Exercício Programa 04 - Aproximação de Modelo Estatístico Multinomial m-dimensional

Alexandre Barsam Junqueira - NºUSP: 12561642
Guilherme Lloret Cavalcante - NºUSP: 14576152

13 de maio de 2024

Resumo

Esse relatório busca demonstrar a aproximação de um Modelo Estatístico Multinomial m-dimensional por meio de uma interpolação polinomial $U(v)$ que aproxime a função verdade do modelo $W(v)$. Ao mesmo tempo, também busca ser computacionalmente eficiente.

1 Modelo Estatístico

Considere o modelo estatístico multinomial m-dimensional com observações x, informação a priori y e parâmetro θ, $x, y \in N^m$, $\theta \in \Theta = S^m = \{\theta \in R_+^m : \theta^T \mathbf{1} = 1\}$. Esse modelo estatístico é composto pelas seguintes igualdades.

Potencial a posteriori:

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

Conjunto de corte:

$$T(v) = \{\theta \in \Theta : f(\theta \mid x, y) \leq v\}, \quad v \geq 0$$

Função verdade:

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

$W(v)$ é a massa de probabilidade a posteriori dentro de $T(v)$, ou seja, a massa de probabilidade onde o potencial a posteriori, $f(\theta \mid x, y)$, não excede a cota v. O objetivo é obter uma função $U(v)$ que aproxime adequadamente a função $W(v)$ com $\epsilon \leq 0,05\%$.

2 Abordagem do Problema

A solução consiste nas seguintes etapas:

1. Defina k pontos de corte, $0 = v_0 < v_1 < \cdots < v_k = \sup f(\theta)$, da seguinte forma:

2. Use um gerador de números aleatórios Gamma para gerar n pontos em Θ, $\theta_1, \ldots, \theta_n$, distribuídos de acordo com a função de densidade a posteriori.

3. Use a fração de pontos simulados θ_t dentro de cada bin, $v_{j-1} < f(\theta_t) < v_j$, como uma aproximação de $W(v_j) - W(v_{j-1})$.

4. Ajuste dinamicamente as bordas de cada bin, v_j, para obter bins com pesos aproximadamente iguais, isto é, $W(v_j) - W(v_{j-1}) \approx \frac{1}{k}$.

5. Obtenha como saída uma função $U(v)$ que dê uma boa aproximação de $W(v)$.

2.1 Gerando Pontos segundo a Função Densidade

Observando a função de densidade a posteriori, ela é simplesmente o potencial da densidade Dirichlet

$$Dirichlet(\theta \mid a) = \frac{1}{B(a)} \prod_{i=1}^{m} \theta_i^{a_i - 1}$$

. Portanto, pode-se gerar pontos distribuídos segundo a função densidade a posteriori utilizando a seguinte fórmula:

$$y_i = \frac{x_i}{\sum_{i=1}^{m} x_i}$$

em que $y \sim Dirichlet(a)$ e $x \sim (a, 1)$.

2.2 Definição dos Cortes v_i

Tendo simulados n pontos por meio desse método, pode-se definir, então, os cortes v_i. Definie-se $v_0 = 0$ e $v_k = sup(f) = max(f(\theta))$. Assim, o supremo de $f(\theta)$ é simplesmente o maior valor de f para os pontos θ gerados. Para os outros cortes, busca-se criá-los de tal forma que todos os bins tenham tamanhos semelhantes.

Para isso, defini-se o tamanho dos bins por meio da divisão inteira entre o número de pontos n e o número de bins k. Evidentemente, quando $n \not\equiv 0 \pmod{k}$, os bins não serão todos iguais. Assim, considerando

$$n = q \cdot k + r = (k - r) \cdot q + r \cdot (q + 1)$$

, em que q é o quociente e r o resto, existirão r bins com tamanho $(q + 1)$ e $(k - r)$ com tamanho q.

Assim, basta obter o resultado $f(\theta)$ para cada ponto e ordenar a lista de menor para maior. Com isso, define-se os pontos de corte com base no racional descrito. Logo, tem-se que $W(v_j) - W(v_{j-1}) \approx \frac{1}{k}$.

2.3 Encontrando $U(v)$

Por fim, falta encontrar uma equação $U(v)$ que aproxime $W(v)$. Para isso, será utilizada uma interpolação polinomial que garanta um polinômio monótono crescente. Isso porque, sendo uma função de probabilidade acumulada, necessariamente $W(u) \leq W(v)$ se $u \leq v$.

Por isso, foi utilizado o método Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) para a tarefa. A ideia básica por trás do PCHIP é interpolar localmente segmentos cúbicos de Hermite entre os pontos de dados. Aqui está uma visão geral de como funciona:

1. **Calcula os gradientes locais:** Para cada par de pontos de dados adjacentes, o PCHIP calcula o gradiente (ou derivada) local. Geralmente, isso é feito usando uma diferença finita dividida que leva em conta os valores das funções nos pontos de dados vizinhos.

