Machine Translated by Google

Algumas notas para o curso

INFORMAÇÃO E CODIFICAÇÃO

Armando J. Pinho

Nota importante: Este documento contém um resumo de alguns dos temas abordados no curso "Informação e Codificação". Como tal, deve ser considerado um guia e um complemento, e de forma alguma a única fonte de informação nem um substituto das aulas.

RASCUNHO

(Versão preliminar de 13 de setembro de 2023)

Machine Translated by Google

Machine Translated by Google

Conteúdo

1 Alguns conceitos de processamento digital de sinais **7**

1.1 Sinais 7

1.1.1 Algumas características dos sinais. 9

1.1.2 Conversão analógico-digital 10

1.1.3 Sinais multidimensionais 15

1.1.4 Alguns sinais elementares. 17

1.1.5 Operações sobre a variável independente 19

1.2 Sistemas 21

1.2.1 Propriedades dos sistemas: Estabilidade. 22

1.2.2 Propriedades dos sistemas: Memória 23

1.2.3 Propriedades dos sistemas: Causalidade. 24

1.2.4 Propriedades dos sistemas: Invertibilidade. 24

1.2.5 Propriedades dos sistemas: Linearidade 25

1.2.6 Propriedades dos sistemas: Invariância temporal. 25

1.3 Sistemas lineares e invariantes no tempo. 26

1.3.1 Representação de sinais por superposição de impulsos. 26

1.3.2 Resposta ao impulso. 27

1.3.3 Propriedades da operação de convolução. 28

1.3.4 A resposta ao impulso e algumas propriedades dos sistemas. 29

1.3.5 Resposta a exponenciais complexas. 31

1.4 A transformada discreta de Fourier (DFT). 33

1.4.1 Algumas propriedades da transformada discreta de Fourier 35

1.4.2 A transformada discreta de cosseno 38

1.5 A transformada Z e filtros digitais 38

1.5.1 A propriedade time-shift. 39

1.5.2 Estabilidade do sistema 42

1.5.3 Resposta em frequência 43

2 Dados, informações e alguns problemas relacionados **45**

3 Uma medida combinatória de informação **49**

Machine Translated by Google

4 Uma medida probabilística de informação **55**

4.1 Sobre códigos 56

4.2 Definição de informação segundo Shannon. 61

4.3 Modelagem de origem 64

4.4 Um pouco mais sobre a entropia de Shannon. 70

4.4.1 Entropia, entropia conjunta e entropia condicional. 70

4.4.2 Entropia relativa. 73

4.4.3 Informação mútua 74

5 Compressão de dados **81**

5.1 Codificação de comprimento variável. 81

5.1.1 Códigos ótimos 81

5.1.2 Um argumento de contagem. 83

5.1.3 Códigos de Huffman 83

5.1.4 Códigos Shannon-Fano 85

5.1.5 Extensão do alfabeto 86

5.1.6 Alguns outros códigos de comprimento variável 88

5.1.7 Código Golomb 90

5.2 Compressão baseada em dicionário. 91

5.2.1 Códigos de instalação 92

5.2.2 Análise de strings. 95

5.2.3 Compressão LZ77 95

5.2.4 Compressão LZSS 98

5.2.5 Compressão LZ78 98

5.2.6 Compressão LZW 100

5.3 Codificação aritmética. 103

5.3.1 Motivação 103

5.3.2 Codificação 103

5.3.3 Decodificação 105

5.3.4 Questões de implementação 106

6 Codificação preditiva **109**

6.1 Princípios 109

Machine Translated by Google

6.2 Técnicas de codificação preditiva. 110
6.2.1 Predição 1D simples. 110
6.2.2 Técnicas preditivas de codificação de imagens. 110
6.2.3 Preditores 110
6.3 Codificação preditiva sem perdas. 112
6.3.1 Predição linear: o modo sem perdas do JPEG. 113
6.3.2 O preditor não linear de JPEG-LS. 114
6.4 Previsão temporal e compensação de movimento. 114

7 Uma medida algorítmica de informação **119**
7.1 Motivação 119
7.2 Máquinas de Turing 120
7.3 Complexidade de Kolmogorov (ou entropia algorítmica) 125

Referências **129**

Machine Translated by Google

Machine Translated by Google

1 Alguns conceitos de processamento digital de sinais

1.1 Sinais

Em termos muito gerais, um **sinal** é uma representação de uma certa quantidade que varia ao longo do tempo e/ou espaço ou outras variáveis de interesse. Por exemplo, a voz de uma pessoa pode ser representada como um sinal de tempo que mede mudanças na pressão do ar.

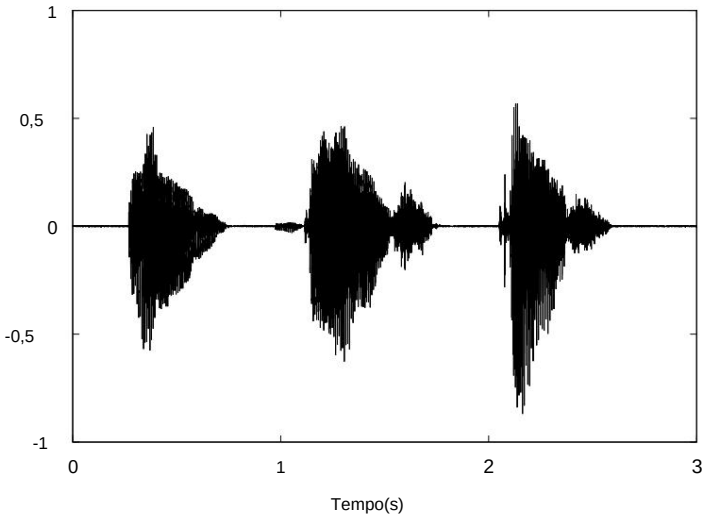


Figura 1.1: Sinal de áudio correspondente às palavras em português “um dois tres”.

Este sinal de pressão pode ser captado por um microfone e convertido em sinal elétrico (sinal de áudio), amplificado e enviado para um alto-falante, que transforma o áudio novamente em som sinais. A Figura 1.1 mostra um pequeno exemplo de sinal de áudio, com duração de 3 segundos, correspondendo às palavras portuguesas “um dois tres” (“um dois três”).

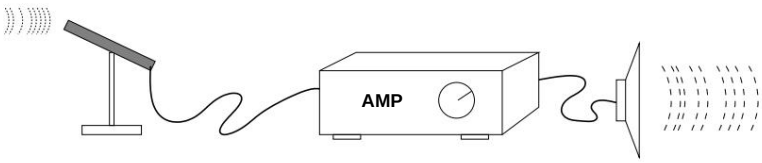


Figura 1.2: Sistema de captação, amplificação e reprodução sonora.

Os sinais são transformados por **sistemas.** No exemplo anterior, podemos identificar claramente três sistemas (Fig. 1.2): um microfone (converte variações audíveis da pressão do ar em elétrica sinais), um amplificador (aumenta a energia do sinal captado pelo microfone) e um alto-falante (converte o sinal de áudio amplificado em ondas sonoras).

Na Figura 1.3 podemos ver outro exemplo de sistema, bem como sinais de vários tipos, a saber:

Machine Translated by Google

pressão, elétrica, óptica e eletromagnética. Na verdade, o conceito de sinal é tão vasto que qualquer quantidade dependente de uma ou mais variáveis pode ser considerada um sinal.

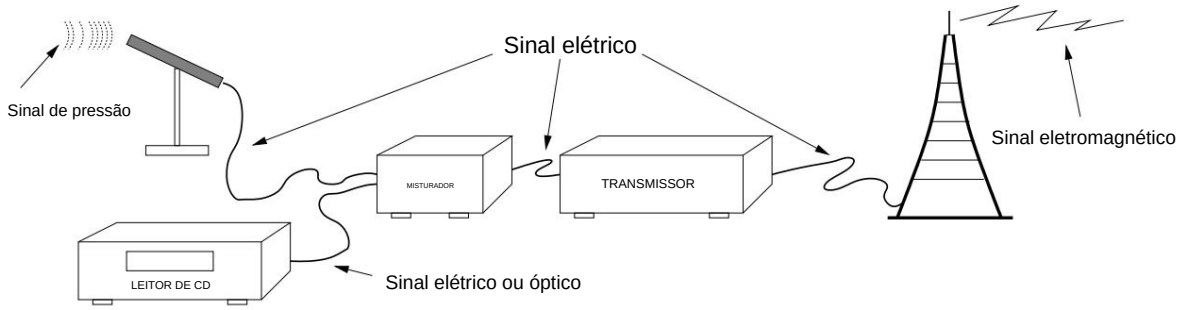


Figura 1.3: Sistema de transmissão de áudio.

Embora um sinal possa depender de diferentes variáveis, é comum utilizar a variável tempo quando queremos representar um sinal genérico. Neste caso, o sinal é representado, por exemplo, por x(t). Portanto, o sinal mostrado na Figura 1.1 poderia ser representado, para fins de manipulação matemática, por x(t). Assim, poderíamos dizer que o valor do sinal no instante t0 é x(t = t0) ou, mais simplesmente, x(t0). Embora esta última notação possa ser confundida com a representação de um sinal completo, em geral o contexto cuida de esclarecer o seu significado.

Um sinal pode ter valores escalares ou vetoriais, reais ou complexos. No exemplo mostrado, o valor de x(t) poderia representar, por exemplo, a tensão elétrica na saída do microfone, caso em que temos x(t) ÿ R. Como tanto t quanto x(t) possuem valores em R, dizemos que x(t) é um sinal analógico, ou seja, tanto a variável de tempo quanto o valor do sinal variam continuamente.
Porém, existem sinais que só possuem valores definidos em determinados instantes de tempo, ti . Estes são chamados de sinais de tempo discreto. Por exemplo, as temperaturas máximas diárias ao longo de um ano formam um sinal de tempo discreto, onde ti são os dias do ano. Neste caso, dizer que x(129) é igual a 34,8 corresponde a dizer que no 129º dia do ano a temperatura máxima foi de 34,8 graus.

Um sinal também pode ter valores discretos. Tomemos, por exemplo, o número de pessoas que estão na fila de um balcão de atendimento. Neste caso, x(t) ÿ N0, ou seja, este sinal possui valores discretos.

Em resumo, temos as seguintes quatro combinações:

- Sinais contínuos tanto no tempo quanto na amplitude x(t) ÿ R
  ou C, t ÿ R
  Exemplo: tensão na saída de um microfone
- Sinais contínuos no tempo e discretos em amplitude x(t) ÿ {. . . , xiÿ1,
  xi , xi+1, . . . } ÿ R ou C, t ÿ R Exemplo: número de pessoas
  em uma fila
- Sinais de tempo discreto e amplitude contínua x(n) ÿ R ou C, n
  ÿ Z
  Exemplo: temperatura máxima diária

Machine Translated by Google

- **Sinais discretos em tempo e amplitude** x(n) ÿ
  {. . . , xiÿ1, xi , xi+1, . . . }ÿ R ou C, n ÿ Z Exemplo: número
  de pessoas atendidas por hora por um serviço

Sinais do último tipo (discretos em tempo e amplitude) são particularmente interessantes para lidar com um computador. Por um lado, um computador possui células de memória discretas, que podem ser usadas para armazenar valores de um sinal em determinados instantes de tempo (tempo discreto). Por outro lado, devido à precisão finita com que os valores podem ser armazenados em um computador, é muito conveniente que o sinal só possa ter valores dentro de um conjunto discreto e finito (amplitude discreta). Como os sinais discretos no tempo e na amplitude são os mais adequados para processamento em um computador digital, nós os chamamos de **sinais digitais.**

Também consideramos sinais com valores que não são numéricos. Nós os chamamos **de sinais simbólicos.** Em geral, os valores dessas sequências de tempo discreto (às vezes chamadas de "séries temporais categóricas") não possuem uma estrutura matemática, além de serem membros de algum conjunto. Observe que embora todos os sinais digitais (numéricos) possam ser vistos como sinais simbólicos, o inverso nem sempre é verdadeiro. No entanto, a maioria dos métodos de compressão de dados que abordaremos mais tarde assumem que os sinais são simbólicos.

1.1.1 Algumas características dos sinais

Além das características de tempo contínuo/tempo discreto, amplitude contínua/amplitude discreta e numérica/simbólica, um sinal também pode ser classificado de acordo com diversas outras propriedades, incluindo

- Seja determinístico ou estocástico
- No caso de ser determinístico, seja periódico ou não
- No caso de ser estocástico, seja estacionário ou não
- Seja em tempo finito ou em tempo infinito

Os **sinais determinísticos** são gerados através de algoritmos que não possuem aleatoriedade e, portanto, são previsíveis. Por exemplo, o sinal x(t) = sin(t) é determinístico. Isto significa que, por exemplo, o gráfico deste sinal para 0 ÿ t ÿ 4ÿ é sempre o mesmo, como mostra a Figura 1.4.

Por outro lado, **os sinais estocásticos** são gerados por sistemas que possuem aleatoriedade, ou seja, cujo valor em um determinado instante de tempo não pode ser determinado com exatidão a priori. A saída do sinal de áudio de um microfone é um exemplo de sinal estocástico. Na verdade, apesar de muitas vezes uma palavra ser repetida diante de um microfone, o sinal gerado será sempre diferente. Apesar disso, algumas características destes sinais podem ser utilizadas, por exemplo, para realizar reconhecimento de voz.

Machine Translated by Google

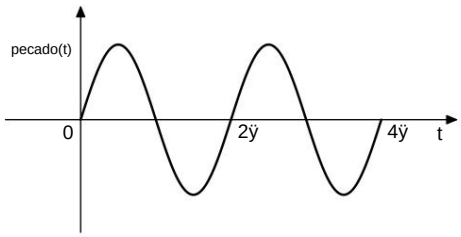


Figura 1.4: Exemplo de sinal determinístico.

Um sinal determinístico pode ser periódico, ou seja, pode se repetir exatamente após um determinado intervalo de tempo, denominado período . No caso de sinais de tempo contínuo, temos x(t) = x(t+T), ÿt, onde T é o período. Obviamente, se x(t) é periódico com período T, então também é periódico para períodos kT com k ÿ N. O menor valor de T, para o qual a relação x(t) = x(t + T), ÿt , é válido, é chamado de **período fundamental.** Salvo indicação em contrário, sempre que nos referimos ao período de um sinal estamos nos referindo ao seu período fundamental. Para tempo discreto, o período é medido em número de amostras, N, ou seja, um sinal x(n) é periódico com período N se x(n) = x(n + N), ÿn.

Um sinal estocástico pode ser **estacionário,** o que significa que a sua caracterização estatística é mantida ao longo do tempo.1 Por exemplo, isto pode indicar que a média e a variância do sinal permanecem constantes ao longo do tempo. Contudo, os sinais mais interessantes não são estacionários. A voz humana é um exemplo. No entanto, quando analisados em intervalos de tempo suficientemente pequenos, podemos muitas vezes considerar que a estacionariedade é verificada.

Um sinal tem **duração finita** se começa em um determinado ponto no tempo ti > ÿÿ e termina em outro ponto no tempo, tf , tal que ti < tf < +ÿ. Caso isso não aconteça, dizemos que o sinal tem **duração infinita.**

1.1.2 Conversão analógico-digital

Existem muitos sinais que gostaríamos de poder processar em um computador, mas não são sinais digitais. Veja o exemplo do sinal de áudio captado por um microfone. Neste caso, podemos recorrer à **conversão analógico-digital** para transformar sinais de tempo contínuo e amplitude contínua em sinais digitais. A conversão analógico-digital envolve duas operações: **amostragem** e **quantização.**

A **operação de amostragem** é responsável pela passagem do tempo contínuo para o tempo discreto. Para este propósito, os valores do sinal são coletados em determinados momentos. Normalmente, esses instantes de tempo são regularmente espaçados de Ts a Ts segundos. Chamamos Ts de **período de amostragem** (Fig. 1.5) e fs = 1/Ts de **frequência de amostragem.** Portanto, quando dizemos que um sinal é amostrado em 500 Hz, estamos indicando que são coletadas 500 amostras para cada segundo do sinal.

Após o sinal x(t) ter sido amostrado de Ts em Ts segundos, temos uma sequência de valores que correspondem aos valores de x(t) nesses instantes de tempo, ou seja, temos a sequência de

1Existem formas estritas de definir a estacionariedade, mas vão além do que queremos apresentar aqui.

Machine Translated by Google

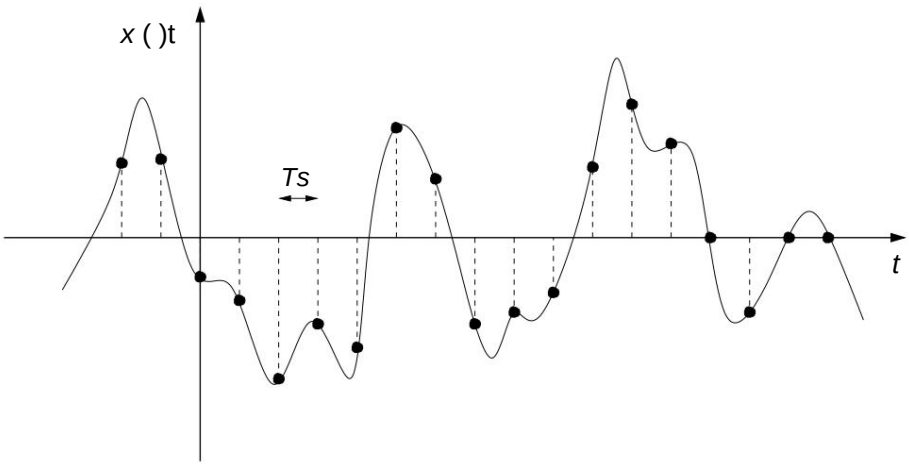


Figura 1.5: Amostragem de sinal.

valores

. . . , x(ÿ2Ta), x(ÿTa), x(0), x(Ta), x(2Ta), . . . , x(nTa), . . . ,

que representamos por x(nTs) ou simplesmente por x(n). [2]

Dependendo de certas propriedades de x(t), é possível encontrar valores de Ts para os quais a sequência x(nTs) descreve completamente x(t). Em outras palavras, nesse caso será possível reconstruir exatamente x(t) com base nas amostras x(nTs).

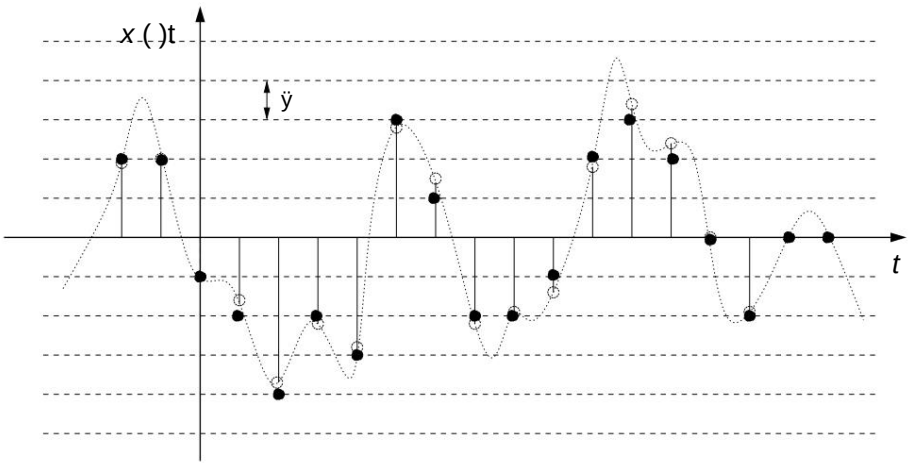


Figura 1.6: Quantização de sinal.

A **operação de quantização** introduz erros (ou **ruído,** de acordo com o processamento do sinal).

2Ao representar a sequência x(nTs) por x(n) estamos simplificando a notação, mas também introduzimos alguma perda de rigor. Na verdade, x(n) designa qualquer sequência de valores, indexada pela variável inteira n, que pode ou não corresponder a uma sequência temporal e, mesmo assim, pode não corresponder a amostras uniformemente espaçadas. No entanto, acreditamos que a simplificação da notação justifica esta imprecisão, embora sempre que haja dúvida, optaremos pela notação completa.

Machine Translated by Google

minologia) nos valores da amostra. Ao contrário da amostragem, que é possível sem introduzir erros, não é possível fazer quantização sem introduzir erros irreversíveis. Porém, é possível calcular valores máximos para o erro cometido e reduzi-lo tanto quanto desejado.
Uma das medidas mais comuns para expressar o erro introduzido é a relação sinal-ruído, que é definida pela expressão

$$SNR = 10 \log10 \frac{Ex}{Er} \text{ dB (decibéis)}, \tag{1.1}$$

onde Ex e Er indicam, respectivamente, a energia do sinal e do ruído, sendo a energia de um sinal em tempo discreto x(n), Ex, calculada através

$$Ex = \sum_{n=ÿÿ}^{ÿ} |x(n)|^2. \tag{1.2}$$

Observe que, conforme (1.2), pode não ser possível calcular a energia de um determinado sinal.
Isso acontece se o respectivo somatório não convergir. Na verdade, existem muitos sinais de energia infinita, sendo um exemplo óbvio a classe dos sinais periódicos. Neste caso podemos calcular a potência do sinal, ou seja, sua energia por unidade de tempo. A potência de um sinal em tempo discreto é dada por

$$Px = \lim_{Nÿÿÿÿ} \frac{1}{2N + 1} \sum_{n = ÿN}^{N} |x(n)|^2. \tag{1.3}$$

Observe que, de acordo com as definições apresentadas, um sinal de energia finita tem potência zero. Esses sinais são chamados de sinais de energia. Por outro lado, um sinal com potência diferente de zero tem necessariamente energia infinita e é chamado de sinal de potência.

No caso de sinais de potência, a relação sinal-ruído definida em (1.1) muda para a forma

$$SNR = 10 \log10 \frac{Px}{Pr.} \text{ dB}, \tag{1.4}$$

onde Px e Pr indicam, respectivamente, a potência do sinal e do ruído.

O cálculo da energia conforme (1.2) ou da potência conforme (1.3), aplica-se aos casos em que o sinal é determinístico, ou então é uma instância de sinal estocástico. No entanto, também podemos estar interessados em determinar a relação sinal-ruído “média” associada a um processo de digitalização de uma determinada classe de sinais estocásticos. Neste caso, teremos que utilizar ferramentas probabilísticas e, em particular, o valor esperado (ou valor médio) de uma variável aleatória, X, que é dado por

$$E[X] = \int_{ÿÿ}^{ÿ} xfX(x)dx, \tag{1.5}$$

onde fX(x) representa a função de densidade de probabilidade de X. [3] Então, o poder de um estocástico

[3] Se uma variável aleatória X é caracterizada por uma função de densidade de probabilidade fX(x), então a probabilidade de que X tem valores entre a e b é dado por $\int_a^b fX(x)dx$.

Machine Translated by Google

sinal é obtido calculando

$$Px = E[X^2] = \int_{ÿÿ}^{ÿ} x^2 fX(x)dx. \quad (1.6)$$

Suponhamos então que um sinal, uniformemente distribuído no intervalo [ÿA/2, A/2], acessado[4] é profissional por um quantizador com b níveis. Se os níveis de quantização estiverem espaçados uniformemente, espaçamento 2 será ÿ = A/2^b então este e o erro máximo de quantização serão rmax = ÿ/2 (Fig. 1.7 (veja um exemplo com 4 níveis).

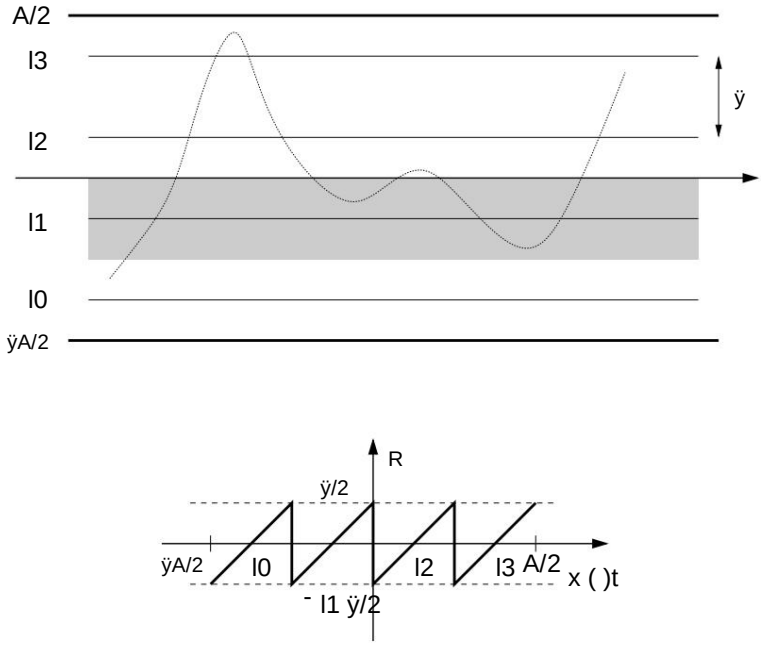


Figura 1.7: Exemplo de quantização em quatro níveis: l0, l1, l2, l3. Observe que todos os valores de sinal que ocorrem dentro da faixa sombreada serão representados pelo valor l1.

Como o sinal é distribuído uniformemente, temos

$$fX(x) = \frac{1}{A}$$

e também

$$fR(r) = ÿ \frac{1}{\ }, $$

já que a variável aleatória associada ao erro de quantização também possui distribuição uniforme.

Então, a potência do ruído de quantização é

$$Pr = E[R^2] = \int_{ÿÿ/2}^{ÿ/2} r^2 \frac{1}{\ } dr = ÿ \frac{ÿ2}{12}$$

4Dizemos que uma variável aleatória, X, está uniformemente distribuída no intervalo SX = [ÿA/2, A/2], se a probabilidade de ela ter valores em um certo subintervalo de SX de comprimento L ÿ A é L /A.

Machine Translated by Google

e a potência do sinal é

$$P_x = E[X^2] = \int_{-A/2}^{A/2} x^2 \frac{1}{A}\,dx = \frac{A^2}{12}.$$

Porque

$$\Delta = \frac{A}{2^b},$$

Nós temos

$$SNR = 10\log_{10}\frac{P_x}{P_r} = 10\log_{10}\frac{A^2}{\Delta^2} \approx 6{,}02b\,dB. \qquad (1.7)$$

A relação (1.7) é bem conhecida e geralmente é utilizada para estimar a relação sinal-ruído obtida ao representar cada amostra do sinal com b bits. Como podemos ver, para cada bit adicional, a relação sinal-ruído aumenta aproximadamente 6 dB. Porém, não podemos esquecer que esta relação foi derivada assumindo a distribuição uniforme (e média zero) do sinal a ser quantizado, característica que muitas vezes não é verificada.

Exemplo 1.1

Deixe um sinal, x(t), ser amostrado em 1 kHz, cujas amostras devem ser armazenadas em 6 bits. Considerando que x(t) possui valores entre −1 e 1, pretendemos determinar uma expressão para os níveis de um quantizador uniforme, de forma a minimizar o erro máximo. Também queremos saber qual será a taxa de dados binários gerada.

O espaçamento entre os níveis do quantizador é $\Delta = \frac{2}{2^6} = \frac{1}{2^5}$. Portanto, os 64 níveis devem ser posicionado em

$$l_k = -1 + k\frac{\Delta}{2} = -1 + \frac{1}{2^5}\frac{k}{2^6}, \quad k = 0, 1, \ldots, 63$$

para que o erro máximo seja minimizado. Neste caso, o erro máximo é $\frac{\Delta}{2} = \frac{1}{2^6}$.

Como 1.000 amostras são geradas por segundo e cada amostra é armazenada em 6 bits, a taxa binária gerada é 6 × 1.000 = 6 kbps (kilobits por segundo).

Note que, genericamente, se o sinal a quantizar tem uma variação entre Amin e Amax, então os níveis do quantizador uniforme são dados por

$$l_k = A_{min} + k\frac{\Delta}{2}, \quad k = 0, 1, \ldots, 2^b - 1,$$

onde

$$\Delta = \frac{A_{max} - A_{min}}{2^b}$$

e b é o número de bits usados para representar cada amostra.

Exemplo 1.2

O sinal de áudio armazenado em um CD é amostrado em 44,1 kHz, sendo cada amostra representada com 16

Machine Translated by Google

pedaços. Pretende-se estimar a relação sinal-ruído resultante do processo de quantização uniforme aplicado ao sinal de áudio, e também calcular a taxa de bits na saída de um CD player.

Se o sinal de áudio fosse distribuído uniformemente, teríamos uma relação sinal-ruído igual a 16 × 6 = 96 dB. Porém, como em geral a distribuição de um sinal de áudio não é uniforme, este valor deve ser entendido meramente como uma estimativa.

A taxa binária na saída do CD player é dada por 44 100×16×2 = 1 411 200 bps (bits por segundo), ou seja, aproximadamente igual a 1,4 Mbps (megabits por segundo). Observe que o fator "×2" resulta do fato do áudio do CD ser estéreo e, portanto, existirem dois canais.

1.1.3 Sinais multidimensionais

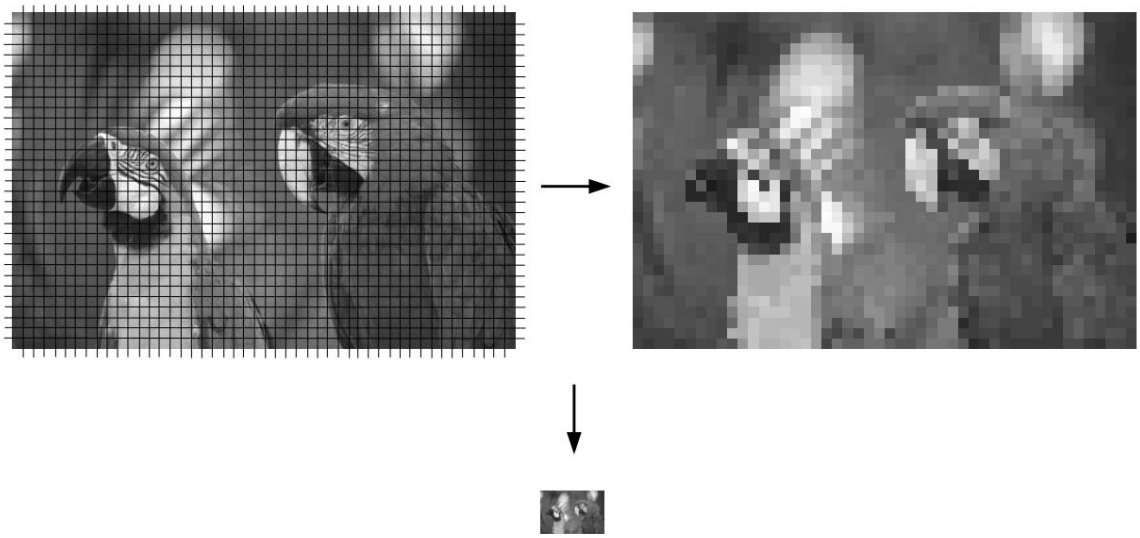


Figura 1.8: Amostragem de imagens.

Diariamente somos confrontados com informação veiculada sob a forma de sinais bidimensionais (imagens) e tridimensionais (vídeo). Uma imagem pode ser vista como um sinal bidimensional, f(x, y), onde xey representam coordenadas espaciais. O valor e o significado de f(x, y) em um determinado ponto do espaço, (x, y), dependem da fonte responsável pela produção da imagem. No entanto, é geralmente assumido que f(x, y) ÿ 0 e que ambos (x, y) e f(x, y) assumem valores contínuos. Portanto, para converter f(x, y) em imagem digital é necessário, por um lado, realizar uma amostragem espacial e, por outro lado, quantizar a amplitude de cada amostra resultante. A Figura 1.8 ilustra o processo de amostragem de uma imagem, enquanto a Figura 1.9 mostra o efeito da quantização.

Uma imagem digital é representada por uma matriz retangular de escalares ou vetores, f(i, j), que chamamos de elementos de imagem ou pixels. Em geral, temos f(i, j) ÿ I ÿ N n 0 , ou seja, cada pixel é um inteiro não negativo ou um vetor de inteiros não negativos.

Machine Translated by Google

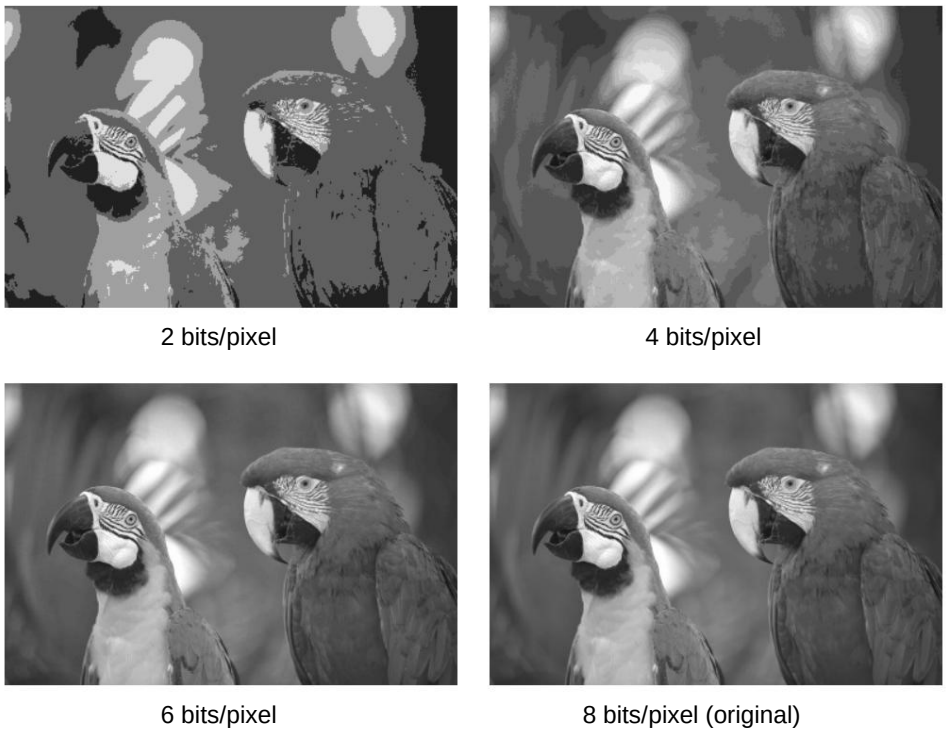


Figura 1.9: Efeitos de quantização em imagens.

Os tipos de imagens mais comuns são (Fig. 1.11 apresenta um exemplo de cada uma dessas quatro tipos):

- **Preto e branco ou binário**
  f(eu, j) ÿ {0, 1}

- **Nível de cinza**
  b f(i, j) ÿ {0, 1, . . . , 2 ÿ1}

- **Cor indexada (índices em um mapa de cores)**
  b f(i, j) ÿ {0, 1, . . . , 2 ÿ 1} ÿÿÿ I ÿ {0, 1, . . . , $2^{b'}$ ÿ1} ^{3}

- **Cor (por exemplo, RGB)**
  b f(i, j) ÿ {0, 1, . . . , 2 ÿ1} ^{3}

A uma sequência temporal de imagens digitais chamamos de **vídeo digital,** que representamos por v(i, j, n) ou por v(i, j, nTs) se quisermos evidenciar o período de amostragem ao longo do tempo. Neste caso, o período Ts é o tempo que decorre entre a apresentação de dois frames de vídeo consecutivos. O inverso de Ts é a **taxa de quadros.**

Para cada valor de n corresponde uma imagem digital, ou seja, podemos escrever f(i, j) = v(i, j, k), k = constante, que denotamos pelo **quadro** k da sequência de vídeo v.

Machine Translated by Google

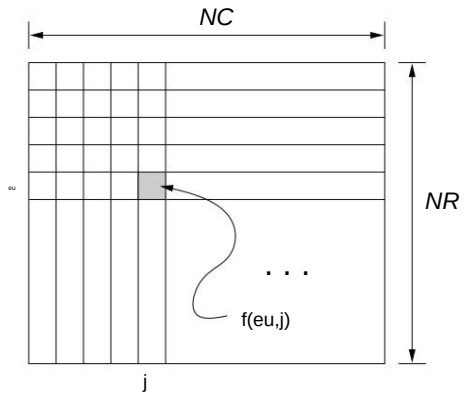


Figura 1.10: Imagem digital, onde NC e NR designam respectivamente o número de colunas e linhas da imagem.

