Universidad de Valladolid
Facultad de Ciencias

Trabajo Fin de Grado

Grado en Matemáticas

Factorización No Negativa de Matrices

Autora: Ana Isabel Rodríguez Vega

Tutor: Eustasio del Barrio Tellado

Índice general

Introducción

En este trabajo estudiaremos varios métodos que pueden ser vistos como la factorización aproximada de una matriz de datos sujeta a diferentes restricciones.

Esta idea se puede aplicar al análisis estadístico de datos multivariantes de la siguiente manera: Dado un conjunto de vectores p-dimensionales, estos se colocan en las filas de una matriz X de dimensiones $n \times p$, donde n es el número de observaciones en el conjunto de datos. Entonces, esta matriz se factoriza en una matriz W de dimensiones $n \times r$ y una matriz H de dimensiones $r \times p$. Por tanto, el objetivo que perseguimos es:

$$X \approx WH,$$

donde "$\approx$" indica que buscamos que X y WH estén próximas en cierto sentido, que se especificará más adelante.

Normalmente, r se elige inferior a n y p para que W y H sean más pequeñas que la matriz original. De hecho, lo que pretendemos es reducir esa dimensión para encontrar una factorización aproximada $X \approx W_r H_r$. Esto tiene como resultado una versión comprimida de la matriz de datos original.

Los métodos estudiados están asociados a diferentes criterios de optimalidad (maximización de la varianza, minimización de diferentes medidas de discrepancia entre las matrices X y WH, etc.) con diferentes restricciones. En ocasiones, se busca comprimir al máximo manteniendo el mínimo error de reconstrucción, pero otras veces se intenta resumir la información mediante representantes *naturales*. Además, a lo largo de toda la memoria hablaremos de H como la matriz que contiene los *prototipos*, que son los representantes o vectores base en los que se resumirá la información. En W tendremos los *pesos* o coeficientes para las combinaciones a partir de las cuales podremos reconstruir los datos de X.

El más antiguo y más conocido de estos métodos es el Análisis en Componentes Principales (ACP), que se introdujo en 1901 gracias a Pearson y se desarrolló independientemente por Hotelling en 1933 (ver [7]). Como muchas

otras técnicas de análisis multivariante, su uso no se extendió hasta la aparición y el desarrollo de los ordenadores. Como veremos a lo largo de esta memoria, en el ACP logramos obtener todas las factorizaciones en cualquier dimensión a partir de un problema de diagonalización y, además, sus aproximaciones son las de mayor calidad.

Sin embargo, más recientemente, se han buscado métodos que nos permitan conseguir unos prototipos más interpretables. En este contexto aparecen técnicas como el método de k-medias (donde tenemos prototipos representativos) o el Análisis de Arquetipos (en el que los prototipos son una combinación de los datos y los datos, una combinación convexa de los prototipos).

Esta última idea tendrá un mal comportamiento sobre conjuntos de datos de imágenes, por ejemplo, ya que estos suelen ser demasiado grandes. Por ese motivo, se han buscado métodos alternativos más sencillos de aplicar a conjuntos de datos de este estilo.

En los años noventa se propuso la Factorización No Negativa de Matrices (NMF) en el artículo [10] publicado en Nature. Con este método se busca mejorar la interpretabilidad de los prototipos, especialmente en el caso concreto de la compresión de imágenes. En el artículo anteriormente citado se exponen sus ventajas y, para ello, se compara con otros a través de un ejemplo aplicado al análisis de un conjunto de fotografías. En esta memoria, seguiremos un esquema similar.

El primer capítulo se centrará en el conocido método de ACP. En él, explicaremos la relación de las componentes principales con el problema que estamos tratando de solucionar, cuál es el criterio que optimizaremos y cómo podemos solucionarlo. También estudiaremos cuál es su relación con la descomposición en valores singulares y para qué nos puede servir a la hora de calcular la factorización $X \approx WH$.

En el segundo capítulo se analizará un método llamado Cuantización Vectorial (VQ), estrechamente relacionado con el análisis cluster y las k-medias. Estudiaremos el problema de minimización que se plantea y el algoritmo más conocido para su resolución. También veremos cómo podemos aplicarlo a la factorización de matrices y, más concretamente, a la compresión de información de imágenes. Finalmente, expondremos brevemente algunas nociones del método conocido como análisis de arquetipos, que está ganando popularidad en los últimos años, y veremos cuál es su relación con el resto de métodos analizados en este trabajo.

Por último, el tercer capítulo estará dedicado al estudio del NMF. Dedicaremos una sección a discutir cuál es la motivación para su uso y, a continuación, explicaremos más detalladamente cuál es el problema que se plantea y qué algoritmos podemos utilizar para resolverlo. Finalmente, haremos una crítica respecto a la unicidad de la factorización en este caso.

En todos los capítulos se incluirá una sección de ejemplos, donde se expondrán los resultados y conclusiones obtenidos a partir del análisis de una base de datos de imágenes faciales. Todos los cálculos han sido realizados en R.

Al final se encontrarán unos breves comentarios como conclusión general, las referencias utilizadas y un apéndice donde se muestra un resumen de los códigos de R usados para las secciones de ejemplos.

Capítulo 1

Análisis en Componentes Principales

La idea central del Análisis en Componentes Principales (ACP) es reducir las dimensiones de un conjunto de datos con un gran número de variables, reteniendo la mayor cantidad posible de su información. Esto se consigue haciendo una transformación a un nuevo conjunto de variables, llamadas *componentes principales*, que están incorreladas y ordenadas de forma que las primeras retienen la mayor parte de la variabilidad presente en todas las variables originales. De esta forma, nos sirven para obtener las mejores aproximaciones a los datos en dimensiones menores, permitiendo una interpretación más sencilla del conjunto de partida sin perder la información esencial.

Esta idea puede ser vista como la búsqueda de un subespacio de menor dimensión sobre el que podamos proyectar los datos con la menor pérdida de información. Para lograr este objetivo, se puede intentar encontrar el subespacio que recoja la mayor cantidad de varianza o bien el subespacio que minimice el error de reconstrucción. Dicho subespacio será el que identificaremos con la matriz H de la descomposición $X \approx WH$ buscada y, por lo tanto, el criterio que emplearemos en este caso será el de la maximización de la variabilidad recogida o la minimización del error de reconstrucción. Primero, nos centraremos en el primer punto de vista y, finalmente, veremos que ambas aproximaciones son equivalentes.

También veremos una forma sencilla de calcular la calidad de la aproximación en una dimensión menor y la relación de este método con la descomposición de valores singulares. Finalmente, aplicaremos ACP al conjunto de datos que vamos a utilizar como ejemplo a lo largo del trabajo para poder observar más fácilmente todas las ideas expuestas en este capítulo.

La búsqueda de las componentes principales es una idea con un uso muy

extendido y se pueden encontrar múltiples referencias al respecto. Para este capítulo se ha tratado de hacer una síntesis basada en los libros [7] y [12] para dar una justificación al uso de este método a la hora de resolver el problema de factorización matricial que estamos analizando.

1.1. Definiciones y propiedades principales

Llamaremos X a la matriz del conjunto de datos, con $X \in \mathcal{M}_{n\times p}(\mathbb{R})$, donde n es el número de casos y p el número de variables, y representaremos sus filas como

$$X = \begin{bmatrix} x_1^T \\ \vdots \\ x_n^T \end{bmatrix},$$

con $x_i \in \mathbb{R}^p, \quad \forall i = 1, ..., n.$

De esta forma, cada uno de los vectores x_i se puede entender como la representación de los datos del caso i en cada una de las p variables.

1.1.1. Reconstrucciones de baja dimensión

A lo largo de esta sección nos centraremos en la búsqueda de un subespacio de menor dimensión que recoja la mayor cantidad de varianza. Esta idea tiene sentido puesto que, si proyectamos sobre un subespacio donde haya mayor variabilidad, tendremos más información y diferencia en la estructura de los datos. Veamos un ejemplo sencillo para ilustrar la situación.

Ejemplo 1.1.1. Sea X la matriz de datos definida de la siguiente manera:

$$X = \begin{bmatrix} x_1 & 0 \\ \vdots & \vdots \\ x_n & 0 \end{bmatrix},$$

donde cada fila consiste en un vector cuya primera coordenada es $x_i \in [-1, 1]$ y su segunda coordenada es cero. Proyectando estos datos sobre los subespacios determinados por los ejes de coordenadas en la base canónica, obtenemos la representación expuesta en la Figura 1.1. Las proyecciones de los puntos sobre el eje horizontal están dibujadas en azul y las correspondientes al eje vertical, en rojo. En este ejemplo se ve claramente que el subespacio que explica mejor la estructura de los datos coincide con el que recoge mayor variabilidad. Es decir, el eje horizontal.

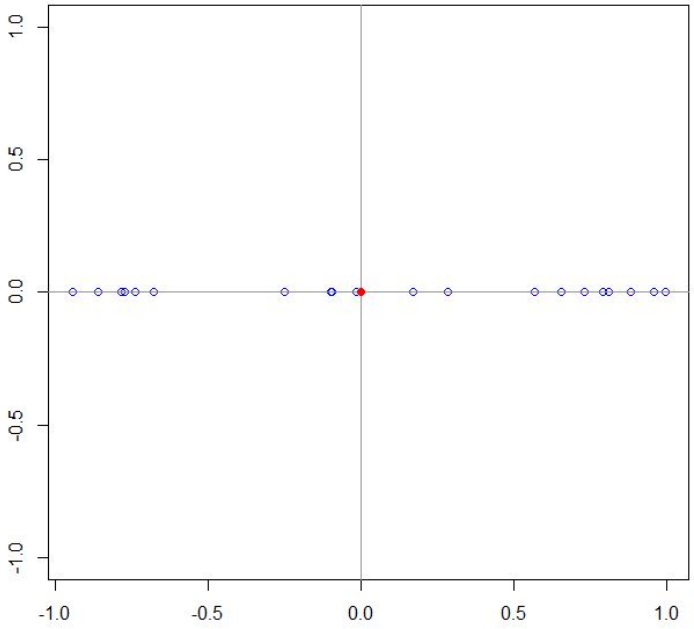

Figura 1.1: Ejemplo

Empezaremos concretando algunas de estas ideas y definiendo los elementos que vamos a manejar de ahora en adelante.

Definición 1.1.2. Dada una muestra de tamaño $n, x_1, x_2, \cdots, x_n$, se define la varianza muestral como el promedio del cuadrado de las distancias a su media

$$Var(x_1, \cdots, x_n) = \frac{1}{n} \sum_{i=1}^{n} \|x_i - \bar{x}\|^2.$$

Sea X la matriz descrita anteriormente, por analogía al caso unidimensional, generalmente escribiremos $Var(X)$ donde $x_1, \ldots x_n$ son las filas de X.

Podemos suponer que estamos trabajando con una matriz centrada (si no fuese el caso, restaríamos la media a sus columnas), es decir, que tenemos que

$$\frac{1}{n} \sum_{i=1}^{n} x_i = 0.$$

Equivalentemente, $\mathbf{1}^T X = 0$, donde $\mathbf{1}^T = [1, \cdots, 1]$. Por tanto, en ese caso, al estar ante vectores centrados, su varianza se puede hallar como la suma

de los cuadrados de sus componentes. Esto es:

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n} \|x_i\|^2 = \frac{1}{n} Tr(\Sigma).$$

En la ecuación anterior $\Sigma = X^T X = \sum_{i=1}^{n} x_i x_i^T$ es la matriz de covarianzas de X, con $\Sigma \in \mathcal{M}_{p\times p}(\mathbb{R})$. Mantendremos esta notación para Σ a lo largo del trabajo.

El problema de las componentes principales es encontrar el subespacio lineal H de $\mathbb{R}^p$ con $dim(H) = r$ que recoja la mayor variabilidad de los datos. Entonces, sea H el subespacio generado por $h_1, ..., h_r \in \mathbb{R}^p$ (un conjunto de vectores linealmente independientes), también denotaremos por H, con un ligero abuso de notación, a la matriz formada por $H = [h_1, \cdots, h_r]$, con $H \in \mathcal{M}_{p\times r}(\mathbb{R})$.

De forma más concreta, lo que vamos a hacer es fijarnos en la proyección de los puntos sobre el subespacio H y medir la varianza. Recordemos que la proyección de un punto x sobre un subespacio lineal H es el elemento de H más próximo a x. Es un hecho bien conocido que la proyección es un operador lineal al que denotaremos Pr_H. La matriz asociada es $H(H^T H)^{-1} H^T$, es decir,

$$Pr_H(x) = H(H^T H)^{-1} H^T x. \tag{1.1}$$

Probaremos a continuación un resultado que nos permite calcular de forma simple la variabilidad de las proyecciones de los datos originales sobre el subespacio H. Denotaremos por $Var_H(X)$ a la varianza recogida por H de un conjunto de datos X, es decir, $Var_H(X)$ es la varianza asociada a los puntos $Pr_H(x_1), \ldots, Pr_H(x_n)$. De esta forma, en el caso $Var_{\mathbb{R}^n}(X) = Var(X)$ mantenemos la notación original.

Con esta notación, podemos formular el problema expuesto anteriormente de la siguiente manera: Buscamos el subespacio H de dimensión r tal que $Var_H(X) \geq Var_{\tilde{H}}(X)$, $\forall \tilde{H}$ subespacio lineal de $\mathbb{R}^p$ con $dim(\tilde{H}) = r$.

Lema 1.1.3. Sea H un subespacio lineal de $\mathbb{R}^n$ y Pr_H la proyección sobre H. La varianza recogida por H de unos datos centrados es:

$$Var_H(X) = \sum_{i=1}^{n} \|Pr_H(x_i)\|^2 = Tr(H(H^T H)^{-1} H^T \Sigma) \tag{1.2}$$

Demostración. Por un lado, tenemos que

$$Var_H(X) = \frac{1}{n}\sum_{i=1}^{n} \|Pr_H(x_i) - \bar{y}\|^2,$$

donde $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} Pr_H(x_i)$. Luego, si demostramos que $\bar{y} = 0$, habremos probado la primera igualdad.

Ver que $\bar{y}$ es igual a cero es equivalente a mostrar que $\sum_{i=1}^{n} Pr_H(x_i) = 0$. Y, dado que Pr_H es un operador lineal, tenemos que

$$\sum_{i=1}^{n} Pr_H(x_i) = Pr_H(\sum_{i=1}^{n} x_i),$$

que es igual a cero, puesto que estamos manejando datos centrados.

Veamos ahora la segunda igualdad del lema. Teniendo en cuenta que $\|x\|^2 = x^T x$, entonces

$$\|Pr_H(x)\|^2 = (x^T H(H^T H)^{-1} H^T)(H(H^T H)^{-1} H^T x) = x^T H(H^T H)^{-1} H^T x.$$

Por otro lado, utilizando la propiedad cíclica de la traza, tenemos que

$$Tr(x^T H(H^T H)^{-1} H^T x) = Tr(H(H^T H)^{-1} H^T x x^T).$$

Como la traza también es un operador lineal (la suma de las trazas es la traza de la suma), acabamos con las siguientes igualdades.

$$Var_H(X) = Tr(H(H^T H)^{-1} H^T \cdot \sum_{i=1}^{n} x_i \cdot x_i^T) = Tr(H(H^T H)^{-1} H^T \Sigma).$$

□

Este lema nos proporciona un camino para tratar de resolver el problema de las componentes principales.

La matriz Σ se puede diagonalizar en una base ortonormal y sus autovalores son positivos, puesto que Σ es simétrica y definida positiva. Por eso, existen $v_1, ..., v_p \in \mathbb{R}^p$ tales que $\|v_j\| = 1$ y $v_i \cdot v_j = \delta_{i,j}$, que cumplen la igualdad

$$\Sigma v_j = \lambda_j v_j, \qquad j = 1, \ldots, p \tag{1.3}$$

y con λ_j que suponemos ordenados de forma que $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p \geq 0$.

Sea $V = [v_1, \cdots, v_p]$, tenemos que esta matriz es ortonormal. Es decir, que cumple que

$$V^T V = Id = V V^T \Rightarrow V^T = V^{-1}.$$

Entonces, con $\Lambda = diag\{\lambda_1, \cdots, \lambda_p\}$, podemos escribir (1.3) en forma matricial de la siguiente manera:

$$\Sigma \cdot V = [\lambda_1 v_1, \cdots, \lambda_p v_p] = V \cdot \Lambda,$$

o, equivalentemente,

$$\Sigma = V\Lambda V^T = [\lambda_1 v_1, \cdots, \lambda_p v_p] \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix} = \sum_{j=1}^{p} \lambda_j v_j v_j^T.$$

Llevando esto último a (1.2) y utilizando de nuevo las propiedades de la traza,

$$\begin{aligned} Var_H(X) &= \sum_{j=1}^{p} Tr(H(H^TH)^{-1}H^T\lambda_j v_j v_j^T) \\ &= \sum_{j=1}^{p} \lambda_j Tr(v_j^T H(H^TH)^{-1}H^T v_j) \\ &= \sum_{j=1}^{p} \lambda_j \|H(H^TH)^{-1}H^T v_j\|^2. \end{aligned}$$

Por otro lado, como $\|x\|^2 = \|Pr_H(x)\|^2 + \|Pr_{H^\perp}(x)\|^2$, tenemos que $\|Pr_H(x)\|^2 \leq \|x\|^2$ y esto implica que

$$\|H(H^TH)^{-1}H^T v_j\|^2 \leq \|v_j\|^2 = 1.$$

La igualdad se da si y sólo si $v_j \in H$, puesto que, en ese caso, $Pr_H(v_j) = v_j$.

Ahora estamos en condiciones de probar que el subespacio H_r, generado por los r primeros vectores (es decir, a los asociados a los mayores autovalores), es la solución al problema de las componentes principales. Para ello, razonaremos de la siguiente forma.

