UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E COMPUTAÇÃO
DEPARTAMENTO DE ESTATÍSTICA

# Compressão de dados baseada nos modelos de Markov mínimos

**Karina Yuriko Yaginuma**

Orientador: Prof. Dr. Jesus Enrique Garcia

Dissertação apresentada junto ao Departamento de Estatística do Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas, para obtenção do Título de MESTRE em Estatística.

Campinas
2010

† Este trabalho contou com o apoio financeiro do CAPES.

COMPRESSÃO DE DADOS BASEADA NOS
MODELOS DE MARKOV MÍNIMOS

Este exemplar corresponde à redação final da dissertação devidamente corrigida e defendida por Karina Yuriko Yaginuma e aprovada pela comissão julgadora.

Campinas, 12 de março de 2010.

Prof. Dr. Jesus Enrique Garcia
Orientador

Banca Examinadora:
1. Prof. Dr. Jesus Enrique Garcia (IMECC - UNICAMP)
2. Profa. Dra. Nancy Lopes Garcia (IMECC - UNICAMP)
3. Prof. Dr. Jefferson Antonio Galves (IME - USP)

Dissertação apresentada ao Instituto de Matemática, Estatística e Computação Científica, UNICAMP, como requisito parcial para obtenção do Título de MESTRE em ESTATÍSTICA.

**FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP**
Bibliotecária: Maria Fabiana Bezerra Müller – CRB8 / 6162

Yaginuma, Karina Yuriko
Y1c Compressão de dados baseada nos modelos de Markov mínimos/ Karina Yuriko Yaginuma -- Campinas, [S.P. : s.n.], 2010.

Orientador : Jesus Enrique Garcia
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1.Markov, Processos de. 2.Compressão de dados (Computação). 3.Comprimento Mínimo de Descrição (Teoria da informação) . I. Garcia, Jesus Enrique. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Título em inglês: Data compression based on the minimal Markov models

Palavras-chave em inglês (Keywords): 1. Markov processes. 2. Data compression (Computer science). 3. Minimum description length (Information theory).

Área de concentração: Probabilidade e Estatística

Titulação: Mestre em Estatística

Banca examinadora: Prof. Dr. Jesus Enrique Garcia (IMECC-UNICAMP)
Profa. Dra. Nancy Lopes Garcia (IMECC-UNICAMP)
Prof. Dr. Jefferson Antonio Galves (IME-USP)

Data da defesa: 12/03/2010

Programa de Pós-Graduação: Mestrado em Estatística

**Dissertação de Mestrado defendida em 12 de março de 2010 e aprovada**

**Pela Banca Examinadora composta pelos Profs. Drs.**

**Prof(a). Dr(a). JESUS ENRIQUE GARCIA**

**Prof(a). Dr(a). NANCY LOPES GARCIA**

**Prof(a). Dr(a). JEFFERSON ANTONIO GALVES**

# Agradecimentos

Agradeço primeiramente aos meus pais, Marlene Hatsuyo Yaginuma e Yuji Yaginuma, pelo apoio e incentivo que sempre me deram durante todos esses anos.

Ao professor Jesus Enrique Garcia, pela orientação, dedicação, apoio e por sempre estar presente durante a elaboração desta dissertação.

Ao professor Antonio Galves e a professora Verônica González-López, pela ajuda e orientações que me proporcionaram durante a conclusão da dissertação.

As minhas queridas irmãs, Vanessa, Erika e Simony, e aos meus queridos amigos, Letícia, Mariana, Fernanda, Atahualpa e Vinícius, por estarem sempre do meu lado, nos momentos bons e ruins.

Aos amigos que fiz dentro do IMECC. Em especial, Aldo, Marta e Márcio.

Aos participantes da banca examinadora, pelas sugestões.

A Capes, pelo apoio financeiro a este projeto.

# Resumo

Nesta dissertação é proposta uma metodologia de compressão de dados usando Modelos de Markov Mínimos (MMM). Para tal fim estudamos cadeias de Markov de alcance variável (VLMC, Variable Length Markov Chains) e MMM. Apresentamos então uma aplicação dos MMM à dados linguísticos. Paralelamente estudamos o princípio MDL (*Minimum Description Length*) e o problema de compressão de dados. Propomos uma metodologia de compressão de dados utilizando MMM e apresentamos um algoritmo próprio para compressão usando MMM. Comparamos mediante simulação e aplicação a dados reais as características da compressão de dados utilizando as cadeias completas de Markov, VLMC e MMM.

# Abstract

In this dissertation we propose a methodology for data compression using Minimal Markov Models (MMM). To this end we study Variable Length Markov Chains (VLMC) and MMM. Then present an application of MMM to linguistic data. In parallel we studied the MDL principle (Minimum Description Length) and the problem of data compression. We propose a method of data compression using MMM and present an algorithm suitable for compression using MMM. Compared through simulation and application to real data characteristics of data compression using the complete Markov chains, VLMC and MMM.

# Sumário

**1 Introdução** **1**

**2 Princípio MDL e compressão de dados** **3**

2.1 Princípio MDL . . . . . 3

2.2 Alguns conceitos de teoria da informação . . . . . 4

2.3 Compressão de dados . . . . . 8

**3 Modelos de Markov mínimos** **17**

3.1 Cadeia de Markov de alcance variável . . . . . 18

3.2 Modelos de Markov Mínimos . . . . . 20

**4 Um algoritmo de seleção de modelo de Markov mínimo** **25**

4.1 Seleção de modelos de Markov mínimo usando BIC . . . . . 25

4.2 Algoritmo MMM . . . . . 30

4.3 Simulação . . . . . 32

**5 Compressão de dados baseada nos modelos de Markov mínimos** **35**

5.1 Algortimo de compressão de dados . . . . . 37

5.2 Simulação . . . . . 41

**6 Aplicação à linguística** **47**

6.1 Apresentação dos dados . . . . . 48

6.2 Resultados obtidos da aplicação do algoritmo . . . . . 49

**7 Aplicação à compressão de dados** **53**

7.1 Apresentação dos dados . . . . . 53

7.2 Resultados . . . . . 54

**8 Conclusão** **57**

**A Resultados Auxiliares** **61**

**B Descrição das amostras - Poetas** **63**

**C Implementação dos algoritmos** **65**

C.1 Algoritmo MMM . . . . . 65

C.2 Algoritmos de compressão de dados baseada nos modelos de Markov mínimos 70

C.2.1 Algoritmo codificação de Huffman . . . . . 70

C.2.2 Algoritmo de compressão usando Huffman . . . . . 75

C.2.3 Algoritmo codificação de Huffman em blocos . . . . . 77

C.2.4 Algoritmo de compressão usando Huffman em blocos . . . . . 82

# Capítulo 1

# Introdução

Nesta dissertação consideramos processos estacionários discretos sobre um alfabeto finito $A$. Cadeias de Markov de ordem $M$ são amplamente utilizadas para modelar processos estacionários com memória finita. Um problema com as cadeias de Markov de ordem $M$ é que o número de parâmetros a serem estimados cresce exponencialmente com a ordem do modelo. Outro problema é que a classe de cadeias de Markov completas não é muito rica, fixando o alfabeto existe apenas um modelo para cada ordem. Uma classe mais rica de modelos de Markov de ordem finita são as cadeia de Markov de alcance variável (VLMC), introduzidas por (Rissanen, 1983) e (Buhlmann & Wyner, 1999). As VLMC consideram apenas os passados relevantes para a determinação do próximo símbolo, o tamanho dessa porção relevante varia de um passado para outro e é chamada de *contexto*. Dentro da classe das VLMC, cada modelo pode ser identificado pelo conjunto de contextos. O conjunto de contextos de uma VLMC pode ser representada por uma árvore de contexto. As VLMC têm sido estudadas e utilizadas na modelagem de vários problemas práticos, como por exemplo na análise de dados linguísticos (Galves et al., 2009), na classificação de proteínas (Leonardi, 2007) e na identificação de genes (Bulhmann & Wyner, 1999).

Baseado no princípio MDL (*Minimum Description Length*), introduzimos uma nova classe de modelos de Markov de ordem finita. A idéia do princípio MDL é que o melhor modelo para uma dada amostra, é o modelo que melhor comprime os dados considerando tanto a codificação da amostra como a do modelo. Baseado nessa idéia estamos interessados nos modelos com o menor número de parâmetros, ou seja, queremos minimizar o

tamanho da codificação do modelo reduzindo o número de parâmetros a serem enviados pela codificação. Assim surge a classe de modelos de Markov Mínimos (MMM), introduzida por (García & González-López, 2010), cada modelo é determinado pela escolha de uma partição do espaço de estados, essa classe de modelos incluem as cadeias de Markov completas de ordem finita e as VLMC.

O objetivo desta dissertação é propor uma metodologia de compressão de dados baseada nos MMM e apresentar um algoritmo de compressão usando os MMM. Abordamos o problema de seleção de modelo dentro da classe dos MMM, sabendo-se que o modelo pode ser selecionado consistentemente usando o Critério da Informação Bayesiana (BIC), (García & González-López, 2010), também conhecido como critério de Schwarz (Schwarz, 1978). O critério BIC foi primeiramente utilizado para estimar a ordem de uma cadeia de Markov, em (Csiszár & Talata, 2006) foi provado que o critério BIC pode ser usado para escolher consistentemente o modelo VLMC, com memória não necessariamente limitada.

A dissertação está organizada da seguinte maneira. No Capítulo 2 introduzimos o conceito do princípio MDL para seleção de modelos, apresentamos também alguns conceitos e resultados de compressão de dados e introduzimos alguns conceitos da teoria da informação.

No Capítulo 3 apresentamos as VLMC e os MMM, ilustramos a relação entre os modelos. No Capítulo 4 apresentamos o algoritmo MMM para seleção de modelos de Markov Mínimos baseado no critério BIC.

No Capítulo 5 introduzimos uma metodologia de compressão de dados baseada nos MMM e apresentamos um algoritmo de compressão de dados.

No Capítulo 6 utilizamos o algoritmo MMM para seleção de modelos de dados linguísticos, mais especificamente em um conjunto de poemas codificados. No Capítulo 7 aplicamos o algoritmo de compressão de dados apresentado no Capítulo 5 no mesmo conjunto de poemas, comparamos os resultados da compressão baseada nos MMM com os resultados da compressão utilizando as VLMC e as cadeias de ordem completa. Uma discussão final é apresentada no Capítulo 8.

# Capítulo 2

# Princípio MDL e compressão de dados

O princípio MDL (*Minimum Description Length*) basea-se na compressão de dados. A compressão de dados pode ser alcançada através da atribuição de pequenas descrições para os símbolos mais frequentes e descrições maiores para os símbolos menos frequentes. Na Seção 2.1 introduzimos o conceito do princípio MDL. Na Seção 2.2 apresentamos os conceitos de entropia e entropia relativa e algumas de suas propriedades. Na Seção 2.3 introduzimos a definição de sistema de codificação e alguns tipos de códigos e apresentamos o algoritmo de Huffman.

## 2.1 Princípio MDL

A tarefa da inferência indutiva é encontrar leis de regularidade em um certo conjunto de dados. Essas regularidades são usadas para ganhar algum tipo de conhecimento sobre os dados.

MDL (*Minimum Description Length*) é um princípio geral para inferência indutiva, baseado na idéia que quanto mais podemos comprimir os dados, i.e. descrever os dados utilizando uma quantidade inferior de símbolos que o necessário para descrevê-los literalmente, mais regularidades existe nos dados e consequentemente mais informação sobre os

dados teremos.

O princípio estabelece que o melhor modelo para um dado conjunto de dados, é aquele que leva à maior compressão (considerando tanto a codificação dos dados como a do modelo) se for usado para codificar os dados, isto é, o modelo que encontra ou melhor descreve as regularidades presentes nos dados. Os métodos baseados no princípio MDL geralmente são resistentes a *overfitting* e, em alguns casos, estimam ao mesmo tempo a estrutura e os parâmetros do modelo.

As aplicações do princípio MDL foram desenvolvidos principalmente por J. Rissanen em vários artigos começando com (Rissanen, 1978). Sua raíz encontra-se na teoria da complexidade de Kolmogorov, desenvolvida nos anos 60 por (Solomonoff, 1964) e (Kolmogorov, 1965).

## 2.2 Alguns conceitos de teoria da informação

Dois conceitos fundamentais da teoria da informação são o conceito de entropia e o conceito de entropia relativa. Primeiro vamos introduzir o conceito de entropia, que é uma medida de incerteza de uma variável aleatória.

**Definição 2.1** (Entropia)**.** *Seja $X$ uma variável aleatória discreta com distribuição de probabilidade $P$, $\mathcal{X}$ o espaço amostral. A entropia $H(X)$ da variável aleatória $X$ é definida por*

$$H(X) = E_P[-\log(P(X))] = -\sum_{x \in \mathcal{X}} P(x) \log(P(x)), \tag{2.1}$$

*onde $0 \log 0$ é definido como $\lim_{x \downarrow 0} x \log x = 0$; se $-\sum_x P(x) \log P(x) = \infty$, então $H(X) = \infty$. Onde $E_P$ denota o valor esperado calculado usando como probabilidade $P$. $P(x)$ denota $\mathbb{P}(X = x)$.*

Se $\mathcal{X}$ é contínuo, então não existe uma única definição de entropia. Uma definição usada frequentemente é a da entropia diferencial, definida pela substituição da somatória pela integral em (2.1). Nesta dissertação, denotaremos por log o logaritmo na base 2.

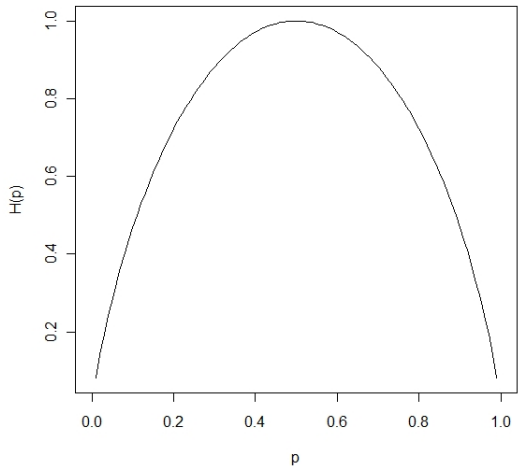


*Figura 2.1:* $H(p)$ versus $p$

**Exemplo 2.1.** *Seja*

$$X = \begin{cases} 1 & \text{com probabilidade } p, \\ 0 & \text{com probabilidade } 1-p. \end{cases}$$

*Onde* $0 \leq p \leq 1$, *então*

$$H(p) = -p \log p - (1-p) \log (1-p). \tag{2.2}$$

*A Figura 2.1 mostra o gráfico da função* $H(p)$. *Notamos que* $H(p) = 0$ *quando* $p = 0$ *ou* $1$, *o que faz sentido pois quando* $p = 0$ *ou* $1$ *a variável não é aleatória e não existe incertezas. Similarmente, temos incerteza máxima quando* $p = 0.5$, *que corresponde ao valor máximo da entropia.*

No contexto de codificação interpretamos a entropia da distribuição de probabilidade $P$ como o número esperado de bits necessários para codificar um elemento gerado por $P$ quando utilizamos um código cujo tamanho seja $-\log(P(x))$ para todo $x \in \mathcal{X}$.

Outro conceito muito importante é a da entropia relativa, que é uma medida de "distância" entre duas distribuições $Q$ e $P$ sobre $\mathcal{X}$.

**Definição 2.2** (Entropia Relativa)**.** *Seja* $X$ *uma variável aleatória discreta com espaço amostral* $\mathcal{X}$ *finito ou enumerável, e sejam* $P$ *e* $Q$ *distribuições de probabilidade sobre* $\mathcal{X}$.

*A entropia relativa ou distância de Kullback-Leibler (KL) entre duas distribuições $P$ e $Q$, é definida por*

$$\begin{aligned} D(P||Q) &= E_P[-\log Q(X)] - E_P[-\log P(X)] \\ &= \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)} \end{aligned} \tag{2.3}$$

*onde, para $a > 0$, $a\log 0 = \infty$ e $0\log 0 = 0$. Se os dois termos $E_P[-\log Q(X)]$ e $E_P[-\log P(X)]$ são infinitos, $D(P||Q)$ é indefinido. Se $\mathcal{X}$ é contínuo, então trocamos a somatório pela integral.*

Pela Definição 2.2, temos que $D(P||Q)$ é o valor adicional esperado quando utilizamos a distribuição $Q$ ao invés da distribuição $P$ para codificar um elemento gerado por $P$.

Mostraremos que a entropia relativa sempre é não negativa e é zero se e somente se $P = Q$. Entretanto, não é uma verdadeira distância entre duas distribuições, por não ser simétrica e não satisfazer a desigualdade triangular.

**Exemplo 2.2.** *Seja $\mathcal{X} = \{0, 1\}$ e considere duas distribuições $P$ e $Q$ em $\mathcal{X}$. Seja $P(0) = 1 - r$, $P(1) = r$, e seja $Q(0) = 1 - s$, $Q(1) = s$. Então,*

$$D(P||Q) = (1-r)\log\frac{1-r}{1-s} + r\log\frac{r}{s}$$

*e*

$$D(Q||P) = (1-s)\log\frac{1-s}{1-r} + s\log\frac{s}{r}$$

*Se $r = s$, então $D(P||Q) = D(Q||P) = 0$. Se $r = 1/2$, $s = 1/4$, então*

$$D(P||Q) = \frac{1}{2}\log\frac{\frac{1}{2}}{\frac{3}{4}} + \frac{1}{2}\log\frac{\frac{1}{2}}{\frac{1}{4}} = 0.2075 \text{ } bits,$$

*enquanto que*

$$D(Q||P) = \frac{3}{4}\log\frac{\frac{3}{4}}{\frac{1}{2}} + \frac{1}{4}\log\frac{\frac{1}{4}}{\frac{1}{2}} = 0.1887 \text{ } bits.$$

*Note que em geral $D(P||Q) \neq D(Q||P)$.*

**Teorema 2.1** (Desigualdade de Informação)**.** *Sejam $P$ e $Q$ duas distribuições de probabilidade sobre $\mathcal{X}$. Então*

$$D(P||Q) \geq 0, \tag{2.4}$$

*com igualdade se e somente se $P = Q$.*

*Demonstração.* Seja $A = \{x : P(x) > 0\}$ o conjunto suporte de $P(x)$. Então

$$\begin{aligned} -D(P||Q) &= -\sum_{x \in A} P(x) \log \frac{P(x)}{Q(x)} \\ &= \sum_{x \in A} P(x) \log \frac{Q(x)}{P(x)}. \end{aligned} \tag{2.5}$$

Pela desigualdade de Jensen, temos que

$$\begin{aligned} -D(P||Q) &\leq \log \left\{ \sum_{x \in A} P(x) \frac{Q(x)}{P(x)} \right\} \\ &= \log \left\{ \sum_{x \in A} Q(x) \right\} \\ &\leq \log \left\{ \sum_{x \in \mathcal{X}} Q(x) \right\} \\ &= \log\{1\} \\ &= 0. \end{aligned} \tag{2.6}$$

Como log é uma função estritamente côncava, temos a igualdade em (2.6) se e somente se $P(x) = Q(x)$. Então temos que $D(P||Q) = 0$ se e somente se $P(x) = Q(x)$ para todo $x$. □

Suponha que $X$ possui distribuição $P$. A desigualdade de informação diz que entre todos os possíveis códigos para $\mathcal{X}$, o código com tamanho $-\log P(X)$ nos fornece "na média" a menor codificação para as realizações de $P$,

$$E_P[-\log Q(X)] > E_P[-\log P(X)]. \tag{2.7}$$

O próximo resultado fornece um limitante superior para a entropia relativa.