2. **Constrói segmentos cúbicos:** Com os gradientes calculados, o PCHIP constrói segmentos cúbicos de Hermite locais entre cada par de pontos de dados. Um segmento cúbico de Hermite é uma função polinomial de terceiro grau que é especificada por seus valores e gradientes em ambos os pontos finais do segmento.

3. **Suaviza a curva:** O PCHIP assegura que os segmentos cúbicos adjacentes se juntem suavemente, criando uma curva contínua e diferenciável. Isso é feito garantindo que os gradientes dos segmentos cúbicos coincidam nos pontos de dados compartilhados.

4. **Interpolação:** Uma vez que os segmentos cúbicos locais foram construídos, o PCHIP pode ser usado para interpolar valores em qualquer ponto dentro do intervalo dos pontos de dados. Isso é feito aplicando o segmento cúbico apropriado para cada ponto de interpolação.

O PCHIP é especialmente útil quando os dados têm comportamento oscilatório ou não são uniformemente espaçados. Ele tende a produzir resultados mais suaves e menos suscetíveis a oscilações indesejadas do que outros métodos de interpolação. O método PCHIP também mantém a monotonicidade ajustando os gradientes dos segmentos cúbicos de Hermite para garantir que a função interpoladora seja monótona crescente ou decrescente entre os pontos de dados.

2.4 Erro da Função $U(v)$

O erro da função obtida pode ser divido em dois: (i) margem de erro do estimador $\hat{p_v}$; (ii) margem de erro por discretização em bins. Para o cálculo da primeira parcela, basta entender que o estimador $\hat{p_v}$ de p_v atravpes de n pontos gerados aproxima de uma curva normal quando $n \to \infty$. Isto é:

$$\hat{p_v} \xrightarrow{n \to \infty} N\left(W(v), \frac{W(v)(1 - W(v))}{n}\right)$$

.

Dessa forma, definindo um intervalo de confiança, pode-se obter a fórmula do erro para o estimador como:

$$\epsilon_n = \frac{z_{\gamma/2}}{2\sqrt{n}}$$

.

Ao mesmo tempo, ao observar a discretização da função, esta gerará um erro de, no máximo, $W(v_{j+1}) - W(v_j) \approx 1/k$. Desse modo, pode-se determinar:

$$\epsilon_k = \frac{1}{k}$$

.

Por fim, o erro total será:

$$\epsilon = \epsilon_n + \epsilon_k$$

.

Para que o erro total seja menor que $0,05\%$, é necessario que tanto ϵ_n quanto ϵ_k o sejam. O erro por discretização decresce mais rapidamente com a crescente de k do que o erro do estimador com a crescente de n. Além disso, no processo de computação de $U(v)$, mais processos dependem do valor de n do que do valor de k. Portanto, para preservar uma velocidade alta de computação, compensa deixar um valor mais alto de k do que de n. Assim, valores adequados são:

$$n = 1.7 \cdot 10^4$$

$$k = 5 \cdot 10^6$$

3 Do Algoritmo

3.1 Variáveis Iniciais

O algoritmo possui as seguintes variáveis:

- **n:** número de pontos/vetores θ que serão utilizados.
- **k:** número de bins que serão utilizados para a definição de cortes v_i.
- **m:** representa a dimensionalidade do modelo, influenciando o tamanho dos vetores θ, x e y.
- α**:** soma vetorial das observações (x) e das informações a priori (y) (no código, foi utilizado um gerador de números inteiros aleatórios).

3.2 Funções e Otimizações

As funções utilizadas para a construção da função $U(v)$ foram as descritas abaixo. Para otimizar todo o processo, foi constante a utilização de *numpy* para vetorizar as contas e aumentar a velocidade do processamento.

- $f(\theta, , B(\alpha))$**:** recebe θ e e calcula o potencial a posteriori e depois divide pela constante de normalização $B(a)$. Para acelerar, foi feito o cálculo em log.
- $B(\alpha)$**:** calcula o valor da função beta multivariada do vetor α. Essa função também utiliza log para acelerar o processo.

- $Gerador_Dirichlet(\alpha, n)$**:** gerador de pontos segundo a Distribuição Dirichlet. Para isso, gera diversos vetores x, com $x_i \sim Gamma(\alpha)$, e depois utiliza $y_i = \frac{x_i}{\sum_{i=1}^{m}}$ para gerar o vetor θ.

- **Criar Lista de Cortes (v_i):** define o resto (r) e o tamanho do bin (q) segundo os valores de n e k. Assim, pega-se a lista ordenada dos resultados de $f(\theta, \alpha, B(\alpha))$ e escolhe-se os pontos q, 2q, ..., kq. Porém, considerando a possibilidade de $n \not\equiv 0 \pmod k$, adiciona-se vetorialmente um "ajuste"que faz com que os primeiros r bins tenham $(q+1)$ de tamanho. Além disso, define-se o primeiro corte como 0.