Estimando o valor de uma integral a partir de um gerador MCMC

Laboratório de Computação e Simulação

(MAP2212)

Alunos:	**Lucka de Godoy Gianvechio e Milton Leal Neto**
Curso:	Bacharelado de Matemática Aplicada Computacional
NUSP:	11352442 e 8973974
Professor:	Julio Stern

São Paulo, 30 de junho de 2021

Conteúdo

1 Apresentação

Este relatório apresenta uma solução para o quinto Exercício Programa (EP) proposto pelo professor Julio Stern no âmbito da disciplina de Laboratório de Computação e Simulação (MAP 2212) do Bacharelado de Matemática Aplicada Computacional (BMAC) do Instituto de Matemática e Estatística (IME) da Universidade de São Paulo (USP).

A título de registro, nos foi solicitado que obtivéssmos uma função $U(v)$ que pudesse estimar, com erro $< 0.05\%$, a função verdade

$$W(v) = \int_{T(v)} f(\theta|x,y)d\theta$$

que representa a massa de probabilidade a posteriori no domínio $T(v)$, ou seja, a massa de probabilidade correspondente da função $f(\theta|x,y)$ que não ultrapassa um determinado nível v.

A função f, que tem distribuição de probabilidades Dirichlet, dada por

$$f(\theta|x,y) = \frac{1}{B(x+y)} \prod_{i=1}^{m} \theta_i^{x_i+y_i-1}$$

representa o modelo estatístico m-dimensional Multinomial-Dirichlet e recebe como parâmetros um vetor de observações x, um vetor de informações a priori y e um vetor de probabilidades θ, sendo que $x, y \in N^m, \theta \in \Theta = S_m = \{\theta \in R_m^+ | \theta'1 = 1\}$ e B representa a distribuição *Beta*. Aqui trabalhamos com $m = 3$.

Além disso, vale ressaltar que a região $T(v)$ foi definida como

$$T(v) = \{\theta \in \Theta | f(\theta|x,y) \leq v\}.$$

Especificamente no EP5, nos foi pedido para construir um gerador de números aleatórios com distribuição Dirichlet utilizando o método conhecido como Monte Carlo Markov Chain com a utilização do algoritmo de aceitação Metropolis. De posse desse gerador, calculamos a função $U(v)$.

2 Estratégia de resolução

Nos foi solicitado que utilizássemos a distribuição Normal Multivariada com vetor de médias igual a zero e matriz de covariância Σ como a distribuição proposta do Monte Carlo Markov Chain. Por isso, o primeiro passo foi construir tal matriz para podermos gerar pontos dessa distribuição e começarmos a construir a cadeia.

Como conhecemos as fórmulas analíticas da variância e covariância da Dirichlet, montamos a matriz simétrica Σ com a diagonal composta pelas variâncias e as demais entradas com a respectiva covariância, sempre levando em conta os vetores de parâmetros x e y.

De acordo com Gilks et. al (1997), é possível mostrar que a matriz de covariância ótima é aquela construída conforme descrito acima e multiplicada pela constante $2.38^2 \cdot d^{-1}$, onde d é a dimensão da distribuição em questão. No nosso caso, temos $d = 2$, pois geramos θ_3 a partir de $1 - (\theta_1 + \theta_2)$. Dessa forma, não precisamos lançar mão do método adaptativo do MCMC, que utiliza os dados gerados para ir atualizando a matriz de covariância até que se obtenha uma matriz adequada para rodar o MCMC.

De posse da matriz Σ, construímos o programa levando em conta o algoritmo de Metropolis, que é uma espécie de algoritmo de aceitação e rejeição.

Com o objetivo de esquentar a cadeia de Markov, decidimos estipular que os 1000 primeiros pontos gerados seriam descartados.

3 Definindo n

Assim como no EP4, também utilizamos o Teorema Central do Limite para estimar o valor ótimo de n que nos levasse à acurácia desejada, considerando o erro absoluto.

Para tanto, utilizamos a fórmula

$$n_{final} = \frac{\Phi^{-1}(1 - \frac{\delta}{2})^2 \cdot \hat{\sigma}^2}{\epsilon^2},$$

na qual $\Phi^{-1}(1-\frac{\delta}{2})$ corresponde ao percentil de uma distribuição $Normal(0,1)$, $\hat{\sigma}^2$ corresponde à variância da amostra piloto e ϵ corresponde ao erro máximo suportado, no caso 0.0005. Neste trabalho, decidimos considerar $\delta = 95\%$.

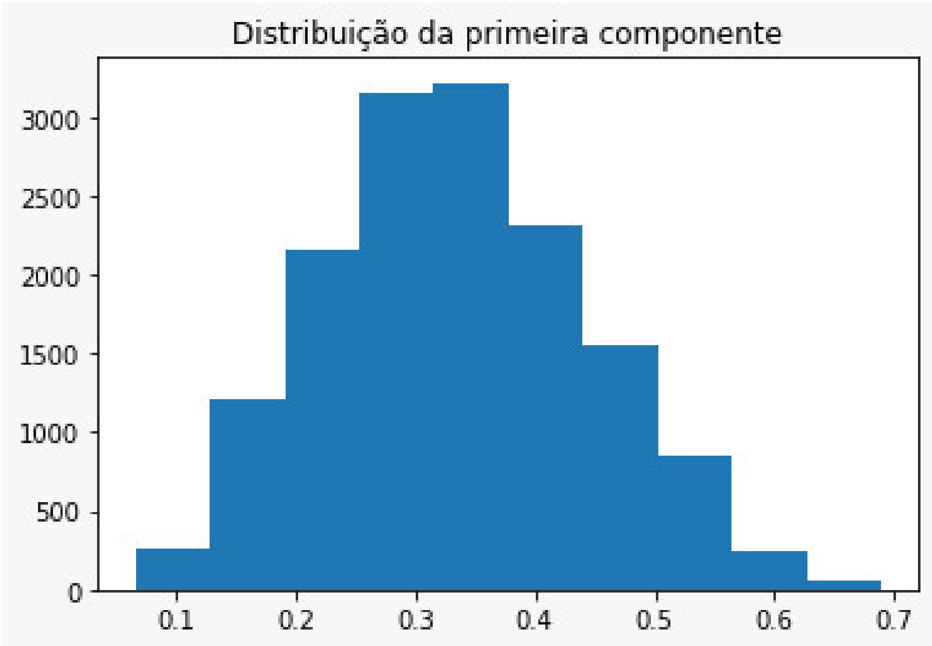
Para estimarmos a variância, criamos 100 experimentos aleatórios nos quais geramos 1000 pontos aleatórios de uma Dirichlet, avaliamos a f nestes pontos, ordenamos os resultados e consideramos a média entre os dois menores valores obtidos em cada simulação como uma estimativa para a posição daquele que poderia ser considerado o v_1, ou seja, o primeiro nível de corte a ser feito na função.