**Proposição 2.1.** *Para duas distribuições de probabilidade $P$ e $Q$ sobre $\mathcal{X}$,*

$$D(P(\cdot)||Q(\cdot)) \leq \sum_{x\in\mathcal{X}} \frac{(P(x)-Q(x))^2}{Q(x)}. \tag{2.8}$$

*Demonstração.* Pela definição 2.2,

$$\begin{aligned} D(P||Q) &= \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)} \\ &\leq \sum_{x\in\mathcal{X}} P(x)\left(\frac{P(x)}{Q(x)}-1\right) \end{aligned}$$

temos a desigualdade pelo fato que $\log a \leq (a-1),\ \forall a > 0$. Completanto os quadradados, temos que

$$D(P||Q) \leq \sum_{x\in\mathcal{X}} \frac{(P(x)-Q(x))^2}{Q(x)}.$$

□

## 2.3 Compressão de dados

Seja $A$ um alfabeto, formalmente um alfabeto é algum conjunto finito ou contável cujos elementos são chamados de símbolos. Seja $x_1^n = (x_1, ..., x_n) \in \mathcal{X}^n$ uma amostra, onde $\mathcal{X}$ é o espaço amostral. Vamos assumir que $\mathcal{X}$ é finito ou contável; se $\mathcal{X}$ não é contável, na prática os $x_i$ serão sempre armazenados com uma precisão finita, portanto ainda podemos utilizar o mesmo procedimento.

Estamos interessado no tamanho de alguma descrição de $x_1^n$. Em específico, sempre codificaremos $x_1^n$ utilizando um alfabeto $B = \{0, 1\}$. Desta forma cada sequência $x_1^n$ é mapeada para alguma sequência binária no conjunto $B^* = \cup_{k\geq 1} B^k$.

**Definição 2.3.** *Um sistema de codificação $C$ para uma variável aleatória $X$ é uma função de $\mathcal{X}$, espaço amostral em $B^*$. Denotamos por $C(x)$ o código correspondente a $x$ e por $l_C(x)$ o tamanho de $C(x)$.*

**Exemplo 2.3.** *Seja $X$ uma variável aletatória com espaço amostral $\mathcal{X} = \{3, 4, 5\}$. $C(3) = 0, C(4) = 10$ e $C(5) = 11$ é um sistema de codificação para $\mathcal{X}$ sobre o alfabeto $B = \{0, 1\}$.*

**Definição 2.4.** *O tamanho esperado $L(C)$ de um sistema de codificação $C$ para uma variável aleatória $X$ com distribuição de probabilidade $P$ é dado por*

$$L(C) = \sum_{x \in \mathcal{X}} P(x) l_C(x), \tag{2.9}$$

*onde $l_C(x)$ é o tamanho do código associada à $x$.*

**Exemplo 2.4.** *Seja $X$ uma variável aleatória com a seguinte distribuição de probabilidade e codificação,*

$$\begin{aligned} &\mathbb{P}(X = 1) = 1/2, \quad C(1) = 0; \\ &\mathbb{P}(X = 2) = 1/4, \quad C(2) = 10; \\ &\mathbb{P}(X = 3) = 1/8, \quad C(3) = 110; \\ &\mathbb{P}(X = 4) = 1/8, \quad C(4) = 111. \end{aligned}$$

*O comprimento esperado*

$$L(C) = \frac{1}{2} l_C(1) + \frac{1}{4} l_C(2) + \frac{1}{8} l_C(3) + \frac{1}{8} l_C(4) = 1.75 \text{ bits.}$$

**Definição 2.5.** *Um sistema de codificação $C$ é dito ser não-singular se para todo $x_i, x_j \in \mathcal{X}$,*

$$x_i \neq x_j \Rightarrow C(x_i) \neq C(x_j). \tag{2.10}$$

Não-singulariedade é suficiente para evitar descrições ambíguas para um único valor de $X$. Mas ambiguidades podem ocorrer se desejamos codificar uma sequência de valores de $X$, como mostramos no exemplo 2.6.

**Definição 2.6.** *A extensão $C^*$ de um sistema de codificação $C$ é o mapeamento de uma sequência finita de valores em $\mathcal{X}$ para uma sequência finita de valores em $B^*$, definida por*

$$C^*(x_1 x_2 ... x_n) = C(x_1) C(x_2) ... C(x_n), \tag{2.11}$$

*onde $C(x_1)C(x_2)...C(x_n)$ representa concatenação dos códigos correspondentes.*

**Exemplo 2.5.** *Se $C(x_1) = 00$ e $C(x_2) = 11$, então $C^*(x_1 x_2) = 0011$.*

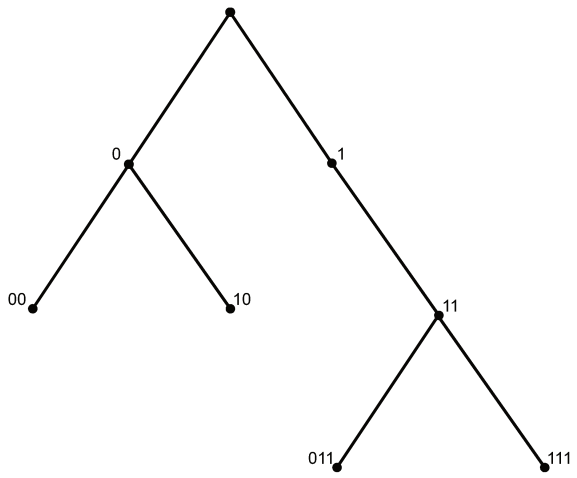


*Figura 2.2:* Exemplo de um código sufixo binário representado por uma árvore de sufixo, cada código é representado por um nó terminal.

**Definição 2.7.** *Um sistema de codificação é chamado de código unicamente decodificável se a extensão é não-singular.*

Ou seja, qualquer sequência de códigos tem apenas uma única sequência de símbolos possível que o produziu. No entanto, talvez seja necessário observar toda a sequência para determinar até mesmo o primeiro símbolo da sequência correspondente, como será mostrado no exemplo 2.6

**Definição 2.8.** *Um sistema de codificação é dito ser um código sufixo se nenhum código é sufixo de outro código.*

Um código sufixo pode ser decodificado sem nenhuma referência ao código futuro já que o final de cada código é imediatamente reconhecido. Podemos representar um código sufixo através de uma árvore de sufixo, onde cada nó terminal da árvore representa o código de um símbolo. Um exemplo de uma árvore de sufixo binária é ilustrada na Figura 2.2.

**Exemplo 2.6.** *A tabela seguinte ilustra as diferenças entre as classes de código. Para o código não-singular, a sequência* 010 *possui três possíveis sequências fontes:* 2, 14 *ou* 31*, portanto o código não é unicamente decodificável. O código unicamente decodificável não é um código sufixo, já que* $C(3) = 11$ *é sufixo de* $C(4) = 011$*.*

| X | Não-singular | Unicamente decodificável | Sufixo |
|---|---|---|---|
| 1 | 0 | 01 | 00 |
| 2 | 010 | 00 | 10 |
| 3 | 01 | 11 | 011 |
| 4 | 10 | 011 | 111 |

No próximo exemplo, apresentaremos um tipo de sistema de codificação que será utilizado nesta dissertação.

**Exemplo 2.7** (Código Uniforme)**.** *Seja $A$ um alfabeto finito, $|A| = m$, e seja $d \in \mathbb{N}$ tal que $2^{d-1} < |A| \leq 2^d$, onde $|S|$ denota o cardinal do conjunto $S$. O código uniforme mapea cada elemento de $A$ para uma sequência binária de tamanho $d$. Denotaremos por $C_U$ o código unifome e por $l_U$ o tamanho do código. Para todo $a \in A$ o tamanho do código $C_U$ é*

$$l_U(a) = \lceil \log m \rceil ,$$

*onde $\lceil \cdot \rceil$ é a função teto. Se $|A| = 4$, então o código uniforme codifica cada elemento de $A$ por um código de tamanho 2. Como todos os códigos possuem o mesmo tamanho , então nenhum código é sufixo de outro código. Portanto o código uniforme é um código de sufixo.*

Podemos interpretar uma codificação como uma mensagem que algum codificador **A** envia para algum decodificador **B**. Estamos interessados em minimizar o tamanho da mensagem a ser enviada, de tal maneira que **B** possa decodificar de forma única a mensagem enviada por **A**. Devido a estrutura do código sufixo não podemos atribuir pequenos códigos para todos os símbolos. Porém, o conjunto dos possíveis tamanhos dos códigos para um código sufixo é limitado pela seguinte desigualdade.

**Teorema 2.2** (Desigualdade de Kraft)**.** *Para qualquer código sufixo $C$ sobre um alfabeto binário, os tamanho dos códigos $l_1, l_2, \ldots, l_m$ devem satisfazer a desigualdade*

$$\sum_{i=1}^{m} 2^{-l_i} \leq 1. \tag{2.12}$$

*Inversamente, dado um conjunto de códigos cujos tamanhos satisfazem a desigualdade, então existe um código sufixo com o mesmo tamanho desses códigos.*

*Demonstração.* Seja $l_{max} = \max_{i:1\leq i\leq m}\{l_i\}$ o tamanho do maior código. Uma árvore de profundidade $l_{max}$ possui $2^{l_{max}}$ possíveis códigos. O número dos possíveis descendentes na profundidade $l_i$ é $2^{l_{max}-l_i}$. Então,

$$\sum_i 2^{l_{max}-l_x} \leq 2^{l_{max}} \tag{2.13}$$

$$\sum_i 2^{-l_i} \leq 1. \tag{2.14}$$

Inversamente, dado um conjunto de códigos de tamanhos $l_1, \ldots, l_m$ satisfazendo a equação (2.12), sempre podemos contruir uma árvore de sufixo, associando um nó terminal de profundidade $l_1$ como sendo o código do símbolo 1, depois associando outro nó terminal de profundidade $l_2$, que não seja descendente do nó terminal associado à 1, como sendo o código do símbolo 2, procedendo dessa maneira até o elemento $m$. Desta forma, construimos um código sufixo cujo conjunto constituído pelos tamanhos dos códigos é $\{l_1, \ldots, l_m\}$. □

Portanto, qualquer conjunto de códigos que satisfazem a condição do código sufixo tem que satisfazer a desigualdade de Kraft e a desigualdade de Kraft é uma condição suficiente para a existência de um código sufixo. Considere agora o problema de encontrar o código sufixo com o mínimo tamanho esperado. A partir do Teorema 2.2, isto é equivalente a encontrar o conjunto $\{l_1, l_2, \ldots, l_m\}$ que satisfaz a desigualdade de Kraft e cujo tamanho esperado seja inferior ao tamanho esperado de qualquer outro código sufixo. Este é um problema de otimização padrão.

Seja $X$ uma variável aleatória com distribuição de probabilidade $P$, $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ o espaço amostral. Denotaremos por $l_i = l(x_i)$ e $P(x_i) = p_i$. Queremos minimizar,

$$L = \sum_{i=1}^{m} p_i l_i \tag{2.15}$$

para todos os $l_1, l_2, ..., l_m$ satisfazendo

$$\sum_{i=1}^{m} 2^{-l_i} \leq 1. \tag{2.16}$$

Utilizando o método dos multiplicadores de Lagrange,

$$J = \sum_{i=1}^{m} p_i l_i + \lambda \left( \sum_{i=1}^{m} 2^{-l_i} \right). \tag{2.17}$$

Derivando com relação à $l_i$, para $i = 1, 2, \ldots, m$, obtemos

$$\frac{\partial J}{\partial l_i} = p_i - \lambda 2^{-l_i} \ln 2 \tag{2.18}$$

e igualando a zero, temos que

$$2^{-l_i} = \frac{p_i}{\lambda \ln 2}. \tag{2.19}$$

Substituindo na restrição (2.16), encontramos $\lambda = 1/\ln 2$ e $p_i = 2^{-l_i}$, finalmente obtemos o tamanho de código ótimo, denotado por $l_i^*$ para cada $i = 1, 2, \ldots, m$ e dado por

$$l_i^* = -\log(p_i). \tag{2.20}$$

Calculando o tamanho esperado $L^*$ encontramos a seguinte relação com a entropia

$$L^* = \sum p_i l_i^* = -\sum p_i \log(p_i) = H(X). \tag{2.21}$$

**Teorema 2.3.** *Seja $X$ uma variável aleatória com distribuição de probabilidade $P$. O tamanho esperado $L$ de qualquer código sufixo, sobre o alfabeto binário, da variável aleatória $X$ é maior ou iqual a entropia $H(X)$,*

$$L \geq H(X) \tag{2.22}$$

*com igualdade se e somente se $2^{-l_i} = p_i$, onde $p_i = P(x_i)$ para cada $x_i$ valor de $X$.*

*Demonstração.* A diferença entre o tamanho esperado $L$ e a entropia $H(X)$ é dada por

$$\begin{aligned} L - H(X) &= \sum_i p_i l_i - \sum_i p_i \log\left(\frac{1}{p_i}\right) \\ &= -\sum_i p_i \log 2^{-l_i} + \sum_i p_i \log(p_i) \\ &= \sum_i p_i \left[\log(p_i) - \log 2^{-l_i}\right] \\ &= \sum_i p_i \log\left(\frac{p_i}{2^{-l_i}}\right) \end{aligned}$$

Seja $r_i = 2^{-l_i} / \sum_j 2^{-l_j}$ e $c = \sum_j 2^{-l_i}$, obtemos

$$\begin{aligned} L - H(X) &= \sum p_i \log \left( \frac{p_i}{r_i} \right) - \log c \\ &= D(P||R) + \log \frac{1}{c} \\ &\geq 0 \end{aligned}$$

onde $R = (r_1, r_2, \ldots, r_m)$. A última desigualdade é dada pelo Teorema 2.1 e pelo fato que $c \leq 1$ (desigualdade de Kraft). □

Consideramos que um código é ótimo se o tamanho esperado do código é mínimo. Um código sufixo ótimo, para uma distribuição pode ser construído através do algoritmo de Huffman (Huffman, 1952), o qual é apresentado a seguir para um alfabeto binário.

**Algoritmo 2.1.** *(Algoritmo de Huffman)*
*Seja $X$ uma variável aleatória, $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$ espaço amostral, com distribuição de probabilidade $P$. Para $i \in \{1, 2, \ldots, m\}$, denotaremos por $p_i = P(x_i)$, $C_i = C(x_i)$ e $l_i = l(x_i)$. Seja*

$$\mathcal{P} = \{p_1, p_2, \ldots, p_m\}.$$

1. *Encontre $I^* = \arg\min\{i : p_i \in \mathcal{P}\}$ e $J^* = \arg\min\{j : p_j \in \mathcal{P}\}$.*
2. *Defina $H = I^* \cup J^*$,*
   - *2.1 se $|H| = 2$ e $l_h = 0 \; \forall h \in H$, então $C_{I^*} = 0$ e $C_{J^*} = 1$;*
   - *2.2 caso contrário,*
     - *2.2.1 para $h \in I^*$, $C_h = C_h 0$, a extensão do código $C_h$ com o símbolo 0;*
     - *2.2.2 e para $h \in J^*$, $C_h = C_h 1$, a extensão do código $C_h$ com o símbolo 1.*
3. *Defina $I' = I^* \cup J^*$, $\mathcal{P}' = \{p_k : k \in I'\}$ e $p_{I'} = \sum_{k \in I'} p_k$*
4. *$\mathcal{P} = \mathcal{P} \backslash \mathcal{P}' \cup \{p_{I'}\}$,*
   - *4.1 se $|\mathcal{P}| = 1$, ir para 5;*
   - *4.2 caso contrário, volte para 1.*
5. *Fim.*

**Exemplo 2.8.** *Considere uma variável aleatória $X$ tomando valores no conjunto $\mathcal{X} = \{1, 2, 3, 4, 5\}$ com probabilidades $0.25, 0.25, 0.20, 0.15, 0.15$, respectivamente. A próxima tabela ilustra o código gerado pelo algoritmo de Huffman.*

| Código | $X$ | Probabilidades | | | | |
|---|---|---|---|---|---|---|
| 10 | 1 | 0.25 | 0.25 | **0.25** | **0.55** | 1 |
| 11 | 2 | 0.25 | **0.25** | 0.45 | **0.45** | |
| 01 | 3 | 0.20 | **0.20** | **0.30** | | |
| 000 | 4 | **0.15** | 0.30 | | | |
| 100 | 5 | **0.15** | | | | |

Sem perda de generalidade, assumiremos que $p_1 \geq p_2 \geq ... \geq p_m$.

**Lema 2.1.** *Para qualquer distribuição, existe um código sufixo ótimo que satisfaz as seguintes propriedades:*

1. *Se $p_j > p_k$, então $l_j \leq l_k$.*

2. *Os dois maiores códigos possuem o mesmo tamanho.*

*Demonstração.* Seja $C$ um código sufixo ótimo e considere $C'$ outro código em que apenas a $j$-ésimo e $k$-ésimo códigos estão trocados de posição, denotemos por $l_i$ e $l_i^{'}$ os tamanhos dos códigos $C(x_i)$ e $C'(x_i)$ respectivamente, note que $l'_j = l_k$ e $l'_k = l_j$ e $l_i = l'_i$, $\forall i \neq k$ e $\forall i \neq j$. Então,

$$\begin{aligned} L(C') - L(C) &= \sum p_i l'_i - \sum p_i l_i \\ &= p_j l_k + p_k l_j - p_j l_j - p_k l_k \\ &= (p_j - p_k)(l_k - l_j). \end{aligned}$$

Entretanto, $p_j - p_k > 0$ e como $C$ é ótimo então $L(C') - L(C) \geq 0$. Portanto, devemos ter $l_k \geq l_j$. Se os dois maiores códigos não possuem o mesmo tamanho, então podemos deletar o último bit do maior, preservando a propriedade de sufixo e alcançando um valor esperado menor.

□

Mostramos que existe um código ótimo satisfazendo as propriedades do Lema 2.1. O código gerado pelo algoritmo de Huffman é um código ótimo, a demonstração da otimalidade do código gerado pelo algoritmo de Huffman pode ser encontrada em (Cover & Thomas, 1991).

# Capítulo 3

# Modelos de Markov mínimos

Considere uma cadeia de Markov de ordem $M$ sobre um alfabeto finito $A$ e as respectivas probabilidades de transição, a cadeia possui $(|A|-1)|A|^M$ parâmetros, onde $|A|$ representa a cardinalidade de $A$. É possível que existam várias sequências de tamanho $M$ com as mesmas probabilidades de transição, neste caso podemos reduzir o número de parâmetros para expressar o modelo. Um exemplo de que nem sempre se tem o modelo completo são as VLMC. A idéia no MMM é encontrar o modelo de Markov com o número mínimo de parâmetros. Para encontrar este modelo mínimo, (García & González-López, 2010) introduziram o conceito de mínima partição boa de tal maneira que todas as sequências de tamanho $M$ que possuem as mesmas probabilidades de transição são reunidas em uma única categoria. Desta forma, todas as sequências de uma categoria são associadas ao mesmo conjunto de probabilidades de transição, produzindo um modelo cujo conjunto de parâmetros é mínimo.

Seja $A$ um alfabeto finito, o espaço de estados das cadeias estacionárias consideradas. Para $m \leq n$ denotamos a sequência $a_m a_{m-1} ... a_n$ por $a_m^n$, onde $a_i \in A$, $m \leq i \leq n$, de tamanho $l(a_m^n) = n - m + 1$. Dadas duas sequências $s = s_m^n$ e $r = r_j^k$ denotamos por $rs$ a sequência obtida pela concatenação das duas sequências. Diremos que uma sequência $r$ é sufixo da sequência $s$ se existe $u$, com $l(u) \geq 1$, tal que $s = ur$. Seja $A^*$ o conjunto de todas as sequências finitas sobre $A$.

