EXERCICIO PROGRAMA 5 - MAP2212 2024

Victor Rocha Cardoso Cruz 11223757
Larissa Aparecida Marques Pimenta Santos 12558620

1 Enunciado

- Reenvie o seu EP 4, substituindo o amostrador exato da distribuição Dirichlet por um amostrador Markov Chain Monte Carlo (MCMC) baseado apenas no conhecimento de um potencial para a distribuição Dirichlet.
- Use como núcleo para o MCMC uma normal multivariada $N(0, \Sigma)$, onde a matriz de covariância é ajustada baseada em seu conhecimento prévio dos parâmetros da Dirichlet e/ou algumas sequências de amostragem iniciais ou de teste.
- No seu relatório, inclua o critério de diagnóstico utilizado para avaliar o progresso do MCMC, para ajustar o núcleo do MCMC, para decidir o número de pontos usados no aquecimento do núcleo do MCMC e para computar W - a função verdade do modelo estatístico.

2 Introdução

Como realizado no exercício anterior, foi solicitado que seja feita uma aproximação da função verdade $W(v)$ (1) através de uma função $U(v)$ e cujo erro não supere 0,05%. A região de integração para essa estimativa é $T(v)$, definida por 2.

$$W(v) = \int_{T(v)} f(\theta \mid x, y)\, d\theta \tag{1}$$

$$T(v) = \{\theta \in \Theta \mid f(\theta \mid x, y) \leq v\} \tag{2}$$

Em que v é um nível preestabelecido, θ é um vetor de parâmetros no simplex m-dimensional e $f(\theta \mid x, y)$ é a uma densidade definida pela expressão 3. Para o cenário desse exercício, os vetores θ são tridimensionais, uma vez que $m = 3$.

$$f(\theta \mid x, y) = \frac{1}{B(x+y)} \prod_{i=1}^{m} \theta_i^{x+y-1} \tag{3}$$

A função densidade f (3) é conhecida como função densidade de probabilidade de Dirichlet, em que x é um vetor de observações e y um vetor de informações a priori, ambos com dimensão m. Nesse exercício, a FDP Dirichlet deve ser gerada utilizando o método de Markov Chain Monte Carlo (MCMC), diferentemente do exercício anterior, no qual utilizou-se a FDP gerada pelo Python através da biblioteca NumPy.

3 Funcionamento do gerador de vetores θ

3.1 Matriz de Covariância

Para a futura geração dos candidatos no gerador, é necessário gerar uma matriz de covariância (2X2, pois um dos elementos é gerado por $1 - (\theta_1 + \theta_2)$). Como a matriz de covariância de uma Distribuição Dirichlet é conhecida, utilizou-se essa informação para fixar a matriz, de modo que não seja necessário ajustar a mesma durante a execução do algoritmo.

3.2 Geração dos passos

Os passos são gerados pelo gerador de normal multivariada do NumPy. Para esse gerador é necessária a matriz de covariância previamente explicada. Esse gerador, como foi visto em aula, permite que a aceitação/rejeição dos candidatos seja feita de maneira mais simples.

Esse gerador normal multivariado gera um vetor inicial com tamanho $2n$, em que n é a quantidade necessária de pontos para que a precisão seja alcançada. Caso ainda seja preciso mais candidatos a serem aceitos, podem ser gerados mais vetores de tamanho n.

Empiricamente, $2n$ se mostrou mais do que o suficiente.

3.3 Admissão dos candidatos

Inicia-se com um candidato conhecido da distribuição (no caso inicial [1/3, 1/3, 1/3]) e calcula-se o seu potencial (potencial atual). Em seguida, soma-se o passo gerado pela normal multivariada e estima-se a função potencial dessa soma. Verifica-se se os valores do vetor a ser avaliado são maiores que 0.

Em seguida, é feita a divisão entre o potencial da soma e o potencial atual. Se essa divisão for maior ou igual a 1, o candidato é aceito automaticamente. Caso essa divisão seja menor que 1, é sorteado um valor uniforme entre [0,1] e, se o valor da divisão for maior que o valor da uniforme, o candidato é aceito, caso não, é rejeitado.

