UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Cristina Elisabeth Ricken

# SISTEMA ADAPTATIVO NEURAL PARA COMPRESSÃO SEQÜENCIAL E CLASSIFICAÇÃO DE TEXTOS

Dissertação de mestrado

Florianópolis
2001

UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Cristina Elisabeth Ricken

# SISTEMA ADAPTATIVO NEURAL PARA COMPRESSÃO SEQÜENCIAL E CLASSIFICAÇÃO DE TEXTOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina como requisito parcial para obtenção do título de Mestre em Engenharia de Produção

**Orientador**: Raul Sidnei Wazlawick, Dr.

Florianópolis
2001

*Aos meus pais e meus irmãos*

*Aos meus avós (in memoriam)*

*Ao Erich*

# RESUMO

RICKEN, Cristina E.. **Sistema Adaptativo Neural para Compressão Seqüencial e Classificação de Textos**. 2001. Dissertação (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2001.

O contexto de crescente disponibilidade de informação textual em formato digital evidencia a importância de mecanismos de compactação de dados sem perda e de classificação automática de textos para a gestão de informações. Esta dissertação apresenta um novo sistema para compressão de dados sem perda, utilizando uma rede neural artificial baseada na Teoria da Ressonância Adaptativa (*Adaptive Resonance Theory* - ART) para modelagem preditiva de seqüências discretas. Uma rede fuzzy ARTMAP modificada gera modelos para estimações probabilísticas e é integrada a um codificador aritmético. O sistema adaptativo neural de compressão desenvolvido realiza o aprendizado incremental dos padrões observados nas seqüências apresentadas, executando a compactação seqüencial e a descompactação exata de seqüências discretas sem conhecimento prévio da estrutura estatística da fonte das mensagens. O sistema foi testado diante de uma base de dados pública para *benchmark* (formada por arquivos binários e de texto) para avaliação de seu desempenho em relação a compactadores de texto tradicionais, atingindo taxas de compressão melhores que o *software* gzip. Além da viabilidade de utilização da rede neural proposta no estágio de modelagem do processo de compressão sem perda, a capacidade do sistema desenvolvido foi testada em duas tarefas de classificação automática de textos: identificação de idiomas e classificação por gênero de textos. A classificação por gênero de textos, por meio da abordagem do presente trabalho, visa designar textos a classes de publicações digitais, conforme a similaridade em relação ao modelo que representa cada classe. A técnica neural de compressão foi aplicada a estas tarefas, medindo a entropia cruzada entre cada exemplar de teste e um modelo gerado. A similaridade entre uma seqüência de texto e cada uma das classes é determinada autonomamente pelo sistema, sem a pré-definição de atributos ou conhecimento analítico sobre o texto ou um idioma específico. Na tarefa de identificação de idiomas todos os itens de teste foram perfeitamente reconhecidos, e na tarefa de classificação por gênero de textos, o sistema classificou corretamente 95,83% dos exemplares de teste apresentados. A compressão sem perda de seqüências discretas propicia um ambiente para estudo do comportamento da rede neural proposta em tarefas que requerem adaptação e estimação probabilística *on-line*. Além da compressão de dados sem perda, o sistema neural desenvolvido pode ser aplicado a outras áreas que requerem aprendizado de padrões, modelagem preditiva e classificação de seqüências, como descoberta de conhecimento em bases de dados para gestão de informações e inteligência de negócios.

**PALAVRAS-CHAVE**: Redes Neurais Artificiais; Teoria da Ressonância Adaptativa; Compressão sem perda; Teoria da Informação; Classificação Automática de Textos.

# ABSTRACT

RICKEN, Cristina E.. **Adaptive Neural System for Sequential Compression and Text Classification**. 2001. Dissertation (Mestrado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2001.

The context of increasing availability of textual information in digital form highlights the importance of mechanisms for lossless data compression and automatic text classification for information management. This dissertation presents a novel system for lossless data compression, using an artificial neural network based on the Adaptive Resonance Theory (ART) for predictive modeling of discrete sequences. A modified fuzzy ARTMAP neural network generates models for probability estimations and is integrated to an arithmetic coder. The developed neural adaptive compression system performs incremental learning of the patterns observed in the sequences presented, performing sequential compression and exact decompression of discrete sequences without previous knowledge of the statistical structure of the source messages. The system was tested on a public benchmark database (consisting of binary files and text) for performance evaluation and comparison to traditional text compressors, achieving better compression rates than the gzip software. Besides the feasibility of using the proposed neural network in the modeling stage of the lossless compression process, the ability of the developed system was tested in two tasks of automatic text classification: identification of languages and text genre classification. The text genre classification, following the approach of the present work, aims to assign texts to classes of digital publications, based on the similarity with the model that represents each class. The neural compression technique was applied to these tasks, measuring the cross-entropy between each test exemplar and a generated model. The similarity between a text sequence and each class is determined autonomously by the system without the predefinition of attributes or analytical knowledge about the text or a specific language. In the language identification task all test items were perfectly recognized and for the text genre classification task, the system correctly classified 95,83% of the test exemplars presented. The lossless compression of discrete sequence provides an environment to study the behavior of the proposed neural network in tasks that require on-line adaptation and probabilistic estimation. In addition to lossless data compression the developed neural system can be applied to other areas that require learning of patterns, predictive modeling and classification of sequences, such as knowledge discovery in databases for information management and business intelligence.

**KEYWORDS**: Artificial Neural Networks; Adaptive Resonance Theory; Lossless Compression; Information Theory; Automatic Text Classification.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE QUADROS

# LISTA DE TABELAS

# LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

ART Adaptive Resonance Theory (Teoria da Ressonância Adaptativa)

ASCII American Standard Code for Information Interchange

BP Backpropagation

BWT Burrows-Wheeler Transform

DMC Dynamic Markov Compression (Compressão de Markov Dinâmica)

gzip GNU zip (Software Livre de compressão sem perda)

HTML HyperText Markup Language (Linguagem de anotação para hipertexto)

HTTP HyperText Transfer Protocol (Protocolo de transferência para hipertexto)

JPEG Joint Photographic Expert Group

LZ Lempel - Ziv

LZW Lempel - Ziv - Welch

LVQ Learning Vector Quantization

MLP Multilayer Perceptron

MPEG Motion Picture Expert Group

CEP Controle Estatístico de Processos

PDP Parallel Distributed Processing

PPM Prediction by Partial Matching

RBF Radial-Basis Function

RNA Rede Neural Artificial

SMTP Simple Mail Transfer Protocol

SOM Self-Organizing Maps (Mapas auto-organizáveis)

VLSI Very Large-Scale Integration

VQ Vector Quantization (Quantização de vetor)

WTA Winner-Take-All (Regra do “vencedor leva tudo”)

XML eXtensible Markup Language (Linguagem de anotação extensível)

# 1 INTRODUÇÃO

## 1.1 Apresentação

O avanço de tecnologias para aquisição, transmissão e armazenamento de dados vem sendo acompanhado pelo crescimento acelerado do volume de dados e de informações disponibilizados em formato digital por meio da internet, de intranets corporativas e de grandes bases de dados. Especificamente, documentos eletrônicos e campos de *texto* em bancos de dados formam uma grande parte do conteúdo informacional manuseado por empresas, organizações de governo, de ensino e de pesquisa, repositórios de patentes industriais e de propriedade intelectual, assim como por veículos de publicação *on-line.* Este contexto evidencia a importância de dois recursos essenciais para o processamento eficiente dos dados em formato digital: *a compactação sem perda*, que permite a perfeita restauração dos dados durante a descompactação e reverte na diminuição dos custos associados à sua comunicação, e a classificação automática de seqüências discretas, que proporciona um aumento na eficiência e efetividade dos processos de gestão de informações.

Visando a proposta de uma alternativa para técnicas de compactação e para processos de classificação existentes, este trabalho apresenta um novo sistema adaptativo para compactação *sem perda* baseado em redes neurais artificiais. No contexto da área de concentração em Inteligência Aplicada, o presente trabalho explora a capacidade de modelagem e estimação preditiva de uma rede neural artificial por meio do sistema proposto e sua aplicação à compressão de seqüências discretas e à classificação automática de *textos.*

Mecanismos de compactação realizam a conversão dos dados para uma representação de menor tamanho que a original, viabilizando um melhor aproveitamento da largura de banda para transmissão e menor espaço para armazenamento dos dados. A compactação *sem perda*, requerida para dados que precisam ser descompactados com absoluta fidelidade em relação à versão original, é imprescindível para seqüências discretas como as que formam textos em linguagem natural, planilhas de dados, programas executáveis e códigos fonte.

A aplicação de técnicas computacionais de classificação (ou categorização) automática de seqüências de texto propicia um ganho de produtividade aos processos de gerenciamento de informações, substituindo os laboriosos procedimentos manuais. A classificação automática de seqüências textuais abrange aplicações como: a indexação e a busca de documentos em bases de dados, a categorização de documentos no ambiente organizacional, a filtragem de

*spam*, o roteamento de textos relevantes ao destinatário, a categorização de páginas da WEB. Para abordar a classificação de textos têm sido adotadas diversas técnicas de aprendizado de máquina, dentre as quais, Redes Neurais Artificiais (RNAs). Com amplo espectro de aplicações em diversas áreas (ver Capítulo 3 do presente trabalho), RNAs representam estruturas de processamento de informações em que modelos matemáticos, juntamente com alguns princípios gerais que procuram emular funções neurais biológicas, podem ser desenvolvidos, aplicados e analisados.

As áreas de compressão de *texto* e de classificação automática de textos apresentam ambientes de aplicação que têm sido tratados por redes neurais artificiais através de abordagens distintas. Em síntese, conforme a abordagem tradicionalmente adotada por redes neurais para a construção de classificadores de texto, estas lidam com representações vetoriais dos documentos, em que os atributos que descrevem cada qual são definidos por meio dos procedimentos usuais de pré-processamento aplicados a esta tarefa. Por meio de um treinamento prévio, utilizando um conjunto formado por exemplos rotulados, as redes aprendem a discriminar entre as classes de textos às quais deverão ser designados os exemplares de teste apresentados posteriormente. A aplicação de redes neurais para modelagem probabilística em compressão de *texto* conta com poucos exemplos documentados até o momento. Neste caso, as redes neurais trabalham com subseqüências de entrada para as quais devem ser previstos os símbolos sucessores, sendo que a forma de treinamento está relacionada ao tipo de modelagem executada para compactação.

O presente trabalho aborda a compressão e a classificação de seqüências discretas por meio de um sistema que integra uma rede neural artificial originária da *Teoria da Ressonância Adaptativa* para a modelagem das seqüências de entrada e um codificador estatístico para sua codificação. Essencialmente, a extração das regularidades observadas em uma seqüência de dados é tratada como um aprendizado de padrões ao longo da construção de um *modelo* utilizado para a realização de predições seqüenciais no processo de compactação. A função da rede neural é a geração de um modelo preditor para os dados a serem codificados, com reversão do processo para sua descompactação exata. Por meio de um sistema modular e de conceitos da Teoria da Informação, o método é aplicado à classificação de textos, diferenciando-se das abordagens tradicionalmente adotadas por redes neurais e por outras técnicas de aprendizado de máquina para esta tarefa. Ao extrair automaticamente as características de cada seqüência processada, o sistema neural concebido pode atuar, tanto como compactador de *texto* de propósito geral, como também ser utilizado para geração de modelos de linguagem aplicados para a classificação de seqüências discretas.

Este capítulo segue com a apresentação da motivação do trabalho e dos temas abordados, expõe as questões da pesquisa realizada, destacando os objetivos desta dissertação, assim como a justificativa da abordagem desenvolvida e a metodologia seguida

## 1.2 Motivação e Contextualização do Trabalho

Um dos focos de pesquisa em inteligência aplicada é o desenvolvimento de sistemas adaptativos que possam incorporar conhecimento sobre um determinado domínio, automaticamente, através de aprendizado. Neste sentido, o "conhecimento" adquirido é representado por meio de um *modelo* utilizado para fornecer respostas apropriadas ao ambiente externo (HAYKIN, 1999). Redes neurais artificiais têm sido utilizadas como técnicas para aquisição de conhecimento por meio de exemplos extraídos do ambiente de dados, procurando construir, no processo, um modelo interno deste. Inspiradas em sistemas neurais biológicos e constituídas por muitos elementos interligados de cômputo local, denominados de neurônios, redes neurais artificiais armazenam em sua estrutura o "conhecimento"[1] adquirido ao longo do aprendizado. Assim, com base em um *modelo* que represente as características aprendidas do ambiente de dados em questão, uma rede neural pode responder adequadamente a estímulos futuros e, inclusive, a dados inéditos do espaço de dados do problema abordado.

Diante da variedade de redes neurais artificiais existentes (HAYKIN, 1999; SKAPURA, 1996; FAUSETT, 1996), biologicamente plausíveis ou não, uma das questões de interesse é o desenvolvimento de sistemas com capacidade de aprendizado incremental e *on-line* no ambiente de aplicação. A adaptabilidade de uma rede neural artificial pode ser verificada diante de ambientes mutáveis ou cuja estrutura estatística é desconhecida *a priori*, em situações nas quais o aprendizado deve ser realizado ao longo do processo de execução. Nestas condições, é necessário que a rede aprenda novos padrões, *incrementalmente*, à medida que são apresentados, sem destruir o conhecimento útil já adquirido. A atualização da rede deve ser realizada de modo *on-line* para capturar as características do ambiente em que opera sem depender do acesso a todo o conjunto de dados anteriormente visto.

No presente trabalho, o perfil adaptativo de uma rede neural artificial será explorado por meio de sua exposição a um ambiente de entradas seqüenciais para compactação *sem perda* de arquivos binários e de texto, tarefa denominada, genericamente, de compressão de *texto*.

[1] Este "conhecimento" é codificado na rede (em seus pesos conectivos) ao longo do processo de aprendizado realizado através de operações algorítmicas, sendo utilizado para responder aos casos/estímulos apresentados, e podendo ser extraído da rede (na forma de regras, por exemplo) por meio de técnicas apropriadas.

Para adequar-se a este ambiente, a rede neural deve apresentar características que também são compatíveis com outras aplicações que demandam a realização de estimações probabilísticas e a adaptação *on-line*. A construção de um modelo preditor na aquisição de conhecimento de um ambiente de dados é requisito para a abordagem de inúmeros problemas que envolvem a predição de eventos futuros com base em seqüências passadas, o reconhecimento e a classificação de padrões.

A aplicação de redes neurais artificiais à compressão de *texto* parte da idéia de que estas possam ser utilizadas na etapa de *modelagem* das seqüências de entrada. Como será visto ao longo deste trabalho, existem diferentes tipos de modelagem, cada qual com características funcionais próprias e um grau de adequação que pode variar, conforme a aplicação.

### 1.2.1 Compressão sem Perda - A Compressão de *Texto*

O exemplo típico para a compactação *sem perda* é a compressão de *texto* para seqüências emitidas por uma *fonte* discreta de informação. Aplicada sobre arquivos binários e de texto de grandes bases de dados ou computadores pessoais, a compressão de *texto* é, também, vastamente aplicável no tráfego de dados via internet, em que a maioria dos protocolos importantes é baseada em *texto*, tais como os protocolos HTTP e HTML, o protocolo de *e-mail* SMTP, e o XML, que pretende ser a “língua universal" para o intercâmbio de dados. Além de propiciar a diminuição dos custos de transmissão e armazenamento de dados, um algoritmo de compressão de *texto* apresenta outras funcionalidades. Sua utilidade estende-se à área de segurança de dados, na medida em que a remoção das redundâncias nos dados de um arquivo, por meio de sua compactação, antes deste ser criptografado, reduz imensamente as oportunidades de *cripto-análise*. No presente trabalho, além da compressão de arquivos binários e de texto, o compactador neural desenvolvido explora a funcionalidade de classificação de seqüências discretas, sendo testado em classificação automática de textos.

Diferentes abordagens de compressão sem perdas podem ser adotadas conforme os tipos de aplicação. Técnicas que trabalham com algum conhecimento prévio das características estatísticas das seqüências de dados a serem processados possibilitam o alcance de versões bastante reduzidas destas, mas tem sua efetividade de compressão restrita a estas. Para compactar dados originários de *fontes* de informação com estruturas estatísticas distintas, com um espectro maior de aplicações, devem ser adotadas técnicas que processam as seqüências sem assumir qualquer conhecimento prévio sobre estas. Algoritmos de compressão de *texto* podem comprimir de modo efetivo grande parte dos arquivos binários e de *texto* encontrados em sistemas computacionais, sendo aplicáveis a dados que abrangem códigos fonte e

programas executáveis, textos em linguagem natural, tabelas Excel e, mesmo sem ser talhados para operar com seqüências de DNA ou com imagens de *bitmap*, estes podem realizar a compactação destes tipos de seqüência de dados de modo eficaz. É preciso ressaltar que não existe um compactador de *texto* que seja capaz de comprimir sem perda todas as seqüências possíveis, de modo que a compressão de determinados arquivos sempre implicará na expansão de outros. Assim, um compactador de *texto* de propósito geral procura obter um compromisso entre a capacidade de atingir taxas de compressão satisfatórias e uma maior flexibilidade de operação diante de uma grande diversidade de estruturas seqüenciais.

Os métodos de compressão de *texto* mais conhecidos, atualmente, são baseados nos algoritmos LZ (Lempel-Ziv), PPM (*Prediction by Partial Matching*) e BWT (*Burrows-Wheeler Transform*). Dentre os compactadores de texto populares, destaca-se, por sua eficiência e velocidade, o programa *gzip*, baseado no algoritmo LZ77 (ZIV e LEMPEL, 1977). Variantes do método PPM (WITTEN, MOFFAT, BELL, 1999) proporcionam, até o momento, as melhores taxas em compressão de *texto*, mas são menos velozes que o *gzip* e demandam mais recursos de memória para operação. O estágio atual destas técnicas é resultado dos sucessivos aprimoramentos realizados sobre os métodos e algoritmos que as originaram e contrasta com a relativamente recente incursão de redes neurais artificiais na área de compressão de *texto*.

Trabalhos anteriores de redes neurais em compressão de *texto* (SCHMIDHUBER e HEIL, 1996; NATSEV, 1997; LONG, NATSEV e VITTER, 1999) utilizam redes *Feedforward* com múltiplas camadas, treinadas previamente por meio do algoritmo *Backpropagation*. As referidas abordagens alcançaram resultados promissores em termos da razão média de compactação, diante de uma base de dados formada por seqüências de texto em linguagem natural, como será visto no Capítulo 3 deste trabalho.

### 1.2.2 Classificação Automática de Textos

A classificação (ou categorização) automática de textos compreende a designação de seqüências textuais a classes pré-definidas, e tem sido abordada por diversas técnicas, dentre as quais: árvores de decisão, classificadores de *Bayes*, SVM (*Support Vector Machines*), *k-nn* (*k-nearest neighbor*) e *redes neurais artificiais*. A classificação de textos pode ser utilizada em tarefas específicas como: a identificação de idiomas, a atribuição e verificação de autoria, a categorização por tópicos e por *gênero* de texto. Destas tarefas, a designação de textos a classes definidas por *tópicos* consiste em uma das mais exploradas formas para organização de informação disponível em formato digital. No entanto, a classificação por tópicos pode não

ser suficiente para a filtragem eficaz ou a determinação da relevância de documentos no processo de recuperação de informação (*Information Retrieval*). Como enfatizam Kessler, Nunberg e Schutze (1997), à medida que as bases de dados disponíveis se tornam maiores e mais heterogêneas, a identificação do *gênero* ou *estilo* de um texto torna-se cada vez mais importante como complemento à classificação por tópicos.

Segundo Rauber e Müller-Kögler (2001), a análise do *gênero* de texto procura identificar, dentre um conjunto de itens, grupos de documentos que compartilham de uma forma comum de transmissão, propósito e propriedades de discurso. Neste sentido, textos científicos especializados em um determinado tópico, por exemplo, apresentam uma terminologia própria e uma expressão de linguagem distinta da utilizada em textos jornalísticos de divulgação científica do mesmo assunto.

Diante da massa de documentos disseminados pela internet, a classificação do gênero em que está expresso um texto pode propiciar um refinamento do processo de busca, recuperando e agrupando os documentos que possam ter maior relevância para o usuário. Por exemplo, o gênero de um texto que aborda "Teoria da Informação" sob uma perspectiva introdutória ou interdisciplinar atende a requisitos de interesses diferentes de outro que discorre sobre este tema, mas com um enfoque especializado e maior desenvolvimento matemático.

O presente trabalho concentra-se na aplicação de um novo método neural de compressão para classificação de seqüências textuais a partir das características estatísticas observadas, procurando identificar a *fonte* de informação que possa ter originado o texto. Como uma discussão sobre análise de estilo e gênero de textos foge ao escopo deste trabalho, podem ser consultadas fontes como (KARLGREN, 2000) para um maior aprofundamento na área.

Dentre as técnicas adotadas para identificação de gênero ou estilo de texto, podem ser citados métodos baseados em árvores de decisão (ARGAMON, KOPPEL e AVNERI, 1998), análise discriminante (KARLGREN, 2000), além de redes neurais artificiais *Feedforward* com *Backpropagation* (KESSLER, NUNBERG e SCHÜTZE, 1997) e, recentemente, mapas *SOM* (RAUBER, MÜLLER-KÖGLER, 2001). A abordagem mais comum da tarefa consiste na utilização de vetores de atributos (definidos conforme o objetivo da tarefa e extraídos do texto por meio de técnicas apropriadas) que formam a entrada para o classificador em questão. Uma modelagem estatística de linguagem, baseada no método de compressão de texto PPM foi aplicada, anteriormente, à identificação de idiomas e de autoria (TEAHAN, 1998), tendo sido utilizado para classificação de textos por tópicos em (FRANK, CHUI e WITTEN, 2000).

### 1.2.3 Uma Rede Neural para Compressão e Classificação de *Textos*

Em um ambiente de compressão de *texto*, métodos que executam uma modelagem adaptativa devem ajustar-se dinamicamente às estatísticas da seqüência em compressão, capturando regularidades que são utilizadas para gerar uma representação mais curta da seqüência original, por meio de um mecanismo reversível que permita sua perfeita reconstrução durante a descompactação. Portanto, a rede neural aplicada à modelagem adaptativa deve aprender as características dos dados apresentados seqüencialmente, enquanto constrói o *modelo* por meio do qual compacta a seqüência de entradas.

A compressão *sem perda* com modelagem adaptativa oferece um ambiente de aplicação e de estudo do comportamento de uma rede neural artificial ao atuar como estimador probabilístico *on-line*, impondo condições nas quais a rede deve aprender e operar sem conhecimento prévio das estatísticas da seqüência processada.

A operação de uma rede neural artificial em um ambiente em que é necessária a adaptação autônoma em relação aos dados apresentados seqüencialmente remete ao *dilema da estabilidade-plasticidade* (CARPENTER e GROSSBERG, 1987a; 1988). Em termos gerais, este se refere ao problema da manutenção da estabilidade do conhecimento já representado em uma rede neural artificial, juntamente com a capacidade adaptativa (ou *plástica*) de aprendizado de novos dados de entrada. Neste processo, é necessário preservar o conhecimento relevante adquirido, sem que este seja completamente erodido por meio do aprendizado de novos padrões. Uma solução para este dilema foi proposta pela Teoria da Ressonância Adaptativa (*Adaptive Resonance Theory - ART*) (GROSSBERG, 1976; CARPENTER e GROSSBERG, 1987a) que deu origem a uma família de redes neurais artificiais. Essencialmente, uma rede baseada em *ART* pode aprender os dados de entrada de modo incremental, conforme sua similaridade com protótipos já formados internamente, ou acomodar conhecimento novo por meio da construção de novas categorias para representá-lo.

As características construtivas e a capacidade de aprendizado incremental de uma rede da classe ARTMAP, originária da Teoria da Ressonância Adaptativa, são aqui exploradas por meio de sua aplicação na etapa de modelagem em compressão de *texto*. Especificamente, as modificações implementadas sobre uma rede fuzzy ARTMAP possibilitam que esta atue de modo efetivo como *modelador neural preditivo* para realização de estimações probabilísticas *on-line*. Por meio da rede neural artificial proposta é visada a construção incremental de um *modelo* do ambiente de entradas que permita efetuar predições seqüenciais durante o processo de compactação. Sem conhecimento *a priori* do modelo estatístico da *fonte* emissora dos

dados sobre os quais opera, a rede neural aprende, dinamicamente, um *modelo* preditor adaptativo com base no qual estima a distribuição probabilística para os eventos que podem suceder as subseqüências apresentadas. As estimativas probabilísticas geradas, convertidas por um codificador em um fluxo de bits de saída, definem a compressão da mensagem. Neste contexto, a modelagem para dados originários de diferentes tipos de *fonte de mensagem* e a medição da *quantidade de informação* da mensagem em relação ao modelo construído pela rede neural serão aplicadas à *compressão* e à *classificação de textos*.

## 1.3 Definição do Problema e Questões de Pesquisa

Os aspectos que definem o problema de aplicação de uma *Rede Neural Artificial* (RNA) à *Compressão sem perda* de arquivos binários e de texto, assim como à *Classificação automática de textos* por intermédio de compressão, são aqui sistematizados e associados às questões tratadas pelo presente trabalho.

- A compressão de texto *on-line* com *modelagem adaptativa* impõe condições nas quais um algoritmo (baseado em uma RNA) precisa aprender incrementalmente, a distribuição probabilística dos dados, enquanto realiza as estimações preditivas, a cada passo de compressão, em *uma única passada* pela seqüência em processamento. Deste modo, a RNA deve adaptar-se a arquivos com estruturas estatísticas previamente desconhecidas, revertendo o processo para a descompactação exata. Neste contexto, como definir uma arquitetura neural apropriada que seja capaz de comprimir arquivos com distribuições probabilísticas distintas?

- A adaptação às seqüências em processamento aponta para a utilização de uma RNA com capacidade de aprendizado incremental e *on-line*. Par*a* contornar a necessidade de pré-fixação da estrutura, propõe-se adotar uma RNA que acrescente novas unidades escondidas, ao longo aprendizado. Neste trabalho, considera-se, como candidata, uma RNA da classe ARTMAP (originária de *ART*). No entanto, como adequar esta rede para uma atuação eficiente como um modelo gerador de estimações probabilísticas no modo *on-line*, diante de padrões apresentados seqüencialmente?

- Durante a compressão de *texto*, as distribuições probabilísticas estimadas por um *modelador* deverão ser utilizadas por um codificador, de modo que o processo seja reversível e garanta a restauração exata da seqüência compactada à sua versão original. Assim, como integrar uma RNA modeladora a um codificador, de modo que esta reconstrua no descompactador um modelo idêntico ao gerado no compactador, em um *processo adaptativo, de aprendizado e compressão incrementais*?

- O sistema neural desenvolvido deverá compactar adaptativamente arquivos binários e de *texto*, sem prévio treinamento *off-line*, apenas com o conhecimento do alfabeto discreto ao longo do qual as mensagens podem assumir seus valores. Diante destes requisitos, qual será seu desempenho em relação a compactadores de *texto* tradicionais?
- O método de compressão concebido é aplicado à *classificação de textos* através da construção de *modelos de linguagem* a partir de exemplares que representem cada classe. O propósito do sistema é a classificação automática de textos novos e desconhecidos, através da medição de *entropia cruzada* de cada exemplar apresentado em relação aos *modelos* gerados por módulos da rede neural proposta. Assim, uma vez que a classificação seja realizada sem a pré-seleção de atributos que representem uma determinada classe de textos, a metodologia aplicada por meio do modelador neural será capaz de apresentar resultados satisfatórios na abordagem do problema?

## 1.4 Objetivos do Trabalho

### 1.4.1 Objetivo Geral

Desenvolver um sistema de compressão sem perda através da utilização de uma rede neural artificial baseada na Teoria da Ressonância Adaptativa, mais especificamente, uma rede fuzzy ARTMAP modificada.

### 1.4.2 Objetivos Específicos

- Propor e implementar modificações sobre a rede neural artificial fuzzy ARTMAP, com base em características observadas, que possam adequá-la à função de modelo gerador de estimações probabilísticas no contexto do sistema adaptativo de compressão de texto para operação *on-line*.
- Desenvolver e implementar o Sistema Adaptativo Neural de Compressão, testando-o em relação a uma base de dados pública de referência.
- Desenvolver, implementar e testar uma metodologia para identificação de idioma e classificação por gênero de texto, baseada na técnica de compactação desenvolvida.
- Apresentar, por meio do ambiente de compressão sem perda, uma aplicação em que as características de aprendizado seqüencial, incremental e *on-line* da rede neural possam ser exploradas.

## 1.5 Síntese da Abordagem e Justificativa

A abordagem do trabalho consiste em uma alternativa ainda não explorada[2], em que uma rede fuzzy ARTMAP modificada é utilizada como modelo gerador de estimações probabilísticas para compressão sem perda. Os requisitos de operação *on-line* do sistema de compressão sem perda apontaram para a alternativa de utilização de uma RNA da classe ARTMAP, sendo que a rede neural implementada com as devidas modificações é aqui denominada de *c*-ARTMAP (*compressor* ARTMAP). Essencialmente, o sistema de compressão desenvolvido utiliza *c*-ARTMAP para a construção dinâmica de um modelo do ambiente de entradas e para a formação de previsões a cada passo que sucede uma entrada para o sistema. O sistema realiza a compactação, ao mesmo tempo em que aprende os padrões apresentados seqüencialmente à rede neural, adaptando-se às entradas. Por meio da operação do sistema é possível medir a entropia ou *quantidade de informação* da *mensagem* (seqüência de entradas), em relação ao modelo construído incrementalmente pela rede neural.

Durante a compressão de *texto on-line*, para cada subseqüência de *texto*, dada como entrada, a rede neural deve prever o símbolo seguinte, provendo uma distribuição probabilística ao longo dos símbolos que possam sucedê-la. A distribuição probabilística estimada é utilizada por um codificador aritmético para conversão das estimativas em uma seqüência de bits de saída. Para descompactação, o decodificador deve reconstruir um modelo neural idêntico, partindo do mesmo estado inicial que o codificador para possibilitar a restauração do arquivo sem perda de informação. A atualização da rede neural, com a incorporação do conhecimento sobre o símbolo que efetivamente sucede cada subseqüência, só ocorre após este ter sido codificado.

Como extensão do sistema, é proposta e aplicada uma metodologia para classificação de textos baseada na técnica de compactação desenvolvida, sendo testada para identificação de idiomas e classificação por gênero de texto. Na classificação por gênero de texto o propósito será categorizar textos em classes de publicações, conforme descrito na metodologia (Seção 1.6). A abordagem de classificação do presente trabalho concentra-se na discriminação destes conforme sua similaridade em relação a modelos construídos pela rede neural, sem considerar a formatação do texto e sem utilização de conhecimento sintático ou lexical.

[2] Após pesquisa bibliográfica nas bases de dados de textos completos (IEEE, ACM, Pergamon-Elsevier, *Science Direct*, CiteSeer), assim como na internet pública, não foi encontrado, até abril de 2000, qualquer trabalho que utilizasse uma RNA baseada em *ART* para compressão de *texto*.

### 1.5.1 Justificativa

A proposta do presente trabalho é justificada pelo conjunto de questões salientadas a seguir. Estas sintetizam as motivações já descritas ao longo deste capítulo, as decisões de desenvolvimento que respondem às questões de pesquisa formuladas, com base nas quais foram enunciados os objetivos do presente trabalho.

**O volume crescente de informação em formato digital** motiva o esforço de pesquisa para o desenvolvimento de mecanismos de compressão de dados *sem perdas* e de técnicas para extração de conhecimento em bases de dados textuais.

