Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Sistema para la obtención de tópicos mediante el modelo probabilístico Latent Dirichlet Allocation (LDA)

Tesis para obtener el grado de:

Ingeniero en Ciencias de la Computación

Presenta:

Jonathan Serrano Pérez

Asesor: Dr. Gerardo Martínez Guzmán

Co-Asesor: Dr. José Alejandro Rangel Huerta

Puebla, Pue. Agosto 2017

Agradecimientos

Agradezco a mis padres, Juan Serrano Corona y Verónica Pérez Ruíz, por el apoyo que me han brindado en todo este tiempo, ya que sin su apoyo nada de esto hubiese sido posible, con cariño muy especial, este trabajo es dedicado a ustedes.

Agradezco a mi asesor de tesis el Dr. Gerardo Martínez Guzmán, por el tiempo que me ha brindado para enseñarme nuevos temas, despejar mis dudas que en este trabajo se me fueron presentando y sobre todo por sus consejos. También agradezco a mi Co-Asesor de tesis el Dr. José Alejandro Rangel Huerta por el apoyo otorgado para la realización de este trabajo.

Y finalmente, los autores agradecen al Laboratorio Nacional de Supercómputo del Sureste de México, por los recursos computacionales, el apoyo y la asistencia técnica a través del proyecto número O-2017/037.

Resumen

Este trabajo de tesis consiste en la aplicación de un algoritmo propuesto por Blei (Blei & Jordan, 2002) para la obtención de tópicos de un corpus en español llamado Wikicorpus (Reese, Boleda, Cuadros, Padró, & Rigau, 2010) el cual consiste en aproximadamente 250000 documentos de los cuales en total suman aproximadamente 250 millones de palabras. Los tópicos nos darán una idea más o menos clara de los temas que se tratan en los documentos, trabajo realmente difícil de elaborar si no se considera la ayuda de un ordenador. Por la cantidad de información de los documentos es necesario implementar un algoritmo de manera que algunas partes del programa se puedan llevar a cabo en programación en paralelo, ya que la cantidad de información es demasiado grande y cualquier equipo con un programa secuencial se tardaría mucho tiempo en procesar la información. Por lo que es necesario implementar un programa en paralelo y en un lenguaje de programación que mejore el tiempo de ejecución. El programa en paralelo realmente demostraría su eficacia si se corre en un equipo de supercómputo, por lo cual se implementó un proyecto en el Laboratorio Nacional de Supercómputo del Sureste de México (LNS), el cual fue aprobado y a partir del cual se llevó a cabo dicha tarea. El sistema ha sido programado con el Lenguaje C++ y con la API de OpenMP, donde en cada etapa paralela son creados 24 hilos, con la intención de que cada hilo sea ejecutado por un core diferente. Gracias a este proyecto, el sistema LDA ha sido ejecutado satisfactoriamente en los nodos normales, donde cada nodo tiene 2 procesadores Intel Xeon E5-2680 v3, y cada procesador tiene 12 cores con una frecuencia de 2.5 GHz, además cuentan con 126 Gigabytes de memoria RAM.

Introducción

Este trabajo de tesis se desarrolla a lo largo de cuatro capítulos, en el primer capítulo se introducen los conceptos fundamentales de la Teoría Bayesiana de tal forma que se entienda que el razonamiento de la inferencia bayesiana, es radicalmente diferente a la inferencia clásica o frecuentista, que desdeña en lo formal toda información previa de la realidad que se examina. La inferencia bayesiana es muy cercana al modo de proceder cotidiano, e inductivo, ya que cualquier información empírica, combinada con el conocimiento que ya se tenga del problema que se estudia, "actualiza" dicho conocimiento.

En el segundo capítulo se introducen algoritmos para realizar algunas simulaciones de distribuciones de probabilidad. Un conocimiento de las matemáticas del modelado así como también sus bases teóricas y la práctica de la programación de computadoras son esenciales para llevar a cabo una correcta aplicación de la simulación en áreas de la ciencia. Hoy en día, la modelación y la simulación son esenciales en el desarrollo de procesos y es conveniente cuando no existe una formulación matemática analíticamente resoluble o existe una formulación matemática, pero es difícil obtener una solución analítica.

En el tercer capítulo se desarrolla un modelo de tópicos que se basa en la idea de que los documentos son mezclas de tópicos. Los modelos de tópicos extraen conocimiento solo de los datos. Una de las aplicaciones específica más útiles comprende la extracción de temáticas a partir de cuerpos de documentos cuya envergadura vence nuestras competencias para obtener manualmente temas. Esta metodología parte de los mismos datos para obtener los tópicos en los que luego serán agrupados los documentos. Para llevar a cabo esta tarea se selecciona automáticamente palabras del corpus y en base a su frecuencia en los documentos indicaría que podría pertenecer o no a cierto tema, lo que clasificaría las palabras sin intervención humana. Uno de los modelos más usados de modelación de tópicos es el Latent Dirichlet Allocation (LDA).

Finalmente en el cuarto capítulo se describe el desarrollo del sistema, el cual se llevó a cabo en programación paralela. La Computación Paralela consiste en la explotación de varios procesadores para que trabajen de forma conjunta en la resolución de un problema computacional. Normalmente cada procesador trabaja en una parte del problema y se realiza intercambio de datos entre los procesadores. Según cómo se realice este intercambio podemos tener modelos distintos de programación paralela. La necesidad de la computación paralela se origina por las limitaciones de los computadores secuenciales: integrando varios procesadores para llevar a cabo la computación es posible resolver problemas que requieren de más memoria o de mayor velocidad de cómputo. Hay dificultades lógicas, como la de desarrollar compiladores y entornos de programación eficientes para los sistemas paralelos, que son más complejos que los secuenciales. La computación paralela se utiliza para reducir el tiempo de resolución de problemas computacionales, o bien para resolver problemas grandes que no cabrían en la memoria de un procesador secuencial.

Contenido

Capítulo I Introducción a la Teoría Bayesiana

Distribución Bernoulli

Iniciaremos suponiendo que se lanza una moneda al aire. Entonces, definimos la variable aleatoria X como:

$$X = \begin{cases} 1 \; si \; cae \; cara \\ 0 \; si \; cae \; cruz. \end{cases}$$

Si $P(X = 1) = \theta$, la función de densidad de X es:

$$f(x;\theta) = \theta^x(1-\theta)^{1-x} \qquad x = 0{,}1$$

llamada función de densidad de Bernoulli, con esperanza y varianza:

$$E(X) = \theta.$$

$$Var(X) = \theta(1-\theta).$$

Si lanzamos la moneda n-veces, entonces obtenemos una sucesión de valores consistiendo de ceros y unos.

$$D = (x_1, x_2, \dots, x_n) \qquad x_i = 0 \text{ ó } 1.$$

La función de verosimilitud de n-variables aleatorias $X_1, X_2, \dots, X_n$ es la densidad conjunta de las n-variables, $g = (x_1, x_2, \dots, x_n; \theta)$ la cual es función de θ. En particular si $x_1, x_2, \dots, x_n$ es una muestra aleatoria de la densidad $f(x;\theta)$, la función de verosimilitud es:

$$g(x_1, x_2, \dots, x_n; \theta) = f(x_1;\theta)f(x_2;\theta) \dots f(x_n;\theta),$$

que es comúnmente denotada por:

$$L(\theta) = f(x_1;\theta)f(x_2;\theta) \dots f(x_n;\theta).$$

Ahora sí, $x_1, x_2, \dots, x_n$ son los valores particulares observados. Se desea saber de qué densidad es más probable que proceda este conjunto de valores. Cuando θ toma diferentes valores queda definida una familia de densidades, y queremos saber que densidad da la mayor posibilidad de obtener el conjunto $x_1, x_2, \dots, x_n$, es decir, queremos hallar el valor de θ, que haga máxima la función de verosimilitud $L(\theta)$.

Si $\hat{\theta}$ es el valor de θ que hace máxima a $L(\theta)$, entonces $\hat{\theta} = d(x_1, x_2, \dots, x_n)$ es el estimador de máxima verosimilitud de θ.

Frecuentemente, la función de verosimilitud satisface condiciones de regularidad, de modo que el valor máximo se obtiene igualando a cero las derivadas de la función de verosimilitud y resolviendo el sistema de ecuaciones respecto a los parámetros.

También $L(\theta)$ y $\log(L(\theta))$ toman su máximo para el mismo valor de θ por lo que algunas veces resulta más fácil hallar el máximo del logaritmo de la verosimilitud.

Por ejemplo, si la moneda se lanza n-veces, entonces la función de verosimilitud de la muestra $(x_1, x_2, \dots, x_n)$ es:

$$L(\theta) = \theta^{x_1}(1-\theta)^{1-x_1}\ \theta^{x_2}(1-\theta)^{1-x_2}\ \dots\ \theta^{x_n}(1-\theta)^{1-x_n}$$

$$L(\theta) = \theta^{\sum_{i=1}^{n} x_i}(1-\theta)^{n-\sum_{i=1}^{n} x_i}$$

$$\log\big(L(\theta)\big) = \left(\sum_{i=1}^{n} x_i\right)\log(\theta) + \left(n - \sum_{i=1}^{n} x_i\right)\log(1-\theta)$$

$$\frac{d\ \log\big(L(\theta)\big)}{d\theta} = \frac{\sum_{i=1}^{n} x_i}{\theta} - \frac{n - \sum_{i=1}^{n} x_i}{1-\theta} = 0$$

$$\frac{\sum_{i=1}^{n} x_i}{\theta} = \frac{n - \sum_{i=1}^{n} x_i}{1-\theta}$$

$$\sum_{i=1}^{n} x_i = n\theta$$

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

Lo cual prueba que el Estimador de Máxima Verosimilitud es

$$\hat{\theta} = \frac{\sum_{i=1}^{n} x_i}{n} = \bar{x},$$

que coincide con la media muestral.

Multinomial

Ahora considere una variable aleatoria X con k estados $\{1,2,\dots,k\}$ y sea $P(X=j) = \theta_j$ entonces la función de densidad de X es

$$f(x;\theta) = \theta_1^{I(x=1)}\theta_2^{I(x=2)}\dots\theta_k^{I(x=k)} \qquad con \qquad \sum_{i=1}^{k} \theta_i = 1,$$

Donde la función Indicadora se define como

$$I(x=j) = \begin{cases} 1\ si\ x = j \\ 0\ si\ x \neq j \end{cases},$$

la densidad de X es llamada la densidad Multinomial.

Para encontrar los Estimadores de Máxima Verosimilitud, se procede a repite el experimento N veces independientes, obteniendo la muestra de tamaño N:

$$D = \{x_1, x_2, \dots, x_N\},$$

donde cada x_i puede tomar cualquiera de los k estados.

La función de verosimilitud tiene la forma:

$$L(\theta) = \left(\theta_1^{I(x_1=1)} \dots \theta_k^{I(x_1=k)}\right) \dots \left(\theta_1^{I(x_N=1)} \dots \theta_k^{I(x_N=k)}\right).$$

La cual se puede escribir como:

$$L(\theta) = \theta_1^{N_1}\theta_2^{N_2} \dots \theta_k^{N_k},$$

donde $N_j = \sum_{k=1}^{N} I(x_k = j)$.

Aplicando logaritmo a la verosimilitud de los datos obtenemos la ecuación:

$$log\, L(\theta) = \sum_{i=1}^{k} N_i log\theta_i,$$

que está sujeta a la restricción:

$$\sum_{i=1}^{k} \theta_i = 1.$$

Aplicando los multiplicadores de Lagrange, la función con restricción es

$$l = \sum_{i=1}^{k} N_i log\theta_i + \lambda\left(1 - \sum_{i=1}^{k} \theta_i\right).$$

La derivada respecto de θ_k produce

$$\frac{\partial l}{\partial \theta_k} = \frac{N_k}{\theta_k} - \lambda = 0.$$

La derivada respecto de λ

$$\frac{\partial l}{\partial \lambda} = \left(1 - \sum_{i=1}^{k} \theta_i\right) = 0.$$

De las dos últimas igualdades obtenemos

$$\mathrm{N}_k = \lambda\theta_k$$

$$\sum_{i=1}^{k} N_i = \lambda \sum_{i=1}^{k} \theta_i\,.$$

De aquí, los estimadores de máxima verosimilitud de una multinomial son:

$$\hat{\theta}_1 = \frac{N_1}{N}, \hat{\theta}_2 = \frac{N_2}{N}, \dots, \hat{\theta}_k = \frac{N_k}{N}.$$

Funciones de densidad Poisson y Normal

Asimismo, se pueden demostrar que los estimadores de máxima verosimilitud para las densidades de Poisson, Normal y Beta son los siguientes.

Densidad de Poisson:

$$f(x;\lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \qquad x = 0, 1, 2, \ldots$$

Estimador de Máxima Verosimilitud $\hat{\lambda} = \bar{x}$

Densidad Normal:

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}.$$

Estimadores de Máxima Verosimilitud

$$\hat{\mu} = \bar{x}$$

$$\hat{\sigma}^2 = \frac{\sum(x_i - \bar{x})^2}{n}.$$

La Densidad Beta

La densidad beta se define en términos de la función gamma la cual es definida como:

$$\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} du \qquad x > 0$$

y cumple las igualdades siguientes

$$\Gamma(x+1) = x\Gamma(x), \;\; \Gamma(1) = 1, \;\; \Gamma(x+1) = x!.$$

Una variable aleatoria X tiene una densidad beta si su función de densidad está dada por:

$$f(x;\alpha_1,\alpha_0) = \frac{\Gamma(\alpha_1+\alpha_0)}{\Gamma(\alpha_1)\Gamma(\alpha_0)} x^{\alpha_1-1}(1-x)^{\alpha_0-1} \qquad 0 < x < 1.$$

La media, varianza y moda de una densidad beta son:

$$media = \frac{\alpha_1}{\alpha_1 + \alpha_0}$$

$$varianza = \frac{\alpha_1 \alpha_0}{(\alpha_1 + \alpha_0)^2(\alpha_1 + \alpha_0 - 1)}$$

$$moda = \frac{\alpha_1 - 1}{\alpha_1 + \alpha_0 - 2}.$$

Estimador de Bayes

Se han considerado densidades de la forma $f(x;\theta)$ donde X es una variable aleatoria y θ es un parámetro desconocido. Pero resulta que en situaciones reales existe información adicional sobre θ. Así por ejemplo, el experimentador puede tener la evidencia de que θ se comporta como una variable aleatoria que tiene una función de densidad. Cuando este es el caso, la notación $f(x;\theta)$ para representar la densidad de una variable aleatoria X con parámetro θ, se cambia por la notación $f(x|\theta)$ cuando el parámetro también es una variable aleatoria.

Sea $x_1, x_2, \dots, x_n$ una muestra aleatoria de tamaño n de la densidad $f(x|\theta)$. Se desea estimar que valor de θ es el que determina la densidad de la que procede la muestra aleatoria. Como estamos suponiendo que θ también es una variable aleatoria, entonces suponemos que tiene función de densidad $p(\theta)$.

Para evaluar una función de decisión $\hat{\theta}$ y seleccionar las adecuadas se introduce una función de perdida $l(\hat{\theta};\theta)$ esta es una función real no negativa que refleja la perdida al tomar la acción $\hat{\theta}$ cuando θ es el parámetro. La perdida depende de las observaciones particulares x_i por lo que es más conveniente la perdida promedio, es decir, el valor esperado de la perdida llamado el riesgo y denotado por $R(\hat{\theta};\theta)$.

Por lo tanto, si la función de densidad de las observaciones es:

$$g(x_1, x_2, \dots, x_n|\theta) = f(x_1;\theta)f(x_2;\theta)\dots f(x_n;\theta),$$

entonces:

$$R(\hat{\theta};\theta) = E\left(l(\hat{\theta};\theta)\right) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty} l(\hat{\theta};\theta)g(x_1, x_2, \dots, x_n|\theta)\, dx_1\, dx_2 \dots dx_n.$$

Una buena función de decisión es toda aquella que hace mínimo el riesgo para cada valor de θ.

El riesgo $R(\hat{\theta};\theta)$ depende de θ, y como θ es una variable aleatoria entonces nuevamente es más conveniente tomar el valor promedio de $R(\hat{\theta};\theta)$, es decir, nos interesa determinar $\hat{\theta}$ que hace mínimo a $E(R(\hat{\theta};\theta))$.

Por lo tanto:

$$E\left(R\left(\hat{\theta};\theta\right)\right) = \int_{-\infty}^{\infty} R\left(\hat{\theta};\theta\right)P(\theta)d\theta = \int_{-\infty}^{\infty} E\left(l\left(\hat{\theta};\theta\right)\right)P(\theta)d\theta$$

$$= \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty} l\left(\hat{\theta};\theta\right)g(x_1,\dots,x_n|\theta)dx_1\dots dx_n\right]P(\theta)d\theta.$$

El estimador $\hat{\theta}$ (que depende de las x_i) que haga mínimo a $E\left(R\left(\hat{\theta};\theta\right)\right)$ recibe el nombre de estimador de Bayes.

Haciendo un cambio de integrales obtenemos:

$$\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} l\left(\hat{\theta};\theta\right)g(x_1,\dots,x_n|\theta)P(\theta)d\theta\right]dx_1\dots dx_n.$$

Entonces $E\left(R\left(\hat{\theta};\theta\right)\right)$ se hará mínima si es posible hallar una función $\hat{\theta}$ de las x_i que haga mínima la cantidad:

$$\int_{-\infty}^{\infty} l\left(\hat{\theta};\theta\right)g(x_1,\dots,x_n|\theta)P(\theta)d\theta.$$

Obviamente la cantidad $g(x_1,x_2,\dots,x_n|\theta)P(\theta)$ es la densidad conjunta de la variable aleatoria $X_1,X_2,\dots,X_n,\theta$, la cual describiremos como:

$$q(x_1,\dots,x_n,\theta) = g(x_1,\dots,x_n|\theta)P(\theta).$$

Entonces la densidad marginal de las X_i esta dado por:

$$k(x_1,\dots,x_n) = \int_{-\infty}^{\infty} q(x_1,\dots,x_n,\theta)d\theta,$$

y la densidad condicional de θ dados los valores $x_1,x_2,\dots,x_n$ es:

$$h(\theta|x_1,\dots,x_n) = \frac{q(x_1,\dots,x_n,\theta)}{k(x_1,\dots,x_n)},$$

la función $h(\theta|x_1,\dots,x_n)$ es llamada la densidad a *posteriori.*

De lo anterior tenemos

$$\int_{-\infty}^{\infty} l\left(\hat{\theta};\theta\right)g(x_1,\dots,x_n|\theta)P(\theta)d\theta =$$

$$\int_{-\infty}^{\infty} l\left(\hat{\theta};\theta\right)k(x_1,\dots,x_n)h(\theta|x_1,\dots,x_n)d\theta$$

$$= k(x_1, \dots, x_n) \int_{-\infty}^{\infty} l(\hat{\theta}; \theta) h(\theta | x_1, \dots, x_n) d\theta.$$

Luego un estimador de Bayes es el valor de $\hat{\theta}$ que, para cada muestra posible $x_1, x_2, \dots, x_n$ hace mínima la cantidad:

$$v(\hat{\theta}; x_1, \dots, x_n) = \int_{-\infty}^{\infty} l(\hat{\theta}; \theta) h(\theta | x_1, \dots, x_n) d\theta.$$

La función $v(\hat{\theta}; x_1, x_2, \dots, x_n)$ representa el riesgo posterior para estimar θ.

Si la pérdida es $l(\hat{\theta}; \theta) = (\hat{\theta} - \theta)^2$ llamado el error cuadrático medio, entonces el estimador de Bayes está dado por $\hat{\theta} = E(\theta | x_1, \dots, x_n)$.

En efecto,

$$v(\hat{\theta}; x_1, \dots, x_n) = \int_{-\infty}^{\infty} (\hat{\theta} - \theta)^2 h(\theta | x_1, \dots, x_n) d\theta$$

$$= \hat{\theta}^2 - 2\hat{\theta} E(\theta | x_1, \dots, x_n) + E(\theta^2 | x_1, \dots, x_n).$$

Derivando respecto de $\hat{\theta}$ e igualando a cero tenemos

$$2\hat{\theta} - 2E(\theta | x_1, \dots, x_n) = 0.$$

Entonces

$$\hat{\theta} = E(\theta | x_1, \dots, x_n),$$

que es un mínimo ya que la segunda derivada es $2 > 0$.

Si la pérdida es el error cuadrático medio, entonces el estimador de Bayes se puede escribir como:

$$\hat{\theta} = E(\theta | x_1, \dots, x_n) = \int_{-\infty}^{\infty} \theta h(\theta | x_1, \dots, x_n) d\theta$$

$$\hat{\theta} = E(\theta | D) = \int_{-\infty}^{\infty} \theta h(\theta | D) d\theta,$$

donde $D = \{x_1, \dots, x_n\}$ son los datos observados.