1.1.4 Alguns sinais elementares

Embora a maioria dos sinais da vida real tenham formas complexas, sinais simples são frequentemente usados para estudar sistemas. Existem várias razões para esta abordagem, entre elas a facilidade de fazer manipulações matemáticas, o facto de alguns destes sinais simples se aproximarem de alguns sinais da vida real. sinais relativamente bem, e provavelmente a razão mais importante, porque é possível representar sinais complicados da vida real combinando sinais simples.

Um desses sinais elementares é o **degrau unitário** (Fig. 1.12), definido como

$$\text{você}(n) = \begin{matrix} 0, & n < 0 \\ 1, & n \ddot{y} 0 \end{matrix} \quad . \tag{1.8}$$

O **impulso unitário** (Fig. 1.13) é outro sinal elementar importante e é definido através

$$\ddot{y}(n) = \begin{matrix} 0, & n\ddot{y} = 0 \\ 1, & n = 0 \end{matrix} \quad . \tag{1.9}$$

Os sinais **exponenciais ,**

$$x(n) = \text{uma}^{n}, \qquad \text{uma } \ddot{y} \text{ R}, \tag{1.10}$$

também desempenham um papel importante no processamento de sinais (lembre-se do que acontece, em termos da evolução de x(n), em função de a).

Os sinais **senoidais** também são muito importantes no processamento de sinais. Estas estão relacionadas com **exponenciais complexas,** já que, pela relação de Euler, temos

$$e^{j\ddot{y}} = \cos \ddot{y} + j \text{ sen } \ddot{y}, \tag{1.11}$$

onde j = ÿ ÿ1 é a unidade imaginária.

Deste resultado podem ser tiradas duas relações muito importantes, nomeadamente,

$$\text{porque}\ddot{y} = \frac{e^{j\ddot{y}} + e^{\ddot{y}j\ddot{y}}}{2} \tag{1.12}$$

Machine Translated by Google

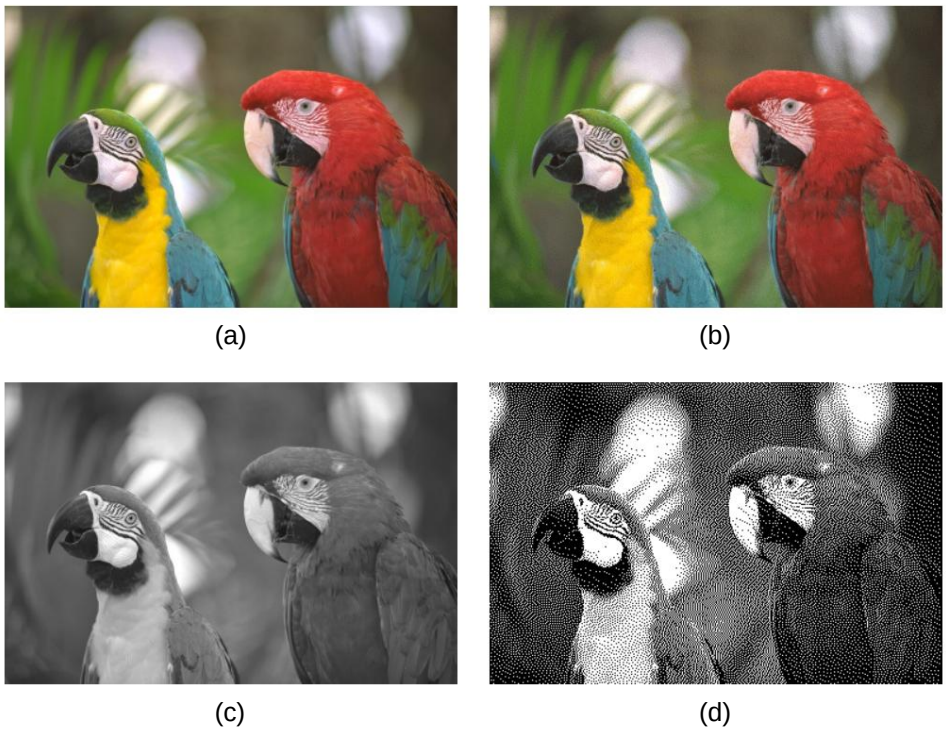


Figura 1.11: Exemplos de imagens digitais comuns: (a) coloridas; (b) cor indexada (256 cores); (c) nível de cinza (256 níveis); (d) binário (2 níveis).

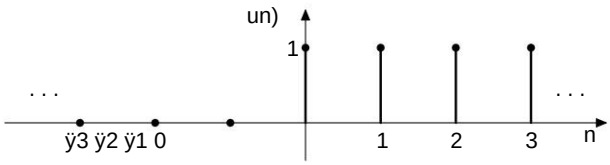


Figura 1.12: Passo unitário de tempo discreto.

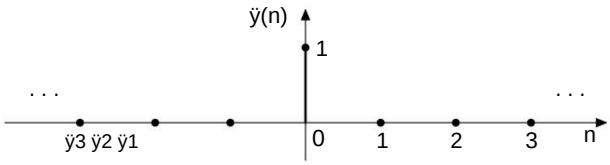


Figura 1.13: Impulso unitário de tempo discreto.

e

$$\text{pecado } ÿ = \frac{ejÿ\ ÿ\ e\ \ ÿjÿ}{2j}. \tag{1.13}$$

Em tempo contínuo, um sinal senoidal é representado por

$$x(t) = \sin(2ÿf\ t) = \sin(ÿt), \tag{1.14}$$

Machine Translated by Google

onde f e ÿ representam a frequência senoidal, respectivamente, em Hz e rad/s. Para o caso de tempo discreto, temos

$$x(n) = \text{sen}(2\ddot{y}fnTs) = \text{sen } 2\ddot{y} \; fs \; \frac{f}{\;} n \quad , \tag{1.15}$$

onde Ts e fs representam, respectivamente, o período de amostragem e a frequência de amostragem. Observe que a amostragem de um sinal periódico com período T não resulta necessariamente em um sinal periódico com o mesmo período (na verdade, o sinal resultante pode nem ser periódico). Observe também que

$$fs \; fs \; \frac{ff + kfs}{\;} \text{ sen } 2\ddot{y} \; n = \frac{\text{sen } 2\ddot{y}}{\;} n \quad , k \; \ddot{y} \; N0. \tag{1.16}$$

(Mostre o porquê e tente descobrir algumas implicações desta propriedade).

1.1.5 Operações na variável independente

Existem diversas operações no processamento de sinais que podem ser descritas através de modificações ou transformações da variável independente, t ou n. Entre eles estão inversão de tempo, atraso e avanço de tempo, compressão e expansão de tempo.

Consideremos o sinal

$$y(t) = x(\ddot{y}t). \tag{1.17}$$

O sinal y(t) consiste na **inversão temporal** do sinal x(t). Esse é o efeito que acontece, por exemplo, quando tocamos uma gravação no sentido contrário. O equivalente para o caso digital é óbvio: se o sinal original for x(n), então o sinal invertido no tempo é x(ÿn).

Consideremos agora o sinal

$$y(t) = x(t \; \ddot{y} \; \ddot{y} \;). \tag{1.18}$$

Observe que o sinal y(t) tem valor igual a x(0) para t = ÿ . Portanto, se ÿ > 0, esse valor acontece temporalmente após t = 0. Portanto, diz-se que o sinal y(t) é uma versão **atrasada** do sinal x(t). No caso em que ÿ < 0, o sinal y(t) apresenta o valor x(0) antes do instante t = 0, portanto nesta situação o sinal y(t)) é uma versão avançada de x(t) . Aqui, também, a adaptação à situação de tempo discreto é óbvia,

$$y(n) = x(n \; \ddot{y} \; k), \; k \; \ddot{y} \; Z. \tag{1.19}$$

Considere agora o sinal

$$y(t) = x(\ddot{y}t), \; \ddot{y} \; \ddot{y} \; R. \tag{1.20}$$

O sinal y(t) é uma versão de x(t) que é “mais rápido” (se |ÿ| > 1) ou “mais lento” (se |ÿ| < 1), obtendo assim compressão temporal na primeira **situação** e **temporal expansão** no segundo caso. Como exemplo prático ilustrativo destas operações, podemos apontar o caso em que uma gravação é reproduzida ao dobro da velocidade (compressão temporal) ou à metade da velocidade original (expansão temporal).

Machine Translated by Google

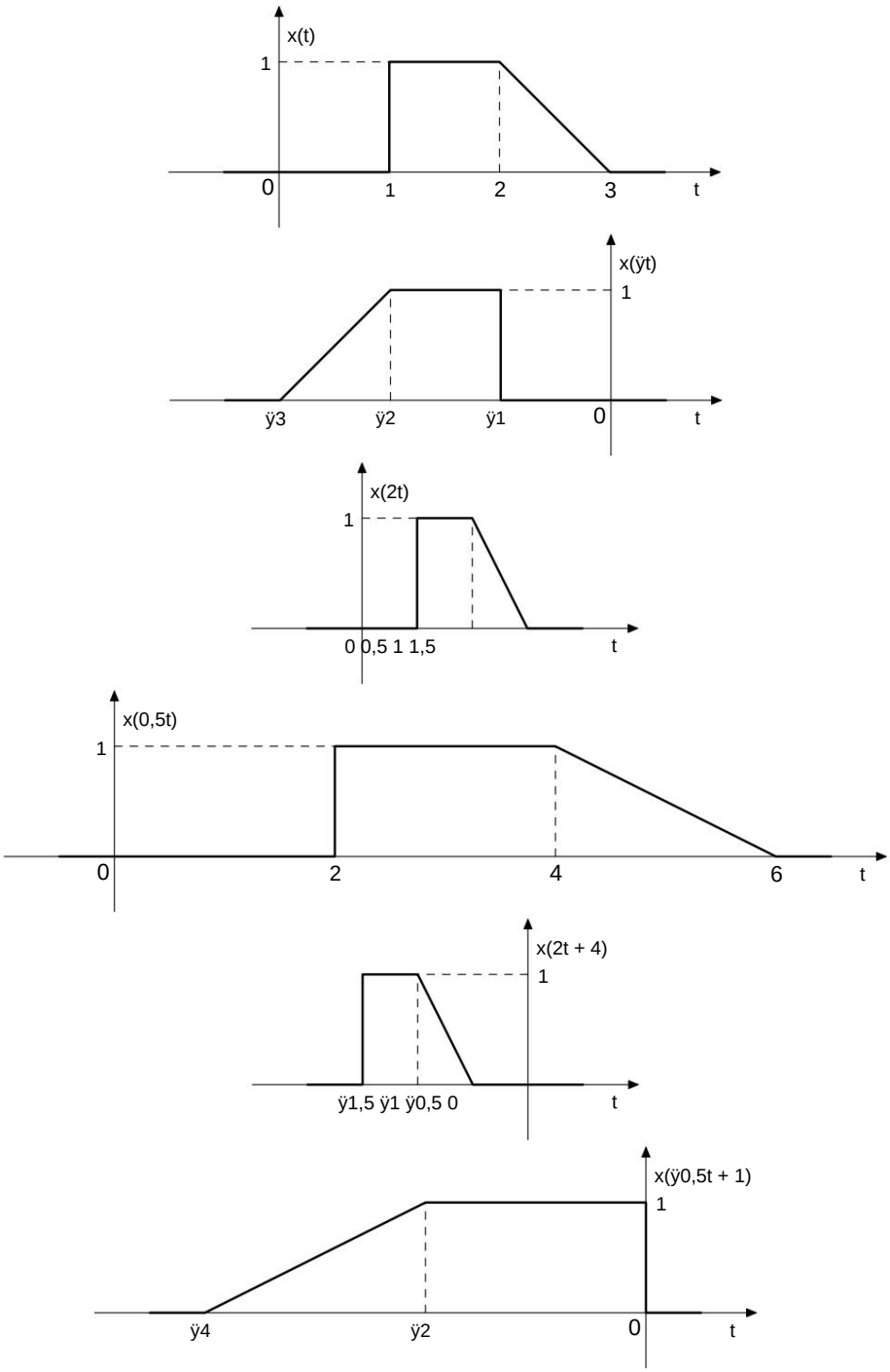


Figura 1.14: Exemplo das diversas operações na variável independente.

Qualquer transformação linear da variável independente pode ser expressa combinando as operações que acabamos de descrever (inversão de tempo, expansão ou compressão de tempo e atraso ou avanço). Por exemplo, o sinal x(ÿt ÿ ÿ) resulta da aplicação de um atraso (avanço se ÿ for negativo) de ÿ

Machine Translated by Google

unidades de tempo, seguida de uma compressão (expansão se |ÿ| < 1) e até uma inversão se ÿ < 0.
A Figura 1.14 apresenta exemplos das diversas operações sobre a variável independente, bem como combinações entre elas.

1.2 Sistemas

No contexto do processamento de sinais, um sistema é um dispositivo, processo ou algoritmo que, dado um sinal de entrada x(n), produz um sinal de saída y(n). Expressamos a ação de um sistema, H, sobre um sinal, x(n), escrevendo

$$y(n) = H[x(n)]. \tag{1.21}$$

A Figura 1.15(a) apresenta uma representação gráfica de um sistema genérico com N entradas e M saídas. Nossa atenção, entretanto, irá para sistemas com uma entrada e uma saída, como é o caso do sistema ilustrado na Figura 1.15(b).

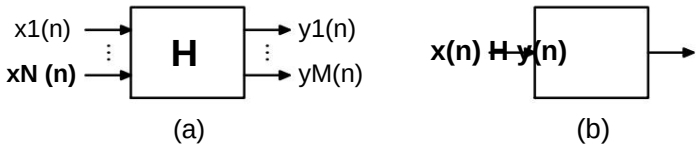


Figura 1.15: Sistema: (a) com N entradas e M saídas; (b) com uma entrada e uma saída.

Um dos sistemas mais simples de descrever matematicamente é o **amplificador ideal,**

$$y(n) = H[x(n)] = ÿx(n), \; ÿ > 0.$$

A constante ÿ é chamada de **ganho** do amplificador. Se 0 < ÿ < 1, o sistema funciona como um atenuador, ou seja, a amplitude do sinal de saída será menor que a amplitude do sinal de entrada. Se ÿ > 1, o sistema amplificará o sinal de entrada, ou seja, a amplitude do sinal de saída será maior que a amplitude do sinal de entrada. A Figura 1.16 mostra a relação entre o sinal de entrada, x(n), e o sinal de saída, y(n), de um amplificador com ganho ÿ.

O **retificador de meia onda ideal** também é um sistema fácil de representar matematicamente,

$$y(n) = H[x(n)] = \begin{matrix} x(n), \; x(n) \; ÿ \; 0 \; 0, \\ x(n) < 0 \end{matrix} , \tag{1.22}$$

bem como o **retificador de onda completa ideal,**

$$y(n) = H[x(n)] = |x(n)|. \tag{1.23}$$

A Figura 1.17 mostra a relação entre a entrada e a saída desses dois sistemas. A Figura 1.18 mostra um exemplo de retificação de meia onda e de onda completa.

Outro exemplo de sistema simples de descrever matematicamente, embora de grande importância, é o **amplificador de ganho variável,**

$$y(n) = H[x(n)] = ÿ(n)x(n). \tag{1.24}$$

Machine Translated by Google

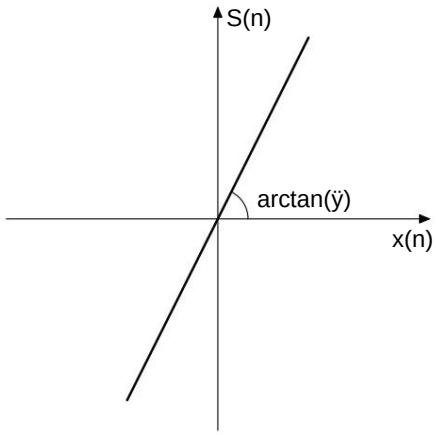


Figura 1.16: Relação entre entrada e saída de um amplificador ideal com ganho ÿ.

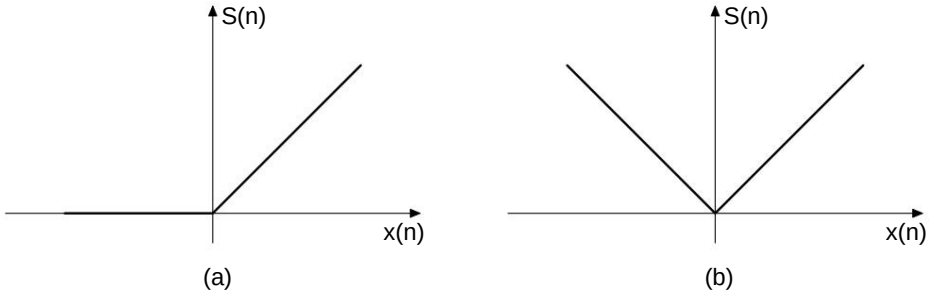


Figura 1.17: Relação entre entrada e saída de: (a) um retificador ideal de meia onda; (b) um retificador de onda completa ideal.

1.2.1 Propriedades dos sistemas: Estabilidade

Um sistema é estável se, para qualquer sinal de entrada com amplitude limitada, produzir um sinal de saída com amplitude limitada, isto é, se

$$|s(n)| \text{ ÿ } Meu < \text{ÿ}, \text{ ÿ} x(n) : |x(n)| \text{ ÿ } Mx < \text{ÿ}.$$

Por exemplo, o amplificador ideal é um sistema estável, porque se

$$|x(n)| \text{ ÿ } Mx < \text{ÿ},$$

então

$$|s(n)| = |\text{ÿ}x(n)| = |\text{ÿ}||x(n)| \text{ ÿ } |\text{ÿ}|Mx < \text{ÿ}, \text{ para } |\text{ÿ}| < \text{ÿ}.$$

Por outro lado, o sistema dado por

$$y(n) = 2y(n \text{ ÿ } 1) + x(n)$$

não é estável. Na verdade, se considerarmos um impulso como entrada do sistema, ou seja, fazendo x(n) =

Machine Translated by Google

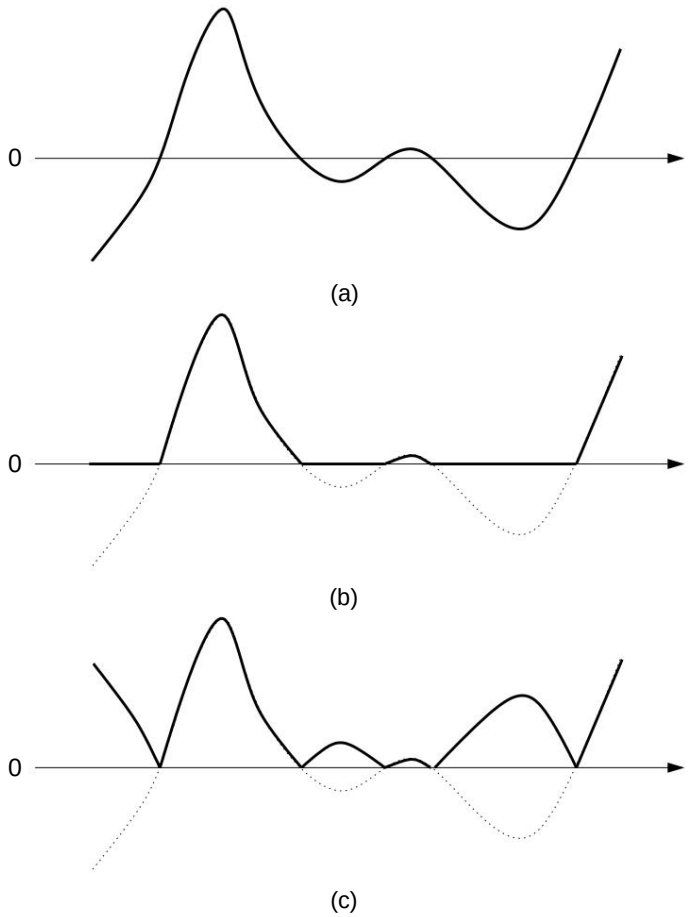


Figura 1.18: Exemplo de retificação de um sinal: (a) sinal original; (b) retificação de meia onda; (c) retificação de onda completa.

ÿ(n), teremos a seguinte sequência de valores de saída (assumimos y(n) = 0, n < 0):

y(0) = 2y(ÿ1) + ÿ(0) = 0 + 1 = 1 y(1) = 2y(0)
+ ÿ(1) = 2 + 0 = 2 y(2) = 2y(1) + ÿ(2) = 4
+ 0 = 4
. . .
y(k) = 2k
. . .

Assim, para uma entrada com amplitude limitada, obtemos uma resposta que cresce sem limite, portanto o sistema é instável.

1.2.2 Propriedades dos sistemas: Memória

Um sistema tem memória quando, para determinar sua saída em um determinado instante n, são necessários outros valores de entrada além de x(n). Por exemplo, o amplificador ideal não possui memória, pois para obter a saída

Machine Translated by Google

valor no tempo n, ou seja, y(n), só é necessário saber o valor de entrada no tempo n, ou seja, x(n).

Por outro lado, o sistema definido pela relação y(n) = 2y(n ÿ 1) + x(n) possui memória.
Na verdade, para determinar o valor de y(n), além do valor de x(n), também é necessário o valor de saída no instante anterior, ou seja, y(n ÿ 1).

1.2.3 Propriedades dos sistemas: Causalidade

Um sistema é causal se, para calcular sua saída em um determinado instante n, são necessários apenas valores atuais ou passados. Portanto, um sistema causal não precisa conhecer o futuro para determinar o resultado num determinado momento, o que, fisicamente, parece bastante óbvio. Contudo, sob certas condições, podemos realmente ter sistemas que não são causais. Comecemos por considerar o sistema causal definido através

y(n) = x(n) + x(n ÿ 1) ÿ 0,5y(n ÿ 1). (1,25)

Como podemos ver, a saída deste sistema depende apenas de valores do presente, x(n), e do passado, x(n ÿ 1) e y(n ÿ 1).

Considere agora o sistema

y(n) = x(n) + x(n + 1) ÿ 0,5y(n ÿ 1). (1.26)

Neste caso, além dos valores presentes e passados, também é necessário um valor futuro, ou seja, x(n + 1). No entanto, podemos transformar este sistema não causal em um sistema causal alterando a variável usando m = n + 1,

y(m ÿ 1) = x(m ÿ 1) + x(m) ÿ 0,5y(m ÿ 2). (1.27)

Note-se que, neste caso, a transformação do sistema não causal em sistema causal foi feita ao custo da introdução de um atraso de uma unidade de tempo no sistema. Em geral, isso só é possível se não houver requisitos de tempo real, ou seja, se não for necessário ter a saída do sistema no menor tempo possível (ou dentro de um limite máximo pré-definido) após a entrada ter sido oferecido.

1.2.4 Propriedades dos sistemas: Invertibilidade

Um sistema é invertível se for possível determinar o sinal de entrada a partir do sinal de saída.
Portanto, para que isso seja possível, cada sinal de entrada distinto deve produzir um sinal de saída distinto. Portanto, se o sistema H, tal que y(n) = H[x(n)], for invertível, haverá um sistema inverso, Hÿ1 , de tal modo que

x(n) =H ÿ1 [s(n)], ÿx(n). (1.28)

Por exemplo, o sistema

y(n) = H[x(n)] = x 3 (n) (1.29)

é invertível, mas o sistema

y(n) = H[x(n)] = x 2 (n) (1h30)

não é (por quê?).

Machine Translated by Google

1.2.5 Propriedades dos sistemas: Linearidade

A linearidade é uma das duas propriedades mais importantes de um sistema, pois, quando verificada, permite a utilização de uma ampla gama de ferramentas para análise e síntese de sistemas. A outra propriedade importante é a invariância temporal, que será discutida na próxima seção. Para ser linear, um sistema deve ser **aditivo** e **homogêneo.**

Um sistema é aditivo se

$$H[x1(n) + x2(n)] = H[x1(n)] + H[x2(n)], \ddot{y}x1(n), x2(n). \tag{1.31}$$

Esta propriedade é representada em termos de um diagrama de blocos na Figura 1.19.

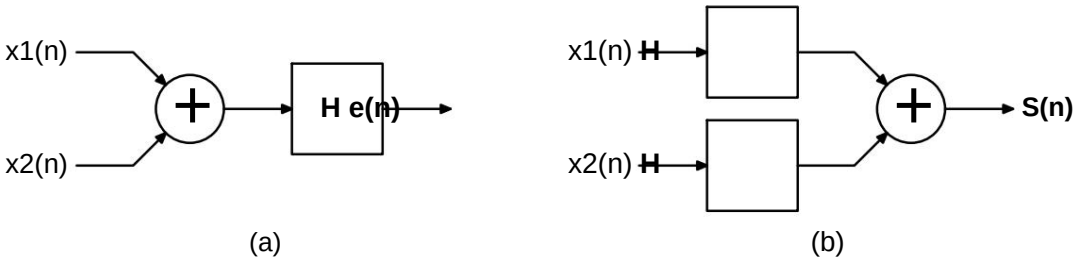


Figura 1.19: A propriedade de aditividade permite permutar o bloco soma com o sistema, mantendo a mesma funcionalidade.

Um sistema é homogêneo se

$$H[\ddot{y}x(n)] = \ddot{y}H[x(n)], \ddot{y}x(n), \ddot{y}. \tag{1.32}$$

A Figura 1.20 ilustra esta propriedade, mostrando que neste caso é possível permutar o sistema com o bloco de ganho.

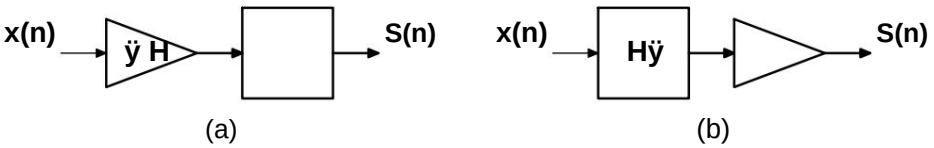


Figura 1.20: A propriedade de homogeneidade permite permutar o bloco de ganho com o sistema, mantendo a mesma funcionalidade.

1.2.6 Propriedades dos sistemas: Invariância temporal

Um sistema é invariante no tempo se um deslocamento no tempo do sinal de entrada (atraso ou avanço) produz apenas um deslocamento no tempo igual do sinal de saída. Portanto, um sistema y(n) = H[x(n)] é invariante no tempo se

$$y(n \ddot{y} k) = H[x(n \ddot{y} k)], \ddot{y}x(n), k. \tag{1.33}$$

Machine Translated by Google

Como pode ser visto na Figura 1.21, isso significa que é possível trocar um bloco de atraso com o sistema, obtendo um resultado equivalente.

Por exemplo, o retificador de onda completa ideal é um sistema invariante no tempo, porque

$$H[x(n \ddot{y} k)] = |x(n \ddot{y} k)| = y(n \ddot{y} k). \tag{1.34}$$

Por outro lado, o amplificador de ganho variável não é invariante no tempo, porque

$$H[x(n \ddot{y} k)] = \ddot{y}(n)x(n \ddot{y} k) \ddot{y}= y(n \ddot{y} k) = \ddot{y}(n \ddot{y} k)x(n \ddot{y} k). \tag{1,35}$$

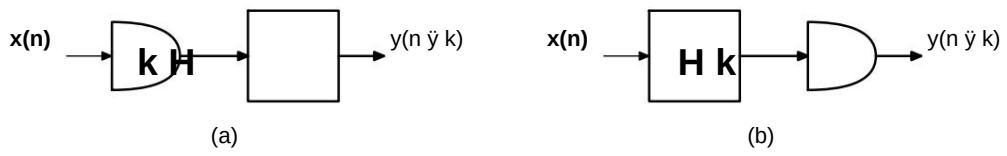


Figura 1.21: A propriedade de invariância temporal permite permutar o bloco de atraso com o sistema.

1.3 Sistemas lineares e invariantes no tempo

1.3.1 Representação de sinais por superposição de impulsos

Como pode ser visto na Figura 1.22, multiplicar um impulso unitário em n = k por qualquer sinal de tempo discreto x(n) resulta em um novo impulso também em n = k e com amplitude igual a x(k), ou seja,

$$x(n)\ddot{y}(n \ddot{y} k) = x(k)\ddot{y}(n \ddot{y} k), \ddot{y}k \ddot{y} Z.$$

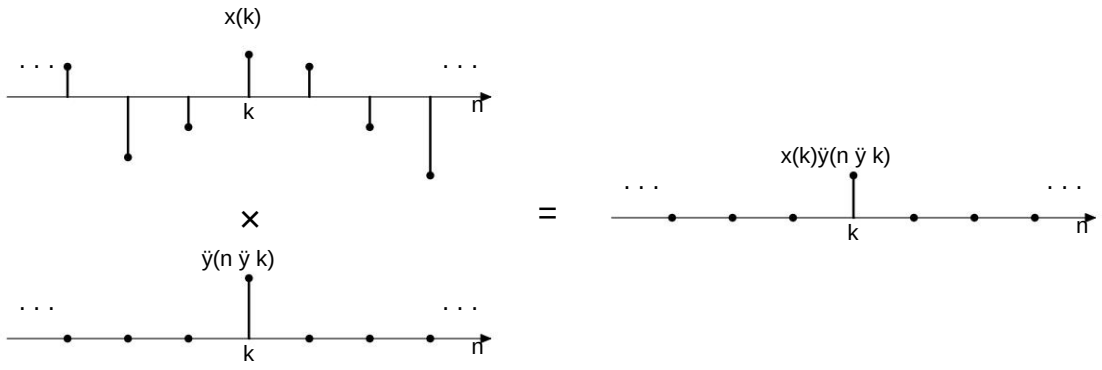


Figura 1.22: Resultado da multiplicação de um impulso unitário em n = k por um sinal discreto x(n).

Com base nesta observação, é possível representar qualquer sinal de tempo discreto, x(n), através de uma soma de impulsos, conforme exemplificado na Figura 1.23,

$$x(n) = \sum_{k = \ddot{y}\ddot{y}}^{\ddot{y}} x(k)\ddot{y}(n \ddot{y} k).$$

Machine Translated by Google

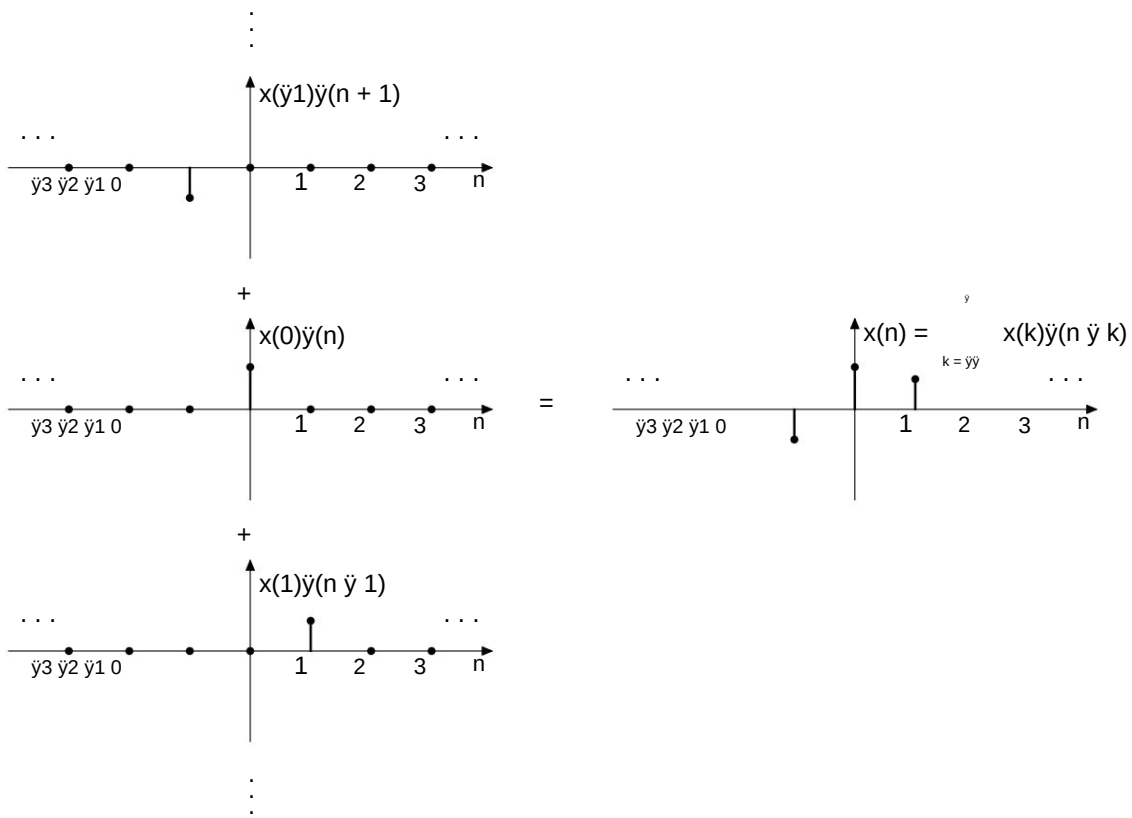


Figura 1.23: Representação de um sinal discreto, x(n), através de uma soma de impulsos.

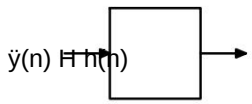


Figura 1.24: Resposta ao impulso de um sistema H.

1.3.2 Resposta ao impulso

Vamos agora considerar um sistema de tempo discreto linear e invariante no tempo, H, e sua resposta ao impulso ÿ(n), que representaremos por h(n) e chamaremos de resposta ao impulso do sistema (ver Figura 1.24). A resposta do sistema H a um sinal de entrada arbitrário, x(n), pode ser obtida por

$$y(n) = H[x(n)] = H \sum_{k=\ddot{y}\ddot{y}}^{\ddot{y}} x(k)\ddot{y}(n \ddot{y} k) . \tag{1.36}$$

Como o sistema é linear e portanto aditivo, podemos escrever

$$H \sum_{k=\ddot{y}\ddot{y}}^{\ddot{y}} x(k)\ddot{y}(n \ddot{y} k) = \sum_{k=\ddot{y}\ddot{y}}^{\ddot{y}} H[x(k)\ddot{y}(n \ddot{y} k)], \tag{1.37}$$

Machine Translated by Google

e também, usando a propriedade de homogeneidade,

$$\sum_{k=-\infty}^{\infty} H[x(k)\delta(n-k)] = \sum_{k=-\infty}^{\infty} x(k)H[\delta(n-k)], \tag{1.38}$$

e finalmente, fazendo uso da propriedade de invariância no tempo, temos

$$\sum_{k=-\infty}^{\infty} x(k)H[\delta(n-k)] = \sum_{k=-\infty}^{\infty} x(k)h(n-k). \tag{1.39}$$

Esta última expressão nos mostra uma maneira de calcular a resposta de qualquer resposta linear e invariante no tempo. sistema de tempo discreto, com resposta ao impulso h(n), para qualquer sinal de entrada, x(n). Esta operação é chamada convolução linear discreta e é representada pelo operador "δ",

$$x(n) \delta h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k). \tag{1,40}$$

Exemplo 1.3

Considere um sistema com resposta ao impulso $h(n) = 3\delta(n) + 2\delta(n-1) + \delta(n-2)$. Nos queremos determine a resposta deste sistema ao sinal $x(n) = \delta(n) \delta \delta(n-1)$.

A resposta pode ser calculada através

$$y(n) = x(n) \delta h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k) = \tag{1.41a}$$

$$= x(0)h(n-0) + x(1)h(n-1) = h(n) - h(n-1) = = 3\delta(n) + 2\delta(n-1) \tag{1.41b}$$

$$+ \delta(n-2) \delta 3\delta(n-1) \delta 2\delta(n-2) \delta \delta(n-3) = 3\delta(n) \delta \delta(n-1) \delta \delta(n-2) \delta \delta(n-3). \tag{1.41c}$$

$$\tag{1.41d}$$

1.3.3 Propriedades da operação de convolução

Vimos acima que a operação de convolução desempenha um papel muito importante no contexto da linearidade. e sistemas invariantes no tempo. Vejamos agora algumas de suas propriedades e suas implicações.

Uma das propriedades mais óbvias é a comutatividade, ou seja,

$$\text{uma}(n) \delta b(n) = \sum_{k=-\infty}^{\infty} \text{uma}(k)b(n-k) = \sum_{k=-\infty}^{\infty} \text{uma}(n-k)b(k) = b(n) \delta \text{uma}(n), \tag{1.42}$$

relação que pode ser verificada através de uma simples mudança de variável.