**Alternativa às abordagens atuais de RNAs em compressão de *texto*.** Mesmo diante dos muitos métodos de compressão eficientes, é possível que RNAs ofereçam uma perspectiva alternativa à compressão sem perda ao atuar na etapa de modelagem, por meio do aprendizado de regularidades entre padrões adquiridos. Abordagens anteriores (SCHMIDHUBER e HEIL, 1996; NATSEV, 1997; LONG, NATSEV e VITTER, 1999) que utilizaram redes *Feedforward* com múltiplas camadas, previamente treinadas por meio do algoritmo *Backpropagation*, restringiram-se à compressão de textos originários da mesma classe de *fonte* de linguagem utilizada no treinamento. Uma abordagem que seja apropriada à compressão de *texto* de propósito geral demanda uma RNA com capacidade de adaptação aos arquivos em compressão. Uma modelagem totalmente adaptativa pode ser executada por uma rede neural da classe ARTMAP (originária de *ART*) com modificações propostas que resultaram na rede c-ARTMAP (apresentada no Capítulo 5). Os testes do sistema desenvolvido, diante de uma base de dados pública para *benchmark* em compressão de texto, possibilitam uma visão mais crítica da capacidade do método de compressão proposto.

**Aplicações de classificação automática de textos no ambiente organizacional.** A classificação automática de textos pode ser vista como uma das tarefas de processos de mineração de textos (TAN, 1999), área de pesquisa que permite a descoberta de conhecimento a partir de textos. Esta área apresenta importância crescente para a inteligência de negócios, com diversas aplicações nas áreas de *marketing*, análise de perfil e organização de informações de *feedback* do consumidor, em estratégia de negócios para análises de patentes, descoberta de conhecimento e oportunidades de mercado, além de organização, filtragem e disseminação de documentos que atendam aos interesses de destinatários.

**Classificação automática de textos por meio do método de compressão desenvolvido**. Além das características adaptativas, RNAs podem acomodar mais informação no modelo gerado a partir da mensagem que, uma vez extraída, pode oferecer outras opções além da

compressão. A rede neural artificial proposta é utilizada para o aprendizado de um *modelo* probabilístico preditivo que extrai as características de cada seqüência durante sua compactação, utilizando-o no processo de classificação automática de textos. Por meio da técnica de compressão, os textos são processados automaticamente, sem extração prévia de atributos, simplificando os procedimentos geralmente implementados por métodos de aprendizado de máquina aplicados à classificação de textos. A similaridade entre o exemplar e cada uma das classes é estabelecida autonomamente pelo sistema, sem a pré-definição de atributos e sem a necessidade de conhecimento analítico acerca do texto ou do idioma.

**Desenvolvimento de uma arquitetura modular e flexível.** A concepção de um sistema de compactação sem perda que utiliza a estrutura básica *modelador-codificador* possibilita o desenvolvimento de uma arquitetura modular e flexível. A RNA proposta é integrada a um codificador estatístico por meio de uma interface de comunicação, de maneira que o modelador possa utilizar outra variante ARTMAP e o módulo de codificação comporte um codificador aritmético ou um codificador de *Huffman*. Neste trabalho, o desempenho do compactador será considerado em função das taxas de compressão obtidas. Portanto, o desenvolvimento do sistema neural de compactação (e descompactação) não estabeleceu, como objetivo, o alcance de *performance* de velocidade de processamento competitiva com a dos compactadores disponíveis em *softwares* comerciais e livres, especificamente desenhados para esta função. Futuras otimizações (já previstas) na implementação do sistema poderão propiciar-lhe um aumento de velocidade considerável.

## 1.6 Metodologia do Trabalho

Partindo da motivação inicial e de subsídios teóricos, foi formulada a hipótese de que uma rede neural artificial da classe ARTMAP poderia ser utilizada para a modelagem e geração de estimações probabilísticas em compressão de *texto*. Para verificar esta viabilidade em um ambiente de aplicação, visando à obtenção de resultados quantitativos, foi desenvolvido, implementado e testado um sistema neural de compressão para arquivos binários e de texto. Para testar o sistema neural foi utilizada uma base de dados de domínio publico, sendo os resultados obtidos comparados com os atingidos por compactadores de *texto* tradicionais. Para verificar o desempenho do método neural de compactação desenvolvido diante da tarefa de classificação automática de textos, este foi testado em duas modalidades de aplicação. A metodologia adotada para o alcance dos objetivos definidos para este trabalho é detalhada a seguir.

**Fundamentação Teórica e Revisão da Literatura**

- Fundamentação teórica básica em compactação *sem perda* aplicada à compressão de *texto* e revisão de métodos tradicionais desta área.
- Breve embasamento conceitual em classificação automática de textos.
- Levantamento dos requisitos necessários à modelagem adaptativa em compressão de *texto* e sua operação coordenada com a de um codificador estatístico para operação *on-line*.
- Revisão de conceitos relativos a Redes Neurais Artificiais (RNAs) para contextualização de redes baseadas na Teoria da Ressonância Adaptativa (*ART*) e de trabalhos anteriores de aplicação de RNAs em compressão de texto.
- Estudo da RNA fuzzy ARTMAP para avaliar sua adequação à função de "estimador probabilístico" em compressão de texto.

**Desenvolvimento e Implementação da Rede Neural e do Sistema de Compressão**

- Definição e implementação do modelo modificado da rede fuzzy ARTMAP, denominado de *c*-ARTMAP, para a realização de estimações probabilísticas no modo *on-line* em compressão de *texto*.
- Desenvolvimento e implementação da arquitetura do sistema de compactação (e descompactação) com integração dos três módulos componentes principais: *c*-ARTMAP para modelagem, codificador aritmético e controle de operação do processo, com definição do método para *tratamento de escape* a ser utilizado.

**Aplicação do Sistema Desenvolvido para Compressão de Texto**

- Sistematização do procedimento experimental para avaliação do sistema e aferição do seu desempenho diante de bases de dados públicas para compressão de texto: *Canterbury* e *Large corpus* (BELL, 1998), utilizando a taxa de compressão em *bits por caracter* como medida. A base *Canterbury* é considerada *benchmark* de referência (WITTEN, MOFFAT, BELL, 1999) para teste de novos algoritmos de compressão de texto. Esta base incorpora: imagem de bitmap (fax), planilha Excel, código fonte de programa, além de exemplares de diversos textos em linguagem natural no idioma inglês. O *Large corpus* é utilizado para testar algoritmos de compressão diante de arquivos mais extensos (seqüência de DNA; versão da bíblia (*King James Bible*); relatório da C.I.A. com informações geo-políticas de diversos países).
- Calibragem da rede neural modeladora para obtenção de um conjunto de parâmetros de operação satisfatórios que definem o estado inicial do sistema para execução dos testes, utilizando arquivo *alice.txt* do *Canterbury corpus*. Calibração e testes com utilização da

rede fuzzy ARTMAP para estimações probabilísticas no módulo modelador para comparação com a rede *c*-ARTMAP proposta.

- Realização dos testes e comparação dos resultados atingidos pelo sistema neural de compressão em relação aos obtidos pelos compactadores tradicionais *gzip*, *compress*, *pack* e o método estado da arte PPM, diante das referidas bases de *benchmark*.
- Análise e discussão dos resultados e do desempenho do sistema desenvolvido.

**Aplicação do Método Neural de Compressão à Classificação de Textos**

- Desenvolvimento da metodologia baseada no método de compressão desenvolvido, com aplicação do conceito de *entropia cruzada* à classificação automática de textos.
- Definição do procedimento experimental e formação de uma base de dados para identificação de idiomas e classificação por *gênero* de textos. Para a primeira modalidade foram utilizados textos da bíblia em seis idiomas e, para a classificação por *gênero* de textos foram utilizadas publicações *on-line* (cujos títulos estão listados no Anexo A).
- Realização de testes e discussão sobre os resultados obtidos.

**Conclusão do Trabalho**

- Conclusão relativa aos resultados das aplicações e às observações do comportamento da rede neural e do sistema de compressão, levantamento de aprimoramentos previstos e perspectivas para trabalhos futuros.

**Nota em Relação à Terminologia, Simbologia e Notações Matemáticas Utilizadas.**

Ressalta-se que no decorrer do presente trabalho os termos *compressão* e *compactação* são utilizados como sinônimos. Conforme Bell, Cleary e Witten (1990), uma diferenciação poderia ser feita quanto a esta terminologia, de modo que o termo compactação (*compaction*) corresponderia à redução de tamanho de um arquivo por meio de um processo irreversível (sem a remoção de qualquer informação relevante), e a compressão (*compression*) seria uma redução completamente reversível. No entanto, quando aqui empregados para a abordagem de compressão de *texto*, ambos os termos correspondem à modificação na representação de dados de um arquivo, sem perda de informação, sendo este 100% reversível à forma original, quando descompactado.

O significado da simbologia e das notações matemáticas adotadas ao longo deste trabalho é explicitado no contexto de cada capítulo da dissertação.

## 1.7 Estrutura da Dissertação

O trabalho está organizado em nove capítulos, seguidos da bibliografia e do Apêndice A.

**Capítulo 1: Introdução**. Ao longo deste capítulo foram introduzidos os temas tratados nesta dissertação, apresentando a motivação, o contexto, as questões de pesquisa, os objetivos do presente trabalho e sua justificativa, assim como a metodologia seguida no seu desenvolvimento.

**Capítulo 2: Compressão sem perda - A Compressão de Texto**. Neste capítulo é realizada uma introdução à compressão de texto, abordando conceitos necessários à fundamentação da técnica de compactação desenvolvida. São apresentadas, brevemente, definições da Teoria da Informação, entropia, modelagem, codificação, e são descritos alguns dos principais métodos de compactação existentes, com ênfase em métodos adaptativos e codificação aritmética.

**Capítulo 3: Redes Neurais Artificiais e a Teoria da Ressonância Adaptativa**. São revisados conceitos relativos a RNAs e é apresentada uma introdução à *ART*. São descritos alguns modelos baseados em *ART*, com suas respectivas características, assim como abordagens de trabalhos anteriores em que RNAs foram utilizadas para compressão de texto.

**Capítulo 4: Rede Neural Artificial fuzzy ARTMAP**. São descritas as características, a arquitetura e a dinâmica desta rede neural, assim como seus modos de operação para classificação, previsão de padrões e para estimação probabilística.

**Capítulo 5: Modelo de Rede Neural fuzzy ARTMAP Adaptado**. São apresentados: o algoritmo, a arquitetura e as características do modelo *c*-ARTMAP desenvolvido a partir de modificações sobre fuzzy ARTMAP. São ressaltados os aspectos que conduziram às adaptações realizadas com base na avaliação do algoritmo e comportamento de operação da rede fuzzy ARTMAP original.

**Capítulo 6: Sistema Adaptativo Neural para Compressão de Texto**. É apresentado o Sistema Adaptativo Neural de Compressão sem perda, suas características, sua arquitetura, e é descrito o processo de compactação e descompactação *on-line*.

**Capítulo 7: Experimentos, Testes de Compressão e Análise de Resultados**. São relatados os procedimentos experimentais e resultados alcançados nos testes, sendo feita uma análise do desempenho do sistema adaptativo neural para compressão de *texto*.

**Capítulo 8: Aplicação do Sistema Adaptativo Neural para Compressão à Classificação de Textos**. É apresentada a metodologia aplicada à classificação de textos, baseada na técnica de compactação que utiliza *c*-ARTMAP. São relatados os procedimentos

experimentais e os resultados dos testes de identificação de idiomas e de classificação de textos por similaridade de gênero.

**Capítulo 9: Conclusão**. São apresentadas as conclusões relativas às características e à adequação da rede neural *c*-ARTMAP no contexto do sistema adaptativo de compactação desenvolvido e sua aplicação à classificação de textos. São apontados aprimoramentos previstos para a rede neural e para o sistema de compressão sem perda, assim como perspectivas para trabalhos futuros.

**Apêndice A**. Apresenta a lista de títulos dos textos utilizados nas tarefas de Identificação de Idioma e Classificação por Gênero de Texto (disponíveis publicamente nos *sites* de origem na internet).

# 2 COMPRESSÃO SEM PERDA - A COMPRESSÃO DE *TEXTO*

O método neural proposto e integrado ao sistema de compactação desenvolvido é aplicado à compressão de *texto*. A compressão de *texto* corresponde à compressão *sem perda* de seqüências discretas que formam arquivos binários e de texto comumente encontrados em sistemas computacionais. Para fundamentação teórica em compressão de *texto* e sua aplicabilidade à classificação automática de textos são apresentados, neste capítulo, conceitos relativos aos tópicos: Teoria da Informação, modelagem e codificação, modelos de contexto finito, métodos Lempel-Ziv, PPM e Codificação Aritmética.

## 2.1 Introdução

A compactação de dados possibilita uma diminuição nos custos associados à sua comunicação ao viabilizar uma redução no tempo necessário à transmissão dos dados e no espaço requerido para seu armazenamento. Um algoritmo de compressão opera sobre as regularidades, redundâncias e padrões identificados em uma seqüência de dados, gerando para estes uma representação mais compacta que utiliza um número de bits inferior ao utilizado pela versão original. O desenvolvimento de um compactador deve, necessariamente, visar à restauração dos dados a ser realizada por meio do correspondente descompactador. Assim, quanto ao grau de reversibilidade dos dados à sua forma original, os métodos de compressão são divididos em métodos com perda (*lossy*) e métodos sem perda (*lossless*) de informação.

Para dados originados de sinais analógicos, como imagens, seqüências de vídeo e de áudio são, geralmente, aplicadas técnicas de compressão com perda, em que parte da informação original pode ser descartada em favor de uma melhor taxa de compactação. Durante a codificação, as redundâncias existentes nestes sinais são exploradas por meio de operações que produzem representações extremamente compactas destes. Assim, a versão aproximada restaurada na descompactação apresenta um grau de distorção em relação à original, com uma degradação tolerável ou sensorialmente imperceptível na qualidade desta.

Para determinadas aplicações, no entanto, é necessária a compressão sem perda de informação, de modo que os dados sejam perfeitamente reconstruídos à forma original na descompactação. Portanto, uma técnica de compressão sem perda realiza a codificação completamente reversível de uma seqüência de dados que, convertida para uma representação mais eficiente, utiliza um número menor de bits que a versão original. Alguns casos de aplicação para compressão sem perda são citados a seguir.

- Imagens em aplicações médicas, como exames tomográficos e ultra-sonografias, em que qualquer inexatidão na versão reconstruída do arquivo pode conduzir a erros de diagnóstico.
- Imagens de documentos e de exames médicos que precisam ser utilizados com finalidades legais, assim como imagens de valor histórico.
- Imagens de sensoriamento remoto que são processadas para avaliação posterior e cuja reconstrução aproximada pode ser inadequada para a efetiva utilização. A improbabilidade de se adquirir dados exatamente iguais aos capturados anteriormente, assim como o alto custo envolvido no processo de aquisição destas imagens, justificam a compressão sem perda.
- Seqüências de dados que são intrinsecamente discretos e que requerem absoluta fidelidade da versão reconstruída em relação à original, como: arquivos de textos em linguagem natural, códigos fonte, programas executáveis e planilhas de dados. Para estes exemplos são comumente adotados os princípios da compressão de *texto*. Programas de compressão sem perda, como *pkzip* e *gzip*, podem ser aplicados à compressão de *texto*, abordando de forma satisfatória uma variedade de tipos de seqüências discretas (binárias ou de *texto*).

O presente trabalho enfoca a denominada compressão de *texto*, ou seja, a compactação *sem perda* de seqüências discretas por meio de um algoritmo neural desenvolvido e aplicável a arquivos de textos em linguagem natural, imagens de *bitmap*, códigos fonte, programas executáveis, planilhas de dados e seqüências de DNA.

Seqüências discretas são constituídas por símbolos que assumem valores em um determinado alfabeto que pode ser definido, por exemplo, pelo código ASCII (com 256 símbolos distintos), representando muitos dos arquivos binários e de texto encontrados em sistemas computacionais. Uma das características de dados de seqüências de texto é a de que os valores correspondentes aos símbolos sucessivos de uma seqüência são, geralmente, não-correlacionados, em contraste com os dados em uma imagem, por exemplo, em que, freqüentemente, existe alto grau de correlação entre os valores de *pixels* adjacentes. Para um texto em linguagem natural a relação de dependência entre caracteres sucessivos em uma subseqüênci reflete as regularidades de um determinado idioma, conteúdo ou gênero de texto. Em função das características destes dados e da necessária exatidão da versão reconstruída a partir de sua representação compactada, a quantidade de compressão alcançada para

seqüências de *texto* é, normalmente, muito menor que a obtida para dados que podem ser comprimidos com perda de informação.

Para realizar a compressão de *texto* pode-se trabalhar com a construção de um *modelo* que procura refletir ou ser uma aproximação da *fonte* discreta que emite as seqüências de símbolos ou mensagens. O paradigma *modelagem-codificação* (MOFFAT, BELL e WITTEN, 1995) estabelece a clara separação entre a função de geração de um *modelo* e a função de codificação. Esta separação de estágios permite o foco no desenvolvimento de modelos que procuram capturar as características de uma seqüência processada, visando à codificação desta por meio de um número de bits próximo do ótimo para sua compactação sem perda. A Teoria da Informação (SHANNON, 1948) estabelece o limite de compressibilidade de uma mensagem, e seus conceitos básicos, aplicados à compressão de *texto*, são apresentados na Seção 2.2. A interação entre o processo de modelagem e codificação, assim como as várias formas de modelagem, são tópicos tratados na Seção 2.3. Conceitos de modelos de *contexto finito* que constituem a base para diversos métodos de compressão de *texto* são apresentados na Seção 2.2.1.1 e na Seção 2.3.3.

Dentre as técnicas mais conhecidas aplicadas à compressão de *texto* podem ser destacadas: a codificação de *Huffman* (SAYOOD, 2000), a Codificação Aritmética (WITTEN, MOFFAT e BELL, 1999; SAYOOD, 2000), a família de métodos *Lempel-Ziv* (ZIV e LEMPEL, 1977; 1978), os métodos PPM (*Prediction by Partial Matching*) (CLEARY e WITTEN, 1984) e BWT (*Burrows-Wheeler Transform*) (SAYOOD, 2000). Algumas variantes da família de métodos *Lempel-Ziv* são apresentadas na Seção 2.4.1. O método PPM é apresentado na Seção 2.4.2, e o Codificador Aritmético é descrito na Seção 2.4.3.1. A Seção 2.5 discorre sobre os critérios de desempenho aplicados à compressão de *texto*.

## 2.2 Conceitos da Teoria da Informação em Compressão de Texto

Introduzida por Claude Shannon (SHANNON, 1948), a Teoria da Informação fornece uma base conceitual e matemática para diversas áreas, dentre as quais: compressão e comunicação de dados, processamento de sinais, criptografia, modelagem computacional de linguagem e reconhecimento de padrões.

A Teoria da Informação estabelece os limites fundamentais para o número mínimo de bits necessários para representar cada símbolo de uma *fonte* de informação, formando a base teórica para a compressão de dados. Uma *fonte* de informação pode ser descrita por um processo estocástico que origina *mensagens* a serem transmitidas a um terminal receptor

através de um canal de comunicação. Uma *fonte* discreta emite mensagens formadas por símbolos que assumem valores discretos ao longo de um conjunto finito, denominado de alfabeto da fonte.

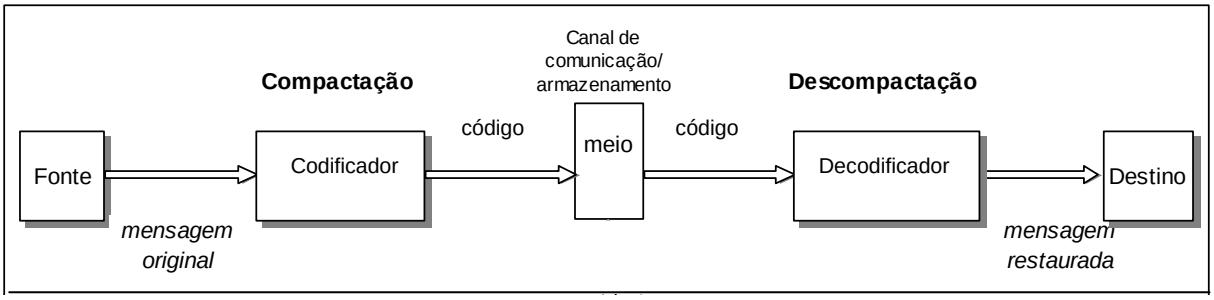



Figura 1: Esquema de envio e recepção de uma mensagem através de um canal de comunicação.

A Figura 1 faz uma descrição do esquema básico de transmissão e recepção de uma mensagem através de um canal de comunicação com compactação e descompactação de dados. O processo de compactação é efetuado pelo *codificador* que constrói um mapeamento dos símbolos do alfabeto da *fonte* para o código, utilizando-o para codificar, em uma seqüência de bits, a mensagem original emitida pela *fonte*. Esta seqüência é enviada através de um canal de transmissão (ou mantida em um meio para armazenamento) que, neste trabalho, será considerado sem ruído. O processo de descompactação é realizado pelo *decodificador* que, por meio do processo inverso, deve restaurar a mensagem codificada à forma original.

Cada módulo codificador (e decodificador) pode ser subdividido em dois estágios: a construção de um *modelo* da fonte de informação e a *codificação* que utiliza este *modelo* para geração da seqüência de bits que forma o código da mensagem. O objetivo da codificação da fonte é a construção de um código que represente a mensagem com o número mínimo de bits por símbolo, respeitando o limite da entropia da mensagem, e que permita a decodificação desta sem ambigüidade.

### 2.2.1 Definições Básicas de Entropia

Do ponto de vista da Teoria da Informação (SHANNON, 1948), a entropia consiste na medida de incerteza de ocorrência de uma determinada mensagem, ou, mais especificamente, na quantidade de incerteza envolvida na seleção de um símbolo dentre os símbolos possíveis do alfabeto da *fonte* da mensagem. Isto significa que, quanto maior a entropia, maior a incerteza em relação à ocorrência de um determinado símbolo. A entropia da fonte pode ser considerada como a medida da quantidade média de informação de uma mensagem, sendo que mensagens com menor probabilidade associada (menos previsíveis) contêm maior

quantidade de informação, quando reveladas, que as mensagens com maior probabilidade de ocorrência (mais previsíveis).

Uma *fonte* discreta de informação pode ser modelada como um processo estocástico de tempo discreto $\{X_t, t=1, 2, 3,...\}$ que gera variáveis aleatórias $X_1, X_2, X_3, ...$, em que $X_t$ assume como saída um símbolo do alfabeto finito e discreto da fonte.

Seja uma variável aleatória discreta $X$, que pode assumir como saída um dos símbolos de um alfabeto designado por $\Omega$, contendo $|\Omega|=m$ símbolos distintos com distribuição de probabilidades $P=\{p_1, p_2, ..., p_m\}$, em que $p_i \geq 0$ e $\sum_{i=1}^{m} p_i = 1$, para $i=1, 2, ..., m$. Para uma *fonte* discreta *sem memória*, em que as variáveis aleatórias são independentes e identicamente distribuídas (i.i.d.), a entropia para $X$ é dada por:

$$H(X) = -\sum_{i=1}^{m} p_i \log_2 p_i \tag{2.1}$$

A entropia $H(X)$ fornece o menor número de bits necessários, na média, para representar um símbolo desta fonte, utilizando um modelo que supõe a independência estatística entre os símbolos emitidos pela fonte. Convenciona-se que $0 \log_2 0 = 0$, uma vez que $x \log_2 x \rightarrow 0$ para $x \rightarrow 0$ (COVER e THOMAS, 1991).

A quantidade de informação individual $h_i$ para cada símbolo distinto de $\Omega$ corresponde ao mínimo número de bits requeridos para codificá-lo, sendo dada por:

$$h_i = -\log_2 p_i \tag{2.2}$$

Associadas à definição da entropia $H(X)$ são listadas algumas propriedades.

- $H(X)$ varia conforme $0 \leq H(X) \leq \log_2 |\Omega|$.
- Para qualquer $X$, $H(X) \geq 0$, uma vez que $0 \leq p_i \leq 1$ e $-\log_2 p_i \geq 0$.
- A entropia de uma mensagem é máxima, com $H(X)=\log_2 |\Omega|$, para o caso de símbolos equiprováveis ao longo de $\Omega$, ou seja, com $p_i = 1/|\Omega|$, para $\forall\ i$.
- A entropia de uma mensagem é mínima, ou seja, $H(X)=0$, se $p_i=1$, para apenas um dos símbolos de $\Omega$, e todos os demais símbolos, com $i \neq j$, tem probabilidades $p_j=0$.
- Quanto maior a probabilidade de um evento, menor a sua quantidade de informação. A certeza de ocorrência de um evento (ou seja, $p_i=1$) implica na ausência de ganho de informação com a sua realização. Em outras palavras, não existe necessidade de envio de uma mensagem com quantidade de informação nula.

Sejam seqüências denotadas por $x_1^T = (x_1, x_2,..., x_T)$, geradas pela fonte a partir do conjunto $\Omega^T$ de seqüências possíveis de tamanho $T$, com função de distribuição de probabilidades $p(x_1^T) = p(x_1, x_2,..., x_T)$, sendo $p(x_1^T) \geq 0$ e $\sum_{x_1^T} p(x_1^T) = 1$, para toda $x_1^T \in \Omega^T$. Considerando-se o somatório efetuado sobre todas as seqüências $x_1^T$, têm-se a entropia $H(X_1, X_2, ..., X_T)$ para uma seqüência de variáveis aleatórias:

$$H(X_1, X_2,..., X_T) = -\sum_{x_1^T \in \Omega^T} p(x_1^T) log_2 p(x_1^T) \tag{2.3}$$

A entropia $H(X_1, X_2,..., X_T)$ fornece o mínimo número de bits, na média, necessários para codificar as seqüências geradas pelo processo estocástico que descreve esta fonte de informação.

O aumento da entropia para uma seqüência de variáveis aleatórias $X_1, X_2,..., X_T$, em decorrência do crescimento de $T$, é dado pela *taxa de entropia* do processo estocástico (COVER e THOMAS, 1991), definida por:

$$H = \lim_{T \to \infty} \frac{1}{T} H(X_1, X_2,..., X_T) \tag{2.4}$$

A taxa de entropia corresponde ao esperado número médio de bits por símbolo necessários para descrever este processo, para seqüências de $T$ variáveis aleatórias emitidas pela fonte, com $T$ tendendo ao infinito. (Para a Equação 2.4 assume-se que o limite exista)

Em geral, é extremamente difícil determinar a entropia para uma *fonte* arbitrária apenas por meio da observação das saídas que esta emite, pois são necessárias seqüências muito longas para o cômputo acurado das probabilidades. O valor exato da entropia depende do conhecimento da estrutura estatística da *fonte*. Como, na prática, a estrutura desta é, geralmente, desconhecida (SAYOOD, 2000), procura-se estimá-la a partir da distribuição probabilística provida por um *modelo*, sendo que modelos diferentes produzem estimativas diferentes. O propósito de um modelo é deduzir uma aproximação do processo estatístico subjacente da *fonte*, com base nas mensagens geradas por esta. Um modelo capaz de capturar as características da estrutura estatística da fonte possibilitaria a codificação de uma mensagem em um número de bits próximo de sua entropia, mas não inferior a esta. Claude Shannon (1948) demonstrou que não é possível compactar uma seqüência sem perda de informação com um número de bits abaixo do valor de sua entropia. De acordo com o *Teorema da Codificação da Fonte sem ruído*[3], concebido por Claude Shannon (1948), a

[3] *Shannon's Noiseless Source Coding Theorem*

entropia define o limite mínimo para o número médio de bits por símbolo da fonte, requeridos para codificar uma mensagem por meio de um dado modelo, de modo que esta seja reconstruída sem perda na sua decodificação.

#### 2.2.1.1 Entropia e Modelos de Contexto Finito

Seja a seqüência discreta $x_1^T = x_1, x_2, ..., x_T$, gerada por um processo estocástico, em que símbolos $x_t$, $t \leq T$, assumem valores de um alfabeto finito Ω, contendo |Ω| símbolos distintos, com função de distribuição de probabilidades $p(x_1^T)=p(x_1, x_2, \ldots, x_T)$. De acordo com os preceitos de entropia da Teoria da Informação, o mínimo comprimento possível $l$, em bits, para o código que representa uma seqüência observada $x_1^T =(x_1, x_2, ..., x_T)$ é dado por meio de sua quantidade de informação $h$:

$$h = -log_2 p\left(x_1^T\right) \tag{2.5}$$

Considera-se que $p(x_1, x_2,..., x_T)$ para a seqüência $x_1^T$ é dado por:

$$p(x_1, x_2,..., x_T) = p(x_1)\, p(x_2|\, x_1) \ldots p(x_T|\, x_1\, x_2 \ldots x_{T-1})$$

$$= \prod_{t=1}^{T} p\left(x_t \mid x_1,..., x_{t-1}\right) \tag{2.6}$$

Assim, dado que, para o menor comprimento de código possível, $h = l$:

$$l = -log_2 \prod_{t=1}^{T} p\left(x_t \mid x_1,..., x_{t-1}\right)$$

$$= -\sum_{t=1}^{T} log_2 p\left(x_t|\, x_1,..., x_{t-1}\right) \tag{2.7}$$

Deste modo, um símbolo $x_t$ é codificado por meio da distribuição de probabilidades condicionais $p(x_t|x_1,..., x_{t-1})$, utilizando a seqüência $x_1,..., x_{t-1}$ que o precede. Na prática, para aproximar estas probabilidades em um processo de compressão sem perda são comumente adotados modelos de *contexto finito*, baseados na concepção de cadeias de *Markov*[4] de tempo discreto (SAYOOD, 2000).

Para a compressão de texto, um modelo de contexto finito baseia-se na suposição de que as seqüências emitidas pela fonte seguem um modelo de *Markov* de *ordem r*. Seja uma seqüência discreta $x_1^T=x_1, x_2,\ldots, x_T$, em que símbolos $x_t$, para $t$=1,..., $T$, assumem valores de um alfabeto da fonte. A seqüência segue um modelo de *Markov* de *ordem r* (sendo $r$ um valor inteiro e finito, $r<t$), assumindo-se que, para estimar a probabilidade para o símbolo $x_t$, o

[4] Em homenagem ao matemático russo Andrey Andreyevich Markov.

conhecimento dos $r$ símbolos consecutivos que antecedem $x_t$ na seqüência é equivalente ao conhecimento de toda a *história* passada $x_1, x_2, \ldots, x_{t-1}$, ou seja:

$$p(x_t \mid x_1, x_2, ..., x_{t-1}) = p(x_t \mid x_{t-r}, .., x_{t-1}) \tag{2.8}$$

A seqüência $x_{t-r}, \ldots, x_{t-1}$ observada corresponde ao *contexto*, designado por $s$, formado pelos $r$ símbolos consecutivos utilizados para estimar a probabilidade para o símbolo subseqüente $x_t$.

Seja uma fonte discreta, da qual cada saída pode assumir um símbolo dentre os símbolos possíveis de um alfabeto finito denotado por $\Omega=\{\varpi_1, \varpi_2, ..., \varpi_m\}$, com distribuição de probabilidades $P=\{p_1, p_2, ..., p_m\}$, sendo que $p_i \geq 0$ e $\sum_{i=1}^{m} p_i = 1$, para $i=1, 2, ..., m$. A quantidade de informação decorrente de um símbolo $\varpi_i$ é dada por $-log_2 p_i$, o que fornece o número de bits necessários para codificá-lo. Assim, um símbolo que possui pouca probabilidade associada contém maior quantidade de informação e necessita de um maior número de bits para ser representado do que um símbolo com grande probabilidade de ocorrência. A quantidade média de informação por símbolo do alfabeto $\Omega$ pode ser computada por meio da ponderação da quantidade de informação de cada símbolo por sua probabilidade de ocorrência, sendo dada por meio da Equação 2.1. A entropia referente à Equação 2.1 corresponde à entropia da distribuição probabilística em bits por símbolo, dada em função de um modelo de *ordem zero*[5]. Esta fornece o número de bits, em média, necessários para codificar cada elemento de uma seqüência emitida por esta fonte, em que os elementos sucessivos são estatisticamente independentes entre si.

