Algoritmo de convolución mediante clases S4 en lenguaje R utilizando el método FFT

Cristian Aguirre Janampa
Jesús Cóndor Torres
Jean Alvitres Palomino

24 de Junio del 2018

Resumen

Los algoritmos de convolución son una herramienta muy útil en la parte de distribuciones de variables aleatorias, sean univariadas o multivariadas. Sin embargo el aprovechar estos algoritmos a su máxima utilidad depende mucho del lenguaje de programación que usamos como soporte; en esta oportunidad implementaremos nuestros algoritmos en el lenguaje R, y así estableceremos el ánalisis de cada algoritmo implementado a cada código, para luego ver el comportamiento de ciertas distribuciones, de como puede cambiar su comportamiento si vamos cambiando algunos parámetros de dichas dsitribuciones asi como la distancia de una variación entre otra variación de las distribuciones que se mencionarán. El uso de las clases S4 en el lenguaje R nos da una herramienta poderosa al momento de resolver problemas de convoluciones por medio de varios métodos, en este caso el método FFT (Fourier Fast Transform).

1. Introducción

En las últimas décadas la programación y las matemáticas han ido desarrollando una interacción muy fuerte de modo que hoy en día se puede decir que la matemática necesita a la programación y la programación necesita de las matemáticas ya sea para cálculos, simulaciones, gráficos, etc. Dentro de las matemáticas tenemos lo que es la probabilidad y la estadística que concierne básicamente en la incertidumbre de cierto evento o suceso que pueda ocurrir en base a ciertos datos recopilados o sino definir una variable aleatoria y trabajar con ella bajo cierto espacio muestral. En esta oportunidad utilizaremos las propiedades de las variables aleatorias y nos enfocaremos en las transformaciones de esta y como esto puede ser llevado a la programación.

La "Programación orientado a objetos"(OOP) nos ayuda grandemente en la implementación de convolución de distribuciones de variables aleatorias de valor real (discretas o continuas). Por tanto, es bueno observar las características de la OOP y darle un soporte en base a los algoritmos de convolución que nos ayuda mucho en el análisis probabilístico, como por ejemplo en las transformaciones de funciones, en este caso las transformaciones de variables aleatorias y esto llevarlo a la programación en clase S4.

Las distribuciones más conocidas, tanto de variable aleatoria discreta o continua se encuentran en los paquetes de distintos lenguajes de programación, en esta oportunidad el lenguaje de programación que utilizaremos será el lenguaje R, que nos ayudará en la parte de los objetos y también de cálculos. Además, trataremos los siguientes puntos:

* Programación orientada a objetos para distribuciones de probabilidad.
* La clase S4.
* Implementación de las clases de distribuciones dentro del concepto S4.
* El algoritmo de convolución y FFT.
* Algoritmos basados en DFT (Transformación Discreta de Fourier).

2. Estado del Arte

La convolución de las distribuciones de probabilidad surge en la teoría de la probabilidad y la estadística como la operación en términos de distribuciones de probabilidad que corresponde a la adición de variables aleatorias independientes y por extensión, a la formación de combinaciones lineales de variables aleatorias.

Para su implementación, el uso de la Transformada Rápida de Fourier(FFT) ha sido muy común desde la aparición del algoritmo de Cooley y Tukey, el cual es un algoritmo eficiente que ayuda a calcular la Transformada de Fourier Discreta(DFT) y su inversa.

Las apariciones de arquitecturas informáticas paralelas dan más importancia al uso de la FFT, en contraste con la convolución mediante cálculos directos y si sumamos a esto un enfoque de Programación Orientada a Objetos (POO) hace que esta técnica sea aún más atractiva ya que podemos usarla como un algoritmo predeterminado en situaciones donde no existe una alternativa mejor, mientras que en casos especiales como por ejemplo , las variables aleatorias normales o de Poisson, donde la convolución se reduce a la transformación de los parámetros correspondientes.

En este contexto, la convolución es un arma potente para establecer una aritmética completa de operaciones matemáticas que actúan sobre objetos de distribución, que comprende, entre otros, operadores como la se suma, resta, multiplicación, división y exponenciales. En esta aritmética, identificamos distribuciones con variables aleatorias independientes: si X1 y X2 son variables de aleatorias independiente, X1 + X2 producirá la distribución de la suma de variables aleatorias (independientes) respectivas, es decir, su convolución. Técnicamente, hablando en términos de programación, hemos sobrecargado el operador + para distribuciones univariables. La propia convolución se calcula de acuerdo con las clases reales de los operandos, con métodos particulares (exactos) para, por ejemplo, distribuciones normales o de Poisson.

