes que expliquen "la mayor cantidad de variabiliad" posible. Más adelante entenderemos cómo la varianza es el parámetro a tener en cuenta al momento de realizar el cambio de ejes que mencionamos antes.

En general trabajaremos con más de una variable aleatoria, por lo tanto debemos analizar también la covarianza:

Definición 9. *La* covarianza *de dos variables aleatorias X e Y denotada $Cov(X, Y)$ está definida del siguiente modo:*

$$Cov(X, Y) = E[(X - \mu_X)(Y - \mu_Y)],$$

en donde mu_X y mu_Y corresponden a las esperanzas de los vectores X y Y respectivamente. También nos referiremos a la covarianza de X e Y con la letra griega $\sigma_{X,Y}$.

Usando las propiedades de linealidad de la esperanza podemos deducir a partir de esta definición que

$$\begin{aligned}
&\sigma_{X,Y} = E[(X - E(X))(Y - E(Y))] \\
&= E[XY - XE(Y) - E(X)Y + E(X)E(Y)] \\
&= E(XY) - E(X)E(Y) - E(X)E(Y) + E(X)E(Y) \\
&= E(XY) - E(X)E(Y).
\end{aligned}$$

Definición 10. *Dado un vector de variables aleatorias $\vec{X} = (X_1, ..., X_n)$, la* matriz de covarianza *está dada por:*

$$Cov(\vec{X}) = \begin{pmatrix} Cov(X_1, X_1) & Cov(X_1, X_2) & \cdots & Cov(X_1, X_n) \\ Cov(X_2, X_1) & Cov(X_2, X_2) & \cdots & Cov(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_n, X_1) & Cov(X_n, X_2) & \cdots & Cov(X_n, X_n) \end{pmatrix}.$$

Ahora bien, la interpretación de la covarianza es algo crucial: ésta mide cuán parecidos son los comportamientos de dos variables, o cuánto difieren. En otras palabras, es una medida de qué tan fuerte es su correlación. Por lo tanto, si dos variables no están relacionadas, la covarianza es cero. Esto lleva a preguntarnos qué significa exactamente que dos (o más) variables estén relacionadas, y qué significa que sean independientes. Para empezar, la idea de "correlación" tiene una contrapartida formal:

Definición 11. *El* coeficiente de correlación *de dos variables aleatorias X e Y se calcula como*

$$corr(X,Y) = \frac{\sigma_{X,Y}}{\sqrt{var(X)var(Y)}},$$

es decir, la covarianza normalizada. Este coeficiente, denotado generalmente con la letra r, cumple que $-1 \leq r \leq 1$. Dependiendo del valor que adquiera r podemos interpretar tres cosas distintas:

r **cerca de 1** esto implica que las variables involucradas tienen una fuerte relación lineal positiva, esto es, cuando el valor de una aumenta, también lo hace la otra. Análogamente con valores decrecientes.

r **cerca de 0** Si esto ocurre, podemos asumir que no existe relación lineal entre las variables, es decir, no están correlacionadas.

r **cerca de -1** Opuesto al primer caso, esto significa que ambas variables tienen una relación lineal negativa, es decir, cuando una aumenta, la otra decrece.

Cuando r es cero, hemos dicho, las variables son no correlacionadas. Esto quiere decir que los valores que adquiera una no afectan los valores que tome la otra. Dado que estamos planteando lo que significa que dos variables estén relacionadas, no estamos exentos de revisar con cierta profundidad lo que es independencia, si bien no nos detendremos demasiado. Esto haremos a continuación.

Dos eventos son independientes cuando la ocurrencia de uno no afecta la probabilidad de ocurrencia del otro. De lo contrario, la probabilidad de que un evento ocurra se ve condicionada por el hecho de que el otro haya ocurrido o vaya a ocurrir. Para formalizar acerca de esto debemos explorar algunas definiciones:

Definición 12. *Dados dos eventos $\mathcal{A}$ y $\mathcal{B}$ con P($\mathcal{B}$) >0, la* probabilidad condicional *de $\mathcal{A}$ dado $\mathcal{B}$ se calcula como*

$$P(\mathcal{A}|\mathcal{B}) = \frac{P(\mathcal{A} \cap \mathcal{B})}{P(\mathcal{B})},$$

en donde $P(\mathcal{A} \cap \mathcal{B})$ es la probabilidad de que ocurran ambos eventos al mismo tiempo.

Definición 13. *Dos eventos $\mathcal{A}$ y $\mathcal{B}$ son* independientes *sii*

$$P(\mathcal{A} \cap \mathcal{B}) = P(\mathcal{A})P(\mathcal{B}).$$

Notar que si $\mathcal{A}$ y $\mathcal{B}$ son independientes

$$\begin{aligned} P(\mathcal{A}|\mathcal{B}) &= \frac{P(\mathcal{A} \cap \mathcal{B})}{P(\mathcal{B})}, \text{ por definición 12} \\ &= \frac{P(\mathcal{A})P(\mathcal{B})}{P(\mathcal{B})}, \text{ por definición 13} \\ &= P(\mathcal{A}). \end{aligned}$$

Esto significa que $\mathcal{B}$ no afecta la ocurrencia de $\mathcal{A}$. Con cálculos análogos podemos llegar a la conclusión de que $P(\mathcal{B}|\mathcal{A}) = P(\mathcal{B})$.

A partir de aquí el camino se torna intrincado, pues esta definición no es tan intuitiva en lo que respecta a variables aleatorias.

Definición 14. *Dos variables aleatorias discretas X e Y son independientes si para todo x_1, $x_2 \in \mathbb{R}$ con $Pr(Y = x_2) \neq 0$ se tiene:*

$$Pr(X = x_1|Y = x_2) = Pr(X = x_1)Pr(Y = x_2).$$

Esta definición nos dice que dos variables aleatorias discretas X e Y son independientes si los eventos $\{X = x_1\}$ y $\{Y = x_2\}$ lo son para todo $x_1, x_2 \in \mathbb{R}$.

En PCA la independencia o correlación de variables aleatorias es en particular importante, dado que, como ya hemos mencionado, es el punto clave para el cambio de ejes y la reducción dimensional. A partir de lo que hemos detallado en esta subsubsección podemos construir los conocimientos formales que conforman el marco teórico de lo que es nuestro objeto de análisis.

2.3. PCA propiamente dicho.

A partir de aquí y hasta que termine este capítulo exploraremos cómo se derivan las componentes principales a partir de un vector de variables aleatorias, y qué propiedades algebraicas tienen, según se trate de PCA aplicado a una población o a una muestra.

2.3.1. Derivación de componentes principales.

Supongamos que tenemos un vector de n variables aleatorias $\vec{x} = (x_1, ..., x_n)$, y supongamos también que queremos seleccionar el menor número de elementos de ese vector de manera tal de explicar la mayor cantidad de varianza posible. Lo que estamos buscando entonces, es una transformación lineal sobre $\vec{x}$,

$$\alpha_1'\vec{x} = \alpha_{1,1}x_1 + ... + \alpha_{1,n}x_n = \sum_{i=1}^{n} \alpha_{1,i}x_i$$

de modo que en esa dirección se tenga máxima varianza. A continuación, buscaremos una segunda transformación lineal sobre $\vec{x}$, $\alpha_2'\vec{x}$, ortogonal a $\alpha_1'\vec{x}$, que tenga máxima varianza, y así sucesivamente. La k-ésima componente principal estará dada por $\alpha_k'\vec{x}$. La manera de encontrar estas transformaciones lineales son diversas, nosotros usaremos una en particular que describiremos a continuación.

Asumimos conocida la matriz de covarianzas del vector $\vec{x}$, y la llamamos Σ. Si la matriz Σ es $n \times n$, para dos números i, j tales que $1 \leq i, j \leq n$, se tiene que $\Sigma_{i,j}$ contiene la covarianza entre el i-ésimo y el j-ésimo elemento de $\vec{x}$ si $i \neq j$, y en caso contrario, $\Sigma_{i,j}$ será la varianza del i-ésimo elemento de $\vec{x}$. Esto significa que la diagonal de Σ contiene la varianza de $\vec{x}$. Recordemos tres propiedades muy importantes de la matriz de covarianza como operador lineal:

Propiedad 1. *Si φ es una combinación lineal de $\vec{x}$ de manera tal que $\varphi\vec{x} = \varphi_1x_1 + ... + \varphi_nx_n$ entonces $\varphi'\Sigma = Cov(\varphi'\vec{x}, \vec{x})$.*

Propiedad 2. *Consideremos la misma combinación lineal φ de $\vec{x}$ y una segunda combinación lineal del mismo vector, γ. Entonces $\gamma'\Sigma\varphi = Cov(\gamma'\vec{x}, \varphi'\vec{x})$.*

Propiedad 3. *De la propiedad anterior se desprende que $\varphi'\Sigma\varphi = Cov(\varphi'\vec{x}, \varphi'\vec{x})$.*

Lo que queremos, decíamos, es una función $\alpha_1'\vec{x}$ que maximice $Var(\alpha_1'\vec{x}) = \alpha_1'\Sigma\alpha_1$ bajo la condición de que α_1 tenga norma uno, esto es, $\alpha_1'\alpha_1 = 1$. Para

ello usaremos optimización con multiplicadores de Lagrange, de modo que ahora el problema es maximizar

$$\alpha_1'\Sigma\alpha_1 - \lambda(\alpha_1'\alpha_1 - 1)$$

en donde λ es un multipicador de Lagrange. Para encontrar el máximo debemos derivar e igualar a cero. Si derivamos con respecto a α_1' obtenemos

$$\Sigma\alpha_1 - \lambda\alpha_1 = 0 \tag{2.3.1}$$

lo cual implica que

$$\Sigma\alpha_1 = \lambda\alpha_1$$

De esto de desprende que λ es un autovalor de Σ y que α_1 es su correspondiente autovector. Observar que

$$\alpha_1'\Sigma\alpha_1 = \alpha_1'\lambda\alpha_1 = \lambda\alpha_1'\alpha_1 = \lambda = Var(\alpha_1'\vec{x})$$

lo cual implica que α_1 es el autovector que corresponde al autovalor más grande de Σ, digamos, λ_1, i.e., tomamos $\lambda_1 = \lambda$.

Una vez encontrada la primera componente principal, querremos encontrar la segunda, $\alpha_2'\vec{x}$. Esta deberá tener correlación cero con la primera, o sea, $Corr(\alpha_1'\vec{x}, \alpha_2'\vec{x}) = 0$. Con esta condición nos aseguramos que la dirección con la segunda mayor varianza no está relacionada con la primera. La segunda componente principal maximiza $Var(\alpha_2'\vec{x}) = \alpha_2'\Sigma\alpha_2$ sujeto a $\alpha_2'\alpha_2 = 1$ y a $Corr(\alpha_1'\vec{x}, \alpha_2'\vec{x}) = 0$, lo que es equivalente a decir $Cov(\alpha_1'\vec{x}, \alpha_2'\vec{x}) = 0$. Pero

$$Cov(\alpha_1'\vec{x}, \alpha_2'\vec{x}) = \alpha_1'\Sigma\alpha_2 = \alpha_2'\Sigma\alpha_1 = \alpha_2'\lambda_1\alpha_1 = \lambda_1\alpha_2'\alpha_1 = \lambda_1\alpha_1'\alpha_2 \tag{2.3.2}$$

Esto significa que podemos igualar a cero cualquiera de estos términos. Tomamos alguno de ellos al azar y lo igualamos a cero, con lo que obtenemos $\alpha_2'\alpha_1 = 0$ dado que λ_1 se asume distinto de cero. Nuevamente tenemos un problema de maximización similar al planteado anteriormente, en donde la cantidad a ser maximizada es la siguiente:

$$\alpha_2'\Sigma\alpha_2 - \lambda(\alpha_2'\alpha_2 - 1) - \phi\alpha_2'\alpha_1,$$

con λ y ϕ multiplicadores de Lagrange. Derivamos con respecto a α_2' e igualamos a cero para obtener

$$\Sigma\alpha_2 - \lambda\alpha_2 - \phi\alpha_1 = 0. \tag{2.3.3}$$

Para arribar a una conclusión análoga a la de la ecuación (2.3.1) debemos "deshacernos" del útlimo término. Notar que si multiplicamos ambos lados de la ecuación (2.3.3) por α_1' obtenemos

$$\alpha_1'\Sigma\alpha_2 - \alpha_1'\lambda\alpha_2 - \alpha_1'\phi\alpha_1 = 0.$$

De la ecuación (2.3.2) igualada a cero obtenemos que los dos primeros términos son nulos. Además sabemos que $\alpha_1'\alpha_1 = 1$, por lo tanto ϕ debe ser cero. Luego

$$\Sigma\alpha_2 - \lambda\alpha_2 = 0,$$

lo que implica que

$$\Sigma\alpha_2 = \lambda\alpha_2.$$

Por los mismos argumentos expuestos anteriormente, podemos concluir que λ es un autovalor de Σ al cual le corresponde el autovector α_2. Definimos $\lambda_2 = \lambda$. De nuevo se cumple que $\lambda_2 = \alpha_2' \Sigma \alpha_2$ por lo que debe ser el mayor autovalor distinto de λ_1, lo cual lo convierte en el segundo mayor autovalor de Σ.

El procedimiento recién descripto define un algoritmo mediante el cual podemos realizar PCA sobre conjuntos de datos, este puede sintetizarse en los siguientes pasos:

1. Encontrar los autovectores de Σ y sus correspondientes autovalores.
2. Ordenar los autovalores en orden decreciente.
3. Seleccionar las componentes acorde al orden de los autovalores, tantas como sea necesario.

2.3.2. Propiedades algebraicas de componentes principales en poblaciones.

Usando matriz de covarianzas. En esta sección estudiaremos algunas de las conclusiones a las que podemos arribar con PCA a partir de una matriz de covarianza conocida de una población. Si bien el análisis de componentes principales en la práctica se realiza sobre una muestra, debemos también profundizar acerca de las múltiples implicaciones en el marco de una población que pueden inferirse de la experimentación.

Teniendo en cuenta la derivación de componentes principales antes descripta, sea A la matriz cuyas columnas son los autovectores de la matriz de covarianza dada, Σ, ordenados de mayor a menor según sus correspondientes autovalores, esto es,

$$A_{n\times n} = \left[\alpha_1 \mid \cdots \mid \alpha_n\right],$$

en donde n corresponde a la cantidad de variables aleatorias del vector $\vec{x}$, y por lo tanto, a la cantidad total de componentes, entonces

$$\vec{z} = (z_1, ..., z_n) = A'\vec{x} \tag{2.3.4}$$

será el vector tal que z_k contiene la k-ésima componente principal para $k = 1, ..., n$. Como cada α_i es un autovector, por definición se cumple

$$\Sigma\alpha_i = \lambda_i\alpha_i = \alpha_i\lambda_i,$$

con λ_i el autovalor correspondiente, para todo $i = 1, ..., n$. Luego, si definimos

$$\Lambda = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

obtenemos

$$\Sigma A = A\Lambda. \tag{2.3.5}$$

Las dos ecuaciones para nosotros más importantes que podemos deducir de esto son las siguientes:

$$A'\Sigma A = \Lambda, \tag{2.3.6}$$

y

$$\Sigma = A\Lambda A'.^{1} \tag{2.3.7}$$

Los resultados que se exponen a continuación tienen que ver con la optimización de determinados criterios que el PCA logra respecto de la transformación lineal de x, $A'x$.

