Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

INFERENCIA VARIACIONAL

Autor: Miguel Cubero Gómez
Tutor: Eustasio del Barrio Tellado
Año 2023

Índice

1. Introducción a la inferencia variacional

Uno de los problemas principales de la estadística moderna es aproximar densidades difíciles de calcular. Este problema es muy importante en la estadística bayesiana, que abarca toda la inferencia acerca de cantidades desconocidas y el cálculo de la distribución a posteriori.
¿Pero qué es la Inferencia Bayesiana y en qué se diferencia de la inferencia clásica? En el marco de la inferencia clásica bajo el modelo del muestreo simple se asume que se observan $X_1, ..., X_n$ variables independientes igualmente distribuidas y con densidad $f(x_i)$. Ahora consideramos $X = (X_1, ...X_n)$ con densidad conjunta $f(x_{1:n}) = f(x_1)...f(x_n)$. Para ahorrar escribimos $x_{1:n}$. Frecuentemente $f(x_i) = f(x_i|\theta)$ es un elemento de una familia paramétrica de distribuciones y en este caso la distribución de la muestra está dada por

$$f(x_i, ..., x_n|\theta) = \prod_{i=1}^{n} f(x_i|\theta)$$

El parámetro θ es desconocido, está fijo y pertenece al espacio de parámetros Θ . El productorio $\prod_{i=1}^{n} f(x_i|\theta)$ es la verosimilitud.
En la Inferencia Bayesiana $\theta \in \Theta$ es un vector aleatorio que tendrá una distribución $\pi(\theta)$ también conocida como distribución a priori. Esto nos permite considerar la densidad conjunta $\pi(\theta)f(x|\theta)$. En esta inferencia usamos los datos para modificar la distribución de los parámetros. En mucha literatura bayesiana se mezclan los conceptos de densidad y distribución. Definimos la distribución a posteriori como la distribución de los parámetros dados los datos. En el caso con densidades la distribución a posteriori está determinada por la densidad condicionada.

$$f(\theta|x) = \frac{f(x|\theta)\pi(\theta)}{f(x)} = \frac{f(x|\theta)\pi(\theta)}{\int_\Theta \pi(\theta)f(x|\theta)d\theta}$$

En dicha ecuación $\pi(\theta)$ es la densidad a priori, $f(x|\theta)$ la verosimilitud y $f(x)$ la densidad marginal. Hemos usado la expresión usual para calcular la densidad condicionada, conocida como regla de Bayes. En la anterior igualdad hemos sustituido $f(x)$ por la integral de la densidad conjunta con respecto al parámetro θ.
Más adelante va a ser conveniente reservar la notación θ para otros parámetros y por tanto, usaremos la notación habitual de la Inferencia Bayesiana:

$$p(z|x) = \frac{p(z,x)}{p(x)} = \frac{p(z)p(x|z)}{\int_Z p(x,z)dz} = \frac{p(z)p(x|z)}{\int_Z p(z)p(x|z)dz}$$

Aunque dicha ecuación es válida para quienes sean x y z, en nuestro caso x serán los datos observados y z serán las variables desconocidas o latentes que pertenecen a un conjunto $\mathcal{Z}$.
Antes de continuar insistimos en lo que significan los términos de la igualdad :

1. $p(z)$ se conoce como densidad a priori. Codifica cualquier creencia previa o conocimiento del dominio que tengamos.

2. $p(x|z)$ se conoce como la verosimilitud. Mide como es de posible obtener los datos observados dado el modelo.

3. $p(z|x)$ se conoce como densidad a posteriori. Refleja el hecho de que nuestro conocimiento de los parámetros se actualiza después de observar datos, es la densidad de z condicionada por los datos.

4. $p(x)$ es la densidad marginal de las observaciones. A menudos nos referiremos a este término como la evidencia.

En los modelos bayesianos, los parámetros van a jugar el papel de variables latentes. La inferencia en un modelo bayesiano equivale a condicionar por los datos y calcular la distribución a posteriori. En muchos modelos bayesianos complejos, la distribución a posteriori no se puede calcular y se hace necesario recurrir a aproximación. Un ejemplo de este problema lo veremos en la sección 1.1.
Durante décadas, la aproximación de la inferencia se resolvía mediante MCMC: métodos de Montecarlo basados en cadenas de Markov. En MCMC primero se construye una cadena de Markov ergódica en z cuya distribución estacionaria es la distribución a posteriori $p(z|x)$. A través de un proceso iterativo, se generan sucesivas muestras de la variable de interés, teniendo en cuenta la información de los datos observados. Cada paso de la iteración propone un nuevo valor para la variable y se decide su aceptación basándose en las distribuciones a posteriori y de transición. A medida que avanza la iteración, la cadena explora más exhaustivamente las regiones de alta probabilidad, permitiendo obtener una aproximación de la distribución a posteriori y realizar estimaciones estadísticas. Ver capítulo 7 en Bremaud (1999).
La inferencia variacional se enfrenta a este problema usando optimización. Primero, seleccionamos una familia de densidades aproximantes $\mathcal{Q}$ sobre las variables latentes. Luego, tratamos de encontrar el miembro de la familia que

minimice la divergencia de Kullback-Leibler(KL) a la distribución a posteriori:

$$q^* = \arg\min_{q \in \mathcal{Q}} KL(q \mid\mid p(\cdot|x)) \tag{1}$$

La divergencia de Kullback-Leibler es una medida de la separación de probabilidades. Como es un concepto importante en la inferencia variacional se le dedica la sección 2.
De este modo la inferencia variacional resuelve el problema de aproximar la posterior en un problema de optimización, cuya complejidad depende de la flexibilidad de la familia $\mathcal{Q}$. Una de las claves de la inferencia variacional es elegir correctamente la familia $\mathcal{Q}$ para encontrar una densidad cercana a $p(z|x)$ pero lo suficientemente sencilla para no complicar el proceso de optimización.
Diferencias y similitudes entre MCMC y la inferencia variacional. ¿Cuándo usar MCMC y cuándo los métodos variacionales? Los métodos MCMC son computacionalmente más costosos pero ofrecen mejores resultados que los métodos variacionales. Esto se debe a que asintóticamente con MCMC generamos muestras exactas a la distribución a posteriori mientras que usando inferencia variacional solo podremos acercarnos tanto como la familia $\mathcal{Q}$ nos lo permita. Por esto, la inferencia variacional encaja mejor con conjuntos de datos más grandes y escenarios donde queremos una respuesta más rápida. MCMC encaja mejor para conjuntos de datos más pequeños y escenarios en los que a pesar de pagar un precio computacional más alto necesitamos muestras exactas.

1.1. Ejemplo motivador: mezcla bayesiana de gaussianas

Vamos a completar esta introducción con un ejemplo motivador.
Consideramos una mezcla de gaussianas univariantes. Para generar una muestra de tamaño n, primero tomaremos K muestras provenientes de una distribución gaussiana $\mathbf{N}(0, \sigma^2)$, donde sigma es un hiperparámetro. Serán las medias que usaremos más adelante $\mu = (\mu_1, ..., \mu_K)$. Ahora, generaremos n vectores de dimensión K que nos dirán a que asignación latente pertenecen los datos x_i. Entonces, c_i será el vector i-ésimo y estará formado por todo ceros salvo un uno en la posición que nos dará una distribución de Bernouilli generalizada o multinomial.
Finalmente generaremos x_i de la correspondiente gaussiana dada por $\mathbf{N}(c_i^T \mu, 1)$.

El modelo resumido es :

$$\mu_k \sim \mathbf{N}(0, \sigma^2) \qquad k = 1, ..., K$$
$$c_i \sim multinomial(1/K, ..., 1/K) \qquad i = 1, ...n$$
$$x_i | c_i, \mu \sim \mathbf{N}(c_i^T \mu, 1) \qquad i = 1, ...n$$

Para una muestra de tamaño n, la densidad conjunta de las variables latentes $z = (\mu, c)$ y los datos es :

$$p(\mu, c, x) = p(\mu) \prod_{i=1}^{n} p(c_i) p(x_i | c_i, \mu) \tag{2}$$

También lo podemos escribir como

$$p(\mu_{1:K}, c_{1:n}, x_{1:n}) = \prod_{l=1}^{K} p(\mu_l) \prod_{i=1}^{n} p(c_i) p(x_i | c_i, \mu_{1:K})$$

Recordamos que $x_{1:n}$ es una forma abreviada de referirse a toda la muestra. Ahora si marginalizamos sobre las variables latentes tenemos que:

$$p(x_{1:n}) = \int \prod_{l=1}^{K} p(\mu_l) \prod_{i=1}^{n} \sum_{c_i} p(c_i) p(x_i | c_i, \mu_{1:K}) d\mu$$

Entonces la distribución a posteriori viene dada por :

$$p(\mu_{1:K}, c_{1:n} | x_{1:n}) = \frac{\prod_{l=1}^{K} p(\mu_l) \prod_{i=1}^{n} p(c_i) p(x_i | c_i, \mu_{1:K})}{\int \prod_{l=1}^{K} p(\mu_l) \prod_{i=1}^{n} \sum_{c_i} p(c_i) p(x_i | c_i, \mu_{1:K}) d\mu}$$

El numerador o la densidad conjunta, es fácil de calcular, el problema esta en el denominador.
El integrando no contiene factores separados para cada μ_k, cada μ_k aparece en los n factores del integrando. De este modo, la integral no se reduce a un producto de n integrales unidimensionales sobre los $\mu_k' s$.
Alternativamente, podemos mover el sumatorio fuera de la integral y escribir la densidad marginal como una suma de todas las posibles configuraciones de c:

$$p(x_{1:n}) = \sum_{c_i} p(c_i) \int \prod_{l=1}^{K} p(\mu_l) \prod_{i=1}^{n} p(x_i | c_i, \mu_{1:K}) d\mu$$

Ahora, podemos calcular cada miembro de la integral, pero hay K^n componentes, una para cada componente de los grupos dados por c. Por tanto, calcular la densidad marginal no es posible.

2. La divergencia Kullback-Leibler

Las divergencias son medidas que nos permiten cuantificar la diferencia o discrepancia entre dos distribuciones de probabilidad. Estas medidas son fundamentales para comparar distribuciones y realizar análisis estadísticos en una amplia variedad de campos.
En nuestro caso usaremos la divergencia de Kullback-Leibler, también conocida como entropía relativa. Se utiliza para medir la información que se pierde al aproximar una distribución por otra. Es decir, nos permite cuantificar como de lejos está una distribución de otra.

Para definir la divergencia de Kullback-Leibler necesitamos conocer el concepto de continuidad absoluta (ver Apéndice B).

Definición 1. Sean P y Q probabilidades definidas en el mismo espacio. Si $P \ll Q$ entonces:

$$D_{KL}(P,Q) = \int \log \frac{dP}{dQ} dP \tag{3}$$

Si P no es absolutamente continua respecto de Q entonces $D_{KL}(P,Q) = +\infty$.

Observación 1. Notemos que por la regla de la cadena, si P,Q son probabilidades en el mismo espacio y μ-σ finitas tal que $P \ll \mu$, $Q \ll \mu$ y $P \ll Q$ se tiene que:

$$D_{KL}(P,Q) = \int \log \frac{\frac{dP}{d\mu}}{\frac{dQ}{d\mu}} \frac{dP}{d\mu} d\mu \tag{4}$$

Siempre podemos encontrar dicha medida μ, bastaría considerar $\mu = \frac{P+Q}{2}$. Si $\mu(E) = 0$ entonces como P y Q son medidas positivas se tendría que $P(E) = 0$ y $Q(E) = 0$ luego $P \ll \mu$, $Q \ll \mu$.
Por otro lado, veamos que la expresión (4) no depende de la medida μ que consideremos. Sea ν tal que $P \ll \nu$, $Q \ll \nu$. Definimos $\rho = \frac{\mu+\nu}{2}$, luego $\mu \ll \rho$ y $\nu \ll \rho$. Además por la regla de la cadena tenemos que $\frac{dP}{d\rho} = \frac{dP}{d\mu}\frac{d\mu}{d\rho}$ y $\frac{dQ}{d\rho} = \frac{dQ}{d\mu}\frac{d\mu}{d\rho}$. Entonces:

$$\int \log \frac{dP}{dQ} dP = \int \log \frac{\frac{dP}{d\rho}}{\frac{dQ}{d\rho}} \frac{dP}{d\rho} d\rho = \int \log \frac{\frac{dP}{d\mu}\frac{d\mu}{d\rho}}{\frac{dQ}{d\mu}\frac{d\mu}{d\rho}} \frac{dP}{d\mu}\frac{d\mu}{d\rho} d\rho = \int \log \frac{\frac{dP}{d\mu}}{\frac{dQ}{d\mu}} \frac{dP}{d\mu} d\mu$$

Si P y Q son probabilidades en $\mathbb{R}^d$ con densidades p y q, respectivamente y consideramos la medida de Lebesgue:

$$D_{KL}(P,Q) = \int_{\mathbb{R}^d} \log \frac{p(x)}{q(x)} p(x)dx \tag{5}$$

Proposición 2. *Si P y Q son probabilidades en el mismo espacio entonces* $D_{KL}(P,Q) \geq 0$. *La igualdad se da si y solo si* $P = Q$.

Demostración. Sean p y q las densidades respectivas de P y Q.
Denotamos $R = \frac{P}{Q}$, con densidad r. Entonces la divergencia se puede escribir como

$$D_{KL}(P,Q) = \int_{\mathbb{R}^d} p(x) \log \frac{p(x)}{q(x)} d\mu(x) = \int_{\mathbb{R}^d} q(x)r(x) \log r(x) d\mu(x)$$

Notemos que si consideramos la función $t \log(t)$, es continua en $[0, \infty)$. Además, notemos que $t \log(t) \geq t - 1$. Por tanto:

$$D_{KL}(P,Q) = \int_{\mathbb{R}^d} q(x)r(x) \log r(x) d\mu(x) \geq \int_{\mathbb{R}^d} q(x)(r(x) - 1) d\mu(x) =$$

$$= \int_{\mathbb{R}^d} p(x) d\mu(x) - \int_{\mathbb{R}^d} q(x) d\mu(x) = 1 - 1 = 0$$

Para la segunda parte de la proposición tenemos que si lo anterior es una igualdad entonces $q(x)r(x) \log r(x) = q(x)(r(x) - 1)$ μ - para casi todo punto x. Como $q(x) > 0$ en un conjunto de probabilidad 1 tenemos que $r(x) \log r(x) = r(x) - 1$ y por tanto $r(x) = 1$. Es decir $p(x) = q(x)$ μ - para casi todo punto y la igualdad de las densidades implica la igualdad de las dsitribuciones. □

Acabamos de probar que $D_{KL}(P,Q)$ es positiva. En el caso discreto obtenemos como corolario la desigualdad de Gibbs:

$$-\sum_{i=1}^{n} p_i \log p_i \leq -\sum_{i=1}^{n} p_i \log q_i \tag{6}$$

Esta divergencia está muy relacionada con el estimador máximo verosimil. Para entender esta conexión asumimos que tenemos $X_1, ..., X_n$ una muestra aleatoria simple de la densidad f y que el modelo $\{f(x|\theta) : \theta \in \Theta\}$ es

correcto, es decir, que hay un $\theta_0 \in \Theta$ tal que $f = f(\cdot|\theta_0)$. Decimos en esta situción que θ_0 es el verdadero valor del parámetro. Escribimos P_θ para la probabilidad asociada a la densidad $f(\cdot|\theta)$ y asumimos que el modelo es identificable, es decir, que distintos valores de θ se corresponden con probabilidades P_θ diferentes. Entonces θ_0 es el minimizador de la divergencia de Kullback-Leibler:

$$D_{KL}(P_{\theta_0}, P_\theta), \qquad \theta \in \Theta.$$

El estimador máximo verosimil, $\hat{\theta}_n$, es el argumento que maximiza la función $l_n(\theta) = \frac{1}{n}\sum_{i=1}^n \log(X_i|\theta)$. Por tanto también es el maximizador de

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

Por la Ley de los Grandes Números este expresión converge a $-D_{KL}(P_{\theta_0}, P_\theta)$ para cada $\theta \in \Theta$. Intuitivamente, como las funciones K_n se van aproximando a menos la divergencia KL, los maximimizadores de K_n deberán acercarse al minimizador de D_{KL}. En algunos casos se puede probar que la convergencia es uniforme casi seguro:

$$\sup_{\theta\in\Theta} |K_n(\theta) + D_{KL}(P_{\theta_0}, P_\theta)| \to_{c.s.} 0$$

Probar esta convergencia requiere técnicas muy similares a probar el Teorema de Glivenko-Cantelli y esto escapa del objetivo del trabajo. Ver capítulo 4, sección 20 en Billingsley(1976).