A suposição da dependência entre elementos sucessivos de uma seqüência gerada pela fonte pode ser representada utilizando-se modelos de contexto finito, considerando-se um conjunto de contextos possíveis, designado por $S$, dentre os quais, um contexto $s$ assume um determinado valor. Para cada $s \in S$, existe um conjunto de probabilidades condicionais $p(\varpi|s)$, de que $s$ seja sucedido um dos possíveis símbolos $\varpi \in \Omega$. A entropia da fonte para este modelo será definida como a média das entropias para cada $s$, ponderadas de acordo com a função de distribuição de probabilidades $p(s)$ de ocorrência dos contextos possíveis, ou seja:

$$H = -\sum_{s \in S} p(s) \sum_{\varpi \in \Omega} p(\varpi \mid s) log_2 p(\varpi \mid s) \tag{2.9}$$

Uma vez que este modelo reflita as características observadas de dependência do processo da *fonte*, sua entropia é menor do que a que seria obtida para a mesma situação, se fosse

[5] Este índice para a *ordem* do modelo segue a convenção de (BELL, CLEARY e WITTEN, 1990), em que um contexto formado por $r$ símbolos é relativo a um modelo de *ordem r*.

utilizado um modelo para o qual é assumido que os símbolos são estatisticamente independentes. O conhecimento do contexto antecessor reduz a incerteza para a seleção de um símbolo sucessor dentre os vários possíveis em um dado alfabeto finito.

#### 2.2.1.2 Entropia Cruzada e Modelos de Linguagem

O conceito de entropia pode ser aplicado a *modelos de linguagem* (TEAHAN, 1998), sendo uma dada linguagem vista como um processo estocástico e considerada como uma *fonte* de informação. Seja uma *linguagem* $\mathfrak{I}$, em que seqüências $x_1^T=(x_1, x_2,..., x_T)$ são geradas conforme uma distribuição probabilística, e em que os símbolos assumem valores em um determinado alfabeto discreto e finito. A entropia da linguagem $\mathfrak{I}$ pode ser considerada no limite, à medida que o comprimento da mensagem se torna muito longo, sendo dada por:

$$H(\mathfrak{I}) = -\lim_{T\to\infty} \frac{1}{T} \sum_{x_1^T \in \mathfrak{I}} p(x_1^T) \log_2 p(x_1^T) \tag{2.10}$$

O somatório é efetuado sobre todas as seqüências $x_1^T$ de comprimento $T$, ao longo do alfabeto da linguagem $\mathfrak{I}$. Geralmente, a distribuição probabilística de $\mathfrak{I}$ é desconhecida, mas para aproximá-la pode ser utilizado um *modelo de linguagem M*, por meio de função de distribuição de probabilidades $p_M(\cdot)$.

Para expressar a relação entre $\mathfrak{I}$ e *M*, é necessário considerar, inicialmente, o conceito de entropia relativa, ou divergência de *Kullback-Leibler* (HAYKIN, 1999) entre duas distribuições de probabilidades sobre o mesmo espaço de eventos, sendo que $p(x)$ denota a função de distribuição "real" e $q(x)$ denota uma função de distribuição estimada:

$$D(p\|q) = \sum_x p(x) \log_2 \frac{p(x)}{q(x)} \tag{2.11}$$

A divergência $D(p||q)$ é sempre positiva, porém não simétrica, igualando-se a zero apenas para $p(x)$ e $q(x)$ idênticos. Considerando $q(x)$ como um modelo para $p(x)$, pode-se obter a entropia cruzada entre estes, ou seja, o número médio de bits para codificar uma saída de $X$ se, em lugar de $p(x)$, for utilizada uma distribuição dada por $q(x)$.

$$H(X,q) = H(X) + D(p\|q) \tag{2.12}$$

$$= \sum_x p(x) \log_2 \frac{1}{p(x)} + \sum_x p(x) \log_2 \frac{p(x)}{q(x)}$$

$$= -\sum_x p(x) \log_2 q(x)$$

Estendendo esta definição para uma linguagem $\mathfrak{I}$ com distribuição de probabilidades dada por $p(x_1^T)$ e, sendo a distribuição estimada pelo modelo dada por $p_M(x_1^T)$, a entropia cruzada (*cross-entropy*) de $\mathfrak{I}$ em relação ao modelo *M* é descrita por:

$$H(\mathfrak{I},M) = -\lim_{T\to\infty} \frac{1}{T} \sum_{x_1^T \in \mathfrak{I}} p(x_1^T) \log_2 p_M(x_1^T) \tag{2.13}$$

A entropia cruzada $H(\mathfrak{I},M)$ é dada em função do número médio de bits por símbolo necessários para codificar uma linguagem $\mathfrak{I}$, utilizando o modelo *M*. A entropia cruzada pode, portanto, ser aplicada para comparação da acurácia de modelos que competem entre si. Uma vez que $H(\mathfrak{I}) \leq H(\mathfrak{I},M)$, o modelo com a menor entropia cruzada (TEAHAN, 1998) pode ser considerado como o que mais se aproxima da *fonte* da mensagem, que é o melhor modelo de linguagem possível.

É preciso notar que a "real" distribuição dada por $p(x_1^T)$ é desconhecida. Porém, se $\mathfrak{I}$ for modelada como um processo ergódico, com seqüências suficientemente $x_1^T$ longas, com *T* tendendo ao infinito, a entropia cruzada $H(\mathfrak{I},M)$ pode ser aproximada, conforme (TEAHAN, 1998), por meio de:

$$H(\mathfrak{I},M) = -\lim_{T\to\infty} \frac{1}{T} \log_2 p_M(x_1^T) \tag{2.14}$$

Métodos de compressão baseados em modelos de contexto finito podem ser utilizados para a construção de modelos de linguagem, estimando a distribuição probabilística para uma seqüência discreta por meio de $p_M(\cdot)$. No Capítulo 8, o conceito de entropia cruzada é utilizado para comparar modelos de linguagem gerados por uma rede neural artificial no processo de compressão de seqüências discretas, sendo aplicado à classificação de textos.

## 2.3 Modelagem e Codificação

Conceitualmente, algoritmos ou sistemas de compressão sem perda podem ser separados em duas partes: a responsável pela *modelagem* e a que executa a *codificação* (MOFFAT, BELL e WITTEN, 1995). Por meio da modelagem opera-se sobre as regularidades e similaridades observadas nos dados emitidos pela *fonte*, de modo a descrevê-las na forma de um *modelo*. O termo codificação costuma referir-se, em um sentido mais amplo, ao processo de compressão completo. No entanto, de acordo com Bell et al (1989), este se distingue do termo *codificação* que é associado à função exercida pelo *codificador* na produção de um fluxo de bits a partir do *modelo* gerado. A estrutura que une um modelador a um codificador

no processo de compactação e descompactação procura explicitar estes aspectos, sendo descrita por meio da Figura 2.

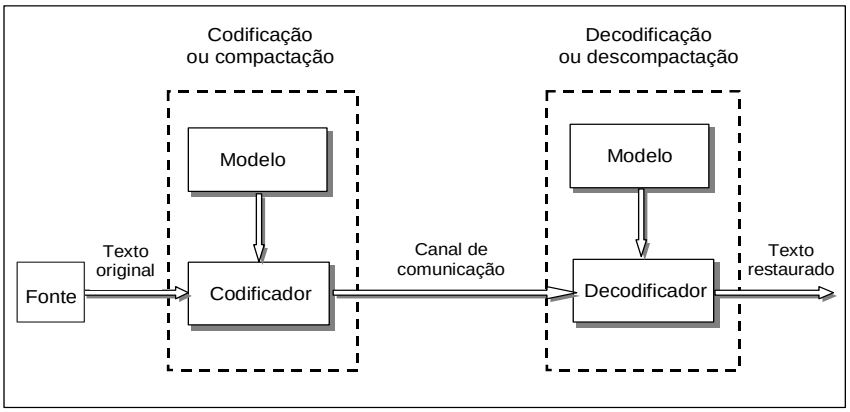



Figura 2: Esquema "modelo e codificador" para compactação e descompactação, baseado em (BELL, CLEARY e WITTEN, 1990; WITTEN, MOFFAT e BELL, 1999)

A separação entre as funções de modelo e de codificador foi proposta, inicialmente, por Rissanen e Langdon, em 1981 (*apud* MOFFAT, BELL e WITTEN, 1995). No entanto, por questões práticas de eficiência na implementação, para muitos métodos não há uma distinção clara entre *modelagem* e *codificação* (MOFFAT, BELL e WITTEN, 1995; BELL, CLEARY e WITTEN, 1990), tal qual a descrita por meio do esquema genérico da Figura 2. Enfatiza-se que, no caso do sistema de compressão de dados sem perda desenvolvido no presente trabalho, a modelagem realizada pela rede neural artificial é claramente separada da codificação realizada por um codificador aritmético.

Na abordagem estatística adaptativa, a cada passo de compactação, o modelo deve estimar a distribuição probabilística sobre todos os símbolos do alfabeto da fonte que possam suceder a seqüência de entrada, sem qualquer conhecimento prévio sobre qual, dentre os símbolos possíveis, será representado. O codificador utiliza esta distribuição probabilística, juntamente com a informação daquele que é, efetivamente, o símbolo subseqüente, para transmitir ao decodificador uma seqüência de bits que representa este símbolo. O modelo só é atualizado com a incorporação do conhecimento sobre o símbolo subseqüente, após este ter sido codificado (ou transmitido). No módulo de decodificação (descompactação) é reconstruído o mesmo modelo, sendo que este precisa prover uma distribuição de probabilidades idêntica à utilizada durante a codificação (compactação). O decodificador extrai e interpreta um número de bits apropriado do fluxo recebido e determina o símbolo correspondente do alfabeto da *fonte* por meio do modelo. Durante a fase de atualização do modelo no descompactador, este símbolo é utilizado para reproduzir fielmente quaisquer mudanças que tenham sido realizadas no modelo do compactador. Assim, o próximo símbolo da seqüência pode ser processado da

mesma maneira, uma vez que codificador e decodificador possuem, a esta altura, modelos idênticos. Assumindo-se que o codificador possa designar uma seqüência ótima de bits, baseado nas probabilidades geradas pelo modelo, quanto mais acurado o modelo, melhor a compressão final da mensagem.

### 2.3.1 Codificação

A codificação consiste na designação de seqüências binárias aos símbolos do alfabeto da *fonte* e uma palavra-código é uma seqüência binária utilizada para representação de um símbolo (ou mensagem) em particular. O código forma o conjunto das palavras-código capazes de representar as saídas possíveis da fonte de mensagem, sendo que as palavras-código podem ter um número fixo ou um número variável de bits para representar cada símbolo. A utilização de palavras-código de tamanho variável permite a geração de códigos mais eficientes, por meio da designação de seqüências binárias mais longas para representar símbolos com menor probabilidade, e seqüências mais curtas para símbolos com maior probabilidade associada.

Para minimizar o espaço requerido pelo código, com o objetivo de maximizar a compressão, este deve ser construído de modo a ter mínima redundância, minimizando seu comprimento médio ao longo das palavras-código, dado por $l$:

$$l = \sum_{i=1}^{m} p_i l_i \tag{2.15}$$

Em que $l_i$ é o comprimento, em bits, da palavra-código designada ao *i-ésimo* símbolo do alfabeto, $p_i$ a sua probabilidade, e $m$ o número de símbolos distintos do alfabeto.

Além de buscar por um código com comprimento médio que possibilita alcançar um valor mínimo de bits por mensagem, a seqüência de palavras-código deve ser *unicamente decodificável*. Para que um código seja unicamente decodificável, de modo que não haja ambigüidade na decodificação, uma palavra-código não deve ser prefixo de outra (MOFFAT, BELL e WITTEN, 1995). Uma condição necessária para que o código seja "livre-de-prefixo" é a de que este satisfaça a inequação de *Kraft* (SAYOOD, 2000):

$$\sum_{i=1}^{m} 2^{-l_i} \leq 1 \tag{2.16}$$

O menor comprimento possível, em bits, para codificar um símbolo, ou seja, o comprimento ideal para a palavra-código que o representa, é dado por sua quantidade de informação $h_i$. A entropia, que corresponde à quantidade média de informação para uma mensagem, estabelece o comprimento médio ideal por palavra-código.

A eficiência de um código é medida por sua redundância *R*, sendo esta dada pela diferença entre o comprimento médio por palavra-código e seu comprimento médio ideal:

$$R = \sum_{i=1}^{m} p_i l_i - \left( \sum_{i=1}^{m} p_i log_2 \frac{1}{p_i} \right) \tag{2.17}$$

A codificação de *Huffman* possibilita a construção de códigos minimamente redundantes por meio de palavras-código de tamanho inteiro. A codificação Aritmética vai além, permitindo a designação de palavras-código de comprimento próximo do ótimo, utilizando frações de bits. Como o problema da construção de um código com um número mínimo de bits (conforme os preceitos da entropia da mensagem) é considerado satisfatoriamente resolvido, é necessária a geração de um bom modelo da mensagem para estimar as probabilidades para cada símbolo a ser codificado.

### 2.3.2 Modelagem

Existem diversas formas por meio das quais o compactador e o descompactador podem manter o mesmo modelo em um processo de compressão sem perda. Em (WITTEN, MOFFAT e BELL, 1999) são destacados três tipos de modelagem: *estática*, *adaptativa* e *semi-estática* (esta última, referida como "semi-adaptativa" em (BELL et al, 1990)). Além das três formas descritas em (WITTEN, MOFFAT e BELL, 1999), aqui é acrescentada a designação de modelagem *semi-adaptativa* para o caso em que, durante a compressão, um modelo pré-treinado é atualizado, adaptando-se à seqüência que está sendo compactada.

- **Modelagem Estática**. O modelo para compactação é fixo, independentemente da mensagem a ser codificada. Através de um treinamento *off-line* (utilizando uma amostra representativa de uma classe de fonte de mensagem) é construído um modelo que é mantido estático para a compactação das seqüências de texto futuras. Como os parâmetros do modelo não se ajustam à seqüência em curso, será prejudicada a taxa de compressão para um texto cuja distribuição probabilística dos símbolos desvie muito do modelo estático definido.
- **Modelagem Semi-estática**. O método de compressão cria um modelo baseado na mensagem a ser codificada, sendo que este modelo precisa ser transmitido ao decodificador, precedendo o código gerado. Em uma passada preliminar pela seqüência é construído o modelo em relação ao qual esta será codificada em uma segunda passada. O decodificador precisará, portanto, receber o modelo seguido da mensagem codificada para poder decodificá-la em relação a este. Este tipo de modelagem possui a vantagem de

utilizar um modelo que estará bem talhado ao texto a ser codificado, mas requer duas passadas para codificação e precisa contabilizar o tamanho do modelo no arquivo compactado.

- **Modelagem Adaptativa**. O modelo é gerado adaptativamente, com base na mensagem que está sendo compactada. O modelo construído durante o processamento da seqüência é utilizado para codificar e transmitir uma mensagem que será decodificada por meio do mesmo modelo reconstruído de modo idêntico no descompactador. Na modelagem adaptativa estatística, apenas o conhecimento da porção já processada da seqüência é utilizado na estimação probabilística para os símbolos que a sucedem para realizar sua codificação. Partindo de estados iniciais iguais, os modelos do compactador e descompactador serão atualizados por meio do mesmo algoritmo, mantendo-se sincronizados durante o processo (contanto que não ocorram erros de transmissão). Como os parâmetros do modelo adaptam-se às características mutáveis dos dados, de acordo com a seqüência, este se ajustará dinamicamente ao texto que está sendo compactado. Assim, a taxa de compressão tenderá a melhorar ao longo do processo, à medida que o modelo se torne mais representativo do texto em compactação. Como não há transmissão do modelo, é requerida uma única passada pela seqüência para realizar a codificação, mas o processo tende a ser computacionalmente mais dispendioso.
- **Modelagem semi-adaptativa**. Corresponde ao treinamento prévio de um modelo, utilizado posteriormente para uma compressão adaptativa. O modelo, uma vez treinado, é instalado no emissor (ou compactador) e no receptor (ou descompactador) e as seqüências de texto são compactadas e descompactadas seguindo o mesmo algoritmo, sem a necessidade de envio do modelo. Neste caso, o modelo gerado por meio de um conjunto de treino sofre ajustes que devem ser reproduzidos no descompactador, para cada exemplar em compressão. A diferença em relação à modelagem completamente adaptativa é a de que o modelo treinado já incorpora algum conhecimento inicial. No entanto, há necessidade de treinamento por meio de um conjunto suficientemente representativo da classe de textos que serão futuramente compactados ou terão sua quantidade de informação medida em relação a este. Este conhecimento *a priori*, incorporado ao modelo, favorece melhores taxas de compressão para textos cuja distribuição probabilística não desvie muito do modelo inicial. No entanto, caso os textos a serem compactados tenham uma estrutura estatística muito diferente em relação ao modelo treinado, a compressão tende a ser prejudicada. Por exemplo, um modelo treinado por meio de textos literários no

idioma inglês, propiciaria melhores taxas de compressão para textos neste mesmo idioma do que as atingidas por textos em português compactados em relação a este modelo.

### 2.3.3 Modelos de Contexto Finito

Por meio da construção de um modelo aproximado da estrutura da *fonte* é almejada uma diminuição na quantidade de bits utilizados para codificar uma mensagem. Em (BELL, CLEARY e WITTEN, 1990) são descritos diferentes tipos de modelos que podem ser adotados, tais como: modelos baseados em *grammars*, modelos ergódicos, modelos de estado finito (baseados em máquinas de estado finito) e modelos de contexto finito (que utilizam a suposição de que a *fonte* de informação segue um processo de *Markov*). Nesta seção serão enfocados apenas modelos de contexto finito para compressão e o conceito de modelos de linguagem baseados em *n-gramas*.

Modelos de contexto finito baseiam a previsão do símbolo em uma subseqüência formada por *r* símbolos consecutivos que o precedem imediatamente em uma seqüência. Esta previsão corresponde à estimação da distribuição probabilística para os símbolos sucessores possíveis, baseada na freqüência com que estes apareceram logo após o determinado contexto de *ordem r*, ao longo da seqüência.

Conjuntos de *n* símbolos consecutivos que aparecem em um determinado texto definem *n-gramas* (sendo *n* um número inteiro maior que 0). Modelos baseados em *n-gramas*, em que *n* pode corresponder a uma unidade representada por caracter, palavra ou segmentos de fala, dependendo do tipo de aplicação, são amplamente utilizados para construção de modelos de linguagem. As freqüências de *n-gramas* podem ser usadas para construir modelos de contexto finito em que os *n*-1 primeiros símbolos do *n-grama* são utilizados para prever o *n-ésimo* símbolo subseqüente (BELL, CLEARY e WITTEN, 1990). Modelos de *contexto finito* de ordem *r* (em que *r*=*n*-1) utilizam a suposição de que a seqüência segue um modelo de *Markov* de *ordem r*, de modo que, para estimar a probabilidade para cada símbolo da seqüência, é utilizado um contexto formado pelos *r* símbolos que a precedem.

Seja a seqüência discreta $x_1^T = x_1, x_2, ..., x_T$, em que símbolos $x_t$, $t \leq T$, assumem valores de um alfabeto finito $\Omega=\{\varpi_1, \varpi_2, ..., \varpi_m\}$, contendo $|\Omega|=m$ símbolos distintos. Como visto na Seção 2.2.1.1, o menor comprimento possível, em bits, para o código que representa a seqüência observada $x_1^T$ é dado por meio de $-log_2 p\left(x_1^T\right)$, com $p(x_1^T)=p(x_1, x_2,\ldots, x_T)$. Um modelo de linguagem deve estimar $p(x_1^T)$, sendo que a probabilidade para um símbolo $x_t$ é condicionada pela seqüência antecessora, ou *história* $(x_1, x_2,\ldots, x_{t-1})$. No entanto, é

excessivamente dispendioso construir um modelo probabilístico com base na história passada completa. Portanto, por meio de um modelo de *Markov* de *ordem r* (sendo *r* um valor inteiro e finito, $r<t$), assume-se que, para estimar a probabilidade para o símbolo $x_t$, o conhecimento dos *r* símbolos consecutivos que precedem $x_t$ na seqüência é equivalente ao conhecimento de toda a *história* passada $x_1, x_2,\ldots, x_{t-1}$, ou seja:

$$p(x_t \mid x_1, x_2, ..., x_{t-1}) = p(x_t \mid x_{t-r}, .., x_{t-1})$$

Utilizando-se um modelo de *Markov* de *ordem r*, $p(x_1^T)$ pode ser aproximada por:

$$p\left(x_1^T\right) = \prod_{t=1}^{T} p(x_t \mid x_{t-r}, ..., x_{t-1}) \tag{2.18}$$

Para o estado inicial do processo (em que $t \leq r$) deve ser definido um tratamento específico para estimação de $x_t$. Assim, por exemplo, se $r$=2, para estimar a probabilidade para $x_1$ é utilizada a seqüência ($x_{-1}$ $x_0$) de símbolos arbitrados e, para $x_2$, utiliza-se ($x_0$ $x_1$), em que $x_0$ é um símbolo arbitrado e $x_1$ é um símbolo observado.

Para a compressão de *texto on-line* é construído um modelo preditor para estimação probabilística dos símbolos que podem ser assumidos por $x_t$, com base no contexto *s* do conjunto S de contextos possíveis. O modelo para distribuição de probabilidades condicionais $p(\varpi|s)$ de que um símbolo $\varpi \in \Omega$ sucederá um determinado contexto finito $s \in$ S de comprimento *r* pode ser incrementalmente construído com base na sua freqüência de ocorrência ao longo do texto que está sendo processado. Muito embora contextos de maior comprimento possam propiciar previsões mais acuradas, dada sua maior especificidade, há algumas restrições. Primeiramente, a utilização de contextos mais longos aumenta a incidência de subseqüências (*n*-gramas) que aparecem poucas vezes ou mesmo uma única vez no texto. Além disso, como é necessário estimar a probabilidade para cada contexto, e o número de contextos possíveis cresce exponencialmente com a ordem do modelo, torna-se dispendiosa a memória necessária para armazená-los.

### 2.3.4 Estimação das Probabilidades por um Modelo

Durante a fase de *estimação da distribuição probabilística* no processo de compressão, o modelo preditor estatístico construído é utilizado para que seja codificado o símbolo corrente da seqüência em compactação. Durante a denominada fase de *atualização do modelo*, são ajustados os parâmetros do modelo, modificando-o para refletir as mudanças nas entradas recebidas. A atualização do modelo pode ser realizada simplesmente por meio do incremento de um contador que grava a ocorrência dos símbolos ocorridos e estima as probabilidades

com base nestas freqüências. Outra abordagem de atualização envolve, além do ajuste das probabilidades associadas a classes condicionantes que representam contextos, o ajuste da estrutura do modelo e a criação incremental de novas classes condicionantes sobre as quais poderão basear-se as estimativas probabilísticas futuras (MOFFAT, BELL, WITTEN, 1995).

Para estimar a probabilidade de um símbolo $x_t$, dado que o contexto antecessor é denotado por $x_{t-n+1}^{t-1}=x_{t-n+1},\ldots, x_{t-1}$, computa-se a freqüência com que o *n-grama* $x_{t-n+1}^{t}=x_{t-n+1},\ldots, x_t$ ocorre, dada por $c(x_{t-n+1}^{t})$, em relação à freqüência do contexto $c(x_{t-n+1}^{t-1})$:

$$p\left(x_t \middle| x_{t-n+1}^{t-1}\right)=\frac{c\left(x_{t-n+1}^{t}\right)}{c\left(x_{t-n+1}^{t-1}\right)} \tag{2.19}$$

Esta abordagem para a geração de modelos de linguagem precisa, porém, lidar com o problema de *n-gramas* que não ocorrem no conjunto utilizado para geração do modelo e que teriam sua probabilidade designada como *zero* durante o processo. Este é o denominado de *problema da freqüência zero*, tratado a seguir.

#### 2.3.4.1 O Problema da "Freqüência zero"

Durante o processo de estimação da distribuição probabilística *on-line*, também denominada de adaptativa, depara-se com o problema da designação de probabilidade a um evento ainda não observado até aquele momento do processo. Como as probabilidades são estimadas no decorrer do processo de compressão, só existe a necessidade de armazenar os contextos que já apareceram na seqüência que está sendo codificada. Porém, ao longo da compressão seqüencial (em uma única passada pela seqüência) é preciso codificar símbolos que ainda não ocorreram em um determinado contexto e que, com uma freqüência igual a zero, receberiam uma designação de probabilidade igual zero. Para os métodos estatísticos de compressão é preciso sempre designar alguma probabilidade para o símbolo a ser codificado. Os modelos baseados em contexto finito, construídos durante uma compressão de texto *on-line* devem, portanto, lidar com o denominado problema da *freqüência zero* (CLEARY, TEAHAN, 1995), reservando sempre certa probabilidade para um evento novo. Conforme Bell, Cleary e Witten (1990), não existe, ao que tudo indica, uma maneira racional, única ou mais adequada de designação de probabilidade a um símbolo nestas condições. Em (WITTEN, MOFFAT e BELL, 1999) são apresentadas algumas formas de suavização no cálculo para estimação de probabilidades, incorporadas ao método PPM (Seção 2.4.2) que implementa o denominado *tratamento de escape*.

## 2.4 Métodos de Compressão de Texto

A necessidade de armazenar ou enviar uma mensagem por meio de uma representação que ocupe um espaço menor e seja reversível à sua forma original deu origem a inúmeros métodos de compactação. Podem ser citadas técnicas simples, tais como *run-length encoding* e *move-to-front* (BELL, CLEARY e WITTEN, 1990), até outras mais elaboradas e que possibilitam melhores taxas de compressão, como as que serão vistas a seguir.

A codificação de *Huffman* (SAYOOD, 2000; WITTEN, MOFFAT e BELL, 1999), concebida por David Huffman no início da década de 50, foi tida por muito tempo como uma das melhores técnicas de compressão. Na década de 70, a compressão *Lempel-Ziv*[6] (ZIV e LEMPEL, 1977; ZIV e LEMPEL, 1978) e o conceito moderno da Codificação Aritmética (Rissanen e Langdon, 1979 *apud* SAYOOD, 2000) abriram novas perspectivas para melhores taxas de compactação. Ambas são baseadas na idéia de compressão adaptativa, em que as entradas são compactadas, via codificação dinâmica, em relação a um modelo construído a partir do texto que está sendo compactado (WITTEN, MOFFAT e BELL, 1999). Baseando o modelo em dados que estão sendo vistos durante o processo, métodos adaptativos são capazes de codificar em uma única passada pela seqüência.

A maior parte das técnicas de compressão de texto pode ser dividida, genericamente, em: métodos *baseados em dicionário* e métodos *baseados em símbolos* (WITTEN, MOFFAT e BELL, 1999). Métodos baseados em dicionário constroem um *dicionário* a partir das estruturas redundantes existentes na sucessão de cadeias de caracteres (*strings*) que compõem o texto. Em termos gerais, estes métodos substituem *strings* por um índice ou ponteiro às ocorrências anteriores destas, sendo estas representadas por uma entrada que as identifica em um dicionário (livro-código). A compressão é obtida por meio da representação de muitos símbolos por uma palavra-código de saída, de modo que esta necessite de menos bits para representação do que os símbolos que referencia. Métodos baseados em *Lempel-Ziv*, estão entre os exemplos de métodos baseados em dicionário mais conhecidos e amplamente adotados.

Os métodos *baseados em símbolos* codificam um símbolo a cada vez, designando uma palavra-código baseada na probabilidade estimada para a ocorrência deste. O desempenho de compressão depende da geração de boas estimativas probabilísticas para o símbolo a ser codificado, o que resultará em um código de comprimento menor como saída. Assim, símbolos mais previsíveis terão associados a si uma palavra-código mais curta e símbolos

[6] *Lempel-Ziv* (LZ), devido à inversão histórica na ordem de apresentação dos autores.

menos prováveis, uma palavra-código mais longa. Como dependem de estimações estatísticas acuradas, métodos baseados em *símbolos* são também denominados de *métodos estatísticos* (BELL, CLEARY e WITTEN, 1990). Esta abordagem estatística é, comumente, baseada na utilização de um modelo para estimação das probabilidades e na codificação por meio do método de *Huffman* ou da Codificação Aritmética.

Além dos métodos *Lempel-Ziv* e PPM, aqui apresentados, diversos outros podem ser encontrados em (BELL, CLEARY e WITTEN, 1990; WITTEN, MOFFAT e BELL, 1999; SAYOOD, 2000), tais como: o método DMC (*Dynamic Markov Compression*), que utiliza um modelo baseado em máquina de estado finito; a compressão baseada em *block sorting*, também conhecida por BWT (*Burrows-Wheeler Transform*), que aplica uma transformação sobre o texto, seguido da compactação por meio de *run-length encoding*; métodos que utilizam *palavras* como unidades a serem compactadas, e o método PPM, baseado em contexto finito, que será visto com mais detalhes na Seção 2.4.2.

## 2.4.1 Métodos *Lempel-Ziv*

Desenvolvidas por Jacob Ziv e Abraham Lempel (ZIV e LEMPEL, 1977; ZIV e LEMPEL, 1978), as técnicas *LZ*77 e *LZ*78 consistem em duas abordagens diferentes para a construção de *dicionários* e que deram origem a muitas variantes que formam a família dos métodos *LZ*. Existem muitas variações que diferem na abordagem de análise da seqüência e na codificação dos símbolos, assim como nas técnicas de atualização do dicionário. No entanto, a característica marcante de todos estes é a velocidade de processamento, especialmente nas versões otimizadas, o que os torna muito populares em aplicações correntes, em computadores pessoais, etc. Em (BELL, CLEARY e WITTEN, 1990) são referenciados 12 métodos baseados em *Lempel-Ziv*, designados pelo prefixo *LZ*, sendo que aqui serão descritos, brevemente, apenas os métodos *LZ*77, *LZ*78 e *LZW*.

### 2.4.1.1 LZ77

O método *LZ*77 (ZIV e LEMPEL, 1977) pode ser descrito de maneira simplificada, como segue. A seqüência de entrada é examinada pelo codificador através de uma janela deslizante que consiste de duas partes: "o *buffer* de *busca*"[7], designado por *bb*, contendo uma porção da seqüência codificada recentemente, e o "*buffer* de inspeção à *frente*"[8], designado por *bf*, que contém a próxima porção da seqüência a ser codificada. Esta abordagem utiliza o passado

[7] Do original: *Search buffer*.
[8] Do original: *Look-ahead buffer*.

mais recente da seqüência como um dicionário, com uma janela de busca de comprimento fixo e retroativo em relação ao início da janela de inspeção à frente.