Propiedad algebraica 1. *Para cualquier entero q tal que $1 \leq q \leq p$, consideremos la transformación lineal ortonormal*

$$y = B'x, \tag{2.3.8}$$

en donde y es una vector de q elementos y B' es una matriz $(q \times p)$, y sea $\Sigma_y = B'\Sigma B$ la matriz de covarianza de y. Entonces la traza de Σ_y, denotada $tr(\Sigma_y)$, está maximizada por $B = A_q$ en donde A_q consiste en las primeras q columnas de la matriz A.

Demostración. Veamos primero qué forma tiene $B_(p \times q)$. Sea β_k la k-ésima columna de B, como las columnas de A forman una base del espacio p-dimensional podemos expresar a β_k como combinación lineal de ellas:

$$\beta_k = \sum_{i=1}^{p} c_{i,k}\alpha_i,\ k = 1, 2, ..., q,$$

en donde $c_{i,k}, i = 1, ..., p,\ k = 1, ..., q$ son constantes. Luego $B = AC$, en donde $C_(p \times q)$ es la matriz cuyo (i,k)-ésimo elemento es $c_{i,k}$. Si exploramos qué aspecto tiene ahora Σ_y vemos que

$$\begin{aligned} B'\Sigma B &= C'A'\Sigma AC \\ &= C'\Lambda C \qquad\qquad \text{, usamos (2.3.6)} \end{aligned}$$

en donde c'_j es la j-ésima fila de C. Calculemos ahora la traza de Σ_y:

$$\begin{aligned} tr(B'\Sigma B) &= tr(C'\Lambda C) \\ &= tr(\Lambda CC') \\ &= \sum_{i=1}^{p}\sum_{j=1}^{q} \Lambda_{i,j} {c^2}_{i,j}, \end{aligned}$$

como $\Lambda_{i,j}$ es cero cuando $i \neq j$ la ecuación continúa de este modo:

$$= \sum_{i=1}^{p} \Lambda_{i,i} \sum_{j=1}^{q} {c^2}_{i,j},$$

pero $\Lambda_{i,j} = \lambda_i$ por definición de Λ,

$$= \sum_{i=1}^{p} \lambda_i \sum_{j=1}^{q} {c^2}_{i,j}. \tag{2.3.9}$$

[1] Este es el resultado del Teorema Espectral.

Ahora bien, como $C = A'B$ tenemos que

$$\begin{aligned} C'C &= B'AA'B \\ &= B'B && \text{, por ser } A \text{ ortogonal} \\ &= I_{(q\times q)} && \text{, por ser } B \text{ ortogonal.} \end{aligned}$$

Esto nos dice que C es ortonormal. Asumamos que C es la matriz formada por las primeras q columnas de otra matriz ortonormal $p \times p$ que llamaremos D. Notar que, si d_j es la j-ésima columna de D se cumple $d_j' d_j = 1$, para todo $j = 1, ..., p$. Por cómo está definida C, cada una de sus filas está formada por los primeros q elementos de la correspondiente fila de D, por lo tanto

$$c_j' c_j \leq 1. \tag{2.3.10}$$

De la ecuación (2.3.11) se desprende que $tr(\Sigma_y)$ será maximizada si los coeficientes de los λ_i, $\sum_{j=1}^{q} {c^2}_{i,j}$, son iguales a 1. ¿Qué $c_{i,j}$ podemos elegir de manera tal que

$$\sum_{j=1}^{q} {c^2}_{i,j} = \begin{cases} 1 & \text{, si } i = 1, ..., q \\ 0 & \text{, si } i = q+1, ..., p \end{cases} \tag{2.3.11}$$

Observar que si tomamos $B = A_q$ entonces $C = A'A_q$, esto es,

$$C = \begin{pmatrix} \alpha_1 \cdot \alpha_1 & \alpha_1 \cdot \alpha_2 & \cdots & \alpha_1 \cdot \alpha_q \\ \alpha_2 \cdot \alpha_1 & \alpha_2 \cdot \alpha_2 & \cdots & \alpha_2 \cdot \alpha_q \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_p \cdot \alpha_1 & \alpha_p \cdot \alpha_2 & \cdots & \alpha_p \cdot \alpha_q \end{pmatrix}.$$

en donde el operador "$\cdot$" indica producto interno. Como los α_i son autovectores ortogonales de a pares se tiene que $\alpha_i \cdot \alpha_j = 0$ si $i \neq j$. Luego

$$C_{i,j} = \begin{cases} 1 & \text{, si } 1 \leq i = j \leq q \\ 0 & \text{, en caso contrario.} \end{cases}$$

Con esto demostramos que $tr(\Sigma_y)$ alcanza su máximo valor tomando $B = A_q$. □

En otras palabras, $z = A'x$ es la combinación lineal de $\vec{x}$ que mejor explica la traza de su matriz de covarianza.

Propiedad algebraica 2. *Consideremos nuevamente la transformación ortonormal*

$$y = B'x, \tag{2.3.12}$$

con x, B, A, y Σ_y definidos como en la propiedad anterior. Entonces $tr(\Sigma_y)$ está acotada inferiormente por $tr(\bar{A}_q' \Sigma \bar{A}_q)$, esto es, tomando $B = \bar{A}_q$, en donde $\bar{A}_q$ es la matriz formada por las últimas q columnas de A.

Demostración. Consideremos el algoritmo para hallar las componentes principales descripto en la sección 2.3.1 y supongamos que queremos encontrarlas en el orden inverso al propuesto, esto es, ordenar las transformaciones lineales no relacionadas de menor varianza a mayor varianza. Con un razonamiento análogo al de la prueba de la propiedad algebráica 1 podemos probar esta propiedad. □

La implicación de la propiedad 2, que bien podría interpretarse también como un corolario de la propiedad 1, es que las últimas componentes no están desestructuradas, y que por lo tanto son de utilidad en otras aplicaciones. La próxima propiedad tiene que ver con el Teorema Espectral.

Propiedad algebraica 3 (Expansión de la descomposición espectral de Σ.)**.**

$$\Sigma = \lambda_1\alpha_1\alpha_1' + \lambda_2\alpha_2\alpha_2' + ... + \lambda_p\alpha_p\alpha_p' \tag{2.3.13}$$

Demostración. Esto se deduce directamente de la ecuación (2.3.7), pues expandiéndola obtenemos

$$\Sigma = \sum_{i=1}^{p} \lambda_i\alpha_i\alpha_i'.$$

□

Apelando a la intuición, esto nos dice que las componentes principales nos proveen de un criterio efectivo para diagonalizar la matriz Σ.

Corolario 1. *Dadas las primeras q componentes principales, z_q, con $q = 1, 2, ..., (p-1)$ se cumple que la siguiente combinación lineal de correlación 0 respecto de las anteriores, de máxima varianza, es la $(q+1)$-ésima componente principal.*

Propiedad algebraica 4. *Consideremos nuevamente la transformación lineal*

$$y = B'x. \tag{2.3.14}$$

Luego, $det(\Sigma_y)$ está maximizado cuando $B = A_q$.

Demostración. Sea k un entero tal que $1 \leq k \leq q$. Comenzaremos definiendo S_k como el subespacio de vectores p-dimensionales ortogonales a $\alpha_1, ..., \alpha_{k-1}$. Notar que

$$dim(S_k) = p - (k-1) = p - k + 1. \tag{2.3.15}$$

Luego, el próximo autovalor de Σ, λ_k, satisface

$$\lambda_k = \sup_{\alpha \in S, \alpha \neq 0} \left\{ \frac{\alpha'\Sigma\alpha}{\alpha'\alpha} \right\} \tag{2.3.16}$$

Definiremos dos subespacios más. Sean $\mu_1 > \mu_2 > ... > \mu_q$ los autovalores de Σ_y y $\delta_1, ..., \delta_q$ sus respectivos autovectores. Definimos, T_k como el subespacio vectorial formado por los vectores de dimensión q que son ortogonales a $\delta_{k+1}, ..., \delta_q$. Luego tenemos que

$$\frac{\delta'\Sigma_y\delta}{\delta'\delta} \geq \mu_k, \tag{2.3.17}$$

para cualquier vector $\delta \neq 0$ de T_k. Definimos también $\hat{S_k}$ como el subespacio conformado por los vectores p-dimensionales que tienen la forma $B\delta$, para algún $\delta \in T_k$. Por cómo definimos $\hat{S_k}$ sabemos que

$$dim(\hat{S_k}) = k. \tag{2.3.18}$$

La idea a partir de aquí es analizar $S_k \cap \hat{S_k}$ para concluir que $\mu_k < \lambda_k$. Notar que

$$dim(S_k + \hat{S_k}) \leq p.^1 \tag{2.3.19}$$

Luego,

$$\begin{aligned} dim(\hat{S}_k \cap S_k) &= dim(S_k) + dim(\hat{S}_k) - dim(S_k + \hat{S}_k) \\ &\leq (p-k+1) + k - p, \text{ por (2.3.15), (2.3.18), y (2.3.19)} \\ &= 1. \end{aligned}$$

Esto significa que la interescción de S_k y $\hat{S}_k$ es no vacía, lo cual implica que hay al menos un vector $\alpha \neq 0$ tal que

$$\alpha = B\delta \tag{2.3.20}$$

para algún $\delta \in T_k$. De esto se desprende que

$$\begin{aligned} \mu_k &\leq \frac{\delta' \Sigma_y \delta}{\delta' \delta} && \text{, por (2.3.17)} \\ &= \frac{\delta' B' \Sigma B \delta}{\delta' B' B \delta} && \text{, por definición de } \Sigma_y \text{ y porque } B'B = I \\ &= \frac{\alpha' \Sigma \alpha}{\alpha' \alpha} && \text{, por (2.3.20)} \\ &\leq \lambda_k && \text{, por (2.3.16).} \end{aligned}$$

Hasta aquí hemos probado que los autovalores de Σ_y son menores o iguales que los de Σ. Veamos $det(\Sigma_y)$:

$$det(\Sigma_y) = \prod_{k=1}^{q} \mu_k \leq \prod_{k=1}^{q} \lambda_k$$

Por lo tanto, si tomamos $B = A_q$, $det(\Sigma_y)$ alcanza su máximo valor.

□

Propiedad algebraica 5. *Supongamos que queremos estimar la j-ésima variable aleatoria del vector $\vec{x}$, x_j, mediante una transformación lineal $y = B'x$. Si llamamos σ_j^2 a la varianza residual que obtenemos al predecir x_j con y entonces*

$$\sum_{j=1}^{p} \sigma_j^2$$

se minimiza cuando $B = A_q$.

Esta propiedad nos dice que z_j es quien mejor estima x_j, más aún, el operador lineal que mejor ajusta a $\vec{x}$ en un espacio q-dimensional está definido por las primeras q componentes principales.

Hasta aquí hemos analizado las propiedades de las componentes principales obtenidas a partir de una matriz de covarianza. Si bien es la opción que elegimos al momento de proponer un algoritmo para derivar las componentes principales de un vector de variables aleatorias (ver 2.3.1), también existe la posibilidad de usar una matriz de correlación. Exploraremos esto en la próxima sección.

[1] Si S_k y $\hat{S}_k$ fuesen independientes entonces la dimensión de la suma sería igual a p.

Usando matriz de correlaciones. No sólo mediante la matriz de covarianza podemos definir las componentes principales, de hecho, si definimos $x*$ del siguiente modo

$$x* = \left(\frac{x_1}{\sigma^2_{1,1}}, ..., \frac{x_n}{\sigma^2_{n,n}} \right), (\text{Con } \sigma_{i,i} \text{ igual a la varianza de } x_i.)$$

es decir, como variables aleatorias estandarizadas, podemos obtener las componentes principales a partir de la transformación lineal

$$z* = A'x* \tag{2.3.21}$$

en donde los vectores columna de A son ahora los autovectores de la matriz de correlación. El hecho de que obtengamos $x*$ a partir de x mediante una operación tan secilla como dividir por $\sigma^2_{i,i}$ cada x_i nos invita a pensar que ambas transformaciones, z y $z*$, están relacionadas de manera directa. Sin embargo las PC son invariantes sólo bajo transformaciones ortogonales, esto es, para transformaciones T tales que, para cada u, $v \in V$ se cumple

$$\langle u, v \rangle = \langle Tu, Tv \rangle,$$

de modo que la transformación realizada sobre x para llegar a $x*$, que no es ortogonal, no preserva PCs. Esto implica que las PCs derivadas de ambas matrices no pueden ser obtenidas la una a partir de la otra.

Entonces sabemos que podemos calcular la matriz de correlación de x para obtener las PCs pero, ¿qué nos motivaría a hacerlo? Veamos un ejemplo. Calculemos las componentes principales de dos variables, y_1 y y_2, de las cuales la primera está medida en centímetros y la segunda en gramos. Supongamos que la matriz de covarianza Σ está dada por:

$$\Sigma = \begin{pmatrix} 10 & 5 \\ 5 & 10 \end{pmatrix}$$

Corramos el algoritmo que presentamos anteriormente implementado en este sencillo código python:

```
# -*- coding: utf-8 -*-
import numpy
import file_to_matrix as fm


def main():
        cm = fm.arrayFromFile ('cov_file.txt')

        f = open('salida_cov.txt','w+')
        f.write("Matriz ingresada:\n" + str(cm) +
        "\n-----------------------------------\n")

        evl, evc = numpy.linalg.eig (cm)
        f.write("Autovalores:\n"+str(evl)+
        "\n-----------------------------------\n")
```

```
        f.write("Autovectores:\n"+str(evc)+
        "\n--------------------------------------\n")

        f.write("Porcentaje de la varianza de la
        primera componente:\n"+str(evl[0]/cm.trace())
        +"\n--------------------------------------
        \n--------------------------------------\n\n")

if __name__ == "__main__":
        main()
```

El módulo `file_to_matrix` toma un archivo y lee el contenido para transformarlo en una matriz. Los resultados arrojados son:

```
Matriz ingresada:
[[ 10.   5.]
 [  5.  10.]]
---------------------------------------
Autovalores:
[ 15.   5.]
---------------------------------------
Autovectores:
[[ 0.70710678 -0.70710678]
 [ 0.70710678  0.70710678]]
---------------------------------------
Porcentaje de la varianza de la primera componente:
0.75
---------------------------------------
---------------------------------------
```

O sea que la primera componente es $0{,}7y_1+0{,}7y_2$, y se lleva el %75 de la variación total. Ahora veamos qué ocurre si hacemos un cambio de unidad de manera tal que y_1 estará medida en milímetros.

```
 Matriz ingresada:
[[ 100.   50.]
 [  50.   10.]]
---------------------------------------
Autovalores:
[ 122.26812024  -12.26812024]
---------------------------------------
Autovectores:
[[ 0.91350006 -0.40683858]
 [ 0.40683858  0.91350006]]
---------------------------------------
Porcentaje de la varianza de la primera componente:
1.11152836578
---------------------------------------
---------------------------------------
```

Como vemos, aún tratándose de datos equivalentes, los resultados son muy diferentes. Esto es consecuencia de lo sensibles que son las PCs a los cambios de

escalas cuando son calculadas mediante la matriz de covarianza. En este sentido la matriz de correlación es más robusta porque sus valores están estandarizados. Aún así, ambas tienen ventajas y desventajas. Algunas ventajas del uso de la matriz de correlación son:

1. Mayor robustez cuando las variables de $\vec{x}$ tienen diferentes unidades o varianzas muy distintas.
2. Mayor facilidad en cuanto a la interpretación de los resultados al haber variables sin unidad o variables inconmensurables.
3. La comparación de diferentes análisis se torna más simple.