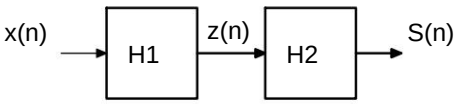
A operação de convolução também é associativa, ou seja,

$$a(n) \delta b(n) \delta c(n) = a(n) \delta b(n) \delta c(n) = a(n) \delta b(n) \delta c(n). \tag{1.43}$$

Machine Translated by Google

Exemplo 1.4

Pretendemos determinar o sistema equivalente à seguinte associação em série (ou cascata) dos sistemas H1 e H2,



Suponha que você coloque na entrada desta cascata um sinal delta, ÿ(n). Determinar a saída sinal e então obter a resposta ao impulso equivalente.

A operação de convolução também é distributiva em relação à adição, ou seja,

$$a(n) \text{ ÿ } b(n) + c(n) = a(n) \text{ ÿ } b(n) + a(n) \text{ ÿ } c(n). \tag{1.44}$$

(Nota: utilize esta propriedade para determinar o sistema equivalente numa associação paralela de sistemas).

1.3.4 A resposta ao impulso e algumas propriedades dos sistemas

Memória: Um sistema não tem memória se usar apenas o valor real de entrada para calcular o saída. Portanto, como a saída de um sistema linear e invariante no tempo pode ser calculada usando

$$s(n) = \sum_{k=\text{ÿÿ}}^{\text{ÿ}} h(k)x(n \text{ ÿ } k), \tag{1,45}$$

em um sistema sem memória temos y(n) = cx(n), onde c é uma constante arbitrária, ou seja, h(n) = cÿ(n).

Causalidade: Um sistema é causal se, para calcular sua saída em um determinado instante n, apenas a corrente ou valores de entrada anteriores são necessários. Portanto, a saída de um sistema causal no instante n só pode ser calculado com base em x(n ÿ k), k ÿ 0. Isso implica h(n) = 0, n < 0, ou seja, a saída de um sistema causal linear e invariante no tempo pode ser calculado através

$$s(n) = \sum_{k=0}^{\text{ÿ}} h(k)x(n \text{ ÿ } k). \tag{1.46}$$

Estabilidade: Como vimos, um sistema é estável se responder a entradas com amplitude limitada com saídas com amplitude limitada. Portanto, um sistema linear e invariante no tempo discreto é estável se,

$$\text{ÿ}x(n) \text{ e } |x(n)| \text{ ÿ } Mx < \text{ÿ},$$

Machine Translated by Google

Nós temos

$$|s(n)| \text{ ÿ Meu} < \text{ÿ}.$$

Então,

$$|s(n)| = \sum_{k=\text{ÿÿ}}^{\text{ÿ}} h(k)x(n \text{ ÿ } k) \text{ ÿ} \sum_{k=\text{ÿÿ}}^{\text{ÿ}} |h(k)x(n \text{ ÿ } k)| = \tag{1.47a}$$

$$\sum_{k=\text{ÿÿ}}^{\text{ÿ}} |h(k)||x(n \text{ ÿ } k)| \text{ ÿ MX} \sum_{k=\text{ÿÿ}}^{\text{ÿ}} |h(k)|. \tag{1.47b}$$

Portanto, se o sistema verificar a condição

$$\sum_{k=\text{ÿÿ}}^{\text{ÿ}} |h(k)| < \text{ÿ}, \tag{1,48}$$

então é estável.

Exemplo 1.5

Considere um sistema linear e invariante no tempo cuja resposta ao impulso é dada por

$$h(n) = anu\ (n),$$

onde a ÿ R. Queremos determinar para quais valores do parâmetro a este sistema é estável.
Como vimos, a resposta ao impulso deve obedecer

$$\sum_{k=\text{ÿÿ}}^{\text{ÿ}} |h(k)| < \text{ÿ},$$

através do qual

$$\sum_{k=\text{ÿÿ}}^{\text{ÿ}} |um\ ku(k)| = \sum_{k=0}^{\text{ÿ}} |a^k| < \text{ÿ},$$

ou seja, |a| <1.

Invertibilidade: Um sistema é invertível se sua entrada puder ser obtida de sua saída. Chamemos (n). a resposta ao impulso do sistema inverso (quando tal sistema existe) h visto, a $^{\text{ÿ}1}$ Como nós temos resposta de um sistema linear invariante no tempo pode ser calculada usando a operação de convolução. Por exemplo, no caso de tempo discreto temos y(n) = x(n) ÿh(n). A recuperação de x(n) de y(n) é chamada **deconvolução.** Portanto, h (n) deve ser tal que x(n) = y(n) ÿ h $^{\text{ÿ}1}$ (n), isto é,

$$x(n) = y(n) \text{ ÿ } h^{\text{ÿ}1}(n) = [x(n) \text{ ÿ } h(n)] \text{ ÿ } h^{\text{ÿ}1}(n) = x(n) \text{ ÿ } [h(n) \text{ ÿ } h^{\text{ÿ}1}(n)], \tag{1,49}$$

implicando

$$h(n) \text{ ÿ} h^{\text{ÿ}1}(n) = \text{ÿ}(n). \tag{1,50}$$

Machine Translated by Google

Exemplo 1.6

Consideremos um sistema linear, invariante no tempo e de tempo discreto que reproduz o sinal de entrada adicionado a uma réplica de entrada atrasada por uma unidade de tempo (eco), ou seja,

$$y(n) = x(n) + \text{machado}(n \ddot{y} 1).$$

Queremos determinar (caso exista) o sistema inverso que eliminará o eco no receptor.

Começamos escrevendo a resposta ao impulso deste sistema, que é dada por

$$h(n) = \ddot{y}(n) + a\ddot{y}(n \ddot{y} 1).$$

Então,

$$h(n) \ddot{y} h^{\ddot{y}1}(n) = [\ddot{y}(n) + a\ddot{y}(n \ddot{y} 1)] \ddot{y} h^{\ddot{y}1}(n) = h^{\ddot{y}1}(n) + ah\ddot{y}1 (n \ddot{y} 1) = \ddot{y}(n).$$

Considerando que o sistema inverso é causal, temos que h pode escrever $h^{\ddot{y}1}(n) = 0$, $n < 0$. Então, para $n = 0$, nós

$$h^{\ddot{y}1}(0) + ah\ddot{y}1 (\ddot{y}1) = \ddot{y}(0),$$

ou seja, $h^{\ddot{y}1}(0) = 1$. Para $n = 1$ temos

$$h^{\ddot{y}1}(1) + ah\ddot{y}1 (0) = \ddot{y}(1),$$

ou seja, $h^{\ddot{y}1}(1) = \ddot{y}$uma. Para $n = k \ddot{y} 0$ temos

$$\text{hora}^{\ddot{y}1}(k) = (\ddot{y}a)^{k}.$$

Portanto, a resposta ao impulso do sistema inverso é $h^{\ddot{y}1}(n) = (\ddot{y}a)nu(n)$.

Observe que o sistema inverso só é estável se

$$\sum_{n=\ddot{y}\ddot{y}}^{\ddot{y}} |h^{\ddot{y}1}(n)| = \sum_{n=0}^{\ddot{y}} |(\ddot{y}a)n| < \ddot{y},$$

implicando, neste caso, que $|a| <1$.

1.3.5 Resposta a exponenciais complexas

Consideremos um sistema linear, invariante no tempo e em tempo discreto com resposta ao impulso h(n). O $j\ddot{y}n$ é dado pela sistema ao sinal $x(n) = e$ resposta deste

$$y(n) = h(n) \ddot{y} x(n) = \sum_{k=\ddot{y}\ddot{y}}^{\ddot{y}} h(k)e^{j\ddot{y}(n\ddot{y}k)} = e^{j\ddot{y}n} \sum_{k=\ddot{y}\ddot{y}}^{\ddot{y}} h(k)e^{\ddot{y}j\ddot{y}k} = H(e^{j\ddot{y}})e^{j\ddot{y}n}, \quad (1.51)$$

Machine Translated by Google

com

$$\text{Ele}^{\text{jÿ}}) = \sum_{k=\text{ÿÿ}}^{\text{ÿ}} (k)e^{\text{ÿjÿkh}}, \qquad (1,52)$$

que chamamos **de resposta em frequência** do sistema.

Este resultado mostra uma característica muito importante dos sistemas lineares e invariantes no tempo, que servirá de base para outros resultados que apresentaremos posteriormente: a resposta ao sinal e é o sinal H(e jÿ)e jÿn exponencial é onde H(e jÿ) ÿ C é dado por (1.52). Por esta razão, dizemos que jÿn complexas são **sinais próprios** dos sistemas lineares e invariantes no tempo.

Como vimos, H(e jÿ) depende, por um lado, do próprio sistema e, por outro lado, da frequência, ÿ, da exponencial complexa na entrada do sistema. Portanto, para um determinado sistema, o valor de H(e jÿ) depende apenas da frequência do sinal.

Exemplo 1.7

Considere um sistema com resposta ao impulso

$$h(n) = \begin{cases} 0{,}5, & n = 0, 1 \\ 0, & \text{outros} \end{cases}.$$

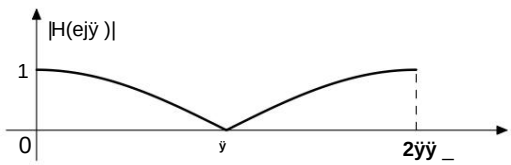
A resposta de frequência deste sistema é dada por

$$\text{Ele}^{\text{jÿ}}) = \sum_{n=\text{ÿÿ}}^{\text{ÿ}} h(n)e^{\text{ÿjÿn}} = \frac{1}{2} + \frac{1}{2}\text{ÿjÿ e} = \frac{1}{2}e^{\text{ÿjÿ/2 (e jÿ/2}} + e^{\text{ÿjÿ/2}}) = e^{\text{ÿjÿ/2}} \text{ porque } \frac{Ah}{2}.$$

Como, em geral, H(e jÿ) ÿ C, é usual calcular seu módulo e fase. Neste caso, o módulo é dado por

$$|H(e^{\text{jÿ}} \; 2)| = e^{\text{ÿjÿ/}} \cos \frac{Ah}{2} = \text{porque } \frac{Ah}{2},$$

que tem o seguinte formato:



Exemplo 1.8

Considere agora um sistema com resposta ao impulso

$$h(n) = \begin{cases} \text{ÿ} & 0{,}5, \; n = 0 \\ \text{ÿ0,5}, & n = 1 \\ 0, & \text{outros} \end{cases}.$$

Machine Translated by Google

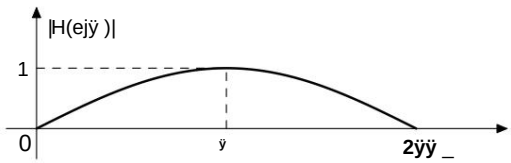
A resposta de frequência deste sistema é dada por

$$\text{Ele}^{j\ddot{y}}) = \sum_{n=\ddot{y}}^{\ddot{y}} h(n)e^{\ddot{y}j\ddot{y}n} = \frac{1}{2} \cdot \frac{1}{2}\ddot{y}j\ddot{y}\, e = \frac{1}{2} e^{\ddot{y}j\ddot{y}/2}(e^{j\ddot{y}/2} \quad {}^{\ddot{y}\, e} \quad \ddot{y}j\ddot{y}/2) = je^{\ddot{y}j\ddot{y}/2} \operatorname{sen} 2\frac{\ddot{y}}{}\cdot$$

Seu módulo é

$$|H(e^{j\ddot{y}} \operatorname{sen})| = \frac{\ddot{y}j\ddot{y}/2}{\text{eu } 2} \quad \frac{\ddot{y}}{} = \operatorname{sen} 2\frac{\ddot{y}}{},$$

que tem o seguinte formato:



1.4 A transformada discreta de Fourier (DFT)

Consideremos um sinal periódico e de tempo discreto, x(n), com período N. Pretendemos determinar se é possível representar (ou pelo menos aproximar) x(n) através da sobreposição de exponenciais complexas, ou seja,

$$x(n) \ddot{y} \tilde{x}(n) = \sum_{k} X(k)e^{j\ddot{y}kn}, \qquad (1,53)$$

onde x˜(n) é uma versão aproximada de x(n).

Antes de prosseguirmos, vejamos que tal representação é vantajosa. Na verdade, como vimos acima, a resposta de um sistema linear e invariante no tempo, cuja resposta ao impulso é h(n), a um sinal x(n), pode ser obtida através da soma de convolução, ou seja,

$$y(n) = x(n) \ddot{y} h(n) = \sum_{\text{eu}} h(m)x(n \ddot{y} m) = \sum_{\text{eu}} h(m) \sum_{k} X(k)e^{j\ddot{y}k(n\ddot{y}m)} = \qquad (1,54a)$$

$$\sum_{k} X(k)e^{j\ddot{y}kn} \sum_{\text{eu}} h(m)e^{\ddot{y}j\ddot{y}km} = \sum_{k} X(k)H(e^{j\ddot{y}k} j\ddot{y}kn)e = \sum_{k} S(k)e^{j\ddot{y}kn}, \qquad (1.54b)$$

onde H(e jÿk) denota o valor da resposta de frequência do sistema na frequência ÿk radianos. Portanto, se for possível representar x(n) como uma soma de exponenciais complexas, a saída de qualquer sistema linear e invariante no tempo também será uma soma de exponenciais complexas, cujos coeficientes, Y(k), são simplesmente obtidos por calculando os produtos X(k)H(e jÿk). Resumindo, a operação de convolução é transformada na operação de multiplicação.

Machine Translated by Google

Primeiro, vejamos quais restrições devem ser aplicadas às frequências ÿk para que o sinal resultante de (1.53) também seja periódico com período N. Porque, para verificar esta condição, temos que ter

$$\tilde{x}(n) = \tilde{x}(n + N) = \sum_k X(k)e^{j\ddot{y}k(n+N)} = \sum_k X(k)e^{j\ddot{y}kn} e^{j\ddot{y}kN}, \tag{1,55}$$

então

$$\ddot{y}kN = 2\ddot{y}k, \; k \; \ddot{y} \; Z, \tag{1,56}$$

ou seja,

$$2\ddot{y}k\ddot{y}k = \frac{}{N}, \quad k \; \ddot{y} \; Z. \tag{1,57}$$

Portanto, todas as frequências ÿk devem ser múltiplos da frequência fundamental, ou seja, múltiplos de 2ÿ/N.

Por outro lado, desde

$$j e^{\frac{2\ddot{y}kn}{N}j} = e^{\frac{2\ddot{y}}{N}(k+N)n}, \tag{1,58}$$

então existem apenas N exponenciais distintas, então a soma em (1.53) pode ser limitada a N termos, ou seja,

$$x(n) \; \ddot{y} \; \tilde{x}(n) = \sum_{k=\ddot{y}N\ddot{y}} X(k)e^{j\frac{2\ddot{y}kn}{N}}, \tag{1,59}$$

onde a notação ÿNÿ significa qualquer conjunto de N inteiros consecutivos5 .

Vejamos agora que, de fato, é possível representar com exatidão qualquer sinal periódico, x(n), por uma soma de exponenciais complexas. Para isso, mostraremos primeiro que as N exponenciais são todas ortogonais entre si, ou seja, que formam uma base no espaço de sinais periódicos de tempo discreto de dimensão N.

O produto interno entre dois sinais discretos, a(n) e b(n), com duração N, é definido como

$$\ddot{y}a(n), b(n)\ddot{y} = \sum_{n=\ddot{y}N\ddot{y}} \text{uma}(n)b \; \ddot{y} \; (n). \tag{1,60}$$

Dois sinais são ortogonais se seu produto interno for zero. Desde

$$e^{j\frac{2\ddot{y}}{N}kn}, e^{j\frac{2\ddot{y}}{N}Em} = \sum_{n=\ddot{y}N\ddot{y}} e^{j\frac{2\ddot{y}}{N}\text{joelho}\,\ddot{y}j\frac{2\ddot{y}}{N}Em} = \tag{1.61a}$$

$$= \sum_{n=0}^{N-1} e^{j\frac{2\ddot{y}}{N}(k\ddot{y}l)n} = \begin{cases} \ddot{y} \; N, & k = eu \\ \ddot{y}\,\dfrac{j\,1\,\ddot{y}\,e^{\frac{2\ddot{y}}{N}(k\ddot{y}l)N}}{2\ddot{y}\,j\,1\,\ddot{y}\,e^{\frac{2\ddot{y}}{N}(k\ddot{y}l)}} = 0, & k \; \ddot{y} = l \end{cases}, \tag{1.61b}$$

5Observe que, devido à periodicidade das exponenciais, não importa onde começa este conjunto de N inteiros consecutivos. Porém, às vezes, motivado pela conveniência da manipulação matemática, um determinado conjunto é utilizado, por exemplo, {0, 1, . . . , N ÿ 1}.

Machine Translated by Google

temos então que, de fato, as exponenciais complexas N são todas ortogonais entre si. Portanto, qualquer sinal periódico, x(n), com período N, pode ser representado por

$$x(n) = \sum_{k=ÿNÿ} X(k)e^{\frac{2ÿkn}{N}j}. \tag{1,62}$$

Os coeficientes X(k) podem ser obtidos calculando o produto interno entre o sinal e os sinais base, ou seja,

$$x(n), ej^{\frac{2ÿ}{N}sabe} = \sum_{n=ÿNÿ} x(n)e^{ÿj\frac{2ÿ}{N}sabe} = \sum_{n=ÿNÿ\ eu=ÿNÿ} X(l)e^{j\frac{2ÿ}{N}linha\ ÿj\frac{2ÿ}{N}sabe} = \tag{1.63a}$$

$$\sum_{eu=ÿNÿ} X(l) \sum_{n=ÿNÿ} j\,e^{\frac{2ÿ}{N}(l-k)n} = NX(k). \tag{1.63b}$$

Por isso,

$$X(k) = \frac{1}{N} \sum_{n=ÿNÿ} x(n)e^{ÿj\frac{2ÿ}{N}sabe}. \tag{1,64}$$

A operação que transforma o sinal x(n) nos coeficientes X(k) é chamada de **transformada (ou análise) discreta direta de Fourier.** A operação que transforma os coeficientes X(k) no sinal x(n) é chamada de **transformada discreta inversa de Fourier (ou síntese).** Às vezes, usaremos a notação

$$x(n) \overset{DFT}{ÿÿ} X(k) \text{ para} \tag{1,65}$$

indicar que x(n) e X(k) estão relacionados através da transformada discreta de Fourier (DFT).

1.4.1 Algumas propriedades da transformada discreta de Fourier

Existem diversas propriedades associadas à transformada discreta de Fourier que podem ser muito úteis, tanto para simplificar alguns cálculos como para ajudar a compreender melhor o seu significado.

A **linearidade** é uma das propriedades mais óbvias e fáceis de mostrar. De facto, considerando dois sinais, a(n) e b(n), com duração N, e as respectivas transformadas discretas de Fourier, A(k) e B(k),

$$uma(n) \overset{DFT}{ÿÿ} UMA(k) \tag{1.66a}$$

$$b(n) \overset{DFT}{ÿÿ} B(k), \tag{1.66b}$$

podemos mostrar isso

$$c1a(n) + c2b(n) \overset{DFT}{ÿÿ} c1A(k) + c2B(k), \tag{1,67}$$

com c1, c2 ÿ C.

Algumas propriedades interessantes têm a ver com vários tipos de **simetrias.** Por exemplo, se o sinal x(n) ÿ R, então

$$X(k) = X\,ÿ\,(ÿk), \tag{1,68}$$

Machine Translated by Google

ou seja, neste caso o módulo de X(k) tem simetria par e a fase tem simetria ímpar.
Pode-se verificar também que, se, além de ser real, o sinal possui simetria par, então X(k) ÿ R.

Se um sinal, x(n), com transformada discreta de Fourier X(k), for **atrasado** n0 unidades de tempo, então

$$\frac{1}{N}\sum_{n=ÿNÿ} x(n ÿ n0)e^{ÿj\frac{2ÿkn}{N}} = \frac{1}{N}\cdot\sum_{m=ÿNÿ} x(m)e^{ÿj\frac{2ÿ}{N}k(m+n0)} = e^{ÿj\frac{2ÿ}{N}kn0}X(k). \qquad (1,69)$$

Portanto, se

$$x(n) \overset{DFT}{ÿÿ} X(k), \qquad (1,70)$$

então

$$x(n ÿ n0) \overset{DFT}{ÿÿ} e^{ÿj\frac{2ÿ}{N}kn0}X(k). \qquad (1,71)$$

Observe que uma **mudança de tempo** causa apenas uma mudança na fase da transformada discreta de Fourier.

Por outro lado, se introduzirmos uma **mudança de frequência** de k0 unidades, teremos

$$\sum_{k=ÿNÿ} X(k ÿ k0)e^{j\frac{2ÿkn}{N}} = \sum_{eu=ÿNÿ} X(l)e^{j\frac{2ÿ}{N}(l+k0)n} = e^{j\frac{2ÿ}{N}k0n}x(n). \qquad (1,72)$$

Portanto, se

$$x(n) \overset{DFT}{ÿÿ} X(k), \qquad (1,73)$$

então

$$e^{j\frac{2ÿ}{N}k0n}x(n) \overset{DFT}{ÿÿ} X(k ÿ k0). \qquad (1,74)$$

Exemplo 1.9

Considere um sinal x(n) de duração N, com N par, tal que $x(n) \overset{DFT}{ÿÿ} X(k)$. Se for realizada uma mudança na frequência de N/2 posições, teremos $e^{jÿn}x(n) \overset{DFT}{ÿÿ} X(kN/2)$, ou seja, $(ÿ1)^n x(n) \overset{DFT}{ÿÿ} X(kN/2)$. Portanto, se um determinado sinal do tipo passa-baixa for multiplicado por $(ÿ1)^n$, ele terá uma característica passa-alta e vice-versa.

Uma das propriedades mais importantes da transformada discreta de Fourier tem a ver com a **conservação de energia** (ignorando uma possível constante multiplicativa), ao passar de um domínio para outro (tempo para frequência ou vice-versa). A energia de um sinal x(n) com duração N é dada por

$$Ex = \sum_{n=ÿNÿ} |x(n)|^2 = \sum_{n=ÿNÿ} x(n)x^ÿ(n) = \sum_{n=ÿNÿ} x(n) \sum_{k=ÿNÿ} X^ÿ(k)e^{2ÿ ÿj\frac{}{N}sabe} = (1,75a)$$

$$\sum_{k=ÿNÿ} X^ÿ(k) \sum_{n=ÿNÿ} x(n)e^{ÿj\frac{2ÿ}{N}sabe} = \sum_{k=ÿNÿ} X^ÿ(k)NX(k) = N\sum_{k=ÿNÿ} |X(k)|^2. \qquad (1.75b)$$

Machine Translated by Google

Este resultado é conhecido como relação de Parseval.

Consideremos agora dois sinais, a(n) e b(n), com duração N, e as respectivas transformadas discretas de Fourier, A(k) e B(k). Pretendemos determinar a transformada discreta de Fourier do sinal a(n)b(n), ou seja,

$$\frac{1}{N}\sum_{n=ÿNÿ} a(n)b(n)e^{ÿj\frac{2ÿ}{N}sabe} = \frac{1}{N}\sum_{n=ÿNÿ}\sum_{ÿ\,eu=ÿNÿ} A(l)e^{j\frac{2ÿ}{N}lnÿ}\sum_{ÿ} b(n)e^{ÿj\frac{2ÿ}{N}sabe} = \tag{1.76a}$$

$$\sum_{eu=ÿNÿ} UMA(eu)\sum_{ÿ}\frac{1}{N}\sum_{n=ÿNÿ} b(n)e^{ÿj\frac{2ÿ}{N}(kÿl)n}\sum_{ÿ} = \sum_{eu=ÿNÿ} A(l)B(k\,ÿ\,l) = A(k)\,ÿ\,B(k). \tag{1.76b}$$

Observe que a última soma apresentada está na forma de uma convolução. No entanto, como os sinais são periódicos e esta soma só é definida em números inteiros consecutivos N, então o resultado desta convolução é também um sinal periódico com período N. É por isso que usamos o símbolo “ÿ” para denotar este sinal periódico (ou convolução circular), em vez do símbolo “ÿ” que introduzimos anteriormente para denotar a operação de convolução linear.

Ao calcular a transformada discreta inversa de Fourier do produto A(k)B(k) um resultado semelhante também é obtido. Na verdade,

$$\sum_{k=ÿNÿ} A(k)B(k)e^{j\frac{2ÿkn}{N}} = \sum_{k=ÿNÿ}\sum_{ÿ}\frac{1}{N\cdot m=ÿNÿ} a(m)e^{ÿj\frac{2ÿ}{N}kmÿÿ} B(k)e^{j\frac{2ÿkn}{N}} = \tag{1.77a}$$

$$\frac{1}{N\cdot m=ÿNÿ} sou)\sum_{ÿ\,ÿ\,k=ÿNÿ} B(k)e^{j\frac{2ÿ}{N}k(nÿm)}\sum_{ÿ} = \frac{1}{N\cdot m=ÿNÿ} uma(m)b(n\,ÿ\,m) = \frac{1}{N} uma(n)\,ÿb(n). \tag{1.77b}$$

Em suma, se

$$uma(n) \overset{DFT}{ÿÿ} UMA(k) \tag{1.78a}$$

$$b(n) \overset{DFT}{ÿÿ} B(k), \tag{1.78b}$$

então

$$uma(n)b(n) \overset{DFT}{ÿÿ} UMA(k)\,ÿ\,B(k) \tag{1.79a}$$

$$uma(n)\,ÿ\,b(n) \overset{DFT}{ÿÿ} NA(k)B(k), \tag{1.79b}$$

isto é, multiplicações de sinais no domínio do tempo são transformadas em convoluções no domínio da frequência e vice-versa.

Machine Translated by Google

1.4.2 A transformada discreta de cosseno

Mencionamos que se um sinal x(n), de comprimento N, é real e tem simetria par, então os coeficientes de sua DFT, X(k), são números reais. Esta é uma situação conveniente em muitos casos.
Além disso, devido à extensão periódica intrínseca que x(n) sofre quando calculamos sua DFT, geralmente ocorrem problemas de descontinuidade, implicando componentes de alta frequência com valores significativos. Para obter compressão, é desejável ter a energia do sinal compactada no menor número possível de coeficientes de transformada, portanto a DFT geralmente não é uma boa escolha.

A **transformada discreta de cosseno (DCT)** aborda esses dois problemas. Por um lado, fornece coeficientes de transformação reais. Por outro lado, proporciona embalagem de alta energia.
Existem várias versões do DCT. O utilizado para compressão é conhecido como DCT-II e tem relação direta

$$X(k) = \frac{\sqrt{2}}{N}\,\ddot{y}(k)\sum_{n=0}^{N-1} x(n)\cos\frac{\ddot{y}k(2n+1)}{2N}, \quad k = 0, 1, \ldots, N\ddot{y}1 \tag{1,80}$$

e a relação inversa

$$x(n) = \frac{\sqrt{2}}{N}\sum_{k=0}^{N-1} \ddot{y}(k)X(k)\cos\frac{\ddot{y}k(2n+1)}{2N}, \quad n = 0, 1, \ldots, N\ddot{y}1, \tag{1,81}$$

com

$$\ddot{y}(0) = \frac{1}{\ddot{y}\,2} \quad \text{e } \ddot{y}(k) = 1,\ k\ddot{y}= 0.$$

Esta versão do DCT pode ser calculada usando uma DFT de 2N pontos (ou FFT6), espelhando x(n) (por exemplo, “abcd” é transformado em “dcbaabcd”) e realizando um ajuste de mudança de tempo apropriado.

1.5 A transformada Z e filtros digitais

Consideremos um sistema discreto, H, linear e invariante no tempo, com resposta ao impulso h(n), e o sinal

$$x(n) = z\,A^{n} = rej\ddot{y}n \quad .$$

resposta de H a x(n) pode ser obtida através da soma de convolução, ou seja,

$$y(n) = x(n)\,\ddot{y}\,h(n) = \sum_{k=\ddot{y}\ddot{y}}^{\ddot{y}} h(k)x(n\ddot{y}k) = \sum_{k=\ddot{y}\ddot{y}}^{\ddot{y}} h(k)z^{n-k}\,n = z\sum_{k=\ddot{y}\ddot{y}}^{\ddot{y}} h(k)z^{\ddot{y}k} = znH(z).$$

A função

$$H(z) = \sum_{n=\ddot{y}\ddot{y}}^{\ddot{y}} h(n)z\,\ddot{y}n \tag{1,82}$$

6A Transformada Rápida de Fourier (FFT) é um algoritmo eficiente para calcular os coeficientes da DFT. Em vez de de exigir O(N2) operações, a FFT precisa apenas de O(N log N) operações, para um sinal de comprimento N.

Machine Translated by Google

é a transformada Z de h(n).

Exemplo 1.10

Calcule a transformada Z, X(z), do sinal x(n) = a nu(n).

Nós temos

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} = \sum_{n=0}^{\infty} (az^{-1})^n = \frac{1}{1-az^{-1}}, \quad |z| > |a|.$$

Note que uma região de convergência está associada a X(z), ou seja, neste caso a função X(z) obtida só é válida para valores de z tais que |z| > |a|.

Exemplo 1.11

Calcule a transformada Z do sinal x(n) = an cos(ωn)u(n).

Neste caso X(z) é dado por

$$X(z) = \sum_{n=-\infty}^{\infty} x(n)z^{-n} = \frac{1}{2}\sum_{n=0}^{\infty} (ae^{j\omega}z^{-1})^n + (ae^{-j\omega}z^{-1})^n = \tag{1.83a}$$

$$\frac{1}{2}\left[\frac{1}{1-ae^{j\omega}z^{-1}} + \frac{1}{1-ae^{-j\omega}z^{-1}}\right], \quad |z| > |a|, \tag{1.83b}$$

que pode ser simplificado para

$$X(z) = \frac{1-a\cos\omega z^{-1}}{1-2a\cos\omega z^{-1}+a^2z^{-2}}. \tag{1,84}$$

1.5.1 A propriedade time-shift

Consideremos um sinal, x(n), com transformada Z X(z). Queremos determinar a transformada Z do sinal x(n − k), ou seja,

$$\sum_{n=-\infty}^{\infty} x(n-k)z^{-n} = \sum_{n=-\infty}^{\infty} x(n)z^{-(n-k)} = z^{-k}\sum_{n=-\infty}^{\infty} x(n)z^{-n} = z^{-k}X(z).$$

Portanto, se

$$x(n) \overset{Z}{\longleftrightarrow} X(z),$$

então

$$x(n-k) \overset{Z}{\longleftrightarrow} z^{-k}X(z).$$

Machine Translated by Google

Esta propriedade permite uma fácil transformação para o domínio Z de equações de diferenças lineares de coeficientes constantes, ou seja, do tipo

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k),$$

desde então, porque

$$y(n-k) \xrightarrow{Z} z^{-k} Y(z) \text{ e } x(n-k) \xrightarrow{Z} z^{-k} X(z),$$

Nós temos

$$\sum_{k=0}^{N} a_k y(n-k) \xrightarrow{Z} \sum_{k=0}^{N} a_k z^{-k} Y(z) \text{ e } \sum_{k=0}^{M} b_k x(n-k) \xrightarrow{Z} \sum_{k=0}^{M} b_k z^{-k} X(z)$$

e

$$\sum_{k=0}^{N} a_k z^{-k} Y(z) = \sum_{k=0}^{M} b_k z^{-k} X(z).$$

Também podemos escrever

$$H(z) = \frac{S(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}}.$$

Se x(n) é o sinal de entrada e y(n) é o respectivo sinal de saída de um certo sistema linear invariante no tempo, então a função H(z) é a função de **transferência** desse sistema.

Exemplo 1.12

Determine a função de transferência de um sistema descrito pela equação

$$y(n) = x(n) - x(n-1) + 0{,}5y(n-1).$$

Calculando a transformada Z termo a termo, obtemos

$$Y(z) = X(z) - z^{-1}X(z) + 0{,}5z^{-1}Y(z)$$

e, portanto,

$$H(z) = \frac{S(z)}{X(z)} = \frac{1 - z^{-1}}{1 - 0{,}5z^{-1}}.$$

As raízes do numerador e do denominador de H(z) são chamadas, respectivamente, de **zeros** e **pólos** da função de transferência. Um sistema que não possui pólos (isto é, o denominador de H(z) é uma constante) é chamado de sistema **FIR (resposta ao impulso finita) ,** caso contrário é chamado de sistema **IIR (resposta ao impulso infinita) .**

Machine Translated by Google

Exemplo 1.13

Determine os pólos e zeros da função de transferência do sistema mostrado no Exemplo 1.12.

Os pólos de um sistema são as raízes do denominador da função de transferência, então, neste caso, temos

$$1 - 0{,}5z^{-1} = 0 \Rightarrow z = 0{,}5.$$

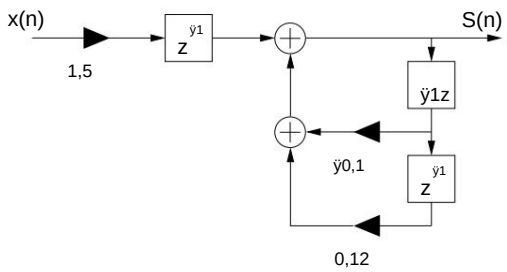
Os zeros de um sistema são as raízes do numerador da função de transferência, então, neste caso, temos

$$1 - z^{-1} = 0 \Rightarrow z = 1.$$

Portanto, o sistema tem um pólo em z = 0,5 e um zero em z = 1.

Exemplo 1.14

Determine a função de transferência do sistema descrito pelo seguinte diagrama de blocos



A equação diferencial que define este sistema é

$$y(n) = 1{,}5x(n-1) - 0{,}1y(n-1) + 0{,}12y(n-2).$$

Transformando-se no domínio Z,

$$Y(z) = 1{,}5X(z)z^{-1} - 0{,}1Y(z)z^{-1} + 0{,}12Y(z)z^{-2},$$

e, portanto, a função de transferência é

$$H(z) = \frac{S(z)}{X(z)} = \frac{1{,}5z^{-1}}{1 + 0{,}1z^{-1} - 0{,}12z^{-2}}.$$

A função de transferência, H(z), de um determinado sistema, corresponde à transformada Z da resposta ao impulso, h(n), desse mesmo sistema, ou seja,

$$h(n) \overset{Z}{\longleftrightarrow} H(z). \qquad (1{,}85)$$

Na verdade, porque

$$y(n) = h(n) * x(n) = \sum_{k=-\infty}^{\infty} h(k)x(n-k),$$

Machine Translated by Google

então

$$S(z) = \sum_{n=ÿÿ}^{ÿ} y(n)z^{ÿn} = \sum_{n=ÿÿ}^{ÿ} \sum_{k=ÿÿ}^{ÿ} h(k)x(n ÿ k)z^{ÿn} = \tag{1.86a}$$

$$\sum_{k=ÿÿ}^{ÿ} h(k) \sum_{n=ÿÿ}^{ÿ} x(n ÿ k)z^{ÿn} = \sum_{k=ÿÿ}^{ÿ} h(k) \sum_{n=ÿÿ}^{ÿ} x(n)z^{ÿ(n+k)} = \tag{1.86b}$$

$$X(z) \sum_{k=ÿÿ}^{ÿ} h(k)z^{ÿk} = X(z)H(z). \tag{1.86c}$$

Este resultado mostra a relação entre a operação de multiplicação em um domínio e a operação de convolução no outro domínio.

1.5.2 Estabilidade do sistema

Sob certas condições (que representam uma classe importante de sistemas), a função de transferência pode ser escrita como

$$H(z) = \sum_{k=1}^{N} \frac{Ak\ 1}{ÿ\ pkz^{ÿ1}},$$

onde pk representa os pólos (simples7) do sistema e Ak são constantes. Como a resposta ao impulso do sistema, h(n), corresponde à transformada Z inversa de H(z), então (ver Exemplo 1.10)

$$h(n) = \sum_{k=1}^{N} Akp^{n}\ ku(n).$$

Em geral, h(n) só converge se todos os termos da soma convergirem, para o que é necessário que |pk| < 1, ÿk. Na verdade, como vimos anteriormente quando discutimos as propriedades da resposta ao impulso, para um sistema ser estável devemos ter

$$\sum_{n=ÿÿ}^{ÿ} |h(n)| < ÿ,$$

então para que esta condição seja verificada temos que garantir que |pk| < 1, ÿk, ou seja, todos os pólos do sistema devem ter módulo menor que um8 .

7Isto significa que o denominador da função de transferência tem apenas raízes simples.

