UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRIA EN CIENCIAS MENCIÓN OPTIMIZACIÓN

COMPLETACIÓN DE MATRICES VÍA ESTADÍSTICA BAYESIANA

Trabajo presentado para optar al grado de
Magister Scientiarum mención Optimización

Autor: Lcda. Anaís Frangeline Acuña Sosa
Tutor: Msc. Jhonny Escalona.

Barquisimeto - Venezuela
Septiembre, 2014.

Completación de matrices vía estadística Bayesiana

Lcda. Anaís Acuña

RESUMEN

En este trabajo estudiamos el problema de completación de matrices, es decir, la recuperación de una matriz de datos desde una muestra de sus entradas. Se plantea una propuesta para resolver este problema en el marco de la estadística Bayesiana, considerando diversas herramientas expuestas; un algoritmo EM para poblaciones normales, análisis de componentes principales y el diseño de una versión de aproximación estocástica del algoritmo EM (SAEM, por sus siglas en inglés), considerando el modelo de factorización probabilística de matrices. Mostraremos algunos resultados de la implementación del método propuesto para matrices generadas de forma aleatoria con entradas faltantes.

A mi abuela, Ana Josefa.

Índice general

Índice de tablas

Agradecimientos

Quiero comenzar agradeciendo a Dios por todas las oportunidades y a mis padres, quienes con esfuerzo y dedicación me han acompañado.

A mi abuela, Ana Josefa, por ser un pilar esencial en mi vida. A mi hermana, por apoyarme y brindarme su cariño.

A Jacobo, por ayudarme y por ser parte fundamental de este logro. A mi amiga Mariana Álvarez, quien desde la distancia siempre me apoyó.

A los profesores, Javier Hernández, Freddy Torrealba, Eibar Hernández, Rómulo Castillo, y de forma especial a Hugo Lara. Sus enseñanzas han sido invaluables.

A mi tutor, Jhonny Escalona, por su tiempo, dedicación y gran disposición en este trabajo; por brindarme su amistad y apoyo.

A los profesores Clavel Quintana y Alí Duin, por ofrecerme sus comentarios de ayuda.

A todos que de alguna u otra forma me ayudaron en la realización de este trabajo.

Introducción

El problema de completación de matrices se presenta en muchos problemas prácticos de interés en diversos campos como la teoría de sistemas y control; procesamiento de imágenes y filtrado colaborativo, entre otros. En este problema tenemos una matriz de datos $F \in \mathbb{R}^{N \times M}$, la cual queremos conocer con la mayor precisión posible, pero solo conocemos un subconjunto de sus entradas.

Ha sido demostrado en [5] y [16] que bajo algunas condiciones una solución del problema de rango mínimo puede ser encontrada a través de optimización convexa, minimizando la norma nuclear. Se han propuesto varios tipos de algoritmos para recuperar una solución de rango mínimo, pero muchos de estos comprenden la descomposición de valores singulares de la matriz, trayendo consigo un alto costo computacional.

Desde la perspectiva Bayesiana han surgido diversas propuestas, Salakhutdinov y Mnih en [14], proponen un modelo de factorización probabilística de matrices, en el cual utilizan un algoritmo basado en métodos Monte Carlo con cadenas de Markov (MCMC, por sus siglas en inglés) para aproximar la distribución predictiva que permite completar la matriz F, y aunque este método evita la descomposición de los valores singulares de la matriz y mostró tener buenos resultados para el problema de Netflix, es costoso computacionalmente.

Considerando el modelo de factorización probabilística de matrices propuesto

en [14], en este trabajo se propuso resolver problemas de completación de matrices, encontrando dos matrices $U \in \mathbb{R}^{D\times N}$ y $V \in \mathbb{R}^{D\times M}$ tales que: $F = U'V$. Se considera que las filas de la matriz F están formadas por muestras de un vector aleatorio proveniente de una distribución normal; esto permite a través de la implementación de un algoritmo EM para poblaciones normales y un análisis de componentes principales obtener la matriz $V \in \mathbb{R}^{D\times M}$ deseada. Para la estimación de la matriz $U \in \mathbb{R}^{D\times N}$ se diseña un algoritmo SAEM acoplado con un algoritmo MCMC, el cual es implementado para cada columna de esta matriz.

El trabajo se encuentra estructurado de la siguiente manera: en el capítulo 1, son presentadas definiciones y resultados útiles en su desarrollo; en el capítulo 2, se expone teoría de estimación por máxima verosimilitud, donde son mostrados el algoritmo EM y el algoritmo SAEM; en el capítulo 3, exponemos las técnicas empleadas por los algoritmos MCMC, presentamos una forma general del algoritmo Metropolis-Hastings y el muestreador de Gibbs; en el capítulo 4, planteamos el problema de completación de matrices, el modelo de factorización probabilística de matrices y formulamos nuestra propuesta. En el capítulo 5, se muestran resultados de la implementación del método propuesto y para dos tipos de problemas se exhiben comparaciones con uno o dos métodos diferentes. Por último, se presentan las conclusiones de este trabajo y se exponen las ventajas de utilizar el método propuesto, así como aspectos que pueden ser mejorados.

Capítulo 1

Preliminares

En este capítulo enunciaremos algunas definiciones y resultados que serán de utilidad en el desarrollo de este trabajo, estas pueden ser consultadas en [18].

1.1. Estadística vectorial

Definición 1.1.1 *Se dice que* $X = \left(X_1, X_2, ..., X_p\right)$ *es un p-vector aleatorio, si cada* X_i *con* $i \in \{1, 2, \ldots, p\}$ *es una variable aleatoria unidimensional o escalar.*

1.1.1. Esperanza y matriz de covarianza de vectores aleatorios

Definición 1.1.2 *Sea X un p-vector aleatorio,* $X = \left(X_1, X_2, ..., X_p\right)$*, se define el vector de esperanza* $\mathbb{E}(X)$ *como:*

$$\mathbb{E}(X) := \left(\mathbb{E}(X_1), \mathbb{E}(X_2), ..., \mathbb{E}(X_p)\right),$$

donde $\mathbb{E}(X_i)$ *con* $i \in \{1, \ldots, p\}$ *es la esperanza o valor esperado de la variable aleatoria unidimensional* X_i*.*

Propiedad 1.1.1 *Sea X un* $p-$*vector aleatorio y* $T : \mathbb{R}^p \to \mathbb{R}^q$ *una transformación lineal. Si el vector de esperanza de X existe, entonces existe el vector de esperanza*

del $q-$vector aleatorio TX, y además:

$$\mathbb{E}(TX) = T\mathbb{E}(X).$$

Nota 1.1.1 *Dada una transformación lineal $T : \mathbb{R}^p \to \mathbb{R}^q$ y $x \in \mathbb{R}^p$, escribiremos Tx en lugar de $T(x)$.*

Propiedad 1.1.2 *Sean X y Y p-vectores aleatorios cualesquiera. Si $\mathbb{E}(X)$ y $\mathbb{E}(Y)$ existen, entonces:*

- *Para todo $\alpha \in \mathbb{R}$, $\mathbb{E}(\alpha X)$ existe y además:*

$$\mathbb{E}(\alpha X) = \alpha \mathbb{E}(X).$$

- $\mathbb{E}(X + Y)$ *existe, y se cumple:*

$$\mathbb{E}(X + Y) = \mathbb{E}(X) + \mathbb{E}(Y).$$

- *Para todo $\boldsymbol{a} \in \mathbb{R}^p$, $\mathbb{E}(X + \boldsymbol{a})$ existe y se tiene que:*

$$\mathbb{E}(X + \boldsymbol{a}) = \mathbb{E}(X) + \boldsymbol{a}.$$

Definición 1.1.3 *Sea X un p-vector aleatorio, $X = \left(X_1, X_2, ..., X_p\right)$, se define la matriz de covarianza de X, como:*

$$cov(X) = \begin{pmatrix} cov(X_1, X_1) & \cdots & cov\left(X_1, X_p\right) \\ cov(X_2, X_1) & \cdots & cov\left(X_2, X_p\right) \\ \vdots & & \vdots \\ cov\left(X_p, X_1\right) & \cdots & cov\left(X_p, X_p\right) \end{pmatrix},$$

donde $cov\left(X_i, X_j\right) = \mathbb{E}\left[\left(X_i - \mathbb{E}(X_i)\right)\left(X_j - \mathbb{E}(X_j)\right)\right] = cov\left(X_j, X_i\right)$, para todo $i, j \in \{1, 2, ..., p\}$.

Observación 1.1.1 *Dado que* $cov\left(X_i, X_j\right) = cov\left(X_j, X_i\right)$*, para todo* $i, j \in \{1, 2, ..., p\}$*, entonces la matriz de covarianza para cualquier p-vector aleatorio X, es simétrica.*

Observación 1.1.2 *En la práctica cuando* $\sigma^2_{X_i} < \infty$ *y* $\sigma^2_{X_j} < \infty$*, es usada la propiedad:*

$$cov\left(X_i, X_j\right) = \mathbb{E}\left(X_i X_j\right) - \mathbb{E}\left(X_i\right)\mathbb{E}\left(X_j\right).$$

Definición 1.1.4 *Dada una transformación lineal* $T : \mathbb{R}^p \to \mathbb{R}^q$ *se define la matriz de dimensión* $q \times p$ *asociada a* T*, llamada la matriz de* T *y denotada por* $[T]$*, como:*

$$[T]_{ij} := \left(Te_j\right)_i \quad \forall i \in \{1, 2, \ldots, q\}, \forall j \in \{1, 2, \ldots, p\},$$

donde e_j *es el* $j-$*ésimo vector canónico de* $\mathbb{R}^p$ *y* $Te_j = \left((Te_j)_1, (Te_j)_2, \ldots, (Te_j)_q\right)$*,* $\forall j \in \{1, 2, \ldots, p\}$*.*

Nota 1.1.2 *Toda transformación lineal* $T : \mathbb{R}^p \to \mathbb{R}^q$ *posee una matriz* $q \times p$ *asociada. Recíprocamente, para toda matriz* M *de dimensión* $q \times p$ *existe una única transformación lineal* $T : \mathbb{R}^p \to \mathbb{R}^q$ *tal que* $[T] = M$*. Esto es, existe una correspondencia biyectiva entre las transformaciones lineal* $T : \mathbb{R}^p \to \mathbb{R}^q$ *y las matrices de dimensión* $q \times p$*.*

Definición 1.1.5 *La transformación lineal* $C(X) : \mathbb{R}^p \to \mathbb{R}^p$ *asociada a la matriz de covarianza* $cov(X)$ *de* $X = \left(X_1, X_2, ..., X_p\right)$*, es llamada el operador de covarianza de X.*

Proposición 1.1.1 *Si* $X = \left(X_1, X_2, ..., X_p\right)$ *es un* $p-$*vector aleatorio, entonces:*

(i) Las varianzas de todas las componentes $X_1, X_2, ..., X_p$ *existen si y solo sí, la varianza de* $\langle X, a\rangle$ *existe para todo* $a \in \mathbb{R}^p$*.*

(ii) Si las varianzas de todas las componentes $X_1, X_2, ..., X_p$ existen, entonces el operador de covarianza es la única transformación lineal $T : \mathbb{R}^p \to \mathbb{R}^q$ la cual satisface que:

$$cov\left(\langle X, a\rangle\langle X, b\rangle\right) = \langle Ta, b\rangle, \quad \forall a, b \in \mathbb{R}^p.$$

Nota 1.1.3 *$\langle\cdot,\cdot\rangle$ denotará el producto interno usual o estándar en $\mathbb{R}^p$, definido como:*

$$\langle x, y\rangle = x_1y_1 + \cdots + x_py_p, \quad \forall x, y \in \mathbb{R}^p.$$

Proposición 1.1.2 *El operador de covarianza $C(X)$ de cualquier $p-$vector aleatorio X es auto-adjunto y definido positivo.*

Nota 1.1.4

- *Para toda transformación lineal $T : \mathbb{R}^p \to \mathbb{R}^q$, existe una única transformación lineal $T^* : \mathbb{R}^q \to \mathbb{R}^p$ tal que:*

$$\langle Tx, y\rangle = \langle x, T^*y\rangle, \quad \forall x \in \mathbb{R}^p \ y \ \forall y \in \mathbb{R}^q.$$

- *Una transformación lineal $T : \mathbb{R}^p \to \mathbb{R}^p$ se dice auto-adjunta (o simétrica) si $T^* = T$. Por tanto, una transformación lineal es auto-adjunta si y solo sí su matriz asociada es simétrica.*

- *Una transformación lineal $T : \mathbb{R}^p \to \mathbb{R}^p$ se dice definida positiva si:*

$$\langle Tx, x\rangle \geq 0, \quad \forall x \in \mathbb{R}^p.$$

Observación 1.1.3 *De la proposición anterior, tenemos que la matriz de covarianza $cov(X)$ de cualquier $p-$vector aleatorio X, es simétrica y definida positiva.*

Proposición 1.1.3 *Si X un $p-$vector aleatorio cuyas componentes poseen varianza finita, entonces:*

(i) Para todo $a \in \mathbb{R}^p$, se tiene que:

$$C(X + a) = C(X).$$

(ii) Para toda transformación lineal $T : \mathbb{R}^p \to \mathbb{R}^q$ las componentes del $q-$vector aleatorio TX poseen varianza finita y:

$$C(TX) = TC(X)T^*.$$

Definición 1.1.6 *Se dice que un p-vector aleatorio $X = \left(X_1, X_2, ..., X_p\right)$ posee componentes incorreladas si $cov(X_i, X_j) = 0$, $\forall i \neq j$.*

Proposición 1.1.4 *Las siguientes proposiciones son equivalentes:*

(i) $X = \left(X_1, X_2, ..., X_p\right)$ posee componentes incorreladas.

(ii) Existen escalares $\lambda_1, \ldots, \lambda_p \geq 0$ tales que $C(X) = diag\left(\lambda_1, \ldots, \lambda_p\right)$, donde $diag\left(\lambda_1, \ldots, \lambda_p\right)$ denota el operador diagonal definido como:

$$diag\left(\lambda_1, \ldots, \lambda_p\right)\left(X_1, X_2, ..., X_p\right) = \left(\lambda_1 X_1, \ldots, \lambda_p X_p\right).$$

1.1.2. Muestras vectoriales

Definición 1.1.7 *Se dice que los vectores aleatorios $X_1, \ldots, X_n$ es una muestra vectorial (o muestra multivariada) de tamaño n, si constituyen un conjunto de vectores aleatorios estadísticamente independientes con una misma distribución de probabilidad. Esta distribución común, es llamada distribución de probabilidad de la población.*

Definición 1.1.8 *Dada una muestra vectorial $X_1, \ldots, X_n$, se define la media muestral como:*

$$\overline{X} = \frac{X_1 + \cdots + X_n}{n}.$$

Proposición 1.1.5 *Sea $X_1, \ldots, X_n$ una muestra vectorial de una población con media μ y operador de covarianza C, se tiene que:*

(i) $\mathbb{E}\left(\overline{X}\right) = \mu$,

(ii) $C(\overline{X}) = \frac{1}{n}C$.

