Universidade Estadual Paulista "Júlio de Mesquita Filho"
Instituto de Geociências e Ciências Exatas
Campus de Rio Claro

Codificação de Fonte

Juliano de Falque Bonfim

Dissertação apresentada ao Programa de Pós-Graduação – Mestrado Profissional em Matemática Universitária do Departamento de Matemática como requisito parcial para a obtenção do grau de Mestre

Orientadora
Profa. Dra. Eliris Cristina Rizziolli

2014

TERMO DE APROVAÇÃO

Juliano de Falque Bonfim
Codificação de Fonte

Dissertação APROVADA como requisito parcial para a obtenção do grau de Mestre no Curso de Pós-Graduação Mestrado Profissional em Matemática Universitária do Instituto de Geociências e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", pela seguinte banca examinadora:

Profa. Dra. Eliris Cristina Rizziolli
Orientadora

Prof. Dr. Aldício José Miranda
Faculdade de Matemática - Universidade Federal de Uberlândia - MG

Prof. Dr. Henrique Lazari
Departamento de Matemática - IGCE - UNESP/Rio Claro - SP

Rio Claro, 21 de Março de 2014

À Gilberto, meu pai, Madalena, minha mãe, e Fabiano, meu irmão,
minha família a qual amo e sem a qual não vivo.
Aos meus amigos que acreditaram em mim e sempre incentivaram me dando forças
para concluir este sonho.

Agradecimentos

Agradeço primeiramente a Deus pelo dom da vida e por proporcionar mais esta alegria. À minha família: meu pai Gilberto, minha mãe Madalena e meu irmão Fabiano que são meu porto seguro e minha paz, as pessoas que sempre acreditaram em mim e que me inspiraram a sempre seguir em frente. Amo muito vocês!

À professora Elíris Cristina Rizziolli que me acolheu e aceitou a orientação deste trabalho e foi, além de uma grande orientadora, uma grande amiga, paciente e carinhosa, sempre me incentivando e me dando segurança para vencer mais essa etapa.

Ao professor Henrique Lazari que foi um amigo imprescindível e um grande mentor sempre disposto a ajudar no que fosse necessário.

Meus sinceros agradecimento a toda a equipe do Departamento de Matemática por toda a dedicação, orientação e apoio, e em especial às professoras Alice e Suzi pelo acolhimento. Aos demais professores deste programa minha gratidão por tudo o que fizeram por mim e meus colegas, e meus sinceros parabéns pelo belo trabalho que realizaram e realizam.

Aos companheiros de trabalho da E.E. Dom Henrique Mourão de Lins por todo apoio. Em especial agradeço a amiga Rosana Faduti, pela paciência, esforço e carinho que dedicou ao revisar esta dissertação.

Aos meus grandes amigos de Lins por sempre terem me encorajado e ajudado nessa caminhada. Aos amigos de Rio Preto os meus agradecimentos especiais por todo carinho e apoio perante tudo o que enfrentamos juntos até esse momento. Vocês foram muito importantes para que eu chegasse até aqui. Obrigado a todos!

Aos "amigos de viagem": Edmar, Franciéli, Erica, Daniele, João Paulo, Denis, Mariana, Aline, Vinicius, Renato e a todos os outros que aqui não mencionei! Obrigado por todo apoio e companheirismo que dedicaram nas horas boas e ruins. Mesmo quando algo não ocorria como planejado sempre existia alguém ao nosso lado com quem tinhamos a certeza de que podiamos contar. Somente vocês sabem as dúvidas e as dificuldades das viagens semanais que enfrentamos sempre juntos, e que hoje, olhando para trás, temos a certeza de que tudo valeu apena! Obrigado meus "companheiros de batalha" por terem me dado a oportunidade de compartilhar com vocês esses momentos tão importantes de nossas vidas.

Enfim, agradeço a todos os que participaram direta ou indiretamente de mais esta conquista na minha vida. Obrigado!

A vida é uma peça de teatro que não permite ensaios. Por isso, cante, chore, dance, ria e viva intensamente, antes que a cortina se feche e a peça termine sem aplausos.

Charles Chaplin

Resumo

Este trabalho consiste em um estudo sobre os princípios básicos de codificação de fonte. Neste sentido desenvolvemos os conceitos básicos de fonte de dados, informação e entropia, requisitos para apresentar o Teorema da Codificação de Fonte de Shannon [1] como também o resultado que fornece os parâmetros e as limitações para a codificação de fonte sem perda de informação. O Código de Huffman [2] e [3], é discutido pela sua importância teórica e o trabalho é concluído com uma descrição do Algoritmo Lempel-Ziv [4] e [5].

Palavras-chave: Criptografia, Código de fonte, Código de Huffman, Algoritmo Lempel-Ziv.

Abstract

This work consists of a study about the basic principles of source coding. In this sense we develop the basics of data source, information and entropy, requirements to present the Shannon's Source Coding Theorem [1] as well as the result that provides the parameters and limitations for source coding without loss of information. The Huffman Code [2] and [3] is discussed for their theoretical importance and the work is concluded with a description of the Lempel-Ziv Algorithm [4] and [5].

Keywords: Cryptography, Source Code, Huffman Code, Lempel-Ziv Algorithm.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

Este trabalho concerne em desenvolver conceitos básicos sobre Teoria da Informação, tendo como fio condutor apresentar o Teorema de Codificação de Fonte de Shannon. Historicamente, o marco que estabeleceu a Teoria da Informação e chamou imediatamente a atenção mundial foi o artigo [1] escrito por Claude Shannon em 1948.

Neste artigo, Shannon apresenta pela primeira vez um modelo quantitativo e qualitativo da comunicação, vista como um processo estatístico subjacente à teoria da informação.

A Teoria da Informação desde a sua formulação e sistematização por Shannon [1] é responsável por uma parte significativa do perfil atual da nossa sociedade. O processo de comunicação envolve, entre outros, três aspectos essenciais que são: a compressão de dados, a codificação para a correção de erros de transmissão e a criptografia. O presente trabalho consiste no estudo e na descrição dos algoritmos de Huffman e Lempel-Ziv, fundamentais tanto para o estudo acadêmico como para implementações práticas de codificação de fontes de informação.

Este trabalho está organizado da seguinte maneira: após esta introdução no capítulo 2, apresentamos algumas das noções básicas de códigos e entropia como a construção de códigos instantâneos e livres de prefixo, as desigualdades de Kraft e McMillan que serão essenciais para a demonstração do primeiro teorema de Shannon e os conceitos de informação e entropia que são vitais para o entendimento da importância do código de Huffman; no capítulo 3, demonstraremos o Primeiro Teorema de Shannon e estudaremos os códigos de Huffman; e, por fim, no capítulo 4, descrevemos os algoritmos de Lempel-Ziv em suas duas versões originais (LZ77 e LZ78) e encerraremos demonstrando a otimalidade de ambos.

2 Noções sobre Códigos e Entropia

Neste capítulo descrevemos os objetos necessários para o entendimento deste material. Primeiramente apresentamos a simbologia utilizada.

2.1 Decodificação

Inicialmente é preciso fazer a distinção entre símbolos que são transmitidos, por exemplo, as letras do alfabeto, e símbolos que são utilizados pelo sistema de sinalização ou símbolos processados pelo codificador de fonte, os quais, em geral, são representados binariamente, ou seja, por "0"s e "1"s. Neste contexto, representamos os símbolos que são transmitidos pela sequência $a_1, a_2, ..., a_k$ enquanto que os símbolos do alfabeto do código são realizados por "0"e "1". Chamaremos de **código r-ário** o código cujo alfabeto possui r símbolos.

A primeira propriedade que precisamos garantir ao executar uma codificação é a **decodificação única**. Afinal, a mensagem recebida deve ter uma única interpretação possível.

Vejamos alguns exemplos.

Exemplo 2.1. Consideremos um código tal que o alfabeto da fonte $\mathcal{J}$ tenha 4 símbolos a serem codificados na forma binária, isto é,

$$\mathcal{J} = \{a_1, a_2, a_3, a_4\}$$

Suponha que o código $\mathcal{C}_1$, mostrado na Tabela 2.1, seja utilizado.

$\mathcal{J}$	$\mathcal{C}_1$
a_1	0
a_2	01
a_3	11
a_4	00

Tabela 2.1: Código para a fonte $\mathcal{J}$.

Suponha ainda que a mensagem recebida seja 0011, então temos que 0011 pode ser decodificado como a_4a_3 ou como $a_1a_1a_3$.

Para decodificação única a condição necessária é que não devem existir duas mensagens codificadas igualmente.

Para adequar este exemplo basta gerar uma sequência de símbolos da fonte de forma que a sequência codificada não resulte em mais de uma sequência de símbolos da fonte após o processo de decodificação.

Para fontes cuja cardinalidade do conjunto de símbolos é pequena, é possível realizar este teste heurístico. Já para fontes cuja cardinalidade é grande, a complexidade do teste para verificação se o código a ser utilizado é unicamente decodificável é enorme, podendo conduzir a resultados errôneos. Porém, existe um teste que sempre poderá ser utilizado para determinar se um código é unicamente decodificável ou não, esse teste tem sua garantia através do seguinte teorema.

Teorema 2.1. *Consideremos $\mathcal{C}_0$ um código e, para $n > 0$, definimos:*

$$\mathcal{C}_n = \{w \mid uw = v \text{ onde } u \in \mathcal{C}_0,\ v \in \mathcal{C}_{n-1} \text{ ou } u \in \mathcal{C}_{n-1},\ v \in \mathcal{C}_0\}$$

Assim, $\mathcal{C}_0$ é unicamente decodificável se, e somente se, nenhum dos conjuntos $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n, \ldots$ contém uma palavra-código que pertença ao conjunto $\mathcal{C}_0$.

Demonstração. Vide página 332, de [6].

O exemplo a seguir ilustra a utilização deste teorema.

Exemplo 2.2. Considere uma fonte $\mathcal{J}$ com alfabeto $\{a_1, a_2, ..., a_7\}$ tal que o código associado seja $\mathcal{C}_0 = \{a, c, ad, abb, bad, deb, bbcde\}$. Deseja-se determinar se $\mathcal{C}_0$ é unicamente decodificável. O procedimento a ser seguido é mostrado na Tabela 2.2.

$\mathcal{C}_0$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_6$	$\mathcal{C}_7$
a	d	eb	de	b	ad	d	eb
c	bb	cde			bcde		
ad							
abb							
bad							
deb							
bbcde							

Tabela 2.2: Procedimento de verificação de códigos unicamente decodificáveis.

Inicie a construção desta tabela alocando as palavras-código do código $\mathcal{C}_0$ na forma de coluna. As colunas $\mathcal{C}_i$'s são preenchidas utilizando-se do seguinte procedimento:

selecione uma palavra-código de $\mathcal{C}_0$, por exemplo, a palavra-código **a**. Esta palavra-código é denominada de **prefixo**. Identifique nas demais palavras-código de $\mathcal{C}_0$ as palavras-código que tenham **a** como prefixo. Os **sufixos** das correspondentes palavras-código são alocadas na coluna $\mathcal{C}_1$. Neste caso, **a** é **prefixo** das palavras-código **ad** e **abb**. Portanto, os **sufixos d** e **bb** são alocados em $\mathcal{C}_1$. Como **a** é a única palavra-código em $\mathcal{C}_0$ que é prefixo, segue que a coluna de $\mathcal{C}_1$ está completa. Verifique se alguma palavra-código em $\mathcal{C}_1$ é também uma palavra-código de $\mathcal{C}_0$ em caso positivo, temos que $\mathcal{C}_0$ não é unicamente decodificável em caso negativo, continue o procedimento, porém considerando os códigos $\mathcal{C}_0$ e $\mathcal{C}_1$ na montagem do código $\mathcal{C}_2$.

Agora consideramos a construção da coluna $\mathcal{C}_2$. A cada palavra-código de $\mathcal{C}_0$ considerada como prefixo, identifique em $\mathcal{C}_1$ quais palavras-código possuem a palavra-código de $\mathcal{C}_0$ como prefixo. Os correspondentes sufixos devem ser colocados em $\mathcal{C}_2$. Repita o procedimento considerando agora as palavras-código de $\mathcal{C}_1$ que são prefixo das palavras-código de $\mathcal{C}_0$. Os correspondentes sufixos deverão ser alocados em $\mathcal{C}_2$. Completando este procedimento, $\mathcal{C}_2$ está determinado. Verifique se alguma palavra-código de $\mathcal{C}_2$ é também uma palavra-código de $\mathcal{C}_0$. Em caso positivo, segue que $\mathcal{C}_0$ não é unicamente decodificável. Em caso negativo, continue o procedimento, porém considerando os códigos $\mathcal{C}_0$ e $\mathcal{C}_2$ na montagem do código $\mathcal{C}_3$.

Esses passos devem ser seguidos até que se obtenha uma das seguintes possibilidades:

1. Existe uma palavra-código comum tanto a $\mathcal{C}_0$ quanto a $\mathcal{C}_i$;

2. A condição anterior não é verificada e que a partir de um determinado valor de i, digamos i=n, $\mathcal{C}_n$ é vazio.

Nesse exemplo, note que $\mathcal{C}_n$ é vazio para $n > 7$.

Como em $\mathcal{C}_5$ temos uma palavra-código de $\mathcal{C}_0$ então este código não é unicamente decodificável.

Note que poderia ocorrer o fato de se argumentar que uma vez que no conjunto $\mathcal{C}_0$ existe uma palavra-código que é prefixo de uma outra, então o código não é unicamente decodificável, sem necessitar dessa forma do emprego do teorema. Este raciocínio é falso, pois o fato de uma palavra-código ser prefixo de uma outra somente implica que este código não é instantâneo.

Um contra-exemplo a este argumento é suficiente para caracterizar a importância do teorema.

Exemplo 2.3. Considere uma fonte $\mathcal{J}$ com alfabeto $\{a_1, a_2, a_3, a_4\}$ e cujo código associado é $\mathcal{C}_0 = \{10, 100, 1000, 10000\}$. Note que este código contém várias palavras-código que são prefixo de outras. Pelo argumento apresentado no parágrafo anterior, este código não é unicamente decodificável. Esta conclusão é falsa, como veremos a seguir.

Considerando o Teorema 2.1, apresentamos na Tabela 2.3 a verificação de que este código é unicamente decodificável.

$\mathcal{C}_0$	$\mathcal{C}_1$
10	0
100	00
1000	000
10000	

Tabela 2.3: Verificação se o código $\mathcal{C}_0$ é unicamente decodificável.

Como os $\mathcal{C}_n$, para $n > 2$, são conjuntos vazios e em $\mathcal{C}_1$ os sufixos (palavras-código) não pertencem ao código $\mathcal{C}_0$, então o código é unicamente decodificável. Todavia, se ao código $\mathcal{C}_0$ for incluída a palavra-código 0, então o código $\mathcal{C}_0 = \{0, 10, 100, 1000, 10000\}$ não será unicamente decodificável. Se ao invés do 0, contiver a palavra-código 1, então o código $\mathcal{C}_0 = \{1, 10, 100, 1000, 10000\}$ é unicamente decodificável.

2.2 Códigos Instantâneos

No que segue definimos como

- **Árvore de decisão:** um grafo cujos nós representam palavras-código de um código r-ário.
- **Ramos:** os caminhos de uma árvore de decisão que ligam dois, e apenas dois, nós.
- **Nó raiz:** o nó do topo da árvore, do qual descendem os demais nós. É o primeiro nó da árvore.
- **Nós intermediários:** nós do interior da árvore (que possuem descendentes).
- **Nós terminais:** nós que não possuem descendentes.

Agora tratamos brevemente dos Códigos Instantâneos (ou Códigos Decodificáveis Instantaneamente). Para tanto, considere o código $\mathcal{C}_0$ de comprimento variável associado a fonte $\mathcal{J}$ esta com alfabeto $\{a_1, a_2, a_3, a_4\}$, como mostrado na Tabela 2.4.

Suponha que a fonte $\mathcal{J}$ envie uma sequência de símbolos em que cada símbolo é codificado segundo a correspondente palavra-código do código $\mathcal{C}_0$. Vejamos qual deve ser a estratégia a ser usada de tal forma que da sequência de palavras-código recebida o decodificador decodificará corretamente na sequência de símbolos emitidos pela fonte.

$\mathcal{J}$	$\mathcal{C}_0$
a_1	0
a_2	10
a_3	110
a_4	111

Tabela 2.4: Código $\mathcal{C}_0$ de comprimento variável associado a fonte $\mathcal{J}$.

A estratégia é utilizar a árvore de decisão no processo de codificação, como mostrado na Fig. 2.1.

Iniciando o processo de decodificação na raíz desta árvore, o primeiro bit recebido seguirá o ramo em direção ao "0"se este bit for "0". Por outro lado, irá para o nó intermediário **a** se o bit for "1". Em seguida irá para o "10"se o segundo bit recebido for "0"ou para o nó intermediário **b** se for "1". Finalmente, irá para o nó terminal "110"se "0"for recebido ou para "111"se "1"for recebido.

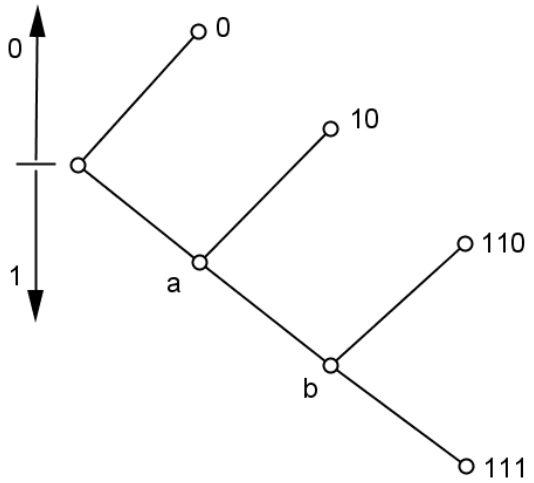


Figura 2.1: Árvore de decisão.