Como tenemos que $H_r = \langle v_1, ..., v_r \rangle$, entonces $dim(H_r) = r$ y $Var_{H_r}(X) = \sum_{j=1}^{r} \lambda_j$, puesto que la proyección es v_j si $j \leq r$ y es 0 si $j \geq r$. Demostraremos a continuación que $Var_H(X) \leq \sum_{j=1}^{r} \lambda_j$ para cualquier otro H tal que $dim(H) = r$.

Suponiendo que $H = [h_1, ..., h_r]$, con $h_i^T \cdot h_j = \delta_{i,j}$, usaremos entonces que $H^TH = Id_r$.

$$\begin{aligned}\sum_{j=1}^{p} \|H(H^TH)^{-1}H^Tv_j\|^2 &= \sum_{j=1}^{p} Tr(v_j^TH(H^TH)^{-1}H^Tv_j)\\ &= \sum_{j=1}^{p} Tr(H(H^TH)^{-1}H^Tv_jv_j^T)\\ &= Tr(H(H^TH)^{-1}H^T\sum_{j=1}^{p} v_jv_j^T)\\ &= Tr(H(H^TH)^{-1}H^TId_p)\\ &= Tr(H(H^TH)^{-1}H^T)\\ &= Tr((H^TH)^{-1}H^TH)\\ &= Tr(Id_r)\\ &= r.\end{aligned}$$

Entonces, si $dim(H) = r$ y denotando $w_j = \|H(H^TH)^{-1}H^Tv_j\|^2$, tenemos que $Var_H(X) = \sum_{j=1}^{p} w_j\lambda_j$ con $0 \leq w_j \leq 1$ y $\sum_{j=1}^{p} w_j = r$.

Como los λ_j están ordenados de mayor a menor, la cantidad $Var_H(X) = \sum_{j=1}^{p} w_j\lambda_j$ alcanza su máximo valor si hacemos $w_j = 1$ para $1 \leq j \leq r$ y $w_j = 0$ para el resto (es decir, si damos el mayor peso posible a los primeros). Por tanto, $Var_H(X) \leq \sum_{j=1}^{r} \lambda_j = Var_{H_r}(X)$. Luego el subespacio H_r, generado por los r autovectores asociados a los r mayores autovalores, es el que recoge mayor varianza entre todos los subespacios de dimensión r.

Resumimos todas estas conclusiones en el siguiente resultado.

Teorema 1.1.4. Sea $X \in \mathcal{M}_{n\times p}(\mathbb{R})$ una matriz de datos centrada y $\mathcal{H}_r$ el conjunto de subespacios $H \subset \mathbb{R}^n$ tales que $dim(H) = r$. Entonces,

$$\max_{H\in\mathcal{H}_r} Var_H(X) = Var_{H_r}(X) = \sum_{i=1}^{r} \lambda_i,$$

donde H_r es el subespacio de $\mathcal{H}_r$ generado por los r primeros autovectores de la matriz $\Sigma = X^TX$ y λ_i, con $i = 1, \ldots, r$, son sus mayores autovalores (asociados a esos autovectores).

El subespacio H_r del teorema 1.1.4 será el que, con un ligero abuso de notación, identificaremos con la matriz de prototipos H_r en la factorización aproximada

$$X \approx W_rH_r.$$

Calidad de la reconstrucción:

El ACP destaca por la facilidad a la hora de calcular la calidad de la reconstrucción en una dimensión menor o, lo que es lo mismo, la proporción de varianza recogida por el subespacio H_r sobre el que proyectamos los datos.

A partir del razonamiento descrito en este apartado se deduce que la proporción de varianza recogida por el subespacio H_r que hemos construido anteriormente es

$$\frac{\sum_{j=1}^{r} \lambda_j}{\sum_{j=1}^{p} \lambda_j} \in (0,1),$$

puesto que la varianza total es la suma de todos los autovalores.

1.1.2. Interpretaciones alternativas

Otra forma natural de enfrentarse al problema es observar el error que se comete al aproximar los datos por sus proyecciones sobre un subespacio de menor dimensión (error de reconstrucción). Si se utiliza la norma al cuadrado como forma de medida, esto conduce al siguiente criterio a minimizar:

$$\sum_{i=1}^{n} \|x_i - Pr_H(x_i)\|^2. \tag{1.4}$$

Probaremos a continuación que ambos criterios son equivalentes.

Lema 1.1.5. Buscar el subespacio H de dimensión r que minimiza el error de reconstrucción (1.4) equivale a buscar el H de dimensión r que recoge la mayor cantidad de varianza.

Demostración. Por un lado tenemos que

$$\|x_i\|^2 = \|Pr_H(x_i)\|^2 + \|(Id - Pr_H)(x_i)\|^2.$$

Y además

$$\|(Id - Pr_H)(x_i)\|^2 = \|x_i - Pr_H(x_i)\|^2.$$

Luego

$$\|x_i - Pr_H(x_i)\|^2 = \|x_i\|^2 - \|Pr_H(x_i)\|^2.$$

Entonces

$$\sum_{i=1}^{n} \|x_i - Pr_H(x_i)\|^2 = \sum_{i=1}^{n} \|x_i\|^2 - \sum_{i=1}^{n} \|Pr_H(x_i)\|^2.$$

Minimizar el término del lado izquierdo de esa ecuación equivale a maximizar $\sum_{i=1}^{n} \|Pr_H(x_i)\|^2$, que es lo mismo que hemos hecho para buscar el subespacio que recoge la mayor cantidad de varianza. □

Esta equivalencia dada por el lema 1.1.5 garantiza que el error relativo de aproximación también se calcula fácilmente en términos de los autovalores. Más concretamente,

$$\begin{aligned}\frac{\sum_{i=1}^{n}\|x_i - Pr_H(x_i)\|^2}{\sum_{i=1}^{n}\|x_i\|^2} &= 1 - \frac{\sum_{i=1}^{n}\|Pr_H(x_i)\|^2}{\sum_{i=1}^{n}\|x_i\|^2} \\ &= 1 - \frac{\sum_{j=1}^{r}\lambda_j}{\sum_{j=1}^{p}\lambda_j}\end{aligned}$$

1.2. Descomposición en Valores Singulares

El ACP está muy relacionado con la descomposición en valores singulares (SVD). Esta descomposición consiste en la factorización de una matriz rectangular en una matriz diagonal y dos matrices formadas por vectores ortonormales. El resultado fundamental relacionado con esta descomposición es el siguiente.

Teorema 1.2.1. Dada una matriz X de dimensión $n \times p$ (n observaciones y p variables), X se puede escribir

$$X = UDV^T \tag{1.5}$$

donde:
(i) U, V son matrices de dimensión $n \times r$ y $p \times r$ respectivamente, con columnas ortonormales de forma que $U^TU = Id_r$ y $V^TV = Id_r$.
(ii) D es una matriz diagonal de dimensión $r \times r$.
(iii) r es el rango de X.

Demostración. Este resultado puede probarse a partir de la descomposición espectral de Σ. Como ya hemos utilizado en la sección anterior, se puede escribir $\Sigma = V\Lambda V^T$, con Λ diagonal y V ortonormal. Desarrollando el producto matricial de la parte derecha de esa ecuación tenemos

$$\Sigma = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \cdots + \lambda_p v_p v_p^T = \sum_{j=1}^{p} \lambda_j v_j v_j^T. \tag{1.6}$$

En este caso, hay que tener en cuenta que los últimos $p - r$ términos de (1.6) son cero, puesto que los últimos $p - r$ autovalores son cero si X, y por tanto Σ, tiene rango r. Entonces

$$\Sigma = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + \cdots + \lambda_r v_r v_r^T.$$

Definimos V como la matriz de dimensiones $p \times r$ cuya k-ésima columna es v_k, U como la matriz de dimensiones $n \times r$ cuya k-ésima columna es

$$u_k = \lambda_k^{-1/2} X v_k, \quad k = 1, 2, ..., r,$$

y D como la matriz diagonal de dimensiones $r \times r$ cuyo k-ésimo elemento diagonal es $\lambda_k^{1/2}$. V es una matriz de columnas ortonormales, puesto que los vectores v_k son ortonormales. U también lo es, dado que

$$\begin{aligned} u_i^T u_j &= (\lambda_i^{-1/2} X v_i)^T (\lambda_j^{-1/2} X v_j) \\ &= \lambda_i^{-1/2} \lambda_j^{-1/2} v_i^T X^T X v_j \\ &= \lambda_i^{-1/2} \lambda_j^{-1/2} v_i^T \lambda_j v_j \\ &= \lambda_i^{-1/2} \lambda_j^{1/2} v_i^T v_j. \end{aligned}$$

puesto que v_j es autovector de $X^T X$. Y, entonces, tenemos que $u_i^T u_j = \delta_{i,j} = v_i^T v_j, \forall i, j = \{1, ..., r\}$.

De esta forma, U, D, V satisfacen las condiciones (i) y (ii) anteriores, y sólo queda probar que $X = UDV^T$.

$$\begin{aligned} UDV^T &= U \begin{bmatrix} \lambda_1^{1/2} v_1^T \\ \lambda_2^{1/2} v_2^T \\ \vdots \\ \lambda_r^{1/2} v_r^T \end{bmatrix} \\ &= \sum_{k=1}^{r} \lambda_k^{-1/2} X v_k \lambda_k^{1/2} v_k^T = \sum_{k=1}^{r} X v_k v_k^T \\ &= \sum_{k=1}^{p} X v_k v_k^T, \end{aligned}$$

porque $v_k, k = r+1, r+2, ..., p$ son los autovectores de Σ correspondientes a los autovalores nulos, por lo que

$$X^T X v_k = 0 \Rightarrow v_k^T X^T X v_k = 0 \Rightarrow \|Xv\|^2 = 0 \Rightarrow \|X v_k\| = 0$$

y de ahí que $X v_k = 0$, para $k = r+1, r+2, ..., p$. Luego

$$UDV^T = X \sum_{k=1}^{p} v_k v_k^T = X$$

como queríamos, pues esta última igualdad viene dada por la ortonormalidad de los vectores v_k. □

Existen procedimientos computacionalmente eficientes para calcular el SVD y, por tanto, estos métodos nos ayudan a encontrar las componentes principales. Si podemos encontrar U, D y V que satisfacen (1.5), entonces V y D nos dan los autovectores y las raíces cuadradas de los autovalores de $\Sigma = X^T X$. Además, en U obtenemos las versiones estandarizadas de los pesos de las componentes principales.

De esta forma, obtenemos la descomposición $X = WH$ buscada, donde $W = UD$ y $H = V^T$.

1.3. Algoritmos de cálculo

Se han realizado una serie de simulaciones en R para tratar de comparar dos algoritmos que nos sirven para hallar las componentes principales de un conjunto de datos y, de esta forma, poder justificar el estudio del método de descomposición en valores singulares en este capítulo.

En ambos algoritmos partimos de la matriz de datos X de dimensiones $n \times p$ y operamos con ella hasta hallar sus componentes principales. En el primer caso, que llamaremos *diagonalización*, esto se hace construyendo $\Sigma = X^T X$ y calculando sus autovalores y autovectores con ayuda de la función *eigen*. En el segundo caso, utilizamos la función *svd* para calcular las matrices U, D y V que, como se ha visto anteriormente, contienen toda la información necesaria para el ACP.

En las siguientes tablas se muestran los tiempos medios de simulación obtenidos con los dos métodos dependiendo del número de filas y columnas de la matriz X (expresados en segundos). Las situaciones en las que no ha sido posible realizar el cálculo por falta de espacio a la hora de almacenar la información están representadas con una "x".

Tiempos de simulación para el método de diagonalización:

n \ p	10	50	100	500	1000	5000	10000
10	0.0011	0.0006	0.0022	0.1677	1.1632	131.9474	x
50	0.0002	0.0005	0.0019	0.1915	1.0664	131.4391	x
100	0.0003	0.0009	0.0024	0.1442	1.0829	132.0075	x
500	0.0003	0.0012	0.0041	0.1968	1.2469	137.5407	x
1000	0.0002	0.0015	0.0062	0.2454	1.5215	144.5014	x
5000	0.0006	0.0059	0.0247	0.7891	3.6881	203.8083	x
10000	0.0008	0.0112	0.0515	1.4596	6.3114	271.8141	x

Tiempos de simulación para el método de valores singulares:

n \ p	10	50	100	500	1000	5000	10000
10	0.0005	0.0001	0.0000	0.0001	0.0005	0.0016	0.0037
50	0.0000	0.0011	0.0012	0.0063	0.0062	0.0229	0.0446
100	0.0000	0.0011	0.0037	0.0113	0.0204	0.0837	0.1716
500	0.0002	0.0031	0.0127	0.3233	0.6292	2.3026	4.5616
1000	0.0004	0.0057	0.0221	0.7365	2.4612	9.8274	17.6847
5000	0.0012	0.0263	0.1031	2.8569	12.2852	332.8090	683.1549
10000	0.0028	0.0519	0.2062	5.5109	22.6352	780.7732	x

La diferencia más notable es la simetría que se puede observar en la segunda. En el caso del método SVD, vemos que se obtienen tiempos similares intercambiando los valores de n y p. Es decir, no hay una gran diferencia entre hacer el cálculo para una matriz con un gran número de filas y pocas columnas o hacerlo para su traspuesta. Sin embargo, no ocurre lo mismo con el primer método, donde se ve claramente que los tiempos son mucho mayores en los casos en los que $p \gg n$.

Ese hecho también está relacionado con las ocasiones en las que no se ha podido calcular por falta de espacio para almacenar los datos. Mientras que el método de diagonalización ha dado errores siempre que $p = 10000$, con el SVD se ha podido obtener la descomposición en todos los casos, salvo para la matriz de dimensiones 10000×10000.

Estos resultados no son sorprendentes desde el punto de vista teórico, puesto que el SVD realiza el cálculo operando sobre la matriz X, pero para la diagonalización se debe construir $\Sigma = X^T X$. De esta forma, finalmente se trabaja con datos de dimensión $p \times p$, lo que hace que el número de columnas sea un dato muy importante.

También se han realizado unas simulaciones para comprobar cómo afectan las pequeñas perturbaciones a ambos métodos. Para ello, se han creado matrices aleatorias a partir de unos autovalores conocidos y se les han sumado unas matrices de perturbación. En la siguiente tabla se representan las distancias medias entre los autovalores de las matrices perturbadas y los autovalores originales según ambos métodos, utilizando la distancia L1 y la distancia euclídea.

	Diagonalización	SVD
L1	0.01148573	0.006432907
Euclídea	0.01065644	0.005338551

A partir de los resultados obtenidos se puede ver que los datos se ven afectados el doble por el método de diagonalización que con el método SVD.

Por estos motivos está justificado el interés en el SVD a la hora de realizar un análisis en componentes principales.

1.4. Ejemplo

El siguiente ejemplo está basado en la base de datos faciales de Yale, utilizada en el artículo [1] y que se puede encontrar entre las diversas bases de datos que se pueden descargar de www.face-rec.org/databases. Se trata de un conjunto de datos que contiene la información de $n = 165$ imágenes faciales en escala de grises, cada una compuesta por $p = 243 \times 320$ píxeles, correspondientes a 15 individuos fotografiados en 11 condiciones diferentes de iluminación, expresión y gafas. De esta forma, manejaremos una matriz X de dimensiones $n \times p$, donde cada fila se corresponde con una imagen y cada columna representa los valores que toman sus píxeles. Esta situación se repetirá a lo largo del trabajo, pues aplicaremos los distintos métodos a la misma matriz para poder comparar los resultados al final.

La mayor parte de los lenguajes incluyen un gran conjunto de utilidades para el tratamiento de imágenes en diferentes formatos. Para el análisis en este trabajo se han utilizado las librerías png y pixmap. En la primera se puede encontrar la función *readPNG*, que permite leer ficheros de imágenes y convertirlas en matrices cuyos elementos se corresponden con las distintas intensidades de gris. En *pixmap* se encuentra la función *pixmapGrey*, con la que podemos representar imágenes a partir de matrices como las descritas anteriormente. En las secciones de ejemplos, comprobaremos de forma práctica todas las cuestiones que mencionamos teóricamente. Es decir, veremos cómo son los prototipos, reconstruiremos varias imágenes en diferentes dimensiones, mediremos la calidad de la reconstrucción y, finalmente, comprobaremos lo que ocupa la información comprimida mediante la factorización.

Volviendo al caso del ACP, empezaremos analizando los prototipos para nuestra matriz X. La información para su representación se encuentra en la matriz H y, como ya hemos visto, se trata de las direcciones principales sobre las que proyectaremos los datos para reconstruir de nuevo las imágenes iniciales. La Figura 1.2 muestra una cuadrícula 7x7 donde se puede observar la apariencia de los prototipos obtenidos a partir de los 49 primeros autovectores.

Lee y Seung, en el artículo [10] publicado en Nature, afirmaban que las imágenes base o prototipos para el ACP son *autocaras* (*eigenfaces* en inglés) basadas en autovectores y representan versiones distorsionadas de caras completas. Podemos comprobar que esto es cierto, pues en dicha figura se puede observar que cada prototipo es la silueta de una cara completa.

