Modelos Ocultos de Markov

Parte III. Estimação dos parâmetros.

Fernando Lucambio

Departamento de Estatística
Universidade Federal do Paraná

Agosto, 2020

Como vimos na Seção II.3, a verossimilhança L_T de um HMM com base nas observações $s_1, s_2, \cdots, s_T$ pode ser escrita como pode ser escrita como

$$L_T = P(S_1 = s_1, \cdots, S_T = s_T) = \delta B_1 \cdots B_T \mathbf{1}^\top,$$

onde

$$B_t = \Gamma P(s_t) = \begin{pmatrix} \gamma_{1,1} & \cdots & \gamma_{1,m} \\ \vdots & \ddots & \vdots \\ \gamma_{m,1} & \cdots & \gamma_{m,m} \end{pmatrix} \begin{pmatrix} p_1(s_t) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p_m(s_t) \end{pmatrix}.$$

Agora, lembre-se que no início da Seção II.1 introduzimos a notação para a história de um HMM $\{S_t\}$ até o tempo t, ou seja, $S^{(t)}$. A mesma notação pode ser usada para o histórico das observações e se obtém:

$$\left.\begin{array}{rcl} S^{(t)} & = & \{S_1, \cdots, S_t\} \\ s^{(t)} & = & \{s_1, \cdots, s_t\} \end{array}\right\} \text{história ate ao tempo t}.$$

Além disso, é possível introduzir uma nova notação também para a L_T. A verossimilhança L_T pode ser reescrita como

$$L_T = \underbrace{\delta B_1 \cdots B_t}_{\alpha_t} \underbrace{B_{t+1} \cdots B_T \mathbf{1}^\top}_{\beta_t^\top} = \alpha_t \beta_t^\top \qquad \text{para qualquer } t \in \{1, 2, \cdots, T\},$$

onde β_T é definida como $\beta_T^\top = \mathbf{1}^\top$ e $\alpha_0 = \delta$. Note que tanto α_t como β_t são vectores com m entradas, nomeadamente

$$\alpha_t = (\alpha_t(1), \cdots, \alpha_t(m)) \qquad \text{e} \qquad \beta_t = (\beta_t(1), \cdots, \beta_t(m)),$$

respectivamente. No que se segue, a principal preocupação é a interpretação dos vetores α_t e β_t.

Teorema III.1

Seja $\{S_t, C_t : t \in \mathbb{N}\}$ um Modelo Oculto de Markov. Então o vetor $\alpha_t = (\alpha_t(1), \cdots, \alpha_t(m))$ contém os seguintes vetores, chamados probabilidades forward:

$$\alpha_t = \left(P(S^{(t)} = s^{(t)}, C_t = 1), \cdots, P(S^{(t)} = s^{(t)}, C_t = m)\right).$$

O enunciado do Teorema III.1 significa que cada componente $\alpha_t(i), i = 1, 2, \cdots, m$ do vector α_t pode ser interpretado como a probabilidade de a história do HMM até o tempo t e da Cadeia de Markov estar no estado i no tempo t. Por exemplo, para $t = 10$, $\alpha_{10}(2)$ é a probabilidade do histórico até $t = 10$ e a Cadeia de Markov estar no estado 2 quando $t = 10$.

O vetor

$$\beta_t = \big(\beta_t(1), \beta_t(2), \cdots, \beta_t(i), \cdots, \beta_t(m)\big)$$

pode ser interpretado de maneira similar. Ele contém as chamadas probabilidades de backward $\beta_t(i)$, onde

$$\beta_t(i) = P\big(S_{t+1} = s_{t+1}, S_{t+2} = s_{t+2}, \cdots, S_T = s_T \mid C_t = i\big). \tag{1}$$

Isso significa que $\beta_t(i)$ representa uma probabilidade condicional, isto é, a probabilidade das futuras observações $s_{t+1}, \cdots, s_T$ dado que a Cadeia de Markov está no estado i no momento t.

Um método para estimar os parâmetros de um HMM é o algoritmo Baum-Welch, um exemplo do que mais tarde se tornou conhecido como o Algoritmo EM (Baum et al., 1970). Para detalhes da descrição do algoritmo EM e sua implementação no contexto dos HMMs nos referimos a Baum et al. (1970) e MacDonald and Zucchini (1997).