Uma vez que a codificação de uma mensagem atinge um nó terminal, o decodificador volta para a raiz da árvore de decisão. É importante frisar que cada bit recebido, da cadeia de bits, é **examinado somente uma única vez** e que os nós terminais desta árvore compõe as quatro palavras-código 0, 10, 110, 111, associadas aos correspondentes símbolos da fonte.

Neste exemplo, a **decodificação é instantânea** pois quando uma sequência de símbolos do alfabeto do código é recebida, o receptor reconhece imediatamente cada palavra-código nesta sequência recebida. Portanto, não é preciso continuar o processo de busca antes de decidir que símbolo da fonte foi transmitido.

Definição 2.1. *Um código é instantâneo se, e somente se, nenhum símbolo codificado deste código é prefixo de qualquer outro símbolo.*

Exemplo 2.4. O código 0, 01, 101 não é instantâneo pois "0"é prefixo de "01", isto é, o decodificador tem que esperar pelo próximo dígito para determinar que mensagem foi recebida. Este código não é unicamente decodificável pois 0101 pode ser decodificado como 0, 101 ou 01, 01.

Exemplo 2.5. Considere o código 0, 01, 011, 111. Note que este código possui palavras com o mesmo prefixo que outras palavras-código do código, portanto, não é instantâneo, mas é unicamente decodificável. Para isso, considere a sequência

$$0\ 1\ 1\ 1\ .....\ \overbrace{1\ 1\ 1}^{a_4}\ \overbrace{1\ 1\ 1}^{a_4}$$

Nesse exemplo vemos que o decodificador tem que esperar pelo recebimento de toda a cadeia de bits transmitidos para poder fazer a decodificação. Caso contrário, nenhuma conclusão pode ser realizada.

2.3 Construção de Códigos Instantâneos

Deve estar claro que dentre todos os códigos **unicamente decodificáveis**, os códigos **instantâneos** tem preferência devido ao fato de que não existe nenhuma complexidade adicional para a sua obtenção.

Considere o código $\mathcal{C}_0 = 0, 10, 110, 111$. A correspondente árvore de decisão é mostrada na Fig. 2.2. Note que as palavras-código do código $\mathcal{C}_0$ estão associadas aos nós terminais da árvore de decisão.

Portanto, as palavras-código de um código instantâneo devem ser identificadas como sendo os correspondentes caminhos do nó raiz até os nós terminais de uma árvore. Consequentemente, dada uma árvore, se na ramificação da árvore de decisão binária nenhum nó terminal é deixado em aberto, significa que o código é instantâneo e ótimo, caso fique um ramo em aberto o código é instantâneo porém não ótimo.

2.4 Desigualdade de Kraft

Nas codificações anteriores a construção de códigos instantâneos foi tratada, de certa forma, heuristicamente. Todavia, existe um resultado devido a Kraft, que constrói um código instantâneo. Mais especificamente, este resultado exibe uma desigualdade que estabelece a condição de existência de códigos instantâneos para um dado conjunto de valores, esta desigualdade ocorre sob os **comprimentos** das palavras-código; esta desigualdade é conhecida como desigualdade de Kraft.

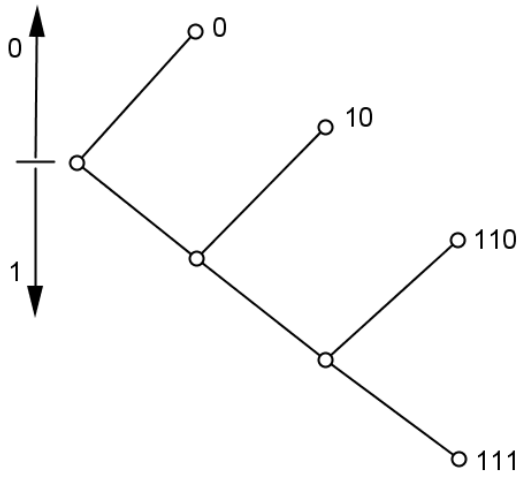


Figura 2.2: Árvore de decisão do código instantâneo.

Teorema 2.2. *(Desigualdade de Kraft) A condição necessária e suficiente para a existência de um código instantâneo $\mathcal{C}$ composto das palavras-código c_i, para $i = 1, 2, ..., J$, com respectivos comprimentos $l_1 \leq l_2 \leq ... \leq l_J$ é que*

$$\sum_{i=1}^{J} \left(\frac{1}{r}\right)^{l_i} \leq 1,$$

em que r é a cardinalidade do alfabeto do código.

Demonstração. Na demonstração deste teorema, fazemos uso do fato de que em uma árvore r-ária completa de comprimento (profundidade) N, existem r^{N-l} nós terminais que emanam de cada nó à profundidade l, em que $l < N$.

Suponha primeiramente que exista um código r-ário instantâneo cujas palavras-código tenham comprimento $l_1, l_2, ..., l_J$. Seja $N = max_i\ l_i$, e considere a construção de um grafo para este código através de podas a serem realizadas na árvore completa r-ária de comprimento N. Começando com $i = 1$, encontramos um nó z_i nesta árvore e, se $l_i < N$, então iremos podar esta árvore de modo a tornar este nó um nó terminal à profundidade l_i. Por este procedimento, estamos eliminando r^{N-l_i} nós terminais da árvore completa que poderiam ser utilizados para identificar outras palavras-código, uma vez que nenhum destes nós poderiam ser utilizados como palavras-código dada a condição do código ser instantâneo. Como existem r^N nós terminais que podem ser eliminados, deveremos encontrar, após eliminarmos todas as outras possibilidades de palavras-código, que

$$r^{N-l_1} + r^{N-l_2} + ... + r^{N-l_J} \leq r^N \tag{2.1}$$

Dividindo por r^N, conduz à desigualdade 2.1 como a condição necessária para o código ser instantâneo.

Reciprocamente, suponha que $l_1 \leq l_2 \leq \ldots \leq l_J$. Considere então o seguinte algoritmo: Seja $N = max_i\ l_i$ e que comece com a árvore r-ária completa de comprimento N.

1. $i \to 1$;

2. Escolha z_i como sendo qualquer nó sobrevivente à profundidade l_i (que não foi usado como uma palavra-código) e, se $l_i < N$, pode a árvore em z_i. Pare se não existir um nó sobrevivente;

3. Se $i = J$, pare. Caso contrário, faça $i \leftarrow i + 1$ e vá para o Passo 2.

Se formos capazes de escolher z_J no Passo 2, então teremos construído um código r-ário instantâneo satisfazendo os comprimentos das palavras-código. Mostraremos que de fato podemos escolher z_i no Passo 2 para todo $i \leq J$. Suponha que $z_1, z_2, \ldots, z_{i-1}$ tenham sido escolhidos. O número de nós sobreviventes à profundidade N que não emanam de qualquer palavra-código é

$$r^N - \left(r^{N-l_1} + r^{N-l_2} + \ldots + r^{N-l_{i-1}}\right) = r^N - \left(1 - \sum_{j=1}^{i-1} r^{-l_j}\right).$$

Assim, se $i < J$, a condição 2.1 mostra que o número de nós sobreviventes à profundidade N é maior do que zero. Mas se existem nós sobreviventes à profundidade N, então também deve existir (não utilizados) nós sobreviventes à profundidade $l_i < N$. Como $l_1 \leq l_2 \leq \ldots \leq l_{i-1} \leq l_i$, nenhuma palavra-código já escolhida pode emanar deste nó sobrevivente e então ela poderá ser escolhida como z_i. Desse modo, a condição 2.1 é suficiente para a construção de um código r-ária instantâneo com os comprimentos das palavras-código especificados. □

Observação 2.1. Note que a demonstração da desigualdade de Kraft é construtiva. Observe que esta construção baseia-se em um algoritmo para a construção de um código r-ário instantâneo sabendo-se os comprimentos das palavras-código $l_1, l_2, \ldots, l_J$ sempre que este código existir, isto é, sempre que a condição 2.1 seja satisfeita. Não é necessário iniciar o processo a partir da árvore completa de comprimento $max_i\ l_i$, uma vez que o algoritmo oriundo da demonstração do teorema, é completamente equivalente à geração da árvore a partir do nó raiz, já que seleciona qualquer nó à profundidade l_i para a palavra-código z_i com a condição de que as palavras-código de menores comprimentos tenham sido selecionadas anteriormente.

Exemplo 2.6. Vejamos como construir um código binário livre de prefixo cujas palavras-código tem comprimentos $l_1 = 2, l_2 = 2, l_3 = 2, l_4 = 3, l_5 = 4$.

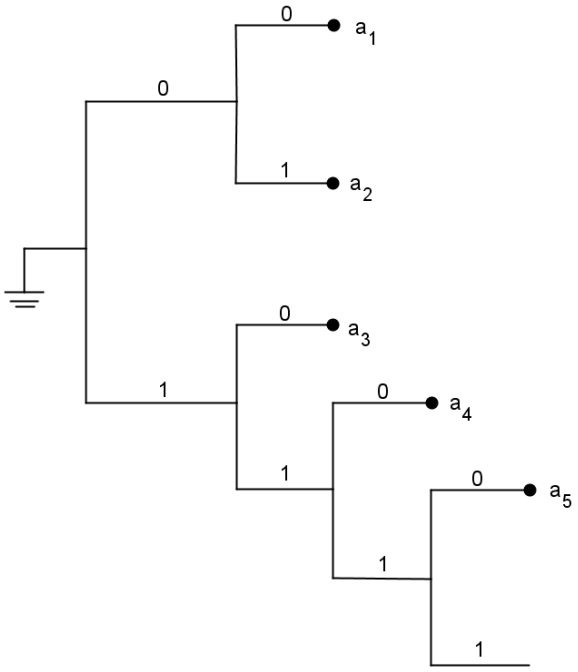


Figura 2.3: Código binário livre de prefixo

U	Z
a_1	00
a_2	01
a_3	10
a_4	110
a_5	1110

Tabela 2.5: Código binário livre de prefixo.

Uma vez que $\sum_{i=1}^{5} 2^{-l_i} = \frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} = \frac{15}{16} < 1$, sabemos que existe o código binário livre de prefixo. A árvore associada a este código é mostrada na Fig. 2.3. As correspondentes palavras-código são mostradas na Tabela 2.5.

Exemplo 2.7. Seja agora um código binário livre de prefixo cujas palavras-código tem comprimentos $l_1 = 1, l_2 = 2, l_3 = 3, l_4 = 4, l_5 = 5$.

Pelo teorema anterior não é possível construir um código-livre, uma vez que $\sum_{i=1}^{5} 2^{-l_i} = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} = \frac{19}{16} > 1$.

Ou seja, este não satisfaz a desigualdade de Kraft.

Exemplo 2.8. Encontremos um código instantâneo com comprimentos 1, 2, 2, 2, 2, 2, 3, 3, 3, 3 com $r = 3$. Na desigualdade de Kraft temos que

$$\frac{1}{3} + 5\frac{1}{9} + 4\frac{1}{27} = \frac{28}{27} > 1.$$

Portanto, é impossível determinar um código instantâneo com esses comprimentos. Por outro lado, se usarmos os seguintes comprimentos $1, 2, 2, 2, 2, 2, 3, 3, 3$. Temos que

$$\frac{1}{3} + 5\frac{1}{9} + 3\frac{1}{27} = \frac{27}{27} = 1.$$

Consequentemente, é possível determinar tal código. A árvore de decisão correspondente é como mostra a Fig. 2.4.

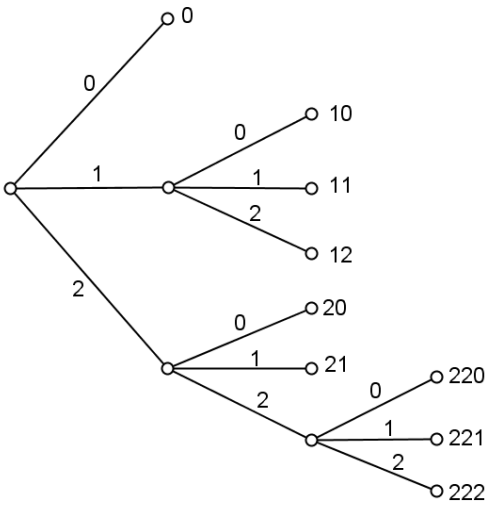


Figura 2.4: Árvore de decisão ternária

2.5 Códigos de Bloco Reduzido

Vejamos nessa seção que é possível reduzir palavras de um código. Para tanto, suponha que um código de bloco tenha exatamente 2^m palavras. Deste modo, poderíamos usar m dígitos para representar cada símbolo. Por outro lado, suponha que o número de mensagens não seja um submúltiplo de 2^m, por exemplo, $2^m - 1$. Como poderíamos a partir do código de bloco com 2^m palavras construir um código de comprimento variável com um número de palavras menor que 2^m?

Nesta direção, considere o caso em que a fonte contém 5 mensagens. Seja $m = 3$. Neste caso, o código tem 8 palavras dadas por 000, 001, 010, 011, 100, 101, 110, 111. Suponha que descartemos deste código as seguintes palavras 001, 011 e 101. Com isso reduzimos de 3 ramos a árvore de decisão, porém, mantivemos a condição de decodificação instantânea, como mostra a Fig.2.5.

Como outro exemplo, considere o caso onde eliminamos as palavras 001, 010 e 011 do código original. A nova árvore de decisão é como mostra a Fig. 2.6.

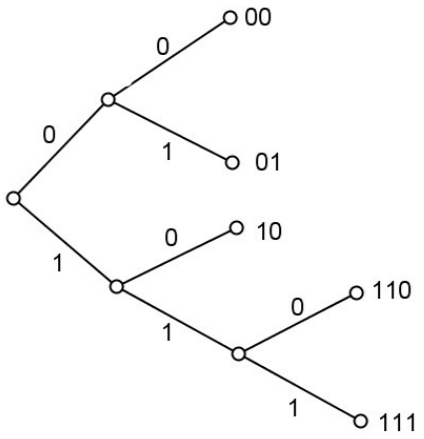


Figura 2.5: Árvore de decisão do código reduzido (eliminando as palavras 001, 011 e 101)

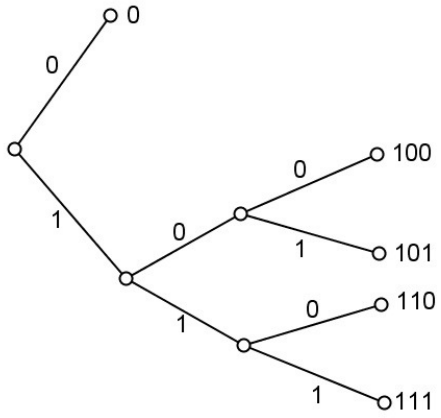


Figura 2.6: Árvore de decisão do código reduzido (eliminando as palavras 001, 010 e 011)

2.6 Desigualdade de McMillan

A desigualdade de Kraft se aplica a códigos instantâneos. Tais códigos formam uma classe especial de códigos unicamente decodificáveis. Por outro lado, McMillan mostrou que a desigualdade de Kraft continua sendo válida para códigos unicamente decodificáveis. Ou seja, ele provou o seguinte teorema.

Teorema 2.3. *Se um código unicamente decodificável tem palavras-código de comprimento* $l_1, l_2, \ldots, l_J$*, então*

$$\sum_{i=1}^{J} \left(\frac{1}{r}\right)^{l_i} \leq 1.$$

Das considerações realizadas até o presente momento com relação a existência e construção de códigos instantâneos apenas consideramos os parâmetros **comprimento**

das palavras-código e a **cardinalidade do alfabeto**. Todavia, as fontes de informação são inerentemente aleatórias implicando que as mensagens emitidas ocorrem segundo uma caracterização estatística da fonte. No caso de uma fonte discreta esta caracterização estatística vem com a probabilidade de ocorrência da mensagem. Diante disto, fica claro que temos que considerar as probabilidades de ocorrência das palavras-código nos códigos que foram construídos.

A introdução das probabilidades de ocorrência das mensagens e consequentemente das palavras-código tem como objetivo principal codificar as mensagens da fonte $\mathcal{J}$ de modo a minimizar o comprimento médio das palavras-código, denotado por $E[L]$ ou $L_{\text{médio}}$. Dos Exemplos 2.6 e 2.8 fica claro que, de uma forma intuitiva, deveríamos associar as palavras-código de menor comprimento às mensagens mais prováveis da fonte $\mathcal{J}$. Porém, quais devem ser os comprimentos das palavras-código? Ou ainda, qual é o menor valor de $E[L]$ que pode ser alcançado? Estas questões são tratadas mais adiante.

Na próxima seção iremos considerar os conceitos de informação e de entropia, elementos fundamentais para o estabelecimento dos resultados sobre codificação de fontes discretas.

2.7 Conceitos de Informação e Entropia

Um método geral para a descrição da fonte necessariamente utiliza do conceito de **entropia**. Entropia é um conceito bastante utilizado em Física Quântica e Termodinâmica. Basicamente este conceito fornece o grau de liberdade associado a estrutura e a organização do sistema. Neste contexto, a entropia é descrita como sendo uma simples função de distribuição de probabilidades p_i.

