TRABALHO DE GRADUAÇÃO

UTILIZAÇÃO DE APRENDIZADO POR REFORÇO EM DECODIFICADORES DE CANAL POR ALGORITMO DE VITERBI DE DECISÃO SOFT

Carlos Alexandre Griebler

Brasília, dezembro de 2014

UNIVERSIDADE DE BRASILIA

FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Departamento de Engenharia Elétrica

TRABALHO DE GRADUAÇÃO

UTILIZAÇÃO DE APRENDIZADO POR REFORÇO EM DECODIFICADORES DE CANAL POR ALGORITMO DE VITERBI DE DECISÃO SOFT

Carlos Alexandre Griebler

Relatório submetido como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação.

Banca Examinadora

Prof. Dr. João Paulo Leite, ENE/UnB
Orientador _____

Prof. Dr. Alexandre Ricardo Soares Romariz, ENE/UnB
Examinador _____

Prof. Dr. Paulo Henrique Portela de Carvalho, ENE/UnB
Examinador _____

Dedico este trabalho a minha família,
professores e colegas de curso.

Agradecimentos

Aos meus irmãos e, especialmente, à minha mãe, por todo apoio necessário durante a realização deste trabalho.

Ao meu orientador, professor João Paulo, pelo incentivo, motivação, inspiração e por trilhar os caminhos deste trabalho.

À Andressa, pelo companheirismo, apoio e assistência prestados ao longo de minha graduação, especialmente durante o desenvolvimento deste trabalho.

A todos meus colegas de curso e universidade, pela amizade e suporte.

À Anna Carolina, pela solução de todos os problemas administrativos e pela "coorientação".

RESUMO

Métodos de aprendizado de máquina são amplamente utilizados nas áreas de robótica, controle e automação para resolver problemas dinâmicos de tomada de decisão e otimização. Na área de comunicações, contudo, sua utilização é menos expressiva.

Este trabalho apresenta uma introdução às comunicações digitais e à teoria da informação, para o entendimento dos decodificadores de canal baseados no algoritmo de Viterbi, e a introdução ao aprendizado por reforço, um método de aprendizado de máquina.

O objetivo do trabalho é construir um decodificador de decisão *soft*, controlado dinamicamente por um agente de aprendizado baseado no algoritmo Q-Learning, capaz de alterar seus níveis de quantização para obter ganhos de desempenho.

Conforme mostram os resultados, com uma modelagem adequada do sistema, é possível fazer com que o agente aprenda a realizar essa tarefa, sugerindo a potencial aplicação de técnicas de aprendizado por reforço na área de comunicações digitais.

Palavras-chave: Viterbi, Q-Learning, aprendizado por reforço, comunicações digitais, codificação de canal, códigos convolucionais.

ABSTRACT

Machine learning methods are widely used in the area of robotics, control and automation to solve dynamic decision making and optimization problems. In communications, however, its use is less significant.

This work presents an introduction to digital communications and information theory, to understand the channel decoders based on the Viterbi algorithm, and the introduction to reinforcement learning, a machine learning method.

The objective is to build a soft decision decoder, dynamically controlled by a learning agent based on Q-learning algorithm, able to change its quantization levels for performance gains.

As the results show, with proper system modeling, it is possible to the agent learns how to accomplish this task, suggesting the potential application of reinforcement learning techniques in digital communications.

Keywords: Viterbi, Q-Learning, reinforcement learning, digital communications, channel coding, convolutional codes.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE SIGLAS E SÍMBOLOS

Siglas

AM	*Amplitude modulation* (modulação em amplitude)
AM-DSB/SC	*Amplitude modulation double-sideband with supressed carrier*
ARQ	*Automatic repeat request*
ASK	*Amplitude* shift keying (chaveamento por deslocamento de amplitude)
AWGN	*Additive white gaussian noise* (ruído aditivo branco gaussiano)
BER	*Bit error rate* (taxa de erro de *bit)*
bps	*Bits* por segundo
BPSK	*Binary phase shift keying* (chaveamento binário por deslocamento de fase)
BSC	*Binary symmetric channel* (canal binário simétrico)
CPF	Cadastro de pessoa física
DC	*Discret component* (componente discreta)
DI	Deslocamento individual de limiares
DL	Deslocamento de limiares
DT	Diferença temporal
FSK	*Frequency shift keying* (chaveamento por deslocamento de frequência)
IEEE	*Institute of Electrical and Electronics Engineers*
MC	Monte Carlo
ML	*Maximum likelihood* (máxima verossimilhança)
MLSE	*Maximum likelihood sequence estimator* (estimador de sequência de máxima verossimilhança)
PD	Programação dinâmica
PM	*Phase modulation* (modulação em fase)
PSK	*Phase sift keying* (chaveamento por deslocamento de fase)
QAM	*Quadrature amplitude modulation* (modulação
QPSK	*Quadrature phase shift keying* (chaveamento por deslocamento de fase em quadratura)
SL	Salto de limiares
SNR	*Signal to noise ratio* (relação sinal-ruído)
TCP	*Transmission control protocol*

Sobrescritos

max	Máximo
min	Mínimo

Símbolos

a	Fator de *rolloff*
A_p	Amplitude de ajuste de potência/energia
d_H	Distância de Hamming
d_{min}	Distância mínima de um código corretor de erro
E_b	Energia de *bit*
E_s	Energia de símbolo
k_B	Constante de Boltzmann
k_ω	Constante de ajuste de banda de sinais modulados por ângulo
N_0	Densidade espectral de potência unilateral de um canal AWGN
P_e	Probabilidade de erro
R_b	Taxa de *bits*
R_c	Taxa de código
R_s	Taxa de símbolos
R_t	Recompensa acumulada
r_t	Recompensa
t	Instante de tempo
α	Constante de aprendizado
γ	Fator de desconto
Δ	Variação entre duas grandezas similares
ε	Probabilidade de exploração
μ	Média de uma variável aleatória
σ	Desvio padrão de uma variável aleatória
σ^2	Variância de uma variável aleatória

1. INTRODUÇÃO

1.1. MOTIVAÇÃO

As comunicações desempenham um papel fundamental nas sociedades modernas, sendo responsáveis pela união de pessoas, transmissão de culturas e ideias. Sua importância é tamanha que a evolução das tecnologias de comunicação são responsáveis por marcar gerações, como a invenção do telégrafo e do telefone, dos telefones celulares e da Internet. Atualmente, vivemos a convergência da informação digital nas comunicações. As redes de computadores possuem uma expressão tão grande que, segundo pesquisa da Amdocs [1], em 2014, 98% do tráfego gerado por sistemas móveis (ex: telefonia celular) foi exclusivamente de dados.

Um sistema de comunicação é digital quando a informação trocada entre o transmissor e o receptor é de natureza digital, isto é, pode ser representada por um conjunto de símbolos finitos, como um alfabeto binário, dígitos de 0 a 9, etc. Uma das maiores vantagens da informação digital é sua capacidade de processamento digital, isto é, por meio da computação. Além disso, a alta padronização de sistemas digitais facilita a interoperabilidade de redes de comunicação. Vários órgãos mundiais e nacionais se destinam a esse fim.

Um dos principais responsáveis pelo desenvolvimento das comunicações digitais foi o surgimento da teoria da informação desenvolvida por Shannon [2]. Dentre suas ideias centrais estavam a compressão e a codificação dos dados.

Como um canal de comunicação é capaz de gerar erros na informação transmitida, este é um limitante natural de desempenho nas comunicações. A codificação de canal, área da teoria de Shannon que representa a aplicação de processamento de dados para desenvolver a detecção e correção de erros, juntamente com as modulações digitais, busca fazer com que uma transmissão digital atinja seu maior desempenho possível, respeitando a limitação imposta pelo canal. A prova disso é o mais fundamental teorema de Shannon aplicado nesta área, o Teorema de Codificação de Canal [2].

Para atingir o desempenho ótimo, garantido pelo teorema, vários esquemas de codificação de canal foram, e ainda são, propostos [3]. A principal classe de códigos corretores de erro é formada pelos códigos lineares, que se subdividem em códigos de bloco e códigos lineares [3]. Os códigos de bloco são importantes devido a sua estrutura simples e bem definida, porém, os maiores ganhos de desempenho são advindos das comunicações que utilizam códigos convolucionais [4].

Grandes padrões de tecnologia, como por exemplo o IEEE 812.11, popularmente

conhecido por Wi-Fi, utilizam códigos convolucionais devido a sua elevada capacidade de correção de erro aliada a altas taxas de transmissão [5]. A maior complicação de um código convolucional é o seu processo reverso, isto é, sua decodificação. Diferentemente dos códigos de bloco, os códigos convolucionais não podem ser decodificados por meio de relações matemáticas simples.

O mais conhecido meio de se decodificar este tipo de código foi proposto por Viterbi em 1967 [6] e consiste em um algoritmo capaz de estimar o código mais provável de ter sido enviado pelo canal [3]. Decodificadores de Viterbi, como são conhecidos os decodificadores de códigos convolucionais baseados no algoritmo de Viterbi, são capazes de decodificar tanto informações binárias quanto informações reais, assumindo a função de um demodulador.

Quando um decodificador opera com sinais binários, é preciso que um demodulador converta as informações recebidas em *bits* de acordo com a modulação utilizada, e diz-se que este decodificador possui decisão *hard*. A outra forma de operação consiste em, diretamente, utilizar os sinais recebidos para decodificar o sinal, dispensando a presença do demodulador, neste caso, diz-se que o decodificador opera com decisão *soft*, apresentando desempenho superior aos decodificadores de decisão *hard* devido à maior quantidade de informação contida no valor real com relação a um valor digital [7].

Naturalmente, os sinais recebidos são sinais reais, enquanto que as implementações dos decodificadores são feitas digitalmente. Há, portanto, a impossibilidade da aplicação direta do decodificador nestes casos. Para contornar o problema os valores reais são quantizados, assumindo valores discretos e finitos, tornando possível sua representação digital. A digitalização do sinal resulta na perda de informação, que reflete na perda de desempenho do codificador.

Um decodificador de oito níveis apresenta desempenho similar ao de um decodificador *soft* sem quantização [7]. Contudo, até mesmo a escolha dos níveis de quantização podem alterar o desempenho do decodificador. Em [4] é proposta uma forma de adaptar dinamicamente os níveis do quantizador de modo a obter o melhor desempenho possível utilizando técnicas de aprendizado por reforço.

O aprendizado por reforço é uma forma de aprendizado de máquina baseada em recompensas. Amplamente utilizado em diversas áreas como robótica [8, 9, 10], controle [11, 12] e automação [13, 14], é capaz de resolver problemas de tomadas de decisão, adaptando ações do agente ao ambiente. Contudo, os resultados de suas aplicações em telecomunicações não são tão expressivos, embora algumas pesquisas na área [15, 16], indiquem a relevância do tema.

1.2. OBJETIVOS

Este trabalho apresenta uma tentativa de unir o aprendizado de máquina às comunicações digitais, construindo um sistema de quantização capaz de se adaptar ao meio de maneira a aumentar desempenho de um decodificador, reduzindo sua taxa de erro de *bit.* Para isso, é proposto um decodificador de canal de códigos convolucionais por meio do algoritmo de Viterbi de decisão *soft* capaz de adaptar seus níveis de quantização face à relação sinal-ruído do canal.

O modelo proposto utiliza o algoritmo e Q-Learning, uma técnica de aprendizado por reforço, para controlar os limiares de quantização de um decodificador de Viterbi com decisão *soft*, criando um decodificador dinâmico, capaz de se adaptar ao canal e obter melhoras em seu desempenho a partir de suas experiências, isto é, decodificações anteriores.

1.3. ORGANIZAÇÃO DO TEXTO

No capítulo 1 foi apresentada uma introdução ao trabalhado, expondo suas principais motivações e objetivos.

No capítulo 2 é apresentada a base matemática das comunicações digitais, apresentando os conceitos e ideias necessários para o entendimento do projeto proposto.

No capítulo 3 é feita a introdução à teoria da codificação de canal, abordando temas da teoria da informação aplicados às comunicações digitais, dentre os quais se encontram os decodificadores de Viterbi.

O capítulo 4 aborda a base do aprendizado por reforço, apresentando seus principais elementos e algoritmos. Além disso, algumas aplicações são propostas para melhor entendimento do conteúdo.

No capítulo 5 é proposta uma solução para o problema da adaptação dinâmica do quantizador de um decodificador de Viterbi *soft* utilizando aprendizado por reforço. O modelo utilizado e seus resultados são apresentados neste capítulo.

O capítulo 6 conclui o trabalho, apresentando as principais conclusões acerca do tema e apresentando propostas de continuação do trabalho.

2. COMUNICAÇÕES DIGITAIS

2.1. INTRODUÇÃO

As comunicações digitais ganharam espaço na transmissão informações principalmente devido a facilidade de interoperação entre sistemas digitais e à capacidade de processamento de dados digitais [17].

Para compreender como processar informações digitais e como funcionam os sistemas digitais de comunicação é importante saber em que pontos as informações analógicas e digitais se diferenciam. Ainda, a simulação de sistemas digitais dependem fundamentalmente do conhecimento da base teórica de transmissão de sinais digitais.

Neste capítulo, é feita uma breve revisão do conteúdo de sinais analógicos, de maneira a permitir o desenvolvimento da ideia de sinais digitais. Com isso, são apresentadas algumas técnicas de modulação digital necessárias para o entendimento do problema central, juntamente com formas de recepção e demodulação dos sinais quando estes são submetidos à canais de comunicação com imperfeições, principalmente, no que se refere à presença de ruído.

2.2. INTRODUÇÃO À INFORMAÇÃO DIGITAL

Dizer que um sistema de comunicação é digital significa que a informação trafegada é digital, isto é, assume valores discretos dentro de um conjunto finito. A forma mais comum de representação de dados digitais é o alfabeto binário, com apenas dois símbolos, sendo que a representação desses símbolos varia. O código Morse, no qual as letras e números são representados com um tom curto ou um tom longo, é um exemplo de alfabeto binário. Na eletrônica digital, também é utilizado um alfabeto binário, sendo utilizados dois valores de tensão para representar os dados, definidos por algum padrão. De modo mais geral, utiliza-se para a representação binária os números 0 e 1, chamados de *bit.*

Por outro lado, a natureza das informações não são digitais, mas, sim, analógicas. Por exemplo, a temperatura de um determinado corpo ou os sons que ouvimos. Porém, conseguimos medir a temperatura de um corpo com um termômetro digital e reproduzir uma música no nosso computador, que também é digital. Tudo isso é possível devido ao princípio mais básico do mundo digital, a digitalização. Como as informações analógicas variam dentro de um universo infinito, ou seja, pode assumir qualquer valor real, certamente haverá perda de informação quando esta informação for convertida para digital.

O primeiro passo para a digitalização é a amostragem. Informações analógicas são contínuas tanto no tempo quanto na amplitude, enquanto informações digitais não são

representadas continuamente no tempo e nem na amplitude. A amostragem consiste em discretizar a informação no tempo. Tomando como exemplo uma onda senoidal de frequência 500 *Hz*, escrevendo uma função matemática para ela temos algo como

$$v(t) = \sin(1000\pi t). \tag{2.1}$$

Para qualquer instante de tempo t escolhido haverá um valor para v. Como o sinal é analógico, é possível escolher um valor $t + \Delta t$ e não importa o quão pequeno seja Δt, ainda assim haverá um valor real para v. Discretizar este sinal significa escolher apenas amostras em alguns instantes de tempo, por isso o nome amostragem. O novo sinal, agora dito discreto, não é mais uma função do tempo, mas uma função da amostra. Para isso, é necessário definir um intervalo de amostragem, que é o intervalo de tempo entre cada amostra do sinal. Nesse exemplo, tome o intervalo de amostragem como 200 μs. Cada amostra será igualmente espaçada de $T = 200\ \mu s$, ou seja, a primeira amostra será tomada em $t = 0$, a segunda em $t = 200\ \mu s$, a terceira em $t = 400\ \mu s$ e assim por diante. De forma geral, a i-ésima amostra será tomada em $t = (i - 1)\cdot T$ para $i = 1, 2, 3, \ldots$. Por definição, utilizamos a notação entre colchetes para representar um sinal discreto em função do número da amostra n, para $n = 0, 1, 2, \ldots$ é

$$v[n] = \sin(1000\pi nT), \quad n = 0, 1, 2, \ldots \tag{2.2}$$

A função acima define, portanto, um sinal discreto e é uma sequência e não mais uma função contínua. A Figura 2.1 mostra graficamente o exemplo.

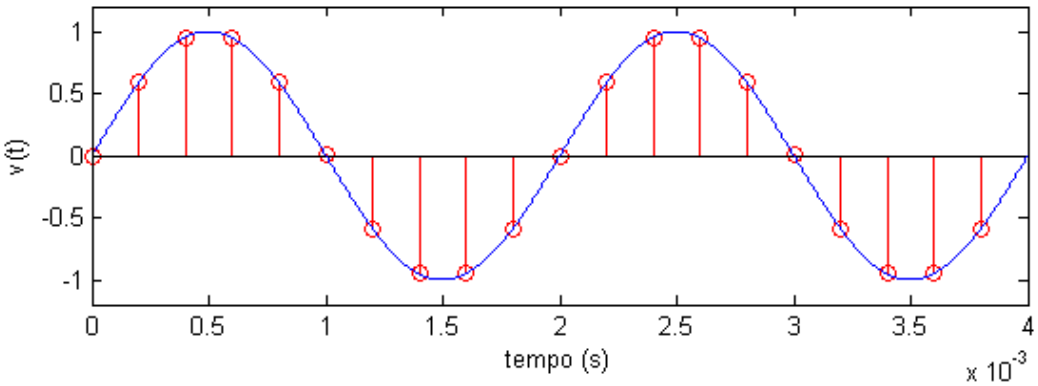


Figura 2.1. Função (2.1) analógica em azul e a sequência (2.2) em vermelho.

Concluída a amostragem do sinal, ainda temos informações contidas em um universo infinito, pois as amostras ainda podem assumir qualquer valor real. Como um sinal digital deve ser representado por um universo finito, o próximo passo para a digitalização é a quantização do sinal. Cada amostra deve ser representada por um símbolo, de modo que estes símbolos sejam limitados. Por exemplo, no caso binário, tem-se dois símbolos, 0 e 1. Utilizando um alfabeto binário para representar as amostras, é preciso definir o que seriam os zeros e o que seriam os uns. O caso mais simples é definir como 0 as amostras de valor

negativo e como 1 as amostras de valor não negativo. Nesse caso, tem-se o sinal dito quantizado mostrado na Figura 2.2.

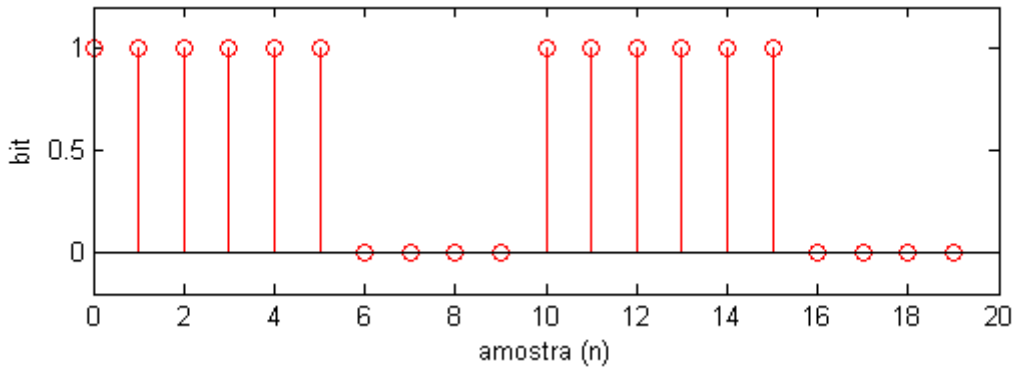


Figura 2.2. Sequência (2.2) quantizada em 2 níveis.

Este sinal é representado pela sequência binária 11111100001111110000 quantizado com um *bit* por amostra. Este tipo de digitalização é conhecido como PCM (*pulse code modulation*) [18] e é uma representação digital de um sinal analógico. É facilmente observável o erro introduzido ao sinal original. Esse erro pode ser minimizado ao ser utilizado um número maior de valores na quantização, consequentemente, utilizando mais símbolos por amostra. Para continuar utilizando um alfabeto binário, devido à compatibilidade com os equipamentos eletrônicos em geral, que utilizam a lógica binária, é possível representar um símbolo com mais de um *bit*. Com dois *bits*, temos os símbolos 00, 01, 10 e 11, totalizando quatro símbolos. Ou seja, com *b bits* é possível a formação de 2^b símbolos.

Tão importante quanto definir o número de níveis na quantização é definir quais valores serão quantizados em cada símbolo. No primeiro exemplo, valores negativos eram quantizados em 0 e valores não negativos em 1. Agora, utilizando dois *bits* para a quantização, é necessário definir quatro níveis de quantização e seus intervalos de quantização. Como o sinal se trata de uma onda senoidal que varia entre os valores –1 e +1, uma boa escolha para os níveis de quantização são os níveis igualmente espaçados –0,75, –0,25, +0,25 e +0,75. Os intervalos de quantização podem ser definidos como na Tabela 1.1 e a curva de quantização desse quantizador é mostrada na Figura 2.3.

Tabela 2.1. Valores e intervalos de quantização.

Valor amostrado	**Valor quantizado**	**Símbolo**
$v < -0,5$	–0,75	11
$-0,5 \leq v < 0$	–0,25	10
$0 \leq v < +0,5$	+0,25	00
$v \geq +0,5$	+0,75	01

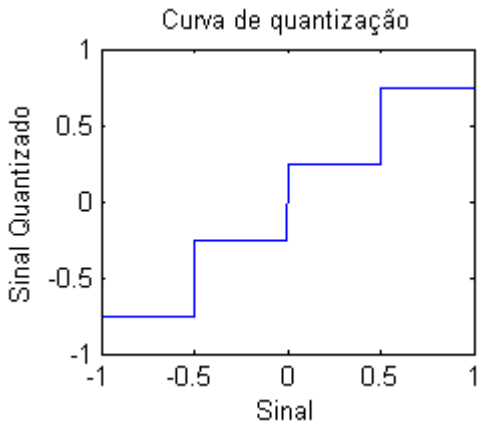


Figura 2.3. Curva do quantizador da Tabela 2.1.

Na Figura 2.4 já é possível perceber um sinal mais parecido com o sinal senoidal original. A mensagem binária neste caso seria 0001010101001111111100010101010011111111, duas vezes maior que a mensagem quantizada com apenas um *bit*. Sendo assim, quando o número de *bits* é aumentado, aumenta-se tanto a qualidade da informação quanto o tamanho da mensagem, ou seja, não é possível aumentar livremente a qualidade do sinal quantizado, havendo sempre a troca pela complexidade da mensagem.

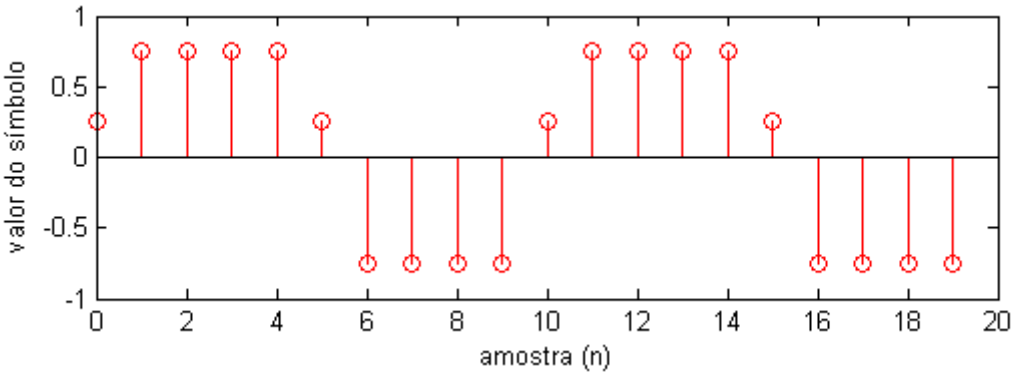


Figura 2.4. Sequência (2.2) quantizada em 4 níveis.

Existem ainda outras formas de digitalização. Por exemplo, é possível que a informação amostrada seja apenas a diferença entre o valor de duas amostras consecutivas. Entretanto, todas essas formas seguem o mesmo princípio de funcionamento.

2.3. TRANSMISSÃO DE SINAIS DIGITAIS

A transmissão de dados, tanto analógica quanto digital, se dá pela propagação eletromagnética dos sinais, por meio de um guia de onda ou não. Um guia é nada mais um caminho físico pelo qual o sinal será transmitido, como exemplo temos o cabo coaxial, o par trançado e a fibra óptica. Em uma transmissão não guiada, o sinal é propagado por um meio livre, como o ar, o vácuo ou a água. Transmissões não guiadas são também chamadas de sem fio.

Em uma transmissão de sinal analógico, o próprio sinal é enviado, como ocorre com a fala, em que sinal é gerado nas cordas vocais, propaga pelo meio e é recebido por outras pessoas do mesmo modo que foi gerado. Outra forma de transmissão analógica é por meio de uma portadora. Uma portadora é um tom senoidal que contém informações sobre a mensagem enviada em qualquer uma de suas variáveis, amplitude, frequência ou fase. A vantagem de se utilizar uma portadora, que possui uma frequência bem maior que as componentes de frequência do sinal que a modula, é justamente a possibilidade de transmissão de diversos sinais em um mesmo meio. Lembrando da teoria das comunicações, sinais isolados na frequência podem ser recuperados separadamente e, por isso, se utilizam portadoras, para carregar os sinais para bandas isoladas do espectro.

Como exemplo da diferença entre sinais transmitidos da maneira que são, chamados de sinais em banda base (ou banda básica), e sinais transmitidos por uma portadora, chamados de sinais em banda passante, pode ser analisado a fala entre um grupo de pessoas e um sistema de radiodifusão. Quando uma pessoa está conversando com outra, o receptor consegue facilmente entender a conversa. Porém, a medida que mais pessoas falam ao mesmo tempo, suas falas sofrem interferências. Mesmo que a voz não seja uma onda eletromagnética, o princípio é o mesmo: vários sinais foram transmitidos em uma mesma faixa de frequências. Já em um sistema de radiodifusão, vários sinais são transmitidos simultaneamente, cada um em uma faixa de frequência sem interferências, podendo o receptor escolher qual quer escutar ao sintonizar a frequência da portadora em seu rádio.

Como dito anteriormente, o sinal de informação modula uma portadora em alguma de suas variáveis. As modulações analógicas, portanto, são realizadas modificando a amplitude da portadora com a mensagem, chamada de *amplitude modulation* (AM), modificando a frequência da portadora, chamada de *frequency modulation* (FM), ou modificando a fase da portadora, chamada de *phase modulation* (PM). Cada uma dessas modulações ainda possui algumas variações, como por exemplo o *amplitude modulation double-sideband with supressed carrier* (AM-DSB/SC). A Figura 2.4 exemplifica alguns esquemas de modulação analógica.

Como pode ser observado, não há muitas diferenças entre o sinal modulado em frequência e em fase, especialmente para o caso do sinal de informação ter a forma senoidal. Isto se deve ao fato da frequência ser a variação instantânea da fase. Para um sinal de informação senoidal, sua variação instantânea também é um sinal senoidal, porém deslocada no tempo.

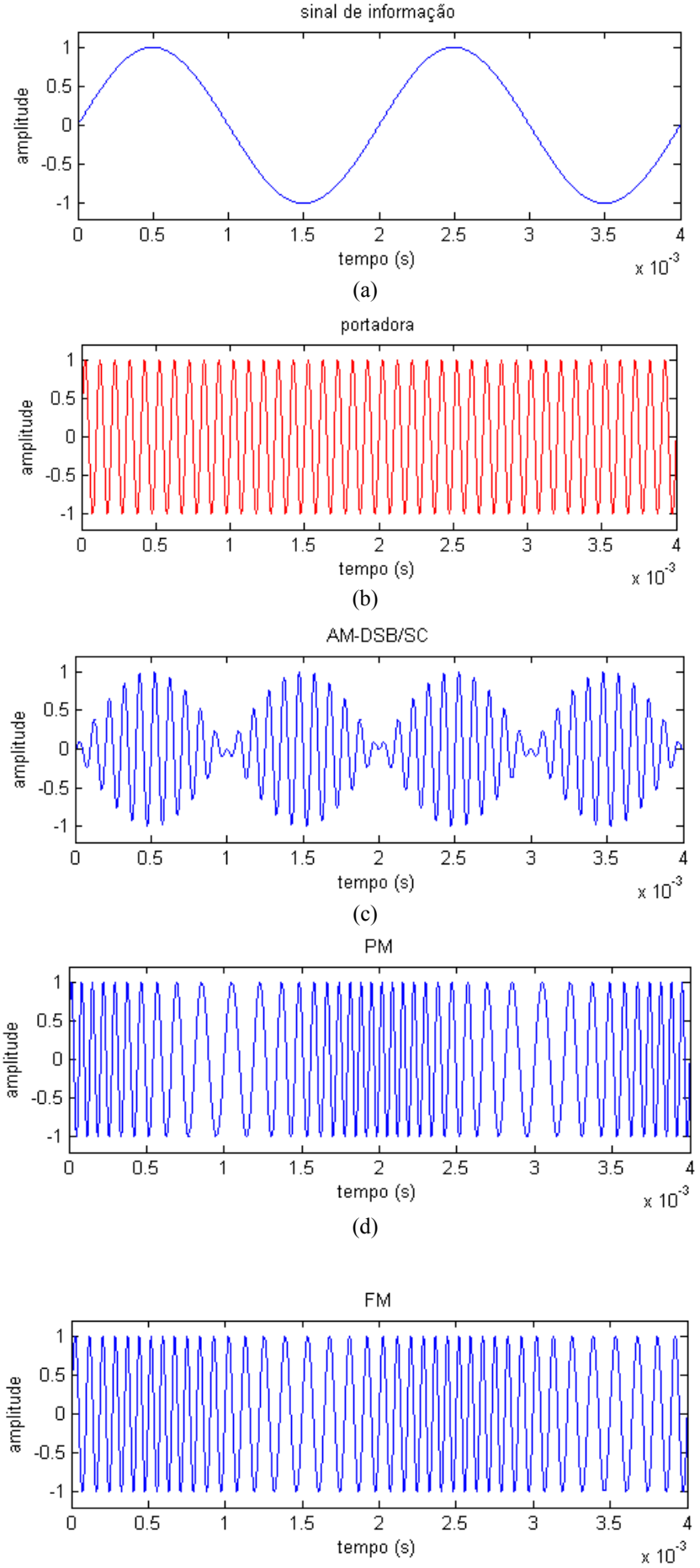


Figura 2.5. (a) Sinal de informação, (b) portadora e modulações analógicas: (c) AM-DSB/SC, (d) PM e (e) FM.

Sendo $m(t)$ o sinal de informação a ser modulado e a portadora um tom cossenoidal de frequência ω_c, as modulações analógicas pode ser obtidas pelas equações [18]

$$s(t)_{AM-DSB/SC} = A_p m(t)\cos(\omega_c t)$$
$$s(t)_{PM} = A_p \cos(\omega_c t + k_\omega^{PM} m(t))$$
$$s(t)_{FM} = A_p \cos(\omega_c t + k_\omega^{FM} \int_{-\infty}^{t} m(a)da),$$

sendo A_p uma constante para ajuste da potência do sinal e k_ω^{PM} e k_ω^{FM} constantes para o ajuste das bandas dos sinais PM e FM, respectivamente.

Portanto, em um sistema analógico em banda passante, tem-se alguns dos sinais das Figuras 2.5(c), 2.5(d) ou 2.5(e) transmitidos, enquanto em um sistema em banda básica, envia-se apenas o sinal de informação da Figura 2.5(a).

Para os sinais digitais, todos estes conceitos são válidos. A diferença está no fato de que o sinal transmitido deve ser analógico, devido a natureza das ondas eletromagnéticas ser analógica. Deste fato surge a necessidade de transformar a informação digital, que nada mais é do que uma sequência de símbolos, em um sinal.

No caso da transmissão digital em banda passante, a tradução da sequência em um sinal digital também se dá por meio de uma portadora. Do mesmo modo como na transmissão analógica, os mesmos parâmetros podem ser variados, porém, diferentemente deste, as variações ocorrem de maneira abrupta e não mais suavemente. Além disso, as transições devem ocorrer em intervalos de tempo determinados. O tempo de *bit* ou duração do *bit* é o intervalo de tempo em que o sinal carrega a informação de cada *bit*. O mesmo serve para a transmissão em banda passante, isto é, sequências de símbolos transformadas em sinais que variam no tempo. O tempo de *bit* está ligado com a taxa de transmissão, indicada pela unidade de *bits* por segundo (bps). Por exemplo, se um *bit* tem duração de meio segundo, a taxa de transmissão desse sistema é de 2 bps, pois a cada um segundo dois *bits* são enviados. Seguindo a mesma nomenclatura do caso analógico, modulação digital é a representação da informação digital por meio de ondas analógicas [19].

2.3.1. Modulação Digital

Na modulação digital, cada *bit* (ou símbolo) é mapeado em uma forma de onda de duração determinada pelo tempo de símbolo. Para o caso mais simples, essa forma de onda pode ter a forma de um pulso retangular, com amplitude -1 para o *bit* 1 e amplitude +1 para o *bit* 0. A Figura 2.6 mostra como ficaria a sequência de *bits* 0110001011 modulada por este pulso.