Ejemplo
Gracias a la expresión anterior podemos calcular la divergencia KL entre distribuciones normales.
Si $P = N_d(\mu_1, \Sigma_1), Q = N_d(\mu_2, \Sigma_2)$, entonces:

$$D_{KL}(P, Q) = \frac{1}{2}\left[\log\frac{det(\Sigma_2)}{det(\Sigma_1)} - d + Tr(\Sigma_2^{-1}\Sigma_1) + (\mu_1 - \mu_2)'\Sigma_2^{-1}(\mu_1 - \mu_2)\right] \tag{7}$$

Para demostrar la igualdad anterior deberemos hacer uso de:

1. *Tr(AB)=Tr(BA).*
   Para que lo anterior tenga sentido A debe ser $n \times m$ y B $m \times n$.
   Ahora bien, para la traza tan solo debemos fijarnos en los productos

que nos generan los valores que aparecen en la diagonal, es decir, fila por columna cuando el número de la fila es el mismo que el número de la columna. De esta forma, tenemos que para una fila j fija (columna j por tanto) el elemento de la posición jj de la matriz AB es de la forma $\sum_{i=1}^{m} a_{ji}b_{ij}$. Luego su traza será $\sum_{j=1}^{n}\sum_{i=1}^{m} a_{ji}b_{ij}$ porque AB es $n \times n$. Análogamente tenemos que la traza de BA será $\sum_{j=1}^{m}\sum_{i=1}^{n} b_{ji}a_{ij}$ porque B es $m \times m$. Por tanto:

$$Tr(AB) = \sum_{j=1}^{n}\sum_{i=1}^{m} a_{ji}b_{ij} = \sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij}a_{ji} = \sum_{j=1}^{m}\sum_{i=1}^{n} b_{ji}a_{ij} = Tr(BA)$$

En la última igualdad hemos cambiado el subíndice i por j y viceversa.

2. Si $X \approx N_d(\mu_1, \Sigma_1)$, entonces

$$\begin{aligned} E(X-\mu_2)(X-\mu_2)' &= E(X-\mu_1+\mu_1-\mu_2)(X-\mu_1+\mu_1-\mu_2)' \\ &= E(X-\mu_1)(X-\mu_1)' + E(X-\mu_1)(\mu_1-\mu_2)' \\ &\quad + E(\mu_1-\mu_2)(X-\mu_1)' + E(\mu_1-\mu_2)(\mu_1-\mu_2)' \\ &= \Sigma_1 + (\mu_1-\mu_2)(\mu_1-\mu_2)'. \end{aligned}$$

Como $E(X) = \mu_1$ los términos del medio se anulan y el último elemento no depende de X.

3. Linealidad de la traza y la integral $Tr(E[x]) = E[Tr(x)]$.

4. La dimensión de $(\mu_1-\mu_2)\Sigma_2^{-1}(\mu_1-\mu_2)'$ es 1×1.

Entonces si $X \sim N_d(\mu_1, \Sigma_1)$:

$$\begin{aligned} D_{KL}(P,Q) &= \int_{\mathbb{R}^d} \log\left(\frac{\frac{1}{(2\pi)^{1/2}|\Sigma_1|^{1/2}}\exp(-\frac{1}{2}(x-\mu_1)')\Sigma_1^{-1}(x-\mu_1)}{\frac{1}{(2\pi)^{1/2}|\Sigma_2|^{1/2}}\exp(-\frac{1}{2}(x-\mu_2)')\Sigma_2^{-1}(x-\mu_2)}\right) \\ &\qquad \times \frac{1}{(2\pi)^{1/2}|\Sigma_1|^{1/2}}\exp(-\frac{1}{2}(x-\mu_1)')\Sigma_1^{-1}(x-\mu_1)dx \\ &= \log\frac{|\Sigma_2|^{1/2}}{|\Sigma_1|^{1/2}} + \frac{1}{2}\int_{\mathbb{R}^d}\left[(x-\mu_2)')\Sigma_2^{-1}(x-\mu_2) - (x-\mu_1)')\Sigma_1^{-1}(x-\mu_1)\right] \\ &\qquad \times \frac{1}{(2\pi)^{1/2}|\Sigma_1|^{1/2}}\exp(-\frac{1}{2}(x-\mu_1)')\Sigma_1^{-1}(x-\mu_1)dx \end{aligned}$$

$$= \frac{1}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|} + Tr(E\left[(X-\mu_2)(X-\mu_2)'\Sigma_2^{-1}\right] - Tr(E\left[(X-\mu_1)(X-\mu_1)'\Sigma_1^{-1}\right]\right)$$

$$= \frac{1}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|} + Tr(\Sigma_1 + (\mu_1-\mu_2)\Sigma_2^{-1}(\mu_1-\mu_2)') - Tr(\Sigma_1\Sigma_1^{-1})\right)$$

$$= \frac{1}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|} + Tr(\Sigma_1\Sigma_2^{-1}) + (\mu_1-\mu_2)'\Sigma_2^{-1}(\mu_1-\mu_2) - d\right)$$

Veamos como actúa la divergencia cuando d=1.
Supongamos que P$\sim \mathbf{N}(0,1)$ y Q$\sim \mathbf{N}(\mu,1)$ entonces por la fórmula (7) tenemos que

$$D_{KL}(P,Q) = \frac{1}{2}\left[\log 1 - 1 + 1 + \mu^2\right] = \frac{\mu^2}{2}$$

Evidentemente cuanto más cercano sea μ a 0 menor será la divergencia, y crece cuadráticamente.
Veamos que ocurre si ahora las varianzas son distintas. Si P$\sim \mathbf{N}(0,1)$ y Q$\sim \mathbf{N}(0,\sigma^2)$ entonces:

$$D_{KL}(P,Q) = \frac{1}{2}\left[2\log\sigma - 1 + \frac{1}{\sigma^2}\right]$$

Si consideramos la divergencia como una función de σ, nos damos cuenta de que alcanza su mínimo en 1 y crece cuadráticamente en cuanto nos alejamos de dicho valor.

2.1. La divergencia Kullback-Leibler y la inferencia variacional

En la sección 1, a la vista de (1) puede parecer sorprendente que podamos conocer la divergencia KL entre dos elementos de los cuales hay uno que no podemos calcular. En esta sección veremos como resolver este problema.
Como se ha comentado, el objetivo es resolver el problema (1), queremos encontrar el elemento q^* de una familia de densidades $\mathcal{Q}$ que minimice la divergencia KL entre dicho elemento y la densidad a posteriori.
Vamos a ver que la divergencia KL entre $q(z)$ y $p(z|x)$ no requiere conocer completamente $p(z|x)$. Va a depender de la densidad de un miembro de la familia $\mathcal{Q}$, de la densidad conjunta y de la densidad marginal de x, que es

una constante en nuestro problema.
Recordemos que podíamos expresar la divergencia KL como:

$$\begin{aligned} D_{KL}(q \mid\mid p(\cdot|x)) &= \int q(z) \log \frac{q(z)}{p(z|x)} dz \\ &= \int q(z) \log(q(z)) dz - \int q(z) \log p(z|x) dz \\ &= E_q \left[\log q(z)\right] - \int q(z) \log \frac{p(z,x)}{p(x)} dz \\ &= E_q \left[\log q(z)\right] - E_q \left[\log p(z,x)\right] + \log p(x) \int q(z) dz \\ &= E_q \left[\log q(z)\right] - E_q \left[\log p(z,x)\right] + \log p(x) \end{aligned}$$

Todas las esperanzas se toman con respecto a $q(z)$.
Entonces notemos que lo anterior se puede escribir como:

$$D_{KL}(q(z)||p(z|x)) + E_q \left[\log p(z,x)\right] - E_q \left[\log q(z)\right] = \log p(x)$$

Como hemos visto que la divergencia es no negativa tenemos que:

$$\log p(x) \geq E_q \left[\log p(z,x)\right] - E_q \left[\log q(z)\right]$$

Notemos que $E_q \left[\log p(z,x)\right] - E_q \left[\log q(z)\right]$ es una cota inferior para la evidencia. En inglés dicha cota se conoce como Evidence Lower Bound(ELBO). Como vamos a trabajar bastante con dicha cota a partir de ahora escribiremos:

$$ELBO := E_q \left[\log p(z,x)\right] - E_q \left[\log q(z)\right] \tag{8}$$

Entonces la divergencia se puede escribir como:

$$D_{KL}(q(z)||p(z|x)) = \log p(x) - ELBO$$

Para minimizar la divergencia, como no podemos calcular $\log p(x)$, podemos maximizar $E_q \left[\log p(z,x)\right] - E_q \left[\log q(z)\right]$. De esta forma podemos acercarnos a la densidad a posteriori sin conocerla mediante densidades que sí podemos calcular.

3. La familia variacional de campo medio

En la sección 2 hemos descrito la función a optimizar, ahora nos centraremos en el problema de minimización. Dicho problema puede ser muy complejo debido a la forma paramétrica de las densidades conjuntas. Con el objetivo de simplificarlo describiremos la familia variacional $\mathcal{Q}$ de campo medio. Su particularidad es que las densidades de las componentes latentes son independientes entre sí. Esta familia nos permitirá recurrir a la actualizacón por coordenadas en la optimización. Un miembro genérico de la familia se escribiría como:

$$q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j) \tag{9}$$

Cada variable latente z_j esta gobernada por su propio factor variacional, la densidad $q_j(z_j)$. Estos factores variacionales son elegidos para maximizar $E_q[\log p(z, x)] - E_q[\log q(z)]$ en la optimización.

Tenemos que enfatizar que la familia variacional de campo medio no es un modelo de los datos observados, notemos que no aparecen en la ecuación anterior (9). En su lugar, es el ELBO, y el problema de minimización de la divergencia KL los que conectan las densidades variacionales con los datos y el modelo.

Fijarse en estas familias, va a tener el beneficio de usar el método de actualización por coordenadas en la minimización. Lo desarrollaremos más profundamente en la sección 3.1.

De momento no hemos especificado la forma paramétrica de los individuos variacionales de la familia. En principio, cada uno puede tener cualquier forma apropiada dependiendo de la variable aleatoria correspondiente. Por ejemplo, una variable aleatoria continua puede tener un factor gaussiano; una variable multinomial normalmente tendrá un factor multinomial.

Hay que mencionar que la familia variacional de campo medio no es la única forma de resolver nuestro problema, hay familias mucho más complicadas pero hay que pagar un precio computacional.

Ejemplo

La familia variacional de campo medio para una familia de gaussianas contiene densidades de la forma:

$$q(\mu, c) = \prod_{k=1}^{K} q(\mu_k; m_k, s_k^2) \prod_{i=1}^{n} q(c_i; \varphi_i) \tag{10}$$

Siguiendo la fórmula de la familia de campo medio, cada media latente está gobernada por su propio factor variacional, el factor $q(\mu_k; m_k, s_k^2)$ es una densidad gaussiana respecto a la medida de Lebesgue, en la que μ_k representa el parámetro de media de la k-ésima componente gaussiana; y m_k y s_k^2 son los parámetros de la densidad variacional que aproximará la densidad posterior de μ_k.
El factor $q(c_i; \varphi_i)$ es la función de masa de probabilidad de las asignaciones, es una densidad variacional, donde c_i representa la asignación de clase del i-ésimo dato y φ_i son las probabilidades de las clases. En el contexto de la mezcla bayesiana de gaussianas, las asignaciones de clase c_i indican a que componente gaussiana pertenece cada dato i, y la densidad variacional $q(c_i; \varphi_i)$ se utiliza para aproximar la incertidumbre en estas asignaciones de clase.

3.1. Inferencia variacional con una familia de campo medio usando ascenso coordenado

Usando la cota inferior y la familia variacional de campo medio hemos convertido el problema de la inferencia variacional en un problema de optimización. El análisis numérico nos aporta técnicas para resolverlo. En el campo del aprendizaje automático destacan sobre todo los métodos basados en ascenso de gradiente y los algoritmos de ascenso coordenado. A continuación hablaremos de estos métodos y cúal es más adecuado en nuestro caso.
El objetivo es minimizar la función objetivo $f : \mathbb{R}^d \to \mathbb{R}$. Y suponemos que es suficientemente diferenciable. Considerando su desarrollo de Taylor de primer orden tenemos que

$$f(y) = f(x) + \nabla f(x)(y - x) + o(||x - y||) \tag{11}$$

Entonces para buscar la dirección que nos interesa tomamos $u \in \mathbb{R}^d$ tal que $||u|| = 1$ y $h > 0$, por tanto la expresión (11) se escribiría como:

$$f(x + hu) = f(x) + h\ \nabla f(x) \cdot u + o(||h||)$$

La dirección u que satisface $\nabla f(x) \cdot u < 0$ se denomina dirección de descenso porque así $f(x + hu) < f(x)$. Dejando fijo h, se sigue que cuanto menor sea $\nabla f(x) \cdot u$, más rápido se minimiza la función. Además como $||u|| = 1$ se tiene que:

$$|\nabla f(x) \cdot u| \leq ||\nabla f(x)||\ ||u|| = ||\nabla f(x)||$$

Entonces

$$-||\nabla f(x)|| \leq \nabla f(x) \cdot u \leq ||\nabla f(x)||$$

Por tanto

$$\min_{||u||=1} \nabla f(x) \cdot u = -||\nabla f(x)||$$

Y alcanzaremos ese valor cuando $u = -\frac{\nabla f(x)}{||\nabla f(x)||}$ si $\nabla f(x) \neq 0$.
Notemos que en este caso

$$\nabla f(x) \cdot u = -\frac{\nabla f(x)^2}{||\nabla f(x)||} < 0$$

Por tanto dado el iterante x_n:

$$x_{n+1} = x_n - \gamma_n \nabla f(x_n), \tag{12}$$

donde γ_n es la tasa de aprendizaje. La ecuación (12) es la llamada iteración de descenso de gradiente. Los métodos de descenso de gradiente se pueden mejorar incluyendo la hessiana en el desarrollo de Taylor, pero el coste computacional de calcular y almacenar la Hessiana crece más deprisa que el coste del gradiente. Este es el algoritmo más usado pero al elegir una familia de campo medio hay una alternativa más eficiente.

Escoger una familia variacional de campo medio nos permite recurrir a la actualización por coordenadas o ascenso coordenado, un método menos costoso que el descenso de gradiente que hemos explicado anteriormente. Este método actualiza las variables una a la vez, en lugar de actualizar todas las variables simultáneamente como en el método del descenso de gradiente. En cada iteración, se selecciona una variable y se actualiza su valor, manteniendo constantes los valores de las demás variables. Dicha actualización se realiza utilizando una fórmula específica que depende de la función objetivo.
En esta sección, describiremos el uso de la actualización por coordenadas en la inferencia variacional. La actualización por coordenadas optimiza iteradamente cada factor de la familia variacional de campo medio, mientras mantiene los otros fijos. Este algoritmo hace converger la cota inferior (8) a un máximo local.
Como se ha comentado queremos maximizar la expresión (8) ya que así se minimiza la divergencia KL.
A continuación desarrollamos como se implementa el paso por coordenadas.

