UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

GUSTAVO HENRIQUE TASCA

MODELOS MARKOVIANOS DE PARTIÇÃO E SUAS GENERALIZAÇÕES

CAMPINAS

2021

GUSTAVO HENRIQUE TASCA

MODELOS MARKOVIANOS DE PARTIÇÃO E SUAS GENERALIZAÇÕES

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutor em Estatística.

Orientadora: Verónica Andrea González López
Coorientador: Jesus Enrique Garcia

ESTE TRABALHO CORRESPONDE À VERSÃO FINAL DA TESE DEFENDIDA PELO ALUNO GUSTAVO HENRIQUE TASCA, E ORIENTADA PELA PROFA. DRA. VERÓNICA ANDREA GONZÁLEZ LÓPEZ.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

Tasca, Gustavo Henrique, 1990-
T181m Modelos markovianos de partição e suas generalizações / Gustavo Henrique Tasca. – Campinas, SP : [s.n.], 2021.

Orientador: Verónica Andrea González López.
Coorientador: Jesus Enrique Garcia.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Cadeias de Markov. 2. Códigos binários. 3. Critério de informação Bayesiano (BIC). I. González-López, Verónica Andrea, 1970-. II. Garcia, Jesus Enrique, 1966-. III. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Partition Markov models and their generalizations
Palavras-chave em inglês:
Markov chains
Binary codes
Bayesian Information Criterion (BIC)
Área de concentração: Estatística
Titulação: Doutor em Estatística
Banca examinadora:
Verónica Andrea González-López
Roberto Andreani
Ricardo Sandes Ehlers
Márcio Luis Lanfredi Viola
Marcio Alves Diniz
Data de defesa: 09-09-2021
Programa de Pós-Graduação: Estatística

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0001-6917-7072
- Currículo Lattes do autor: http://lattes.cnpq.br/5883346461318637

Tese de Doutorado defendida em 09 de setembro de 2021 e aprovada pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). VERÓNICA ANDREA GONZÁLEZ LÓPEZ

Prof(a). Dr(a). ROBERTO ANDREANI

Prof(a). Dr(a). RICARDO SANDES EHLERS

Prof(a). Dr(a). MÁRCIO LUIS LANFREDI VIOLA

Prof(a). Dr(a). MARCIO ALVES DINIZ

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Agradecimentos

Agradeço a Deus, por me agraciar com a força necessária para aproveitar todas as oportunidades com as quais tem me presenteado.

Aos meus pais, Natale e Cida, minhas irmãs, Michelle e Vanessa, e meu padrinho, André Vecchiato, por todo apoio oferecido.

À minha noiva, Thaíse Mendes, por ser uma companheira fiel durante esta jornada.

À professora Verónica González López, por ter aceitado me orientar no programa de doutorado e por ter me provido com toda a atenção, compreensão e auxílio, dentro e fora do circuito acadêmico.

Ao professor Jesús García, por ter aceitado me coorientar no programa de doutorado e me socorrer durante atribulações computacionais.

Aos professores Roberto Andreani, Ricardo Ehlers, Márcio Viola e Marcio Diniz, por terem aceitado o convite para a composição da banca examinadora deste trabalho, e pelas valiosas sugestões para o melhoramento do mesmo.

Aos amigos Fernando El Kadri, Claudia Koda, Rodney Fonseca, José Alejandro Ordoñez e Mário de Sousa, que, de uma forma ou de outra, me apoiaram e contribuíram para a conclusão desta etapa.

Ao apoio financeiro recebido para a conclusão deste programa de doutorado. O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Resumo

Retornamos aos fundamentos dos códigos de Huffman, justificamos, com base em elementos da teoria de informação, sua praticidade, limitações e apontamos as vantagens e desvantagens de considerarmos suas versões condicionais. Utilizamos cadeias de Markov para compor classes de condicionamento para códigos de Huffman e evidenciamos que, apesar dessa estratégia permitir uma redução no comprimento esperado por elemento da mensagem, exige a comunicação de um grande número de parâmetros para estabelecer o modelo empregado. Reintroduzimos o conceito e resultados associados a modelos markovianos de partição. Essa família de modelos organiza o espaço de estados associado a uma cadeia de Markov em partes baseadas em suas funções de probabilidade de transição. Estabelecemos a aliança entre códigos condicionais e tais modelos a fim de construir uma metodologia com potencial de reduzir o número de parâmetros a serem comunicados durante o processo de codificação de mensagens, provocando uma representação binária mais compacta. Além disso, estabelecemos uma nova classe de processos markovianos: as cadeias de Markov com interstício. Esse delineamento possibilita a inclusão de um passado distante e relevante na modelagem do processo estocástico, cujo manejo apropriado permite a redefinição dos modelos de partição e a construção de uma família ainda mais flexível: os G3M (modelos markovianos mínimos com interstício). Mostramos como utilizar os G3M para uma modelagem eficiente do genoma da Covid-19, para codificação de matrizes e para compressão de imagens com perda de qualidade.

Palavras-chave: cadeias de Markov, códigos binários, critério de informação bayesiano (BIC).

Abstract

We come back to the roots of the Huffman codes and, based on elements of information theory, we justify its practicality and limitations, as well the benefits and drawbacks of its conditional forms. We design conditioning classes to Huffman codes based on Markov chains and point out that this strategy allows a reduction on the message average length per symbol, but demands the transmission of a large set of parameters to establish the selected model. We reintroduce the concept of partition Markov models and their related results. This family of models arranges the state space of a Markov chain in sets (parts) based on its transition probability functions. We establish the alliance between conditional encoding and such models for the purpose of deriving a methodology with the potential to lessen the number of parameters to be communicated during the encoding process, which yields a shorter binary representation. Furthermore, we establish a new class of Markovian processes: the Markov chain with a gap. This design allows that a faraway, but relevant, past to be included during the modeling of the stochastic process, whose proper handling leads to a redefinition of the partition models and the characterization of an even more flexible family: the G3M (minimal Markovian models with a gap). We show how to make use of the G3M to efficiently model the Covid-19 genome, and also how to integrate it to matrix encoding processes and lossy image compression.

Keywords: Markov chains, binary codes, Bayesian information criterion (BIC).

Sumário

Introdução

Ao longo da história, a comunicação teve sua importância muito bem estabelecida nas mais diferentes culturas. A fim de transmitir conhecimento a si mesmo ou a outrem, em possíveis diferentes momentos do espaço-tempo, várias estratégias são empregadas, como a fala, a escrita e a arte gráfica. O uso de cada uma dessas abordagens varia conforme a conveniência dos indivíduos envolvidos no processo de comunicação, assim como a informação a ser repassada. Por exemplo, para a comunicação de um pensamento, as produções falada e escrita podem ser adequadas, mas, para a comunicação de um estímulo externo (como a descrição de um objeto), uma imagem pode ser mais indicada.

Ao passo que as relações humanas se desenvolviam, surgia a necessidade de técnicas que permitissem a transmissão de conhecimento de maneira simples e rápida. Por exemplo, para adaptar a linguagem escrita para o uso diário e informal, decorreu o surgimento de alfabetos cursivos (forma de escrita mais rápida e fluida) em diferentes civilizações e em diferentes períodos históricos ([7] aponta seu uso antes de 300 a.C. na cultura grega). Outra necessidade que se manifestou ao longo da história foi a de rápida comunicação entre indivíduos em diferentes localidades. Para isso, estímulos visuais (como sinais luminosos ou de fumaça) ou sonoros (como ressoo de instrumentos, por exemplo, tambores ou sinos) são emitidos pelo transmissor da mensagem e captados por receptores dentro do raio de alcance do meio empregado. Inicialmente, a informação transmitida pertencia a um pequeno universo de possibilidades preestabelecidas, como “perigo” ou “reunir o povo”, mas, com o desenvolvimento de novas técnicas de comunicação, o conjunto de mensagens possíveis de serem transmitidas passou a crescer exponencialmente.

Seguido do avanço nos conhecimentos sobre eletromagnetismo, a comunicação à distância obteve um salto significativo na primeira metade do século XIX com a criação do telégrafo elétrico. Este aparelho era capaz de transmitir pulsos elétricos de maneira

controlada, permitindo a associação de significado a cada sequência de estímulos compostos por pulso e silêncio (intervalo em que não há pulso). Dessa forma, dado um alfabeto preestabelecido entre emissor e receptor, foi possível associar diferentes sequências de estímulos a diferentes elementos do alfabeto, permitindo que mensagens mais elaboradas fossem transmitidas. O código morse, atribuído a Samuel F. B. Morse, é um dos alfabetos mais conhecidos entre os sistemas de codificação elétricos e foi utilizado até o início do século XXI em áreas da aviação.

Com a ascensão dos circuitos digitais em meados do século XX, foi possível o desenvolvimento de computadores digitais. Essas máquinas, em geral, são capazes de trabalhar velozmente com informações provenientes do menor alfabeto possível, o sistema binário. Como o próprio nome aponta, este conjunto é formado apenas por dois elementos (comumente denotados por "0" e "1") e, apesar de ter suas origens enraizadas na antiguidade, um refinamento notável de sua lógica foi concebido no século XVIII por Gottfried W. Leibniz. Assim como para o telégrafo elétrico, após preestabelecido um conjunto de regras entre emissor e receptor, mensagens podem ser transmitidas e interpretadas utilizando-se o sistema binário. O interessante é que, devido à agilidade dos computadores digitais em processar informações, mensagens mais complexas podem ser consideradas, como textos extensos, áudios e imagens.

Repare que, em todos os processos de comunicação supracitados, a mensagem alvo da transmissão é convertida em uma componente constituída por elementos de algum conjunto conveniente. Por exemplo, uma tribo pode predeterminar que um sinal de fumaça de certa cor representa a informação "perigo", enquanto esta mesma mensagem é transmitida como pulsos elétricos em um telégrafo ou como 0's e 1's em um computador. Essa transformação da mensagem original em uma componente constituída por elementos de outro conjunto é o que chamaremos, genericamente, de codificação, e o resultante da codificação de mensagem codificada. Em geral, a codificação é construída visando eficiência tanto na compreensibilidade, ou seja, na capacidade de permitir que o receptor interprete a mensagem original a partir da mensagem codificada, quanto na compressibilidade, ou seja, na transmissão da mensagem original com o mínimo de recursos possível. Nos atentemos ao código morse como exemplo. Nele, as 26 letras do alfabeto da língua inglesa, os algarismos arábicos (0 a 9) e um conjunto de sinais (que incluem um pequeno conjunto de pontuações e sinais auxiliares, como as separações entre letras e palavras) são mapeados em sequências

singulares de pulsos elétricos e silêncios, permitindo que um receptor interprete de maneira única a mensagem codificada. Além disso, para a elaboração de tal codificação, houve um estudo prévio para associar as letras mais comumente utilizadas na língua inglesa a elementos de codificação mais curta, ou seja, a elementos que são transmitidos mais rapidamente. Para este contexto, a letra "e" possuiu a codificação mais curta dentre todos os elementos do conjunto a ser codificado. Assim, o código morse, desde sua concepção, teve em vista as qualidades de compreensibilidade e compressibilidade.

Objetivando otimizar as características das diferentes metodologias de comunicação, uma teoria com forte cerne matemático vem sendo desenvolvida. A teoria de informação é o conjunto de conhecimentos que visa, principalmente, a eficiência da transmissão e do armazenamento de dados, e cuja base possui elementos de diversos domínios como probabilidade, estatística, ciências da computação e engenharia elétrica. Em 1948, Claude E. Shannon estabeleceu em seu artigo *A Mathematical Theory of Communication* (veja [28]) o conceito de entropia da informação, que é definida no contexto de um modelo probabilístico e pode ser interpretada como a quantificação da incerteza associada a uma variável aleatória discreta. Além disso, é possível mostrar que a entropia de Shannon está relacionada à compressibilidade das mais importantes famílias de codificação binária (veja [3]).

Devido à ligação entre entropia e compressibilidade, estudos sobre a natureza estocástica das mensagens alvo da codificação tomaram forma. A intensão desses trabalhos, resumidamente, é a de tentar estabelecer relações matemáticas que expliquem a composição da informação a ser transmitida, pois, assim, a estrutura da mensagem passa a ser melhor compreendida (diminui-se a entropia) e codificações mais eficientes (mais compactas, neste caso) podem ser propostas. Uma abordagem viável e comum é a modelagem da sequência dos elementos que compõem a informação considerada por cadeias de Markov (veja discussões em [3], [25] e [28], por exemplo). Essa família de modelos pressupõe que, dada uma ordem de leitura da mensagem, o ocorrido em um passado distante não influencia o comportamento de observações futuras, e possui aplicabilidade em diversas áreas como biologia, linguística e economia.

Apesar das cadeias de Markov permitirem o delineamento de estruturas de dependência de maneira simples, sua aplicabilidade na compressão de dados pode ser

comprometida em problemas complexos. Em busca de uma maior flexibilização, em [26], Jorma Rissanen estabeleceu as bases do modelo que, posteriormente, viria a ser conhecido como cadeias de Markov de alcance variável (VLMC, do inglês *Variable Length Markov Chain*). Essa família de modelos procura uma representação mais parcimoniosa da dependência entre uma observação e seu passado. O ganho na simplicidade da expressão do processo resultou em avanços no desenvolvimento de códigos mais eficientes no quesito compressão.

Mais recentemente, Jesús E. García e Verónica A. González-López, em [8], progrediram no estudo de uma representação mais econômica da estrutura de dependência de uma cadeia de Markov, culminando na elaboração dos modelos markovianos mínimos (MMM). Essa família de modelos busca criar grupos (partes) mutuamente exclusivos (de interseção vazia) do passado relevante (espaço de estados associado à cadeia de Markov) baseados nas probabilidades de transição observadas na mensagem. Tais modelos englobam as cadeias de Markov tradicionais e as VLMC's, e, dentre os processos com estrutura markoviana, são os que possuem representação mais econômica. Em [9], é discutida a consistência da estimação dos MMM, assim como sua relação com o critério de informação bayesiano (BIC), e em [30] é apresentada uma primeira aplicação desses modelos no âmbito de compressão de dados.

Neste texto, damos continuidade ao estudo do uso dos modelos markovianos mínimos no contexto de codificação de mensagens de diversas naturezas. Em particular, desejamos delinear procedimentos de codificação de matrizes baseados nos MMM. Para isso, revisamos conceitos básicos da teoria de informação e reintroduzimos algumas definições associadas ao estudo de processos estocásticos. Detalhamos procedimentos suficientes para a transmissão de mensagens textuais, matrizes e imagens, e comparamos os resultados obtidos pela abordagem MMM com a de cadeias de Markov tradicionais.

No capítulo 1, tratamos das definições necessárias sobre processos estocásticos, em particular, processos markovianos. Reintroduzimos resultados sobre modelos markovianos mínimos (apresentados em [8] e [9]), discutimos suas vantagens inferenciais sobre as cadeias de Markov tradicionais e as VLMC's e expomos uma aplicação na modelagem do genoma da Covid-19. Além disso, apresentamos um novo processo markoviano cuja flexibilidade é responsável pela maioria das contribuições propostas por este texto: a

cadeia de Markov com interstício. Nele, é permitido que exista uma lacuna entre as observações mais relevantes do passado. Mostramos que, para este novo processo, podemos estender as definições do modelo mínimo realizando algumas adaptações simples. Também, exemplificamos como este modelo pode incrementar positivamente os resultados já obtidos para o MMM tradicional na análise do genoma da Covid-19 (veja também [11], [12], [13] e [14]). Ademais, estabelecemos com detalhes a associação entre cadeias de Markov com interstício e modelos para estruturas vicinais. No capítulo 2, apresentamos uma relação de resultados da teoria de informação que serão úteis para o entendimento deste texto. Essencialmente, desejamos compreender o significado de entropia e como podemos relacioná-la com a compressibilidade de uma mensagem. Em seguida, estudamos resultados assintóticos que servem como motivação para o processo de quantização (aproximação) de imagens construído no capítulo 4. Também delineamos um algoritmo para a obtenção de códigos de Huffman, que é uma codificação ótima (segundo os requesitos determinados nesse mesmo capítulo) e, por isso, será utilizada recorrentemente durante o texto.

O capítulo 3 apresenta com detalhes todos os estágios envolvidos na transmissão de uma mensagem, assim como os paradigmas de escolha envolvidos. Iniciando com a codificação textual, delineamos um algoritmo que utiliza os modelos markovianos mínimos para a obtenção de um sistema de compressão que, em diversos casos, apresenta desempenho superior à cadeia de Markov tradicional (sobre a relação entre códigos condicionais e modelos mínimos, veja [15]). Seguidamente, construímos um procedimento para compactação de matrizes fazendo uso do modelo mínimo para cadeias de Markov com interstício e evidenciamos que a inclusão de tal estrutura de dependência é benéfica para a compressibilidade. O capítulo 4 é uma extensa aplicação dos métodos de compressão desenvolvidos até então, mas para o contexto de transmissão de imagens. Ali tratamos da compactação de dados com perda de informação, ou seja, uma aproximação (mais simples de ser codificada) da imagem original é a que será efetivamente transmitida. Para a obtenção de tal aproximação, utilizamos conceitos de modelos de partição estabelecidos no capítulo 1 e da teoria de informação introduzidos no capítulo 2. Exibimos os benefícios da inclusão dos modelos mínimos no processo de compactação, assim como evidenciamos seus prejuízos devido a seu detalhamento e propomos estratégias para amenizá-los. Por fim, comentamos sobre técnicas utilizadas por compressores comerciais de imagens e como elas ainda podem incrementar os resultados obtidos em nossos métodos.

Capítulo 1

Modelos Markovianos de Partição

Modelos markovianos são ferramentas que há décadas recebem destaque devido a sua versatilidade. Com isso, novas abordagens e generalizações são propostas constantemente. Por exemplo, em [9], foi apresentado o conceito de modelos markovianos de partição, uma família mais geral de modelos markovianos que engloba os modelos de cadeia de Markov de alcance variável (VLMC) e as cadeias de Markov completas. Com esse enfoque, [10] deriva um método para decidir se duas amostras independentes provenientes de processos markovianos discretos são governadas pela mesma lei estocástica. Em [11], foi proposta uma nova modelagem que generaliza um pouco mais a lei que rege a markovianidade de um processo estocástico, a cadeia de Markov com interstício (*G-Markov model*). Nessa produção, foi permitido que alguma observação do passado distante fosse considerada relevante (trouxesse informação), ou seja, permite-se um interstício entre as observações mais relevantes do passado contínuo e uma observação relevante ainda mais distante. Essa generalização permitiu o acesso de informações importantes do passado sem que fosse necessário um grande aumento na ordem da cadeia de Markov, o que poderia comprometer a qualidade da estimação. Com isso, reduziu-se o número de parâmetros a serem estimados e uma simplicação no espaço de estados do processo.

Na seção 1.1 apresentamos uma breve revisão de processos markovianos e algumas classes de processos estacionários, como as tradicionais cadeias de Markov de memória finita e as cadeias de Markov de alcance variável. Na seção 1.2 introduzimos formalmente as definições e resultados associados a modelos markovianos de partição, que nos permitem enxergar os processos markovianos supracitados como casos particulares

deste último. Apresentamos também os benefícios proporcionados aos procedimentos inferenciais quando utilizamos a nova abordagem, assim como um exemplo de aplicação na modelagem do genoma da Covid-19. Na seção 1.3 definimos um processo markoviano mais geral que a cadeia de Markov tradicional: a cadeia de Markov com interstício. Essa generalização permite uma simplificação ainda maior do espaço de estados associado ao processo markoviano, permitindo, além de vantagens no âmbito de estimação, a possibilidade da inclusão de um passado distante relevante para compor a estrutura de dependência do processo estocástico, que não seria possível a partir da abordagem tradicional. Também evidenciamos como transportar os resultados obtidos para o modelo de partição para este processo mais amplo. Por fim, encerramos o capítulo mostrando como estes novos modelos markovianos podem aperfeiçoar os resultados obtidos para a modelagem do genoma da Covid-19 apresentado na seção 1.2, e também introduzimos o uso das cadeias de Markov com interstício para aplicações no escopo da modelagem de estruturas de dependência vicinal bidimensionais.

1.1 Processos Markovianos

Iniciamos esta seção com a definição de um processo estocástico, conforme apresentado na definição 1.

Definição 1. *Um processo estocástico* $\{X_t\}_{t\in\mathbb{N}}$ *é uma sequência indexada de variáveis aleatórias definidas sobre um alfabeto* A *e caracterizada por uma função de probabilidade conjunta*

$$p(x_1,\ldots,x_n) \coloneqq Pr(X_1 = x_1,\ldots,X_n = x_n),$$

em que $(x_1,\ldots,x_n) \in A^n$ *e* $n \in \mathbb{N}$.

Nota 1. *Neste texto, as variáveis aleatórias de interesse estarão sempre definidas sobre um alfabeto discreto e finito. Esse alfabeto é também conhecido como espaço amostral. Além disso, o conjunto de índices a ser considerado será sempre um subconjunto de* $\mathbb{N}$. *Dessa forma, a definição 1 é suficiente para nosso estudo.*

Notação 1. *Utilizaremos também a forma concatenada de vetores, ou seja,* $X_1^n \coloneqq X_1\ldots X_n$ *e* $x_1^n \coloneqq x_1\ldots x_n$, $x_i \in A$, *quando for conveniente. O resultado da concatenação é chamado de string.*

A estrutura de dependência de um processo estocástico $\{X_t\}_{t\in\mathbb{N}}$ pode ter inúmeras formas (veja a construção da estrutura de dependência geral de processos discretos em [16]). Ao longo deste texto, estaremos interessados em processos estacionários, cuja caracterização é apresentada na definição 2.

Definição 2. *Um processo estocástico $\{X_t\}_{t\in\mathbb{N}}$ é estacionário se a distribuição de probabilidade conjunta de qualquer subconjunto da sequência de variáveis aleatórias é invariante com respeito a deslocamentos no índice de tempo t, ou seja,*

$$Pr(X_1^n = x_1^n) \;=\; Pr(X_{1+k}^{n+k} = x_1^n),$$

para qualquer n e deslocamento $k \in \mathbb{N}$, e para todo $x_1^n \in A^n$.

Outra estrutura de dependência de interesse é a que delineia a chamada cadeia de Markov, cujo conceito é apresentado na definição 3. Nesse processo, cada variável aleatória depende apenas de um número finito (e fixo) de observações passadas e é condicionalmente independente de todo o passado residual.

Definição 3. *Um processo estocástico $\{X_t\}_{t\in\mathbb{N}}$ é uma cadeia de Markov a tempo discreto (ou seja, $t \in \mathbb{N}$) de ordem (memória) finita o, $o \geq 1$, se, para $n \in \mathbb{N}$ e $n > o$,*

$$Pr(X_n = x_n \,|\, X_1^{n-1} = x_1^{n-1}) \;=\; Pr(X_n = x_n \,|\, X_{n-o}^{n-1} = x_{n-o}^{n-1}), \tag{1.1}$$

para todo $x_1^n \in A^n$.

Notação 2. *O espaço produto $\mathcal{S} \coloneqq A^o$ será chamado de espaço de estados associado a uma cadeia de Markov de ordem o e as probabilidades condicionais $p(a|s) \coloneqq Pr(X_{o+1} = a \,|\, X_1^o = s)$ de probabilidades de transição, em que $a \in A$ e $s \in \mathcal{S}$.*

Neste texto, todas as cadeias de Markov serão consideradas estacionárias. Essa condição nos permite escrever, para $n \in \mathbb{N}$ e $n > o$,

$$\Pr(X_n = a \,|\, X_{n-o}^{n-1} = s) \;=\; \Pr(X_{o+1} = a \,|\, X_1^o = s), \quad \forall a \in A \text{ e } \forall s \in \mathcal{S}. \tag{1.2}$$

Note que, pelas equações (1.1) e (1.2), podemos escrever a probabilidade conjunta que caracteriza uma cadeia de Markov a tempo discreto de ordem $o \geq 1$, para

$n \in \mathbb{N}$ e $n > o$, para qualquer $x_1^n \in A^n$, como

$$\begin{aligned} p(x_1^n) &= p(x_n \,|\, x_1^{n-1})p(x_1^{n-1}) = p(x_n \,|\, x_{n-o}^{n-1})p(x_1^{n-1}) \\ &\vdots \\ &= p(x_1^o)p(x_{o+1} \,|\, x_1^o) \dots p(x_n \,|\, x_{n-o}^{n-1}) \\ &= p(x_1^o) \prod_{\substack{a \in A \\ s \in \mathcal{S}}} p(a \,|\, s)^{N_n(sa)}, \end{aligned} \tag{1.3}$$

em que $\mathcal{S}$ é o espaço de estados, $N_n(sa) = |\{t : o < t \leq n,\, x_{t-o}^{t-1} = s,\, x_t = a\}|$, $\forall s \in \mathcal{S}$, $\forall a \in A$, e $|\cdot|$ denota a cardinalidade do conjunto em seu argumento.

Exemplo 1. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o = 1$ definida sobre o alfabeto $A = \{a, b, c\}$, cujas probabilidades de transição são dadas por*

$$\begin{aligned} p(a \,|\, a) &= 1/4, \quad p(b \,|\, a) = 1/2, \\ p(a \,|\, b) &= 1/8, \quad p(b \,|\, b) = 3/8, \\ p(a \,|\, c) &= 1/4, \quad p(b \,|\, c) = 1/4. \end{aligned}$$

Suponha que sabemos que $X_1 = a$ e queremos calcular a probabilidade do evento $x_2^{10} = ababccbac$. Utilizando a ideia de decomposição apresentada na equação (1.3), temos

$$\begin{aligned} Pr(X_2^{10} = x_2^{10} \,|\, X_1 = a) &= Pr(X_{10} = c \,|\, X_1^9 = aababccba)Pr(X_2^9 = ababccba \,|\, X_1 = a) \\ &= Pr(X_{10} = c \,|\, X_9 = a)Pr(X_2^9 = ababccba \,|\, X_1 = a) \\ &= p(c \,|\, a)Pr(X_2^9 = ababccba \,|\, X_1 = a) \\ &\vdots \\ &= p(a \,|\, a)^1 p(b \,|\, a)^2 p(a \,|\, b)^2 p(c \,|\, b)^1 p(c \,|\, c)^1 p(b \,|\, c)^1 p(c \,|\, a)^1 = \frac{1}{2^{16}}. \end{aligned}$$

Na prática, dificilmente teremos acesso aos verdadeiros valores dos parâmetros (probabilidades) que caracterizam um processo estocástico. Dada uma amostra x_1^n do processo $\{X_t\}_{t\in\mathbb{N}}$, queremos tirar conclusões sobre os possíveis valores desconhecidos. Em nosso caso, considerando o processo $\{X_t\}_{t\in\mathbb{N}}$ como uma cadeia de Markov, temos interesse em inferir sobre as probabilidades de transição. Antes de definirmos estimadores para essas quantidades, estabelecemos o conceito de função de verossimilhança para distribuições de probabilidade discretas, apresentado na definição 4.

Definição 4. *Seja $\{X_t\}_{t\in\mathbb{N}}$ um processo estocástico definido sobre um alfabeto discreto,*

cujas distribuições que o caracterizam são indexadas pelo espaço paramétrico Θ*. Dada uma amostra* x_1^n *desse processo, definimos a função de verossimilhança como* $V_n(\theta; x_1^n) = Pr(X_1^n = x_1^n \,|\, \theta)$*, vista em função de* $\theta \in \Theta$*.*

A definição 4 indica que a verossimilhança é uma função do parâmetro θ que, para uma dada amostra x_1^n, nos entrega a plausibilidade do valor de θ para o modelo considerado. É interessante, fixado um modelo, encontrarmos qual elemento do espaço paramétrico (se existir) que maximiza a verossimilhança, uma vez que este é o que melhor suporta uma dada amostra ao modelo. Dessa forma, definimos a estimativa de máxima verossimilhança como a função da amostra que maximiza a função de verossimilhança, ou seja, fixado um modelo, a estimativa de máxima verossimilhança é o valor no espaço paramétrico Θ que torna a amostra a ser observada a mais provável possível. A definição 5 apresenta a definição formal dessa estimativa.

Definição 5. *Seja* $\{X_t\}_{t\in\mathbb{N}}$ *um processo estocástico definido sobre um alfabeto discreto, cujas distribuições que o caracterizam são indexadas pelo espaço paramétrico* Θ*, e* x_1^n *uma amostra de* $\{X_t\}_{t\in\mathbb{N}}$*. Então, a estimativa de máxima verossimilhança para* $\theta \in \Theta$ *é* $\widehat{\theta}(x_1^n) = \underset{\theta\in\Theta}{arg\,max}\, V_n(\theta; x_1^n)$*, em que* V_n *é a função de verossimilhança do modelo estabelecida na definição 4.*

Nota 2. *Observe que a estimativa de máxima verossimilhança é uma função de uma dada amostra* x_1^n*. No entanto, podemos também considerar quando essa função está definida para a amostra aleatória* X_1^n *e, de maneira análoga, podemos definir o estimador de máxima verossimilhança (EMV)* $\widehat{\theta}(X_1^n)$*, que, consequentemente, é uma variável aleatória.*

No caso de uma cadeia de Markov, o espaço paramétrico Θ é composto pelo conjunto de todas as probabilidades de transição associadas ao espaço de estados $\mathcal{S}$. O teorema 1 apresenta a forma das estimativas de máxima verossimilhança para as probabilidades de transição de uma cadeia de Markov de ordem o.

Teorema 1. *Seja* $\{X_t\}_{t\in\mathbb{N}}$ *uma cadeia de Markov a tempo discreto, de ordem finita* o*, sobre um alfabeto* A *discreto e espaço de estados* $\mathcal{S} = A^o$*. Seja* x_1^n *uma amostra desse processo. As estimativas de máxima verossimilhança para as probabilidades de transição* $p(\cdot \,|\, \cdot)$ *são da forma*

$$\widehat{p}(a|s) = \frac{N_n(sa)}{N_n(s)}, \quad \text{para } N_n(s) > 0, \tag{1.4}$$

para todo $a \in A$, $s \in \mathcal{S}$, *em que* $N_n(sa) = |\{t : o < t \leq n, x_{t-o}^{t-1} = s, x_t = a\}|$ *e* $N_n(s) = |\{t : o < t \leq n, x_{t-o}^{t-1} = s\}|$.

Demonstração. *Seja* x_1^n *uma amostra do processo* $\{X_t\}_{t\in\mathbb{N}}$. *Pela forma da probabilidade conjunta para uma cadeia de Markov, apresentada na equação (1.3), temos que a função de verossimilhança é, para* θ *como o conjunto de probabilidades de transição* $p_\theta(\cdot \mid \cdot)$,

$$V_n(\theta; x_n) = p(x_1^o) \prod_{\substack{a\in A \\ s\in\mathcal{S}}} p_\theta(a \mid s)^{N_n(sa)}, \tag{1.5}$$

em que $N_n(sa) = |\{t : o < t \leq n, x_{t-o}^{t-1} = s, x_t = a\}|$, $\forall s \in \mathcal{S}$, $\forall a \in A$.

Temos interesse em encontrar os valores de $p_\theta(\cdot \mid \cdot)$ *que maximizam* $V_n(\theta; x_n)$. *Neste caso, será mais conveniente trabalharmos com o logaritmo da verossimilhança, uma vez que seus argumentos de máximo são os mesmos, pois o logaritmo é uma função estritamente crescente. Assim, definamos* $LV_n(\theta; x_n) = \ln V_n(\theta; x_n)$. *Além disso, devemos considerar que, para cada* $s \in \mathcal{S}$, $\sum_{a\in A} p_\theta(a|s) = 1$, *ou seja, estamos em um problema de maximização com múltiplas restrições. Fazendo uso do método dos multiplicadores de Lagrange para encontrar a solução, temos interesse no funcional*

$$\begin{aligned} J(p_\theta) &= LV_n(\theta; x_n) - \sum_{s\in\mathcal{S}} \lambda_s \left(\sum_{a\in A} p_\theta(a|s) - 1 \right) \\ &= \ln p_\theta(x_1^o) + \sum_{\substack{a\in A \\ s\in\mathcal{S}}} N_n(sa) \ln p_\theta(a|s) - \sum_{s\in\mathcal{S}} \lambda_s \left(\sum_{a\in A} p_\theta(a|s) - 1 \right). \end{aligned}$$

Calculando as derivadas parciais para cada $a \in A$ *e* $s \in \mathcal{S}$ *e tomando-as como 0, obtemos*

$$\begin{aligned} \frac{\partial J(p_\theta)}{\partial p_\theta(a|s)} = \frac{N_n(sa)}{p_\theta(a|s)} - \lambda_s = 0 &\Rightarrow \\ \lambda_s = \frac{N_n(sa)}{p_\theta(a|s)} &\Rightarrow \\ p_\theta(a|s) = \frac{N_n(sa)}{\lambda_s}. & \end{aligned} \tag{1.6}$$

Substituindo (1.6) na restrição, temos

$$\sum_{a\in A} \frac{N_n(sa)}{\lambda_s} = 1 \qquad \Rightarrow$$

$$\lambda_s = \sum_{a \in A} N_n(sa) = N_n(s). \tag{1.7}$$

Substituindo a equação (1.7) em (1.6), temos que a verossimilhança é maximizada quando as probabilidades de transição são da forma

$$\widehat{p}(a|s) = \frac{N_n(sa)}{N_n(s)},$$

para todo $a \in A$, $s \in \mathcal{S}$. Este é realmente um ponto de máximo conforme pode ser verificado pelo fato da matriz hessiana associada a $LV_n(\theta; x_n)$ ser negativa definida (condição de segunda ordem).

□

A cadeia de Markov descrita na definição 3 também é chamada de cadeia de Markov completa, uma vez que seu modelo inclui as probabilidades de transição para todos os elementos de seu espaço de estados associado $\mathcal{S}$. Apesar de ser um modelo bem detalhado do ponto de vista probabilístico, pode não ser muito apropriado sob o escopo de estimação. Citando um exemplo de [2], suponha que $\{X_t\}_{t \in \mathbb{N}}$ é uma cadeia de Markov a tempo discreto, ordem o, sobre o alfabeto $A = \{a, b, c, d\}$ e espaço de estados $\mathcal{S} = A^o$. Note que o número de parâmetros a serem estimados para este modelo é uma função da ordem o. Se denotarmos o conjunto das probabilidades de transição do modelo por θ, temos que $|\theta| = |A|^o(|A| - 1)$. Se observarmos a seguinte listagem

o	$\lvert\theta\rvert$
1	12
2	48
3	192
4	768
5	3072

notamos que existe um salto muito grande na complexidade (número de parâmetros livres) dos modelos. Essa falta de maleabilidade pode provocar um superajuste (ou subajuste) do modelo aos dados. Uma flexibilização para modelos de cadeias de Markov de ordem finita é a cadeia de Markov de alcance variável (VLMC), apresentada em [2] e [26]. Essa classe de modelos permite que a memória de uma cadeia de Markov estacionária possua alcance

variável, que é uma função de observações passadas. As probabilidades de transição $\Pr(X_n = x_n \,|\, X_1 = x_1, \ldots, X_{n-1} = x_{n-1})$ são funções que dependem apenas de uma quantidade variável h de observações passadas, em que $h = h(x_1, \ldots, x_{n-1})$. Em particular, $h = h(x_1, \ldots, x_{n-1})$ é o menor inteiro tal que, para cada $a \in A$,

$$\Pr(X_n = a \,|\, X_1 = x_1, \ldots, X_{n-1} = x_{n-1}) = \Pr(X_n = a \,|\, X_{n-h} = x_{n-h}, \ldots, X_{n-1} = x_{n-1}).$$

Se $h(x_1^{n-1}) = o$, para todo $x_1^{n-1} \in A^{n-1}$, obtemos uma cadeia de Markov completa de ordem o. Se $h(x_1^{n-1})$ não for constante para todo $x_1^{n-1} \in A^{n-1}$ e $\sup h(x_1^{n-1}) = o$, então ainda temos uma cadeia de Markov de ordem o, mas com uma estrutura adicional de uma memória de alcance variável. Antes de enunciarmos a definição formal de uma cadeia de Markov de alcance variável, apresentamos, na definição 6, o conceito de contexto que é interpretado como a porção do passado que influencia a próxima observação da variável aleatória.

Definição 6. *Seja $\{X_t\}_{t\in\mathbb{N}}$ um processo estacionário assumindo valores em um alfabeto finito A. Para $n > 1$, defina h como*

$$h = h(x_1^{n-1}) = \begin{cases} 0, \quad \textit{se } Pr(X_n = a \,|\, X_1^{n-1} = x_1^{n-1}) = Pr(X_n = a),\ \forall a \in A; \\ \min\{k \in \mathbb{N} :\ Pr(X_n = a \,|\, X_1^{n-1} = x_1^{n-1}) \\ \qquad = Pr(X_n = a \,|\, X_{n-k}^{n-1} = x_{n-k}^{n-1}),\ \forall a \in A\}, \quad \textit{caso contrário.} \end{cases}$$

Seja $c : A^{n-1} \to A^{n-1}$ uma função que mapeia $x_1^{n-1} \mapsto x_{n-h}^{n-1}$. A função c é chamada de função contexto. Diremos também que o caso $h \equiv 0$ corresponde à independência.

Note que, pela definição 6, temos que o alcance do contexto da variável aleatória X_t é $h = h(x_1^{t-1}) = |c(x_1^{t-1})|$, em que $|\cdot|$ representa o comprimento de seu argumento. A definição 7 traz a definição formal de uma cadeia de Markov de alcance variável.

Definição 7. *Seja $\{X_t\}_{t\in\mathbb{N}}$ um processo estacionário assumindo valores em um alfabeto finito A e $c(\cdot)$ a sua função contexto. Seja o, $0 \leq o < \infty$ o menor inteiro tal que*

$$|c(x_1^{n-1})| = h(x_1^{n-1}) \leq o,\ \forall x_1^{n-1} \in A^{n-1}.$$

Então, $c(\cdot)$ é chamada de função contexto de ordem o, e $\{X_t\}_{t\in\mathbb{N}}$ é chamado de cadeia de

Markov de alcance variável (VLMC). A imagem da função $c(\cdot)$ é chamada de árvore de contexto e será denotada por τ.

Suponha que $\{X_t\}_{t\in\mathbb{N}}$ é uma VLMC de ordem finita $o > 1$, assumindo valores em um alfabeto finito A e $c(\cdot)$ sua função contexto. Suponha também que, para determinado $x_1^{n-1} \in A^{n-1}$, $n > o$, temos que $|c(x_1^{n-1})| = o - 1$. Por definição de VLMC de ordem o, as observações x_1^{n-o-1} são irrelevantes para a variável X_n. Então, podemos escrever

$$\begin{aligned}\Pr(X_n = x_n \,|\, X_1^{n-1} = x_1^{n-1}) &= \Pr(X_n = x_n \,|\, X_{n-o}^{n-1} = x_{n-o}^{n-1}) \\ &= \Pr(X_n = x_n \,|\, X_{n-o} = a, X_{n-o+1}^{n-1} = x_{n-o+1}^{n-1}), \quad \forall a \in A. \end{aligned} \tag{1.8}$$

A equação (1.8) evidencia que a memória variável pode ser vista como um processo de agrupamento de estados de uma cadeia de Markov completa baseado na irrelevância de observações do passado.

Exemplo 2. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o = 2$ definida sobre o alfabeto $A = \{a, b\}$, cujas probabilidades de transição são dadas por*

$$p(a|aa) = 1/4, \quad p(a|ab) = 1/2, \quad p(a|ba) = 1/4 \quad e \quad p(a|bb) = 1/3.$$

Perceba que $\{X_t\}_{t\in\mathbb{N}}$ é uma cadeia de Markov de alcance variável de ordem 2 com a seguinte função contexto, para $t > 2$,

$$c(x_1^{t-1}) = \begin{cases} a, \quad se\ x_1^{t-1} = \bullet a, & em\ que\ \bullet\ é\ uma\ concatenação\ de\ comprimento\ t-2 \\ & de\ elementos\ de\ A; \\ ab, \quad se\ x_1^{t-2} = \star ab, & em\ que\ \star\ é\ uma\ concatenação\ de\ comprimento\ t-3 \\ & de\ elementos\ de\ A; \\ bb, \quad se\ x_1^{t-2} = \star bb, & em\ que\ \star\ é\ uma\ concatenação\ de\ comprimento\ t-3 \\ & de\ elementos\ de\ A. \end{cases}$$

Note que a definição de uma cadeia de Markov de alcance variável pode ser vista como a construção de uma partição dentro do espaço de estados baseada na função contexto associada a esse processo. Em particular, a construção é feita agrupando-se passados irrelevantes. Podemos então construir outras estruturas que nos levem a diferentes partições no espaço de estados. Com essa motivação, [8] introduz o conceito de modelos

markovianos de partição. Essa metodologia é baseada no agrupamento de quaisquer estados que definem a mesma função de probabilidade de transição, o que é uma condição menos rígida que a de VLMC e que torna este último um caso particular dos modelos markovianos de partição. A seção a seguir apresenta as definições e resultados associados a essa família de modelos.

1.2 Modelos Markovianos de Partição

Conforme mencionado na seção anterior, dada uma cadeia de Markov de ordem finita o sobre um alfabeto finito A e espaço de estados $\mathcal{S} = A^o$, temos interesse em construir uma partição em $\mathcal{S}$ baseada nas probabilidades de transição do processo. A definição 8 apresenta o conceito de uma cadeia de Markov com partição $\mathcal{L}$.

Definição 8. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$. Então,*

1. *$s, r \in \mathcal{S}$ são equivalentes se $p(a|s) = p(a|r)$, $\forall a \in A$;*
2. *$\{X_t\}_{t\in\mathbb{N}}$ é uma cadeia de Markov com partição $\mathcal{L} = \{L_1, L_2, \ldots, L_{|\mathcal{L}|}\}$ no espaço de estados $\mathcal{S}$ se esta partição for a resultante da relação de equivalência definida no item anterior.*

Note que a partição induzida pela relação de equivalência apresentada no item 1 da definição 8 possui a seguinte propriedade: dois estados $s, r \in \mathcal{S}$ estão em diferentes partes se, e somente se, eles definem diferentes funções de probabilidade de transição. Ou seja, se s e r estão na mesma parte $L \in \mathcal{L}$ (e definem a mesma probabilidade de transição) não precisamos estimar individualmente $p(\cdot\,|s)$ e $p(\cdot\,|r)$, basta apenas estimarmos $p(\cdot\,|L)$, que é a função de probabilidade que utiliza como condicionamento toda a parte de $\mathcal{L}$ a que s e r pertencem. Mais formalmente, dada $\{X_t\}_{t\in\mathbb{N}}$ uma Cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$ e partição $\mathcal{L} = \{L_1, L_2, \ldots, L_{|\mathcal{L}|}\}$, podemos definir, $\forall a \in A$ e $L \in \mathcal{L}$, $p(L, a) = \sum_{s\in L} \Pr(X_{t-o}^{t-1} = s, X_t = a)$ e $p(L) = \sum_{s\in L} \Pr(X_{t-o}^{t-1} = s)$. Se $p(L) > 0$, podemos definir também $p(a|L) = \frac{p(L,a)}{p(L)}$.

Exemplo 3. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma Cadeia de Markov a tempo discreto de ordem $o = 2$ sobre o alfabeto $A = \{a, b\}$, cujas probabilidades de transição são dadas por*

$$p(a|aa) = 0,2,$$

$$p(a|ab) = 0,6,$$
$$p(a|ba) = 0,2,$$
$$p(a|bb) = 0,4.$$

Pela definição 8, essa configuração nos leva à partição $\mathcal{L} = \{L_1 = \{aa, ba\}, L_2 = \{ab\}, L_3 = \{bb\}\}$. *Observe que esta partição nos diz que, se* $X_{t-1} = a$, *o valor de* X_{t-2} *é irrelevante para a variável* X_t. *Isso equivale a um modelo VLMC com árvore de contexto* $\tau = \{T_1 = \{a\}, T_2 = \{ab\}, T_3 = \{bb\}\}$, *ou seja,* T_1 *pode ser visto como a união dos elementos de* L_1.

Lembremos que, para uma cadeia de Markov completa, temos a necessidade de estimar $|A|^o(|A|-1)$ probabilidades de transição, um número que cresce exponencialmente com a ordem o. No entanto, dada uma partição do espaço de estados, precisamos estimar apenas $|\mathcal{L}|(|A|-1)$ dessas probabilidades, ou seja, há potencialmente uma redução na complexidade do modelo. Então, o processo inferencial para essa família de modelos pode ser dividido em duas etapas: (i) delinear a partição $\mathcal{L}$ do espaço de estados; e (ii) estimar as probabilidades de transição $p(a|L)$, $\forall a \in A$. Temos interesse, portanto, em desenvolver uma maneira consistente de obtenção da partição $\mathcal{L}$.

Seja x_1^n uma amostra da cadeia de Markov $\{X_t\}_{t\in\mathbb{N}}$ a tempo discreto de ordem o sobre um alfabeto finito A, espaço de estados $\mathcal{S}$ e partição $\mathcal{L}$, com $n > o$. Na equação (1.3), vimos que a probabilidade conjunta para uma amostra é

$$p(x_1^n) = p(x_1^o) \prod_{\substack{a\in A \\ s\in\mathcal{S}}} p(a \mid s)^{N_n(sa)},$$

onde $N_n(sa)$ é o número de vezes em que a concatenação sa, $a \in A$ e $s \in \mathcal{S}$, esteve presente em x_1^n. Definamos duas quantidades: o número de ocorrências de elementos de L que são seguidos por $a \in A$, $N_n(L, a)$; e o número total de ocorrências de *strings* pertencentes a L, $N_n(L)$. Formalmente,

$$N_n(L, a) = \sum_{s\in L} N_n(sa);$$
$$N_n(L) = \sum_{s\in L} N_n(s).$$

Dessa forma, podemos reescrever a equação (1.3) como

$$p(x_1^n) = p(x_1^o) \prod_{\substack{a\in\mathcal{A}\\ L\in\mathcal{L}}} p(a \mid L)^{N_n(L,a)}. \tag{1.9}$$

Repare, pela equação (1.9), que a função de probabilidade conjunta passa a depender dos elementos da partição e não mais dos estados individualmente. Com isso, análogo ao que foi feito para cadeias de Markov completas, vamos construir a função de verossimilhança do processo, mas agora em função da partição $\mathcal{L}$, ou seja,

$$V_n(\mathcal{L}; x_1^n) = p(x_1^o) \prod_{\substack{a\in\mathcal{A}\\ L\in\mathcal{L}}} p(a \mid L)^{N_n(L,a)}.$$

Como nosso interesse é maximizar a função de verossimilhança em $\mathcal{L}$, note que, independentemente da partição escolhida, os valores das probabilidades condicionais que maximizam o produtório são da forma $\widehat{p}(a \mid L) = \frac{N_n(L,a)}{N_n(L)}$, conforme o argumento utilizado no teorema 1. Além disso, o termo $p(x_1^o)$ também independe da partição e não precisamos incluí-lo no processo de busca de $\mathcal{L}$. Assim, podemos considerar apenas a parte conveniente da verossimilhança e definir uma função de verossimilhança modificada, VM, da forma

$$VM(\mathcal{L}; x_1^n) = \prod_{\substack{a\in\mathcal{A}\\ L\in\mathcal{L}}} \left(\frac{N_n(L,a)}{N_n(L)}\right)^{N_n(L,a)}, \quad \text{com } N_n(L) \neq 0, L \in \mathcal{L}. \tag{1.10}$$

Na seção anterior, argumentamos que um dos objetivos de se desenvolver modelos alternativos às cadeias de Markov completas era evitar um sobreajuste (número excessivo de parâmetros). De fato, quando estamos ajustando um modelo, a simples atitude de incluir parâmetros (o que tende a interpolar a amostra observada) pode causar a inflação da função de verossimilhança. Dessa forma, é interessante que tenhamos um critério de bondade de ajuste que leve em consideração tanto a informação contida nos dados quanto o número de parâmetros utilizados no modelo para descrevê-los. Para esse fim, vários critérios foram desenvolvidos, como AIC, BIC e DIC. Aqui, iremos nos concentrar no critério de informação bayesiano (BIC), que é apresentado na definição 9, para um modelo markoviano de partição. A escolha do BIC como ferramenta de comparação de modelos markovianos se deve a suas propriedades de consistência, como as exploradas em [4], [5] e

[6].

Definição 9. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$ e partição $\mathcal{L}$. Dada uma amostra x_1^n do processo, o BIC do modelo dado pelo item 2 da definição 8, e de acordo com a função de verossimilhança modificada dada pela equação (1.10), é*

$$BIC(\mathcal{L}; x_1^n) = \ln\left[VM(\mathcal{L}; x_1^n)\right] - \frac{(|A|-1)|\mathcal{L}|}{2}\ln(n). \tag{1.11}$$

Nosso objetivo é encontrar a partição $\tilde{\mathcal{L}}$ dentro do espaço de partições, $\mathcal{P}$, do espaço de estados, $\mathcal{S}$, que maximize o BIC dada uma amostra. Antes de apresentarmos os resultados que nos auxiliam nessa busca, precisamos introduzir alguns conceitos e notações preliminares.

Definição 10. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$ e partição $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$. Então,*

1. *$L \in \mathcal{L}$ é uma parte boa de $\mathcal{L}$ se, $\forall r, s \in L$, $Pr(X_t = \cdot\,|X_{t-o}^{t-1} = s) = Pr(X_t = \cdot\,|X_{t-o}^{t-1} = r)$, para $t > o$;*

2. *$\mathcal{L}$ é uma partição boa de $\mathcal{S}$ se L_i verifica o item anterior, para todo $i \in \{1, \ldots, |\mathcal{L}|\}$.*

Notação 3. *Seja $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$ uma partição de $\mathcal{S}$. Então,*

1. *denotemos por $\mathcal{L}^{ij}$ a partição $\mathcal{L}^{ij} = \{L_1, \ldots, L_{i-1}, L_{ij}, L_{i+1}, \ldots, L_{j-1}, L_{j+1}, \ldots, L_{|\mathcal{L}|}\}$, em que $L_{ij} = L_i \cup L_j$, para $1 \leq i < j \leq |\mathcal{L}|$;*

2. *para $a \in A$, definamos $p(L_{ij}, a) = p(L_i, a) + p(L_j, a)$, $p(L_{ij}) = p(L_i) + p(L_j)$, $N_n(L_{ij}, a) = N_n(L_i, a) + N_n(L_j, a)$ e $N_n(L_{ij}) = N_n(L_i) + N_n(L_j)$.*

Note que, a princípio, uma partição boa não precisa necessariamente satisfazer a condição 2 da definição 8 (a qual será chamada de partição boa mínima). Observe também que, trivialmente, o espaço de estados $\mathcal{S}$ é uma partição boa de si mesmo. Além disso, se $\mathcal{L}$ é uma partição boa de $\mathcal{S}$ e $L_i, L_j \in \mathcal{L}$ definem a mesma lei de probabilidade condicional, $\mathcal{L}^{ij}$ também é uma partição boa. Apesar de parecerem duas informações simplórias, elas são fundamentais para a construção de algoritmos recursivos que geram, a partir de uma partição boa inicial, partições aptas a satisfazer a condição 2 da definição 8.

Apresentaremos os resultados necessários para justificar a consistência de um algoritmo em particular que visa a obtenção da almejada partição. Esses teoremas foram demonstrados com detalhes em [9] e suas provas serão omitidas neste texto. Tendo em vista a obtenção da partição boa mínima, primeiramente, gostaríamos de construir um critério que nos diga quando duas partes boas de uma partição devem ser unidas. Com este intuito, foi desenvolvido o teorema 2.

Teorema 2. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$ e partição $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$. Seja x_1^n uma amostra do processo. Suponha que há i e j, com $i \neq j$, tais que L_i e L_j são partes boas. Então, $p(a|L_i) = p(a|L_j)$, $\forall a \in A$, se, e somente se, eventualmente quase certamente, quando $n \to \infty$, $BIC(\mathcal{L}^{ij}; x_1^n) > BIC(\mathcal{L}; x_1^n)$.*

Corolário 1. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$ e partição $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$. Seja x_1^n uma amostra do processo. Suponha que a partição $\mathcal{L}$ possui K partes boas, denotadas por $\{L_{i_j}\}_{j=1}^K$, e seja T um conjunto de índices com $T \subseteq \{1, \ldots, K\}$. Então, $p(a|L_{i_j}) = p(a|L_{i_l})$, $\forall a \in A$, $\forall j, l \in T$, se, e somente se, eventualmente quase certamente, quando $n \to \infty$, $BIC(\mathcal{L}^T; x_1^n) > BIC(\mathcal{L}; x_1^n)$, em que $\mathcal{L}^T$ denota a partição que une as partes boas L_{i_j}, com $j \in T$, em $L_T = \bigcup_{j\in T} L_{i_j}$.*

O corolário 1 nos entrega um critério que decide se mais de duas partes boas devem ser colocadas em uma mesma partição. Um outro ponto de vista para a construção da partição boa mínima seria a partir de uma métrica definida dentro de uma partição, que une partes cuja distância esteja abaixo de um limiar e as mantém separadas se estiver acima. A definição 11 apresenta a função $d_{\mathcal{L}}$ definida para uma partição $\mathcal{L}$ e o teorema 3 elucida que esta função é uma métrica em $\mathcal{L}$.