No obstante estas ventajas, aún hay motivos por los cuales queramos usar la matriz de covarianza:

1. facilita el estudio de la estadística inferencial en cuanto a PCs, esto es, la tarea de inducir propiedades de las PCs de una población a partir de muestras.
2. Conviene ser usada cuando todas las variables tienen la misma unidad.

Exploremos ahora dos propiedades de este método.

Propiedad algebraica 6. *Las componentes principales dependen sólo de los cocientes de los coeficientes de correlación. En otras palabras, si multiplicamos los elementos de la matriz de correlación, excepto aquellos de la diagonal, por una constante c, entonces las componentes siguen siendo las mismas.*

Demostración. Sea Σ^* la matriz de correlación y sea $\hat{\Sigma}^*$ la matriz obtenida de Σ^* multiplicando sus elementos por una constante c, excepto los de la diagonal. Entonces $\hat{\Sigma}^*$ cumple la siguiente ecuación:

$$\hat{\Sigma}^* = c(\Sigma^* - I) + I,$$

en donde I es la matriz identidad del tamaño correspondiente. Luego,

$$\Sigma^* = \frac{1}{c}(\hat{\Sigma}^* - I) + I \tag{2.3.22}$$

Recordar que la matriz de correlación tiene unos en su diagonal. Basta con probar que los autovectores de $\hat{\Sigma}^*$ y Σ^* son los mismos, por lo que queremos descubrir la relación que hay entre ellos. Sea α un autovector de Σ^* y λ^* su autovalor correspondiente, veamos que α es también un autovector de $\hat{\Sigma}^*$:

$$\begin{aligned}
\Sigma^*\alpha &= \lambda^*\alpha \\
&\Downarrow \\
\Sigma^*\alpha - \lambda^*\alpha &= \vec{0} \\
&\Downarrow \\
(\Sigma^* - \lambda^* I)\alpha &= \vec{0} \\
&\Downarrow \\
((\frac{1}{c}(\hat{\Sigma}^* - I) + I) - \lambda^* I)\alpha &= \vec{0}
\end{aligned}$$

Por ecuación (2.3.22)

$$\Downarrow$$
$$(\frac{1}{c}\hat{\Sigma}^* - \frac{1}{c}I + I - \lambda^* I)\alpha = \vec{0}$$
$$\Downarrow$$
$$(\frac{1}{c}\hat{\Sigma}^* - I(\frac{1}{c} - 1 + \lambda^*))\alpha = \vec{0}$$
$$\Downarrow$$
$$(\hat{\Sigma}^* - I(1 - c + c\lambda^*))\alpha = \vec{0}$$

De esto último se desprende que α también es autovector de $\hat{\Sigma}^*$. Más aún, su autovalor corresponderá a la constante que multiplica a I, de manera tal que $1 - c + c\lambda^* = \hat{\lambda}^*$. □

Es decir, las componentes principales no se ven afectadas por la escala. Puede que un cambio de escala modifique los autovalores, pero sus proporciones quedan intactas y también las direcciones.

Propiedad algebraica 7. *Si tomamos la restricción*

$$\tilde{\alpha}_k'\tilde{\alpha}_k = \lambda_k$$

para cada $k = 1, ..., n$ en lugar de $\alpha_k'\alpha_k = 1$ durante la derivación, entonces el j-ésimo elemento del vector $\tilde{\alpha}_k$, $\tilde{\alpha}_{k,j}$, es la correlación entre la j-ésima variable estandarizada, x_j* y la k-ésima PC.

Demostración. Queremos probar que

$$Corr(x_j*, \alpha_k'x*) = \tilde{\alpha}_{k,j}.$$

Comencemos desarrollando el lado derecho de la ecuación:

$$\begin{aligned} Corr(x_j*, \alpha_k'x*) &= Corr(x_j*, z_k*) \\ &= \frac{Cov(x_j*, z_k*)}{\sigma_{x_j*}\sigma_{z_k*}} \end{aligned}$$

Sabemos que $Cov(x_j*, z_k*) = j$-ésimo valor de $\Sigma\alpha_k$, pero $\Sigma\alpha_k = \lambda_k\alpha_k$, entonces la ecuación continúa

$$= \frac{\lambda_k\alpha_{j,k}}{\sqrt{Var(x_j*)Var(z_k*)}}$$

Como x_j* está estandarizada tiene varianza 1. Además, $Var(z_k*) = \lambda_k$, luego la ecuación continúa

$$\begin{aligned} &= \frac{\lambda_k\alpha_{j,k}}{\sqrt{\lambda_k}} \\ &= \sqrt{\lambda_k}\alpha_{j,k} \\ &= \tilde{\alpha}_{k,j} \end{aligned}$$

□

2.3.3. Propiedades algebráicas de componentes principales en muestras.

En las últimas dos secciones hemos estudiado importantes propiedades algebráicas de componentes principales en poblaciones, no obstante es necesario detenernos en el marco muestral de este análisis porque es de nuestro interés el uso del PCA sobre imágenes, las cuales constituyen muestras del espacio de imágenes naturales. Otra motivo de ello es que lo que ya hemos explorado hasta aquí son resultados que pueden inferirse a partir de la experimentación. Por estas dos razones desarrollaremos aquí lo que necesitamos saber acerca de componentes principales en muestras.

Básicamente el comportamiento es el mismo, aún tenemos un vector $\vec{x}$ de variables aleatorias pero su distribución nos es desonocida. En su lugar contamos con m observaciones de $\vec{x}$, las cuales llamaremos $x_1, x_2, .., x_m$. Al igual que en la sección (2.3.1) buscamos una primera combinación lineal $z_{i,1} = a_1' x_i, i = 1, 2, ..., m$ que maximice cierto criterio, que esta vez será la varianza muestral dada por

$$s^2{}_1 = \frac{1}{m-1}\sum_{i=1}^{m}(z_{i,1} - \bar{z}_1)^2$$

sujeto a que $a_1' a_1 = 1$. Una vez encontrada, querremos otra combinación $z_{i,2} = a_2' x_i, i = 1, 2, ..., m$ de manera tal que maximice

$$s^2{}_2 = \frac{1}{m-1}\sum_{i=1}^{m}(z_{i,2} - \bar{z}_2)^2$$

sujeto a $a_2' a_2 = 1$ y a que $z_{i,1}$ y $z_{i,2}$ no estén correlacionadas. De este modo, nuestra k-ésima componente principal muestral estará dada por $a_k' x$ y $z_{i,k}$ será la proyección de x_i sobre la k-ésima componente. Para ver esto en términos de matrices definamos X como la matriz $(m \times n)$ cuyo lugar (i, j) está ocupado por el j-ésimo valor de la i-ésima observación de $\vec{x}$, y sea

$$A_{n\times n} = (a_1|...|a_n). \tag{2.3.23}$$

Entonces podemos expresar los valores de $z_{i,k}$ en una matriz Z de modo que

$$Z = XA$$

es una matriz $(m \times n)$ tal que $Z_{(i,j)} = z_{i,j}$. Definiremos también una matriz de covarianza muestral que llamaremos S que es tal que

$$S_{(i,j)} = \frac{1}{m-1}\sum_{k=1}^{m}(x_{k,i} - \bar{x}_i)(x_{k,j} - \bar{x}_j).$$

A los fines de representar S en términos de las observaciones $x_1, ..., x_m$ redefiniremos X de manera tal que contenga los valores de la muestra con su media extraída. Entonces podemos escribir

$$S = \frac{1}{m-1}X'X,$$

en donde ahora $X_{(i,j)} = x_{i,j} - \bar{x}_j$.

De manera análoga al caso poblacional, tenemos que

$$a_k' S a_k = l_k,$$

en donde a_k es un autovector de S y l_k su correspondiente autovalor, que además es el k-ésimo más grande.

En este trabajo no implementaremos el algoritmo de PCA muestral como indica la derivación recién expuesta, sino que usaremos lo que en inglés se llama *singular value decomposition*, o SVD por sus siglas. Este es un resultado del álgebra matricial que describe un método para descomponer una matriz cualquiera $X_{(m\times n)}$ (m observaciones, n variables aleatorias) en el producto de otras tres tal que X puede expresarse como

$$X = ULT' \tag{2.3.24}$$

de modo que

1. U es una matriz $(m \times r)$ cuyas columnas son ortonormales,
2. T es una matriz $(n \times r)$ cuyas columnas son también ortonormales,
3. L es una matriz $(r \times r)$ diagonal que contiene las raíces cuadradas de los primeros r autovalores de X.

¿Pero quiénes son U, T, y L? Recordemos que el teorema espectral nos provee condiciones bajo las cuales una matriz puede ser diagonalizada y representada en términos de sus autovectores y autovalores, en el caso de S esto sería [1]

$$(m-1)S = XX' = A\Lambda A' \tag{2.3.25}$$

en donde A está definida como en (2.3.23) y Λ contiene en su diagonal los autovalores l_k de S. Si desarrollamos la ecuación (2.3.24) obtenemos

$$S = l_1 a_1' a_1 + ... + l_n a_n' a_n.$$

Definamos $T_{(n\times r)}$ del siguiente modo:

$$T = (a_1|...|a_r),$$

y definamos también U como la matriz $(m \times r)$ formada por las columnas

$$u_k = \sqrt{\frac{1}{l_k}} X a_k,$$

para $k = 1, 2, ..., r$, y L como la matriz diagonal $(r \times r)$ que es tal que

$$L_{(i,i)} = \sqrt{l_k}.$$

[1] Notar que los autovectores de S y los de $(m-1)S$ son los mismos, y que sus correspondientes autovalores varían en la misma proporción conservando el orden descendente. Por lo tanto, y a los fines de realizar PCA, nos es indiferente considerar $S = XX'$ o $S = \frac{1}{m-1} XX'$.

U, T y L están bien definidas porque cumplen con las tres condiciones arriba impuestas y porque $X = ULT'$. Veamos esto último.

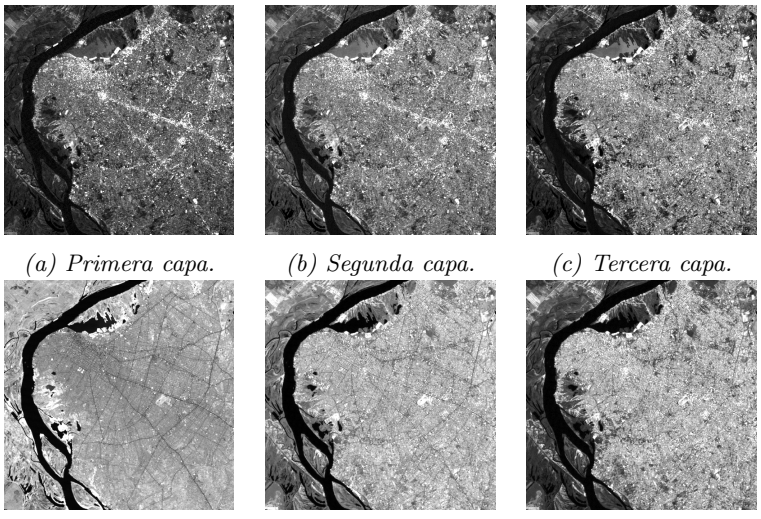
$$
\begin{aligned}
& ULT' \\
& \quad = U \begin{pmatrix} \sqrt{l_1} & 0 & \cdots & 0 \\ 0 & \sqrt{l_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{l_r} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_r \end{pmatrix} \\
& \quad = U \begin{pmatrix} \sqrt{l_1} a_1' \\ \sqrt{l_2} a_2' \\ \vdots \\ \sqrt{l_r} a_r' \end{pmatrix} \\
& \quad = \sum_{k=1}^{r} l_k^{-1/2} X a_k l_k^{1/2} a_k' \\
& \quad = \sum_{k=1}^{r} X a_k a_k' \\
& \quad = \sum_{k=1}^{n} X a_k a_k'
\end{aligned}
$$

Esta factorización de matrices nos provee un algoritmo eficiente para computar las componentes principales, pues una vez hecha la descomposición sobre la matriz de covarianza S encontramos sus autovectores en T, y las raíces cuadradas de sus correspondientes autovalores en L (que serían las desviaciones estándar de las componentes principales). Además podemos "recuperar" las proyecciones de las observaciones sobre las componentes mediante la operación $Z = UL$[1].

2.4. PCA en acción.

Cada una de las siguientes corresponde a una capa de una imagen de la ciudad de Asunción tomada por el Landsat-5:

[1]Observar que en U cada proyección $z_{i,k}, i = 1, ..., n$ está multiplicada por $\sqrt{\frac{1}{l_k}}$.



(a) Primera capa. *(b) Segunda capa.* *(c) Tercera capa.*

(d) Cuarta capa. *(e) Quinta capa.* *(f) Sexta capa.*

Figura 3: Capas de la imagen.

En estas figuras podemos ver la similitud de los valores que hay en cada capa, de hecho es muy probable que mucha de la información contenida en los datos sea redundante.

