Exercício Programa 4

Ana Paula Lopes Cavalcante
Gabriel Tetsuo Haga

Junho 2021

1 Introdução

O objetivo deste relatório é apresentar o programa feito para solucionar o seguinte problema: Considere um modelo estatístico m-dimensional multinomial, com observações x, e informações *a priori* y e parâmetro θ, tal que $x, y \in \mathcal{N}^m$ e $\theta \in \Theta = S_m = \{\theta \in R_m^+ | \theta' 1 = 1\}$, $m = 3$. E esse modelo segue a função densidade de probabilidade *a posteriori* de Dirichlet dada por:

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

sendo $B(x+y)$ a função Beta multivariável. Considere agora duas funções uma $T(v)$ e $W(v)$, definidas como:

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

Assim o problema proposto consiste em criar uma função $U(v)$ que estime o valor de $W(v)$. Para tal deve-se definir k pontos de cortes: $0 = v_0 < v_1 < v_2 < ... < v_k = \sup f(\theta)$, e também deve-se criar n pontos θ_i, sendo $i \in \{1, 2, 3, ..., n\}$ que siga a distribuição de Dirichlet. Após isso estima-se que o $W(v_j) - W(v_{j-1})$ será, aproximadamente, a fração $p = \frac{l}{n}$, tal que $l = \sum_{i=1}^{n} R(\theta_i, v_{k-1}, v_k)$, sendo a função $R(\theta, v_{j-1}, v_j)$ definida como:

$$R(\theta, v_{k-1}, v_k) = \begin{cases} 1, \text{ se } v_{j-1} \leq f(\theta) \leq v_j \\ 0, \text{ se } f(\theta) < v_{j-1} \text{ ou } f(\theta) > v_j \end{cases}$$

A função deverá também ajustar, dinamicamente, os cortes v_j, para tal que $W(v_j) - W(v_{j-1}) \approx \frac{1}{k}$. Assim a função $U(v)$ irá utilizar esses cortes para estimar o valor de $W(v)$.

2 O código e a escolha de n

O código escrito em Python tem como objetivo estimar a função $W(v)$ com um erro menor que 0.0005. Desse modo escrevemos algumas funções, que juntas auxiliam a função principal a devolver $U(v)$, o valor da integral calculado até o ponto de corte v.

As funções `x()` e `y()` servem, intuitivamente, para o usuário escolher os vetores x e y e a partir deles a função `theta(n, x, y)` calcula n triplas ordenadas, com distribuição Dirichlet, tal que a soma dos valores dentro de cada tripla é igual a 1. A função `f(theta, x, y)`, usando os vetores x, y e θ, calcula a função de densidade de probabilidade posterior assim como esperado, gerando n valores.

A função `hitu(v1, v2, n, theta, x, y)` usa `f(theta, x, y)` dentro do comando `np.argwhere` para que uma lista seja criada com a posição dos valores que estavam entre v_i e v_{i+1}, posteriormente usando `len` para obter o número total de valores que caíram dentro desse intervalo e dividindo pelo número total de valores n.

A estimativa de $W(v)$ é calculada em si pela função `u()`, que usa todas as funções citadas anteriormente. Após ter chamado `x()` e `y()`, pedimos que o usuário insira k (número de cortes) como parâmetro e o valor v, tal que a integral da função será calculada até ele.

Tendo as entradas e o valor de n definido calculamos o vetor θ e aplicamos na função $f(\theta|x, y)$. A função `quantiles` da biblioteca `statistics` gera uma lista com $k - 1$ pontos de corte, pois não inclue os extremos, então adicionamos zero ao começo da lista e o *sup* de f no final, resultando numa lista de comprimento $k + 1$, tal que o número de valores que estão entre v_i e v_{i+1} sobre o número total de pontos é aproximadamente $\frac{1}{k}$. A função `quantiles` divide os valores de $f(\theta|x, y)$ em quantis de igual probabilidade (no nosso caso $\frac{1}{k}$), junto com os extremos, ou seja, a soma de todos os *bins*, deve ser um.

Após termos os cortes definidos, descartamos aqueles que são maiores que v, resultando numa lista de valores de comprimento menor ou igual que a anterior. Como sabemos que o valor da área em cada *bin* é $\frac{1}{k}$, assumimos que a área é $\frac{1}{k}$ vezes o comprimento da nova lista menos 1, ou seja, a soma de todas as parcelas, já que cada parcela tem área igual a $\frac{1}{k}$ e o número de parcelas é de fato o comprimento da nova lista menos 1. Depois checamos se o valor v estava na lista, se não e se o comprimento da lista está menor calculamos a área entre o último valor da lista e v, somando o resultado ao valor da área previamente calculado e assim obtemos a estimativa da integral.

Porém, precisamos de n para realizar esses cálculos. Afim de definir n usamos a função:

$$\epsilon = \frac{z\sigma}{\sqrt{n}}$$

Como podemos ver, n não é calculado diretamente. Na função começamos com $n = 100000$ e a partir disso calculamos cinco estimativas da integral e o desvio padrão entre elas. Tomando $z = 2.575$, para um confiança de 99%, usando o n escolhido e o desvio padrão obtido calculamos o erro. Se o erro for menor que 0.0005 prosseguimos, caso contrário dobramos o valor de n e realizamos o cálculo novamente.

Assim que obtivermos um n que gera um erro menor que 0.0005 calculamos a área até o corte v e printamos o resultado.

3 Resultados e Conclusão

Nessa seção discute-se os resultados obtidos pelo código. Primeiramente, será falado do tempo que a função demorou para rodar, após alguns testes com variando os parâmetros como x, y e k, notou-se um tempo de simulação por volta de 2 minutos, contudo esse tempo varia com os parâmetros e também com o valor de v escolhido. Esse tempo médio foi considerado, pelos integrantes do grupo, um valor aceitável para o tempo. Outro ponto avaliado no código foi o valor do erro da estimativa para a $W(v)$, os valores do erro absoluto, em geral, ficaram próximos de 0.0004, que é abaixo do valor máximo estabelecido de 0.0005, ou seja, atende esse critério. Para verificar a coerência do código foi calculado a $U(v)$ para $v = \sup f(\theta)$, teoricamente o valor encontrado deve ser 1, ao fazer algumas vezes variando os parâmetros, verificou-se que o valor retornado era 1, assim como o esperado, logo pode-se dizer que a função está coerente com sua definição.

Com isso é possível afirmar que a função gera estimativas boas para o valor da função $W(v)$, já que pode-se perceber que o erro absoluto dela fica abaixo de 0.0005. Além disso pode-se dizer que o tempo que leva é, razoavelmente, baixo, logo o código feito segue as condições impostas no enunciado desse EP.