Regla de Bayes

La actualización de nuestra creencia acerca del valor del parámetro θ después de haber visto los datos:

$$D = \{x_1, x_2, \dots, x_n\},$$

puede ser calculada usando la regla de Bayes:

$$P(\theta|D) = \frac{P(\theta)P(D|\theta)}{P(D)},$$

donde $P(D|\theta)$ es llamado la verosimilitud de los datos observados dado el parámetro θ, $P(\theta)$ representa nuestra creencia prior y $P(\theta|D)$ viene siendo nuestra creencia posterior.

$P(D)$ es la constante de normalización, la cual es llamada la verosimilitud marginal, la cual puede calcularse mediante

$$P(D) = \int P(D, \theta) d\theta.$$

Prior Conjugado

Un prior es llamado conjugado si cuando lo multiplicamos por la verosimilitud $P(D|\theta)$ el resultado posterior está en la misma familia paramétrica que el prior (Decimos que el modelo es cerrado bajo actualización Bayesiana).

Veamos un ejemplo del prior conjugado. Para esto consideramos nuevamente el ejemplo de la moneda, donde θ era la probabilidad de obtener cara y $P(\theta)$ era la función de densidad de θ.

Ya que la verosimilitud de los datos tiene la forma:

$$P(D|\theta) = \theta^{N_1}(1-\theta)^{N_0}$$

donde N_1 es el número de caras y N_0 el número de cruces y

$$N = N_1 + N_0.$$

Entonces se quiere un prior de la forma:

$$P(\theta) \propto \theta^{\alpha_1 - 1}(1-\theta)^{\alpha_0 - 1},$$

el símbolo $\propto$ representa proporcionalidad. De esta manera, la posterior se puede obtener por adición de exponentes:

$$P(\theta|D) \propto P(D|\theta)P(\theta)$$

$$P(\theta|D) \propto \theta^{N_1}(1-\theta)^{N_0}[\theta^{\alpha_1 - 1}(1-\theta)^{\alpha_0 - 1}]$$

$$P(\theta|D) \propto \theta^{N_1 + \alpha_1 - 1}(1-\theta)^{N_0 + \alpha_0 - 1},$$

donde α_1, α_0 son los parámetros del prior (hiperparámetros) y corresponden al número virtual de caras y cruces. El número $\alpha = \alpha_1 + \alpha_0$ es llamado la fuerza del prior.

Formalmente, el siguiente prior es llamado un prior Beta:

$$P(\theta|\alpha_1,\alpha_0) = Beta(\theta;\alpha_1,\alpha_0) = \frac{\Gamma(\alpha_1+\alpha_0)}{\Gamma(\alpha_1)\Gamma(\alpha_0)}\theta^{\alpha_1-1}(1-\theta)^{\alpha_0-1}.$$

Por lo que la creencia a posterior es:

$$P(\theta|\alpha_1,\alpha_0,N_1,N_0) = Beta(\theta|N_1+\alpha_1,N_0+\alpha_0).$$

$$Beta(\theta|N_1+\alpha_1,N_0+\alpha_0) = \frac{\Gamma(N_1+\alpha_1+N_0+\alpha_0)}{\Gamma(N_1+\alpha_1)\Gamma(N_0+\alpha_0)}\theta^{N_1+\alpha_1-1}(1-\theta)^{N_0+\alpha_0-1}. \quad (1)$$

Si empezamos con el prior beta $Beta(\theta|\alpha_1,\alpha_0)$ y sean N_1 número de caras y N_0 número de cruces, terminamos con un beta posterior igual al de la ecuación (1). Por ejemplo, si se comienza con un $Beta(\theta|\alpha_1=2,\alpha_0=2)$ y se observa $x=1$, donde $N_1=1$ y $N_0=0$, por lo tanto la posterior es $Beta(\theta|\alpha_1=3,\alpha_0=2)$. Por lo tanto los cambios del estimador de máxima verosimilitud (EMV) al estimador de Bayes son:

$$E(\theta) = {}^{1}/_{2} \qquad EMV.$$

$$E(\theta|D) = {}^{3}/_{5} \qquad Estimador\ de\ Bayes.$$

Además, podemos realizar actualizaciones secuencialmente, esto es supongamos que comenzamos con un prior beta $Beta(\theta|\alpha_1,\alpha_0)$, si observamos N tiradas con N_1 caras y N_0 cruces, entonces la posterior es:

$$P(\theta|\alpha_1,\alpha_0,N_1,N_0) = Beta(\theta|N_1+\alpha_1,N_0+\alpha_0) = Beta(\theta;\alpha'_1,\alpha'_0),$$

donde ahora α'_1,α'_0 son los parámetros del nuevo prior. Si observamos ahora N' tiradas con N'_1 caras y N'_0 cruces, entonces la posterior es:

$$P(\theta|\alpha'_1,\alpha'_0,N'_1,N'_0) = Beta(\theta|N'_1+\alpha'_1,N'_0+\alpha'_0)$$
$$= Beta\big(\theta|,N'_1+N_1+\alpha_1,N'_0+N_0+\alpha_0\big).$$

Esto es útil para el aprendizaje en línea y para el procesamiento de un conjunto de datos grande, ya que no se necesita almacenar los datos originales.

Como el estimador puntual de Bayes está dado por:

$$\hat{\theta} = P(x=1|D) = \int_0^1 \theta P(\theta|D)d\theta = E(\theta|D),$$

y sabemos que:

$$P(\theta|D) = Beta(\theta|N_1+\alpha_1,N_0+\alpha_0),$$

entonces

$$\hat{\theta} = E(\theta|D) = E\big(Beta(\theta|N_1+\alpha_1,N_0+\alpha_0)\big) = \frac{N_1+\alpha_1}{N_1+\alpha_1+N_0+\alpha_0}. \quad (1)$$

Por ejemplo, con un prior uniforme $\alpha_1 = \alpha_0 = 1$ tenemos:

$$\hat{\theta} = E(\theta|D) = \frac{N_1 + 1}{N_1 + N_0 + 2}.$$

Lo cual evita la escasez de datos. En estadística Bayesiana un estimador de *Probabilidad Posterior Máxima (MAP por sus siglas en inglés)* es la moda de la distribución posterior. El MAP puede ser usado para obtener una estimación puntual de una cantidad no observada en la base de datos empíricos.

Fuerza del Prior

Sea $N = N_1 + N_0$ el número de observaciones y sea N' el número de seudo-observaciones (Fuerza del Prior) y definimos:

$$\alpha_1 = \mathrm{N}'\alpha'_1 \quad (3)$$

$$\alpha_0 = \mathrm{N}'\alpha'_0 \quad (4)$$

$$\alpha'_1 + \alpha'_0 = 1$$

$$0 < \alpha'_1, \alpha'_0 < 1.$$

Ahora, de la ecuación (2) tenemos:

$$\hat{\theta} = E(\theta|D) = \frac{N_1 + \alpha_1}{N_1 + \alpha_1 + N_0 + \alpha_0}. \quad (5)$$

Sustituyendo (3) y (4) en la ecuación (5) obtenemos:

$$\frac{N_1 + \alpha_1}{N_1 + \alpha_1 + N_0 + \alpha_0} = \frac{N'\alpha'_1 + N_1}{N + N'}$$

$$= \frac{N'}{N + N'}\alpha'_1 + \frac{N}{N + N'}\frac{N_1}{N} = \frac{N'}{N + N'}\alpha'_1 + \left(1 - \frac{N'}{N + N'}\right)\frac{N_1}{N}$$

$$= \lambda\alpha'_1 + (1 - \lambda)\frac{N_1}{N},$$

donde

$$\lambda = \frac{N'}{N + N'}.$$

Considerando nuestro ejemplo de la moneda, si supone que tenemos un prior uniforme $\alpha'_1 = \alpha'_0 = 0.5$ y observamos $N_1 = 3,\ N_0 = 7$. Si consideramos un prior débil $N' = 2$ entonces el estimador de Bayes usando la ecuación (5) es:

$$\hat{\theta} = \frac{1 + 3}{1 + 3 + 1 + 7} = \frac{1}{3} \cong 0.333$$

Ahora si usamos un prior fuerte $N' = 20$ tenemos

$$\hat{\theta} = \frac{10 + 3}{10 + 3 + 10 + 7} = \frac{13}{30} \cong 0.433 \quad (6)$$

Sin embargo, el Estimador de Máxima Verosimilitud es:

$$\hat{\theta} = \frac{3}{10} = 0.3 \quad (7)$$

Se puede notar claramente que el estimador de la ecuación (6) se aleja del valor del EMV (7) ya que el prior es fuerte, pero los datos $N_1 = 3,\ \ N_0 = 7$ no concuerda con este prior. Así pues, con un prior fuerte los datos se mueven menos lejos de la media del prior, que en este caso es $\frac{10}{20} = 0.5$.

Ahora imaginemos que estamos realizando actualizaciones secuenciales de nuestro estimador. El EMV puede cambiar drásticamente con pequeños tamaños de muestra. Sin embargo, el estimador de Bayes cambia mucho más suave, claro está, que depende de la fuerza del prior.

Si observamos $N_1 = 300$, $N_0 = 700$ entonces, no importa si nuestra fuerza prior es N'=20 ya que por el estimador de Bayes obtenemos:

$$\hat{\theta} = \frac{1 + 300}{2 + 1000} \cong \frac{10 + 300}{1000 + 20} \cong 0.3$$

Y el EMV es:

$$\hat{\theta} = \frac{3}{10} = 0.3$$

Por lo que:

$$\hat{\theta} = E(\theta|D) \to \hat{\theta}(EMV).$$

Si $N_1 = 300$ y $N_0 = 700$, que es una muestra grande, entonces la fuerza del prior deja de importar. Pero si la muestra es pequeña se debe tener cuidado con la fuerza de prior.

La Densidad de Dirichlet

El prior conjugado es llamado Dirichlet y tiene parámetros α_j con $j = 1,2, \dots, k$ si

$$P(\theta|\alpha) = \frac{\Gamma\left(\sum_{j=1}^{k} \alpha_j\right)}{\prod_{j=1}^{k} \Gamma(\alpha_j)} \theta_1^{\alpha_1 - 1} \dots \theta_k^{\alpha_k - 1}.$$

Si $\alpha = \sum_{j=1}^{k} \alpha_j$ es la fuerza total del prior, entonces se puede demostrar que:

$$E(\theta_k) = \frac{\alpha_k}{\alpha}$$

$$Var(\theta_k) = \frac{\alpha_k(\alpha - \alpha_k)}{\alpha^2(\alpha + 1)}$$

$$moda(\theta_k) = \frac{\alpha_k - 1}{\alpha - k}.$$

Un prior de Dirichlet es análogo al caso beta-Bernoulli.

Es común usar el prior $\alpha = (\alpha', \alpha', \dots, \alpha')$ donde $k\alpha'$ es la fuerza del prior. Con frecuencia tomamos $\alpha' = 1/k$ o $\alpha' = 1$, llamados prior Jeffrey's y prior uniforme respectivamente.

Prior:

$$P(\theta|\alpha) = \frac{\Gamma(\alpha_1 + \cdots + \alpha_k)}{\Gamma(\alpha_1) \dots \Gamma(\alpha_k)} \theta_1^{\alpha_1 - 1} \dots \theta_k^{\alpha_k - 1}.$$

Verosimilitud de los datos:

$$P(D|\theta) = \prod_{j=1}^{k} \theta_j^{N_j}.$$

Posterior:

$$P(\theta|D, \alpha) = Dir(\alpha_1 + N_1, \dots, \alpha_k + N_k).$$

Predicción posterior:

$$P(\theta_j|D, \alpha) = \frac{\alpha_j + N_j}{N + \sum \alpha_k}.$$

Veamos un ejemplo. El color de ojos de la población, para las personas que tienen el cabello negro, se divide en cuatro categorías c_1 –cafés, c_2 –azules, c_3 –avellanas, c_4 - verdes.

Con el fin de utilizar un prior de Dirichlet se debe especificar $\alpha_1, \alpha_2, \alpha_3, \alpha_4$. La forma más fácil de hacerlo es aprovechar el hecho de que:

$$E(\theta_i) = \frac{\alpha_i}{\alpha}$$

donde α sería igual a:

$$\alpha = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4$$

y $E(\theta_i)$ es la esperanza (prior) para la proporción θ_i de la población en la categoría c_i . Basado en un conocimiento limitado se especifica que:

$$E(\theta_1) = 0.5, \quad E(\theta_2) = 0.3, \quad E(\theta_3) = E(\theta_4) = 0.1$$

Además, seleccionamos $\alpha = 30$.

Ahora se obtiene una muestra de tamaño $N = 108$ donde:

$$n_1 = 68, \ n_2 = 20, \ n_3 = 15, \ n_4 = 5$$

Y los Estimadores de Máxima verosimilitud son:

$$\hat{\theta}_1 = \frac{68}{108} = 0.63, \qquad \hat{\theta}_2 = \frac{20}{108} = 0.185, \ \hat{\theta}_3 = \frac{15}{108} = 0.139, \qquad \hat{\theta}_4 = \frac{5}{108} = 0.046$$

De estos datos obtenemos:

$$E(\theta_i|D) = w_1 E(\theta_i) + w_2 \hat{\theta}_i$$

donde:

$$w_1 = \frac{\alpha}{\alpha + N} \qquad w_2 = 1 - w_1$$

Entonces la esperanza posterior:

$E(\theta_1|D) = w_1(0.5) + w_2(0.630) = 0.601$

$E(\theta_2|D) = w_1(0.3) + w_2(0.185) = 0.210$

$E(\theta_3|D) = w_1(0.1) + w_2(139) = 0.130$

$E(\theta_4|D) = w_1(0.1) + w_2(0.046) = 0.058$

Sin embargo, podemos mirar algo más que una media. La distribución posterior de θ_1, θ_2, θ_3, θ_4 es:

$$Dirichlet(0.5\alpha + 68, 0.3\alpha + 20, 0.1\alpha + 15, 0.1\alpha + 5).$$

De modo que cada θ_i tiene una distribución beta posterior. Por ejemplo, la distribución posterior de θ_1 es:

$\beta\big(0.5\alpha + 68, (0.3 + 0.1 + 0.1)\alpha + (20 + 15 + 5)\big) = \beta(0.5\alpha + 68, 0.5\alpha + 40)$.

Capitulo 2 Simulación de Muestras Aleatorias

Supongamos que $X_1, X, \dots, X_n$ es una sucesión de variables aleatorias independientes y cada una de ellas con la misma μ. Para un valor de n, consideramos la variable aleatoria M_n la media de la muestra definida como

$$M_n = \frac{X_1 + X_2 + \cdots + X_n}{n}.$$

Cuando n es grande que podemos decir acerca de M_n.

Ley débil de los grandes números

Sea $X_1, X_2, \dots, X_n$ una sucesión de variables aleatorias independientes, cada una de ellas con la misma media μ y con la misma varianza σ^2 . Entonces para todo $\varepsilon > 0$ se tiene:

$$\lim_{n\to\infty} P(|M_n - \mu| \geq \varepsilon) = 0.$$

Se dice que las medias muestrales convergen en probabilidad a la media común μ.

$$M_n \xrightarrow{P} \mu.$$

Ley fuerte de los grandes números

Sea $X_1, X_2, \dots, X_n$ una sucesión de variables aleatorias independientes, cada una de ellas con la misma media finita μ . Entonces:

$$P\left(\lim_{n\to\infty} M_n = \mu\right) = 1.$$

Se dice entonces que las medias convergen con probabilidad uno a μ.

$$M_n \xrightarrow{c \cdot s} \mu.$$

Teorema Central del Límite

Sea $X_1, X_2, \dots, X_n$ una sucesión de variables aleatorias independientes, con media μ y varianza finita σ^2, entonces:

$$Z_n = \frac{M_n - \mu}{\frac{\sigma}{\sqrt{n}}} \xrightarrow{D} N(0,1),$$

la variable aleatoria Z_n converge en distribución a la normal estándar.

Para cada valor prefijado de $x \in \mathbb{R}$, tenemos

$$\lim_{n\to\infty} P(Z_n \leq x) = \Phi(x),$$

donde Φ es la función de distribución acumulada de la densidad normal estándar.

Así que, para cada $x \in \mathbb{R}$:

$$\lim_{n\to\infty} P\left(M_n \leq \mu + x\frac{\sigma}{\sqrt{n}}\right) = \Phi(x).$$

Entonces

$$\lim_{n\to\infty} P\left(-3 < \frac{M_n - \mu}{\sigma/\sqrt{n}} < 3\right) = \Phi(3) - \Phi(-3) = 0.9974$$

$$\lim_{n\to\infty} P\left(M_n - \frac{3\sigma}{\sqrt{n}} < \mu < M_n + \frac{3\sigma}{\sqrt{n}}\right) = 0.9974$$

De aquí el intervalo:

$$\left(M_n - \frac{3\sigma}{\sqrt{n}}\ ,\ M_n + \frac{3\sigma}{\sqrt{n}}\right)$$

Contiene el valor desconocido de μ con un determinado nivel de certeza (probabilidad del 0.9974). Siendo la longitud de este intervalo:

$$\left(M_n + \frac{3\sigma}{\sqrt{n}}\right) - \left(M_n - \frac{3\sigma}{\sqrt{n}}\right) = 2\left(\frac{3\sigma}{\sqrt{n}}\right),$$

la mitad de este intervalo, $3\sigma/\sqrt{n}$, nos da una valoración del error de la aproximación M_n . Observe que:

$$var(M_n) = \frac{\sigma^2}{n}.$$

De forma que dicha mitad del intervalo es igual a tres veces la desviación estándar de M_n . Para n grande el intervalo se puede aproximar por el intervalo:

$$\left(M_n - \frac{3\sigma}{\sqrt{n}}\ ,\ M_n + \frac{3\sigma}{\sqrt{n}}\right) \cong \left(M_n - \frac{3s}{\sqrt{n}}\ ,\ M_n + \frac{3s}{\sqrt{n}}\right)$$

donde:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=0}^{n}(x_i - \bar{x})^2}\ \ ,$$

este intervalo contiene el verdadero valor de μ con una probabilidad del 0.9974, además la mitad del intervalo es una medida del error de la estimación M_n. A la cantidad $s/\sqrt{n}$ se le denomina error estándar de la estimación M_n.

Algoritmo adecuado para generar variables independientes

La función de distribución inversa F^{-1} de una función de densidad permite generar un conjunto de observaciones de la función de densidad continua. Primero usamos un generador de números aleatorios para obtener observaciones independientes $u_1, u_2, \ldots, u_n$ de $U(0,1)$. Después transformamos estas observaciones mediante la inversa de la función de distribución para obtener los correspondientes cuantiles de F,

$$x_i = F^{-1}(u_i) \qquad i = 1,2,\dots,n.$$

Entonces $x_1, x_2, \dots, x_n$ son observaciones independientes de la función de densidad cuya función de distribución es F .

La distribución exponencial de media θ

La función de densidad exponencial con media θ tiene la forma

$$f(x) = \frac{1}{\theta} e^{-x/\theta} \qquad x > 0.$$

Y su función de distribución es

$$F(x) = \int_0^x \frac{1}{\theta} e^{-t/\theta} dt = 1 - e^{-\frac{x}{\theta}} \qquad x > 0.$$

Despejando x de la expresion $u = F(x)$ en función de u, se obtiene:

$$x = F^{-1}(u) = -\theta \ln(1-u)$$

Si $u \sim U(0,1)$ entonces $-\theta \ln(1-u)$ tiene una distribución exponencial de media θ .

Como ejemplo generemos un conjunto de 10 valores de la densidad exponencial con media $\theta = 1/2$.

Los siguientes números son números aleatorios:

0.2214 0.8254 0.3403 0.2439 0.1343
0.9385 0.2584 0.7670 0.0007 0.6333

Usando estos números como valores de u en la ecuación obtenemos, respectivamente:

0.1251 0.8741 0.2080 0.1398 0.0721
1.3944 0.1495 0.7284 0.0004 0.5016

La cual es una muestra de tamaño 10 de una exponencial de media $\theta = 1/2$.

Simulación de una distribución exponencial

Para comprobar el ejemplo anterior, se ha desarrollado un sencillo programa en java que permite crear muestras de tamaño n con media θ de una distribución exponencial, y se ha usado JFreeChart para graficar la densidad de la muestra.

El código en java para generar una muestra de tamaño n y con media θ es el siguiente:

```
public double[] genMuestraExponencial(int n, double theta){

    if(n<1 || theta<=0)
        return null;

    double muestra[]=new double[n];
    for(int i=0; i<n;i++)
        muestra[i]=-theta*Math.log(1-Math.random());
    return muestra;
}
```

En la Fig. 1 se muestran los resultados de una muestra Exponencial con diversos valores, donde se puede apreciar claramente que si es una muestra pequeña, fig. 1 a), pareciera que no converge a una función exponencial, sin embargo, si la muestra es grande, esta converge claramente a la función exponencial.