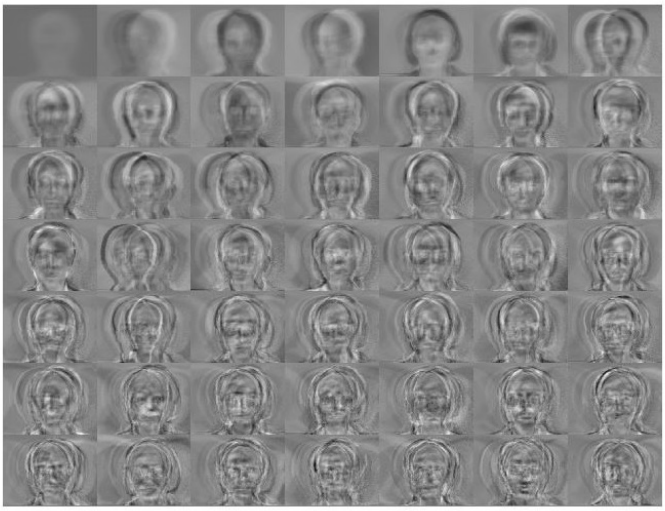

Figura 1.2: Representación de los prototipos obtenidos en H

Cabe destacar que, al hacer los cálculos para el ACP, obtenemos toda la información para representar los datos en las dimensiones que queramos. Si decidimos cambiar el número de direcciones principales sobre las que proyectaremos los puntos para buscar, por ejemplo, una mayor calidad de la reconstrucción, no hará falta volver a realizar las operaciones.

La Figura 1.3 muestra las reconstrucciones de dos de las imágenes pertenecientes al conjunto de datos X a partir de la información de 25, 50, 75, 100 y todos los autovectores respectivamente. Es decir, podemos ver una comparación entre las imágenes originales (las que se encuentran a la derecha) y diversas aproximaciones en distintas dimensiones (colocadas de menor a mayor).

Figura 1.3: Reconstrucción de dos imágenes

De esta forma, es fácil comprobar a simple vista que, en dimensión 100, tenemos una muy buena representación de los datos y, en dimensión 75, también es bastante adecuada.

En el gráfico de la Figura 1.4 se recogen los datos correspondientes a la calidad de la reconstrucción en las diferentes dimensiones obtenidas según el número de componentes elegidas. Como se ha visto en la sección dedicada a su cálculo, resulta muy sencillo obtenerlo a partir de los autovalores asociados a los autovectores con los que construyamos el subespacio sobre el que se proyectan los datos. En este caso, la calidad en dimensión x se trata de la suma de los x mayores autovalores, dividida entre la suma de todos ellos.

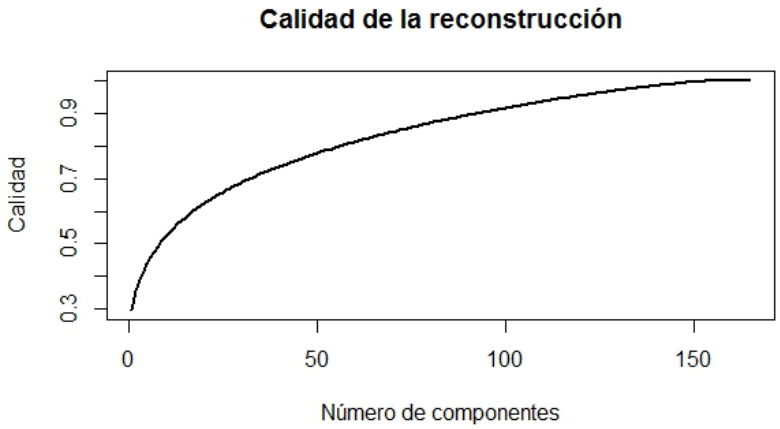

Figura 1.4: Gráfico de proporción de variabilidad explicada.

Esto nos ayuda a comprobar lo que ya habíamos visto en la figura anterior, pues a partir de dimensiones 75 y 100 la proporción de varianza recogida por el subespacio es elevada. Los datos exactos para dimensión 25, 50, 75 y 100 son los resumidos en la siguiente tabla:

Dimensión	25	50	75	100
Proporción de variabilidad explicada	0.6574	0.7769	0.8564	0.9159

Sin embargo, dado que en los siguientes métodos no se podrá hallar la calidad de la reconstrucción de la misma forma, se ha construido también un segundo gráfico que podremos repetir en todos los casos para poder comparar. Se trata de la Figura 1.5, donde se representa el error de reconstrucción según la dimensión. Se ha utilizado la distancia Euclídea para medir la diferencia entre la matriz original X y las matrices reconstruidas en varias dimensiones. En el gráfico se representa la raíz cuadrada de dichas diferencias al cuadrado.

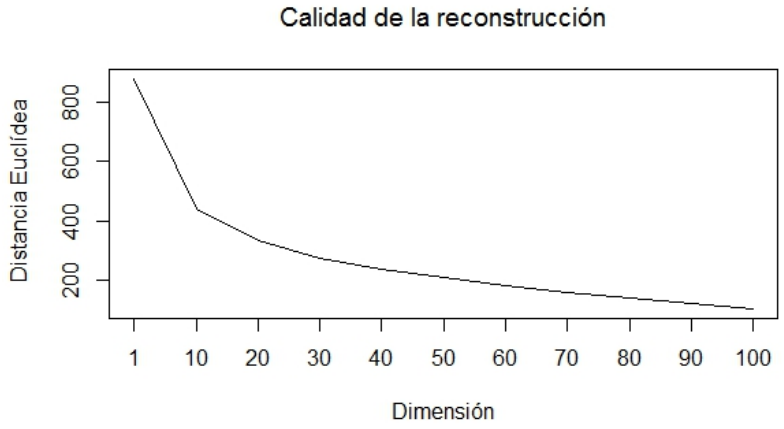

Figura 1.5: Gráfico del error de reconstrucción

En la siguiente tabla se puede ver un resumen de los errores relativos exactos para dimensión 25, 50, 75 y 100.

Dimensión	25	50	75	100
Error relativo	0.107	0.074	0.053	0.037

Por otro lado, es importante analizar cuánto ocupa la información que se debe guardar para reconstruir las imágenes con cada una de las aproximaciones elegidas. La información total, guardada en la matriz X con las coordenadas en la base canónica, ocupa 97.9 Mb.

Como ya hemos visto, tenemos que la matriz de prototipos H se corresponde con la matriz V que contiene la información de los autovectores. Luego, al aproximar, tenemos la correspondencia $H_k = V_k^T$. Además, sabemos que

$$X = UDV^T \Rightarrow XV = UD,$$

luego la matriz W en dimensión menor podremos obtenerla como $W_k = XV_k$.

En la siguiente tabla se expone un resumen de los tamaños de las matrices W y H para las dimensiones 25, 50, 75 y 100. Se puede observar claramente que la compresión de datos es notable incluso para las dimensiones que ya hemos comprobado que nos permiten realizar buenas aproximaciones. Por ejemplo, en el caso de la dimensión 75, logramos reducirlo más de la mitad.

	25	50	75	100
W	34 Kb	66.2 Kb	98.5 Kb	130.7 Kb
H	14.8 Mb	29.7 Mb	44.5 Mb	59.3 Mb

En las siguientes secciones de ejemplos, incluidas al final de cada capítulo, se compararán estos resultados con los que se obtengan gracias a los otros métodos expuestos en este trabajo. De esta forma, podremos extraer unas conclusiones al final de la memoria.

Capítulo 2

Cuantización Vectorial

En el primer capítulo hemos analizado el método del ACP, basado en reconstrucciones de baja dimensión óptimas en cuanto al error de reconstrucción o la maximización de la varianza recogida, lo que nos llevaba a restricciones de ortogonalidad para las columnas de la matriz H. Sin embargo, un error de aproximación bajo no es el único aspecto a tener en cuenta cuando se busca una representación de baja dimensión de un conjunto de puntos. Con vistas a una mayor interpretabilidad, puede resultar conveniente encaminar la búsqueda hacia *prototipos* que sean modelos representativos aproximadamente correctos para describir los puntos del conjunto de datos.

En el caso de las imágenes, como las componentes principales tienen que respetar la ortogonalidad, los prototipos no van a ser también imágenes (de hecho, tenemos combinaciones lineales con datos positivos y negativos). Si se quiere resumir la información de imágenes con modelos que se puedan seguir entendiendo como tal, hay que cambiar el criterio. Una manera de imponer que un prototipo de imágenes sea una imagen es forzar a que los prototipos sean promedios de los puntos del conjunto. El promedio de un conjunto de imágenes es, de manera natural, una imagen. En este capítulo, se estudian técnicas de factorización en las que los prototipos cumplen el requisito de ser promedios.

El método principal en esta categoría es un método de análisis cluster basado en la búsqueda de centros o centroides: el conocido método de k-medias. A lo largo de este capítulo vamos a estudiarlo y analizar su aplicación al ejemplo de imágenes faciales que ya hemos manejado anteriormente a través de lo que se conoce como Cuantización Vectorial (VQ). Además, explicaremos una de sus muchas variantes: el Análisis de Arquetipos de Cutler y Breiman (ver [2]), estrechamente relacionado con la Factorización No Negativa de Matrices, que estudiaremos en el tercer capítulo.

Al igual que en el caso del ACP, también existen múltiples referencias

sobre el análisis cluster, el método de k-medias, la Cuantización Vectorial y el Análisis de Arquetipos. Para este trabajo, nos hemos basado especialmente en los libros [5], [6] y [8] durante las primeras secciones y en los artículos [2] y [3] para la dedicada al Análisis de Arquetipos.

2.1. Análisis Cluster y método de k-medias

El análisis cluster, también llamado segmentación de datos, tiene varios objetivos. Todos están relacionados con agrupar o segmentar una colección de objetos en subconjuntos o *clusters*, tales que aquellos que pertenecen a un mismo cluster están más estrechamente relacionados entre sí que los objetos asignados a diferentes clusters.

El método de k-medias es uno de los más populares para el análisis cluster. Dada la matriz de datos $X \in \mathcal{M}_{n\times p}(\mathbb{R})$, este método consiste en buscar unos vectores $m_1, ..., m_k$ pertenecientes a $\mathbb{R}^p$ que minimicen la función objetivo

$$\sum_{i=1}^{n} \min_{1\leq j\leq k} \|x_i - m_j\|^2 \tag{2.1}$$

donde x_i denota a cada fila de la matriz X. Dichos vectores $m_1, ..., m_k$ son los que se conocen por el nombre de k-medias. Una vez calculados los m_j, se definen los clusters asociados, C_j, como el conjunto de índices formado por las observaciones más próximas al centro m_j, es decir,

$$C_j = \{i : \|x_i - m_j\| < \|x_i - m_l\|, \quad \forall\, l \neq j\}.$$

Los puntos de C_j son los que están más cerca de m_j que de cualquier otro centro.

Con esta notación podemos reescribir la función objetivo (2.1) en la forma

$$\sum_{j=1}^{k} \sum_{i\in C_j} \|x_i - m_j\|^2$$

Observamos que, una vez fijado el cluster C_j, $\sum_{i\in C_j} \|x_i - m_j\|^2$ se minimiza si $m_j = \frac{1}{n_j} \sum_{i\in C_j} x_i$, donde $n_j = \sharp\{i : i \in C_j\}$ es el número de observaciones asignadas al conjunto C_j. En otras palabras, las k-medias $m_1, ..., m_k$ son promedios de puntos en el conjunto de datos.

El método de k-medias trata de buscar k prototipos de forma que los puntos en cada cluster sean parecidos entre sí y también al prototipo, mientras que se intenta que los prototipos sean claramente diferentes entre sí. Trataremos de explicar el sentido de esta frase a continuación.

Sea $\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \in \mathbb{R}$ la media de los datos, consideramos la varianza total

$$Var(X) = \sum_{i=1}^{n} \|x_i - \mu\|^2.$$

A continuación veremos que $Var(X)$ se puede descomponer en dos sumandos. El primero de ellos se corresponderá con la variación dentro de un mismo cluster y el segundo, con la variación entre clusters.

Por lo tanto, tenemos que

$$\begin{aligned}\|x_i - \mu\|^2 &= \|x_i - m_j + m_j - \mu\|^2 \\ &= \|x_i - m_j\|^2 + \|m_j - \mu\|^2 + 2\langle x_i - m_j, m_j - \mu\rangle .\end{aligned}$$

Y de ahí que

$$\sum_{i\in C_j} \|x_i - \mu\|^2 = \sum_{i\in C_j} \|x_i - m_j\|^2 + n_j\|m_j - \mu\|^2 + 2\left\langle \sum_{i\in C_j}(x_i - m_j), m_j - \mu \right\rangle .$$

El último de estos sumandos es igual a 0 ya que tomamos cada m_j como la media del grupo de observaciones correspondiente, luego $\frac{1}{n_j}\sum_{i\in C_j} x_i = m_j$.

Escribiéndolo en general, sumando las cantidades correspondientes para todo j con $1 \leq j \leq k$, llegamos a un resultado conocido como Principio de Huygens.

$$\sum_{j=1}^{k} \frac{1}{n} \sum_{i\in C_j} \|x_i - \mu\|^2 = \sum_{j=1}^{k} \frac{n_j}{n} \left[\sum_{i\in C_j} \|x_i - m_j\|^2\right] + \sum_{j=1}^{k} \frac{n_j}{n} \left[\sum_{i\in C_j} \|m_j - \mu\|^2\right]. \tag{2.2}$$

En (2.1) tenemos la minimización de la cantidad correspondiente al primer sumando de esta última ecuación. Dado que la varianza total es un dato constante, gracias a (2.2) sabemos que minimizar la variación dentro de clusters es equivalente a maximizar la variación entre clusters.

2.2. Algoritmos

Se pueden buscar soluciones al problema de minimización expuesto en la sección anterior a través de diversos algoritmos. En la literatura, en ocasiones se confunde la idea del método de k-medias con la explicación de un algoritmo iterativo que lo intenta solucionar.

Los algoritmos para el análisis cluster más populares asignan directamente cada observación a un grupo o cluster sin tener en cuenta un modelo de probabilidad que describa los datos. Son los llamados *algoritmos combinatorios*. Estos algoritmos son aplicables a funciones algo más generales que (2.1). En principio, la discrepancia entre la observación i y la observación i' se mide con una función $d(x_i, x_{i'})$. Cada observación se relaciona con un cluster, identificado con un entero $k \in \{1, ..., K\}$, a través de una asignación que se puede escribir como $k = C(i)$. De esta forma, el objetivo es buscar la asignación que minimiza:

$$W(C) = \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} d(x_i, x_{i'}). \tag{2.3}$$

La función objetivo (2.1) encaja en este tipo tomando $d(x_i, x_{i'}) = \|x_i - x_{i'}\|^2$ como detallaremos más adelante.

A la función (2.3) se le llama generalmente en la literatura *función de energía*.

El análisis cluster por optimización combinatoria es sencillo en principio. Se trata de minimizar W entre todas las posibles asignaciones de los n puntos en los K clusters. Desgraciadamente, dicha optimización revisando todas las posibilidades es factible sólo para conjuntos de datos muy pequeños. En realidad, el número de asignaciones distintas es

$$S(n, K) = \frac{1}{K!} \sum_{k=1}^{K} (-1)^{K-k} \binom{K}{k} k^n. \tag{2.4}$$

La demostración de esta fórmula se puede encontrar en [6] y su razonamiento es el siguiente:

Denotemos por $S(n, K)$ el número de formas posibles de agrupar n objetos en K grupos. El orden de los objetos en cada grupo y el orden de los propios grupos es irrelevante. Los grupos vacíos no se cuentan. Podemos escribir una ecuación en diferencias para $S(n, K)$ de la siguiente manera. Suponemos que tenemos una lista con todas las agrupaciones de $n-1$ objetos. Una agrupación de n objetos se puede formar de dos formas a partir de esta lista:
- El objeto n-ésimo se puede añadir como un grupo de un único elemento a cada miembro de la lista con exactamente $K - 1$ grupos.
- El objeto n-ésimo se puede añadir a cada grupo de cualquier miembro de la lista con exactamente K grupos.

Por tanto,

$$S(n, K) = S(n - 1, K - 1) + KS(n - 1, K).$$

Las condiciones frontera de esta ecuación son $S(n,1) = 1$, $S(n,n) = 1$, $S(n,K) = 0$ si $K > n$.
La solución a esta ecuación para $S(n,K)$ requiere que los valores $S(j,p)$ sean conocidos para el conjunto $\{(j,p) : 1 \leq j \leq n-2, 1 \leq p \leq K\}$.
Las soluciones a la ecuación en diferencias se llaman números de Stirling de segundo orden:

$$S(n,K) = \frac{1}{K!}\sum_{k=1}^{K}(-1)^{K-k}\binom{K}{k}k^n.$$

En la siguiente tabla se representa el número de asignaciones posibles de n observaciones en K clusters para ciertos valores de estos parámetros.

N \ k	2	3	4	5
5	15	25	10	1
10	511	9330	34105	42525
20	524287	580606446	45232115901	749206090500
50	$5.6295 \cdot 10^{14}$	$1.196497 \cdot 10^{23}$	$5.281866 \cdot 10^{28}$	$7.400959 \cdot 10^{32}$

Se puede ver que, para valores pequeños de n y K es bastante factible. Sin embargo, $S(n,K)$ crece muy rápido al aumentar los valores de sus argumentos. Cualquiera de los valores obtenidos para los casos en los que tenemos 50 observaciones son mayores que 10^{10}, y la mayoría de los problemas de análisis cluster involucran conjuntos de datos mucho más grandes que $n = 50$. Por esta razón, los algoritmos prácticos para el análisis cluster son capaces de examinar sólo una muy pequeña parte de todas las posibles codificaciones $k = C(i)$. El objetivo es identificar un pequeño subconjunto que probablemente contenga el óptimo, o al menos una buena partición subóptima.

Existen varios algoritmos para lograr este objetivo y funcionan de la siguiente manera: Primero se especifica una partición inicial. En cada paso iterativo, las asignaciones de cluster se cambian de tal manera que el valor del criterio mejore respecto a su valor anterior. Los algoritmos de este tipo difieren en sus condiciones para modificar las asignaciones de cluster en cada iteración. Cuando no se puede proporcionar una mejora, el algoritmo termina con las asignaciones actuales como su solución. Dado que la asignación de observaciones a clusters en cualquier iteración es una perturbación de la iteración anterior, sólo se examina una fracción muy pequeña de todas las asignaciones posibles (2.4). Sin embargo, estos algoritmos convergen a óptimos locales que pueden ser subóptimos que se acercan bastante al óptimo global.