Definição 11. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$ e $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$ uma partição boa de $\mathcal{S}$. Seja x_1^n, $n > o$, uma amostra do processo. Defina*

$$d_{\mathcal{L}}(i,j) = \frac{2}{(|A|-1)\ln(n)} \sum_{a\in A} \left\{ N_n(L_i,a) \ln\left(\frac{N_n(L_i,a)}{N_n(L_i)}\right) + N_n(L_j,a) \ln\left(\frac{N_n(L_j,a)}{N_n(L_j)}\right) - N_n(L_{ij},a) \ln\left(\frac{N_n(L_{ij},a)}{N_n(L_{ij})}\right) \right\}.$$

Teorema 3. *Sob as condições da definição 11, para cada n, e para quaisquer $i, j, k \in \{1, 2, \ldots |\mathcal{L}|\}$,*

1. $d_{\mathcal{L}}(i, j) \geq 0$ *com igualdade se, e somente se,* $\frac{N_n(L_i,a)}{N_n(L_i)} = \frac{N_n(L_j,a)}{N_n(L_j)}$, $\forall a \in A$;
2. $d_{\mathcal{L}}(i, j) = d_{\mathcal{L}}(j, i)$;
3. $d_{\mathcal{L}}(i, k) \leq d_{\mathcal{L}}(i, j) + d_{\mathcal{L}}(j, k)$.

Logo, $d_{\mathcal{L}}$ é uma métrica em $\mathcal{L}$.

Munidos da definição 11 e do teorema 3, temos a ideia que duas partes próximas, segundo a distância $d_{\mathcal{L}}$, são aquelas que possuem as estimativas das probabilidades condicionais suficientemente próximas, ou seja, $\frac{N_n(L_i,a)}{N_n(L_i)} \approx \frac{N_n(L_j,a)}{N_n(L_j)}$, $\forall a \in A$. O corolário 2 apresenta como deve ser esse "suficientemente próximo" para que a distância $d_{\mathcal{L}}$ possa se relacionar com o critério BIC apresentado no teorema 2.

Corolário 2. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$ e partição $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$. Seja x_1^n uma amostra do processo. Suponha que há i e j, com $i \neq j$, tais que L_i e L_j são partes boas. Então,*

$$BIC(\mathcal{L}^{ij}; x_1^n) > BIC(\mathcal{L}; x_1^n) \iff d_{\mathcal{L}}(i, j) < 1.$$

Antes de prosseguirmos, devemos nos questionar: quando consideramos uma amostra suficientemente grande, a partição $\mathcal{L}$ que maximiza o BIC dentro do espaço de todas as partições é a partição boa mínima? A resposta é sim, e esse resultado é declarado de maneira formal no teorema 4.

Teorema 4. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$. Seja x_1^n uma amostra do processo. Denote por $\mathcal{P}$ o conjunto de todas as partições de $\mathcal{S}$. Defina*

$$\widehat{\mathcal{L}}_n = \underset{\mathcal{L}\in\mathcal{P}}{arg\,max} \{BIC(\mathcal{L}; x_1^n)\}. \tag{1.12}$$

Então, quase certamente eventualmente, quando $n \to \infty$, $\widehat{\mathcal{L}}_n = \tilde{\mathcal{L}}$, em que $\tilde{\mathcal{L}}$ é a partição boa mínima de $\mathcal{S}$, conforme o item 2 da definição 8.

Dessa forma, basta possuirmos um método que nos possibilite encontrar a partição descrita pela equação (1.12) que alcançaremos de maneira consistente a partição boa mínima desejada. Para este intuito, apresentamos o algoritmo A.1 (apêndice A), introduzido por [8].

Para uma amostra suficientemente grande, temos que o algoritmo A.1 retorna de maneira consistente a partição boa mínima, conforme é estabelecido pelo corolário 3.

Corolário 3. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$. Seja x_1^n uma amostra do processo. A partição $\widehat{\mathcal{L}}_n$ obtida pelo algoritmo A.1 converge quase certamente eventualmente para $\tilde{\mathcal{L}}$, em que $\tilde{\mathcal{L}}$ é a partição boa mínima.*

Nota 3. *Neste texto, se nada for preestabelecido, utilizaremos o espaço de estados completo $\mathcal{S}$ como a partição boa inicial para o algoritmo A.1.*

Exemplo 4. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o = 3$ assumindo valores no alfabeto $A = \{a, b, c\}$, com partição do espaço de estados $\mathcal{L} = \{L_1, L_2, L_3\}$ com configuração e probabilidades de transição dadas pela tabela 1.1.*

Parte	Elementos	$p(a\|L)$	$p(b\|L)$
L_1	$aac, bab, abc, aca, bcb, ccc, aaa, bac, abb, acb, bcc, cca$	0,2	0,3
L_2	$bbc, cbc, aab, baa, aba, acc, bca, ccb, cac$	0,4	0,3
L_3	$caa, cab, bba, cbb, bbb, cba$	0,4	0,1

Tabela 1.1: Partição $\mathcal{L}$ e probabilidades de transição para cada elemento de A.

Simulamos uma amostra de tamanho $n = 5 \times 10^3$ do processo descrito acima e aplicamos o algoritmo A.1 para a recuperação da partição original. O procedimento, de fato, retornou a partição almejada. A fim de visualizarmos a eficiência do algoritmo, repetimos a simulação 100 vezes para três diferentes tamanhos de amostra. A tabela 1.2 apresenta a contagem do número de partes nas partições obtidas pelo algoritmo A.1 em 100 simulações com os tamanhos de amostra (n) 5×10^3, 10^4 e 5×10^4. Para o caso $n = 5 \times 10^3$, houve 11 casos em que três partes foram identificadas, mas estas não condiziam completamente com as definidas na tabela 1.1. Isso não ocorreu para os demais tamanhos amostrais, evidenciando a consistência do algoritmo.

n	3 partes	4 partes
5×10^3	88	12
10^4	94	6
5×10^4	100	0

Tabela 1.2: Número de partes das partições estimadas em 100 simulações, cada uma de tamanho n.

1.2.1 Modelos Markovianos para DNA de Covid-19

Abrimos um breve espaço para ilustrarmos como os modelos de partição nos trazem vantagens práticas. Neste tópico, temos interesse em estudar o comportamento estocástico de sequenciamentos completos de genomas provenientes do coronavírus Covid-19 em seres humanos. Esse vírus está associado ao surto de doenças respiratórias que teve início em Wuhan, da província de Hubei, na China, e se espalhou por todo o mundo. A sequência de DNA utilizada foi extraída de um paciente do sexo masculino, de 41 anos, proveniente de Wuhan e que não possuía histórico de tuberculose, diabetes ou hepatite. O indivíduo foi recebido e hospitalizado no Hospital Central de Wuhan no dia 26 de dezembro de 2019, seis dias após o início dos sintomas, os quais foram febre, fraqueza, tosse, dores no corpo, em particular, no peito. Informações completas sobre o caso podem ser consultadas em [29] e o sequenciamento completo do genoma que estudaremos pode ser obtido em https://www.ncbi.nlm.nih.gov/nuccore/MN908947. Em nossa análise, os dados são compostos por 29.903 observações assumindo valores no conjunto de bases $A = \{a, c, g, t\}$.

Iremos iniciar modelando o sequenciamento por uma cadeia de Markov completa. Para isso, precisamos estabelecer qual será a memória o escolhida. Em um processo markoviano discreto com alfabeto discreto e finito usamos o critério $o < \lfloor \log_{|A|}(n) \rfloor - 1$, em que $\lfloor x \rfloor$ é o maior inteiro menor ou igual a x. Logo, para nosso caso, $o < 6$. Temos, então, cinco possibilidades para o valor da ordem da cadeia e precisamos decidir qual será utilizada. Uma opção é estabelecer um critério e estimar um modelo para cada memória e verificar qual deles otimiza o critério escolhido. Novamente, utilizaremos o BIC para a comparação de modelos. A tabela 1.3 apresenta os valores observados do máximo do logaritmo da verossimilhança e o BIC associados aos modelos com diferentes memórias o.

Podemos notar, pela tabela 1.3, que o máximo do logaritmo da verossimilhança do modelo cresce conforme aumentamos o valor da memória o, mas o BIC nos indica que esse aumento não justifica a inclusão do número de parâmetros necessários para a

o	$\max_{\theta\in\Theta} \ln(V(\theta, x_1^n))$	BIC
1	-39.903,25	-39.965,08
2	-39.756,11	-40.003,45
3	-39.548,82	-40.538,17
4	-39.176,91	-43.134,31
5	-37.880,74	-53.710,32

Tabela 1.3: Valores do logaritmo da máxima verossimilhança e do BIC associados aos modelos de cadeias de Markov completas com diferentes memórias o.

construção de um modelo com ordem elevada. Ou seja, modelos que utilizam memória longa estão inflando nosso espaço paramétrico. Para tentar incluir o máximo de informação do nosso processo sem que os modelos sejam muito penalizados pelo grande número de parâmetros, utilizamos a abordagem de modelos markovianos de partição. A tabela 1.4 apresenta os valores observados do número de partes e do BIC associados aos modelos com diferentes memórias o.

o	$\lvert\mathcal{L}\rvert$	BIC
1	4	-39.965,08
2	7	-39.925,36
3	9	-39.832,03
4	15	-39.661,36
5	30	-38.848,28

Tabela 1.4: Valores do número de partes e do BIC associados aos modelos markovianos de partição com diferentes memórias o.

Podemos notar, pela tabela 1.4, que, para toda ordem o, o BIC para o modelo de partição é superior ao da cadeia completa. Além disso, para as memória apresentadas, o BIC foi crescente em o, ou seja, a expansão realizada no número de parâmetros foi benéfica para o nosso modelo. Retornaremos a este exemplo na seção 1.3.

1.3 Cadeias de Markov com Interstício

Imaginemos que estamos observando um processo estocástico $\{X_t\}_{t\in\mathbb{N}}$ e que julgamos uma modelagem via cadeia de Markov adequada para o fenômeno estudado. Suponha que temos o conhecimento que informações de um passado distante sejam relevantes e que informações de um passado mais recente sejam irrelevantes para uma futura observação. Ou seja, existe um interstício (um "vácuo") que separa as observações relevantes do passado. Fenômenos sazonais tendem a se comportar dessa maneira, como

os relacionados ao mercado financeiro, a produções agrícolas, ao clima etc. Dessa forma, precisamos considerar uma memória grande para modelar toda a informação útil no processo? Não estaremos carregando um número desnecessário de parâmetros e dificultando a interpretabilidade do modelo? Com essa motivação, [11] introduziu o conceito de cadeias de Markov com interstício (*G-models*) que permitem a existência de uma observação relevante anterior a uma sequência de observações passadas irrelevantes. Neste trabalho, apresentamos uma extensão da cadeia de Markov com interstício introduzindo um novo parâmetro que permitirá contemplar observações prévias ao interstício. Dessa maneira, considere $M, G, g \in \mathbb{N}$, com $G > M$ finitos e A um alfabeto também finito. Suponha $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o = G + g$, sobre o alfabeto A, com espaço de estados $\mathcal{S} = A^{G+g}$. A ideia geral da cadeia de Markov com interstício é a possibilidade de supor que, além do passado anterior ao tempo $t - o$, as observações nos tempos $t - G + 1$ a $t - M - 1$ também são irrelevantes para o tempo t. Essa generalização torna as cadeias de Markov tradicionais um caso particular e nos permite trabalhar com modelos mais flexíveis. A definição 12 formaliza este novo processo.

Definição 12. *Sejam $M, G, g \in \mathbb{N}$, com $G > M$ finitos e A um alfabeto também finito. O processo estocástico $\{X_t\}_{t\in\mathbb{N}}$ é uma cadeia de Markov com interstício a tempo discreto de ordem imediata M, interstício G, ordem de interstício g, sobre um alfabeto A, com espaço de estados $\mathcal{S}^* = A^{g+1} \times A^M$, se*

$$Pr(X_t = a|X_1^{t-1} = \bullet\, y \star s) = Pr(X_t = a|X_{t-G-g}^{t-G} = y, X_{t-M}^{t-1} = s), \tag{1.13}$$

em que $a \in A$, $y \in A^{g+1}$, $s \in A^M$, e "$\bullet$" e "$\star$" são concatenações quaisquer de tamanhos $t - G - g - 1$ e $G - M - 1$ de elementos de A, respectivamente.

Nota 4. *Ao longo deste texto, também nos referiremos a $\mathcal{S}^*$ como espaço de estados estendido.*

Como mencionado anteriormente, as cadeias de Markov tradicionais formam uma subcategoria das cadeias de Markov com interstício, pois uma cadeia de Markov de ordem o equivale a uma cadeia de Markov com interstício com $M = o - 1$, $G = o$ e $g = 0$, por exemplo. Apesar de ser mais flexível, essa maior família de modelos também carrega os problemas de parametrização das cadeias de Markov tradicionais: pequenos aumentos no espaço de estados podem inflacionar o espaço paramétrico. Para contornar

esse obstáculo, desejamos estender o conceito de modelos de partição para cadeias de Markov com interstício e verificar se os resultados de consistência apresentados em [9] (enunciados aqui como os teoremas 2, 3 e 4 e corolários 1, 2 e 3) permanecem válidos. A definição 13 apresenta a definição de uma cadeia de Markov com interstício com partição $\mathcal{L}$.

Definição 13. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov com interstício a tempo discreto de ordem direta $M < \infty$, interstício G, ordem de interstício g, sobre um alfabeto finito A, com espaço de estados $\mathcal{S}^* = A^{g+1} \times A^M$. Então,*

1. *$s, r \in \mathcal{S}^*$ são equivalentes se $p(a|s) = p(a|r)$, $\forall a \in A$;*
2. *$\{X_t\}_{t\in\mathbb{N}}$ é uma cadeia de Markov com interstício com partição $\mathcal{L} = \{L_1, L_2, \ldots, L_{|\mathcal{L}|}\}$ no espaço de estados $\mathcal{S}^*$ se esta partição for a resultante da relação de equivalência definida no item anterior.*

Nota 5. *A partição estabelecida no item 2 da definição 13 será chamada de partição boa mínima. Além disso, a adaptação do conceito de parte boa apresentado na definição 10 segue de maneira análoga.*

Os resultados de consistência da obtenção da partição boa mínima para o caso de uma cadeia de Markov tradicional eram baseados essencialmente na forma do BIC do modelo, e este, por sua vez, tem sua forma herdada da função de verossimilhança modificada VM. Para visualizarmos a função de verossimilhança para nosso novo modelo, suponha $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov com interstício a tempo discreto de ordem direta $M < \infty$, interstício G, ordem de interstício g, sobre um alfabeto finito A, com espaço de estados $\mathcal{S}^* = A^{g+1} \times A^M$, e x_1^n uma amostra desse processo. Podemos escrever a função de probabilidade conjunta da amostra, denotada $p(x_1^n) = \Pr(X_1^n = x_1^n)$, como

$$\begin{aligned} p(x_1^n) &= p(x_1^{G+g}) p(X_{G+g+1}^n = x_{G+g+1}^n \,|\, X_1 = x_1, \ldots, X_{G+g} = x_{G+g}) \\ &= p(x_1^{G+g}) \prod_{k=G+g+1}^{n} \Pr(X_k = x_k \,|\, X_{k-G-g}^{k-G} = x_{k-G-g}^{k-G}, X_{k-M}^{k-1} = x_{k-M}^{k-1}) \\ &= p(x_1^{G+g}) \prod_{a\in A, s\in\mathcal{S}^*} p(a|s)^{N_n(sa)}, \end{aligned}$$

utilizando a mesma ideia de decomposição das probabilidades condicionais aplicada nas equações (1.3). Note que, com excessão do termo marginal $p(x_1^{G+g})$ (que não nos interessa),

a função de probabilidade conjunta para uma cadeia de Markov com interstício é a mesma que para uma cadeia de Markov tradicional, alterando apenas o espaço de estados sobre o qual o processo está definido (agora o condicionamento ocorre sobre o espaço de estados estendido $\mathcal{S}^*$). Dessa forma, seguindo os mesmos passos descritos na seção anterior, se o espaço de estados estendido possui partição $\mathcal{L} = \{L_1, L_2, \ldots, L_{|\mathcal{L}|}\}$, temos que

$$p(x_1^n) = p(x_1^{G+g}) \prod_{a \in A, L \in \mathcal{L}} p(a|L)^{N_n(L,a)}, \tag{1.14}$$

em que $N_n(L,a) = \sum_{s \in L} N_n(sa)$. Novamente, nosso interesse é maximizar a função de verossimilhança em $\mathcal{L}$. Baseado no teorema 1 e no argumentado na seção anterior, temos que a função de verossimilhança modificada, VM, para este modelo é

$$VM(\mathcal{L}; x_1^n) = \prod_{a \in A, L \in \mathcal{L}} \left(\frac{N_n(L,a)}{N_n(L)} \right)^{N_n(L,a)}, \text{ com } N_n(L) \neq 0, L \in \mathcal{L}, \tag{1.15}$$

em que $N_n(L) = \sum_{s \in L} N_n(s)$ e $N_n(L,a) = \sum_{s \in L} N_n(sa)$. Esta função de verossimilhança modificada é então utilizada para a definição do BIC para este modelo, conforme apresentada na definição 14.

Definição 14. *Seja $\{X_t\}_{t \in \mathbb{N}}$ uma cadeia de Markov com interstício a tempo discreto de ordem direta $M < \infty$, interstício G, ordem de interstício g, sobre um alfabeto finito A, com espaço de estados $\mathcal{S}^* = A^{g+1} \times A^M$ e partição $\mathcal{L}$. Dada uma amostra x_1^n do processo, o BIC do modelo dado pelo item 2 da definição 13, e de acordo com a função de verossimilhança modificada dada pela equação (1.15), é*

$$BIC(\mathcal{L}; x_1^n) = \ln\left[VM(\mathcal{L}; x_1^n)\right] - \frac{(|A|-1)|\mathcal{L}|}{2} \ln(n). \tag{1.16}$$

Perceba que os BIC's para uma cadeia de Markov com interstício e para uma cadeia de Markov tradicional possuem exatamente a mesma forma, diferindo apenas nos espaços de estados sobre os quais os processos estão definidos, conforme podemos observar pelas equações (1.11) e (1.16). Essa estabilidade na construção do BIC é a chave para que os resultados de consistência para o modelo markoviano de partição tradicional possam ser estendidos para o novo processo com interstício.

Apesar de todas as demonstrações realizadas em [9] serem as mesmas para

cadeias de Markov com interstício, gostaríamos de ilustrar, a partir de um exemplo simples, porque continuam válidas. Suponha que $\{X_t\}_{t\in\mathbb{N}}$ é uma cadeia de Markov com interstício com $M = 1$, $G = 3$ e $g = 1$ sobre o alfabeto $A = \{a, b, c\}$, com espaço de estados $S^* = A^2 \times A$ e partição $\mathcal{L}^*$. Suponha $L_1^* = \{abb\}$ e $L_2^* = \{ccc\}$ duas partes boas de $\mathcal{L}^*$. Perceba que isso quer dizer que

$$\begin{aligned} p_X(\cdot\,|L_1^*) &= \Pr(X_t = \cdot\,|\,X_{t-4} = a, X_{t-3} = b, X_{t-1} = b) \\ &= \Pr(X_t = \cdot\,|\,X_{t-4} = a, X_{t-3} = b, X_{t-2} = \star, X_{t-1} = b), \end{aligned} \tag{1.17}$$

em que "$\star$" é um elemento qualquer de A. A mesma ideia vale para $p_X(\cdot\,|L_2^*)$. Suponha agora $\{Y_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov tradicional de ordem $o = 4$ sobre o alfabeto $A = \{a, b, c\}$, com espaço de estados $S = A^4$ e partição $\mathcal{L}$. Considere que a partição possui uma configuração tal que existam duas partes boas $L_1 = \{abab, abbb, abcb\}$ e $L_2 = \{ccac, ccbc, cccc\}$. Note que, por definição, $p_Y(\cdot\,|L_1)$ obedece todas as equações de (1.17), ou seja, $p_Y(\cdot\,|L_1)$ e $p_X(\cdot\,|L_1^*)$ definem a mesma função de probabilidade. Logo, a partir de $\mathcal{L}^*$ podemos mapear uma partição conveniente $\mathcal{L}$ que transporte a estrutura de dependência do espaço de estados estendido do processo $\{X_t\}_{t\in\mathbb{N}}$ para o espaço de estados do processo $\{Y_t\}_{t\in\mathbb{N}}$. E, como L_1 e L_2 são partes boas em $\mathcal{L}$, podemos aplicar, por exemplo, o teorema 2 para decidir se essas partes devem ser unidas. Se forem unidas, teremos uma nova parte $L_{1,2} = L_1 \cup L_2$ e a propriedade $p_Y(\cdot\,|L_1) = p_Y(\cdot\,|L_2)$, que implica $p_X(\cdot\,|L_1^*) = p_X(\cdot\,|L_2^*)$. Ou seja, L_1^* e L_2^* também devem ser unidas em uma única parte em $\mathcal{L}^*$. Esta ilustração traz a útil ideia de partes equivalentes em diferentes processos, desde que ambos cubram o mesmo alcance de observações relevantes do passado. Utilizaremos isso na comparação de resultados obtidos para modelos de partição tradicionais e com interstício.

Agora, iremos apenas reintroduzir alguns conceitos essenciais da seção anterior, mas adaptados para os modelos com interstício.

Definição 15. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov com interstício a tempo discreto de ordem direta $M < \infty$, interstício G, ordem de interstício g, sobre um alfabeto finito A, com espaço de estados $\mathcal{S}^* = A^{g+1} \times A^M$ e $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$ uma partição boa de $\mathcal{S}^*$. Seja x_1^n uma amostra do processo. Defina*

$$d_{\mathcal{L}}(i,j) = \frac{2}{(|A|-1)\ln(n)} \sum_{a\in A} \left\{ N_n(L_i, a) \ln\left(\frac{N_n(L_i,a)}{N_n(L_i)}\right) + N_n(L_j, a) \ln\left(\frac{N_n(L_j,a)}{N_n(L_j)}\right) \right.$$

$$- N_n(L_{ij}, a) \ln \left(\frac{N_n(L_{ij}, a)}{N_n(L_{ij})} \right) \Bigg\}.$$

Corolário 4. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov com interstício a tempo discreto de ordem direta $M < \infty$, interstício G, ordem de interstício g, sobre um alfabeto finito A, com espaço de estados $\mathcal{S}^* = A^{g+1} \times A^M$. Seja x_1^n uma amostra do processo. A partição $\widehat{\mathcal{L}}_n$ obtida pelo algoritmo A.1, adaptado para a distância $d_{\mathcal{L}}$ apresentada na definição 15, converge quase certamente eventualmente para $\tilde{\mathcal{L}}$, em que $\tilde{\mathcal{L}}$ é a partição boa mínima do espaço de estados $\mathcal{S}^*$.*

O modelo que visa utilizar a partição boa mínima será chamado de "modelo markoviano mínimo com interstício" (doravante chamado G3M).

Notação 4. *Um modelo markoviano mínimo com interstício de ordem direta M, interstício G e ordem de interstício g é denotado por G3M(g, G, M).*

Exemplo 5. *Considere uma cadeia de Markov com interstício com $A = \{a, b, c\}$, $M = 1$, $G = 3$, $g = 1$ e partição $\mathcal{L} = \{L_1, L_2, L_3\}$ com configuração e probabilidades de transição dadas pela tabela 1.5. Note que, por exemplo, $p(a|L_1) = 0,2$ implica $Pr(X_t = a|X_{t-4}^{t-1} = aa \star a) = 0,2$, $\forall \star \in A$, ou seja, a informação sobre o valor na posição $t-2$ é irrelevante para a observação no tempo t.*

Parte	Elementos	$p(a\|L)$	$p(b\|L)$
L_1	$aac, bab, abc, aca, bcb, ccc, aaa, bac, abb, acb, bcc, cca$	0,2	0,3
L_2	$bbc, cbc, aab, baa, aba, acc, bca, ccb, cac$	0,4	0,3
L_3	$caa, cab, bba, cbb, bbb, cba$	0,4	0,1

Tabela 1.5: Partição $\mathcal{L}$ e probabilidade de transição para cada elemento de A.

Simulamos uma amostra de tamanho $n = 5 \times 10^4$ do processo descrito acima e aplicamos o algoritmo A.1 em duas configurações: (i) utilizando $o = 4$, espaço de estados $\mathcal{S} = A^4$ e $d_{\mathcal{L}}$ como estabelecido na definição 11; e (ii) utilizando $M = 1$, $G = 3$, $g = 1$, espaço de estados $\mathcal{S}^* = A^2 \times A$ e $d_{\mathcal{L}}$ como na definição 15. Note que ambos os modelos incluem o passado relevante proposto na definição do processo, mas a configuração (ii) proporciona um espaço de estados menor e, consequentemente, um modelo com menor complexidade que (i). Com a amostra simulada, obtivemos que (ii) recuperou a partição original proposta na tabela 1.5. No entanto, para (i), obtivemos a partição apresentada na tabela 1.6.

Parte	Elementos
L_1	$aaaa, acaa, ccaa, acba, aaca, acca, ccca, baab, abab, acab, babb, abbb, acbb,$ $bcbb, bacb, accb, aaac, baac, abac, bcac, ccac, abbc, ccbc, aacc, abcc, bccc$
L_2	$baaa, abaa, bcaa, baba, abba, bcba, baca, abca, bcca, aaab, ccab, aabb, ccbb, aacb,$ $cccb, caac, bbac, cbac, acac, cabc, bbbc, cbbc, acbc, cacc, bbcc, cbcc, accc$
L_3	$caaa, bbaa, cbaa, caba, bbba, cbba, caca, bbca, cbca, caab, bbab, cbab, cabb,$ $bbbb, cbbb, cacb, bbcb, cbcb$
L_4	$aaba, ccba, bcab, abcb, bccb, aabc, babc, bcbc, bacc, cccc$

Tabela 1.6: Partição estimada para o espaço de estados $\{a, b, c\}^4$ - procedimento (i).

Note que, sob a configuração (i), cada elemento do espaço de estados é uma concatenação de $o = 4$ elementos de A, por exemplo, o elemento abcb. Esse elemento sob a configuração (ii) é denotado por abb, pois sob $M = 1$, $G = 3$ e $g = 1$, a observação na posição $t - 2$ é irrelevante. Também, observe que, apesar de quatro partes serem obtidas, as partes L_2 e L_3 da tabela 1.6 são equivalentes a L_2 e L_3 propostas na tabela 1.5, mas as partes L_1 e L_4, que deveriam ser apenas uma, ainda não foram capazes de se unirem. Ou seja, quando utilizamos o modelo (i) para estimar a partição, observamos um certo embaralhamento na recuperação da estrutura original.

A fim de visualizar a recuperação da partição original por ambas abordagens, repetimos a simulação 100 vezes para quatro diferentes tamanhos de amostra. A tabela 1.7 apresenta a contagem do número de partes nas partições obtidas pelo algoritmo A.1 em 100 simulações com os tamanhos de amostra (n) 5×10^4, 10^6, 5×10^6 e 10^7. Apontamos que todos os casos em que a partição foi constituída de três partes, esta era equivalente à original. Notamos que, para o processo estocástico descrito no início deste exemplo, a abordagem (ii) recupera a partição original de maneira mais eficiente, ou seja, requer uma amostra menor que a abordagem (i) para a obtenção do resultado desejado.

(i) Número de partes em $\mathcal{S} = \{a, b, c\}^4$

n	3 partes	4 partes	5 partes	6 partes
5×10^4	20	45	31	4
10^6	63	35	2	0
5×10^6	82	18	0	0
10^7	92	8	0	0

(ii) Número de partes em $\mathcal{S}^* = \{a, b, c\}^2 \times \{a, b, c\}$

n	3 partes	4 partes
5×10^4	97	3
10^6	100	0
5×10^6	100	0
10^7	100	0

Tabela 1.7: Número de partes das partições estimadas para as configurações (i) e (ii) em 100 simulações, cada uma de tamanho n.

Encerraremos este capítulo apresentando aplicações específicas dos modelos

G3M, mas antes alguns comentários relevantes. Como dito anteriormente, modelar um processo utilizando cadeias de Markov completas pode não ser eficiente do ponto de vista inferencial, devido ao crescimento exponencial do espaço paramétrico em função da memória do processo. As cadeias de Markov de alcance variável (VLMC) apresentam uma amenização desse problema, pois permitem uma flexibilização do tamanho do passado relevante, que é uma função de observações decorridas. Dessa forma, como o comprimento do intervalo do passado relevante é variável, há uma potencial redução no espaço paramétrico do modelo. Como interpretado na seção 1.1, uma VLMC pode ser vista como uma cadeia de Markov completa que possui uma partição específica no espaço de estados: agrupamento de estados baseado em passados irrelevantes. Motivados pelo conceito de construção de partições no espaço de estados, foram desenvolvidos os modelos markovianos de partição. Essa família de modelos permite um delineamento mais geral de partições no espaço de estados. Essa estrutura é baseada apenas nas funções de probabilidades de transição, o que torna os modelos baseados em cadeias de Markov de alcance variável um caso particular. Assim, uma nova redução no espaço paramétrico pôde ser alcançada. Neste texto, aproveitamos a ideia de observações decorridas irrelevantes e propomos o caso em que a irrelevância também possa estar contida em um intervalo de informações relevantes. Para isso, foi introduzido o conceito de cadeia de Markov com interstício que visa desconsiderar trechos irrelevantes do passado. Adaptando esse novo processo para o conceito de modelos markovianos de partição, alcançamos uma abordagem mais eficiente e econômica (espaço paramétrico reduzido) que a tradicional, conforme ilustrado no exemplo 5.

Algo conveniente sobre os modelos para cadeias de Markov com interstício é a possibilidade de inclusão de um passado distante relevante que uma cadeia de Markov tradicional não seria capaz de incluir. Por exemplo, suponha que, para uma variável aleatória X_t, apenas as observações nos tempos $t-1$ e $t-12$ influenciam em seu resultado. Uma cadeia de Markov tradicional necessitaria de uma memória $o = 12$ para descrever seu passado relevante, enquanto uma cadeia de Markov com interstício com $M = 1$, $G = 12$ e $g = 0$ seria suficiente. Muitas vezes a amostra disponível não possui observações suficientes para que uma memória o tão grande possa ser considerada. Além disso, note que a primeira abordagem possui espaço de estados A^{12}, enquanto o da segunda é $A \times A$, que é mais econômico e mais simples de ser interpretado.

Outro fato conveniente é que podemos criar um processo ainda mais geral que a cadeia de Markov com interstício, que permita múltiplos saltos. Esse processo terá função de verossimilhança modificada com mesma estrutura que a apresentada na equação (1.15) e, consequentemente, o mesmo BIC (respeitando sempre as alterações necessárias para o novo espaço de estados estendido). Isso possibilitará que os resultados sobre a consistência da obtenção da partição boa mínima para esse processo continuem válidos. Dessa maneira, podemos descrever processos cujas estruturas de dependência no passado sejam cada vez mais elaboradas, sem que seja necessário aumentar de maneira exagerada a complexidade do modelo.

1.3.1 Modelos Markovianos para DNA de Covid-19 (Continuação)

Retornemos para o exemplo do sequenciamento do genoma da Covid-19. Temos interesse em aplicar o desenvolvido até aqui sobre modelos de partições para cadeias de Markov com interstício. Como visto anteriormente, o modelo de partições tradicional com memória $o = 5$ foi o que proporcionou o maior BIC entre os apresentados. Para esse modelo, o algoritmo A.1 encontrou uma partição composta por apenas 30 partes. Consequentemente, temos que estimar $|\mathcal{L}|(|A| - 1) = 30 \times 3 = 90$ parâmetros (probabilidades de transição), uma quantidade bem menor que a necessária para uma cadeia de Markov completa de ordem $o = 5$: $|A^5|(|A| - 1) = 1024 \times 3 = 3072$. Vamos permitir agora que possam existir observações de um passado mais distante que sejam relevantes, cuja informação queremos incluir no modelo. Para respeitar o tamanho máximo dos espaços de estados propostos na subseção 1.2.1 ($|A^5| = 1024$), vamos utilizar uma ordem direta $M = 4$ para nossas cadeias de Markov com interstício. A tabela 1.8 apresenta o valor do BIC observado para diferentes modelos de partição tradicionais e com interstício.

Pela tabela 1.8, podemos notar que os maiores BIC's ocorrem para os modelos que consideram um interstício nas observações relevantes do passado. Além disso, o modelo G3M($g = 0, G = 8, M = 4$) foi o que apresentou o maior BIC, maior até do que o obtido para um modelo de partições tradicional de ordem $o = 5$. Ou seja, a modelagem da estrutura de dependência que faz uso de um passado mais distante pode nos trazer mais informações sobre o processo do que uma modelagem que utiliza informações de

(i)		
o	$\lvert\mathcal{L}\rvert$	BIC
1	4	-39.965,08
2	7	-39.925,36
3	9	-39.832,03
4	15	-39.661,36
5	30	-38.848,28

(ii) com $M = 4$ e $g = 0$		
G	$\lvert\mathcal{L}\rvert$	BIC
6	27	-38.821,34
7	26	-38.797,09
8	30	-38.795,27
9	28	-38.801,49
10	26	-38.841,19
11	28	-38.837,77
12	25	-38.826,37

Tabela 1.8: À esquerda, configuração sob a perspectiva (i): modelo de partição tradicional, memória o, cardinalidade da partição $|\mathcal{L}|$ e BIC observado. À direita, configuração sob a perspectiva (ii): modelo de partição com interstício G, memória direta $M = 4$, ordem de interstício $g = 0$, cardinalidade da partição $|\mathcal{L}|$ e BIC observado.

um passado mais recente, mas que são menos relevantes. Note que, para descrever o modelo G3M($g = 0, G = 8, M = 4$), são necessários $|\mathcal{L}|(|A| - 1) = 30 \times 3 = 90$ parâmetros, uma quantidade factível para estimação. Portanto, conseguimos absorver informações longínquas importantes para o processo e desconsideramos as menos úteis ao longo do caminho.

1.3.2 Dependência Vicinal

Neste tópico, apresentamos uma aplicação das cadeias de Markov com interstício na modelagem de dependências vicinais bidimensionais sobre uma grelha finita, cujos vértices assumem valores em um alfabeto finito A. Em particular, podemos visualizar tal grelha como uma matriz, denotada por $\mathbf{B}$, com um número finito de colunas, $W \in \mathbb{N}$, e cada ponto como uma coordenada de $\mathbf{B}$, como ilustrado na figura 1.1.

A princípio, os modelos de partição apresentados neste capítulo têm como ponto de partida uma *string* de elementos provenientes de um alfabeto finito A. Por esse motivo, iremos transformar a matriz $\mathbf{B}$ em uma *string* T. Para isso, devemos definir uma regra de leitura, conforme apresentada na definição 16.

Definição 16. *Seja* $\mathbf{B}$ *uma matriz com* $W \in \mathbb{N}$ *colunas e* $R \in \mathbb{N}$ *linhas, com* $W < \infty$ *e* $R < \infty$*, cujos elementos são representados por* $B_{i,j}$*, para* $1 \leq i \leq R$ *e* $1 \leq j \leq W$*. Definimos que a H-leitura da matriz* $\mathbf{B}$ *é a string* T *constituída de forma que* $T_{(i-1)W+j} = B_{i,j}$*, para* $1 \leq i \leq R$ *e* $1 \leq j \leq W$*, em que* T_k *é o* k*-ésimo elemento da string* T*, para* $1 \leq k \leq |\mathbf{B}|$*, e* $|\mathbf{B}| = RW$ *é o número de elementos na matriz* $\mathbf{B}$*.*

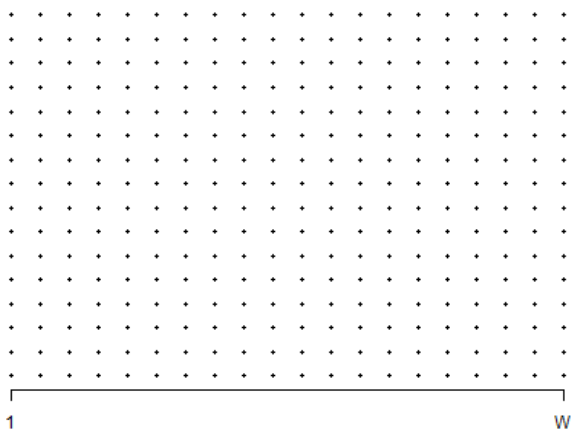

Figura 1.1: Visualização de uma matriz **B** de pontos com W colunas.

Um exemplo de H-leitura está apresentado na figura 1.2. Note que podemos definir agora qualquer vizinhança de pontos da matriz **B** como uma coleção de observações selecionadas da *string* T, gerada pela H-leitura de **B**. A estrutura de vizinhança que utilizaremos neste texto está formalizada na definição 17.

Definição 17. *Seja* **B** *uma matriz com* W *colunas e* R *linhas,* $W < \infty$ *e* $R < \infty$*. Seja* T *a string resultante da H-leitura de* **B**. *A vizinhança de um ponto* $p = B_{i,j} \in \mathbf{B}$*, que se corresponde com o elemento* $T_{(i-1)W+j}$*, é definida pelo conjunto de pontos da matriz* **B** *de coordenadas* $\{(i-1, j-1), (i-1, j), (i, j-1)\}$*. Equivalentemente, essa vizinhança pode ser traduzida pela concatenação dos seguintes elementos da string T:* $T_{(i-2)W+(j-1)}$, $T_{(i-2)W+j}$ *e* $T_{(i-1)W+(j-1)}$.

A vizinhança em **B** proposta na definição 17 é representada pelos pontos contidos na região tracejada na figura 1.3.

Nota 6. *É importante ressaltar que a vizinhança pode ser definida de diversas maneiras e possuir vários formatos. A escolha de vizinhança que fizemos foi devida a sua simples modelagem e fácil associação às cadeias de Markov com interstício, que serão apresentadas a seguir. Nos capítulos 3 e 4 sua forma parcimoniosa nos será conveniente para ilustrar procedimentos de codificação de matrizes e imagens, respectivamente.*

Exemplo 6. *Seja* **B** *uma matriz, cujas coordenadas assumem valores em* $A = \{a, b, c\}$,

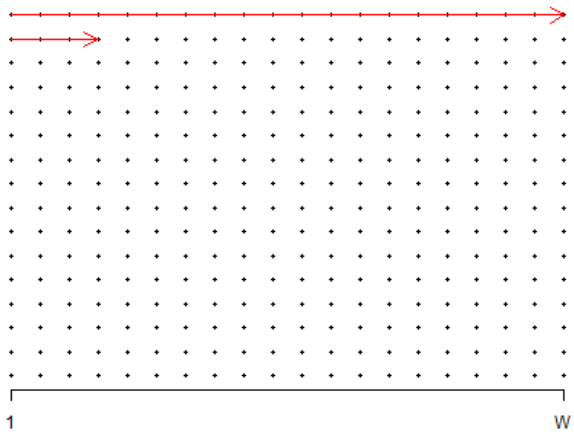

Figura 1.2: Visualização da H-leitura de uma matriz.

tal que

$$\mathbf{B} = \begin{bmatrix} a & a & c & b \\ c & b & c & a \\ a & c & a & b \end{bmatrix}.$$

A string T gerada pela H-leitura de $\mathbf{B}$ *é* $T = aacbcbcaacab$. *A vizinhança do ponto de coordenadas* $(2, 3)$ *proposta pela definição 17 é a string acb.*

Perceba que, fixado um formato de vizinhança, podemos construir probabilidades de transição de um ponto em $\mathbf{B}$ assumir determinado valor, dado que seus arredores possuem determinada configuração. Defina as probabilidades de transição $p(\cdot|v)$ para todo $v \in \mathcal{V}$, em que $\mathcal{V}$ é o conjunto de todas as possíveis vizinhanças delineadas pela definição 17 para $\mathbf{B}$ (ou equivalentemente, T, isto é, todas as possibilidades para $\Pr(T_{(i-1)W+j} = \cdot | T_{(i-2)W+(j-1)}\, T_{(i-2)W+j}\, T_{(i-1)W+(j-1)} = t_1t_2t_3)$, para toda combinação de $t_1t_2t_3 \in A^3$). Note que, ao adequarmos as ideias de vizinhança e probabilidades de transição definidas para a matriz $\mathbf{B}$ para a *string* T, obtemos o equivalente a uma cadeia de Markov com interstício apresentada na definição 12. Em particular, para a vizinhança ilustrada na figura 1.3, temos a cadeia associada com $M = 1$, $G = W$ e $g = 1$. Com essa construção, podemos adaptar os modelos de partição discutidos neste capítulo para submatrizes de $\mathbf{B}$ e aplicar as mesmas ferramentas inferenciais já trabalhadas, como o algoritmo A.1. Dessa forma, concluímos um paralelo entre algumas formas de dependência

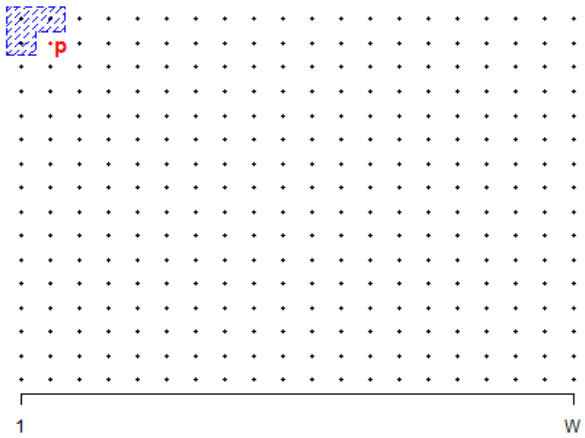

Figura 1.3: Visualização da vizinhança de um ponto p de uma matriz.

vicinal e cadeias de Markov com interstício. Retornaremos para este assunto com mais detalhes no capítulo 3, onde aplicaremos estes conceitos no escopo de compactação de matrizes.

1.4 Conclusão

Neste capítulo, apresentamos uma breve revisão de conceitos associados a processos estocásticos (cadeias de Markov, em particular) e a complexidade de seus métodos inferenciais. Apontamos que a estimação dos parâmetros (probabilidades de transição) do modelo é um procedimento delicado e requer tamanhos amostrais muito grandes para delineamentos elaborados, pois as quantidades a serem estimadas crescem exponencialmente com a memória do processo markoviano. Por esse motivo, estratégias para a simplificação do espaço de estados associado a cadeias de Markov foram desenvolvidas, como as cadeias de Markov de alcance variável e os modelos markovianos de partição, as quais possuem potencial de reduzir o número de parâmetros que requerem estimação. Este texto introduz o conceito de cadeias de Markov com interstício, que permite a inclusão de um passado distante relevante para a compreensão do processo sem que a complexidade do respectivo modelo tenha um grande aumento na quantidade de parâmetros livres. Indicamos, na seção 1.3, que as definições e resultados apresentados na seção 1.2 sobre modelos de partições podem ser estendidas para esta família de processos estocásticos mais ampla, culminando

nos modelos G3M.

A abordagem G3M é uma ferramenta maleável do ponto de vista de modelagem e parcimoniosa. Nas subseções 1.3.1 e 1.3.2 apresentamos aplicações da nova técnica para modelagem do genoma da Covid-19 e para delineamento de estruturas de dependência vicinal, respectivamente. Além disso, a abordagem G3M pode ser aplicada para complementar técnicas de comparação entre processos estocásticos, veja, por exemplo, [11], [12] e [14]. Em particular, em [11], a abordagem G3M foi utilizada para se estabelecer um modelo apropriado para um genoma sequenciado da Covid-19, de identificação MN908947 (o mesmo utilizado neste texto). Em seguida, utilizando o ferramental para comparação de processos estocásticos introduzido em [10], foi possível efetuar uma análise de conglomerados, permitindo visualizar que, dentre um conjunto de outros 11 genomas referentes a diferentes tipos de coronavírus, a sequência MN908947 era mais similar à variedade presente em morcegos do que às demais. Em [12], um ferramental análogo foi utilizado para indicar que variantes da Covid-19 estariam surgindo no território brasileiro e também para visualizar como estas se aproximavam das presentes em outros países, como Espanha e Itália. Em [14], detalhou-se o comportamento estocástico das bases nitrogenadas que constituem a variante P.1 da Covid-19. Nele, notou-se que, apesar da estrutura estocástica geral ser próxima da observada na sequência MN908947, original de Wuhan, há algumas construções particulares que se diferenciam claramente na composição do genoma. Atualmente, estudos semelhantes estão sendo realizados com as variantes B.1.1.7 e B.1.351.

Capítulo 2

Introdução à Teoria de Informação

A teoria de informação é o estudo que se preocupa essencialmente com o armazenamento e transmissão de dados (informação). Teve sua base estabelecida na primeira metade do século XX pelos trabalhos de Claude E. Shannon (veja [28]). No entanto, a ideia de entropia, pedra fundamental para a teoria da informação, já havia sido introduzida no campo da termodinâmica. No final do século XIX, o físico austríaco Ludwig E. Boltzmann propôs a ligação entre entropia termodinâmica e o logaritmo do número de microestados em um macroestado de um sistema. Com essa abordagem, Boltzmann conseguiu mostrar que a segunda lei da termodinâmica é um fato estatístico. Nessa abordagem, a entropia já podia ser associada a uma quantificação de "incerteza". O que Shannon introduziu foi a entropia como a quantificação de incerteza para qualquer variável aleatória discreta. Essa entropia passou a ser definida como o valor esperado da autoinformação de uma variável aleatória. A autoinformação quantifica a incerteza associada a um valor da variável aleatória, enquanto a entropia quantifica o quão informativa é a variável aleatória como um todo (ponderada sobre as probabilidades dos diferentes resultados possíveis). O estudo das quantificações de incerteza tem como um dos seus objetivos responder questões referentes ao campo de comunicação. Neste texto, avançamos o suficiente na teoria de informação para justificarmos o uso de alguns algoritmos de compactação de *strings*, matrizes e imagens (os quais também utilizarão os modelos markovianos de partição apresentados no capítulo 1) que são construídos nos próximos capítulos.

Na seção 2.1 apresentamos as definições básicas de quantidades de incerteza

para uma e múltiplas variáveis aleatórias, assim como algumas propriedades selecionadas. A escolha das propriedades expostas está intimamente ligada ao objetivo majoritário deste capítulo: a construção de códigos ótimos, cujos critérios serão apresentados na seção 2.3. Na seção 2.2 apresentamos o teorema da propriedade de equipartição assintótica, as definições de menores conjuntos de alta probabilidade e conjuntos típicos, assim como o conceito de alfabeto efetivo. Essa seção (em particular, o conceito de alfabeto efetivo) é uma peça chave para a construção do método de quantização de matrizes (procedimento utilizado para a compressão de imagens com perda de qualidade) explorado no capítulo 4. Na seção 2.3 apresentamos um panorama geral sobre o problema de codificação e compactação, que culminará na codificação de Huffman tradicional e na codificação de Huffman condicional. Exibimos resultados sobre os limitantes de taxas de compressão, que são há muito conhecidos e nos servirão como base para a seleção de códigos ótimos.

2.1 Quantificando Incerteza

Nesta seção expomos os conceitos de entropia, entropia condicional e taxa de entropia, assim como algumas de suas propriedades. Essas quantidades são fundamentais para os resultados associados à propriedade de equipartição assintótica e a sistemas de codificação apresentados nas seções 2.2 e 2.3, respectivamente. Imaginemos que um indivíduo (transmissor) envia mensagens para outro indivíduo (receptor). Cada mensagem pode ser enviada de diversas formas, contanto que ambos os indivíduos combinem previamente como esta deve ser interpretada. Por exemplo, uma frase na língua portuguesa pode ser transmitida em sua forma lexicográfica ou por uma coleção de outros símbolos que a represente, desde que essa regra já tenha sido preestabelecida pelo transmissor e pelo receptor. Um conjunto muito utilizado para a transmissão de mensagens é o alfabeto $\mathcal{B} = \{0, 1\}$, conhecido como alfabeto binário. Os elementos provenientes desse alfabeto são chamados de *bits* e constituem o menor conjunto de símbolos necessários para se transmitir uma mensagem. Por esse motivo, o alfabeto binário é extensivamente utilizado na teoria da informação, computação e comunicação digital.

Agora, podemos apresentar de maneira adequada o quantificador de incerteza, chamado de entropia, para uma variável aleatória discreta. Iremos considerar uma variável aleatória discreta X definida sobre um alfabeto A e com função de probabilidade

$p_X(x) = \Pr(X = x)$, $x \in A$. A definição 18 apresenta o conceito de entropia que utilizaremos.

Definição 18. *Seja X uma variável aleatória discreta definida sobre um alfabeto A e com função de probabilidade p_X. Definimos a entropia de X, $H(X)$, como*

$$H(X) = -\sum_{x \in A} p_X(x) \log p_X(x),$$

em que o log *utilizado é na base 2 e a entropia é expressa em bits.*

Nota 7. *Utilizaremos a convenção que $0 \log 0 = 0$. Com isso, a adição de termos com probabilidade zero não altera o valor da entropia.*

Exemplo 7. *Suponha X uma variável aleatória discreta com distribuição Bernoulli(p), ou seja, X é tal que*

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

Pela definição 18, temos que a entropia de X é

$$H(X) = -p \log p - (1-p) \log(1-p).$$

Note que podemos extrair algumas informações interessantes dessa simples entropia. Primeiramente, temos que essa entropia é bem definida para todo $p \in [0,1]$ e simétrica em $p = \frac{1}{2}$. Além disso, assume valor 0 para $p = 0$ e $p = 1$. Perceba que, para esses dois valores de p, não há mais incerteza sobre o possível valor de X, ou seja, a variável deixa de ser aleatória, o que justifica a entropia ser zero. Também, a entropia é máxima quando $p = \frac{1}{2}$ (ocasionando $H(X) = 1$ bit), ou seja, quando não há nenhum resultado de X mais provável que outro, que equivale ao máximo de incerteza que podemos ter sobre a variável aleatória.

Exemplo 8. *Suponha X uma variável aleatória discreta assumindo valores no alfabeto $A = \{2, 5, 6\}$ tal que*

$$X = \begin{cases} 2 & \textit{com probabilidade } \frac{1}{2}; \\ 5 & \textit{com probabilidade } \frac{3}{8}; \\ 6 & \textit{com probabilidade } \frac{1}{8}. \end{cases}$$

Pela definição 18, temos que a entropia de X é

$$H(X) = -\frac{1}{2}\log\frac{1}{2} - \frac{3}{8}\log\frac{3}{8} - \frac{1}{8}\log\frac{1}{8} = \frac{1}{2} + \frac{3}{8}(3 - \log 3) + \frac{3}{8} \approx 1,41 \; bit.$$

É importante notar que, tanto no exemplo 7 quanto no exemplo 8, o valor assumido pela variável aleatória não influencia no cálculo da entropia, ou seja, a entropia é inerente à distribuição das probabilidades apenas. Além disso, podemos observar que a entropia é máxima no caso das probabilidades serem uniformemente distribuídas no alfabeto (esse resultado é formalizado no teorema 5).

Teorema 5. *Seja X uma variável aleatória discreta definida sobre um alfabeto A finito. Então, $H(X) \leq \log|A|$, com igualdade se, e somente se, X possui distribuição uniforme em A.*

Nota 8. *As demonstrações dos teoremas e corolários deste capítulo que forem omitidas podem ser encontradas com detalhes em [3].*

Também, observe que $p_X(x) \in (0,1] \implies \log\frac{1}{p_X(x)} \geq 0$, logo, $H(X) \geq 0$. Dessa forma, quanto mais próxima de zero for a entropia, menos incerteza a variável aleatória carrega. Note também que, pela definição 18, a entropia da variável aleatória X pode ser vista como o valor esperado de outra variável aleatória: $\log\frac{1}{p_X(X)}$; em que X possui função de probabilidade p_X. Dessa forma,

$$H(X) = \mathbb{E}_{p_X}\left[\log\frac{1}{p_X(X)}\right].$$

É possível também definir um quantificador de incerteza para um conjunto de variáveis aleatórias discretas (que pode ser visto como um único vetor aleatório discreto), conforme apresentado na definição 19 para um par de variáveis (X,Y).

Definição 19. *Sejam X e Y variáveis aleatórias discretas definidas sobre os alfabetos A_X e A_Y, respectivamente, com função de probabilidade conjunta $p_{X,Y}(x,y)$. Então, a entropia conjunta de X e Y, $H(X,Y)$, é dada por*

$$H(X,Y) = -\sum_{x\in A_X}\sum_{y\in A_Y} p_{X,Y}(x,y)\log p_{X,Y}(x,y),$$

que também pode ser vista como $H(X,Y) = -\mathbb{E}_{p_{X,Y}}[\log p_{X,Y}(X,Y)]$.

Podemos também estabelecer o conceito de entropia condicional de uma variável aleatória dada outra. Essa quantidade é calculada como o valor esperado das entropias das distribuições condicionais ponderadas pela variável de condicionamento, conforme apresentado na definição 20.

Definição 20. *Sejam X e Y variáveis aleatórias discretas definidas sobre os alfabetos A_X e A_Y, respectivamente, com função de probabilidade conjunta $p_{X,Y}(x,y)$. Defina $p_{Y|X}(y|x) = \frac{p_{X,Y}(x,y)}{p_X(x)}$, se $p_X(x) > 0$, e $p_{Y|X}(y|x) = 0$, caso contrário. Então, a entropia condicional de Y dado X, $H(Y|X)$, é definida como*

$$\begin{aligned} H(Y|X) &= \sum_{x \in A_X} p_X(x) H(Y|X=x) = -\sum_{x \in A_X} \sum_{y \in A_Y} p_{X,Y}(x,y) \log p_{Y|X}(y|x) \\ &= -\mathbb{E}_{p_{X,Y}} \left[\log p_{Y|X}(Y|X) \right], \end{aligned}$$

já que, pela definição 18, $H(Y|X=x) = -\sum_{y \in A_Y} p_{Y|X}(y|x) \log p_{Y|X}(y|x)$.