## 3.1 Cadeia de Markov de alcance variável

Seja $A$ um alfabeto finito. Podemos definir as VLMC como Cadeias de Markov as quais consideram apenas a parte relevante do passado para cada estado presente.

**Definição 3.1.** *Seja $(X_t)_{t\in\mathbb{Z}}$ um processo estacionário com valores em $A$. Diremos que o processo $(X_t)_{t\in\mathbb{Z}}$ é uma Cadeia de Markov de Alcance Variável se para toda seqüência $x_0, ..., x_n$ satisfazendo*

$$\mathbb{P}(X_0 = x_0, ..., X_{n-1} = x_{n-1}) > 0, \tag{3.1}$$

*existe um inteiro $k$, determinado a partir de $x_0, ..., x_{n-1}$ tal que*

$$\mathbb{P}(X_n = x_n | X_0^{n-1} = x_0^{n-1}) = \mathbb{P}(X_n = x_n | X_{n-k}^{n-1} = x_{n-k}^{n-1}). \tag{3.2}$$

Assume-se que $k(x_0^{n-1})$ é o mínimo inteiro que satisfaz (3.2). Assim, $k$ é o comprimento da memória do processo, dado que o passado foi $x_0, ..., x_{n-1}$. Ou seja, cada estado presente da cadeia depende de um passado até encontrar um determinado evento o qual faz com que todo o passado restante seja irrelevante.

As seqüências finitas $(x_{n-k}, ..., x_{n-1})$ são os passados relevantes. Como o processo é estacionário, o inteiro $k$ não depende de $n$. Portanto, denotaremos as sequências passadas por $(x_{-k}, ..., x_{-1})$.

**Definição 3.2.** *Um subconjunto finito $\tau$ de $A^*$ é uma árvore irredutível se satisfaz as seguintes propriedades.*

1. *Propriedade de sufixo. Nenhuma sequência $s \in \tau$ é um sufixo de uma outra sequência $r \in \tau$.*

2. *Irredutibilidade. Nenhuma sequência $s \in \tau$ pode ser substituída por um sufixo dela sem violar a propriedade do sufixo.*

Note que o conjunto $\tau$ pode ser identificado pelo conjunto de nós terminais de uma árvore com um número enumerável de ramos finitos.

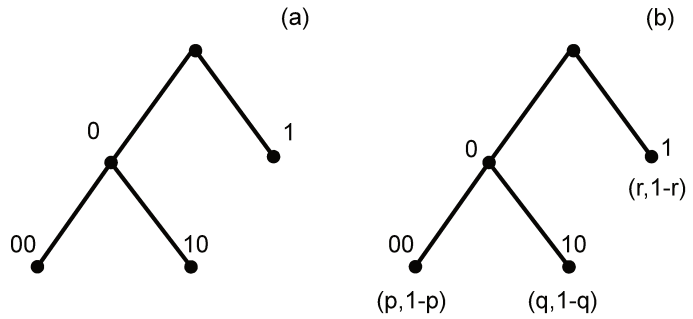


*Figura 3.1:* (a) Representação do conjunto de contexto $\{1, 00, 10\}$ por uma árvore. (b) Árvore de contexto com probabilidades de transição associado a cada nó terminal.

**Definição 3.3.** *A sequência $s \in A^*$ é um contexto para o processo $(X_t)$ se $P(s) > 0$ e para toda sequência $x_{-\infty}^{-1}$ tal que $s$ é sufixo da sequência $x_{-\infty}^{-1}$ temos que*

$$\mathbb{P}(X_0 = a | X_{-1}^{-\infty} = x_{-\infty}^{-1}) = P(a|s), \forall a \in A,$$

*e nenhum sufixo de $s$ tem essa propriedade.*

Com essa definição podemos ver que o conjunto de contextos de um proceso $(X_t)_{t\in\mathbb{Z}}$ é uma árvore irredutível. O conjunto de contexto associado ao processo $(X_t)_{t\in\mathbb{Z}}$ é chamada de *árvore probabilística de contexto.*

**Definição 3.4.** *Uma **árvore probabilística de contexto** é um par ordenado $(\tau, p)$ tal que*

1. *$\tau$ é uma árvore irredutível;*

2. *$p = \{P(\cdot|s) : s \in \tau\}$ é uma família de probabilidades de transição sobre $A$.*

Em uma VLMC o conjunto de sequências passadas relevantes é o conjunto de contextos. Essa característica permite representar o modelo como uma árvore probabilística de contexto, em que cada contexto é representado por um nó terminal. Chamaremos por árvore de contexto as árvores probabilísticas de contexto.

**Exemplo 3.1.** *Suponhamos que as probabilidades de transição da cadeia $(X_t)_{t\in\mathbb{Z}}$ satisfazem a seguintes propriedades*

$$\mathbb{P}(X_n = x_n | X_0^{n-1} = x_0^{n-1}) = \begin{cases} \mathbb{P}(X_n = x_n | X_{n-2}^{n-1} = x_{n-2}^{n-1}) & se\ X_{n-1} = 0, \\ \mathbb{P}(X_n = x_n | X_{n-1} = x_{n-1}) & se\ X_{n-1} = 1, \end{cases}$$

*e que as probabilidades de transição são dadas por*

$$\begin{aligned} \mathbb{P}(X_n = 0 | X_{n-1} = 0, X_{n-2} = 0) &= p, \\ \mathbb{P}(X_n = 0 | X_{n-1} = 0, X_{n-2} = 0) &= q, \qquad (3.3) \\ \mathbb{P}(X_n = 0 | X_{n-1} = 1) &= r, \qquad (3.4) \end{aligned}$$

*onde* $0 < p, q, r < 1$. *Quando olharmos para o passado para predizer o símbolo atual da sequência, se o símbolo imediato for* 1, *não precisamos observar mais nenhuma informação da sequência passada para determinar a distribuição de probabilidade de transição. Por outro lado, se o símbolo imediato for* 0, *é necessário observar um símbolo a mais no passado para predizer o próximo símbolo. Na Figura 3.1(a) ilustramos a árvore de contexto que representa o conjunto de contextos dado por* $\{00, 10, 1\}$, *e a Figura 3.1(b) representa a árvore de contexto do processo descrito acima.*

## 3.2 Modelos de Markov Mínimos

Seja $(X_t)_{t\in\mathbb{Z}}$ uma cadeia de Markov de ordem $M$ sobre um alfabeto finito $A$. Chamamos de $\mathcal{S} = A^M$ o espaço de estados. E denotamos por $P(a|s)$ a probabilidades de transição da cadeia, tal que

$$P(a|s) = \mathbb{P}(X_t = a | X_{t-M}^{t-1} = s), a \in A, s \in \mathcal{S}. \qquad (3.5)$$

**Definição 3.5.** *Uma partição* $\mathcal{L} = \{L_1, L_2, ..., L_K\}$ *de* $\mathcal{S}$ *é uma* ***partição boa*** *de* $\mathcal{S}$ *se para cada* $s, s' \in L$, $L \in \mathcal{L}$,

$$\mathbb{P}(X_t = .|X_{t-M}^{t-1} = s) = \mathbb{P}(X_t = .|X_{t-M}^{t-1} = s'). \qquad (3.6)$$

Os conjuntos $L_i \in \mathcal{L}$, $1 \leq i \leq K$, são categorias do espaço de estados $\mathcal{S}$. Pela defnição 3.5, os elementos dentro de cada categoria possuem as mesmas probabilidades de transição. Assim, todo elemento de $\mathcal{S}$ pertencentes à mesma categoria ativam o mesmo

mecanismo aleatório para escolher o próximo símbolo na cadeia de Markov. No entanto, é possível que existam $L_i, L_j \in \mathfrak{L}$, $i \neq j$, tal que para $s \in L_i$ e $r \in L_j$

$$P(a|s) = P(a|r) \ \ \forall a \in A.$$

**Definição 3.6.** *Uma partição $\mathfrak{L} = \{L_1, L_2, ..., L_K\}$ de $\mathcal{S}$ é a* ***mínima partição boa*** *de $\mathcal{S}$ se é uma partição boa de $\mathcal{S}$ e para todo $s, s' \in \mathcal{S}$ tal que*

$$I\!P(X_t = .|X_{t-M}^{t-1} = s) = I\!P(X_t = .|X_{t-M}^{t-1} = s'), \tag{3.7}$$

*existe $L \in \mathfrak{L}$ e $s, s' \in L$.*

Se $\mathfrak{L}$ é uma mínima partição boa, então não existem elementos de diferentes conjuntos que possuem as mesmas probabilidades de transição.

A árvore de contexto para uma VLMC com profundidade finita $M$, define uma partição boa no espaço de estados $\mathcal{S} = A^M$, considere como exemplo a VLMC definida no próximo exemplo.

**Exemplo 3.2.** *Considere uma VLMC sobre o alfabeto $A = \{0, 1\}$ com profundidade $M = 3$. Os contextos e probabilidades de transição são dados por*

$$\begin{aligned}
\{0\}, &\quad P(0|\{0\}) = 0.3, &\quad P(1|\{0\}) = 0.7;\\
\{01\}, &\quad P(0|\{01\}) = 0.4, &\quad P(1|\{01\}) = 0.6;\\
\{011\}, &\quad P(0|\{011\}) = 0.3, &\quad P(1|\{011\}) = 0.7;\\
\{111\}, &\quad P(0|\{111\}) = 0.8, &\quad P(1|\{111\}) = 0.2.
\end{aligned}$$

*Esta árvore de contexto corresponde a seguinte partição boa,*

$$\mathfrak{L} = \{\{\{000\}, \{100\}, \{010\}, \{110\}\}, \{\{001\}, \{101\}\}, \{011\}, \{111\}\},$$

*onde $L_1 = \{\{000\}, \{100\}, \{010\}, \{110\}\}$, $L_2 = \{\{001\}, \{101\}\}$, $L_3 = \{011\}$ e $L_4 = \{111\}$.*

*Porém, $\mathfrak{L}$ definida pelos contextos não é necessariamente uma mínima partição boa. Devido à propriedade de sufixo presente nas árvores de contextos, é possível que diferentes contextos possuam as mesmas probabilidades de transição como mostra o exemplo.*

A mínima partição boa de $\mathcal{S}$ define o modelo. Mais precisamente, o modelo de Markov Mínimo é definido por,

**Definição 3.7.** *Seja $(X_t)_{t\in\mathbb{Z}}$ uma cadeia de Markov de ordem $M$ sobre um alfabeto finito $A$ e seja $\mathfrak{L} = \{L_1, L_2, ..., L_K\}$ uma partição de $\mathcal{S} = A^M$. Diremos que $(X_t)$ é um modelo de Markov mínimo com partição $\mathfrak{L}$ se para cada $r, s \in \mathcal{S}$*

$$P(a|s) = P(a|r) \text{ para todo } a \in A,$$

*se, e somente se existe $i$, $1 \leq i \leq K$ tal que $r, s \in L_i$.*

A seguir definiremos a taxa de entropia relativa para dois processos estacionários.

**Definição 3.8.** *Sejam $(X_t)_{t\in\mathbb{Z}}$ e $(Y_t)_{t\in\mathbb{Z}}$ dois processos estacionários sobre um alfabeto finito $A$. A taxa de entropia relativa (entropia por unidade de tempo) entre dois processos estacionários é calculada pela expressão*

$$\lim_{n\to\infty} \frac{1}{n} D\left((X_t)_{t=1,2,...,n} || (Y_t)_{t=1,2,...,n}\right), \tag{3.8}$$

*quando o limite existe.*

O resultado a seguir é uma generalização do resultado para árvores de contextos proposto por (Lanfredi, 2010). O resultado proporciona uma fórmula para o cálculo da distância entre dois MMM.

**Teorema 3.1.** *Seja $A$ um alfabeto finito. Considere dois processos sobre $A$ com leis $P$ e $Q$. A taxa da entropia relativa do modelo com lei $P$ ao modelo com lei $Q$ é dada por*

$$D(P||Q) = \sum_{s\in\mathcal{S}} P(s)\left(D(P(.|s)||Q(.||s))\right), \tag{3.9}$$

*onde $\mathcal{S} = A^M$ e $M$ é o máximo entre as ordens do modelo com lei $P$ e do modelo com $Q$.*

*Demonstração.* Denotaremos por $P_n$ e $Q_n$ as probabilidades $P(x)$ e $Q(x)$ para algum $x \in A^n$, respectivamente. Dado uma sequência $z \in A^{n+1}$, $n > M$, exitem $x \in A^n$ e $a \in A$ tais que $z = xa$. Então,

$$\begin{aligned} D(P_{n+1}||Q_{n+1}) &= \sum_{z\in A^{n+1}} P(z)\log\frac{P(z)}{Q(z)} = \sum_{x\in A^n}\sum_{a\in A} P(xa)\log\frac{P(xa)}{Q(xa)} \\ &= \sum_{x\in A^n}\sum_{a\in A} P(xa)\log\frac{P(x)}{Q(x)} + \sum_{x\in A^n}\sum_{a\in A} P(x)P(a|x)\log\frac{P(a|x)}{Q(a|x)} \end{aligned}$$

Usando o fato que $\sum_{a\in A} P(xa) = P(x)$ e utilizando as definições

$$D(P_n||Q_n) = \sum_{z\in A^n} P(z)\log\frac{P(z)}{Q(z)}$$

e

$$D(P(.|X)||Q(.|X)) = \sum_{a\in A} P(a|X)\log\left(\frac{P(a|X)}{Q(a|X)}\right), \tag{3.10}$$

temos que

$$D(P_{n+1}||Q_{n+1}) = D(P_n||Q_n) + \sum_{x\in A^n} P(x)D(P(.|x)||Q(.|x)). \tag{3.11}$$

Sejam $y \in A^{n-M}$ e $s \in \mathcal{S}$ tais que $x = ys$, da equação (3.11) temos que

$$D(P_{n+1}||Q_{n+1}) = D(P_n||Q_n) + \sum_{x\in A^n} P(ys)D(P(.|ys)||Q(.|ys)). \tag{3.12}$$

pela definição (3.10)

$$\begin{aligned} D(P(.|ys)||Q(.|ys)) &= \sum_{a\in A} P(a|ys)\log\frac{P(a|ys)}{Q(a|ys)} \\ &= \sum_{a\in A} P(a|s)\log\frac{P(a|s)}{Q(a|s)} \\ &= D(P(.|s)||Q(.|s)), \end{aligned} \tag{3.13}$$

note que temos a segunda igualdade pelo fato que $s \in \mathcal{S} = A^M$.

Como $\sum_{y\in A^{n-M}} P(ys) = P(s)$, das equações (3.12) e (3.13) obtemos

$$D(P_{n+1}||Q_{n+1}) = \sum_{s\in\mathcal{S}} P(s)D(P(.|s)||Q(.|s)) + D(P_n||Q_n). \tag{3.14}$$

Utilizando o mesmo procedimento descrito acima, $D(P_n||Q_n)$ pode ser escrito como

$$D(P_n||Q_n) = \sum_{s\in\mathcal{S}} P(s)D(P(.|s)||Q(.|s)) + D(P_{n-1}||Q_{n-1}). \tag{3.15}$$

Substituindo (3.15) em (3.14), temos que

$$D(P_{n+1}||Q_{n+1}) = 2\sum_{s\in\mathcal{S}} P(s)D(P(.|s)||Q(.|s)) + D(P_{n-1}||Q_{n-1}). \tag{3.16}$$

Desenvolvendo, recursivamente, a expressão $D(P_{n-1}||Q_{n-1})$ obtemos

$$D(P_{n+1}||Q_{n+1}) = (n-M)\sum_{s\in\mathcal{S}} P(s)D(P(.|s)||Q(.|s)) + D(P_M||Q_M). \tag{3.17}$$

Pela definição da taxa de entropia relativa entre dois processos, quando o limite existe, temos que

$$\begin{aligned} D(P||Q) &= \lim_{n\to\infty}\left(\frac{n-M}{n}\sum_{s\in\mathcal{S}} P(s)D(P(.|s)||Q(.|s)) + \frac{1}{n}D(P_M||Q_M)\right) \\ &= \lim_{n\to\infty}\left(\frac{n-M}{n}\sum_{s\in\mathcal{S}} P(s)D(P(.|s)||Q(.|s))\right) + \lim_{n\to\infty}\frac{1}{n}D(P_M||Q_M) \\ &= \sum_{s\in\mathcal{S}} P(s)D(P(.|s)||Q(.|s)). \end{aligned}$$

□

# Capítulo 4

# Um algoritmo de seleção de modelo de Markov mínimo

Neste capítulo apresentamos uma metodologia para seleção do MMM utilizando o Critério da Informação Bayesiana (BIC), ou critério de Schwarz. García & González-López (2010) mostram que o critério BIC pode ser usado para determinar consistentemente se dois conjuntos de probabilidades de transição devem ser consideradas iguais ou não. Usando este resultado, descrevemos um algoritmo de seleção de MMM.

## 4.1 Seleção de modelos de Markov mínimo usando BIC

Nesta seção apresentamos um dos resultados principais de (García & González-López, 2010) que será utilizado posteriormente. Assumindo a notação da Seção 3.2. Seja $\mathfrak{L} = \{L_1, L_2, \cdots, L_K\}$ uma partição de $\mathcal{S}$, para $a \in A$, $L \in \mathfrak{L}$, defina

$$P(L,a) = \sum_{s \in L} I\!P(X_{t-M}^{t-1} = s, X_t = a) \;\; a \in A, \;\; L \in \mathfrak{L};$$

$$P(L) = \sum_{s \in L} I\!P(X_{t-M}^{t-1} = s), \;\; L \in \mathfrak{L}.$$

Seja $x_1^n$ uma amostra do processo $(X_t)_{t\in\mathbb{Z}}$. Para $L \in \mathcal{L}$, $a \in A$ e $n > M$, denotamos por $N_n(L, a)$ o número de ocorrência dos elementos pertencentes à $L$ seguido por $a$ na amostra $x_1^n$,

$$N_n(L,a) = \sum_{s\in L}\sum_{t=M+1}^{n} I\{x_{t-M}^{t-1} = s, x_t = a\}, \tag{4.1}$$

onde $I\{A\}$ é a função indicadora de $A$. O número de ocorrências dos elementos de $L$ na amostra $x_1^n$ é denotado por $N_n(L)$,

$$N_n(L) = \sum_{s\in L}\sum_{t=M+1}^{n} I\{x_{t-M}^{t-1} = s\}. \tag{4.2}$$

Denotaremos por $\mathcal{L}^{ij}$ a partição

$$\mathcal{L}^{ij} = \{L_1, \ldots, L_{i-1}, L_{ij}, L_{i+1}, \ldots, L_{j-1}, L_{j+1}, \ldots, L_K\}, \tag{4.3}$$

onde $L_{ij} = L_i \cup L_j$, para $1 \leq i < j \leq K$.