Definición 1.1.9 *Dada una muestra vectorial $X_1, \ldots, X_n$, se define el operador de covarianza muestral como:*

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

donde para cada $x, y \in \mathbb{R}^p$ el operador lineal $x \otimes y : \mathbb{R}^p \to \mathbb{R}^p$, esta definido por:

$$(x \otimes y)(z) = \langle z, y\rangle x, \quad \forall z \in \mathbb{R}^p.$$

Proposición 1.1.6 *Si $X_1, \ldots, X_n$ es una muestra vectorial de una población con operador de covarianza C, entonces:*

$$\mathbb{E}\left(S^2\right) = C,$$

esto es, S^2 es un estimador insesgado de C.

1.1.3. Análisis de componentes principales

Un problema central en el análisis de datos multivariados es la reducción de la dimensionalidad: si es posible describir con precisión los valores de p variables aleatorias por un pequeño subconjunto $r < p$ de ellas, se habrá reducido la dimensión del problema a costa de una pequeña pérdida de información.

La técnica de componentes principales es debida a Hotelling (1933), aunque sus orígenes se encuentran en los ajustes ortogonales por mínimos cuadrados introducidos por K. Pearson (1901). Su utilidad es doble:

1. Permite representar óptimamente en un espacio de dimensión pequeña, observaciones de un espacio general p-dimensional. En este sentido componentes principales es el primer paso para identificar posibles variables no observadas, que están generando la variabilidad de los datos.

2. Permite transformar las variables originales, en general correladas, en nuevas variables incorreladas, facilitando la interpretación de los datos.

Sea $\left(X_1, X_2, ..., X_p\right)$ un vector aleatorio con operador de covarianza $C(X)$. Considerando una adecuada "rotación" de este, siempre es posible obtener un vector aleatorio con componentes incorreladas. Dado que el operador de covarianza $C(X)$ es auto-adjunto, existe una base ortonormal $\left\{V_1, \ldots, V_p\right\}$ en $\mathbb{R}^p$, la cual consiste de los vectores propios de $C(X)$. Denotemos por λ_i al auto-valor asociado a V_i, con $i \in \{1, 2, \ldots, p\}$.

Haciendo $Y_i = \langle X, V_i \rangle$, se obtiene que:

$$cov\left(Y_i, Y_j\right) = cov\left(\langle X, V_i\rangle, \langle X, V_j\rangle\right) = \langle C(X)V_i, V_j\rangle = \lambda_i \langle V_i, V_j\rangle = \begin{cases} 0, \text{para } i \neq j \\ \lambda_i, \text{para } i = j \end{cases}.$$

Por tanto, las variables $Y_1, \ldots, Y_p$ son incorrelacionas y $var(Y_i) = \lambda_i, \forall i \in \{1, \ldots, p\}$.

Definición 1.1.10 *Dado $\left\{v_1, \ldots, v_p\right\}$ un conjunto de vectores en $\mathbb{R}^p$, existe una única transformación lineal $T : \mathbb{R}^p \to \mathbb{R}^p$ tal que:*

$$Te_1 = v_1, \quad Te_2 = v_2, \ \ldots \ , Tep = v_p.$$

Así, decimos que T "convierte la base estándar en el conjunto de vectores $\left\{v_1, \ldots, v_p\right\}$".

Si los vectores $Te_1, \ldots, Te_p$ constituyen una base ortonormal de $\mathbb{R}^p$, entonces T se dice un operador lineal ortogonal.

Dada esta definición, si definimos $Y = \left(Y_1, \ldots, Y_p\right)$ y tomamos un operador lineal ortogonal Q que convierta la base $\left\{e_1, \ldots, e_p\right\}$ en $\left\{V_1, \ldots, V_p\right\}$, tenemos que:

$$\langle X, V_i\rangle = Y_i = \langle Y, e_i\rangle = \langle Y, Q^{-1}V_i\rangle = \langle QY, V_i\rangle.$$

Por tanto:

$$X = QY, \tag{1.1}$$

y entonces:

$$\begin{aligned} Y &= Q^{-1}X && (1.2)\\ &= X[Q], && (1.3)\end{aligned}$$

donde $[Q]$ denota la matriz asociada a la transformación lineal Q.

Luego, por definición:

$$\begin{aligned}[Q] &= \begin{pmatrix} (Qe_1)_1 & (Qe_2)_1 & \cdots & (Qe_p)_1 \\ (Qe_1)_2 & (Qe_2)_2 & \cdots & (Qe_p)_2 \\ \vdots & \vdots & & \vdots \\ (Qe_1)_p & (Qe_2)_p & \cdots & (Qe_p)_p \end{pmatrix} \\ &= \begin{pmatrix} V_{11} & V_{21} & \cdots & V_{p1} \\ V_{12} & V_{22} & \cdots & V_{p2} \\ \vdots & \vdots & & \vdots \\ V_{1p} & V_{2p} & \cdots & V_{pp} \end{pmatrix}.\end{aligned}$$

Con lo cual hemos mostrado que dado cualquier $p-$vector aleatorio X, es posible construir un $p-$vector aleatorio Y con componentes incorrelacionadas tal que:

$$Y = X[Q], \tag{1.4}$$

donde las columnas de la matriz $[Q]$ están formadas por los vectores propios de la matriz de covarianza de X.

Las componentes de Y son llamadas las componentes principales de X; Y_1 la primera componente principal, Y_2 la segunda componentes principal, etc.

Ejemplo 1 *Denotemos por X_1 y X_2 la estatura y peso de un hombre escogido de forma arbitraria. Sea el $2-$vector aleatorio X, dado por: $X = (X_1, X_2)$. Supongamos que conocemos la matriz de covarianza de X:*

$$cov(X) = \begin{pmatrix} 3.572 & 4.365 \\ 4.365 & 8.004 \end{pmatrix}.$$

Calculamos los auto-valores de cov(X), los cuales son:

$$\lambda_1 = 10.683 \; y \; \lambda_2 = 0.893.$$

correspondientes a los vectores propios:

$$V_1 = (0.523, 0.852) \; y \; V_2 = (0.852, -0.523).$$

Entonces, la primera componente principal de X es:

$$Y_1 = \langle X, V_1 \rangle = 0.523X_1 + 0.852X_2,$$

y su segunda componente principal es:

$$Y_2 = \langle X, V_2 \rangle = 0.852X_1 - 0.523X_2.$$

Las variables Y_1 y Y_2 son incorrelacionadas y además:

$$X = Y_1V_1 + Y_2V_2.$$

Observemos que la varianza de Y_1 es mucho mas grande que la varianza de Y_2.

Dado que el problema que se desea resolver es cómo encontrar un espacio de dimensión más reducida que represente adecuadamente los datos, escogeremos la primera componente principal de X, como la componente que posea varianza máxima, así el vector propio asociado a ella es el correspondiente al mayor

auto-valor de $cov(X)$. De igual forma, la segunda componente principal de X, será aquella asociada al vector propio correspondiente al segundo mayor auto-valor; así sucesivamente.

De esta forma, puede demostrarse que el espacio de dimensión r que mejor representa a las p variables viene definido por los vectores propios asociados a los r mayores auto-valores de $cov(X)$. Además, estos r auto-valores miden la varianza capturada.

1.1.4. Distribución normal multivariada

La distribución normal posee una singular importancia dentro de la estadística, una de las principales razones esta relacionada con sus propiedades matemáticas. Su definición puede ser extendida al caso multivariado o vectorial, como veremos a continuación.

Definición 1.1.11 *Se dice que un p-vector aleatorio* $X = \left(X_1, X_2, ..., X_p\right)$ *posee distribución normal (multivariada), si las variables escalares* $X_1, X_2, ..., X_p$ *constituyen un sistema estadísticamente independiente y cada una de ellas posee distribución normal (escalar o univariada).*

Definición 1.1.12 *La* ***función de densidad*** f *de un p-vector aleatorio X, el cual posee distribución normal (multivariada) con media* $\mu = \left(\mu_1, \mu_2, \ldots, \mu_p\right)$ *y matriz de covarianza C, donde* $C \in M_{p\times p}$ *es una matriz definida positiva, viene dada por:*

$$f(X) = \frac{1}{(2\pi)^{\frac{p}{2}} |C|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(X-\mu)' C^{-1}(X-\mu)\right).$$

Observación 1.1.4 *Denotaremos por* $X \sim N(\mu, C)$*, cuando el p-vector aleatorio X posee distribución normal con media* μ *y matriz de covarianza C.*

Propiedad 1.1.3 *Si* $X \sim N(\mu, C)$ *y* $\boldsymbol{a}$ *es un p-vector escalar, entonces* $X + \boldsymbol{a}$ *posee distribución normal, con media* $\mu + \boldsymbol{a}$ *y matriz de covarianza C, esto es,* $X + \boldsymbol{a} \sim N(\mu + \boldsymbol{a}, C)$*.*

Propiedad 1.1.4 *Si $X_1, \ldots, X_n$ es una muestra $p-$dimensional de una población $N(\mu, C)-$distribuida, entonces $\overline{X} \sim N(\mu, C/n)$.*

1.1.5. Distribución Wishart

La distribución Wishart, tiene un papel importante en el análisis del estimador de la matriz de covarianza de un vector aleatorio con distribución normal, juega en el análisis multivariante un papel semejante al de la distribución χ^2 en el estudio inferencial unidimensional.

Definición 1.1.13 *Sea $X_1, \ldots, X_n$ una muestra $p-$dimensional de una población $N(0, C)-$distribuida, entonces la distribución de la variable:*

$$\sum_{i=1}^{n} X_i \otimes X_i,$$

es llamada distribución Wishart con parámetros n y C, donde n es llamado el número de grados de libertad de la distribución y C la matriz de escala. A esta distribución se le denotará como $W(n, C)-$distribución.

Definición 1.1.14 *La **función de densidad** $\mathcal{W}$ de una matriz $\Lambda \in M_{p\times p}$ con distribución Wishart con parámetros n y $C \in M_{p\times p}$, viene dada por:*

$$\mathcal{W}(\Lambda) = B|\Lambda|^{(n-p-1)/2} \exp\left(-\frac{1}{2} Tr\left(C^{-1}\Lambda\right)\right), \tag{1.5}$$

donde Tr denota la traza de una matriz y B es una constante de normalización.

Teorema 1.1.1 *Si $X_1, \ldots, X_n$ es una muestra $p-$dimensional de una población $N(\mu, C)-$distribuida, entonces la variable S^2 es $W(n-1, C)-$ distribuida.*

1.1.6. Versión vectorial de la ley fuerte de los grandes números

La ley fuerte de los grandes números establece que el promedio de una muestra de gran tamaño, tomada al azar de una población tenderá a estar cerca de la

media de la población completa. Este resultado fue enunciado por primera vez sin prueba por el matemático italiano Gerolamo Cardano (1565). Dada su utilidad, enunciaremos la versión vectorial de este teorema.

Sean X un p-vector aleatorio, $X_1, X_2, \ldots$ una secuencia de p-vectores aleatorios y $(\Omega, \mathfrak{U}, \mathbb{P})$ el espacio de probabilidad asociado a estos vectores.

Definición 1.1.15 *Se dice que la secuencia $X_1, X_2, \ldots$ converge fuertemente (o casi segura) a X, si existe un conjunto $A \in \mathfrak{U}$ tal que $\mathbb{P}(A) = 1$ y se satisfaga que:*

$$\lim_{n\to\infty} X_n(w) = X(w), \quad \forall w \in A.$$

Teorema 1.1.2 *Sea $X_1, X_2, \ldots$ una muestra infinita de una población con vector esperanza μ. Entonces,*

$$\frac{X_1 + X_2 + \cdots + X_n}{n} \to \mu, \quad \textit{fuertemente.}$$

Capítulo 2

Estimación por máxima verosimilitud

En estadística, la estimación por máxima verosimilitud es un método habitual para ajustar un modelo y encontrar sus parámetros. Esta técnica fue introducida por R. A. Fisher, genetista y experto en estadística en la década de 1920.

Definición 2.0.16 *Sea $X_1, \ldots, X_n$ una muestra aleatoria de una población con densidad de probabilidad f. La **función de verosimilitud** asociada a la muestra es la función de densidad del vector $(X_1, ..., X_n)$. Es decir, la función de verosimilitud es la función $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ definida por:*

$$\mathcal{L}(x_1, \ldots, x_n) := \prod_{i=1}^{n} f(x_i),$$

donde $(x_1, \ldots, x_n)$ es un resultado de la muestra.

Si la muestra $X_1, \ldots, X_n$ proviene de una población con densidad de probabilidad f_θ, donde $\theta \in \Theta$, entonces la función de verosimilitud depende de $\theta \in \Theta$ y la denotaremos por $\mathcal{L}_\theta$ en lugar de $\mathcal{L}$.

Definición 2.0.17 *Dado $(x_1, \ldots, x_n) \in \mathbb{R}^n$ un resultado de la muestra $(X_1, ..., X_n)$,*

maximizamos la función $\theta \to \mathcal{L}_\theta(x_1, \ldots, x_n)$. Supongamos que existe exactamente un punto $\widehat{\theta} \in \Theta$, donde este máximo es alcanzado. El punto $\widehat{\theta}$ es llamado el ***estimador de máxima verosimilitud*** *de θ.*

Observación 2.0.5 *Es importante notar que $\widehat{\theta}$ depende generalmente de $(x_1, \ldots, x_n)$, es por esto que muchas veces se escribe $\widehat{\theta}(x_1, \ldots, x_n)$ en lugar de $\widehat{\theta}$.*

Observación 2.0.6 *En la práctica se suele obtener el punto de máximo del logaritmo de la función de verosimilitud (función log-verosimilitud), dado que ambas funciones poseen el mismo punto de máximo, pero trabajar con la función log-verosimilitud es más cómodo.*

Ejemplo 2 ***(Estimación de parámetros de $p-$vectores aleatorios con distribución normal multivariada):*** *Sea $X_1, \ldots, X_n$ una muestra $p-$dimensional de una población $N(\mu, C)-$distribuida, donde los parámetros μ y C son desconocidos. Haciendo $\theta = (\mu, C)$, la función de verosimilitud asociada a la muestra $X_1, \ldots, X_n$, viene dada por:*

$$\mathcal{L}_\theta(x_1, \ldots, x_n) = \prod_{i=1}^{n} \frac{1}{(2\pi)^{\frac{p}{2}} |C|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x_i - \mu)' C^{-1}(x_i - \mu)\right). \tag{2.1}$$

Así, despreciando las constantes, la función log-verosimilitud L, será:

$$L_\theta(x_1, \ldots, x_n) = -\frac{n}{2}\log|C| - \frac{1}{2}\sum_{i=1}^{n}(x_i - \mu)' C^{-1}(x_i - \mu). \tag{2.2}$$

Además, la función (2.2) puede ser escrita como:

$$L_\theta(x_1, \ldots, x_n) = -\frac{n}{2}\log|C| - \frac{n}{2}Tr\left(C^{-1}S\right) - \frac{n}{2}\left(\overline{X} - \mu\right)' C^{-1}\left(\overline{X} - \mu\right), \tag{2.3}$$

donde $\overline{X}$ es la media muestral y $S = \frac{1}{n}\sum_{i=1}^{n}\left(x_i - \overline{X}\right)'\left(x_i - \overline{X}\right)$.