Originalmente, o algoritmo EM foi desenvolvido para estimar os parâmetros de um modelo no caso de dados faltantes. Por exemplo, imagine a situação em que uma ou mais observações estão faltando para uma certa experiência. Seja $Y = (Y^{obs}; Y^{mis})$ denotando todas as observações, Y^{obs} denota as observações conhecidas e Y^{mis} os valores dos dados em falta. Considere também que o desenho do experimento leva a supor que todas as observações são distribuídas de acordo com uma função de distribuição paramétrica com o parâmetro θ.

O algoritmo EM lida com estes problemas da seguinte forma. Primeiramente, um palpite inicial para o parâmetro tem que ser feito, denotado por θ^0. Ele pode basear-se na experiência ou ser uma escolha aleatória. Então, os dois passos seguintes têm de ser executados alternadamente.

Passo E

Calcular a esperança condicional das observações em falta, tendo em conta os dados observados e θ^t, E($Y^{mis} \mid Y^{obs}, \theta^t$), onde $\theta^t = \theta^0$ no início do algoritmo. Em seguida, avaliar a log-verossimilhança dos dados completos, substituindo as funções dependendo de Y^{mis} pelas funções correspondentes dependendo da esperança condicional; isto produz a esperada log-verossimilhança. Para ilustrar o que acontece, pode-se imaginar que os valores em falta sejam substituídos por valores estimados com base no parâmetro θ^t assumido e das observações Y^{obs} conhecidas, mesmo que isso não seja correto do ponto de vista matemático.

Paaso M

Maximizar a esperança da log-verossimilhança em relação a θ. O resultado é um aumento da esperança da log-verossimilhança, bem como um novo parâmetro θ^{t+1}.

Os dois passos se repetem até que o aumento da log-verossimilhança caia abaixo de um determinado limite ou um certo número de repetições sejam levados a cabo.

As estimativas resultantes de θ convergem para valores que maximizam a verossimilhança do modelo. A propriedade de maior endereçamento do algoritmo EM é que a log-verossimilhança aumenta monotonicamente na sequência de iterações e converge para um valor estacionário. No entanto, a taxa de convergência pode tornar-se muito lenta se faltarem muitas observações.

No contexto dos HMMs, os estados não observados ocupados pela Cadeia de Markov são considerados como as observações em falta. Então, os dois passos do algoritmo EM podem ser implementados usando as probabilidades de avanço (forward) e recuo (backward) se soluções fechadas estiverem disponíveis para o passo M e os valores em falta não ocorrerem.

Outros métodos de estimação:

- Maximização direta da verossimilhança
- Maximização com restrições nos parâmetros

Exemplo III.2: Algoritmo EM.

Utilizamos os dados de série de vendas semanais de um produto de sabão específico para mostrarmos uma das formas de estimar os parâmetros de um modelo HMMs Poisson com dois estados. Neste caso foi utilizada a função **depmix**, na libraria **depmixS4**, para especificarmos o modelos a ser ajustado na forma de Modelo Oculto de Markov, a opção **nstates** indica quantos estados consideramos na Cadeia de Markov oculta e com a opção **family** que o modelo seja Poisson.

```
> library(depmixS4)
> (modelo02 = depmix(soap ~ 1, data = dados1, nstates = 2, family=poisson()))
Initial state probabilties model
pr1 pr2
0.5 0.5

Transition matrix
       toS1 toS2
fromS1  0.5  0.5
fromS2  0.5  0.5

Response parameters
Resp 1 : poisson
    Re1.(Intercept)
St1               0
St2               0
```

Exemplo III.2: Algoritmo EM (continuação).