3. Diseño del experimento

Dado que el presente proyecto se basa en la implementacion de clases S4 en R, necesitamos tener los conceptos claros para poder desarrollar nuestro método FFT. Entonces tenemos que tener en cuenta los siguientes puntos:

3.1. OOP para distribuciones de probabilidad

Como nuestro objetivo es ver los resultados de ciertos algoritmos de convolución que nos permitirá observar el desarrollo y análisis de los datos que se podrán obtener al ejecutar el código, necesitaremos un lenguaje de programación, en esta oportunidad utilizaremos el lenguaje R, debido a que se esta haciendo en este lenguaje el código abierto necesitamos ciertos paquetes que se encuentra distribuido en este lenguaje para poder usar funciones y operaciones que necesitamos en nuestros algoritmos.

A pesar de tener paquetes disponibles en nuestro lenguaje para realizar nuestros algoritmos, pueda que tengamos algunas limitaciones, ya sea por la memoria de la maquina o la estructura de los paquetes, que tienen un rol importante para nuestras distribuciones. Para este proyecto utilizaremos los paquetes "distr" y "distrEx", además de ciertas funciones que están incluido en esos paquetes. Ahora como ya tenemos formulados nuestros algoritmos, entonces necesitamos tipos de datos particulares para las distribuciones además de la ejecución del código, es posible que se quiera formular declaraciones para la varianza de funciones de variables aleatorias, la cual no importa si la suma es finita o una suma de infinitos sumandos (integral), donde el resultado puede resultar en una nueva distribución. Si bien es cierto que el lenguaje que utilizamos es R, hay que mencionar que en el leguaje java existe un paquete de simulación la cual se llama MCMC HYDRA que dirige un enfoque muy similar a OOP en R.

3.2. El concepto de clase S4

Antes de implementar nuestros algoritmos, veamos algunos conceptos de la clase S4, primeramente para hablar de la clase S4, debemos hablar de la clase S3 que tiene contraste con COOP (Programación orientada a objetos de clase), estos son métodos que nos ayudan a manejar o tener acceso de los objetos que se utilizan al momento de programar; sin embargo estas no eran factibles al momento de utilizar funciones, debido a ese problema es que se creo la clase S4 que tiene el estilo FOOP (Programación orientada a objetos de funciones).

Mientras que el estilo COOP proporciona acceso o manipulación de un objeto, el FOOP hace llamadas a las funciones genéricas abstractas que permiten argumentos de tipo – clase de variable. También tiene algunas ventajas para funciones, como por ejemplo el símbolo "+" que tiene un significado para muchos tipos/ clases de operando como en nuestro caso de convolución. Entonces vemos que la clase S4 tiene una relación con las funciones que se pueden utilizar en nuestro algoritmo de convolución, sin embargo, ha sido muy difícil encontrar algoritmos de convolución con clase S4, por lo que hemos visto a analizar los códigos que se encuentran en el link que dejo del proyecto.

Cabe mencionar que el estilo COOP se ha permitido en los conceptos de la clase S4, pero hay que tener en cuenta que esas funciones en S4 no es estándar, luego para nuestras clases de distribuciones podemos usar los miembros del tipo funcional, aunque de forma muy limitada.

3.3. Descripción de los objetos

En esta parte se describirán los objetos y funciones que se usaron en los códigos fuente, tambien hay paquetes del lenguaje R que se tuvo primero que descargar y luego llamar a través de la función "library".

•Librería "distr": Ayuda en el tratamiento de las distribuciones en clase S4.
•Librería "distrEX": Paquete de R que proporciona algunas extensiones de distr.
•Librería "plot": Paquete de R que proporciona los gráficos en el lenguaje.
•Método q(X): Calculo de la función "quantile" de la distribución X.
•Método p(X): Método del paquete distr. Halla el CDF de una distribución X.
•Método r(X): Método del paquete distr. Halla valores aleatorios de la distribuciónX.
•Método d(X): Método del paquete distr. Halla el PDF de una distribución X.
•Método size: Da como entrada el tamaño de la variable o vector.
•Método prob: Da como entrada un número que esta entre 0 y 1.
•Método N: Da como entrada el número de sumas.
•Método Lambda: Da como entrada el parámetro de la distribución de Poisson.
•Método cex.inner: Da como entrada un número, cantidad de texto de trazado.
•Método inner: Da como entrada el nombre de los gráficos.
•Método to.draw.arg: Da como entrada el número de gráficos.

3.4. Descripción de las funciones y técnicas a utilizar

Las funciones que utilizaremos son las siguientes:
•Función Norm: Usa la distribución normal estándar.
•Función Pois: Utiliza el parámetro "Lambda"de la distribución de Poisson.
•Función Bin: Utiliza la distribución binomial con parámetros size y prob.
•Función convpow: Determina la distribución de la suma de la N univariantes.
•Función as: Convierte un objeto en una clase determinada.
•Función TotalVarDist: Calcula la distancia de variación total entre 2 distribuciones.
•Función KolmogorovDist: Calcula la distancia de Kolgomorov de 2 distribuciones.