8Às vezes nos referimos a esta condição de estabilidade dizendo que todos os pólos de um sistema em tempo discreto devem ser localizado dentro do círculo unitário do plano complexo.

Machine Translated by Google

1.5.3 Resposta de frequência

Visto que, para sistemas descritos por equações de diferenças lineares de coeficientes constantes, temos

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{\sum_{k=0}^{N} a_k z^{-k}} = \frac{\sum_{k=0}^{M} b_k z^{M-k}}{\sum_{k=0}^{N} a_k z^{N-k}} z^{N-M} = \frac{b_0 \prod_{k=1}^{M} (z - z_k)}{a_0 \prod_{k=1}^{N} (z - p_k)},$$

então, o módulo de sua resposta em frequência (ou ganho do sistema) pode ser escrito como[9]

$$|H(e^{j\omega})| = \frac{b_0}{a_0} \frac{\prod_{k=1}^{M} |e^{j\omega} - z_k|}{\prod_{k=1}^{N} |e^{j\omega} - p_k|}.$$

Observe que esta última expressão pode ser interpretada como o quociente entre o produto de jÿ aos zeros H(z). ponto e de H(z) e o produto das distâncias das distâncias do ponto e jÿ aos pólos de

---

Exemplo 1.15

Pretendemos calcular o módulo da resposta em frequência do sistema cuja função de transferência é

$$H(z) = \frac{1}{1 - a z^{-1}}.$$

Nós temos

$$|H(e^{j\omega})| = \left|\frac{1}{1 - a e^{-j\omega}}\right|.$$

Por exemplo, o ganho deste sistema para f = 0 é

$$|H(e^{0})| = |H(1)| = \left|\frac{1}{1 - a}\right|,$$

e para f = fs/2 é

$$|H(e^{j\omega})| = |H(-1)| = \left|\frac{1}{1 + a}\right|.$$

---

[9] Compare com os resultados obtidos na Seção 1.3.5. Observe que $e^{j\omega}$ são valores da circunferência unitária do plano complexo e são iguais aos da Equação (1.52). Portanto, a resposta em frequência de um sistema pode ser calculada usando sua função de transferência calculada para z = e jÿ, ou seja, sobre a circunferência unitária do plano complexo.

Machine Translated by Google

Para f = fs/4 temos

$$|H(e\,j\ddot{y}/2\,)| = |H(j)| = |1 + ja| \quad \frac{1}{\ddot{y}\,1 + uma^2} = \frac{1}{\ddot{y}\,1 + uma^2}.$$

Observe que, se 0 < a < 1, o sistema tem um comportamento passa-baixo (ver Fig. 1.25), enquanto que se ÿ1 < a < 0, então o sistema tem uma característica passa-alta (ver Fig. 1.26).

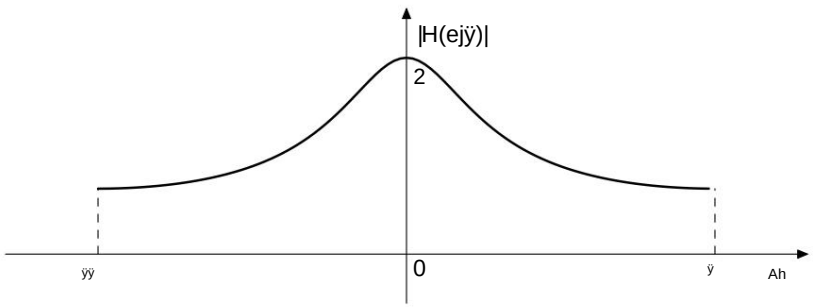


Figura 1.25: Resposta em frequência de um filtro passa-baixa digital IIR com pólo em z = 0,5.

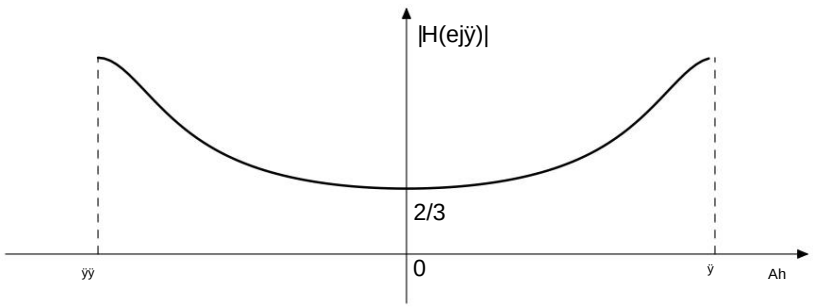


Figura 1.26: Resposta em frequência de um filtro digital passa-altas IIR com pólo em z = ÿ0,5.

Machine Translated by Google

2 Dados, informações e alguns problemas relacionados

O que são “dados”? O que é “informação”?

Esta seção aborda alguns problemas que, de alguma forma, têm a ver com essas questões.
Embora não seja esperado que você tenha respostas para todas elas agora, você nunca deve começar a pensar nelas.

Tópico de discussão 2.1
Considere as seguintes questões e tente esboçar respostas para elas:

- O que são dados?
- Como medir a quantidade de dados?
- O que é informação?
- Como medir a quantidade de informação?
- O que é um cálculo?
- Existem problemas que os computadores não conseguem resolver?

Tópico de discussão 2.2
Considere a seguinte afirmação:

> Um bit de dados contém, no máximo, um bit de informação.

Você acha que isso é verdade? Por que?

Tópico de discussão 2.3
Alice disse a Bob que encontrou um software de compactação de imagem que é capaz de reduzir o tamanho de cada imagem em, pelo menos, 50% de seu tamanho original - por exemplo, se originalmente a imagem ocupa 1.000.000 bytes, após a compactação exigirá no máximo 500.000 bytes.

Bob não ficou muito impressionado com a declaração de Alice. Porém, quando Alice acrescentou que ela também seria capaz de recuperar sempre a imagem original de sua versão compactada, Bob respondeu imediatamente:

> "ISSO É IMPOSSÍVEL! Todo método de compactação sem perdas (ou seja, reversível) é limitado, ou seja, não pode compactar todas as mensagens!”

Bob está correto? Por que?

Machine Translated by Google

Tópico de discussão 2.4

Considere o seguinte jogo, que é um caso de sistema pós-canônico10. O objetivo do jogo é descobrir a sequência de regras (dentro de um conjunto previamente especificado) que permite transformar uma determinada palavra em outra palavra.

Considere, por exemplo, o caso em que as palavras podem ser formadas usando apenas as letras M, I e U, e que as regras de transformação permitidas são:

1. xI ÿ xIU
2. Mx ÿ Mxx
3. xIIIy ÿ xUy
4. xUUy ÿ xy

Vamos tentar algumas transformações:

1. Dê os passos para transformar MI em MUI.
2. Você pode transformar UIM em MIU? Por que?
3. E MIU em UIM?
4. Quais são os passos para transformar MI em MU?

Você consegue pensar em um algoritmo que responda questões do tipo “É possível transformar a palavra w na palavra z, usando um determinado conjunto de regras”?

Tópico de discussão 2.5

Considere o conjunto de todas as funções f : N ÿ {0, 1}. Alice disse a Bob que existem funções nesse conjunto que não podem ser calculadas por nenhum programa finito, independentemente da linguagem de programação usada para implementá-lo. Bob acha que Alice está errada. Ela está errada? Por que?

Tópico de discussão 2.6

Alice quer jogar um jogo com Bob, onde uma moeda deve ser lançada. Para mostrar que está usando uma moeda honesta, ela a lança vinte vezes. No entanto, em todos os testes, surge cara. Bob pode confiar nesta moeda? Por que?

Seria diferente se a sequência de resultados fosse “00101001110101010110”? Por que?

10Este problema pode ser reformulado como um sistema de reescrita de strings, também conhecido como sistema semi-Thue, que é relacionado à noção de gramática irrestrita.

Machine Translated by Google

Tópico de discussão 2.7

Considere o seguinte procedimento, sugerido por John von Neumann (1903–1957):

1. Jogue uma moeda duas vezes
2. Se os resultados coincidirem, recomece, esquecendo os dois resultados
3. Se os resultados forem diferentes, use o primeiro resultado, esquecendo o segundo,

com o qual afirmou ser possível obter o equivalente a uma sequência gerada por uma moeda justa, mesmo que a moeda seja injusta.

você concorda com ele? Por que?

Tópico de discussão 2.8

Você trabalha em uma empresa que vende grandes sequências de números aleatórios. Seu chefe acha que precisa de alguma forma de testar a "aleatoriedade" das sequências antes de serem enviadas aos clientes (consideremos isso como uma espécie de controle de qualidade da produção...). Por ser o melhor engenheiro informático da empresa, ele pede que você desenvolva um programa para testar a aleatoriedade de sequências arbitrárias.

O que você deve responder a ele? Você tem alguma ideia de como fazer isso? Você pode definir "aleatoriedade"?

Tópico de discussão 2.9

John von Neumann disse uma vez:

> "Qualquer um que considere métodos aritméticos para produzir dígitos aleatórios está, obviamente, em estado de pecado."

O que você acha que ele quis dizer com essa afirmação?

Tópico de discussão 2.10

Considere a seguinte definição de um determinado número:

"O menor número inteiro positivo não definível em menos de doze palavras".

Esse número sai? Por que?

(Isso é conhecido como paradoxo de Berry.)

Machine Translated by Google

Tópico de discussão 2.11

Você está em uma festa com amigos. Depois de alguns drinks, um deles diz:

> "Tive uma ótima ideia! Vou escrever um programa que será capaz de analisar outros programas e nos dizer se estão livres de bugs ou não! Eu me pergunto por que ninguém fez isso antes!"

O que você diria ao seu amigo?

Tópico de discussão 2.12

Considere o seguinte segmento de pseudocódigo:

```
F(uint x) { while(x
        > 1) { if(x % 2 == 0) x =
                x / 2

                outro
                        x = 3 * x + 1
        }

        retornar
}
```

A função F retorna para todo x?

(Isso é conhecido como conjectura de Collatz.)

A seguir, serão discutidas diversas formas de medir a informação, nomeadamente, uma abordagem combinatória, uma abordagem probabilística e uma abordagem algorítmica. Além disso, serão apresentadas diversas técnicas para representar uma certa quantidade de informação no menor conjunto possível de dados. Eles são conhecidos como algoritmos de compactação de dados.

Machine Translated by Google

3 Uma medida combinatória de informação

Problema 3.1

Alice está jogando uma moeda repetidamente. Ela quer enviar os resultados para Bob. Para enviar a mensagem, Alice só consegue utilizar uma lanterna, ligando-a e desligando-a uma vez por segundo.

1. Suponha que Alice e Bob concordaram antecipadamente sobre o tamanho da mensagem, n, ou seja, sobre quantos resultados de lançamento de moeda serão enviados em cada mensagem. Proponha um procedimento para este esquema de transmissão.

2. Suponha agora que Alice pretende enviar várias mensagens de comprimento n1, n2, . . . , não conhecido antecipadamente por Bob. Você pode sugerir um procedimento de transmissão que funcione neste caso?

Problema 3.2

Alice quer enviar uma carta para Bob, usando o mesmo esquema de transmissão. Considere um alfabeto de 27 símbolos (26 letras mais o espaço). Como isso pode ser feito?

Problema 3.3

Agora Alice quer jogar com Bob (eles finalmente estão cara a cara!). Ela pensa em um número entre 1 e 100. Bob pode fazer todas as perguntas que quiser, mas só recebe sim ou não.
responder.

1. Qual é o número mínimo de perguntas que Bob precisa fazer sequencialmente para Alice (a próxima pergunta só é feita depois de saber a resposta da pergunta anterior), para descobrir o número em que Alice pensou? Quais deveriam ser essas perguntas?

2. Agora Bob só pode fazer todas as perguntas de uma vez. O número mínimo de perguntas obrigatórias é o mesmo? Neste caso, quais devem ser as perguntas?

O número mínimo de questões sequenciais é alcançado, por exemplo, reduzindo sucessivamente pela metade o conjunto de possibilidades (uma espécie de busca binária...). Tomando como exemplo o jogo de adivinhar o número, temos

$$^{b2} = 100,$$

onde b é o número de perguntas. Por isso,

$$b = \log2\ 100\ ÿ\ 6{,}644,$$

o que significa que, em média, são necessárias cerca de 6.644 perguntas para encontrar o número – em alguns casos, seis perguntas são suficientes, enquanto em outros casos são necessárias sete perguntas.

Machine Translated by Google

Exemplo 3.1
Adivinhe um número entre 1 e 100:

É x ÿ 50? ÿ 1 É x ÿ
75? ÿ 0 É x ÿ 62?
ÿ 1 É x ÿ 68? ÿ 0 É
x ÿ 65? ÿ 0 É x ÿ
63? ÿ 1 É x = 63?
ÿ 0

Portanto, x deve ser 64 (são necessárias sete perguntas sim/não).

Exemplo 3.2
Adivinhe um número entre 1 e 100:

É x ÿ 50? ÿ 1 É x ÿ
75? ÿ 0 É x ÿ 62?
ÿ 1 É x ÿ 68? ÿ 0 É
x ÿ 65? ÿ 1 É x ÿ
66? ÿ 0

Portanto, x tem que ser 65 (neste caso, seis perguntas sim/não foram suficientes).

Problema 3.4

Utilizando este esquema de pesquisa, o número médio de perguntas necessárias é 6,72. Verifique este valor (calculando quantos dos 100 números exigem que as sete questões sejam definidas). Por que esse valor é maior que o limite teórico de log2 100 ÿ 6,644 bits?

Agora imagine que x é representado em binário. Quais deveriam ser as questões “naturais” a serem colocadas neste caso? Usando essa abordagem, você consegue atingir um número médio de questões menor que 6,72?

Observe que, se todas as perguntas tivessem que ser feitas de uma vez, então no exemplo de adivinhar um número entre 1 e 100 teríamos que fazer sempre um mínimo de ÿlog2 100ÿ = 7 perguntas.

Problema 3.5

Suponha agora que o número a ser adivinhado esteja entre 1 e 3. Compare e discuta o impacto no número médio de questões que devem ser feitas se:

Machine Translated by Google

(a) As perguntas são colocadas sequencialmente, depois de conhecidas as respostas anteriores;

(b) As perguntas são fornecidas em lote.

Discuta também aspectos como o caso em que os números não são igualmente prováveis e também a diferença entre log2 3 ÿ 1,585 e o número médio de questões estimadas para os diversos cenários abordados.

Programação 3.1

Escreva um programa que gere a sequência mínima de respostas sim/não que permita descobrir um determinado número entre 1 e n.

Problema 3.6

Usando as ideias acima, mostre que, em geral, a ordenação tem um limite inferior médio de operações de comparação ÿ(n log n).

Problema 3.7

Como o jogo estava ficando chato, Alice diz a Bob que ela poderia começar a mentir, no máximo, em uma das perguntas sim/não que ela está respondendo. Você pode ajudar Bob a descobrir o número mínimo de perguntas adicionais que ele precisa fazer para compensar uma possível resposta errada de Alice?

Para podermos corrigir um possível erro, podemos utilizar o esquema simples de colocar três vezes a mesma questão e depois decidir por maioria (neste caso, devemos utilizar um total de 21 questões). Essa abordagem é conhecida como **código de repetição.** Se as questões forem colocadas sequencialmente, é fácil perceber que este número pode ser reduzido, no pior dos casos, para 15 (Como?). Além disso, mesmo que todas as questões sejam colocadas de uma só vez, é possível resolver o problema com apenas 11 questões!

Para ver como isso pode ser feito, precisamos primeiro estar convencidos de que a correção de erros de um bit é possível se cada **palavra-código** diferir pelo menos em três bits de todas as outras palavras-código possíveis (isto é, a **distância de Hamming** entre palavras-código é de pelo menos três). Portanto, cada palavra-código precisa ter uma "região proibida" de raio um ao seu redor (conhecida como bola de raio um):

- Uma string de n bits possui n + 1 strings a uma distância de Hamming menor ou igual a um.
- Para k bits de dados, precisamos de 2^k palavras-código e, portanto, 2^k bolas disjuntas.
- Isso requer 2^k (n + 1) strings distintas de n bits.
- Portanto,

$$2^n \text{ k ÿ } 2 \text{ (n + 1), e k ÿ n ÿ log2 (n + 1).}$$

Machine Translated by Google

No caso do nosso problema (k = 7), esta condição é satisfeita para n = 11. Um código eficaz pode ser construído usando ideias propostas inicialmente por Richard Hamming (1915–1998) (Hamming, 1950). Embora sem entrar em muitos detalhes, podemos ver os códigos de Hamming construídos pela introdução de bits de paridade (calculados de acordo com regras específicas) nas posições da palavra-código que são potências de dois. No nosso caso, as palavras-código teriam a forma

p1|p2|d1|p3|d2|d3|d4|p4|d5|d6|d7

onde pi são os bits de paridade e di os bits de dados11, originando um código com taxa de 7/11 ÿ 0,64.

Problema 3.8

Três jogadores usando chapéus entram em uma sala. Os chapéus podem ser pretos ou brancos e são atribuídos jogando uma moeda justa. As regras do jogo são:

- Nenhum jogador pode ver o seu próprio chapéu, mas pode ver o chapéu de todos os outros jogadores.
- Quando estão na sala, os jogadores não conseguem se comunicar. Porém, antes de ter os chapéus, eles podem decidir sobre uma estratégia de jogo.
- Cada jogador pode anunciar o seu palpite relativamente à cor do seu chapéu ou passar.
- Todos os jogadores farão o anúncio ou passarão simultaneamente.

O grupo ganha o jogo se pelo menos uma pessoa acertar e ninguém acertar.

Qual a melhor estratégia para dar ao grupo a maior probabilidade possível de vencer o jogo?
Você pode relacionar esse problema com um código de Hamming de comprimento três?

Com n = 3, podemos formar $2^3 = 8$ strings diferentes de 3 bits. Suponha que escolhamos as strings 000 e 111 como possíveis palavras-código. Então, o conjunto de 8 strings de 3 bits forma um código Hamming(3, 1) (verifique isso). Se cada uma das strings de 3 bits ocorrer com igual probabilidade, então há 1/4 de probabilidade de gerar uma palavra-código correta (2 em 8) e 3/4 de probabilidade de gerar uma palavra-código errada (6 em 8). Portanto, a estratégia do jogo é a seguinte:

1. Se o jogador vir os outros dois jogadores com a mesma cor de chapéu, então ele assume que tem uma cor de chapéu diferente, pois este seria o caso mais provável. Portanto, ele anuncia essa cor.

11É claro que a posição dos bits de paridade é arbitrária, desde que tanto o codificador quanto o decodificador conheçam seus posições. Por exemplo, o Hamming(7, 4) pode ser obtido usando

d1, d2, d3, d4, d2 ÿ d3 ÿ d4, d1 ÿ d3 ÿ d4, d1 ÿ d2 ÿ d4.

Machine Translated by Google

2. Se o jogador vir os outros dois jogadores com chapéus de cores diferentes, então, como nesse caso a probabilidade de o seu chapéu ter qualquer uma das duas cores é a mesma, ele passa.

Observe que, utilizando esta estratégia, a probabilidade de escolher a cor errada ainda é de 50% cada vez que alguém anuncia uma cor. O ganho é que os palpites errados não são distribuídos uniformemente – do total de doze anúncios associados às oito combinações, os seis errados concentram-se em apenas dois casos; todos os outros seis casos estão livres de erros.

Trabalho de casa

3.1 Tente descobrir qual é a maior probabilidade de ganhar o jogo quando há sete jogadores.
Dica: use o código Hamming(7, 4). Qual deve ser a estratégia de jogo neste caso?

Os problemas que temos abordado consideram uma medida de informação que normalmente é chamada de **combinatória,** porque se preocupa apenas com o número de possíveis objetos envolvidos, assumindo que eles ocorrem com igual probabilidade. Portanto, para m objetos distintos (um alfabeto de tamanho m), a quantidade de **informação combinatória** necessária para especificar cada um deles é

$$\log_2 m \text{ bits.}$$

Esta abordagem para medir informações pode ser rastreada pelo menos até os trabalhos de Nyquist e Hartley:

- Harry Nyquist (1889–1976), Certos Fatores que Afetam a Velocidade Telegráfica, Sistema Bell Jornal Técnico (Nyquist, 1924):

  > "Este artigo considera dois fatores fundamentais que influenciam a velocidade máxima de **transmissão de inteligência** por telégrafo. Esses fatores são a formação do sinal e a escolha dos códigos."

- Ralph Hartley (1888–1970), Transmissão de Informações, Bell System Technical Journal (Hartley, 1928):

  > "É desenvolvida uma medida quantitativa de "informação" que se baseia em considerações físicas em contraste com considerações psicológicas."

Em seu artigo, Hartley propõe medir o conteúdo de informação, H, de uma mensagem como

$$H = n \log m = \log mn \quad ,$$

onde m denota o número de símbolos possíveis (o tamanho do alfabeto) en o número de símbolos na mensagem. Assim, Hartley definiu a quantidade de informação em uma mensagem como o logaritmo do número de mensagens possíveis (espaço de mensagem).

A base do logaritmo determina a unidade de informação utilizada: **hartley** para base 10, **nat** para base e, **bit** para base 2.

Machine Translated by Google

Problema 3.9

Alice tem um baralho de cartas (52 cartas). Depois de embaralhar o baralho, Alice quer enviar informações suficientes para Bob para que ele coloque seu baralho na mesma ordem. Quantos bits Alice precisa enviar a Bob para comunicar essas informações? Você pode propor um método por fazer isso?

Machine Translated by Google

4 Uma medida probabilística de informação

Voltemos a Alice e Bob e consideremos o seguinte novo problema:

Problema 4.1

Usando o sistema de comunicação com lanterna, Alice deseja comunicar a Bob os resultados do lançamento de um par de dados (ou seja, um número de 2 a 12). Porém, Alice está ficando sem pilhas para a lanterna e, por isso, quer preservá-las o máximo possível. Então, ela se pergunta se, neste caso, é possível usar, em média, menos que log2 11 ÿ 3,46 bits para enviar a Bob o resultado de cada tentativa. . . Você pode ajudá-la a resolver esse problema?

O primeiro aspecto a levar em consideração é que os resultados possíveis do lançamento de dois dados não ocorrem com a mesma frequência. Denotemos por ÿ o alfabeto que representa esses resultados, ou seja, ÿ = {2, 3, . . . , 12}. Por exemplo, existe apenas uma combinação para obter um doze, enquanto para obter um sete existem seis combinações possíveis. Conseqüentemente, Alice terá que transmitir o símbolo “7” cerca de seis vezes mais frequentemente do que o símbolo “12” do alfabeto.

A ideia chave é usar representações mais curtas, ou seja, menos bits, para os símbolos mais frequentes.

O código Morse (século XIX) é um bom exemplo deste princípio, reconhecido muito antes dos fundamentos da teoria da informação serem estabelecidos por Claude Shannon (1916–2001) em meados do século XX:

- Claude Shannon, Uma Teoria Matemática da Comunicação, Bell System Technical Diário (Shannon, 1948):

  > “O recente desenvolvimento de vários métodos de modulação, como PCM e PPM, que trocam largura de banda pela relação sinal-ruído, intensificou o interesse em uma teoria geral da comunicação. A base para tal teoria está contida nos importantes artigos de Nyquist e Hartley sobre este assunto. No presente artigo iremos alargar a teoria para incluir uma série de novos factores, em particular o efeito do ruído no canal, e as poupanças possíveis devido à estrutura estatística da mensagem original **e** devido à natureza do destino final da mensagem. a informação."

Tópico de discussão

4.1 O código Morse é composto por sequências de pontos e traços de vários comprimentos, de acordo com a convenção mostrada na Figura 4.1. Neste código, os caracteres de uma palavra precisam ser separados por espaço adicional (três unidades de tempo, onde uma unidade de tempo equivale ao comprimento de um ponto). Por exemplo, as letras “ET” originam a sequência “· —”, para distinguir da letra “A” que é codificada como “· —”. Você acha que essa separação explícita poderia ser evitada? Se você pensa assim, como o código deve ser construído?

Machine Translated by Google

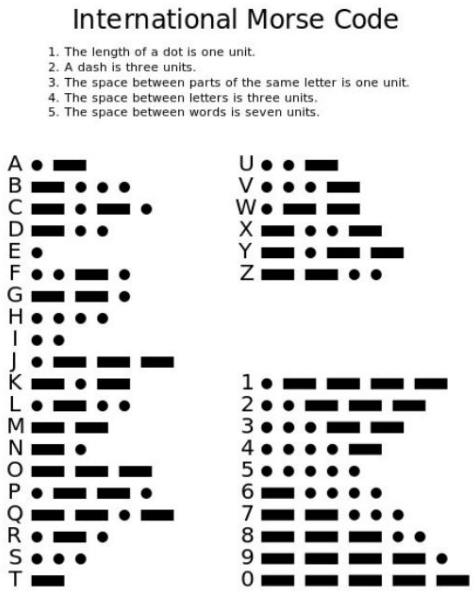


Figura 4.1: Tabela em código Morse.

4.1 Sobre códigos

Considere um alfabeto com m símbolos, ÿ = {ÿ1, ÿ2, . . . , ÿm}, e as correspondentes **frequências estimadas de ocorrência** de cada símbolo, pi . Claro,

$$\sum_{eu=1}^{eu} pi = 1.$$

Se o símbolo ÿi ÿ ÿ for representado por uma palavra-código wi , então o objetivo é **minimizar**

$$\sum_{eu=1}^{eu} pi\text{-}li \quad , \tag{4.1}$$

onde li ÿ N + é o comprimento da palavra-código wi , ou seja, seu número de bits. Freqüentemente, representaremos o comprimento de uma string binária wi por |wi |, ou seja, li = |wi |.

Obviamente, precisamos impor restrições adicionais à minimização de (4.1), caso contrário poderíamos chegar a soluções completamente inúteis, como li = 1, ÿi . . .

Tópico de discussão

4.2 Você pode sugerir restrições que deveriam ser impostas na minimização de (4.1)?

Claro que uma das restrições deveria ser que wi ÿ= wj , ÿiÿ=j , caso contrário o código não é reversível, ou seja, não é possível recuperar a mensagem original. Mas esta condição não é

Machine Translated by Google

o suficiente para reversibilidade (você consegue ver por quê?). Na verdade, precisamos de uma condição mais forte: o código deve ser **decodificável de forma única.** Isto implica que duas mensagens diferentes sempre terão duas codificações diferentes. Lembre-se da codificação de "ET" e "A" usando o código Morse da Figura 4.1 e a necessidade de espaço adicional para separar as letras. Sem ela, duas mensagens diferentes originariam a mesma codificação, ou seja, "· —".

Observe que, para nossos propósitos, uma **codificação** é a atribuição de strings binárias a elementos do alfabeto. Algumas codificações têm **comprimento fixo** (a codificação ASCII é um exemplo óbvio), enquanto outras codificações têm **comprimento variável.** Para obter compressão, precisamos do último
uns.

Problema 4.2

Encontre códigos para os quais wi ÿ= wj , ÿiÿ=j , mas que não sejam exclusivamente decodificáveis.

Uma condição suficiente (mas não necessária) para a decodificação única é **a decodificação imediata.**
Isto significa que o símbolo ÿi é imediatamente determinado assim que o último bit (ou seja, o mais à direita) de wi é lido durante a decodificação. Para que isso aconteça, as palavras-código precisam ser **livres de prefixo,** ou seja, uma palavra-código mais curta não pode ser um prefixo de uma palavra-código mais longa (por que precisamos desse requisito?).

Problema 4.3

Mostre que o código Morse não é um **código sem prefixo** (Sugestão: tente construir uma árvore de decodificação).
Quais são as implicações práticas desta limitação do código Morse?

Letters	Probability	Code 1	Code 2	Code 3	Code 4
a_1	0.5	0	0	0	0
a_2	0.25	0	1	10	01
a_3	0.125	1	00	110	011
a_4	0.125	10	11	111	0111
Average length		1.125	1.25	1.75	1.875

(Saood 2012)

Tabela 1: Exemplo de vários códigos, com propriedades diferentes.

A Tabela 1 mostra quatro códigos diferentes para um alfabeto de 4 símbolos (embora nem todos sejam úteis...). Eles têm propriedades diferentes, a saber:

- O código 1 não é exclusivamente decodificável.
- O código 2 também não é exclusivamente decodificável.
- O Código 3 é exclusivamente decodificável e instantâneo.

Machine Translated by Google

- O código 4 é exclusivamente decodificável, mas não instantâneo.

Problema 4.4

Considere agora o seguinte código exclusivamente decodificável, mas não instantâneo:

Palavra-código do símbolo	
ÿ1	0
ÿ2	01
ÿ3	11

Como decodificar a string "01111111111111111"?
(Observe que o primeiro símbolo pode ser ÿ1 ou ÿ2. . .)

Problema 4.5

Considere agora o seguinte código:

Palavra-código do símbolo	
ÿ1	0
ÿ2	01
ÿ3	10

Como decodificar a string "01010101010101010"?
(Este código não é instantâneo nem decodificável exclusivamente)

Algoritmo 4.1 (decodificabilidade única)

Suponha que temos duas palavras-código binárias a e b, com k = |a|, n = |b|, com k < n, e onde a é um prefixo de b. Referimo-nos aos últimos n ÿ k bits de b como o sufixo pendente.

O algoritmo para descobrir se um código é exclusivamente decodificável é o seguinte:

- Construa um conjunto com todas as palavras-código.
- Para cada par de palavras-código, gere o sufixo pendente (se existir) e adicione-o ao definir.
- Repita até: (a) obter um sufixo pendente que é uma palavra-código; (b) Não há mais sufixos pendentes exclusivos.

Se o algoritmo terminar com a condição (a), ou seja, se encontrar um sufixo pendente que seja uma palavra-código, então o código não será decodificável exclusivamente.

Exemplo: {0, 01, 10} ÿ {0, 01, 10, 1} ÿ {0, 01, 10, 1, 0}.

Machine Translated by Google

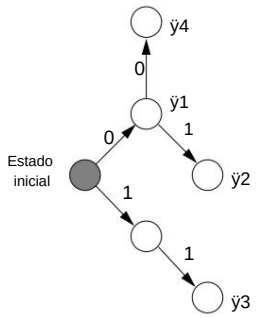


Figura 4.2: Exemplo de uma árvore de decodificação de um código não instantâneo (e também não decodificável exclusivamente).

Se o algoritmo terminar com a condição (b), ou seja, se não houver mais sufixos pendentes exclusivos, então o código será decodificável exclusivamente.

Exemplo: {0, 01, 11} ÿ {0, 01, 11, 1}.

Conforme mencionado, se, em uma determinada codificação, nenhuma das palavras-código for um prefixo de outra palavra-código, então o código é necessariamente decodificável de forma única. Uma maneira fácil de verificar se um código é livre de prefixo é construir a árvore binária enraizada correspondente. Em um código sem prefixo, as palavras-código estão associadas apenas aos nós externos da árvore.

Exemplo 4.1

Considere um alfabeto com símbolos ÿ1, ÿ2, ÿ3 e ÿ4, e a seguinte atribuição de palavra-código: ÿ1 ÿ 0, ÿ2 ÿ 01, ÿ3 ÿ 11, ÿ4 ÿ 00.

Se o decodificador receber “0001”, ele não poderá determinar se o codificador enviou a string “ÿ1ÿ1ÿ2” ou “ÿ4ÿ2”. Como pode ser visto usando a árvore de decodificação correspondente (Fig. 4.2), este código apresenta vários problemas:

- O caminho para o nó ÿ3 é feito por um nó intermediário que não contém bifurcação: o código é ineficiente.

- O nó ÿ1 não é um nó terminal. Os caminhos para os nós ÿ2 e ÿ4 incluem o nó ÿ1: o código não é instantâneo (na verdade, também não é exclusivamente decodificável, como pode ser visto usando o Algoritmo 4.1).

Exemplo 4.2

Consideremos agora que cada ramo da árvore se conecta a um nó terminal ou a um nó de decisão que leva a um nó terminal. Com base nesta restrição, chegamos à seguinte atribuição de palavras de código: ÿ1 ÿ 0, ÿ2 ÿ 10, ÿ3 ÿ 110, ÿ4 ÿ 111. Como pode ser confirmado (ver Fig. 4.3), este código é exclusivamente decodificável (e também instantâneo).

Machine Translated by Google

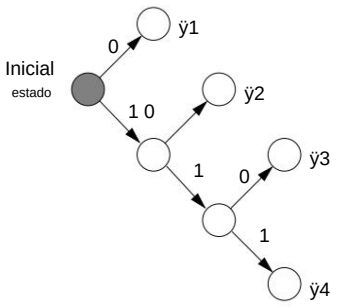


Figura 4.3: Exemplo de árvore de decodificação de um código unicamente decodificável (instantâneo). Este código não contém prefixo.

Outra forma de verificar se um código é livre de prefixo é mapear as palavras-código no intervalo unitário, onde cada palavra-código ocupa uma fração não sobreposta de comprimento 2 ÿli do intervalo unitário (observe que quando uma palavra-código de comprimento k é usada , uma fração 2 da árvore binária associada é "obtida", devido à condição sem prefixo).
ÿk

Problema 4.6

Suponha que Alice queira construir um código sem prefixo tal que l1 = l4 = 2, l2 = l3 = 3 e l5 = 1. Você pode mostrar a Alice que isso não é possível? (Tente ambas as analogias: a árvore binária e a cobertura do intervalo unitário)

Tópico de discussão 4.3 Dado

um conjunto de comprimentos, li , em ordem crescente, discuta estratégias para cobrir o intervalo unitário da forma mais eficiente possível. A cobertura pode ser feita de forma tão eficiente como anteriormente, mesmo que os comprimentos sejam dados em ordem arbitrária?

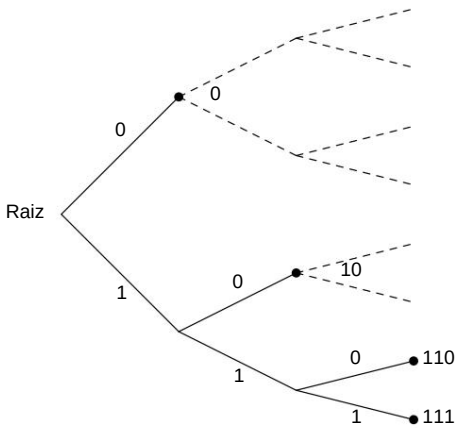


Figura 4.4: Árvore associada a um código sem prefixo.

Machine Translated by Google

Teorema 4.1 (desigualdade de Kraft)
Para qualquer código binário instantâneo (ou seja, sem prefixo), a palavra-código tem comprimento l1, l2, . . . , lm deve satisfazer a desigualdade

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

Prova. Referindo-se à árvore de código da Figura 4.4, considere todos os nós no nível lmax (o comprimento da palavra-código mais longa). Uma palavra-código no nível li tem 2 descendentes lmax-li no nível lmax. Devido à condição livre de prefixo, cada um desses conjuntos descendentes deve ser disjunto. Além disso, o número total de nós nesses conjuntos deve ser menor ou igual a 2 lmax . Portanto, somando todas as palavras-código, temos $\sum 2^{l_{max}-l_i} \leq 2^{l_{max}}$ ou $\sum 2^{-l_i} \leq 1$. □

Teorema 4.2 (contraparte da desigualdade de Kraft)
Se um conjunto de comprimentos l1, . . . , Se satisfizermos a desigualdade de Kraft, então existe um instante l2, . código com esses comprimentos de palavra.

Prova. Construa uma árvore de profundidade lmax. Rotule o primeiro nó (lexicograficamente) de profundidade l1 como palavra-código 1 e remova seus descendentes da árvore. Em seguida, rotule o primeiro nó restante de profundidade l2 como palavra-código 2 e assim por diante. Procedendo desta forma, construímos um código de prefixo com os l1, l2, especificados . . . , lm.

□

4.2 Definição de informação segundo Shannon

A definição de **informação** segundo Shannon está associada à probabilidade de determinados eventos (os símbolos). Considere E um evento, ou seja, um conjunto de resultados de algum experimento aleatório.
Se P(E) é a probabilidade de ocorrência do evento E, então a informação (também chamada por Shannon de "autoinformação") associada à ocorrência de E é dada por

$$I(E) = \log_b \frac{1}{P(E)} = -\log_b P(E).$$

Lembre-se de que log(1) = 0 e que ÿ log(x) aumenta à medida que x diminui de um para zero. Portanto, um evento com probabilidade um não carrega nenhuma informação. Por outro lado, à medida que a probabilidade do evento se aproxima de zero, a informação que pode ser associada a esse evento aproxima-se do infinito.