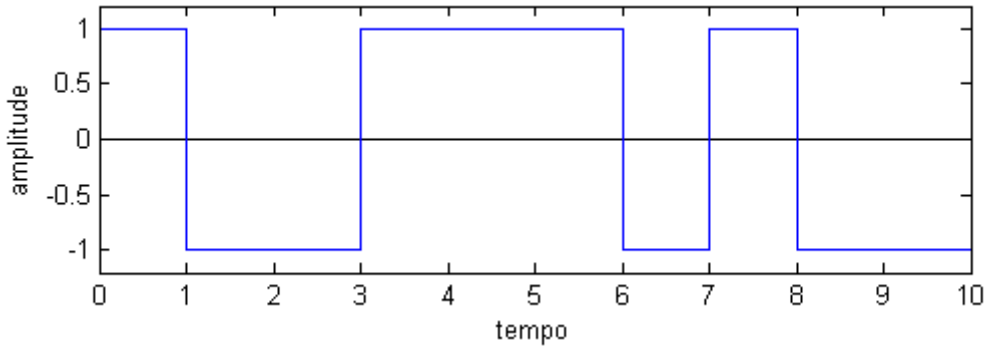


Figura 2.6. Forma de onda de um sinal digital modulado por um pulso retangular.

Modulações deste tipo são modulações em banda básica, pois a potência está concentrada nas componentes de baixa frequência. Basicamente, a densidade espectral de potência de um sinal digital modulado em banda básica depende da forma de pulso utilizado. Um pulso retangular, tem como densidade espectral de potência algo com a forma de uma função $\text{sinc}^2(f)$, em que $\text{sinc}(x) = \sin(x)/x$, com a largura do lóbulo principal determinada pelo inverso da duração do pulso. Já um pulso de Manchester, mostrado na Figura 2.7, possui como densidade espectral de potência algo proporcional a $\sin(f)\cdot\text{sinc}^2(f/2)$. Como sin(0) é zero, a componente contínua (componente DC) do sinal é eliminada, porém a largura de banda fundamental é dobrada. Ou seja, a depender do canal, cada formatação de pulso tem suas vantagens e desvantagens.

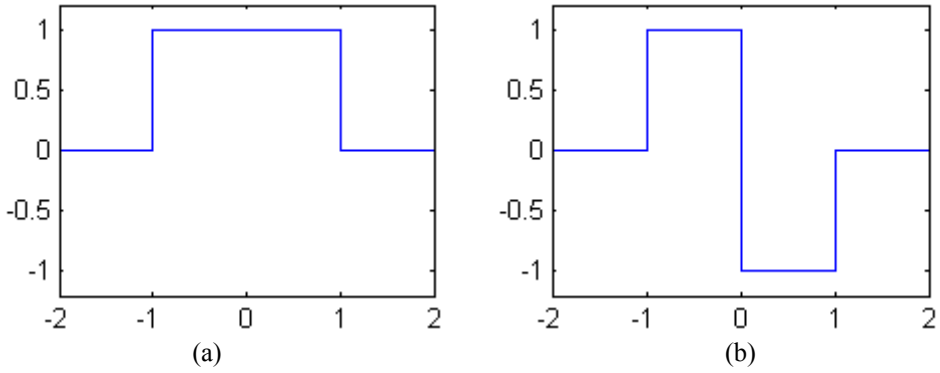


Figura 2.7. (a) Pulso retangular e (b) pulso de Manchester.

É importante notar que até aqui, apenas *bits* estão sendo modulados, sendo necessárias apenas duas formas de onda, uma para cada *bit*. Da mesma maneira como no processo de quantização, é possível agrupar *bits* para formar símbolos e aumentar a quantidade de formas de onda. Para isso, é conveniente definir um espaço de sinais, isto é, um conjunto de funções que serão utilizadas para modular o sinal digital.

Seja $\Phi_N = \{\varphi_1(t), \varphi_2(t), \varphi_3(t), \ldots, \varphi_N(t)\}$ um espaço de sinais, sendo que $\varphi_i(t)$ representa uma função de t e é um sinal que compõe o espaço de sinais Φ_N. O pulso $p(t)$ transmitido para cada símbolo será um mapeamento dos símbolos da mensagem $x[n]$ em uma combinação

linear dos sinais de Φ_N, dado por

$$p_k(t) = \sum_{i=1}^{N} x_{k,i} \cdot \varphi_i(t), \tag{2.3}$$

em que $x_{k,i}$ corresponde ao mapeamento do k-ésimo símbolo de $x[n]$ na função $\varphi_i(t)$ do espaço Φ_N. Por exemplo, $x[n]$ = 01000010111001 e a transmissão ocorre com quatro símbolos. O espaço de sinais utilizado é o da Figura 1.6, isto é, um pulso retangular e um pulso de Manchester, ambos com duração T. Primeiramente, é preciso fazer o mapeamento dos símbolos nas funções do espaço. A princípio, essa escolha será feita de maneira arbitrária, como na Tabela 2.2

Tabela 2.2. Mapeamento dos símbolos nas funções de base.

Símbolo	**Mapeamento $\{x_1, x_2\}$**
00	{+1, +1}
01	{+1, –1}
10	{–1, +1}
11	{–1, –1}

Separando os *bits* de $x[n]$ em blocos de dois (assim tem-se quatro símbolos possíveis), o sinal é transmitido segundo o mapeamento. O sinal resultante é obtido pela Equação (2.4) e é mostrado na Figura 2.8. Sabendo que cada sinal tem a duração de T segundos, é dito que a taxa de símbolo do sistema é de $R_s = 1/T$ *baud*, em que *baud* é a unidade símbolos por segundo. Além disso, como cada símbolo contém $b = 2$ *bits,* a taxa de *bits* do sistema é $R_b = R_s \cdot b = 2R_s$ bps.

$$s(t) = \sum_{k} p_k(t - kT) \tag{2.4}$$

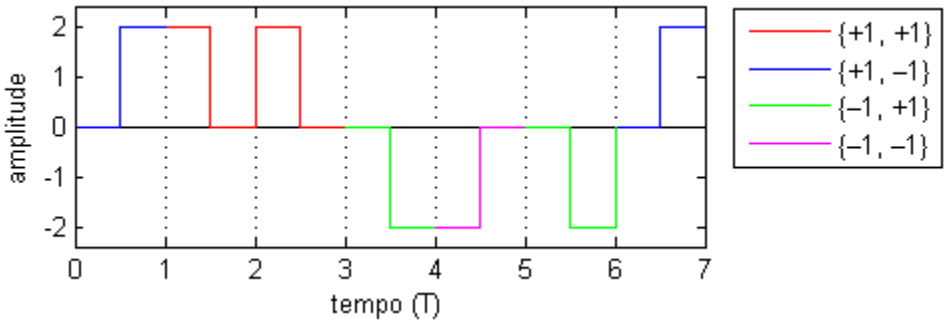


Figura 2.8. Sinal transmitido com o mapeamento sugerido.

Embora seja fácil fazer o processo inverso, isto é, a demodulação do sinal, pela forma de onda observada, não é este o objetivo da modulação digital, pois deseja-se recuperar a

sequência transmitida por meio de operações eletrônicas. Para tanto é necessário que a base de sinais utilizada seja uma base ortonormal. Uma base de funções é dita ortonormal quando satisfaz a condição

$$\langle \varphi_i, \varphi_j \rangle = \begin{cases} 0, & se\ \varphi_i \neq \varphi_j \\ 1, & se\ \varphi_i = \varphi_j \end{cases}, \tag{2.5}$$

em que

$$\langle \varphi_i, \varphi_j \rangle = \int_{-\infty}^{\infty} \varphi_i(t)\varphi_j^*(t)\,dt. \tag{2.6}$$

No exemplo anterior, facilmente verifica-se que as funções escolhidas não formam uma base ortonormal simplesmente por não satisfazerem a condição de possuírem energia unitária. Por definição, a energia de um sinal é obtida pela função norma da Equação (2.6) de um sinal consigo mesmo [20], isto é,

$$E_\varphi = \int_{-\infty}^{\infty} \varphi(t)\varphi^*(t)\,dt = \int_{-\infty}^{\infty} |\varphi(t)|^2\,dt. \tag{2.7}$$

Portanto, para que uma base de funções ortogonais seja ortonormais, basta normalizar cada função pela raiz quadrada de sua energia [20]. No exemplo dos sinais da Figura 2.7, sendo T suas durações, facilmente calcula-se que $E_\varphi = T$, portanto, sendo $\varphi_1(t)$ e $\varphi_2(t)$ os pulsos retangular e de Manchester, respectivamente, a base ortonormal $\{\phi_1(t), \phi_2(t)\}$ é

$$\begin{aligned} \phi_1(t) &= \frac{1}{\sqrt{T}}\varphi_1(t), \\ \phi_2(t) &= \frac{1}{\sqrt{T}}\varphi_2(t). \end{aligned} \tag{2.8}$$

Assim como na álgebra linear, bases ortonormais são de grande importância para a decomposição dos sinais em termos dessas funções, o que tornará possível o processo inverso à modulação. A definição de um espaço ortonormal de sinais é a forma mais genérica de modulação digital. Os casos anteriores representam modulações em banda básica. Contudo, as mesmas definições serão utilizadas para derivar as modulações em banda passante.

Seguindo os mesmos conceitos do caso analógico, as modulações em banda passante consistem na modulação de uma portadora, isto é, um sinal senoidal que tem suas variáveis modificadas pelo sinal de informação. A nomenclatura das modulações digitais segue a mesma linha das modulações analógicas, porém, como agora as variações ocorrem bruscamente devido ao fato da informação digital ser discreta, a palavra modulação é substituída por chaveamento. Em inglês, as modulações são:

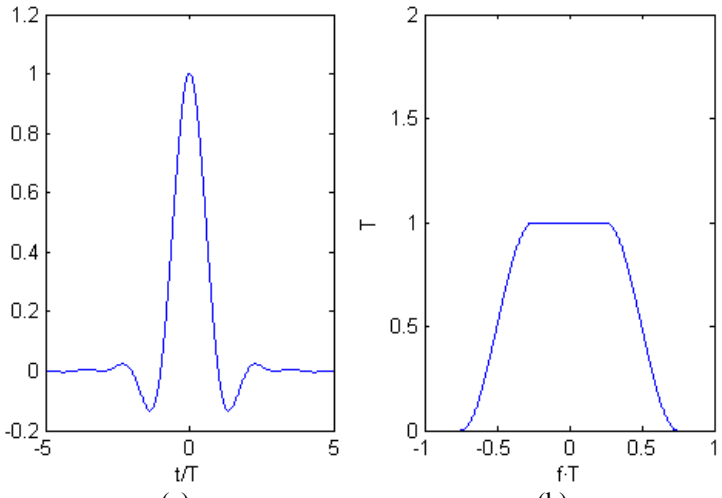
- ASK (*amplitude shift keying*): a amplitude de uma portadora senoidal é variada de acordo com a informação digital;

- PSK (*phase shift keying*): a fase da portadora é modificada de acordo com a informação digital;
- FSK (*frequency shift keying*): a frequência da portadora é modificada de acordo com a informação digital.

Para a modulação em amplitude (ASK), é preciso apenas um sinal no espaço, lembrando que a amplitude do sinal transmitido varia com os sinais de mapeamento. Portanto, apenas um sinal $\phi(t) = A_p \cdot \cos(\omega_c t)$ é suficiente para esta técnica de modulação. O valor de A_p deve ser calculado de maneira a manter a energia unitária do sinal. Como o sinal é finito, seu espectro possui réplicas de tal modo que devem atender ao critério de Nyquist para ser demodulado posteriormente. No caso mais simples, a onda senoidal é tornada finita por uma função rect(t/T), porém, existem outros pulsos mais indicados para transmitir sinais finitos. Estes pulsos atendem ao critério de Nyquist [19]. Portanto, muitas vezes, o sinal de fato transmitido é dado por $\phi(t) = p(t) \cdot A_p \cdot \cos(\omega_c t)$, sendo $p(t)$ a forma do pulso utilizado. Um dos mais utilizados é o pulso de cosseno levantado, dado por

$$p(t) = \operatorname{sinc}\left(\frac{t}{T}\right)\frac{\cos\left(\frac{\pi a t}{T}\right)}{1-\left(\frac{2at}{T}\right)^2}. \tag{2.9}$$

Embora o pulso $p(t)$ seja infinito no tempo, este pulso rapidamente converge para zero, fazendo com que mesmo sendo truncado no tempo ainda satisfaça o critério de Nyquist [20], não gerando interferência intersimbólica. A constante a, valor entre 0 e 1, é chamada de fator de *rolloff* e é o quanto a banda do sinal excede a banda necessária para alcançar a taxa de transmissão desejada. A Figura 2.9 mostra como o pulso de cosseno levantado se comporta no tempo e na frequência.



(a) (b)

Figura 2.9. Pulso cosseno levantado representado (a) no tempo e (b) na frequência para fator de *rolloff* a = 0,5.

Sendo assim, pode-se calcular a energia do sinal $\phi(t)$ por meio de

$$E_\phi = \int_{-\frac{T}{2}}^{\frac{T}{2}} |A_p\, p(t)\cos(\omega_c t)|^2\, dt = \int_{-\frac{T}{2}}^{\frac{T}{2}} A_p^2 |p(t)|^2 \cos^2(\omega_c t)\, dt = \frac{A_p^2}{2}\int_{-\frac{T}{2}}^{\frac{T}{2}} |p(t)|^2 [1+\cos(2\omega_c t)]\, dt$$

$$= \frac{A_p^2}{2}\left[\int_{-\frac{T}{2}}^{\frac{T}{2}} |p(t)|^2\, dt + \int_{-\frac{T}{2}}^{\frac{T}{2}} |p(t)|^2 \cos(2\omega_c t)\, dt\right] \approx \frac{A_p^2 E_p}{2}$$

Pelo lema de Riemann-Lebesgue [21], esta aproximação é válida desde que o período da onda cossenoidal seja muito menor que a duração do símbolo, *T*. Para obter a energia unitária do sinal,

$$E_\phi = 1 \quad \Leftrightarrow \quad \frac{A_p^2 E_p}{2} = 1 \quad \Rightarrow \quad A_p = \sqrt{\frac{2}{E_p}}. \tag{2.10}$$

Com a base de sinais obtida, o próximo passo é fazer um bom mapeamento dos símbolos. Como a energia do sinal enviado depende apenas da amplitude da onda senoidal, a energia média do sinal vai depender da escolha do mapeamento dos símbolos. Em um sistema binário, em que apenas os símbolos 1 e 0 são transmitidos, é possível fazer o mesmo mapeamento dos exemplos anteriores: 0 → +1 e 1 → –1. Os dois sinais enviados para cada símbolo são

$$s_0(t) = \sqrt{\frac{2}{E_p}}\, p(t)\cos(\omega_c t) \quad e \quad s_1(t) = -\sqrt{\frac{2}{E_p}}\, p(t)\cos(\omega_c t).$$

A energia média é calculada por

$$E_s = \sum_x P(x) E_x, \tag{2.11}$$

em que $P(x)$ é a probabilidade do símbolo x ser transmitido e E_x é a energia do pulso utilizado para transmitir o símbolo x. A energia do pulso depende apenas do mapeamento do símbolo, pois, sendo x_i o mapeamento do símbolo x no sinal $\phi_i(t)$,

$$E_x = \int_{-\infty}^{\infty} \left|\sum_i x_i \phi_i(t)\right|^2 dt\,. \tag{2.12}$$

Expandindo o somatório e aplicando o quadrado, todos os termos $\phi_i(t)\phi_j(t)$, para $i \neq j$, serão nulos, restando apenas

$$E_x = \sum_i x_i^2 \int_{-\infty}^{\infty} |\phi_i(t)|^2 dt = \sum_i x_i^2. \tag{2.13}$$

Com esse resultado, é possível construir uma representação gráfica para a modulação ASK. Uma linha horizontal representa o sinal da base ortonormal. Cada ponto nesta linha é o

mapeamento de um símbolo. No exemplo binário acima, existiriam dois pontos, em +1 e –1. Essa representação é chamada de constelação e a Figura 2.10 mostra a constelação da modulação binária ASK para um mapeamento geral de amplitude A. Além disso, supondo os símbolos equiprováveis, é possível calcular a energia média do sinal, sendo que a energia de cada símbolo é igual a distância ao quadrado de um símbolo ao zero na constelação.

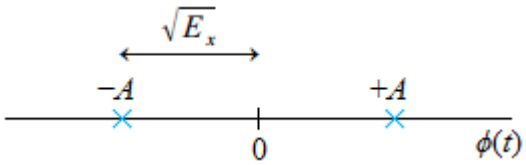


Figura 2.10. Constelação de um sinal ASK binário.

Neste caso, $P(x) = 1/2$ e $E_x = A^2$, fazendo com que $E_s = A^2/2 + A^2/2 = A^2$. Como um símbolo possui apenas um *bit* nessa modulação, $E_b = E_s$. Aumentando o número de *bits* por símbolo, mantendo a distância entre os símbolos constantes, obtém-se a vários esquemas de modulação, conhecidos como M-ASK, em que M representa o número de símbolos possíveis na modulação. A Figura 2.11 mostra o caso de uma modulação 8-ASK com os símbolos vizinhos equidistantes na modulação. Supondo símbolos equiprováveis, a probabilidade de ocorrência de qualquer símbolo é 1/8. Utilizando as distâncias para o cálculo da energia, facilmente verifica-se que a energia média do sinal 8-ASK mostrada é

$$E_{8-ASK} = \frac{2}{8}(7\mathrm{A})^2 + \frac{2}{8}(5\mathrm{A})^2 + \frac{2}{8}(3\mathrm{A})^2 + \frac{2}{8}A^2 = 21\mathrm{A}^2.$$

De maneira geral, a energia de um sinal M-ASK com símbolos vizinhos equidistante e símbolos equiprováveis é dada pela Equação (2.14), como é demonstrado em [20].

$$E_{M-ASK} = \frac{A^2}{3}(M^2 - 1) \qquad (1.14)$$

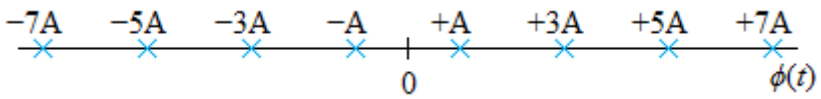


Figura 2.11. Constelação de uma modulação 8-ASK.

Para as modulações em fase, PSK, a característica variante do sinal é a fase. Como o mapeamento é capaz de alterar somente a amplitude, em um primeiro momento, há que se pensar que são necessárias várias funções de base. Por exemplo, para uma modulação com quatro símbolos, a base ortonormal de funções $\Phi_4 = \{\phi_1(t), \phi_2(t), \phi_3(t), \phi_4(t)\}$, pode ser definida como

$$\phi_1(t) = \sqrt{\frac{2}{E_p}}\, p(t)\cos(\omega_c t) \qquad \phi_2(t) = \sqrt{\frac{2}{E_p}}\, p(t)\cos\left(\omega_c t + \frac{\pi}{2}\right)$$
$$\phi_3(t) = \sqrt{\frac{2}{E_p}}\, p(t)\cos(\omega_c t + \pi) \quad \phi_4(t) = \sqrt{\frac{2}{E_p}}\, p(t)\cos\left(\omega_c t + \frac{3\pi}{2}\right)$$

$$x_1 = (1,0,0,0) \quad x_2 = (0,1,0,0) \quad x_3 = (0,0,1,0) \quad x_4 = (0,0,0,1)$$

Deste modo, cada símbolo seria associado a um sinal, sem a necessidade de um mapeamento. Contudo, ao se aplicar a relação trigonométrica

$$\cos(a+b) = \cos(a)\cos(b) - \operatorname{sen}(a)\operatorname{sen}(b), \tag{2.15}$$

tem-se que

$$\cos\left(\omega_c t + \frac{\pi}{2}\right) = \cos(\omega_c t)\cos\left(\frac{\pi}{2}\right) - \operatorname{sen}(\omega_c t)\operatorname{sen}\left(\frac{\pi}{2}\right) = -\operatorname{sen}(\omega_c t)$$
$$\cos(\omega_c t + \pi) = \cos(\omega_c t)\cos(\pi) - \operatorname{sen}(\omega_c t)\operatorname{sen}(\pi) = -\cos(\omega_c t)$$
$$\cos\left(\omega_c t + \frac{3\pi}{2}\right) = \cos(\omega_c t)\cos\left(\frac{3\pi}{2}\right) - \operatorname{sen}(\omega_c t)\operatorname{sen}\left(\frac{3\pi}{2}\right) = \operatorname{sen}(\omega_c t)$$

Ou seja, apenas duas funções são necessárias para gerar o mesmo espaço. Assim, o novo espaço e mapeamento seriam

$$\phi_1(t) = \sqrt{\frac{2}{E_p}}\, p(t)\cos(\omega_c t) \qquad \phi_1(t) = -\sqrt{\frac{2}{E_p}}\, p(t)\operatorname{sen}(\omega_c t)$$
$$x_1 = (+1,0) \quad x_2 = (0,+1) \quad x_3 = (-1,0) \quad x_4 = (0,-1)$$

Seguindo este mesmo raciocínio, todos os esquemas de modulação *M*-PSK podem ser representados por este espaço ortonormal de sinais, variando apenas o mapeamento. A constelação do exemplo anterior, 4-PSK ou QPSK, é representada na Figura 2.12(a). Fazendo um mapeamento um pouco diferente, utilizando

$$x_1 = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right) \quad x_2 = \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right) \quad x_3 = \left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right) \quad x_4 = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right)$$

obtém-se a modulação QPSK com diferença de fase de π/4, representada na Figura 2.12(b). A componente enviada com o sinal de cosseno é chamada de componente em quadratura, já a componente enviada com o sinal de seno é chamada de componente em fase. Por isso, os eixos das constelações formadas por bases de sinais deste tipo são representados pelas letras Q (quadratura) e I (em fase).

Neste caso, a energia do sinal é igual a 1. Multiplicando qualquer um dos mapeamentos por *A*, obtém-se um mapeamento de energia média igual a A^2. Como cada símbolo carrega dois *bits,* a energia de um *bit* é a metade da energia de um símbolo. No caso geral, sendo *b* o

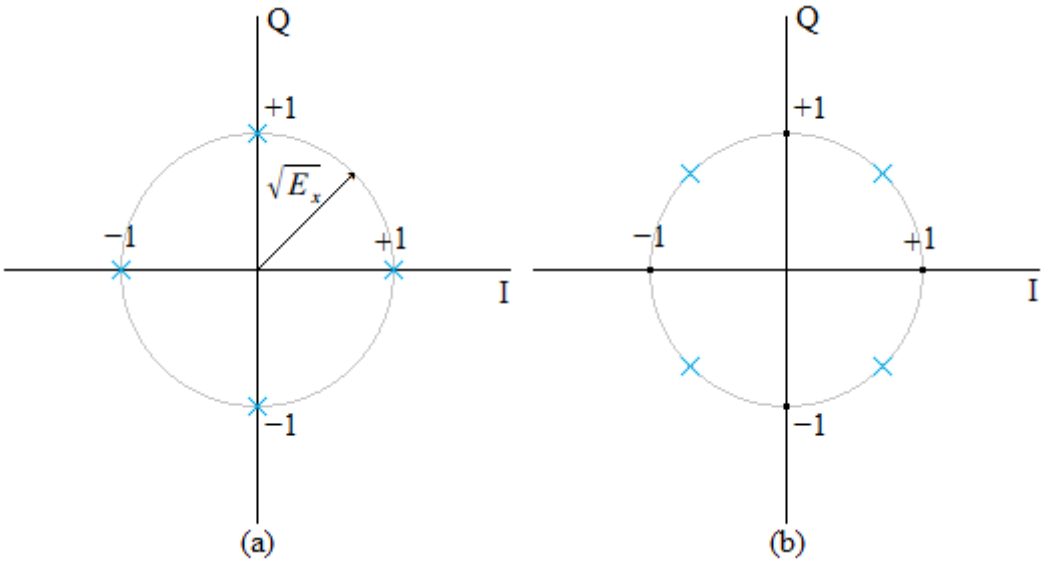


Figura 2.12. Constelações 4-PSK ou QPSK.

número de *bits* em um símbolo, a emergia de *bit,* $E_b = E_s / b$. Para a modulação PSK, é importante notar que no caso de apenas 1 *bit* por símbolo (BPSK, *binary phase shift keying*), tem-se a mesma modulação B-ASK, sendo necessário apenas um sinal da base.

Além das modulações em fase e em amplitude, há uma modulação mista, que é chamada de modulação em quadratura (*quadrature amplitude modulation*, QAM). Neste esquema de modulação, ambos os sinais da base, em quadratura e em fase, são modulados em amplitude por meio de um mapeamento. Em geral, é preferível que os símbolos distem igualmente uns dos outros nos eixos da constelação. A Figura 2.13 traz o exemplo de uma modulação 16-QAM.

Assim como nos esquemas anteriores, a energia média do sinal é calculada pela distância dos pontos à origem da constelação. Para o 16-QAM, utilizando a simetria dos símbolos, chega-se em

$$E_s = \frac{1}{16}\left[4\cdot(2\mathrm{A}^2)+8\cdot(10\mathrm{A}^2)+4\cdot(18\mathrm{A}^2)\right] = \frac{1}{16}(8\mathrm{A}^2+80\mathrm{A}^2+72\mathrm{A}^2) = 10\mathrm{A}^2.$$

De maneira geral, a energia média de um sinal *M*-QAM é dada por [20]

$$E_{M-QAM} = \frac{4\mathrm{A}^2(M-1)}{6} \tag{2.16}$$

Comparado com a modulação ASK, percebe-se que o uso de dois sinais representa uma enorme economia de energia. Lembrando que a modulação com metade dos símbolos modulados em ASK possui mais que o dobro de energia da modulação QAM.

O último esquema a ser apresentado é a modulação em frequência. Para este caso, a única representação possível é atribuir um sinal para cada símbolo. Exemplo de uma base de

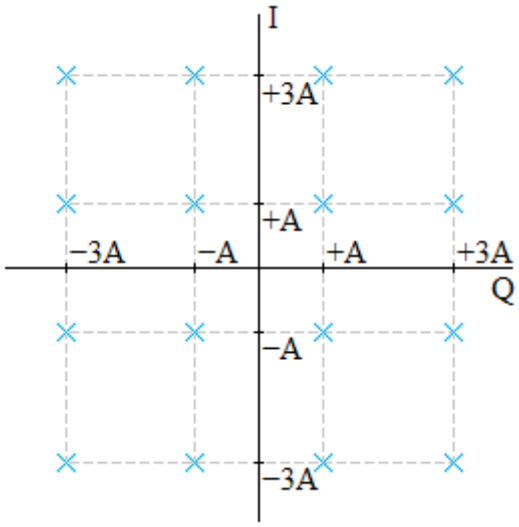


Figura 2.13. Constelação da modulação 16-QAM.

sinais BFSK é

$$\phi_1(t) = \sqrt{\frac{2}{E_p}}\, p(t)\cos(\omega_1 t),$$
$$\phi_2(t) = \sqrt{\frac{2}{E_p}}\, p(t)\cos(\omega_2 t). \tag{2.17}$$

Além desta representação genérica de sinais, as modulações digitais em banda base também podem ser obtidas da mesma forma como as modulações analógicas. O sinal modulante será a modulação digital em banda básica e, então, este sinal é usado para modular uma portadora. Para os casos da modulação QAM, utilizam-se duas modulações ao mesmo tempo em uma portadora ou modulam-se em amplitude duas portadoras ortogonais na mesma frequência.

A Figura 2.14 mostra graficamente os sinais de alguns tipos de modulação digital em banda passante, com $p(t)$ sendo um pulso retangular.

2.3.2. Demodulação

Para a demodulação de um sinal digital, é preciso recuperar o mapeamento dos símbolos a partir do sinal e recebido e, então, associar cada mapeamento a seu respectivo símbolo. Utilizando um espaço ortonormal de sinais para a modulação, os mapeamentos x_i podem ser recuperados a partir de cada pulso recebido calculando-se a função norma do pulso com a função de base. Lembrando que o pulso para o k-ésimo símbolo transmitido é dado pela Equação (2.3), tem-se que

$$\langle p_k(t), \phi_i(t)\rangle = \langle \sum_j x_{k,j}\phi_j(t), \phi_i(t)\rangle = \sum_j x_{k,j}\int_{-\infty}^{\infty} \phi_j(t)\phi_i(t)\,dt. \tag{2.18}$$

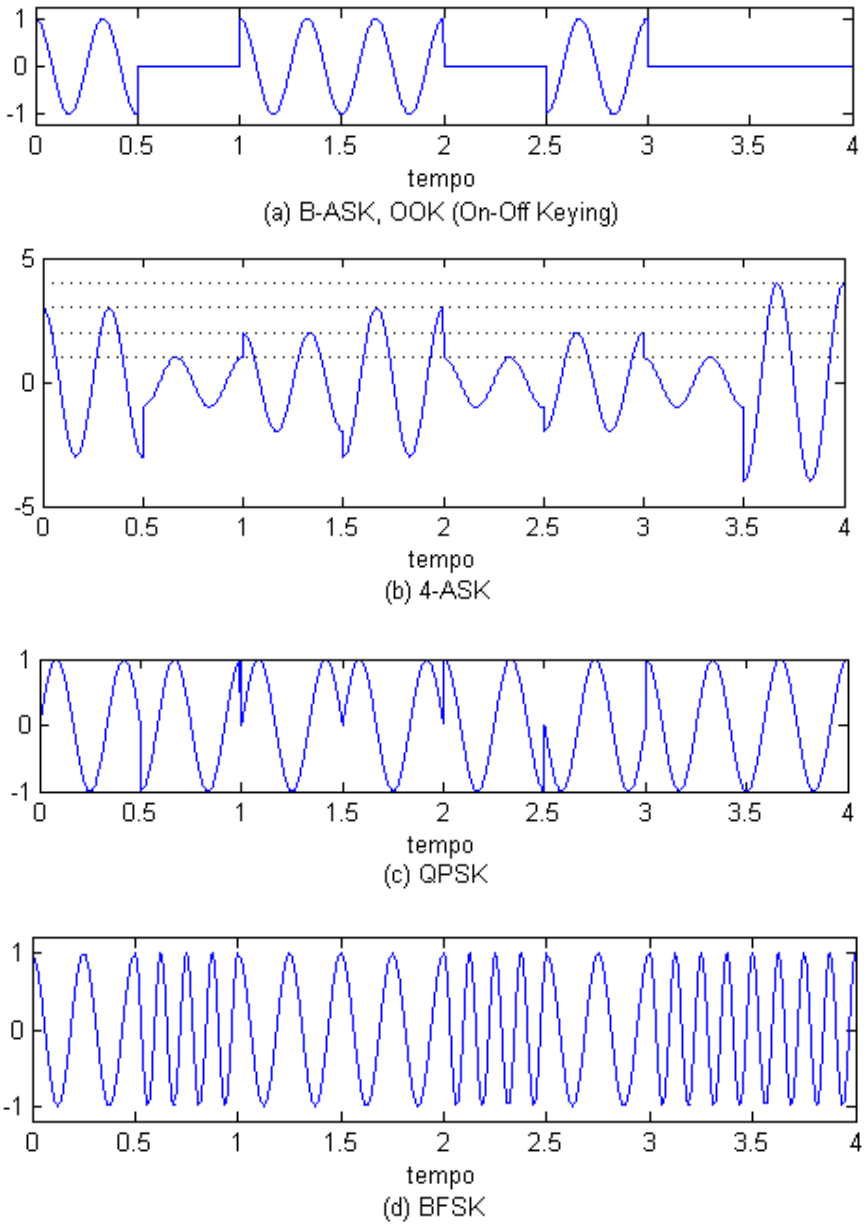


Figura 2.14. Esquemas de modulação digital em banda passante.

Como a base de sinais é ortonormal, a integral da Equação (2.18) tem o resultado igual a 1 para $i = j$ e 0 para $i \neq j$, resultado em

$$\langle p_k(t), \phi_i(t) \rangle = x_{k,i}, \tag{2.19}$$

recuperando o mapeamento do *k*-ésimo pulso com relação a função de base *i*. Contudo, o sinal recebido no demodulador não é dividido pulso a pulso, tornando complicada a aplicação direta dessa ferramenta matemática. Para contornar o problema, utiliza-se a ferramenta de convolução,

$$p(t) * q(t) = \int_{-\infty}^{\infty} p(\tau) q(t-\tau) d\tau. \tag{2.20}$$

Fazendo-se a convolução entre $s(t)$ e $\phi_i(-t)$, sendo $s(t)$ dado pela Equação (2.4), tem-se

$$s(t) * \phi_i(-t) = \int_{-\infty}^{\infty} s(\tau) \phi_i(-t+\tau) d\tau = \sum_k \int_{-\infty}^{\infty} p_k(\tau - kT) \phi_i(\tau - t) d\tau.$$

Tomando o resultado dessa convolução nos instantes $t = kT$,

$$s(t)*\phi_i(-t)\big|_{t=kT} = \sum_k \int_{-\infty}^{\infty} p_k(\tau - kT)\phi_i(\tau - kT)\,d\tau$$

e com o resultado da Equação (2.18),

$$s(t)*\phi_i(-t)\big|_{t=kT} = x_{k,i}. \tag{2.21}$$

Ou seja, para recuperar o mapeamento sinal transmitido referente a função $\phi_i(t)$, basta passar o sinal por um filtro de resposta impulsional igual a $\phi_i(-t)$ e amostrar em instantes de tempo iguais a T. O esquema completo do funcionamento de um demodulador desse tipo é mostrado na Figura 2.15.

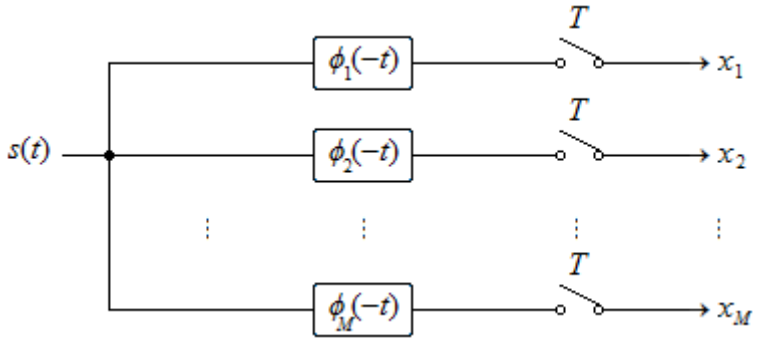


Figura 2.15. Demodulador digital.