Para codificar uma seqüência de entrada no *bf*, o codificador desloca o ponteiro de busca retroativamente, através do *bb*, até encontrar uma equivalência com primeiro símbolo de *bf*. Partindo desta posição, são examinados, à frente, os símbolos consecutivos para detectar uma possível *string* igual a uma contida em *bf*. O codificador busca pela *string* igual mais longa que, uma vez encontrada, é codificada por meio da *tripla* <*d*, *l*, *code*>. O deslocamento *d* designa a distância do ponteiro em relação ao início da janela do *bf*, em função da posição em que é encontrada, em *bb*, uma equivalência com a *string* a ser codificada,. O comprimento *l* designa o número de símbolos consecutivos de *bb* que formam uma *string* em *bf*, a partir do primeiro símbolo. A palavra-código *code* é correspondente ao símbolo que sucede a *string* igual encontrada no *bf*. Para ilustrar este procedimento, considera-se a seqüência:

**... m x z y z v y x w y z v y v v y v v y w ...**

A Figura 3 apresenta o procedimento de janela deslizante no processo de codificação. Neste exemplo simples, são definidos: *bb*=8 e *bf*=6. A janela do *bf* avança *l*+1 posições, conforme são codificadas *strings* de comprimento *l*. Sendo "**m x z y z v y x**" a porção da seqüência já codificada, tem-se, no estado atual, "**w y z v y v**" na janela do *bf*. (A janela do *bb* é representada pelas linhas cheias e a janela do *bf* pelas linhas tracejadas). As Tabelas 1 e 2, a seguir, descrevem o esquema de codificação e de decodificação, respectivamente, para a dada seqüência.

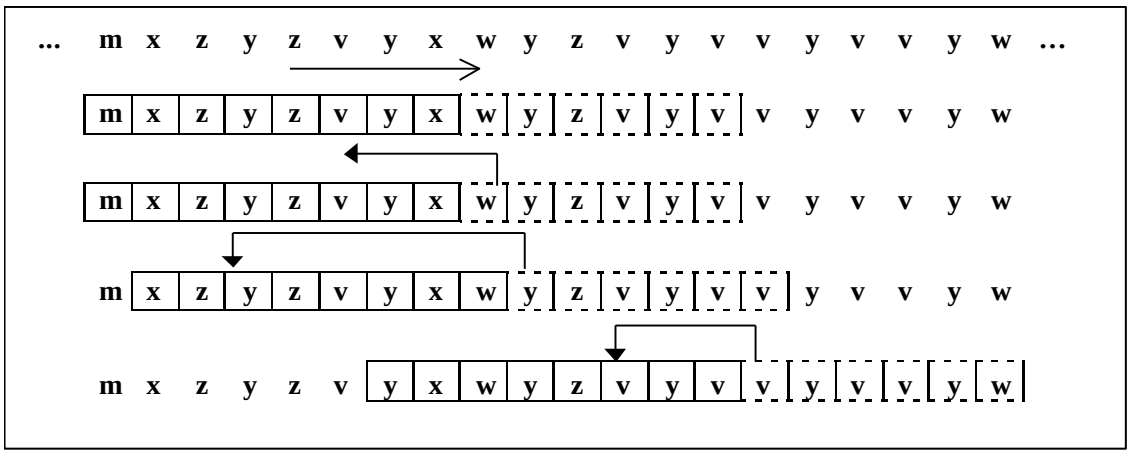



Figura 3: Janela Deslizante para LZ77

Tabela 1: Exemplo de Codificação para LZ77

| Codificação | | | | |
|---|---|---|---|---|
| *buffer* de busca | *buffer* de inspeção à frente | *string* | *tripla* transmitida | Comentários |
| m x z y z v y x | w y z v y v | w | <0, 0, *code*(w)> | Não existe equivalência com o símbolo w**,** portanto, *d* = 0, *l* = 0 e c(w) é o código para o símbolo w. |
| x z y z v y x w | y z v y v v | y z v y | <6, 4, *code*(v)> | Busca pela *string* mais longa, percorrendo retroativamente a janela de *bb* até *d*=6. Após a codificação, a janela de *bf* desliza 5 posições à frente dentro da janela de *bf*. |
| y x w y z v y v | v y v v y w | v y v v y | <3, 5, *code*(w)> | O ponteiro retroage 3 posições no *bb* e avança 2 posições dentro de *bf*. |

Tabela 2: Exemplo de Decodificação para LZ77

| Decodificação | | |
|---|---|---|
| *string* já decodificada | *tripla* decodificada | Comentários |
| m x z y z v y x | <0, 0, *code* (w)> | Não existe equivalência dentro da *string* já decodificada e o próximo símbolo é w. |
| x z y z v y x w | <6, 4, *code* (v)> | O decodificador move o ponteiro de cópia retroativamente 6 símbolos e, a partir deste ponto, copia 4 símbolos à frente, sendo estes relativos *string* y z v y. |
| y x w y z v y v | <3, 5, *code* (w)> | O decodificacor move o ponteiro de cópia retroativamente 3 posições e copia 3 símbolos à frente, sendo estes: v y v. A partir daí, mais 2 símbolos são copiados, sendo estes: v y, formando a *string* v y v v y. |

A maior parte dos aprimoramentos sobre *LZ77* é baseada na codificação eficiente das triplas por meio de códigos de comprimento variável. Estes códigos podem ser adaptativos ou mesmo semi-adaptativos, caso sejam utilizadas duas passadas pelo texto. Pacotes de compressão como, PKZip, Winzip, Zip, Lharc, gzip e ARJ, utilizam um algoritmo baseado em *LZ*77 sucedido de um codificador de comprimento variável (SAYOOD, 2000).

O *software gzip* consiste em uma variação altamente otimizada do método *LZ*77. A saída no *gzip*, compreendendo os ponteiros aos fragmentos anteriores de texto, é codificada por meio da codificação de *Huffman*. O usuário pode regular o grau de compressão em função da velocidade. O modo *gzip-f* (*Gzip Fast*) propicia uma compressão rápida, limitando a intensidade de busca por *strings* iguais, enquanto o modo *gzip-b*, (*Gzip Best*) alcança melhor compressão por meio de uma busca mais extensa por ocorrências anteriores de *strings*.

### 2.4.1.2 LZ78

O algoritmo *LZ*78 (ZIV e LEMPEL, 1978) constrói dinamicamente um dicionário das frases anteriormente vistas na seqüência, em lugar de utilizar uma janela do texto já visto. Para isto é utilizada uma estratégia de *parsing*, em que as *strings* analisadas na porção do texto que antecede o ponto atual da codificação têm uma referência de entrada no dicionário. Quando uma ocorrência repetida da *string* é encontrada no texto, esta é codificada por meio da sua referência. Os símbolos a serem codificados são representados por <*i, code*>, sendo *i* um índice correspondente à *string* mais longa analisada que se iguala à entrada, e *code* designa o código para o símbolo que sucede a porção igualada da entrada. O dicionário deve ser construído de maneira idêntica no decodificador.

### 2.4.1.3 LZW

O *LZW* consiste em uma modificação do algoritmo *LZ*78, realizada por Terry Welch (WELCH, 1984), voltada para implementação em *hardware* para controladores de disco de alto desempenho. Por meio de uma técnica que elimina a necessidade de codificação de *code* do par <*i, code*>, o codificador envia apenas o índice da ocorrência anterior da *string*. Para isso, o dicionário precisa incorporar, inicialmente, todos os símbolos do alfabeto de entrada.

As primeiras referências do dicionário, relativas aos primeiros códigos 0-255 (utilizando símbolos de 8 bits), são correspondentes aos caracteres do alfabeto de entrada, sendo que os códigos 256-4095 são designados às *strings* durante o processo. Cada vez que um novo código é gerado, significa que uma nova *string* foi analisada. Novas *strings*, resultado da junção da entrada com uma *string* existente, são incluídas no dicionário, juntamente com seus respectivos novos códigos. Um exemplo bastante simples para o método *LZW*, adaptado de (NELSON, 1989) e descrito a seguir, ilustra, por meio das Tabelas 3 e 4, o processo de codificação e decodificação de uma seqüência.

Dada a *string* de entrada **+xyz+xy+xyy+xyw** é feita, inicialmente, uma busca pela *string* "**+x**" na tabela. Como "**+x**" corresponde à primeira *string* nova, esta é adicionada à tabela com a designação do código "256" e o codificador fornece, como saída, o código relativo a "**+"**. A entrada seguinte é o símbolo seguinte "**y**" e, como a *string* "**xy**" não existe na tabela, esta é adicionada com a designação do código "257", e o código relativo a "**x**" é dado como saída. Prossegue-se até que a primeira igualdade é encontrada, com a leitura dos símbolos "**+**" e "**x**" que correspondem à string "**+x**". Assim, o código "256", relativo a "**+x**", é dado como saída e a *string* "**+xy**" é adicionada à tabela. O processo continua até o fim da *string*, com a recepção do símbolo EOF (fim de arquivo) e a emissão de todos os códigos.

Tabela 3: Exemplo de Codificação para LZW

| Codificação da seqüência: **+xyz+xy+xyy+xyw** | | | |
|---|---|---|---|
| entrada de símbolo | saída de código | nova *string* | novo valor de código |
| +x | + | +x | 256 |
| y | x | xy | 257 |
| z | y | yz | 258 |
| + | z | z+ | 259 |
| xy | 256 | +xy | 260 |
| + | y | y+ | 261 |
| xyy | 260 | +xyy | 262 |
| +x | 261 | y+x | 263 |
| yw | 257 | xyw | 264 |
| EOF | w | | |

Tabela 4: Exemplo de Decodificação para LZW

| Decodificação do código: **+ x y z 256 y 260 261 257 w** | | | |
|---|---|---|---|
| código de entrada | *String* de saída | nova *string* | novo valor do código |
| + | + | +x | 256 |
| x | x | xy | 257 |
| y | y | yz | 258 |
| z | z | z+ | 259 |
| 256 | +x | +xy | 260 |
| y | y | y+ | 261 |
| 260 | +xy | +xyy | 262 |
| 261 | y+ | y+x | 263 |
| 257 | xy | xyw | 264 |
| w | w | | |

O decodificador constrói o seu dicionário durante o processo de decodificação, descrito por meio da Tabela 4. Ao receber o fluxo de códigos, utiliza-os para reconstruir os dados de entrada originais, adicionando uma nova *string* ao dicionário a cada vez que lê um código novo. Cada código de entrada recebido é traduzido em uma *string* dada como saída. Ao final do processo, codificador e decodificador possuem dicionários idênticos sem que estes tenham que ser transmitidos explicitamente durante o processo.

O exemplo simples apresentado procura enfatizar a representação das substituições de *strings* por códigos de modo que a compressão é obtida quando um único código é emitido como saída em lugar de uma *string* de símbolos. No exemplo apresentado, os 15 símbolos da *string* de entrada puderam ser codificados por meio de 4 substituições e 6 símbolos. Além disso, o método LZW deve conter um mecanismo para lidar com casos em que deve ser decodificado um ponteiro que não possui uma *string* correspondente completa no dicionário

do decodificador. Considerações relativas a implementações eficientes podem ser encontradas em (NELSON, 1989).

Algumas das aplicações mais conhecidas para o LZW são: o GIF (*Graphics Interchange Format*) para compressão de imagens, a recomendação CCITT V.42bis, padrão de compressão via *modems*. Além destas, há o programa *compress* do UNIX, que iniciou como uma implementação do LZW e foi modificado sucessivamente para atingir melhor desempenho e maior velocidade de compactação.

### 2.4.2 Métodos PPM

Os métodos PPM (*Prediction by Partial Matching*) (CLEARY e WITTEN, 1984; (WITTEN, MOFFAT, BELL, 1999) realizam uma modelagem estatística, baseada na predição da distribuição probabilística do símbolo que sucede um determinado contexto em uma seqüência. Com base em um contexto de entrada, formado caracteres que antecedem o símbolo a ser previsto, a informação estatística extraída do arquivo a ser compactado é utilizada para estimar a distribuição probabilística de ocorrência deste símbolo. Para codificá-lo, a distribuição é utilizada por um codificador estatístico, geralmente, baseado em Codificação Aritmética (Seção 2.4.3.1). A probabilidade associada a cada símbolo é aproximada pela proporção de vezes em que este ocorre após um contexto específico de comprimento *r*. As probabilidades para os símbolos que sucedem um contexto são atualizadas adaptativamente durante a compressão da seqüência.

O PPM utiliza vários modelos de contexto-finito de diferentes ordens que são agregados para realizar a predição do próximo símbolo e codificá-lo. Inicia-se com um modelo de ordem *r*, com base no qual é realizada a previsão do próximo símbolo e, se este ainda não tiver sido visto neste comprimento de contexto, utiliza-se um modelo de ordem menor. Esta situação caracteriza o mecanismo por meio do qual o PPM comuta para um modelo de ordem menor, utilizando uma probabilidade de escape para codificar um símbolo ainda não visto naquele contexto. O algoritmo comuta, se necessário, até $r$-$r^*$, em que $r^*$ é um número inteiro positivo e $r^* \leq r$. Para $r^*=r$ é utilizado um modelo de *ordem 0*, em que a probabilidade para cada caracter é independente de quaisquer caracteres precedentes, baseando-se apenas na freqüência com que o caracter foi observado na seqüência. Para o caso de caracteres novos que não tenham sido vistos anteriormente em qualquer ordem de modelo, utiliza-se um modelo de contexto de ordem $r$=-1, em que todos os caracteres são equiprováveis. Cada vez que ocorre a comutação de um modelo de ordem maior para um de ordem imediatamente

menor, o codificador envia um caracter de escape, que indica ao decodificador a utilização da mesma ordem de modelo para decodificar o caracter.

A necessidade de designação de uma probabilidade para um caracter ainda não observado na seqüência, até aquele dado ponto do processamento, ocorre mais freqüentemente durante o processo inicial da estimação probabilística adaptativa. Em (WITTEN, MOFFAT e BELL, 1999) são apresentadas algumas formas de solução para este problema por meio de um tratamento especial para um caracter com probabilidade igual a zero. O tratamento para o caracter de escape e o cálculo de sua probabilidade é parte integrante do método PPM descrito na Seção 2.4.2.1, por meio de um exemplo que ilustra o processo.

As diferenças existentes entre os diversos tipos de métodos PPM recaem sobre o modo de operação e designação de probabilidades ao caracter de escape e na maneira de agregar as estimativas de probabilidades dos modelos de diversas ordens, realizando o denominado *blending* para obtenção da estimativa total. Várias formas para estimação da probabilidade de escape foram propostas, dentre as quais, os métodos de escape “A” “B”, “C” descritos em (BELL, CLEARY e WITTEN, 1990) e utilizados, respectivamente, por PPMA, PPMB, PPMC, e o método “D”, aplicado no PPMD (WITTEN, MOFFAT e BELL, 1999). Historicamente, o PPM foi introduzido com utilização de um contexto máximo formado por 3 caracteres, mas versões mais recentes, com contexto de tamanho máximo igual a 5 ou 7, e mesmo versões sem limite para o tamanho máximo do contexto, como o PPM* (WITTEN, MOFFAT e BELL, 1999), atingem melhores taxas de compactação. É preciso notar que, apesar de modelos de maior ordem possibilitarem previsões mais acuradas, na prática, modelos com contextos acima de 7 caracteres não resultam em melhora substancial de compressão em relação à obtida por contextos formados por 5 caracteres

Segundo Witten, Moffat e Bell (1999, p.100), a técnica PPM foi primeiramente apresentada por Cleary e Witten, em 1984, tendo sido refinada por Moffat que, em 1990, apresentou uma implementação para o PPMC, e por outros autores. Atualmente, os métodos PPM estão entre os compactadores que atingem as melhores taxas de compressão. No entanto, os algoritmos baseados em Lempel-Ziv (LZ) têm-se mantido mais populares por uma série de razões, dentre as quais: padronização, maior velocidade e menor requisição de memória durante o processo de compactação e descompactação.

#### 2.4.2.1 PPMC

Apresenta-se aqui, um exemplo simples de processamento de uma seqüência de texto por meio de um PPMC (PPM com cálculo do caracter de escape *esc*, baseado no *método C*). Seja

$c(s^r, \varpi)$ a freqüência com que um dado caracter $\varpi_i$ do alfabeto Ω, para $\Omega=\{\varpi_i: i=1, 2,..., m\}$ sucedeu um determinado contexto $s^r$ de ordem $r$ na seqüência, e $d(s^r)$ o número de caracteres $\varpi$ distintos em $(s^r, \varpi)$. A probabilidade $p'(\varpi \mid s^r)$ designada pelo modelo de contexto finito de ordem $r$ para cada caracter $\varpi$ como sucessor do contexto $s^r$ é dada por:

$$p'(\varpi \mid s^r) = \frac{c(s^r, \varpi)}{c(s^r) + d(s^r)} \tag{2.20}$$

em que, $c(s^r)$ representa o total de vezes que um determinado contexto $s^r$ foi visto na seqüência.

A probabilidade $p(esc)$ para o caracter de escape *esc* é dada por:

$$p(esc) = \frac{d(s^r)}{c(s^r) + d(s^r)} \tag{2.21}$$

Seja, por exemplo, a seqüência **x y z x v m x y w x y z x** já processada até o ponto em que deve ser estimada a distribuição probabilística para o símbolo imediatamente sucessor, por meio de um PPMC com ordem máxima $r$=2 e |Ω|=256. A Tabela 5 esquematiza todos os contextos ocorridos em cada modelo de ordem $r$ até aquele ponto do processo, assim como sua ocorrência na seqüência e suas previsões associadas. O contador de escapes $c(esc)$ equivale ao número de caracteres distintos vistos em um determinado contexto de ordem $r$. Os contadores $c(\varpi)$ e $c(esc)$ são designados por $c(\cdot)$ e as probabilidades $p(\varpi)$ e $p(esc)$ designadas por $p(\cdot)$. O modelo $r$=2 baseia suas previsões nas probabilidades condicionadas por uma subseqüência formada por 2 caracteres antecedentes e o modelo $r$=1 as baseia na subseqüência formada por 1 caracter antecedente. O modelo $r$=0 registra apenas a ocorrência de cada caracter até aquele ponto da seqüência, baseando suas previsões em probabilidades independentes. O modelo $r$= -1 designa probabilidades iguais para todos os caracteres do alfabeto *A*, inicialmente dada por 1/|Ω | denota o número de caracteres do alfabeto.

Tabela 5: Modelos de ordem *r* em um PPMC para seqüência já processada:
**x y z x v m x y w x y z x**

| Ordem r = 2 | | | | | Ordem r = 1 | | | | | Ordem r = 0 | | | | Ordem r = -1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| previsões | | | | | previsões | | | | | previsões | | | | previsões | | |
| s | | $\varpi$ | *c* (.) | *p* (.) | s | | $\varpi$ | *c* (.) | *p* (.) | $\varpi$ | | *c* (.) | *p* (.) | $\varpi$ | *c* (.) | *p* (.) |
| xy | → | z | 2 | 2/5 | x | → | y | 3 | 3/6 | → | x | 5 | 5/19 | → Ω | 1 | 1/\| Ω |
| | → | w | 1 | 1/5 | | → | v | 1 | 1/6 | → | y | 3 | 3/19 | | | |
| | → | esc | 2 | 2/5 | | → | esc | 2 | 2/6 | → | z | 2 | 2/19 | | | |
| | | | | | | | | | | → | v | 1 | 1/19 | | | |
| yz | → | x | 2 | 2/3 | y | → | z | 2 | 2/5 | → | w | 1 | 1/19 | | | |
| | → | esc | 1 | 1/3 | | → | w | 1 | 1/5 | → | m | 1 | 1/19 | | | |
| | | | | | | → | esc | 2 | 2/5 | → | esc | 6 | 6/19 | | | |
| zx | → | v | 1 | 1/2 | | | | | | | | | | | | |
| | → | esc | 1 | 1/2 | z | → | x | 2 | 2/3 | | | | | | | |
| | | | | | | → | esc | 1 | 1/3 | | | | | | | |
| xv | → | m | 1 | 1/2 | | | | | | | | | | | | |
| | → | esc | 1 | 1/2 | v | → | m | 1 | 1/2 | | | | | | | |
| | | | | | | → | esc | 1 | 1/2 | | | | | | | |
| vm | → | x | 1 | 1/2 | | | | | | | | | | | | |
| | → | esc | 1 | 1/2 | m | → | x | 1 | 1/2 | | | | | | | |
| | | | | | | → | esc | 1 | 1/2 | | | | | | | |
| mx | → | y | 1 | 1/2 | | | | | | | | | | | | |
| | | esc | 1 | 1/2 | w | → | x | 1 | 1/2 | | | | | | | |
| | | | | | | → | esc | 1 | 1/2 | | | | | | | |
| yw | → | x | 1 | 1/2 | | | | | | | | | | | | |
| | → | esc | 1 | 1/2 | | | | | | | | | | | | |
| wx | → | y | 1 | 1/2 | | | | | | | | | | | | |
| | → | esc | 1 | 1/2 | | | | | | | | | | | | |

A Tabela 6 ilustra alguns casos para os quais o método realiza suas estimativas para o caracter que sucede imediatamente a seqüência **x y z x v m x y w x y z x**, processada até aquele ponto. Assumindo-se que o codificador aritmético possa gerar um número ótimo de bits com base na distribuição que recebe, o símbolo pode ser codificado por meio de $-log_2 p(.)$ bits. Por exemplo, se o caracter que sucede a seqüência **z x** for **v**, o número de bits necessários para codificá-lo será dado em função da probabilidade que o modelo atribui a este no contexto **z x**, ou seja, 1 bit. Porém, se o caracter subseqüente for **m,** um caracter de escape (*esc*) será emitido, pois o modelo *r*=2 não comporta uma previsão para **m** no contexto **z x.** O caracter *esc* será codificado com a probabilidade designada a ele e um modelo de menor ordem será chamado para codificação de **m**. No entanto, novamente, não há previsão para **m** no modelo *r*=1, agora com o contexto **x**. Assim, um novo caracter de escape será necessário e o modelo

de ordem *r*=0 será utilizado para codificação de **m**. Ao final, 6,83 bits serão necessários para codificá-lo. No caso da seqüência ser sucedida por um caracter não visto até aquele momento, tal como **a**, por exemplo, este será codificado pela acumulação dos caracteres de escape necessários para chamar cada modelo, da ordem *r*=2 até *r* = -1. Para este caso, em particular, o número de bits necessários para compactá-lo é bem superior aos 8 bits utilizados para representar sua versão não-compactada.

Tabela 6: Exemplo de Previsões do PPMC Para o Caracter que Sucede a Seqüência já Processada **x y z x v m x y w x y z x**

| | | Caracter subseqüente | Probabilidades codificadas | | | | Espaço ocupado pelo código |
|---|---|---|---|---|---|---|---|
| Caso1 | s/ exc. | v | 1/2 | | | | $-\log_2 (1/2) = 1$ bit |
| Caso2 | s/ exc. | m | 1/2 | 2/6 | 1/19 | | $-\log_2 (1/2 \cdot 2/6 \cdot 1/19) = 6,83$ bits |
| Caso3 | s/ exc. | a | 1/2 | 2/6 | 6/19 | 1/256 | $-\log_2 (1/2 \cdot 2/6 \cdot 6/19 \cdot 1/256) = 12,25$ bits |
| Caso2 | c/ exc. | m | 1/2 | 2/5 | 1/15 | | $-\log_2 (1/2 \cdot 2/5 \cdot 1/15) = 6,23$ bits |
| Caso3 | c/ exc. | a | 1/2 | 2/5 | 6/15 | 1/250 | $-\log_2 (1/2 \cdot 2/5 \cdot 6/15 \cdot 1/250) = 11,61$ bits |

Os casos (1, 2 e 3; s/exc.), descritos na Tabela 6, têm suas estimações baseadas sem a aplicação de uma técnica denominada de *exclusão* que pode ser utilizada para melhorar a estimação de probabilidades. Os casos (2 e 3; c/exc.), em que há comutação para modelos de ordem inferior, ilustram a codificação de **m** e de **a**, utilizando a *exclusão*. Por exemplo, para o caso da codificação do caracter **m** (caso 2; c/exc.) exclui-se o caracter **v** da possibilidade de ocorrência no contexto **x** para a ordem *r*=1, uma vez que este já não correspondia ao caracter correto no contexto **z x** em *r*=2. O mesmo ocorre para o modelo *r*=0, em que as contagens para **v** e **y** são excluídas do cômputo das probabilidades, reduzindo de 6,83 para 6,23 o número de bits necessários para codificar o caracter **m**.

Este exemplo simples ilustra, também, o problema da *freqüência zero*. Assim, é maior o número de bits necessários para codificar caracteres que não estão representados em um modelo de ordem maior, e para os quais é necessária a emissão de um caracter de escape na comutação para modelos de ordem inferior. No entanto, ao longo da compressão, com a adaptação ao texto que está sendo processado e, à medida que o modelo se torna mais representativo da seqüência em progresso, a tendência é a de que as previsões se tornem mais acuradas e que a necessidade de emissão de caracteres de escape diminua bastante.

Durante o processo de descompactação, o decodificador interpreta o caracter a ser decodificado a partir do fluxo de bits recebido com auxílio do mesmo modelo de probabilidades (no caso, a combinação dos modelos de diversas ordens) do compactador.

Como o compactador e o descompactador partem de estados iniciais iguais e o modelo construído no descompactador é atualizado por meio do mesmo algoritmo que o modelo do compactador, a sincronicidade entre estes é mantida.

## 2.4.3 Codificadores Estatísticos

Métodos baseados na previsão de símbolos dependem de um codificador para realizar a conversão das distribuições probabilísticas providas pelo *modelo* em um fluxo de bits de saída. A codificação do símbolo previsto pode ser realizada, por exemplo, por meio da Codificação de *Huffman* ou da Codificação Aritmética.

O método de *Huffman*, cuja descrição pode ser encontrada em (WITTEN, MOFFAT e BELL, 1999; SAYOOD, 2000) consiste em um dos algoritmos de codificação mais comumente utilizados, sendo que variantes deste são aplicadas em compressão de texto, áudio e vídeo. A codificação de *Huffman* gera códigos "livre-de-prefixo", designando palavras-código de tamanho inteiro e variável a partir de uma distribuição probabilística não-uniforme ao longo de um alfabeto finito. Os códigos gerados são ótimos (do ponto de vista da entropia) somente se as probabilidades são potência negativa de 2 (BELL, CLEARY e WITTEN, 1990; SAYOOD, 2000). Por outro lado, a Codificação Aritmética permite a designação de palavras-código de comprimento próximo do ótimo, utilizando frações de bits.

### 2.4.3.1 Codificação Aritmética

A codificação aritmética possibilita a codificação de uma mensagem em um número de bits extremamente próximo à sua entropia em relação a um modelo (BELL, CLEARY e WITTEN, 1990). Um codificador aritmético gera códigos de comprimento variável e unicamente decodificáveis a partir das probabilidades fornecidas por um modelo qualquer, o que possibilita uma separação clara e prática em relação ao modelo gerador de probabilidades no contexto do esquema genérico *modelo-codificador*. O codificador aritmético realiza a codificação símbolo a símbolo, porém, em lugar de designar uma palavra-código a cada símbolo individual, um codificador aritmético designa uma palavra-código à seqüência inteira. Assim, uma de suas principais vantagens é apresentar uma eficiência muito próxima do limite teórico de compressão, designando códigos de comprimento praticamente correspondentes a $-log_2 p(.)$ bits, de modo que os símbolos são codificados utilizando um número não inteiro de bits.

A codificação aritmética é baseada na idéia de que a mensagem será representada por um intervalo de números reais, sendo que sua saída é a representação binária de um ponto dentro

do correspondente intervalo. Os símbolos que compõem a mensagem são processados seqüencialmente, com suas probabilidades geradas por um modelo. Cada símbolo processado reduz o intervalo, inicialmente definido como [0, 1), para a porção alocada a este símbolo, proporcionalmente em relação à sua probabilidade. Este processo pode ser ilustrado através de um exemplo simples, utilizando um modelo estático que define probabilidades fixas para o alfabeto pequeno dado por Ω={v, w, x, y, z, !}. Para simplificar a descrição, em lugar de ser apresentada a representação binária, serão utilizados valores decimais. As probabilidades e os respectivos intervalos para cada símbolo são descritos na Tabela 7.

Tabela 7: Probabilidades para os símbolos do alfabeto Ω exemplificado

| Símbolo | Probabilidades $p_i$ | Intervalo inicial |
|---|---|---|
| v | 0,2 | [0,0 , 0,2) |
| w | 0,3 | [0,2 , 0,5) |
| x | 0,1 | [0,5 , 0,6) |
| y | 0,1 | [0,6 , 0,7) |
| z | 0,2 | [0,7 , 0,9) |
| ! | 0,1 | [0,9 , 1) |

Para transmitir a mensagem “xvzz!”, começa-se reduzindo o intervalo inicial [0 , 1) para a partição deste definida pelo modelo para o símbolo “x”, o que resulta em [0,5 , 0,6). O segundo símbolo “v” reduz este sub-intervalo para sua primeira quinta-parte, pois é definido na partição [0,0 , 0,2), o que resulta em [0,5 , 0,52). A operação completa é ilustrada por meio da Tabela 8 e da Figura 4.

Tabela 8: Processo de Codificação Aritmética

| Seqüência | Intervalo |
|---|---|
| Inicialmente | [0 , 1,0) |
| Após processar x | [0,5 , 0,6) |
| Após processar v | [0,5 , 0,52) |
| Após processar z | [0,514 , 0,518) |
| Após processar z | [0,5168 , 0,5176) |
| Após processar ! | [0,51752 , 0,5176) |

Ao final do processamento, a mensagem “xvzz!” é representada na forma do intervalo [0,51752 , 0,5176). Para armazenar ou transmitir esta mensagem codificada é desnecessário enviar os limites do intervalo, bastando enviar um número qualquer contido neste, como o valor 0,51752, convertido para binário. Outro valor possível seria 0,51754, por exemplo.

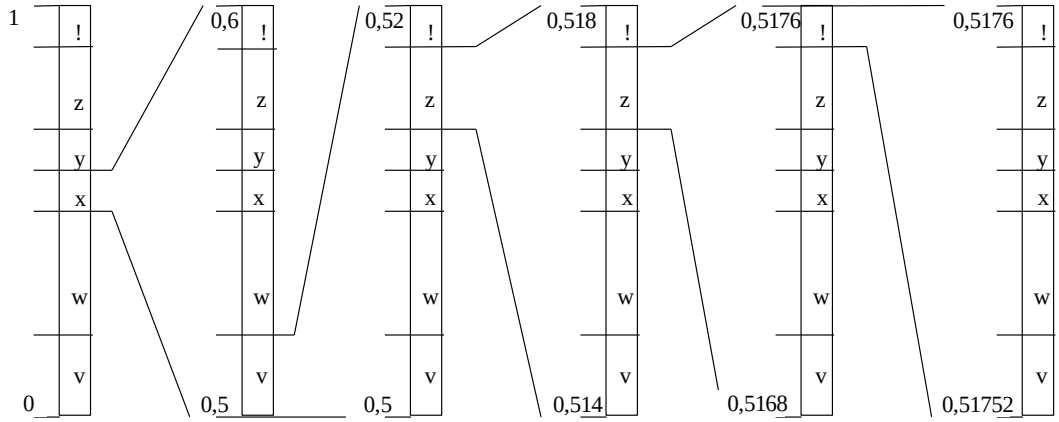



Figura 4: Processo de Codificação Aritmética

O processo de decodificação é análogo, sendo resumido na Tabela 9. Ao receber a seqüência de bits relativa ao valor 0,51752, o decodificador identifica seu enquadramento na partição do intervalo [0,5 , 0,6), correspondente à "x", fornecido pelo seu modelo. Desta forma, é possível repetir o primeiro passo do codificador, reduzindo o intervalo inicial [0 , 1) para [0,5 , 0,6). Neste ponto, o caracter "x" pode ser removido do número codificado, subtraindo-se deste o valor inferior do intervalo conhecido para "x", o que resulta em 0,01752, que, por sua vez, é dividido pelo valor 0,1 do intervalo relativo a "x". Assim, chega-se a 0,1752, contido na partição correspondente ao caracter "v", identificado como o segundo caracter a ser decodificado. O processo segue deste modo até decodificar toda a seqüência conforme ilustrado na Tabela 2.9.