Hemos implementado una versión rudimentaria del algoritmo de componentes principales a los fines de mostrar cómo funciona "por dentro". Esta implementación toma como entrada la imagen cruda, que en este caso es un arreglo de tamaño $512 \times 512 \times 6$ y realiza una transformación de las dimensiones de manera tal de obtener un arreglo de tamaño $512^2 \times 6$, de este modo cada capa de la imagen corresponderá a una variable aleatoria (habrá seis de ellas) que constituyen una muestra de tamaño 512^2 (cada píxel es un elemento de la muestra). Estos son los primeros tres elementos y los últimos tres de la muestra:

```
[[ 49  56  37 183 118  56]
 [ 31  37  12 208 101  63]
 [ 56  85  56 211 141  88]
 ...,
 [ 31  56  62 164 161 120]
 [ 43  56  56 148 150 117]
 [ 62  75  74 161 150 120]].
```

En un segundo paso centramos los datos calculando las medias de cada columna:

```
  medias:
[  86.  111.  108.  151.  150.  132.]
```

y luego restándolas a la muestra. Llamaremos X_0 a la matriz de datos centrados. La porción de X_0 que se corresponde con los datos de más arriba serían

```
[[-37. -55. -71.  32. -32. -76.]
 [-55. -74. -96.  57. -49. -69.]
 [-30. -26. -52.  60.  -9. -44.]
 ...,
```

```
 [-55. -55. -46.  13.  11. -12.]
 [-43. -55. -52.  -3.   0. -15.]
 [-24. -36. -34.  10.   0. -12.]]
```

El tercer paso consiste en calcular la matriz de covarianzas, S, mediante la operación $X_0'X_0$:

```
matriz de covariaza S:

[[ 2840.63452009  2780.97938148  3045.85633414   724.40577853
   2193.93738151  2819.89209325]
 [ 2780.97938148  3021.24133774  3326.42347879  1158.38280252
   2619.91740767  3161.98228066]
 [ 3045.85633414  3326.42347879  4001.55654357   993.35799163
   3070.48956486  3868.85701697]
 [  724.40577853  1158.38280252   993.35799163  2897.558264
2206.47528257
   1529.05262013]
 [ 2193.93738151  2619.91740767  3070.48956486  2206.47528257
   3677.22292031  3783.95625288]
 [ 2819.89209325  3161.98228066  3868.85701697  1529.05262013
   3783.95625288  4488.36516329]]
```

En el inciso anterior planteamos cómo descomponer la matriz de datos centrados en sus valores singulares nos proporciona un algoritmo de PCA, sin embargo, su costo computacional es tan elevado que implementarlo en una computadora estándar conlleva a un error de memoria. Por esta razón hemos optado por una alternativa más eficiente, que es la aplicar SVD a la matriz de covarianzas S. Dado que esta es simétrica y definida positiva, su descomposición en valores singulares es exactamente su descomposición espectral.[1] Entonces

$$S = ULT' = P\Lambda P', \tag{2.4.1}$$

en donde U, L, y T son las matrices obtenidas a partir del SVD, P es la matriz tal que su i-ésima columna es el autovector de S correspondiente al i-ésimo autovalor más grande, y Λ es una matriz diagonal que contiene los autovalores de S.

Hemos usado la función de Python `numpy.linalg.svd` para realizar SVD sobre S. Su output consiste en

1. una matriz U que será P de la ecuación 2.4.1,
2. una matriz V que será P',
3. un arreglo plano L que contiene la diagonal de Λ:

```
U:
[[-0.36583486 -0.3132296   0.48374292  0.64444621 -0.12237688
0.32211464]
```

[1] La descomposición espectral es un caso particular de SVD.

```
 [-0.40505942 -0.17875674  0.39160383 -0.19243242  0.31321242 -0.71797429]
 [-0.46798549 -0.28528272  0.01351371 -0.67371575 -0.07668844
0.48953723]
 [-0.21181881  0.81733789  0.4229417  -0.12155968 -0.30195103
0.04754676]
 [-0.43471976  0.34706865 -0.35780094  0.21856052  0.68755062
0.20505124]
 [-0.49923224 -0.00700989 -0.55277067  0.17669317 -0.56315227 -0.31112967]]
_____________

V:
[[-0.36583486 -0.40505942 -0.46798549 -0.21181881 -0.43471976 -0.49923224]
 [-0.3132296  -0.17875674 -0.28528272  0.81733789  0.34706865 -0.00700989]
 [ 0.48374292  0.39160383  0.01351371  0.4229417  -0.35780094 -0.55277067]
 [ 0.64444621 -0.19243242 -0.67371575 -0.12155968  0.21856052
0.17669317]
 [-0.12237688  0.31321242 -0.07668844 -0.30195103  0.68755062 -0.56315227]
 [ 0.32211464 -0.71797429  0.48953723  0.04754676  0.20505124 -0.31112967]]
_____________

L:
[ 16690.73653213   2943.70561854    965.33662691    206.61050709
     69.40331436     50.78614998]
_____________
```

Para elegir las componentes principales le echamos un vistazo al valor de

$$t = \frac{\sum_{i=0}^{r} L[i]}{Tr(S)}$$

en donde r es un número que varía entre 0 y 5[1] y que será el índice del arreglo L. Entonces t es una función de r que indica la variabilidad acumulada, esto es, la cantidad de varianza explicada por las primeras r componentes. En el presente ejemplo, con sólo tres componentes basta para explicar más del %95 de la varianza:

```
 Variabilidad acumulada:
0.984383497401


********************
 Cantidad de componentes:  3
********************
```

Definimos `u_tilde` como la matriz formada por las primeras tres columnas de `u`:

```
u_tilde:

[[-0.36583486 -0.3132296   0.48374292]
 [-0.40505942 -0.17875674  0.39160383]
 [-0.46798549 -0.28528272  0.01351371]
```

[1] El número r es como mínimo 1 y a lo sumo la cantidad de variables aleatorias involucradas. En la implementación computacional del algoritmo los índices de los arreglos comienzan en 0, por eso es que $0 \leq r \leq 5$ y no $1 \leq r \leq 6$.

```
 [-0.21181881   0.81733789   0.4229417 ]
 [-0.43471976   0.34706865  -0.35780094]
 [-0.49923224  -0.00700989  -0.55277067]]
```

Si tomamos de P tantas columnas como indica el número de componentes seleccionadas para formar $\tilde{P}$, y procedemos del mismo modo con Λ para formar $\tilde{\Lambda}$ entonces podemos recuperar las covarianzas en términos de las componentes principales del siguiente modo:

$$\tilde{S} = \tilde{P}\tilde{\Lambda}\tilde{P}'$$

, en nuestra implementación obtenemos este resultado mediante la operación de Python `numpy.dot(u_tilde,numpy.dot(numpy.diag(s[0:i]), v[0:i]))` lo que devuelve

```
 S_tilde:

[[ 2748.51806875  2821.00717597  3126.90149048   737.2489767
2167.32137938
   2796.67221476]
 [ 2821.00717597  2980.60229186  3319.15463113  1161.84727481
   2621.13791157  3169.90437152]
 [ 3126.90149048  3319.15463113  3895.1985957    973.64807157
   3099.47399576  3898.19000842]
 [  737.2489767   1161.84727481   973.64807157  2888.06261202
   2225.87798726  1522.44000321]
 [ 2167.32137938  2621.13791157  3099.47399576  2225.87798726
   3632.40930867  3806.09003262]
 [ 2796.67221476  3169.90437152  3898.19000842  1522.44000321
   3806.09003262  4454.98790059]]
```

Resta proyectar nuestros datos en las componentes principales seleccionadas y sumarles la media. Lo primero se obtiene así:

$$\tilde{X} = (X_0\tilde{P})\tilde{P}'.$$

Realizadas estas operaciones nuestro programa arroja una matriz de datos finales. He aquí una parte de ella:

```
 Datos finales:
[[  39.18444396   64.95716716   38.62040008  184.87641
110.74714534
    59.92575664]
 [  18.80181846   48.07288049   18.32275294  203.15732856
112.02273459
    49.48396042]
 [  54.20019796   84.449155     59.32946801  211.52607777
140.56557513
    86.79982204]
 ...,
 [  26.41648168   60.0646164    62.65031724  165.34378273
156.38537512
   122.8994337 ]
```

```
[  31.58625467   61.95349442   64.32528938  150.09064083
144.67470212
   116.47937558]
 [  52.8170924    81.82584007   78.49801519  161.46909014
147.79428981
   118.69610743]]
```

A partir de ellos podemos reconstruir una imagen comprimida. Es importante observar que la reducción dimensional lograda ha sido muy importante y pudo llevarse a cabo a un bajo costo computacional.

3. PCA y clasificación de imágenes.

La cuestión de clasificación en sí está fuera del alcance de este trabajo, sin embargo investigaremos cómo puede el PCA ayudar en esta tarea. Recordaremos brevemente algunas ideas.

3.1. El problema de la clasificación.

Los algoritmos de clasificación contribuyen a facilitar la interpretación de los datos provistos por una imagen. En uno de sus libros, M. Anji Reedy (2008) [7] define la clasificación como el procedimiento para categorizar de manera automática los píxeles de una imagen en clases. Por ejemplo, tomemos la siguiente imagen satelital:

Figura 4: Zona de Hong Kong.

Supongamos que nos interesa medir la superficie edificada. Notar que en la imagen pueden distinguirse diversos elementos como playones, rutas, agua, botes, casas, espacios verdes, montañas, etc. El ojo humano puede realizar esta distinción facilmente, pero un software no. Para facilitar esta tarea de interpretación hemos implementado en python el algoritmo de clasificación denominado *kmeans*, que se agregó en el apéndice (A.1). El resultado puede verse en esta imagen:

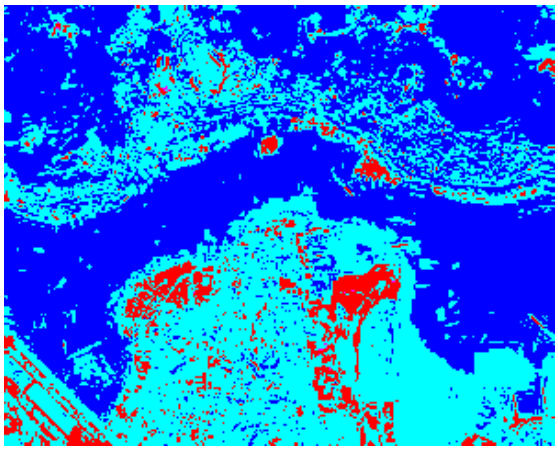


Figura 5: Figura anterior clasificada mediante kmeans.

De este modo vemos que se ha pintado con color celeste la región edificada y con azul el resto. Además, el algoritmo ha puesto en una tercera categoría lo que podría interpretarse como playones o espacios grandes pavimentados.

No está de más mencionar que la clasificación puede llevarse a cabo píxel por píxel o acorde al contexto. La diferencia es muy importante: la primera forma tiene que ver con categorizar el píxel según el valor que tome dentro del espectro, la segunda es más compleja y califica como procedimiento de "tercer nivel", teniendo en cuenta los niveles de complejidad de procesamiento que mencionamos en la introducción. Clasificar por contexto implica atribuírle a un píxel un significado que está determinado por algún patrón que obedecen sus vecinos, de modo que es factible que dos píxeles que se parecen mucho en cuanto a valores en el espectro pertenezcan a categorías muy distintas. Un ejemplo de esto último puede verse en la siguiente imagen:

Figura 6: Notar que los leones tienen un "color" muy parecido al de los pastizales circundantes.

Si asumimos un criterio en el que nos interesa separar el pastizal de los animales, corremos el riesgo de no obtener resultados satisfactorios con una clasificación píxel por píxel dado que los colores del pelaje de los leones es muy

parecido al del suelo. La siguiente figura resulta de aplicar el algoritmo *kmeans* a la imagen anterior con seis categorías. Notar que incluso con un alto número de clases el algoritmo apenas distingue animales de pastizal.

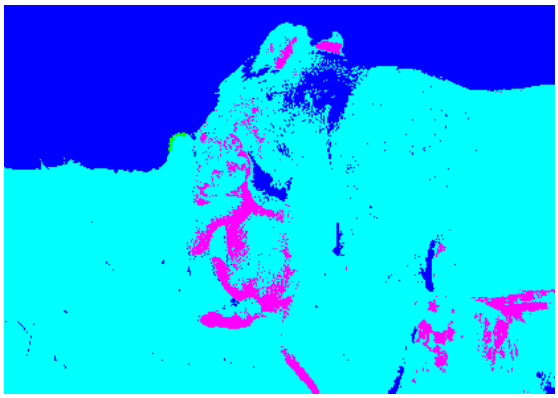


Figura 7: Figura anterior clasificada mediante kmeans.

Para profundizar acerca de este tema se pueden consultar los libros de M. Anji Reedy ya citado, el de O. H. Bustos et. al. (2004) [1], y el de J. R. Jensen (2005) [5].

3.2. Algunos algoritmos de clasificación.

Los algoritmos de clasificación pueden pertenecer a dos grupos, los supervisados y los no supervisados. Los supervisados son aquellos que requieren un grupo de entrenamiento, esto es, un conjunto de píxeles de los que se conoce su clase de pertenencia. Teniendo este grupo de "píxeles clasificados" la categorización de uno de clase desconocida se reduce a medir las distancias entre él y cada uno de los del grupo de entrenamiento e incluirlo en la clase de aquel que más cerca está. Notar que además necesitamos tener definidas las categorías previamente. La distancia y lo que "cerca" significa varían según el algoritmo. Enumeraremos algunos de ellos a continuación. Para ilustrar las ideas, asumamos que para cada categoría C tenemos un conjunto de entrenamiento S_C formado por píxeles[1]. Sean IM una imagen de n bandas a clasificar y ϑ el conjunto de clases.

1. Paralelepípedo. Para cada clase C podemos delimitar un paralelepípedo del siguiente modo:

$$P_C = [min\{IM(i,j,0)/(i,j) \in S_C\}, max\{IM(i,j,0)/(i,j) \in S_C\}] \times \\ ... \times [min\{IM(i,j,n)/(i,j) \in S_C\}, max\{IM(i,j,n)/(i,j) \in S_C\}].$$

   Entonces un píxel (i_0, j_0) será clasificado como perteneciente a la clase C si $(i_0, j_0) \in P_C$.

2. Mínima distancia. Se calcula para cada clase C un centroide m_C, y se mide la distancia de estos con el píxel a clasificar. Este se incluirá en la categoría cuyo centroide se encuentra más cerca. Una vez clasificado, por ejemplo, como perteneciente a la clase A, se recomputa el centroide m_A.

[1] Recordemos que los píxeles son pares (i, j)

3. Máxima verosimilitud. Este algoritmo asume que las clases tienen una distribución, en general gaussiana. Es decir que a cada clase C le corresponde una función de densidad $f_C : \mathbb{R}^n \to \mathbb{R}^{>0}$, y un píxel (i_0, j_0) se categoriza como perteneciente a la clase C si y sólo si ocurre que

$$f_C(IM(i,j,0),...,IM(i,j,n)) > f_L(IM(i,j,0),...,IM(i,j,n))$$

para todo $L \in \vartheta$.

Los algoritmos no supervisados, por el contrario, no se basan en la comparación de los píxeles con aquellos pertenecientes al conjunto de entrenamiento, sino que se analizan y categorizan de acuerdo a propiedades inherentes a los datos mismos. Algunos ejemplos son los siguientes:

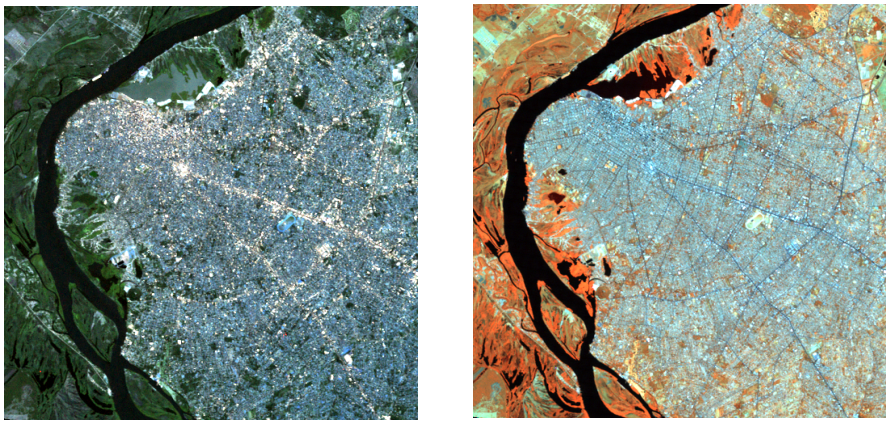
1. K-means. Se asume que clasificaremos los píxeles en k clases. La idea es minimizar la suma de los cuadrados de las distancias entre el píxel a clasificar y la media de cada clase. Al iniciarse el algoritmo se toma un arreglo de medias arbitrarias, $\mu_{C_1}, ..., \mu_{C_k}$. De manera análoga al algoritmo de distancias mínimas, el píxel (i_0, j_0) se categoriza como perteneciente a C si la distancia a su media, μ_C, es la mínima. También se recomputa μ_C.