Adaptando a notação definida para a partição $\mathcal{L}$ para a nova partição $\mathcal{L}^{ij}$. Para $a \in A$ escrevemos

$$P(L_{ij}, a) = P(L_i, a) + P(L_j, a); \tag{4.4}$$

$$P(L_{ij}) = P(L_i) + P(L_j). \tag{4.5}$$

O número de ocorrências dos elementos pertencentes à $L_{ij}$ seguidos por $a \in A$ na amostra $x_1^n$ é dado por

$$N_n(L_{ij}, a) = N_n(L_i, a) + N_n(L_j, a), \tag{4.6}$$

e o número de ocorrências dos elementos de $L_{ij}$ na amostra é dado por

$$N_n(L_{ij}) = N_n(L_i) + N_n(L_j). \tag{4.7}$$

Denotaremos a probabilidade conjunta de $X_1^n = (X_1, X_2, ..., X_n)$ igual a $x_1^n = (x_1, x_2, ..., x_n)$ por

$$P(x_1^n) = I\!P(X_1^n = x_1^n),$$

Da notação dada pelas equações (3.5) e (4.1),

$$P(x_1^n) = P(x_1^M) \prod_{L\in\mathcal{L},a\in A} P(a|L)^{N_n(L,a)}. \tag{4.8}$$

Da mesma forma como em (Csiszár & Talata, 2006), García & González-López (2010) definem o critério BIC usando uma máxima verossimilhança modificada. Chamando de máxima verossimilhança a maximização do segundo termo da equação (4.8) para uma dada observação. Para amostra $x_1^n$ a máxima verossimilhança é dada por

$$\mathrm{ML}(\mathcal{L}, x_1^n) = \prod_{L\in\mathcal{L},a\in A} \left(\frac{N_n(L,a)}{N_n(L)}\right)^{N_n(L,a)}. \tag{4.9}$$

**Definição 4.1.** *Dada uma amostra $x_1^n$, o critério BIC para o modelo (4.9) é definido por*

$$BIC(\mathcal{L}, x_1^n) = \ln\left(ML(\mathcal{L}, x_1^n)\right) - \frac{(|A|-1)|\mathcal{L}|}{2}\ln(n). \tag{4.10}$$

Para $L_i, L_j \in \mathcal{L}$, suponha que $P(a|L_i) = P(a|L_j)$ para todo $a \in A$. Neste caso define-se

$$P(a|L_{ij}) = P(a|L_i) = P(a|L_j). \tag{4.11}$$

Pela equação (4.8) e suposição (4.11), temos que

$$P(x_1^n) = P(x_1^M) \prod_{L\in\mathcal{L}^{ij},a\in A} P(a|L)^{N_n(L,a)}, \tag{4.12}$$

a máxima verossilhança para o segundo termo é

$$\mathrm{ML}(\mathcal{L}^{ij}, x_1^n) = \prod_{L\in\mathcal{L}^{ij},a\in A} \left(\frac{N_n(L,a)}{N_n(L)}\right)^{N_n(L,a)}, \tag{4.13}$$

e o valor BIC para este modelo é igual a

$$\mathrm{BIC}(\mathcal{L}^{ij}, x_1^n) = \ln(\mathrm{ML}(\mathcal{L}^{ij}, x_1^n)) - \frac{(|A|-1)(|\mathcal{L}|-1)}{2}\ln(n). \tag{4.14}$$

O seguinte teorema mostra que para $n$ suficientemente grande a partição $\mathcal{L}^{ij}$ é preferível a $\mathcal{L}$ se, e somente se $P(a|L_i) = P(a|L_j)$, para todo $a \in A$.

**Teorema 4.1.** *Seja $(X_t)_{t\in\mathbb{Z}}$ uma cadeia de Markov de ordem $M$ sobre um alfabeto finito $A$, $\mathbb{S} = A^M$ o espaço de estados. Se $\mathfrak{L}$ é uma partição boa de $\mathbb{S}$ e $L_i \neq L_j$, $L_i, L_j \in \mathfrak{L}$. Então, eventualmente quase certamente quando $n \to \infty$,*

$$I\{BIC(\mathfrak{L}^{ij}, x_1^n) > BIC(\mathfrak{L}, x_1^n)\} = 1$$

*se, e somente se*

$$P(a|L_i) = P(a|L_j)\ \forall a \in A.$$

*Onde $I\{A\}$ é a função indicadora de $A$, e a partição $\mathfrak{L}^{ij}$ é definida sob $\mathfrak{L}$ pela equação (4.3).*

*Demonstração.* Como consequência da Definição 4.1, sua adaptação equação (4.14), e das equações (4.9) e (4.13),

$$\begin{aligned}\mathrm{BIC}(\mathfrak{L}, x_1^n) - \mathrm{BIC}(\mathfrak{L}^{ij}, x_1^n) = \sum_{a\in A}\Bigg\{ N_n(L_i,a)\ln\left(\frac{N_n(L_i,a)}{N_n(L_i)}\right) \\ + N_n(L_j,a)\ln\left(\frac{N_n(L_j,a)}{N_n(L_j)}\right) - N_n(L_{ij},a)\ln\left(\frac{N_n(L_{ij},a)}{N_n(L_{ij})}\right)\Bigg\} - \frac{(|A|-1)}{2}\ln(n). \quad (4.15)\end{aligned}$$

Suponha que $I\{BIC(\mathfrak{L}^{ij}, x_1^n) > BIC(\mathfrak{L}, x_1^n)\} = 1$, então

$$\begin{aligned}\sum_{a\in A}\Bigg\{ N_n(L_i,a)\ln\left(\frac{N_n(L_i,a)}{N_n(L_i)}\right) + N_n(L_j,a)\ln\left(\frac{N_n(L_j,a)}{N_n(L_j)}\right) \\ - N_n(L_{ij},a)\ln\left(\frac{N_n(L_{ij},a)}{N_n(L_{ij})}\right)\Bigg\} < \frac{(|A|-1)\ln(n)}{2n}. \quad (4.16)\end{aligned}$$

Como $N_n(L,a)$ e $N_n(L)$ são não negativas, usando a desigualdade de Jensen e pelas equações (4.6) e (4.7), temos que

$$N_n(L_i,a)\ln\left(\frac{N_n(L_i,a)}{N_n(L_i)}\right) + N_n(L_j,a)\ln\left(\frac{N_n(L_j,a)}{N_n(L_j)}\right) \quad to \quad (4.18)$$

$$-N_n(L_{ij},a)\ln\left(\frac{N_n(L_{ij},a)}{N_n(L_{ij})}\right) \geq 0. \quad (4.19)$$

Da equação (4.19),

$$\sum_{a\in A}\left\{N_n(L_i,a)\ln\left(\frac{N_n(L_i,a)}{N_n(L_i)}\right)+N_n(L_j,a)\ln\left(\frac{N_n(L_j,a)}{N_n(L_j)}\right)\right.$$
$$\left.-N_n(L_{ij},a)\ln\left(\frac{N_n(L_{ij},a)}{N_n(L_{ij})}\right)\right\}\geq 0, \tag{4.20}$$

com igualdade se e somente se $\frac{N_n(L_i,a)}{N_n(L_i)}=\frac{N_n(L_j,a)}{N_n(L_j)}\ \forall a\in A$.

Pela proposição A.1 e tomando o limite em $n$ das equações (4.16) e (4.20), considerando que o $\lim_{n\to\infty}\frac{(|A|-1)\ln(n)}{2n}=0$, obtemos a seguinte igualdade

$$\sum_{a\in A}\left\{P(L_i,a)\ln\left(\frac{P(L_i,a)}{P(L_i)}\right)+P(L_j,a)\ln\left(\frac{P(L_j,a)}{P(L_j)}\right)-P(L_{ij},a)\ln\left(\frac{P(L_{ij}a)}{P(L_{ij})}\right)\right\}=0,$$

usando a desigualdade de Jensen novamente e as notações (4.4) e (4.5), a igualdade acima é satisfeita se e somente se $\frac{P(L_i,a)}{P(L_i)}=\frac{P(L_j,a)}{P(L_j)}\ \forall a\in A$, ou equivalentemente $P(a|L_i)=P(a|L_j)\ \forall a\in A$.

Para a outra parte da demonstração. Suponha que $P(a|L_i)=P(a|L_j)\ \forall a\in A$,

$$\mathrm{BIC}(\mathfrak{L},x_1^n)-\mathrm{BIC}(\mathfrak{L}^{ij},x_1^n)=\ln\left(\prod_{a\in A}\left(\frac{N_n(L_i,a)}{N_n(L_i)}\right)^{N_n(L_i,a)}\right)$$
$$+\ln\left(\prod_{a\in A}\left(\frac{N_n(L_j,a)}{N_n(L_j)}\right)^{N_n(L_j,a)}\right)-\ln\left(\prod_{a\in A}\left(\frac{N_n(L_{ij},a)}{N_n(L_{ij})}\right)^{N_n(L_{ij},a)}\right)-\frac{(|A|-1)}{2}\ln(n).$$

Considerando que $\frac{N_n(L_{ij},a)}{N_n(L_{ij})}$ é o estimador de máxima verossimilhança de $P(a|L_{ij})$,

$$\mathrm{BIC}(\mathfrak{L},x_1^n)-\mathrm{BIC}(\mathfrak{L}^{ij},x_1^n)\leq\ln\left(\prod_{a\in A}\left(\frac{N_n(L_i,a)}{N_n(L_i)}\right)^{N_n(L_i,a)}\right)$$
$$+\ln\left(\prod_{a\in A}\left(\frac{N_n(L_j,a)}{N_n(L_j)}\right)^{N_n(L_j,a)}\right)-\ln\left(\prod_{a\in A}P(a|L_{ij})^{N_n(L_{ij},a))}\right)-\frac{(|A|-1)}{2}\ln(n).$$
$$=N_n(L_i)D\left(\frac{N_n(L_i,.)}{N_n(L_i)}\middle|\middle|P(.|L_i)\right)+N_n(L_j)D\left(\frac{N_n(L_j,.)}{N_n(L_j)}\middle|\middle|P(.|L_j)\right)-\frac{(|A|-1)}{2}\ln(n).$$

Onde $D(p||q)$ é a entropia relativa, dada pela definição 2.2. A iguadade é consequência da equação (4.11). Usando a proposição 2.1 e a proposição A.2, para qualquer $\delta>0$ e $n$

grande o suficiente. Para $L = L_i$ ou $L = L_j$ temos que,

$$D\left(\frac{N_n(L,.)}{N_n(L)}\middle\|P(.|L)\right) \leq \sum_{a\in A}\frac{\left(\frac{N_n(L,a)}{N_n(L)} - P(a|L)\right)^2}{P(a|L)} \tag{4.21}$$

$$\leq \sum_{a\in A}\frac{\frac{\delta\ln(n)}{N_n(L)}}{P(a|L)}. \tag{4.22}$$

Portanto, para $\delta > 0$ e $n$ grande o suficiente,

$$\begin{aligned}\mathrm{BIC}(\mathfrak{L}, x_1^n) - \mathrm{BIC}(\mathfrak{L}^{ij}, x_1^n) &\leq \frac{2\delta|A|}{p}\ln(n) - \frac{(|A|-1)}{2}\ln(n)\\ &= \ln(n)\left(\frac{2\delta|A|}{p} - \frac{(|A|-1)}{2}\right),\end{aligned}$$

onde $p = \min\{P(a|L) : a \in A, L \in L_{ij}\}$.

Em particular, se $\delta < \frac{p(|A|-1)}{4|A|}$, para $n$ grande o suficiente,

$$\mathrm{BIC}(\mathfrak{L}, x_1^n) - \mathrm{BIC}(\mathfrak{L}^{ij}, x_1^n) < 0.$$

□

O Teorema 4.1 ainda é válido se mudarmos o valor da constante, na penalização do critério BIC, para qualquer número positivo (e finito). No caso das VLMC, o problema de encontrar uma constante melhor foi abordada em diversos trabalhos como, por exemplo (Galves et al., 2009) e (Buhlmann & Wyner, 1999) .

## 4.2 Algoritmo MMM

Nesta seção apresentaremos o algoritmo MMM (Modelo Markov Mínimo) proposto por (García & González-Lopéz, 2010). Baseado no Teorema 4.1 de (García & González-Lopéz, 2010). O algoritmo estima consistentemente a mínima partição boa definida em 3.5. O próximo corolário do Teorema 4.1 proporciona a relação que utilizaremos para a construção do algoritmo MMM.

**Corolário 4.1.** *Seja $(X_t)_{t\in\mathbb{Z}}$ uma cadeia de Markov de ordem $M$ sobre um alfabeto finito $A$, $\mathbb{S} = A^M$ o espaço de estados. Se $\mathcal{L}$ é uma partição boa de $\mathbb{S}$ e $L_i \neq L_j$, $L_i, L_j \in \mathcal{L}$. Então, para $n$ suficientemente grande*

$$BIC(\mathcal{L}, x_1^n) - BIC(\mathcal{L}^{ij}, x_1^n) \ < 0 \quad \Longleftrightarrow \ d_{\mathcal{L}}(i,j) < \frac{(|A|-1)}{2},$$

*onde $d_{\mathcal{L}}(i,j)$ é definido por*

$$d_{\mathcal{L}}(i,j) = \frac{1}{\ln(n)} \sum_{a\in A} \left\{ N_n(L_i,a) \ln\left(\frac{N_n(L_i,a)}{N_n(L_i)}\right) + N_n(L_j,a) \ln\left(\frac{N_n(L_j,a)}{N_n(L_j)}\right) \right.$$
$$\left. - N_n(L_{ij},a) \ln\left(\frac{N_n(L_{ij},a)}{N_n(L_{ij})}\right) \right\}. \quad (4.23)$$

*Demonstração.* Da equação (4.15) temos a validade do resultado. □

Usando o corolário 4.1, implementamos o algoritmo que une dois conjuntos $L_i, L_j \in \mathcal{L}$ se, e somente se $P(a|L_i) = P(a|L_j)\ \forall a \in A$.

**Algoritmo 4.1.** *(Algoritmo MMM)*
*Considere $x_1^n$ uma amostra de um processo de Markov sobre $A$. Seja $\mathcal{L}^0 = A^M$ ou $\mathcal{L}^0 = \{L_1, L_2, \ldots, L_K\}$ qualquer partição boa de $\mathbb{S}$, e seja*

$$C = \{(i,j) : 1 \leq i,j \leq |\mathcal{L}^0| \text{ e } N_n(L_i), N_n(L_j) \geq \alpha\}.$$

1. *Defina $\mathcal{L} = \mathcal{L}^0$.*
2. *Calcule $(i^*, j^*) = \arg\min_{(i,j)\in C: i\neq j}\{d_{\mathcal{L}}(i,j)\}$.*
   - *2.1 Se $d_{\mathcal{L}}(i^*,j^*) > \frac{|A|-1}{2}$, então $\mathcal{L}' = \mathcal{L}$ e ir para 3,*
   - *2.2 caso contrário $\mathcal{L}' = \mathcal{L}\backslash\{\{L_{i^*}\},\{L_{j^*}\}\} \cup L_{i^*j^*}$;*
     - *2.2.1 se $|\mathcal{L}'| = 1$ ir para 3,*
     - *2.2.2 caso contrário, $\mathcal{L}^0 = \mathcal{L}'$ e voltar para 1.*
3. *$\hat{\mathcal{L}} = \mathcal{L}'$.*

Os parâmetros que devem ser escolhidos pelo usuário para a implementação do algoritmo são: $M$, a profundidade máxima das sequências e $\alpha$, o número mínimo de vezes que uma sequência deve aparecer na amostra.

Para $n$ suficientemente grande, o algoritmo retorna a mínima partição boa. Note que podemos iniciar o algoritmo utilizando qualquer partição boa, como por exemplo a partição boa definida por uma árvore de contexto.

**Corolário 4.2.** *Seja $\{X_t\}_{t\in\mathbb{Z}}$ uma cadeia de Markov de ordem $M$ sobre o alfabeto finito $A$, $\mathcal{S} = A^M$ e $x_1^n$ uma amostra do processo. $\hat{\mathcal{L}}$ dado pelo algoritmo 4.1 converge quase certamente eventualmente para $\mathcal{L}^*$, onde $\mathcal{L}^*$ é a mínima partição boa de $\mathcal{S}$.*

*Demonstração.* Pelo teorema 4.1, como $|\mathcal{L}| < \infty$, para $n$ suficientemente grande, o algoritmo retorna a mínima partição boa. □

## 4.3 Simulação

Considere uma cadeia de Markov de ordem $M = 3$ sobre o alfabeto $A = \{0, 1, 2\}$, $\mathcal{S} = A^M$ o espaço de estados. A partição $\mathcal{L} = \{L_1, L_2, \ldots, L_5\}$ de $\mathcal{S}$, onde

$$\begin{aligned} L_1 &= \{000, 100, 200, 010, 110, 210, 020, 120, 220, 022, 122, 222\}, \\ L_2 &= \{001, 101, 201, 011, 111, 211, 021, 121, 221\}, \\ L_3 &= \{012, 112, 212, 002\}, \\ L_4 &= \{102\}, \\ L_5 &= \{202\}, \end{aligned} \tag{4.24}$$

e probabilidades de transição,

$$\begin{aligned} P(0|L_1) = 0.2, \quad P(1|L_1) = 0.3, \\ P(0|L_2) = 0.4, \quad P(1|L_2) = 0.3, \\ P(0|L_3) = 0.4, \quad P(1|L_3) = 0.1, \\ P(0|L_4) = 0.1, \quad P(1|L_4) = 0.4, \\ P(0|L_5) = 0.3, \quad P(1|L_5) = 0.5. \end{aligned} \tag{4.25}$$

definem o MMM para o processo.

Implementamos um estudo de simulação para o modelo descrito acima. Simulamos 1000 amostras do processo para cada tamanho de amostra 6000, 8000, 10000 e 12000. Para cada amostra utilizamos o Algoritmo 4.1 para selecionar a mínima partição boa, começando pela partição boa $\mathcal{S}$.

Calculamos a proporção de erro para os diferentes tamanho de amostras, consideramos um erro de estimação a classificação de qualquer sequência em uma partição incorreta. Os resultados estão descritos na Tabela 4.1.

| Tamanho da amostra | Proporção de erro |
|---|---|
| 6000 | 0.465 |
| 8000 | 0.272 |
| 10000 | 0.190 |
| 12000 | 0.104 |

*Tabela 4.1:* Proporção de erro na estimação das partições do modelo

Para as amostras de tamanho superior a 10000 temos uma redução na proporção de erro. Devemos ressaltar que qualquer classificação incorreta é considerada um erro, portanto, uma única sequência classificada erroneamente é considera uma erro de estimação. A Tabela 4.2 mostra a média de sequências classificadas incorretamente por amostra.

| Tamanho da amostra | Média de erro |
|---|---|
| 6000 | 0.092 |
| 8000 | 0.023 |
| 10000 | 0.013 |
| 12000 | 0.008 |

*Tabela 4.2:* Média de sequências classificadas erroneamente por amostra.

# Capítulo 5

# Compressão de dados baseada nos modelos de Markov mínimos

O princípio MDL estabelece que o melhor modelo para um dado conjunto de dados, é aquele que leva à uma maior compressão quando usado para codificar os dados, considerando tanto a codificação dos dados como a codficação do modelo. Nesta dissertação concentramos no problema de minimização do número de parâmetros do modelo, reduzindo o tamanho da codificação para o modelo.

Como discutido anteriormente, podemos interpretar uma codificação como uma mensagem que algum codificador **A** envia para algum decodificador **B**. Para isso, devemos enviar a informação da codificação utilizada para que **B** possa decodificar a informação enviada por **A**. Ou seja, devemos enviar as informações sobre os parâmetros e sobre a estrutura do modelo utilizado na codificação dos dados.

Para identificar um MMM precisamos das informações das probabilidades de transição e da mínima partição boa. No caso das VLMC, precisamos das informações das probabilidades de transição e do conjunto de contextos. Para as cadeias de Markov de ordem finita, precisamos das informações das probabilidades de transição e da ordem $M$ do modelo. Portanto, no caso dos MMM devemos enviar as informações da mínima partição boa e das probabilidades de transição estimadas para o conjunto de dados. Para o caso das cadeias de Markov de ordem $M$ enviamos as informações da ordem da cadeia e das probabilidades de transição, e para as VLMC enviamos as informações dos contextos e

| | Parâmetros | Modelo |
|---|---|---|
| Cadeias de Markov de ordem $M$ | $\|A\|^M(\|A\|-1)\times 32$ | 0 |
| Cadeias de Markov de alcance variável | $\|\tau\|(\|A\|-1)\times 32$ | $\|\tau\|(\lceil\log\|\tau\|\rceil)$ |
| Modelos de Markov mínimos | $\|\mathcal{L}\|(\|A\|-1)\times 32$ | $\|\tau\|(\lceil\log\|\tau\|\rceil)+\|\tau\|(\lceil\log\|\mathcal{L}\|\rceil)$ |

*Tabela 5.1:* Informações em bits que devem ser enviadas para cada modelo.

das probabilidades de transição.