Observación 2.0.7 *La función (2.3) solo depende de la muestra a través de $\overline{X}$ y S, por tanto estos son estimadores suficientes de μ y C.*

Luego, maximizando L con respecto a μ y C, obtenemos que los estimadores de máxima verosimilitud de μ y C son $\overline{X}$ y S, respectivamente.

2.1. Algoritmo EM

El algoritmo esperanza-maximización o algoritmo EM es una herramienta muy utilizada en estadística para encontrar el estimador de máxima verosimilitud (o máximo a posteriori) de parámetros en modelos probabilísticos, es de especial utilidad cuando se tienen observaciones incompletas. Es propuesto por Arthur Dempster, Nan Laird y Donald Rubin (1977) en [4].

Supongamos que tenemos dos muestras aleatorias X e Y, donde los datos observados son una realización y de Y y el correspondiente x de X no es observado directamente. Supongamos que existe un mapeo de X a Y, $x \to \widehat{y}(x)$ y la realización x es conocida solo en $X(y)$, un subconjunto de X determinado por la ecuación $y = \widehat{y}(x)$, donde y son los datos observados, esto es, x solo es observable indirectamente a través de y. Llamaremos a x los datos completos y a y los datos incompletos.

Supongamos que las muestras X e Y provienen de poblaciones con densidades de probabilidad f_θ, g_θ donde $\theta \in \Theta$, respectivamente.

Sea $\mathcal{L}_\theta$ la función de verosimilitud asociada a la muestra X y $\widehat{\mathcal{L}}_\theta$ la función de verosimilitud asociada a la muestra Y. Estas funciones de verosimilitud están relacionadas mediante la ecuación:

$$\widehat{\mathcal{L}}_\theta(y) = \int_{X(y)} \mathcal{L}_\theta(x)dx$$

El algoritmo EM realiza la búsqueda del θ que maximiza $\widehat{\mathcal{L}}_\theta$, dadas las observaciones y, mediante el uso de $\mathcal{L}_\theta$.

Cada iteración del algoritmo EM consiste en dos pasos, un paso de expectativa (E-paso) y un paso de maximización (M-paso).

Definiendo:

$$Q(\theta'|\theta) = \mathbb{E}\left(\log \mathcal{L}_{\theta'}(x)|y, \theta\right) \tag{2.4}$$

el algoritmo EM está formado por los siguientes pasos:

1. **E-paso:** Evaluar $Q(\theta|\theta^{(p)})$.
2. **M-paso:** Escoger $\theta^{(p+1)}$ como el mejor valor de θ en Θ tal que maximice $Q(\theta|\theta^{(p)})$, es decir:
$$\theta^{(p+1)} = \arg\max_{\theta\in\Theta} Q(\theta|\theta^{(p)})$$

En algunos casos especiales se obtienen expresiones cerradas para el E-paso o M-paso.

2.1.1. Algoritmo EM generalizado

En general, algoritmo iterativo significa tener una regla aplicable para cualquier punto inicial, es decir, un mapeo $\theta \to M(\theta)$ de Θ en Θ tal que cada paso $\theta^{(p)} \to \theta^{(p+1)}$, esta definido por:

$$\theta^{(p+1)} = M(\theta^{(p)}).$$

Definición 2.1.1 *Un algoritmo iterativo con mapeo $M(\theta)$ es un algoritmo EM generalizado (GEM), si:*

$$Q\left(M(\theta)|\theta\right) \geq Q\left(\theta|\theta\right), \quad \forall\theta \in \Theta.$$

Notemos que la definición anterior del algoritmo EM requiere que:

$$Q\left(M(\theta)|\theta\right) \geq Q\left(\theta'|\theta\right),$$

para todo par (θ', θ) en $\Omega \times \Omega$, es decir, $\theta' = M(\theta)$ maximiza $Q(\theta'|\theta)$.

En [4] se demuestra que bajo ciertas condiciones el algoritmo GEM converge al estimador de máxima verosimilitud. Además, este algoritmo puede ser modificado fácilmente para producir la moda a posteriori de θ en lugar del estimador de máxima verosimilitud de θ.

2.1.2. Algoritmo EM para familias exponenciales

Supongamos que $\mathcal{L}_\theta(x)$ tiene la forma regular de una familia exponencial:

$$\mathcal{L}_\theta(x) = b(x) \exp\left(\theta t(x)'\right) / a(\theta), \tag{2.5}$$

donde θ denota el vector parámetro de dimensión $1 \times r$ y $t(x)$ denota un vector $1 \times r$ de estadísticos suficientes de los datos completos.

El parámetro θ es restringido a un conjunto convexo Θ de dimensión r, tal que (2.5) defina una densidad para todo θ en Θ.

Supongamos que θ^p denota el valor actual de θ luego de p iteraciones del algoritmo, cuando (2.5) se cumple la iteración $p + 1$ del algoritmo EM es descrita por los siguientes dos pasos:

1. **E-paso:** Estimar el estadístico suficiente $t(x)$:

$$t^{(p)} = \mathbb{E}\left(t(x)|y, \theta^{(p)}\right). \tag{2.6}$$

2. **M-paso:** Determinar $\theta^{(p+1)}$ como solución de la ecuación:

$$\mathbb{E}\left(t(x)|\theta\right) = t^{(p)}. \tag{2.7}$$

Cuando las ecuaciones (2.7) poseen solución para $\theta \in \Theta$, entonces esta solu-

ción es única, gracias a la propiedad de convexidad de la función log-verosimilitud para familias exponenciales.

Algoritmo EM para poblaciones normales (ver [2])

Debido a la importancia de la distribución normal multivariada, mostraremos como estimar los parámetros de una población con esta distribución, cuando disponemos de observaciones incompletas.

Sea $x_1, \ldots, x_n$ una realización de $X_1, \ldots, X_n$ una muestra $p-$dimensional de una población $N(\mu, C)-$distribuida, donde los parámetros μ y C son desconocidos. Haciendo $\theta = (\mu, C)$, la función de verosimilitud para una muestra sin valores ausentes, viene dada por (2.1) y la función log-verosimilitud puede escribirse como (2.3). Como vimos en el ejemplo 2, los estimadores de máxima verosimilitud de μ y C son:

$$\widehat{\mu} = \overline{X} = \sum_{i=1}^{n} \frac{x_i}{n},$$

$$\widehat{C} = S = \sum_{i=1}^{n} \frac{x_i' x_i}{n} - \widehat{\mu}\,' \widehat{\mu},$$

respectivamente.

Supongamos que los vectores $x_1, \ldots, x_m$ con $m < n$ son observados y que los vectores $x_{m+1} = z_1, \ldots, x_n = z_{n-m}$ carecen de los valores de algunas variables (o de todas ellas).

Denotemos por y el conjunto de datos disponibles y por z las variables ausentes. Dado $\widehat{\theta}^{(0)} = \left(\widehat{\mu}^{(0)}, \widehat{C}^{(0)}\right)$. La $k-$ésima iteración del algoritmo EM para estimar $\widehat{\mu}$ y $\widehat{C}$, consiste en los siguientes dos pasos:

1. **E-paso:** En este paso hay que calcular las estimaciones $\widehat{x_i}^{(k)}$ para x_i y $\widehat{x_i' x_i}^{(k)}$ para $x_i' x_i$, $\forall\ i \in \{1, 2, \ldots, n\}$. Veamos:

a) Cálculo de $\widehat{x}_i^{(k)}$:

1) Para $i \leq m$: $\widehat{x}_i^{(k)} = x_i$.

2) Para $i > m$: $\widehat{x}_i^{(k)} = \mathbb{E}\left(x_i|y, \widehat{\theta}^{(k-1)}\right)$. Además:

2.1) Si el vector x_i es completamente inobservado, entonces:

$$\mathbb{E}\left(x_i|y, \widehat{\theta}^{(k-1)}\right) = \widehat{\mu}^{(k-1)}.$$

2.2) Si el vector $x_i = [x_{1i}\ x_{2i}]$ se observa parcialmente, de modo que x_{1i} es desconocido y x_{2i} es observado, tenemos que:

$$\mathbb{E}\left(x_i|y, \widehat{\theta}^{(k-1)}\right) = \left[\widehat{x}_{i1}\ x_{2i}\right].$$

Donde:

$$\widehat{x}_{1i} = \mathbb{E}\left(x_{1i}|x_{2i}, \widehat{\theta}^{(k-1)}\right) = \widehat{\mu}_1^{(k-1)} + \widehat{C}_{12}^{(k-1)}\widehat{C}_{22}^{(k-1)^{-1}}\left(x_{2i} - \widehat{\mu}_2^{(k-1)}\right),$$

donde hemos particionado el vector $\widehat{\mu}^{(k-1)}$ y la matriz $\widehat{C}^{(k-1)}$ con relación a los dos bloques de variables (x_{i1} y x_{2i}).

b) Cálculo de $\widehat{x_i x_i}^{(k)}$:

1) Para $i \leq m$: $\widehat{x_i x_i}^{(k)} = x_i' x_i$.

2) Para $i > m$:

2.1) Si el vector x_i es completamente inobservado:

$$\widehat{x_i x_i}^{(k)} = \widehat{C}^{(k-1)} - \widehat{\mu}^{(k-1)'}\widehat{\mu}^{(k-1)}.$$

2.2) Si el vector $x_i = [x_{1i}\ x_{2i}]$ se observa parcialmente, de modo que x_{i1} es desconocido y x_{2i} es observado:

$$\widehat{x_i x_i}^{(k)} = \begin{bmatrix} \mathbb{E}\left(x_{1i}' x_{1i}|y, \widehat{\theta}^{(k-1)}\right) & \mathbb{E}\left(x_{1i}' x_{2i}|y, \widehat{\theta}^{(k-1)}\right) \\ \mathbb{E}\left(x_{2i}' x_{1i}|y, \widehat{\theta}^{(k-1)}\right) & x_{2i}' x_{2i} \end{bmatrix},$$

donde:

$$\mathbb{E}\left(x'_{1i}x_{1i}|y,\widehat{\theta}^{(k-1)}\right) = \mathbb{E}\left(x'_{1i}x_{1i}|x_{2i},\widehat{\theta}^{(k-1)}\right) = \widehat{C}_{1.2} + \widehat{x}_{1i}\,'\,\widehat{x}_{1i},$$

con $\widehat{C}_{1.2} = \widehat{C}_{11}^{(k-1)} - \widehat{C}_{12}^{(k-1)}\widehat{C}_{22}^{(k-1)^{-1}}\widehat{C}_{21}^{(k-1)}$.

$$\mathbb{E}\left(x'_{1i}x_{2i}|y,\widehat{\theta}^{(k-1)}\right) = \mathbb{E}\left(x'_{1i}x_{2i}|x_{2i},\widehat{\theta}^{(k-1)}\right) = \widehat{x}_{1i}\,'x_{2i},$$

$$\mathbb{E}\left(x'_{2i}x_{1i}|y,\widehat{\theta}^{(k-1)}\right) = \mathbb{E}\left(x'_{2i}x_{1i}|x_{2i},\widehat{\theta}^{(k-1)}\right) = x'_{2i}\widehat{x}_{1i}.$$

2. **M-paso:** Calculamos:

$$\widehat{\mu}^{(k)} = \sum_{i=1}^{n} \frac{\widehat{x}_i^{(k)}}{n},$$

$$\widehat{C}^{(k)} = \sum_{i=1}^{n} \frac{\widehat{x_i x_i}^{(k)}}{n} - \widehat{\mu}^{(k)\,\prime}\widehat{\mu}^{(k)}.$$

Observación 2.1.1 *El algoritmo finaliza cuando el cambio en los parámetros de una iteración a la siguiente sea menor a una tolerancia dada.*

2.2. Algoritmo SAEM

2.2.1. Método de aproximaciones estocásticas

La versión de aproximación estocástica del algoritmo EM (SAEM, por sus siglas en inglés) reemplaza el usual E-paso del algoritmo EM, por un procedimiento de aproximaciones estocásticas. Es por esto que antes de presentar el algoritmo SAEM, haremos una breve exposición del método de aproximaciones estocásticas.

El método de aproximaciones estocásticas originalmente es presentado por

Robbins y Monro (1951) en [9], como un procedimiento recursivo para encontrar la raíz de una función real $\overline{g}(\cdot)$ de una variable real θ. La función es desconocida, pero pueden tomarse observaciones en valores de θ seleccionados por el experimentador.

Si consideramos que $\overline{g}(\cdot)$ es conocida y continuamente diferenciable, entonces este sería un problema clásico de análisis numérico y podríamos usar un procedimiento de Newton para resolverlo. El procedimiento de Newton genera una sucesión $\{\theta_n\}_{n\geq 1}$ de estimadores de la raíz $\overline{\theta}$, definida recursivamente por:

$$\theta_{n+1} = \theta_n - \left[\overline{g}_\theta(\theta_n)\right]^{-1} \overline{g}(\theta_n),$$

donde $\overline{g}_\theta(\cdot)$ denota la derivada de $\overline{g}(\cdot)$ con respecto de θ.

Supongamos que $\overline{g}(\theta) < 0$ para $\theta > \overline{\theta}$, $\overline{g}(\theta) > 0$ para $\theta < \overline{\theta}$ y que $\overline{g}_\theta(\theta)$ es estrictamente negativa y está acotada en una vecindad de $\overline{\theta}$. Entonces, $\{\theta_n\}_{n\geq 1}$ converge a $\overline{\theta}$, si θ_0 está en una vecindad suficientemente pequeña de $\overline{\theta}$. Una alternativa más simple pero menos eficiente, consiste en fijar $\epsilon > 0$ suficientemente pequeño y usar la siguiente ecuación recursiva:

$$\theta_{n+1} = \theta_n + \epsilon\overline{g}(\theta_n). \tag{2.8}$$

El algoritmo definido por la ecuación (2.8) no requiere la diferenciabilidad de $\overline{g}(\cdot)$, además esta garantizada su convergencia si $\theta_0 - \overline{\theta}$ es suficientemente pequeño. Existen procedimientos más rápidos y simples en este caso, cuando los valores de $\overline{g}(\theta)$ y sus derivadas pueden ser calculadas.