2.3. Algoritmo de k-medias

El algoritmo de k-medias es uno de los métodos iterativos más populares para el análisis cluster. Se utiliza para situaciones en las que se manejan variables cuantitativas y se elige

$$d(x_i, x_{i'}) = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = \|x_i - x_{i'}\|^2$$

como forma de medida basada en la distancia Euclídea.

La fórmula (2.3) puede escribirse como

$$\begin{aligned} W(C) &= \frac{1}{2} \sum_{k=1}^{K} \sum_{C(i)=k} \sum_{C(i')=k} \|x_i - x_{i'}\|^2 \\ &= \sum_{k=1}^{K} n_k \sum_{C(i)=k} \|x_i - \bar{x}_k\|^2, \end{aligned}$$

donde $\bar{x}_k = (\bar{x}_{1k}, ..., \bar{x}_{pk})$ es el vector de medias asociado al k-ésimo cluster, y $n_k = \sum_{i=1}^{N} I(C(i) = k)$ es el número de observaciones en el cluster k. Por tanto, el criterio se minimiza asignando las n observaciones a los K clusters de forma que, en cada cluster, se minimiza la diferencia promedio entre las observaciones de la media del grupo, tal como se define por los puntos de ese grupo.

Un algoritmo de descenso iterativo para resolver

$$C^* = \min_{C} \sum_{k=1}^{K} n_k \sum_{C(i)=k} \|x_i - \bar{x}_k\|^2$$

se puede obtener al observar que para cualquier conjunto de observaciones S

$$\bar{x}_S = argmin_m \sum_{i \in S} \|x_i - m\|^2. \tag{2.5}$$

De ahí podemos obtener C^* resolviendo el siguiente problema de optimización.

$$C^* = \min_{C, \{m_k\}_1^K} \sum_{k=1}^{K} n_k \sum_{C(i)=k} \|x_i - m_k\|^2. \tag{2.6}$$

Esto se puede minimizar con un procedimiento de optimización alternativo dado en el Algoritmo 2.3.1.

Algoritmo 2.3.1.

1- Para una asignación de cluster C dada, la varianza total de clusters (2.6) se minimiza respecto a $\{m_1, ..., m_k\}$ produciendo las medias de los clusters asignados actualmente (2.5).

2- Dado el actual conjunto de medias $\{m_1, ..., m_k\}$, (2.6) se minimiza asignando cada observación a la actual media de cluster más cercana. Esto es,

$$C(i) = argmin_{1\leq k\leq K}\|x_i - m_k\|^2. \tag{2.7}$$

3- Se iteran los pasos 1 y 2 hasta que las asignaciones no cambien.

Cada uno de los pasos 1 y 2 reduce el valor del criterio (2.6), por lo que la convergencia está asegurada. Sin embargo, el resultado puede representar un mínimo local subóptimo, como se ha explicado anteriormente.

2.4. Cuantización Vectorial

El algoritmo de k-medias es una herramienta clave en el área aparentemente no relacionada de compresión de imagen y señal, particularmente en la cuantización vectorial o VQ (Vector Quantization).

Podemos ver cada una de las filas de la matriz de datos X (de dimensión $n \times p$) como uno de los puntos de datos que tenemos como observaciones y deseamos asignar a uno de los clusters. Eligiendo k como número de clusters, lo que queremos es buscar los puntos $\{\hat{m}_1, ..., \hat{m}_k\}$ que minimicen

$$\sum_{i=1}^{n} \min_{1\leq j\leq k} \|x_i - \hat{m}_j\|^2$$

donde cada x_i denota una fila de la matriz X.

Utilizando el algoritmo de k-medias descrito anteriormente, llegamos a que la compresión de los datos consiste en identificar cada x_i con su mejor aproximación dada por este método. Esto es, la media del cluster al que se le haya asignado. Por tanto, tendríamos que

$$\tilde{x}_i = \hat{m}_j(i), \tag{2.8}$$

donde $1 \leq j(i) \leq k$.

En este caso, buscamos aproximar $X \approx WH$ de la misma forma que hemos hecho en ACP, con W de dimensión $n \times r$ y H de dimensión $r \times p$, pero

con r igual al número de k-medias. Por tanto, la factorización $\tilde{X} = W_k H_k$ queda de la siguiente forma:

- Cada una de las filas de la matriz H_k se corresponde con una de las medias obtenidas mediante el algoritmo y son lo que hemos llamado anteriormente *prototipos*. Sus dimensiones son $k \times p$.

$$H = \begin{bmatrix} \hat{m}_1 \\ \vdots \\ \hat{m}_k \end{bmatrix}$$

- La matriz W_k guarda la información de las *etiquetas* $j(i)$, que nos sirven para saber con qué prototipo se corresponde cada una de las filas de X. La fila w_i está compuesta por ceros y un único uno, en la posición $j(i)$. Sus dimensiones son $n \times k$.

De esta forma, al multiplicar ambas matrices obtenemos la relación definida en la ecuación (2.8).

2.5. Ejemplo

Hemos aplicado la función *kmeans* de R a la matriz X que contiene los datos de las imágenes que ya utilizamos en la sección 1.4. De esta forma, analizaremos los datos obtenidos y podremos comprarlos con lo que hemos visto en el método de ACP. A lo largo de esta sección veremos cómo son los prototipos dependiendo de la dimensión pedida, cómo son las aproximaciones de dos de las imágenes, cuáles son los errores de reconstrucción y, finalmente, cuánto ocupa la información que debemos almacenar en cada caso.

Comenzaremos de nuevo representando los prototipos que se obtienen al utilizar el algoritmo en dimensión 49. En la Figura 2.1 tenemos una cuadrícula 7x7 donde podemos observar las imágenes base que forman parte de la matriz H. Como ya hemos dicho, se trata de las k-medias.

A diferencia del ACP, en el caso del método de VQ sí que debemos repetir las operaciones si queremos representar los datos en una dimensión menor. Por este motivo hemos creado las Figuras 2.2 y 2.3, donde podemos ver que se han obtenido unos prototipos diferentes para 25 y 9 k-medias respectivamente.

A continuación, vamos a repetir la reconstrucción en dimensión 25, 50, 75 y 100 para las dos imágenes que utilizamos en la sección 1.4. Se encuentran representadas en la Figura 2.4 de las misma forma que lo hicimos anteriormente (de menor a mayor dimensión y las originales a la derecha).

Es esta figura se puede ver que, con el método de k-medias, también llegamos a que las representaciones más adecuadas aparecen a partir de las

Figura 2.1: Prototipos guardados en H (dimensión 49)

dimensiones 75 o 100. Con estos datos conseguimos que, incluso en las dimensiones más bajas, las aproximaciones no sean muy malas. Esto puede deberse a que nuestros datos se dividen en unos grupos claramente diferenciados (se trata de fotografías de 15 individuos en diversas condiciones).

Ahora veremos numéricamente cuál es la calidad de la reconstrucción conseguida con este método para poder comparar exactamente cuál de ellos explica mejor su variabilidad. Para ello, vamos a utilizar la misma forma de medida que ya explicamos en el caso del ACP. Cuantificaremos el error de reconstrucción a partir de la distancia Euclídea, es decir, calculando la raíz cuadrada de norma al cuadrado de la distancia entre la matriz original y la matriz aproximada por la multiplicación de los factores en cierta dimensión. La Figura 2.5 muestra un gráfico hasta dimensión 100.

En la siguiente tabla se puede ver un resumen de los errores relativos exactos para dimensión 25, 50, 75 y 100. De esta forma, al compararlos con los resultados obtenidos para el ACP, podemos llegar a la conclusión de que los errores son mayores en este caso. Esto no es sorprendente, ya que el método del ACP está creado con el propósito de minimizar dicho error cuando disminuimos la dimensión.

Dimensión	25	50	75	100
Error relativo	0.143	0.106	0.081	0.059

Por último, nos detendremos a analizar cómo es la compresión de información y la cantidad de espacio requerido para almacenarla. Es importante

Figura 2.2: Prototipos guardados en H (dimensión 25)

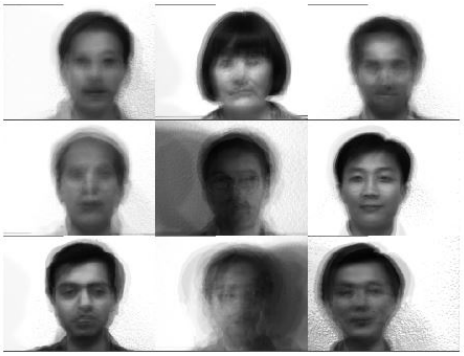

Figura 2.3: Prototipos guardados en H (dimensión 9)

destacar que, en este método, no necesitamos guardar las matrices W y H. Nos basta con tener las k-medias (que forman H) y un vector de asignaciones, que nos indica con qué k-media se corresponde cada una de las observaciones (ambos obtenidos como salidas del algoritmo *kmeans* que hemos utilizado). Por lo tanto, en vez de tener que almacenar la información de una matriz W de dimensión $n \times r$, nos basta con un vector de longitud n. De esta forma, vemos que la compresión de información es obviamente mayor en este caso.

La siguiente tabla muestra un resumen de estos datos para las dimensiones 25, 50, 75 y 100.

	25	50	75	100
Vector de asignaciones	2.2 Kb	2.2 Kb	2.2 Kb	2.2 Kb
H (k-medias)	14.8 Mb	29.7 Mb	44.5 Mb	59.3 Mb

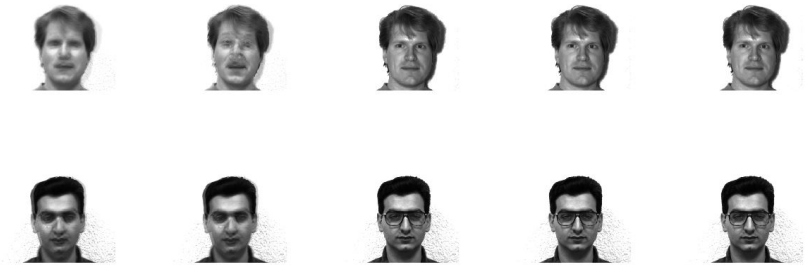

Figura 2.4: Reconstrucción de dos imágenes

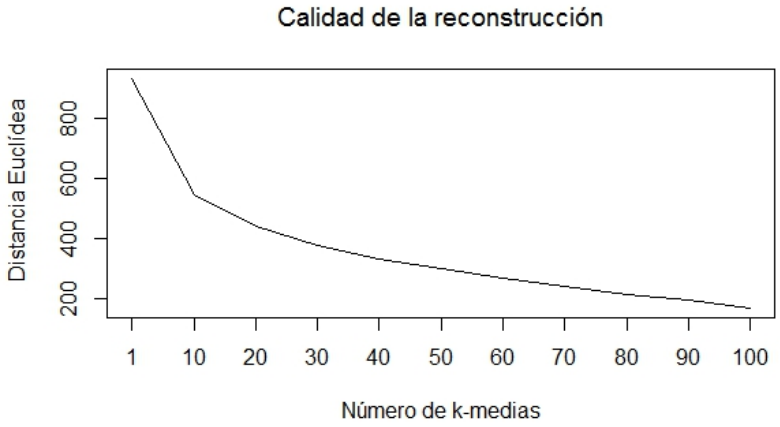

Figura 2.5: Calidad de la reconstrucción

Al final del siguiente capítulo, en la sección 3.5, se verá una comparación de los resultados obtenidos con los tres métodos estudiados a lo largo de este trabajo.

2.6. Análisis de Arquetipos

Este método, debido a Cutler y Breiman (ver [2]), aproxima los datos por prototipos que son ellos mismos combinaciones lineales de los datos. En este sentido es similar al método de k-medias. Sin embargo, en vez de aproximar cada punto de datos por un único prototipo cercano, el análisis de arquetipos aproxima cada punto por una combinación convexa de una colección de prototipos. El uso de una combinación convexa fuerza a los prototipos a estar contenidos en la envolvente convexa de la nube de datos. En este sentido, los prototipos son *puros* o *arquetípicos*.

La matriz de datos X de dimensiones $n \times p$ se modela como

$$X \approx WH, \tag{2.9}$$

donde W es $n \times r$ y H es $r \times p$. Suponemos que $w_{ik} \geq 0$ y $\sum_{k=1}^{r} w_{ik} = 1 \quad \forall i$. Por tanto, los n puntos de datos (filas de X) en espacio p-dimensional se representan por combinaciones convexas de los r arquetipos (filas de H). También suponemos que

$$H = BX, \tag{2.10}$$

donde B es $r \times n$ con $b_{ki} \geq 0$ y $\sum_{i=1}^{n} b_{ki} = 1 \quad \forall k$. Así que los arquetipos son ellos mismos combinaciones convexas de los datos. Usando (2.9) y (2.10), minimizamos

$$\begin{aligned} J(W, B) &= \|X - WH\|^2 \\ &= \|X - WBX\|^2 \end{aligned} \tag{2.11}$$

respecto de las variables W y B. Esta función se minimiza de forma alterna, con cada minimización por separado implicando una optimización convexa.

Este método es bastante parecido a los métodos de proyecciones alternadas para la factorización no negativa de matrices que se describen en el tercer capítulo de esta memoria. En particular, el que busca la minimización del error en norma de Frobenius. La principal peculiaridad en el caso del algoritmo de arquetipos es que se debe respetar la restricción de permanencia en el simplex.

El análisis de arquetipos es un método que se está desarrollando especialmente en los últimos años, como muestran los artículos altamente citados que se incluyen como referencias a esta sección ([2] y [3]).

No obstante, la restricción antes mencionada de permanencia en el simplex hace aumentar el coste computacional de este algoritmo. Para el conjunto de imágenes que hemos analizado en los demás métodos, los algoritmos disponibles (en particular, la librería *archetypes* de R) no han producido una iteración convergente. Por esta razón, el estudio numérico del rendimiento del método que se ha llevado a cabo en otras secciones de esta memoria no ha sido posible en este caso.

Capítulo 3

Factorización No Negativa de Matrices

La Factorización No Negativa de Matrices (NMF) es un método que ha ganado popularidad en los últimos años. Consiste en la factorización aproximada de una matriz X de entradas positivas, imponiendo como restricciones la no negatividad de los factores matriciales W y H obtenidos.

Como referencias para este capítulo se han utilizado diversos artículos y libros relacionados con este tema. Las primeras secciones son una síntesis de las ideas expuestas en [5], [9] y [10], pero para la tercera nos hemos basado especialmente en [4] y [11].

3.1. Motivación

Al igual que en los capítulos anteriores, nuestro objetivo es la factorización de una matriz X que nos permita aproximarla reduciendo las dimensiones de sus factores. Es decir, buscar unas matrices W y H tales que

$$X \approx WH.$$

De esta forma, podemos comprimir los datos, de forma que necesitemos un menor espacio para almacenarlos, procurando mantener la mayor cantidad de información posible. Para ello, ya hemos visto el método de ACP, que busca minimizar el error de reconstrucción, o la cuantización vectorial, que utiliza el método de k-medias para resumir los datos a través de unos prototipos que sean promedios. La particularidad del método de factorización no negativa reside en el hecho de que tanto la matriz X, como los factores matriciales W y H, se restringen a ser no negativos. Los puntos reconstruidos son ahora una combinación cónica de los prototipos. Una normalización

recuperaría una combinación convexa, pero la eliminación de restricciones produce, esperamos, ventajas computacionales.

La motivación para este capítulo es el ya citado artículo [10], publicado en Nature en 1999. En él, Lee y Seung hacían una comparación de los tres métodos que analizamos en este trabajo (ACP, VQ y NMF) y los aplicaban a una base de datos de imágenes faciales. Los resultados que exponían se pueden ver en la Figura 3.1. Afirmaban que, mientras que las *autocaras* (*eigenfaces* en inglés) y los prototipos obtenidos a partir de ACP y VQ se trataban de representaciones *holísticas* y no tenían una interpretación sencilla, gracias a NMF se podía conseguir una descomposición basada en partes de caras. De esta forma, pese a perder calidad o nivel de compresión dada la fuerte restricción de no negatividad, se podía ganar en interpretabilidad de los resultados.

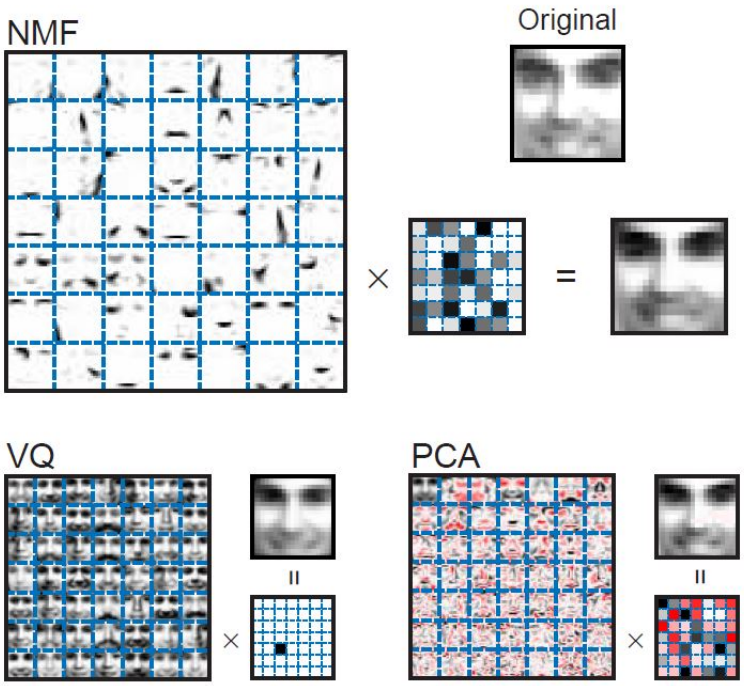

Figura 3.1: Ejemplo de Lee y Seung en Nature

En el caso de esas imágenes y las que analizamos en este trabajo, este método tiene especial sentido, dado que los datos que forman la matriz X son intensidades de gris (cantidades entre 0 y 1). Por lo tanto, estamos manejando números positivos y que, de hecho, podremos suponer naturales con la normalización adecuada. A diferencia de la restricción de VQ, donde

cada cara se aproximaba por un único prototipo, las restricciones de no negatividad que pedimos con NMF sí que permiten la combinación de varias imágenes base para representar una cara. No obstante, sólo están permitidas las combinaciones aditivas, pues los elementos distintos de cero en W y H son todos positivos. En contraste con el ACP, aquí no pueden ocurrir sustracciones. Por estas razones, las restricciones de no negatividad parecen ser compatibles con la noción intuitiva de combinar partes para formar una imagen completa.