Além disso, a informação associada à ocorrência de dois eventos **independentes ,** E1 e

Machine Translated by Google

E2, [12] é i(E1, E2) = i(E1) + i(E2), porque

$$i(E1, E2) = ÿ \log b\, P(E1, E2) = ÿ \log b\, P(E1)P(E2) =$$

$$= ÿ \log b\, P(E1)\, ÿ \log b\, P(E2) = i(E1) + i(E2).$$

Como mencionado anteriormente, a unidade de informação depende da base do logaritmo, b: se b = 2, medimos a informação em bits; se b = e, em nats; se b = 10, em Hartleys. A partir de agora, quando a base do logaritmo não for especificada, assumimos b = 2.

Para uma determinada partição do espaço amostral, S, em conjuntos Ej , ou seja, se

$$\underset{j}{} Ej = S \text{ e } Ej\, ÿ\, Ek = ÿ,\ ÿj\, ÿ= k,$$

a **autoinformação média** associada é dada por

$$H = \underset{j}{} P(Ej)i(Ej) = - \underset{j}{} P(Ej) \log P(Ej),$$

mais comumente chamado **de entropia.** Observe que a entropia é completamente definida por uma distribuição de probabilidade – daí o nome **"probabilística"** nesta definição de informação.

Exemplo 4.3

Considere o lançamento de uma moeda honesta. Neste caso, temos:

- Dois símbolos (ou seja, ÿ = {cara, coroa}), Pi = 0,5
- H = ÿ(0,5 log 0,5 + 0,5 log 0,5) = 1

  =ÿ Um bit para cada tentativa
- R = log |ÿ| ÿ H = log 2 ÿ 1 = 0

  Isso é chamado de **redundância** e mede a **diferença entre as medidas de informação combinatória e probabilística.**

Exemplo 4.4

O caso de duas moedas honestas e independentes:

- Quatro símbolos, Pi = 0,25
- $H = - \underset{eu=1}{\overset{4}{}} Pi \log Pi = 2$ bits

12Lembre-se de que se E1 e E2 são independentes, então P(E1, E2) = P(E1)P(E2).

Machine Translated by Google

- R = log |ÿ| ÿ H = log 4 ÿ 2 = 0

Exemplo 4.5

Duas moedas independentes, mas tendenciosas:

- Quatro símbolos, P1 = 0,5625, P2 = P3 = 0,1875 e P4 = 0,0625
- H ÿ 1,62 bits
- R = log |ÿ| ÿ H = log 4 ÿ 1,62 = 0,38
- Neste caso, representar cada símbolo requer em média apenas 1,62 bits.
- Isto significa que, por exemplo, 1.000 resultados do processo podem ser representados por aproximadamente 1.620 bits, em vez de 2.000 bits, como sugerido pela definição combinatória de informação.
- Como já mencionado, esta redução de dados pode ser obtida utilizando **códigos de comprimento variável.**

Shannon demonstrou que, se uma **fonte de informação** produz eventos com probabilidades P(Ei), onde {Ei} é uma partição de S, então, **em média,** não é possível usar menos que H bits para representar cada evento.

Normalmente, e para simplificar, rotulamos os eventos Ei com símbolos de um alfabeto ÿ que é frequentemente um subconjunto dos inteiros, por exemplo, ÿ = {1, 2, . . . , m}. No entanto, por uma questão de generalidade, usaremos o mais genérico ÿ = {ÿ1, ÿ2, . . . , ÿm} alfabeto.

Suponhamos que uma fonte de informação produza sequências de símbolos X = {X1, X2, . . . }, com[13] Xi ÿ ÿ. A entropia de enésima ordem, Hn, é definida como

$$Hn = - \frac{1}{n} \sum_{x1=ÿ1}^{ÿm} \cdots \sum_{xn=ÿ1}^{ÿm} P(Xk+1 = x1, \ldots, Xk+n = xn) \log P(Xk+1 = x1, \ldots, Xk+n = xn),$$

onde {Xk+1, Xk+2, . . . , Xk+n} são subsequências de comprimento n geradas pela fonte de informação. Shannon mostrou que, para uma **fonte estacionária,**

$$H = \lim_{nÿÿ} Hn,$$

onde H é a **entropia da fonte.** Observe que, se os eventos são **independentes e distribuídos de forma idêntica** (iid), então

$$H = H1 = Hn = ÿ \sum_{x=ÿ1}^{ÿm} P(Xk = x) \log P(Xk = x), ÿk, \tag{4.2}$$

ou seja, a entropia e a entropia de 1ª ordem da fonte são iguais.

13X é um processo aleatório em tempo discreto e, portanto, os Xi são variáveis aleatórias.

Machine Translated by Google

Problema 4.7

Mostre que (4.2) vale ÿn, se os eventos forem independentes e distribuídos de forma idêntica.

Geralmente, a entropia de uma fonte de informação não é conhecida. Portanto, temos que calcular **estimativas** para isso.

4.3 Modelagem de origem

Considere a seguinte sequência de símbolos, produzida por uma determinada fonte de informação:

1 2 3 2 3 4 5 4 5 6 7 8 9 8 9 10

Se tomarmos isso como uma "boa" representação das estatísticas da fonte de informação, então podemos estimar as probabilidades dos símbolos como

P(1) = P(6) = P(7) = P(10) = 1/16 P(2) = P(3)
= P(4) = P(5) = P(8) = P(9) = 16/02.

Assumindo iid, a entropia desta fonte é de 3,25 bits.

No entanto, se transformarmos a sequência original,

1 2 3 2 3 4 5 4 5 6 7 8 9 8 9 10

em

1 1 1 -1 1 1 1 -1 1 1 1 1 1 -1 1 1

agora temos apenas dois símbolos diferentes (ÿ1 e 1), para os quais P(1) = 13/16 e P(ÿ1) = 3/16. Agora, a entropia é de apenas 0,7 bits. . .

Tópico de discussão

4.4 Vimos que a entropia é uma medida da média da informação produzida por uma fonte.
De acordo com o exemplo acima, aparentemente é possível alterar a quantidade de informação produzida por uma fonte, utilizando uma transformação reversível (mostre que a transformação utilizada acima é, de fato, reversível).

Você acha que isso é realmente possível? [14]

Denotemos pelo $_1^n$ = x1x2 . . . xn, xi ÿ ÿ, a sequência de saídas (símbolos da fonte (ou alfabeto x ÿ) que uma fonte de informação foi gerada até o instante n. Muitas vezes nos referimos a x $_1^n$

14Na verdade, não é! Então, o que há de errado nesse raciocínio?

Machine Translated by Google

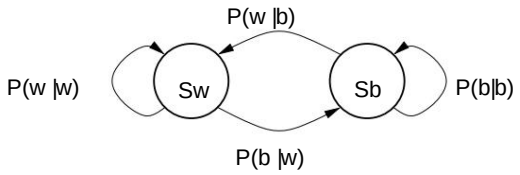


Figura 4.5: Exemplo de diagrama de transição de um modelo de Markov para uma imagem binária.

simplesmente x, se o tamanho puder ser omitido) como uma string sobre o alfabeto ÿ. A transformação realizada no exemplo acima corresponde à observação de que a relação

$$xn = xnÿ1 + rn$$

fornece um modelo "satisfatório" para esta fonte de informação. Nesse caso, apenas os valores de rn precisam ser codificados. Este modelo é denominado estático porque não muda com n. Caso contrário, o modelo é denominado adaptativo.

Tópico de discussão 4.5
Como sabemos que um determinado modelo é "satisfatório"? Por exemplo, por que é razoável dizer que o modelo acima é "satisfatório"?

Modelar corretamente uma fonte de informação (e, portanto, encontrar a estrutura subjacente dos dados) é um dos aspectos mais importantes na compressão de dados.

Uma das abordagens mais utilizadas para representar dependências de dados baseia-se no uso de modelos de Markov. Na compactação de dados sem perdas, usamos um tipo específico, chamado cadeia de Markov de tempo discreto ou modelo de contexto finito.

Um modelo de Markov de ordem k verifica

$$P(xn|xnÿ1 \ldots xnÿk) = P(xn|xnÿ1 \ldots xnÿk \ldots),$$

onde a string c = xnÿ1 . . . xnÿk é chamado de estado ou contexto do processo.

Um modelo de Markov de 1ª ordem se reduz a

$$P(xn|xnÿ1) = P(xn|xnÿ1xnÿ2 \ldots).$$

Por exemplo, considere que queremos modelar uma imagem binária. Temos dois estados, Sw (pixel branco) e Sb (pixel preto), e quatro transições possíveis, nomeadamente Sw ÿ Sw, Sw ÿ Sb, Sb ÿ Sw, Sb ÿ Sb. O diagrama de transição de estado deste modelo de 1ª ordem é mostrado na Figura 4.5.

A entropia de um processo com N estados Si é simplesmente o valor médio da entropia de cada estado, ou seja,

$$H = \sum_{eu=1}^{N} P(Si)H(Si),$$

Machine Translated by Google

onde H(Si) denota a entropia do estado Si e Pi é a probabilidade de ocorrência do estado Si . Portanto, em nosso exemplo,

$$H(Sw) = ÿP(b|w) \log P(b|w) ÿ P(w|w) \log P(w|w).$$

Exemplo 4.6
Considere um modelo de 1ª ordem com

$$P(w|b) = 0,3 \text{ e } P(b|w) = 0,01.$$

As probabilidades de estado podem ser calculadas de acordo com

$$P(Sw) = ÿ \frac{P(w|b)}{0,968\, P(w|b) + P(b|w)}$$

e

$$P(Sb) = \frac{P(b|w)}{P(w|b) + P(b|w)} ÿ 0,032.$$

Portanto, considerando a independência na ocorrência de símbolos, temos,

$$H = ÿ0,968 \log_2 0,968 ÿ 0,032 \log_2 0,032 ÿ 0,204 \text{ bps}$$

Porém, considerando o modelo de Markov de 1ª ordem, obtemos uma estimativa da entropia da fonte que é

$$H(Sb) = ÿ0,3 \log_2 0,3 ÿ 0,7 \log_2 0,7 ÿ 0,881 \text{ bps}$$

$$H(Sw) = ÿ0,01 \log_2 0,01 ÿ 0,99 \log_2 0,99 ÿ 0,081 \text{ bps}$$

$$H = 0,968 \times 0,081 + 0,032 \times 0,881 ÿ 0,107 \text{ bps}$$

Os modelos de Markov são particularmente úteis na compressão de texto, porque a próxima letra de uma palavra é geralmente fortemente influenciada pelas letras anteriores. Na verdade, o uso de modelos de Markov para o inglês escrito apareceu na obra original de Shannon. Em 1951, ele estimou que a entropia do inglês estava entre 0,6 e 1,3 bits por letra.

Para simplificar, considere apenas 26 letras e o caractere de espaço. A frequência das letras em inglês está longe de ser uniforme: a mais comum, “E”, ocorre cerca de 13%, enquanto as menos comuns, “Q” e “Z”, ocorrem cerca de 0,1% das vezes. A frequência dos pares de letras também não é uniforme. Por exemplo, a letra “Q” é sempre seguida por um “U”. O par mais frequente é “TH” (ocorre cerca de 3,7%).

Podemos utilizar a frequência dos pares para estimar a probabilidade de uma letra seguir qualquer outra letra. Este raciocínio também pode ser usado para construir modelos de ordem superior. No entanto, o tamanho do modelo cresce exponencialmente. Por exemplo, para construir um modelo de Markov de terceira ordem, precisamos estimar os valores de p(xn|xnÿ1xnÿ2xnÿ3). Isto requer uma tabela com 274 = 531.441 entradas e texto suficiente para estimar corretamente as probabilidades.

Os exemplos a seguir foram construídos usando distribuições empíricas coletadas de amostras de texto (Shannon, 1948, 1951).

Machine Translated by Google

Exemplo 4.7 (Aproximação de ordem zero (símbolos independentes e equiprováveis))

XFOML RXKHRJFFJUJ ZLPWCFWKCYJ FFJEYVKCQSGXYD
QPAAMKBZAACIBZLHJQD

Entropia: log 27 = 4,76 bits por letra.

Exemplo 4.8 (Aproximação de primeira ordem (Independente, mas com o p(x)) correto

OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI ALHENHTTPA OOBTTVA NAH
reais

Entropia estimada: ÿ 4,03 bits por letra.

Exemplo 4.9 (Aproximação de segunda ordem (modelo de Markov de 1ª ordem, ou seja, p(xn|xnÿ1)))

NO IE ANTSOUTINYS SÃO T INCTORE ST BE S DEAMY ACHIN D ILONASIVE
TUCOOWE EM TEASONARE FUSO TIZIN ANDY TOBE SEACE CTISBE

Entropia estimada: ÿ 3,32 bits por letra.

Exemplo 4.10 (Aproximação de terceira ordem (modelo de Markov de 2ª ordem, ou seja, p(xn|xnÿ1xnÿ2)))

IN NO IST LAT WHEY CRATICT FROURE BERS GROCID PONDENOME DE
DEMONSTURAS DA REPTAGINA É REGOAÇÃO DA CRE

Entropia estimada: ÿ 3,1 bits por letra (excluindo espaços).

Exemplo 4.11 (Aproximação de palavras de primeira ordem)

REPRESENTAR E VELOCIDADE É UM BOM APTO OU VEM PODE DIFERENTE
NATURAL AQUI ELE VEIO PARA O ESPECIALISTA GREY VEM PARA
FORNECE QUE A MENSAGEM DA LINHA ERA ESTAS

Exemplo 4.12 (aproximação de palavras de segunda ordem)

A CABEÇA E NO ATAQUE FRONTAL A UM ESCRITOR INGLÊS QUE O
O CARÁTER DESTE PONTO É, PORTANTO, OUTRO MÉTODO PARA AS CARTAS
QUE O TEMPO DE QUEM DISSE O PROBLEMA PARA UM INESPERADO

Machine Translated by Google

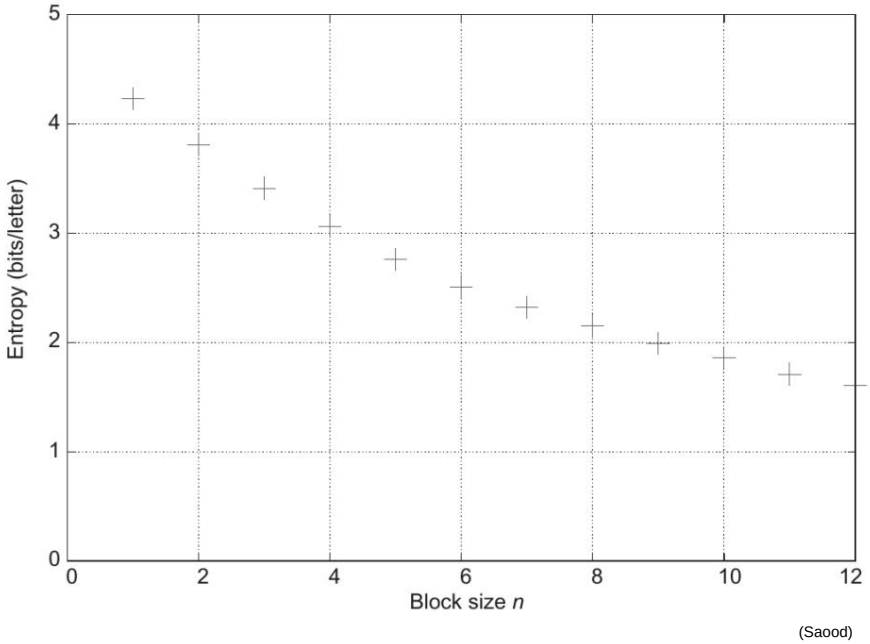


Figura 4.6: Valor estimado da entropia de um texto utilizando diversas ordens.

A Figura 4.6 mostra a evolução da estimativa da entropia de um livro quando a ordem do modelo aumenta.

O número de diferentes contextos (ou estados do processo de Markov) é dado por $|\ddot{y}|^k$, onde k é a ordem do modelo, que claramente cresce exponencialmente com o tamanho do alfabeto. Para por exemplo, em um alfabeto de 2 símbolos, um modelo de contexto finito de profundidade 4 origina apenas $2^4 = 16$ estados ou contextos diferentes. Contudo, se o alfabeto tiver tamanho 256 (por exemplo, se considerarmos uma imagem em nível de cinza), então o número de contextos salta para 2.564 = 4.294.967.296!

Muitas vezes, os modelos de contexto finito são "aprendidos" online, ou seja, à medida que a sequência de símbolos é processada (observe que no Exemplo 4.6 o modelo é estático, ou seja, as probabilidades são consideradas fixas). durante a operação do modelo). No caso da operação online, como os modelos probabilísticos estão em constante adaptação, após o processamento dos primeiros n símbolos de x, o número médio de bits associados a um modelo de contexto finito de ordem k é dado por

$$H_n = -\frac{1}{n}\sum_{i=1}^{n} \log P(x_i \mid x_{i-k}^{i-1}). \tag{4.3}$$

Problema 4.8

Verifique (4.3) e discuta as implicações práticas de x_{n-k}^{n-1} quando n < k (observe que a convenção é que $x_n^n = x_n$) e propor soluções para lidar com isso.

No caso de aprendizagem on-line do modelo de contexto finito, uma tabela (ou alguma outra estrutura de dados mais sofisticada, como uma tabela hash) é usada para coletar contagens que representam o número

Machine Translated by Google

número de vezes que cada símbolo ocorre em cada contexto. Por exemplo, suponha que, em um determinado momento, uma fonte binária (|ÿ| = 2) seja modelada por um modelo de contexto finito de ordem 3 representado por a mesa

xiÿ3	xiÿ2	xiÿ1	n0	n1
0	0	0	10	25
0	0	1	4	12
0	1	0	15	2
0	1	1	3	4
1	0	0	34	78
1	0	1	21	5
		0	17	9
1 1	1 1	1	0	22

onde ni indica quantas vezes o símbolo “i” ocorreu seguindo aquele contexto. Portanto, em neste caso, podemos estimar a probabilidade de um símbolo “0” seguir a string “100” como sendo 34/(34 + 78) ÿ 0,30.

Esta forma de estimar probabilidades, baseada apenas nas frequências relativas de eventos ocorridos anteriormente, sofre do problema de atribuir probabilidade zero a eventos que não foram (ainda) visto. Esse seria o caso, neste exemplo, se utilizássemos a mesma abordagem para estimar a probabilidade de haver um “0” após uma sequência de três ou mais “1s”.

Problema 4.9

Discuta as implicações potenciais do problema da “probabilidade zero” e dê sugestões aos
Evite isso.

Em 1774, Laplace (1749-1827) propôs uma regra para estimar a probabilidade de um determinado evento tem sucesso, após ocorrer n1 sucessos e n2 falhas em n = n1 + n2 tentativas, como

$$\frac{n1 + 1}{n + 2}.$$

Esta fórmula é de fato correta se as tentativas forem independentes e se uma prioridade uniforme sobre a a probabilidade estimada é considerada.

Para o caso mais geral de um beta anterior, B(ÿ1, ÿ2), o estimador adota a forma (GF Hardy, 1889)

$$\frac{n1 + ÿ1}{n + ÿ1 + ÿ2}.$$

A generalização multinomial da fórmula leva a

$$\frac{ni + ÿ}{n + ÿ|ÿ|},$$

onde, por simplicidade, consideramos ÿ = ÿ1 = · · · = ÿ|ÿ| , e os ÿi são os parâmetros do Família de distribuições Dirichlet.

Machine Translated by Google

Um aspecto digno de nota deste estimador é que, definindo

$$\mu = \frac{n}{n + \ddot{y}|\ddot{y}|},$$

pode ser escrito como

$$\text{não}\,\mu \frac{}{n} + (1 \ddot{y} \mu)\, |\ddot{y}| \frac{1}{},$$

mostrando como ele evolui de um estimador uniforme para um estimador de frequência, à medida que n aumenta.

4.4 Um pouco mais sobre a entropia de Shannon

4.4.1 Entropia, entropia conjunta e entropia condicional

Como já mostrado, dada uma distribuição de probabilidade, podemos definir uma quantidade chamada **entropia,** que possui muitas propriedades que concordam com a nossa noção intuitiva do que deveria ser uma medida de informação. Vimos que a entropia expressa a **quantidade de incerteza** sobre uma determinada variável aleatória. Ele mede a **quantidade de informação** necessária **em média** para descrever a variável aleatória.

Esta noção pode ser estendida para definir **informação mútua,** que é uma medida da quantidade de informação que uma variável aleatória contém sobre outra. Na verdade, a informação mútua é um caso especial de uma quantidade mais geral chamada **entropia relativa,** que dá uma medida da "distância" entre duas distribuições de probabilidade.

Considere X uma variável aleatória discreta que assume valores em um alfabeto finito $\ddot{y} = \{\ddot{y}1, \ldots, \ddot{y}m\}$ e tem uma função de massa de probabilidade $p(x) = P(X = x)$, $x \ddot{y} \ddot{y}$.

A entropia de X pode ser interpretada como o **valor esperado** da variável aleatória $\log p(X) \frac{1}{}$,

onde X é desenhado de acordo com p(x). Portanto,

$$H(X) = E \log p(X) \frac{1}{} = E[\ddot{y} \log p(X)],$$

onde E[g(X)] denota o valor esperado de g(X), definido como

$$E[g(X)] = \underset{x\ddot{y}\ddot{y}}{} g(x)p(x).$$

Exemplo 4.13

Seja X uma variável aleatória binária com alfabeto $\ddot{y} = \{0, 1\}$ e função de massa de probabilidade $P(X = 1) = p$ e $P(X = 0) = 1 \ddot{y} p$. Então, a entropia de X é $H(X) = \ddot{y}p \log p \ddot{y} (1 \ddot{y} p) \log(1 \ddot{y} p)$ (ver Fig. 4.7).

Machine Translated by Google

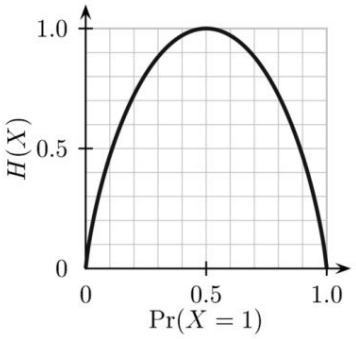


Figura 4.7: Curva de entropia para um alfabeto binário.

Teorema 4.3

0 ÿ H(X) ÿ log |ÿ|, onde |ÿ| denota a cardinalidade de ÿ. Além disso, H(X) = 0 se p(x) = 1 para algum x ÿ ÿ, e H(X) = log |ÿ| se p(x) = 1/|ÿ|.

Prova de H(X) ÿ 0. Como cada p(x) ÿ 1, então cada termo p(x) log p(x) ÿ 0, implicando H(X) ÿ 0. Além disso, H(X) = 0 requer que p(x) log p(x) = 0, ÿx ÿ ÿ, implicando p(x) = 1 ou p(x) = 0.

Portanto, H(X) = 0 se p(x) = 1 para algum x ÿ ÿ (e, obviamente, p(x) = 0 para o outro x).

□

Prova de H(X) ÿ log |ÿ|. Considere a desigualdade ln x ÿ x ÿ 1 e o subconjunto ÿ que p(x) > 0, ÿx ' ÿ ÿ tal ÿ ÿ '. Então

$$H(X) \text{ ÿ } \log |\text{ÿ}' | = \sum_{x \text{ÿÿÿ}} p(x) \log \frac{1}{p(x)} - \sum_{x \text{ÿÿÿ}} \log(x) \text{ ÿ } |\text{ÿ}' | =$$

$$= \frac{1}{\text{Em}} \sum_{2\ x \text{ÿÿÿ}} p(x) \ln \text{ ÿ } |\text{ÿÿ}| p(x) \ln \frac{1}{2\ x \text{ÿÿÿ}} \frac{1}{p(x)} \text{ ÿ } 1 = |\text{ÿÿ}|$$

$$= \frac{1}{\text{Em } 2} \sum_{x \text{ÿÿÿ}} \frac{1}{|\text{ÿ}' |} - \sum_{x \text{ÿÿÿ}} p(x) = 0.$$

O resto da prova fica como exercício. □

Podemos estender a definição de entropia a um par de variáveis aleatórias (X, Y) com distribuição conjunta p(x, y) = P(X = x, Y = y). Para simplificar, assumiremos que ambas as variáveis aleatórias são extraídas do mesmo alfabeto ÿ.

Machine Translated by Google

Definição 4.1 (Entropia Conjunta)
A entropia conjunta H(X, Y) de um par de variáveis aleatórias discretas (X, Y), com uma distribuição conjunta p(x, y), é definida como

$$H(X, Y) = \ddot{y}_{(x,y)\ddot{y}\ddot{y}2} p(x, y) \log p(x, y) = E[\ddot{y} \log p(X, Y)].$$

Observe que se p(x, y) = p(x)p(y) (ou seja, se X e Y são variáveis aleatórias independentes), então H(X, Y) = H(X) + H(Y). Por outro lado, se X e Y são dependentes, então H(X, Y) < H(X) + H(Y).

Definição 4.2 (Entropia Condicional)
Para um par de variáveis aleatórias discretas (X, Y), tendo uma distribuição conjunta p(x, y), definimos entropia condicional H(Y |X) como

$$H(S|X) = \underset{x\ddot{y}\ddot{y}}{} p(x)H(Y|X = x) = - \underset{x\ddot{y}\ddot{y}}{} p(x) \underset{sim\ddot{y}\ddot{y}}{} p(y|x) \log p(y|x) =$$

$$= - \underset{(x,y)\ddot{y}\ddot{y}2}{} p(x, y) \log p(y|x) = E[\ddot{y} \log p(Y|X)].$$

Teorema 4.4 (Regra da cadeia para a entropia de um par de variáveis aleatórias)

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

Prova.

$$H(X, Y) = \ddot{y}\ p(x, y) \log p(x, y) = \ddot{y}\ p(x, y) \log p(x)p(y|x) = \underset{(x,y)\ddot{y}\ddot{y}2}{} \underset{(x,y)\ddot{y}\ddot{y}2}{}$$

$$= - \underset{}{} p(x, y) \log p(x)\ \ddot{y}\ p(x, y) \log p(y|x) = \underset{(x,y)\ddot{y}\ddot{y}2}{} \underset{(x,y)\ddot{y}\ddot{y}2}{}$$

$$= - \underset{x\ddot{y}\ddot{y}}{} p(x) \log p(x) + H(Y|X) = H(X) + H(Y|X).$$

□

Exemplo 4.14

Machine Translated by Google

Considere a seguinte distribuição conjunta de (X, Y):

S\X	1	2	3	4
1	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{32}$
2	$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{32}$	$\frac{1}{32}$
3	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$	$\frac{1}{16}$
4	$\frac{1}{4}$	0	0	0

As distribuições marginais de X e Y são, respectivamente, (H(X) $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}$) e ($\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}$). Por isso, = 1,75 bits e H(Y) = 2 bits. Além disso,

$$H(X|Y) = \sum_{eu=1}^{4} P(Y = i)H(X|Y = i) = \frac{1}{4}H\left(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{8}\right) +$$

$$\frac{1}{4}H\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{8}, \frac{1}{8}\right) + \frac{1}{4}H\left(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}\right) + \frac{1}{4}H(1, 0, 0, 0) = 1{,}375 \text{ bits.}$$

Da mesma forma, H(Y |X) = 1,625 bits e H(X, Y) = 3,375 bits.

Observe que, geralmente, H(Y |X) ÿ= H(X|Y), mas que H(X) ÿ H(X|Y) = H(Y) ÿ H(Y |X).

4.4.2 Entropia relativa

A entropia relativa é uma medida da “distância” entre duas distribuições. Expressa a ineficiência de assumir que a distribuição é q(x) quando a distribuição verdadeira é p(x).

Definição 4.3 (Entropia relativa)
A entropia relativa ou distância de Kullback-Leibler entre duas funções de massa de probabilidade, p(x) e q(x), é definida como

$$D(p\ddot{y}q) = \sum_{x\ddot{y}\ddot{y}} \frac{p(x)}{q(x)} \frac{p(X)}{q(X)} \frac{p(x)}{\text{registro}} = Ep \log \frac{p(X)}{q(X)} .$$

Observe que consideramos as convenções: $0 \log \frac{0}{0} = 0$, $0 \text{ registro } \frac{0}{q} = 0$ e $p \log \frac{p}{0} = \ddot{y}$.

Exemplo 4.15
Seja ÿ = {0, 1} e considere duas distribuições, p(x) e q(x), nos elementos de ÿ. Seja p(0) = 1 ÿ r, p(1) = r, q(0) = 1 ÿ s e q(1) = s. Então

$$D(p\ddot{y}q) = (1 \ddot{y} r) \log \frac{1 \ddot{y} r}{1 \ddot{y} s} + r \log \frac{R}{é}$$

Machine Translated by Google

e

$$D(q\ddot{y}p) = (1 \ddot{y} s) \log \frac{1 \ddot{y} s}{1 \ddot{y} r} + s \log \frac{s}{r}$$

Se r = 1/2 e s = 1/4, então D(pÿq) = 0,208 bits, D(qÿp) = 0,189 bits.

Observe que, em geral, D(pÿq) ÿ= D(qÿp), e, se p(x) = q(x), então D(pÿq) = D(qÿp) = 0 .

4.4.3 Informação mútua

A informação mútua expressa a quantidade de informação que uma variável aleatória contém sobre outro trailer.

Definição 4.4 (Informação mútua)
Considere duas variáveis aleatórias X e Y , com função de massa de probabilidade conjunta p(x, y) e funções marginais p(x) e p(y).

A informação mútua, I(X; Y), pode ser definida como a entropia relativa entre o conjunto distribuição e o produto p(x)p(y), ou seja,

$$Eu(X; Y) = D(p(x, y)\ddot{y}p(x)p(y)) =$$

$$= \sum_{(x,y)\ddot{y}\ddot{y}2} \text{registro } p(x, y) \frac{p(x, y)}{p(x)p(y)} = Ep(x,y) \log \frac{p(X, Y)}{p(X)p(Y)} .$$

Além disso, a informação mútua indica a redução na incerteza de uma variável aleatória devido ao conhecimento do outro. Na verdade, podemos escrever

$$Eu(X; Y) = \sum_{(x,y)\ddot{y}\ddot{y}2} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} = \sum_{(x,y)\ddot{y}\ddot{y}2} p(x, y) \log \frac{p(x|y)}{p(x)} =$$

$$= - \sum_{(x,y)\ddot{y}\ddot{y}2} p(x, y) \log p(x) + \sum_{(x,y)\ddot{y}\ddot{y}2} p(x, y) \log p(x|y) =$$

$$= - \sum_{x\ddot{y}\ddot{y}} p(x) \log p(x) + \sum_{(x,y)\ddot{y}\ddot{y}2} p(x, y) \log p(x|y) =$$

$$= H(X) \ddot{y} H(X|Y) = H(Y) \ddot{y} H(Y |X) = H(X) + H(Y) \ddot{y} H(X, Y).$$

Machine Translated by Google

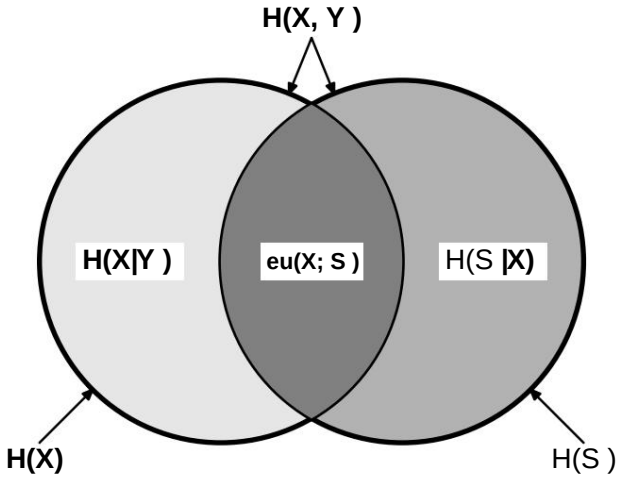


Figura 4.8: Algumas relações entre entropia e informação mútua.

Exemplo 4.16

Considerando novamente a distribuição conjunta de (X, Y),

S\X	1	2	3	4
1	1/8	1/16	1/32	1/32
2	1/16	1/8	1/32	1/32
3	1/16	1/16	1/16	1/16
4	1/4	0	0	0

onde H(X) = 1,75 bits, H(Y) = 2 bits, H(X|Y) = 1,375 bits e H(Y |X) = 1,625, obtemos uma informação mútua entre X e Y de

$$I(X; Y) = H(X) ÿ H(X|Y) = H(Y) ÿ H(Y |X) = 0,375 \text{ bits.}$$

Teorema 4.5 (Relações entre a informação mútua e a entropia)

As seguintes relações entre a informação mútua de um par de variáveis aleatórias, X e Y , e as entropias correspondentes são válidas (ver também Fig. 4.8):

$$eu(X; Y) = H(X) ÿ H(X|Y)$$
$$Eu(X; Y) = H(Y) ÿ H(Y |X)$$
$$Eu(X; Y) = H(X) + H(Y) ÿ H(X, Y)$$
$$Eu(X; Y) = Eu(S; X)$$
$$Eu(X; X) = H(X) ÿ H(X|X) = H(X).$$

Machine Translated by Google

A informação mútua de uma variável aleatória consigo mesma é a entropia da variável aleatória (verifique isso). É por isso que a entropia é às vezes chamada de **autoinformação.**

Teorema 4.6 (Regra da cadeia para a entropia)
deixe X1, X2 ,. . . , Xn seja um conjunto de variáveis aleatórias sorteadas de acordo com p(x1, x2, . . . , xn). Então

$$H(X1, X2, ..., Xn) = \sum_{eu=1}^{n} H(Xi \,|Xiÿ1, . . . , X1).$$

Prova.

$$H(X1, X2) = H(X1) + H(X2|X1),$$

$$H(X1, X2, X3)=H(X1) + H(X2, X3|X1)=H(X1) + H(X2|X1) + H(X3|X2, X1),$$

$$\vdots$$

$$H(X1, X2, . . . , Xn) = H(X1) + H(X2|X1) + · · · + H(Xn|Xnÿ1, . . . , X1).$$

□

Definição 4.5 (Informação mútua condicional)
A informação mútua condicional das variáveis aleatórias X e Y dada Z é definida como

$$I(X; Y \,|Z) = H(X|Z) ÿ H(X|Y, Z) = Ep(x,y,z) \log \frac{p(X, Y \,|Z)}{p(X|Z)p(Y \,|Z)} .$$

(Lembre-se que I(X; Y) = H(X) ÿ H(X|Y))

Teorema 4.7 (Regra da cadeia para a informação)

$$Eu(X1, X2, . . . , Xn; S) = \sum_{eu=1}^{n} eu(Xi \,; Y \,|Xiÿ1, Xiÿ2, . . . , X1).$$

Prova.

$$I(X1, X2, . . . , Xn; Y) = H(X1, X2, . . . , Xn) ÿ H(X1, X2, . . . , Xn|Y) =$$

$$= \sum_{eu=1}^{n} H(Xi \,|Xiÿ1, . . . , X1) - \sum_{eu=1}^{n} H(Xi \,|Xiÿ1, . . . , X1, Y) =$$

$$= \sum_{eu=1}^{n} I(Xi \,; Y \,|Xiÿ1, Xiÿ2, . . . , X1).$$

□

Machine Translated by Google

Teorema 4.8 (desigualdade de Jensen)
Se f(x) é uma função convexa e X é uma variável aleatória,

$$E[f(X)] \ddot{y} f(E[X]).$$

Além disso, se f(x) for estritamente convexa, a igualdade implica que X = E[X] com probabilidade 1, ou seja, X é constante.