Podemos interpretar a entropia condicional como a quantidade de incerteza que uma variável carrega após seu par ter sido observado. Essa interpretação é formalizada no teorema 6.

Teorema 6. *Sejam X e Y variáveis aleatórias discretas. Então,*

$$H(Y|X) = H(X,Y) - H(X).$$

Outra característica interessante sobre a entropia condicional é apresentada no teorema 7, o qual postula que informações adicionais sobre outras variáveis só podem, em média, reduzir a incerteza sobre a variável de interesse. Este é um princípio importante, principalmente quando estamos no campo da modelagem, pois garante que, ao incluirmos variáveis em um modelo, mesmo que estas não carreguem informação relevante sobre o fenômeno estudado, não estamos aumentando (em média) a incerteza associada.

Teorema 7. *Sejam X e Y variáveis aleatórias discretas. Então, $H(Y|X) \leq H(Y)$, com igualdade se, e somente se, X e Y são independentes.*

Corolário 5. *Sejam X e Y variáveis aleatórias discretas. Se X e Y são independentes, então $H(X,Y) = H(X) + H(Y)$.*

Exemplo 9. *Sejam X e Y variáveis aleatórias definidas sobre o mesmo alfabeto $A = \{a, b\}$,*

com função de probabilidade conjunta $p_{X,Y}(x,y)$ *dada por*

$$p_{X,Y}(a,a)=\frac{1}{2},\quad p_{X,Y}(a,b)=\frac{1}{4},\quad p_{X,Y}(b,a)=\frac{1}{8}\quad e\quad p_{X,Y}(b,b)=\frac{1}{8}.$$

Pela definição 19, temos que a entropia conjunta de X *e* Y *é dada por*

$$\begin{aligned}H(X,Y)=-\sum_{x\in A}\sum_{y\in A}p_{X,Y}(x,y)\log p_{X,Y}(x,y)&=-\left[\frac{1}{2}\log\frac{1}{2}+\frac{1}{4}\log\frac{1}{4}+\frac{1}{8}\log\frac{1}{8}+\frac{1}{8}\log\frac{1}{8}\right]\\&=\frac{14}{8}=1,75.\end{aligned}$$

A partir da função de probabilidade conjunta $p_{X,Y}(x,y)$ *podemos definir as funções de probabilidade marginais de* X *e* Y*, denotadas, respectivamente, por* $p_X(x)$ *e* $p_Y(y)$*. Assim,*

$$p_X(a)=p_{X,Y}(a,a)+p_{X,Y}(a,b)=\frac{1}{2}+\frac{1}{4}=\frac{3}{4}$$

e

$$p_Y(a)=p_{X,Y}(a,a)+p_{X,Y}(b,a)=\frac{1}{2}+\frac{1}{8}=\frac{5}{8}.$$

Com isso, podemos calcular as entropias de X *e* Y *como*

$$H(X)=-\left[\frac{3}{4}\log\frac{3}{4}+\frac{1}{4}\log\frac{1}{4}\right]\approx 0,8113$$

e

$$H(Y)=-\left[\frac{5}{8}\log\frac{5}{8}+\frac{3}{8}\log\frac{3}{8}\right]\approx 0,9544.$$

Também, conforme a definição 20, podemos calcular a entropia de Y *dado* X *como*

$$\begin{aligned}H(Y|X)&=-\sum_{x\in A}\sum_{y\in A}p_{X,Y}(x,y)\log\frac{p_{X,Y}(x,y)}{p_X(x)}\\&=-\left[p_{X,Y}(a,a)\log\frac{p_{X,Y}(a,a)}{p_X(a)}+p_{X,Y}(a,b)\log\frac{p_{X,Y}(a,b)}{p_X(a)}+p_{X,Y}(b,a)\log\frac{p_{X,Y}(b,a)}{p_X(b)}\right.\\&\quad\left.+p_{X,Y}(b,b)\log\frac{p_{X,Y}(b,b)}{p_X(b)}\right]\\&=-\left[\frac{1}{2}\log\frac{2}{3}+\frac{1}{4}\log\frac{1}{3}+\frac{1}{8}\log\frac{1}{2}+\frac{1}{8}\log\frac{1}{2}\right]\approx 0,9387.\end{aligned}$$

Note que $H(Y)=0,9544>0,9387=H(Y|X)$*, conforme já havia sido estabelecido pelo teorema 7, ou seja, a informação sobre a variável aleatória* X *diminuiu a incerteza sobre a variável* Y*. Além disso, observamos que* $H(X,Y)=1,75=0,9387+0,8113=$

$H(Y|X) + H(X)$, conforme já enunciado no teorema 6.

Exemplo 10. *Sejam X e Y variáveis aleatórias definidas sobre o mesmo alfabeto $A = \{a, b\}$, com função de probabilidade conjunta $p_{X,Y}(x, y)$ dada por*

$$p_{X,Y}(a,a) = \frac{1}{8}, \quad p_{X,Y}(a,b) = \frac{1}{8}, \quad p_{X,Y}(b,a) = \frac{3}{8} \quad e \quad p_{X,Y}(b,b) = \frac{3}{8}.$$

Pela definição 19, temos que a entropia conjunta de X e Y é dada por

$$H(X,Y) = -\sum_{x\in A}\sum_{y\in A} p_{X,Y}(x,y)\log p_{X,Y}(x,y) = -\left[\frac{1}{8}\log\frac{1}{8} + \frac{1}{8}\log\frac{1}{8} + \frac{3}{8}\log\frac{3}{8} + \frac{3}{8}\log\frac{3}{8}\right]$$
$$\approx 1,8113.$$

A partir da função de probabilidade conjunta $p_{X,Y}(x, y)$ podemos definir as funções de probabilidade marginais de X e Y, denotadas, respectivamente, por $p_X(x)$ e $p_Y(y)$. Assim,

$$p_X(a) = p_{X,Y}(a,a) + p_{X,Y}(a,b) = \frac{1}{8} + \frac{1}{8} = \frac{1}{4}$$

e

$$p_Y(a) = p_{X,Y}(a,a) + p_{X,Y}(b,a) = \frac{1}{8} + \frac{3}{8} = \frac{1}{2}.$$

Observe que $p_{X,Y}(x, y) = p_X(x)p_Y(y)$, $\forall x, y \in \{a, b\}$, ou seja, X e Y são variáveis aleatórias independentes.

Podemos calcular as entropias de X e Y como

$$H(X) = -\left[\frac{1}{4}\log\frac{1}{4} + \frac{3}{4}\log\frac{3}{4}\right] \approx 0,8113$$

e

$$H(Y) = -\left[\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right] = 1.$$

Também, conforme a definição 20, podemos calcular a entropia de Y dado X como

$$\begin{aligned} H(Y|X) &= -\sum_{x\in A}\sum_{y\in A} p_{X,Y}(x,y)\log\frac{p_{X,Y}(x,y)}{p_X(x)} \\ &= -\left[p_{X,Y}(a,a)\log\frac{p_{X,Y}(a,a)}{p_X(a)} + p_{X,Y}(a,b)\log\frac{p_{X,Y}(a,b)}{p_X(a)} + p_{X,Y}(b,a)\log\frac{p_{X,Y}(b,a)}{p_X(b)}\right. \\ &\quad \left. + p_{X,Y}(b,b)\log\frac{p_{X,Y}(b,b)}{p_X(b)}\right] \end{aligned}$$

$$= -\left[\frac{1}{8}\log\frac{1}{2} + \frac{1}{8}\log\frac{1}{2} + \frac{3}{8}\log\frac{1}{2} + \frac{3}{8}\log\frac{1}{2}\right] = 1.$$

Note que $H(Y) = 1 = H(Y|X)$, ou seja, a informação sobre a variável aleatória X não modificou a incerteza associada à variável Y. Esse resultado concorda com o estabelecido no teorema 7 para um par de variáveis aleatórias independentes. Além disso, observamos $H(X,Y) = 1,8113 = 0,8113 + 1 = H(X) + H(Y)$, conforme o enunciado pelo corolário 5.

A definição 19 adaptou o conceito de entropia para um par de variáveis aleatórias. O próximo passo é obter uma quantificação de incerteza para um processo estocástico $\{X_t\}_{t\in\mathbb{N}}$. Neste caso, temos interesse na taxa de crescimento da entropia de uma sequência de variáveis aleatórias. Esse conceito é denominado taxa de entropia e sua formalização é apresentada na definição 21.

Definição 21. *Seja $\{X_t\}_{t\in\mathbb{N}}$ um processo estocástico definido sobre um alfabeto finito A. Considere $X_1, \ldots, X_n$ uma amostra do processo $\{X_t\}_{t\in\mathbb{N}}$. Então, a taxa de entropia do processo estocástico $\{X_t\}_{t\in\mathbb{N}}$, denotada por $H(A)$, é definida por*

$$H(A) = \lim_{n\to\infty} \frac{1}{n} H(X_1, \ldots, X_n),$$

quando o limite existe.

Teorema 8. *Suponha $\{X_t\}_{t\in\mathbb{N}}$ um processo estocástico definido sobre um alfabeto finito A. Então, quando a taxa de entropia $H(A)$ está bem definida, temos que $H(A) \leq \log|A|$.*

Demonstração. *Considere a sequência de variáveis aleatórias $X_1, \ldots, X_n$, $n > 1$. Tomando $(X_2, \ldots, X_n)$ como um vetor aleatório, pelo teorema 6, temos que*

$$H(X_1, \ldots, X_n) = H(X_1) + H(X_2, \ldots, X_n|X_1). \tag{2.1}$$

Aplicando recursivamente a equação (2.1), obtemos

$$H(X_1, \ldots, X_n) = \sum_{i=1}^{n} H(X_i|X_1, \ldots, X_{i-1}). \tag{2.2}$$

Pelo teorema 7, temos que o condicionamento não aumenta a entropia. Logo, $H(X_i|X_1, \ldots, X_{i-1}) \leq H(X_i)$, para $i \in \{1, \ldots, n\}$. E pelo teorema 5, $H(X_i) \leq \log|A|$.

Assim, assumindo que o limite existe,

$$H(A) = \lim_{n\to\infty} \frac{1}{n} H(X_1, \ldots, X_n) \leq \lim_{n\to\infty} \frac{1}{n} n \log |A| = \log |A|. \tag{2.3}$$

□

É interessante notar que, nas condições do teorema 8, a igualdade é alcançada quando as variáveis aleatórias são independentes e uniformemente distribuídas em A, pois, pela equação (2.2) e pelo corolário 5, temos

$$\begin{aligned} H(A) &= \lim_{n\to\infty} \frac{1}{n} H(X_1, \ldots, X_n) = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H(X_i|X_1, \ldots, X_{i-1}) \\ &= \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H(X_1) = H(X_1) \\ &= \log |A|. \end{aligned} \tag{2.4}$$

Encerramos esta seção com uma última observação que será utilizada nos capítulos seguintes. Pelos teoremas 6 e 7, podemos extrair que $H(X, Y) \leq H(X) + H(Y)$, com igualdade se, e somente se, X e Y são independentes. Ou seja, a independência acarreta na incerteza máxima. Aplicando esse raciocínio na equação (2.2), obtemos

$$H(X_1, \ldots, X_n) = \sum_{i=1}^{n} H(X_i|X_1, \ldots, X_{i-1}) \leq \sum_{i=1}^{n} H(X_i),$$

com igualdade se, e somente se, $X_1, \ldots, X_n$ são variáveis aleatórias independentes. Assim, se a taxa de entropia está bem definida,

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} H(X_i) \tag{2.5}$$

também é um limite superior para $H(A)$, quando o limite existe. Em particular, se $X_1, \ldots, X_n$ são variáveis aleatórias independentes e identicamente distribuídas, $H(A) = H(X_1)$.

2.2 Propriedade de Equipartição Assintótica

No capítulo anterior, mostramos nosso interesse sobre as propriedades de processos estocásticos e, como estabelecido na definição 1, um processo estocástico é uma sequência indexada de variáveis aleatórias. Nesta seção, apresentamos alguns resultados importantes sobre o comportamento assintótico das quantificações de incerteza, apresentadas na seção 2.1, que abrangem muitos casos dessas sequências de variáveis. Iniciamos apresentando o teorema da propriedade de equipartição assintótica (AEP, do inglês *Asymptotic Equipartition Property*), conforme estabelecido no teorema 9. Essa propriedade nos diz, basicamente, que, se $X_1, \ldots, X_n$ são variáveis aleatórias independentes e identicamente distribuídas, então o valor de $\frac{1}{n} \log \frac{1}{p_{X_1,\ldots,X_n}(X_1,\ldots,X_n)}$ é próximo do valor da entropia $H(X_1)$. A derivação desse resultado é devida a uma aplicação da lei fraca dos grandes números.

Teorema 9. *Sejam $X_1, \ldots, X_n$ variáveis aleatórias discretas independentes e identicamente distribuídas, com função de probabilidade p_{X_1}. Denote por $p_{X_1,\ldots,X_n}$ a função de probabilidade conjunta de $X_1, \ldots, X_n$. Então,*

$$-\frac{1}{n} \log p_{X_1,\ldots,X_n}(X_1, \ldots, X_n) \to H(X_1) \quad \textit{em probabilidade.} \tag{2.6}$$

Note que podemos rearranjar os termos da equação (2.6) de forma a isolar a componente $p_{X_1,\ldots,X_n}(X_1, \ldots, X_n)$. Isso nos dá a ideia de um critério que possibilite segregar o conjunto de todas as possíveis observações $(x_1, \ldots, x_n) \in A^n$ em dois grupos baseados no valor da probabilidade $p_{X_1,\ldots,X_n}(x_1, \ldots, x_n)$. As sequências cujas probabilidades estejam próximas de $2^{-nH(X_1)}$ formarão o conjunto típico e as demais formarão o conjunto atípico. A definição 22 estabelece formalmente o conjunto típico e o teorema 10 apresenta algumas propriedades importantes. Relembramos que as demonstrações dos resultados apresentados podem ser consultadas em detalhes em [3].

Definição 22. *Sejam $X_1, \ldots, X_n$ variáveis aleatórias discretas independentes e identicamente distribuídas, definidas sobre o alfabeto A, com função de probabilidade p_X e função de probabilidade conjunta $p_{X_1,\ldots,X_n}$. Para $\varepsilon > 0$, o conjunto típico $A_\varepsilon^{(n)}$ é o conjunto das observações $(x_1, \ldots, x_n) \in A^n$ tais que*

$$2^{-n(H(X_1)+\varepsilon)} \leq p_{X_1,\ldots,X_n}(x_1, \ldots, x_n) \leq 2^{-n(H(X_1)-\varepsilon)}.$$

Teorema 10. *Sejam $X_1, \ldots, X_n$ variáveis aleatórias discretas definidas sobre o alfabeto A, independentes e identicamente distribuídas, com função de probabilidade p_X e função de probabilidade conjunta $p_{X_1,\ldots,X_n}$. Para $\varepsilon > 0$, seja $A_\varepsilon^{(n)}$ o conjunto típico associado à sequência $X_1, \ldots, X_n$. Então,*

1. *se $(x_1, \ldots, x_n) \in A_\varepsilon^{(n)}$, temos $H(X_1) - \varepsilon \leq -\frac{1}{n} \log p_{X_1,\ldots,X_n}(x_1, \ldots, x_n) \leq H(X_1) + \varepsilon$;*
2. *$Pr\left(A_\varepsilon^{(n)}\right) > 1 - \varepsilon$ para n suficientemente grande;*
3. *$\left|A_\varepsilon^{(n)}\right| \leq 2^{n(H(X_1)+\varepsilon)}$;*
4. *$\left|A_\varepsilon^{(n)}\right| \geq (1-\varepsilon)2^{n(H(X_1)-\varepsilon)}$.*

Apesar de apresentar propriedades muito importantes para nosso estudo, as demonstrações do teorema 10 são simples e serão brevemente comentadas. A demonstração do item 1 do teorema é imediata do conceito de conjunto típico apresentado na definição 22. A do item 2 segue direto da convergência em probabilidade enunciada no teorema 9. A demonstração do item 3 utiliza a lei da probabilidade total para dois conjuntos: o conjunto típico $A_\varepsilon^{(n)}$ e seu complementar. Com essa separação de probabilidades, uma desigualdade apropriada é delineada e a definição 22 é novamente aplicada. Por fim, a demonstração do item 4 parte do estabelecido no item 2 deste mesmo teorema, em que, após recorrermos mais uma vez à definição 22, alcançamos a desigualdade desejada.

Perceba que algumas interpretações interessantes podem ser feitas a partir do teorema 10. A primeira delas é que podemos construir um conjunto típico com probabilidade arbitrariamente próxima de 1. Outro fato relevante é que, para n suficientemente grande, as sequências pertencentes ao conjunto típico são praticamente equiprováveis. Por fim, os itens 3 e 4 do teorema 10 nos dão uma noção do número de elementos no conjunto típico: aproximadamente $2^{nH(X_1)}$. É interessante notar que, conforme a distribuição das variáveis aleatórias se distancia da uniformidade sobre o alfabeto A, menor é o tamanho do conjunto típico quando comparado ao número total das possibilidades das sequências $X_1, \ldots, X_n$, $|A|^n$. Dessa forma, o conjunto típico pode ser um conjunto consideravelmente pequeno, mas que concentra grande parte da probabilidade. Motivados pela ideia de concentração de probilidade em pequenos conjuntos, a definição 23 apresenta o conceito de menor conjunto de alta probabilidade.

Definição 23. *Sejam $X_1, \ldots, X_n$ variáveis aleatórias definidas sobre o alfabeto A. Então,*

para $\delta > 0$ e para $n \in \mathbb{N}$, o menor conjunto de alta probabilidade, $B_\delta^{(n)}$, é o menor conjunto (em cardinalidade) formado por observações $(x_1, \ldots, x_n) \in A^n$ com $Pr(B_\delta^{(n)}) \geq 1 - \delta$.

Como o conjunto típico é um conjunto denso em probabilidade (poucos elementos concentram muita probabilidade) é de se esperar que boa parte de seus elementos também façam parte do menor conjunto para dado limiar da probabilidade. Dessa forma, é razoável pensar que ambos os conjuntos possuam aproximadamente a mesma quantidade de elementos. De fato, isso acontece e é uma consequência direta do teorema 11, apresentada no corolário 6.

Teorema 11. *Sejam $X_1, \ldots, X_n$ variáveis aleatórias discretas independentes e identicamente distribuídas. Para $\delta < \frac{1}{2}$ e $\delta' > 0$, se $Pr\left(B_\delta^{(n)}\right) > 1 - \delta$, então*

$$\frac{1}{n} \log \left|B_\delta^{(n)}\right| > H(X_1) - \delta' \quad \textit{para } n \textit{ suficientemente grande.}$$

Corolário 6. *Sejam $X_1, \ldots, X_n$ variáveis aleatórias discretas independentes e identicamente distribuídas. Se $\varepsilon_n \to 0$ e $\delta_n \to 0$, então*

$$\lim_{n\to\infty} \frac{1}{n} \log \frac{\left|B_{\delta_n}^{(n)}\right|}{\left|A_{\varepsilon_n}^{(n)}\right|} = 0 \quad e \quad \lim_{n\to\infty} \frac{1}{n} \log \frac{\left|A_{\varepsilon_n}^{(n)}\right|}{2^{nH(X_1)}} = 0.$$

Apesar de não aparecer de maneira explícita, o conceito de conjunto típico é necessário para a obtenção do resultado estabelecido no teorema 11. Em particular, é a partir da interseção do menor conjunto de alta probabilidade $B_\delta^{(n)}$ e do conjunto típico $A_{\delta'}^{(n)}$ que conseguimos alcançar a desigualdade almejada. Lembremos que, pelo item 2 do teorema 10, para n suficientemente grande, $\Pr\left(A_\varepsilon^{(n)}\right) > 1 - \varepsilon$. Além disso, pelo conceito de menor conjunto de alta probabilidade apresentado na definição 23, temos que $B_\delta^{(n)}$ é o menor conjunto tal que $\Pr\left(B_\delta^{(n)}\right) \geq 1 - \delta$. Dessa forma, para $\delta = \varepsilon$ (ou também, para δ suficientemente próximo de ε), $\left|B_\delta^{(n)}\right| \leq \left|A_\varepsilon^{(n)}\right|$. Equivalentemente, temos que $\log \frac{|B_\delta^{(n)}|}{|A_\varepsilon^{(n)}|} \leq 0$. Assim, o corolário 6 propõe que, conforme n cresce, o tamanho do conjunto $A_\varepsilon^{(n)}$ se aproxima do tamanho do conjunto $B_\delta^{(n)}$. Mais especificamente, para n suficientemente grande, os tamanhos do conjunto típico e do menor conjunto de alta probabilidade são iguais até a primeira ordem do expoente. Além disso, pela segunda igualdade do corolário, o tamanho de ambos os conjuntos é de aproximadamente $2^{nH(X_1)}$,

para n suficientemente grande.

Lembremos que, pela definição 18, a entropia de uma variável aleatória X é $H(X) = \sum_{x \in A} p_X(x) \log \frac{1}{p_X(x)}$. A função $p \log \frac{1}{p}$ é côncava, assimétrica à direita para $p \in [0, 1]$, e assume valor 0 para $p = 0$ e $p = 1$. Assim, valores de p muito próximos de 0 ou 1 contribuem proporcionalmente pouco para a entropia (uma probabilidade muito próxima de 1 implica automaticamente em pouca incerteza sobre a variável aleatória X). Dessa forma, valores de p próximos de 0 ou 1 contribuem pouco também para os tamanhos do conjunto típico e do menor conjunto de alta probabilidade (uma probabilidade muito próxima de 1 implica as demais muito próximas de 0). Ou seja, podem existir elementos no alfabeto cujas presenças são escassas nos menores conjuntos. Como, para uma amostra de tamanho n de X, o tamanho do menor conjunto é uma contagem n-dimensional, temos que o correspondente tamanho de cada dimensão é $\left(|B_\delta^{(n)}|\right)^{\frac{1}{n}}$, e, pelo teorema 11,

$$\left(|B_\delta^{(n)}|\right)^{\frac{1}{n}} \approx \left(2^{nH(X)}\right)^{\frac{1}{n}} = 2^{H(X)}, \quad \text{para } n \text{ suficientemente grande.} \tag{2.7}$$

Informalmente, a equação (2.7) nos diz que aproximadamente $2^{H(X)}$ elementos do alfabeto de X são capazes de descrever os possíveis valores da variável aleatória que concentram as maiores probabilidades. Essa quantidade é o que chamaremos de tamanho do alfabeto efetivo da variável aleatória X.

A propriedade de equipartição assintótica, inicialmente definida para uma sequência de variáveis aleatórias independentes e identicamente distribuídas, pode ser estendida para o caso de um processo estocástico estacionário e ergódico. Esse resultado é chamado de teorema de Shannon–McMillan–Breiman e é apresentado no teorema 12.

Teorema 12. *Seja $\{X_t\}_{t \in \mathbb{N}}$ um processo estocástico estacionário e ergódico definido sobre um alfabeto finito A. Então,*

$$-\frac{1}{n} \log p_{X_1,\ldots,X_n}(X_1, \ldots, X_n) \to H(A) \quad \textit{com probabilidade 1,}$$

em que $H(A)$ é a taxa de entropia do processo, dada pela definição 21.

É interessante notar que o teorema 12 não se preocupa em questionar a existência da taxa de entropia $H(A)$. Isso é devido ao fato que $H(A)$ é bem definida se $\{X_t\}_{t \in \mathbb{N}}$ é um processo estacionário (veja [3]). Além de ser um resultado mais forte que o teorema 9, as

propriedades apresentadas no teorema 10 mantém-se válidas sob as suposições do teorema 12 e quando consideramos conjuntos típicos para subsequências do processo estocástico. De forma análoga ao argumentado anteriormente, para cada subsequência de tamanho $n \in \mathbb{N}$ podemos construir conjuntos típicos cujas probabilidades são arbitrariamente próximas de 1, e que possuirão aproximadamente $2^{nH(A)}$ elementos aproximadamente equiprováveis. Por fim, podemos também transportar a ideia do tamanho do alfabeto efetivo, apresentado na equação (2.7), de maneira que o tamanho do alfabeto efetivo para o processo $\{X_t\}_{t\in\mathbb{N}}$ será $2^{H(A)}$. Este resultado será invocado no capítulo 4, quando estabeleceremos uma quantidade de representantes para uma matriz de intensidades associada a uma imagem.

2.3 Compressão de Dados

No início deste capítulo, dissemos que uma das principais aplicações da teoria da informação é o estudo de técnicas de armazenamento de dados. Nesta seção, apresentamos alguns resultados que relacionam os conceitos de incerteza introduzidos anteriormente a taxas de compressão de dados. A compressão será analisada através da comparação dos comprimentos de *strings*. Suponha que queremos armazenar ou transmitir uma mensagem (*string*) formada por elementos de um alfabeto A. No entanto, nem sempre podemos manusear com facilidade o domínio original A. Em computação e comunicação digital, geralmente, é necessário converter as mensagens de alfabetos complexos em mensagens de alfabetos mais simples, como o binário. Dessa forma, é necessário definir uma função que mapeie A (ou mensagens formadas por elementos de A) para *strings* formadas por elementos do alfabeto binário. Essa função é chamada de código e é estabelecida formalmente na definição 24.

Definição 24. *Sejam X uma variável aleatória sobre um alfabeto A, $\mathcal{B} = \{0, 1\}$ um alfabeto binário e $\mathcal{B}^*$ o conjunto das strings de comprimento finito constituídas por símbolos de $\mathcal{B}$. Um código (ou sistema de codificação) para uma variável aleatória X é uma função $C : A \to \mathcal{B}^*$. Denotamos por $C(x)$ o código correspondente a x e por $l(x)$ o comprimento de $C(x)$, em que x é um valor assumido pela variável aleatória X.*

Notação 5. *Utilizaremos indiscriminadamente a função C como extensão de um sistema de codificação para strings provenientes do alfabeto original A. Dessa forma, para $x_1, \ldots, x_n$*

elementos de A, temos que a codificação da string formada pela concatenação $x_1 \dots x_n$ é

$$C(x_1 \dots x_n) = C(x_1) \dots C(x_n),$$

em que $C(x_1) \dots C(x_n)$ é a concatenação dos códigos individuais.

Exemplo 11. *Suponha $A = \{a, g, o, t\}$ um alfabeto e C um sistema de codificação tal que*

$$C(a) = 0, \quad C(g) = 01, \quad C(o) = 10 \quad e \quad C(t) = 00. \tag{2.8}$$

Então, a codificação da string "gato" é dada por $C(gato) = C(g)C(a)C(t)C(o) =$ 01 0 00 10.

Imagine que, após a codificação da mensagem apresentada no exemplo 11, um decodificador deseje recuperar a mensagem original, tendo conhecimento apenas do sistema de codificação definido em (2.8). Perceba que haverá complicações para recuperar a *string gato*, pois a mensagem 0100010 pode se referir aos códigos das *strings aoto*, *gaaao* ou *aoaao*, por exemplo. Dessa forma, precisamos colocar restrições em nossos sistemas de codificação para evitar ambiguidades. A definição 25 apresenta o conceito de um código unicamente decodificável.

Definição 25. *Sob as condições da definição 24, C é um código unicamente decodificável se,*

$$s \neq s' \implies C(s) \neq C(s'), \ \forall s, s' \in A^*,$$

em que A^* é o conjunto de todas as strings de comprimento finito formadas por elementos do alfabeto A.

A definição 25 estabelece que o mapeamento do código C deve levar diferentes mensagens provenientes do alfabeto original para diferentes *strings* binárias. No entanto, essa função pode ser definida de diversas formas. Neste texto, estaremos interessados em códigos instantâneos, que são sistemas de codificação que possibilitam a decodificação de cada símbolo no momento de sua leitura, sem a necessidade de que a *string* binária seja processada como um todo. Um dos códigos usuais com essa propriedade são os códigos de prefixo, introduzidos na definição 26.

Definição 26. *Sejam A um alfabeto e C um sistema de codificação sobre A. O código C*

é um código de prefixo se, $\forall x, x' \in A$*, com* $x \neq x'$*, temos que* $C(x)$ *não é prefixo de* $C(x')$*.*

Exemplo 12. *Suponha* $A = \{a, b, c\}$ *um alfabeto e* C *um sistema de codificação tal que*

$$C(a) = 0, \quad C(b) = 10 \quad e \quad C(c) = 01.$$

Note que o código C *não é de prefixo, pois* $C(a)$ *é prefixo de* $C(c)$*, especificamente,* $C(c) = C(a)1$*. Em contrapartida, o sistema de codificação* C^**, tal que*

$$C^*(a) = 0, \quad C^*(b) = 10 \quad e \quad C^*(c) = 11,$$

é um código de prefixo, pois nenhum elemento codificado de A *é prefixo de outro.*

Suponha que os dados que temos interesse em codificar são gerados segundo uma lei de probabilidade. Dessa forma, uma quantidade interessante para análise é o comprimento médio do código, que nada mais é que a soma dos comprimentos das representações binárias de cada elemento do alfabeto ponderada pela função de probabilidade que supostamente gera os dados. Essa ideia é formalizada na definição 27.

Definição 27. *Seja* X *uma variável aleatória definida sobre o alfabeto* A *e com função de probabilidade* p_X*. Seja* C *um sistema de codificação para* X*. Então, o comprimento esperado do código* C *é dado por*

$$L(C) = \sum_{x \in A} p_X(x) l(x),$$

em que $l(x)$ *é o comprimento de* $C(x)$*, para* $x \in A$*.*

Em geral, dois aspectos são essenciais quando estamos tratando de codificação de dados: taxa de compressão (representação da mensagem original em uma *string* binária com o menor comprimento esperado possível); e eficiência de compressão/descompressão (custo computacional necessário para codificar e decodificar uma mensagem). Neste texto, tratamos apenas do primeiro tópico. Então, a primeira pergunta natural é: se temos interesse em codificar uma mensagem cujos símbolos obedecem a alguma lei estocástica, conseguimos estabelecer limitantes para o comprimento esperado da *string* codificada? A resposta é sim e [3] apresenta uma sequência de resultados sobre o assunto, da qual extrairemos os que nos serão úteis na sequência do trabalho. O primeiro deles é o teorema

13 que estabelece que o comprimento esperado de um código é limitado inferiormente pela entropia da variável aleatória em questão. A essência da demonstração deste teorema (apresentada de maneira simples em [3]) está no uso da desigualdade de Kraft (veja [22]), que estipula um limitante para a soma das potências dos comprimentos dos códigos associados a um sistema de codificação. Uma versão simplificada da desigualdade de Kraft é apresentada no teorema 14.

Teorema 13. *Seja X uma variável aleatória definida sobre o alfabeto A e com função de probabilidade p_X. Seja C um sistema de codificação binário para X. O comprimento esperado L de um código binário de prefixo para uma variável aleatória X é maior ou igual a sua entropia $H(X)$, ou seja,*

$$H(X) \leq L,$$

com igualdade se, e somente se, $p_X(x) = 2^{-l(x)}$, $\forall x \in A$, em que $l(x)$ é o comprimento de $C(x)$.

Teorema 14. *Sejam C um sistema de codificação de prefixo definido sobre um alfabeto A e $l(x_i)$ o comprimento associado a $C(x_i)$, para $x_i \in A$ e $i \in \{1, \ldots, |A|\}$. Então,*

$$\sum_{i=1}^{|A|} 2^{-l(x_i)} \leq 1.$$

Reciprocamente, dado um conjunto de comprimentos $\{l_1^*, \ldots, l_{|A|}^*\}$ que satisfaça a desigualdade acima, temos que existe um código de prefixo, C^*, com tais comprimentos.

Antes de prosseguirmos, destacamos novamente nosso objetivo. Desejamos obter um código instantâneo (mais especificamente, de prefixo) cujo comprimento esperado seja mínimo. Um sistema de codificação que cumpre essas condições será chamado de código de prefixo ótimo. Para este sistema de codificação, o teorema 15 apresenta um limite superior para o comprimento esperado do código. Esse limitante superior é obtido utilizando a recíproca da deligualdade de Kraft para o conjunto de comprimentos $\{l_1^*, \ldots, l_{|A|}^*\}$ tal que $l_i^* = \left\lceil \log \frac{1}{p_X(x_i)} \right\rceil$, para $x_i \in A$ e $i \in \{1, \ldots, |A|\}$, em que $\lceil x \rceil$ é o menor inteiro maior ou igual a x.

Teorema 15. *Seja X uma variável aleatória definida sobre o alfabeto A e com função de probabilidade p_X. Seja C^* um código binário de prefixo ótimo para X. Sejam $l^*(x_i)$, $i \in \{1, \ldots, |A|\}$, os comprimentos dos códigos associados a cada elemento de A, e $L^* =$*

$\sum_{i=1}^{|A|} p_X(x_i)\, l^*(x_i)$ *o comprimento esperado do código. Então,*

$$H(X) \leq L^* < H(X) + 1.$$

Note que, a partir do teorema 15, podemos começar a calcular os tamanhos esperados de mensagens codificadas otimamente a partir de determinados tipos de fonte de dados. Por exemplo, suponha que cada caractere que compõe uma mensagem é escolhido segundo uma lei de probabilidade fixa e independentemente de outros caracteres. Ou seja, a mensagem pode ser vista como a concatenação de variáveis aleatórias $X_1, \ldots, X_n$ independentes e identicamente distribuídas (i.i.d.). Dessa forma, temos interesse em obter os limites superior e inferior do comprimento esperado associados a um código de prefixo ótimo. Note que, como as variáveis em questão são i.i.d., um código de prefixo ótimo para X_i também o é para X_j, $1 \leq i, j \leq n$. Consideremos que C^* é um código de prefixo ótimo para X_i, $1 \leq i \leq n$, e L^* é o comprimento esperado de C^*. Com um pouco de abuso de notação, denominemos por $L^*(X_1 \ldots X_n)$ o comprimento esperado da mensagem $X_1 \ldots X_n$ codificada por C^*, ou seja, $C^*(X_1 \ldots X_n)$, conforme a notação 5. Então, nosso interesse está em limitar o comprimento esperado de $C^*(X_1 \ldots X_n)$. Como $C^*(X_1 \ldots X_n) = C^*(X_1) \ldots C^*(X_n)$, temos que $L^*(X_1 \ldots X_n) = nL^*$. Então, pelo teorema 15, temos

$$nH(X_1) \leq L^*(X_1 \ldots X_n) < n\left[H(X_1) + 1\right]. \tag{2.9}$$

Note que, pela equação (2.9), um código de prefixo ótimo para uma fonte de dados, que gera símbolos independentes e identicamente distribuídos, codifica mensagens de comprimento n em *strings* binárias de comprimento esperado entre nH e $n\left[H + 1\right]$ *bits*, em que H é a entropia associada a uma observação.

O próximo passo desejado para o estudo de códigos de prefixo ótimos é a generalização dos limites do comprimento esperado para codificações de mensagens provenientes de fontes mais sofisticadas que o caso i.i.d. Para isso, o teorema 16 apresenta limites para o caso de processos estocásticos mais gerais.

Teorema 16. *Suponha* $\{X_t\}_{t\in\mathbb{N}}$ *um processo estocástico sobre um alfabeto* A*. Sejam* C^* *um código binário de prefixo ótimo e* L_n^* *o comprimento esperado por caractere codificado por* C^**, ou seja,* $L_n^* = \frac{\mathbb{E}[l^*(X_1 \ldots X_n)]}{n}$*, em que* $l^*(X_1 \ldots X_n)$ *é o comprimento da mensagem*

$X_1 \dots X_n$ codificada por C^*. Então,

$$\frac{H(X_1, \dots, X_n)}{n} \leq L_n^* < \frac{H(X_1, \dots, X_n)}{n} + \frac{1}{n} .$$

Além disso, se $\{X_t\}_{t\in\mathbb{N}}$ for um processo estocástico estacionário, $L_n^* \to H(A)$, em que $H(A)$ é a taxa de entropia do processo.

O teorema 16 nos diz que, em média, $H(A)$ *bits* são suficientes para descrever um símbolo de uma mensagem, desde que esta seja gerada a partir de um processo estacionário com taxa de entropia $H(A)$. Dessa forma, podemos interpretar $H(A)$ como a melhor taxa de compressão possível para o processo $\{X_t\}_{t\in\mathbb{N}}$, ou seja, $H(A)$ é o limite inferior para o número médio de *bits* necessários para descrever uma observação do processo $\{X_t\}_{t\in\mathbb{N}}$.

Pensemos agora na estrutura de um código de prefixo ótimo. Pela definição 27, temos que o comprimento esperado de um código C é da forma $L(C) = \sum_{x\in A} p_X(x)l(x)$. Assim, se queremos minimizar $L(C)$, devemos ter que os elementos mais prováveis do alfabeto possuem uma representação binária não maior que os menos prováveis. Pela restrição de prefixo, temos também que os dois elementos codificados com maior representação binária devem ter o mesmo comprimento. Visando essas características, [18] introduziu um sistema de codificação de prefixo ótimo que viria a ser conhecido como código de Huffman. Essa abordagem permite a construção de um código de prefixo ótimo e um exemplo de construção é apresentado no algoritmo A.2. O exemplo 13 detalha uma execução do algoritmo A.2.

Exemplo 13. *Suponha X uma variável aleatória assumindo valores no alfabeto $A = \{a, b, c, d\}$ com função de probabilidade p_X tal que*

$$p_X(a) = \frac{1}{2}, \quad p_X(b) = \frac{1}{4}, \quad p_X(c) = \frac{7}{32} \quad e \quad p_X(d) = \frac{1}{32}.$$

Denotemos $p_{x_i} = p_X(x_i)$, $C_{x_i} = C(x_i)$ e $l_{x_i} = l(x_i)$, para $x_i \in A$. Inicializemos o algoritmo A.2 com a configuração $\mathbb{P} = \left\{p_a = \frac{1}{2}, p_b = \frac{1}{4}, p_c = \frac{7}{32}, p_d = \frac{1}{32}\right\}$ e $G = \{a, b, c, d\}$.

- *Iteração 1:*

  $I = d$, $J = c$ *e* $H = \{c, d\}$;

  $|H| = 2$, $l_c = 1$ *e* $l_d = 1 \implies C_I = 0$ *e* $C_J = 1 \implies C_d = 0$ *e* $C_c = 1$;

$I' = \{c, d\}$ *e* $p_{I'} = \frac{7}{32} + \frac{1}{32} = \frac{1}{4}$;
$\mathbb{P} = \left\{p_a = \frac{1}{2}, p_b = \frac{1}{4}, p_{\{c,d\}} = \frac{1}{4}\right\}$ *e* $G = \{a, b, \{c, d\}\}$.

- *Iteração 2:*
  $I = \{c, d\}$, $J = b$ *e* $H = \{b, c, d\}$;
  $|H| = 3 \implies C_I = 0C_I$ *e* $C_J = 1C_J \implies C_d = 00$, $C_c = 01$ *e* $C_b = 1$;
  $I' = \{b, c, d\}$ *e* $p_{I'} = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$;
  $\mathbb{P} = \left\{p_a = \frac{1}{2}, p_{\{b,c,d\}} = \frac{1}{2}\right\}$ *e* $G = \{a, \{b, c, d\}\}$.

- *Iteração 3:*
  $I = \{b, c, d\}$, $J = a$ *e* $H = \{a, b, c, d\}$;
  $|H| = 4 \implies C_I = 0C_I$ *e* $C_J = 1C_J \implies C_d = 000$, $C_c = 001$, $C_b = 01$ *e* $C_a = 1$.

Portanto, temos que um código de Huffman para a variável aleatória X é da forma:

$$C(a) = 1,$$
$$C(b) = 01,$$
$$C(c) = 001,$$
$$C(d) = 000.$$

Nota 9. *Perceba que, para uma determinada variável aleatória, não existe apenas uma forma para a codificação de Huffman. Note que, no exemplo 13, podemos intercambiar as codificações $C(c)$ e $C(d)$ e ainda assim obtemos uma codificação de prefixo ótima. Ou também, podemos intercambiar todos os 0's e 1's do código e continuarmos tendo uma codificação de prefixo ótima.*

O algoritmo A.2 apresenta um procedimento de obtenção de uma codificação de prefixo ótima para uma variável aleatória isoladamente. No entanto, na prática, este é raramente o caso. Em geral, trabalhamos com sequências de variáveis aleatórias provenientes de algum processo estocástico $\{X_t\}_{t\in\mathbb{N}}$. Note que, se as variáveis que compõem esse processo forem independentes e identicamente distribuídas, podemos codificar uma mensagem de n elementos do alfabeto original como n codificações isoladas utilizando a mesma distribuição de probabilidade. No entanto, se $\{X_t\}_{t\in\mathbb{N}}$ for um processo estocástico que apresenta uma estrutura de dependência, como ocorre em cadeias de Markov, e

construirmos códigos isolados a partir de uma única distribuição de probabilidade, não estaremos incluindo toda a informação útil que possuímos. Dessa forma, podemos construir um sistema de codificação que leve em conta observações anteriores. Esse sistema será chamado de código condicional e uma conceituação geral é apresentada na definição 28.

Definição 28. *Seja $\{X_t\}_{t\in\mathbb{N}}$ um processo estocástico definido sobre um alfabeto A. Considere A^* como o conjunto de todas as strings de comprimento finito formadas por elementos de A. Seja $g : A^* \to Z$, com $Z = \{1, \ldots, K\}$, uma função que particiona A^* em $K \in \mathbb{N}$ classes. A função g é chamada função de estrutura e cada uma das classes geradas por g é denominada por classe de condicionamento. Além disso, definimos o sistema de codificação $C(\cdot|z)$ como o sistema de codificação restrito à classe z (ou código condicional dado z), $z \in Z$.*

Exemplo 14. *Seja $\{X_t\}_{t\in\mathbb{N}}$ uma cadeia de Markov a tempo discreto de ordem $o = 1$ sobre o alfabeto $A = \{a, b, c\}$. Como discutido no capítulo 1, para uma observação no tempo t, a estrutura de dependência em questão pode ser visualizada como a partição do conjunto de todas as strings de comprimento finito, A^*, baseada na observação no tempo $t-1$. Dessa forma, podemos construir a função de estrutura g e as classes de condicionamento da seguinte forma:*

$$g(x_1^{t-1}) = \begin{cases} 1, & se \quad x_1^{t-1} = \star a, \\ 2, & se \quad x_1^{t-1} = \star b, \\ 3, & se \quad x_1^{t-1} = \star c, \end{cases}$$

em que $\star$ representa uma concatenação de comprimento $t-2$ de elementos de A. Assim, temos A^* particionado em 3 classes, o que nos permitirá construir códigos condicionados aos possíveis valores de $g(x_1^{t-1})$ para qualquer tempo $t > 1$ do processo. Ou seja, para todo $t > 1$, utilizando a função de estrutura do modelo, podemos estabelecer um código condicional $C(x_t|g(x_1^{t-1}))$ que aprecie todo o passado x_1^{t-1}. Além disso, como a imagem da função g se relaciona bijetivamente com a observação no tempo $t-1$, podemos escrever

$$C(x_t|g(x_1^{t-1})) = C(x_t|x_{t-1}). \tag{2.10}$$

Perceba que essa construção permite que a estrutura de dependência delineada para uma cadeia de Markov de ordem $o = 1$ (a informação de todo o passado sobre a

observação atual pode ser resumida na última observação) seja transportada para o sistema de codificação.

Exemplo 15. *Continuemos com as condições do exemplo 14. Suponha que o processo* $\{X_t\}_{t\in\mathbb{N}}$ *é definido pelas probabilidades condicionais*

$$\begin{aligned} p(a\,|\,a) &= 1/4, \quad p(b\,|\,a) = 1/2, \\ p(a\,|\,b) &= 1/8, \quad p(b\,|\,b) = 3/8, \\ p(a\,|\,c) &= 1/4, \quad p(b\,|\,c) = 1/4. \end{aligned}$$

Perceba que, para cada $k \in A = \{a, b, c\}$*, podemos construir um código de Huffman condicional,* $C(\cdot|k)$*, baseado na função de probabilidade condicional* $p(\cdot|k)$*. Dessa forma, aplicando o algoritmo A.2 para cada classe de condicionamento, obtemos o seguinte conjunto de codificações condicionais (com notação seguindo a apresentada na equação (2.10)):*

$$\begin{aligned} C(a\,|\,a) &= 00, \quad C(b\,|\,a) = 1, \quad C(c\,|\,a) = 01, \\ C(a\,|\,b) &= 00, \quad C(b\,|\,b) = 01, \quad C(c\,|\,b) = 1, \\ C(a\,|\,c) &= 00, \quad C(b\,|\,c) = 01, \quad C(c\,|\,c) = 1. \end{aligned}$$

Note que as funções $C(\cdot|b)$ *e* $C(\cdot|c)$ *são equivalentes, mas o mesmo não ocorre com* $C(\cdot|a)$*.*

Os exemplos 14 e 15 ilustram como podemos obter uma codificação para uma mensagem, cujos elementos são obtidos conforme uma cadeia de Markov, tal que o código para cada observação seja de prefixo e possua o menor comprimento esperado possível. Dessa forma, podemos definir um código de prefixo ótimo para um processo como o conjunto das codificações condicionais que permite a otimalidade de codificação em cada tempo do processo, conforme estabelecido na definição 29.

Definição 29. *Seja* $\{X_t\}_{t\in\mathbb{N}}$ *um processo estocástico definido sobre um alfabeto* A*. Sejam* $g : A^* \to Z$ *uma função de estrutura do processo e* $Z = \{1, \ldots, K\}$ *o conjunto de* K *classes de condicionamento geradas por* g*. Um código de prefixo ótimo para o processo* $\{X_t\}_{t\in\mathbb{N}}$ *é o conjunto de funções* $C^* = \{C(\cdot|z) : z \in Z,\ C(\cdot|z)$ *é código de prefixo ótimo restrito à classe* $z\}$*.*

Perceba que, para o exemplo 15, o código de prefixo ótimo para o processo estocástico $\{X_t\}_{t\in\mathbb{N}}$ em questão é o conjunto de funções $C^* = \{C(\cdot|a), C(\cdot|b), C(\cdot|c)\}$.

Esse sistema de codificação permite que uma amostra (mensagem) $X_1 \dots X_n$ do processo $\{X_t\}_{t\in\mathbb{N}}$ tenha cada um de seus elementos X_i codificado de maneira ótima (propriedade de prefixo e menor comprimento esperado de código), para $i > 1$.

Uma outra estratégia para a construção de códigos de prefixo ótimos, mas que não exploraremos neste texto, é a codificação de Huffman por blocos. Nessa abordagem, um número b de elementos são codificados de uma só vez. Dessa forma, se o alfabeto original é A, a codificação deverá ocorrer sobre um alfabeto estendido de A, $\tilde{A} = A^b$, que possui $|A|^b$ elementos. Apesar de, no geral, produzir códigos com menor comprimento esperado por caractere, a codificação por blocos dificultará a aplicação dos procedimentos desenvolvidos para compactação de imagens apresentados no capítulo 4. Por isso, neste texto, exploramos apenas a composição de códigos de Huffman tradicionais (que constituem os códigos instantâneos mais ingênuos) com os resultados do capítulo 1 sobre modelos markovianos de partição. Com alterações apropriadas, os mesmos procedimentos podem ser construídos para códigos de Huffman por blocos e outros sistemas de codificação instantâneos.

2.4 Conclusão

O objetivo deste capítulo foi fornecer subsídio teórico suficiente para os trabalhos de codificação apresentados nos capítulos seguintes. Vimos que, para compreendermos o comportamento de taxas de compressão, devemos, primeiramente, assimilar o conceito de entropia e suas diversas propriedades, como as apresentadas na seção 2.1. Tais propriedades também foram utilizadas para o delineamento de resultados assintóticos expostos na seção 2.2. Um destes, nomeado “alfabeto efetivo”, terá um papel importante durante o processo de compressão de imagens com perda, explorado no capítulo 4. A seção 2.3 apresentou o que desejamos (neste texto) de um sistema de codificação: instantaneidade e mínima representação binária. Nela, delineamos um código que possui tais propriedades, o código de Huffman, e desenvolvemos um algoritmo para obtê-lo (algoritmo A.2). Introduzimos também os códigos de Huffman condicionais, cuja aplicabilidade é justificada no capítulo 3. Além disso, visualizamos como os limitantes de taxas de compressão de códigos ótimos se relacionam com o conceito de entropia e como podemos abordar o problema de compactação no escopo da modelagem de processos estocásticos.

Capítulo 3

Compactação

Neste capítulo, delineamos procedimentos suficientes para que a codificação de *strings* e matrizes, geradas a partir de um alfabeto finito, via códigos de Huffman seja viável. A fim de obtermos representações binárias mais econômicas das mensagens, unimos os conceitos de modelos markovianos de partição (apresentados no capítulo 1) e os resultados da teoria de informação (apresentados no capítulo 2), evidenciando que uma modelagem apropriada da mensagem nos conduz a melhores taxas de compressão. Em [30], essa abordagem foi considerada para a compactação de *strings* cujas leis de probabilidade foram modeladas por cadeias de Markov tradicionais. Aqui, utilizaremos a flexibilidade das cadeias de Markov com interstício para modelar estruturas mais sofisticadas de dependência, como dependências vicinais, conforme introduzido na subseção 1.3.2.

Na seção 3.1, detalhamos o fluxo de informações necessárias (para nosso estudo) entre um transmissor e um receptor a fim de que uma mensagem (*string*) possa ser codificada e unicamente decodificada. Discorremos sobre cada um dos blocos que compõem a *string* binária e disponibilizamos diferentes técnicas para suas produções. Além disso, apresentamos como utilizar os modelos markovianos de partição para obtermos uma representação binária mais compacta da mensagem original. Na subseção 3.1.1, comparamos métodos de compactação via modelos de partição e via cadeias de Markov completas para dados simulados, evidenciando quando um possui vantagens sobre o outro e apontamos sua equivalência assintótica. E, na subseção 3.1.2, apresentamos um comparativo das duas abordagens de compressão para o sequenciamento do genoma da Covid-19 trabalhado no capítulo 1, em que o procedimento que utiliza os modelos markovianos de partição possuiu

melhor desempenho. Na seção 3.2, expomos como incrementar os procedimentos da seção 3.1 para que sejam aplicáveis à codificação de matrizes e mostramos como podemos utilizar as cadeias de Markov com interstício para uma potencial compressão dos dados.

3.1 Compressão de *Strings*

Coloquemo-nos, novamente, no contexto em que um transmissor tem interesse em comunicar uma mensagem (*string*) s, formada por elementos provenientes de um alfabeto A, a um receptor. Neste texto, supomos que essa comunicação só pode ocorrer de maneira binária. Dessa forma, a mensagem transmitida deve ser $C(s)$, em que C é um sistema de codificação unicamente decodificável sobre o alfabeto A, conforme apresentado na definição 25. Perceba que, neste início, já podemos identificar uma informação crítica que o transmissor deve informar ao receptor: qual é o alfabeto A que está sendo considerado. Isso pode ser realizado de diversas maneiras. Neste texto, trabalhamos com duas delas: o método de listagem; e o método de enumeração. Ambas as abordagens partem do pressuposto que o alfabeto A pode ser ordenado. Com isso, o método de listagem transmite uma lista binária que informa se cada elemento de A está presente ou não na mensagem s, enquanto a enumeração transmite as coordenadas dos membros de A existentes em s. Explicações detalhadas sobre estes procedimentos encontram-se em D.1 (apêndice D).