Utilizamos ponto flutuante para a representação binária dos parâmetros, especificamente utilizamos ponto flutuante de precisão simples, 32 bits, para representar cada probabilidade de transição.

Para as cadeias de Markov de ordem $M$, a informação sobre o modelo que devemos enviar é apenas a ordem da cadeia, assumimos que o decodificador **B** já possui a informação de $M$ e do alfabeto $A$, então nenhuma informação sobre o modelo precisa ser enviada. No entanto, o número de parâmetros é superior quando comparado ao número de parâmetros das VLMC e dos MMM. Como podemos representar uma VLMC por uma árvore de contexto, a informação sobre o modelo a ser enviada é a própria árvore de contexto. Utilizando um código uniforme para codificar os contextos da árvore, obtemos um código de tamanho $\lceil\log|\tau|\rceil$ para cada contexto. O tamanho da informação da árvore de contexto é $|\tau|(\lceil\log|\tau|\rceil)$ bits.

A informação da partição do MMM pode ser representada pela árvore de contexto correspondente à partição, acrescentando a identificação dos contextos que pertencem ao mesmo conjunto da partição. Usamos um código uniforme para codificar os índices dos conjuntos de $\mathcal{L}$, assim identificamos o conjunto ao qual o contexto pertence a partir de um código de tamanho $\lceil\log|\mathcal{L}|\rceil$. Obtemos uma codificação de tamanho $|\tau|(\lceil\log|\tau|\rceil)+|\tau|(\lceil\log|\mathcal{L}|\rceil)$ em bits para a informação da partição. A Tabela 5.1 ilustra o tamanho da informação dos parâmetros e dos modelos, em bits, que devemos enviar para cada tipo de modelo.

**Exemplo 5.1.** *Considere o processo descrito no Capítulo 4, equações (4.24) e (4.25), para esse processo temos que* $|\mathcal{L}|=5$. *A cadeia de Markov de alcance variável associada à partição* $\mathcal{L}$ *é definida pelo seguinte conjunto de contexto,*

$$\tau=\{0,1,12,22,002,102,202\},$$

*e pela família de probabilidades de transição* $p = \{P(.|s) : s \in \tau\}$, $|\tau| = 7$. *Na Tabela 5.2 apresentamos o tamanho da informação, em bits dos parâmetros e dos modelos, para o processo descrito.*

| | Parâmetros | Modelo | Total |
|---|---|---|---|
| Cadeias de Markov de ordem $M$ | 1728 | 0 | 1728 |
| Cadeias de Markov de alcance variável | 448 | 21 | 469 |
| Modelos de Markov mínimos | 320 | 36 | 356 |

*Tabela 5.2:* Tamanho da informação em bits.

*Note que o tamanho da informação que devemos enviar sobre os parâmetros da cadeia de Markov de ordem* 3 *é superior ao tamanho da informação dos outros modelos.*

## 5.1 Algortimo de compressão de dados

Neste seção apresentaremos um algoritmo de compressão de dados baseado nos modelos de Markov mínimos. Afim de alcançar uma melhor compressão dos dados modificamos o algoritmo de Huffman, apresentado no Capítulo 2.

A codificação pelo algortimo de Huffman basea-se na frequência de cada símbolo dentro dos dados. Utilizamos o algoritmo de Huffman como base para construir um algoritmo de compressão de dados usando os modelos de Markov mínimos. Para isso, considere um sistema de codificação para uma sequência $x_1^n$ sobre um alfabeto $A$ que pode ser descrito da seguinte maneira, codificamos $x_j$ usando um sistema de codificação $C(.|x_m^{j-1})$, $1 \leq m < j$, com propriedade de sufixo que depende das observações passadas da sequência, este sistema de codificação é conhecido como sistema de codificação condicional e possui a propriedade de sufixo. Note que para as cadeias de Markov de ordem $M$, os códigos são condicionados às sequências de tamanhos $m = M$, para as VLCM e para os MMM os códigos são condicionados às sequências de tamanhos máximo $M$. O próximo exemplo ilustra um sistema de codificação condicional.

**Exemplo 5.2.** *Considere uma cadeia de Markov sobre o alfabeto* $A = \{1, 2, 3, 4\}$*, com*

*probabilidades de transição*

$$P(1|1) = 0.30, \quad P(2|1) = 0.25, \quad P(3|1) = 0.15,$$
$$P(1|2) = 0.20, \quad P(2|2) = 0.30, \quad P(3|2) = 0.10,$$
$$P(1|3) = 0.50, \quad P(2|3) = 0.20, \quad P(3|3) = 0.20,$$
$$P(1|4) = 0.30, \quad P(2|4) = 0.25, \quad P(3|4) = 0.15.$$

*Para cada $a \in A$, usamos o algoritmo de Huffman para gerar uma codificação para os elementos de $A$ utilizando a distribuição de probabilidade $P(.|a)$. A Tabela 5.3 ilustra o sistema de codificação condicional resultante para o modelo descrito.*

| | $C(.\|1)$ | $C(.\|2)$ | $C(.\|3)$ | $C(.\|4)$ |
|---|---|---|---|---|
| 1 | 00 | 011 | 0 | 00 |
| 2 | 01 | 01 | 01 | 01 |
| 3 | 11 | 111 | 011 | 11 |
| 4 | 10 | 0 | 111 | 10 |

*Tabela 5.3:* Código de Huffman condicional

*Observe que não precisamos condicionar o sistema de codificação a um passado maior que 1, devido ao fato que o processo é uma cadeia de Markov de ordem* 1. *Como $P(a|1) = P(a|4)$, para todo $a \in A$, temos que $C(a|1) = C(a|4)$, $\forall a \in A$. Então podemos substituir o sistema de codificação $C(.|1)$ e $C(.|4)$ por um sistema de codificação $C(.|\{1,4\}) = C(.|1) = C(.|4)$ que une as informaçõs dos elementos* 1 *e* 4.

A idéia do algoritmo de compressão de dados proposto nesta dissetação é utilizar os MMM gerados pelo algoritmo MMM para construir um sistema de codificação condicionado à mínima partição boa. O algortimo é construído da seguinte maneira.

**Algoritmo 5.1.** *(Algoritmo de compressão de dados)*
*Seja $x_1^n$ uma sequência sobre um alfabeto finito $A$. Seja $\mathcal{L}^* = \{L_1, L_2, \ldots, L_K\}$ a mínima partição boa de $\mathcal{S} = A^M$ gerada pelo algoritmo MMM. Seja*

$$\{\hat{p}(a|L) : a \in A, L \in \mathcal{L}^*\},$$

*o conjunto das probabilidades de transição estimadas para a sequência $x_1^n$.*

1. *Para cada $L \in \mathcal{L}^*$ geramos uma codificação pelo algoritmo de Huffman descrito em 2.1, usando as probabilidades de transição estimadas $\hat{p}(.|L)$, assim obtemos um código condicional $C(.|L)$ para cada $L \in \mathcal{L}^*$.*

2. *Para codificar um símbolo $x_j$ da sequência $x_1^n$, observamos o passado $s = x_{j-M}^{j-1}$ e codificamos o símbolo $x_j$ usando $C(x_j|L)$ tal que $s \in L$.*

Para as cadeias de Markov de ordem $M$, a compressão dos dados é análoga. Geramos uma codificação de Huffman condicionada à $s$, para cada $s \in \mathcal{S} = A^M$, usando as estimativas de máxima verossimilhança das probabilidades de transição dadas por

$$\hat{p}(a|s) = \frac{\sum_{t=M+1}^{n} I\{x_{t-M}^{t-1} = s, x_t = a\}}{\sum_{t=M+1}^{n} I\{x_t = s\}}, \ a \in A, \ s \in \mathcal{S}. \tag{5.1}$$

O mesmo procedimento pode ser usado para as VLMC. Utilizando algum algoritmo de estimação da árvore de contexto. Como exemplos, o algoritmo de Contexto (Rissanen, 1983), o algoritmo PST (Ron et al., 1996) e o algoritmo CTW (Csiszár & Talata, 2006). Podemos construir uma sistema de codificação condicionado aos contextos.

**Exemplo 5.3** (Continuação do Exemplo 5.1)**.** *O sistema de codificação condicionado a mínima partição boa $\mathcal{L}$, definida na equação (4.24), para os elementos do alfabeto $A = \{0, 1, 2\}$, gerado pelo Algoritmo 5.1 está apresentado na tabela abaixo.*

| | $C(.\|L_1)$ | $C(.\|L_2)$ | $C(.\|L_3)$ | $C(.\|L_4)$ | $C(.\|L_5)$ |
|---|---|---|---|---|---|
| 0 | 10 | 0 | 11 | 01 | 01 |
| 1 | 11 | 10 | 10 | 00 | 1 |
| 2 | 0 | 11 | 0 | 1 | 00 |

*As probabilidades de $L \in \mathcal{L}$ são dadas por,*

$$\begin{aligned} P(L_1) &= 0.48, \\ P(L_2) &= 0.29, \\ P(L_3) &= 0.12, \\ P(L_4) &= 0.06, \\ P(L_5) &= 0.05. \end{aligned}$$

*O comprimento esperado de* $C(.|\mathcal{L})$ *é dado por*

$$L(C(.|\mathcal{L})) = \sum_{L\in\mathcal{L}}\sum_{a\in A} l_{C(.|L)}(a)P(a|L)P(L) = 1.52 bits.$$

O código de Huffman pode ser ineficiente para os alfabetos com cardinalidade pequena. Por exemplo, para um alfabeto $A$, com $|A| = 2$, o código de Huffman trata os dois elementos de $A$ como se tivessem a mesma probabilidade, codificando cada elemento por 0 ou 1, independentemente das probabilidades o que não comprime os dados. Uma maneira de melhorar a eficiência do código de Huffman é concatenar os elementos para criar um novo alfabeto. Fixamos $b \geq 1$, para cada $s \in A^M$, ao invés de criarmos um código de Huffman baseado nas probabilidades $P(a|s)$, para todo $a \in A$, criamos um código de Huffman baseado nas probabilidades $P(r|s)$, para todo $r \in A^b$. Chamaremos esse código de *código de Huffman em blocos.*

Suponha que $M = 3$, $b = 3$. Sejam $s = s_1s_2s_3$ e $r = r_1r_2r_3$. A probabilidade de transição $P(r|s)$ é dado por

$$\begin{aligned} P(r|s) &= P(r_1r_2r_3|s_1s_2s_3) \\ &= P(r_3|s_1s_2s_3r_1r_2r_3)P(r_2|s_1s_2s_3r_1)P(r_1|s_1s_2s_3) \\ &= P(r_3|s_3r_1r_2)P(r_2|s_2s_3r_1)P(r_1|s_1s_2s_3). \end{aligned}$$

Como as sequências $s_3r_1r_2$ e $s_2s_3r_1$ são elementos de $A^M$, então todas as probabilidades estão bem definidas e são as mesmas definidas por $P(a|s),\ \forall a \in A, s \in A^M$. Portanto, a informação sobre os parâmetros que devemos mandar continua a mesma. O próximo algoritmo é uma modificação do Algoritmo 5.1, ao invés de usarmos as probabilidades de transição $P(a|L)$, $a \in A$, $L \in \mathcal{S}$, para criar um sistema de codificação condicional, usamos as probabilidades de transição $P(r|L)$, $r \in A^b$, $L \in \mathcal{S}$.

**Algoritmo 5.2.** *Seja* $x_1^n$ *uma sequência sobre um alfabeto finito A. Seja* $\mathcal{L}^* = \{L_1, L_2, \ldots, L_K\}$ *a mínima partição boa de* $\mathcal{S} = A^M$ *gerada pelo algoritmo MMM e seja b um inteiro positivo fixo. Seja*

$$\{\hat{p}(r|L) : r \in A^b, L \in \mathcal{L}^*\},$$

*o conjunto das probabilidades de transição estimadas para a sequência* $x_1^n$.

1. *Para cada $L \in \mathcal{L}^*$ geramos um código de Huffman agrupado, usando as probabilidades de transição estimadas $\hat{p}(r|L)$, $r \in A^b$, assim obtemos um código condicional $C(.|L)$ para cada $L \in \mathcal{L}^*$.*

2. *Para codificar uma sequência $x_j^{b+(j-1)} \in A^b$ de $x_1^n$, observamos o passado $s = x_{j-M}^{j-1}$ e codificamos $x_j^{b+(j-1)}$ usando $C(x_j^{b+(j-1)}|L)$ tal que $s \in L$.*

## 5.2 Simulação

Implementamos um estudo de simulação para quatros modelos diferentes. Para cada modelo simulamos 100 amostra de tamanhos 2500, 5000, 10000 e 20000. Utilizamos o algoritmo 5.2 para codificar cada amostra e calculamos o tamanho da informação do modelo de Markov mínimo estimado pelo algoritmo MMM. Realizamos o mesmo procedimento para a codificação da amostra utilizando a cadeia de Markov de alcance variável e a cadeia de Markov de ordem $M$, estimamos a árvore de contexto pelo algoritmo CTW (Csiszár & Talata, 2006) e as probabilidades de transição da cadeia de Markov de ordem $M$ pelos estimadores de máxima verossimilhança.

O modelo 1 é o processo descrito no Capítulo 4. Denotaremos por $\mathcal{L}_1$ mínima partição boa definida na equação (4.24). A árvore de contexto $\tau_1$ associada à partição $\mathcal{L}_1$ está apresentada na Figura 5.1

Para o modelos 2 consideramos o mesmo alfabeto do modelo 1, $A = \{0, 1, 2\}$. A mínima partição boa $\mathcal{L}_2$ que define o modelo 2 é dada por

$$\begin{aligned}
L_1 &= \{000, 010, 020, 022, 100, 110, 120, 122, 200, 210, 220, 222\},\\
L_2 &= \{001, 011, 021, 101, 111, 121, 201, 211, 221\},\\
L_3 &= \{002, 012, 112, 212\},\\
L_4 &= \{102\},\\
L_5 &= \{202\},
\end{aligned}$$

e a árvore de contexto $\tau_2$ associada à partição $\mathcal{L}_2$ está apresentada na Figura 5.2, note que a árvore associada à partição $\mathcal{L}_2$ é uma árvore de contexto completa de profundidade 3.

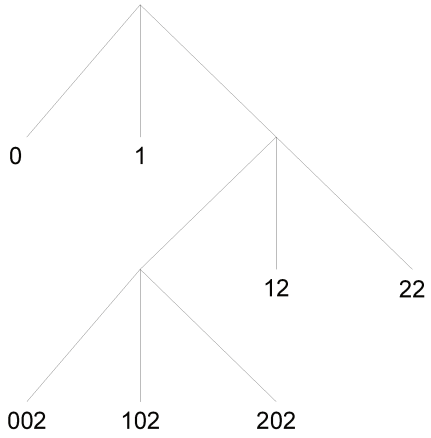


*Figura 5.1:* Árvore de contexto $\tau_1$ associada à partição $\mathcal{L}_1$.

Para os modelos 3 e 4 consideramos um alfabeto de tamanho $|A| = 4$, $A = \{0, 1, 2, 3\}$. A mínima partição boa $\mathcal{L}_3$ que define o modelo 3 é dada por

$$\begin{aligned}
L_1 &= \{000, 001, 002, 003, 010, 011, 012, 013, 020, 021, 022, 023, 030, 031, 032, 033, \\
&\quad 220, 221, 222, 223, 230, 231, 232, 233\}, \\
L_2 &= \{100, 101, 102, 103, 110, 111, 112, 113, 120, 121, 122, 123, 130, 131, 132, 133, \\
&\quad 210, 211, 212, 213\}, \\
L_3 &= \{201, 300, 301, 302, 303, 310, 311, 312, 313, 320, 321, 322, 323, 330, 331, 332, 333\}, \\
L_4 &= \{200, 203\}, \\
L_5 &= \{202\},
\end{aligned}$$

a árvore de contexto $\tau_3$ associada à partição $\mathcal{L}_3$ é uma árvore de contexto completa de profundidade 3.

Para modelo 4 consideramos $M = 4$. A mínima partição boa $\mathcal{L}_4$ que define o modelo

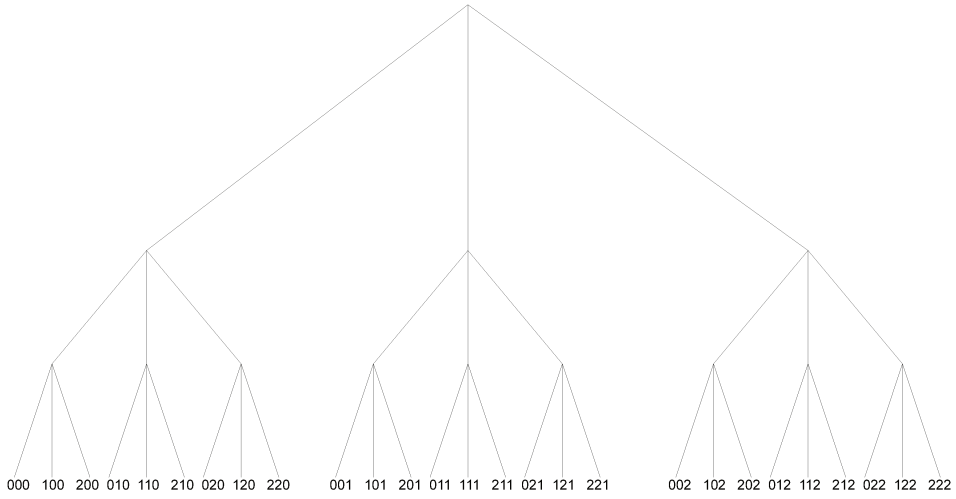


*Figura 5.2:* Árvore de contexto $\tau_2$ associada à partição $\mathcal{L}_2$.

4 é dada por

$$\begin{aligned} L_1 &= \{0**0, *0*1, **02, ***3\}, \\ L_2 &= \{1**0, *1*1, **12\}, \\ L_3 &= \{2**0, *2*1, **22\}, \\ L_4 &= \{3**0, *3*1, **32\}, \end{aligned}$$

onde $*0 = \{00, 10, 20, 30\}$, ou seja, o cojunto de todas as sequências possíveis que terminam em 0. A árvore de contexto $\tau_4$ associada à partição $\mathcal{L}_4$ é dada pelo conjunto

$$\tau_4 = \{3, 02, 12, 22, 32, 0**0, *0*1, 1**0, *1*1, 2**0, *2*1, 3**0, *3*1\}.$$

Os modelos 2 e 3 consideram uma árvore de contexto completa de profundidade 3, o que pode trazer desvantagens para a compressão baseada nas cadeias de Markov de alcance variável. Para os modelos 1 e 4, não favorecemos nenhum modelo. Na Tabela 5.4 apresentamos um resumo dos quatros tipos de modelos considerados.

Os resultados estão descritos nas tabelas seguintes. Para cada tamanho de amostra calculamos a média e o desvio padrão do tamanho total da codificação, tamanho da

| | $A$ | $M$ | $\lvert\mathcal{L}\rvert$ | $\lvert\tau\rvert$ |
|---|---|---|---|---|
| Modelo 1 | $\{0,1,2\}$ | 3 | 5 | 7 |
| Modelo 2 | $\{0,1,2\}$ | 3 | 5 | 27 |
| Modelo 3 | $\{0,1,2,3\}$ | 3 | 5 | 64 |
| Modelo 4 | $\{0,1,2,3\}$ | 4 | 4 | 85 |

*Tabela 5.4:* Modelos utilizados para o estudo de simulação.

amostra codificada e o tamanho da codificação do modelo. Calculamos também a média e o desvio padrão da diferenças entre os tamanhos das codificações dos modelos.