Supongamos que tenemos $z_{1:m}$ variables latentes y $x_{1:n}$ datos observados, entonces por la regla de la cadena :

$$p(z_{1:m}, x_{1:n}) = p(x_{1:n}) \prod_{j=1}^{m} p(z_j | z_{1:(j-1)}, x_{1:n})$$

Al tomar logaritmos tenemos que :

$$\log p(z_{1:m}, x_{1:n}) = \log p(x_{1:n}) + \sum_{j=1}^{m} \log p(z_j | z_{1:(j-1)}, x_{1:n}) \tag{13}$$

Por otro lado, como estamos en una familia de campo medio:

$$q(z_{1:m}) = q_1(z_1) q_2(z_2) ... q_m(z_m)$$

$$\log q(z_{1:m}) = \log q_1(z_1) q_2(z_2) ... q_m(z_m) = \sum_{j=1}^{m} \log q_j(z_j)$$

Por tanto, al tomar esperanzas, el segundo término de (8) se puede escribir como:

$$\begin{aligned} E\left[\log q(z_{1:m})\right] &= \int \left(\sum_{j=1}^{m} \log q_j(z_j) \prod_{j=1}^{m} q_j(z_j) \right) dz_{1:m} \\ &= \sum_{j=1}^{m} \int \log q_j(z_j) q_j(z_j) dz_j \\ &= \sum_{j=1}^{m} E_j\left[\log q_j(z_j)\right] \end{aligned}$$

E_j denota la esperanza con respecto a la densidad variacional q_j.
Tomando esperanzas en (13) obtenemos el primer término del ELBO y por la linealidad de la esperanza tenemos que:

$$ELBO = \log p(x_{1:n}) + \sum_{j=1}^{m} E\left[\log p(z_j | z_{1:(j-1)}, x_{1:n})\right] - E_j\left[\log q_j(z_j)\right]$$

A partir de ahora consideraremos z_k como la última variable en la lista y en vez de $p(z_k | z_{1:(k-1)})$ escribiremos por ahorrar en notación $p(z_k | z_{-k})$, es decir,

la densidad condicionada p de z_k dadas todas las demás variables latentes. Entonces para cada $q_k(z_k)$ la función objetivo a maximizar es:

$$\mathcal{F} = E\left[\log p(z_k|z_{-k}, x_{1:n})\right] - E_k\left[\log q_k(z_k)\right] + const.$$

Operando con la esperanza del primer término:

$$E\left[\log p(z_k|z_{-k}, x_{1:n})\right] = \int q(z_{1:m}) \log p(z_k|z_{-k}, x_{1:n}) dz_{1:m}$$

$$= \int q_1(z_1)...q_m(z_m) \log p(z_k|z_{-k}, x_{1:n}) dz_1...dz_m$$

$$= \int q_k(z_k)q_1(z_1)...q_m(z_m) \log p(z_k|z_{-k}, x_{1:n}) dz_k dz_1...dz_m$$

$$= \int q_k(z_k) E_{-k}\left[\log p(z_k|z_{-k}, x_{1:n})\right] dz_k$$

Sustituyendo lo anterior en la función objetivo obtenemos:

$$\mathcal{F} = \int q_k(z_k) E_{-k}\left[\log p(z_k|z_{-k}, x_{1:n})\right] dz_k - \int q_k(z_k) \log q_k(z_k) dz_k + const.$$

Denotamos f_k a $E_{-k}\left[\log p(z_k|z_{-k}, x_{1:n})\right]$, que no depende de la densidad q_k

$$\mathcal{F} = \int q_k(z_k) f_k(z_k) dz_k - \int q_k(z_k) \log q_k(z_k) dz_k$$

$$= \int q_k(z_k) f_k(z_k) - q_k(z_k) \log q_k(z_k) dz_k$$

$$= \int q_k(z_k) \left(f_k(z_k) - \log q_k(z_k)\right) dz_k$$

Entonces son equivalentes:

$$\arg\max_q \int q_k(z_k) \left(f_k(z_k) - \log q_k(z_k)\right) dz_k$$

$$= \arg\min_q \int q_k(z_k) \left(\log q(z_k) - f_k(z_k)\right) dz_k$$

Sujeto a que $\int q_k(z_k) dz_k = 1$. Sea $B(z_k) = \exp(f_k(z_k))$ entonces $f_k(z_k) = \log B(z_k)$. Consideramos ahora $\tilde{B}(z_k) = B(z_k)/\int B(z_k)$, denotamos $c = \int B(z_k) dz_k$ una constante. Luego $\log B(z_k) = \log c + \log \tilde{B}(z_k)$.

Entonces

$$\int q_k(z_k)\left(\log q(z_k) - f_k(z_k)\right) dz_k =$$

$$= \int q_k(z_k)\left(\log q_k(z_k) - \log c - \log \tilde{B}(z_k)\right) dz_k$$

$$= \int q_k(z_k)\left(\log \frac{q_k(z_k)}{\tilde{B}(z_k)}\right) dz_k - \log c$$

$$= D_{KL}(q_k, \tilde{B}) - \log c$$

Queremos encontrar el mínimo de la siguiente expresión

$$\arg\min_q \int q_k(z_k)\left(\log q(z_k) - f_k(z_k)\right) dz_k = \arg\min_q D_{KL}(q_k, \tilde{B}) - \log c$$

Por lo visto en la sección 2, sabemos que la divergencia KL es no negativa y alcanza el valor 0 cuando ambas densidades son iguales. Por tanto

$$\arg\min_q \int q_k(z_k)\left(\log q(z_k) - f_k(z_k)\right) dz_k = \tilde{B} \approx \exp(f_k)$$
$$\approx \exp(E_{-k}\left[\log p(z_k|z_{-k}, x_{1:n})\right]) \tag{14}$$

La esperanza $E_{-k}\left[\log p(z_k|z_{-k}, x_{1:n})\right]$ se toma con respecto a la densidad variacional dada por z_{-k}, $\prod_{j\neq k} q_j(z_j)$, que son valores fijos. Por otro lado, notemos que el valor óptimo es proporcional a la exponencial del logaritmo conjunto:

$$q_k^*(z_k) \approx \exp(E_{-k}\left[\log p(z_k, z_{-k}, x_{1:n})\right]) \tag{15}$$

Esto se debe a que :

$$\log p(z_j|z_{-j}, x) = \log \frac{p(z_j, z_{-j}, x)}{p(z_{-j}, x)} = \log p(z_j, z_{-j}, x) - \log p(z_{-j}, x)$$

Por tanto tenemos que:

$$\exp\left(E_{-j}\left[\log p(z_j|z_{-j}, x)\right]\right) = \exp\left(E_{-j}\left[\log p(z_j, z_{-j}, x) - \log p(z_{-j}, x)\right]\right)$$
$$= \exp\left(E_{-j}\left[\log p(z_j, z_{-j}, x)\right] - E_{-j}\left[\log p(z_{-j}, x)\right]\right) \tag{16}$$

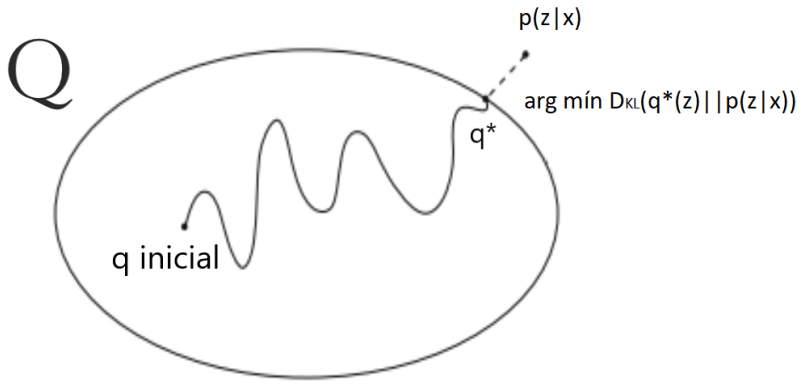
Como $E_{-j}\left[\log p(z_{-j}, x)\right]$ será una constante podemos concluir (15).
Ahora bien, ¿cómo podemos saber si $q_k^*(z_k)$ sigue estando en la familia $\mathcal{Q}$?

Esto dependerá de la elección de dicha familia: si suponemos que es totalmente libre, entonces $q_k^*(z_k)$ volverá a estar en $\mathcal{Q}$ ya que contiene a todas las densidades que necesitemos. En los demás casos tendremos que asegurarnos de esto previamente. En la siguiente sección veremos que en la familia de exponenciales la actualización vuelve a ser una exponencial. Esto nos será útil en el ejemplo de la mezcla de gaussianas.
Estas son las ecuaciones que están detrás del algoritmo de ascenso por coordenadas. Dicho algoritmo maximiza la cota inferior para minimizar la divergencia KL.

Algoritmo 1: Ascenso Coordenado en Inferencia Variacional

Input: un modelo $p(\mathbf{x},\mathbf{z})$, un conjunto de datos $\mathbf{x}$
Output: Una densidad variacional $q(\mathbf{z}) = \prod_{j=1}^{m} q_j(z_j)$
Inicialización: q(z) inicial con sus factores variacionales $q_j(z_j)$
Mientras no converja el ELBO **hacer**:
$\longrightarrow$ **para** j $\in$ 1,...,m **hacer**:
$\longrightarrow$ Establecer $q_j(z_j) \approx \exp(E_{-k}\left[\log(p(z_k|z_{-k}, x_{1:n}))\right])$
$\longrightarrow$ **fin**
$\longrightarrow$ Calcular el ELBO=$E\left[\log p(\mathbf{x},\mathbf{z})\right] - E\left[\log q(\mathbf{z})\right]$
fin
Devolver $q(\mathbf{z})$

El siguiente dibujo nos indica como funciona nuestro algoritmo:



Empezamos con un miembro q de la familia de densidades $\mathcal{Q}$ y vamos iterando hasta conseguir el miembro de la familia que minimice la divergencia de Kullback-Leibler.

4. Inferencia variacional con familias exponenciales

Hemos descrito la inferencia variacional de campo medio y explicado el algoritmo de ascenso por coordenadas, un algoritmo de ascenso coordenado para optimizar la cota inferior (8). Posteriormente, en la sección 5, veremos como ejemplo una mezcla de gaussianas, donde cada actualización será nuevamente una gaussiana y cada factor multinomial será también un factor multinomial, como se explicará en este capítulo. La mezcla de gaussianas es un miembro de la importante clase de modelos donde el condicional completo está en la familia exponencial.

Trabajar en la familia de exponenciales simplifica la inferencia variacional. Es más sencillo de computar el algoritmo y permite escalar la inferencia variacional a grandes cantidades de datos.

La familia exponencial usual tiene la siguiente forma:

$$f(x|\theta) = h(x)\exp(\eta(\theta)T(x) - A(\eta(\theta)))$$

Donde T(x) es un estimador suficiente de la distribución, η es el llamado parámetro natural y tanto $A(\eta(\theta))$ como y $h(x)$ son factores de normalización de la distribución.

Ahora consideraremos un modelo genérico $p(z, x)$ en el que cada condicioanl completo está en la familia exponencial:

$$p(z_j|z_{-j}, x) = h(z_j)\exp(\eta_j(z_j, x)^T z_j - a(\eta_j(z_{-j}, x))) \tag{17}$$

Como es una densidad condicional, $\eta_j(z_{-j}, x)$ es una función de las variables que condicionamos.

Considerando la inferencia variacional de campo medio para esta familia: $q(z) = \prod_{j=1}^{n} q_j(z_j)$ y usando la expresión (14):

$$\begin{aligned} q_j(z_j) &\approx \exp(E_{-j}\left[\log(p(z_k|z_{-k}, x_{1:n}))\right]) \\ &\approx \exp\{E_{-j}\left[\log h(z_j) + \log(\exp(\eta_j(z_j, x)^T z_j - a(\eta_j(z_{-j}, x)))\right]\} \\ &\approx \exp\{\log h(z_j) + E_{-j}\left[\eta_j(z_j, x)^T\right] z_j - E_{-j}\left[a(\eta_j(z_{-j}, x))\right]\} \\ &\approx h(z_j)\exp\{E_{-j}\left[\eta_j(z_j, x)^T\right] z_j\} \end{aligned} \tag{18}$$

Esta es la forma de los factores variacionales que actualizaremos.
Cada uno de dichos factores está en la misma familia exponencial que el condicional completo que vimos en (17). Sus parámetros tienen la misma dimensión y las mismas funciones h y a. Tanto las normales como los factores multinomiales forman parte de la familia exponencial, siendo estos últimos un caso sencillo de dicha familia.
Sea ν_j el parámetro variacional para el j-ésimo factor. Cuando actualizamos cada factor, ajustamos su parámetro como la esperanza del condicional completo:

$$\nu_j = E_{-j}\left[\eta_j(z_j, x)^T\right]$$

Esta expresión facilita muchos algoritmos de ascenso coordenado para modelos complicados.

5. Aplicaciones

Volvemos al ejemplo de mezcla de gaussianas vista en la sección 3.
En dicha sección asumíamos que teníamos K componentes y n datos observados $x_{1:n}$. Las variables latentes son K vectores con valores reales que servirán como parámetros de las medias: $\mu = \mu_{1:K}$ y n asignaciones latentes $\mathbf{c} = c_{1:n}$. La asignación c_i nos indica de qué componente proviene el dato x_i. Siendo más precisos, c_i es un vector con K componentes, de las cuales todas son cero excepto un uno en la componente correspondiente a x_i. Tenemos un hiperparámetro σ^2, que es la varianza de la normal a priori de la que provendrán las medias. Asumimos que la varianza de las observaciones es 1 para tener una densidad a priori uniforme sobre la mezcla de las componentes. Recordemos que tenemos dos tipos de parámetros variacionales: los parámetros multinomiales φ_i, para aproximar la asignación del i-ésimo dato a su factor correspondiente de la distribución a posteriori y los parámetros gaussianos m_k y s_k^2, para aproximar la distribución a posteriori de la k-ésima componente de la mezcla.
Queremos maximizar la cota inferior (8), que depende del logaritmo de la densidad conjunta y del logaritmo de la densidad variacional conjunta latente. La densidad conjunta de los datos y de las variables latentes está escrita en la ecuación (2) y la familia variacional en la ecuación (10). Tomando logaritmos

en (2) tenemos que

$$\log p(\mu, c, x) = \sum_{k=1}^{K} \log p(\mu_k) + \sum_{i=1}^{n} \log p(c_i) + \sum_{i=1}^{n} \log p(x_i|c_i, \mu) \tag{19}$$

Por otro lado como estamos usando la familia de campo medio

$$q(\mu_{1:k}, c_{1:n}) = \prod_{k=1}^{K} q(\mu_k) \prod_{i=1}^{n} q(c_i) \tag{20}$$

Combinando (19) y (20) podemos calcular el ELBO:

$$\begin{aligned} ELBO &= E\left[\log\left(\prod_{k=1}^{K} p(\mu_k) \prod_{i=1}^{n} p(c_i) p(x_i|c_i, \mu_{1:K}\right)\right] \\ &\quad - E\left[\log\left(\prod_{k=1}^{K} q(\mu_k) \prod_{k=1}^{n} q(c_i)\right)\right] \\ &= \sum_{k=1}^{K} E\left[\log p(\mu_k)\right] + \sum_{i=1}^{n} \left(E\left[\log p(c_i)\right] + E\left[\log p(x_i|c_i, \mu_{1:K})\right]\right) \\ &\quad - \sum_{i=1}^{n} E\left[\log q(c_i)\right] - \sum_{k=1}^{K} E\left[\log q(\mu_k)\right] \end{aligned} \tag{21}$$

El algoritmo de ascenso por coordenadas actualiza cada parámetro variacional siguiendo la fórmula (15) que vimos en la sección 3.1.
Ahora vamos a considerar (21) como una función de los factores de las asignaciones y busqcaremos su máximo, posteriormente haremos lo mismo para los factores variacionales de las componentes.

La densidad variacional de las asignaciones de la mezcla

Usando la fórmula (15) y quedándonos solo con los términos que dependen de c_i en (19):

$$\log q(c_i) \approx E_{-i}\left[\log p(\mu, c, x)\right] \approx \log p(c_i) + E_{-i}\left[\log p(x_i|c_i, \mu)\right] \tag{22}$$

Por la definición de c_i tenemos que $p(c_i) = \frac{1}{K}$ luego $\log p(c_i) = -\log K$.