2. Isodata. Este es similar a K-means, salvo que no se necesita especificar la cantidad de clases. Esto ocurre porque dos clases se unen si se parecen mucho y una clase de divide si tiene mucha variabilidad. No entraremos en los detalles formales porque, como dijimos antes, escapan a los propósitos de este trabajo.

3.3. PCA y Kmeans: ida y vuelta.

Ahora nos centraremos en la pregunta principal de esta sección: ¿qué puede hacer el PCA por la clasificación de píxeles y viceversa? Veámoslo por nosotros mismos.

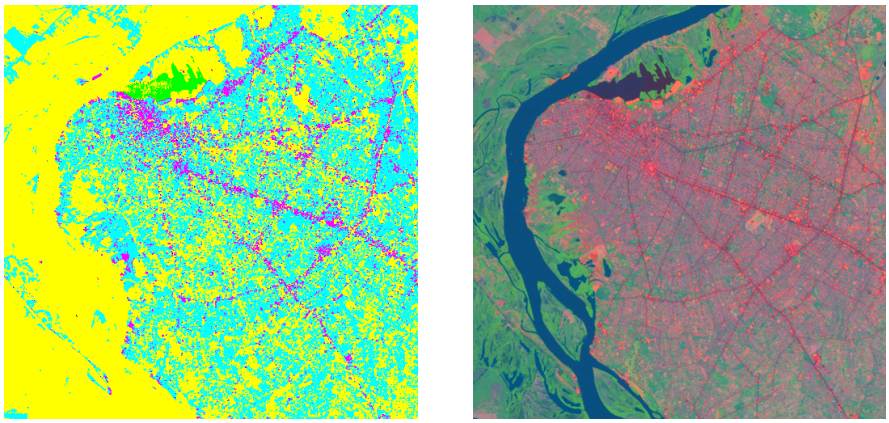
Consideremos la imagen de seis bandas de la ciudad de Asunción tomada por el Landsat-5 de la sección (2.3.4). Su formato original es HDR (por sus siglas en inglés *high dynamic range*), lo que significa que guarda una mayor cantidad de información acerca de la intensidad luminosa de la escena que está capturando. En principio el formato en el que se almacena una imagen de esta naturaleza no es apropiado para realizar las operaciones de clasificación y componentes principales según la implementación que se toma en este trabajo. Por esta razón hemos desglosado la imagen en otras dos de tres bandas cada una, de modo que, como puede verse en la figura 7, la imagen (a) guarda las tres primeras bandas y la (b) las últimas tres.



(a) Primeras tres bandas. *(b) Últimas tres bandas.*

Figura 8: Imagen de seis bandas de Asunción tomada por el Landsat-5.

A los propósitos de aplicar los algoritmos de PCA y *Kmeans*, una vez realizada la lectura de ambas imágenes, se concatenan los arreglos para formar una matriz única que incluya la información de las seis bandas. Sólo por curiosidad, corramos Kmeans por un lado, y PCA por el otro para ver qué obtenemos. La figura 8.(a) es la imagen clasificada con Kmeans, la 8.(b) es la imagen luego de aplicarle el algoritmo de PCA.



(a) Imagen de Asunción de seis bandas clasificada con kmeans. *(b) Imagen de Asunción de seis bandas luego de aplicado el algoritmo de PCA.*

Figura 9: Kmeans *y PCA aplicados a la imagen de Asunción de seis bandas.*

¿Similares? Quitando el hecho de que los colores usados son diferentes, es claro que ambas operaciones captan e interpretan los datos de manera muy parecida. Esto no es coincidencia, y nos invita a pensar que hay un "camino de ida y otro de vuelta" entre PCA y Kmeans.

Llamemos K a la cantidad de clases, $C_1, ..., C_K$ a las categorías, y n_i a la cantidad de elementos de C_i. A partir de ahora hasta que finalice la sección nos abocaremos a estudiar los aspectos formales de lo que ocurre con lo anterior-

mente descripto. Comencemos por un caso sencillo.

3.3.1. K=2.

Bien, queremos explorar aquí cómo obtener una clasificación *kmeans* realizando PCA, para lo cual analizaremos la idea propuesta por Ding y He (2004) [2]. Recordemos la nomenclatura que ya planteamos en secciones anteriores: $X_{(n\times m)}$ es nuestra muestra de tamaño n en donde cada vector x_i tiene longitud m. Si consideramos la matriz Y que representa los datos centrados, a_i es un autovector de la matriz de covarianza muestral $S = \frac{1}{m-1}Y'Y$, y l_i su autovalor correspondiente. Además, llamaremos z_i al vector que contiene la proyección de los datos sobre la componente a_i de modo que

$$z_i = Y a_i \frac{1}{l_i^{\frac{1}{2}}}.$$

Notar que el SVD de Y cumple la siguiente ecuación:

$$\begin{aligned} Y &= \sum_{k=1}^{r} l_k^{-\frac{1}{2}} Y a_k l_k^{\frac{1}{2}} a_k' \\ &= \sum_{k=1}^{r} z_k l_k^{\frac{1}{2}} a_k'. \end{aligned} \tag{3.3.1}$$

Al realizar clasificación por *kmeans* minimizamos la suma de cuadrados de los errores, representada como sigue:

$$J_K = \sum_{k=1}^{K} \sum_{i\in C_k} (x_i - \boldsymbol{m}_k)^2, \tag{3.3.2}$$

en donde

$$\boldsymbol{m}_k = \frac{1}{n_k} \sum_{i\in C_k} x_i,$$

que es el centroide de C_k. Miremos qué ocurre cuando $K = 2$. Podemos imaginarnos que cada cateogría es una bolsa en donde guardamos puntos, entonces minimizar la función J_K sería equivalente a intentar aglutinar los elementos que están dentro de una misma bolsa. La intuición nos invita a pensar que esto es lo mismo que maximizar la distancia entre los puntos de distintas bolsas, si asumimos que están fijas. Definamos formalmente esta distancia.

Definición 15. *La* distancia entre dos clases *se define como sigue:*

$$d(C_k, C_l) = \sum_{i\in C_k} \sum_{j\in C_l} (x_i - x_j)^2.$$

Notar que J_K se puede representar en función de d:

$$\begin{aligned} J_K &= \sum_{k=1}^{K} \sum_{i,j\in C_k} \frac{(x_i - x_j)^2}{2n_k} \\ &= n\bar{x^2} - \frac{1}{2} J_D \end{aligned}$$

en donde para $K = 2$ tenemos

$$J_D = \frac{n_1 n_2}{n} \left(2\frac{d(C_1, C_2)}{n_1 n_2} - \frac{d(C_1, C_1)}{n_1^2} - \frac{d(C_2, C_2)}{n_2^2} \right),$$

que es siempre positivo. Observar que minimizar J_K implica maximizar J_D. Más aún, tenemos

$$Cov(C_k, C_l) = \frac{d(C_k, C_l)}{n_k n_l}$$

y además

$$Var(C_k) = \frac{d(C_k, C_k)}{n_k^2}.$$

Entonces, por cómo hemos definido J_D, al maximizarlo estamos también maximizando la distancia entre clases, y por lo tanto, la covarianza. Como vemos, el problema de *kmeans* ya comienza a tomar la forma de uno de PCA.

Definamos la matriz de distancias $D_{(n\times n)}$ del siguiente modo:

$$D_{(i,j)} = ||x_i - x_j||^2.$$

Definimos también un vector $q_{n\times 1}$ que en su i-ésimo lugar contendrá información acerca de si x_i pertenece a la clase C_1 o no (recordemos que estamos analizando el caso $K = 2$, por lo que contamos sólo con dos clases, C_1 y C_2). Entonces q se define así:

$$q_i = \begin{cases} \sqrt{\frac{n_2}{n_1 n}} & \text{, si } x_i \in C_1 \\ -\sqrt{\frac{n_1}{n_2 n}} & \text{, si } x_i \in C_2 \end{cases}.$$

El vector q no fue elegido al azar, de hecho cumple con propiedades muy importates: (a) $\sum_{i=1}^n q_i = 0$ y (b) $\sum_{i=1}^n q_i^2 = 1$. Además,

$$q'Dq = -J_D.$$

Si consideramos una función J definida de manera tal que

$$J(p) = \frac{p'}{||p||} D \frac{p}{||p||} = \frac{p'Dp}{p'p}$$

en donde p es un vector que puede tomar valores en el intervalo $[-1, 1]$ y que tiene las mismas propiedades que le atribuímos a q, entonces la minimización de J está dada por el autovector correspondiente al mínimo autovalor de la ecuación

$$Dt = \lambda t,$$

que corresponde al mayor autovalor negativo [1]. Más aún, consideremos la matriz de distancias centrada dada por

$$\hat{D} = D - \boldsymbol{\mu_{fil}} \boldsymbol{e'} - \boldsymbol{e} \boldsymbol{\mu'_{col}} + M, \tag{3.3.3}$$

en donde $\boldsymbol{\mu_{fil}}$ y $\boldsymbol{\mu_{col}}$ son los vectores que en el j-ésimo lugar contienen la media de la fila j y la media de la columna j de la matriz D, respectivamente, y M es

[1] Queremos maximizar J_D, lo cual implica minimizar $-J_D$, i.e., la función J.

una matriz tal que $M_{(i,j)} = \boldsymbol{\mu_{total}}$ para todo $i, = 1, ..., n$. Gracias a la propiedad (a) del vector q tenemos que

$$(\boldsymbol{\mu_{fil}e'})q = 0$$

y

$$q'(\boldsymbol{e\mu'_{col}}) = 0.$$

Además, como la matriz M es constante tanto para filas como para columnas, se anula del mismo modo. Luego,

$$q'\hat{D}q = q'Dq = -J_D,$$

por lo que la solución para la minimización de la función J es el autovector correspondiente al menor autovalor de $\hat{D}$ de modo que

$$\hat{D}t = \lambda t. \tag{3.3.4}$$

La belleza de usar $\hat{D}$ en lugar de D es que

$$\hat{D} = -2Y'Y. \tag{3.3.5}$$

Veamos que esto se cumple. Notar que

$$\begin{aligned}
\boldsymbol{\mu_{fil}(i)} &= \frac{\sum_{k=1}^{n} ||x_i - x_k||^2}{n} \\
&= \frac{\sum_{k=1}^{n} < x_i - x_k, x_i - x_k >}{n} \\
&= \frac{\sum_{k=1}^{n} < x_i, x_i > -2 < x_i, x_k > + < x_k, x_k >}{n} \\
&= \frac{\sum_{k=1}^{n} < x_i, x_i >}{n} - 2\frac{\sum_{k=1}^{n} < x_i, x_k >}{n} + \frac{\sum_{k=1}^{n} < x_k, x_k >}{n} \\
&=< x_i, x_i > -2 < \bar{x}, x_i > + \bar{x^2}
\end{aligned}$$

y que

$$\begin{aligned}
M_{(i,j)} &= \frac{1}{n^2} \sum_{i,j=1}^{n} ||x_i - x_j||^2 \\
&= \frac{1}{n^2} \sum_{i,j=1}^{n} < x_i - x_j, x_i - x_j > \\
&= \frac{1}{n^2} \sum_{i,j=1}^{n} (< x_i, x_i > -2 < x_i, x_j > + < x_j, x_j >) \\
&= \frac{1}{n^2} \sum_{i,j=1}^{n} < x_i, x_i > -2\frac{1}{n^2} \sum_{i,j=1}^{n} < x_i, x_j > + \frac{1}{n^2} \sum_{i,j=1}^{n} < x_j, x_j > \\
&= 2\bar{x^2} - 2 < \bar{x}, \bar{x} > \\
&= 2(\bar{x^2} - < \bar{x}, \bar{x} >) \\
&= 2\bar{y^2},
\end{aligned}$$

por lo tanto la ecuación 3.3.3 se desarrollaría de este modo:

$$\begin{aligned}\hat{D}_{(i,j)} &= D_{(i,j)} - (< x_i, x_i > -2 < \bar{x}, x_i > + \bar{x^2}) - (< x_j, x_j > -2 < \bar{x}, x_j > + \bar{x^2}) - 2\bar{y^2} \\ &=< x_i, x_i > -2 < x_i, x_j > + < x_j, x_j > - < x_i, x_i > +2 < x_i, \bar{x} > -\bar{x^2} - < x_j, x_j > +2 < x_j, \bar{x} > \\ &- \bar{x^2} + 2\bar{y^2} \\ &= -2(< x_i, x_j > - < x_i, \bar{x} > - < x_j, \bar{x} > + < \bar{x}, \bar{x} >) \\ &= -2(Y'Y)_{(i,j)}.\end{aligned}$$

La importancia de las ecuaciones 3.3.4 y 3.3.5 reside en el hecho de que reducen el problema de encontrar el mínimo de la función J a hallar el autovector de $Y'Y$ correspondiente a su mayor autovalor, y luego proyectar los datos sobre él. Esto equivale a tomar el mayor z_i de la descomposición de Y en valores singulares que figura en la ecuación 3.3.1. Una vez que hemos obtenido el mayor vector de proyecciones, z_1, definimos

$$C_1 = \{x_i / z_1(i) \leq 0\}$$

y

$$C_2 = \{x_i / z_1(i) > 0\}.$$

De esta manera, el punto x_i pertenece a C_1 si su proyección sobre la primera componente es menor o igual a 0, y a C_2 en caso contrario. Hemos obtenido una inicialización muy buena de las clases. Si computamos los centroides de C_1 y C_2 y corremos *kmeans*, el algoritmo convergirá muy rápido.