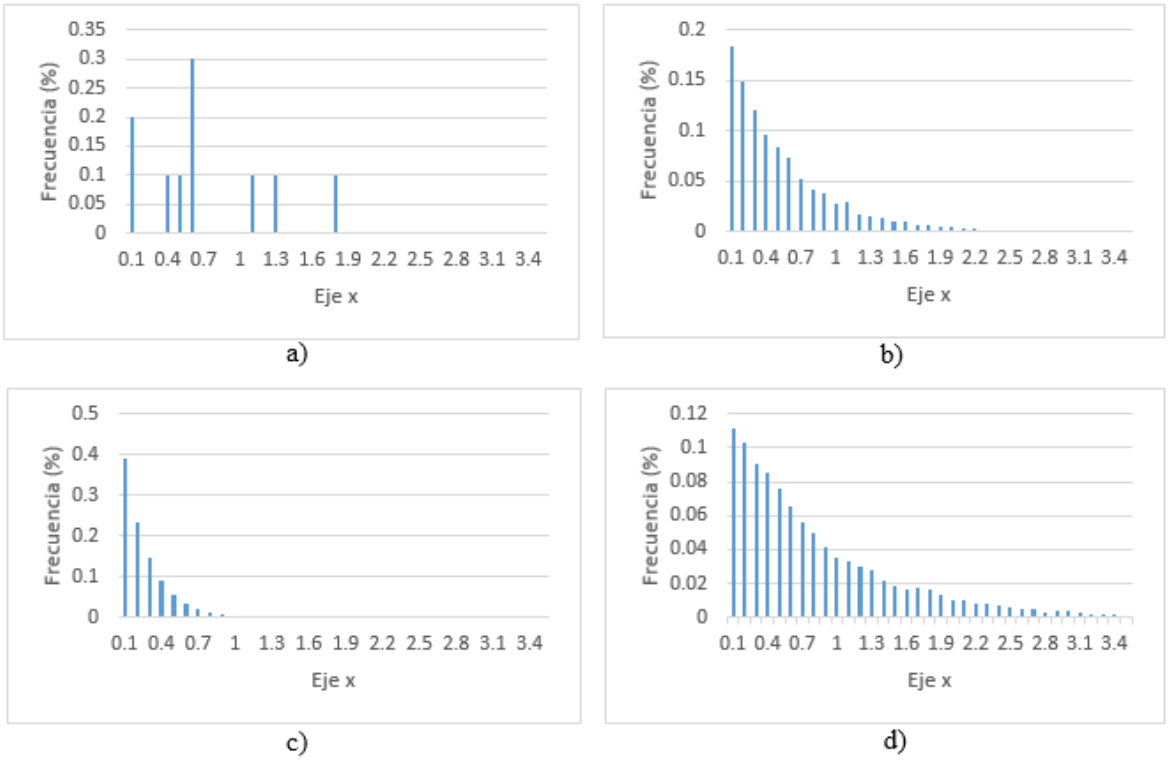


Fig. 1 Muestras de exponenciales a) n=10 y $\theta = 0.5$, las 3 siguientes n=10,000 b) $\theta = 0.5$, c) $\theta = 0.2$, d) $\theta = 0.8$

Distribución Normal $N(\mu, \sigma^2)$

Para simular la densidad normal estándar $N(0,1)$. Supongamos $u_1 \sim U(0,1)$ y $u_2 \sim U(0,1)$ son variables independientes, entonces:

$$f(u_1,u_2)=\begin{cases}1 & 0\le u_1\le 1,\ \ 0\le u_2\le 1\\ 0 & en\ cualquier\ otro\ caso.\end{cases}$$

Definimos

$$X=h_1(u_1,u_2)=\sqrt{2\log(1/u_1)}\ \cos(2\pi\,u_2).$$

$$Y=h_2(u_1,u_2)=\sqrt{2\log(1/u_1)}\ \text{sen}(2\pi\,u_2).$$

Entonces $X\sim N(0,1)$ y $Y\sim N(0,1)$, además que X e Y son independientes. Utilizando esta estrategia, la distribución normal estándar puede simularse fácilmente. Además si $X\sim N(0,1)$, entonces $Z=\sigma X+\mu$ se distribuye como $N(\mu,\ \sigma^2)$.

Simulación de una distribución $N(\mu,\ \sigma^2)$

Aplicando lo anterior se ha desarrollado un programa en java que permite crear muestras de tamaño n con media μ y desviación estándar σ de una distribución normal.

El código en java para generar una muestra de tamaño n con media μ y desviación estándar σ es el siguiente:

```
public double[] genMuestraNormal(int n, double desvEstandar,double media){

    if(n<1 || desvEstandar<=0 || media<=0)
        return null;

    double muestra[]=new double[n];
    for(int i=0; i<n;i++)
        muestra[i]=( Math.sqrt(2*Math.log10(1/Math.random()) )
                     *Math.cos(2*Math.PI*Math.random()) )*desvEstandar
                     + media;
    return muestra;
}
```

Entonces para una muestra de tamaño $n=10$, $\mu=0$ y $\sigma=1$ se puede visualizar en la Fig. 2 a), ahora si aumentamos $n=100$ se puede ver en la Fig. 2 b) que la forma de la normal empieza a aparecer, y si tomamos $n=100{,}000$, se puede ver claramente que el comportamiento de los datos es el de una normal, Fig. 2 c).

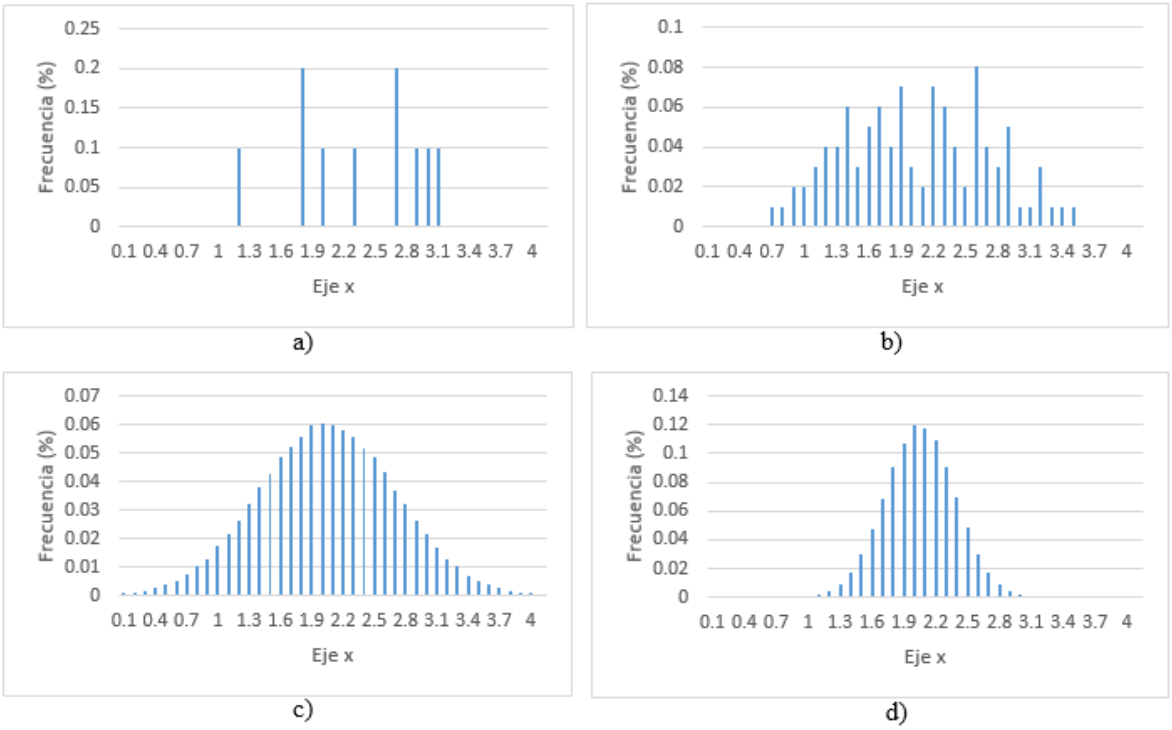


Fig. 2 Muestras de una normal con μ=0, a)n=10, σ=1, b)n=100, σ=1, c)n=100,000, σ=1, d)n=100,000, σ=0.5

Así también, se muestra el comportamiento de una muestra de tamaño $n = 100{,}000$, $\mu = 0$ y $\sigma = 0.5$, Fig. 2 d).

Aproximaciones por Montecarlo

La ley de los grandes números nos dicen que si $x_1, x_2, \dots, x_n$ es una sucesión de variables aleatorias independientes indenticamente distribuidas con media μ, entonces la media muestral

$$M_n = \frac{x_1 + x_2 + \cdots + x_n}{n},$$

cuando n es grande se cumple $M_n \cong \mu$.

Ahora si μ es desconocido podemos considerar M_n, para valores grandes de n, como un estimador de μ. Y la magnitud del error de esta estimación, por el teorema central del límite, es como 3 veces el error estándar. Esto es, el intervalo

$$\left(M_n - \frac{3\,\sigma}{\sqrt{n}},\ \ M_n + \frac{3\,\sigma}{\sqrt{n}}\right),$$

contiene el valor desconocido de μ con un determinado nivel de certeza (una probabilidad de 0.9974). Recuerde que para n grande puede reemplazarse $\sigma \cong s$.

Los estimadores M_n pueden utilizarse para estimar valores puramente matemáticos que son demasiado difíciles de calcular directamente. En este caso los estimadores M_n se denominan aproximaciones por Montecarlo.

Por ejemplo, supongamos que queremos aproximar la integral $\int_a^b g(x)dx$, asumiendo que dicha integral es finita. Sea $f(x)$ una función de densidad en el intervalo (a,b) tal que $f(x) > 0$ para todo $x \in (a,b)$, y supongamos que disponemos de un algoritmo adecuado para generar $x_1, x_2, \dots, x_n$ variables independientes con la distribución dada por $f(x)$. Tendremos que:

$$\int_a^b g(x)dx = \int_a^b \frac{g(x)}{f(x)}\, f(x)\ dx = E\left(\frac{g(x)}{f(x)}\right),$$

cuando x se distribuya con función de densidad $f(x)$.

Entonces podremos estimar la integral $\int_a^b g(x)dx$ mediante:

$$M_n = \frac{1}{n}\sum_{i=1}^{n}\frac{g(x_i)}{f(x_i)} = \frac{1}{n}\sum_{i=1}^{n} T_i,$$

donde:

$$T_i = \frac{g(x_i)}{f(x_i)}.$$

Ahora, supongamos que queremos evaluar:

$$I = \int_0^1 \cos(x^2)\, sen(x^4)dx.$$

El valor no se puede calcular de manera sencilla. Pero puede evaluarse mediante aproximación de Montecarlo.

$$I = E\big(\cos(u^2)\, sen(u^4)\big).$$

Siendo $u \sim Uniforme[0,1]$ .Por lo tanto, para valores grandes de n, la integral I sería aproximadamente igual a:

$$I \cong M_n = \frac{T_1 + \cdots + T_n}{n},$$

donde:

$$T_i = \cos({u_i}^2)\, sen({u_i}^4)$$

Haciendo los respectivos cálculos con $n = 10^4$ y $n = 10^5$ obtenemos la siguiente tabla:

$\boldsymbol{n}$	$\boldsymbol{M_n}$	$\boldsymbol{M_n - 3s/\sqrt{n}}$	$\boldsymbol{M_n + 3s/\sqrt{n}}$
10^4	0.138850	0.134105	0.143595
10^5	0.139484	0.137974	0.140993

En la cual se aprecia la aproximación de la integral I por el Método de Montecarlo.

Cadenas de Markov

Intuitivamente una cadena de Markov representa el movimiento aleatorio de un objeto. Podemos considerar $\{x_n\}$ como la posición o el valor de un objeto en el instante n. En tal caso, existen reglas que permiten conocer las probabilidades de lugar en que se situará el objeto en el siguiente movimiento.

Una cadena de Markov requiere un espacio de estados S que representa el conjunto de todos los lugares a los que puede ir el objeto. Por ejemplo, $S = \{1, 2, 3\}$, o $S = \{fondo, superficie\}$, o S puede ser el conjunto de los enteros positivos.

Una cadena de Markov requiere también unas probabilidades de transición, que dan la probabilidad hacia donde se desplazara el objeto en el siguiente movimiento. Concretamente para $i, j \in S$, el valor P_{ij} es la probabilidad de que, si el objeto esta en i, efectúe el siguiente salto a j. Por lo tanto el conjunto $\{P_{ij}: i, j \in S\}$ de probabilidades de transición cumple que, para todo $i, j \in S$, $P_{ij} \geq 0$ y además para cada $i \in S$ se cumple:

$$\sum_{j \in S} P_{ij} = 1 .$$

También necesitamos establecer el lugar en que empieza la cadena de Markov, generalmente establecemos que $x_0 = s$ para algún estado determinado del espacio de estados.

De forma más general, podríamos tener una distribución inicial $\{\mu_i,\ i \in S\}$, donde $\mu_i = P(x_0 = i)$, en este caso se debe cumplir $\mu_i \geq 0$ para todo $i \in S$ y $\sum_{i \in S} \mu_i = 1$.

Resumiendo, S es el espacio de estados de todos los lugares al los que el objeto puede desplazarse, μ_i representa la probabilidad de que el objeto parta del punto i, y P_{ij} la probabilidad de que si el objeto está en el punto i se desplace al punto j en el siguiente movimiento. En términos de la secuencia de valores aleatorios $x_0, x_1, x_2, \ldots$ tenemos que

$$P(x_{n+1} = j \mid x_n = i) = P_{ij}$$

para cualquier entero positivo n y cualquier $i, j \in S$. Una cadena de Markov cumple que la probabilidad de desplazamiento no dependa de la historia previa de la cadena, es decir, es necesario que se cumpla:

$$P(x_{n+1} = j \mid x_n = i, x_{n-1} = r_{n-1}, \ldots, x_0 = r_0) = P_{ij}$$

para todo n y todo $i, j, r_{n-1}, \ldots, r_0 \in S$.

Por ejemplo, sea $S = \{1,2,3\}$ y las probabilidades de transición $P_{11} = 0,\ P_{12} = 1/2,\ P_{13} = 1/2,\ P_{21} = 1/3,\ P_{22} = 1/3,\ P_{23} = 1/3,\ P_{31} = 1/4,\ P_{32} = 1/4,\ P_{33} = 1/2$. Si la

cadena esta en la situación 3, entonces tiene una probabilidad de 1/4 de desplazarse al estado 1.

Se pueden expresar las probabilidades de transición P_{ij} en forma matricial

$$\begin{matrix} edo.1 & edo,2 & edo.3 \\ \uparrow & \uparrow & \uparrow \end{matrix}$$
$$(P_{ij}) = \begin{bmatrix} 0 & 1/2 & 1/2 \\ 1/3 & 1/3 & 1/3 \\ 1/4 & 1/4 & 1/2 \end{bmatrix} \begin{matrix} \rightarrow edo.1 \\ \rightarrow edo.2 \\ \rightarrow edo.3 \end{matrix}$$

La matriz (P_{ij}) se denomina matriz estocástica.

Observe que la suma de las filas debe ser igual a uno, ya que necesariamente se mueve a cualquier otro estado o se mantiene en el mismo estado.

Cálculos con cadenas de Markov

Sea $\{x_n\}$ una cadena de Markov $\{x_n\}$ con espacio de estados S, probabilidades de transición $\{P_{ij}\}$ y distribución inicial $\{\mu_i\}$. Entonces para todo $i \in S$, se cumple

$$P(x_1 = i) = \sum_{k \in S} \mu_k P_{ki} \, ,$$

que representa la probabilidad de que en el momento 1 se este el estado i.

Distribuciones Estacionarias

Sea $\{\pi_i; i \in S\}$ una distribución definida sobre S tal que, para todo $i \in S$, $\pi_i \geq 0$ y además $\sum_{i \in S} \pi_i = 1$.

Decimos que distribución $\{\pi_i;\ i \in S\}$ es estacionaria para una cadena de Markov con probabilidades de transición $\{P_{ij}\}$ en un espacio de estados S, si se cumple que para todo $j \in S$,

$$\sum_{i \in S} \pi_i \, P_{ij} = \pi_j \, .$$

Ahora si suponemos que $\{\pi_i : i \in S\}$ es una distribución estacionaria para una cadena de Markov con probabilidades de transición $\{P_{ij}\}$ en un espacio de estados S. Si para algún entero n, se tiene que para todo $i \in S$

$$P(X_n = i) = \pi_i \ .$$

Entonces también se tiene que para todo $i \in S$ y para todo $m > n$

$$P(X_m = i) = \pi_i \qquad \forall\, i \in S.$$

Se dice que una cadena de Markov es reversible respecto a una distribución $\{\pi_i\}$ si para todo $i, j \ \in S$ se satisface

$$\pi_i P_{ij} = \pi_j P_{ji} \ .$$

Entonces podemos implicar que si una cadena de Markov es reversible respecto a una distribución $\{\pi_i\}$ entonces $\{\pi_i\}$ es una distribución estacionaria para la cadena.

Teorema del límite para una cadena de Markov*

Decimos que una cadena de Markov es irreducible si la cadena puede desplazarse de un estado a cualquier otro estado. De forma equivalente, la cadena de Markov es irreducible si existe un entero positivo n con:

$$P_i(X_n = j) > 0.$$

Dada una matriz de transición $\{P_{ji}\}$ en un espacio de estados S, y un estado $i \in S$, el periodo de una cadena de Markov de i es el máximo común divisor (m.c.d) del conjunto

$$\left\{n \geq 1;\ P_{ii}^{(n)} > 0\right\},$$

siendo:

$$P_{ii}^{(n)} = P(X_n = i \mid X_0 = i),$$

es decir, el periodo de i es el m.c.d de las veces que es posible desplazarse de i a i. Por ejemplo, el periodo de i es dos si solo es posible desplazarse de i a i en un número par de etapas. Por otra parte si $P_{ii} > 0$ el periodo de i es evidentemente uno.

Decimos que una cadena de Markov es aperiódica si el periodo de cada estado es igual a uno. Si una cadena de Markov satisface para todo $i. j \in S$ que $P_{ij} > 0$, entonces la cadena es irreducible y aperiódica.

Si una cadena de Markov es irreducible y aperiódica, con una distribución estacionaria $\{\pi_i\}$, independiente de la distribución inicial $\{\mu_i\}$, se tiene que para todo $i \in S$

$$\lim_{n\to\infty} P(X_n = i) = \pi_i \ .$$

Cadenas de Markov y Método de Monte Carlo

Para la mayoría de las funciones de distribución no existe una forma directa de simular, mediante un ordenador, variables aleatorias que tengan dichas distribuciones. Sin embargo,

utilizando el método de Monte Carlo, y las cadenas de Markov podemos generar variables aleatorias independientes idénticamente distribuidas, con la distribución deseada.

Para una cadena de Markov Monte Carlo, la aproximación tiene dos fuentes de error:

- El error de Monte Carlo debido a que M no sea lo suficientemente grande.
- El error de la cadena de Markov debido a que N puede no ser lo suficientemente grande.

El Método de Monte Carlo y la cadena de Markov.

Suponga que se desea estimar el valor esperado $A = E(h(z))$, siendo

$$\begin{cases} P(z = j) = \pi_j & j \in S \\ P(z = j) = 0 & j \notin S. \end{cases}$$

Suponga que para $i = 1, 2, \dots, M$ podemos generar los valores $x_1^{[i]}, x_2^{[i]}, \dots, x_N^{[i]}$ a partir de alguna cadena de Markov que sea irreducible y aperiódica y tenga una distribución estacionaria $\{\pi_j\}$. Sea:

$$\hat{A} = \frac{1}{M}\sum_{i=1}^{M} h(x_N^{[i]}).$$

Si M y N son suficientemente grandes, entonces $A \cong \hat{A}$.

Resulta poco eficiente ejecutar M cadenas de Markov diferentes. En su lugar se ejecuta una sola cadena de Markov y se promedian los diferentes valores de la cadena. Para una cadena de Markov irreducible ejecutada un número bastante grande de veces, este procedimiento convergerá a la respuesta correcta.

El Método de Montecarlo y una única Cadena de Markov

Suponga que deseamos estimar un valor esperado $A = E(h(z))$, siendo

$$\begin{cases} P(z = j) = \pi_j & j \in S \\ P(z = j) = 0 & j \notin S. \end{cases}$$

Suponga que podemos generar los valores $x_1, x_2, \dots, x_N$ a partir de una cadena de Markov que es irreducible y aperiódica y tiene una distribución estacionaria $\{\pi_j\}$. Para algún entero $B \geq 0$ sea:

$$\hat{A} = \frac{1}{N - B + 1} \sum_{i=B+1}^{N} h(x_i).$$

Si $N - B$ es suficientemente grande, entonces $A \cong \hat{A}$.