4. Experimentos y resultados

Los resultados obtenidos en código1.R son los siguientes:
Cambios en el parámetro de Poisson

Parámetro de Poisson	Función quantile
1	-0.3471003
2	-1.011822
3	-1.61587
4	-2.259338
5	-2.914877

Figura 1

Ahora veamos los resultados para la función de distribución p(x) con respecto a 3 puntos en la función densidad con el cambio de parámetro lambda desde 1 hasta 10. La siguiente tabla nos muestra los resultados obtenidos.

p(x)1	p(x)2	p(x)3
0.8545304	0.9409595	0.9729868
0.7482524	0.8588449	0.9173257
0.6777285	0.7860775	0.8564942
0.6320522	0.7292077	0.8013658
0.6022044	0.6866537	0.7552880
0.5821775	0.6551044	0.7181417
0.5682618	0.6315319	0.6885803
0.5582238	0.6136309	0.6650536
0.5507182	0.5997634	0.6461911
0.5449224	0.5887960	0.6308986

Figura 2

Ahora veamos los resultados que arrojó r(x) respecto de sus 3 últimos puntos y el cambio de parámetro Lambda con valores de 1 hasta 10:

r(x)3	r(x)4	r(x)5
0.000000	0.000000	2.431134
4.81732061	0.000000	1.88896789
-3.520798	1.869977	2.312758
12.8349148	2.2968920	-0.5722682
15.3149014	4.8949530	-0.2971279

Figura 3

Del segundo algoritmo implementado en codigo1.R, usando el paquete distrEX para poder hallar la distancia de variacion total y distancia de kolmogorov, los resultados obtenidos son respecto de la variación del parámetro de truncamiento "TruncQuantile":

TruncQuantile	Distancia de variación total	Distancia Kolmogorov
1e-11	2.326616 x 1e-12	1.346317 x 1e-12
1e-12	1.326478 x 1e-13	7.726632 x 1e-14
1e-13	1.686842 x 1e-14	1.242756 x 1e-14
1e-14	6.561477 x 1e-15	3.823331 x 1e-15
1e-15	5.238968 x 1e-15	2.747802 x 1e-15

Figura 4

5. Discusión

A partir de los resultados obtenidos, podemos discernir en varios puntos :

- ✓ El algoritmo mostrado en el presente paper nos muestra una tecnica útil para hallar las convoluciones por medio de FFT; sin embargo para ciertas ocasiones, hay varios metodos de obtener resultados. Por ejemplo, en el algoritmo 3 visto en la seccion Experimentos y resultados, hay una forma de obtener el mejor metodo FFT. En dicho codigo el que se aproxima más al resultado "verdadero.es FFT3.

- ✓ La distancia de Kolmogorov que nos permite hallar la precisión de nuestro algoritmo en términos de la varianza total es una prueba confiable, mas no más eficaz que otra pruebas o métodos, como por ejemplo la prueba de Lilliefors puede mejorar la precisión en algunos casos.

- ✓ El algoritmo planteado y los resultados obtenidos nos enseñan a interpretar resultados, esto es, de encontrar varios métodos para hallar un resultado final, cada dato(distribuciones) se puede aproximar más al dato real(otras distribuciones); sin embargo este procedimiento nos enseña también como encontrar otros métodos o caminos.

6. Conclusión y trabajos futuros

- ✓ Pudimos hallar buenas aproximaciones por el algoritmo de convolución por FFT, ajustando el valor del parámetro de truncamiento para tener una mejor aproximación.

- ✓ El aprovechamiento de las librerias de R, especificamente "distrz "distrEx"son de primera necesidad al momento de desarrollar nuestro programa. Se espera que por medio de estas librerias podamos desarrollar nuevas aplicaciones mas avanzadas en clases S4.

- ✓ Como lo mencionamos anteriormente, la distancia de Kolmogorov no es la única manera de hallar la eficacia de nuestro algoritmo. Como un trabajo futuro se puede optimizar dicho método por medio de otras pruebas, como por ejemplo: Prueba de Liliefors, Test de Shapiro-Wilk, etc.

- ✓ La implementacion de los códigos en Jupyter notebook tuvo ciertos problemas en el kernel de Jupyter, por ejemplo habia un error en el kernel con el comando quit(); cuando se escribia otro comando, se ejecutaba quit() y se detenia el codigo y no obteniamos el resultado esperado. Se logró superar el obstáculo modificando los comandos o agrupandolos.

- ✓ Las implementaciones en .Rmd en el repositorio del proyecto se determinó colocarlos dado que es más accesible modificarla conjuntamente que en Jupyter Notebook. Se espera futuramente que proyectos como éstos sirva para poder afianzarse en proyectos más extensos que éstos.

7. Bibliografía y referencias

- ✓ Peter Ruckdeschel y Matthias Kohl, General Purpose Convolution Algorithm in S4-Classes by means of FFT:

  https://arxiv.org/pdf/1006.0764.pdf

- ✓ Prototipo de algoritmo de clases S4 :

  https://gist.github.com/cbare/3670578

- ✓ Matthias Kohl y Peter Ruckdeschel, R package distrMod: S4 Classes and Methods for Probability Models:

  http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.303.9036&rep=rep1&type=pdf