Como c_i es uno de los vectores indicadores de longitud K: $(1, 0, ..., 0)$, $(0, 1, ..., 0), ..., (0, ..., 0, 1)$, podemos escribir

$$p(x_i|c_i, \mu) = \prod_{k=1}^{K} p(x_i|\mu_k)^{c_{ik}} \tag{23}$$

Donde ik representa la k-ésima coordenada del vector c_i. De este modo, los términos que no afectan a la asignación c_i estarán elevados a 0 y por tanto su valor será 1 y nos quedará el término que queremos.
Usando esto estudiaremos lo que ocurre con el término $E_{-i}[\log p(x_i|c_i, \mu)]$:

$$\begin{aligned} E_{-i}[\log p(x_i|c_i, \mu)] &= \sum_{k=1}^{K} c_{ik} E_{-i} \left[\log p(x_i|\mu_k)\right] \\ &= \sum_{k=1}^{K} c_{ik} E_{-i} \left[-(x_i - \mu_k)^2/2\right] + const. \\ &= \sum_{k=1}^{K} c_{ik} \left(E\left[\mu_k\right] x_i - E\left[\mu_k^2\right]/2\right) + const. \end{aligned} \tag{24}$$

Las esperanzas anteriores son sobre las componentes μ de la mezcla de gaussianas. En cada línea de lo anterior hemos quitado los términos que no dependen de c_i. Como c_{ik} será 1 o 0, si volvemos a (22) llegamos a que

$$\log q(c_i) \approx -\log K + E\left[\mu_k\right] x_i - E\left[\mu_k^2\right]/2$$

Como $\log K$ es una constante tenemos que

$$q^*(c_i) \approx \exp\left\{E\left[\mu_k\right] x_i - E\left[\mu_k^2\right]/2\right\} \tag{25}$$

Denotamos por φ_{ik} a la probabilidad de la i-ésima asignación de pertenecer al grupo k, entonces sabemos que $\sum_{k=1}^{K} \varphi_{ik} = 1$, por tanto tenemos que normalizar la expresión de $q^*(c_i)$ para obtener φ_{ik}:

$$\varphi_{ik} = \frac{\exp\left\{E\left[\mu_k\right] x_i - E\left[\mu_k^2\right]/2\right\}}{\sum_{k=1}^{K} \exp\left\{E\left[\mu_k\right] x_i - E\left[\mu_k^2\right]/2\right\}} \tag{26}$$

Este cálculo requiere conocer E$[\mu_k]$ y E$[\mu_k^2]$ para cada componente de la mezcla, ahora veremos que podemos calcular las dos porque conocemos la gaussiana variacional de la k-ésima componente.
Otro dato de importancia es que φ_{ik} solo depende de los parámetros variacionales de la mezcla de las componentes de las medias y de los datos observados.

La densidad variacional de las medias de las componentes de la mezcla

Ahora queremos optimizar la densidad variacional $q(\mu_k)$.
Aplicando logaritmos en (23) tenemos que

$$\log p(x_i|c_i, \mu_{1:K}) = \sum_{k=1}^{K} c_{ik} \log p(x_i|\mu_k)$$

Además como c_{ik} es independiente de $\log p(x_i|\mu_k)$ tenemos que

$$E_{-k}[c_{ik} \log p(x_i|\mu_k)] = E_{-k}[c_{ik}]E_{-k}[\log p(x_i|\mu_k)] \tag{27}$$

Recordemos que c_{ik} es un vector indicador y por tanto su esperanza es la probabilidad de que la i-ésima observación provenga de la k-ésima componente, es decir, φ_{ik}.
Igual que en el caso de las asignaciones, usamos la fórmula (15) y nos quedamos solo con los término que dependen de μ_k en (19):

$$\log q(\mu_k) \approx E_{-k}[\log p(\mu, c, x)] \approx \log p(\mu_k) + \sum_{i=1}^{n} E_{-k}[\log p(x_i|c_i, \mu)]$$

Habíamos asumido que p era una densidad gaussiana, juntándolo con (27) tenemos que:

$$\begin{aligned}
\log q(\mu_k) &\approx \log p(\mu_k) + \sum_{i=1}^{n} E_{-k}[\sum_{l=1}^{K} c_{il} \log p(x_i|\mu_l)] \\
&\approx \log p(\mu_k) + \sum_{i=1}^{n} E_{-k}[c_{ik}]E_{-k}[\log p(x_i|\mu_k)] \\
&\approx \log p(\mu_k) + \sum_{i=1}^{n} \varphi_{ik} \log p(x_i|\mu_k) \\
&\approx -\frac{\mu_k^2}{2\sigma^2} - \sum_{i=1}^{n} \varphi_{ik} \frac{1}{2}(x_i - \mu_k)^2 + const \\
&\approx -\frac{\mu_k^2}{2\sigma^2} + \sum_{i=1}^{n} \varphi_{ik} x_i \mu_k - \frac{1}{2} \sum_{i=1}^{n} \varphi_{ik} \mu_k^2 + const \\
&\approx \left(\sum_{i=1}^{n} \varphi_{ik} x_i \right) \mu_k - \frac{1}{2} \left(\frac{1}{\sigma^2} + \sum_{i=1}^{n} \varphi_{ik} \right) \mu_k^2 + const
\end{aligned}$$

Por tanto, si tomamos exponenciales deducimos que $q(\mu_k) \sim \mathbf{N}(m_k, s_k^2)$ donde

$$m_k = \frac{\sum_{i=1}^n \varphi_{ik} x_i}{\frac{1}{\sigma^2} + \sum_{i=1}^n \varphi_{ik}}, \qquad s_k^2 = \left(\frac{1}{\sigma^2} + \sum_{i=1}^n \varphi_{ik} \right)^{-1}$$

Esto se debe a que si:

$$y = \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(x-\mu)^2}{2\sigma^2}, \qquad \log y \sim \frac{x}{\sigma^2}\mu - \frac{1}{2}\frac{\mu^2}{\sigma^2}$$

Estas actualizaciones están muy relacionadas con la densidad condicionada de la k-ésima componente de la mezcla. Dicha densidad es una gaussiana dada los datos asignados a la k- ésima componente. La actualización variacional es una condicional completa ponderada, donde cada punto de datos se pondera por su probabilidad variacional de ser asignado a la componente.
Notemos que $q(\mu_k)$ depende de los datos y de φ_{ik} y por el contrario $q(c_i)$ depende de los datos y de m_k y s_k^2 y por tanto tenemos una dependencia circular.
Calculemos cada uno de los términos de (21):

1.

$$\begin{aligned} E\left[\log p(\mu_k)\right] &= E\left[-\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\mu_k^2 \right] \\ &= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2} E\left[\mu_k^2\right] \\ &\approx -\frac{1}{2\sigma^2}(s_k^2 + m_k^2) \end{aligned}$$

2.

$$E\left[\log p(c_i)\right] = E\left[\sum_{k=1}^K c_{ik} \log \frac{1}{K} \right] = -\log K$$

3.

$$\begin{aligned} E\left[\log p(x_i|c_i,\mu)\right] &= \sum_{k=1}^K \varphi_{ik} \left(E\left[\mu_k\right] x_i - E\left[\mu_k^2\right]/2 \right) + const \\ &= \sum_{k=1}^K \varphi_{ik} m_k x_i - \frac{1}{2}\varphi_{ik}(m_k^2 + s_k^2) + const \end{aligned}$$

4.

$$\begin{aligned} E\left[\log q(\mu_k)\right] &= E\left[-\frac{1}{2}\log(2\pi s_k^2) - \frac{1}{2_k^2}(\mu_k - m_k)^2\right] \\ &= -\frac{1}{2}\log(2\pi s_k^2) - \frac{1}{2s_k^2}(E[\mu_k^2] - 2m_k E[\mu_k] + m_k^2) \\ &= -\frac{1}{2}\log(2\pi s_k^2) - \frac{1}{2s_k^2}(s_k^2 + m_k^2 - 2m_k^2 + m_k^2) \\ &= -\frac{1}{2}\log(2\pi s_k^2) - \frac{1}{2s_k^2}s_k^2 = -\frac{1}{2}\log(2\pi s_k^2) - \frac{1}{2} \end{aligned}$$

5.

$$E\left[\log q(c_i)\right] = E\left[\sum_{k=1}^{K} c_{ik}\log\varphi_{ik}\right] = \sum_{k=1}^{K}\varphi_{ik}\log\varphi_{ik}$$

Escribiremos el algoritmo de ascenso coordenado para la mezcla de gaussianas

Algoritmo 2: Ascenso coordenado en la mezcla de gaussianas

Input: Un conjunto de datos **x**, K número de componentes de la mezcla, la varianza a priori de las medias de las componentes σ^2
Output: Las densidades variacionales $q(\mu_k)$(gaussiana) y $q(c_i)(Multinomial)$
Inicialización: Los factores variacionales $m = m_{1:K}$, $s^2 = s^2_{1:K}$ y $\varphi = \varphi_{1:n}$
Mientras no converja el ELBO **hacer**:
$\longrightarrow$ **para** i $\in$ 1,...,n **hacer**:
$\longrightarrow$ Establecer $\varphi_{ik} \approx \exp(m_k * x_i - 0{,}5(m_k^2 + s_k^2))$
$\longrightarrow$ **fin**
$\longrightarrow$ **para** k $\in$ 1,...,K **hacer**:
$\longrightarrow$ Establecer $m_k \approx \frac{\sum_{i=1}^{n}\varphi_{ik}x_i}{\frac{1}{\sigma^2}+\sum_{i=1}^{n}\varphi_{ik}}$
$\longrightarrow$ Establecer $s_k^2 \approx \left(\frac{1}{\sigma^2} + \sum_{i=1}^{n}\varphi_{ik}\right)^{-1}$
$\longrightarrow$ **fin**
$\longrightarrow$ Calcular el ELBO=$E\left[\log p(\mathbf{x},\mathbf{z})\right] - E\left[\log q(\mathbf{z})\right]$
fin
Devolver $q(m, s^2, \varphi)$

Una vez tenemos la densidad variacional calculada, podemos usarla como aproximación a la densidad a posteriori.

Por ejemplo, podemos usar la densidad variacional encontrada para aproximar la densidad de los nuevos datos. Asumimos que estos datos siguen una distribución gaussiana con la media de cada componente de la mezcla, es decir, $p(x_{nuevo}|x_{1:n})$. Entonces:

$$p(x_{nuevo}|x_{1:n}) \approx \frac{1}{K}\sum_{k=1}^{K} p(x_{nuevo}|m_k)$$

Esto significa que aproximamos la densidad de los nuevos datos sumando las probabilidades de observar el nuevo dato x_{nuevo} en cada uno de los componentes de la mezcla, ponderadas por $\frac{1}{K}$, donde K es el número de componentes de la mezcla.
Otra opción es obtener una descomposición de los nuevos datos. Asignamos los puntos a su asignación de mezcla más probable $c_i = \arg\max_k \varphi_{ik}$ (porque φ_{ik} es la probabilidad de asignar el punto i al grupo k) y estimamos las medias de los grupos con sus medias variacionales m_k.

5.1. Ejemplo 1. Mezcla de gaussianas en una dimensión.

El siguiente ejemplo nos mostrará como funciona el algoritmo de ascenso coordenado en una mezcla de gaussianas en una dimensión.
Comenzaremos generando los datos a partir del hiperparámetro sigma cuyo valor será 3, tendremos 5 grupos y 1000 observaciones.
El código para este ejemplo se encuentra al final como apéndice (Ver A.1 Código Ejemplo1) y dicho código nos aporta los siguientes gráficos:
El primero son los datos generados a partir de las medias aleatorias mu y de varianza 1. Los distintos colores nos indican el grupo del que provienen.

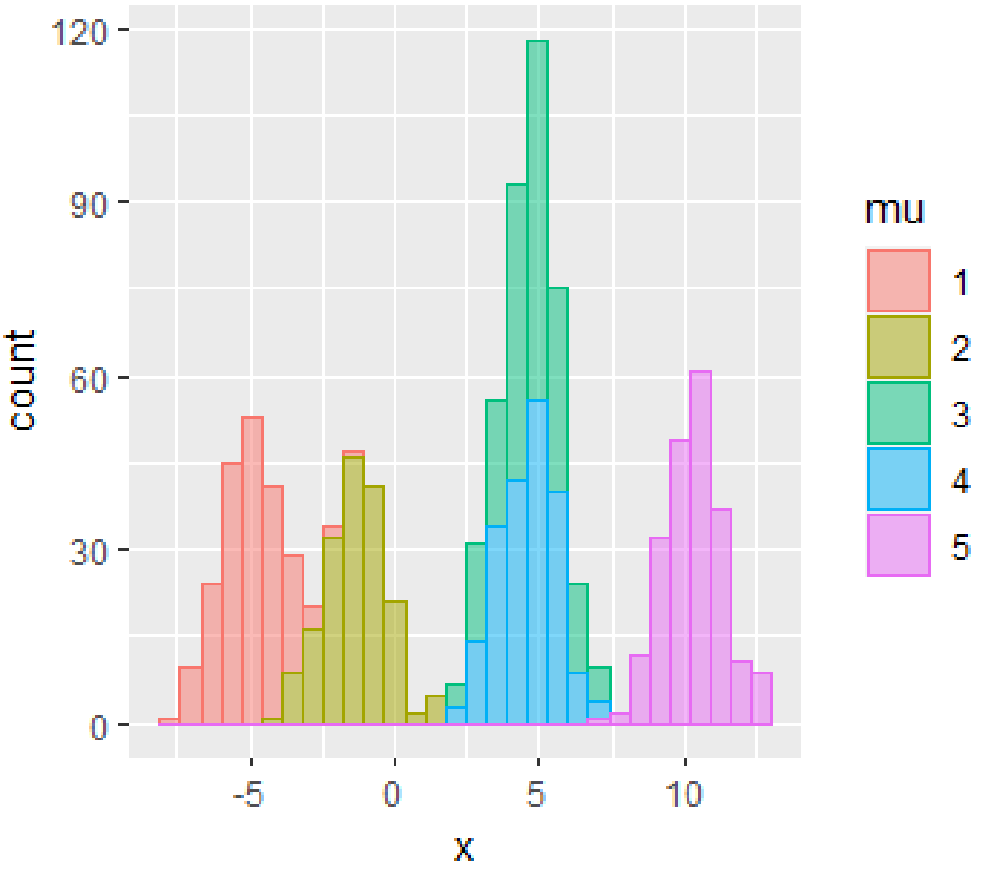


Figura 1

El siguiente gráfico nos mostrará donde se encuentran las medias calculadas con el algoritmo:

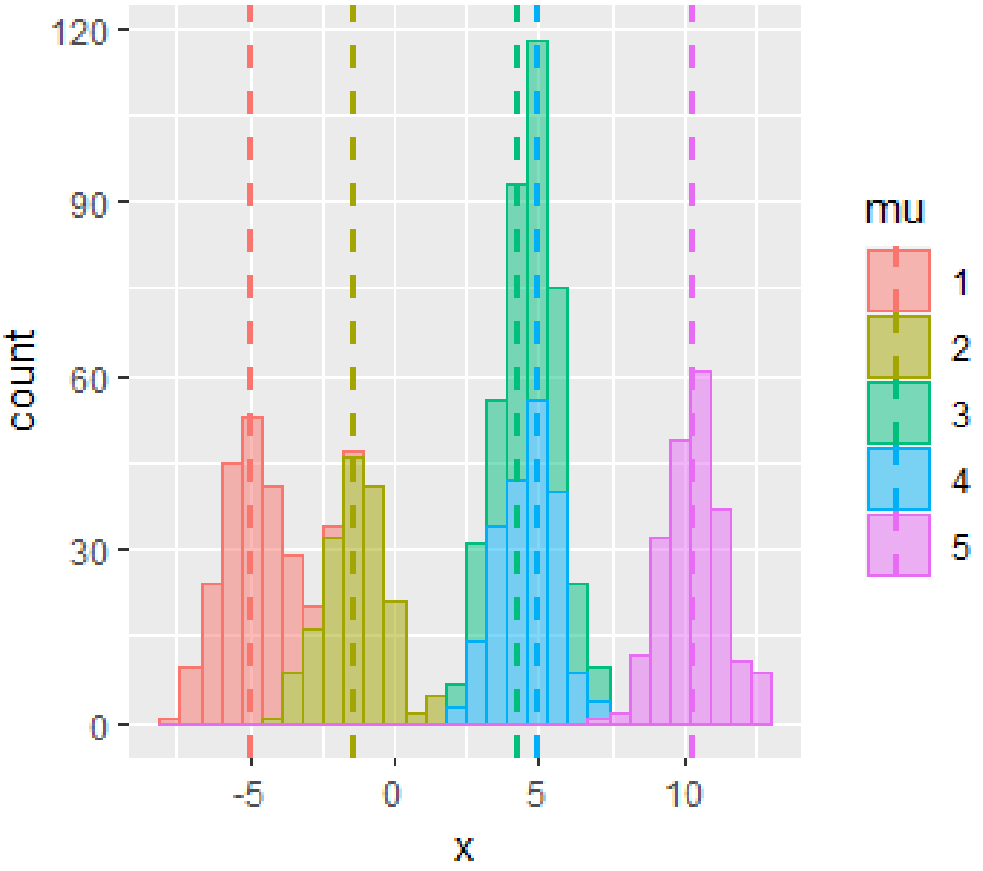


Figura 2

Veamos la diferencia entre las medias originales y las medias aproximadas:

```
> mk=sort(mk);mk
[1] -5.000065 -1.367583  4.306162  4.950129 10.352919
> mu
[1] -5.010925 -1.455997  4.624939  4.660598 10.386097
```

