2
Introdução à Teoria da Informação

Neste capítulo, apresentamos os conceitos e definições básicos de teoria da informação utilizados nesta tese. A notação e conteúdo deste capítulo são baseados no livro de Thomas M. Cover (Cov2006).

Além dos conceitos de entropia e entropia relativa, apresentamos formalmente os códigos (de prefixo) e a desigualdade de Kraft. Ao final, apresentamos a codificação Gama, que é utilizada em todos os capítulos desta tese, e alguns codificadores populares.

2.1
Conceitos Básicos

Seja X uma variável aleatória discreta com alfabeto $\mathcal{X}$ e P a distribuição de probabilidade sobre o alfabeto $\mathcal{X}$. Assim, temos uma função de probabilidade de massa $P(x) = Pr\{X = x\}$, para todo $x \in \mathcal{X}$, que satisfaz $\sum_{x\in\mathcal{X}} P(x) = 1$. *Entropia* é uma medida de incerteza de uma variável aleatória definida para qualquer distribuição de probabilidade, na qual apresentamos a seguir.

Definição 2.1 (Entropia) *A entropia $H(X)$ de uma variável aleatória discreta X é definida por:*

$$H(X) = \sum_{x\in\mathcal{X}} P(x) \log_2 \frac{1}{P(x)}. \tag{2-1}$$

A unidade de medida da entropia é *bits* e a entropia é limitada por $0 \leq H(X) \leq \log_2 |\mathcal{X}|$, onde $|\mathcal{X}|$ é o número de símbolos de $\mathcal{X}$. A entropia é utilizada para determinar uma série de outras medidas como, por exemplo, o tamanho médio da menor descrição de uma variável aleatória.

Para ilustrar o conceito da entropia, fornecemos o seguinte exemplo. Suponha que temos uma fonte com alfabeto ternário, $\mathcal{X} = \{0, 1, 2\}$, e com a distribuição de probabilidade $P = \{1/10, 4/10, 5/10\}$. Neste caso, a entropia é dada por $H(X) = 1,36$ bits por símbolo. Note que se a distribuição P for equiprovável então a entropia $H(X)$ é dado por $\log_2 3 = 1,58$ bits por símbolo.

2.2 Compressão de Dados

Compressão de dados pode ser realizada associando descrições curtas para dados mais freqüentes na fonte e descrições longas para dados menos freqüentes. Chamamos de *codificação* o processo de associar essas descrições a uma partição dos dados. No entanto, ao fazer isso, existe um limite inferior que é dado pela entropia dos dados (Definição 2.1) que se está comprimindo. A codificação que atinge esse limite é dita ótima.

2.2.1 Códigos

Seja $\mathcal{D}^*$ um conjunto de cadeias de comprimento finito de símbolos de um alfabeto D-ário. Um *código para a fonte de dados contável* $(\mathcal{X}, P)$ é um conjunto de palavras-código $C \subseteq \mathcal{D}^*$ e uma função $\mathcal{C} : \mathcal{X} \to C$, que mapeia $\mathcal{X}$ bijetivamente em C. Para todo $x \in \mathcal{X}$, o membro $\mathcal{C}(x)$ do conjunto C é chamado *palavra-código do código* $\mathcal{C}$. A palavra-código é a descrição que nos referimos informalmente no início da Seção 2.2 e, para simplificar, utilizamos apenas *palavra* para nos referirmos à palavra-código.

O conjunto C e o código $\mathcal{C}$ são ditos *unicamente decodificáveis* se e somente se toda concatenação de número finito de palavras de C é uma seqüência distinta de símbolos em $\mathcal{D}^*$, ou seja, a extensão $\mathcal{C}^* : \mathcal{X}^* \to C^*$ de $\mathcal{C}$, na qual mapeia uma seqüência de símbolos em $\mathcal{X}^*$ em uma concatenação de palavras correspondentes, é também uma função bijetiva de $\mathcal{X}^*$ em $\mathcal{D}^*$.

Um conjunto de palavras C é dito ser um *conjunto de prefixo* e o código C é dito ser um *código de prefixo* se e só se nenhuma palavra em C é o começo de outra. Um conjunto de prefixo é unicamente decodificável pois qualquer concatenação de palavras em C^* tem apenas um prefixo que é uma palavra em C.