A lo largo de este capítulo veremos más a fondo en qué consiste el método de NMF, expondremos algunos algoritmos que han sido propuestos para su cálculo y discutiremos la unicidad de la factorización. Por último, lo aplicaremos a la base de datos de imágenes con la que ya hemos trabajado previamente y trataremos de extraer unas conclusiones finales a través de la comparación con los resultados obtenidos con ACP y VQ.

3.2. Descripción del problema

El objetivo que perseguimos con el método de factorización no negativa puede ser resumido en el siguiente problema:

Factorización No Negativa de Matrices (NMF): Dada una matriz de datos $X \in \mathcal{M}_{n\times p}(\mathbb{R})$, con $x_{ij} \geq 0$, encontrar una factorización tal que

$$X \approx WH,$$

donde W es $n \times r$ y H es $r \times p$, con $r \leq \text{máx}(n, p)$. Además, pedimos que $w_{ik}, h_{kj} \geq 0$.

De esta forma, conseguiremos una factorización donde la matriz de pesos W y la matriz de prototipos H conservan la propiedad de no negatividad.

Para encontrar una factorización $X \approx WH$, primero necesitaremos formalizar la noción de proximidad " $\approx$ ". Al igual que en los capítulos anteriores, buscamos las matrices de tal forma que se minimice cierto criterio. En este caso, lo haremos definiendo una función que cuantifique la calidad de la aproximación. Dicha función puede ser construida usando alguna medida de discrepancia entre dos matrices no negativas X y WH de dimensiones $n \times p$ de forma que nuestro objetivo sea

$$\min_{W,H\geq 0} d(X; W, H),$$

donde d es una divergencia o distancia.

Entre las opciones de las que disponemos se encuentran, por ejemplo, la distancia euclídea y la divergencia de Kullback-Leibler. La primera de ellas

es la más natural, por lo que es muy utilizada. Está basada en la norma de Frobenius de una matriz A de dimensión $n \times p$, definida del siguiente modo

$$\|A\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{p} |a_{i,j}|^2.$$

Esta norma da lugar a la siguiente distancia entre las matrices X y WH:

$$\|X - WH\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - (WH)_{ij})^2. \tag{3.1}$$

Basándonos en (3.1), la factorización (aproximada) no negativa de X estará dada por W y H no negativas de dimensiones $n \times r$ y $r \times p$ respectivamente, solución de

$$\min_{W,H \geq 0} \sum_{i=1}^{n} \sum_{j=1}^{p} (x_{ij} - (WH)_{ij})^2. \tag{3.2}$$

La divergencia de Kullback-Leibler o entropía relativa, que denotaremos por d_{KL}, es una medida de discrepancia con motivación probabilística. Si P y Q son dos probabilidades discretas con funciones de masa de probabilidad $p(x)$ y $q(x)$ respectivamente, la divergencia de Kullback-Leibler es

$$d_{KL}(P, Q) = \sum_{i} p(i) \log \frac{p(i)}{q(i)}, \tag{3.3}$$

con $d_{KL}(P, Q) = +\infty$ si $q(i) = 0$ para algún i con $p(i) \neq 0$.

Tenemos que $d_{KL}(P, Q)$ se anula si $P = Q$ y, en cualquier otro caso, es estrictamente positiva. Sin embargo, no se trata de una distancia, pues no satisface la desigualdad triangular ni es simétrica en P y Q. De todas formas, resulta útil dado que tiene un sentido probabilístico interesante, relacionado con el estimador máximo verosímil, que tratamos de explicar a continuación.

Supongamos que $X_1, \ldots, X_n$ es una m.a.s. de la función de probabilidad p y que el modelo $\{p(x, \theta) : \theta \in \Theta\}$ es correcto, es decir, que existe un $\theta_0 \in \Theta$ tal que $p = p(\cdot|\theta_0)$. Escribimos P_θ para la probabilidad asociada a p y asumimos que distintos valores de θ se corresponden con probabilidades P_θ diferentes. Entonces, dada la positividad de d_{KL}, tenemos que θ_0 es el único maximizador de la función

$$K(\theta) := -d_{KL}(P_{\theta_0}, P_\theta), \quad \theta \in \Theta$$

El EMV , $\hat{\theta}_n$, es el argumento que maximiza la función de log-verosimilitud

$$l_n(\theta) = \sum_{i=1}^{n} \log p(X_i|\theta).$$

Obviamente también es el maximizador de

$$K_n(\theta) := \frac{1}{n}\sum_{i=1}^{n} \log p(X_i|\theta) - \frac{1}{n}\sum_{i=1}^{n} \log p(X_i|\theta_0) = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{p(X_i|\theta_0)}{p(X_i|\theta)}.$$

Puesto que $E_{\theta_0}(log\frac{p(X_1|\theta_0)}{p(X_1|\theta)}) = d_{KL}(P_{\theta_0,P_\theta})$, la Ley de los Grandes Números nos dice que $K_n(\theta)$ converge c.s. a $K(\theta)$ para cada $\theta \in \Theta$. Intuitivamente, puesto que las funciones K_n se aproximan a la función K, los maximizadores de K_n se deberían aproximar al maximizador de K, es decir, a θ_0. En algunos casos se puede probar convergencia uniforme c.s. de $K_n(\theta)$ a $K(\theta)$, es decir, que

$$\sup_{\theta\in\Theta} |K_n(\theta) - K(\theta)| \underset{c.s.}{\to} 0$$

y, a partir de esta convergencia uniforme, es posible probar que $\hat{\theta}_n \to_{c.s.} \theta_0$.

Este tipo de resultado da una buena justificación al empleo de la divergencia d_{KL} en contextos estadísticos, puesto que minimizar la divergencia d_{KL} conduce a la estimación consistente del "verdadero valor del parámetro".

Para poder usar la divergencia d_{KL} en el estilo anterior, necesitamos plantear un modelo estadístico para las observaciones $x_{i,j}$. En nuestro caso, asumimos un modelo en el que $x_{i,j}$ tiene una distribución de Poisson de media $(WH)_{ij}$, lo que es bastante razonable para unos datos positivos y que, con la normalización adecuada, podemos suponer naturales. Sin ir más lejos, en el ejemplo que estamos utilizando a lo largo del trabajo, los valores $x_{i,j}$ se corresponden con intensidades de gris, por lo que tiene sentido pensar que podemos medirlos en escala entera.

Dado que la función de probabilidad de un modelo de Poisson de media μ es

$$f(k,\mu) = \frac{e^{-\mu}\mu^k}{k!},$$

su logaritmo es

$$\log \frac{e^{-\mu}\mu^k}{k!} = -\mu + k\log\mu - \log k!.$$

Y de ahí obtenemos la función de log-verosimilitud en este modelo de Poisson,

$$l(W,H) = \sum_{i=1}^{N}\sum_{j=1}^{p} \left[-(WH)_{ij} + x_{i,j}\log(WH)_{ij} - \log x_{ij}\right].$$

Teniendo en cuenta que queremos maximizar respecto de W y H y el último sumando no depende de ello, llegamos a que el EMV es el maximizador de la siguiente función:

$$L(W,H) = \sum_{i=1}^{N}\sum_{j=1}^{p}\left[x_{i,j}\log(WH)_{ij} - (WH)_{ij}\right]. \tag{3.4}$$

Equivalentemente, encontraremos las matrices W y H maximizando

$$L(W,H) = \sum_{i=1}^{N}\sum_{j=1}^{p}\left[x_{ij}\log\left(\sum_{k=1}^{r} w_{ik}h_{kj}\right) - \sum_{k=1}^{r} w_{ik}h_{kj}\right], \tag{3.5}$$

escrito sin la notación matricial.

3.3. Algoritmos

La maximización de las funciones (3.1) y (3.5) no puede hacerse mediante fórmulas explícitas. Por ello, a lo largo de los últimos años, se han propuesto diversos algoritmos para resolver el problema de NMF. A continuación, describimos los dos métodos más usados y probaremos algunas propiedades elementales. Un estudio completo de la convergencia de estos algoritmos requiere una discusión previa sobre la existencia y unicidad de minimizadores de (3.1) y (3.5). Esta cuestión se pospone a la sección 3.4.

En [9] se proponen los siguientes algoritmos:

Algoritmo 3.3.1. El siguiente algoritmo, propuesto por Lee y Seung, es un algoritmo de descenso para la función $\|X - WH\|^2$.

-Inicialización: Partimos de $W^{(0)}$ y $H^{(0)}$.

-Paso principal: Dados $W^{(n)}$ y $H^{(n)}$, hacemos

$$w_{ik}^{(n+1)} \leftarrow w_{ik}^{(n)} \frac{(X(H^{(n)})^T)_{ik}}{(W^{(n)}H^{(n)}(H^{(n)})^T)_{ik}} \tag{3.6}$$

$$h_{kj}^{(n+1)} \leftarrow h_{kj}^{(n)} \frac{((W^{(n+1)})^T X)_{kj}}{((W^{(n+1)})^T W^{(n+1)}H^{(n)})_{kj}} \tag{3.7}$$

Algoritmo 3.3.2. El siguiente algoritmo, propuesto por Lee y Seung, es un algoritmo de descenso para la función $-L(W,H)$.

-Inicialización: Partimos de $W^{(0)}$ y $H^{(0)}$.

-Paso principal: Dados $W^{(n)}$ y $H^{(n)}$, hacemos

$$w_{ik}^{(n+1)} \leftarrow w_{ik}^{(n)} \frac{\sum_{j=1}^{p} h_{kj}^{(n)} x_{ij}/(W^{(n)}H^{(n)})_{ij}}{\sum_{j=1}^{p} h_{kj}^{(n)}} \tag{3.8}$$

$$h_{kj}^{(n+1)} \leftarrow h_{kj}^{(n)} \frac{\sum_{i=1}^{N} w_{ik}^{(n+1)} x_{ij}/(W^{(n+1)}H^{(n)})_{ij}}{\sum_{i=1}^{N} w_{ik}^{(n+1)}} \tag{3.9}$$

En la práctica, se implementa el algoritmo hasta que la diferencia entre iterantes sucesivos es menor que cierta tolerancia. No hay garantías teóricas en la literatura sobre la convergencia de esos iterantes hacia un mínimo global. Un estudio completo al respecto se verá más adelante. En la práctica, las inicializaciones se eligen al azar. Se prueba con varias y se conserva finalmente el mejor resultado obtenido.

¿Qué sentido tienen las iteraciones propuestas en estos algoritmos? Vamos a comprobar que efectivamente conducen a algoritmos de descenso, es decir, que en cada iteración disminuye el valor de la función objetivo.

Para probarlo, utilizaremos el concepto de función mayorizante, definido del siguiente modo:

Definición 3.3.3. Una función $g(x, y)$ se dice que mayoriza una función $f(x)$ si

$$g(x, y) \geq f(x), \quad g(x, x) = f(x)$$

para todo x, y en el dominio.

Este concepto es útil por el siguiente lema.

Lema 3.3.4. Si $g(x, y)$ mayoriza $f(x)$, entonces $f(x)$ es decreciente para la sucesión definida mediante la iteración

$$x^{(n+1)} = \underset{x}{argmin}\, g(x, x^{(n)}). \tag{3.10}$$

Demostración. Según la definición anterior, si $g(x, y)$ mayoriza $f(x)$, tomando $x = x^{(n+1)}$ e $y = x^{(n)}$, tenemos que

$$f(x^{(n+1)}) \leq g(x^{(n+1)}, x^{(n)}) \leq g(x^{(n)}, x^{(n)}) = f(x^{(n)})$$

□

Lee y Seung en el artículo [9] afirman que $f(x^{(n+1)}) = f(x^{(n)})$ sólo si $x^{(n)}$ es un mínimo local de $g(x, x^{(n)})$ y que iterando (3.10) se obtiene una sucesión de estimadores que converge a un mínimo local $x_{min} = argmin_x f(x)$ de la función f.

$$f(x_{min}) \leq \ldots \leq f(x^{(n+1)}) \leq f(x^{(n)}) \leq \ldots \leq f(x_1) \leq f(x_0)$$

Sin embargo, esto no es cierto en todos los casos.

Ejemplo 3.3.5. Supongamos $f(x) = x$. La función $g(x, y) = x + \frac{(y-x)^2}{2}$ mayoriza f. Es fácil comprobar que $g(x, y) \geq f(x)$ y que $g(x, x) = f(x)$.

Pero

$$\underset{y}{argmin}\, g(x,y) = x = f(x)$$

Luego,

$$x^{(n+1)} = \underset{x}{argmin}\, g(x, x^{(n)}) = x^{(n)}$$

La iteración converge a un punto que no es ningún mínimo local, pues la función no tiene.

Vamos a demostrar a continuación que, definiendo la función auxiliar $g(x,y)$ adecuada para $\|X - WH\|^2$ y $-L(W,H)$, los algoritmos anteriormente expuestos pertenecen a la clase de algoritmos de minorización/mayorización. De esta forma, tendremos que los pasos de actualización van mejorando el resultado con cada actualización.

En el caso de la distancia obtenida a partir de la norma de Frobenius, la descompondremos por componentes de forma que nos encontremos ante una maximización de una suma de términos

$$\|X - WH\|^2 = \sum_{j=1}^{m} \|x_j - Wh_j\|^2,$$

donde cada h_j es una fila de la matriz H.

Lema 3.3.6. Si $K(h^{(n)})$ es la matriz diagonal

$$K_{a,b}(h^{(n)}) = \delta_{a,b}(W^T W h^{(n)})_a / h_a^{(n)}.$$

Entonces, la función

$$g(h, h^{(n)}) = f(h^{(n)}) + (h - h^{(n)})^T \nabla f(h^{(n)}) + \frac{1}{2}(h - h^{(n)})^T K(h^{(n)})(h - h^{(n)}) \tag{3.11}$$

mayoriza

$$f(h) = \frac{1}{2}\sum_i (x_i - \sum_j W_{i,j} h_j)^2.$$

Demostración. Como $g(h,h) = f(h)$ es obvio, sólo necesitamos probar que $g(h, h^{(n)}) \geq f(h)$ para tener las condiciones de la definición 3.3.3. Para hacer esto, comparamos

$$f(h) = f(h^{(n)}) + (h - h^{(n)})^T \nabla f(h^{(n)}) + \frac{1}{2}(h - h^{(n)})^T (W^T W)(h - h^{(n)})$$

con la función (3.11) para ver que $g(h, h^{(n)}) \geq f(h)$ es equivalente a

$$0 \leq (h - h^{(n)})^T [K(h^{(n)}) - W^T W](h - h^{(n)})$$

Para probar que es semidefinida positiva, consideramos la matriz

$$M_{i,j}(h^{(n)}) = h_i^{(n)}(K(h^{(n)}) - W^T W)_{i,j} h_j^{(n)},$$

que es simplemente un cambio de escala de las componentes de $K - W^T W$.

Entonces, $K - W^T W$ es semidefinida positiva si y sólo si M lo es. Es decir, si para todo vector v se tiene que $v^T M v \geq 0$.

$$\begin{aligned} v^T M v &= \sum_{i,j} v_i M_{i,j} v_j \\ &= \sum_{i,j} h_i^{(n)} (W^T W)_{i,j} h_j^{(n)} v_i^2 - v_i h_i^{(n)} (W^T W)_{i,j} h_j^{(n)} v_j \\ &= \sum_{i,j} (W^T W)_{i,j} h_i^{(n)} h_j^{(n)} \left[\frac{1}{2} v_i^2 + \frac{1}{2} v_j^2 - v_i v_j \right] \\ &= \frac{1}{2} \sum_{i,j} (W^T W)_{i,j} h_i^{(n)} h_j^{(n)} (v_i - v_j) \geq 0. \end{aligned} \tag{3.12}$$

□

La función g es una forma cuadrática definida positiva, convexa, luego alcanza su máximo donde el gradiente se anula.

$$\nabla f(h) = \nabla f(h^{(n)}) + K(h^{(n)})(h - h^{(n)}) = 0.$$

Esto ocurre si y sólo si

$$K(h^{(n)})(h - h^T) = -\nabla f(h^{(n)}).$$

Luego

$$h^{(n+1)} = h^{(n)} - K(h^{(n)})^{-1} \nabla f(h^{(n)}).$$

Dado que g mayoriza f, tenemos que f es decreciente para esa sucesión, según el lema 3.3.4. Escribiendo las componentes de esta ecuación explícitamente, obtenemos

$$h_i^{(n+1)} = h_i^{(n)} \frac{(W^T x)_i}{(W^T W h^{(n)})_i}.$$

Utilizando un mismo razonamiento, podemos mostrar que f es decreciente para los pasos de actualización de W.

A continuación, vamos a utilizar el lema 3.3.4 y la definición 3.3.3 para analizar el algoritmo propuesto para la función obtenida a partir de la divergencia de Kullback-Leibler.