Figura 3

Como podemos observar las medias son bastante parecidas a las medias originales. Las mayores diferencias están presentes en las medias 3 y 4. Esto se debe a que dichas medias están muy cerca y por tanto el algoritmo tiende a fallar cuando las medias son muy parecidas.
Este gráfico nos indica la convergencia del ELBO partiendo de distintos valores iniciales:

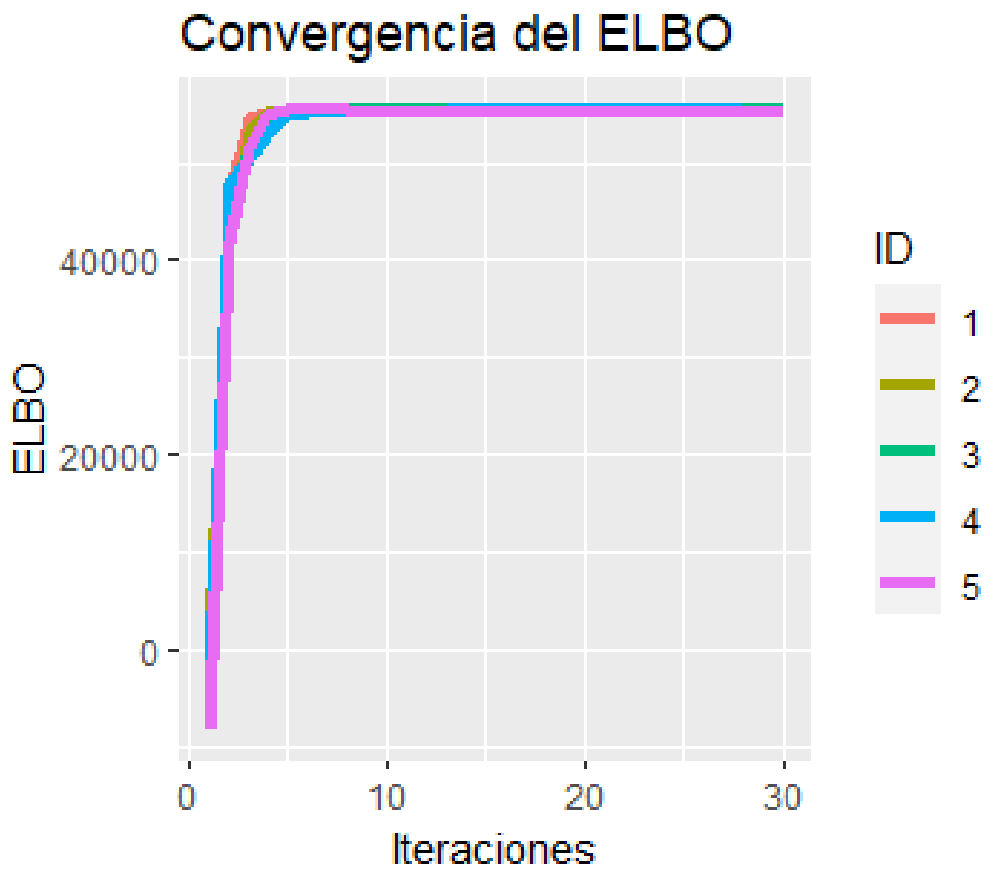


Figura 4

Podemos observar la rápida convergencia pues en menos de 10 iteraciones todos los valores son muy similares.
Además podemos comprobar en el codo que dependiendo de los datos iniciales tardará alguna iteración más o menos en llegar al mismo resultado que las

demás.
Por último veamos como iteración tras iteración el ELBO va convergiendo:

```
Iteración:  1 Diferencia de ELBO:  43106.4
Iteración:  2 Diferencia de ELBO:  9931.845
Iteración:  3 Diferencia de ELBO:  1997.746
Iteración:  4 Diferencia de ELBO:  229.7369
Iteración:  5 Diferencia de ELBO:  5.312356
Iteración:  6 Diferencia de ELBO:  3.792772
Iteración:  7 Diferencia de ELBO:  5.381752
Iteración:  8 Diferencia de ELBO:  5.271156
Iteración:  9 Diferencia de ELBO:  4.39941
Iteración:  10 Diferencia de ELBO:  3.424774
Iteración:  11 Diferencia de ELBO:  2.607488
Iteración:  12 Diferencia de ELBO:  1.990707
Iteración:  13 Diferencia de ELBO:  1.53885
Iteración:  14 Diferencia de ELBO:  1.202967
Iteración:  15 Diferencia de ELBO:  0.9424389
Iteración:  16 Diferencia de ELBO:  0.7283301
Iteración:  17 Diferencia de ELBO:  0.5410982
Iteración:  18 Diferencia de ELBO:  0.3676164
Iteración:  19 Diferencia de ELBO:  0.1989465
Iteración:  20 Diferencia de ELBO:  0.02903678
```

Figura 5

La imagen 5 nos muestra las diferencias de dos iteraciones consecutivas del ELBO de uno de los 5 inicios. Con los datos iniciales usados para dicha repetición podemos ver que el ELBO converge en 20 iteraciones. En otros casos las iteraciones podrán llegar al máximo sin que el ELBO converja, esto se arreglaría aumentando el número de iteraciones.

5.2. Ejemplo 2. Mezcla de gaussianas en dos dimensiones

Veremos ahora un ejemplo parecido al anterior pero ahora en dos dimensiones(Ver A.2 Código Ejemplo2) . Tenemos que tener en cuenta que ahora las normales son multivariantes, así como los datos, que son vectores de dimensión 2 y no escalares como antes. Esto hace que el código se vea afectado a la hora de realizar las multiplicaciones pero la base sigue siendo la misma.

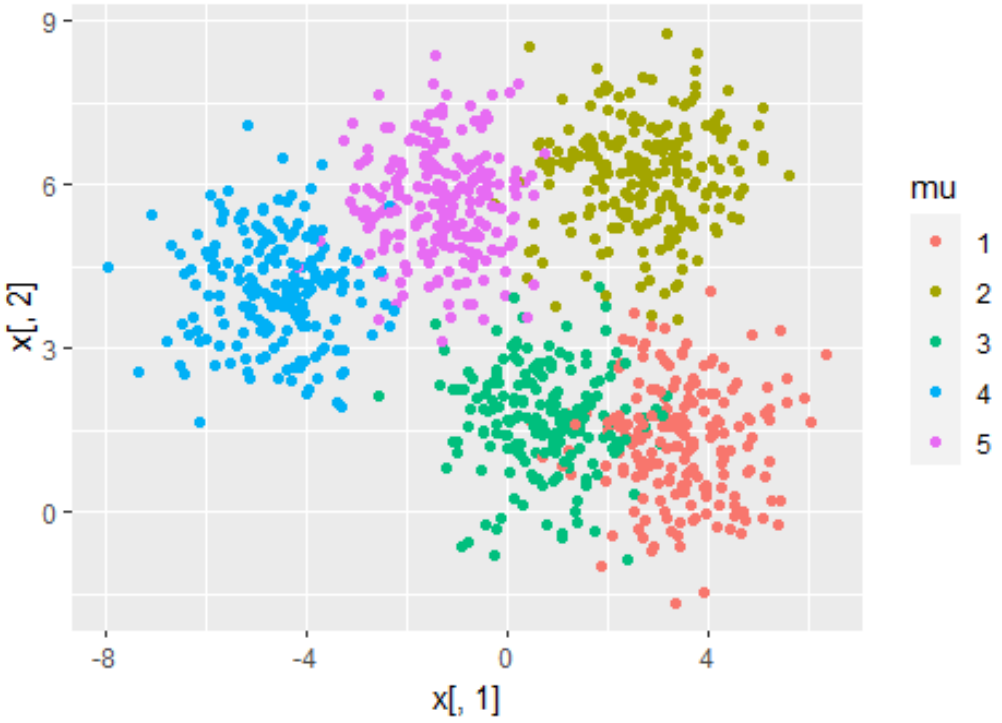


Figura 6

La imagen anterior nos representa nuestros datos.
Son 1000 observaciones aleatorias en las que cada color nos indica de qué grupo proviene cada dato. Igual que antes,usaremos 5 grupos distintos pero el hiperparámetro será $\sigma = 4$.

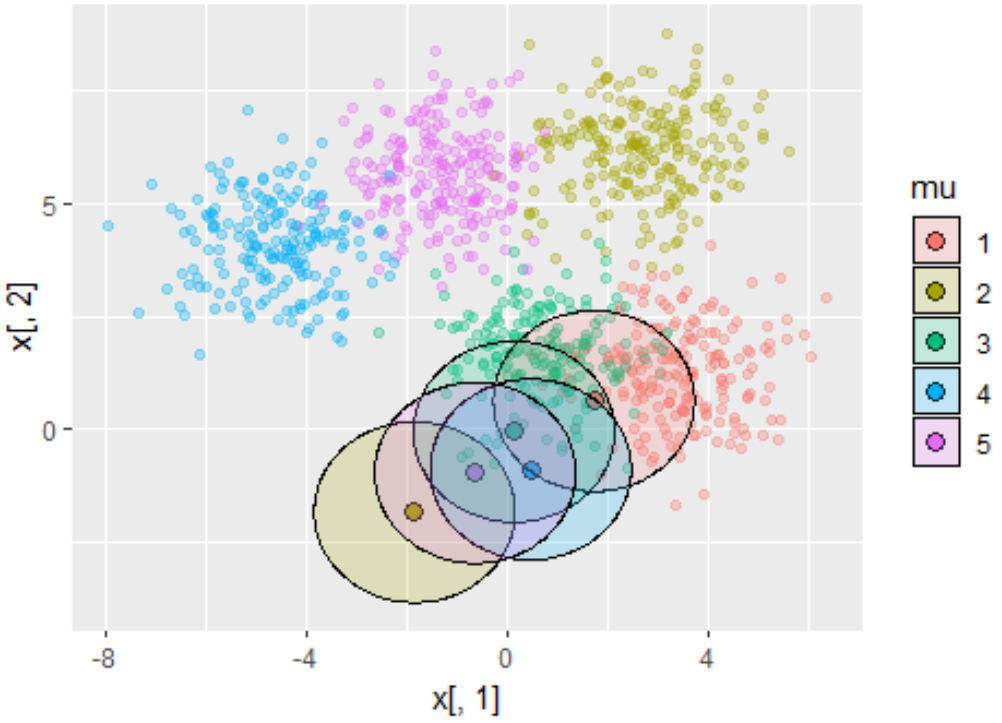


Figura 7

En la imagen 7 hemos dibujado donde se encuentran las medias iniciales que hemos generado para comenzar nuestro algoritmo. Están generadas aleatoriamente para intervenir lo menos posible.

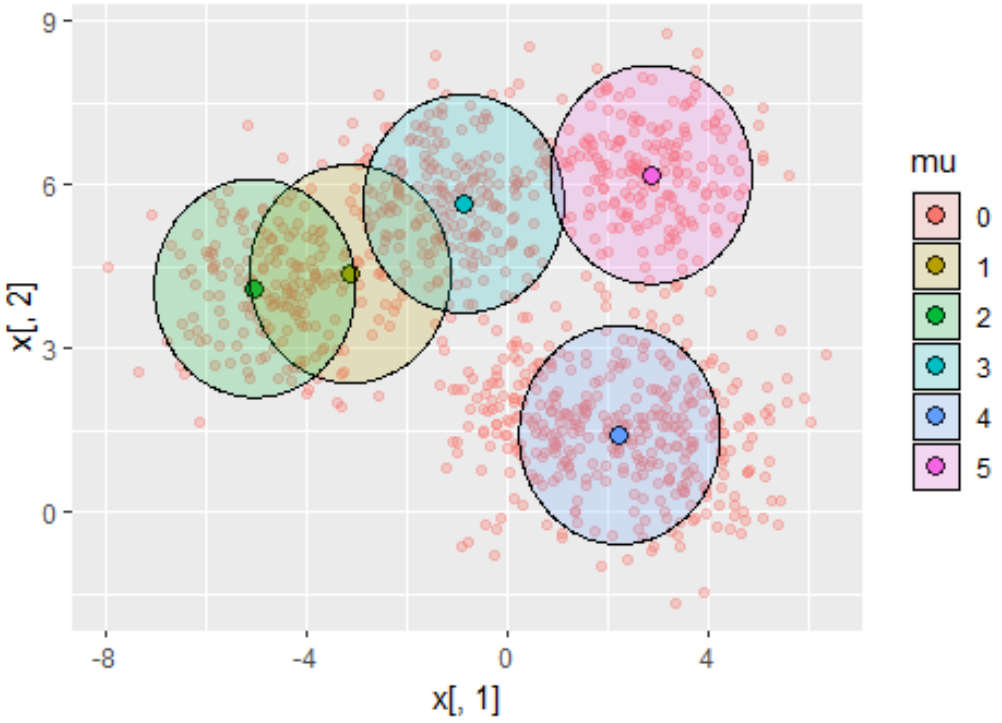


Figura 8

La imagen anterior nos dibuja donde se encuentran las medias tras 5 iteraciones. Como vemos todavía no han terminado de converger hacia el centro de cada grupo de observaciones.

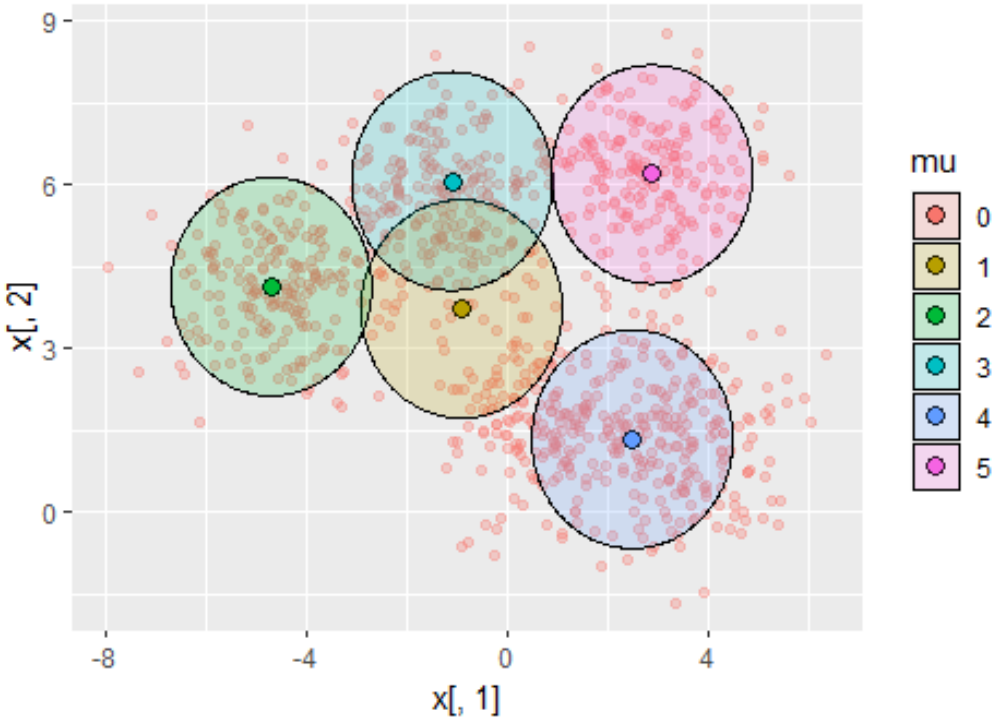


Figura 9

La imagen 9 nos dibuja donde se encuentran las medias tras 20 iteraciones. En este caso las medias se encuentran más próximas a los centros verdaderos que tras solo 5 iteraciones. A pesar de haber mejorado todavía pueden acercarse más.