Teorema 4.9 (Desigualdade de informação)
Sejam p(x) e q(x) duas funções de massa de probabilidade, com x ÿ ÿ. Então, com igualdade se p(x) = q(x), ÿx,

$$D(p\ddot{y}q) \ddot{y} 0.$$

Prova da não negatividade. Deixe ÿ' = {x|p(x) > 0} seja o conjunto suporte de p(x). Então q(x)

$$\ddot{y}D(p\ddot{y}q) = \sum_{x\ddot{y}\ddot{y}\ddot{y}} \log \ddot{y} \text{ (pela desigualdade de Jensen) } p(x)$$

$$\ddot{y} \text{ registro} \sum_{x\ddot{y}\ddot{y}\ddot{y}} p(x) = \frac{q(x)}{\log p(x)} \sum_{x\ddot{y}\ddot{y}\ddot{y}} q(x) \ddot{y} \log \sum_{x\ddot{y}\ddot{y}} q(x) = \log 1 = 0,$$

ou seja, D(pÿq) > 0. □

Prova de que D(pÿq) = 0 se p = q. Como log x é uma função estritamente côncava de x, temos q(x)

$$\sum_{x\ddot{y}\ddot{y}\ddot{y}} \log = \log \frac{p(x)}{p(x)} \sum_{x\ddot{y}\ddot{y}\ddot{y}} p(x) \frac{q(x)}{p(x)}$$

se q(x)/p(x) é constante em todos os lugares (ou seja, se q(x) = cp(x), ÿx). Portanto,

$$\sum_{x\ddot{y}\ddot{y}\ddot{y}} q(x) = c \sum_{x\ddot{y}\ddot{y}\ddot{y}} p(x) = c.$$

Além disso, é necessário que $\sum_{x\ddot{y}\ddot{y}\ddot{y}} q(x) = 1$ por ter

$$\text{registro} \sum_{x\ddot{y}\ddot{y}\ddot{y}} q(x) = \log \sum_{x\ddot{y}\ddot{y}} q(x),$$

implicando c = 1. Portanto, temos D(pÿq) = 0 se p(x) = q(x), ÿx. □

Corolário 4.9.1 (Não negatividade da informação mútua)
Para quaisquer duas variáveis aleatórias, X e Y,

$$eu(X; Y) \ddot{y} 0,$$

com igualdade se X e Y forem independentes.

Machine Translated by Google

Prova.

$$eu(X; Y) = D(p(x, y)ÿp(x)p(y)) ÿ 0,$$

com igualdade se p(x, y) = p(x)p(y), ou seja, quando X e Y são independentes. □

Teorema 4.10 (Limite superior da entropia)

$$H(X) ÿ \log |X |,$$

com igualdade se X tem distribuição uniforme sobre X ·

Prova. Seja você(x) = 1/|ÿ| seja a função de massa de probabilidade uniforme sobre ÿ, e seja p (x) o função de massa de probabilidade para X. Então,

$$D(pÿu) = \sum_{xÿÿ} p(x) \log \frac{p(x)}{você(x)} = \log |ÿ| ÿ H(X) ÿ 0.$$

Portanto, H(X) ÿ log |ÿ|, com igualdade se p(x) = u(x). □

Teorema 4.11 (O condicionamento reduz a entropia)

$$H(X|Y) ÿ H(X)$$

com igualdade se X e Y forem independentes.

Prova.

$$0 ÿ I(X; Y) = H(X) ÿ H(X|Y).$$

□

Este teorema afirma que conhecendo outras variáveis aleatórias, Y só pode reduzir a incerteza em X. No entanto, isso só é verdade **em média.** Especificamente, H(X|Y = y) pode ser maior que ou menor ou igual a H(X), mas, em média, H(X|Y) = p(y)H(X|Y = y) ÿ H(X).

sim

Exemplo 4.17

Deixe (X, Y) ter a seguinte distribuição conjunta,

S\X	1	2
1	0	$\frac{3}{4}$
2	$\frac{1}{8}$	$\frac{1}{8}$

Machine Translated by Google

Então, $H(X) = H(\frac{1}{8}, \frac{7}{8})$ ÿ 0,544 bits, $H(X|Y = 1) = 0$ e $H(X|Y = 2) = 1$ bit. No entanto,

$$H(X|Y) = \frac{3}{4} H(X|Y = 1) + \frac{1}{4} H(X|Y = 2) = 0{,}25 \text{ bits.}$$

Assim, a incerteza em X aumenta se Y = 2 for observado e diminui se Y = 1. No entanto, em média, a incerteza diminui.

Teorema 4.12 (Independência ligada à entropia)
Seja $X_1, X_2, \ldots, X_n$ seja desenhado de acordo com $p(x_1, x_2, \ldots, x_n)$. Então,

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

com igualdade se as variáveis aleatórias X_i forem independentes.

Prova. Pela regra da cadeia para entropias e usando o fato de que o condicionamento reduz a entropia,

$$H(X_1, X_2, \ldots, X_n) = \sum_{eu=1}^{n} H(X_i | X_i \text{ÿ} 1, \ldots, X_1) \text{ ÿ } \sum_{eu=1}^{n} H(X_i),$$

com igualdade se os X_i forem independentes. □

Machine Translated by Google

Machine Translated by Google

5 Compressão de dados

O principal objetivo de um algoritmo de compressão de dados é reduzir o tamanho dos dados necessários para representar uma certa quantidade de informação. Neste capítulo abordamos algumas das abordagens mais importantes de compressão de dados, caracterizadas por permitir a recuperação total dos dados originais, ou seja, sem perder nenhuma informação. Eles são conhecidos como algoritmos de compressão de dados sem perdas.

5.1 Codificação de comprimento variável

5.1.1 Códigos ótimos

Para uma dada distribuição fonte pi , queremos encontrar o código livre de prefixo com o comprimento mínimo esperado, ou seja, queremos minimizar

$$L = pi\text{-}li \qquad ,$$

sobre todos os inteiros l1, l2, . . . , estou satisfazendo a desigualdade de Kraft, ou seja,

$$2 \ddot{y} li \ddot{y} 1.$$

Ao relaxar a restrição inteira em li , obtemos os comprimentos de código ideais, l ÿ i , usando um método de otimização15, que dá

$$li^{\ddot{y}} = \ddot{y} \log pi.$$

Esta escolha não inteira de comprimentos de palavras-código produz um comprimento médio de palavras-código

$$eu^{\ddot{y}} = pi\ eu \qquad {}^{\ddot{y}}_{eu} = \ddot{y}\ pi \log pi = H.$$

Teorema 5.1 O
comprimento médio L de qualquer código binário instantâneo para uma distribuição fonte pi é maior ou igual à entropia H, ou seja,

$$eu \ddot{y} H,$$

com igualdade se 2 ÿli = pi .

Prova.

$$H \ddot{y} L = \ddot{y}\ pi \log pi \ddot{y}\ pi\ li = pi \log 2\ddot{y} li \ddot{y}\ pi \log pi =$$

$$= pi \log \ddot{y} \log \frac{2 \ddot{y} li}{pi} \qquad pi \frac{2 \ddot{y} li}{pi} = 0,$$

onde a desigualdade resulta do uso da desigualdade de Jensen. □

15Considere, por exemplo, que li = ÿ log qi . Usando multiplicadores de Lagrange para resolver esta otimização restrita problema, temos que encontrar o gradiente de ÿ pi log qi + ÿ(qi ÿ 1).

Machine Translated by Google

Conforme mostrado, a escolha do comprimento das palavras li = ÿ log pi resulta em L = H. No entanto, na prática, esses valores podem não ser inteiros. Portanto, podemos querer construir um código arredondando os comprimentos ÿ log pi, ou seja,

li = ÿÿ log piÿ.

Observe que esses comprimentos satisfazem a desigualdade de Kraft, porque

2 ÿÿÿ log piÿ ÿ 2 ÿ(ÿ log pi) = pi = 1.

Além disso, podemos escrever

ÿ log pi ÿ li < ÿ log pi + 1 ÿ H ÿ L < H + 1.

Observe que o código ideal só pode ser melhor que este código.

Teorema 5.2

Seja lÿ 1, lÿ 2, . . . , lÿn sejam os comprimentos ideais de palavras-código para uma distribuição de origem pi e seja L comprimento esperado associado de um código ideal, ou seja, L ÿ ÿ o = pi l ÿi . Então

H ÿ L ÿ < H + 1.

Prova. Vimos que se li = ÿÿ log piÿ então

H ÿ L = pi li < H + 1.

Como L ÿ é menor que L e como L ÿ ÿ H, temos o teorema. □

Neste último teorema temos um overhead de no máximo um bit, devido ao fato de ÿ log pi nem sempre ser um número inteiro. Podemos reduzir a sobrecarga por símbolo espalhando-a por vários símbolos. Portanto, consideremos um sistema no qual enviamos sequências de n símbolos de uma só vez.

Considere que Ln é o comprimento esperado da palavra-código por símbolo de entrada, ou seja, se l(x1x2 · · · xn) é o comprimento da palavra-código associada à string x1x2 · · · xn, onde xi ÿ ÿ, então

$$L_n = \frac{1}{n} \quad p(x_1x_2 \cdots x_n)l(x_1x_2 \cdots x_n) = \frac{1}{n} E[l(X_1, X_2, \ldots, X_n)].$$

Teorema 5.3

O comprimento mínimo esperado da palavra-código por símbolo satisfaz

$$\frac{H(X_1, X_2,..., X_n) \ ÿ \ L}{n} \quad ÿ_n < \frac{H(X_1, X_2, \ldots, X_n)}{n} 1 + \frac{}{n}.$$

Além disso, se X1, X2, . . . , Xn é um processo estocástico estacionário,

$$L_n^{ÿ} \ ÿ \ H(X),$$

onde H(X) é a taxa de entropia do processo.

Particularmente, se o processo for iid, então

$$H ÿ L_n^{ÿ} < H + \frac{1}{n}.$$

Machine Translated by Google

5.1.2 Um argumento de contagem

Consideremos que existe um algoritmo, C, que é capaz de comprimir todas as mensagens de n bits. Por hipótese, C é capaz de reduzir o tamanho de cada um (ou mais). Existem 2 n das 2 n mensagens com menos de n bits.

Quantas palavras-código podem ser formadas com menos de n bits? Existem 2 2 = 1 bit, . . . , 2^{n-1} com n ÿ 1 bits, 2^{n-2} com n ÿ 2 bits, . . . , 2 ÿ n = 2n ÿ 1 bits.

Portanto, o número total de palavras-código com menos de n bits é

$$\sum_{i=0}^{n-1} 2^{i} = \frac{1 - 2^{n}}{1-2} = 2^{n} - 1.$$

Assim, pelo menos duas das mensagens devem partilhar a mesma palavra-código, o que significa que é impossível ter uma atribuição de palavra-código um-para-um. Isto implica que é impossível ter um algoritmo de compressão (reversível) capaz de comprimir todas as mensagens. Para alguns deles, não irá comprimir ou mesmo expandir seu comprimento.

5.1.3 Códigos de Huffman

Os códigos Huffman são códigos de comprimento variável desenvolvidos por David A. Huffman (1925–1999) (Huff man, 1952). Estes códigos são óptimos, no sentido de que minimizam o comprimento médio das mensagens codificadas, entre todos os códigos de comprimento variável possíveis.

São códigos sem prefixo, ou seja, nenhuma das palavras-código possui um prefixo composto por uma palavra-código mais curta. Lembre-se que, por exemplo, se uma determinada palavra-código for representada pela cadeia binária “100”, então a cadeia binária “10001” não pode pertencer a esse código. Conforme mostrado anteriormente, esta propriedade garante uma decodificação única e instantânea.

O procedimento de Huffman para construir os códigos é baseado em duas observações sobre códigos de prefixo ideais:

- Num código óptimo, os símbolos que ocorrem com mais frequência devem ter palavras-código mais curtas do que os símbolos que ocorrem com menos frequência.
- Em um código ideal, os dois símbolos que ocorrem com menos frequência devem ter o mesmo comprimento.

O procedimento de Huffman contém a restrição adicional de que:

- As palavras-código dos dois símbolos de menor probabilidade diferem apenas no último bit.

Portanto, para a construção dos códigos de Huffman, os símbolos (nós das árvores) com as menores probabilidades são combinados primeiro. O novo nó obtém a soma das probabilidades dos dois nós combinados. Este procedimento é repetido até que se tenha um único nó com probabilidade igual a um. Por fim, as palavras-código são obtidas associando 0's e 1's aos ramos do árvore.

Machine Translated by Google

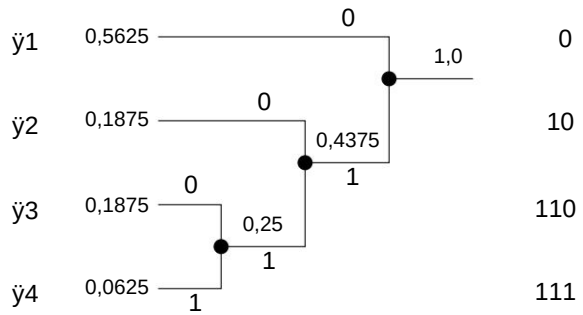


Figura 5.1: Exemplo de árvore associada a um código Huffman de 4 símbolos.

Exemplo 5.1

No exemplo da Fig. 5.1, o **número médio de bits/símbolo** é

$$1 \times 0{,}5625 + 2 \times 0{,}1875 + 3 \times 0{,}1875 + 3 \times 0{,}0625 \text{ ÿ } 1{,}69$$

A entropia desta fonte de informação é de 1,62 bits. Portanto, a **redundância** deste código é ÿ 0,07 bits/símbolo.

O processo de decodificação é direto. Por exemplo, suponha que o decodificador recebeu a string binária:

0100111101100

Então, a sequência de símbolos é

0	10	0	111 10	110 0
ÿ1	ÿ2 ÿ1	ÿ4 ÿ2	ÿ3 ÿ1	

Quando existem vários nós disponíveis para combinação (ou seja, com probabilidades iguais), é possível projetar códigos diferentes. Neste caso, qual devemos escolher? Por exemplo, considere a seguinte tabela de probabilidades, associada a um alfabeto de 5 símbolos,

	ÿ1 ÿ2 ÿ3 ÿ4 ÿ5 pi 0,4 0,2
0,2	0,1 0,1

A entropia de primeira ordem desta fonte de informação é de 2,12 bits, e a Fig. 5.2 mostra dois códigos possíveis para esta distribuição de probabilidade. Como pode ser verificado, ambos os códigos possuem o mesmo número médio de bits/símbolo:

$$2 \times 0{,}4 + 2 \times 0{,}2 + 2 \times 0{,}2 + 3 \times 0{,}1 + 3 \times 0{,}1 = 2{,}2$$

$$1 \times 0{,}4 + 3 \times 0{,}2 + 3 \times 0{,}2 + 3 \times 0{,}1 + 3 \times 0{,}1 = 2{,}2$$

Machine Translated by Google

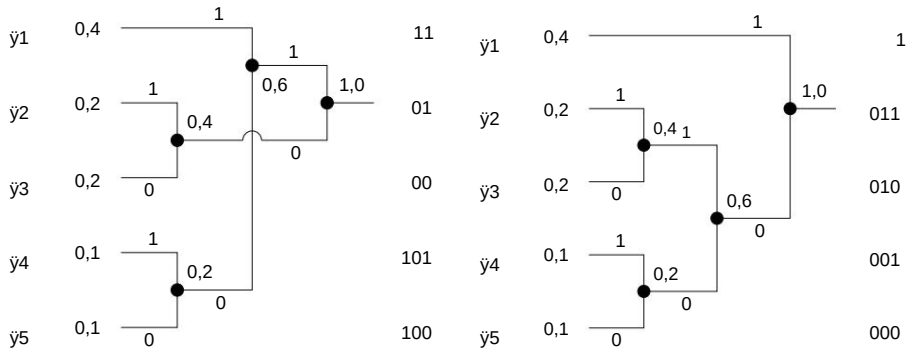


Figura 5.2: Dois códigos de Huffman alternativos para a mesma distribuição de probabilidade.

Após um tempo de transmissão suficientemente longo, ambos os códigos são equivalentes. No entanto, aqueles que têm variações mais altas podem ser menos convenientes, porque podem exigir taxas de bits instantaneamente mais altas e, portanto, buffers maiores. O primeiro código tem variação

$$0{,}4(2 \ddot{y} 2{,}2)2 + 2 \times 0{,}2(2 \ddot{y} 2{,}2)2 + 2 \times 0{,}1(3 \ddot{y} 2{,}2)2 = 0{,}16,$$

enquanto o segundo tem variância de 0,96.

Programação 5.1 Escreva um programa que simule o processo de codificação para uma dada distribuição de probabilidade e o código de Huffman correspondente (por exemplo, aqueles mostrados na Figura 5.2). Utilizando esse programa, analise as estatísticas do fluxo de bits produzido pelo codificador com o objetivo de descobrir se ele ainda é compressível.

Programação 5.2 Usando simulação, estime o tempo médio para overflow/underflow do buffer, em função do tamanho do buffer. Como possível exemplo, verifique que o código mais à esquerda da Fig. 5.2 é menos exigente que o código mostrado à direita, no que diz respeito ao tamanho do buffer.

5.1.4 Códigos Shannon-Fano

A codificação Shannon-Fano foi proposta antes da codificação Huffman e, embora não seja ideal como a codificação Huffman, ainda assim fornece bons códigos de tamanho variável. Os códigos de Shannon-Fano são como os códigos de Huffman, no sentido de que também são códigos de comprimento variável e imediatamente decodificáveis.

Para construir estes códigos, os símbolos são primeiro ordenados por probabilidade crescente ou decrescente. Em seguida, os símbolos são divididos em dois conjuntos com capacidades de probabilidade totais tão semelhantes quanto possível. Um dos conjuntos recebe o rótulo “0”, enquanto o outro recebe o rótulo “1”. Isto se repete até obter conjuntos com apenas um ou dois símbolos.

Machine Translated by Google

ÿ6 0,25	1	**1**		
ÿ 0,20	1	**0**		
ÿ 0,15 4	0	1	**1**	
ÿ 0,15 5	0	1	**0**	
ÿ1 0,10	0	0	1	
ÿ7 0,10	0	0	0	**1**
ÿ2 0,05	0	0	0	**0**

Figura 5.3: Exemplo de construção de um código Shannon-Fano.

Exemplo 5.2
Considere a seguinte tabela de probabilidade:

ÿ1	ÿ2	ÿ3	ÿ4	ÿ5	ÿ6	ÿ7
0,10	0,05	0,20	0,15	0,15	0,25	0,10

O código Shannon-Fano para esta distribuição de probabilidade é mostrado na Figura 5.3.

Observe que enquanto a árvore Shannon-Fano é criada da raiz às folhas, o algoritmo de Huffman funciona das folhas à raiz.

5.1.5 Extensão do alfabeto

Quando

$$p_i \ddot{y} 2^{\ddot{y}k}, \quad k \ddot{y} N, \ddot{y}i,$$

a eficiência dos códigos de comprimento variável aproxima-se da entropia da fonte de informação. Particularmente, quando as probabilidades são potências de 1/2, o comprimento médio dos códigos é igual à entropia da fonte. Nesse caso,

$$\ddot{y} \log p_i \ddot{y} N, \ddot{y}i,$$

isto é, os comprimentos ótimos das palavras-código são inteiros. No entanto, para alfabetos pequenos e para distribuições de probabilidade altamente distorcidas, os códigos de comprimento variável correspondentes podem estar longe de ser ideais!

Exemplo 5.3
Consideremos dois símbolos, ÿ1 e ÿ2. Variando os valores de p1 e p2, obtemos os seguintes valores de entropia:

p1	0,9	0,8	0,7	0,6	0,5	p2	0,1	0,2	0,3
0,4	0,5	H	0,47	0,72	0,88	0,97	1,00		

Machine Translated by Google

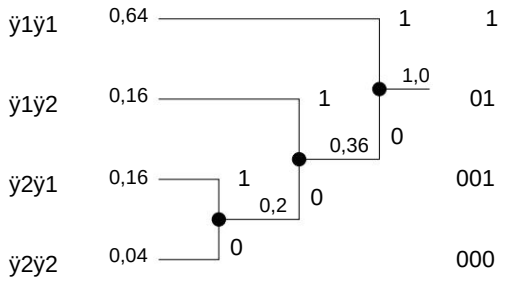


Figura 5.4: Código de Huffman resultante da agregação de pares de símbolos.

Como esperado, quanto maior a diferença entre estas duas probabilidades, menos bits, em média, são necessários em teoria para representar uma mensagem.

Considere, por exemplo, o caso em que p1 = 0,8 e p2 = 0,2:

- Usando codificação Huffman ou codificação Shannon-Fano, obtemos um comprimento médio de código de um bit por símbolo.
- Isto é 28% mais longo do que o teoricamente necessário.

Consideremos agora agrupamentos de pares de símbolos e assumimos independência. As probabilidades das quatro combinações são

Par de
probabilidade ÿ1ÿ1 0,8 × 0,8
= 0,64 ÿ1ÿ2 0,8 × 0,2 = 0,16
ÿ2ÿ1 0,2 × 0,8 = 0,16 ÿ2ÿ2
0,2 × 0,2 = 0,04

Construindo um código de Huffman para esses novos símbolos (ver Fig. 5.4), obtemos um comprimento médio de

$$1 \times 0{,}64 + 2 \times 0{,}16 + 3 \times 0{,}16 + 3 \times 0{,}04 = 1{,}56 \text{ bits},$$

isto é, um comprimento médio de 0,78 bits/símbolo original.

Considere agora agrupamentos de três símbolos, com probabilidades

Probabilidade de grupo ÿ1ÿ1ÿ1
0,8 × 0,8 × 0,8 = 0,512 ÿ1ÿ1ÿ2 0,8 × 0,8
× 0,2 = 0,128 ÿ1ÿ2ÿ1 0,8 × 0,2 × 0,8 =
0,128 ÿ1ÿ2ÿ2 0,8 × 0,2 × 0,2 = 0,032
ÿ2ÿ1ÿ 1 0,2 × 0,8 × 0,8 = 0,128 ÿ2ÿ1ÿ2
0,2 × 0,8 × 0,2 = 0,032 ÿ2ÿ2ÿ1 0,2 × 0,2
× 0,8 = 0,032 ÿ2ÿ2ÿ2 0,2 × 0,2 × 0,2 =
0,008

Machine Translated by Google

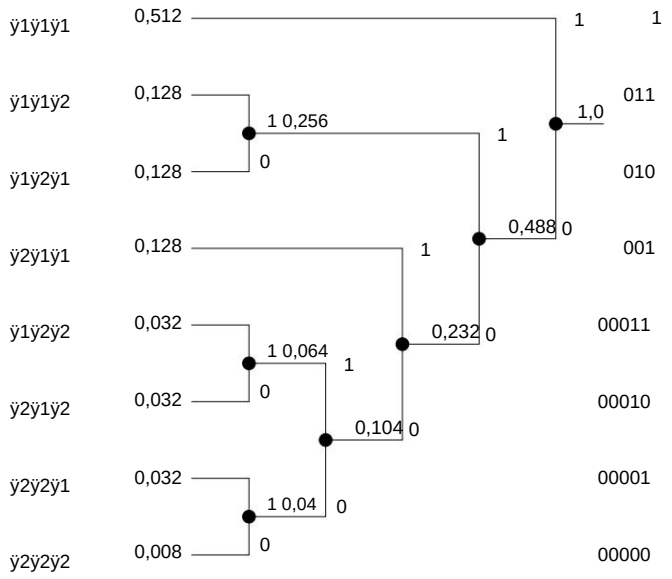


Figura 5.5: Código de Huffman para grupos de três símbolos.

O comprimento médio do código de Huffman correspondente é de 2,184 bits, ou seja, 0,728 bits/símbolo original (ver Fig. 5.5).

5.1.6 Alguns outros códigos de comprimento variável

Às vezes, é necessário projetar códigos para representar, de forma eficiente, números inteiros cujas probabilidades diminuem com seus valores. Neste caso é possível construir códigos mais simples que os códigos de Huffman, embora por vezes com alguma perda de eficiência. Por outro lado, estes são códigos “abertos”, ou seja, são capazes de acomodar números raros, mas possíveis (grandes).

O código de vírgula (também conhecido como código unário) é o código de comprimento variável mais simples. Cada palavra-código é construída usando uma sequência de 0's (1's), terminada por um 1 (0):

0 ÿ 0 1 ÿ 10 2
ÿ 110

3 ÿ 1110 4 ÿ 11110

. . .

Observe que o decodificador é apenas um contador.

Machine Translated by Google

Problema 5.1

Mostre que o código unário é ótimo se os inteiros, i, ocorrerem com probabilidades

$$pi = \frac{1}{2\ (eu+1)}, \qquad eu = 0, 1, 2, \ldots$$

Os códigos de deslocamento são generalizações do código de vírgula.

B = 1	B = 2	B = 3
0	00	000
10	01	001
110	10	010
1110	1100	011
11110	1101	100
111110	1110	101
1111110	111100	110
11111110	111101	111000
. . .	. . .	. . .

- Esses códigos são organizados por níveis, com base em um conjunto de 2^B palavras-código de comprimento B.
- Uma dessas palavras-código é usada para indicar a mudança para o próximo nível.
- Para B = 1 temos o código da vírgula.

Outro tipo de código de turno utiliza, além da base, B, um incremento, I. O primeiro nível é formado por todas as palavras-código de comprimento B, exceto a palavra-código de deslocamento. O próximo nível (nível 2) é formado por todas as palavras-código de bits B + I, exceto a palavra-código shift, concatenada com o (prefixo) mudar a palavra-código do nível anterior.

Generalizando, o nível L consiste em todas as palavras-código de bits B + I (L ÿ 1), exceto para a sequência de deslocamento, concatenado com o prefixo (palavra-código shift) do nível L ÿ 1.

Machine Translated by Google

Exemplo 5.4

B = 1, I = 1	B = 2, I = 1	B = 1, I = 2
L Palavra-código L	Palavra-código L	
1 0 1 0	100	
2 100	1 01	2 1000
2 101	1 10	2 1001
2 110	2 11.000	2 1010
3 111000	2 11001	2 1011
3 111001	2 11010	2 1100
3 111010	2 11011	2 1101
3 111011	2 11100	2 1110
3 111100	2 11101	3 111100000
3 111101	2 11110	3 111100001
3 111110	3 111110000	3 111100010
...	...	...

5.1.7 Código Golomb

O código Golomb foi proposto por Solomon W. Golomb (1932–2016) (Golomb, 1966). Isto depende da separação de um número inteiro em duas partes:

- Uma dessas partes é representada por um código unário.
- A outra parte é representada através de um código binário.

Na verdade, o código Golomb é uma família de códigos que depende de um parâmetro inteiro, m > 0. O O código Golomb é ideal para uma fonte de informação seguindo uma distribuição

$$pi = ÿ \quad ^{eu} (1 ÿ ÿ), \; i = 0, 1, 2, . . .$$

onde

$$m = - \frac{1}{logÿ} .$$

Um inteiro i ÿ 0 é representado por dois números, q e r, onde

$$q = \frac{^{eu}}{eu} \qquad e \qquad r = eu ÿ qm.$$

O quociente, q, pode ter os valores 0, 1, 2,. . . , e é representado pelo unário correspondente código. O restante da divisão, r, pode ter os valores 0, 1, 2, . . . , mÿ1, e é representado pelo código binário correspondente.

Machine Translated by Google

Exemplo 5.5
Considere i = 11 e m = 4. Então,

$$q = \frac{11}{4} = 2, \; r = 11 \text{ ÿ } 2 \times 4 = 3 \text{ ÿ } 001\ 11$$

Se m não for uma potência de 2, então o código binário (que representa o resto da divisão), não é eficiente. Nesse caso, o número de bits usados pode ser reduzido usando um **binário truncado código:**

- Primeiro, defina b = ÿlog mÿ.
- Codifique os 2 primeiros^b ÿ m valores de r usando os 2 primeiros^b ÿ m palavras-código binárias de b ÿ 1 bits.
- Codifique os valores restantes de r codificando o número r + 2b ÿ m em palavras-código binárias de b bits.

Exemplo 5.6
Considere m = 5. Neste caso, temos b = ÿlog 5ÿ = 3. Portanto, os valores r = 0, 1, 2 são representado usando 2 bits (respectivamente “00”, “01” e “10”), enquanto os valores r = 3, 4 serão ser codificado usando 3 bits (respectivamente “110” e “111”).

Exemplo 5.7
Código Golomb para m = 5 e i = 0, 1,. . . , 15:

Palavra-código iqr	Palavra-código iqr
0 0 0 000	8 1 3 10110
1 0 1 001 2 0	9 1 4 10111
2 010	10 2 0 11.000
3 0 3 0110 4 0	11 2 1 11001
4 0111 5 1 0	12 2 2 11010
1000 6 1 1 1001	13 2 3 110110
7 1 2 1010	14 2 4 110111
	15 3 0 111000

5.2 Compressão baseada em dicionário

Em muitos casos, a fonte de informação produz padrões recorrentes. Uma abordagem possível a ser tomada A vantagem deste comportamento é manter um **dicionário** com os padrões (mais frequentes). Quando

Machine Translated by Google

um desses padrões ocorre, ele é codificado usando uma referência ao dicionário. Se não aparecer no dicionário, pode ser codificado usando algum outro método (geralmente menos eficiente).
Portanto, a ideia é dividir os padrões em duas classes: frequentes e pouco frequentes.

Exemplo 5.8
Suponha que temos um alfabeto fonte com 32 símbolos (por exemplo, 26 letras mais alguns sinais de pontuação). Para uma fonte iid, precisaríamos de 5 bits/símbolo. Tratar todos os 324 (1 048 576) padrões de 4 símbolos como igualmente prováveis implicaria 20 bits para cada padrão de 4 símbolos.

Agora, suponha que coloquemos os 256 padrões mais comuns em um dicionário. Se o padrão estiver no dicionário, envie '0' seguido do índice (8 bits) do padrão no dicionário. Caso contrário, envie '1' seguido dos 20 bits que codificam o padrão.

Observe que, se p for a probabilidade de encontrar o padrão no dicionário, então o comprimento médio do código é

$$L = 9p + 21(1 \text{ ÿ } p) = 21 \text{ ÿ } 12p.$$

Para ser útil, este esquema deve atingir $L < 20$. Isto acontece para p ÿ 0,084. No entanto, para uma fonte iid, a probabilidade de um padrão de 4 símbolos sobre um alfabeto de 32 símbolos é de apenas 0,00000095. . .

5.2.1 Códigos de instalação

Como vimos, os códigos de comprimento variável atribuem sequências de bits de comprimento variável aos símbolos do alfabeto. Um dos problemas desta abordagem é que os erros nos dados codificados se propagam facilmente.

No **código Tunstall,** todas as palavras-código têm o mesmo comprimento. Em vez disso, cada palavra-código representa grupos de símbolos alfabéticos de comprimento variável.

Exemplo 5.9
Considere o seguinte código Tunstall de 2 bits para o alfabeto ÿ = {A, B}:

Palavra-código de sequência	
AAA 00	
AAB	01
AB	10
B	11

Então, a sequência “AAABAABAABAABAAA” é codificada como “001101010100”.

Machine Translated by Google

O projeto de um código que possui um comprimento fixo de palavra-código, mas um número variável de símbolos por palavra-código, precisa atender às duas condições a seguir:

- Devemos ser capazes de analisar todas as strings de origem em substrings de símbolos que aparecem no livro de códigos (dicionário). Esta propriedade é geralmente conhecida como **"generalidade".**
- Devemos maximizar o número médio de símbolos fonte representados por cada código palavra.

Exemplo 5.10

Para entender o significado da primeira condição, considere agora o seguinte código:

Palavra-código de sequência	
AAA	00
ABA	01
AB	10
B	11

Vamos tentar codificar a mesma sequência, "AAABAABAABAABAAA", mas agora usando esta nova código:

- Primeiro, codificamos "AAA" com o código "00".
- Em seguida, codificamos "B" com o código "11".
- Em seguida, teremos problemas, porque não há como prosseguir. . .

Lema 5.1 (Generalidade do dicionário)

Um dicionário que consiste em strings de um alfabeto ÿ satisfaz a condição de generalidade se e se for para cada sequência possível de símbolos x de ÿ existe pelo menos um prefixo de x no dicionário.

Tópico de discussão 5.1

Discuta as condições que um dicionário deve atender para obedecer à restrição de generalidade.

Um código Tunstall de n bits para uma fonte iid em um alfabeto de tamanho m pode ser construído usando o seguinte algoritmo:

Machine Translated by Google

Algoritmo 5.1 (código Tunstall)

- Comece com os m símbolos do alfabeto no livro de códigos.
- Em seguida, remova a entrada com maior probabilidade e adicione as m strings obtidas por concatenando este símbolo com todos os símbolos do alfabeto. As novas probabilidades são o produto (porque a independência é assumida) das probabilidades dos símbolos concatenados - observe que esta operação aumenta o tamanho do livro de códigos de m para m + (m ÿ 1).
- Repita este procedimento k vezes, sujeito à restrição m + k(m ÿ 1) ÿ 2 n , onde k está o maior número inteiro possível que verifica a condição.

Exemplo 5.11

Considere o projeto de um código Tunstall de 3 bits para o alfabeto

Probabilidade de símbolo	
Um 0,60	
B	0h30
C	0,10

Como “A” é o símbolo mais provável, primeiro obtemos

Probabilidade de sequência	
B 0,30	
C 0,10	
AA	0,36
AB	0,18
AC	0,06

Finalmente, temos

Código de probabilidade de sequência		
B 0,300 000		
C	0,100 001	
AB	0,180 010	
AC	0,060 011	
AAA	0,216	100
AAB	0,108	101
AAC	0,036	110

Machine Translated by Google

Tópico de discussão 5.2

Os códigos Tunstall são exclusivamente decodificáveis? Por que?

5.2.2 Análise de strings

O objetivo final da compactação de dados baseada em dicionário é analisar a string de origem em substrings (também conhecidas como frases ou fatores) cujas palavras-código correspondentes tenham o menor comprimento total possível. A análise, ou seja, a criação de uma partição da string, desempenha um papel importante neste processo.

Para um dicionário arbitrário, obedecendo à restrição de generalidade, existem maneiras ótimas (no sentido de produzir o menor número de frases) de analisar a string a ser compactada.
Normalmente, eles envolvem mais de uma passagem pela sequência de dados, usando programação dinâmica. Abordagens mais diretas, embora possivelmente subótimas, baseiam-se em estratégias gananciosas. Neste caso, assumindo que o processamento é feito da esquerda para a direita, o algoritmo ganancioso escolhe o prefixo mais longo possível da string ainda não compactada que existe no dicionário.

Para alguns casos especiais, a estratégia gananciosa pode ser, de facto, óptima. Isso acontece quando os dicionários são completos com prefixo ou completos com sufixo. Um dicionário é **prefixado** se, para cada string do dicionário, todos os seus prefixos também estiverem no dicionário. Um dicionário é **sufixo completo** se, para cada string do dicionário, todos os seus sufixos também estiverem no dicionário.

Teorema 5.4 (Cohn e Khazan 1996)

Com relação a um dicionário com sufixo completo, a análise gananciosa da esquerda para a direita é ideal; duplamente, com relação a um dicionário com prefixo completo, a análise gananciosa da direita para a esquerda é ideal.

5.2.3 Compressão LZ77

Na década de 1970, Jacob Ziv (1931–) e Abraham Lempel (1936–) propuseram duas abordagens diferentes para compressão de dados, usando princípios baseados em dicionário. Ao contrário do código Tunstall que acabamos de ver, onde o dicionário é construído previamente (ou seja, é **estático),** esses métodos são **adaptativos – o** dicionário é "aprendido" ao mesmo tempo que a compactação é executada.

O primeiro algoritmo, normalmente conhecido como LZ77 (ou LZ1), foi apresentado em

- J. Ziv e A. Lempel, Um algoritmo universal para compressão sequencial de dados, IEEE Transações na Teoria da Informação (Ziv e Lempel, 1977).

Essencialmente, baseia-se na separação dos dados em duas partes:

Machine Translated by Google

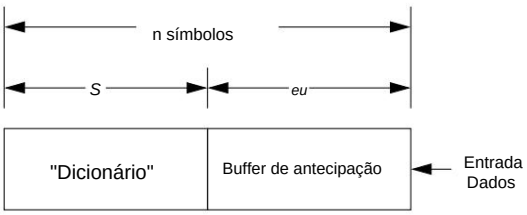


Figura 5.6: Separação entre o “dicionário” e os dados a serem codificados em LZ77.

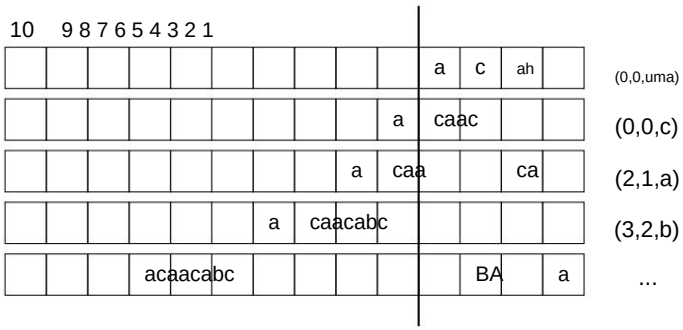


Figura 5.7: Exemplo simples de codificação LZ77.