Notação 6. *Denotamos por $I(i, b)$ a função que entrega a representação binária do inteiro não negativo i em um número fixo $b \in \mathbb{N}$ de bits.*

Exemplo 16. *Suponha que transmissor e receptor combinam que todas as mensagens comunicadas serão compostas por elementos provenientes do alfabeto prefixado*

$$A = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}.$$

Considere que uma mensagem s_1 é composta apenas por elementos do alfabeto $A_1 = \{a, e, f, m, r, u, y\}$. Seja $\mathcal{A}_{1,0}$ a string binária que representa a listagem do alfabeto utilizado na mensagem s_1. Repare que o elemento na primeira posição de A é "a", e $a \in A_1$. Dessa forma, o primeiro bit da string $\mathcal{A}_{1,0}$ deve ser 1. O segundo elemento de A é "b", e $b \notin A_1$. Logo, o segundo bit de $\mathcal{A}_{1,0}$ deve ser 0. Prosseguindo para todos os

elementos do alfabeto A, obtemos

$$\mathcal{A}_{1,0} = 10001100000010000100100010.$$

E seja $\mathcal{A}_{1,1}$ *a string binária que representa a enumeração do alfabeto utilizado na mensagem* s_1*. Primeiramente, devemos adaptar o alfabeto prefixado para que comporte o símbolo de parada de leitura "!". Assim, podemos definir* $A^* = \{A, !\}$*. Uma vez que* $\lceil \log(|A^*|) \rceil = 5$*, se considerarmos o primeiro símbolo de* A^* *como o de posição zero, o segundo como o de posição um e assim sucessivamente, podemos informar o alfabeto utilizado como*

$$\begin{aligned}\mathcal{A}_{1,1} &= I(0,5)\, I(4,5)\, I(5,5)\, I(12,5)\, I(17,5)\, I(20,5)\, I(24,5)\, I(26,5) \\ &= 00000\,00100\,00101\,01100\,10001\,10100\,11000\,11010,\end{aligned}$$

pois "a" ocupa a posição zero em A, "e" ocupa a posição quatro, "f" a posição cinco etc. Note que $\mathcal{A}_{1,0}$ *é composta por 26 bits, enquanto* $\mathcal{A}_{1,1}$ *é composta por 40. Agora, considere que uma mensagem* s_2 *é composta apenas por elementos do alfabeto* $A_2 = \{a, e, f\}$*. Se* $\mathcal{A}_{2,0}$ *é a string binária que representa a listagem do alfabeto utilizado na mensagem* s_2*, temos que*

$$\mathcal{A}_{2,0} = 10001100000000000000000000.$$

Se $\mathcal{A}_{2,1}$ *é a string binária que representa a enumeração do alfabeto utilizado na mensagem* s_2*, temos que*

$$\mathcal{A}_{2,1} = I(0,5)\, I(4,5)\, I(5,5)\, I(26,5) = 00000\,00100\,00101\,11010.$$

Note que $\mathcal{A}_{2,0}$ *ainda é composta por 26 bits, enquanto* $\mathcal{A}_{2,1}$ *é composta por 20. Ou seja, dependendo do tamanho do alfabeto utilizado na mensagem, temos que uma representação binária pode ser mais econômica que outra.*

Perceba que o transmissor, antes de comunicar qual o subconjunto do alfabeto prefixado está presente na *string* original, precisa também informar ao receptor qual o método de declaração que será utilizado para codificar tal conjunto (listagem ou enumeração). Uma maneira simples é reservar um *bit* para essa finalidade. Por exemplo, precedendo

a *string* que define o alfabeto podemos alocar um *bit* que assume valor 0 se o método utilizado for listagem e valor 1 se o método for enumeração. No caso do exemplo 16, se desejamos informar que o alfabeto utilizado A_1 será transmitido pelo método de listagem, devemos informar a concatenação $0\mathcal{A}_{1,0}$, ou seja, a *string* binária

$$\mathcal{A}^* = 0\mathcal{A}_{1,0} = 0100011000000100001001000010.$$

De maneira equivalente, se desejamos informar que o alfabeto utilizado A_2 será transmitido pelo método de enumeração, devemos informar a concatenação $1\mathcal{A}_{2,1}$, ou seja, a *string* binária

$$\mathcal{A}^* = \mathcal{A}_{2,1} = 1\,00000\,00100\,00101\,11010.$$

Com isso, estabelecemos, sem ambiguidade, uma maneira de comunicar binariamente a composição do alfabeto sobre o qual uma mensagem é composta.

Outra informação crucial para a reconstrução da mensagem é o conhecimento, por parte do receptor, sobre o sistema de codificação utilizado pelo transmissor para codificar os dados originais. Uma grande quantidade de sistemas de codificação foram propostos na literatura ao longo dos anos, como a codificação de Huffman apresentada no capítulo 2, codificações de Shannon-Fano-Elias (precursoras do código aritmético, veja [24]) e codificações de Lempel-Ziv (veja [32]), que são amplamente utilizadas para compressão de imagens, confira, por exemplo, [21]. Neste texto, trabalhamos exclusivamente com codificações de Huffman, em particular, codificações de Huffman condicionais, conforme ilustradas no exemplo 15. Dessa forma, não precisamos incluir na *string* transmitida a informação sobre qual o método de codificação escolhido, mas toda a informação sobre a classe de condicionamento considerada e os parâmetros utilizados para a construção do código empregado. Como discutido no capítulo 1, cadeias de Markov com interstício são capazes de modelar estruturas de dependência sofisticadas que abrangem as cadeias de Markov completas. Por esse motivo, utilizamos a estrutura de dependência de cadeias de Markov com interstício para construir as classes de condicionamento que serão utilizadas na elaboração de um código de prefixo ótimo para um processo, conforme estabelecido na definição 29. Dessa forma, transmissor e receptor estão de acordo que a amostra do processo (mensagem original) foi codificada utilizando codificações de Huffman condicionais,

tendo as classes de condicionamento determinadas por modelos de cadeias de Markov com interstício. Assim, são necessárias as informações de ordem imediata, M, interstício, G, e ordem de interstício, g.

Nota 10. *De acordo com a definição 12, uma cadeia de Markov com interstício com* $M = 0$, $G = 0$ *e* $g = 0$ *representa um processo com observações independentes.*

Uma maneira simples de se transmitir o modelo utilizado é reservando um número fixo de *bits* para cada uma das quantidades M, G, g. Em particular, é interessante que, ao invés de se transmitir a tripla (g, G, M), seja transmitida a tripla $(g, G - M, M)$, pois $G - M$ é um inteiro menor que G, requisitando um menor número de *bits* para sua representação binária, e permite que o modelo original seja estabelecido apropriadamente.

Exemplo 17. *Suponha que transmissor e receptor combinam que a comunicação do modelo utilizado para a codificação dos dados será dada pela concatenação da representação binária da tripla* $(g, G - M, M)$*, em que serão reservados 4 bits para o valor de* g*, 8 bits para o de* $G - M$ *e 4 bits para o de* M*. Considere que, para a codificação de certa mensagem, a classe de condicionamento utilizada foi a de uma cadeia de Markov com interstício com* $M = 6$, $G = 250$ *e* $g = 2$*. Assim, o modelo utilizado é transmitido como a string binária*

$$\mathcal{E} = I(g, 4)\, I(G - M, 8)\, I(M, 4) = I(2, 4)\, I(244, 8)\, I(6, 4) = 0010\,11110100\,0110. \quad (3.1)$$

Dessa forma, ao receber a string binária apresentada na equação (3.1), o decodificador saberá, sem ambiguidade, qual o modelo utilizado.

No exemplo 17, repare que um número maior de *bits* foi reservado para o valor de interstício G do que para M e g. Essa decisão é razoável, pois a complexidade do modelo (número de parâmetros livres) baseado em uma cadeia de Markov com interstício cresce exponencialmente em M e g, mas não em G, uma vez que o espaço de estados associado a esse processo é da forma $\mathcal{S}^* = A^{g+1} \times A^M$. Assim, mesmo para valores grandes de G, a estimação dos parâmetros do modelo ainda é factível.

Se retornarmos ao algoritmo A.2 e ao exemplo 15, temos que, para determinar plenamente o código de Huffman utilizado, o transmissor deve informar ao receptor o conjunto de probabilidades utilizado. Para a codificação de uma única *string* de comprimento finito, podemos estabelecer simplesmente que $\mathbb{P} = \widehat{\mathbb{P}} = \{\widehat{p}_1, \ldots, \widehat{p}_{|A|}\}$, em

que $\widehat{p}_i$ é a frequência relativa do i-ésimo elemento do alfabeto A na mensagem original, $i \in \{1, \ldots, |A|\}$. Isso nos garantirá que, para esta exata mensagem, a codificação de Huffman obtida seja um código de prefixo ótimo. No caso de códigos condicionais, o conjunto de probabilidades a ser transmitido deve incluir todas as possibilidades da classe de condicionamento, ou seja, $\mathbb{P} = \{\widehat{p}(a|z) : a \in A, z \in Z\}$, em que $Z = \{1, \ldots, K\}$ é o conjunto das classes de condicionamento e $\widehat{p}(a|z)$ é a frequência relativa do elemento $a \in A$ condicionado à classe $z \in Z$ (pelo exemplo 14, as classes de condicionamento z estão associadas à porção relevante do passado no caso em que a função de estrutura do processo é oriunda de uma cadeia de Markov, portanto, há sentido na escrita $\widehat{p}(a|z)$). Além disso, repare que as quantidades $\widehat{p}_i$ são valores entre 0 e 1, necessitando (em alguns casos) alta precisão ou arredondamentos para serem adequadamente representados. Essa abordagem acarreta em uma extensa representação binária, tornando a transmissão da mensagem mais custosa. No entanto, observe que, no algoritmo A.2, a multiplicação dos elementos do conjunto de probabilidades $\mathbb{P}$ por uma constante não afeta o sistema de codificação final obtido. Em particular, se denominarmos por n o comprimento da *string* (mensagem) original, o conjunto de probabilidades $\{\widehat{p}_1, \ldots, \widehat{p}_{|A|}\}$ nos leva ao mesmo código de Huffman que o conjunto $\mathcal{N} = \{n\widehat{p}_1, \ldots, n\widehat{p}_{|A|}\}$. Repare que $\mathcal{N}$ não é mais um conjunto de frequências relativas, mas de frequências absolutas e pode ser denotado por $\mathcal{N} = \{n_1 \ldots, n_{|A|}\}$, em que $n_i = n\widehat{p}_i$ é a frequência absoluta do i-ésimo elemento do conjunto A na mensagem original, $i \in \{1, \ldots, |A|\}$. Como números inteiros possuem representação binária mais simples, a comunicação sobre o sistema de codificação entre transmissor e receptor se torna mais eficiente. Ainda assim, algumas estratégias podem ser adotadas para a transmissão do conjunto $\mathcal{N}$.

Da mesma maneira realizada para a definição do alfabeto e para a declaração do modelo utilizado, consideramos que transmissor e receptor têm preestabelecido qual o número máximo de elementos, N, que uma mensagem pode ter. Com isso, podemos transmitir o conjunto das contagens $\mathcal{N}$ de diversas maneiras. Neste texto, trabalhamos com duas delas: Modo 0; e Modo 1. O Modo 0 comunica cada frequência observada utilizando um número fixo de *bits*, enquanto o Modo 1 possibilita o uso de comprimentos variáveis. Explicações detalhadas sobre estes procedimentos encontram-se em D.2.

Exemplo 18. *Suponha que, em uma determinada mensagem de comprimento* $n = 500$*, foram utilizados os símbolos do alfabeto* $A = \{a, b, c\}$ *e observadas as frequências* $n_a = 300$*,*

$n_b = 150$ *e* $n_c = 50$. *Estamos interessados em transmitir essas frequências. Considere que transmissor e receptor combinaram previamente qual o alfabeto* A *utilizado e que a maior mensagem que poderá ser comunicada possuirá* $N = 2^{10} - 1 = 1.023$ *elementos de* A. *Pelo Modo 0, a string binária que representa as frequências de interesse é*

$$\begin{aligned}\mathcal{D}_0 &= I(n_a, \lfloor \log N \rfloor + 1)\, I(n_b, \lfloor \log N \rfloor + 1)\, I(n_c, \lfloor \log N \rfloor + 1) \\ &= I(300, 10)\, I(150, 10)\, I(50, 10) \\ &= 0100101100\ 0010010110\ 0000110010.\end{aligned}$$

Para o Modo 1, iniciamos computando o conjunto $\mathcal{N}_B = \{I(n_a, \lfloor \log n_a \rfloor + 1), I(n_b, \lfloor \log n_b \rfloor + 1), I(n_c, \lfloor \log n_c \rfloor + 1)\}$. *Neste caso,* $\mathcal{N}_B = \{100101100,\ 10010110,\ 110010\}$. *Em seguida, construímos o conjunto dos comprimentos dos elementos de* $\mathcal{N}_B$, $\mathcal{J} = \{j_a, j_b, j_c\} = \{9, 8, 6\}$ *e calculamos* $j* = \max_{j_i \in \mathcal{J}} j_i = 9$. *Com isso, a string binária que representa as frequências de interesse é*

$$\begin{aligned}\mathcal{D}_1 = {} & I(\lfloor \log j^* \rfloor + 1, \lfloor \log(\lfloor \log N \rfloor + 1) \rfloor + 1)\ I(j_a, \lfloor \log j^* \rfloor + 1)\, I(n_a, j_a) \\ & I(j_b, \lfloor \log j^* \rfloor + 1) I(n_b, j_b)\ I(j_c, \lfloor \log j^* \rfloor + 1) I(n_c, j_c) \\ = {} & I(4, 4)\ I(9, 4) I(300, 9)\ I(8, 4) I(150, 8)\ I(6, 4) I(50, 6) \\ = {} & 0100\ 1001\ 100101100\ 1000\ 10010110\ 0110\ 110010.\end{aligned}$$

Repare que $\mathcal{D}_0$ *possui comprimento de 30 bits, enquanto* $\mathcal{D}_1$ *de 39 bits.*

Exemplo 19. *Consideremos as condições do exemplo 18, mas assumamos que a mensagem original possui frequências* $n_a = 494$, $n_b = 3$ *e* $n_c = 3$. *Seguindo a construção realizada anteriormente, pelo Modo 0,*

$$\mathcal{D}_0 = I(494, 10)\, I(3, 10)\, I(3, 10) = 0111101110\ 0000000011\ 0000000011.$$

Pelo Modo 1, $\mathcal{N}_B = \{111101110,\ 11,\ 11\}$, $\mathcal{J} = \{9, 2, 2\}$ *e* $j^* = 9$. *Então,*

$$\begin{aligned}\mathcal{D}_1 = {} & I(4, 4)\ I(9, 4) I(494, 9)\ I(2, 4) I(3, 2)\ I(2, 4) I(3, 2) \\ = {} & 0100\ 1001\ 111101110\ 0010\ 11\ 0010\ 11.\end{aligned}$$

Repare que $\mathcal{D}_0$ *ainda possui comprimento de 30 bits, enquanto* $\mathcal{D}_1$ *de 29 bits.*

Exemplo 20. *Consideremos as condições do exemplo 18, mas assumamos que a mensagem original é constituída por elementos do alfabeto* $A = \{a, b, c, d, e, f\}$ *e com frequências* $n_a = 180$, $n_b = 150$, $n_c = 80$, $n_d = 50$, $n_e = 30$ *e* $n_f = 10$. *Seguindo a construção realizada anteriormente, pelo Modo 0,*

$$\begin{aligned}\mathcal{D}_0 &= I(180,10)\, I(150,10)\, I(80,10)\, I(50,10)\, I(30,10)\, I(10,10)\\ &= 0010110100\, 0010010110\, 0001010000\, 0000110010\, 0000011110\, 0000001010.\end{aligned}$$

Pelo Modo 1, $\mathcal{N}_B = \{10110100,\ 10010110,\ 1010000,\ 110010,\ 11110,\ 1010\}$, $\mathcal{J} = \{8, 8, 7, 6, 5, 4\}$ *e* $j^* = 8$. *Então,*

$$\begin{aligned}\mathcal{D}_1 &= I(4,4)\ I(8,4)I(180,8)\ I(8,4)I(150,8)\ I(7,4)I(80,7)\\ &\quad I(6,4)I(50,6)\ I(5,4)I(30,5)\ I(4,4)I(10,4)\\ &= 0100\ 1000\, 10110100\ 1000\, 10010110\ 0111\, 1010000\ 0110\, 110010\ 0101\, 11110\ 0100\, 1010.\end{aligned}$$

Repare que $\mathcal{D}_0$ *possui comprimento de 60 bits, enquanto* $\mathcal{D}_1$ *de 66 bits.*

Exemplo 21. *Consideremos as condições do exemplo 20, mas assumamos que a mensagem original possui frequências* $n_a = 220$, $n_b = 220$, $n_c = 30$, $n_d = 20$, $n_e = 7$ *e* $n_f = 3$. *Seguindo a construção realizada anteriormente, pelo Modo 0,*

$$\begin{aligned}\mathcal{D}_0 &= I(220,10)\, I(220,10)\, I(30,10)\, I(20,10)\, I(7,10)\, I(3,10)\\ &= 0011011100\, 0011011100\, 0000011110\, 0000010100\, 0000000111\, 0000000011.\end{aligned}$$

Pelo Modo 1, $\mathcal{N}_B = \{11011100,\ 11011100,\ 11110,\ 10100,\ 111,\ 11\}$, $\mathcal{J} = \{8, 8, 5, 5, 3, 2\}$ *e* $j^* = 8$. *Então,*

$$\begin{aligned}\mathcal{D}_1 &= I(4,4)\ I(8,4)I(220,8)\ I(8,4)I(220,8)\ I(5,4)I(30,5)\\ &\quad I(5,4)I(20,5)\ I(3,4)I(7,3)\ I(2,4)I(3,2)\\ &= 0100\ 1000\, 11011100\ 1000\, 11011100\ 0101\, 11110\ 0101\, 10100\ 0011\, 111\ 0010\, 11.\end{aligned}$$

Repare que $\mathcal{D}_0$ *ainda possui comprimento de 60 bits, enquanto* $\mathcal{D}_1$ *de 59 bits.*

Exemplo 22. *Consideremos as condições do exemplo 18, mas assumamos que a mensagem original possui comprimento* $n = 100$ *e frequências* $n_a = 60$, $n_b = 30$ *e* $n_c = 10$. *Seguindo*

a construção realizada anteriormente, pelo Modo 0,

$$\mathcal{D}_0 = I(60,10)\, I(30,10)\, I(10,10) = 0000111100\ 0000011110\ 0000001010.$$

Pelo Modo 1, $\mathcal{N}_B = \{111100,\ 11110,\ 1010\}$, $\mathcal{J} = \{6,5,4\}$ *e* $j^* = 6$. *Então,*

$$\begin{aligned}\mathcal{D}_1 &= I(3,4)\ I(6,3) I(60,6)\ I(5,3) I(30,5)\ I(4,3) I(10,4)\\ &= 0011\ 110\,111100\ 101\,11110\ 100\,1010.\end{aligned}$$

Repare que $\mathcal{D}_0$ *ainda possui comprimento de 30 bits, enquanto* $\mathcal{D}_1$ *de 28 bits.*

Pelos exemplos 18, 19, 20, 21 e 22, podemos notar que o Modo 0 não é uniformemente mais econômico que o Modo 1 e vice-versa. O método preferível varia conforme o tamanho do alfabeto A considerado, a distribuição do conjunto de frequências $\mathcal{N}$ e o comprimento n da mensagem em relação ao limitante superior prefixado N. Por isso, é interessante que cada caso seja avaliado individualmente e, analogamente ao método de seleção de alfabetos, reservamos um *bit* para a comunicação de qual modo foi utilizado. Dessa forma, estabelecemos que a concatenação $\mathcal{D}^* = 0\mathcal{D}_0$ informa as frequências pelo Modo 0, e a concatenação $\mathcal{D}^* = 1\mathcal{D}_1$ informa as frequências pelo Modo 1.

Para a transmissão de uma mensagem, a comunicação pode ser dividida em dois blocos:

- Cabeçalho, que consiste do conjunto de informações necessárias para a compreensão do sistema de codificação utilizado;
- Mensagem codificada, que consiste da *string* binária resultante do código (identificado no cabeçalho) aplicado à mensagem original.

Para nosso caso, em que utilizamos códigos de Huffman para codificar uma *string*, temos que o cabeçalho é formado pela declaração do alfabeto, definição do modelo utilizado para a classe de condicionamento e parâmetros utilizados (frequências observadas). Com essas informações, podemos estabelecer de maneira apropriada um sistema de codificação, conforme já ilustrado nos exemplos 13 e 15.

Exemplo 23. *Suponha que transmissor e receptor já tenham estabelecido previamente o seguinte conjunto de configurações:*

- *Alfabeto prefixado:* $A = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$*;*
- *Bits reservados para a definição do modelo para a classe de condicionamento:* 4 *bits para o valor* g*,* 8 *bits para* $G - M$ *e* 4 *bits para* M*;*
- *Comprimento da maior mensagem a ser transmitida:* $N = 2^{10} - 1 = 1.023$.

Considere que desejamos comunicar a mensagem T*, tal que*

$T =$ *waaaawhwhaaahhaaahaaaawaawawwahahwwhwaahaahahwhaawwwaawhhhhwa hahhaahhwawaahahwhwwhaaaaaaahaaaawwhhhwaawaawahahhawwhwaahwaah hwhaahhaawaaawawahhwhahwhwwhhhhwawaawawahahaaawwaawaawhahhhha hhahhhaawwaaaaww.*

A mensagem T *possui comprimento* $n = 200$*, alfabeto utilizado* $A_1 = \{a, h, w\}$ *e frequências* $\mathcal{N} = \{n_a = 90, n_h = 60, n_w = 50\}$*. Suponha que o transmissor julga conveniente que as classes de condicionamento do modelo sejam as de uma cadeia de Markov com interstício com* $M = 0$*,* $G - M = 0$ *e* $g = 0$*, ou seja, as observações são tratadas como provenientes de um processo independente.*

Temos interesse em definir uma única string binária, $\mathcal{U}$*, que contenha toda a informação do cabeçalho necessário para a transmissão da mensagem* T*, ou seja,* $\mathcal{U}$ *deve ser tal que* $\mathcal{U} = \mathcal{A}^*\mathcal{E}\mathcal{D}^*$*, em que* $\mathcal{A}^*$*,* $\mathcal{E}$ *e* $\mathcal{D}^*$ *são strings binárias que representam, respectivamente, o alfabeto utilizado, o modelo selecionado para as classes de condicionamento e o conjunto de frequências observadas. A seguir, descrevemos os procedimentos já trabalhados nos exemplos 16 a 22:*

- *Declaração do alfabeto:*
  1. *Listagem: o elemento "a" ocupa a posição zero em* A*, o elemento "h" a posição sete e o elemento "w" a posição vinte e dois. Assim, a string binária correspondente é*

$$\mathcal{A}_{1,0} = 10000001000000000000001000;$$

  2. *Enumeração: como* $\lceil \log(|A| + 1) \rceil = \lceil \log 27 \rceil = 5$*, temos que a string binária*

correspondente é

$$\mathcal{A}_{1,1} = I(0,5)I(7,5)I(22,5)I(26,5) = 00000\ 00111\ 10110\ 11010.$$

Note que o método da listagem utiliza 26 bits enquanto o da enumeração utiliza 20. Dessa forma, como o método da enumeração é mais econômico, o escolhemos e declaramos a escolha pelo bit de identificação. Dessa forma, estabelecemos o alfabeto utilizado na mensagem pela string binária $\mathcal{A}^* = 1\mathcal{A}_{1,1}$*;*

- *Declaração do modelo para classe de condicionamento: utilizando os bits reservados pela configuração, temos que o modelo é representado pela string binária*

$$\mathcal{E} = I(0,4)I(0,8)I(0,4) = 0000\ 00000000\ 0000;$$

- *Declaração das contagens: como o modelo utilizado para as classes de condicionamento é o de um processo independente, devemos apenas transmitir o conjunto de frequências* $\mathcal{N} = \{n_a, n_h, n_w\}$.

  1. *Modo 0: como* $\lfloor \log N \rfloor + 1 = \lfloor \log 1.023 \rfloor + 1 = 10$*, temos que a string binária correspondente é*

$$\mathcal{D}_0 = I(90,10)I(60,10)I(50,10) = 0001011010\,0000111100\,0000110010.$$

  2. *Modo 1: temos que* $\mathcal{N}_B = \{I(n_a, \lfloor \log n_a \rfloor + 1), I(n_h, \lfloor \log n_h \rfloor + 1), I(n_w, \lfloor \log n_w \rfloor + 1)\} = \{1011010,\ 111100,\ 110010\}$*,* $\mathcal{J} = \{\lfloor \log n_a \rfloor + 1, \lfloor \log n_h \rfloor + 1, \lfloor \log n_w \rfloor + 1\} = \{7, 6, 6\}$ *e* $j^* = \max_{j_i \in \mathcal{J}} j_i = 7$*. Então,*

$$\begin{aligned}\mathcal{D}_1 &= I(3,4)\, I(7,3)I(90,7)\, I(6,3)I(60,6)\, I(6,3)I(50,6)\\ &= 0011\ 111\,1011010\ 110\,111100\ 110\,110010.\end{aligned}$$

Note que o Modo 0 utiliza 30 bits enquanto o Modo 1 utiliza 32. Dessa forma, como o Modo 0 é mais econômico, o escolhemos e declaramos a escolha pelo bit de identificação. Dessa forma, estabelecemos as contagens utilizadas para a construção do código de Huffman pela string binária $\mathcal{D}^* = 0\mathcal{D}_0$.

Com isso, podemos concatenar as strings binárias $\mathcal{A}^*$, $\mathcal{E}$ *e* $\mathcal{D}^*$ *para formar uma única string* $\mathcal{U} = \mathcal{A}^*\mathcal{E}\mathcal{D}^*$ *que representa o cabeçalho de nossa mensagem. Agora, necessitamos construir o código que será utilizado para transmitir a mensagem original. Para isso, basta aplicarmos o algoritmo A.2 com* $A = A_1 = \{a, h, w\}$ *e* $\mathbb{P} = \mathcal{N} = \{n_a, n_h, n_w\}$. *Dessa forma, um código de Huffman é*

$$C(a) = 0,\ C(h) = 11\ e\ C(w) = 10. \tag{3.2}$$

Denominemos por $C(T)$ *a codificação apresentada em (D.1) aplicada em cada elemento da mensagem* T. *Como observamos os comprimentos de código* $l_a = 1$, $l_h = 2$ *e* $l_w = 2$, *temos que a mensagem codificada possui*

$$90 \times 1 + 60 \times 2 + 50 \times 2 = 310\ \textit{bits}.$$

Por fim, podemos realizar a concatenação $\mathcal{T} = \mathcal{U}C(T)$, *que une o cabeçalho e a mensagem codificada. Dessa forma, ao ser transmitida a string binária* $\mathcal{T}$, *um receptor conseguirá recuperar a mensagem original* T *de maneira única. Além disso, como o cabeçalho* $\mathcal{U}$ *é composto por* $21 + 16 + 31 = 68$ *bits, temos que a transmissão da mensagem* T *(de 200 caracteres) necessitou de* $68 + 310 = 378$ *bits para ser realizada.*

O exemplo 23 trata a mensagem original como a visualização de um processo estocástico $\{X_t\}_{t\in\mathbb{N}}$, cuja estrutura de dependência entre observações é a independência. Se utilizarmos a notação introduzida na definição 28, a função de estrutura g particiona o processo em apenas uma classe de condicionamento, ou seja, $g : A^* \to Z = \{1\}$. Por esse motivo, foi necessário apenas um código de Huffman para descrever o sistema de codificação a ser utilizado em toda a mensagem do exemplo 23.

Agora, temos interesse em transmitir mensagens utilizando códigos condicionais cuja função de estrutura pode ser derivada de uma cadeia de Markov qualquer. Os exemplos 14 e 15 apresentaram uma construção de código de prefixo ótimo para um processo $\{X_t\}_{t\in\mathbb{N}}$ quando este foi modelado por uma cadeia de Markov de ordem $o = 1$. Note que, no exemplo 15, foram necessárias todas as probabilidades condicionais $p(x|s)$, em que $x \in A$ e $s \in \mathcal{S} = A$, $\mathcal{S}$ espaço de estados, para se construir o conjunto dos códigos de prefixo ótimos. Dessa forma, se desejamos utilizar a estrutura de dependência da nossa

fonte de dados para otimizar sua codificação, devemos também transmitir esse conjunto de probabilidades (ou contagens) durante a comunicação da mensagem.

Um último detalhe antes da construção de um novo exemplo se refere aos primeiros elementos da mensagem que desejamos transmitir. No caso da modelagem do processo por uma cadeia de Markov de ordem o, as o primeiras observações da mensagem não podem ter sua classe de condicionamento bem definida. Por esse motivo, devem ser tratadas de maneira diferenciada. Este fenômeno é análogo ao que ocorre na função de verossimilhança de uma cadeia de Markov de ordem o, em que as o primeiras observações são consideradas apenas marginalmente, conforme apresentado na equação (1.1). Para essas observações, consideramos duas abordagens. A primeira delas é codificar esses elementos a partir de um código de Huffman para observações independentes, sendo necessária também a transmissão das contagens que o delinearam. A segunda é a definição prévia de uma codificação para o alfabeto prefixado e que será utilizada nessas observações iniciais. Dessa maneira, cada elemento do alfabeto utilizado A_1 é codificado em uma *string* binária única de comprimento $\lceil \log |A_1| \rceil$ *bits*. A segunda abordagem é vantajosa quando o número o de observações iniciais ou o tamanho do alfabeto utilizado A_1 forem pequenos. Como no próximo capítulo trabalharemos com valores de o grandes e A_1 de comprimento médio, consideramos apenas a primeira abordagem apresentada.

Nota 11. *Repare que em momento algum estamos assumindo que transmissor e codificador combinam uma distribuição prévia sobre o alfabeto prefixado ou sobre a fonte de dados. Isso é feito para que o procedimento de codificação seja o mais universal possível.*

Exemplo 24. *Consideremos as configurações prefixadas do exemplo 23 e que temos interesse em transmitir novamente uma mensagem T cujos elementos são provenientes do alfabeto $A_1 = \{a, h, w\}$. Desta vez, suponhamos que as observações da string T se comportam como uma cadeia de Markov de ordem $o = 2$. As probabilidades de transição estão organizadas em três grupos baseados no espaço de estados $\mathcal{S} = A_1^2$: $L_1 = \{aa, hh, ww\}$; $L_2 = \{ha, wa, ah, wh, hw\}$; e $L_3 = \{aw\}$. Dessa forma,*

- *para $s \in L_1$,*

$$p(a|s) = 0,1, \quad p(h|s) = 0,2 \quad e \quad p(w|s) = 0,7;$$

- *para* $s \in L_2$,

$$p(a|s) = 0,4, \quad p(h|s) = 0,3 \quad e \quad p(w|s) = 0,3;$$

- *para* $s \in L_3$,

$$p(a|s) = 0,8, \quad p(h|s) = 0,1 \quad e \quad p(w|s) = 0,1.$$

Simulamos uma mensagem T *de comprimento* $n = 200$ *seguindo as configurações estabelecidas. Por simplicidade, as* $o = 2$ *primeiras observações foram obtidas de maneira independente a partir de uma distribuição uniforme em* A_1*. A mensagem simulada foi*

$T =$ *wahhhwwwwwhaahawahhaawahahhhwaaawaawaahhwhahhwhawawwwawaw aahhwhwaawaahawahawawwaaahwaawhhwhawwwwwwwhwhaawahwaawaaw aaawahaawaawwwwwhhwwwwwhaawahhwaawwhaaahhaaawawaawawawawaa hawaahhhwwwwwaawaawahw.*

Pelo já exposto no exemplo 23, temos que o alfabeto é comunicado pelo método de enumeração, como $\mathcal{A}^* = 1\mathcal{A}_1$*, em que*

$$\mathcal{A}_1 = 00000\ 00111\ 10110\ 11010.$$

Consideremos que, de fato, o transmissor irá utilizar uma cadeia de Markov de ordem $o = 2$ *para descrever as classes de condicionamento. Dessa forma, deve comunicar a escolha dos parâmetros escolhidos, ou seja,* $g = 0$*,* $G - M = 1$ *e* $M = 1$ *para uma cadeia de Markov com interstício (modelagem preestabelecida). Essa comunicação é feita pela string binária*

$$\mathcal{E} = I(0,4)I(1,8)I(1,4) = 0000\ 00000001\ 0001.$$

Devemos transmitir também as informações necessárias para a construção do código condicional utilizado pelo transmissor, ou seja, devemos transmitir o conjunto de contagens restritas a cada elemento da classe de condicionamento, neste caso, o próprio espaço de estados $\mathcal{S} = A_1^2$*. Consideremos os conjuntos de contagens condicionais* $\mathcal{N}_s = \{n_j^{(s)} : j \in A_1\}$*, em que* $n_j^{(s)} = |\{t > o : T_{t-2}T_{t-1} = s, T_t = j\}|$ *e* T_k *é o* k*-ésimo elemento da string* T*, para* $s \in \mathcal{S}$*. Com isso, temos os conjuntos de contagens condicionais*

$\mathcal{N}_{aa} = \{n_a^{(aa)} = 5,\, n_h^{(aa)} = 8,\, n_w^{(aa)} = 17\}$, $\mathcal{N}_{ha} = \{n_a^{(ha)} = 7,\, n_h^{(ha)} = 2,\, n_w^{(ha)} = 6\}$,
$\mathcal{N}_{wa} = \{n_a^{(wa)} = 18,\, n_h^{(wa)} = 8,\, n_w^{(wa)} = 8\}$, $\mathcal{N}_{ah} = \{n_a^{(ah)} = 6,\, n_h^{(ah)} = 9,\, n_w^{(ah)} = 3\}$,
$\mathcal{N}_{hh} = \{n_a^{(hh)} = 2,\, n_h^{(hh)} = 3,\, n_w^{(hh)} = 9\}$, $\mathcal{N}_{wh} = \{n_a^{(wh)} = 7,\, n_h^{(wh)} = 2,\, n_w^{(wh)} = 2\}$,
$\mathcal{N}_{aw} = \{n_a^{(aw)} = 25,\, n_h^{(aw)} = 1,\, n_w^{(aw)} = 5\}$, $\mathcal{N}_{hw} = \{n_a^{(hw)} = 5,\, n_h^{(hw)} = 5,\, n_w^{(hw)} = 3\}$,
$\mathcal{N}_{ww} = \{n_a^{(ww)} = 3,\, n_h^{(ww)} = 5,\, n_w^{(ww)} = 24\}$.

Assim como no exemplo 23, precisamos decidir qual o Modo utilizado para se transmitir cada um dos grupos de frequências condicionais. Para cada grupo, pelo Modo 0, são necessários 3×10 *bits, conforme o já exposto no exemplo 23. Para o Modo 1, temos*

$$\begin{aligned}
\mathcal{D}_{1,aa} &= I(3,4)\, I(3,3) I(5,3)\, I(4,3) I(8,4)\, I(5,3) I(17,5) \\
&= 0011\ 011\,101\ 100\,1000\ 101\,10001, \\
\mathcal{D}_{1,ha} &= I(2,4)\, I(3,2) I(7,3)\, I(2,2) I(2,2)\, I(3,2) I(6,3) = 0010\ 11\,111\ 10\,10\ 11\,110, \\
\mathcal{D}_{1,wa} &= I(3,4)\, I(5,3) I(18,5)\, I(3,3) I(7,3)\, I(4,3) I(8,4) \\
&= 0011\ 101\,10010\ 011\,111\ 100\,1000, \\
\mathcal{D}_{1,ah} &= I(3,4)\, I(3,3) I(6,3)\, I(4,3) I(9,4)\, I(2,3) I(3,2) = 0011\ 011\,110\ 100\,1001\ 010\,11, \\
\mathcal{D}_{1,hh} &= I(3,4)\, I(2,3) I(2,2)\, I(2,3) I(3,2)\, I(4,3) I(9,4) = 0011\ 010\,10\ 010\,11\ 100\,1001, \\
\mathcal{D}_{1,wh} &= I(2,4)\, I(3,2) I(7,3)\, I(2,2) I(2,2)\, I(2,2) I(2,2) = 0010\ 11\,111\ 10\,10\ 10\,10, \\
\mathcal{D}_{1,aw} &= I(3,4)\, I(5,3) I(25,5)\, I(1,3) I(1,1)\, I(3,3) I(5,3) = 0011\ 101\,11001\ 001\,1\ 011\,101, \\
\mathcal{D}_{1,hw} &= I(2,4)\, I(3,2) I(5,3)\, I(3,2) I(5,3)\, I(2,2) I(3,2) = 0010\ 11\,101\ 11\,101\ 10\,11, \\
\mathcal{D}_{1,ww} &= I(3,4)\, I(2,3) I(3,2)\, I(3,3) I(5,3)\, I(5,3) I(24,5) = 0011\ 010\,11\ 011\,101\ 101\,11000.
\end{aligned}$$

Note que $\mathcal{D}_{1,s}$ *possui menos de 30 bits, para todo* $s \in \mathcal{S}$*. Dessa forma, utilizamos para cada grupo um bit de identificação do Modo escolhido e obtemos* $\mathcal{D}^*_{1,s} = 1\mathcal{D}_{1,s}$*. Para encerrarmos a declaração dessas contagens, concatenamos, em uma ordem preestabelecida (lexicográfica, por exemplo), as strings* $\mathcal{D}^*_{1,s}$*,* $\forall s \in \mathcal{S}$*, e denominamos o resultado por* $\mathcal{D}^*_1$*.*

Também precisamos transmitir o conjunto de frequências utilizado para a codificação dos $o = 2$ *primeiros elementos de* T*. Neste caso, isso é simples, pois temos o conjunto de frequências* $\mathcal{N} = \{n_a = 1,\, n_h = 0,\, n_w = 1\}$*. Aqui, o Modo 0 pode ser adaptado, pois, para as* o *primeiras observações, o pior dos casos (frequências acumulados em um único valor) é um elemento com frequência* o*, que necessitaria de* $\lfloor \log o \rfloor + 1$ *bits para ser representado. Portanto, o Modo 0 representa as contagens do conjunto* $\mathcal{N}$ *em*

$3 \times (\lfloor \log 2 \rfloor + 1) = 6$ *bits, pela string binária*

$$\mathcal{D}_{2,0} = I(1,2)I(0,2)I(1,2) = 01\,00\,01.$$

Em contrapartida, utilizando a mesma tática do pior dos casos, temos, para o Modo 1, $\mathcal{N}_B = \{1,0,1\}$, $\mathcal{J} = \{1,1,1\}$ *e* $j^* = 1$, *o que produz a representação binária*

$$\mathcal{D}_{2,1} = I(1,2)\ I(1,1)I(1,1)\ I(1,1)I(0,1)\ I(1,1)I(1,1) = 01\ 1\,1\ 1\,0\ 1\,1,$$

que possui comprimento de 8 bits. Como o Modo 0 é mais econômico, o escolhemos e declaramos a escolha pelo bit de identificação. Dessa forma, estabelecemos as frequências dos o primeiros elementos pela string binária $\mathcal{D}_2^* = 0\mathcal{D}_{2,0}$. *Por fim, realizamos a concatenação* $\mathcal{D}^* = \mathcal{D}_1^*\mathcal{D}_2^*$, *que representa todas as frequências utilizadas para construção de codificações de Huffman para toda a mensagem* T.

Com isso, podemos concatenar as strings binárias $\mathcal{A}^*$, $\mathcal{E}$ *e* $\mathcal{D}^*$ *para formar uma única string* $\mathcal{U} = \mathcal{A}^*\mathcal{E}\mathcal{D}^*$ *que representa o cabeçalho de nossa mensagem. Agora, necessitamos construir o código que será utilizado para transmitir a mensagem original. Utilizando as frequências dos primeiros* $o = 2$ *elementos da mensagem* T *temos o código de Huffman*

$$C(a) = 1,\ C(h) = 00\ e\ C(w) = 01.$$

Assim, para codificar os $o = 2$ *primeiros elementos da mensagem* T *(wa) são necessários 3 bits.*

Para cada elemento do espaço de estados $\mathcal{S}$, *temos os códigos de Huffman condicionais*

$$\begin{array}{lll}
C(a|aa) = 00, & C(h|aa) = 01, & C(w|aa) = 1, \\
C(a|ha) = 0, & C(h|ha) = 10, & C(w|ha) = 11, \\
C(a|wa) = 1, & C(h|wa) = 00, & C(w|wa) = 01, \\
C(a|ah) = 01, & C(h|ah) = 1, & C(w|ah) = 00, \\
C(a|hh) = 00, & C(h|hh) = 01, & C(w|hh) = 1, \\
C(a|wh) = 1, & C(h|wh) = 01, & C(w|wh) = 00,
\end{array}$$

$$C(a|aw) = 1, \qquad C(h|aw) = 00, \qquad C(w|aw) = 01,$$
$$C(a|hw) = 0, \qquad C(h|hw) = 11, \qquad C(w|hw) = 10,$$
$$C(a|ww) = 00, \qquad C(h|ww) = 01, \qquad C(w|ww) = 1.$$

Utilizando os conjuntos de frequências condicionais $\mathcal{N}_s$ e os códigos condicionais $C(\cdot\,|s)$, $s \in \mathcal{S}$, temos que os elementos de posição $o+1 = 3$ a 200 são representados por 275 bits. Podemos definir então um código de prefixo ótimo para a mensagem T como $C^* = \{C(\cdot)\} \cup \{C(\cdot\,|s) : s \in \mathcal{S}\}$. Denominemos por $C^*(T)$ o código de prefixo ótimo C^* aplicado à mensagem T. Dessa forma, $C^*(T)$ é composto por 3+275 = 278 bits.

Finalmente, realizamos a concatenação $\mathcal{T} = \mathcal{U}C^*(T)$, que une o cabeçalho e a mensagem codificada. Note que o cabeçalho é representado por 21+16+200+7 = 244 bits. Assim, temos que a transmissão da mensagem T necessitou de 244+278 = 522 bits para ser realizada.

O exemplo 24 ilustra com detalhes o procedimento básico de codificação de *strings* que utilizamos neste texto. Ele também serve para apontar características muito importantes dessa abordagem. Comparando os exemplos 23 e 24, visualizamos que a consideração da estrutura de dependência dos dados reduz o número de *bits* necessários para a codificação da mensagem. No entanto, a transmissão das contagens condicionais utiliza um número grande de *bits*, que se torna infactível quando o tamanho do alfabeto ou a ordem do modelo são grandes. Nos concentramos, então, em estratégias de comunicar ao receptor a codificação utilizada pelo transmissor de maneira econômica. Uma primeira abordagem é encontrar partições no espaço de estados da cadeia de Markov para construir classes de condicionamento com menos elementos, reduzindo, dessa forma, o número de contagens condicionais a serem transmitidas. [30] apresenta a incorporação dos modelos markovianos de partição como possibilidade de construção de novas classes de condicionamento. Para que essa estratégia seja válida, devemos transmitir também a estrutura no espaço de estados adicional proposta pela partição delineada.

Exemplo 25. *Retornemos às condições do exemplo 24. Note que $\mathcal{L} = \{L_1, L_2, L_3\}$ é uma partição do espaço de estados $\mathcal{S}$, conforme a apresentada na definição 8. Suponha que queremos transmitir a organização de $\mathcal{L}$. Uma abordagem ingênua é a enumeração de cada elemento do espaço de estados utilizando $\lceil \log |\mathcal{L}| \rceil$ bits para identificá-los. Considere que,*

s	Parte	Binário
aa	1	00
ah	2	01
aw	3	10
ha	2	01
hh	1	00
hw	2	01
wa	2	01
wh	2	01
ww	1	00

Tabela 3.1: Identificação dos elementos s do espaço de estados $\mathcal{S}$ às suas respectivas partes da partição $\mathcal{L}$ e sua representação binária, para o exemplo 25.

após a definição dos parâmetros da cadeia de Markov terem sido estabelecidos, transmissor e receptor impõem uma ordem sobre o espaço de estados $\mathcal{S}$, por exemplo, ordem lexicográfica. Com isso, cada elemento do espaço de estados pode ser facilmente identificado como apresentado na tabela 3.1.

Se definirmos $\mathcal{L}_B$ como a concatenação da coluna "Binário" da tabela 3.1, temos que, ao receber o bloco $\mathcal{L}_B$, um receptor é capaz de reconstruir, de maneira única, a partição do espaço de estados $\mathcal{S}$ utilizada.

Consideremos os conjuntos de contagens condicionais $\mathcal{N}_L = \{n_j^{(L)} : j \in A_1\}$, em que $n_j^{(L)} = |\{t > o : T_{t-2}T_{t-1} = s, s \in L, T_t = j\}|$ e T_k é o k-ésimo elemento da string T, para $L \in \mathcal{L}$. Dessa forma,

$\mathcal{N}_{L_1} = \{n_a^{(L_1)} = 10,\, n_h^{(L_1)} = 16,\, n_w^{(L_1)} = 50\}$, $\mathcal{N}_{L_2} = \{n_a^{(L_2)} = 43,\, n_h^{(L_2)} = 26,\, n_w^{(L_2)} = 22\}$, $\mathcal{N}_{L_3} = \{n_a^{(L_3)} = 25,\, n_h^{(L_3)} = 1,\, n_w^{(L_3)} = 5\}$.

Como já realizado nos exemplos anteriores, devemos decidir qual o Modo de transmissão dessas frequências. Pelo discutido nos exemplos 23 e 24, o Modo 0 utiliza 30 bits para transmitir cada um dos grupos de frequências condicionais. Para o Modo 1,

$$\begin{aligned}
\mathcal{D}_{1,L_1} &= I(3,4)\, I(4,3) I(10,4)\, I(5,3) I(16,5)\, I(6,3) I(50,6) \\
&= 0011\ 100\, 1010\ 101\, 10000\ 110\, 110010, \\
\mathcal{D}_{1,L_2} &= I(3,4)\, I(6,3) I(43,6)\, I(5,3) I(25,5)\, I(5,3) I(22,5) \\
&= 0011\ 110\, 101011\ 101\, 11001\ 101\, 10110, \\
\mathcal{D}_{1,L_3} &= I(3,4)\, I(5,3) I(25,5)\, I(1,3) I(1,1)\, I(3,3) I(5,3)
\end{aligned}$$

$$= 0011\ 101\ 11001\ 001\ 1\ 011\ 101.$$

Note que, novamente, $\mathcal{D}_{1,L}$ possui menos de 30 bits, para todo $L \in \mathcal{L}$. Dessa forma, utilizamos para cada grupo um bit de identificação do Modo escolhido e obtemos $\mathcal{D}^*_{1,L} = 1\mathcal{D}_{1,L}$. Para encerrarmos a declaração dessas contagens, concatenamos, em uma ordem preestabelecida (lexicográfica, por exemplo), as strings $\mathcal{D}^*_{1,L}$, $\forall L \in \mathcal{L}$, e denominamos o resultado por $\mathcal{D}^*_1$. Perceba que, agora, nosso cabeçalho $\mathcal{U}$ é representado pela concatenação $\mathcal{U} = \mathcal{A}^*\mathcal{E}\mathcal{L}_B\mathcal{D}^*_1\mathcal{D}^*_2$ e possui 21+16+18+82+7 = 144 bits. Note que esse valor é muito inferior ao encontrado para o cabeçalho do exemplo 24, 244 bits.

Resta apenas definirmos as codificações condicionais a cada $L \in \mathcal{L}$. A partir do algoritmo A.2, derivamos os códigos de de Huffman condicionais

$$\begin{aligned} C(a|L_1) &= 00, \quad C(h|L_1) = 01, \quad C(w|L_1) = 1, \\ C(a|L_2) &= 0, \quad C(h|L_2) = 11, \quad C(w|L_2) = 10, \\ C(a|L_3) &= 1, \quad C(h|L_3) = 00, \quad C(w|L_3) = 01. \end{aligned}$$

Utilizando os conjuntos de frequências condicionais $\mathcal{N}_L$ e os códigos condicionais $C(\cdot\,|L)$, $L \in \mathcal{L}$, temos que os elementos de posição $o + 1 = 3$ a 200 são representados por 278 bits. Podemos definir então um código de prefixo ótimo para a mensagem T como $C^* = \{C(\cdot)\} \cup \{C(\cdot\,|L) : L \in \mathcal{L}\}$, em que $C(\cdot)$ é o sistema de codificação marginal já calculado no exemplo 24. Denominemos por $C^*(T)$ o código de prefixo ótimo C^* aplicado à mensagem T. Dessa forma, $C^*(T)$ é composto por 3+278 = 281 bits. Perceba que este valor é ligeiramente maior do que o obtido no exemplo 24. Isso se deve ao fato de não utilizarmos a codificação condicional baseada em cada $s \in \mathcal{S}$ individualmente (código mais detalhado), mas uma codificação condicional baseada em cada $L \in \mathcal{L}$ (código menos detalhado). Esse efeito é menos perceptível em grandes mensagens, pois, conforme o tamanho amostral aumenta, mais próximo $p(x|s)$ é de $p(x|L)$, ocasionando $C(x|s) = C(x|L)$, para todo $x \in A$, $L \in \mathcal{L}$ e $s \in L$.

Por fim, realizamos a concatenação $\mathcal{T} = \mathcal{U}C^*(T)$, que une o cabeçalho e a mensagem codificada. Dessa forma, temos que a transmissão da mensagem T necessitou de 144+281 = 425 bits para ser realizada, quantidade menor que a apresentada no exemplo 24.

Nota 12. *Na prática, dificilmente temos acesso à partição $\mathcal{L}$ do espaço de estados do*

processo trabalhado. No entanto, podemos estimá-la de maneira consistente utilizando, por exemplo, o algoritmo A.1 e transmitir essa estimativa da mesma maneira realizada no exemplo 25.

Note que, mesmo havendo uma redução significativa no tamanho do cabeçalho, o número de *bits* necessários para expressar a partição do espaço de estados utilizado e suas contagens condicionais ainda é consideravelmente grande, e crescerá ainda mais com o aumento do alfabeto. Dessa forma, temos interesse em estabelecer técnicas que consigam transmitir os códigos de Huffman utilizados de maneira mais eficiente e de fácil compreensão para o receptor. Uma estratégia é transmitir tais códigos como árvores binárias. Com essa motivação, consideramos uma terceira alternativa para a comunicação de códigos de Huffman: o Modo Árvore. Essa abordagem permite a conversão de um sistema de codificação de Huffman em uma *string* composta por dois blocos, em que o primeiro delineia o formato da árvore binária correspondente e o segundo identifica suas folhas como membros do alfabeto A. Explicações detalhadas sobre este procedimento encontram-se em D.2. Dessa maneira, junto com os Modos 0 e 1 de codificação de frequências, temos três possibilidades para comunicação dos códigos de Huffman transmitidos. Como o Modo 0 é geralmente menos econômico que o Modo 1, não o carregaremos para nossos exemplos. Assim, iremos decidir entre o Modo 1 (utilizando valor 1 como *bit* de identificação) e o Modo Árvore (valor 0 como *bit* de identificação).

Por fim, encerramos esta seção apresentando, na figura B.1, um fluxograma para nosso método de codificação de *strings*, um fluxograma para a decodificação, na figura B.2 (ambos alocados no apêndice B) e suas aplicações em exercícios de simulação mais extensos. Em seguida, na subseção 3.1.1, comparamos procedimentos de compactação que utilizam como classe de condicionamento modelos markovianos completos e modelos markovianos de partição, um conjunto de exemplos ilustrativos e uma aplicação na codificação do sequenciamento completo do genoma do coronavírus Covid-19 já estudado no capítulo 1.

Nota 13. *Retornemos ao exemplo 25. Nele, foi realizada a etapa de transmissão da partição $\mathcal{L}$ através de uma listagem de inteiros binários de $\lceil \log |\mathcal{L}| \rceil$ bits cada, representada na tabela 3.1. Essa estratégia é equivalente à enumeração realizada na declaração do alfabeto utilizado, em que cada elemento codificado representa um índice dentro do alfabeto preestabelecido. No entanto, perceba que, conforme o aumento do espaço de estados $\mathcal{S}^*$ e*

do número de partes na partição $\mathcal{L}$, esse procedimento de enumeração pode ser exaustivo e pouco eficiente, principalmente se existem partes de tamanhos (números de elementos) muito discrepantes. Uma possível estratégia para economia é: declarar o número de partes observadas; computar as frequências de cada uma dessas partes; construir um código de Huffman referente a essas proporções; e codificar os índices das partes utilizando tal codificação. Para que essa abordagem possa ser utilizada, o transmissor deve informar ao receptor o código de Huffman utilizado, fazendo uso do Modo 0, Modo 1 ou Modo Árvore, já discutidos para a transmissão das codificações da mensagem original. Para o exemplo 26, consideramos apenas o Modo 1 e o Modo Árvore para a declaração da partição.

Exemplo 26. *Suponha que transmissor e receptor já tenham estabelecido previamente o seguinte conjunto de configurações:*

- *Alfabeto prefixado:* $A = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$*;*
- *Bits reservados para a definição do modelo para a classe de condicionamento:* 4 *bits para o valor* g*,* 16 *bits para* $G - M$ *e* 4 *bits para* M*;*
- *Comprimento da maior mensagem a ser transmitida:* $N = 2^{20} - 1 = 1.048.575$.

Consideremos que temos interesse em transmitir uma mensagem T cujos elementos são provenientes do alfabeto $A_1 = \{a, d, g, t, u, z\}$. Suponhamos que as observações da string T se comportem conforme um modelo $G3M(g = 1, G = 300, M = 1)$. Perceba que o espaço de estados estendido desse processo é $\mathcal{S}^* = A_1^3$, e que $|\mathcal{S}^*| = 6^3 = 216$. Consideremos que $\mathcal{S}^*$ está organizado em uma partição com seis partes, $\mathcal{L} = \{L_1, \ldots, L_6\}$, organizadas como na tabela 3.2, e as respectivas probabilidades de transição são apresentadas na tabela 3.3.

Parte	Elementos
L_1	$adt, utu, tgu, aau, ddz, zat, gut, gda, uzz, ugg, zut, uzg$
L_2	$utt, tga, tag, agu, uza, tza, gzz, zzz, azg, dtg$
L_3	$ztt, gua, ggu, ttu, uaz, auu, dza, gzd$
L_4	$aua, ggz, dzg, ztz, ggd, zzt$
L_5	tad, gzu, uuz, aug
L_6	$\mathcal{S}^* \setminus \{L_1, \ldots, L_5\}$

Tabela 3.2: Partição $\mathcal{L}$ do espaço de estados estendido $\mathcal{S}^*$ do exemplo 26.