A caracterização de uma fonte discreta consiste de um conjunto finito de mensagens, denotado por $\mathcal{J} = \{a_1, a_2, \ldots, a_J\}$ e de um conjunto de probabilidades $\mathbb{P} = \{p(a_1), p(a_2), \ldots, p(a_J)\}$, onde $p(a_i)$ denota a probabilidade de ocorrência do símbolo a_i. Dessa forma, dizemos que a fonte discreta $\mathcal{J}$ é caracterizada por ($\mathcal{J}$,$\mathbb{P}$). O grau de liberdade ou de estruturação desta fonte é dado pelo conceito de entropia. Mais especificamente, este número estabelece a quantidade mínima de dígitos que deverão ser utilizados para caracterizar de maneira biunívoca cada símbolo com a respectiva *representação digital*.

Caracterizamos a seguir a função que permite tratar da quantidade de informação. A saber, sabe-se que a saída de uma fonte discreta é uma sequência aleatória de símbolos de um alfabeto com J valores, isto é, $\{a_1, a_2, \ldots, a_J\}$. Esta sequência de símbolos é produzida segundo uma regra ou lei de probabilidade.

Se a fonte for determinística, a incerteza associada é zero, portanto sem interesse algum em termos da transmissão da informação associada. Logo, este tipo de fonte não será considerada.

Se a saída da fonte a_j ocorrer com probabilidade $p(a_j) = p_j$, então definimos a quantidade de informação associada como sendo a função

$$I(a_j) = -\log p(a_j) = -\log p_j = \log \frac{1}{p_j}.$$

Se a base do logaritmo for 2 então a unidade associada é o **bit** (binary digit). Se a base do logaritmo for e, então a unidade associada será o **nat** (natural unit). O **nat** é mais fácil de se tratar matematicamente pois as operações com derivadas são menos complexas.

Se para uma fonte discreta sem memória (o símbolo emitido pela fonte no instante k não depende do símbolo que foi emitido no instante $k-1$) a probabilidade de ocorrer o símbolo a_j é $p(a_j)$, consequentemente a informação gerada é $I(a_j) = -\log p(a_j)$.

Por outro lado, considere uma sequência muito longa, digamos $N >> 1$, de símbolos emitidos pela fonte. A frequência relativa, denotada por f_j, do símbolo a_j é dada pela razão do número de vezes que o símbolo a_j ocorre nesta sequência, denotado por $n(a_j)$, pelo comprimento da sequência, assumido N, isto é, $f_j = n(a_j)/N$. Sabe-se que para N tendendo a infinito, f_j tende a probabilidade de ocorrência do símbolo a_j, isto é,

$$\lim_{N\to\infty} f_j = p(a_j).$$

Portanto, $p(a_j) = n(a_j)/N$.

Desta equação podemos fazer a interpretação de que o número de vezes que o símbolo a_j ocorreu na sequência (de comprimento N) de símbolos emitidos pela fonte é $n(a_j) = Np(a_j)$. Por outro lado, cada vez que o símbolo a_j ocorre, a quantidade de informação associada em **bits** vale $I(a_j) = -\log p(a_j)$. Consequentemente, como o símbolo a_j ocorreu $n(a_j)$ vezes, a quantidade total de informação associada vale $I_{total}(a_j) = -Np(a_j)\log p(a_j)$, onde $j = 1, 2, 3, \ldots, J$.

Dessa forma, a quantidade de informação associada à sequência emitida pela fonte é dada pela soma das quantidades de informação $I_{total}(a_j)$, $j = 1, 2, \ldots, J$, isto é,

$$Np(a_1)\log\frac{1}{p(a_1)} + Np(a_2)\log\frac{1}{p(a_2)} + \ldots + Np(a_J)\log\frac{1}{p(a_J)} \quad bits$$

Se dividirmos esta expressão por N, temos a quantidade média de informação gerada pela fonte a cada símbolo. Esta é a **entropia** (ou incerteza). Ou seja,

$$H(p_1, p_2, \ldots, p_J) = H(\mathbf{p}) = -\sum_{j=1}^{J} p(a_j)\log p(a_j) = -\sum_{j=1}^{J} p_j \log p_j, \quad bits/smbolo$$

Também usamos a notação de que a fonte é uma variável aleatória X. Quando esta variável aleatória assume o valor a_j, isto é, $X = a_j$ com probabilidade $p(a_j)$, então a notação $H(X)$ é considerada, $(H(X) = H(\mathbf{p}))$.

Vejamos agora algumas propriedades da Função Entropia.

Proposição 2.1. *$H(\mathbf{p})$ é uma função contínua em $\mathbf{p}$. Note que pequenas variações em $\mathbf{p}$, implicam em pequenas variações em $H(\mathbf{p})$.*

Demonstração. O resultado segue pois H é dada em termos de funções contínua. □

Proposição 2.2. *1. $H(\mathbf{p}) \geq 0$;*

2. $H(\mathbf{p}) = 0$ se, e somente se, $\mathbf{p}$ é tal que um $p_j \neq 0$ e os demais são zeros.

Demonstração. Este fato segue pela própria definição de H. □

Proposição 2.3. *Para uma dada fonte discreta $\mathcal{J}$, temos que $H(\mathbf{p}) \leq \log J$ com igualdade se, e somente se, $p_j = 1/J$.*

Demonstração. Esta propriedade é demonstrada usando a desigualdade $\log x \leq x - 1$, isto é,

$$H(\mathbf{p}) - \log J = -\sum_{j=1}^{J} p_j \log p_j - \log J = -\sum_{j=1}^{J} p_j \log p_j - \sum_{j=1}^{J} p_j \log J = -\sum_{j=1}^{J} p_j \log Jp_j$$

$$= \sum_{j=1}^{J} p_j \log \frac{1}{Jp_j} \leq \sum_{j=1}^{J} p_j \left(\frac{1}{Jp_j} - 1 \right) = -\sum_{j=1}^{J} \left(\frac{1}{J} - p_j \right) = 1 - 1 = 0$$

Logo, $H(\mathbf{p}) - \log J \leq 0$, e portanto, $H(\mathbf{p}) \leq \log J$. □

Proposição 2.4. *A função $H(\mathbf{p})$ é côncava, isto é,*

$$H(\lambda \mathbf{p}' + \bar{\lambda}\mathbf{p}'') \geq \lambda H(\mathbf{p}') + \bar{\lambda} H(\mathbf{p}''), \quad \textit{para} \quad 0 \leq \lambda \leq 1.$$

Demonstração. Esta propriedade é provada através do uso da desigualdade $\log x \geq 1 - 1/x$.
Seja $\bar{\lambda} = 1 - \lambda$, onde $0 \leq \lambda \leq 1$. Então,

$$H(\lambda \mathbf{p}' + \bar{\lambda}\mathbf{p}'') - \lambda H(\mathbf{p}') - \bar{\lambda} H(\mathbf{p}'') = \lambda \sum_{j=1}^{J} p'_j \log \frac{p'_j}{\lambda p'_j + \bar{\lambda} p''_j} + \bar{\lambda} \sum_{j=1}^{J} p''_j \log \frac{p''_j}{\lambda p'_j + \bar{\lambda} p''_j}$$

usando da desigualdade temos

$$\geq \lambda \sum_{j=1}^{J} p'_j \left(1 - \frac{\lambda p'_j + \bar{\lambda} p''_j}{p'_j} \right) + \bar{\lambda} \sum_{j=1}^{J} p''_j \left(1 - \frac{\lambda p'_j + \bar{\lambda} p''_j}{p''_j} \right) = \lambda(1-1) + \bar{\lambda}(1-1) = 0,$$

logo,

$$H(\lambda \mathbf{p}' + \bar{\lambda}\mathbf{p}'') \geq \lambda H(\mathbf{p}') + \bar{\lambda} H(\mathbf{p}'').$$

□

Proposição 2.5. *A entropia de um par de variáveis aleatórias independentes é igual a soma de cada uma das entropias. A entropia associada a n variáveis aleatórias independentes é igual a soma das n entropias.*

Observação 2.2. Se uma fonte binária contém somente dois símbolos, digamos a_1 e a_2 com probabilidades p e $1-p$, respectivamente. A entropia de uma fonte binária é descrita pela **função entropia binária** como

$$H_b(p) = -p\log p - (1-p)\log(1-p).$$

Note que $H_b(p)$ é uma função de uma única variável, p. O gráfico de $H_b(p)$ é mostrado na Fig. 2.7

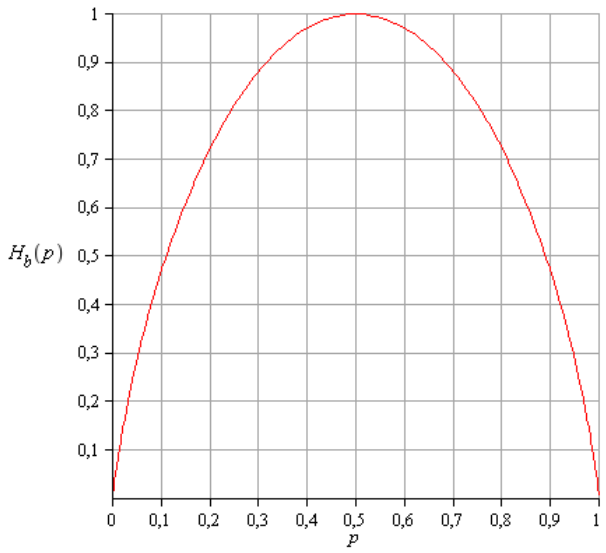


Figura 2.7: Função entropia binária

Chamamos a atenção para a diferença entre **dígito binário** que é a **saída de uma fonte binária** e **bit** usados como **medida de informação**. A fonte binária fornece **1 bit de informação** para cada escolha de símbolo somente quando os dois símbolos são equiprováveis.

Veja ainda que uma variável aleatória pode ser condicionada em uma informação extra. Denotaremos a distribuição de probabilidade condicional $p(b_k/a_j)$ por $p(y/x)$ ou $p(a_j/b_k)$ por $p(x/y)$, onde $x \in \{a_1, a_2, \ldots, a_J\}$ e $y \in \{b_1, b_2, \ldots, b_K\}$.

Se x (ou y) é fixado, então a entropia condicionada em x (ou em y) é dada por

$$H(Y/X=x) = -\sum_y p(y/x)\log p(y/x),$$

ou

$$H(X/Y=y) = -\sum_x p(x/y)\log p(x/y),$$

onde pela fórmula de Bayes temos,

$$p(y/x) = \frac{p(x,y)}{p(x)} = \frac{p(x/y)p(y)}{\sum_y p(x/y)p(y)}.$$

Se quisermos saber o valor médio da variável aleatória $H(Y/X=x)$ basta considerar

$$H(Y/X)=\sum_x p(x)H(Y/X=x)=-\sum_x p(x)\sum_y p(y/x)\log p(y/x).$$

O mesmo ocorre com o valor médio da variável aleatória $H(X/Y=y)$, isto é,

$$H(X/Y)=\sum_y p(y)H(X/Y=y)=-\sum_y p(y)\sum_x p(x/y)\log p(x/y).$$

Teorema 2.4. *A informação extra nunca aumenta a incerteza. Isto é,*

$$H(X/Y)\leq H(X).$$

Demonstração. Sabemos que

$$H(X)-H(X/Y)=-\sum_x p(x)\log p(x)+\sum_y p(y)\sum_x p(x/y)\log p(x/y).$$

Agora, uma vez que $p(x)=\sum_y p(y)p(x/y)$, então

$$H(X)-H(X/Y)=\sum_x\sum_y p(y)p(x/y)\log p^{-1}(x)+\sum_x\sum_y p(y)p(x/y)\log p(x/y)$$

$$=\sum_x\sum_y p(y)p(x/y)\log\frac{p(x/y)}{p(x)}=\sum_x\sum_y p(y)p(x/y)\log\frac{p(x/y)}{\sum_y p(y)p(x/y)},$$

como $\log x\geq 1-\frac{1}{x}$, então

$$H(X)-H(X/Y)\geq\sum_x\sum_y p(y)p(x/y)\left(1-\frac{\sum_y p(y)p(x/y)}{p(x/y)}\right)$$

$$=\sum_x\sum_y p(y)p(x/y)-\left(\sum_y p(y)\right)\left(\sum_x\sum_y p(y)p(x/y)\right)=1-1=0.$$

Logo,

$$H(X/Y)\leq H(X).$$

□

Teorema 2.5. *A entropia é aditiva para eventos aleatórios independentes.*

Demonstração. Temos que,

$$H(X_1,X_2,\ldots,X_n)=-\sum_{x_1}\sum_{x_2}\cdots\sum_{x_n}p(x_1,x_2,\ldots,x_n)\log p(x_1,x_2,\ldots,x_n)\qquad(2.2)$$

Como

$$p(x_1,x_2,\ldots,x_n)=\prod_{i=1}^{n}p(x_i),$$

então

$$\log p(x_1, x_2, \ldots, x_n) = \log \prod_{i=1}^{n} p(x_i) = \sum_{i=1}^{n} \log p(x_i). \tag{2.3}$$

Substituindo 2.3 em 2.2 temos,

$$= -\sum_{x_1}\sum_{x_2}\cdots\sum_{x_n} p(x_1, x_2, \ldots, x_n)\left[\sum_{i=1}^{n} \log p(x_i)\right]$$

$$= -\sum_{x_1}\sum_{x_2}\cdots\sum_{x_n} p(x_1)p(x_2)\ldots p(x_n)\left[\sum_{i=1}^{n} \log p(x_i)\right].$$

para cada x_i temos,

$$H(X_i) = -\sum_{x_i} p(x_i)\log p(x_i).$$

Como existem n termos, segue que

$$H(X_1, X_2, \ldots, X_n) = \sum_{i=1}^{n} H(X_i),$$

isto conclui a prova do teorema e da Proposição 2.5. □

2.8 Códigos Livres de Prefixo

Para o que segue definimos:

Definição 2.2. *Uma árvore r-ária é uma árvore tal que r ramos ou nenhum ramo derivam de cada nó.*

Definição 2.3. *Um árvore r-ária completa de comprimento N é uma árvore r-ária cujos nós terminais são os r^N nós à profundidade N do nó raiz.*

Considere o esquema de codificação como mostrado na Fig. 2.8.

Vamos supor que $\mathcal{J}$ é uma variável aleatória com alfabeto $\{a_1, a_2, \ldots, a_J\}$; cada X_i assume valores em um alfabeto r-ário, denotado por $\{0, 1, \ldots, r-1\}$; e ainda que L é uma variável aleatória, isto é, os valores da variável aleatória Z são sequências r-árias de comprimento variável.

Adotamos o menor valor possível sob $E[L]$, isto é, $min\{E[L]\}$, onde $E[L]$ denota o comprimento médio das palavras-código. Se $z_i = [x_{i1}, x_{i2}, \ldots, x_{il_i}]$ é a palavra-código para a_i e l_i é o comprimento desta palavra-código, então o comprimento médio das palavras-código é dado por

$$E[L] = \sum_{i=1}^{J} l_i P_{\mathcal{J}}(a_i),$$

uma vez que podemos pensar que L é uma função de valor real da variável aleatória $\mathcal{J}$. Note que $X_1, X_2, \ldots, X_L$ são, em geral, variáveis aleatórias definidas condicionalmente pois X_i assume valores somente quando $L \geq i$.

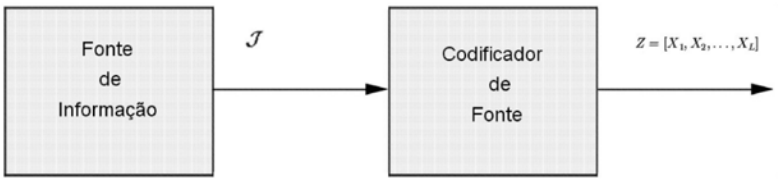


Figura 2.8: Esquema de codificação de comprimento variável

A seguir, estabelecemos os requisitos que um esquema de codificação para $\mathcal{J}$ tem que satisfazer:

1. Nenhuma palavra-código pode ser igual a outra palavra-código, isto é, $z_i \neq z_j$ para $i \neq j$. Este requisito assegura que $H(\mathcal{J}) = H(Z)$;

2. Nenhuma palavra-código é prefixo de uma outra palavra-código de comprimento maior. Este requisito assegura que uma palavra-código pode ser identificada logo após o recebimento de seu último digito, mesmo que o codificador seja utilizado ininterruptamente no envio de mensagens.

Um código para $\mathcal{J}$ que satisfaz as condições 1. e 2. é chamado um **código livre de prefixo** ou um **código que é decodificável instantaneamente**. Note que se usarmos a convenção que uma sequência é prefixo dela mesma, poderíamos estabelecer 1. e 2. juntamente com a condição que nenhuma palavra-código seja prefixo de uma outra palavra-código.

Exemplo 2.9. Considere $\mathcal{J}$ e Z como mostrados na Tabela 2.6.

$\mathcal{J}$	Z
a_1	0
a_2	10
a_3	11

Tabela 2.6: Código livre de prefixo.

Exemplo 2.10. Considere $\mathcal{J}$ e Z como mostrados na Tabela 2.7.

$\mathcal{J}$	Z
a_1	1
a_2	00
a_3	11

Tabela 2.7: Código que não é livre de prefixo.

De modo a dar um significado à natureza dos códigos livres de prefixo, em geral se associa os dígitos das palavras-código aos ramos de uma árvore. A Fig. 2.9 mostra as árvores binárias correspondentes aos códigos dos Exemplos 2.9 e 2.10. Note que as palavras-código do código livre de prefixo estão identificadas com os nós terminais da árvore, estes marcados com círculos pretos, enquanto que a do código que não é livre de prefixo, uma das palavras-código, círculo preto, é um nó intermediário da árvore binária correspondente.

Figura 2.9: a) Árvore binária livre de prefixo do Exemplo 2.9; b) Árvore binária que não é livre de prefixo do Exemplo 2.10.

Estabelecemos a seguir algumas definições no sentido de tornar mais precisas as considerações anteriores.