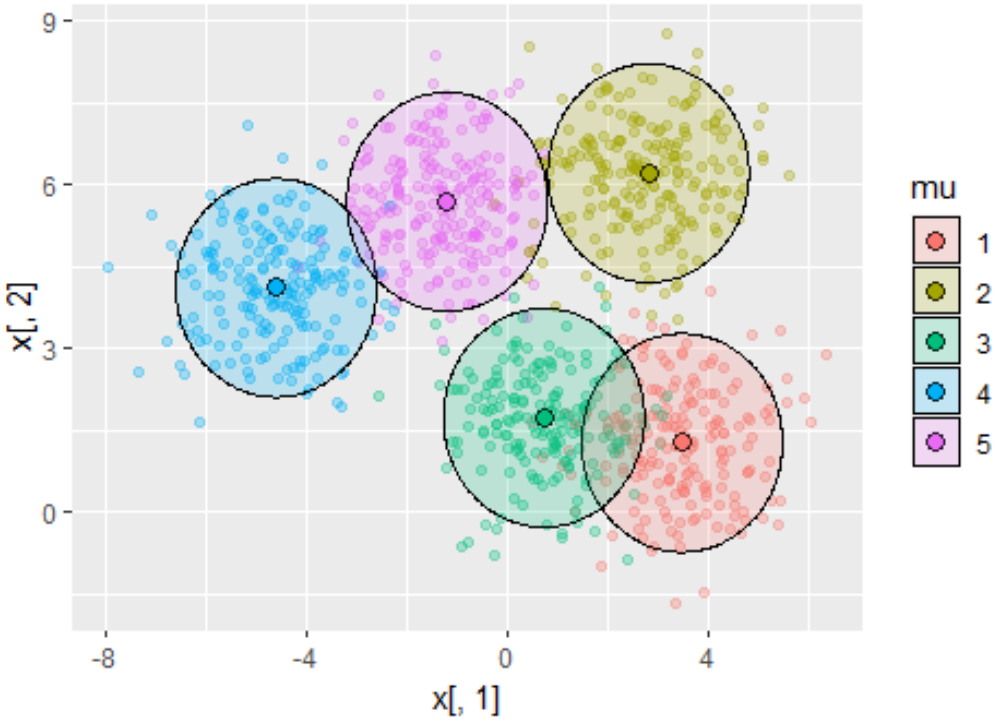


Figura 10

La imagen 10 representa las medias de cada grupo que hemos obtenido tras 50 iteraciones.
Observamos lo mucho que se parece a las medias verdaderas y el cambio realizado al realizar otras 30 iteraciones.

```
> mk
           [,1]     [,2]
[1,]  0.7180882 1.725996
[2,]  3.4619873 1.269371
[3,] -1.2056658 5.682920
[4,] -4.6090613 4.107232
[5,]  2.7952615 6.206112
> mu
           [,1]     [,2]
[1,]  0.8124281 1.637103
[2,]  3.4713545 1.287129
[3,] -1.2296674 5.670851
[4,] -4.5012768 4.058144
[5,]  2.7344218 6.211690
```

Figura 11

La imagen 11 nos muestra las medias verdaderas y las medias obtenidas tras 50 iteraciones del algoritmo. Observemos su gran similitud.
Ahora para la figura 12, utilizaremos la función *kmeans* de la librería *mclust* para comparar las medias obtenidas con nuestro algoritmo, las medias reales y las medias de *mclust*. Observamos que en este caso, nuestro algoritmo mejora la aproximación de las kmedias.

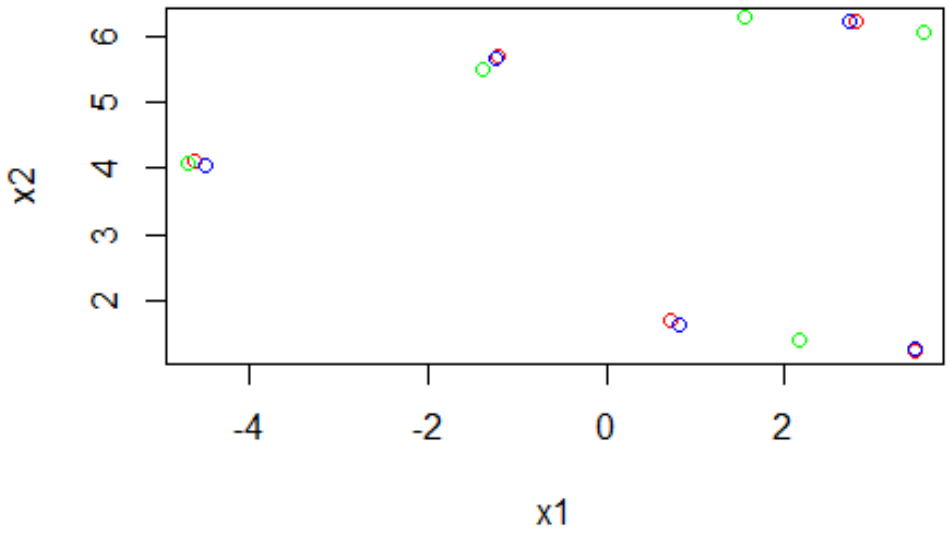


Figura 12

En rojo las medias obtenidas con inferencia variacional, en azul las medias reales y en verde las kmedias.

5.3. Ejemplo 3. Análisis de imágenes

En este último ejemplo vamos a realizar un análisis de los histogramas de las imágenes del conjunto de datos de Kaggle llamado "Landscape Recognition | Image Dataset | 12k Images"(Ver A.3 Código Ejemplo3.). Dicho conjunto de imágenes contiene 5 grupos de distintas imágenes(montaña, costa, desierto, bosque y glaciar). En nuestro caso eliminaremos el grupo "montaña" y trabajaremos con los otros 4 grupos. Tenemos en cada grupo 2000 imágenes para nuestro modelo de entrenamiento y 100 imágenes test. Esto hace un total de 8000 imágenes de entrenamiento y 400 de test.
Como trabajar con todos los datos de las imágenes es muy pesado, estudiaremos los histogramas de rojo, azul y verde de cada imagen. Tomaremos

histogramas de tamaño 100 y al concatenarlos obtendremos un vector de tamaño 300. De esta forma trabajaremos con 8000 observaciones de tamaño 300.
A continuación realizaremos nuestro algoritmo de ascenso coordenado con la mezcla de gaussianas para encontrar 4 histogramas tipo, es decir, las medias. Finalmente veremos los 400 histogramas de las imágenes a ver a que histograma de los que hemos obtenido se parece más. En nuestro conjunto de imágenes hay oasis en los desiertos que tendrán un histograma más parecido a un bosque e incluso bosques con hojas de color rojo que recordarán a un atardecer en el desierto.

Estas primeras figuras son los histogramas de rojo, verde y azul de la primera imagen.

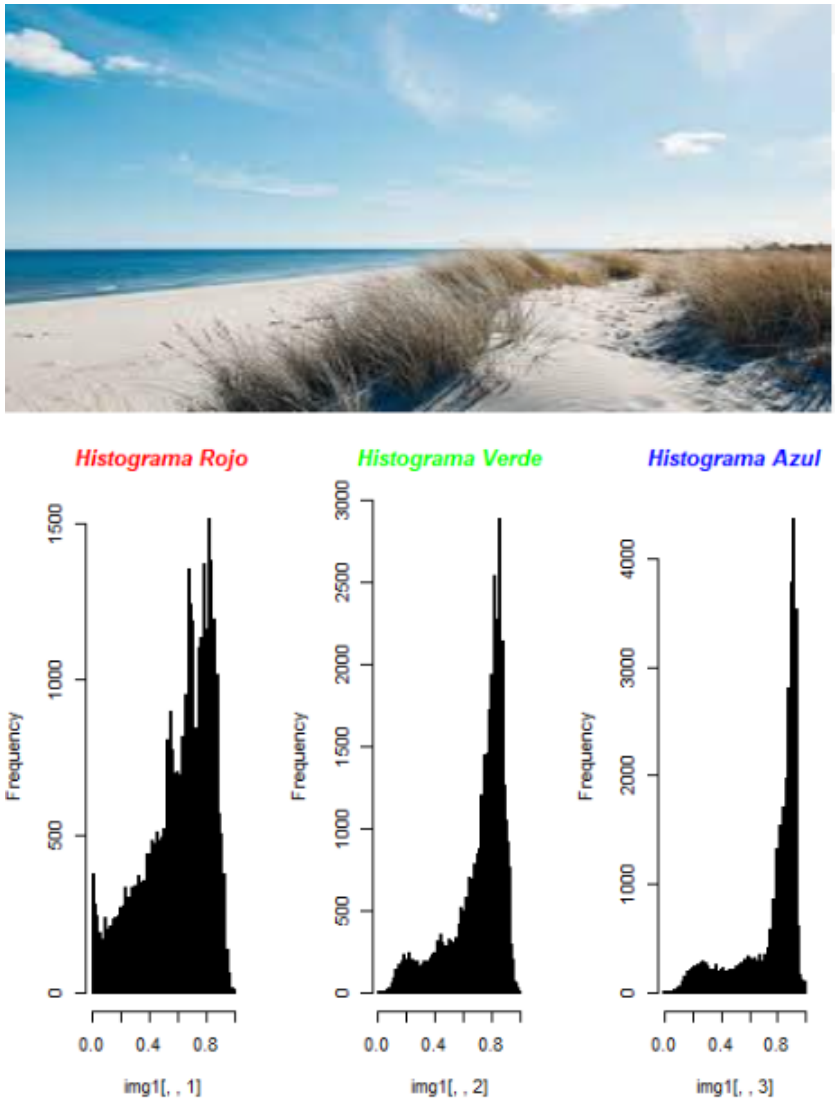


Figura 13

A continuación mostramos los 4 histogramas normalizados que hemos obtenido después de 30 iteraciones:

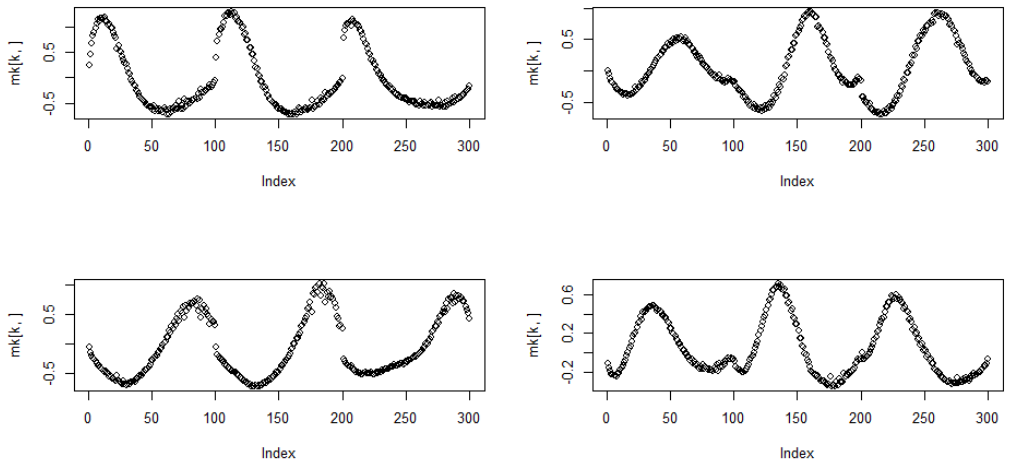


Figura 14

Después de ver en qué grupo se encuentran las 400 imágenes, mostramos su clasificación en los 4 grupos y algunas imágenes cuyo histograma es parecido:

	Grupo 1	Grupo 2	Grupo 3	Grupo 4
Playa	6	53	11	34
Desierto	4	43	5	48
Bosque	12	9	71	7
Glaciar	79	6	10	5

Figura 15

Figura 16

Figura 17

Como podemos observar, las imágenes de glaciares y bosques son muy fáciles de clasificar esto se debe a sus altos valores de verdes y azules respectivamente. Sin embargo, los histogramas de playas y desiertos son muy similares, esto se debe a que se confunde el azul del cielo con el azul del oceáno y el color de las playas con el de las dunas.

6. Inferencia variacional estocástica

Las aplicaciones actuales de modelos probabilísticos requieren de analizar datos masivos. Sin embargo, muchos modelos que funcionan con datos no tan grandes, no escalan y no podemos utilizarlos para grandes conjuntos de datos. El algoritmo de actualización por coordenadas no es una excepción. Recordemos las fórmulas de dicho modelo en la mezcla de gaussianas:

$$m_k = \frac{\sum_{i=1}^{n} \varphi_{ik} x_i}{\frac{1}{\sigma^2} + \sum_{i=1}^{n} \varphi_{ik}}, \qquad s_k^2 = \left(\frac{1}{\sigma^2} + \sum_{i=1}^{n} \varphi_{ik} \right)^{-1} \qquad \forall k$$

$$\varphi_{ik} \approx exp\left\{ E\left[\mu_k\right] x_i - E\left[\mu_k^2\right]/2 \right\} \qquad \forall i, k$$

Notemos que el sumatorio que aparece en s_k^2 es sobre todo el conjunto de datos. Para m_k tenemos que calcular otro sumatorio para todos los datos, además de usar el sumatorio mencionado antes. Por otro lado para todo dato de nuestro conjunto debemos calcular φ_{ik} y posteriormente normalizar este parámetro. Estos cálculos escalan muy rápido cuando el conjunto de datos crece, por tanto debemos buscar alternativas.

Una opción al ascenso coordenado es usar optimación basada en el gradiente, como por ejemplo el ascenso de gradiente, el cual optimiza el ELBO calculando y siguiendo su gradiente en cada iteración. Este método fue explicado al comienzo de la sección 3.1. Como dijimos entonces, este método tiene problemas a la hora de ser escalado. Ahora veremos una forma alternativa para aplicarlo en grandes conjuntos de datos.

El ascenso de gradiente estocástico es un método de ascenso de gradiente en el que se usan parcialmente los datos, tomando una muestra aleatoria de los mismos. En este método, igual que durante todo el trabajo, tenemos una familia variacional $\mathcal{Q}$ en la que las densidades dependen del parámetro θ y actúan sobre las variables latentes z. Sin embargo, ahora optimizaremos el parámetro θ con el objetivo de estar más próximos de la distribución a posteriori. Para ello tendremos que maximizar la cota inferior :

$$ELBO = E_{Z \sim q_\theta(z)} \left[\log p(x|z) p(z) - \log q_\theta(z) \right]$$

Usaremos ascenso de gradiente con respecto a los parametros θ de q. Actualizaremos el valor del parámetro siguiendo ascenso de gradiente:

$$\theta^{(t+1)} = \theta^{(t)} + \gamma_t \cdot \nabla_\theta ELBO^{(t+1)}$$

Como vimos en la sección 3.1, γ_t es la tasa de aprendizaje. Si elegimos γ_t de manera correcta el algortimo de ascenso de gradiente convergerá rápidamente hacia un máximo local, no podemos asegurar que sea global. El algorimo va recorriendo el dominio de la función cambiando los parámetros para que $q_\theta(z)$ se encuentre lo más cercano de la distribución a posteriori posible.
Notemos que en cada iteración, si tenemos el actual valor de θ dado z, podemos calcular el integrando $\log p(x|z)p(z) - \log q_\theta(z)$. Esto se debe a que conocemos la densidad $\log q_\theta(z)$ con la que vamos a aproximar(por ejemplo una gaussiana o una exponencial) y también suponemos que tenemos la densidad conjunta $p(x, z)$ o en su versión extendida $p(x|z)p(z)$.
Ahora bien, como la integral normalmente no se puede calcular usaremos estimaciones de Monte Carlo para calcular la esperanza:

$$E_{Z\sim q_\theta(z)} \approx \frac{1}{N}\sum_{i=1}^{N} \log p(x|z_i)p(z_i) - \log q_\theta(z_i)$$

El nombre de estocástico proviene tanto de usar las estimaciones de Monte Carlo como de muestrar los datos aleatoriamente.
Veamos como funciona en el caso de la mezcla de gaussianas:

1. Obtenemos un dato aleatoriamente del conjunto total de los datos.

2. Usamos los parámetros globales actuales (μ) para calcular los parámetros locales óptimos (c_i) para el dato que hemos muestreado.

3. Ajustamos los parámetros globales usando ascenso de gradiente. De esta forma reemplazamos los sumatorios sobre todos los datos.

6.1. El gradiente de la función objetivo

A continuación calcularemos el gradiente de la función objetivo, es decir, el ELBO, con respecto al parámetro θ.

$$\begin{aligned} &\nabla_\theta E_{Z\sim q_\theta(z)} \left[\log p(x|z)p(z) - \log q_\theta(z)\right] = \\ &= \nabla_\theta \int q_\theta(z) \cdot \left[\log p(x|z)p(z) - \log q_\theta(z)\right] dz \\ &= \int \nabla_\theta \; q_\theta(z) \cdot \left[\log p(x|z)p(z) - \log q_\theta(z)\right] dz \end{aligned} \tag{28}$$

Donde en la última igualdad hemos usado la derivación bajo el signo integral. Notemos que para poder usarlo la familia Q a la que pertenece $q_\theta(z)$ debe ser lo suficientemente regular para que se cumplan las hipótesis de dicho teorema.
Ahora queremos llegar a $E_q[\nabla f(z)]$ para aproximar la esperanza(con el método de Monte Carlo) a la vez que optimizamos usando el gradiente.
Para llegar a dicho objetivo podemos proceder de dos formas distintas que nos darán dos modelos distintos de inferencia variacional.