Aquí B es el tiempo de eliminación, establecido para eliminar la influencia de x_1, el valor de partida de la cadena. Existen controversias entre los estadísticos sobre la mejor selección de B. Sin embargo, si el valor inicial x_1 es razonable es correcto tomar $B = 0$ con la condición de que N sea suficientemente grande.

Cómo podemos construir una cadena de Markov que tenga una distribución estacionaria en concreto. Sorprendentemente, este problema resulta más fácil de resolver de lo que puede esperarse.

Algoritmo Metropolis-Hastings

Suponga que disponemos de una distribución de probabilidad $\{\pi_j\}$ en el espacio de estados S. ¿Cómo podemos construir una cadena de Markov en S que tenga como distribución estacionaria $\{\pi_j\}$?

El algoritmo Metropolis-Hastings diseña una cadena de Markov que seleccionando correctamente la probabilidad de aceptación, terminamos por crear una cadena de Markov que tiene una distribución estacionaria $\{\pi_i\}$.

Detalles del Algoritmo:

Partimos de un espacio de estados S y una distribución de probabilidad $\{\pi_i \neq 0\}$ definida sobre S. A continuación, se considera una cadena de Markov sencilla con probabilidades de transición $\{q_{ij}\,;\, i,j \in S\}$, conjunto que se denomina distribución propuesta, e imponemos que $q_{ij} \geq 0$ y $\sum_{j \in S} q_{ij} = 1$ para cada $i \in S$. La cadena $\{q_{ij}\}$ puede incluso no tener distribución estacionaria.

Dado $x_n = i$, el algoritmo Metropolis-Hastings calcula el valor x_{n+1} como sigue:

1. Selecciona $y_{n+1} = j$ de acuerdo con la cadena de Markov $\{q_{ij}\}$.
2. Establece para $i \neq j \;\; y \;\; q_{ij} \neq 0$,

$$\alpha_{ij} = min\left\{1, \frac{\pi_j q_{ji}}{\pi_i q_{ij}}\right\}.$$

Si $i = j \ \ o \ \ i \neq j$ pero $q_{ij} = 0$, definimos $\alpha_{ij} = 0$.

Ahora definimos la matriz de transición P como sigue:

Para $i \neq j$, con probabilidad $p_{ij} = q_{ij}\alpha_{ij}$, se establece $x_{n+1} = y_{n+1} = j$.

Para $i = j$, con probabilidad $p_{ii} = 1 - \sum_{i \neq j} q_{ij}\alpha_{ij}$, se establece $x_{n+1} = x_n = i$.

Entonces $P = \{p_{ij}\}$ es la matriz de transición que conduce a una cadena de Markov a que tenga a $\{\pi_i\}$ como distribución estacionaria.

El Muestreador de Gibbs

Es una versión especial del algoritmo Metropolis-Hastings, diseñado para distribuciones multivariantes y que selecciona las probabilidades propuestas q_{ij} de tal forma que siempre tengamos $\alpha_{ij} = 1$.

Sea $S = (i_1, i_2) = \{\dots, -2, -1, 0, 1, 2 \dots\} \times \{\dots, -2, -1, 0, 1, 2 \dots\}$, el conjunto de todos los pares ordenados enteros, $\{\pi_i\}$ una distribución sobre S y defina la distribución propuesta $\{q_{ij}^{(1)}\}$ como sigue:

Sea $v(i) = \{j \in S;\ j_2 = i_2\}$

$v(i)$ es el conjunto de todos los estados $j \in S$ tales que i y j coincidan en sus segundas coordenadas. Es decir, $v(i)$ es una línea horizontal definida en S que pasa por el punto i. Definimos $q_{ij}^{(1)} = 0$ si $j \notin v(i)$, es decir, si i y j difieren en la segunda coordenada. Si $j \in v(i)$, es decir, si i y j coinciden en la segunda coordenada, se define:

$$q_{ij}^{(1)} = \frac{\pi_j}{\sum_{k \in v(i)} \pi_k}$$

Si $j \in v(i)$ entonces $i \in v(j)$, además $v(j) = v(i)$.

Por lo tanto

$$\alpha_{ij} = min\left\{1, \frac{\pi_j q_{ji}^{(1)}}{\pi_i q_{ij}^{(1)}}\right\} = min\left\{1, \frac{\pi_j\left(\frac{\pi_i}{\sum_{k \in v(j)} \pi_k}\right)}{\pi_i\left(\frac{\pi_j}{\sum_{l \in v(i)} \pi_l}\right)}\right\}$$

$$\alpha_{ij} = min\left\{1, \frac{\pi_j \pi_i}{\pi_i \pi_j}\right\} = \min\{1,1\} = 1$$

Este algoritmo por sí solo no es muy útil, porque solamente propone estados en $v(i)$, y nunca cambia el valor de la segunda coordenada.

También podemos definir una línea vertical que pase por i como $H(i) = \{j \in S;\ j_1 = i_1\}$ de modo que $H(i)$ es el conjunto de todos los estados j tales que i y j coinciden en su primera coordenada, es decir, $H(i)$ es una línea vertical definida en S que pasa por el punto i.

Entonces, podemos definir $q_{ij}^{(2)} = 0$ si $j \notin$ H(i), es decir, si i y j difieren en la primera coordenada, mientras que si $j \in$ H(i), es decir, si i y j coinciden en la primer coordenada, entonces:

$$q_{ij}^{(2)} = \frac{\pi_j}{\sum_{k \in H(i)} \pi_k}$$

Como antes, calculamos que para esta propuesta siempre tendremos que $\alpha_{ij} = 1$.

El muestreador de Gibbs trabaja combinando alternativamente estos 2 algoritmos Metropolis-Hastings diferentes. Es decir, dado un valor $x_n = i$, proporciona un valor x_{n+1} de la siguiente forma:

1. Propone un valor $y_{n+1} \in v(i)$ de acuerdo con la distribución propuesta $\{q_{ij}^{(1)}\}$
2. Establece $j = y_{n+1}$, es decir, se desplaza horizontalmente.
3. Propone un valor $z_{n+1} \in H(j)$ de acuerdo con la distribución propuesta $\{q_{ij}^{(2)}\}$, es decir, ahora se desplaza verticalmente.
4. Establece $x_{n+1} = z_{n+1}$

De esta forma, el muestreador de Gibbs realiza un zigzag por el espacio de estados S, moviéndose alternativamente en la dirección horizontal y en la dirección vertical. El algoritmo del muestreador de Gibbs resulta ser una cadena de Markov $x_0, x_1, \dots, x_n, \dots$ que tiene como distribución estacionaria $\{\pi_i\}$.

Aplicación del Muestreo de Gibbs.

Supongamos que se desea generar muestras de la distribución conjunta de $(Y_1, Y_2, \dots, Y_k) \in \mathbb{R}^k$. Para esto suponemos que se pueden generar muestras para cada una de las distribuciones condicionales $Y_i | Y_{i-1} = y_{-i}$ siendo $Y_{i-1} = (Y_1, \dots, Y_{i-1}, Y_i, \dots, Y_k)$.

El muestreador de Gibbs procede entonces de la siguiente forma iterativa:

1. Especifica un valor inicial $(y_{1(0)}, \dots, y_{k(0)})$ para $(Y_1, \dots, Y_k)$
2. Para $N > 0$, genera $Y_{i(N)}$ a partir de su distribución condicional dado $(y_{1(N)}, \dots, y_{i-1(N)}, y_{i+1(N-1)}, \dots, y_{k(N-1)})$ para cada $i = 1,2, \dots, k$.

Por ejemplo:

Si $k = 3$, primero especificamos $(y_{1(0)}, y_{2(0)}, y_{3(0)})$ y a continuación generamos:

$$Y_{1(1)} \mid Y_{2(0)}, Y_{3(0)}$$

$$Y_{2(1)} \mid Y_{1(1)}, Y_{3(0)}$$

$$Y_{3(1)} \mid Y_{1(1)}, Y_{2(1)}$$

De este modo se obtiene $(Y_{1(1)}, Y_{2(1)}, Y_{3(1)})$, seguidamente generamos:

$$Y_{1(2)} \mid Y_{2(1)}, Y_{3(1)}$$

$$Y_{2(2)} \mid Y_{1(2)}, Y_{3(1)}$$

$$Y_{3(2)} \mid Y_{1(2)}, Y_{2(2)}$$

Y se obtiene $(Y_{1(2)}, Y_{2(2)}, Y_{3(2)})$, etc. En realidad no se necesita especificar $Y_{1(0)}$ puesto que nunca se utiliza.

Puede demostrarse que en condiciones generales de imparcialidad $(Y_{1(N)}, \dots, Y_{k(N)})$ converge en distribución a la distribución conjunta de $(Y_1, Y_2, \dots, Y_k)$ cuando $N \to \infty$.

Además, bajo ciertas condiciones:

$$\bar{\omega} = \frac{1}{N} \sum_{i=1}^{N} \omega(Y_{1(i)}, Y_{2(i)}, \dots, Y_{k(i)})$$

Converge casi seguramente a $E(\omega(Y_1, Y_2, \dots, Y_k))$.

Algunas veces, incluso el muestro de Gibbs no puede implementarse directamente porque no se dispone de algoritmos para generar valores de todas las posibles condicionales. Existen varias técnicas para tratar esta situación, pero para muchas aplicaciones estadísticas la técnica de las variables latentes acostumbra a ser adecuada.

Ejemplo: Se aplicara el muestreo de Gibbs a la función bivariante:

$$f(x, y) = (1/28)(2x + 3y + 2)$$

La distribución condicional para x es:

$$f(x|y) = \frac{f(x, y)}{f(y)} = \frac{2x + 3y + 2}{6y + 8}$$

La distribución para y es:

$$f(y|x) = \frac{f(x, y)}{f(x)} = \frac{2x + 3y + 2}{4x + 10}$$

Por lo tanto, un muestreador de Gibbs para el muestreo de x e y seguirá los siguientes pasos:

1. Iniciar $j = 0$ y establecer los valores iniciales. Aquí colocaremos $x^{j=0} = -5$ e $y^{j=0} = -5$
2. Asignar a x^{j+1} una muestra de $f(x|y = y^j)$.
3. Asignar a y^{j+1} una muestra de $f(y|x = x^{j+1})$.
4. Incrementar $j = j + 1$ y regresar al paso 2 hasta $j = 2000$.

El problema que se tiene en el algoritmo anterior es el hecho de generar muestras de las distribuciones condicionales. Sin embargo, al ser distribuciones univariadas y al ser posible calcular $F^{-1}(x)$ para cada una, se puede usar una subrutina de inversión para obtener una muestra de cada densidad condicional.

Por lo que, para encontrar la inversa de la densidad condicional de $y|x$, se necesita resolver lo siguiente:

$$u = \int_0^z \frac{2x + 3y + 2}{4x + 10}\, dy$$

Entonces

$$u(4x + 10) = (2x + 2)y + (3/2)y^2|_0^z$$

$$u(4x + 10) = (2x + 2)z + (3/2)z^2$$

$$(2/3)u(4x + 10) = z^2 + (2/3)(2x + 2)z$$

Completando cuadrados para z y resolviendo para z se obtiene:

$$z = \sqrt{(2/3)u(4x + 10) + ((1/3)(2x + 2))^2} - \left(\frac{1}{3}\right)(2x + 2),$$

donde $u \sim Uniforme(0,1)$.

Dando el valor actual de x y un valor aleatorio de u, z es un valor aleatorio de la densidad condicional $y|x$. Del mismo modo, se puede encontrar la inversa de $x|y$.

Lo siguiente, es una función en java que implementa el muestreo de Gibbs:

```
public void ejemploGibss(int tamMuestra){
    //prueba con tamMuestra=2000
    double x[]=new double[tamMuestra];
    double y[]=new double[tamMuestra];
    double u;

    x[0]=-5;
    y[0]=-5;
    for(int i=1;i<tamMuestra;i++){
        // muestra de x|y
        u=Math.random();
        x[i]=Math.sqrt(u*(6*y[i-1]+8)
                + Math.pow(1.5*y[i-1]+1, 2))-(1.5*y[i-1]+1);
        //muestra de y|x
        u=Math.random();
        y[i]=Math.sqrt((2*u*(4*x[i]+10))/3
                + Math.pow((2*x[i]+2)/3, 2))-(2*x[i]+2)/3;
    }
}
```

Esta función da el valor de -5 a los primeros valores de x e y. Entonces, actualiza el valor de x usando el valor actual de y. Después, actualiza el valor de y usando el nuevo valor de x. Ambas son actualizadas usando el método de inversión para muestrear.

Este algoritmo produce muestras de las distribuciones marginales de x e y. En las gráficas 3 a) y 3 b) se puede observar el valor que toma x e y en cada iteración, respectivamente. Las gráficas 3 c) y 3 d) representan la densidad marginal de las 2 primeras gráficas, respectivamente.

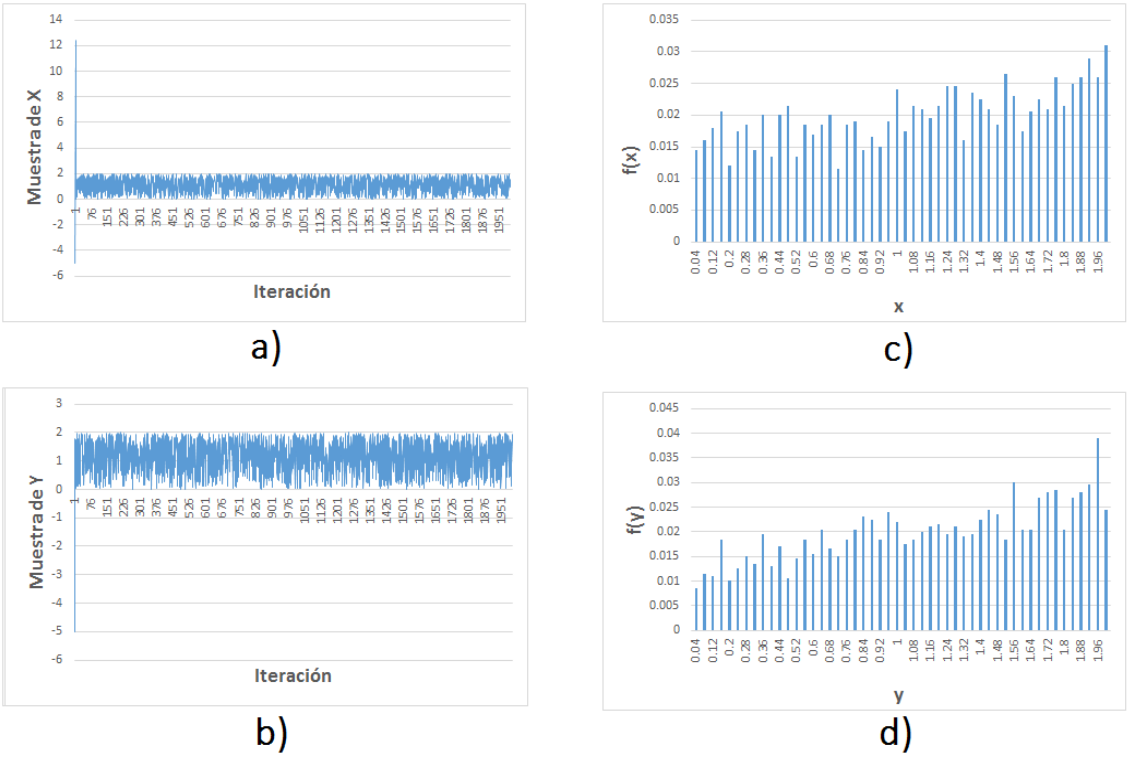


Fig. 3 Resultados del muestreo de Gibbs usando el método de inversión para muestrear densidades condicionales

Se debe tener en cuenta que los valores iniciales son muy pobres (ya que -5 no es un punto valido en ninguna dimensión de la densidad), sin embargo, el algoritmo converge rápidamente a la región apropiada, [0,2]. Esto generalmente le toma un numero de iteraciones a la MCMC encontrar la región apropiada. Por lo que, generalmente se descarta una cantidad de las primeras iteraciones antes de hacer cálculos. Las distribuciones marginales son producidas de las últimas 1500 iteraciones del algoritmo.

Aparte de examinar las densidades marginales de x e y, se examina también la densidad conjunta. En la figura 4 se pueden observar diferentes graficas tomadas después de algunas iteraciones, donde cada punto representa una iteración. En la gráfica 4.a) se pueden observar valores pobres, que se encuentran fuera de la región apropiada, sin embargo, en las siguientes iteraciones, podemos ver que el algoritmo encuentra la región y se mantiene ahí.

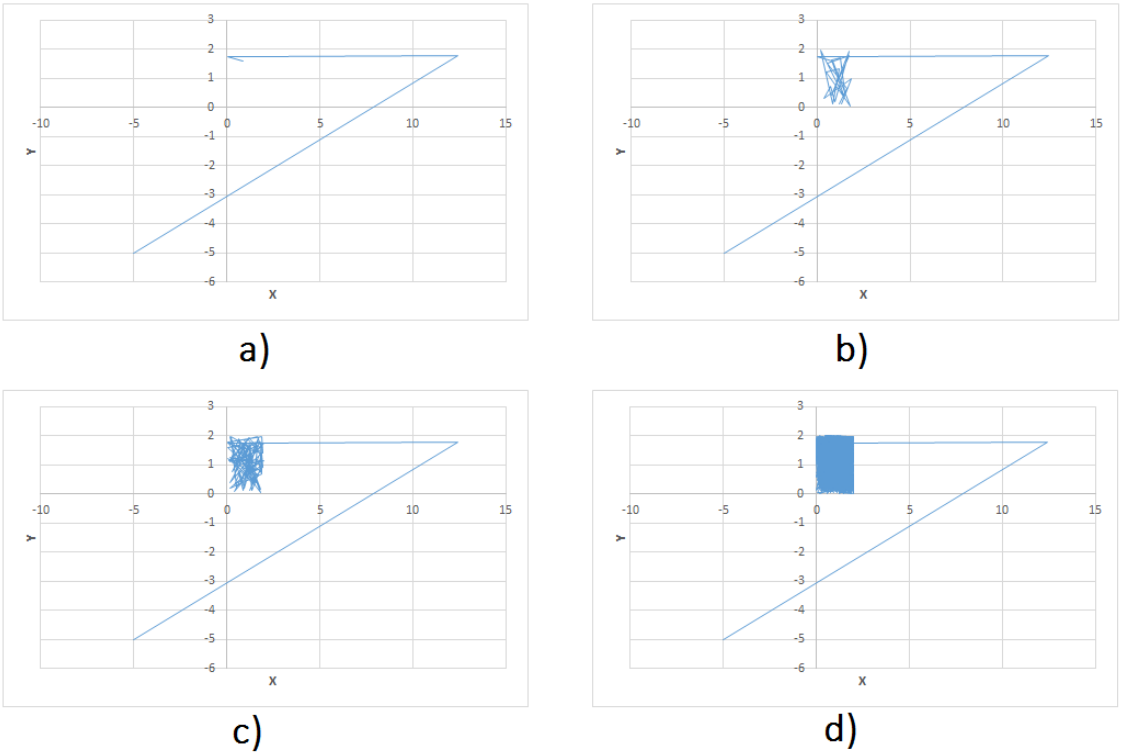


Fig. 4 Resultado del muestreo de Gibbs usando el método de inversión para muestrear a densidades condicionales. Vista después de a) 5, b) 25, c) 100 y d) 2000 iteraciones.

Capitulo 3 Modelos de Tópicos.

Un modelo de tópicos es un modelo generativo de documentos. Los modelos de tópicos se basan en la idea de que los documentos son mezclas de tópicos, en los que un tópico es una distribución de probabilidad sobre palabras. Para hacer un documento, se opta por una distribución sobre tópicos, posteriormente cada palabra en el documento, se elige de un tópico. Técnicas estadísticas estándar se pueden utilizar para invertir este proceso, inferir el conjunto de tópicos que son responsables de la generación de una colección de documentos.

Un modelo generativo para documentos se basa en reglas de muestreo probabilístico que describen cómo las palabras en los documentos podrían ser generadas sobre la base de variables latentes (escondidas). Durante el ajuste de un modelo generativo, el objetivo es encontrar el mejor conjunto de variables latentes que pueden explicar los datos observados (las palabras en los documentos), en el supuesto de que el modelo genera realmente los datos.