Agora utilizamos o comando **fit** para executar o algoritmo EM construído anteriormente e guardado em **modelo02**. Posteriormente, com **summary** mostramos as estimativas da matriz de probabilidades de transição e dos parâmetros λ das funções de probabilidade Poisson.

```
> (modelo02.fit = fit(modelo02))
.........................................
converged at iteration 24 with logLik: -618.4545
Convergence info: Log likelihood converged to within tol. (relative change)
log Lik. -618.4545 (df=5)
AIC:  1246.909
BIC:  1264.354
> summary(modelo02.fit)
Transition matrix
        toS1  toS2
fromS1 0.911 0.089
fromS2 0.367 0.633

Response parameters
Resp 1 : poisson
    Re1.(Intercept)
St1           1.391
St2           2.429
```

Exemplo III.3: Retornos

S& P 500, abreviação de Standard & Poor's 500, trata-se de um índice composto por quinhentos ativos (ações) cotados nas bolsas de NYSE ou NASDAQ, qualificados devido ao seu tamanho de mercado, sua liquidez e sua representação de grupo industrial. No Yahoo finance, a fonte de dados, o ticker do S& P 500 é ^GSPC.

Aqui não estamos interessados nem no valor do índice nem no seu retorno, estamos interessados na codificação resultante da Cadeia de Markov subjacente, não observada, a qual deve indicar os períodos de alta, estável ou baixa do mercado; caso a cadeia seja de três estados e alta ou baixa, caso seja de dois estados a cadeia do modelo estimado adequado aos dados.

Leitura e apresentação dos dados obtidos:

```
> library(quantmod);library(mhsmm);library(depmixS4)
> getSymbols.yahoo("^GSPC", env=globalenv(), from="2000-02-01", to="2019-12-31",
            periodicity="daily")
[1] "^GSPC"
> GSPCr = diff(log(GSPC$GSPC.Close))[-1] # retornos aproximados
> plot(GSPCr, main="S&P 500 Retornos", type="n")
> lines(GSPCr)
```