Para todos os modelos considerados e para os diferentes tamanhos de amostras, obtivemos melhores resultados para a compressão baseada nos modelos de Markov mínimos.

| Tamanho da amostra | | 2500 | 5000 | 10000 | 20000 |
|---|---|---|---|---|---|
| Média | MMM | **4059.29** | **7827.59** | **15306.72** | **30281.49** |
| | VLMC | 4196.16 | 7938.42 | 15407.25 | 30383.30 |
| | CC | 5427.34 | 9169.09 | 16637.62 | 31612.00 |
| Desvio Padrão | MMM | 45.03 | 45.33 | 52.66 | 69.98 |
| | VLMC | 25.07 | 39.98 | 51.57 | 69.17 |
| | CC | 28.38 | 40.43 | 52.26 | 69.85 |
| Média da diferença entre MMM e VLMC/CC | VLMC | 136.87 | 110.83 | 100.53 | 101.81 |
| | CC | 1368.05 | 1341.5 | 1330.9 | 1330.51 |
| Desvio padrão da diferença entre MMM e VLMC/CC | VLMC | 43.03 | 28.72 | 17.54 | 16.19 |
| | CC | 46.08 | 33 | 22.62 | 23.74 |

*Tabela 5.5:* Resultados da simulação do modelo 1, utilizamos a sigla CC para indicar as cadeias de Markov de ordem completa.

| Tamanho da amostra | | 2500 | 5000 | 10000 | 20000 |
|---|---|---|---|---|---|
| Média | MMM | **4050.06** | **7913.05** | **15445.4** | **30363.72** |
| | VLMC | 4373.33 | 8929.57 | 16725.77 | 31657 |
| | CC | 5416.38 | 9144.09 | 16611.7 | 31522 |
| Desvio Padrão | MMM | 50.16 | 53.99 | 48.09 | 64.64 |
| | VLMC | 165.49 | 190.63 | 56.62 | 64.06 |
| | CC | 31.53 | 38.95 | 44.29 | 64.06 |
| Média da diferença entre MMM e VLMC/CC | VLMC | 323.27 | 1016.52 | 1280.37 | 1293.28 |
| | CC | 1366.32 | 1231.04 | 1166.3 | 1158.28 |
| Desvio padrão da diferença entre MMM e VLMC/CC | VLMC | 153.93 | 171.64 | 49.04 | 22.88 |
| | CC | 56.71 | 49.56 | 26.52 | 22.88 |

*Tabela 5.6:* Resultados da simulação do modelo 2, utilizamos a sigla CC para indicar as cadeias de Markov de ordem completa.

| Tamanho da amostra | | 2500 | 5000 | 10000 | 20000 |
|---|---|---|---|---|---|
| Média | MMM | **4416.11** | **8466.49** | **16716.33** | **32848.82** |
| | VLMC | 5373.64 | 10204.09 | 20628.43 | 37982.23 |
| | CC | 9964.68 | 13962.31 | 21937.29 | 37859.94 |
| Desvio Padrão | MMM | 83.68 | 106.98 | 139.79 | 193.4 |
| | VLMC | 220.96 | 296.94 | 410.85 | 275.63 |
| | CC | 70.25 | 94.53 | 127.6 | 178.51 |
| Média da diferença entre MMM e VLMC/CC | VLMC | 957.53 | 1737.6 | 3912.1 | 5133.41 |
| | CC | 5548.57 | 5495.82 | 5220.96 | 5011.12 |
| Desvio padrão da diferença entre MMM e VLMC/CC | VLMC | 212.73 | 256.82 | 357.26 | 181.62 |
| | CC | 70.12 | 65.13 | 82.21 | 78.09 |

*Tabela 5.7:* Resultados da simulação do modelo 3, utilizamos a sigla CC para indicar as cadeias de Markov de ordem completa.

| Tamanho da amostra | | 2500 | 5000 | 10000 | 20000 |
|---|---|---|---|---|---|
| Média | MMM | **5036.17** | **9156.17** | **17210.42** | **33609.42** |
| | VLMC | 5514.93 | 10600.95 | 19768.5 | 39049.95 |
| | CC | 27386.08 | 31963.05 | 40071.87 | 56223.27 |
| Desvio Padrão | MMM | 108.01 | 115.47 | 153.91 | 227.49 |
| | VLMC | 439.61 | 102.27 | 716.02 | 591.43 |
| | CC | 294.61 | 121.94 | 105.29 | 158.71 |
| Média da diferença entre MMM e VLMC/CC | VLMC | 478.76 | 1444.78 | 2558.08 | 5440.53 |
| | CC | 22349.91 | 22806.88 | 22861.45 | 22613.85 |
| Desvio padrão da diferença entre MMM e VLMC/CC | VLMC | 399.83 | 112.08 | 633.06 | 477.1 |
| | CC | 303.97 | 127.06 | 134.45 | 154.11 |

*Tabela 5.8:* Resultados da simulação do modelo 4, utilizamos a sigla CC para indicar as cadeias de Markov de ordem completa.

# Capítulo 6

# Aplicação à linguística

Neste capítulo apresentamos os resultados obtidos a partir da aplicação do algoritmo MMM para a seleção dos modelos de Markov Mínimo, para um conjunto de dados compostos por poemas codificados. O intuito da aplicação do algoritmo MMM é verificar mudanças no modelo do contorno acentual dos poemas para diferentes movimentos literários. O conjunto dos poemas codificados estão separados por três movimentos literários o Romantismo, Parnasianismo e o Modernismo. Cada movimento literário possui diferentes características tanto na temática, como também na métrica e rima.

A poesia romântica surge em meio aos fervores independentistas da primeira metade do século XIX, tendo como marco inicial a obra de Gonçalves de Magalhães, "Suspiros Poéticos e Saudades"em 1836. O romantismo se caracterizava pelo enfoque na temática das poesias, com um alto teor emotivo, deixando de lado o rigor da métrica poética, tornado os poemas românticos mais diversificados em relação à métrica e a rima.

O parnasianismo tem seu marco inicial com a publicação de "Fanfarra"de Teófilo Dias, em 1882. Caracteriza-se pela sacralidade da forma, pelo respeito às regras de versificação, pelo preciosismo rítmico e vocabular, pela rima rica e pela preferência por estruturas fixas, assim como os sonetos.

Considera-se a Semana de Arte Moderna, realizada em São Paulo, em 1922, como ponto de partida do modernismo no Brasil. O objetivo do movimento era o rompimento com o tradicionalismo (parnasianismo, simbolismo e a arte acadêmica), a libertação estética, aproximação com a linguagem falada e a mistura do verso livre com formas tra-

dicionais de compor poemas. Na literatura o modernismo está dividido em três fases, dentro da poesia modernista destacam-se a primeira e segunda fase.

## 6.1 Apresentação dos dados

Os dados que analizamos são sequências de poemas codificados de diferentes movimentos literários, especificamente o romantismo, parnasianismo e o modernismo, separando o conjunto dos poemas do modernismo em primeira e segunda fase. Os poemas foram extraídos do site `www.jornaldepoesia.jor.br/`. Cada movimento literário é representado por um conjunto de poemas dos cinco principais poetas do movimento, as informações sobre os poetas considerados nas amostras encontram-se no Apêndice B.

Codificamos os dados da seguinte maneira, atribuímos o valor 0 ou 1 para cada sílaba do texto. Codificamos com o valor 1 as sílabas tônicas e com o valor 0 as sílabas átonas. A ortografia da língua portuguesa é construída de tal maneira que sílabas tônicas podem ser obtidas automaticamente pela forma como as palavras são escritas. Permitindo a construção do conjunto de dados de forma automática. Um programa escrito em *Perl*, de autoria de Miguel Galves, foi desenvolvido para esse propósito e para sua utilização nesta aplicação foi modificada. Para maiores discussões sobre a codificação de textos escritos indicamos (Galves et al., 2009).

Antes de codificar os poemas, realizamos um pré-processamento dos dados devido à estrutura diferenciada dos poemas. Retiramos todos os pontos de exclamação, interrogação e ponto final, encontrados nos poemas, por serem considerados finais de frase pelo algoritmo utilizado para codificar os mesmos. O próximo exemplo ilustra a forma de codificação dos poemas.

**Exemplo 6.1.** *Ilustração da codificação dos poemas.*

| *Lon* | *ge* | *de* | *ti* | *se* | *es* | *cu* | *to* | *por* | *ven* | *tu* | *ra* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |

Assim obtemos uma amostra de 14825 símbolos para o romantismo, 7984 símbolos para o parnasianismo, 14150 símbolos para o modernismo primeira fase e 19893 símbolos para o modernismo segunda fase.

## 6.2 Resultados obtidos da aplicação do algoritmo

Nesta seção apresentaremos os resultados obtidos da aplicação do algoritmo 4.1 no conjunto de poemas codificados, cujo alfabeto é $A = \{0, 1\}$. Inicializamos o algoritmo a partir da partição boa $\mathcal{S} = A^M$, onde $M = 3$ e $\alpha = 5$ são os parâmetros de entrada escolhidos.

Os resultados para cada movimento literário estão descritos na Tabela 6.1. A tabela apresenta a mínima partição boa gerada pelo algoritmo e as probabilidades de transição estimadas associada a cada partição.

| Romantismo | | | Parnasianismo | | |
|---|---|---|---|---|---|
| $L$ | $P(0\|L)$ | $P(1\|L)$ | $L$ | $P(0\|L)$ | $P(1\|L)$ |
| $\{000\}$ | 0.21 | 0.79 | $\{000\}$ | 0.26 | 0.74 |
| $\{001\}$ | 0.91 | 0.09 | $\{011\}$ | 0.82 | 0.18 |
| $\{100\}$ | 0.40 | 0.60 | $\{100\}$ | 0.45 | 0.55 |
| $\{101\}$ | 0.87 | 0.13 | | | |
| $\{010, 110\}$ | 0.60 | 0.40 | $\{010, 110\}$ | 0.58 | 0.42 |
| $\{011, 111\}$ | 0.81 | 0.19 | $\{001, 101, 111\}$ | 0.92 | 0.08 |
| Modernismo primeira fase | | | Modernismo segunda fase | | |
| $L$ | $P(0\|L)$ | $P(1\|L)$ | $L$ | $P(0\|L)$ | $P(1\|L)$ |
| $\{000\}$ | 0.32 | 0.68 | $\{000\}$ | 0.30 | 0.70 |
| $\{001\}$ | 0.90 | 0.10 | $\{010\}$ | 0.62 | 0.38 |
| $\{100\}$ | 0.44 | 0.56 | $\{100\}$ | 0.40 | 0.60 |
| $\{010, 110\}$ | 0.60 | 0.40 | $\{110\}$ | 0.54 | 0.46 |
| $\{011, 101, 111\}$ | 0.83 | 0.17 | $\{001, 011, 101, 111\}$ | 0.88 | 0.12 |

*Tabela 6.1:* Seleção dos modelos de Markov mínimos pelo algoritmo MMM para as amostras de poemas codificados.

As mínimas partições geradas pelo algoritmo para a amostra do parnasianismo e modernismo primeira fase diferem apenas na classificação dos elementos $\{011\}$ e $\{001\}$. Os conjuntos $\{000\}$ e $\{100\}$ aparecem em todas os modelos gerados pelo algorimto MMM.

Comparamos os modelos gerados pelo algoritmo MMM para os diferentes movimentos literários usando a taxa de entropia relativa. Utilizando o resultado do Teorema 3.1,

calculamos a taxa de entropia relativa dos MMM estimados para o conjunto de dados. A Tabela 6.2 apresenta os valores da taxa de entropia relativa.

| | Romantismo | Parnasianismo | Modernismo primeira fase | Modernismo segunda fase |
|---|---|---|---|---|
| Romantismo | 0 | 0.0060 | 0.0082 | 0.0071 |
| Parnasianismo | 0.0065 | 0 | 0.0111 | 0.0089 |
| Modernismo primeira fase | 0.0067 | 0.0095 | 0 | 0.0048 |
| Modernismo segunda fase | 0.0064 | 0.0083 | 0.0051 | 0 |

*Tabela 6.2:* Taxa de entropia relativa dos MMM estimados para o conjunto de poemas codificados.

Observamos que a taxa de entropia para todas as combinações dois a dois dos modelos é inferior à 0.012. Portanto, as "distâncias" entre os modelos são próximos de zero. Ou seja, não encontramos uma diferença significativa entre os MMM para os diferentes movimentos literários.

Estratificamos a amostra de cada movimento literário em cinco subconjuntos para realizar um estudo de validação cruzada dos modelos estimados. Usamos quatro dos cinco subconjuntos como amostra de treinamento e aplicamos o algoritmo MMM para construir o MMM, o subconjunto que não foi utilizado no ajuste do modelo é a amostra de validação, repetimos esse procedimento cinco vezes para cada amostras dos movimentos literários. Os resultados estão apresentados na Tabela 6.3.

Para o parnasianismo e modernismo primeira fase, o algoritmo MMM gera a mesma mínima partição boa para todos os subconjuntos das amostras. Para o romantismo apenas o elemento $\{101\}$ é classificado em diferentes conjuntos, pela Tabela 6.1 podemos observar que as probabilidades de transição do conjunto $\{101\}$ e $\{011, 111\}$ são próximas. No modernismo segunda fase apenas um subconjunto da amostra apresentou diferenças na mínima partição boa.

Denotaremos por $\hat{f}^{-i}(x_{j-M}^{j-1})$ o valor predito pelo $i$-ésimo modelo dado $x_{j-M}^{j-1}$, observado na amostra de validação, e dado por

$$\hat{f}^{-i}(x_{j-M}^{j-1}) = \arg\max_{a \in A} I\!P(X_j = a | X_{j-M}^{j-1} = x_{j-M}^{j-1}). \tag{6.1}$$

Seja $N^i$ o tamanho da amostra de validação do $i$-ésimo modelo. O erro de predição da validação cruzada estimado é dado por

| Romantismo | | | | |
|---|---|---|---|---|
| Modelo 1 | Modelo 2 | Modelo 3 | Modelo 4 | Modelo 5 |
| {000} | {000} | {000} | {000} | {000} |
| {100} | {100} | {100} | {100} | {100} |
| {001} | {001} | {001} | {001} | {001} |
| {101} | | | | {101} |
| {010, 110} | {010, 110} | {010, 110} | {010, 110} | {010, 110} |
| {011, 111} | {011, **101**, 111} | {011, **101**, 111} | {011, **101**, 111} | {011, 111} |
| Parnasianismo | | | | |
| Modelo 1 | Modelo 2 | Modelo 3 | Modelo 4 | Modelo 5 |
| {000} | {000} | {000} | {000} | {000} |
| {100} | {100} | {100} | {100} | {100} |
| {011} | {011} | {011} | {011} | {011} |
| {010, 110} | {010, 110} | {010, 110} | {010, 110} | {010, 110} |
| {001, 101, 111} | {001, 101, 111} | {001, 101, 111} | {001, 101, 111} | {001, 101, 111} |
| Modernismo primeira fase | | | | |
| Modelo 1 | Modelo 2 | Modelo 3 | Modelo 4 | Modelo 5 |
| {000} | {000} | {000} | {000} | {000} |
| {100} | {100} | {100} | {100} | {100} |
| {001} | {001} | {001} | {001} | {001} |
| {010, 110} | {010, 110} | {010, 110} | {010, 110} | {010, 110} |
| {011, 101, 111} | {011, 101, 111} | {011, 101, 111} | {011, 101, 111} | {011, 101, 111} |
| Modernismo segunda fase | | | | |
| Modelo 1 | Modelo 2 | Modelo 3 | Modelo 4 | Modelo 5 |
| {000} | {000} | {000} | {000} | {000} |
| {100} | {100} | {100} | {100} | {100} |
| {010} | {010, **110**} | {010} | {010} | {010} |
| {110} | {**001**, **011**} | {110} | {110} | {110} |
| {001, 011, 101, 111} | {101, 111} | {001, 011, 101, 111} | {001, 011, 101, 111} | {001, 011, 101, 111} |

*Tabela 6.3:* Resultado da validação cruzada cinco vezes estratificada.

| | Romantismo | Parnasianismo | Modernismo primeira fase | Modernismo segunda fase |
|---|---|---|---|---|
| Modelo 1 | 0.28 | 0.27 | 0.29 | 0.28 |
| Modelo 2 | 0.28 | 0.29 | 0.30 | 0.28 |
| Modelo 3 | **0.38** | 0.28 | 0.30 | 0.27 |
| Modelo 4 | 0.29 | 0.27 | 0.30 | 0.27 |
| Modelo 5 | 0.27 | 0.29 | 0.30 | 0.29 |

*Tabela 6.4:* Erro de predição estimado.

$$\mathrm{VC}(\hat{f}^{-i}) = \frac{1}{N^i - M} \sum_{j=M+1}^{N^i} \delta(x_j, \hat{f}^{-i}(x_{j-M}^{j-1})), \tag{6.2}$$

onde $\delta(., \hat{f}^{-i}(.))$ é alguma função de perda.

Para estimar os erros de predição dos modelos utilizamos a função de perda dada por

$$\delta(X_j, \hat{f}^{-i}(x_{j-M}^{j-1})) = \begin{cases} 1 & \text{se } X_j \neq \hat{f}^{-i}(x_{j-M}^{j-1}), \\ 0 & \text{caso contrário.} \end{cases}$$

Para cada movimento literário utilizamos as amostras de validação para estimar os erros de predição de cada modelo. Os resultados estão apresentados na Tabela 6.4, note que em apenas uma amostra de validação o erro de predição estimado foi superior à 0.30.

# Capítulo 7

# Aplicação à compressão de dados

Neste capítulo apresentamos os resultados obtidos a partir da aplicação do algoritmo de compressão de dados baseada nos modelos de Markov mínimos para as amostras dos poemas codificados. Usando o algoritmo de compressão de dados apresentado no Capítulo 5, comparamos a compressão de dados gerada pelo o algoritmo utilizando os modelos de Markov mínimos, as cadeias de Markov de ordem $M$ e as VLMC.

## 7.1 Apresentação dos dados

Para o estudo de compressão dados, utilizaremos o mesmo conjunto de poemas codificados descritos no Capítulo 6, modificando a forma de codificação das sílabas. Codificamos cada sílaba por $\{0, 1, 2, 3\}$ de acordo com suas características acentuais e morfológica, onde

- 0 = sílaba átona, não é a sílaba inicial de uma palavra prosódica;
- 1 = sílaba tônica, não é a sílaba inicial de uma palavra prosódica;
- 2 = sílaba átona, é a sílaba inicial de uma palavra prosódica;
- 3 = sílaba tônica, é a sílaba inicial de uma palavra prosódica.

A *palavra prosódica* é o conjunto formado pela palavra com seu acento principal e todas as palavras funcionais que a precedem até encontrar a palavra anterior com acento. Acrescentamos o símbolo 4 para indicar o final de cada verso. Assim obtemos quatro amostras (movimentos literários) de símbolos sobre o alfabeto $A = \{0, 1, 2, 3, 4\}$.

Antes de codificar os poemas, retiramos todos os pontos de interrogação, exclamação e pontos finais encontrados no poemas. Diferente da codificação utilizada no Capítulo 6, acrescentamos ponto final no fim de cada verso, para indicar o término de cada verso.