Las siguientes figuras ilustran el enfoque de modelado de tópicos de dos formas distintas: como un modelo generativo y como un problema de inferencia estadística. La figura 5 muestra el proceso generativo que se ilustra con dos tópicos que son temáticamente relacionados con “herramientas” y “animales domésticos” que se ilustran como bolsas que contienen diferentes distribuciones sobre palabras. Diferentes documentos pueden ser producidos escogiendo las palabras de un tópico en función del peso dado al tópico. Por ejemplo, los documentos 1 y 3 se generaron mediante el muestreo sólo del tópicos 1 y 2, respectivamente, mientras que el documento 2 se generó por una mezcla de los dos tópicos. Tenga en cuenta que los números sobrescritos asociados con las palabras en los documentos indican qué tópicos se utilizó para elegir la palabra. En la forma en que se define el modelo, no hay exclusividad de que las palabras formen parte de un solo tópico, esto permite que los modelos de tópicos sirvan para captar la polisemia, donde la misma palabra tiene varios significados. Por ejemplo, ambos tópicos “herramientas” y “animales domésticos” pueden dar una alta probabilidad a la palabra perico, lo que es razonable dado el carácter polisémico de la palabra.

El proceso generativo no hace ninguna suposición sobre el orden de las palabras que aparecen en los documentos. La única información relevante para el modelo es el número de veces que se producen las palabras. Esto se conoce como la suposición de “bolsa de palabras”.

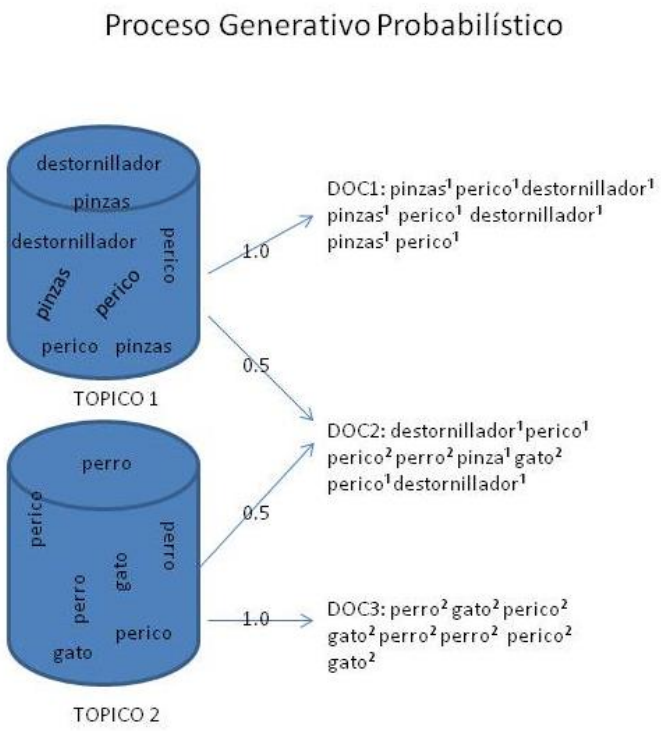


Fig. 5 (Idea original de Mark Steyvers, Tom Griffiths))

En la figura 6 se ilustra el problema de la inferencia estadística. Teniendo en cuenta las palabras observadas en un conjunto de documentos, nos gustaría saber la clase de tópicos más probables que hayan generado los datos. Esto implica inferir la distribución de probabilidad sobre palabras asociadas con cada tópico y la distribución sobre los tópicos para cada documento.

Una variedad de modelos probabilísticos de tópicos se han utilizado para analizar el contenido de los documentos y el significado de las palabras. Todos estos modelos utilizan la misma idea fundamental - que un documento es una mezcla de tópicos - pero hacen ligeramente diferentes supuestos estadísticos.

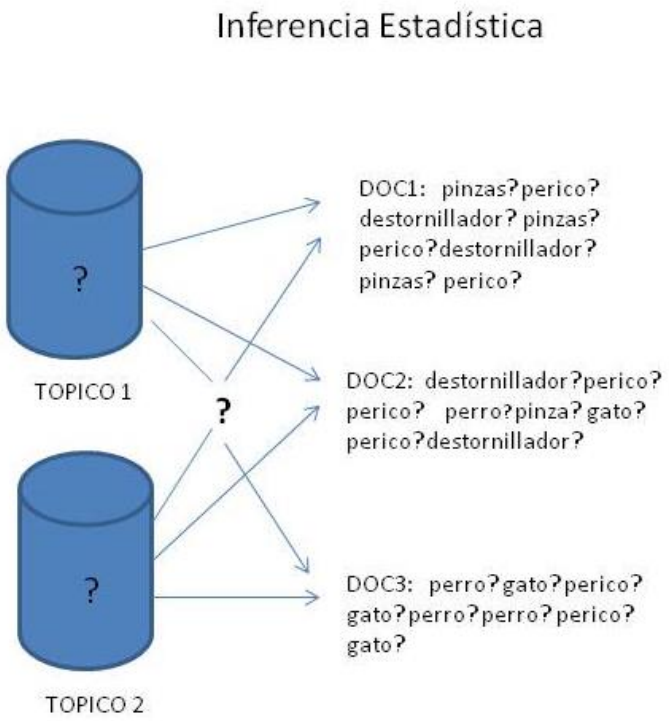


Fig. 6 (Idea original de Mark Steyvers, Tom Griffiths)

Modelo Generativo LDA

En LDA (**Latent Dirichlet Allocation)**, si las observaciones son palabras en documentos, cada documento puede verse como una mezcla de varios tópicos. Esto es similar a (pLSA) (probabilistic latent semantic analysis) excepto que en LDA se asume que la distribución de tópicos tiene una distribución a priori de Dirichlet.

$$Dir(\boldsymbol{p};\boldsymbol{\alpha}) = \frac{1}{B(\alpha)}\prod_{i=1}^{n} p_i^{\alpha_i - 1} \qquad donde \qquad B(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{n}\Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n}\alpha_i)} \tag{1}$$

Como $Dir(\boldsymbol{p};\boldsymbol{\alpha})$ es una función de densidad, entonces se cumple

$$\int \frac{1}{B(\boldsymbol{\alpha})}\prod_{i=1}^{n} p_i^{\alpha_i - 1}\, d\boldsymbol{p} = 1 \tag{2}$$

Lo cual implica que

$$\int \prod_{i=1}^{n} p_i^{\alpha_i - 1}\, d\boldsymbol{p} = B(\boldsymbol{\alpha}) \tag{3}$$

Desarrollo del modelo LDA

Primeramente describiremos la lista de símbolos en el modelo LDA.

Variables	Significado
N	Número de palabras en el corpus.
$W = \{w_i\}$	El corpus y w_i denota una palabra.
$Z = \{z_i\}$	Tópicos latentes, z_i tópico asignado a la palabra i-esima.
$W_{-i} = W - \{w_i\}$	El corpus excluyendo w_i.
$Z_{-i} = Z - \{z_i\}$	Tópicos latentes excluyendo z_i.
K	Número de tópicos.
V	Número de palabras (términos) en el vocabulario.
D	Número de documentos.
N_d	Número de palabras en el documento d.
v	Denota una palabra (termino) en el vocabulario.
$\boldsymbol{\alpha}$	Vector K-dimensional de reales positivos.
$\boldsymbol{\beta}$	Vector V-dimensional de reales positivos.
$\boldsymbol{\theta}$	Vector K-dimensional de probabilidades.
$\boldsymbol{\phi}$	Vector V-dimensional de probabilidades.

El modelo especifica la siguiente distribución sobre palabras dentro de un documento.

$$p(w_i) = \sum_{j=1}^{K} p(w_{i,} z_j) = \sum_{j=1}^{K} p(w_i / z_i = j) p(z_i = j)$$

donde $p(z_i = j)$ es la probabilidad de que el j-esimo tópico fue elegido para la i-esima palabra y $p(w_i / z_i = j)$ es la probabilidad de elegir la palabra w_i bajo el tópico j.

La siguiente tabla es un ejemplo de la distribución de palabras en un corpus.

	Term_1	Term_2	Term_3	Term_4	Term_5
Doc_1	0.2	0.1	0.1	0.4	0.2
Doc_2	0.5	0.2	0.1	0.1	0.1
Doc_3	0.0	0.2	0.4	0.0	0.4
Doc_4	0.2	0.0	0.3	0.2	0.3

La probabilidad $p(W/Z)$ es una multinomial sobre palabras para los tópicos, es decir,

$$p(W/Z) = p(W/Z, \boldsymbol{\phi}) = \prod_{k=1}^{K} \prod_{v=1}^{V} \phi_{k,v}^{n_{\circ},k,v} \tag{4}$$

donde $\phi_{k,v}$ es la probabilidad de que el termino v sea asignado al tópico k para cualquier documento. $n_{\circ,k,v}$ es el número de veces que el termino v es asignado al tópico k para todo el corpus. El operador $\circ$, dependiendo de su ubicación en $n_{d,k,v}$ significa para cualquier documento, tópico o palabra respectivamente.

La siguiente tabla es un ejemplo de la distribución de términos en los tópicos.

	Term_1	Term_2	Term_3	Term_4	Term_5
Tóp_1	0.2	0.1	0.1	0.4	0.2
Tóp_2	0.5	0.2	0.1	0.1	0.1
Tóp_3	0.0	0.2	0.4	0.0	0.4
Tóp_4	0.2	0.0	0.3	0.2	0.3

El modelo LDA introduce una distribución de Dirichlet prior sobre $\boldsymbol{\phi}$ (Blei 2003). El prior para $\boldsymbol{\phi}$ es el siguiente:

$$p(\boldsymbol{\phi}/\boldsymbol{\beta}) = \prod_{k=1}^{K} \frac{\Gamma(\sum_{v=1}^{V} \beta_v)}{\prod_{v=1}^{V} \Gamma(\beta_v)} \prod_{v=1}^{V} \phi_{k,v}^{\beta_v - 1} \tag{5}$$

Los hiperparámetros β pueden interpretarse como el recuento de las observaciones previas sobre el número de veces que las palabras son elegidas de un tópico antes de cualquier palabra observada en el corpus. Esto suaviza la distribución de palabras en cada tópico, con la cantidad de suavizado determinado por β.

Ahora la probabilidad $p(Z)$ es también una distribución multinomial sobre los tópicos para el documento d, es decir,

$$p(Z) = p(Z/\boldsymbol{\theta}) = \prod_{d=1}^{D} \prod_{k=1}^{K} \theta_{d,k}^{n_{d,k,\circ}} \tag{6}$$

donde $\theta_{d,k}$ es la probabilidad de que el tópico k sea asignado al documento d para cualquier palabra. Aquí $n_{d,k,\circ}$ es el número de veces que el tópico k es asignado al documento d.

La siguiente tabla es un ejemplo de la distribución de tópicos en documentos.

	Tóp_1	Tóp_2	Tóp_3	Tóp_4
Doc_1	0.5	0.2	0.1	0.2
Doc_2	0.8	0.0	0.2	0.0
Doc_3	0.35	0.15	0.25	0.25

Se introduce una Dirichlet prior sobre $\boldsymbol{\theta}$.

$$p(\boldsymbol{\theta}/\boldsymbol{\alpha}) = \prod_{d=1}^{D} \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{d,k}^{\alpha_k - 1} \tag{7}$$

Los parámetros de esta distribución se especifican por $\alpha_1 \ldots \alpha_K$. Cada hiperparámetro α_i se puede interpretar como un recuento de las observaciones previas para el número de veces que el tópico j se muestrea en un documento, antes de haber observado todas las palabras reales de ese documento. Es conveniente utilizar una distribución de Dirichlet simétrica con un solo hiperparámetro α tal que $\alpha_1 = \alpha_2 = \ldots = \alpha_K = \alpha$. Al colocar un Dirichlet previa sobre la distribución tópicos $\boldsymbol{\theta}$, el resultado es una distribución tópicos suavizado, con la cantidad de suavizado determinado por el parámetro α.

Buenas opciones para los hiperparámetros $\boldsymbol{\alpha}$ y $\boldsymbol{\beta}$ dependerán del número de tópicos y el tamaño del vocabulario. A partir de investigaciones previas, se ha encontrado que $\alpha = 50 / T$ y $\beta = 0{,}01$ trabajan bien con muchas colecciones de textos diferentes.

Las principales variables de interés en el modelo son las distribuciones de palabras-tópicos $\boldsymbol{\phi}$ y las distribuciones de tópicos-documentos $\boldsymbol{\theta}$. Hofmann (1999) utiliza el algoritmo de expectativa de maximización (EM) para obtener estimaciones directas de $\boldsymbol{\varphi}$ y $\boldsymbol{\theta}$. Este enfoque implica problemas de máximos locales de la función de verosimilitud, lo que ha motivado la búsqueda de mejores algoritmos de estimación (Blei y col., 2003; Buntine, 2002; Minka, 2002). En lugar de estimar directamente las distribuciones palabras-tópicos y tópicos-documentos, otro enfoque es estimar directamente la distribución posterior sobre z dadas las palabras observadas W.

En este trabajo vamos a describir un algoritmo que utiliza el muestreo de Gibbs, una forma de cadena de Markov Monte Carlo, que es fácil de implementar y proporciona un método relativamente eficiente de extraer un conjunto de tópicos de un gran corpus (Griffiths y Steyvers, 2004; véase también Buntine, 2004, 2002 Erosheva y Pritchard et al., 2000). Más información acerca de otros algoritmos para la extracción de los tópicos a partir de un corpus puede obtenerse en las referencias.

El muestreo de Gibbs (también conocida como la alternancia de muestreo condicional), una forma específica de MCMC, simula una distribución multidimensional mediante el muestreo de subconjuntos de variables de dimensiones inferiores, donde cada subconjunto está condicionado al valor de todas las demás. El muestreo se realiza de forma continúa hasta que los valores muestreados se aproximan a la distribución de destino. Si bien el procedimiento de Gibbs que describiremos no proporciona estimaciones directas de $\boldsymbol{\phi}$ y $\boldsymbol{\theta}$, vamos a mostrar cómo $\boldsymbol{\phi}$ y $\boldsymbol{\theta}$ se puede aproximar utilizando estimaciones posteriores de z.

Muestreo de Gibbs (MG)

Describiremos el método de muestreo de Gibbs que es una clase de método de muestreo conocido como cadenas de Markov método de Montecarlo (MCMC).

Para obtener una muestra de $p(Z/W)$ usamos el método de muestreo de Gibbs, para esto necesitamos $p(z_i/Z_{-i}, W)$, que puede escribir como:

$$p(z_i/Z_{-i}, W, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{p(Z, W/\boldsymbol{\alpha}, \boldsymbol{\beta})}{p(Z_{-i} W, \boldsymbol{\alpha}, \boldsymbol{\beta})} \tag{8}$$

Por otro lado, sabemos que,

$$p(Z, W/\boldsymbol{\alpha}, \boldsymbol{\beta}) = p(W/Z, \boldsymbol{\beta}) p(Z/\alpha), \tag{9}$$

y también

$$p(W/Z, \boldsymbol{\beta}) = \int p(W/Z, \boldsymbol{\phi}) p(\boldsymbol{\phi}/\boldsymbol{\beta}) d\boldsymbol{\phi}. \tag{10}$$

Utilizando (4) y (5) tenemos

$$\begin{aligned} p(W/Z, \boldsymbol{\beta}) &= \int \prod_{k=1}^{K} \frac{\Gamma(\sum_{v=1}^{\mathrm{V}} \beta_v)}{\prod_{v=1}^{V} \Gamma(\beta_v)} \prod_{v=1}^{V} \phi_{k,v}^{n_{\circ,k,v}+\beta_v-1} \, d\boldsymbol{\phi} \\ &= \prod_{k=1}^{\mathrm{K}} \left(\frac{\Gamma(\sum_{v=1}^{\mathrm{V}} \beta_v)}{\prod_{v=1}^{V} \Gamma(\beta_v)} \int \prod_{v=1}^{V} \phi_{k,v}^{n_{\circ,k,v}+\beta_v-1} \, d\phi_k \right) \\ &= \prod_{k=1}^{\mathrm{K}} \frac{B(n_{\circ,k,\circ} + \boldsymbol{\beta})}{B(\boldsymbol{\beta})} \end{aligned} \tag{11}$$

Ahora

$$p(Z/\boldsymbol{\alpha}) = \int p(Z/\boldsymbol{\theta}) p(\boldsymbol{\theta}/\boldsymbol{\alpha}) d\boldsymbol{\theta}$$

Utilizando (6) y (7) y (3) tenemos

$$\begin{aligned} p(Z/\boldsymbol{\alpha}) &= \prod_{d=1}^{\mathrm{D}} \left(\int \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} \prod_{k=1}^{K} \theta_{d,k}^{n_{d,k,\circ}+\alpha_k-1} \, d\theta_d \right) \\ &= \prod_{d=1}^{\mathrm{D}} \frac{B(n_{d,\circ,\circ} + \boldsymbol{\alpha})}{B(\boldsymbol{\alpha})} \end{aligned} \tag{12}$$

Sustituyendo (11) y (12) en (9) obtenemos

$$p(Z, W/\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{k=1}^{\mathrm{K}} \frac{B(n_{\circ,k,\circ} + \boldsymbol{\beta})}{B(\boldsymbol{\beta})} \prod_{d=1}^{\mathrm{D}} \frac{B(n_{d,\circ,\circ} + \boldsymbol{\alpha})}{B(\boldsymbol{\alpha})} \tag{13}$$

Como z_i depende solo de w_i, la expresión (8) se puede escribir como

$$p(z_i/Z_{-i}, W, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto \frac{p(Z, W/\boldsymbol{\alpha}, \boldsymbol{\beta})}{p(Z_{-i}, W_{-i} / \boldsymbol{\alpha}, \boldsymbol{\beta})}$$

Pero sabemos que:

$$p(Z_{-i}, W_{-i}/\boldsymbol{\alpha}, \boldsymbol{\beta}) = p\big(W_{-i}/Z_{-i}, \boldsymbol{\beta}\big)p(Z_{-i}/\boldsymbol{\alpha})$$

Procediendo de la misma forma que se obtuvo (13) tenemos

$$p(Z_{-i}, W_{-i}/\boldsymbol{\alpha}, \boldsymbol{\beta}) = \prod_{k=1}^{\mathrm{K}} \frac{B({n_{\circ,k,\circ}}^{-i} + \boldsymbol{\beta})}{B(\boldsymbol{\beta})} \prod_{d=1}^{\mathrm{D}} \frac{B({n_{d,\circ,\circ}}^{-i} + \boldsymbol{\alpha})}{B(\boldsymbol{\alpha})} \tag{14}$$

donde ${n_{\circ,k,v}}^{-i}$ es el número de veces que el termino v es asignado al tópico k, pero con la i-ésima palabra y su tópico asignado excluidos, y ${n_{d,k,\circ}}^{-i}$ es el número de veces que el tópico k es asignado al documento d excepto la i-ésima palabra y su tópico asignado.