Por enquanto, este demodulador é apenas um demapeador, visto que os mapeamentos recebidos são exatamente os enviados. Com isso, basta conhecer o mapeamento dos símbolos para descobrir qual símbolo foi recebido. Conforme erro for introduzido ao sinal, algumas medidas devem ser tomadas para decidir a qual símbolo o mapeamento recebido equivale. Basicamente, erro é introduzido devido a imperfeições no canal de comunicação, como a presença de ruído.

2.3.3. Canal de Comunicação

Canal de comunicação é o meio no qual as mensagens entre o transmissor e o receptor são transportadas. Se o canal para uma comunicação fosse perfeito, a informação recebida seria exatamente a informação transmitida. Infelizmente, não é isso que ocorre e o sinal está sujeito às imperfeições dos canais, que são o atraso, a atenuação e o ruído [20].

Atenuação é a perda da intensidade do sinal, que pode ocorrer por vários motivos. A propagação do sinal é motivo suficiente para a perda da potência do sinal. Da teoria eletromagnética, é sabido que a potência de uma onda é atenuada proporcionalmente ao quadrado da distância percorrida [22]. Outros fatores podem ainda contribuir para a atenuação dos sinais, como os fenômenos de espalhamento e absorção.

Por mais que os sinais eletromagnéticos viagem pelo meio muito rapidamente, na velocidade da luz, mais precisamente, a troca de informações não é instantânea. Sempre haverá uma diferença de tempo do momento em que o sinal é enviado até o momento em que este é recebido. Dependendo do meio em que o sinal é transmitido, haverá mais ou menos atraso. Esse fenômeno gera uma complicação maior ainda, que são os multipercursos. Quando um sinal é transmitido por um meio capaz de espalhar o sinal, o sinal pode acabar chegando a seu destino por mais de um caminho, devido a reflexões sofridas, como exemplificado na Figura 2.16 para o caso de uma comunicação num meio livre sem fio.

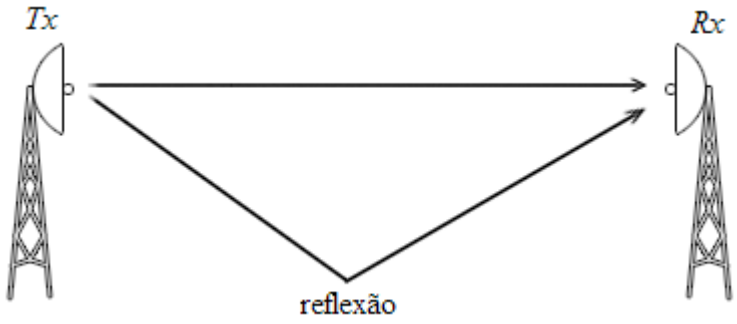


Figura 2.16. Multipercurso.

Como mostra a figura, o mesmo sinal transmitido por *Tx* é recebido duas vezes por *Rx*, sendo que, devido ao caminho mais longo, o sinal refletido chegará com um atraso maior. Muito provavelmente, a componente de multipercurso também chegará em *Rx* com menor intensidade. Canais com atraso e multipercurso podem ser modelados matematicamente por suas respostas impulsionais. Por exemplo, se há apenas uma componente de multipercurso e esta apresenta um atraso de τ unidades de tempo em relação a chegada da informação sem atraso e, ainda, apresenta atenuação α, a resposta impulsional deste canal é descrita como

$$h(t) = \delta(t) + \alpha \cdot \delta(t - \tau).$$

A depender da intensidade das componentes refletidas, o sinal pode ser destruído no receptor. Por exemplo, um pulso retangular de duração T é transmitido por um canal de resposta impulsional $h(t) = \delta(t) - 0{,}5 \cdot \delta(t - \tau) + 0{,}3 \cdot \delta(t - 3\tau)$, sendo $\tau = T/4$. O resultado da forma do pulso na transmissão e recepção é mostrado na Figura 2.17. Como pode ser visto, o pulso distorcido de duração maior que o pulso original, ou seja, o pulso se estende por mais de um símbolo. A interferência de um símbolo em outro é chamada de interferência intersimbólica, ou ISI (*intersymbol interference*). Para combater a interferência intersimbólica, é necessário que o canal seja equalizado, isto é, que sua resposta impulsional seja compensada. Existem diversas formas de se equalizar o canal, porém, todas elas precisam de uma boa estimativa da resposta impulsional do canal, que na prática é desconhecida e

precisa ser estimada. O caso mais simples de equalização é a técnica chamada *zero-forcing* (ZF). A equalização ZF consiste em apenas filtrar o sinal na entrada do receptor por um filtro cuja função de transferência é o inverso da função de transferência do canal [18]. Sendo a função de transferência de um filtro é igual a transformada de Fourier de sua resposta impulsional. Ou seja,

$$H(\omega) = \boldsymbol{TF}\{h(t)\} \tag{2.22}$$

e

$$F_{ZF}(\omega) = \frac{2\pi}{H(f)}, \tag{2.23}$$

sendo $F_{ZF}(\omega)$ a função de transferência do filtro equalizador e $\boldsymbol{TF}\{.\}$ a operação transformada de Fourier, dada pelo par de equações

$$\boldsymbol{TF}\{f(t)\} = F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt$$

$$\boldsymbol{TF}^{-1}\{F(\omega)\} = f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}\,dt\,.$$

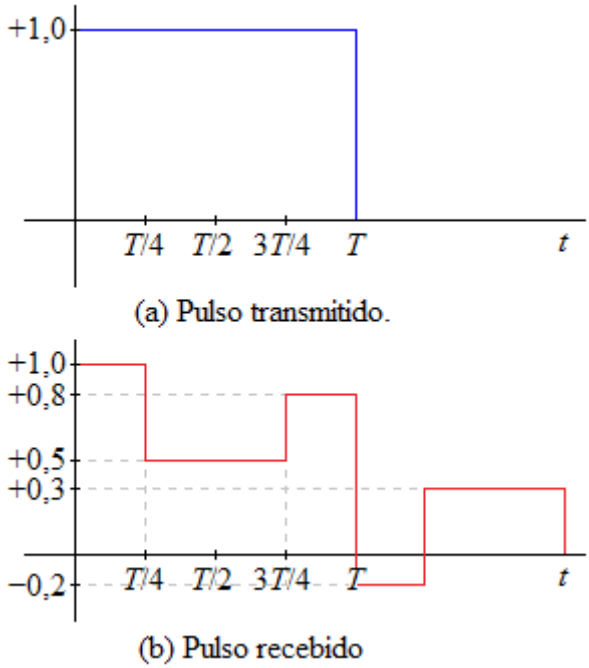


Figura 2.17. Forma do pulso (a) antes e (b) depois de ser transmitido pelo canal com multipercurso.

Além desses problemas, canais de comunicação também geram variações no sinal, chamadas de ruído. O principal fator responsável pelo ruído é a agitação das moléculas no meio de transmissão, gerando ruído térmico [20]. A Figura 2.18 traz um exemplo de ruído adicionado a um sinal. O modelo de canal com ruído mais comum é o canal AWGN (*additive white gaussian noise* – ruído aditivo branco gaussiano). O nome é devido à algumas características desse tipo de ruído, sendo que é branco pois sua densidade espectral de potência é constante, interferindo em todas as frequências igualmente, é gaussiano pois é

modelado por uma variável aleatória gaussiana, isto é, com uma distribuição normal de probabilidades, e é aditivo pois é somado ao sinal. A função de densidade de probabilidade do ruído gaussiano branco possui média nula e variância $N_0/2$, sendo N_0 o valor da densidade espectral de potência do ruído, que é constante. Isto é,

$$n(t) \sim N\left(0,\, N_0/2\right) \qquad (2.24)$$

e

$$\Psi_n(f) = N_0,$$

sendo que $X \sim N(\mu, \sigma^2)$ indica uma variável aleatória X de distribuição normal com média μ e variância σ^2. Neste caso, $n(t)$ é o sinal de ruído e $\Psi_n(f)$ a densidade espectral de potência de $n(t)$ para frequências positivas. Para o caso do ruído térmico, presente na maioria dos canais, $N_0 = k_B T$, sendo $k_B = 1{,}38 \cdot 10^{-23}$ J/K a constante de Boltzmann e T a temperatura equivalente de ruído [20].

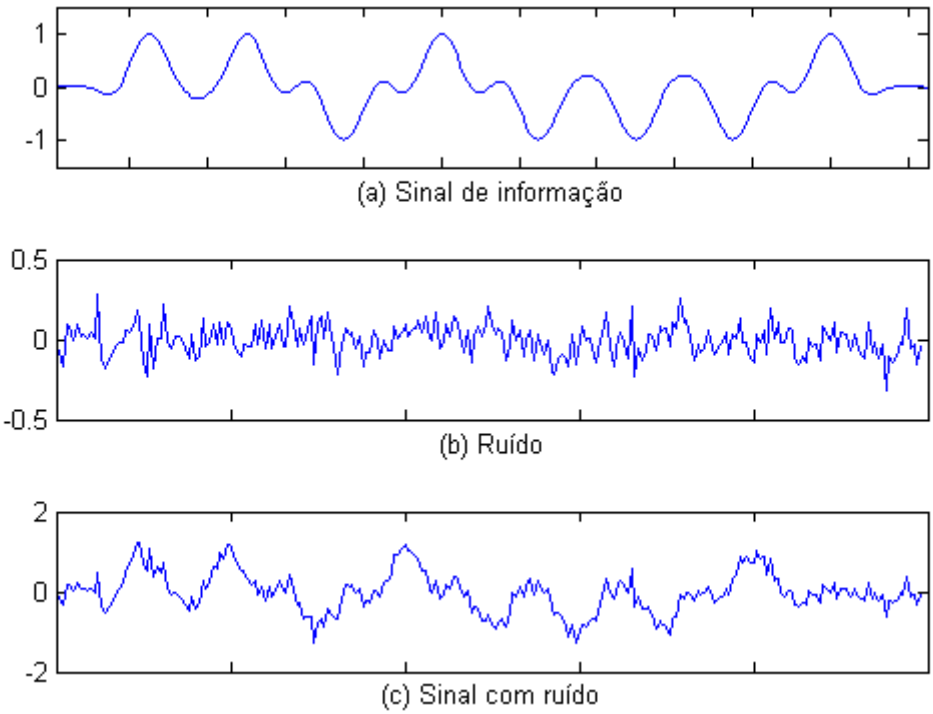


Figura 2.18. Sinal adicionado de ruído.

Uma modelagem adequada para um sistema de comunicação com um canal AWGN é simplesmente a adição de um sinal aleatório $n(t)$, como descrito pela Equação (2.24), ao sinal transmitido $s(t)$, gerando um sinal $r(t)$, assim como mostra a Figura 2.19.

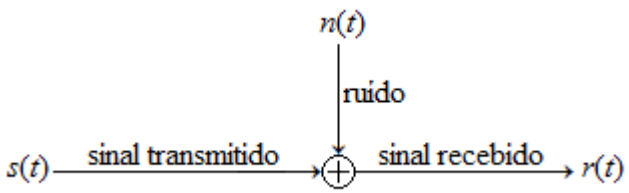


Figura 2.19. Modelo de um canal AWGN.

Sendo assim, o sinal recebido não é mais o mesmo transmitido, porém, a demodulação do sinal continua com os mesmos princípios. Realizando a convolução do sinal recebido pelas funções de base, tem-se que

$$\begin{aligned} r(t)*\phi_i(-t)\big|_{t=kT} &= [s(t)+n(t)]*\phi_i(-t)\big|_{t=kT} \\ &= s(t)*\phi_i(-t)\big|_{t=kT} + n(t)*\phi_i(-t)\big|_{t=kT}, \end{aligned}$$

devido à linearidade da convolução. O primeiro termo da soma é o mapeamento do *k*-ésimo símbolo com relação a base *i*, resultado da Equação (2.21). O segundo termo, devido a natureza aleatória de $n(t)$, continua sendo uma variável aleatória gaussiana, porém, é preciso conhecer sua média e variância. A média do sinal pode ser calculada por sua esperança, definida para sinais reais como

$$E[X] = \int_{-\infty}^{\infty} x f_X(x)\, dx,$$

sendo $f_X(x)$ a função densidade de probabilidade (PDF, *probability density function*) da variável aleatória X [20]. Assim,

$$E[n(t)*\phi_i(-t)] = \int_{-\infty}^{\infty} E[n(\tau)]\phi_i(\tau - t)\, d\tau = 0, \tag{2.25}$$

pois $\phi_i(t)$ é um sinal determinístico e $n(t)$ tem média nula. Para o cálculo da variância, que é definida como $VAR(X) = E[(X - \mu_X)^2]$, sendo $\mu_X = E[X]$,

$$\begin{aligned} VAR[n(t)*\phi_i(-t)] &= E\left\{\left[n(t)*\phi_i(-t)\right]^2\right\} \\ &= E\left[\int_{-\infty}^{\infty} n(\tau)\phi_i(\tau - t)\, d\tau \int_{-\infty}^{\infty} n(a)\phi_i(a - t)\, da\right] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E[n(\tau)n(a)]\phi_i(\tau - t)\phi_i(a - t)\, d\tau\, da \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \sigma_n^2 \delta(\tau - a)\phi_i(\tau - t)\phi_i(a - t)\, d\tau\, da \\ &= \sigma_n^2 \int_{-\infty}^{\infty} \phi_i(a - t)\phi_i(a - t)\, da \\ &= \sigma_n^2, \end{aligned}$$

em que σ_n^2 é a variância de $n(t)$ e vale $N_0/2$. O resultado final depende do fato que $\phi_i(t)$ pertence a uma base ortonormal de sinais e, ainda, $E[n(\tau)n(a)]$ só é não-nulo para $\tau = a$ [18]. Ou seja, o mapeamento recebido é

$$r_{k,i} = r(t)*\phi_i(-t)\big|_{t=kT} = x_{k,i} + z, \tag{2.26}$$

sendo $$Z \sim N(0, N_0/2).$$

Este resultado mostra que o mapeamento obtido possui uma componente de ruído com as mesmas características do canal. Com isso, há uma certa probabilidade do processo de demodulação conter error. Para ilustrar, seja uma modulação simples, como a BPSK, cuja

constelação é mostrada na Figura 2.10 (que também é uma modulação BPSK). Uma regra simples de decisão para o símbolo é escolher como o símbolo 0 o mapeamento demodulado com valor positivo e como símbolo 1 o mapeamento demodulado com valor negativo. A Equação (2.26) mostra que o sinal recebido demodulado é uma variável aleatória gaussiana com média $x_{k,i}$ e variância $N_0/2$. Como na modulação BPSK $x = \{+A, -A\}$, a PDF condicional do mapeamento recebido é como mostrado na Figura 2.20.

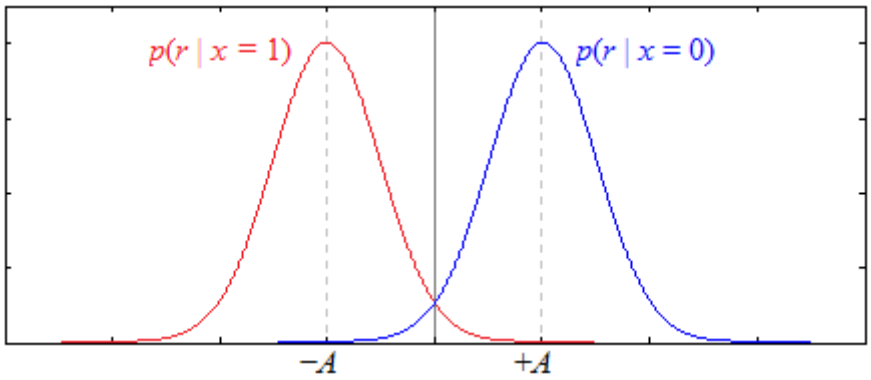


Figura 2.20. PDF condicional do mapeamento BPSK recebido.

Pela figura, é possível perceber que se a variância (potência) do ruído for grande o suficiente, o mapeamento será demodulado com erro. Com a regra de demodulação descrita, haverá erro com probabilidade

$$p_e = p(r>0, x=1) + p(r<0, x=0),$$

que pode ser calculada pela regra de Bayes como

$$p_e = p(r>0 \mid x=1)\, p(x=1) + p(r<0 \mid x=0)\, p(x=0).$$

Como os símbolos são equiprováveis, $p(x = 0) = p(x = 1) = 1/2$. Além disso, a distribuição de probabilidades das curvas são gaussianas com médias $+A$ e $-A$ e variâncias iguais a $N_0/2$. Da teoria de probabilidades, a probabilidade de uma distribuição gaussiana X de média μ e variância σ^2 pode ser calculada pela função Q como

$$p(X>x) = Q\left(\frac{x-\mu}{\sigma}\right), \tag{2.27}$$

sendo a função Q definida como um menos a função distribuição acumulada de uma variável aleatória gaussiana [20], em que

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{u^2}{2}}\, du,$$

com a propriedade

$$1 - Q(x) = Q(-x). \tag{2.28}$$

A probabilidade de erro da modulação BPSK com o limiar de decisão igual a 0 é, então,

$$p_e = \frac{1}{2}Q\left(\frac{0+A}{\sqrt{N_0/2}}\right)+\frac{1}{2}\left[1-Q\left(\frac{0-A}{\sqrt{N_0/2}}\right)\right] = \frac{1}{2}Q\left(\sqrt{\frac{2A^2}{N_0}}\right)+\frac{1}{2}Q\left(\sqrt{\frac{2A^2}{N_0}}\right)$$

$$p_e = Q\left(\sqrt{\frac{2A^2}{N_0}}\right). \tag{2.29}$$

Nas comunicações analógicas, a qualidade do sinal recebido é quantificada pela razão entre a potência do sinal e a potência do ruído, conhecida como relação sinal-ruído, ou SNR (s*ignal to noise ratio*). Em comunicações digitais, a potência do sinal está diretamente ligada à energia por *bit* da modulação, E_b, e a potência do ruído é ligada à sua densidade espectral de potência, N_0. Então, uma quantificação da qualidade do sinal nas modulações digitais é definida pela razão E_b/N_0. Como visto anteriormente, a energia média da modulação BPSK é A^2. Para melhor comparação entre os esquemas de modulação, a probabilidade de erro de *bit* será uma função dessa razão, assim, a Equação (2.29) se torna

$$p_e = Q\left(\sqrt{2\frac{E_b}{N_0}}\right). \tag{2.30}$$

A taxa de erro de *bit,* BER (*bit error rate*), de uma transmissão é a razão entre quantidade de *bits* demodulados com erro e a quantidade de *bits* enviados. Sendo d_i o i-ésimo *bit* demodulado e b_i o i-ésimo *bit* enviado, a BER da transmissão de N_b *bits* é

$$BER = \frac{1}{N_b}\sum_i (d_i \neq b_i) \tag{2.31}$$

e, pela lei dos grandes números [23], a medida que $N \to \infty$, $BER \to p_e$. Resta saber se o limiar igual a 0 é a melhor forma de optar pelo símbolo demodulado.

O critério de demodulação utilizado no exemplo foi puramente intuitivo. Para formalizar o critério de decisão, será utilizado o critério de máxima verossimilhança, ML (*maximum likelihood*), que consiste em maximizar a probabilidade de acertar o *bit* demodulado [19]. Por esse critério, como é mostrado em [18, 19], maximizar a probabilidade de acerto consiste em minimizar a distância entre o mapeamento recebido e o mapeamento da modulação. Para o caso do BPSK, qualquer ponto localizado na região positiva da modulação estará mais próximo do símbolo mapeado em $+A$ e qualquer ponto na região negativa, mais próximo do símbolo mapeado em $-A$. Ou seja, o limiar igual a zero é o limiar ótimo de decisão. Portanto, um demodulador digital sob um canal AWGN é um tomador de decisão com critério de máxima verossimilhança como é mostrado na Figura 2.21.

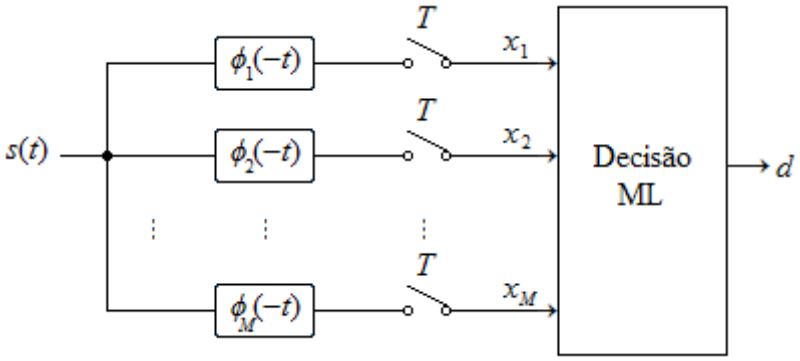


Figura 2.21. Demodulador digital com critério de máxima verossimilhança em canal AWGN.

Pelo critério ML, é possível traçar na constelação regiões de decisão. Essas regiões delimitam as mínimas distâncias entre um mapeamento recebido e o mapeamento da constelação. Na modulação BPSK, por exemplo, existem apenas duas regiões de decisão, a região maior ou igual a zero e a região menor que zero. No 8-PSK, as regiões de decisão formam uma região de decisão como mostra a Figura 2.22. Basicamente, para se obter as regiões de decisão, basta traçar o Diagrama de Voronoi [24] dos pontos da constelação, para isso, é preciso traçar um seguimento de reta entre os pontos vizinhos da constelação e, então, traçar uma linha ortogonal a cada seguimento que corta sua mediana. Esse última linha será a delimitação da região do símbolo, que se estende até encontrar a próxima linha de delimitação, como mostra a Figura 2.23.

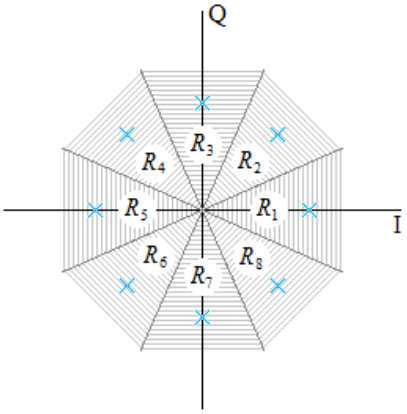


Figura 2.22. Regiões de decisão 8-PSK.

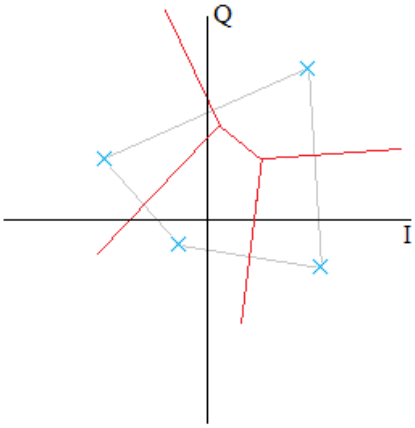


Figura 2.23. Construção das regiões de decisão de uma constelação qualquer (Diagrama de Voronoi).

2.4. CONCLUSÃO

As comunicações digitais uniram as comunicações e a informática. A demanda por processamento e transmissão de dados é cada vez maior, requerendo técnicas cada vez mais avançadas para alcançar os novos objetivos. Como visto, o emprego de modulações carregando muitos *bits* por símbolo pode elevar significativamente as taxas de dados. Por outro lado, a transmissão de dados está sujeita a erros, como o modelo de canal AWGN mostrou. Um relação direta que pode ser inferida deste capítulo é que quanto maior o número de símbolos por *bit,* em geral, maior também é a taxa de erro do sistema, devido a constelações com pontos muito próximos. No próximo capítulo será visto como técnicas de processamento de dados podem ajudar a detectar ou até mesmo corrigir eventuais erros ocorridos na transmissão. Outro ponto importante é o limite das comunicações digitais, isto é, até que ponto é possível aumentar a taxa de dados de um sistema e este continuar operando satisfatoriamente, isto é, com baixas taxas de erro.

3. CÓDIGOS CORRETORES DE ERRO

3.1. INTRODUÇÃO

Como visto na subseção 2.3.3, o canal de comunicação pode alterar o sinal de tal maneira que haja erro durante a transmissão de dados digitais. Na maioria das vezes, os usuários de tecnologias que fazem uso das comunicações digitais contam com a garantia da entrega de seus dados. Por exemplo, ao utilizar um aplicativo de mensagens instantâneas, o remetente confia que o que foi lido por seu destinatário é exatamente o que foi escrito por ele.

Um sistema digital de comunicação deve, portanto, ser capaz de reconhecer erros e agir. Existem, basicamente, duas estratégias consolidadas para garantir a transmissão correta dos dados. A primeira consiste em apenas identificar o erro e fazer a retransmissão até que os dados sejam corretamente recebidos, por exemplo, o método ARQ (*automatic repeat request*). A segunda, que será abordada neste capítulo, consiste em, além de identificar o erro, ser capaz de tratá-lo, corrigindo-o eventualmente e, caso não seja possível, solicitar a retransmissão [3].

Neste capítulo, serão abordadas as técnicas lineares de codificação de canal, iniciando o estudo sobre códigos de bloco sistemáticos para, então, concluir com a introdução dos códigos convolucionais. Serão vistas as diferentes formas de decodificação destes códigos, enfatizando a decodificação por decisão *soft*.

3.2. REDUNDÂNCIA

O fundamento dos códigos corretores de erro é a inserção de redundância nos dados, adicionando informações extras à mensagem com o propósito de verificar a validade dos dados que originalmente a compunham. A exemplo disso estão os dígitos verificadores presentes, por exemplo, no Cadastro de Pessoa Física (CPF) brasileiro. O CPF é composto por 11 números no total, sendo os dois últimos, seus dígitos verificadores. A função dos dígitos verificadores é validar o conjunto de números apresentados. No caso do CPF, composto por 9 números mais dois dígitos verificadores, no formato $a_1a_2a_3\,.\,a_4a_5a_6\,.\,a_7a_8a_9$ - d_1d_2, a validação é dada pelo algoritmo

$$
\begin{aligned}
d_1 &\leftarrow 1\cdot a_1 + 2\cdot a_2 + 3\cdot a_3 + \ldots + 9\cdot a_9 \\
d_1 &\leftarrow d_1 \bmod 11 \\
d_1 &\leftarrow d_1 \bmod 10 \\
d_2 &\leftarrow 1\cdot a_2 + 2\cdot a_3 + \ldots + 8\cdot a_9 + 9\cdot d_1 \\
d_2 &\leftarrow d_2 \bmod 11 \\
d_2 &\leftarrow d_2 \bmod 10,
\end{aligned}
$$

sendo x mod y a operação módulo, que é o resto inteiro da divisão de x por y. Sendo assim, é

possível dizer que o CPF 123.456.789-09 é um CPF válido, o que não é verdade para o mesmo CPF com o último dígito verificador trocado, 123.456.789-02, por exemplo. O CPF contém informações acerca de um cidadão brasileiro, como sua unidade federativa de nascimento, por exemplo. Por isso, é importante que quando alguém apresente seu CPF, este seja um número válido. O mesmo serve para as comunicações digitais. Muitos protocolos de rede utilizam verificadores para validar os dados enviados, como por exemplo o campo *checksum* do protocolo TCP [25]. Um outro mecanismo similar é o verificador de paridade, cujo funcionamento consiste em adicionar um *bit* na mensagem que é 1 se o número de *bits* 1 na mensagem for par e 0 se este número for ímpar, por exemplo. Assim, a mensagem 11001011 com verificador de paridade seria 110010110, com o último *bit* sendo o *bit* de paridade.

Essa é uma primeira forma de verificação de erros. Contudo, não é possível corrigir o erro quando este é detectado. Para realizar a correção de erros, é preciso que mais informação redundante seja adicionada, de tal forma um código corresponda a apenas uma mensagem. Um exemplo para isso e uma forma trivial de codificação é o código de repetição, que consiste em enviar a mesma informação mais de um vez. Por exemplo, para enviar a mensagem binária 1101001 com repetição de 3 *bits* pra 1, será enviada a mensagem 111111000111000000111. Deste modo, o código 111 sempre corresponde à mensagem 1, o mesmo para o código 000 com relação à mensagem 0.

Contudo, inserir informação redundante na mensagem implica em reduzir a taxa efetiva de transmissão do sistema. Uma mensagem com redundância de 3 para 1 terá sua velocidade de transmissão reduzida para um terço apenas. É importante, então, definir até que ponto a inserção de redundância é benéfica ao sistema. Um exemplo em que a redundância é interessante são as aplicações que necessitam de baixa latência e taxas de dados não muito elevadas, como chamadas de voz. Nesses casos, retransmitir os dados acarretaria um aumento significativo na latência, sendo válido investir em codificações com melhores capacidades de detecção/correção, até porque alguns erros podem ser tolerados nesses tipos de aplicações.

Para analisar qualitativamente o processo de codificações é preciso um meio de quantizar o ganho advindo da adição de redundância. Primeiramente, será necessário modelar o canal de comunicação de maneira a possibilitar uma análise do desempenho das codificações.

Como visto no capítulo anterior, os canais podem possuir elevada complexidade. A fim de simplificar o processo de transmissão dos sinais, um modelo genérico de canal pode ser elaborado.

3.3. CANAL BINÁRIO SIMÉTRICO

O canal binário simétrico, BSC (*binary symmetric channel*), é uma simplificação de todo sistema de comunicação, desde o transmissor até o receptor. Este modelo de canal reduz todas as etapas da comunicação (modulação, canal, recepção e demodulação) em apenas uma transição de *bits*, com probabilidades de erro e sucesso para o *bit* enviado. A Figura 3.1 mostra essa simplificação do canal [3].

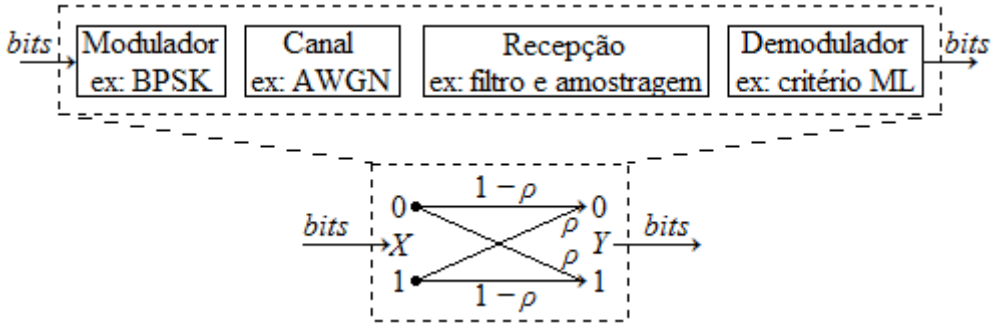


Figura 3.1. Simplificação do canal de comunicação para o canal BSC.

Como o canal é binário, apenas os símbolos 0 e 1 são transmitidos e recebidos e, por ser simétrico, as probabilidades de falha são iguais, isto é, $p(y = 0 \mid x = 1) = p(x = 1 \mid x = 0) = \rho$, sendo x o *bit* enviado e y o *bit* recebido. Analogamente, a probabilidade de sucesso é dada por $p(y = 0 \mid x = 0) = p(y = 1 \mid x = 1) = 1 - \rho$. Para exemplificar, uma comunicação que utiliza modulação BPSK sobre um canal AWGN pode ser satisfatoriamente aproximada por um canal BSC com $\rho = Q\left(\sqrt{2\,E_b/N_0}\right)$, como dado pela Equação (2.30).

Da teoria da informação, desenvolvida por Shannon [2], os conceitos de entropia e informação mútua serão aplicados para motivar e otimizar o uso de códigos corretores de erro. A entropia de uma variável aleatória é uma medida da quantidade média de informação contida nas mensagens e é calculada como

$$H(X) = -\sum_{x \in X} p(x) \log_2 p(x). \tag{3.1}$$

A base do logaritmo define o tipo de informação é medida pela entropia, a base 2 calcula a quantidade média de *bits* de informação na mensagem. Para um alfabeto binário $B = \{b_1, b_2\}$ com distribuição de probabilidade $P_B = \{p, 1 - p\}$, define-se a entropia binária em função de uma de suas probabilidades como

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$

Além disso, a informação mútua de duas variáveis é quanto de informação uma variável carrega acerca de outra, e é calculada como

$$I(X, Y) = H(Y) - H(Y \mid X), \tag{3.2}$$

sendo $$H(Y \mid X) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y \mid x) \tag{3.3}$$

a entropia conjunta de X e Y. Com essas informações é possível definir o conceito de capacidade de canal no canal BSC.

3.3.1. Capacidade de Canal

A capacidade de canal é um dos principais resultados da teoria da informação [23] e é definida como o máximo da informação mútua por cada utilização do canal com relação às distribuições de probabilidade *a priori*, ou seja,

$$C = \max_{p(X)} \; I(X, Y). \tag{3.3}$$

A capacidade de canal é medida em unidades de informação (neste trabalho, essa unidade é o *bit*) por intervalo de utilização do canal.