Supongamos ahora que los valores de $\overline{g}(\theta)$ no son conocidos, pero pueden tomarse observaciones en valores seleccionados de θ. En este caso no puede ser usado el procedimiento de Newton, pero se podría considerar un procedimiento como el definido por (2.8), reemplazando $\overline{g}(\theta_n)$ por una buena estimación de su

valor, promediando muchas observaciones.

Se demostró en [9] que tomando un número muy grande de observaciones de cada uno de los θ_n y considerando el promedio en lugar de $\overline{g}(\theta_n)$ en (2.8) el algoritmo es ineficiente, ya que θ_n es solo un intermediario en el cálculo y el valor de $\overline{g}(\theta_n)$ es sólo de interés en la medida que nos lleva en la dirección correcta. En este artículo proponen la siguiente ecuación recursiva:

$$\theta_{n+1} = \theta_n + \epsilon_n Y_n, \tag{2.9}$$

donde $\{\epsilon_n\}_{n\geq 1}$ es una sucesión decreciente de tamaños de pasos positivos que converge a 0, satisfaciendo que $\sum_n \epsilon_n \to \infty$, y Y_n es una estimación del valor de $\overline{g}(\theta_n)$. En [9], también se usó la condición $\sum_n \epsilon_n^2 < \infty$, pero posteriormente se demostró que puede ser debilitada.

Los tamaños de paso decrecientes implican que la velocidad de cambio de θ_n se frena cuando n tiende a infinito. La buena escogencia de la secuencia $\{\epsilon_n\}_{n\geq 1}$ es importante, dado que es fundamental para la eficiencia del algoritmo definido por (2.9). La idea detrás de este método es que los tamaños de pasos decrecientes proporcionan un promedio implícito de las observaciones. Esto, además de las pruebas de convergencia asociadas para un gran número de aplicaciones, pueden ser encontradas en una gran cantidad de literatura sobre algoritmos estocásticos, por ejemplo en [8].

La forma recursiva del estimador de mínimos cuadrados lineal del valor esperado de una variable aleatoria, proporciona otra motivación para la forma de (2.9) y ayuda a explicar como tamaños de pasos decrecientes producen en realidad un promedio de las observaciones. Sea $\{\xi_n\}_{n\geq 1}$ una sucesión de variables aleatorias independientes e idénticamente distribuidas con varianza finita y valor esperado $\overline{\theta}$, desconocido. Dadas las observaciones ξ_i con $i \in \{1, \ldots, n\}$, el estimador de

mínimos cuadrados lineal de $\overline{\theta}$ es $\theta_n = \sum_{i=1}^{n} \frac{\xi_n}{n}$. El cual se puede escribir de forma recursiva como:

$$\theta_{n+1} = \theta_n + \epsilon_n \left[\xi_{n+1} - \theta_n\right], \tag{2.10}$$

donde $\theta_0 = 0$ y $\epsilon_n = \dfrac{1}{n+1}$.

Así, con el uso de la sucesión de tamaños de pasos decreciente, definida por $\epsilon_n = \dfrac{1}{n+1}$, se obtiene un estimador que es equivalente al obtenido por un promedio directo de las observaciones, y (2.10) es un caso especial de (2.9). Además, el "filtro recursivo" de (2.10) proporciona una visión más clara del proceso de estimación, ya que muestra que el estimador cambia solo por $\epsilon_n \left[\xi_{n+1} - \theta_n\right]$, que se puede describir como el producto del coeficiente de "fiabilidad" ϵ_n y la "estimación del error" $\xi_{n+1} - \theta_n$. La intuición detrás del cálculo del promedio se mantiene incluso en algoritmos mucho más complicados, bajo condiciones generales.

En el procedimiento general de Robbins-Monro, Y_n y θ_n toman valores en $\mathbb{R}^n$, donde Y_n es una observación de una función vectorial $\overline{g}(\cdot)$ evaluada en θ_n, y se desea encontrar la raíz de está función. Desde la obra inicial de Robbins y Monro, ha habido un aumento constante de investigaciones con aplicaciones en muchas áreas diferentes, por ejemplo en redes de colas, comunicaciones inalámbricas, sistemas de fabricación, problemas de aprendizaje y redes neuronales, entre otros.

2.2.2. Algoritmo SAEM

Como hemos comentado el algoritmo SAEM reemplaza el usual E-paso del algoritmo EM, por un procedimiento de aproximaciones estocásticas, esto con el objeto de estimar la ecuación (2.4). Por tanto, es de utilidad cuando el E-paso del algoritmo EM no se puede o es difícil de calcular. Este algoritmo es propuesto por Delyon, Lavielle y Moulines (1999) en [3].

Sea x los datos completos, hagamos $x = (y, z)$, donde y son los datos observados y z los datos no observados. Definamos:

$$p(z;\theta) = \begin{cases} \mathcal{L}_\theta(z)/\widehat{\mathcal{L}}_\theta(y), & si \quad \widehat{\mathcal{L}}_\theta(y) \neq 0 \\ \\ 0, & si \quad \widehat{\mathcal{L}}_\theta(y) = 0 \end{cases}$$

Así, p es la distribución a posteriori de los datos no observados z, dado los datos observados y.

Dado que en muchos casos la evaluación numérica de (2.4) es complicada, Wei y Tanner en [7], proponen utilizar un método Monte Carlo en el E-paso del algoritmo EM, y de esta forma se obtiene el algoritmo MCEM. En particular, la ecuación (2.4) motiva el siguiente esquema: dada la aproximación $\theta^{(p)}$, generar una muestra $z_p^{(j)}$ con $j = 1, \ldots, m(p)$ de $p(z;\theta^{(p)})$ y aproximar $Q(\theta|\theta^{(p)})$ con:

$$\widehat{Q}_p(\theta) = \frac{1}{m(p)} \sum_{j=1}^{m(p)} \log L_\theta(z_p^{(j)}). \tag{2.11}$$

El M-paso consiste en maximizar el lado derecho de (2.11).

Una alternativa para el algoritmo MCEM es el algoritmo SAEM, el cual como hemos mencionado es propuesto en [3] y utiliza procedimientos de aproximaciones estocásticas para estimar la ecuación (2.4). Los siguientes pasos resumen el algoritmo SAEM:

1. **Simulación:** Generar muestras $z_p^{(j)}$ con $j = 1, \ldots, m(p)$ de $p(z;\theta^{(p)})$.

2. **Aproximación estocástica:** Actualizar $\widehat{Q}_p(\theta)$, como:

$$\widehat{Q}_p(\theta) = \widehat{Q}_{p-1}(\theta) + \gamma_p \left(\frac{1}{m(p)} \sum_{j=1}^{m(p)} \log L_\theta(z_p^{(j)}) - \widehat{Q}_{p-1}(\theta) \right),$$

donde $\{\gamma_p\}_{p\geq 1}$ es una sucesión decreciente de tamaños de pasos positivos, la cual converge a 0 y satisface que $\sum_p \gamma_p = \infty$.

3. **Maximización:** Maximizar $\widehat{Q}_p(\theta)$ en Θ. Esto es, hallar $\theta^{(p+1)} \in \Theta$ tal que:

$$\widehat{Q}_p(\theta^{(p+1)}) \geq \widehat{Q}_p(\theta), \quad \forall \theta \in \Theta.$$

El algoritmo SAEM se puede escribir en términos de los estadísticos suficientes, cuando la función de verosimilitud de los datos completos pertenece a la familia exponencial; considerando un procedimiento de aproximaciones estocásticas para estimar el lado derecho de la ecuación (2.6). Además, en [3] se demuestra que bajo supuestos muy generales el algoritmo SAEM converge al máximo (local o global) de la función de verosimilitud de las observaciones.

En muchas situaciones practicas, no es posible generar los datos no observados z_p exactamente de la distribución condicional $p(\cdot|\theta^{(p)})$, por lo que el paso de simulación puede realizarse a través de un algoritmo de método Monte Carlo con cadenas de Markov (MCMC, por sus siglas en inglés). Con esta modificación en el paso de simulación, muchas veces llamado S-paso, los autores de [6], demuestran que bajo ciertas condiciones el algoritmo SAEM acoplado con un algoritmo MCMC converge al estimador de máxima verosimilitud.

Capítulo 3

Algoritmos MCMC

En este capítulo mostraremos dos importantes algoritmos de simulación, el muestreador de Gibbs y el algoritmo Metropolis-Hastings, los cuales pertenecen a la clase de algoritmos MCMC (Monte Carlo Markov Chain). Estos algoritmos nos permiten la obtención de pseudo-muestras provenientes de una distribución de probabilidad, donde estas muestras pueden cumplir con ciertas condiciones que permiten estimar propiedades probabilísticas que no pueden ser obtenidas por métodos analíticos.

Lo expuesto a lo largo de este capítulo, puede ser consultado en [1], [10] y [17].

3.1. Introducción

Los métodos Bayesianos son reconocidos como una manera coherente de hacer inferencia, en contraste con los métodos clásicos, donde los datos obtenidos de estudios observacionales y/o experimentales son analizados con modelos que dependen del tipo de datos con procedimientos de inferencia y decisión particulares para cada caso. El análisis Bayesiano trata de una manera unificada la inferencia y la decisión, tomando en consideración la incertidumbre asociada al modelo y a los

parámetros, proporcionando de una vez las herramientas para cuantificar esta incertidumbre. Por otra parte, el tratamiento de las cantidades no observadas como variables aleatorias y el análisis condicional, permiten naturalmente considerar modelos jerárquicos que son difíciles o imposibles de manejar con la estadística clásica. Su nombre de "Bayesiana" proviene del uso frecuente que hace del teorema de Bayes, el cual establece lo siguiente:

Suponga que se observa una variable aleatoria Z y se desea hacer inferencias acerca de otra variable aleatoria θ, donde θ se extrae de alguna densidad $\pi(\theta)$, a esta distribución se le conoce como distribución previa o a priori de θ. Entonces:

$$\pi(\theta|Z) = \frac{\pi(Z|\theta)\,\pi(\theta)}{\pi(Z)}, \tag{3.1}$$

donde:

- En el caso discreto: $\pi(Z) = \sum_{\theta} \pi(\theta)\,\pi(Z|\theta)$ (suma sobre los posibles valores de θ).
- En el caso continuo: $\pi(Z) = \int \pi(\theta)\,\pi(Z|\theta)\,d\theta$.

Este resultado fue enunciado por Thomas Bayes en 1763, pero no es hasta finales de la década de 1980 y principios de la década de 1990 con la aparición de métodos computacionales basados en simulación, los que permiten implementar el paradigma Bayesiano en la práctica. En la inferencia Bayesiana se requiere encontrar la densidad a posteriori de los parámetros o cantidades desconocidas de los modelos, la cual se define como sigue.

Definición 3.1.1 *Sea $Z = (z_1, z_2, \ldots, z_n)$ una muestra aleatoria de una población con función de densidad f_θ, donde θ es un parámetro desconocido y función de verosimilitud $\mathcal{L}_\theta(Z) = \prod_{i=1}^{n} f_\theta(z_i)$. Se define la densidad a posteriori para el caso*

continuo como:

$$\begin{aligned} \pi(\theta|Z) &= \frac{\mathcal{L}_\theta(Z)\,\pi(\theta)}{\int \mathcal{L}_\theta(Z)\,\pi(\theta)\,d\theta} \\ &\propto \mathcal{L}_\theta(Z)\,\pi(\theta), \end{aligned}$$

donde $\propto$ *significa "proporcional a" y la distribución* $\pi(\theta)$ *es la distribución previa o a priori de* θ*.*

Obtener expresiones analíticas de la densidad a posteriori sólo es posible en casos particulares que usualmente representan modelos muy sencillos; en general es posible obtener la estimación de la densidad a posteriori través de métodos de simulación, como lo son los algoritmos presentados en este capítulo. Como este tipo de métodos introducen un nivel de aleatoriedad en el análisis, se les considera como métodos de Monte Carlo, en honor al famoso casino del principado de Mónaco, además, como ya fue mencionado, ambos algoritmos pertenecen a la clase de algoritmos MCMC.

3.2. Cadenas de Markov

Antes de presentar el muestreador de Gibbs y el algoritmo Metropolis-Hastings, exponemos algunos resultados necesarios relativos a las cadenas de Markov.

Definición 3.2.1 *Un proceso estocástico es una familia de variables aleatorias,* X_t*, donde* $t \in T$*, y* T *es un conjunto adecuado de índices.*

Definición 3.2.2 *Un proceso de Markov* $\{X_t\}$ *es un proceso estocástico con la siguiente propiedad: dado el valor de* X_t*, los valores de* X_s *con* $s > t$ *no están influenciados por los valores de* X_u *con* $u < t$*. Es decir, la probabilidad de cualquier comportamiento futuro del proceso, cuando el estado actual se conoce con exactitud, no se ve alterada por el conocimiento de su comportamiento en el pasado.* ***Una cadena de Markov de tiempo discreto****, es un proceso de Markov cuyo*

espacio-estado (rango de posibles valores de las variables aleatorias) es finito o un conjunto numerable, y donde el conjunto de índices es $T = (0, 1, 2, 3, \ldots)$.

En términos formales, la propiedad de Markov es la siguiente:

$$Pr\{X_{n+1} = j | X_0 = i_0, \ldots, X_{n-1} = i_{n-1}, X_n = i\} = Pr\{X_{n+1} = j | X_n = i\}, \qquad (3.2)$$

para todo tiempo n y todos los estados $i_0, i_1, \ldots, i_{n-1}, i, j$.

Observación 3.2.1 *En una cadena de Markov, interpretamos a* X_n *como el "estado del sistema en el instante n".*

La probabilidad de que X_{n+1} este en el estado j dado que X_n esta en el estado i es llamada **la probabilidad de transición de un paso** y es denotado por $P_{ij}^{n,n+1}$, esto es:

$$P_{ij}^{n,n+1} = Pr\{X_{n+1} = j | X_n = i\}.$$

Esta notación hace hincapié en que en general la probabilidad de transición no solo es función del estado inicial y final, si no también del tiempo de transición. Cuando la probabilidad de transición de los pasos son independientes de la variable de tiempo n, se dice que la cadena de Markov es **homogénea**. Como la mayoría de las cadenas de Markov que se encuentran poseen esta propiedad, pondremos nuestro interés en este caso. Así, tenemos que $P_{ij}^{n,n+1} = P_{ij}$, donde P_{ij} es la probabilidad condicional de que el valor del estado i se mueva al estado j en un paso.