Quando o candidato é aceito, o valor da soma é aceito como o próximo valor da distribuição (novo valor atual), seu valor potencial é armazenado e soma-se 1 ao contador. Quando o candidato é rejeitado, mantém-se no valor atual e testa-se o próximo passo. O processo é repetido até atingir o valor n supra definido. Neste programa, os primeiros 5.000 valores são desconsiderados do conjunto final, essa técnica é utilizada para "aquecer" o *kernel* do MCMC, fornecendo maior convergência à matriz obtida.

4 Estimativa do erro

O cálculo do erro para o algoritimo foi feito da mesma forma que o exercício 4, conforme descritivo abaixo.

4.1 Partições

O enunciado solicitado que a diferença $W(v_j) - W(v_{j-1})$ seja aproximadamente igual a $1/k$. O erro máximo admitido para essa diferença é de 0,05%.

Dessa forma, tem-se:

$$\frac{1}{K} \leq 0,05\%$$
$$K \geq 2000$$

Definiu-se $K = 2.500$ a partir desse critério.

4.2 Número de elementos gerados

Para isso, considera-se que a probabilidade de um ponto qualquer recair numa partição específica é uma variável Bernoulli. A variância de uma variável aleatória Bernoulli é dada pela multiplicação desses dois valores ($\sigma^2 = p(1-p)$). Pode ser mostrado que a variável de Bernoulli possui uma variância máxima igual a 0,25. Assim, essa variância foi escolhida para a estimativa de pontos necessários.

Utiliza-se o Teorema Central do Limite para aproximar-se a Bernoulli por uma Distribuição Normal.

Sabe-se que o erro em uma estimação por intervalo, em uma normal, é dado por:

$$\epsilon = z_{\frac{\gamma}{2}} \frac{\sigma}{\sqrt{n}}$$

Manipulando a expressão acima, é possível isolar n na forma:

$$n = z_{\frac{\gamma}{2}}^2 \frac{\sigma^2}{\epsilon^2}$$

Substituindo $z_{\frac{\gamma}{2}}$ por 1,96, quantil da normal correspondente à 95% de taxa de acerto, chega-se a um valor estipulado para n:

$$n = 3.841.600$$

Adotou-se:

$$n = 3.850.000$$

Essa é a quantidade necessária para o programa ficar dentro da precisão requerida 95% das vezes.

5 Implementação do Programa

Inicialmente, são estabelecidos os valores padrão para os vetores X e Y. Geram-se 3.850.000 valores da função potencial, utilizando o gerador previamente explicado.

Os valores da função potencial são inseridos num vetor, chamado vetor *valores*, que é posteriormente ordenado e separado nas 2.500 partes. São colocados em um segundo vetor, chamado *vetor fronteira*, os maiores pontos de cada parte.

É solicitado que o usuário insira um valor v maior ou igual a zero. Esse número é multiplicado pela constante de normalização e é buscada sua posição no vetor fronteira. Essa posição é multiplicada por 1.540 e o resultado alocado em uma variável.

A posição encontrada no vetor fronteira refere-se a uma partição específica. Dessa forma, também é realizada a busca da posição do valor v nessa partição. Esta posição é somada ao valor atual da variável. A aproximação de $W(v)$ é dada pela divisão entre o valor da variável e o valor total de pontos gerados. Essa aproximação é arredondada para 4 casas decimais e apresentada ao usuário.

6 Conclusão

Os resultados do programa, para as entradas padrão fornecidas (x = [1, 2, 3] e y = [4, 5, 6]) foram comparadas com sucesso aos resultados esperados. O tempo de execução é principalmente devido à aceitação e rejeição, o que demanda um maior refino na quantidade de pontos necessários para o programa funcionar adequadamente e num tempo razoável.

Tempo considerável foi reduzido ao produzir uma quantidade alta de passos de uma vez, ainda que ocupe mais memória. O método sugerido pelo enunciado, utilizando passos normais $[0, \Sigma]$, permite essa manipulação.

Em uma máquina mediana, o tempo de execução não leva mais que 80 segundos, o que foi considerado bastante razoável. O método implementado no EP 4, em utilizar a Função Potencial e não a FDP, ou seja, trabalhar inicialmente sem a constante de normalização, aumenta a velocidade de execução, sem prejuízo dos resultados finais.