MAP2212

Laboratório de Computação e Simulação

Enzo Valentim Cappelozza
Lucas Panfilo Donaire

Professor:
Julio Michael Stern

IME-USP

29 de maio de 2022

1 Introdução

O presente estudo tem por finalidade estudar a *Função Verdade* $W(v)$, definida por

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

onde $T(v) = \{\theta \in \Theta \ : \ f(\theta \mid x, y) < v\}$ e $\theta = (\theta_1, \theta_2, \theta_3)$, e $f(\theta \mid x, y) = \theta_1^{x_1+y_1-1}\theta_2^{x_2+y_2-1}\theta_3^{x_3+y_3-1}$. Isto é, temos por finalidade responder a seguinte pergunta:

Dada a restrição $S_3 = \{\theta \in \mathbb{R}^3 \ : \theta_1+\theta_2+\theta_3 = 1\}$, qual é o valor da integral de f no espaço paramétrico $T(v)$, dado v fixado ?

Claramente, não poderemos resolver a integral acima, dado que não há computador com processamento infinito a fim de tratar um espaço paramétrico contínuo. Devemos, portanto, discretizá-lo e estudar a integral com um erro arbitrariamente pequeno, o qual fixaremos em 0.05%.

2 Desenvolvimento do algoritmo

Ao desenvolver o algoritmo, não conseguimos obter uma visualização geométrica para o problema, dado que a variação dos vetores $x = (x_1, x_2, x_3)$ e $y = (y_1, y_2, y_3)$ dificulta a intuição. Também reconhecemos que não temos o devido ferramental matemático para tentar resolver o problema "no papel" majorando ou minorando as integrais. De tal maneira, e sabendo que a função é contínua, pois se trata de um produto de polinômios (que são contínuos), fizemos as seguintes suposições:

- 1 - Devemos um cuidado especial, sobretudo, em áreas de maior variação da função. Isto é, quando maior a variação do valor da função (derivada) numa dada vizinhança, mais pontos devemos inserir nas partições do domínio ao redor de tal vizinhança.

- 2 - Pelo método anterior, ao colocarmos mais partições em locais cuja sensibilidade à alterações na imagem é maior, iremos naturalmente inserir mais partições perto do ponto de máximo da função.

- 3 - Ao inserir mais partições nas vizinhanças do máximo da função, teremos uma boa aproximação para tal valor e, assim, poderemos prosseguir com o resto do processo.

O modo de funcionamento do algoritmo é o seguinte:

- 1 - Inicia-se o Estimador com um número n de pontos pre-setados pelo usuário.
- 2 - São criados n pontos θ da forma $(\theta_1, \theta_2, \theta_3)$ com distribuição Dirichlet a serem distribuídos nos bins.
- 3 - É calculado o valor da função nos pontos que foram criados, os resultados são listados de maneira crescente e é selecionado o ponto cuja imagem por f é o máximo.
- 4 - Criamos mais bins em locais estratégicos cuja sensibilidade da função é maior.

O método funciona da seguinte maneira: Calculamos a quantidade de vetores θ tal que $f(\theta) < v$, para todos os bins e, guardamos esses valores em um vetor, (no código, FV) após isso, fizemos um vetor de diferença (dif) da função no bin i para o bin $i+1$, isto é: $dif[i] = FV[i+1] - FV[i]$. Assim, identificamos as áreas entre os bins que a função $U(v)$ é mais "sensível", isto é: uma perturbação em v resulta numa grande perturbação de $U(v)$. Algo como uma analogia da derivada com valor grande no ponto. Assim, a cada loop, colocamos mais bins (no total) da nossa lista, estes uniformemente espaçados, em cada intervalo de bins da iteração anterior, numa quantidade proporcional à diferença $FV[i+1] - FV[i]$. Ou seja, onde a função é mais sensível à mudança, colocamos mais pontos de corte (bins) para entender melhor a especificidade do comportamento dela. Após todos os intervalos cumprirem o critério de parada de ter no máximo 1 ponto entre os bins, temos a disposição final dos bins.

3 Resultados obtidos

Vetores	x:(1,2,3) e y:(2,4,6)
U(1)	0.039266666666666665
U(2)	0.08616666666666667
U(3)	0.13636666666666666
U(4)	0.18876666666666667
U(5)	0.243
U(6)	0.29796666666666666
U(7)	0.35446666666666665
U(8)	0.4125333333333333
U(9)	0.4682
U(10)	0.5281666666666667
U(11)	0.5902
U(12)	0.6500333333333334
U(13)	0.7116666666666667
U(14)	0.7748666666666667
U(15)	0.8364333333333334
U(16)	0.8994
U(17)	0.9630333333333333
U(18)	1

Vetores	x:(1,2,3) e y:(1,2,1)
U(1)	0.05793333333333334
U(2)	0.14546666666666666
U(3)	0.2445
U(4)	0.3513
U(5)	0.47026666666666667
U(6)	0.5965666666666667
U(7)	0.7265666666666667
U(8)	0.8668666666666667
U(9)	1

Vetores	x:(5,6,2) e y:(3,7,2)
U(1)	0.030933333333333334
U(2)	0.06653333333333333
U(3)	0.10223333333333333
U(4)	0.13743333333333332
U(5)	0.17523333333333332
U(6)	0.213
U(7)	0.25266666666666665
U(8)	0.2901
U(9)	0.32913333333333333
U(10)	0.36866666666666664
U(11)	0.40813333333333335
U(12)	0.44863333333333333
U(13)	0.4888
U(14)	0.5337333333333333
U(15)	0.5733
U(16)	0.6158333333333333
U(17)	0.6580333333333334
U(18)	0.7013666666666667
U(19)	0.7433333333333333
U(20)	0.7854
U(21)	0.8294
U(22)	0.8752
U(23)	0.9170666666666667
U(24)	0.9615666666666667
U(25)	1

4 Conclusão do estudo feito

De fato, não podermos calcular analiticamente o máximo da função é um grande empecilho. Isto é, para uma análise básica do que ocorre com a função, é necessário apelar para métodos computacionais não óbvios. Dito isso, tentamos fazer a implementação da biblioteca *Scipy* com o uso do pacote *Optimize* aliado ao uso de restrições lineares. No entanto, não conseguimos qualquer tipo de regularidade quanto ao ponto de máximo da função que nos estava sendo retornado. Claramente haviam pontos cuja imagem pela aplicação de f

era superior ao que estávamos recebendo como vetor resposta. Passamos alguns dias otimizando uma rotina errada por causa disso. Não esperávamos que um pacote feito justamente para isso nos retornaria resultados tão incoerentes. Disso, tiramos que, por vezes, o método mais eficiente é por "estimação cega", confiando na teoria e no algoritmo de amostragem. Isto é, quando não temos qualquer tipo de informação sobre a função, talvez seja melhor usar algum método *à la* Monte-Carlo ao invés de usar uma rotina pronta para uma função sobre a qual sabemos pouca (ou quase nenhuma) informação.

Referências