A Fig. 2.10 ilustra exemplos de duas árvores r-árias, a primeira binária e a segunda ternária completa.

Observe que um código livre de prefixo r-ário pode ser identificado com um conjunto terminal de nós de uma árvore r-ária. Reciprocamente, qualquer conjunto de nós terminais de uma árvore r-ária define um código livre de prefixo r-ário. Convencionamos que nenhum ramo é derivado a partir de um nó terminal associado a uma palavra-código de um código livre de prefixo.

Considere o código livre de prefixo dado por $z_1 = [011], z_2 = [10], z_3 = [11]$ e $z_4 = [00]$. A árvore correspondente é mostrada na Fig. 2.11.

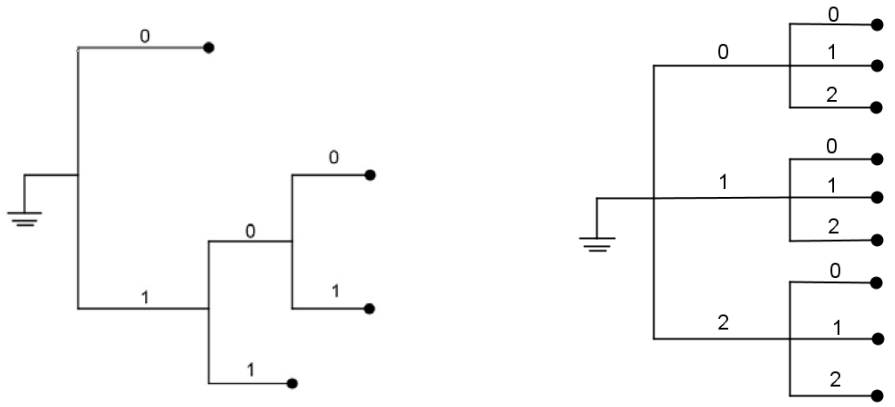


Figura 2.10: Exemplos de árvores r-árias.

2.9 Árvore com Probabilidades

Por uma árvore com probabilidades entendemos uma árvore finita com números não negativos (probabilidades) associados a cada um dos nós tal que:

(a) Ao nó raiz associamos probabilidade 1;

(b) A probabilidade de cada nó intermediário (incluindo o nó raiz) é a soma das probabilidades dos nós à profundidade 1 na subárvore que deriva deste correspondente nó intermediário. Observe que não estamos impondo a condição de que a árvore seja r-ária, isto é, ter o mesmo número de ramos derivados de todos os nós intermediários.

Definição 2.4. *Uma árvore é classificada como uma árvore com probabilidades se, e somente se, satisfaz as condições:*

1. $p_i \geq 0$, *onde os* p_i *são as probabilidades associadas aos nós terminais para* $i = 1, 2, \ldots, N$;
2. $\sum_{i=1}^{N} p_i = 1$, *isto é, a soma de todas as probabilidades dos nós terminais é igual a 1;*
3. P_i *denota a soma das probabilidades dos nós terminais da subárvore com profundidade 1 que emanam do nó à profundidade* l_i, *isto é,* $P_i = \sum_j p_j$.

Exemplo 2.11. A Fig. 2.12 mostra uma árvore com probabilidades. Note que esta árvore não é uma árvore ternária nem uma árvore binária.

Exemplo 2.12. A Fig. 2.13 mostra uma árvore com probabilidades.

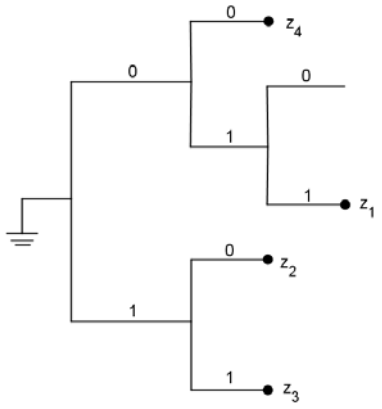


Figura 2.11: Árvore associada ao código livre de prefixo.

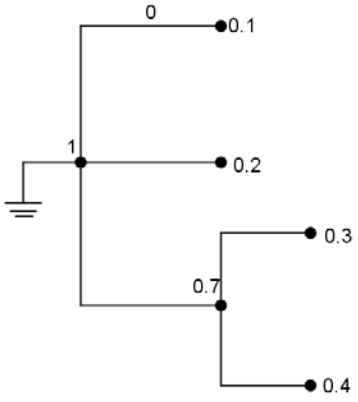


Figura 2.12: Árvore com probabilidades I.

Note também que em uma árvore com probabilidades a soma das probabilidades dos nós terminais deve ser 1.

Lema 2.1. *(Comprimento de Caminho) Em uma árvore com probabilidades, a profundidade média dos nós terminais iguala a soma das probabilidades dos nós intermediários (incluindo o nó raiz).*

Demonstração. A probabilidade de cada nó intermediário é igual a soma das probabilidades dos nós terminais da subárvore que deriva de cada nó intermediário. Porém, um nó terminal à profundidade d está em uma das d tais subárvores correspondentes aos nós intermediários no caminho que vai do nó raiz ao nó terminal. Assim, a soma das probabilidades dos nós intermediários iguala a soma dos produtos da probabilidade de cada nó terminal pela sua profundidade ou comprimento, mas esta soma é exatamente a profundidade média dos nós terminais.

□

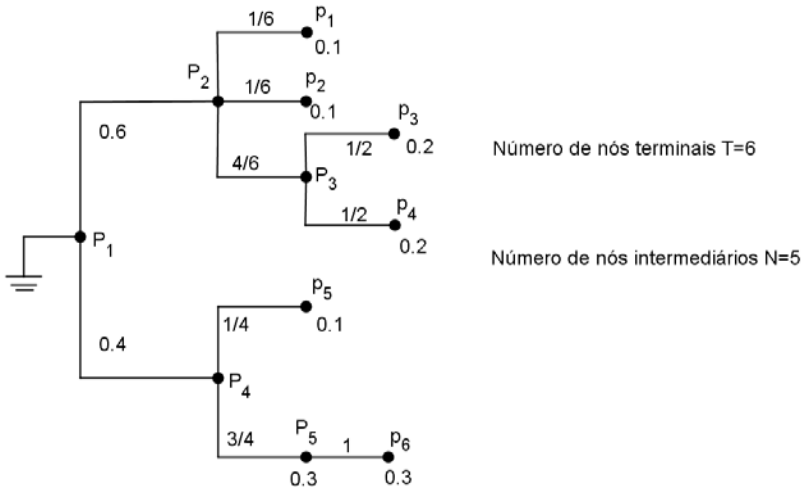


Figura 2.13: Árvore com probabilidades II.

No Exemplo 2.11, a profundidade média dos nós terminais é $1 + 0,7 = 1,7$, pelo Lema anterior. Como uma verificação, note que $1.(0,1) + 1.(0,2) + 2.(0,3) + 2.(0,4) = 1,7$.

Podemos considerar várias incertezas, que naturalmente podem ser definidas, para uma árvore com probabilidades. Ainda podemos pensar nas probabilidades de cada nó como a probabilidade que alcançaríamos um nó em uma caminhada aleatória através da árvore, iniciando no nó raiz e concluindo a jornada em algum nó terminal. Então, dado que atingimos um determinado nó intermediário, a probabilidade condicional de escolher cada um dos ramos que derivam deste nó como sendo o próximo passo a ser dado ao longo desta jornada é a probabilidade de atingir o próximo nó intermediário dividida pela probabilidade do nó intermediário anterior. Por exemplo, no Exemplo 2.11, a probabilidade de deslocar para os nós terminais com probabilidades $0,3$ e $0,4$, dado que estava no nó intermediário com probabilidade $0,7$ é $3/7$ e $4/7$, respectivamente.

De modo geral, suponha que uma árvore tenha T nós terminais cujas probabilidades são $p_1, p_2, \ldots, p_T$. Definimos a **incerteza nó terminal da árvore com probabilidades** como sendo

$$H_T = -\sum_{i:p_i \neq 0} p_i \log p_i. \tag{2.4}$$

Note que H_T pode ser considerada como a incerteza $H(\mathcal{J})$ de uma variável aleatória $\mathcal{J}$ cujos valores especificam os nós terminais alcançados durante a caminhada aleatória descrita anteriormente.

Suponha que a árvore tenha N nós intermediários ou não terminais, incluindo o nó raiz, cujas probabilidades são $P_1, P_2, \ldots, P_N$. Recorde do Lema do Comprimento de Caminho que $P_1 + P_2 + \ldots + P_N$ iguala ao comprimento médio, em ramos, da

caminhada aleatória descrita acima. Iremos, agora, definir a incerteza de ramificação para cada um dos nós intermediários tal que iguale a incerteza da variável aleatória que especifica o ramo saindo deste nó, dado que aquele nó foi alcançado na caminhada aleatória em consideração. Suponha que $q_{i1}, q_{i2}, \ldots, q_{iL_i}$ sejam as probabilidades dos nós (alguns podem ser nós intermediários e outros nós terminais) após L_i ramos tenham sido atravessados do nó cuja probabilidade é P_i. Então, a incerteza de ramificação, H_i, no i-ésimo nó intermediário é dada por

$$H_i = -\sum_{i:q_{ij}\neq 0} \frac{q_{ij}}{P_i} \log\left(\frac{q_{ij}}{P_i}\right), \tag{2.5}$$

pois q_{ij}/P_i é a probabilidade condicional de escolher o j-ésimo destes ramos como o próximo passo nesta caminhada dado que estamos no i-ésimo nó intermediário.

Exemplo 2.13. Suponha que os 4 nós terminais e os 2 nós não terminais da árvore do Exemplo 2.11 tenham sido numerados tal que $p_1 = 0,1, p_2 = 0,2, p_3 = 0,3, p_4 = 0,4, P_1 = 1, P_2 = 0,7$. Então,

$$H_T = -\sum_{i=1}^{4} p_i \log p_i = 1,846, \quad bits$$

Com isso, temos que $L_1 = 3$ e $q_{11} = 0,1, q_{12} = 0,2, q_{13} = 0,7$. Assim,

$$H_1 = -0,1\log 0,1 - 0,2\log 0,2 - 0,7\log 0,7 = 1,157, \quad bits$$

Finalmente, vemos que $L_2 = 2$, $q_{21} = 0,3$ e que $q_{22} = 0,4$. Assim,

$$H_2 = -\frac{3}{7}\log\frac{3}{7} - \frac{4}{7}\log\frac{4}{7} = 0,985, \quad bits$$

Devido à segunda condição da definição de árvore com probabilidades, temos

$$P_i = \sum_{j=1}^{L_i} q_{ij}. \tag{2.6}$$

De (2.6) juntamente com $\log(q_{ij}/P_i) = \log q_{ij} - \log P_i$ em (2.5), obtemos para o produto P_iH_i o seguinte valor

$$P_iH_i = -\sum_{i:q_{ij}\neq 0} q_{ij}\log q_{ij} + P_i \log P_i. \tag{2.7}$$

Utilizaremos a (2.7) para provar o seguinte resultado.

Teorema 2.6. *(Entropia da Folha) A incerteza nó terminal de uma árvore com probabilidades é igual a soma sobre todos os nós não terminais (incluindo o nó raiz) das incertezas de ramificação naquele nó ponderadas pelas probabilidades dos nós, isto é,*

$$H_T = \sum_{i=1}^{N} P_iH_i. \tag{2.8}$$

Demonstração. De (2.7) temos que o k-ésimo nó não terminal, se este não é o nó raiz, contribuirá com $P_k \log P_k$ para o termo na soma em (2.8) com $i = k$, mas contribuirá com $-P_k \log P_k$ para o termo i tal que $q_{ij} = P_k$ (isto é, para o termo i tal que o k-ésimo nó não terminal está ao final do ramo emanando do nó terminal i). Então, a contribuição total de todos os nós não terminais, exceto o nó raiz, para a soma em (2.8) é zero. O nó raiz, digamos $i = 1$, contribui somente com o termo $P_1 \log P_1$ para a soma em (2.8), mas este valor é também zero pois $P_1 = 1$. Finalmente, de (2.7), vemos que o k-ésimo nó terminal contribui com $-p_k \log p_k$ para a soma em (2.8), uma vez que a mesma afeta somente o termo para aquele i tal que $q_{ij} = p_k$. Então, acabamos de provar que

$$\sum_{i=1}^{N} P_i H_i = -\sum_{k=1}^{T} p_k \log p_k. \tag{2.9}$$

□

Exemplo 2.14. Continuando o Exemplo 2.13, calculamos H_T através da equação (2.8) de modo a obter

$$H_T = 1.H_1 + 0,7.H_2 = 1,157 + (0,7).(0,985) = 1,846, \quad bits$$

de acordo com os cálculos diretos realizados no Exemplo 2.13.

3 Algoritmo de Huffman

3.1 Primeiro Teorema de Shannon

Procedemos da seguinte maneira para apresentar o primeiro teorema de Shannon relativo a codificação de fontes: estabelecemos um limitante inferior absoluto do comprimento médio das palavras-código de um código ótimo; em seguida apresentamos o esquema de codificação proposto por Fano e Shannon, em que um limitante superior é estabelecido, finalmente através da estratégia de extensão de fonte proposta por Shannon, mostramos que o limitante superior poderá ser feito tão preciso quanto desejado através de quanto se queira estender a fonte.

O procedimento acima exposto tem como ponto de partida o problema de codificação sem ruído.

No que segue, seja $\mathcal{J}$ uma variável aleatória que pode assumir valores no conjunto $\{a_1, a_2, \ldots, a_J\}$ com probabilidades $p(a_1) = p_1, p(a_2) = p_2, \ldots, p(a_J) = p_J$, respectivamente. Considere também um código $\mathcal{C}$ cujas palavras-código são $c_1, c_2, \ldots, c_J$, com comprimentos $l_1, l_2, \ldots, l_J$. As palavras-código $c_i \in \mathcal{C}$ são compostas por símbolos de um alfabeto r-ário, $r \geq 2$, isto é, $\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$.

Queremos construir um código unicamente decodificável e ainda que minimize o comprimento médio das palavras-código, isto é, $L_{médio} = \sum_{i=1}^{J} p_i l_i$.

A solução deste problema inicia com o estabelecimento de um limitante inferior absoluto. Isto se dá pelo próximo teorema.

Teorema 3.1. *Se $L_{médio} = \sum_{i=1}^{J} p_i l_i$ é o comprimento médio das palavras-código de um código unicamente decodificável para a variável aleatória $\mathcal{J}$, então $L_{médio} \geq H(\mathcal{J})/\log r$ com a igualdade ocorrendo se, e somente se, $p_i = (1/r)^{l_i}$ para $i = 1, 2, \ldots, J$. Note que $H(\mathcal{J})/\log r$ é a incerteza de $\mathcal{J}$ tendo como base do logaritmo o valor r, isto é,*

$$\frac{H(\mathcal{J})}{\log r} = -\sum_{i=1}^{J} p_i \frac{\log p_i}{\log r} = -\sum_{i=1}^{J} p_i \log_r p_i.$$

Demonstração. Note que $L_{\text{médio}} \geq \dfrac{H(\mathcal{J})}{\log r}$ implica em $H(\mathcal{J}) - L_{\text{médio}} \log r \leq 0$.

Assim,

$$\sum_{i=1}^{J} p_i \log \frac{1}{p_i} - \sum_{i=1}^{J} p_i l_i \log r = \sum_{i=1}^{J} p_i \log \left[\frac{1}{p_i r^{l_i}} \right].$$

Como $\log z \leq (z-1)\log e$, temos

$$\sum_{i=1}^{J} p_i \log \left[\frac{r^{-l_i}}{p_i} \right] \leq \sum_{i=1}^{J} p_i \left[\frac{r^{-l_i}}{p_i} - 1 \right] \log e$$

$$\sum_{i=1}^{J} \left[r^{-l_i} - p_i \right] \log e = \left(\sum_{i=1}^{J} r^{-l_i} - \sum_{i=1}^{J} p_i \right) \log e.$$

Pela desigualdade de MacMillan, concluímos que

$$H(\mathcal{J}) - L_{\text{médio}} \log r \leq 0$$

a igualdade ocorre se, e somente se, $p_i = r^{-l_i}$, $i = 1, 2, \ldots, J$, ou seja, a máxima entropia. □

Um código unicamente decodificável cujo comprimento médio atinge a igualdade é dito **absolutamente ótimo**. Um exemplo de um código binário absolutamente ótimo é mostrado na Tabela 3.1.

$\mathcal{J}$	*Probab*	C	*Entropia*
a_1	1/2	0	
a_2	1/4	10	$H(\mathcal{J}) = L_{médio} = 7/4$
a_3	1/8	110	
a_4	1/8	111	

Tabela 3.1: Código binário absolutamente ótimo.

Como em geral não é possível construirmos códigos absolutamente ótimos para um dado conjunto de probabilidades $p_1, p_2, \ldots, p_J$, uma vez que se escolhermos l_i para satisfazer $p_i = (1/r)^{l_i}$, então $l_i = \log p_i / \log r$ pode não ser um inteiro. Porém, sabemos que existe um número inteiro entre a e $a+1$, onde a é um número real qualquer. Segundo este resultado, se $\log p_i / \log r$ não é um número inteiro, então podemos determinar um número inteiro para l_i entre $\log p_i / \log r$ e $\log p_i / \log r + 1$, isto é,

$$-\frac{\log p_i}{\log r} \leq l_i \leq -\frac{\log p_i}{\log r} + 1, \tag{3.1}$$

para $i = 1, 2, \ldots, J$.