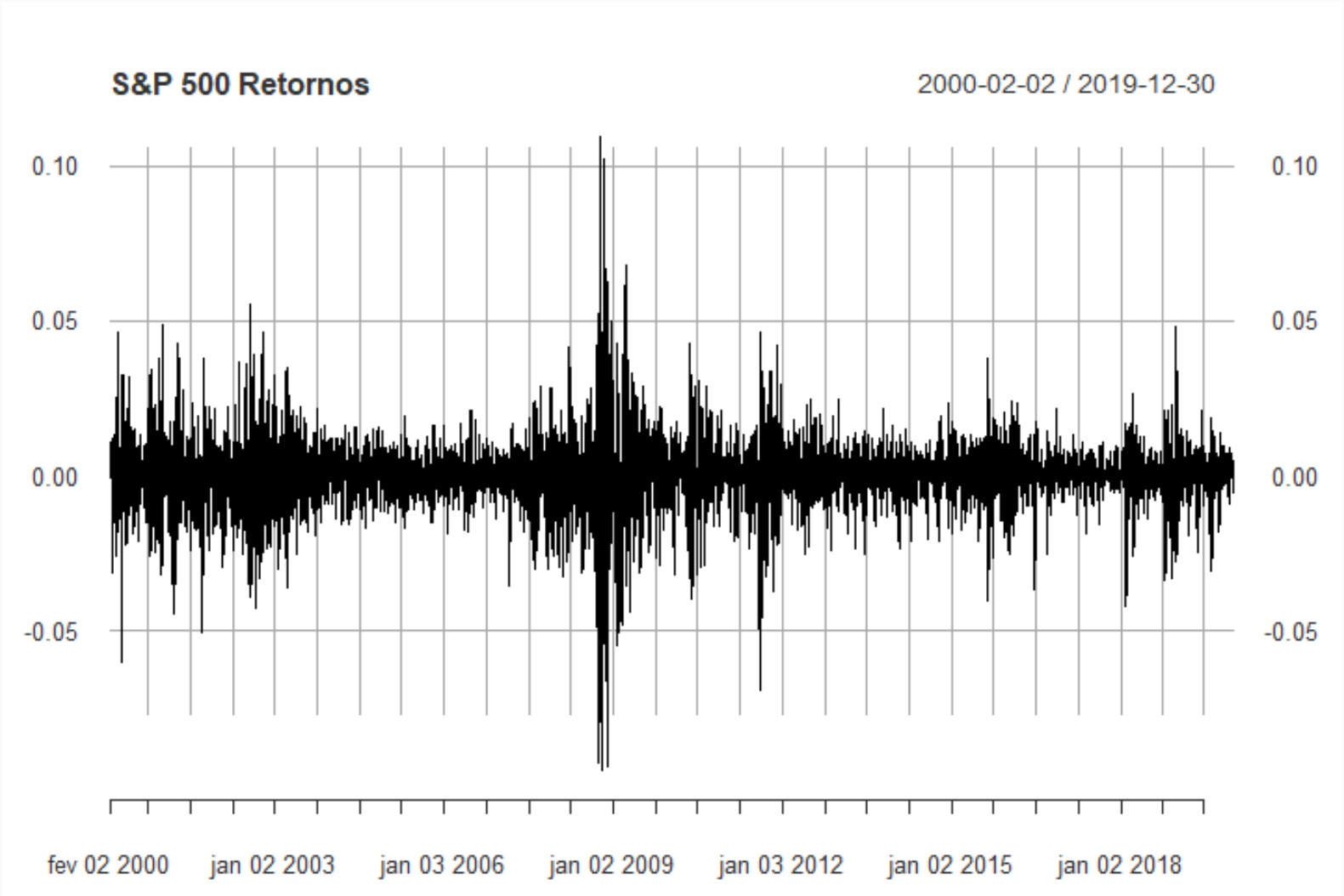

Figura III.1: Retornos (log-retornos) do índice S&P 500.

Exemplo III.3: Retornos (continuação).

```
> Modelo.Data = data.frame(GSPCr) # criar o arquivo de dados para o nosso modelo HMM
> HMM = depmix(GSPC.Close~1, data = Modelo.Data, nstates = 3, family=gaussian())
> HMMfit = fit(HMM, verbose = FALSE)
converged at iteration 82 with logLik: 16331.63
> summary(HMMfit)
Initial state probabilities model
pr1 pr2 pr3
  0   1   0

Transition matrix
        toS1  toS2  toS3
fromS1 0.967 0.026 0.008
fromS2 0.023 0.977 0.000
fromS3 0.033 0.000 0.967

Response parameters
Resp 1 : gaussian
    Re1.(Intercept) Re1.sd
St1           0.000  0.012
St2           0.001  0.005
St3          -0.001  0.026
```

Exemplo III.3: Retornos (continuação).

O ajuste acontece quando executamos o comando **fit** e, perceba que antes do ajuste devemos indicar o modelo, isto é feito com o comando **depmix**.

```
> HMM1 = depmix(GSPC.Close~1, data = Modelo.Data, nstates = 2, family=gaussian())
> HMMfit1 = fit(HMM1, verbose = FALSE)
converged at iteration 43 with logLik: 16113.4
> summary(HMMfit1)
Initial state probabilities model
pr1 pr2
  0   1

Transition matrix
        toS1  toS2
fromS1 0.974 0.026
fromS2 0.012 0.988

Response parameters
Resp 1 : gaussian
    Re1.(Intercept) Re1.sd
St1          -0.001  0.019
St2           0.001  0.007
> llratio(HMM,HMM1)
log Likelihood ratio (chi^2): 0 (df=7), p=1.
```

A maximização numérica direta da verossimilhança tem algumas vantagens sobre o algoritmo EM, não menos importante sendo a facilidade com que o software para um modelo específico pode ser modificado para se adequar a modelos alternativos e mais complexos. Outra vantagem é que o método lida facilmente com observações ausentes. Por sua vez, isto facilita, por exemplo, a verificação de outliers.

No entanto, dois grandes problemas ocorrem quando se aplica a maximização numérica direta da verossimilhança para estimar os parâmetros de um HMM.

- Os parâmetros são limitados, ou seja, as estimativas de máxima verosimilhança para a matriz de probabilidades de transição Γ e a maioria dos parâmetros da distribuição de estado dependente, por exemplo, λ no caso de um HMM Poisson, não é permitido tomar todos os valores nos reais.

- Ocorre subfluxo numérico, ou seja, à medida que o T aumenta a L_T torna-se menor e pode ser arredondada a zero pelo software para o T grande.

Restrições nos parâmetros

Como já mencionado acima, usando a maximização numérica direta da verossimilhança para a estimação dos parâmetros tem de se considerar que, como a maioria dos parâmetros são restritos, é necessário realizar uma otimização linearmente limitada ou reparametrizar o modelo para utilizar algoritmos do tipo quase-Newton oferecidos nos software estatístico padrão como o R (Zucchini and MacDonald, 1998).