3.3.2. Caso general.

Habiendo comprendido cómo funciona esta idea para $K = 2$, nos preguntamos cómo generalizarla. Estudiemos la ecuación **??**, tenemos

$$\begin{aligned}J_K &= \sum_{k=1}^{K} \sum_{i,j \in C_k} \frac{(x_i - x_j)^2}{2n_k} \\ &= \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,j \in C_k} (x_i - x_j)^2 \\ &= \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,j \in C_k} < x_i, x_i > -2 < x_i, x_j > + < x_j, x_j > \\ &= \frac{1}{2} \sum_{k=1}^{K} \sum_{i,j \in C_k} < x_i, x_i > - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,j \in C_k} < x_i, x_j > + \frac{1}{2} \sum_{k=1}^{K} \sum_{i,j \in C_k} < x_j, x_j > \\ &= \sum_{i=1}^{n} < x_i, x_i > - \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,j \in C_k} < x_i, x_j > .\end{aligned}$$

Observar que el primer término corresponde a la traza de la matriz XX' y que el segundo término puede ser interpretado como la traza de una matriz $\mathscr{H}$ de la cual sólo sabemos que contiene información acerca de qué puntos pertenecen a

esta o aquella categoría. Entonces, ¿quién es $\mathscr{H}$? Definamos el vector de tamaño n, $\boldsymbol{h_k}$, de manera tal que

$$\boldsymbol{h_k}(i) = \begin{cases} 1/n_k^{\frac{1}{2}} & \text{, si } x_i \in C_k \\ 0 & \text{, caso contrario} \end{cases},$$

para $k = 1, ..., K$, y definamos también

$$H_{(K\times K)} = \begin{pmatrix} \boldsymbol{h_1'} \\ \vdots \\ \boldsymbol{h_K'} \end{pmatrix}. \tag{3.3.6}$$

Tomamos $\mathscr{H} = HXX'H'$. La matriz $\mathscr{H}$ es tal que

$$\begin{aligned} \mathscr{H}_{(i,j)} &= \boldsymbol{h_i'} XX' \boldsymbol{h_j} \\ &= \frac{1}{\sqrt{n_i n_j}} \sum_{p\in C_i, k\in C_j} < x_p, x_k >, \end{aligned}$$

luego tenemos que

$$\begin{aligned} Tr(\mathscr{H}) &= \mathscr{H}_{(1,1)} + ... + \mathscr{H}_{(K,K)} \\ &= \boldsymbol{h_1'} XX' \boldsymbol{h_1} + ... + \boldsymbol{h_K'} XX' \boldsymbol{h_K} \\ &= \frac{1}{n_1} \sum_{i,j\in C_1} < x_i, x_j > + ... + \frac{1}{n_K} \sum_{i,j\in C_K} < x_i, x_j > \\ &= \sum_{k=1}^{K} \frac{1}{n_k} \sum_{i,j\in C_k} < x_i, x_j > . \end{aligned}$$

Por lo tanto

$$J_K = Tr(XX') - Tr(\mathscr{H}). \tag{3.3.7}$$

Con esto hemos llevado el problema de la minimización de J_K a un problema de minimización de trazas. Sin embargo, dado que los valores de la muestra son constantes, la traza de XX' también lo es, esto significa que lo que en realidad buscaremos es maximizar $Tr(\mathscr{H})$. Recordemos la propiedad algebráica 1 de la sección (2.3.2) y observemos que la matriz $\mathscr{H} = HXX'H'$ no cumple con todos los requerimientos de la hipótesis[1], pues la matriz H no es ortogonal. Además

$$\sum_{k=1}^{K} n_k^{1/2} \boldsymbol{h_k} = \boldsymbol{e}$$

implica que existe un r, $1 \leq r \leq K$ tal que

$$\boldsymbol{h_r} = \frac{1}{n_r^{1/2}} \boldsymbol{e} - \sum_{k=1, k\neq r}^{K} \frac{n_k^{1/2}}{n_r^{1/2}} \boldsymbol{h_k},$$

[1]Si bien en esta propiedad se hace mención de una matriz de covarianza, el resultado es válido para cualquier matriz simétrica definida positiva.

esto es, existe un $\boldsymbol{h_r}$ que puede ser expresado como combiandión lineal de los demás. Realizamos entonces una transformación lineal T sobre H del siguiente modo:

$$Q'_K = (\boldsymbol{q_1}|...|\boldsymbol{q_K}) = H'T, \tag{3.3.8}$$

en donde $T_{(K\times K)}$ es una matriz ortonormal cuya última columna cumple

$$\boldsymbol{t_K} = \begin{bmatrix} \sqrt{\frac{n_1}{n}} \\ \vdots \\ \sqrt{\frac{n_K}{n}} \end{bmatrix}.$$

Los vectores $\boldsymbol{q_i}$ son ortogonales. Veamos ésto. Tomemos k, l tal que $1 \leq k, l \leq K, k \neq l$, entonces

$$\begin{aligned} < \boldsymbol{q_k}, \boldsymbol{q_l} > &=< H'\boldsymbol{t_k}, H'\boldsymbol{t_l} > \\ &= \langle \sum_{p=1}^{K} \boldsymbol{h_p} T_{(p,k)}, \sum_{p=1}^{K} \boldsymbol{h_p} T_{(p,k)} \rangle \\ &=< \boldsymbol{h_1}, \boldsymbol{h_1} > T_{(1,k)}T_{(1,l)} + ... + < \boldsymbol{h_K}, \boldsymbol{h_K} > T_{(K,k)}T_{(K,l)} \end{aligned}$$

Como los $\boldsymbol{h_i}$ son ortogonales y de norma 1 la ecuación continúa así:

$$\begin{aligned} &= T_{(1,k)}T_{(1,l)} + ... + T_{(K,k)}T_{(K,l)} \\ &=< \boldsymbol{t_k}, \boldsymbol{t_l} > \\ &= 0. \end{aligned}$$

Tomamos las primeras $K-1$ columnas de Q'_K definiendo

$$Q'_{K-1} = (\boldsymbol{q_1}|...|\boldsymbol{q_{K-1}}).$$

Q_{K-1} es ortogonal, esto es, $Q'_{K-1}Q_{K-1} = I_{K-1}$ y además cada uno de los $\boldsymbol{q_i}$ es ortogonal al vector $\boldsymbol{e}$. Si tenemos en cuenta que

$$Tr(Q_K XX' Q'_K) = \boldsymbol{q'_1} XX' \boldsymbol{q_1} + ... + \boldsymbol{q'_K} XX' \boldsymbol{q_K}$$

por definición de $\boldsymbol{q_i}$,

$$\begin{aligned} &= (T_{(1,1)}\boldsymbol{h'_1} + ... + T_{(K,1)}\boldsymbol{h'_K})XX'(T_{(1,1)}\boldsymbol{h_1} + ... + T_{(K,1)}\boldsymbol{h_K}) \\ &+ ... + (T_{(1,K)}\boldsymbol{h'_1} + ... + T_{(K,K)}\boldsymbol{h'_K})XX'(T_{(1,K)}\boldsymbol{h_1} + ... + T_{(K,K)}\boldsymbol{h_K}) \\ &= (T^2_{(1,1)}\boldsymbol{h'_1}XX'\boldsymbol{h_1} + ... + T^2_{(K,1)}\boldsymbol{h'_K}XX'\boldsymbol{h_K}) \\ &+ ... + (T^2_{(1,K)}\boldsymbol{h'_1}XX'\boldsymbol{h_1} + ... + T^2_{(K,K)}\boldsymbol{h'_K}XX'\boldsymbol{h_K}) \\ &= \boldsymbol{h'_1}XX'\boldsymbol{h_1}\left(\sum_{p=1}^{K} T^2_{(1,p)}\right) + ... + \boldsymbol{h'_K}XX'\boldsymbol{h_K}\left(\sum_{p=1}^{K} T^2_{(K,p)}\right) \end{aligned}$$

Por ser T ortogonal,

$$\begin{aligned} &= \boldsymbol{h'_1}XX'\boldsymbol{h_1} + ... + \boldsymbol{h'_K}XX'\boldsymbol{h_K} \\ &= Tr(HXX'H'), \end{aligned}$$

entonces podemos reescribir J_K:

$$\begin{aligned} J_K &= Tr(XX') - Tr(Q_K XX' Q_K') \\ &= Tr(XX') - \boldsymbol{q_1'} XX' \boldsymbol{q_1} - ... - \boldsymbol{q_{K-1}'} XX' \boldsymbol{q_{K-1}} - \boldsymbol{q_K'} XX' \boldsymbol{q_K} \end{aligned}$$

Por definición de Q_{K-1},

$$= Tr(XX') - \boldsymbol{q_K'} XX' \boldsymbol{q_K} - Tr(Q_{K-1} XX' Q_{K-1}')$$

Por definición de $\boldsymbol{q_K}$,

$$= Tr(XX') - (T_{(1,K)} \boldsymbol{h_1'} + ... + T_{(K,K)} \boldsymbol{h_K'}) XX' (T_{(1,K)} \boldsymbol{h_1} + ... + T_{(K,K)} \boldsymbol{h_K}) - Tr(Q_{K-1} XX' Q_{K-1}')$$

Por definición de t_K,

$$\begin{aligned} &= Tr(XX') - \left(\sqrt{\frac{n_1}{n}} \boldsymbol{h_1'} + ... + \sqrt{\frac{n_K}{n}} \boldsymbol{h_K'} \right) XX' \left(\sqrt{\frac{n_1}{n}} \boldsymbol{h_1} + ... + \sqrt{\frac{n_K}{n}} \boldsymbol{h_K} \right) - Tr(Q_{K-1} XX' Q_{K-1}') \\ &= Tr(XX') - \sqrt{\frac{1}{n}} \boldsymbol{e'} XX' \sqrt{\frac{1}{n}} \boldsymbol{e} - Tr(Q_{K-1} XX' Q_{K-1}') \\ &= Tr(XX') - \frac{1}{n} \boldsymbol{e'} XX' \boldsymbol{e} - Tr(Q_{K-1} XX' Q_{K-1}'). \end{aligned} \tag{3.3.9}$$

No perdamos de vista la definición de J_K expresada en la ecuación 3.3.2. Allí vemos que podemos reemplazar en 3.3.9 X por Y dado que en J_K, al usar Y, las medias serán cero y no habrá diferencia alguna. Obtenemos entonces

$$J_K = Tr(YY') - Tr(Q_{K-1} YY' Q_{K-1}'), \tag{3.3.10}$$

dado que el segundo término de 3.3.9 se hace cero valuado en Y.

Asumamos que H puede tomar valores continuos entre 0 y 1 y supongamos que Q_K es un caso particular para la H definida como en 3.3.6. Esto significa que estamos pensando en la transformación $H'T$ como en una función que dada una matriz H devuelve otra $Q_{(K \times K)}$. Entonces $Tr(Q_{K-1} YY' Q_{K-1}')$ también es un caso particular de una función de Q[1]. La propiedad algebraica 1 nos indica que $Tr(Q_{K-1} YY' Q_{K-1}')$ alcanza su máximo cuando

$$Q_{K-1}' = (z_1 | ... | z_{K-1}), \tag{3.3.11}$$

en donde $z_1, ..., z_{K-1}$ son los autovectores de YY' correspondientes a los primeros $K-1$ autovalores de YY' ordenados de mayor a menor. Notar que $z_1, ..., z_{K-1}$ se obtienen del SVD dada por 3.3.1. Más aún,

$$Tr(Q_{K-1} YY' Q_{K-1}') = l_1 + ... + l_{K-1}.$$

Esto es equivalente a decir que los vectores $\boldsymbol{q_1}, ..., \boldsymbol{q_{K-1}}$ son una aproximación discreta de la solución continua dada por $z_1, ..., z_{K-1}$.

Estamos en condiciones de resumir un importantísimo resultado en una propiedad:

[1] A la Q devuelta por la función $H'T$ le quitamos la última columna.

Propiedad 4. *Llamemos P a la matriz de proyección de un vector cualquiera $\boldsymbol{x}$ sobre el espacio generado por los centroides $\boldsymbol{m}_i$, esto es,*

$$P = \sum_{k=1}^{K} < \boldsymbol{m}_k, \boldsymbol{m}_k >,$$

de modo que

$$P\boldsymbol{x} = \sum_{k=1}^{K} < \boldsymbol{m}_k, \boldsymbol{x} > \boldsymbol{m}_k.$$

Entonces

$$P = \sum_{k=1}^{K-1} l_k a_k a_k'. \tag{3.3.12}$$

En otras palabras, el subespacio generado por los centroides de los clusters *es el mismo que generan las $K-1$ componentes principales.*

Demostración. Sea

$$\tilde{\boldsymbol{m}}_{\boldsymbol{k}} = \frac{1}{n_k} \sum_{i \in C_k} y_i$$

el centroide de los datos de la clase C_k centrados. Observar que $\boldsymbol{m}_1, ..., \boldsymbol{m}_K$ y $\tilde{\boldsymbol{m}}_1, ..., \tilde{\boldsymbol{m}}_K$ son bases del mismo subespacio dado que

$$\boldsymbol{m}_k = \tilde{\boldsymbol{m}}_k + \bar{x}.$$

Entonces el subespacio generado por $\boldsymbol{m}_1, ..., \boldsymbol{m}_K$ puede ser representado mediante $\sum_{i \in C_1} y_i, ..., \sum_{i \in C_K} y_i$. Para un determinado k tenemos

$$\sum_{i \in C_k} y_i = \sum_i \boldsymbol{h}_k(i) y_i = Y' \boldsymbol{h}_k.$$

Luego

$$\begin{aligned}
P &= \sum_{k=1}^{K} Y' \boldsymbol{h}_k \boldsymbol{h}_k' Y \\
&= Y' \left(\sum_{k=1}^{K} \boldsymbol{h}_k \boldsymbol{h}_k' \right) Y \\
&= Y' \left(\sum_{k=1}^{K} \boldsymbol{q}_k \boldsymbol{q}_k' \right) Y && \text{, porque } TT' = I. \\
&= Y' \left(\frac{\boldsymbol{e}\boldsymbol{e}'}{n} + \sum_{k=1}^{K-1} \boldsymbol{q}_k \boldsymbol{q}_k' \right) Y && \text{, separamos el término } k = K. \\
&= Y' \frac{\boldsymbol{e}\boldsymbol{e}'}{n} Y + Y' \left(\sum_{k=1}^{K-1} \boldsymbol{q}_k \boldsymbol{q}_k' \right) Y \\
&= Y' \left(\sum_{k=1}^{K-1} \boldsymbol{q}_k \boldsymbol{q}_k' \right) Y.
\end{aligned}$$

Considerando la ecuación 3.3.11 podemos afirmar que el subespacio generado por $\sum_{k=1}^{K-1} \boldsymbol{q}_k \boldsymbol{q}_k'$ está dado por $\sum_{k=1}^{K-1} z_k z_k'$, entonces la ecuación anterior continúa:

$$\begin{aligned}
&= Y' \left(\sum_{k=1}^{K-1} z_k z_k' \right) Y \\
&= \sum_{k=1}^{K-1} Y' z_k z_k' Y \\
&= \sum_{k=1}^{K-1} Y' z_k (Y' z_k)' \\
&= \sum_{k=1}^{K-1} \frac{Y'Y a_k}{l_k^{1/2}} \frac{(Y'Y a_k)'}{l_k^{1/2}} \\
&= \sum_{k=1}^{K-1} l_k a_k a_k'.
\end{aligned}$$

Con eso finaliza la prueba. □

3.3.3. ¿Quién es la transformación T?

Notar que habiendo computado PCA estamos a una transformación lineal T de distancia de kmeans, y viceversa. Entonces, ¿cómo averiguar T?

Definamos los $K(K-1)/2$ números positivos arbitrarios $\alpha_{i,j}$, $i, j = 1, ..., n$, que son tales que

- $\sum_{1 \leq i \leq j \leq n} \alpha_{i,j} = 1$, y
- $\alpha_{i,j} = \alpha_{j,i}$.