**Exemplo 7.1.** *Ilustração da codificação dos poemas.*

| *Lon* | *ge* | *de* | *ti* | *se* | *es* | *cu* | *to* | *por* | *ven* | *tu* | *ra* | . |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 0 | 2 | 1 | 2 | 0 | 1 | 0 | 2 | 0 | 1 | 0 | 4 |

Obtemos uma amostra de tamanho 16188 para os poemas do romantismo, 8639 para o parnasianismo, 15424 para o modernismo primeira fase e 21955 para o modernismo segunda fase.

## 7.2 Resultados

Nesta seção apresentamos os resultados obtidos pelo algoritmo de compressão de dados. Para cada movimento literário obtemos uma amostra sobre o alfabeto $A = \{0, 1, 2, 3, 4\}$. Estimamos os modelos de Markov mínimos pelo algoritmo MMM, utilizando como parâmetros de entrada do algoritmo $M = 3$ e $\alpha = 1$. Para as cadeias de Markov de ordem $M = 3$, estimamos as probabilidades de transição usando o estimador de máxima verossimilhança. Usamos o algoritmo CTW proposto em (Csiszár & Talata, 2006) para estimar a árvore de contexto, com profundidade máxima igual à 3. Os resultados da compressão de dados para os conjuntos dos poemas condificados estão apresentados na Tabela 7.1. Na tabela descrevemos o tamanho em bits da informação que devemos enviar sobre as probabilidades de transição, o tamanho em bits da informação da estrutura dos modelos e o tamanho em bits da codificação da amostra baseada nos MMM, VLMC e cadeias de Markov de ordem completa, para cada movimento literário. As informações das probabilidades de transição e dos modelos foram calculadas pelas fórmulas apresentadas na Tabela 5.1.

| Romantismo | Parâmetros | Modelo | Amostra | Total |
|---|---|---|---|---|
| Modelo de Markov Mínimo | 288 | 370 | 23977 | 24635 |
| Cadeia de Markov de alcance variável | 1184 | 222 | 23924 | 25330 |
| Cadeia de Markov de ordem 3 | 1824 | 0 | 23893 | 25717 |
| Parnasianismo | Parâmetros | Modelo | Amostra | Total |
| Modelo de Markov Mínimo | 224 | 136 | 12841 | 13201 |
| Cadeia de Markov de alcance variável | 544 | 85 | 12829 | 13458 |
| Cadeia de Markov de ordem 3 | 1824 | 0 | 12800 | 14624 |
| Modernismo primeira fase | Parâmetros | Modelo | Amostra | Total |
| Modelo de Markov Mínimo | 288 | 189 | 23740 | 24217 |
| Cadeia de Markov de alcance variável | 672 | 105 | 23736 | 24513 |
| Cadeia de Markov de ordem 3 | 1888 | 0 | 23655 | 25543 |
| Modernismo segunda fase | Parâmetros | Modelo | Amostra | Total |
| Modelo de Markov Mínimo | 352 | 370 | 33431 | 34153 |
| Cadeia de Markov de alcance variável | 672 | 222 | 33505 | 34911 |
| Cadeia de Markov de ordem 3 | 2176 | 0 | 33340 | 35516 |

*Tabela 7.1:* Tamanho da codificação para as amostras dos poemas codificados.

Quando utilizamos os MMM para comprimir os dados obtemos melhores resultados em comparação aos outros modelos, considerando a codificação da amostra e a codificação do modelo. Pela Tabela 7.1, notamos que o tamanho da codificação das amostra é menor quando usamos as cadeias de Markov de ordem $M$. No entanto, a codificação do modelo é superior aos outros modelos.

# Capítulo 8

# Conclusão

Baseado no princípio MDL definem-se os modelos de Markov mínimos; o princípio MDL estabelece que o melhor modelo é aquele que minimiza a codificação da amostra e do modelo. A definição dos modelos de Markov mínimos é motivada pelo particionamento do espaço de estados em categorias cujos estados sejam equivalentes, permitindo a modelagem da redundância que aparece em muitos processos, reduzindo o número de parâmetros do modelo.

Cada categoria no espaço de estado possui um significado muito claro, qualquer sequência de símbolos ema uma mesma categoria tem o mesmo efeito sobre a distribuição futura do processo. Em outras palavras, os membros da categoria ativam o mesmo mecanismo aleatório para escolher o próximo símbolo no processo. Podemos pensar na mínima partição resultante como uma lista de contextos relevantes para o processo e seus sinônimos.

García & González-López (2010) apresentam uma metodologia e demonstram que para um processo estacionário de memória finita é possível encontrar consistentemente um modelo de Markov mínimo para representar o processo, mostram que ainda podem encontrar consistentemente um modelo de Markov mínimo na prática utilizando o algoritmo MMM. A implicação utilitária do fato de que o processo de seleção de modelo pode ser iniciado a partir de uma partição induzida por uma árvore de contexto, é que os modelos de Markov mínimos podem ser facilmente ajustados para processos estacionários aonde os modelos de cadeia de Markov de alcance variável já funcionam, como no caso de dados linguísticos.

Nesta dissertação aplicamos o algortimo MMM para um conjunto de dados compostos por poemas codificados. Avaliamos por validação cruzada os modelos de Markov mínimos obtidos pelo algoritmo MMM para o conjunto de dados. Obtemos resultados satisfatórios, encontrando diferenças de estimação em apenas duas amostras.

Finalmente, propomos uma metodologia de compressão de dados baseada nos modelos de Markov mínimos. Essa compressão considera tanto a codificação dos dados como a codificação do modelo utilizado. Desenvolvemos um algoritmo de compressão de dados usando os modelos de Markov mínimos. Avaliamos a capacidade de compressão baseada nos modelos de Markov mínimos por simulações e pela aplicação do algoritmo à dados linguísticos. Comparando os resultados com as compressões geradas pelas cadeias de Markov de alcance variável e pelas cadeias de Markov de ordem $M$. Os resultados apresentados pela compressão utilizando os modelos de Markov mínimos, mostram melhor desempenho em comparação à compressão gerada pelas cadeias de Markov de ordem $M$ e pelas cadeias de Markov de alcance variável.

# Referências Bibliográficas


Buhlmann, P. e Wyner, A. (1999). Variable length Markov chains. *Ann. Statist*, **27**: 480-513.

Cover, T. e Thomas, J. (1991). Elements of Information Theory. *New York: Springer-Verlag*.

Cuesta-Albertos, J. A., Fraiman, R., Galves, A., Garcia, J. e Svarc, M. (2007). Identifying rhythmic classes of languages using their sonority: a Kolmogorov-Smirnov approach. *Journal of Applied Statistics*, **34**: 749-761.

Csiszár, I. e Shields, P. C. (2000).The consistency of the BIC Markov order estimator. *Ann. Statist.*, **28**: 1601-1619.

Csiszár, I. (2002). Large-scale typicality of Markov sample paths and consistency of MDL order estimators. *IEEE Trans. Inform. Theory*, **48**: 1616-1628.

Csiszar, I. e Talata, Z. (2006). Context tree estimation for not necessarily finite memory processes, via BIC and MDL. *IEEE Trans. Inf. Theory*, **52**(3): 1007-1016.

Galves A., Galves C., Garcia N. L. e Leonardi F. (2009). Context tree selection and linguistic rhythm retrieval from written texts, arXiv:0902.3619.

Garcia, J. e Lopez, V.(2010). Minimal Markov Models, arXiv:1002.0729v1.

Grünwald, P.D. (2007). The Minimum Description Length Principle. *MIT Press*.

Huffman, D.A. (1952). A method for the construction of minimum redundancy codes. *Proc. IRE*, **40**:1098-1101.

Kolmogorov, A. (1965). Three approaches to the quantitative definition of information. *Problems of Information Transmission 1*, **1**: 1-7.

Lanfredi, M. (2010). Inferência para cadeias de Markov. Tese de doutorado. *Universidade Estadual de Campinas* (versão preliminar).

Leonardi, F. (2007). Cadeias estocásticas parcimoniosas com aplicações à classificação e filogenia das sequências de proteínas. Tese de doutorado. *Universidade de São Paulo*.

Rissanen, J. (1983). A universal prior for integers and estimation by minimum description length. *Annals of Statistics* **11**: 416-431.

Rissanen, J. (1983). A Universal Data Compression System. *IEEE Transactions on Information Theory*, **29**: 656-664.

Rissanen, J. (1984). Universal coding, information, prediction and estimation. *IEEE Transactions on Information Theory*, **30**: 629-636.

Rissanen, J. (1986). Stochastic Complexity and Modeling. *Annals of Statistics*, **14**: 1080-1100.

Ron, D., Singer, Y. e Tishby, N. (1996). The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, **25**(2-3): 117-149.

Schwarz, G. (1978). Estimating the dimension of a model. *The Annals of Statistics*, **6**: 461-464.

Solomonoff, R. (1964). A formal theory of inductive inference, part 1 nd part 2. *Information and Control*, **7**: 1-22, 224-254.

# Apêndice A

# Resultados Auxiliares

**Proposição A.1.** *Seja* $\mathcal{L} = \{L_1, L_2, \ldots, L_K\}$ *uma partição boa de* $\mathcal{S}$. *Defina, para cada* $a \in A,\ L \in \mathcal{L}$, $r_n(L,a) = \frac{N_n(L,a)}{n}$ *and* $r_n(L) = \frac{N_n(L)}{n}$. *Então, para* $a \in A, L \in \mathcal{L}$,

*i.* $r_n(L,a)$ *converge para verdadeira probabilidade* $P(L,a)$ *quase certamente quando* $n \to \infty$;

*ii.* $r_n(L)$ *converge para verdadeira probabilidade* $P(L)$ *quase certamente quando* $n \to \infty$;

*iii.* $\frac{r_n(L,a)}{r_n(L)}$ *converge para verdadeira probabilidade* $P(a|L)$ *quase certamente quando* $n \to \infty$.

**Proposição A.2.** *Seja* $(X_t)_{t\in\mathbb{Z}}$ *uma cadeia de Markov de ordem* $M$ *sobre um alfabeto finito* $A$, $\mathcal{S} = A^M$ *o espaço de estados. Seja* $P(\cdot|L)$ *a probabilidade condicional sobre* $A$, *onde* $L \in \mathcal{L}$ *e* $\mathcal{L}$ *uma partição boa de* $\mathcal{S}$. *Para qualquer* $\delta > 0$ *existe* $\alpha > 0$ *tal que, eventualmente quase certamente quando* $n \to \infty$,

$$\left| \frac{N_n(L,a)}{N_n(L)} - P(a|L) \right| < \sqrt{\frac{\delta \ln(n)}{N_n(L)}} \tag{A.1}$$

*com* $M + 1 \leq \alpha \ln(n)$.

*Demonstração.* Do corolário 2 (Csiszár, I (2002)), temos que para qualquer $\epsilon > 0$ existe $\alpha > 0$ tal que, eventualmente quase certamente quando $n \to \infty$,

$$\left|\frac{N_n(s,a)}{N_n(s)} - P(a|s)\right| < \sqrt{\frac{\epsilon \ln(N_n(s))}{N_n(s)}} \tag{A.2}$$

com $M+1 \leq \alpha \ln(n)$. Onde $N_n(s,a)$ e $N_n(s)$ é definido por

$$N_n(s,a) = \left|\{t : M < t \leq n, x_{t-M}^{t-1} = s, x_t = a\}\right|;$$
$$N_n(s) = \left|\{t : M < t \leq n, x_{t-M}^{t-1} = s\}\right|.$$

Seja $\delta > 0$ e $\epsilon = \frac{\delta}{|A|^{2M}}$, temos que

$$N_n(s,a) - N_n(s)P(a|s) \;\leq\; \sqrt{\frac{\delta \ln(N_n(s))}{|A|^{2M}} N_n(s)} \tag{A.3}$$

Como $\mathfrak{L}$ é uma partição boa de $\mathcal{S}$, $s \in L$ e $L \in \mathfrak{L}$,

$$\sum_{s\in L} N_n(s,a) - P(a|L)\sum_{s\in L} N_n(s) \;\leq\; \sum_{s\in L}\sqrt{\frac{\delta \ln(N_n(s))}{|A|^{2M}} N_n(s)} \tag{A.4}$$

Seguindo as definições apresentadas no Capítulo 4, equações (4.1) e (4.2),

$$N_n(L,a) - P(a|L)N_n(L) \;\leq\; \frac{\sqrt{\delta \ln(n)}}{|A|^M}\sum_{s\in L}\sqrt{N_n(s)}. \tag{A.5}$$

Então, temos que

$$\begin{aligned}
\frac{N_n(L,a)}{N_n(L)} - P(a|L) &\leq \frac{\sqrt{\delta \ln(n)}}{|A|^M N_n(L)}|L|\sqrt{\max_{s\in L}(N_n(s))} \\
&\leq \frac{\sqrt{\delta \ln(n)}}{|A|^M N_n(L)}|A|^M\sqrt{\sum_{s\in L} N_n(s)} \\
&= \frac{\sqrt{\delta \ln(n)}}{N_n(L)}\sqrt{N_n(L)} \\
&= \sqrt{\frac{\delta \ln(n)}{N_n(L)}}
\end{aligned} \tag{A.6}$$

□

# Apêndice B

# Descrição das amostras - Poetas

Os poetas considerados nas amostras das aplicações dos Capítulos 6 e 7 para cada movimento literário estão descrito na tabela seguinte, juntamente com o número de poemas para cada poeta.

| Romantismo | | Parnsianismo | |
|---|---|---|---|
| Poetas | Poemas | Poetas | Poemas |
| Álvares de Azevedo | 7 | Alberto de Oliveira | 6 |
| Casimiro de Abreu | 7 | Luís Delfino | 5 |
| Castro Alvez | 7 | Olavo Bilac | 7 |
| Gonçalvez Dias | 8 | Raimundo Correia | 7 |
| Gonçalves de Magalhães | 3 | Vicente de Carvalho | 6 |
| Modernismo primeira fase | | Modernismo segunda fase | |
| Poetas | Poemas | Poetas | Poemas |
| Cassiano Ricardo | 7 | Carlos Drummond de Andrade | 12 |
| Guilherme de Almeida | 7 | Cecília Meireles | 15 |
| Manuel Bandeira | 7 | Jorge Lima | 7 |
| Mário de Andrade | 8 | Murilo Mendes | 14 |
| Oswald de Andrade | 8 | Vinícius de Moraes | 10 |

# Apêndice C

# Implementação dos algoritmos

Utilizamos a linguagem PERL para a implementação dos algoritmos. A implementação do algoritmo MMM apresentado nesta dissertação é uma modificação do algoritmo implementado em (García & González-López, 2010). Os algoritmos de compressão de dados foram implementados especificamente para as análises apresentadas nesta dissertação.

## C.1 Algoritmo MMM

```
#!/usr/local/bin/perl

# Paramêtros de entrada do algoritmo MMM: $alphabetfile é o arquivo com o alfabeto
A, $datafile é o arquivo dos dados, $Dmax é a profundidade máxima das sequências e
$alpha é o número mínimo de vezes que a sequência deve aparecer na amostra.

$alphabetfile = $ARGV[0];
$datafile = $ARGV[1];
$Dmax = $ARGV[2];
$alpha = $ARGV[3];
$M = $Dmax + 1;

# Alfabeto
```

```
if( open( ABC, "<$alphabetfile" ) ){
        $linea = <ABC>;
        chomp($linea);
        $LA = length($linea);
        $alphabet = $linea;
        @lsep = split( //, $linea );
        close(ABC);
}

# Amostra

$LD=0;
if (open(LIS, "<$datafile")) {
$mmm = 1;
while ($sent = <LIS>) {
        $fil=$sent;
        chomp($fil);
        @data[$mmm] = " ";
        if ( open( DATA, "<$fil" ) ) {
              while ( $linea = <DATA> ) { chomp($linea); $data[$mmm] = $data[$mmm] .
$linea; }
              $LDparcial = length($data[$mmm]);
              $LD=$LD+$LDparcial;
              $penal = log($LD)*($LA - 1)/2;
              $mmm++;
        }
}

# Cria um arquivo com as informações dos valores de N_n(s) e N_n(a, s), ∀s ∈ S^M e ∀a ∈ A,
da amostra.

open(RES, ">NSa.$datafile.txt" );
printf( RES "$LD \n $LA \n");
$l = $Dmax;
$NUMSEQ = $LA**$l;
```

```
for ( $n = 0 ; $n < $NUMSEQ ; $n++ ) {
        @d[0] = $n % $LA;
        $ss = @d[0];
        for ( $k = 1 ; $k < $l ; $k++ ) {
                $r = ( $n - $ss ) / ( $LA**$k );
                @d[$k] = $r % $LA;
                $ss = $ss + @d[$k]*$LA**$k;
        }
        $node = " ";
        for ( $k = 0 ; $k < $l ; $k++ ) {
                $node = $node . @lsep[@d[$k]];
        }
        $seque[$n] = $node;
        $smatch = "?=" . $node;
        $NS = 0;
        for($lll = 1;$lll < $mmm;$lll++){
        $NSparcial = 0;
        $NSparcial = $data[$lll] =~ s/($smatch)/$1/g;
        $NSparcial = 1 * $NSparcial;
        $NS = $NS+$NSparcial;
        }
        if ( $NS >= $alpha ) {
                printf( RES "$node $NS");
                $IS[$n] = 1;
        for ( $k = 0 ; $k < $LA ; $k++ ) {
                $nodea = $node . @lsep[$k];
                $smatch = "?=" . $nodea;
                $NSa = 0;
                for($lll = 1;$lll < $mmm; $lll++){
                        $NSparcial = 0;
                        $NSparcial = $data[$lll] =~ s/($smatch)/$1/g;
                        $NSparcial = 1 * $NSparcial;
```

```
                $NSa=$NSa+$NSparcial;
            }
            printf( RES "$NSa");
        }
        }
        else{ $IS[$n] = 0;}
}
close(RES);

# Encontra a mínima partição boa para a amostra.

if( open( DATA, "< $NSa.$datafile.txt")) {
        $N = <DATA>;
        $K = 0;
        while ( $linea = <DATA> ) {
            chomp($linea);
            $data[$K] = $linea;
            $K++;
        }
        $penal = log($N) * ( $LA - 1 ) / 2;
}
$reduction = 1;
while($reduction == 1){
        $r = 10**30;
        $reduction = 0;
        for ( $n = 0 ; $n < ($K -1) ; $n++ ) {
            @datan= split( / /,$data[$n]);
            for ( $m = ($n+1) ; $m < $K ; $m++ ) {
                @datam = split( / /,$data[$m]);
                $d = 0;
                $Nn = $datan[1];
                $Nm = $datam[1];
                for ( $l = 2 ; $l <= ($LA + 1) ; $l++ ) {
                    $Nna = $datan[$l];
```

```
                    $Nma = $datam[$l];
                    if($Nna > 0){$d = $d + $Nna*log($Nna/$Nn)}
                    if($Nma > 0){$d = $d + $Nma*log($Nma/$Nm)}
                    $nada = $Nma+$Nna;
                    if($nada > 0){$d = $d - ($Nma+$Nna)*log(($Nma+$Nna)/
($Nm+$Nn));}
                }
                $d = $d/$penal;
                if($d < $r){$r = $d; $cn = $n ; $cm = $m;}
            }
        }
        if($r < 1){
            $reduction = 1;
            @datan = split( / /,$data[$cn]);
            @datam = split( / /,$data[$cm]);
            $datan[0] = $datan[0].'#'.$datam[0];
            $datan[1] = $datan[1]+$datam[1];
            $dat = $datan[0].' '.$datan[1];
            for ( $l = 2 ; $l <= ($LA + 1) ; $l++ ) {
                $datan[$l] = $datan[$l]+$datam[$l];
                $dat = $dat.' '.$datan[$l];
            }
            @data[$cn] = $dat;
            @data[$cm] = $data[($K-1)];
            $K - -;
            if($K == 1){$reduction = 0}
        }
}
```

# Cria um arquivo com as informações das categorias $L \in \hat{\mathfrak{L}}$, e as informações de $N_n(L)$ e $N_n(a, L)$, $\forall L \in \hat{\mathfrak{L}}$ e $\forall a \in A$.

```
open(RES, ">Clases.$datafile.txt");
printf( RES "$LD \n $LA \n");
```

```
for ( $n = 0 ; $n < ($K) ; $n++ ) {
      printf( RES "@data[$n] \n");
}
close(RES);
}
```

# Cria um arquivo com as frequências de cada elemento do alfabeto dado que observamos qualquer sequência $s \in L$, $\forall L \in \hat{\mathcal{L}}$. Este arquivo será utilizado posteriormente para a codificação dos dados.

```
@data1 = sort { lc($a) cmp lc($b) } @data;
@data = @data1;
$numn = substr(@data[0], 0, 1);
if($numn eq "#"){@data[0] = substr(@data[0], 1)}
      open(RES,">FREQ.$datafile.txt");
      for ( $n = 0 ; $n < ($K) ; $n++ ) {
            @data1 = split( / /,$data[$n]);
            $imprime = @data1[0]."====";
            printf( RES3 "%s",$imprime);
            for($m = 2;$m <= ($LA+1);$m++){printf( RES3 "@data1[$m]");}
            printf( RES3 "\n");
}
close(RES3);
```

## C.2 Algoritmos de compressão de dados baseada nos modelos de Markov mínimos

Apresentamos a implementação do algoritmo de compressão de dados baseada nos modelos de Markov mínimos, utilizando a codificação de Huffman e a codificação de Huffman em blocos.