Lema 3.3.7. Ignorando constantes, la función

$$g(W,H|W^{(n)},H^{(n)}) = \sum_{i=1}^{N}\sum_{j=1}^{p}\sum_{k=1}^{r} w_{ik}h_{kj} - \sum_{i=1}^{N}\sum_{j=1}^{p}\sum_{k=1}^{r} x_{ij}\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\left(\log w_{ik} + \log h_{kj}\right)$$

mayoriza $-L(W,H)$.

Demostración. Vamos a probar las condiciones descritas en la definición 3.3.3 para ver que, ignorando constantes, este resultado es cierto.

En primer lugar, veamos que $g(W,H|W^{(n)},H^{(n)}) \geq L(W,H)$.

Podemos mostrar que para cualquier conjunto de r valores $y_k \geq 0$ y $0 \leq c_k \leq 1$ con $\sum_{k=1}^{r} c_k = 1$ tenemos que

$$-\log\left(\sum_{k=1}^{r} y_k\right) \leq -\sum_{k=1}^{r} c_k \log(y_k/c_k)$$

usando la concavidad de $\log(x)$.

De ahí obtenemos que

$$-\log\left(\sum_{k=1}^{r} w_{ik}h_{kj}\right) \leq -\sum_{k=1}^{r} \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}} \log\left(\frac{b_{ij}^{(n)}}{a_{ikj}^{(n)}} w_{ik}h_{kj}\right)$$

donde $a_{ikj}^{(n)} = w_{ik}^{(n)}h_{kj}^{(n)}$ y $b_{ij}^{(n)} = \sum_{k=1}^{r} w_{ik}^{(n)}h_{kj}^{(n)}$ y n indica la iteración actual.

Podemos ver que se sigue cumpliendo la desigualdad anterior con los valores $a_{ikj}^{(n)}$ y $b_{ij}^{(n)}$ definidos de esa forma, puesto que

$$\sum_{k=1}^{r} \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}} = \frac{\sum_{k=1}^{r} w_{ik}h_{kj}}{\sum_{k=1}^{r} w_{ik}h_{kj}} = 1.$$

Entonces, tenemos que

$$\begin{aligned}
-L(W,H) &= \sum_{i=1}^{N}\sum_{j=1}^{p}\left[\sum_{k=1}^{r} w_{ik}h_{kj} - x_{ij}\log\left(\sum_{k=1}^{r} w_{ik}h_{kj}\right)\right] \\
&\leq \sum_{i=1}^{N}\sum_{j=1}^{p}\sum_{k=1}^{r} w_{ik}h_{kj} - \sum_{i=1}^{N}\sum_{j=1}^{p}\sum_{k=1}^{r} x_{ij}\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\log\left(\frac{b_{ij}^{(n)}}{a_{ikj}^{(n)}} w_{ik}h_{kj}\right) \\
&= g(W,H|W^{(n)},H^{(n)}) - \sum_{i=1}^{N}\sum_{j=1}^{p} x_{ij}\sum_{k=1}^{r}\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\log\left(\frac{b_{ij}^{(n)}}{a_{ikj}^{(n)}}\right) \\
&\leq g(W,H|W^{(n)},H^{(n)}).
\end{aligned}$$

Esta última desigualdad viene dada por la positividad de los valores x_{ij} y por el siguiente razonamiento.

$$\begin{aligned}\sum_{k=1}^{r} \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}} \log\left(\frac{b_{ij}^{(n)}}{a_{ikj}^{(n)}}\right) &= -\sum_{k=1}^{r} \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}} \log\left(\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\right) \\ &\geq -\log\left(\sum_{k=1}^{r}\left(\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\right)^2\right) \\ &\geq 0.\end{aligned}$$

Puesto que $\sum_{k=1}^{r}\left(\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\right) = 1 \Rightarrow 0 \leq \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}} \leq 1 \Rightarrow \left(\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\right)^2 \leq \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}} \Rightarrow \sum_{k=1}^{r}\left(\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\right)^2 \leq 1 \Rightarrow \log\left(\sum_{k=1}^{r}\left(\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\right)^2\right) \leq 0.$

Por otro lado, tenemos que probar la segunda condición de la definición 3.3.3. Es decir, hay que ver que $g(W, H|W, H) = -L(W, H)$. En este caso, lo demostraremos utilizando los valores $a_{ikj} = w_{ik}h_{kj}$ y $b_{ij} = \sum_{k=1}^{r} w_{ik}h_{kj}$ en la definición de la función g. Esto es,

$$\begin{aligned}g(W, H|W, H) = &\sum_{i=1}^{N}\sum_{j=1}^{p}\sum_{k=1}^{r} w_{ik}h_{kj} \\ &- \sum_{i=1}^{N}\sum_{j=1}^{p}\sum_{k=1}^{r} x_{ij}\frac{w_{ik}h_{kj}}{\sum_{k=1}^{r} w_{ik}h_{kj}} \log(w_{ik}h_{kj}).\end{aligned}$$

Entonces, tenemos que

$$g(W, H|W, H) = -L(W, H)$$

si, y sólo si,

$$\sum_{i=1}^{N}\sum_{j=1}^{p}\sum_{k=1}^{r} x_{ij}\frac{a_{ikj}}{b_{ij}} \log(a_{ikj}) = \sum_{i=1}^{N}\sum_{j=1}^{p} x_{ij}\log(b_{ij}).$$

Definimos la función $\tilde{g}$ de la siguiente manera

$$\tilde{g}(W, H|W^{(n)}, H^{(n)}) = g(W, H|W^{(n)}, H^{(n)}) - \sum_{i=1}^{N}\sum_{j=1}^{p} x_{ij}\sum_{k=1}^{r}\frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\log\left(\frac{b_{ij}^{(n)}}{a_{ikj}^{(n)}}\right)$$

Nótese que la diferencia entre las dos funciones es un término que no depende de W y H.

De esta forma hemos llegado a que

a) $\tilde{g}(W, H|W^{(n)}, H^{(n)}) \geq -L(W, H)$

b) $\tilde{g}(W, H|W, H) = -L(W, H)$

Y, entonces, por a) y b) tenemos que $\tilde{g}$ mayoriza $-L$. Luego, salvo constantes, g mayoriza $-L$. □

Deduciremos los pasos de actualización igualando las derivadas parciales de $g(W, H|W, H)$ a cero.

Partimos de $W^{(0)}, H^{(0)}$. Dado $W^{(n)}, H^{(n)}$

$$(W^{(n+1)}, H^{(n+1)}) = \underset{W,H}{argmin}\, \tilde{g}(W, H|W^{(n)}, H^{(n)}) = \underset{W,H}{argmin}\, g(W, H|W^{(n)}, H^{(n)})$$

$$\frac{\partial g(W, H|W^{(n)}, H^{(n)})}{\partial w_{ik}} = \left(\sum_{j=1}^{p} x_{ij} \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\right) \frac{1}{w_{ik}} - \sum_{j=1}^{p} h_{kj}$$

$$\frac{\partial g(W, H|W^{(n)}, H^{(n)})}{\partial h_{kj}} = \left(\sum_{j=1}^{p} x_{ij} \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}\right) \frac{1}{h_{kj}} - \sum_{i=1}^{N} w_{ik}$$

Igualando a cero y despejando

$$w_{ik} = \frac{\sum_{j=1}^{p} x_{ij} \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}}{\sum_{j=1}^{p} h_{kj}} = \frac{\sum_{j=1}^{p} x_{ij} \frac{w_{ik}^{(n)} h_{kj}^{(n)}}{\sum_{k=1}^{r} w_{ik}^{(n)} h_{kj}^{(n)}}}{\sum_{j=1}^{p} h_{kj}} = \frac{w_{ik}^{(n)} \sum_{j=1}^{p} x_{ij} h_{kj}^{(n)} / (W^{(n)} H^{(n)})_{ij}}{\sum_{j=1}^{p} h_{kj}}$$

$$h_{ik} = \frac{\sum_{i=1}^{N} x_{ij} \frac{a_{ikj}^{(n)}}{b_{ij}^{(n)}}}{\sum_{i=1}^{N} w_{ik}} = \frac{\sum_{i=1}^{N} x_{ij} \frac{w_{ik}^{(n)} h_{kj}^{(n)}}{\sum_{k=1}^{r} w_{ik}^{(n)} h_{kj}^{(n)}}}{\sum_{i=1}^{N} w_{ik}} = \frac{h_{kj}^{(n)} \sum_{i=1}^{N} x_{ij} w_{ik}^{(n)} / (W^{(n)} H^{(n)})_{ij}}{\sum_{i=1}^{N} w_{ik}}$$

Por tanto, los pasos de actualización quedan de la siguiente forma:

$$w_{ik}^{(n+1)} = \frac{w_{ik}^{(n)} \sum_{j=1}^{p} x_{ij} h_{kj}^{(n)} / (W^{(n)} H^{(n)})_{ij}}{\sum_{j=1}^{p} h_{kj}^{(n)}}$$

$$h_{ik}^{(n+1)} = \frac{h_{kj}^{(n)} \sum_{i=1}^{N} x_{ij} w_{ik}^{(n)} / (W^{(n)} H^{(n)})_{ij}}{\sum_{i=1}^{N} w_{ik}^{(n)}}$$

Esto demuestra que este es un algoritmo de mayorización-minimización y garantiza que la función va mejorando con cada iteración. Sin embargo, esto no garantiza que estemos llegando al mínimo global. De hecho, no está claro que haya un único minimizador. En la siguiente sección discutiremos que no podemos afirmar que exista una única factorización, es decir, un par (W, H) único que minimice los criterios anteriores, sin imponer unas condiciones muy estrictas. Típicamente, para probar la convergencia de un algoritmo de descenso al óptimo global, se prueba que la sucesión producida por la iteración es acotada y que los puntos límite son minimizadores del criterio. Entonces, si la función objetivo tiene un único minimizador, se concluye que la iteración es convergente. La unicidad del minimizador es casi un requisito necesario para probar la convergencia del algoritmo.

3.4. Interpretación geométrica del NMF

En esta sección nos plantearemos las hipótesis bajo las que está bien definida la noción de NMF. Es decir, discutiremos bajo qué circunstancias los algoritmos propuestos en [9] pueden llevar a la minimización de la función objetivo, estudiando si dicha factorización es única en algún sentido. Desarrollaremos un punto de vista geométrico tras el método NMF y, a partir de él, desarrollaremos condiciones bajo las que la factorización es esencialmente única para que NMF tenga sentido sea cual sea el algoritmo que está siendo empleado.

Para ello, se interpretará NMF como el problema de encontrar el cono simplicial que contiene una nube de puntos y que está contenido en el ortante positivo. Mostraremos que, bajo ciertas condiciones, básicamente pidiendo que algunos de los datos estén repartidos por las caras del ortante positivo, hay un único cono simplicial como el descrito anteriormente.

Como hemos dicho en la introducción de este capítulo, para esta sección nos basaremos especialmente en los razonamientos expuestos en el artículo [4], complementados con resultados del libro [11].

3.4.1. Análisis de unicidad de la factorización

Cada observación del conjunto de datos X (filas de la matriz) se puede interpretar como un punto en el espacio p-dimensional. El hecho de que los datos sean no negativos significa que cada uno de esos puntos se encuentra en el ortante positivo $\mathcal{P}$ de $\mathbb{R}^p$. La factorización exacta $X = WH$ es válida si y sólo si hay vectores h_j de $\mathbb{R}^p$ tales que todos los puntos x_i tienen una representación como combinaciones lineales no negativas de los h_j. Esta

caracterización algebraica tiene una equivalencia geométrica en la que nos vamos a basar para discutir la unicidad de la factorización.

Definición 3.4.1. El cono simplicial generado por los vectores $\Phi = (\phi_j)_{j=1}^r$ es

$$\Gamma = \Gamma_\Phi = \{\mathbf{x} : x = \sum_{j=1}^{r} \alpha_j \phi_j,\ \alpha_j \geq 0\}$$

La factorización $x^i = \sum_{j=1}^r w_j^i h_j$, con $w_j^i \geq 0$, nos dice geométricamente que todos los x^i se encuentran en el cono simplicial Γ_H generado por los h_j.

En general, para un conjunto de datos X dado, habrá muchos conos simpliciales posibles que contengan los puntos del conjunto de datos. De hecho, si Γ_H es un cono simplicial que contiene a los datos y Γ_Φ es otro cono que contiene al primero, es decir

$$\Gamma_H \subset \Gamma_\Phi,$$

entonces los correspondientes vectores ϕ_j también pueden dar una representación del conjunto de datos X. Por tanto, como para cualquier cono simplicial siempre puede haber otro cono que lo contiene estrictamente, tenemos que hay un número infinito de factorizaciones $X = WH$ con W no negativa y varias matrices H diferentes. De ahí que la restricción $W \geq 0$ no sea suficiente para llegar a una noción de NMF bien definida.

Ahora bien, si incluimos la restricción de positividad para H, geométricamente se traduce por pedir que el cono simplicial Γ_H esté contenido dentro del ortante positivo $\mathcal{P}$. ¿Podremos obtener unicidad añadiendo esta nueva restricción?

Veamos que no podremos si los valores de los datos son estrictamente positivos, tal que

$$x_{i,j} \geq \varepsilon > 0. \tag{3.13}$$

Interpretándolo geométricamente, esta condición implica que los puntos x_i estén contenidos estrictamente dentro del ortante positivo $\mathcal{P}$. Entonces vamos a demostrar que habrá muchos conos simpliciales que contienen los datos.

Por ejemplo, el propio $\mathcal{P}$ es un cono simplicial y contiene a los puntos. Sin embargo, veamos que existen otros muchos conos simpliciales que también los contienen. Dado $\delta > 0$, consideramos la colección de vectores (ϕ^δ) definidos como

$$\phi_j^\delta = e_j + \delta \cdot \mathbf{1},$$

donde e_j denota el vector usual de la base estándar y $\mathbf{1}$ denota el vector de unos. Entonces, tomando $\delta < \varepsilon$, el cono Γ_{ϕ^δ} también contiene todos los puntos. Geométricamente, lo que hacemos al construir Γ_{ϕ^δ} se podría entender

como una contracción del ortante $\mathcal{P}$ hacia la diagonal principal. Dado que la restricción (3.13) obliga a que todos los puntos estén contenidos estrictamente dentro del interior del ortante positivo, una pequeña contracción seguirá conteniendo los datos.

Por tanto, hemos llegado a que, bajo una condición de positividad estricta como la que tenemos en (3.13), habrá también muchas factorizaciones $X = WH$ distintas con $W \geq 0$ y $H \geq 0$. En resumen, para tener unicidad deberemos buscar situaciones donde los datos no obedezcan una positividad estricta.

A continuación vamos a introducir algunos conceptos de geometría convexa que nos serán de utilizad más adelante, a la hora de explicar un ejemplo de unicidad.

Definición 3.4.2. Decimos que $K \subset \mathbb{R}^n$ es un cono convexo si es cerrado para combinaciones lineales con coeficientes positivos.

Definición 3.4.3. Sea K un cono convexo no vacío, con $K \subset \mathbb{R}^n$. Llamamos polar de K al siguiente conjunto, denotado por K°

$$K^{\circ} = \{y \in \mathbb{R}^n : x \cdot y \leq 0, \forall x \in K\}.$$

Definición 3.4.4. Sea K un cono convexo no vacío, con $K \subset \mathbb{R}^n$. Llamamos dual de K al siguiente conjunto, denotado por K^*

$$K^* = \{y \in \mathbb{R}^n : x \cdot y \geq 0, \forall x \in K\}.$$

En la Figura 3.2 están representados el polar y el dual de K (en azul). K° está limitado por las líneas continuas negras y K^* está limitado por las líneas discontinuas.

Nota.

1. K^* es el opuesto de K°:

$$\begin{aligned} K^* &= \{y \in \mathbb{R}^n : x \cdot y \geq 0, \forall x \in K\} \\ &= \{y \in \mathbb{R}^n : x \cdot (-y) \leq 0, \forall x \in K\} = -(K^{\circ}) \end{aligned}$$

2. La dualidad invierte la inclusión de conjuntos:

$$B \subset C \Rightarrow C^* \subset B^*. \tag{3.14}$$

A continuación se enuncia y se demuestra el resultado principal sobre conos duales en relación con el objetivo de la sección.

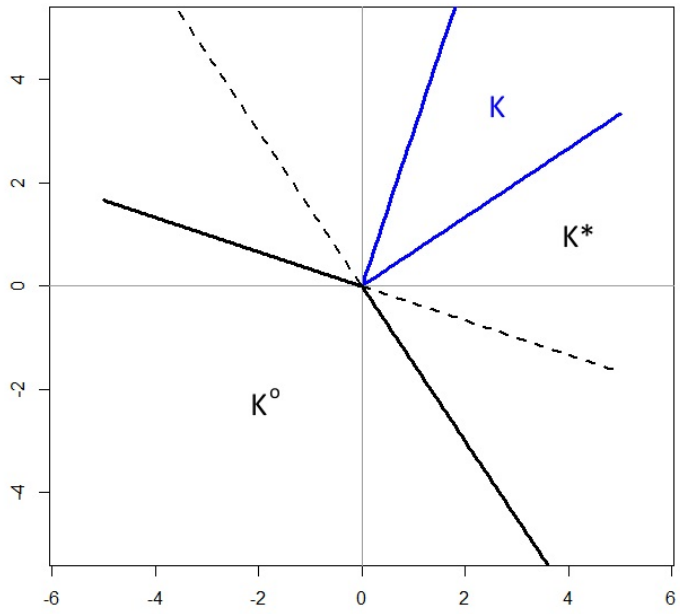

Figura 3.2: Representación del polar y el dual de K

Lema 3.4.5. Si K es un subconjunto de $\mathbb{R}^n$, entonces K^* es un cono convexo y cerrado. Si K es un cono convexo y cerrado, además se tiene que $(K^*)^* = K$.