Subfluxo numérico

Tendo superado as restrições dos parâmetros é preciso lidar com um segundo problema, ou seja, um subfluxo numérico. A solução usual para o problema do subfluxo numérico, ou seja, a maximização da log-verossimilhança em vez da maximização da verossimilhança, não pode ser implementada tão facilmente uma vez que a verossimilhança é um produto de matrizes. No entanto, pode-se derivar uma fórmula fechada para a log-verossimilhança de um HMM aplicando um truque que, basicamente, produz um algoritmo alternativo para avaliar a verossimilhança recursivamente via probabilidades escalonadas para frente.

Exemplo III.4: Retornos (continuação)

Vamos utilizar o Exemplo III.3 como exemplo da optimização direta da verossimilhança. Acontece que o comando fit, no pacote **depmixS4**, ajusta os modelos pelo algoritmo EM se não houverem restrições sobre os parâmetros. Caso contrário, os optimizadores gerais **solnp**, o padrão (do pacote **Rsolnp**) ou **donlp2** (do pacote **Rdonlp2**) são utilizados e lidam com restrições gerais de desigualdades lineares. Estes optimizadores são selecionados definindo **method='rsolnp'** ou **method='donlp'** respectivamente.

Três tipos de restrições podem ser especificados nos parâmetros: restrições fixas, restrições de igualdade, e restrições gerais lineares de desigualdade. Os vectores de restrições devem ser de comprimento **npar(objeto)**, por exemplo:

```
> npar(HMMfit1)
[1] 10
```

neste caso.

O argumento de igualdade é utilizado para especificar restrições de igualdade: os parâmetros que obtêm o mesmo número inteiro neste vector são estimados como sendo iguais. Qualquer número inteiro pode ser utilizado desta forma, excepto 0 e 1, que indicam parâmetros fixos e livres, respectivamente.

Exemplo III.4: Retornos (continuação)

Utilizando **solnp** ou **donlp2** é executado o algoritmo Newton-Raphson para estimar os parâmetros sujeitos a restrições lineares através da imposição:

$$b_l \leq A \times x \leq b_u,$$

onde x é o vetor de parâmetros, b_l é um vector de limites inferiores, b_u é um vector de limites superiores e A é a matriz de restrições.

O argumento **conrows** é utilizado para especificar diretamente as linhas de A e os argumentos **conrows.lower** e **conrows.upper** para especificar os limites das restrições. **conrows** deve ser uma matriz de colunas **npar(objeto)** e uma linha para cada restrição, um vector no caso de uma única restrição.

No referido exemplo fixamos a distribuição inicial da Cadeia de Markov como 0.5 para cada estado. Os parâmetros da matriz de probabilidades de transição foram considerados livres e os parâmetros da distribuição normal foram considerados fixos.

Exemplo III.4: Retornos (continuação)

```
> parametros = c(unlist(getpars(HMMfit1))); parametros[1] = parametros[2] = 0.5
> HMM2 = setpars(HMMfit1,parametros)
> fixos = c(TRUE,TRUE,rep(FALSE,4),rep(TRUE,4))
> HMMfit2 = fit(HMM2, fixed=fixos, method='rsolnp')

Iter: 1 fn: -16113.6081  Pars:  0.98831 0.01169 0.02590 0.97410
solnp--> Completed in 1 iterations
> summary(HMMfit2)
Initial state probabilities model
pr1 pr2
0.5 0.5

Transition matrix
        toS1  toS2
fromS1 0.988 0.012
fromS2 0.026 0.974

Response parameters
Resp 1 : gaussian
    Re1.(Intercept) Re1.sd
St1           0.001  0.007
St2          -0.001  0.019
```

Com o conhecimento das seções anteriores, as estimativas dos parâmetros $\widehat{\Theta} = (\widehat{\Gamma}, \widehat{\Lambda})$ podem ser obtidas maximizando a verossimilhança dos dados fornecidos, observe que a definição de $\widehat{\Theta}$ implica que se está lidando com um HMM Poisson. Para outros modelos o vetor $\widehat{\Lambda}$, na definição de $\widehat{\Theta}$ tem que ser substituído pelo vetor de parâmetros apropriado. A precisão das estimativas não pode ser calculada facilmente, apenas alguns resultados assintóticos estão disponíveis. Sob certas condições, $\widehat{\Theta}$ é consistente e assintoticamente normal, para detalhes ver MacDonald and Zucchini (1997). Contudo, para aplicações práticas, os resultados levam a alguns problemas.