Para o caso do canal BSC, a capacidade pode ser encontrada a partir das Equações (3.1), (3.2) e (3.3). Sejam $X = \{0, 1\}$ e $Y = \{0, 1\}$ duas variáveis representando os *bits* enviados e recebidos, respectivamente, e

$$p_0 = p(X = 0) \;\text{ e }\; p_1 = p(X = 1) = 1 - p_0.$$

Tem, se que

$$p(Y = y) = \sum_{x \in X} p(Y = y \mid X = x)\, p(X = x).$$

Com as propriedades do canal BSC,

$$p(Y = y \mid X = x) = \begin{cases} \rho, & \text{se } x \neq y \\ 1 - \rho, & \text{se } x = y \end{cases}$$

Assim, pela Equação (3.1) mostra-se que

$$H(Y) = -p_y \log_2 p_y - (1 - p_y) \log_2 (1 - p_y),$$

sendo $p_y = p(Y = 0) = p_0 + \rho - 2p_0\rho$. Pela Equação (3.3),

$$\begin{aligned} H(Y \mid X) &= -(1-\rho)\, p_0 \log_2 (1-\rho) - \rho (1 - p_0) \log_2 \rho \\ &\quad - \rho\, p_0 \log_2 \rho - (1-\rho)(1 - p_0) \log_2 (1-\rho) \\ H(Y \mid X) &= -(1-\rho) \log_2 (1-\rho) - \rho \log_2 \rho. \end{aligned}$$

Usando a Equação (3.2),

$$I(X, Y) = H(Y) + \rho \log_2 \rho + (1-\rho) \log_2 (1-\rho).$$

O único termo dependente da probabilidade a priori p_0 é $H(Y)$, que é a entropia binária com

probabilidade $p_y = p_0 + \rho - 2p_0\rho$. O máximo da entropia binária é 1 e ocorre quando os *bits* são equiprováveis, assim, $p_y = 1/2 \rightarrow H(Y) = 1$. Quando $p_y = 1/2$, também implica $p_0 = 1/2$. Ou seja, a capacidade do canal BSC é dada quando os *bits* enviados são equiprováveis e vale

$$C = 1 + \rho \log_2 \rho + (1-\rho)\log_2(1-\rho)$$

ou

$$C = 1 - H(\rho), \tag{3.4}$$

sendo $H(\rho)$ a entropia binária com probabilidades ρ e $1 - \rho$. Esse resultado servirá de base para quantificar o desempenho de códigos corretores de erro, mas primeiramente são necessários alguns conceitos sobre esses códigos.

3.3.2. Tamanho dos Códigos

Quando uma mensagem é codificada, seu tamanho aumenta conforme o tamanho do código utilizado, isto é, a quantidade de símbolos utilizados para codificar uma certa quantidade de símbolos da mensagem. Isto é, na codificação há um mapeamento da mensagem em um código. Sendo assim, cada bloco de k símbolos da mensagem é convertida em um código de n símbolos, com $n > k$. Um código é representado, assim, pela notação (n, k) em que

$$(m_1, m_2, \ldots, m_k) \rightarrow (c_1, c_2, \ldots, c_n).$$

A mensagem codificada final terá, portanto, um comprimento total n/k vezes maior. O inverso dessa quantia é chamado de taxa de código,

$$R_c = \frac{k}{n}. \tag{3.5}$$

A taxa de código está diretamente relacionada com a taxa de transmissão do sistema. A de transmissão do sistema R_s *baud* será utilizada para transmitir dados da mensagem codificados, isto é, com redundância. Na recepção, a redundância será eliminada para recuperar a mensagem original, fazendo com que, de todo dado transmitido, apenas uma fração R_c dos dados seja informação útil. Assim, a taxa de transmissão dita efetiva do sistema R, será $R = R_s \cdot R_c$ *baud*.

3.3.3. Teorema da Codificação de Canal

Um resultado fundamental da teoria da informação na área das comunicações é o teorema da codificação de canal desenvolvido por Shannon [2]. Esse teorema relaciona a capacidade do canal transferir informação com a taxa de código utilizada pela fonte para gerar informação, sendo aplicado a canais discretos e sem memória, como o canal BSC, por exemplo.

Teorema. Um fonte X de entropia $H(X)$ produz símbolos à razão de um símbolo por cada T_s segundos. Se um canal discreto e sem memória de capacidade C *bits*/símbolo é utilizado uma vez a cada T_c segundos, então existe uma técnica de codificação dos símbolos da fonte que permite uma transmissão com probabilidade de erro arbitrariamente pequena tal que

$$\frac{H(x)}{T_s} \leq \frac{C}{T_c}. \tag{3.6}$$

Se, pelo contrário, $\frac{H(x)}{T_s} > \frac{C}{T_c}$, então não é possível garantir a transmissão dos símbolos com probabilidade de erro arbitrariamente pequena.

Aplicando esse resultado ao canal BSC, supondo informações geradas de maneira equiprovável, tem-se que $H(X) = 1$. A fonte gera um *bit* a cada T_s segundos, que são codificados e transmitidos pelo canal. O codificador, por sua vez, gera um *bit* a cada T_c segundos, mesmo período de utilização do canal. Assim, e Equação (3.6) se torna

$$\frac{1}{T_s} \leq \frac{C}{T_c} \Rightarrow \frac{T_c}{T_s} \leq C.$$

Porém, a razão T_c/T_s é a mesma que k/n, ou seja, a taxa de código. Assim, a Equação (3.6) pode ser reescrita para o canal binário simétrico como

$$R_c \leq C. \tag{3.7}$$

Como quanto maior a taxa de código maior é a taxa efetiva de transmissão, o teorema da codificação de canal pode ser utilizado para quantificar a eficiência de um código. A capacidade de canal é uma cota superior da taxa de código, significando que um bom código é aquele que garante taxas de erro arbitrariamente pequenas com uma taxa de código próxima à capacidade de canal.

3.4. CÓDIGOS LINEARES

Códigos lineares são aqueles em que a informação redundante inserida na mensagem é formada por combinações lineares da própria mensagem [3]. Tradicionalmente, os códigos lineares são subdivididos em códigos de bloco e códigos convolucionais.

A partir desta seção, serão abordadas apenas as codificações binárias e as operações binárias serão realizadas em módulo 2, isto é

$$\begin{array}{cccc} 0+0=0 & 0+1=1 & 1+0=1 & 1+1=0 \\ 0\cdot 0=0 & 0\cdot 1=0 & 1\cdot 0=0 & 1\cdot 1=1. \end{array}$$

Um exemplo de código linear é o código de repetição exemplificado na seção 3.2. Códigos de repetição são do tipo $(n, 1)$ com $n = 2q + 1$ sendo um número ímpar. O uso de n

ímpar é devido a decodificação por maioria, isto é, quando um código é recebido, o *bit* decodificado é aquele que mais aparece no código. Por exemplo, com $n = 3$, se o código recebido for 101, o *bit* decodificado por maioria é 1. Um código será decodificado errado se houver erro na maioria dos *bits* deste código, ou seja, se houverem $q + 1$ erros. Em um canal BSC com probabilidade de erro ρ, a probabilidade de erro usando codificação por repetição, como mostrado em [3], é

$$P_e = \sum_{i=q+1}^{n} \binom{n}{i} \rho^i (1-\rho)^{n-i}. \tag{3.8}$$

Para avaliar o desempenho da codificação por repetição, seja o canal BSC um modelo de canal AWGN para uma transmissão BPSK com $E_b/N_0 = 2$ dB. Usando a Equação (2.30), encontra-se que $\rho \approx 3{,}75 \cdot 10^{-2}$. Com isso, a entropia do canal é dada pela entropia binária e é $H(\rho) \approx 0{,}2307$ *bits* e a capacidade do canal, dada pela Equação (3.4), é $C = 0{,}7693$ *bit* / utilização do canal. A Tabela 3.1 relaciona a taxa do código de repetição com a probabilidade de erro de *bit* para este canal.

Tabela 3.1. Probabilidade de erro de *bit* da codificação por repetição em função de sua taxa.

$\mathbf{R_c}$	1/3	1/5	1/7	1/9
$\mathbf{P_e}$	$4{,}11 \cdot 10^{-3}$	$4{,}98 \cdot 10^{-4}$	$6{,}32 \cdot 10^{-5}$	$8{,}24 \cdot 10^{-6}$

A maior taxa de código garante uma probabilidade de erro de *bit* aproximadamente dez vezes menor, porém essa taxa não é nem metade da capacidade do canal. Conforme a taxa diminui, mais distante ela fica da capacidade do canal. Uma comunicação moderna requer taxas de erro de *bit* da ordem de 10^{-6} [22], usando códigos de repetição neste cenário, a taxa que garante isso é 1/9, com valor muito distante da capacidade de canal. Ou seja, códigos de repetição não são eficientes e, segundo o Teorema da Codificação de Canal, existem códigos melhores para taxas de erro de *bit* ainda mais baixas. Para estudar outros sistemas de codificação de canal, é preciso saber como construí-los.

3.4.1. Distância Mínima

A distância mínima de um código linear é definida como a menor distância de Hamming entre suas palavras código. A distância de Hamming entre duas palavras código é a quantidade de *bits* diferentes entre si. Por exemplo, as palavras 110110101 e 110010011 diferem nos *bits* 2, 3 e 6, sendo o *bit* mais a direita o primeiro, e, portanto, a distância de Hamming, d_H, entre elas é igual a 3. Na codificação por repetição, existem apenas duas palavras código formada pela repetição dos *bits*, fazendo com que a distância mínima destes códigos seja igual ao

número de *bits* das palavras códigos, isto é, um código de repetição (n, 1) tem distância mínima $d_{min} = n$.

Códigos lineares são capazes de detectar $d_{min} - 1$ erros e de corrigir $\lfloor (d_{min} - 1)/2 \rfloor$ erros, em que $\lfloor x \rfloor$ o maior inteiro menor ou igual a x [3]. Ou seja, além da taxa de um código, outra propriedade importante é sua distância mínima.

3.4.2. Códigos de Bloco

Nos códigos de bloco, as mensagens são codificadas em blocos. Isto é, uma mensagem composta por k *bits* é mapeada em uma palavra código de n *bits,* sendo que duas mensagens iguais geram duas palavras códigos iguais [3]. Sendo assim, existem 2^k possíveis mensagens codificadas em 2^n possíveis códigos. Porém, destes 2^n códigos, apenas 2^k representam a codificação de uma mensagem, isto é, apenas 2^k são palavras código. A detecção e correção de erro de códigos de bloco se baseia nessa ideia, como é mostrado pela Figura 3.2.

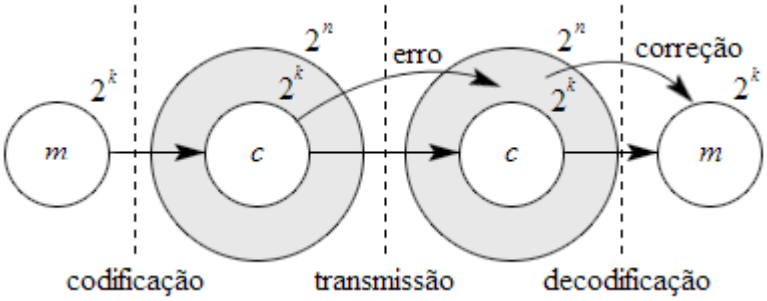


Figura 3.2. Exemplo de comunicação com codificação.

Códigos de bloco são ditos sistemáticos quando as palavras código das mensagens são compostas pela própria mensagem e *bits* de paridade, na forma

$$(c_1, c_2, \ldots, c_n) = (m_1, m_2, \ldots, m_k, p_1, p_2, \ldots, p_{(n-k)}).$$

Por se tratar de um código linear, os *bits* de paridade são combinações lineares dos *bits* da mensagem. Por exemplo, um código de bloco sistemático (6, 4) possui 2 *bits* de paridade e pode ser obtido como

$$c_{(k+1)} = p_1 = m_1 + m_2$$

e

$$c_{(k+2)} = p_2 = m_2 + m_4.$$

Neste exemplo, o código da mensagem 1100 seria 110001, pois os *bits* de paridade são obtidos com $m_1 + m_2 = 1 + 1 = 0$ e $m_2 + m_4 = 1 + 0 = 1$. Organizando a mensagem como um vetor linha, a operação de geração de um código de bloco sistemático pode ser obtida como uma operação matricial. A matriz $\boldsymbol{G}$, chamada de matriz geradora, do código exemplificado é descrita como

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

As k primeiras colunas formam uma matriz identidade, resultando nas k primeiras posições da palavra código. As $n - k$ colunas seguintes representam as combinações dos *bits* da mensagem que formam os *bits* de paridade. Assim, a matriz geradora de um código de bloco sistemático pode ser obtida como

$$\boldsymbol{G}_{k\times n} = \left[\boldsymbol{I}_k \,\middle|\, \boldsymbol{P}_{k\times(n-k)}\right], \tag{3.9}$$

sendo $\boldsymbol{P}_{k\times(n-k)}$ a matriz que determina a paridade dos códigos e $\boldsymbol{I}_k$ a matriz identidade de tamanho $k\times k$. Assim, a palavra código $\boldsymbol{c}$ referente a mensagem $\boldsymbol{m}$ pode ser encontrada como

$$\boldsymbol{m}\boldsymbol{G} = \boldsymbol{c}. \tag{3.10}$$

Os vetores $\boldsymbol{c}$ e $\boldsymbol{m}$ são ambos vetores linha. Além da matriz geradora, os códigos de bloco possuem uma matriz de verificação de paridade, $\boldsymbol{H}$, cuja principal propriedade é

$$\boldsymbol{G}\boldsymbol{H}^T = \boldsymbol{0}, \tag{3.11}$$

Sendo $\boldsymbol{0}$ uma matriz composta apenas de elementos nulos. Com a matriz de verificação de paridade é possível validar se uma palavra recebida é uma palavra código, pois, sendo $\boldsymbol{C}$ o conjunto de todas as 2^k palavras código de um esquema de codificação de bloco,

$$\begin{aligned} &\boldsymbol{c}\boldsymbol{H}^T = \boldsymbol{0}, \text{ se } \boldsymbol{c}\in\boldsymbol{C} \\ \text{e}\quad &\boldsymbol{c}\boldsymbol{H}^T \neq \boldsymbol{0}, \text{ se } \boldsymbol{c}\notin\boldsymbol{C}. \end{aligned} \tag{3.12}$$

A matriz $\boldsymbol{H}$ pode ser construída sistematicamente como

$$\boldsymbol{H}_{(n-k)\times n} = \left[\boldsymbol{P}^T \,\middle|\, \boldsymbol{I}_{(n-k)}\right]. \tag{3.13}$$

3.4.3. Códigos de Hamming

Códigos de Hamming são códigos de bloco lineares com distância mínima igual a 3. Estes códigos são especiais pois são os códigos com as maiores taxas possíveis para essa distância mínima. A estrutura dos códigos de Hamming é $(2^q - 1, 2^q - i - 1)$, $q = 2, 3, 4, \ldots$, em que $q = n - k$ é a quantidade de *bits* de paridade do código [3]

Nos códigos de Hamming sistemáticos, a matriz de paridade, $\boldsymbol{P}$, é construída pelas k combinações linearmente independentes de $n - k$ *bits*, isto é, as combinações binárias $n - k$ *bits* excluídas aquelas com nenhum ou apenas um *bit* 1, arranjadas como vetores linha. Por exemplo, em um código de Hamming (7, 4) sistemático, as combinações de $q = 7 - 4 = 3$ *bits* são [0 0 0], [0 0 1], [0 1 0], [0 1 1], [1 0 0], [1 0 1], [1 1 0] e [1 1 1]. Portanto, as combinações linearmente independentes são [0 1 1], [1 0 1], [1 1 0] e [1 1 1]. Arranjando-as na forma

matricial, tem-se a matriz geradora de um código de Hamming sistemático (7, 4),

$$\boldsymbol{G} = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right].$$

O código de Hamming (3, 1), com 2 *bits* de paridade, possui a matriz geradora $\boldsymbol{G}$ = [1 1 1] e é um caso de código de repetição.

A taxa de um código de Hamming é

$$R_c = \frac{2^q - q - 1}{2^q - 1} = 1 - \frac{q}{2^q - 1}.$$

3.4.4. Decodificação de Códigos de Bloco

Quando uma palavra código é enviada por um canal de comunicação, esta está sujeita a erros na recepção. É possível modelar uma comunicação com erro como

$$\boldsymbol{r} = \boldsymbol{c} + \boldsymbol{e}, \tag{3.14}$$

sendo $\boldsymbol{r}$ o vetor binário recebido e $\boldsymbol{e}$ o vetor de erro, representado por um vetor com n elementos com *bit* 1 na posição em que ocorreu erro. Por exemplo, se a palavra código $\boldsymbol{c}$ = 1101 é transmitida pelo canal e a palavra $\boldsymbol{r}$ = 1111 é recebida, o vetor de erro é 0010. Assim, ao decodificar a mensagem, o resultado será

$$\boldsymbol{r}\,\boldsymbol{H}^T = (\boldsymbol{c} + \boldsymbol{e})\,\boldsymbol{H}^T = \boldsymbol{c}\,\boldsymbol{H}^T + \boldsymbol{e}\,\boldsymbol{H}^T.$$

Como $\boldsymbol{c}$ é um código válido, $\boldsymbol{c}\,\boldsymbol{H}^T = \boldsymbol{0}$ e, portanto,

$$\boldsymbol{r}\,\boldsymbol{H}^T = \boldsymbol{e}\,\boldsymbol{H}^T = \boldsymbol{s}. \tag{3.15}$$

O vetor $\boldsymbol{s}$ resultante é chamado de síndrome e pode indicar em que posição ocorreu o erro. A síndrome é composta por k *bits*, enquanto o vetor de erro é composto por n *bits*. Portanto, uma síndrome pode mapear mais de um vetor de erro. Códigos de distância mínima igual a 3, como visto, são capazes de corrigir um erro, com isso, os decodificadores por síndrome mantém uma tabela de síndrome, mapeando os sinais de erro em apenas 1 *bit*. Por exemplo, a Tabela 3.2 mostra uma tabela de síndrome do código de Hamming (7, 4) exemplificado na seção 3.4.3.

Com a tabela de síndrome, conforme o resultado de $\boldsymbol{r}\,\boldsymbol{H}^T$, a palavra recebida é acrescida do erro referente à síndrome encontrada. Por exemplo, se $\boldsymbol{r}$ = 1111001, $\boldsymbol{s} = \boldsymbol{r}\,\boldsymbol{H}^T$ = 110, que é referente ao erro 0010000. Assim, a palavra código a ser decodificada é $\boldsymbol{r} + \boldsymbol{e}$ = 1101001, que corresponde à mensagem $\boldsymbol{m}$ = 1101 [3].

Tabela 3.2. Tabela de síndrome de um código de Hamming (7, 4) sistemático.

$\boldsymbol{e}$	$\boldsymbol{s}$
0000000	000
0000001	001
0000010	010
0000100	100
0001000	111
0010000	110
0100000	101
1000000	011

3.4.5. Códigos Cíclicos

Embora os códigos de bloco sistemáticos sejam de fácil compreensão e aplicação, os códigos cíclicos possuem uma estrutura matemática mais bem definida, permitindo melhores vantagens na codificação, decodificação e representação [3].

Um código de bloco é cíclico quando uma palavra código pode ser obtida pelo deslocamento cíclico de uma outra palavra código. Isto é, se $\boldsymbol{c}^{(1)} = (c_1, c_2, \ldots, c_n)$ é uma palavra código de $\boldsymbol{C}$, então existe uma palavra código $\boldsymbol{c}^{(2)} = (c_n, c_1, c_2, \ldots, c_{(n-1)})$ em $\boldsymbol{C}$, ou seja, um deslocamento cíclico de $\boldsymbol{c}^{(1)}$. Um exemplo de código cíclico (3, 2) é

$$\boldsymbol{C} = \{000, 011, 101, 110\}.$$

A matriz geradora deste código de exemplo é

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}.$$

É importante notar que o deslocamento pode ser tanto para a direita quanto para a esquerda, visto que, em uma palavra de n *bits*, um deslocamento para a direita equivale a $n - 1$ deslocamentos para a esquerda. Além disso, por ser um código linear, a mensagem formada apenas por *bits* 0 será codificada apenas por *bits* 0 e pode existir a palavra código formada apenas por *bits* 1, isto porque essas duas palavras são deslocamentos cíclicos delas próprias.

Uma das principais vantagens dos códigos cíclicos é sua representação por polinômios [3]. A representação polinomial de palavras binárias é feita com a posição do *bit* na palavra dada pelo grau do polinômio e o coeficiente é o *bit*. Por exemplo, a mensagem

$\boldsymbol{m} = (m_1, m_2, \ldots, m_k) = 1011001$ possui grau máximo igual a $k - 1 = 6$. O maior grau do polinômio é do MSB (*most significant bit*), que é o *bit* mais a esquerda, já o grau 0 é o do LSB (*least significant bit*), que é o *bit* mais a direita. A representação polinomial da mensagem do exemplo é

$$m(x) = 1 \cdot x^6 + 0 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$$

$$m(x) = x^6 + x^4 + x^3 + 1$$

Ou, de maneira geral,

$$m(x) = m_1 \cdot x^{(k-1)} + m_2 \cdot x^{(k-2)} + \ldots + m_k \cdot x^0. \tag{3.16}$$

A representação polinomial de codificações é bastante útil, principalmente em se tratando da geração de códigos cíclicos. Para encontrar uma forma de se obter códigos cíclicos, observe que ao multiplicar o polinômio de uma palavra código $\boldsymbol{c}$, $c(x)$, por x, todos seus elementos são deslocados para a esquerda. Para um código cíclico, seja $c(x)$ o polinômio de uma palavra código e $\tilde{c}(x)$ a palavra código obtida de um deslocamento cíclico para a esquerda de $c(x)$, então

$$c(x) = c_1 \cdot x^{(n-1)} + c_2 \cdot x^{(n-2)} + \ldots + c_i \cdot x^{(n-i)} + \ldots + c_n \cdot x^0$$

e $$\tilde{c}(x) = c_2 \cdot x^{(n-1)} + c_3 \cdot x^{(n-2)} + \ldots + c_i \cdot x^{(n-i+1)} + \ldots + c_n \cdot x^1 + c_1 \cdot x^0.$$

Reescrevendo $\tilde{c}(x)$ em função de $c(x)$,

$$\tilde{c}(x) = c(x) \cdot x - c_1 \cdot (x^n - 1). \tag{3.17}$$

No caso binário, sendo $B = \{0, 1\}$, $a + b = a - b$, com $a, b \in B$, ou seja, todas as operações de subtração podem ser substituídas por adição. Assim, a Equação (3.17) equivale a

$$\tilde{c}(x) = c(x) \cdot x + c_1 \cdot (x^n + 1). \tag{3.18}$$

Com essa notação é possível perceber que o resto da divisão polinomial de $\tilde{c}(x)$ por $(x^n + 1)$ é igual a $x \cdot c(x)$. Ou seja,

$$\tilde{c}(x) = x \cdot c(x) \pmod{x^n + 1}. \tag{3.19}$$

Então, pelo critério de divisibilidade polinomial, um código é cíclico se para todo $c(x) \in \boldsymbol{C}$ o resto da divisão polinomial de $x \cdot c(x)$ por $x^n + 1$ também é o polinômio de uma palavra código. Isso fornece uma outra forma de definir códigos cíclicos por meio de suas respectivas representações polinomiais.

De maneira análoga às matrizes geradoras do demais códigos de bloco lineares, um código cíclico pode ser obtido pelo seu polinômio gerador, $g(x)$, e suas palavras código podem ser verificadas pelo seu polinômio verificador, $h(x)$. Em [3], mostra-se que a relação entre

esses dois polinômios é dada por

$$g(x)h(x) = x^n + 1. \tag{3.20}$$

Além disso

$$c(x) = m(x)g(x). \tag{3.21}$$

Como o grau do polinômio $m(x)$ é $k - 1$, para que $c(x)$ tenha grau $n - 1$ é preciso que o grau do polinômio $g(x)$ seja $q = (n - 1) - (k - 1) = n - k$. Por meio da Equação (3.20), é preciso encontrar uma fatoração de $x^n + 1$ que contenha um polinômio de grau q a ser utilizado como polinômio gerador. Por exemplo, para gerar um código cíclico (7, 4), ou seja, códigos de tamanho $n = 7$ com mensagens de tamanho $k = 4$, é preciso fatorar o polinômio $x^7 + 1$ e encontrar um polinômio de grau $q = 7 - 4 = 3$. Seja a fatoração

$$x^7 - 1 = (x-1)(x^3 + x + 1)(x^3 + x^2 + 1).$$

Como as operações são binárias, trocam-se todas os subtrações por somas, de maneira que

$$x^7 + 1 = \underbrace{(x+1)}_{(i)}\underbrace{(x^3 + x + 1)}_{(ii)}\underbrace{(x^3 + x^2 + 1)}_{(iii)}.$$

Tanto os polinômios (*ii*) quanto (*iii*) possuem o grau necessário, portanto, ambos geram códigos cíclicos. Sendo assim, caso seja escolhido $g(x) = x^3 + x + 1$, por consequência, de acordo com a Equação (3.20), $h(x) = (x + 1)(x^3 + x^2 + 1)$. A Tabela 3.3 mostra a codificação da mensagens utilizando o polinômio gerador escolhido.

Tabela 3.3. Codificação cíclica (7, 4) com $g(x) = x^3 + x + 1$.

mensagem	**código**	**mensagem**	**código**
0000	0000000	1000	1011000
0001	0001011	1001	1010011
0010	0010110	1010	1001110
0011	0011101	1011	1000101
0100	0101100	1100	1110100
0101	0100111	1101	1111111
0110	0111010	1110	1100010
0111	0110001	1111	1101001

A fatoração de $x^7 + 1$ possui três polinômios, permitindo 2^3 combinações para o polinômio gerador, fazendo com que seja possível códigos cíclicos de várias taxas. Além

disso, é importante notar que nem todas as palavras códigos são rotações de uma mesma palavra código, caso contrário, haveria apenas n palavras códigos, portanto, isso só é possível quando $2^k \leq n + 1$ [3].

Assim como todos os códigos de bloco, os códigos cíclicos também podem ser representados por meio de uma matriz geradora. Sua construção não sistemática é dada por

$$\boldsymbol{G} = \begin{bmatrix} g_1 & g_2 & \cdots & \cdots & \cdots & \cdots & g_{(q-1)} & g_q & 0 & 0 & \cdots & 0 \\ 0 & g_1 & g_2 & \cdots & \cdots & \cdots & \cdots & g_{(q-1)} & g_q & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & g_1 & g_2 & \cdots & \cdots & \cdots & \cdots & g_{(q-1)} & g_q \end{bmatrix},$$

sendo g_i os coeficientes do polinômio gerador. Por exemplo, para $g(x) = x^3 + x + 1$, $\boldsymbol{g} = 1011$ e, portanto,

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Para decodificar códigos cíclicos, utilizando as Equações (3.20) e (3.21), tem-se que

$$c(x)h(x) = m(x)g(x)h(x) = m(x)(x^n + 1),$$

portanto,
$$c(x)h(x) = 0 \pmod{x^n + 1}. \tag{3.22}$$

Assim como nos outros códigos de bloco lineares já mencionados, sendo $r(x) = c(x) + e(x)$ o polinômio referente à palavra recebida com $e(x)$ o polinômio do erro,

$$r(x)h(x) = [c(x) + e(x)]h(x) = c(x)h(x) + e(x)h(x)$$

$$r(x)h(x) = e(x)h(x) \pmod{x^n + 1}. \tag{3.23}$$

Com isso, a síndrome de um código cíclico é $s(x) = e(x)h(x)$. Da mesma maneira, é construída uma tabela de síndromes para os casos de erro em um *bit*, utilizada para a correção de erro, e a palavra código decodificada é $c(x) = r(x) + e(x)$, com $e(x)$ estimado pela síndrome $s(x)$.

3.5. CÓDIGOS CONVOLUCIONAIS

Além dos códigos de bloco, os códigos lineares apresentam uma importante variação, os códigos convolucionais. Assim como qualquer código linear, códigos convolucionais são gerados por meio de combinações lineares dos *bits* mensagens. Contudo, além da mensagem atual, as combinações levam em conta as mensagens codificadas anteriormente. Com isso, uma mesma mensagem pode ser codificada por várias palavras código, a depender do que foi codificado previamente.

Um codificador convolucional mantém armazenados os L últimos *bits* codificados para realizar as combinações. O número de unidades de memória, L, adicionado à informação (*bit*) atual é chamando de comprimento de restrição, ou *constraint length* (K), sendo, portanto, $K = L + 1$, e representa o número de *bits* utilizados na codificação. Um exemplo de codificador convolucional com $K = 3$ é mostrado na Figura 3.3 e, neste caso, cada *bit* da mensagem dá origem a 2 *bits* do código, fazendo com que a taxa deste código seja igual a $R_c = k/n = 1/2$.

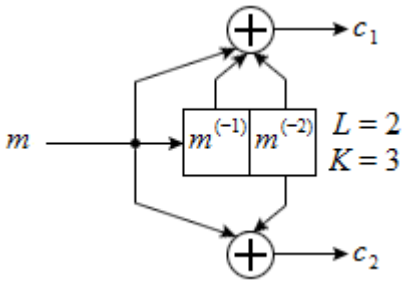


Figura 3.3. Codificador convolucional.

Para gerar códigos mais longos basta aumentar o número de combinações das memórias do codificador e, caso necessário, aumentar o número de memórias. Um codificador de taxa de código $R_c = 1/5$ com *constraint length* $K = 4$ é mostrado na Figura 3.4. Já para obter códigos com $k > 1$ é preciso combinar estruturas de codificadores. As mensagens são codificadas em blocos de k *bits* e, para cada *bit*, o codificador possui uma estrutura de memória, que se combinam gerando n *bits* do código, como mostra o exemplo de codificador convolucional de taxa $R_c = 2/3$ da Figura 3.5.

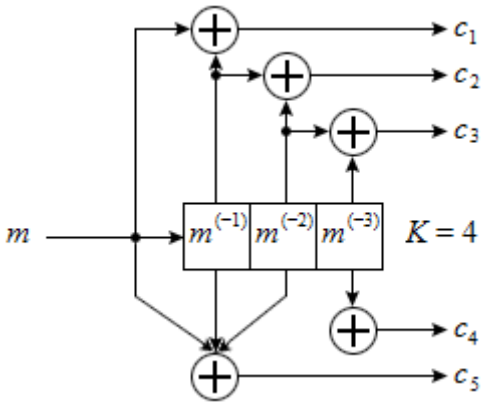


Figura 3.4. Codificador convolucional de taxa $R_c = 1/5$ e *constraint length* $K = 4$.

O processo de codificação é inicializado com as memórias limpas, isto é, preenchidas com zeros. Conforme as mensagens chegam no codificador, o código é gerado e as memórias são preenchidas com os novos *bits*, conforme mostra o processo da Figura 3.6 para o codificador apresentado na Figura 3.5.

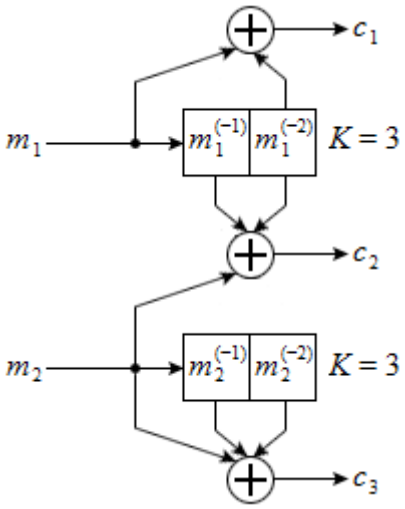


Figura 3.5. Codificador convolucional de taxa $R_c = 2/3$.

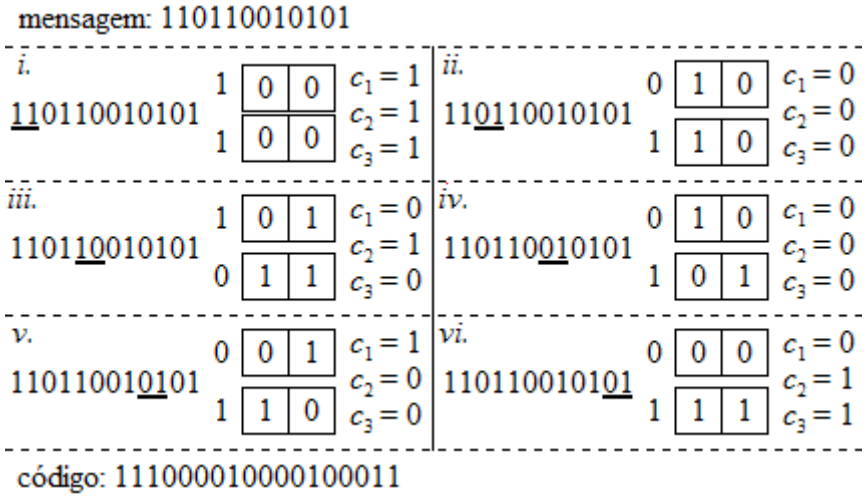


Figura 3.6. Processo de codificação convolucional com codificador da Figura 3.5.

Um codificador convolucional pode ser visto como uma máquina de estados, em que os estados são descritos pelos conteúdos das memórias e cada transição de estado ocorre com a chegada de um novo *bit*. Além disso, cada transição de estado gera uma saída, que é o código referente àquela combinação de *bits* da mensagem. A Figura 3.7 mostra a representação em máquina do codificador convolucional da Figura 3.3.

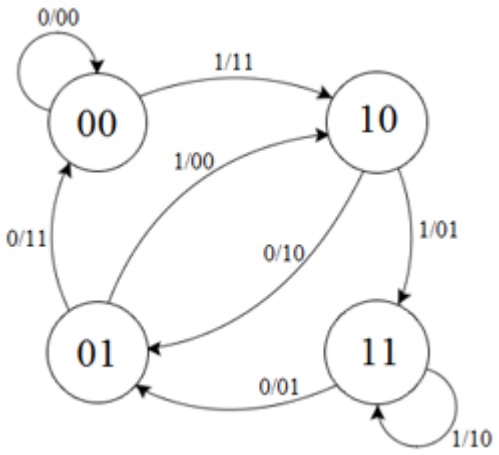


Figura 3.7. Representação do codificadores da Figura 3.3 como máquina de estados.