### C.2.1 Algoritmo codificação de Huffman

```
#!/usr/local/bin/perl
```

# Paramêtros de entrada do algoritmo: $alphabetfile é o arquivo com o alfabeto $A$, $datafile é o arquivo dos dados e $Dmax é a profundidade máxima das sequências.

```
$alphabetfile = $ARGV[0];
$datafile = $ARGV[1];
$Dmax = $ARGV[2];

# Alfabeto

if (open(ABC, "<$alphabetfile")) {
      $linea = <ABC>;
      chomp($linea);
      $LA = length($linea);
      @alpha = split( //, $linea );
      close(ABC);
}
```

# Utilizando o arquivo **FREQ.$datafile.txt** gerado pelo algoritmo MMM para a amostra, o algoritmo cria um arquivo com as informações da codificação de Huffman condicionado à partição $\hat{\mathcal{L}}$.

```
$data = " ";
open( DATA, "<FREQ.$datafile.txt" );
open( RESULTS,">COD.$datafile.txt");
$K=0;
$linea = "UNIFORME====";
for($n=1;$n <= $LA;$n++){$linea = $linea."50 ";}
@pasado = split( /====/,$linea);
$freq = @passado[1];
$LALPH = $LA;
$K = $LA;
$#data = -1;
@data = ();
$#data0 = -1;
@data0 = ();
```

```
$#data1 = -1;
@data1 = ();
$#datan = -1;
@datan = ();
$#data4 = -1;
@data4 = ();
@data4 = split( / /,$freq);
for ( $n = 0 ; $n < $K ; $n++ ){
        $data[$n]=$alpha[$n].“====”.$data4[$n];
        $codigo“$alpha[$n]” = “ ”;
}
while($K > 1){
        @data0 = split( /====/,$data[0]);
        @data1 = split( /====/,$data[1]);
        if($data0[1] <= $data1[1]){
        $r0 = $data0[1];
        $r1 = $data1[1];
        $k0 = 0;
        $k1 = 1;
        $s0 = $data0[0];
        $s1 = $data1[0];
        }
        else{
        $r1 = $data0[1];
        $r0 = $data1[1];
        $k0 = 1;
        $k1 = 0;
        $s0 = $data1[0];
        $s1 = $data0[0];
        }
        $reduction=0;
        for ( $n = 2 ; $n < $K ; $n++ ){
```

```
            @datan = split( /====/,$data[$n]);
            if(@datan[1] <= $r0){
           $r1 = $r0; $r0 = @datan[1]; $k1 = $k0; $k0 = $n; $s1 = $s0; $s0 = @datan[0];}
            elsif(@datan[1] <= $r1){$r1 = @datan[1]; $k1 = $n; $s1 = @datan[0]}
      }
      @simbolos0 = split( /==..==/, $s0);
      @simbolos1 = split( /==..==/, $s1);
      $Ls0 = @simbolos0;
      $Ls1 = @simbolos1;
      for($n=0;$n<$Ls0;$n++){$codigo{"$simbolos0[$n]"} =
"0".$codigo{"$simbolos0[$n]"};}
      for($n=0;$n<$Ls1;$n++){$codigo{"$simbolos1[$n]"} =
"1".$codigo{"$simbolos1[$n]"};}
      $s0 = $s0."==..==".$s1;
      $r0 = $r0+$r1;
     if($k0 < $k1){@data[$k0] = $s0."====".$r0; @data[$k1] = @data[($K-1)]; $K- -;}
      else{@data[$k1] = $s0."====".$r0; @data[$k0] = @data[($K-1)]; $K- -;}
}
$mostrar = $passado[0]."====";
printf(RESULTS "%s",$mostrar);
for ( $n = 0 ; $n < $LALPH ; $n++ ){$DDD = $codigo{"$alpha[$n]"};
printf(RESULTS "$DDD")}
printf(RESULTS "\n");
while ( $linea = <DATA> ) {
      chomp($linea);
      @pasado = split( /====/,$linea);
      $freq = @pasado[1];
      $LALPH = $LA;
      $K = $LA;
      $#data = -1;
      @data = ();
      $#data0 = -1;
```

```
@data0 = ();
$#data1 = -1;
@data1 = ();
$#datan = -1;
@datan = ();
$#data4 = -1;
@data4 = ();
@data4 = split( / /,$freq);
for ( $n = 0 ; $n < $K ; $n++ ){
    $data[$n] = $alpha[$n].“====”.$data4[$n];
    $codigo{“$alpha[$n]”} = “ ”
}
while($K > 1){
    @data0 = split( /====/,$data[0]);
    @data1 = split( /====/,$data[1]);
    if($data0[1] <= $data1[1]){
        $r0 = $data0[1];
        $r1 = $data1[1];
        $k0 = 0;
        $k1 = 1;
        $s0 = $data0[0];
        $s1 = $data1[0];
    }
    else{$r1 = $data0[1]; $r0 = $data1[1]; $k0 = 1; $k1 = 0; $s0 = $data1[0]; $s1
= $data0[0]}
    $reduction=0;
    for ( $n = 2 ; $n < $K ; $n++ ) {
        @datan = split( /====/,$data[$n]);
        if(@datan[1] <= $r0){$r1 = $r0; $r0 = @datan[1]; $k1 = $k0; $k0 = $n;
$s1 = $s0; $s0 = @datan[0];}
        elsif(@datan[1] <= $r1){$r1 = @datan[1]; $k1 = $n; $s1 = @datan[0]}
    }
```

```
            @simbolos0 = split( /==..==/,$s0);
            @simbolos1 = split( /==..==/,$s1);
            $Ls0 = @simbolos0;
            $Ls1 = @simbolos1;
            for($n=0;$n<$Ls0;$n++){$codigo{"$simbolos0[$n]"} =
"0".$codigo{"$simbolos0[$n]"};}
            for($n=0;$n<$Ls1;$n++){$codigo{"$simbolos1[$n]"} =
"1".$codigo{"$simbolos1[$n]´´};}
            $s0 = $s0."==..==".$s1;
            $r0 = $r0+$r1;
            if($k0 < $k1){
                  @data[$k0] = $s0."====".$r0;
                  @data[$k1] = @data[($K-1)];
                  $K- -;
            }
            else{
                  @data[$k1] = $s0."====".$r0;
                  @data[$k0] = @data[($K-1)];
                  $K- -;
            }
      }
$mostrar = $passado[0]."====";
printf(RESULTS "%s",$mostrar);
for ( $n = 0 ; $n < $LALPH ; $n++ ) { $DDD = $codigo{"$alpha[$n]"};
printf(RESULTS "$DDD")}
printf(RESULTS "\n");
}
close(RESULTS);
close(DATA);
```

### C.2.2 Algoritmo de compressão usando Huffman

# Codifica os dados utilizando a codificação gerada pelo algoritmo descrito em C.2.1.

```
$dim = $Dmax;
%codigo = ();
%existecodigo=();
$mmmm = 0;
$LCLASSES = 0;
$data = " ";
open( DATA, "<COD.$datafile.txt" );
$K = 0;
while ( $linea = <DATA> ) {
      chomp($linea);
      $LCLASSES++;
      @passado = split( /====/,$linea);
      @freq = split( / /,@pasado[1]);
      @sequ = split( /#/,@pasado[0]);
      foreach $cod (@sequ){
            for($n=0;$n<$LA;$n++){
                  $codigo{"$cod,$alpha[$n]´´} = $freq[$n];
                  $existecodigo{"$cod"} = 1;
            }
      }
}
close(DATA);
open( data, "<$datafile" );
$data=" ";
while ( $linea = <DATA> ) { chomp($linea); $data = $data . $linea; }
close(data);
$COD = " ";
$LD = length($data);
for($n = 0;$n < $dim; $n++){
      $letra = substr($data,$n,1);
      $COD = $COD.$codigo{"@sequ[0],$letra"};
}
```

```
open( RESULTS,“>COD.FINAL.$datafile.txt”);
printf(RESULTS “%s”, $COD);
close(RESULTS);
$COD=“ ”;
for($n = $dim; $n < $LD; $n++){
      $trecho = substr($data,($n-$dim),$dim);
      $letra = substr($data,$n,1);
      if($existecodigo{“$trecho”}){$COD = $COD.$codigo{“$trecho,$letra”}}
      else{$COD = $COD.$codigo{“@sequ[0],$letra”}}
}
open( RESULTS,“>>COD.FINAL.$datafile.txt”);
printf(RESULTS “%s \n”, $COD);
close(RESULTS);
```

### C.2.3 Algoritmo codificação de Huffman em blocos

```
#!/usr/local/bin/perl
```

# Paramêtros de entrada do algoritmo: $alphabetfile é o arquivo com o alfabeto $A$, $datafile é o arquivo dos dados, $Dmax é a profundidade máxima das sequências e $b é o tamanho do bloco.

```
$alphabetfile = $ARGV[0];
$datafile = $ARGV[1];
$Dmax = $ARGV[2];
$b = $ARGV[3];
```

# Calcula a frequência dos elementos do alfabeto $A^b$ dentro da amostra.

```
if ( open( ABC, “<$alphabetfile” ) ) {
      $linea = <ABC>;
      chomp($linea);
      $LA = length($linea);
      @alpha = split( //, $linea );
close(ABC);
```

```
}
$LALPH = $LA;
$K = $LA;
$data = “”;
open( DATA, “<FREQ.$datafile.txt” ) ;
open( RESULTS,“>BL.FREQ.$datafile.txt”);
while ( $linea = <DATA> ) {
        chomp($linea);
        @pasado = split( /====/,$linea);
        @frequencias = split( / /,$passado[1]);
        @sequencias = split( /#/,$passado[0]);
        $totfreq = 0;
        foreach $rr (@frequencias){$totfreq = $totfreq+$rr}
        foreach $seq (@sequencias){
                for($k = 0;$k < $LA; $k++){$proseqlet{“$seq,$alpha[$k]”} =
$frequencias[$k]/$totfreq}
        }
}
$l = $b;
printf(RESULTS “UNIFORME====”);
for ( $n = 0 ; $n < ( $LA**$l ) ; $n++ ) {
        printf(RESULTS “1.0 ”);
}
printf(RESULTS “\n”);
for ( $n = 0 ; $n < ( $LA**$Dmax ) ; $n++ ) {
        @d[0] = $n % $LA;
        $ss = @d[0];
        for ( $k = 1 ; $k < $l ; $k++ ) {
                $r = ( $n - $ss ) / ( $LA**$k );
                @d[$k] = $r % $LA;
                $ss = $ss + @d[$k] * $LA**$k;
        }
```

```
        $nodes = " ";
        for ( $k = 0 ; $k < $Dmax ; $k++ ){$nodes = $nodes . @lsep[ @d[$k] ];}
        printf(RESULTS "$nodes====");
        for ( $m = 0 ; $m < ( $LA**$l ) ; $m++ ) {
                @d[0] = $m % $LA;
                $ss = @d[0];
                for ( $k = 1 ; $k < $l ; $k++ ) {
                        $r = ( $m - $ss ) / ( $LA**$k );
                        @d[$k] = $r % $LA;
                        $ss = $ss + @d[$k] * $LA**$k;
                }
                $nodea = " ";
                for ( $k = 0 ; $k < $l ; $k++ ){$nodea = $nodea . @lsep[ @d[$k] ];}
                $probcondseqseq{"$nodea,$nodes"} = 1;
                for ( $k = 0 ; $k < $l ; $k++ ) {
                        $cond = substr($nodes,$k,($l-$k)).substr($nodea,0,($k));
                        $letra = substr($nodea,$k,1);
                        $probcondseqseq{"$nodea,$nodes"} =
$probcondseqseq{"$nodea,$nodes"}*$proseqlet{"$cond,$letra"};
                }
                $rr = $probcondseqseq{"$nodea,$nodes"};
                printf(RESULTS "$rr ");
        }
        printf(RESULTS "\n");
}
close(RESULTS);
close(DATA);
```

# Cria uma codificação de Huffman em blocos condicionada à partição $\hat{\mathcal{L}}$.

```
for ( $n = 0 ; $n < ( $LA**$Dmax ) ; $n++ ) {
        @d[0] = $n % $LA;
        $ss = @d[0];
        for ( $k = 1 ; $k < $l ; $k++ ) {
```

```
            $r = ( $n - $ss ) / ( $LA**$k );
            @d[$k] = $r % $LA;
            $ss = $ss + @d[$k] * $LA**$k;
        }
        $nodes = “”;
        for ( $k = 0 ; $k < $Dmax ; $k++ ){$nodes = $nodes . @lsep[ @d[$k] ];}
        @alpha[$n]=$nodes;
}
$LA = ( $LA**$l );
$data = “ ”;
open( DATA, “<BL.FREQ.$datafile.txt” ) ;
open( RESULTS,“>BL.COD.$datafile.txt”);
$K=0;
while ( $linea = <DATA> ) {
        chomp($linea);
        @passado = split( /====/,$linea);
        $freq = @passado[1];
        $LALPH = $LA;
        $K = $LA;
        $#data = -1;
        @data=();
        $#data0 = -1;
        @data0=();
        $#data1 = -1;
        @data1=();
        $#datan = -1;
        @datan=();
        $#data4 = -1;
        @data4 = ();
        @data4 = split( / /,$freq);
        for ( $n = 0 ; $n < $K ; $n++ ){ $data[$n] = $alpha[$n].“====”.$data4[$n];
$codigo{“$alpha[$n]”} = “ ”}
```

```
    while($K > 1){
        @data0 = split( /====/,$data[0]);
        @data1 = split( /====/,$data[1]);
        if($data0[1]<=$data1[1]){$r0 = $data0[1]; $r1 = $data1[1]; $k0 = 0; $k1 = 1;
$s0 = $data0[0]; $s1 = $data1[0]}
        else{$r1 = $data0[1]; $r0 = $data1[1]; $k0 = 1; $k1 = 0; $s0 = $data1[0]; $s1
= $data0[0]}
        $reduction = 0;
        for ( $n = 2 ; $n < $K ; $n++ ) {
            @datan = split( /====/,$data[$n]);
            if(@datan[1]<=$r0){$r1 = $r0; $r0 = @datan[1]; $k1 = $k0; $k0 = $n;
$s1 = $s0; $s0 = @datan[0];}
            elsif(@datan[1]<=$r1){$r1 = @datan[1]; $k1 = $n; $s1 = @datan[0]}
        }
        @simbolos0 = split( /==..==/,$s0);
        @simbolos1 = split( /==..==/,$s1);
        $Ls0 = @simbolos0;
        $Ls1 = @simbolos1;
        for($n=0;$n<$Ls0;$n++){$codigo{"$simbolos0[$n]"} =
"0".$codigo{"$simbolos0[$n]"};}
        for($n=0;$n<$Ls1;$n++){$codigo{"$simbolos1[$n]"} =
"1".$codigo{"$simbolos1[$n]"};}
        $s0 = $s0."==..==".$s1;
        $r0 = $r0+$r1;
        if($k0 < $k1){@data[$k0] = $s0."====".$r0; @data[$k1] = @data[($K-1)];
$K- -;}
        else{@data[$k1] = $s0."====".$r0; @data[$k0] = @data[($K-1)]; $K- -;}
    }
$mostrar = $passado[0]."====";
printf(RESULTS "%s", $mostrar);
for ( $n = 0 ; $n < $LALPH ; $n++ ){$DDD = $codigo{"$alpha[$n]"};
printf(RESULTS "$DDD")}
```

```
printf(RESULTS “\n”);
}
close(RESULTS);
close(DATA);
```

### C.2.4 Algoritmo de compressão usando Huffman em blocos

# Codifica os dados utilizando a codificação gerada pelo algoritmo descrito em C.2.3.

```
$dim = $Dmax;
%codigo = ();
%existecodigo = ();
$mmmm = 0;
$data = “ ”;
open( DATA, “<BL.COD.$datafile.txt” ) ;
$K = 0;
$LCLASSES = 0;
while ( $linea = <DATA> ) {
      chomp($linea);
      @passado = split( /====/,$linea);
      $LCLASSES++;
      @freq = split( / /,@pasado[1]);
      @sequ = split( /#/,@pasado[0]);
      foreach $cod (@sequ){
            for ( $n = 0 ; $n < ( $LA**$l ) ; $n++ ) {
            @d[0] = $n % $LA;
            $ss = @d[0];
            for ( $k = 1 ; $k < $l ; $k++ ) {
                  $r = ( $n - $ss ) / ( $LA**$k );
                  @d[$k] = $r % $LA;
                  $ss = $ss + @d[$k] * $LA**$k;
            }
            $node = “ ”;
            for ( $k = 0 ; $k < $l ; $k++ ) {$node = $node . @lsep[ @d[$k] ];}
```

```
            $codigo{"$cod,$node"} = $freq[$n];
            $existecodigo"$cod" = 1;
            }
        }
}
close(DATA);
open( data, "<$datafile" );
$data=" ";
while ( $linea = <DATA> ) { chomp($linea); $data = $data . $linea; }
close(data);
$COD = " ";
$LD = length($data);
$letra = substr($data,0,$Dmax);
$COD = $COD.$codigo{"@sequ[0],$letra"};
open( RESULTS,">BL.COD.FINAL.$datafile.txt");
printf(RESULTS "%s", $COD);
close(RESULTS);
$COD = " ";
for($n = $dim; $n < ($LD-$Dmax);$n = $n+$Dmax){
        $trecho = substr($data,($n-$Dmax),$Dmax);
        $letra = substr($data,$n,$Dmax);
        if($existecodigo{"$trecho"}){
            $COD = $COD.$codigo{"$trecho,$letra"}
        }
        else{$COD = $COD.$codigo{"@sequ[0],$letra"};}
}
open( RESULTS,">>BL.COD.FINAL.$datafile.txt");
printf(RESULTS "%s\n", $COD);
close(RESULTS);
open( RESULTS,"<BL.COD.FINAL.$datafile.txt");
$linea = <RESULTS>;
chomp($linea);
```

```
$LCOM = length($linea);
close(RESULTS);
```