Compressão de Dados e de Imagens

Rogério Theodoro de Brito
Orientador: José Augusto R. Soares

1 Introdução

A Compressão de Dados e, mais especificamente, a Compressão de Imagens vêm adquirindo grande importância, seja com a necessidade de transferência rápida de dados por meio de redes, seja para o simples propósito de armazenamento.

Normalmente, as mensagens geradas possuem um alto grau de redundância, permitindo, assim, gerarmos mensagens mais compactas.

Intuitivamente, na Compressão de Dados, estamos preocupados em estudar se há formas de transformar uma mensagem (um arquivo, por exemplo) de comprimento n em uma outra mensagem de comprimento $m < n$, de forma que esta última mensagem possua o mesmo conteúdo que a mensagem anterior. Por exemplo, podemos ter uma mensagem escrita em português, usando letras latinas, e tranformá-la em uma outra, com o mesmo conteúdo, usando o código morse.

De uma forma um pouco mais precisa, podemos estabelecer: dados dois alfabetos $\Sigma_1 = \{a_1, \ldots, a_I\}$ e $\Sigma_2 = \{b_1, \ldots, b_J\}$, consideramos $\Sigma_i^* = \{\emptyset, b_1, \ldots, b_J, b_1b_2, \ldots\}$ o conjunto de todas as possíveis seqüências de $\Sigma_i, i = 1, 2$. Nestas condições, queremos encontrar uma função $f: \Sigma_1^* \to \Sigma_2^*$ que seja inversível e tal que se nossa mensagem é denotada por $x \in \Sigma_1^*$, tenhamos, em média, $|f(x)| < |x|$, onde $|x|$ denota o comprimento (número de símbolos) de x. O cálculo do tamanho médio é efetuado levando-se em consideração uma distribuição de probabilidades dos símbolos de Σ_1. A determinação de uma f com as características acima pode ser muito difícil ou, até mesmo, impossível. Um critério para decidirmos se existe tal f é baseado no conceito de *entropia* [Sha48], que é matematicamente definida como $H = -\sum_{i=1}^{I} p_i \log_{|\Sigma_1|} p_i$, onde p_i é a probabilidade do símbolo a_i aparecer em uma mensagem. A entropia H determina a quantidade média de *informação* da mensagem sobre o alfabeto Σ_1.

A Compressão de Imagens também é baseada na redundância contida em imagens. A redundância neste caso é mais fácil de se compreender do que em textos: dois pixels ("pontos") adjacentes numa imagem tendem a compartilhar as mesmas propriedades em relação à cor, luminosidade etc. Enquanto na compressão de dados a entrada do problema é uma mensagem unidimensional, na compressão de imagens é importante se explorar o conceito de vizinhança bi-dimensional. Os principais métodos de compressão de imagens são o DCT (JPEG), o método por Fractais e a compressão por meio de Wavelets, além da utilização de algoritmos de compressão de dados para imagens, como é o exemplo do padrão GIF (Graphics Interchange Format).

2 Algoritmos para Compressão de Dados

Os principais algoritmos de Compressão de Dados são o de Codificação de Huffman [Huf52], o de Codificação Aritmética [RJ79] e os de Codificação por Dicionário [ZL77, ZL78].

- O algoritmo de Huffman funciona baseado no fato de que, se atribuirmos um pequeno código para símbolos mais prováveis e códigos maiores para símbolos menos prováveis, na média teremos

mensagens mais curtas. Por exemplo, para o alfabeto $\Sigma_1 = \{a, e, i, o, u, !\}$[1], podemos considerar os seguintes códigos associados a estes símbolos: $1, 01, 001, 0001, 00001, 00000$. Neste caso, o código gerado para a mensagem *eoiia!* é 010001001001100000, que, em computadores de 8 bits por byte seria representado em 3 bytes, sendo que a mensagem anterior seria representada em 6 bytes. Conseguimos, então, uma redução de 50% no tamanho da mensagem original. Entretanto, pode-se provar que o código gerado por este algoritmo é ótimo apenas no caso em que as probabilidades dos símbolos de Σ_1 são probabilidades com expoentes negativos e inteiros de $|\Sigma_2|$.

- Como uma maneira natural de superar esta dificuldade do código de Huffman, surge a Codificação Aritmética que pode ser descrita da seguinte forma. Imaginamos o intervalo $[0; 1)$ dividido em subintervalos de amplitudes iguais às probabilidades dos símbolos do alfabeto Σ_1. Após observarmos o próximo símbolo, o nosso intervalo corrente é reduzido. O novo intervalo é definido pela projeção do intervalo inicial no intervalo corrente, de onde o subintervalo correspondente ao próximo símbolo é escolhido. Ao final do processo, teremos um intervalo e qualquer número deste intervalo pode ser utilizado para representar nossa codificação aritmética da mensagem.

- A compressão baseada em dicionários (ou substituição textual) dá-se levando-se em conta subseqüências que ocorrem freqüentemente. Mantendo-se um dicionário de subseqüências da mensagem dada, verificamos se a nossa entrada já está presente no dicionário. Caso esteja, como saída geramos apenas um ponteiro indicando seu índice. Caso contrário, fazemos a inclusão da nova entrada e colocamo-la como mensagem de saída. Este algoritmo de substituição foi desenvolvido por Ziv e Lempel [ZL77] e possui diversas variações (são 12 as principais).

3 Desenvolvimento do Trabalho

Atualmente, estamos estudando alguns conceitos fundamentais de *Teoria da Informação*, dentre os quais o principal é o de entropia. Já estudamos uma implementação do algoritmo da codificação de Huffman e uma implementação da codificação aritmética, que utiliza multiplicações e divisões. Estudaremos uma segunda implementação que é mais adequada à implementação em hardware. Nosso passo seguinte será estudar algumas variações dos algoritmos baseados em dicionários.

Referências

[Huf52] D. A. Huffman, *A method for the construction of minimum redundancy codes*, Proc. IRE, vol. 40, 1952, p. 1098.

[RJ79] J. Rissanen and G.G. Langdon Jr., *Arithmetic coding*, IBM Journal of Research and Development **23** (1979), no. 2, 149–162.

[Sha48] C.E. Shannon, *A mathematical theory of communication*, Bell Syst. Tech. J. **27** (1948), 398–403.

[ZL77] J. Ziv and A. Lempel, *A universal algorithm for sequential data compression*, IEEE Trans. on Inf. Theory **IT-23** (1977), no. 3, 337–343.

[ZL78] J. Ziv and A. Lempel, *Compression of individual sequences via variable-rate coding*, IEEE Trans. on Inf. Theory **IT-24** (1978), no. 5, 530–536.

[1] Para nós, o símbolo *!* será utilizado para indicar o final de arquivo.