Simulamos uma mensagem T de comprimento $n = 2 \times 10^5$ seguindo as configurações estabelecidas. Por simplicidade, as $g + G = 301$ primeiras observações foram

Parte (L)	$p(a\|L)$	$p(d\|L)$	$p(g\|L)$	$p(t\|L)$	$p(u\|L)$	$p(z\|L)$
L_1	$0,10$	$0,20$	$0,10$	$0,10$	$0,20$	$0,30$
L_2	$0,30$	$0,20$	$0,10$	$0,10$	$0,20$	$0,10$
L_3	$0,70$	$0,05$	$0,05$	$0,05$	$0,05$	$0,10$
L_4	$0,20$	$0,20$	$0,20$	$0,10$	$0,10$	$0,20$
L_5	$0,10$	$0,60$	$0,10$	$0,10$	$0,05$	$0,05$
L_6	$0,10$	$0,10$	$0,10$	$0,10$	$0,50$	$0,10$

Tabela 3.3: Probabilidades de transição de cada parte da partição $\mathcal{L}$ do exemplo 26.

simuladas de maneira independente a partir de uma distribuição uniforme em A_1*. Diferentemente dos exemplos anteriores, onde mostramos com detalhes cada etapa do processo, aqui nos atemos apenas nos resultados (e número de bits) associados a cada passo do fluxograma apresentado na figura B.1.*

Primeiramente, façamos a etapa de declaração do alfabeto. O método de enumeração utiliza 35 bits, enquanto o de listagem 26. Dessa forma, escolhemos a string proveniente da listagem e a concatenamos com o bit de identificação com valor 0, obtendo $\mathcal{A}^* = 0\mathcal{A}_0$*, em que* $\mathcal{A}_0$ *é a string binária resultante do método de listagem.*

Em seguida, precisamos declarar os valores g*,* G *e* M *utilizados no modelo G3M. Neste caso,*

$$\mathcal{E} = I(1,4)I(299,16)I(1,4) = 0001\,0000000100101011\,0001.$$

Agora, precisamos estimar a partição $\mathcal{L}$*, a partir da string* T*, utilizando o algoritmo A.1. Para esta amostra, o algoritmo estimou a partição* $\widehat{\mathcal{L}} = \{\widehat{L_1}, \ldots, \widehat{L_8}\}$*, cujas partes são detalhadas na tabela 3.4.*

Repare que o algoritmo A.1 estimou uma partição de oito partes. Perceba que as partes L_1 *a* L_5 *do modelo original foram recuperadas com exatidão como as partes* $\widehat{L_1}$ *a* $\widehat{L_5}$*, respectivamente, da partição estimada. No entanto, a parte* L_6 *do modelo original não foi completamente recuperada e apresentou-se dividida em três conjuntos da partição estimada:* $\widehat{L_6}$*,* $\widehat{L_7}$ *e* $\widehat{L_8}$*. Pequenos equívocos de classificação são esperados e tendem a diminuir conforme o comprimento da mensagem* T *aumenta, uma vez que a estimação da partição está associada a resultados assintóticos, de acordo com o estabelecido no corolário 4. Devemos, agora, declarar binariamente a partição estimada* $\widehat{\mathcal{L}}$*. Após relacionar cada elemento do espaço de estados* $\mathcal{S}^*$ *com sua parte correspondente na partição estimada* $\widehat{\mathcal{L}}$

Parte	Elementos
$\widehat{L_1}$	$adt, utu, tgu, aau, ddz, zat, gut, gda, uzz, ugg, zut, uzg$
$\widehat{L_2}$	$utt, tga, tag, agu, uza, tza, gzz, zzz, azg, dtg$
$\widehat{L_3}$	$ztt, gua, ggu, ttu, uaz, auu, dza, gzd$
$\widehat{L_4}$	$aua, ggz, dzg, ztz, ggd, zzt$
$\widehat{L_5}$	tad, gzu, uuz, aug
$\widehat{L_6}$	$ada, zda, aga, dga, uga, gta, zta, dua, uua, aad, dad, tdd, dgd, zgd, dtd, dud, tud,$ $uud, uzd, ddg, tdg, udg, agg, dgg, tgg, gtg, ttg, tug, gzg, aat, uat, gdt, zdt, agt, ggt,$ $dut, azt, dzt, tau, uau, gdu, udu, ugu, daz, taz, zaz, agz, dgz, zgz, ttz, duz, dzz$
$\widehat{L_7}$	$daa, zaa, dda, tda, zga, tta, uta, zza, uad, ddd, agd, tgd, ztd, dzd, uag, zag, utg,$ $dug, gug, zug, dat, ddt, tdt, udt, dgt, tgt, zgt, att, ttt, aut, dau, gau, atu, guu,$ $dzu, tzu, aaz, adz, tdz, udz, ugz, atz, gtz, auz, tzz,$
$\widehat{L_8}$	$\mathcal{S}^* \setminus \{\widehat{L_1}, \ldots, \widehat{L_7}\}$

Tabela 3.4: Partição estimada $\widehat{\mathcal{L}}$ do espaço de estados estendido $\mathcal{S}^*$ do exemplo 26.

(conforme tabulação exemplificada na tabela 3.1), utilizamos um código de Huffman para a codificação da partição, consoante ao argumentado na nota 13, com duas estratégias para transmissão de tal código: Modo 1; e Modo Árvore. Para ambos procedimentos, é necessária a informação de qual o número de partes considerado, e, portanto, existe a necessidade de reservarmos uma determinada quantidade de bits para essa informação. Pensemos que, no pior dos casos, cada elemento do espaço de estados estendido $\mathcal{S}^*$ constitui uma parte da partição $\widehat{\mathcal{L}}$, implicando $|\widehat{\mathcal{L}}| = |\mathcal{S}^*|$. Em nosso caso, como $|\mathcal{S}^*| = 6^3 = 216$, são necessários, no máximo, $\lfloor \log 216 \rfloor + 1 = 8$ bits para representar o número de partes $|\widehat{\mathcal{L}}|$, cuja representação binária denotamos por $|\widehat{\mathcal{L}}|_B$. Utilizando o Modo 1 para a declaração do código de Huffman para as partes obtemos uma string binária com 65 bits, enquanto para o Modo Árvore foram necessários 47 bits. Como o Modo Árvore foi o mais econômico, o escolhemos e adicionamos um bit de identificação inicial com valor 0 e definimos a string binária $\widehat{\mathcal{L}}_{modelo}$. Em seguida, codificamos os elementos do espaço de estados estendido segundo o código de Huffman considerado, confeccionando a string binária $\widehat{\mathcal{L}}_B$, que, em nosso caso, possui 522 bits. Finalizando o processo de declaração da partição, concatenamos todas essas informações em apenas uma única string: $\widehat{\mathcal{L}}_B^* = |\widehat{\mathcal{L}}|_B \, \widehat{\mathcal{L}}_{modelo} \, \widehat{\mathcal{L}}_B$, que é constituída por 8+48+522 = 578 bits.

O próximo passo é a execução sequencial do algoritmo A.2 para obtenção das codificações condicionadas a cada elemento da partição $\widehat{\mathcal{L}}$ que serão utilizadas para codificar a mensagem original. A tabela 3.5 apresenta os códigos de Huffman condicionais para cada elemento da partição $\widehat{\mathcal{L}}$.

Parte (L)	$C(a\|L)$	$C(d\|L)$	$C(g\|L)$	$C(t\|L)$	$C(u\|L)$	$C(z\|L)$
$\widehat{L}_1$	100	101	011	010	00	11
$\widehat{L}_2$	11	01	000	001	101	100
$\widehat{L}_3$	1	0101	000	0100	001	011
$\widehat{L}_4$	01	110	00	101	100	111
$\widehat{L}_5$	011	1	001	010	0001	0000
$\widehat{L}_6$	101	100	1110	110	0	1111
$\widehat{L}_7$	000	0110	010	001	1	0111
$\widehat{L}_8$	1111	110	100	1110	0	101

Tabela 3.5: Códigos de Huffman condicionados a cada parte da partição $\widehat{\mathcal{L}}$ do exemplo 26.

Cada um dos códigos condicionais deve ser transmitido utilizando o Modo 1 ou Modo Árvore. Todas as codificações apresentaram o Modo Árvore mais econômico que o Modo 1. Concatenando um bit de identificação de valor 0 em cada uma dessas strings binárias, obtemos as strings $\mathcal{D}^*_{1,L}$, $L \in \widehat{\mathcal{L}}$. *Por fim, definimos como* $\mathcal{D}^*_1$ *a concatenação das strings* $\mathcal{D}^*_{1,L}$ *em uma única string binária. Em nosso caso,* $\mathcal{D}^*_1$ *é representada por 240 bits.*

Em seguida, devemos construir um código de Huffman associado às frequências das $G + g = 301$ *primeiras observações da string* T. *A codificação utilizada é*

$$C(a) = 111, \quad C(d) = 110, \quad C(g) = 100,$$
$$C(t) = 01, \quad C(u) = 00, \quad C(z) = 101.$$

Para a transmissão do código marginal, consideramos novamente o Modo 1 (que utiliza 42 bits) e o Modo Árvore (que utiliza 29 bits). Como o modo árvore é o mais econômico, o escolhemos e comunicamos nossa decisão incluindo um bit de identificação com valor 0. A identificação do código marginal é denotada por $\mathcal{D}^*_2$ *e possui, portanto, 30 bits. Com isso, podemos definir o cabeçalho de nossa mensagem codificada como a concatenação* $\mathcal{U} = \mathcal{A}^*\mathcal{E}\widehat{\mathcal{L}}^*_B\mathcal{D}^*_1\mathcal{D}^*_2$, *que possui 27+24+578+240+30 = 899 bits.*

Definimos um código de prefixo ótimo para a mensagem T *como a união dos códigos marginal e condicionais, ou seja,* $C^* = \{C(\cdot)\} \cup \{C(\cdot\,|L) : L \in \widehat{\mathcal{L}}\}$. *Denominemos por* $C^*(T)$ *o código de prefixo ótimo* C^* *aplicado à mensagem* T. *Em nosso caso,* $C^*(T)$ *é uma string de 439.063 bits. Por fim, realizamos a concatenação* $\mathcal{T} = \mathcal{U}C^*(T)$, *que une o cabeçalho e a mensagem codificada. Dessa forma, temos que a transmissão da mensagem* T *necessitou de 899+439.063 = 439.962 bits para ser realizada.*

A decodificação da mensagem codificada $\mathcal{T}$ *segue estritamente o procedimento*

ilustrado na figura B.2. Como nele não há paradigmas de escolhas mas apenas leituras apropriadas da mensagem, não iremos desenvolvê-lo aqui para não tornar o exemplo ainda mais exaustivo.

Exemplo 27. *Considere novamente as condições estabelecidas no exemplo 26. Perceba que em nenhum momento foi questionada a escolha dos parâmetros associados à cadeia de Markov com interstício (M, G e g) para a modelagem da mensagem T. Como tal exemplo foi construído apenas como um exercício de aplicação das etapas associadas à compactação de strings, esses três parâmetros foram tratados como conhecidos e o modelo G3M estimado. Na prática, geralmente, não temos acesso aos valores de M, G e g, sendo necessária, então, uma etapa intermediária para a seleção do modelo G3M que melhor descreve os dados observados. Dessa forma, é essencial a escolha de um critério de comparação de modelos que nos permita a seleção daquele que for mais compatível com a amostra considerada. Conforme já explanado no capítulo 1, é interessante que tenhamos um critério de bondade de ajuste que leve em consideração tanto a informação contida nos dados quanto o número de parâmetros utilizados no modelo para descrevê-los. Para manter a consistência deste trabalho, escolhemos o BIC como ferramenta de comparação entre modelos. Assim, dada uma coleção de triplas (g, G, M), podemos realizar os ajustes dos modelos G3M correspondentes e comparar seus BIC's observados. Prosseguindo dessa maneira, podemos selecionar o modelo com o maior BIC e seguir com o processo de compactação, conforme a figura B.1. A tabela C.1 (apêndice C) apresenta os BIC's associados a 40 modelos G3M ajustados para a mensagem T referente ao exemplo 26, assim como os números de bits necessários para descrever binariamente a mensagem utilizando a respectiva modelagem.*

Perceba que o modelo G3M($g = 1$, $G = 300$, $M = 1$) foi o que apresentou maior BIC e uma codificação mais compacta da mensagem T, ou seja, o procedimento aqui discutido foi capaz de não só resgatar a estrutura do modelo original mas também de produzir uma codificação mais econômica. As tabelas C.2, C.3 e C.4 apresentam os resultados para simulações da mensagem T (sob as condições do exemplo 26), mas de comprimentos $n = 10^5$, $n = 10^4$ e $n = 10^3$, respectivamente. Repare que, para $n = 10^3$, o modelo que possuiu maior BIC foi o com a tripla de parâmetros ($g = 2$, $G = 400$, $M = 2$) e, além disso, não foi o que promoveu a representação binária mais econômica da mensagem T. Isso se deve ao fato da amostra ser muito pequena para a complexidade da maioria dos modelos, tornando pobres as estimativas obtidas. Observe também que a inclusão de altos valores de

interstício (parâmetro G) na modelagem diminui a quantidade de $(g+1+M)$-uplas aptas a serem utilizadas durante a estimação (não há probabilidades de transição associadas às primeiras $G+g$ observações). Em particular, para o modelo G3M($g=2$, $G=400$, $M=2$), temos associado o espaço de estados $\mathcal{S}^* = \{a,d,g,t,u,z\}^5$, com $|\mathcal{S}^*| = 6^5 = 7.776$, e temos $\sum_{s\in\mathcal{S}^*} N_n(s) = 598$, ou seja, dispomos apenas de $n-G-g = 598$ $(g+1+M)$-uplas para estimar a partição $\mathcal{L}$ associada ao processo e suas probabilidades de transição. Assim, não conseguimos boas aproximações e nos distanciamos de resultados ótimos. No entanto, conforme o tamanho da mensagem T aumenta, a escolha do modelo via BIC passa a selecionar o modelo original e este se torna o que gera a representação binária mais econômica, conforme observados nas tabelas C.1 a C.3. Dessa forma, evidenciamos a consistência do BIC como critério de seleção do modelo que gerará a representação binária mais compacta da mensagem T, seguindo o procedimento de codificação apresentado na figura B.1.

3.1.1 Compressão via Modelo Completo vs Modelo de Partição

A seção 2.3 apresentou uma sequência de resultados que associam a entropia de um processo estocástico $\{X_t\}_{t\in\mathbb{N}}$ com o número de *bits* necessários para representar binariamente uma amostra $X_1,\ldots,X_n$ desse processo. Na seção 2.1, foi discutido que o condicionamento de uma variável aleatória Y em outra variável aleatória X não aumenta, em média, a incerteza sobre Y, ou seja, $H(Y|X) \leq H(Y)$ (teorema 7). Assim, sejam Y variável aleatória discreta definida sobre o alfabeto A_Y e C_Y^* um código de prefixo ótimo para Y. Pelo teorema 15, temos

$$H(Y) \leq L_Y^* < H(Y) + 1, \tag{3.3}$$

em que L_Y^* é o comprimento esperado do código ótimo, C_Y^*, para Y. Sejam X um vetor aleatório sobre o alfabeto A_X e $C_{Y|X}^*$ um código de prefixo ótimo para a variável Y condicionada ao vetor aleatório X. Novamente pelo teorema 15, temos

$$H(Y|X) \leq L_{Y|X}^* < H(Y|X) + 1, \tag{3.4}$$

em que $L_{Y|X}^*$ é o comprimento esperado do código ótimo, $C_{Y|X}^*$, para Y condicionada em X. Pelo teorema 7, temos $H(Y|X) \leq H(Y)$. Assim, o limite inferior da equação (3.3)

é maior ou igual ao da equação (3.4) e o limite superior da equação (3.4) é menor ou igual ao da equação (3.3). Ou seja, o código $C^*_{Y|X}$ entrega uma codificação para Y cujo comprimento médio é não superior ao obtido por C^*_Y. Dessa forma, temos justificada a utilização da codificação condicional para obtenção de um código com menor comprimento esperado. Por esse motivo, classes de condicionamento capazes de otimizar a taxa de compressão de códigos são almejadas.

Neste texto, como trabalhamos com códigos instantâneos, para codificar um elemento da amostra, podemos utilizar o passado observado como classe de condicionamento. Ou seja, para uma amostra $X_1, \ldots, X_n$ de um processo estocástico $\{X_t\}_{t\in\mathbb{N}}$ assumindo valores em um alfabeto A_X, podemos estabelecer, para cada $i \in \{2, \ldots, n\}$, códigos condicionais para X_i da forma $C(\cdot|X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$. Repare que classes de condicionamento dessa natureza possuem um número muito grande de elementos e não são úteis para a compressão tratada neste trabalho, uma vez que a estrutura condicionante também deve ser informada pela *string* binária a ser transmitida. Uma estratégia para a simplificação dessa estrutura provém do pressuposto que observações de um passado distante revelam pouca informação sobre observações recentes. Quando essa condição é razoável para o processo estudado, modelos markovianos podem ser empregados a fim de obter um conjunto de classes de condicionamento de tamanho reduzido. Assim, para uma amostra $X_1, \ldots, X_n$ de uma cadeia de Markov a tempo discreto de ordem o sobre um alfabeto finito A_X e com espaço de estados $\mathcal{S} = A_X^o$, podemos estabelecer, para cada $i \in \{o+1, \ldots, n\}$, códigos condicionais para X_i da forma $C(\cdot|X_{i-o} = x_{i-o}, \ldots, X_{i-1} = x_{i-1})$. Repare que, pela hipótese de estacionariedade, o código condicional não depende da posição $i \in \{o+1, \ldots, n\}$, mas apenas de $(x_{i-o}, \ldots, x_{i-1}) \in \mathcal{S}$. Dessa forma, o conjunto das classes de condicionamento passa a ter $|\mathcal{S}| = |A_X|^o$ elementos. Assim, para o problema de codificação, se for desejado utilizar a abordagem de cadeias de Markov de ordem o como estratégia de simplificação, todos os códigos condicionais $C(\cdot|s)$, $s \in \mathcal{S}$, devem ser informados, ou seja, além da amostra (mensagem) $X_1, \ldots, X_n$ codificada, $|A_X|^o$ sistemas de codificação condicionais devem ser transmitidos.

Apesar de estabelecer um conjunto de classes condicionantes que possui potencial de reduzir o tamanho da mensagem codificada, a abordagem via cadeias de Markov pode ainda não ser eficiente para a compactação como um todo, pois necessita da transmissão de $|A_X|^o$ codificações condicionais. Por esse motivo, é interessante procurarmos

conjuntos de classes de condicionamento ainda menores, que não prejudiquem muito na redução do comprimento da mensagem codificada. Adaptando para o vocábulo de nosso trabalho, estamos buscando o equilíbrio entre o número de *bits* necessários para estabelecer um modelo (cabeçalho) e o número de *bits* que representam a mensagem codificada por ele. Essa mesma procura incentivou [26] a propor as cadeias de Markov de alcance variável (VLMC). Nele, como visto na seção 1.1, uma função contexto, $c(\cdot)$, é responsável por agrupar estados baseando-se na porção do passado que influencia a próxima observação. A imagem da função contexto é chamada de árvore de contexto e denotada por τ. Pelo fato do modelo VLMC agrupar estados, temos que $|\tau| \leq |\mathcal{S}|$. Assim, para uma amostra $X_1, \ldots, X_n$ de uma cadeia de Markov de alcance variável a tempo discreto de ordem o sobre um alfabeto finito A_X, com espaço de estados $\mathcal{S} = A_X^o$, função contexto $c(\cdot)$ e árvore de contexto τ, podemos estabelecer, para cada $i \in \{o+1, \ldots, n\}$, códigos condicionais para X_i da forma $C(\cdot|c(x_{i-o}^{i-1}))$, em que $c(x_{i-o}^{i-1}) \in \tau$. Repare que o modelo VLMC possibilita que um número menor ($|\tau|$) de códigos condicionais seja necessário para a transmissão da mensagem codificada.

Uma outra estratégia de agrupamento de estados de uma cadeia de Markov é delineada pelos modelos markovianos de partição, introduzidos na seção 1.2. Esses modelos criam uma partição no espaço de estados baseada nas probabilidades de transição, conforme estabelecido na definição 8. No final da seção 1.1, discutimos que os modelos VLMC podem ser vistos como um caso particular dos modelos de partição. Assim, para uma amostra $X_1, \ldots, X_n$ de uma cadeia de Markov a tempo discreto de ordem o sobre um alfabeto finito A_X, com espaço de estados $\mathcal{S} = A_X^o$ e partição $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$, podemos estabelecer, para cada $i \in \{o+1, \ldots, n\}$, códigos condicionais para X_i da forma $C(\cdot|L)$, em que $L \in \mathcal{L}$. Conforme discutido na seção 1.2, o modelo markoviano de partição possui uma complexidade menor que uma cadeia de Markov completa, uma vez que $|\mathcal{L}| \leq |\mathcal{S}|$. Dessa forma, sob o escopo de codificação, o modelo markoviano requer a incorporação de menos códigos condicionais à mensagem binária a ser transmitida.

Nesta subseção, temos interesse em comparar as características associadas à codificação de *strings* quando utilizamos um modelo markoviano completo e um modelo markoviano de partição como classes de condicionamento. Primeiramente, repare que, fixada a memória do processo markoviano (ordem o para cadeia de Markov ou parâmetros g, G e M para cadeia de Markov com interstício), a consideração da partição implica um

encargo extra na mensagem binária a ser transmitida, pois é necessário o detalhamento das partes associadas à partição. Esse encargo não ocorre, por exemplo, quando utilizamos como classe de condicionamento o próprio espaço de estados. No entanto, como observado anteriormente, o modelo de partição necessita informar $|\mathcal{L}|$ códigos condicionais, enquanto o modelo completo necessita transmitir $|\mathcal{S}|$, e, como $|\mathcal{L}| \leq |\mathcal{S}|$, há uma potencial redução na quantidade de sistemas de codificação a serem repassados. Observe que esses dois grupos de informação (detalhamento das classes de condicionamento e códigos condicionais) pertencem ao que chamamos, neste texto, de "cabeçalho". Além disso, os sistemas de codificação baseados em $\mathcal{S}$ geram códigos de menor comprimento médio quando comparados aos baseados em $\mathcal{L}$, por serem mais detalhados e utilizarem o código de Huffman condicional ótimo para cada estado $s \in \mathcal{S}$. Assim, temos uma pergunta crucial a responder: a potencial redução da quantidade de codificações condicionais a serem transmitidas (proposta pelo modelo markoviano de partição) reduz o comprimento da *string* binária final, quando comparada à obtida pelos códigos condicionais baseados em todo o espaço de estados $\mathcal{S}$? Para ilustrar o paradigma proposto, apresentamos uma sequência de exemplos que evidenciam a superioridade de cada abordagem em diferentes configurações. Diferentemente do realizado no exemplo 27, não nos preocupamos com a metodologia de seleção de modelos, ou seja, a memória do processo markoviano será considerada conhecida, uma vez que nosso interesse está apenas nas diferentes características da transmissão de informações dos procedimentos concorrentes a serem apresentados.

Exemplo 28. *Neste exemplo, queremos investigar o comportamento dos procedimentos de compressão baseados em modelos markovianos completos e modelos markovianos de partição quando ambos estão associados a um espaço de estados de tamanho moderado, mas que pode ser representado por uma partição pequena cujas partes possuem probabilidades de transição razoavelmente distintas.*

Consideremos, mais uma vez, as condições estabelecidas no exemplo 26. Então, temos interesse em transmitir uma mensagem T cujos elementos são provenientes do alfabeto $A_1 = \{a, d, g, t, u, z\}$. Além disso, suponhamos que as observações da string T se comportem conforme um modelo $G3M(g = 1, G = 300, M = 1)$, que possui espaço de estados $\mathcal{S}^* = A_1^3$, com $|\mathcal{S}^*| = 6^3 = 216$. Também, consideramos que $\mathcal{S}^*$ está organizado em uma partição com seis partes, $\mathcal{L} = \{L_1, \ldots, L_6\}$, delineadas como na tabela 3.2, e as respectivas probabilidades de transição são apresentadas na tabela 3.3. Agora, geramos três

amostras de tamanhos distintos desse processo ($n = 2 \times 10^3$, $n = 2 \times 10^4$ e $n = 2 \times 10^5$) e realizamos o procedimento de compactação de strings, ilustrado na figura B.1, para cada uma delas em dois cenários para as classes de condicionamento: modelo de partição; e modelo completo. Para ambos, a estrutura de dependência utilizada foi a de uma cadeia de Markov com interstício com a tripla de parâmetros ($g = 1$, $G = 300$, $M = 1$) e reutilizadas as configuraçõs iniciais do exemplo 26 para o procedimento de compressão. As tabelas C.5, C.6 e C.7 apresentam os resultados das codificações para as abordagens de modelo de partição e de modelo completo para amostras de tamanhos $n = 2 \times 10^3$, $n = 2 \times 10^4$ e $n = 2 \times 10^5$, respectivamente. Vale a pena ressaltarmos quais os significados das linhas das tabelas que se seguem:

- *Alfabeto utilizado: string binária que representa quais elementos do alfabeto prefixado estão presentes na mensagem a ser transmitida;*
- *Definição do modelo: string binária que representa os valores da tripla de parâmetros (g, G, M) para os modelos markovianos;*
- *Partição estimada: string binária constituída pela partição codificada e os códigos utilizados para determiná-la (presente apenas nos modelos de partição);*
- *Codificações condicionais: string binária composta por todos os códigos condicionais necessários para codificar a mensagem a ser transmitida;*
- *Codificações marginais: string binária que representa o código que será utilizado para codificar os $g + G$ primeiros elementos da mensagem a ser transmitida;*
- *Mensagem codificada: string binária resultante da codificação da mensagem original pelos códigos marginais e condicionais;*
- *Total: string binária resultante da concatenação do cabeçalho e da mensagem codificada.*

Perceba que, para os três tamanhos amostrais considerados, as tabelas C.5 a C.7 apontam que o modelo markoviano completo codifica a mensagem original de maneira mais econômica que o modelo de partição. No entanto, para que a codificação possa ser realizada, o modelo completo necessita operacionar sobre uma grande quantidade de códigos condicionais, que aumenta em demasia o tamanho da string binária final a ser transmitida. É possível notar que o modelo de partição sacrifica um pouco da taxa de compressão sobre

a codificação da mensagem, mas procura economizar na quantidade de informações a serem transmitidas pelo cabeçalho, gerando uma potencial redução no comprimento da string binária final. Em particular, observamos uma redução de 41%, 10% e 1% no comprimento da string binária final do modelo de partição sobre o modelo completo, para as amostras de tamanhos $n = 2 \times 10^3$, $n = 2 \times 10^4$ *e* $n = 2 \times 10^5$, *respectivamente. Ou seja, para amostras suficientemente grandes, fixado um modelo, as duas abordagens tendem a se equivaler, pois o número de bits gastos em discrepâncias do cabeçalho torna-se muito pequeno quando comparado ao número de bits da mensagem codificada.*

Exemplo 29. *Neste exemplo, queremos investigar o comportamento dos procedimentos de compressão baseados em modelos markovianos completos e modelos markovianos de partição quando o modelo original possui uma partição de tamanho próximo ao do espaço de estados do processo estudado.*

Suponha que transmissor e receptor já tenham estabelecido previamente o seguinte conjunto de configurações:

- *Alfabeto prefixado:* $A = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$;
- *Bits reservados para a definição do modelo para a classe de condicionamento:* 4 *bits para o valor* g, 8 *bits para* $G - M$ *e* 4 *bits para* M;
- *Comprimento da maior mensagem a ser transmitida:* $N = 2^{20} - 1 = 1.048.575$.

Consideremos que temos interesse em transmitir uma mensagem cujos elementos são provenientes do alfabeto $A_1 = \{a, b, c\}$. *Suponhamos que as observações da string se comportem conforme um modelo* $G3M(g = 0,\ G = 2,\ M = 1)$. *Perceba que o espaço de estados estendido desse processo é* $\mathcal{S}^* = A_1^2$, *e que* $|\mathcal{S}^*| = 3^2 = 9$. *Consideremos que* $\mathcal{S}^*$ *está organizado em uma partição com oito partes,* $\mathcal{L} = \{L_1, \ldots, L_8\}$, *delineadas como*

$$\begin{aligned} L_1 &= \{aa,\ ba\}, \quad L_3 = \{ac\}, \quad L_5 = \{bc\}, \quad L_7 = \{cb\}, \\ L_2 &= \{ab\}, \qquad L_4 = \{bb\}, \quad L_6 = \{ca\}, \quad L_8 = \{cc\}, \end{aligned}$$

cujas probabilidades de transição são dadas por

$$\begin{aligned} &p(a|L_1) = 0,1, \quad p(b|L_1) = 0,2, \qquad p(a|L_5) = 0,5, \quad p(b|L_5) = 0,4, \\ &p(a|L_2) = 0,7, \quad p(b|L_2) = 0,2, \qquad p(a|L_6) = 0,4, \quad p(b|L_6) = 0,4, \\ &p(a|L_3) = 0,6, \quad p(b|L_3) = 0,2, \qquad p(a|L_7) = 0,2, \quad p(b|L_7) = 0,4, \\ &p(a|L_4) = 0,1, \quad p(b|L_4) = 0,5, \qquad p(a|L_8) = 0,1, \quad p(b|L_8) = 0,3. \end{aligned}$$

Simulamos mensagens de comprimento $n = 10^3$, $n = 5 \times 10^3$ e $n = 10^4$ seguindo as configurações estabelecidas. Por simplicidade, as $g + G = 2$ primeiras observações foram simuladas de maneira independente a partir de uma distribuição uniforme em A_1. As tabelas C.8, C.9 e C.10 apresentam os resultados das codificações para as abordagens de modelo de partição e de modelo completo para amostras de tamanhos $n = 10^3$, $n = 5 \times 10^3$ e $n = 10^4$, respectivamente.

É interessante notar que, pelas tabelas C.9 e C.10, os modelos completos entregaram strings binárias finais de menores comprimentos que as providas pelos modelos de partição. Note que, para esses dois casos, o detalhamento da partição não compensou a redução da quantidade de codificações condicionais a serem transmitidas, provocando um aumento do cabeçalho. Em particular, para os modelos de partição, foram estimadas partições de tamanhos 5, 7 e 8 para as amostras de tamanho $n = 10^3$, $n = 5 \times 10^3$ e $n = 10^4$, respectivamente. Repare que o aumento do número de partes é diretamente associado ao aumento do número de bits para detalhar a partição, e, para este exemplo, tal detalhamento é vantajoso para o caso da amostra com $n = 10^3$, mas prejudicial para os outros dois casos. No entanto, o algoritmo A.1 (utilizado para estimar a partição) é consistente e, portanto, para amostras suficientemente grandes, deve recuperar a partição original do processo, tornando estável o número de bits necessários para detalhar a partição estimada. Além disso, temos que as estimativas das probabilidades condicionais também devem convergir no caso de grandes amostras, estabilizando o número de bits necessários para transmitir as codificações condicionais. Dessa forma, novamente, para amostras suficientemente grandes, fixado um modelo, as duas abordagens tendem a se equivaler, pois o número de bits gastos em discrepâncias do cabeçalho é estável e torna-se muito pequeno quando comparado ao número de bits da mensagem codificada.

Exemplo 30. *Neste exemplo, queremos investigar o comportamento dos procedimentos de compressão baseados em modelos markovianos completos e modelos markovianos de partição quando o modelo original possui uma partição de cardinalidade consideravelmente menor que a do espaço de estados do processo estudado, mas as probabilidades de transição das partes são parecidas entre si. Ou seja, este é um cenário no qual o algoritmo A.1 deve apresentar dificuldades para recuperar a partição original.*

Suponha que transmissor e receptor já tenham estabelecido previamente o

seguinte conjunto de configurações:

- *Alfabeto prefixado:* $A = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$;
- *Bits reservados para a definição do modelo para a classe de condicionamento:* 4 *bits para o valor* g, 8 *bits para* $G - M$ *e* 4 *bits para* M;
- *Comprimento da maior mensagem a ser transmitida:* $N = 2^{20} - 1 = 1.048.575$.

Consideremos que temos interesse em transmitir uma mensagem cujos elementos são provenientes do alfabeto $A_1 = \{a, b, c\}$. *Suponhamos que as observações da string se comportem conforme um modelo* $G3M(g = 0, G = 2, M = 1)$. *Perceba que o espaço de estados estendido desse processo é* $\mathcal{S}^* = A_1^2$, *e que* $|\mathcal{S}^*| = 3^2 = 9$. *Consideremos que* $\mathcal{S}^*$ *está organizado em uma partição com quatro partes,* $\mathcal{L} = \{L_1, \ldots, L_4\}$, *delineadas como*

$$L_1 = \{ba,\ cb\}, \quad L_2 = \{aa, bb\}, \quad L_3 = \{bc, ab\}, \quad L_4 = \{ca, ac, cc\},$$

cujas probabilidades de transição são dadas por

$$\begin{aligned} &p(a|L_1) = 0,70, \quad p(b|L_1) = 0,20, && p(a|L_3) = 0,68, \quad p(b|L_3) = 0,22, \\ &p(a|L_2) = 0,72, \quad p(b|L_2) = 0,18, && p(a|L_4) = 0,70, \quad p(b|L_4) = 0,10. \end{aligned}$$

Simulamos mensagens de comprimento $n = 10^3$, $n = 5 \times 10^3$ *e* $n = 10^4$ *seguindo as configurações estabelecidas. Por simplicidade, as* $g + G = 2$ *primeiras observações foram simuladas de maneira independente a partir de uma distribuição uniforme em* A_1. *As tabelas C.11, C.12 e C.13 apresentam os resultados das codificações para as abordagens de modelo de partição e de modelo completo para amostras de tamanhos* $n = 10^3$, $n = 5 \times 10^3$ *e* $n = 10^4$, *respectivamente.*

Para os três casos considerados neste exemplo, o algoritmo A.1 estimou a mesma partição $\widehat{\mathcal{L}} = \{\widehat{L_1}, \widehat{L_2}\}$, *em que* $\widehat{L_2} = \{ca, ac, cc\}$ *e* $\widehat{L_1} = \mathcal{S}^* \setminus \widehat{L_2}$. *Repare que* $\widehat{L_2} = L_4$ *e* $\widehat{L_1} = L_1 \cup L_2 \cup L_3$, *em que* L_1, L_2, L_3 *e* L_4 *são as partes que compõem a partição original* $\mathcal{L}$. *Perceba que, apesar do modelo de partição não ter conseguido recuperar com sucesso a partição original do processo, foi capaz de criar uma estimativa da partição que une todos os estados cujas probabilidades de transição foram suficientemente próximas. E essa estimativa foi apropriada para criar um conjunto de classes de condicionamento bom o bastante para codificar os dados sem aumentar o tamanho do cabeçalho. Por fim, vale notar que ambas as abordagens (modelo completo e modelo de partição) são equivalentes para amostras suficientemente grandes, pelos mesmos motivos já explanados nos exemplos*

28 e 29.

Exemplo 31. *Neste exemplo, queremos investigar o comportamento dos procedimentos de compressão baseados em modelos markovianos completos e modelos markovianos de partição quando o modelo original possui uma partição de tamanho próximo ao do espaço de estados do processo estudado e as probabilidades de transição das partes são parecidas entre si.*

Suponha que transmissor e receptor já tenham estabelecido previamente o seguinte conjunto de configurações:

- *Alfabeto prefixado:* $A = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$*;*
- *Bits reservados para a definição do modelo para a classe de condicionamento:* 4 *bits para o valor* g*,* 8 *bits para* $G - M$ *e* 4 *bits para* M*;*
- *Comprimento da maior mensagem a ser transmitida:* $N = 2^{20} - 1 = 1.048.575$.

Consideremos que temos interesse em transmitir uma mensagem cujos elementos são provenientes do alfabeto $A_1 = \{a, b, c\}$*. Suponhamos que as observações da string se comportem conforme um modelo* $G3M(g = 0,\, G = 2,\, M = 1)$*. Perceba que o espaço de estados estendido desse processo é* $\mathcal{S}^* = A_1^2$*, e que* $|\mathcal{S}^*| = 3^2 = 9$*. Consideremos que* $\mathcal{S}^*$ *está organizado em uma partição com oito partes,* $\mathcal{L} = \{L_1, \ldots, L_8\}$*, delineadas como*

$$\begin{array}{llll} L_1 = \{aa,\, ba\}, & L_3 = \{ac\}, & L_5 = \{bc\}, & L_7 = \{cb\}, \\ L_2 = \{ab\}, & L_4 = \{bb\}, & L_6 = \{ca\}, & L_8 = \{cc\}, \end{array}$$

cujas probabilidades de transição são dadas por

$$\begin{array}{llll} p(a|L_1) = 0,10, & p(b|L_1) = 0,20, & p(a|L_5) = 0,08, & p(b|L_5) = 0,22, \\ p(a|L_2) = 0,12, & p(b|L_2) = 0,18, & p(a|L_6) = 0,10, & p(b|L_6) = 0,18, \\ p(a|L_3) = 0,12, & p(b|L_3) = 0,22, & p(a|L_7) = 0,10, & p(b|L_7) = 0,22, \\ p(a|L_4) = 0,08, & p(b|L_4) = 0,18, & p(a|L_8) = 0,12, & p(b|L_8) = 0,20. \end{array}$$

Simulamos mensagens de comprimento $n = 10^3$*,* $n = 5 \times 10^3$ *e* $n = 10^4$ *seguindo as configurações estabelecidas. Por simplicidade, as* $g + G = 2$ *primeiras observações foram simuladas de maneira independente a partir de uma distribuição uniforme em* A_1*. As tabelas C.14, C.15 e C.16 apresentam os resultados das codificações para as abordagens de modelo de partição e de modelo completo para amostras de tamanhos* $n = 10^3$*,* $n = 5 \times 10^3$ *e* $n = 10^4$*, respectivamente.*

Para os três casos considerados neste exemplo, o algoritmo A.1 estimou partições de cardinalidades 1, 1 e 2 para as amostras de tamanhos $n = 10^3$, $n = 5 \times 10^3$ *e* $n = 10^4$, *respectivamente. Perceba que, apesar do modelo de partição não ter conseguido recuperar com sucesso a partição original do processo, foi capaz de criar uma estimativa da partição que une todos os estados cujas probabilidades de transição foram suficientemente próximas, como o ocorrido no exemplo 30. E, novamente, essa estimativa foi apropriada para criar um conjunto de classes de condicionamento bom o bastante para codificar os dados sem aumentar o tamanho do cabeçalho. Além disso, vale notar que ambas as abordagens (modelo completo e modelo de partição) são equivalentes para amostras suficientemente grandes, pelos mesmos motivos já explanados anteriormente.*

Os exemplos 28, 29, 30 e 31 evidenciam que, para amostras suficientemente grandes, o método de codificação de *strings* apresentado na figura B.1 resulta em *strings* binárias de tamanhos muito próximos quando consideramos modelos markovianos completos ou de partição para a construção das classes de condicionamento que serão utilizadas nas codificações condicionais. No entanto, para amostras menores, dependendo da configuração da partição do espaço de estados associado ao processo estudado e de suas probabilidades de transição, uma ou outra abordagem pode ser mais conveniente. Em particular, vimos que os modelos de partição promovem uma melhoria na taxa de compressão dos dados quando a partição possui cardinalidade consideravelmente menor que a do espaço de estados ao qual está associada (exemplo 28). No entanto, quando as cardinalidades são próximas e as respectivas probabilidades de transição são razoavelmente distintas (exemplo 29), o modelo completo apresenta vantagens. Contudo, quando as probabilidades de transição são próximas (exemplos 30 e 31), mesmo quando a partição original do processo não é completamente recuperada, a partição estimada pelo algoritmo A.1 proporciona classes de condicionamento apropriadas para a codificação da mensagem de interesse. Encerramos esta seção comparando as duas abordagens concorrentes aqui apresentadas quando aplicadas sobre o conjunto de bases que compõem o genoma do coronavírus já estudado no capítulo 1.

3.1.2 Compressão de Genoma da Covid-19

Retomemos o estudo de caso do sequenciamento do genoma da Covid-19 explorado no capítulo 1. Nele, o conjunto de dados foi interpretado como uma sequência

de 29.903 observações assumindo valores no conjunto de bases $A = \{a, c, g, t\}$. Também, na tabela 1.8, vimos que o modelo G3M que apresentou maior BIC, dentre os propostos, foi o com a tripla de parâmetros ($g = 0$, $G = 8$, $M = 4$), e, por isso, foi considerado o mais adequado para explicar o processo analisado. A fim de explorarmos o discutido nesta subseção, realizamos o processo de compactação de *strings*, ilustrado na figura B.1, para a amostra do genoma da Covid-19 em dois cenários para as classes de condicionamento: modelo de partição; e modelo completo. Em [11] foi realizado um estudo detalhado deste mesmo genoma via abordagem G3M e, devido às bases nitrogenadas do DNA serem organizadas em triplas, foi estabelecida uma ordem imediata $M = 3$ para o modelo e o valor do interstício G foi selecionado via maximização do critério BIC. Isso resultou na escolha do modelo G3M com a tripla de parâmetros ($g = 0$, $G = 9$, $M = 3$). Na aplicação apresentada nesta subseção, adotaremos tal modelo. Reutilizando as configuraçõs iniciais do exemplo 26 para o processo de compressão, as tabelas 3.6 e 3.7 apresentam os resultados da codificação para a abordagem de modelo de partição e para modelo completo, respectivamente.

Observe que o modelo de partição, apesar de possuir uma codificação da mensagem de comprimento superior, possuiu uma economia no tamanho do cabeçalho. Além disso, a *string* binária resultante dessa abordagem apresentou comprimento total inferior ao de sua concorrente. Ou seja, a simplificação da estrutura de dependência proposta pelo modelo de partição promoveu uma redução no tamanho do conjunto de classes de condicionamento necessário para um sistema de codificação apropriado. Assim sendo, a utilização da abordagem de modelos de partição é capaz de trazer avanços no processo de compactação de *strings*, que resultam em melhores taxas de compressão.

	Fonte	*Bits*
Cabeçalho (1.351 *bits*)	Alfabeto utilizado	26
	Definição do modelo	24
	Partição estimada	1.021
	Codificações condicionais	260
	Codificações marginais	20
	Mensagem codificada	58.385
	Total	59.736

Tabela 3.6: Elementos da *string* binária resultante da compressão do genoma da Covid-19 utilizando a abordagem de modelos de partição, e seus respectivos números de *bits*.

	Fonte	*Bits*
Cabeçalho (5.190 *bits*)	Alfabeto utilizado	26
	Definição do modelo	24
	Partição estimada	0
	Codificações condicionais	5.120
	Codificações marginais	20
	Mensagem codificada	58.100
	Total	63.290

Tabela 3.7: Elementos da *string* binária resultante da compressão do genoma da Covid-19 utilizando a abordagem de modelos completos, e seus respectivos números de *bits*.

3.2 Compressão de Matrizes

Nesta seção, estudamos o problema de um transmissor que tem interesse em comunicar uma matriz a um receptor. Após o desenvolvimento da seção anterior, onde apresentamos uma metodologia para comunicação das quantidades necessárias para a codificação de uma *string*, e do procedimento de conversão de uma matriz em *string* (H-leitura) apresentado na subseção 1.3.2, falta pouco para alcançarmos um método para compressão de matrizes.

Na subseção 1.3.2, vimos como associar um formato de vizinhança de um ponto da matriz a um conjunto de observações anteriores em sua H-leitura. Além disso, discutimos brevemente como esse processo pode ser visualizado como uma cadeia de Markov com interstício, em que cada vizinhança observada corresponde a um elemento do espaço de estados associado. Em particular, o formato de vizinhança ilustrado na figura 1.3 corresponde ao espaço de estados de uma cadeia de Markov com interstício com $g = 1$, $G = W$ e $M = 1$, em que W é o número de colunas da matriz. No entanto, um detalhe muito importante deve ser observado antes de iniciarmos o processo de compressão: em nossa abordagem, fixado um formato de vizinhança, esta é aplicável apenas a uma submatriz da matriz original, **B**. Por exemplo, se estamos utilizando o formato de vizinhança até aqui trabalhado, ele é aplicável na submatriz **B1**, mas não no conjunto de pontos da região **B2**, conforme delineados na figura 3.1.

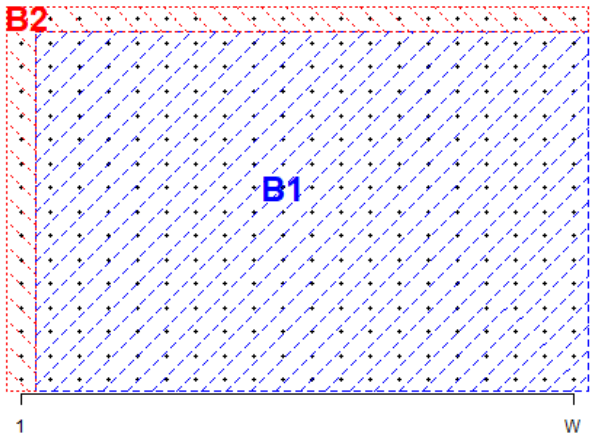

Figura 3.1: Visualização da região em que a vizinhança é aplicável.

Perceba que, para matrizes de grande dimensão, a quantidade de elementos em **B1** é muito maior que a em **B2**. Dessa forma, podemos justificar o uso de um método de codificação em **B1** diferente do utilizado em **B2**, desde que este produza um código com comprimento médio não superior. No entanto, este não é o real problema, pois, conforme o teorema 7, o condicionamento não aumenta a entropia, e, portanto, não aumenta os limites inferior e superior do comprimento esperado do código ótimo, ambos delineados no teorema 15. Assim, como observado para compressão de *strings* na subseção 3.1.1, o problema do uso de códigos condicionais está na transmissão da grande quantidade de parâmetros necessários para descrever o sistema de codificação utilizado. Ou seja, o modelo pode se tornar tão complexo que a taxa de compressão dos dados não justifique a inclusão de tantos parâmetros, ocasionando uma baixa eficiência na transmissão da mensagem como um todo.

Além das configurações iniciais análogas ao método de compressão de *strings*, a primeira etapa do procedimento de compactação de uma matriz $\mathbf{B}$, cujos elementos são provenientes de um alfabeto finito A, será sua H-leitura, ou seja, sua conversão em uma *string* T. Em seguida, devemos escolher o formato de vizinhança que utilizaremos para construir, posteriormente, as classes de condicionamento do código. A definição da vizinhança deve ser feita de forma a permitir sua associação com o espaço de estados de alguma cadeia de Markov conveniente. Por exemplo, uma vizinhança muito elaborada aumenta a dimensão do espaço de estados da cadeia de Markov associada ao modelo

(podendo torná-lo muito complexo), além de reduzir a região na qual pode ser aplicada.

A próxima etapa é referente à estimativa da partição do espaço de estados associada às observações da região **B1**. Para isso, devemos contabilizar as frequências condicionais nessa submatriz e executar o algoritmo A.1 de maneira apropriada. A seguir, devemos executar o algoritmo A.2 condicionado a cada elemento da partição, obtendo o conjunto das codificações condicionais. Em seguida, executamos o algoritmo A.2 sobre a região **B2** para a obtenção da codificação marginal que será utilizada nesses elementos. Por fim, codificamos a *string* T (proveniente da H-leitura de $\mathbf{B}$) utilizando a codificação marginal se o elemento pertencer à região **B2** ou a codificação condicional correspondente se o elemento pertencer à **B1**. O processo de codificação da matriz $\mathbf{B}$ é representado pelo algoritmo A.3 para uma vizinhança cuja equivalência é uma cadeia de Markov com interstício com $M = 1$, $G = W$ e $g = 1$, em que W é o número de colunas da matriz $\mathbf{B}$. O processo geral da transmissão da matriz $\mathbf{B}$ é ilustrado na figura B.3.

Exemplo 32. *Suponha o contexto de comunicação de uma matriz* $\mathbf{B}$ *e que transmissor e receptor já tenham estabelecido previamente o seguinte conjunto de configurações:*

- *Alfabeto prefixado:* $A = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$*;*
- *Bits reservados para a definição do modelo para a classe de condicionamento:* 4 *bits para o valor* g*,* 16 *bits para* $G - M$ *e* 4 *bits para* M*;*
- *Bits reservados para a declaração do número de linhas da matriz:* $R = 16$.

Relembre que, para nossa abordagem, o valor de G *já reflete o número de colunas da matriz* $\mathbf{B}$*. Os elementos da matriz são provenientes do alfabeto* $A_1 = \{a, d, g, t, u, z\}$ *e considere que* $\mathbf{B}$ *pode ser dividida em duas regiões,* **B1** *e* **B2***, conforme ilustrado na figura 3.1. Suponha que, na região* **B1***, as observações são influenciadas pela configuração de sua vizinhança, cujo formato está ilustrado na figura 1.3, e na região* **B2** *as observações são independentes de quaisquer outras e possuem distribuição uniforme em* A_1*. Denomine por* T *a string resultante da H-leitura de* $\mathbf{B}$*. Como já discutido anteriormente, a classe de condicionamento para a região* **B1** *pode ser vista como o espaço de estados de uma cadeia de Markov com interstício,* $\mathcal{S}^*$*, com* $g = 1$*,* $G = W$ *e* $M = 1$*, em que* W *é o número de colunas da matriz* $\mathbf{B}$*. Além disso, suponha que essa cadeia de Markov (restrita à região* **B1***) possua partição em seu espaço de estados como a delineada pela tabela 3.8 e*

probabilidades de transição dadas pela tabela 3.9. Repare que essa configuração do processo nos diz que, na região **B1**, *se um elemento de* A_1 *é predominante na vizinhança, temos uma alta probabilidade de permanecer nele ou em algum de seus vizinhos imediatos no alfabeto* A_1. *Esse tipo de comportamento é comum em matrizes que representam intensidades de cores, em que as transições acontecem de maneira pouco ríspida.*

Parte	Elementos
L_1	$aaa, aad, aag, aat, aau, aaz, ada, aga, ata, aua, aza, daa, gaa, taa, uaa, zaa$
L_2	$ddd, dda, ddg, ddt, ddu, ddz, dad, dgd, dtd, dud, dzd, add, gdd, tdd, udd, zdd$
L_3	$ggg, ggd, gga, ggt, ggu, ggz, gdg, gag, gtg, gug, gzg, dgg, agg, tgg, ugg, zgg$
L_4	$ttt, ttd, ttg, tta, ttu, ttz, tdt, tgt, tat, tut, tzt, dtt, gtt, att, utt, ztt$
L_5	$uuu, uud, uug, uut, uua, uuz, udu, ugu, utu, uau, uzu, duu, guu, tuu, auu, zuu$
L_6	$zzz, zzd, zzg, zzt, zzu, zza, zdz, zgz, ztz, zuz, zaz, dzz, gzz, tzz, uzz, azz$
L_7	$\mathcal{S}^* \setminus \{L_1, \ldots, L_6\}$

Tabela 3.8: Partição $\mathcal{L}$ do espaço de estados estendido $\mathcal{S}^*$ do exemplo 32.

Parte (L)	$p(a\|L)$	$p(d\|L)$	$p(g\|L)$	$p(t\|L)$	$p(u\|L)$	$p(z\|L)$
L_1	0,50	0,20	0,10	0,10	0,05	0,05
L_2	0,20	0,40	0,20	0,10	0,05	0,05
L_3	0,10	0,20	0,40	0,20	0,05	0,05
L_4	0,05	0,05	0,20	0,40	0,20	0,10
L_5	0,05	0,05	0,10	0,20	0,40	0,20
L_6	0,05	0,05	0,10	0,10	0,20	0,50
L_7	0,30	0,10	0,10	0,10	0,10	0,30

Tabela 3.9: Probabilidades de transição de cada parte da partição $\mathcal{L}$ do exemplo 32.

Simulamos uma matriz $\mathbf{B}$ *de dimensão* 400×400 *seguindo as características até aqui desenvolvidas. Primeiramente, façamos a etapa de declaração do alfabeto. Como no exemplo 26, o método de listagem é o mais econômico, utilizando 26 bits. Dessa forma, escolhemos a string binária proveniente da listagem,* $\mathcal{A}_0$, *e a concatenamos com o bit de identificação com valor 0, obtendo* $\mathcal{A}^* = 0\mathcal{A}_0$. *Em seguida, precisamos declarar a tripla* $(g, G-M, M)$ *utilizada no modelo G3M, assim como o número de linhas da matriz* $\mathbf{B}$. *Em nosso caso,*

$$\mathcal{E} = I(1,4)I(399,16)I(1,4)I(400,16) = 0001\,0000000110001111\,0001\,0000000110010000.$$

Agora, precisamos estimar a partição $\mathcal{L}$, *a partir da string* T, *restrita à região* **B1**, *utilizando o algoritmo A.1. Para esta amostra, o algoritmo estimou a partição* $\widehat{\mathcal{L}} = \{\widehat{L_1}, \ldots, \widehat{L_8}\}$, *cujas partes são detalhadas na tabela 3.10.*

Parte	Elementos
$\widehat{L_1}$	$aaa, aad, aag, aat, aau, aaz, ada, aga, ata, aua, aza, daa, gaa, taa, uaa, zaa$
$\widehat{L_2}$	$ddd, dda, ddg, ddt, ddu, ddz, dad, dgd, dtd, dud, dzd, add, gdd, tdd, udd, zdd$
$\widehat{L_3}$	$ggg, ggd, gga, ggt, ggu, ggz, gdg, gag, gtg, gug, gzg, dgg, agg, tgg, ugg, zgg$
$\widehat{L_4}$	$ttt, ttd, ttg, tta, ttu, ttz, tdt, tgt, tat, tut, tzt, dtt, gtt, att, utt, ztt$
$\widehat{L_5}$	$uuu, uud, uug, uut, uua, uuz, udu, ugu, utu, uau, uzu, duu, guu, tuu, auu, zuu$
$\widehat{L_6}$	$zzz, zzd, zzg, zzt, zzu, zza, zdz, zgz, ztz, zuz, zaz, dzz, gzz, tzz, uzz, azz$
$\widehat{L_7}$	$dga, zga, zta, tua, tad, agd, tgd, zgd, azd, uag, zdg, dtg, tug, dzg, tzg, dat$ $gat, udt, agt, dzt, dau, tdu, zdu, agu, gzu, gaz, taz, tdz, ugz, atz, dtz, gtz$
$\widehat{L_8}$	$\mathcal{S}^* \setminus \{\widehat{L_1}, \ldots, \widehat{L_7}\}$

Tabela 3.10: Partição estimada $\widehat{\mathcal{L}}$ do espaço de estados estendido $\mathcal{S}^*$ do exemplo 32.