A notação da transição dos estados é $m_1m_2\ldots m_k/c_1c_2\ldots c_n$, sendo $m_1m_2\ldots m_k$ os *bits* da mensagem que chegam no codificador e $c_1c_2\ldots c_n$ o código gerado. Os estados são representados por $S_1S_2\ \ldots\ S_k$, sendo S_i o estado do codificador i, representando a forma como suas memórias estão ocupadas na ordem $m_i^{(-1)}m_i^{(-2)}\ldots m_i^{(-L)}$.

A representação em máquina de estado de um codificador convolucional pode ser tabelada, gerando uma tabela de codificação. A partir da tabela de codificação é possível representar o processo de codificação por meio de treliças. Seja o codificador convolucional da Figura 3.3, sua tabela de codificação é dada pela Tabela 3.4 e sua treliça, pela Figura 3.8.

Tabela 3.4. Tabela de codificação do codificador da Figura 3.3.

S	$\boldsymbol{m}$	S'	$\boldsymbol{c}$
00	0	00	00
	1	10	11
01	0	00	11
	1	10	00
10	0	01	10
	1	11	01
11	0	01	01
	1	11	10

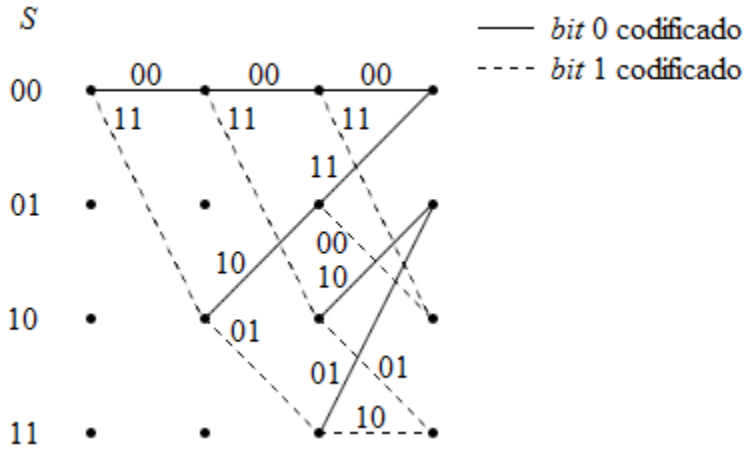


Figura 3.8. Treliça do codificador convolucional da Figura 2.3.

Além disso, os codificadores convolucionais podem ser representados por seus polinômios. Cada grau do polinômio representa uma unidade de atraso, deste modo, cada *bit* do código possui seu polinômio de grau máximo igual a K. Por exemplo, o polinômio do codificador da Figura 2.3 é $c_1 = 1 + x + x^2$ e $c_2 = 1 + x^2$. Para codificadores com $k > 1$ é

conveniente representar qual o *bit* da mensagem está sendo representado pela combinação dos *bits* em atraso. Por meio de subscritos, o polinômio que representa o codificador convolucional da Figura 3.5 é $c_1 = 1_1 + x_1^2$, $c_2 = x_1 + x_1^2 + 1_2$ e $c_3 = 1_2 + x_2 + x_2^2$.

3.5.1. Decodificador de Viterbi

Como visto nos códigos de bloco, a decodificação de códigos lineares era dada por meio da decodificação por síndrome. Nos códigos convolucionais, há um problema quanto a isso, pois não é possível representá-los por meio de estruturas matemáticas bem definidas, como matrizes ou polinômios geradores.

Para resolver o problema da decodificação de códigos convolucionais, vários algoritmos foram propostos, contudo, o mais utilizado é algoritmo proposto por Viterbi [6]. O algoritmo de Viterbi, como foi batizado, consiste em decodificar as sequências de códigos por meio de um estimador de máxima verossimilhança (MLSE – *maximum likelihood sequence estimator*). Basicamente, o algoritmo recebe os dados codificados e calcula a distância entre a sequência recebida e todas as possíveis sequências do codificador, tendo como saída aquela sequência de menor distância.

Como o algoritmo trata de um problema de estimação de máxima verossimilhança, assim como visto no capítulo 2, maximizar as probabilidades de acerto significa reduzir a distância entre os sinais estimado e recebido, sendo que por distância entende-se a distância euclidiana para sinais reais e distância de Hamming para sinais binários.

A distância entre a sequência recebida e a sequência mais provável de ter sido transmitida é calculada através da treliça do codificador. Cada passo do algoritmo calcula a distância entre o código recebido e os códigos de transição para cada estado da treliça. Sendo assim, a distância sempre aumentará a cada passo do algoritmo e essa quantia de aumento é chamada de métrica.

Para exemplificar, seja o codificador convolucional com polinômio $c_1 = 1 + x + x^2$ e $c_2 = 1 + x^2$ e suponha que vários passos do algoritmo já tenham sido executados, de maneira que até o momento a distância para os estados 00, 01, 10 e 11 seja 1, 2, 1 e 3, respectivamente. Neste momento, o decodificador recebe a palavra código 00. As possíveis transições são mostradas na Figura 3.8 e, a partir delas, cada métrica é calculada como a distância de Hamming entre a palavra recebida e o código gerado na transição de estado. Neste exemplo, a métrica entre o estado 01 e o estado 00 é d_H (00, 11) = 2, pois 00 foi recebido e o código gerado da transição de 00 para 01 é 11. A distância dos próximos estados é a soma da distância do estado anterior com a métrica responsável pela transição. No caso de

duas transições convergirem para um mesmo estado, como o algoritmo é um estimador de máxima verossimilhança, sobrevive o caminho com a menor distância acumulada [3]. A Figura 3.9 mostra esse exemplo.

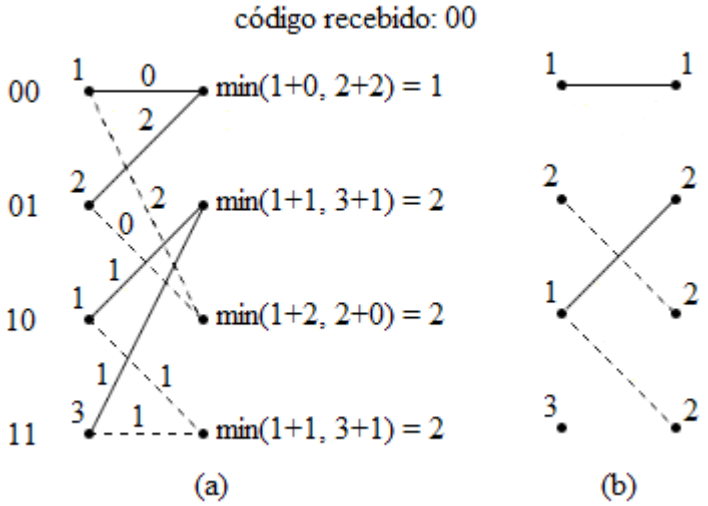


Figura 3.9. (a) Cálculo das métricas e distâncias e (b) caminhos prováveis.

Para exemplificar o esquema de decodificação convolucional por meio do algoritmo de Viterbi, considere ainda o mesmo codificador $\boldsymbol{c} = \{1 + x + x^2, 1 + x^2\}$. A mensagem codificada foi $\boldsymbol{m}$ = 110110010, gerando o código $\boldsymbol{c}$ = 110101000101111110. Ao ser transmitido pelo canal, o sinal demodulado apresentou dois erros, fazendo com que a palavra recebida fosse alterada nas posições 6 e 15, ou seja, $\boldsymbol{r}$ = 110100000101110110. Acompanhando o passo a passo do algoritmo neste caso, tem-se a treliça da Figura 3.10.

Para a Figura 3.10(a), assume-se que o codificador é iniciado com as memórias zeradas, gerando apenas duas transições de estado possíveis. A métrica acumulada dos estados é calculada e, então, calculam-se as distâncias para os estados e as novas métricas, como mostrado pela Figura 3.10(b).

Repetindo estes passos, o próximo passo do algoritmo é mostrado na Figura 3.10(c). É importante notar que não é necessário manter a informação acerca da distância de todas as etapas, mas apenas da última. Assim sendo, é possível que possíveis caminhos deixem de existir quando não direcionam para mais nenhum caminho. No terceiro passo é possível perceber que as transições do estado 10 para os estados 01 e 11 serão eliminadas, portanto, o caminho até aqui não tem necessidade de existir, no caso, a segunda transição do estado 00 para 10.

Estes passos são repetidos até os últimos *bits* recebidos. A treliça do decodificador é mostrada na Figura 3.10(d), decodificando o caminho com menor métrica final, como mostrado na Figura 3.10(e).

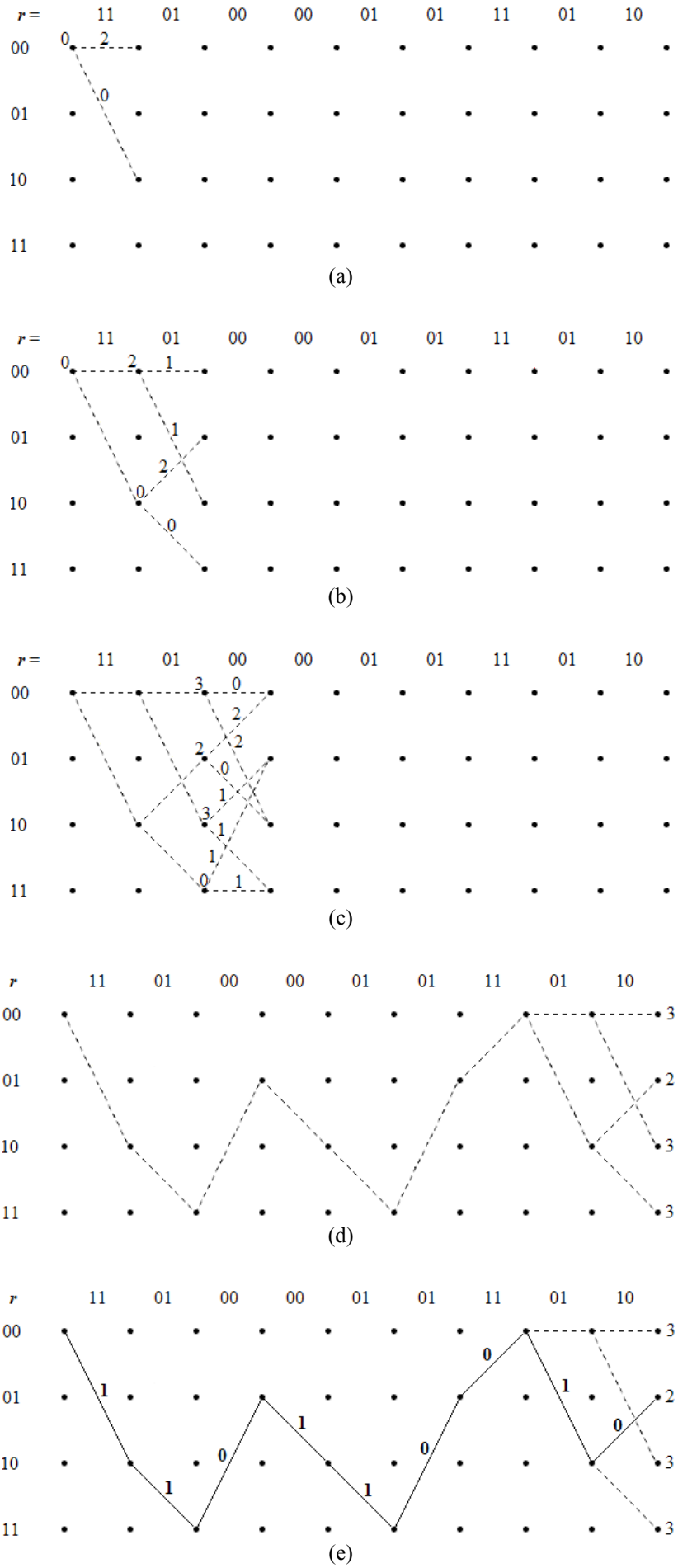


Figura 3.10. Passo a passo do algoritmo de Viterbi.

No final, a mensagem decodificada foi 110110010, que foi a mesma transmitida, ou seja, o decodificador foi capaz de corrigir os dois erros gerados durante a transmissão. Neste exemplo simples com uma mensagem limitada a 8 *bits,* o decodificador pode começar e finalizar a decodificação, mas numa situação mais real, não se sabe o início e o fim de uma comunicação.

O decodificador muitas vezes não pode esperar todos os dados serem recebidos para retornar a sequência decodificada, por isso existem algumas estratégias de implementação. Como visto no exemplo, muitas vezes no meio da decodificação boa parte da treliça está como um caminho único. Neste caso, o decodificador pode retornar os *bits* já decodificados desse caminho. Em outros casos, a comunicação acontece por blocos, caso em que o decodificador pode esperar o término de um bloco para retornar a mensagem decodificada. Nestes casos, para facilitar a implementação do algoritmo e seu desempenho, existem os códigos terminados.

Códigos terminados são mensagens preenchidas com zeros no final a fim de forçar que o codificador termine no estado inicial. Quando o decodificador opera com códigos terminados, a sequência decodificada é aquela que terminar no estado zero. Essa técnica evita problemas simples como, por exemplo, quando há empate entre duas distâncias ao final do algoritmo.

Para implementar o algoritmo de Viterbi como o exemplificado, é preciso realizar primeiramente a demodulação dos símbolos, transformando a entrada do decodificador em *bits,* este modo de operação do algoritmo é conhecido como decisão *hard*. Contudo, é possível que os sinais recebidos sejam diretamente decodificados por meio do algoritmo de Viterbi. Assim sendo, a sequência recebida assume um valor real, aumentando a informação acerca do canal de comunicação. Essa informação extra concede ao decodificador um desempenho melhor na correção de erros.

3.5.2. Decodificador de Viterbi de Decisão *Soft*

Nos casos estudados até aqui, todos os códigos foram decodificados por meio de decisão *hard,* isto é, após passar pelo filtro de recepção, o nível do sinal era convertido em *bits* de acordo com a modulação e só então a sequência de *bits* era decodificada. Esse tipo de tratamento dos sinais pode não ser o mais apropriado e, ainda, ser responsável por erros que poderiam ser corrigidos. Numa modulação BPSK, por exemplo, um sinal na saída do filtro de recepção muito próximo de zero não é um indicador muito confiável, mas poderia ser, caso os símbolos previa e futuramente transmitidos trouxessem alguma informação sobre ele [3].

Os passos do algoritmo de decisão *soft* são os mesmos, porém a métrica é calculada com

a distância euclidiana entre a sequência recebida e a sequência mais provável. Tomando como exemplo uma mensagem codificada convolucionalmente com taxa 1/2 e modulação BPSK de energia unitária, com o codificador $\boldsymbol{c} = \{1 + x + x^2, 1 + x^2\}$, a tabela de codificação *soft* é como dada pela Tabela 3.5.

Tabela 3.5. Tabela de codificação *soft* para a modulação BPSK.

S	$\boldsymbol{m}$	S'	$\boldsymbol{c}$
00	0	00	(+1, +1)
	1	10	(–1, –1)
01	0	00	(–1, –1)
	1	10	(+1, +1)
10	0	01	(–1, +1)
	1	11	(+1, –1)
11	0	01	(+1, –1)
	1	11	(–1, +1)

Com os dados da Tabela 3.5, caso, em um instante, as duas saídas consecutivas do filtro sejam 0,6 e –0,15, a métrica da transição do estado 10 para o estado 11 é, portanto, $\|c - r\|^2$ que, no caso, é $(1 - 0{,}6)^2 + (-1 - (-0{,}15))^2 = 0{,}8825$. Assim, o decodificador passa a ter mais informações sobre o sinal recebido, pois um número real pode conter mais informação que um número binário. Isso representa um ganho no desempenho do decodificador com relação a taxa de erro de *bit* de uma decisão *hard*, como é mostrado na Figura 3.11.

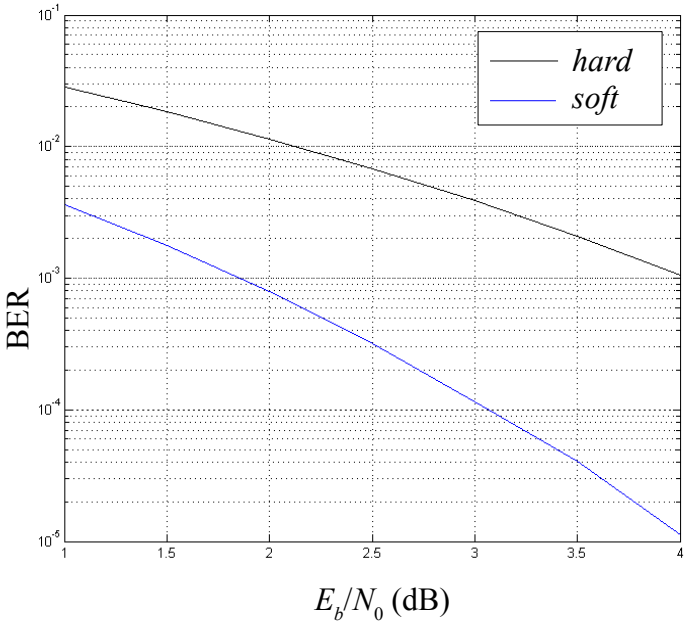


Figura 3.11. Comparação de desempenho *hard* e *soft*.

Porém, todas essas operações são realizadas por meio de equipamentos digitais, o que significa que não é exatamente o valor real da saída do filtro de recepção que é utilizado no cálculo da métrica, mas sim, uma quantização deste valor. A quantização da informação fará com que haja certa perda de informação acerca do sinal transmitido. Isso faz com que o desempenho do decodificador caia um pouco face à taxa de erro de bit, como é mostrado pela Figura 3.12. Ainda assim, o desempenho do decodificador *soft* quantizado é melhor que o desempenho do decodificador *hard.*

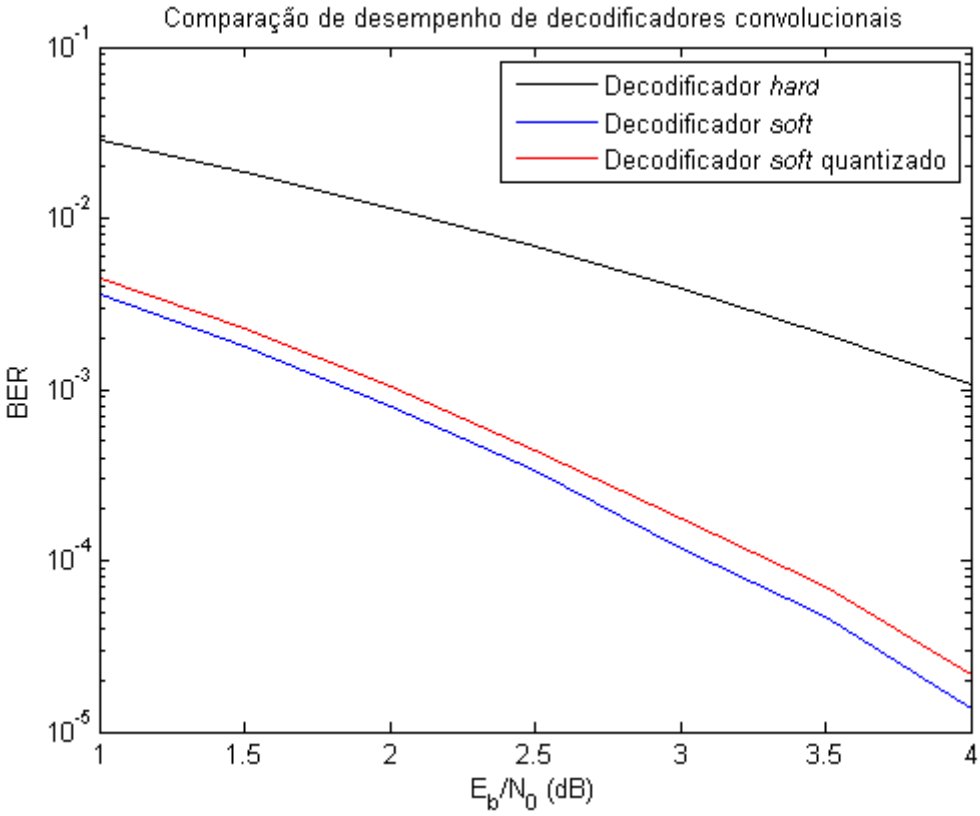


Figura 3.12. Comparação do desempenho dos decodificadores *hard*, *soft* e *soft* quantizado.

Assim como visto na seção 2.2, a quantização de uma informação gera erro, que é tão menor quanto maior for o número de níveis de quantização. Portanto, na construção de um decodificador de Viterbi de decisão *soft*, é de grande importância a definição dos níveis de quantização, tanto na quantidade de *bits* utilizados para representar os valores quantizados quanto no espaçamento entre esses níveis. Mostra-se que é mínima a diferença entre a quantização com 3 *bits* e uma quantização com infinitos níveis, que seria o caso analógico [7], resultando em uma perda de menos de 0,25 dB [3]. Além disso, o espaçamento entre os níveis pode variar de acordo com a relação sinal-ruído [4].

Mesmo com os níveis quantizados, o esforço computacional demandado para o cálculo da métrica seria grande. Para contornar o problema os decodificadores podem manter uma tabela de métricas, relacionando o sinal recebido quantizado com as possíveis transições. Por

exemplo, usando BPSK e 2 *bits* de quantização com o mesmo codificador no início da seção, existem 4 níveis de quantização e 16 combinações de recepção quantizadas, juntamente com 4 possíveis comparações, gerando uma tabela de 64 posições. Este número cresceria muito com o aumento de *bits* de quantização ou usando outra modulação, por isso, como a métrica é uma soma de duas diferenças quadráticas, basta guardar na tabela o valor das diferenças e somá-los. Com o mesmo esquema, existem apenas dois níveis de comparação, +1 e –1, e quatro níveis de recepção. A Tabela 3.6 mostra como ficaria a tabela utilizada caso os valores de quantização fossem –0,75, –0,25, +0,25 e +0,75.

Tabela 3.6. Tabela de métricas *soft*.

	00: +0,75	**01: +0,25**	**10: –0,25**	**11: –0,75**
+1	0,0625	0,5625	1,5625	3,0625
–1	3,0625	1,5625	0,5625	0,0625

Além disso, a tabela ainda é sempre simétrica, sendo possível armazenar apenas metade de seu conteúdo para uma implementação mais eficiente. Também é possível fazer operações lineares com os valores da tabela de modo que o número armazenado seja inteiro, por exemplo. Uma transformação por exemplo é $\tilde{x} = a(x+b)$ com $a = 2$ e $b = -0{,}0625$, gerando os valores da Tabela 3.7.

Tabela 3.7. Tabela de métricas soft normalizada.

	00: +0,75	**01: +0,25**	**10: –0,25**	**11: –0,75**
+1	0	1	3	6
–1	6	3	1	0

Neste caso, *b* foi escolhido de tal modo que o menor valor fosse zero e *a* pôde ser facilmente encontrado em seguida. Portanto, caso a saída do filtro de recepção fosse quantizada em 01 00 (2 *bits* de quantização vezes o número de *bits* da palavra código) a métrica do estado 10 para o estado 00 seria 1 + 6 = 7, pois o código desta transição é (+1, –1) e, utilizando a Tabela 3.7, obtém-se o mapeamento de 01 com +1 e 00 com –1.

O diagrama de blocos dos três tipos de quantizadores, isto é, de decisão *hard*, de decisão *soft* e *soft* quantizado com 3 *bits*, é mostrado na Figura 3.13.

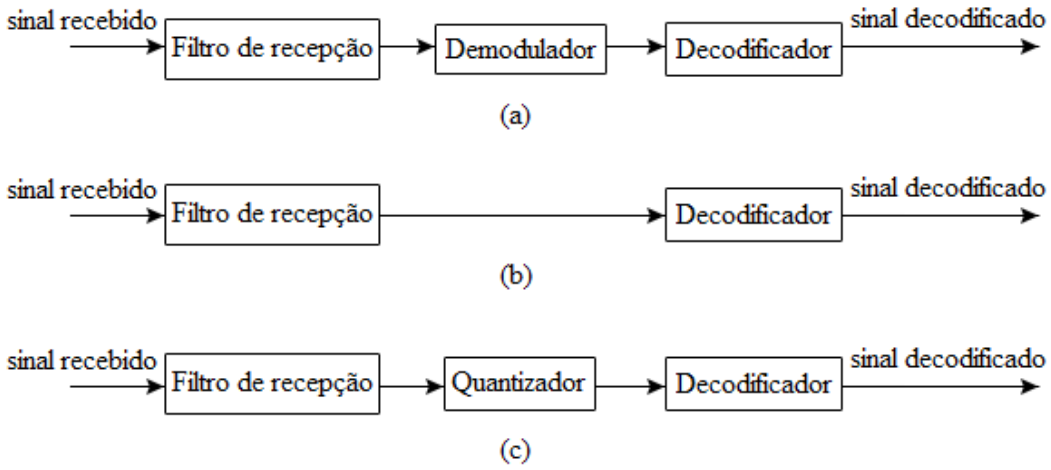


Figura 3.13. Diagrama dos decodificadores (a) *hard*, (b) *soft* e (c) *soft* quantizado.

3.5.3. Renormalizações

Além da limitação de *bits* para representar o sinal quantizado nos decodificadores de decisão *soft*, implementações em *hardware* dos decodificadores contam com a limitação de *bits* para guardar as distâncias. Caso o decodificador tenha uma memória muito limitada e tenha que realizar diversas iterações do algoritmo de Viterbi, a depender do nível da comunicação pode ser que a soma constante de métricas elevadas levem a um *overflow* no armazenamento das distâncias [26].

Para contornar possíveis problemas de *overflow*, existem técnicas de renormalização das distâncias. É facilmente verificado que subtrair o mesmo valor de todas as distâncias não altera o funcionamento do decodificador. Sendo assim, uma técnica simples de renormalização consiste em subtrair das distâncias o valor da menor distância a cada número determinado de iterações. O lado negativo dessa medida é que isso aumenta o processamento necessário, reduzindo o desempenho do decodificador, sendo que quanto mais constante for o número de renormalizações, mais degradado fica o sistema.

Outra técnica é a renormalização por limiar, sendo que o valor da menor distância é subtraído de todas as distâncias quando alguma dessas distâncias atingir um valor limite. O desempenho desta técnica com relação a anterior é melhor, pois evita renormalizações desnecessárias, porém, há uma perda de desempenho advinda da necessidade de monitorar constantemente o valor das distâncias.

Existem ainda diversas outras técnicas de renormalização, como pode ser visto em [26]. Técnicas que permitem renormalizações apenas quando necessário são importantes pois o número de renormalizações pode ser um indicativo do desempenho do decodificador, visto que um codificador *soft* com níveis de quantização mal estipulados pode gerar métricas elevadas desnecessariamente, aumentando a possibilidade de *overflow*.

3.6. CONCLUSÃO

As técnicas de codificação de canal permitem grandes ganhos nas comunicações digitais, diminuindo a probabilidade de erro de *bit* e, com isso, evitando retransmissões a fim de garantir comunicações confiáveis com altas taxas de dados, como pode ser verificado pelo teorema de Shannon. As diversas técnicas de codificação existentes dão imensa liberdade aos projetistas de sistemas de comunicação quanto a forma de combater possíveis erros, não existindo nada consolidado como um esquema mais adequado para todos os sistemas.

Dentre as diversas maneiras de se decodificar um código convolucional, destaca-se o método de decisão *soft*, capaz de garantir taxas de erro de *bit* ainda menores quando comparado a um mesmo decodificador com decisão *hard* [3]. Contudo, a necessidade de quantização do sinal causa perdas no desempenho, que podem variar com a escolha dos níveis de quantização.

Comumente, a quantização na decisão *soft* ocorre de maneira arbitrária. Em geral utilizam-se níveis igualmente espaçados [4]. No próximo capítulo, serão vistas técnicas de aprendizado de máquina, que podem ser aplicadas nos quantizadores a fim de se obter um esquema de quantização dinâmico e que, possivelmente, resulte na menor taxa de erro de *bit.*

4. APRENDIZADO POR REFORÇO

4.1. INTRODUÇÃO

A computação pode ser descrita como a tarefa de processar dados, isto é, obter saídas em função de entradas, a fim de solucionar problemas. Para isto, quando a resolução de um problema é bem definida, podendo ser descrita ou explicada, surgem os algoritmos, que são conjuntos de instruções sequenciais bem definidas e que podem ser seguidas passo a passo de modo finito. Algoritmos não são uma exclusividade da computação, uma receita de bolo, por exemplo, pode ser definida como um algoritmo. Usando este mesmo exemplo, é possível fazer uma analogia com uma tarefa computacional. As entradas do sistema seriam os ingredientes do bolo, ovos, farinha, leite, etc. A saída do sistema é o bolo. Para processar a entrada e obter a saída, basta seguir o algoritmo, isto é, seguir a receita.

A partir deste pensamento, surge um primeiro problema, que é a motivação para se utilizar aprendizado de máquina: e se não for possível descrever como as entradas devem ser processadas? Por exemplo, como descrever em passos o reconhecimento de um carro esportivo? A resposta para estas perguntas é simples, é preciso ensinar a máquina a realizar suas tarefas, sejam problemas de caracterização ou não.

Ensinar uma máquina a realizar tarefas nada mais é do que fornecer a ela um conjunto de amostras e guiar seu aprendizado ou fazer com que ela aprenda através de estímulos em um ambiente. Existem vários tipos de aprendizado de máquina, que são classificados como aprendizado supervisionado e não-supervisionado.

a. **Aprendizado supervisionado**: Neste tipo de aprendizado, juntamente com as amostras, são fornecidas as saídas corretas (ou que a máquina espera que sejam corretas, visto que há a possibilidade de haver erros no ensinamento) com as quais a máquina será treinada. Diz-se, neste caso, que há uma espécie de professor. O objetivo desse método é reduzir o erro entre as amostras corretas e as saídas geradas pela máquina.

b. **Aprendizado não-supervisionado**: Diferentemente do aprendizado supervisionado, neste tipo de aprendizado, a máquina não sabe o que seria o correto a se fazer. Em vez disso, é sua função encontrar padrões nas amostras ou utilizar as respostas de suas ações para guiar seu aprendizado.

Dentro do aprendizado não-supervisionado, encontram-se as técnicas de aprendizado por reforço. Neste tipo de aprendizado, no lugar de um professor, como no aprendizado supervisionado, existe a figura de um crítico, que muitas vezes é o próprio ambiente,

responsável por julgar as ações da máquina e fornecer-lhe informações, hora chamadas de recompensas.

Neste capítulo serão introduzidos os conceitos básicos de aprendizado por reforço para casos discretos, apresentando alguns problemas e a forma de resolvê-los por meio das técnicas apresentadas.

4.2. CONCEITOS BÁSICOS

O aprendizado por reforço é uma técnica de aprendizado de máquina em que o agente de aprendizado não é informado sobre quais ações deve tomar, mas deve aprender a tomar ações por meio de experiências anteriores [27]. Este método de aprendizado se assemelha bastante com o aprendizado animal natural. Um exemplo de analogia é o adestramento de cães, em que um cão é ensinado a realizar tarefas por meio de recompensas (ou punições).

Sendo assim, um modelo simplificado de aprendizado por reforço consiste em um agente interagindo com o ambiente, isto é, praticando ações, e recebendo recompensas por isso. O objetivo de um agente é, portanto, tomar ações que maximizem a recompensa por ele obtida, tornando o sistema de aprendizado um problema de tentativa e erro [27].

Além do agente e suas ações, do ambiente e das recompensas, existem ainda outros elementos que compõem um sistema de aprendizado por reforço. O ambiente é descrito por um conjunto de estados $\boldsymbol{S}$, em que o agente pode tomar as ações, sendo que cada estado possui um conjunto de ações possíveis $\boldsymbol{A}(s)$. O comportamento do agente, isto é, quais ações serão tomadas nos estados é descrito pela política. Uma política $\pi(s, a)$ representa a probabilidade de tomar a ação a no estado s.

O agente interage com o ambiente em tempos discretos, $t = 1, 2, 3, \ldots$. Em cada instante, o agente se encontra em um estado $s_t \in \boldsymbol{S}$ e pratica uma ação $a_t \in \boldsymbol{A}(s_t)$ de acordo com a política π atual. Como consequência, o agente transita para um estado s_{t+1} e recebe uma recompensa r_{t+1}. A notação r_{t+1} é utilizada devido ao fato de que a recompensa é posterior à ação. Um modelo simplificado de aprendizado por reforço é mostrado na Figura 4.1.

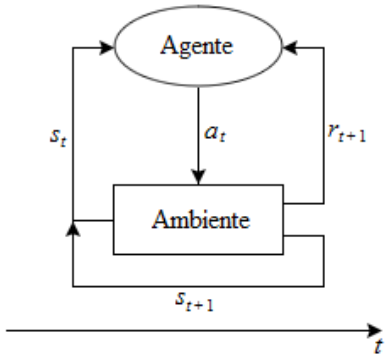


Figura 4.1. Modelo de aprendizado por reforço.

O termo recompensa é bastante utilizado na literatura, embora o termo mais apropriado fosse sinal de reforço, pois a palavra recompensa passa a ideia de algo positivo para o agente. Em alguns casos, quando o sinal de reforço é positivo, este é chamado de recompensa e quando é negativo é chamado de punição. Entretanto, neste texto será utilizado o termo recompensa para representar o sinal de reforço.

As recompensas recebidas pelo agente podem ser determinísticas ou estocásticas, sendo o objetivo do agente maximizar a recompensa acumulada durante o processo de aprendizado. Quando um agente finaliza uma tarefa é dito que se passou um episódio, por exemplo, se a tarefa de um agente é encontrar a saída de um labirinto, se passa um episódio de duração T quando o agente de fato encontra a saída do labirinto após T instantes. O processo de aprendizado pode ser finito ou infinito, no que se refere ao número de instantes necessários para finalizar a tarefa. Num jogo da velha, por exemplo, cada jogador joga, no máximo, quatro vezes. Se a tarefa do agente é aprender a joga jogo da velha, diz-se que o problema tem um horizonte finito. Caso contrário, se o agente executa a tarefa de aprendizado continuamente, diz-se que o problema tem um horizonte infinito.