Observación 3.2.2 *Muchas veces es conveniente etiquetar el espacio-estado de una cadena de Markov con los enteros no-negativos* $\{0, 1, 2, \ldots\}$.

Definición 3.2.3 *Se define la matriz de probabilidades transición o matriz de Markov, como la matriz* $\boldsymbol{P}$ *cuyo elemento* (i, j) *es* P_{ij}*, es decir, la matriz en cuyo elemento* (i, j) *se encuentra la probabilidad de pasar del estado i al estado j.*

Así, la matriz **P** es como sigue:

$$\mathbf{P} = \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} & \cdots \\ P_{10} & P_{11} & P_{12} & P_{13} & \cdots \\ P_{20} & P_{21} & P_{22} & P_{23} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \\ P_{i0} & P_{i1} & P_{i2} & P_{i3} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \end{bmatrix}$$

Si el número de estados es finito, entonces **P** es una matriz cuadrada finita, cuyo orden (el número de filas) es igual al número de estados. Claramente, las cantidades P_{ij} satisfacen las condiciones:

$$P_{ij} \geq 0, \qquad \forall i, j = 0, 1, 2, \ldots \tag{3.3}$$

$$\sum_{j=0}^{\infty} P_{ij} = 1, \qquad \forall i, j = 0, 1, 2, \ldots \tag{3.4}$$

La condición (3.4) simplemente indica que alguna transición se produce en cada ensayo (por conveniencia se dice que la transición se ha producido incluso si el estado se mantiene sin cambios).

Ejemplo 3 *Después de algunos estudios sobre el clima en determinada ciudad, se ha observado que si un día está soleado en el 70 % de los casos el día siguiente continúa soleado y en el 30 % nublado. Por el contrario, si un día está nublado, la probabilidad de que esté soleado el día siguiente es de 60 % y la probabilidad de que se siga nublado es de 40 %. Además, según estos estudios el clima de un determinado día solo depende del clima del día anterior.*

Para estudiar este problema primeramente encontramos las probabilidades de transición, es decir las probabilidades de que teniendo cierto clima un día, al día siguiente se tenga otro o el mismo clima. Así, si indicamos con s a un día soleado y con n a un día nublado, tendremos:

$$P_{ss} = 0.7,$$
$$P_{sn} = 0.3,$$
$$P_{ns} = 0.6,$$
$$P_{nn} = 0.4.$$

Si acomodamos estos datos en una matriz, obtenemos la matriz de probabilidades de transición:

$$\boldsymbol{P} = \begin{bmatrix} 0.7 & 0.3 \\ 0.6 & 0.4 \end{bmatrix},$$

donde las filas indican el clima en el día, fila 1 soleado y fila 2 nublado, y las columnas el clima en el día siguiente, columna 1 soleado y columna 2 nublado.

Observamos que en esta matriz no sólo los coeficientes son no-negativos, sino que al sumarlos por filas obtenemos 1. Este es un ejemplo de cadena de Markov ya que:

1. *Tenemos ciertos estados, en este caso s y n.*
2. *En cada momento estamos en uno de estos estados.*
3. *En el próximo momento volveremos a estar en ese u otro estado.*
4. *Pasamos de un estado a otro con cierta probabilidad, que sólo puede depender del estado inmediatamente anterior y esa probabilidad no cambia con el transcurso del tiempo.*

Un proceso de Markov es completamente definido una vez que la matriz de probabilidades de transición y el estado inicial X_0 (de forma más general, la dis-

tribución de probabilidad de X_0) están especificados. Probemos este hecho:

Sea $Pr\{X_0 = i_0\} = p_{i_0}$. Es suficiente mostrar como calcular las cantidades:

$$Pr\{X_0 = i_0, X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n\}, \tag{3.5}$$

ya que cualquier probabilidad que involucra $X_{j_1}, \ldots, X_{j_k}$, para $j_1 < \ldots < j_k$, puede ser obtenida, acorde al axioma de probabilidad total, por la suma de términos de la forma (3.5).

Por definición de probabilidad condicional:

$$\begin{aligned} Pr \quad & \{X_0 = i_0, X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n\} \\ = \quad & Pr\{X_0 = i_0, X_1 = i_1, X_2 = i_2, \ldots, X_{n-1} = i_{n-1}\} \\ \times \quad & Pr\{X_n = i_n | X_0 = i_0, X_1 = i_1, \ldots, X_{n-1} = i_{n-1}\}. \end{aligned} \tag{3.6}$$

Ahora, por definición de cadena de Markov,

$$\begin{aligned} Pr\{X_n = i_n | X_0 = i_0, X_1 = i_1, X_2 = i_2, \ldots, X_{n-1} = i_{n-1}\} &= Pr\{X_n = i_n | X_{n-1} = i_{n-1}\} \\ &= P_{i_{n-1} i_n}. \end{aligned} \tag{3.7}$$

Sustituyendo (3.7) en (3.6):

$$\begin{aligned} Pr \quad & \{X_0 = i_0, X_1 = i_1, X_2 = i_2, \ldots, X_n = i_n\} \\ = \quad & Pr\{X_0 = i_0, X_1 = i_1, X_2 = i_2, \ldots, X_{n-1} = i_{n-1}\} P_{i_{n-1} i_n}. \end{aligned} \tag{3.8}$$

Al repetir el proceso $n - 1$ veces, (3.5) se convierte en:

$$Pr\{X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n\} = p_{i0} P_{i_0 i_1} \cdots P_{i_{n-2} i_{n-1}} P_{i_{n-1} i_n}.$$

Esto demuestra que toda probabilidad finita dimensional es especificada una vez que la matriz de probabilidades de transición y la distribución del estado inicial son dados, y el proceso es definido por estas cantidades.

Por otro lado, sea F la distribución condicional de X_k dado X_0, para algún k; se puede demostrar que bajo ciertas condiciones esta distribución converge a una única distribución Π, la cual llamamos **distribución estacionaria o invariante**.

Además, bajo algunas condiciones sobre la distribución condicional F, se puede mostrar que una cadena de Markov con distribución estacionaria π es **ergódica**, esto es:

$$\lim_{N\to\infty} \frac{1}{N} \sum_{k=1}^{N} h(X_k) = \mathbb{E}_{\Pi}\left[h(x)\right],$$

donde h es alguna función que se desea estimar.

Esto nos permite estimar valores esperados a partir de los resultados de una realización de la cadena. De esta manera, construyendo una cadena de Markov cuya distribución estacionaria corresponda a la distribución posterior, se puede encontrar la esperanza posterior de cualquier función, simulando la cadena y tomando el promedio de la función sobre los valores simulados. Por supuesto, este procedimiento descansa en los resultados asintóticos (ver sección 1.1.6, versión vectorial de la ley fuerte de los grandes números), por lo que se hace necesario realizar un número grande de simulaciones y estudiar la convergencia de los valores obtenidos.

Así, el objetivo de los métodos Monte Carlo con cadenas de Markov es encontrar una cadena de Markov en el espacio de parámetros, de manera tal que la distribución estacionaria de la cadena coincida con la distribución posterior.

3.3. Algoritmo Metropolis-Hastings

La referencia al algoritmo de Metropolis-Hastings corresponde a un término general que se utiliza para una familia de métodos de simulación, compuesta de algoritmos universales que generan cadenas de Markov con distribución estacionaria que corresponde al posterior de interés, estos se derivan del algoritmo de Metrópolis.

El algoritmo de Metropolis, es una modificación de un paseo al azar que utiliza una regla de aceptación-rechazo para obtener convergencia de la cadena a una distribución específica. Este algoritmo consiste de los siguientes pasos:

1. Simular un punto inicial para el cual $\pi\left(\theta^0|Z\right) > 0$, a partir de de una distribución inicial $p_0\left(\theta\right)$.

Para $s = 1, \ldots, n_{sim}$

2. Obtener una realización candidata ϑ a partir de una distribución de salto en el tiempo s, $J_s\left(\vartheta|\theta^{(s-1)}\right)$. Esta distribución debe ser simétrica en el sentido de que $J_s(\theta^{(a)}|\theta^{(b)}) = J_s(\theta^{(b)}|\theta^{(a)})$, para todo $\theta^{(a)}$, $\theta^{(b)}$, s.

3. Calcular:
$$r = \frac{\pi\left(\vartheta|Z\right)}{\pi\left(\theta^{(s-1)}|Z\right)}.$$

4. Hacer:
$$\theta^{(s)} = \begin{cases} \vartheta, & \text{con probabilidad mín}\left\{1, r\right\} \\ \theta^{(s-1)}, & \text{en otro caso} \end{cases} \tag{3.9}$$

La regla de aceptación y rechazo del algoritmo anterior se puede interpretar como sigue: si el "salto" produce un valor para el que aumenta la densidad a posterior, hacer $\theta^{(s)} = \vartheta$; si el "salto" no aumenta la densidad a posteriori, hacer $\theta^{(s)} = \vartheta$ con probabilidad r y $\theta^{(s)} = \theta^{(s-1)}$, si no. Esto puede ser visto como una versión

estocástica de un algoritmo de búsqueda de moda por pasos.

Esta primera versión del algoritmo fue construida por Metropolis (1949) y más tarde, es generalizada por Hastings (1970), cuya generalización consiste esencialmente en que las reglas de salto, dadas por J_s no necesitan ser simétricas y el radio r es reemplazado por,

$$r = \frac{\pi\left(\vartheta|Z\right)/J_s\left(\vartheta|\theta^{(s-1)}\right)}{\pi\left(\theta^{(s-1)}|Z\right)/J_s\left(\theta^{(s-1)}|\vartheta\right)}.$$

Al ser este algoritmo de aceptación-rechazo, la eficiencia en la generación de la cadena dependerá de las propiedades de la distribución de salto; una buena distribución de salto debería cumplir con las siguientes propiedades:

- Para cualquier θ, es fácil muestrear de $J(\vartheta|\theta)$.
- Es fácil calcular los cocientes de importancia r.
- Cada salto produce resultados a una distancia razonable en el espacio de parámetros.
- Los saltos no son rechazados muy frecuentemente.

En esta sección presentamos el algoritmo RWM (Random Walk Metrópolis) el cual pertenece a la clase más general de los algoritmos Metropolis-Hastings, en la cual es utilizada como distribución de salto, la distribución normal y donde h es la función que se desea estimar.

Algoritmo Random Walk Metropolis

Inicialización:

- Denote la moda del posterior por $\widetilde{\theta}$.
- Sea $\widetilde{\Sigma}$ la estimación de la matriz de covarianza de $\widetilde{\theta}$.

1. Generar $\theta^{(0)}$ desde $N\left(\widetilde{\theta}, c^2\widetilde{\Sigma}\right)$ o dar un valor inicial específico.

Para $s = 1, ..., n_{sim}$.

2. Generar ϑ desde $N\left(\theta^{(s-1)}, c^2\widetilde{\Sigma}\right)$.
3. Calcular:
$$r\left(\theta^{(s-1)}, \vartheta|Z\right) = \frac{\mathcal{L}_{\vartheta}(Z)\,\pi(\vartheta)}{\mathcal{L}_{\theta^{(s-1)}}(Z)\,\pi\left(\theta^{(s-1)}\right)}.$$
4. Con probabilidad mín $\left\{1, r\left(\theta^{(s-1)}, \vartheta|Z\right)\right\}$ aceptar ϑ y hacer $\theta^{(s)} = \vartheta$, en otro caso, rechazar ϑ y hacer $\theta^{(s)} = \theta^{(s-1)}$.
5. Aproximar el valor esperado del posterior de la función $h(\theta)$ por:
$$\frac{1}{n_{sim}} \sum_{s=1}^{n_{sim}} h\left(\theta^{(s)}\right).$$

Observación 3.3.1 *El escalar c utilizado en el algoritmo, permite re-escalar la matriz $\widetilde{\Sigma}$ de tal modo que los vectores generados ϑ, sean aceptados en la mayoría de los casos, lo cual es esperado en este algoritmo.*

3.4. Muestreador de Gibbs

Este algoritmo, también llamado de muestreo condicional alternante, es sumamente útil cuando el espacio de parámetros es altamente multidimensional; fue introducido por los hermanos Geman en 1984, con el fin de resolver problemas de reconstrucción en el tratamiento de imágenes.

Como sabemos, deseamos estimar la distribución posterior $\pi(\theta|Z)$. Supongamos que θ tiene d componentes, es decir: $\theta = (\theta_1, \theta_2, \ldots, \theta_d)$.

En cada iteración s del algoritmo, se escoge un ordenamiento de los d subvectores, y cada θ_j^s es muestreado de la distribución condicional dados todos los demás componentes:

$$\pi\left(\theta_j|\theta_{-j}^{s-1}, Z\right),$$

donde θ_{-j}^{s-1} representa todos los componentes de θ excepto por θ_j, en sus valores actuales, esto es:

$$\theta_{-j}^{s-1} = \left(\theta_1^s, \ldots, \theta_{j-1}^s, \theta_{j+1}^{s-1}, \ldots, \theta_d^{s-1}\right).$$

Así, dado un valor inicial $\theta_0 = \left(\theta_1^0, \ldots, \theta_d^0\right)$, el muestreador de Gibbs, consiste en:

Para $s = 1, \ldots, n_{sim}$:

Muestrear:

$$\begin{aligned}
\theta_1^s &\sim \pi\left(\theta_1|\theta_2^{s-1}, \ldots, \theta_d^{s-1}\right), \\
\theta_2^s &\sim \pi\left(\theta_2|\theta_1^s, \theta_3^{s-1}, \ldots, \theta_d^{s-1}\right), \\
\theta_3^s &\sim \pi\left(\theta_3|\theta_1^s, \theta_2^s, \theta_4^{s-1}, \ldots, \theta_d^{s-1}\right), \\
&\vdots \\
\theta_d^s &\sim \pi\left(\theta_d|\theta_1^s, \theta_2^s, \ldots, \theta_{d-1}^s\right).
\end{aligned}$$

Es importante notar que el muestreador de Gibbs es un caso especial de un algoritmo Metropolis-Hastings con distribución de salto dada por:

$$J_{j,r}^{Gibbs}\left(\theta^{*}|\theta^{s-1}\right)=\begin{cases}\pi\left(\theta_{j}^{*}|\theta_{-j}^{s-1},Z\right), & \text{si } \theta_{-j}^{*}=\theta_{-j}^{s-1}\\ 0, & \text{si no}\end{cases}$$

En este caso, $r = 1$ de manera que todos los saltos son aceptados.