Repare que o algoritmo A.1 estimou uma partição de oito partes. Perceba que as partes L_1 a L_6 do modelo original foram recuperadas com exatidão como as partes $\widehat{L_1}$ a $\widehat{L_6}$, respectivamente, da partição estimada. No entanto, a parte L_7 do modelo original não foi completamente recuperada e apresentou-se dividida em dois conjuntos da partição estimada: $\widehat{L_7}$ e $\widehat{L_8}$. Este fenômeno é equivalente ao já constatado e comentado no exemplo 26, em que foram observados alguns erros de alocação de elementos do espaço de estados dentro da partição estimada. Também de forma análoga à realizada no exemplo 26, utilizamos um código de Huffman para a codificação da partição, considerando as estratégias Modo 1 e Modo Árvore. Ainda reaproveitando os cálculos realizados no exemplo 26, defina $|\widehat{\mathcal{L}}|_B$ a representação binária em 8 bits do número de partes em $\widehat{\mathcal{L}}$. Utilizando o Modo 1 para a declaração do código de Huffman para as partes obtemos uma string binária com 71 bits, enquanto para o Modo Árvore foram necessários 47 bits. Como o Modo Árvore foi o mais econômico, o escolhemos e adicionamos um bit de identificação inicial com valor 0 e definimos a string binária $\widehat{\mathcal{L}}_{modelo}$. Em seguida, codificamos os elementos do espaço de estados estendido segundo o código de Huffman considerado, confeccionando a string binária $\widehat{\mathcal{L}}_B$, que, em nosso caso, possui 568 bits. Finalizando o processo de declaração da partição, concatenamos todas essas informações em apenas uma única string: $\widehat{\mathcal{L}}^*_B = |\widehat{\mathcal{L}}|_B\, \widehat{\mathcal{L}}_{modelo}\, \widehat{\mathcal{L}}_B$, que é constituída por 8+48+568 = 624 bits.

O próximo passo é a execução sequencial do algoritmo A.2 para obtenção das codificações condicionadas a cada elemento da partição $\widehat{\mathcal{L}}$ que serão utilizadas para codificar a região **B1**. *A tabela 3.11 apresenta os códigos de Huffman condicionais para cada elemento da partição $\widehat{\mathcal{L}}$.*

Parte (L)	$C(a\|L)$	$C(d\|L)$	$C(g\|L)$	$C(t\|L)$	$C(u\|L)$	$C(z\|L)$
$\widehat{L_1}$	0	10	1110	1111	1100	1101
$\widehat{L_2}$	10	11	01	001	0001	0000
$\widehat{L_3}$	010	10	11	00	0111	0110
$\widehat{L_4}$	11011	11010	10	0	111	1100
$\widehat{L_5}$	0100	0101	011	00	11	10
$\widehat{L_6}$	1100	1101	100	101	111	0
$\widehat{L_7}$	11	011	000	001	010	10
$\widehat{L_8}$	10	000	001	011	010	11

Tabela 3.11: Códigos de Huffman condicionais a cada parte da partição $\widehat{\mathcal{L}}$ do exemplo 32.

Cada um dos códigos condicionais deve ser transmitido utilizando o Modo 1 ou Modo Árvore. Todas as codificações apresentaram o Modo Árvore mais econômico que o Modo 1. Concatenando um bit de identificação de valor 0 em cada uma dessas strings binárias, obtemos as strings $\mathcal{D}^*_{1,L}$, $L \in \widehat{\mathcal{L}}$. Por fim, definimos como $\mathcal{D}^*_1$ a concatenação das strings $\mathcal{D}^*_{1,L}$ em uma única string binária. Em nosso caso, $\mathcal{D}^*_1$ é representada por 240 bits.

Em seguida, devemos construir um código de Huffman associado às frequências da observações da região **B2**. *A codificação utilizada é*

$$C(a) = 01, \quad C(d) = 101, \quad C(g) = 100,$$
$$C(t) = 111, \quad C(u) = 00, \quad C(z) = 110.$$

Para a transmissão do código marginal, consideraremos novamente o Modo 1 (que utiliza 70 bits) e o Modo Árvore (que utiliza 29 bits). Como o modo árvore é o mais econômico, o escolhemos e comunicamos nossa decisão incluindo um bit de identificação com valor 0. A identificação do código marginal é denotada por $\mathcal{D}^*_2$ e possui, portanto, 30 bits. Com isso, podemos definir o cabeçalho de nossa mensagem codificada como a concatenação $\mathcal{U} = \mathcal{A}^*\mathcal{E}\widehat{\mathcal{L}}^*_B\mathcal{D}^*_1\mathcal{D}^*_2$, que possui 27+40+624+240+30 = 961 bits.

*Definimos um código de prefixo ótimo para a H-leitura T como a união dos códigos marginal (para a região **B2**) e condicionais (para a região **B1**), ou seja, $C^* = \{C(\cdot)\} \cup \{C(\cdot\,|L) : L \in \widehat{\mathcal{L}}\}$. Denominemos por $C^*(T)$ o código de prefixo ótimo C^* aplicado a T. Em nosso caso, $C^*(T)$ é uma string de 363.748 bits. Por fim, realizamos a concatenação $\mathcal{T} = UC^*(T)$, que une o cabeçalho e a mensagem codificada. Dessa forma, temos que a transmissão da mensagem T necessitou de 961+363.748 = 364.709 bits para ser realizada.*

A decodificação de T^* na matriz $\mathbf{B}$ *segue um procedimento análogo ao caso em que a mensagem original é uma string, tendo apenas um último passo adicional: fracionar a mensagem em linhas com G colunas (resgatando a dimensão original de* $\mathbf{B}$*). Novamente, não iremos realizar a decodificação aqui para não tornar o exemplo ainda mais exaustivo.*

Exemplo 33. *Retornemos para o contexto do exemplo 32. Para efeito de comparação, consideremos um procedimento de compressão da matriz* $\mathbf{B}$ *baseado em um código de Huffman tradicional, ou seja, sem levar em consideração possíveis estruturas de dependência entre os pontos de* $\mathbf{B}$*. Para este tipo de procedimento, transmissor e receptor necessitam de uma configuração prévia composta por: alfabeto A; e números máximos de linhas e colunas de* $\mathbf{B}$*. Além disso, deve-se comunicar o código de Huffman utilizado. Dessa forma, o cabeçalho é constituído por: alfabeto utilizado A_1 (26 bits, conforme já calculado); número de colunas de* $\mathbf{B}$ *(16 bits); número de linhas de* $\mathbf{B}$ *(16 bits); e código de Huffman utilizado (30 bits, pelo Modo Árvore). Por tanto, o cabeçalho é constituído por 26+16+16+30 = 88 bits, número muito inferior ao encontrado para o cabeçalho do exemplo 32. Para as frequências na matriz* $\mathbf{B}$*, o código de Huffman obtido é*

$$C(a) = 00, \quad C(d) = 100, \quad C(g) = 110,$$
$$C(t) = 111, \quad C(u) = 101, \quad C(z) = 01.$$

Com esta codificação, a H-leitura de $\mathbf{B}$ *é convertida em uma string binária de 401.614 bits. Portanto, este procedimento utiliza 88+401.614 = 401.702 bits para comunicar a matriz* $\mathbf{B}$*.*

Antes de finalizarmos o capítulo, algumas observações são convenientes para a aplicação dos procedimentos. Pelos exemplos 32 e 33, vimos que um procedimento que leva em consideração possíveis estruturas de dependência entre entradas de uma matriz possui um cabeçalho mais extenso que o que não considera, devido à necessidade de detalhamento do modelo utilizado. No entanto, esse custo adicional pode ser compensado pela capacidade de compressão ganha pelo delineamento de boas classes de condicionamento. Nos exemplos supracitados, o procedimento que utiliza o modelo G3M necessita de uma *string* binária 10% menor que o outro para transmitir a mesma matriz, conforme pode ser observado na tabela 3.12.

Uma observação de natureza prática é referente ao uso do algoritmo A.1. Em muitos casos, nem todos os elementos do espaço de estados estendido, $\mathcal{S}^*$, se apresentam

	Modelo G3M	Modelo independente
Cabeçalho	961	88
Mensagem codificada	363.748	401.614
Total	364.709	401.702

Tabela 3.12: Blocos das *strings* binárias representantes das matrizes compactadas nos exemplos 32 (modelo G3M) e 33 (modelo independente) e seus respectivos números de *bits*.

de maneira significativa em uma mensagem. A pouca frequência desses elementos pode prejudicar a estimação de suas probabilidades de transição e, consequentemente, prejudicar o desempenho do algoritmo A.1 no delinemanto da partição do espaço de estados. Uma alternativa é pré-alocar esses estados em uma única parte separada dos demais e depois executar o algoritmo A.1. Como a frequência desses elementos é pequena na mensagem, a utilização de uma codificação condicional não necessariamente ótima para esse grupo deve impactar pouco a compressão geral. Iremos utilizar nos capítulos seguintes um limitante *ad hoc* para esta causa: agruparemos todos os estados cuja frequência seja menor ou igual a $\alpha = 20 \times |A_1|$. Isso permitirá que a codificação seja eficiente, pelo menos, em classes de condicionamento de frequências relevantes.

3.3 Conclusão

Neste capítulo, apresentamos com detalhes procedimentos para codificação de dados uni e bidimensionais utilizando códigos de Huffman. Expomos que a *string* binária resultante da codificação pode ser separada em dois grandes blocos: cabeçalho; e mensagem codificada. O cabeçalho é responsável por comunicar ao receptor todas as características e informações sobre a codificação utilizada sobre a mensagem alvo da transmissão, enquanto o segundo bloco corresponde exatamente à *string* binária consequente do código delineado no cabeçalho aplicado à mensagem original. Na seção 3.1, descrevemos um conjunto de técnicas capazes de transmitir, sem duplicidade de decodificação, todas as quantidades envolvidas no cabeçalho. Evidenciamos, a partir de um conjunto de exemplos, que, dentre as ferramentas apresentadas, não há uma que se destaque como uniformemente superior, sendo conveniente a intervenção do programador para a seleção em cada etapa.

Na subseção 3.1.1, discutimos os benefícios de considerarmos codificações de Huffman condicionais para a compressão de *strings*. Argumentamos que, pelo fato do

condicionamento possibilitar uma redução da entropia associada a uma variável aleatória (teorema 7), o uso de codificações condicionais pode proporcionar códigos de menores comprimentos médios. Com essa motivação, realizamos um estudo comparativo de exemplos simulados utilizando as abordagens de cadeia de Markov completa e modelos de partição, apontando quando uma possui vantagens sobre a outra. Além disso, apresentamos indícios sobre a relação entre o BIC associado a um modelo markoviano de partição e sua taxa de compressão. Atualmente, estudamos a base teórica que levou à superioridade dos modelos de partição em nossas simulações. Também, realizamos um estudo de caso utilizando o genoma da Covid-19 apresentado no capítulo 1, mostrando que, devido à menor quantidade de parâmetros a serem transmitidos, tal abordagem possuiu uma melhor taxa de compressão que sua concorrente.

A seção 3.2 elucida como, a partir do uso de cadeias de Markov com interstício, podemos enfrentar a problemática de compactação de matrizes como uma variação do processo de compactação de *strings*, permitindo a aplicação do ferramental desenvolvido na seção 3.1 com apenas algumas sutís alterações.

A partir dos exemplos das subseções 3.1.1 e 3.1.2, podemos observar que um dos empecilhos para o uso dos códigos de Huffman via modelos de partição está na extensão do cabeçalho necessário, em particular, os trechos correspondentes à comunicação da partição estimada pelo modelo markoviano mínimo e dos códigos de Huffman operados (isso se tornará mais evidente no capítulo 4). Assim, maneiras mais eficientes de transmissão dessas informações são sempre desejadas. Atualmente, estudamos métodos para a transmissão da partição estimada como estruturas de árvores $|A|$-árias (cada nó interno possui $|A|$ filhos), em que A representa o alfabeto do processo associado. Essa estratégia tem se mostrado eficiente para estruturas específicas do espaço de estados. Além disso, procedimentos alternativos para a transmissão dos códigos de Huffman também são investigados.

Capítulo 4

Compressão de Imagens

O desenvolvimento de técnicas para manipulação de imagens está historicamente atrelado à compreensão de fenômenos físicos (como interação de sinais com diferentes comprimentos de onda) e biológicos (como o funcionamento da visão tricromática em seres humanos e outros primatas). A união dessas áreas de conhecimento permitiu a elaboração de ferramentas matemáticas capazes de construir modelos que gerem espectros de cores adequados à percepção do olho humano, baseados em apenas algumas poucas componentes primárias. A compressão de imagens é uma subárea da compactação de dados, cujo objetivo é a codificação de imagens digitais a fim de reduzir o custo de seu armazenamento e transmissão. Sua evolução permitiu a difusão do uso de imagens na *Internet* e em diversos aparelhos eletrônicos, como máquinas fotográficas digitais e celulares.

Diversos procedimentos foram desenvolvidos com o propósito de otimizar a codificação de imagens, e seus usos variam conforme o objetivo visado (veja [20] e [21], por exemplo). Neste capítulo, temos interesse em aliar os resultados trabalhados no capítulo 1 sobre modelos de partição e os resultados de teoria de informação apresentados no capítulo 2 para construir aproximações de imagens. Além disso, utilizamos os métodos de compactação de matrizes delineados no capítulo 3 para a construção de procedimentos de compressão de imagens com perda de qualidade, ou seja, a compressão ocorre sobre uma aproximação da imagem original. Na seção 4.1, introduzimos brevemente dois dos modelos de cores mais conhecidos em processamento de imagens (RGB e CMYK) e vemos como se relacionam com o problema de compactação de matrizes. Na seção 4.2, apresentamos algumas possibilidades para a compressão de imagens com perda. Discutimos sobre a

dificuldade de sua aplicação devido ao complexo delineamento da partição estimada para o uso das codificações condicionais, e, na subseção 4.2.1 apresentamos algumas estratégias para sua simplificação. Na seção 4.3, discutimos como alguns formatos de arquivos lidam com o armazenamento e compressão de imagens.

4.1 Modelos de Cores

Modelos de cores são conceitos abstratos que estabelecem modelos matemáticos que visam representar cores a partir de um conjunto de componentes que são, geralmente, associadas a diferentes conjuntos de comprimentos de onda. O conjunto das possíveis cores geradas pelo modelo é o que chamamos de espaço de cores. A escolha das componentes base para o espaço de cores deve ser feita a fim de representar de maneira eficiente o espectro perceptível ao olho humano. Estudos sobre efeitos ópticos datam da antiguidade, mas, apenas no século XIX, Thomas Young postulou a existência de três tipos de células fotoreceptoras no olho humano, cada uma sensível a um específico comprimento de onda (veja [31]). Definiu-se, então, o conceito de visão tricromática, que baseia-se na construção de uma gama de cores visíveis baseada no estímulo de três tipos de células fotoreceptoras (células cone). Em particular, no olho humano, cada uma das células cone atinge responsividade máxima nas cores amarela (cones L), verde (cones M) ou violeta (cones S), e a composição desse triestímulo é passada e interpretada pelo cérebro como uma única cor.

Apesar de cada tipo de cone ser mais sensível a um comprimento de onda específico, a visualização de determinada cor (sinal entrando pelo olho) provoca estímulos de diferentes intensidades em cada tipo de cone. Contudo, existem cores em que as responsividades das células receptoras são muito próximas. Em particular, os cones L e M possuem um comportamento parecido em uma área substancial do espectro visível. Dessa forma, utilizar as cores de responsividade máxima das células receptoras como as componentes base para um modelo de cores pode não ser o mais conveniente para a construção do espaço de cores apropriado. Pensando nisso, o modelo RGB (do inglês *Red, Green and Blue*) utiliza como cores primárias (elementos base) as cores vermelha, verde e azul, em que cada uma estimula um dos três tipos de células cone e proporciona o mínimo de estímulo possível nos outros dois. As cores formadas pelo modelo RGB são obtidas a

partir da sobreposição de suas cores primárias, cada uma com sua respectiva intensidade. Devido à ideia de sobreposição, o modelo RGB é caracterizado como "aditivo", pois os comprimentos de onda de três componentes são adicionados para compor a cor final. O espaço de cores gerado pelo modelo RGB promove a cobertura de uma vasta parte do espectro visível ao ser humano.

Outro modelo de cores tradicional é o chamado CMYK (do inglês *Cyan, Magenta, Yellow and Key*), baseado em quatro cores primárias: ciano; magenta; amarela; e preta. Este modelo é, geralmente, utilizado no contexto de impressões físicas, em que se deseja mascarar (cobrir) um fundo de cor branca. Dessa forma, cada componente reduz a intensidade da luz que seria refletida pelo fundo. A ideia de que as componentes "subtraem" cores do substrato branco faz com que o modelo CMYK seja caracterizado como "subtrativo". Neste texto, estudamos o modelo RGB, uma vez que tratamos da representação digital de imagens, ou seja, elas são visualizadas por emissão de uma tela escura, em geral. No entanto, os procedimentos discutidos podem ser diretamente estendidos para o modelo CMYK.

Consideremos uma imagem base que desejamos representar por um número finito de elementos. Suponhamos que possamos cobrir a imagem com uma malha de pontos com espaçamentos regulares. Para cada interseção entre ponto e imagem, atribua a cor do espaço de cores RGB referente àquela localidade. Dessa forma, a imagem original passa a ser representada por um conjunto finito de pontos pertencentes ao espaço de cores RGB. Como discutido anteriormente, o espaço de cores RGB pode ser decomposto em três componentes (doravante denominadas R, G e B), em que cada uma representa a intensidade de um feixe luminoso característico. Assim, cada ponto da malha passa a ser representado por uma tripla de intensidades, (R,G,B). Apesar de estarmos trabalhando no campo de quantidades contínuas (intensidades e comprimentos de onda), como estamos interessados em uma representação factível digitalmente, é necessária a adaptação para registros discretos. Em delineamentos de sistemas digitais, é interessante ter unidades que são potência de dois, uma vez que trabalhamos com *bits*. Em 1993 (veja [19]), foi estabelecida a padronização que um octeto de *bits* é uma unidade de informação digital de tamanho conveniente. Assim, consideramos que cada intensidade pode assumir $2^8 = 256$ valores distintos (enumerados de 0 a 255) em que 0 corresponde à intensidade mínima e 255 à intensidade máxima da componente na cor resultante.

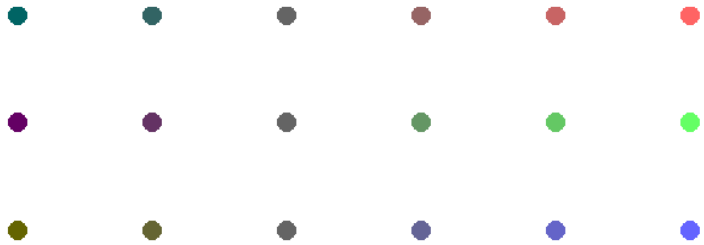

Figura 4.1: Visualização de uma grelha de cores formada por elementos do espaço de cores RGB.

Nessas condições, no modelo RGB, quando todas as intensidades forem 0, obtemos a cor preta e, quando todas forem 255, obtemos a cor branca. Observe que 2^8 possibilidades para cada componente nos possibilita uma gama de $2^{24} = 16.777.216$ cores diferentes.

Exemplo 34. *A grelha de pontos apresentada pela figura 4.1 é representada pelo conjunto de triplas do modelo RGB*

(0,100,100) (50,100,100) (100,100,100) (150,100,100) (200,100,100) (255,100,100)
(100,0,100) (100,50,100) (100,100,100) (100,150,100) (100,200,100) (100,255,100)
(100,100,0) (100,100,50) (100,100,100) (100,100,150) (100,100,200) (100,100,255).

Perceba que, de acordo com as triplas, na primeira linha, a intensidade da componente vermelha aumenta conforme avançamos para a direita. Na segunda linha, a intensidade aumenta na componente verde e, na terceira, na componente azul. Esse aumento das intensidades nas cores resultantes é facilmente perceptível nas linhas da grelha da figura 4.1.

Note que uma grelha de pontos pertencentes ao modelo RGB, ou seja, em que cada entrada é representada por uma tripla (R,G,B), pode ser interpretada como a sobreposição de três malhas primárias, em que cada uma assume os valores de uma componente base. Assim, uma grelha de pontos RGB é vista como a sobreposição das malhas **R**, **G** e **B**, que representam, respectivamente, as intensidades das bases R, G e B em cada ponto da grelha.

Exemplo 35. *A grelha de pontos do exemplo 34 é representada pelas malhas primárias:*

$$\mathbf{R} = \begin{bmatrix} 0 & 50 & 100 & 150 & 200 & 255 \\ 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 & 100 \\ 0 & 50 & 100 & 150 & 200 & 255 \\ 100 & 100 & 100 & 100 & 100 & 100 \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} 100 & 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 & 100 \\ 0 & 50 & 100 & 150 & 200 & 255 \end{bmatrix}.$$

Naturalmente, podemos interpretar as malhas **R**, **G** e **B** como três matrizes cujos elementos são provenientes do alfabeto $A = \{0, \ldots, 255\}$. Dessa forma, podemos encarar a problemática de transmitir uma imagem como a de transmissão de três matrizes independentes.

Nota 14. *Neste texto, trabalhamos com as matrizes* **R**, **G** *e* **B** *de maneira independente, mas há a possibilidade de se utilizar a associação entre cores para a construção de funcionais de componentes base (veja, por exemplo, [17]). Outra abordagem é a decomposição do espaço de cores tradicional em componentes não correlacionadas. Novamente, as técnicas que utilizamos aqui podem ser estendidas diretamente para qualquer contexto que utilize a transmissão de uma ou mais matrizes. Assim sendo, fica a critério do programador incorporar características extras em sua abordagem.*

Note que cada uma das matrizes propostas pode requerer uma grande porção do alfabeto total A, senão sua totalidade. Essa característica pode ser um problema quando queremos transmitir uma imagem sem uma perda substancial em sua qualidade, ou seja, queremos transmitir uma aproximação muito fiel da imagem original. As seções seguintes abordam os paradigmas de transmissão de dados em que há uma perda (prevista) de informação, que impossibilita a recuperação do conjunto de dados original de maneira integral.

4.2 Transmissão de Imagens

Consideremos o processo de transmissão de uma imagem que, conforme discutido na seção anterior, pode ser visto como a transmissão de uma ou mais matrizes. Um ponto importante, também levantado na seção 4.1, é o tamanho do alfabeto necessário para representar cada uma das matrizes em sua totalidade. Em geral, temos interesse em representar qualquer conjunto de informações da maneira mais econômica possível.

No entanto, devido às características dos dados (como distribuição de frequências das observações ou tamanho do alfabeto), é possível que processos de compressão tradicionais sejam muito custosos computacionalmente, ou não produzam uma economia relevante no tamanho da mensagem codificada. Uma abordagem para contornar esta questão é a construção de uma aproximação dos dados originais que possua estrutura propícia aos processos de compactação. Dessa forma, consideramos dois gêneros de compressão: compressão sem perda, em que, após passar pelo processo de compressão e descompressão, uma mensagem é perfeitamente recuperada; e compressão com perda, em que, após passar pelo processo de compressão e descompressão, uma aproximação da mensagem original é obtida. Os exemplos e procedimentos desenvolvidos até aqui trataram de compressões sem perda.

A fim de transmitir um conjunto de dados de maneira econômica, a abordagem de compressão com perda realiza uma etapa prévia à codificação, em que uma aproximação da mensagem original é gerada, a qual será de fato transmitida. Tal aproximação deve promover uma nova estrutura no conjunto de dados que proporcione uma codificação mais eficiente, mas que ainda permita que a nova mensagem, após a decodificação, seja suficientemente próxima à original (as quantificações sobre "suficientemente próxima" não são o foco deste trabalho). Essa etapa foi abordada de diferentes formas na literatura, como a utilização de arredondamentos em transformadas (por exemplo, a transformada discreta de cosseno introduzida em [1], e a transformada discreta de onduleta, que possui raízes nos trabalhos de [23]) e a simples redução do espaço de cores original para outro espaço reduzido. Como estamos interessados apenas na aplicação do ferramental estatístico desenvolvido nos capítulos anteriores, e não na comparação entre métodos de aproximação de imagens, nos atemos, por conveniência, ao método de redução do espaço de cores. Na literatura, o procedimento de simplificação do conjunto dos valores de um processo para outro conjunto de tamanho reduzido é chamado de quantização. Claramente, a simplificação do processo gera discrepâncias com a mensagem original, o que acarretará uma perda irreversível de informação na mensagem recuperada pelo decodificador (receptor). É importante ressaltar que a perda considerada acontece na aproximação da mensagem original realizada pelo codificador (transmissor) durante a quantização apenas, e não no procedimento de compactação em si, que é o foco do presente trabalho.

Prosseguimos esta seção tendo como base a figura 4.2, a qual denominamos

Figura 4.2: Visualização da imagem “Vaca”.

por “Vaca”. Essa imagem é formada por 273.920 pontos organizados em um retângulo com 428 pontos de altura e 640 de comprimento. Cada um dos 273.920 elementos é uma unidade básica da imagem e será denominado por *pixel*. Cada *pixel* emite um estímulo visual associado a uma cor. Em nosso caso, cada *pixel* assume um valor dentro do espaço de cores do modelo RGB. Baseado no discutido anteriormente, podemos considerar que cada *pixel* é representado por uma tripla de intensidades (R,G,B), e que a imagem “Vaca” é composta pela sobreposição das matrizes de intensidades $\mathbf{R}$, $\mathbf{G}$ e $\mathbf{B}$, cada uma com dimensão 428×640. A decomposição da imagem “Vaca” em suas matrizes de intensidades é apresentada na figura 4.3.

É interessante comparar as figuras 4.2 e 4.3 para interpretarmos alguns efeitos visuais na composição das matrizes. Primeiramente, podemos notar que *pixels* da imagem “Vaca” que assumem cor branca possuem intensidade máxima (255) nas respectivas entradas das matrizes $\mathbf{R}$, $\mathbf{G}$ e $\mathbf{B}$. Analogamente, para a cor preta na imagem “Vaca”, temos entradas de valores 0 nas matrizes de intensidades. Outro fenônomeno interessante ocorre quando inspecionamos a decomposição de uma cor próxima a um elemento da base. Por exemplo, a cor azul presente no céu da imagem possui valor máximo (ou próximo a 255) na componente B, mas valores consideravelmente mais baixos nas componentes R e G.

Para iniciarmos o procedimento de compactação com perda, devemos definir qual quantização será utilizada para a aproximação da imagem “Vaca”. Como mencionado anteriormente, por simplicidade, iremos trabalhar com cada matriz de intensidades individualmente, dessa forma, as matrizes $\mathbf{R}$, $\mathbf{G}$ e $\mathbf{B}$ serão aproximadas separadamente. De-

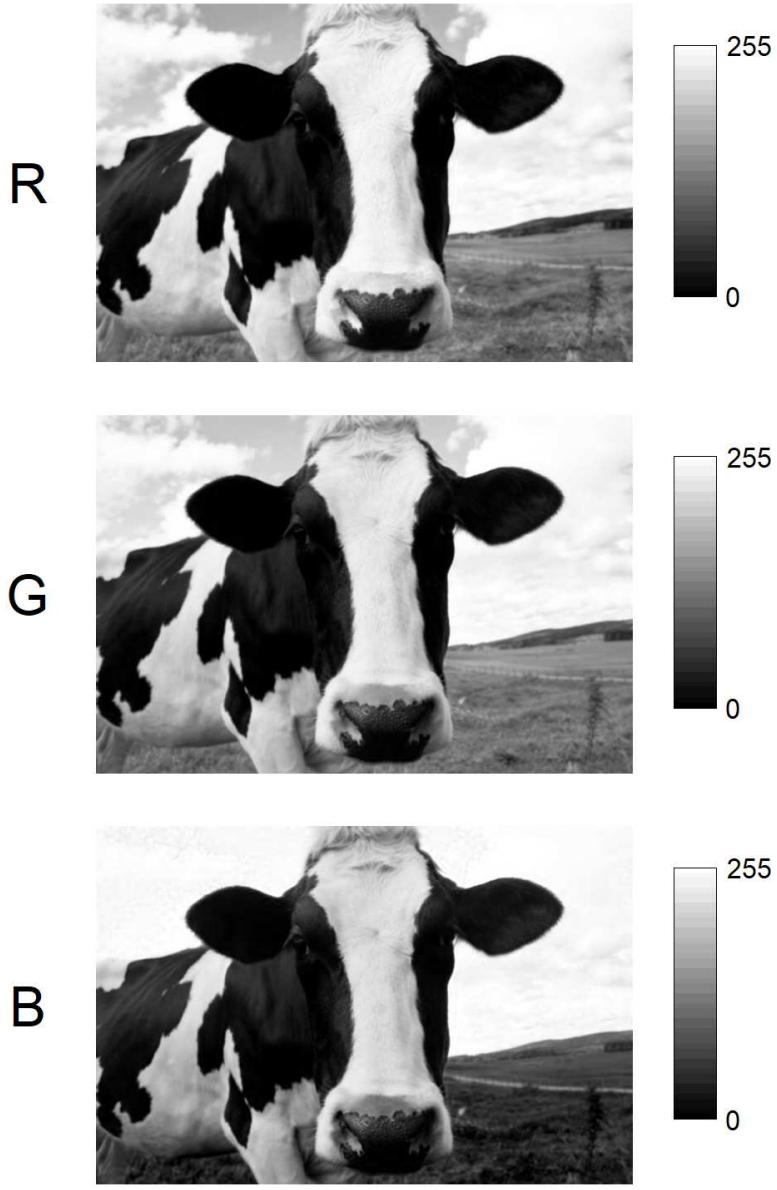

Figura 4.3: Visualização das matrizes de intensidades da imagem “Vaca”.

talharemos todo o procedimento para a matriz **R** e, para as demais matrizes, os resultados serão apresentados de maneira simplificada.

Conforme já definimos, a quantização ocorrerá a partir de técnicas de redução do espaço de intensidades original para outro de dimensão reduzida. Dois questionamentos naturalmente surgem para esta abordagem: quantas componentes definirão o novo conjunto de intensidades? Quem serão esses elementos? Antes de defini-los, é interessante visualizarmos a frequência com que cada intensidade ocorre na matriz **R**. A figura 4.4 apresenta a distribuição das frequências das intensidades na matriz **R**.

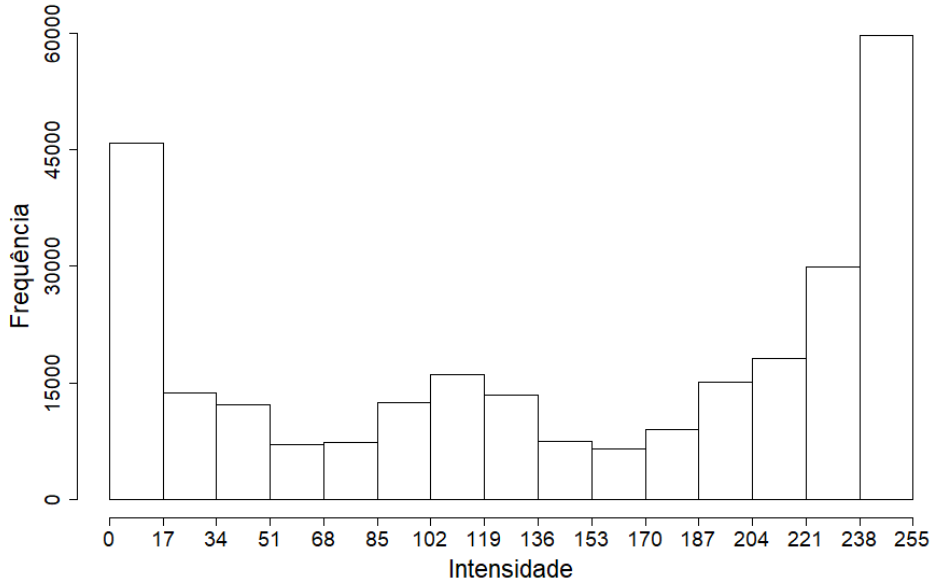

Figura 4.4: Distribuição das frequências das intensidades na matriz **R** da imagem "Vaca".

Notamos, a partir da figura 4.4, que há concentração de observações nos extremos da escala, ou seja, há muitos valores próximos de 0 e 255, mas uma concentração mais modesta em valores intermediários. Em uma imagem tradicional (não composta por ruídos apenas), é razoável a conjectura da existência de uma estrutura de dependência entre um ponto da matriz de intensidades e alguns pontos que o circundam. Essa suposição está associada à tendência de transições suaves de cores na imagem original (fenômeno muito comum em fotografias e pinturas), algo que, de fato, ocorre comumente no universo de processamento de imagens. Dessa maneira, temos interesse em incluir a estrutura vicinal dos dados em nosso processo de seleção do tamanho do novo espaço de intensidades. Em particular, vamos considerar o formato de vizinhança proposto na definição 17. Vale a pena relembrarmos que este formato de vizinhança é aplicável apenas a uma submatriz da matriz original, **B1**, conforme apresentado na figura 3.1. Consideremos a H-leitura da matriz **R**, conforme apresentada na definição 16, e a denotemos por $t_{\mathbf{R}}$, que será considerada uma amostra de um processo estocástico $T_{\mathbf{R}}$. Como já discutido nos capítulos anteriores, podemos utilizar um modelo $G3M(g = 1, G = W, M = 1)$, com W igual ao número de colunas da matriz **R**, para modelarmos a estrutura de dependência desejada.

Seja $A_{\mathbf{R}} = \{0, \ldots, 255\}$ o conjunto de valores presentes na *string* $t_{\mathbf{R}}$, ou seja, $A_{\mathbf{R}}$ é o alfabeto associado ao processo, com $|A_{\mathbf{R}}| = 256$. Inicialmente, temos interesse na mínima partição boa do espaço de estados associado ao processo em questão restrito à região

B1 (conforme a figura 3.1), $\mathcal{L}_{\mathbf{R}}$, e em $\widehat{\mathcal{L}_{\mathbf{R}}}$, a estimativa de $\mathcal{L}_{\mathbf{R}}$ dada pelo algoritmo A.1. Se retornarmos ao corolário 2, visualizamos que, dada uma partição $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$, duas partes boas, L_i e L_j, devem ser unidas se, e somente se, $d_{\mathcal{L}}(i,j) < 1$. Em nosso caso, temos um alfabeto extenso, o que torna o critério "frouxo", uma vez que a constante $\frac{2}{(|A|-1)}$, presente em $d_{\mathcal{L}}$, se torna muito pequena, o que dificulta o alcance dos resultados assintóticos e promove, consequentemente, uma estimativa ruim para a partição do espaço de estados. Além disso, para a matriz $\mathbf{R}$ observada, temos que o estado (0,0,0) é o único que possui pelo menos $\alpha = |A_{\mathbf{R}}| \times 20 = 5.120$ observações (critério introduzido no final do capítulo 3). Dessa forma, a partição estimada pelo algoritmo A.1, $\widehat{\mathcal{L}_{\mathbf{R}}}$, é composta apenas por duas partes: uma com apenas o estado (0,0,0); e outra com todos os demais estados. O resultado obtido é justificado pelo pequeno tamanho amostral quando comparado às exigências de nosso procedimento. Por isso, devemos ajustar nossos critérios para que o algoritmo A.1 possa realizar um maior número de comparações de partes via $d_{\mathcal{L}}$ sem perder a qualidade das estimativas observadas, ou seja, garantindo a propriedade de consistência apresentada no corolário 4.

Nosso interesse está em ajustar as constantes $\alpha = |A_{\mathbf{R}}| \times 20$ e $\beta = \frac{2}{(|A_{\mathbf{R}}|-1)}$ para quantidades mais apropriadas para inferência sobre o processo $T_{\mathbf{R}}$ a partir da amostra $t_{\mathbf{R}}$. Como o alfabeto $A_{\mathbf{R}}$ é extenso e a amostra $t_{\mathbf{R}}$ é limitada, observamos que cada elemento do espaço de estados do processo se comunica com um número consideravelmente pequeno de elementos de $A_{\mathbf{R}}$. Em particular, como $t_{\mathbf{R}}$ representa a leitura de intensidades de uma imagem, uma observação tende a possuir valores próximos àqueles observados em sua vizinhança. Esse fenômeno está novamente associado com as transições suaves de cores que comumente acontecem em imagens. Para a matriz $\mathbf{R}$ da imagem "Vaca", cada uma das 75.314 vizinhanças distintas se comunica com no máximo 10 elementos de $A_{\mathbf{R}}$. Isso quer dizer que, para cada elemento do espaço de estados, teremos estimado no máximo 10 probabilidades de transição diferentes de zero. Definamos, então, $\delta = \max_{s \in \mathcal{S}^*} |\{a \in A_{\mathbf{R}} : N_n(sa) > 0\}|$, em que $\mathcal{S}^*$ é o espaço de estados associado ao processo $T_{\mathbf{R}}$, ou seja, δ é o maior número de conexões distintas que um elemento do espaço de estados $\mathcal{S}^*$ realizou na amostra $t_{\mathbf{R}}$. Dessa maneira, construímos duas novas constantes $\alpha^{(\delta)} = \delta \times 20$ e $\beta^{(\delta)} = \frac{2}{(\delta-1)}$, que substituirão α e β, respectivamente. Como a escolha de α foi feita no intuito de obtermos uma quantidade mínima de observações para estimarmos $(|A_{\mathbf{R}}| - 1)$ probabilidades de transição, sua substituição por $\alpha^{(\delta)}$ parece intuitiva. A substituição

de β por $\beta^{(\delta)}$ é mais delicada, pois os resultados de consistência presentes no capítulo 1 estão relacionados a esta constante. No entanto, pelo exposto em [9], temos que a constante $\beta = \frac{2}{(|A_{\mathbf{R}}|-1)}$ pode ser generalizada para $\beta = \frac{\gamma}{(|A_{\mathbf{R}}|-1)}$, para qualquer γ real positivo, sem prejudicar a consistência do procedimento. Em particular, podemos tomar $\gamma = (|A_{\mathbf{R}}| - 1)\beta^{(\delta)}$ e trivialmente obter $\beta = \beta^{(\delta)}$. Executando novamente o algoritmo A.1, mas utilizando as constantes $\alpha^{(\delta)}$ e $\beta^{(\delta)}$, obtemos que 177 estados tiveram frequência superior a $\alpha^{(\delta)}$ e observamos uma partição com 69 partes. A alteração das quantidades α e β proporcionam regras de análise que se adaptam ao processo observado e permitem a observação de fenômenos que só seriam perceptíveis em uma amostra de tamanho elevado.

Agora, desejamos obter uma estimativa para o tamanho do alfabeto efetivo associado ao processo $T_{\mathbf{R}}$. Notemos que, se assumirmos que os elementos em **B2** são observações independentes, podemos estimar a taxa de entropia associada ao processo $T_{\mathbf{R}} = T_1 T_2 \dots T_{|\mathbf{R}|}$, como

$$\widehat{H}(T_{\mathbf{R}}) = \frac{\widehat{H}(T_1, \dots T_{|\mathbf{R}|})}{|\mathbf{R}|} = \sum_{i=1}^{|\mathbf{R}|} \frac{\widehat{H}(T_i|T_1, \dots, T_{i-1})}{|\mathbf{R}|} \tag{4.1}$$

$$= \sum_{i:T_i \in \mathbf{B2}} \frac{\widehat{H}(T_i)}{|\mathbf{R}|} + \sum_{i:T_i \in \mathbf{B1}} \frac{\widehat{H}(T_i|L_i)}{|\mathbf{R}|}, \tag{4.2}$$

em que (4.1) resulta da equação (2.2) e $L_i \in \widehat{\mathcal{L}}$ é o elemento da mínima partição boa estimada referente à vizinhança da observação T_i. Como $H(T_i) = -\sum_{a \in A} p(a) \log_2 p(a)$, em que $p(a) = \Pr(T_i = a)$, $a \in A$, então, para uma amostra $t_{\mathbf{R}}$ do processo $T_{\mathbf{R}}$, a estimativa de máxima verossimilhança de $H(T_i)$ restrita à região **B2** é $\widehat{H}(T_i) = -\sum_{a \in A} \widehat{p}_{\mathbf{B2}}(a) \log \widehat{p}_{\mathbf{B2}}(a)$, em que $\widehat{p}_{\mathbf{B2}}(a) = N_{\mathbf{B2}}(a)/|\mathbf{B2}|$, para todo $a \in A$, com $N_{\mathbf{B2}}(a) = |\{i : i \in \mathbf{B2},\ t_i = a\}|$. Para obtermos uma estimativa para $H(T_i|L_i)$, lembremos que, pela definição 20, para duas variáveis aleatórias discretas X e Y definidas sobre os alfabetos A_X e A_Y, a entropia condicional de Y dado X, $H(Y|X)$, é dada por

$$H(Y|X) = \sum_{x \in A_X} p_X(x) H(Y|X = x) = -\sum_{x \in A_X} \sum_{y \in A_Y} p_X(x) p_{Y|X}(y|x) \log p_{Y|X}(y|x).$$

Em nosso caso, Y é uma observação T_i do processo e X é o elemento da mínima partição boa estimada, L_i, referente à vizinhança da observação T_i. Dessa forma, fixada a

partição $\widehat{\mathcal{L}}$, a estimativa de máxima verossimilhança de $H(T_i|L_i)$ restrita à região **B1** é

$$\widehat{H}(T_i|L_i) = -\sum_{L\in\widehat{\mathcal{L}}}\sum_{a\in A}\frac{N_{\mathbf{B1}}(L)}{|\mathbf{B1}|}\frac{N_{\mathbf{B1}}(L,a)}{N_{\mathbf{B1}}(L)}\log\frac{N_{\mathbf{B1}}(L,a)}{N_{\mathbf{B1}}(L)} = -\sum_{L\in\widehat{\mathcal{L}}}\sum_{a\in A}\frac{N_{\mathbf{B1}}(L,a)}{|\mathbf{B1}|}\log\frac{N_{\mathbf{B1}}(L,a)}{N_{\mathbf{B1}}(L)},$$

em que $N_{\mathbf{B1}}(L) = \sum_{s\in L} N_{\mathbf{B1}}(s)$, com $N_{\mathbf{B1}}(s) = |\{i : i \in \mathbf{B1},\ t_{i-W-1}t_{i-W}t_{i-1} = s\ \}|$ e $N_{\mathbf{B1}}(L,a) = \sum_{s\in L} N_{\mathbf{B1}}(sa)$, com $N_{\mathbf{B1}}(sa) = |\{i : i \in \mathbf{B1},\ t_{i-W-1}t_{i-W}t_{i-1} = s, t_i = a\ \}|$. Substituindo as estimativas $\widehat{H}(T_i)$ e $\widehat{H}(T_i|L_i)$ em (4.2), temos

$$\widehat{H}(T_{\mathbf{R}}) = -\sum_{i:T_i\in\mathbf{B2}}\sum_{a\in A}\frac{1}{|\mathbf{R}|}\frac{N_{\mathbf{B2}}(a)}{|\mathbf{B2}|}\log\frac{N_{\mathbf{B2}}(a)}{|\mathbf{B2}|} - \sum_{i:T_i\in\mathbf{B1}}\frac{1}{|\mathbf{R}|}\sum_{L\in\widehat{\mathcal{L}}}\sum_{a\in A}\frac{N_{\mathbf{B1}}(L,a)}{|\mathbf{B1}|}\log\frac{N_{\mathbf{B1}}(L,a)}{N_{\mathbf{B1}}(L)}$$

$$= -|\mathbf{B2}|\sum_{a\in A}\frac{1}{|\mathbf{R}|}\frac{N_{\mathbf{B2}}(a)}{|\mathbf{B2}|}\log\frac{N_{\mathbf{B2}}(a)}{|\mathbf{B2}|} - |\mathbf{B1}|\frac{1}{|\mathbf{R}|}\sum_{L\in\widehat{\mathcal{L}}}\sum_{a\in A}\frac{N_{\mathbf{B1}}(L,a)}{|\mathbf{B1}|}\log\frac{N_{\mathbf{B1}}(L,a)}{N_{\mathbf{B1}}(L)}$$

$$= -\frac{1}{|\mathbf{R}|}\left[\sum_{a\in A}N_{\mathbf{B2}}(a)\log\frac{N_{\mathbf{B2}}(a)}{|\mathbf{B2}|} + \sum_{L\in\widehat{\mathcal{L}}}\sum_{a\in A}N_{\mathbf{B1}}(L,a)\log\frac{N_{\mathbf{B1}}(L,a)}{N_{\mathbf{B1}}(L)}\right]. \tag{4.3}$$

Para o caso da matriz $\mathbf{R}$ da imagem "Vaca", obtemos a estimativa $\widehat{H}(T_{\mathbf{R}}) \approx 6,2149$. Com este valor, podemos retornar ao conceito de alfabeto efetivo apresentado na equação (2.7) e propor que um alfabeto de tamanho $\left\lceil 2^{\widehat{H}(T_{\mathbf{R}})}\right\rceil = 75$ é suficiente para expressar as sequências pertencentes ao conjunto típico do processo. Denominemos este novo alfabeto (de intensidades) por $A^e_{\mathbf{R}}$, no qual imporemos duas condições: $A^e_{\mathbf{R}}$ é subconjunto de $A_{\mathbf{R}}$; e $|A^e_{\mathbf{R}}| = 75$. Resta-nos, agora, decidir exatamente quais serão os elemento que comporão o novo alfabeto reduzido, ou seja, precisamos definir qual o procedimento de quantização a ser utilizado. Realçamos, novamente, que não temos o intuito de comparar diferentes métodos de aproximação de imagens, mas apenas exemplificar a utilização dos resultados apresentados neste texto. Dessa forma, apresentamos apenas uma possibilidade para a quantização dentro das inúmeras existentes, ficando a escolha a critério do desenvolvedor.

Uma estratégia simples para a quantização é a separação da escala $0 - 255$ (ou mais precisamente $\min A_{\mathbf{R}} - \max A_{\mathbf{R}}$) em intervalos disjuntos e a atribuição de um único valor (representante) para todos os elementos pertencentes a um mesmo subconjunto. Dessa forma, a escolha do alfabeto $A^e_{\mathbf{R}}$ é vista como um processo de duas etapas: definição

dos intervalos; e escolha dos valores representantes. Poderíamos simplesmente dividir a escala $\min A_{\mathbf{R}} - \max A_{\mathbf{R}}$ em 75 intervalos de mesmo comprimento e utilizar os pontos médios dos intervalos (ou quaisquer outros de preferência) como representantes. No entanto, essa escolha ingênua deixa de lado a estrutura da distribuição das intensidades. Notemos, pela figura 4.4, que os dados possuem uma distribuição nada uniforme e que, em particular, há alta concentração de observações nos extremos da escala. Assim sendo, a utilização de intervalos regulares agregaria intensidades próximas com alta incidência em apenas um representante, o que acarretaria em alta taxa de erro devido à aproximação em uma parte substancial da imagem "Vaca".

Uma estratégia para amenizar o erro de quantização é utilizar características da distribuição das intensidades na escolha dos intervalos. Em nosso exemplo, utilizamos quantis da distribuição amostral como pontos centrais. Um algoritmo para o procedimento é apresentado no algoritmo A.4. Basicamente, iniciamos o algoritmo com 75 quantis de probabilidades igualmente espaçadas de 0 a 1. Se estes quantis formarem um conjunto com 75 valores distintos, aceite-os como pontos centrais. Caso contrário, considere 76 quantis de probabilidades igualmente espaçadas de 0 a 1. Esse processo é repetido até que um conjunto com 75 valores distintos seja obtido e este será aceito como o conjunto de pontos centrais. Estabelecidos esses valores, podemos definir os limites dos intervalos como o ponto médio entre pontos centrais. Por fim, precisamos estabelecer os representantes de cada um dos 75 intervalos. Aqui, utilizamos os próprios pontos centrais (ou, quando necessário, suas partes inteiras) como representantes, mas quaisquer outros valores dentro dos respectivos intervalos podem ser considerados (como a moda ou mediana restritas a seus respectivos subconjuntos). Uma explanação visual do procedimento de quantização para cinco pontos centrais é apresentada na figura 4.5.

Utilizando a quantização proposta, obtemos $\widehat{\mathbf{R}}$ e $t_{\widehat{\mathbf{R}}}$, as aproximações de $\mathbf{R}$ e $t_{\mathbf{R}}$, respectivamente. O algoritmo A.5 apresenta a sistematização do processo de quantização explanado até aqui. A figura 4.6 apresenta a visualização das intensidades propostas pela matriz $\widehat{\mathbf{R}}$ e a figura 4.7 apresenta um mapa do módulo do erro de aproximação causado pela quantização. Notamos que o maior erro observado possuiu módulo seis e que a maior concentração de erros acontece em regiões que possuem as intensidades com menor frequência na matriz $\mathbf{R}$, como é o caso de intensidades de valores intermediários, conforme observado na figura 4.3. Essa característica é herdada do procedimento de escolha do

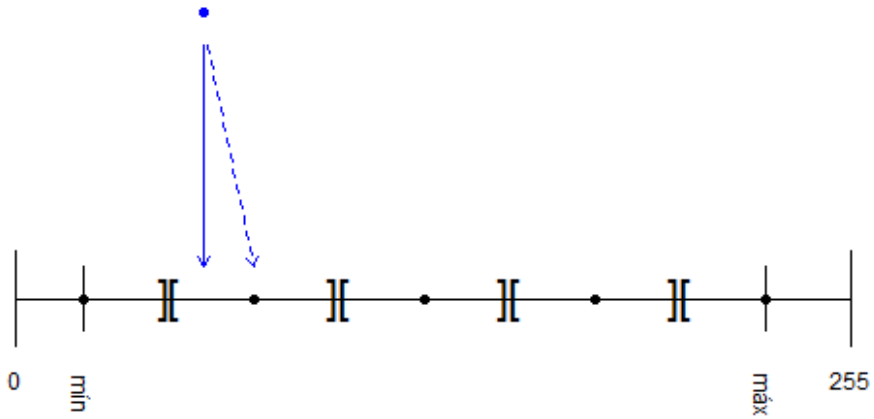

Figura 4.5: Ilustração do procedimento de quantização para cinco pontos centrais, em que a linha cheia representa a localização do valor original da observação e a linha tracejada representa a localização após a quantização.

alfabeto efetivo, que tende a valorizar regiões de alta densidade no alfabeto para a escolha dos representantes, tornando essas faixas de valores mais precisas que as demais.

Após a obtenção da matriz aproximada $\widehat{\mathbf{R}}$, resta-nos apenas realizar o processo de transmissão de matrizes apresentado na seção 3.2 e ilustrado na figura B.3. Para nosso caso, as configurações iniciais estabelecidas entre codificador e decodificador são:

- Alfabeto prefixado: $A = \{0, 1, \ldots, 255\}$;
- Modelo utilizado para a classe de condicionamento: $G3M(g,\, G,\, M)$ (4 *bits* para M e g, e 16 *bits* para G);
- Números de linhas e colunas da maior matriz a ser transmitida: $2^{16} - 1 = 65.535$ linhas e 65.535 colunas (16 *bits* para cada uma das quantidades).

Além disso, podemos realizar uma modificação na transmissão das codificações condicionais que leve em consideração o fato de que cada parte da partição estimada se comunica com poucos elementos do alfabeto utilizado (conforme já discutido). Para cada elemento L da partição estimada $\widehat{\mathcal{L}}$, consideramos a construção da codificação de Huffman utilizando apenas os elementos $a \in A^{e}_{\mathbf{R}}$ tais que $N_{\mathbf{B1}}(L, a) > 0$. Essa alteração permite que menores árvores binárias sejam transmitidas, provocando uma economia no tamanho da mensagem total. A modificação pode ser alcançada incluindo um prefixo de $\lfloor \log |A^{e}_{\mathbf{R}}| \rfloor + 1$

Figura 4.6: Visualização das intensidades para a matriz $\mathbf{R}$ após quantização.

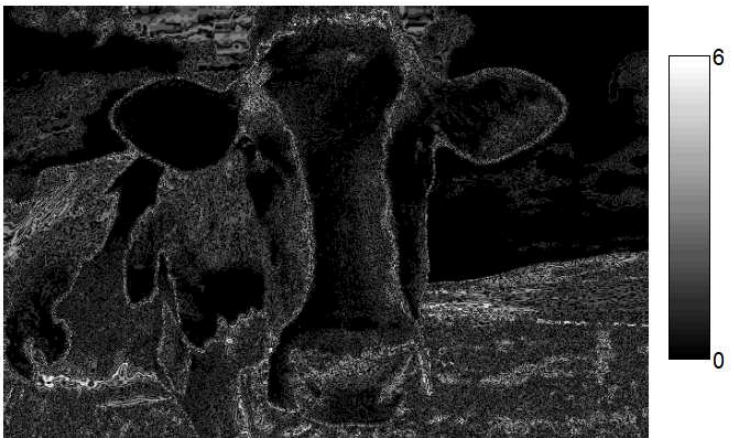

Figura 4.7: Visualização dos módulos dos erros de aproximação para cada entrada da matriz $\widehat{\mathbf{R}}$.

bits que indica o número de folhas (elementos do alfabeto) que serão consideradas na codificação de Huffman a ser transmitida, e o restante do procedimento segue como construído no capítulo 3. A tabela 4.1 apresenta o tamanho de cada componente da *string* $t^*_{\widehat{\mathbf{R}}}$, a codificação binária que representa a matriz $\widehat{\mathbf{R}}$. O mesmo procedimento desenvolvido até aqui para a matriz $\mathbf{R}$ foi aplicado para as matrizes $\mathbf{G}$ e $\mathbf{B}$, gerando, respectivamente, as aproximações $\widehat{\mathbf{G}}$ (com alfabeto efetivo de 68 intensidades) e $\widehat{\mathbf{B}}$ (com alfabeto efetivo de 59 intensidades) e as representações binárias $t^*_{\widehat{\mathbf{G}}}$ e $t^*_{\widehat{\mathbf{B}}}$, conforme apresentadas nas tabelas 4.2 e 4.3.