Tabela 9: Processo de Decodificação Aritmética

| Número codificado | Símbolo de saída | inf | sup | intervalo |
|---|---|---|---|---|
| 0,51752 | x | 0,5 | 0,6 | 0,1 |
| 0,1752 | v | 0,0 | 0,2 | 0,2 |
| 0,876 | z | 0,7 | 0,9 | 0,2 |
| 0,88 | z | 0,7 | 0,9 | 0,2 |
| 0,9 | ! | 0,9 | 1 | 0,1 |
| 0,0 | | | | |

Uma mensagem à qual foi designada uma probabilidade pequena, caracterizando uma grande quantidade de informação, necessita de um maior número de bits para representá-la. Por outro lado, uma mensagem com maior probabilidade, portanto, com menor quantidade de informação, necessita de um menor número de bits para representá-la.

Um problema que surge na decodificação é a detecção do fim da mensagem, uma vez que, por exemplo, o número real 0 (zero) pode representar qualquer quantidade de caracteres

“v”. A solução típica para este problema consiste em definir um símbolo especial para designar o fim da mensagem.

## 2.5 Desempenho de Compressão

### 2.5.1 Compressão *on-line versus* Compressão *off-line*

A compressão com modelagem *off-line*, gerando um modelo estático, restringe a eficácia à compactação de textos com distribuição probabilística similar à do modelo treinado. Porém, sem realizar a adaptação do modelo durante a compressão, o processo tende a ser computacionalmente menos dispendioso. Na modelagem semi-estática o modelo do exemplar a ser compactado é construído em um estágio correspondente a uma primeira passada pelo texto, antecedendo a passada relativa à compressão propriamente dita. Neste caso, a descrição do modelo precisa ser transmitida juntamente com a seqüência codificada, o que pode tornar-se dispendioso, caso seja utilizado um modelo muito complexo. Para determinadas aplicações, tais como a comunicação de dados interativa, estas duas passadas pelo texto são inviáveis.

A compressão *on-line* com modelagem adaptativa requer uma única passada pela seqüência de dados, sem contar com qualquer outro conhecimento prévio sobre a seqüência, além do alfabeto da fonte. Compactadores *on-line* constroem incrementalmente um modelo da seqüência de entrada enquanto a codificam. Neste caso, a transmissão do modelo é desnecessária, uma vez que a sua reconstrução também é realizada de modo incremental pelo decodificador. É preciso observar que as previsões iniciais realizadas são pouco acuradas, porém, ao longo do processo de compressão, o modelo tende a tornar-se mais representativo da seqüência de dados de entrada. Métodos de compressão adaptativos ganham em flexibilidade frente aos métodos estáticos, mas tendem a ser computacionalmente mais intensos e dispendiosos.

O processo de compressão que realiza uma adaptação a um modelo previamente treinado no modo *off-line* utiliza uma modelagem semi-adaptativa que pode propiciar melhores taxas de compressão para textos com estrutura estatística similar à do modelo treinado. No entanto, para textos com distribuição probabilística com muita divergência em relação à distribuição do modelo treinado, a taxa de compressão será prejudicada.

### 2.5.2 Medidas de Desempenho de Compressão

Um algoritmo de compressão pode ser avaliado em relação à sua complexidade, à memória consumida no processo, à velocidade de processamento em uma determinada máquina, à quantidade de compressão, e à fidelidade do arquivo compactado em relação ao original (SAYOOD, 2000). Para o caso de compressão sem perda, como a reconstrução do arquivo compactado precisa ser exata, o critério de fidelidade absoluta entre a versão original e a descompactada precisa ser automaticamente cumprido.

A quantidade de memória requerida durante o processo de compactação e descompactação varia conforme o método de compressão adotado, dependendo, também, do tipo de estrutura de dados utilizada, além dos tipos de dados que estão sendo codificados. Métodos adaptativos necessitam de mais memória para armazenar as tabelas específicas ao texto em codificação. Os métodos *LZ* (*Lempel-Ziv*) utilizam, geralmente, umas poucas dezenas de kilobytes de memória para operação das tabelas que contém *sub-strings* ocorridas anteriormente, usualmente na forma de uma estrutura indexada que permite um acesso rápido. Métodos baseados em símbolos, tais como PPM e DMC (*Dynamic Markov Compression*), armazenam tabelas de contextos e as distribuições de probabilidade dos caracteres que nestes aparecem, trabalhando melhor com centenas de kilobytes, ou mesmo dezenas de megabytes de memória (WITTEN, MOFFAT e BELL, 1999).

A mensuração da velocidade de processamento de um determinado método depende do modo como este foi implementado, da arquitetura da máquina em que este roda e da qualidade do compilador (WITTEN, MOFFAT e BELL, 1999). A decodificação é, geralmente, mais rápida que a codificação, para métodos baseados em *LZ*, como é o caso do *LZSS*. Muitos destes evitam recursos mais dispendiosos, como a codificação aritmética ou a codificação de *Huffman*, utilizando técnicas mais simples para codificação. Em geral, os métodos PPM costumam ser mais lentos que *LZ*. Quanto aos codificadores estatísticos, segundo (WITTEN, MOFFAT e BELL, 1999), a codificação de *Huffman* é mais adequada para aplicações estáticas e a codificação aritmética é preferível em situações em que a codificação precisa ser adaptativa e *on-line*.

Na prática, os algoritmos de compressão são submetidos ao *trade-off* entre o tempo necessário à compactação e a quantidade de compactação obtida. Assim, tende-se a sacrificar um pouco da taxa de compressão que poderia ser atingida, em favor de uma maior velocidade de processamento. O desempenho relativo dos diferentes métodos de compressão depende dos tipos de dados que estão sendo compactados (WITTEN, MOFFAT e BELL, 1999). Assim,

certas imagens de *bitmap* podem propiciar taxas de compressão bem melhores que determinados arquivos de texto em linguagem natural, por exemplo. A taxa de compressão efetivamente atingida pelos melhores, assim como pelos mais populares métodos de compactação existentes, pode variar muito, conforme a natureza do arquivo utilizado para compactação.

A quantidade de compactação obtida pode ser expressa de diversas maneiras. Para exemplificação, será considerado um byte de 8 bits. A *razão*[9] de compressão (SAYOOD, 2000) é dada por meio de:

$$Rc = \frac{Lo}{Lc} \tag{2.22}$$

Sendo *Lo* correspondente ao número de bits requeridos para representar os dados originais e *Lc* o número de bits requeridos para representá-los na versão compactada. Assim, por exemplo, se um arquivo de 160 bytes (1280 bits) for compactado para 40 bytes (320 bits), a razão de compressão atingida é de 4:1.

A quantidade de compactação também pode ser expressa em termos de percentual de redução do arquivo original, dada por:

$$Pc=[(1- (Lc/Lo)]\times 100 \tag{2.23}$$

Para o exemplo considerado, o arquivo é reduzido em 75%, ou seja, o arquivo remanescente após a compactação corresponde a 25% do original.

Outra medida utilizada é a *taxa*[10] de compressão, aferida com base no número médio de bits necessários para representar cada símbolo ou caracter (byte) da entrada, ou seja, o número de *bits por caracter* (bpc) ou *bits por byte*. Utilizando-se o código ASCII formado por 256 símbolos (caracteres) distintos, um caracter é dado por 8 bits. A taxa de compressão em *bits por caracter* (bpc) pode ser calculada por meio de:

$$Pc = \left(\frac{Lc}{Lo}\right)\times 8 \tag{2.24}$$

Para o exemplo considerado, em que a seqüência de 160 bytes é reduzida para 40 bytes, a taxa de compressão foi de 2 bpc. Assim, quanto menos bits forem necessários para representar cada símbolo (caracter) de uma seqüência discreta, melhor a compressão obtida.

---

[9] *Compression ratio*, conforme Sayood (2000, p. 5).

[10] *Compression rate*, conforme Sayood (2000, p. 6), Witten, Moffat e Bell (1999) e Bell, Cleary e Witten (1990). Alguns autores utilizam o termo "taxa" de compressão para designar o que é o resultado do cálculo *da razão* Rc=*Lo*/*Lc*, em que, quanto maior o valor, melhor a quantidade de compressão resultante.

Enfatiza-se que não existe um compactador que seja capaz de comprimir sem perda todas as seqüências discretas possíveis, ou seja, a compressão de determinados arquivos sempre implica na expansão de outros. Portanto, um compactador deve ser desenvolvido de tal modo que a probabilidade de diminuição do tamanho de um arquivo processado é grande, ao passo que a probabilidade de expandir seu tamanho é pequena.

## 2.6 Considerações Finais

Ao longo deste capítulo foram introduzidos conceitos básicos relativos à entropia, modelagem e codificação, para fundamentação teórica em compressão de texto. As técnicas de compressão aqui apresentadas já estão consolidadas e algumas destas já foram incorporadas a compactadores de texto, como o *compress*, baseado em *LZW*, e o *gzip*, baseado no método *LZ*77. Métodos PPM são, atualmente, tidos como o estado da arte em compressão de texto. Em contraste com a velocidade de compressão de técnicas baseadas em Lempel-Ziv (*LZ*), os métodos PPM trabalham com uma grande requisição de memória e menor velocidade de processamento.

A relativamente recente incursão de redes neurais artificiais na área de compressão de texto propõe uma alternativa para modelagem das estimativas probabilísticas por meio de um aprendizado das regularidades da seqüência apresentada. No entanto, a adequação aos diversos tipos de modelagem descritos pode variar conforme as características específicas de cada um dos diversos modelos básicos de redes neurais existentes. O próximo capítulo introduz conceitos relativos a redes neurais artificiais, sendo descritos trabalhos anteriores em que estas foram utilizadas para compressão de texto, evidenciando suas diferenças em relação à abordagem proposta no presente trabalho.

# 3 REDES NEURAIS ARTIFICIAIS E A TEORIA DA RESSONÂNCIA ADAPTATIVA

Neste capítulo são introduzidos conceitos básicos de redes neurais artificiais (RNAs) e da Teoria da Ressonância Adaptativa (*Adaptive Resonance Theory - ART*), juntamente com alguns modelos não-supervisionados e supervisionados baseados em *ART*. São apresentadas abordagens em que redes *feedforward* treinadas por meio do algoritmo *Backpropagation* foram utilizadas para a compressão de texto e que se diferenciam da proposta do presente trabalho, baseada em uma rede da classe ARTMAP, originária de *ART*.

## 3.1 Introdução

Redes neurais artificiais (RNAs)[11] são modelos computacionais desenvolvidos com inspiração em estruturas neurais biológicas, formados pela interconexão e processamento conjunto de muitos elementos de cômputo local, denominados de neurônios artificiais. RNAs representam sistemas extremamente simples, se comparadas a redes neurais biológicas. No entanto, conforme Haykin (1999), RNAs têm alguma semelhança com o cérebro biológico, na medida em que o conhecimento é adquirido do ambiente através de um processo de aprendizado e as forças conectivas (pesos sinápticos) entre os neurônios são utilizadas para armazená-lo. Uma RNA aprende indutivamente um *modelo* do ambiente em que opera através de observações provenientes deste. Os dados são processados pela rede de neurônios com adaptação de seus parâmetros internos para representação das regularidades e características aprendidas do ambiente. O conhecimento adquirido pela RNA, neste sentido, diz respeito ao modelo formado e codificado na rede ao longo do processo de aprendizado e utilizado por esta para responder de modo adequado aos estímulos de entrada do ambiente de operação.

Partindo da motivação biológica, o campo de pesquisa de RNAs passou por diversas fases e marcos, desde a proposta do neurônio artificial, por McCulloch e Pitts (1943), a regra de aprendizado de Hebb (1949), e o *Perceptron* de Rosenblatt (1958), até chegar ao seu atual estágio de desenvolvimento. Após uma demonstração das limitações dos *Perceptrons*, publicada por Minsky e Papert (1969), a área de RNAs passou por um período de obscuridade, em um contexto de poucos recursos computacionais e carência de apoio financeiro. No entanto, o interesse pela área foi renovado no início dos anos 1980 com a

[11]Também denominadas de redes neuronais ou de redes conexionistas.

divulgação de novas abordagens e propostas de soluções[12] para diversas das limitações que haviam sido levantadas em relação às RNAs. Uma coletânea organizada em (ANDERSON e ROSENFELD, 1989) reúne estes trabalhos, além de outros que marcaram esta trajetória.

A evolução do campo de RNAs conduziu à concepção de uma grande variedade de modelos neurais e de algoritmos de aprendizado, com implementações em *software* e em *hardware*, em um contexto de maior disponibilidade de recursos computacionais para desenvolvimento e execução experimental. Atualmente, além das pesquisas motivadas pela modelagem de sistemas neurais biológicos e cognitivos, o campo abrange o desenvolvimento de modelos aprimorados e algoritmos eficientes de RNAs com aplicações[13] em diversas áreas. RNAs têm sido utilizadas em áreas, como, engenharia e computação, finanças e negócios, medicina e ciências em geral, no desenvolvimento de sistemas de previsão, reconhecimento e classificação de padrões, suporte à tomada de decisão, diagnóstico de falhas, identificação e controle de processos. Estas aplicações são motivadas pelas características geralmente associadas às RNAs, como: capacidade de aprendizado por meio de exemplos extraídos do ambiente de dados e adaptabilidade a este, robustez diante de dados incompletos ou com ruído, tratamento de problemas lineares e não-lineares, e capacidade de generalização diante de dados novos do domínio do problema considerado (HAYKIN, 1999). Ressalta-se que as características de RNAs como estruturas interconectadas *maciçamente paralelas* (HAYKIN, 1999) podem ser evidenciadas por meio de implementações paralelas destas, beneficiando sua operação em tempo real em aplicações de larga escala. A integração de RNAs com outras técnicas de inteligência artificial, dentre as quais, sistemas especialistas, raciocínio baseado em casos, algoritmos genéticos (WAZLAWICK, 1993), e lógica *fuzzy* (ZADEH, 1965; KOSKO, 1992; KLIR e YUAN, 1995; ROSS, 1995), permite, ainda, a concepção de sistemas híbridos para a abordagem mais eficaz de determinados problemas.

No presente trabalho, uma RNA originária da Teoria da Ressonância Adaptativa (*ART*) (CARPENTER e GROSSBERG, 1987a) e que integra alguns princípios da teoria dos conjuntos *fuzzy* é aplicada à denominada compressão de *texto* (Capítulo 2) e à classificação automática de textos (Capítulo8). A função da rede neural no sistema desenvolvido para estas aplicações é o aprendizado de um modelo a partir de seqüências de dados, sendo este utilizado

[12]Como exemplos, podem ser citados: a rede recorrente com memória associativa proposta por Hopfield em 1982, marco para a retomada do interesse por RNAs; o algoritmo *Backpropagation* para *Perceptrons Multicamadas*, proposto por Werbos em 1974 e popularizado por Rumelhart, Hinton e Williams (1986); a Teoria da Ressonância Adaptativa de Grossberg (1976; 1980) e os mapas SOM de Kohonen (1982) (HAYKIN, 1999).

[13] Através de buscas em bases de dados como IEEE, ACM, *Science Direct*, assim como na internet, constata-se a diversidade de aplicações de modelos de RNAs nas áreas de ciências e tecnologia, assim como em aplicações comerciais.

para *predição seqüencial* com estimação de uma distribuição probabilística para o símbolo que sucede cada subseqüência de entrada. As estimativas realizadas incrementalmente pela rede são convertidas em seqüencias de bits por um módulo de codificação que, juntamente com a rede neural, integra o sistema de compressão sem perda com modelagem adaptativa. Para a classificação automática de textos é efetuada a medição de similaridade entre uma seqüência de teste e cada modelo gerado pela rede neural por meio de amostras de texto pertencentes a cada classe, utilizando, neste processo, a técnica neural de compressão desenvolvida.

O ambiente para compressão *sem perda* de seqüências discretas, como *texto*, apresenta características e requisitos de operação que devem ser avaliados no contexto do tipo de modelagem requerida e da funcionalidade da RNA considerada para a tarefa. Para realizar uma modelagem adaptativa em compressão de *texto*, uma RNA precisa aprender as entradas de modo *incremental*, ou seja, à medida que estas são apresentadas, em uma *única passada* pela seqüência originada de uma fonte de dados cuja estrutura estatística é desconhecida *a priori*. A rede deve aprender novos dados no modo *on-line*, ou seja, sem ter acesso a todo o conjunto anteriormente visto e, portanto, sem a retenção dos dados já processados para apresentações subseqüentes. Redes que derivam de *ART* (Seção 3.3) podem representar uma alternativa para a compressão de *texto* por meio de RNAs ao viabilizar uma modelagem adaptativa com aprendizado incremental e *on-line*. Ao utilizar uma rede da classe ARTMAP (Seção 3.3.3), originária de *ART*, a proposta do presente trabalho contrapõe-se às abordagens anteriores (Seção 3.4), apresentadas por (SCHMIDHUBER e HEIL, 1996), (NATSEV, 1997) e (LONG, NATSEV e VITTER, 1999), em que foram adotadas redes *feedforward* treinadas por meio do algoritmo de retro-propagação (*Backpropagation* - BP).

Exemplos da aplicação de RNAs para a classificação (ou categorização) de textos podem ser encontrados em (WIENER, PEDERSEN e WEIGEND, 1995), (KESSLER, NUNBERG e SCHÜTZE, 1997), (RUIZ e SRINIVASAN, 1999) e, recentemente, em (RAUBER e MÜLLER-KÖGLER, 2001). No presente trabalho, a aplicação da rede proposta à classificação de textos (Capítulo 8) baseia-se na utilização do método de compressão desenvolvido, segundo uma abordagem que se diferencia da tradicionalmente adotada por RNAs para esta tarefa.

A abordagem relativa às RNAs (Seção 3.2), realizada neste capítulo, visa apenas colocar a área em perspectiva para apresentar modelos baseados em *ART* (Seção 3.3) e trabalhos anteriores de RNAs em compressão de texto (Seção 3.4). Um maior detalhamento sobre os diversos modelos de RNAs e seus respectivos algoritmos pode ser encontrado em (PAO,

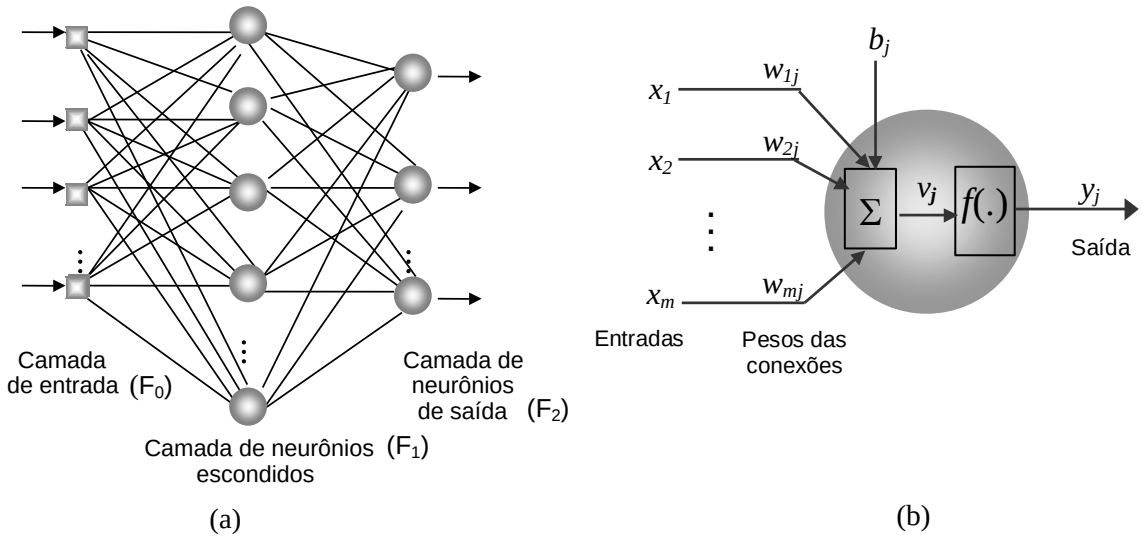
1989; HERTZ, KROGH e PALMER, 1991; FAUSETT, 1994; PANDYA e MACY, 1996; SKAPURA, 1996; BISHOP, 1999; HAYKIN, 1999), dentre outras fontes.

## 3.2 Conceitos Básicos de Redes Neurais Artificiais - RNAs

Conforme Fausett (1994), RNAs têm sido desenvolvidas como generalizações de modelos matemáticos da cognição humana e/ou da neurobiologia, com base nas seguintes suposições:

- o processamento da informação ocorre em muitos elementos simples, denominados de *neurônios* ou *unidades neurais*;
- a transmissão de sinais entre os neurônios é feita através de *conexões*, sendo que cada uma destas possui um *peso* associado que representa a força da sinapse entre estes;
- o sinal transmitido é ponderado (operação realizada em uma rede neural típica) pelo peso da conexão sináptica e uma *função de ativação* é aplicada sobre a entrada (soma das entradas ponderadas) do neurônio para determinar o seu *sinal de saída*.

Os elementos conceituais descritos são ilustrados por meio dos exemplos de uma RNA e de um modelo de neurônio artificial representados nas Figuras 5a e 5b.




Figuras 5: (a) Exemplo de RNA. (b) Exemplo de neurônio artificial, adaptado de HAYKIN (1999).

A Figura 5a apresenta uma das muitas configurações possíveis para uma RNA, sendo que esta, particularmente, possui uma estrutura com uma única camada de neurônios escondidos, sendo *amplamente conectada* (todo o elemento de uma camada liga-se a todo elemento de uma camada adjacente), com propagação direta (sem laços nas conexões) dos sinais na rede. As unidades receptoras (camada F0) enviam os sinais de entrada ponderados pelos pesos nas conexões dirigidas aos neurônios escondidos (camada F1). Estes efetuam um cômputo interno

e propagam as respectivas respostas, ponderadas pelos pesos nas conexões dirigidas aos neurônios de saída da rede (camada F2) que, por sua vez, realizam o seu processamento para emissão das saídas da rede. As definições para estrutura, conectividade, número de camadas escondidas, número de unidades de entrada, tipo e número de neurônios escondidos e de saída, são dependentes do problema a ser abordado, do modelo de RNA utilizada e de sua funcionalidade.

O neurônio *j*, exemplificado na Figura 5b, recebe os sinais de entrada, denotados por ($x_1$, $x_2$, …, $x_m$), para $x_i$, $i$=1, 2, ..., $m$, através de conexões com os respectivos pesos ($w_{1j}$, $w_{2j}$, …, $w_{mj}$) dirigidos de uma unidade $i$ para o neurônio $j$ [14]. O parâmetro externo $b_j$ diz respeito ao termo de *bias*, que tem o efeito de aumentar ou diminuir a entrada ponderada da função de ativação $f(\cdot)$. O termo de *bias* pode ser representado por meio de uma entrada adicional de valor fixo $x_0$=+1, associada a um peso $w_{0j}=b$. Como alternativa ao *bias* pode ser adotado um *threshold* $\theta_j$ (o que procura emular o limiar de disparo do neurônio biológico) com $x_0$=-1 e $w_{0j}=\theta_j$. O nível de ativação $v_j$ do neurônio $j$ exemplificado é dado por:

$$v_j = \sum_{i=1}^{m} w_{ij} x_i + b_j \tag{3.1}$$

e o seu sinal de saída $y_j$ é:

$$y_j = f(v_j) \tag{3.2}$$

A denominada *função de ativação* $f(v)$ define a saída do neurônio em termos do potencial de ativação $v$, limitando sua amplitude da saída. Os modos de ativação e os tipos de funções de ativação utilizadas nos neurônios diferem para os vários modelos básicos de RNAs e estão evidenciados em seus respectivos algoritmos. Para o modelo original do neurônio artificial de McCulloch-Pitts (HAYKIN, 1999) é utilizada uma função de *threshold* que define a saída do neurônio com um valor 1 (um), se o seu nível de ativação total é não-negativo, ou com um valor 0 (zero), caso este seja negativo. Uma função *sigmoidal* (que exibe não-linearidade e é diferenciável) é freqüentemente adotada em redes do tipo perceptron de múltiplas camadas (*Multilayer Perceptrons* - MLP) treinadas por meio do algoritmo de aprendizado de retro-propagação (*Backpropagation* - BP) (HERTZ, KROGH e PALMER, 1991; FAUSETT, 1996; HAYKIN, 1999). Já, RNAs com função de base radial (*Radial-Basis Function* - RBF) (HAYKIN, 1999; BISHOP, 1999) podem utilizar uma função gaussiana na ativação de cada neurônio na camada escondida. As redes fuzzy ART e fuzzy ARTMAP, por sua vez, utilizam

[14] Na convenção adotada em (HAYKIN, 1999), um peso dirigido de $i$ à unidade $j$ seria denotado por $w_{ji}$.

um operador *fuzzy* no cálculo do grau de ativação para cada neurônio que representa uma categoria, como será visto com mais detalhes no Capítulo 4.

### 3.2.1 Classificação de RNAs

Uma RNA pode ser utilizada para o aprendizado de um *modelo* do ambiente de dados ao qual é exposta, a partir de observações ou de exemplos extraídos deste. De forma geral, os sinais de entrada processados e transmitidos entre os neurônios artificiais obedecem à ação de um algoritmo de aprendizado que promove o ajuste dos pesos conectivos, consolidando na rede o "conhecimento" utilizado para produzir as respostas de saída. O modelo construído neste processo deve ser suficientemente consistente com o ambiente real de aquisição dos dados (HAYKIN, 1999) para que a RNA possa responder aos estímulos ambientais futuros e satisfazer os requisitos e objetivos da aplicação para a qual foi projetada. Portanto, são identificados dois momentos de operação de uma RNA: o aprendizado, em que o modelo é formado por meio de ajustes de seus parâmetros internos, e a execução, em que os estímulos são propagados através do modelo aprendido para obtenção das respostas, sem que seus parâmetros sofram ajustes.

Um treinamento aplicado à RNA sistematiza o processo de aprendizagem utilizando um conjunto formado por exemplos representativos do ambiente, sendo que, para aferição da qualidade de seu desempenho, a rede deve ser testada por meio de um conjunto distinto daquele apresentado para treiná-la. Este procedimento visa avaliar a capacidade de *generalização* da rede, ou seja, a capacidade de gerar respostas satisfatórias diante de casos novos do domínio da aplicação em foco.

As classes de problemas em que RNAs são adotadas podem ser genericamente divididas em: classificação de padrões, predição de eventos futuros com base em dados passados, aproximação de funções, memórias associativas, otimização, auto-organização de agrupamentos (*clustering*) e controle de sistemas (JAIN, MAO e MOHIUDDIN, 1996). Em (FAUSETT, 1994; PANDYA e MACY, 1996; SKAPURA, 1996; HAYKIN, 1999) são apresentados diversos modelos (arquiteturas e algoritmos básicos) de RNAs. As características de cada modelo de RNA determinam seu grau de adequação a uma determinada classe de problema e o conhecimento destas norteia a escolha por um modelo básico de rede e o desenvolvimento das necessárias adaptações no contexto de um projeto.

Uma RNA pode ser descrita por meio de sua estrutura e da conectividade entre os neurônios que a compõem, pelo cômputo que ocorre em cada um destes, pelas forças nas suas

conexões e pela forma de aprendizado adotada. Assim, taxonomicamente, estas podem ser classificadas conforme os aspectos detalhados nas Seções 3.2.1.1 a 3.2.1.3.

### 3.2.1.1 Estrutura e Conectividade

O modo como as unidades de uma RNA são conectadas, o tipo de arranjo estrutural, e os tipos de neurônios, são elementos da arquitetura de uma RNA. Em (HAYKIN, 1999) é apresentada uma analogia com grafos direcionados para ilustrar a conectividade em RNAs. RNAs podem ser totalmente conectadas (todo o elemento de uma camada é ligado a todo elemento de uma camada adjacente), podem apresentar uma conectividade especializada com ligações seletivas entre elementos, e até mesmo ligações que rompem com a noção de arranjo em camadas na rede. Quanto à direção de propagação dos sinais para processamento das respostas, as RNAs podem ser divididas nos tipos principais, descritos a seguir.

- **Redes *Feedforward* (propagação à frente).** Os sinais de entrada são propagados no sentido da camada de entrada para a de saída e os grafos da rede não possuem laços (*loops*). Os neurônios são organizados em camadas ligadas por conexões unidirecionais, sem conexões laterais entre nós de uma mesma camada (HAYKIN, 1999). Em redes *feedforward single-layer* (com uma camada de neurônios), as unidades de entrada da rede são projetadas diretamente aos neurônios da camada de saída. Redes *feedforward multilayer* (com múltiplas camadas de neurônios) possuem uma ou mais camadas com neurônios escondidos, sendo que os sinais são propagados em direção à saída através de conexões de camadas de índice inferior dirigidas a camadas de índice superior. A inclusão de camadas escondidas confere à rede a capacidade de criar uma representação interna dos padrões, ampliando seu potencial de mapeamento entre entradas e de saídas. Redes MLPs (HERTZ, KROGH e PALMER, 1991; FAUSETT, 1996; HAYKIN, 1999), e redes RBF (HAYKIN, 1999; BISHOP, 1999) são exemplos de estruturas *feedforward* com múltiplas camadas, aplicadas com sucesso em problemas de classificação de padrões e aproximação de funções.

- **Redes com conexões recorrentes**. Podem ser vistas como sistemas dinâmicos (HAYKIN, 1999) e apresentam laços no processamento em função de conexões de realimentação (*feedback*) de neurônios que enviam seu sinal de saída como entrada aos neurônios de uma camada de índice inferior, ou de neurônios que realimentam a si mesmos com seus sinais de saída (*self-feedback*). A rede de *Hopfield*, as redes de *Elman* e *Jordan*, e a denominada *Recurrent Backpropagation*, constituem exemplos de redes recorrentes (HERTZ, KROGH e PALMER, 1991; FAUSETT, 1994; SKAPURA, 1996).

Redes ART (CARPENTER e GROSSBERG, 1987a; FAUSETT, 1994), apresentadas na Seção 3.3, possuam conexões de *feedback* entre a "camada de comparação" e a "camada de categorias" para teste da similaridade entre padrões de entrada e os protótipos representados nos pesos.

#### 3.2.1.2 Paradigma de Aprendizado

Durante o processo de aprendizado, a rede adquire seu conhecimento sobre o ambiente de dados, ajustando os pesos de suas conexões conforme um determinado algoritmo computacional, permitindo sua resposta aos estímulos subseqüentes. O paradigma de aprendizado utilizado diz respeito ao modo como as informações do ambiente são recebidas pela rede e está relacionado à arquitetura desta e ao seu algoritmo de aprendizado. Os paradigmas de aprendizado podem ser divididos de acordo com tipos descritos a seguir.

- **Aprendizado supervisionado.** No aprendizado supervisionado a RNA recebe a saída correta correspondente a cada dado de entrada. Assim, a rede pode aprender um mapeamento entre entradas e saídas, por meio do ajuste de seus pesos, visando à produção de respostas próximas à saída desejada ou alvo. Este é o tipo de aprendizado implementado, por exemplo: em MLPs com o algoritmo *BP*, em redes LVQ (*Learning Vector Quantization*) (FAUSETT, 1994), e em redes da classe ARTMAP (Seção 3.3.3).

- **Aprendizado por reforço (*Reinforcement learning*)**. Conforme Hertz, Krogh e Palmer (1989), este pode ser considerado como uma modalidade especial de aprendizado supervisionado por meio de um "crítico". Neste caso, a rede não recebe explicitamente a saída correta do ambiente, mas uma resposta de reforço ou de inibição, mediante o resultado de sua ação. Este tipo de aprendizado tem sido adotado em aplicações para controle de sistemas e de processos.