Definamos también

$$\Omega = \begin{pmatrix} \sqrt{n_1} & 0 & 0 & \cdots & 0 \\ 0 & \sqrt{n_2} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sqrt{n_K} \end{pmatrix},$$

y la matriz $\tilde{\Gamma}$ que es tal que

$$\tilde{\Gamma}_{(i,j)} = \begin{cases} -\alpha_{i,j} & \text{, si } i \neq j \\ \sum_{p=1, p \neq i}^{K} \alpha_{i,p} & \text{, si } i = j \end{cases}.$$

Sea

$$\Gamma = \Omega^{-1/2} \tilde{\Gamma} \Omega^{-1/2}.$$

Para cualquier vector $x = (x_1, ..., x_K)'$ se cumple

$$2 \sum_{i,j} x_i \tilde{\Gamma} x_j = \sum_{i,j} \alpha_{i,j} (x_i - x_j)^2.$$

Veamos ésto:

$$\begin{aligned}\sum_{i,j}\alpha_{i,j}(x_i-x_j)^2 &= \sum_{i,j}\alpha_{i,j}(x_i^2-2x_ix_j+x_j^2)\\ &= \sum_{i,j}\alpha_{i,j}x_i^2-2\alpha_{i,j}x_ix_j+\alpha_{i,j}x_j^2\\ &= \sum_i\tilde{\Gamma}_{(i,i)}x_i^2-2\sum_{i,j,i\neq j}\alpha_{i,j}x_ix_j+\sum_i\tilde{\Gamma}_{(i,i)}x_i^2\\ &= 2\sum_{i,j}x_i\tilde{\Gamma}_{(i,j)}x_j.\end{aligned}$$

Entonces Γ es simétrica y definida positiva.

La siguiente propiedad nos revela la identidad de la transformación lineal que estamos buscando:

Propiedad 5. *La transformación lineal T está formada por los primeros K autovectores de Γ.*

La demostración está fuera del alcance de este trabajo.

Si tenemos una clasificación podemos confeccionar la matriz H, que, contando con T, es equivalente a tener Q, y por lo tanto las componentes principales. Si por el contrario computamos PCA, el camino a recorrer para obtener los clusters es más sinuoso dado que no contamos con los valores de $n_1, ..., n_K$ necesarios para definir Ω, y con ello, T. La pregunta entonces es ¿cómo conseguir una clasificación a partir de PCA sin la matriz T? Tengamos en cuenta lo postulado en la propiedad 4: el subespacio generado por los centroides de los clusters es el mismo que generan las primeras $K-1$ componentes principales de la matriz de covarianza $Y'Y$. Miremos el subespacio de mayor dimensión que generan las componentes principales de YY', $z_1, ..., z_{K-1}$,

$$YY' = \sum_{k=1}^{K} l_k z_k z_k'.$$

Como nos interesa el subespacio podemos deshacernos de las constantes l_k. Definimos

$$\begin{aligned}C &= \frac{\boldsymbol{ee'}}{n} + \sum_{k=1}^{K-1} z_k z_k'\\ &= \frac{\boldsymbol{ee'}}{n} + \sum_{k=1}^{K-1} q_k q_k'\\ &= \sum_{k=1}^{K} q_k q_k'\\ &= \sum_{k=1}^{K} h_k h_k' \qquad\qquad \text{, por ser } T \text{ ortogonal.}\end{aligned}$$

Observar que la matriz $\sum_{k=1}^{K} h_k h_k'$ tiene una estructura de bloque en diagonal que puede interpretarse de manera tal que si $C_{(i,j)} > 0$ entonces x_i y x_j se

encuentran en la misma categoría. Más aún, es factible asociar a este hecho una probabilidad $p_{i,j} = \frac{C_{(i,j)}}{C_{(i,i)}^{1/2} C_{(j,j)}^{1/2}}$. Si la clasificación es perfecta, no debería haber números mayores a cero en la periferia de los bloques diagonales, ni $p_{i,j}$ pequeños. Si bien en la práctica esto no se da con frecuencia, podemos realizar una "limpieza" llevando a cero aquellos valores cuya probabilidad es menor a un número η fijo y así mejorar el resultado.

4. Otras técnicas.

CA puede ser combinado con otras técnicas de análisis multivariado. En esta sección mencionaremos algunas de ellas.

4.1. Análisis discriminante.

El análisis discriminante tiene una estructura nueva. Para empezar las observaciones provienen de algún grupo $\mathcal{G}_i$, dados r grupos poblacionales $\mathcal{G}_1, ..., \mathcal{G}_r$, para algún $r \geq 2$, y se asume que la variable aleatoria x asociada a cada observación tiene una distribución particular según el grupo al que pertenezca. Contamos además con un conjunto de entrenamiento formado por las observaciones $x_1, ..., x_n$ cuya pertenencia a tal o cuál grupo nos hes conocida. El objetivo es asignar las futuras observaciones a un determinado grupo obedeciendo alguna regla óptima, teniendo en cuenta la estructura de cada $\mathcal{G}_i$ y cómo se reparten las observaciones del conjunto de entrenamiento. El método de asignación puede ser lineal o no. A continuación describiremos uno lineal: LDA, por sus siglas en inglés *Linear Discriminant Analysis*.

4.1.1. Derivación de LDA

En el caso más usual contamos con dos grupos, $\mathcal{G}_1$ y $\mathcal{G}_2$, cada uno de los cuales tiene media μ_1 y μ_2 respectivamente. Además asumimos que x tiene una distribución normal multivariada y que tienen la misma matriz de covarianza, Σ, para $\mathcal{G}_1$ y $\mathcal{G}_2$. La derivación de la regla óptima para este caso en particular es muy similar a la que usamos para obtener la de PCA. Partimos de la base de que mientras más separados se encuentren los grupos mejor podemos predecir la pertenencia de las futuras observaciones. Esto reduce el problema a encontrar la línea

$$z = w'x$$

que mejor separe $\mathcal{G}_1$ de $\mathcal{G}_2$, es decir, maximizar una función de w. Fischer propuso

$$max \frac{\text{dispersión entre clases}}{\text{dispersión dentro de las clases}} = max_w \frac{w'S_B w}{w'\Sigma w},$$

en donde

$$S_B = (\mu_2 - \mu_1)(\mu_2 - \mu_1)'.$$

Basta con maximizar el numerador manteniendo el denominador constante, digamos $w'\Sigma w = H$. Al igual que en la derivación de PCA, usamos multiplicadores de Lagrange obteniendo

$$w'S_B w - \zeta(w'\Sigma w - H).$$

Haciendo algunas cuentas llegamos a

$$
\begin{aligned}
S_B w &= \zeta \Sigma w \\
&\Rightarrow \Sigma^{-1} S_B w = \zeta w \\
&\Rightarrow \Sigma^{-1}(\mu_2 - \mu_1)(\mu_2 - \mu_1)' w = \zeta w && \text{, reemplazando } S_B \text{ por su valor.} \\
&\Rightarrow \Sigma^{-1}(\mu_2 - \mu_1)\gamma = \zeta w && \text{, porque } (\mu_2 - \mu_1)' w \text{ es un escalar que llamamos } \gamma. \\
&\Rightarrow \Sigma^{-1}(\mu_2 - \mu_1) = \frac{\zeta}{\gamma} w \\
&\Rightarrow \Sigma^{-1}(\mu_2 - \mu_1) = w && \text{, porque no es importante la magnitud de } w.
\end{aligned}
$$

Entonces el criterio para decidir a qué clase pertenece una observación depende de si su correspondiente variable aleatoria se encuentra en un determinado lado del hiperplano perpendicular a w. En la práctica trabajamos con estimadores de Σ, μ_1 y μ_2 de modo que

$$w = S^{-1}(\bar{x}_1 - \bar{x}_2)$$

será la regla en donde $S, \bar{x}_2$ y $\bar{x_1}$ son estimadores de Σ, μ_2 y μ_1 respectivamente.

4.1.2. LDA y PCA.

El objetivo de combinar PCA con LDA es reducir x a sus primeras t componentes principales. Para esto hay varios caminos que se pueden tomar determinados por cuál matriz de covarianza usar.

Covarianza sobre el total de los datos. Es sabido que la probabilidad de clasificar erróneamente una observación está ligada a la distancia entre los grupos, de manera análoga al razonamiento de Kmeans. Entonces el error es una función de la distancia dada por

$$\Delta^2 = (\mu_1 - \mu_2)' \Sigma^{-1} (\mu_1 - \mu_2). \tag{4.1.1}$$

Puede ser demostrado que, dado que Σ es la matriz de covarianza de todos los datos y que, por lo tanto, ignora la estructura de grupos, la distancia entre poblaciones expresada en 4.1.1 basada en la k-ésima componente principal es una función creciente

$$\theta_k = \frac{(z_k'(\mu_1 - \mu_2))^2}{l_k}$$

en donde z_k y l_k tienen los significados usuales. Lo importante es que el autovector de Σ que maximiza θ_k y que por consiguiente puede discriminar mejor las observaciones no es necesariamente la componente principal que maximiza la varianza.

Usar distintas covarianzas para distintos grupos. Si tomamos Σ_i como matriz de covarianza de $\mathcal{G}_i$ perdemos la posibilidad de derivar una regla lineal. Para ello deberíamos asumir que todos los grupos tienen la misma matriz de covarianza, como en el apartado (4.1.1). No obstante es posible tomar otro enfoque: la discriminación cuadrática. Tomemos S_i y $\bar{x}_i$, estimadores de Σ_i y μ_i

respectivamente. Según esta técnica una observación se asigna al conjunto para el cual

$$(x - \bar{x}_i)' S_i^{-1} (x - \bar{x}_i) + ln(|S_i|)$$

es minimizada.

PCA sobre cada grupo. Una alternativa útil para mantener la linearidad del método es computar PCA para cada grupo por separado. Hecho esto, una nueva observación se asigna al grupo que es tal que la distancia entre esta y el hiperplano formado por sus componentes principales es la mínima de todos los grupos.

4.2. Análisis de correlación canónica.

Sean $x = (x_1, ..., x_n)$ y $y = (y_1, ..., y_m)$ vectores de variables aleatorias. Supongamos que queremos estudiar cómo se relacionan. Sabemos que la matriz de covarianzas cruzadas nos será insuficiente porque esta sólo puede explicar cómo se asocian dos variables, y la regresión lineal explica qué relación existe entre una sola variable, digamos x_i, y un conjunto de ellas, y, encontrando una combinación lineal $a'y$ que maximice su correlación con x_i.

Considerando x e y, el objetivo del análisis de correlación canónica (CCA por sus siglas del inglés *Canonical Correlation Analysis*) es encontrar un primer par de combinaciones lineales

$$\{a_1' x, b_1' y\}$$

llamadas *variables canónicas*, tales que la correlación entre ambas, $Corr(a_1' x, b_1' y)$, sea la máxima. Habiendo encontrado este primer par, se busca uno segundo que maximice la función correlación sujeto a la restricción de que no estén relacionadas con el primero. Este comportamiento se repite hasta obtener $min\{n, m\}$ pares de variables canónicas no relacionadas entre sí.

4.2.1. Derivación de CCA.

Tal como en los casos anteriores, este problema puede ser resuelto usando multiplicadores de Lagrange. Sean $\Sigma_{x,x} = Cov(x, x)$, $\Sigma_{y,y} = Cov(y, y)$, $\Sigma_{x,y} = Cov(x, y)$, esta última la matriz $n \times m$ de covarianzas cruzadas. El parámetro a maximizar es

$$max_{a,b}\{\rho\}$$

en donde

$$\rho = \frac{a' \Sigma_{x,y} b}{\sqrt{a' \Sigma_{x,x} a} \sqrt{b' \Sigma_{y,y} b}}.$$

No perdamos de vista que nos es indiferente la escala, sólo nos importa la dirección, por lo que podemos considerar como restricciones

$$a' \Sigma_{x,x} a = 1 \qquad \text{y} \qquad b' \Sigma_{y,y} b = 1. \tag{4.2.1}$$

Llevado a la forma de Lagrange esto sería

$$\mathcal{L}(\zeta_1, \zeta_2, a, b) = a' \Sigma_{x,y} b - \zeta_1 (a' \Sigma_{x,x} a - 1) - \zeta_2 (b' \Sigma_{y,y} b - 1).$$

Calculemos las derivadas y las igualemos a cero:

$$\frac{\partial \mathcal{L}}{\partial a} = \Sigma_{x,y} b - 2\zeta_1 \Sigma_{x,x} a = 0 \tag{4.2.2}$$

$$\frac{\partial \mathcal{L}}{\partial b} = \Sigma_{y,x} a - 2\zeta_2 \Sigma_{y,y} b = 0. \tag{4.2.3}$$

Veamos que $\zeta_1 = \zeta_2$. Si restamos b' veces la segunda ecuación de a' veces la primera obtenemos:

$$\begin{aligned} 0 &= a'(\Sigma_{x,y} b - 2\zeta_1 \Sigma_{x,x} a) - b'(\Sigma_{y,x} a - 2\zeta_2 \Sigma_{y,y} b) \\ &= a'\Sigma_{x,y} b - 2\zeta_1 a'\Sigma_{x,x} a - b'\Sigma_{y,x} a + 2\zeta_2 b'\Sigma_{y,y} b \\ &= 2\zeta_2 b'\Sigma_{y,y} b - 2\zeta_1 a'\Sigma_{x,x} a \\ &= 2(\zeta_2 - \zeta_1) \qquad \text{, por 4.2.1,} \end{aligned}$$

lo que implica que $\zeta_2 = \zeta_1 \doteq \zeta$.

Si despejamos a de 4.2.2, asumiendo $\Sigma_{x,x}$ invertible, obtenemos

$$a = \frac{\Sigma_{x,x}^{-1} \Sigma_{x,y} b}{\zeta},$$

y reemplazando en 4.2.3,

$$\Sigma'_{x,y} \Sigma_{x,x}^{-1} \Sigma_{x,y} b = 2\zeta^2 \Sigma_{y,y} b \tag{4.2.4}$$

y análogamente

$$\Sigma_{x,y} \Sigma_{y,y}^{-1} \Sigma'_{x,y} a = 2\zeta^2 \Sigma_{x,x} a, \tag{4.2.5}$$

de lo que deducimos las siguientes ecuaciones

$$\begin{aligned} (\Sigma'_{x,y} \Sigma_{x,x}^{-1} \Sigma_{x,y} - 2\zeta^2 \Sigma_{y,y}) b = 0 \\ (\Sigma_{x,y} \Sigma_{y,y}^{-1} \Sigma'_{x,y} - 2\zeta^2 \Sigma_{x,x}) a = 0. \end{aligned}$$

¿Cómo hallar las soluciones? Notar que 4.2.5 y 4.2.4 tienen la forma de un problema de autovectores $A\boldsymbol{w} = \lambda B\boldsymbol{w}$ que puede ser llevado a otro de la forma $\tilde{A}\boldsymbol{w} = \tilde{\lambda}\boldsymbol{w}$ si aplicamos la factorización de Cholesky a las matrices $\Sigma_{x,x}$ y $\Sigma_{y,y}$. Las soluciones estarán dadas por:

- a es un autovector de $\Sigma_{x,x}^{-1} \Sigma_{x,y} \Sigma_{y,y}^{-1} \Sigma'_{x,y}$
- b es un autovector de $\Sigma_{y,y}^{-1} \Sigma'_{x,y} \Sigma_{x,x}^{-1} \Sigma_{x,y}$
- ζ es la raíz cuadrada del autovalor de $\Sigma_{y,y}^{-1} \Sigma'_{x,y} \Sigma_{x,x}^{-1} \Sigma_{x,y}$ que corresponde a b, que es igual al autovalor de $\Sigma_{x,x}^{-1} \Sigma_{x,y} \Sigma_{y,y}^{-1} \Sigma'_{x,y}$ que corresponde a a.