Demostración. Para ver que es cerrado, supongamos que $\{y_n\}_{n=1}^{\infty}$ es una sucesión de puntos de K^* que converge a y. Si $x \in K$, entonces $y_n \cdot x \geq 0$. Como el producto escalar es una operación continua, tenemos que $y \cdot x \geq 0$, lo que implica que $y \in K^*$. Luego K^* es cerrado.

Debemos ver que si α, $\beta \geq 0$, para todo $y_1, y_2 \in K^*$ se tiene que $\alpha y_1 + \beta y_2 \in K^*$. Por definición de dual, sabemos que $\forall x \in K$, $y_1 \cdot x \geq 0$ $y_2 \cdot x \geq 0$. Luego $(\alpha y_1 + \beta y_2) \cdot x \geq 0$, lo que implica que $\alpha y_1 + \beta y_2 \in K^*$.

Vemos que es convexo probando $\alpha y_1 + (1-\alpha) y_2 \in K^*$, y esto es obvio, tomando $\alpha = \alpha$ y $\beta = 1 - \alpha$ en lo anterior.

Veamos que si K es un cono convexo y cerrado, además se tiene que $(K^*)^* = K$.

Si $x \in K$ entonces $x \cdot y \geq 0$ $\forall y \in K^*$. Luego tenemos que $x \in (K^*)^*$, así que ya hemos probado que $K \subset (K^*)^*$.

Ahora falta ver la otra inclusión para tener la igualdad. Para ello, comprobaremos que no hay ningún punto que está en $(K^*)^*$ y no está en K. Sea $z \in (K^*)^*$, supongamos que $z \notin K$ (Figura 3.3). Tenemos que

$$z \cdot y \geq 0 \quad \forall y \in K^* \Rightarrow \inf_{y \in K^*, \|y\|=1} z \cdot y = \min_{y \in K^*, \|y\|=1} z \cdot y.$$

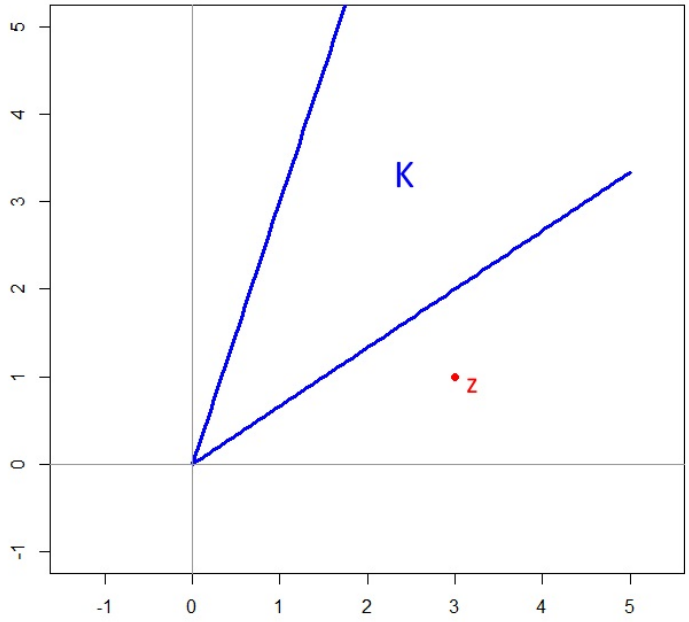

Figura 3.3: Situación de K y z

Si $z \notin K$, como K es cerrado, existe $\varepsilon > 0$ tal que $B(z, \varepsilon) \cap K = \emptyset$. Llamamos $L_{z,\varepsilon}$ al cono $L_{z,\varepsilon} = \{\lambda u, \lambda \geq 0, u \in B(z, \varepsilon)\}$. Entonces, tenemos que $K \cap L_{z,\varepsilon} = \{0\}$. Esto se debe a que, si hubiese otro punto $w \neq 0$, todo el rayo λw estaría contenido en esa intersección. Pero esto es absurdo, puesto que entonces cortaría a la bola y ya hemos visto que sus puntos no están en K.

Sean C_1 y C_2 conjuntos convexos no vacíos de $\mathbb{R}^n$. Para que exista un hiperplano separando C_1 y C_2 estrictamente, es necesario y suficiente que el interior relativo de C_1 y el interior relativo de C_2 no tengan puntos en común.

En nuestro caso, el interior relativo del punto z es el vacío y el cono no se corta con él.

Sean C_1 y C_2 subconjuntos no vacíos de $\mathbb{R}^n$ y al menos uno de ellos es un cono. Si existe un hiperplano que separa C_1 y C_2 estrictamente, entonces existe un hiperplano que separa C_1 y C_2 estrictamente y pasa por el origen.

El hiperplano que pasa por el origen es de la forma $\beta \cdot x = 0$. Supongamos que para los puntos $x_K \in K$ tenemos que $\beta \cdot x_K \geq 0$. Entonces $\beta \in K^*$, pero $\beta \cdot z < 0$. Luego hemos llegado a una contradicción, pues eso significaría que $z \notin (K^*)^*$. □

Enunciamos a continuación otros dos resultados que nos serán de utilidad más adelante y cuyas pruebas se pueden encontrar en [11].

- El dual de un cono simplicial con p generadores es un cono simplicial con p generadores.

- El ortante positivo es el dual de sí mismo $\mathcal{P}^* = \mathcal{P}$.

Definición 3.4.6. Dado un conjunto de puntos x_i, su envolvente cónica es la envolvente simplicial generada por los propios vectores x_i.

Sea $\mathcal{X}$ la envolvente cónica de un conjunto de puntos, otra forma de entender el problema de NMF es la siguiente:

(P1) Cono-simplicial-primal (r, $\mathcal{X}$): Encontrar un cono simplicial con r generadores contenido en $\mathcal{P}$ y conteniendo $\mathcal{X}$.

Consideramos ahora el problema en el dual con las inclusiones inversas.

(P2) Cono-simplicial-dual (r, $\mathcal{X}^*$): Encontrar un cono simplicial con r generadores contenido en $\mathcal{X}^*$ y conteniendo $\mathcal{P}$.

Los dos problemas son duales.

Lema 3.4.7. Toda solución al problema (P1) es dual a una solución de (P2) y viceversa.

Demostración. Lo demostraremos utilizando (3.14). Suponiendo que encontramos un cono simplicial Γ que cumple

$$\mathcal{X} \subset \Gamma \subset \mathcal{P}.$$

Luego, por 3.14,

$$\mathcal{P}^* \subset \Gamma^* \subset \mathcal{X}^*$$

y entonces una solución del primal es una solución del dual.

En la otra dirección, si encontramos un cono simplicial Γ^* que cumple

$$\mathcal{P}^* \subset \Gamma^* \subset \mathcal{X}^*$$

entonces, por 3.14,

$$(\mathcal{X}^*)^* \subset (\Gamma^*)^* \subset (\mathcal{P}^*).$$

Y aplicando $(K^*)^* = K$ tres veces llegamos a que una solución del dual corresponde a una solución del primal.

□

Definición 3.4.8. Un rayo extremo de un cono convexo Γ es un rayo $R_x = \{ax : a \geq 0\}$ donde $x \in \Gamma$ no puede ser representado como combinación convexa estricta de dos puntos x_0 y x_1 que pertenecen a Γ pero no a R_x.

Por ejemplo, un cono simplicial con r generadores linealmente independientes tiene r rayos extremos y cada rayo consiste en todos los múltiplos positivos de un generador.

Lema 3.4.9. Suponiendo que Γ y G son conos convexos , que $\Gamma \subset G \subset \mathbb{R}^r$, que Γ es un cono simplicial con r generadores y que G interseca a Γ en exactamente r rayos que son rayos extremos de G. Entonces:
(a) Esos rayos también son rayos extremos de Γ.
(b) Ningún cono simplicial con r generadores, $\Gamma' \neq \Gamma$, puede satisfacer $\Gamma \subset \Gamma' \subset G$.

Demostración. (a) Dado que los rayos en cuestión son rayos extremos de G, que contiene Γ, también son rayos extremos de Γ.
(b) Ningún cono simplicial Γ' con r generadores que esté "entre"Γ y G también podría intersecar G en los mismos r rayos que Γ. Estos r rayos también tendrían que ser rayos extremos para Γ', porque son rayos extremos para G, que contiene a Γ' por hipótesis. Pero un cono simplicial con r generadores está completamente determinado por sus r rayos extremos. Como Γ y Γ' tienen los mismos rayos extremos, $\Gamma = \Gamma'$. □

Ahora estamos en condiciones de dar un ejemplo de unicidad.
Sea C el cono definido de la siguiente manera:

$$C = \{x : x \cdot 1 \geq \sqrt{p-1}\|x\|\} \tag{3.15}$$

donde p es la dimensión del espacio de datos.

Lema 3.4.10. Hay un único cono simplicial que contiene a C y está contenido en el ortante positivo.

Nota. De hecho, ese único cono es $\mathcal{P}$. Ningún cono simplicial contenido dentro de $\mathcal{P}$ contiene todo C.

El dual del cono definido en 3.15 es:

$$C^* = \{y : y \cdot 1 \geq \|y\|\}. \tag{3.16}$$

Nótese que:
(a) Cada rayo frontera de C^* es extremo.
(b) C^* interseca $\mathcal{P}^*$ en los r vectores unidad e_j. Entonces, por el lema 3.4.9, $\mathcal{P}^*$ resuelve de forma única el problema del cono-simplicial-dual(n, C^*) y $\mathcal{P}$ es la solución única del problema del cono-simplicial-primal(n, C).

A continuación, se describe una familia de imágenes de unas características concretas que nos sirve como ejemplo de unicidad.

Definición 3.4.11. Una familia de articulación factorial separable es una colección X de puntos x que cumplen las siguientes condiciones:

[C1] Modelo generativo: Cada imagen x en la base de datos tiene una representación

$$x = \sum_{q=1}^{p} \sum_{a=1}^{A} \alpha_{q,a} \phi_{q,a},$$

donde los generadores $\phi_{q,a} \in \mathbb{R}^p$ cumplen la restricción de no negatividad $\phi_{q,a} \geq 0$ junto a los coeficientes $\alpha_{q,a} \geq 0$. Hablamos de $\phi_{q,a}$ como la q-ésima parte en la "a"-ésima articulación.

[C2] Separabilidad: para cada q, a existe un píxel $k_{q,a}$ tal que

$$\phi_{q',a'}(k_{q,a}) = 1_{\{a=a',q=q'\}}$$

Es decir, la presencia o ausencia de cada par de articulaciones en la imagen está indicado por un cierto píxel asociado a dicho par.

[C3] Muestreo factorial completo: El conjunto de datos contiene todas las A^p imágenes en las que los P puntos aparecen en todas las combinaciones de A articulaciones.

Teorema 3.4.12. Dada una base de datos que cumple C1, C2 y C3, hay una única envolvente simplicial con $r = AP$ generadores que contiene todos los puntos de la base de datos y está contenida en $\mathcal{P} \cap V$, donde V es el subespacio lineal generado por los r generadores.

Corolario 3.4.13. Sea X generado por las condiciones C1, C2 y C3. Cualquier factorización que cumpla $X = WH$, con W y H positivas, debe recuperar los generadores correctos $\phi_{q,a}$, salvo permutación de índices y reescala.

Prueba del teorema 3.4.12:

Demostración. Necesitamos introducir la noción de dualidad relativa a un espacio vectorial $V \subset \mathbb{R}^p$. En el caso de $V \equiv \mathbb{R}^p$ es simplemente la noción de dualidad introducida anteriormente. Suponemos que tenemos un conjunto $K \subset V$; su dual relativo K^V es el conjunto de vectores y que, vistos como miembros de $\mathbb{R}^p$, también pertenecen a V y que cumplen que $y \cdot x \geq 0$ para $x \in K$. El dual relativo es el dual ordinario tomado en V en vez de $\mathbb{R}^p$. En consecuencia, todas las propiedades vistas anteriormente se cumplen por dualidad relativa, dado que hablamos de conjuntos que son subconjuntos de V; por ejemplo, $(K^V)^V = K$ si K es un subconjunto cerrado y convexo de V.

Definimos $\mathcal{P}_V = V \cap \mathcal{P}$. Este es un cono simplicial en V con r generadores. Denotamos de nuevo por $\mathcal{X}$ la envolvente cónica de X y suponemos que cada cara de dimensión $r-1$ de $\mathcal{P}_V$ contiene $r-1$ puntos linealmente independientes de X. Dado que la cara de un cono es un subespacio lineal, la cara está determinada de forma única por estos $r-1$ puntos. La cara es

parte de un hiperplano de apoyo para $\mathcal{P}_V$ que también es un hiperplano de apoyo para $\mathcal{X}$. El hiperplano de apoyo define un punto $\xi \in V$ que está en común entre los duales $\mathcal{P}_V^V$ y $\mathcal{X}^V$. Algo similar se cumple para las r diferentes $(r-1)$-caras de $\mathcal{P}_V$. Por la independencia lineal mencionada anteriormente, los diferentes hiperplanos de apoyo en el espacio primal se corresponden con rayos extremos en el espacio dual, que son rayos extremos para $\mathcal{P}_V^V$ y $\mathcal{X}^V$. Como es cierto para las r caras de dimensión $(r-1)$, podemos aplicar el Lema 3.4.9 con $G = \mathcal{X}^V$ y $\Gamma = \mathcal{P}_V^V$.

Esto lleva a la conclusión de que $\mathcal{P}_V^V$ es el único cono simplicial con r generadores contenido en $\mathcal{X}^V$ y conteniendo $\mathcal{P}_V^V$. El teorema 3.4.12 se deduce por dualidad.

Falta probar la hipótesis de existencia de $r-1$ puntos linealmente independientes en cada $(r-1)$-cara. Las caras de $\mathcal{P}_V$ son exactamente los r subespacios diferentes

$$F_{q,a} = \{x \in V : \alpha_{q,a} = 0\}.$$

Por la hipótesis [C3], hay $A^{P-1}(A-1)$ puntos de X en una cara. Definimos para cada $(q', a') \neq (q, a)$,

$$\phi_{q',a';q,a} = \{x \in X : \alpha_{q,a} = 0, \alpha_{q',a'} = 1\}.$$

Hay $r-1$ términos de esta forma. Por la condición de separabilidad [C2]:

$$\phi_{q',a';q,a}(k_{q'',a''}) = 1_{\{a'=a'',q'=q''\}}.$$

De ahí que los $\phi_{q',a';q,a} : (q', a') \neq (q, a)$ sean linealmente independientes. Al mismo tiempo,

$$\phi_{q',a';q,a}(k_{q,a}) = 0$$

ya que cada $\phi_{q',a';q,a} \in F_{q,a}$. Por lo que tenemos la independencia lineal deseada en cada cara. □

El significado del teorema 3.4.12 y del corolario 3.4.13 es que se puede garantizar la unicidad bajo ciertas condiciones en la factorización no negativa. En el artículo [4] se muestra un ejemplo interesante de una base de datos de imágenes sintéticas en las que esta unicidad está garantizada. Sin embargo, es muy difícil pensar que las imágenes reales se ajustan a las condiciones impuestas por estos resultados. La consecuencia principal es que no está garantizada la convergencia de los algoritmos de NMF propuestos en [9].

3.5. Ejemplo

Se ha aplicado el método de factorización no negativa de matrices al conjunto de imágenes analizado en los capítulos anteriores. Para ello, se ha utilizado la función *nmf* que pertenece a la librería *NMF* de R. Esta función trabaja por defecto con el algoritmo basado en la divergencia de Kullback-Leibler que hemos descrito anteriormente. De todas formas, también hemos utilizado el algoritmo basado en la norma de Frobenius para comprobar que los resultados a los que llegamos son similares. A continuación, veremos cómo son los prototipos que nos proporciona este método dependiendo de la dimensión elegida, cómo son las reconstrucciones de las imágenes, sus errores de reconstrucción y cuánto ocupa la información comprimida. De esta manera, finalmente podremos compararlo con los resultados obtenidos en el caso del ACP y VQ, ya estudiados en los capítulos anteriores.

Para comenzar, se ha fijado la dimensión $r = 49$ para poder crear unas cuadrículas 7x7 con los 49 prototipos obtenidos igual que hicimos con los métodos vistos anteriormente. Dichas cuadrículas son las representadas en las Figuras 3.4 y 3.5. La primera de ellas muestra las imágenes base que conseguimos al utilizar el algoritmo basado en la divergencia de Kullback-Leibler y la segunda se corresponde con lo que obtenemos al pedir que el algoritmo de cálculo sea el basado en la norma de Frobenius.