Las siguientes expresiones son verdaderas

$$n_{\circ,k,v} = \begin{cases} {n_{\circ,k,v}}^{-i} + 1 & si \quad v = w_i \ \wedge \ k = k_i \\ {n_{\circ,k,v}}^{-i} & en\ los\ demas\ casos \end{cases} \tag{15}$$

$$n_{d,k,\circ} = \begin{cases} {n_{d,k,\circ}}^{-i} + 1 & si \quad d = d_i \ \wedge \ k = k_i \\ {n_{d,k,\circ}}^{-i} & en\ los\ demas\ casos \end{cases} \tag{16}$$

Finalmente usando (13),(14),(15) y (16) la expresión (8) se puede escribir como

$$p(z_i/Z_{-i}, W, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{B(n_{\circ,k,\circ} + \boldsymbol{\beta})}{B({n_{\circ,k,\circ}}^{-i} + \boldsymbol{\beta})} \frac{B(n_{d,\circ,\circ} + \boldsymbol{\alpha})}{B({n_{d,\circ,\circ}}^{-i} + \boldsymbol{\alpha})}.$$

Entonces

$$p(z_i/Z_{-i}, W, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\frac{\prod_{v=1}^{V} \Gamma(n_{\circ,k,v} + \beta_{\boldsymbol{v}})}{\Gamma(\sum_{v=1}^{V} n_{\circ,k,v} + \beta_{\boldsymbol{v}})}}{\frac{\prod_{v=1}^{V} \Gamma({n_{\circ,k,v}}^{-i} + \beta_{\boldsymbol{v}})}{\Gamma(\sum_{v=1}^{V} {n_{\circ,k,v}}^{-i} + \beta_{\boldsymbol{v}})}} \frac{\frac{\prod_{k=1}^{K} \Gamma(n_{d,k,\circ} + \alpha_{\boldsymbol{k}})}{\Gamma(\sum_{k=1}^{K} n_{d,k,\circ} + \alpha_{\boldsymbol{k}})}}{\frac{\prod_{k=1}^{K} \Gamma({n_{d,k,\circ}}^{-i} + \beta_{\boldsymbol{k}})}{\Gamma(\sum_{k=1}^{K} {n_{d,k,\circ}}^{-i} + \beta_{\boldsymbol{k}})}}$$

$$p(z_i = k/Z_{-i}, w_i = v, W_{-i}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{\frac{\Gamma(n_{\circ,k,v} + \beta_{\boldsymbol{w_i}})}{\Gamma(\sum_{v=1}^{V} n_{\circ,k,v} + \beta_{\boldsymbol{v}})}}{\frac{\Gamma({n_{\circ,k,v}}^{-i} + \beta_{\boldsymbol{w_i}})}{\Gamma(\sum_{v=1}^{V} {n_{\circ,k,v}}^{-i} + \beta_{\boldsymbol{v}})}} \frac{\frac{\Gamma(n_{d,k,\circ} + \alpha_{z_i})}{\Gamma(\sum_{k=1}^{K} n_{d,k,\circ} + \alpha_{\boldsymbol{k}})}}{\frac{\Gamma({n_{d,k,\circ}}^{-i} + \alpha_{z_i})}{\Gamma(\sum_{k=1}^{K} {n_{d,k,\circ}}^{-i} + \alpha_{\boldsymbol{k}})}}$$

Ahora las siguientes igualdades son verdaderas

$$\sum_{v=1}^{V} n_{\circ,k,v} = 1 + \sum_{v=1}^{V} {n_{\circ,k,v}}^{-i} \tag{17}$$

$$\sum_{k=1}^{K} n_{d,k,\circ} = 1 + \sum_{k=1}^{K} n_{d,k,\circ}{}^{-i} \tag{18}$$

De (17) y de (18) obtenemos

$$p(z_i = k/Z_{-i}, w = v, W_{-i}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{n_{\circ,k,v} + \beta_v - 1}{\left[\sum_{v=1}^{V} n_{\circ,k,v} + \beta_t\right] - 1} \cdot \frac{n_{d,k,\circ} + \alpha_k - 1}{\left[\sum_{k=1}^{K} n_{d,k,\circ} + \alpha_k\right] - 1}$$

Además la estimación de Bayes de los parámetros de la función de distribución posterior son,

$$\hat{\theta}_{d,k} = \frac{\alpha_k + n_{d,k,\circ}}{\sum_{k=1}^{K}(\alpha_k + n_{d,k,\circ})}$$

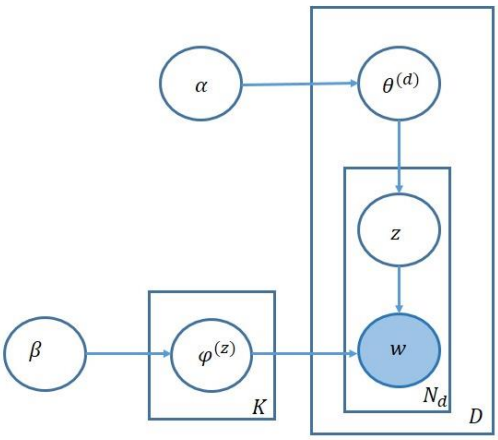
$$\hat{\phi}_{k,v} = \frac{\beta_v + n_{\circ,k,v}}{\sum_{v=1}^{V}(\beta_v + n_{\circ,k,v})}$$

Ahora con esta aproximación de los parámetros podemos calcular la entropía

$$\log(entropía) = -\frac{\sum_{d=1}^{D}\left(\sum_{v=1}^{V} n_{d,\circ,v}\, log\{\sum_{k=1}^{K} \hat{\theta}_{d,k}\, \hat{\phi}_{k,v}\}\right)}{\sum_{m=1}^{D} N_d}.$$

Modelo Gráfico

Los modelos probabilísticos generativos se pueden ilustrar usando la notación de placas como se ilustra en la siguiente figura (Griffiths y Steyvers). En esta notación gráfica, las variables sombreadas indican que son observadas en cambio las no sombreadas indican variables latentes, es decir, variables no observables. Las variables ϕ y θ, así como z son los tres conjuntos de variables latentes que nos gustaría inferir. Las flechas indican dependencias condicionales entre las variables mientras que las placas (las cajas en la figura) con la variable en la esquina inferior derecha se refieren a la repetición de los pasos de muestreo.



La asignación de tópicos a palabras de un corpus. Cada palabra en un documento es asignada a un tópico como se muestra en la siguiente tabla.

	Pal_1	Pal _2	Pal _3	Pal _4	Pal _5	Pal _6
Doc_1	Tóp_2		Tóp_3	Tóp_2	Tóp_3	Tóp_1
Doc_2	Tóp_1	Tóp_3	Tóp_1	Tóp_4	Tóp_1	Tóp_2
Doc_3	Tóp_2	Tóp_4	Tóp_4	Tóp_3		Tóp_1

Nota: En esta tabla puede haber palabras repetidas.

Capítulo 4 Sistema desarrollado en Computo Paralelo

Unos de los objetivos de la tesis, es desarrollar el sistema LDA y que se ejecute de forma paralela, es decir, que este programa utilice varios procesadores y cores, para que reduzca considerablemente su tiempo de ejecución.

Para poder llegar a esto, primero debemos aclarar las diferencias entre un algoritmo serial y un algoritmo paralelo, para posteriormente hablar de cómo fue desarrollado este sistema.

Algoritmo

Primero mostraremos la definición de *algoritmo* según la Real Academia Española (Real Academia Española, 2017):

"Conjunto ordenado y finito de operaciones que permite hallar la solución de un problema."

Entonces, un ejemplo de algoritmo es el siguiente:

```
INICIO
      escribir "Proporciona valor de n"
      leer n
      si n es menor a 1 entonces
            escribir "Error, n debe ser >= 1"
      si no
            factorial ← 1
            para i ← 2 hasta n hacer
                  factorial ← factorial * i
            fin para
            escribir n, "!=", factorial
      fin si
FIN
```

El cual, simplemente calcula el factorial de un número dado, y a pesar de ser un algoritmo sencillo, cumple con la descripción de la definición antes dada.

Algoritmo Secuencial

Un algoritmo secuencial es aquel cuyas componentes, que llamaremos tareas, deben ser ejecutadas en serie, una tras otra. Esto debido a que los datos de entrada de una tarea depende de los datos de salida de la tarea anterior, excepto para la primer tarea, o simplemente el algoritmo fue diseñado de forma secuencial.

Un ejemplo de un algoritmo secuencial es la serie de Fibonacci, la cual tiene el siguiente comportamiento:

$$0 \rightarrow 1 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 13 \rightarrow 21 \ldots$$

o para hacer más claro su comportamiento:

$$a = 0 \rightarrow b = 1 \rightarrow c = a + b \rightarrow d = b + c \rightarrow e = c + d \ldots$$

Se puede notar claramente que una vez definidos los dos primeros valores de la serie, el comportamiento es estrictamente secuencial, ya que para generar el siguiente valor, este depende de los últimos dos valores de la serie.

Algoritmo Paralelo

Un algoritmo paralelo es aquel cuyas tareas no necesitan ejecutarse en serie, es decir, un par de tareas o más, pueden ejecutarse al mismo tiempo.

En la mejor paralelización, debe existir una independencia de datos entre las tareas, es decir, una tarea A no necesita información de la tarea B para realizar su trabajo, y viceversa. Un ejemplo claro de esto, lo vemos a diario en nuestros ordenadores o en nuestros dispositivos móviles (smartphones) donde podemos tener abiertas varias aplicaciones y juegos de manera simultánea, sin que una aplicación interfiera en el funcionamiento de otra.

Sin embargo, existen casos en los que las tareas a pesar de poder ejecutarse de forma paralela, necesitan comunicarse entre ellas, ya sea por paso de mensajes o porque comparten variables, por lo tanto, debe existir un mecanismo de sincronización para acceder a estas variables.

Algoritmo Serial-Paralelo (SPA)

Fayes Gebali (Gebali, 2011), dice que en un SPA las tareas son agrupadas en etapas, donde cada tarea de una etapa puede ser ejecutada en paralelo y las etapas son ejecutadas secuencialmente.

Sin embargo, un SPA llega a ser un algoritmo paralelo cuando el número de etapas es una, así como, un SPA llega a ser un algoritmo secuencial cuando el número de tareas en cada etapa es únicamente una.

En la Fig. 7 se muestra un ejemplo de un SPA, donde cada etapa puede tener una o más tareas

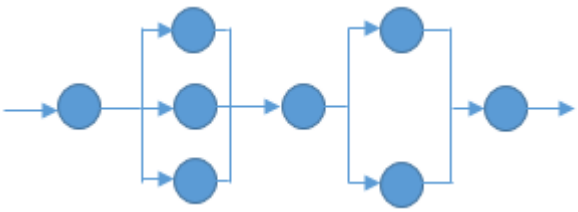


Fig. 7 Ejemplo de Algoritmo Serial-Paralelo (SPA) con 5 etapas.

El autor F. Gebali, también hace mención de algoritmos No Seriales-Paralelos (NSPA) y de RIAs, sin embargo, en este documento no hablaremos de estos últimos dos.

Arquitectura de Memorias para Computo Paralelo

Para desarrollar un programa en paralelo, debemos conocer la arquitectura de memoria de la computadora en donde ejecutaremos nuestro programa, así mismo, debemos tener en cuenta las necesidades y características de nuestro sistema para usar el concepto adecuado.

Es por esto que se presentan tres arquitecturas de memoria que se usan en las computadoras, las cuales son memoria compartida, memoria distribuida y memoria hibrida (Barney, Introduction to Parallel Computing, 2017), (Gebali, 2011), (Pacheco, 2011), (Petersen & Arbenz, 2004),

Memoria Compartida

Las computadoras con memoria compartida tienen por lo general las siguientes características:

- Tienen dos o más procesadores.
- Los procesadores pueden acceder a toda la memoria como espacio de direcciones global.
- Los cambios efectuados en memoria por un procesador son visibles a los otros procesadores.
- Los procesadores pueden trabajar independientemente.
- Los procesadores son iguales.
- El tiempo de acceso a la memoria es el mismo para todos los procesadores.

Este tipo de arquitectura suele ser llamada Memoria de Acceso Uniforme (UMA), la Fig. 8 es un ejemplo de este tipo.

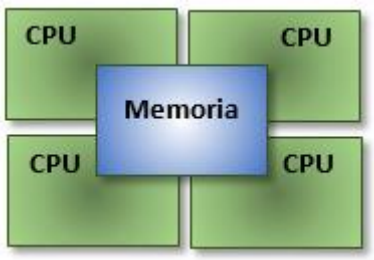


Fig. 8 Arquitectura de Memoria Compartida, o Memoria de Acceso Uniforme (UMA)

Así mismo, existe dentro del concepto de Memoria Compartida el tipo de Memoria de Acceso No Uniforme (NUMA), la cual tiene una arquitectura muy parecida a la de Memoria Distribuida (Fig. 9), pero con la diferencia de que en NUMA los procesadores pueden acceder a la memoria local de los otros procesadores de forma directa, por lo que, el

concepto de NUMA reside en la diferencia de tiempo que le toma a un procesador escribir en su memoria local y el tiempo que le toma escribir en una memoria de otro procesador.

Memoria Distribuida

Los computadores paralelos de memoria distribuida tienen por lo general las siguientes características:

- Dos o más procesadores independientes.
- Cada procesador tiene su propia memoria local.
- Requieren una red de comunicación para interconectar los procesadores y su memoria.
- La comunicación entre procesadores se hace por Paso de Mensajes.

La figura 9 es un ejemplo de este tipo.

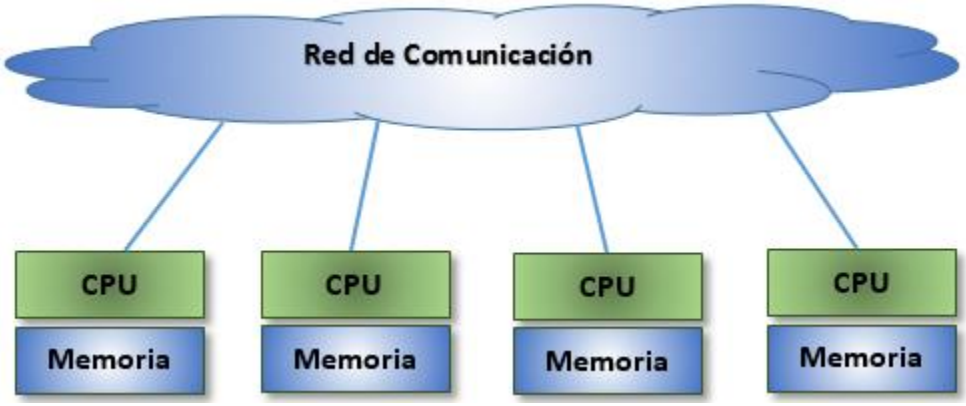


Fig. 9 Arquitectura de Memoria Distribuida

Memoria Hibrida

En esta arquitectura, se usan ambos tipos de memoria mencionados arriba, es decir, computadores de memoria compartida son conectados en red, y además, en esta arquitectura podemos encontrar Unidades de Procesamiento Grafico (GPU) junto a los CPU. En la Fig. 10, podemos ver un ejemplo de este tipo de arquitectura.

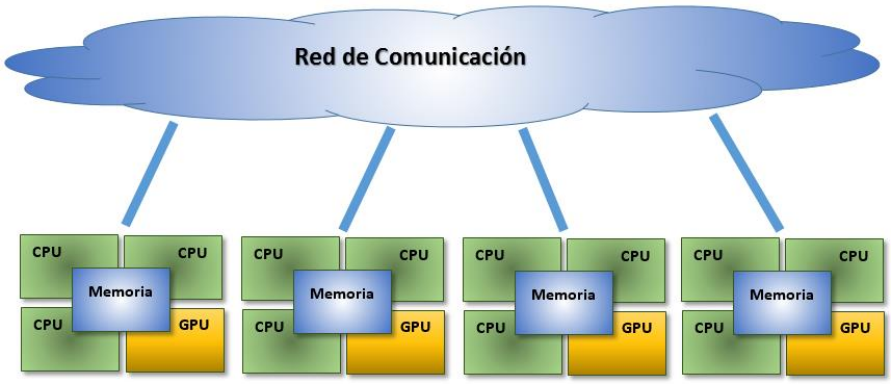


Fig. 10 Arquitectura de Memoria Hibrida

Este tipo de arquitectura es usado en varias de las supercomputadoras más grandes del mundo (Top500.org, 2017), las cuales usan computadores con múltiples procesadores interconectados en red, además de que algunos de estos supercomputadores también cuentan con GPU's.

Granularidad

La granularidad es muy importante al momento de dividir el trabajo en n hilos/procesos. La granularidad suele ser dividida en tres categorías, las cuales se ilustrarán con un ejemplo sencillo.

Supongamos que se tiene una matriz, de tamaño 50×50, llena con número enteros. Se quiere sumar una unidad a cada elemento de la matriz. Por lo que tenemos:

Paralelismo de grano fino: En esta categoría podríamos usar 2500 procesadores, de modo que cada procesador actualizaría una sola casilla de la matriz, y teóricamente, el programa finalizaría en un ciclo de reloj.

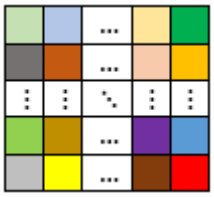


Fig. 11 Paralelismo de grano fino, cada color representa un procesador

Paralelismo de grano medio: en esta categoría se usarían 50 procesadores, de modo que cada procesador actualizaría 50 elementos, por lo tanto el programa finalizaría en 50 ciclos de reloj.

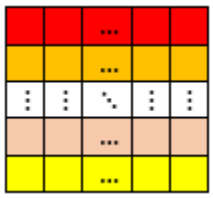


Fig. 12 Paralelismo de grano medio, cada color representa un procesador

Paralelismo de grano grueso: En esta categoría se usa una cantidad mínima de procesadores, supongamos dos, por lo que cada procesador actualizaría 1250 elementos de la matriz, y tomaría 1250 ciclos de reloj para finalizar el programa.

Fig. 13 Paralelismo de grano grueso, cada color representa un procesador.

Lenguajes de Programación

Hoy en día existe una amplia variedad de lenguajes de programación que sirven para hacer computo paralelo o concurrente, entre ellos destacan C/C++, Java, Python, Julia, entre otros.

Sin embargo, lo que se busca para la implementación del sistema LDA, es un lenguaje de alto desempeño. Y se encontró en la página de julia (Bezanson, 2017) la gráfica mostrada en la Fig. 14, en donde se han realizado diferentes "benchmarks", con diferentes lenguajes de programación y se comparan los resultados contra C, donde se puede apreciar claramente que C es superado muy pocas veces en cuanto a su desempeño, y que los lenguajes de programación que más se aproximan al desempeño general de C son Julia y Go.

Pero los lenguajes de programación al solo aproximarse al desempeño de C. Se decidió realizar la programación del sistema LDA en el lenguaje C/C++.

Fig. 14 Tiempos de pruebas comparativas en relación a C (menos es mejor, desempeño de C = 1.0)

Interfaz de Paso de Mensajes (MPI)

MPI está enfocado en el modelo de programación paralelo por paso de mensajes, esto es, datos de un proceso pueden ser copiados a otro proceso a través de operaciones cooperativas en cada proceso.

MPI está desarrollada para funcionar en prácticamente cualquier plataforma de hardware, esto es de:

- Memoria Distribuida
- Memoria Compartida
- Memoria Hibrida

Al funcionar en plataformas de memoria hibrida, es muy útil a la hora de aprovechar los recursos de alguna supercomputadora que posee varios nodos, ya que a cada nodo se le asignará una parte del trabajo total de la tarea.

Sin embargo, MPI no está diseñado para usar variables compartidas entre los procesos, por lo que toda la comunicación debe hacerse por paso de mensajes. Además, si se usan demasiadas operaciones de comunicación entre los procesos, se puede ocasionar un cuello de botella en la red que interconecta a los nodos y por lo tanto al desempeño del mismo sistema. Por lo que de manera personal, recomendamos el uso de MPI, intentando usar la menor cantidad de comunicaciones entre los procesos.

Actualmente, MPI se encuentra en su versión 3.1, si se desea conocer la documentación completa de MPI y todas sus versiones, puede consultarse (MPI Forum, 2017). También se pone a disposición un tutorial del LLNL para MPI (Barney, Message Passing Interface (MPI), 2017), estas dos referencias enfocan el uso de MPI con C/C++. Además, buscando en la web se pueden encontrar tutoriales de su uso en otros lenguajes de programación como son Python, Julia, Fortran, etc.

OpenMP

Es el estándar actual para programar aplicaciones paralelas en sistemas de memoria compartida tipo Multiproceso Simétrico (SMP).

OpenMP no es un lenguaje de programación, sino que es una Interfaz de Programación de Aplicaciones (API). Esta API hace uso de directivas para el compilador, pero en entornos que no usan OpenMP las directivas son tratadas como comentarios, por lo cual son ignoradas.

Los lenguajes de programación en los que trabaja principalmente OpenMP son C/C++ y Fortran. OpenMP permite paralelizar aplicaciones escritas en estos lenguajes, sin la necesidad de hacer cambios significativos al código existente.

El uso de OpenMP consiste en colocar las directivas en la parte superior del código que se quiere paralelizar. La tarea será dividida en varios hilos y posteriormente se recolectaran sus resultados en uno sólo (Master). OpenMP por default crea tantos hilos como numero de núcleos con los que cuenta el computador, sin embargo, este número puede ser modificado. Usar un número de hilos muy grande no significa que la tarea se realizará de forma más rápida, ya que cada procesador puede ejecutar simultáneamente una cantidad limitada de hilos, y se debe tener claro el concepto de granularidad para la división de trabajo.

El modelo de programación paralela que usa OpenMP es Fork-Join, Fork para crear los hilos, también llamados threads, y Join para recuperarlos, como se puede observar en la figura 15.

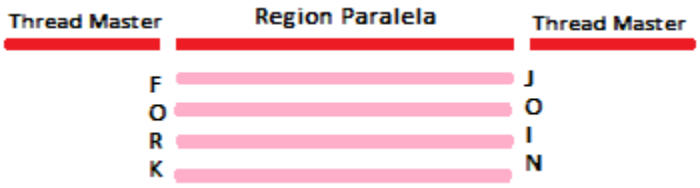


Fig. 15 Ejemplo del funcionamiento de OpenMP

Esto es, todos los hilos ejecutan la misma copia de código. Además, cada hilo recibe un identificador, que sirve para asignar tareas específicas a un hilo específico.