Para más detalles acerca de la derivación de CCA se puede recurrir al artículo de David Weenink (2003) [8] y al reporte técnico de David R. Hardoon et al (2003) [4].

4.2.2. CCA y PCA.

La discusión acerca de cómo puede contribuir el PCA al cálculo de las variables canónicas es muy amplia, de hecho son muchas las posibilidades que se han sugerido y estudiado.

En principio es una buena idea calcular las componentes principales de x e y y aplicar sobre ellas el algoritmo de derivación de CCA. Si bien los resultados son equivalentes, dado que las componentes son combinaciones lineales de los vectores de variables, muchos autores establecen que hay ventajas tanto computacionales como interpretativas en proceder de esta manera. Un punto a tener en cuenta es que ocurre con CCA algo similar que con LDA: las combinaciones de x e y que maximizan la correlación no necesariamente se encuentran en el subespacio generado por las primeras componentes principales.

El autor [6] Jolliffe (2002) menciona casos en los que las técnicas que aquí tratamos tienen una conexión particular. Si buscamos minimizar

$$Var(a'x - b'y)$$

sujeto a

$$a'\Sigma_x a = 1 \qquad \text{y} \qquad b'\Sigma_y b = 1$$

entonces obtendremos el primer par de variables canónicas. Si consideramos la restricción

$$a'a + b'b = 1$$

en lugar de la anterior entonces $(a', b')'$ será la úlitma componente principal del vector $(x', y')'$.

5. Conclusiones.

En este trabajo hemos discutido acerca de la importancia del análisis de componentes principales en lo que respecta al procesamiento de imágenes, hemos visto que es una técnica de amplio uso interdisciplinario que facilita la interpretación de los datos y el análisis cualitativo de la información obtenida. La manera en la que elimina la redundancia conservando la estructura original de los datos nos permite usar esta herramienta como un modo de disminuir el costo computacional de la aplicación de otras técnicas multivariadas. PCA está implementado en múltiples plataformas haciéndolo accesible.

Apéndices

A. Código.

A.1. kmeans

El siguiente es el código python del algoritmo de clasificación *kmeans*. Toma como entrada una matriz, que es la representación numérica de la imagen, la cantidad de categorías que puede variar entre dos y seis, el umbral de convergencia, y el número máximo de iteraciones.

```
# -*- coding: utf-8 -*-
#INPUT: matriz (fil x col x dim), que representa la imagen a clasificar.
#OUTPUT: matriz (fil x col x rgb), rgb = 3, la cual representa otra
#la imagen de entrada clasificada.
import numpy as np
import random
import math as m


def euclidean (p, q):
        d = 0
        t = p.shape[0]
        for i in range (0, t):
                d = d+ (p[i]-q[i])**2
        d = m.sqrt(d)
        return d

def centro (cant_cat, output, img,medias):
        #medias = np.empty(cant_cat)
        cuenta = np.array([0 for i in range (0, 6)])
        for i in range (0, output.shape[0]):
                for j in range (0, output.shape[1]):
                        if (output[i][j] == [255,0,0]).all():
                                medias[0]=(medias[0]*cuenta[0] + img[i,j])
                                / (cuenta[0]+1)
                                cuenta[0]=cuenta[0]+1
                        elif (output[i][j] == [0,255,0]).all():
                                medias[1]=(medias[1]*cuenta[1] + img[i,j])
                                / (cuenta[1]+1)
                                cuenta[1]=cuenta[1]+1
                        elif (output[i][j] == [0,0,255]).all():
                                medias[2]=(medias[2]*cuenta[2] + img[i,j])
                                / (cuenta[2]+1)
                                cuenta[2]=cuenta[2]+1
                        elif (output[i][j] == [255,255,0]).all():
                                medias[3]=(medias[3]*cuenta[3] + img[i,j])
                                / (cuenta[3]+1)
                                cuenta[3]=cuenta[3]+1
                        elif (output[i][j] == [255,0,255]).all():
```

```
                                        medias[4]=(medias[4]*cuenta[4] + img[i,j])
                                        / (cuenta[4]+1)
                                        cuenta[4]=cuenta[4]+1
                                elif (output[i][j] == [0,255,255]).all():
                                        medias[5]=(medias[5]*cuenta[5] + img[i,j])
                                        / (cuenta[5]+1)
                                        cuenta[5]=cuenta[5]+1
        return medias


def kmeans (img, cant_cat, conv, max_iter):
        fil = img.shape[0]
        col = img.shape[1]
        dim = img.shape[2]
        rgb = 3

        #Generamos tantas medias como indica cant_cat,
        #de dimension dim.
        medias = np.zeros([cant_cat, dim])
        for i in range (0, cant_cat):
                for j in range (0, dim):
                        medias[i][j] = random.randint(0,255)

        #Generamos el arreglo que sera la imagen de salida:
        output = np.empty([fil, col, rgb])

        #Clasificamos:
        dis = [-1 for i in range (0,cant_cat)]
        it = 0
        dif = 999
        while (it < max_iter):
                for i in range (0, fil):
                        for j in range (0, col):
                                #Restamos el pixel (i,j) de la imagen
                                #con cada una de las medias y generamos
                                #un arreglo de distancias:
                                for k in range (0, cant_cat):
                                        dis[k] = euclidean(img[i][j],
                                                        medias[k])
                                #Cual es el subindice del menor elemento
                                #de dis? Ese numero indica la categoria
                                #a la cual incluiremos al pixel img[i][j].
                                cat=dis.index(min(dis))
                                if cat == 0:
                                        output[i][j]=[255,0,0]
                                elif cat == 1:
                                        output[i][j]=[0,255,0]
                                elif cat == 2:
                                        output[i][j]=[0,0,255]
                                elif cat == 3:
```

```
                                        output[i][j]=[255,255,0]
                                elif cat == 4:
                                        output[i][j]=[255,0,255]
                                elif cat == 5:
                                        output[i][j]=[0,255,255]
                                else:
                                        print "Demasiadas clases\n."
                        #Recalculamos las medias:
                        medaux = medias
                        medias = centro (cant_cat, output, img,medias)
                        dif = 0
                        for k in range (0,cant_cat):
                                dif = max (dif, abs(euclidean(medaux[k],medias[k])))
                        if (dif<conv):
                                print 'Ha convergido.\n'
                                break
                        it=it+1
        return output
```

A continuación, la función `main` que ejecuta `kmeans.py`:

```
# -*- coding: utf-8 -*-
import sys, optparse
import scipy.misc as sm
import numpy
from kmeans import *


class ErrorLista(Exception):
        def __init__(self, value):
                self.value = value
        def __str__(self):
                return repr(self.value)

def main ():
        parser = optparse.OptionParser(usage="uso: %prog [options]")
        parser.add_option("-c", "--cantidad",
        help="Cantidad de categorías en las que se clasificarán los píxeles."
        , default = 4)
        parser.add_option("-o","--output",help="Nombre del archivo de salida."
        , default='output.png')
        parser.add_option("-l","--convergencia",help="Umbral de convergencia."
        , default=0.01)
        (options, args) = parser.parse_args()
        if len(args)==0:
                raise ErrorLista('Ingresar al menos una imagen.')
        l = []
        for i in args:
                l.append(sm.imread(i))
        fn = options.output
        cant_cat = int(options.cantidad)
```

```
        #Concatenamos las imágenes de entrada que guardamos en l:
        if len(l)==1:
                img = l[0]
        else:
                img = numpy.concatenate((l[0],l[1]),axis=2)
                if len(l[2:])>0:
                        for i in l[2:]:
                                img = numpy.concatenate((img,i),axis=2)

        #Hasta acá todo bien.

        out = kmeans (img, cant_cat, options.convergencia, 1000)
        print ('Imagen clasificada. Resultado guardado en '
        + fn + '\n----------\n')
        sm.imsave(fn, out)


if __name__=='__main__':
        main()
```

A.2. PCA

Código python del algoritmo de componentes principales. Usamos una librería llamada MDP que realiza PCA mediante la función `mdp.pca()`.

```
# -*- coding: utf-8 -*-

import numpy
import mdp
import scipy.misc as sm
import os
import sys, optparse


class ErrorLista(Exception):
        def __init__(self, value):
                self.value = value
        def __str__(self):
                return repr(self.value)

def main ():
        parser = optparse.OptionParser(usage="uso: %prog [options]")
        parser.add_option("-o", "--salida",
        help="Nombre del archivo imagen en el que se guardará el resultado.",
        default='imagen_final_pca.jpg')
        (options, args) = parser.parse_args()

        if len(args)==0:
                raise ErrorLista('Ingresar al menos una imagen.')
        l = []
```

```
        for i in args:
                l.append(sm.imread(i))
        if len(l)==1:
                t = l[0]
        else:
                t = numpy.concatenate((l[0],l[1]),axis=2)
                if len(l[2:])>0:
                        for i in l[2:]:
                                t = numpy.concatenate((t,i),axis=2)

        fil = t.shape[0]
        col = t.shape[1]
        dim = t.shape[2]
        p = numpy.reshape (t, (t.shape[0]*t.shape[1],t.shape[2]))

        #Transormamos en float cada elemento de p:
        p = p.astype (numpy.float32)

        #Realizamos pca en p:
        y = mdp.pca (p)

        #Generamos la imagen resultante a partir de p:
        if dim > 3:
                y = y[:,0:3]
        dim = 3
        imagen_final= numpy.reshape (y, (fil,col,dim))
        sm.imsave(options.salida, imagen_final)
        print '--------------\n PCA finalizado.'
        print 'Imagen resultado guardada en: '+ options.salida
        print '\n--------------'


if __name__ == "__main__":
        main()
```

A.3. Ejemplo de normal multivariada.

Este sencillo código python genera una muestra de una normal multivariada y la grafica.

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.axis
import mdp



def main ():
```

```
        mean = [0, 0]
        cov = [[2,15],[100,80]]
        x, y = np.random.multivariate_normal (mean, cov, 500).T

        x_t=np.transpose (x)
        y_t=np.transpose (y)
        d = np.array ([x_t,y_t])
        d = d.T

        y = mdp.pca (d)
        print y.shape
        f2 = plt.figure()
        ax2 = f2.add_subplot(111)
        ax2.plot (x,y,'x');
        ax2.axis ('equal');
        plt.show()

if __name__ == "__main__":
  main()
```

A.4. Versión rudimentaria de PCA.

El siguiente código es la versión del algoritmo de componentes principales usado para explicar su funcionamiento paso a paso en la sección (2.3.4).

```
# -*- coding: utf-8 -*-

import numpy
import scipy
import os
import scipy.misc as sm

def media_matriz (m):
        media = numpy.zeros(m.shape[1])
        for i in range(0,m.shape[1]):
                media[i]= sum(m[0:m.shape[0],i])/m.shape[0]
        return media

def extraerMedia (m,medias):
        filas = m.shape[0]
        col = m.shape[1]
        res = numpy.zeros([filas, col])
        for i in range(0,col):
                p1 = m[0:filas,i]
                p1 = p1-medias[i]
                res[0:filas,i] = p1
        return res


def pca (p):
```

```
#x= numpy.array ([[.69,-1.31,.39,.09,1.29,.49,.19,-.81,-.31,-.71]
#,[.49,-1.21,.99,.29,1.09,.79,-.31,-.81,-.31,-1.01]])
fil = p.shape[0]
col = p.shape[1]
rgb = p.shape[2]
x = numpy.reshape (p, (p.shape[0]*p.shape[1],p.shape[2]))

print "x:\n\n", x
medias = media_matriz(x)
print "\n medias: \n", medias
x0 = extraerMedia(x,medias)
#x0 =x0.transpose()
print "matriz de entrada:\n"
print x0
#print (x)
t = x0.shape
a = numpy.dot (x0.transpose(), x0)/(t[0]-1)
print "-----------------\nmatriz de covariaza:\n", a
u, s, v = numpy.linalg.svd (a)
print "-----------------\nu:\n",u
print "-----------------\nv:\n", v
print "-----------------\ns:\n", s
traza = numpy.trace (a)
i=0
suma=0
t=0
print "-----------------\nElegimos componentes:\n-----------------\n"
while (t<= 0.95):
        suma = (suma+s[i])
        t = suma/traza
        i=i+1
        cantidad_componentes=i
print "Acumulada:\n", t
#Seleccionamos las columnas que corresponden a los autovetores
#que sirven;

#media = sum(sum(x0))/(x0.shape[0]*x0.shape[1])
u_tilde=u[0:u.shape[0],0:i]
print "u_tilde:\n", u_tilde
a_tilde= numpy.dot(u_tilde ,numpy.dot(numpy.diag(s[0:i]), v[0:i]))
print "a_tilde:\n", a_tilde

x_tilde = numpy.dot(numpy.dot(x0,u_tilde), u_tilde.transpose())
aux=numpy.dot(numpy.dot(x0,u_tilde), u_tilde.transpose())
#Le agregamos la media:
for i in range(0,x_tilde.shape[1]):
        x_tilde[0:x0.shape[0],i] = x_tilde[0:x0.shape[0],i] + medias[i]

print "\n**********\n", x_tilde.shape, "\n**********\n"
print "\n********************\n Cantidad de componentes: ",
```

```
        cantidad_componentes, "\n******************\n"
        imagen_final= numpy.reshape (x_tilde, (fil,col,rgb))

        print "\nDatos finales:\n", x_tilde
        print "\nMatriz para la imagen final: \n", imagen_final
        print imagen_final.shape
        print x_tilde.shape


        #scipy.misc.imsave('imagen_final2.jpg', imagen_final)

def main ():

        t = sm.imread('asuncion.png')
        t = numpy.concatenate((t,sm.imread('asuncion2.png')),axis=2)

        pca(t)



if __name__ == "__main__":
        main()
```

Referencias

[1] Oscar H. Bustos, Alejandro C. Frery, Mario A. Lamfri, and Carlos M. Scavuzzo. Técnicas estadísticas en teledetección espacial. Technical report, Fa.M.A.F., Universidad Nacional de Córdoba., 2004.

[2] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. *Proceedings of the 21st International Conference on Machine Learning*, 2004.

[3] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Prentice Hall, 3rd edition, 2008.

[4] David R. Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis; an overview with application to learning methods. Technical report, University of London, 2003.

[5] J. R. Jensen. *Introductory digital image processing: a remote sensing perspective*. Pearson Prentice Hall, 2005.

[6] I. T. Jolliffe. *Principal Component Analysis*. Springer, 2002.

[7] M. Anji Reddy. *Remote Sensing and Geographical Information Systems*. BS Publications, 2008.

[8] David Weenink. Canonical correlation analysis. *Proceedings of Institute of Phonetic Sciences, University of Amsterdam*, 2003.