Seja $L_{\mathcal{C}}(x)$ o comprimento da palavra $\mathcal{C}(x)$ para todo $x \in \mathcal{X}$. Simplificamos a notação fazendo $L(x) = L_{\mathcal{C}}(x)$. Finalmente, o comprimento médio das palavras de C é dado por:

$$E[L(x)] = \sum_{x \in \mathcal{X}} P(x)L(x). \tag{2-2}$$

2.2.2 Desigualdade de Kraft

Para descrever uma fonte de dados eficientemente, podemos construir códigos de prefixo de comprimento esperado mínimo. No entanto, está claro que não podemos associar palavras-código curtas para todos os símbolos e manter

a propriedade de prefixo. O conjunto de comprimentos de palavras-código para códigos de prefixo é limitado pela *Desigualdade de Kraft*, apresentada a seguir.

Teorema 2.2 (Desigualdade de Kraft) *Para qualquer código de prefixo sobre um alfabeto D-ário, os comprimentos das palavras $L(x)$ tem que satisfazer a desigualdade:*

$$\sum_{x\in\mathcal{X}} D^{-L(x)} \leq 1. \tag{2-3}$$

Reciprocamente, dado um conjunto de comprimentos de palavras que satisfaz essa desigualdade, existe um código de prefixo cujas palavras têm esses comprimentos.

Se a desigualdade de Kraft satisfaz com desigualdade estrita então o código tem ainda alguma redundância; se ela satisfaz com igualdade então o código é dito *completo*; e se ela não é satisfeita então o código não é unicamente decodificável. Perceba que a desigualdade de Kraft é uma condição necessária para um código ser de prefixo, porém não é suficiente. Finalmente, ela também é válida para fontes contáveis e infinitas.

2.2.3 Codificação Gama

A codificação Gama, proposta por Elias (Eli75), é utilizada para representar qualquer número natural $\mathbb{N} = \{1, 2, \ldots\}$ eficientemente de forma que se sabe facilmente em que ponto a representação de cada número começa e termina em uma seqüência de bits. A codificação assume uma distribuição decrescente de probabilidade P sobre $\mathbb{N}$ tal que $P(1) \geq P(2) \geq \ldots$.

Para codificar um número natural N, executamos os seguintes passos. Seja n o número de dígitos na base binária de N. Então, o prefixo da palavra de N é formada por uma cadeia binária de comprimento $n-1$ consistindo apenas de 0-bits consecutivos como prefixo e o sufixo é simplesmente a representação binária do número N. A palavra final de N é a concatenação do prefixo com o sufixo. O comprimento da palavra final de N é dado por $L(N) = 2\lfloor \log_2 N \rfloor + 1$ bits para todo $N \in \mathbb{N}$.

A simplicidade deste procedimento torna a codificação Gama bastante utilizada na prática. Inclusive, ela é utilizada ou, pelo menos, citada na maioria dos capítulos desta tese.

Finalmente, Gallager (Gal78) verificou que, de fato, o código Gama é ótimo para fontes com distribuição geométrica. Vale a pena ressaltar que o código Gama pode ser usado, em certos casos, para codificar eficientemente fontes com distribuição de potências, como será verificado no exemplo da Definição 2.3.

2.2.4
Codificação de Golomb/Rice

A codificação de Golomb, proposta em (Gol66), é utilizada para representar qualquer número natural $\mathbb{N} = \{1, 2, \ldots\}$ de forma semelhante ao código Gama. No entanto, o código de Golomb utiliza um parâmetro b que pode ser ajustado à fonte de dados de modo que os números (símbolos) sejam codificados eficientemente.

Para codificar um número natural N usando o parâmetro b, executamos os seguintes passos. Seja o quociente $q = \lfloor (N-1)/b \rfloor$ e o resto $r = N - qb - 1$. O prefixo da palavra $\mathcal{C}(N)$ é formado pelo número q na codificação unária. O sufixo é o número r na representação binária usando $\lfloor \log_2 b \rfloor$ bits para os valores menores e $\lceil \log_2 b \rceil$ bits para os valores maiores. A palavra final de N é a concatenação do prefixo com o sufixo.