- A amostra é geralmente de tamanho finito T, portanto a intensidade do comportamento assintótico torna-se um aspecto importante.
- Assumindo a normalidade, como pode ser calculado o erro padrão $SE(\widehat{\Theta})$?
- As entradas de $\widehat{\Theta}$ estão correlacionadas. Portanto, a interpretação não é trivial uma vez que os parâmetros estão ligados. Este problema, também ocorrendo em outros contextos, leva ao fato de que as declarações podem ser feitas para os erros padrão dos parâmetros individuais, separadamente, enquanto o erro de todo o modelo, que é mais interessante, não pode ser calculado diretamente.

Uma solução possível para os problemas acima mencionados é o uso do método paramétrico bootstrap.

O bootstrap paramétrico assume que o modelo ajustado com os parâmetros estimados $\widehat{\Theta}$ é o verdadeiro modelo dos dados. Então, a fim de obter as propriedades distribucionais de $\widehat{\Theta}$ os seguintes passos são repetidos para $b \in \{1, 2, \cdots, B\}$.

Gerar uma nova amostra b de observações de comprimento T a partir do modelo ajustado com parâmetros $\widehat{\Theta}$, normalmente, o comprimento deve ser o mesmo que o número de observações originais. Esta amostra é chamada de amostra de bootstrap b.

Estimar o vector dos parâmetros $\widehat{\Theta}_b^*$ para a amostra bootstrap b.

Este procedimento que produz $\widehat{\Theta}_1^*, \cdots, \widehat{\Theta}_B^*, B$ vectores de estimativas dos parâmetros e portanto uma distribuição empírica das estimativas dos parâmetros $\widehat{\Theta}$.