Esta proposta fica sem sentido se não for possível construir um código unicamente decodificável com os comprimentos l_i fornecidos por (3.1). Para tal, usaremos a desigualdade de MacMillan para verificar ou não a existência deste código. Considere o lado esquerdo de (3.1), isto é, $p_i \geq r^{-l_i}$. Somando em i em ambos os lados, temos

$\sum_{i=1}^{J} p_i \geq \sum_{i=1}^{J} r^{-l_i}$. Como $\sum_{i=1}^{J} p_i = 1$, então $\sum_{i=1}^{J} r^{-l_i} \leq 1$. Portanto, existe um código unicamente decodificável.

De modo a obtermos o valor do comprimento médio, iremos multiplicar (3.1) por p_i e somarmos em i, isto é,

$$-\frac{p_i \log p_i}{\log r} \leq p_i l_i \leq -\frac{p_i \log p_i}{\log r} + p_i$$

$$-\sum_{i=1}^{J} \frac{p_i \log p_i}{\log r} \leq \sum_{i=1}^{J} p_i l_i \leq -\sum_{i=1}^{J} \frac{p_i \log p_i}{\log r} + \sum_{i=1}^{J} p_i.$$

Com isso, acabamos de provar o seguinte teorema.

Teorema 3.2. *(Primeiro Teorema de Shannon) Dada uma variável aleatória $\mathcal{J}$ com incerteza $H(\mathcal{J})$, existe um código unicamente decodificável com alfabeto r-ário para a variável aleatória $\mathcal{J}$ cujo comprimento médio das palavras-código $L_{médio}$ satisfaz*

$$\frac{H(\mathcal{J})}{\log r} \leq L_{médio} \leq \frac{H(\mathcal{J})}{\log r} + 1.$$

Sabemos que o limitante inferior do $L_{\text{médio}}$ é o menor possível, porém, o limitante superior não é tão preciso quanto desejado. Entretanto, podemos fazer com que o mesmo seja mais **preciso**. Esta precisão está diretamente relacionada com a extensão de um código. Por extensão de um código queremos dizer que ao invés de codificarmos cada símbolo na saída da fonte, codificamos blocos de n símbolos de cada vez.

Sejam $\mathcal{J}$ o conjunto de símbolos da fonte e $\mathcal{C}$ o correspondente conjunto de palavras-código, isto é, $\mathcal{J} = \{a_1, a_2, \ldots, a_J\}$ e $\mathcal{C} = \{c_1, c_2, \ldots, c_J\}$, em que a_i está relacionado com cada c_i. Assim, dada uma sequência de símbolos da fonte, digamos

$$\ldots, a_1, a_j, a_3, a_5, a_1, a_r, a_1, a_2, a_2, \ldots$$

a correspondente sequência de palavras-código é

$$\ldots, c_1, c_j, c_3, c_5, c_1, c_r, c_1, c_2, c_2, \ldots$$

onde J e r são maiores do que 6.

A n-ésima extensão de $\mathcal{J}$, denotada por $\mathcal{J}^n$, consiste da concatenação de n símbolos de $\mathcal{J}$, conduzindo a todas as possíveis combinações dos J símbolos de $\mathcal{J}$, isto é,

$$\mathcal{J}^n = \{a_1 a_1 \ldots a_1, a_1, a_1 \ldots a_2, \ldots, a_J a_J \ldots a_J\}.$$

Sabemos que a cada símbolo da fonte $\mathcal{J}$ temos a correspondente palavra-código do código $\mathcal{C}$, $\mathcal{C} = \{c_1, c_2, \ldots, c_J\}$. Como consequência, a n-ésima extensão de $\mathcal{C}$, denotada por $\mathcal{C}^n$, é $\mathcal{C}^n = \{c_1 c_1 \ldots c_1, c_1, c_1 \ldots c_2, \ldots, c_J c_J \ldots c_J\}$, em que cada elemento em $\mathcal{C}^n$ é a concatenação de n palavras-código, e está em correspondência biunívoca com os símbolos da fonte.

Por exemplo, considere a seguinte sequência

$$a_1 \quad a_3 \quad a_6 \quad a_4 \quad a_5 \quad a_1 \quad a_2 \quad a_2 \quad a_1 \quad a_5 \quad a_6 \quad a_8$$

de símbolos na saída da fonte. A correspondente sequência codificada é

$$c_1 \quad c_3 \quad c_6 \quad c_4 \quad c_5 \quad c_1 \quad c_2 \quad c_2 \quad c_1 \quad c_5 \quad c_6 \quad c_8$$

Uma possível extensão consiste em agrupar 4 elementos da sequência de símbolos, dando origem à sequência $\{\hat{a}_1, \hat{a}_2, \hat{a}_3, \ldots\}$, onde

$$\overbrace{a_1 \quad a_3 \quad a_k \quad a_4}^{\hat{a}_1} \quad \overbrace{a_5 \quad a_{10} \quad a_{20} \quad a_{12}}^{\hat{a}_2} \quad \overbrace{a_1 \quad a_5 \quad a_6 \quad a_8}^{\hat{a}_3}$$

em que os $\hat{a}_i$'s são os correspondentes símbolos da fonte estendida cujos símbolos pertencem a J^4. A correspondente sequência codificada é $\{\hat{c}_1, \hat{c}_2, \hat{c}_3, \ldots\}$, onde

$$\overbrace{c_1 \quad c_3 \quad c_6 \quad c_4}^{\hat{c}_1} \quad \overbrace{c_5 \quad c_1 \quad c_2 \quad c_2}^{\hat{c}_2} \quad \overbrace{c_1 \quad c_5 \quad c_6 \quad c_8}^{\hat{c}_3}$$

onde os $\hat{c}_i$'s pertencem ao código estendido $\mathcal{C}^n$.

Definição 3.1. *A n-ésima extensão da fonte discreta $\mathcal{J} = \{a_1, a_2, \ldots, a_J\}$ possui símbolos da forma $\hat{a}_{i_1}, \hat{a}_{i_2}, \ldots, \hat{a}_{i_n}$ tendo probabilidades*

$$p(\hat{a}_{i_1}, \hat{a}_{i_2}, \ldots, \hat{a}_{i_n}) = p(\hat{a}_{i_1})p(\hat{a}_{i_2})p(\hat{a}_{i_3}) \ldots p(\hat{a}_{i_n}).$$

Cada bloco de n símbolos da fonte original passa a ser um único símbolo $\hat{a}_{i_k}$ da fonte estendida $\mathcal{J}^n$ com probabilidade $p(\hat{a}_{i_k})$.

Observe que a entropia da fonte estendida $\mathcal{J}^n$ é dada por

$$H(\mathcal{J}^n) = \sum_{i_k=1}^{q} p(\hat{a}_{i_k}) \log \frac{1}{p(\hat{a}_{i_k})},$$

No entanto, como os símbolos $\hat{a}_{i_k}$'s são gerados por n símbolos independentes da fonte $\mathcal{J}$, segue pelo Teorema 2.5 que

$$H(\mathcal{J}^n) = nH(\mathcal{J}).$$

Ou seja, a entropia da n-ésima extensão da fonte é igual a n vezes a entropia do alfabeto original.

A partir do Primeiro Teorema de Shannon, pode-se provar que o limitante superior do comprimento médio do código é $nH(\mathcal{J}) + 1$. Finalmente, o comprimento médio, $\bar{L}_n$, do código estendido é dado por

$$nH(\mathcal{J}) \leq \bar{L}_n \leq nH(\mathcal{J}) + 1$$

$$H(\mathcal{J}) \leq \frac{\bar{L}_n}{n} \leq H(\mathcal{J}) + \frac{1}{n}.$$

Como cada palavra-código $\hat{c}_i$ do código estendido consiste da concatenação de n palavras-código do código original, segue que o comprimento médio das palavras-código do código estendido $\bar{L}_n$ é igual a n vezes o comprimento médio das palavras-código do código original, isto é,

$$\bar{L} = \frac{\bar{L}_n}{n}.$$

Consequentemente,

$$H(\mathcal{J}) \leq \bar{L} \leq H(\mathcal{J}) + \frac{1}{n}.$$

Portanto, se $n \to \infty$, então

$$\bar{L} = H(\mathcal{J}).$$

3.2 Códigos de Huffman

Nesta seção é levado em consideração, no processo de construção de códigos para compactação de dados, a frequência relativa (probabilidade) das mensagens, isto é, o número de vezes que cada símbolo (mensagem) da fonte ocorreu em um longo intervalo de tempo. Seja p_i a probabilidade associada a i-ésima mensagem. Suponha que esta mensagem tenha comprimento l_i. Então, o **comprimento médio** das palavras do código é dado por

$$L_{\text{médio}} = \sum_{i=1}^{J} p_i l_i.$$

Do problema de codificação sem ruído discutido e estabelecido na seção anterior, o que permanece para ser respondido é a apresentação de um método de construção de códigos que minimize o comprimento médio das palavras-código para um dado conjunto de probabilidades $p_1, p_2, \ldots, p_J$. Dentre os códigos unicamente decodificáveis, os códigos instantâneos são aqueles onde poderemos efetuar a busca dos códigos ótimos como mostra o lema a seguir.

Lema 3.1. *Suponha que $\mathcal{C}$ seja o código ótimo pertencente a classe dos códigos instantâneos para um dado conjunto de probabilidades $p_1, p_2, \ldots, p_J$, isto é, não existe um outro código instantâneo para $p_1, p_2, \ldots, p_J$ que apresente um menor valor de $L_{médio}$ do que o apresentado por $\mathcal{C}$. Então $\mathcal{C}$ é o código ótimo na classe dos códigos unicamente decodificáveis.*

Demonstração. A demonstração deste lema usa o fato de que, se $\mathcal{C}'$ é um código unicamente decodificável com $L_{\text{médio}}$ menor do que o apresentado por $\mathcal{C}$, então $\mathcal{C}'$ é um código contendo palavras-código com comprimentos $l'_1, l'_2, \ldots, l'_J$. Este código deve

satisfazer a desigualdade de MacMillan, isto é, $\sum_{i=1}^{J} \left(\frac{1}{r}\right)^{l'_i} \leq 1$. Por outro lado, da desigualdade de Kraft existe um código instantâneo $\mathcal{C}^n$ com palavras-código cujos comprimentos são $l'_1, l'_2, \ldots, l'_J$. Então, o comprimento médio destas palavras-código é o mesmo que aquele de $\mathcal{C}'$. Portanto, contradizendo que $\mathcal{C}'$ é o código instantâneo ótimo.

□

O código ótimo pertence a classe dos códigos instantâneos. A seguir estabelecemos as condições necessárias para que este código ótimo seja instantâneo. Isto é realizado através do seguinte lema.

Lema 3.2. *Seja $\mathcal{C}$ um código binário (q-ário) instantâneo com comprimentos $l_1, l_2, \ldots, l_J$ associado com o conjunto de probabilidades $p_1, p_2, \ldots, p_J$.Suponha que os símbolos sejam arranjados em ordem decrescente de probabilidades e que um grupo de símbolos com a mesma probabilidade é arranjado em ordem crescente de comprimento das palavras. Nestas condições, para que o código ótimo $\mathcal{C}$ seja um código instantâneo, $\mathcal{C}$ deve satisfazer as propriedades*

a. *símbolos com alta probabilidade devem apresentar comprimentos menores, isto é, se $p_i > p_j$ então $l_i < l_j$;*

b. *os dois símbolos menos prováveis apresentam palavras-código com o mesmo comprimento, isto é, $l_{J-1} = l_J$;*

c. *entre as palavras-código de comprimento l_J, deverá existir pelo menos duas palavras-código diferindo apenas no último dígito sendo que os demais são iguais. Por exemplo, o código a seguir não pode ser o ótimo visto que as palavras-código 4 e 5 não estão de acordo nos três primeiros dígitos.*

x_1	0
x_2	100
x_3	101
x_4	1101
x_5	1110

Demonstração. Para provar (**a**), notamos que se $p_i > p_j$ e $l_i > l_j$, é possível construir um código melhor $\mathcal{C}'$ trocando as palavras-código i e j. A diferença entre o comprimento médio da palavra-código de $\mathcal{C}'$ e de $\mathcal{C}$ é

$$\bar{L}' - \bar{L} = p_i l_j + p_j l_i - (p_i l_i + p_j l_j) = (p_i - p_j)(l_j - l_i) < 0;$$

portanto $\mathcal{C}'$ é melhor que $\mathcal{C}$.

Para provar (**b**), notamos primeiro que $l_{j-1} \leq l_j$, pois se $p_{j-1} > p_j$ então $l_{j-1} \leq l_j$ por (**a**); se $p_{j-1} = p_j$, então $l_{j-1} \leq l_j$ pela nossa suposição sobre arranjo de palavras-código associadas com símbolos igualmente prováveis. Agora, se $l_j > l_{j-1}$, podemos retirar o último dígito da j-ésima palavra-código para obter um código que ainda é instantâneo, e melhor do que o código original.

Finalmente, (**c**) é provado pela observação de que, se não há duas palavras-código de comprimento máximo que concordam em todas as posições exceto no último dígito, podemos retirar o último dígito de cada uma para obter um código melhor.

□

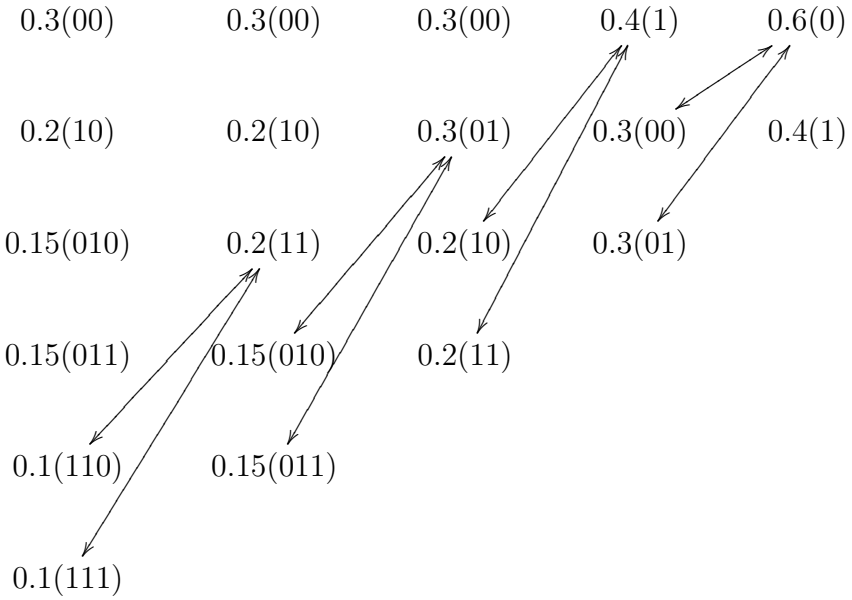
A seguir, apresentaremos o método de construção de códigos ótimos instantâneos proposto por Huffman.

3.2.1 Determinação do Código de Huffman

O algoritmo é realizado com uma rotina, primeiramente arrange em ordem decrescente as probabilidades de ocorrência das mensagens. Se o código é binário, então some as duas menores probabilidades. Rearrange este novo conjunto de probabilidades em ordem decrescente. Note que a este conjunto de probabilidades um código estará sendo associado. Novamente, some os dois menores valores de probabilidades e rearrange esse novo conjunto de probabilidades. Da mesma forma que no passo anterior, a este novo conjunto de probabilidades um novo código estará sendo associado. Continue este processo até que tenha somente dois valores de probabilidades. Atribua valores "0"e "1"a cada uma delas. Em seguida, siga o caminho inverso das somas das probabilidades concatenando "0"e "1"às palavras-código do código obtido no passo anterior, as probabilidades que deram origem à soma destas probabilidades. Continue esse processo até que o arranjo inicial seja alcançado.

Vejamos como este algoritmo funciona na prática. Considere o seguinte exemplo: $p_1 = 0.3$, $p_2 = 0.2$, $p_3 = p_4 = 0.15$, $p_5 = p_6 = 0.1$. Aplicando o algoritmo de Huffman

a este conjunto de probabilidades temos o seguinte diagrama:



Portanto, o código de Huffman é: 00, 10, 010, 011, 110, 111. O correspondente código construído a partir da árvore de decisão é como mostrado na Fig. 3.1.

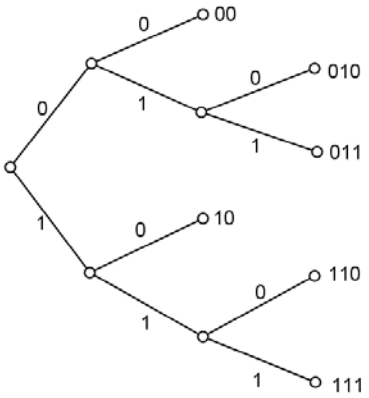


Figura 3.1: Árvore de decisão do código de Huffman.

O comprimento médio deste código é

$$L_{\text{médio}} = 2(0.3) + 2(0.2) + 3(0.15) + 3(0.15) + 3(0.1) + 3(0.1) = 2.5.$$

A entropia da fonte é facilmente calculada e vale $H(\mathcal{J}) = 2.32$. Note que $L_{\text{médio}} > H(\mathcal{J})$, porém é o valor de $L_{\text{médio}}$ mais próximo que podemos chegar de $H(\mathcal{J})$ sem que a fonte seja estendida.

Para o caso em que o alfabeto do código é q-ário, o número de símbolos que a fonte deverá ter é dado por $n = (q-1)k + q$. Assim, para $q = 2$, $n = k + 2$, para $q = 3$, $n = 2k + 3$, ...; para $k = 0, 1, 2, \ldots$.

Para finalizar este capítulo façamos algumas considerações sobre o comprimento variável para fontes discretas sem memória.

Primeiramente, note que a maioria das fontes de informação não emitem uma única variável aleatória, mas ao invés, imite uma sequência de variáveis aleatórias.