Vamos ilustrar a codificação Golomb com o seguinte exemplo. Suponha que temos o parâmetro $b = 5$. Neste caso, resto r pode ser entre 0 e 4, ou seja, 00, 01, 100, 101 e 110. Perceba que usamos $\lfloor \log_2 5 \rfloor = 2$ bits quando o resto é 0 ou 1 e usamos $\lceil \log_2 5 \rceil = 3$ bits quando o resto é entre 2 e 4. Então, a palavra do número $N = 11$ usando o parâmetro $b = 5$ é formado pelo prefixo 001 e sufixo 00.

O código de Rice, proposto independentemente em (Ric79), é igual ao código de Golomb, com o parâmetro b é uma potência de 2. Desta forma, todas as operações podem ser feitas usando apenas deslocamentos (*shifts*). Devido a este fato, o código de Rice é bastante utilizado na prática.

2.2.5
Taxonomia dos Codificadores

Um codificador pode ser separado em duas partes (Ris79): modelagem e codificação. A primeira parte é responsável pela geração de uma seqüência de probabilidades enquanto que a segunda, implementada por um *codificador*, é responsável por representar estas probabilidades em bits. Um codificador projetado para maximizar a taxa de compressão atingindo o limite inferior da entropia é classificado como um *codificador de entropia*. Os codificadores de entropia mais populares são Huffman (Huf52) e o codificador aritmético (Ris79).

Em aplicações modernas, no entanto, atingir o limite da entropia não é necessariamente o objetivo principal. Alguns fatores, como velocidade de codificação, limitações do hardware e de memória principal assim como otimização do cache podem ser mais importantes. Codificadores que priorizam alguns destes fatores ao invés de atingir o limite inferior da entropia são classi-

ficados como *não-codificadores de entropia* (*non-entropy coders*) (Mof97). Em geral, eles usam a mesma representação independente da distribuição de probabilidade. Exemplos de não-codificadores de entropia são: o codificador Gama (Eli75), o codificador de Golomb (Gol66) e o codificador de Rice (Ric79).

Codificadores também podem ser classificados como *estáticos* ou *adaptativos*. Outra maneira de classificar codificadores é de acordo com o tipo da fonte que eles podem modelar: binário ou multi-alfabeto; sem memória ou Markov (com memória); estacionário ou não-estacionário.

2.3 Entropia Relativa

Por fim, vale a pena mencionar o conceito de entropia relativa, pois pode ser utilizado para medir a ineficiência de códigos. *Entropia relativa* é a medida de distância entre duas distribuições de probabilidade, na qual apresentamos a seguir.

Definição 2.3 (Entropia Relativa) *A entropia relativa $D(P||Q)$ ou distância de Kullback Leibler entre duas funções de probabilidade de massa $P(x)$ e $Q(x)$ é definida por:*

$$D(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log_2 \frac{P(x)}{Q(x)}. \tag{2-4}$$

A entropia relativa $D(P||Q)$ também pode ser vista como a medida de ineficiência de assumir que a distribuição é Q quando a distribuição real é P. Por exemplo, se soubéssemos que a distribuição real de uma variável aleatória P então poderíamos construir um código com o comprimento médio das palavras dado por $H(P)$. No entanto, se usássemos o código para a distribuição Q então precisaríamos de $H(P) + D(P||Q)$ bits por símbolo, na média, para descrever a variável aleatória.

Para ilustrar a entropia relativa, fazemos a seguinte verificação. Em certos casos, é possível utilizar um código ótimo para uma fonte com distribuição geométrica $G(x) \approx {c_1}^{-x}$, onde c_1 é uma constante positiva, para codificar eficientemente uma fonte com distribuição de potências $P(x) \approx x^{-c_2}$, onde c_2 é uma constante positiva. A distribuição de potências aparece com bastante freqüência nos grafos web. A ineficiência I é calculada pela entropia relativa como segue:

$$I = \sum_x P(x) \log_2 \frac{P(x)}{G(x)} \approx \sum_x x^{-c_2} \log_2 \frac{x^{-c_2}}{{c_1}^{-x}}$$
$$= \log_2 c_1 \sum_x x^{1-c_2} - c_2 \sum_x \frac{\log_2 x}{x^{c_2}}.$$

Para $c_2 > 2$, a ineficiência I converge. Para $c_2 > 2.1$, I converge rapidamente. Por exemplo, para $x = 1, \ldots, \infty$ e com constantes $c_1 = 2$ e $c_2 = 2.4$, a ineficiência I é cerca de 1.5 bits/símbolo.