A recompensa acumulada a partir de um instante t é simplesmente a soma das recompensas futuras recebidas ao longo do tempo,

$$R_t = r_{t+1} + r_{t+2} + ... + r_T. \tag{4.1}$$

Pode ser que seja inconveniente somar infinitamente todas as recompensas ao longo do tempo, por isso, recompensas futuras podem ser descontadas por um fator γ, assim,

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{4.2}$$

sendo $0 \leq \gamma \leq 1$. Caso $\gamma = 0$, o agente é dito míope, pois leva em conta apenas a recompensa imediata. Quanto mais próximo de 1, mais informações futuras são levadas em conta e no caso que $\gamma = 1$ tem-se a recompensa para horizontes finitos.

Quando o processo de aprendizagem por reforço é também um processo Markoviano, isto é, a recompensa e a transição de estado depende apenas do estado e ação atuais, ou seja

$$\begin{aligned} P^a_{ss'} &= Pr\{s_{t+1} = s' \mid s_t, a_t, s_{t-1}, a_{t-1}, s_{t-2}, a_{t-2}, ...\} \\ &= Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}, \end{aligned} \tag{4.3}$$

com

$$\sum_{s'} P^a_{ss'} = 1,$$

este é chamado de um processo de decisão de Markov (*Markov decision process,* MDP).

De maneira análoga, em um MDP, dado o estado atual s, a ação praticada a e o próximo estado observado s', o valor esperado para a recompensa é

$$R_{ss'}^{a} = Pr\{r_{t+1} \mid s_t = s, a_t = a, s_{t+1} = s'\}. \tag{4.4}$$

4.2.1. Funções de Valor

As funções de valor em uma tarefa de aprendizado por reforço definem para o agente o quão bom é estar em um estado (ou praticar uma ação no estado), no sentido de recompensa acumulada esperada, visto que o melhor para o agente é a expectativa de maior recompensa.

Lembrando que uma política π define os passos do agente no ambiente, o valor de um estado, $V^{\pi}(s)$, é definido por sua política como sendo o valor esperado da recompensa acumulada a partir daquele estado seguindo a política π, isto é

$$V^{\pi}(s) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s\right\}, \tag{4.5}$$

em que $E_{\pi}\{.\}$ é a esperança seguindo a política π. Similarmente, as ações da política também podem ser mensuradas pela função de valor ao tomar a ação a no estado s e seguir a politica π como

$$Q^{\pi}(s, a) = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a\right\}. \tag{4.6}$$

Por exemplo, um agente tem a tarefa de encontrar a saída de um labirinto com o menor número de instantes possíveis. O labirinto é modelado como um tabuleiro de 9×9 posições, em que cada coordenada representa um estado e as ações possíveis em um estado são aquelas que não colidem com as paredes do labirinto. Este modelo é representado na Figura 4.2, em que a linha vermelha indica a política atual do agente. A coordenada de início é H1 e a coordenada da saída é E9. Para forçar o agente a sair com a menor quantidade de movimentos, as recompensas podem ser dadas com o valor de –1 para qualquer passo (ação) do agente e $\gamma = 1$. Usando as Equações (4.5) e (4.6), o valor do estado (coordenada) C6 para a política indicada é –9, pois o agente levará 9 passos para sair do labirinto seguindo essa política. Já o valor da ação "ir para o norte" no estado C6 é –11, pois tomando esta ação o agente irá para o estado B6 e depois seguirá a política, completando o labirinto em mais 10 passos, 11 no total.

O estado final de uma tarefa, isto é, que indica o término de um episódio, é chamado de estado terminal. Como explicado no exemplo do labirinto, o estado terminal é a coordenada E9. Além disso, o valor de um estado terminal é sempre zero.

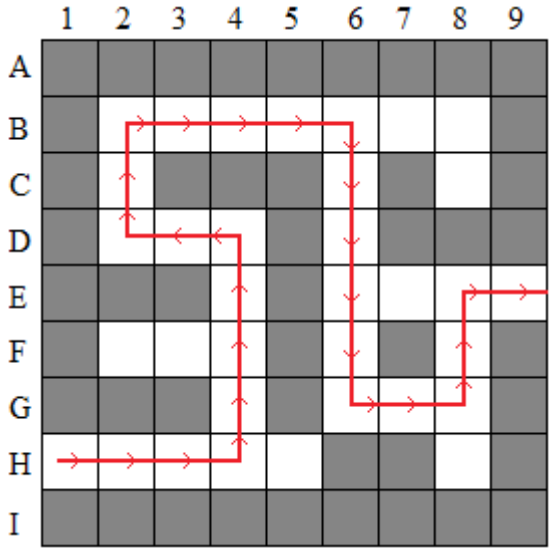


Figura 4.2. Problema do labirinto.

Uma propriedade fundamental das funções de valor é a recursividade. O valor de um estado pode ser calculado em função de seus estados sucessivos. Como é mostrado em [27],

$$V^{\pi}(s) = E_{\pi}\left\{ r_{t+1} + \gamma V^{\gamma}(s_{t+1}) \,\middle|\, s_t = s \right\}.$$

A *Equação de Bellman* representa tal propriedade como

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} [R^{a}_{ss'} + \gamma V^{\pi}(s')]. \tag{4.7}$$

4.2.2. Políticas Ótimas

Uma política ótima, π^*, é aquela que maximiza as funções de valores dos estados. Portanto, o objetivos dos algoritmos de aprendizado por reforço é encontrar a política ótima para a solução das tarefas. No exemplo do labirinto da Figura 4.2, a política indicada não é ótima, pois não fornece a maior recompensa acumulada, como pode ser verificado no movimento da coordenada E6.

Uma política ótima resulta em valores dos estados ótimos, V^*, que são os máximos valores de recompensa acumulada possíveis para cada estado do ambiente. O valor de estado ótimo é definido como

$$V^{*}(s) = \max_{\pi} V^{\pi}(s), \tag{4.8}$$

para todo $s \in \boldsymbol{S}$. Do mesmo modo, a função de valor de ação ótima, Q^*, que indica os maiores valores possíveis de recompensa acumulada ao praticar as ações no ambiente é

$$Q^{*}(s,a) = \max_{\pi} Q^{\pi}(s,a), \tag{4.9}$$

para todo $s \in \boldsymbol{S}$ e $a \in \boldsymbol{A}(s)$. Aplicando a *Equação de Bellman* nas Equações (4.8) e (4.9), chega-

se nas *Equações de Bellman de Otimalidade*, como é demonstrado em [27],

$$V^*(s) = \max_{a \in \boldsymbol{A}(s)} \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^*(s') \right] \tag{4.10}$$

e

$$Q^*(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma \max_{a'} Q^*(s',a') \right]. \tag{4.11}$$

Para MDPs finitos, isto é, quando os estados, ações e recompensas são finitos, a Equação (4.10) possui solução, sendo um sistema de equações de $|\boldsymbol{S}|$ equações e $|\boldsymbol{S}|$ incógnitas, em que $|\boldsymbol{S}|$ é o número total de estados. O problema em encontrar valores ótimos pelas equações de Bellman é a necessidade do modelo completo do ambiente, isto é, o conhecimento de todos os estados, ações, transições e recompensas do ambiente, por meio de suas probabilidades de transição, Equação (4.3), e recompensa, Equação (4.4), de um MDP finito.

4.3. PROGRAMAÇÃO DINÂMICA

Programação dinâmica (PD) consiste nas técnicas de aprendizado por reforço baseadas na solução das equações de otimalidade de Bellman. Portanto, é necessário um modelo completo do ambiente. A hipótese de um ambiente completamente modelado torna o uso dos algoritmos de programação dinâmica muito limitado. Porém, esses métodos provém uma base teórica importante para o entendimento de outras técnicas mais eficientes, como o método das diferenças temporais.

A ideia central dos algoritmos de PD é utilizar os valores de estado calculados pelas equações de otimalidade de Bellman para encontrar boas, ou talvez ótimas, políticas.

4.3.1. Avaliação de Política

Primeiramente, antes de iniciar a busca por boas políticas, é preciso encontrar o valor dos estados para uma determinada política. Essa fase é conhecida como avaliação de política e utiliza a Equação de Bellman como base.

Inicialmente, definida um política π, todos os valores dos estados são definidos arbitrariamente, com exceção do estado terminal, que deve ter valor igual a zero. Por exemplo, iniciando com zeros os valores dos estados tem-se $V_0(s) = 0$, $\forall\, s \in \boldsymbol{S}$. A partir dessa primeira estimativa, calcula-se pela Equação (4.7) uma próxima aproximação, sendo que os valores das transições são os valores da estimativa anterior, isto é,

$$V_1(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_0(s') \right],$$

para todo $s \in \boldsymbol{S}$. Depois, calcula-se uma nova avaliação dos valores, V_2, utilizando os

resultados de V_1. O procedimento é repetido inúmeras vezes, obtendo-se V_0, V_1, V_2, … por meio de

$$V_{k+1}(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V_k(s') \right], \tag{4.12}$$

para todo $s \in \boldsymbol{S}$. Por meio de simulações, pode ser mostrado que $V_k(s) \rightarrow V^\pi$ conforme $k \rightarrow \infty$, conforme [27]. Este método de avaliações sucessivas é chamado de avaliação iterativa de política e é mais vantajoso computacionalmente do que calcular as soluções das equações de Bellman [27]. Em termos de algoritmos, a avaliação iterativa de política deve ser executada até que a diferença entre todos os novos valores calculados e os valores antigos seja menor que um dado limiar δ, indicando a convergência dos valores, isto é, $V_k(s)$ é recalculado para todo $s \in \boldsymbol{S}$ até que

$$\max_s \left| V_k(s) - V_{k-1}(s) \right| < \delta, \quad \forall\, s \in \boldsymbol{S}.$$

4.3.2. Melhoria de Política

Uma vez encontrados os valores de uma política, é possível fazer alterações nesta, de modo a buscar uma melhoria, isto é, encontrar ações que levam a maiores valores de estado. Assim, para algum estado s, busca-se por alguma ação $a \neq \pi(s)$ que leve a um valor de estado maior que o atual. Por meio da avaliação de política, sabe-se quão bom é estar em um estado seguindo a política π. Da mesma maneira, é possível encontrar o quão bom é tomar uma ação em determinado e depois seguir a política. Usando o conceito de valor da ação de um estado, dado pela Equação (4.6), na Equação (4.7), obtém-se

$$Q^\pi(s,a) = \sum_{s'} P^a_{ss'} \left[R^a_{ss'} + \gamma V^\pi(s') \right]. \tag{4.13}$$

O critério de melhoria é, então, encontrar uma ação a para o estado s de modo que

$$Q^\pi(s,a) \geq V^\pi(s). \tag{4.14}$$

Uma vez encontrada uma ação a que satisfaça a Equação (3.14) para todo estado s, a política π é modificada para tomar essas novas ações, gerando uma política alterada π'. No exemplo do labirinto da Figura 4.2, o valor da ação "ir para a direita" no estado E6 é maior que o valor do estado F7, indicando uma melhoria de política. Para demonstrar o resultado final, aplicando a recursão na Equação (4.14), tem-se

$$\begin{aligned} V^\pi(s) &\leq Q^\pi(s, \pi'(s)) \\ &= E_{\pi'} \left\{ r_{t+1} + \gamma V^\pi(s_{t+1}) \,\middle|\, s_t = s \right\}. \end{aligned}$$

Novamente utilizando a propriedade da Equação (4.14) no último termo encontrado,

reescreve-se a desigualdade em termos dos valores das ações, Q, obtendo

$$\begin{aligned} V^{\pi}(s) &\leq E_{\pi'}\left\{r_{t+1}+\gamma Q^{\pi}(s_{t+1},\pi'(s_{t+1})) \,\middle|\, s_t=s\right\} \\ &= E_{\pi'}\left\{r_{t+1}+\gamma E_{\pi'}\left\{r_{t+2}+\gamma V^{\pi}(s_{t+2})\right\} \,\middle|\, s_t=s\right\} \\ &= E_{\pi'}\left\{r_{t+1}+\gamma r_{t+2}+\gamma^2 V^{\pi}(s_{t+2}) \,\middle|\, s_t=s\right\}. \end{aligned}$$

Estes passos sucessivos de recursão resultam em

$$\begin{aligned} V^{\pi}(s) &\leq E_{\pi'}\left\{r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\gamma^3 V^{\pi}(s_{t+3}) \,\middle|\, s_t=s\right\} \\ &\vdots \\ &\leq E_{\pi'}\left\{r_{t+1}+\gamma r_{t+2}+\gamma^2 r_{t+3}+\gamma^3 r_{t+4}+\ldots \,\middle|\, s_t=s\right\} \\ &= V^{\pi'}(s). \end{aligned}$$

Que demonstra que a nova escolha das ações resultou na melhoria da política, pois

$$V^{\pi'}(s) \geq V^{\pi}(s). \tag{4.14}$$

Encontrar ações que maximizam o valor dos estados resulta na busca por uma política gulosa. Uma política gulosa toma sempre a ação que resulta no maior valor de ação do estado e é definida como

$$\begin{aligned} \pi'(s) &= \arg\max_a \; Q^{\pi}(s,a) \\ &= \arg\max_a \; \sum_{s'} P^a_{ss'}\left[R^a_{ss'}+\gamma V^{\pi}(s')\right], \end{aligned} \tag{4.15}$$

Sendo arg max o argumento a que faz com que a função seja maximizada.

Supondo, então, uma política gulosa, π', obtida através de uma política π, de tal modo que $V^{\pi'} = V^{\pi}$. Então, para todo $s \in \boldsymbol{S}$,

$$\begin{aligned} V^{\pi'} &= \max_a \; E\left\{r_{t+1}+\gamma V^{\pi'}(s_{t+1}) \,\middle|\, s_t=a, a_t=a\right\} \\ &= \max_a \; \sum_{s'} P^a_{ss'}\left[R^a_{ss'}+\gamma V^{\pi'}(s')\right], \end{aligned}$$

que é o mesmo resultado da Equação (4.10), ou seja, a equação de otimalidade de Bellman. Assim, $V^{\pi} = V^{\pi'} = V^*$ e, consequentemente, $\pi = \pi' = \pi^*$.

Sendo assim, por meio de sucessivas melhorias na política é possível chegar na política ótima. Porém, a princípio, só é possível realizar a melhoria em uma política conhecendo-se todos os seus valores. A solução de problemas de aprendizado por reforço por meio de programação dinâmica consiste em encontrar formas de aliar etas duas ferramentas, avaliação e melhoria de política, de maneira a encontrar políticas ótimas. Basicamente, existem dois tipos de solução para esses problemas, uma é por meio dos algoritmos de iteração de política, outra é por meio dos algoritmos de iteração de valor. No fundo, ambos possuem o mesmo

princípio de funcionamento, mas cada um pode vir a se adequar diferentemente dependendo do modelo do ambiente, como número de estados, fator de desconto e recompensa [28].

4.3.3. Iteração de Política

Como visto anteriormente, para chegar em uma política ótima por meio da avaliação de política e melhoria de política são necessárias várias repetições destes passos. Primeiramente, inicia-se arbitrariamente uma política π^0, executa-se a avaliação de política para obter V^{π^0}. Depois, melhora-se a política, encontrado π^1. Novamente, executa-se a avaliação da nova política. Estes passos se repetem até que não seja mais possível melhorar a política, indicando que a política é ótima. Ou seja,

$$\pi^0 \xrightarrow{A} V^{\pi^0} \xrightarrow{M} \pi^1 \xrightarrow{A} V^{\pi^1} \xrightarrow{M} \pi^2 \xrightarrow{A} \ldots \xrightarrow{M} \pi^* \xrightarrow{A} V^*,$$

em que $\xrightarrow{A}$ indica a avaliação de política e $\xrightarrow{M}$ indica a melhoria de política. Este modo de encontrar políticas ótimas por meio de programação dinâmica é conhecido como iteração de política. O Algoritmo 4.1 mostra o pseudocódigo para a resolução de problemas de aprendizado por reforço com programação dinâmica por meio da iteração de política.

```
1. para todo s ∈ S faça
2.    inicialize V(s) e π(s) arbitrariamente
3. fim
4. faça
5.     Δ ← 0
6.    para todo s ∈ S faça
7.        v ← V(s)
8.        V(s) ← Σ_a π(s,a) Σ_s' P^a_ss' [R^a_ss' + γ V(s')]
9.        Δ ← max(Δ, |v − V(s)|)
10.   fim
11. enquanto Δ > δ
12. politica_instavel ← falso
13. para todo s ∈ S faça
14.    b ← π(s)
15.    π(s) ← arg max_a Σ_s' P^a_ss' [R^a_ss' + γ V^π(s')]
16.    se b ≠ π(s)
17.       politica_instavel ← verdadeiro
18.    fim
19. fim
20. se politica_instavel
21.    volta para o passo 4.
22. fim
```

Algoritmo 4.1. Iteração de política.

4.3.4. Iteração de Valor

Na iteração de política, a procura por ações que maximizem o valor dos estados é feita após a nova avaliação de política encontrada. Isso pode levar a um aumento considerável no

número de iterações necessárias para a convergência da política, embora o algoritmo de iteração de política convirja rápido, conforme [27]. Porém, existem meios de reorganizar os passos do algoritmo, juntando a iteração de valor com a avaliação de política. Um dos modos de se fazer isso é modificando a equação da avaliação de valor como

$$V_{k+1}(s) = \max_a \sum_{s'} P^a_{ss'}\left[R^a_{ss'} + \gamma V_k(s')\right]. \tag{4.16}$$

A partir da Equação (4.16), seguindo os mesmos passos da avaliação de valor, chega-se na política ótima. Este método de solução é conhecido por iteração de valor e seu algoritmo é descrito pelo pseudocódigo do Algoritmo 4.2.

```
1. para todo s ∈ S faça
2.    inicialize V(s) e π(s) arbitrariamente
3. fim
4. faça
5.     Δ ← 0
6.    para todo s ∈ S faça
7.        v ← V(s)
8.        V(s) ← max_a Σ_s' P^a_ss' [R^a_ss' + γ V(s')]
9.        Δ ← max(Δ, |v − V(s)|)
10.   fim
11. enquanto Δ > δ
```

Algoritmo 4.2. Iteração de valor.

4.4. MÉTODOS DE MONTE CARLO

Os métodos de Monte Carlo (MC) representam uma forma inicial de solucionar problemas de aprendizado por reforço sem a necessidade de um modelo completo do ambiente, como ocorria nos métodos de PD. Ao invés de calcular os valores dos estados por meio do conhecimento do ambiente, os métodos de MC se baseiam nas experiências anteriores para estimar os valores.

O aprendizado de Monte Carlo é baseado numa estimativa de episódio por episódio. Assim como nos métodos de PD, a procura pela política ótima se dá por meio de avaliações e melhorias das políticas, com a diferença que todos os valores são estimativas.

4.4.1. Avaliação de Política

Há duas formas de se calcular a estimativa dos valores baseando-se apenas em experiências, por meio do método de primeira-visita ou o método de todas-visitas. Por meio do método de primeira-visita, a partir do momento que o agente passa por um estado *s* pela primeira vez, a recompensa descontada é acumulada até o término do episódio. No fim do episódio, o valor do estado *s* é uma média das recompensas acumuladas pelo estado *s* de todos os episódios já realizados. O Algoritmo 4.3 exemplifica o pseudocódigo.

```
para todo s ∈ S faça
    inicialize V(s) e π(s) arbitrariamente
    inicialize Returns(s) como uma lista vazia
fim
repita para sempre
    gere um episódio E a partir de π
    para cada s ∈ E faça
        R ← recompensa a partir da primeira visita
        anexe R em Returns(s)
        V(s) ← média(Returns(s))
    fim
fim
```

Algoritmo 4.3. Método de primeira-visita de Monte Carlo.

Sendo assim, os métodos MC necessitam de tarefas baseadas em episódios para serem implementados, não sendo possível utilizá-los em caso de tarefas contínuas, isto é, sem um estado terminal.

A estimativa dos valores dos estados é suficiente quando se tem algum conhecimento sobre o ambiente, como suas transições de estado, pois basta olhar o próximo passo e decidir pelo maior valor de estado para caracterizar a política ótima. Caso contrário, é mais apropriada a aplicação dos métodos de MC para encontrar os valores das ações dos estados, $Q^{\pi}(s, a)$. Sendo assim, um dos principais objetivos dos métodos de MC é estimar $Q^*(s, a)$ [27].

A avaliação de política segue os mesmos fundamentos do caso da avaliação dos valores dos estados, porém deve ser analisada cada ocorrência do par (s, a), e não somente s. Pode ser que um par (s, a) nunca seja visitado simplesmente seguindo os passos do algoritmo. Por isso, para a avaliação de política por meio do valor das ações, o agente conta com a exploração inicial. Na exploração inicial, a cada episódio, todos os pares (s, a) tem probabilidade não nula de serem escolhidos como o estado e ação iniciais, garantindo, assim, que todos os pares (s, a) sejam visitados.

4.4.2. Estimação da Política Ótima

Da mesma forma como nos métodos de PD, a estimação da política ótima nos métodos de Monte Carlo é feita por sucessivas avaliações e melhorias de política. Para a melhoria de política, baseando-se nas estimativas já encontradas, no término de cada episódio, a política é modificada para uma política gulosa, isto é, a ação tomada no estado s é aquela com maior valor. Com essa modificação, o algoritmo para a busca de políticas ótimas é como mostrado pelo pseudocódigo do Algoritmo 4.4.

```
para todo (s, a) faça
   inicialize Q(s, a) e π(s) arbitrariamente
   inicialize Returns(s, a) como uma lista vazia
fim
repita para sempre
   gere um episódio E por π com exploração inicial
   para cada (s, a) ∈ E faça
      R ← recompensa a partir da primeira visita
      anexe R em Returns(s, a)
      Q(s, a) ← média(Returns(s, a))
   fim
   para cada s ∈ E faça
      π(s) ← arg max_a Q(s, a)
   fim
fim
```

Algoritmo 4.4. Estimação da política ótima por Monte Carlo.

Existem ainda maneiras de evitar a exploração inicial. Para isso, é preciso um controle diferente nas melhorias de política. Umas das alterações necessárias é a utilização de uma estratégia de exploração ε-gulosa. Nessa estratégia, o agente segue uma política gulosa, porém, com probabilidade ε o agente executa uma ação qualquer dentre as ações possíveis. Além disso, é possível avaliar uma política enquanto se segue outra política. Técnicas deste tipo são conhecidas como treinamento *off-policy*. Até aqui, a maioria dos treinamentos abordados é *on-policy*, isto é, a politica avaliada é a própria que está sendo seguida.

4.5. APRENDIZADO POR DIFERENÇA TEMPORAL

Os métodos de aprendizado por diferença temporal (DT) são uma tentativa de solucionar problemas por meio da combinação dos métodos de PD e MC. Como visto, nas soluções por PD é necessário o modelo perfeito do ambiente para, com isso, calcular novas estimativas com base nas anteriores sem a necessidade de esperar pelo fim de um episódio, portanto, fazem *bootstrap*, isto é, recalculam os valores a cada passo do algoritmo. Contudo, métodos de PD não permitem o aprendizado *online,* isto é, não permitem que o agente aprenda enquanto pratica as ações. Já os métodos de MC não necessitam de um modelo completo do ambiente e são meios de aprendizado *online,* com a desvantagem de terem que esperar pelo fim de um episódio para recalcular suas estimativas, impossibilitando a solução de problemas contínuos.

Os métodos de DT são meios de aprendizado online baseado em experiências, sem a necessidade de um modelo perfeito do ambiente, que utiliza algumas propriedades da PD para recalcular suas estimativas a cada passo do agente, ou seja, fazem *bootstrap.*

4.5.1. Avaliação de Política

Na avaliação de política por meio dos métodos de MC, era necessário esperar até o término do episódio, ou seja, o método calcula apenas R_t, como na Equação (4.2), e faz a estimativa do valor de cada estado como a média de R_t, como na Equação (4.5). A chave para a avaliação de política dos métodos de DT é fazer uso da propriedade

$$\begin{aligned} V^{\pi}(s) &= E_{\pi}\left\{\sum_{k=0}^{\infty}\gamma^{k} r_{t+k+1} \,\middle|\, s_t = s\right\} \\ &= E_{\pi}\left\{r_{t+1} + \gamma V^{\pi}(s_{t+1}) \,\middle|\, s_t = s\right\}, \end{aligned}$$

assim como era feito nos métodos de PD. Assim, quando o agente se encontra num estado s, pratica uma ação, observando a recompensa r e transitando para um estado s', seguindo a política π, seu valor é atualizado para uma nova estimativa $V^{\pi}(s) = r + \gamma V^{\pi}(s')$. Contudo o valor do estado contido anteriormente não deve ser descartado, sendo que o novo valor deve ser uma atualização para a estimativa. Uma regra simples de atualização para este caso é

$$V^{\pi}(s_t) = V^{\pi}(s_t) + \alpha\left[r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)\right], \tag{4.17}$$

sendo $0 \leq \alpha \leq 1$ a taxa de aprendizado e, como pode ser verificado, quando $\alpha = 1$ o valor contido anteriormente no estado é descartado. Esse modo de atualização é conhecido como TD(0) (TD refere-se a *temporal difference*).

4.5.2. Estimação da Política Ótima: Sarsa

Assim como nos métodos de MC, estimar o valor dos estados pode não ser a melhor solução para problemas de aprendizado com pouco conhecimento sobre o ambiente. Sendo assim, os métodos de DT também buscam por políticas ótimas através da estimativa do valor das ações. Além disso, existe a necessidade de exploração para garantir que todos os estados e ações sejam visitados. A exploração consiste em sair da política e praticar uma ação arbitrária, como já mencionado, uma estratégia de exploração ε-gulosa faz justamente isso.

O primeiro modo de se buscar políticas ótimas tem como base as técnicas utilizadas nos métodos de PD e MC e consiste em uma forma *on-policy* de estimação, isto é, as melhorias são encontradas seguindo a própria política. Modificando a Equação (4.17) como

$$Q^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q^{\pi}(s_{t+1}, a_{t+1}) - Q^{\pi}(s_t, a_t)\right], \tag{4.18}$$

obtém-se uma regra de atualização para a estimativa dos valores das ações. Sendo assim, para atualizar o valor de uma ação a_t do estado s_t é preciso observar a recompensa r_{t+1} e, ainda, ter conhecimento sobre o valor da ação a_{t+1} no estado s_{t+1}, que é $\pi(s_{t+1})$ devido à característica *on-policy* do método. O quinteto (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}), necessário para a atualização, batiza

este algoritmo como Sarsa, cujo pseudocódigo é mostrado no Algoritmo 4.5.

```
para todo (s, a) faça
   inicialize Q(s, a) arbitrariamente
fim
repita para cada episódio
   inicialize s
   escolha a por uma política de Q de s com exploração
   para cada passo faça
      pratique a e observe r e s'
      escolha a' por uma política de Q de s' com exploração
      Q(s,a) ← Q(s,a)+α[r+γQ(s',a')−Q(s,a)]
      s ← s'; a ← a'
   até que s seja terminal
fim
```

Algoritmo 4.5. Sarsa.

4.5.3. Estimação da Política Ótima: Q-Learning

Introduzido pela primeira vez por [30], o algoritmo de Q-Learning é um dos principais resultados na área de aprendizado por reforço [27]. Q-Learning é um algoritmo *off-policy*, isto é, encontra a solução para a política ótima praticando uma política qualquer. Sua regra de atualização é dada por

$$Q^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_{a} Q^{\pi}(s_{t+1}, a) - Q^{\pi}(s_t, a_t)\right]. \tag{4.19}$$

Nota-se que a atualização independe da ação que será praticada pela política no próximo estado. Em vez disso, o algoritmo atualiza o valor da ação com o valor da ação máximo no próximo estado, fazendo dele um algoritmo de aprendizado *off-policy*. O algoritmo, em sua estrutura, diverge muito pouco do algoritmo Sarsa e seu pseudocódigo é mostrado no Algoritmo 4.6.

```
para todo (s, a) faça
   inicialize Q(s, a) arbitrariamente
fim
repita para cada episódio
   inicialize s
   para cada passo faça
      escolha a por uma política de Q de s com exploração
      pratique a e observe r e s'
      Q(s,a) ← Q(s,a)+α[r+γ max_a' Q(s',a')−Q(s,a)]
      s ← s'
   até que s seja terminal
fim
```

Algoritmo 4.6. Q-Learning.

Mostra-se em [31] que, conforme o número de visitas aos pares (s, a) cresce, $Q(s, a)$ converge para Q^* com probabilidade 1. Sendo assim é necessário que todos os pares (s, a) sejam visitados um grande número de vezes, por isso a escolha de uma estratégia de exploração, como a ε-gulosa, por exemplo.

4.6. EXEMPLOS DE APLICAÇÃO

Para mostrar a aplicação das técnicas de aprendizado por reforço, utilizando os métodos das diferenças temporais, três exemplos serão construídos e resolvidos.

Os dois primeiros exemplos são problemas clássicos da literatura e são didaticamente simples de serem construídos e abordados. Por sua vez, o terceiro exemplo é uma aplicação mais voltada para a área central do trabalho, tentando resolver um problema de quantização.

4.6.1. O Tabuleiro com Ventos

O primeiro exemplo de aplicação do método das diferenças temporais consiste no problema de atravessar uma região com ventos que empurram o agente. O exemplo foi retirado de [27] com algumas modificações nos resultados.

O agente se encontra em um tabuleiro como o da Figura 4.3. Os números indicam a força do vento agindo sobre determinada coluno, de modo que quando o agente sai daquela coluna ele é empurrado para cima o número de casas da intensidade do vento. O objetivo da tarefa é atravessar o tabuleiro, indo da posição inicial I até a posição final F, com o menor número de passos possível.

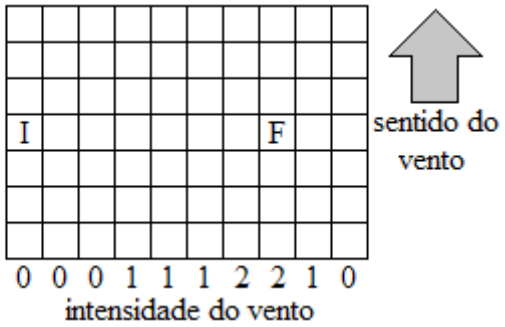


Figura 4.3. Problema do tabuleiro com ventos.

O agente pode ser mover em qualquer sentido para os lados, para cima, para baixo, em diagonal e pode também ficar parado, deixando o vento o levar. Para solucionar o problema, a estratégia de exploração será ε-gulosa com Sarsa e Q-Learning, para fins de comparação. A exploração ocorre com probabilidade de 10%,ou seja, $\varepsilon = 0{,}1$, para forçar o agente a encontrar o menor caminho, as recompensas são sempre iguais a –1 e não há desconto nas recompensas, isto é, $\gamma = 1$;

O resultado obtido, exibido na Figura 4.4, mostra quantos passos foram necessários para cumprir a tarefa durante determinado episódio. O resultado é apresentado como um gráfico de escada, sendo que o número do degrau indica o número do episódio e a largura do degrau indica quantos passos o agente ficou naquele episódio. De acordo com a Figura 4.4, o agente levou 20 passos para concluir o episódio de número 100 e 7 passos para concluir o episódio de número 101.

Por meio dessa análise, resolver vários episódios com poucos passos resulta em uma escada bastante inclinada, enquanto que resolver episódios longos, ou seja, com muitos passos, leva a uma menor inclinação da curva.

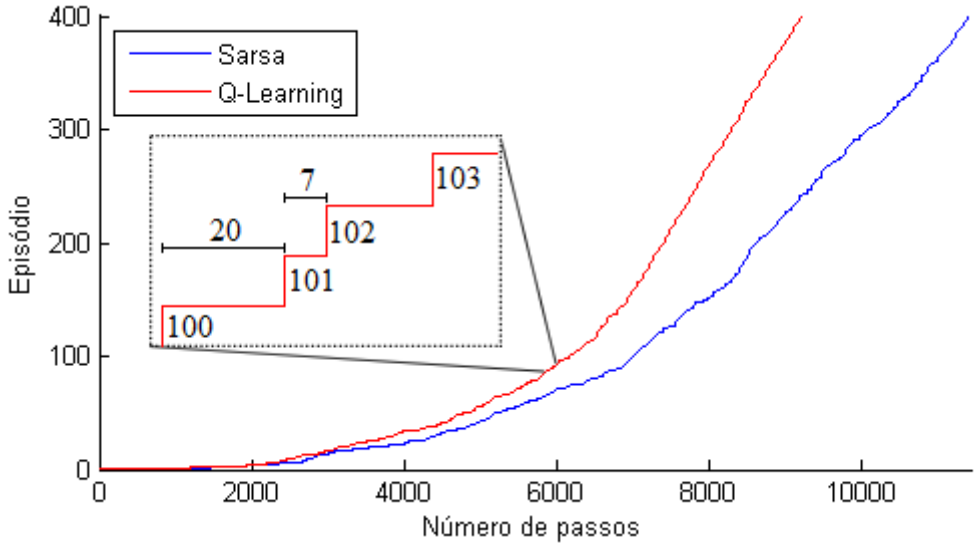


Figura 4.4. Resultado do problema do tabuleiro com ventos.

Levando em consideração a inclinação das curvas, o desempenho do Q-Learning foi superior, sendo que sua curva se torna uma reta com a maior inclinação possível (menor número de passos) antes da curva do Sarsa. Isso porque o algoritmo Sarsa leva em consideração a probabilidade de exploração na estimação dos valores, resultando numa convergência mais demorada para um problema simples. A flutuação nas curvas também é explicada pela exploração feita pelo agente.