Cuando no es posible muestrear de alguna o de todas las distribuciones condicionales $\pi\left(\theta_j|\theta_{-j}^{s-1},Z\right)$, pero si de aproximaciones $g\left(\theta_j|\theta_{-j}\right)$, se puede usar la misma estrategia de muestreo condicional alternante, compensado por la aproximación, con la siguiente función de salto para el $j-$ésimo paso de Metropolis en la iteración s:

$$J_{j,s}\left(\theta^{*}|\theta^{s-1}\right)=\begin{cases}g\left(\theta_{s}^{*}|\theta_{-j}^{s-1}\right), & \text{si } \theta_{-j}^{*}=\theta_{-j}^{s-1}\\ 0, & \text{si no}\end{cases}$$

Se calculan los radios de aceptación r y se usa la regla de asignación de la ecuación (3.9), para este caso.

Capítulo 4

Completación de matrices

En este capítulo mostraremos el problema de completación de matrices; el modelo de factorización probabilística de matrices (PMF, por su siglas en inglés), el cual es propuesto en [14] y [15], y ha sido utilizado para resolver algunos de estos problemas. Posteriormente, describimos nuestra propuesta, haciendo uso de las herramientas expuestas en los capítulos anteriores.

4.1. Problema de completación de matrices

En el problema de completación de matrices tenemos una matriz de datos, la cual queremos conocer con la mayor precisión posible pero solo conocemos un subconjunto de sus entradas, como fruto de su propia naturaleza. Se presenta en muchos problemas prácticos de interés en diversos campos como la teoría de sistemas y control; procesamiento de imágenes y filtrado colaborativo, entre otros. Las matrices que tratamos en este problema suelen recibir el nombre de matrices parciales.

Definición 4.1.1 *Dada una matriz $A \in \mathbb{R}^{n\times m}$, se dice que A es una matriz parcial si parte de sus elementos son conocidos y el resto están por especificar.*

Ejemplo 4 *La matriz A dada por:*

$$\begin{pmatrix} 1 & * & 3 & 9 \\ 2 & -1 & 4 & * \\ * & 0 & 3 & 0 \\ * & * & 1 & 1 \end{pmatrix},$$

donde $*$ representa los elementos desconocidos de la matriz, es una matriz parcial.

Cuando damos valores a los elementos no especificados de una matriz parcial A, obtenemos una matriz convencional, que recibe el nombre de completación de A.

Ejemplo 5 *La matriz:*

$$\begin{pmatrix} 1 & 8 & 3 & 9 \\ 2 & -1 & 4 & 1 \\ 0 & 0 & 3 & 0 \\ -\frac{1}{2} & 4 & 1 & 1 \end{pmatrix},$$

es una completación de la matriz parcial del ejemplo anterior.

Así, el problema de completación de matrices consiste en encontrar una completación de una matriz parcial dada; pero de tal forma que ésta verifique ciertas propiedades o condiciones deseadas.

Este problema ha sido estudiado desde diversos puntos de vista, por ejemplo, a través de optimización convexa; demostrándose en [5] y [16] que bajo algunas condiciones puede ser resuelto. Esto, considerando el problema de minimización del rango y encontrando una solución minimizando la norma nuclear; dado que una hipótesis muy común es la de suponer que la matriz posee rango bajo o aproximadamente bajo. Varios tipos de algoritmos han sido propuestos para resolver el problema de la minimización del rango, entre ellos un método de Lagrangeano aumentado, propuesto por Chen y colaboradores en [12].

Desde la perspectiva Bayesiana, han surgido diversas propuestas, una de estas es basada en un modelo Gaussiano y un algoritmo MAP-EM, mostrado en [11]. Por otro lado, en [13] desarrollan un modelo de factorización probabilística de matrices no lineal con procesos Gaussianos. Ambas propuestas tratan en específico el problema particular de completación de matrices para filtrado colaborativo.

4.2. Factorización probabilística de matrices

Supongamos que tenemos una matriz parcial, $F \in \mathbb{R}^{N\times M}$ con entradas conocidas F_{ij} con $(i, j) \in \Omega$, donde $\Omega \subset \{(i, j)|1 \leq i \leq N, 1 \leq j \leq M\}$.

Salakhutdinov y Mnih en [14] y [15] proponen un método de factorización de rango bajo, como ha sido propuesto a través de optimización convexa, pero desde un enfoque probabilístico. Esta factorización probabilística de matrices, es un modelo probabilístico lineal con ruido Gaussiano.

La idea detrás de esta propuesta consiste en encontrar dos matrices $U \in \mathbb{R}^{D\times N}$ y $V \in \mathbb{R}^{D\times M}$, tales que $F = U'V$. Sean U_i, V_j con $i \in \{1, 2, \ldots, N\}$ e $j \in \{1, 2, \ldots, M\}$, vectores columna correspondientes a la $i-$ésima columna de U y $j-$ésima columna de V, respectivamente. En este modelo se define la distribución condicional sobre F_{ij} con $(i, j) \in \Omega$, y las distribuciones de los prior sobre $U \in \mathbb{R}^{D\times N}$ y $V \in \mathbb{R}^{D\times M}$ como:

$$p\left(F|U, V, \alpha\right) = \prod_{i=1}^{N}\prod_{j=1}^{M}\left[N\left(F_{ij}|U_i'V_j, \alpha^{-1}\right)\right]^{I_{ij}}, \tag{4.1}$$

$$p\left(U|\mu_U, \Sigma_U\right) = \prod_{i=1}^{N} N\left(U_i|\mu_U, \Sigma_U\right), \tag{4.2}$$

$$p\left(V|\mu_V, \Sigma_V\right) = \prod_{j=1}^{M} N\left(V_j|\mu_V, \Sigma_V\right), \tag{4.3}$$

donde $N(x|\mu, \Sigma)$ denota la distribución normal con media μ y matriz de covarianza

Σ, I_D denota la matriz identidad de dimensión D e I_{ij} es una función indicadora la cual es igual a 1, si la entrada F_{ij} es conocida e igual a 0, en otro caso.

En [15] se considera $\mu_U = 0$, $\mu_V = 0$ y $\Sigma_U^{-1} = \alpha_U^{-1} I_D$, $\Sigma_V^{-1} = \alpha_V^{-1} I_D$ matrices diagonales. La estimación en este modelo se realiza mediante la maximización del log-posterior sobre U y V con hiperparámetros fijos (i.e la varianza del ruido y las varianzas de los prior):

$$\ln p\left(U, V | F, \alpha, \alpha_V, \alpha_U\right) = \ln p\left(F | U, V, \alpha\right) + \ln\left(U | \alpha_U\right) + \ln\left(V | \alpha_V\right) + C, \tag{4.4}$$

donde C es una constante que no depende de los parámetros.

La maximización de (4.4) es equivalente a minimizar la función de la suma de los errores al cuadrado con términos de regularización cuadráticos:

$$E = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} I_{ij} \left(F_{ij} - U_i' V_j\right)^2 + \frac{\lambda_U}{2} \sum_{i=1}^{N} \|U_i\|_{Fro}^2 + \frac{\lambda_V}{2} \sum_{j=1}^{M} \|V_j\|_{Fro}^2, \tag{4.5}$$

donde $\lambda_U = \alpha_U / \alpha$, $\lambda_V = \alpha_V / \alpha$ y $\|\cdot\|_{Fro}^2$ denota la norma Frobenius. Un mínimo local de la función definida en (4.5) se puede encontrar usando algún método de optimización basado en gradientes sobre U y V.

En este modelo es necesario controlar la complejidad manualmente, buscando valores apropiados de los parámetros de regularización λ_U y λ_V. Por ejemplo, en un conjunto razonable de valores para los parámetros, se podría estimar un modelo para cada ajuste de los parámetros, y elegir el modelo que interpreta mejor un conjunto de validación. Este enfoque es computacionalmente muy costoso, dado que requiere de la estimación de muchos modelos en lugar de estimar solo uno. Alternativamente, podríamos introducir priors para los hiperparámetros y maximizar el log-posterior del modelo sobre los parámetros e hiperparámetros, lo que permite que la complejidad del modelo sea controlada de forma automática basa-

da en los datos. Se ha demostrado que este enfoque funciona en la práctica pero no esta bien fundamentado teóricamente, y es posible construir ejemplos en los que no produce los resultados deseados.

Basados en este inconveniente en [14] se propone un tratamiento totalmente Bayesiano del modelo de factorización probabilístico de matrices (BPMF). En este caso se consideran priors Gaussianos-Wishart para los hiperparámetros, tomando $\Theta_U = (\mu_U, \Sigma_U)$ y $\Theta_V = (\mu_V, \Sigma_V)$:

$$\begin{aligned} p(\Theta_U|\Theta_0) &= p(\mu_U|\Sigma_U)\, p(\Sigma_U) \\ &= N\left(\mu_U|\mu_0, (\beta_0\Sigma_U)^{-1}\right) W(\Sigma_U|W_0, \nu_0), \end{aligned} \tag{4.6}$$

$$\begin{aligned} p(\Theta_V|\Theta_0) &= p(\mu_V|\Sigma_V)\, p(\Sigma_V) \\ &= N\left(\mu_V|\mu_0, (\beta_0\Sigma_V)^{-1}\right) W(\Sigma_V|W_0, \nu_0), \end{aligned} \tag{4.7}$$

donde $\Theta_0 = \{\mu_0, \beta_0, \nu_0, W_0\}$ y W es la distribución de Wishart con ν_0 grados de libertad y matriz de escala W_0 de dimensión $D \times D$.

La distribución predictiva de F^*_{ij}, es obtenida por la marginalización sobre los parámetros e hiperparámetros del modelo:

$$p\left(F^*_{ij}|F, \Theta_0\right) = \int\int p\left(F^*_{ij}|U_i, V_j\right) p(U, V|F, \Theta_U, \Theta_V)\, p(\Theta_U, \Theta_V|\Theta_0)\, d\{U, V\}\, d\{\Theta_U, \Theta_V\} \tag{4.8}$$

Como la evaluación exacta de la distribución predictiva es analíticamente intratable debido la complejidad del posterior, debemos recurrir a inferencia aproximada. Los algoritmos MCMC pueden ser usados para aproximar la distribución predictiva de la ecuación (4.8) a través de:

$$p\left(F^*_{ij}|F, \Theta_0\right) \cong \frac{1}{T}\sum_{t=1}^{T} p\left(F^*_{ij}|U_i^{(t)}, V_j^{(t)}\right), \tag{4.9}$$

donde las muestras $\left\{U_i^{(t)}, V_j^{(t)}\right\}$ son generadas mediante la ejecución de una cadena de Markov cuya distribución estacionaria es la distribución posterior a través de los parámetros e hiperparámetros del modelo $\{U, V, \Theta_U, \Theta_V\}$.

En [14] los autores proponen utilizar un muestreador de Gibbs, el cual como conocemos es un algoritmo MCMC, para encontrar las muestras deseadas y aproximar la distribución predictiva (4.8) mediante (4.9). El cual tiene la siguiente forma:

Muestreador de Gibbs para BPMF

1. Inicializar los parámetros del modelo $\left\{U^1, V^1\right\}$.
2. Para t=1,. . . ,T

- Obtener muestras de los parámetros:

$$\begin{aligned} \Theta_U^t &\sim p\left(\Theta_U | U^t, \Theta_0\right) \\ \Theta_V^t &\sim p\left(\Theta_V | V^t, \Theta_0\right) \end{aligned}$$

- Para cada $i = 1, \ldots, N$:

$$U_i^{t+1} \sim p\left(U_i | F, V^t, \Theta_U^t\right)$$

- Para cada $i = 1, \ldots, M$:

$$V_i^{t+1} \sim p\left(V_i | F, U^{t+1}, \Theta_V^t\right)$$

4.3. Un método para resolver el problema de completación de matrices

Dada una matriz parcial $F \in \mathbb{R}^{N\times M}$, con entradas conocidas F_{ij} con $(i, j) \in \Omega$, donde $\Omega \subset \{(i, j)|1 \leq i \leq N, 1 \leq j \leq M\}$. Con el método propuesto deseamos encontrar dos matrices $U \in \mathbb{R}^{D\times N}$ y $V \in \mathbb{R}^{D\times M}$ tales que: $F = U'V$.

Consideremos que las filas de la matriz F están formadas por N muestras de un $M-$vector aleatorio, proveniente de una distribución normal con media μ y matriz de covarianza Σ; dado que F es una matriz parcial tenemos un problema de datos incompletos. La implementación del algoritmo EM para poblaciones normales (ver capítulo 2, sección 2.1.2) nos permite estimar los estimadores de máxima verosimilitud $\widehat{\mu}$, $\widehat{\Sigma}$ de μ y Σ, respectivamente.

Con la estimación $\widehat{\Sigma}$ de Σ es posible obtener la matriz $V \in \mathbb{R}^{D\times M}$ deseada; tomando V' como una matriz cuyas columnas corresponden a los D vectores propios asociados a los D mayores autovalores $\lambda_1, \ldots, \lambda_D$ de $\widehat{\Sigma}$. Así, por análisis de componentes principales (ver capítulo 1, sección 1.1.3), tenemos la siguiente relación:

$$U' = F\ V',$$

donde $VV^t \simeq I_D$ y el valor D es escogido de tal forma que $\lambda_1 + \cdots + \lambda_D$ represente el porcentaje de varianza que se desea capturar.

De este modo, solo resta estimar la matriz $U \in \mathbb{R}^{D\times N}$. Considerando el modelo PMF dado por las ecuaciones (4.1), (4.2) y (4.3), donde $U_1, \ldots, U_N$ se considerada una muestra vectorial, suponemos que $U_1, \ldots, U_N$ constituyen un conjunto de vectores aleatorios estadísticamente independientes con $U_i \sim N(\mu_{U_i}, \Sigma_{U_i})$, $\forall i \in \{1, \ldots, N\}$. Esto permite implementar de forma sencilla un algoritmo SAEM para cada U_i para estimar $\widehat{\mu}_{U_i}$, $\widehat{\Sigma}_{U_i}$, los estimadores de máxima verosimilitud de

μ_{U_i} y Σ_{U_i}, respectivamente.

En este caso, la distribución condicional sobre F_{ij} con $(i, j) \in \Omega$ se puede escribir como:

$$\mathcal{L}_{U_i}(F) = \prod_{j=1}^{M} \left[N\left(F_{ij} | U_i' V_j, \alpha^{-1}\right) \right]^{I_{ij}},$$

donde $N(x|\mu, \sigma^2)$ denota la distribución normal con media μ y varianza σ^2, I_D denota la matriz identidad de dimensión D e I_{ij} es una función indicadora la cual es igual a 1, si la entrada F_{ij} es conocida e igual a 0, en otro caso.