Nota 15. *Estamos cientes de que o bloco "Definição do modelo e dimensões da imagem" poderia estar presente apenas na string binária* $t^*_{\widehat{\mathbf{R}}}$ *e seu valor ser reutilizado para as demais strings. No entanto, como a economia é muito pequena, decidimos repeti-lo por*

	Fonte	*Bits*
Cabeçalho (439.739 *bits*)	Alfabeto utilizado	257
	Definição do modelo e dimensões da imagem	56
	Partição estimada	427.307
	Codificações condicionais	11.548
	Codificações marginais	571
	Matriz codificada	825.749
	Total	1.265.488

Tabela 4.1: Elementos da *string* binária representante da matriz $\widehat{\mathbf{R}}$ e seus respectivos números de *bits*.

	Fonte	*Bits*
Cabeçalho (332.378 *bits*)	Alfabeto utilizado	257
	Definição do modelo e dimensões da imagem	56
	Partição estimada	319.935
	Codificações condicionais	11.611
	Codificações marginais	519
	Matriz codificada	683.417
	Total	1.015.795

Tabela 4.2: Elementos da *string* binária representante da matriz $\widehat{\mathbf{G}}$ e seus respectivos números de *bits*.

	Fonte	*Bits*
Cabeçalho (219.531 *bits*)	Alfabeto utilizado	257
	Definição do modelo e dimensões da imagem	56
	Partição estimada	209.906
	Codificações condicionais	8.861
	Codificações marginais	451
	Matriz codificada	714.725
	Total	934.256

Tabela 4.3: Elementos da *string* binária representante da matriz $\widehat{\mathbf{B}}$ e seus respectivos números de *bits*.

conveniência de notação e para possíveis comparações.

Por fim, a figura 4.8 apresenta a visualização da imagem "Vaca" após os procedimentos de compressão e descompressão das matrizes $\mathbf{R}$, $\mathbf{G}$ e $\mathbf{B}$. Lembremos que a compressão foi realizada com perda de informação e que a imagem apresentada na figura 4.8 é apenas uma aproximação de "Vaca". Em particular, a aproximação é composta somente por 11.023 cores (apenas 12% da quantidade total de cores da imagem original).

Exemplo 36. *Suponha que desejamos transmitir a imagem "Vaca" e que realizamos o processo de quantização desenvolvido neste capítulo, obtendo as mesmas matrizes* $\widehat{\mathbf{R}}$, $\widehat{\mathbf{G}}$ *e* $\widehat{\mathbf{B}}$,

Figura 4.8: Visualização da imagem "Vaca" após os procedimentos de compressão e descompressão.

ou seja a mesma aproximação da imagem "Vaca" será transmitida. No entanto, considere que, para cada matriz, utilizaremos um único código de Huffman, ou seja, não incluiremos o elemento condicionante na modelagem dos processos T_i, $i \in \{\widehat{\mathbf{R}}, \widehat{\mathbf{G}}, \widehat{\mathbf{B}}\}$. *Utilizando essa abordagem, obtemos as strings binárias* $t^*_{\widehat{\mathbf{R}}}$, $t^*_{\widehat{\mathbf{G}}}$ *e* $t^*_{\widehat{\mathbf{B}}}$ *que representam, respectivamente, as matrizes* $\widehat{\mathbf{R}}$, $\widehat{\mathbf{G}}$ *e* $\widehat{\mathbf{B}}$, *cujos detalhes são informados nas tabelas 4.4, 4.5 e 4.6.*

	Fonte	*Bits*
Cabeçalho (987 *bits*)	Alfabeto utilizado	257
	Definição do modelo e dimensões da imagem	56
	Partição estimada	0
	Codificações condicionais	0
	Codificações marginais	674
	Matriz codificada	1.704.870
	Total	1.705.857

Tabela 4.4: Elementos da *string* binária representante da matriz $\widehat{\mathbf{R}}$, para uma codificação de Huffman única, e seus respectivos números de *bits*.

	Fonte	*Bits*
Cabeçalho (924 *bits*)	Alfabeto utilizado	257
	Definição do modelo e dimensões da imagem	56
	Partição estimada	0
	Codificações condicionais	0
	Codificações marginais	611
	Matriz codificada	1.662.474
	Total	1.663.398

Tabela 4.5: Elementos da *string* binária representante da matriz $\widehat{\mathbf{G}}$, para uma codificação de Huffman única, e seus respectivos números de *bits*.

	Fonte	*Bits*
Cabeçalho (784 *bits*)	Alfabeto utilizado	257
	Definição do modelo e dimensões da imagem	56
	Partição estimada	0
	Codificações condicionais	0
	Codificações marginais	471
	Matriz codificada	1.601.249
	Total	1.602.033

Tabela 4.6: Elementos da *string* binária representante da matriz $\widehat{\mathbf{B}}$, para uma codificação de Huffman única, e seus respectivos números de *bits*.

Perceba, pelas tabelas 4.1 a 4.3 e pelo exemplo 36, que a inclusão de uma codificação condicional baseada no modelo G3M reduziu consideravelmente os tamanhos totais das *strings* binárias. Em particular, essa redução é devida à notável queda nos comprimentos médios dos códigos das matrizes a serem transmitidas. No entanto, no caso do modelo mais detalhado, note que o cabeçalho de cada *string* binária ocupa basicamente um terço do tamanho da mensagem total. Além disso, quase a totalidade do cabeçalho é devida à transmissão da partição estimada do espaço de estados. Esse é um dos efeitos que devem ser balanceados na abordagem construída até aqui: o detalhamento dos parâmetros utilizados na modelagem podem não compensar a redução no comprimento médio do código da matriz a ser transmitida. Dessa forma, desejamos um equilíbrio entre tamanho de cabeçalho e o tamanho da matriz codificada. A subseção a seguir trata de abordagens convenientes para alguns problemas com características específicas.

4.2.1 Ferramentas Auxiliares para Compressão de Matrizes

Neste tópico, tratamos, brevemente, de duas abordagens que podem ser convenientemente adotadas em problemas específicos. A primeira delas trata do estudo do equilíbrio entre o tamanho do cabeçalho e o tamanho da matriz codificada. Como visto anteriormente, pelas tabelas 4.1, 4.2 e 4.3, o detalhamento da partição estimada do espaço de estados pode ser uma tarefa muito custosa, uma vez que, para o modelo considerado, temos um espaço de estados com $|\mathcal{S}^*| = |A|^3$ elementos. Se nos fixarmos novamente na matriz $\widehat{\mathbf{R}}$ da imagem "Vaca", temos que $|A^e_{\mathbf{R}}| = 75$, e, portanto, $|\mathcal{S}^*| = 75^3 = 421.875$. Dessa forma, quanto maior o alfabeto considerado, mais custosa se torna a transmissão da partição estimada, podendo chegar ao ponto em que sua manipulação seja inviável por limitações computacionais. Para ilustração, nesta subseção, utilizamos novamente a

imagem “Vaca”.

Imaginemos um processo de transmissão de imagens em duas etapas. Na primeira, é transmitida uma prévia da imagem, em que apenas grandes contrastes são comunicados e, na segunda, um detalhamento das cores ocorre. Com essa motivação, consideremos que o alfabeto $A^e_{\mathbf{R}}$ pode ser particionado como

$$A^e_{\mathbf{R}} = A_1 \cup A_2 \cup \ldots \cup A_k, \tag{4.4}$$

em que $A_i = \{a_{i,1}, a_{i,2}, \ldots a_{i,m_i}\}$ é um subconjunto de elementos sequenciais de $A^e_{\mathbf{R}}$ (lembre que $A^e_{\mathbf{R}}$ é um alfabeto ordenado), com $i \in \{1, \ldots, k\}$, para qualquer $k \geq 1$. Perceba que podemos adequar a matriz $\widehat{\mathbf{R}}$, e, consequentemente, a *string* $t_{\widehat{\mathbf{R}}}$, a essa partição e obter uma nova matriz (e *string*) composta por k diferentes valores arbitrários. O conjunto desses k valores será denominado por alfabeto base, $A^{base}_{\widehat{\mathbf{R}}} = \{b_1, \ldots, b_k\}$, e a nova matriz por matriz base, $\widehat{\mathbf{R}}^{base}$. Também definimos $t_{\widehat{\mathbf{R}}^{base}}$ a H-leitura da nova matriz $\widehat{\mathbf{R}}^{base}$. Repare que, com isso, podemos estabelecer um modelo G3M sobre a *string* $t_{\widehat{\mathbf{R}}^{base}}$ e codificá-la de forma análoga à realizada anteriormente neste capítulo. Notemos que, se representantes convenientes forem selecionados em cada grupo (ou seja, cada um dos $b_i \in A^{base}_{\widehat{\mathbf{R}}}$, $i \in \{1, \ldots, k\}$, for bem escolhido), a matriz $\widehat{\mathbf{R}}^{base}$ terá a função de prévia desejada, em que apenas grandes contrastes de intensidades serão computados. Essa escolha de representantes pode ser feita de diversas formas, como a seleção via elementos centrais dos conjuntos A_i, $i \in \{1, \ldots, k\}$, ou via aproximações de seus valores médios, ou via seus extremos etc.

Em seguida, desejamos transmitir o detalhamento de intensidades presentes na imagem. Como discutido diversas vezes neste capítulo, os elementos da matriz de intensidades $\widehat{\mathbf{R}}$ têm suas diferentes vizinhanças compostas por poucos elementos do alfabeto $A^e_{\mathbf{R}}$. Mostraremos como podemos nos aproveitar desta característica, baseando-nos na construção de k modelos G3M diferentes. Para o desenvolvimento do procedimento, utilizaremos concomitantemente as *strings* $t_{\widehat{\mathbf{R}}}$ e $t_{\widehat{\mathbf{R}}^{base}}$. Iniciamos a construção do primeiro modelo definindo o conjunto $A^{(1)}_s = A_1 \cup A^{base}_{\widehat{\mathbf{R}}} \setminus b_1$, sendo $A_1 = \{a_{1,1}, a_{1,2}, \ldots a_{1,m_1}\}$ o primeiro elemento da partição delineada na equação (4.4). Dessa forma, $A^{(1)}_s = \{a_{1,1}, a_{1,2}, \ldots a_{1,m_1}, b_2, b_3, \ldots, b_k\}$ é o que será denominado subalfabeto para o primeiro modelo. A partir desse subalfabeto, podemos construir uma nova *string*,

$t^{(1)}_{\widehat{\mathbf{R}}}$, baseada em $t_{\widehat{\mathbf{R}}}$ e $t_{\widehat{\mathbf{R}}^{base}}$ da seguinte maneira, para $j \in \{1, \ldots, |\widehat{\mathbf{R}}|\}$: se a j-ésima coordenada de $t_{\widehat{\mathbf{R}}}$ pertence a A_1, então a j-ésima coordenada de $t^{(1)}_{\widehat{\mathbf{R}}}$ recebe esse valor; caso contrário, recebe o valor de $t_{\widehat{\mathbf{R}}^{base}}$. Com isso, $t^{(1)}_{\widehat{\mathbf{R}}}$ é uma *string* composta por elementos de $A^{(1)}_s$. Assim sendo, podemos construir um modelo G3M para essa amostra e elaborar códigos de Huffman condicionais de forma análoga à já realizada diversas vezes neste texto. Procedendo dessa maneira, podemos construir os subalfabetos, modelos e códigos restantes.

Após obter os k modelos e k códigos condicionais pelo método descrito anteriormente, podemos iniciar a codificação da matriz $\widehat{\mathbf{R}}$ como um todo, tendo o auxílio da matriz $\widehat{\mathbf{R}}^{base}$. Considere que, para cada elemento do alfabeto base $A^{base}_{\widehat{\mathbf{R}}}$, temos um subalfabeto associado. Por exemplo, para o elemento $b_2 \in A^{base}_{\widehat{\mathbf{R}}}$, temos associado o subalfabeto $A^{(2)}_s = \{b_1, a_{2,1}, a_{2,2}, \ldots a_{2,m_2}, b_3, b_4, \ldots, b_k\}$. Assim sendo, para cada coordenada da matriz $\widehat{\mathbf{R}}$ em que o formato de vizinhança é aplicável, construíremos vizinhanças baseadas nos diferentes subalfabetos. Por exemplo, suponha que, em determinada coordenada da matriz $\widehat{\mathbf{R}}^{base}$, observou-se o valor b_3 e a vizinhança $b_3b_2b_3$. E nessa mesma coordenada, na matriz $\widehat{\mathbf{R}}$, observou-se o valor $a_{3,2}$ e a vizinhança $a_{3,1}a_{2,4}a_{3,2}$. O subalfabeto associado ao valor observado b_3 é $A^{(3)}_s$ e a vizinhança na matriz $\widehat{\mathbf{R}}$ adaptada a esse subalfabeto é a *string* $a_{3,1}b_2a_{3,2}$. Perceba que essa *string* é um elemento do espaço de estados do terceiro modelo G3M e pertence a uma parte L dentro da partição estimada. Dessa forma, somos capazes de codificar a observação de valor $a_{3,2}$ como $C_3(a_{3,2}|L)$, em que C_3 indica que o sistema de codificação utilizado é proveniente do terceiro subalfabeto.

Agora, a pergunta natural é: a abordagem de subalfabetos apresentada, além de uma pré-imagem, é capaz de trazer algum ganho real para o processo de compressão? A resposta é sim, quando o alfabeto $A^{e}_{\mathbf{R}}$ é muito grande. Perceba que o modelo G3M sobre o alfabeto base $A^{base}_{\widehat{\mathbf{R}}}$ possui um espaço de estados associado com k^3 elementos. Já, cada um dos k modelos G3M possui um espaço de estados associado com $|A^{(j)}_s|^3 = (k - 1 + |A_j|)^3$, A_j como apresentado em (4.4), $j \in \{1, \ldots, k\}$. Assim sendo, o processo de definição do modelo inclui a identificação de

$$k^3 + \sum_{j=1}^{k} |A^{(j)}_s|^3 = k^3 + \sum_{j=1}^{k} (k - 1 + |A_j|)^3 \tag{4.5}$$

elementos de diferentes espaços de estados. Para o caso da imagem “Vaca”, tomemos

$k = \left\lceil \sqrt{|A^e_{\mathbf{R}}|} \right\rceil = \left\lceil \sqrt{75} \right\rceil = 9$ na tentativa de cobrir o alfabeto $A^e_{\mathbf{R}}$ com subalfabetos de tamanhos os mais próximos possíveis. Ainda com essa motivação, consideremos a partição de elementos sequenciais do alfabeto de intensidades $A^e_{\mathbf{R}} = A_1 \cup A_2 \cup \ldots \cup A_9$, tais que $|A_j| = 9$, se $1 \leq j \leq 3$, e $|A_j| = 8$, se $4 \leq j \leq 9$. Substituindo essas quantidades na expressão apresentada em (4.5), temos que será necessária a comunicação de 40.044 elementos para se definir as estruturas das partições. Repare que este número é muito inferior a $75^3 = 421.875$, número de elementos do espaço de estados $\mathcal{S}^* = {A^e_{\mathbf{R}}}^3$. Aplicando todo o processo de compactação por subalfabetos aqui discutido nas matrizes de intensidades **R**, **G** e **B** da imagem "Vaca", obtemos os comprimentos de *strings* apresentados, respectivamente, nas tabelas 4.7, 4.8 e 4.9. O algoritmo A.6 sistematiza a obtenção das *strings* binárias associadas à codificação de matrizes pelo método de subalfabetos.

	Fonte	*Bits*
	Definições da imagem e do modelo	313
Base	Definição da partição e codificações	2.483
	Matriz codificada	301.951
Subalfabetos	Definição das partições e codificações	67.828
	Matriz codificada	582.654
	Total	955.229

Tabela 4.7: Elementos da *string* binária representante da matriz $\widehat{\mathbf{R}}$, para uma abordagem de subalfabetos, e seus respectivos números de *bits*.

	Fonte	*Bits*
	Definições da imagem e do modelo	313
Base	Definição da partição e codificações	2.296
	Matriz codificada	300.572
Subalfabetos	Definição das partições e codificações	60.896
	Matriz codificada	517.153
	Total	881.230

Tabela 4.8: Elementos da *string* binária representante da matriz $\widehat{\mathbf{G}}$, para uma abordagem de subalfabetos, e seus respectivos números de *bits*.

Algumas comparações interessantes podem ser realizadas entre as tabelas 4.1 a 4.3 e 4.7 a 4.9. A primeira delas é a redução notável no comprimento total da *string* alcançada pela abordagem de subalfabetos. Notemos que essa economia é devida, exclusivamente, pela possibilidade de elaborarmos um modelo com descrição mais compacta, ou seja, que pode ser expresso em um menor número de quantidades. No entanto, para

	Fonte	*Bits*
	Definições da imagem e do modelo	313
Base	Definição da partição e codificações	1.961
	Matriz codificada	300.302
Subalfabetos	Definição das partições e codificações	44.676
	Matriz codificada	526.994
	Total	874.246

Tabela 4.9: Elementos da *string* binária representante da matriz $\widehat{\mathbf{B}}$, para uma abordagem de subalfabetos, e seus respectivos números de *bits*.

que essa economia seja alcançada, foi necessário o sacrifício de um pouco do comprimento médio do código necessário para descrever cada entrada da matriz a ser codificada, uma vez que cada elemento é codificado duas vezes (uma pelo alfabeto base e outra pelo subalfabeto). Outro fato importante é que a taxa de redução observada está intimamente ligada ao tamanho do alfabeto efetivo utilizado na matriz. Por exemplo, as matrizes $\widehat{\mathbf{R}}$, $\widehat{\mathbf{G}}$ e $\widehat{\mathbf{B}}$ possuem alfabeto de 75, 68 e 59 e proporcionaram economia de 25%, 13% e 7%, respectivamente. Isso se deve, novamente, ao fato de que a redução proposta pela abordagem dos subalfabetos visa construir uma estrutura mais econômica para se descrever as classes de condicionamento, o que é um ponto fraco para a abordagem tradicional para alfabetos grandes, pois quanto maior o alfabeto, maior o espaço de estados que precisa ser detalhado ($\mathcal{S}^* = A^3$). Assim sendo, o método dos subalfabetos se propõe a confrontar exatamente essa limitação. Por isso, esta abordagem é recomendada para codificação de matrizes com grandes alfabetos.

Uma última abordagem, que será apenas comentada neste texto, tem sua aplicabilidade na transmissão de matrizes de intensidades, em que a percepção de contrastes é mais importante que transições suaves dos estímulos visuais. Dessa forma, é uma alteração no processo de quantização proposto anteriormente. Um exemplo de uso ocorre em estudos sobre desmatamento via imagens de satélite, em que transições entre tons de verde podem não ser tão importantes quanto distinguir a cor verde da marrom, em suas diferentes tonalidades. Sendo assim, para cada matriz de intensidades, podemos particionar o respectivo alfabeto de maneira análoga à realizada em (4.4), ou seja, segregar o alfabeto em subconjuntos de elementos sequenciais como $A = A_1 \cup A_2 \cup \ldots \cup A_n$, em que $|A_i| = \varepsilon_i$, para $i \in \{1, \ldots, n\}$, e associar um representante para cada conjunto A_i. Perceba que esta abordagem equivale ao uso apenas da matriz base utilizada no método de subalfabetos,

Figura 4.9: Visualização da aproximação da imagem “Vaca” utilizando, para cada matriz de intensidades, alfabeto com sete representantes.

simplesmente flexibilizando o tamanho de cada subconjunto (ε_i) conforme a necessidade. Um exemplo de utilização desta abordagem para imagem “Vaca” é apresentado na figura 4.9. Nele, para cada matriz de intensidades, foram considerados 7 grupos, com $\varepsilon_i = 23$, para $i \in \{1, 7\}$, e $\varepsilon_i = 42$, para $i \in \{2, \ldots, 6\}$. Para o grupo A_1 foi escolhido o representante 0 (intensidade mínima), para o grupo A_7 foi escolhido o representante 255 (intensidade máxima), e para os demais grupos foi tomado o valor inteiro da intensidade mediana.

4.3 Formatos Tradicionais para Imagens Digitais

Nesta seção, comentaremos superficialmente sobre os formatos de arquivos mais utilizados para a transmissão e armazenamento de imagens sem e com perda. O formato PNG (*Portable Network Graphics*) é comumente utilizado para compressão de imagens sem perda de informação. Assim como trabalhado neste capítulo, as imagens PNG são definidas por um modelo RGB, ou RGBA, em que A representa uma quarta matriz de intensidades $\mathbf{A}$, denominada *Alpha*, que define o grau de transparência de um *pixel*. O formato PNG possui uma grande capacidade de personalização, que permite uma boa adequação às necessidades do desenvolvedor. Características da imagem e preferências de usuário são organizadas em trechos denominados *chunks*. Entre as preferências, há a possibilidade de exigir, ou não, a construção de pré-imagens (em sete visualizações progressivas) durante a descompactação do arquivo, que aumenta levemente seu tamanho total, mas é muito conveniente para usuários que necessitam de uma rápida exibição.

Apesar de todas as características mencionadas, a que desejamos salientar é o método de compressão utilizado para codificar as matrizes de intensidades. Geralmente, um filtro é aplicado para transformar as sequências de *bytes* (conjuntos de 8 *bits*) obtidos na leitura de cada linha da imagem. Esse processo não causa nenhuma redução no número de *bits* a serem codificados, mas os organiza de forma a otimizar a compressão a ser realizada. A compactação é realizada utilizando um método sem perdas, que emprega uma combinação dos códigos de Lempel-Ziv (veja [32]) e Huffman. Para mais detalhes sobre o formado PNG, veja [21] e [27].

O formato JPEG (*Joint Photographic Experts Group*), também conhecido por JPG, é comumente utilizado para compressão de imagens com perda de qualidade (o formato também permite uma personalização para compressão sem perda, mas não nos atemos a ela). Tendo suas origens no início dos anos 1990, a eficiência na compactação de imagens digitais apresentada por este formato permitiu a expansão do uso de figuras na *Internet*. Devido às manipulações utilizadas durante o processo de compressão, em particular a transformada discreta de cosseno (DCT, do inglês *Discrete Cosine Transform*, veja [1]), o formato JPEG tem melhor desempenho em imagens em que as transições de cores ocorrem de maneira suave. No entanto, sua taxa de compressão pode ser ajustada para promover um maior equilíbrio entre a qualidade da imagem e o tamanho total do arquivo. Para o processo de compactação, a imagem é dividida em blocos e o único passo em que há perda de informação ocorre durante o procedimento de quantização, em que arredondamentos numéricos são empregados. Após as transformações convenientes e a quantização, são utilizados códigos aritméticos (veja [24]) ou códigos de Huffman para a compressão dos dados. Para mais detalhes sobre o formato JPEG, veja [20].

Apresentamos, nesta seção, os formatos PNG e JPEG apenas para exemplificar como o campo de transmissão/armazenamento de imagens é tratado nos dias atuais. O intuito deste texto não é estabelecer uma competição entre os métodos desenvolvidos neste capítulo e outros já bem estabelecidos. Queremos apenas ilustrar que há técnicas utilizadas em outros formatos que podem ser aplicadas e, possivelmente, trazer benefícios para os procedimentos aqui estudados. Por exemplo, para futuros estudos, pode ser interessante a utilização de transformadas na matriz de intensidades e o desenvolvimento de outras codificações que levem em consideração a relação que uma observação possui com seus vizinhos.

4.4 Conclusão

Neste capítulo, apresentamos como o problema de codificação de imagens pode ser interpretado como uma questão de codificação de matrizes. Na seção 4.2, sugerimos um método de quantização (aproximação) que tem como base elementos da teoria de informação expostos no capítulo 2. Evidenciamos, pelo exemplo 36, que a inclusão de um modelo G3M na construção das classes de condicionamento do processo tem potencial para gerar códigos com tamanhos médios menores. No entanto, vimos que um modelo muito detalhado pode sofrer de superparametrização e causar um aumento desnecessário no comprimento da *string* a ser transmitida. Para amenizar esse problema, técnicas adequadas podem ser empregadas, como a de subalfabetos explorada na subseção 4.2.1, que tem como objetivo a simplificação do modelo a ser trabalhado. Por fim, declaramos que o procedimento de codificação de imagens apresentado neste capítulo é apenas um exercício de aplicação que tem o intuito de ilustrar a utilidade, flexibilidade e diversas características dos modelos markovianos de partição e sua aliança com codificações instantâneas, em particular, com códigos de Huffman.

Capítulo 5

Considerações Finais

Este trabalho possui um objetivo claro e direto: estudar a implementação das estruturas propostas pelos modelos markovianos de partição no escopo de transmissão/armazenamento de dados. Para alcançar esta meta, alguns desenvolvimentos preparatórios foram necessários.

No capítulo 1, apresentamos os fundamentos essenciais sobre processos estocásticos, em particular cadeias de Markov, e como podemos simplificar as estruturas de dependência delineadas por um processo utilizando um manejo conveniente do espaço de estados associado. Argumentamos que, apesar de sua praticidade, as cadeias de Markov possuem restrições em sua aplicação devido a questões inferenciais, pois, para modelos complexos, requerem amostras muito grandes para estimar todos os parâmetros envolvidos. Buscando simplificação do espaço de estados associado ao processo, reintroduzimos o conceito de modelos markovianos de partição, que, devido a sua flexibilidade, possui potencial de, fixado um valor para a memória, reduzir a quantidade de parâmetros (probabilidades de transição) necessários para se definir o modelo. Essa redução de complexidade faz com que tenhamos mais observações por parâmetro livre, provocando uma melhoria da estimação das quantidades. Ainda visando uma representação econômica, introduzimos uma nova família de processos markovianos: a cadeia de Markov com interstício. Essa classe de modelos permite a existência de um intervalo entre as observações mais relevantes do passado. Tal consideração permitiu que uma memória maior do processo fosse possível de ser apreciada sem que a qualidade da estimação dos parâmetros fosse comprometida. Evidenciamos que essa abordagem traz benefícios para a modelagem de sequenciamentos

genéticos, como o da Covid-19 apresentado na subseção 1.3.1, e possui potencial para modelagem de estruturas vicinais, conforme exemplificado na subseção 1.3.2.

Como nosso propósito é investigar as vantagens da inclusão de modelos estatísticos na codificação de dados, apresentamos, no capítulo 2, uma relação de resultados associados à teoria de informação, um campo da ciência em que um dos interesses primordiais consiste na eficiência da transmissão de mensagens. Ali, determinamos todas as diretrizes sobre os sistemas de codificação que temos interesse neste trabalho: códigos de prefixo (permitem uma decodificação instantânea, pois nenhum elemento codificado é prefixo de outro) e com representação binária com menor comprimento médio possível. Para este fim, estabelecemos um algoritmo para obtenção de uma codificação de Huffman, a qual possui ambas as propriedades desejadas e, por isso, é denominada ótima. Além disso, definimos a interpretação da mensagem de interesse como a observação de uma amostra proveniente de um processo estocástico, estabelecemos o conceito de taxa de entropia e como ele está associado aos limites do comprimento médio de um código ótimo. Também, apresentamos resultados sobre entropia condicional e como o condicionamento de uma variável aleatória em um vetor aleatório pode reduzir a incerteza associada, possibilitando a construção de codificações condicionais mais eficientes (menores comprimentos médios).

Apesar do capítulo 2 estabelecer uma ligação inicial entre processos estocásticos e sistemas de codificação ótimos, é no capítulo 3 que introduzimos todos os passos suficientes para a transmissão de *strings* e matrizes utilizando cadeias de Markov. Nele, além de apresentarmos todos os paradigmas de escolha envolvidos na transmissão das quantidades necessárias para a decodificação da mensagem, evidenciamos a maior dificuldade do uso de cadeias de Markov para compressão de dados: o grande volume de probabilidades condicionais a serem transmitidas. Este é o ponto chave deste trabalho, pois os modelos de partição aparecem como uma opção para a redução do número dessas probabilidades (veja também [15]). Esta alternativa, apesar de necessitar da inclusão da descrição da partição durante o processo de transmissão, é mais econômica em muitas situações, conforme evidenciado na subseção 3.1.1, e assintoticamente equivalente à abordagem tradicional. Também, no final desse capítulo, apresentamos como a compressão de matrizes via modelos de partição pode ser vista como uma simples adaptação do procedimento de codificação de *strings* utilizando uma leitura conveniente da matriz e as cadeias de Markov com interstício.

O capítulo 4 é um exercício de aplicação abrangente, onde colocamos em prática todo o desenvolvimento dos capítulos anteriores. Nele, elucidamos o significado de transmissão com perda e como ele se aplica ao caso de comunicação de imagens. Para que o procedimento pudesse ser executado, apresentamos um método de quantização que utiliza elementos fundamentados na teoria de informação introduzidos no capítulo 2 e nos modelos markovianos de partição desenvolvidos no capítulo 1. Com a aproximação resultante da quantização, aplicamos os procedimentos de codificação estudados no capítulo 3 e evidenciamos que a inclusão dos modelos markovianos de partição podem trazer benefícios para a compressão de imagens. No entanto, como exemplificado na subseção 4.2.1, há diversas minúcias (como o alto custo para se comunicar as partições estimadas) que precisam de atenção para tornar o procedimento eficiente e aplicável para compactação prática.

O presente trabalho não tem o intuito de ser concludente para o tema "compressibilidade", mas de possibilitar novas linhas alternativas de estudo. Um primeiro tópico é o desdobramento de aplicações das cadeias de Markov com interstício, além das apresentadas neste texto. Por exemplo, [11] utiliza a flexibilidade de tal modelagem para adaptar os resultados de [10] sobre comparação de processos estocásticos e permitir uma análise de conglomerados eficiente sobre sequenciamentos genéticos de Covid-19 (para mais exemplos sobre comparações de processos estocásticos utilizando os princípios de modelos de partição, confira [12], [13] e [14]). Um segundo ponto de estudo é referente aos resultados desenvolvidos no capítulo 3. Ali, determinamos que toda mensagem codificada é precedida por um cabeçalho que comunica ao decodificador todas as informações necessárias para a compreensão do sistema de codificação utilizado. Dessa forma, buscamos maneiras mais econômicas para a transmissão das quantidades envolvidas, em particular, a comunicação da partição estimada pelo modelo markoviano mínimo e dos códigos de Huffman operados. Atualmente, estudamos a transmissão da partição de interesse empregando uma estrutura de árvore $|A|$-ária (cada nó interno possui $|A|$ filhos), em que A representa o alfabeto associado à mensagem a ser codificada. Tal estratégia tem se mostrado eficiente para estruturas específicas do espaço de estados. Um terceiro tópico estende-se sobre a escolha da vizinhança delineada na subseção 1.3.2 utilizada como classe de condicionamento para a construção dos códigos de Huffman condicionais. Por simplicidade, em nenhum momento questionamos se o formato considerado promoveria a maior compressão da matriz a ser

codificada. Assim, é interessante o desenvolvimento de procedimentos que permitam acessar qual formato de vizinhança, dentro de um conjunto preestabelecido, é o mais propenso a prover uma representação mais compacta. Uma ideia inicial é a escolha da vizinhança cujo modelo que a use como classe de condicionamento maximize o valor do BIC associado, pois, conforme evidenciado na subseção 3.1.1, há uma relação entre taxa de compressão e tal critério de informação. Um quarto ponto é referente à quantização empregada no capítulo 4. Como ela não era nosso principal interesse, novamente por conveniência, não exploramos suas propriedades. Dessa forma, é interessante o estudo detalhado sobre os erros devidos às aproximações das matrizes que compõem a imagem. Um último tópico, já manifestado na seção 4.3, é referente à agregação dos códigos de Huffman via modelos de partição em procedimentos de processamento de imagens já bem estabelecidos, como os utilizados pelos formatos PNG e JPEG.

Por fim, concluímos sobre o potencial da relação entre modelos markovianos de partição e compressão de dados uni e bidimensionais, anuindo que ainda há um amplo horizonte a ser vislumbrado sobre esta aliança.

Referências Bibliográficas

[1] Ahmed. N., & Natarajan, T., & Rao, K. (1974). Discrete cosine transform. *IEEE Transactions on Computers*, *100* (1), pp. 90–93.

[2] Bühlmann, P., & Wyner, A. J. (1999). Variable length Markov chains. *The Annals of Statistics*, *27* (2), pp. 480-513.

[3] Cover, T. M., & Thomas, J. A. (2006). *Elements of information theory*. John Wiley & Sons.

[4] Csiszar, I., & Shields, P. C. (2000). The consistency of the BIC Markov order estimator. *The Annals of Statistics*, *28* (6), pp. 1601–1619. (http://www.jstor.org/stable/2673999)

[5] Csiszar, I., & Talata, Z. (2006). Context tree estimation for not necessarily finite memory processes, via BIC and MDL. *IEEE Transactions on Information Theory*, *52* (3), pp. 1007-1016.

[6] Finesso, L., & Kimura, H., & Kodama, S. (1992). Estimation of the order of a finite Markov chain. Em *Recent Advances in Mathematical Theory of Systems, Control, Networks and Signal Processing*, pp. 643-645.

[7] Fischer, S. R. (2003). *History of writing*. Reaktion books.

[8] García, J. E., & González-López, V. A. (2011). Minimal Markov models. Em *Fourth Workshop on Information Theoretic Methods in Science and Engineering*, p. 25.

[9] García, J. E., & González-López, V. A. (2017). Consistent estimation of partition Markov models. *Entropy*, *19* (4), 160. (https://doi.org/10.3390/e19040160)

[10] García, J. E., & Gholizadeh, R., & González-López, V. A. (2018). A BIC-based consistent metric between Markovian processes. *Applied Stochastic Models in Business and Industry*, *34* (6), pp. 868-878. (https://doi.org/10.1002/asmb.2346)

[11] García, J. E., & González-López, V. A., & Tasca, G. H. (2020). Partition Markov model for Covid-19 virus. *4open*, *3*, 13.

[12] García, J. E., & González-López, V. A., & Tasca, G. H. A stochastic inspection about genetic variants of Covid-19 circulating in Brazil during 2020. *AIP Conference Proceedings*. (forthcoming)

[13] García, J. E., & González-López, V. A., & Tasca, G. H. Comparison between genomic sequences of SARS-CoV 2 virus: original versus P.1 variant. *AIP Conference Proceedings*. (forthcoming)

[14] García, J. E., & González-López, V. A., & Tasca, G. H. Stochastic comparison between the original SARS-CoV 2 genetic Structure and SARS-CoV 2 - P.1 variant. Em *Quantitative Methods in Demography: Methods and Related Applications in the Covid-19 Era*. Springer, Cham. (Editores: Christos H. Skiadas, & Charilaos Skiadas - The Springer Series on Demographic Methods and Population Analysis) (forthcoming)

[15] García, J. E., & González-López, V. A., & Tasca, G. H. Huffman coding and minimal Markov models to improve data compression. *AIP Conference Proceedings*. (forthcoming)

[16] Gikhman, I. I., & Skorokhod, A. V. (1969). *Introduction to the theory of random processes*. WB Saunders Co., p. 297.

[17] Goffman-Vinopal, L., & Porat, M. (2002). Color image compression using inter-color correlation. *Proceedings. International Conference on Image Processing* (Vol. 2).

[18] Huffman, D. A. (1952). A Method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, *40* (9), pp. 1098-1101.

[19] ISO/IEC 2382-1:1993 – Information technology — Vocabulary — Part 1: Fundamental terms.

[20] ISO/IEC 10918:1994 – Information technology — Digital compression and coding of continuous-tone still images: Requirements and guidelines.

[21] ISO/IEC 15948:2004 – Information technology – Computer graphics and image processing – Portable Network Graphics (PNG): Functional specification.

[22] Kraft, L. G. (1949). A device for quantizing, grouping, and coding amplitude-modulated pulses. Tese (doutorado) - Massachusetts Institute of Technology.

[23] Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *11* (7), pp. 674-693.

[24] Rissanen, J. (1976). Generalized Kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, *20* (3), pp. 198–203.

[25] Rissanen, J., & Langdon, G. G. (1981). Universal modeling and coding. *IEEE Transactions on Information Theory*, *27* (1), pp. 12–23.

[26] Rissanen, J. (1983). A universal data compression system. *IEEE Transactions on Information Theory*, *29* (5), pp. 656-664.

[27] Roelofs, G. (1999). *PNG: The Definitive Guide*. O'Reilly & Associates.

[28] Shannon, C. E. (1948). A Mathematical theory of communication. *The Bell System Technical Journal*, *27* (3), pp. 379-423.

[29] Wu, F., & Zhao, S., & Yu, B., & Chen, Y. M., & Wang, W., & Song, Z. G., & Hu, Y., & Tao, Z. W., & Tian, J. H., & Pei, Y. Y., & Yuan, M. L., & Zhang, Y. L., & Dai, F. H., & Liu, Y., & Wang, Q. M., & Zheng, J. J., & Xu, L., & Holmes, E. C., & Zhang, Y. Z. (2020). A new coronavirus associated with human respiratory disease in China. *Nature*, *579*, pp. 265-269. (https://doi.org/10.1038/s41586-020-2008-3).

[30] Yaginuma, K. Y. (2010). Compressão de dados baseada nos modelos de Markov mínimos. Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Matematica, Estatistica e Computação Cientifica, Campinas, SP. (http://www.repositorio.unicamp.br/handle/REPOSIP/307243)

[31] Young, T. (1802). II. The Bakerian Lecture. On the theory of light and colours. *Philosophical transactions of the Royal Society of London*, *92*, pp. 12–48.

[32] Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, *23* (3), pp. 337–343.

APÊNDICE A

Algoritmos

Algoritmo A.1 : Obtenção da partição boa mínima a partir de partições boas.

Seja x_1^n uma amostra de uma cadeia de Markov a tempo discreto de ordem $o < \infty$, sobre um alfabeto finito A, com espaço de estados $\mathcal{S} = A^o$. Seja $\mathcal{L} = \{L_1, \ldots, L_K\}$ uma partição boa de $\mathcal{S}$.
$i \leftarrow 0$
$j \leftarrow 1$
Enquanto $i < K - 1$ **Faça**
  $i \leftarrow i + 1$
  Enquanto $j < K$ **Faça**
    $j \leftarrow j + 1$
    $d \leftarrow d_{\mathcal{L}}(i, j)$
    Enquanto $d < 1$ **Faça**
      $L_i \leftarrow L_i \cup L_j$
      Para $l \in \{j, \ldots, K - 1\}$ **Faça**
        $L_l \leftarrow L_{l+1}$
      Finalize Para
      $K \leftarrow K - 1$
      $\mathcal{L} = \{L_1, \ldots, L_K\}$
      $d \leftarrow d_{\mathcal{L}}(i, j)$
    Finalize Enquanto
  Finalize Enquanto
Finalize Enquanto
Retorne $\widehat{\mathcal{L}}_n = \{L_1, \ldots, L_K\}$.

Algoritmo A.2 : Obtenção de um código de Huffman.

Seja X uma variável aleatória definida sobre o alfabeto $A = \{x_1, \ldots, x_m\}$, com função de probabilidade p_X. Para $i \in \{1, \ldots, m\}$, denote por $p_{x_i} = p_X(x_i) = \Pr(X = x_i)$, $C_{x_i} = C(x_i)$ e $l_{x_i} = l(x_i)$. Sejam $\mathbb{P} = \{p_{x_1}, \ldots, p_{x_m}\}$ e $G = \{x_1, \ldots, x_m\}$. Inicialize com $C_{x_i} = \emptyset$ e $l_{x_i} = 0$, para $i \in \{1, \ldots, m\}$, em que $\emptyset$ denota a *string* vazia.
Se $|\mathbb{P}| = 1$ **então**
  $C_{x_1} = 0$.
Caso contrário
  Enquanto $|\mathbb{P}| > 1$ **Faça**
    Encontre $I = \arg\min\{p_{x_i} \in \mathbb{P}\}$ e $J = \arg\min\{p_{x_j} \in \mathbb{P} \setminus \{p_{x_i} : i \in I\}\}$.
    Defina $H = I \cup J$.
    Se $|H| = 2$ e $l_h = 0$, $\forall h \in H$ **então**
      $C_I = 0$ e $C_J = 1$.
    Caso contrário
      Para $h \in I$, $C_h = 0C_h$.
      Para $h \in J$, $C_h = 1C_h$.
    Finalize Se
    Defina $I' = H$, $\mathbb{P}' = \{p_k : k \in I'\}$, $p_{I'} = \sum_{k \in I'} p_k$ e $G' = \{k : k \in I'\}$.
    Atualize $\mathbb{P} = \mathbb{P} \setminus \mathbb{P}' \cup \{p_{I'}\}$ e $G = G \setminus G' \cup \{I'\}$.
  Finalize Enquanto
Finalize Se
Retorne o conjunto $\{C_{x_1}, \ldots, C_{x_m}\}$.

Algoritmo A.3 : Codificação de matriz utilizando como formato de vizinhança uma cadeia de Markov com interstício com $M = 1$, $G = W$ e $g = 1$.

Seja $\mathbf{B}$ uma matriz cujos elementos são provenientes de um alfabeto finito A.
Seja $t = x_1 x_2 \dots x_{|\mathbf{B}|}$ a *string* proveniente da H-leitura de $\mathbf{B}$.
Seja $\widehat{\mathcal{L}} = \{\widehat{L_1}, \dots, \widehat{L_{|\widehat{\mathcal{L}}|}}\}$ a partição estimada (pelo algoritmo A.1) do espaço de estados referente a uma cadeia de Markov com interstício com $M = 1$, $G = W$ e $g = 1$, para as observações de t restritas à região **B1**.
Sejam $\mathcal{N}_{B1} = \{n_a^{(L)} : a \in A, L \in \widehat{\mathcal{L}}\}$ o conjunto das frequências condicionais para os elementos em **B1** e $\mathcal{N}_{B2} = \{n_a : a \in A\}$ o conjunto das frequências marginais para os elementos em **B2**.
Para $k \in \{1, \dots, |\widehat{\mathcal{L}}|\}$ **Faça**
  Gere uma codificação de prefixo de Huffman condicional usando as frequências condicionais $\{n_a^{(L_k)} : a \in A\}$, obtendo uma codificação condicional $C(\cdot|L_k)$.
Finalize Para
Gere uma codificação de prefixo de Huffman padrão usando as frequências marginais $\mathcal{N}_{B2}$, obtendo uma codificação $C(\cdot)$.
Para $i \in \{1, \dots, |\mathbf{B}|\}$ **Faça**
  Se $x_i \in$ **B1 então**
    Encontre $L \in \widehat{\mathcal{L}}$ tal que $s = (x_{i-G-1}\ x_{i-G}\ x_{i-1}) \in L$.
    Codifique o símbolo x_i usando $C(x_i|L)$.
  Caso contrário
    Codifique o símbolo x_i usando $C(x_i)$.
  Finalize Se
Finalize Para

Algoritmo A.4 : Seleção do conjunto de pontos centrais para quantização.

Seja t uma *string* de elementos provenientes do alfabeto ordenado A. Seja k o número de pontos centrais desejado.
Compute os quantis amostrais referentes aos elementos da *string* t: $q_0, q_{\frac{1}{k-1}}, \dots, q_{\frac{k-2}{k-1}}, q_1$.

Se $\{q_0, q_{\frac{1}{k-1}}, \dots, q_{\frac{k-2}{k-1}}, q_1\}$ possui k valores distintos **então**
  $D = \{q_0, q_{\frac{1}{k-1}}, \dots, q_{\frac{k-2}{k-1}}, q_1\}$.
Caso contrário
  $l = k$.
  Defina D como o conjunto dos valores distintos em $\{q_0, q_{\frac{1}{l-1}}, \dots, q_{\frac{l-2}{l-1}}, q_1\}$.
  Enquanto $|D| < k$ **Faça**
    $l = k + 1$.
    Compute os quantis $q_0, q_{\frac{1}{l-1}}, \dots, q_{\frac{l-2}{l-1}}, q_1$.
    Defina D como o conjunto dos valores distintos em $\{q_0, q_{\frac{1}{l-1}}, \dots, q_{\frac{l-2}{l-1}}, q_1\}$.
  Finalize Enquanto
Finalize Se
Retorne D.

Algoritmo A.5 : Quantização de uma matriz de intensidades.

Sejam $\mathbf{B}$ uma matriz de intensidades de dimensão $K \times W$, com entradas provenientes do alfabeto $A_{\mathbf{B}}$, e $t_{\mathbf{B}}$ sua H-leitura.
Seja $\mathcal{S}^*$ o espaço de estados associado a uma cadeia de Markov com interstício definida sobre o alfabeto $A_{\mathbf{B}}$, com parâmetros $g = 1$, $G = W$ e $M = 1$.
Para a *string* $t_{\mathbf{B}}$, compute $\delta = \max_{s \in \mathcal{S}^*} |\{a \in A_{\mathbf{B}} : N_n(sa) > 0\}|$ e $\alpha^{(\delta)} = \delta \times 20$.
Defina, sob as mesmas condições da definição 15,

$$d_{\mathcal{L}}^{(\delta)}(i,j) = \frac{2}{(\delta - 1)\ln(n)} \sum_{a \in A_{\mathbf{B}}} \left\{ N_n(L_i, a) \ln\left(\frac{N_n(L_i, a)}{N_n(L_i)}\right) + N_n(L_j, a) \ln\left(\frac{N_n(L_j, a)}{N_n(L_j)}\right) - N_n(L_{ij}, a) \ln\left(\frac{N_n(L_{ij}, a)}{N_n(L_{ij})}\right) \right\}.$$

Agrupe os estados $s \in S^*$ cujas frequências observadas na *string* $t_{\mathbf{B}}$ sejam menores que $\alpha^{(\delta)}$ em uma única parte e aplique o algoritmo A.1 (adaptado à $d_{\mathcal{L}}^{(\delta)}$) para os demais estados. Denomine a partição estimada por $\widehat{\mathcal{L}}$.
Compute $\widehat{H}$ como na equação (4.3) e compute $Q = \lceil 2^{\widehat{H}} \rceil$.
Aplique o algoritmo A.4 com $t = t_{\mathbf{B}}$ e $k = Q$, e obtenha o conjunto de pontos centrais D.
Adeque os valores de $\mathbf{B}$ e $t_{\mathbf{B}}$ utilizando o conjunto de pontos centrais D, conforme a figura 4.5, e obtenha $\widehat{\mathbf{B}}$ e $t_{\widehat{\mathbf{B}}}$, respectivamente.
Retorne $\widehat{\mathbf{B}}$ e $t_{\widehat{\mathbf{B}}}$.

Algoritmo A.6 : Obtenção das *strings* binárias associadas ao alfabeto base e subalfabetos.

Sejam $\mathbf{B}$ uma matriz de dimensão $K \times W$, com entradas provenientes do alfabeto ordenado $A_{\mathbf{B}} = \{a_1, \ldots, a_m\}$, e $t_{\mathbf{B}}$ sua H-leitura.
Compute $l = \lceil \sqrt{m} \rceil$, $V = m$ e $m_0 = 0$.
Para $i \in \{1, \ldots, l\}$ **Faça**
  Compute $m_i = \left\lceil \frac{V}{(l-i+1)} \right\rceil$.
  Atualize $V = V - m_i$.
  Compute $A_i = \{a_{\sum_{j<i} m_j + 1}, \ldots, a_{\sum_{j \leq i} m_j}\}$.
Finalize Para
Defina o conjunto de índices $A_{\mathbf{B}}^{base} = \{1, \ldots, l\}$ e $t^{base} = \emptyset$, em que $\emptyset$ é a *string* vazia.
Para $i \in \{1, \ldots, K \times W\}$ **Faça**
  Encontre j tal que $t_i \in A_j$, em que t_i é o i-ésimo elemento da *string* $t_{\mathbf{B}}$.
  Concatene $t^{base} = t^{base}\, j$.
Finalize Para
Construa a matriz $\mathbf{B}^{base}$, de dimensão $K \times W$, a qual t^{base} representa a H-leitura.
Aplique o procedimento de codificação de matrizes, apresentado na figura B.3, para $\mathbf{B}^{base}$, cujas entradas são provenientes do conjunto de índices (alfabeto base) $A_{\mathbf{B}}^{base} = \{1, \ldots, l\}$, e obtenha a *string* binária t^*_{base}.
Para $i \in \{1, \ldots, l\}$ **Faça**
  $t^{(i)} = t_{\mathbf{B}}$.
  Para $j \in \{1, \ldots, K \times W\}$ **Faça**
    Se $t_j \in A_i$ **então**
      $t_j^{(i)} = t_j$, em que $t_j^{(i)}$ é o j-ésimo elemento da *string* $t^{(i)}$.
    Caso contrário
      $t_j^{(i)} = t_j^{base}$, em que t_j^{base} é o j-ésimo elemento da *string* t^{base}.
    Finalize Se
  Finalize Para
  Defina $A_s^{(i)} = A_i \cup \{A_{\mathbf{B}}^{base} \setminus \{i\}\}$ e obtenha, via algoritmo A.1, a partição $\widehat{\mathcal{L}}_s^{(i)}$ associada ao modelo G3M(1,W,1) referente à *string* $t^{(i)}$ e (sub) alfabeto $A_s^{(i)}$.
  Compute o conjunto de códigos de Huffman condicionais $C_i = \{C_i(\cdot|L) : L \in \widehat{\mathcal{L}}_s^{(i)}\}$.
Finalize Para
Defina $t^*_{sub} = \emptyset$.
Compute $C(\cdot)$, código de Huffman sobre as frequências dos elementos em **B2** (ilustrada na figura 3.1).
Para $j \in \{1, \ldots, K \times W\}$ **Faça**
  Se $t_j \in$ **B2 então**
    Concatene $t^*_{sub} = t^*_{sub}\, C(t_j)$.
  Caso contrário
    Encontre i tal que $t_j \in A_i$, e $L \in \widehat{\mathcal{L}}_s^{(i)}$ tal que $(t_{j-W-1}^{(i)}\, t_{j-W}^{(i)}\, t_{j-1}^{(i)}) \in L$.
    Concatene $t^*_{sub} = t^*_{sub}\, C_i(t_j^{(i)}|L)$.
  Finalize Se
Finalize Para
Retorne t^*_{base} e t^*_{sub}.

APÊNDICE B

Figuras

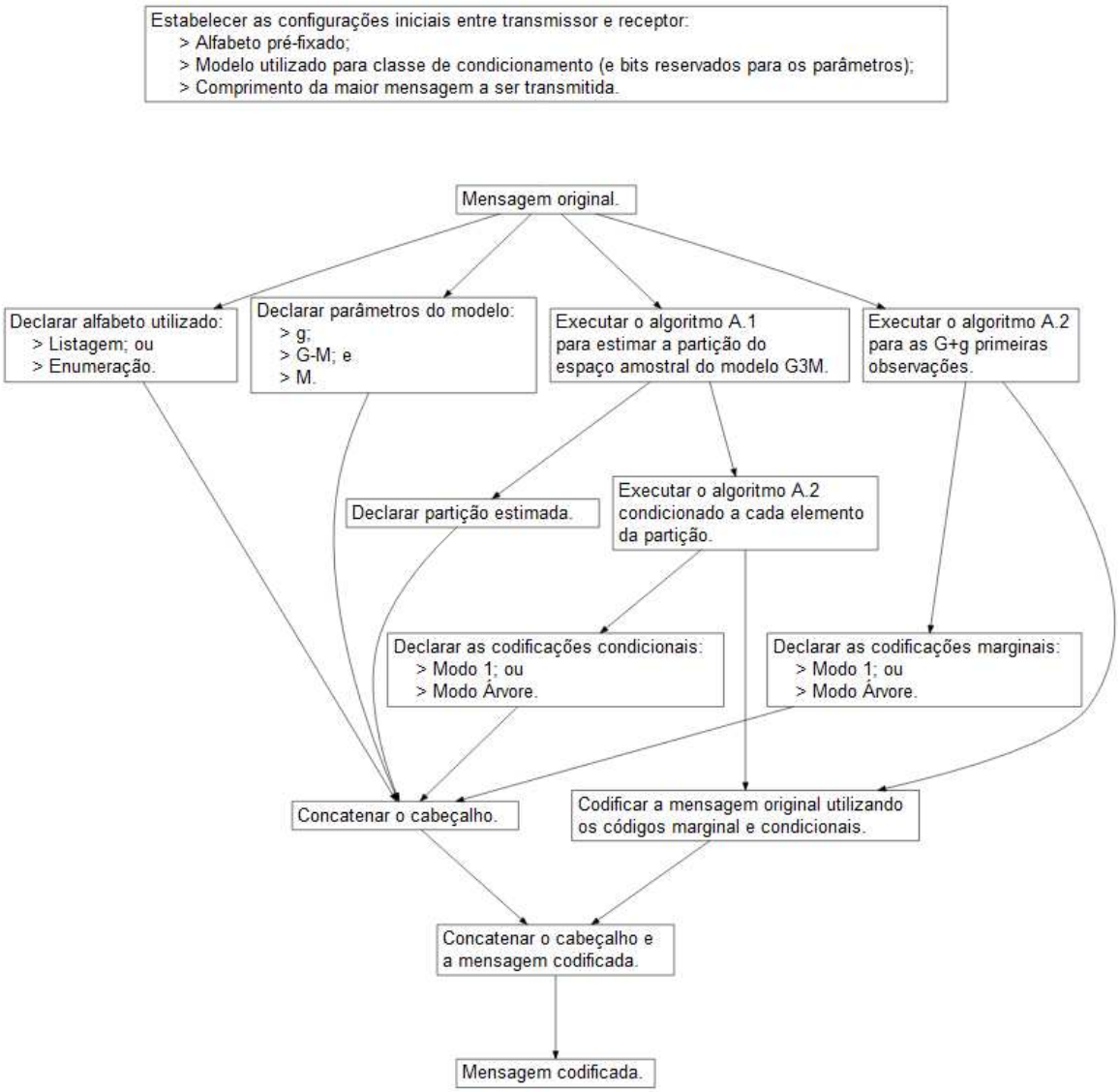

Figura B.1: Fluxograma para codificação de *strings*.