- **Aprendizado não-supervisionado**. No aprendizado não-supervisionado, em lugar de receber a saída desejada associada a cada dado de entrada, a rede *auto-organiza* os padrões recebidos, podendo formar agrupamentos (*clusters*) em que são alocados os padrões que têm alguma relação de similaridade entre si. Este aprendizado possibilita a descoberta de regularidades no espaço de entradas quando não há conhecimento da resposta correta (ou do rótulo) correspondente a cada padrão de entrada apresentado. Mapas auto-organizáveis SOM de *Kohonen* (HAYKIN, 1999) e VQ (*Vector Quantization*) (HERTZ, KROGH e PALMER, 1991), assim como redes ART1, ART2, ART2-A, ART3 e fuzzy ART (Seção 3.3.2), são exemplos de redes em que opera um aprendizado não-supervisionado.

- **Aprendizado híbrido**. O paradigma híbrido combina os tipos de aprendizado supervisionado e não-supervisionado para treinamento de uma RNA. Em uma rede com funções de base radial (RBF) pode ser utilizado um aprendizado não-supervisionado (*Kohonen*, por exemplo) para os neurônios na camada escondida da rede, sendo que o mapeamento para a camada de saída pode ser realizado, de modo supervisionado, por meio do cálculo da pseudo-inversa, por exemplo, (BISHOP, 1999).

#### 3.2.1.3 Tipo de Aprendizado

O algoritmo de aprendizado define o procedimento e as regras segundo as quais os pesos conectivos de uma rede neural são ajustados. Existem diversos algoritmos de aprendizado (e suas variantes), cada qual apropriado a um modelo de RNA e à função a ser desempenhada por esta. Segundo Haykin (1999), no desenvolvimento de uma RNA podem ser adotados os tipos de aprendizado: baseado na regra de *Hebb*, baseado em memória, o aprendizado de *Boltzmann* (HERTZ, KROGH e PALMER, 1991), além do aprendizado por correção de erro e do aprendizado competitivo, sendo estes dois últimos descritos a seguir.

- **Aprendizado por correção de erro**. Consiste em um procedimento para aprendizado supervisionado, por meio do qual os pesos entre neurônios são atualizados em proporção ao erro entre a saída desejada e a saída efetiva de cada neurônio da camada de saída. Os ajustes iterativos de correção de erro procuram aproximar o sinal de saída efetivo ao sinal de saída desejada, minimizando uma função de custo, baseada no erro, utilizando, comumente, a regra delta (ou regra de *Widrow-Hoff*) para ajuste dos pesos. O algoritmo *Backpropagation* (*BP*), utilizado em redes MLP, é baseado no conceito da correção de erro, de modo que o erro da camada de saída é *retropropagado* a uma camada escondida, sendo utilizado para cálculo do gradiente local para cada neurônio escondido na atualização dos seus pesos. O *BP* é um dos algoritmos mais populares, sendo descrito na maioria dos textos que versam sobre RNAs como, por exemplo, em (HERTZ, KROGH e PALMER, 1991; FAUSETT, 1994; PANDYA e MACY, 1996; SKAPURA, 1996; BISHOP, 1999; HAYKIN, 1999).

- **Aprendizado competitivo**. O aprendizado competitivo comporta um mecanismo por meio do qual as unidades neurais disputam entre si para definir qual será ativada diante de um padrão de entrada apresentado. Cada neurônio de uma camada competitiva tem armazenado em seus pesos o denominado protótipo do *cluster* (aglomerado formado pelos atributos relevantes de padrões aprendidos por este). Diante da apresentação de um padrão de entrada, os neurônios competem entre si, sendo que o neurônio (ou alguns poucos

neurônios) com a uma melhor resposta suprime a atividade dos demais, aprendendo por meio da modificação de seus pesos. Assim, os padrões apresentados tendem a ser agrupados por meio de aprendizado nos *clusters* com os quais encontram maior similaridade. O caso extremo de aprendizado competitivo é ditado pela regra *WTA* ("*Winner-Takes-All*"), por meio da qual o padrão de entrada é aprendido apenas pelo neurônio que apresenta a melhor resposta a este. O aprendizado competitivo é utilizado, por exemplo, em mapas auto-organizáveis de *Kohonen* e em redes ART (Seção 3.3).

### 3.2.2 Considerações sobre Aprendizado e Execução de RNAs

Para implementação, uma vez definidos os requisitos da aplicação e levantadas as características do ambiente de dados, é formada a base de dados a ser utilizada pela RNA. O conhecimento disponível do domínio do problema em questão pode ser utilizado em uma fase de pré-processamento dos dados de entrada, para normalização, extração de características, a redução da dimensionalidade do vetor de entradas ou o tratamento de dados incompletos, assim como orientar um pós-processamento das saídas da rede.

As definições iniciais para uma RNA são norteadas pela tarefa a ser executada (classificação, aproximação de funções, predição seqüencial, formação de agrupamentos, otimização) que orienta a escolha por um modelo de rede, conforme o paradigma de aprendizado, tipo de conectividade e estrutura básica da rede, tipos de neurônios e suas respectivas funções de ativação, e o modo de operação (*off-line*; *on-line*) requerido no ambiente de operação. As especificações da composição do vetor de entrada e forma de representação dos dados (o que define o número de unidades de entrada dos dados), da representação da saída e do modo de interpretação das respostas da rede (o que define os neurônios de saída), e dos critérios e medidas para avaliação de desempenho da rede, são dependentes dos tipos de dados e do modo como a rede fornecerá as respostas decorrentes do processamento.

Para ilustrar alguns aspectos do projeto da estrutura, em termos do número de camadas e de neurônios, é considerado aqui, o caso da forma tradicional de implementação de redes MLP, treinadas por meio do algoritmo *BP*. Estas redes têm sido amplamente utilizadas para classificação e para aproximação de funções em aplicações que envolvem: reconhecimento de padrões, sistemas preditivos, modelagem e controle de processos. Tipicamente, redes MLP apresentam uma estrutura em que o número de camadas e de neurônios precisa ser fixado antes de ser iniciado um processo de aprendizado, relegando ao projetista a seleção de uma

configuração estrutural adequada para a execução de uma determinada tarefa. Caso a estrutura comporte um número insuficiente de neurônios escondidos, a capacidade representacional da rede pode ser limitada, com repercussão negativa sobre seu desempenho. Por outro lado, uma estrutura formada por um excesso de neurônios escondidos pode especializar-se de tal maneira em relação ao conjunto de treinamento, que há grande perda na capacidade de generalização. Na prática, o projeto de uma rede MLP demanda a aplicação de estratégias de crescimento e poda até a obtenção de uma estrutura satisfatória para execução da tarefa em questão. Ressalta-se que, neste caso, para cada definição de estrutura, deve ser refeito o processo de aprendizado para ajuste de pesos por meio de todo o conjunto de treinamento.

Redes ART (Seção 3.3), em contrapartida, apresentam propriedades construtivas que possibilitam o aprendizado incremental de categorias (ou seja, os neurônios que representam as categorias) à medida que são apresentados os dados de entrada. Deste modo, pode ser definida uma memória inicial arbitrariamente grande, disponibilizada para a formação incremental de novas categorias, sem que seja necessário fixar previamente o número de neurônios na correspondente camada representacional.

#### 3.2.2.1 Treinamento, Testes e Execução de RNAs

A preparação tradicional de uma RNA para atuação no ambiente de aplicação envolve um processo de treinamento e a realização dos testes para verificação de seu desempenho segundo os critérios definidos. A base dos dados a serem apresentados à rede é dividida em um conjunto de *treinamento* e um conjunto de *teste*, sendo estes independentes entre si. Um conjunto de dados adicional para *validação* pode ser aplicado sobre configurações de redes treinadas, em um processo de *crossvalidation* (RIPLEY, 1996; HAYKIN, 1999) para seleção do modelo final a ser submetido ao conjunto de *teste*.

O conjunto de treinamento é utilizado no processo de aprendizado por meio do qual são obtidos os pesos conectivos para desempenho satisfatório de uma determinada tarefa. Durante o aprendizado, uma *época* corresponde a uma apresentação do conjunto de treino inteiro. No denominado modo *batch* (HAYKIN, 1999; BISHOP, 1999), a atualização dos pesos é feita após cada passada completa de um lote de padrões de entrada. No modo de atualização *por padrão* (também denominado de incremental), o ajuste dos pesos da rede é realizado após cada padrão apresentado. Em um processo de aprendizado *off-line*, o conjunto de dados é armazenado de modo que os padrões possam ser novamente reapresentados durante o treinamento. Já, no aprendizado *on-line*, cada padrão pode ser descartado após seu processamento e a atualização dos pesos da rede neural.

Podem ser treinadas configurações diferentes para uma rede neural, em que, por exemplo, cada configuração pode ter um número diferente de unidades escondidas. Um conjunto de validação é aplicado sobre cada uma destas configurações para seleção da mais adequada, segundo os critérios de avaliação definidos. Passada esta fase, a rede neural selecionada deve ser submetida a um conjunto de teste para aferição final de seu desempenho. Enfatiza-se que o conjunto de teste deve ser formado por casos novos para a rede, caso contrário, a medida de desempenho alcançada não será confiável o bastante para considerar sua adequação à aplicação em foco.

**Treinamento *off-line***. Para muitas aplicações, uma rede pode ter sido projetada para responder ao ambiente do problema apenas com base no treinamento *off-line* pelo qual passou, sem adaptar-se a dados novos durante a execução propriamente dita neste ambiente. Assim, se o ambiente de dados sofrer mudanças significativas, a rede deve passar por um novo processo de treinamento para o aprendizado dos novos padrões. Por exemplo, a rede pode ter sido treinada para uma determinada tarefa de reconhecimento de padrões e, caso sejam incorporados novos exemplos à base de dados, causando grandes mudanças na sua distribuição probabilística, é preciso realizar um novo treinamento por meio da nova formação da base para que a rede seja capaz de generalizar satisfatoriamente durante a execução. Este processo costuma ser necessário, por exemplo, no caso de uma rede MLP treinada por meio do algoritmo *Backpropagation* (BP).

**Aprendizado *on-line*.** Em determinadas aplicações, porém, é requerida a adaptação a ambientes mutáveis, não-estacionários ou ambientes em que as entradas são apresentadas seqüencialmente (à medida que são feitas as observações), de modo que a rede precisa ajustar seus pesos para o aprendizado *on-line* dos novos padrões recebidos, durante a execução. A capacidade de aprendizado *on-line* é importante, por exemplo, para a operação de sistemas adaptativos autônomos. Neste caso, a rede pode ter passado por um processo de treinamento prévio, mas deve ser capaz de aprender novos dados enquanto opera ou interage com o ambiente. Em RNAs com características construtivas, como redes ART (Seção 3.1) novas categorias podem ser aprendidas incrementalmente, durante o processo de aprendizado de novos padrões, possibilitando sua adaptação às mudanças no ambiente de dados do problema.

Para vários modelos de RNAs um treinamento *off-line* chega a envolver um grande número de épocas (dezenas ou mesmo milhares, conforme o caso), podendo demandar um longo tempo de processamento. No entanto, uma vez suspenso o aprendizado, o tempo necessário para a fase de teste tende a ser consideravelmente menor. Ressalta-se que o tempo envolvido em cada uma destas fases depende do modelo de RNA, de suas especificações, da

implementação realizada, do ambiente de dados, e da necessidade ou não de adaptação através de aprendizado, ao longo do processo de execução.

A seguir, é apresentada a família de redes ART que se distinguem em diversos aspectos de outros modelos de RNAs como, redes MLP treinadas por meio de *BP*. Em redes ART novos neurônios podem ser alocados para representar novas categorias durante o aprendizado de padrões de entrada, em contraste com o necessário pré-dimensionamento do número de camadas e de unidades escondidas, como em uma rede MLP treinada por meio de BP. Geralmente, para redes ART são necessárias poucas épocas de aprendizado para alcance de bom desempenho, contrastando com redes MLP treinadas por meio de *BP*, que tendem a necessitar de um grande número de épocas até alcançar a convergência.

## 3.3 RNAs Baseadas na Teoria da Ressonância Adaptativa - *ART*

A família de RNAs conhecidas, genericamente, por redes ART, originou-se da Teoria da Ressonância Adaptativa (*Adaptive Resonance Theory - ART*). Esta família engloba modelos que realizam um aprendizado não-supervisionado (por exemplo: ART1, ART2, ART2A, ART3, *fuzzy* ART, dART) e modelos com aprendizado supervisionado (por exemplo: ARTMAP, *fuzzy* ARTMAP, *gaussian* ARTMAP, ART-EMAP, ARTMAP-IC, *d*ARTMAP).

A Teoria da Ressonância Adaptativa foi desenvolvida a partir de estudos sobre o processamento cognitivo da informação e a codificação estável em um ambiente de entradas complexo (GROSSBERG, 1976a; 1976b; 1980). Elaborada por Stephen Grossberg (GROSSBERG, 1976b), a teoria *ART* conjuga a análise de aspectos matemáticos, psicológicos e neurofisiológicos de processos como: percepção, cognição e aprendizado (GROSSBERG, 1987; 1988). Conforme Carpenter e Grossberg (1988), *ART* foi introduzida para a modelagem de um aprendizado competitivo em uma estrutura de controle auto-regulada, cujo aprendizado e reconhecimento autônomo poderiam realizar-se de maneira estável e eficiente, em resposta a uma seqüência arbitrária de padrões de entrada. *ART* consiste em uma extensão do aprendizado competitivo com uma solução para o dilema de *estabilidade-plasticidade* que pode ser expresso por meio das seguintes questões, aqui traduzidas da versão original de Carpenter e Grossberg (1988, p.77):

> De que maneira um sistema de aprendizado pode ser desenvolvido para manter-se plástico, ou adaptativo, em resposta a eventos significativos, e ainda conservar-se estável em resposta a eventos irrelevantes? Como este sistema pode decidir quando comutar entre o modo estável e o plástico para alcançar estabilidade sem rigidez, e plasticidade sem caos? Como o sistema pode preservar o conhecimento previamente adquirido enquanto continua a aprender fatos novos? O que evita que o novo aprendizado apague conhecimentos do aprendizado anterior?

Os conceitos básicos da Teoria da Ressonância Adaptativa conduziram à concepção da rede neural artificial ART1 (CARPENTER e GROSSBERG, 1987a), precursora de uma família de arquiteturas baseadas em *ART* desenvolvidas, posteriormente, por estes e por outros autores. RNAs ART podem executar tarefas de categorização, classificação, predição e reconhecimento de padrões, tendo sido adotadas em aplicações que requerem aprendizado incremental no modo *on-line* e *off-line*. Em síntese, estas redes procuram solucionar, operacionalmente, o dilema de *estabilidade-plasticidade* em resposta às seqüências arbitrárias de entradas, mantendo a estabilidade do conhecimento relevante já representado na rede, juntamente com a capacidade adaptativa de aprendizado de novos padrões.

Ao longo da Seção seguinte são apresentados os conceitos que fundamentam as redes neurais ART, ilustrados por meio da rede ART1 que implementa aprendizado não-supervisionado. Na seqüência, são descritos, brevemente, outros modelos não-supervisionados (ARTs) e alguns modelos supervisionados (ARTMAPs), baseados em *ART*.

### 3.3.1 Conceitos de Redes Neurais Artificiais Baseadas em *ART*

Os objetivos fundamentais de *design* computacional de redes ART incluem estabilidade de memória com aprendizado rápido ou lento em ambientes de entradas estacionários e não-estacionários (CARPENTER e GROSSBERG, 1987a; 1987b). Como um modelo de processos dinâmicos em tempo real, uma rede neural artificial ART é definida em termos de equações diferenciais não-lineares para as quais são utilizadas soluções analíticas ou aproximações nos algoritmos desenvolvidos para implementação. Ressalta-se que, no contexto do presente trabalho, particularmente, nos Capítulos 4 e 5, a ênfase é dada em relação à descrição algorítmica para aplicação de uma rede ARTMAP.

**Plasticidade e Estabilidade.** Redes baseadas em *ART* incorporam um mecanismo de criação de novas categorias diante de dados completamente novos, característicos de um ambiente dinâmico, permitindo que a rede se adapte por meio de plasticidade, mas mantenha sua estabilidade, sem "esquecer" drasticamente o que já aprendeu. A capacidade de aprender novos padrões sem esquecer os antigos é implementada com o auxílio de um mecanismo de *feedback* entre uma camada competitiva (formada por categorias) e a camada de entrada que representa os padrões apresentados. Estas camadas são conectadas por meio de pesos de *bottom-up* (*bu*), dirigidos da camada de entrada para a de categorias, e de *top-down* (*td*), dirigidos da camada de categorias para a de entradas. A camada de entrada funciona como "camada de comparação" entre o sinal de entrada e a expectativa interna de *td*. Este mecanismo possibilita a estabilização das categorias e o aprendizado de novas informações,

comutando entre o modo plástico e o estável. (Estes elementos são ilustrados na Figura 6 e descritos na Seção 3.3.1.1, no contexto da arquitetura de ART1). A estabilidade de uma rede baseada em *ART* está relacionada ao aprendizado que possibilita mudanças nos traços de memória apenas quando um fato assemelha-se, em determinado grau, às expectativas formadas internamente, ou diante de fatos completamente novos para a rede. Ao longo do aprendizado, diversos padrões de entrada podem ser codificados no vetor de pesos, formando um protótipo que representa as características relevantes dos padrões agrupados em uma categoria. No caso em que a rede auto-organiza seus padrões de entrada, as categorias também podem ser denominadas de *clusters* (agrupamentos). Conforme Fausett (1994), o vetor de peso associado ao *cluster* (*code vector*) pode ser entendido como um exemplar representativo dos padrões alocados neste.

**Vigilância, Ressonância e Aprendizado.** Redes baseadas em *ART* podem aprender os dados de entrada conforme sua similaridade com protótipos já formados internamente, ou acomodar conhecimento novo por meio da construção de novas categorias para representá-lo. As unidades na camada de categorias competem entre si pelo aprendizado de um padrão apresentado e, no caso da regra *WTA*, apenas a categoria com a maior ativação permanece ativa como candidata ao aprendizado deste padrão. Um critério de similaridade entre o padrão de entrada e o protótipo armazenado no vetor de pesos associado à categoria candidata deve satisfazer um *parâmetro de vigilância*, denotado por $\rho$, para que esta seja aceita para o aprendizado. O parâmetro de vigilância determina o grau de especificidade com que a rede categoriza os padrões, ou seja, o grau mínimo de similaridade exigido entre a entrada e o vetor de pesos da categoria. A suficiente similaridade entre estes conduz ao processo de *ressonância*, como resultado da propagação da informação entre a camada de comparação e a camada de categorias via os pesos adaptativos de *bu* e de *td*. Uma vez que uma categoria tenha sido selecionada e aceita para o aprendizado, os sinais de peso de *bu* e de *td* são mantidos por um período extenso, durante o qual entram em *ressonância*. Este é o período durante o qual os pesos sofrem mudanças e ocorre o aprendizado. Se não houver similaridade suficiente entre o padrão apresentado e o protótipo armazenado pela categoria candidata, esta é inibida pelo tempo de duração do padrão de entrada corrente e outra categoria é escolhida. Caso nenhuma das categorias sucessivamente selecionadas apresente suficiente similaridade com o padrão de entrada, uma nova é criada para representá-lo.

**Vigilância nos modelos com aprendizado não-supervisionado e supervisionado.** Nos modelos não-supervisionados (ARTs) o valor para $\rho$ é definido pelo usuário e mantém-se

constante durante o processo de aprendizado. Nos modelos supervisionados (ARTMAPs) que implementam o mecanismo de *match tracking*, ρ é controlado internamente pelo sistema, a partir do seu valor inicial definido. Para a obtenção de poucas categorias, mas com maior grau de generalização, define-se o parâmetro de vigilância com um valor baixo. Utilizando-se um parâmetro de vigilância com valor mais alto, o grau de similaridade exigido entre os padrões alocados será maior, gerando um maior número de categorias.

**Criação incremental de categorias.** Inicialmente, pode ser estabelecido na memória da rede um número máximo de *clusters* ou de categorias disponíveis para a categorização dos padrões de entrada. De modo simplificado, para criação incremental de categorias um primeiro padrão apresentado é alocado em uma categoria que o representará; o padrão seguinte será comparado com a categoria já formada e, caso atenda ao critério de similaridade, será incorporado a esta; caso contrário, uma nova categoria será criada para representar este novo padrão. Este processo segue até ter sido aprendido todo o conjunto de dados. As categorias utilizadas ficam *comprometidas* com os padrões aprendidos e, as que ainda não foram recrutadas, mantêm-se *não-comprometidas*, ou seja, disponíveis para representar novos protótipos. Padrões que não encontram suficiente similaridade com o conhecimento já armazenado na rede não são forçados a encaixar-se em alguma categoria. Conforme são recebidos novos padrões de entrada, novas categorias podem ser recrutadas até que o conjunto de unidades não-comprometidas seja esgotado. A estratégia a ser adotada para o caso de esgotamento da memória da rede é definida pelo projeto no contexto da aplicação, uma vez que a arquitetura da rede, em si, não impõe a pré-fixação do número de categorias.

**Aprendizado *rápido* ou *lento.*** A operação das redes ART pode ser realizada no *modo de aprendizado rápido* ou no *modo lento*, conforme cada modelo e as características de categorização objetivadas. Durante o aprendizado no *modo rápido* assume-se que a atualização de pesos, durante a ressonância, ocorre rapidamente em relação à duração do tempo em que um padrão de entrada é apresentado em uma tentativa de aprendizado, de modo que a estabilização dos pesos pode ocorrer em uma ou poucas épocas. Já, durante o *modo lento*, as mudanças de peso ocorrem lentamente em relação à duração da apresentação do padrão, os pesos não alcançam equilíbrio em uma tentativa de aprendizado, e é necessário um número maior de apresentações de cada padrão, até haver estabilização de aprendizado.

#### 3.3.1.1 Rede ART1

Os conceitos-chave da Teoria da Ressonância Adaptativa podem ser ilustrados por meio da descrição da seqüência do ciclo de "teste de hipóteses" que se processa em ART1, primeira

RNA desenvolvida com base em ART. A rede ART1 auto-organiza padrões de entrada binários em categorias de reconhecimento através de aprendizado não-supervisionado. A descrição da arquitetura de ART1 (Figura 6), e os diagramas que descrevem a seqüência do ciclo de "teste de hipóteses" que se processa em ART1 a partir da apresentação de um padrão de entrada (Figuras 7(a-d)), são baseados nos esquemas expostos em (CARPENTER e GROSSBERG, 1987a; 1987b; 1988), tendo sido adaptados para o exemplo aqui apresentado.

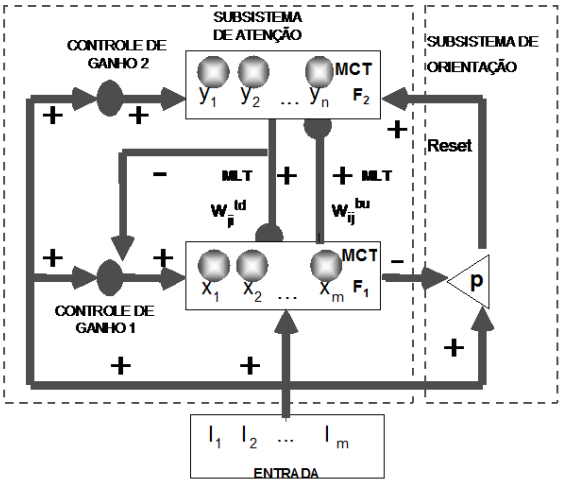


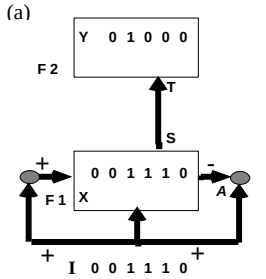

Figura 6: Modelo ART1, adaptado de (CARPENTER e GROSSBERG, 1987a; 1988).

- A camada $F_1$ é formada por *m* nós que respondem às características do padrão de entrada, de modo que cada *i*-ésimo nó de $F_1$ recebe um valor do componente da entrada apresentada, sendo $i$=1, 2, ..., $m$. A camada $F_2$ é composta pelos *n* nós *j* que representam as categorias que são formadas ao longo do aprendizado, sendo, portanto, $j$=1, 2, ..., $n$..
- As camadas $F_1$ e $F_2$ são amplamente conectadas com pesos adaptativos (representados por meio das flechas cheias). Os pesos $w_{ij}^{bu}$ de *bu* (*bottom-up*) são dirigidos de um nó *i* de $F_1$ para um nó *j* de $F_2$ e os pesos $w_{ji}^{td}$ de *td* (*top-down*), que realizam o *feedback*, são dirigidos de um nó *j* de $F_2$ para um nó *i* de $F_1$. Os caminhos de *bu* e de *td* entre $F_1$ e $F_2$ contêm os traços adaptativos de *memória de longo termo* (MLT) que multiplicam os sinais nestes caminhos. Durante o aprendizado, os pesos adaptam-se aos novos padrões de entrada, codificando os traços de MLT que armazenarão o conhecimento da rede em relação aos padrões aprendidos.
- As camadas $F_1$ e $F_2$ do subsistema de atenção codificam os padrões de ativação nos traços de *memória de curto termo* (MCT) que existem apenas temporariamente, durante a apresentação de cada vetor de entrada. A camada $F_1$ equivale a uma "camada de comparação" entre o padrão de entrada e a expectativa de *td*, cujo grau de similaridade é determinado pelo parâmetro de vigilância ρ.

- O subsistema de atenção permite que as unidades de processamento de $F_1$ sejam acionadas apenas quando um padrão de entrada é apresentado. O subsistema de orientação remove as unidades de processamento de $F_2$ do conjunto de vencedores possíveis por meio de uma operação de *reset* (ver Figura 7c).
- A modulação dos processos de MCT e de MLT é realizada pelo *controle de ganho* que habilita $F_1$ a distinguir entre padrões de *bottom-up* e os padrões de expectativa de *top-down*, assim como a realizar a comparação entre estes, segundo a *regra dos 2/3* (dois-terços). A *regra dos 2/3* é necessária para a regulagem do ciclo de "teste de hipóteses" e a auto-estabilização de aprendizado em um sistema ART1. Em síntese, *a regra dos 2/3* estabelece que ao menos duas, dentre as três fontes de sinais, ativem um nó de $F_1$, sendo estas: uma entrada de *bottom-up* (*bu*) uma entrada de *top-down* (*td*) e uma entrada de controle de ganho. A *regra dos 2/3* pode ser ilustrada por meio das Figuras 7(a-d), que descrevem o processo de comparação entre um padrão de entrada binário e protótipos já armazenados na rede.

Conforme as Figuras 7(a) e 7(d), apenas os nós de $F_1$ que recebem entradas de *bu* e do controle de ganho ficam ativas e, de acordo com a Figura 3.3b, quando um padrão de entrada de *bu* e um protótipo de *td* estão simultaneamente ativos, apenas os nós que recebem entradas de ambos podem ser ativados. Os sinais de controle de ganho também habilitam $F_2$ a reagir aos sinais vindos de $F_1$, enquanto um padrão de entrada está ligado enquanto um padrão de entrada está ligado. O subsistema de orientação gera um sinal de *reset* para $F_2$ quando os padrões de *bu* e de *td* não forem confirmados em $F_1$, como na Figura 7(c). Este sinal de *reset* inibe seletivamente as unidades de $F_2$, anteriormente ativas, até a entrada ser desligada.

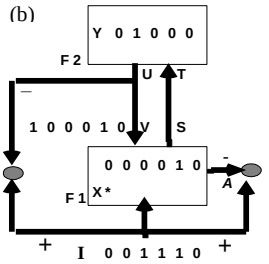
O denominado "teste de hipóteses", iniciado ao ser apresentado um padrão de entrada, é repetido, automaticamente, até que uma das seguintes situações ocorra: é selecionado um nó em $F_2$, cuja expectativa de *td* **V** é suficientemente similar ao padrão de entrada **I** ou é recrutado um nó não-comprometido de $F_2$. Ao ser satisfeita uma destas condições, o "teste de hipóteses" é finalizado e a rede entra em ressonância para o aprendizado do padrão de entrada. Caso a capacidade da rede tenha sido exaurida, de modo que não haja mais nós não-comprometidos para o aprendizado de um novo padrão de entrada, este não poderá ser acomodado na rede. Uma das seguintes medidas pode ser adotada nesta situação: o acréscimo de mais nós à memória do sistema, ou a operação da rede utilizando um parâmetro de vigilância mais baixo, ou a designação do padrão de entrada em questão como um elemento que não pode ser classificado conforme os parâmetros da rede.




O padrão de entrada **I** gera um padrão de atividade **X** de MCT em $F_1$ e ativa o controle de ganho de $F_1$ e o subsistema de orientação ***A***.

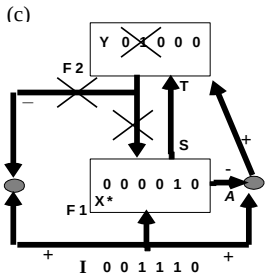
O padrão **X** inibe ***A*** e gera o padrão de saída **S** transformado pelos traços de filtros adaptativos de MLT, no padrão de entrada **T** que ativa o padrão de atividade **Y** em $F_2$. Ou seja, $\mathbf{T}=\mathbf{W}^{bu}\mathbf{S}$, em que $\mathbf{W}^{bu}$ é a matriz de traços de *bottom-up* de MLT.

As camadas $F_1$ e $F_2$ são amplamente conectadas entre si mas, após a competição "vencedor-leva-tudo" em $F_2$, apenas o nó de $F_2$ com maior entrada ponderada é ativado. O vetor de atividade **Y** representa apenas o nó que recebe a máxima entrada de $F_1 \rightarrow F_2$ de MCT em $F_2$.



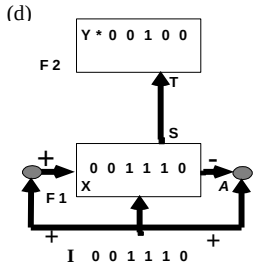

O padrão **Y** gera o padrão de saída de *top-down* **U** de MLT, transformado pelo filtro adaptativo $\mathbf{V} = \mathbf{W}^{td}\mathbf{U}$ no padrão de expectativa **V**, sendo que $\mathbf{W}^{td}$ é a matriz de traços de *top-do*wn de MLT,.

**Y** inibe o controle de ganho de modo que, quando **V** é comparado a **I**, apenas a porção de $F_1$ que representa corretamente **V** permanece ativa. Se **V** não confirmar **I** em $F_1$, um novo padrão de atividade de MCT, denominado **X*,** é gerado em $F_1$. **X*** representa a porção ativa do padrão **X**. A redução na atividade total MCT que ocorre quando **X** é transformado em **X*** causa uma diminuição na inibição total de $F_1$ para ***A***.




O parâmetro de vigilância $\rho$ determina o grau de similaridade exigida entre a expectativa **V** e o padrão **I**.

Se o critério de vigilância não for satisfeito, ***A*** envia um sinal de *reset* a $F_2$ e inibe o padrão de MCT **Y** em $F_2$ para esta rodada de apresentação. A inibição de **Y** leva, também, à remoção da expectativa de *top-down* **V**.




Após a inibição de **Y**, sua expectativa de *top-down* é eliminada e **X** pode ser reinstalado em $F_1$. Assim, **X** gera, novamente, um padrão de entrada **T** para $F_2$ mas, como **Y** continua inbido, **T** pode ativar um padrão de MCT **Y*** diferente em $F_2$. Se a expectativa de *top-down* devida a **Y*** também desconfirmar **I** em F1, a busca rápida continuará até ser encontrada uma categoria apropriada em $F_2$.

Quando a expectativa de *top-down* devida a **Y*** confirmar uma suficiente similaridade com **I**, satisfazendo $\rho$, a rede entrará em ressonância para o aprendizado de **I**, atualizando os pesos $W^{bu}$ e $W^{td}$ das conexões do nó vencedor de $F_2$.