- Dados já codificados;
- Dados que devem ser codificados,

conforme mostrado na Figura 5.6.

Durante a operação, os dados passam primeiro por um **buffer de antecipação** e depois por um **buffer de pesquisa** (o “dicionário”). O algoritmo procura, no “dicionário”, o maior prefixo possível da string no buffer look-ahead. Obviamente, quanto maior for o prefixo, maior será a eficiência da codificação.

A palavra-código gerada em cada etapa de codificação é composta por **três componentes:**

1. Um ponteiro que indica a posição da string correspondente no “dicionário”.
2. O comprimento da string correspondente.
3. O primeiro símbolo no buffer de antecipação segue a string correspondente.

A Figura 5.7 mostra um exemplo simples de codificação. O buffer look-ahead também pode ser usado como uma extensão de “dicionário”, como exemplificado na Figura 5.8.

Na codificação LZ77, uma palavra-código requer

$$\text{ÿlog2 Sÿ} + \text{ÿlog2 (L ÿ 1)ÿ} + \text{ÿlog2 mÿ}$$

bits, onde S é o tamanho do “dicionário”, L é o tamanho do buffer de antecipação e m é o tamanho do alfabeto. Tomando como exemplo S = 2048, L = 33 e m = 256, uma palavra-código seria

Machine Translated by Google

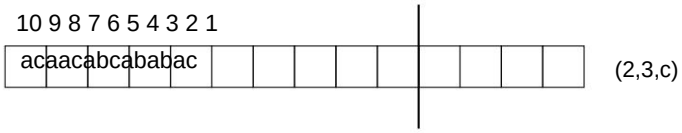


Figura 5.8: Exemplo de uso do buffer look-ahead para extensão “dicionário” na codificação LZ77.

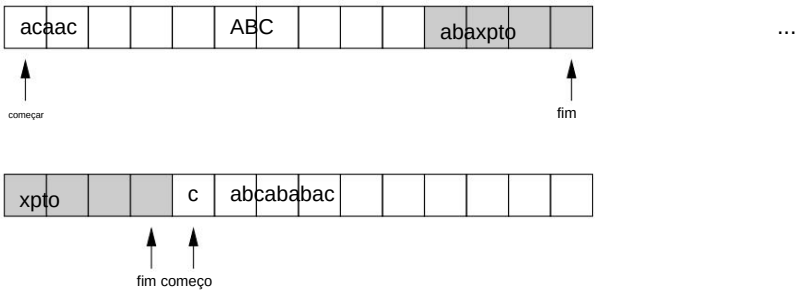


Figura 5.9: Buffer look-ahead implementado usando uma fila circular.

requerem 11 + 5 + 8 = 24 bits, que é três vezes o número de bits necessários para representar um único símbolo (8 bits).

Tópico de discussão

5.3 Há uma série de questões práticas que precisam ser abordadas ao implementar este algoritmo de compressão, como

- O tempo de busca requerido por cada etapa de codificação em função do tamanho do dicionário.
- A probabilidade de encontrar sequências correspondentes em função do tamanho do dicionário.
- O tempo de processamento exigido por cada etapa de codificação em função do buffer de antecipação tamanho.
- A probabilidade de encontrar sequências correspondentes em função do tamanho antecipado.
- O impacto no comprimento da palavra-código quando os tamanhos do dicionário e/ou buffer de antecipação são alterados.
- A potencial ineficiência do uso de palavras-código de três componentes ao codificar símbolos isolados.

Discuta-os.

A implementação de algoritmos do tipo LZ77 também precisa levar em consideração o uso eficiente de estruturas de dados. Por exemplo, o uso de buffers circulares (ver Fig. 5.9) e estruturas em árvore (ver Fig. 5.10) pode melhorar drasticamente a velocidade de processamento.

Machine Translated by Google

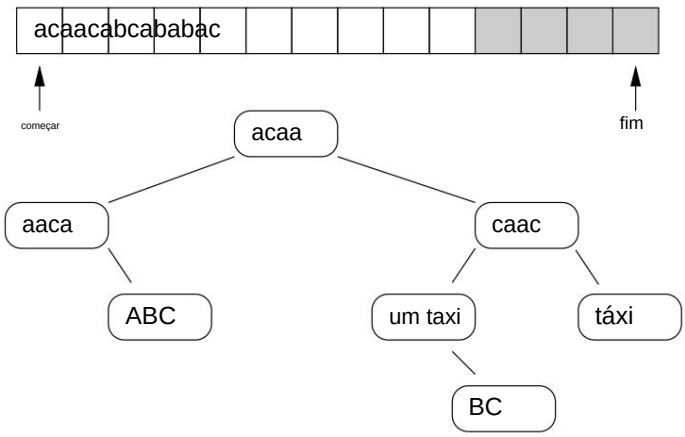


Figura 5.10: Dicionário implementado utilizando uma árvore binária.

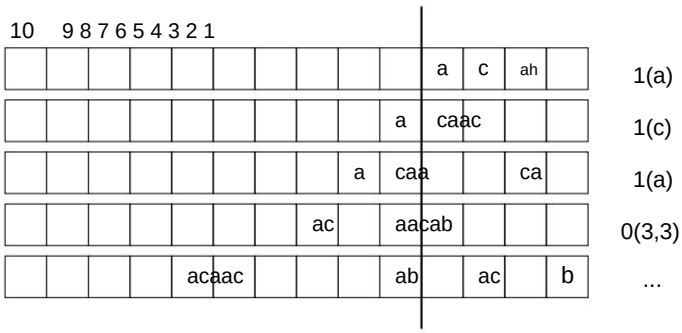


Figura 5.11: Exemplo de codificação LZSS.

5.2.4 Compressão LZSS

LZSS é uma variante do LZ77 proposta por James Storer (1953–) e Thomas Szymanski (Storer e Szymanski, 1982). Como mostrado na seção anterior, com o LZ77 às vezes é possível para produzir palavras-código que usam mais bits do que os bits originais da string que ela representa. Com LZSS, isso é evitado aplicando palavras-código com apenas **dois componentes – um** bit é usado para indicar se os dados são codificados ou enviados em formato literal e não codificado. Portanto, quando uma sequência não for encontrado no dicionário, o código do símbolo será enviado, em vez de (0, 0, . . .) como em LZ77. Para distinguindo ambos os casos, um bit é usado. A Figura 5.11 dá um exemplo de codificação usando LZSS.

5.2.5 Compressão LZ78

O segundo algoritmo proposto por Jacob Ziv e Abraham Lempel, conhecido como LZ78 (ou LZ2), foi publicado em

- J. Ziv e A. Lempel, Compressão de sequências individuais via codificação de taxa variável,

Machine Translated by Google

Dicionário:	um	aa	b	aaa	d	aab	aad	o
Índice:	1	2	3	4	5	6	7	8
Palavra-código:	0,a	1,a	0,b	2,a	0,d	2,b	2,d	0,o

Figura 5.12: Codificação LZ78 da string “aaabaaadaabaado”.

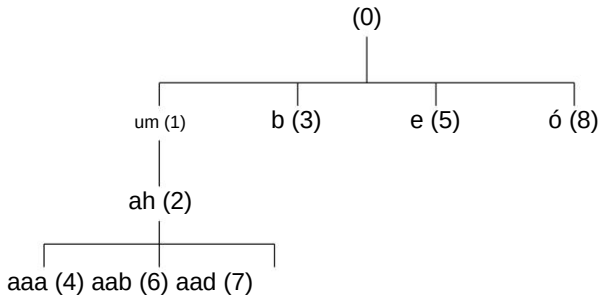


Figura 5.13: Estrutura de dados em árvore para acesso rápido ao dicionário LZ78.

IEEE Trans. na Teoria da Informação (Ziv e Lempel, 1978).

LZ78 conta com uma abordagem distinta daquela de LZ77, onde o uso de um dicionário explícito é probabilidade o mais óbvio. Na codificação LZ78:

- Os dados são analisados em strings.
- Cada string é construída a partir da maior string correspondente encontrada até o momento, mais a primeira não símbolo correspondente.
- A nova string é representada usando o índice da string correspondente (que está no dicionário), seguido do código do símbolo sem correspondência.
- Finalmente, a nova string é adicionada ao dicionário.

A Figura 5.12 mostra como a string “aaabaaadaabaado” é representada pelo algoritmo de codificação LZ78. Vale a pena notar que, aparentemente, esta nova estratégia parece não colocar restrições à distância na qual ocorrem padrões repetidos, enquanto com LZ77, se esta distância exceder o tamanho da área de pesquisa (o “dicionário”), eles não são levados em consideração. Mesmo assim, foi mostraram que ambas as técnicas são assintoticamente ótimas para fontes ergódicas, ou seja, assintoticamente, sua eficiência de codificação se aproxima da entropia da fonte.

Tal como acontece com algoritmos do tipo LZ77, é fundamental usar estruturas de dados apropriadas para acelerar em processamento. No caso do LZ78, uma possibilidade é representar o dicionário utilizando estruturas em árvore, como no exemplo da Fig. 5.13.

Tópico de discussão 5.4

Na prática, o dicionário não pode crescer sem limites, porque, por exemplo,

Machine Translated by Google

- O tamanho das palavras-código está relacionado ao tamanho do dicionário.
- A quantidade de memória para armazenar o dicionário (tanto pelo codificador como pelo decodificador) pode ser muito grande.

Portanto, na prática, é necessária a implementação de mecanismos que limitem o crescimento do dicionário. Discutir vantagens e desvantagens de possíveis mecanismos para fazê-lo, tais como,

- Dicionário "congelado", quando atinge algum tamanho pré-definido.
- Quando o dicionário atinge um tamanho predefinido, ele é excluído e reiniciado (observe que isso é idêntico à codificação em bloco).
- Quando o dicionário está cheio, algumas entradas são apagadas, por exemplo, aquelas que não são utilizadas por um mais tempo ou menos usado.

O tópico de discussão

5.5 LZ77 depende de um dicionário com sufixo completo, enquanto LZ78 depende de um dicionário com prefixo completo (explique por quê). Com base no Teorema 5.4, discuta as possíveis implicações deste fato.

5.2.6 Compressão LZW

Em 1984, Terry Welch (1939–1988) publicou um artigo onde são propostas soluções para alguns dos problemas associados ao LZ78, nomeadamente (Welch, 1984):

- Os símbolos vistos pela primeira vez são codificados de forma mais eficiente (em LZ78, dois componentes palavras-código são geradas neste caso).
- O número de componentes das palavras-código é reduzido a apenas **um.**

Inicialmente, todos os símbolos do alfabeto são inseridos no dicionário. Então, o algoritmo opera da seguinte forma:

- Os símbolos são lidos, um por um, da string a ser codificada.
- Esses símbolos (ÿ) são concatenados a uma string, x, enquanto xÿ pode ser encontrado no dictio nada.
- Quando não for possível adicionar mais um símbolo, o índice de x no dicionário é enviado, xÿ é inserido no dicionário e x é inicializado com o símbolo ÿ.

Machine Translated by Google

Exemplo 5.12

Considere a codificação da string “aaabaaadaabaado”. Inicialmente, o dicionário contém todos símbolos do alfabeto:

0 nulo	. . .	97 ab	98	99	100 c	d. . .	. . .	255

String: aa abaaadaabaado; Palavra-código enviada: 97

. . .	97	98	99	100	. . . ab c	d. . .	. . .	256 ah

String: aab aaadaabaado; Palavra-código enviada: 256

. . .	97	98	99	100	. . . ab c	d. . .	. . .	256 ah 257 aab

String: aaa ba aadaabaado; Palavra-código enviada: 98

. . . . .	97 a	98	99	100 b c	d. . .	. . .	256 ah	257 aab 258 BA

String: aaab aaa daabaado; Palavra-código enviada: 256

. . . . .	97 a	98	99	100 b c	d. . .	. . .	256 ah	257 aab	258 BA 259 aaa

String: aaabaa ad aabaado; Palavra-código enviada: 97

. . . . .	97 a	98	99	100 b c	d. . .	. . .	256 ah	257 aab ba	258 aaa anúncio 259 260

String: aaabaaa da abaado; Palavra-código enviada: 100

. . . . .	97 a	98	99	100 b cd . . . aa aab	. . .	256	257	258	259 260 BA	261 aaa anúncio da

String: aaabaaad aaba ado; Palavra-código enviada: 257

. . .	97	98	99	100 . . . ab c	d. . .	. . .	256 ah	257 aab	258 259 260 ba aaa ad	261 262 da aaba

String: aaabaaadaab aad o; Palavra-código enviada: 256

. . .	97	98	99	100 . . . abc d. . .	. . . aa aab	256 ba	257 aaa	258 ad	259 260 da aaba	261 aad	262	263

Machine Translated by Google

. . .

Exemplo 5.13

A decodificação é realizada da seguinte maneira. Inicialmente, o dicionário contém todos os símbolos do alfabeto.

0 nulo. . .	. . . 97 98 99 100	ab		c	d. . .	. . .	255

Palavra-código recebida: 97; Sequência: uma?

	. . . 97 98 99 100		. . . ab	c	d. . .	. . .	256 a?

Palavra-código recebida: 256; Sequência: aa?

	. . . 97 98 99 100		. . . ab	c	d. . .	. . .	256 257 ah ah?

Palavra-código recebida: 98; Sequência: aaa b?

. . . 97 98 99 100b . . .	a		c	d. . .	. . .	256 257 258 ah aab	b?

Palavra-código recebida: 256; Sequência: aaab aa?

. . . 97 98 99 100b . . .	a		c	d. . .	. . .	256 257 258 259 ah aab BA	ah?

Palavra-código recebida: 97; String: aaabaa a?

. . . 97 98 99 100b . . .	a		c	d. . .	. . .	256 257 258 259 260 ah aab ba aaa a?	

Palavra-código recebida: 100; String: aaabaaa d?

. . . 97 98 99 100b . . .	a		cd . . . aa aab	. . .	256 257 258 259 260 261	BA aaa anúncio d?	

Palavra-código recebida: 257; String: aaabaaad aab?

. . . 97 98 99 100			. . . ab c	d. . .	. . .	256 257 258 259 260 261 262 ah aab ba aaa ad da aab?	

Palavra-código recebida: 256; String: aaabaaadaab aa?

. . . 97 98 99 100			. . . abc d. . .	. . . aa aab ba aaa ad da aaba aa?	256 257 258 259 260 261			262	263

. . .

Machine Translated by Google

5.3 Codificação aritmética

5.3.1 Motivação

Vimos que uma das principais limitações dos códigos de comprimento variável é que eles só são ótimos se as probabilidades dos símbolos forem potências de 1/2.

Exemplo 5.14
Suponha que temos

$$p_1 = \frac{1}{4}, \quad p_2 = \frac{1}{4}, \quad p_3 = \frac{1}{2}$$

A entropia de primeira ordem desta fonte de informação é de 1,5 bits/símbolo. Usando um código de Huffman, temos, por exemplo,

$$\ddot{y}_1 \rightarrow 0 \quad \ddot{y}_2 \rightarrow 10 \quad \ddot{y}_3 \rightarrow 11$$

O comprimento médio deste código é de 1,5 bits/símbolo e, portanto, sua redundância é zero (ou seja, este código é ótimo para esta fonte).

Exemplo 5.15
Considere agora que a distribuição de probabilidade dos símbolos fonte é

$$p_1 = \frac{2}{3}, \quad p_2 = \frac{1}{6}, \quad p_3 = \frac{1}{6}.$$

Neste caso, a entropia da fonte é ÿ 1,26. Contudo, como no caso anterior, o algoritmo de codificação de Huffman atribui 1 bit a ÿ1 e 2 bits aos outros dois símbolos. O comprimento médio deste código é ÿ 1,33, ou seja, 0,07 bits por símbolo a mais que a entropia da fonte. A diferença entre o comprimento médio do código e a entropia aumenta quando max(pi) > 0,5 e max(pi) ÿ 1.

Tal como acontece com a codificação de Huffman, a codificação aritmética necessita de uma estimativa das probabilidades dos símbolos do alfabeto. A codificação aritmética representa a mensagem inteira por uma única palavra-código: um número no intervalo [0, 1). É por isso que a codificação aritmética não é afetada pela assimetria da distribuição de probabilidade, como podem ser os códigos de comprimento variável. Além disso, a codificação aritmética tem a vantagem de permitir uma separação clara entre o processo **de codificação** e **a modelagem da fonte.**

5.3.2 Codificação

Para codificação, o intervalo [0, 1) é particionado de acordo com a distribuição de probabilidade dos símbolos. A única restrição é que o decodificador use o mesmo particionamento.

Machine Translated by Google

Exemplo 5.16

Considere o seguinte alfabeto e distribuição de probabilidade correspondente:

Símbolo	Prob.	Intervalo
d	0,1	[0, 0,1)
e	0,3	[0,1, 0,4)
f	0,1	[0,4, 0,5)
g	0,05	[0,5, 0,55)
n	0,1	[0,55, 0,65)
R	0,2	[0,65, 0,85)
você	0,05	[0,85, 0,9)
espaço	0,1	[0,9, 1)

Consideremos a codificação da mensagem “fred”.

Inicialmente, o intervalo de codificação é [0, 1). O primeiro símbolo a codificar, “f”, tem o intervalo [0,4, 0,5) atribuído a ele, de acordo com a tabela de probabilidades. Após a codificação deste símbolo, o intervalo de codificação será [0,4, 0,5). O próximo símbolo, “r”, está associado ao intervalo [0,65, 0,85). Então, o limite inferior do novo intervalo de codificação será

$$0,4 + 0,65(0,5 - 0,4) = 0,465$$

e o limite superior

$$0,4 + 0,85(0,5 - 0,4) = 0,485.$$

O próximo símbolo, “e”, transforma o intervalo de codificação atual, [0,465, 0,485), em [0,467, 0,473), de acordo com

$$0,467 = 0,465 + 0,1(0,485 - 0,465)$$

e

$$0,473 = 0,465 + 0,4(0,485 – 0,465).$$

Após codificar o símbolo “d”, obtemos o intervalo [0,467, 0,4676).

Portanto, se o intervalo de codificação atual for [L_n, H_n), e se o próximo símbolo a codificar for repre enviado pelo intervalo [l, h), então

$$eu^{n+1} = eu^{n} + eu(Hn - eu_{n})$$

e

$$H^{n+1} = eu^{n} + h(Hn - eu_{n}).$$

Exemplo 5.17

Considere a codificação da string binária 0110, usando P(0) = 0,6 e P(1) = 0,4. O a mensagem codificada é um número no intervalo [0,504, 0,5616) (ver Fig. 5.14). Outro possível A visão do processo de codificação pode ser vista na Fig. 5.15, neste caso com intervalos renormalizados.

Machine Translated by Google

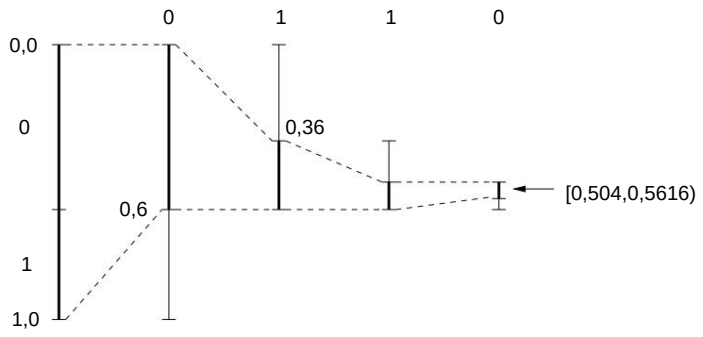


Figura 5.14: Exemplo de codificação aritmética de uma string simples.

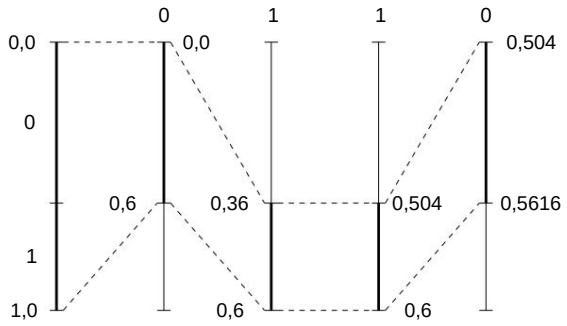


Figura 5.15: O mesmo exemplo da Fig. 5.14, mas com uma visualização diferente dos intervalos.

5.3.3 Decodificação

Para decodificar a mensagem “fred”, procedemos da seguinte forma. Seja x o valor recebido pelo decodificador (ou seja, a mensagem codificada), tal que x ÿ [0,467, 0,4676). Por exemplo, considere x = 0,467. Como x ÿ [0,4, 0,5), então o primeiro símbolo da mensagem é “f”. O efeito de este símbolo é então eliminado da mensagem codificada, utilizando o seguinte procedimento:

1. Subtraia o limite inferior do intervalo “f” de x, ou seja, x ÿ 0,4 = 0,467 ÿ 0,4 = 0,067.

2. Divida esse novo valor pela faixa do intervalo “f” (0,1, neste caso): 0,067/0,1 = 0,67.

O novo número, x = 0,67, está no intervalo [0,65, 0,85), o que significa que “r” é o próximo símbolo. Renormalizando, obtemos x = (0,67 ÿ 0,65)/(0,85 ÿ 0,65) = 0,1. Porque x ÿ [0,1, 0,4), então o próximo símbolo é “e”. Renormalizando novamente, x = (0,1 - 0,1)/(0,4 - 0,1) = 0. Como x ÿ [0, 0,1), então o símbolo é “d”.

Observe que não há como detectar, automaticamente, o final de uma mensagem codificada usando codificação aritmética. Portanto, é necessário indicar o comprimento da mensagem (ou seja, o número de símbolos da mensagem) ou usar um símbolo especial de “fim de mensagem”.

Machine Translated by Google

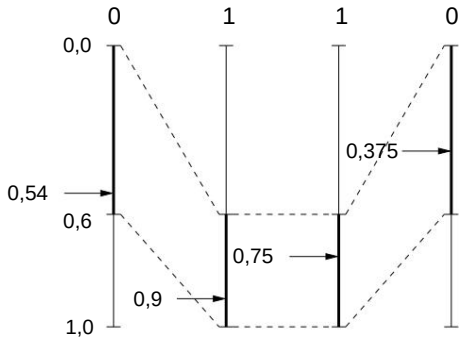


Figura 5.16: Decodificação do exemplo representado nas Figs. 5.14 e 5.15.

Exemplo 5.18 O
processo de decodificação do exemplo anterior pode ser visto na Figura 5.16, onde foi considerado x = 0,54.

5.3.4 Questões de implementação

Obviamente, não é possível obter um codificador útil através da implementação direta do processo descrito. Na verdade, a precisão da representação de ponto flutuante do computador se esgotaria rapidamente. No entanto, mesmo que tivéssemos acesso a um computador com precisão infinita, certamente não seria eficiente realizar operações aritméticas com vários milhares ou milhões de casas decimais. Além disso, geralmente não é aceitável ter que esperar pelo fim da mensagem para começar a receber os primeiros bits da mensagem codificada.

Como vimos anteriormente, à medida que novos símbolos chegam ao codificador, os limites do intervalo de codificação vão se aproximando, implicando na necessidade de uma precisão crescente para distingui-los. Contudo, é razoável admitir que, a certa altura, ambos os limites ficarão acima de 0,5 ou abaixo de 0,5. Quando isso acontece, o bit mais significativo de L e H permanecerá permanentemente igual a 0, se L < 0,5 e H < 0,5, ou igual a 1, se L ÿ 0,5 e H ÿ 0,5. Notar que

x ÿ 0,5(10) ÿÿ x = 0,1 . . .(2) x <
0,5(10) ÿÿ x = 0,0 . . .(2)

Portanto, este bit pode ser enviado pelo codificador.

Por outro lado, podemos expandir o intervalo da seguinte maneira,

Bit enviado: 0 [0, 0,5) ÿÿ [0, 1) x ÿ 2x [0,5, 1) ÿÿ [0,
Bit enviado: 1 1) x ÿ 2(x ÿ 0,5),

o que, em princípio, nos permite manter a precisão sob controle.

Mas se L ÿ 0,0111 . . .(2) e H ÿ 0,1000 . . .(2)?

Machine Translated by Google

Para levar em conta esta situação, sempre que o intervalo de codificação pertencer a [0,25, 0,75), realizamos a expansão

[0,25, 0,75) ÿÿ [0, 1) x ÿ 2(x ÿ 0,25),

e um contador anota quantas vezes essas redimensionações ocorreram desde que o último bit foi enviado pelo codificador.

Observe que esta operação é realizada quando 0,25 ÿ L < 0,5 (L = 0,01 . . . (2)) e 0,5 ÿ H < 0,75 (H = 0,10 . . . (2)). Após esta mudança de escala, obtemos L = 0,0. . .(2) e H = 0,1 . . .(2).
O resultado desta operação é que o segundo bit mais significativo é “excluído”.

Se, após incorporar o próximo símbolo, L ÿ 0,5 e H ÿ 0,5, então é enviado o “1” correspondente, seguido de tantos bits “0” quanto o valor do contador. Por outro lado, se L < 0,5 e H < 0,5, é enviado o bit “0” correspondente, seguido de tantos bits “1” quanto o valor do contador.

Machine Translated by Google

Machine Translated by Google

6 Codificação preditiva

6.1 Princípios

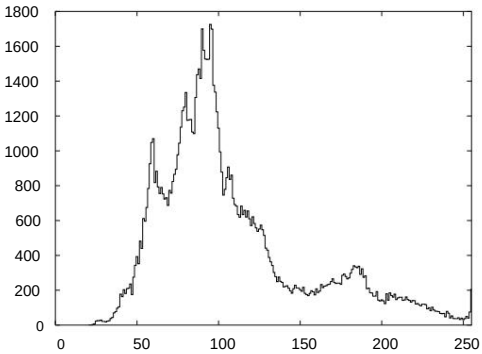
Deixe x_1^n = x1x2. . . xn será a sequência de valores (escalares ou vetores) produzidos por uma fonte de informação até o momento n. A codificação preditiva é uma técnica que se baseia na codificação da sequência R_1^n =r1r2. . . rn, em vez da sequência original x_1^n, onde

$$r_n = x_n \ddot{y} \hat{x}_n \tag{6.1}$$

e

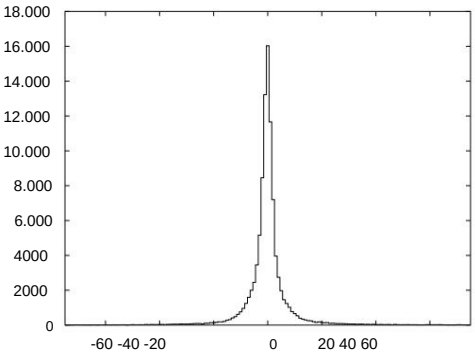
$$\hat{x}_n = p(x_1^{n-1}) = p(x_1x_2 \ldots x_n\ddot{y}1). \tag{6.2}$$

O $\hat{x}_n$ é a estimativa no tempo n e o valor de rn é conhecido como resíduo. A função p() é o estimador ou preditor. O objetivo da codificação preditiva é ter L(r_1^n) < L(x_1^n), onde L() denota o comprimento do código alcançado por métodos de compressão "adequados" para comprimir r_1^n e x_1^n. A Figura 6.1 dá um exemplo deste princípio.



Original

H = 7,26 bits/símbolo



Preditor 1 JPEG

H = 4,49 bits/símbolo

Figura 6.1: Comparação da entropia de primeira ordem calculada sobre os valores originais dos pixels da imagem e dos resíduos resultantes de um preditor simples. Os gráficos mostram os histogramas de x_1^n (à esquerda) e de r_1^n (à direita).

Machine Translated by Google

6.2 Técnicas de codificação preditiva

6.2.1 Previsão 1D simples

Alguns codificadores de áudio sem perdas (por exemplo, FLAC) usam preditores polinomiais simples, como

$$\begin{cases} \hat{x}_n^{(0)} = 0 \\ \hat{x}_n^{(1)} = x_{n-1} \\ \hat{x}_n^{(2)} = 2x_{n-1} - x_{n-2} \\ \hat{x}_n^{(3)} = 3x_{n-1} - 3x_{n-2} + x_{n-3} \end{cases} \tag{6.3}$$

O primeiro, $\hat{x}_n^{(0)}$, fornece uma estimativa que é sempre zero (isto é útil, por exemplo, quando (1) o codificador está iniciando, porque nenhum valor anterior está disponível ainda). A segunda, $\hat{x}_n^{(1)}$, assume que o sinal é constante, ou seja, que o próximo valor é igual ao valor presente. O terceiro utiliza uma extrapolação por linha reta e o último por uma função parabólica.

Problema 6.1

Tente deduzir as fórmulas para $\hat{x}_n^{(2)}$ e para $\hat{x}_n^{(3)}$ mostrado em (6.3).

No caso particular dos preditores dados em (6.3), os resíduos correspondentes podem ser calculados eficientemente de forma recursiva, usando

$$\begin{cases} R_n^{(0)} = x_n(0) \\ r_n^{(1)} = r_n - r_{n-1}^{(0)} \\ r_n^{(2)} = r_n^{(1)} - r_{n-1}^{(1)} \\ R_n^{(3)} = r_n^{(2)} - r_{n-1}^{(2)} \end{cases} . \tag{6.4}$$

6.2.2 Técnicas preditivas de codificação de imagens

Normalmente, as imagens são codificadas da esquerda para a direita, de cima para baixo, ou seja, em ordem **de varredura raster ,** conforme ilustrado na Figura 6.2.

Neste caso, a sequência x_1^n é obtido concatenando as primeiras linhas da imagem $\lfloor n/NC \rfloor$, mais pixels mod NC do número da linha $\lfloor n/NC \rfloor + 1$.

Outras abordagens utilizam decomposições hierárquicas (ou **multi-resolução).** É o caso do método HINT (Hierarchical INTerpolation), ilustrado na Figura 6.3.

6.2.3 Preditores

Obviamente, bons preditores são fundamentais na codificação preditiva. Para uma codificação eficiente, os valores estimados devem ser o mais próximos possível dos valores reais, ou seja, os valores de r_1^n devem ser os menores possíveis.

Machine Translated by Google

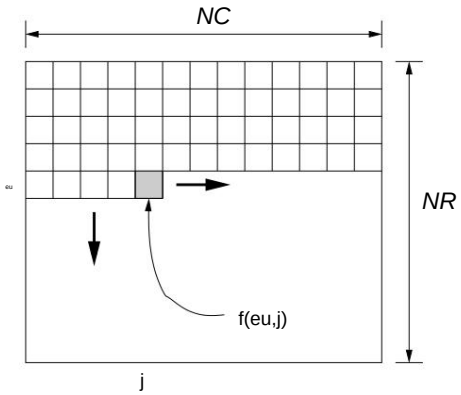


Figura 6.2: Movimento raster-scan ao longo de uma imagem com linhas NR e colunas NC , onde f(i, j) denota o valor do pixel na linha i e na coluna j.

1	5	3	5	1	5	3	5	1	5	3	5	1
5	4	5	4	5	4	5	4	5	4	5	4	5
3	5	2	5	3	5	2	5	3	5	2	5	3
5	4	5	4	5	4	5	4	5	4	5	4	5
1	5	3	5	1	5	3	5	1	5	3	5	1
5	4	5	4	5	4	5	4	5	4	5	4	5
3	5	2	5	3	5	2	5	3	5	2	5	3
5	4	5	4	5	4	5	4	5	4	5	4	5
1	5	3	5	1	5	3	5	1	5	3	5	1

Figura 6.3: Abordagem de varredura utilizada pelo método HINT. Primeiro, os pixels marcados com “1” estão codificados. Então, os pixels marcados com “2” são previstos a partir daqueles marcados com “1” e os resíduos correspondentes são codificados. Pixels marcados com “3” são codificados de maneira semelhante, mas neste caso a previsão é feita a partir daqueles marcados com “1” e “2”, e assim por diante.

Observe que o decodificador deve ser capaz de gerar a mesma sequência, $\hat{x}_1^{n-1}$, de valores estimados. Em outras palavras, **o preditor não pode introduzir nenhum erro** durante a codificação/decodificação, caso contrário o sinal decodificado divergirá do original. Portanto, o preditor deve ser **causal** e, no caso de codificação com perdas, o preditor no codificador **deve usar os valores reconstruídos,** $\tilde{x}_1^{n-1}$, em vez dos valores originais, x_1^{n-1}. A Figura 6.4 mostra um diagrama de blocos de um codec baseado na codificação preditiva, onde os resíduos sofrem alguma forma de quantização (ou seja, o codec é com perdas).

Problema 6.2

Procure encontrar exemplos que demonstrem esta divergência, ou seja, quando o resíduo calculado pelo decodificador é diferente daquele usado pelo codificador.

Machine Translated by Google

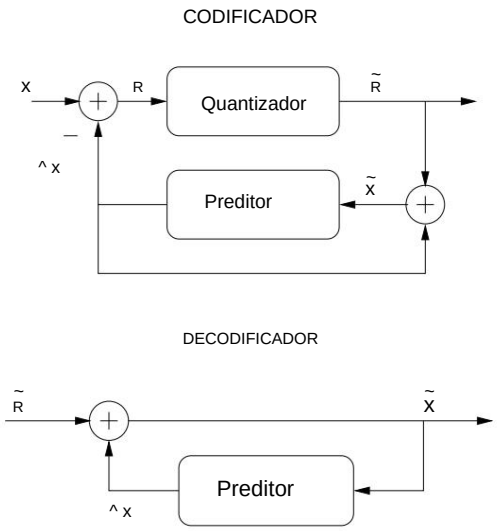


Figura 6.4: Diagrama de blocos de um codec com perdas baseado em predição.

Geralmente, a complexidade do preditor depende de dois aspectos:

- O número de valores utilizados para calcular as estimativas **(a ordem do preditor).**
- A configuração espacial (ou temporal) destes valores.

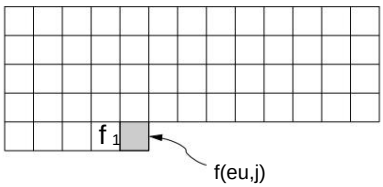


Figura 6.5: Exemplo de preditor de imagem de ordem 1, onde a predição corresponde ao valor do pixel previamente codificado.

Considere o exemplo de um preditor espacial de ordem 1, onde o valor estimado é dado pelo valor imediatamente anterior (conforme mostrado na Figura 6.5). Este tipo de preditor pode ser facilmente estendido para ordens superiores, utilizando os últimos k pixels processados da imagem. Porém, para pedidos superiores a 3 ou 4, a eficiência não aumenta significativamente (na verdade, pode até diminuir).
Isso acontece porque as imagens são sinais 2D, e não sequências de dados 1D. Portanto, geralmente, as configurações espaciais utilizadas para codificação preditiva de imagens possuem um formato 2D. A Figura 6.6 mostra um exemplo de preditor de imagem 2D.

6.3 Codificação preditiva sem perdas

Uma das principais vantagens da codificação preditiva é permitir um design simples de codificadores sem perdas. Na verdade, a maioria dos codificadores sem perdas para áudio e imagem dependem de codificação preditiva

Machine Translated by Google

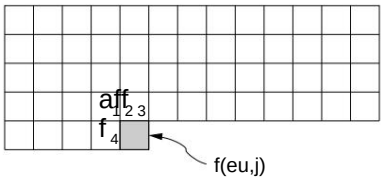


Figura 6.6: Exemplo de preditor de imagem 2D. Neste caso, além do valor previamente codificado (denotado aqui como f4), o preditor também usa valores vizinhos do previamente codificado linha.

técnicas. Contudo, para codificação sem perdas, existe uma restrição adicional em relação ao preditor: as estimativas geradas devem ser independentes de plataforma. Esta restrição pode ser atendida, por por exemplo, usando aritmética inteira. A Figura 6.7 mostra um diagrama de blocos de um sistema baseado em predição codec sem perdas.

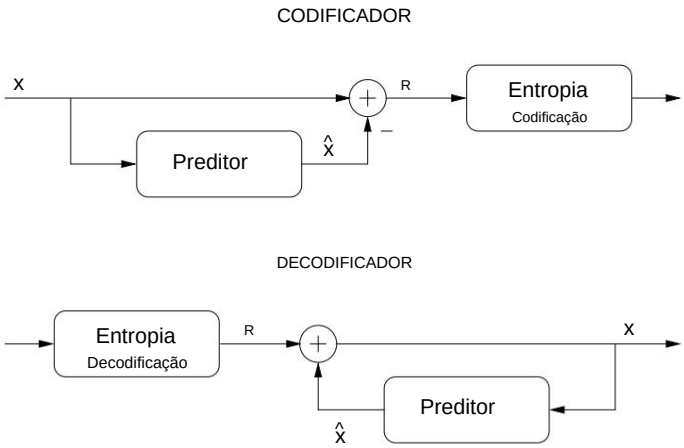
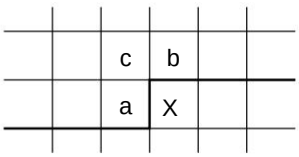


Figura 6.7: Diagrama de blocos de um codec sem perdas baseado em predição.