A diferencia de MPI, OpenMP está desarrollado para funcionar en plataformas de memoria compartida, lo cual significa que por sí mismo, no podrá hacer uso de más de un nodo. Pero a diferencia de MPI, en OpenMP sí se pueden usar variables compartidas, lo cual es una gran ventaja, debido a que no se tienen que hacer copias de todas las variables para cada hilo, además, todos los hilos podrán leer y escribir en las variables compartidas, aunque hay que definir un método de acceso a estas variables, principalmente cuando varios procesos quieren actualizar el valor de estas variables (para ello existen los candados y/o semáforos). Si se desea conocer más acerca de OpenMP, remitimos a las referencias (Pacheco, 2011), (Software Intel, 2017) y (Barney, OpenMP, 2015).

De la programación del sistema LDA

Para la programación del sistema LDA, se ha usado un algoritmo del tipo Serial-Paralelo (SPA), el cual, se puede ver gráficamente en la Fig. 16.

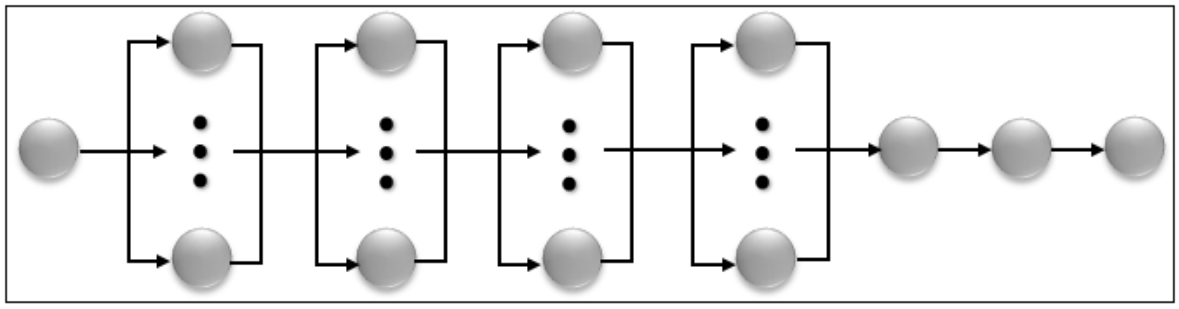
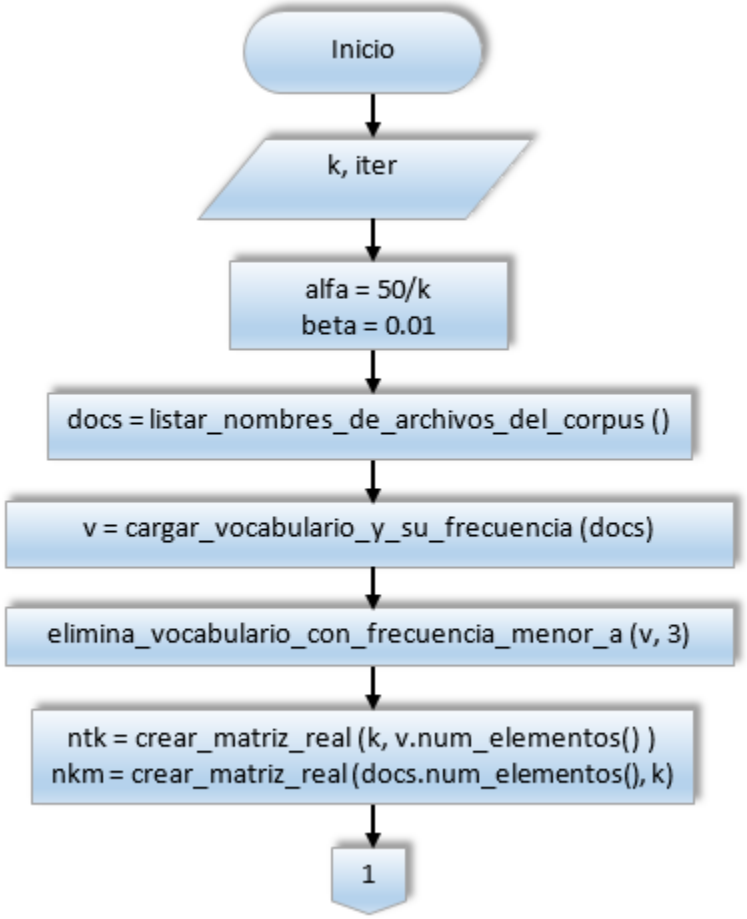


Fig. 16 Representación gráfica del algoritmo SPA para el sistema LDA, el cual consta de 8 etapas.

El lenguaje de programación es C/C++, por motivos especificados arriba. Para poder llevar a cabo la programación paralela, se ha decido utilizar OpenMP, esto debido a que en las etapas paralelas, todos los hilos creados, deben compartir ciertas variables, y el hecho de que en MPI esto no sea posible, hace que lo descartemos inmediatamente. Por lo tanto, la arquitectura de memoria que se ocupo fue la de memoria compartida.

Ahora, se explicará lo que sucede en cada una de las ocho etapas de nuestro algoritmo SPA para el sistema LDA.

Etapa 1, Inicialización



En esta etapa, es llevada a cabo la inicialización del sistema, es decir, aquí es definida la cantidad de tópicos para el corpus (*k*), es definida la cantidad de iteraciones (*iter*) para el sistema LDA y son calculados los valores *alfa-beta* de la ecuación (num. ecuación).

Una vez hecho esto, deben ser leídos y almacenados, los nombres de los archivos que pertenecen al corpus, para después, cargar todo el vocabulario junto con su frecuencia en un diccionario. Cuando esta creado el diccionario, son eliminadas todas las palabras cuya frecuencia en el corpus sea menor a tres. Esto es, las palabras con una frecuencia muy reducida no pueden definir tópicos, además, solo ocasionarían ruido en los resultados finales.

Conociendo, la cantidad de archivos que pertenecen al corpus, y la cantidad de palabras en el diccionario, son creadas dos matrices de números reales, la primera representa tópicos por vocabulario (que llamaremos ntk), y la segunda representa a los documentos por tópicos (que llamaremos nkm).

Se debe tener especial cuidado en esta parte, debido a que las matrices creadas pueden llegar a ser muy grandes, supongamos la siguiente situación, se escogen 1,000 topicos y el

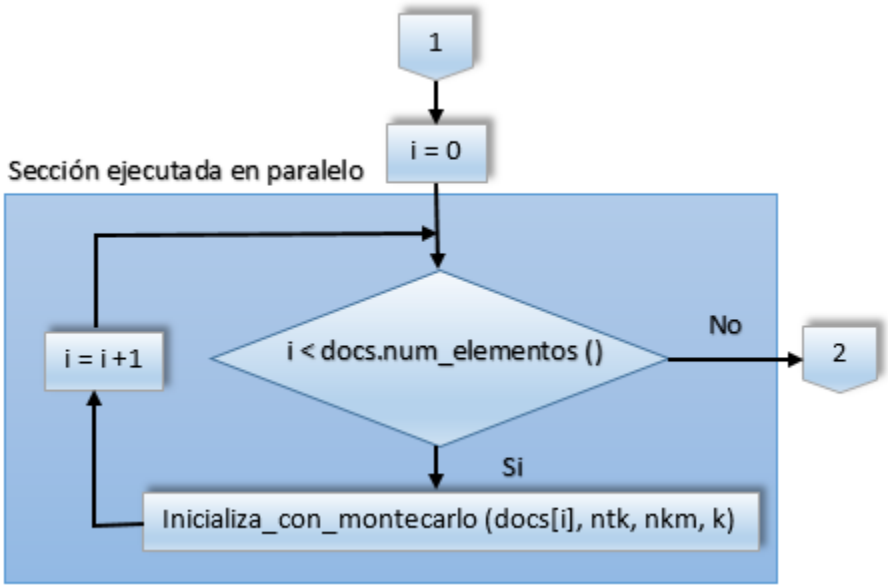
vocabulario con frecuencia mayor o igual a 3 tiene un tamaño de 500,000 palabras, y se tiene un total de 100,000 archivos, y al ser reales las matrices se ocupa como tipo de dato el *double*.

Se puede usar la siguiente fórmula para tener una idea del espacio en memoria RAM para poder ejecutar el sistema (la formula da el resultado en Gigabytes):

$$mem = \frac{(tamaño\_vocabulario + num\_archivos) * k * 8}{1024^3}$$

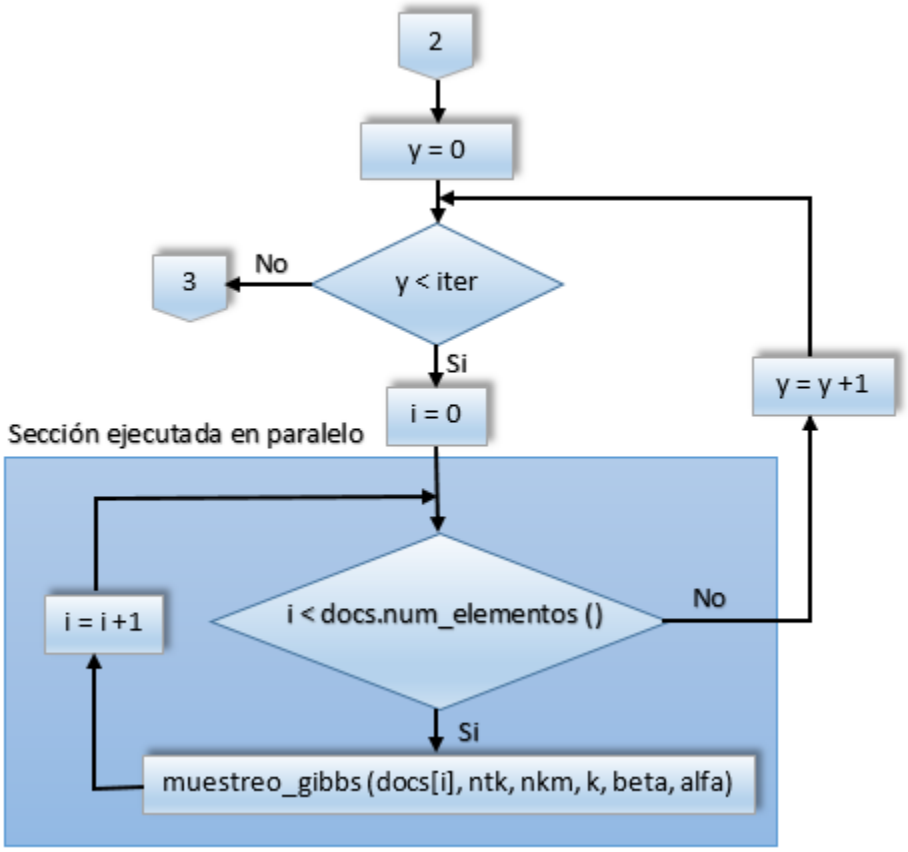
Por lo que para esta situación son necesarios 4.47 Gigabytes. Esta fórmula es solo como referencia para la creación de estas matrices, ya que en las etapas paralelas cada hilo ocupa ciertas variables para desempeñar su trabajo, por lo cual se recomienda que la memoria RAM sea de al menos el doble del resultado de *mem.*

Etapa 2, Primera iteración



Como el nombre de esta etapa lo indica, esta etapa está basada en la primera iteración sobre el corpus. Para llevar a cabo la paralización, se reparten los archivos entre los hilos, y las matrices ntk y nkm son variables compartidas entre estos. Por lo tanto, las matrices son inicializadas con el Método de Montecarlo por todos los hilos.

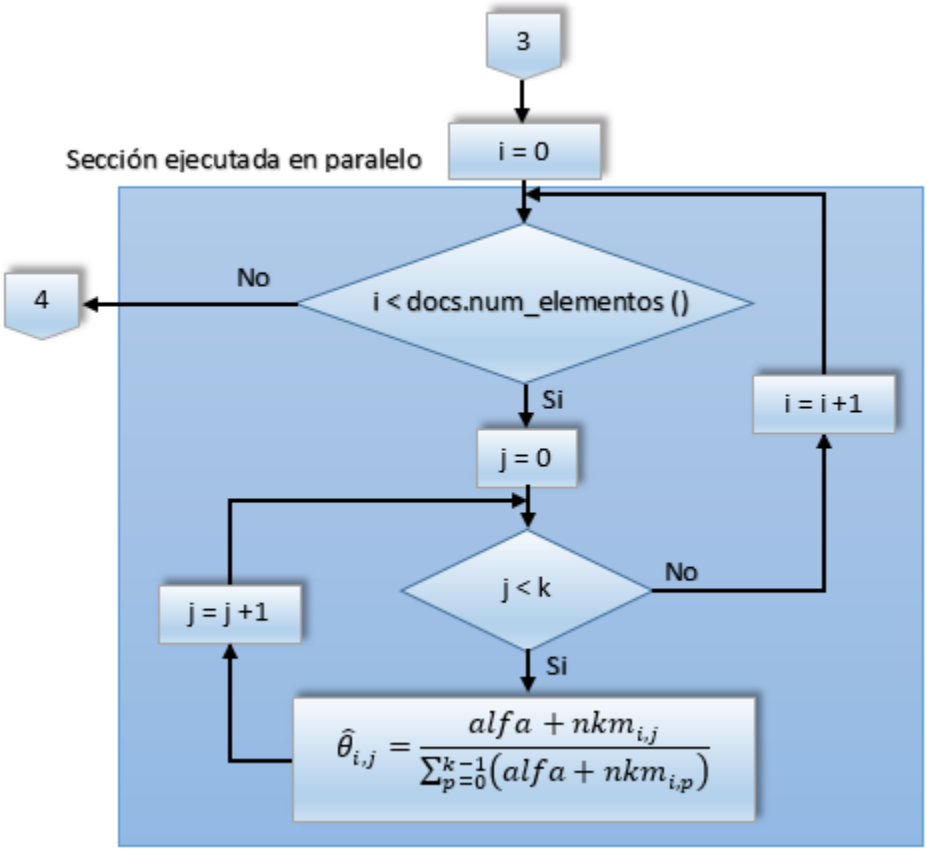
Etapa 3, Iteraciones



Esta etapa es quizá la más importante, la causante de que se desarrollara el sistema de forma paralela, esto debido a que según el número de iteraciones, es el número de veces que se tiene que abrir cada archivo, es decir, si se tiene un corpus con 100,000 archivos y se quieren hacer 1000 iteraciones, entonces se tendrían que abrir un total de 100, 000,000 archivos, lo cual, no es una cantidad pequeña.

Por lo cual, de manera similar a la etapa 1, en cada iteración, los archivos son divididos entre todos los hilos, por lo que las matrices *ntk* y *nkm*, vuelven a ser compartidas, además de *k*, *beta*, *alfa*, y se hace la actualización de las dos matrices usando el muestreo de Gibbs.

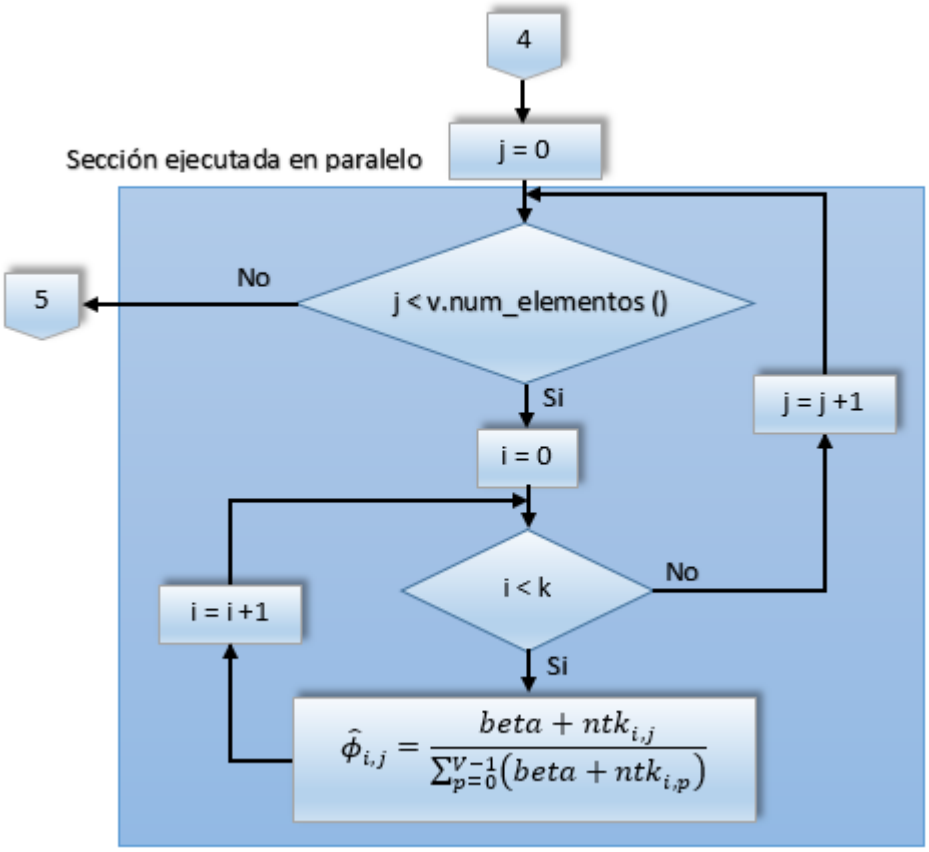
Etapa 4, Creación de Theta



Una vez terminada la parte pesada del sistema, es calculada theta. Al finalizar los cálculos, theta contendrá las probabilidades, de cada tópico, de pertenecer a cada documento.

En esta etapa la paralelización es posible, repartiendo lo que son las filas de la matriz theta entre los hilos.

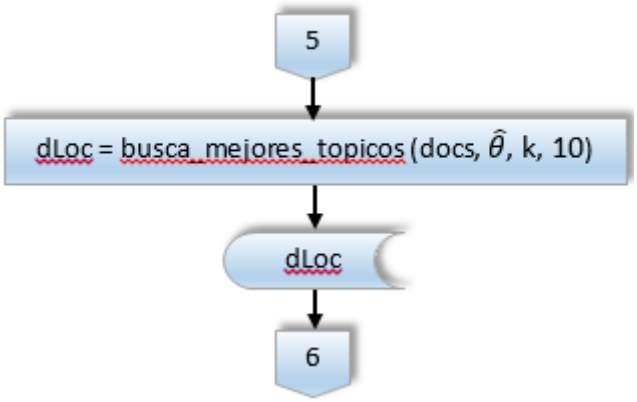
Etapa 5, Creación Phi



La etapa 4, es muy similar a la etapa 3, con la diferencia que ahora phi, contendrá las probabilidades de cada palabra, de pertenecer a cada tópico.

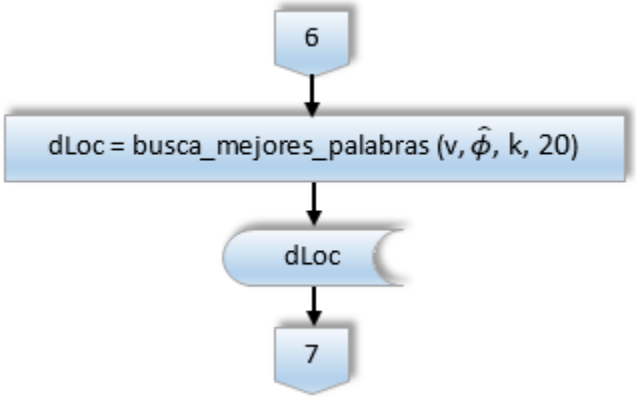
De igual manera, la paralelización es posible dividiendo las filas de phi entre los hilos

Etapa 6, Tópicos más representativos de un documento



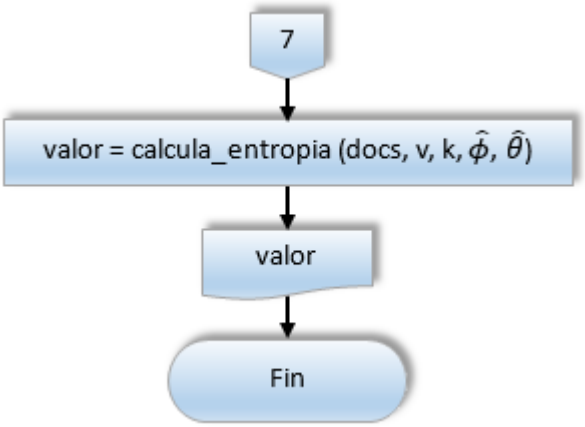
En este punto del programa, solo nos interesan las palabras que mejor describan a cada documento/archivo, por lo que en esta parte, son buscados en theta los 10 tópicos con las probabilidades más altas para cada archivo. Una vez encontrados son almacenados en un fichero de texto.

Etapa 7, Palabras más representativas de un Tópico



Similar a la etapa 5, en esta etapa nos interesa buscar las palabras que mejor representen a los tópicos, por lo que en phi son buscadas, para cada tópico, las 20 palabras con probabilidades más altas. Una vez encontradas, son almacenadas en un fichero de texto.