4.6.2. O Penhasco

O segundo exemplo, também retirado de [27], visa a comparação dos algoritmos Sarsa e Q-Learning em um cenário onde a exploração pode ser um problema considerando a modelagem do ambiente. A Figura 4.5 mostra o tabuleiro do problema do penhasco. O objetivo do agente é sair do ponto inicial I e chegar até o ponto final F com o menor número de passos. Diferentemente do problema anterior, o agente pode apenas se mover para os lados, para cima ou para baixo. Com o objetivo de minimizar seus passos, o agente sempre recebe

recompensa igual a –1, a menos que ele pise no penhasco, neste caso, o agente retorna para a posição inicial e recebe uma recompensa igual a –100. Da mesma forma como no problema anterior, as recompensas não são descontadas e a probabilidade de exploração é de 10% para uma estratégia de exploração ε-gulosa.

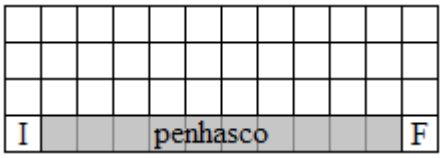


Figura 4.5. Problema do penhasco.

Devido à possibilidade de exploração, o caminho que seria ótimo se torna perigoso, pois percorrendo-o o agente caminha mais próximo ao penhasco e, devido à estratégia de exploração, suas chances de cair são mais altas. Resolvendo o problema com ambos os algoritmos, verifica-se que a política final do Q-Learning é a política com o menor caminho, resultando em algumas quedas. Já a política final do Sarsa foi um caminho mais seguro, isto é, longe do penhasco, resultando em menos quedas. A Figura 4.6 mostra como ficaram as políticas obtidas.

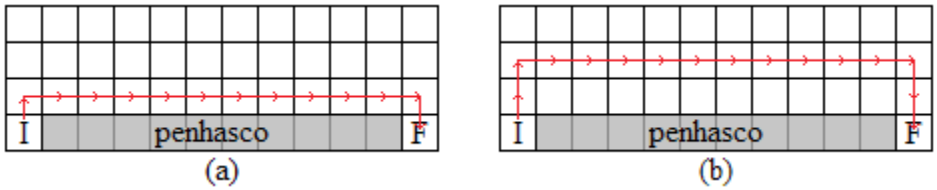


Figura 4.6. Política do problema do penhasco utilizando (a) Q-Learning e (b) Sarsa.

Na Figura 4.7 está a comparação da recompensa acumulada por episódio. O resultado exibido está suavizado por uma função de média móvel de 200 amostras para 700 episódios gerados. Analisando o gráfico, na média, o algoritmo Sarsa se sobressaiu, isso porque levou em consideração a probabilidade de exploração, evitando quedas no penhasco. Essa afirmação é corroborada pela Figura 4.8, que exibe a recompensa acumulada por episódio sem suavização.

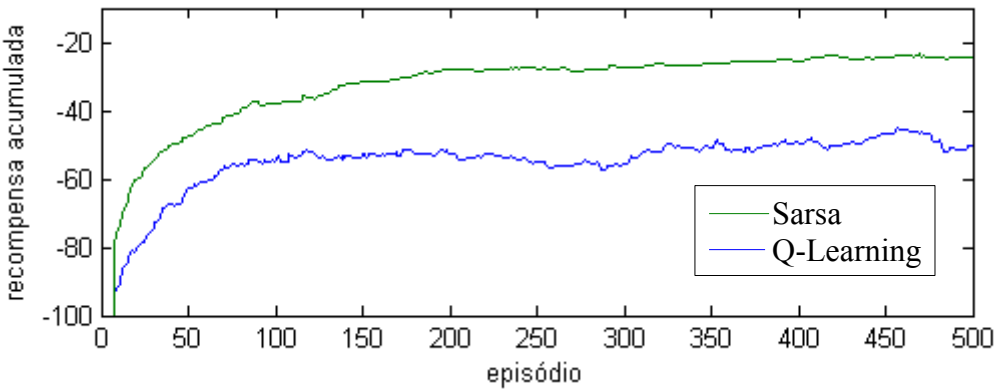


Figura 4.7. Recompensa acumulada por episódio no problema do penhasco (suavizado).

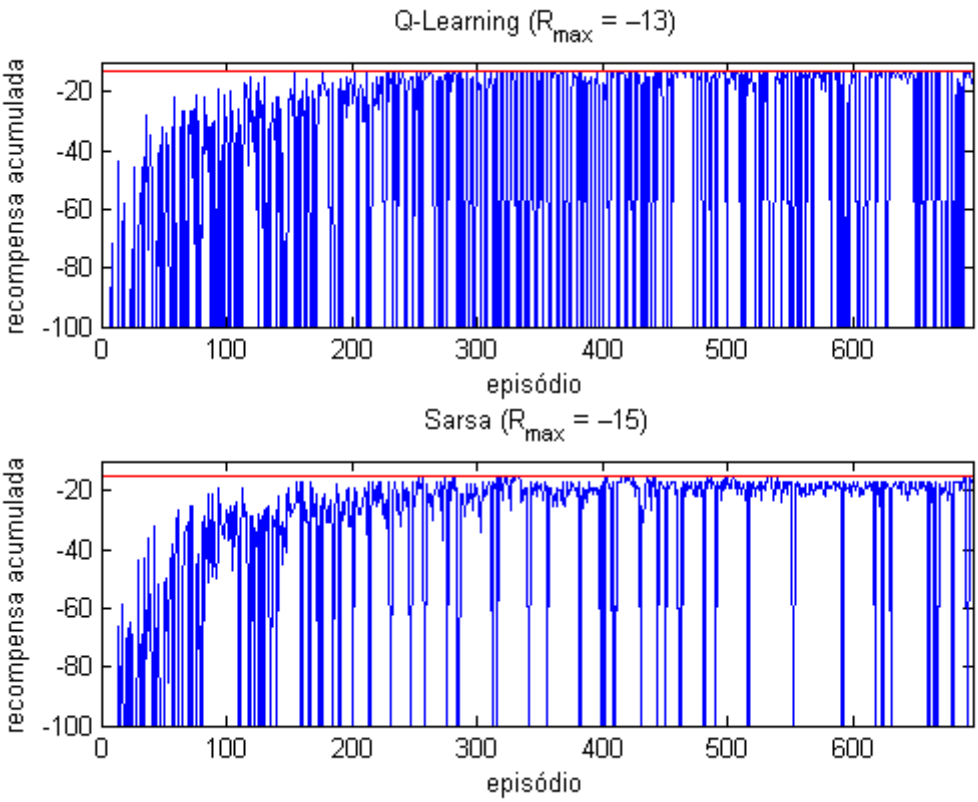


Figura 4.8. Recompensa acumulada por episódio no problema do penhasco.

Como pode ser visto, a recompensa máxima do algoritmo Q-Learning foi superior, porém, é possível perceber que a frequência de quedas foi muito maior, resultando em recompensas acumuladas muito baixas. Esse fato foi o responsável por prejudicar o desempenho médio do Q-Learning face ao desempenho médio do Sarsa quando a estrutura do problema torna a exploração um possível fator de prejuízo.

4.6.3. Otimização de um Quantizador com Q-Learning

Neste problema, um sinal digital modulado em banda base por pulsos cossenoidais, como mostra a Figura 4.9, deve ser quantizado em 6 níveis simétricos.

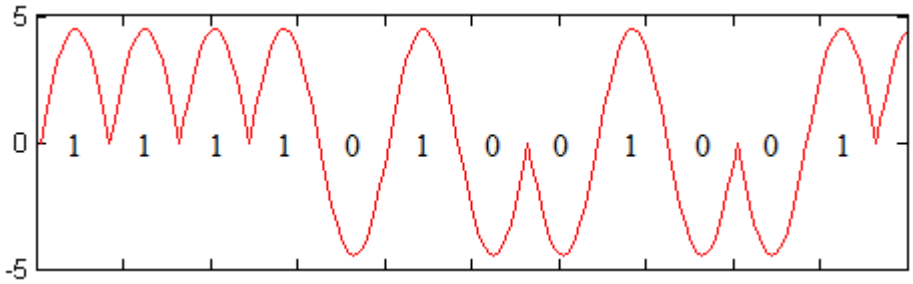


Figura 4.9. Sinal digital que deve ser quantizado.

O objetivo do agente é escolher os limiares do quantizador de modo que o sinal quantizado tenha o menor erro de quantização possível. O sinal total é composto por 2.000

bits, sendo que o agente é recompensado a cada 10 *bits*, ou seja, 10 pulsos, quantizados. Como o quantizador é simétrico, o agente controla apenas 3 dos 6 níveis de quantização. Cada nível de quantização é representado por sua largura, que pode assumir os valores de 0,3 a 3,0 em passos de 0,3, ou seja, 10 larguras diferentes. O valor de quantização é igual ao valor intermediário da região de quantização definida pelo níveis, por exemplo, se os níveis possuem todos a mesma largura de 0,6, o primeiro valor de quantização é 0,3, o segundo é 0,3 + 0,6 = 0,9 e o terceiro é 0,3 + 1,2 = 1,5.

Os níveis de quantização são os estados do ambiente. O agente pode aumentar ou diminuir um nível ao passo de 0,3 ou, ainda, permanecer no nível que está. O agente controla apenas um nível por vez e é recompensado por isso. Como o objetivo do agente é minimizar o erro, a recompensa recebida é igual ao negativo do erro quadrático médio entre o sinal na entrada do quantizador e o sinal quantizado com os níveis atuais. As recompensas são descontadas com um fator $\gamma = 0{,}5$ apenas para informar ao agente sobre ações passadas. Assim como nos demais casos, é aplicada uma estratégia de exploração ε-gulosa com probabilidade de exploração igual a 10%. A Figura 4.10 mostra o resultado obtido após todos os pulsos terem sido quantizados.

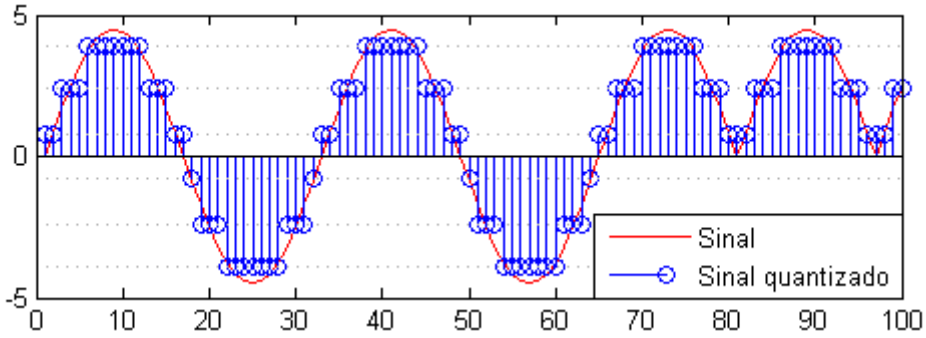


Figura 4.10. Quantização com Q-Learning

A curva da evolução do erro quadrático médio com o passar dos instantes, exibida na Figura 4.11, mostra a convergência do algoritmo para uma política ótima, capaz de manter os níveis nos quais o erro de quantização é o menor possível, igual a 0,2392.

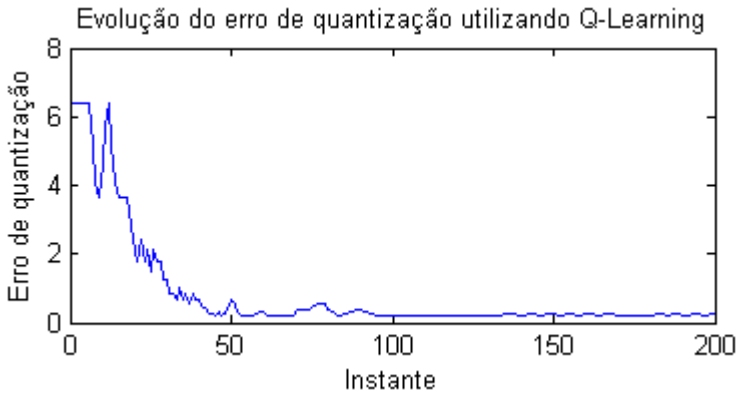


Figura 4.11. Erro quadrático médio de quantização.

4.7. CONCLUSÃO

Neste capítulo foram introduzidos os conceitos básicos de aprendizado por reforço para casos discretos. Existem ainda modificações nos algoritmos aqui mostrados que permitem a solução de sistemas mais complexos, porém, para os objetivos deste trabalho, os conceitos até aqui abordados são suficientes.

O aprendizado de máquina pode ser bastante útil para resolver problemas dinâmicos. Porém os algoritmos propostos não são capazes de resolver tarefas por si só, pois é preciso um modelo do ambiente, sendo que este modelo deve ser completamente descrito para os casos dos métodos de programação dinâmica.

Os métodos de diferença temporal, como visto, são os mais simples e eficientes métodos disponíveis para o aprendizado por reforço. Porém, a simplicidade do problema não está no algoritmo utilizado, como o Q-Learning. Nas aplicações vistas no capítulo, foi necessário modelar o ambiente, definindo ações, transições e recompensas. A modelagem do ambiente é, então, o fator crucial para o sucesso da solução de um problema por aprendizado por reforço, sendo que qualquer fator pode influenciar na velocidade de convergência ou até mesmo na não convergência da solução.

Sendo assim, os métodos de aprendizado por reforço se mostram suficientes para resolver alguns problemas das comunicações, como é o caso de equalizadores e estimadores de canal. Além disso, uma aplicação direta é o controle do quantizador de um decodificador de Viterbi, como comentado no capítulo anterior. A procura por uma solução deste problema é abordada no próximo capítulo.

5. DECODIFICADOR *SOFT* ADAPTATIVO

5.1. INTRODUÇÃO

Conforme estudado no capítulo 3, a implementação de um decodificador de Viterbi de decisão *soft* necessita de uma quantizador. A quantização dos valores resulta em uma perda de informação e, consequentemente, acarreta em perdas no desempenho do decodificador. Além disso, a configuração dos níveis de quantização também podem influenciar no desempenho do quantizador em um canal AWGN [4].

Neste capítulo será empregado o algoritmo Q-Learning apresentado no capítulo anterior para tentar resolver o problema de encontrar dinamicamente os níveis ótimos de quantização para um decodificador de Viterbi de decisão *soft*.

5.2. DESCRIÇÃO

Um resultado bastante consolidado, como mencionado no capítulo 3, é que um quantizador de 3 *bits* tem o desempenho bastante próximo de um quantizador de níveis infinitos, ou seja, não apresenta muitas diferenças se comparado a um sistema não quantizado.

O funcionamento de um quantizador de 3 *bits* é mostrado na Figura 5.1. Como pode ser visto, existem 8 regiões separadas por 7 limiares. Nos problemas abordados por este trabalho, o codificador será simétrico, isto é, os limiares das regiões negativas serão os mesmos das regiões positivas. Ainda, um dos limiares é definido no zero. Sendo assim, para caracterizar o quantizador, basta definir 3 limiares, L_1, L_2 e L_3.

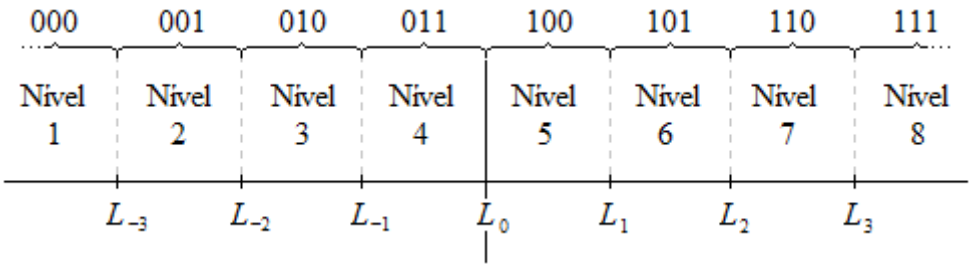


Figura 5.1. Quantizador.

Utilizando o algoritmo Q-Learning, o agente deve ser capaz de aprender os níveis de quantização ótimos no sentido de melhorar o desempenho do decodificador. Para atingir este objetivos, algumas estratégias de solução foram elaboradas. Contudo, para compreender o problema, é preciso descrever o ambiente em que o agente está inserido, abordando a descrição dos estados, ações e recompensas.

O ambiente de aprendizado é composto pelo decodificador de Viterbi e, ainda, uma estimativa do canal de comunicação, avaliada por sua relação sinal-ruído no receptor. O

agente tem informações sobre os limiares de quantização do decodificador e sobre a relação sinal-ruído do canal e monta seus estados e ações com base nisso. A cada bloco decodificado, o agente é recompensado e calcula os valores de seus pares estado-ação. Antes do início de uma nova decodificação, o agente informa ao decodificador quais níveis utilizar com base em seu estado e será posteriormente recompensado por isso. O ciclo se repete para garantir um grande número de visitas aos estados e ações. A Figura 5.2 ilustra funcionamento do sistema.

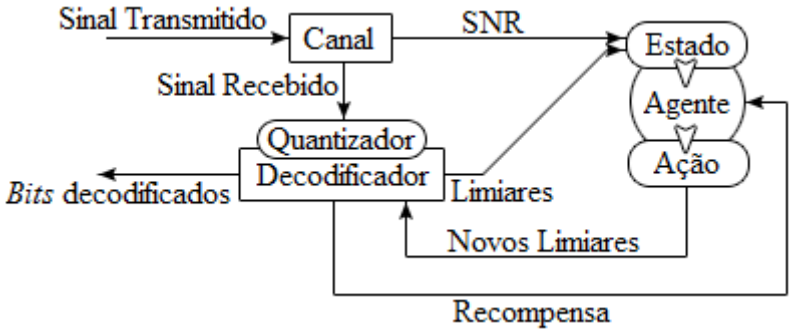


Figura 5.2. Sistema de aprendizado.

5.3. MODELO DO PROBLEMA

Para resolver o problema de se encontrarem os melhores níveis utilizando aprendizado por reforço, é preciso modelar o problema. Como visto no capítulo anterior, é preciso um modelo de estados, recompensas e ações. Para este problema, algumas modelagens diferentes são testadas e comparadas.

Nesta seção, será apresentada a modelagem de estados e ações. Além disso, são propostas três estratégias combinando diferentes abordagens entre estados e ações. Serão abordadas também três formas diferentes de se gerar o sinal de recompensa para o agente.

5.3.1. Estados

Os estados do sistema são descritos pelos limiares do quantizador. Na modelagem utilizada, os limiares podem variar discretamente entre valores diferentes previamente definidos. Cada estado descreve a largura dos níveis, ou seja, a diferença entre dois limiares consecutivos. Como a simulação é baseada na modulação BPSK dos sinais, em que os níveis dos símbolos transmitidos podem assumir os valores +1 ou –1, a escolha para a variação dos limiares foi $L_i^{min} = 0{,}1$ e $L_i^{max} = 1{,}0$ em variações de $\Delta L = 0{,}1$, formando um total de 10 valores para cada limiar.

Ainda, a relação sinal-ruído deve ser levada em conta. Para todas as estratégias, a relação E_b/N_0 foi dividida em 7 regiões, gerando ainda mais estados. O estado é formado, então, pela combinação dos níveis e da atual relação sinal-ruído.

5.3.2. Ações

Foram propostas duas abordagens para as ações do agente. Na primeira delas, o agente é capaz apenas de deslocar os níveis por passos, isto é, cada nível pode ser modificado pela variação mínima ou não ser alterado, ou seja, $L_i^{t+1} = L_i^t + \{\Delta L; -\Delta L; 0\}$. Sendo assim, para cada limiar há um total de 3 ações.

Na outra abordagem, a partir de um estado o agente pode alterar seu limiar para qualquer limiar disponível, totalizando 10 ações para cada limiar.

5.3.3. Estratégia: Deslocamento de Limiares (DL)

Nessa estratégia, o agente controla simultaneamente todos os níveis, porém, cada nível pode ser alterado apenas pelo passo mínimo ΔL. Como o agente deve manter a configuração de cada um dos três limiares, existe um total de 10×10×10 configurações de limiares possíveis. Com a adição da informação sobre a relação sinal-ruído, o agente conta com um total de 7×10×10×10 = 7.000 estados.

Quanto às ações, o agente deve ser capaz de controlar cada um dos limiares simultaneamente, gerando um total de 3×3×3 = 27 ações. Sendo assim, o total de pares estado-ação do agente é 7.000×27 = 189.000.

5.3.4. Estratégia: Salto de Limiares (SL)

Nessa estratégia, o agente utiliza as mesmas configurações de estados da estratégia de deslocamento de limiares (DL). A diferença está nas ações disponíveis para cada estado, pois o agente pode ir de uma configuração de limiares para qualquer configuração de limiares. Novamente, como o agente deve controlar os três limiares simultaneamente, existe um total de 10×10×10 = 1.000 ações para cada estado.

O total de pares estado-ação do agente é, portanto, 7.000×1.000 = 7.000.000.

5.3.5. Estratégia: Deslocamento Individual de Limiares (DI)

No deslocamento individual de limiares (DI), diferentemente das estratégias de deslocamento dos limiares (DL) e salto de limiares (SL), não existe apenas um, mas três agentes. Cada agente é responsável por controlar um único limiar e é recompensado individualmente por suas ações. Assim como nas estratégias anteriores, cada limiar pode assumir 10 valores diferentes com as mesmas variações. Adicionada a informação sobre a relação sinal-ruído do canal, cada agente possui um total de 7×10 estados.

Assim como na estratégia DL, os agentes podem apenas deslocar seus limiares em

passos de ΔL. Como cada limiar é controlado separadamente, cada agente conta com apenas 3 ações. Cada agente possui, portanto, 70×3 = 210 pares estado-ação.

5.3.6. Recompensa: Taxa de Erro de *Bit*

A recompensa de um problema de aprendizado por reforço define os objetivos do agente. Inicialmente, os objetivos do problema eram encontrar níveis de quantização que minimizassem a BER. Com esse objetivo, a recompensa dada ao agente foi definida como o inverso da taxa de erro de *bit*. Para isso, a cada bloco de *N bits* decodificados, o decodificador estimava a BER e recompensava o agente.

O problema dessa abordagem de recompensa é que, caso o decodificador não erre, a recompensa será nula. Com isso, assumindo a inicialização dos valores das ações com valores nulos, para garantir a convergência dos níveis para níveis ótimos, seria necessário uma grande quantidade de blocos decodificados e, considerando blocos nulos, o número de blocos com erro deveria ser grande.

Assumindo uma episódio de um milhão de *bits*, o número de recompensas dadas ao agente é 1.000.000/*N*, em que *N* é o número de *bits* de cada bloco decodificado. Espera-se que diminuindo o número de *bits* por bloco o número de recompensas aumente. Porém, para um aprendizado inicial, apenas as recompensas não nulas são efetivamente úteis. Fixando em cem mil o número total de *bits* em um episódio e variando o tamanho dos blocos, o número de recompensas não nulas em função da relação sinal-ruído e do tamanho dos blocos é mostrada na Figura 5.3.

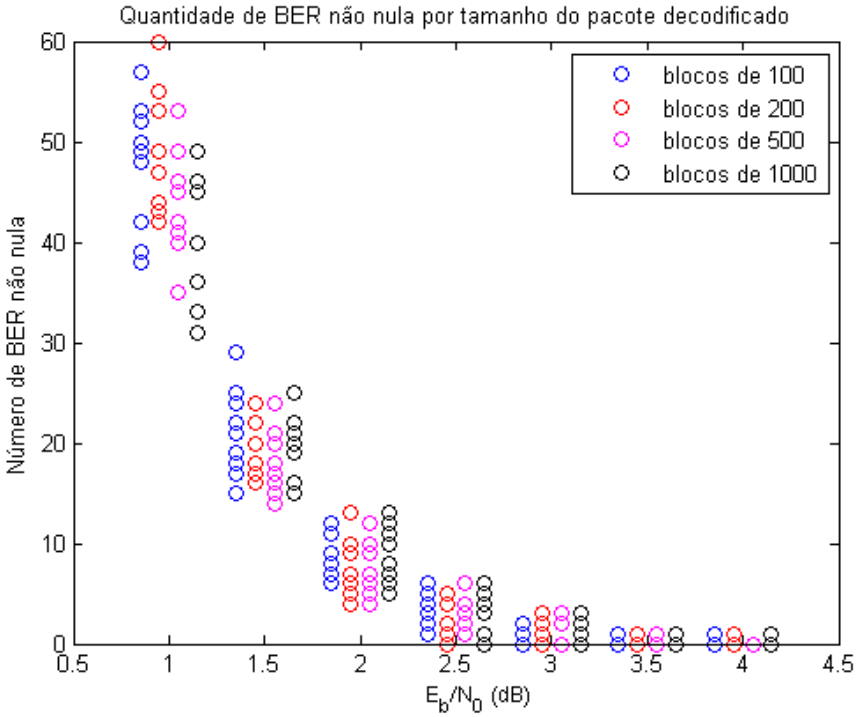


Figura 5.3. Número de recompensas não nulas de acordo com o tamanho dos blocos.

Como pode ser visto, o número de recompensas não nulas não varia muito com a mudança dos blocos, apenas mudando em função da relação sinal-ruído. Sendo assim, não importa o tamanho dos blocos decodificados, mas apenas o número total de *bits* gerados no episódio. Além disso, a BER de cada bloco apresenta variância elevada, principalmente para relação sinal-ruído baixa, não sendo fiel o suficiente aos níveis de quantização a ponto de ser uma recompensa confiável.

Com isso, o aprendizado com recompensas baseadas na BER precisa de muitos *bits* por episódio e, ainda, não converge com facilidade. Por estes motivos, a recompensa por BER foi descartada. Para resolver o problema da recompensas, baseado nos resultados de [4], outros esquemas de recompensas foram elaborados.

5.3.7. Recompensa: Erro de Quantização

A segunda alternativa consiste em recompensar o agente por meio do erro de quantização. Assim, o objetivo seria reduzir o erro quadrático médio entre os sinais de entrada e suas respectivas quantizações em cada bloco decodificado. Assim como pode ser visto na Figura 5.4, esta estratégia de recompensa parece razoável, tenho em vista que o erro diminui com a BER, apresenta variância independente da relação sinal-ruído e é sensível aos níveis de quantização.

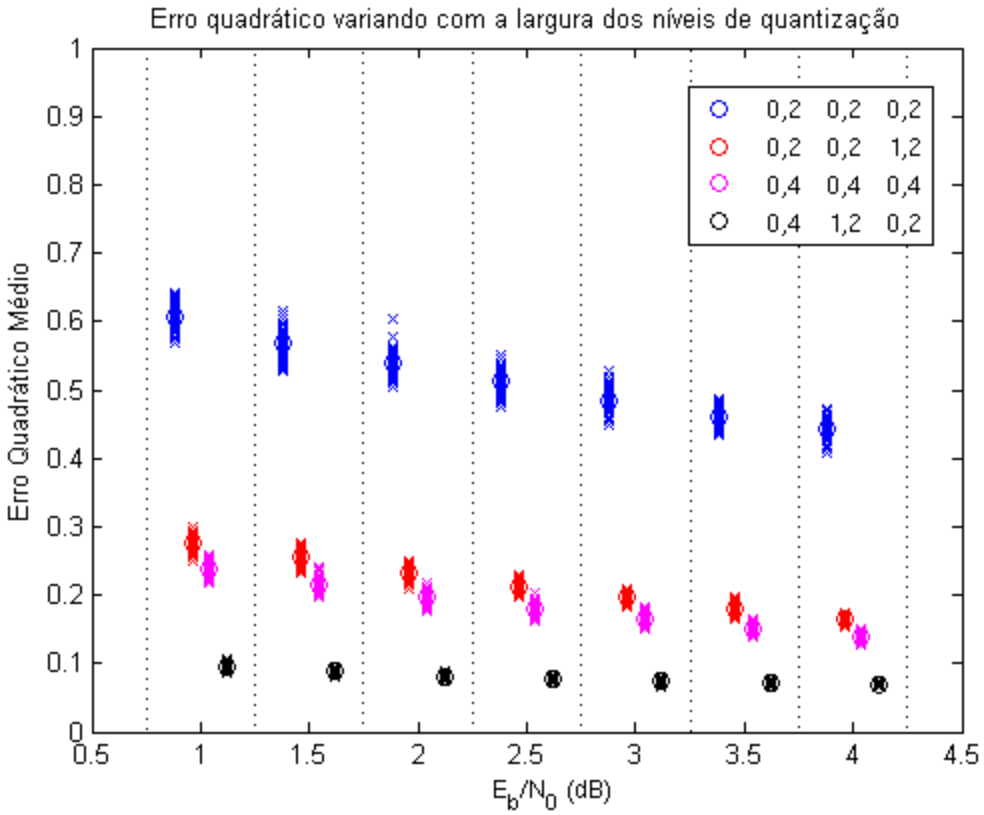


Figura 5.4. Erro quadrático médio gerado pela quantização.

Para verificar a validade deste técnica de recompensa, é preciso verificar a relação entre o erro de quantização e a BER. O resultado de simulações para validar essa dependência é mostrado na Figura 5.5. Como pode ser visto, os limiares com menor BER não são os mesmos com menor erro. Sendo assim, o desempenho do decodificador não seria alterado, fazendo com que outra estratégia de recompensa seja pensada.

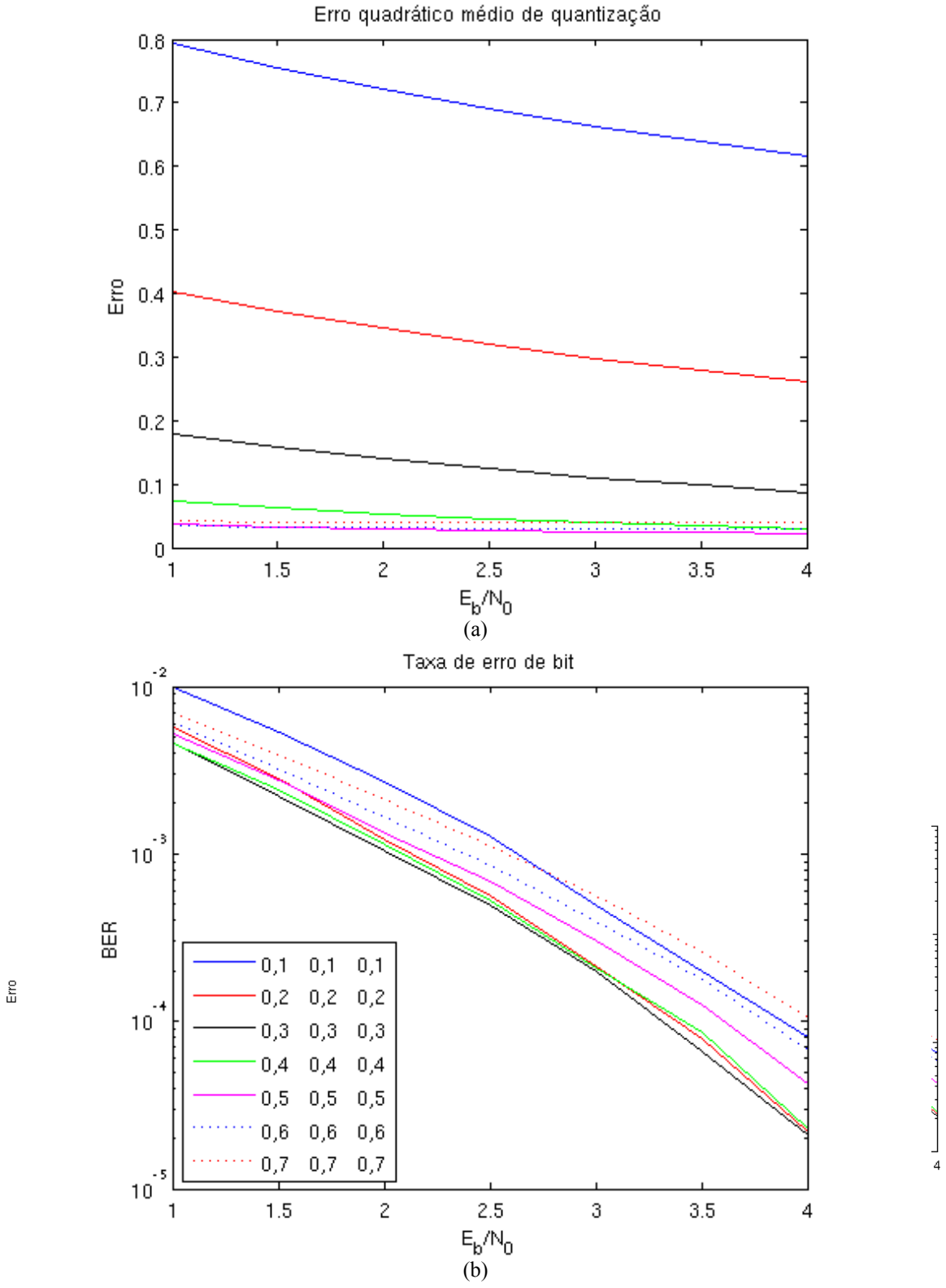


Figura 5.5. Relação entre (a) o erro de quantização e (b) a BER para alguns limiares de quantização.

5.3.8. Recompensa: Renormalizações

Na solução utilizada, a recompensa foi baseada no desempenho do quantizador, aferido pela frequência de renormalização [32]. Como é abordado em [4] e mostrado em [32], a frequência de renormalizações de um decodificador com decisão *soft* é uma forma de mensurar o desempenho do quantizador. No capítulo 3, foi visto que uma renormalizações das métricas ocorre sempre que as métricas totais estão muito altas para evitar *overflow*.

Para encontrar a frequência de renormalização, foi utilizada a estratégia de renormalização seguinte: sempre que a menor métrica de um passo do algoritmo de Viterbi é maior que um limiar fixado, todas as métricas daquele passo são subtraídas desse limiar. Com isso, a frequência de normalização pode ser dada pela razão entre o número de ocorrências de renormalizações e o número de símbolos decodificados no intervalo observado.