Observación 4.3.1 *La varianza del ruido α^{-1} se considera conocida. Además, en este paso la matriz $V \in \mathbb{R}^{D \times M}$ ha sido obtenida.*

Los datos completos son $x = (F, U_i)$, donde la matriz parcial F son las observaciones. Supongamos que Σ_{U_i} es definida positiva, así un estadístico suficiente para $\Theta_{U_i} = (\mu_{U_i}, \Sigma_{U_i})$ es $S(\Theta_{U_i}) = (S(\mu_{U_i}), S(\Sigma_{U_i}))$, donde:

$$S(\mu_{U_i}) = \sum_{j=1}^{m} U_{ij}, \tag{4.10}$$

$$S(\Sigma_{U_i}) = \sum_{j=1}^{m} U_{ij} U_{ij}', \tag{4.11}$$

y las muestras $\left\{U_{ij}\right\}_{j=1}^{m}$ son generadas mediante la ejecución de una cadena de Markov cuya distribución estacionaria corresponde a la distribución posterior $\mathcal{L}_{U_i}(F)\, N(U_i | \mu_{U_i}, \Sigma_{U_i})$.

Como la distribución de U_i con $i \in \{1, \ldots, N\}$, pertenece a la familia exponencial, el paso de aproximación del algoritmo SAEM para estimar los parámetros de máxima verosimilitud de μ_{U_i} y Σ_{U_i} de cada U_i, se reduce a actualizar los estadísticos suficientes (4.10) y (4.11). Por su parte, el paso de maximización consiste en actualizar los estimadores de máxima verosimilitud, en función de las actualizaciones de los estadísticos suficientes. El algoritmo SAEM utilizado tendrá la

siguiente forma:

Algoritmo SAEM

Inicialización:

- Proporcionar una tolerancia $\epsilon > 0$.
- Calcular: $\widehat{\mu}_{U_i}^{(0)} = V\,\widehat{\mu}$, $\widehat{\Sigma}_{U_i}^{(0)} = V\widehat{\Sigma}V'$.
- Hacer: $U_i^{(0)} = \widehat{\mu}_{U_i}^{(0)}$.

Para $k \geq 1$, haga mientras $|\mathcal{L}_{\widehat{U_i}^{(k-1)}}(F) - \mathcal{L}_{\widehat{U_i}^{(k)}}(F)| > \epsilon$:

1. **Simulación:** Realizar $m(k)$ iteraciones de un algoritmo Metropolis-Hastings, para obtener $U_{i1}^{(k)}, \ldots, U_{im(k)}^{(k)}$.

2. **Aproximación estocástica:** Actualizar $S(\mu_{U_i})^{(k-1)}$ y $S(\Sigma_{U_i})^{(k-1)}$, con:

$$\begin{aligned} S(\mu_{U_i})^{(k)} &= S(\mu_{U_i})^{(k-1)} + \gamma_k \left(\sum_{j=1}^{m(k)} U_{ij}^{(k)} - S(\mu_{U_i})^{(k-1)} \right), \\ S(\Sigma_{U_i})^{(k)} &= S(\Sigma_{U_i})^{(k-1)} + \gamma_k \left(\sum_{j=1}^{m(k)} U_{ij}^{(k)} U_{ij}^{(k)'} - S(\Sigma_{U_i})^{(k-1)} \right), \end{aligned}$$

   donde $\{\gamma_k\}_{k\geq 1}$ es una sucesión decreciente de tamaños de pasos positivos, la cual converge a 0 y satisface que $\sum_k \gamma_k = \infty$.

3. **Maximización:** Actualizar $\widehat{\mu}_{U_i}^{(k-1)}$ y $\widehat{\Sigma}_{U_i}^{(k-1)}$, con:

$$\begin{aligned} \widehat{\mu}_{U_i}^{(k)} &= \frac{1}{m(k)} S(\mu_{U_i})^{(k)}, \\ \widehat{\Sigma}_{U_i}^{(k)} &= \frac{1}{m(k)} S(\Sigma_{U_i})^{(k)} - \widehat{\mu}_{U_i}^{(k)} \widehat{\mu}_{U_i}^{(k)'} \end{aligned}$$

4. Hacer: $U_i^{(k)} = \widehat{\mu}_{U_i}^{(k)}$.

Observación 4.3.2 *Puede ser necesario considerar una regularización de las matrices* $\widehat{\Sigma}_{U_i}^{(k)}$*, para* $k \geq 1$*; haciendo:*

$$\widehat{\Sigma}_{U_i}^{(k)} = \widehat{\Sigma}_{U_i}^{(k)} + \widehat{\epsilon}\, I_D,$$

donde $\widehat{\epsilon}$ *es una constante pequeña.*

Esto garantiza que cada matriz $\widehat{\Sigma}_{U_i}^{(k)}$ *sea de rango completo.*

Observación 4.3.3 *La sucesión de tamaños de pasos positivos* $\{\gamma_k\}_{k\geq 1}$*, puede estar definida de muchas maneras, por ejemplo:*

1.

$$\gamma_k = \frac{1}{k},\ \forall k \geq 1.$$

2.

$$\gamma_k = \begin{cases} 1, & si\ 1 \leq k \leq K_1 \\ \frac{1}{k-K_1}, & si\ k > K_1 \end{cases},$$

donde K_1 *es dado.*

Las cuales han mostrado ser buenas escogencias en muchos casos prácticos.

La utilización del algoritmo SAEM posee ventajas, dadas sus buenas propiedades teóricas; además, en la práctica ha mostrado ser menos costoso computacionalmente, comparado con algoritmos similares.

Capítulo 5

Resultados numéricos

En el presente capítulo se exponen algunos resultados de la implementación del método propuesto, realizando dos tipos de experimentos con matrices simuladas (generadas aleatoriamente). En ambos casos realizamos comparaciones de su rendimiento; en los experimentos numéricos que llamaremos de tipo 1, comparamos con un método de lagrangeano aumentado inexacto (inexact ALM), propuesto por Chen y colaboradores en [12], además del método de factorización probabilística Bayesiana de matrices (BPMF), expuesto en el capítulo 4 y propuesto por Salakhutdinov y Mnih en [14]. En los experimentos numéricos de tipo 2, compararemos sólo con este último. Todos los programas empleados se encuentran desarrollados en MATLAB[1], hemos utilizado la versión R2011a de éste.

La implementación del método propuesto es realizada como se ha descrito en el capítulo 4, sección 4.3. En la implementación del algoritmo EM para poblaciones normales, se utilizó una tolerancia $\epsilon_1 = 1e-4$ y 500 como número máximo de iteraciones. El rango de la descomposición es seleccionado considerando el porcentaje de varianza que se desea capturar. Luego de obtener la matriz V desea-

[1]Los programas utilizados en la implementación de los métodos inexact ALM y BPMF han sido proporcionados por sus autores y se encuentran disponibles en:
http://perception.csl.illinois.edu/matrix-rank/home.html
http://www.cs.toronto.edu/~rsalakhu/BPMF.html

da procedemos a estimar la matriz U, donde una iteración del algoritmo SAEM utilizado para estimar una columna de esta matriz, realiza k iteraciones del algoritmo Metropolis-Hastings. Consideramos una tolerancia $\epsilon_2 = 1e-4$ y un número máximo de iteraciones de 25, las cuales no suelen ser alcanzadas, dada la rápida convergencia.

Es importante mencionar que el muestreador de Gibbs empleado en el método BPMF (ver capítulo 4, sección 4.2) es inicializado con los parámetros del modelo $\left\{U^1, V^1\right\}$ obtenidos de la solución del método PMF, lo que trae consigo algunos de los inconvenientes relacionados a éste. En este algoritmo se realizan 1000 iteraciones para obtener los resultados.

Tanto en el método propuesto como en BPMF la varianza del ruido es fijado en $\alpha^{-1} = \frac{1}{2}$.

5.1. Experimentos numéricos de tipo 1

Para realizar estos experimentos crearemos matrices parciales de forma aleatoria por el siguiente procedimiento:

1. Generamos dos matrices aleatorias $U \in \mathbb{R}^{D\times N}$ y $V \in \mathbb{R}^{D\times M}$ con entradas independiente e idénticamente distribuidas provenientes de una distribución normal estándar.
2. Hacemos $F = U'V$.
3. Borramos aleatoriamente alrededor del 50 % de las entradas de la matriz F, para obtener una matriz parcial $\widetilde{F}$.

Luego de la estimación calculamos el error relativo como:

$$\frac{\|F - \widehat{F}\|_{\mathcal{F}}}{\|F\|_{\mathcal{F}}},$$

donde $\widehat{F}$ es una completación para la matriz parcial $\widetilde{F}$ y $\|\cdot\|_{\mathcal{F}}$ denota la norma de Frobenius.

En la tabla 5.1, mostramos los resultados del cálculo del error relativo (**Error Rel.**), el rango seleccionado de la descomposición (**Rg. est.**) y el tiempo de ejecución en segundos (**T. Ej. (seg)**) para el método propuesto, BPMF e inexact ALM. Esto, para matrices parciales $\widetilde{F} \in \mathbb{R}^{N\times M}$, generadas según el procedimiento descrito. Denotamos al método propuesto por "**M. propuesto (1)**" cuando el número de iteraciones realizadas por el algoritmo Metropolis-Hastings es 50, y por "**M. propuesto (2)**" si el número de éstas iteraciones es 100.

	N	M	D	Error Rel.	Rg. est.	T. Ej. (seg)
M. propuesto (1)	50	30	5	0.5191	5	22.7892
M. propuesto (2)	50	30	5	0.2652	5	41.8364
BPMF	50	30	5	1.1494	-	12.1142
inexact ALM	50	30	5	0.4804	29	0.3378
M. propuesto (1)	100	80	30	0.8301	21	56.5566
M. propuesto (2)	100	80	30	0.6739	21	220.8104
BPMF	100	80	30	1.2111	-	87.8891
inexact ALM	100	80	30	0.5482	61	1.5338
M. propuesto (1)	180	120	60	0.9388	36	59.1361
M. propuesto (2)	180	120	60	0.8779	36	281.0380
BPMF	180	120	60	1.2461	-	489.6364
inexact ALM	180	120	60	0.6031	120	2.6289

Tabla 5.1: Comparación entre el método propuesto, BPMF e inexact ALM en experimentos numéricos de tipo 1.

En los resultados de BPMF en la columna correspondiente a **Rg. est.**, hemos empleado el símbolo "-" para indicar que este método no realiza el cálculo. Cuando mencionamos el rango seleccionado de la descomposición, hacemos referencia al valor D seleccionado. En el método BPMF este valor debe ser fijado por el experimentador y en sus implementaciones para estos experimentos, hemos usado el valor conocido de D.

Por su parte, el método propuesto realiza el cálculo del valor D, considerando que el porcentaje de varianza que se desea capturar es de 95 %. Podemos notar que los valores de **Rg. est.**, obtenidos con el método propuesto son más cercanos al verdadero (valor D) que en los resultados de inexact ALM.

Se ha considerado la implementación del método propuesto con diferentes números de iteraciones del algoritmo Metropolis-Hastings (50 y 100) para evidenciar la importancia de realizar un número suficiente. Como se puede notar al realizar 100 iteraciones de éste algoritmo se obtienen menores errores relativos.

Podemos observar en la tabla 5.1 que el error con el método BPMF es siempre mayor, mostrando que este método no es eficaz con esta clase de matrices parciales (las cuales poseen entradas no enteras y algunas negativas). En relación al método propuesto (cuando se realizan 100 iteraciones del algoritmo Metropolis-Hastings) e inexact ALM, sus errores relativos pueden ser considerados cercanos. Aunque los tiempos de ejecución de inexact ALM son considerablemente menores, como ya mencionamos el valor de **Rg. est.** en todos los casos es más cercano al verdadero con el método propuesto.

5.2. Experimentos numéricos de tipo 2

En estos experimentos crearemos de forma aleatoria matrices parciales, simulando problemas de filtrado colaborativo. Este problema comprende datos acerca de las calificaciones dadas a ciertos artículos, pero ya que los usuarios solo califican un subgrupo de estos (generalmente muy pocos), la matriz de preferencias posee pocas entradas, así el problema radica en completar esta matriz para que el vendedor pueda hacer recomendaciones a sus usuarios.

Observación 5.2.1 *En los problemas de filtrado colaborativo por lo general, ca-*

da fila de la matriz de preferencias representa un usuario y cada columna un artículo. Sea F una matriz de preferencias. Si existen N usuarios y M artículos entonces, $F \in \mathbb{R}^{N\times M}$ y es una matriz parcial.

Para crear las matrices parciales de estos experimentos seguiremos el siguiente procedimiento:

1. Generamos aleatoriamente matrices $\widetilde{F} \in \mathbb{R}^{N\times M}$ con entradas enteras de $r_{\text{mín}}$ a $r_{\text{máx}}$ y solo alrededor del 30 % de sus entradas.

2. Seleccionamos por cada fila (la cual representa a un usuario) una calificación conocida, con estas calificaciones formamos un subconjunto Ω de las entradas conocidas de $\widetilde{F}$.

3. Borramos las entradas de la matriz $\widetilde{F}$ pertenecientes al conjunto Ω, para obtener una nueva matriz parcial F. Al conjunto Ω lo llamaremos conjunto de prueba, y al conjunto restante de entradas conocidas se le llamará conjunto de estimación.

Luego de realizar la estimación con algún método, se evalúa su rendimiento en el conjunto de prueba a través del cálculo del error absoluto medio normalizado (NMAE), definido como:

$$NMAE = \frac{1}{(r_{\text{máx}} - r_{\text{mín}})\,|\Omega|} \sum_{(i,j)\in\Omega} |F_{ij} - \widehat{F}_{ij}|,$$

donde $\widehat{F}$ es una completación para la matriz parcial F.

En nuestro caso hemos considerado $r_{\text{mín}} = 1$ y $r_{\text{máx}} = 5$.

En la tabla 5.2, presentamos los resultados de la implementación del método propuesto y de BPMF, para matrices parciales $F \in \mathbb{R}^{N\times M}$ generadas según el procedimiento descrito para los experimentos numéricos de tipo 2. En la tercera columna se encuentran los errores absoluto medio normalizado (**NMAE**) para

cada caso, y en la última columna los tiempos de ejecución correspondientes, en segundos (**T. Ej. (seg)**).

En estos experimentos, el rango de la descomposición para el método BPMF es fijado en 30 para las matrices parciales de orden 50×30, para las matrices de orden 120×100 y 250×180 es fijado en 60, y para las de orden 500×350 es fijado en 120.

Con el método propuesto el rango de la descomposición es seleccionado considerando que se desea capturar el 99 % de la varianza, y se realizaron 100 iteraciones del algoritmo Metropolis-Hastings en cada iteración del algoritmo SAEM utilizado para estimar una columna de la matriz U.