Este fato foi considerado quando do estudo realizado para a codificação de variáveis aleatórias por códigos livre de prefixo, de tal forma que pudesse ser detectado o final das palavras-código associadas à sequência de palavras-código.

Ao intuito de abordar esta vertente tratamos, a seguir, dos Códigos de Comprimento Variável para Fontes Discretas sem Memória

Definição 3.2. *Uma fonte discreta sem memória (DMS) é um dispositivo cuja sequência de saída $\mathcal{J}_1, \mathcal{J}_2, \ldots$ é uma sequência de variáveis aleatórias estatisticamente independentes e identicamente distribuídas.*

Ou seja, uma fonte discreta sem memória é a mais simples das fontes de informação a ser considerada. Além disso, é um modelo apropriado para certas aplicações práticas. Entretanto, muitas fontes de informações reais possuem memória, no sentido de que sucessivos símbolos na saída da fonte não são estatisticamente independentes dos símbolos que aconteceram anteriormente. O resultado a seguir mostra que a **taxa de informação** de uma fonte discreta sem memória é precisamente $H(\mathcal{J})$ bits/símbolo, onde $H(\mathcal{J})$ é a incerteza associada a um símbolo na saída da fonte.

Teorema 3.3. *Existe um código r-ário livre de prefixo para um bloco de comprimento N de símbolos de uma fonte discreta sem memória tal que o comprimento médio da palavra-código satisfaz*

$$\frac{E[L]}{N} < \frac{H(\mathcal{J})}{\log r} + \frac{1}{N}, \tag{3.2}$$

onde $H(\mathcal{J})$ denota a incerteza de um único símbolo da fonte. Reciprocamente, para todo código r-ário livre de prefixo para um bloco de N símbolos,

$$\frac{E[L]}{N} \geq \frac{H(\mathcal{J})}{\log r}. \tag{3.3}$$

Demonstração. Primeiramente, note que podemos codificar a saída de uma fonte discreta sem memória através da codificação de variáveis aleatórias V, em que $V = [\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_N]$. Assim, do Teorema 2.5, temos

$$H(V) = H(\mathcal{J}_1) + H(\mathcal{J}_2) + \ldots + H(\mathcal{J}_N) = NH(\mathcal{J}), \tag{3.4}$$

onde a segunda igualdade segue da condição de que as variáveis $\mathcal{J}_1, \mathcal{J}_2, \ldots, \mathcal{J}_N$ são identicamente distribuídas.

Com isso, podemos aplicar o Primeiro Teorema de Shannon para concluir que

$$E[L] \geq \frac{H(V)}{\log r} = \frac{NH(\mathcal{J})}{\log r},$$

do qual a equação (3.3) segue.

Reciprocamente, podemos fazer uso do Primeiro Teorema de Shannon para concluir que existe um código livre de prefixo para V tal que

$$E[L] \leq \frac{H(V)}{\log r} + 1 = \frac{NH(\mathcal{J})}{\log r} + 1.$$

Após a divisão por N chegamos a equação (3.2).

□

4 Algoritmos Lempel-Ziv

Para abordarmos este tema é preciso primeiramente estabelecer definições, resultados e notações acerca da teoria sobre processos estocásticos e outros tópicos. Para mais detalhes veja [7].

4.1 Preliminares

Para o que segue consideramos como variáveis aleatórias i.i.d. (independentes e identicamente distribuídas) variáveis que são independentes entre si e possuem mesma função de distribuição de probabilidades.

Definição 4.1. *(Convergência de variáveis aleatórias). Dada uma sequência de variáveis aleatórias, $X_1, X_2, \ldots$, dizemos que a sequência $X_1, X_2, \ldots$ converge para uma variável aleatória X:*

1. **Em probabilidade** *se para cada $\epsilon > 0$, $Pr\{\mid X_n - X \mid > \epsilon\} \to 0$*
2. **Em média quadrática** *se $E(X_n - X)^2 \to 0$*
3. **Com probabilidade 1***(também chamada* **quase certamente***) se*

$$Pr\left\{\lim_{n\to\infty} X_n = X\right\} = 1$$

Note que para variáveis aleatórias independentes e identicamente distribuídas (i.i.d.), $\frac{1}{n}\sum_{i=1}^{n} X_i$ está próximo de seu valor esperado $E(X)$ para grandes valores de n.

Teorema 4.1. *Se $X_1, X_2, \ldots$ são i.i.d. com distribuição de probabilidade $p(x)$, então*

$$-\frac{1}{n}\log p(X_1, X_2, \ldots, X_n) \to H(X) \text{ em probabilidade.}$$

Em que H é a entropia.

Demonstração. Vide página 58, de [7].

Ou seja, esse teorema garante que $\frac{1}{n}\log\frac{1}{p(X_1, X_2, \ldots, X_n)}$ está próximo da entropia H, em que $X_1, X_2, \ldots, X_n$ são variáveis aleatórias i.i.d. e $p(X_1, X_2, \ldots, X_n)$ é a probabilidade de observar a sequência $X_1, X_2, \ldots, X_n$.

Definição 4.2. *A* **entropia** *de um processo estocástico $\{X_i\}$ é definida por*

$$H(\mathcal{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, \ldots, X_n)$$

quando o limite existe.

Teorema 4.2. *Se H é a taxa de entropia de um valor finito de processos ergódicos estacionários $\{X_n\}$, então*

$$-\frac{1}{n}\log p(X_1, X_2, \ldots, X_{n-1}) \to H(\mathcal{X}) \text{ com probabilidade 1.}$$

Demonstração. Vide páginas 645 e 646, de [7].

Observação 4.1. Para definir com rigor o significado de um processo ergódico e estacionário é preciso um aprofundamento em teoria da probabilidade, o que não faremos aqui. Porém, apresentamos uma breve noção, com efeito:

Uma fonte ergódica é definida em um espaço de probabilidade $(\Omega, \mathcal{B}, P)$, onde $\mathcal{B}$ é um σ-algebra de subconjuntos de Ω, e P é uma medida de probabilidade. Uma variável aleatória X é definida como uma função $X(\omega)$, $\omega \in \Omega$, no espaço de probabilidade. Temos também uma transformação $T : \Omega \longrightarrow \Omega$, que desempenha o papel de um desvio de tempo. Dizemos que a transformação é **estacionária** se $P(TA) = P(A)$ para todo $A \in \mathcal{B}$. A tranformação é chamada **ergódica** se cada conjunto A de tal modo que $TA = A$, satisfaz $P(A) = 0$ ou 1. Se T é estacionária e ergódica, dizemos que o processo definido por $X_n(\omega) = X(T^n\omega)$ é estacionário e ergódico.

Para convergência de um processo estacionário e ergódico, usaremos também o seguinte resultado.

Lema 4.1. *(Aproximações de Markov) Para um processo estocástico estacionário e ergódico $\{X_n\}$,*

$$-\frac{1}{n}\log p^k(X_0^{n-1}) \longrightarrow H^k \text{ com probabilidade 1,}$$

$$-\frac{1}{n}\log p(X_0^{n-1} \mid X_{-\infty}^{-1}) \longrightarrow H^\infty \text{ com probabilidade 1.}$$

Demonstração. Vide página 647, de [7].

Para abordarmos o último resultado necessário para o desenvolvimento deste capítulo enunciamos a seguinte definição.

Definição 4.3. *A função $f(x)$ é dita* **convexa** *ao longo de um intervalo (a, b) se para cada $x_1, x_2 \in (a, b)$ e $0 \leq \lambda \leq 1$,*

$$f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2).$$

Uma função f é dita **estritamente convexa** *se a igualdade vale apenas se $\lambda = 0$ ou $\lambda = 1$.*

Além disso, dizemos que uma função f é **côncava** se $-f$ é convexa.

Exemplos de funções convexas incluem $x^2, |\ x\ |, e^x, x \log x$ (para $x \geq 0$), entre outras. Enquanto que $\log x$ e $\sqrt{x}$, para $x \geq 0$, são funções côncavas.

Teorema 4.3. *(Desigualdade de Jensen) Se f é uma função convexa e X é uma variável aleatória,*

$$E\left(f(X)\right) \geq f(E(X)). \tag{4.1}$$

Além disso, se f é estritamente convexa, a desigualdade (4.1) implica que $X = E(X)$ com probabilidade 1 (isto é, X é uma constante).

Demonstração. Vide página 27, de [7].

4.2 Codificação Lempel-Ziv

Apresentaremos neste último capítulo uma destacada classe de técnicas de codificação de fonte cuja taxa de compressão assintótica se aproxima da taxa de entropia da fonte para qualquer fonte ergódica estacionária e, além disso, este tipo de codificação é simples de implementar, a saber, codificação Lempel-Ziv.

Esta classe de algoritmos é chamada de Lempel-Ziv, em homenagem aos autores de dois artigos [4] e [5], que descrevem os dois algoritmos básicos que fundamentam esta classe.

Por outro lado, estes algoritmos podem também ser descritos como algoritmos de compressão adaptativa baseados em dicionário. A ideia de usar dicionários para compressão remonta à invenção do telégrafo. Nessa época, as empresas eram cobradas pelo número de letras usadas, e muitas empresas produziram livros de códigos para as frases usadas com maior frequência e usavam as palavras-código para a sua comunicação telegráfica. Outro exemplo é a noção de telegramas de saudações que são populares na Índia, há um conjunto de saudações padrão, como "25:Merry Christmas"("25: Feliz Natal") e "26:May Heaven's choicest blessings be showered on the newly married couple."("26: Que os céus de maio derramem as mais preciosas bênçãos sobre os recém-casados). Uma pessoa que deseja enviar um cumprimento precisa apenas especificar o número, o qual é usado para gerar o cumprimento real no local de destino.

Sistemas baseados em dicionário de adaptação não foram explorados até que Ziv e Lempel escreveram seus trabalhos em 1977 e 1978. Existem dois artigos que descrevem

duas versões distintas do algoritmo, nos referimos a estes trabalhos como LZ77 (ou **janela deslizante de Lempel-Ziv**) e LZ78 (ou **árvore-estruturada de Lempel-Ziv**).

Destacamos que o principal objetivo das versões do algoritmo Lempel-Ziv é analisar a sequência de símbolos e substituir a sequência pela indicação de onde ocorreu a mesma sequência no passado. O que distingue estas versões é o conjunto de possíveis localizações das sequências (e comprimentos das sequências), que o algoritmo permite.

No que segue, exploraremos LZ77 e LZ78.

4.2.1 Algoritmo de janela deslizante de Lempel-Ziv

O algoritmo descrito no artigo de 1977 consiste em procurar sequências repetidas, isto é, comparar o conteúdo de uma janela que contém os caracteres ainda não codificados com uma sequência já codificada em uma outra janela (designada por dicionário), tentando encontrar a maior sequência igual àquelas já codificadas. A sequência encontrada é substituída por um par de números composto por: uma distância, que representa a diferença entre o início da sequência já codificada e o início da sequência ainda não codificada; um comprimento, que representa o número de caracteres iguais na sequência do dicionário. O algoritmo LZ77 divide a sequência de símbolos a codificar num conjunto de subsequências, e associa a cada subsequência um código composto por 2 inteiros e o último símbolo (caractere) dessa subsequência .

Exemplo 4.1. Suponha que queremos codificar a sequência "a_asa_da_casa", sendo $N_d = 8$ o comprimento do dicionário e $N_b = 4$ o comprimento da janela que contém os próximos N_b símbolos a serem codificados. Esse tipo de janela é chamada **buffer**. Usando o algoritmo teremos o seguinte:

Dicionário	Buffer	Restante da sequência	Código
	a_as	a_da_casa	$(0, 0, a)$
a	_asa	_da_casa	$(0, 0, \_)$
a_	asa_	da_casa	$(2, 1, s)$
a_as	a_da	_casa	$(4, 2, d)$
a_asa_d	a_ca	sa	$(3, 2, c)$
asa_da_c	asa		$(8, 3, null)$

Tabela 4.1: Codificação pelo algoritmo LZ77

A palavra *null* significa apenas que não há mais símbolos a serem codificados. Assim, a saída completa do codificador é

$$(0, 0, a)\ (0, 0, \_)\ (2, 1, s)\ (4, 2, d)\ (3, 2, c)\ (8, 3, null)$$

Veremos no próximo exemplo que o processo de descodificação consiste somente em interpretar as palavras de códigos recebidas. Ao longo da formação da sequência estabelecemos a base para descodificar futuras palavras de código.

Exemplo 4.2. Suponhamos que a sequência de entrada de um descodificador LZ77 seja a sequência que acabamos de codificar: $(0,0,a)$ $(0,0,\_)$ $(2,1,s)$ $(4,2,d)$ $(3,2,c)$ $(8,3,null)$. O descodificador usa uma janela com as mesmas dimensões ($N_d = 8$ e $N_b = 4$). Assim, temos:

Apontador 1 $(0,0,a)$

Sequência de saída: **a**

Apontador 2 $(0,0,\_)$

Sequência de saída: a_

Apontador 3 $(2,1,s)$

Sequência de saída: a_ **as**

Apontador 4 $(4,2,d)$

Sequência de saída: a_as **a_d**

Apontador 5 $(3,2,c)$

Sequência de saída: a_asa_d **a_c**

Apontador 6 $(8,3,null)$

Sequência de saída: a_asa_da_c **asa**

Vejamos uma síntese de como é implementado esse algoritmo. No que segue consideramos

N_b = comprimento do buffer.

N_d = comprimento do dicionário.

- Inicializa-se o buffer (de dimensão N_b) com os primeiros N_b símbolos da sequência a codificar.
- Inicialmente o dicionário não contém nenhum símbolo.

  Enquanto houver símbolos para codificar

  - Identificar no dicionário a maior sequência de símbolos que também esteja presente no buffer (a começar no cursor).

– Associar à sequência o seguinte código $(p, l, \mathbf{c})$, onde
p é a posição relativa (a contar do cursor) da maior sequência do dicionário.
l é o comprimento da maior sequência.
c é o símbolo do buffer que se segue à sequência.

- Deslocar as janelas (dicionário e buffer) de $l + 1$ símbolos

O código $(p, l, \mathbf{c})$ significa que a sequência no buffer pode ser obtida recuando p símbolos no dicionário, copiando l símbolos a partir daí, e acrescentando no fim o símbolo $\mathbf{c}$.

4.2.2 Otimalidade de LZ77

No artigo original de Ziv e Lempel, os autores descreveram o algoritmo básico LZ77 e provaram que poderiam comprimir qualquer sequência. No entanto, eles não demonstravam se este algoritmo alcançava a otimalidade assintótica (isto é, que a taxa de compressão converge para a entropia da fonte). Este resultado foi provado por Wyner e Ziv [8].

A prova conta com um lema simples devido a Kac: a duração média de tempo que você precisa esperar para ver um símbolo especial é o inverso da probabilidade de um símbolo. Assim, estamos propensos a ver as sequências de caracteres de alta probabilidade dentro da janela e codificar essas sequências de forma eficiente. As sequências que não se encontram dentro da janela tem baixa probabilidade, de modo que assintoticamente, eles não influenciam a compressão alcançada.

Em vez de provar a otimalidade da versão prática do LZ77, vamos apresentar uma prova mais simples para uma versão diferente do algoritmo, que, embora não seja tão prático, captura algumas das ideias básicas.

Este algoritmo assume que ambos o emissor e o receptor tenham acesso ao passado infinito da sequência, ou seja, a todas as vezes que esta sequência ocorreu no passado, e representa uma sequência de comprimento n, apontando para a última vez que a sequência ocorreu no passado.

Supomos que temos um processo estacionário e ergódico definido para o tempo de $-\infty$ a ∞, e que ambos o codificador e o decodificador tem acesso a $\ldots, X_{-2}, X_{-1}$, o infinito passado da sequência, ou seja, acesso a todos os símbolos já vistos na sequência. Em seguida, para codificar $X_0, X_1, \ldots, X_{n-1}$ (ou seja, um bloco de comprimento n), encontramos a última vez que vimos esses n símbolos no passado. Seja então

$$R_n(X_0, X_1, \ldots, X_{n-1}) = max\{j < 0 : (X_{-j}, X_{-j+1}, \ldots, X_{-j+n-1}) = (X_0, \ldots, X_{n-1})\} \tag{4.2}$$

Então, para representar $X_0, \ldots, X_{n-1}$, precisamos apenas enviar R_n para o receptor, que pode, então, olhar para R_n bits atrás no passado e recuperar $X_0, \ldots, X_{n-1}$. Assim, o custo da codificação é o custo de representar R_n. Vamos mostrar que este custo é de

aproximadamente $\log R_n$ e que assintoticamente $\frac{1}{n}E[\log R_n] \longrightarrow H(\mathcal{X})$, comprovando, assim, a otimização assintótica desse algoritmo.

No que segue precisamos dos seguintes lemas.