Figura 7 (a - d): Seqüência de "teste de hipóteses" em ART1, adaptado de (CARPENTER e GROSSBERG, 1987a; 1988) para o exemplo apresentado.

As atualizações de pesos de *bottom-up* e *top-down* são controladas por equações diferenciais. Porém, como na implementação algorítmica de ART1 é assumido que este atua no modo de aprendizado rápido, as soluções iterativas destas operações não são necessárias. Isto se deve ao alcance de equilíbrio dos pesos durante cada apresentação de padrão e à invariância das ativações de $F_1$ durante a fase de ressonância, o que permite a determinação exata dos pesos de equilíbrio (FAUSETT, 1994).

A rede ART1 pode alocar um número arbitrário de padrões em categorias e realiza uma codificação estável, restringindo-se, porém, à classificação de padrões binários. ART1 é

adequada para aplicações que requerem codificação imediata de representações binárias, tendo sido aplicada em áreas como processamento de imagem digital e em sistemas de recuperação de informação. Um dos exemplos bem-sucedidos de aplicação de ART1 é o sistema de recuperação de partes de *design* em engenharia, utilizado pela BOEING (Quadro 1, Seção 3.3.5).

### 3.3.2 Modelos ART com Aprendizado Não-Supervisionado - ARTs

Os modelos *ART* com aprendizado não-supervisionado possibilitam a auto-organização de padrões em agrupamentos (*clusters*), conforme um critério interno de similaridade, sem recebimento de uma instrução externa sobre a saída correta associada à entrada recebida. Após a concepção de **ART1** (CARPENTER e GROSSBERG, 1987a), outras redes com aprendizado não-supervisionado, baseadas em *ART*, foram desenvolvidas, sendo algumas destas listadas a seguir.

- **ART2** (CARPENTER e GROSSBERG, 1987b). Categoriza padrões de entrada binários e analógicos. Sua estrutura básica é semelhante à de ART1, sendo que a camada $F_1$ é substituída por várias subcamadas, devido ao processamento de vetores analógicos. Além da comparação entre o vetor de entrada e a expectativa, requerida pelo subsistema de orientação, estas subcamadas promovem a normalização e a supressão de ruído.
- **ART3** (CARPENTER & GROSSBERG, 1990). Realiza busca paralela de códigos de reconhecimento distribuídos em uma rede hierárquica *multi-nível* e incorpora um paradigma de transmissores químicos para controlar os processos de busca. Até o momento, não se tem notícia de aplicações deste modelo de rede ART.
- **ART2A** (CARPENTER, GROSSBERG e ROSEN, 1991a). Consiste em uma versão eficiente do algoritmo de aprendizado de ART2 (cerca de duas a três ordens de grandeza mais rápido que do algoritmo de ART2).
- **Fuzzy ART** (CARPENTER, GROSSBERG e ROSEN, 1991b). A rede fuzzy ART generaliza a rede ART1, possibilitando o aprendizado de padrões binários e analógicos ao modificar os operadores de intersecção *crisp* (com valores definidos no conjunto {0, 1}) de ART1 por operadores de conjuntos *fuzzy* (com valores definidos no intervalo [0, 1]) nas regras de operação da rede. Sua arquitetura resulta em uma operação bem mais simples do que a que se processa em uma rede ART2. Este modelo é um módulo componente básico da rede fuzzy ARTMAP, apresentada no Capítulo 4.

- **dART** (CARPENTER, 1997). Opera por meio de uma regra de aprendizado que possibilita a representação distribuída dos códigos na rede, permitindo a adaptação rápida, sem incorrer no denominado "esquecimento catastrófico". O esquecimento catastrófico ocorre em modelos de redes com aprendizado distribuído quando o aprendizado de um novo padrão erode drasticamente a memória dos padrões anteriormente representados na rede. Em teoria, os modelos ART podem realizar ativação distribuída durante o aprendizado, mas, na prática, costumam utilizar a regra WTA. Em dART a ativação distribuída é adotada durante as fases de aprendizado e de teste, melhorando sua tolerância ao ruído e a redução do número de categorias necessárias para codificação de padrões na rede. Adicionalmente, uma nova dinâmica mantém a capacidade de aprendizado rápido e estável, encontrada nos modelos que operam no modo WTA. Ao operar no modo WTA, a rede dART equivale, computacionalmente, a um fuzzy ART. Esta rede é utilizada como um módulo da arquitetura supervisionada dARTMAP (Seção 3.3.2.2).

### 3.3.3 Modelos ART com Aprendizado Supervisionado - ARTMAPs

Um sistema ARTMAP é constituído por dois subsistemas de redes ART integrados por meio de um mapa associativo para aprendizado supervisionado da relação entre padrões de entrada categorizados em um módulo ARTa, e seus correspondentes alvos, categorizados em um módulo ARTb. Diversos modelos foram desenvolvidos a partir desta estrutura básica, sendo a rede neural ARTMAP (CARPENTER, GROSSBERG e REYNOLDS, 1991), formada por dois módulos ART de processamento de dados binários, a precursora destes. Os módulos ART são capazes de auto-organizar categorias de reconhecimento estáveis em resposta a seqüências arbitrárias de padrões de entrada.

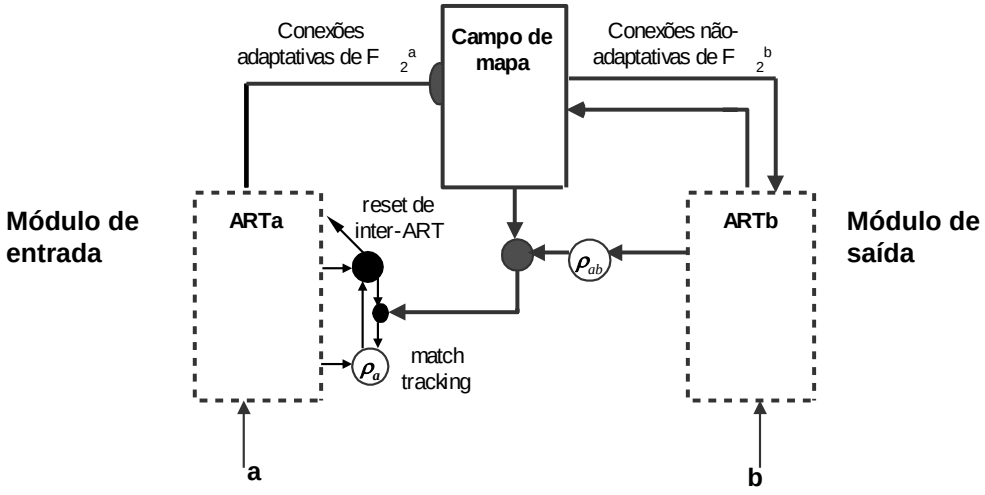



Figura 8: Estrutura básica de ARTMAP

Durante o aprendizado supervisionado em um modelo ARTMAP (ilustrado por meio da Figura 8), o módulo ARTa recebe um fluxo de padrões de {**a**(*n*)} do espaço de entradas e o módulo ARTb recebe padrões {**b**(*n*)} do espaço de saídas, em que *n* designa o índice do padrão, sendo que **b**(*n*) corresponde à predição correta, dado um padrão **a**(*n*). O módulo denominado de campo de mapa (inter-ARTs) associa categorias formadas no módulo de entrada ARTa com categorias formadas no módulo de saída ARTb. As categorias de ARTa são ligadas via conexões adaptativas com pesos ajustáveis aos nós do campo de mapa que, por sua vez, possuem conexões sem pesos nas ligações um-a-um com os correspondentes nós de categorias de ARTb.

No início de cada apresentação de uma entrada **a**, o parâmetro de vigilância $\rho_a$ assume um valor inicial, definido pelo usuário, e que determina o mínimo critério de similaridade aceito pelo sistema entre **a** e uma categoria ativada em ARTa. O valor de $\rho_a$ é controlado autonomamente pelo sistema por meio de um processo denominado de *match tracking* (*MT*) e retorna ao seu valor de base ao término deste. A categoria escolhida em ARTa que satisfizer o critério de similaridade com **a** deve aprender a predizer a classe ativada em ARTb, dado o padrão-alvo **b**. Caso a predição realizada por ARTa não seja confirmada pelo parâmetro de vigilância $\rho_{ab}$ do campo de mapa, é acionado o *MT* em ARTa para correção deste erro preditivo. Em conseqüência do *MT*, a categoria atual de ARTa é inibida por meio do mecanismo de *reset*, e é iniciada uma busca por outra categoria que satisfaça a associação com o alvo em ARTb e, na ausência desta, uma nova é recrutada para o aprendizado do padrão de entrada. A categoria de ARTa que satisfizer as condições descritas codifica **a** e aprende sua associação com a classe alvo de ARTb. A auto-organização dos padrões-alvo em ARTb, cujo processo é similar ao de ARTa, obedece ao critério de similaridade definido pelo parâmetro de vigilância $\rho_b$, mas que é mantido fixo durante todo o processo de aprendizado supervisionado.

No modo de predição de teste, apenas ARTa recebe padrões de entrada. Não há *MT* nesta fase e a categoria escolhida em ARTa fornecerá diretamente sua resposta de predição para **b,** por intermédio de suas conexões de pesos ao campo de mapa, com base nas associações já aprendidas. Ressalta-se que o modelo ARTMAP apresentado opera com ativação WTA nas camadas de categorias de cada módulo ART e no campo de mapa, tanto durante o modo de aprendizado, quanto no modo de teste.

Após a concepção de ARTMAP com módulos ART1, outras variantes que operam com aprendizado supervisionado foram desenvolvidas, sendo algumas destas descritas a seguir.

Os modelos apresentados pertencem à classe denominada, genericamente, de ARTMAP. Sua modularidade é definida segundo um *framework* geral que une dois subsistemas de categorização através de um módulo de ligação para o aprendizado de associações entre entradas e saídas corretas (ou alvos). Nos casos em que não há auto-organização dos padrões de saída **b**, não é necessário o módulo de saída ARTb. Em lugar deste, utiliza-se uma versão simplificada da rede ARTMAP em que os padrões de entrada são auto-organizados em ARTa e associados às saídas que podem ser definidas por $\mathbf{b} = (b_1, \ldots b_k, \ldots b_{Nb},)$ em que $k$=1, 2 ..., $Nb$ designa o índice dos nós de saída.

As diversas variantes de ARTMAP procuram (cada qual com suas especificidades) apresentar soluções mais apropriadas para aspectos que emergem do domínio de uma determinada aplicação. Algumas das redes da classe ARTMAP são descritas na seqüência.

- **Fuzzy ARTMAP** (CARPENTER et al, 1992). Fuzzy ARTMAP foi concebida como uma extensão de uma rede ARTMAP, substituindo os módulos ART1 por módulos de redes fuzzy ART, o que possibilita a categorização de padrões binários e analógicos. Esta rede neural será descrita com mais detalhes no Capítulo 4.

- **ART-EMAP** (CARPENTER e ROSS, 1995). Consiste em uma extensão de fuzzy ARTMAP, sendo que sua arquitetura permite acumular evidências espaço-temporais para reconhecimento de objetos-alvo e classes de padrões em situações de ruído ou ambigüidade. No módulo inter-ARTs, diversos estágios possibilitam a realização de classificação por evidências coletadas dos padrões. O aprendizado em ART-EMAP ocorre no modo WTA. Porém, durante a fase de predição, a rede pode operar com ativação distribuída, em que diversas categorias participam para a formação de uma resposta de saída. Sua aplicação foi testada e bem-sucedida no reconhecimento de objetos tridimensionais (3D) a partir de vistas bidimensionais (2D), superando a rede fuzzy ARTMAP em acurácia nesta tarefa.

- **PROBART** (MARRIOT e HARRISON, 1995). Consiste em uma modificação de fuzzy ARTMAP em que o campo de mapa acumula as freqüências de associação entre fuzzy ARTa e fuzzy ARTb, em lugar dos pesos destas ligações, durante o aprendizado. A resposta de predição de teste é dada em função de um mapa probabilístico obtido por meio destas freqüências. Este modelo procura sanar as limitações de fuzzy ARTMAP para aproximação incremental de mapeamentos com ruído. Não há atuação do *match tracking*, $\rho_{ab}$ não é requerido, de modo que toda associação feita entre ARTa e ARTb é computada, durante o aprendizado. O valor para o parâmetro de vigilância $\rho_a$ é mantido constante

evitando que, diante de padrões com ruído, sejam criadas categorias específicas para representá-los em fuzzy ARTa. Conforme os resultados reportados por Marriot e Harrison (1995, p.638) para uma tarefa de mapeamento complexo, ainda que PROBART tenha requerido um menor número de categorias para atingir um desempenho similar ao de fuzzy ARTMAP, não chegou a solucionar o problema de generalização. A abordagem por meio de PROBART melhorou o perfil de erro em tarefas de mapeamento com ruído, diminuiu o problema de proliferação de categorias associado à rede fuzzy ARTMAP (Capítulo 4), mas apresentou desempenho inferior a esta em tarefas de classificação.

- ***Gaussian* ARTMAP** (WILLIAMSON, 1996). Consiste em uma síntese de um classificador gaussiano e uma rede neural ART. O modelo utiliza categorias definidas por distribuições gaussianas no módulo *gaussian* ART que exerce um papel análogo ao de ART em uma ARTMAP. Durante o teste, a ativação é distribuída ao longo de todas as categorias que são mapeadas a cada predição. Conforme os experimentos relatados em (WILLIAMSON, 1996), *Gaussian* ARTMAP produz uma representação interna de categorias mais eficiente e é mais resistente a ruído que a rede fuzzy ARTMAP. Esta variante apresenta, no entanto, problemas de estabilidade no modo de aprendizado rápido.

- **ARTMAP-IC** (CARPENTER e MARKUZON, 1998). Esta é uma extensão de fuzzy ARTMAP, desenvolvida para trabalhar com casos inconsistentes (entradas iguais para saídas distintas) em modelagem de diagnóstico médico, tendo sido aplicada a uma base de dados com apenas duas classes de saídas possíveis. ARTMAP-IC realiza aprendizado via WTA com uma contagem de casos designados a cada categoria e implementa um *match tracking* com decremento de $\rho_a$, permitindo que entradas iguais sejam designadas a classes de saída diferentes. Durante a fase de teste é executada uma predição distribuída, em que as categorias que apresentam maior ativação são utilizadas para computar uma distribuição probabilística e a classe com a maior probabilidade designada fornece a resposta de saída.

- **dARTMAP** (CARPENTER, 1998). Foi desenvolvida para realização de aprendizado distribuído em uma estrutura ARTMAP com o objetivo de melhorar a tolerância a ruído e a capacidade de generalização, mantendo o aprendizado rápido e estável. A rede substitui os módulos ART por módulos dART e incorpora alguns recursos computacionais adicionais, tais como: memória endereçável por conteúdo, contagem de instâncias e um controle interno que permite ao sistema alternar entre o modo distribuído e o WTA. Se o aprendizado e a previsão ocorrerem no modo WTA, a rede passa a equivaler,

computacionalmente, a uma rede fuzzy ARTMAP. Ainda pouco explorada, a rede dARTMAP inclui vários parâmetros de operação a mais que fuzzy ARTMAP.

### 3.3.4 Sumário de Características Gerais de Redes Baseadas em *ART*

Existem muitas variações de redes baseadas em *ART*, com especificidades próprias, além daquelas relatadas aqui. Como características gerais associadas a redes ART, podem ser citadas:

- Aprendizado por comparação do grau de similaridade entre o padrão de entrada e o protótipo que representa uma categoria
- Capacidade de aprendizado incremental dos padrões de entrada
- Operação com modo de aprendizado rápido ou com modo de aprendizado lento
- Capacidade de aprendizado *on-line* e *off-line*
- Características construtivas por meio do recrutamento de novos neurônios para formação de categorias ou de agrupamentos (*clusters*) na rede
- Regulagem da granularidade na formação de categorias por meio de um parâmetro com valor definível pelo usuário
- Estabilidade na codificação de padrões já aprendidos e plasticidade para permitir o aprendizado de novos padrões e a formação de novas categorias
- Obtenção de boa acurácia em um número pequeno de épocas de aprendizado
- Modelos com aprendizado supervisionado e não-supervisionado

Novos algoritmos de aprendizado e modificações nas arquiteturas baseadas em *ART* partem das características genéricas associadas a estas redes para propor soluções que possam atenuar os problemas de sensibilidade a padrões com ruído, dependência à ordem de apresentação das entradas e tendência à proliferação de categorias, além de sua adequação aos domínios de aplicação específicos.

### 3.3.5 Aplicações de ARTs e ARTMAP

Redes ART e ARTMAP são apropriadas a problemas que requerem aprendizado *on-line* de grandes bases de dados em expansão. Modelos baseados em *ART* com aprendizado estável e autônomo têm sido utilizados para classificação e predição em ambientes mutáveis e em aplicações de larga escala, como: bases de dados médicas, sinais de sonar e de radar, sensoriamento remoto e robótica (CARPENTER, 1997). O Quadro 1 sistematiza alguns dos trabalhos que utilizaram modelos não-supervisionados (ARTs) e supervisionados (ARTMAPs) em aplicações de mundo real. Em grande parte das implementações e aplicações

reportadas na literatura especializada, estas têm sido utilizadas para realizar tarefas de classificação e agrupamento auto-organizado de padrões, assim como para predição e reconhecimento de padrões.

Quadro 1: Aplicações de ARTs e ARTMAPs

| Aplicação | Modelo de RNA ART | Referência | Ano |
|---|---|---|---|
| Classificação de caracteres chineses | ART1 | Kim, Jung, Kim, Kim, 1992 (apud CARPENTER, 1997) | 1992 |
| Reconhecimento Adaptativo de objetos em 3D a partir de vistas múltiplas | ART 2 | Seibert e Waxman, 1992 (apud CARPENTER, 1997) | 1992 |
| Controle sensório-motor de robôs | fuzzy ART | Bachelder, Waxman, Seibert, 1993 (apud CARPENTER, 1997) | 1993 |
| Reconhecimento de faces | ART 2<br>ART 2A | Seibert e Waxman, 1993 (apud CARPENTER, 1997) | 1993 |
| Predição de estrutura secundária de proteína | fuzzy ARTMAP | Mehta, Vij, Rabelo, 1993 (apud CARPENTER, 1997) | 1993 |
| Reconhecimento de sinais de eletrocardiograma | ART2<br>ART2A | Suzuki, Abe, Ono, 1993 (apud CARPENTER, 1997) | 1993 |
| Compressão de imagem (compressão com perda) | ART1<br>fuzzy ART | WU, SUNG, SOLIMAN,1993 | 1993 |
| Sistema neural de recuperação de informações de *design* em engenharia para a empresa BOEING | ART1 | Caudell, Smith, Escobedo, Anderson, 1994 (apud CARPENTER, 1997) | 1994 |
| Visão de máquina | LAPART (*laterally-primed* ART) | Caudell, Healy, 1994 (apud CARPENTER, 1997) | 1994 |
| Arquiteturas VIEWNET para reconhecimento de objetos em 3D a partir de múltiplas vistas em 2D | fuzzy ARTMAP | BRADSKI e GROSSBERG,1995 | 1995 |
| Aplicação em dispositivos de sistemas eletromagnéticos | fuzzy ARTMAP | Christodoulou, Huang, Georgiopoulos, Liou, 1995 (apud CARPENTER, 1997) | 1995 |
| Reconhecimento automático de alvos | ART2<br>ART2A | Koch, Moya, Hostetler, Fogler, 1995 (apud CARPENTER, 1997) | 1995 |
| Reconhecimento automático de alvos utilizando sinais de radar | fuzzy ARTMAP<br>ART-EMAP | RUBIN, 1995 | 1995 |
| Identificação de padrões em gráficos de controle estatístico de processos (CEP) | fuzzy ARTMAP<br>RBFfuzzyARTMAP | TONTINI, 1995 | 1995 |
| Classificação de arritmias cardíacas | fuzzy ARTMAP | HAM e HAN, 1996 | 1996 |
| Aplicação em tarefas de classificação de padrões na área médica | fuzzy ARTMAP | DOWNS, HARRISON, KENNEDY, CROSS, 1996 | 1996 |
| Classificação de Vegetação a partir de dados de Sensoriamento remoto | fuzzy ARTMAP | CARPENTER, GJAJA, GOPAL, WOODCOCK, 1997 | 1997 |
| Classificação de cobertura terrestre a partir de conjunto de dados AVHRR | fuzzy ARTMAP | GOPAL, WOODCOCK, STRAHLER, 1999 | 1999 |
| Classificação de dados em Nariz eletrônico | fuzzy ARTMAP | LLOBET, HINES, GARDNER, BARTLETT, MOTTRAM, 1999 | 1999 |
| Diagnóstico de falhas em linhas de transmissão multi-circuitos | fuzzy ARTMAP | AGGARWAL, XUAN, JOHNS, LI, BENNETT, 1999 | 1999 |

Ressalta-se que redes ART1 e fuzzy ART foram anteriormente utilizadas para compactar dados de imagens (WU, SUNG e SOLIMAN, 1993), empregando compressão *com perda*. No

entanto, não foram encontrados trabalhos anteriores em que redes das classes ART e ARTMAP tivessem sido aplicadas à compressão de *texto*, conforme constatado após uma pesquisa bibliográfica nas bases de dados de textos completos: IEEE, ACM, Pergamon-Elsevier, *Science* Direct, CiteSeer e na internet, até abril de 2000.

## 3.4 Redes Neurais Artificiais para Compressão de *Texto*

Aplicações de RNAs em compactação de imagens têm sido desenvolvidas com a utilização de redes baseadas em *Vector Quantization* (VQ) e de redes MLP, treinadas por meio de *Backpropagation*. Existem, também, conforme visto no Quadro 1, exemplos de redes não-supervisionadas ART1 e fuzzy ART, empregadas na compactação de imagem *com perdas*. No entanto, a utilização de RNAs para compressão de *texto*, que demanda a compressão *sem perda* de informação, não admitindo distorção nos dados reconstruídos, é um evento relativamente recente[15] e conta com poucos exemplos publicados.

A aplicação de RNAs para a compressão de texto parte da idéia de que estas possam ser adotadas para a formação de um modelo a partir do aprendizado dos padrões em uma seqüência de texto. O modelo aprendido é utilizado para estimar a distribuição probabilística para cada caracter que será codificado por meio de um codificador estatístico. A descrição da entrada de uma seqüência de texto para uma RNA genérica é ilustrada por meio da Figura 9. Uma janela deslizante de comprimento *r* percorre a seqüência de texto e, para cada seqüência de entrada ($x_{t-r}$, $x_{t-r+1}$,..., $x_{t-1}$), correspondente ao contexto formado pelos *r* últimos caracteres sucessivos lidos, a RNA realiza suas estimativas preditivas para o caracter $x_t$ que a sucede. A representação dos vetores de entrada e de saída, assim como a arquitetura, algoritmo e modo de operação da RNA, são definições dependentes da RNA implementada.

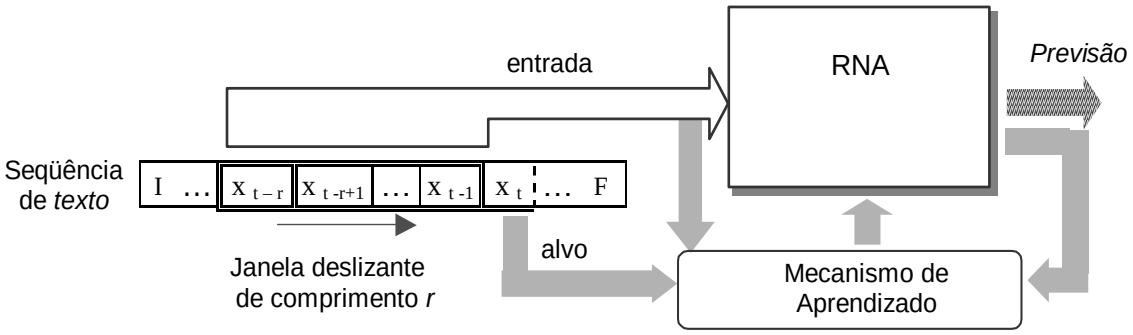



Figura 9: Entrada de uma seqüência de texto para compressão por meio de uma RNA

[15] Após um levantamento feito até 8/04/2000 nas principais bases de dados de textos completos (IEEE, ACM, Pergamon -Elsevier, *Science* Direct, CiteSeer) e na internet foram encontradas duas abordagens de aplicação de RNAs para compressão de texto.

Durante o aprendizado supervisionado, a RNA aprende a associar um contexto (entrada) ao caracter sucessor de fato (alvo) por meio de um mecanismo que ajusta seus parâmetros internos (pesos e/ou estrutura) para atualização do *modelo* em formação. (As setas cheias representam o fluxo de dados durante o estágio de aprendizado da rede). O tipo de operação realizada sobre a saída emitida pela rede, durante o processo de aprendizado, é determinado pelo algoritmo de aprendizado implementado. Durante a previsão, a RNA recebe apenas uma subseqüência de entrada para a qual deve estimar uma distribuição probabilística (seta listrada) para o caracter sucessor, utilizando o conhecimento já adquirido e representado no *modelo* formado durante o aprendizado. A distribuição probabilística emitida pela RNA é posteriormente convertida em uma seqüência de bits por um codificador estatístico para a compactação do caracter que efetivamente ocorre naquele ponto da seqüência em processamento.

A modelagem realizada pela RNA pode basear-se em um conjunto de treino para geração do modelo, ou no aprendizado do modelo a partir da própria seqüência processada, sem treinamento prévio. Na modelagem *estática* o modelo neural é gerado por meio de um treinamento prévio sem sofrer ajustes durante a compactação dos arquivos. Na compressão por meio de modelagem *semi-estática*, a RNA utiliza o próprio texto a ser compactado para aprender um modelo que deve, portanto, ser enviado juntamente com o código gerado a partir deste modelo para permitir a descompactação do texto. A modelagem *semi-adaptativa* utiliza uma RNA pré-treinada que deve ajustar-se aos textos novos em compressão. Por fim, na modelagem *adaptativa* o modelo é gerado dinamicamente pela RNA ao longo do próprio processo de compressão da seqüência.

### 3.4.1 Trabalhos Relacionados

O trabalho de Schmidhuber e Heil (1996) abordou, pioneiramente[15], a compressão de texto por meio de RNAs, tendo sido sucedido pela abordagem desenvolvida por Natsev (1997) e Long, Natsev e Vitter (1999). Nestes trabalhos, foram implementadas redes *feedforward* com múltiplas camadas de neurônios, com aprendizado supervisionado utilizando o algoritmo *Backpropagation* (BP), e um prévio treinamento *off-line*. Nestas abordagens, a RNA é utilizada para estimar a distribuição probabilística para cada caracter a ser compactado, condicionado por um contexto de entrada de tamanho fixo.

A base de dados utilizada pelas redes neurais em ambos os trabalhos é aqui descrita, conforme os relatos de (SCHMIDHUBER e HEIL, 1996; NATSEV, 1997; LONG, NATSEV, VITTER, 1999). Para estes, o conjunto de treinamento foi formado por 40 artigos de um

jornal de idioma alemão (*Münchner Merkur*). Para o conjunto de *teste 1* foram utilizados 20 artigos não vistos do mesmo jornal por meio do qual foram treinadas, sem sobreposição com o conjunto de treino. Para o conjunto de *teste 2* foram utilizados 10 artigos de um jornal diferente no mesmo idioma (*Frankenpost*). O tamanho de cada um dos arquivos (tanto do conjunto de treino quanto dos conjuntos de teste) variou entre 10 e 20 KB.

O desempenho destes métodos foi aferido com base na *razão de compressão média* ao longo dos arquivos de cada conjunto de testes. Utilizando a base de dados descrita, as razões de compressão médias para cada um dos conjuntos de teste, obtidas em ambas as abordagens, foram melhores que as atingidas pelo *software* gzip (que consiste em um método *on-line*), e que os compactadores pack e compress. Em ambas as abordagens (SCHMIDHUBER e HEIL, 1996) e (NATSEV, 1997; LONG, NATSEV e VITTER, 1999) foi relatado um longo tempo para treinamento e processamento da rede neural.

- ***Sequential Neural Prediction* - Schmidhuber e Heil (1996)**

Neste trabalho foi utilizada uma arquitetura *feedforward* com uma única camada escondida formada por 430 neurônios, uma camada de entrada com $|\Omega|\cdot r=400$ unidades (sendo $|\Omega|=80$ o número de caracteres do alfabeto $\Omega$ e $r=5$ o comprimento do contexto) e uma camada de saída formada pelos 80 caracteres possíveis de $\Omega$. Para a entrada foi utilizada a representação binária, em que apenas o componente correspondente à posição do caracter ativo presente no contexto tem seu valor diferente de zero. O treinamento prévio da rede foi realizado por meio do algoritmo *BP* com taxa de aprendizado fixa. Durante os testes não foram realizados ajustes de pesos na rede neural, ou seja, não houve adaptação aos arquivos em compactação. Esta abordagem caracteriza, portanto, uma modelagem estática para compressão de *texto*.

Nos experimentos realizados, a distribuição probabilística estimada pela rede neural foi utilizada pelo codificador de *Huffman* em uma das variantes de operação e pelo codificador aritmético em outra variante. A rede neural atingiu uma razão de compressão média (ao longo de cada conjunto de arquivos de testes) melhor que as obtidas pelos compactadores *gzip*, pack e compress (tanto em sua variante integrada ao codificador de *Huffman*, quanto na variante que utilizou o codificador aritmético).

- ***Prediction by Smooth Mapping* (*PSM*) -Natsev (1997); Long, Natsev e Vitter (1999)**

A abordagem anterior foi ampliada por Natsev (1997) e Long, Natsev e Vitter (1999) por meio da utilização de uma arquitetura *feedforward* com 2 camadas escondidas, 256 unidades da camada de saída e algumas especificidades de arranjo estrutural que possibilitam a *re-*

*representação off-line* do código ASCII com |Ω|=256 caracteres. A arquitetura da rede, denominada de *PSM* (*Prediction by Smooth Mapping*), foi concebida com uma camada de entrada formada por $|\Omega| \cdot r$ unidades para as quais foi também utilizada a representação binária. Para o treinamento foram utilizados: contexto $r$=5 (resultando em 1280 unidades de entrada) e 400 neurônios escondidos (ao todo). Finalizado o treinamento, foram fixados os pesos entre a camada de entrada e a primeira camada escondida, correspondentes à *re-representação* utilizada para pré-processar os dados antes de propagá-los ao resto da rede para a compactação de cada arquivo de teste. Nesta abordagem foi utilizado o algoritmo de treinamento *BP* com decremento da taxa de aprendizado e o codificador aritmético para codificação.

Após um prévio treinamento *off-line*, a rede foi atualizada durante a compressão de cada arquivo, caracterizando uma modelagem semi-adaptativa. Diante de ambos os conjuntos de teste, os resultados de compressão atingidos rela rede *PSM* foram melhores que os obtidos por meio do modelo de Schmidhuber e Heil (1996) e por meio de um PPMC3 (PPMC com contexto de tamanho máximo igual a três). Uma versão "treinada" do PPMC3 superou o desempenho de todos os outros métodos relatados em relação ao conjunto de *teste 1* e obteve razão de compressão média equivalente à atingida pela rede neural *PSM* para o conjunto de *teste 2*.