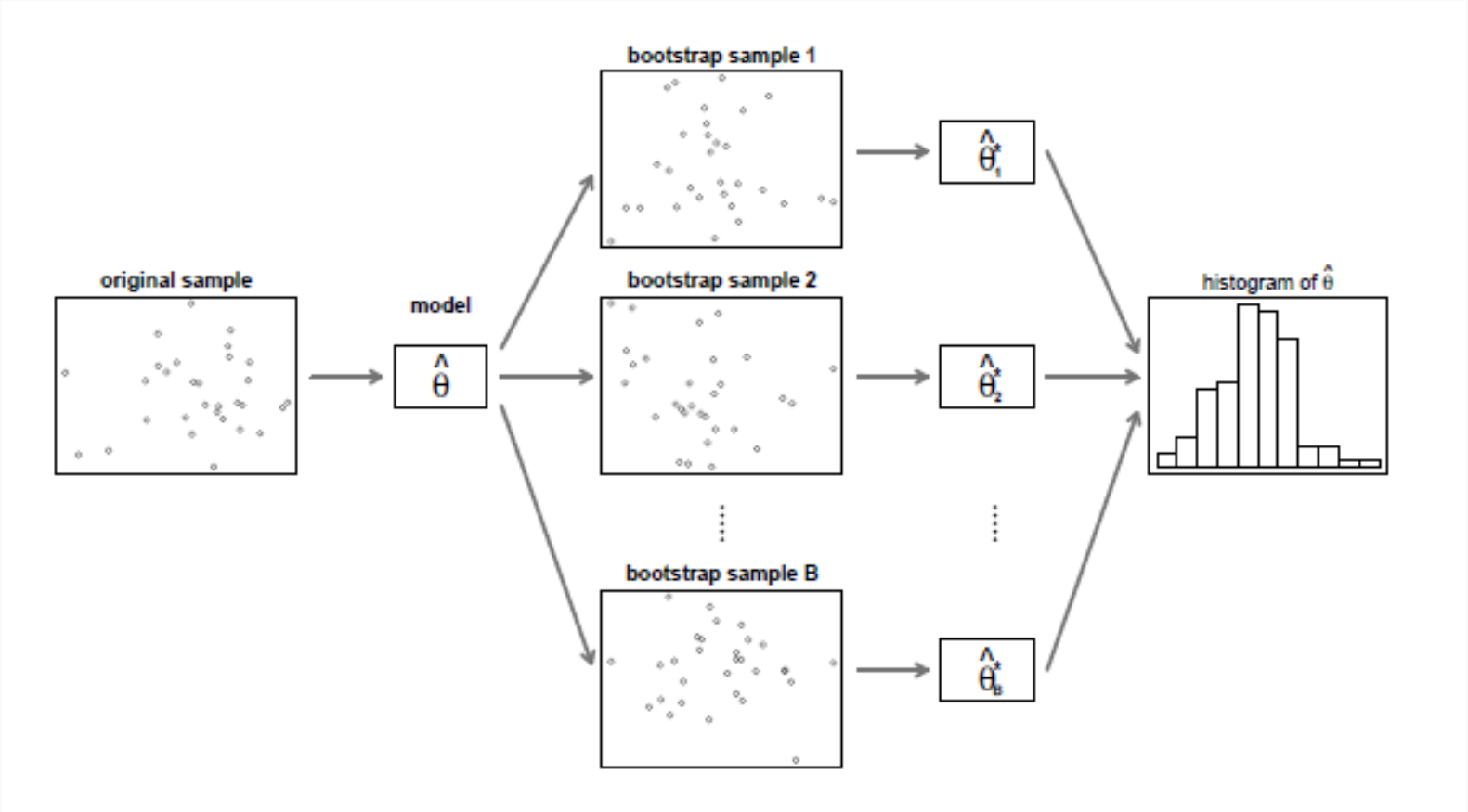

Figura III.2: Bootstrap paramétrico.

Exemplo III.4: Bootstrap.

Utilizando os dados e resultados do Exemplo III.2, mostramos como implementar as estimativas bootstrap nesta situação.

```
> ## -- modelo -- ##
> modelo = modelo02.fit
> ## -- MLEs -- ##
> para.mle = as.vector(getpars(modelo)[-(1:modelo@nstates)])
> ## -- Bootstrap -- ##
> set.seed(666); n.obs = length(soap); n.boot = 10
> para.star = matrix(NA, nrow = n.boot, ncol = (modelo@npars-modelo@nstates))
> n.param = (modelo@nstates)^2
> respst = para.mle[(n.param+1):(modelo@npars-modelo@nstates)];
                                                    trst = para.mle[1:(n.param)]
> for(nb in 1:n.boot){
   mod = simulate(modelo02)
   y.star = as.vector(mod@response[[1]][[1]]@y)
   dfy = data.frame(y.star)
   mod.star = depmix(y.star~1, data = dfy, respst = respst, trst = trst,
                                          nst = modelo@nstates, family = poisson())
   fm.star = fit(mod.star, emcontrol = em.control(tol = 1e-5), verbose = FALSE)
   para.star[nb,] = as.vector(getpars(fm.star)[-(1:modelo@nstates)])
 }
```

Exemplo III.4: (continuação).

Utilizando os dados e resultados do Exemplo III.2, mostramos como implementar as estimativas bootstrap nesta situação.

```
> permu = matrix(c(0,1,1,0), 2,2)
> SE = sqrt(apply(para.star, 2, var) + (apply(para.star, 2, mean)-para.mle)^2)
> (SE.mtrans.mle = permu%*%round(t(matrix(SE[1:4],2,2)),3)%*%permu)
      [,1]  [,2]
[1,] 0.025 0.025
[2,] 0.071 0.071
> (SE.norms.mle = round(matrix(SE[5:6], 1,2), 3)%*%permu)
      [,1]  [,2]
[1,] 1.388 2.572
```

Observe que, para melhor visualização, $B = 10$ neste caso, ou seja, o número de réplicas bootstraps é 10 um valor pequeno, em situações práticas costuma-se utilizar 100, 500 ou mais réplicas Monte Carlo. O resultado deste exercício indica que o desvio padrão dos parâmetros da matriz de probabilidades de transição é 0.025 na primeira linha e 0.071 na segunda linha. Os desvios padrão dos coeficientes da distribuição Poisson são 1.388 e 2.572, respectivamente.