Sendo assim, para reduzir o número de renormalizações e, por consequência, aumentar o desempenho do decodificador, a recompensa dada ao agente é função da frequência de renormalizações do decodificador. O agente monitora as frequências de renormalização e, caso o valor atual seja menor que a média, houve melhora no desempenho e o agente recebe uma recompensa positiva com valor +1 e, caso contrário, o agente é recompensado com um valor negativo igual a –1. Isso faz com que o agente busque por melhoras de desempenho.

5.3.9. Constantes de Aprendizado

Para garantir a convergência do aprendizado, é utilizada a estratégia de exploração ε-gulosa, com uma probabilidade de exploração baixa, mas o suficiente para garantir visitas a todos os estados. Por isso foi escolhido $\varepsilon = 0{,}1$.

Como as recompensas dadas ao agente não são certas, isto é, para uma mesma ação em um mesmo estado em dois instantes diferentes, a recompensa pode ser diferente. Sendo assim, os valores antigos são de grande importância e, por isso, a constante de aprendizado α foi escolhida como $\alpha = 0{,}1$, a fim de preservar os valores anteriormente calculados. Valores altos para α resultaram no não aprendizado do agente. Já o fator de desconto das recompensas foi escolhido como $\gamma = 0{,}5$ para estimular a busca pelo melhor desempenho com base nas melhorias ocorridas anteriormente.

5.4. COMPARATIVO DAS ESTRATÉGIAS

Para modelar o sistema e resolver o problema, foram escolhidas as melhores estratégias, ou seja, aquelas que permitissem a convergência da solução. Conforme já mencionado, a estratégia de recompensa utilizada foi a recompensa por frequência de renormalização. Para

definir a estratégia de estados e ações, foi preciso compará-las. A Tabela 5.1 compara as estratégias DL (subseção 5.3.3), SL (subseção 5.3.4) e DI (subseção 5.3.5).

Tabela 5.1. Comparação entre as estratégias propostas.

Estratégia	Agentes	Estados	Ações por estado	Pares estado-ação
DL	1	7.000	27	189.000
SL	1	7.000	1.000	7.000.000
DI	3	70	3	210

Devido à enorme diferença entre as estratégias, a única que se mostrou compatível com a solução do problema foi a estratégia de deslocamento individual de limiares (DI). Para reduzir a complexidade das demais estratégias, é possível reduzir o número de valores possíveis para cada limiar, porém, ainda assim, utilizar mais de um agente para resolver o problema é a solução mais viável.

Como visto nas subseções anteriores, as recompensas são entregues ao agente a cada bloco de *N bits* decodificados. A princípio, não há um estado terminal para o qual o agente deva seguir, entretanto, para fins de análise de convergência do desempenho do decodificador, um episódio será descrito como a decodificação de *M* blocos.

5.5. DECODIFICADOR

Toda implementação do trabalho foi realizada em MATLAB. Como é necessária uma informação adicional do decodificador para recompensar o agente, as funções pré-definidas do MATLAB não puderam ser utilizadas, sendo necessário o desenvolvimento de uma função.

Para simplificar o algoritmo, o decodificador é capaz de decodificar apenas códigos terminados, isto é, códigos com nulos suficientes para preencher todas as memórias do codificador. Juntamente com a mensagem decodificada, a função que representa o decodificador de Viterbi com decisão *soft* também retorna a recompensa escolhida, isto é, frequência de renormalizações. Como entrada, a função recebe o código adicionado de ruído, a estrutura de treliça utilizada para a codificação e os três limiares necessários para a quantização do sinal para a decisão *soft*.

Para o cálculo das métricas de cada passo do algoritmo, os valores são quantizados no valor médio das regiões de decisão. O exemplo do valor de quantização é mostrado na Figura 5.6, indicados pela letra *Q*. De acordo com a Figura 5.6, valores na região *i* são quantizados com o valor Q_i. Importante notar que apenas os valores positivos estão mostrados, para valores negativos, como o quantizador é simétrico, basta alterar o sinal do valor.

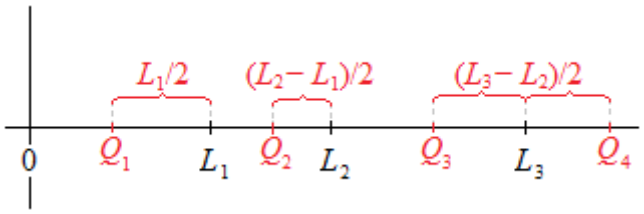


Figura 5.6. Valores de quantização para o cálculo das métricas.

As métricas foram calculadas com a distância euclidiana entre os valores referentes à quantização do sinal recebido e os valores referentes aos *bits* codificados. Para essa simulação, não foi atribuída uma tabela de métricas como mencionado na subseção 3.5.2. Contudo, como os níveis eram discretos e finitos, a tabela poderia ter sido criada, mas como processamento não é um problema nesta simulação, para simplicidade da função, foi preferível fazer a operação matemática.

5.6. ESTRUTURA DA SIMULAÇÃO

Os *bits* da simulação foram gerados aleatoriamente com a mesma probabilidade de ocorrência. Para cada episódio foram gerados 1.000 *bits*, agrupados em blocos de 100 e de 10 *bits*, para fins de comparação de desempenho, sendo gerados 1.000 episódios. No total, portanto, são gerados 1.000.000 de *bits*, permitindo uma resolução de BER de 10^{-6}.

Cada bloco foi codificado convolucionalmente por um codificador de taxa 1/2 e *constraint length* 3, descrito pelo polinômio

$$c_1 = 1 + x + x^2$$
$$c_2 = 1 + x^2.$$

Os *bits* codificados foram modulados por meio da modulação BPSK e transmitidos por um canal AWGN (adição de ruído), para relações sinal-ruído de 1,0 a 4,0 dB em passos de 0,5 dB.

A cada bloco decodificado, o agente é recompensado de acordo com a estratégia de recompensa escolhida. Nos dois casos testados, portanto, o agente com blocos de 10 *bits* foi recompensado mais vezes, porém, com uma menor confiabilidade da recompensa, visto que esta é uma média que depende do número de *bits* decodificados.

Com a estratégia DI, cada agente era recompensado uma vez a cada três blocos, isto é, a cada bloco decodificado apenas um dos três agentes agia. Isso ocorre devido à impossibilidade de manter a informação sobre quem foi o responsável pela melhora ou degradação do sistema, sendo que cada um mantém sua média de desempenho. Para corrigir isto, seria possível o emprego de algoritmos mais completos, capazes de utilizar traçados de elegibilidade [27].

5.7. RESULTADOS DAS SIMULAÇÕES

O primeiro modelo de simulação realizado utilizou a estratégia SL. Como o número de estados e ações é muito grande, o modelo se tornou inviável. Porém, antes de partir para uma outra abordagem, o número de estados foi reduzido por meio da redução de valores possíveis para os limiares. Alterando o número de valores de cada limiar para 3, o agente passou a contar com $3\times3\times3\times7 = 189$ estados e 27 ações, totalizando 5.103 pares estado-ação.

Utilizando blocos de 10 *bits* para recompensar o agente, o método de recompensas se tornou inadequado, devido à grande variância dos valores das recompensas. Isso se deve ao fato de a frequência de renormalizações ser um estimador, cuja variância é maior para um menor número de amostras. Portanto, ficou definido que as recompensas seriam entregues em blocos de 100 *bits*.

Para garantir uma única visita em cada par estado-ação, seriam necessários, portanto, 5.103 blocos, totalizando 510.300 *bits*. Como visto no capítulo 4, um grande número de visitas é esperado para que o algoritmo Q-Learning convirja, o que demandaria a decodificação de uma grande quantidade de *bits*.

Como o decodificador foi totalmente implementado em MATLAB, que é uma linguagem de baixo desempenho, a decodificação de 10.000.000 de *bits* se tornou praticamente inviável, levando a procura por um modelo mais eficiente.

Posteriormente foi pensada a estratégia DL, com menos ações. Contudo, com a mesma redução de estados proposta para a simplificação da estratégia SL, o modelo fica com a mesma quantidade de pares estado-ação e, com a mesma justificativa, sua implementação se tornou inviável.

O modelo mais adequado encontrado para solucionar o problema foi, portanto, a utilização da estratégia DI, que conta com três agentes controlando o quantizador. Como cada agente possui apenas 210 pares estado-ação, como mostra a Tabela 5.1, e a cada bloco decodificado apenas um agente é recompensado, são necessários pelo menos 630 blocos para recompensar uma vez cada par estado-ação do sistema completo.

A drástica redução do número de pares estado-ação do sistema, permitindo ainda a possibilidade de variação entre 10 valores para cada nível, fez dessa estratégia a mais adequada para a solução do problema.

Para o cálculo das recompensas, de acordo com a estratégia de renormalizações utilizada, definida na subseção 5.3.8, o limiar das métricas foi definido como 0,25, isto é, sempre que a menor métrica de um passo do algoritmo de Viterbi for maior que 0,25, é feita uma

renormalização. Como os blocos são fixos de 100 *bits* codificados, o número total de renormalizações divido por 100 é igual a frequência de renormalização do bloco.

Com 2.400.000 *bits* gerados para cada SNR, recompensando, assim, 8.000 vezes cada agente, foram obtidas as curvas de melhorias no desempenho e recompensa acumulada por bloco decodificado, visto que a redução na frequência de renormalizações resulta em um ganho no desempenho [32]. A Figura 5.7 mostra a convergência dos agentes com relação ao aumento de desempenho e a Figura 5.8 mostra a recompensa acumulada pelos agentes.

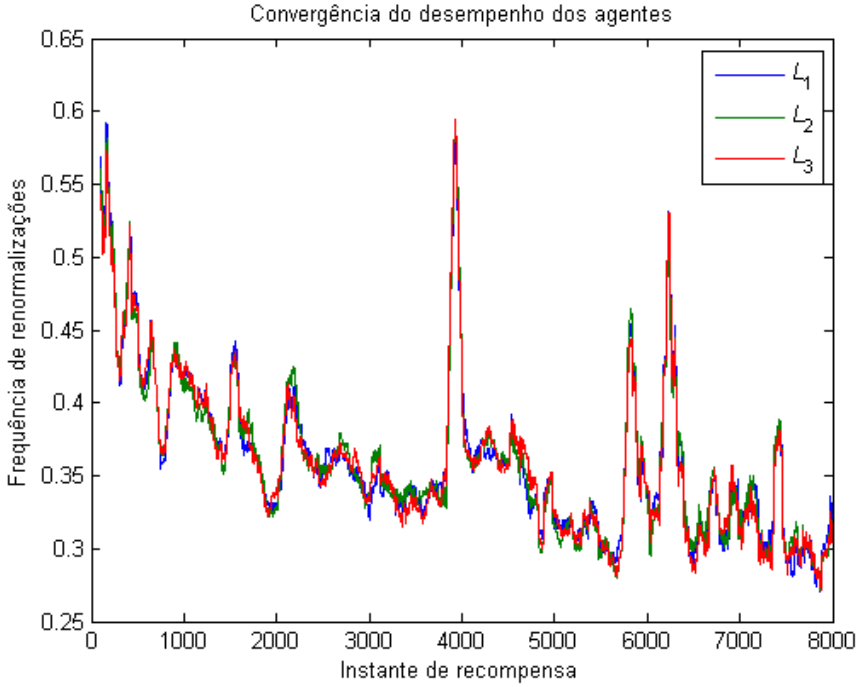


Figura 5.7. Melhoria do desempenho dos agentes.

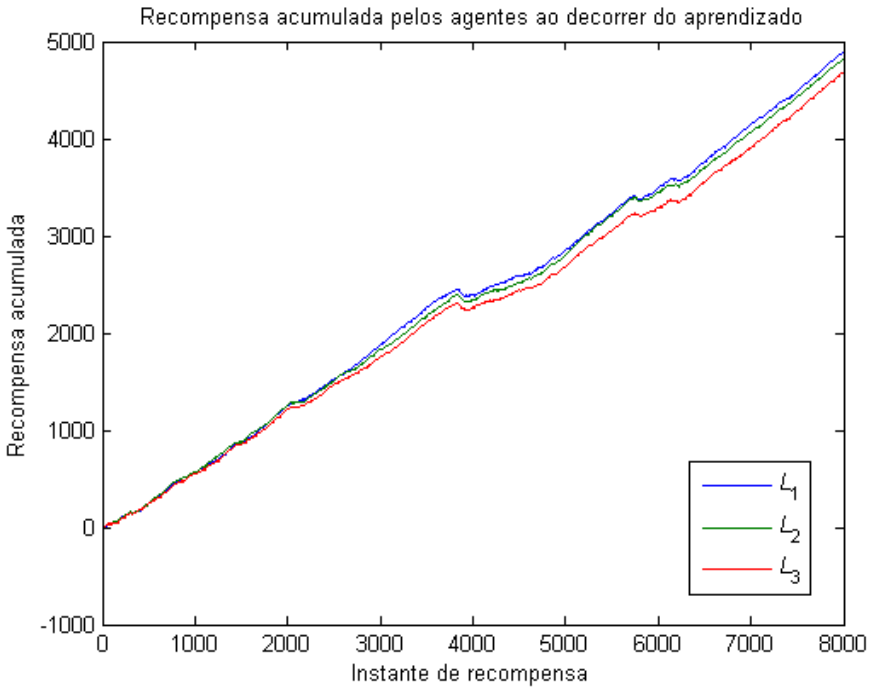


Figura 5.8. Recompensa acumulada pelos agentes.

Como pode ser visto nas Figuras 5.7 e 5.8, mesmo após 8.000 recompensas, ainda há tendência de crescimento da recompensa e de desempenho. O tempo longo de simulação impediu resultados melhores. Porém, comparando o desempenho obtido com o de um decodificador de níveis fixos, é possível observar a melhora. Os picos observados na curva da Figura 5.7 ocorrem devido à estratégia de exploração.

A comparação de desempenho foi feita com um decodificador com os limiares de quantização fixos nos valores 0,3, 0,6 e 0,9. Por meio de algumas simulações, este decodificador obteve os melhores desempenho com relação à taxa de erro de *bit*, assim como pode ser verificado na Figura 5.5, por exemplo. O desempenho obtido por este decodificador é mostrado na Figura 5.9.

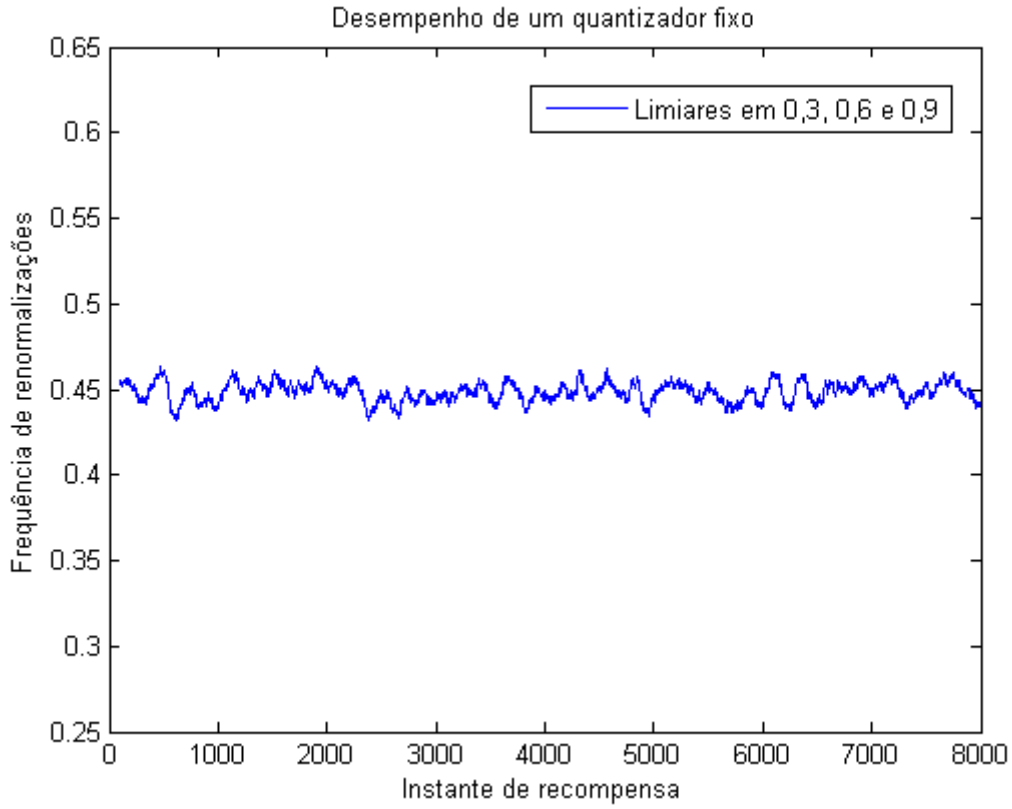


Figura 5.9. Desempenho de um decodificador com níveis de quantização fixos.

Comparando os desempenhos observados, certamente o decodificador controlado pelos agentes de aprendizado foram capazes de se adaptar ao canal, superando o desempenho de um decodificador fixo.

Quanto à taxa de erro de *bit*, após o término de algumas simulações, os limiares encontrados foram comparados com 64 combinações diferentes de níveis. Cada limiar poderia assumir 4 valores, variando de 0,15 a 0,60 em passos de 0,15, dando origem às 64 curvas mostradas na Figura 5.10, juntamente com a comparação com os limiares encontrados na tarefa de aprendizagem. Foram simulados 100.000 *bits* decodificados em blocos de 100.

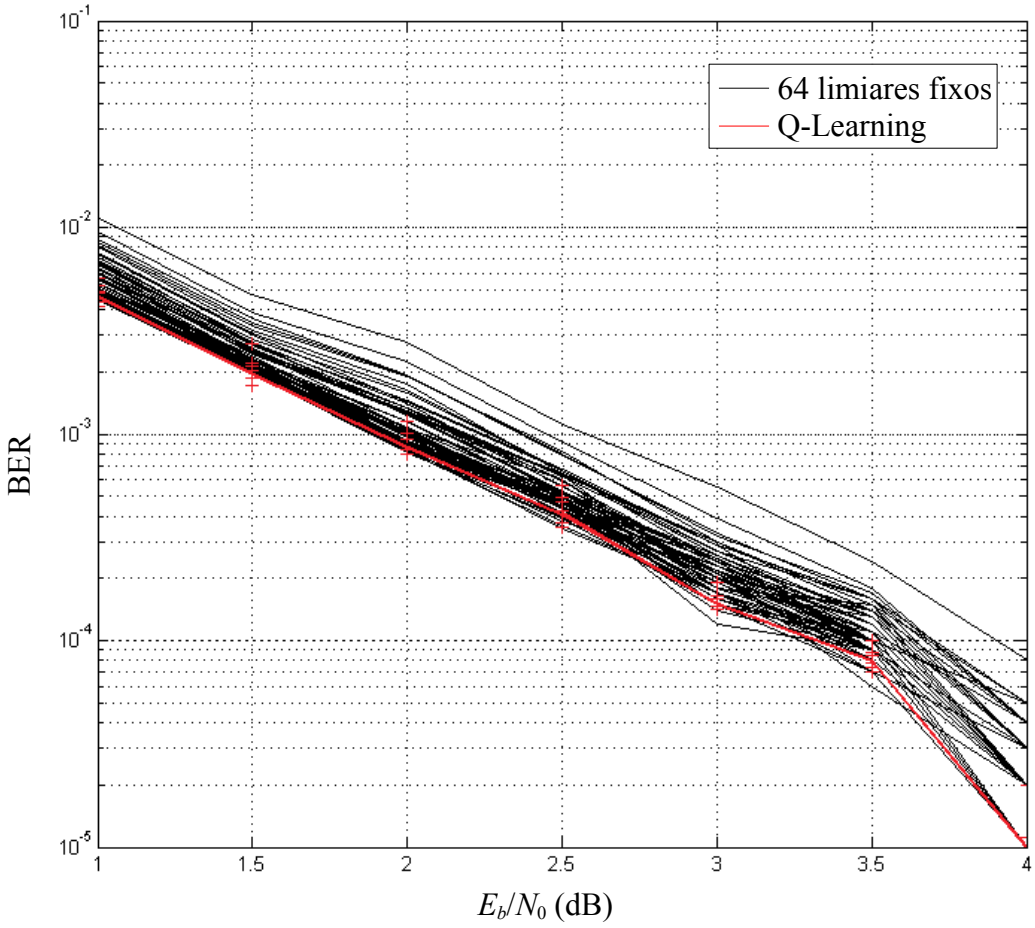


Figura 5.10. Comparação do desempenho do decodificador com Q-Learning e decodificadores de níveis fixos.

Pela figura, percebe-se que o desempenho obtido pelo decodificador utilizando aprendizado por reforço não é ótimo. Por outro lado, seu desempenho ficou acima da média das demais configurações, tendo em vista que sua taxa de erro de *bit* ficou abaixo da média. Além disso, não é um mesmo nível que garante sempre o melhor desempenho. Levando isto em consideração, o decodificador obteve um desempenho bastante estável. Outra limitação de desempenho foi que o agente não convergiu por completo na solução da tarefa, principalmente devido a limitações de tempo geradas pela função implementada.

É importante notar que o decodificador parte do completo desconhecimento do sistema e se adapta de maneira a tentar minimizar os erros de decodificação. Porém, devido a grande variação do ambiente, o agente tem dificuldades em encontrar rapidamente o comportamento ótimo no sentido de limiares que resultem na menor taxa de erro de *bit*. Em casos mais controlados, como os casos com relação sinal-ruído mais elevada, o desempenho do decodificador é praticamente ótimo, visto que apenas três dos 64 decodificadores apresentaram melhor desempenho na correção de erros.

A maior vantagem de se utilizar um decodificador que se adapta ao ambiente é que este não se limita às restrições do canal. Neste trabalho, devido a sua simplicidade, foi escolhido

um canal AWGN, mas nada impede o decodificador de ser aplicado em canais mais complexos, como, por exemplo, canais com desvanecimento e canais variantes no tempo.

5.8. CONCLUSÃO

Conclui-se neste capítulo a empregabilidade de técnicas de aprendizado por reforço na área de comunicações digitais. Os resultados obtidos pelo decodificador de Viterbi controlado dinamicamente se mostraram satisfatórios frente às dificuldade e limitações impostas pela modelagem do problema.

Novamente, percebe-se que a maior dificuldade em um problema de aprendizado por reforço consiste em encontrar a modelagem mais adequada para o sistema. No decorrer deste trabalho, como apresentado neste capítulo, diversos modelos tiveram que ser completamente descartados por falta de adequação ao problema até que um modelo capaz de resolver o problema fosse proposto. Ainda assim, não é possível concluir que o modelo apresentado é ótimo.

Portanto, a utilização de técnicas de aprendizado de máquina na área de comunicações digitais merece ser explorada com a devida atenção, sendo capaz de gerar tanto resultado quanto nas áreas de robótica, controle e automação.

6. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foram apresentadas bases da teoria de comunicações digitais juntamente com uma introdução ao aprendizado por reforço, uma das linhas do aprendizado de máquina. A partir desses conceitos, foi possível desenvolver uma estratégia de adaptação de um decodificador de Viterbi de decisão *soft* ao canal de comunicação.

Compreender a base das comunicações digitais é o primeiro passo para propor novas soluções nesta área. Como visto no decorrer do trabalho, conforme novas soluções aparecem, outras complicações são descobertas. Embora o ritmo de desenvolvimento das comunicações digitais tenha freado, sempre vão existir problemas em aberto. Um desses problemas se refere ao teorema de Shannon, a procura por códigos capazes de garantir a máxima capacidade de canal.

Conforme comentado no capítulo 3, as técnicas de codificação e decodificação de canal desempenham um papel fundamental nas comunicações digitais. Diversos sistemas são construídos baseados na taxa de erro de *bit* tolerada [22]. Com isso, uma forma de aumentar o desempenho de um sistema é por meio da aplicação de códigos corretores de erro capazes de garantir uma grande capacidade de correção de erros juntamente com altas taxas de código. Como visto, os códigos convolucionais atendem a tais requisitos. Diferentemente dos códigos de bloco, códigos convolucionais mantém uma relação entre cada novo código transmitido por meio de unidades de memória. Isso permite que mesmo códigos com taxas altas alcancem baixas taxas de erro de *bit*, porém, com a desvantagem de uma decodificação mais complicada.

Outra vantagem dos códigos convolucionais é a possibilidade da decodificação *soft*. Comparada com a decodificação *hard*, este meio de decodificação leva em consideração a informação contida no ruído adicionado à informação. Essa informação extra leva a um desempenho superior na correção de erros. O principal limitante de decodificadores de decisão *soft* é a necessidade de quantização da informação.

Para tentar contornar o problema de quantização, tipicamente fixa, foi proposta uma solução dinamicamente adaptativa por meio das técnicas de aprendizado por reforço. O aprendizado de máquina não-supervisionado baseado em sinais de reforço é capaz adaptar um agente ao ambiente para realizar tarefas com base em recompensas a ele fornecidas. O principal método visto neste trabalho, o algoritmo Q-Learning, é capaz de aprender a realizar tarefas dinamicamente, isto é, praticar ações e aprender o comportamento desejado no ambiente em que está inserido por meio delas.

Aplicando o aprendizado por reforço em um decodificador de canal por algoritmo de Viterbi de decisão *soft*, foi possível obter um decodificador capaz de controlar dinamicamente seus limiares de quantização para, assim, gerar ganhos no desempenho em face ao ruído presente no canal. A falta de controle completo sobre o ambiente, no caso, a aleatoriedade do ruído, compromete o desempenho do agente, impedindo-o de alcançar um resultado ótimo do ponto de vista de taxa de erro de *bit*.

A maior dificuldade em se lidar com aprendizado por reforço é a questão da modelagem do problema. Como visto no capítulo 5, vários modelos foram propostos e descartados até que fosse possível obter uma forma de controlar o agente. Ainda assim, o modelo realizado não é garantido como um modelo ótimo.

Portanto, neste trabalho fica verificada a possibilidade de utilização de aprendizado de máquina em comunicações digitais, com a consideração de que deve sempre existir uma modelagem adequada e satisfatória do problema. Vale lembrar que, embora o nome possa transmitir uma ideia diferente, o sucesso das técnicas de aprendizado de máquina dependem não da máquina, mas do esforço empenhado na modelagem do problema, pois, no fundo, o agente ainda é uma forma sistemática de programação.

6.1. SUGESTÕES DE TRABALHOS FUTUROS

Como visto nos resultados apresentados, uma série de limitações foram necessárias para a convergência dos resultados desejados. A primeira restrição diz respeito a variação discreta e limitada dos limiares de quantização. Neste trabalho, a granulação dos limiares foi proposta por observação do desempenho de outros decodificadores. Portanto, como primeira proposta de continuação, fica a utilização de técnicas de aprendizado por reforço também para a determinação da variação desses níveis, como valor mínimo, máximo e passo.

Além disso, existem técnicas de aprendizado por reforço baseadas em estados contínuos. A aplicação destas técnicas é um possível avanço na ideia de controle de níveis de quantização, permitindo até mesmo encontrar níveis ótimos de quantização.

Tanto para técnicas de estados discretos quanto contínuos, a aplicação de algoritmos mais sofisticados é uma ideia promissora no desenvolvimento de agentes mais capazes de solucionar problemas de aprendizado. A simples implementação de traçados de elegibilidade [27] é capaz de aumentar consideravelmente o desempenho do algoritmo Q-Learning, em técnicas conhecidas como TD(λ).

Por fim, a aplicação do aprendizado de máquina em comunicações digitais não deve se limitar à codificação em canais AWGN. A abordagem em outros modelos de canal mais

complexos, como canais variantes no tempo ou com desvanecimento, ou a aplicação em técnicas de equalização de canal, por exemplo, são também áreas a serem exploradas dentro dessa área das comunicações.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Amdocs Research: 98 Percent of Mobile Network Sessions Now Data. Disponível em: < http://www.amdocs.com/News/Pages/amdocs-research-mobile-data.aspx >. Acesso em 03/12/2014.

[2] SHANNON, C. E. A mathematical theory of communication. *Bell System Technical Journal*, vol. 27, p. 379-423 e 623-656, 1948

[3] MOON, T. K. *Error Correction Coding*: Mathematical Methods and Algorithms. New Jersey: John Wiley & Sons, Inc., 2005, 756 p.

[4] WU, Y.; ALSTON, M.; CHAU, P. Dynamic Adaptation of Quantization Thresholds for Soft Decision Viterbi Decoder with a Reinforcement Learning Neural Network. *Journal of VLSI Signal Processing*, v. 6, p. 77-84, 1993.

[5] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS – IEEE. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2012. Disponível em: < http://standards.ieee.org/getieee802/download/802.11-2012.pdf >. Acesso em 26/11/2014.

[6] VITERBI, A. J. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, v. 3, n. 2, p. 260-269, 1967.

[7] YASUDA, Y.; HIRATA, Y.; OGAWA, A. Optimum Soft Decision for Viterbi Decoding. *Proceedings of the 5th Int. Conf. of Digital Satellite Communications*, p. 251-158, 1981.

[8] MAHADEVAN, S.; CONNELL, J. Automatic programming of behavior-based robots using reinforcement learning. *Artificial Intelligence*, v. 55, p. 311-365, 1992.

[9] MATARIC, M. J. Reinforcement Learning in the Multi-Robot Domain. Autonomous Robots, v. 4, p. 73-83, 1997.

[10] SANS-MUNTADAS, A.; DUARTE, J. E.; REIKENSMEYER, D. J. Robot-assisted motor training: Assistance decreases exploration during reinforcement learning. *36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, p. 3516-3520, 2014.

[11] CRITES, R.; BARTO, A. Improving Elevator Performance Using Reinforcement Learning. *Advances in Neural Information Processing Systems*, v. 8, p. 1017-1023, 1996.

[12] DEISENROTH, M. P.; RASMUSSEN, C. E. *Efficient Reinforcement Learning for Motor Control.* University of Cambridge, 6 p.

[13] VVARSHAVSKAYA, P.; KAELBLING, L. P.; RUS, D. Automated Design of Adaptive Controllers for Modular Robots using Reinforcement Learning. MIT, 26 p.

[14] DUTREILH, X.; KIRGIZOV, S.; MELEKHOVA, O.; MALENFANT, J.; RIVIERRE, N.; TRUCK, I. Using Reinforcement Learning for Autonomic Resource Allocation in Clouds: Towards a Fully Automated Workflow. *ICAS 2011 : The Seventh International Conference on Autonomic and Autonomous System*, p. 67-74, 2011.

[15] LEITE, J. P. ; CARVALHO, P. H. P. ; VIEIRA, R. D. . Modulação e Codificação Adaptativa em

Sistemas OFDM Utilizando Aprendizado por Reforço. *XXX Simpósio Brasileiro de Telecomunicações, SBrT*, 2012.

[16] PRASHANTH, L. A.; CHATTERJEE, A.; BHATNAGAR, S. Adaptive sleep-wake control using reinforcement learning in sensor networks. *Sixth International Conference on Communication Systems and Networks (COMSNETS)*, p. 1-8, 2014.

[17] BALAKRISHNAN, H.; TERMAN, C. J.; VERGHESE, G. C. *Bits, Signals, and Packets*: An Introduction to Digital Communications & Networks. M.I.T. 6.02 Lecture Notes, 2012, 302 p.

[18] LATHI, B. P.; DING, Z. *Modern Digital and Analog Communication Systems*. 4. ed. New York: Oxford University Press, 2009, 1008 p.

[19] MADHOW, U. *Fundamentals of Digital Communication*. New York: Cambridge University Press, 2008, 499 p.

[20] GALLAGER, R. G. *Principles of Digital Communication*. New York: Cambridge University Press, 2008, 407 p.

[21] GRADDHTEYN, I. S.; RYZHIK, I. M. *Tables of Integrals, Series, and Products,* 6. ed. California: Academic Press, p. 1101-2000.

[22] FREEMAN, R. L. *Radio System Design for Telecommunications*. 3. ed. New Jersey: John Wiley & Sons, Inc., 2007, 880 p.

[23] COVER, T. M.; THOMAS, J. A. *Elements of Information Theory*. 2. ed. New York: John Wiley & Sons, Inc., 2006, 776 p.

[24] AURENHAMMER, F.; KLEIN, R. Voronoi Diagrams. *Handbook of Computational Geometry*. Amsterdam, Netherlands: North-Holland, p. 201-290, 2000.

[25] INFORMATION SCIENCES INSTITUTE, UNIVERSITY OF SOUTHERN CALIFORNIA. RFC 793, Transmission Control Protocol. Disponível em: http://www.ietf.org/rfc/rfc793.txt. Acesso em 22 de nov. De 2014.

[26] SHUNG C. et al. VLSI Architectures for Metric Normalization in the Viterbi Algorithm. *IEEE International Conference on Communications*, v. 4, p. 1723-1728, 1990.

[27] SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning*: An Introduction. Massachusetts: MIT Press, 1998, 322 p.

[28] MENEGAZ, M. *Aplicação da rede GTSOM para navegação de robôs móveis utilizando aprendizado por reforço*. 2009. 31 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, 2009.

[29] WASKOW, S. J. *Aprendizado por Reforço utilizando Tile Coding em Cenários Multiagente*. 2010. 45 f. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, 2010.

[30] WATKINS, C. J. C. H. *Learning from Delayed Rewards*. 1989. 117 f. Tese (Doutorado) – Cambridge University, 1989.

[31] WATKINS, C. J. C. H.; DAYA, P. Q-Learning. *Machine Learning*, v. 8, p. 279-292, 1992.

[32] SHR, K.; HUANG, Y. SNR Estimation Based on Metric Normalization Frequency in Viterbi Decoder. *IEEE Communication Letters*, v. 15, n. 6, p. 668-670, 2011.