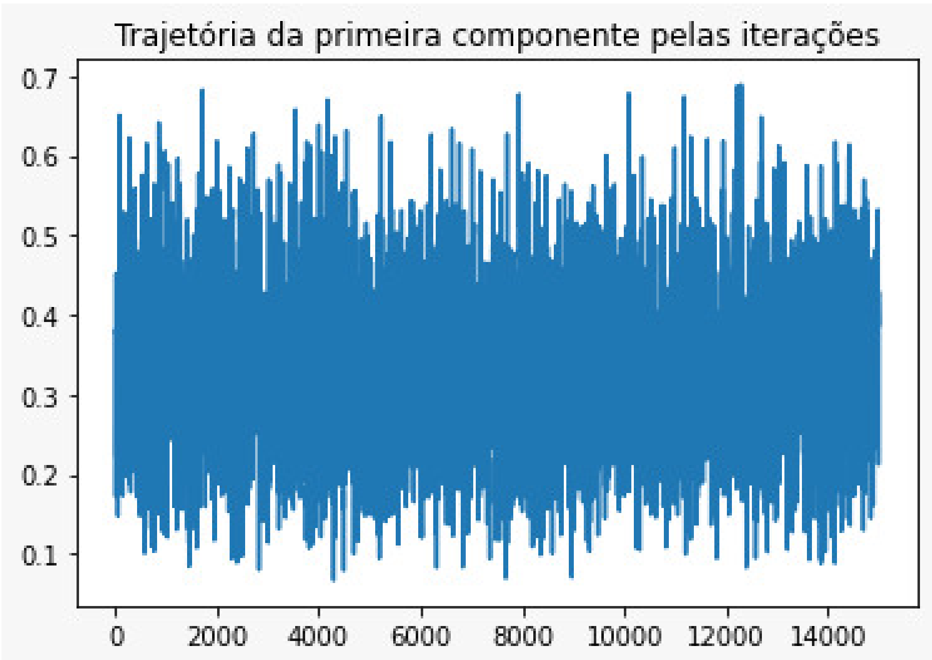
Dessa forma, obtivemos 100 valores de v_1 e calculamos a variância destes valores e a utilizamos como variânia amostral do nosso experimento.

Diferentemente do EP4, que obteve valor final de n na casa de 20 mil, no EP5, devido ao aumento da variância dos pontos, o n final estimado, a depender dos vetores de entrada, ficou entre 100 e 300 mil pontos.

4 Resultados do Gerador

Abaixo vemos gráficos que mostram que o gerador de pontos da Dirichlet construído via MCMC ficou dentro do que seria esperado para os pontos, dado o vetor de parâmetros $x + y = [6, 10, 2]$.





5 Estrutura do programa

O programa escrito em *Python* está estruturado em sete funções:

1) matriz_de_covariacia():

Calcula as variâncias e covariâncias e cria a matriz Σ.

2) calcula_f_indicadora():

Calcula a função f usada no algoritmo de aceitação se nenhum dos pontos tiver alguma coordenada negativa. Essa função atua basicamente como uma função indicadora.

3) gera_dir():

Gera os pontos com distribuição Dirichlet. Contém o MCMC.

4) calcula_n_final():

Realiza o experimento para estimar a variância amostral e calcula o n_{final} a ser utilizado no programa.

5) calcula_f():

Computa a constante de normalização, a função f e ordena os resultados obtidos.

6) estima_W():

Verifica quantos pontos existem abaixo do nível de corte desejado e retorna a proporção de pontos em relação ao total de pontos gerados como resultado da função $U(v)$, que estima a $W(v)$.

7) main():

Chamada principal do programa. Inclui as linhas de código que interagem com o usuário e que imprimem os resultados na tela.

6 Conclusão

Podemos concluir que o método de MCMC proposto neste trabalho funciona para gerar pontos de uma distribuição, porém ele não é tão eficiente quanto outras alternativas que existem como o algoritmo Hamiltoniano.

Obtivemos uma taxa de aceitação de pontos na faixa de 34% que, segundo Gilks, está dentro da faixa considerada ótima para o algoritmo de Metropolis.

No que tange aos resultados obtidos na estimativa da função $U(v)$, concluímos que obtivemos resultados muito próximos em relação ao EP4, com diferença na terceira ou quarta casas decimais.

7 Referências

[1] Andrew Gelman, Walter R Gilks, and Gareth O Roberts. Weak convergence and optimal scaling of random walk metropolis algorithms. The annals of applied probability, 7(1):110–120, 1997.

[1] Morris H. DeGroot and Mark J. Schervish. Probability and Statistics. Pear- son, 4th edition, 2012.