Reparametrizando las densidades

El objetivo es que tras reparametrizar la nueva densidad no dependa del parámetro θ. Esto lo podemos expresar de la siguiente manera:
$q_\theta(z) \to q(g(z))$ siendo g la función que hace la reparametrización.
En el caso de la familia de las gaussianas tenemos que: $q(z) = \mathbf{N}(\mu, \sigma^2)$, en este caso $\theta = \{\mu, \sigma^2\}$ y $\eta = g(z) = \frac{z-\mu}{\sigma^2}$, luego $\eta = \mathbf{N}(0, 1)$.
Entonces lo anterior pasa a ser:

$$\nabla_\theta E_{Z\sim q_\theta(z)} \left[\log p(x|z)p(z) - \log q_\theta(z)\right] = E_{\eta\sim q_\eta(z)} \nabla_\theta \left[\log p(x|z)p(z) - \log q_\theta(z)\right]$$

Esto se debe a que la integral no se toma con respecto a $q_\theta(z)$ si no a $q(\eta)$.
Ahora sustituiremos z por $g_\theta^{-1}(\eta)$ que depende de θ. Entonces:

$$E_{\eta\sim q_\eta(z)} \nabla_\theta \left[\log p(x|g_\theta^{-1}(\eta))p(g_\theta^{-1}(\eta)) - \log q_\theta(g_\theta^{-1}(\eta))\right] |det\ J|$$

Donde $|det\ J|$ es el determinante de la transformación.
De esta forma hemos simplificado el gradiente de la esperanza y podemos usar estimaciones de Monte Carlo muestreando η para reemplazar la esperanza:

$$\begin{aligned}\nabla_\theta ELBO^{(t+1)} \approx \frac{1}{N} \sum_{i=1}^{N} E_{\eta\sim q_\eta(z)} \nabla_\theta |det\ J| [\log p(x|g_\theta^{-1}(\eta_i))p(g_\theta^{-1}(\eta_i)) \\ - \log q_\theta(g_\theta^{-1}(\eta_i))]\end{aligned} \tag{29}$$

Finalmente actualizaremos el valor del parámetro:

$$\theta^{(t+1)} = \theta^{(t)} + \gamma_t \cdot \nabla_\theta ELBO^{(t+1)} \tag{30}$$

Usando la reparametrización obtenemos un modelo conocido como derivación automática. Ver Kucukelbir, A. (2017). La derivación automática se refiere al uso de técnicas computacionales que permiten calcular derivadas de manera automática y precisa, evitando la necesidad de derivar manualmente las funciones.

Usando el gradiente logarítmico

En este caso usaremos una expresión muy conocida :

$$\nabla_\theta q_\theta(z) = q_\theta(z)\nabla_\theta \log q_\theta(z) \tag{31}$$

Volvamos a la expresión (28), si extendemos dicha integral tenemos que:

$$\int \nabla_\theta \ (q_\theta(z) \cdot [\log p(x|z)p(z) - \log q_\theta(z)])dz =$$

$$= \int \nabla_\theta \ q_\theta(z)([\log p(x|z)p(z) - \log q_\theta(z)] \, dz$$

$$+ \int \nabla_\theta \left[\log p(x|z)p(z) - \log q_\theta(z)\right] q_\theta(z)dz$$

Notemos que el segundo término es igual a cero porque:

$$\begin{aligned}\int \nabla_\theta \left[\log p(x|z)p(z) - \log q_\theta(z)\right] q_\theta(z)dz &= -\int \frac{1}{q_\theta(z)}\nabla_\theta q_\theta(z) \ q_\theta(z)dz \\ &= -\int \nabla_\theta q_\theta(z)dz = -\nabla_\theta \int q_\theta(z)dz \\ &= -\nabla_\theta \ 1 = 0\end{aligned}$$

En la penúltima igualdad hemos usado la derivación bajo el signo integral en el sentido opuesto al habitual y para la última hemos tenido en cuenta que $q_\theta(z)$ es una densidad y por tanto su integral es 1.
Para el primer término usaremos la fórmula (31):

$$\int \nabla_\theta \ q_\theta(z)([\log p(x|z)p(z) - \log q_\theta(z)] \, dz = \int q_\theta(z)\nabla_\theta \log q_\theta(z) \, (\log p(x|z)p(z) - \log q_\theta(z)) \ dz$$

Y como podemos ver este término ya lo podemos escribir como una esperanza:

$$E_{Z\sim q_\theta(z)} \left[\nabla_\theta \log q_\theta(z) \, (\log p(x|z)p(z) - \log q_\theta(z))\right]$$

Ahora podemos usar ascenso de gradiente estocástico:
Empezaremos muestreando un $z_i \sim q_\theta^{(t)}(z)$, donde t indica la iteración.

A continuación calcularemos una versión estocástica del gradiente del ELBO de la siguiente forma :

$$\nabla_\theta ELBO^{(t+1)} \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log q_{\theta^{(t)}}(z_i)\left([\log p(x|z_i)p(z_i) - \log q_{\theta^{(t)}}(z_i)]\right) \quad (32)$$

Finalmente actualizaremos el valor del parámetro:

$$\theta^{(t+1)} = \theta^{(t)} + \gamma_t \cdot \nabla_\theta ELBO^{(t+1)}$$

Un modelo que usa esta forma de operar con el gradiente se conoce como modelo de caja negra. Ver Ranganath, R., Gerrish, S., & Blei, D. (2013). El nombre de caja negra en ciencias de la computación se refiere a un sistema o componente cuyo funcionamiento interno no son conocidos. En la inferencia variacional, en este modelo no se tiene un conocimiento explícito sobre la forma funcional de la distribución posterior que se desea aproximar. Este modelo utiliza también técnicas de diferenciación automática para calcular gradientes de forma eficiente y precisa.

6.2. El gradiente natural

Un problema que tiene el uso del gradiente usual es que en muchas familias de distribuciones, el espacio de parámetros puede tener una estructura no lineal o curva y como este sigue es una dirección recta en el espacio de parámetros puede no ser del todo eficiente.
A continuación definieremos el gradiente natural, que ajusta los parámetros en la dirección del gradiente pero tomando en cuenta la estructura curva del espacio de parámetros. Al considerar la geometría intrínseca de la familia de distribuciones, el gradiente natural puede adaptarse mejor a la forma de la distribución y conducir a actualizaciones más eficientes de los parámetros.
Para definir el gradiente natural debemos conocer la matriz de Fisher:

Definición 3. La matriz de Fisher se define como:

$$F \;:=\; E_{z\sim q_\theta(z)}\left[\nabla_\theta \log q_\theta(z) \nabla_\theta \log q_\theta(z)^T\right]$$

Ver Amari, Shun'ichi Capítulo 12 (2010).
A veces esta esperanza puede ser complicada de calcular y podemos usar estimaciones de Monte Carlo para calcularla sobre las variables latentes:

$$F \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log q_\theta(z_i) \nabla_\theta \log q_\theta(z_i)^T$$

Definición 4. El gradiente natural se define como:

$$\hat{\nabla}_\theta := F^{-1}\nabla_\theta$$

Ver Amari, Shun'ichi Capítulo 12 (2010).
El gradiente natural ajusta los parámetros en la dirección del gradiente pero tomando en cuenta la estructura curva del espacio de parámetros. El uso del gradiente natural puede mejorar la eficiencia y la convergencia del proceso de optimización en inferencia variacional.
Si sustituimos el gradiente usual por el gradiente natural en (32) tenemos las siguientes ecuaciones:

$$\hat{\nabla}_\theta ELBO^{(t+1)} \approx F^{-1} \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log q_{\theta^{(t)}}(z_i) \left([\log p(x|z_i)p(z_i) - \log q_{\theta^{(t)}}(z_i)]\right)$$

Y la actualización del parámetro se escribiría :

$$\theta^{(t+1)} = \theta^{(t)} + \gamma_t \cdot \hat{\nabla}_\theta ELBO^{(t+1)}$$

7. Conclusiones y posibles extensiones

Hemos presentado la inferencia variacional como una técnica computacional útil en el contexto de la Inferencia Bayesiana. Posteriormente hemos definido y trabajado la divergencia de Kullback-Leibler, una forma de medir distancia entre densidades. En la siguiente sección, introdujimos la familia variacional de campo medio, donde suponemos que las variables latentes son independientes entre sí. Continuamos construyendo un algoritmo de ascenso coordenado en la familia de campo medio. Luego mostrábamos tres ejemplos de dicho algoritmo considerando una mezcla bayesiana de gaussianas. Terminábamos explicando la inferencia variacional estocástica, una forma de escalar la inferencia variacional para grandes conjuntos de datos. Para finalizar el trabajo hablaremos de posibles extensiones que existen en la inferencia variacional.

En el trabajo nos hemos centrado en optimizar la divergencia $KL(q||p(\cdot|x))$. Probablemente estaría bien manejar otras divergencias porque la divergencia KL en algunos problemas no funciona. Otra opción es optimizar otras divergencias que nos permitan calcularlas sin conocer la distribución a posteriori. También podemos estudiar la divergencia KL en el sentido contrario, $KL(p(\cdot|x)||q)$. Además existen otras cotas inferiores mejores que el ELBO. Estas opciones por lo general resultan más complicadas de desarrollar y de implementar.

Aunque es flexible, la familia variacional de campo medio asume independencia entre las variables latentes, impidiendo su uso en la aproximación de modelos que no tengan independencia. La familia de campo medio nos ayuda a escalar el problema pero limita la expresión de la familia variacional. Un posible desarrollo se centraría en estudiar este problema.

A. Código en R para los ejemplos

A.1. Código Ejemplo 1

```
    #Algoritmo CAVI para la mezcla bayesiana de gaussianas

library(mvtnorm) #Para las normales multivariantes
library(extraDistr) #Para la distribucion multinomial
library(pracma) #Para integrar
library(matlib) #Para resolver ecuaciones
library(ggplot2) #Para dibujar

library(dplyr)
library(MASS)
library(ggforce)

#Hiperparametros
sigma = 3 #Varianza de mu
K = 5 #Numero de clusters o grupos
n = 1000 #Numero de observaciones

########################################
       #gaussiana en una dimension#
########################################

mu = rnorm(K,mean = 0, sd=sigma) # Medias
mu = sort(mu) #Para despues dibujarlas y compararlas bien
cs = rcat(n, rep(1/K,K)) # Asignaciones
x  = rnorm(n,mean=mu[cs], sd=1) #Generamos los datos

#Dibujamos los datos
df = data.frame(x=x, mu=as.factor(cs))
ggplot(df, aes(x=x,color=mu, fill=mu)) +geom_histogram(alpha=0.5)

#Funcion para calcular el ELBO#
ELBO=function(mk,sk2,phis){
  t = sk2+mk^2
  a = -(1/(2*sigma^2))*sum(t)
  b = 2*sum(sweep(sweep(phis,MARGIN = 2,mk,'*'),MARGIN =
      1,x,'*'))-0.5*sum(sweep(phis,MARGIN = 2,t,'*' ))
```

```
  c = -0.5*sum(log(2*pi*sk2))
  d = sum(phis*log(phis))
  return(a+b+c+d)
}
#creamos una lista de data.frames para poder dibujar las distintas
    iteraciones del ELBO
lista_df <- list()

#Realizamos el algoritmo CAVI 5 veces para comparar su
    convergencia.
for (z in 1:5){
### Algoritmo CAVI ###
mk= rnorm(K) #Generamos la muestra inicial
sk2=rgamma(K,5) #Varianzas aleatorias positivas
phis=rdirichlet(n,c(1,1,1,1,1)) #Asignaciones aleatorias


iter=30 #Iteraciones
elbos=rep(NA,iter+1) #Inicializamos el ELBO
elbos[1]=ELBO(mk,sk2,phis)
#Con las ecuaciones obtenidas anteriormente realizamos hasta la
    convergencia del ELBO
#o hasta llegar al numero de iteraciones
for (i in 1:iter){
  phis.new=matrix(nrow=n,ncol = K)
  #Actualizamos las asignaciones
  for (j in 1:n){
    phis.new[j,]=exp(x[j]*mk-0.5*(sk2+mk^2))
    phis.new = phis.new/rowSums(phis.new)
  }
  phis=phis.new

  mk.new=rep(NA,K)
  sk2.new=rep(NA,K)
  #Actualizamos los componentes de la mezcla
  for (k in 1:K){
    sk2.new[k] = 1/(1/sigma^2+sum(phis[,k]))
    mk.new[k] = sk2.new[k]*sum(phis[,k]*x)
  }

```

```
  sk2=sk2.new
  mk=mk.new
  #Calculamos el ELBO y comprobamos su convergencia.
  elbos[i+1]=ELBO(mk,sk2,phis)
  cat("Iteracion: ",i, "Diferencia de ELBO:
      ",abs(elbos[i+1]-elbos[i]),"\n")
  if (abs(elbos[i+1]-elbos[i])<0.1) break
}
#Anadimos los datos a la lista de los data.frame
lista_df[z] <- list(data.frame(Iter = seq(1, i, 1), ELBO =
    elbos[1:i]))

}
#Generamos un data.frame completo y dibujamos el ELBO y el numero
    de iteraciones.
df_completo <- bind_rows(lista_df, .id = "ID")
ggplot(data = df_completo, aes(x = Iter, y = ELBO, color = ID))
    +geom_line(size=1.5) +
  labs(title="Convergencia del ELBO ", x="Iteraciones", y="ELBO")

#Ordenamos las medias y las comparamos con las originales
mk=sort(mk);mk
mu
sk2

#Dibujamos los datos con las medias que hemos calculado
    previamente.
ggplot(df,aes(x=x,color=mu,fill=mu)) + geom_histogram(alpha=0.5)+
  geom_vline(data=data.frame(x=mk),aes(xintercept=x,
      color=as.factor(c(1,2,3,4,5))),linetype="dashed",size=1)

#Comprobemos como generamos los datos con las medias obtenidas

cs2 = rcat(n, rep(1/K,K)) # Asignaciones
xnew  = rnorm(n,mean=mk[cs2], sd=1) #Generamos los datos,

df2 = data.frame(x=xnew, mu=as.factor(cs2))
ggplot(df2, aes(x = x, color = mu, fill = mu)) +
    geom_histogram(alpha = 0.5)
```

A.2. Código Ejemplo 2

```
1
2  ######################
3  ##   2D gaussiana   ##
4  ######################
5
6  #Hiperparametro
7  sigma = 4 #Varianza de mu
8  K = 5 #Numero de clusters o grupos
9  n = 1000 #Numero de observaciones
10
11 #Generamos los datos que ahora son datos multivariantes.
12 I=diag(c(1,1))
13 mu = rmvnorm(K, mean=c(0,0), sigma = sigma^2*I)
14 cs=rcat(n, rep(1/K,K))
15 x=mu[cs,] + rmvnorm(n, mean=c(0,0),sigma = I)
16
17 #Funcion para ordenar las medias segun su norma, nos servira para
       compararlas despues
18 ordenarmedias = function(medias){
19   normas = apply(medias, 1, function(x) sqrt(sum(x^2)))
20   medias = medias[order(normas),]
21   return(medias)
22 }
23
24
25 #Dibujemos los datos
26 df = data.frame(x = x, mu = as.factor(cs))
27 ggplot(df, aes(x = x[,1], y = x[,2], color = mu, fill = mu)) +
       geom_point()
28
29 ###Algoritmo CAVI###
30 #Generamos los datos iniciales pero teniendo en cuenta que ahora
       son multivariantes,
31 #por tanto tomamos I
32 mk = rmvnorm(K, mean = c(0,0), sigma = I)
33 sk2 = rgamma(K, 5)
34 phis = rdirichlet(n, c(1,1,1,1,1))
35
```

```r
#Funcion para dibujar los datos y las medias
plotClusters = function(){
  ggplot(df, aes(x = x[,1], y = x[,2], color = mu, fill = mu)) +
    geom_point(alpha = 0.3) +
    geom_point(data = data.frame(x1 = mk[,1], x2 = mk[,2], mu =
        as.factor(c(3,5,4,2,1))),
               aes(x = x1, y = x2, color = mu), size = 3, colour =
                   "black") +
    geom_point(data = data.frame(x1 = mk[,1], x2 = mk[,2], mu =
        as.factor(c(3,5,4,2,1))),
               aes(x = x1, y = x2, color = mu), size = 2) +
    geom_circle(data = data.frame(x1 = mk[,1], x2 = mk[,2], mu =
        as.factor(c(3,5,4,2,1))),
                aes(x0 = x1, y0 = x2, r = sigma/2, fill = mu, x =
                    x1, y = x2), color = "black",alpha = 0.2)
}

plotClusters()

#Numero de iteraciones
iter = 50
#Realizamos el algoritmo hasta llegar a las 30 iteraciones
#Notemos que ahora no son escalares y por tanto tenemos que tener
    cuidado al multiplicar vectores.
for (i in 1:iter){
  phis.new = matrix(nrow = n, ncol = K)

  for (j in 1:n){
    for (k in 1:K){
      phis.new[j,k] = exp(t(x[j,]) %*% mk[k,] - 0.5 * (2 * sk2[k]
          + t(mk[k,])
                                                       %*% mk[k,]))
    }
    phis.new = phis.new / rowSums(phis.new)
  }
  phis = phis.new

  mk.new = matrix(rep(NA, K * 2), ncol = 2)
  sk2.new = rep(NA, K)