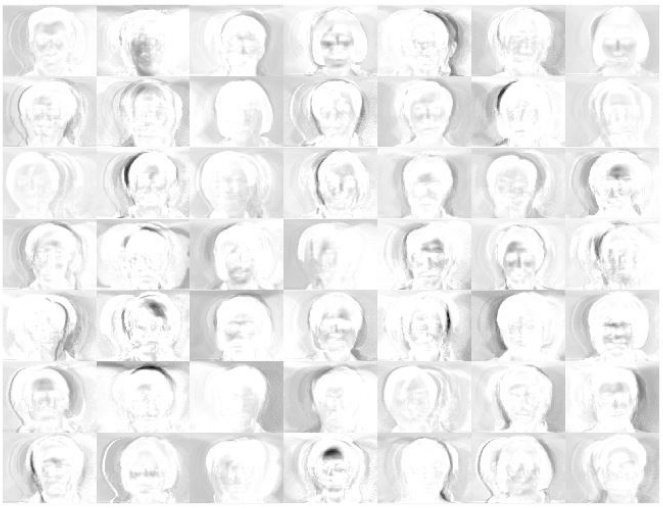

Figura 3.4: Representación de los prototipos de H (Kullback)

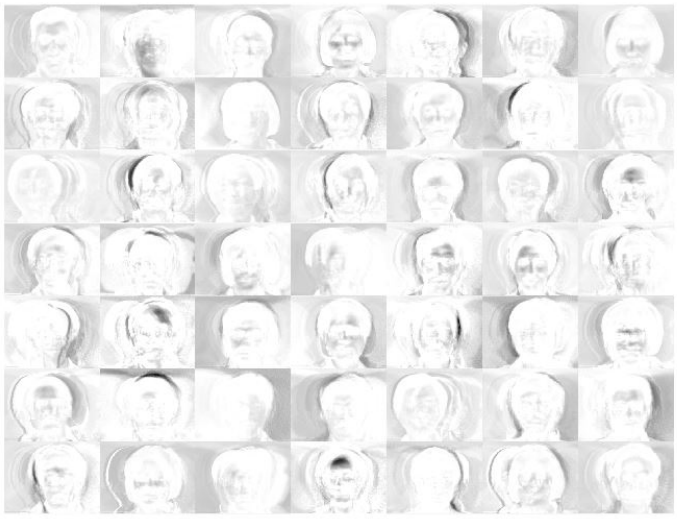

Figura 3.5: Representación de los prototipos de H (Frobenius)

Podemos observar que los prototipos obtenidos en ambos casos son similares y que, a simple vista, no conseguimos distinguir una diferencia clara. Más adelante veremos si conseguimos encontrar algunos puntos a favor o en contra de un algoritmo frente al otro.

En el caso de NMF, al igual que ocurría en el método de k-medias, los prototipos cambian dependiendo de la dimensión pedida. Esto se puede ver en la Figura 3.6, donde tenemos una cuadrícula 5x5 para representar los prototipos que conseguimos en dimensión 25 con el algoritmo basado en la divergencia de Kullback-Leibler. Esto es un inconveniente frente al ACP, puesto que ya hemos comentado que no es necesario repetir los cálculos si queremos una compresión mayor de la información.

También podemos concluir que no es tan evidente lo que afirman Lee y Seung en el artículo [10], puesto que no es fácil distinguir que estos prototipos estén formados por partes de caras con una interpretación sencilla. Por este motivo es importante continuar el análisis, discutiendo cómo es la calidad de la representación en una dimensión menor. De esta forma, podremos valorar correctamente sus ventajas e inconvenientes frente a los otros métodos, teniendo en cuenta que ya hemos visto que sus prototipos no son tan interpretables como esperábamos a la vista de dicho artículo.

Entonces, vamos a continuar viendo cómo es la reconstrucción de las dos imágenes que ya hemos analizado en ACP y VQ. Al igual que en los casos anteriores, hemos creado las Figuras 3.7 y 3.8 para representar sus aproxi-

maciones en dimensiones 25, 50, 75 y 100, y hemos colocado las fotografías originales a la derecha para poder compararlas. La reconstrucción de la primera figura está hecha a partir de los datos obtenidos con el algoritmo basado en la divergencia de Kullback-Leibler y, en el segundo caso, representamos los resultados del algoritmo de la norma de Frobenius.

Figura 3.6: Representación de los prototipos de H (dimensión 25)

Gracias a esta figura, podemos ver que la calidad de la reconstrucción, a simple vista, parece menor que en los otros métodos. Sin embargo, tenemos que esperar a ver los datos numéricos respecto a los errores. Para ello, hemos creado la misma gráfica que utilizamos en las secciones anteriores, donde se representa el error medido a partir de la distancia Euclídea, con el cuadrado de la norma de la diferencia entre la matriz original y las matrices aproximadas creadas con factores de varias dimensiones (hasta 100). Esta información se encuentra en la Figura 3.9 y corresponde con los la factorización que nos ha dado el algoritmo de Kullback-Leibler.

Figura 3.7: Comparación de dos caras aproximadas y las originales (Kullback)

En la siguiente tabla se recoge un resumen de los errores relativos exactos para las dimensiones 25, 50, 75 y 100 según las aproximaciones obtenidas con el algoritmo basado en la divergencia de Kullback y las aproximaciones obtenidas con el algoritmo basado en la norma de Frobenius.

Dimensión	25	50	75	100
Error Kullback	0.126	0.098	0.081	0.067
Error Frobenius	0.121	0.094	0.077	0.066

Comparando entre estos dos algoritmos, podemos observar que el que se basa en la norma de Frobenius nos proporciona unas aproximaciones ligeramente más acertadas. De todas formas, en ambos casos llegamos a tener unos errores de reconstrucción mayores que en el caso del ACP y, para dimensión 100, también son mayores que los obtenidos con el método de k-medias.

Figura 3.8: Comparación de dos caras aproximadas y las originales (Frobenius)

A la hora de comparar los métodos según la cantidad de información que debemos almacenar, nos encontramos con la misma compresión en este caso y en el del ACP. Esto se debe a que, a diferencia del método de k-medias, aquí sí que debemos guardar también las matrices W y H completas. Hay que señalar que los datos que se muestran en la siguiente tabla se corresponden tanto con el algoritmo de la divergencia de Kullback como con el algoritmo de la norma de Frobenius. No es de extrañar, puesto que en ambos casos tenemos unos factores matriciales con las mismas dimensiones.

	25	50	75	100
W	34 Kb	66.2 Kb	98.5 Kb	130.7 Kb
H	14.8 Mb	29.7 Mb	44.5 Mb	59.3 Mb

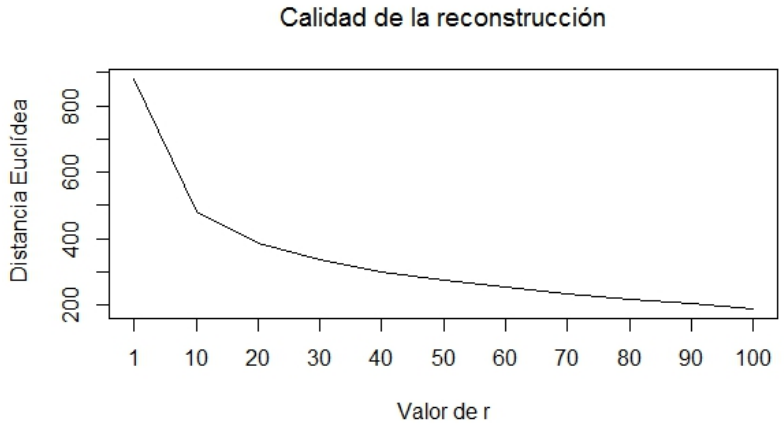

Figura 3.9: Gráfico del error de reconstrucción

Tras haber analizado los tres métodos propuestos en este trabajo aplicados a un mismo ejemplo, podemos extraer diversas conclusiones finales:

Pese a que en el ACP no tenemos unos prototipos interpretables, es uno de los métodos con más ventajas. En primer lugar, no se tienen que repetir los cálculos para hallar la factorización en otra dimensión. Esto es de gran ayuda si se quieren obtener y comparar los resultados obtenidos para varias aproximaciones. Además, el ACP tiene el objetivo de minimizar el error de reconstrucción, luego es evidente que sus resultados serán los más óptimos en este sentido. Esto es algo que hemos podido comprobar echando un vistazo a las reconstrucciones de dos de las imágenes del ejemplo para varias dimensiones.

En el caso del método VQ, basado en las k-medias, hemos visto que las reconstrucciones también son buenas, pero el error es mayor en general. Esto es obvio, dado que con este procedimiento imponemos unas restricciones más fuertes que en ACP. Sin embargo, la ventaja más destacable es que nos proporciona la mayor compresión de información, al no necesitar almacenar las dos matrices W y H completas, sino una matriz de prototipos H y un vector de asignaciones.

Finalmente, al estudiar el método de NMF y aplicarlo al ejemplo utilizado, hemos visto que los prototipos no son tan interpretables como se decía en el artículo [10], pues no se distinguen a simple vista partes de caras. Sin embargo, no se puede negar que en el caso de las imágenes parece más natural entender la descomposición con factores positivos, puesto que tiene más sentido pensar en mezclas aditivas.

Por otro lado, como es evidente, los errores son mayores y la reconstrucción es peor, dado que en este caso se imponen restricciones muy fuertes de no negatividad con el objetivo de ganar en interpretabilidad. Esto nos lleva a tener un menor nivel de compresión para obtener una calidad de reconstrucción similar al ACP, por ejemplo.

Además, como hemos visto, la factorización no es única en cualquier caso.

Conclusión

A lo largo de este trabajo se han analizado diferentes formas de conseguir una factorización aproximada para una matriz X, de dimensión $n \times p$, de forma que

$$X \approx WH,$$

donde W y H tienen dimensión $n \times r$ y $r \times p$ respectivamente. Los diferentes métodos expuestos se basan en la minimización de un cierto criterio, atendiendo a diversas restricciones, y tienen como objetivo comprimir los datos, explicando la mayor cantidad de información posible al disminuir el valor de r.

Inicialmente, hemos estudiado el análisis en componentes principales, cuyo objetivo es la minimización del error de reconstrucción con aproximaciones de menor dimensión. Hemos visto que este método se basa en la búsqueda de espacios ortogonales sobre los que proyectar los datos y que, en la práctica, esto se corresponde con el uso de los autovectores de la matriz de covarianzas $\Sigma = X^T X$ para construir la matriz H.

Hemos llegado a la conclusión de que las ventajas del ACP son numerosas. Destaca principalmente por ser el que busca recoger la mayor cantidad de variabilidad, luego sus aproximaciones son las de mayor calidad. Además, disponemos de métodos como la descomposición en valores singulares, que nos proporcionan maneras computacionalmente eficientes de calcularlo. Por otro lado, hemos visto que no es necesario repetir las operaciones si deseamos obtener los resultados para distintas dimensiones, pues al encontrar los autovectores conseguimos la información necesaria para las factorizaciones con cualquier valor de r.

No obstante, es cierto que los prototipos que conseguimos mediante este método carecen de una interpretación sencilla. Como se criticaba en el artículo [10], se trata de *autocaras* que representan siluetas de caras completas y de las que no se puede extraer mayor información.

Más adelante hemos estudiado la cuantización vectorial, un método que se basa en el análisis cluster y las k-medias. Su objetivo es la partición de los datos en varios grupos, de forma que se minimicen las distancias dentro de ellos. Gracias a este método, conseguimos una reconstrucción a base de prototipos que son promedios de los datos más *parecidos* (los que pertenecen a un mismo cluster).

Evidentemente, VQ nos permite obtener unas aproximaciones de menor calidad que el ACP. Sin embargo, destaca especialmente en la compresión de información, pues nos basta con almacenar la matriz H y un vector de *etiquetas* que indique con qué prototipo o k-media se corresponde cada dato original.

También hemos hablado de una variante de este último método que está a medio camino entre VQ y NMF: el análisis de arquetipos. Donde tenemos prototipos que son ellos mismos una combinación convexa de los datos originales.

Finalmente, hemos visto otras posibilidades (los algoritmos de NMF), que parecían añadir interpretabilidad a costa de imponer una restricción más estricta. Dicha restricción se trataba de la no negatividad de los factores matriciales que utilizamos en la aproximación a una matriz no negativa. Esta idea tiene sentido cuando estamos hablando de compresión de imágenes, como es el caso del ejemplo utilizado, puesto que nos encontramos ante una matriz X de valores no negativos (las intensidades de gris) y su descomposición en W y H puede ser pensada como una combinación aditiva de partes de imágenes para formar las originales. No obstante, hemos visto que, con nuestra base de datos, los prototipos no son tan interpretables como esperábamos a la vista de lo expuesto en el artículo que ha servido de motivación a este trabajo (referencia [10]).

Además, gracias al estudio de su interpretación geométrica, hemos podido comprobar que las factorizaciones obtenidas a partir de los algoritmos de NMF no son únicas en cualquier caso, sino que se requieren unas condiciones muy concretas para que esté bien definido.

Bibliografía

[1] Belhumeur, P.N., Hespanha, J.P. and Kriegman, D.J. - *Eigenfaces vs. Fisherfaces: recognition using class specific linear projection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 19, pág.711-720, 1997.

[2] Cutler, A. and Breiman, L. - *Archetypal Analysis*, Technometrics, 36, pág.338-347, 1994.

[3] Damle, A. and Sun, Y. - *A Geometric Approach to Archetypal Analysis and Nonnegative Matrix Factorization*, Technometrics, 59:3, pág.361-370, 2017.

[4] Donoho, D. and Stodden, V. - *When does non-negative matrix factorization give a correct descomposition into parts?*, Advances in Neural Information Processing Systems 16th conference (NIPS 2003), pág.1141-1148, 2004.

[5] Hastie, T., Tibshirani, R. and Friedman, J. - *The Elements of Statistical Learning*, 2ª Edición, Springer, 2009.

[6] Jain, A.K. and Dubes, R.C. - *Algorithms for Clustering Data*, Prentice-Hall, 1988.

[7] Jolliffe, I.T. - *Principal Component Analysis*, Springer, 1986.

[8] Lebart, L., Morineau, A. and Warwick, K. M. - *Multivariate Descriptive Statistical Analysis*, Wiley, 1984.

[9] Lee, D.D. and Seung, H.S. - *Algorithms for non-negative matrix factorization*, Advances in Neural Information Processing Systems 13th conference (NIPS 2000), pág.556-562, 2001.

[10] Lee, D.D. and Seung, H.S. - *Learning the parts of objects by non-negative matrix factorization*, Nature, Vol.401, pág.788-791, 1999.

[11] Rockafellar, R. T. - *Convex Analysis*, Princeton University Press, 1970.

[12] Trefethen, L.N. and Bau, D. III - *Numerical Linear Algebra*, SIAM, 1997.

Apéndice A

Códigos de R

En este apéndice se recogen algunos códigos que se han utilizado para las secciones de ejemplos.

- **Manipulación de imágenes:**

```
library(png)
temp = list.files(pattern="*.png")
myfiles = lapply(temp, readPNG)

#Construcción de la matriz de datos:
X<-as.vector(myfiles[[1]])
for (i in 2:165){
  vect<-as.vector(myfiles[[i]])
  X<-rbind(X,vect)
}
```

- **Cálculos para ACP (ejemplo en dimensión 50):**

```
svdX <- svd(X)
V<-svdX$v #Los autovectores obtenidos con SVD

#Reconstruir la matriz X aproximada en dimensión 50:
#Utilizando los 50 primeros autovectores
H50<-V[,1:50] #La matriz con los prototipos en las columnas (H traspuesta)
W50<-X%*%V50 #Matriz de coeficientes W
recX50acp<-W50%*%t(H50)

#Reconstrucción del primer prototipo:
library(pixmap)
```

```
Prot1<-V[,1]
dim(Prot1)<-c(243,320)
plot(pixmapGrey(Prot1))

#Reconstrucción de la primera imagen en dimensión 50:
library(pixmap)
recX50.1<-recX50acp[1,]
dim(recX50.1)<-c(243,320)
plot(pixmapGrey(recX50.1))

#Lo que ocupa la información:
print(object.size(recX50acp),units="Mb")
print(object.size(W50),units="Kb")
print(object.size(H50),units="Mb")

#Cálculo de la variabilidad recogida según los autovalores:
cal50<-sum(svdX$d[1:50])/sum(svdX$d)

#Error de reconstrucción con distancia Euclídea:
difXXacp<-sqrt(sum((recX50acp-X)^2))/sqrt(sum((X)^2))
```

- **Cálculos para VQ (ejemplo en dimensión 50):**

```
library(cluster)
kmediasX50<-kmeans(X, 50, iter.max = 500, nstart = 250)

#Reconstrucción de la matriz X aproximada en dimensión 50:
recX50<-kmediasX50$centers[kmediasX50$cluster[1],]
for (i in 2:165){
recX50<-rbind(recX50, kmediasX50$centers[kmediasX50$cluster[i],])
}

#Reconstrucción del primer prototipo:
library(pixmap)
Prot.1<-kmediasX50$centers[1,]
dim(Prot.1)<-c(243,320)
plot(pixmapGrey(Prot.1))

#Reconstrucción de la primera imagen:
library(pixmap)
X.rec50.1<-recX50[1,]
```

```
dim(X.rec50.1)<-c(243,320)
plot(pixmapGrey(X.rec50.1))

#Lo que ocupa la información:
print(object.size(kmediasX50$cluster),units="Kb")
print(object.size(kmediasX50$centers),units="Mb")

#Error de reconstrucción con distancia Euclídea:
difXXrec<-sqrt(sum((recX50-X)^2))/sqrt(sum((X)^2))
```

- **Cálculos para NMF (ejemplo en dimensión 50):**

```
X<-t(X) #Con estas dimensiones, hay que trasponer para que no dé error
library(NMF)
res50<-nmf(X,50)

#Reconstrucción de la matriz X aproximada en dimensión 50:
X.hat50<-fitted(res50)
dim(X.hat)
#Los prototipos se encuentran por columnas en "basis(res50)"
#La información de los coeficientes de W está en "coef(res50)"

#Reconstrucción del primer prototipo:
library(pixmap)
Prot.1.nmf<-basis(res50)[,1]
dim(Prot.1.nmf)<-c(243,320)
plot(pixmapGrey(Prot.1.nmf))

#Reconstrucción de la primera imagen:
library(pixmap)
X.hat.1<-X.hat50[,1]
dim(X.hat.1)<-c(243,320)
plot(pixmapGrey(X.hat.1))

#Lo que ocupa la información:
print(object.size(basis(res50)),units="Mb")
print(object.size(coef(res50)),units="Kb")

#Error de reconstrucción con distancia Euclídea:
difXXrec<-sqrt(sum((X.hat50-X)^2))/sqrt(sum((X)^2))
```