6.3.1 Predição linear: o modo sem perdas do JPEG

O modo sem perdas do JPEG (ISO/IEC 10918-1, ITU-T T.81, 1992) fornece sete **preditores lineares,** de acordo com



Preditor de modo	
1	a
2	b
3	c
4	uma + b - c
5	uma + (b - c)/2
6	b + (uma - c)/2
7	(a + b)/2

(6.5)

Machine Translated by Google

Preditor	2	3	4	5	6	7
1 Entropia 4,49	4,21	4,74	4,17	4,16	**4,04**	4,10

Figura 6.8: Exemplo de entropia de primeira ordem dos resíduos de predição fornecidos pelos sete preditores lineares do modo sem perdas de JPEG.

Geralmente, o desempenho dos vários preditores pode variar consideravelmente de imagem para imagem. Se o tempo de codificação não for um problema, então todos eles poderão ser testados e aquele com o melhor taxa de compressão escolhida. A Figura 6.8 mostra um exemplo do desempenho alcançado por esses diferentes preditores, estimados usando uma medida de entropia de primeira ordem.

6.3.2 O preditor não linear de JPEG-LS

JPEG-LS (ISO/IEC 14495-1, ITU-T T.87, 1999) usa um preditor baseado no mesmo configuração como a do JPEG, mas, em vez de um preditor linear, usa o **preditor não linear**

$$\hat{x} = \begin{cases} \text{min}(a, b) \text{ se } c \geq \text{max}(a, b) \\ \text{max}(a, b) \text{ se } c \geq \text{min}(a, b) \\ a + b - c \text{ caso contrário} \end{cases} \tag{6.6}$$

Observe que a parte linear deste preditor (a+b÷c) é igual ao preditor número 4 do JPEG.
A Figura 6.9 mostra exemplos do desempenho (estimado usando entropia de primeira ordem) alcançado por este preditor, bem como aqueles de JPEG sem perdas.

6.4 Previsão temporal e compensação de movimento

Normalmente, as diferenças entre um quadro e o quadro anterior de uma sequência de vídeo são devido ao movimento dos vários elementos da cena visual (ver Fig. 6.10). Exceções ocorrem quando há mudanças de cena, operações de zoom in/zoom out ou tradução de câmera.

Para explorar esta redundância, é frequente utilizar **a previsão temporal,** que normalmente se baseia em **compensação de movimento.** Esta é uma das principais técnicas usadas para obter compressão entre quadros.

Machine Translated by Google

(a) (b)

Preditor	1	2	3	4	5	6	7	JLS
Entropia (a)	4,49	4,21	4,74	4,18	4,16	4,04	4,10	**3,98**
Entropia (b)	5,60	5,05	5,82	5,19	5,23	4,97	5,15	**4,93**

Figura 6.9: Exemplo de entropia de primeira ordem dos resíduos de predição fornecidos pelo preditor não linear do JPEG-LS e também, para comparação, dos sete preditores lineares do JPEG sem perdas.

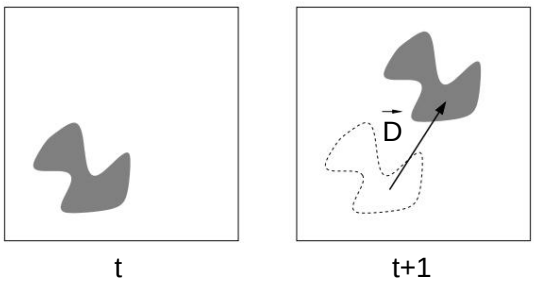


Figura 6.10: Ilustração do princípio por trás da compensação de movimento, onde a posição do o objeto no quadro t + 1 é estimado (previsto) a partir de sua posição no tempo t.

Uma técnica mais simples que também aproveita a correlação temporal entre quadros é codificação de vídeo **de reposição condicional .** Encontra zonas no quadro de vídeo onde havia muda em relação ao quadro anterior, e apenas essas zonas são codificadas. Esta técnica não usa compensação de movimento. Ele apenas realiza uma detecção de atividade temporal.

De forma mais geral, a codificação de vídeo baseada na **compensação de movimento** envolve as seguintes etapas:

- Estimativa dos vetores de movimento.
- Compensação, ou seja, previsão temporal.
- Codificação dos vetores de movimento.
- Codificação dos resíduos de predição.

Machine Translated by Google

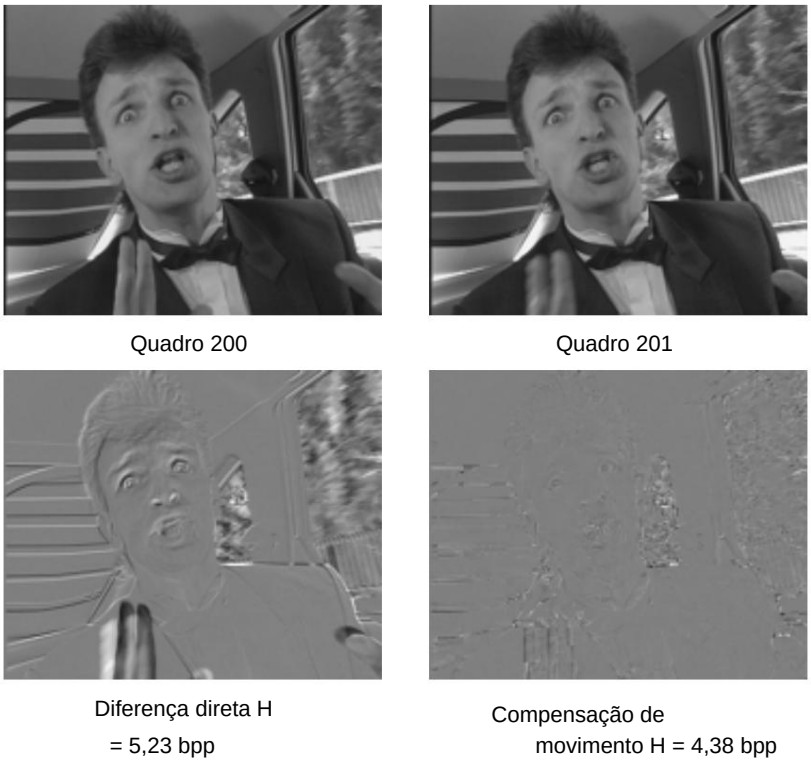


Figura 6.11: Entropia de primeira ordem dos resíduos obtidos por diferença direta de referenciais e após compensação de movimento.

Há um grande número de técnicas para detecção de movimento, mas uma delas é claramente a abordagem mais comum para codificação de vídeo. Para cada bloco de quadro (por exemplo, de N × N pixels), ele busca a posição onde minimiza alguma medida em relação ao quadro anterior **(o quadro de referência).** Observe que esta abordagem tenta encontrar a posição que minimiza uma medida de interesse, que pode não corresponder ao verdadeiro movimento da cena.

Normalmente, queremos minimizar alguma medida D(i, j), como

$$D(eu, j) = \sum_{r=1}^{N} \sum_{c=1}^{N} dg(r, c, t) \ddot{y} g(i + r, j + c, t + 1) \tag{6.7}$$

onde d(·) é, por exemplo, $(\cdot)^2$ ou $|\cdot|$.

Devido a restrições de complexidade, a busca é limitada a uma vizinhança de (N +2ÿ)×(N +2ÿ) pixels ao redor do bloco, ou seja, ÿÿ ÿ i, j ÿ ÿ. A Tabela 2 mostra exemplos de entropia de primeira ordem dos resíduos e vetores de movimento para alguns valores de N e ÿ (ver também Fig. 6.11).

Se for utilizada uma busca exaustiva, é garantido encontrar o mínimo de D(i, j). No entanto, esta abordagem é geralmente muito exigente do ponto de vista computacional, portanto, outras técnicas abaixo do ideal foram propostas.

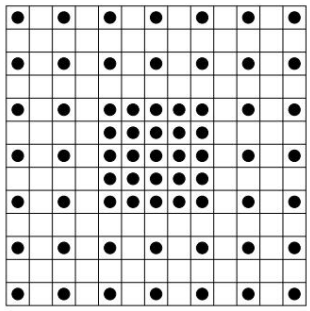
Várias das abordagens superótimas para encontrar o melhor bloco de referência dependem de subamostragem espacial. Por exemplo, considerando que a zona mais provável para encontrar o bloco de referência

Machine Translated by Google

Tabela 2: Entropia de primeira ordem dos resíduos de predição e vetores de movimento (MV) obtidos para vários valores do tamanho do bloco, N, e área de pesquisa, ÿ.

ÿN	Entropia			
	Residuais VM	Nominal VM	Total	
3 4 7 4	4,29	0,50	0,31	4,60
15 4 3	4,03	1,00	0,39	4,42
8 7 8	3,91	2,00	0,47	4,38
15 8 3	4,63	0,12	0,07	4,70
16	4,43	0,25	0,09	4,52
	4,36	0,50	0,10	4,46
	4,84	0,03	0,01	4,85
7 16 15	4,76	0,06	0,02	4,78
16	4,72	0,12	0,02	4,74

Total: 169 blocos



Abaixo do ideal: 65 blocos

Figura 6.12: Exemplo de uma busca abaixo do ideal para o bloco de referência que usa uma busca mais refinada em torno da posição original do bloco.

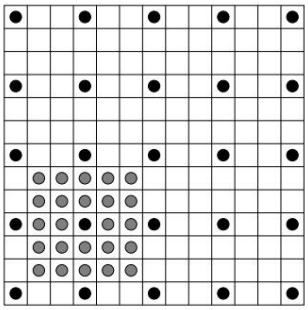
está na vizinhança do bloco, então podemos usar um esquema como o mostrado na Fig. 6.12.

Se considerarmos que depois de encontrar um bloco de referência razoavelmente bom é provável que outros melhor do que ele mesmo pode ser encontrado na vizinhança, então podemos usar uma busca gananciosa, como aquele ilustrado na Fig. 6.13.

Muitas outras variantes de busca local foram propostas, como, por exemplo, busca logarítmica. pesquisa (ver Fig. 6.14).

Machine Translated by Google

Total: 169 blocos



Abaixo do ideal: 49 blocos

Figura 6.13: Exemplo de uma busca subótima para o bloco de referência que utiliza uma busca mais detalhada em torno de um bloco encontrado após uma busca grosseira inicial.

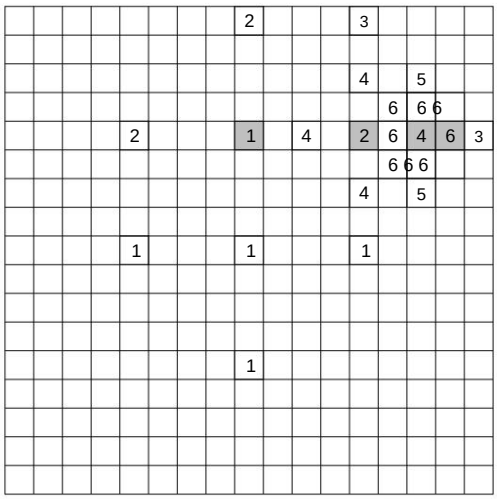


Figura 6.14: Exemplo de busca subótima (logarítmica) do bloco de referência.

Machine Translated by Google

7 Uma medida algorítmica de informação

7.1 Motivação

Conforme mostrado anteriormente, a formulação de uma medida probabilística de informação explora regularidades estatísticas dos dados, por meio de modelagem probabilística das fontes de informação.
No entanto, existem regularidades que não podem ser capturadas apenas por modelos probabilísticos. Por exemplo, diversas abordagens que envolvem a análise estatística dos dígitos iniciais de ÿ não conseguiram revelar desvios significativos daquilo que é normalmente entendido como **aleatoriedade estatística.** No entanto, a sequência inicial de dígitos de ÿ pode ser gerada por programas simples, mostrando que o número ÿ está longe de ser aleatório! Na verdade, intuitivamente, podemos dizer que uma string é aleatória se não puder ser **comprimida,** ou seja, se não for possível encontrar uma **descrição mais curta** para ela. Esta é precisamente a ideia central por trás da medida algorítmica da informação.

Tópico de discussão
7.1 Na compactação de dados, qual é o papel do descompressor e da string que representa os dados compactados? Faz sentido considerar o descompressor como um "intérprete" de uma determinada linguagem, formado por todas as strings comprimidas possíveis?

Exemplo 7.1
Considere as três strings binárias a seguir:

1. 0101010101010101010101010101010101010101010101010101010101010101

2. 0110101000001001111001100110011111110011101111001100100100001000

3. 1101111001110101111101101111101110101101111000101110010100111011

Quais devem ser as descrições mais curtas para eles? O primeiro parece ser fácil de descrever.
Aparentemente, o segundo parece aleatório, mas na verdade é apenas a representação binária inicial de ÿ 2 ÿ 1.
O terceiro parece aleatório e provavelmente é (no entanto, em geral, é impossível prová-lo!).

Agora, considere a representação de uma sequência gerada pelo lançamento de uma moeda honesta n vezes, tal como

00100101110101010 . . . 010

Existem 2 n dessas sequências e elas são igualmente prováveis. É altamente provável que tal sequência não possa ser comprimida. Portanto, pode não haver um programa melhor (mais curto) para gerar esta sequência como "print 00100101110101010 . . . 010". Assim, a complexidade descritiva de uma sequência binária aleatória é tão longa quanto a própria sequência.

Machine Translated by Google

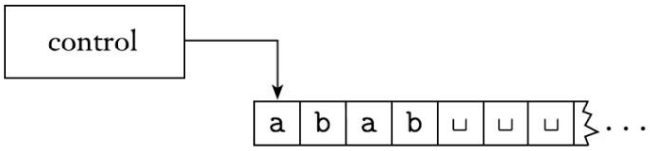


Figura 7.1: Representação esquemática de uma máquina de Turing.

7.2 Máquinas de Turing

Uma questão que pode surgir quando se pensa em computadores é: um computador pode resolver qualquer tipo de problema? E, se a resposta a esta pergunta for “não”: Quais são os problemas que não podem ser resolvidos pelos computadores?

Na verdade, de acordo com Nigel J. Cutland (1944–), no seu livro “Computability: An introdution to recursive function theory”, Cambridge University Press (Cutland, 1980),

> “Poderíamos descrever a teoria da computabilidade, do ponto de vista da ciência da computação, como começando com a questão O que os computadores podem fazer em princípio (sem restrições de espaço, tempo ou dinheiro)? – e, por implicação – Quais são as suas limitações teóricas inerentes? ”

Assim, um dos objetivos do campo da teoria da computabilidade é justamente estudar quais problemas são solucionáveis por computadores e quais não são. Contudo, para atingir este objectivo, precisamos primeiro de ter uma definição precisa do que queremos dizer com computador digital.

Primeiro, vamos introduzir alguma notação. Definimos um alfabeto como um conjunto finito não vazio. Referimo-nos aos membros do alfabeto como símbolos. Normalmente, e sem perda de generalidade, consideramos o alfabeto ÿ = {0, 1}. Uma string sobre um alfabeto é uma sequência finita de símbolos desse alfabeto. Por exemplo, se ÿ = {0, 1}, então 011101 é uma string sobre o alfabeto ÿ. Se s for uma string, então representamos por |s| o comprimento (isto é, número de símbolos) de s. Denotamos por ÿ ÿ o conjunto de todas as cadeias de comprimento finito, ou seja,

$$ÿ^{ÿ} = \{ÿ, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, \ldots\},$$

onde ÿ denota a string vazia, ou seja, |ÿ| = 0.

Por conveniência, assumimos uma ordenação das strings, tal que elas apareçam ordenadas por tamanho e, para o mesmo tamanho, ordenadas lexicograficamente (esta é a ordenação usada para exibir o conjunto acima). Observe que uma dessas ordenações permite colocar todas as strings de comprimento finito em uma correspondência um-para-um com os inteiros e, portanto, tratar as strings como inteiros.

Uma linguagem, A, é um subconjunto de ÿ ÿ , ou seja, A ÿ ÿ ÿ Uma linguagem é livre de prefixo se nenhuma string da linguagem é um prefixo próprio de outra string da linguagem.

Uma máquina de Turing é composta por um controle baseado em um conjunto finito e não vazio de estados e acesso a uma quantidade ilimitada e irrestrita de memória (Fig. 7.1). É um modelo matemático (ou abstração) de um dispositivo de computação hipotético onde:

Machine Translated by Google

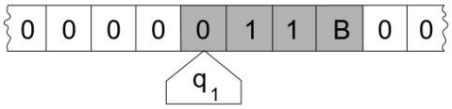


Figura 7.2: Exemplo de fita e cabeçote de uma máquina de Turing. O q1 escrito dentro do cabeçote indica o estado interno atual da máquina.

1. Em cada etapa, a máquina está em um conjunto predefinido de estados finitos (não vazios).
2. Símbolos de um conjunto finito (não vazio) predefinido podem ser lidos e escritos em um fita (ver Fig. 7.2, por exemplo).
3. A cabeça de leitura e gravação pode se mover tanto para a esquerda quanto para a direita.
4. A fita é ilimitada.
5. No início, a fita está em branco, exceto por um número finito de quadrados.
6. A máquina para quando um estado especial (estado de parada) é encontrado ou quando não consegue encontrar um estado de continuação.
7. Os estados especiais para rejeitar e aceitar strings de uma linguagem, que no caso das máquinas de Turing entram em vigor imediatamente após a parada, mesmo sem esgotar a entrada. Pode haver um estado de parada específico para aceitação e outro para rejeição, ou apenas um único estado de parada e a interpretação de aceitação ou rejeição deixada escrita na fita.

Segundo Marvin Minsky (1927-2016), em seu livro “Computation: Finite and Infinite Ma chines”, Prentice-Hall, Inc.

> “Uma máquina de Turing é uma máquina de estado finito associada a um armazenamento externo ou meio de memória. Este meio tem a forma de uma sequência de quadrados, marcados em uma fita linear. A máquina é acoplada à fita através de uma cabeça que fica situada, a cada momento, em algum quadrado da fita (Fig. 7.2). A cabeça possui três funções, todas exercidas em cada ciclo de operação da máquina de estados finitos. Essas funções são: ler o quadrado da fita que está sendo “digitalizada”, escrever no quadrado digitalizado e mover a máquina para um quadrado adjacente (que se tornará o quadrado digitalizado no próximo ciclo de operação).”

Uma máquina de Turing equipada com estados de aceitação e rejeição, iniciada com uma determinada string escrita na fita, pode ter um de três comportamentos possíveis: ela para em um estado de aceitação; pára num estado de rejeição; ele faz um loop para sempre.

Definição 7.1

Uma linguagem é **reconhecível por Turing / recursivamente enumerável** se alguma máquina de Turing a reconhecer.

Machine Translated by Google

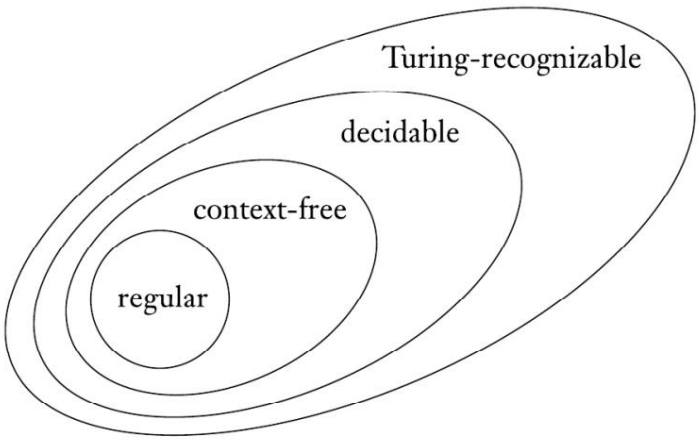


Figura 7.3: Hierarquia das classes de línguas (Sipser, 2012).

Isso significa que uma máquina de Turing reconhece uma linguagem se ela aceitar todas as cadeias da linguagem e rejeitar ou não parar nas cadeias que não estão na linguagem.

Definição 7.2

Uma linguagem é Turing-decidível/decidível/recursiva, se alguma máquina de Turing decidir isso.

Isso significa que uma máquina de Turing decide uma linguagem se aceitar todas as strings da linguagem e rejeitar todas as strings que não estejam na linguagem. A Figura 7.3 mostra a relação hierárquica entre as línguas.

Existem muitas variantes de máquinas de Turing, por exemplo, usando mais de uma fita/cabeça, usando fitas unidirecionais ou bidirecionais ilimitadas, usando alfabetos de diferentes tamanhos e com símbolos diferentes, e até mesmo usando não determinismo. No entanto, todos eles compartilham o mesmo poder computacional.

Na verdade, há uma série de fatos notáveis sobre as máquinas de Turing, alguns dos quais são os seguintes:

- Uma máquina de Turing pode ser restrita a um alfabeto de dois símbolos, sem perder nenhum de seu poder computacional. Por exemplo, suponha que uma certa máquina de Turing T use k símbolos diferentes e que n = ÿlog kÿ. Então, podemos construir outra máquina de Turing, T , substituindo cada símbolo de T por sua representação binária de n bits e em relação à fita de T como uma sequência de blocos de n bits. É claro que, para atingir a equivalência entre T e T também é necessário alterar adequadamente a máquina de estados finitos de T (o que necessariamente tem mais estados que a máquina T).

Machine Translated by Google

- Também pode ser demonstrado que máquinas de Turing que possuem fitas ilimitadas em ambas as direções têm o mesmo poder computacional que máquinas que possuem fitas ilimitadas em apenas uma direção.

- Não é uma vantagem (nem uma desvantagem) ter múltiplas fitas em uma máquina de Turing – ela sempre pode ser convertida em uma máquina de fita única equivalente.

- Provavelmente mais surpreendente é o facto de qualquer máquina de Turing ser equivalente a uma máquina de Turing com apenas dois estados internos. Claro, isso só pode ser conseguido ampliando o alfabeto dos símbolos da fita.

A ideia por trás de uma máquina de Turing foi proposta pela primeira vez por Alan Turing (1912-1954), em seu fa papel especial:

> Alan Turing, “On Computable Numbers, with an Application to the Entsghei dungsproblem”, Proceedings of the London Mathematical Society (Turing, 1936).

Quase simultaneamente, Alonzo Church (1903-1995) propôs uma noção diferente, mas equivalente, de computabilidade, usando o que ficou conhecido como cálculo ÿ:

> Alonzo Church, “Um problema insolúvel da teoria elementar dos números”, Ameri can Journal of Mathematics (Church, 1936).

Na tentativa de resolver alguns problemas relacionados aos fundamentos da matemática, ambos tiveram a necessidade de definir com rigor o conceito de “computação”. Note-se que, naquela época, um “computador” sempre foi visto como um computador humano, ou seja, alguém que realiza cálculos com lápis e papel.

Basicamente, dizemos que algo é computável se pode ser produzido, em tempo finito, por um processo equivalente à nossa noção intuitiva usual de algoritmo ou procedimento eficaz. Esta definição foi proposta por Alan Turing no seu artigo de 1936, onde também refere como equivalente a ideia de “calculabilidade efectiva” de Alonzo Church.

Novamente, de Minsky (1967):

> “A ideia de um algoritmo ou procedimento eficaz surge sempre que nos é apresentado um conjunto de instruções sobre como nos comportarmos. Isso acontece quando, ao trabalhar um problema, descobrimos que determinado procedimento, se bem executado, acabará nos dando a resposta. Uma vez feita essa descoberta, a tarefa de encontrar a solução é reduzida de uma questão de descoberta intelectual a uma mera questão de esforço; de realizar o procedimento descoberto – obedecendo às instruções especificadas.”

Esta noção de algoritmo motivou uma conjectura, conhecida como tese de Church-Turing, afirmando que algo é computável por um ser humano seguindo um algoritmo (é claro, ignorando

Machine Translated by Google

recursos, como tempo, número de folhas de papel e lápis), se e somente se for computável por uma máquina de Turing. Não há provas para esta afirmação. Contudo, já se passaram 80 anos desde a documentos de Church e Turing e nenhum contra-exemplo foi ainda encontrado!

Exemplo 7.2

Considere o seguinte exemplo de máquina de Turing, retirado do artigo original de Turing (Turing, 1936). O conjunto de instruções da máquina é:

Configuração m-	Comportamento		
config.	operações de símbolos	m-config	final.
b	Nenhum	P0, RR	c
c	Nenhum		e
e	Nenhum	P1, R	f
f	Nenhum	R	b

Se iniciada em uma fita vazia, a máquina escreve 0 e 1 alternados na fita, deixando um quadrado em branco entre cada par de dígitos.

Problema 7.1

O que acontece se, no exemplo anterior, a fita não estiver totalmente vazia?

Um dos resultados mais importantes incluídos no artigo de Turing (Turing, 1936) mostra que é possível construir máquinas de Turing universais. Uma máquina de Turing universal é uma máquina que pode simular qualquer outra máquina de Turing. Uma consequência direta desta descoberta é a noção de computador com programa armazenado. Todos, exceto os computadores mais triviais, são universais, no sentido de que pode imitar as ações de outros computadores. No entanto, é possível construir um sistema universal máquina de Turing padrão usando 4 estados e 6 símbolos. Além disso, foi recentemente sugerido (embora ainda debatido) que uma máquina de Turing universal sem estado de parada é possível usando 2 estados e 3 símbolos!

Um dos problemas mais famosos da teoria da computabilidade, que não pode ser resolvido sistematicamente por um computador, é decidir se outro programa, prog, irá parar para uma determinada entrada x. Em seguida nós mostrar que tal programa não pode existir.

Vamos considerar que tal programa, will halt(prog,x), existe. Então, considere um programa prog, tal que

```
programa(x)
    if(will_halt(x,x) == verdadeiro)
        enquanto (verdadeiro)
    outro
        saída
```

Machine Translated by Google

Agora consideremos o que acontece se prog for chamado, tendo como argumento uma descrição de si mesmo, ou seja, o que acontece ao executar prog(prog). Observe que, dentro do prog, há uma chamada para will halt(prog,prog). Se esta chamada retornar verdadeiro, ou seja, se nosso programa hipotético que decide se um programa irá parar para uma determinada entrada disser que irá parar, então prog entra em um loop infinito e não para. Pelo contrário, if will halt(prog,prog) retorna falso, ou seja, se indica que prog não irá parar quando chamado tendo ele mesmo como argumento, então ele para. Esta contradição mostra que não é possível construir um programa que, para todos os casos possíveis, verifique se algum outro programa para ou não.

7.3 Complexidade de Kolmogorov (ou entropia algorítmica)

Suponha que queiramos representar um determinado objeto por uma sequência finita de bits. Geralmente, um objeto pode ter diversas representações (ou descrições) possíveis, mas cada representação deve descrever apenas um objeto (queremos reconstruir inequivocamente o objeto a partir de sua descrição). Então, o comprimento da mais curta de todas as descrições disponíveis para o objeto pode ser usado como uma medida de sua complexidade (e, de fato, de sua aleatoriedade).

No entanto, o significado de “descrição” deve ser definido com precisão ou caso contrário poderemos deparar-nos com problemas semelhantes aos conhecidos como paradoxo de Berry, onde um certo número é definido como “o menor número inteiro positivo não definível em menos de doze palavras”. Se este número existe, acabamos de descrevê-lo em onze palavras, o que contradiz a sua definição! Por outro lado, se o número não existe, então temos de concluir que todos os números inteiros positivos podem ser descritos em menos de doze palavras!

A abordagem a seguir depende de algoritmos para medir a quantidade de informação transmitida por uma determinada string, x. Uma maneira de resolver este problema poderia envolver a construção de um (compacto) Máquina de Turing, Tx, que, quando iniciada em uma fita virgem, escreveria a string x na fita e pararia. Então, a descrição de x seria uma string descrevendo Tx, ÿTxÿ, usando alguma codificação fixa para as máquinas de Turing e, portanto, o comprimento da descrição seria |ÿTxÿ|.

Entretanto, em vez de iniciar a máquina em uma fita virgem, outra abordagem possível é usar uma string auxiliar escrita nela. Neste caso, a string x é descrita usando uma máquina de Turing M e uma entrada w que é usada por M, e a descrição de x é dada por alguma codificação ÿM, wÿ (observe que, em princípio, esta codificação precisa inclua informações que permitam separar ÿMÿ de w).

Suponha que um determinado método de especificação, f, associe no máximo um objeto, x, a uma descrição, p. Podemos pensar em f como uma função (ou programa) de descompressão, de um conjunto de descrições, P, para o conjunto de objetos, X, ou seja, f : P ÿ X. Para serem úteis, as descrições devem ser finitas. Isto implica que só é possível descrever um número contável de objetos diferentes.

Para a maioria das funções f, o comprimento da descrição do objeto x de acordo com f depende não apenas de x, mas também de f. Porém, para avaliar objetivamente a complexidade dos objetos, precisamos de funções que atribuam um valor de complexidade a x que dependa apenas de x. Além disso, se p é a descrição de x em relação a f, então x tem de ser gerado a partir de f e p através de “um procedimento eficaz” ou, por outras palavras, f tem de ser uma função computável.

Machine Translated by Google

Definição 7.3 (complexidade algorítmica)
Dada uma função computável, f, definimos a complexidade algorítmica de um objeto,16 x, de acordo com f, como

Kf (x) = min{|p| : f(p) = x}, onde x,

p ÿ {0, 1} ÿ , ou seja, corresponde ao comprimento mínimo da descrição, sobre todas as descrições que produzem x até f. Se não existe p capaz de produzir x usando f, então dizemos que Kf (x) = ÿ para tal x. Portanto, embora computável, f não precisa ser uma função total
ção.

Observe que podemos pensar em p como um programa e em f como um computador (especializado). Se Kf (x) = ÿ, então não é possível produzir x com o computador f. Além disso, observe que, sem o requisito de computabilidade, f poderia ser qualquer função parcial arbitrária de string para string, obedecendo apenas à condição de que |{x : Kf (x) = k}| ÿ 2^k onde strings de k bits são usadas como a descrição de strings x para as quais Kf (x) = k. Claro, isso nos permitiria encontrar trivial e arbitrariamente, para um certo x, af tal que Kf (x) = 0 (mapeando a string vazia para ele)!

Considere agora r métodos de especificação distintos, f1, f2, . . . , frag. É fácil construir um novo método, f, que atribui a cada complexidade xa Kf (x) que excede apenas em cerca de log r bits o mínimo de Kf1 (x), Kf2 (x), . . . , Kfr (x). A ideia é usar os primeiros log r bits de p para identificar o método fi e depois o programa apropriado para fi . Portanto, podemos dizer que

Kf (x) ÿ Kfi (x) + c, ÿi,

isto é, que o método de descrição f **minoriza** (aditivamente) todo método de descrição fi .

O desenvolvimento da teoria da informação algorítmica (também conhecida como **complexidade de Kolmogorov)** é possível pelo fato notável de que a classe de funções computáveis possui um elemento universal que é aditivamente ótimo. Sob esta restrição relativamente natural na classe de métodos de descrição (isto é, computabilidade), obtemos uma hierarquia de complexidades bem comportada e, portanto, um elemento mínimo.

Definição 7.4 (Complexidade algorítmica condicional)
Seja x, y, p ÿ N (lembre-se que isso é o mesmo que considerar x, y, p ÿ {0, 1} ÿ , de acordo com a enumeração usual de strings). Qualquer função recursiva parcial ÿ, junto com p e y, tal que ÿ(ÿy, pÿ) = x, é uma descrição de x. A complexidade Kÿ de x condicionada a y é definida por Kÿ(x|y) = min{|p| :

ÿ(ÿy, pÿ) = x},

e Kÿ(x|y) = ÿ se não existir tal p. Então, p é um programa para calcular x por ÿ, dado y.

16Sem perda de generalidade, doravante nos referiremos a estes objetos como strings e também consideraremos que são binários, ou seja, x, p ÿ {0, 1} ÿ . Além disso, como mencionado anteriormente, assumimos a ordem usual das strings, ÿ, 0, 1, 00, 01, 10, 11, 000, 001,. . . e a correspondência implícita um-para-um entre cadeias binárias e inteiros não negativos (por exemplo, o número natural “5” tem o mesmo significado que a cadeia “10”).

Machine Translated by Google

O ponto-chave desta teoria não é que o método de descrição universal fornece necessariamente a descrição mais curta em cada caso, mas em vez disso, nenhum outro método de descrição pode melhorá-lo infinitamente mais do que uma constante fixa.

Definição 7.5 (complexidade de Kolmogorov)
Vamos fixar um universal aditivamente ótimo ÿ0 (a função de referência) e definir a complexidade condicional de Kolmogorov K(x|y) = Kÿ0 (x|y). Além disso, fixe uma máquina de Turing específica (a máquina de referência), U, que calcula ÿ0. A complexidade incondicional de Kolmogorov é definida por

K(x) = K(x|ÿ).

Teorema 7.1
Para toda função computável, f, existe uma constante, c, tal que K(f(x)) ÿ K(x) + c, para todo x tal que f(x) está definido.

Teorema 7.2 (Não computabilidade de K(x))
A complexidade de Kolmogorov não é computável.

Para mostrar por que a complexidade de Kolmogorov não é uma função computável, considere o seguinte pseudocódigo, que chama uma função hipotética “K”, que calcula a complexidade de Kolmogorov da string na enésima posição da ordenação usual de strings, K(n) :

```
print_complex_string(min_k)n = 0

    while(true) if(K(n)
        > min_k) print(string(n))
            exit

        outro
            n++
```

Obviamente, para um valor de “min_k” maior que o tamanho total dos programas envolvidos, o programa imprimiria uma string com complexidade de Kolmogorov maior que o tamanho do programa utilizado para gerá-la, o que é uma contradição! Portanto, não é possível ter um programa que calcule a complexidade de Kolmogorov de strings arbitrárias.

Machine Translated by Google

Machine Translated by Google

Referências

Igreja, A. (1936). Um problema insolúvel da teoria elementar dos números. American Journal of Mathematics, 58(2), 345–363.

Cutland, Nova Jersey (1980). Computabilidade: Uma introdução à teoria da função recursiva. Cambridge Jornal universitário.

Golomb, S. (1966). Codificações de comprimento de execução. IEEE Trans. sobre Teoria da Informação, 12(3), 399–401.

Hamming, R. (1950). Códigos de detecção e correção de erros. Bell System Technical Journal, 29(2), 147–160.

Hartley, RVL (1928). Transmissão de informações. Jornal Técnico do Sistema Bell, 7(3), 535–563.

Huffman, D. (1952). Um método para a construção de códigos de redundância mínima. Continuar do IRE, 40(9), 1098–1101.

Minsky, M.L. (1967). Computação: Máquinas finitas e infinitas. Prentice-Hall, Inc., Inglês madeira Cliffs, NJ.

Nyquist, H. (1924). Certos fatores que afetam a velocidade do telégrafo. Trad. do Instituto Americano de Engenheiros Elétricos, **XLIII,** 412–422.

Shannon, CE (1948). Uma teoria matemática de comunicação. Técnico do Sistema Bell Diário, **27,** 379–423,623–656.

Shannon, CE (1951). Previsão e entropia do inglês impresso. Jornal Técnico Bell Labs, 30(1), 50–64.

Sipser, M. (2012). Introdução à teoria da computação. Cengage Learning, 3ª edição.

Storer, JA e Szymanski, TG (1982). Compressão de dados via substituição textual. Jornal do ACM, 29(4), 928–951.

Turing, A. (1936). Em números computáveis, com aplicação ao Entscheidungsproblem. Anais da Sociedade Matemática de Londres, 42(2), 230–265.

Welch, TA (1984). Uma técnica para compactação de dados de alto desempenho. Computador IEEE, 17(6), 8–19.

Ziv, J. e Lempel, A. (1977). Um algoritmo universal para compressão de dados sequencial. IEEE Trans. na Teoria da Informação, **23,** 337–343.

Ziv, J. e Lempel, A. (1978). Compressão de sequências individuais através de codificação de taxa variável. IEEE Trans. sobre Teoria da Informação, 24(5), 530–536.