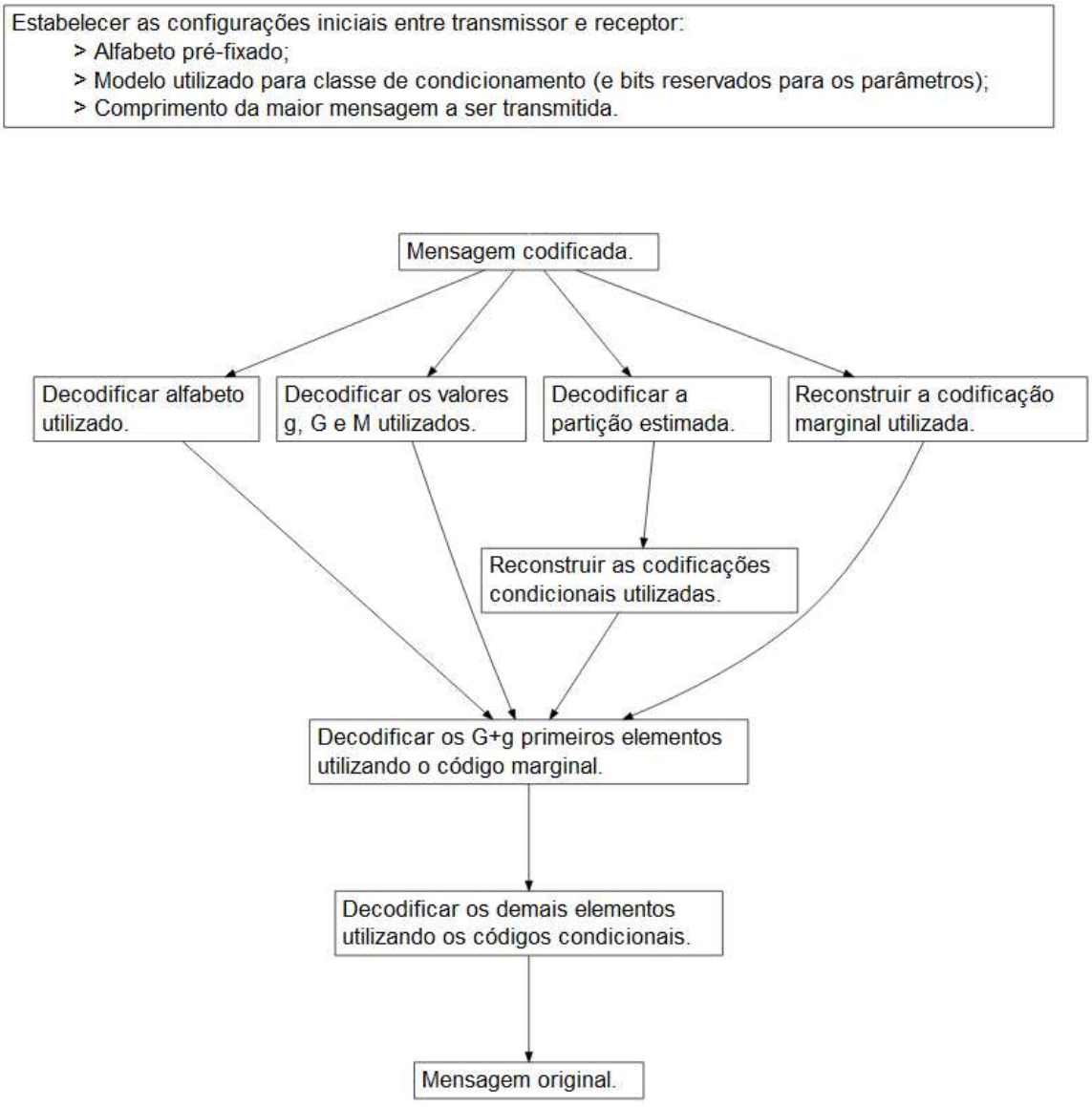

Figura B.2: Fluxograma para decodificação de *strings*.

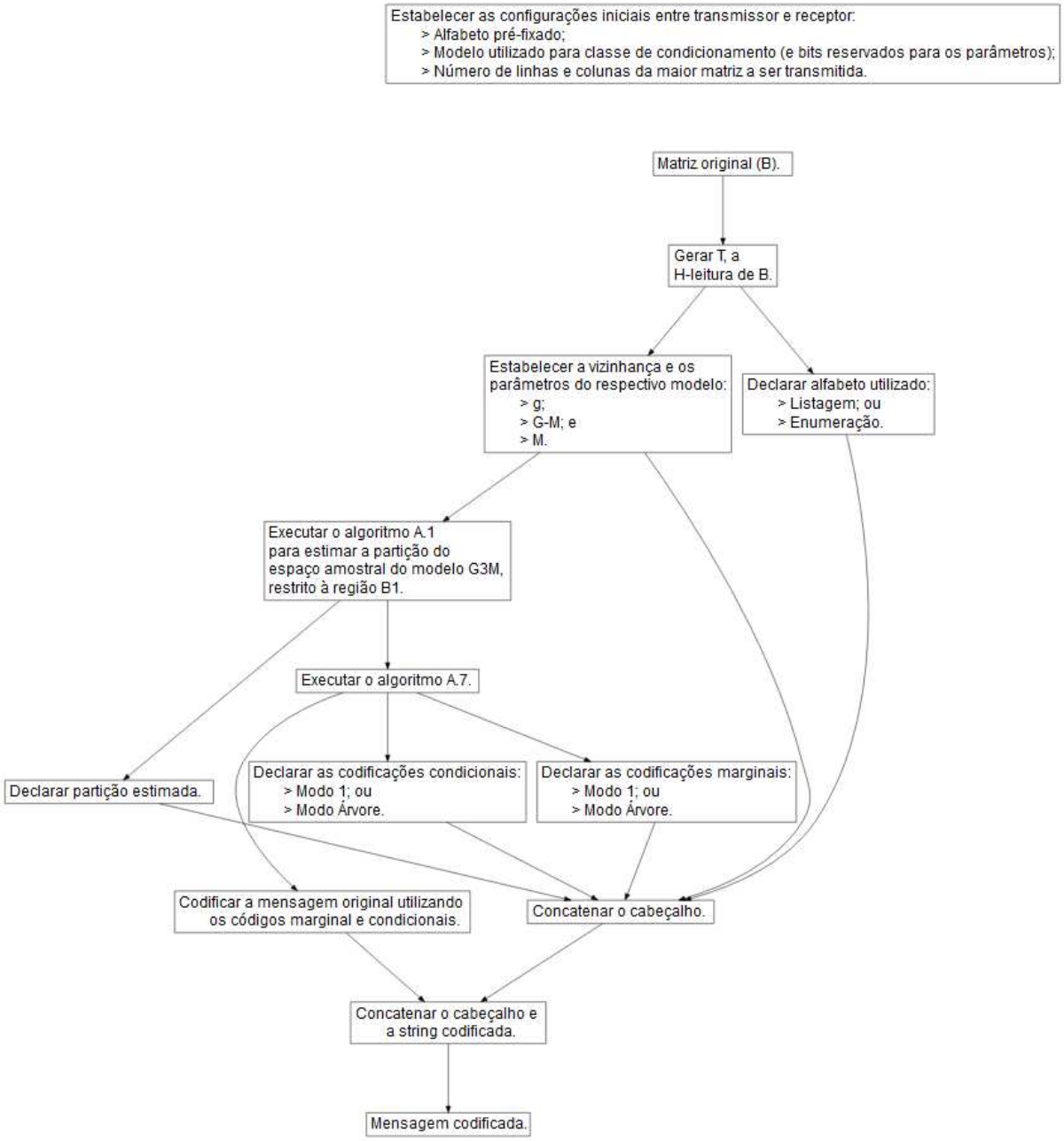

Figura B.3: Fluxograma para codificação de matrizes.

APÊNDICE C

Tabelas

Modelo	N° Partes	BIC	Cabeçalho (*bits*)	Mensagem (*bits*)	Total (*bits*)
G3M(0,1,0)	5	-318.383,78	250	466.929	467.179
G3M(0,2,1)	5	-318.379,56	333	466.928	467.261
G3M(1,2,1)	6	-318.352,19	851	466.809	467.660
G3M(2,2,1)	9	-318.726,44	2.582	467.016	469.598
G3M(0,100,0)	1	-319.161,62	118	467.603	467.721
G3M(0,100,1)	5	-318.191,63	346	466.895	467.241
G3M(0,100,2)	6	-318.145,06	850	466.873	467.723
G3M(1,100,0)	1	-319.159,63	121	467.603	467.724
G3M(1,100,1)	6	-318.133,23	845	466.747	467.592
G3M(1,100,2)	9	-318.539,40	2.577	466.950	469.527
G3M(2,100,0)	3	-319.126,07	516	467.605	468.121
G3M(2,100,1)	8	-318.534,47	2.495	466.946	469.441
G3M(2,100,2)	6	-318.905,96	8.673	467.387	476.060
G3M(0,200,0)	1	-318.965,56	118	467.564	467.682
G3M(0,200,1)	5	-317.997,37	344	466.863	467.207
G3M(0,200,2)	6	-317.965,94	859	466.754	467.613
G3M(1,200,0)	1	-318.964,72	121	467.566	467.687
G3M(1,200,1)	6	-317.935,23	843	466.809	467.652
G3M(1,200,2)	8	-318.336,66	2.511	466.938	469.449
G3M(2,200,0)	5	-318.926,88	736	467.428	468.164
G3M(2,200,1)	8	-318.347,12	2.484	466.921	469.405
G3M(2,200,2)	8	-318.687,76	8.869	467.357	476.226
G3M(0,300,0)	5	-318.346,22	267	467.486	467.753
G3M(0,300,1)	10	-314.847,82	546	463.301	463.847
G3M(0,300,2)	10	-314.782,61	1.077	463.017	464.094
G3M(1,300,0)	11	-313.167,01	581	460.329	460.910
G3M(1,300,1)	**8**	**-298.438,43**	**899**	**439.063**	**439.962**
G3M(1,300,2)	11	-303.933,37	2.671	446.150	448.821
G3M(2,300,0)	11	-312.999,18	1.168	459.656	460.824
G3M(2,300,1)	11	-304.032,95	2.648	446.283	448.931
G3M(2,300,2)	10	-310.714,08	8.889	455.804	464.693
G3M(0,400,0)	1	-318.617,84	118	467.556	467.674
G3M(0,400,1)	5	-317.646,49	346	466.860	467.206
G3M(0,400,2)	8	-317.614,60	1.002	466.387	467.389
G3M(1,400,0)	1	-318.617,00	121	467.557	467.678
G3M(1,400,1)	6	-317.606,83	859	466.625	467.484
G3M(1,400,2)	7	-318.006,65	2.376	467.098	469.474
G3M(2,400,0)	4	-318.584,26	656	467.544	468.200
G3M(2,400,1)	9	-318.059,70	2.563	467.047	469.610
G3M(2,400,2)	8	-318.339,19	8.869	467.218	476.087

Tabela C.1: Modelos G3M ajustados para a mensagem T do exemplo 26 (com $n = 2 \times 10^5$), seus BIC's observados e comprimentos (em *bits*) das respectivas codificações obtidas. Em negrito, o modelo com o maior BIC observado.

Modelo	N° Partes	BIC	Cabeçalho (*bits*)	Mensagem (*bits*)	Total (*bits*)
G3M(0,1,0)	4	-15.9036,62	216	233.000	233.216
G3M(0,2,1)	4	-15.9034,32	289	232.997	233.286
G3M(1,2,1)	5	-15.8991,14	738	233.014	233.752
G3M(2,2,1)	18	-15.8689,82	4.848	231.940	236.788
G3M(0,100,0)	1	-15.9185,59	118	233.268	233.386
G3M(0,100,1)	4	-15.8828,56	299	232.949	233.248
G3M(0,100,2)	5	-15.8807,29	754	232.963	233.717
G3M(1,100,0)	1	-15.9183,39	121	233.267	233.388
G3M(1,100,1)	5	-15.8795,38	744	232.842	233.586
G3M(1,100,2)	14	-15.8517,33	4.389	232.044	236.433
G3M(2,100,0)	4	-15.9145,73	661	233.249	233.910
G3M(2,100,1)	15	-15.8524,28	4.517	232.055	236.572
G3M(2,100,2)	14	-15.8709,94	10.657	232.140	242.797
G3M(0,200,0)	1	-15.8986,18	118	233.227	233.345
G3M(0,200,1)	4	-15.8634,96	298	232.928	233.226
G3M(0,200,2)	5	-15.8605,86	739	232.858	233.597
G3M(1,200,0)	1	-15.8983,98	121	233.226	233.347
G3M(1,200,1)	5	-15.8591,55	754	232.939	233.693
G3M(1,200,2)	16	-15.8310,67	4.634	231.901	236.535
G3M(2,200,0)	5	-15.8941,62	729	233.161	233.890
G3M(2,200,1)	16	-15.8236,52	4.620	231.478	236.098
G3M(2,200,2)	17	-15.8370,25	11.176	231.917	243.093
G3M(0,300,0)	4	-15.8628,28	233	233.212	233.445
G3M(0,300,1)	8	-15.7083,20	464	231.126	231.590
G3M(0,300,2)	9	-15.7016,48	1.019	230.892	231.911
G3M(1,300,0)	9	-15.6096,53	507	229.684	230.191
G3M(1,300,1)	**9**	**-14.8972,34**	**989**	**218.700**	**219.689**
G3M(1,300,2)	16	-14.9220,99	4.570	218.641	223.211
G3M(2,300,0)	10	-15.5951,32	1.079	229.235	230.314
G3M(2,300,1)	20	-14.9334,75	4.987	218.683	223.670
G3M(2,300,2)	16	-15.2955,76	10.942	224.080	235.022
G3M(0,400,0)	1	-15.8645,76	118	233.236	233.354
G3M(0,400,1)	4	-15.8293,55	299	232.961	233.260
G3M(0,400,2)	4	-15.8254,49	661	232.917	233.578
G3M(1,400,0)	1	-15.8644,92	121	233.237	233.358
G3M(1,400,1)	5	-15.8256,94	754	232.950	233.704
G3M(1,400,2)	13	-15.7967,53	4.358	231.685	236.043
G3M(2,400,0)	4	-15.8604,64	661	233.205	233.866
G3M(2,400,1)	14	-15.7970,75	4.400	231.936	236.336
G3M(2,400,2)	14	-15.8157,35	10.711	231.983	242.694

Tabela C.2: Modelos G3M ajustados para uma mensagem T com configurações análogas ao do exemplo 26 (com $n = 10^5$), seus BIC's observados e comprimentos (em *bits*) das respectivas codificações obtidas. Em negrito, o modelo com o maior BIC observado.

Modelo	N° Partes	BIC	Cabeçalho (*bits*)	Mensagem (*bits*)	Total (*bits*)
G3M(0,1,0)	2	-16.053,40	141	23.521	23.662
G3M(0,2,1)	3	-16.049,27	235	23.489	23.724
G3M(1,2,1)	3	-16.052,11	490	23.491	23.981
G3M(2,2,1)	2	-16.069,51	1.446	23.516	24.962
G3M(0,100,0)	1	-15.863,59	118	23.488	23.606
G3M(0,100,1)	2	-15.852,16	191	23.466	23.657
G3M(0,100,2)	3	-15.873,83	499	23.431	23.930
G3M(1,100,0)	1	-15.861,60	121	23.488	23.609
G3M(1,100,1)	4	-15.869,37	573	23.449	24.022
G3M(1,100,2)	2	-15.879,67	1.456	23.488	24.944
G3M(2,100,0)	5	-15.871,32	668	23.410	24.078
G3M(2,100,1)	2	-15.878,05	1.456	23.488	24.944
G3M(2,100,2)	2	-15.880,75	7.938	23.488	31.426
G3M(0,200,0)	1	-15.662,99	118	23.439	23.557
G3M(0,200,1)	2	-15.648,83	191	23.429	23.620
G3M(0,200,2)	4	-15.661,37	588	23.325	23.913
G3M(1,200,0)	1	-15.660,88	121	23.439	23.560
G3M(1,200,1)	5	-15.667,74	645	23.320	23.965
G3M(1,200,2)	2	-15.679,40	1.456	23.436	24.892
G3M(2,200,0)	4	-15.669,06	589	23.362	23.951
G3M(2,200,1)	3	-15.680,73	1.530	23.411	24.941
G3M(2,200,2)	2	-15.679,81	7.938	23.424	31.362
G3M(0,300,0)	1	-15.467,97	118	23.394	23.512
G3M(0,300,1)	4	-15.384,33	279	23.283	23.562
G3M(0,300,2)	7	-15.410,57	770	23.109	23.879
G3M(1,300,0)	4	-15.271,14	286	23.048	23.334
G3M(1,300,1)	**7**	**-14.680,99**	**730**	**22.015**	**22.745**
G3M(1,300,2)	5	-15.214,91	1.623	22.892	24.515
G3M(2,300,0)	5	-15.281,31	667	22.994	23.661
G3M(2,300,1)	5	-15.292,11	1.613	23.007	24.620
G3M(2,300,2)	4	-15.430,59	8.036	23.230	31.266
G3M(0,400,0)	1	-15.306,58	118	23.414	23.532
G3M(0,400,1)	2	-15.304,71	191	23.405	23.596
G3M(0,400,2)	4	-15.308,24	585	23.372	23.957
G3M(1,400,0)	1	-15.304,22	121	23.412	23.533
G3M(1,400,1)	4	-15.301,53	590	23.349	23.939
G3M(1,400,2)	2	-15.320,01	1.456	23.390	24.846
G3M(2,400,0)	3	-15.305,38	501	23.374	23.875
G3M(2,400,1)	2	-15.321,30	1.456	23.392	24.848
G3M(2,400,2)	2	-15.325,18	7.938	23.409	31.347

Tabela C.3: Modelos G3M ajustados para uma mensagem T com configurações análogas ao do exemplo 26 (com $n = 10^4$), seus BIC's observados e comprimentos (em *bits*) das respectivas codificações obtidas. Em negrito, o modelo com o maior BIC observado.

Modelo	N° Partes	BIC	Cabeçalho (*bits*)	Mensagem (*bits*)	Total (*bits*)
G3M(0,1,0)	1	-1.700,75	101	2.501	2.602
G3M(0,2,1)	3	-1.692,88	234	2.414	2.648
G3M(1,2,1)	11	-1.562,17	1.210	2.138	3.348
G3M(2,2,1)	16	-936,82	3.976	1.518	5.494
G3M(0,100,0)	1	-1.510,67	118	2.485	2.603
G3M(0,100,1)	1	-1.510,67	121	2.485	2.606
G3M(0,100,2)	9	-1.375,95	1.076	2.149	3.225
G3M(1,100,0)	2	-1.503,08	191	2.405	2.596
G3M(1,100,1)	9	-1.357,92	1.088	2.135	3.223
G3M(1,100,2)	12	-813,39	3.562	1.632	5.194
G3M(2,100,0)	9	-1.372,39	1.096	2.150	3.246
G3M(2,100,1)	14	-852,79	3.695	1.641	5.336
G3M(2,100,2)	10	-416,65	10.364	1.375	11.739
G3M(0,200,0)	1	-1.318,87	118	2.446	2.564
G3M(0,200,1)	2	-1.315,11	191	2.417	2.608
G3M(0,200,2)	9	-1.202,34	1.091	2.177	3.268
G3M(1,200,0)	2	-1.313,46	191	2.396	2.587
G3M(1,200,1)	9	-1.179,98	1.091	2.149	3.240
G3M(1,200,2)	13	-759,85	3.402	1.717	5.119
G3M(2,200,0)	9	-1.166,44	1.097	2.113	3.210
G3M(2,200,1)	13	-714,90	3.440	1.692	5.132
G3M(2,200,2)	11	-381,02	10.193	1.482	11.675
G3M(0,300,0)	1	-1.124,21	118	2.407	2.525
G3M(0,300,1)	1	-1.124,21	121	2.407	2.528
G3M(0,300,2)	8	-1.007,31	1.025	2.172	3.197
G3M(1,300,0)	2	-1.114,77	191	2.375	2.566
G3M(1,300,1)	10	-929,94	1.147	2.030	3.177
G3M(1,300,2)	12	-595,21	3.165	1.788	4.953
G3M(2,300,0)	9	-971,65	1.074	2.131	3.205
G3M(2,300,1)	13	-582,93	3.269	1.753	5.022
G3M(2,300,2)	9	-327,03	9.758	1.627	11.385
G3M(0,400,0)	1	-961,66	118	2.417	2.535
G3M(0,400,1)	2	-961,28	191	2.400	2.591
G3M(0,400,2)	8	-861,30	1.008	2.209	3.217
G3M(1,400,0)	2	-957,46	191	2.386	2.577
G3M(1,400,1)	9	-832,05	1.083	2.164	3.247
G3M(1,400,2)	11	-509,68	2.919	1.856	4.775
G3M(2,400,0)	9	-807,52	1.073	2.147	3.220
G3M(2,400,1)	11	-504,12	2.996	1.866	4.862
G3M(2,400,2)	**9**	**-267,55**	**9.564**	**1.740**	**11.304**

Tabela C.4: Modelos G3M ajustados para uma mensagem T com configurações análogas ao do exemplo 26 (com $n = 10^3$), seus BIC's observados e comprimentos (em *bits*) das respectivas codificações obtidas. Em negrito, o modelo com o maior BIC observado.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	27	27
	Definição do modelo	24	24
	Partição estimada	778	0
	Codificações condicionais	287	5.048
	Codificações marginais	30	30
	Mensagem codificada	4.228	3.966
	Total	5.374	9.095

Tabela C.5: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 2 \times 10^3$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 28.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	27	27
	Definição do modelo	24	24
	Partição estimada	584	0
	Codificações condicionais	240	6.471
	Codificações marginais	30	30
	Mensagem codificada	43.798	42.987
	Total	44.703	49.539

Tabela C.6: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 2 \times 10^4$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 28.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	27	27
	Definição do modelo	24	24
	Partição estimada	532	0
	Codificações condicionais	240	6.480
	Codificações marginais	30	30
	Mensagem codificada	438.444	436.757
	Total	439.297	443.318

Tabela C.7: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 2 \times 10^5$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 28.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	21	21
	Definição do modelo	16	16
	Partição estimada	46	0
	Codificações condicionais	60	108
	Codificações marginais	10	10
	Mensagem codificada	1.429	1.429
	Total	1.582	1.584

Tabela C.8: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 10^3$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 29.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	21	21
	Definição do modelo	16	16
	Partição estimada	56	0
	Codificações condicionais	84	108
	Codificações marginais	10	10
	Mensagem codificada	7.224	7.224
	Total	7.411	7.379

Tabela C.9: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 5 \times 10^3$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 29.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	21	21
	Definição do modelo	16	16
	Partição estimada	60	0
	Codificações condicionais	96	108
	Codificações marginais	10	10
	Mensagem codificada	14.424	14.424
	Total	14.627	14.579

Tabela C.10: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 10^4$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 29.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	21	21
	Definição do modelo	16	16
	Partição estimada	21	0
	Codificações condicionais	24	108
	Codificações marginais	10	10
	Mensagem codificada	1.277	1.277
	Total	1.369	1.432

Tabela C.11: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 10^3$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 30.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	21	21
	Definição do modelo	16	16
	Partição estimada	21	0
	Codificações condicionais	24	108
	Codificações marginais	11	11
	Mensagem codificada	6.460	6.460
	Total	6.553	6.616

Tabela C.12: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 5 \times 10^3$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 30.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	21	21
	Definição do modelo	16	16
	Partição estimada	21	0
	Codificações condicionais	24	108
	Codificações marginais	10	10
	Mensagem codificada	12.921	12.921
	Total	13.013	13.076

Tabela C.13: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 10^4$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 30.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	21	21
	Definição do modelo	16	16
	Partição estimada	8	0
	Codificações condicionais	12	108
	Codificações marginais	10	10
	Mensagem codificada	1.316	1.314
	Total	1.383	1.469

Tabela C.14: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 10^3$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 31.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	21	21
	Definição do modelo	16	16
	Partição estimada	8	0
	Codificações condicionais	12	108
	Codificações marginais	10	10
	Mensagem codificada	6.608	6.608
	Total	6.675	6.763

Tabela C.15: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 5 \times 10^3$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 31.

	Fonte	MMP	MMC
Cabeçalho	Alfabeto utilizado	21	21
	Definição do modelo	16	16
	Partição estimada	21	0
	Codificações condicionais	24	108
	Codificações marginais	10	10
	Mensagem codificada	13.149	13.149
	Total	13.241	13.304

Tabela C.16: Elementos da *string* binária resultante da compressão da amostra de tamanho $n = 10^4$ pelas abordagens de modelo markoviano de partição (MMP) e modelo markoviano completo (MMC), e seus respectivos números de *bits*, referentes ao exemplo 31.

APÊNDICE D

Declaração de Componentes do Cabeçalho

Este apêndice tem o objetivo de complementar os detalhes técnicos necessários para o entendimento das metodologias empregadas nos processos de codificação desenvolvidos no capítulo 3. Delineamos com um pouco mais de detalhes os métodos básicos de transmissão binária de alfabetos e códigos de Huffman, que são componentes essenciais do cabeçalho da mensagem binária a ser transmitida, segundo a abordagem deste texto.

D.1 Declaração de Alfabetos

Consideremos o caso em que um transmissor deseja comunicar, de maneira binária, uma determinada mensagem, s, a um receptor. Suponha que ambas as partes tenham previamente combinado qual o universo de elementos passível de compor as mensagens a serem transmitidas. Denominemos esse conjunto por alfabeto prefixado (ou preestabelecido) e o denotemos por A. No entanto, a mensagem s pode ser composta por elementos de um subconjunto A_1 de A apenas, $A_1 \subseteq A$. O subconjunto A_1 é chamado de alfabeto associado à mensagem s, mas, por simplicidade, o chamamos apenas de alfabeto. Neste tópico exploramos duas estratégias para a comunicação do alfabeto A_1 de maneira binária: listagem; e enumeração.

D.1.1 Método de Listagem

Suponha que o alfabeto prefixado A pode ser ordenado segundo algum critério (ordem lexicográfica, por exemplo). Dessa forma, dado um alfabeto $A_1 \subseteq A$, podemos definir $\mathcal{A}_{1,0}$ como a *string* binária de comprimento $|A|$, em que $|\cdot|$ representa a cardinalidade do conjunto em seu argumento. Assim, $\mathcal{A}_{1,0} = a_1 a_2 \dots a_{|A|}$, $a_i \in \{0, 1\}$ e $1 \leq i \leq |A|$, tal que $a_i = 1$ se o i-ésimo elemento de A também pertence a A_1, e $a_i = 0$ caso contrário. Esta abordagem de transmissão do alfabeto é denominada por método de listagem.

D.1.2 Método de Enumeração

Uma vez que o alfabeto prefixado é ordenado, podemos associar facilmente um índice a cada um de seus elementos, por exemplo, ao primeiro elemento associamos o valor 0, ao segundo o valor 1 e assim sucessivamente até o $|A|$-ésimo elemento assumir o valor $|A| - 1$. O método de enumeração consiste na construção da *string* binária $\mathcal{A}_{1,1}$ que é uma sequência de blocos de $\lceil \log(|A| + 1) \rceil$ *bits* cada, os quais representam as coordenadas dos elementos do alfabeto prefixado que estão presentes no alfabeto A_1, em que log denota o logaritmo de base 2 e $\lceil x \rceil$ é o menor inteiro maior ou igual a x. Além disso, a enumeração necessita da inclusão de um elemento de parada de leitura (denotamos por "!"), que indica que não há mais elementos a serem lidos na *string* codificada que servirão para a identificação do alfabeto. Devido ao acréscimo desse elemento é que os blocos possuem tamanho $\lceil \log(|A| + 1) \rceil$ *bits* e não $\lceil \log |A| \rceil$.

D.2 Declaração de Códigos de Huffman

Atemo-nos novamente ao contexto de comunicação de uma mensagem s, em que cada elemento que a compõe provém do alfabeto A_1. Como mencionado anteriormente, a transmissão de s será feita de forma binária. Sendo assim, devemos estabelecer um sistema de codificação binário sobre s que permita sua comunicação. Denotemos por $C(\cdot)$ um código binário, ou seja, C é uma função $C : A_1 \to \mathcal{B}^*$, em que $\mathcal{B}^*$ é o conjunto das *strings* binárias de comprimento finito constituídas por elementos do alfabeto binário $\mathcal{B} = \{0, 1\}$. Assim, representamos por $C(x)$ o código correspondente ao elemento x, $\forall x \in A_1$. Considerando um pequeno abuso de linguagem, definamos $C(s)$ a *string* binária resultante da concatenação da codificação de cada elemento de s por C, ou seja, se

$s = x_1x_2 \dots x_n$, $x_i \in A_1$ e $1 \leq i \leq n$, então $C(s) = C(x_1x_2 \dots x_n) = C(x_1)C(x_2) \dots C(x_n)$. Para a comunicação da mensagem s, é a *string* binária $C(s)$ que efetivamente é transmitida e interpretada pelo receptor. Para que este último possa recuperar a informação original, devemos informá-lo sobre o sistema de codificação utilizado para se obter $C(s)$. Neste texto, consideramos C como codificações de Huffman apenas. Dessa forma, apresentamos três estratégias para que o transmissor possa informar binariamente ao receptor quais os códigos de Huffman utilizados: Modo 0; Modo 1; e Modo Árvore.

D.2.1 Modo 0

Suponha C um código de Huffman de prefixo. Uma maneira de determinar a composição binária de $C(x)$, $x \in A_1$, é a partir da comunicação das frequências observadas de cada $x \in A_1$ na mensagem s. Assim sendo, a transmissão binária do código C utilizado para codificar a mensagem s pode ser feita pela transmissão binária das frequências observadas de x na mensagem s, para cada $x \in A_1$. O aqui denominado "Modo 0"corresponde à estratégia mais ingênua para transmissão de contagens e possui um desenvolvimento muito simples. Suponha que transmissor e receptor têm preestabelecido qual o número máximo de elementos, N, que uma mensagem pode ter e seja $\mathcal{N} = \{n_1 \dots, n_{|A_1|}\}$ o conjunto de frequências, em que n_i é a frequência absoluta do i-ésimo elemento do conjunto A_1 na mensagem original, $i \in \{1, \dots, |A_1|\}$. Repare que cada $n_i \in \mathcal{N}$, $i \in \{1, \dots, |A_1|\}$, pode ser representado binariamente por uma *string* de comprimento $\lfloor \log N \rfloor + 1$ *bits*, em que $\lfloor x \rfloor$ é o maior inteiro menor ou igual a x. O limite de $\lfloor \log N \rfloor + 1$ *bits* serve para nos precavermos do "pior dos casos"em que um único elemento do alfabeto é predominante na mensagem s. Dessa forma, as contagens podem ser transmitidas pela concatenação

$$\mathcal{D}_0 = I(n_1, \lfloor \log N \rfloor + 1)\, I(n_2, \lfloor \log N \rfloor + 1)\ \dots\ I(n_{|A_1|}, \lfloor \log N \rfloor + 1).$$

Nota 16. *O Modo 0 pode ser incrementado se, previamente à string $\mathcal{D}_0$, inserirmos o comprimento n da mensagem s, ou seja, podemos fazer a concatenação $I(n, \lfloor \log N \rfloor + 1)\mathcal{D}_0$. Dessa forma, podemos escrever $\mathcal{D}_0$ como*

$$\mathcal{D}_0 = I(n_1, \lfloor \log n \rfloor + 1)\, I(n_2, \lfloor \log n \rfloor + 1)\ \dots\ I(n_{|A_1|}, \lfloor \log n \rfloor + 1),$$

cuja representação pode ser mais econômica que a descrita originalmente. Outra possibilidade é definirmos $n^* = \max_{n_i \in \mathcal{N}} n_i$ *que indica qual a maior frequência observada em* $\mathcal{N}$ *e, portanto, necessita do maior número de bits para ser expressa. Com isso, podemos fazer a concatenação* $I(n^*, \lfloor \log N \rfloor + 1)\mathcal{D}_0$ *e escrever* $\mathcal{D}_0$ *como*

$$\mathcal{D}_0 = I(n_1, \lfloor \log n^* \rfloor + 1)\, I(n_2, \lfloor \log n^* \rfloor + 1)\ \ldots\ I(n_{|A_1|}, \lfloor \log n^* \rfloor + 1).$$

Esta última representação pode ser ainda mais econômica que as demais. O uso ou não de tais estratégias fica a critério do programador.

D.2.2 Modo 1

Consideremos o mesmo contexto apresentado em D.2.1, em que desejamos transmitir o conjunto de contagens $\mathcal{N} = \{n_1 \ldots, n_{|A_1|}\}$ para a determinação do código de Huffman correspondente. Pela estrutura do Modo 0, observamos algo que pode comprometer sua eficiência: o uso de representações com mesma quantidade de *bits* para todos os inteiros envolvidos. Por exemplo, o inteiro 300 necessita de $\lfloor \log 300 \rfloor + 1 = 9$ *bits* para ser representado, enquanto 50 precisa de $\lfloor \log 50 \rfloor + 1 = 6$ *bits*. Como o Modo 0 utiliza representações de comprimentos fixos, alguns *bits* podem ser desnecessários à codificação. Com essa motivação, o "Modo 1"apresenta uma possibilidade de se declarar as contagens utilizando *strings* de comprimentos variáveis. Assim, iniciemos calculando a representação binária dos elementos de $\mathcal{N}$ utilizando o mínimo de *bits* para cada uma. Definamos então, $\mathcal{N}_B = \{I(n_1, \lfloor \log n_1 \rfloor + 1), \ldots, I(n_{|A_1|}, \lfloor \log n_{|A_1|} \rfloor + 1)\}$. Consideremos $\mathcal{J}$ o conjunto do número de *bits* utilizados para representar cada elemento de $\mathcal{N}_B$, ou seja, $\mathcal{J} = \{j_1, \ldots, j_{|A_1|}\}$, em que $j_i = \lfloor \log n_i \rfloor + 1$, para $i \in \{1, \ldots, |A_1|\}$. Definamos também $j^* = \max_{j_i \in \mathcal{J}} j_i$. Com esses valores, as contagens podem ser transmitidas pela concatenação

$$\begin{aligned}\mathcal{D}_1 = \ & I(\lfloor \log j^* \rfloor + 1, \lfloor \log(\lfloor \log N \rfloor + 1) \rfloor + 1)\, I(j_1, \lfloor \log j^* \rfloor + 1)\, I(n_1, j_1) \\ & \ldots\ I(j_{|A_1|}, \lfloor \log j^* \rfloor + 1)\, I(n_{|A_1|}, j_{|A_1|}).\end{aligned}$$

O Modo 1 é um procedimento mais complexo e seu detalhamento é conveniente. Idealmente, gostaríamos de representar cada elemento de $\mathcal{N}$ pelo menor número de *bits* possível. Essa representação é obtida pelos elementos $\mathcal{N}_B = \{I(n_1, \lfloor \log n_1 \rfloor +$

$1), \ldots, I(n_{|A_1|}, \lfloor \log n_{|A_1|} \rfloor + 1)\}$. No entanto, a não uniformidade do comprimento dessas *strings* binárias impossibilita um decodificador estabelecer quando um inteiro termina e outro começa. Para contornar esse problema, para cada elemento de $\mathcal{N}_B$, podemos incluir um prefixo informando o número utilizado de *bits* para a representação do inteiro em questão. Esses prefixos são os denominados j_i (em que $j_i = \lfloor \log n_i \rfloor + 1$) e constituem o conjunto de prefixos $\mathcal{J}$. No entanto, cada um desses prefixos também pode possuir representações binárias com diferentes comprimentos. Por isso, definimos um limitante superior para o comprimento desses prefixos: $\lfloor \log j^* \rfloor + 1$, em que $j^* = \max_{j_i \in \mathcal{J}} j_i$. Novamente, para nos precavermos do pior dos casos, esse limitante é representado por $\lfloor \log(\lfloor \log N \rfloor + 1) \rfloor + 1$ *bits*, em que N é o comprimento máximo (preestabelecido entre transmissor e receptor) das mensagens a serem transmitidas. Assim, ao iniciar a leitura da *string* binária que representa a transmissão das frequências pelo Modo 1 ($\mathcal{D}_1$), um decodificador saberá que deve interpretar os $\lfloor \log(\lfloor \log N \rfloor + 1) \rfloor + 1$ primeiros *bits* como o valor do inteiro j^*, ou seja, terá a informação que cada frequência a ser lida terá um prefixo de j^* *bits*. Em seguida, lerá os próximos j^* *bits* e os interpretará como o valor inteiro j_1 (índice que determina o número de *bits* necessários para a representação da frequência n_1). A seguir, deve ler os próximos j_1 *bits* e os interpretar como o inteiro que representa a frequência n_1. Em seguida, lerá os próximos j^* *bits* e os interpretará como o valor inteiro j_2 e assim sucessivamente até que todas as frequências n_i, $i \in \{1, \ldots |A_1|\}$ sejam estabelecidas.

D.2.3 Modo Árvore

Os Modos 0 e 1 apresentados anteriormente constituem abordagens para declaração de códigos de Huffman baseadas na codificação de um conjunto de frequências. No entanto, tais procedimentos podem não ser os mais mais econômicos em diversas situações, variando conforme o tamanho do alfabeto e do conjunto de frequências considerados. Por este motivo, estabelecemos mais uma técnica para transmissão de códigos de Huffman: o Modo Árvore. Um código de prefixo de Huffman pode ser visto como uma árvore binária em que o trajeto até cada folha (nó terminal) representa a codificação de um elemento do alfabeto original. Por exemplo, o código de Huffman apresentado na equação (D.1),

$$C(a) = 0, \quad C(h) = 11 \quad \text{e} \quad C(w) = 10, \tag{D.1}$$

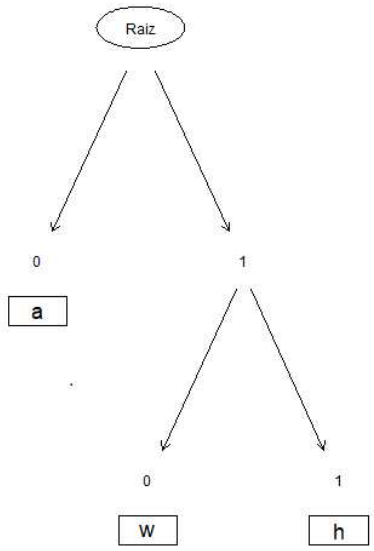

Figura D.1: Árvore binária para o código de Huffman apresentado na equação (D.1).

pode ser expresso pela árvore binária apresentada na figura D.1.

Então, devemos estabelecer um procedimento que permita converter uma árvore binária em uma *string* binária e que esta possa reconstruir de maneira única a árvore original. Introduzimos um conjunto de algoritmos que permite a construção de uma árvore binária referente a uma codificação de Huffman, a transformação da árvore em *string*, reconstrução da *string* em árvore e conversão da árvore na codificação de Huffman original. Dessa forma, teremos disponível mais uma opção para a comunicação dos códigos de Huffman utilizados para codificar uma mensagem, juntamente com os métodos de transmissão de contagens (Modo 0 e Modo 1). Os algoritmos D.7 e D.8 são os procedimentos utilizados pelo transmissor para a comunicação do código de Huffman utilizado, enquanto os algoritmos D.9 e D.10 são os procedimentos utilizados pelo receptor para a recuperação de tal codificação. Utilizamos como ilustração do procedimento o simples código apresentado na equação (D.1).

Para o primeiro algoritmo (transformação da codificação de Huffman em uma árvore binária), devemos impor algumas regras. As regras primárias são referentes à leitura e ordenação de *strings* binárias. A leitura da *string* acontecerá da esquerda para a direita, ou seja, em uma *string* $s = x_1x_2 \ldots x_n$, $x_i \in \{0, 1\}$ e $i \in \{1, \ldots, n\}$, o primeiro elemento é x_1, o segundo é x_2 e o último é x_n. A ordenação de duas *strings*, $s_1 = x_1x_2 \ldots x_{n_1}$ e

Algoritmo D.7 : Construção de uma árvore binária a partir de um código de Huffman.

Seja F o conjunto cujos elementos são os códigos de uma codificação de prefixo de Huffman para um alfabeto A.
Ordene os códigos de $F = \{f_1, \ldots, f_{|A|}\}$.
Inicie a árvore O apenas com o nó "Raiz".
Para $i \in \{1, \ldots, |A|\}$ **Faça**
  Trace em O a trajetória Raiz $\rightarrow f_i$ seguindo a ordem de leitura para *strings* binárias.
Finalize Para
Retorne O.

$s_2 = y_1 y_2 \ldots y_{n_2}$, em que $x_i, y_j \in \{0, 1\}$, $i \in \{1, \ldots, n_1\}$ e $j \in \{1, \ldots, n_2\}$, ocorre pela comparação de um par de coordenadas por vez. Dessa forma, a ordenação ocorre pela comparação lexicográfica dos elementos seguindo a ordem de leitura da *string*. No caso em que s_1 é prefixo de s_2, s_1 será considerada de ordem inferior. Por exemplo, para o conjunto de *strings* $\{010, 1, 0001, 000, 00\}$ o conjunto ordenado é $\{00, 000, 0001, 010, 1\}$. Com essas regras, podemos construir uma árvore binária referente a uma codificação de Huffman, conforme o algoritmo D.7. Nele, iniciamos ordenando o conjunto imagem do código em questão. Para o código em (D.1), temos o conjunto ordenado $F = \{C(a) = 0, C(w) = 10, C(h) = 11\}$. F é o conjunto de trajetórias até as folhas (nós terminais) da árvore binária (denominada O), definindo-a biunivocamente. O resultado desse procedimento já foi ilustrado na figura D.1.

O segundo algoritmo, que consiste na conversão de uma árvore binária em uma *string* binária, é apresentado pelo algoritmo D.8. Para que ele opere de maneira apropriada devemos impor um par de regras sobre a leitura de uma árvore. A primeira é que o avanço para níveis superiores tem prioridade pelo valor 0, ou seja, se a leitura está em um nó interno (nó que não é folha), então o próximo nó a ser lido é o nó cuja trajetória segue para o valor 0. A segunda regra é referente à leitura de folhas. Após uma folha ser computada, o próximo elemento a ser lido é, se houver, um nó anterior que possui apenas uma ramificação até então. Quando, após ler uma folha, não existirem mais nós internos com apenas uma ramificação, o processo de leitura está concluído. O algoritmo D.8 faz uso dessa regra de uma forma simples: cada nó visitado recebe um *bit* de identificação que assume valor 0 no caso de ser nó interno e valor 1 no caso de ser folha. Utilizemos novamente o código em (D.1) e a árvore ilustrada na figura D.1. Chamemos de η a concatenação dos *bits* de identificação propostos na leitura. Iniciamos no nó "Raiz", que é um nó interno e, portanto, sua identificação é 0. Dessa forma, temos,

até aqui, $\eta = 0$. Pela preferência de avanço na tragetória, temos que o próximo nó a ser lido é 0. Como este é uma folha, sua identificação é 1. Com isso, η é atualizado para $\eta = 01$. Como o último elemento lido é uma folha, devemos retornar para o último nó interno com apenas uma ramificação (neste caso, a própria "Raiz") e avançar para o sentido 1. Portanto, o próximo nó a ser lido é 1, que é um nó interno e, portanto, sua identificação é 0. Assim, η é atualizado para $\eta = 010$. O próximo elemento a ser lido é o nó 10, que é uma folha, e, portanto, sua identificação é 1. Dessa forma, $\eta = 0101$. Por fim, retornamos para o último nó com apenas uma ramificação e avançamos para o sentido 1. Portanto, o próximo nó a ser lido é 11, que é uma folha. Com isso, $\eta = 01011$. Como não existem mais nós internos com apenas uma ramificação, a leitura da árvore binária está concluída. Perceba que não existe nenhum indicativo que ligue o conjunto de folhas com os elementos do alfabeto. Dessa forma, precisamos também enviar uma *string* que permita identificar os elementos do alfabeto utilizado no conjunto de folhas. Isso pode ser feito pela comunicação da concatenação da representação binária (cada uma contendo $\lceil \log |A| \rceil$ *bits*) da ordem da folha dentro do alfabeto utilizado. Em nosso caso, o elemento a é o primeiro de A, h o segundo e w o terceiro. Portanto, se iniciarmos a contagem dos elementos de A em 0 (por economia), junto com a *string* η devemos transmitir as identicações $I(0,2)I(2,2)I(1,2) = 00\,10\,01$. Definindo η^* como a concatenação de η e a *string* de identificação, temos

$$\eta^* = \eta\,00\,10\,01 = 01011001001. \tag{D.2}$$

Agora, iremos desenvolver um par de algoritmos que será utilizado pelo receptor para reconstruir o sistema de codificação original. O algoritmo D.9 consiste na conversão de uma *string* binária em árvore. Para seu funcionamento apropriado, basta utilizarmos as regras de leitura e trajetória já introduzidas. Utilizemos como exemplo a *string* apresentada na equação (D.2). Como o primeiro elemento de η^* é 0, sabemos que trata-se de um nó interno, neste caso, a "Raiz". Segundo a regra de leitura estabelecida, devemos avançar no sentido 0. O segundo elemento de η^* é 1, indicando que o nó atual, 0, é uma folha. Como o último nó computado foi uma folha, devemos retornar ao último nó interno com apenas uma ramificação (neste caso, a "Raiz") e avançar no sentido 1. O nó atual, 1, é um nó interno, pois o terceiro elemento de η^* é 0. Pela regra de leitura, avançamos

Algoritmo D.8 : Construção de uma *string* binária a partir de uma árvore binária de Huffman.

Seja O uma árvore binária referente a um código de Huffman sobre um alfabeto A, com um conjunto de trajetórias $F = \{f_1, \ldots, f_{|A|}\}$.
Inicie a leitura no nó "Raiz" com $\eta = 0$ e $\Gamma = \emptyset$, em que $\emptyset$ representa a *string* vazia.
Enquanto existir nó interno visitado menos de duas vezes **Faça**
  Se a leitura anterior é nó interno **então**
    Avance por 0.
  Caso contrário
    Avance por 1.
  Finalize Se
  Se nó atual é folha **então**
    $\eta = \eta 1$ e $\Gamma = \Gamma I(k, \lceil \log |A| \rceil)$, em que k é a posição do elemento representado pela folha atual dentro do alfabeto A.
    Se existir nó interno visitado apenas uma vez **então**
      Retorne para o último nó interno visitado apenas uma vez.
    Finalize Se
  Caso contrário
    $\eta = \eta 0$.
  Finalize Se
Finalize Enquanto
Retorne $\eta^* = \eta\Gamma$.

no sentido 0, alcançando o nó 10, que é uma folha, pois o quarto elemento de η^* é 1. Como o último nó computado foi uma folha, devemos retornar ao último nó interno com apenas uma ramificação (neste caso, o nó 1) e avançar no sentido 1, chegando ao nó 11. Este nó é uma folha, pois o quinto elemento de η^* é 1. Como não existem mais nós internos com apenas uma ramificação, a árvore binária (denominada O) está completa, restando apenas identificar suas folhas dentro do alfabeto considerado. Como a árvore possui três folhas, a identificação delas está representada em três conjuntos de $\lceil \log 3 \rceil = 2$ *bits* cada, indicando as respectivas posições dentro do alfabeto A. Defina $I^{-1}(k)$ a função que entrega a representação decimal do inteiro binário k. Dessa forma, o conjunto $\Psi = \{I^{-1}(00), I^{-1}(10), I^{-1}(01)\} = \{0, 2, 1\}$ define a ordenação (iniciada em zero) das folhas dentro do alfabeto considerado. Como o alfabeto é $A = \{a, h, w\}$, temos que as folhas representam os elementos de A na ordem $A_\Psi = \{a, w, h\}$. Com isso, terminamos a reconstrução da árvore apresentada na figura D.1.

Resta-nos, agora, construir um código de Huffman tendo como base a árvore O e as identificações Ψ produzidas pelo algoritmo D.9. O algoritmo D.10 incumbe-se dessa tarefa, que pode ser vista como uma inversão da realizada pelo algoritmo D.7. O

Algoritmo D.9 : Construção de uma árvore binária a partir de uma *string* binária.

Seja $\eta^* = x_1 \dots x_n$ uma *string* binária referente a uma árvore binária construída a partir de um código de Huffman sobre um alfabeto A.
Seja O o nó "Raiz" da árvore binária, $j = 1$ e $k = 0$.
"Raiz" é nó interno.
Enquanto $k < |A|$ **Faça**
  Se nó interno atual não possui ramificações **então**
    Avance em O por 0.
  Caso contrário
    Avance em O por 1.
  Finalize Se
  $j = j + 1$.
  Se $x_j = 0$ **então**
    Atualize: nó atual é interno.
  Caso contrário
    Atualize: nó atual é folha e $k = k + 1$.
  Finalize Se
  Se $k < |A|$ **então**
    Retorne ao último nó interno visitado com menos de duas ramificações.
  Finalize Se
Finalize Enquanto
Compute o conjunto $\Psi = \{I^{-1}(\psi_1), \dots, I^{-1}(\psi_k)\}$, em que ψ_i são as *substrings* $\psi_i = x_{j+1+(\lceil \log k \rceil)(i-1)} \dots x_{j+(\lceil \log k \rceil)i}$, $i \in \{1, \dots, k\}$.
Retorne O e Ψ.

funcionamento desse algoritmo é novamente baseado nas regras de leitura e trajetória já estabelecidas anteriormente. Chamemos de $C(\cdot)$ a função de codificação que estamos recuperando. Utilizando a árvore O e as identificações Ψ apresentadas na figura D.1, iniciamos a leitura no nó "Raiz". Como este nó é interno, avançamos pelo sentido 0 até encontrarmos uma folha. O nó 0 é uma folha e, portanto, é um elemento do código de Huffman que está sendo recuperado. Além disso, esse código é referente ao primeiro valor de Ψ, que é 0. Isso quer dizer que o código associado à folha 0 é o primeiro elemento do alfabeto utilizado A, que é a. Ou seja, temos $C(a) = 0$. Como o último nó lido foi uma folha, devemos retornar ao último nó interno com apenas uma ramificação e avançar pelo sentido 1. Dessa forma, avançamos para o nó 1. Este é um nó interno e, portanto, não é um elemento da imagem de $C(\cdot)$. Em seguida, avançamos pelo sentido 0 e alcançamos o nó 10. Como este nó é uma folha, pertence à imagem de $C(\cdot)$. Além disso, esse código é referente ao segundo valor de Ψ, que é 2. Isso quer dizer que o código associado à folha 10 é o terceio elemento do alfabeto utilizado A, que é w. Ou seja, temos $C(w) = 10$. Por fim, retornamos ao nó 1 (último nó visitado com apenas uma ramificação) e avançamos pelo

Algoritmo D.10 : Construção de um código de Huffman a partir de uma árvore binária.

Seja O uma árvore binária referente a um código de Huffman sobre um alfabeto A e $\Psi = \{\psi_1, \ldots, \psi_{|A|}\}$ o conjunto de identificações.
Seja $\Psi_A : \Psi \to A$ a função que retorna o elemento do alfabeto A que está na posição determinada por $\psi \in \Psi$.
Seja $k = 1$ e $c_j \equiv \emptyset$, com $j \in \{1, \ldots, |A|\}$, em que $\emptyset$ é a *string* vazia.
Seja $d = \emptyset$, em que d armazenará a trajetória da raiz até o último nó interno com apenas uma ramificação.
Inicie a leitura no nó "Raiz".
Avance por 0.
Concatene $c_k = c_k 0$.
Enquanto $k \leq |A|$ **Faça**
  Se nó atual é interno **então**
    Se nó atual não possui ramificação **então**
      Avance por 0.
      Concatene $c_k = c_k 0$.
    Caso contrário
      Avance por 1.
      Concatene $c_k = c_k 1$.
    Finalize Se
  Caso contrário
    Retorne c_k.
    Retorne para o último nó interno com apenas uma ramificação.
    $k = k + 1$.
    Atualize d para a trajetória da raiz até o último nó interno com apenas uma ramificação.
    $c_k = d$.
  Finalize Se
Finalize Enquanto
Determine pontualmente a função C de modo que $C(\Psi_A(\psi_j)) = c_j$, para $j \in \{1, \ldots, |A|\}$.

Retorne C.

sentido 1, alcançando o nó 11. Este nó é uma folha e seu referente valor em Ψ é 1. Com isso, $C(h) = 11$. Como não existem mais folhas, a reconstrução do código de Huffman está finalizada. Repare que obtemos com êxito o código C apresentado em (D.1).

Em resumo, os algoritmos D.7 e D.8 constituem um método alternativo para que o transmissor possa comunicar um código de Huffman para o receptor, aqui denominado por "Modo Árvore". Enquanto isso, os algoritmos D.9 e D.10 são utilizados para compreender a transmissão.