Lema 4.2. *Existe um código livre de prefixo para os números inteiros de tal forma que o comprimento da palavra-código para o inteiro k é*

$$\log k + 2\log\log k + O(1). \tag{4.3}$$

Demonstração. Se soubéssemos que $k \leq m$, poderíamos codificar k com $\log m$ bits. No entanto, uma vez que não temos um limite superior para k, precisamos dizer ao receptor o comprimento da codificação de k (ou seja, é preciso especificar $\log k$). Considere a seguinte codificação para o número inteiro k: representamos primeiro $\lceil \log k \rceil$ em unário, seguido pela representação binária de k:

$$C_1(k) = \underbrace{00\ldots0}_{\lceil \log k \rceil \ 0's} \ 1 \quad \underbrace{xx\ldots x}_{k \text{ em binário}} \quad .$$

Note que o comprimento desta representação é $2\lceil \log k \rceil + 1 \leq 2\log k + 3$. Isso é mais do que o comprimento que procuramos já que usamos o código unário muito ineficiente para enviar o $\log k$. No entanto, se usarmos C_1 para representar $\log k$, essa representação tem um comprimento inferior a $\log k + 2\log\log k + 4$, o que prova o lema. □

O resultado fundamental subjacente à prova da otimização de LZ77 é o lema de Kac, que relaciona o tempo médio de recorrência para a probabilidade de um símbolo para qualquer processo ergódico estacionário. Por exemplo, se $X_0, X_1, \ldots, X_{n-1}$ é um processo i.i.d.(independente e identicamente distribuida), perguntamo-nos qual é o tempo médio de espera para ver o símbolo a novamente, condicionada ao fato de que $X_0 = a$. Neste caso, o tempo de espera tem uma distribuição geométrica com parâmetro $p = p(a) := Pr(X_0 = a)$, e, assim, o tempo médio de espera é $1/p(a)$. O resultado surpreendente é que o mesmo é verdadeiro mesmo se o processo não é i.i.d., mas estacionário e ergódico. A razão intuitiva simples para isso é que, em uma longa amostra de tamanho n, esperaríamos ter a aproximadamente $np(a)$ vezes, e a distância média entre estas ocorrências de a é $n/(np(a))$(isto é, $1/p(a)$).

Lema 4.3. *(Kac) Seja $\ldots, U_{-2}, U_{-1}, U_0, U_1, \ldots$ um processo ergódico e estacionário em um alfabeto numerável. Para qualquer u tal que $p(u) := Pr(U_0 = u) > 0$ e $i = 1, 2, \ldots$, considere*

$$Q_u = Pr\left\{U_{-i} = u; U_j \neq u \text{ para } -i < j < 0 \mid U_0 = u\right\} \tag{4.4}$$

isto é, $Q_u(i)$ é a probabilidade condicional de que a ocorrência anterior mais recente do símbolo u é i, dado que $U_0 = u$.

Sob estas condições

$$E(R_1(U) \mid U_0 = u) = \sum_i iQ_u(i) = \frac{1}{p(u)}. \tag{4.5}$$

Ou seja, o tempo médio de espera condicional para ter o símbolo u novamente, anteriormente ao zero, é $1/p(u)$.

Observe o fato interessante de que o tempo de recorrência esperado é

$$E(R_1(U)) = \sum p(u)\frac{1}{p(u)} = m, \tag{4.6}$$

em que m é o tamanho do alfabeto.

Demonstração. Seja $U_0 = u$. Defina os eventos para $j = 1, 2, \ldots$ e $k = 0, 1, 2, \ldots$:

$$A_{jk} = \{U_{-j} = u, U_l \neq u, -j < l < k, U_k = u\}. \tag{4.7}$$

O evento A_{jk} corresponde ao caso em que a última ocorrência de u que é anterior ao zero, acontece no tempo $-j$, a primeira vez após o zero em que o processo é igual a u é k. Estes eventos são disjuntos, e pela ergodicidade, a probabilidade $Pr\left\{\bigcup_{j,k} A_{jk}\right\} = 1$. Assim,

$$1 = Pr\left\{\bigcup_{j,k} A_{jk}\right\} \tag{4.8}$$

$$\stackrel{(a)}{=} \sum_{j=1}^{\infty}\sum_{k=0}^{\infty} Pr\{A_{jk}\} \tag{4.9}$$

$$= \sum_{j=1}^{\infty}\sum_{k=0}^{\infty} Pr(U_k = u) Pr\{U_{-j} = u, U_l \neq u, -j < l < k \mid U_k = u\} \tag{4.10}$$

$$\stackrel{(b)}{=} \sum_{j=1}^{\infty}\sum_{k=0}^{\infty} Pr(U_k = u) Q_u(j+k) \tag{4.11}$$

$$\stackrel{(c)}{=} \sum_{j=1}^{\infty}\sum_{k=0}^{\infty} Pr(U_0 = u) Q_u(j+k) \tag{4.12}$$

$$= Pr(U_0 = u) \sum_{j=1}^{\infty}\sum_{k=0}^{\infty} Q_u(j+k) \tag{4.13}$$

$$\stackrel{(d)}{=} Pr(U_0 = u) \sum_{i=1}^{\infty} iQ_u(i), \tag{4.14}$$

onde (a) decorre do fato de que o A_{jk} são disjuntos, (b) segue-se a partir da definição de $Q_u(\cdot)$, (c) segue da estacionariedade, e (d) decorre do fato de que há i pares (j, k) de tal modo que $j + k = i$. Ou seja, $1 = p(u) \sum_{i=1}^{\infty} iQ_u(i)$. □

Corolário 4.1. *Seja $\ldots, X_{-1}, X_0, X_1, \ldots$ um processo ergódico e estacionário e seja $R_n(X_0, \ldots, X_{n-1})$ o tempo de recorrência olhando para trás como definido em (4.2). Então*

$$E\left[R_n(X_0, \ldots, X_{n-1}) \mid (X_0, \ldots, X_{n-1}) = x_0^{n-1}\right] = \frac{1}{p(x_0^{n-1})}. \tag{4.15}$$

Demonstração. Defina um novo processo com $U_i = (X_i, X_{i+1}, \ldots, X_{i+n-1})$. O processo de U também é estacionário e ergódico, e assim pelo lema de Kac o tempo médio de recorrência para U condicionado a $U_0 = u$ é $1/p(u)$. Adaptando isso para o processo X temos o corolário. □

Estamos agora em condições de provar o principal resultado, que mostra que a taxa de compressão para a versão simples de Lempel-Ziv com tempo de recorrência se aproxima da entropia. O algoritmo descreve X_0^{n-1} descrevendo $R_n(X_0^{n-1})$, que pelo Lema 4.2 pode ser feito com $\log R_n + 2\log\log R_n + 4$ bits. Vamos agora provar o teorema que comprova a otimalidade de LZ77.

Teorema 4.4. *Seja $L_n(X_0^{n-1}) = \log R_n + 2\log\log R_n + O(1)$ o comprimento da descrição para X_0^{n-1} no algoritmo simples descrito acima. Então*

$$\frac{1}{n} E[L_n(X_0^{n-1})] \longrightarrow H(\mathcal{X}) \tag{4.16}$$

quando $n \to \infty$, onde $H(\mathcal{X})$ é a entropia do processo $\{X_i\}$.

Demonstração. Vamos provar que existem limites superiores e inferiores para $E(L_n)$. O limite inferior segue diretamente dos resultados de codificação de fonte padrão (ou seja, o $E(L_n) \geq nH$ para qualquer código livre de prefixo). Para provar que o limite superior existe, mostraremos primeiro que

$$\limsup \frac{1}{n} E(\log R_n) \leq H \tag{4.17}$$

e depois limitar os outros termos na expressão para L_n.

Para provar o limite para o $E(\log R_n)$, expandimos a expectativa pelo condicionamento sobre o valor de X_0^{n-1} e, em seguida, aplicamos a desigualdade de Jensen 4.1. Assim,

$$\frac{1}{n} E(\log R_n) = \frac{1}{n} \sum_{x_0^{n-1}} p(x_0^{n-1}) E[\log R_n(X_0^{n-1}) \mid X_0^{n-1} = x_0^{n-1}] \tag{4.18}$$

$$\leq \frac{1}{n} \sum_{x_0^{n-1}} p(x_0^{n-1}) \log E[R_n(X_0^{n-1}) \mid X_0^{n-1} = x_0^{n-1}] \tag{4.19}$$

$$\overset{(a)}{=} \frac{1}{n} \sum_{x_0^{n-1}} p(x_0^{n-1}) \log \frac{1}{p(x_0^{n-1})} \tag{4.20}$$

$$= \frac{1}{n} H(X_0^{n-1}) \tag{4.21}$$

onde (a) segue do Lema 4.3.

Logo,

$$\frac{1}{n}E(\log R_n) = \frac{1}{n}H(X_0^{n-1}) \longrightarrow H(\mathcal{X}), \text{ quando } n \to \infty$$

O segundo termo na expressão para L_n é $\log\log R_n$, e queremos mostrar que

$$\frac{1}{n}E[\log\log R_n(X_0^{n-1})] \longrightarrow 0. \tag{4.22}$$

Novamente, usamos a desigualdade de Jensen,

$$\frac{1}{n}E[\log\log R_n] \leq \frac{1}{n}\log E[\log R_n(X_0^{n-1})] \tag{4.23}$$

$$\leq \frac{1}{n}\log H(X_0^{n-1}), \tag{4.24}$$

onde a última desigualdade segue de (4.21).

Para qualquer $\epsilon > 0$, para um n suficientemente grande, $H(X_0^{n-1}) < n(H+\epsilon)$, e, por conseguinte,

$$\frac{1}{n}E[\log\log R_n] \leq \frac{1}{n}\log H(X_0^{n-1}) < \frac{1}{n}\log[n(H+\epsilon)] = \frac{1}{n}\log n + \frac{1}{n}\log(H+\epsilon) \to 0.$$

□

Assim, um sistema de compressão que representa uma sequência pela codificação feita a última vez que foi observada a mesma sequência no passado é assintoticamente ótima. É claro que este sistema não é prático, uma vez que pressupõe que o remetente e o destinatário tenham acesso ao infinito passado de uma sequência. Para sequências mais longas, seria necessário estudar mais o passado para encontrar uma correspondência. Por exemplo, se a taxa de entropia é $\frac{1}{2}$ e a sequência tem comprimento de 200 bits, seria preciso procurar uma média de $2^{100} \approx 10^{30}$ bits no passado para encontrar uma correspondência. Embora isso não seja viável, o algoritmo ilustra a ideia básica de que combinando com o passado é assintoticamente ideal. A prova da otimização da versão prática de LZ77 com uma janela finita é baseado em ideias semelhantes. Não vamos apresentar os detalhes aqui, mas a prova original pode ser encontrado em [8].

4.2.3 Algoritmos de árvore-estruturada de Lempel-Ziv

No artigo de 1978, Ziv e Lempel descreveram um algoritmo que analisa uma sequência em frases, em que cada frase é a frase mais curta não vista anteriormente. Este algoritmo pode ser visto como a construção de um dicionário, sob a forma de uma árvore, onde os nós correspondem a frases já vistas.

O algoritmo é particularmente simples de implementar e tornou-se popular como um dos primeiros algoritmos padrões para compressão de arquivos em computadores por causa de sua velocidade e eficiência. É também usado para a compressão de dados em modems de alta velocidade.

A sequência de origem é analisada dividindo-a em subsequências que ainda não haviam aparecido até então.

Por exemplo, se a sequência é ABBABBABBBAABABAA..., Vamos analisá-la da seguinte maneira: A, B, BA, BB, AB, BBA, ABA, BAA ... . Na Figura 4.1 fica claro que a maneira como a sequência é analisada forma uma árvore onde os nós representam as frases formadas. Depois de cada vírgula, olhamos ao longo da sequência de entrada até que chegamos à sequência de caractéres mais curta que ainda não tenha sido analisada. Uma vez que esta é a mais curta ainda não analisada, todos os seus prefixos já devem ter sido analisados anteriormente (Assim, podemos construir uma árvore com essas frases). Em particular, a sequência que consiste de todos os caracteres. Codificaremos esta frase dando a localização do prefixo e valor do último símbolo da frase. Assim, a sequência considerada seria representada por $(0, A)$, $(0, B)$, $(2, A)$, $(2, B)$, $(1, B)$, $(4, A)$, $(5, A)$, $(3, A)$,....

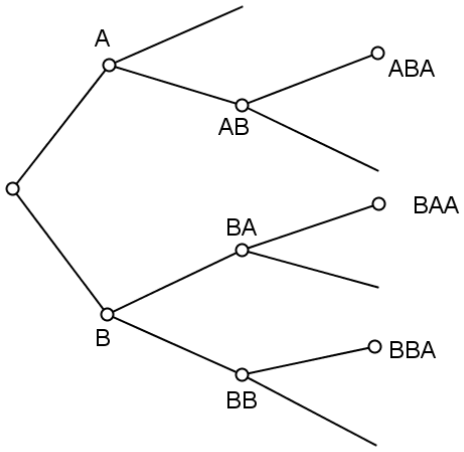


Figura 4.1: Árvore das frases.

Envio de um caractere descompactado em cada frase resulta em uma perda de eficiência. É possível contornar esta situação, considerando o caractere de extensão (como é chamado o último caractere da frase atual), como parte da próxima frase. Esta variação, devido à Welch [9], é a base da maioria das implementações práticas de LZ78, como compressão em Unix, na compressão em modems, e nos arquivos de imagem em formato GIF. Essa variação ficou conhecida como LZW.

Uma das vantagens do algoritmo LZ78 em relação ao LZ77 é que frases frequentes podem ser codificadas mais eficientemente, pois no algoritmo LZ77 as frases só permanecem no dicionário durante um tempo limitado e, normalmente, curto.

Neste algoritmo não existem restrições, a principio, quanto à distância (em termos temporais) a que se podem fazer referências a sequências passadas.

Contudo, o dicionário não pode crescer indefinidamente, por isso, é necessário algum mecanismo que permita a eliminação de sequências pouco referenciadas.

4.2.4 Otimalidade de LZ78

Consideremos agora a versão árvore-estruturada de Lempel-Ziv, onde a sequência de entrada é analisada em frases, cada frase sendo a sequência de caracteres mais curta que ainda não foi vista até aquele momento.

A prova da otimalidade deste algoritmo é muito diferente da prova para LZ77, a essência da prova é um argumento de contagem que mostra que o número de frases não pode ser muito grande se elas forem todas diferentes, e que a probabilidade de qualquer sequência de símbolos pode ser limitada por uma função do número de frases distintas na análise da sequência.

O algoritmo descrito em 4.2.3 requer duas passagens sobre a sequência - na primeira passagem, analisamos a sequência e calculamos $c(n)$, o número de frases na sequência analisada. Em seguida, usamos isso para decidir quantos bits $[\log c(n)]$ para colocarmos os ponteiros no algoritmo. Na segunda passagem, calculamos os ponteiros e produzimos a sequência de codificação, tal como indicado acima.

O algoritmo pode ser modificado sendo necessário apenas uma passagem sobre a sequência e também utilizando menos bits para os ponteiros iniciais. Essas modificações não afetam a eficiência assintótica do algoritmo. Alguns dos detalhes de implementação são discutidos por Welch [10] e Bell [11].

Vamos mostrar que, assim como a versão de janela deslizante de Lempel-Ziv, este algoritmo atinge assintoticamente a taxa de entropia para a fonte ergódica. Primeiro, definimos uma análise da sequência para ser uma decomposição em frases.

Definição 4.4. *Uma análise S de uma sequência binária $x_1x_2\ldots x_n$ é uma divisão da sequência em frases, separadas por vírgulas. Uma* **análise distinta** *é uma análise de tal forma que não existam duas frases idênticas.*

Exemplo 4.3. $0, 111, 1$ é uma análise distinta de 01111, mas $0, 11, 11$ é uma análise que não é distinta.

O algoritmo LZ78 descrito acima dá uma análise distinta da sequência fonte. Sendo $c(n)$ o número de frases na análise LZ78 de uma sequência de comprimento n, então, $c(n)$ depende da sequência X^n.

A sequência comprimida (depois da aplicação do algoritmo Lempel-Ziv) consiste de uma lista de $c(n)$ pares de números, cada par consistindo de um ponteiro que aponta para a ocorrência anterior do prefixo da frase e o último bit da frase.

Cada ponteiro requer $\log c(n)$ bits, e, por conseguinte, o comprimento total da sequência comprimida é $c(n)[\log c(n) + 1]$ bits.

Vamos agora mostrar que $\frac{c(n)(\log c(n)+1)}{n} \longrightarrow H(X)$ para uma sequência ergódica estacionária $X_1, X_2, \ldots, X_n$.

Nossa prova é baseada na prova simples de otimização assintótica de codificação LZ78 devido a Wyner e Ziv [12].

Antes de passarmos para os detalhes da prova, fornecemos um resumo das principais ideias. O primeiro lema mostra que o número de frases em uma análise distinta de uma sequência é menor do que $n/\log n$, o principal argumento da prova se baseia no fato de não haver frases curtas distintas suficientes. Este limite é válido para qualquer análise distinta da sequência, e não apenas a análise LZ78.

A segunda ideia-chave é um limite sobre a probabilidade de uma sequência com base no número de frases distintas. Para ilustrar isso, considere uma sequência de variáveis aleatórias i.i.d. X_1, X_2, X_3, X_4 que pode assumir quatro valores possíveis, $\{A, B, C, D\}$, com probabilidades p_A, p_B, p_C e p_D , respectivamente. Considere agora a probabilidade de uma sequência $P(D, A, B, C) = p_D p_A p_B p_C$ (visto que as variáveis aleatórias são i.i.d.). Uma vez que $p_A + p_B + p_C + p_D = 1$, o produto $p_D p_A p_B p_C$ é maximizado quando as probabilidades são iguais (isto é, o valor máximo da probabilidade de uma sequência de quatro símbolos distintos é de $1/256$) . Por outro lado , se nós considerarmos uma sequência A, B, A, B, a probabilidade desta sequência é maximizada se $p_A = p_B = \frac{1}{2}, p_C = p_D = 0$, e a probabilidade máxima para A, B, A, B é $\frac{1}{16}$. Uma sequência da forma A, A, A, A, poderia ter uma probabilidade de 1. Todos estes exemplos ilustram um ponto básico - sequências com um grande número de símbolos (ou frases) distintos não pode ter uma grande probabilidade. Desigualdade de Ziv (Lema 4.6) é a extensão dessa ideia ao caso Markov, onde os símbolos distintos são as frases da análise distinta da sequência de origem.