```

```
  for (k in 1:K){
    sk2.new[k] = 1 / (1 / sigma^2 + sum(phis[,k]))
    mk.new[k,] = sk2.new[k] * colSums(phis[,k] * x)
  }

  sk2 = sk2.new
  mk = mk.new

  #Para guardar la evolucion tras 5 iteraciones
  if (i == 5){
    df_iter5 <- df
    df_iter5$mu <- as.factor(colSums(phis == max.col(phis)))
    mk_iter5 <- ordenarmedias(mk)
  }
  #Para guardar la evolucion tras 20 iteraciones

  if (i == 20){
    df_iter20 <- df
    df_iter20$mu <- as.factor(colSums(phis == max.col(phis)))
    mk_iter20 <- ordenarmedias(mk)
  }
}

#Para dibujar como estan las medias tras 5 iteraciones
ggplot(df_iter5, aes(x = x[,1], y = x[,2], color = mu, fill = mu))
    +
  geom_point(alpha = 0.3) +
  geom_point(data = data.frame(x1 = mk_iter5[,1], x2 =
      mk_iter5[,2], mu = as.factor(c(4,1,3,2,5))),
            aes(x = x1, y = x2, color = mu), size = 3, colour =
                "black") +
  geom_point(data = data.frame(x1 = mk_iter5[,1], x2 =
      mk_iter5[,2], mu = as.factor(c(4,1,3,2,5))),
            aes(x = x1, y = x2, color = mu), size = 2) +
  geom_circle(data = data.frame(x1 = mk_iter5[,1], x2 =
      mk_iter5[,2], mu = as.factor(c(4,1,3,2,5))),
             aes(x0 = x1, y0 = x2, r = sigma/2, fill = mu, x = x1,
                 y = x2), color = "black", alpha = 0.2)

#Para dibujar como estan las medias tras 20 iteraciones
```

```r
ggplot(df_iter20, aes(x = x[,1], y = x[,2], color = mu, fill = mu)) +
  geom_point(alpha = 0.3) +
  geom_point(data = data.frame(x1 = mk_iter20[,1], x2 = mk_iter20[,2], mu = as.factor(c(4,1,3,2,5))),
             aes(x = x1, y = x2, color = mu), size = 3, colour = "black") +
  geom_point(data = data.frame(x1 = mk_iter20[,1], x2 = mk_iter20[,2], mu = as.factor(c(4,1,3,2,5))),
             aes(x = x1, y = x2, color = mu), size = 2) +
  geom_circle(data = data.frame(x1 = mk_iter20[,1], x2 = mk_iter20[,2], mu = as.factor(c(4,1,3,2,5))),
              aes(x0 = x1, y0 = x2, r = sigma/2, fill = mu, x = x1, y = x2), color = "black", alpha = 0.2)

plotClusters()

#Para verlas en orden
mk=ordenarmedias(mk)
mu=ordenarmedias(mu)

#Comparamos el resultado con el algoritmo kmeans
#Dibujamos donde se encuentran las medias artificiales, las reales y las kmedias.
kmeans_result <- kmeans(x, centers = 5)
plot(mk,col="red",xlab = "x1",ylab = "x2");
points(mu,col="blue")
points(kmeans_result$centers,col="green")
```

A.3. Código Ejemplo 3

```r
library(tidyverse)
library(mvtnorm) #Para las normales multivariantes
library(extraDistr) #Para la distribucin categorica
library(pracma)
library(matlib) #Para resolver ecuaciones
library(ggplot2)
library(dplyr)
library(MASS)
```

```r
library(ggforce)
library(imager)
library(jpeg)

#Cargamos las imagenes para trabajar con ellas
dir <- "C:/Users/Miguel/Desktop/TFG/Datos/Train/"
files <- list.files(dir, full.names = TRUE)
imgs <- list()

for (file in files) {
  img <- load.image(file)
  imgs[[file]] <- img
}
num_bins <- 100

#Veamos los histogramas de rojo, verde y azul de la primera imagen
par(mfrow=c(1, 3))
img1 <- imgs[[1]]

hist_red <- hist(img1[,,1], breaks = num_bins,
                 main = "Histograma Rojo", col.main = "red",
                    font.main = 4)$counts

hist_green <- hist(img1[,,2], breaks = num_bins,
                   main = "Histograma Verde", col.main = "green",
                      font.main = 4)$counts


hist_blue <- hist(img1[,,3], breaks = num_bins,
                  main = "Histograma Azul", col.main = "blue",
                     font.main = 4)$counts



#Reescalamos las histogramas para que todas tengan el mismo tamao
    y poder trabajar con ellas

len_red <- length(hist_red)
len_green <- length(hist_green)
len_blue <- length(hist_blue)
```

```
45
46 if ((len_red !=100) | (len_green !=100) | (len_blue !=100)){
47   if (len_red > 100){
48     hist_red <- hist_red[1:100]
49   }
50   if (len_red < 100){
51     hist_red <- c(hist_red,rep(0, 100 - len_red))
52   }
53   if (len_green > 100){
54     hist_green <- hist_green[1:100]
55   }
56   if (len_green < 100){
57     hist_green <- c(hist_green,rep(0, 100 - len_green))
58   }
59   if (len_blue > 100){
60     hist_blue <- hist_blue[1:100]
61   }
62   if (len_blue < 100){
63     hist_blue <- c(hist_blue,rep(0, 100 - len_blue))
64   }
65
66 }
67 # Unir los histogramas normalizados en un vector que tendra tamao
     300
68 vec_img11 <- c(hist_red, hist_green, hist_blue)
69 M<-vec_img11
70 ##############################
71
72 for (j in 1:7999){
73   img <- imgs[[j+1]]
74
75   hist_red <- hist(img[,,1], breaks = num_bins, plot = FALSE)$counts
76   hist_green <- hist(img[,,2], breaks = num_bins, plot =
       FALSE)$counts
77   hist_blue <- hist(img[,,3], breaks = num_bins, plot =
       FALSE)$counts
78
79   len_red <- length(hist_red)
80   len_green <- length(hist_green)
81   len_blue <- length(hist_blue)
```

```
82
83    if ((len_red !=100) | (len_green !=100) | (len_blue !=100)){
84      if (len_red > 100){
85        hist_red <- hist_red[1:100]
86      }
87      if (len_red < 100){
88        hist_red <- c(hist_red,rep(0, 100 - len_red))
89      }
90      if (len_green > 100){
91        hist_green <- hist_green[1:100]
92      }
93      if (len_green < 100){
94        hist_green <- c(hist_green,rep(0, 100 - len_green))
95      }
96      if (len_blue > 100){
97        hist_blue <- hist_blue[1:100]
98      }
99      if (len_blue < 100){
100       hist_blue <- c(hist_blue,rep(0, 100 - len_blue))
101     }
102
103   }
104   vec_img <- c(hist_red, hist_green, hist_blue)
105   M<-rbind(M,vec_img)
106 }
107 ########################Inicializacion de los
      datos#################################
108
109
110 sigma=1
111
112 #Algoritmo Cavi
113 # Funcion para actualizar los parametros utilizando el algoritmo
      CAVI
114 cavi_update_params <- function(M, K, iter) {
115   #Escalamos M para no lidiar con infinitos.
116   scaledM <- scale(M)
117   # Inicializar los parametros
118   n <- nrow(scaledM)
119   D <- ncol(scaledM)
```

```r
  mk <- matrix(rnorm(K * D), ncol = D)
  sk2 <- rgamma(K,5)
  phis <- matrix(nrow = n, ncol = K)

  # Iterar para actualizar los parametros
  for (i in 1:iter) {
    for (j in 1:n) {
      for (k in 1:K) {
        phis[j, k] <- exp(t(scaledM[j, ]) %*% mk[k, ] - 0.5 * (2 *
            sk2[k] + t(mk[k, ])
                                                               %*% mk[k,
            ]))
      }
      phis[j, ] <- phis[j, ] / sum(phis[j, ])
    }

    mk_new <- matrix(rep(NA, K * D), ncol = D)
    sk2_new <- rep(NA, K)

    for (k in 1:K) {
      sk2_new[k] <- 1 / (1 / sigma^2 + sum(phis[, k]))
      mk_new[k, ] <- sk2_new[k] * colSums(phis[,k] * scaledM)
    }

    mk <- mk_new
    sk2 <- sk2_new
  }

  return(list(mk = mk, sk2 = sk2))
}



# Ejemplo de uso
# Matriz de histogramas de rojo, verde y azul de las imgenes
K <- 4
iter <- 1500

result <- cavi_update_params(M, K, iter)

```

```
mk <- result$mk
sk2 <- result$sk2

par(mfrow=c(2,2))
for (k in 1:K) {
  plot(mk[k,])
}

#########Veamos si las imagenes del conjunto Test se parecen
dir2 <- "C:/Users/Miguel/Desktop/TFG/Datos/Test/"
files2 <- list.files(dir2, full.names = TRUE)
imgs2 <- list()


for (file in files2) {
  img <- load.image(file)
  imgs2[[file]] <- img
}

M2 <- matrix(nrow = 400, ncol = 300)
for (j in 1:400){
  img <- imgs2[[j]]

  hist_red <- hist(img[,,1], breaks = num_bins, plot = FALSE)$counts
  hist_green <- hist(img[,,2], breaks = num_bins, plot =
      FALSE)$counts
  hist_blue <- hist(img[,,3], breaks = num_bins, plot =
      FALSE)$counts

  len_red <- length(hist_red)
  len_green <- length(hist_green)
  len_blue <- length(hist_blue)

  if ((len_red !=100) | (len_green !=100) | (len_blue !=100)){
    if (len_red > 100){
      hist_red <- hist_red[1:100]
    }
    if (len_red < 100){
      hist_red <- c(hist_red,rep(0, 100 - len_red))
    }
```

```
    if (len_green > 100){
      hist_green <- hist_green[1:100]
    }
    if (len_green < 100){
      hist_green <- c(hist_green,rep(0, 100 - len_green))
    }
    if (len_blue > 100){
      hist_blue <- hist_blue[1:100]
    }
    if (len_blue < 100){
      hist_blue <- c(hist_blue,rep(0, 100 - len_blue))
    }

  }
  vec_img <- c(hist_red, hist_green, hist_blue)
  M2[j, ] <- vec_img
}

#Escalamos la matriz M2 iugal que en la funcion de actualizar los
    parametros.
scaledM2<-scale(M2)

distclust1 <- rep(0,400)
distclust2 <- rep(0,400)
distclust3 <- rep(0,400)
distclust4 <- rep(0,400)

for (i in 1:400){
  distclust1[i] <- sum((mk[1,]-scaledM2[i,])^2)
  distclust2[i] <- sum((mk[2,]-scaledM2[i,])^2)
  distclust3[i] <- sum((mk[3,]-scaledM2[i,])^2)
  distclust4[i] <- sum((mk[4,]-scaledM2[i,])^2)
}

# Calculo de los indices minimos para cada fila de distancias
cluster_indices <- apply(cbind(distclust1, distclust2, distclust3,
    distclust4), 1, which.min)

# Asignacion de los datos a los clusters
grupos1 <- which(cluster_indices == 1)
```

```
grupos2 <- which(cluster_indices == 2)
grupos3 <- which(cluster_indices == 3)
grupos4 <- which(cluster_indices == 4)
grupos1;grupos2;grupos3;grupos4

resultado <- sum(grupos1>= 0 & grupos1<= 100);print(resultado)
resultado <- sum(grupos2>= 0 & grupos2<= 100);print(resultado)
resultado <- sum(grupos3>= 0 & grupos3<= 100);print(resultado)
resultado <- sum(grupos4>= 0 & grupos4<= 100);print(resultado)

resultado <- sum(grupos1>= 100 & grupos1<= 200);print(resultado)
resultado <- sum(grupos2>= 100 & grupos2<= 200);print(resultado)
resultado <- sum(grupos3>= 100 & grupos3<= 200);print(resultado)
resultado <- sum(grupos4>= 100 & grupos4<= 200);print(resultado)

resultado <- sum(grupos1>= 200 & grupos1<= 300);print(resultado)
resultado <- sum(grupos2>= 200 & grupos2<= 300);print(resultado)
resultado <- sum(grupos3>= 200 & grupos3<= 300);print(resultado)
resultado <- sum(grupos4>= 200 & grupos4<= 300);print(resultado)

resultado <- sum(grupos1>= 300 & grupos1<= 400);print(resultado)
resultado <- sum(grupos2>= 300 & grupos2<= 400);print(resultado)
resultado <- sum(grupos3>= 300 & grupos3<= 400);print(resultado)
resultado <- sum(grupos4>= 300 & grupos4<= 400);print(resultado)
```

B. Algunos resultados en teoría de la medida

Definición 5. Sean λ y μ dos medidas positivas. Se dice que λ es absolutamente continua con respecto a μ, y se escribe $\lambda \ll \mu$, si para cualquier conjunto medible E tal que $\mu(E) = 0$ se tiene que $\lambda(E) = 0$.

El Teorema de Radon-Nikodym nos indica la forma en la que se caracterizan las medidas absolutamente continuas respecto de las medidas σ finitas.

Una consecuencia del **Teorema de Lebesgue-Radon-Nikodym**(Ver Folland, Gerald B. Teorema 3.8 (1984)) es que si μ es una medida positiva σ-finita y λ una medida real σ-finita, tales que $\lambda \ll \mu$, entonces existe una función medible h con valores reales tal que

$$\lambda(A) = \int_A h d\mu$$

Cualquier otra función que verifique estas propiedades coincide con h casi seguro salvo en un conjunto de medida μ nula.
También notemos que si λ es positiva entonces $h \geq 0$ en μ casi todo punto.
La función h se llama derivada de Radon-Nikodym de λ con respecto a μ. La ecuación anterior se puede expresar como $d\lambda = hd\mu$, o de esta forma: h=$\frac{d\lambda}{d\mu}$.
Observemos que si nos encontramos en las condiciones del teorema anterior se tiene que si g es una función medible tenemos que

$$\int g d\lambda = \int g \frac{d\lambda}{d\mu} d\mu$$

En particular, el lado izquierdo es integrable si y solo si lo es el lado derecho, y en ese caso coinciden. Es decir, si g $\in L^1(\lambda)$, entonces $g\frac{d\lambda}{d\mu} \in L^1(\mu)$.
Deducimos de aquí la regla de la cadena, es decir si $\lambda \ll \mu$ y $\mu \ll \rho$ entonces $\lambda \ll \rho$ y

$$\frac{d\lambda}{d\rho} = \frac{d\lambda}{d\mu}\frac{d\mu}{d\rho}$$

Referencias

[1] Blei, D. M., Kucukelbir, A., & McAuliffe, J. D. (2017). Variational Inference: A Review for Statisticians. Journal of the American Statistical Association, 112(518), 859-877. DOI: 10.1080/01621459.2017.1285773.

[2] Folland, Gerald B. (1984). Real Analysis. Modern Techniques and Their Applications.

[3] Billingsley, Patrick (1976). Probability and measure.

[4] Bremaud, P. (1999). Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues. New York: Springer.

[5] Sun, W. (2006). Optimization theory and methods. New York: Springer.

[6] Dreizler, R. M., & Gross, E. K. U. (1990). Density Functional Theory: An Advanced Course. Berlin: Springer.

[7] Kucukelbir, A. (2017). Automatic Differentiation Variational Inference. Journal of Machine Learning Research, 18(1), 1-45.

[8] Ranganath, R., Gerrish, S., & Blei, D. (2013). Black Box Variational Inference. Retrieved from `https://arxiv.org/abs/1401.0118`

[9] Amari, S. (2010). Information geometry and its applications (Vol. 194). Berlin: Springer Science & Business Media.