	N	**M**	**NMAE**	**T. Ej. (seg)**
M. propuesto	50	30	0.2950	14.5384
BPMF	50	30	0.3900	73.1588
M. propuesto	120	100	0.2729	40.2140
BPMF	120	100	0.3000	326.4987
M. propuesto	250	180	0.2800	516.2469
BPMF	250	180	0.2570	$1.2523e+003$
M. propuesto	500	350	0.2820	$3.8892e+003$
BPMF	500	350	0.2565	$6.2269e+003$

Tabla 5.2: Comparación entre el método propuesto y BPMF en experimentos numéricos de tipo 2.

Con los resultados del cálculo de **NMAE**, mostrados en la tabla 5.2, se puede notar que el método propuesto presenta mejores resultados en dos de los casos, y aunque produzca por una diferencia pequeña un mayor **NMAE** en los últimos de éstos, debemos considerar que en todos ellos los tiempos de ejecución son considerablemente menores.

Conclusiones

En este trabajo hemos propuesto un método para resolver problemas de completación de matrices, donde tenemos una matriz parcial $F \in \mathbb{R}^{N\times M}$ la cual deseamos completar, encontrando dos matrices $U \in \mathbb{R}^{D\times N}$ y $V \in \mathbb{R}^{D\times M}$ tales que $F = U'V$. La implementación de este método incluye el uso de un análisis de componentes principales y la implementación de un algoritmo EM, para encontrar la matriz V deseada. Para estimar la matriz U, se diseñó e implemento un algoritmo SAEM, el cual es utilizado para estimar una a una las columnas de esta matriz. Además, hemos incluido una revisión exhaustiva de cada una de las técnicas empleadas.

De los resultados numéricos mostrados en el capítulo 5, podemos notar que el método propuesto se muestra competitivo con respecto al método BPMF propuesto en [14]. Debemos destacar que sus tiempos de ejecución son considerablemente menores, además no requiere escoger parámetros de regularización como en el método PMF, algo que lo hace muchas veces difíciles de implementar, y con el cual se obtiene los parámetros iniciales del modelo para implementar el método BPMF. En relación al método inexact ALM propuesto en [12] con el cual realizamos comparaciones en los experimentos numéricos de tipo 1, podemos mencionar que si bien los tiempos de ejecución de esté son menores y en algunos casos se obtienen mejores errores relativos, con el método propuesto hemos obtenido mejores aproximaciones del rango de la descomposición.

Como estudios futuros relacionados a este trabajo, se puede considerar el estudio de métodos o técnicas que mejoren el proceso de estimación de la matriz $\widehat{\Sigma}$, ya que la implementación del algoritmo EM para poblaciones normales muchas veces resulta costosa computacionalmente.

Apéndice A

Códigos desarrollados en MATLAB

A.1. CP_SAEM.m

Esta función resuelve el problema de completar una matriz parcial, siguiendo el método propuesto.

Entradas:

- `F`: matriz parcial.
- `tol`: tolerancia (>0) para los algoritmos SAEM.

Salidas:

- `F_est`: completación resultante de `F`.
- `U`, `V`: matrices encontradas tales que `F_est=U'V`.

```
1 function [F_est,U,V]=CP_SAEM(F,tol)
2 N=size(F,1);
3
4 %Estimación de la media y matriz de covarianza (para ...
    poblaciones normales):
```

```
5  [Mean, Covar,Iteration] = EM(F,'twostage',500,1e-4);
6  %Obs: 'twostage' indica que la media inicial se calcula
7  %con los datos disponibles y que la matriz de covarianza
8  %inicial es calculada luego de sustituir las entradas
9  % desconocidas de F por la media inicial. 500 es el
10 %número de iteraciones máximas y 1e-4 la tolerancia.
11
12 fprintf('Número de iteraciones realizadas por el EM: %i',...
       Iteration);
13
14 [Vec,Aut] = eig(Covar);
15 Aut=diag(Aut);
16 Naut=length(Aut);
17
18 %Escogencia de D:
19 for k=1:Naut
20     f=(sum(Aut(1:k))*100)/sum(Aut);
21     D=k;
22
23     if f>=99 %este valor puede ser cambiado según el % de ...
           varianza que se desee capturar
24         break
25     end
26 end
27
28 fprintf('Rango seleccionado de la descomposición: %i',D);
29
30 %Construcción de la matriz V por componentes principales:
31 V=Vec(:,1:D)';
32
```

```
33 %Número de iteraciones a realizar con el Metropolis-...
      Hastings:
34 Nsim=100;
35
36 %Número de iteraciones máximas a realizar con el SAEM:
37 Nmax=25;
38
39 %Inicialización de los parámetros:
40 MeanU=V*Mean;
41 CovarU=V*Covar*V';
42
43 %Inicializando la matriz U:
44 U=repmat(MeanU,1,N);
45
46 for i=1:N
47
48    diff=1;
49
50    %Inicialización:
51    Ui=U(:,i);
52
53    %Inicialización de los parámetros del vector Ui:
54    MeanUi=MeanU;
55    CovarUi=CovarU;
56
57    %Contador de la iteraciones realizadas por el SAEM:
58    k=0;
59
60    AproxMean=zeros(D,1);
61    AproxCovar=zeros(D,D);
```

```
62
63    %Cálculo de la log-verosimilitud de Ui inicial:
64    [L0]=logVerosimilitud(F,Ui,V,i);
65
66    while diff>tol && k<Nmax
67
68        k=k+1;
69
70        %Cálculo de las aproximaciones estocástica de los ...
              estadísticos suficientes de los parámetros:
71        [AproxMean,AproxCovar]=AproxEstocasticas(F,Ui,...
              AproxMean,AproxCovar,V,MeanUi,CovarUi,Nsim,k,i);
72
73        %Actualización de los parámetros:
74        [MeanUi,CovarUi]=ActHiperparametros(AproxMean,...
              AproxCovar,Nsim);
75
76        %Actualización de Ui:
77        Ui=MeanUi;
78
79        %Cálculo de la log-verosimilitud de Ui:
80        [L1]=logVerosimilitud(F,Ui,V,i);
81
82        diff=abs(L1-L0);
83        L0=L1;
84    end
85
86    str = sprintf('Fueron necesarias %i iteraciones del ...
          SAEM para estimar la columna %i de U',k,i);
87    disp(str);
```

```
88      U(:,i)=MeanUi;
89      clear MeanUi CovarUi diff
90  end
91  F_est=U'*V;
```

A.2. AproxEstocasticas.m

Esta función calcula las actualizaciones de las aproximaciones estocásticas de los estadísticos suficientes de la media y matriz de covarianza de un vector U_i con $i \in \{1, \ldots, N\}$, correspondientes a una iteración del algoritmo SAEM.

Entradas:

- `F`: matriz parcial.
- `Ui`: estimación actual del vector `Ui` (i-ésima columna de `U`).
- `AproxMean`: aproximación estocástica actual del estadístico suficiente de la media de `Ui`.
- `AproxCovar`: aproximación estocástica actual del estadístico suficiente de la matriz de covarianza de `Ui`.
- `V`: matriz deseada obtenida por medio de componentes principales.
- `MeanU`: estimación actual del estimador de máxima verosimilitud de la media de `Ui`.
- `CovarU`: estimación actual del estimador de máxima verosimilitud de la matriz de covarianza de `Ui`.
- `Nsim`: número de iteraciones a realizar con el algoritmo Metropolis-Hastings.
- `k`: número de la iteración que se esta realizando del algoritmo SAEM.

- `i`: número de la columna de la matriz `U` que se ésta estimando.

Salidas:

- `AproxMean`: actualización de la aproximación estocástica del estadístico suficiente de la media de `Ui`.
- `AproxCovar`: actualización de la aproximación estocástica del estadístico suficiente de la matriz de covarianza de `Ui`.

```
1  function [AproxMean,AproxCovar]=AproxEstocasticas(F,Ui,...
       AproxMean,AproxCovar,V,MeanU,CovarU,Nsim,k,i)
2  K1=5; %se puede escoger otro valor de K1, por ejemplo 10
3
4  [Sum_Ui,Sum_UiUi]=MetropolisHastings(Ui,F,V,MeanU,CovarU,...
       Nsim,i);
5
6  if k<=K1
7      c=1;
8  else
9      c=1/(k-K1);
10 end
11
12 %otra escogencia para la sucesión decreciente de tamaños ...
       de pasos
13 %positivos:
14 % c=1/k;
15
16 %Aproximación estocástica de los estadísticos suficientes ...
       para la media
17 %y matriz de covarianza:
18 AproxMean=AproxMean+c*(Sum_Ui-AproxMean);
19 AproxCovar=AproxCovar+c*(Sum_UiUi-AproxCovar);
```

A.3. MetropolisHastings.m

La siguiente función corresponde al algoritmo Metropolis-Hasting el cual es utilizado para obtener muestras del vector U_i con $i \in \{1, \ldots, N\}$ y calcular los estadísticos suficientes de sus parámetros.

Entradas:

- `Ui`: estimación actual del vector `Ui` (i-ésima columna de `U`).
- `F`: matriz parcial.
- `V`: matriz deseada obtenida por medio de componentes principales.
- `MeanU`: estimación actual del estimador de máxima verosimilitud de la media de `Ui`.
- `CovarU`: estimación actual del estimador de máxima verosimilitud de la matriz de covarianza de `Ui`.
- `Nsim`: número de iteraciones a realizar con el algoritmo Metropolis-Hastings.
- `i`: número de la columna de la matriz `U` que se ésta estimando.

Salidas:

- `Sum_Ui`: suma de los vectores generados aceptados en el algoritmo Metropolis-Hasting.
- `Sum_UiUi`: suma del producto de los vectores generados aceptados en el algoritmo Metropolis-Hasting con su vector transpuesto.

```
1 function [Sum_Ui,Sum_UiUi]=MetropolisHastings(Ui,F,V,MeanU...
     ,CovarU,Nsim,i)
2 c=0.1; %escalar que permite re-escalar la matriz CovarU
```

```
3  D=length(Ui);
4
5  %Evaluando la función de verosimilitud:
6  [loglike]=logVerosimilitud(F,Ui,V,i);
7
8  %Evaluando el prior para U:
9  [logpri]=prior(Ui,MeanU,CovarU);
10
11 acceptance=0;
12 Sum_Ui=zeros(D,1);
13 Sum_UiUi=zeros(D,D);
14
15 for k=1:Nsim
16
17    lam = chol((c^2)*CovarU);
18    Uf = lam*randn(D,1)+Ui;
19
20    [newlogpri]=prior(Uf,MeanU,CovarU);
21
22    if newlogpri>-Inf
23       [newloglike]=logVerosimilitud(F,Uf,V,i);
24
25       if newloglike==-Inf;
26          ratio=0;
27       else
28          ratio=exp((newloglike+newlogpri)-(loglike+logpri...
               )); %radio de aceptación
29       end
30
31       if rand<=ratio;
```

```
32            loglike=newloglike;
33            logpri=newlogpri;
34            Ui=Uf;
35            acceptance=acceptance+1;
36        end
37    end
38    Sum_Ui=Ui+Sum_Ui;
39    Sum_UiUi=Ui*Ui'+Sum_UiUi;
40 end
41
42 % Cálculo de la tasa de aceptación:
43 G=(acceptance/Nsim)*100;
44 str = sprintf('La tasa de aceptación de la columna %i es ...
      de %g porciento',i,G);
45 disp(str);
46 clear Uf Ui
```

A.4. ActHiperparametros.m

La función `ActHiperparametros.m` es la encargada de actualizar las estimaciones de los estimadores de máxima verosimilitud de los parámetros del vector U_i con $i \in \{1, \ldots, N\}$, en cada iteración del correspondiente algoritmo SAEM .

Entradas:

- `AproxMean`: aproximación estocástica actual del estadístico suficiente de la media de `Ui`.
- `AproxCovar`: aproximación estocástica actual del estadístico suficiente de la matriz de covarianza de `Ui`.
- `Nsim`: número de iteraciones a realizar con el algoritmo Metropolis-Hastings.

Salidas:

- `MeanU`: actualización de la estimación del estimador de máxima verosimilitud de la media de `Ui`.
- `CovarU`: actualización de la estimación del estimador de máxima verosimilitud de la matriz de covarianza de `Ui`.

```
1 function [MeanU,CovarU]=ActHiperparametros(AproxMean,...
     AproxCovar,Nsim)
2 MeanU=(1/Nsim)*AproxMean;
3 CovarU=(1/Nsim)*AproxCovar-MeanU*MeanU';
4
5 %Regularización de CovarU:
6 D=length(AproxMean);
7 epsilon = 1e-8;
8 CovarU=CovarU+epsilon.*eye(D);
```

REFERENCIAS

[1] LLatas I. Bravo L. and Pérez E. *Análisis de datos con técnicas Bayesianas*. XXI escuela venezolana de matemáticas, Mérida, Venezuela, 2008.

[2] Peña D. *Análisis de datos multivariantes*. McGraw-Hill, Madrid, España, 2002.

[3] Lavielle M. Delyon B. and Moulines E. Convergence of a stochastic approximation version of the em algorithm. *The Annals of Statistics*, 27(1):94–128, 1999.

[4] Laird N. Dempster A. and Rubin D. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological).*, 39(1):1–38, 1977.

[5] Candès E. and Recht B. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[6] Kuhn E. and Lavielle M. Coupling a stochastic approximation version of em with a mcmc procedure. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2003.

[7] Wei G. and Tanne M. A monte carlo implementation of the em algorithm and the door man's data argumentation algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990.

[8] Kushner H. and Yin G. *Stochastic Approximation Recursive Algorithms and Applications*. Second Edition. Springer, New York, EE.UU, 2003.

[9] Robbins H. and Monro S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[10] Taylor H. and Karlin S. *An Introduction to Stochastic Modeling*. Academic Press, Londrés, Reuno Unido, 1998.

[11] Yu G. Léger F. and Sapiro G. Efficient matrix completion with gaussian models. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2011.

[12] Wu L. Lin Z., Chen M. and Ma Y. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. 2009.

[13] Lawrence N. and Urtasun R. Non-linear matrix factorization with gaussian processes. *In Proc. ICML*, 2009.

[14] Salakhutdinov R. and Mnih A. Bayesian probabilistic matrix factorization using markov chain monte carlo. *Proceedings of the International Conference in Machine Learning*, pages 872–879, 2008.

[15] Salakhutdinov R. and Mnih A. Probabiltic matrix factorization. *Advances in Neural Information Processing Systems*, pages 1257–1264, 2008.

[16] Fazel M. Recht B. and Parrilo P. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. 2007.

[17] Ross S. *Introduction to Probability Models*. Academic Press, Berkeley, EEUU, 2007.

[18] Pestman W. *Mathematical Statistics*. Walter de Gruyter, Berlín, Alemania, 1998.