Uma vez que o comprimento de uma sequência descrito, após a análise, cresce à medida que $c \log c$, as sequências que têm poucas frases distintas podem ser comprimidas de forma eficiente e correspondem às sequências que poderiam ter uma alta probabilidade.

Por outro lado, as sequências que têm um grande número de frases distintas não compactam bem, mas a probabilidade dessas sequências não pode ser muito grande pela desigualdade de Ziv.

Assim, a desigualdade de Ziv nos permite conectar o logaritmo da probabilidade da sequência com o número de frases na sua análise, e este é por fim utilizado para mostrar que o algoritmo árvore-estruturada Lempel-Ziv é assintoticamente ótimo.

Vamos primeiro demonstrar alguns lemas que necessitamos para a prova do teorema. O primeiro é um limite para o número de frases possíveis em uma análise distinta de uma sequência binária de comprimento n.

Lema 4.4. *(Lempel e Ziv [5]) O número de frases $c(n)$ em uma análise distinta de uma sequência binária $X_1, X_2, \ldots, X_n$ satisfaz*

$$c(n) \leq \frac{n}{(1 - \epsilon_n) \log n}, \tag{4.25}$$

onde $\epsilon_n = min\left\{1, \dfrac{\log(\log n)+4}{\log n}\right\} \longrightarrow 0$ *quando* $n \to \infty$ *e* $\log = \log_2$.

Demonstração. Seja

$$n_k = \sum_{j=1}^{k} j2^j = (k-1)2^{k+1} + 2 \tag{4.26}$$

sendo a soma dos comprimentos de todas as sequências distintas de comprimento inferior ou igual a k. O número de frases c em uma análise distinta de uma sequência de comprimento n é maximizado quando todas as frases são tão curtas quanto possível. Se $n = n_k$, isto ocorre quando as frases são de comprimento $\leq k$, e assim

$$c(n_k) \leq \sum_{j=1}^{k} 2^j = 2^{k+1} - 2 < 2^{k+1} \leq \frac{n_k}{k-1}. \tag{4.27}$$

Se $n_k \leq n < n_{k+1}$, podemos escrever $n = n_k + \triangle$, onde $\triangle < (k+1)2^{k+1}$. Em seguida, a análise em frases mais curtas tem cada uma das frases de comprimento $\leq k$ e $\triangle/(k+1)$ frases de comprimento $k+1$. Assim,

$$c(n) \leq \frac{n_k}{k-1} + \frac{\triangle}{k+1} \leq \frac{n_k + \triangle}{k-1} = \frac{n}{k-1}. \tag{4.28}$$

Agora limitamos o tamanho de k por um dado n. Temos $n_k \leq n < n_{k+1}$. Então

$$n \geq n_k = (k-1)2^{k+1} + 2 \geq 2^k,$$

e, por conseguinte

$$k \leq \log n. \tag{4.29}$$

Além disso,

$$n \leq n_{k+1} = k2^{k+2} + 2 \leq (k+2)2^{k+2} \leq (\log n + 2)2^{k+2},$$

por (4.29), temos

$$k + 2 \geq \log \frac{n}{\log n + 2},$$

ou para todo $n \geq 4$,

$$\begin{aligned} k - 1 &\geq \log n - \log(\log n + 2) - 3 \\ &= \left(1 - \frac{\log(\log n + 2) + 3}{\log n}\right) \log n \\ &\geq \left(1 - \frac{\log(2\log n) + 3}{\log n}\right) \log n \\ &= \left(1 - \frac{\log(\log n) + 4}{\log n}\right) \log n \\ &= (1 - \epsilon_n) \log n. \end{aligned} \tag{4.30}$$

Note que $\epsilon_n = min\left\{1, \dfrac{\log(\log n)+4}{\log n}\right\}$. Combinando (4.30) com (4.28), provamos o lema. □

Vamos precisar da propriedade de entropia máxima para a prova do teorema principal.

Para o próximo resultado, considere o seguinte problema: maximizar a entropia $h(f)$ ao longo de todas as densidades de probabilidade f satisfazendo

1. $f(x) \geq 0$, com igualdade fora do conjunto suporte S
2. $\int_S f(x)dx = 1$
3. $\int_S f(x)r_i(x)dx = \alpha_i$ para $1 \leq i \leq m$.

Assim, f é uma densidade no conjunto suporte S satisfazendo certas condições $\alpha_1, \alpha_2, \ldots, \alpha_m$.

Seja $f^*(x) = f_\lambda(x) = e^{\lambda_0 + \sum_{i=1}^m \lambda_i r_i(x)}$, $x \in S$, em que $\lambda_0, \ldots, \lambda_m$ são escolhidos de modo que f^* satisfaça *1,2* e *3*.

Teorema 4.5. *(Distribuição de máxima entropia) Nas condições acima f^* maximiza exclusivamente $h(f)$ sobre todas as densidades de probabilidade f satisfazendo as condições 1,2,3.*

Demonstração. Vide página 410, de [7].

Lema 4.5. *Se Z é uma variável aleatória não negativa de valores inteiros com média μ. Então a entropia $H(Z)$ é limitada por*

$$H(Z) \leq (\mu + 1)\log(\mu + 1) - \mu \log \mu.$$

Demonstração. O lema segue diretamente a partir dos resultados do Teorema 4.5, que mostram que a distribuição geométrica maximiza a entropia de uma variável aleatória não negativa de valores inteiros sujeita a uma média constante. □

Seja $\{X_i\}_{i=-\infty}^{\infty}$ um processo estacionário e ergódico com função densidade de probabilidade $P(x_1, x_2, \ldots, x_n)$. Para um inteiro k fixado, definimos a aproximação Markov de k-ésima ordem para P como

$$Q_k(x_{-(k-1)}, \ldots, x_0, x_1, \ldots, x_n) \stackrel{\triangle}{=} P(x_{-(k-1)}^0) \prod_{j=1}^{n} P(x_j \mid x_{j-k}^{j-1}),$$

onde $x_i^j \stackrel{\triangle}{=} (x_i, x_{i+1}, \ldots, x_j)$, $i \leq j$, e o estado inicial $x_{-(k-1)}^0$ é parte das especificações de Q_k. Como $P(X_n \mid X_{n-k}^{n-1})$ é em si um processo ergódico, temos

$$-\frac{1}{n}\log Q_k(X_1, X_2, \ldots, X_n \mid X_{-(k-1)}^0) = -\frac{1}{n}\sum_{j=1}^{n} \log P(X_j \mid X_{j-k}^{j-1})$$

$$\to -E[\log P(X_j \mid X_{j-k}^{j-1})]$$
$$= H(X_j \mid X_{j-k}^{j-1}).$$

Teremos limitado a taxa do código LZ78 pela taxa de entropia da aproximação Markov de k-ésima ordem para todo k.

A taxa de entropia da aproximação Markov $H(X_j \mid X_{j-k}^{j-1})$ converge para a taxa de entropia do processo quando $k \to \infty$, e isto prova o resultado.

Suponhamos que $X_{-(k-1)}^{n} = x_{-(k-1)}^{n}$, e também que x_1^n seja analisado em c frases distintas, $y_1, y_2, \ldots, y_c$.

Seja v_i o índice do início da i-ésima frase (isto é, $y_i = x_{v_i}^{v_i+1-1}$).

Para cada $i = 1, 2, \ldots, c$, definimos $s_i = x_{v_i-k}^{v_i-1}$. Assim, s_i representa os k bits de x precedentes de y_i, veja que, $s_1 = x_{-(k-1)}^{0}$.

Seja c_{ls} o número de frases y_i com comprimento l e precedente $s_i = s$ para $l = 1, 2, \ldots$ e $s \in \mathcal{X}^k$. Então,

$$\sum_{l,s} c_{ls} = c \tag{4.31}$$

e

$$\sum_{l,s} l c_{ls} = n \tag{4.32}$$

Mostraremos agora um surpreendente limitante superior para a probabilidade de uma sequência baseada na análise da sequência, a saber

Lema 4.6. *(Desigualdade de Ziv) Para qualquer análise distinta (em particular, a análise LZ78) da sequência $x_1 x_2 \ldots x_n$, temos*

$$\log Q_k(x_1, x_2, \ldots, x_n \mid s_1) \leq -\sum_{l,s} c_{ls} \log c_{ls}.$$

Note que o lado direito da desigualdade não depende de Q_k.

Demonstração. Temos

$$Q_k(x_1, x_2, \ldots, x_n \mid s_1) = Q_k(y_1, y_2, \ldots, y_c \mid s_1)$$
$$= \prod_{i=1}^{c} P(y_i \mid s_i)$$

ou

$$\log Q_k(x_1, x_2, \ldots, x_n \mid s_1) = \sum_{i=1}^{c} \log P(y_i \mid s_i)$$
$$= \sum_{l,s} \sum_{i:|y_i|=l, s_i=s} \log P(y_i \mid s_i)$$
$$= \sum_{l,s} c_{ls} \sum_{i:|y_i|=l, s_i=s} \frac{1}{c_{ls}} \log P(y_i \mid s_i)$$

$$\leq \sum_{l,s} c_{ls} \log \left(\sum_{i:|y_i|=l,s_i=s} \frac{1}{c_{ls}} P(y_i \mid s_i) \right),$$

onde a desigualdade segue a desigualdade de Jensen e da concavidade do algoritmo.

Agora, já que as y_i são distintas, nós temos $\sum_{i:|y_i|=l,s_i=s} P(y_i \mid s_i) \leq 1$. Assim,

$$\log Q_k(x_1, x_2, \ldots, x_n \mid s_1) \leq \sum_{l,s} c_{ls} \log \frac{1}{c_{ls}},$$

provando assim o lema. □

Agora podemos provar o teorema principal sobre a otimalidade de LZ78.

Teorema 4.6. *Seja $\{X_n\}$ um processo estacionário ergódico binário com taxa de entropia $H(\mathcal{X})$, e seja $c(n)$ o número de frases de uma análise distinta de uma amostra de tamanho n a partir deste processo. Então*

$$\limsup_{n\to\infty} \frac{c(n) \log c(n)}{n} \leq H(\mathcal{X})$$

com probabilidade 1.

Demonstração. Começamos com a desigualdade de Ziv, que reescrevemos como

$$\log Q_k(x_1, x_2, \ldots, x_n \mid s_1) \leq -\sum_{l,s} c_{ls} \log \frac{c_{ls} c}{c}$$

$$= -c \log c - c \sum_{ls} \frac{c_{ls}}{c} \log \frac{c_{ls}}{c}.$$

Escrevendo $\pi_{ls} = \frac{c_{ls}}{c}$, temos

$$\sum_{l,s} \pi_{ls} = 1, \qquad \sum_{l,s} l\pi_{ls} = \frac{n}{c},$$

de (4.31) e (4.32).

Vamos agora definir as variáveis aleatórias U, V tais que

$$Pr(U = l, V = s) = \pi_{ls}.$$

Assim, $E(U) = \frac{n}{c}$ e

$$\log Q_k(x_1, x_2, \ldots, x_n \mid s_1) \leq cH(U, V) - c \log c$$

ou

$$-\frac{1}{n} \log Q_k(x_1, x_2, \ldots, x_n \mid s_1) \geq \frac{c}{n} \log c - \frac{c}{n} H(U, V).$$

Agora

$$H(U, V) \leq H(U) + H(V)$$

e $H(V) \leq \log |\mathcal{X}|^k = k$. Pelo Lema 4.5, temos

$$\begin{aligned} H(U) &\leq (E(U)+1)\log(E(U)+1) - (E(U))\log(E(U)) \\ &= \left(\frac{n}{c}+1\right)\log\left(\frac{n}{c}+1\right) - \frac{n}{c}\log\frac{n}{c} \\ &= \log\frac{n}{c} + \left(\frac{n}{c}+1\right)\log\left(\frac{c}{n}+1\right). \end{aligned}$$

Assim,

$$\frac{c}{n}H(U,V) \leq \frac{c}{n}k + \frac{c}{n}\log\frac{n}{c} + o(1).$$

Para um dado n, o máximo de $\frac{c}{n}\log\frac{n}{c}$ é alcançado para o valor máximo de c (para $\frac{c}{n} \leq \frac{1}{e}$). Mas do Lema 4.4, $c \leq \frac{n}{\log n}(1+o(1))$. Assim,

$$\frac{c}{n}\log\frac{n}{c} \leq O\left(\frac{\log\log n}{\log n}\right),$$

e, portanto $\frac{c}{n}H(U,V) \longrightarrow 0$ visto que $n \to \infty$. Por consequência,

$$\frac{c(n)\log c(n)}{n} \leq -\frac{1}{n}\log Q_k(x_1, x_2, \ldots, x_n \mid s_1) + \epsilon_k(n),$$

onde $\epsilon_k(n) \to 0$ enquanto $n \to \infty$.

Portanto com probabilidade 1,

$$\begin{aligned} \limsup_{n\to\infty} \frac{c(n)\log c(n)}{n} &\leq \lim_{n\to\infty} -\frac{1}{n}\log Q_k(X_1, X_2, \ldots, X_n \mid X_{-(k-1)}^0) \\ &= H(X_0 \mid X_{-1}, \ldots, X_{-k}) \end{aligned}$$

Mas $H(X_0 \mid X_{-1}, \ldots, X_{-k}) \longrightarrow H(\mathcal{X})$, quando $k \to \infty$.

Portanto, $\limsup_{n\to\infty} \frac{c(n)\log c(n)}{n} \leq H(\mathcal{X})$ □

No próximo resultado provamos que a codificação LZ78 é assintoticamente ideal.

Teorema 4.7. *Se $\{X_i\}_{-\infty}^{\infty}$ é um processo estocástico ergódico estacionário binário e $l(X_1, X_2, \ldots, X_n)$ é o comprimento da palavra código de LZ78 associado com $X_1, X_2, \ldots, X_n$. Então*

$$\limsup_{n\to\infty} \frac{1}{n} l(X_1, X_2, \ldots, X_n) \leq H(\mathcal{X}) \text{ com probabilidade 1,}$$

em que $H(\mathcal{X})$ é a taxa de entropia do processo.

Demonstração. Mostramos que o $l(X_1, X_2, \ldots, X_n) = c(n)(\log c(n) + 1)$, onde $c(n)$ é o número de frases na análise LZ78 da sequência $X_1, X_2, \ldots, X_n$.

Pelo Lema 4.4, $\limsup c(n)/n = 0$, e, assim, o Teorema 4.6 estabelece que

$$\begin{aligned} \limsup \frac{l(X_1, X_2, \ldots, X_n)}{n} &= \limsup\left(\frac{c(n)\log c(n)}{n} + \frac{c(n)}{n}\right) \\ &\leq H(\mathcal{X}), \text{ com probabilidade 1.} \end{aligned}$$

□

Assim, o comprimento por símbolo fonte da codificação LZ78 de uma fonte ergódica é assintoticamente não maior do que a taxa de entropia da fonte.

Existem algumas características interessantes da prova da otimização de LZ78, por exemplo, os limites sobre o número de frases distintas e a desigualdade de Ziv se aplicam a qualquer análise distinta da sequência de caracteres, e não apenas a versão de análise usada no algoritmo. A prova pode ser estendida em muitas formas com variações sobre o algoritmo de análise. Desigualdade de Ziv (Lema 4.6) permanece particularmente intrigante, uma vez que se relaciona a probabilidade de um lado, com uma função puramente determinística da análise de uma sequência de outro.

Os códigos Lempel-Ziv são exemplos simples de um código universal (isto é, um código que não depende da distribuição da fonte). Este código pode ser usado, sem o conhecimento da distribuição da fonte e ainda alcançará uma compressão assintóticamente igual à taxa de entropia da fonte.

Referências

[1] SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, p. 379–423, 623–656, 1948.

[2] HUFFMAN, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the I.R.E.*, p. 1098–1102, 1952.

[3] HAYKIN, S. *Digital Communications*. 1. ed. New York: Wiley, 1988.

[4] ZIV, J.; LEMPEL, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, v. 23, n. 3, p. 337–343, 1977.

[5] ZIV, J.; LEMPEL, A. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, v. 24, n. 5, p. 530–536, 1978.

[6] BANDYOPADHYAY, G. A simple proof of the decipherability criterion of sardinas and patterson. *Information and Control*, v. 6, p. 331–336, 1963.

[7] COVER, T. M.; THOMAS, J. A. *Elements of Information Theory*. 1. ed. New York: Wiley - Interscience, 1991.

[8] WYNER, A. D.; ZIV, J. The sliding window lempel-ziv algorithm is asymptotically optimal. *Proc. IEEE*, v. 82(6), p. 872–877, 1994.

[9] WELCH, T. A. A technique for high-performance data compression. *Computer*, v. 17, p. 8–19, 1984.

[10] WELCH, T. A. A technique for high-performance data compression. *Computer*, v. 17(1), p. 8–19, 1984.

[11] BELL, T. C.; CLEARY, J. G.; WITTEN, I. H. *Text Compression*. 1. ed. NJ: Prentice-Hall,Englewood Cliffs, 1990.

[12] WYNER, A.; ZIV, J. On entropy and data compression. *IEEE Trans. Inf. Theory*, 1991.

[13] STORER, J. A.; SZYMANSKI, T. G. Data compression via textual substitution. *J. ACM*, v. 29(4), p. 928–951, 1982.

[14] BIRKHOFF, G.; BARTEE, T. C. *Modern Applied Algebra*. 1. ed. New York: McGrow-Hill, 1970.

[15] ASH, R. B. *Information Theory*. New York: Interscience, 1965.

[16] HERSTEIN, I. N. *Topics In Álgebra*. 2. ed. New York: Wiley, 1975.

[17] SINGH, S. *O livro dos códigos*. São Paulo: Editora Record, 2004.