Etapa 8, Calculo de entropía



Como un extra, se ha agregado esta etapa, que nos ayuda a medir la entropía de los resultados. Esta parte sirve para identificar una cantidad optima de tópicos, es decir, en un inicio podemos no saber la cantidad exacta de tópicos que contiene el corpus, pero después de ejecutar el sistema con diferentes cantidades de tópicos, se puede construir una gráfica con el resultado de esta etapa, y en base a la gráfica se podrá observar la cantidad optima de tópicos para el corpus.

De la ejecución del sistema LDA

Corpus

Primero se dará la definición de *corpus* según la Real Academia Española (Real Academia Española, 2017):

"Conjunto lo más extenso y ordenado posible de datos o textos científicos, literarios, etc., que pueden servir de base a una investigación."

Ahora que se sabe que un corpus es una colección de documentos, se mencionará al corpus que se analizó con el sistema LDA. El corpus que se usó como base es el Wikicorpus en Español (Reese, Boleda, Cuadros, Padró, & Rigau, 2010), el cual contiene alrededor de 120 millones de palabras.

Pre-procesamiento de Datos

Esta parte es fundamental en la ejecución de este sistema, ya que es donde se "limpian" los datos de entrada para que se obtengan buenos resultados.

Abella y Medina (Abella & Medina, 2014), hablan un poco del pre-procesamiento de datos, de los cuales destacan:

- Eliminación de signos de puntuación.
- Reducción de mayúsculas.
- Eliminación de etiquetas, si proviene de un documento tipo XML.
- Eliminación de *palabras vacías*, son aquellas que no aportan un significado al texto, un ejemplo de estas son los artículos, preposiciones, conjunciones, etc.
- Extracción de raíces o lemas.

Teniendo en mente este pre-procesamiento, se encontró que la Universidad Politécnica de Cataluña y la Universidad Pompeu Fabra (Universitat Politècnica de Catalunya. Research Group on Natural Language Processing; Gemma Boleda; Universitat Pompeu Fabra. Institut Universitari de Lingüística Aplicada (IULA), 2012), han procesado el wikicorpus, este grupo a lematizado y etiquetado el corpus, de modo que el contenido de sus archivos tienen la siguiente forma:

```
37      TOK     Doña            Doña\JQ—FS
##      TAG     <name>
38      TOK     Rosita          Rosita\N4666
##      TAG     </name>
39      TOK     la              el\AFS
40      TOK     soltera         soltero\N5-FS
41      TOK     o               o\C
42      TOK     el              el\AMS
43      TOK     lenguaje                lenguaje\N5-MS
44      TOK     de              de\P
```

45	TOK	las		el\AFP
46	TOK	flores		flor\N5-FP
---	DLD	.	EOS	=\DELIM
##	TAG	</s>		
##	TAG	<s>		
47	TOK	Tras	BOS	tras\P
48	TOK	esta		este\ED--FS
49	TOK	obra		obra\N5-FS

Sin embargo, estos archivos aun no pueden ser usados en el sistema LDA. Por lo que el trabajo que se realizó fue, identificar aquellas palabras que tengan la etiqueta *TOK*, se guardó la parte lematizada, posteriormente se hizo la reducción a minúsculas y finalmente se eliminaron las *palabras vacías*.

De modo que los archivos resultantes se ven de la siguiente manera:

```
doña
rosita
soltero
lenguaje
flor
obra
```

Una vez hecho este pre-procesamiento sobre el corpus de (Universitat Politècnica de Catalunya. Research Group on Natural Language Processing; Gemma Boleda; Universitat Pompeu Fabra. Institut Universitari de Lingüística Aplicada (IULA), 2012), el corpus final servirá para el sistema LDA.

Debido a limitaciones en el comienzo del proyecto, se ha usado una fracción de este corpus, lo que corresponde a alrededor de 195,000 archivos, los que han sido procesados.

Ejecución

El sistema LDA ha sido ejecutado en el Laboratorio Nacional de Supercomputo del Sureste de México (LNS, 2017) en los nodos normales, donde cada nodo tiene 2 procesadores Intel Xeon E5-2680 v3, y cada procesador tiene 12 cores con una frecuencia de 2.5 GHz, además cuentan con 126 Gigabytes de memoria RAM.

Recordando que el sistema ha sido programado con el Lenguaje C++ y con la API de OpenMP, en cada etapa paralela son creados 24 hilos, con la intención de que cada hilo sea ejecutado por un core diferente.

Resultados del sistema LDA

El sistema LDA fue ejecutado con 300 iteraciones y con una cantidad variable de tópicos, que va desde los 100 hasta los 4000 tópicos, los resultados pueden ser vistos en la siguiente tabla:

Tópicos	Entropía	Diferencia entre tópicos
100	9.11148	
200	9.01459	0.09689
300	8.94128	0.07331
400	8.88203	0.05925
500	8.83977	0.04226
600	8.79597	0.0438
700	8.76163	0.03434
800	8.72798	0.03365
900	8.704	0.02398
1000	8.67401	0.02999
1100	8.66157	0.01244
1200	8.63675	0.02482
1300	8.61248	0.02427
1400	8.59993	0.01255
1500	8.58606	0.01387
1600	8.57487	0.01119
1700	8.55774	0.01713
1800	8.54389	0.01385
1900	8.52989	0.014
2000	8.51713	0.01276
3000	8.43023	0.0869
4000	8.36932	0.06091

Con estos datos se obtuvo la gráfica de la figura 11, donde se puede ver tanto en la tabla como en la gráfica, que después de los 1300 tópicos, la entropía empieza a decrecer muy poco. Por lo que se escogió como cantidad de tópicos optima a 1300.

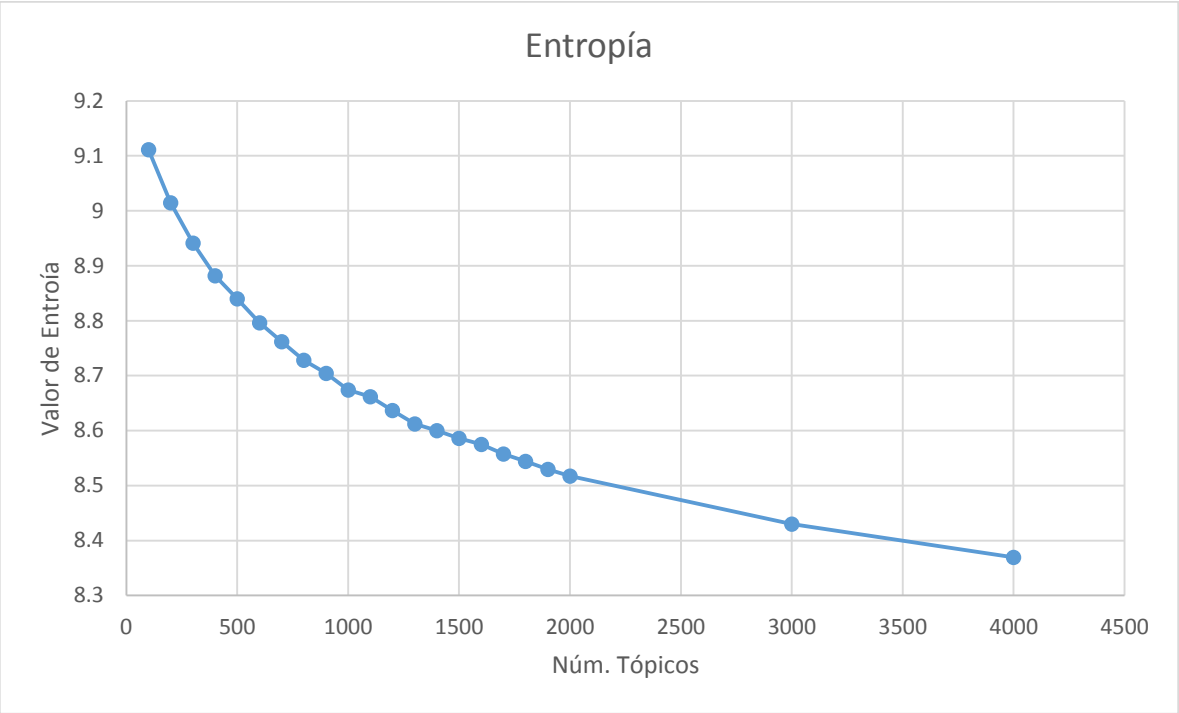


Fig. 17 Gráfica de la Entropía de cada ejecución

Una vez escogido el valor de 1300, se ha ejecutado nuevamente el sistema LDA, solo que esta vez se han hecho 1000 iteraciones.

A continuación, se mostraran algunos tópicos, con sus palabras y sus respectivas probabilidades en el tópico, generados por el sistema LDA para 1300 tópicos con 1000 iteraciones:

Tópico 73		Tópico 89		Tópico 91	
olímpico	0.069787	ejército	0.00971551	jugar	0.0134707
medalla	0.0509437	batalla	0.00750859	partido	0.0128869
juegos	0.0444426	recluta	0.00712142	equipo	0.0126209
ganar	0.0314628	eure	0.00704299	club	0.0113091
argentina	0.0311572	guerrillero	0.00677679	gol	0.010086
oro	0.0181991	a.c	0.00621277	primero	0.00910095
equipo	0.0172583	armar	0.00617381	méndez	0.00860453
obtener	0.0172258	Ávila	0.00583231	fútbol	0.00857132
argentino	0.0139198	no	0.00577099	año	0.00764117
set	0.0135931	ieyasu	0.00542672	selección	0.00742191
plata	0.0113827	armadura	0.00535993	temporada	0.00655635
deporte	0.0106947	tropa	0.00494752	no	0.00620183

tercero	0.00885215	año	0.00474775	jugador	0.0055934
segundo	0.00776569	ira	0.00469496	liga	0.00550053
punto	0.00766544	poder	0.00425318	nacho	0.00540602
mundial	0.0075415	fuerza	0.00416388	anotar	0.00538315
primero	0.00751234	soldado	0.00409081	manitoba	0.00507496
roedor	0.00743972	maestre	0.00385474	ametralladora	0.00471993
campeón	0.00685581	también	0.00378127	nacional	0.00437986
cuarto	0.00603892	odontoglossum	0.0037286	dos	0.00428268

Tópico 148		Tópico 153		Tópico 170	
romano	0.0186566	municipio	0.0211601	colombiano	0.0247742
filipo	0.0160991	localidad	0.00926092	colombia	0.0148142
macedonia	0.015153	situar	0.00842069	farc	0.0137519
ejército	0.0135362	parroquia	0.00827022	gobierno	0.00830832
a.	0.0114892	año	0.00807779	secuestrar	0.00807173
c.	0.0100232	provincia	0.00793562	presidente	0.00742514
ciudad	0.00997288	jurisdicción	0.00695374	ciudad	0.00668873
rey	0.00901031	habitante	0.00681432	algeciras	0.00573182
guerra	0.00756475	comarca	0.00639666	santa_fe	0.00524154
griego	0.00741049	ayuntamiento	0.00603696	año	0.00521306
tratado	0.00641418	contar	0.00596907	luke	0.00516378
macedonio	0.00618773	formar	0.005812	chávez	0.00497069
infantería	0.00553675	antártico	0.00567056	no	0.00482203
roma	0.00546188	castilla	0.00554039	militar	0.00448771
territorio	0.00543799	parte	0.00535504	venezolano	0.00410687
no	0.00535381	alcalde	0.0050608	también	0.00402899
oviedo	0.0052445	capital	0.00495482	país	0.00387524
aliado	0.00485094	burgos	0.00484009	primero	0.00378961
tropa	0.00470575	isla	0.00463832	donde	0.00373829
grecia	0.00452498	españa	0.00442034	después	0.00366255

Tópico 209		Tópico 213		Tópico 264	
canción	0.0247988	especie	0.0568592	piano	0.0325702
álbum	0.021148	í	0.0471959	orquesta	0.0322793
dylan	0.0161113	flor	0.0406296	música	0.0306427
grabar	0.0108135	planta	0.0308207	sinfonía	0.0272495
bob_dylan	0.00950325	hoja	0.0271946	compositor	0.0246308
ecuatoriano	0.00925443	externo	0.026449	obra	0.0218228
soprano	0.00819425	o.	0.0243557	concierto	0.0197398
concierto	0.00813805	color	0.0226904	musical	0.0128931
primero	0.00803234	género	0.0191103	beethoven	0.012828
año	0.00797899	cm	0.0189815	sinfónico	0.0110908

grabación	0.00791289	ophrys	0.0183646	composición	0.0105541
gira	0.00774977	familia	0.0178358	ópera	0.0105261
emelec	0.00739459	perteneciente	0.0135923	movimiento	0.00833572
publicar	0.00697351	enlace	0.0108993	componer	0.00790022
tema	0.00680265	tallo	0.0104539	nú	0.00732599
banda	0.00568481	insecto	0.009297	coro	0.00724842
versión	0.00506371	botánico	0.00848312	min	0.00704828
jugar	0.00501927	ssp.	0.00788901	músico	0.00702686
humberto	0.00467762	pequeño	0.00695982	director	0.00674265
voz	0.00463092	seleccionar	0.0068616	cuerda	0.00662168

Tópico 405		Tópico 449		Tópico 662	
perú	0.058633	verbo	0.0288322	buque	0.0185779
colombia	0.0461755	oración	0.0126578	naval	0.0162757
ecuador	0.0389508	pronombre	0.00861237	submarino	0.0162642
masdevallia	0.0192476	gramática	0.00775082	navío	0.0159404
venezuela	0.0177176	no	0.00752139	corbeta	0.0104022
image	0.0171211	lengua	0.00629681	peruano	0.00766564
costa_rica	0.0164851	dos	0.00581088	grau	0.00714169
m.	0.0124754	fachada	0.0056368	marina	0.00713736
cívica	0.00959381	tiempo	0.00548891	puerto	0.0068996
panamá	0.00803523	poder	0.00533638	guerra	0.00678844
nacional	0.00739118	participio	0.00517033	nave	0.00651325
guayaquil	0.00654252	primero	0.00475749	capitán	0.00581695
estados_unidos	0.00644308	también	0.00471036	her	0.00577063
brasil	0.00555669	riley	0.00463012	alemán	0.0057655
américa	0.00503204	sujeto	0.00460528	crucero	0.00555494
peruano	0.00501101	presente	0.00458983	no	0.00501964
presidente	0.00466925	otro	0.00449879	marino	0.00467932
no	0.00422018	nominal	0.0042991	otro	0.00441112
también	0.00415946	forma	0.0038632	isla	0.00431113
organización	0.00415388	parte	0.00386002	mar	0.00423846

Se debe destacar que las palabras que mejor describen a un tópico, son aquellas con las más altas probabilidades, por lo cual están ordenadas en forma descendente. Además, solo se muestran las 20 primeras palabras.

Conclusión del sistema LDA

Como se puede observar en los tópicos, se han encontrado muy buenos resultados, claro está, que estos resultados no están completamente limpios, ya que hay palabras que desde nuestro punto de vista no pertenecen al tópico, a pesar de ello, varios tópicos son entendibles y se ligan fácilmente a temas específicos.

El hecho de conseguir estos tópicos es muy importante, esto es, si se desea conocer de qué habla cierto archivo, solo sería necesario ver los tópicos a los que está ligado, y de este modo podemos tener una idea del contenido del archivo sin necesidad de leerlo completamente.

También se debe hacer la mención de que no todos los 1300 tópicos representan a temas diferentes, es decir, es posible encontrar tópicos similares, así como, tópicos que contengan palabras de varios temas, lo cual dificulta asociar estos tópicos a algún tema específico.

LDA es uno de los algoritmos más usados en el campo de Topic Modeling que se basa como hemos descrito en este trabajo en el concepto de estructura latente de tópicos dentro de una gran colección de documentos. El método de muestreo de Gibbs es el algoritmo más usado dentro de la familia de los métodos de muestreo. Sin embargo, existen otro tipo de algoritmos que no aproximan la distribución a posteriori con muestras, sino que son de tipo determinista tales como Variational Bayesian Inference (VB), Expectation Propagation (EP), Variational EM entre otros, que podrían implementarse para comparar resultados.

Finalmente, para trabajo futuro, sería muy interesante aplicar este sistema a música e imágenes, sin embargo, se debe estudiar el procesamiento de transformar música e imágenes a texto.

Referencias

Abella, R., & Medina, J. E. (2014). Segmentación lineal de texto por tópicos. *Serie Gris, CENATAV*.

Barney, B. (2015). *OpenMP*. Obtenido de https://computing.llnl.gov/tutorials/openMP/

Barney, B. (2017). *Introduction to Parallel Computing*. Obtenido de https://computing.llnl.gov/tutorials/parallel_comp/

Barney, B. (2017). *Message Passing Interface (MPI).* Obtenido de https://computing.llnl.gov/tutorials/mpi/

Bezanson, J. (2017). *Julia Benchmarks*. Obtenido de https://julialang.org/benchmarks/

Blei, D., & Jordan, M. (2002). Modeling Annotated Data. Technical Report UCB//CSD-02-1202, U.C. Berkeley Computer Science Division.

Dickey, J., Jiang, J.-M., & Kadane, J. (1987). Bayesian Methods for Censored Categorical Data. *Journal of the American Statistical Association, 82*, 773-781.

Gebali, F. (2011). *Algorithms and parallel computing.* Hoboken, N.J.: Wiley.

Griffiths, T., & Steyvers, M. (2002). A probabilistic approach to semantic representation. *In Proceedings of the 24th Annual Conference of Cognitive Science Society*.

Griffiths, T., & Steyvers, M. (2003). Prediction and Semantic Association. 11-18.

Griffiths, T., & Steyvers, M. (s.f.). Finding scientific topics. *Proceedings of the National Academy of Science, 101,* 5228-5235.

Griffiths, T., Steyvers, M., Blei, D., & Tenenbaum, J. (2005). Integrating Topics and Syntax. *In Advances in Neural Information Processing Systems, 17*.

Heinrich, G. (2008). Parameter estimation for text analysis. *Technical Report Fraunhofer*.

Hofmann, T. (1999). Probabilistic Latent Semantic Analysis. In Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence.

Hofmann, T. (1999). Probabilistic Latent Semantic Indexing. Proceedings of the Twenty-Second Annual International SIGIR Conference, 1999. F. Jelinek. Statistical Methods for Speech Recognition. MIT Press, Cambridge, MA, 1997. .

Hofmann, T. (2001). Unsupervised Learning by Probabilistic Latent. *Machine Learning, 42*, 177-196.

Jordan, M. (1999). *Learning in Graphical Models.* Cambridge: MIT Press.

Jordan, M., Ghahramani, Z., Jaakkola, T., & Saul, L. (1999). *An Introduction to Variational Methods for Graphical Models* (Vol. 37). Machine Learning.

Landauer, T., & Dutnais, S. (1997). A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review, 104*, 2011-240.

Landauer, T., Foltz, P., & Laham, D. (1998). An Introduction to Latent Semantic Analysis. *Discourse Processes, 25*, 259-284.

LNS. (2017). *Laboratorio Nacional de Supercomputo*. Obtenido de http://www.lns.buap.mx

Minka, T. (2000). Estimating a Dirichlet distribution. *Technical report, M.I.T.*

Minka, T., & Lafferty, J. (2002). Expectation-Propagation for the Generative Aspect Model. *In Uncertainty in Artificial Intelligence (UAI).*

MPI Forum. (2017). *MPI Documents*. Obtenido de http://mpi-forum.org/docs/

Pacheco, P. S. (2011). *An introduction to parallel programming.* Amsterdam: Morgan Kaufmann.

Petersen, W., & Arbenz, P. (2004). *Introduction to parallel computing.* Oxford: Oxford University Press.

Real Academia Española. (2017). *Real Academia Española*. Obtenido de www.rae.es

Reese, S., Boleda, G., Cuadros, M., Padró, L., & Rigau, G. (2010). Wikicorpus: A Word-Sense Disambiguated Multilingual Wikipedia Corpus. La Valleta, Malta: In Proceedings of 7th Language Resources and Evaluation Conference (LREC'10).

Software Intel. (2017). *OpenMP* Pragmas and Clauses Summary*. Obtenido de https://software.intel.com/es-es/node/522685

Steyvers, M., & Griffiths, T. (s.f.). Probabilistic Topic Models . *In T. Landauer, D McNamara, S. Dennis, and W. Kintsch (eds).*

Top500.org. (2017). *TOP500 Supercomputer Sites*. Obtenido de https://www.top500.org/

Universitat Politècnica de Catalunya. Research Group on Natural Language Processing; Gemma Boleda; Universitat Pompeu Fabra. Institut Universitari de Lingüística Aplicada (IULA). (2012). GrAF version of Spanish portions of Wikipedia Corpus.

Wang, Y. (2008). Distributed Gibbs Sampling of Latent Topic Models: The Gritty Details.