Neste trabalho foi acrescentado um terceiro experimento, em que a rede PSM foi treinada por meio de obras literárias de um autor de língua inglesa, com o arquivo totalizando pouco mais que 1 MB. Para este experimento, com conjunto de teste formado por meio de outras obras do mesmo autor, o desempenho de *PSM* com utilização de $r$=5, foi superior ao do *gzip* e ao do "PPMC3 treinado", mas inferior ao do PPMC3 padrão. A rede de Schmidhuber e Heil (1996) não foi avaliada neste experimento. Para superar a razão de compressão do PPMC3, a rede PSM utilizou: $r$=10 (portanto, 2560 unidades de entrada), 850 neurônios escondidos (ao todo) e 256 unidades de saída (NATSEV, 1997; LONG, NATSEV e VITTER, 1999).

#### 3.4.1.1 Discussão

Os trabalhos de Schmidhuber e Heil (1996), de Natsev (1997) e de Long, Natsev e Vitter (1999) obtiveram ótimos resultados de compressão para a base de textos utilizada. Redes *feedforward* multicamadas e treinadas por meio do algoritmo *BP* são eficazes para a realização de uma representação interna distribuída dos padrões apresentados, possuindo bom potencial de generalização em relação ao conjunto de treinamento representativo utilizado. A codificação distribuída nas camadas escondidas as torna mais resistentes a ruído, além de

favorecer o melhor aproveitamento da memória interna na rede, na medida em que é necessário um menor número de neurônios para representar a informação.

Ainda que o tempo de processamento seja um aspecto fundamental para a viabilidade prática de um algoritmo de compressão, a análise aqui feita restringe-se à quantidade de compressão obtida. Parte-se do pressuposto de que estes algoritmos ainda não estão otimizados e do fato de que a avaliação do tempo de processamento depende do ambiente de implementação (máquina e estrutura de dados) e de condições padronizadas para os testes.

Com relação ao modelo *PSM*, apresentado por Natsev (1997) e Long, Natsev e Vitter (1999), é preciso observar o fato de que um contexto maior (composto por 5 caracteres) tende a favorecer tanto a compressão por meio da rede neural descrita, quanto teria favorecido o método PPMC (uma vez que um PPMC5 já demonstrou atingir melhores taxas de compressão que o PPMC3 diante de bases de *benchmark*). Portanto, para uma comparação mais efetiva quanto ao desempenho de compressão poderiam ter sido relatados, também, experimentos utilizando contexto de igual tamanho para a rede neural *PSM* e para o PPMC.

A eficácia da *re-representação off-line* de *PSM* foi demonstrada para situações em que havia relativa similaridade entre a *fonte* de linguagem dos exemplares de treino e os utilizados nos testes. No terceiro experimento, a rede teve que refazer sua *re-representação* por meio do treinamento diante de textos de mesma autoria daqueles para os quais seria testada. Para este experimento foram alterados: o tamanho do contexto de entrada (que passou a $r$=10), a configuração de sua estrutura (aumentando o número de unidades escondidas) e a taxa de aprendizado. Em decorrência do processo de treinamento, a *re-representação* encontra uma nova representação para o código ASCII, a partir dos exemplos por meio dos quais a rede foi treinada. Porém, é possível que a *re-representação* obtida não seja eficaz para compressão de textos originários de *fontes* com estrutura estatística muito diferente daqueles para os quais a rede neural foi treinada, ainda que no mesmo idioma.

Em ambos os trabalhos, o bom desempenho de compressão é condicionado pelo treinamento prévio por meio de um conjunto de dados representativo do domínio considerado no teste. Os modelos de rede adotados demandam um novo *treinamento*, caso haja uma mudança na *fonte* de origem dos textos, sendo que a configuração estrutural para representação do conhecimento proveniente de uma *fonte* pode não ser a mais adequada para outra com distribuição probabilística diferente. Por outro lado, a compressão para uma classe específica de textos, por meio da qual a rede é previamente treinada, pode ser a opção mais indicada em determinadas situações, favorecendo estas abordagens.

## 3.5 Considerações Finais

Neste capítulo foram apresentadas características de RNAs e conceitos-chave da Teoria da Ressonância Adaptativa (*Adaptive Resonance Theory - ART*), juntamente com diversos modelos baseados em *ART* que implementam aprendizado supervisionado (ARTMAPs). Foram descritas, também, aplicações de redes MLP à compressão de texto, sendo que estas alcançaram boas taxas de compactação, realizando um treinamento prévio por meio de textos do mesmo domínio de *linguagem* do qual se originaram os exemplares que foram utilizados posteriormente para teste.

A aplicação de RNAs para compressão de texto com modelagem adaptativa precisa lidar com aspectos críticos como: a configuração da estrutura e conectividade da rede para um desempenho adequado, o processo de treinamento e o "re-treinamento" diante de mudanças no ambiente de dados. Algumas características de RNAs baseadas em ART, como a capacidade de aprendizado incremental *on-line* e *off-line* e a construção de novas categorias para alocar novos padrões de entrada podem mostrar-se úteis para lidar com as questões levantadas. Dentre as diversas redes com aprendizado supervisionado da classe ARTMAP, a rede fuzzy ARTMAP tem sido difundida por meio de inúmeras aplicações, muitas das quais citadas no Quadro 1. Ressalta-se que a partir de pesquisa bibliográfica nas bases de dados de textos completos: IEEE, ACM, Pergamon-Elsevier, *Science* Direct, CiteSeer e na internet, até abril de 2000 não foram encontrados trabalhos anteriores em que redes das classes ART e ARTMAP tivessem sido aplicadas à compressão de *texto*. Antecedendo a apresentação do modelo adaptado desenvolvido (Capítulo 5), apresenta-se, no próximo capítulo, a rede original fuzzy ARTMAP.

# REFERÊNCIAS

AGGARWAL, Raj K., XUAN, Q. Y.; JOHNS, Allan T.; LI, Furong; BENNETT, Allen. A Novel Approach to Fault Diagnosis in Multicircuit Transmission Lines Using Fuzzy ARTMAP Neural Networks. **IEEE Transactions on Neural Networks**, v. 10, n. 5, p.1214-1221, September 1999.

ANDERSON, James A. ; ROSENFELD, Edward, (eds). **Neurocomputing** - Foundation of Research. USA, Massachusetts Institute of Technology: MIT Press, 1989.

ARGAMON, Shlomo; KOPPEL, Moshe; AVNERI, Galit. Routing Documents according to Style. In: FIRST INTERNATIONAL WORKSHOP ON INNOVATIVE INTERNET INFORMATION SYSTEMS (IIIS), 1, June 1998, Pisa, Italy. **Proceedings ….** Disponível em: <http://www.idt.ntnu.no/~monica/iiis-98/proceedings_on_line.html> Acesso em: 9 de dezembro de 2000.

ARNOLD, Ross; BELL, Timothy C.. A Corpus for the Evaluation of Lossless Compression Algorithms. In: DATA COMPRESSION CONFERENCE (DCC), 7 March 1997, Snowbird, Utah, USA. **Proceedings …** James A. Storer, Martin Cohn (Eds.), IEEE Computer Society Press, 1997, p. 201-210.

BELL, Timothy C.. **The Canterbury Corpus,** 1998. Disponível em: <http://corpus.canterbury.ac.nz>Acesso em: 12 de março de 2000.

BELL, Timothy C.; CLEARY, John G.; WITTEN, Ian H.. **Text Compression**. Englewood Cliffs, New Jersey: Prentice Hall Advanced Reference Series, 1990.

BELL, Timothy C.; WITTEN, Ian H.; CLEARY, John G.. Modeling for Text Compression. **ACM Computing Surveys**, v.21, n.4, p. 557-591, December 1989.

BERKELEY TECHNOLOGY LAW JOURNAL. Disponível em: <http://www.btlj.boalt.org> Acesso em: 12 de setembro de 2000

BISHOP, Chistopher M.. **Neural Networks for Pattern Recognition**. New York: Oxford University Press Inc., reprinted 1999.

BRADSKI, Gary; GROSSBERG, Stephen. Fast-Learning VIEWNET Architectures for Recognizing Three-dimensional Objects from Multiple Two-dimensional Views. **Neural Networks**, v.8, n. 7/8, p.1053-1080, 1995.

BEHAVIORAL & BRAIN SCIENCES. Disponível em: <http://www.bbsonline.org> Acesso em: 12 de setembro de 2000.

CARPENTER, Gail A.. Distributed Learning, Recognition and Prediction by ART and ARTMAP Neural Networks. **Neural Networks**, v. 10, n. 8, p.1473-1494, 1997.

CARPENTER, Gail A.; GJAJA, Marin. N.; GOPAL, Sucharita; WOODCOCK, Curtis. E. ART neural networks for remote sensing: Vegetation classification from Landsat TM and terrain data. **IEEE Trans. Geosci. Remote Sens**. v. 35, p. 308-325, 1997.

CARPENTER, Gail A.; GROSSBERG, Stephen. A Massively Parallel Architecture for a Self-organizing Neural Pattern Recognition Machine. **Computer Vision, Graphics and Image Processing**, v. 37, p. 54-115, 1987a.

_____ ART2: Self-organization of Stable Category Recognition Codes for Analog Input Patterns. **Applied Optics**, v. 26, n. 23, p. 4919-4930, December 1987b.

_____ The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network. **IEEE Computer**, v. 21, n. 3, p.77-88, March 1988.

_____ ART3: Hierarchical Search using Chemical Transmitters in Self-Organizing Pattern Recognition Architectures. **Neural Networks**, v.3, n.23, p.129-152, 1990.

_____ Fuzzy ARTMAP: A synthesis of Neural Networks and Fuzzy Logic for Supervised Categorization and Nonstationary Prediction. In: YAGER, R.R. & ZADEH, L.A. (eds.). **Fuzzy Sets, Neural Networks and Soft Computing**. New York: Van Nostrand Reinhold, p.126-165, 1994.

CARPENTER, Gail A.; GROSSBERG, Stephen; MARKUZON, Natalya; REYNOLDS, John H.; ROSEN, David B.. Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. **IEEE Transactions on Neural Networks**, v. 3, n. 5, p. 698-713, September 1992.

CARPENTER, Gail A.; GROSSBERG, Stephen; REYNOLDS, John H.. ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network. **Neural Networks**, v. 4, n.5, p. 565-588, 1991.

CARPENTER, Gail A.; GROSSBERG, Stephen; REYNOLDS, John H.. A Fuzzy ARTMAP Nonparametric Probability Estimator for Nonstationary Pattern Recognition Problems. **IEEE Transactions on Neural Networks**, v. 6, n. 6, p. 1330-1336, November 1995.

CARPENTER, Gail A.; GROSSBERG, Stephen; ROSEN, David B.. ART2-A: An Adaptive Resonance Algorithm for Rapid Category Learning and Recognition. **Neural Networks**, v. 4, p. 493-504, 1991a.

_____ Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. **Neural Networks**, v. 4, p.759-771, 1991b.

CARPENTER, Gail A.; MARKUZON, Natalya. ARTMAP-IC and Medical Diagnosis: Instance counting and inconsistent cases. **Neural Networks**, v. 11, n. 2, p.323-336, 1998.

CARPENTER, Gail A.; ROSS, William D.. ART-EMAP: A Neural Network Architecture for Learning and Prediction by Evidence Accumulation. **IEEE Transactions on Neural Networks**, v. 6, n. 4, p. 805-818, July 1995.

CARPENTER, Gail A.; TAN, Ah-Hwee. Rule extraction, fuzzy ARTMAP, and medical databases. In: WORLD CONGRESS ON NEURAL NETWORKS (WCNN) 1993. **Proceedings …** p. 501-506. Technical Report CAS/CNS-TR-93-016, Boston MA: Boston University, 1993.

CLEARY, John G.; TEAHAN, William J.. Some experiments on the zero frequency problem. In: DATA COMPRESSION CONFERENCE (DCC) 1995, Snowbird, Utah. **Proceedings …** James A. Storer, Martin Cohn (Eds.), IEEE Computer Society Press, 1995.

CLEARY, John G.; WITTEN, Ian H.. Data compression using Adaptive Coding and Partial String Matching. **IEEE Transactions on Communications,** v. 32, n. 4, p. 396-402, April 1984.

COVER, Thomas M.; THOMAS, Joy A.. **Elements of Information Theory**. New York: John Wiley & Sons, Inc., 1991.

DAGHER, I.; GEORGIOPOULOS, M.; HEILEMAN, G.L.; BEBIS, G.. An ordering Algorithm for Pattern Presentation in Fuzzy ARTMAP that tends to Improve Generalization Performance. **IEEE Transactions on Neural Networks**, v.10., n. 4, p. 768 -778, July 1999.

DOWNS, Joseph; HARRISON, Robert. F.; KENNEDY, R. Lee; CROSS, Simon. S. . Application of the fuzzy ARTMAP neural network model to medical pattern classification tasks. **Artificial Intelligence in Medicine**. n. 8, p. 403-428, 1996.

FAUSETT, Laurene V.. **Fundamentals of Neural Networks:** Architectures, Algorithms, and Applications. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1994.

FRANK, Eibe; CHUI, Chang; WITTEN, Ian H.. Text categorization using compression models. In: DATA COMPRESSION CONFERENCE (DCC), 7, 2000, Snowbird, Utah, USA. **Proceedings …** , IEEE Computer Society Press, 2000.

GOPAL, Sucharita; WOODCOCK, Curtis E.; STRAHLER, Alan H.. Fuzzy ARTMAP classification of global land cover from the 1 degree AVHRR data set. **Remote Sens. Environ.**, 67, p. 230–243, 1999.

GROSSBERG, Stephen. Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors. **Biological Cybernetics**, 23, p. 121– 134, 1976a.

GROSSBERG, Stephen. Adaptive pattern classification and universal recoding: II. Feedback, Expectation, Olfaction, Illusions. **Biological Cybernetics,** 23, p. 187-202, 1976b.

_____ How does the brain build a cognitive code?. **Psychological Review**, 87, 151. 1980.

_____**The Adaptive Brain**. Vol. I e II. Stephen Grossberg (ed.), Amsterdam: Elsevier/North Holland, 1987.

_____ (Ed.). **Neural Networks and Natural Intelligence**. Cambridge, Massachusetts: MIT Press, 1988.

HAM, Fredric M.; HAN, Soowhan. Classification of Cardiac Arrhythmias using Fuzzy ARTMAP. **IEEE Transactions on Biomedical Engineering**, v.43, n.4, P. 425-430, April 1996.

HAYKIN, Simon. **Neural Networks:** A Comprehensive Foundation. 2. edition. USA, Upper Sadle river, N.J.: Prentice-Hall, Inc., 1999.

HERTZ, John; KROGH, Anders; PALMER, Richard. **Introduction to the Theory of Neural Computation**. Santa Fe Institute: Addison-Wesley, 1991.

IBM SYSTEMS JOURNAL. Disponível em: <http://www.research.ibm.com> Acesso em: 12 de setembro de 2000.

JAIN, Anil K.; MAO, Jianchang, K.; MOHIUDDIN, K. M.. **Artificial Neural Networks: A Tutorial.** IEEE Computer, v.29, n.3., p. 31- 44, 1996.

JOURNAL OF NEUROPHYSIOLOGY. Disponível em: <http://www.jn.org/> Acesso em: 12 de setembro de 2000.

KARLGREN, Jussi. **Stylistic Experiments for Information Retrieval**. 2000. Dissertation (PhD in Computational Linguistics) - Swedish Institute of Computer Sciences, Stockholm University, Sweden.

KESSLER, Brett; NUNBERG, Geoffrey; SCHÜTZE, Hinrich. Automatic Detection of Text Genre In: 35th ANNUAL MEETING OF ASSOCIATION FOR COMPUTATIONAL LINGUISTICS AND 8th CONFERENCE OF EUROPEAN CHAPTER OF ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 1997, Madrid, Spain. **Proceedings …** . Somerset, New Jersey: Philip R. Cohen and Wolfgang Wahlster (eds.), Association for Computational Linguistics, p. 32-38, 1997.

KLIR, George J.; YUAN, Bo. **Fuzzy Sets and Fuzzy Logic:** Theory and Applications. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1995.

KOSKO, Bart. **Neural Networks and Fuzzy Systems:** A Dynamical Systems Approach to Machine Intelligence. USA, Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1992.

LLOBET, Eduard; HINES, Evor L.; GARDNER, Julian W.; BARTLETT, Philip N.; MOTTRAM, Toby T.. Fuzzy ARTMAP based electronic nose data analysis. **Sensors and Actuators B: Chemical**, v. 61, Issues 1-3, 14, p. 183-190, December 1999.

LONG, Philip. M.; NATSEV, Apostol. I.; VITTER, Jeffrey S.. Text compression via alphabet re-representation. **Neural Networks**, v.12, p. 755-765, 1999.

MARRIOT, Shaun; HARRISON, Robert F.. A Modified Fuzzy ARTMAP architecture for the aproximation of noisy mapping. **Neural Networks**, v. 8, n.4, pp.619-641, 1995.

MOFFAT, Alistair; BELL, Timothy; WITTEN, Ian H.. **Lossless Compression for Text and Images**. October, 1995.

NATSEV, Apostol. **Text Compression via Alphabet Re-representation**. 1997. Master Thesis - Duke University, Durham, NC., USA. <http://www.cs.duke.edu/~natsev/thesis/thesis.html>. Acesso em 07 de abril de 2000.

NELSON, Mark. LZW Data Compression. **Dr. Dobb's Journal.** October, 1989. Disponível em <http://dogma.net/markn/index.html>. Acesso em 8 de janeiro de 2000.

PANDYA, Abhijit S.; MACY, Robert B.. **Pattern Recognition with Neural Networks in C++**. Florida Atlantic University, Boca Raton, Florida: CRC Press, 1995.

PAO, Yoh-Han. **Adaptive Pattern Recognition and Neural Networks**. USA: Addison-Wesley Publishing Company, Inc., 1989.

PROJECT GUTENBERG. <http://promo.net/pg/> e <http://www.gutenberg.org/> Acesso em 10 de setembro de 2000.

RAUBER, Andreas; MÜLLER-KÖGLER, Alexander. Integrating Automatic Genre Analysis into Digital Libraries. In: First ACM/IEEE JOINT CONFERENCE ON DIGITAL LIBRARIES (JCDL01), June 24-28, 2001, Roanoke, VA. **Proceedings ….** Roanoke, VA: Fox, E.A., and Borgman, C.L. (eds.), ACM, p.1-10, 2001.

RICKEN, Cristina E.. **Classificação Automática de Imagens Multiespectrais através de Múltiplas Redes Fuzzy Artmap**. Trabalho Final para disciplina de Redes Conexionistas - PPGEP. Florianópolis, 1997.

_____ **Aprendizado Rápido versus Aprendizado Lento no Campo de Mapa de Fuzzy ARTMAP**. Trabalho não submetido à publicação. Florianópolis.1999.

RIPLEY, Brian D.. **Pattern Recognition and Neural Networks.** Cambridge, UK: Cambridge University Press, reprinted 1996.

ROSS, Timothy J.. **Fuzzy Logic with Engineering Applications**. McGraw Hill, 1995.

RUBIN, Mark A.. Application of fuzzy ARTMAP and ART-EMAP to automatic target recognition using radar range profiles. **Neural Networks** - Special Issue on Automatic Target Recognition, v. 8 , p. 1109-1116, 1995.

RUIZ, Miguel E.; SRINIVASAN, Padmini. Hierarchical neural networks for text categorization. In: 22nd ACM INTERNATIONAL CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, SIGIR-99**,** 1999, Berkeley, US. **Proceedings ….** New York, US. Marti A. Hearst, Fredric Gey, and Richard Tong, (eds), ACM Press, p. 281-282, 1999.

SAYOOD, Khalid. **Introduction to Data Compression**. 2.ed. USA: Morgan Kaufmann, 2000.

SCIENCE DIRECT. Disponivel em: <http://www.sciencedirect.com>. Acesso em: 12 de setembro de 2000

SCHMIDHUBER, Jürgen; HEIL, Stefan. Sequential Neural Text Compression. **IEEE Transactions on Neural Networks**, v. 7, n.1, p. 142-146, January 1996.

SEBASTIANI, Fabrizio. A tutorial on automated text categorisation. Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence, Buenos Aires, AR, 1999. In Analia Amandi and Alejandro Zunino (eds.), **Proceedings**…, pp. 7-35, 1999.

SERRANO-GOTARREDONA, Teresa; LINARES-BARRANCO, Bernabe. An ART1 Microchip and Its Use in Multi-ART1 Systems. **IEEE Transactions on Neural Networks**, v.8, n.5, p. 1184-1194, September 1997.

SHANNON, Claude E. . A Mathematical theory of communication. **Bell Systems Technical Journal**, 27, p. 379-423, 623-656, 1948.

SKAPURA, David M.. **Building Neural Networks**. USA, NY - New York: ACM Press Books, 1996.

TAN, Ah-Hwee. Cascade ARTMAP: Integrating Neural Computation and Symbolic Knowledge Processing. **IEEE Transactions on Neural Networks**, v. 8, n. 2, p. 237-250, March 1997.

TAN, Ah-Hwee. Text Mining: The state of the art and the challenges. In: PACIFIC ASIA CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING (PAKDD'99), WORKSHOP ON KNOWLEDGE DISCOVERY FROM ADVANCED DATABASES, 1999. **Proceedings**… p. 65-70, 1999.

TEAHAN, William. J.. **Modelling English Text.** 1998. Thesis (Ph.D.) - University of Waikato, New Zealand, 1998.

TONTINI, Gerson. **Automatização da Identificação de Padrões em Gráficos de Controle Estatístico de Processos (CEP) Através de Redes Neurais com Lógica Difusa**. 1995. Tese (Doutorado em Eng. Mecânica) - Curso de Pós-Graduação em Engenharia Mecânica, UFSC, Florianópolis.

WAZLAWICK, Raul S.. **Um Modelo Operatório para Construção de Conhecimento**. 1993. Tese (Doutorado em Eng. de Produção) - Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis, 1993.

WELCH, Terry A. A Technique for High-Performance Data Compression. **IEEE Computer**, v. 17, n. 6, p. 8-19, June 1984.

WIENER, Erik; PEDERSEN, Jan O.; WEIGEND, Andreas S. . A neural network approach for topic spotting. In: FOURTH ANNUAL SYMPOSIUM ON DOCUMENT ANALYSIS AND INFORMATION RETRIEVAL (SDAIR), 1995. **Proceedings …**, pp. 317-332, 1995.

WILLIAMSON, J. R.. Gaussian ARTMAP - A neural network for fast incremental learning of noisy multidimensional maps. **Neural networks**, v. 9, n. 5, p. 881-897, 1996.

WITTEN, Ian H. ; BRAY, Zane; MAHOUI, Malika; TEAHAN, William J.. Text mining: A new frontier for lossless compression. In: DATA COMPRESSION CONFERENCE, 1999, Snowbird, Utah. **Proceedings …**, Los Alamitos, CA: IEEE Press, p. 198-207, 1999.

WITTEN, Ian H.; MOFFAT, Alistair; BELL, Timothy. **Managing Gygabytes:** Compressing and Indexing Documents and Images. 2. ed. San Francisco, California: Morgan Kauffman Publishers, Inc., 1999.

WITTEN, Ian H.; NEAL, Radford M.; CLEARY, John G. Arithmetic coding for data compression. **Communications of the ACM**, v. 30, n. 6, p. 520–540, June 1987.

WU, C. J. ; SUNG, A.H.; SOLIMAN, H. S. Image Data Compression using ART Networks. In: ARTIFICIAL NEURAL NETWORKS IN ENGINEERING (ANNIE 93), 1993. **Proceedings …** pp. 417-422.

ZADEH, Lofti A.. Fuzzy Sets. **Information and Control**, v.8, p. 338-353, 1965.

ZIV, Jacob; LEMPEL, Abraham. A Universal Algorithm for Sequential Data Compression, **IEEE Transactions on Information Theory**, v. 23, n. 3, p. 337-343, May 1977.

_____ Compression of individual sequences via variable-rate coding, **IEEE Transactions Information Theory**, IT-24, n.5, p. 530-536, September 1978.

# APÊNDICE A

## Base de dados para classificação automática por gênero de texto

Tabela de arquivos de treinamento para classificação por gênero de texto

| *Classe* | *KB* | *Parte / Inteiro* | *Título* |
|---|---|---|---|
| Bbs | 229 KB | Inteiro | The detection and generation of sequences as a key to cerebellar function. Experiments and theory |
| | | Inteiro | Developmental structure in brain evolution |
| Blj | 229 KB | Inteiro | Intellectual property and the digital economy: why the anti-circumvention regulations need to be revised |
| | | Inteiro | Clash of the titans: regulating the competition between established and emerging electronic payment systems |
| Ibm | 229 KB | Inteiro | Information in places |
| | | Inteiro | Optimizing array reference checking in java programs |
| Jno | 229 KB | Inteiro | Role of frontal eye fields in countermanding saccades: visual, movement, and fixation activity |
| | | Inteiro | Interpreting neuronal population activity by reconstruction: unified framework with application to hippocampal place cells |
| Lit | 229 KB | Parte | Robinson Crusoe |
| | | Parte | Three musketeers, the |
| Nns | 229 KB | Inteiro | Multi-agent reinforcement learning: weighting and partitioning |
| | | Inteiro | Generative character of perception: a neural architecture for sensorimotor anticipation |

Tabela de arquivos de teste para classificação por gênero de texto

| *Classe* | *KB* | *#* | *Parte / Inteiro* | *Título* |
|---|---|---|---|---|
| bbs | 10 KB | 1 | Parte | Peripheral and central hyperexcitability: differential signs and symptoms in persistent pain |
| | | 2 | Parte | A theory of visual stability across saccadic eye movements |
| | | 3 | Parte | Co-evolution of neocortex size, group size and language in humans |
| | 25 KB | 1 | Inteiro | Recoverin and Ca2+ in vertebrate phototransduction |
| | | 2 | Inteiro | Evolutionary internalized regularities |
| | | 3 | Parte | Dreaming and the brain: toward a cognitive neuroscience of conscious states |
| | 50 KB | 1 | Inteiro | Long-lasting potentiation of gabaergic inhibitory synaptic transmission in cerebellar purkinje cells: its properties and possible mechanisms |
| | | 2 | Inteiro | Central inhibitory dysfunctions: mechanisms and clinical implications |
| | | 3 | Parte | Second language acquisition: theoretical and experimental issues in contemporary research |
| | 100 KB | 1 | Inteiro | Resolving the contradictions of addiction |
| | | 2 | Inteiro | Theory of mind in nonhuman primates |
| | | 3 | Parte | Neurobiology of the structure of personality: dopamine, facilitation of incentive motivation, and extraversion |
| blj | 10 KB | 1 | Inteiro | Article 2b and mass market license contracts: a Japanese perspective |
| | | 2 | Inteiro | On self-enforcing contracts, the right to hack, and willfully ignorant agents |
| | | 3 | Parte | Progressing towards a uniform commercial code for electronic commerce or racing towards nonuniformity? |
| | 25 KB | 1 | Inteiro | The limits in open code: regulatory standards and the future of the net |
| | | 2 | Inteiro | Commentary: black holes of innovation in the software arts |
| | | 3 | Parte | Database protection at the crossroads: recent developments and their impact on science and technology |
| | 50 KB | 1 | Inteiro | Patents, products, and public health: an analysis of the cellpro march-in petition |
| | | 2 | Inteiro | Of governments and governance |
| | | 3 | Parte | The internet gambling fallacy craps out |
| | 100 KB | 1 | Inteiro | As many as six impossible patents before breakfast: property rights for business concepts and patent system reform |
| | | 2 | Inteiro | Safety in numbers: revisiting the risks to client confidences and attorney-client privilege posed by internet electronic mail |
| | | 3 | Parte | Controlling market power in telecommunications: antitrust vs. Sector-specific regulation |
| ibm | 10 KB | 1 | Inteiro | Technical note a proposal to simplify data flow diagrams |
| | | 2 | Inteiro | Asparagus soup |
| | | 3 | Parte | The evolution of java security |
| | 25 KB | 1 | Inteiro | Personal area networks: near-field intrabody communication |
| | | 2 | Inteiro | Enterprise solutions structure |
| | | 3 | Parte | Designing a generic payment service |
| | 50 KB | 1 | Inteiro | Family traits in business objects and their applications |
| | | 2 | Inteiro | Capitalizing on intellectual assets |
| | | 3 | Parte | Support for enterprise javabeans in component broker |
| | 100 KB | 1 | Inteiro | S/390 cluster technology: parallel sysplex |
| | | 2 | Inteiro | The software bookshelf |
| | | 3 | Parte | Adaptive algorithms for managing a distributed data processing workload |

| | | | | |
|---|---|---|---|---|
| jno | 10 KB | 1 | Inteiro | Example of 2:1 interlimb coordination during fictive rostral scratching in a spinal turtle |
| | | 2 | Inteiro | Noise-induced spiral waves in astrocyte syncytia show evidence of self-organized criticality |
| | | 3 | Parte | Identification and characterization of catecholaminergic neuron b65, which initiates and modifies patterned activity in the buccal ganglia of aplysia |
| | 25 KB | 1 | Inteiro | Protein kinase c is required for long-lasting synaptic enhancement by the neuropeptide drnflrfamide in crayfish |
| | | 2 | Inteiro | Theta-frequency resonance in hippocampal ca1 neurons in vitro demonstrated by sinusoidal current injection |
| | | 3 | Parte | Sharp, local synchrony among putative feed-forward inhibitory interneurons of rabbit somatosensory cortex |
| | 50 KB | 1 | Inteiro | Pet study of pointing with visual feedback of moving hands |
| | | 2 | Inteiro | Acute and chronic increases in excitability in rat hippocampal slices after perinatal hypoxia in vivo |
| | | 3 | Parte | Mechanisms underlying the synchronizing action of corticothalamic feedback through inhibition of thalamic relay cells |
| | 100 KB | 1 | Inteiro | Organization of reaching and grasping movements in the primate cerebellar nucleias revealed by focal muscimol inactivations |
| | | 2 | Inteiro | Chemical stimulation of the intracranial dura induces enhanced responses to facial stimulation in brain stem trigeminal neurons |
| | | 3 | Inteiro | Envelope coding in the lateral superior olive. Iii. Comparison with afferent pathways |
| lit | 10 KB | 1 | Parte | Around the world in 80 days |
| | | 2 | Parte | Don Quixote |
| | | 3 | Parte | War of the worlds |
| | 25 KB | 1 | Parte | Adventures of Sherlock Holmes, the |
| | | 2 | Parte | Wuthering heights |
| | | 3 | Parte | Frankenstein |
| | 50 KB | 1 | Parte | Ben-Hur: a tale of the Christ |
| | | 2 | Parte | Picture of Dorian Gray, the |
| | | 3 | Parte | Strange case of dr. Jekyll and mr. Hyde, the |
| | 100 KB | 1 | Parte | Gulliver's travels |
| | | 2 | Parte | Ivanhoe |
| | | 3 | Parte | Moby Dick |
| nns | 10 KB | 1 | Inteiro | Incorporation of long-range feedback in neural networks under stability conditions |
| | | 2 | Inteiro | A unified approach for neural network-like approximation of non-linear functionals |
| | | 3 | Parte | Improved bidirectional retrieval of sparse patterns stored by hebbian learning |
| | 25 KB | 1 | Inteiro | Ensemble learning via negative correlation |
| | | 2 | Inteiro | How stereovision interacts with optic flow perception: neural mechanisms |
| | | 3 | Parte | Organization of face and object recognition in modular neural network models |
| | 50 KB | 1 | Inteiro | Neuro-fuzzy feature evaluation with theoretical analysis |
| | | 2 | Inteiro | Multi-sensor integration for on-line tool wear estimation through radial basis function networks and fuzzy neural network |
| | | 3 | Parte | Faithful representations with topographic maps |
| | 100 KB | 1 | Inteiro | Neural networks for predicting conditional probability densities: improved training scheme combining EM and RVFL |
| | | 2 | Inteiro | Modeling parietal-premotor interactions in primate control of grasping |
| | | 3 | Inteiro | Contextually guided unsupervised learning using local multivariate binary processors |